CFD and state-to-state of hypersonic flows using GPUs

Francesco Bonelli[a], Michele Tuttafesta[b], Gianpiero Colonna[c],
Luigi Cutrone[d], Giuseppe Pascazio[a]

[a]DMMM,Politecnico di Bari via Re David 200, 70125, Bari, Italy
[b]Liceo Scientifico Statale "L. da Vinci", Via Cala dell'Arciprete 1 - 76011 Bisceglie (BT), Italy
[c]CNR-IMIP, via Amendola 122/D - 70126 Bari (Italy)
[d]Centro Italiano Ricerche Aerospaziali (CIRA), Capua, 81043, Italy

bonellifra@alice.it

Objective and Outline

Objective:

Development of a High Performance Computing (HPC) CFD code for the investigation of high enthalpy flows

Outline:

- **Motivation: the atmospheric entry problem**
- **Governing equations**
- **Numerical method**
- **Thermochemical non–equilibrium models**
- **GPU and multi–GPUs parallel computing with CUDA and MPI–CUDA**
- **Results**
- **Conclusions**

Why hypersonic flows?

Space exploration: the atmospheric entry problem

- a strong shock wave is formed in front of the vehicle
- kinetic energy of the incoming molecules is converted into internal energy
- a tremendous heat load weighs on the vehicle
- a suitable Thermal Protection System (TPS) is needed

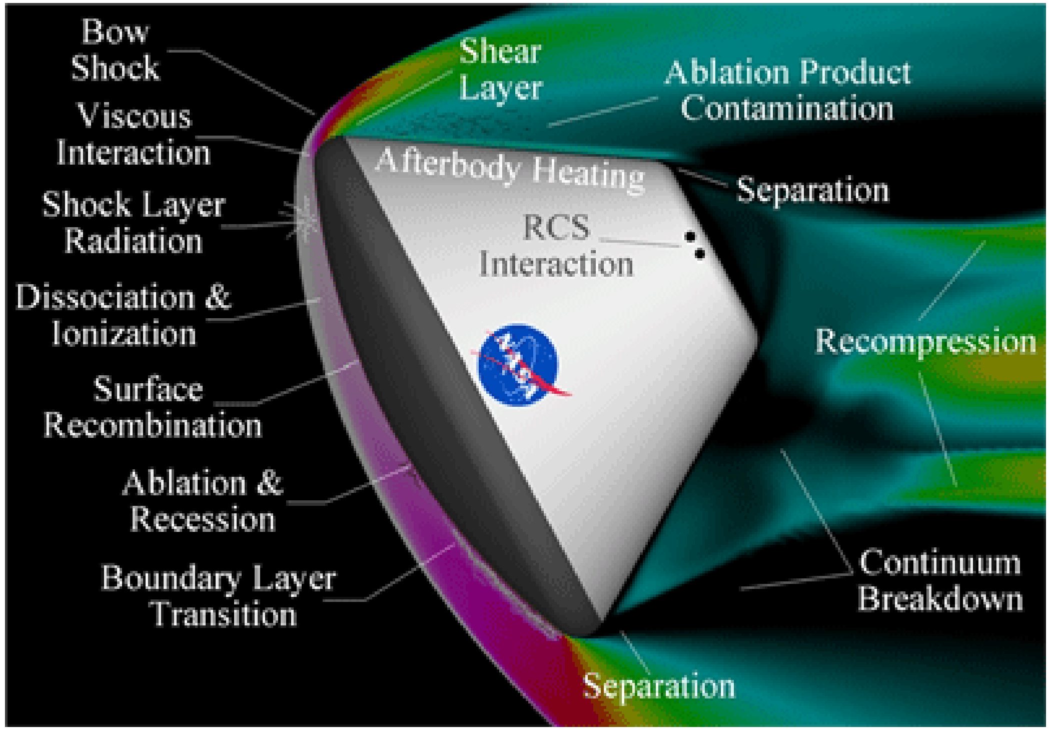


Figure taken from http://class.tamu.edu/media/22851/pecos.gif

Atmospheric entry: a multi-physics problem

- a mixture of vibrationally/electronically excited and chemical reacting non-equilibrium flow is formed;
- de-excitation of the electronic mode causes a significant amount of radiation;
- temperature drops in the boundary layer are strong enough to cause recombination;
- at the surface of the vehicle a huge amount of heat is transferred.

