A MULTIBLOCK GRID GENERATION TECHNIQUE APPLIED TO A JET ENGINE CONFIGURATION

Mark E. M. Stewart
Institute for Computational Mechanics in Propulsion
NASA Lewis Research Center
Cleveland, Ohio 44135

SUMMARY

Techniques are presented for quickly finding a multiblock grid for a two-dimensional geometrically complex domain from geometrical boundary data. An automated technique for determining a block decomposition of the domain is explained. Techniques for representing this domain decomposition and transforming it are also presented. Further, a linear optimization method may be used to solve the equations which determine grid dimensions within the block decomposition. These algorithms automate many stages in the domain decomposition and grid formation process and limit the need for human intervention and inputs. They are demonstrated for the meridional or throughflow geometry of a bladed jet engine configuration.

INTRODUCTION

A central issue in performing numerical simulations in complex geometries is determining a set of points on which the equations can be discretized, and then finding techniques which will numerically solve this set of discrete equations. There are many approaches to this problem, but the sampling points are either locally structured or unstructured. Structured grids have a trivial local relationship between neighbors while the neighbor relationships in unstructured grids must be stored in tables. This difference has implications for the algorithms which may be implemented, the methods of generating grids and ancillary issues. For example, matrix formulations of implicit schemes on unstructured grids lose the bandedness possible with structured grids. However, when using structured grids, the burden of dealing with complex geometries shifts to how the blocks interact with each other.

For these classes of grids, there are many specific approaches to complex geometry simulations. Euler solutions for aircraft configurations using unstructured grids have been presented by Jameson, Baker, and Weatherill[1] and Lohner and Parikh.[2] Buning et. al.[3] have simulated an ascending Space Shuttle using overset grids, and Chesshire and Henshaw[4] have demonstrated Navier-Stokes solutions with overlapping grids in two dimensions. Non-overlapping, structured multiblock grids and solutions for aircraft configurations have been found by Sawada and Takanashi[5] while Whitfield et. al.[6] have found Euler solutions for counter-rotating propfans where the structured grids are in relative motion.

Stewart[7,8] has used domain decomposition techniques and multiblock grids to find Euler solutions for multi-element airfoil sections and unbladed jet engine configurations.

The focus of this paper is on non-overlapping structured grids and techniques for quickly generating domain decompositions and grids for multiblock flow solvers. There have been several approaches to this problem. Steinbrenner et. al.[9] have demonstrated an interactive system for three-dimensional domain decomposition and grid generation. Vogel[10] has studied knowledge-based techniques for performing domain decomposition. The approach is to define operations acting on bodies in a two-dimensional domain which include enclosing a body with a region, connecting several bodies or dividing a region. The operations are organized in a depth-first search which is managed by a rule-based system incorporating expert knowledge. The rules are specialized to deal with the bodies within the domain. Allwright[11] has demonstrated a three-dimensional grid generation system which starts from an initial approximation to a decomposition and inserts sub-components into the decomposition. Dannenhoffer[12] has presented techniques for two-dimensional domain decomposition which include both the idea of an initial approximation to the topology and specialized rules in an expert system to find a suitable set of grid blocks.

In the following section, an algorithm for determining a domain decomposition is presented. Then, a representation of a decomposed domain is explained which allows rule-driven transformations to be applied. In the next section, an algorithm for dimensioning grids is explained. Lastly, these techniques are demonstrated for a jet engine meridional flow plane.

DOMAIN DECOMPOSITION

The decomposition of a two-dimensional domain may be performed using a search algorithm which finds boundary conforming regions in a two-dimensional domain. In the same way that the skin of a balloon will conform to the bounding walls when blown up in a confined space, this algorithm refines a trivial, coarse approximation to the perimeter of a region so that it conforms to any neighboring boundaries without excessive stretching. To do this, the algorithm uses both a directional probe to look for a bounding point above an interval and a set of rules to interpret the results. Based on the endpoints and which curves they lie on, the rules determine which intervals to probe, which bounding points to add or remove from the perimeter and hence the new subsegments of the perimeter. These subsections and searches are organized in a tree structure which may be significantly pruned by optimizations.

