Supercomputer modeling compressible flows with a Cartesian grid method

I. Menshov, KIAM RAS

P. Pavlukhin, Research center «Kvant», KIAM RAS

Japan-Russia Workshop on Supercomputer Modeling, Instability and Turbulence in Fluid Dynamics (JRSMIT), March 3-6, 2015*, KIAM RAS, Moscow*

Motivation

- Supercomputers today and of the near future – hybrid SIMD architectures with massive multithreading systems (GPGPU, PHI).
- Effectiveness of using these systems depends on the complexity of numerical methods to be implemented.
- Simple explicit schemes are best fitted for parallel implementation on multithreading supercomputer systems. Minus – time step restriction.
- More accurate simulations we need, more fine grid spatial resolutions are required.

Deadlock situation: an increase in problem size leads to smaller time steps, and drastical (as square) growing of computational work.

Purposes

- Develop a numerical approach well fitted for using on modern and perspective massively multithreaded supercomputer systems.
- Basic requirements to the method should be:
  - algorithmic homogeneity;
  - computational primitivism;
  - with no stiff limitation on time step;
  - simple gridding the computational domain;
  - handling complex spatial geometries;
- Cartesian grid discretization most closely meets the requirements. ⇒ Aim: devise a Cartesian grid method (1) with no time step restriction (2) able to work with complex geometries.

Basic numerical method

Governing equations:

$$\frac{\partial \mathbf{q}}{\partial t}+\frac{\partial \mathbf{f}_k}{\partial x_k}=0$$

A Cartesian grid $\{h_k\}$: semidiscrete equations

$$\frac{d\mathbf{q}_i}{dt}=-\left(\frac{\Delta \mathbf{F}_k}{h_k}\right)_i \qquad \Delta \mathbf{F}_k=\mathbf{F}_{k,i+1/2}-\mathbf{F}_{k,i-1/2} \qquad \mathbf{F}_{k,i+1/2}=\mathbf{F}_k\left(\mathbf{z}_i^+,\mathbf{z}_{i+1}^-\right)$$

MUSCL-type cell reconstruction: $\mathbf{z}^{\pm}=\mathbf{z}\pm 0.5s\delta^{\pm}\left[(1-sk^{\pm})\Delta^{\mp}+(1+sk^{\pm})\Delta^{\pm}\right]$

Godunov-type flux approximation:

$\mathbf{F}_k\left(\mathbf{z}_i^+,\mathbf{z}_{i+1}^-\right)=\mathbf{f}_k\left[\mathbf{Z}^{R,k}\left(0,\mathbf{z}_i^+,\mathbf{z}_{i+1}^-\right)\right]$ - exact

$\mathbf{F}_k\left(\mathbf{z}_i^+,\mathbf{z}_{i+1}^-\right)=\frac{1}{2}\left\{\mathbf{f}_k\left(\mathbf{z}_i^+\right)+\mathbf{f}_k\left(\mathbf{z}_{i+1}^-\right)-\left(|u_k|+c\right)_{i+1/2}\left[\mathbf{q}\left(\mathbf{z}_{i+1}^-\right)-\mathbf{q}\left(\mathbf{z}_i^+\right)\right]\right\}$ - approximate (Rusanov)

Two-stage predictor/corrector explicit scheme:

$$\tilde{\mathbf{q}}_i=\mathbf{q}_i^n-\frac{\Delta t}{2}\left(\frac{\Delta \mathbf{F}_k(\mathbf{z}^n)}{h_k}\right)_i \quad \Longrightarrow \quad \mathbf{q}_i^{n+1}=\mathbf{q}_i^n-\Delta t\left(\frac{\Delta \mathbf{F}_k(\tilde{\mathbf{z}})}{h_k}\right)_i$$

The resulting scheme is warranted to be stable provided that the CFL-condition valid:

$\Delta t\le \lambda_i(\mathbf{z}^n)$ for all i

$$\lambda_i(\mathbf{z}^n)=K_s\left(\frac{|u_k|+C}{h_k}\right)^{-1}$$

Relaxing Timestep Restriction: Explicit/Implicit Hybridization

An intermediate time level parameter: $\omega_i, \quad 0 \le \omega_i \le 1.$

An intermediate time level solution vector: $\mathbf{q}^{\omega} = \omega \mathbf{q}^{n} + (1-\omega)\mathbf{q}^{n+1}.$

Write the baseline explicit scheme: $\mathbf{q}_i^{n+1} = \mathbf{q}_i^{n} + \Delta t L_2\left(\Delta t, \mathbf{q}^{n}\right)$ ⟹

⟹ **The hybrid explicit/implicit scheme**: $\mathbf{q}_i^{n+1} = \mathbf{q}_i^{n} + \Delta t L_2\left(\omega_i \Delta t, \mathbf{q}^{\omega}\right)$

The ω should be taken much closer to 1 providing stability holds: $\left\|\mathbf{q}^{n+1}\right\|_{\infty} \le \left\|\mathbf{q}^{n}\right\|_{\infty}$

Lemma. If ω is taken so that $\left\|\mathbf{q}^{n+1}\right\|_{\infty} \le \left\|\mathbf{q}^{\omega}\right\|_{\infty}$, then necessarily $\left\|\mathbf{q}^{n+1}\right\|_{\infty} \le \left\|\mathbf{q}^{n}\right\|_{\infty}$

Recasting the hybrid scheme: $\mathbf{q}_i^{n+1} = \mathbf{q}_i^{\omega} + \omega_i \Delta t L_2\left(\omega_i \Delta t, \mathbf{q}^{\omega}\right)$ ⟹

It looks as the explicit scheme launched from $t^{\omega} = \omega t^{n} + (1-\omega) t^{n+1}$ to t^{n+1}

Therefore, $\left\|\mathbf{q}^{n+1}\right\|_{\infty} \le \left\|\mathbf{q}^{\omega}\right\|_{\infty}$ providing that the CFL is correct, $\omega_i \Delta t \le \lambda_i(\mathbf{z}^{\omega})$

⟹ $\omega_i = \min\left[1, \dfrac{\lambda_i(\mathbf{z}^{\omega})}{\Delta t}\right]$ or $\omega_i = \min\left[1, \dfrac{K_s}{\Delta t}\left(\dfrac{\left|u_k^{\omega}\right| + C^{\omega}}{h_k}\right)_i^{-1}\right]$

Dealing with geometry on Cartesian grids: Free Boundary

Standard boundary value problem: $\frac{\partial \mathbf{q}}{\partial t}+\frac{\partial \mathbf{f}_k}{\partial x_k}=0 \qquad u_k n_k=0,\ \mathbf{x}\in\Gamma$ (A)

Alternative statement – in the whole space: $\frac{\partial \mathbf{q}}{\partial t}+\frac{\partial \mathbf{f}_k}{\partial x_k}=-\mathbf{F}_w \qquad \in\ \Re^3$ (B)

⟹ **The compensating flux $\mathbf{F}_w$: $\mathbf{q}_A=\mathbf{q}_B|_\Omega$**

This is achieved if the compensating flux is taken in the form:

$$\mathbf{F}_w=\begin{pmatrix}\rho u_k n_k\\ \rho u_k u_m n_k+(p-p_w)n_m\\ \rho u_k n_k H\end{pmatrix}\delta(\mathbf{x},\Gamma)$$

with $\delta(\mathbf{x},\Gamma)$ being Dirac's function of the surface Γ:

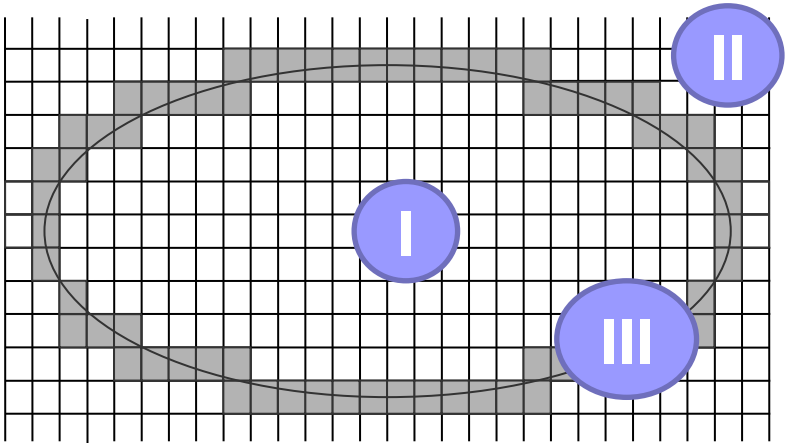
$$\int_V \delta(\mathbf{x},S)\varphi(\mathbf{x})dV=\int_{V\cap\Gamma}\varphi(\mathbf{x})dS,\quad \forall V\in\Re^3$$

p_w - is the wall reaction pressure (depends on local *M*): e.g., if $u_k n_k<0$

$$p_w=p\left[1+\frac{\gamma(\gamma+1)}{4}M^2+\sqrt{\gamma^2M^2+\frac{\gamma^2(\gamma+1)^2}{16}M^4}\right]$$

Numerical implementation:

A Cartesian grid overlaps the geometry ⟶

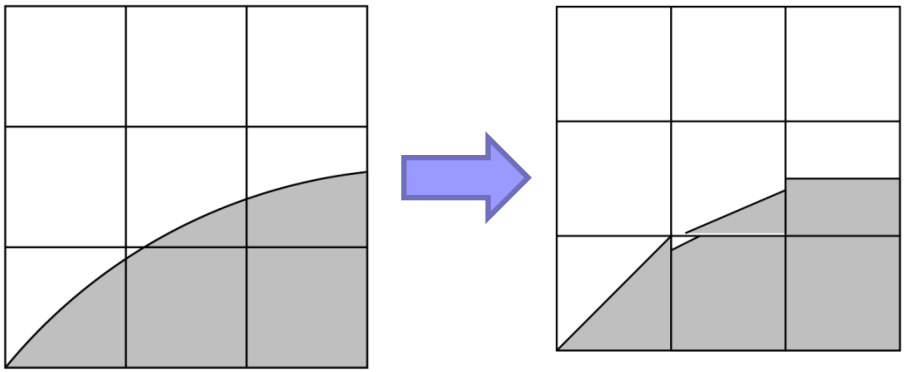


Computational cells are classified into 3 groups:
- solid cells (I);
- fluid cells (II) cut cells;
- cut cells (III).

A linear sub-cell structure of the geometry is introduced for cut cells:

ω_f = the volume fraction occupied by fluid;

$\mathbf{n}_f$, $|\mathbf{n}_f| = s_f$ - outward normal, s_f = the area of the geometry inside the cell.



Numerical integration: splitting in physics

Calculations are performed in two stages.

- I) Baseline calculation loop over the set of **fluid and cut** cells: hybrid explicit/implicit

$$\mathbf{q}_i^* = \mathbf{q}_i^n - \Delta t \left(\frac{\Delta \mathbf{F}_k(\tilde{\mathbf{z}})}{h_k} \right)_i \qquad \mathbf{F}_{k,i+1/2} = \mathbf{F}_k\left(\tilde{\mathbf{z}}_i^+, \tilde{\mathbf{z}}_{i+1}^-\right)$$ **with no geometry**

- II) Correction **parameters of cut cells:** integrating over fluid part of the cell

$$\omega_f V_i \frac{d\mathbf{q}_i}{dt} = -\sum_{\sigma \in f} \mathbf{F}_\sigma s_\sigma + \mathbf{F}_p s_f \Rightarrow \omega_f V_i \frac{d\mathbf{q}_i}{dt} = -\mathbf{F}_c s_f + \mathbf{F}_p s_f$$

$$\mathbf{F}_p = (0, p_w n_m, 0)^T \qquad \mathbf{F}_c = \left(\rho u_k n_k, \rho u_k u_m n_k + p n_m, \rho u_k n_k H\right)^T \qquad \mathbf{F}_w = \mathbf{F}_c - \mathbf{F}_p$$

Applying implicit time integration:

$$\mathbf{q}_i^{n+1} = \mathbf{q}_i^* - \frac{\Delta t s_f}{\omega_f V_i} \mathbf{F}_w\left(\mathbf{q}_i^{n+1}\right) \Rightarrow$$

$$\mathbf{q}_i^{n+1} = \mathbf{q}_i^n - \Delta t \left(\frac{\Delta \mathbf{F}_k(\tilde{\mathbf{z}})}{h_k} \right)_i - \frac{\Delta t s_f}{\omega_f V_i} \mathbf{F}_w\left(\mathbf{q}_i^{n+1}\right),$$

$$\mathbf{F}_{k,i+1/2} = \mathbf{F}_k\left(\tilde{\mathbf{z}}_i^+, \tilde{\mathbf{z}}_{i+1}^-\right),$$

To handle the geometry on a Cartesian grid we need only ω_f **and** $\mathbf{n}_f$, $|\mathbf{n}_f| = s_f$

Solving discrete equations: LU-SGS

If $\omega \neq 1$, the scheme is implicit and needs solving non-linear equations:
Newton's iterations -

$$\left(I + \frac{\Delta t s_f}{\omega_f V_i} A_w^s \right) \delta^s \mathbf{q}_i = -\Delta^s \mathbf{q}_i - \Delta t \left(\frac{\Delta^s \mathbf{F}_k}{h_k} \right)_i - \frac{\Delta t s_f}{\omega_f V_i} \mathbf{F}_w \left(\mathbf{q}_i^{n+1,s} \right) -$$

$$- \Delta t \frac{\delta^s \mathbf{F}_{k,i+1/2} - \delta^s \mathbf{F}_{k,i-1/2}}{h_k}$$

When linearizing, we admit approximations:

- intermediate level parameter ω is frozen;
- instead of Godunov we use Rusanov flux;
- no sub-cell interpolation (1st order scheme).

