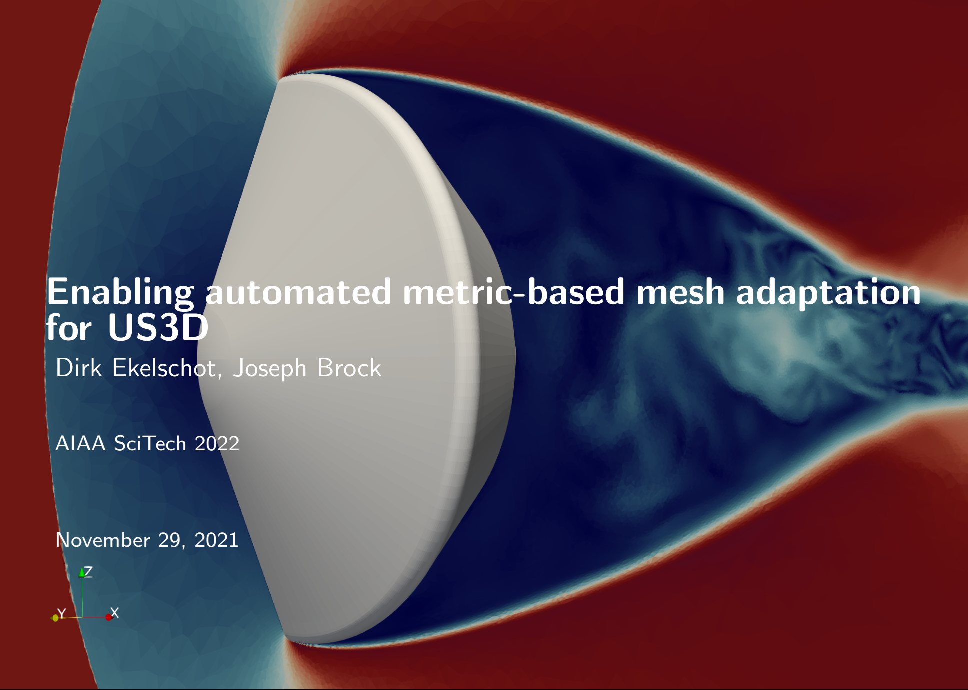


Enabling automated metric-based mesh adaptation for US3D

Dirk Ekelschot, Joseph Brock

AIAA SciTech 2022

November 29, 2021

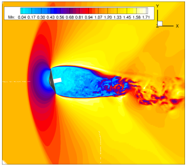
Table of Contents

Table of Contents

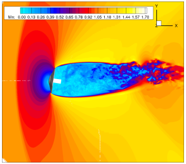
Why anisotropic mesh adaptation?

Goals

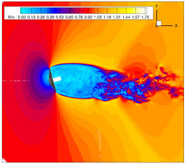
- **Simplify the mesh generation process for unsteady flows past complex geometries.**
- **Improve the resolution of free-flight CFD simulations of atmospheric entry vehicles using US3D.**



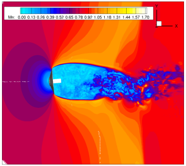
Mach at t≈ 0*s*



Mach at t≈ 2*s*

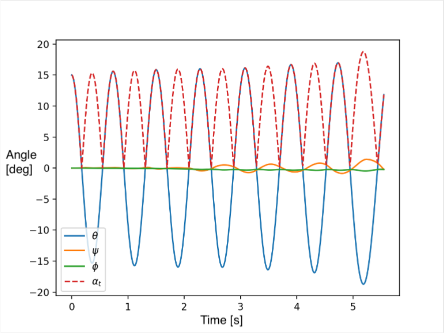


Mach at t≈ 4*s*



Mach at t≈ 5*s*

Mach solution for ADEPT at three different time instances.



Time history of the attitude for the ADEPT geometry.

Table of Contents

Serial implementation

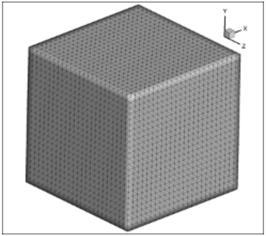
Required developments

- **Calculate metric data based on a US3D flow solution that can drive anisotropic mesh adaptation.**
- Implemented a prismatic boundary layer domain that is kept fixed.
- Linked US3D to MMG and demonstrated iterative adaptation in serial.

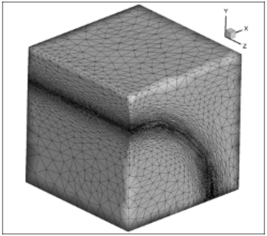
The aim of metric-based mesh adaptation is to control the local error by aligning the local size and directionality of the mesh edges/elements. Based on a given state/derived variable u we can calculate/reconstruct the metric tensor field that perscribes the **preferred direction** and **size** of the local edges/elements[a]:

$$M_{Lp} = D_{Lp} \left(\det \| H_u \| \right)^{\frac{-1}{2p+3}} \mathbf{R}_u^{-1} \| \Lambda \| \mathbf{R}_u \qquad (1)$$

Pass the metric tensor field to a adaptive capability that computes an aligned mesh.



Initial mesh that consists of 82000 tetrahedra.

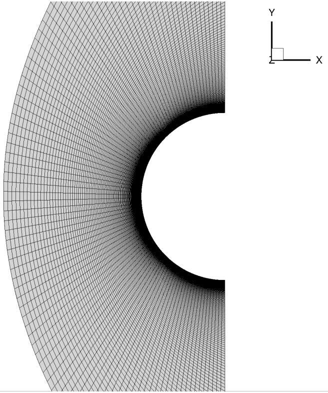


Adapted mesh that consists of 72000 tetrahedra.

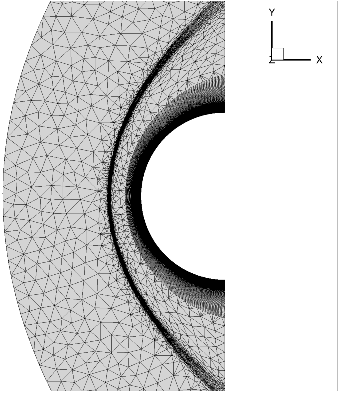
Serial implementation

Required developments

- Calculate metric data based on a US3D flow solution that can drive anisotropic mesh adaptation.
- **Implemented a prismatic boundary layer domain that is kept fixed.**
- Linked US3D to MMG and demonstrated iterative adaptation in serial.



Original hexahedral mesh.

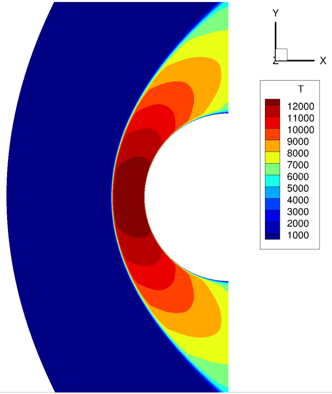


Final hybrid (prisms+tetrahedra) mesh.

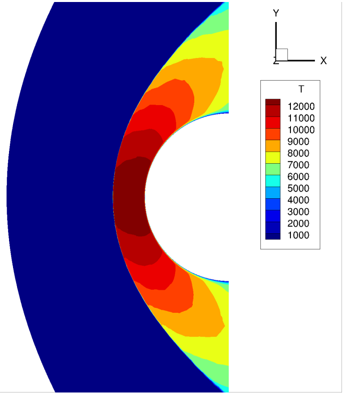
Serial implementation

Required developments

- Calculate metric data based on a US3D flow solution that can drive anisotropic mesh adaptation.
- Implemented a prismatic boundary layer domain that is kept fixed.
- **Linked US3D to MMG and demonstrated iterative adaptation in serial.**



Temperature computed using original hexahedral mesh.

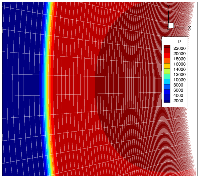


Temperature computed using the final hybrid mesh.

