THE OVERLAPPING GRID TECHNIQUE FOR THE TIME ACCURATE SIMULATION OF ROTORCRAFT FLOWS

Thorsten Schwarz

German Aerospace Center (DLR)
in the Helmholtz-Association
Institute for Aerodynamics and Flow Technology
Lilienthalplatz 7, D-38108 Braunschweig, Germany

Abstract

This paper presents the implementation of the overlapping grid technique into DLR's structured flow solver FLOWer. The hole cutting procedure and the search methods are described. Trilinear interpolation is used to transfer flow data between grids. For data interpolation close to solid surfaces, a projection algorithm is applied which corrects discretization effects. The approach used to prescribe rigid body motions for unsteady flow simulations allows to relate the motion of any body to any other body. Unsteady simulations also require to consider the hole motion during the definition of the interpolation points. For the integration of forces on grids with mesh overlap on body surfaces a unique surface is generated. Background grids are created with a Cartesian grid generator, which allows for cubic and anisotropic cells. Several examples of applications demonstrate the generality of the implementation of the Chimera technique.

Nomenclature

$\Delta a, \Delta b, \Delta c$	spacings of cuboidal cell
d_1, d_2	range of weighting function
$IBLANK$	Chimera tag array
$\vec{x}_1 \ldots \vec{x}_4$	vertex coordinates of tetrahedron
$\vec{x}_P$	coordinates of interpolation point
$\vec{x}_{P*}$	coordinates of virtual interpolation point
w	weighting function
$\vec{W}$	vector of conservative variables
$\Delta\vec{W}$	change of conservative variables per time step
$\Delta x, \Delta y, \Delta z$	spacings of Cartesian cell
$\gamma_1, \gamma_2, \gamma_3$	interpolation coefficients
α, β	auxiliary variables
$\vec{\varepsilon}$	projection vector
φ, Θ	spherical coordinates

Introduction

The numerical solution of the Navier-Stokes equations for helicopter type applications is a demanding task: First of all, the flow solver must be able to account for the moving parts of the helicopter, e.g. the main and tail rotors, see Figure 1. The second difficulty is to create the computational grids, particularly if block structured solvers are used and the configuration has a complex shape. Finally, the long run time of unsteady flow simulations requires efficient solution algorithms in order to minimize the execution time.

Figure 1: BO 105 Helicopter

One approach to reduce the effort for a flow simulation is to use the so called Chimera or overlapping grid approach (ref. [1]). This method allows to generate the computational meshes for the rotors and the fuselage independently. In a subsequent step, the grids are embedded into each other with an arbitrary overlap, see the 2D example in Figure 2. Grids associated with moving bodies move with the bodies without stretching or distorting the grids. If some grid points of one mesh fall within a solid body, these points are blanked and excluded from the flow computation. The blanked regions of the grid are often called hole. The communication among the grids is usually established by interpolation techniques.

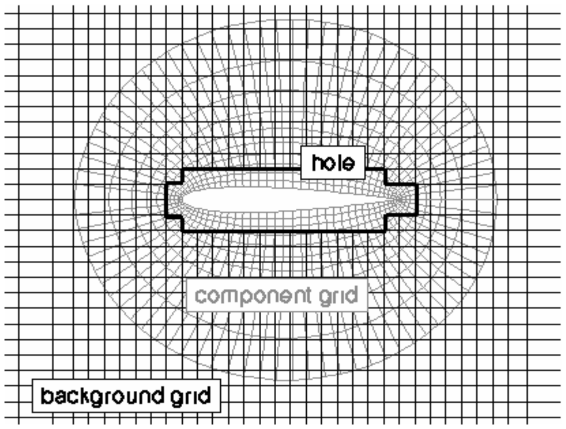


Figure 2: Overlapping grid system

The overlapping grid technique also allows to simplify the grid generation for complex geometries. This is achieved by breaking down the geometry into simple shaped components, for which individual grids can easily be created.

The Chimera technique has now been used for rotorcraft applications for some time. It has been implemented in many flow solvers used in industry, for example Beggar (ref. [17]), OVERFLOW (ref. [5]), CFD-Fastran (ref. [21]), elsA (ref. [8]) and FLOWer (ref. [11]). Best practice experiences have been summarized in ref. [6].

In the following, the Chimera capabilities of the block structured flow solver FLOWer will be described in detail. FLOWer is developed by the German research center DLR with contributions from universities and aerospace industry. The extension of FLOWer for helicopters like configuration has been supported during the last six years by the French-German cooperative project Complete Helicopter AdvaNced Computational Environment (CHANCE) (ref. [16]). In this context, the implementation of the Chimera technique in FLOWer was widely extended. Starting with a basic Chimera-scheme six years ago, FLOWer has now many capabilities to handle moving grids and to simplify grid generation. The method has been verified and validated for a wide range of applications and much experience in its use has been gained.

Flow solver

The flow solver FLOWer (ref. [11]) solves the Reynolds averaged Navier-Stokes equations with a second order accurate finite volume discretization on structured, multi-block grids. The fluxes can be computed either with a central scheme or with various upwind schemes. Turbulence is modeled by several 0-, 1-, 2- or 7-equation models, e.g. Spalart-Almaras, kω, kω-SST, EARSM or RSM. For steady flow simulations the time integration of the main equations is advanced with a five stage Runge-Kutta method with multigrid acceleration. For the turbulence equations, an implicit DDADI method is used. Unsteady flow simulations are performed with the implicit dual time stepping formulation. For high performance computing, FLOWer is parallelized based on MPI and is optimized for vector computers.

Chimera technique

Hole cutting

The first step in a Chimera calculation is the blanking of grid points, which are inside solid bodies. For this purpose, the user generates one or more auxiliary grids, which totally include the solid body, see Figure 3. Now all cell midpoints of the computational grids overlaying the solid body are checked, if they are inside the auxiliary mesh. Any cell being inside the auxiliary grid is flagged and is excluded from the flow computation.

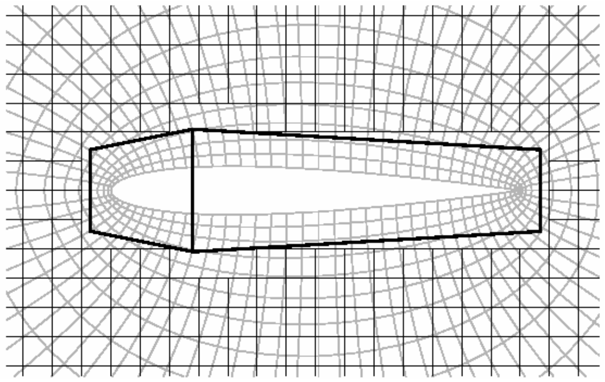


Figure 3: Auxiliary grid (thick lines) with two cells for hole cutting

No grid hierarchy is used to blank cells. Instead, any auxiliary grid can cut holes in any grid which does not contain the solid body enclosed by the auxiliary grid.

Although the presented hole cutting method is non automatic, the effort for the user is usually small, since the auxiliary grids can be coarse and they do not have to fulfill any quality requirements. The advantage of the chosen approach is its very simple implementation and its high execution speed. Furthermore it allows the user to control the exact geometry of the hole. An example of a set of auxiliary grids for a complex configuration is shown in Figure 4.