The algorithm is demonstrated in figure 1 by considering the simple case of an airfoil in a box, $PQRS$, and an initial, coarse approximation to the region below the airfoil, $ACRS$. To transform $ACRS$ to a region which conforms to the lower surface of the airfoil, the probe is applied to AB and determines the highest flat ceiling above the middle third of AB, EF, as in figure 1b). Limiting the depth of this probe to the height Search Depth prevents finding points on the segment PQ which would yield a perimeter with excessive stretching. The probe considers the curves which define the domain and finds the point D. The perimeter is then modified to $ADBC$ as in figure 1c), and two child segments, AD and DB, are created from AB.

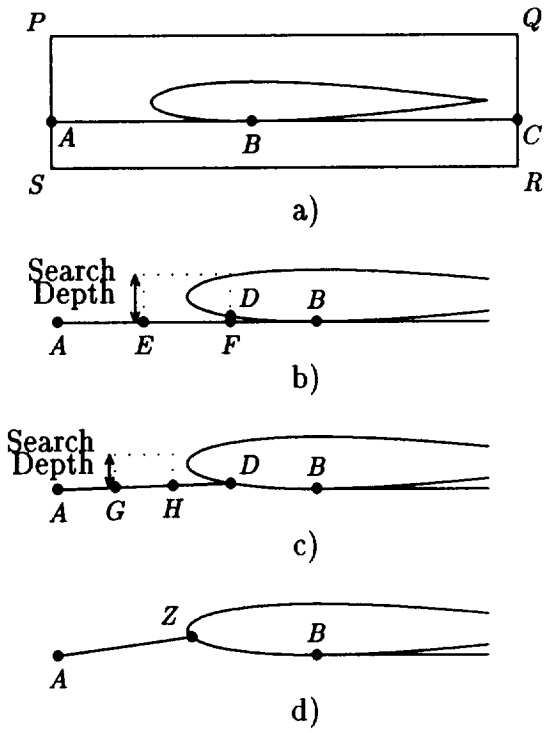


Figure 1. Steps in finding a boundary conforming region.

The refinement continues at the next finer scale by considering AD and DB for refinement. The refinement rules dictate that AB be refined since the endpoints, A and B, lie on different curves. DB is not refined since its endpoints lie on the same boundary defining curve. Probing above the middle third of AD, GH, yields no point within search depth and refinement rules require that the two outer segments, AG and HD, be refined. This failure to find a highest ceiling implies that there will be a transition between bodies in the perimeter. The refinement is extended in a tree structure by considering successively smaller sub-segments as above, and eventually yields the perimeter AZB of figure 1d).

The example demonstrates refinement in one direction on one side of a block, but the technique may be applied on four sides in the outward direction. This allows the perimeter to be extended on all sides. Further, by alternating the direction of search at corners, the directionality of the scheme is reduced.

Once a region is found, it may be removed from the domain to yield a reduced domain. The algorithm may then be applied again to the reduced domain. In this manner, non-overlapping, boundary conforming regions are successively found and removed until the domain is covered.

MANIPULATION OF A DECOMPOSITION

This decomposition strategy finds a decomposition of a complex two-dimensional geometry in terms of topologically rectangular regions. However, for manipulation of this decomposition, a simple and consistent representation is necessary. The decomposition is represented by representation entities linked together in a network to form a semantic network. Geometrical and logical information about the domain and its decomposition is encoded in variables and pointers bundled with each instance of an entity. Processes may then be constructed which traverse this network and decide how to supplement or transform the representation.

A structure is a "C" language programming construct which is used to represent the blocks, curve segments and other entities in the decomposition. A structure bundles together variables describing an entity. For example, a basic entity represented by a structure is the perimeter curve segment, seg, which is a section of the perimeter defining a block. Bundled in its structure are variables describing type and geometrical information and pointers to its neighbors in the perimeter and in adjacent blocks. The links defined by these pointers are shown conceptually in figure 2. By having segments point to neighboring curves in the perimeter, a circular linked list is formed which defines the block's perimeter. By having a segment point to a curve across a block interface in an adjacent block, local decomposition information is represented.

