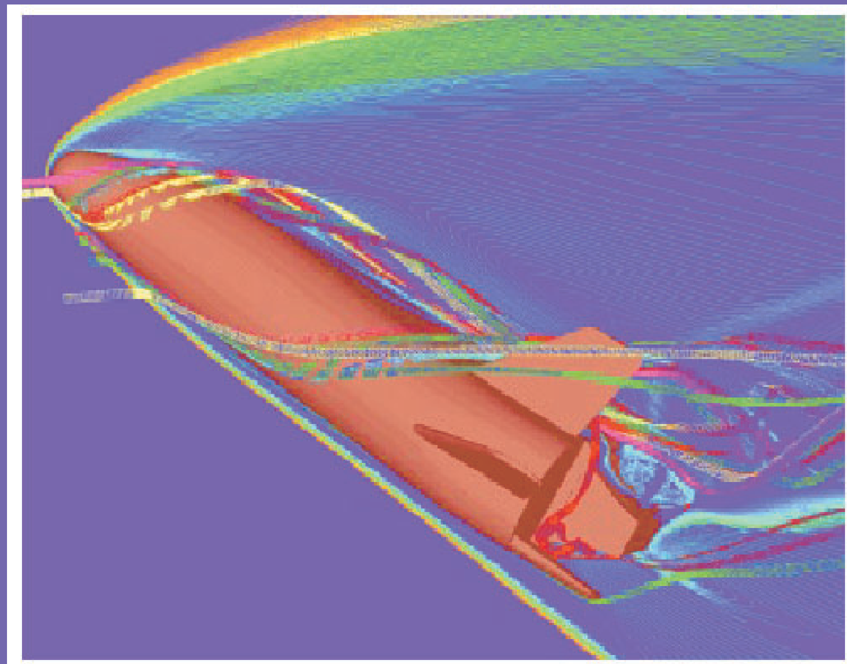
From CAD to Simulation Run

Automatic Grid Generation and Interfacing



Gamm Conference

25 – 28 March 2002, Augsburg, Germany

J. Häuser Dept. of HPCC, CLE, Salzgitter, Germany
P.R. Eiseman, Z. Cheng, PDC, White Plains, N.Y., U.S.A.
H.-G. Paap, HPC Consultants, Regensburg, Germany

Acknowledgments

This research was partly funded by the ministry of Science and Culture of the State of Lower Saxony, Germany and the European Commission under contract JavaPar 1997.262 and EXTV 1999.045.

The authors are grateful to T. Gollnick, T. Ludewig, Dept. of HPCC, CLE and Y. Xia, ESTEC-ESA, NL for providing material for this presentation.

Roy D. Williams, Center of Advanced Computational Research, Caltech, U.S.A. provided computing time.

The authors are obliged to Jean Muylaert, ESA, NL for geometry data and numerous stimulating discussions.

Presentation Overview

The following topics will be discussed with varying level of detail, each being itself a major research area

CAD Data Conversion

Grid Generation for Complex Geometries

Grid generation and HPCC

JavaGrid Concept

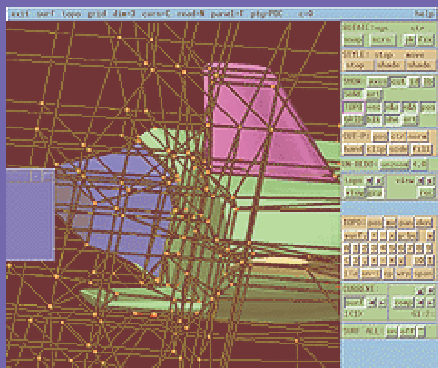
Presentation Objective and Summary

The need for accurate three-dimensional simulation in numerous fields of engineering and science for real processes requires the development of ever more sophisticated three-dimensional grids as well as powerful *High Performance Computing and Communications* (*HPCC*) resources.

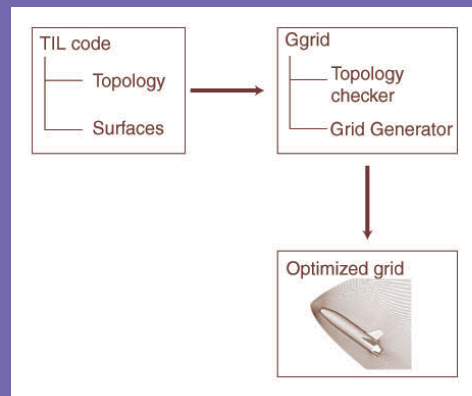
These notes present the steps from the original CAD data to the final simulated solution. The Topology Input language for multi-block grids is explained along with the software *Automatic Zoning Manager* (AZ) for interactive topology design. Numerous utilities for grid clustering, merging, setting BCs for solvers are available, too. Furthermore, the strategy for parallelization of complex grids and, in particular, for Java as the language for High Performance Computing is presented. The concept of a ***JavaGrid*** is introduced, that promises to provide the combined power of networked computational resources for solving most complex scientific and engineering problems both in geometry and in physics.

Software GridPro Overview

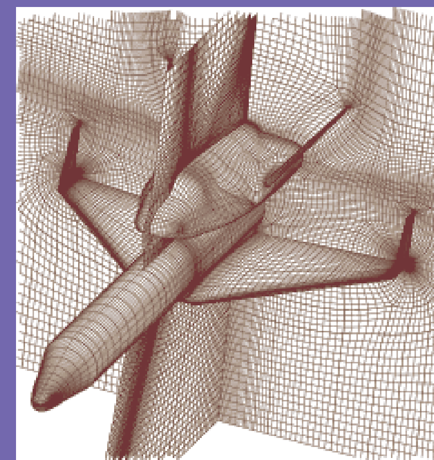
AZ-Manager



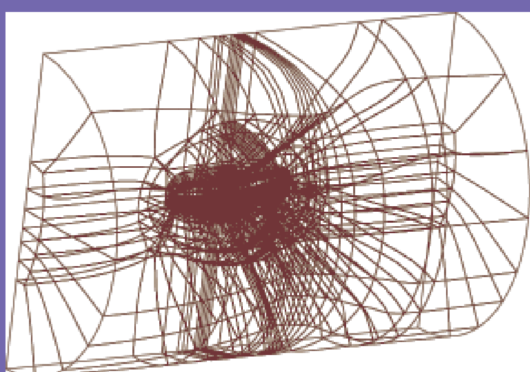
Grid generation engine



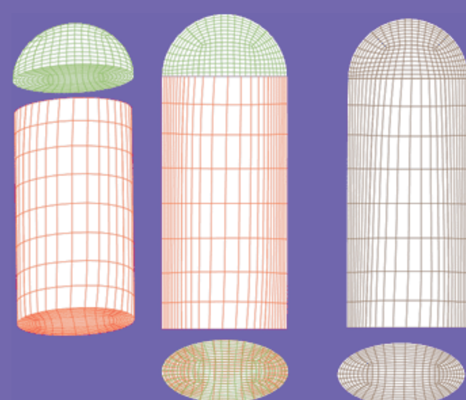
Cluster



Merge-block



Weld



Utility suite

CAD Data Conversion

Discrete Surface Descriptions

Engineering computations deal with complex geometries.

CAD data is in general not directly usable, since patches may contain overlaps, intersections, or voids. There is no guarantee for a close d surafce.

Importing Geometry

Build-in surface types: plane, ellipsoid, tube, linear, periodic boundary

Digital surfaces: Surfaces based on triangles or quads, e.g. Partran, Nastran, STL, Plot3d.

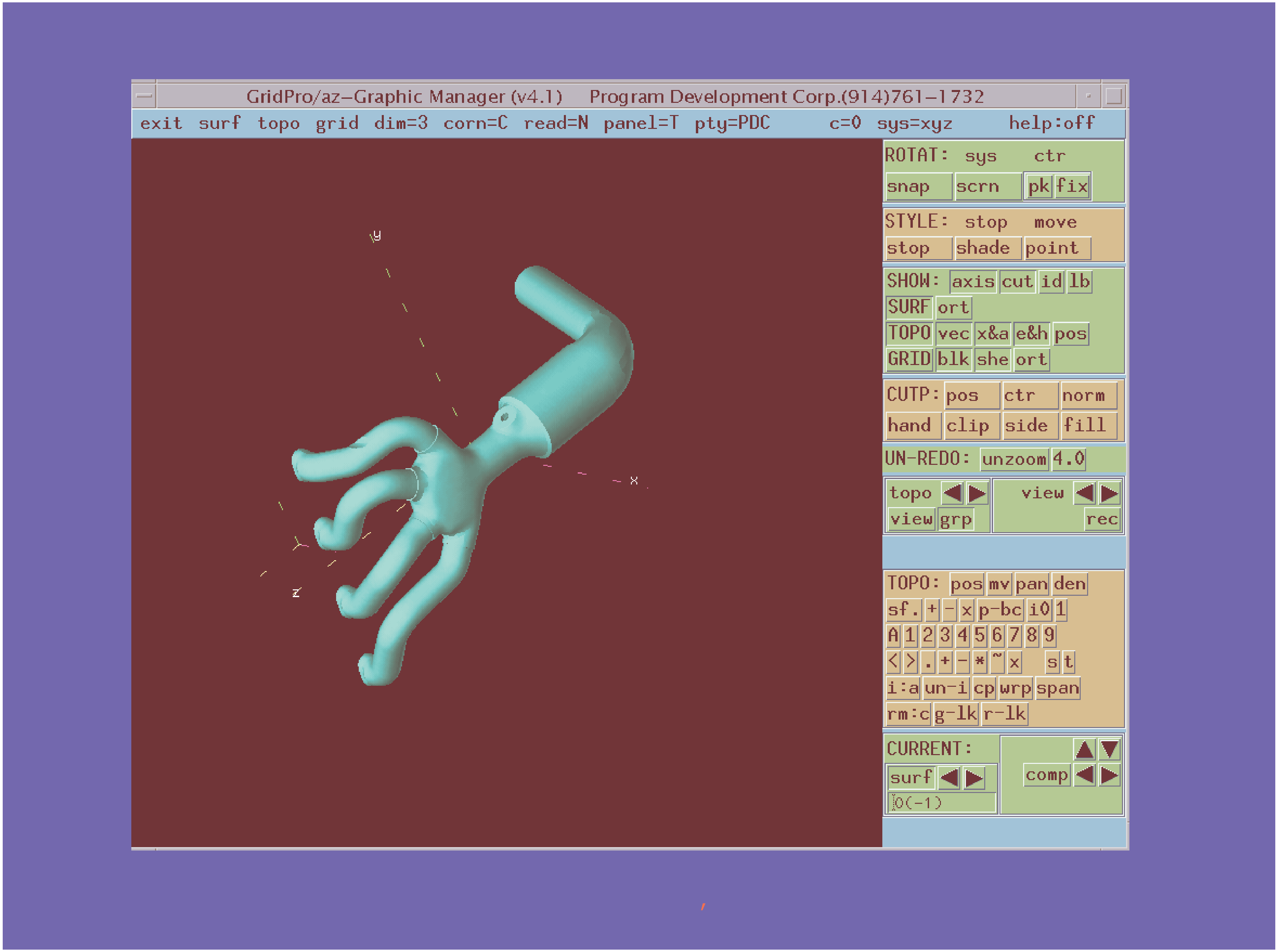
Currently under development: IGES (Tetin is similar to IGES)

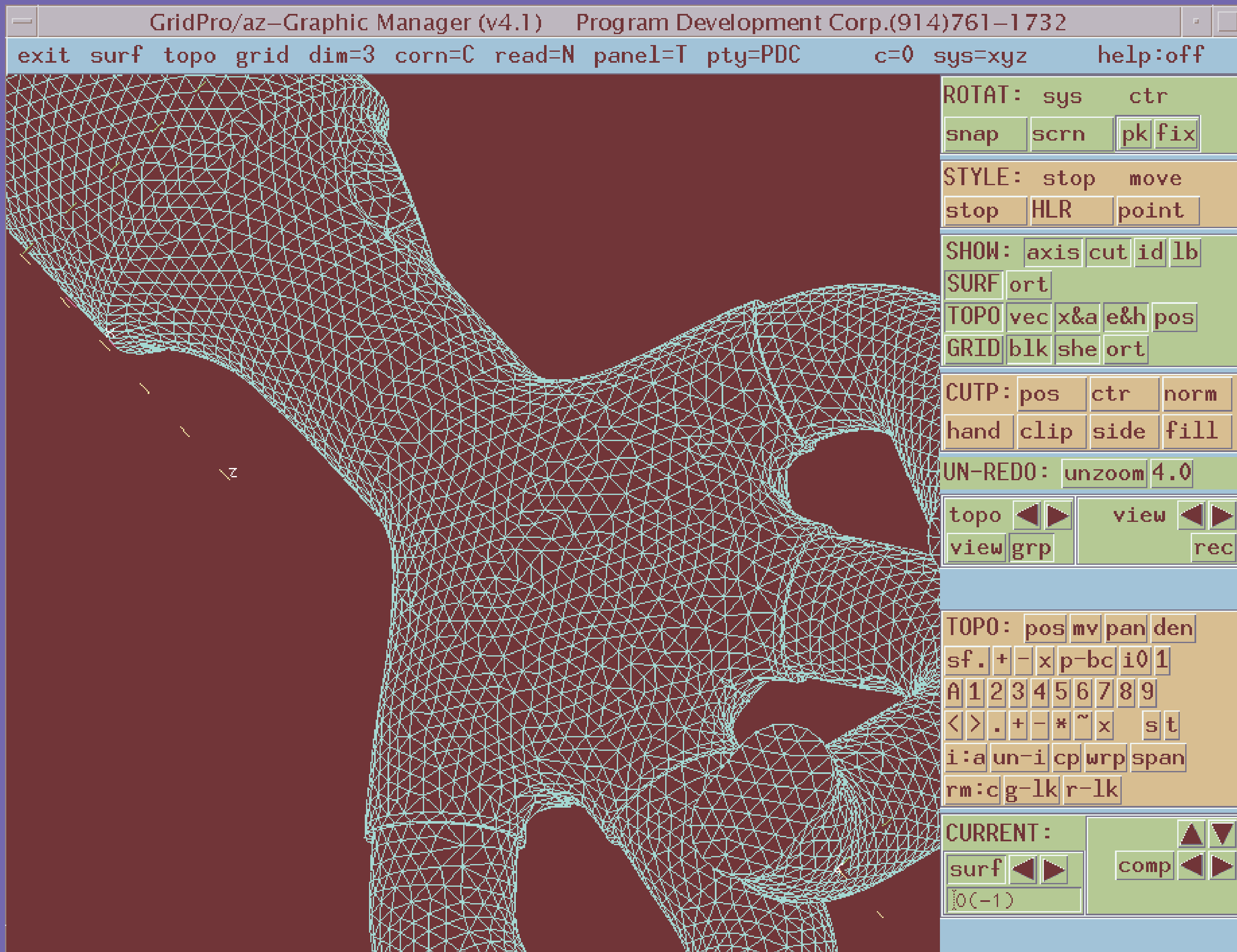
Topology Input Language (TIL)

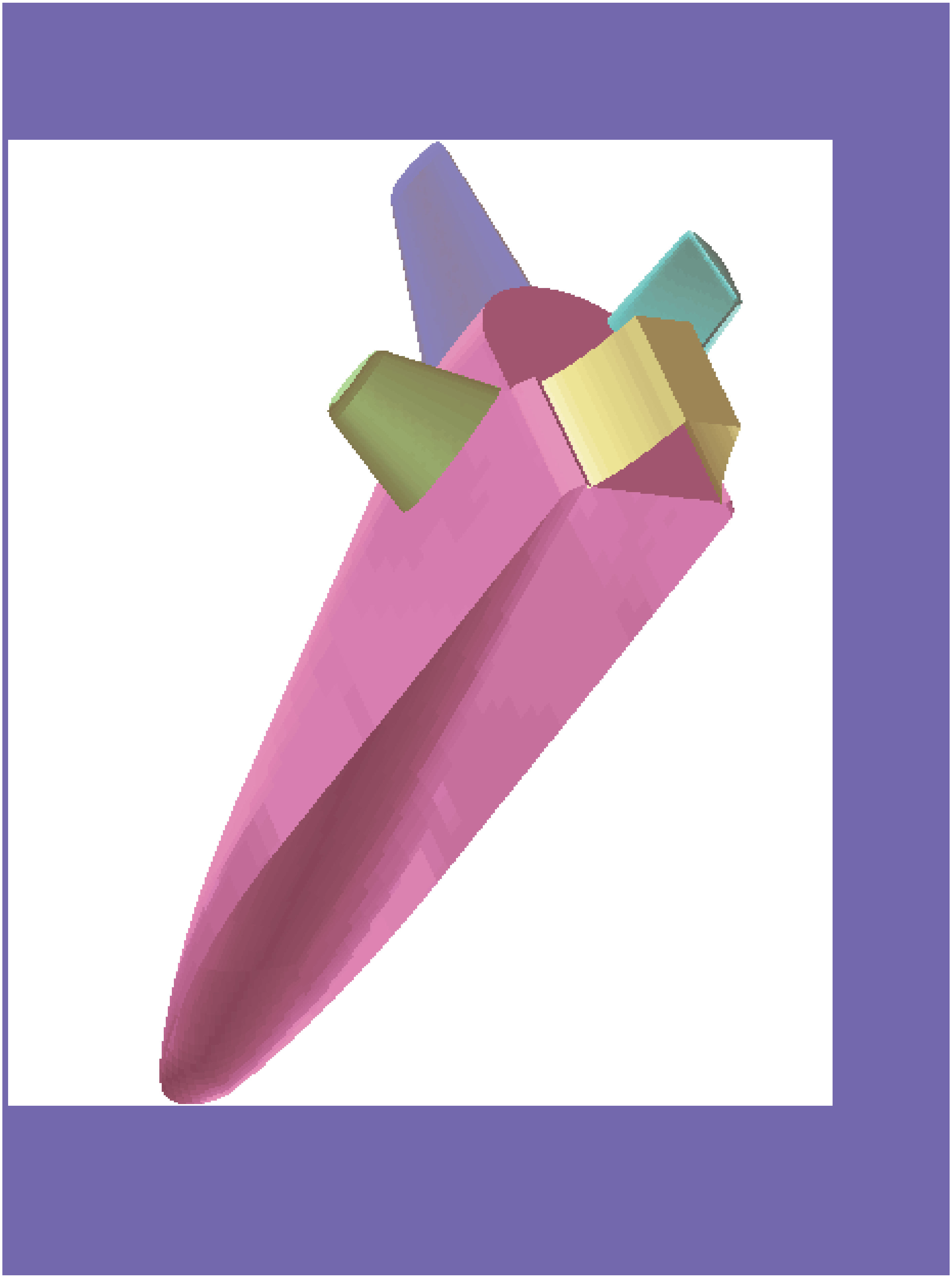
TIL code can be integrated into own software packages.

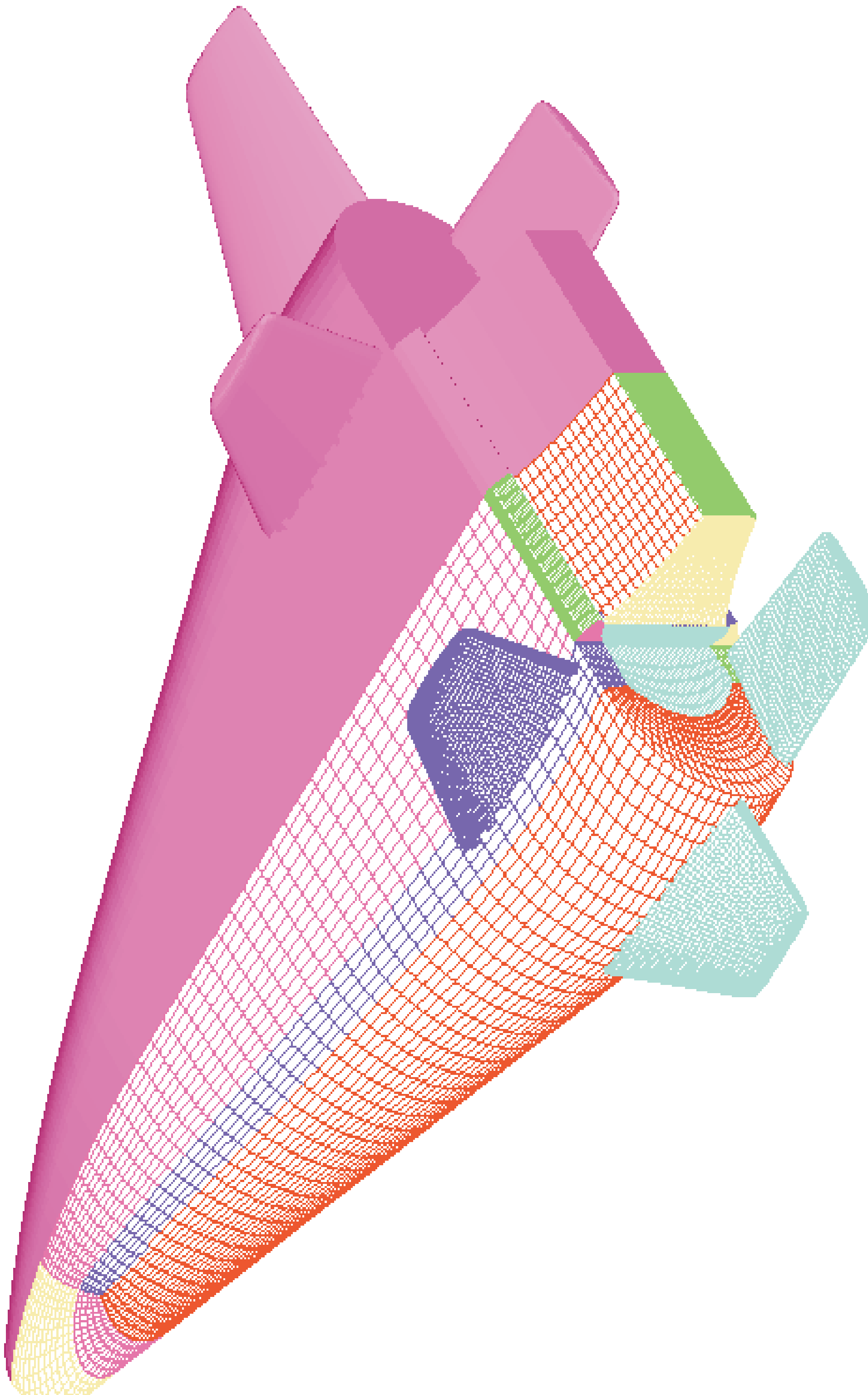
Example: CCAD from Concepts, ETI supports features for creating geometry for turbo-machinery and in addition generates TIL code automatically.