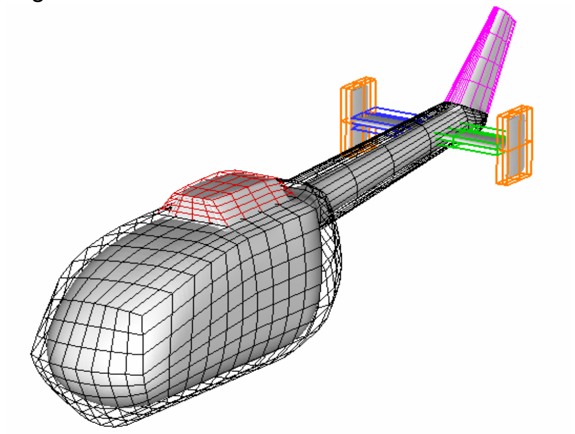


Figure 4: Set of seven auxiliary grids enclosing a helicopter fuselage

Search and interpolation

Communication between the overlapping grids is established by interpolating data from overlapping grids. In order to set up the interpolation, at first the interpolation cells have to be identified. This is accomplished by checking for each cell, if the spatial discretization operator accesses cells inside holes or non existent cells at the outer grid boundary. In FLOWer two layers of interpolation cells are required at outer grid boundaries and at the hole fringe, see Figure 5.

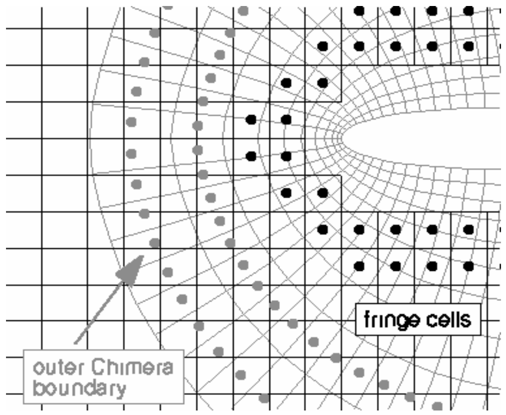


Figure 5: Interpolation points

Now a Chimera tag array *IBLANK* is set up, which indicates the status of a cell. It is:

IBLANK = 0 : hole cell or interpolated cell (excluded from flow computation)

IBLANK = 1 : valid cell

Following the definition of the Chimera tag array, a search is started to find appropriate donor cells for the interpolation cells. Since the flow variables are given at the centers of the cells, the search must be performed for the dual meshes that connect the coordinates of the cell centers. In FLOWer the search on general type meshes is performed with an Alternating Digital Tree method (ADT) (ref. [2]). The ADT method uses a tree like data structure to store the boxes given by the minimum and maximum coordinates of the dual grid cells. During the search process, all boxes enclosing the interpolation point are identified with $\log_2(N)$ operations, where N is the total number of grid points. In a subsequent step, each cell associated with a retrieved box is checked if it includes the interpolation point. To this end, the cell is subdivided into six tetrahedra and the linear system of equations

$$\gamma_1(\vec{x}_2-\vec{x}_1)+\gamma_2(\vec{x}_3-\vec{x}_1)+\gamma_3(\vec{x}_4-\vec{x}_1)$$
$$=\vec{x}_P-\vec{x}_1$$

is solved for the unknown coefficients $\gamma_1,\gamma_2,\gamma_3$, where $\vec{x}_1\ldots\vec{x}_4$ are the vertex coordinates of the tetrahedron and $\vec{x}_P$ denotes the coordinates of the interpolation point. If the conditions

$$\gamma_1,\gamma_2,\gamma_3\geq 0 \quad , \quad \gamma_1+\gamma_2+\gamma_3\leq 1$$

are true, then the interpolation point is inside the tetrahedron. Additionally it is tested, if the cells at the vertices of the tetrahedron are not blanked (*IBLANK=1*). If all checks are successful, the cell is a valid donor cell and the flow values at the vertices of the tetrahedron are used for the trilinear interpolation of flow data with the interpolation coefficients $\gamma_1,\gamma_2,\gamma_3$.

A Chimera grid system often consists of more than two overlapping grids. In this case, FLOWer searches all grids for donor cells without preferring one grid to another. If more than one cell including the interpolation point is found, the cell with the smallest volume is chosen to be the donor cell. This choice is based on the assumption, that the discretization error of the Navier-Stokes equations is smallest on the finest grid and thus the interpolated flow data will be most accurate.

In many applications the ADT search algorithm and the interpolation method have been proven to be very robust even for grids of bad quality.

Background grids for Chimera computations are often Cartesian with an equidistant or non equidistant distribution of grid points. For these types of meshes FLOWer offers a specialized search procedure which is much faster than the ADT method. While for equidistant meshes the *i,j,k*-indices of the donor cell can be directly computed based on the cell spacing, for non equidistant grids the indices are determined by applying a bisection method in one index-direction after the other.

Sometimes the computation of interpolation coefficients is not possible, since the grid overlap is not sufficient or because parts of the overlapping grids are blanked. This situation requires to improve the grids. But for large grid systems with many overlapping grids around a complicated configuration the optimization of the grids may be very time consuming. Therefore a method is available in FLOWer to enable a flow computation even if a non sufficient overlap exists: After the computation of interpolation coefficients has failed for a cell, all overlapping grids are searched for the cell with the smallest distance to the interpolation point. The flow data of this cell are subsequently used to update the interpolation cell data during the flow simulation. The described method reduces the accuracy of the flow solution locally. Thus the number of cells which requires its application should be kept to a minimum.

For unsteady flow simulations including rigid body motion an extension of the hole cutting procedure is required to avoid an invalid update of flow data close to holes, see section “Grids in relative motion”.

Flow simulation

Little extension to the flow solver is necessary to enable computations on overlapping grids. In the case of a purely explicit solver, any spatial or temporal discretization operator remains unchanged. Only the update of the conservative variables $\vec{W}$ in the blanked regions must be suppressed by multiplying the change of the conservative variables per time step $\Delta\vec{W}$ with the tag array *IBLANK*

$$\vec{W}^{t+1} = \vec{W}^{t} + IBLANK * \Delta\vec{W} \quad .$$

The adaptation of the implicit residual smoothing is accomplished by multiplying the residual before and after the smoothing with *IBLANK*. The implicit treatment of the turbulence equation with the DDADI algorithm is adjusted accordingly by multiplying the intermediate and final results with *IBLANK*.

For the modifications of the multigrid method the following approach has been chosen. During the hole cutting procedure, all cells of coarse meshes are flagged, if the restriction or prolongation operators access cells inside the holes on the next finer grid. Interpolation of flow data from an overlapping grid is only performed on the finest grid, whereas for coarser grids the data at hole fringes and outer grid boundaries are interpolated from the next finer level of the same grid. The *IBLANK* values are set analogous to the finest grid, where *IBLANK=0* indicates hole cells or interpolated cells. During the flow simulation the forcing function and the residuals computed on the coarse grid levels are multiplied with *IBLANK* in order to avoid an update of the coarse grid data. A modification of the fine grid data is prevented by multiplying the coarse grid corrections before and after the prolongation with the *IBLANK* values of the coarse and fine grid respectively.