$\Rightarrow$ $D\delta\mathbf{q} + L(\delta\mathbf{q}) + U(\delta\mathbf{q}) = \mathbf{R}$

With approximate factorization: $(D+L)D^{-1}(D+U)\delta\mathbf{q} = \mathbf{R}$ $\Rightarrow$ 2 sweeps:

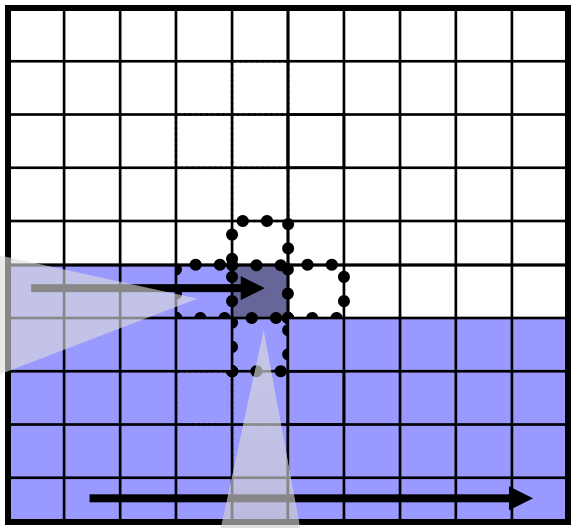
forward - $\delta\mathbf{q}^* = D^{-1}[\mathbf{R} - L(\delta\mathbf{q}^*)]$,

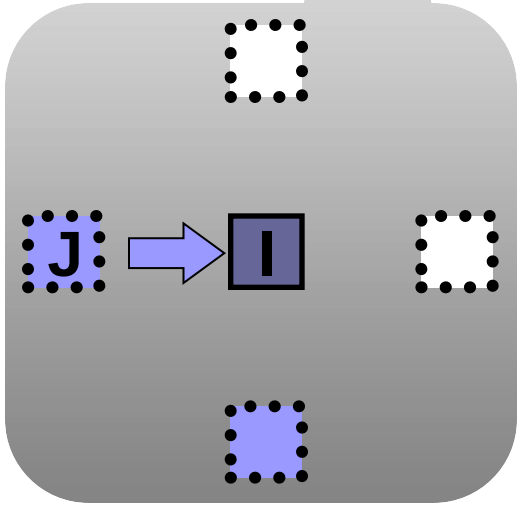
backward - $\delta\mathbf{q}^s = \delta\mathbf{q}^* - D^{-1}U(\delta\mathbf{q}^s)$.

Coding algorithm for multithreaded systems:

```
if (I > J) {//сосед обсчитан
   ...
   cell[I] = f1(cell[J]);
   ...
}

if (I < J) {//сосед не обсчитан
   ...
   cell[I] = f2(cell[J]);
   cell[J]+= ...
}
```



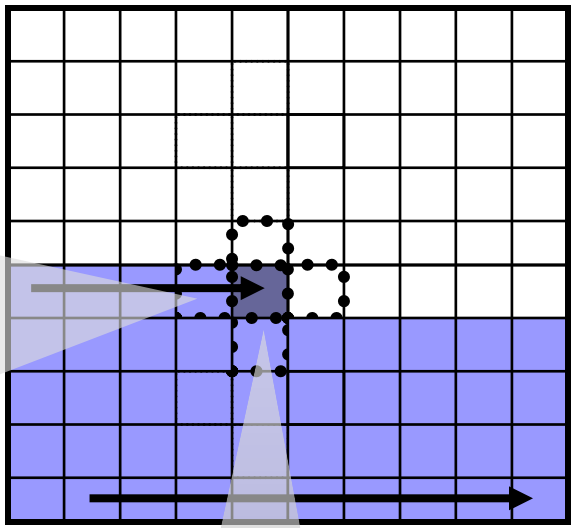


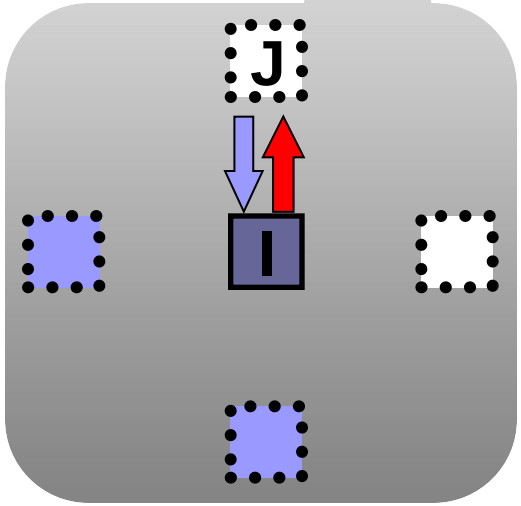
- updated
- not yet updated

Coding algorithm for multithreaded systems:

```
if (I > J) {//сосед обсчитан
   ...
   cell[I] = f1(cell[J]);
   ...
}
```

```
if (I < J) {//сосед не обсчитан
   ...
   cell[I] = f2(cell[J]);
   cell[J]+= ...
}
```





- updated
- not yet updated

Coding algorithm for multithreaded systems:

An algorithm of two-level parallelism is developed to realized the numerical method on multithreaded high-performance cluster systems.

The parallel computational work is organized in two levels:

- Splitting the work between several GPU;
- Multithreaded computing in the GPU;

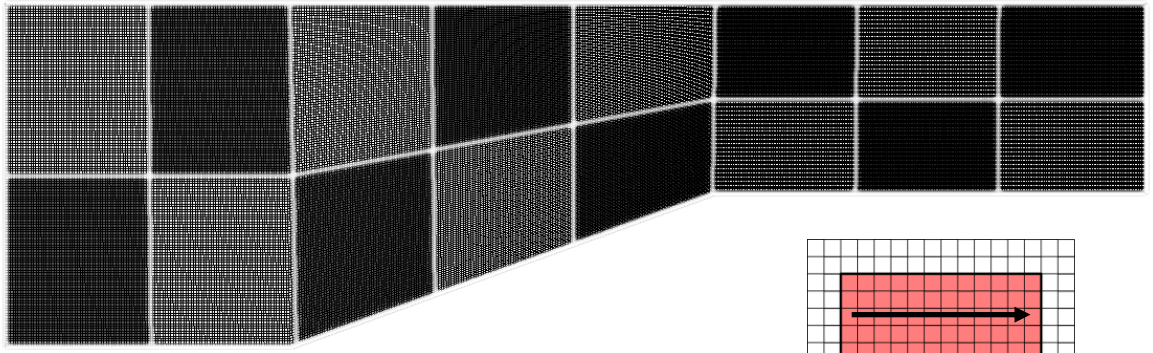
Two things should be paid attention to:

- Avoiding data coalescence;
- Exact reproducing the original LU-SGS method;

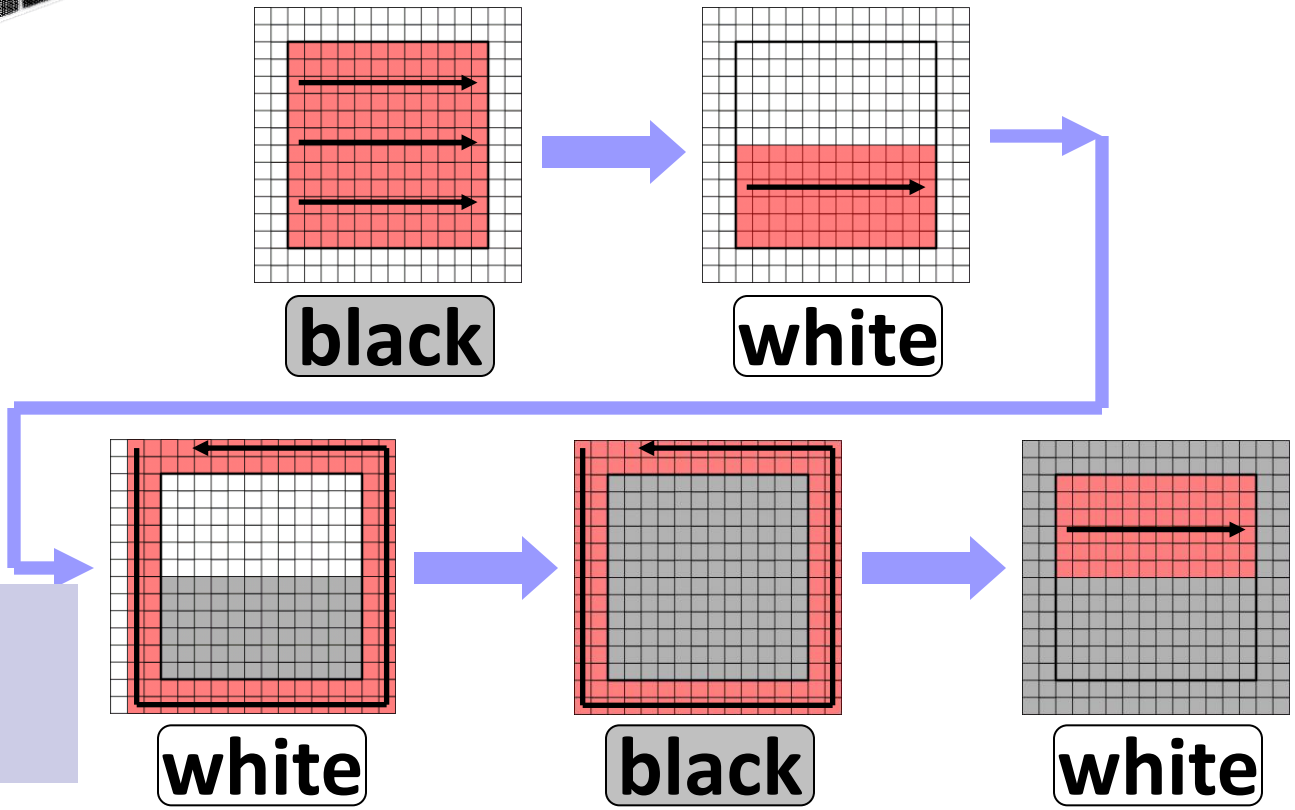
To meet these requirements: **device a special order of performing working loops** over computational cells.

Global order of performing loops: chess-type stepping

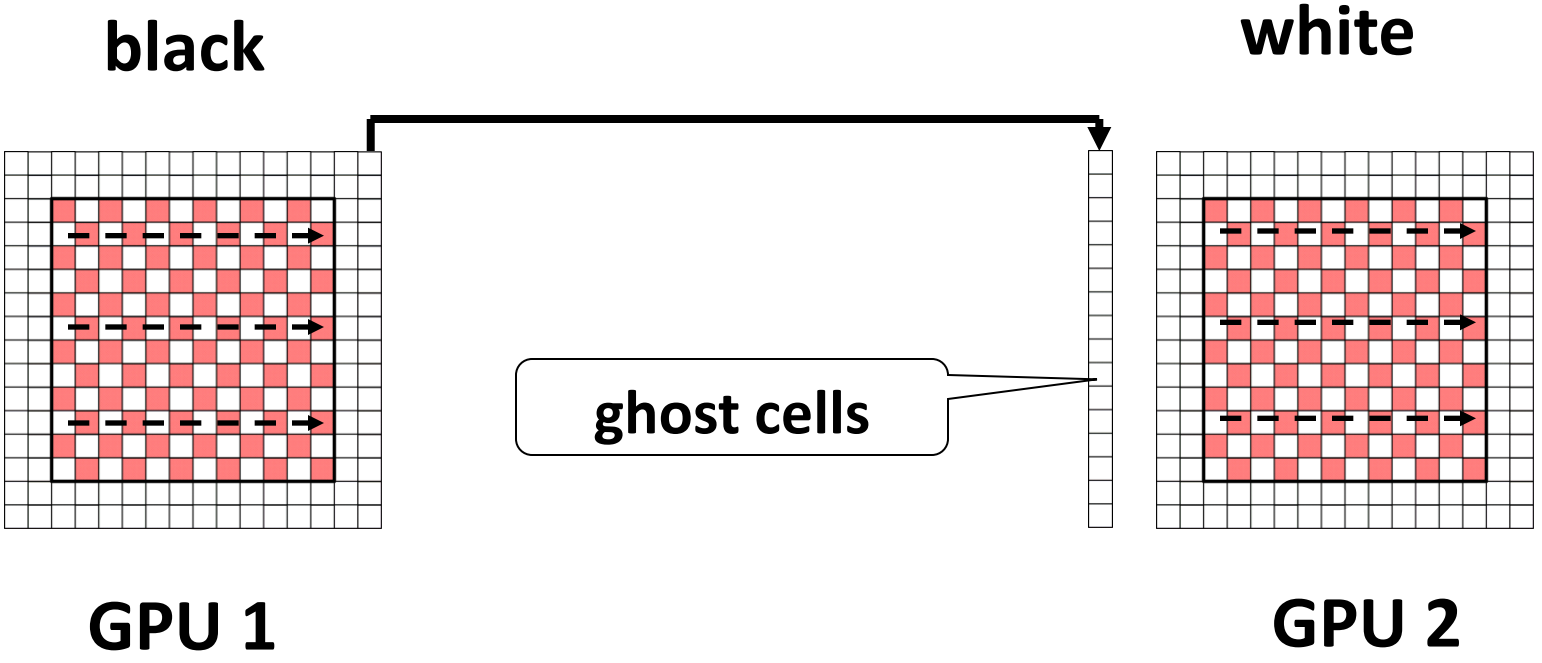
1. Decomposition of the computational domain – «black» и «white» blocks:



2. Performing the loop:



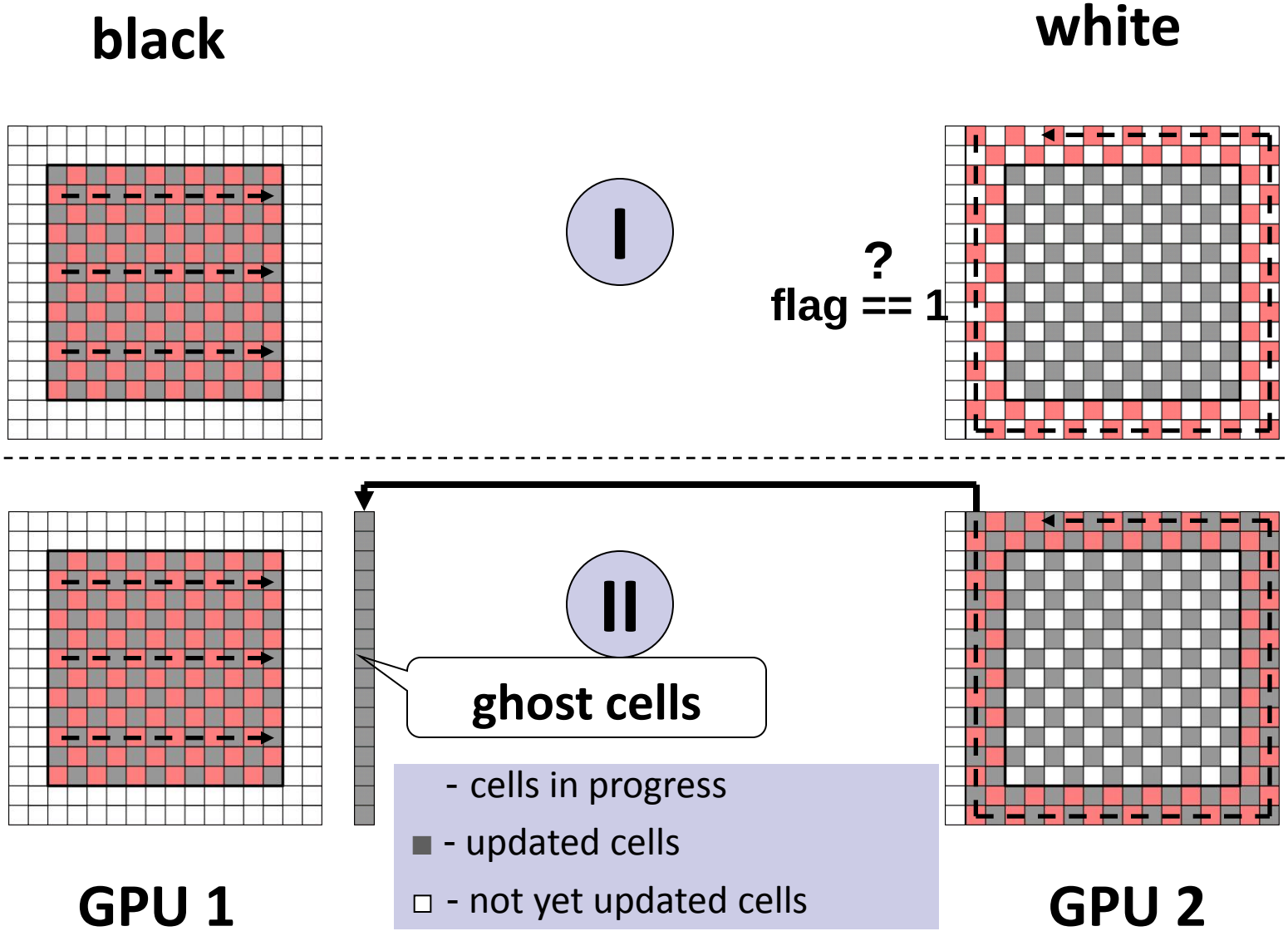
Example of stepping (multi-GPU): stage I



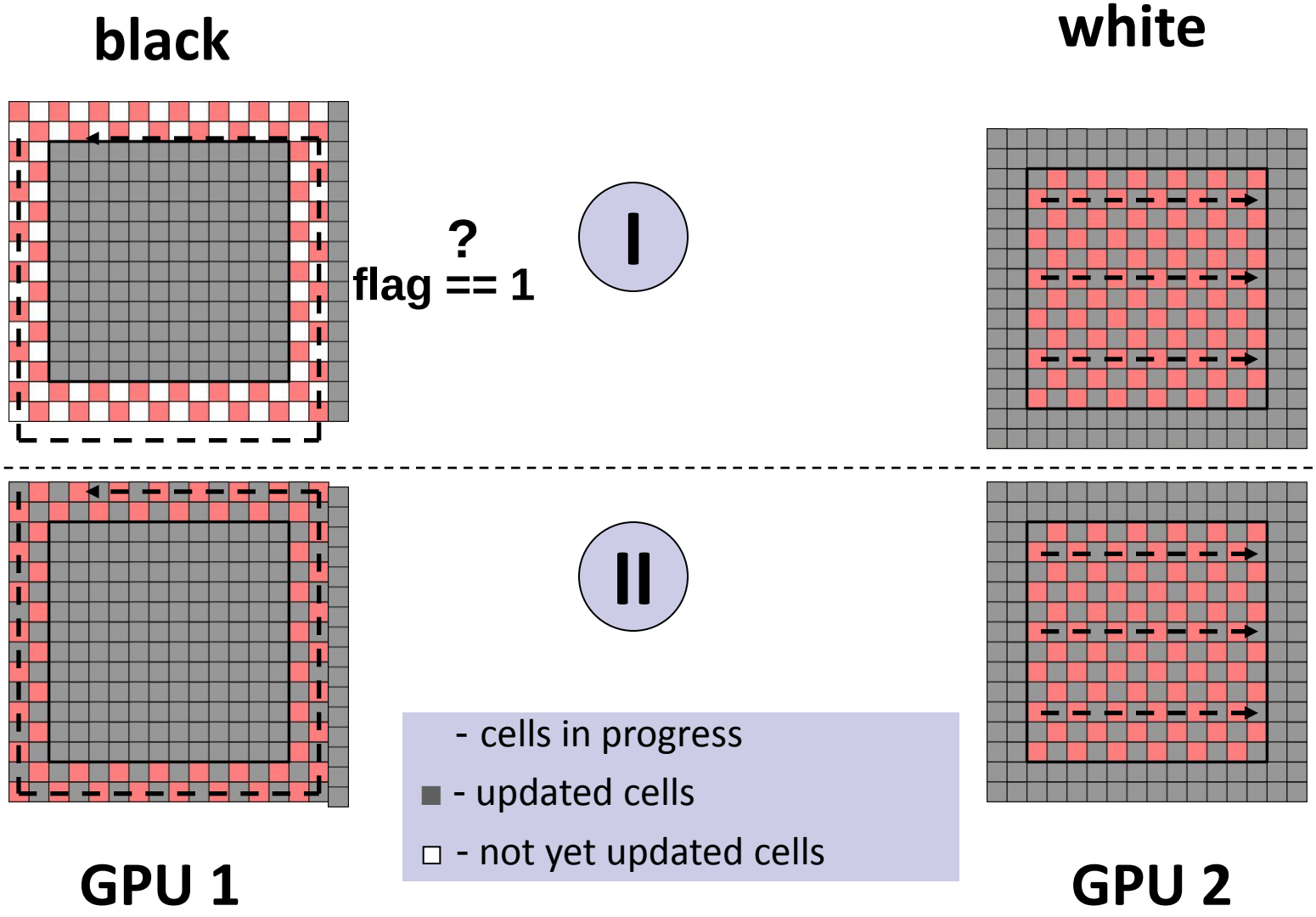
Doubled boundary perimeter; inner cells are ordered chess-like

- cells in progress
■ - updated cells
□ - not yet updated cells

Example of stepping (multi-GPU): stage II



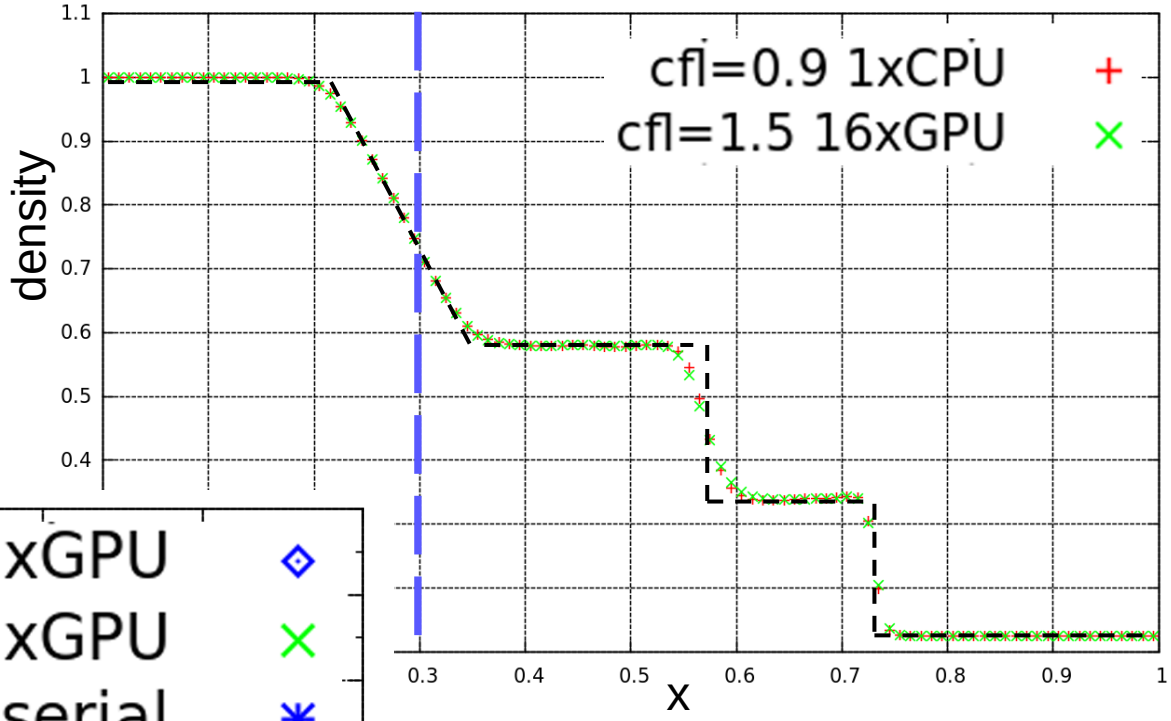
Example of stepping (multi-GPU): stage II