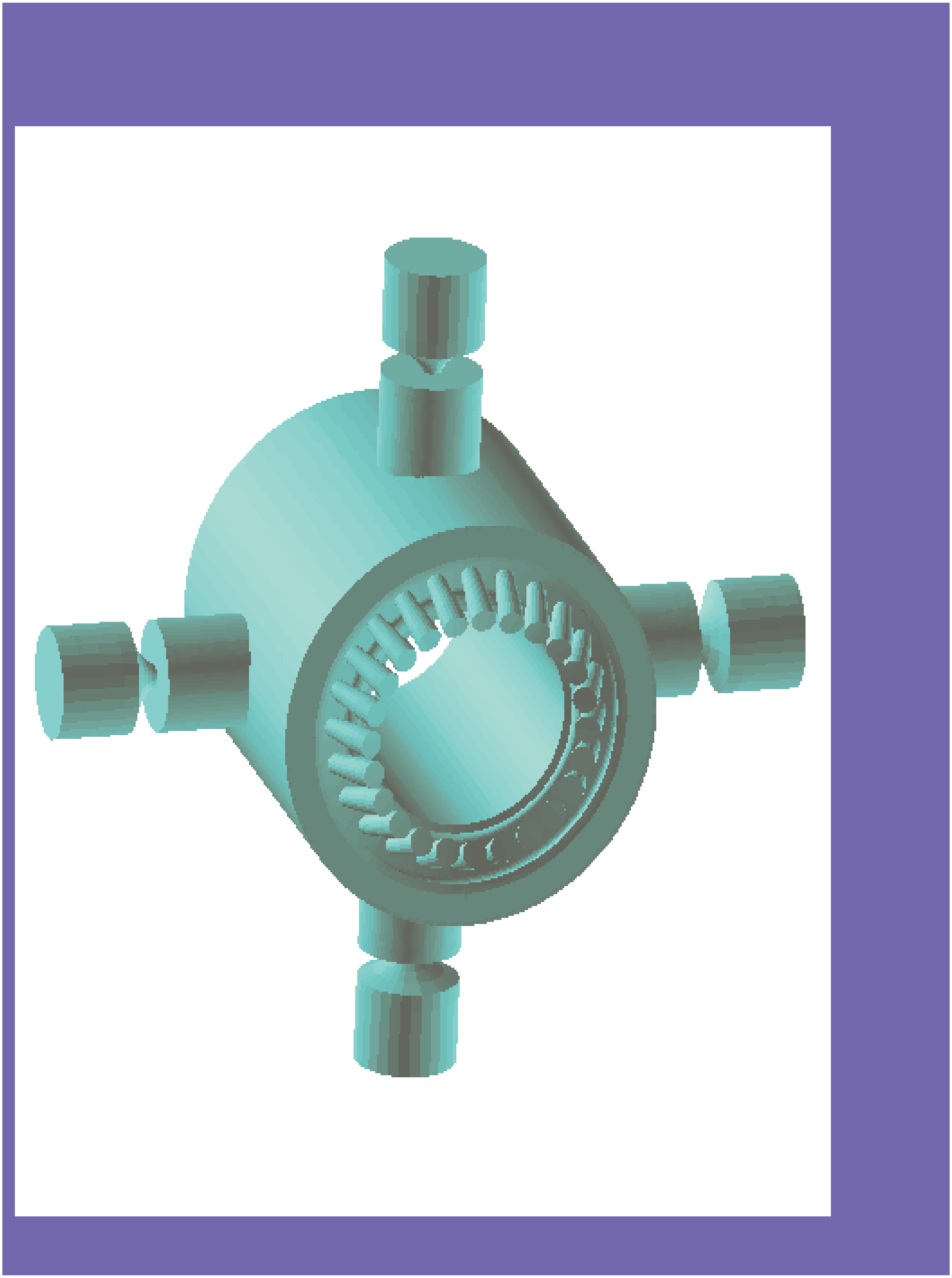
This clearly demonstrates the flexibility and strength of TIL code.


GridPro/az–Graphic Manager (v4.1) Program Development Corp.(914)761–1732
exit surf topo grid dim=3 corn=C read=N panel=T pty=PDC c=0 sys=xyz help:off
ROTAT: sys ctr
snap scrn pk fix
STYLE: stop move
stop shade point
SHOW: axis cut id lb
SURF ort
TOPO vec x&a e&h pos
GRID blk she ort
CUTP: pos ctr norm
hand clip side fill
UN-REDO: unzoom 4.0
topo view
view grp rec
TOPO: pos mv pan den
sf. + - x p-bc i0 1
A 1 2 3 4 5 6 7 8 9
< > . + - * ~ x s t
i:a un-i cp wrp span
rm:c g-lk r-lk
CURRENT:
surf comp
0(-1)
y
x
z


GridPro/az–Graphic Manager (v4.1) Program Development Corp.(914)761–1732
exit surf topo grid dim=3 corn=C read=N panel=T pty=PDC c=0 sys=xyz help:off
ROTAT: sys ctr
snap scrn pk fix
STYLE: stop move
stop HLR point
SHOW: axis cut id lb
SURF ort
TOPO vec x&a e&h pos
GRID blk she ort
CUTP: pos ctr norm
hand clip side fill
UN-REDO: unzoom 4.0
topo view
view grp rec
TOPO: pos mv pan den
sf. + - x p-bc i0 1
A 1 2 3 4 5 6 7 8 9
< > . + - * ~ x s t
i:a un-i cp wrp span
rm:c g-lk r-lk
CURRENT:
surf comp
0(-1)





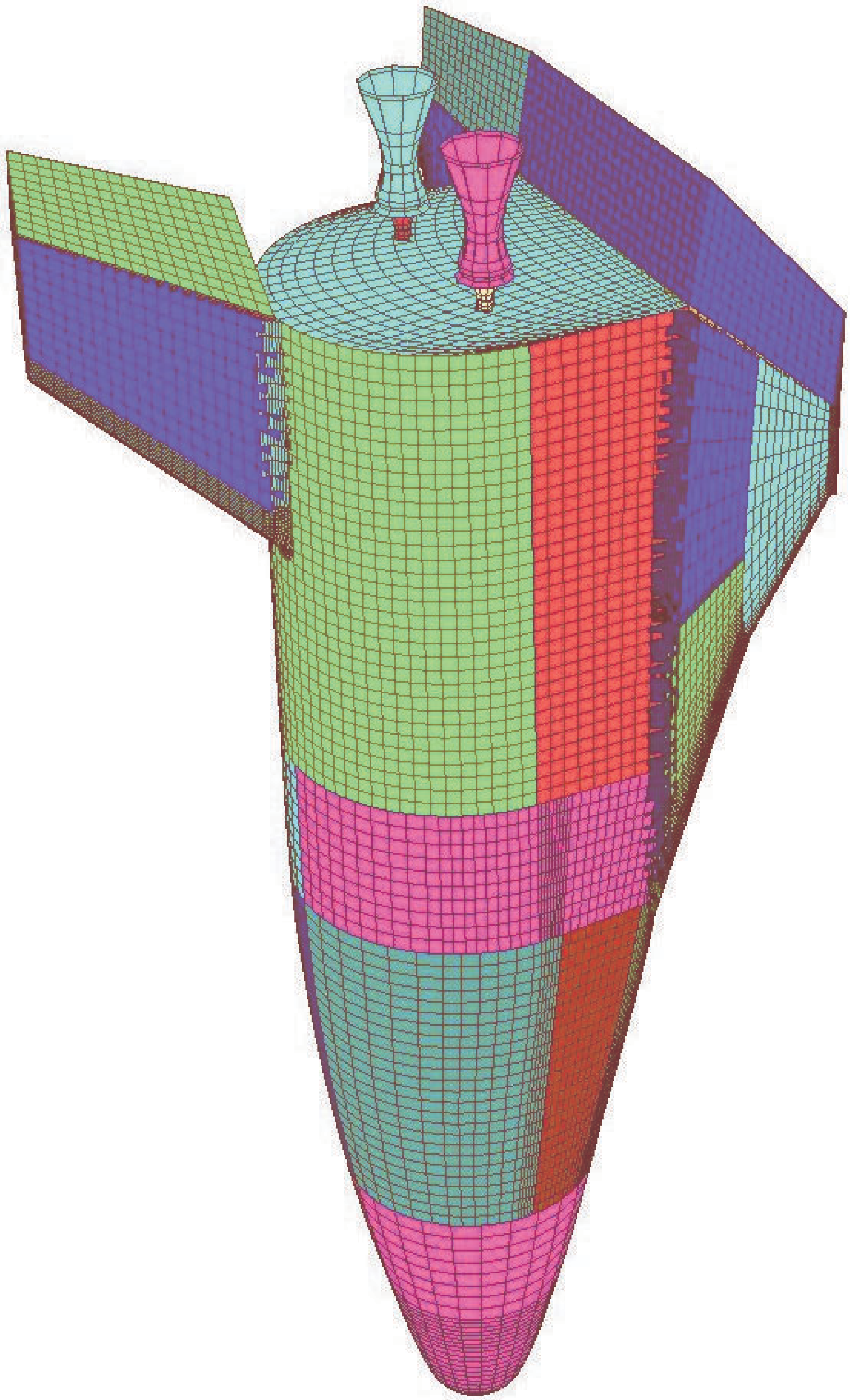


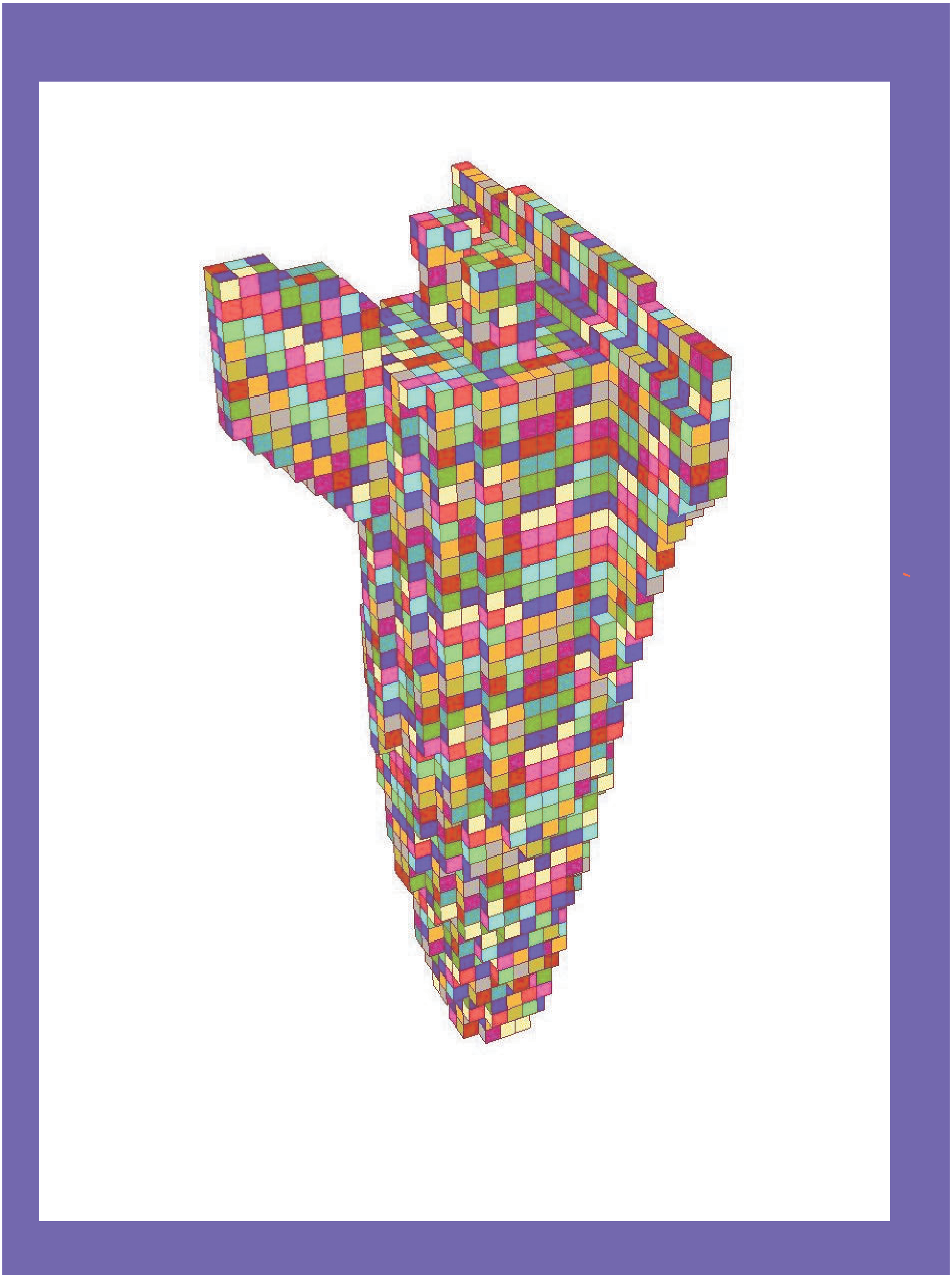
Cad Data Conversion

Closed Surface Construction

A novel algorithm that works globally

In order to construct a closed surface from the CAD patches, one needs to start from a closed surface, that is the CAD geometry (see next picture) is embedded within a closed surface. In order to construct a closed surface which is topologically similar similar to the original CAD geometry, the surrounding space is subdivided into uniform cubes. The size of a cube depends on the characteristic length scale of the geometry. Cubes containing data points , obtained from discretized CAD data, are marked. All other cubes are eliminated. Thus an initial closed surface is obtained as shown in the following picture. Finally, the idea is to shrink the surface like a balloon, and if close, to project. The simplest way of automatic shrinking is done by averaging neighboring data points. Finally, shrinking can be combined by projection, as shown i n the final picture. To deal with multiply connected domains, a curvature dependent shrinking algorithm is needed.



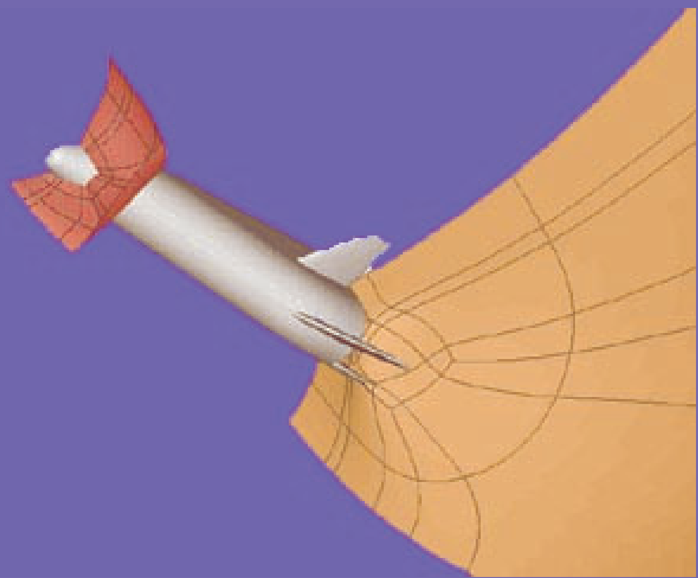


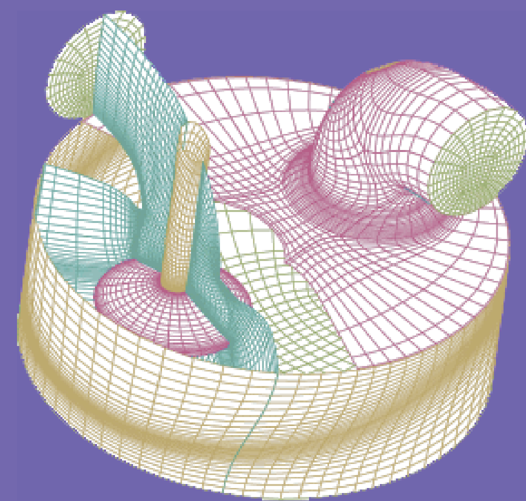
Complex Grids and their Features

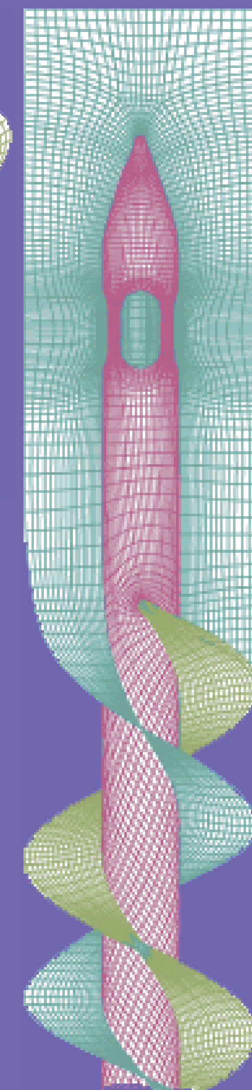
As an example we consider aerospace: In recent years there has been a slew of activities in designing advanced aerospace transportation vehicles both in the U.S. and Europe.

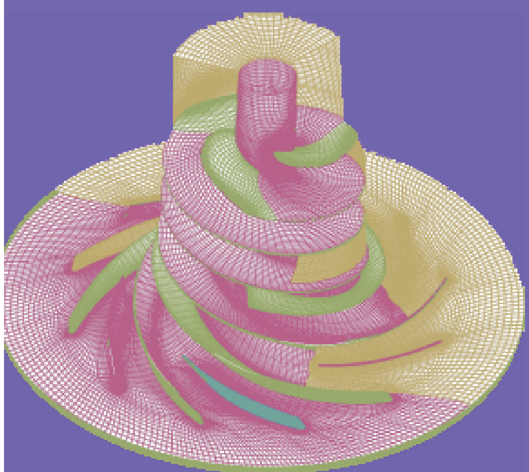
Related grid generation activities are in turbomachinery.