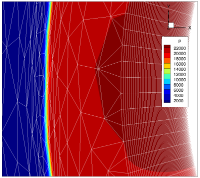
Serial implementation

Required developments

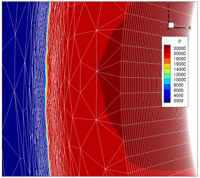
- Calculate metric data based on a US3D flow solution that can drive anisotropic mesh adaptation.
- Implemented a prismatic boundary layer domain that is kept fixed.
- **Linked US3D to MMG and demonstrated iterative adaptation in serial.**



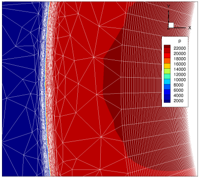
Mesh 1 (hexahedra)



Mesh 2 (prisms+tetrahedra)



Mesh 3 (prisms+tetrahedra)

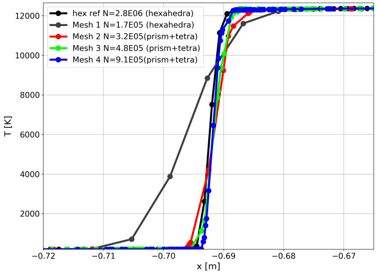


Mesh 4 (prisms+tetrahedra)

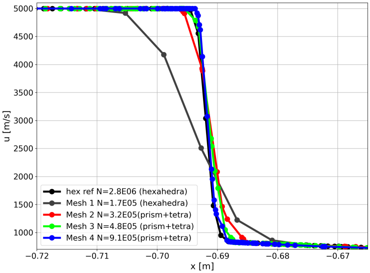
Recap

Required developments

- Calculate metric data based on a US3D flow solution that can drive anisotropic mesh adaptation.
- Implemented a prismatic boundary layer domain that is kept fixed.
- **Linked US3D to MMG and demonstrated iterative adaptation in serial.**



Temperature profile along the stagnation line.

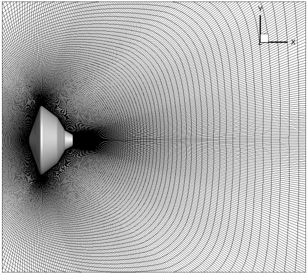


u velocity along the stagnation line.

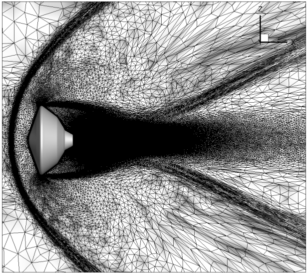
Serial implementation

Required developments

- Calculate metric data based on a US3D flow solution that can drive anisotropic mesh adaptation.
- Implemented a prismatic boundary layer domain that is kept fixed.
- **Linked US3D to MMG and demonstrated iterative adaptation in serial.**



Initial mesh (hexahedra).

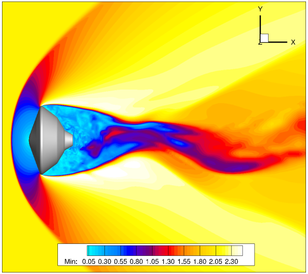


Adapted mesh (prisms+tetrahedra).

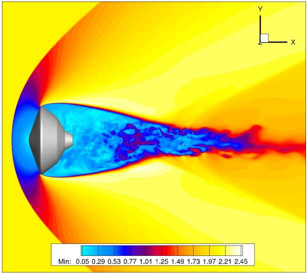
Serial implementation

Required developments

- Calculate metric data based on a US3D flow solution that can drive anisotropic mesh adaptation.
- Implemented a prismatic boundary layer domain that is kept fixed.
- **Linked US3D to MMG and demonstrated iterative adaptation in serial.**



Initial mesh (hexahedra).



Adapted mesh (prisms+tetrahedra).

Limitations

- **Adapting a mesh that consists of approximately** 50×10^6 **elements takes about 1.5 hours in serial using MMG.**
- **Memory usage is limited being to able to just use one node.**

Table of Contents

Parallel implementation

Serial implementation

- Parallel reading from HDF5.
- Parallel reconstruction of the Hessian on a hybrid (prisms+tetrahedra) meshes and hexahedral meshes.
- Prism layer extraction from hexahedral meshes.
- Serial adaptation using MMG.
- Serial writing to HDF5 format compatible with US3D.

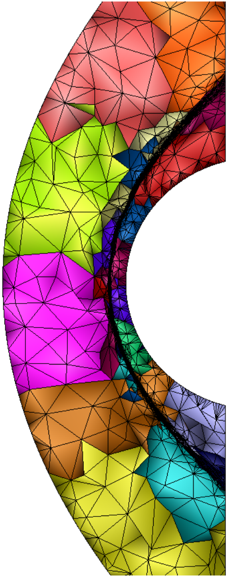
Parallel implementation

Serial implementation

- Parallel reading from HDF5.
- Parallel reconstruction of the Hessian on a hybrid (prisms+tetrahedra) meshes and hexahedral meshes.
- Prism layer extraction from hexahedral meshes.
- Serial adaptation using MMG.
- Serial writing to HDF5 format compatible with US3D.

New features

- **Adapt the tetrahedra in parallel using ParMMG.**
  - **Repartition/redistribute the tetrahedra since ParMMG does not accept prisms.**
  - **Determine the correct shared interfaces between partitions.**
- Patch the adapted tetrahedra topology onto the fixed prismatic boundary layer topology.
- Apply wake refinement by letting the Turbulent Kinetic Energy (TKE) drive isotropic refinement.
- Output the distributed mesh into a single "grid.h5" file that is compatible with US3D.



Parallel implementation

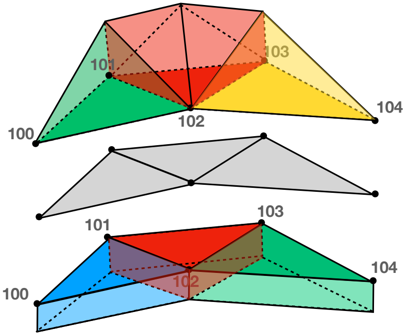
Serial implementation

- Parallel reading from HDF5.
- Parallel reconstruction of the Hessian on a hybrid (prisms+tetrahedra) meshes and hexahedral meshes.
- Prism layer extraction from hexahedral meshes.
- Serial adaptation using MMG.
- Serial writing to HDF5 format compatible with US3D.

New features

- Adapt the tetrahedra in parallel using ParMMG.
  - Repartition/redistribute the tetrahedra since ParMMG does not accept prisms.
  - Determine the correct shared interfaces

    between partitions.
- **Patch the adapted tetrahedra topology onto the fixed prismatic boundary layer topology.**
- Apply wake refinement by letting the Turbulent Kinetic Energy (TKE) drive isotropic refinement.
- Write the distributed mesh into a single "grid.h5" (HDF5) file that is compatible with US3D.



Parallel implementation

Serial implementation

- Parallel reading from HDF5.
- Parallel reconstruction of the Hessian on a hybrid (prisms+tetrahedra) mesh.
- Prism layer extraction from hexahedral meshes.
- Serial adaptation using MMG.
- Serial writing to HDF5 format compatible with US3D.

New features

- Adapt the tetrahedra in parallel using ParMMG.
  - Repartition/redistribute the tetrahedra since ParMMG does not accept prisms.
  - Determine the correct shared interfaces

    between partitions.
- Patch the adapted tetrahedra topology onto the fixed prismatic boundary layer topology.
- **Apply wake refinement by letting the Turbulent Kinetic Energy (TKE) drive isotropic refinement.**
- Write the distributed mesh into a single "grid.h5" (HDF5) file that is compatible with US3D.

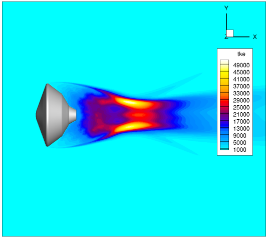
The anisotropic mesh refinement is driven by the metric term that relates the local mesh sizing and directionality to the local spatial linear interpolation error:

$$M_{Lp} = D_{Lp} \left(\det \|H_u\|\right)^{\frac{-1}{2p+3}} \mathbf{R}_u^{-1} \|\Lambda\| \mathbf{R}_u \tag{2}$$

The anisotropic mesh adaptation based on the Mach number is now combined with isotropic mesh adaptation based on the Turbulent Kinetic Energy (TKE).

$$M_t = (\xi - 1) M_{Lp} + \xi M_i \tag{3}$$

where the weights $\xi = f(k_t/k_{t,max})$ are determined linearly based on the TKE.



Parallel implementation

Serial implementation

- Parallel reading from HDF5.
- Parallel reconstruction of the Hessian on a hybrid (prisms+tetrahedra) mesh.
- Prism layer extraction from hexahedral meshes.
- Serial adaptation using MMG.
- Serial writing to HDF5 format compatible with US3D.

New features

- Adapt the tetrahedra in parallel using ParMMG.
  - Repartition/redistribute the tetrahedra since ParMMG does not accept prisms.
  - Determine the correct shared interfaces between partitions.
- Patch the adapted tetrahedra topology onto the fixed prismatic boundary layer topology.
- Apply wake refinement by letting the Turbulent Kinetic Energy (TKE) drive isotropic refinement.
- **Write the distributed mesh into a single "grid.h5" (HDF5) file that is compatible with US3D.**

The anisotropic mesh refinement is driven by the metric term that relates the local mesh sizing and directionality to the local spatial linear interpolation error:

$$M_{Lp} = D_{Lp} \left(\det \|H_u\|\right)^{\frac{-1}{2p+3}} \mathbf{R}_u^{-1} \|\Lambda\| \mathbf{R}_u \qquad (4)$$

The anisotropic mesh adaptation based on the Mach number is now combined with isotropic mesh adaptation based on the Turbulent Kinetic Energy (TKE).

$$M_t = (\xi - 1) M_{Lp} + \xi M_i \qquad (5)$$

where the weights $\xi = f(k_t/k_{t,max})$ are determined linearly based on the TKE.

