Recent examples of compressible aerodynamics simulation with the AMROC framework

Ralf Deiterding

Aerodynamics and Flight Mechanics Research Group
University of Southampton
Boldrewood Campus
Southampton SO17 1BJ, UK
Email: r.deiterding@soton.ac.uk

INPE
3rd October, 2019

Outline

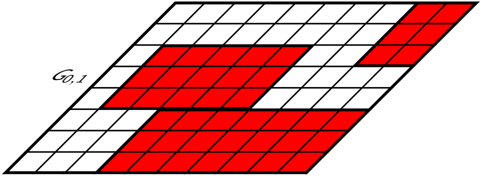
Collaboration with

- Train aerodynamics
  - Jose M. Garro Fernandez (now PhD student)
- High-enthalpy solver for AMROC
  - Chay Atkins (PhD student, Dstl)
- DNS for hypersonic boundary layers
  - Adriano Cerminara (now University of Wolverhampton)
  - Neil Sandham

Block-structured adaptive mesh refinement (SAMR)

For simplicity $\partial_t \mathbf{q}(x,y,t) + \partial_x \mathbf{f}(\mathbf{q}(x,y,t)) + \partial_y \mathbf{g}(\mathbf{q}(x,y,t)) = 0$

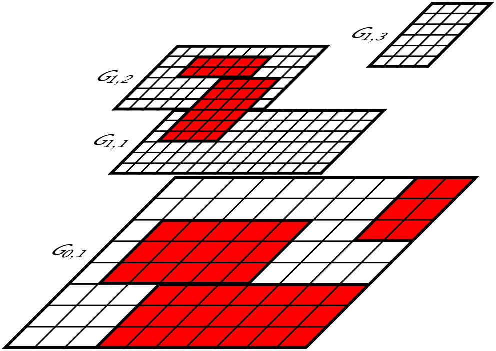
- Refined blocks overlay coarser ones



Block-structured adaptive mesh refinement (SAMR)

For simplicity $\partial_t \mathbf{q}(x,y,t) + \partial_x \mathbf{f}(\mathbf{q}(x,y,t)) + \partial_y \mathbf{g}(\mathbf{q}(x,y,t)) = 0$

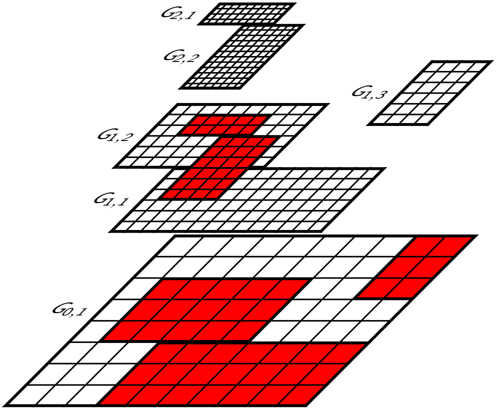
- Refined blocks overlay coarser ones



Block-structured adaptive mesh refinement (SAMR)

For simplicity $\partial_t \mathbf{q}(x,y,t) + \partial_x \mathbf{f}(\mathbf{q}(x,y,t)) + \partial_y \mathbf{g}(\mathbf{q}(x,y,t)) = 0$

- Refined blocks overlay coarser ones



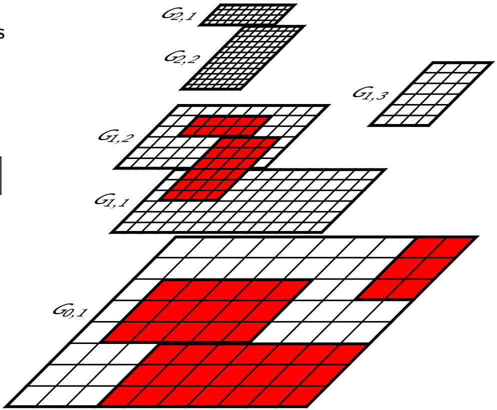
Block-structured adaptive mesh refinement (SAMR)

For simplicity $\partial_t \mathbf{q}(x,y,t) + \partial_x \mathbf{f}(\mathbf{q}(x,y,t)) + \partial_y \mathbf{g}(\mathbf{q}(x,y,t)) = 0$

- Refined blocks overlay coarser ones
- Refinement in space *and time* by factor r_l [Berger and Colella, 1988]
- Block (aka patch) based data structures
- Numerical scheme

$$\mathbf{Q}_{jk}^{n+1} = \mathbf{Q}_{jk}^{n} - \frac{\Delta t}{\Delta x}\left[\mathbf{F}_{j+\frac{1}{2},k} - \mathbf{F}_{j-\frac{1}{2},k}\right] - \frac{\Delta t}{\Delta y}\left[\mathbf{G}_{j,k+\frac{1}{2}} - \mathbf{G}_{j,k-\frac{1}{2}}\right]$$

only for single patch necessary



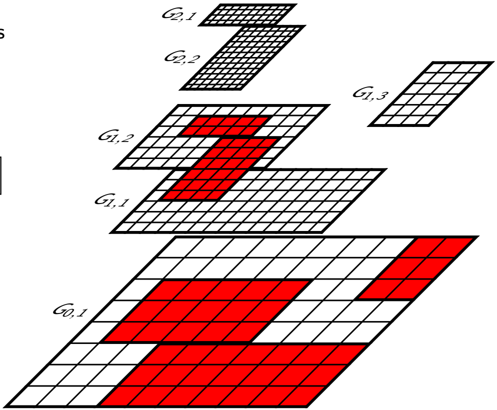
Block-structured adaptive mesh refinement (SAMR)

For simplicity $\partial_t \mathbf{q}(x,y,t) + \partial_x \mathbf{f}(\mathbf{q}(x,y,t)) + \partial_y \mathbf{g}(\mathbf{q}(x,y,t)) = 0$

- Refined blocks overlay coarser ones
- Refinement in space *and time* by factor r_l [Berger and Colella, 1988]
- Block (aka patch) based data structures
- \+ Numerical scheme

$$\mathbf{Q}^{n+1}_{jk} = \mathbf{Q}^n_{jk} - \frac{\Delta t}{\Delta x}\left[\mathbf{F}_{j+\frac{1}{2},k} - \mathbf{F}_{j-\frac{1}{2},k}\right] - \frac{\Delta t}{\Delta y}\left[\mathbf{G}_{j,k+\frac{1}{2}} - \mathbf{G}_{j,k-\frac{1}{2}}\right]$$

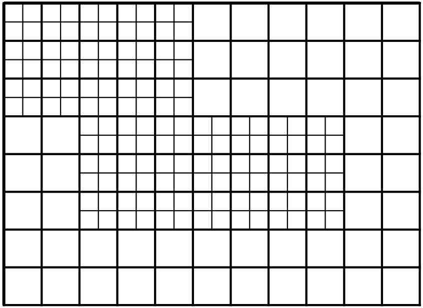
  only for single patch necessary
- \+ Efficient cache-reuse / vectorization possible
- \- Cluster-algorithm necessary
- Papers: [Deiterding, 2011a, Deiterding et al., 2009b, Deiterding et al., 2007]



Level transfer / setting of ghost cells

Conservative averaging (restriction):

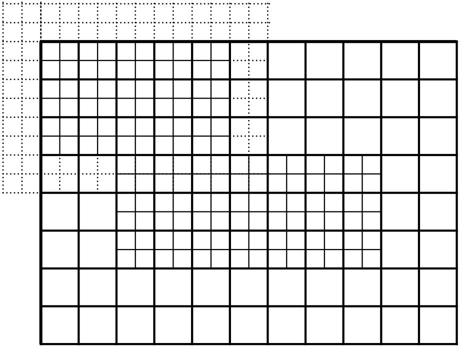
$$\hat{\mathbf{Q}}^l_{jk} := \frac{1}{(r_{l+1})^2} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{Q}^{l+1}_{v+\kappa,w+\iota}$$



Level transfer / setting of ghost cells

Conservative averaging (restriction):

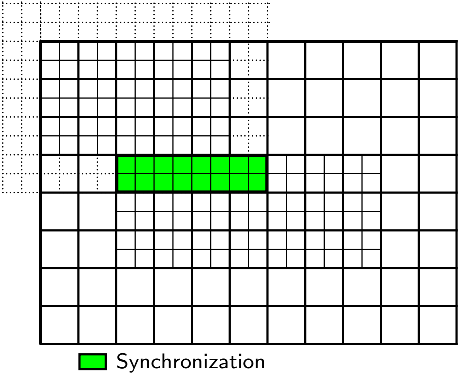
$$\hat{\mathbf{Q}}^{l}_{jk} := \frac{1}{(r_{l+1})^2} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{Q}^{l+1}_{v+\kappa,w+\iota}$$



Level transfer / setting of ghost cells

Conservative averaging (restriction):

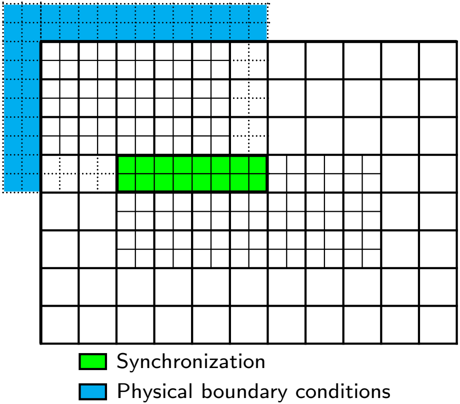
$$\hat{\mathbf{Q}}^l_{jk} := \frac{1}{(r_{l+1})^2} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{Q}^{l+1}_{v+\kappa,w+\iota}$$



Level transfer / setting of ghost cells

Conservative averaging (restriction):

$$\hat{\mathbf{Q}}^{l}_{jk} := \frac{1}{(r_{l+1})^2} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{Q}^{l+1}_{v+\kappa,w+\iota}$$



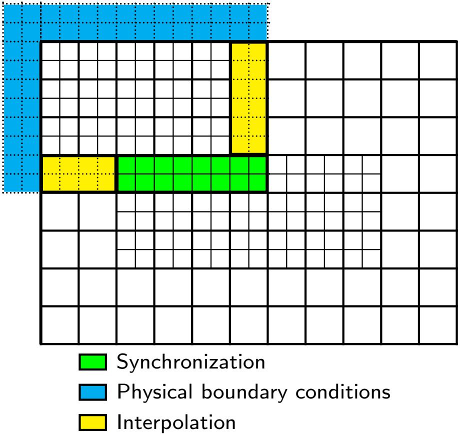
Level transfer / setting of ghost cells

Conservative averaging (restriction):

$$\hat{\mathbf{Q}}^l_{jk} := \frac{1}{(r_{l+1})^2} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{Q}^{l+1}_{v+\kappa,w+\iota}$$

Bilinear interpolation (prolongation):

$$\check{\mathbf{Q}}^{l+1}_{vw} := (1-f_1)(1-f_2)\,\mathbf{Q}^l_{j-1,k-1} + f_1(1-f_2)\,\mathbf{Q}^l_{j,k-1} + (1-f_1)f_2\,\mathbf{Q}^l_{j-1,k} + f_1 f_2\,\mathbf{Q}^l_{jk}$$

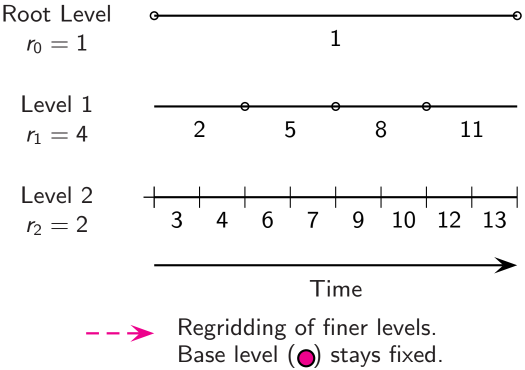


For boundary conditions: linear time interpolation

$$\tilde{\mathbf{Q}}^{l+1}(t+\kappa\Delta t_{l+1}) := \left(1-\frac{\kappa}{r_{l+1}}\right)\check{\mathbf{Q}}^{l+1}(t) + \frac{\kappa}{r_{l+1}}\check{\mathbf{Q}}^{l+1}(t+\Delta t_l) \quad \text{for } \kappa = 0,\ldots r_{l+1}$$

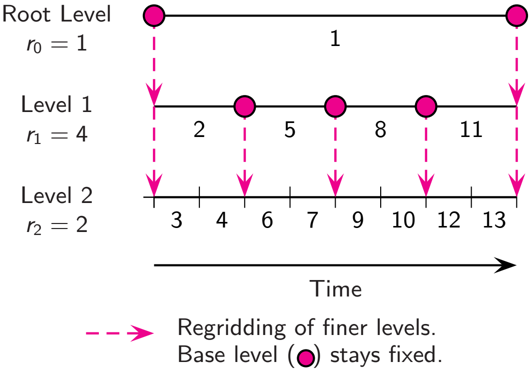
Recursive integration order

- Space-time interpolation of coarse data to set I_l^s, $l > 0$



Recursive integration order

- Space-time interpolation of coarse data to set I_l^s, $l > 0$
- Regridding:
  - Creation of new grids, copy existing cells on level $l > 0$
  - Spatial interpolation to initialize new cells on level $l > 0$

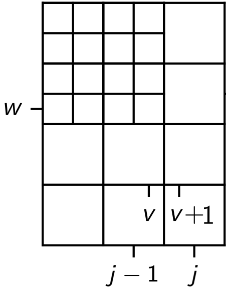


Conservative flux correction

Example: Cell j, k

$$\check{\mathbf{Q}}^l_{jk}(t+\Delta t_l) = \mathbf{Q}^l_{jk}(t) - \frac{\Delta t_l}{\Delta x_{1,l}} \left(\mathbf{F}^l_{j+\frac{1}{2},k} - \frac{1}{r^2_{l+1}} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{F}^{l+1}_{v+\frac{1}{2},w+\iota}(t+\kappa\Delta t_{l+1}) \right)$$
$$- \frac{\Delta t_l}{\Delta x_{2,l}} \left(\mathbf{G}^l_{j,k+\frac{1}{2}} - \mathbf{G}^l_{j,k-\frac{1}{2}} \right)$$

Correction pass:



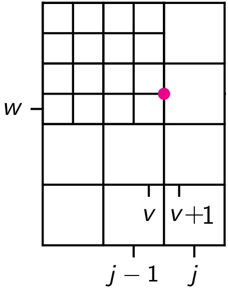
Conservative flux correction

Example: Cell j, k

$$\check{\mathbf{Q}}^l_{jk}(t+\Delta t_l) = \mathbf{Q}^l_{jk}(t) - \frac{\Delta t_l}{\Delta x_{1,l}} \left(\mathbf{F}^l_{j+\frac{1}{2},k} - \frac{1}{r^2_{l+1}} \sum_{\kappa=0}^{r_{l+1}-1} \sum_{\iota=0}^{r_{l+1}-1} \mathbf{F}^{l+1}_{v+\frac{1}{2},w+\iota}(t+\kappa\Delta t_{l+1}) \right) - \frac{\Delta t_l}{\Delta x_{2,l}} \left(\mathbf{G}^l_{j,k+\frac{1}{2}} - \mathbf{G}^l_{j,k-\frac{1}{2}} \right)$$

Correction pass:

1. $\delta \mathbf{F}^{l+1}_{j-\frac{1}{2},k} := -\mathbf{F}^l_{j-\frac{1}{2},k}$



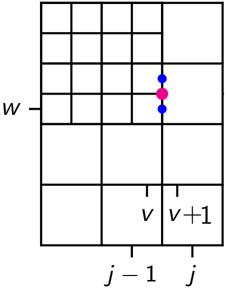
Conservative flux correction

Example: Cell $j,\ k$

$$\check{\mathbf{Q}}^l_{jk}(t+\Delta t_l) = \mathbf{Q}^l_{jk}(t) - \frac{\Delta t_l}{\Delta x_{1,l}}\left(\mathbf{F}^l_{j+\frac{1}{2},k} - \frac{1}{r^2_{l+1}}\sum_{\kappa=0}^{r_{l+1}-1}\sum_{\iota=0}^{r_{l+1}-1}\mathbf{F}^{l+1}_{v+\frac{1}{2},w+\iota}(t+\kappa\Delta t_{l+1})\right)$$
$$- \frac{\Delta t_l}{\Delta x_{2,l}}\left(\mathbf{G}^l_{j,k+\frac{1}{2}} - \mathbf{G}^l_{j,k-\frac{1}{2}}\right)$$

Correction pass:

1. $\delta\mathbf{F}^{l+1}_{j-\frac{1}{2},k} := -\mathbf{F}^l_{j-\frac{1}{2},k}$
2. $\delta\mathbf{F}^{l+1}_{j-\frac{1}{2},k} := \delta\mathbf{F}^{l+1}_{j-\frac{1}{2},k} + \frac{1}{r^2_{l+1}}\sum_{\iota=0}^{r_{l+1}-1}\mathbf{F}^{l+1}_{v+\frac{1}{2},w+\iota}(t+\kappa\Delta t_{l+1})$



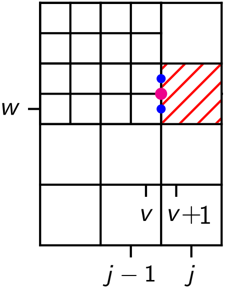
Conservative flux correction

Example: Cell j, k

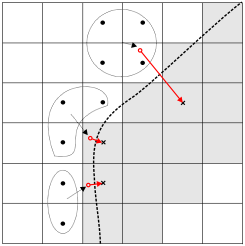
$$\check{\mathbf{Q}}^l_{jk}(t+\Delta t_l) = \mathbf{Q}^l_{jk}(t) - \frac{\Delta t_l}{\Delta x_{1,l}}\left(\mathbf{F}^l_{j+\frac{1}{2},k} - \frac{1}{r^2_{l+1}}\sum_{\kappa=0}^{r_{l+1}-1}\sum_{\iota=0}^{r_{l+1}-1}\mathbf{F}^{l+1}_{v+\frac{1}{2},w+\iota}(t+\kappa\Delta t_{l+1})\right) - \frac{\Delta t_l}{\Delta x_{2,l}}\left(\mathbf{G}^l_{j,k+\frac{1}{2}} - \mathbf{G}^l_{j,k-\frac{1}{2}}\right)$$

Correction pass:

1. $\delta\mathbf{F}^{l+1}_{j-\frac{1}{2},k} := -\mathbf{F}^l_{j-\frac{1}{2},k}$
2. $\delta\mathbf{F}^{l+1}_{j-\frac{1}{2},k} := \delta\mathbf{F}^{l+1}_{j-\frac{1}{2},k} + \frac{1}{r^2_{l+1}}\sum_{\iota=0}^{r_{l+1}-1}\mathbf{F}^{l+1}_{v+\frac{1}{2},w+\iota}(t+\kappa\Delta t_{l+1})$
3. $\check{\mathbf{Q}}^l_{jk}(t+\Delta t_l) := \mathbf{Q}^l_{jk}(t+\Delta t_l) + \frac{\Delta t_l}{\Delta x_{1,l}}\,\delta\mathbf{F}^{l+1}_{j-\frac{1}{2},k}$

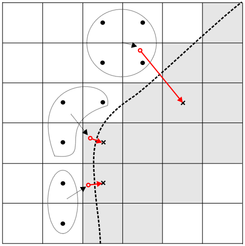


Level-set method for boundary embedding



- Implicit boundary representation via distance function φ, normal $\mathbf{n} = \nabla\varphi/|\nabla\varphi|$
- Complex boundary moving with local velocity $\mathbf{w}$, treat interface as moving rigid wall [Deiterding et al., 2007]
- Construction of values in embedded boundary cells by interpolation / extrapolation [Deiterding, 2009, Deiterding, 2011a]
- Creation of level set from triangulated surface data with closest-point-transform (CPT) algorithm [Mauch, 2003, Deiterding et al., 2006]

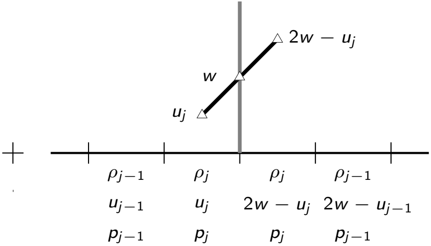
Level-set method for boundary embedding



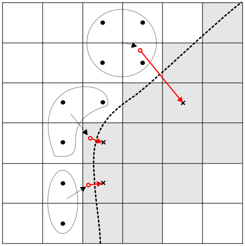
Interpolate / constant value extrapolate values at

$$\tilde{\mathbf{x}} = \mathbf{x} + 2\varphi\mathbf{n}$$

- Implicit boundary representation via distance function φ, normal $\mathbf{n} = \nabla\varphi / |\nabla\varphi|$
- Complex boundary moving with local velocity $\mathbf{w}$, treat interface as moving rigid wall [Deiterding et al., 2007]
- Construction of values in embedded boundary cells by interpolation / extrapolation [Deiterding, 2009, Deiterding, 2011a]
- Creation of level set from triangulated surface data with closest-point-transform (CPT) algorithm [Mauch, 2003, Deiterding et al., 2006]



Level-set method for boundary embedding



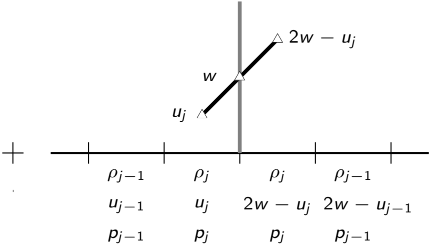
- Implicit boundary representation via distance function φ, normal $\mathbf{n} = \nabla\varphi/|\nabla\varphi|$
- Complex boundary moving with local velocity $\mathbf{w}$, treat interface as moving rigid wall [Deiterding et al., 2007]
- Construction of values in embedded boundary cells by interpolation / extrapolation [Deiterding, 2009, Deiterding, 2011a]
- Creation of level set from triangulated surface data with closest-point-transform (CPT) algorithm [Mauch, 2003, Deiterding et al., 2006]

Interpolate / constant value extrapolate values at

$$\tilde{\mathbf{x}} = \mathbf{x} + 2\varphi\mathbf{n}$$

Velocity in ghost cells
No-slip: $\mathbf{u}' = 2\mathbf{w} - \mathbf{u}$
Slip:

$$\mathbf{u}' = (2\mathbf{w}\cdot\mathbf{n} - \mathbf{u}\cdot\mathbf{n})\mathbf{n} + (\mathbf{u}\cdot\mathbf{t})\mathbf{t}$$
$$= 2\left((\mathbf{w}-\mathbf{u})\cdot\mathbf{n}\right)\mathbf{n} + \mathbf{u}$$



Parallelization

Rigorous domain decomposition

- Data of all levels resides on same node
- Grid hierarchy defines unique "floor-plan"
- Workload estimation

$$\mathcal{W}(\Omega) = \sum_{l=0}^{l_{\max}} \left[\mathcal{N}_l(G_l \cap \Omega) \prod_{\kappa=0}^{l} r_\kappa \right]$$

Parallelization

Rigorous domain decomposition

- Data of all levels resides on same node
- Grid hierarchy defines unique "floor-plan"
- Workload estimation

$$\mathcal{W}(\Omega) = \sum_{l=0}^{l_{\max}} \left[\mathcal{N}_l(G_l \cap \Omega) \prod_{\kappa=0}^{l} r_\kappa \right]$$

- Parallel operations
  - Synchronization of ghost cells
  - Redistribution of data blocks within regridding operation
  - Flux correction of coarse grid cells
- Dynamic partitioning with space-filling curve

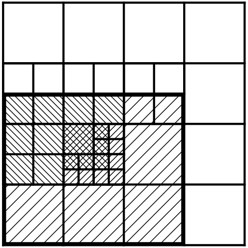
Parallelization

Rigorous domain decomposition

- Data of all levels resides on same node
- Grid hierarchy defines unique "floor-plan"
- Workload estimation

$$\mathcal{W}(\Omega) = \sum_{l=0}^{l_{\max}} \left[\mathcal{N}_l(G_l \cap \Omega) \prod_{\kappa=0}^{l} r_\kappa \right]$$

- Parallel operations
  - Synchronization of ghost cells
  - Redistribution of data blocks within regridding operation
  - Flux correction of coarse grid cells
- Dynamic partitioning with space-filling curve



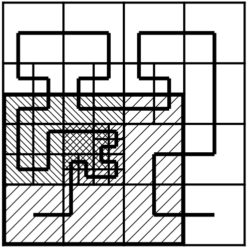
Parallelization

Rigorous domain decomposition

- Data of all levels resides on same node
- Grid hierarchy defines unique "floor-plan"
- Workload estimation

$$\mathcal{W}(\Omega) = \sum_{l=0}^{l_{\max}} \left[\mathcal{N}_l(G_l \cap \Omega) \prod_{\kappa=0}^{l} r_\kappa \right]$$

