Towards an ultra efficient kinetic scheme

J. Narski[1],
G. Dimarco[1,2], R. Loubère[1], T. Rey[3] V. Rispoli[1]

[1]CNRS and Institut de Mathématique de Toulouse (IMT), Toulouse, France

[2]Dipartimento di matematica, Ferrara, Italia.

[3]Laboratoire Paul Painlevé, Université Lille 1, Lille, France.

`http://sites.google.com/site/jaceknarski/`

Shark May ’16

Motivation

Motivations

- Modeling of non equilibrium gas flows (plasma, hypersonic flow)
- Kinetic equations extremely difficult to solve numerically (7 dimensions)

Purposes

- Develop an efficient kinetic scheme
- Simulate 2D$\times$3D on 'normal' machines
- Simple, reusable and evolutive scheme

Bibliography

[1] Towards an ultra efficient kinetic scheme Part I: basics on the BGK equation , G. Dimarco, R. Loubère, Journal of Computational Physics, Volume 255, 2013, pp 680-698.
[2] Towards an ultra efficient kinetic scheme Part II: The High-order case , G. Dimarco, R. Loubère, Journal of Computational Physics, Volume 255, 2013, pp 699-719.
[3] Towards an ultra efficient kinetic scheme Part III: High Performance Computing: OpenMP & MPI, JN, G. Dimarco, R. Loubère, J.Comput.Phys. 284, 22-39, 2015
[4] Towards an ultra efficient kinetic scheme Part IV: Boltzmann equation, G. Dimarco, R. Loubère, JN, in preparation 2016
[5] Towards an ultra efficient kinetic scheme Part V: Massivly Parallel Architecures, G. Dimarco, R. Loubère, JN, in preparation 2016

Kinetic - Fluid models

Boltzmann-BGK description of rarefied gaz dynamics

$$\partial_t f + \boldsymbol{V} \cdot \nabla_{\boldsymbol{X}} f = \frac{1}{\tau}(M_f - f) \quad \boldsymbol{X} \in \Omega \subset \mathbb{R}^3, \boldsymbol{V} \in \mathbb{R}^3, \tag{1}$$

$f = f(\boldsymbol{X}, \boldsymbol{V}, t)$ density of particles, $\tau > 0$ is the relaxation time. BGK= collisions modeled by relaxation towards the local thermodynamical equilibrium defined by the Maxwellian distribution

$$M_f = M_f[\rho, \boldsymbol{U}, T]\,(\boldsymbol{V}) = \frac{\rho}{(2\pi\theta)^{d/2}} \exp\left(\frac{-\|\boldsymbol{U} - \boldsymbol{V}\|^2}{2\theta}\right), \tag{2}$$

where $\rho \in \mathbb{R},\ \rho > 0$ and $\boldsymbol{U} = (u, v, w)^t \in \mathbb{R}^3$ are the density and mean velocity, θ defined as $\theta = RT$ with T the temperature, R gas constant.

Macroscopic moments

Moments ρ, $\boldsymbol{U}$ and T are related to f in 3D by:

$$\rho = \int_{\mathbb{R}^3} f\, d\boldsymbol{V}, \qquad \boldsymbol{U} = \frac{1}{\rho} \int_{\mathbb{R}^3} \boldsymbol{V} f\, d\boldsymbol{V}, \qquad \theta = \frac{1}{3\rho} \int_{\mathbb{R}^3} \|\boldsymbol{V} - \boldsymbol{U}\|^2 f\, d\boldsymbol{V},$$

with total energy: $E = \frac{1}{2} \int_{\mathbb{R}^3} \|\boldsymbol{V}\|^2 f\, d\boldsymbol{V} = \frac{1}{2}\rho\|\boldsymbol{U}\|^2 + \frac{3}{2}\rho\theta,$

Kinetic - Fluid models

The limit of $\tau \to 0$

If number of collisions goes to infinity, then $\tau \to 0$ and $f \to M_f$. One retrieves compressible Euler equations

$$\begin{aligned}
&\frac{\partial \rho}{\partial t} + \nabla_{\boldsymbol{x}} \cdot (\rho \boldsymbol{U}) = 0,\\
&\frac{\partial (\rho \boldsymbol{U})}{\partial t} + \nabla_{\boldsymbol{x}} \cdot (\rho \boldsymbol{U} \otimes \boldsymbol{U} + pI) = 0,\\
&\frac{\partial E}{\partial t} + \nabla_{\boldsymbol{x}} \cdot ((E + p)\boldsymbol{U}) = 0,\\
&p = \rho\theta, \quad E = \frac{3}{2}\rho\theta + \frac{1}{2}\rho\|\boldsymbol{U}\|^2,
\end{aligned}$$

where I is the identity and p the pressure given by a perfect equation of state with gas constant $\gamma = 5/3$ in 3D. This is the fluid/macroscopic model.

Fast Kinetic Scheme

DVM

Semi-Lagrangian scheme for Discrete Velocity Model (DVM) approximation of Boltzmann-BGK equation.

DVM

Let $\mathcal{K}$ be a bounded set of N_v^3 multi-indices of $\mathbb{N}^3$. Let $\mathcal{V}$ be a Cartesian grid given by

$$\mathcal{V} = \{ \boldsymbol{V}_k = k\Delta v + \boldsymbol{W},\ k \in \mathcal{K} \} ,$$

where Δv is the grid step in the velocity space. The generic cell in the velocity space is $\omega_{k+1/2} = [\boldsymbol{V}_k;\ \boldsymbol{V}_{k+1}]$. We denote the discrete collision invariants on $\mathcal{V}$ by

$$m_k = \left(1,\, \boldsymbol{V}_k,\, \frac{1}{2}\|\boldsymbol{V}_k\|^2\right)^t$$

The continuous distribution function f is replaced by a vector

$$f_{\mathcal{K}}(\boldsymbol{X}, t) = (f_k(\boldsymbol{X}, t))_k\,, \qquad f_k(\boldsymbol{X}, t) \approx f(\boldsymbol{X}, \boldsymbol{V}_k, t).$$

Fluid quantities:

$$F(\boldsymbol{X}, t) = \sum_{k \in \mathcal{K}} m_k\, f_k(\boldsymbol{X}, t)\, \Delta v.$$

Fast Kinetic Scheme

Equations

Set of N_v^3 evolution equations

$$\partial_t f_k + \boldsymbol{V}_k \cdot \nabla_{\boldsymbol{X}} f_k = \frac{1}{\tau}(\mathcal{E}_k[F] - f_k) \tag{1}$$

Space and time discretization

Cartesian uniform grid $\mathcal{X} = \{\boldsymbol{X}_j = j\Delta x + \boldsymbol{Y},\ j \in \mathcal{J}\}$, Δx is the grid step, $\boldsymbol{Y}$ is a vector in $\mathbb{R}^3$ and $\mathcal{J}$ is a subset of $\mathbb{N}^3$.
$t^{n+1} = t^n + \Delta t$, time step Δt defined by a CFL condition.