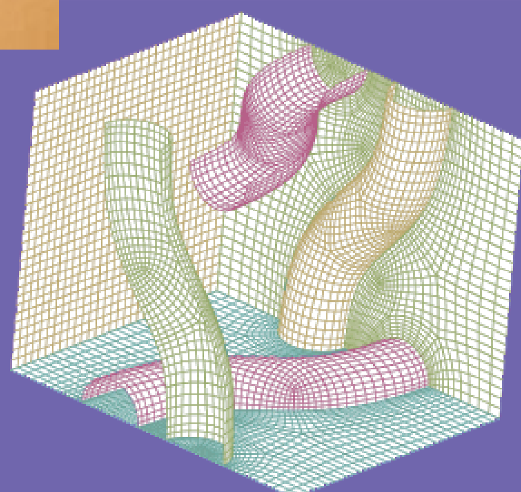
Space Activities and Spin -Offs





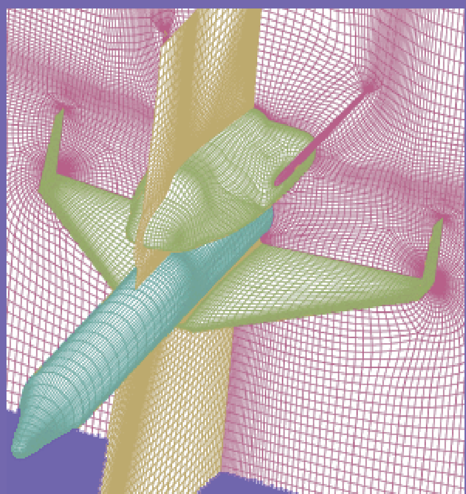




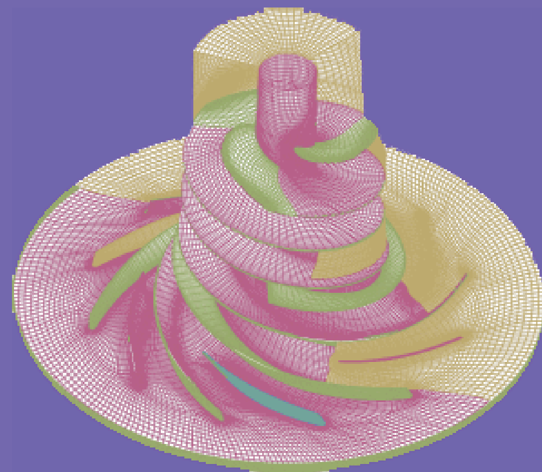


GridPro Grid Gallery

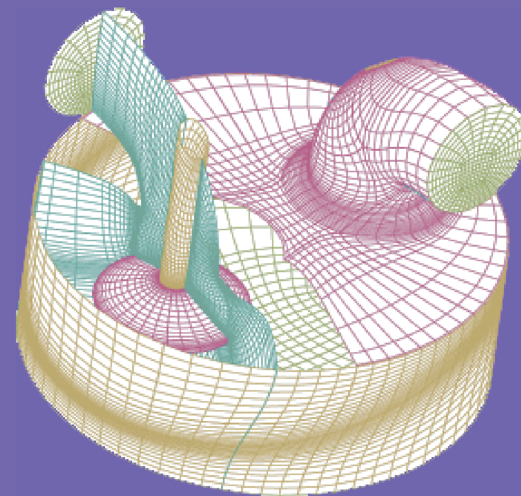
Aerospace



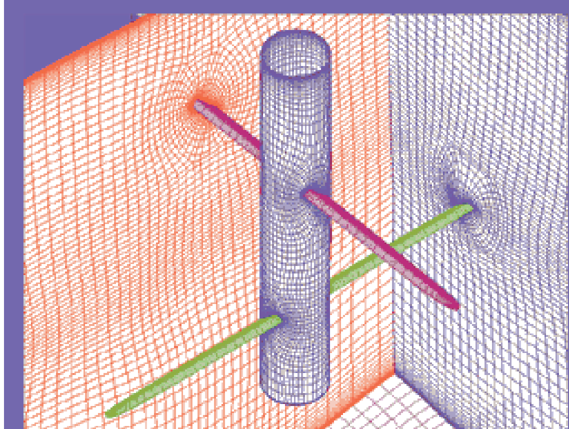
Turbomachinery



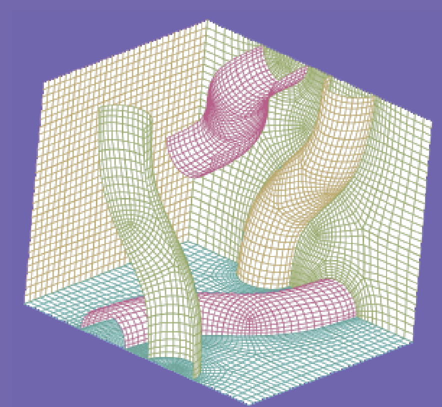
Automotive



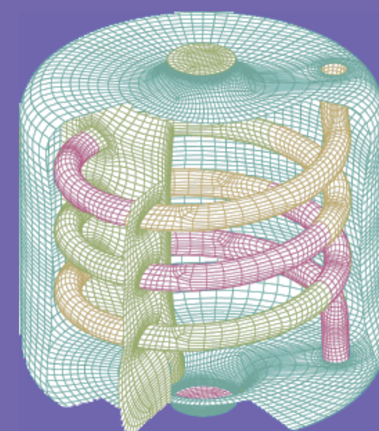
Chemical

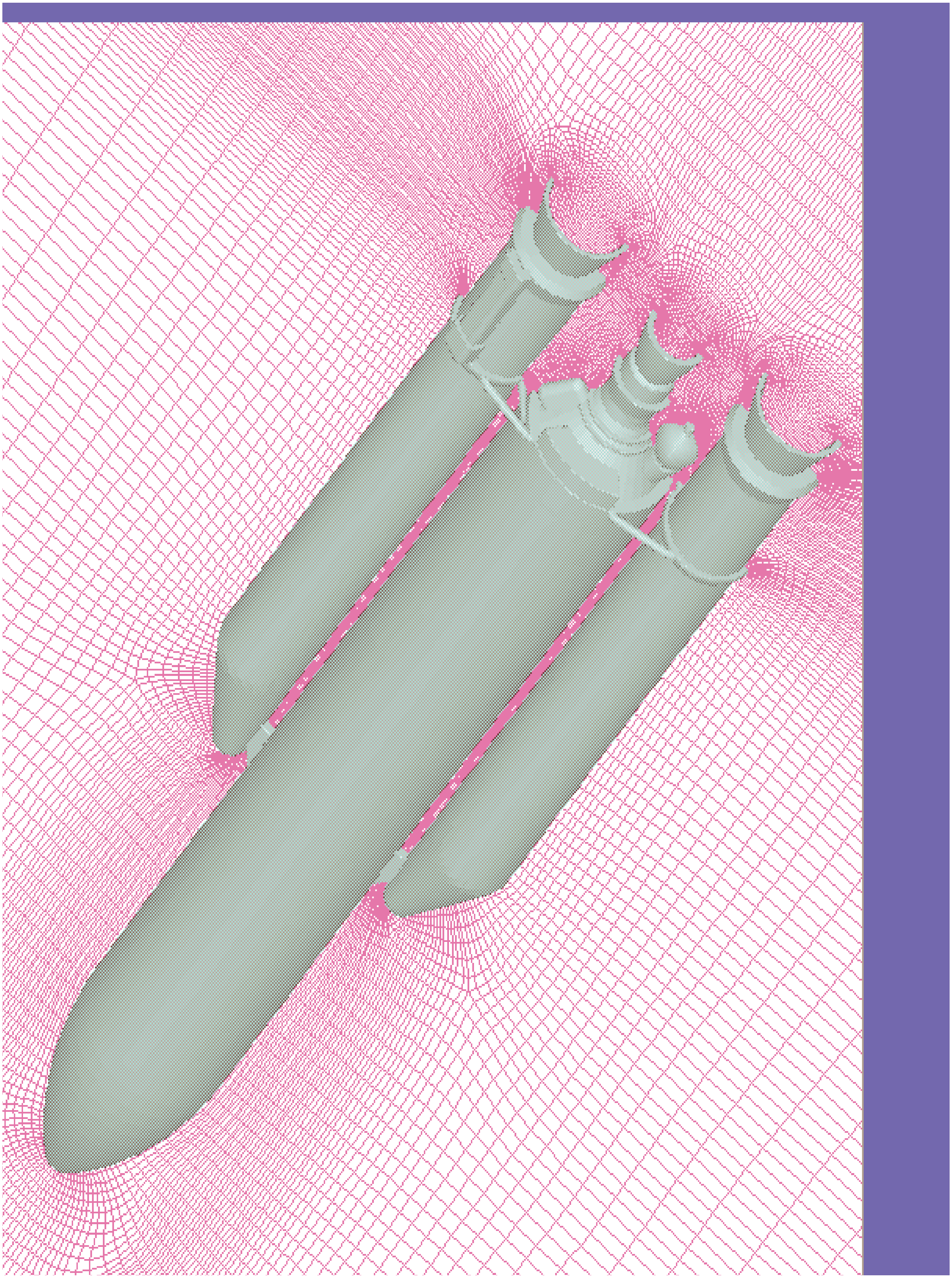


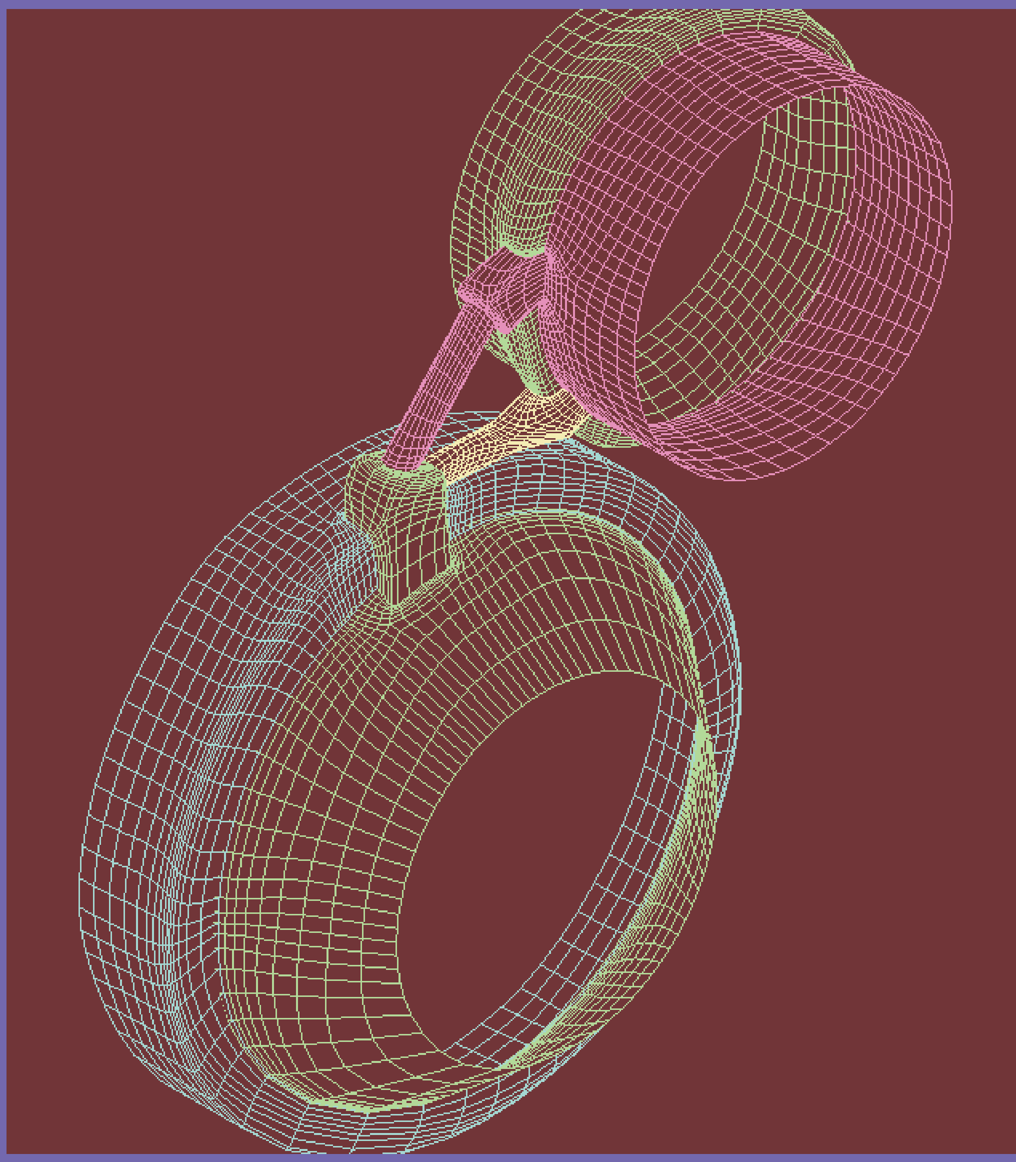
Fibre

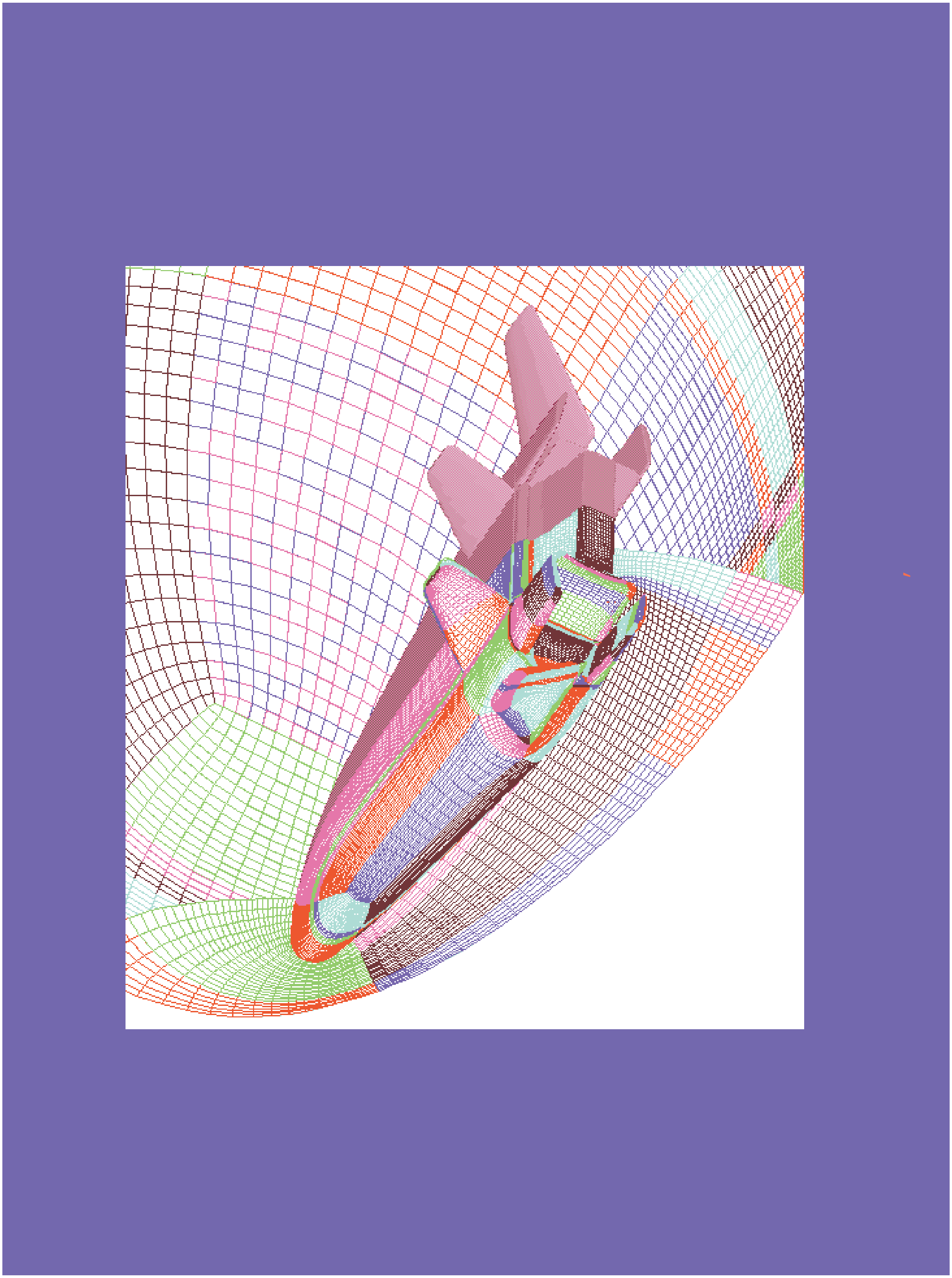


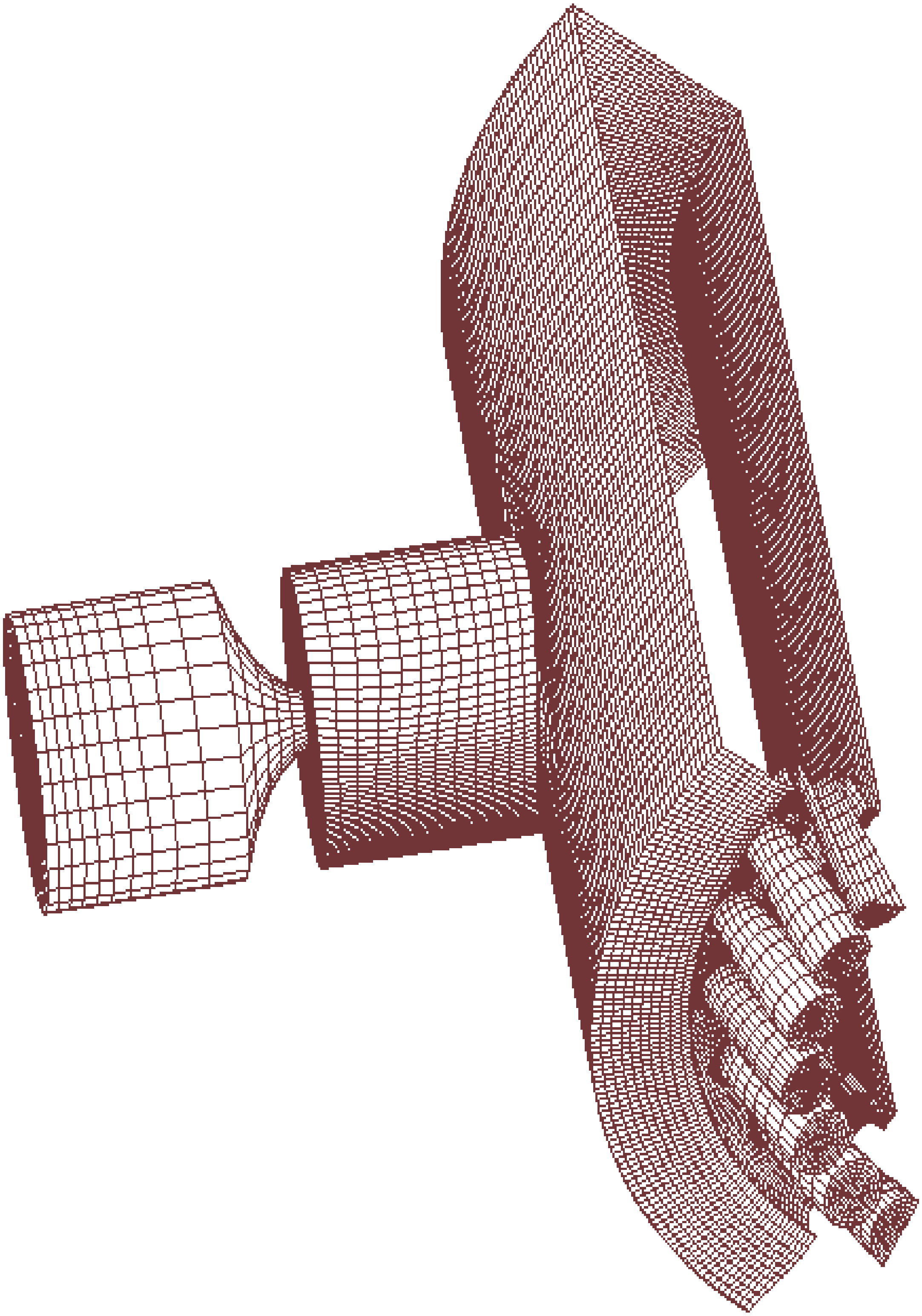
Others



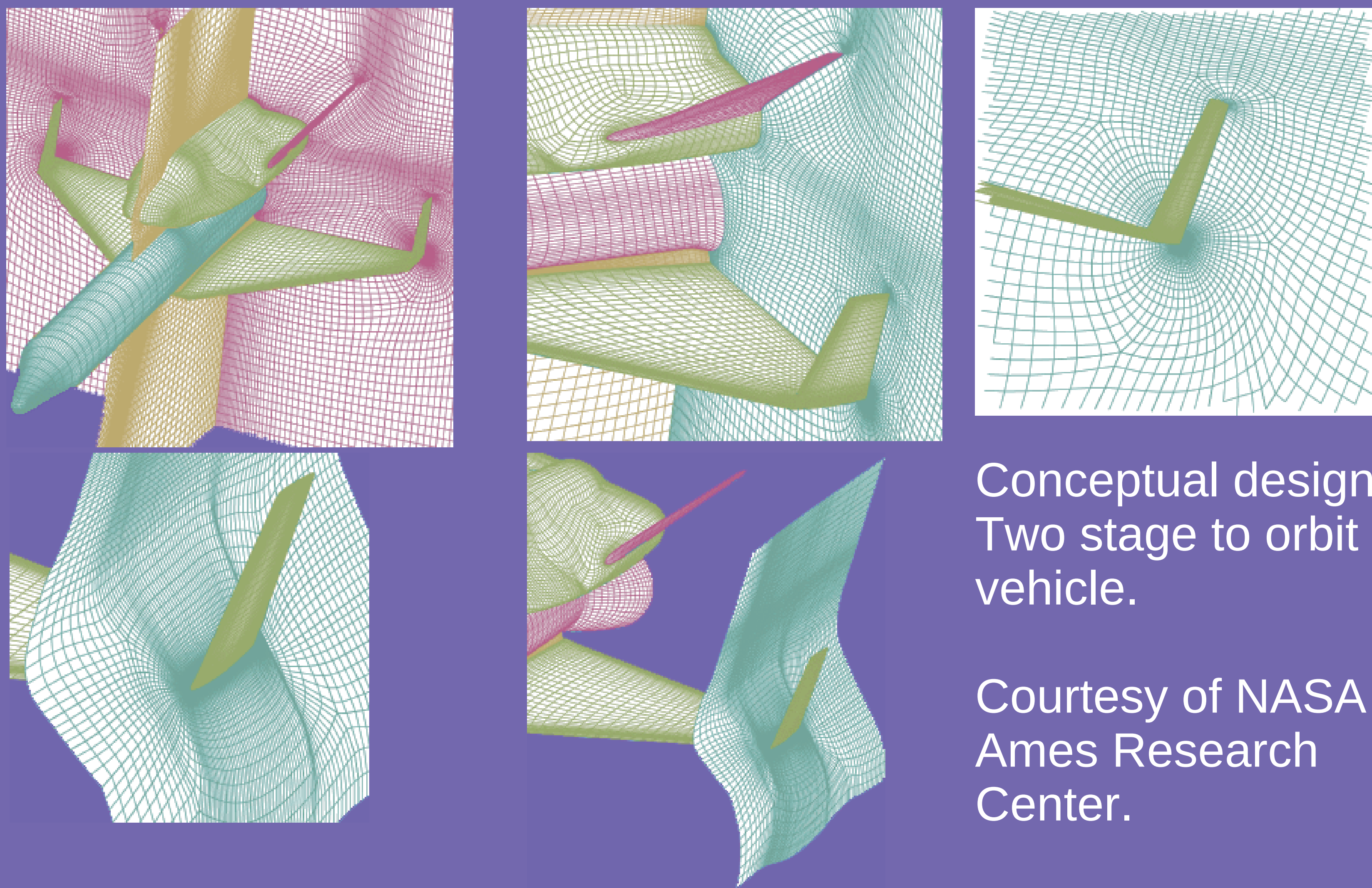








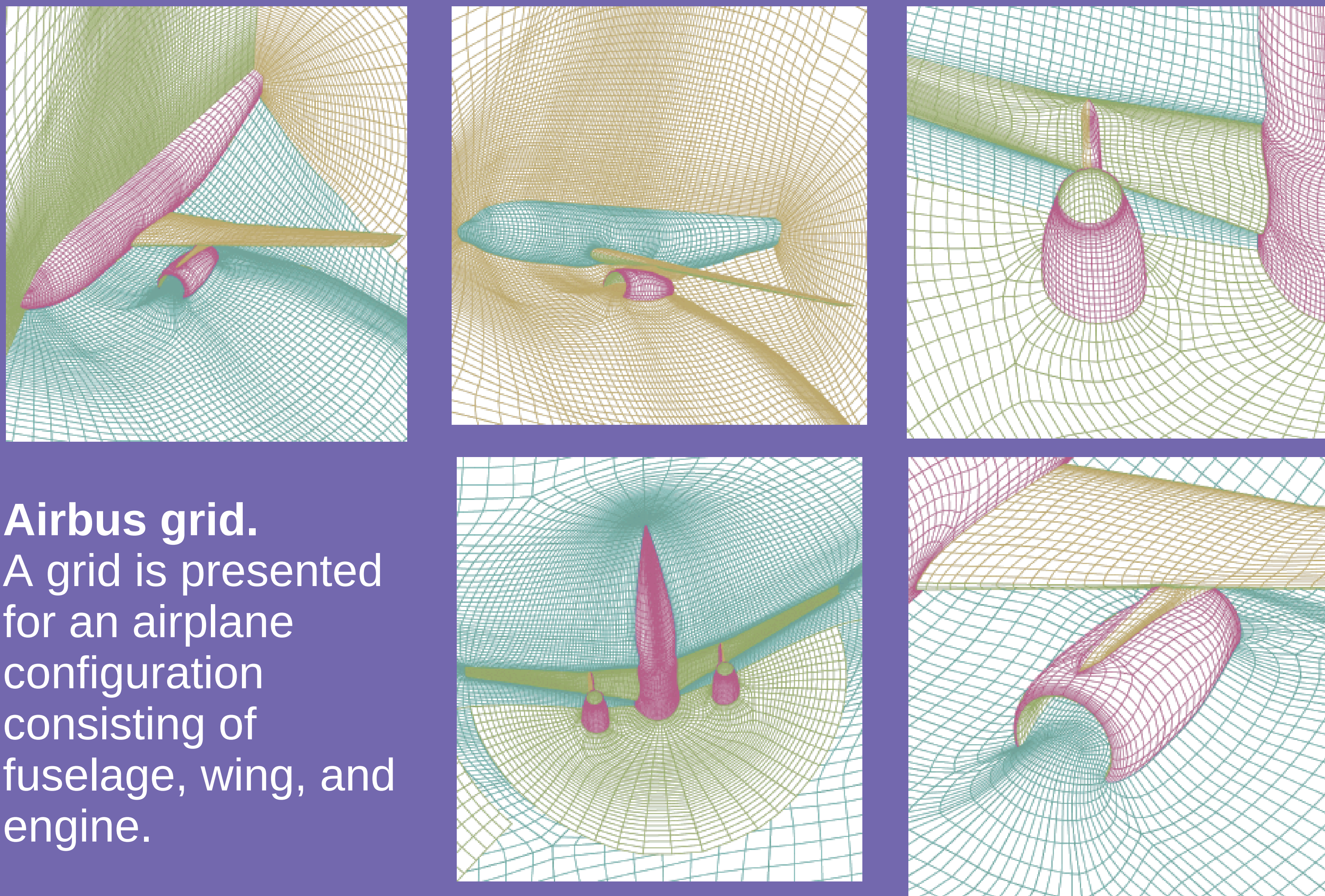
Two-Stage to Orbit Vehicle



Conceptual design of Two stage to orbit vehicle.