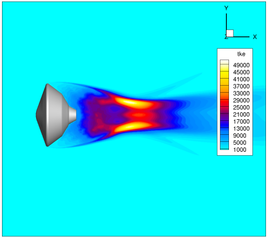
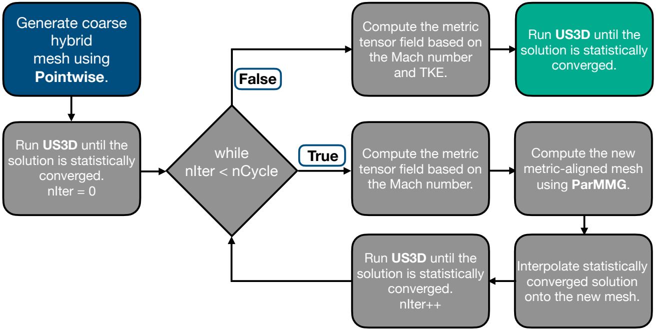
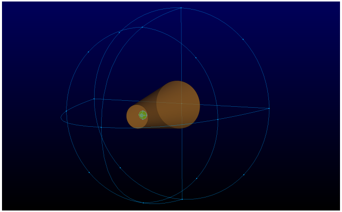


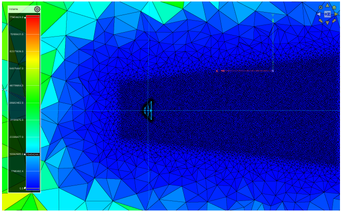
Table of Contents

Numerical examples





Initial mesh definition in Pointwise



A cut of the coarse initial volume mesh.

Numerical examples

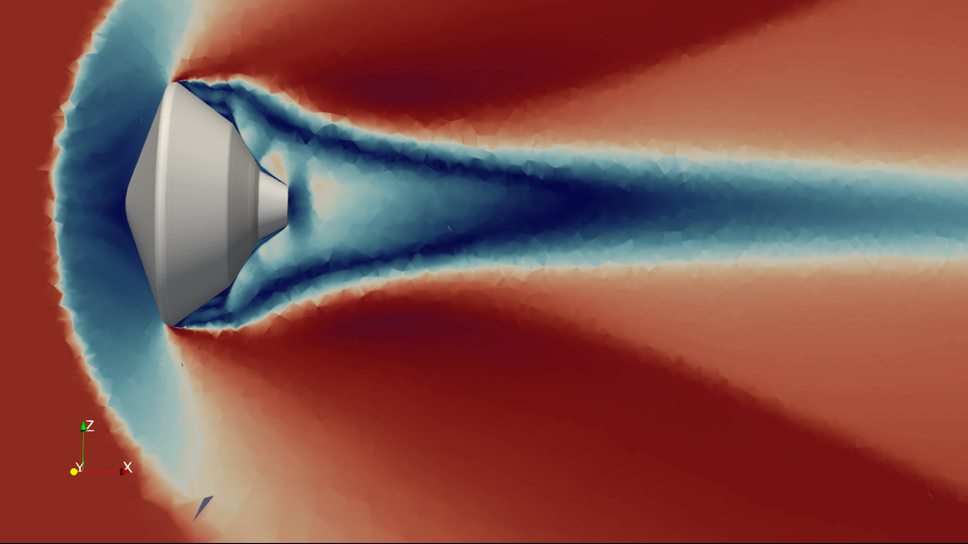
```
#PBS -S /bin/bash
#PBS -N eev_adaptionCycle
#PBS -q devel
#PBS -l select=100:ncpus=40:mpiprocs=40:model=sky_ele
#PBS -l walltime=2:00:00

export FOCUS=/modules/./test15

mkdir adaptCycle
cd adaptCycle
mkdir inputs
cd inputs
ln -s ../../grid.h5 grid.h5
ln -s ../../conn.h5 conn.h5
ln -s ../../data.h5 data.h5
cp -r ../../metric_new.inp metric.inp
cd ..
cp -r ../input.inp .

mpiexec -np 560 $FOCUS > adaption.log

mpiexec -np 40 us3d-prepar --grid=grid.h5 --conn=conn.h5 >
connBuild.log

mpiexec -np 1 us3d-interp --sdir=../ --ddir=../
adaptCycle_r2 --method=1 > interpolation.log

mv data.h5 data_restart.h5

mpiexec -np 4000 us3d --grid=grid.h5 --conn=conn.h5 --
restart=data_restart.h5 --data=data.h5 > wash.log
```

example .pbs file

Supersonic flow ($M = 2.0$, $Re \approx 3.0E05$) past MSL

Ne = 8.1E06 (4.3E06 Prisms and 3.8E06 Tetrahedra)

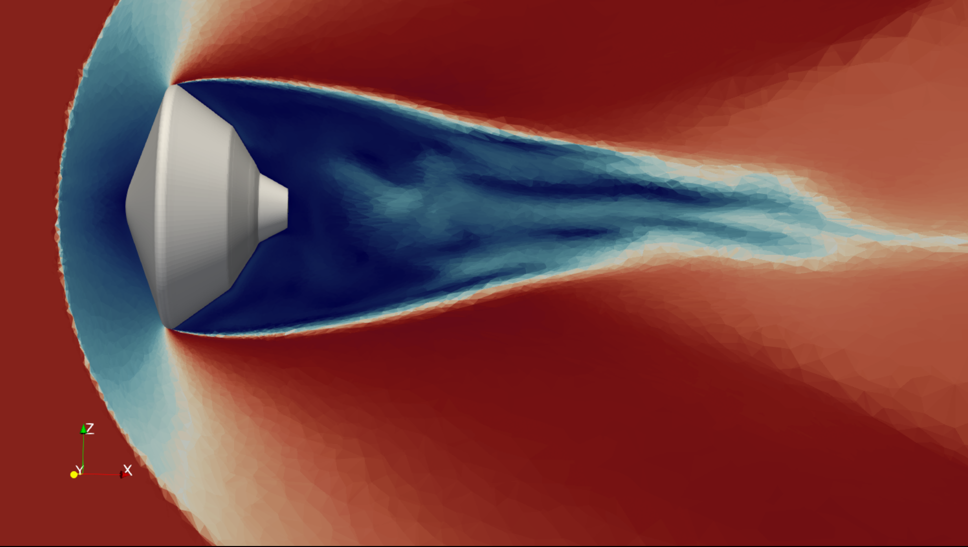
Nn = 2.7E06



Supersonic flow ($M = 2.0$, $Re \approx 3.0E05$) past MSL

Ne = 6.2E06 (4.3E06 Prisms and 1.9E06 Tetrahedra)

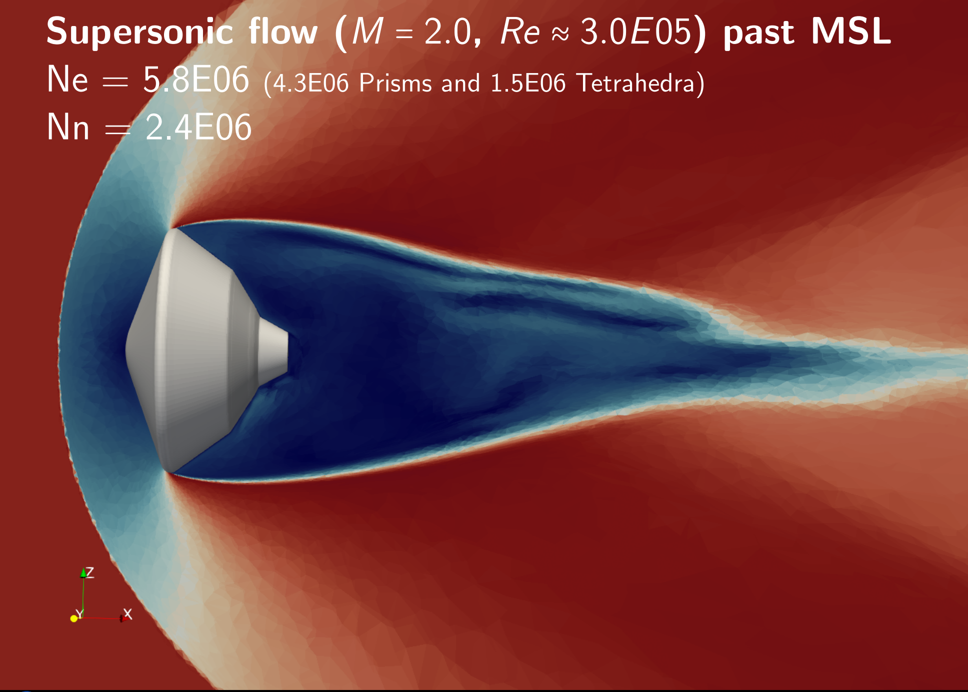
Nn = 2.4E06



Supersonic flow ($M = 2.0$, $Re \approx 3.0E05$) past MSL

Ne = 5.8E06 (4.3E06 Prisms and 1.5E06 Tetrahedra)