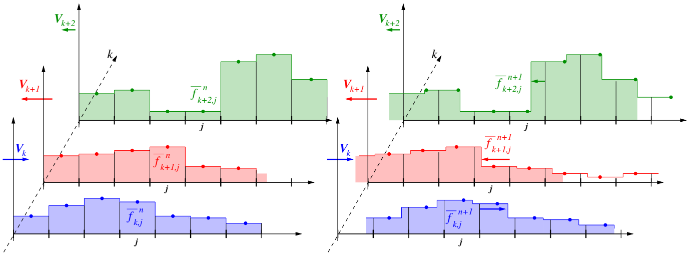
Splitting

Each equation (1) is solved by a time splitting. Transport stage solves LHS, relxation stage solves RHS using solution from transport stage

$$\begin{aligned} \textit{Transport stage} \quad &\longrightarrow \quad \partial_t f_k + \boldsymbol{V}_k \cdot \nabla_{\boldsymbol{X}} f_k = 0, \\ \textit{Relaxation stage} \quad &\longrightarrow \quad \partial_t f_k = \frac{1}{\tau}(\mathcal{E}_k[F] - f_k). \end{aligned}$$

Fast Kinetic Scheme

Transport stage



V_{k+2}
k
V_{k+1}
$\overline{f}^{\,n}_{k+2,j}$
j
V_k
$\overline{f}^{\,n}_{k+1,j}$
j
$\overline{f}^{\,n}_{k,j}$
j
V_{k+2}
k
V_{k+1}
$\overline{f}^{\,n+1}_{k+2,j}$
j
V_k
$\overline{f}^{\,n+1}_{k+1,j}$
j
$\overline{f}^{\,n+1}_{k,j}$
j

Let $f^n_{j,k}$ be the pointwise approximation at discrete time t^n of the distribution f: $f^n_{j,k} = f(\boldsymbol{X}_j, \boldsymbol{V}_k, t^n)$ and $\mathcal{E}^n_{j,k}[F]$ be the equilibrium distribution approximation of $M^n_{j,k} = M_f(\boldsymbol{X}_j, \boldsymbol{V}_k, t^n)$ defined at any point $\boldsymbol{X}_j$ of space at discrete time $t = t^n$.
Let $\overline{f}^n_k$ be a piecewise constant function associated with velocity $\boldsymbol{V}_k$ at time t^n defined at each space cell by

$$\overline{f}^n_{k,j} = \frac{1}{|\Omega_j|} \int_{\Omega_j} f(\boldsymbol{X}, \boldsymbol{V}_k, t^n)\, d\boldsymbol{X}$$

Exact transport during Δt:

$$\overline{f}^{*,n+1}_k = \overline{f}^n_k(\boldsymbol{X} - \boldsymbol{V}_k \Delta t), \qquad \forall \boldsymbol{X} \in \Omega$$

Fast Kinetic Scheme

Relaxation stage

Relaxation step

$$\partial_t f_{j,k} = \frac{1}{\tau}(\mathcal{E}_{j,k}[F] - f_{j,k})$$

Initial data is given by the result of the transport step $f^{*,n+1}_{j,k} = \overline{f}^{*,n+1}_k(\boldsymbol{X}_j)$.
Maxwellian computed using macroscopic quantities

$$F^{n+1}_j = F^{*,n+1}_j = \sum_{k\in\mathcal{K}} m_k\, f^{*,n+1}_{j,k}\, \Delta v$$

Preservation of macroscopic quantities: moments before ($F^{*,n+1}_j$) and after (F^{n+1}_j) unchanged. Then

$$f^{n+1}_{j,k} = \exp(-\Delta t/\tau) f^{*,n+1}_{j,k} + (1-\exp(-\Delta t/\tau))\mathcal{E}_k[F^{n+1}_j],$$

New value of f^{n+1} only in the cell centers, we need f^{n+1} in whole domain for the transport step.
Define $\mathcal{E}_k$ as the equilibrium function with the discontinuities located in the same positions as f_k

$$\overline{\mathcal{E}}^{n+1}_k(\boldsymbol{X})[F] = \mathcal{E}^{n+1}_{j,k}[F], \quad \forall \boldsymbol{X} \text{ such that } \overline{f}^{*,n+1}_k(\boldsymbol{X}) = \overline{f}^{*,n+1}_k(\boldsymbol{X}_j)$$

Finally

$$\overline{f}^{n+1}_k(\boldsymbol{X}) = \overline{f}_k(\boldsymbol{X}, t^n + \Delta t) = \exp(-\Delta t/\tau)\overline{f}^*_k(\boldsymbol{X}) + (1-\exp(-\Delta t/\tau))\overline{\mathcal{E}}^{n+1}_k(\boldsymbol{X})[F]$$

Fast Kinetic Scheme

Conservation of macroscopic quantities

Constrained optimization formulation ($d_x = 3$)

- let $\widehat{f} = \left(\widehat{f}_1, \widehat{f}_2, \dots, \widehat{f}_N\right)^t$ be the pointwise distribution vector and $f = (f_1, f_2, \dots, f_N)^t$ be the unknowns which fulfill the conservation of moments
- $C_{(d_x+2)\times N} = \left((\Delta v)^3, \boldsymbol{V}_k(\Delta v)^3, \|\boldsymbol{V}_k\|^2(\Delta v)^3\right)^t$ a matrix constant in time
- $F_{(d_x+2)\times 1} = (\rho\ \rho\boldsymbol{U}\ E)^t$ be the vector of the conserved quantities.

Conservation can be imposed solving[a]:

$$\text{Given } \widehat{f} \in \mathbb{R}^N,\ C \in \mathbb{R}^{(d_x+2)\times N},\ \text{and } F \in \mathbb{R}^{(d_x+2)\times 1},$$
$$\text{find } f \in \mathbb{R}^N \text{ such that } \quad \|\widehat{f} - f\|_2^2 \text{ is min under constraints } Cf = F.$$

Using a Lagrange multiplier $\lambda \in \mathbb{R}^{d_x+2}$, the objective function to be optimized is $L(f,\lambda) = \sum_{k=1}^N |\widehat{f}_k - f_k|^2 + \lambda^T(Cf - F)$. Exactly solved into

$$f = \widehat{f} + C^T(CC^T)^{-1}(F - C\widehat{f}).$$

Also done for the equilibrium distribution $M_f[F] = M_f(\boldsymbol{X}_j, \boldsymbol{V}_k, t_n)[F]$

$$\mathcal{E}[F] = M_f[F] + C^T(CC^T)^{-1}(F - CM_f[F]),$$

[a] Gamba et al JCP 228 2009

HOFKS - high order extension

Second order in time

Time splitting with Strang splitting strategy. CFL:

$$\Delta t \max_K \frac{||\boldsymbol{V}_k||}{L_c} \leq 1$$

- performs well in collisionless regimes
- scheme stable for $\Delta t > CFL$
- projection over the equilibrium of first order $\rightarrow$ loss of accuracy close to fluid regime

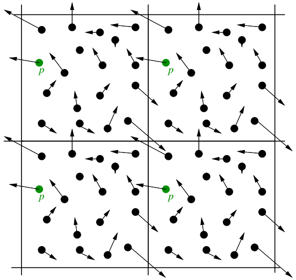
$$\bar{f}_k^{n+1}(\boldsymbol{X}) = \exp(-\Delta t/\tau)\bar{f}_k^{*}(\boldsymbol{X}) + (1 - \exp(-\Delta t/\tau))\bar{\mathcal{E}}_k^{n+1}(\boldsymbol{X})[F]$$

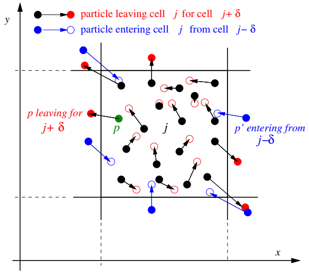
Solve the equilibrium part with of the distribution function with a macroscopic scheme instead of kinetic.