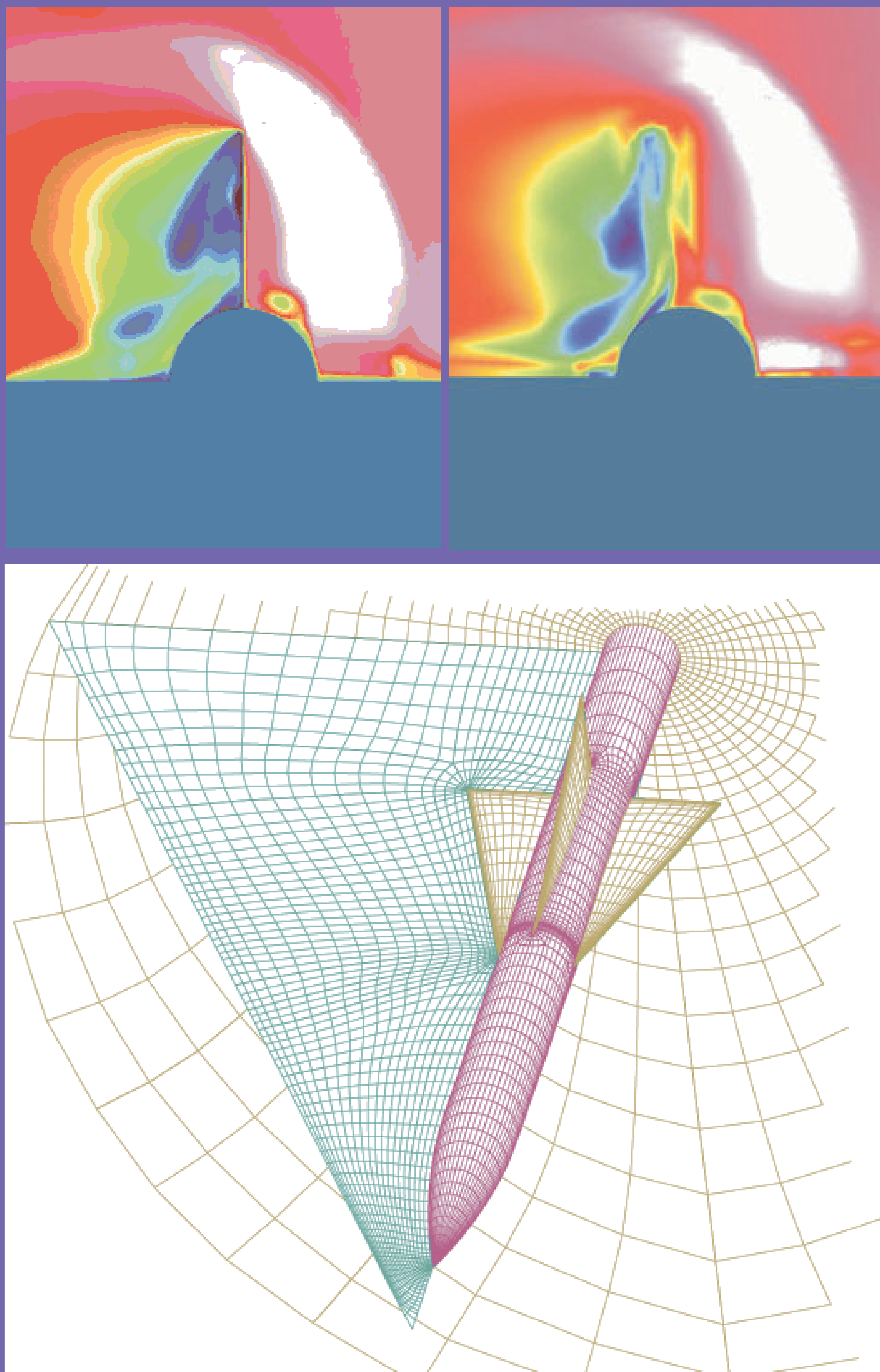
Courtesy of NASA Ames Research Center.

Airbus Grid

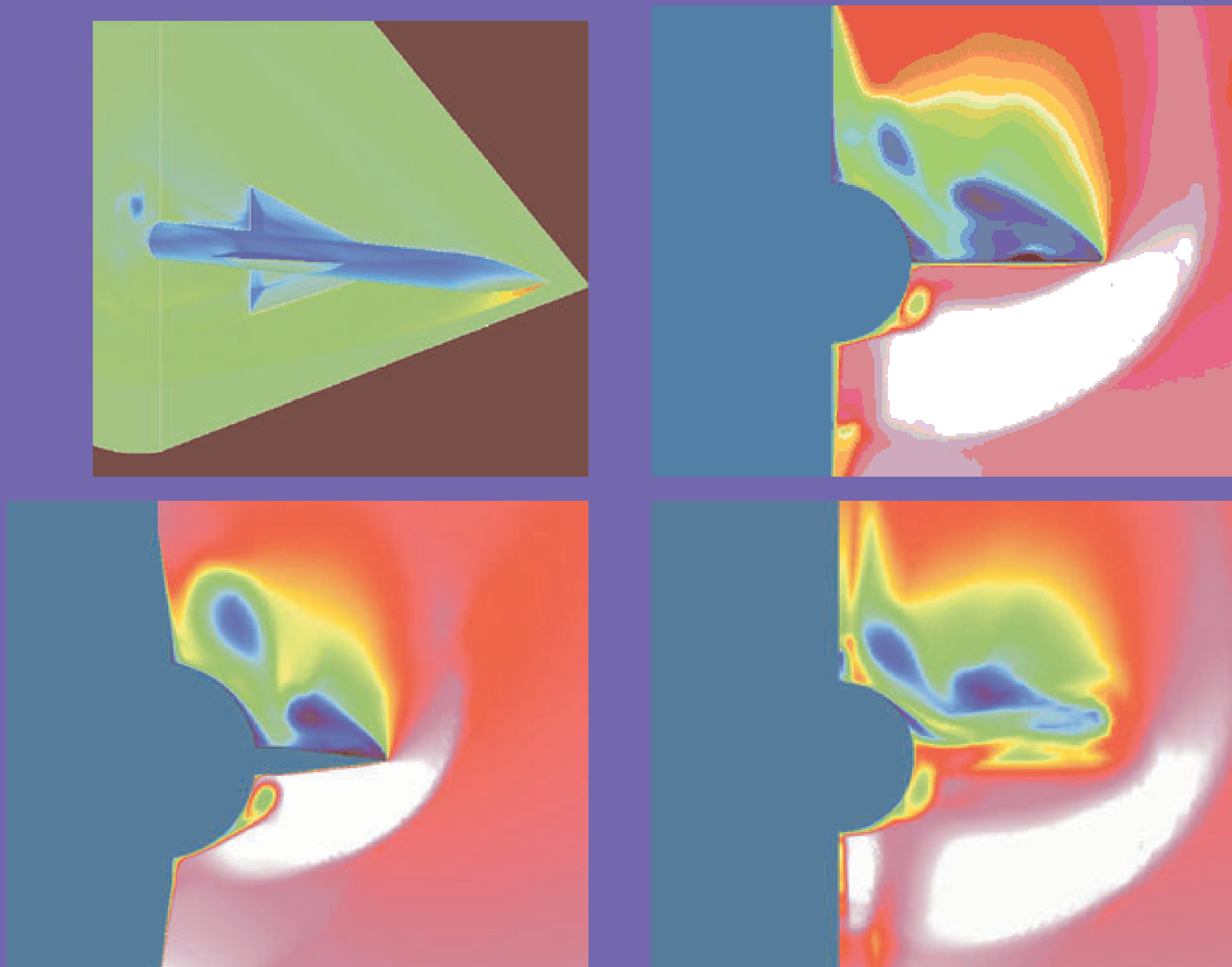


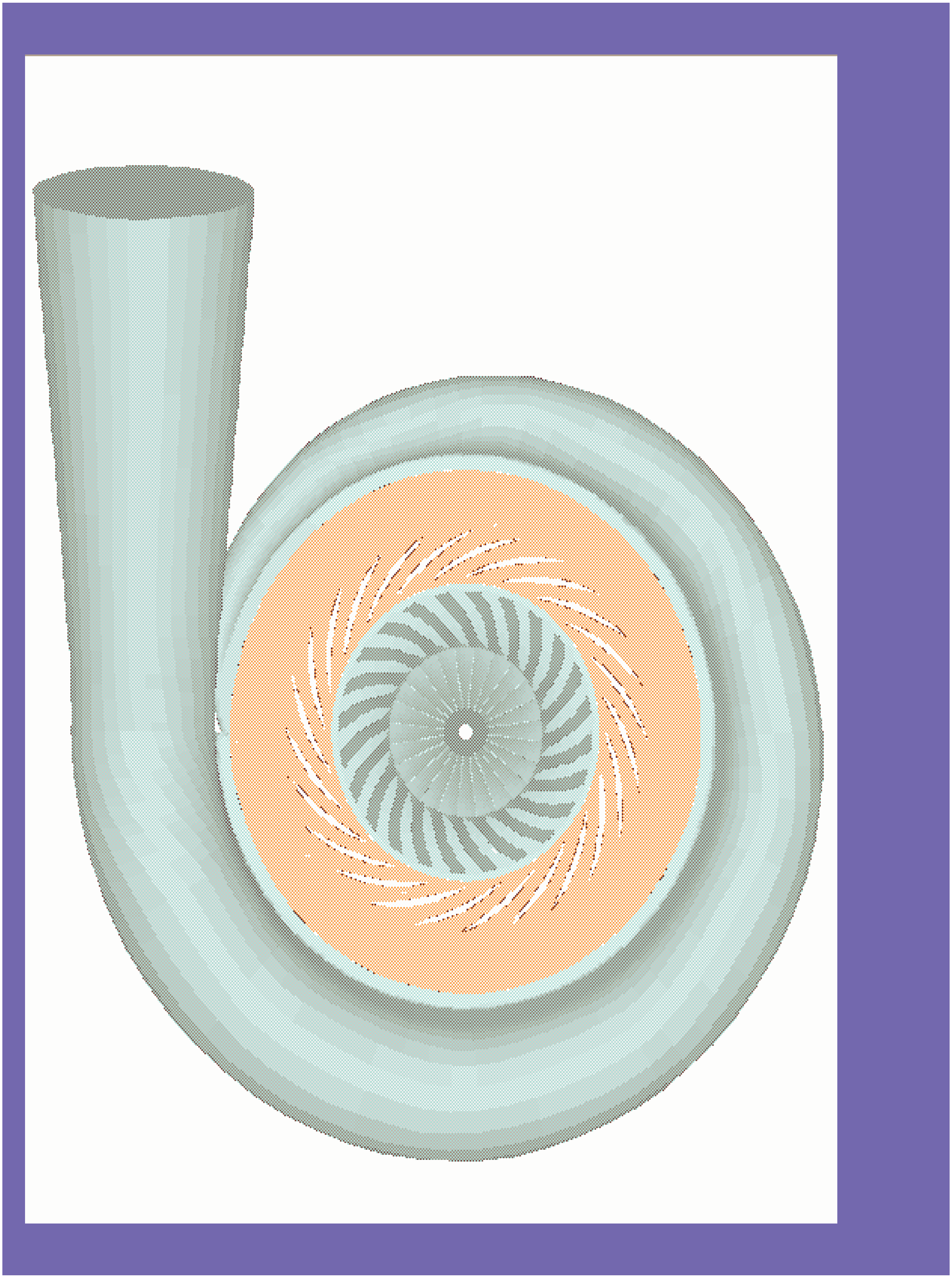
Airbus grid.
A grid is presented for an airplane configuration consisting of fuselage, wing, and engine.