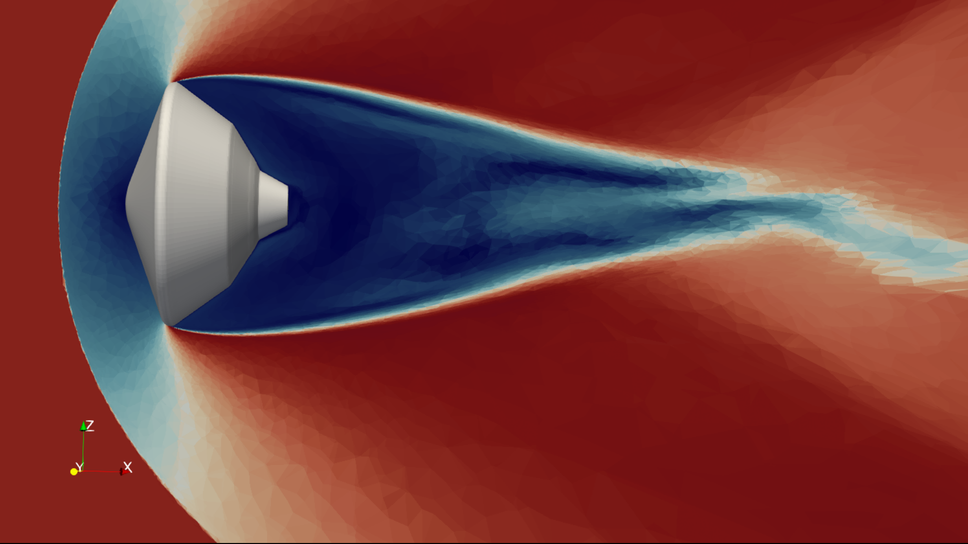
Nn = 2.4E06



Supersonic flow ($M = 2.0$, $Re \approx 3.0E05$) past MSL

Ne = 5.8E06 (4.3E06 Prisms and 1.5E06 Tetrahedra)

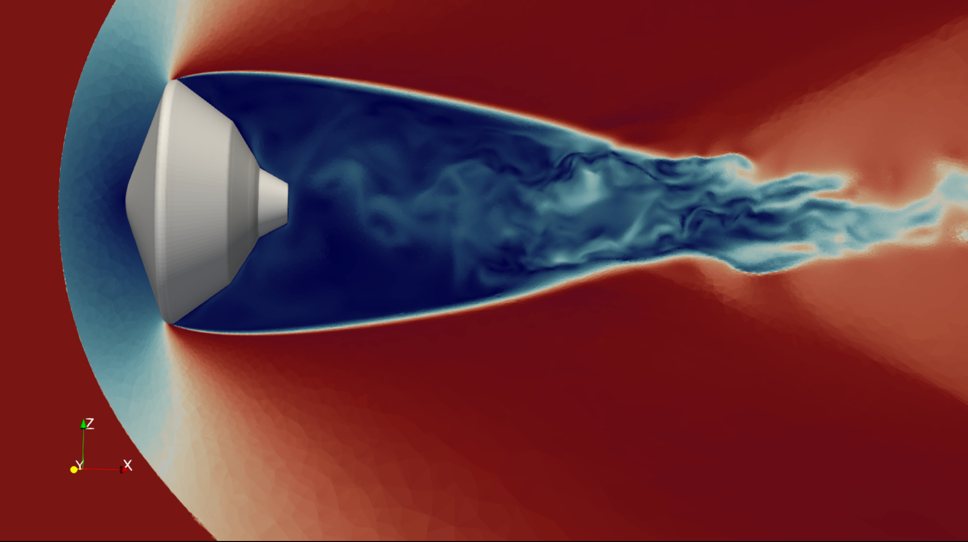
Nn = 2.4E06



Supersonic flow ($M = 2.0$, $Re \approx 3.0E05$) past MSL

Ne = 27.0E06 (4.3E06 Prisms and 22.5E06 Tetrahedra)

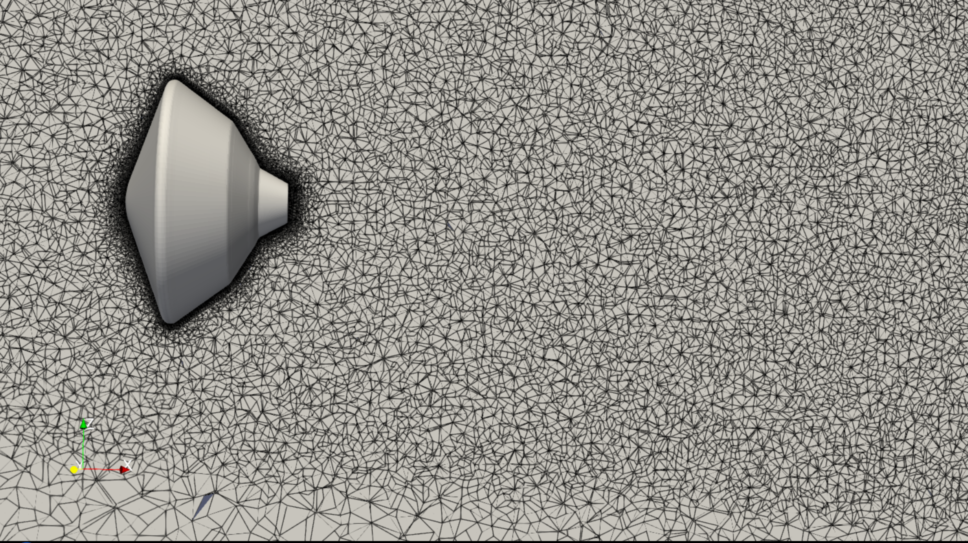
Nn = 5.9E06



Supersonic flow ($M = 2.0$, $Re \approx 3.0E05$) past MSL

$\mathcal{T}_0$ has Ne = 8.1E06 (4.3E06 Prisms and 3.8E06 Tetrahedra)

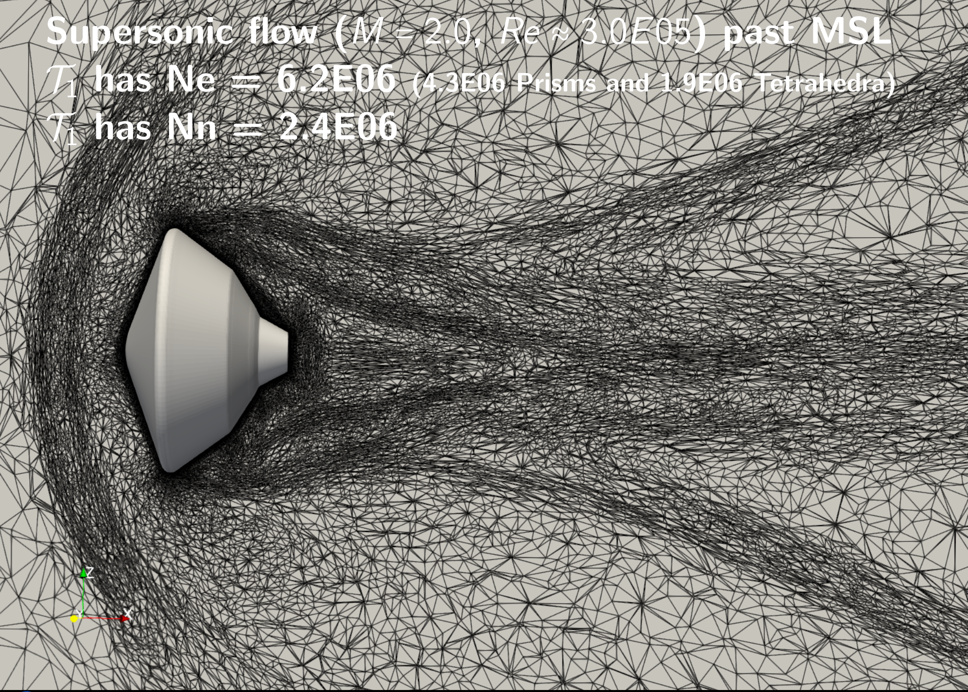
$\mathcal{T}_0$ has Nn = 2.7E06



Supersonic flow ($M = 2.0$, $Re \approx 3.0E05$) past MSL

$\mathcal{T}_1$ **has Ne = 6.2E06** (4.3E06 Prisms and 1.9E06 Tetrahedra)