Moments from the transport stage are replaced by a solution of Euler equations. We use MUSCL scheme. Stability condition:

$$\Delta t < \frac{1}{2}\frac{\Delta x}{\alpha_{\max}}$$

Efficient implementation




y
x
particle leaving cell j for cell j+ δ
particle entering cell j from cell j− δ
p leaving for j+ δ
p
j
p' entering from j−δ

Particle implementation: Initially N_v^3 particles are positioned at the cell center

$$\boldsymbol{X}_p^0 = (\Delta x/2, \Delta y/2, \Delta z/2)^t$$

each particle has a unique constant velocity $\boldsymbol{V}_p$ from the velocity space, $p = 1, \cdots, N_v^3$. The transport of these particles during Δt follows

$$\widetilde{\boldsymbol{X}}_p^{n+1} = \boldsymbol{X}_p^n + \Delta t\, \boldsymbol{V}_p.$$

Same set of particles in every space cell, only positions and velocities of particles in generic cell kept in memory. Memory consumption reduced by 85%.

Particle mass

Each particle p in cell j carries its “mass” which is updated defined at t^n thanks to the previous mass $m_{j,p}^{n-1}$ and updated moments ρ_j^n, $\boldsymbol{U}_j^n$, θ_j^n as

$$m_{j,p}^n = \exp(-\Delta t/\tau) m_{j,p}^{n-1} + (1 - \exp(-\Delta t/\tau)) M_f[\rho_j^n, \boldsymbol{U}_j^n, \theta_j^n](\boldsymbol{V}_p)$$

Because the fluid quantities are obtained through discrete summations on particles in cell j:

$$F_j^n = \sum_{p=1}^{N_v^3} m_{j,p}^n \, \Delta v$$

the updated fluid quantities are therefore obtained after the transport step following

$$F_j^{n+1} = F_j^n - \underbrace{\sum_{p,\ \tilde{\boldsymbol{x}}_p^{n+1} \notin \Omega_j} m_{j,p}^n \, \Delta v}_{\text{Leaving particles}} + \underbrace{\sum_{p',\ \tilde{\boldsymbol{x}}_{p'}^{n+1} \in \Omega_j} m_{j-\delta,p'}^n \, \Delta v}_{\text{Entering particles}} .$$

Recall that these conservative cell centered fluid quantities are constituted of mass, momentum and total energy whereas primitive ones are density, velocity and temperature. Then a mapper from conservative $F_j^{n+1} = (\mathcal{F}_1, \boldsymbol{\mathcal{F}_2}, \mathcal{F}_3)_j^{n+1}$ to primitive $(\rho, \boldsymbol{U}, T)_j^{n+1}$ variables is defined as

$$\begin{aligned}
\rho_j^{n+1} &= \mathcal{F}_1^{n+1}, \\
\boldsymbol{U}_j^{n+1} &= \boldsymbol{\mathcal{F}_2^{n+1}} / \mathcal{F}_1^{n+1}, \\
\theta_j^{n+1} &= \frac{2}{3} \left(\mathcal{F}_3^{n+1} - \frac{\|\boldsymbol{\mathcal{F}_2^{n+1}}\|^2}{2\mathcal{F}_1^{n+1}} \right) / \mathcal{F}_1^{n+1}.
\end{aligned}$$

Generic algorithm

1. *Relaxation step.* Compute masses of N_v^3 particles, store them in an array of the size $N_v^3 \times N^3$
2. *Transport of particles.* Displace N_v^3 particles, produce a list of N_{out} particles escaping the generic cell and store the δ determining the destination and provenance of associated sister particles.
3. *Update conservative variables* F_j^{n+1}
4. *Update primitive variables*

FKS under HPC

OpenMP algorithm

1. *Relaxation step.* Compute in parallel masses of N_v^3 particles with, parallelization is performed on the external loop over N_v^3 particles.
2. *Transport of particles.* Move in parallel N_v^3 particles
3. *Update conservative variables.* Test in a parallel loop over N_v^3 particles if a particle has escaped from the generic cell. If so, add a contribution to F_j^{n+1} for every space cell. Update the particle position and exchange particle mass with the associated sister particle.
4. *Update primitive variables*

GPU algorithm – Easily extendable to multi GPU architectures

1. *Copy from CPU to GPU*. Copy to the GPU memory all primitive and conservative variables.
2. *Loop over particles*
    1. *Relaxation step* Compute relaxed masses of particles for every space cell using CUDA. Store the result on GPU.
    2. *Transport step* Move every particle and test if it has escaped the generic cell. If so, store the provenance of the sister cell.
    3. *Update conservative variables*. If the particle has escaped the generic cell, add contribution to conservative variables and assign mass to he one of the incoming sister particle.
    4. *Copy from GPU to CPU.* Copy the resulting mass array from the GPU memory to the CPU memory.
3. *Update primitive variables* in parallel on GPU.
4. *Copy from GPU to CPU*. Write to the CPU memory the updated conservative variables.

Machines

HOFKS and HOFKS-OMP

Serial version implemented in C++ compiled with gcc 4.7.2 and -Ofast optimization flag

Computational server with 4 Intel(R) Xeon(TM) E5-4650 processors running at 2.7 GHz (giving a total of 32 physical cores and 64 logical) with 512GB of RAM running under Debian Wheezy

HOFKS-GPU

GPU version implemented in CUDA 5.5 and gcc 4.7.2 and -Ofast optimization flag

Computational server with dual Intel(R) Xeon(TM) E5-2650 processor running at 2.0 GHz (16 physical and 32 virtual cores) with 128GB and 2 Nvidia GTX 780 units (3GB of memory, 2304 CUDA cores at 900MHz each) running under Debian Wheezy

Decent card for gaming, not designed for professional applications (lack of memory error correction, double precision, worse copy engine than Tesla/Quadro)

Machines

CUDA architecture

- Massively parallel : 12 multiprocessors consisting of 192 CUDA cores
- Functions executed on GPU (kernels) are executed in warps involving 32 threads
- Parallelization strategy : replace every loop over space cells by a call to suitable CUDA kernel
- Slow CPU $\leftrightarrow$ GPU memory transfer (8Gb/s at most)
- Example: $200^3 \times 15^3$ particles equals to 100Gb of data (mass vector) — 25s lost on transfer from and to GPU
- Sometimes better to recompute some values than to copy them from CPU memory
- Not really possible in this case
- Possibility to use two concurent copy engine (Tesla/Quadro) : mass array of 1 particle is copied to GPU, second particle is processed by GPU and the updated masses of third particle are transfered back to CPU at the same time $\Rightarrow$ time lost on transfered reduced to 0.

SOD

Parallelization test only

- $\Omega = [0,2]^3$, ball centered at $(1,1,1)$, radius $r = 0.2$, number of space cells $N^3 = 25, 50, 100, 200$, velocity space $[-15,15]^3$ discretized with $N_V = 15^3$ points
- The relaxation parameter $\tau = 10^{-4}$
- Δt fixed at maximal time step

Convergence tests in

- G. DIMARCO, R. LOUBÈRE, *Towards an ultra efficient kinetic scheme. Part I: basics on the BGK equation*, J. Comput. Phys., Vol. 255, 2013, pp 680-698.
- G. DIMARCO, R. LOUBÈRE, *Towards an ultra efficient kinetic scheme. Part II: the high order case*, J. Comput. Phys., Vol. 255, 2013, pp 699-719.