- Parallel operations
  - Synchronization of ghost cells
  - Redistribution of data blocks within regridding operation
  - Flux correction of coarse grid cells
- Dynamic partitioning with space-filling curve



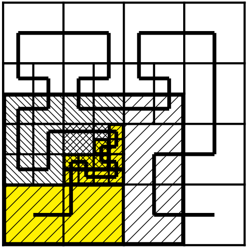
Parallelization

Rigorous domain decomposition

- Data of all levels resides on same node
- Grid hierarchy defines unique "floor-plan"
- Workload estimation

$$\mathcal{W}(\Omega) = \sum_{l=0}^{l_{\max}} \left[\mathcal{N}_l(G_l \cap \Omega) \prod_{\kappa=0}^{l} r_\kappa \right]$$

- Parallel operations
  - Synchronization of ghost cells
  - Redistribution of data blocks within regridding operation
  - Flux correction of coarse grid cells
- Dynamic partitioning with space-filling curve

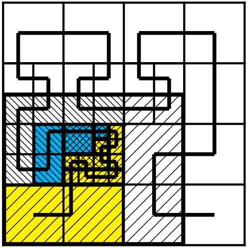


Parallelization

Rigorous domain decomposition

- Data of all levels resides on same node
- Grid hierarchy defines unique "floor-plan"
- Workload estimation

$$\mathcal{W}(\Omega) = \sum_{l=0}^{l_{\max}} \left[\mathcal{N}_l(G_l \cap \Omega) \prod_{\kappa=0}^{l} r_\kappa \right]$$



- Parallel operations
  - Synchronization of ghost cells
  - Redistribution of data blocks within regridding operation
  - Flux correction of coarse grid cells
- Dynamic partitioning with space-filling curve

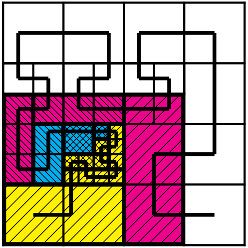
Parallelization

Rigorous domain decomposition

- Data of all levels resides on same node
- Grid hierarchy defines unique "floor-plan"
- Workload estimation

$$\mathcal{W}(\Omega) = \sum_{l=0}^{l_{\max}} \left[\mathcal{N}_l(G_l \cap \Omega) \prod_{\kappa=0}^{l} r_\kappa \right]$$

- Parallel operations
  - Synchronization of ghost cells
  - Redistribution of data blocks within regridding operation
  - Flux correction of coarse grid cells
- Dynamic partitioning with space-filling curve



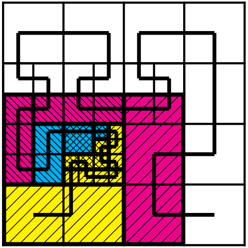
Parallelization

Rigorous domain decomposition

- Data of all levels resides on same node
- Grid hierarchy defines unique "floor-plan"
- Workload estimation

$$\mathcal{W}(\Omega) = \sum_{l=0}^{l_{\max}} \left[\mathcal{N}_l(G_l \cap \Omega) \prod_{\kappa=0}^{l} r_\kappa \right]$$

- Parallel operations
  - Synchronization of ghost cells
  - Redistribution of data blocks within regridding operation
  - Flux correction of coarse grid cells
- Dynamic partitioning with space-filling curve



[Deiterding, 2005, Deiterding, 2011a]

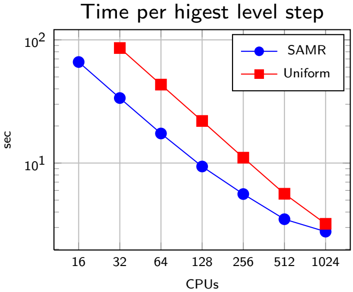
AMROC framework and most important patch solvers

- Implements described algorithms and facilitates easy exchange of the block-based numerical scheme
- Shock-induced combustion with detailed chemistry: [Deiterding, 2003, Deiterding and Bader, 2005, Deiterding, 2011b, Cai et al., 2016, Cai et al., 2018]
- Hybrid WENO methods for LES and DNS: [Pantano et al., 2007, Lombardini and Deiterding, 2010, Ziegler et al., 2011, Cerminara et al., 2018]
- Lattice Boltzmann method for LES: [Fragner and Deiterding, 2016, Feldhusen et al., 2016, Deiterding and Wood, 2016]
- FSI deformation from water hammer: [Cirak et al., 2007, Deiterding et al., 2009a, Perotti et al., 2013, Wan et al., 2017]
- Level-set method for Eulerian solid mechanics: [Barton et al., 2013]
- Ideal magneto-hydrodynamics: [Gomes et al., 2015, Souza Lopes et al., 2018]
- $\sim 500,000$ LOC in C++, C, Fortran-77, Fortran-90
- V2.0 plus FSI coupling routines as open source at http://www.vtf.website
- Used here V3.0 with significantly enhanced parallelization (V2.1 not released)

AMROC strong scalability tests

3D wave propagation method with Roe scheme: spherical blast wave

- Tests run IBM BG/P (mode VN)



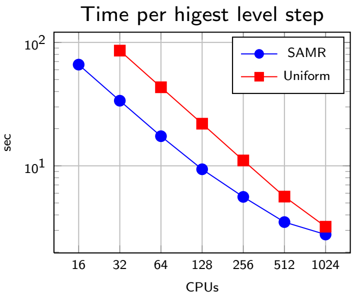
$64 \times 32 \times 32$ base grid, 2 additional levels with factors 2, 4; uniform $512 \times 256 \times 256 = 33.6 \cdot 10^6$ cells

Level	Grids	Cells
0	1709	65,536
1	1735	271,048
2	2210	7,190,208

AMROC strong scalability tests

3D wave propagation method with Roe scheme: spherical blast wave

- Tests run IBM BG/P (mode VN)

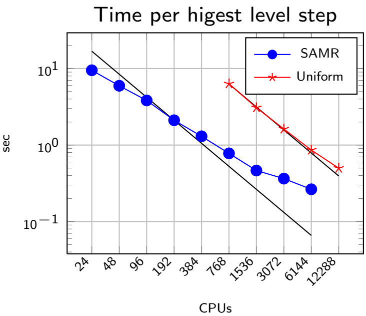


$64 \times 32 \times 32$ base grid, 2 additional levels with factors 2, 4; uniform $512 \times 256 \times 256 = 33.6 \cdot 10^6$ cells

Level	Grids	Cells
0	1709	65,536
1	1735	271,048
2	2210	7,190,208

3D SRT-lattice Boltzmann scheme: flow over rough surface of $19 \times 13 \times 2$ spheres

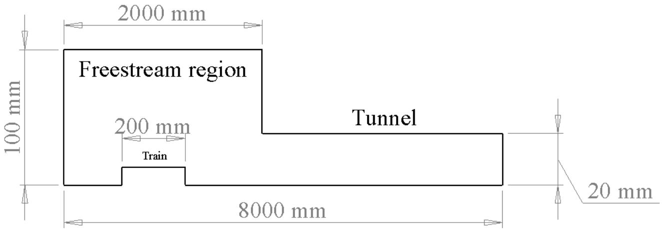
- Tests run Cray XC30m (Archer)



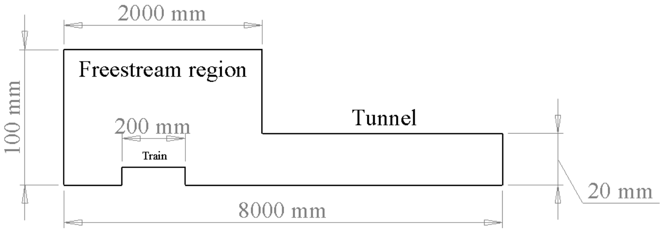
$360 \times 240 \times 108$ base grid, 2 additional levels with factors 2, 4; uniform $1440 \times 1920 \times 432 = 1.19 \cdot 10^9$ cells

Level	Grids	Cells
0	788	9,331,200
1	21367	24,844,504
2	1728	10,838,016

Laboratory tunnel simulator [Zonglin et al., 2002]



Laboratory tunnel simulator [Zonglin et al., 2002]

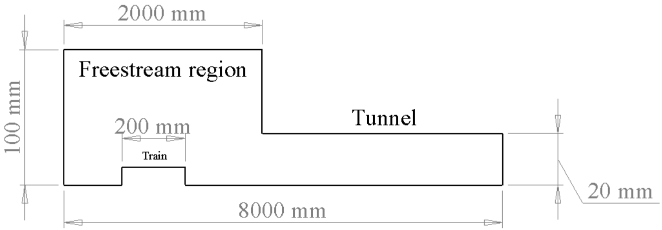


Model solves the inviscid Euler equations

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{u}) = 0$$
$$\partial_t(\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla p = 0$$
$$\partial_t(\rho E) + \nabla \cdot ((\rho E + p)\mathbf{u}) = 0$$

with $p = (\gamma - 1)(\rho E - \frac{1}{2}\rho \mathbf{u}^T \mathbf{u})$

Laboratory tunnel simulator [Zonglin et al., 2002]



Model solves the inviscid Euler equations

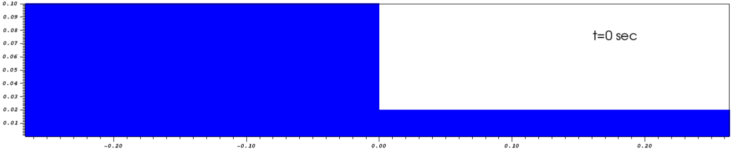
$$\begin{aligned}\partial_t \rho + \nabla \cdot (\rho \mathbf{u}) = 0 \\ \partial_t(\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla p = 0 \\ \partial_t(\rho E) + \nabla \cdot ((\rho E + p)\mathbf{u}) = 0\end{aligned}$$

with $p = (\gamma - 1)(\rho E - \frac{1}{2}\rho \mathbf{u}^T \mathbf{u})$

- Two-dimensional axi-symmetric computation
- $p_0 = 100\,\mathrm{kPa}$, $\rho_0 = 1.225\,\mathrm{kg/m^3}$, $\gamma = 1.4$
- Roe shock-capturing scheme blended with HLL
- 2nd order accuracy achieved with MUSCL-Hancock method

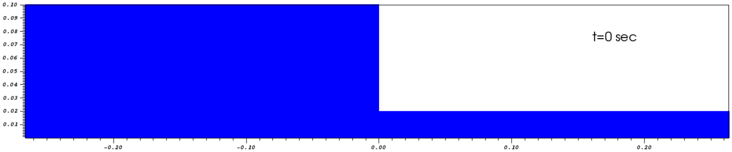
Basic phenomena – $v_0 = 100\,\mathrm{m/s}$

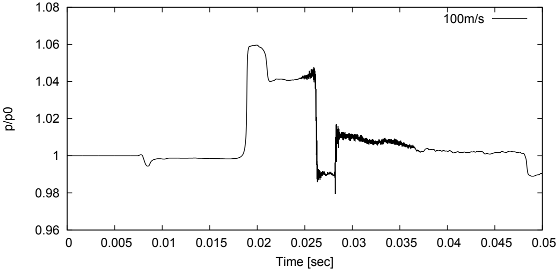
- 800×25 mesh with Cartesian cut-out $(200, 5)$ to $(800, 25)$
- 2 level of additional refinement by factor 2



Basic phenomena – $v_0 = 100\,\mathrm{m/s}$

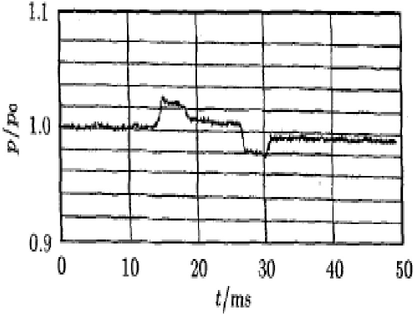
- 800×25 mesh with Cartesian cut-out $(200, 5)$ to $(800, 25)$
- 2 level of additional refinement by factor 2

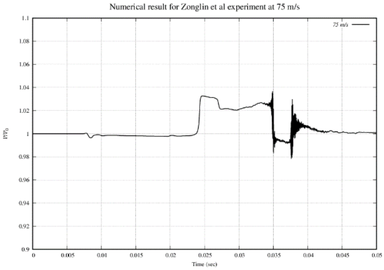




Pressure record at location $(1020\,\mathrm{mm}, 20\,\mathrm{mm})$ inside tunnel

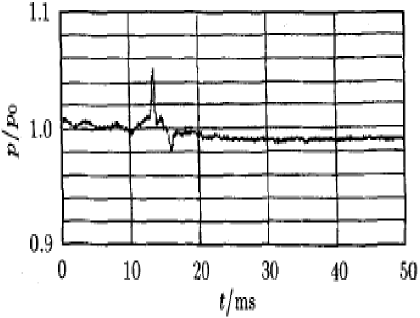
Comparison with experiment – I

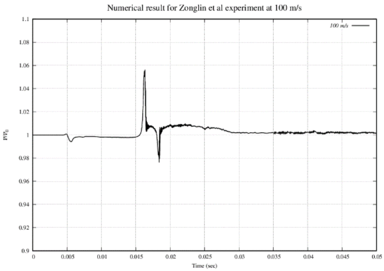




Pressure record at $(1020\,\mathrm{mm}, 20\,\mathrm{mm})$ for $v_0 = 75\,\mathrm{m/s}$. Experiment (left) and AMROC (right)

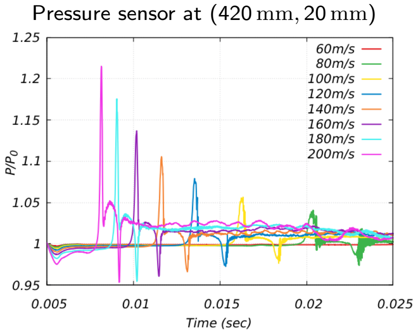
Comparison with experiment – I

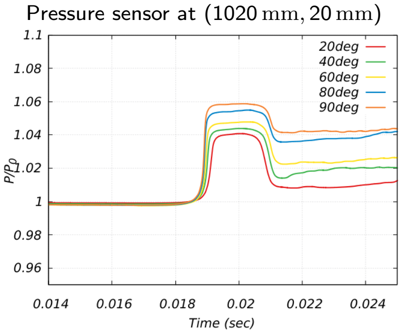




Pressure record at $(40\,\mathrm{mm}, 20\,\mathrm{mm})$ for model velocity $v_0 = 100\,\mathrm{m/s}$. Experiment (left) and AMROC (right)

Variation of velocity and nose half angle





- Dependence on v_0^2 is the dynamic pressure influence (left)
- For constant blockage ratio and body velocity, using more pointed noses alleviates the maximal pressure level (right, nose half angle varied)
- For $v_0 \approx 140\,\mathrm{m/s}$ a shock wave (tunnel boom) can be observed. Sharper noses also delay this phenomenon.

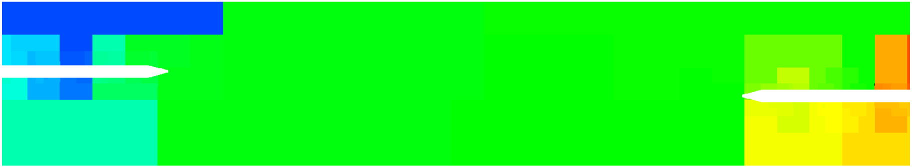
NGT2 prototype setup

- Next Generation Train 2 (NGT2) geometry by the German Aerospace Centre (DLR) [Fragner and Deiterding, 2016, Fragner and Deiterding, 2017]
- Mirrored train head of length $\sim 60\,\mathrm{m}$, no wheels or tracks, train models $0.17\,\mathrm{m}$ above ground above the ground level.
- Train velocities $100\,\mathrm{m/s}$ and $-100\,\mathrm{m/s}$, middle axis $6\,\mathrm{m}$ apart, initial distance between centers $200\,\mathrm{m}$
- Base mesh of $360 \times 40 \times 30$ for domain of $360\,\mathrm{m} \times 40\,\mathrm{m} \times 30\,\mathrm{m}$
- Two/three additional levels, refined by $r_{1,2,3} = 2$. Refinement based on pressure gradient and level set and regenerated at every coarse time step. Parallel redistribution at every level-0 time step.
- On 96 cores Intel Xeon E5-2670 $2.6\,\mathrm{GHz}$ a final $t_e = 3\,\mathrm{sec}$ was reached after $12,385\,\mathrm{sec}$ / $43,395\,\mathrm{sec}$ wall time, i.e., $330\,\mathrm{h}$ and $1157\,\mathrm{h}$ CPU