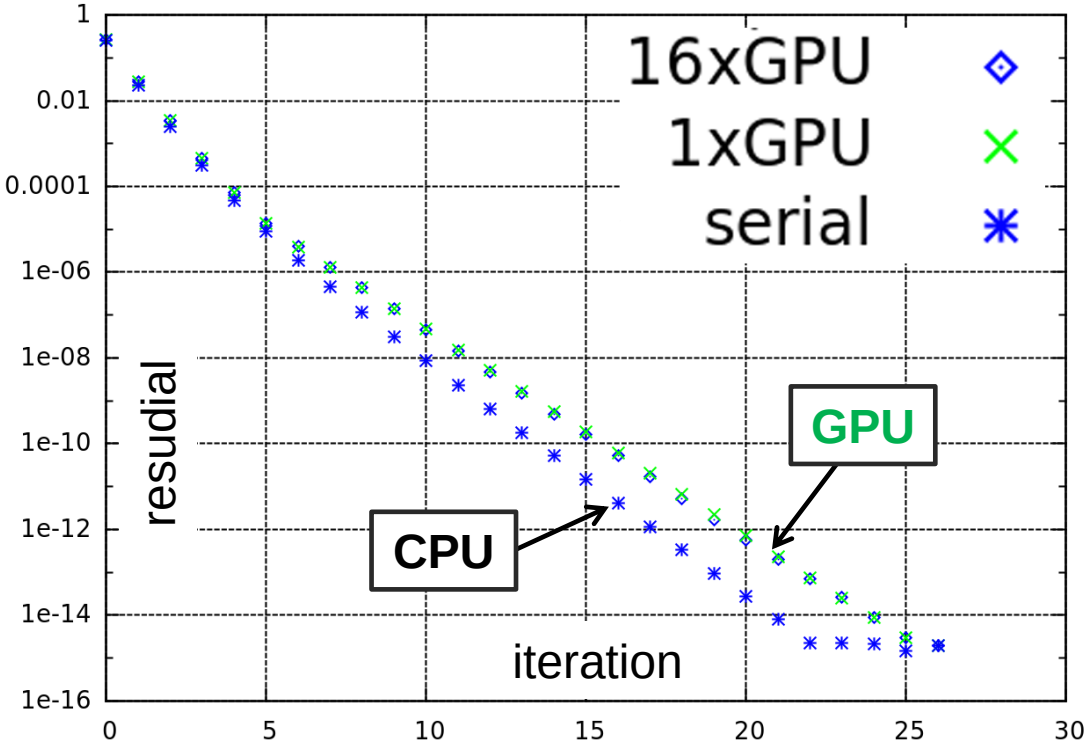


- Simulteneously: computing and copying to Cuda (streams) and MPI;