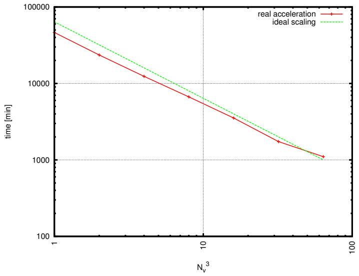
SOD

Sequential, OpenMP and GPUversions compared in terms of CPU time

Cell # $N_c \times N_v^3$	**Cycle** N_{cycle}	**Time** T (s)	T/**cycle** T_{cycle} (s)	T/**cell** T_{cell} (s)	**Mem** (GB)
$25^3 \times 15^3 = 52.7 \times 10^6$	13	204s (3.5mn) 6.77s 6.1s	15.69 0.52 0.47	1×10^{-3} 33×10^{-6} 30×10^{-6}	0.23
$50^3 \times 15^3 = 421.9 \times 10^6$	25	3244s (54mn) 86.6s (1.43mn) 46s	129.76 3.46 1.84	1×10^{-3} 27.7×10^{-6} 14.7×10^{-6}	1.6
$100^3 \times 15^3 = 3.4 \times 10^9$	50	46408s (13h) 1102s (18.4mn) 486s (8.1mn)	928 22.04 9.7	0.9×10^{-3} 22.04×10^{-6} 9.7×10^{-6}	12
$200^3 \times 15^3 = 27 \times 10^9$	98	784×10^3s (9d) 17036s (4.73h) 9353s (2.6h)	8000 174 95	1×10^{-3} 21.7×10^{-6} 12×10^{-6}	101

OpenMP scalability

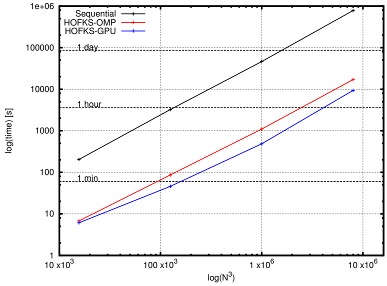
# of cores	**Time** T (s)	**Time/cycle** T_{cycle} (s)	**Time/cell** T_{cell} (s)	**Speed up**
1	46408	928	928×10^{-6}	1
2	23573	471	471×10^{-6}	1.96
4	12395	248	248×10^{-6}	3.74
8	6674	133.5	133.5×10^{-6}	6.95
16	3536	70.7	70.7×10^{-6}	13.12
32	1735	34.7	34.7×10^{-6}	26.74
64	1102	22.04	22.04×10^{-6}	42.11

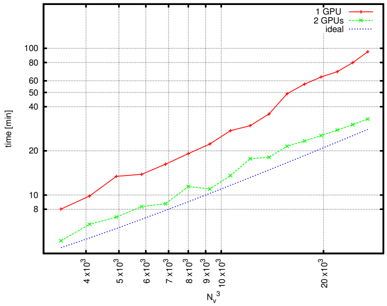


- Smaller speed-up when no. of threads exceeds no. of physical cores
- FKS seems “embarrassingly parallel”

GPU scalability

SOD test again, $N = 100^3$, N_v varies from 15^3 to 30^3

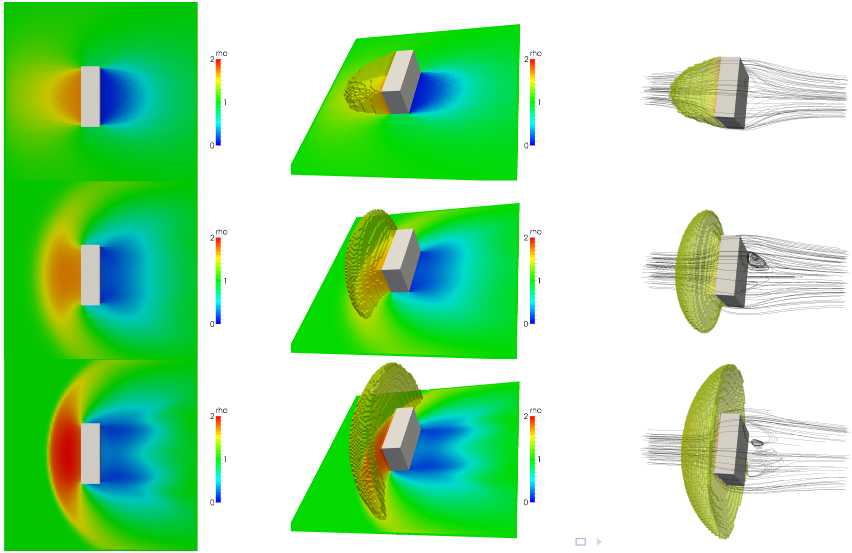




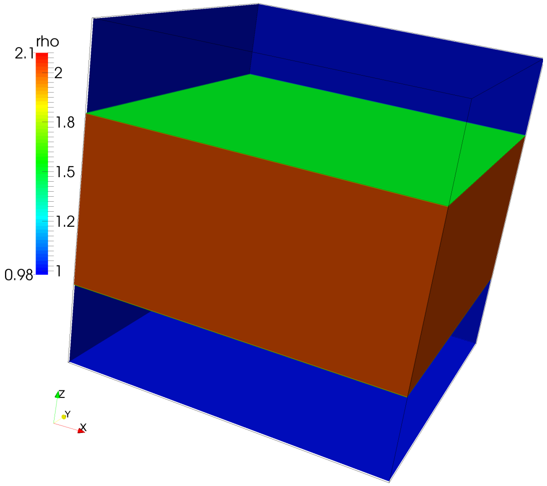
- Computational time grows linearly with number of particles
- 2 GPUs almost twice as fast as single GPU

3D reentry test case

Top-bottom $\tau = 3.10^{-1}, 3.10^{-2}, 3.10^{-4}$, density, streamlines



Kelvin-Helmholtz instabilities



3D instability

Shear flow along horizontal plane.
$N_v = 15$, $N_c = 100^3$, $\tau = 10^{-4}$
Density plot (hidden cells are such that $\rho < 1.2$)
14000 time steps
Computational time: 11.5h on 2 GPUs, equivalent to 77 days on serial machine

High-Order Fast Kinetic Scheme (HOFKS) under HPC

Profiling 3D Sod shock tube for $\tau = 10^{-4}$

Cost of major subroutines - OpenMP FKS code (1 or 16 threads) - GPU FKS code (1, 2 cards)

	OMP code		GPU code	
	1 thread $25^3 \times 15^3$	16 threads $25^3 \times 15^3$	1 GPU $100^3 \times 15^3$	2 GPU $100^3 \times 15^3$
Subroutines	**CPU (s)** (%)	**CPU (s)** (%)	**CPU (s)** (%)	**CPU (s)** (%)
Transport	<0.01 (0%)	<0.001 (0%)	0.0043 (0%)	0.0035 (0%)
EulerHO	7.08 (3%)	1.39 (6%)	0.8844 (0.2%)	1.14 (0.4%)
Relaxation	203.58 (90%)	19.11 (89%)	471.76 (97%)	277.00 (95%)
Primitive	<0.01 (0%)	<0.001 (0%)		
Initial./Comm.	14.33 (6%)	0.99 (5%)	15 (3%)	15 (5%)
Total	225 (100%)	21.5 (100%)	487.6 (100%)	293.1 (100%)

GPU mesh convergence $N^3 \times 15^3$, $N = 25, 50, 100$.