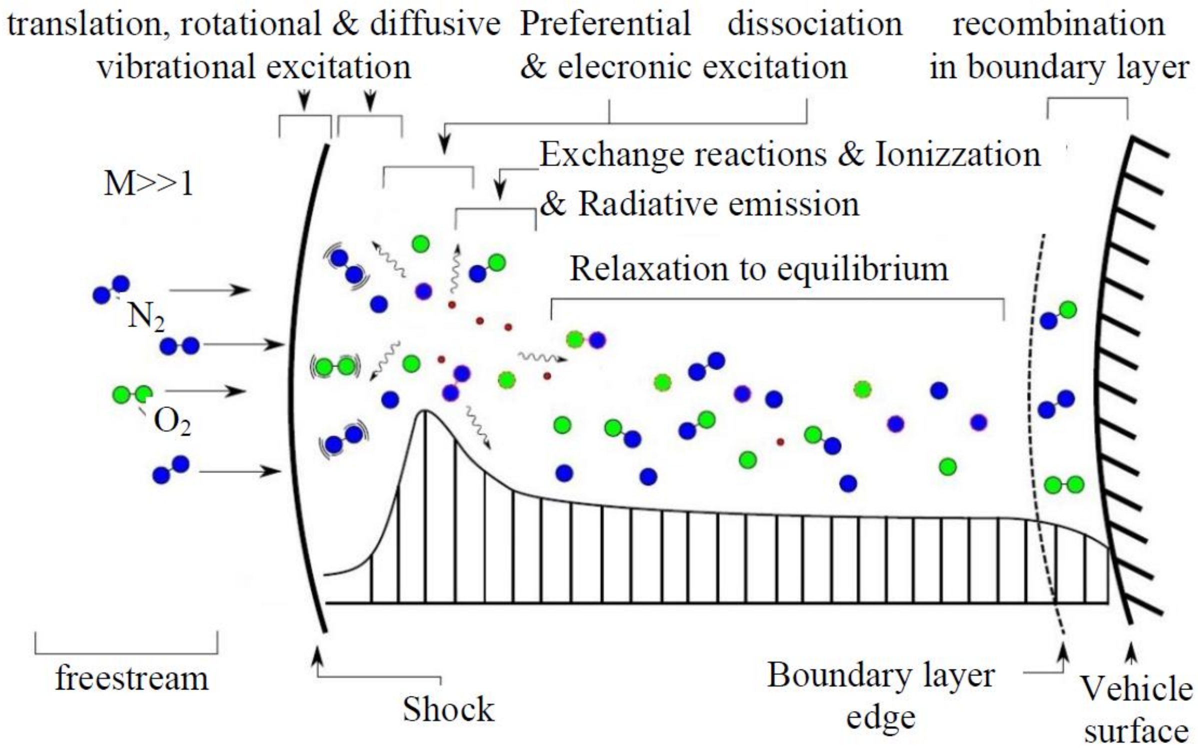


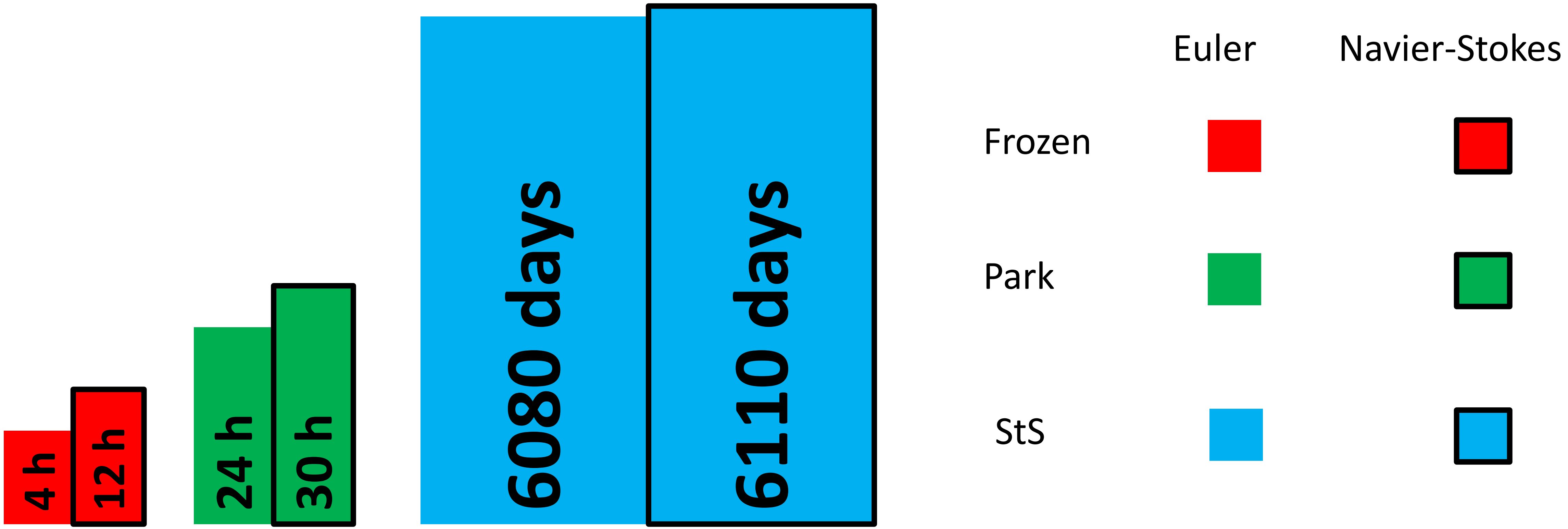
Figure taken from: D.F. Potter, Modelling of radiating shock layers for atmospheric entry at Earth and Mars, PhD thesis, The University of Queensland, Australia, 2011