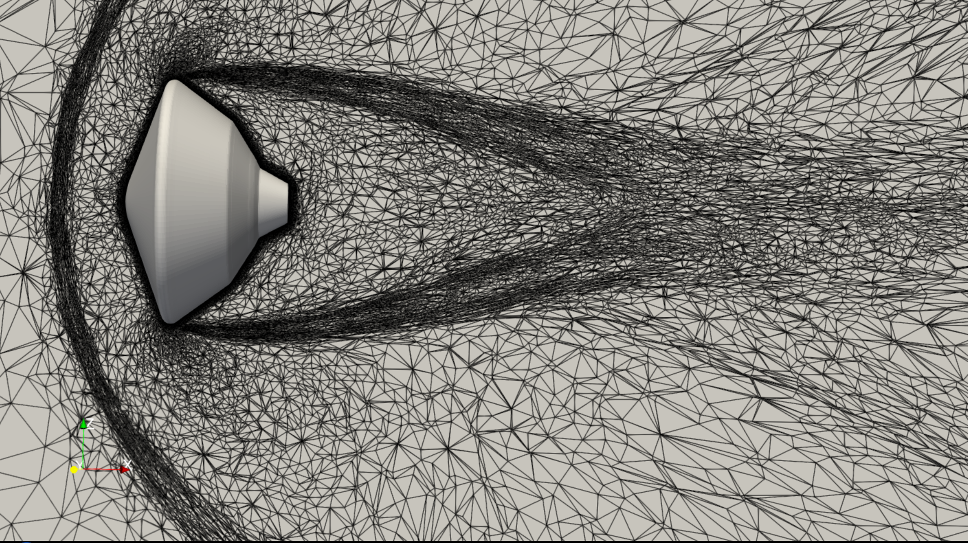
$\mathcal{T}_1$ **has Nn = 2.4E06**



Supersonic flow ($M = 2.0$, $Re \approx 3.0E05$) past MSL

$\mathcal{T}_2$ has Ne = 5.8E06 (4.3E06 Prisms and 1.5E06 Tetrahedra)

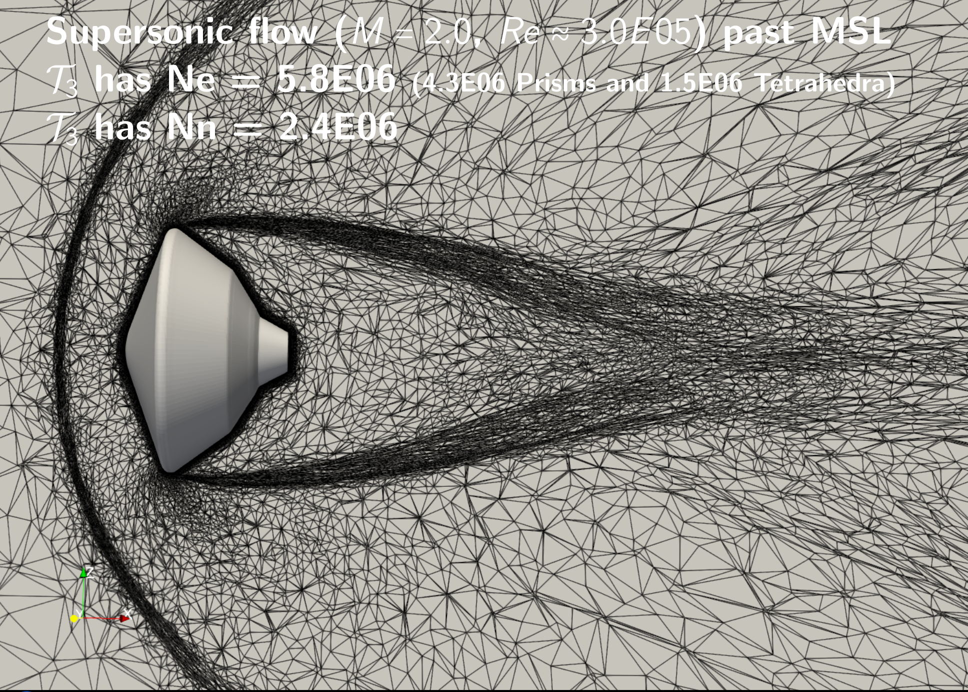
$\mathcal{T}_2$ has Nn = 2.4E06



Supersonic flow ($M = 2.0$, $Re \approx 3.0E05$) past MSL

$\mathcal{T}_3$ has Ne = 5.8E06 (4.3E06 Prisms and 1.5E06 Tetrahedra)

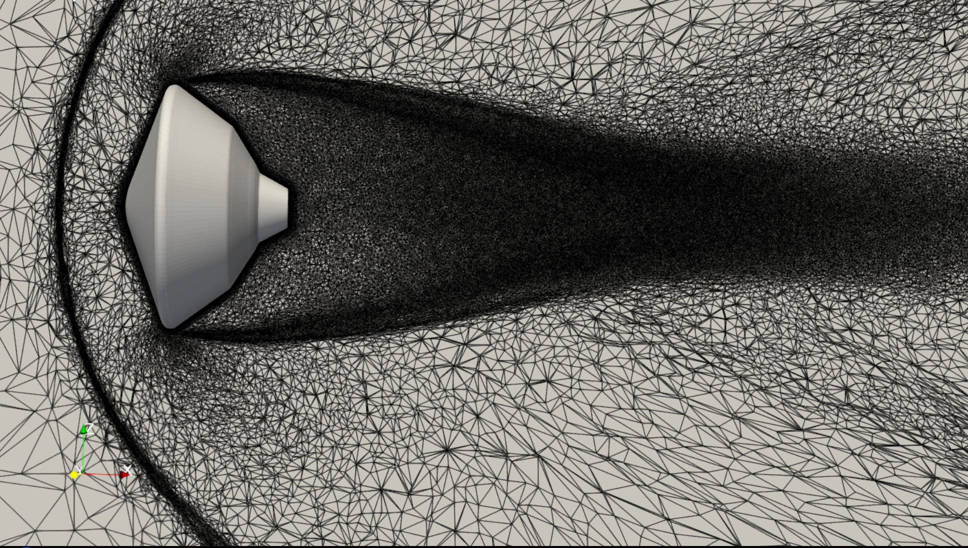
$\mathcal{T}_3$ has Nn = 2.4E06



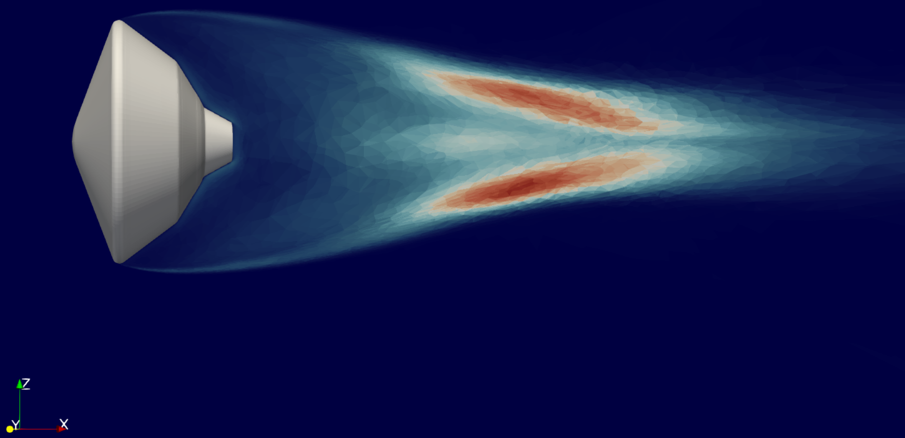
Supersonic flow ($M = 2.0$, $Re \approx 3.0E05$) past MSL

$\mathcal{T}_4$ has Ne = 27.0E06 (4.3E06 Prisms and 22.5E06 Tetrahedra)