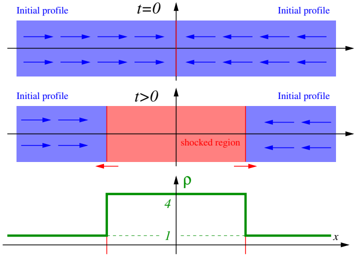
	1 GPU			2 GPUs		
	$N = 25$	$N = 50$	$N = 100$	$N = 25$	$N = 50$	$N = 100$
Relaxation+Prim.	60%	93%	97%	45%	83%	95%

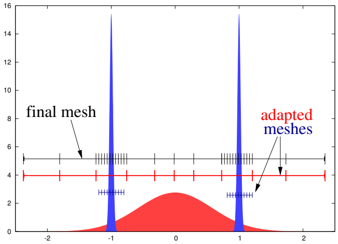
The relaxation stage always consumes a large amount of ressources

Velocity Adaptive Mesh Refinement (AMR)

Discrete Velocity Model issues

Fixed velocity space bounds and constant velocity mesh spacing $\Delta v \Rightarrow$ inaccuracy





Example: explosion like problem in 1D, mesh adaptation is mandatory!
Hyp. 10 cells per Maxwellian, $\tau = 10^{-a}$, $a = 3, 4, 5, 6$ leads to $\sim 200, 700, 2200, 7000$ uniform cells. Only $\leq$ 20 affordable in 3D

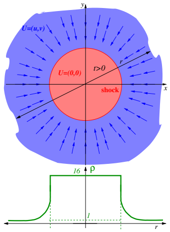
Strategy for velocity AMR for FKS - Choice: only one velocity mesh! [**]

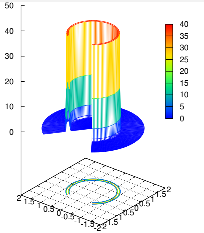
- Adapt the velocity mesh to given data - Start with valid velocity mesh
- Define a frequency of regridding - When should the grid be reshaped?
- Design some sort of remapping technique - Transfer data from old to new velocity mesh

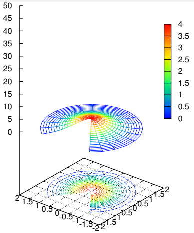
[*] Towards... Part IV: Adaptive velocity Mesh Refinement, G. Dimarco, RL, J.Narski, V.Rispoli, in preparation 2015.
[**] C. Baranger, J. Claudel, N. Hérouard, L. Mieussens, Locally refined discrete velocity grids for stationary rarefied flow simulations, J. Comput. Phys., 257(15), 572-593 (2014) .

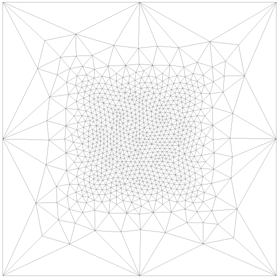
Velocity Adaptive Mesh Refinement (AMR)

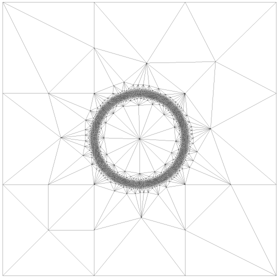
Preliminary results - 2D explosion problem - Annulus of Maxwellians

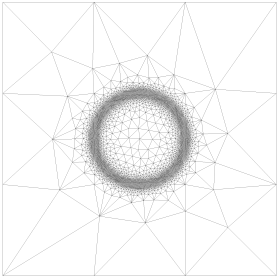







Initial guess


Adapted $t = 0$


Adapted $t > 0$

Boltzmann collision operator discretization

Boltzmann collision operator $\mathcal{Q}(f)$ in $\partial_t f + \boldsymbol{V} \cdot \nabla_{\boldsymbol{X}} f = \frac{1}{\tau}\mathcal{Q}(f)$

Assume B locally integrable and short range interaction models [9]

$$\mathcal{Q}(f)(v) = \int_{v_\star \in \mathbb{R}^d} \int_{\sigma \in \mathbb{S}^{d-1}} B(|v - v_\star|, \cos\theta)(f'_\star f' - f_\star f) d\sigma dv_\star$$

$$v' = v - 1/2((v - v_\star) - |v - v_\star|\sigma), \quad v'_\star = v - 1/2((v - v_\star) + |v - v_\star|\sigma),$$

$$\cos\theta = u.\sigma, \quad u = v - v_\star$$

Maxw. molecules model
$B(u, \cos\theta) = 1$
Hard Sphere model
$B(u, \cos\theta) = |u|$
Variable HS model
$B(u, \cos\theta) = C_\gamma |u|^\gamma$

Classical spectral method [8] or fast (FFT) spectral method [7,9]

BGK relaxation model was expensive, nothing to compare against Boltzmann !

Derivation based on periodized truncation of the operator, truncated Fourier expansion of f and projection of collision operator on $\mathbb{P}_N$ (polyn. of degree n in each direction) [9] leads to a syst. of ODEs. $\widehat{\beta}(l, m)$ may be complex depending on the model.

$$\partial_t \widehat{f_k} = \sum_{\substack{|l|,|m| \le n \\ |k| \le n}}^{l+m=k} \widehat{\beta}(l, m) \widehat{f_l} \widehat{f_m}$$

[7] A. V. Bobylev and S. Rjasanow, Difference scheme for the Boltz. eq. based on the FFT, Eur. J. Mech. B Fluids, 16 (1997)
[8] L. Pareschi and G. Russo, Numerical solution of the Boltz. eq. I:..., SIAM J. Numer. Anal., 37 (2000)
[9] F.Filbet On deterministic approx of the Boltzmann equation... SIAM MULTISCALE MODEL. SIMUL., 10, 3, 2011

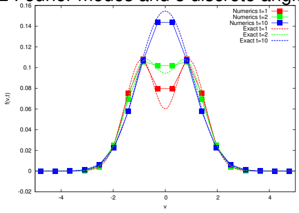
Boltzmann collision operator discretization

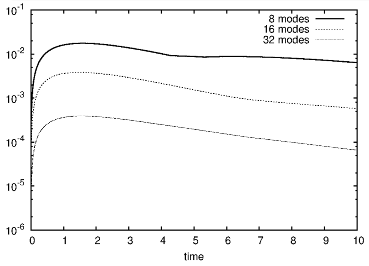
Preliminary validation tests in 0D×2D

0D×2D and Maxwellian molecule model

Velocity domains: $[-V, V]^2$, $V = 4, 6, 9$ for respect. $N_m = 8, 16, 32$ Fourier modes and 8 discrete angles θ.
Test problem with an exact solution

$$\begin{aligned} f(v,t) &= \frac{\exp(-v^2/2S)}{2\pi S^2}\left[2S - 1 + \frac{1-S}{2S}v^2\right] \\ S &= 1 - \exp(-t/8)/2, \qquad t = 10 \end{aligned}$$





L_1 and L_2 errors for the fast spectral method

# modes	Error E_1	Error E_2
8	0.0063	0.0038
16	0.00057	0.00029
32	0.000065	0.000072

Figure for L_1 error.