Numerical Results:Scheme

Modified Sod's problem



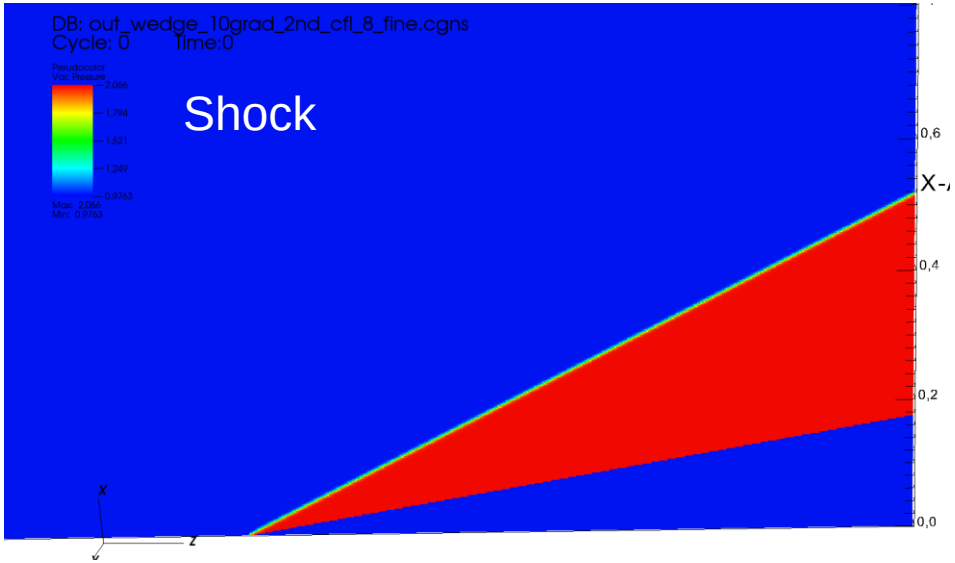


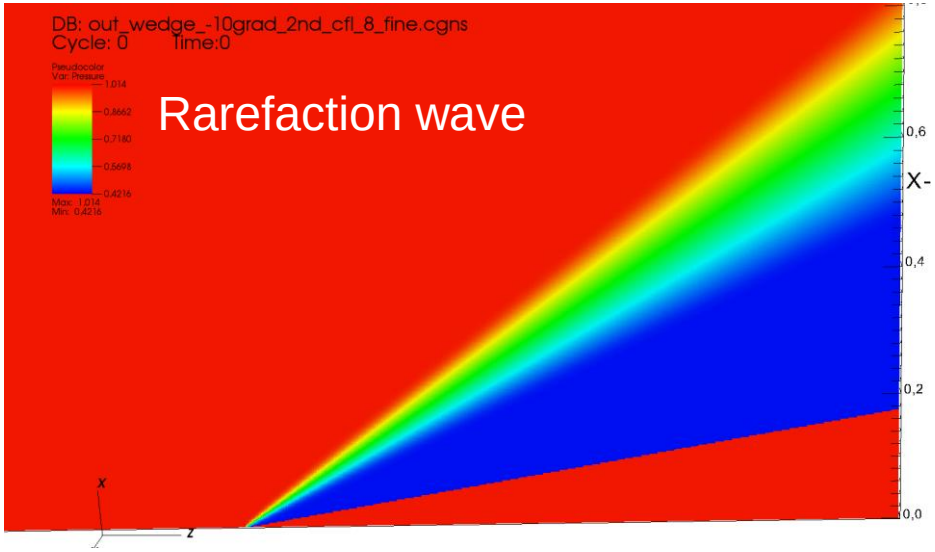
Residual vs iterations

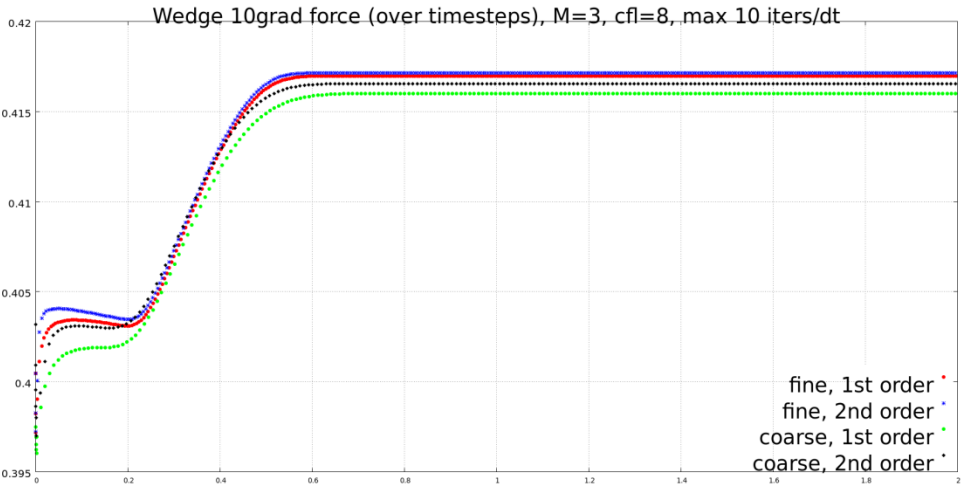
Numerical results: Steady

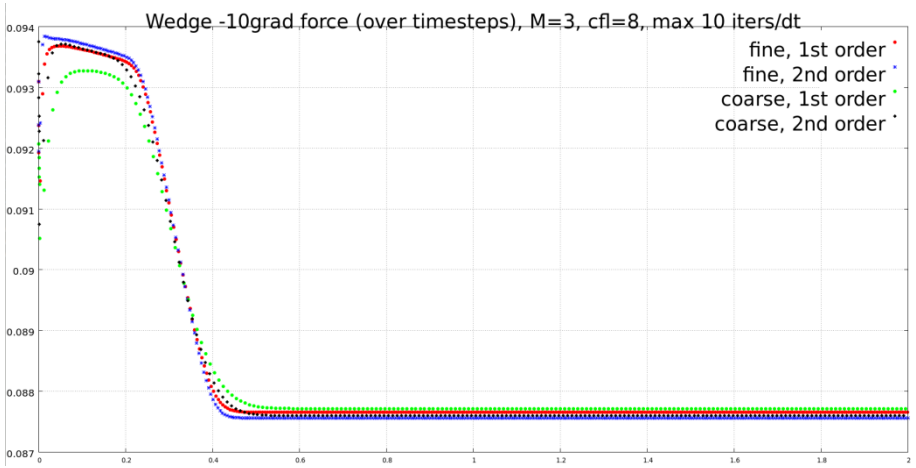
Wedge 10°, +/-10°, inflow M=3

Numerical, β°	Analytical, β°	Numerical, γ°	Analytical, γ°
17.4	17.383	13.2	13.24



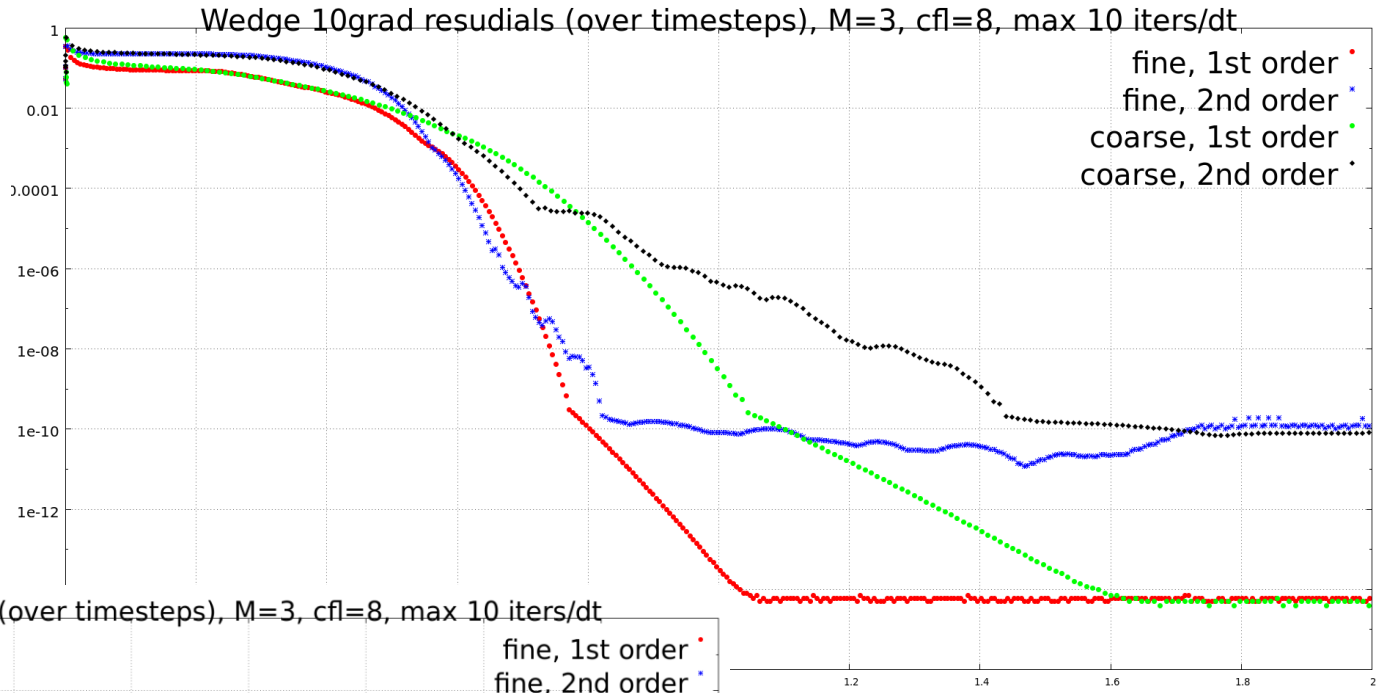


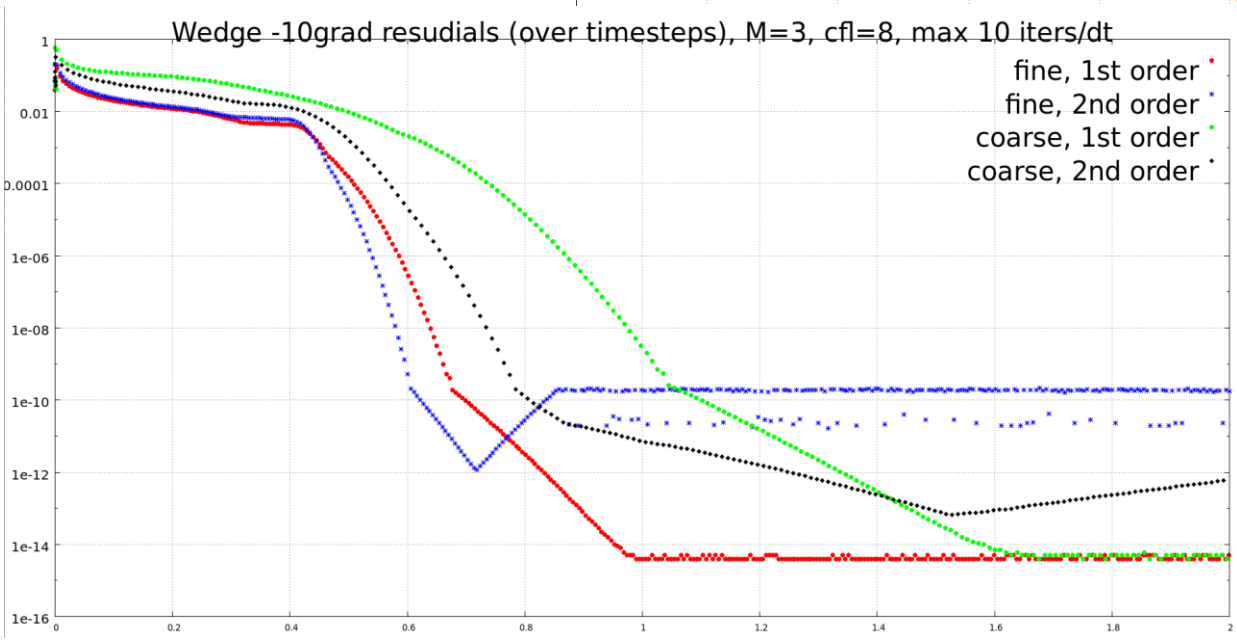




Numerical Results: Steady

Wedge 10°, Shock

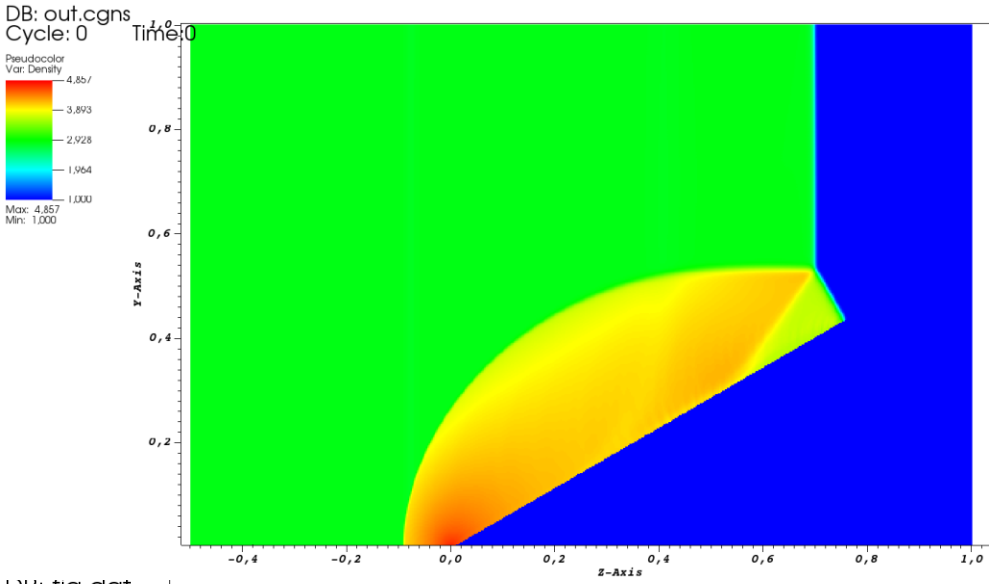




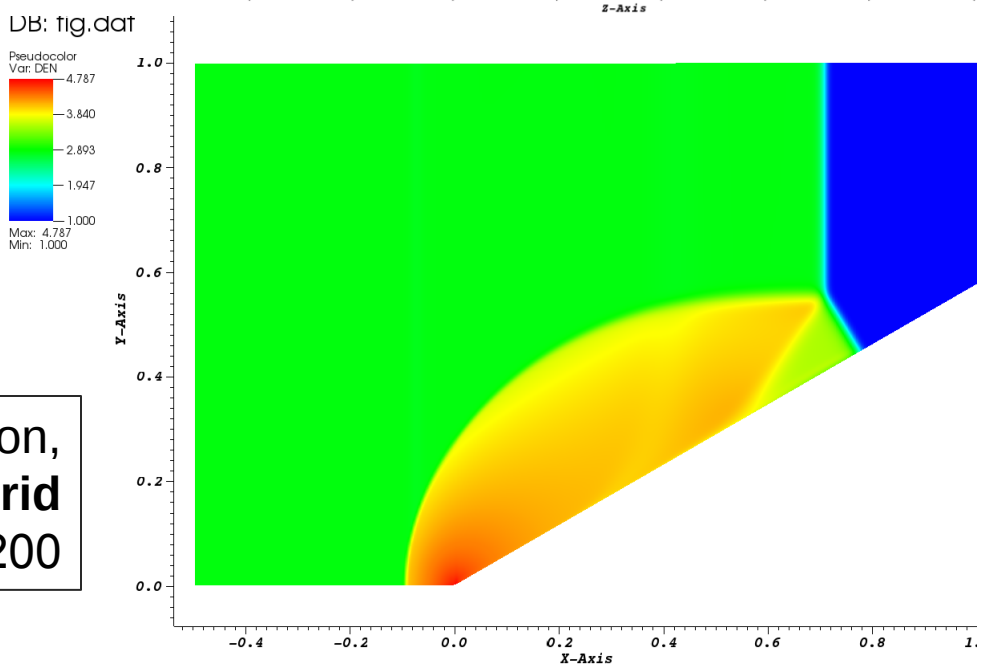
Wedge 10°, rarefaction wave

Numerical Results: Unsteady

3D, pressure distribution, t=0.32 **Cartesian grid** 600x400x6, free boundary method



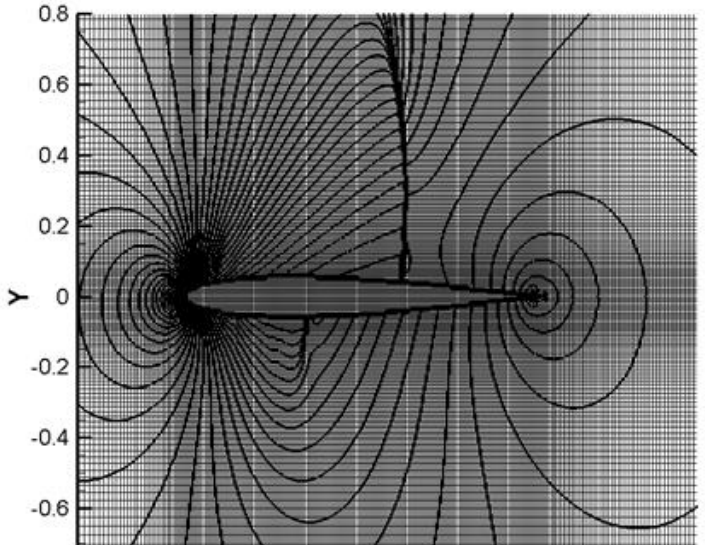
Wedge, M=2.12, θ=30°.
Shock difraction.

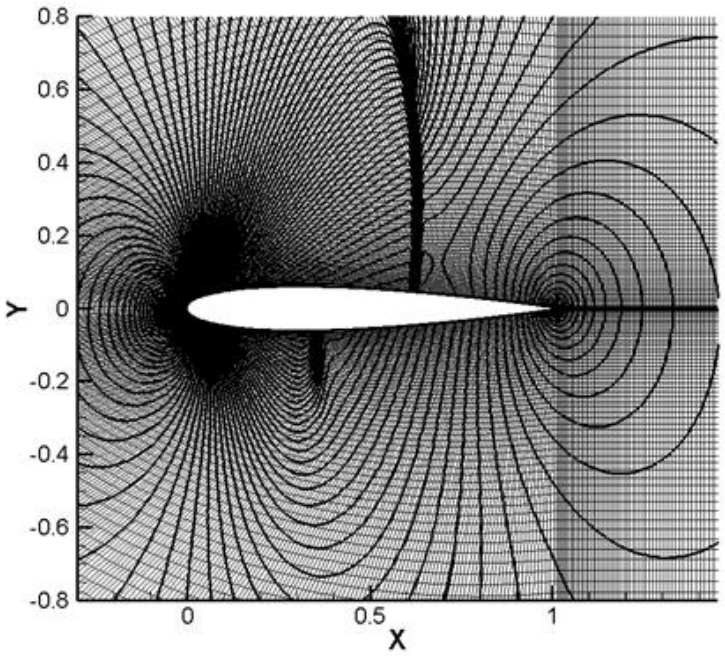


2D, pressure distribution, t=0.32 **body fitted grid** 300x200

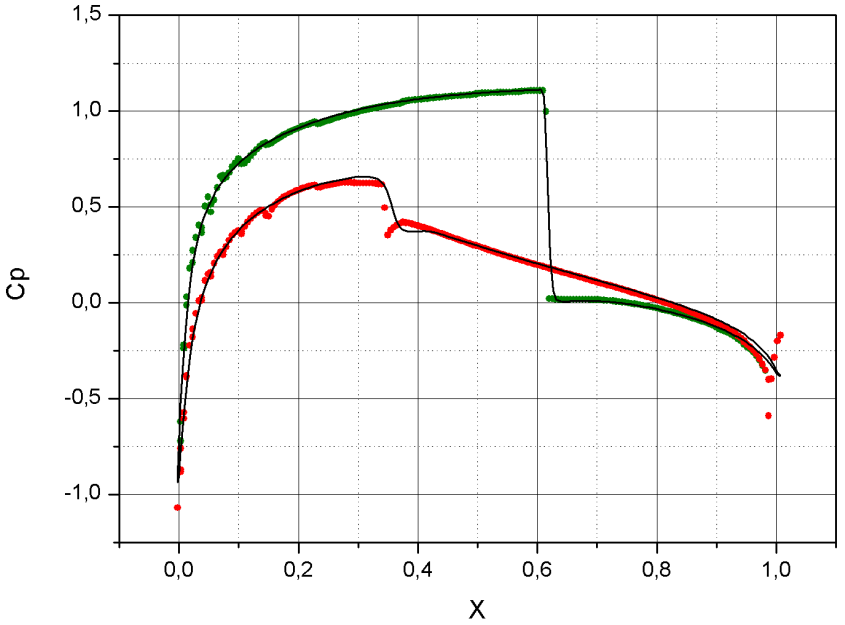
Numerical Results:Transonic

A/D coeffic	Cartesian grid	C-grid body fitt	CFL3D (C-grid)
Cl	0,3012	0,3036	0,3546
Cd	0,02184	0,02199	0,02261

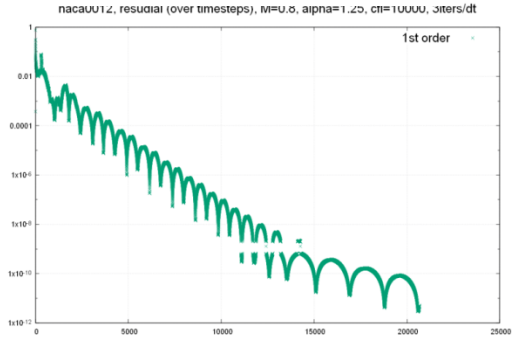




markers - cartesian grid: Cx=0.006868, Cy=0.1388
line - body fitted grid: Cx=0.006885, Cy=0.1362



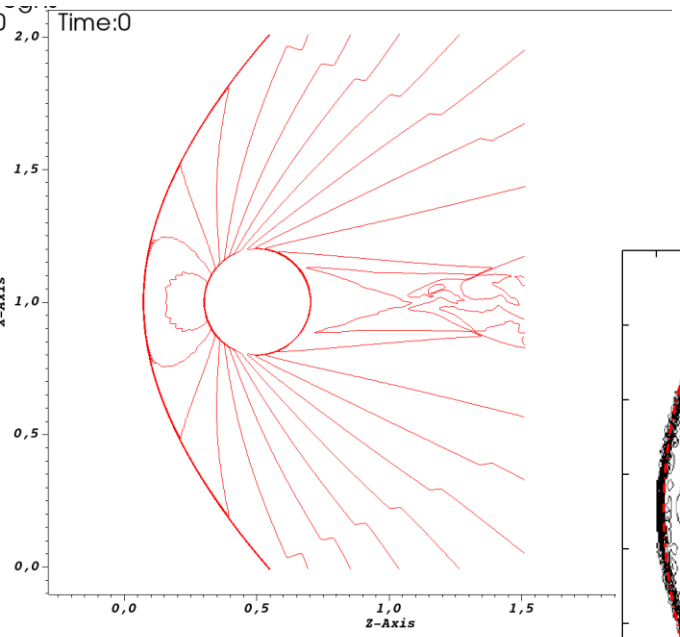
Cartesian grid: 200x25 (inner), 150(upstream), 300(downstream), 150 (up/down), M=0.8, alpha=1.25 grad, 2nd order, dt=0.01(cfl=10)