Other possibilities to adapt the multigrid technique for Chimera computations have been published in ref. [9]. The method implemented in FLOWer has been chosen, because the author believes it to be the most robust one.

Grid overlap on body surfaces

In many cases the Chimera grids overlap on body surfaces. One example is presented in Figure 6 showing the junction of a wing and a body, where the wing and the body are discretized with different grids.

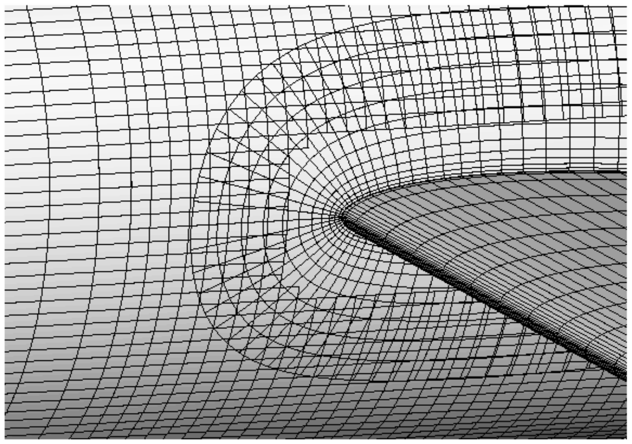


Figure 6: Grids with overlap on body surface

A grid overlap on body surfaces requires a modification in the computation of interpolation coefficients, since the standard interpolation described in section “Search and interpolation” may yield inaccurate results. The reason is sketched in

Figure 7. Due to the different discretization of a curved analytic surface in the two overlapping grids, the distance of a point P to the discrete surface representations is unequal, $\delta_1 \neq \delta_2$. Therefore all interpolation points close to a curved surface will interpolate flow data from locations with different wall distance than expected. This may for example cause problems when transferring data within boundary layers. Sometimes an interpolation point may even be outside of the overlapping grid making an interpolation impossible.

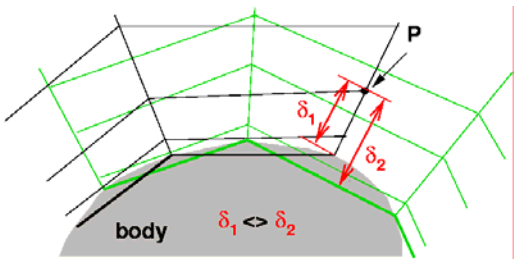


Figure 7: Non unique wall distance in overlap region for point P

In order to circumvent the described problem a correction method was developed. The procedure starts by searching the grid containing the interpolation point P for the point P_S on the surface, which is closest to point P, see Figure 8, top. The point P_S is projected onto the discrete surface representation of the overlapping grid. The projection vector is denoted with $\vec{\varepsilon}$.

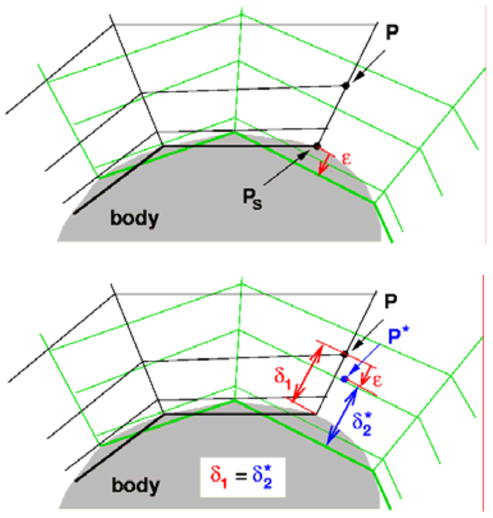


Figure 8: Correction of computation of interpolation coefficients, top: projection of wall point P_w on surface of other grid, bottom: introduction of virtual interpolation point P*

The projection vector is now multiplied with a scaling function and added to the coordinates of the interpolation point $\vec{x}_P$ giving the coordinates $\vec{x}_{P*}$ of a virtual interpolation point P*

$$\vec{x}_{P*} = w \cdot \vec{\varepsilon} + \vec{x}_P \ ,$$

see Figure 8, bottom. The weighting function is used to apply a full correction close to the surface and to reduce the correction with increasing distance from the surface

$$w = \begin{cases} 1 & if \quad 0 \leq |\vec{x}_P - \vec{x}_{P*}| < d_1 \\ \dfrac{d_2 - |\vec{x}_P - \vec{x}_{P*}|}{d_2 - d_1} & if \quad d_1 \leq |\vec{x}_P - \vec{x}_{P*}| < d_2 \\ 0 & if \quad d_2 \leq |\vec{x}_P - \vec{x}_{P*}| \end{cases} ,$$

where

$$d_1 = 10 \cdot |\vec{\varepsilon}| \quad , \quad d_2 = 30 \cdot |\vec{\varepsilon}|$$

gives good results. The coordinates of the virtual interpolation point are now used to search a donor cell and to compute the interpolation coefficients. The presented algorithm ensures an identical wall distance of the interpolation point and the interpolated data, $\delta_1 = \delta_2^*$, see Figure 8, bottom.

If the interpolation point P is overlapped by more than one grid, the correction method is to be applied for each donor grid, since in general the projection vector $\vec{\varepsilon}$ will be different.

Slightly different correction methods are known from literature, see ref. [19] and ref. [6]. But instead of introducing virtual interpolation points, the algorithms temporarily modify the grid coordinates before the computation of interpolation coefficients. As a consequence, a correct calculation of the interpolation coefficients is not possible if more than two grids overlap on the body surface, because the coordinates can only be adjusted to match with one overlapping grid. This difficulty is avoided by the introduction of the virtual target point, since its coordinates are computed for each donor grid independently.

Grids in relative motion

The overlapping grid technique allows to move the grids relatively to each other without affecting the grid's quality. It is therefore an appropriate method for the simulation of bodies in relative motion. This requires to update the intergrid communication each time the grids have been shifted to a new position. Therefore, the hole cutting procedure and the calculation of interpolation coefficients must be executed repeatedly.

Unsteady flow simulations require an extension of the method used to identify interpolation points. Whereas in section “Search and interpolation” only the spatial discretization operator was checked if it accesses hole cells, for unsteady flow computations also the temporal discretization operator must be taken into account. Otherwise the correct calculation of the flow data cannot be guaranteed in case of holes with varying positions. Such a situation is depicted in Figure 9, top, showing a one dimensional grid with a hole moving two cells to the right at each time step. The spatial discretization operator is assumed to have a width of five cells. Thus two layers of interpolation cells are necessary at the hole fringe. If now a time discretization operator is used, which requires valid data on two previous time steps, a valid update of some cells in the ‘wake’ of the hole is not possible.