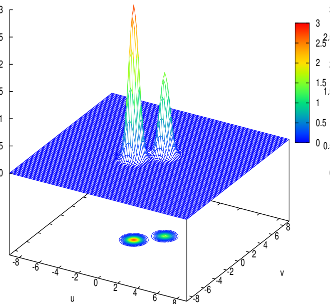
Boltzmann collision operator discretization

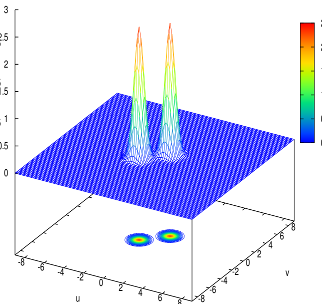
Preliminary validation tests in 0D×3D

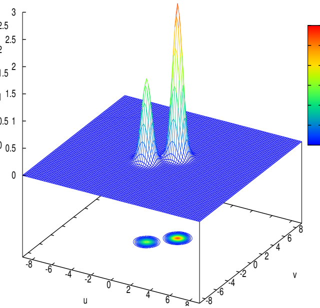
3D Maxwellian molecules and the initial condition

$$f(v, t = 0) = \frac{1}{2(2\pi\sigma^2)^{3/2}} \left[\exp\left(-\frac{|v - v_1|^2}{2\sigma^2} \right) + \exp\left(-\frac{|v + v_1|^2}{2\sigma^2} \right) \right],$$

$\sigma^2 = 0.2$, $v_1 = (-1, -1, -0.25)$, $t_{\text{final}} = 2$.





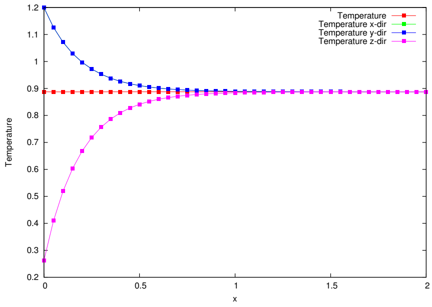


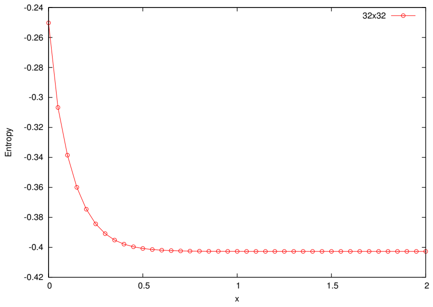
Boltzmann collision operator discretization

Preliminary validation tests in 0D×3D - directional temperature, entropy

$$T_i(t) = \frac{1}{\rho}\int_{\mathbb{R}^3} (v_i - u_i)^2 \, f(t,v)\, dv, \quad i = 1,2,3, \qquad H(t) = \int_{\mathbb{R}^3} f(t,v)\, \log(f(t,v))\, dv.$$

32 Fourier modes



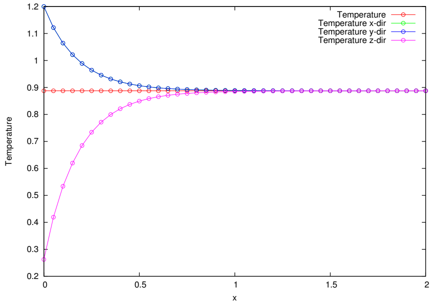


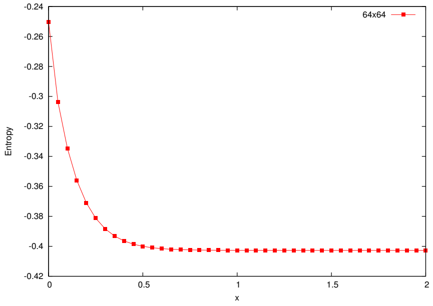
Boltzmann collision operator discretization

Preliminary validation tests in 0D×3D - directional temperature, entropy

$$T_i(t) = \frac{1}{\rho} \int_{\mathbb{R}^3} (v_i - u_i)^2\, f(t,v)\, dv, \quad i = 1,2,3, \qquad H(t) = \int_{\mathbb{R}^3} f(t,v)\, \log(f(t,v))\, dv.$$

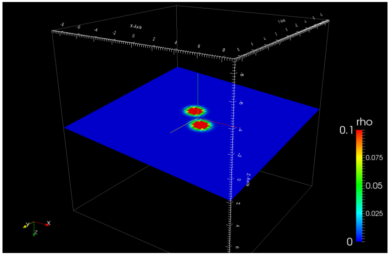
64 Fourier modes

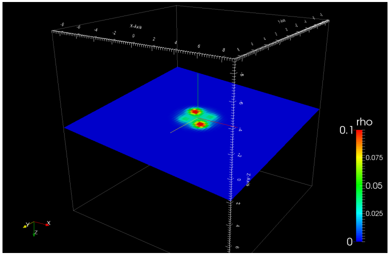


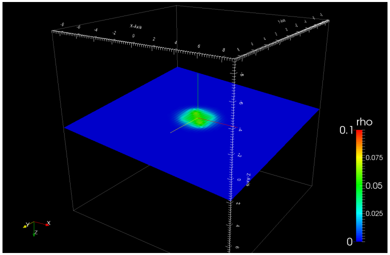


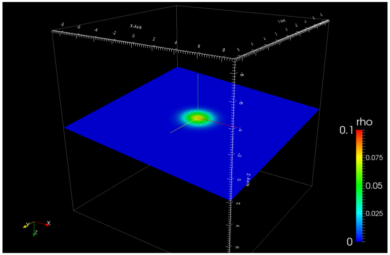
Boltzmann collision operator discretization

Preliminary validation tests in 0D$\times$3D - distribution function





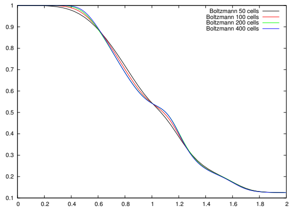


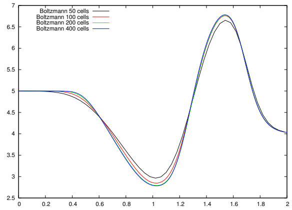


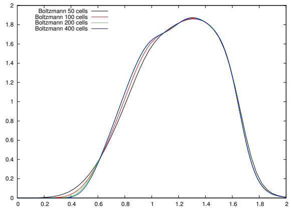
Boltzmann collision operator discretization

Preliminary tests in 1D$\times$2D - convergence of the model

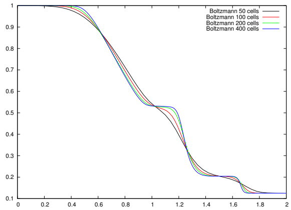
$\tau = 10^{-3}$

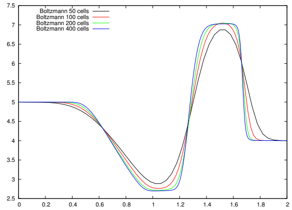


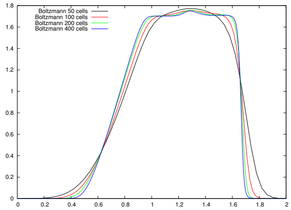




$\tau = 10^{-4}$



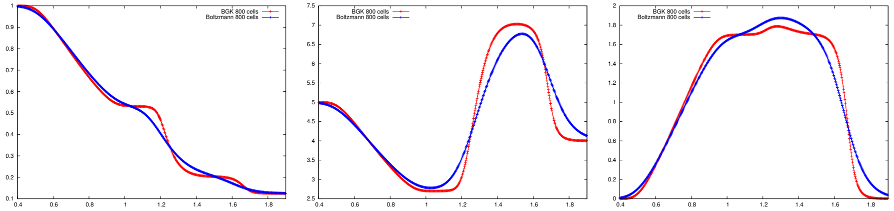




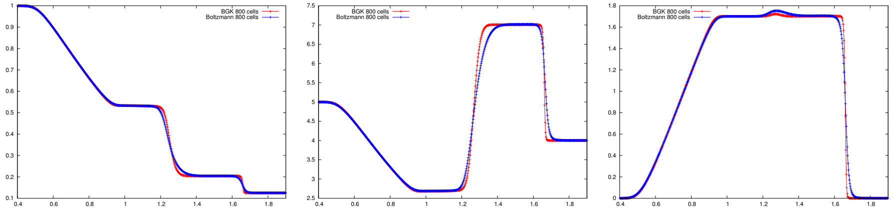
Boltzmann collision operator discretization