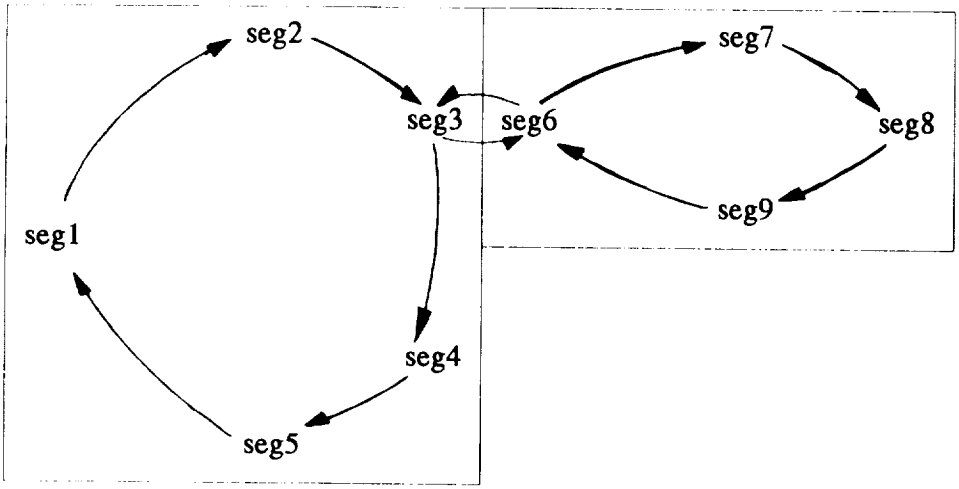


Figure 2. Stylized blocks from a decomposition showing the links both between adjacent blocks and between segments in a perimeter. Additional data is bundled in structures representing each segment.

Replication of these basic, generic units reduces a complex geometry and its decomposition to a set of similar objects (curve segments and blocks) and their important semantic associations. However, processes may then be written which interact with this representation of blocks and segments and supplement it or isomorphically transform it. For example, the perimeter of a topologically rectangular region has four corners. To find appropriate corner positions for a raw perimeter, geometrical information from the perimeter curve entities is interpreted by a set of rules in one process. Similarly, blocks in the decomposition may be straightened by using rules to interpret geometrical information

from the representation and decide where cuts should be made. Further, to balance the computational load of a simulation in parallel, another process uses rules which interpret grid size and topological features to decide where to merge and cut blocks and grids. Each of these processes work on the decomposition and yield rule driven transformations of the decomposition.

GRID DIMENSIONING AND FORMATION

One of the processes which acts on this data structure finds grid dimensions within each block of the decomposition. To have a structured grid within each block, it is clear that the number of cells on opposite sides of a block must be equal. For simplicity in the numerical scheme, it is assumed that normal coordinate lines match at block interfaces. To satisfy this condition, the number of cells on opposite sides of a block interface must be equal. These two constraints give two equations with integer coefficients for each block.

For example, figure 2 depicts two blocks from a decomposition. If the number of cells along a segment is denoted $seg_i \rightarrow n$, then the resultant equations are

$$seg_1 \rightarrow n = seg_3 \rightarrow n + seg_4 \rightarrow n, \tag{1}$$

$$seg_2 \rightarrow n = seg_5 \rightarrow n,$$

$$seg_7 \rightarrow n = seg_9 \rightarrow n,$$

$$seg_3 \rightarrow n = seg_8 \rightarrow n.$$

A system of equations is formed from the equations from all blocks, and this system is generally underconstrained.

The system of equations may be simplified by removing the trivial equations, giving a system where the k^{th} equation is

$$\sum_i a_{ki} n_i = 0, \tag{2}$$

where $seg_i \rightarrow n = n_i \in \mathbf{Z}^+$ and $a_{ki} \in \mathbf{Z}$ are coefficients. If we define z_k such that

$$\sum_i a_{ki} n_i = -z_k, \tag{3}$$

then optimizing the objective function

$$-\sum_k z_k = \sum_k \sum_i a_{ki} n_i \tag{4}$$

with the constraints (3) is a linear programming problem which may be solved using the simplex method.

If in the solution we have $z_k \neq 0$ for any k, then the constraint equations (2) are inconsistent and there is no solution. Since a_{ki} are integers and the simplex method involves operations which are

closed under the rational numbers, the solutions will be rational. Consequently, integer solutions are ensured by taking integer multiples of the dimension solutions.

Since the system of equations (2) is generally underconstrained, the solution is not unique. Consequently, the grid may be adapted to give appropriate resolution of the solution. If $\{n_i\}$ is the solution set, then an integer multiple, l, of the solution, $\{ln_i\}$, is also a solution. Further, the solution vector may also have independent subspaces where solutions, n_i, are not subject to all the constraints (2). In practice, many of the dimensions, n_i, are not constrained at all and may be freely varied. Consequently, adaptation of the solution is often straightforward.