Passing in open space – AMR and dynamic distribution

Domains of three-level refinement

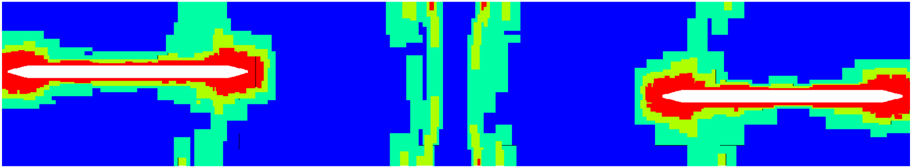
Distribution to 96 processors



Enlargement of domain center shown

Passing in open space – AMR and dynamic distribution

Domains of three-level refinement



Distribution to 96 processors

Enlargement of domain center shown

Passing in open space – AMR and dynamic distribution

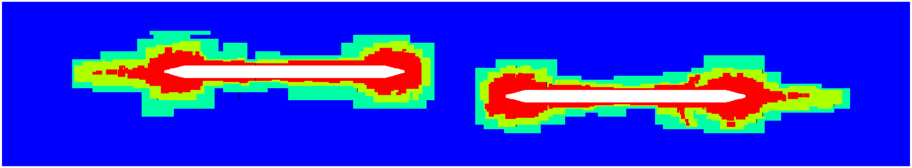
Domains of three-level refinement

Distribution to 96 processors

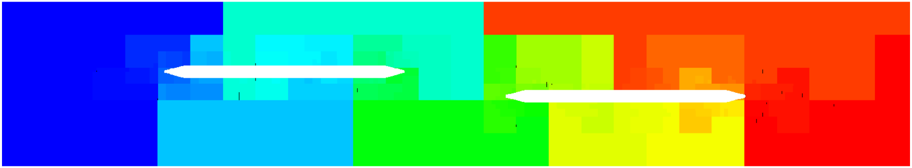
Enlargement of domain center shown

Passing in open space – AMR and dynamic distribution

Domains of three-level refinement



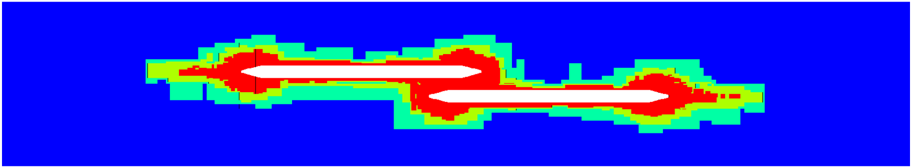
Distribution to 96 processors



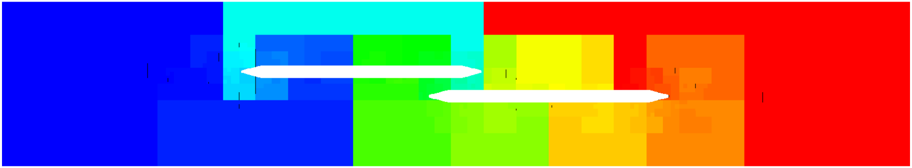
Enlargement of domain center shown

Passing in open space – AMR and dynamic distribution

Domains of three-level refinement



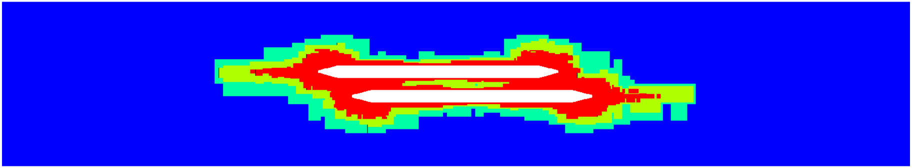
Distribution to 96 processors



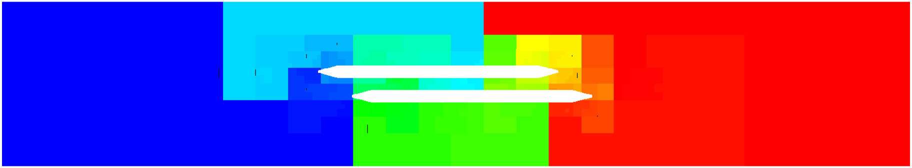
Enlargement of domain center shown

Passing in open space – AMR and dynamic distribution

Domains of three-level refinement



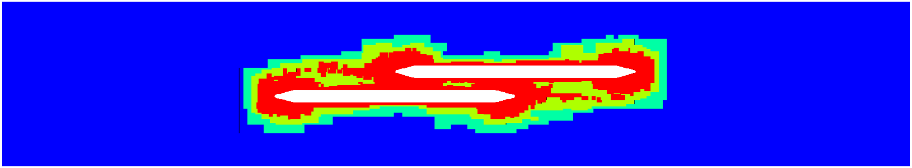
Distribution to 96 processors



Enlargement of domain center shown

Passing in open space – AMR and dynamic distribution

Domains of three-level refinement

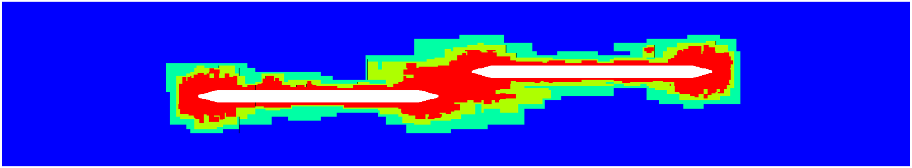


Distribution to 96 processors

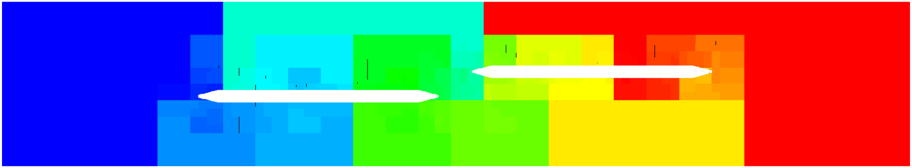
Enlargement of domain center shown

Passing in open space – AMR and dynamic distribution

Domains of three-level refinement



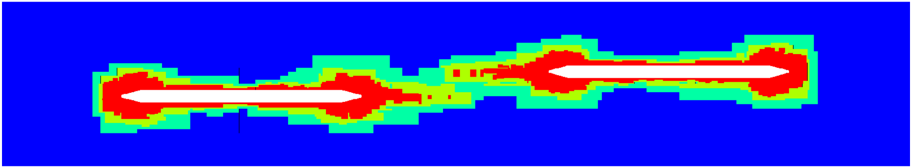
Distribution to 96 processors



Enlargement of domain center shown

Passing in open space – AMR and dynamic distribution

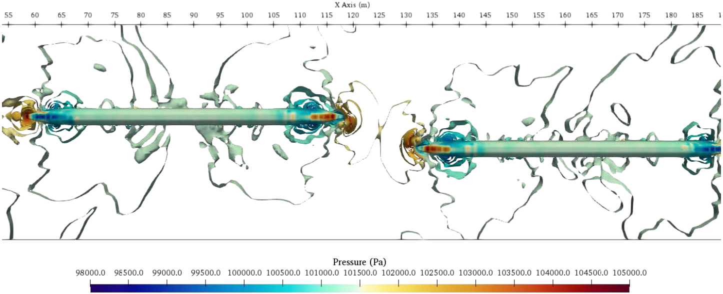
Domains of three-level refinement

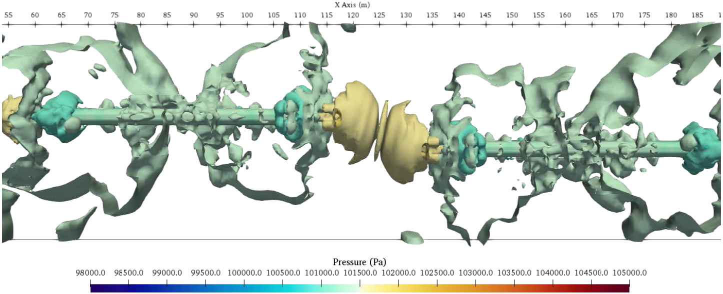


Distribution to 96 processors

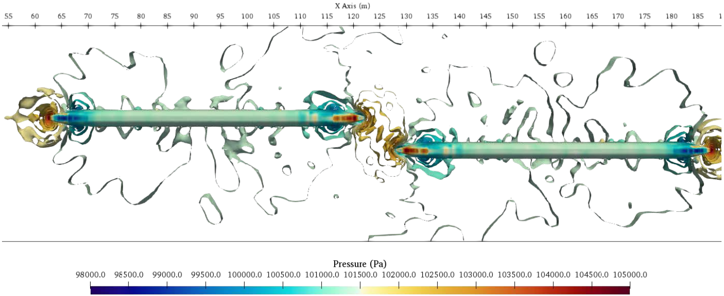
Enlargement of domain center shown

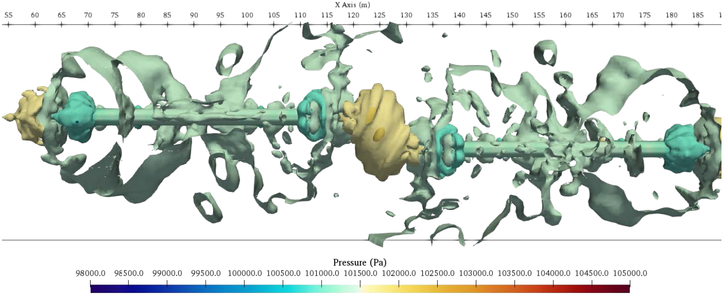
Pressure isosurfaces



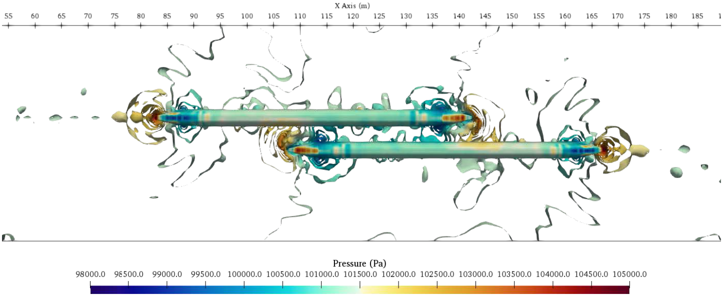


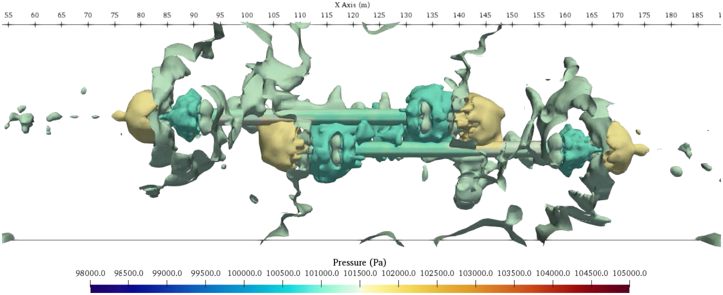
Pressure isosurfaces



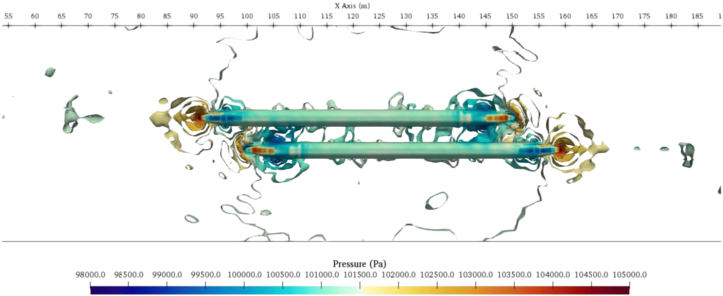


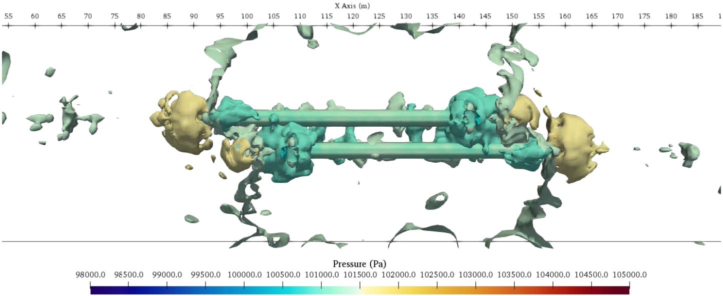
Pressure isosurfaces



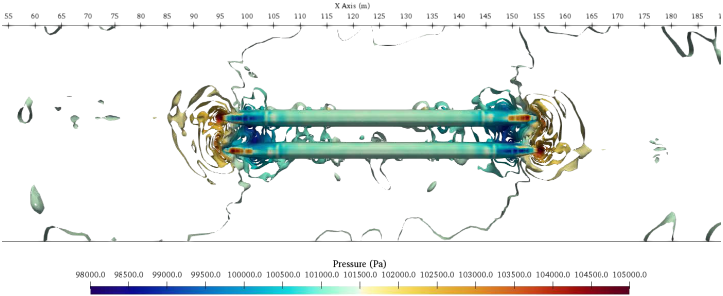


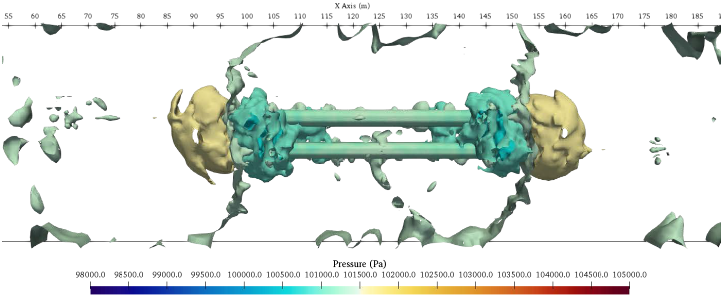
Pressure isosurfaces



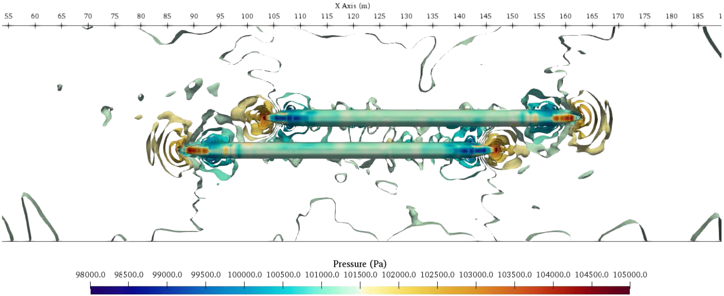


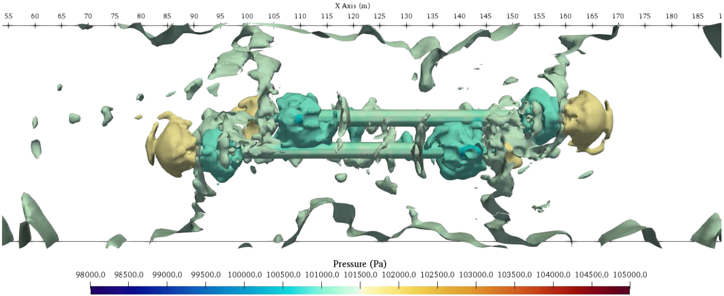
Pressure isosurfaces



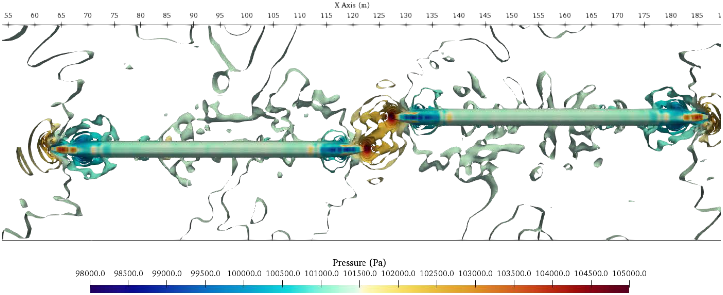


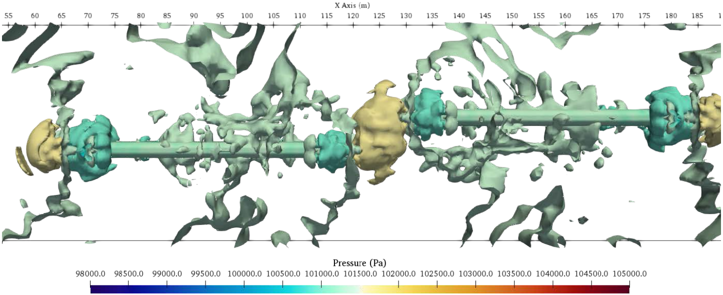
Pressure isosurfaces



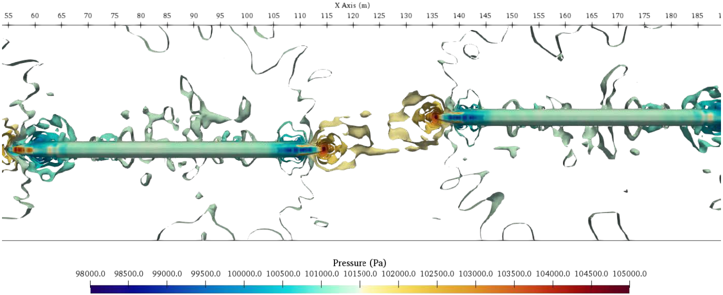


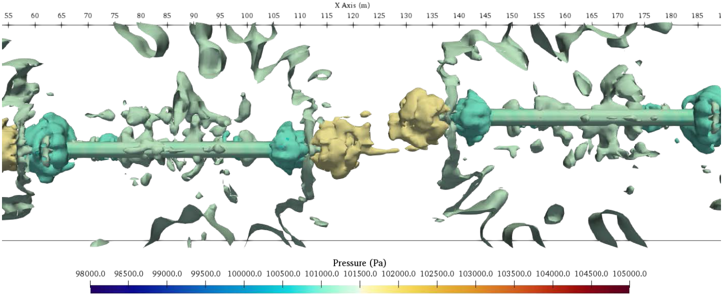
Pressure isosurfaces



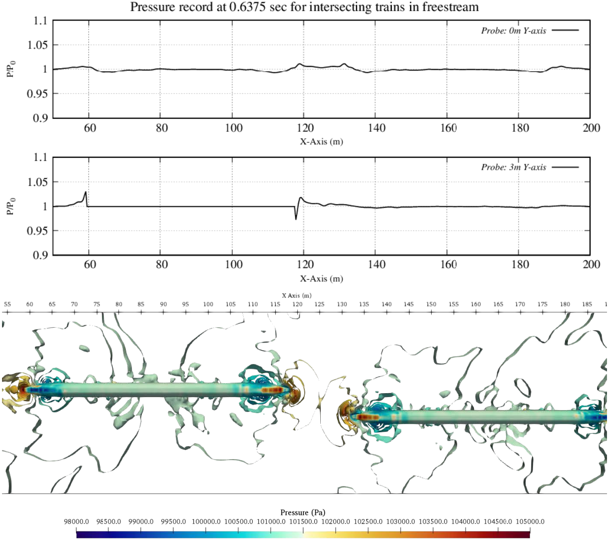


Pressure isosurfaces

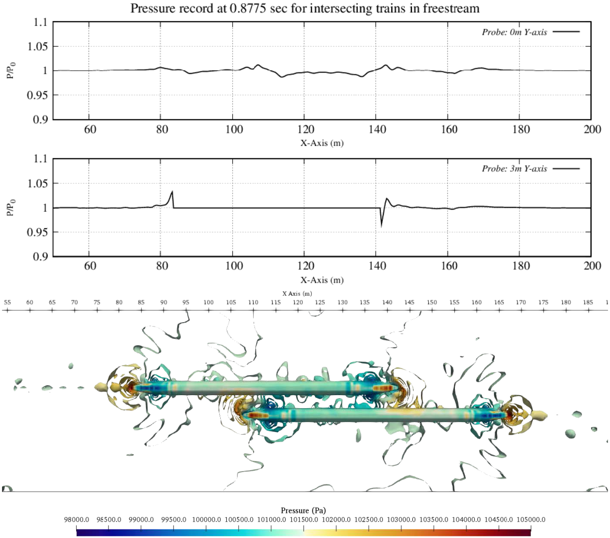




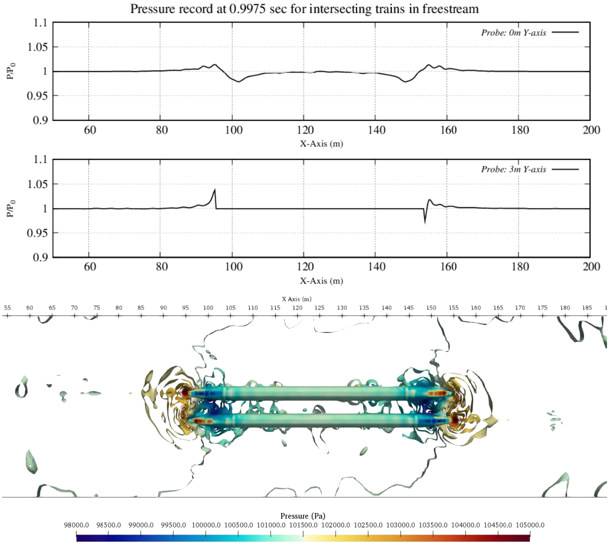
Pressure transects



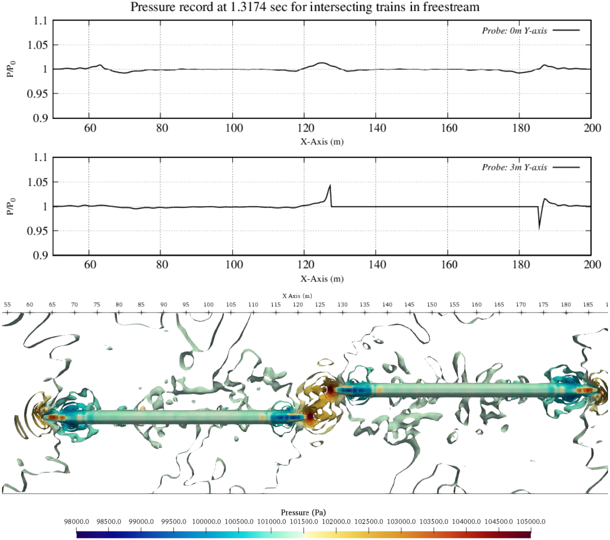
Pressure transects



Pressure transects

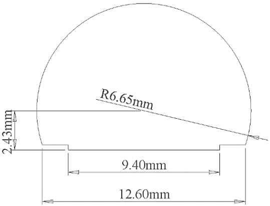


Pressure transects



Setup with realistic tunnel shape

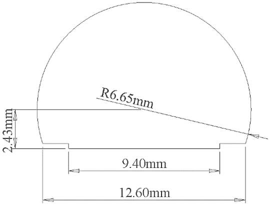
- Two NGT2 trains again at velocities $100\,\mathrm{m/s}$ and $-100\,\mathrm{m/s}$
- Prototype straight double track tunnel of $640\,\mathrm{m}$ length, initial distance between centers of trains $820\,\mathrm{m}$
- Base mesh of $1060 \times 36 \times 24$ for domain of $1060\,\mathrm{m} \times 36\,\mathrm{m} \times 24\,\mathrm{m}$, three levels refined by $r_{1,2,3} = 2$
- On 96 cores Intel Xeon E5-2670 $2.6\,\mathrm{GHz}$ a final $t_e = 5\,\mathrm{sec}$ was reached after $84,651\,\mathrm{sec}$ wall time, i.e., $2257\,\mathrm{h}$ CPU



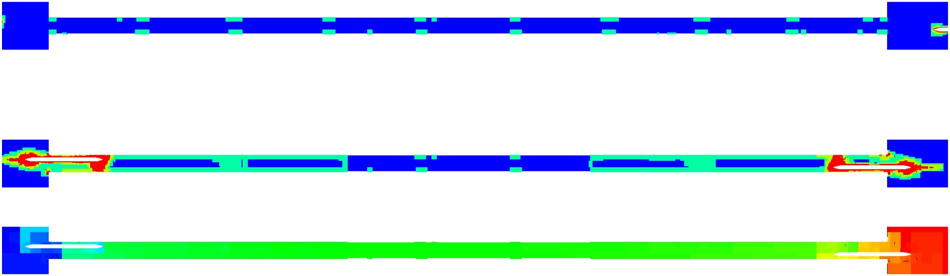
Tunnel shape

Setup with realistic tunnel shape

- Two NGT2 trains again at velocities $100\,\mathrm{m/s}$ and $-100\,\mathrm{m/s}$
- Prototype straight double track tunnel of $640\,\mathrm{m}$ length, initial distance between centers of trains $820\,\mathrm{m}$
- Base mesh of $1060 \times 36 \times 24$ for domain of $1060\,\mathrm{m} \times 36\,\mathrm{m} \times 24\,\mathrm{m}$, three levels refined by $r_{1,2,3} = 2$
- On 96 cores Intel Xeon E5-2670 $2.6\,\mathrm{GHz}$ a final $t_e = 5\,\mathrm{sec}$ was reached after $84,651\,\mathrm{sec}$ wall time, i.e., $2257\,\mathrm{h}$ CPU

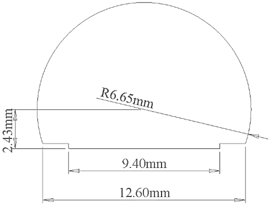


Tunnel shape

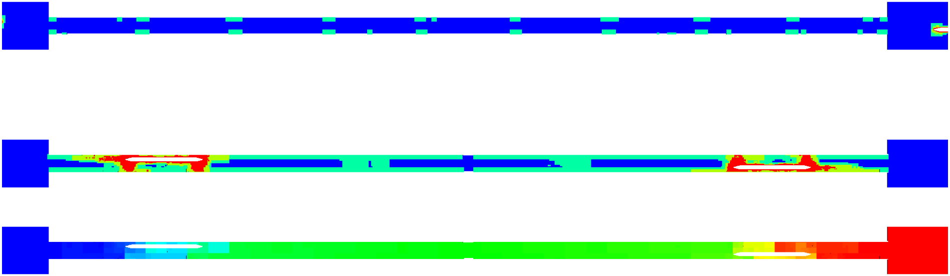


Setup with realistic tunnel shape

- Two NGT2 trains again at velocities $100\,\mathrm{m/s}$ and $-100\,\mathrm{m/s}$
- Prototype straight double track tunnel of $640\,\mathrm{m}$ length, initial distance between centers of trains $820\,\mathrm{m}$
- Base mesh of $1060 \times 36 \times 24$ for domain of $1060\,\mathrm{m} \times 36\,\mathrm{m} \times 24\,\mathrm{m}$, three levels refined by $r_{1,2,3} = 2$
- On 96 cores Intel Xeon E5-2670 $2.6\,\mathrm{GHz}$ a final $t_e = 5\,\mathrm{sec}$ was reached after $84,651\,\mathrm{sec}$ wall time, i.e., $2257\,\mathrm{h}$ CPU

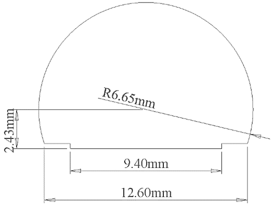


Tunnel shape

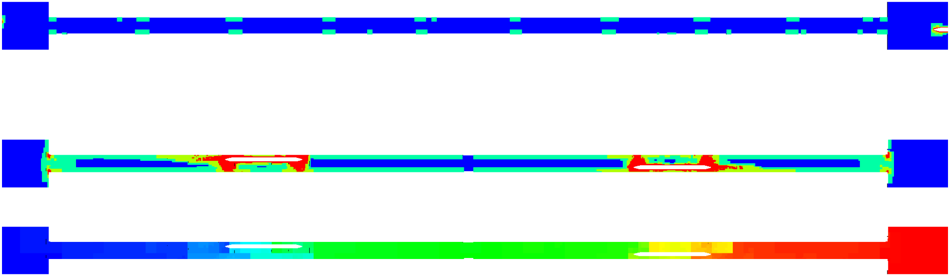


Setup with realistic tunnel shape

- Two NGT2 trains again at velocities $100\,\mathrm{m/s}$ and $-100\,\mathrm{m/s}$
- Prototype straight double track tunnel of $640\,\mathrm{m}$ length, initial distance between centers of trains $820\,\mathrm{m}$
- Base mesh of $1060 \times 36 \times 24$ for domain of $1060\,\mathrm{m} \times 36\,\mathrm{m} \times 24\,\mathrm{m}$, three levels refined by $r_{1,2,3} = 2$
- On 96 cores Intel Xeon E5-2670 $2.6\,\mathrm{GHz}$ a final $t_e = 5\,\mathrm{sec}$ was reached after $84,651\,\mathrm{sec}$ wall time, i.e., $2257\,\mathrm{h}$ CPU

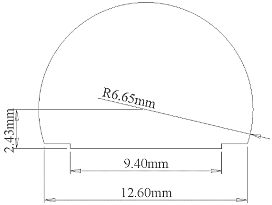


Tunnel shape

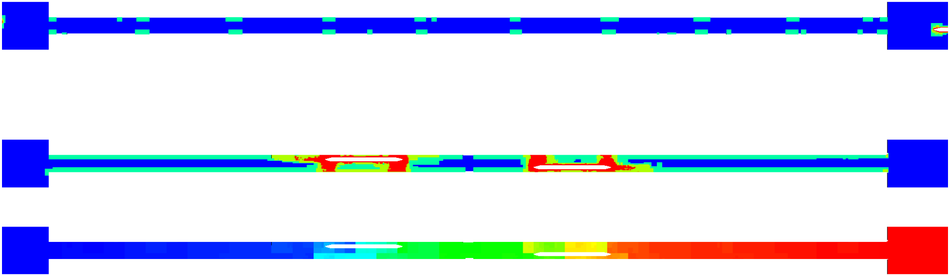


Setup with realistic tunnel shape

- Two NGT2 trains again at velocities $100\,\mathrm{m/s}$ and $-100\,\mathrm{m/s}$
- Prototype straight double track tunnel of $640\,\mathrm{m}$ length, initial distance between centers of trains $820\,\mathrm{m}$
- Base mesh of $1060 \times 36 \times 24$ for domain of $1060\,\mathrm{m} \times 36\,\mathrm{m} \times 24\,\mathrm{m}$, three levels refined by $r_{1,2,3} = 2$
- On 96 cores Intel Xeon E5-2670 $2.6\,\mathrm{GHz}$ a final $t_e = 5\,\mathrm{sec}$ was reached after $84,651\,\mathrm{sec}$ wall time, i.e., $2257\,\mathrm{h}$ CPU

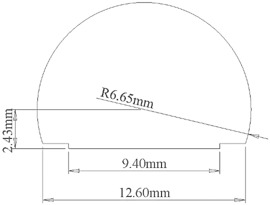


Tunnel shape

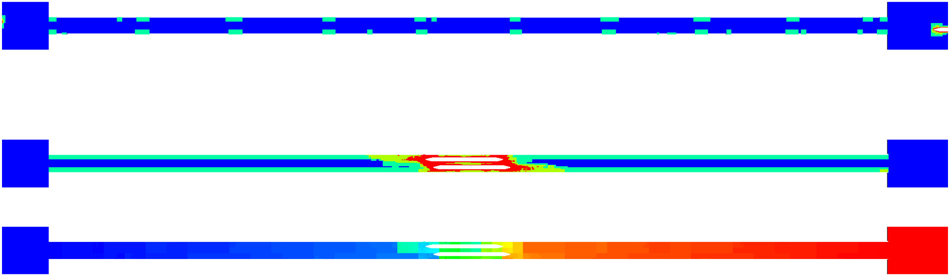


Setup with realistic tunnel shape

- Two NGT2 trains again at velocities $100\,\mathrm{m/s}$ and $-100\,\mathrm{m/s}$
- Prototype straight double track tunnel of $640\,\mathrm{m}$ length, initial distance between centers of trains $820\,\mathrm{m}$
- Base mesh of $1060 \times 36 \times 24$ for domain of $1060\,\mathrm{m} \times 36\,\mathrm{m} \times 24\,\mathrm{m}$, three levels refined by $r_{1,2,3} = 2$
- On 96 cores Intel Xeon E5-2670 $2.6\,\mathrm{GHz}$ a final $t_e = 5\,\mathrm{sec}$ was reached after $84,651\,\mathrm{sec}$ wall time, i.e., $2257\,\mathrm{h}$ CPU