Preliminary tests in 1D$\times$2D - Boltzmann vs. BGK

$\tau = 10^{-3}$, $N_c = 800$, $N_v^2 = 64^2$

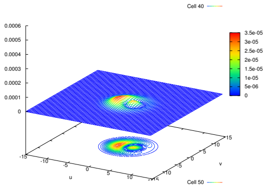


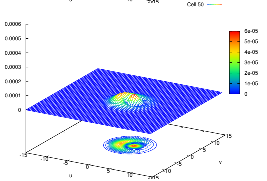
$\tau = 10^{-4}$, $N_c = 800$, $N_v^2 = 64^2$

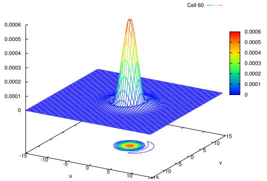


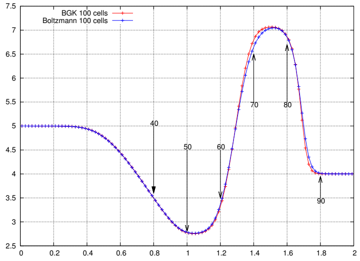
Boltzmann vs. BGK

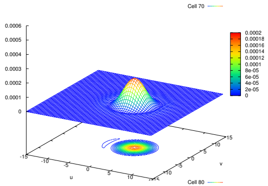
Sod in 1D$\times$2D

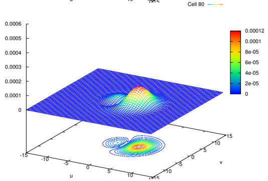


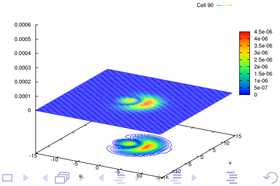












Boltzmann vs. BGK

Isentropic vortex in 2D$\times$2D

Vortex in motion

$$u = u_\infty + \delta u, \quad v = v_\infty + \delta v, \quad \theta^* = \theta^*_\infty + \delta\theta^*$$

$$\rho_\infty = 1.0, \quad u_\infty = 1.0, \quad v_\infty = 1.0, \quad p_\infty = 1.0, \quad \theta^*_\infty = 1.0$$

$$\delta u = -y' \frac{\beta}{2\pi} \exp\left(\frac{1-r^2}{2}\right), \quad \delta v = x' \frac{\beta}{2\pi} \exp\left(\frac{1-r^2}{2}\right), \quad \delta\theta^* = -\frac{(\gamma-1)\beta}{8\gamma\pi^2} \exp\left(1-r^2\right),$$

$$r = \sqrt{x'^2 + y'^2}, \quad x' = x - x_{\text{vortex}}, \quad y' = y - y_{\text{vortex}}$$

$$\rho = \rho_\infty \left(\frac{\theta^*}{\theta^*_\infty}\right)^{\frac{1}{\gamma-1}} = \left(1 - \frac{(\gamma-1)\beta}{8\gamma\pi^2} \exp\left(1-r^2\right)\right)^{\frac{1}{\gamma-1}}$$

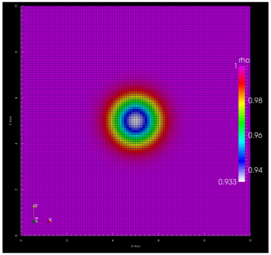
Parameters

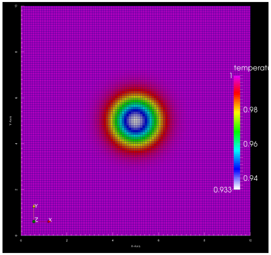
- Space domain : $[0;10]^2$ discretized with $N_c = N \times N = 100 \times 100$ points
- Velocity domain : $[-7.5;7.5]^2$ discretized with $N_v = M \times M = 32 \times 32$ points
- Periodic boundary conditions everywhere
- Simulation time: $t_f = 10$ (vertex back in its initial position)
- $\tau = 10^{-1} \Rightarrow$ no analytical solution