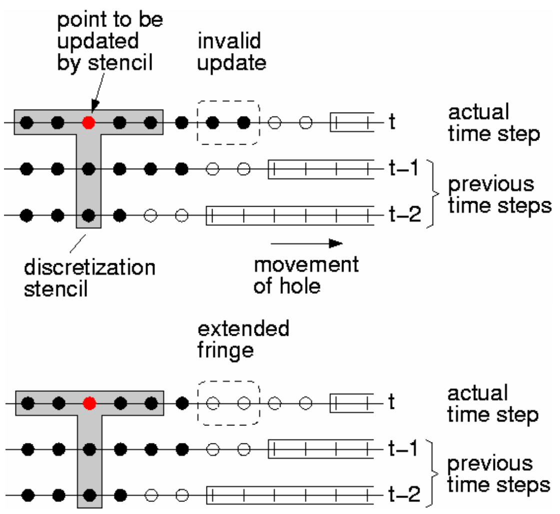


Figure 9: One dimensional grid with a moving hole, top: invalid update of flow data, bottom: extension of hole fringe enables correct flow simulation,
● : cell with *IBLANK* = 1, ○ : interpolated cell, ⌗ : hole cell

In order to solve this problem, all cells whose time discretization operator accesses hole cells are added to the list of interpolation cells. This results in an extended hole fringe in the wake of the hole, see Figure 9, bottom.

The motion of a rigid body is either defined based on the inertial frame of reference or relatively to another moving geometry. For example, the fuselage of a helicopter moves relatively to the inertial frame, whereas the rotational velocity of the rotor shaft is constant relative to the fuselage while the flapping of a blade can best be defined relatively to the rotor hub. According to these relations, a tree like data structure is used in FLOWer to specify rigid body motions, see Figure 10. This allows to define a complex body movement by a series of simple transformations. In FLOWer each transformation may include a translation in any direction and a rotation around an arbitrary axis. The time dependent translation and rotation are specified by polynomial and Fourier series.

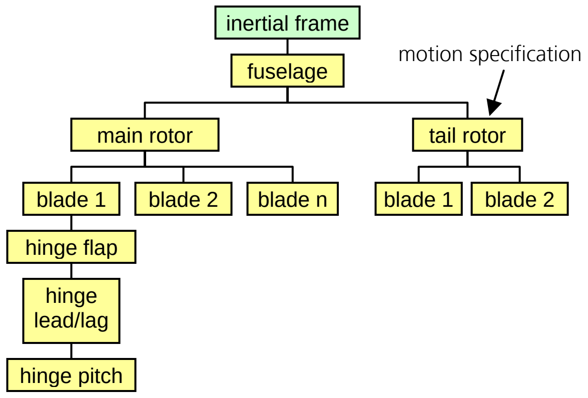


Figure 10: Tree like data structure for the definition of rigid body motions for helicopter

After the user has defined the motion relations each of the overlapping grids is linked to any of the specified transformations. During a subsequent flow computation the transformations are used to position the grids in space and to transform vectorial quantities when they are transferred between the grids.

Calculation of global forces

The standard procedure to calculate global forces like lift and drag is to sum up the pressure and friction forces of all cells on the body surface. This approach fails in case of a grid overlap on body surfaces, since the forces in the area of the grid overlap are counted twice. Therefore a method is to be used, which generates a unique surface before integrating the forces. In FLOWer this is accomplished by a postprocessing tool which removes the overlap and fills the resulting gap with triangles, see Figure 11.

The procedure can be subdivided into the following steps. At first, a criterion has to be defined which one of two overlapping cells has to be removed. To this end, each cell is assigned a priority. Cells at Chimera grid boundaries and at hole fringes are set to priority one. This is the lowest priority. All other cells get a priority, that is increased by one compared to their neighboring cells. Therefore the priority of the cells is increasing with increasing distance to Chimera boundaries. In the next step, each cell of the surface grid is checked on overlaps with another cell. If this is true, the cell with the lower priority is removed. After the execution of this step for all cells, the overlap of the surface grid is removed. Due to the priority tagging, the resulting gap is placed approximately in the middle between the Chimera grid and hole boundaries.

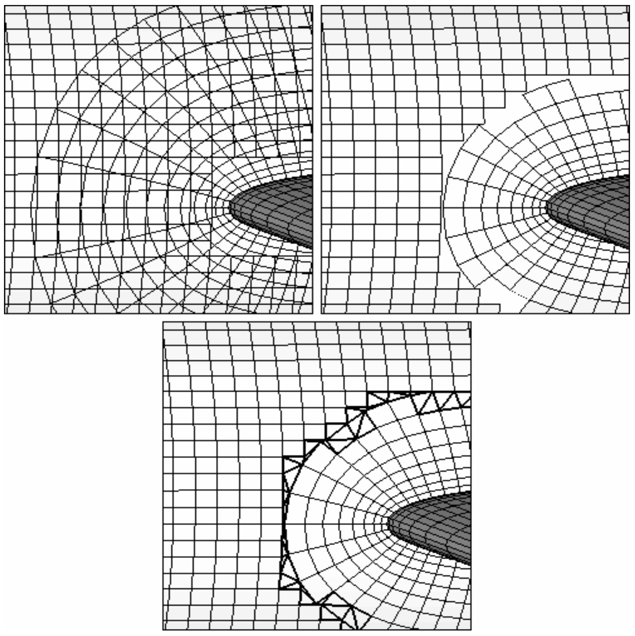


Figure 11: Generation of a unique surface for the calculation of global forces

In the last step the gap is filled with triangles by using a Delauny triangulation method. In contrast to other triangulation methods (e.g. ref. [4]), that always connect points of one side of the gap with points of the other side, the Delauny triangulation can build a triangle with any points at the gap border. Thus a special treatment at branches of the gap is not required. Additionally, the quality of the triangulation with respect to triangle distorsion and stretching is better.

Cartesian background grids

The component grids around a configuration extend often only a short distance into the computational domain. They must therefore be embedded into a background grid. While its manual creation is time consuming, it may instead be created automatically. The approach followed with FLOWer is to place one layer of fine Cartesian grids around the component grids and to iteratively wrap them with coarser Cartesian meshes. This procedure is repeated until the complete computational domain is covered by the background grid, see Figure 12.

The different cell size of neighboring blocks results in patched grid interfaces with hanging grid nodes. This type of background grid has already been presented in ref. [12]. Instead of using grids with patched interfaces, the Cartesian grid blocks may also overlap each other, see ref. [3]. But this approach does not ensure the flux conservation inside the background grid, which may result in higher numerical errors in the computed flow. Therefore the first-mentioned approach is used.

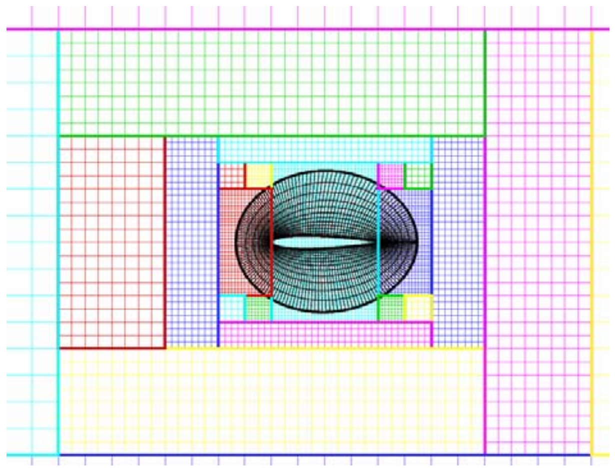


Figure 12: Cartesian background grid

In addition to the methods known from literature, the Cartesian grid generator used in this work does not only allow to create grids with cubic cells but also with anisotropic cells, see Figure 13. This increases the similarity of the cell geometries of the component and background grids. Hence interpolation errors are reduced. In addition, anisotropic cells may also help to ensure a sufficient overlap between the grids if the component grid extends only little into the far field direction.