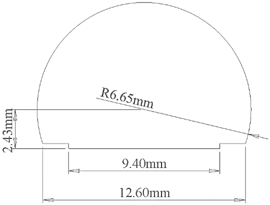


Tunnel shape

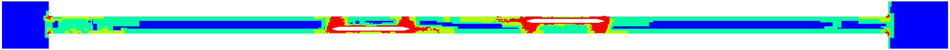


Setup with realistic tunnel shape

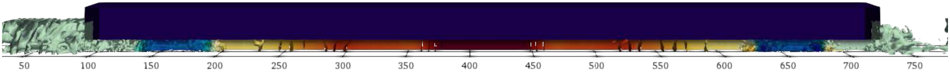
- Two NGT2 trains again at velocities $100\,\mathrm{m/s}$ and $-100\,\mathrm{m/s}$
- Prototype straight double track tunnel of $640\,\mathrm{m}$ length, initial distance between centers of trains $820\,\mathrm{m}$
- Base mesh of $1060 \times 36 \times 24$ for domain of $1060\,\mathrm{m} \times 36\,\mathrm{m} \times 24\,\mathrm{m}$, three levels refined by $r_{1,2,3} = 2$
- On 96 cores Intel Xeon E5-2670 $2.6\,\mathrm{GHz}$ a final $t_e = 5\,\mathrm{sec}$ was reached after $84,651\,\mathrm{sec}$ wall time, i.e., $2257\,\mathrm{h}$ CPU

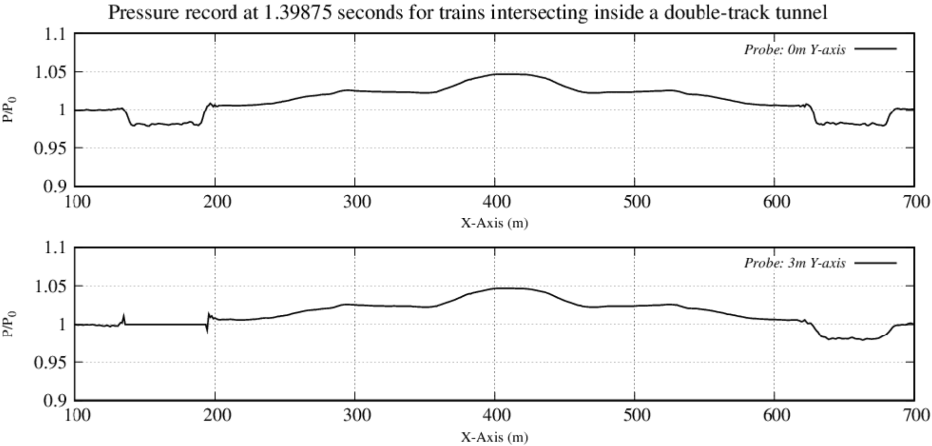


Tunnel shape

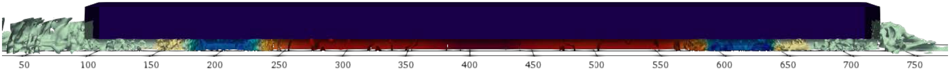


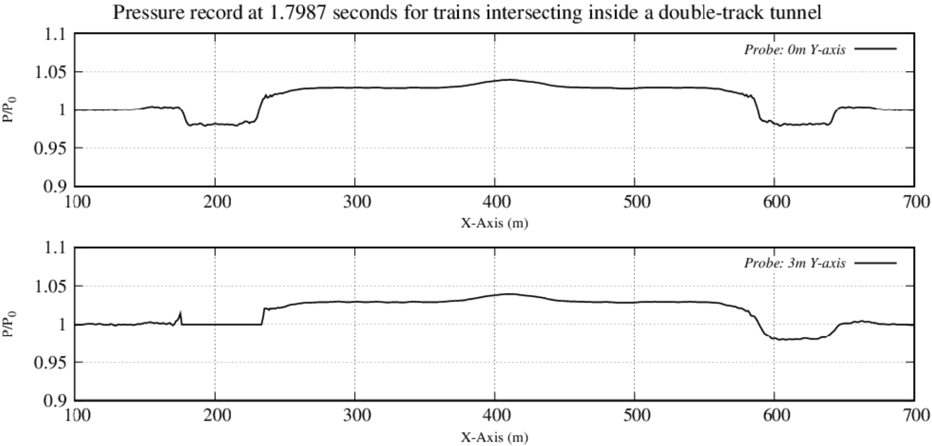
Pressure transects



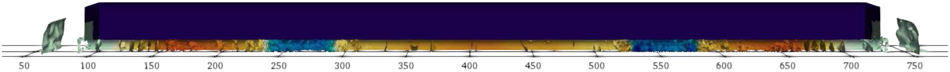


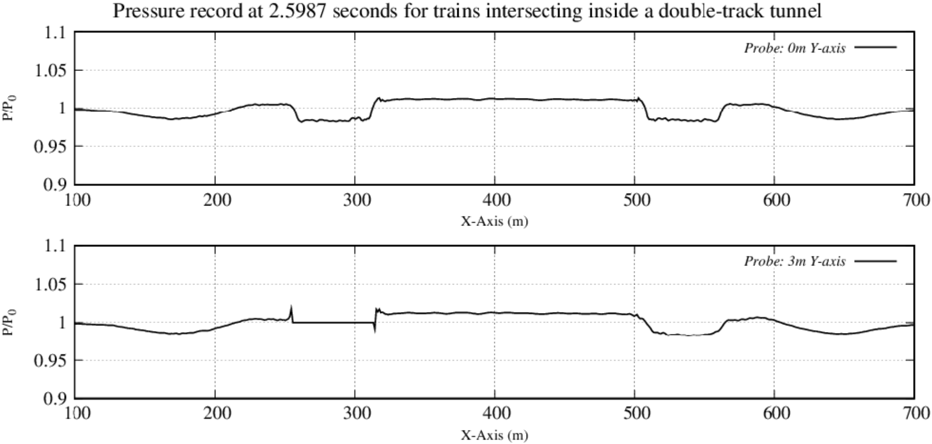
Pressure transects



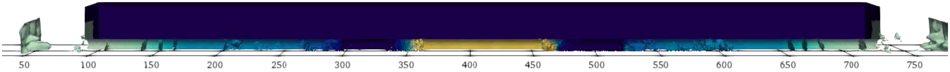


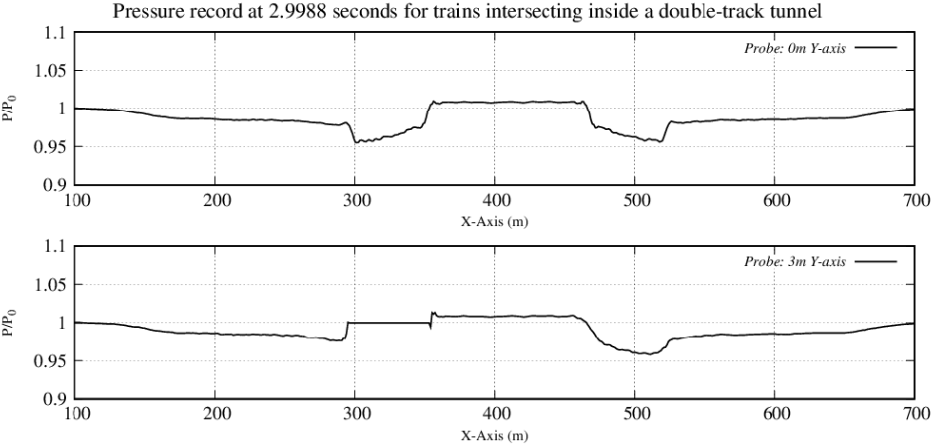
Pressure transects



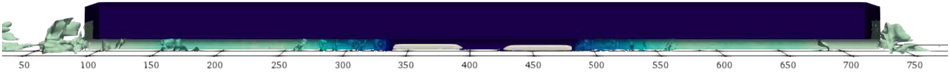


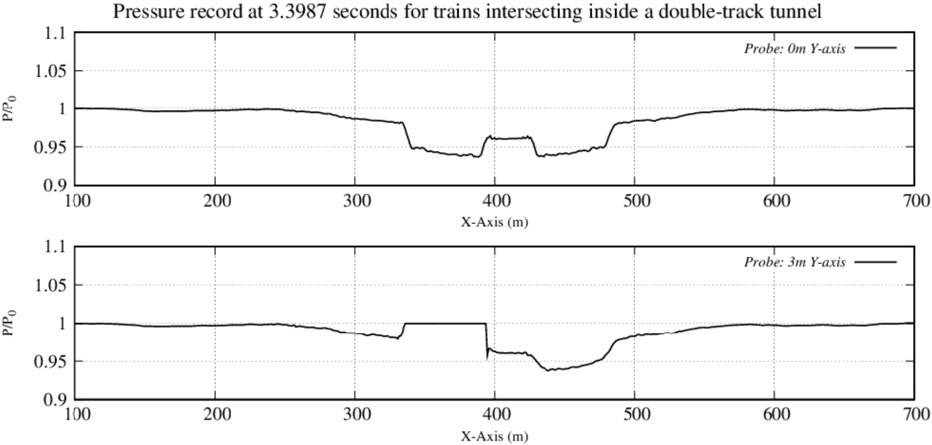
Pressure transects





Pressure transects





Thermodynamic Model

The two temperature thermodynamic model is used to model the thermodynamic nonequilibrium,

Thermodynamic Model

The two temperature thermodynamic model is used to model the thermodynamic nonequilibrium,

$$e_s(T_{tr}, T_{ve}) = e_s^t(T_{tr}) + e_s^r(T_{tr}) + e_s^v(T_{ve}) + e_s^{el}(T_{ve}) + e_s^0$$

Thermodynamic Model

The two temperature thermodynamic model is used to model the thermodynamic nonequilibrium,

$$e_s(T_{tr}, T_{ve}) = e_s^t(T_{tr}) + e_s^r(T_{tr}) + e_s^v(T_{ve}) + e_s^{el}(T_{ve}) + e_s^0$$

- Computationally efficient,
- Widely used,
- Integrated into the open source library Mutation++ [Scoggins and Magin, 2014].

Thermodynamic Model

The two temperature thermodynamic model is used to model the thermodynamic nonequilibrium,

$$e_s(T_{tr}, T_{ve}) = e_s^t(T_{tr}) + e_s^r(T_{tr}) + e_s^v(T_{ve}) + e_s^{el}(T_{ve}) + e_s^0$$

- Computationally efficient,
- Widely used,
- Integrated into the open source library Mutation++ [Scoggins and Magin, 2014].

The internal energies are calculated within the Mutation++ library using the Rigid-Rotator Harmonic-Oscillator (RRHO) model.

Governing Equations

The two temperature thermodynamic model has been implemented using the equations,

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = \mathbf{W}$$

Governing Equations

The two temperature thermodynamic model has been implemented using the equations,

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} = \mathbf{W}$$

where,

$$\mathbf{Q} = \begin{bmatrix} \rho_1 \\ \vdots \\ \rho_{N_s} \\ \rho u \\ \rho v \\ \rho e^{ve} \\ \rho E \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho_1 u \\ \vdots \\ \rho_{N_s} u \\ \rho u^2 + p \\ \rho v u \\ \rho e^{ve} u \\ (\rho E + p) u \end{bmatrix}, \quad \mathbf{G} = \begin{bmatrix} \rho_1 v \\ \vdots \\ \rho_{N_s} v \\ \rho u v \\ \rho v^2 + p \\ \rho e^{ve} v \\ (\rho E + p) v \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} \dot{w}_1 \\ \vdots \\ \dot{w}_{N_s} \\ 0 \\ 0 \\ Q_{ve} \\ 0 \end{bmatrix}.$$

Source Terms

The net species production rates,

$$\dot{w}_s = M_s \sum_{r=1}^{N_r} (\beta_{sr} - \alpha_{sr}) \left[k_{f,r} \prod_{i=1}^{N_s} \left(\frac{\rho_i}{M_i} \right)^{\alpha_{ir}} - k_{b,r} \prod_{i=1}^{N_s} \left(\frac{\rho_i}{M_i} \right)^{\beta_{ir}} \right] \text{, with}$$

$$k_{f,r}(T_c) = A_{f,r} T_c^{\eta_{f,r}} \exp\left[-\theta_r / T_c\right] \;, \quad T_C = \sqrt{T_{tr} T_{ve}}$$

Source Terms

The net species production rates,

$$\dot{w}_s = M_s \sum_{r=1}^{N_r} (\beta_{sr} - \alpha_{sr}) \left[k_{f,r} \prod_{i=1}^{N_s} \left(\frac{\rho_i}{M_i} \right)^{\alpha_{ir}} - k_{b,r} \prod_{i=1}^{N_s} \left(\frac{\rho_i}{M_i} \right)^{\beta_{ir}} \right] \text{, with}$$

$$k_{f,r}(T_c) = A_{f,r} T_c^{\eta_{f,r}} \exp\left[-\theta_r / T_c\right] \ , \ \ T_C = \sqrt{T_{tr} T_{ve}}$$

and the energy transfer rate (neutral mixture),

$$Q_{ve} = \sum_s Q_s^{T-V} + Q_s^{C-V} + Q_s^{C-el} \ ,$$

$$Q_s^{T-V} = \rho_s \frac{e_s^v(T_{tr}) - e_s^v}{\tau_{v,s}^{T-V}} \ ,$$

$$Q_s^{C-V} = c_1 \dot{w}_s e_s^v, \quad Q_s^{C-el} = c_1 \dot{w}_s e_s^{el} \ ,$$

are both calculated using the Mutation++ library.

Numerical Integration

Finite volume method with two flux schemes implemented,

Numerical Integration

Finite volume method with two flux schemes implemented,

- Van Leer's flux vector splitting method [van Leer, 1982],
- The AUSM scheme [Liou and Steffen Jr, 1993].

Numerical Integration

Finite volume method with two flux schemes implemented,

- Van Leer's flux vector splitting method [van Leer, 1982],
- The AUSM scheme [Liou and Steffen Jr, 1993].

Second order in space and time,

- The MUSCL-Hancock scheme is used for the fluxes.
- Strang splitting is used to integrate the source term.

Double Wedge

Simulation of a double wedge in a high enthalpy flow of air [Pezzella et al., 2015].

T_∞	p_∞	U_∞	M_∞	L_1	θ_1	L_2	θ_2
$710\,\mathrm{K}$	$0.78\,\mathrm{kPa}$	$3812\,\mathrm{m/s}$	7.14	$50.8\,\mathrm{mm}$	30°	$25.4\,\mathrm{mm}$	55°

Table: Double wedge geometry and experimental conditions.

Double Wedge

Simulation of a double wedge in a high enthalpy flow of air [Pezzella et al., 2015].

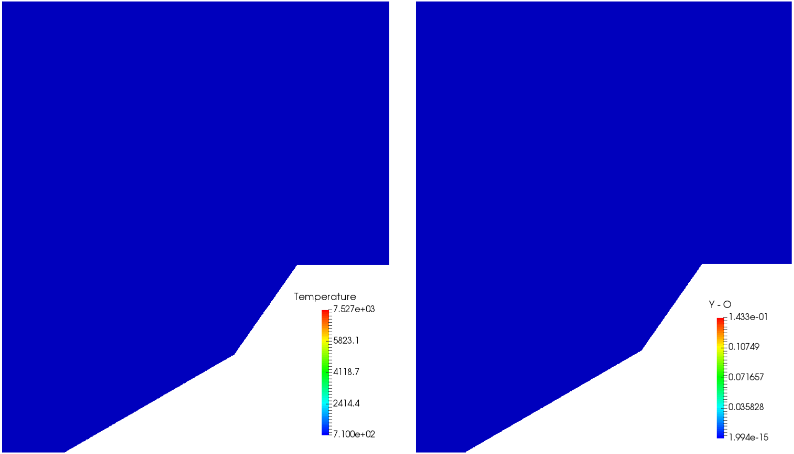
T_∞	p_∞	U_∞	M_∞	L_1	θ_1	L_2	θ_2
710 K	0.78 kPa	3812 m/s	7.14	50.8 mm	30°	25.4 mm	55°

Table: Double wedge geometry and experimental conditions.

- Five species mixture of air.
- Initial 200×200 cell mesh, with 3 levels of refinement.
- Embedded boundary used to define geometry.
- Van Leer flux scheme.
- Physical time of 242 μs.

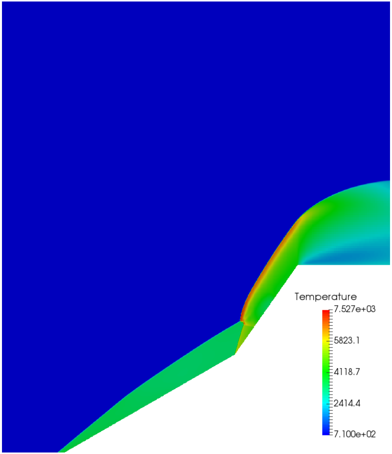
Double Wedge

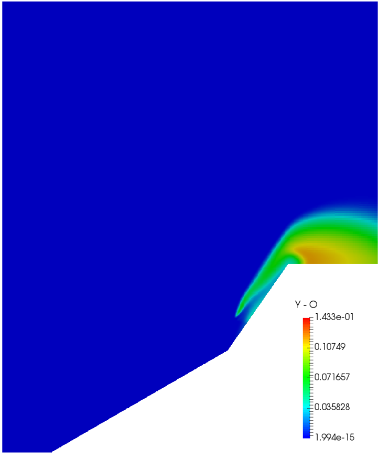
The temperature and mass fraction of atomic oxygen.



Double Wedge

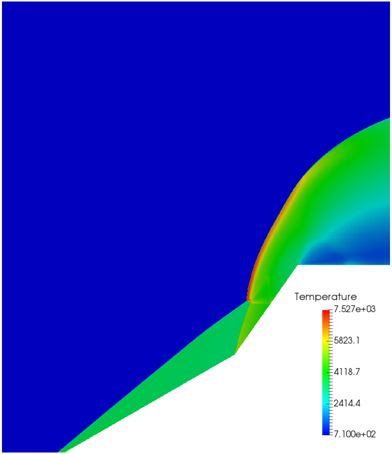
The temperature and mass fraction of atomic oxygen.

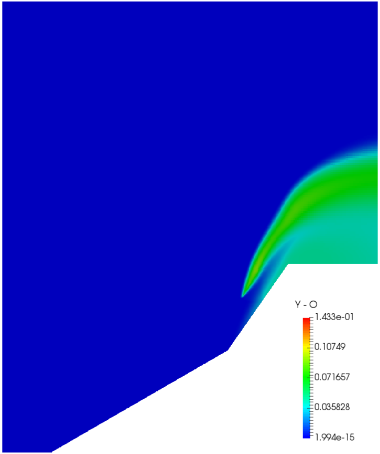




Double Wedge

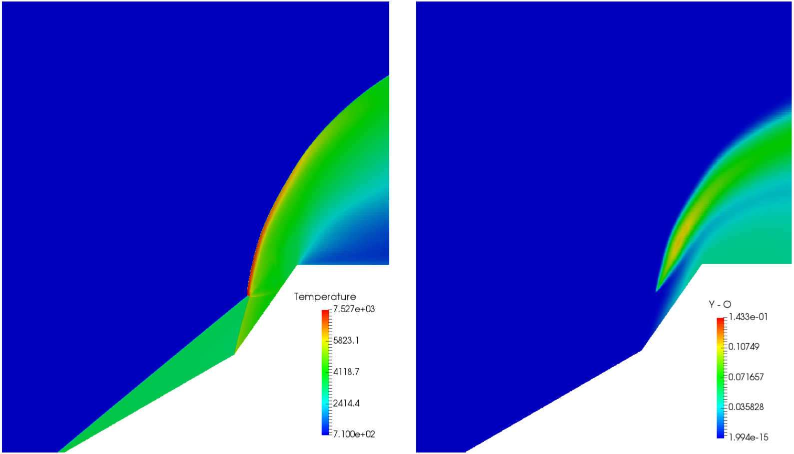
The temperature and mass fraction of atomic oxygen.





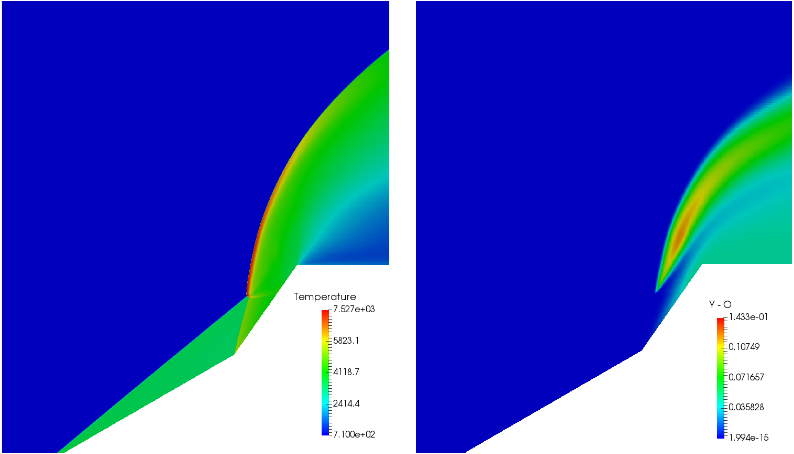
Double Wedge

The temperature and mass fraction of atomic oxygen.



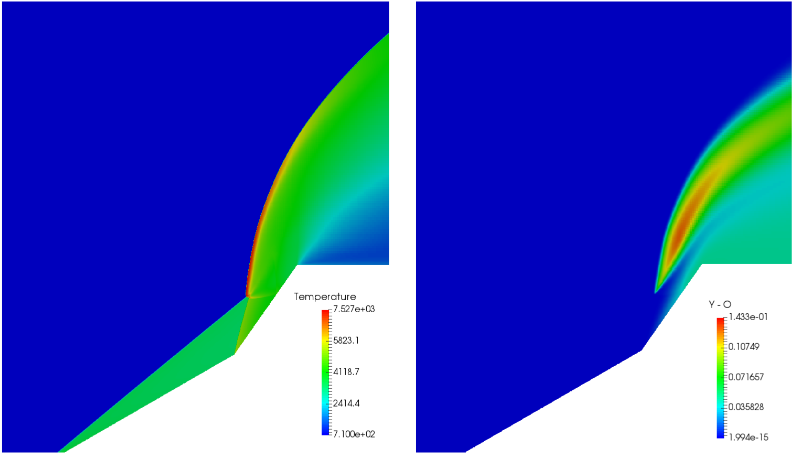
Double Wedge

The temperature and mass fraction of atomic oxygen.



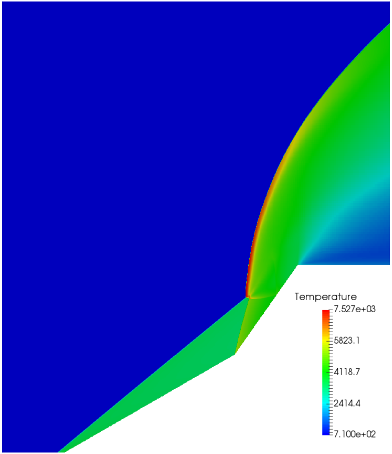
Double Wedge

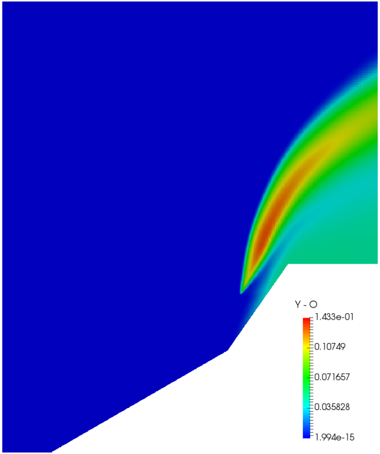
The temperature and mass fraction of atomic oxygen.



Double Wedge

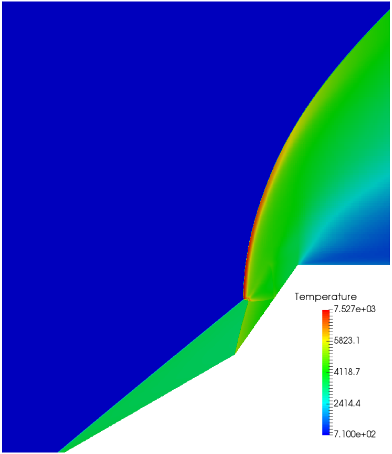
The temperature and mass fraction of atomic oxygen.