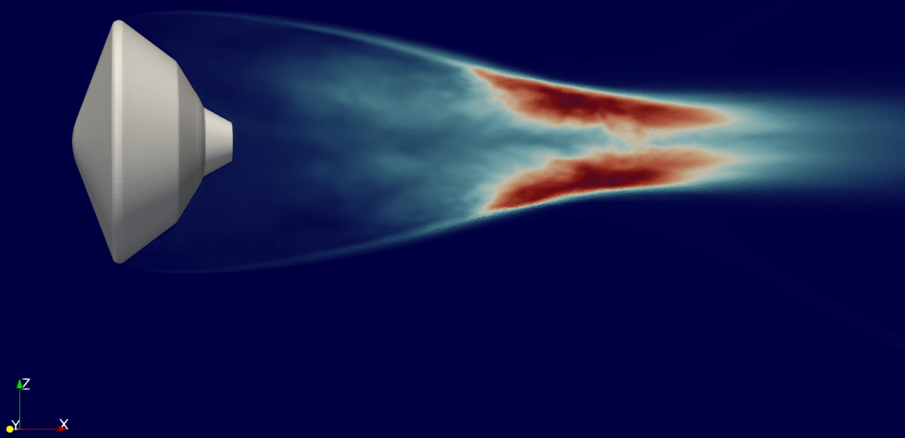
$\mathcal{T}_4$ has Nn = 5.9E06



TKE on $\mathcal{T}_3$



TKE on $\mathcal{T}_4$



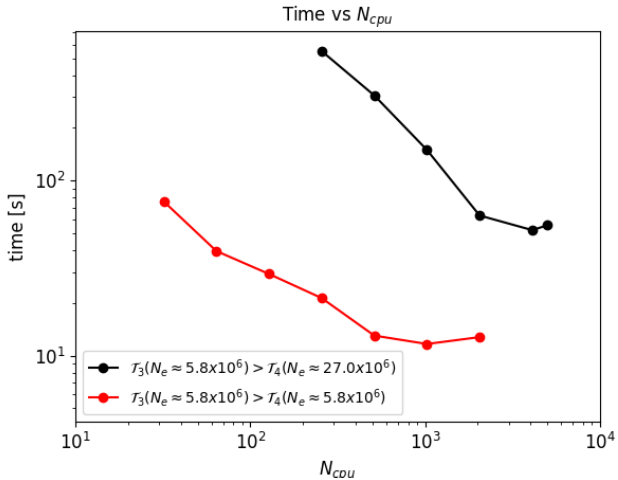
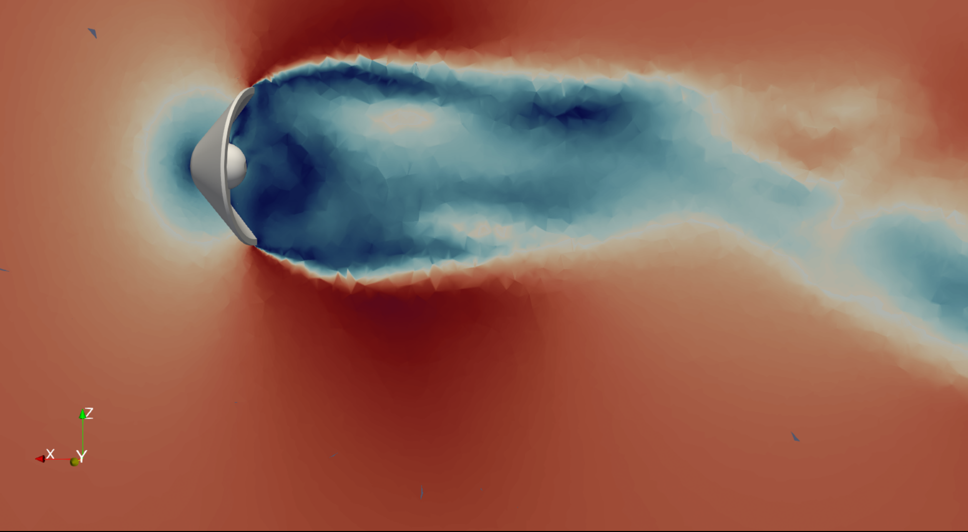


Figure: Time versus number of CPUs used to perform the adaptation for the last two adaptation iterations for supersonic flow past the MSL geometry.

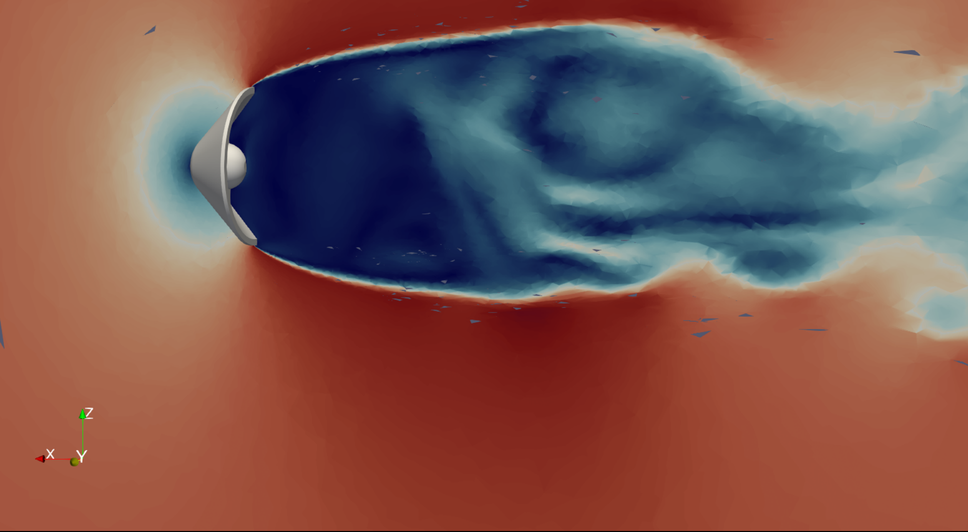
Transonic flow ($M = 0.8$, $Re \approx 4.0E05$) past EEV

Ne = 42.7E06 (37.6E06 Prisms and 5.1E06 Tetrahedra)

Nn = 19.9E06



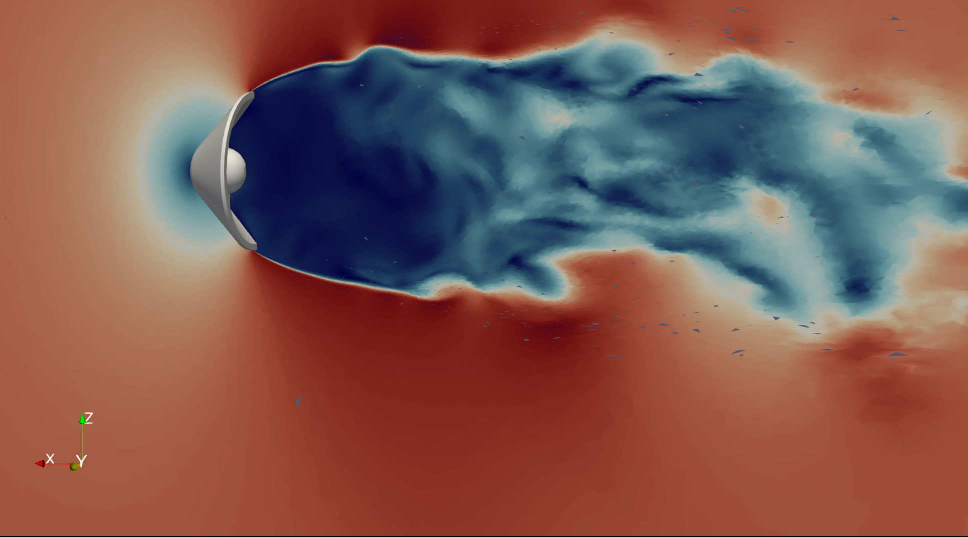
Transonic flow ($M = 0.8$, $Re \approx 4.0E05$) past EEV
Ne = 42.1E06 (37.6E06 Prisms and 4.6E06 Tetrahedra)
Nn = 19.7E06



Transonic flow ($M = 0.8$, $Re \approx 4.0E05$) past EEV

Ne = 48.4E06 (37.6E06 Prisms and 10.8E06 Tetrahedra)