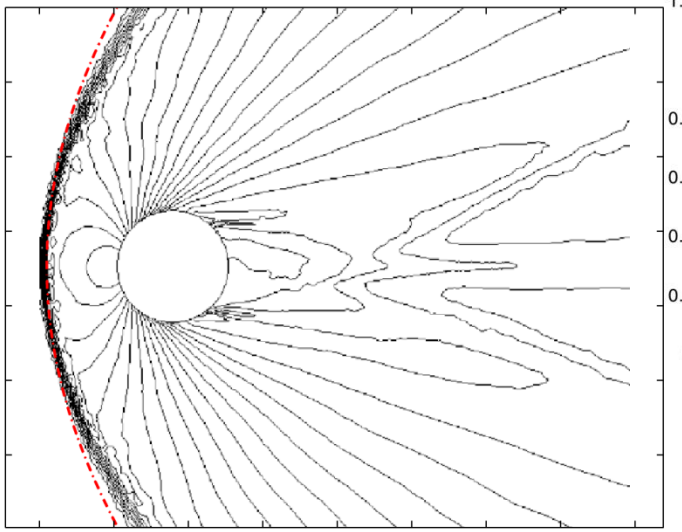
Numerical Results: Steady, Comparison

2D, Обтекание цилиндра, M=2, 1024x1024

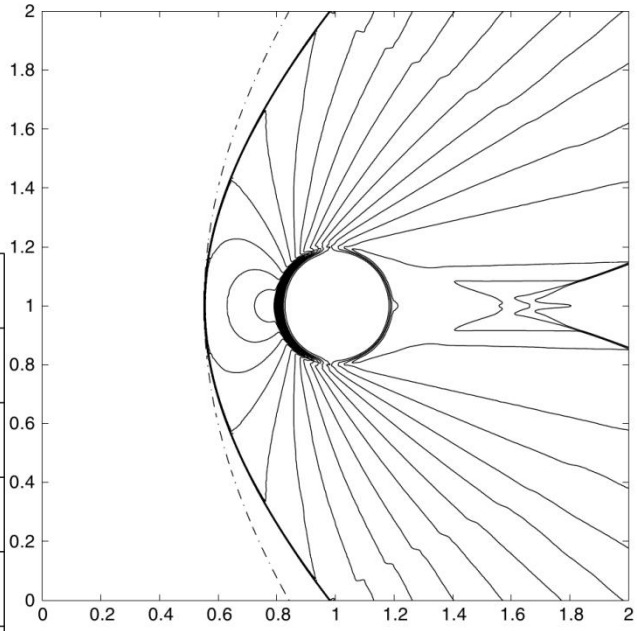
Isodensity lines



Free Boundary Method
(32 GPU)



Fluent (unstructured)

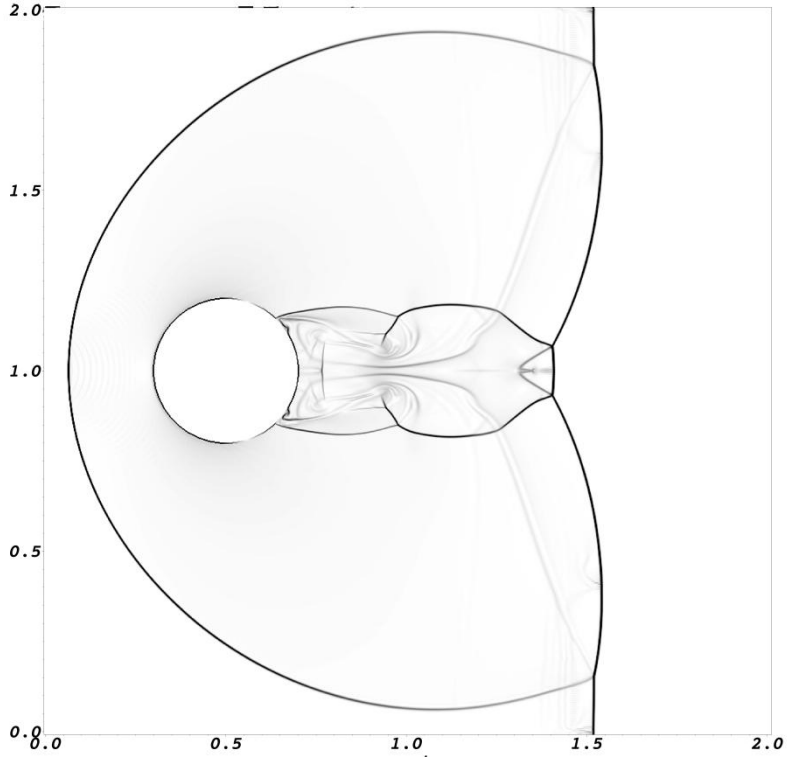


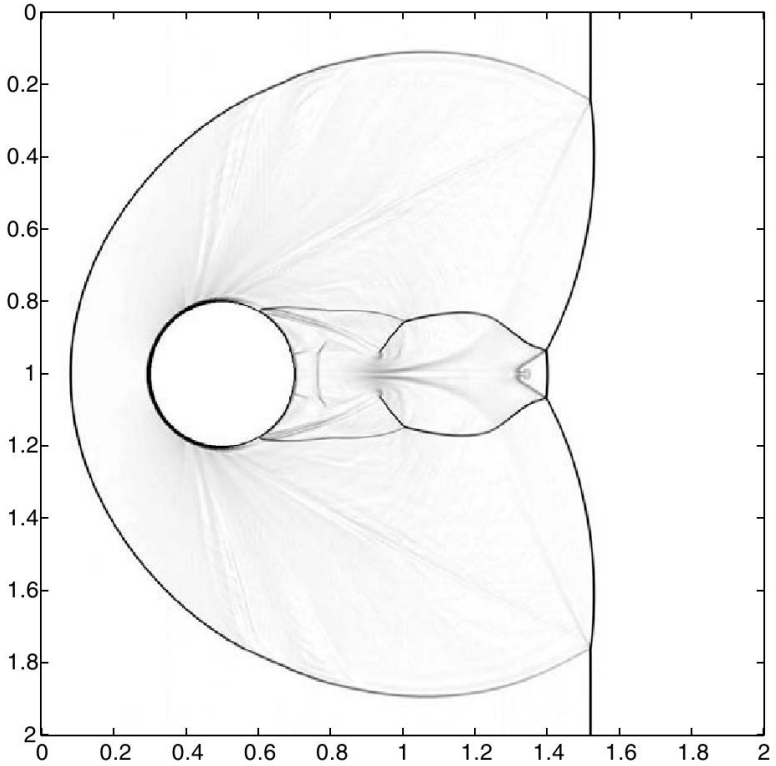
Penalisation Method

Numerical Results: Unsteady, Comparison

2D, shock difraction, M=3, 1024x1024, t=0.4

Numerical Schlieren





Free Boundary Method (32 GPU)

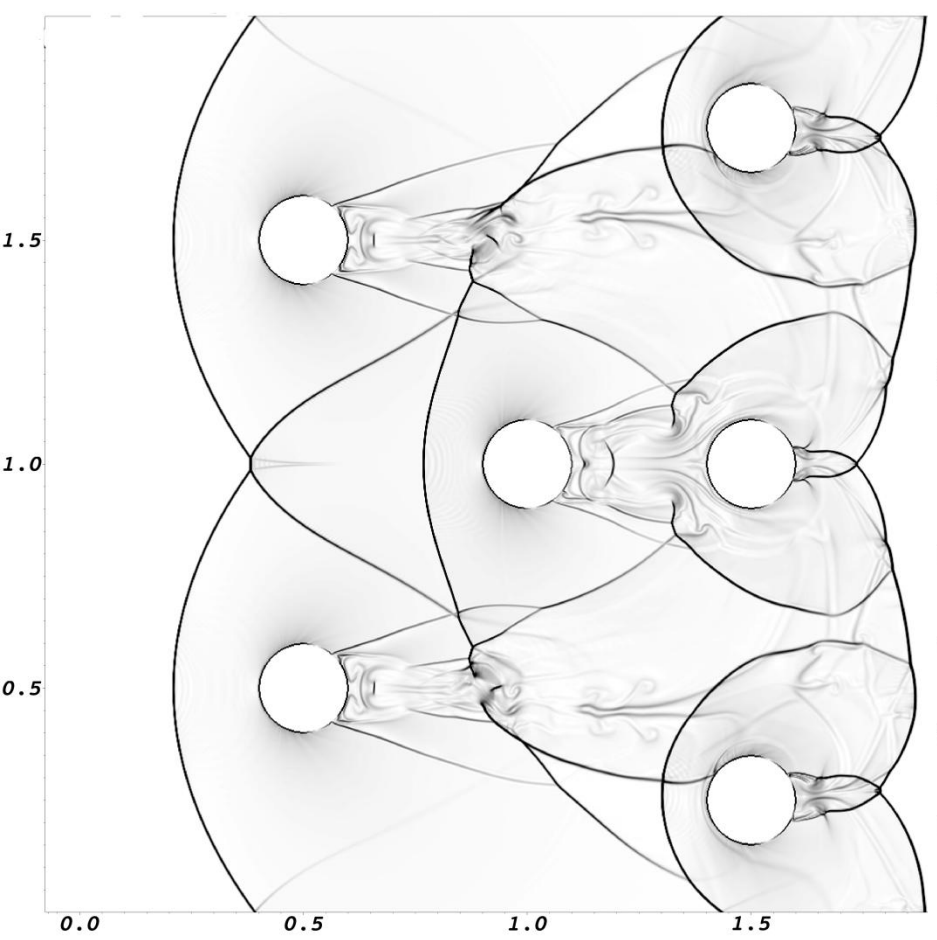
Penalisation Method*

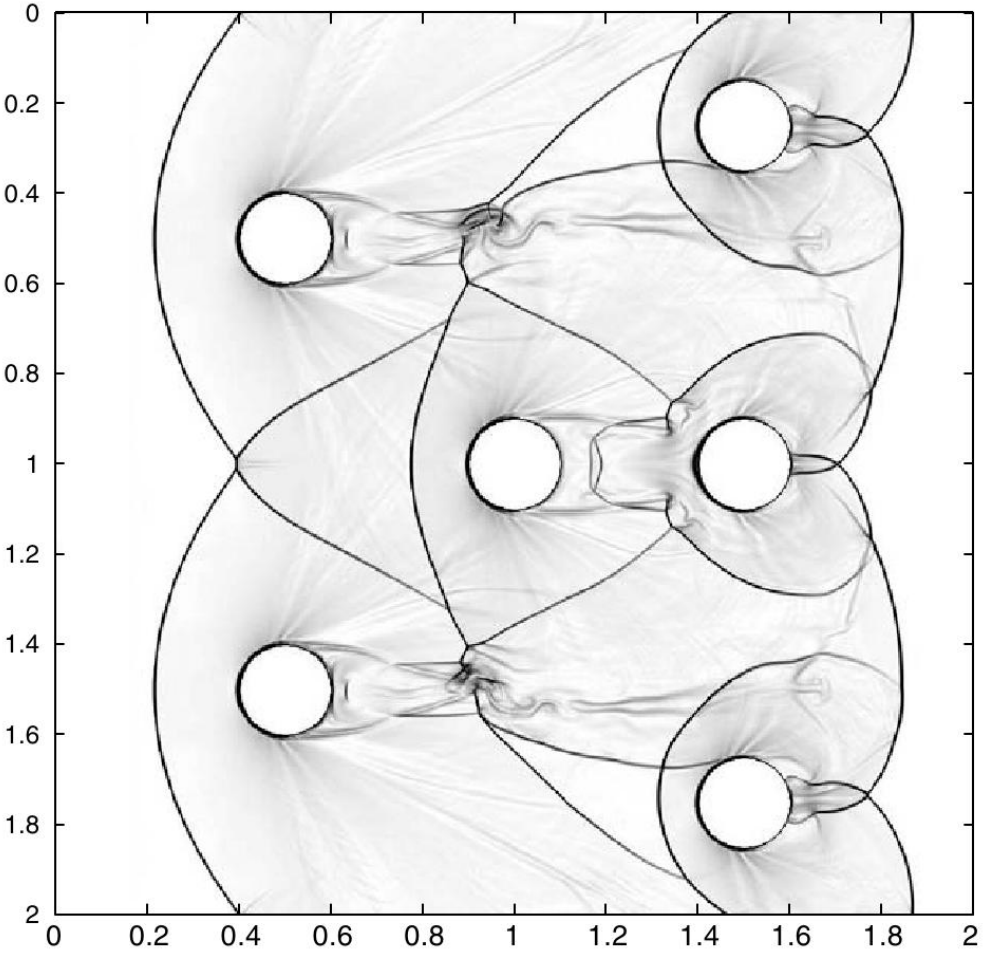
*O. Boirona, G. Chiavassa, R. Donat. *A high-resolution penalization method for large Mach number flows in the presence of obstacles* // Computers & Fluids, N 38, pp 703-714, 2009.