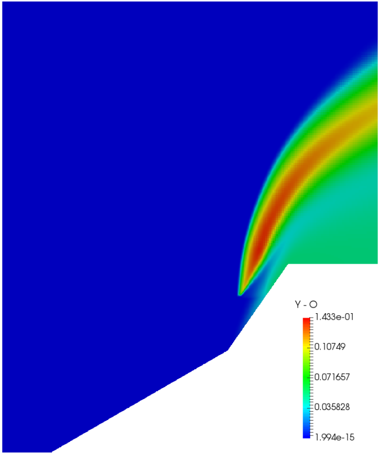




Double Wedge

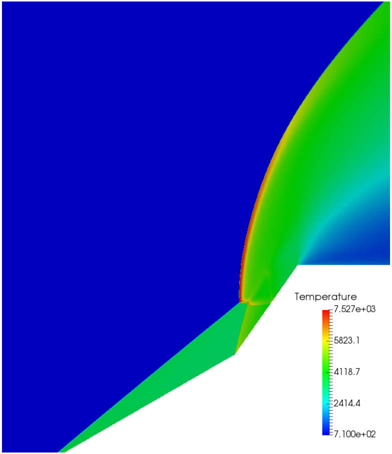
The temperature and mass fraction of atomic oxygen.

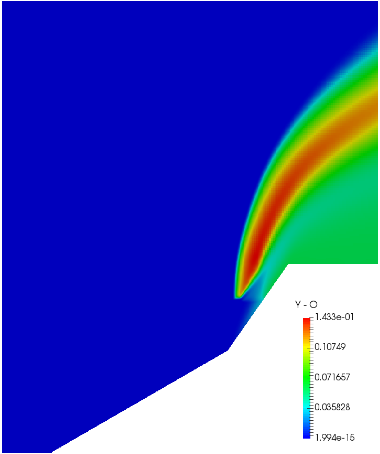




Double Wedge

The temperature and mass fraction of atomic oxygen.

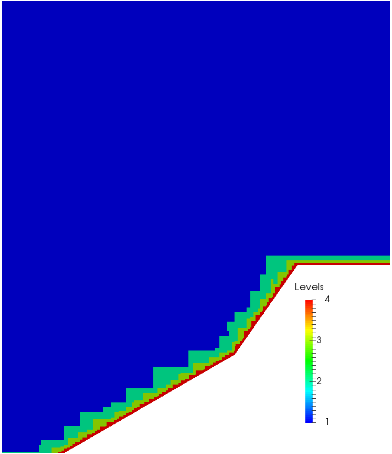


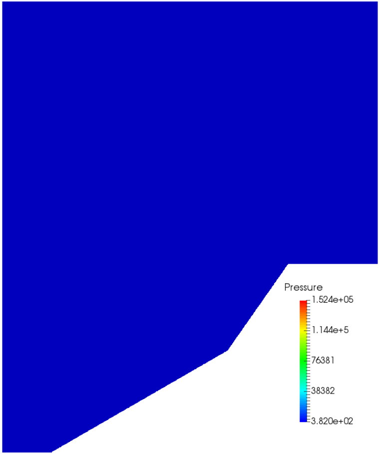


$t = 242\mu$secs.

Double Wedge

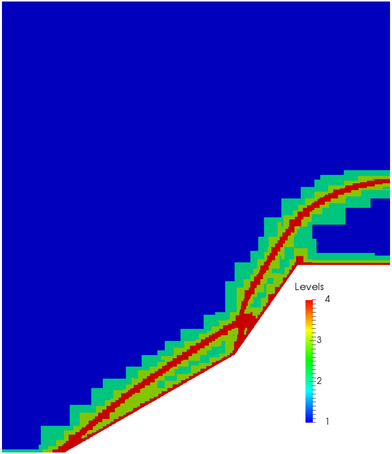
The mesh was refined using pressure and density gradients.

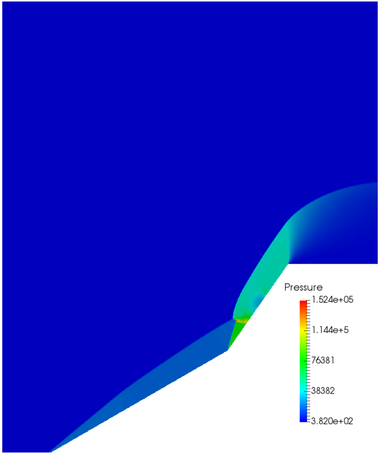




Double Wedge

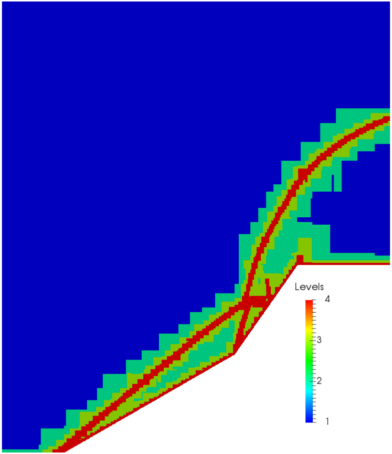
The mesh was refined using pressure and density gradients.

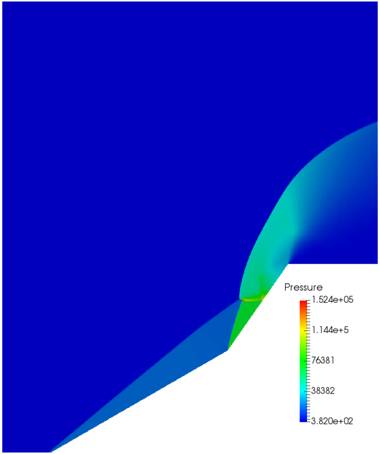




Double Wedge

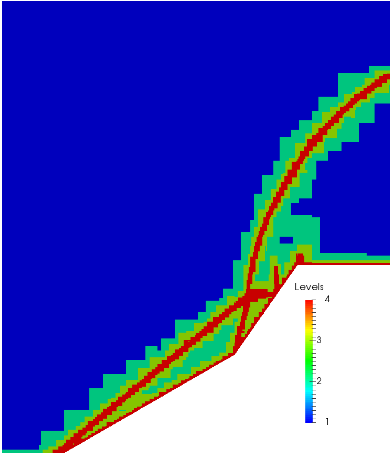
The mesh was refined using pressure and density gradients.

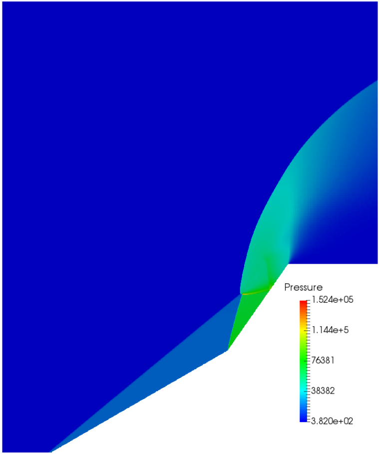




Double Wedge

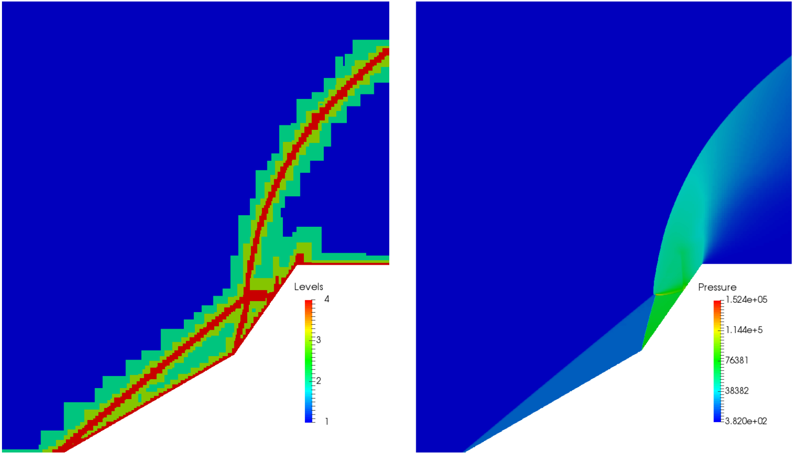
The mesh was refined using pressure and density gradients.





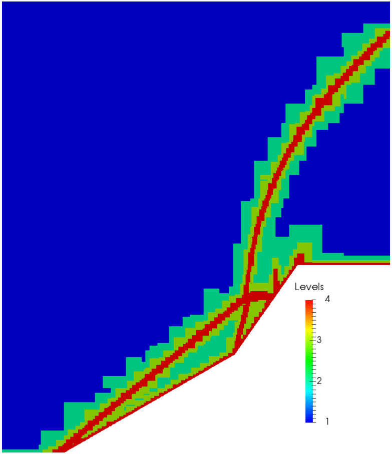
Double Wedge

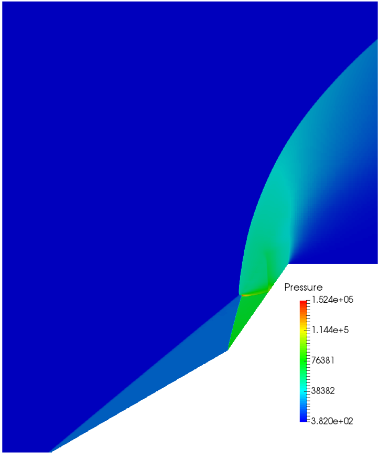
The mesh was refined using pressure and density gradients.



Double Wedge

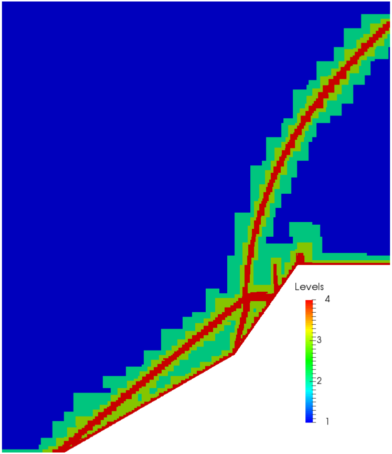
The mesh was refined using pressure and density gradients.

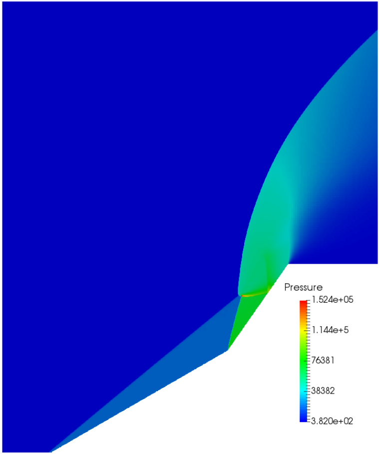




Double Wedge

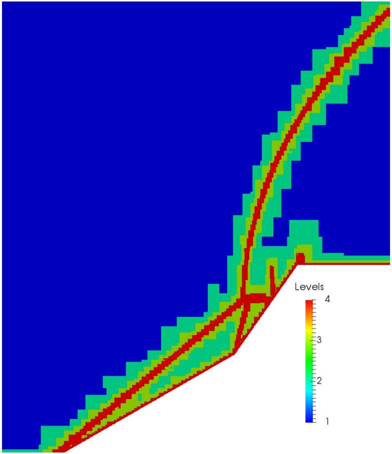
The mesh was refined using pressure and density gradients.

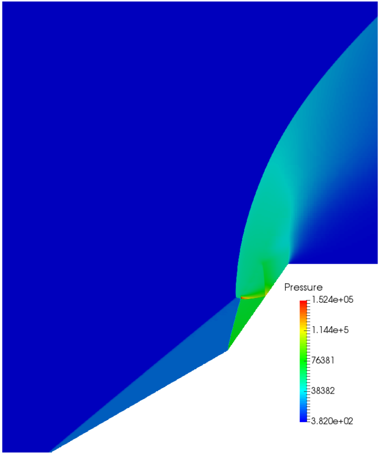




Double Wedge

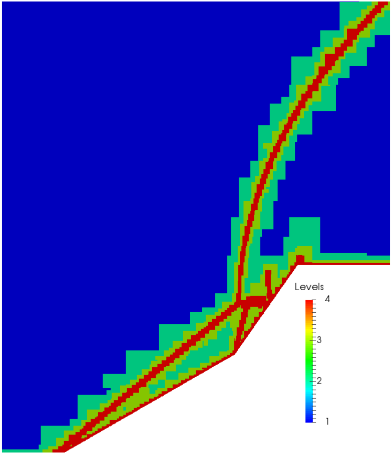
The mesh was refined using pressure and density gradients.

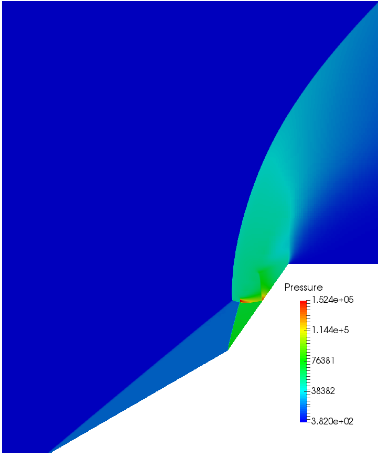




Double Wedge

The mesh was refined using pressure and density gradients.

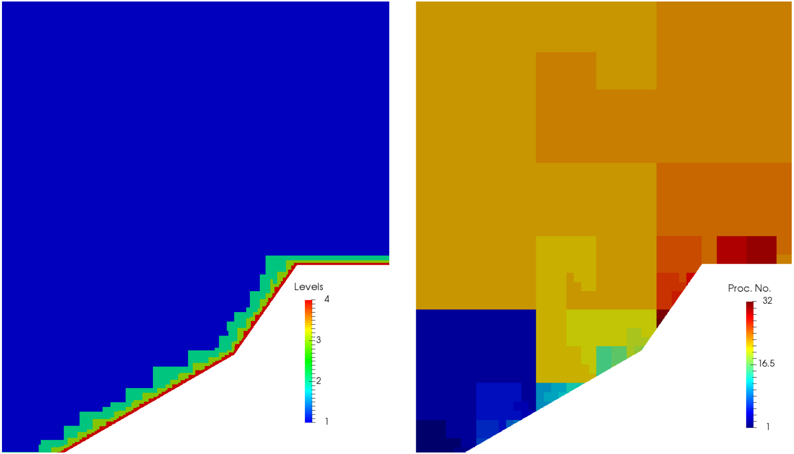




$t = 242\,\mu\mathrm{secs}.$

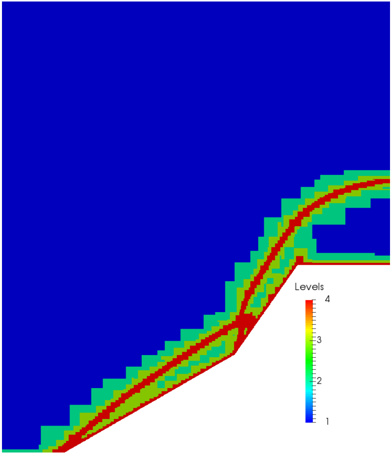
Double Wedge

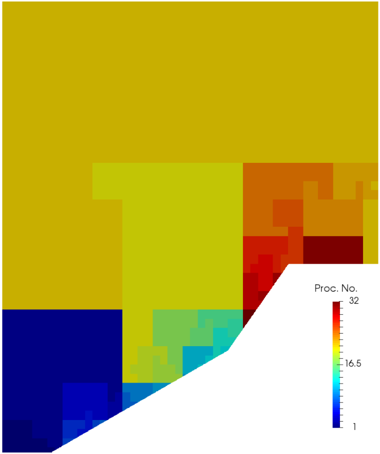
Dynamic load balancing distributes the cells across the processors.



Double Wedge

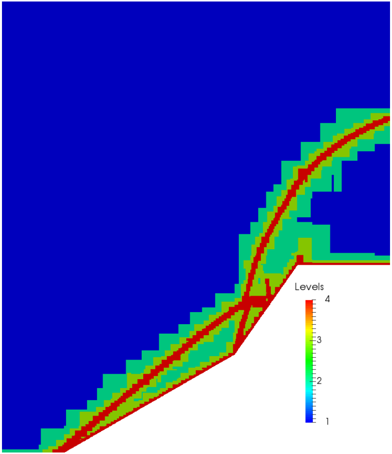
Dynamic load balancing distributes the cells across the processors.

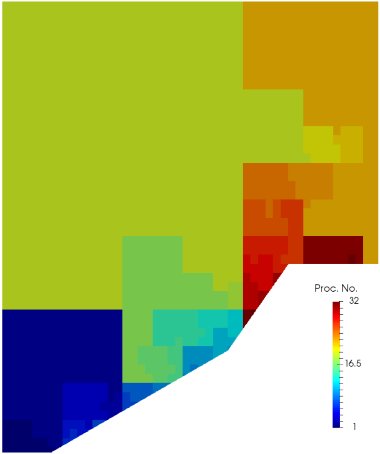




Double Wedge

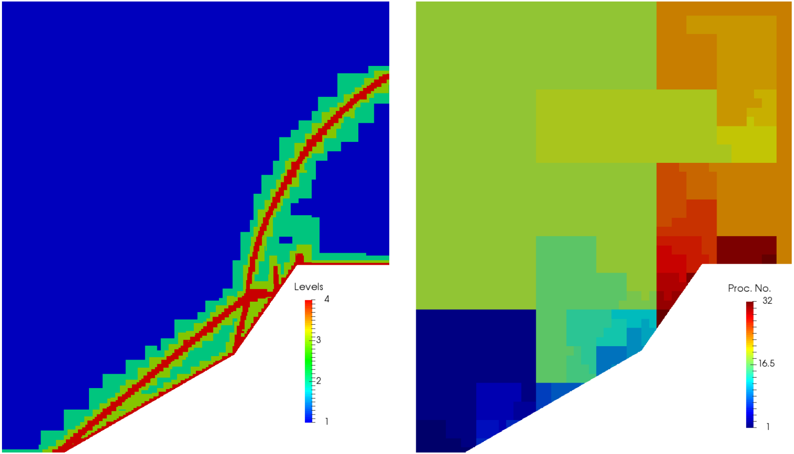
Dynamic load balancing distributes the cells across the processors.





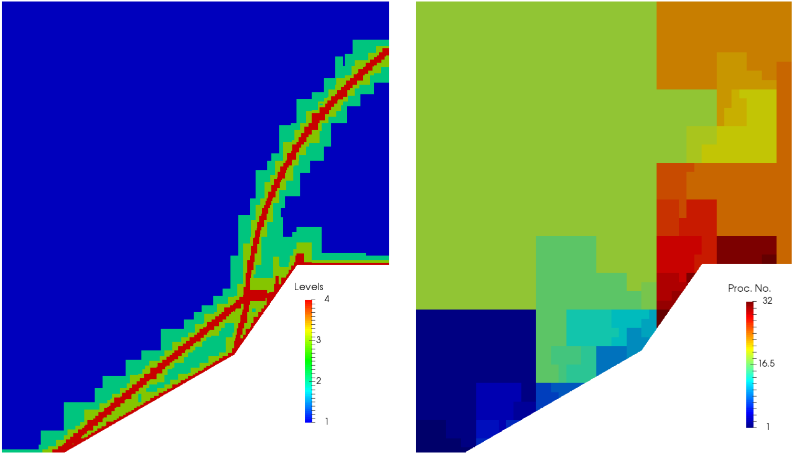
Double Wedge

Dynamic load balancing distributes the cells across the processors.



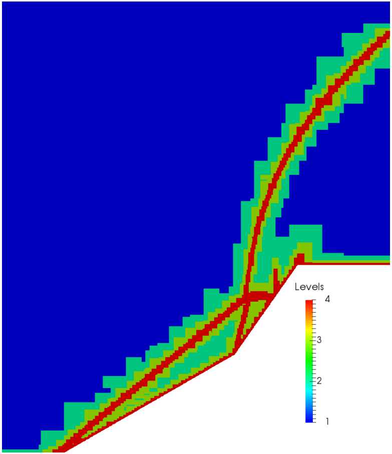
Double Wedge

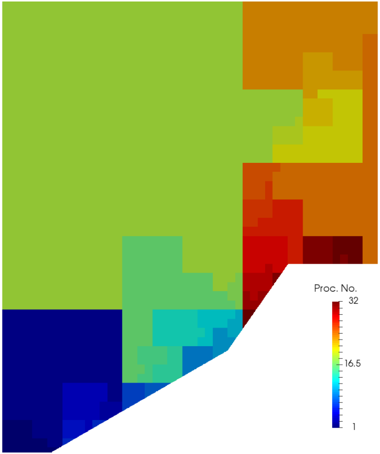
Dynamic load balancing distributes the cells across the processors.



Double Wedge

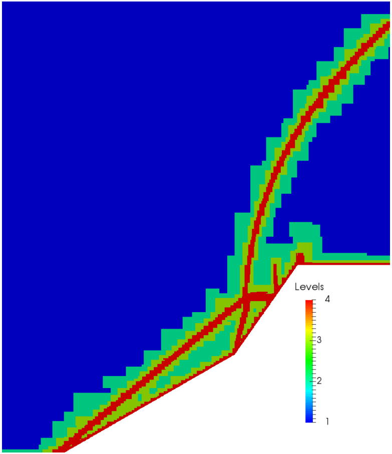
Dynamic load balancing distributes the cells across the processors.

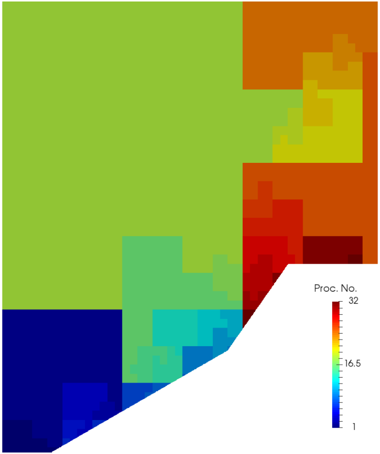




Double Wedge

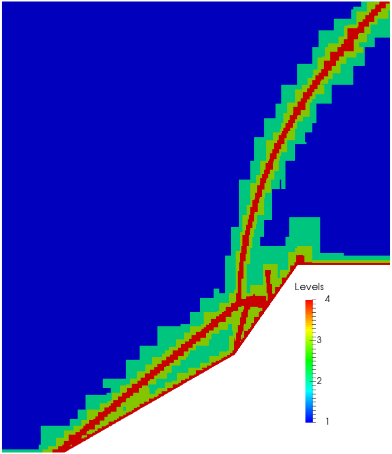
Dynamic load balancing distributes the cells across the processors.

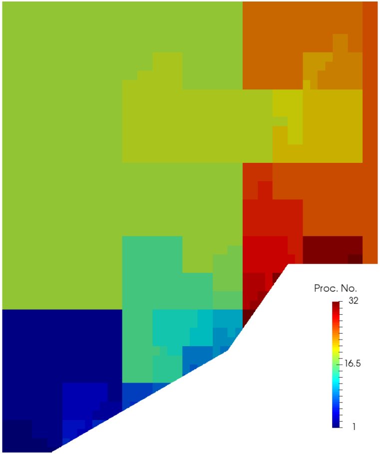




Double Wedge

Dynamic load balancing distributes the cells across the processors.

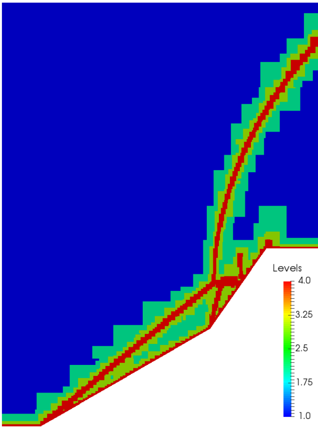


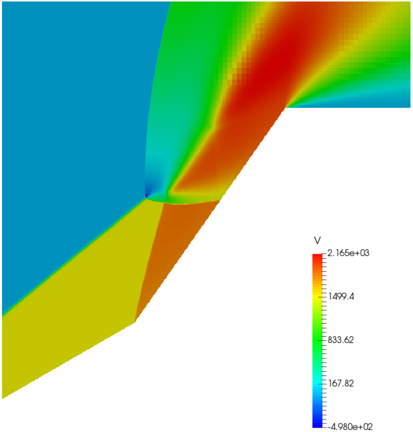


$t = 242\,\mu\mathrm{secs}.$

Double Wedge

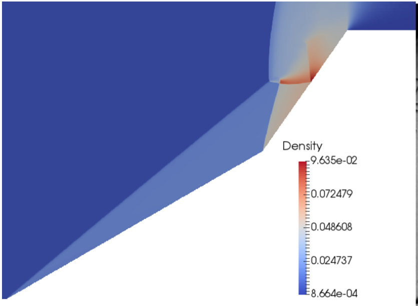
The AMR enables the flow features to be captured in detail.