In order to properly predict such phenomena a key role is played by the thermochemical non-equilibrium model. Two different approaches can be followed:

- **the classical multi-temperature approach: based on simplified hypothesis; not computational demanding (17 reactions for a neutral air mixture)**
- **the State-to-State (StS) approach: no simplified hypothesis; very computational demanding (thousand of reactions)**

Does StS need parallel computing? YES

Single core CPU computational time to complete a simulation for a hypersonic flow of a 5 species neutral air mixture past a sphere: 2D 512x256 mesh



Governing equations

Compressible Navier-Stokes (N-S) equations for a multicomponent mixture of reacting gases in thermochemical non-equilibrium for both Park and StS models

$$\frac{\partial}{\partial t}\int_{V_0}\mathbf{U}dV+\oint_{S_0}\mathbf{F}\cdot\mathbf{n}dS=\int_{V_0}\mathbf{W}dV$$

$$\mathbf{U}=[\rho_{1,1},..,\rho_{1,V1},..,\rho_{S,1},..,\rho_{S,V_S},\rho u,\rho v,\rho e,\rho_1 e_{vib,1},...,\rho e_{vib,M}]^T$$

$$\mathbf{F}=\left(\mathbf{F}_E-\mathbf{F}_V,\mathbf{G}_E-\mathbf{G}_V\right)$$

$$\mathbf{F}_{\mathrm{E}}=[\rho_{1,1}u,..,\rho_{1,V1}u,..,\rho_{S,1}u,..,\rho_{S,V_S}u,\rho u^2+p,\rho uv,(\rho e+p)u,\rho_1 e_{vib,1}u,...,\rho e_{vib,M}u]^T$$

$$\mathbf{G}_{\mathrm{E}}=[\rho_{1,1}v,...,\rho_{1,V1}v,...,\rho_{S,1}v,...,\rho_{S,V_S}v,\rho uv,\rho v^2+p,(\rho e+p)v,\rho_1 e_{vib,1}v,...,\rho e_{vib,M}v]^T$$

$$\mathbf{W}=[\dot{\omega}_{1,1},...,\dot{\omega}_{1,V_1},...,\dot{\omega}_{S,1},...,\dot{\omega}_{S,V_1},0,0,0,\dot{\omega}_{vib,1},....,\dot{\omega}_{vib,M}]^T$$

U is the vector of the conservative variables, $\mathbf{F}_E$ / $\mathbf{F}_V$ and $\mathbf{G}_E$ / $\mathbf{G}_V$ are the inviscid/viscous flux vectors and W is the source terms vector.