Numerical Results: Unsteady, Comparison

2D, shock difraction, M=3, 1024x1024, t=0.5

Numerical Schlieren





Free Boundary Method (32 GPU)

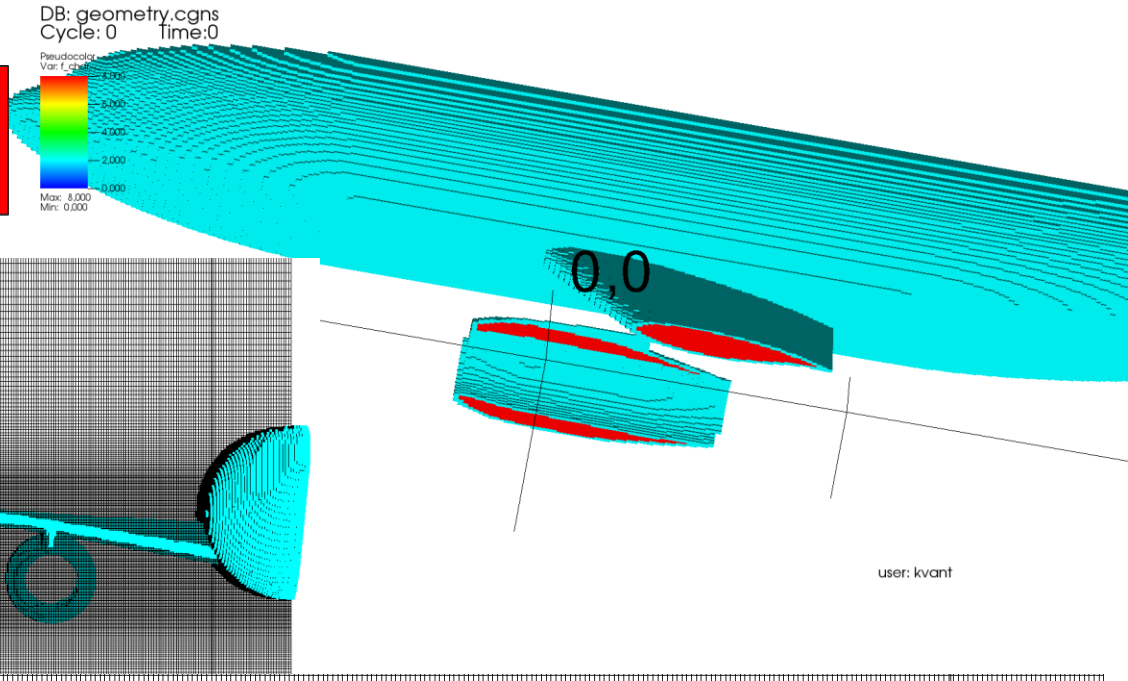
Penalisation Method

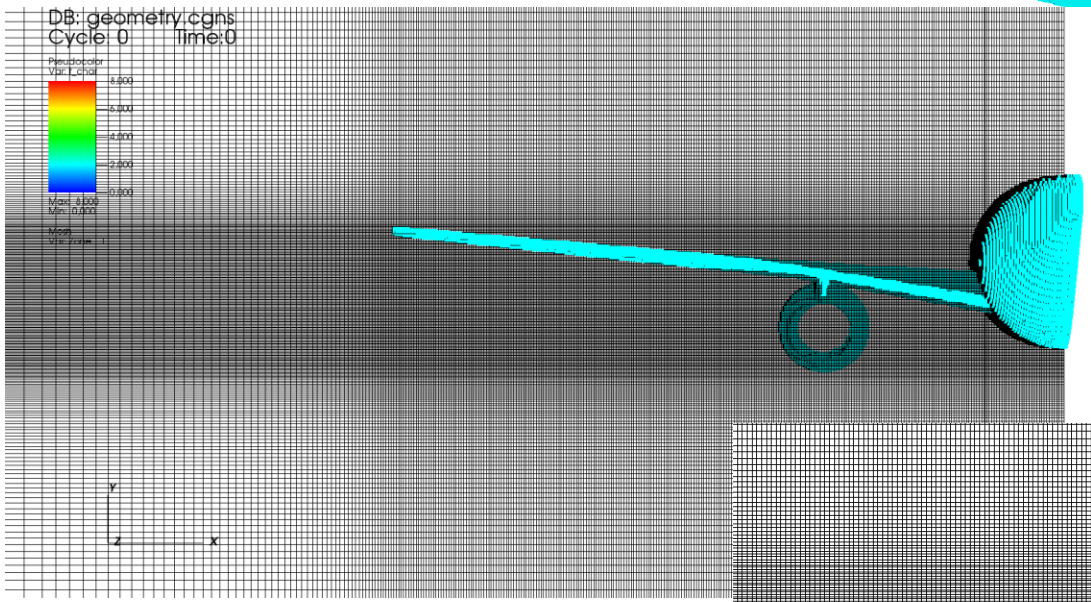
Numerical Results: 3D Cartesian grid

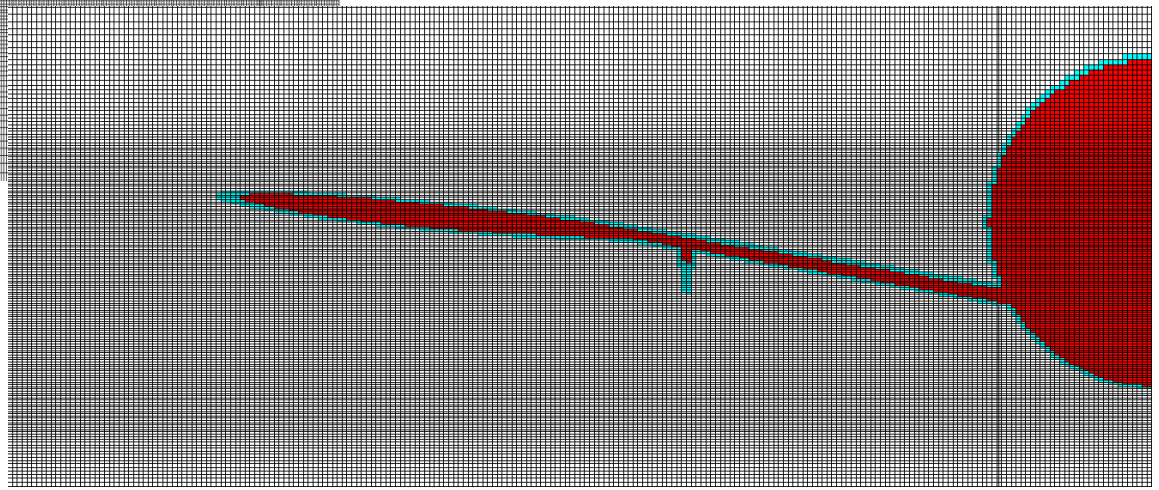
DLR F6

Numerical Results: 3D Cartesian grid

Automatic grid generation and geometry representation



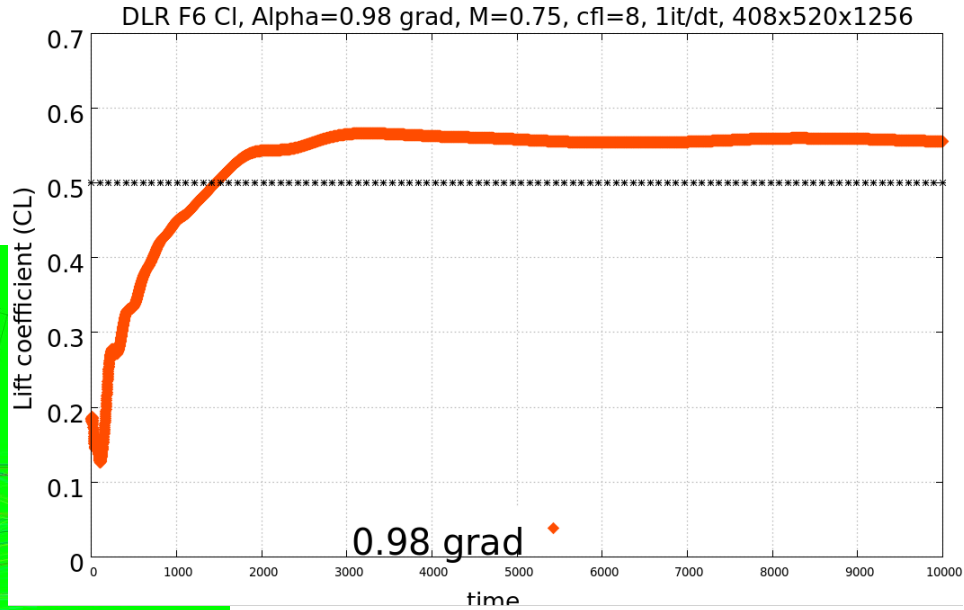


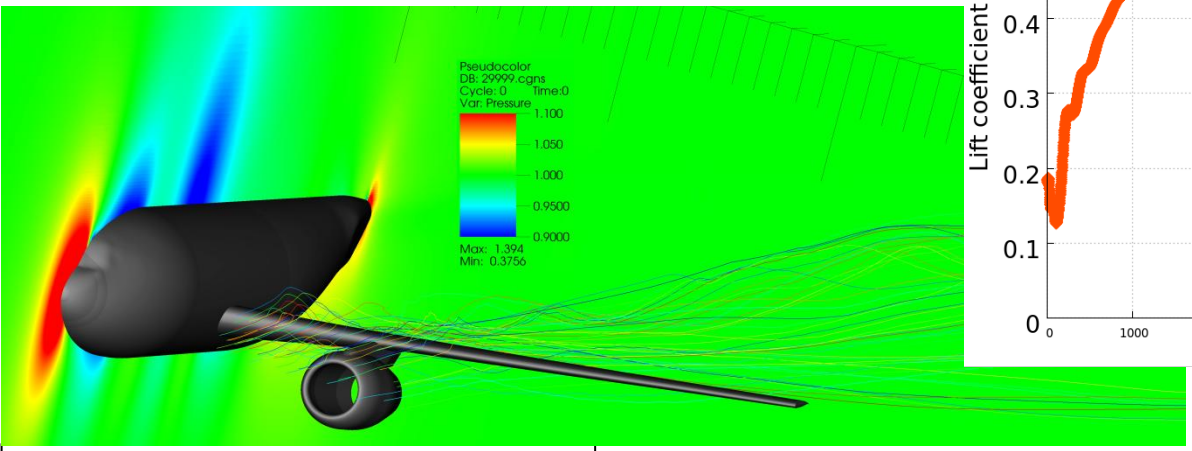


- cut cells
- solid cells

Numerical Results

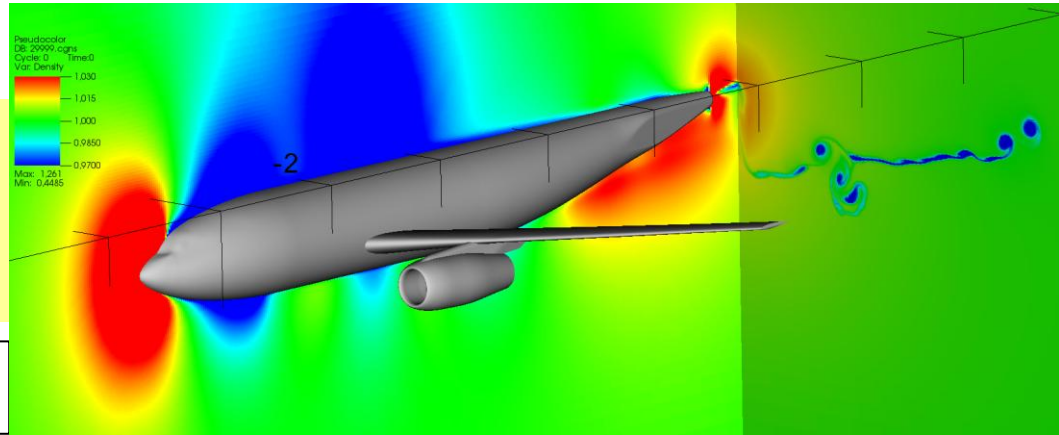
3D, DLR F6, M=0.75,α=0.98°, 260 M comput cells, CFL=8





Pressure & streamlines

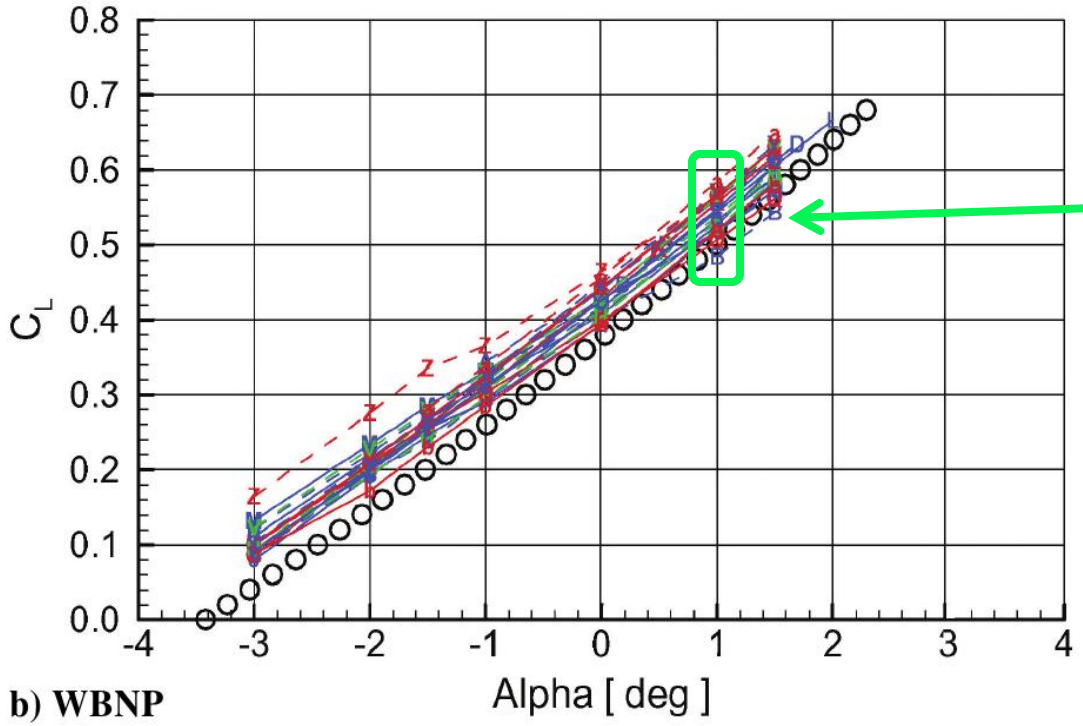
162 GPU, 3 h
SC «Lobachevskii»,
SU Nijnii Novgorod