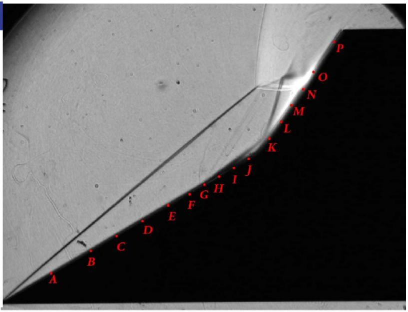




Double Wedge

The AMR enables the flow features to be captured in detail.





The schlieren image is taken from [Pezzella et al., 2015].

Hypersonic Sphere

Simulations of a half inch sphere travelling at hypersonic speeds in air [Lobb, 1964].

Mach number range between 8.4 and 16.1, with $p_\infty = 1333\,\mathrm{Pa}$ and $T_\infty = 293\,\mathrm{K}$.

The shock standoff distance was measured at each condition.

Hypersonic Sphere

Simulations of a half inch sphere travelling at hypersonic speeds in air [Lobb, 1964].

Mach number range between 8.4 and 16.1, with $p_\infty = 1333\,\mathrm{Pa}$ and $T_\infty = 293\,\mathrm{K}$.

The shock standoff distance was measured at each condition.

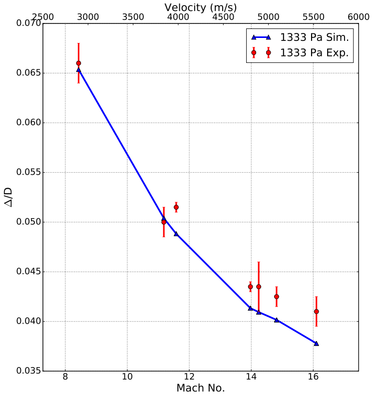
The shock standoff distance is used to validate the non-equilibrium model.

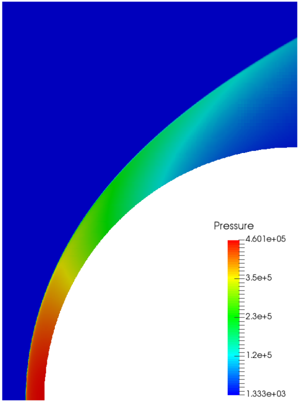
Validation of the axi-symmetric source term.

$$\mathbf{W}_{\mathrm{axi}} = -\frac{1}{y}\begin{bmatrix} \rho_1 v \\ \vdots \\ \rho_N v \\ \rho u v \\ \rho v^2 \\ (\rho E + p)v \end{bmatrix}$$

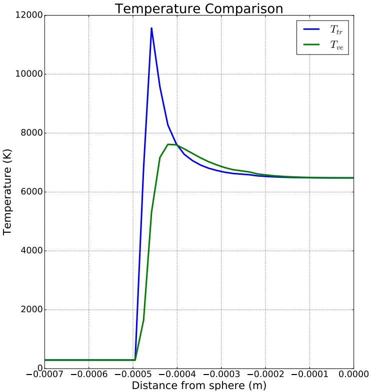
Hypersonic Sphere

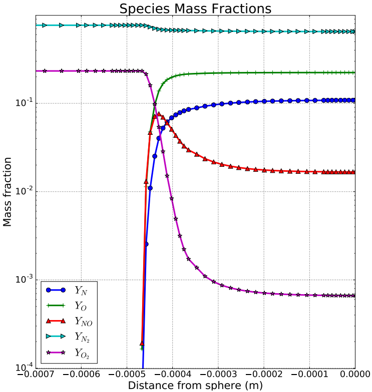
Computed shock standoff distances compared with experimental data.





Hypersonic Sphere





Mapped Solution Update

Within the AMROC-Clawpack framework, the solution is stored in physical (x, y) space and the fluxes are mapped from computational (ξ, η) space.

Mapped Solution Update

Within the AMROC-Clawpack framework, the solution is stored in physical (x, y) space and the fluxes are mapped from computational (ξ, η) space.
Using dimensional splitting the solution update is given by:

Mapped Solution Update

Within the AMROC-Clawpack framework, the solution is stored in physical (x, y) space and the fluxes are mapped from computational (ξ, η) space.
Using dimensional splitting the solution update is given by:

$$\mathbf{Q}^*_{i,j} = \mathbf{Q}^n_{i,j} - \frac{\Delta t}{\Delta \xi}\left[\left(\hat{\mathbf{F}} - \hat{\mathbf{F}}^v\right)_{i+1,j} - \left(\hat{\mathbf{F}} - \hat{\mathbf{F}}^v\right)_{i,j}\right]\frac{\Delta\eta\Delta\xi}{V_{i,j}},$$

Mapped Solution Update

Within the AMROC-Clawpack framework, the solution is stored in physical (x, y) space and the fluxes are mapped from computational (ξ, η) space.
Using dimensional splitting the solution update is given by:

$$\mathbf{Q}^{*}_{i,j} = \mathbf{Q}^{n}_{i,j} - \frac{\Delta t}{\Delta \xi}\left[\left(\hat{\mathbf{F}} - \hat{\mathbf{F}}^{v}\right)_{i+1,j} - \left(\hat{\mathbf{F}} - \hat{\mathbf{F}}^{v}\right)_{i,j}\right]\frac{\Delta\eta\Delta\xi}{V_{i,j}},$$

$$\mathbf{Q}^{n+1}_{i,j} = \mathbf{Q}^{*}_{i,j} - \frac{\Delta t}{\Delta \eta}\left[\left(\hat{\mathbf{G}} - \hat{\mathbf{G}}^{v}\right)_{i,j+1} - \left(\hat{\mathbf{G}} - \hat{\mathbf{G}}^{v}\right)_{i,j}\right]\frac{\Delta\eta\Delta\xi}{V_{i,j}}.$$

Mapped Solution Update

Within the AMROC-Clawpack framework, the solution is stored in physical (x, y) space and the fluxes are mapped from computational (ξ, η) space.
Using dimensional splitting the solution update is given by:

$$\mathbf{Q}^{*}_{i,j} = \mathbf{Q}^{n}_{i,j} - \frac{\Delta t}{\Delta \xi}\left[\left(\hat{\mathbf{F}} - \hat{\mathbf{F}}^{v}\right)_{i+1,j} - \left(\hat{\mathbf{F}} - \hat{\mathbf{F}}^{v}\right)_{i,j}\right]\frac{\Delta\eta\Delta\xi}{V_{i,j}}\,,$$

$$\mathbf{Q}^{n+1}_{i,j} = \mathbf{Q}^{*}_{i,j} - \frac{\Delta t}{\Delta \eta}\left[\left(\hat{\mathbf{G}} - \hat{\mathbf{G}}^{v}\right)_{i,j+1} - \left(\hat{\mathbf{G}} - \hat{\mathbf{G}}^{v}\right)_{i,j}\right]\frac{\Delta\eta\Delta\xi}{V_{i,j}}\,.$$

where $V_{i,j}$ is the volume of cell i, j in physical space. $\hat{\mathbf{F}}$, $\hat{\mathbf{F}}^{v}$, $\hat{\mathbf{G}}$, $\hat{\mathbf{G}}^{v}$ are the physical fluxes **per computational unit length**.

Mapped Mesh Computation

In the mapped mesh computations, the flux is transformed to align with the cell face,

$$\hat{\mathbf{F}} = T^{-1}\mathbf{F}_n(T\,\mathbf{Q}_l, T\,\mathbf{Q}_r)\,,$$

Mapped Mesh Computation

In the mapped mesh computations, the flux is transformed to align with the cell face,

$$\hat{\mathbf{F}} = T^{-1}\mathbf{F}_n(T\,\mathbf{Q}_l, T\,\mathbf{Q}_r)\,,$$

where T is the transformation matrix,

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \ddots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \hat{n}^x & \hat{n}^y & 0 & 0 \\ 0 & 0 & 0 & -\hat{n}^y & \hat{n}^x & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Mapped Inviscid Fluxes

The inviscid fluxes per computational unit length are found by:

Mapped Inviscid Fluxes

The inviscid fluxes per computational unit length are found by:

- Rotating the momentum components to be normal to the face,
- Calculating the flux with the rotated solution vectors,
- Rotating the solution vector back,
- Scaling the flux using the ratio of the computational face to the mapped face

Mapped Inviscid Fluxes

The inviscid fluxes per computational unit length are found by:

- Rotating the momentum components to be normal to the face,
- Calculating the flux with the rotated solution vectors,
- Rotating the solution vector back,
- Scaling the flux using the ratio of the computational face to the mapped face

In the ξ directional sweep, this gives

$$\mathbf{F}_{i-1/2,j} = T^{-1}_{i-1/2,j}\mathbf{F}_n\left(T_{i-1/2,j}\mathbf{Q}_{i-1,j},\, T_{i-1/2,j}\mathbf{Q}_{i,j}\right).$$

where T is the rotation matrix used to rotate the momentum components, and $\mathbf{F}_n$ is the normal flux through the face.

Mapped Inviscid Fluxes

The inviscid fluxes per computational unit length are found by:

- Rotating the momentum components to be normal to the face,
- Calculating the flux with the rotated solution vectors,
- Rotating the solution vector back,
- Scaling the flux using the ratio of the computational face to the mapped face

In the ξ directional sweep, this gives

$$\mathbf{F}_{i-1/2,j} = T^{-1}_{i-1/2,j}\mathbf{F}_n\big(T_{i-1/2,j}\mathbf{Q}_{i-1,j},\, T_{i-1/2,j}\mathbf{Q}_{i,j}\big)\,.$$

where T is the rotation matrix used to rotate the momentum components, and $\mathbf{F}_n$ is the normal flux through the face.
The scaling is given by:

$$\hat{\mathbf{F}}_{i,j} = \frac{\left|\mathbf{n}_{i-1/2,j}\right|}{\Delta\eta}\,\mathbf{F}_{i-1/2,j}\,,$$

Mapped Mesh Computation

Experiments of a cylinder in hypersonic flow [Hornung, 1972] were simulated with the mapping and initial conditions given by,

$$x = \xi \cos(\eta), \quad y = -\xi \sin(\eta).$$

Radius	Y_{N_2}	Y_N	T_∞	p_∞	U_∞	M_∞
0.0127 m	0.927	0.073	1833 K	2.91 kPa	5590 m/s	6.14

Table: Cylinder geometry and freestream conditions

Mapped Mesh Computation

Experiments of a cylinder in hypersonic flow [Hornung, 1972] were simulated with the mapping and initial conditions given by,

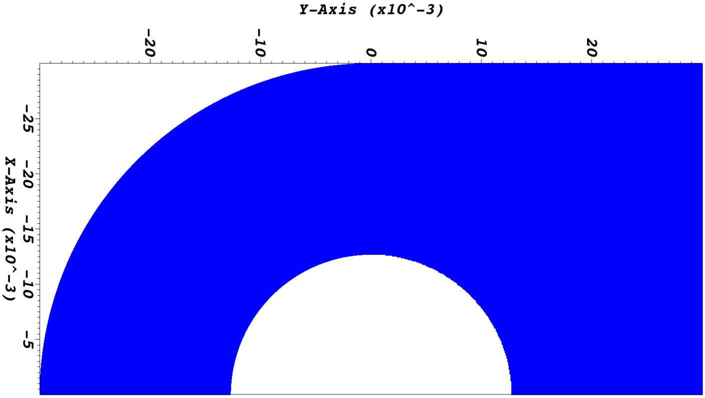
$$x = \xi \cos(\eta), \quad y = -\xi \sin(\eta).$$

Radius	Y_{N_2}	Y_N	T_∞	p_∞	U_∞	M_∞
$0.0127\,\mathrm{m}$	0.927	0.073	$1833\,\mathrm{K}$	$2.91\,\mathrm{kPa}$	$5590\,\mathrm{m/s}$	6.14

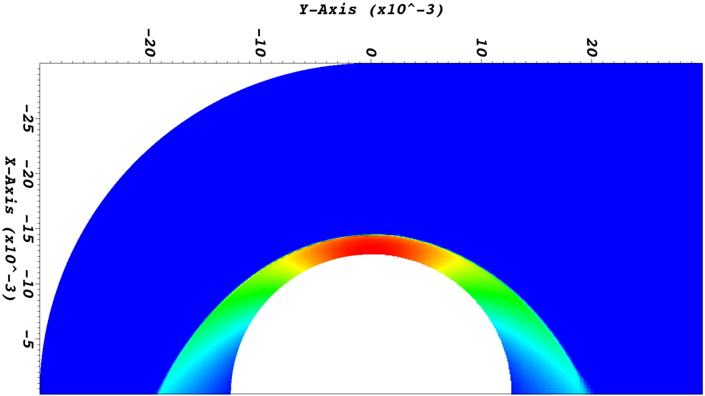
Table: Cylinder geometry and freestream conditions

The implementation was verified by comparing a mapped computation with a embedded boundary computation.

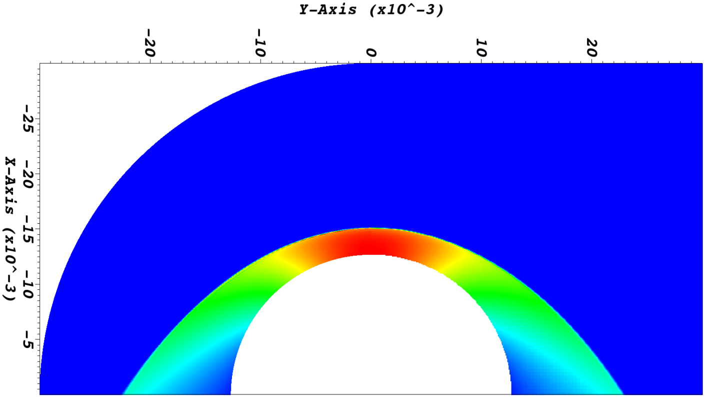
Mapped Mesh Computation



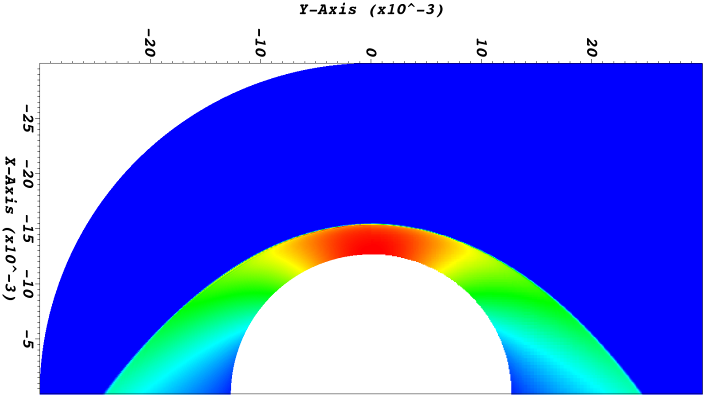
Mapped Mesh Computation



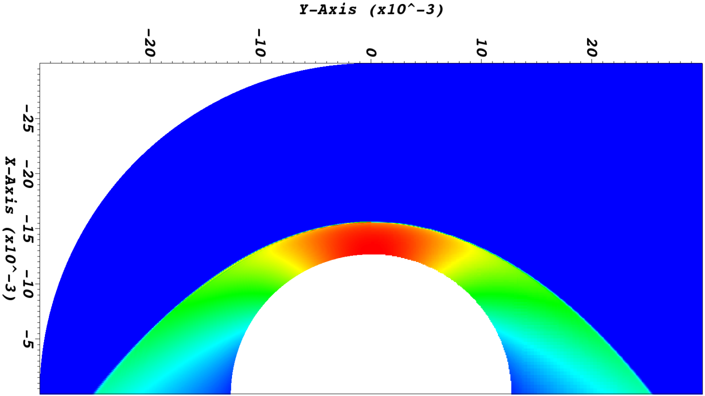
Mapped Mesh Computation



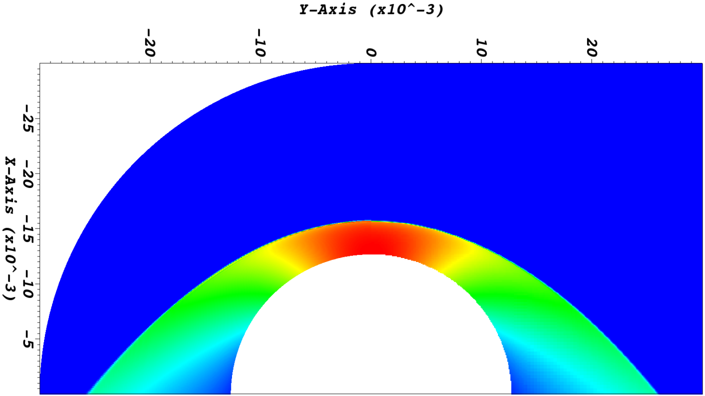
Mapped Mesh Computation



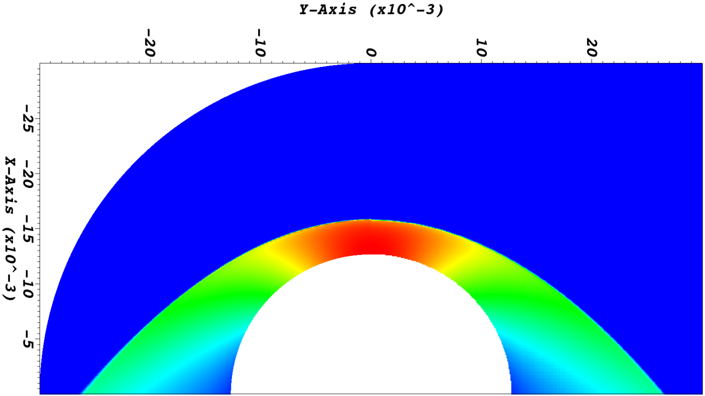
Mapped Mesh Computation



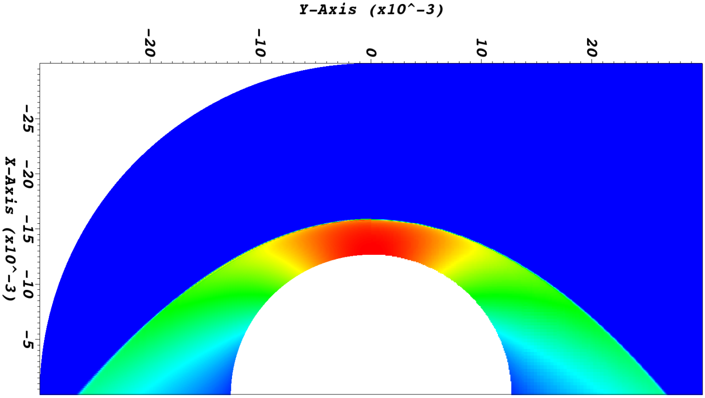
Mapped Mesh Computation



Mapped Mesh Computation

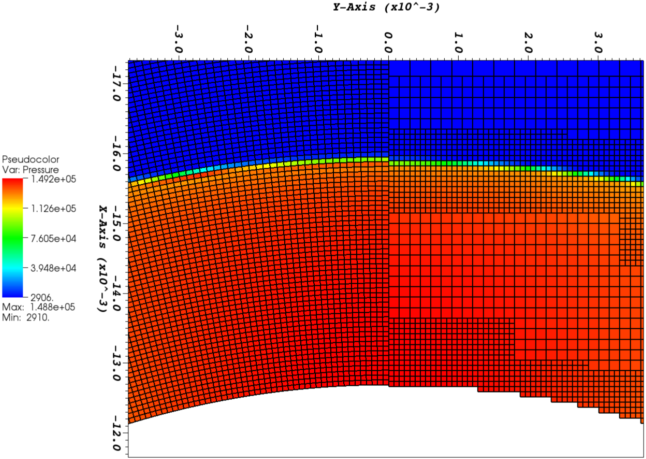


Mapped Mesh Computation



$t = 100\,\mu\text{sec}$

Mapped Mesh Computation



Viscous Computations

Preliminary results have been obtained for computations including the viscous flux vectors,

$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \left(\mathbf{F} - \mathbf{F}^v\right)}{\partial x} + \frac{\partial \left(\mathbf{G} - \mathbf{G}^v\right)}{\partial y} = \mathbf{W}$$

Viscous Computations

Preliminary results have been obtained for computations including the viscous flux vectors,

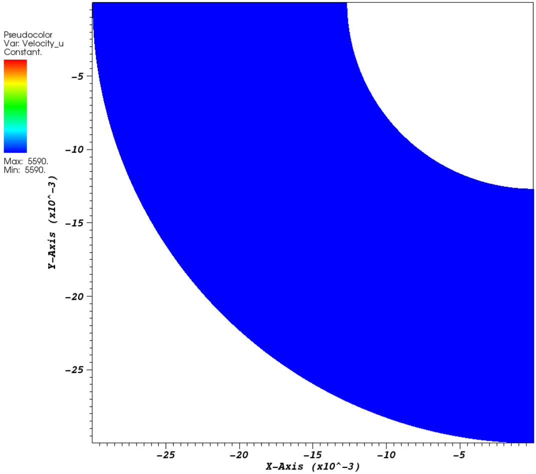
$$\frac{\partial \mathbf{Q}}{\partial t} + \frac{\partial \left(\mathbf{F} - \mathbf{F}^v\right)}{\partial x} + \frac{\partial \left(\mathbf{G} - \mathbf{G}^v\right)}{\partial y} = \mathbf{W}$$

where,

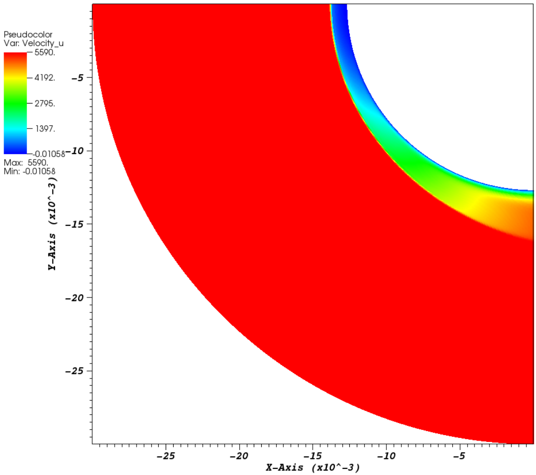
$$\mathbf{F}^v = \begin{bmatrix} -J_{x,1} \\ \vdots \\ -J_{x,N_s} \\ \tau_{x,x} \\ \tau_{y,x} \\ \kappa_{ve}\frac{\partial T_{ve}}{\partial x} - \sum_{s=1}^{N_s} J_{x,s} e_{ve} \\ \kappa_{tr}\frac{\partial T_{tr}}{\partial x} + \kappa_{ve}\frac{\partial T_{ve}}{\partial x} + u\tau_{x,x} + v\tau_{y,x} - \sum_{s=1}^{N_s} J_{x,s} h_s \end{bmatrix}.$$

and a similar expression is obtained for $\mathbf{G}^v$.