S **is the number of chemical components, the s^th one having *Vs* internal levels, the state-to-state approach considers** $N=\sum_{s=1}^{S}V_s$ **independent species, whereas *Vs=1* in the case of the Park's model so that *N=S***

Governing equations

$$\left(\mathbf{F}_{\mathrm{V}}, \mathbf{G}_{\mathrm{V}}\right) = \left[-\rho_i \mathbf{u}_i, \underline{\underline{\tau}}, \mathbf{u}\cdot\underline{\underline{\tau}} - \mathbf{q}, -\mathbf{q}_{vib,1}, \ldots\ldots, -\mathbf{q}_{vib,M} \right]^T$$

$$-\rho_i \mathbf{u}_i = -\rho D_i \nabla Y_i$$

$$\underline{\underline{\tau}} = \mu\left[\nabla\mathbf{u} + \left(\nabla\mathbf{u}\right)^T\right] - \frac{2}{3}\mu\nabla\cdot\mathbf{u}\mathbf{I}$$

$$\mathbf{q} = -\lambda_t \nabla T - \lambda_{vib}\nabla T_{vib} + \sum h_i \rho_i \mathbf{u_i}$$

$$\mathbf{q}_{vib,m} = -\lambda_{vib}\nabla T_{vib} + e_{vib,m}\rho_m \mathbf{u}_m$$

F_V and G_V are the viscous flux vectors

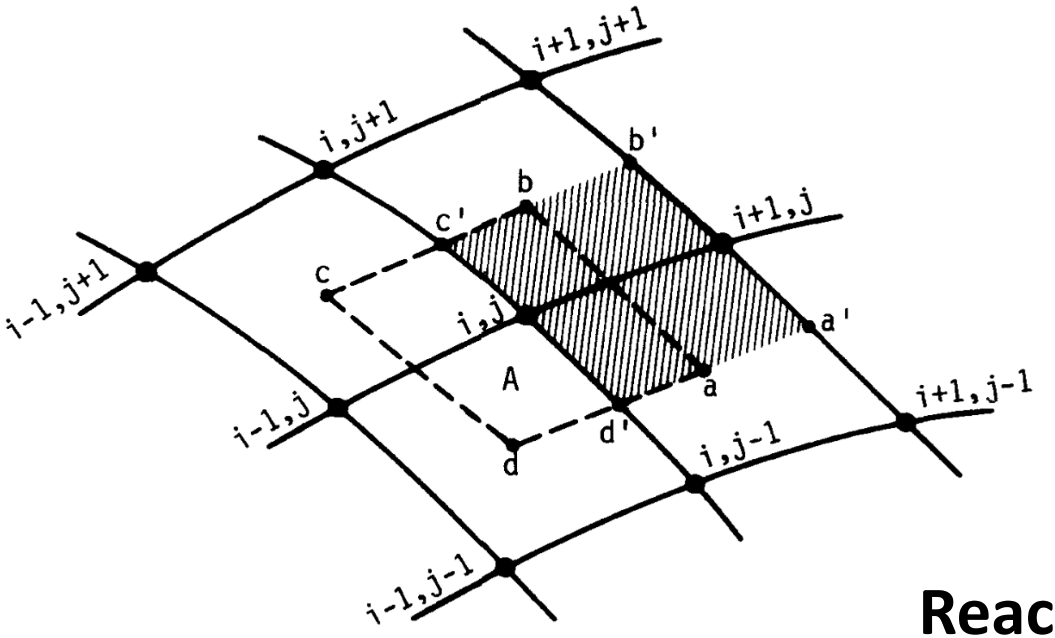
In the present implementation transport properties of single species are evaluated by using Gupta's curve fits*. Classical mixing rules are used for mixture properties

*Gupta et al. A review of reaction rates and thermodynamic and transport properties for an 11-species air model for chemical and thermal nonequilibrium calculations to 30000 K, NASA. Reference. Publication. 1232. 1990

Numerical method

$$V_{i,j}\frac{d\mathbf{U}_{i,j}}{dt}+\sum_{Faces}\mathbf{F}_{num}\cdot\mathbf{n}\Delta S=V_{i,j}\mathbf{W}_{i,j}$$

Cell-centered Finite Volume Space discretization on a Multi-block structured mesh



$$\mathbf{F}_{num}=\mathbf{F}_{E,num}-\mathbf{F}_{V,num}$$

Reactive Navier-Stokes equations:

- **Advection and pressure term** (hyperbolic)
- **Shear-stress, heat flux terms** (diffusive)
- **Chemical source terms** (stiffness)

Figure taken from: John C. Tannehill, Dale Anderson, Richard H. Pletcher, Computational Fluid Mechanics and Heat Transfer, Taylor & Francis 1997