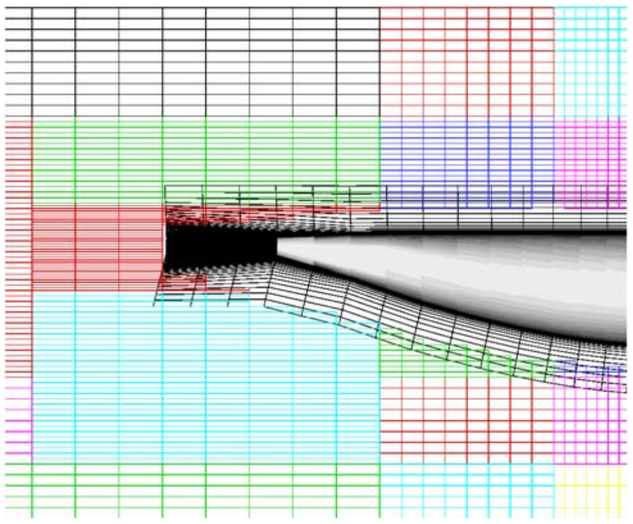


Figure 13: Cartesian background grid with anisotropic cells

A description of the numerical treatment of the hanging grid nodes in FLOWer can be found in ref. [20].

Generation of Cartesian grids

The creation of the Cartesian grid starts with a very coarse mesh of $n \times n \times n$ cells, which covers the entire computational domain. n is usually taken to be 4, 8 or 12. The initial grid is iteratively refined and subdivided into subblocks of size $n \times n \times n$, until the grid blocks have approximately the same cell spacing as the overlapped cells of the component grid. As long as a grid is partitioned into eight finer grids, the grid cells are cubic. A subdivision of a grid into one or two index directions yields a grid with anisotropic cells. The number of refinements and the refinement directions are calculated with a sensor function that analyses the geometry of every cell of the component grid. Details of the sensor function will be reported in the subsequent paragraph “Adaptation sensor”.

During the first adaptation cycle, the grids are refined without considering the cell spacing of neighboring blocks. Therefore the grids are refined further until the refinement is not larger than by a factor of two. In order to limit the use of grid blocks with anisotropic cells, the index direction with the largest spacing is coarsened only, if the cell is already cubic.

Some blocks of the background grid may be entirely inside the auxiliary grids used for the hole cutting. For these grids all cells would be blanked in a subsequent flow computation. They can therefore be removed from the grid without affecting the flow solution. This minimizes the number of grid cells.

After the adaptation process is finished the grid consists of a large number of grid blocks of size $n \times n \times n$ cells. The number of grid blocks is now reduced by merging grid blocks with the same cell spacing. To this end the method of the weakest descent (ref. [18]) is used. The algorithm tries to maximize the number of merging steps in order to minimize the number of the resulting grid blocks. At the end of this step, the grid coordinates and the information on the patched grid interfaces are written to disk.

During the grid generation procedure the grid data are stored in an Alternating Digital Tree (ADT) data structure (ref. [2]). This method allows to keep grid blocks with an arbitrary number of cells where the cells can be either cubic or anisotropic. In the tree, the minimum coordinates of the grid blocks, the number of cells in each index direction and the refinement level in each index direction are saved. Therefore only nine variables are stored per grid block resulting in small memory consumption. The actual grid coordinates and the information on the grid boundary conditions are only computed during the final data output.

Adaptation sensor

The adaptation of the background grid to the component grids requires to use a sensor function which calculates for each of the arbitrarily shaped cells the dimensions of a similar Cartesian grid cell. The sensor used here requires three steps. At first each cell is transformed into a parallelepiped, where the edges of the parallelepiped are set identical to the length and direction of the lines, which connect the mid points of opposite faces of the cell, see Figure 14a, b. The parallelepiped is next transformed into a cuboid. This is accomplished by shifting the longest edges parallel to each other, until they are perpendicular to the second longest edges. The largest cell faces are now shifted parallel to each other until all edges are perpendicular to each other, see Figure 14c. Both operations do not change the volume of the parallelepiped (Cavalieri’s principle).

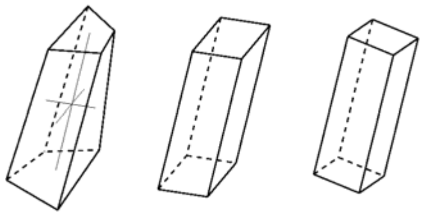


Figure 14: Transformation of a hexahedral (a) to a parallelepiped (b) and to a cuboid (c)

For the third transformation, a local coordinate system *(a,b,c)* is introduced, which is aligned with three edges of the cuboid. The length of the edges will be denoted by Δa, Δb, Δc. The origin of the coordinate system is now shifted to the origin of the inertial coordinate system *(x,y,z)*. Then, the axis directions of the inertial frame are calculated in spherical coordinates of the local frame. This gives the angles (φ_x,Θ_x), (φ_y,Θ_y), (φ_z,Θ_z), see Figure 15.

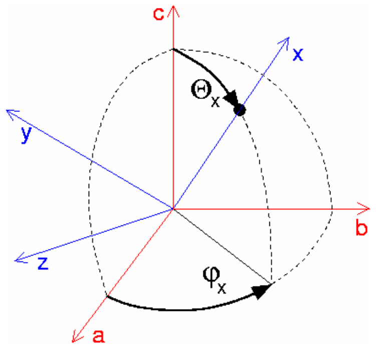


Figure 15: Transformation relations

The angles are used to compute the sensor function:

$$\Delta x = \Delta a^{\alpha_x(1-\beta_x)} \cdot \Delta b^{(1-\alpha_x)(1-\beta_x)} \cdot \Delta c^{\beta_x}$$
$$\text{with } \alpha_x = \cos^2(\varphi_x)\ ,\ \beta_x = \cos^2(\Theta_x)\ ,$$
$$\Delta y = \Delta a^{\alpha_y(1-\beta_y)} \cdot \Delta b^{(1-\alpha_y)(1-\beta_y)} \cdot \Delta c^{\beta_y}$$
$$\text{with } \alpha_y = \cos^2(\varphi_y)\ ,\ \beta_y = \cos^2(\Theta_y)\ ,$$
$$\Delta z = \Delta a^{\alpha_z(1-\beta_z)} \cdot \Delta b^{(1-\alpha_z)(1-\beta_z)} \cdot \Delta c^{\beta_z}$$
$$\text{with } \alpha_z = \cos^2(\varphi_z)\ ,\ \beta_z = \cos^2(\Theta_z)\ ,$$

where Δx, Δy, Δz are the spacings of the Cartesian cell in x-, y-, z-coordinate direction.

The sensor function has been chosen for three reasons: 1) the volume of the original cuboid is preserved, 2) the spacings of the Cartesian cell are identical to those of the cuboid, if the inertial and the local frame are identical, 3) if one axis of the local and the inertial frame is identical, the spacing in this direction is unchanged.