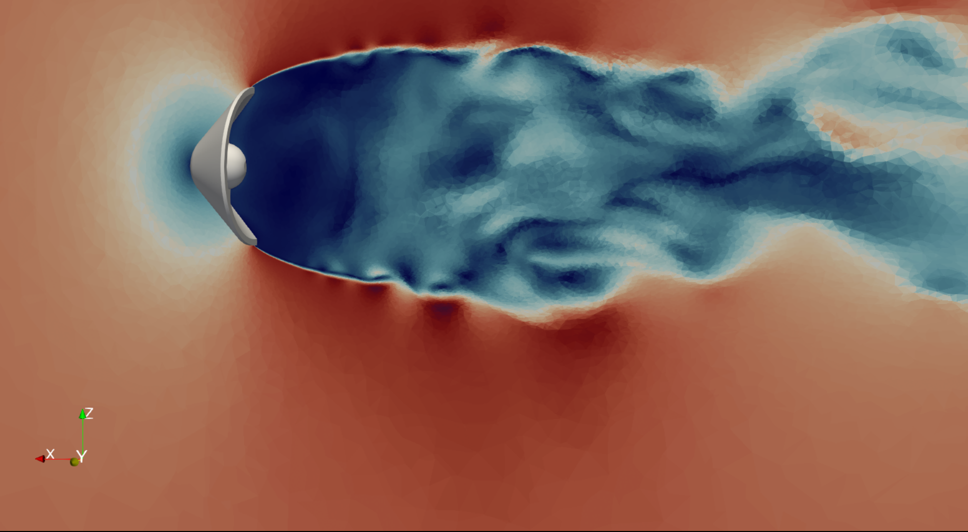
Nn = 20.7E06



Transonic flow ($M = 0.8$, $Re \approx 4.0E05$) past EEV

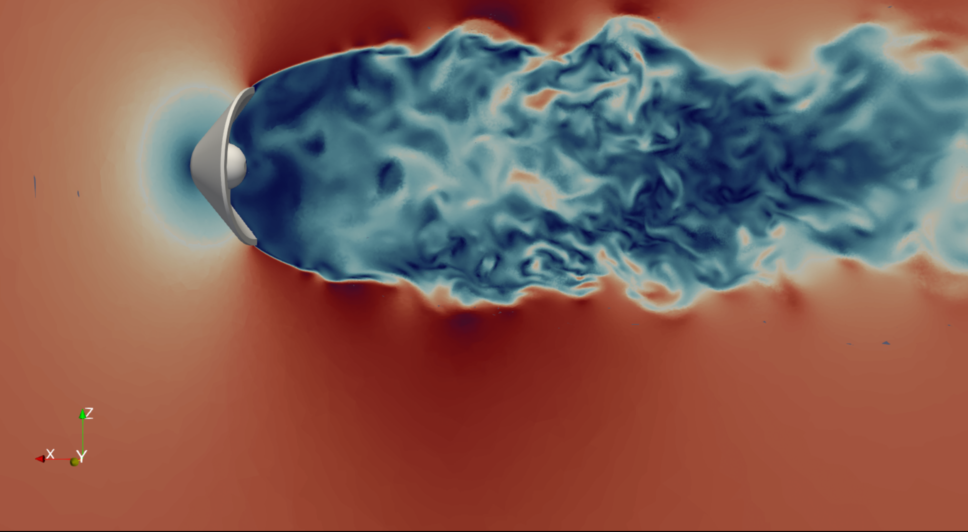
Ne = 47.5E06 (37.6E06 Prisms and 9.9E06 Tetrahedra)

Nn = 20.6E06



Transonic flow ($M = 0.8$, $Re \approx 4.0E05$) past EEV

Ne = 101.2E06 (37.6E06 Prisms and 63.6E06 Tetrahedra)
Nn = 29.6E06

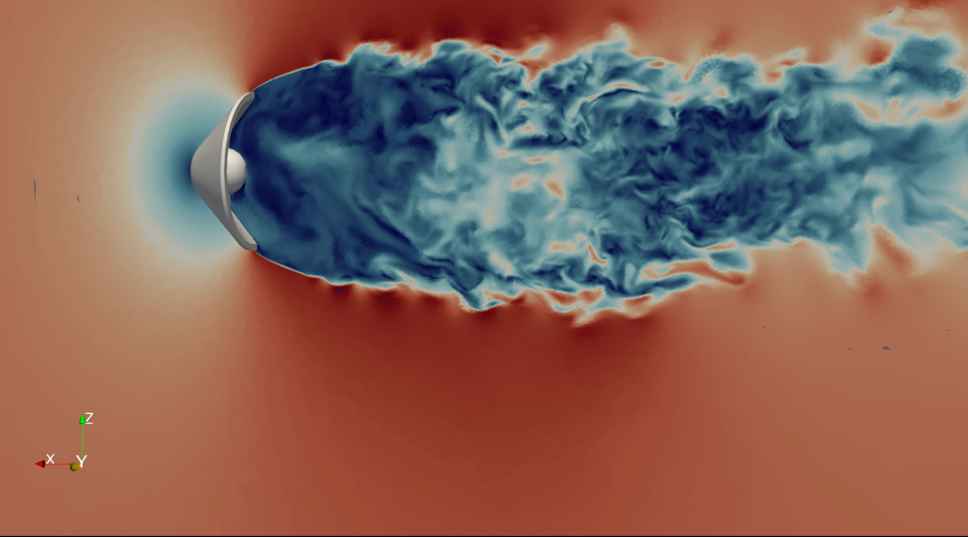


Transonic flow ($M = 0.8$, $Re \approx 4.0E05$) past EEV

Ne = 101.2E06 (37.6E06 Prisms and 63.6E06 Tetrahedra)

Nn = 29.6E06

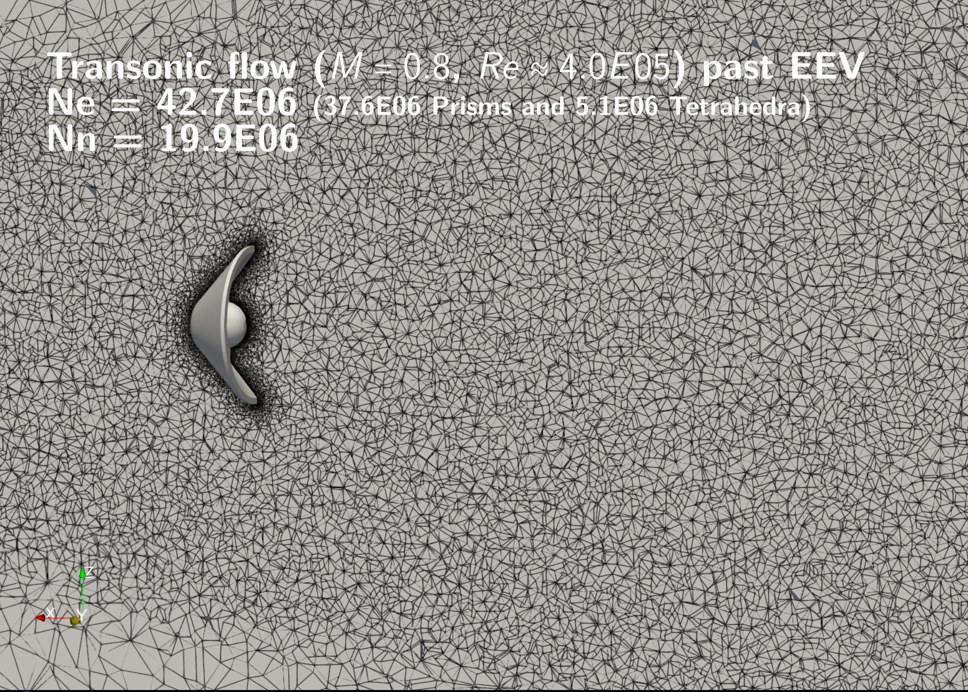
KEC flux



Transonic flow ($M = 0.8$, $Re \approx 4.0E05$) past EEV

Ne = 42.7E06 (37.6E06 Prisms and 5.1E06 Tetrahedra)

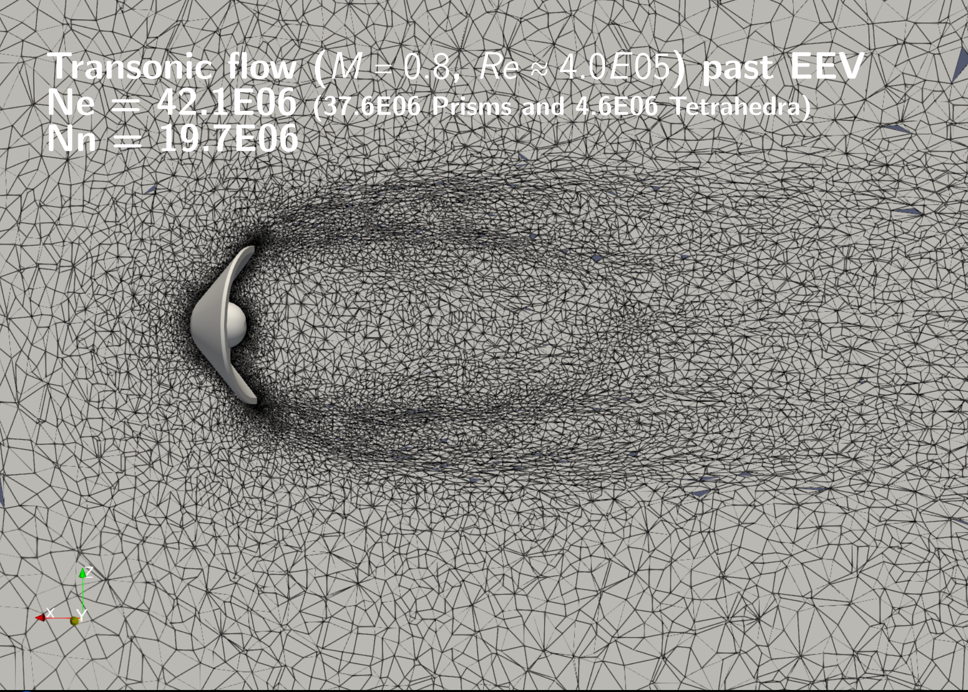
Nn = 19.9E06



Transonic flow ($M = 0.8$, $Re \approx 4.0E05$) past EEV

Ne = 42.1E06 (37.6E06 Prisms and 4.6E06 Tetrahedra)