Numerical method

$$V_{i,j}\frac{d\mathbf{U}_{i,j}}{dt}+\sum_{Faces}\mathbf{F}_{num}\cdot\mathbf{n}\Delta S=V_{i,j}\mathbf{W}_{i,j}$$

$$\mathbf{F}_{num}=\mathbf{F}_{E,num}-\mathbf{F}_{V,num}$$

Solution strategy:

- Operator splitting approach: Frozen step + Chemical step
  - ✓ Frozen step: Method of Lines:
    - Space discretization + Time integration
      - Space dicretization: Inviscid & Viscous terms scheme
      - Time integration: Runge-Kutta scheme
  - ✓ Chemical step: implicit scheme for stiff terms

Frozen step

$$V_{i,j}\frac{d\mathbf{U}_{i,j}}{dt}+\sum_{Faces}\mathbf{F}_{num}\cdot\mathbf{n}\Delta S=0$$

Frozen equation

Semi-Discrete Schemes or Method of Lines

$$\frac{d\mathbf{U}_{i,j}}{dt}=-\frac{1}{V_{i,j}}\sum_{Faces}\left(\mathbf{F}_{E,num}-\mathbf{F}_{V,num}\right)\cdot\mathbf{n}\Delta S$$

ODE solved with an explicit Runge-Kutta schemes

$\mathbf{F}_{E,num}$ **Methods for solving non-linear hyperbolic conservation laws**

Frozen step: inviscid flux space discretization

Steger and Warming Flux Vector Splitting

The discretisation of the equations on a mesh is performed according to the direction of propagation of information on that mesh.

Upwinding is performed by splitting the flux in positive and negative components.

$$\mathbf{U}_t + \mathbf{F}_x(\mathbf{U}) = 0 \longrightarrow \mathbf{U}_t + \frac{\partial \mathbf{F}}{\partial \mathbf{U}}\frac{\partial \mathbf{U}}{\partial \mathbf{X}} = 0 \longrightarrow \mathbf{U}_t + \mathbf{A}\mathbf{U}_x = 0$$

$$\mathbf{F} = \mathbf{A}\mathbf{U}$$ homogeneous function of degree one

$$\mathbf{F} = \mathbf{K}\Lambda\mathbf{K}^{-1}\mathbf{U} = \mathbf{K}\left(\Lambda^{+} + \Lambda^{-}\right)\mathbf{K}^{-1}\mathbf{U} = \left(\mathbf{A}^{+} + \mathbf{A}^{-}\right)\mathbf{U} = \mathbf{F}^{+} + \mathbf{F}^{-}$$

$$\lambda_i^{-} = \min(\lambda_i, 0) = \frac{1}{2}\left(\lambda_i - |\lambda_i|\right) \qquad \lambda_i^{+} = \max(\lambda_i, 0) = \frac{1}{2}\left(\lambda_i + |\lambda_i|\right)$$

$$\mathbf{F}^{\pm}_{1/2} = \frac{\rho}{2\gamma}\begin{bmatrix} 2(\gamma-1)\lambda_1^{\pm} + \lambda_2^{\pm} + \lambda_3^{\pm} \\ 2(\gamma-1)\lambda_1^{\pm}u + \lambda_2^{\pm}(u+a) + \lambda_3^{\pm}(u-a) \\ (\gamma-1)\lambda_1^{\pm}u^2 + \frac{\lambda_2^{\pm}}{2}(u+a)^2 + \frac{\lambda_3^{\pm}}{2}(u-a)^2 + \frac{(3-\gamma)\left(\lambda_2^{\pm} + \lambda_3^{\pm}\right)a^2}{2(\gamma-1)} \end{bmatrix}$$