Grids are formed in the multiblock grid based on the perimeter data and grid dimensions. Initial grids are found with a Coons patch followed by elliptic smoothing in the interior of each grid block and at common interfaces. Singularities in the grid must also be smoothed.

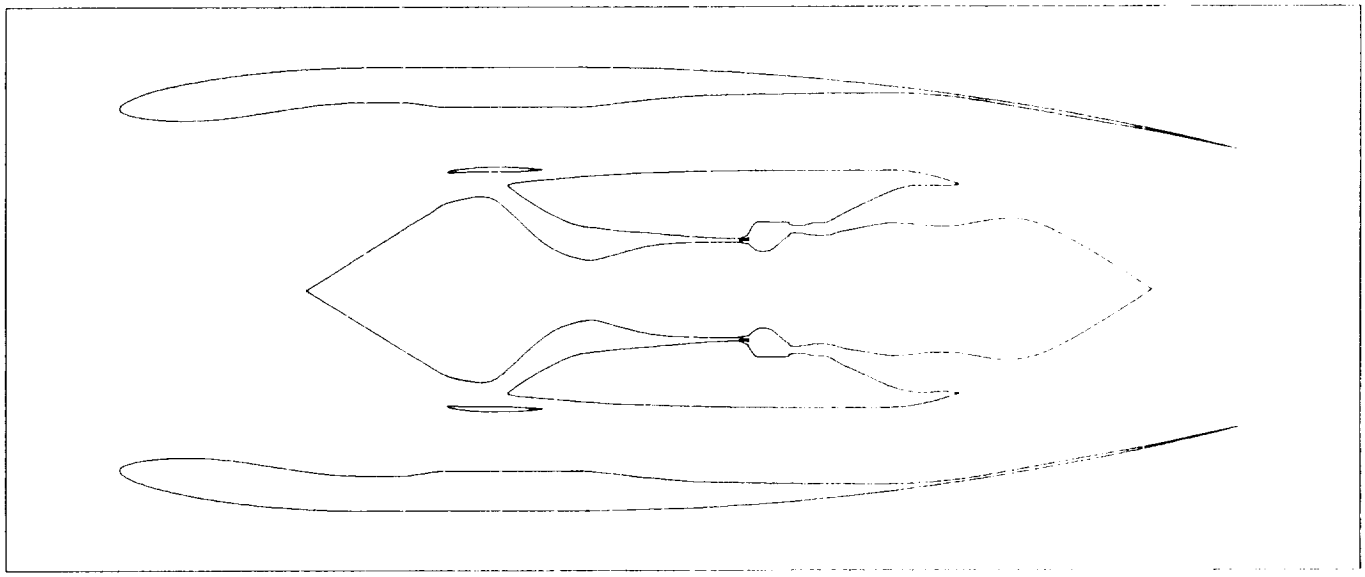
To simplify modeling blade effects in the engine meridional section, these grid smoothing techniques must be specialized to align coordinate lines with the leading and trailing edges of blade rows. The grid may be searched to find the appropriate coordinate line and align it with the leading or trailing edge.