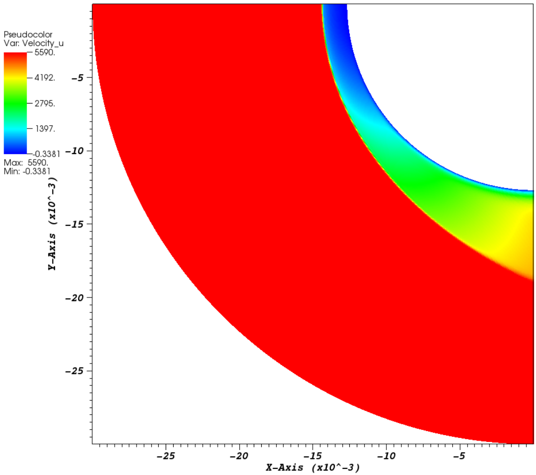
Viscous Computations



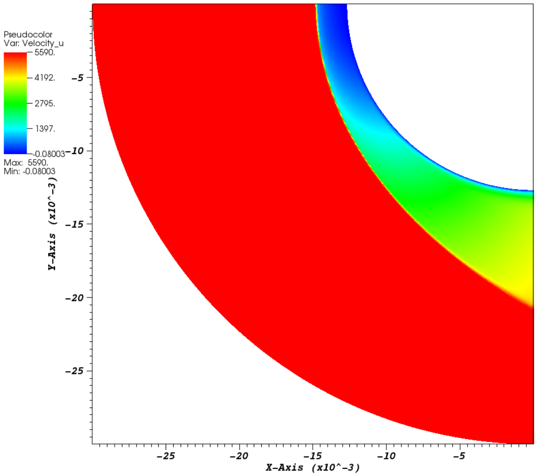
Viscous Computations



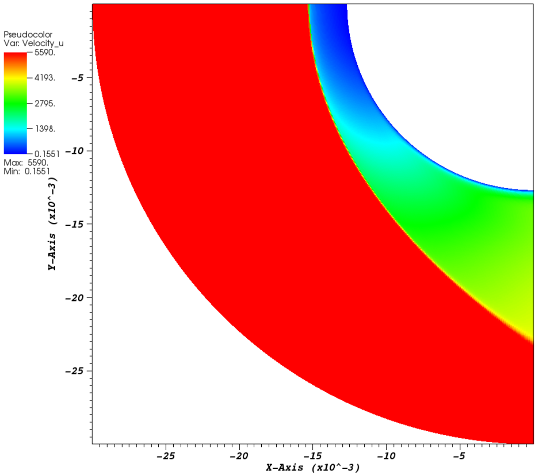
Viscous Computations



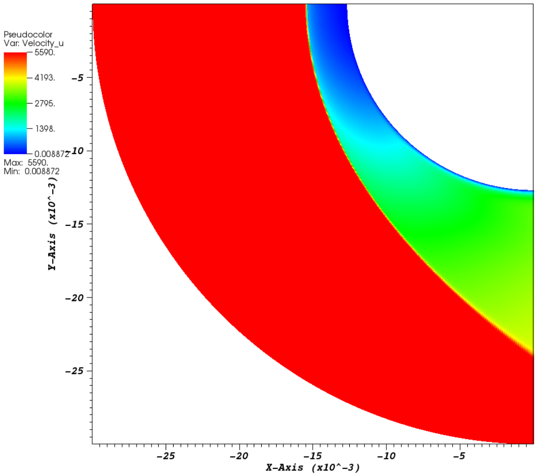
Viscous Computations



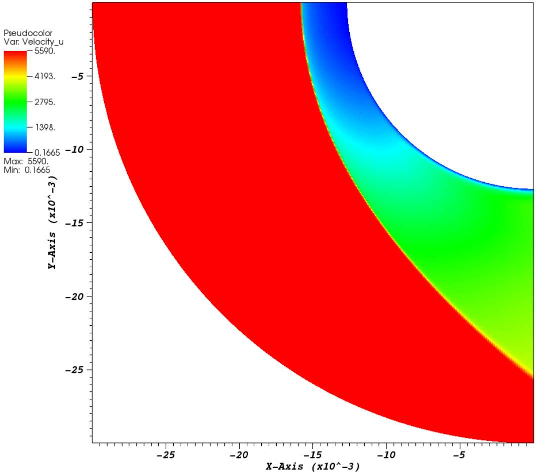
Viscous Computations



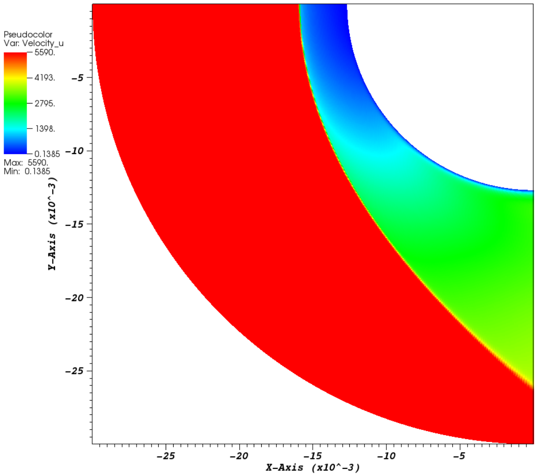
Viscous Computations



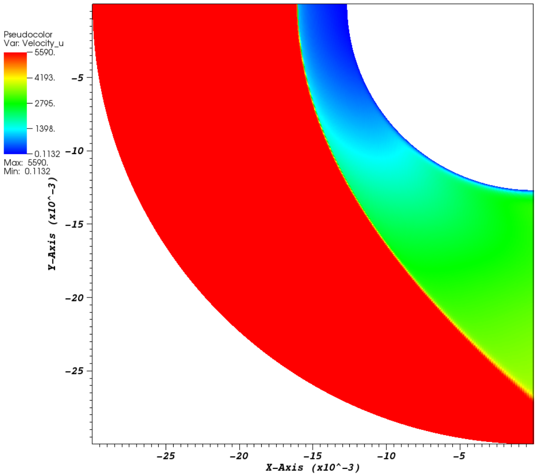
Viscous Computations



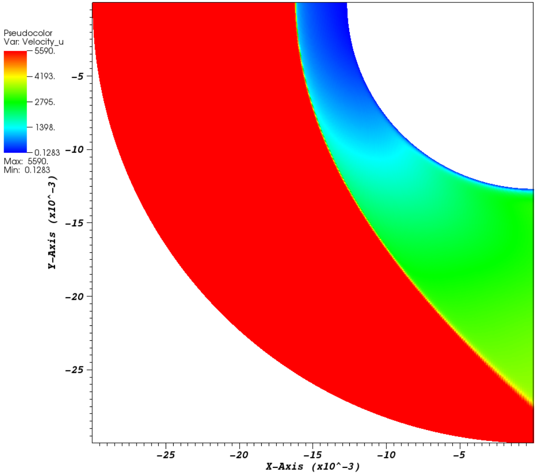
Viscous Computations



Viscous Computations



Viscous Computations



$t = 60\,\mu\text{secs}$.

Cylinder Heat Flux Computation

The mapped mesh solver has been validated by simulating a cylinder in a nonequilibrium, high enthalpy flow.

Cylinder Heat Flux Computation

The mapped mesh solver has been validated by simulating a cylinder in a nonequilibrium, high enthalpy flow.
The inflow conditions and results were taken from [Degrez et al., 2009].

T_∞	ρ_∞	U_∞	Y_{N_2}	Y_N	Y_{O_2}	Y_O	Y_{NO}
694 K	3.26 $\mathrm{g/m^3}$	4776 $\mathrm{m/s}$	0.7356	0.0	0.1340	0.07955	0.0509

Table: Freestream conditions for the HEG cylinder simulation.

Cylinder Heat Flux Computation

The mapped mesh solver has been validated by simulating a cylinder in a nonequilibrium, high enthalpy flow.
The inflow conditions and results were taken from [Degrez et al., 2009].

T_∞	ρ_∞	U_∞	Y_{N_2}	Y_N	Y_{O_2}	Y_O	Y_{NO}
694 K	3.26 $\mathrm{g/m^3}$	4776 $\mathrm{m/s}$	0.7356	0.0	0.1340	0.07955	0.0509

Table: Freestream conditions for the HEG cylinder simulation.

Cylinder Heat Flux Computation

The mapped mesh solver has been validated by simulating a cylinder in a nonequilibrium, high enthalpy flow.
The inflow conditions and results were taken from [Degrez et al., 2009].

T_∞	ρ_∞	U_∞	Y_{N_2}	Y_N	Y_{O_2}	Y_O	Y_{NO}
694 K	3.26 $\mathrm{g/m^3}$	4776 $\mathrm{m/s}$	0.7356	0.0	0.1340	0.07955	0.0509

Table: Freestream conditions for the HEG cylinder simulation.

A cylinder mesh was generated with hyperbolic tangent stretching away from the wall using a 1e-6 initial spacing.

Cylinder Heat Flux Comparison

The simulated results show good agreement with the experimental results:

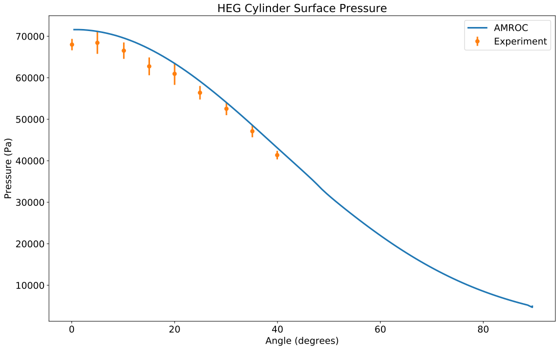


Figure: A comparison of the experimental and simulated surface pressures in the HEG cylinder experiment.

Cylinder Heat Flux Comparison

The simulated results show good agreement with the experimental results:

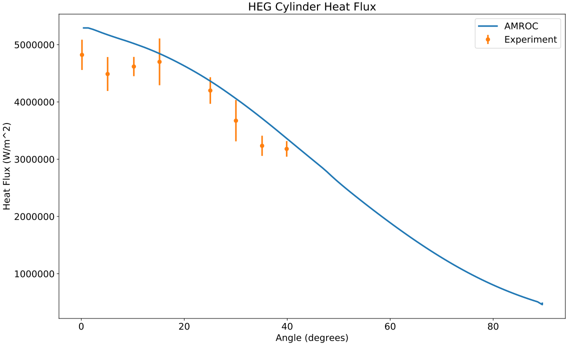


Figure: A comparison of the experimental and simulated surface heat fluxes in the HEG cylinder experiment.

Hybrid method

Convective numerical flux is defined as

$$\mathbf{F}^n_{inv} = \begin{cases} \mathbf{F}^n_{inv-WENO}, & \text{in } \mathcal{C} \\ \mathbf{F}^n_{inv-CD}, & \text{in } \overline{\mathcal{C}}, \end{cases}$$

Hybrid method

Convective numerical flux is defined as

$$\mathbf{F}^n_{inv} = \begin{cases} \mathbf{F}^n_{inv-WENO}, & \text{in } \mathcal{C} \\ \mathbf{F}^n_{inv-CD}, & \text{in } \overline{\mathcal{C}}, \end{cases}$$

- For LES: 3rd order WENO method, 2nd order TCD [Hill and Pullin, 2004]

Hybrid method

Convective numerical flux is defined as

$$\mathbf{F}^n_{inv} = \begin{cases} \mathbf{F}^n_{inv-WENO}, & \text{in } \mathcal{C} \\ \mathbf{F}^n_{inv-CD}, & \text{in } \overline{\mathcal{C}}, \end{cases}$$

- For LES: 3rd order WENO method, 2nd order TCD [Hill and Pullin, 2004]
- For DNS: Symmetric 6th order WENO, 6th-order CD scheme

J. Ziegler, RD, J. Shepherd, D. Pullin, *J. Comput. Phys.* 230(20):7598-7630, 2011.

Hybrid method

Convective numerical flux is defined as

$$\mathbf{F}^n_{inv} = \begin{cases} \mathbf{F}^n_{inv-WENO}, & \text{in } \mathcal{C} \\ \mathbf{F}^n_{inv-CD}, & \text{in } \overline{\mathcal{C}}, \end{cases}$$

- For LES: 3rd order WENO method, 2nd order TCD [Hill and Pullin, 2004]
- For DNS: Symmetric 6th order WENO, 6th-order CD scheme

  J. Ziegler, RD, J. Shepherd, D. Pullin, *J. Comput. Phys.* 230(20):7598-7630, 2011.

Use WENO scheme to only capture shock waves but resolve interface between species.

Hybrid method

Convective numerical flux is defined as

$$\mathbf{F}^n_{inv} = \begin{cases} \mathbf{F}^n_{inv-WENO}, & \text{in } \mathcal{C} \\ \mathbf{F}^n_{inv-CD}, & \text{in } \overline{\mathcal{C}}, \end{cases}$$

- For LES: 3rd order WENO method, 2nd order TCD [Hill and Pullin, 2004]
- For DNS: Symmetric 6th order WENO, 6th-order CD scheme

J. Ziegler, RD, J. Shepherd, D. Pullin, *J. Comput. Phys.* 230(20):7598-7630, 2011.

Use WENO scheme to only capture shock waves but resolve interface between species.

Shock detection based on using two criteria together:

1. Lax-Liu entropy condition $|u_R \pm a_R| < |u_* \pm a_*| < |u_L \pm a_L|$ tested with a threshold to eliminate weak acoustic waves. Used intermediate states at cell interfaces:

$$u_* = \frac{\sqrt{\rho_L u_L} + \sqrt{\rho_R u_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \quad a_* = \sqrt{(\gamma_* - 1)(h_* - \frac{1}{2}u_*^2)}, \ \ldots$$

Hybrid method

Convective numerical flux is defined as

$$\mathbf{F}^n_{inv} = \begin{cases} \mathbf{F}^n_{inv-WENO}, & \text{in } \mathcal{C} \\ \mathbf{F}^n_{inv-CD}, & \text{in } \overline{\mathcal{C}}, \end{cases}$$

- For LES: 3rd order WENO method, 2nd order TCD [Hill and Pullin, 2004]
- For DNS: Symmetric 6th order WENO, 6th-order CD scheme

J. Ziegler, RD, J. Shepherd, D. Pullin, *J. Comput. Phys.* 230(20):7598-7630, 2011.

Use WENO scheme to only capture shock waves but resolve interface between species.

Shock detection based on using two criteria together:

1. Lax-Liu entropy condition $|u_R \pm a_R| < |u_* \pm a_*| < |u_L \pm a_L|$ tested with a threshold to eliminate weak acoustic waves. Used intermediate states at cell interfaces:

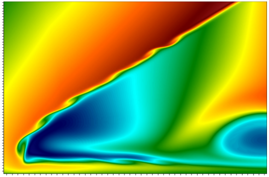
$$u_* = \frac{\sqrt{\rho_L u_L} + \sqrt{\rho_R u_R}}{\sqrt{\rho_L} + \sqrt{\rho_R}}, \quad a_* = \sqrt{(\gamma_* - 1)(h_* - \frac{1}{2}u_*^2)}, \ \dots$$

2. Limiter-inspired discontinuity test based on mapped normalized pressure gradient θ_j

$$\phi(\theta_j) = \frac{2\theta_j}{(1+\theta_j)^2} \quad \text{with} \quad \theta_j = \frac{|p_{j+1} - p_j|}{|p_{j+1} + p_j|}, \quad \phi(\theta_j) > \alpha_{Map}$$

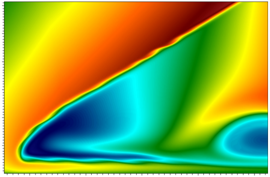
Results for shear layer in Mach reflection pattern

WENO/CD - 6 levels



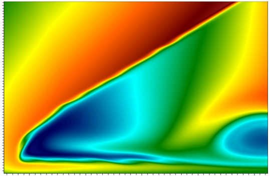
$\Delta x_{\min} = 3.91 \cdot 10^{-6}\,\mathrm{m}$

WENO/CD - 7 levels



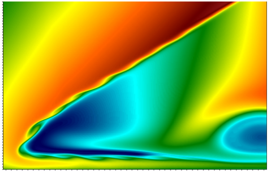
$\Delta x_{\min} = 1.95 \cdot 10^{-6}\,\mathrm{m}$

WENO/CD - 8 levels



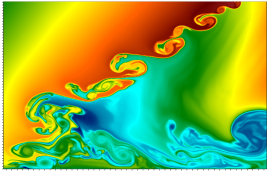
$\Delta x_{\min} = 9.77 \cdot 10^{-7}\,\mathrm{m}$

MUSCL - 7 levels



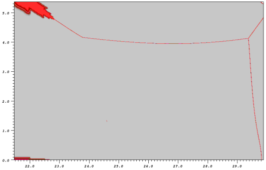
$\Delta x_{\min} = 1.05 \cdot 10^{-6}\,\mathrm{m}$

MUSCL - 7 levels - Euler



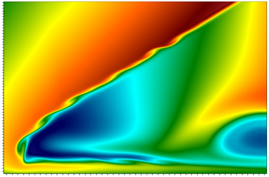
$\Delta x_{\min} = 1.05 \cdot 10^{-6}\,\mathrm{m}$

Usage of WENO for WENO/CD - 8 levels



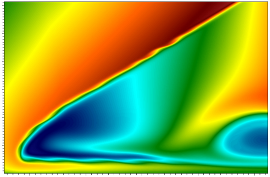
Results for shear layer in Mach reflection pattern

WENO/CD - 6 levels



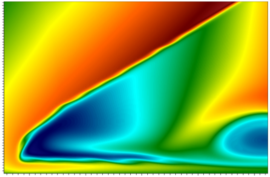
$\Delta x_{\min} = 3.91 \cdot 10^{-6}\,\mathrm{m}$

WENO/CD - 7 levels



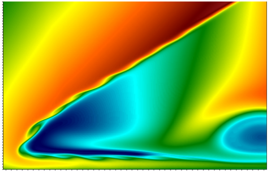
$\Delta x_{\min} = 1.95 \cdot 10^{-6}\,\mathrm{m}$

WENO/CD - 8 levels



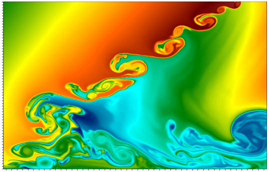
$\Delta x_{\min} = 9.77 \cdot 10^{-7}\,\mathrm{m}$

MUSCL - 7 levels



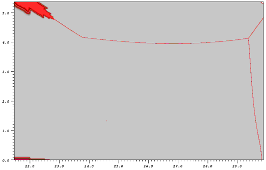
$\Delta x_{\min} = 1.05 \cdot 10^{-6}\,\mathrm{m}$

MUSCL - 7 levels - Euler



$\Delta x_{\min} = 1.05 \cdot 10^{-6}\,\mathrm{m}$

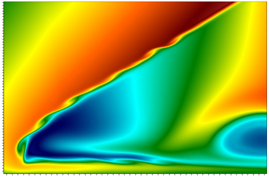
Usage of WENO for WENO/CD - 8 levels



- WENO/CD/RK3 gives results comparable to 4x finer resolved optimal 2nd-order scheme, but CPU times with SAMR 2-3x larger

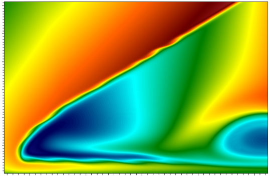
Results for shear layer in Mach reflection pattern

WENO/CD - 6 levels



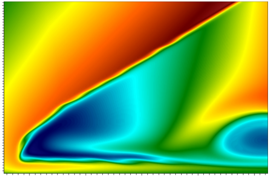
$\Delta x_{\rm min} = 3.91 \cdot 10^{-6}\,\mathrm{m}$

WENO/CD - 7 levels



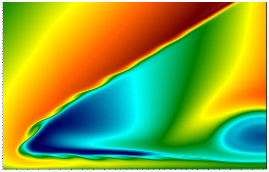
$\Delta x_{\rm min} = 1.95 \cdot 10^{-6}\,\mathrm{m}$

WENO/CD - 8 levels



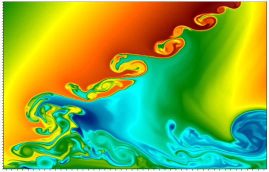
$\Delta x_{\rm min} = 9.77 \cdot 10^{-7}\,\mathrm{m}$

MUSCL - 7 levels



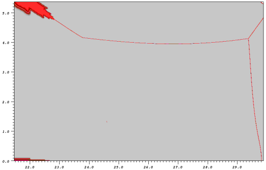
$\Delta x_{\rm min} = 1.05 \cdot 10^{-6}\,\mathrm{m}$

MUSCL - 7 levels - Euler



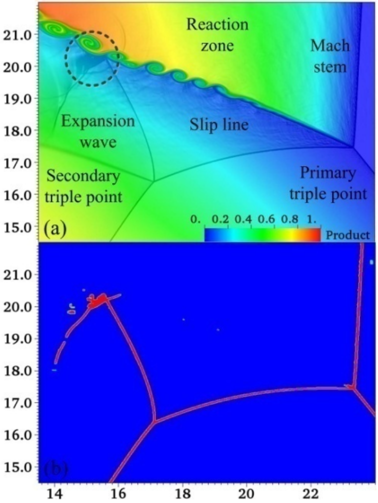
$\Delta x_{\rm min} = 1.05 \cdot 10^{-6}\,\mathrm{m}$

Usage of WENO for WENO/CD - 8 levels



- WENO/CD/RK3 gives results comparable to 4x finer resolved optimal 2nd-order scheme, but CPU times with SAMR 2-3x larger
- Gain in CPU time from higher-order scheme roughly one order

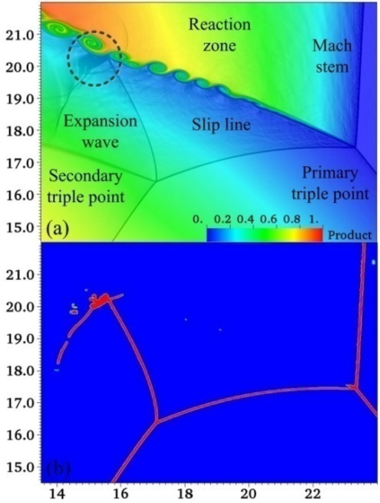
Detonation ignition by hot jet in 2d



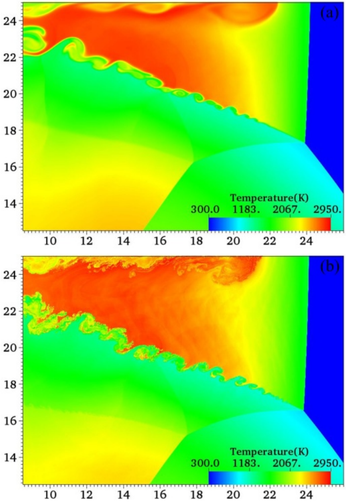
(a) Detailed structure, (b) WENO usage

X. Cai, RD, J. Liang, Y. Mahmoudi, *Proc. Combust. Institute* 36(2): 2725–2733, 2017

Detonation ignition by hot jet in 2d



(a) Detailed structure, (b) WENO usage



(a) Navier-Stokes, (b) Euler

X. Cai, RD, J. Liang, Y. Mahmoudi, *Proc. Combust. Institute* 36(2): 2725–2733, 2017

Conclusions

- A Cartesian embedded boundary method for compressible flows with block-based adaptive mesh refinement is an efficient and scalable prediction tool for pressure and shock waves created by moving bodies

Conclusions

- A Cartesian embedded boundary method for compressible flows with block-based adaptive mesh refinement is an efficient and scalable prediction tool for pressure and shock waves created by moving bodies
- Multi-resolution and fluid-structure coupling problems can be tackled without expensive mesh regeneration
  - Level set approach easily handles large motions, element failure and removal
  - Dynamic adaptation ensures high resolution at embedded boundaries and essential flow features

Conclusions

- A Cartesian embedded boundary method for compressible flows with block-based adaptive mesh refinement is an efficient and scalable prediction tool for pressure and shock waves created by moving bodies
- Multi-resolution and fluid-structure coupling problems can be tackled without expensive mesh regeneration
  - Level set approach easily handles large motions, element failure and removal
  - Dynamic adaptation ensures high resolution at embedded boundaries and essential flow features
- Aerodynamics of bodies with large motion are easily accessible
  - Current inviscid approach predicts maximal overpressure in front of trains reliably
  - For predicting the flow around entire trains, the boundary layer growing over the train body needs to be considered.
  - AMROC solvers for the compressible Navier-Stokes equations and even LES are already available, however, for this particular application a turbulent wall function on the embedded boundary first needs to be implemented. Such a wall function is currently work-in-progress for the LBM-LES solver.

Conclusions – Hypersonics

- A two-temperature model solver that is suitable for very high temperatures, i.e., high enthalpy re-entry flows, has been developed.
- The Cartesian version is fully integrated into SAMR AMROC-Clawpack; structured non-Cartesian version runs also within AMROC-Clawpack but only on non-adaptive meshes so far
- SAMR framework can remain basically unchanged; however mapping needs to be considered in prolongation and restriction, flux correction, visualization (work in progress)

Conclusions – Hypersonics

- A two-temperature model solver that is suitable for very high temperatures, i.e., high enthalpy re-entry flows, has been developed.
- The Cartesian version is fully integrated into SAMR AMROC-Clawpack; structured non-Cartesian version runs also within AMROC-Clawpack but only on non-adaptive meshes so far
- SAMR framework can remain basically unchanged; however mapping needs to be considered in prolongation and restriction, flux correction, visualization (work in progress)
- For moving geometries, the goal is a Chimera-type approach that constructs non-Cartesian boundary layer meshes near the body and uses SAMR in the far field
- Incorporation of the methodology into the hybrid WENO/CD scheme for high enthalpy DNS in 3D is proposed within the next two years

References I

[Barton et al., 2013] Barton, P. T., Deiterding, R., Meiron, D. I., and Pullin, D. I. (2013). Eulerian continuum model and adaptive finite-difference method for high-velocity impact and penetration problems. *J. Comput. Phys.*, 240:76–99.

[Bell et al., 1994] Bell, J., Berger, M., Saltzman, J., and Welcome, M. (1994). Three-dimensional adaptive mesh refinement for hyperbolic conservation laws. *SIAM J. Sci. Comp.*, 15(1):127–138.