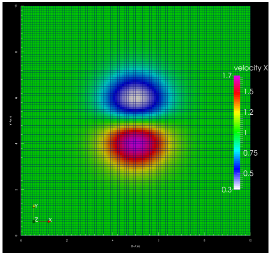
Boltzmann vs. BGK

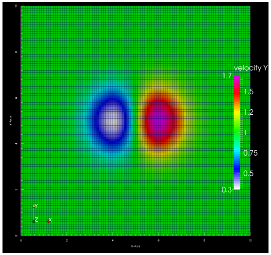
Isentropic vortex in 2D$\times$2D

Initial conditions





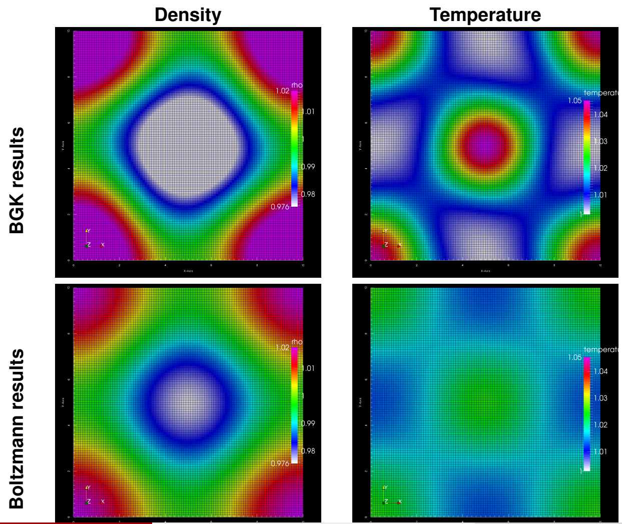




Boltzmann vs. BGK

Isentropic vortex in 2D×2D

Final time



Boltzmann vs. BGK

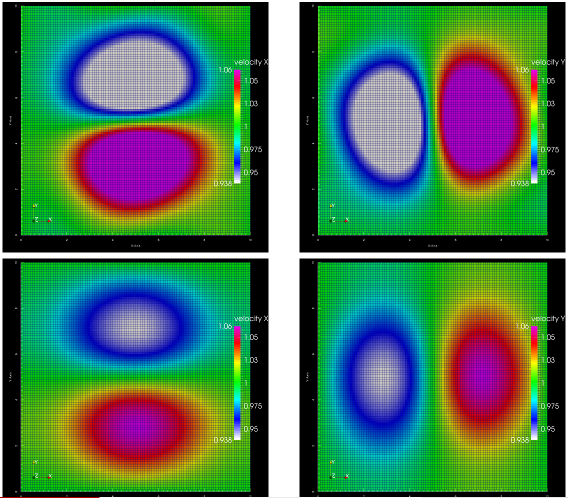
Isentropic vortex in 2D×2D

Final time

Velocity u **Velocity** v

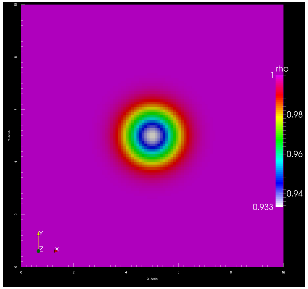
BGK results

Boltzmann results

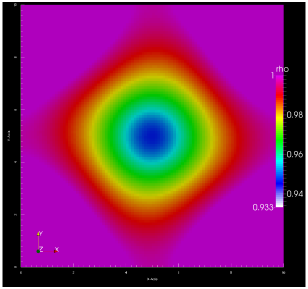


Boltzmann vs. BGK

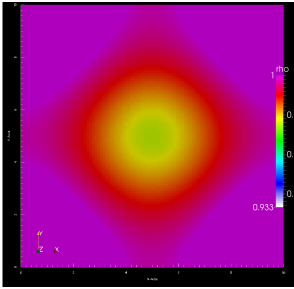
Isentropic vortex in 2D$\times$2D



Initial density



BGK model results



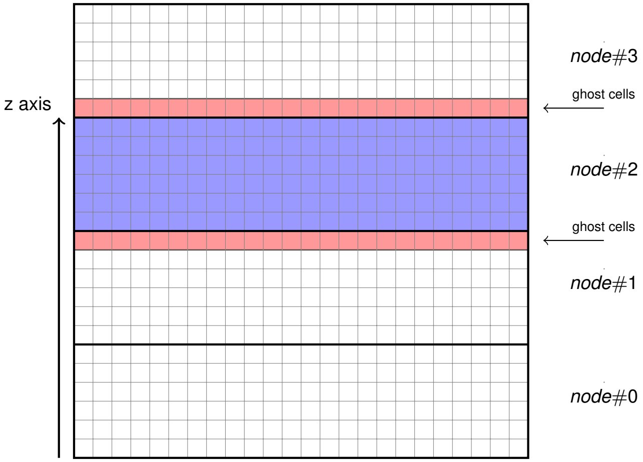
Boltzmann model results

Boltzmann vs. BGK

Isentropic vortex in 2D$\times$2D

Model	Velocity N_v	Vel.	Cell # $N_c \times N_v$	Cycle N_{cycle}	Time T (s)	T/cycle T_{cycle} (s)	T/cell T_{cell} (s)	T/d.o.f T_{dof} (s)
Vortex problem in 2D$\times$2D								
BGK	32^2	$[-7.5, 7.5]$	$25^2 \times 32^2 = 64 \times 10^4$	351	2.61	0.0035	5.60×10^{-6}	5.47×10^{-9}
			$50^2 \times 32^2 = 256 \times 10^4$	701	13.77	0.0196	7.84×10^{-6}	7.66×10^{-9}
			$100^2 \times 32^2 = 1024 \times 10^4$	1400	102.42	0.0732	7.32×10^{-6}	7.15×10^{-9}
			$200^2 \times 32^2 = 4096 \times 10^4$	2800	785.54	0.2806	7.02×10^{-6}	6.85×10^{-9}
Boltzmann	32^2	$[-7.5, 7.5]$	$25^2 \times 32^2 = 64 \times 10^4$	351	32.92	0.0938	1.50×10^{-4}	1.47×10^{-7}
			$50^2 \times 32^2 = 256 \times 10^4$	701	245.03	0.350	1.40×10^{-4}	1.37×10^{-7}
			$100^2 \times 32^2 = 1024 \times 10^4$	1400	2008.56	1.435	1.44×10^{-4}	1.40×10^{-7}
			$200^2 \times 32^2 = 4096 \times 10^4$	2800	15762	5.630	1.41×10^{-4}	1.37×10^{-7}

MPI domain decomposition



Strategy

- Divide physical domains into slices along *z* axis
- Each slice contains $N_x \times N_y \times N_{z,loc} \times N_v^3$ particles
- 2 ghost layers of the size $N_x \times N_y \times N_v^3$
- One slice per MPI process
- After each iteration : broadcast masses of particles escaping from a slice to neighbouring MPI processes ($N_x \times N_y \times N_v^3$ at most)

Speedup

EOS supercomputer

- 612 nodes
- 64Gb per node (total 39Tb)
- 2 $\times$ 10 core Intel Ivybridge 2.8 GHz processor per core (total of 12240 computational cores)

Simulations were run on different meshes:

BGK

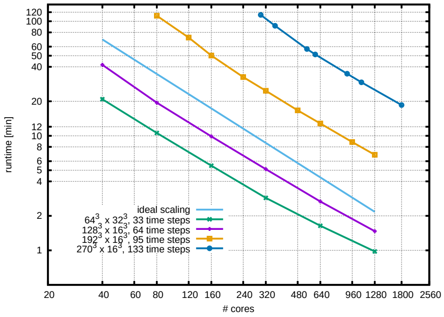
- $N = 64^3$ and $N_v = 32$, $64Gb$, at least 2 nodes,
- $N = 128^3$ and $N_v = 16$, $64Gb$, at least 2 nodes,
- $N = 192^3$ and $N_v = 16$, $216Gb$, at least 4 nodes,
- $N = 270^3$ and $N_v = 16$, $600Gb$, at least 15 nodes,

Boltzmann

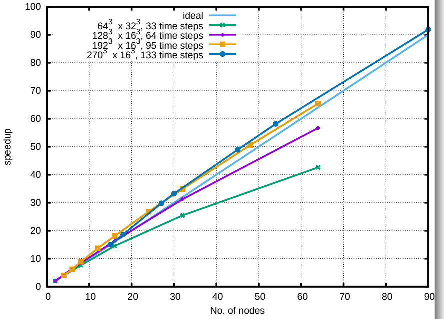
- $N = 64^3$ and $N_v = 16$, $8Gb$, at least 2 nodes,
- $N = 64^3$ and $N_v = 32$, $64Gb$, at least 2 nodes,

Speedup

BGK



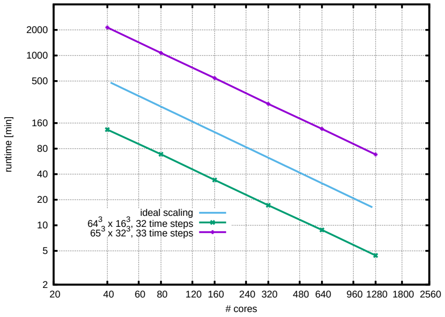
runtime [min]
# cores
ideal scaling
64^3 x 32^3, 33 time steps
128^3 x 16^3, 64 time steps
192^3 x 16^3, 95 time steps
270^3 x 16^3, 133 time steps
120
100
80
60
50
40
20
12
10
8
6
5
4
2
1
20
40
60
80
120
160
240
320
480
640
960
1280
1800
2560

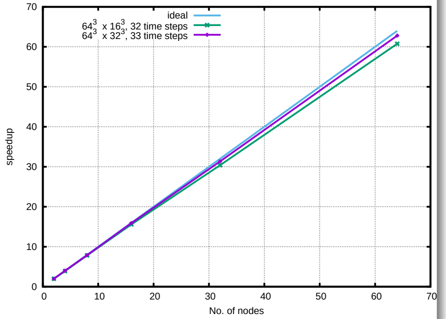


speedup
No. of nodes
ideal
64^3 x 32^3, 33 time steps
128^3 x 16^3, 64 time steps
192^3 x 16^3, 95 time steps
270^3 x 16^3, 133 time steps
100
90
80
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90

Speedup

Boltzmann





Acknowledgements

This work was partially supported by the PHC Gallileo 2015-2016 number 32272UL.

This work has been supported by the French 'Agence Nationale pour la Recherche (ANR)' in the frame of the contract “MOONRISE” (ANR- 14-CE23-0007-01).

This work was performed using HPC resources from CALMIP (Grant 2015-P1542)

THANK YOU FOR YOUR ATTENTION