RESULTS

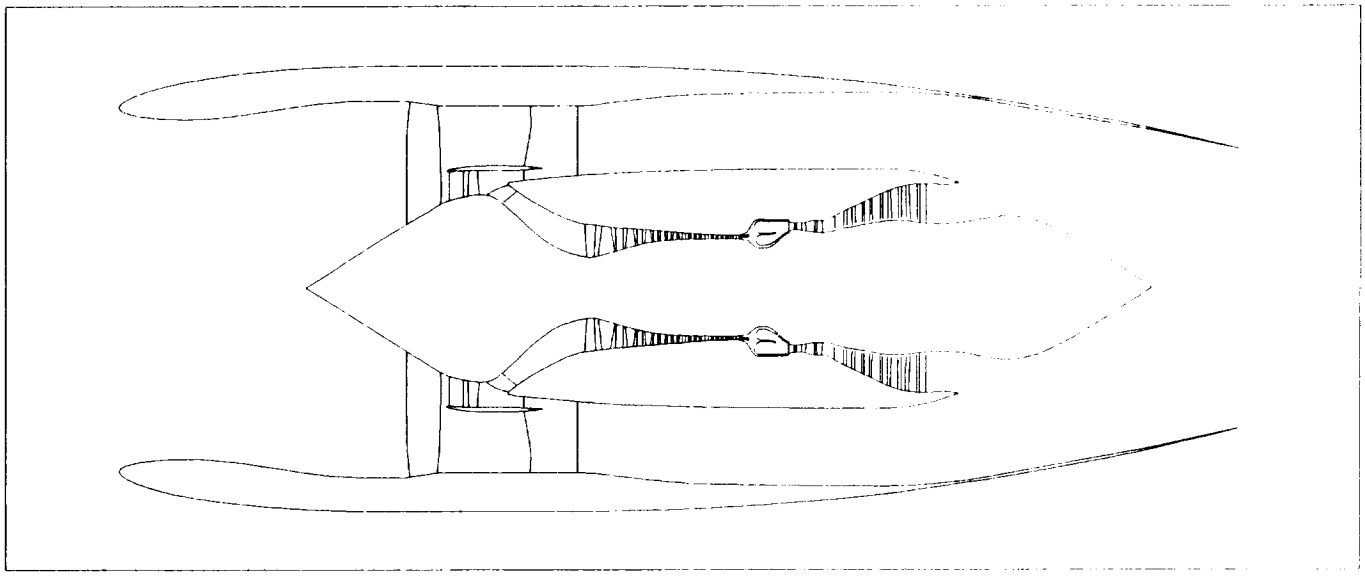
As a demonstration of these techniques, the meridional or throughflow plane of a jet engine is considered. The engine geometry, shown in figure 3, is based on the Energy Efficient Engine (E^3) which was built by General Electric Aircraft Engines under contract to NASA.[13] The engine flow involves external flow about the nacelle and internal flow through the inlet, bypass duct and core compressor, combustor and turbines which are shown in figure 3a. The throughflow geometry is axisymmetric about the centerline and both half planes are included in the domain for solution comparison. In the engine ducts there are alternating stationary and rotating blade rows which turn the flow and either do work by compressing the flow or extract energy as the flow expands. The profiles of these blade rows are included in the grid to facilitate modeling blade effects and they are shown in figure 3b.

The domain decomposition algorithm is applied to this geometry in the form of coordinates for the boundaries shown in figure 3a. As a specialization of the technique to accommodate blade models, block interfaces are forced at the blade leading and trailing edges shown in figure 3b. The resulting decomposition is shown in figure 4. The decomposition is then transformed by several processes traversing the representation. For example, the boundary layer blocks which ensure that grid singularities are off the surface are merged with adjacent blocks. Similarly, the placement of corners is changed. Some manual transformations have also been performed to simplify the decomposition.

The system of dimensioning equations for the final block decomposition (2) contains 76 non-trivial equations which relate 20 of the 230 solution variables, n_i, which represent grid dimensions. The remaining 210 solutions, n_i, are independent of the constraint equations (2). The independence of solution variables is demonstrated in any of the blocks which span a duct in figure 5. The constraint



a)



b)

Figure 3. a) Flowpath geometry for the engine meridional geometry and b) flowpath geometry with blade and combustor component profiles.

equations (1) require the number of cells along opposite walls to be equal and for topological reasons this dimension is independent of all other block dimensions. The dependencies between dimension solutions, n_i, are demonstrated in the transverse direction where the grid dimension couples with blocks throughout the engine.

After solving these equations for a coarse grid, the dimension solution, n_i, is refined and adapted by finding other solutions to the equations. Multigrid levels are added by multiplying all dimensions, n_i, by a power of two. Further adaptation is done by changing variables in the independent subspaces