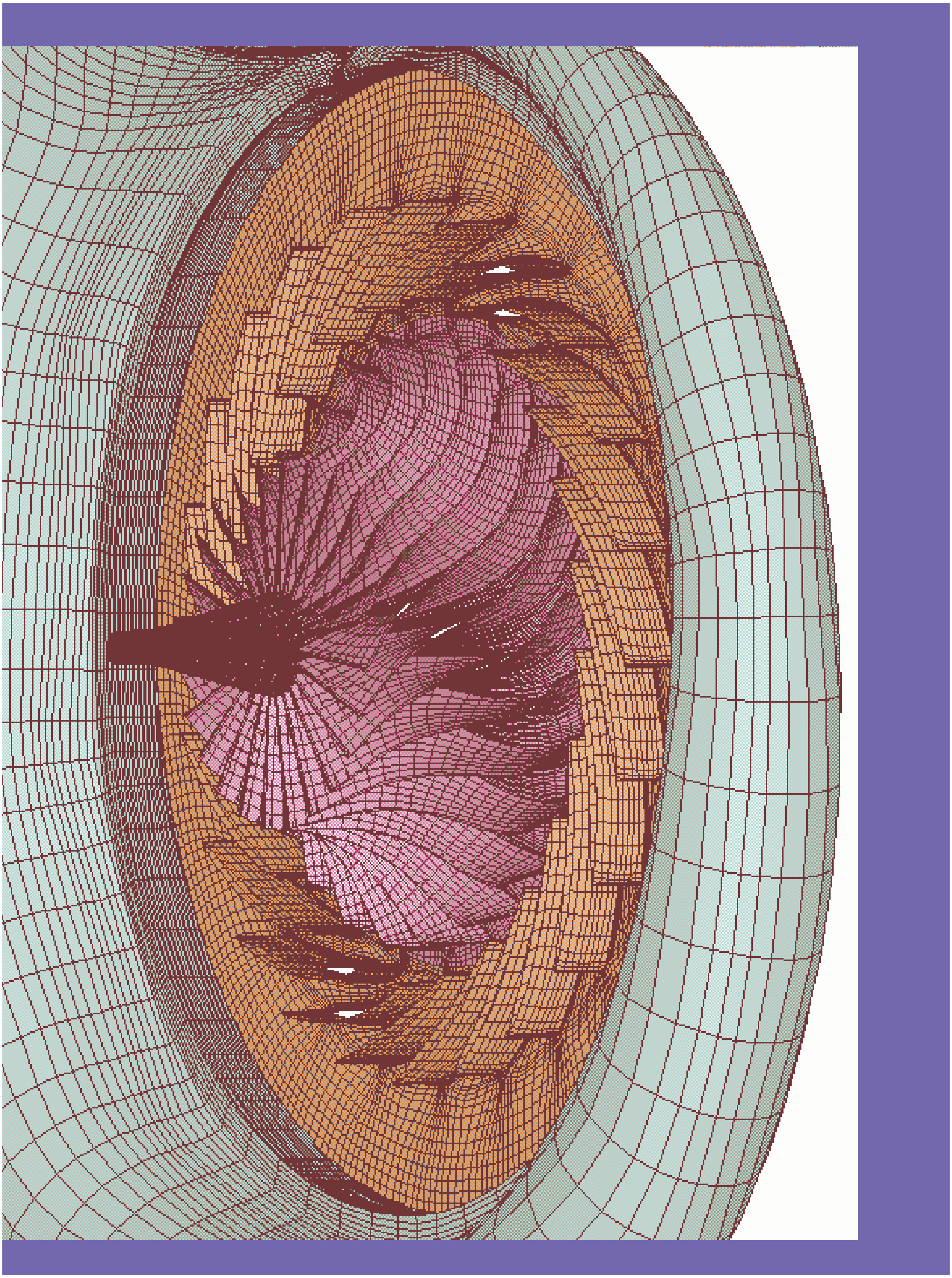
Overlap Grids in GridPro



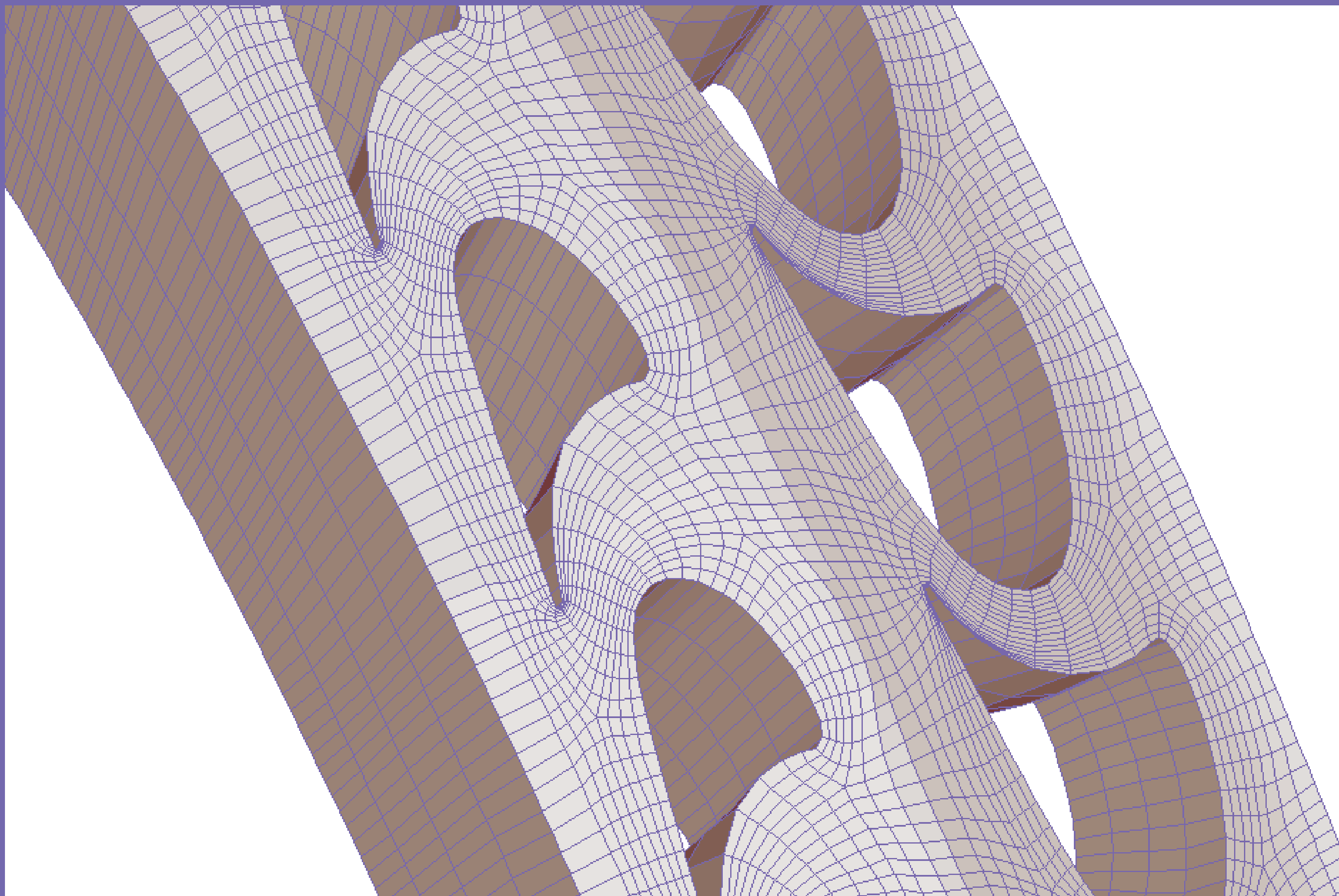
CFD Solution for Generic Missile



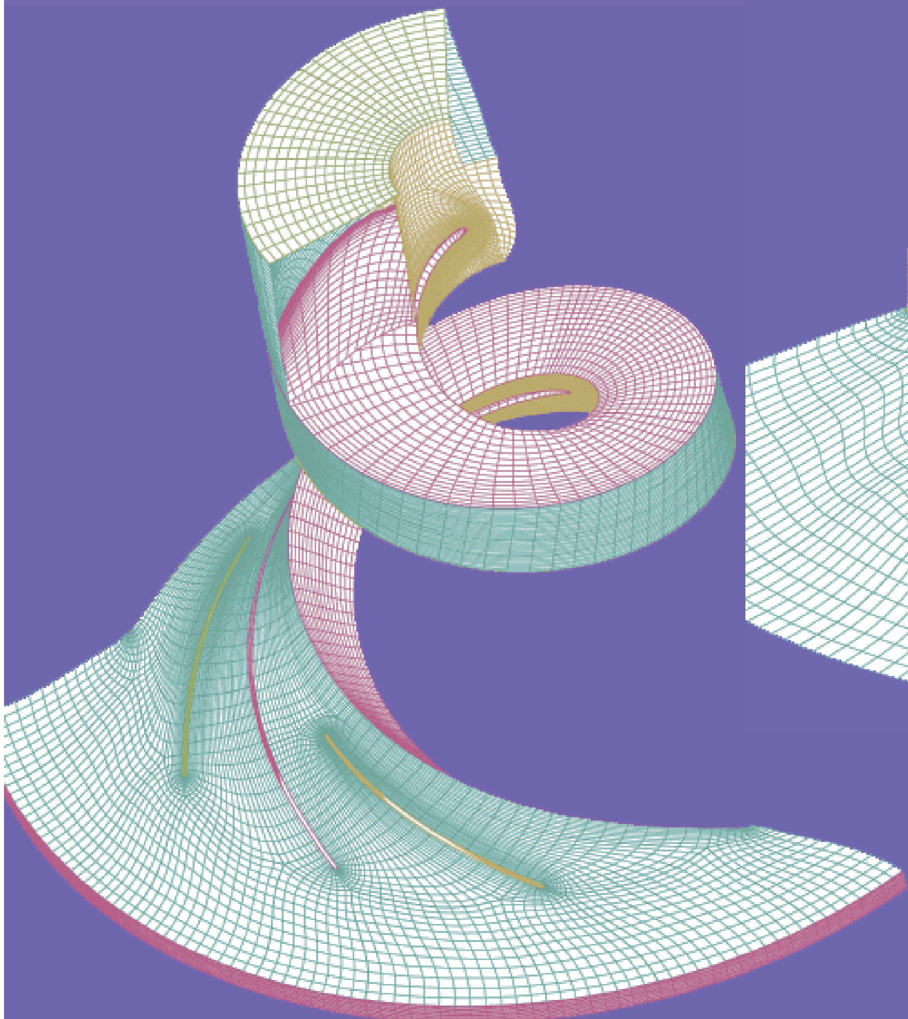


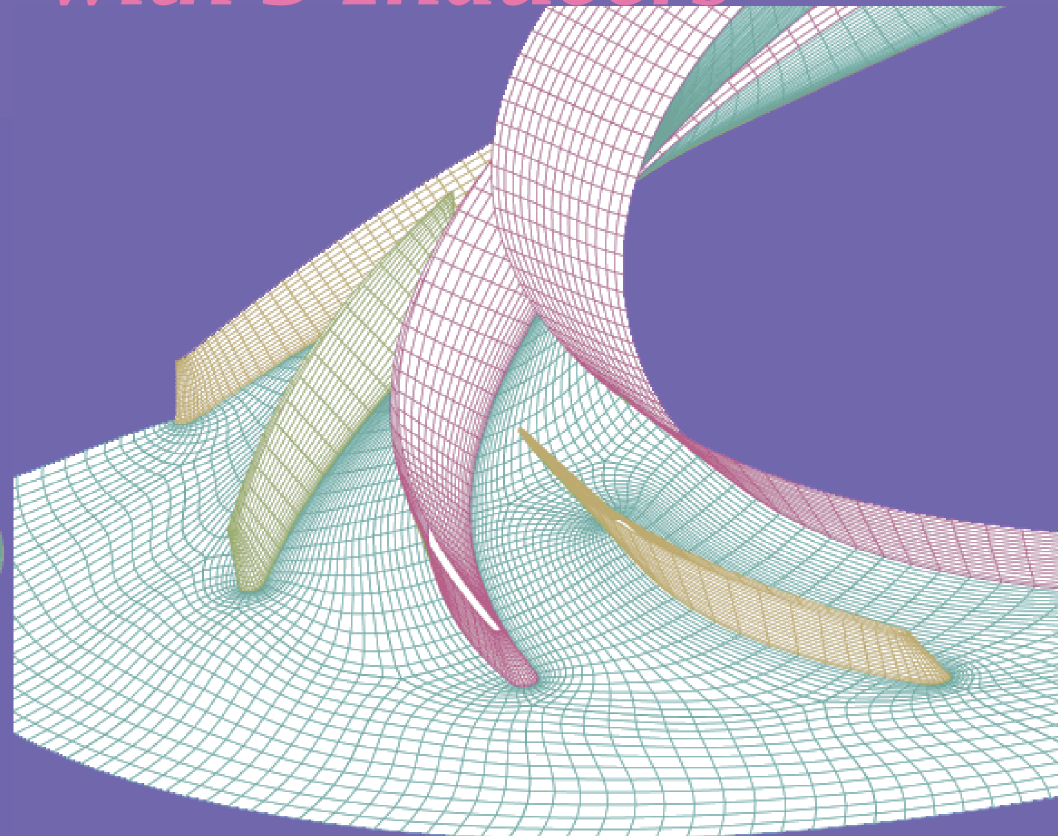


Blade Rows after Welding



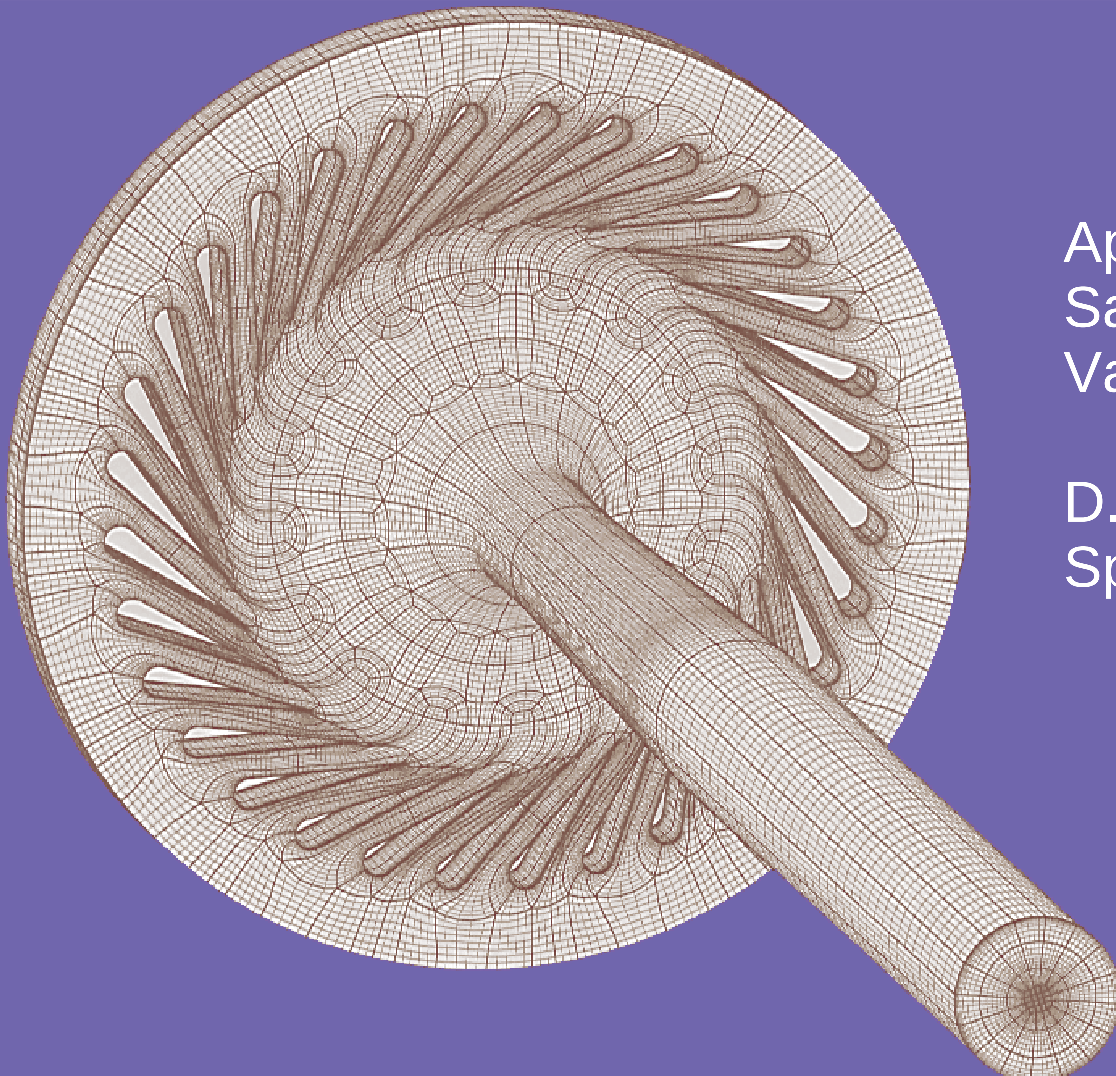
Impeller with 3 Inducers





120 degrees periodic piece with one inducer and 3 splitters.
The geometry for periodic boundaries is automatically determined along with the grid.

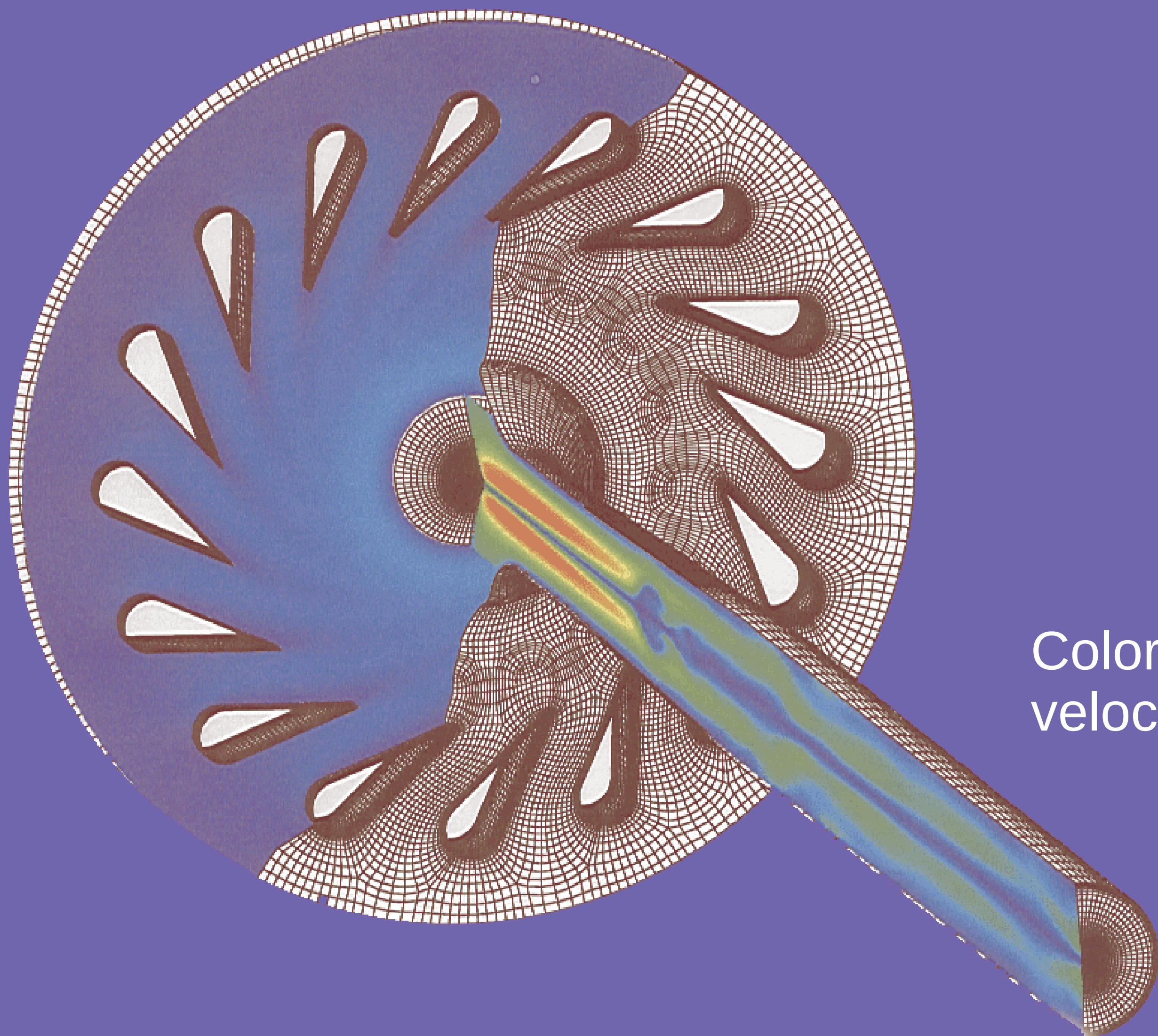
3D Laminar Vortex Breakdown



Apparatus of Sarpkaya/Leibovich (16 Vanes, 38 deg. Angle)

D. O. Snyder, R. E. Spall

3D Laminar Vortex Breakdown



Colors indicate the flow velocity.

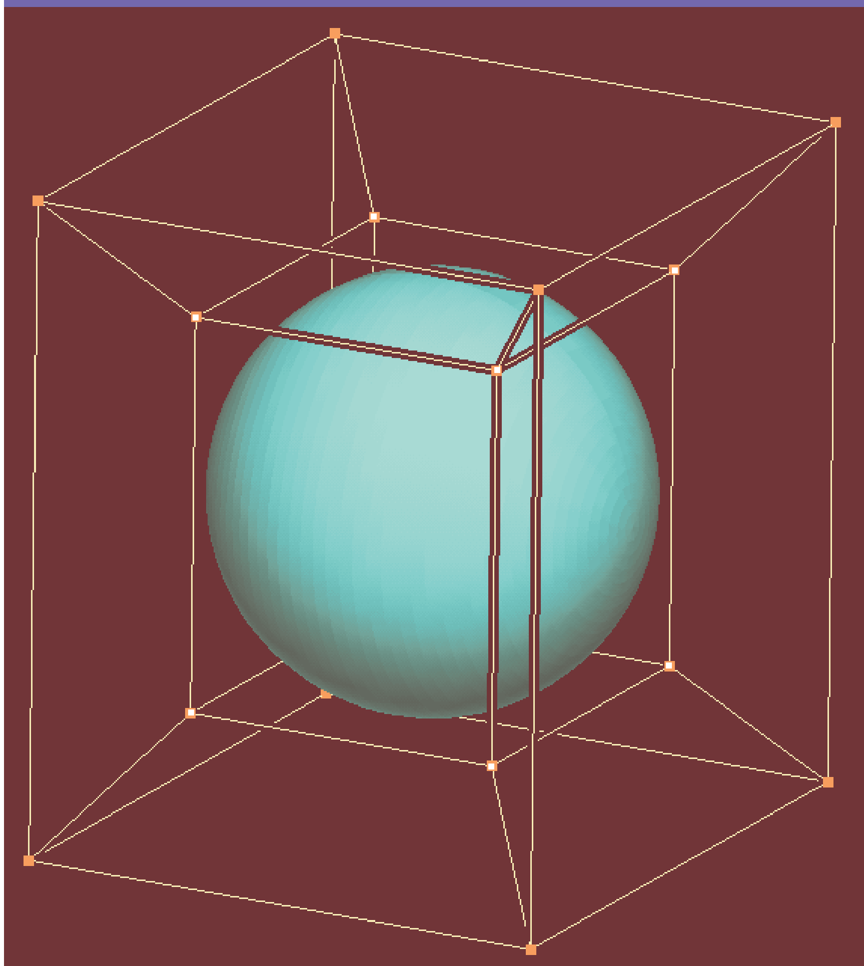
Topology Input Language
Grid Generation for Complex Geometries

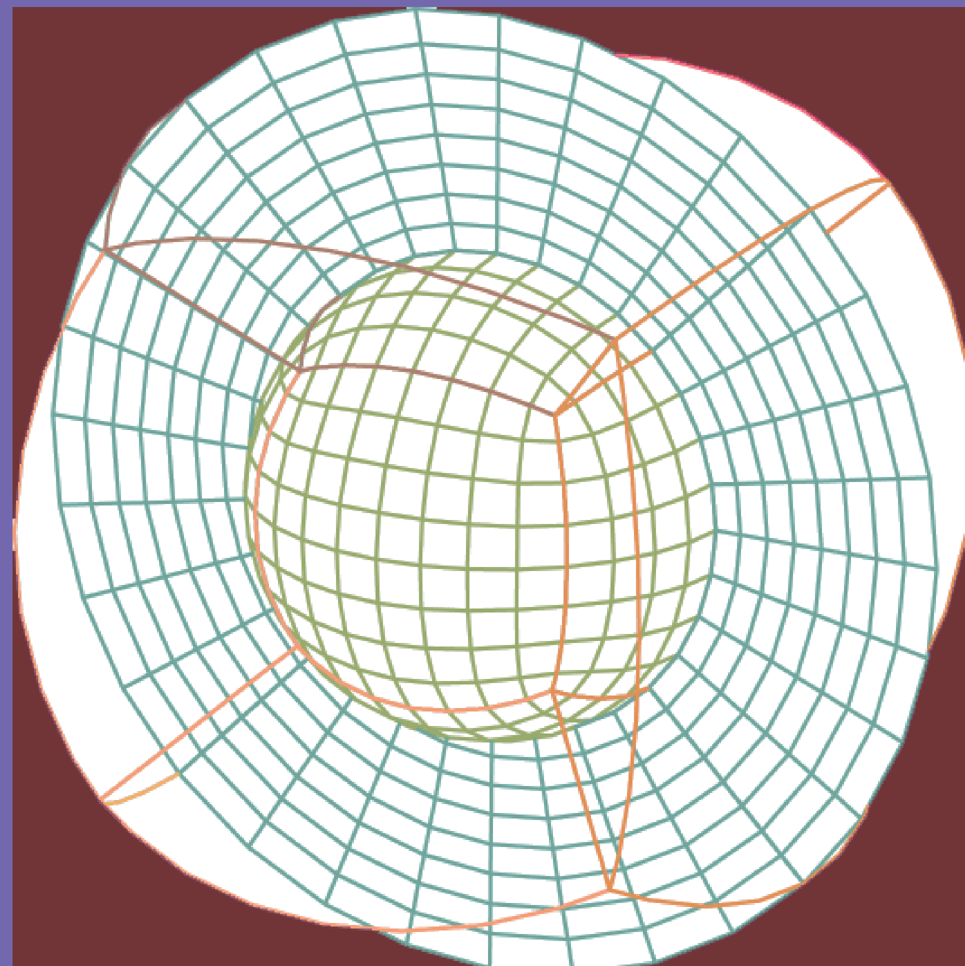
The basic strategy for obtaining a complex (multiblock) grid is the separation of topology space and physical space.

Once the topology is specified, either interactively by, for instance, the GridPro AZ-manager or writing GridPro TIL code (Topology Input Language, see Ref. 1), surface and volume grids are generated completely automatic.

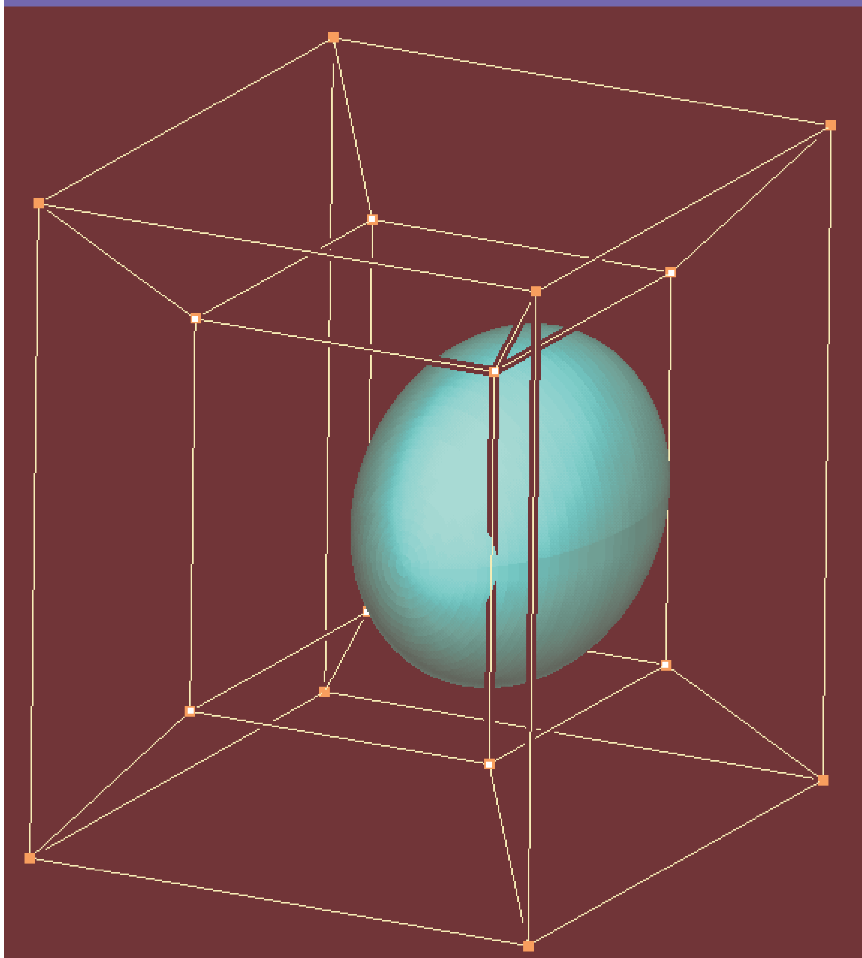
In the following a relatively simple grid example is presented. It will be seen that hundreds or thousands of blocks of different size are generated, necessitating a special parallelization strategy

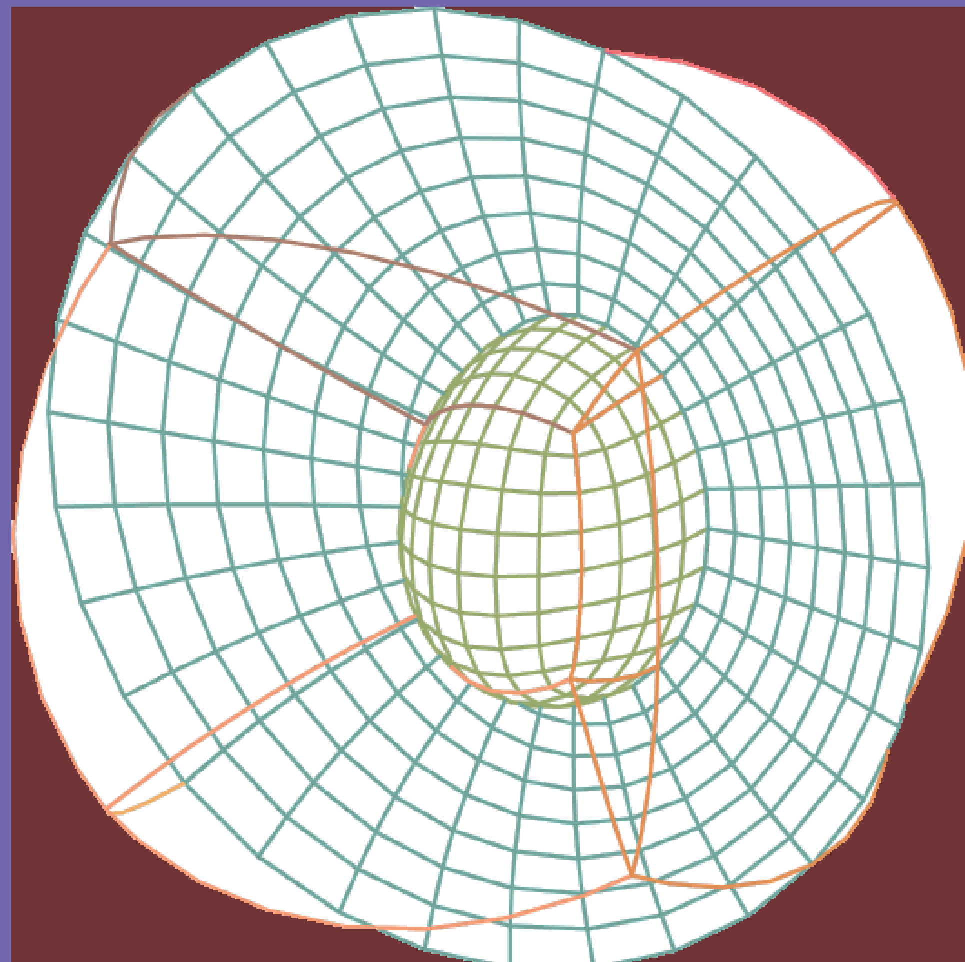
Design Variation (Initial Configuration)