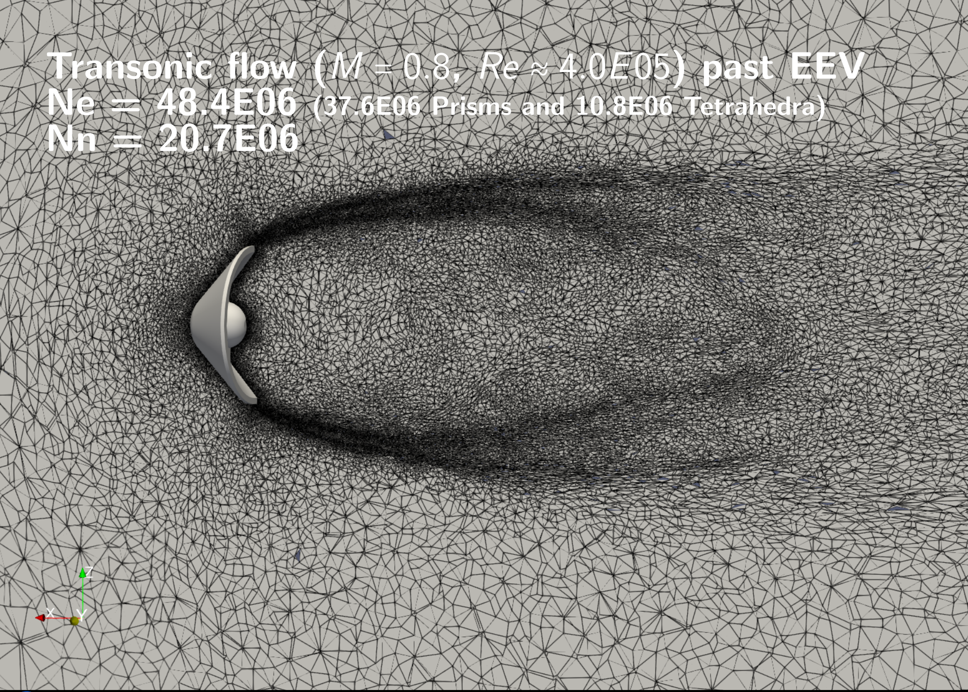
Nn = 19.7E06



Transonic flow ($M = 0.8$, $Re \approx 4.0E05$) past EEV

Ne = 48.4E06 (37.6E06 Prisms and 10.8E06 Tetrahedra)

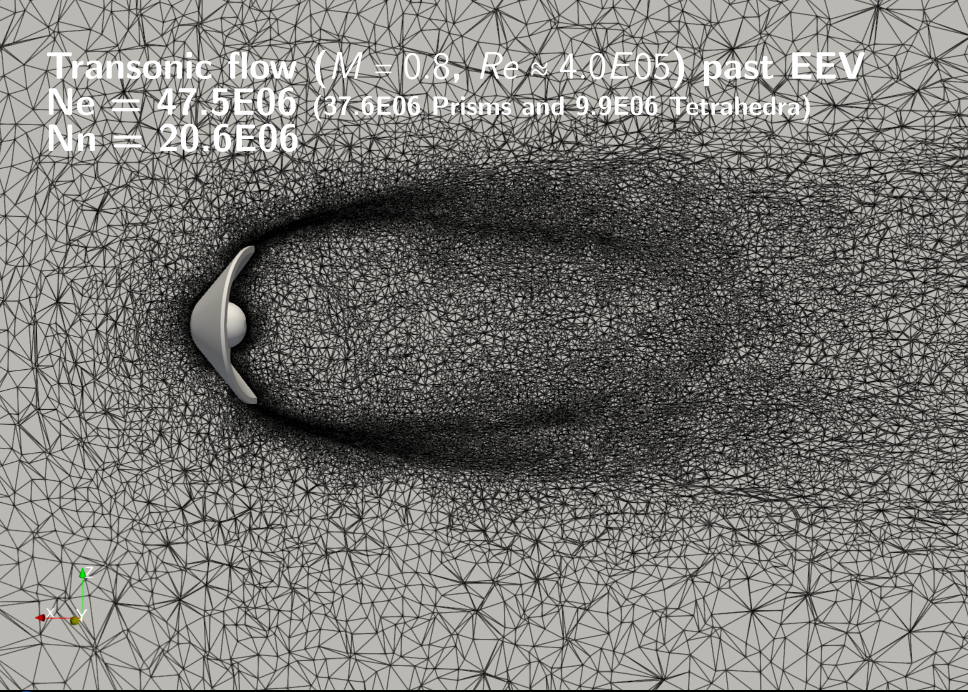
Nn = 20.7E06



Transonic flow ($M = 0.8$, $Re \approx 4.0E05$) past EEV

Ne = 47.5E06 (37.6E06 Prisms and 9.9E06 Tetrahedra)

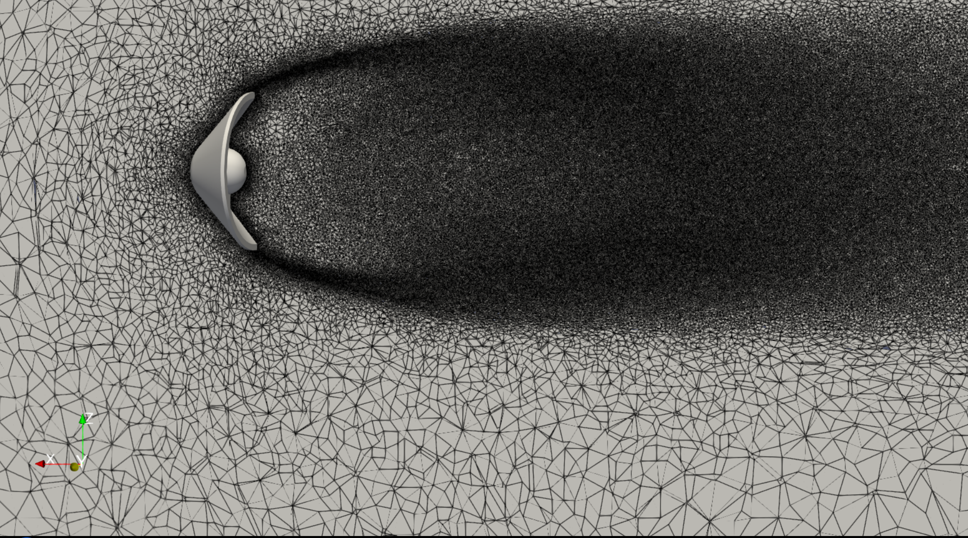
Nn = 20.6E06



Transonic flow ($M = 0.8$, $Re \approx 4.0E05$) past EEV

Ne = 101.2E06 (37.6E06 Prisms and 63.6E06 Tetrahedra)

Nn = 29.6E06



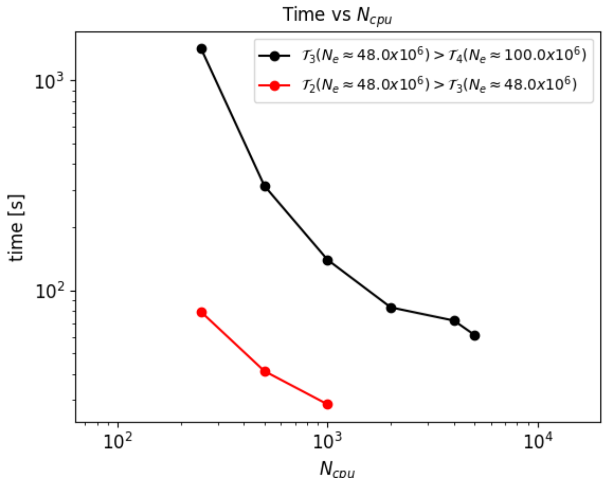


Figure: Time versus number of CPUs used to perform the adaptation for the last two adaptation iterations for supersonic flow past the EEV geometry.

Table of Contents

In summary...

Conclusions

- We currently have an anisotropic mesh adaptation utility that is compatible with US3D and is able to compute a solution-informed adapted mesh for realistic applications that are of interest to the EDL community.
- We have demonstrated a simplified simulation strategy that starts with a small and coarse mesh and applies iterative mesh adaptation to improve the resolution of the simulation while keeping the element count low.
- Currently we are able to generate a anisotropic adapted mesh of approximately 100×10^6 elements in 60 seconds.

Future work

- Streamline the workflow process of running the adaptive capability with US3D to improve usability of the tool.
- Improve the error estimation and include a temporal component to the metric tensor field computation.
- Potentially start employing anisotropic mesh refinement to the dynamic stability calculations for the Earth Entry Vehicle geometry.