Density

Numerical Results

3D, DLR F6, M=0.75,α=0.98°, 260 M cells,



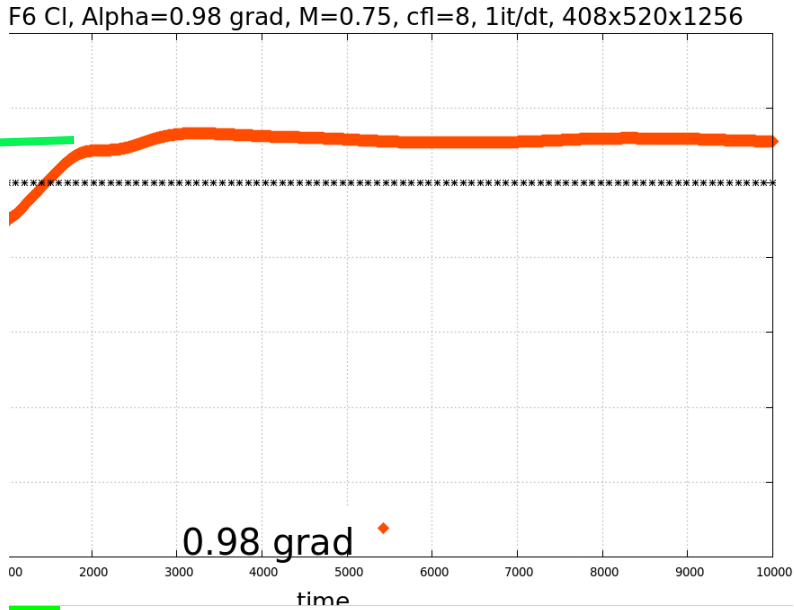
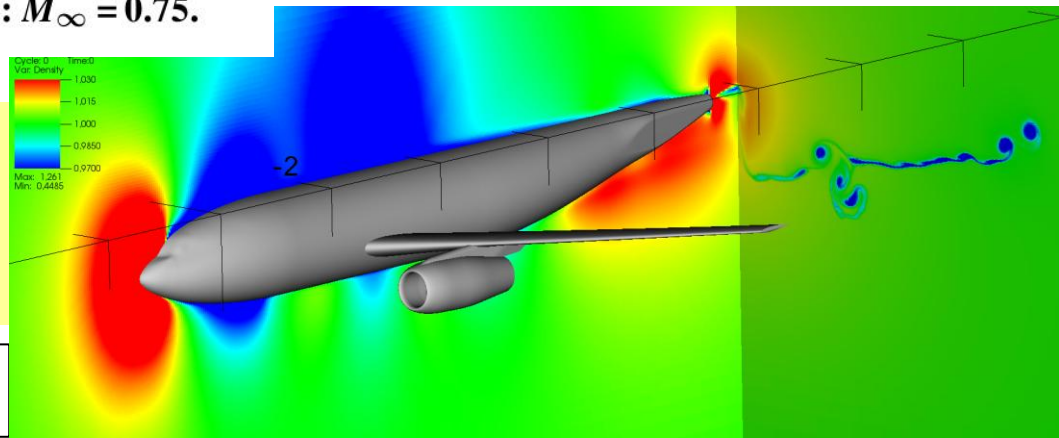


Fig. 10 Composite lift curve results for case 2: $M_\infty = 0.75$.

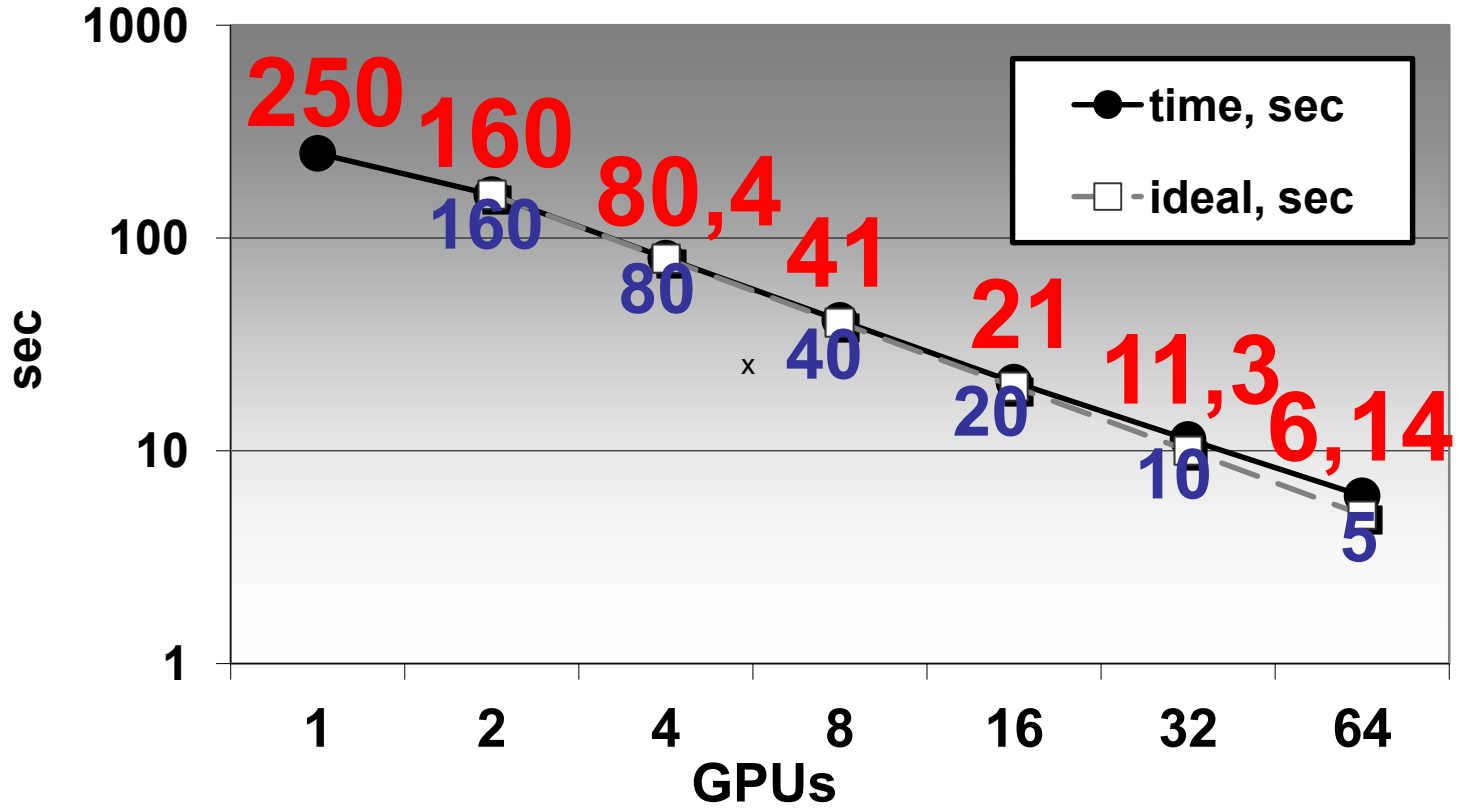
162 GPU, 3 h
SC «Lobachevskii»,
SU Nijnii Novgorod



Density

Numerical Results: 2D Scalability

39 M cells, M=2,

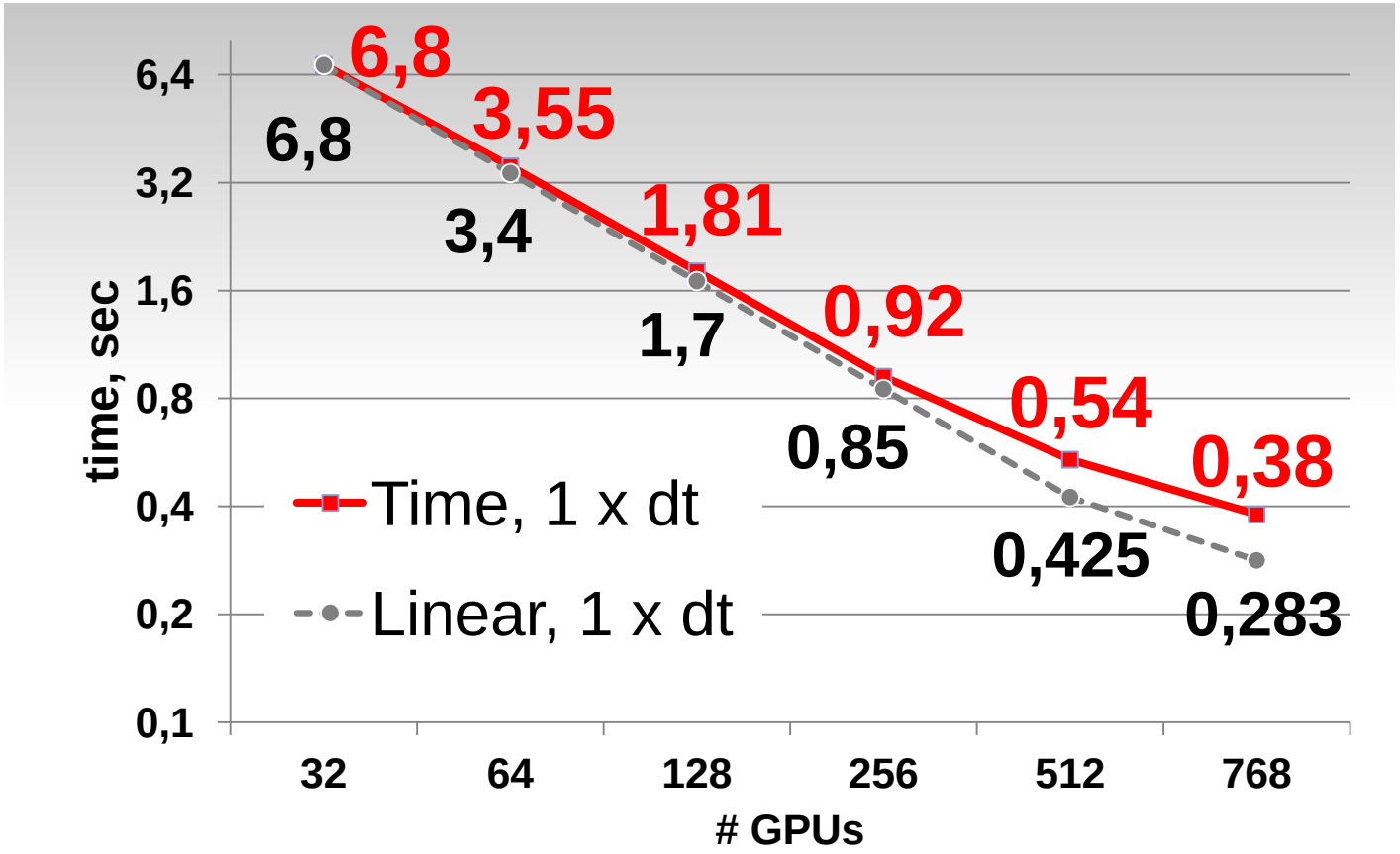


SC K100, KIAM RAS

Effectiveness– **80%** на 64 GPU

Numerical Results: 3D Scalability

150 M cells, (shock-boundary layer interaction)



SC «Lomonosov», SU Moscow

effectiveness – **75%** на **768 GPU**

acceleration 1 GPU / 1 CPU(core) ~ **15-30x**

Summary

- Построен параллельный алгоритм для метода LU-SGS, доказана его корректность и эквивалентность последовательной версии;
- Реализован эффективный программный комплекс на основе параллельной версии LU-SGS с использованием метода свободной границы и декартовых сеток для расчета задач газовой динамики на multi-GPU системах;
- Предварительные результаты показали корректность работы программного комплекса и его хорошую масштабируемость на системах петафлопного уровня;
- Проведено численное моделирование ряда задач аэродинамики;

Future Works

- Учет вязких диссипативных эффектов;
- Решение сопряженных задач газовой динамики и механики твердого тела;
- Оптимизация решателя под современные и будущие архитектуры GPU и других сопроцессоров;
- Real-time визуализация расчетов;

Спасибо за внимание!