Design Variation (Modified Configuration)





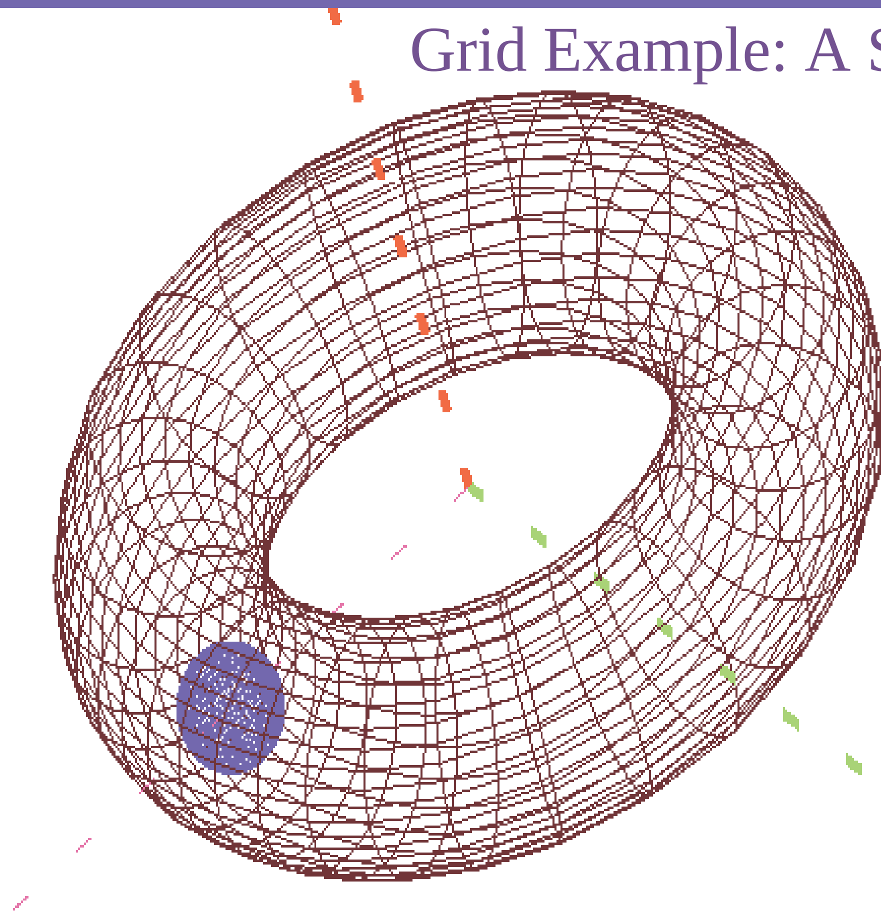
TIL Code Sphere in Box Example

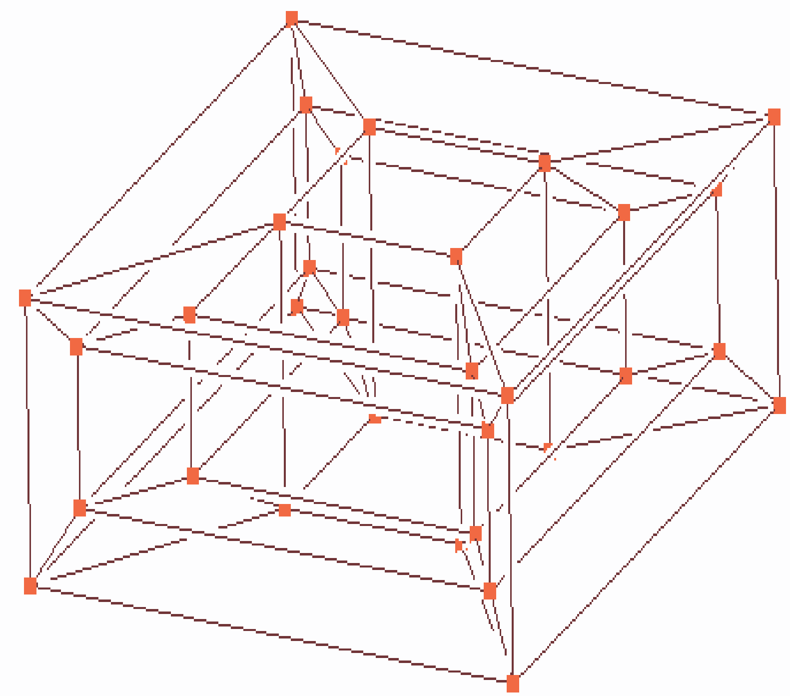
```
SET GRIDDEN 8
SET DISPLAY.SURF ON

COMPONENT main()
BEGIN
  INPUT 1 surf(sOUT (0..1));
  INPUT 2 corn(sIN (1:1..2),cIN (-16));
END

COMPONENT surf()
BEGIN
s    0 -ellip (1 1 1)  -o     ;#TIL:1:1
s    1 -ellip (3.33333333333 2.5 2)  -t 0.2 0 0     ;#GI:0:2
END
```

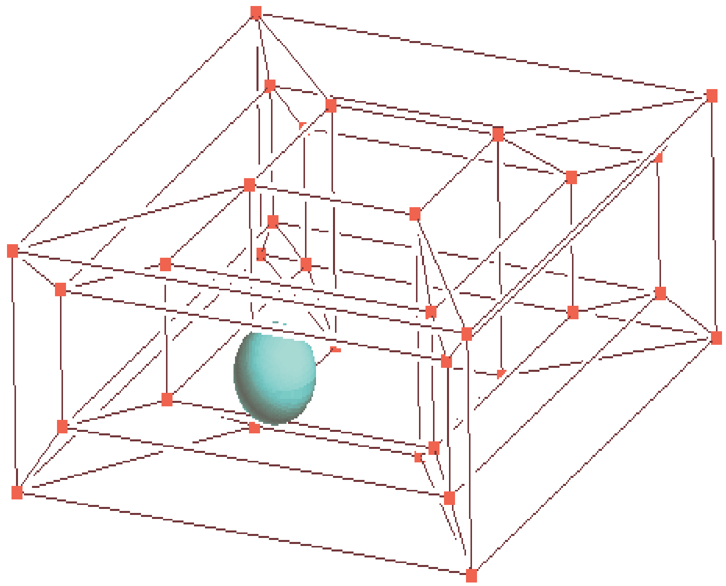
Grid Example: A Sphere in a Torus

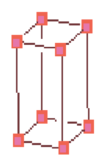




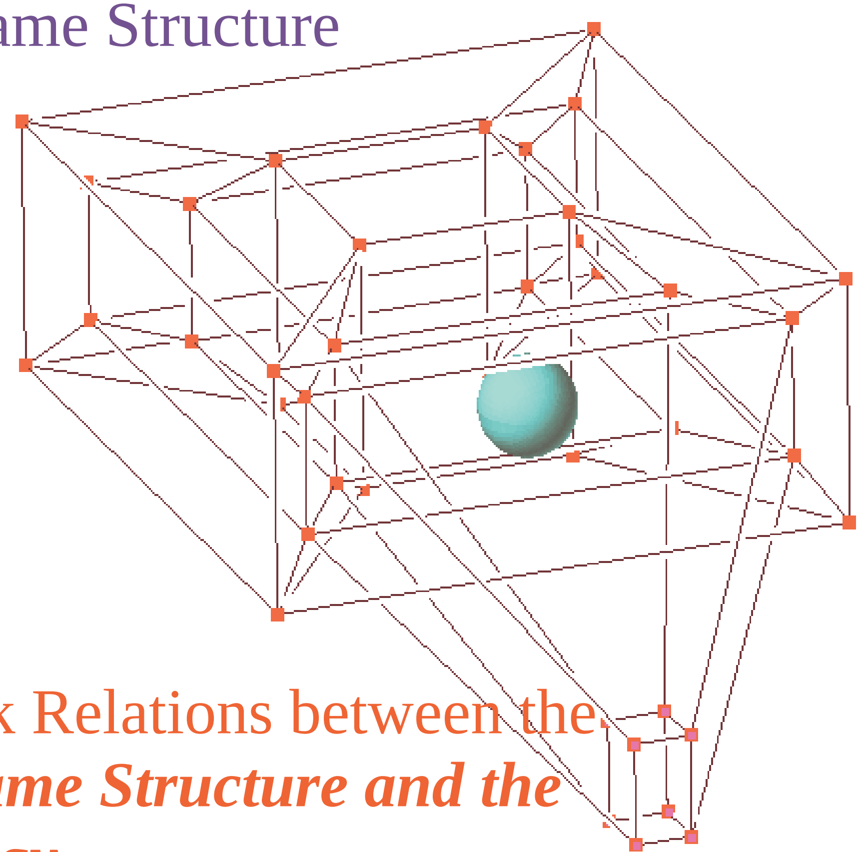
The solution domain is **defined between the torus and the sphere.**
A topology wireframe is constructed as shown.

Local Wireframe Structure



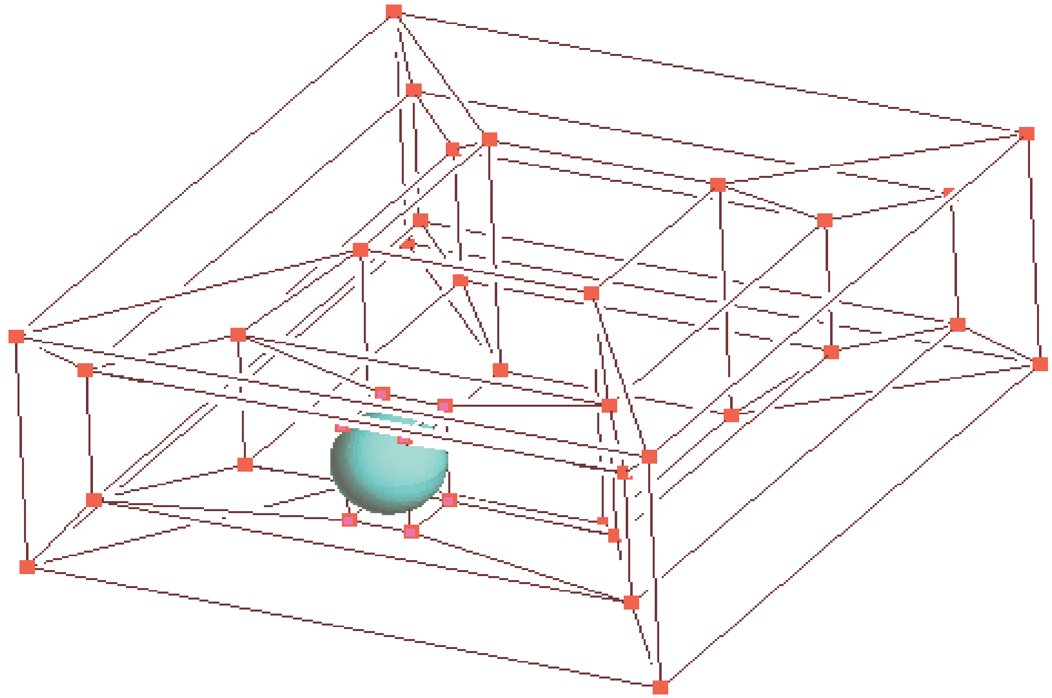


Generate Link Relations between the Main ***Wireframe Structure and the Local Topology***



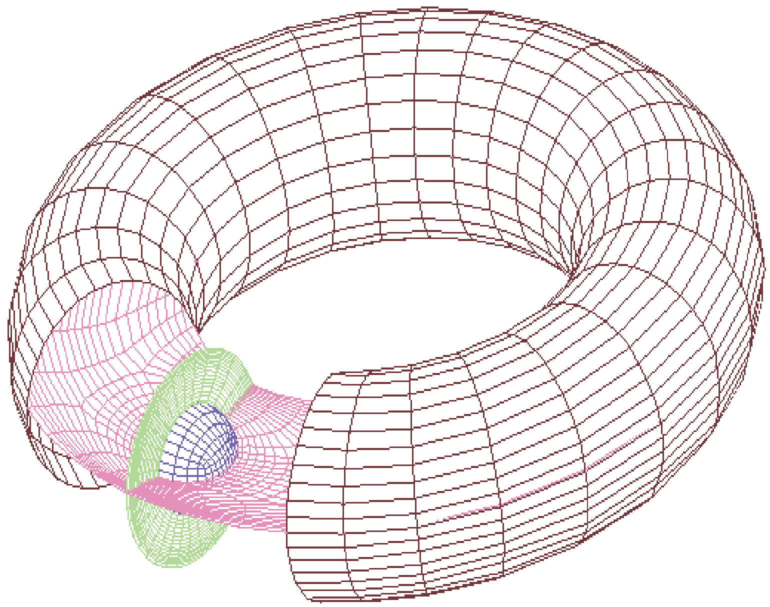
he local wireframe topology is built outside the main wireframe structure nd then linked to the main wireframe topology. The sphere gives the real osition of the local topology.

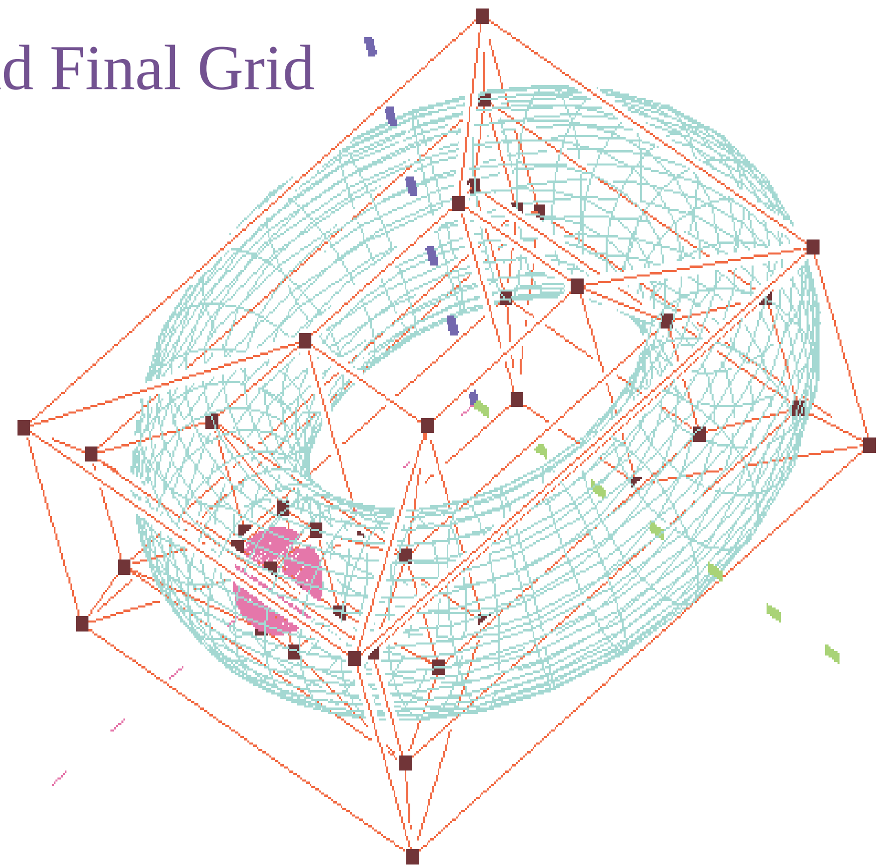
Position the Local Topology



The local wireframe topology is placed in its real position.

Grid Topology and Final Grid





The final grid is generated based on the given topology.

TIL Code Ball in Torus Example

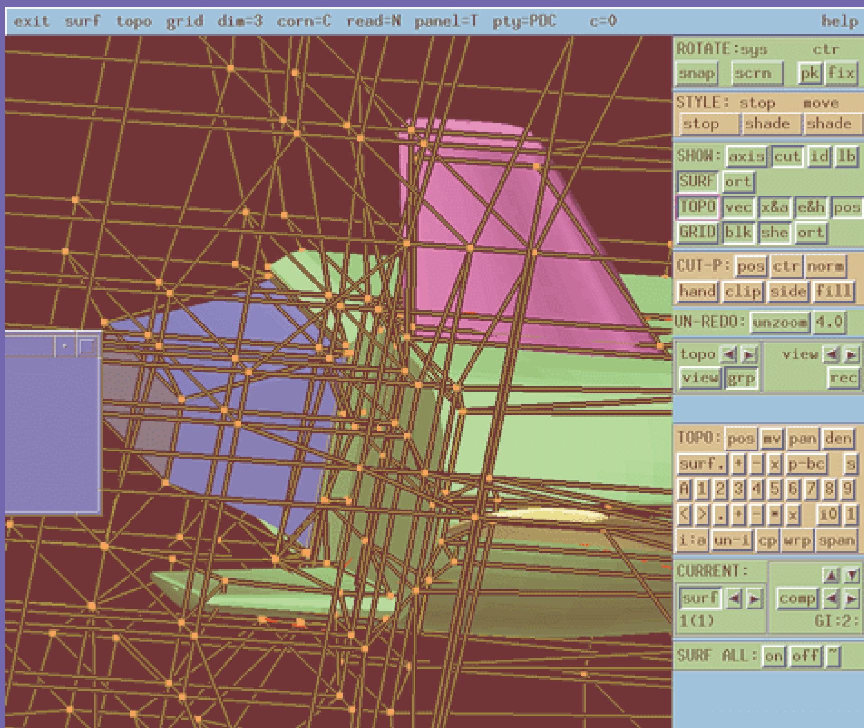
```
#TIL code Ball in Torus
SET GRIDDEN 8
SET DISPLAY.SURF ON

COMPONENT main()
BEGIN
   s 1 -implic "torus.h"; #analytical surface for torus
   INPUT 1 torus(sIN (1),cOUT(1:1..4, 2:1..4));
   INPUT 2 (0 0 1.5)*ball(cIN (1:1..8);
END
#surface description file for torus:
define FUNCUya=sqrt(y*y+z*z)-1.5, 1-(ya*ya+x*x)*4
```

TIL Code Ball in Torus Example

```
COMPONENT ball(cIN c[1..8])
BEGIN
  s 1 -ellip(4 4 4);
  c 1 -0.35  0.35 -0.35 -s 1 -L c:1;
  c 2 -0.35  0.35  0.35 -s 1 -L c:2 1;
  . . .
  c 8  0.35 -0.35 -0.35 -s 1 -L c:8 4 7 5;
END
```

AZ-Manager

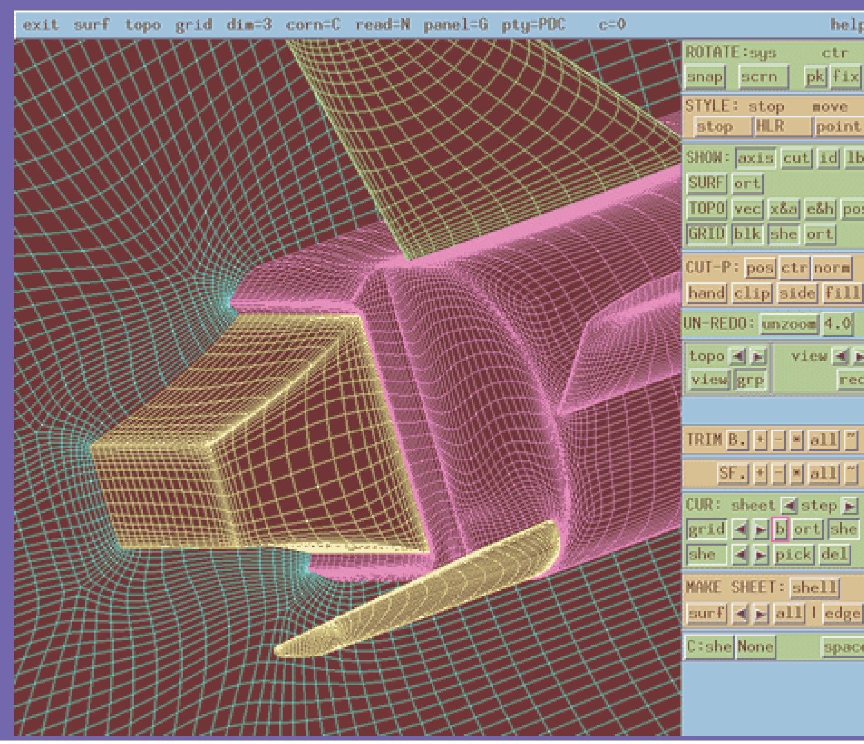


Topology Panel

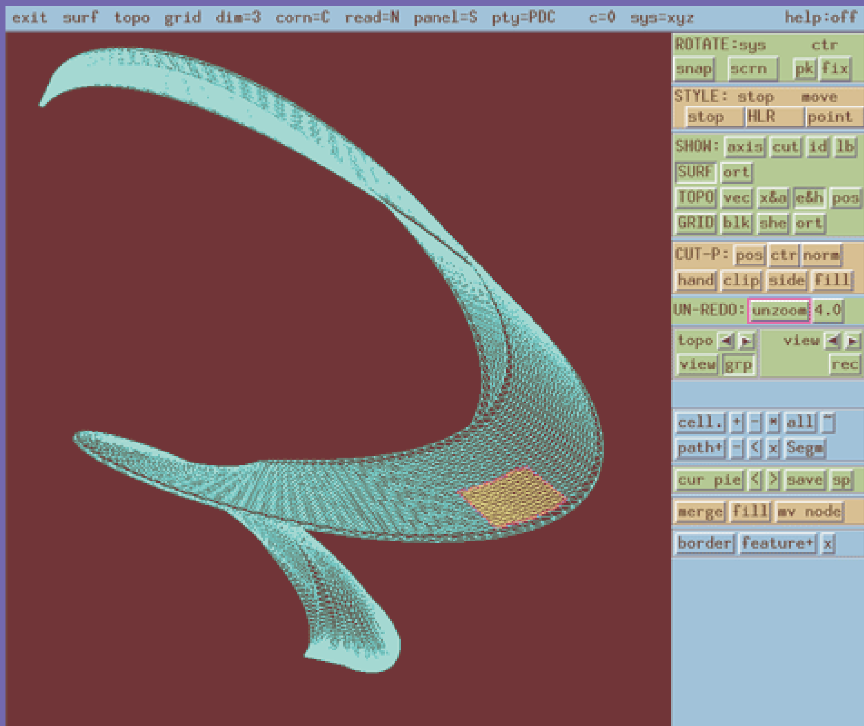
The Topology Builder Panel is used to interactively select surfaces and to construct the block topology, i.e. corners and edges. Surfaces can be imported using several different standards, but build-in surfaces can also be used.