By using the adaptation sensor outlined above, the Cartesian background grid will have the same grid resolution as the underlying component grid and a similar cell stretching. In Figure 1, the shape of the computed Cartesian cell is plotted for a rotated cuboid.

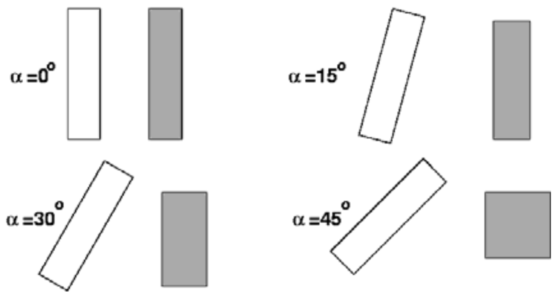


Figure 16: Shape of Cartesian cell (grey) for rotated cuboidal cell with stretching 1:4

Examples of applications

In the framework of the CHANCE project, the Chimera technique has been validated for various types of helicopter applications. Three test cases will be exemplified in the following.

Isolated rotor in forward flight

In ref. [15] the aerodynamics of the ONERA 7A rotor in forward flight including elastic blade deformation and trim has been investigated by embedding individual meshes for the blades into a background grid (3.2 million grid points in the whole grid system), see Figure 17. In Figure 18 the distribution of the normal force and the pitching moment close to the blade tip computed with two different methods is shown in comparison with experimental data. The first calculation (S4 no coupl.) was performed with the DLR rotor simulation code S4. In S4 the aerodynamics of the rotor is calculated by using the blade element theory based on measured airfoil tables including unsteady and Mach effects, a dynamic stall model, varying velocity effects and a prescribed wake model. For the second simulation (FLOWer/S4) the FLOWer code was applied. In both calculations the elastic blade deformation is simulated with the structural dynamics module of the S4 code. The comparison of the trimmed simulations presented in Figure 18 shows the significantly improved results when using the Navier-Stokes solver.

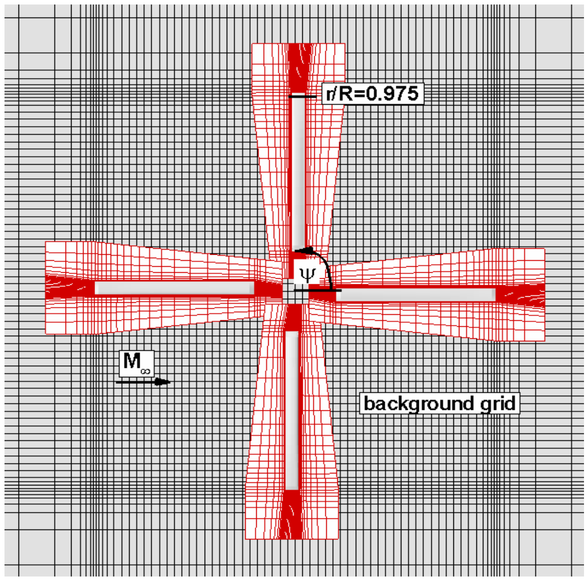


Figure 17: Overlapping grids for isolated rotor in forward flight including elastic blade deformation (ref. [15], with permission)

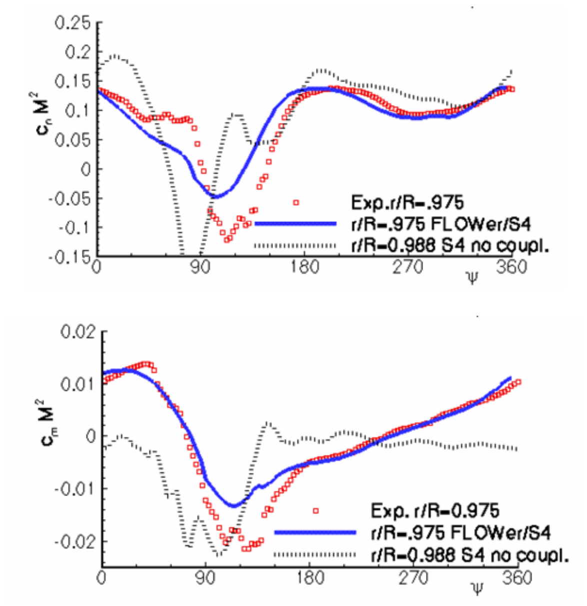


Figure 18: Normal forces (top) and pitching moment (bottom) for one revolution of 7A main rotor (ref. [15], with permission)

Actuator disc modelling

The time averaged effects of the main and tail rotor on the fuselage of an EC 145 helicopter have been analyzed in ref. [12]. This was achieved by embedding grids for the actuator discs of main and tail rotor into an existing mesh for a helicopter fuselage, see Figure 19. In Figure 20 the computed surface pressure distribution and the surface friction lines are shown. The large flow separation at the boot of the fuselage is clearly captured.

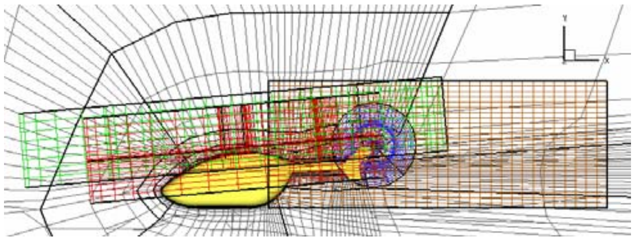


Figure 19: Chimera grid for EC 145 helicopter fuselage with actuator discs for main and tail rotor, every fourth grid line plotted (ref. [12], with permission)

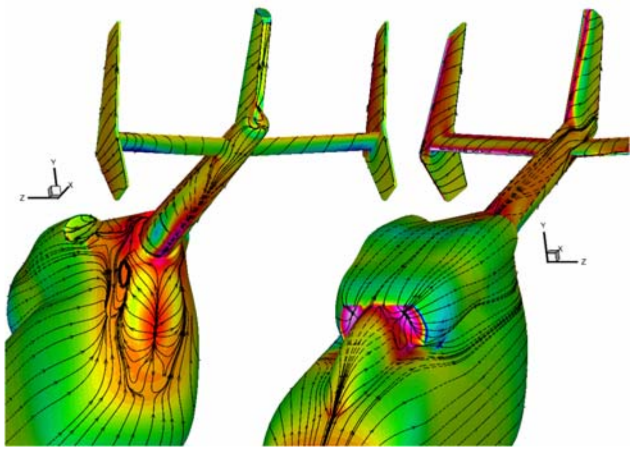
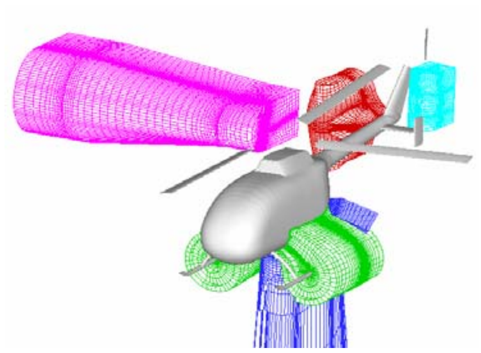


Figure 20: Surface pressure distribution and surface friction lines on fuselage of EC 145 helicopter (ref. [12], with permission)

Complete helicopter

The simulation of an almost complete helicopter configuration is reported in ref. [10], where the unsteady flow around a BO 105 wind tunnel model including main and tail rotor, skids and wind tunnel support strut was computed. The authors generated twelve overlapping grids for the components of the configuration and embedded the component grids into an automatically created Cartesian background grid, see Figure 21. One result obtained during the simulation is the pressure distribution at the symmetry plane of the fuselage, see Figure 22. The agreement between experimental and computational results is good. Unsteady pressure distributions on the fin for a half revolution of the main rotor or 2.5

revolutions of the tail rotor, respectively, are plotted in Figure 23. Although some offset can be observed in the c_p-values, the unsteady variations of the pressure are well captured.