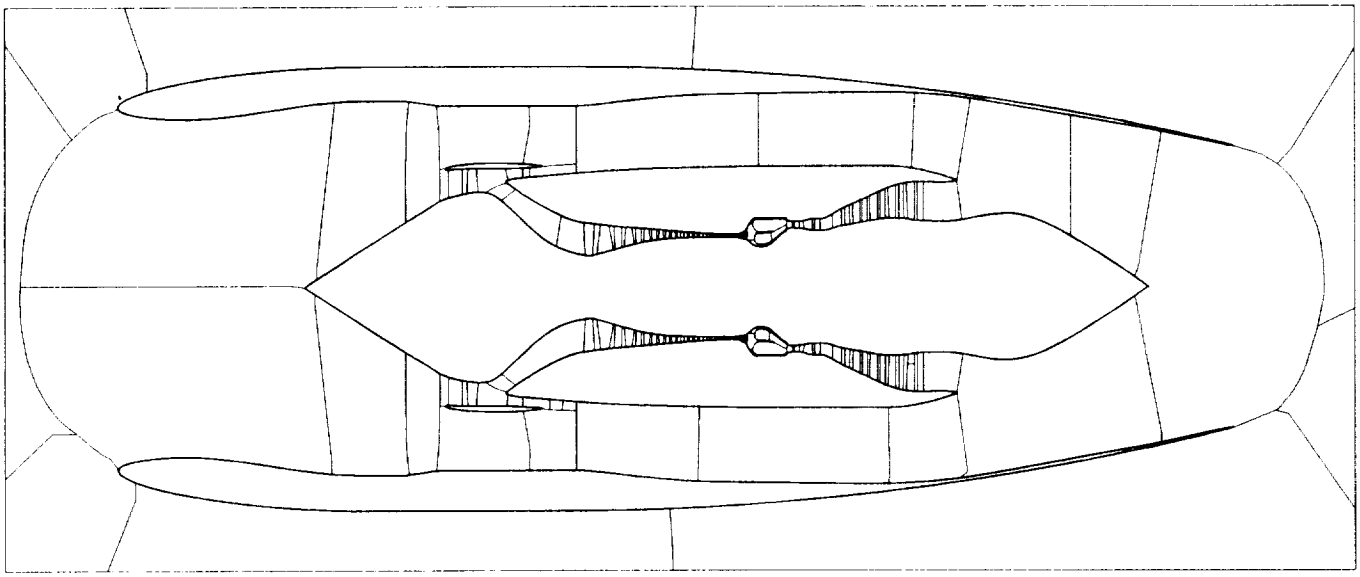


Figure 4. The domain decomposition produced by the decomposition algorithm. Block interfaces are forced to resolve the profiles of blades and other components. The decomposition consists of and is represented by blocks containing curve segments.

noted above. One grid permitted by the dimensioning equations is shown in figure 5. After merging grid blocks for vector architectures, the grid contains 26440 cells in 36 blocks which are indicated by the solid grid lines in the diagram. The grid extends to a circular boundary in the far field.