[Berger, 1986] Berger, M. (1986). Data structures for adaptive grid generation. *SIAM J. Sci. Stat. Comput.*, 7(3):904–916.

[Berger and Colella, 1988] Berger, M. and Colella, P. (1988). Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:64–84.

[Berger and Rigoutsos, 1991] Berger, M. and Rigoutsos, I. (1991). An algorithm for point clustering and grid generation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1278–1286.

[Cai et al., 2018] Cai, X., Deiterding, R., Liang, J., Sun, M., and Mahmoudi, Y. (2018). Experimental and numerical investigations on propagating modes of detonations: detonation wave/boundary layer interaction. *Combust. Flame*, 190:201–215.

[Cai et al., 2016] Cai, X., Liang, J., Deiterding, R., Che, Y., and Lin, Z. (2016). Adaptive mesh refinement based simulations of three-dimensional detonation combustion in supersonic combustible mixtures with a detailed reaction model. *Int. J. Hydrogen Energy*, 41:3222–3239.

[Cerminara et al., 2018] Cerminara, A., Deiterding, R., and Sandham, N. (2018). Dns of hypersonic flow over porous surfaces with a hybrid method. In *AIAA Aerospace Sciences Meeting, AIAA Science and Technology Forum and Exposition*.

[Cirak et al., 2007] Cirak, F., Deiterding, R., and Mauch, S. P. (2007). Large-scale fluid-structure interaction simulation of viscoplastic and fracturing thin shells subjected to shocks and detonations. *Computers & Structures*, 85(11-14):1049–1065.

[Degrez et al., 2009] Degrez, G., Lani, A., Panesi, M., Chazot, O., and Deconinck, H. (2009). Modelling of High Enthalpy, High Mach Number Flows. *Journal of Physics D: Applied Physics*, 42.

[Deiterding, 2003] Deiterding, R. (2003). *Parallel adaptive simulation of multi-dimensional detonation structures*. PhD thesis, Brandenburgische Technische Universität Cottbus.

[Deiterding, 2005] Deiterding, R. (2005). Construction and application of an AMR algorithm for distributed memory computers. In Plewa, T., Linde, T., and Weirs, V. G., editors, *Adaptive Mesh Refinement - Theory and Applications*, volume 41 of *Lecture Notes in Computational Science and Engineering*, pages 361–372. Springer.

References II

[Deiterding, 2009] Deiterding, R. (2009). A parallel adaptive method for simulating shock-induced combustion with detailed chemical kinetics in complex domains. *Computers & Structures*, 87:769–783.

[Deiterding, 2011a] Deiterding, R. (2011a). Block-structured adaptive mesh refinement - theory, implementation and application. *European Series in Applied and Industrial Mathematics: Proceedings*, 34:97–150.

[Deiterding, 2011b] Deiterding, R. (2011b). High-resolution numerical simulation and analysis of Mach reflection structures in detonation waves in low-pressure $\mathrm{h}_2 : \mathrm{o}_2 : \mathrm{ar}$ mixtures: a summary of results obtained with adaptive mesh refinement framework AMROC. *J. Combustion*, 2011:738969.

[Deiterding and Bader, 2005] Deiterding, R. and Bader, G. (2005). High-resolution simulation of detonations with detailed chemistry. In Warnecke, G., editor, *Analysis and Numerics for Conservation Laws*, pages 69–91. Springer.

[Deiterding et al., 2009a] Deiterding, R., Cirak, F., and Mauch, S. P. (2009a). Efficient fluid-structure interaction simulation of viscoplastic and fracturing thin-shells subjected to underwater shock loading. In Hartmann, S., Meister, A., Schäfer, M., and Turek, S., editors, *Int. Workshop on Fluid-Structure Interaction. Theory, Numerics and Applications, Herrsching am Ammersee 2008*, pages 65–80. kassel university press GmbH.

[Deiterding et al., 2007] Deiterding, R., Cirak, F., Mauch, S. P., and Meiron, D. I. (2007). A virtual test facility for simulating detonation- and shock-induced deformation and fracture of thin flexible shells. *Int. J. Multiscale Computational Engineering*, 5(1):47–63.

[Deiterding et al., 2009b] Deiterding, R., Domingues, M. O., Gomes, S. M., Roussel, O., and Schneider, K. (2009b). Adaptive multiresolution or adaptive mesh refinement? A case study for 2D Euler equations. *European Series in Applied and Industrial Mathematics: Proceedings*, 29:28–42.

[Deiterding et al., 2006] Deiterding, R., Radovitzky, R., Mauch, S. P., Noels, L., Cummings, J. C., and Meiron, D. I. (2006). A virtual test facility for the efficient simulation of solid materials under high energy shock-wave loading. *Engineering with Computers*, 22(3-4):325–347.

[Deiterding and Wood, 2016] Deiterding, R. and Wood, S. L. (2016). Predictive wind turbine simulation with an adaptive lattice Boltzmann method for moving boundaries. *J. Phys. Conf. Series*, 753:082005.

[Feldhusen et al., 2016] Feldhusen, K., Deiterding, R., and Wagner, C. (2016). A dynamically adaptive lattice Boltzmann method for thermal convection problems. *J. Applied Math. and Computer Science*, 26:735–747.

References III

[Fragner and Deiterding, 2016] Fragner, M. M. and Deiterding, R. (2016). Investigating cross-wind stability of high speed trains with large-scale parallel cfd. *Int. J. Comput. Fluid Dynamics*, 30:402–407.

[Fragner and Deiterding, 2017] Fragner, M. M. and Deiterding, R. (2017). Investigating side-wind stability of high speed trains using high resolution large eddy simulations and hybrid models. In Diez, P., Neittaanmäki, P., Periaux, J., Tuovinen, T., and Bräysy, O., editors, *Computational Methods in Applied Sciences*, volume 45, pages 223–241. Springer.

[Gomes et al., 2015] Gomes, A. K. F., Domingues, M. O., Schneider, K., Mendes, O., and Deiterding, R. (2015). An adaptive multiresolution method for ideal magnetohydrodynamics using divergence cleaning with parabolic-hyperbolic correction. *Applied Numerical Mathematics*, 95:199–213.

[Hill and Pullin, 2004] Hill, D. J. and Pullin, D. I. (2004). Hybrid tuned center difference - WENO method for large eddy simulations in the presence of strong shocks. *J. Comput. Phys.*, 194(2):435–450.

[Hornung, 1972] Hornung, H. G. (1972). Non-equilibrium dissociating nitrogen flow over spheres and circular cylinders. *Journal of Fluid Mechanics*, 53:149–176.

[Liou and Steffen Jr, 1993] Liou, M.-S. and Steffen Jr, C. J. (1993). A new flux splitting scheme. *Journal of Computational Physics*, 107:23–39.

[Lobb, 1964] Lobb, K. R. (1964). Experimental Measurement of Shock Detachment Distance on Spheres Fired in Air at Hypervelocities. *High Temperature Aspects of Hypersonic Flows*, 14(5):519–527.

[Lombardini and Deiterding, 2010] Lombardini, M. and Deiterding, R. (2010). Large-eddy simulation of Richtmyer-Meshkov instability in a converging geometry. *Physics of Fluids*, 22(9):091112.

[Mauch, 2003] Mauch, S. P. (2003). *Efficient Algorithms for Solving Static Hamilton-Jacobi Equations*. PhD thesis, California Institute of Technology.

[Pantano et al., 2007] Pantano, C., Deiterding, R., Hill, D. J., and Pullin, D. I. (2007). A low-numerical dissipation patch-based adaptive mesh refinement method for large-eddy simulation of compressible flows. *J. Comput. Phys.*, 221(1):63–87.

[Perotti et al., 2013] Perotti, L. E., Deiterding, R., Inaba, K., Shepherd, J. E., and Ortiz, M. (2013). Elastic response of water-filled fiber composite tubes under shock wave loading. *Int. J. Solids and Structures*, 50(3-4):473–486.

References IV

[Pezzella et al., 2015] Pezzella, G., de Rosa, D., and Donelli, R. (2015). Computational Analysis of Shock Wave Boundary Layer Interactions in Non-equilibrium Hypersonic Flow. *20th AIAA International Space Planes and Hypersonic Systems and Technologies Conference*, (July):1–12.

[Scoggins and Magin, 2014] Scoggins, J. B. and Magin, T. E. (2014). Development of Mutation++ : MUlticomponent Thermodynamics And Transport properties for IONized gases library in C++. In *11th AIAA/ASME Joint Thermophysics and Heat Transfer Conference*.

[Sethian, 1999] Sethian, J. A. (1999). *Level set methods and fast marching methods*. Cambridge University Press, Cambridge, New York.

[Souza Lopes et al., 2018] Souza Lopes, M. M., Deiterding, R., Gomes, A. K. F., Mendes, O., and Domingues, M. O. (2018). An ideal compressible magnetohydrodynamic solver with parallel block-structured adaptive mesh refinement. *Computers & Fluids*. in press.

[van Leer, 1982] van Leer, B. (1982). Flux Vector Splitting for the Euler Equations. In *Eighth international conference on numerical methods in fluid dynamics*.

[Wan et al., 2017] Wan, Q., Jeon, H., Deiterding, R., and Eliasson, V. (2017). Numerical and experimental investigation of oblique shock wave reflection off a water wedge. *J. Fluid Mech.*, 826:732–758.

[Ziegler et al., 2011] Ziegler, J. L., Deiterding, R., Shepherd, J. E., and Pullin, D. I. (2011). An adaptive high-order hybrid scheme for compressive, viscous flows with detailed chemistry. *J. Comput. Phys.*, 230(20):7598–7630.

[Zonglin et al., 2002] Zonglin, J., Matsuoka, K., Sasoh, A., and Takayama, K. (2002). Numerical and experimental investigation of wave dynamics processes in high-speed train/tunnels. *Chinese Journal of Mechanics Press*, 18(3):210–226.

Clustering by signatures

				x	x	x	x	x	x	6
				x	x	x	x	x	x	6
			x	x	x					3
	x	x	x							3
	x	x								2
	x	x								2
	x	x								2
										0
	x	x								2
	x	x								2
Υ	6	6	2	3	2	2	2	2	2	

Υ Flagged cells per row/column

Δ Second derivative of Υ, $\Delta = \Upsilon_{\nu+1} - 2\,\Upsilon_{\nu} + \Upsilon_{\nu-1}$

Technique from image detection: [Bell et al., 1994], see also [Berger and Rigoutsos, 1991], [Berger, 1986]

Clustering by signatures

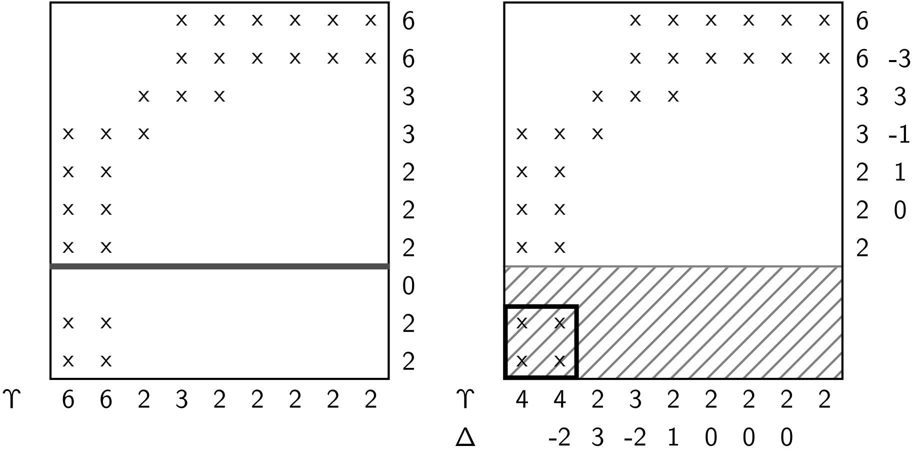
				x	x	x	x	x	x	6
				x	x	x	x	x	x	6
			x	x	x					3
	x	x	x							3
	x	x								2
	x	x								2
	x	x								2
										0
	x	x								2
	x	x								2
Υ	6	6	2	3	2	2	2	2	2	

Υ Flagged cells per row/column

Δ Second derivative of Υ, $\Delta = \Upsilon_{\nu+1} - 2\,\Upsilon_{\nu} + \Upsilon_{\nu-1}$

Technique from image detection: [Bell et al., 1994], see also [Berger and Rigoutsos, 1991], [Berger, 1986]

Clustering by signatures

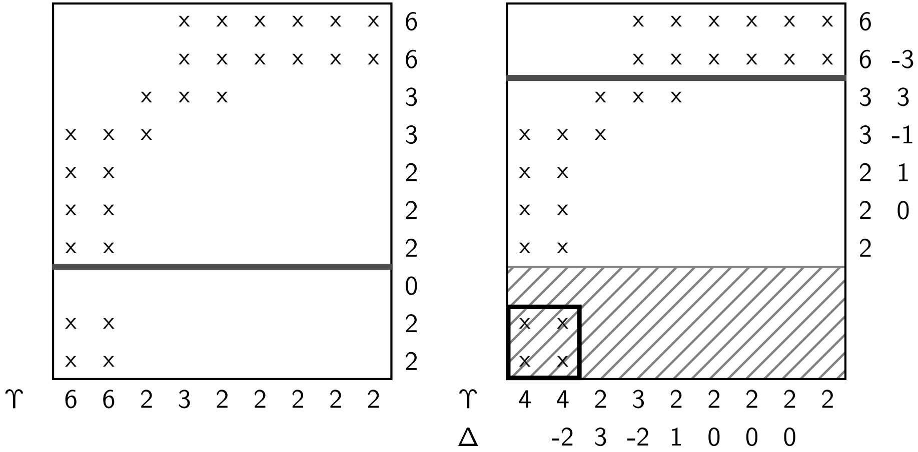


Υ Flagged cells per row/column

Δ Second derivative of Υ, $\Delta = \Upsilon_{\nu+1} - 2\,\Upsilon_{\nu} + \Upsilon_{\nu-1}$

Technique from image detection: [Bell et al., 1994], see also [Berger and Rigoutsos, 1991], [Berger, 1986]

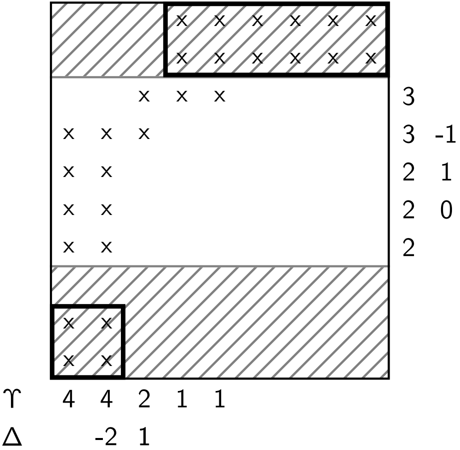
Clustering by signatures



Υ Flagged cells per row/column

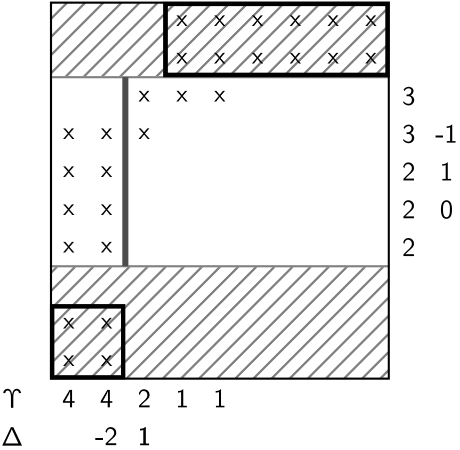
Δ Second derivative of Υ, $\Delta = \Upsilon_{\nu+1} - 2\,\Upsilon_{\nu} + \Upsilon_{\nu-1}$

Technique from image detection: [Bell et al., 1994], see also [Berger and Rigoutsos, 1991], [Berger, 1986]



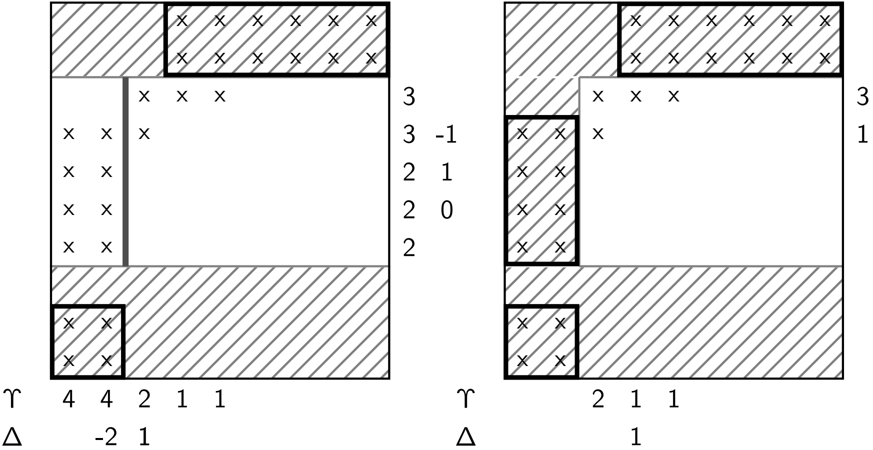
Recursive generation of $\breve{G}_{l,m}$

1. 0 in Υ
2. Largest difference in Δ
3. Stop if ratio between flagged and unflagged cell $> \eta_{tol}$



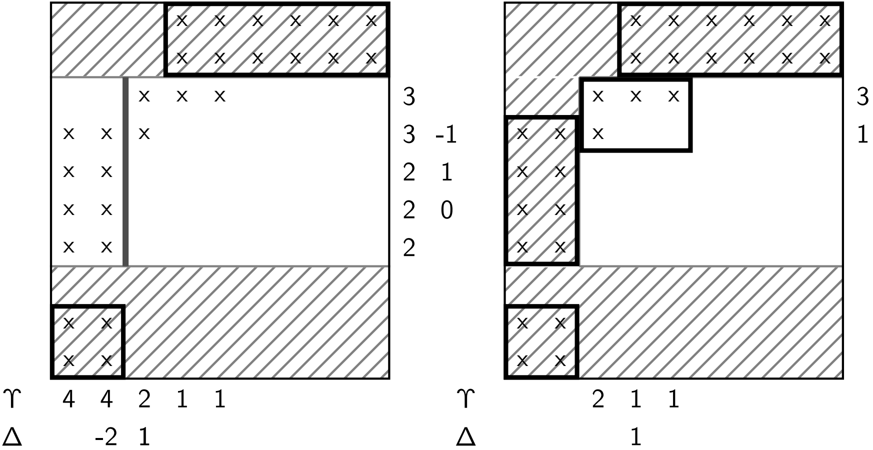
Recursive generation of $\breve{G}_{l,m}$

1. 0 in Υ
2. Largest difference in Δ
3. Stop if ratio between flagged and unflagged cell $> \eta_{tol}$



Recursive generation of $\breve{G}_{l,m}$

1. 0 in Υ
2. Largest difference in Δ
3. Stop if ratio between flagged and unflagged cell $> \eta_{tol}$



Recursive generation of $\breve{G}_{l,m}$

1. 0 in Υ
2. Largest difference in Δ
3. Stop if ratio between flagged and unflagged cell $> \eta_{tol}$

Closest point transform algorithm

The signed distance φ to a surface $\mathcal{I}$ satisfies the eikonal equation [Sethian, 1999]

$$|\nabla\varphi| = 1 \quad \text{with} \quad \varphi\big|_{\mathcal{I}} = 0$$

Solution smooth but non-diferentiable across characteristics.

Closest point transform algorithm

The signed distance φ to a surface $\mathcal{I}$ satisfies the eikonal equation [Sethian, 1999]

$$|\nabla\varphi| = 1 \quad \text{with} \quad \varphi\big|_{\mathcal{I}} = 0$$

Solution smooth but non-diferentiable across characteristics.

Distance computation trivial for non-overlapping elementary shapes but difficult to do efficiently for triangulated surface meshes:

- Geometric solution approach with plosest-point-transform algorithm [Mauch, 2003]

Closest point transform algorithm

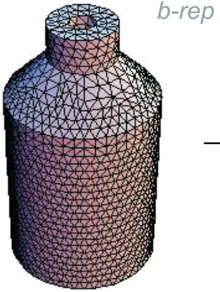
The signed distance φ to a surface $\mathcal{I}$ satisfies the eikonal equation [Sethian, 1999]

$$|\nabla\varphi| = 1 \quad \text{with} \quad \varphi\big|_{\mathcal{I}} = 0$$

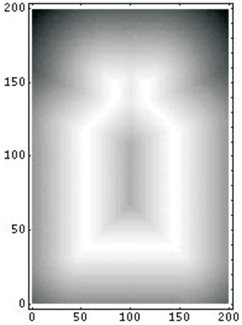
Solution smooth but non-diferentiable across characteristics.

Distance computation trivial for non-overlapping elementary shapes but difficult to do efficiently for triangulated surface meshes:

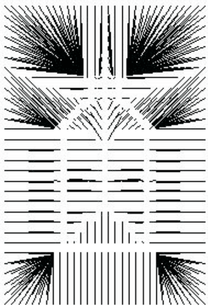
- Geometric solution approach with plosest-point-transform algorithm [Mauch, 2003]



Surface mesh $\mathcal{I}$



Distance φ

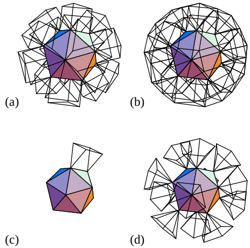


Normal to closest point

The characteristic / scan conversion algorithm

1. Build the characteristic polyhedrons for the surface mesh

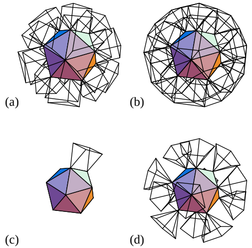
Characteristic polyhedra for faces, edges, and vertices



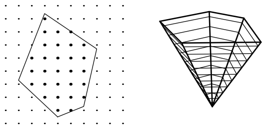
The characteristic / scan conversion algorithm

1. Build the characteristic polyhedrons for the surface mesh
2. For each face/edge/vertex
   2.1 Scan convert the polyhedron.

Characteristic polyhedra for faces, edges, and vertices



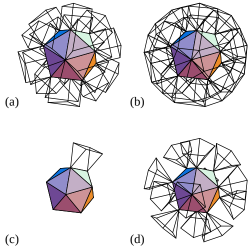
Slicing and scan conversion of apolygon



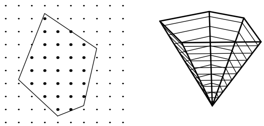
The characteristic / scan conversion algorithm

1. Build the characteristic polyhedrons for the surface mesh
2. For each face/edge/vertex
   2.1 Scan convert the polyhedron.
   2.2 Compute distance to that primitive for the scan converted points

Characteristic polyhedra for faces, edges, and vertices



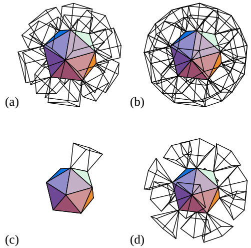
Slicing and scan conversion of apolygon



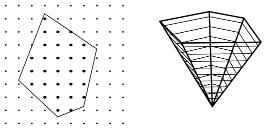
The characteristic / scan conversion algorithm

1. Build the characteristic polyhedrons for the surface mesh
2. For each face/edge/vertex
   2.1 Scan convert the polyhedron.
   2.2 Compute distance to that primitive for the scan converted points
3. Computational complexity.
   - $O(m)$ to build the b-rep and the polyhedra.
   - $O(n)$ to scan convert the polyhedra and compute the distance, etc.

Characteristic polyhedra for faces, edges, and vertices



Slicing and scan conversion of apolygon

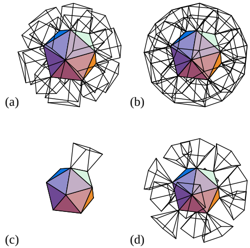


The characteristic / scan conversion algorithm

1. Build the characteristic polyhedrons for the surface mesh
2. For each face/edge/vertex
   2.1 Scan convert the polyhedron.
   2.2 Compute distance to that primitive for the scan converted points
3. Computational complexity.
   - $O(m)$ to build the b-rep and the polyhedra.
   - $O(n)$ to scan convert the polyhedra and compute the distance, etc.
4. Problem reduction by evaluation only within specified max. distance

[Mauch, 2003], see also [Deiterding et al., 2006]

Characteristic polyhedra for faces, edges, and vertices



Slicing and scan conversion of apolygon