Grid Panel

The Grid Viewer Panel is used to view the grid in various ways and thereby check its quality by visual inspection. Furthermore, global grid quality can be checked by applying the utility qchk to the grid data file.



AZ-Manager

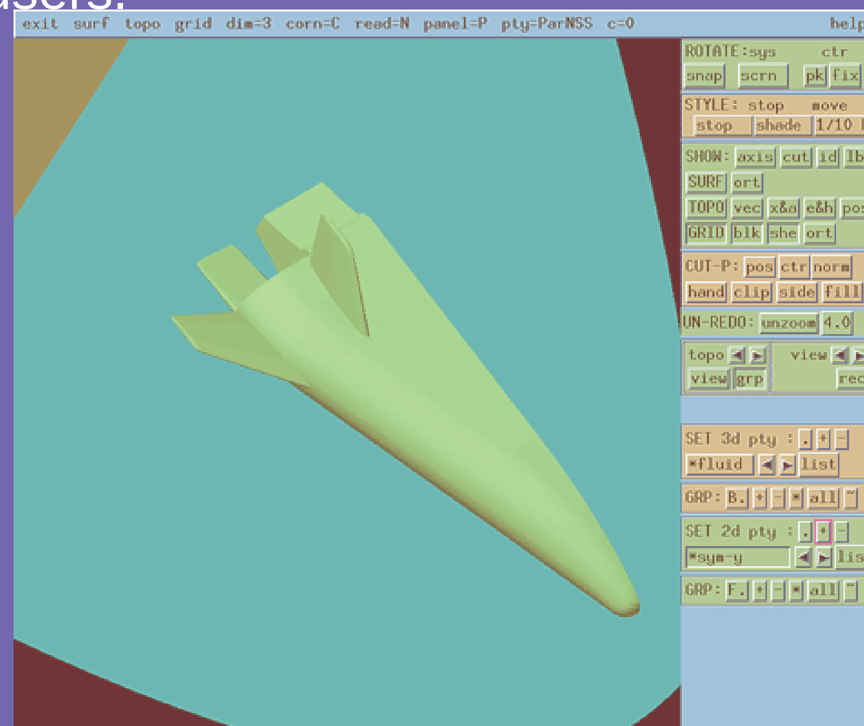


Mini Cad Panel

The miniCAD Panel is used to repair surfaces or modify surfaces for grid generation.

Property Setter Panel

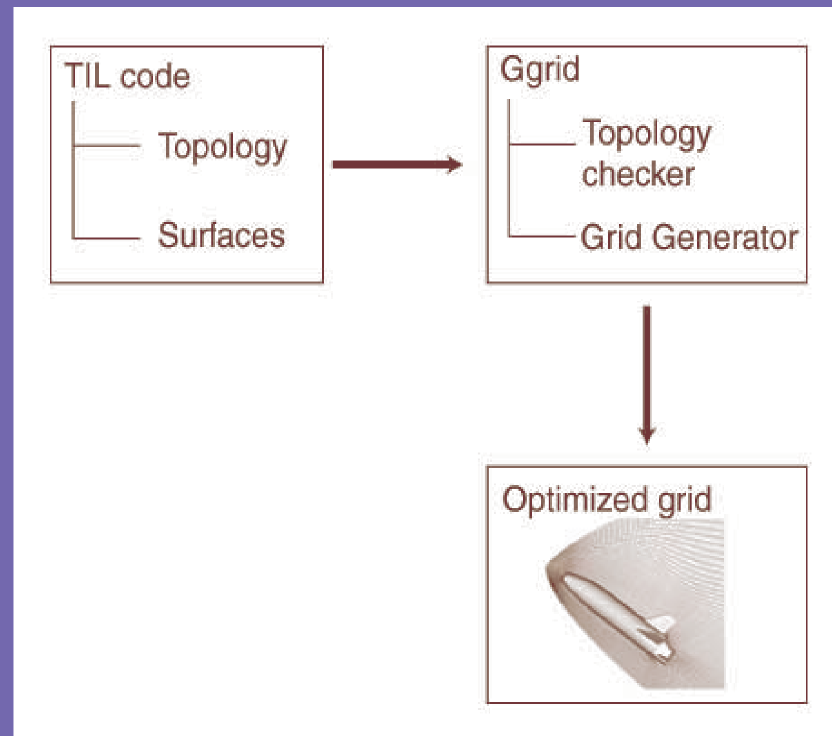
The Property Setter Panel is used for setting boundary conditions for the generated grid. Various flow solvers are already supported, but the list will be further increased, depending on the needs of our users.



Grid generation engine (Ggrid)

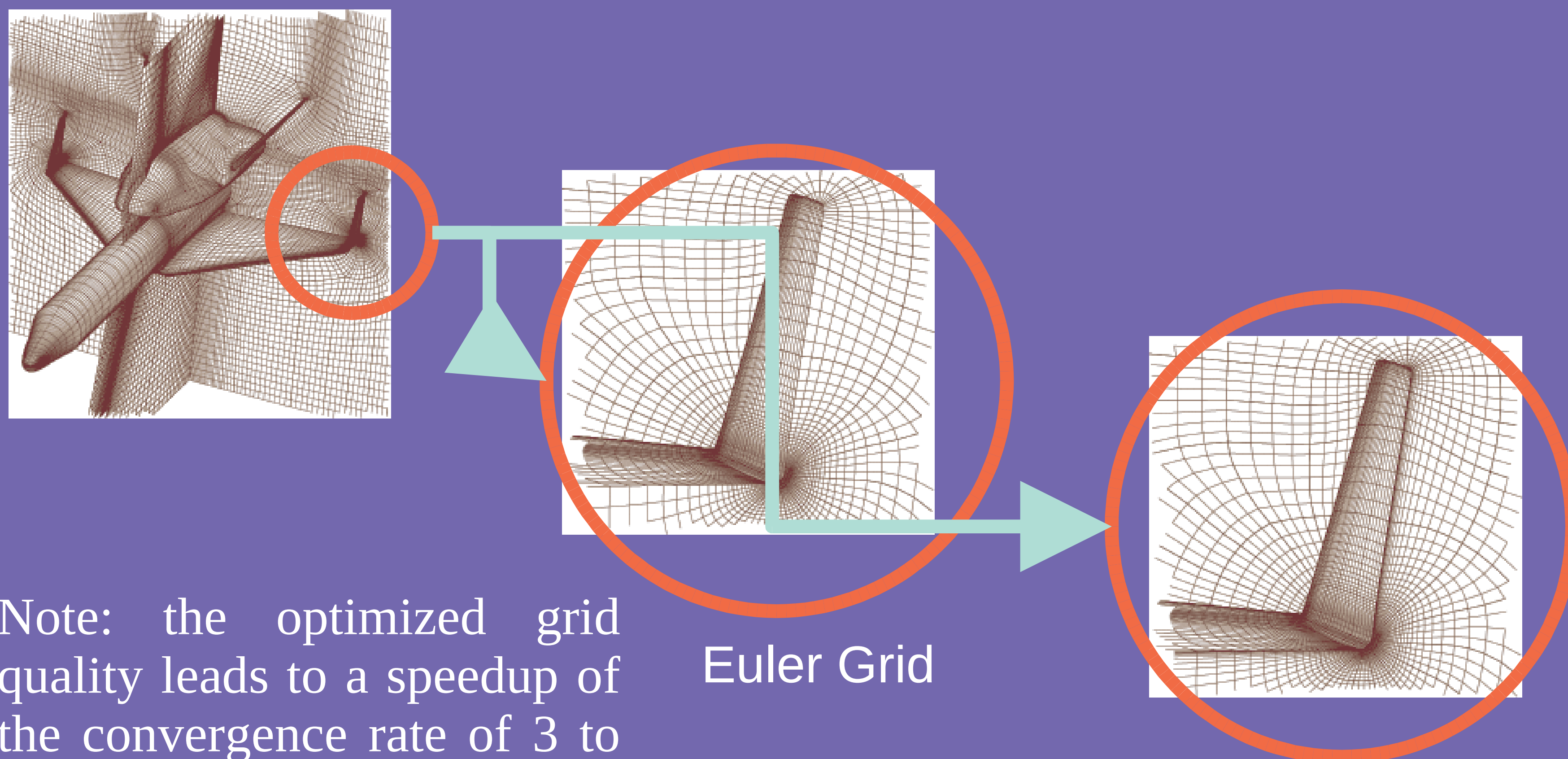
Ggrid is the topology and grid generation engine. It uses the TIL code, i.e. the topology and the surface description, as input. First, the topology is parsed. Then, in the second stage, a multi-block grid is generated. Ggrid provides algorithms to optimize the grid quality with regard to smoothness and orthogonality throughout the entire grid, which translates into greater CFD accuracy and efficiency.

Usually, Ggrid is launched from the AZ-Manager after the topology has been designed and the surfaces have been prepared. However, it can be also launched from a command line, i.e. DOS prompt or UNIX shell. This gives, for instance, users the opportunity to integrate Ggrid into a design loop. After generating the grid, the flow solver is launched to compute the flow field, followed by a post optimizer that modifies the surfaces, which are then used to generate a modified grid. For moderate changes in the surface description the topology does not have to be changed. Thus, human interaction can be entirely be removed from the design loop.



Cluster

Cluster tool converts Euler- into Navier-Stokes grids.



Note: the optimized grid quality leads to a speedup of the convergence rate of 3 to 10, when compared with traditionally generated grids.

Cluster   From Euler to Navier-Stokes

First, generate Euler grid, then apply cluster tool to convert it into a Navier-Stokes grid. Same Euler grid can be used to generate Navier-Stokes grids for different Reynolds numbers.

Variety of different strategies are implemented to steer cell heights, grid line distributions, etc..

Support for multi-grid technique.

Works also for non GridPro structured multi-block grids.

Utilities

chfmt	Convert and translate data formats
cutg	Extract cells from a grid bounded by planes
extconn	Extends connectivity information for periodic and reflected boundary conditions
genconn	Generate connectivity information based on a given multi-block grid
getg	Extract blocks, surfaces, or sheets from given grid data file
grid2til	Convert an elementary multi-block grid into a TIL code
hex2mb	Reverse engineer a multi-block grid from an unstructured hexahedral grid

Utilities

iges2gp	Convert IGES data into GridPro data format
makgman	Convert GridPro grid into WIND input format
mkrib	Generate a ribbon from 3 path lines
mkolp	Post processing (smooth) overlap grid form mrgb -O
mrgg	Merge two grid files into one file and generate connectivity information
mrgn	Merge (equivalent) nodes and cells by tolerance
qchk	Grid quality checker

Utilities

segb	Segment general multi-block data into elementary block data
siz	Check the physical size of data
syncb	Synchronize or re-orient block boundaries or block indices
thin	Remove nodes from triangular mesh while preserving geometry.
trf	Transform grid data (translate, rotate); coarsening grid based on interpolation
xsec	Compute planer cross section of digitized surface

Grid Generation and HPCC

Until recently, there were 3 basic parallelization strategies

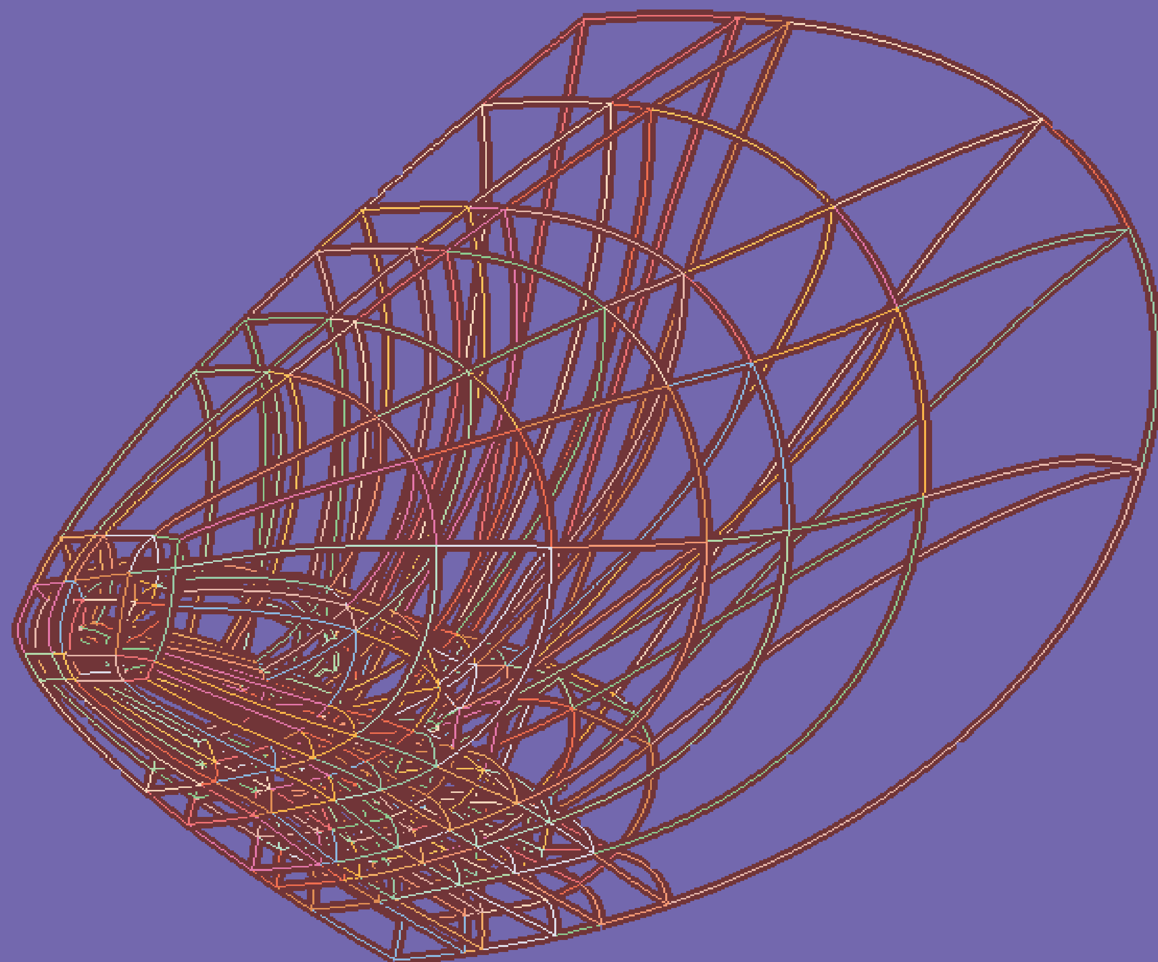
do loop parallelization
parallelization of numerical algorithm
domain decomposition

With the advent of Java a new strategy emerged

thread based parallelization

Parallelization by Domain Decomposition

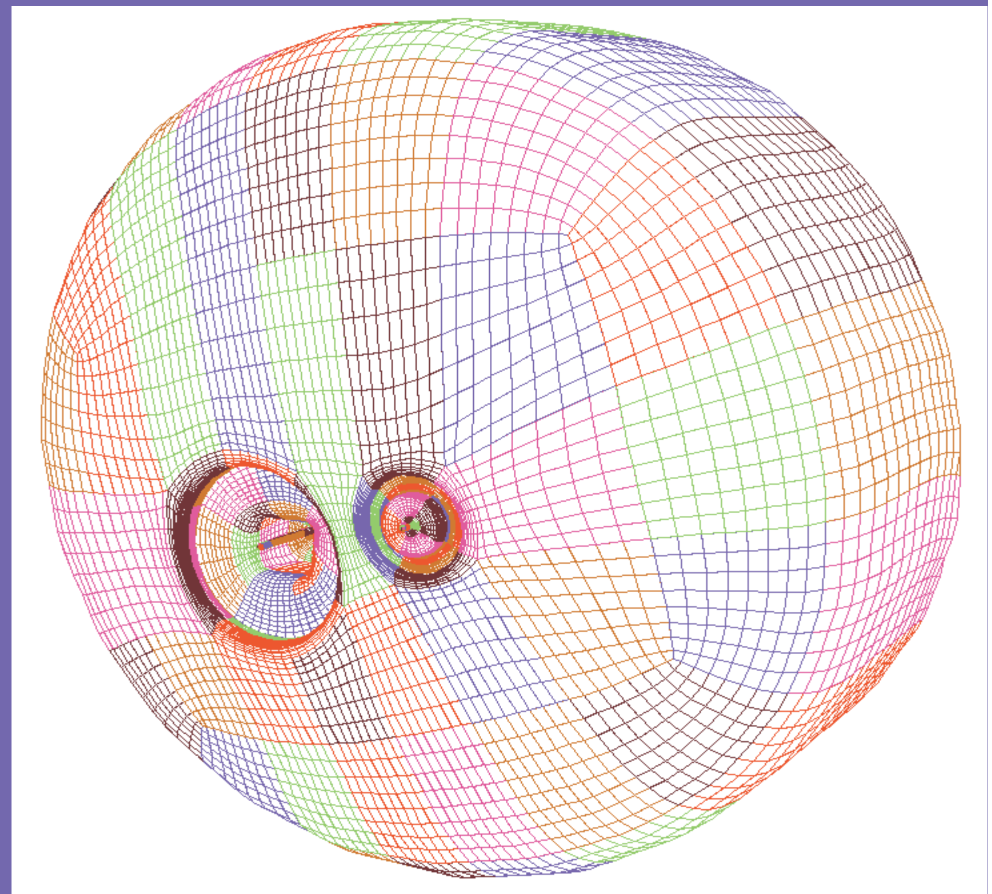
do loop and algorithm parallelization do NOT work for complex geometries, but Domain Decomposition does



Block topology for X-33 configuration

Task mapping for 462 block Huygens space probe

- ***Block to processor***
- ***mapping for 8 processors***
- *(i) Heuristic method*
- *(bin-packing)*
- *(ii) Recursive bisection*



JavaGrid Concept

how to build a computational grid using the internet for large scale computing applications

Java for HPCC

So far, software for computational science and engineering has been written mainly in Fortran, and in recent years the more advanced C programming language has been employed for visualization tasks. Unfortunately, these procedural languages force the programmer to think like a computer, breaking the problem down into a set of basic data types. ***Object-oriented languages***, on the other hand, allow programmers not only to think more efficiently, but also to collaborate more effectively with others.

Ten reasons for using Java as the language for HPCC

1. **Object-Oriented Programming**
2. **Robustness, Understandability**
3. **Computational Efficiency**
4. **Concurrent, distributed, parallel**
5. **Portability**
6. **Leveraging Business Investment**
7. **Multithreading, Dynamic linking**
8. **Database Connectivity**
9. **Remote Method Invocation, Internet**
10. **Security**

C++ is an obscure, hard to debug, unreadable and costly language

Why Threads are good for HPCC

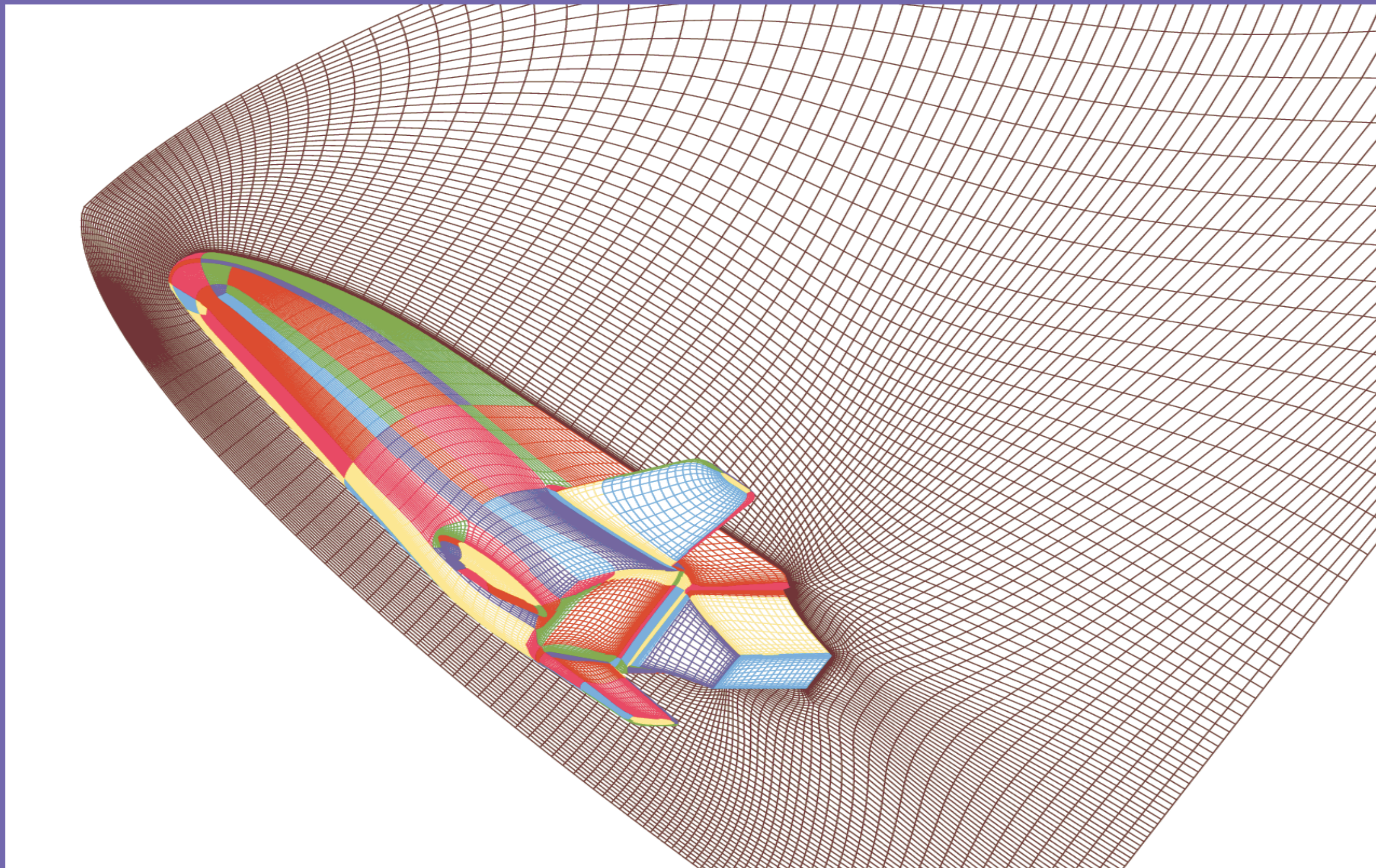
Threads allow straightforward implementation of macro- and microparallelism

Threads provide the general parallelization strategy for HPC codes. Threads are allocated and handled by the OS not by the user. Processor allocation and scheduling is done by the OS.