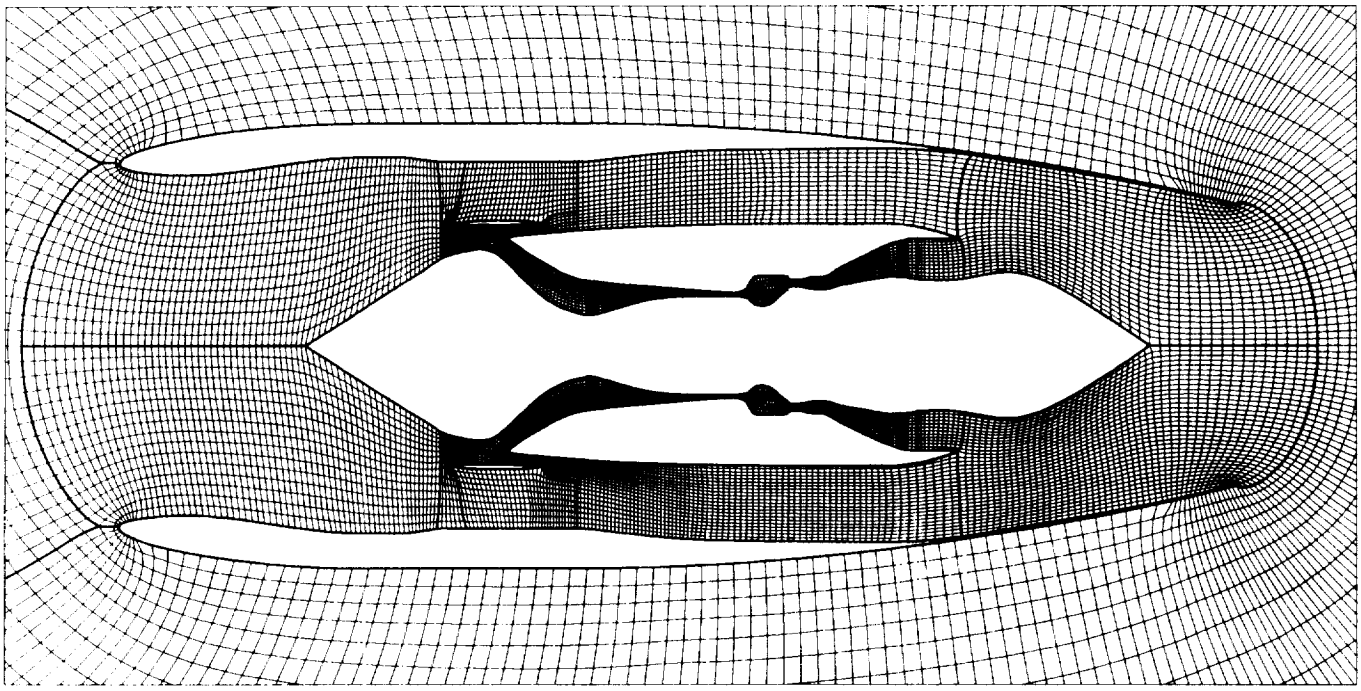


Figure 5. The inner region of the grid for the engine meridional plane. The grid contains 26440 cells in 36 blocks which are indicated by the bold grid lines. Grid lines are aligned with the profiles of blades and combustor components.

CONCLUSIONS

Techniques for generating multiblock grids have been presented and demonstrated for a jet engine meridional or throughflow section. In particular, an algorithm for determining a domain decomposition from boundary coordinate data is explained. Also, representation and transformation of this decomposition is noted. Techniques for dimensioning grids within the decomposition are also formulated. These algorithms automate many stages in the domain decomposition and grid formation process.

REFERENCES

[1] Jameson, A.; Baker, T. J.; and Weatherill, N. P.: Calculation of Inviscid Transonic Flow over a Complete Aircraft. AIAA-86-0103, Jan. 1986.

[2] Lohner, R.; and Parikh, P.: Generation of Three-Dimensional Unstructured Grids by the Advancing-Front Method. AIAA-88-0515, Jan. 1988.

[3] Buning, P.; Chiu, I.; Obayashi, S.; Rizk, Y.; and Steger, J.: Numerical Simulation of the Integrated Space Shuttle Vehicle in Ascent. AIAA-88-4359, Aug. 1988.

[4] Chesshire, G.; and Henshaw, W.: Composite Overlapping Meshes for the Solution of Partial Differential Equations. *Journal of Computational Physics*, Vol. 90, 1990, pp. 1-64.

[5] Sawada, K.; and Takanashi, S.: A Numerical Investigation on Wing/Nacelle Interferences of USB Configuration. AIAA-87-0455 Jan. 1987.

[6] Whitfield, D.; Swafford, T.; Mulac, R.; Belk, D.; and Janus, J.: Three-dimensional Unsteady Euler Solutions for Propfans and Counter-rotating Propfans in Transonic Flow. AIAA-87-1197 June 1987.

[7] Stewart, M.: A General Decomposition Algorithm Applied to Multi-Element Airfoil Grids. AIAA-90-1606, Seattle, June 1990.

[8] Stewart, M.: Euler Solutions for an Unbladed Jet Engine Configuration. AIAA-92-0544, Reno, January 1992.

[9] Steinbrenner, J.; Chawner, J.; Fouts, C.: A Structured Approach to Interactive Multiple Block Grid Generation. AGARD FDP meeting Leon, Norway, May 1989.

[10] Vogel, A.: *A Knowledge-Based Approach to Automated Flow-Field Zoning for Computational Fluid Dynamics*, NASA Tech. Memo 101072, April 1989.

[11] Allwright, S.: Techniques in Multiblock Domain Decomposition and Surface Grid Generation, Proceedings of 2nd International Conference on Numerical Grid Generation in Computational Fluid Dynamics, Miami 1988.

[12] Dannenhoffer, J.: Computer-Aided Block-Structuring Through the Use of Optimization and Expert System Techniques. AIAA-91-1585, June 1991.

[13] Stearns, E. M.; *Energy Efficient Engine Integrated Core/Low Spool Design and Performance Report*, NASA CR-168211, Feb. 1985.