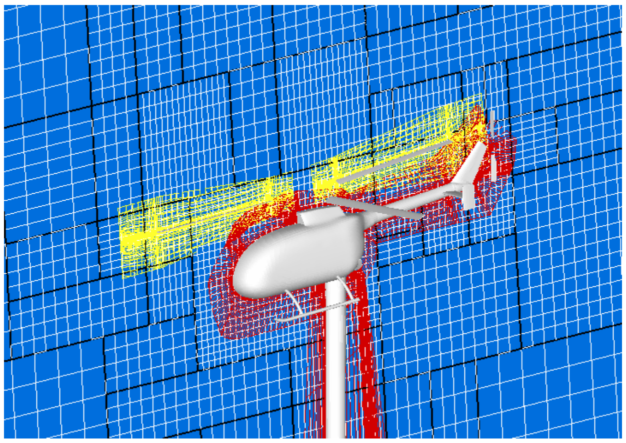


Figure 21: Chimera grid system for the time accurate simulation of the flow around a BO 105 wind tunnel model, every second grid line plotted, top: component grids without mesh for fuselage, bottom: cut at symmetry plane (ref. [10])

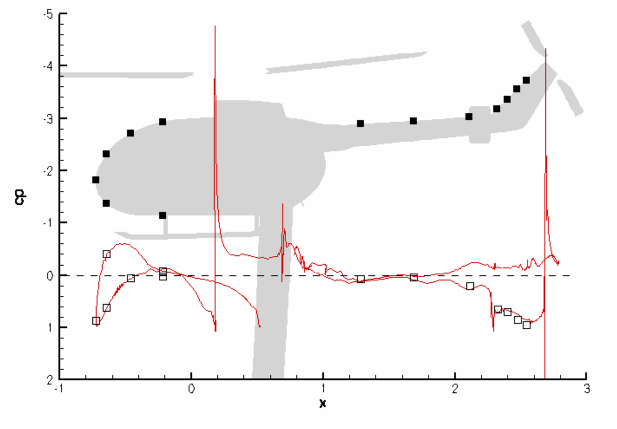


Figure 22: computed pressure distribution compared to experimental data in symmetry plane for unsteady simulation of BO 105 (ref. [10])

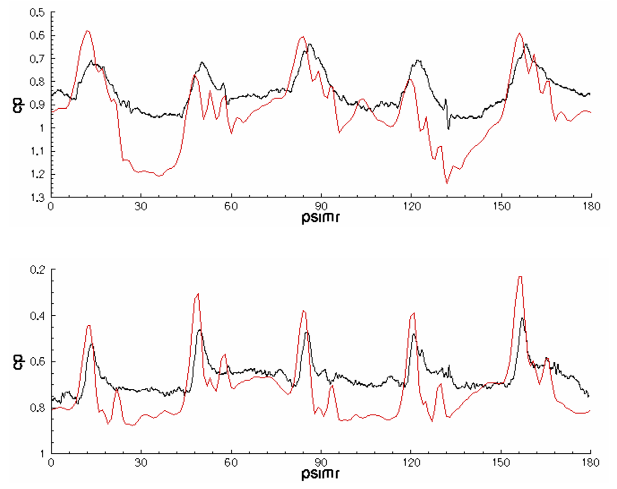


Figure 23: unsteady pressure distribution on tail fin at 50% radius (top) and at the outer radius (bottom) of the tail rotor of BO 105 helicopter, red: CFD, black: experiment (ref. [10])

Applications by industry

The Chimera technique in FLOWer has also been successfully applied by industrial customers.

Eurocopter Germany (ECD) has investigated the fuselage aerodynamics of the EC145 helicopter including the time averaged influence of the main rotor by embedding a separate grid for the actuator disc into the fuselage grid (ref. 13). The grid system was used to compute polars for the helicopter.

Airbus Germany has used the Chimera technique to embed individual grids for ailerons and spoilers into an existing grid for a wing-body airplane configuration (ref. [14]). The computational results showed good agreement with wind tunnel experiments.

Performance

Unsteady flow simulations are often very time consuming. Therefore efficient solution algorithms and their parallelized and vectorized implementation are required to minimize the execution time.

In order to demonstrate the performance of FLOWer, the time needed for one physical time step of a flow simulation was measured. The test case chosen was the configuration presented in section "Complete helicopter". This test case consists of 11.8 million grid cells and 480 grid blocks. The time integration was performed with the dual time stepping method which required 50 iterations of the flow solver to converge the flow equations at each physical time step. Convergence was accelerated by implicit residual smoothing and three levels multigrid. Turbulence was modeled with the kω-turbulence model. The time consumption of the FLOWer flow solver on a NEC SX8 vector computer and for a PC-Cluster with INTEL Xeon Processors with 3.06 GHz is presented in the following table:

	No. of procs	execution time hole cutting & search	execution time flow solver
NEC SX8	1	176.7 s	50 * 30.98 s
	2	100.7 s	50 * 16.44 s
	4	66.9 s	50 * 9.03 s
	8	47.6 s	50 * 5.47 s
PC	8	55.0 s	50 * 72.6 s

A flow simulation with less then eight processors on the PC Cluster was not possible due to memory limitations. The total time consumption for one physical time step is the sum of the time needed for the hole cutting and search procedures at the beginning of a physical time step and the time required for 50 iterations of the flow solver to converge the flow equations. According to the table the Chimera algorithms require on the vector computer less then 15% of the total CPU time and on the scalar computer less than 2%. This shows that the Chimera routines have only a minor impact on the total CPU time. The relatively high time consumption of the Chimera algorithms on the NEC SX8 is due to the ADT search method, which is not vectorized because of its recursive algorithm.

The speed up

$$\text{speed up} = \frac{\text{time consumption one processor}}{\text{time consumption n processors}}$$

for the NEC SX8 is shown in the following table:

	No. of procs	speed up hole cutting & search	speed up flow solver
NEC SX8	1	1	1
	2	1.8	1.9
	4	2.6	3.4
	8	3.7	5.7

The theoretical speed up of 8.0 for a computation on eight processors is not reached by the Chimera algorithms. This is due to the load balancing algorithm which is optimized for the flow solver and does not take into account the time consumption of the hole cutting and search procedures.
A good speed up is obtained for the flow solver up to four processors. With increasing number of processors, the time needed to solve the flow equations is reduced whereas the time needed for interprocessor communication is almost constant. This explains the non optimum speed up when using eight processors of the NEC SX8.