Advanced numerical schemes, for instance, in CFD, i.e. GMRES, do not require the same computational work for each grid cell, i.e., load changes dynamically.

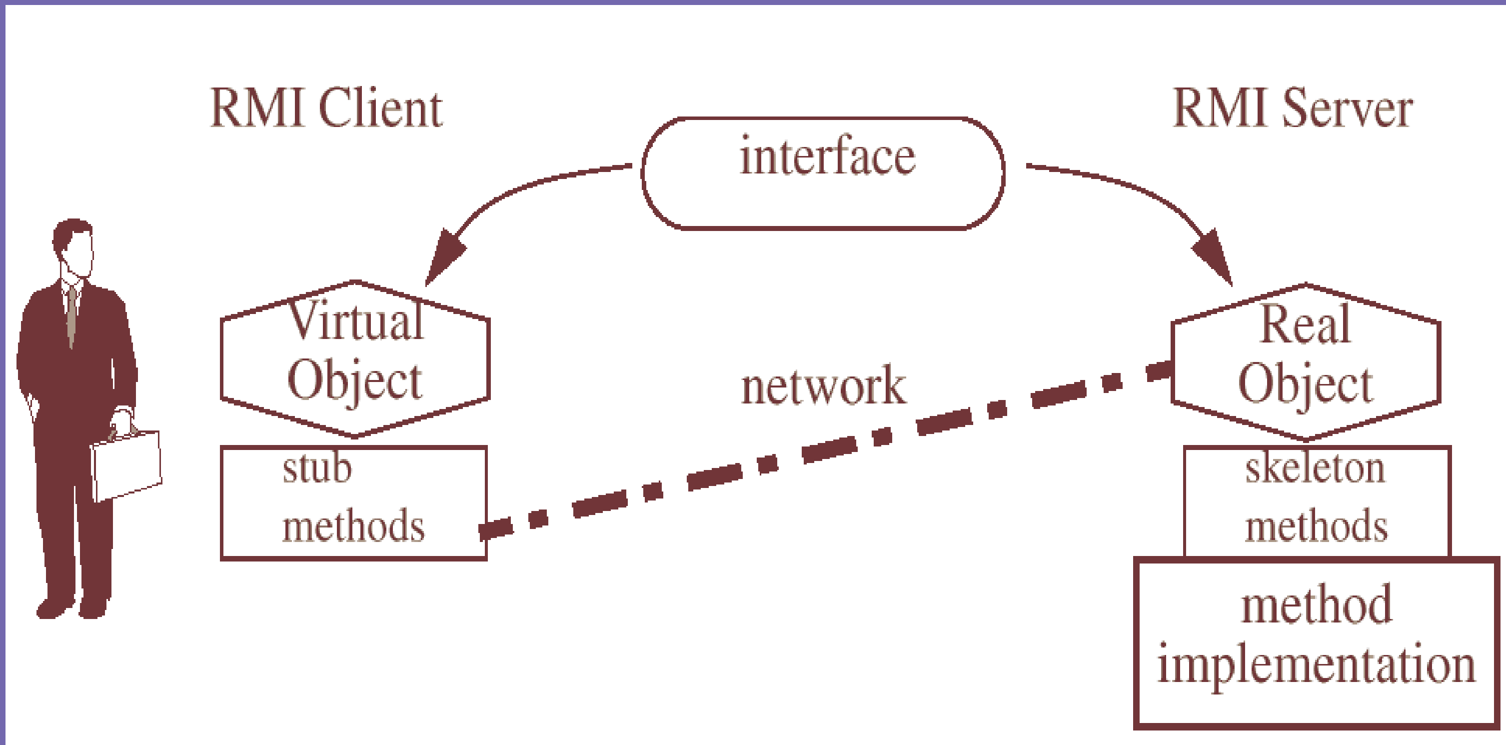
Without threads, highly sophisticated (dynamic) load balancing algorithms are needed for parallel efficiency.

X-33 grid with aerospike engine by GridPro

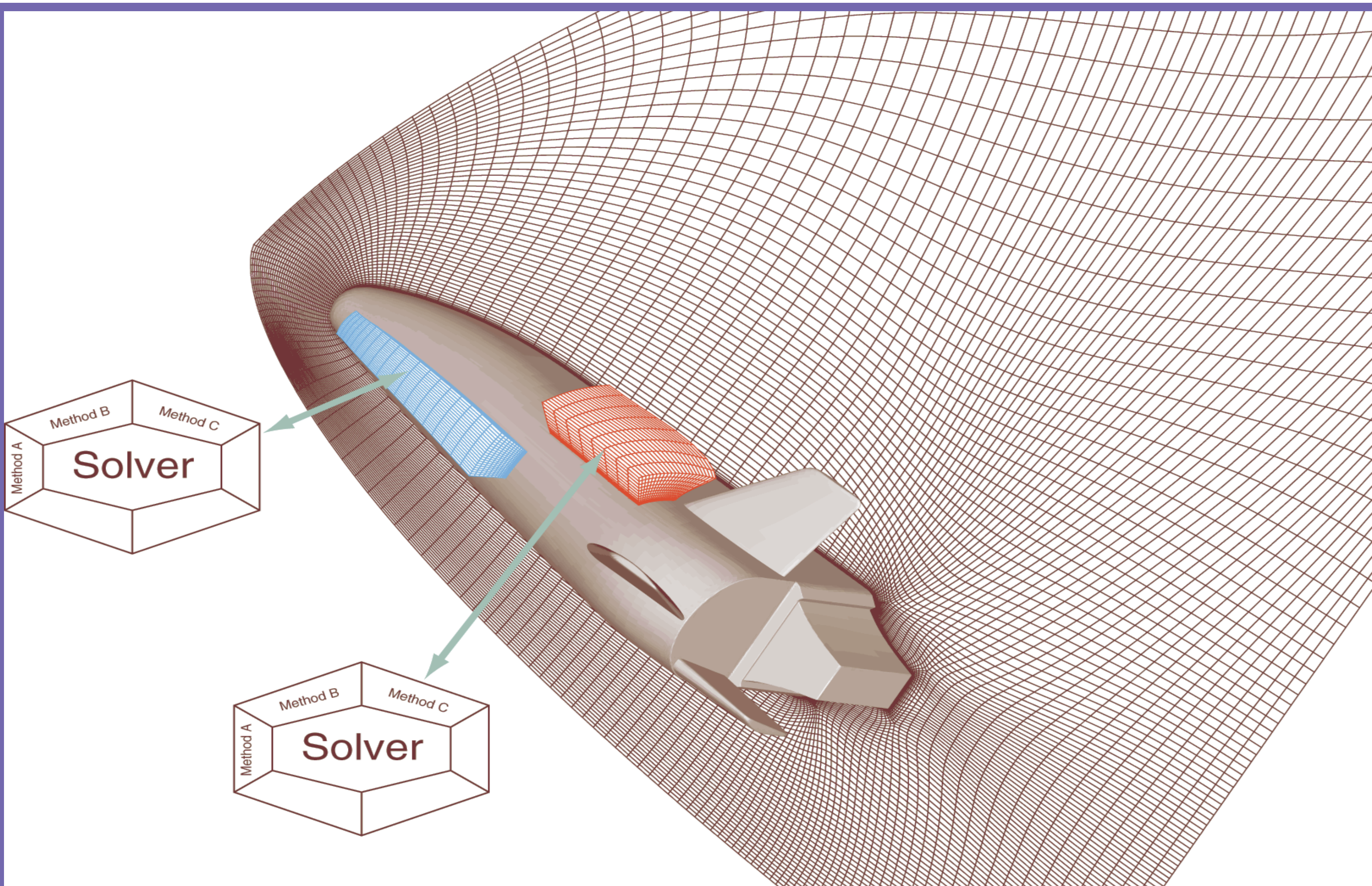


In this figure a multiblock grid for the X-33 vehicle is shown. Each block is run in its own thread. Grids may have thousands of blocks, and thus the OS has to create the corresponding number of threads and is also responsible for starting and stopping all threads.

Remote Objects: Client-Server

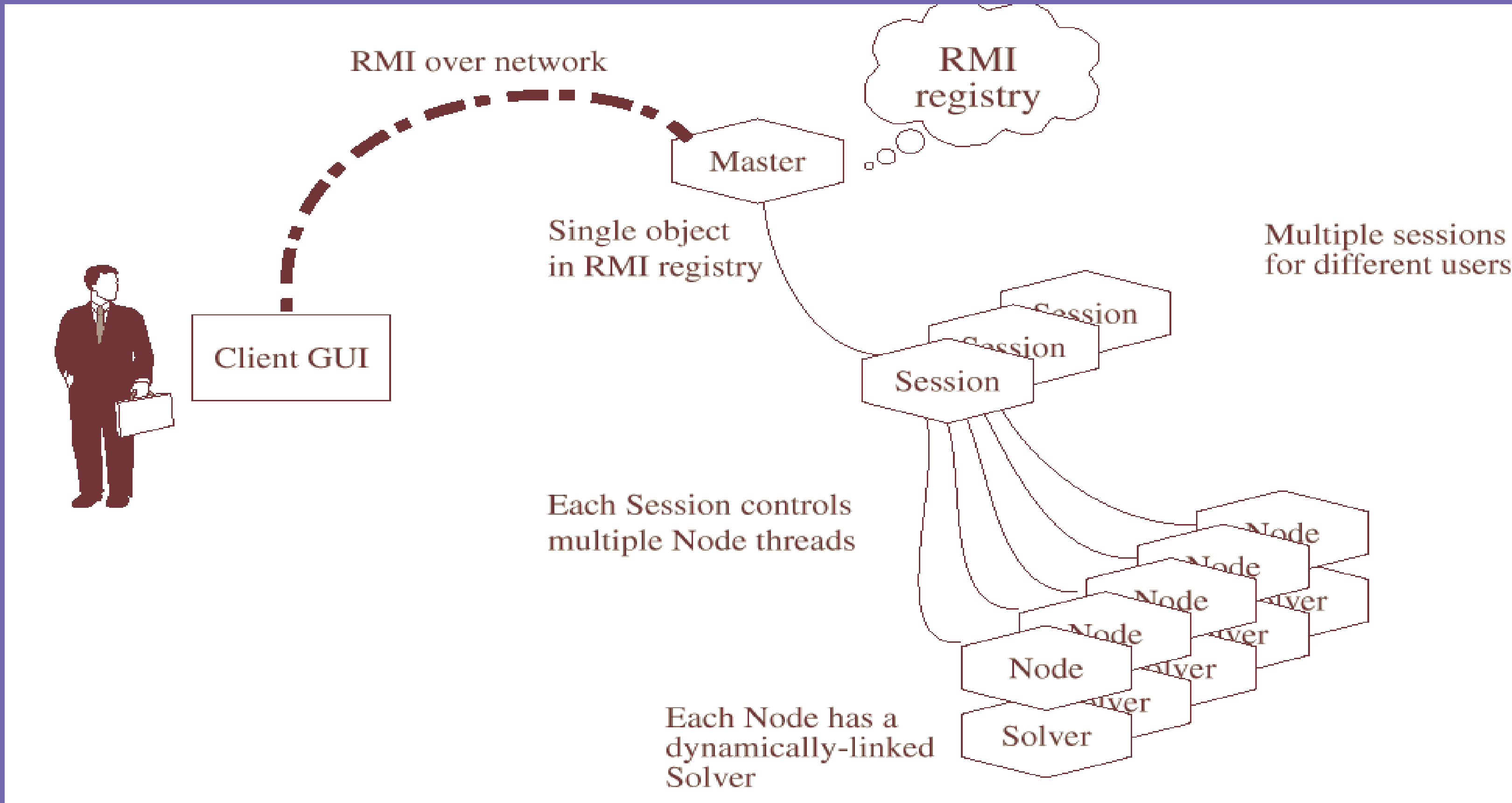


Client-Server communication through Java RMI (Remote Method Invocation). Each shared object has an interface, common to client and server sides, that defines what methods are available. The client can invoke methods on the object, but these are executed on the server where the object actually resides.



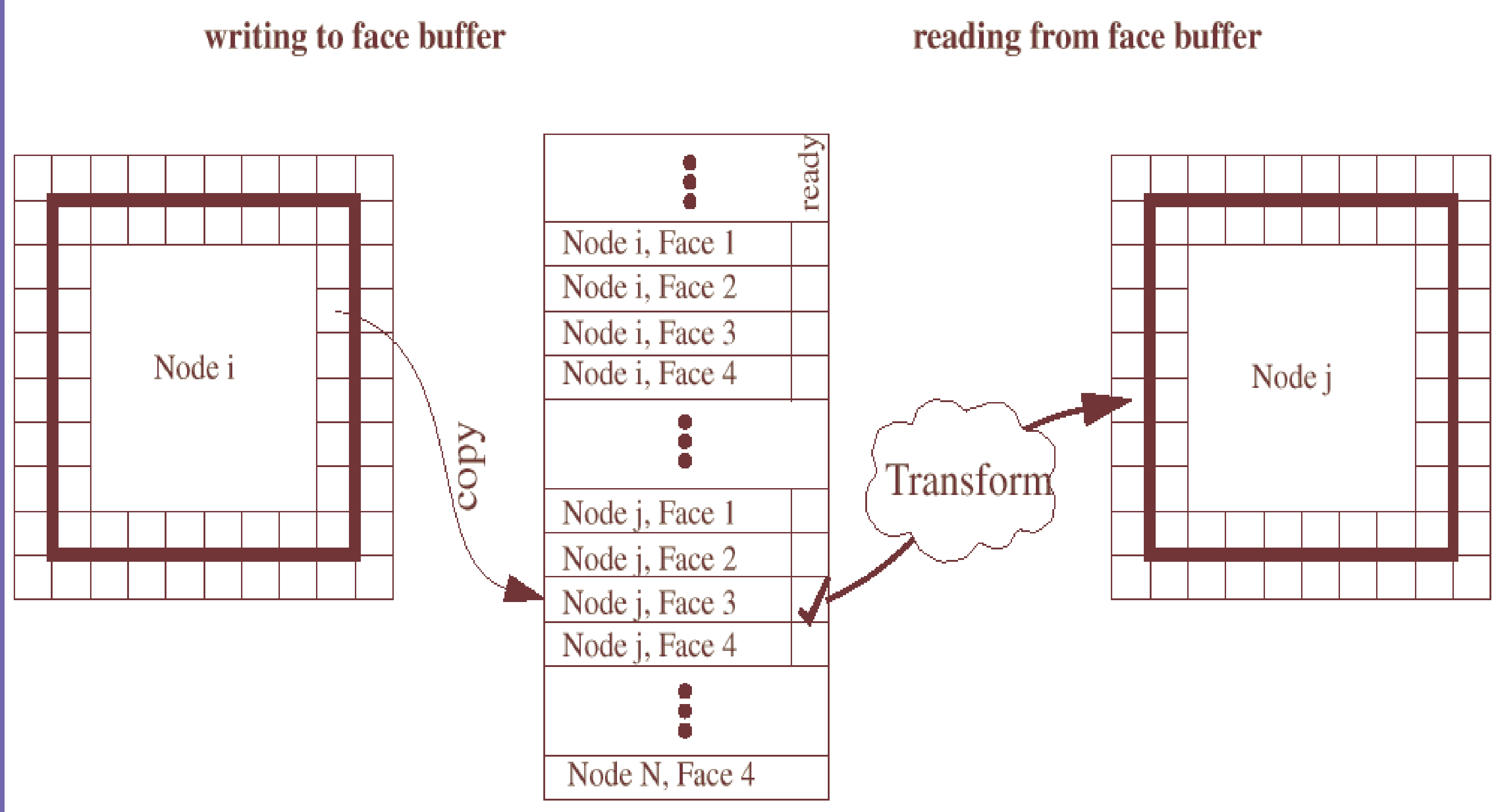
Every solver object contains the data of and the numerics for one block.The solver class is sent from the client to the server that is, different users may use different solvers.

Parallel Structure of the JavaGrid Strategy



The client controls remote objects by invoking methods on them. There is only one registered RMI object, the Master, which can spawn Sessions so that multiple users can work. Each Session can spawn a number of Node threads, each of which can dynamically load a Solver, which is responsible for computation in a single block of the grid.

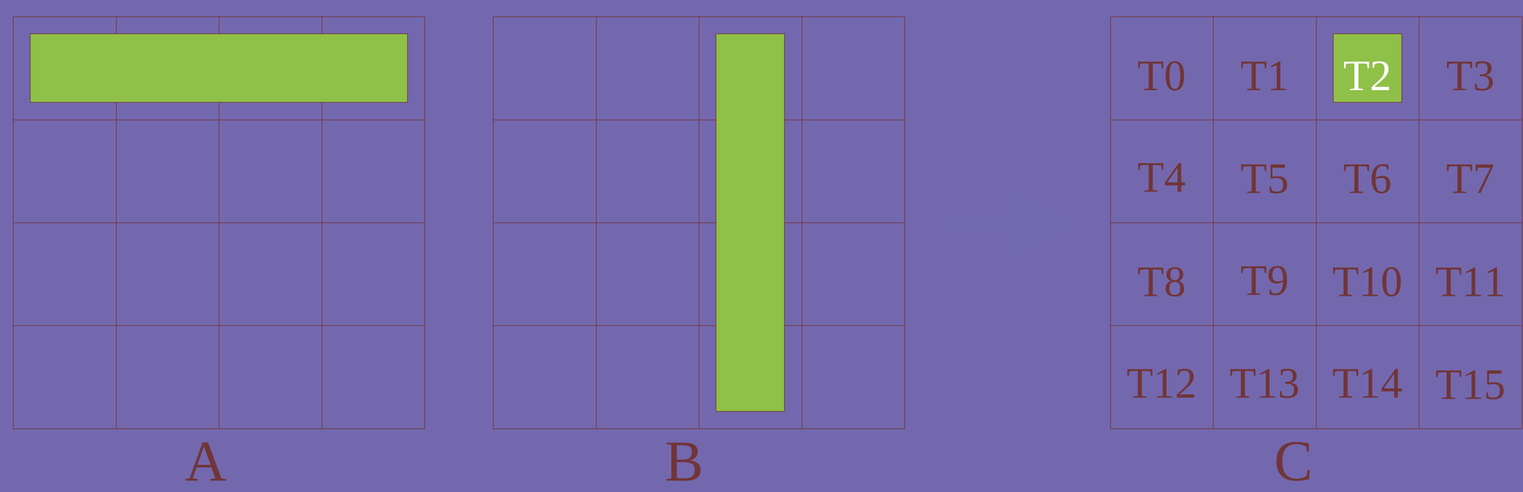
Communication between Blocks (Subdomains)



ommunication between blocks. Each block copies the first layer of internal points nto the face buffer, and the flag is set to “ready” for that face, meaning that the eighbor block can read it. The neighbor reads from the buffer into its halo layer. he word “Transform” refers to the difficult problem (in 3D) of mapping the face rray to one face of the block, or other protocol translation from block to block.

Java Parallel Matrix Multiply

Parallel matrix multiplication is implemented by block matrices, as shown in the Figure Matrices A and B are multiplied to produce C



The multi-threaded matrix multiplication is performed by splitting matrix C into partitions. Each partition is then calculated by one thread, with the thread numbering as shown for matrix C. Concurrent access to the memory containing A and B is necessary: here we see the memory that thread 2 accesses.

Multi-threaded Matrix Multiplication

Megaflop rates for the pure Java multithreaded matrix-multiply benchmark.

number of threads	HP using 16 CPUs		HP using 1 CPU		Sun using 4 CPUs	
	30x30	300x300	30x30	300x300	30x30	300x300
1	7.01	8.64	6.51	8.66	13.40	9.06
4	11.38	33.49	3.86	8.68	19.21	23.56
9	6.33	72.53	2.40	8.73	12.25	22.00
16		118.68		8.69		28.13
25	2.62	112.97	1.04	8.65	5.14	27.84
36	1.83	110.66	0.75	8.64	3.75	30.07
100	0.64	109.53	0.29	8.44	1.57	33.93

On the HP architecture a maximal speedup of 13,74 using 16 processors for the 300x300 matrix example was measured.

Java Parallel Euler Solver

Caltech HP V-Class times for 320 iterations

number of blocks	number of cells	time in seconds		parallel speedup
		single processor	multi processor (16)	
16	121104	3246.73	541.13	6.00
16	200704	6908.88	1077.20	6.41
16	484416	12905.88	2720.48	4.74
48	118803	2980.93	225.76	13.20
48	202800	5190.54	436.09	11.90
48	480000	12663.30	1162.54	10.89

Conclusions

Both structured as well as unstructured hexahedral grids can be interactively constructed for very complex geometries providing extremely high grid quality.

In large scale computations a 3 to 10 fold speed up has been observed in comparison with traditionally generated grids.

Multiblock grids are straightforward to parallelize achieving perfect linear parallel scalability.

The Java thread concept has proved extremely well suited for large scale parallelization and for Grid computing.

Java with its OOP design and unique Internet and security features is the language for HPCC in science and engineering.Numerical performance rivals or exceeds C++.

Future Work

Grid generation

- automatic repair of CAD data
- automatic topology design
- Object-Oriented grid generation
- moving grids with variable topology
- automatic extension to hybrid grids

HPCC and Java

handling of large number of threads (up to ten thousand)
dynamic load balancing of the OS with regard to thread allocation
extension to distributed parallel architectures (Java Spaces)

There is no doubt that Java is the most effective and efficient language for scientific and technical computing.

Java possesses, however, a steep learning curve.

References

Eiseman, Peter R., GridPro v4.1, *The CFD Link to Design, Topology Input Language Manual*, Program Development Company Inc., 300 Hamilton Ave., Suite 409, White Plains, NY 10601, 2000.

Handbook of Grid Generation, Ed. J. F. Thompson et al. , CRC Press, 1999.

Soni, B. et al, Numerical Grid Generation, Proceedings 7th International Grid Generation Conference, Whistler, July 2000, ISGG, MSU Press.

Häuser J. et al., *Parallel Multiblock Grids*, Chap. 12 in Handbook of Grid Generation, Ed. J. F. Thompson et al. , CRC Press, 1999.

Moreira, J.E. et al. *Java Programming for High Performance Numerical Computing*, IBM Systems Journal Vol 39, 1 , 2000.

Häuser, J., et al., *A Pure Java Parallel Flow Solver*, 37th AIAA Aerospace Sciences Meeting and Exhibit, AIAA 99-0549 Reno, NV, USA, 11-14 January 1999.

Winkelmann, R., Häuser J., Williams R.D, *Strategies for Parallel and Numerical Scalability of Large CFD Codes*, Comput. Methods Appl. Mech. Engrg. 174 (1999) 433-456, 1999.

see also **www.cle.de/cfd** for downloading additional information.