Summary

In this paper the implementation of the Chimera technique in DLR's structured flow solver FLOWer is presented. The hole cutting algorithm uses auxiliary grids to blank all cells which are in its interior. An ADT-search algorithm or a specialized method for Cartesian grids is used to find appropriate donor cells for the trilinear interpolation of flow data. If the grids overlap on a body surface, virtual interpolation points are introduced. They enable an accurate calculation of interpolation coefficients near surfaces despite the different surface discretizations. No grid hierarchy is used during hole cutting and data interpolation. Instead any solid body can cut holes in any grid and all meshes are searched for donor cells. For unsteady flow simulations the motion of grids are defined with a hierarchical data structure. This allows to define complex motions by a sequence of simple transformations. A valid update of flow data on moving grids is ensured by interpolating data for all cells, for which the discretization operator accesses hole cells either in spatial or in temporal direction.

Chimera flow simulations in general do not require a specialized postprocessing. One exception is the calculation of forces and moments which must be adapted if a grid overlap exists on body surfaces. In order to create a unique surface for the integration of forces and moments a tool was developed which removes the grid overlap and fills the resulting gap with triangles.

The decomposition of the computational domain into several independently created grids offers the possibility to create the background mesh automatically. To this end a Cartesian mesh generator is used which adapts an initially very coarse mesh to the cell size of the component grids. The resulting mesh has cubic and anisotropic cells which minimizes interpolation errors and may reduce the required overlap width. The spacings of the Cartesian cells are computed with a novel adaptation sensor.

The implementation of the Chimera technique in FLOWer has been validated for several helicopter applications. It has been shown that overlapping grids can be used to simplify grid generation and to simulate the flow around bodies in relative motion.

Performance measurements on a PC-Cluster and a parallel vector computer show that the hole cutting and search procedures have a minor impact on the total CPU time consumption.

References

[1] BENEK, J.; STEGER, J. L.; DOUGHERTY, F. C.: A Flexible Grid Embedding Technique with Application to the Euler Equations. *AIAA Paper 83-1944*, 1983

[2] BONET, J.; PERAIRE, J.: An Alternating Digital Tree (ADT) Algorithm for 3D Geometric Searching and Intersection problems. In: *International Journal for Numerical Methods in Engineering*, Vol. 31, 1991, pp. 1-17

[3] BLAYLOCK, T. A.; ONSLOW, S. H.; ALBONE, C. M.: Mesh Generation and Flow Solution for Complex Configurations using the FAME System. In: *Proceedings of the 1993 European Forum on Recent Developments and Applications in Aeronautical CFD*, Royal Aeronautical Sosiety, Bristol, England, 1. - 3. September, 1993, pp. 8.1 – 8.14

[4] CHAN, W. M.; BUNING, P. G.: Zipper grids for Force and Moment Computation on Overset Grids. *AIAA Paper 95-1681*, 1995

[5] CHAN, W. M.; MEAKIN, R. L.; POTSDAM, M. A.: CHSSI Software for Geometrically Complex Unsteady Aerodynamic Applications. *AIAA Paper 2001-0593*, 2001

[6] CHAN, W.; GOMEZ III, R. J.; ROGERS, S. E.; BUNING, P. G.: Best Practices in Overset Grid Generation. *AIAA Paper 2002-3191*, 2001

[7] D'ALASCIO, A.; BERTHE, A.; LE CHUITON, F.: Application of CFD to the Fuselage Aerodynamics of the EC145 Helicopter. Prediction of Unsteady Phenomena and of the Time Averaged Influence of the Main Rotor. In: *Proceedings of the 29th European Rotorcraft Forum*, Paper 39, Friedrichshafen, Germany, September 16-18, 2003

[8] JEANFAVRE, G.; BENOIT, C.; LE PAPE, M.-C.: Improvement of the Robustness of the Chimera Method. *AIAA Paper 2002-3290*, 2002

[9] JUVIGNY, X.; CANNONE, E.; BENOIT, C.: Multigrid Algorithms for the Chimera Method. *AHS Paper 2004-0758*, 2004

[10] KHIER,W.; SCHWARZ, T.: Time-accurate simulation of the flow around the complete BO 105 wind tunnel model. In: *Proceedings of the 31st European Rotorcraft Forum*, Paper 87, Florence, Italy, September 13 – 15, 2005

[11] KROLL, N.; ROSSOW, C.-C.; BECKER, K.; THIELE, F.: The MEGAFLOW Project. In: *Aerospace Science and Technology*, Vol. 4, 2002, pp. 223-237

[12] LE CHUITON, F.: Chimera Simulation of a complete helicopter with rotors as actuator discs. 14th Symposium of STAB, Bremen, Germany, November 16. – 18., 2004, to appear in: *Notes in Numerical Fluid Mechanics and Multidisciplinary Design*, Springer, 2005

[13] MEAKIN, R. L.: An efficient means of Adaptive Refinement within Systems of Overset Grids. *AIAA Paper 95-1722*, 1995

[14] MERTINS, R.; ELSHOLZ, E.; COLAK, B.; BARAKAT, S.: 3D Viscous Flow Analysis on Wing-Body-Aileron-Spoiler Configurations. In: *Proceedings of the Deutscher Luft- und Raumfahrtkongress 2003*, Paper DGLR-2003-125, Munich, Germany, November 17-20, 2003

[15] PAHLKE, K.; VAN DER WALL, B.: Chimera Simulations of Multibladed Rotors in High-Speed Forward Flight with weak fluid-structure coupling. In: *Aerospace Science and Technology*, Vol. 9, No. 5, 2005, pp. 377-389

[16] PAHLKE,K.; COSTES, M.; D'ALASCIO, A.; CASTELLIN, C.; ALTMIKUS, A.: The 6-year French-German CHANCE Project. In: *Proceedings of*

the 31st European Rotorcraft Forum, Paper 69, Florence, Italy, Spetember 15-17, 2005

[17] PREWITT, N. C.; BELK, D. M.; MAPLE, R. C.: Multiple Body Trajectory Calculations Using the Beggar Code. In: *Journal of Aircraft*, Vol. 36, No. 5, 1999, pp. 802-808

[18] RIGBY, D. L.: Method of the Weakest Descent for Automatic Block Merging. In: *Proceedings of the 15th International Conference on Numerical Methods in Fluid Dynamics*, Monterey, California, USA, June 1996

[19] SCHWARZ, T.: Development of a Wall Treatment for Navier-Stokes Computations using the Overset-Grid Technique. In: *Proceedings of the 26th European Rotorcraft Forum*, The Hague, The Netherlands

[20] SCHWARZ, T.: Enhancement of a Navier-Stokes Flow Solver for Patched Grids with Non-Coincident Grid Nodes. In: *Notes on Numerical Fluid Mechanics*, Vol. 77, Springer, 2002, pp. 312-319

[21] WANG, Z. J.; PARTHASARATHY, V.: A Fully Automated Chimera Methodology for Multiple Moving Body Problems. In: *International Journal for Numerical Methods in Fluids*, Vol. 33, 2000, pp. 919-938