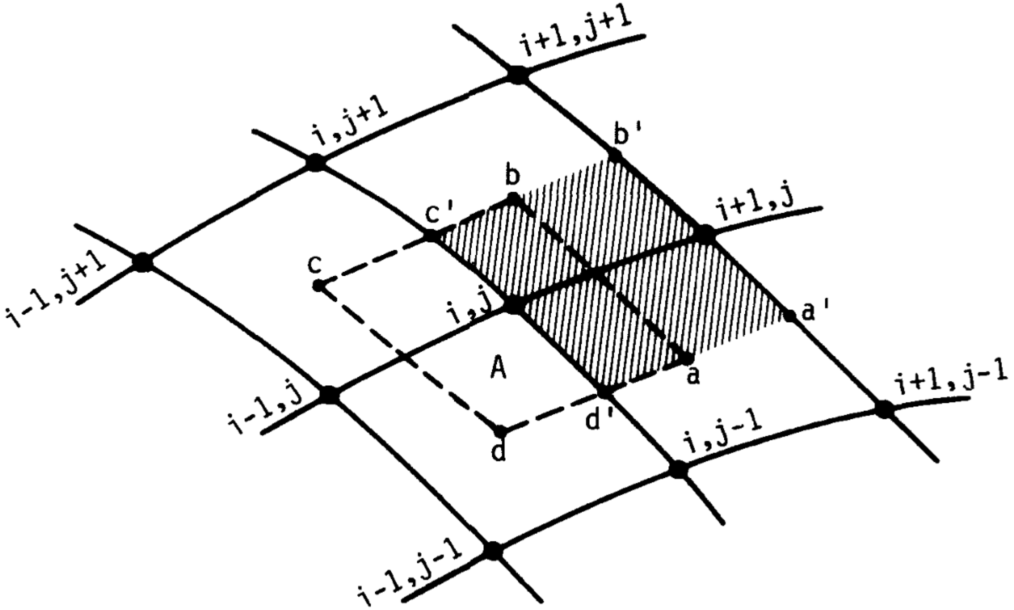
J.L. Steger, R.F. Warming , Flux vector splitting of the inviscid gasdynamic equations with application to finite-difference methods, Journal of Computational Physics 40 (2) 263-293, 1981

Frozen step: viscous schemes

Viscous terms involve gradients that have to be determined on the cell faces

Due to their dissipative nature central differences are used

A good procedure for generalized curvilinear coordinates is to apply the Gauss divergence theorem



Control volume and Gauss cell (shaded area) for cell-faces derivatives

$$\int_{V_0} \nabla u dV = \oint_{S_0} u d\mathbf{S}$$

$$\nabla u = \frac{1}{V} \sum_{i=faces} u_i d\mathbf{S}_i$$

$$\begin{pmatrix} u_x \\ u_y \end{pmatrix} = \frac{1}{V} \begin{pmatrix} \sum_{i=faces} u_i dS_{x,i} \\ \sum_{i=faces} u_i dS_{y,i} \end{pmatrix}$$

Figure taken from: John C. Tannehill, Dale Anderson, Richard H. Pletcher, Computational Fluid Mechanics and Heat Transfer, Taylor & Francis 1997

Numerical method

Operator splitting approach

$$\frac{\partial}{\partial t}\int_{V_0} \mathbf{U} dV + \oint_{S_0} \mathbf{F} \cdot \mathbf{n} dS = 0$$

Frozen step

Inviscid flux: Flux Vector Splitting of Steger and Warming or AUSM with MUSCL approach for higher order accuracy;

Viscous flux: gradients of the primitive variables are evaluated by applying Gauss theorem

Time integration: Runge-Kutta scheme up to third order for time integration

Numerical method

Operator splitting approach

$$\frac{\partial}{\partial t}\int_{V_0}\mathbf{U}dV=\int_{V_0}\mathbf{W}dV$$

Chemical step

$$\Delta t_c^{(v)}=\Delta t_f\,/\,n$$

Sub-time step

$$\frac{\partial\mathbf{y}}{\partial t}=\mathbf{P}-\mathbf{L}\mathbf{y}\qquad \mathbf{y}=\{\rho_i\}_{0\le i\le N}$$

P is a vector and L a diagonal matrix.
***P**$_i$* **and** ***L**$_i$**y**$_i$* **are non-negative and represent, respectively, production and loss terms for component** ***y**$_i$*

$$y_i^k(t+\Delta t_c^{(v)})=\frac{\Delta t_c^{(v)}P_i(\mathbf{y}^{k-1})+y_i(t)}{1+\Delta t_c^{(v)}L(\mathbf{y}^{k-1})}$$

Gauss-Seidel iterative scheme

Thermochemical non-equilibrium models for a 5 species neutral air mixture

MULTI-TEMPERATURE 5 SPECIES PARK MODEL[1]

- 17 reactions + 3 transport equations for the vibrational energies
- Arrhenius type rate coefficients function of an effective temperature calculated as a geometrical mean of translational (T) and vibrational temperatures (Tv)
- Vibrational levels follow a Boltzmann distribution at temperature Tv
- Tuned on experimental measures
- Not computationally demanding
- It may fail when the conditions are far from those for which it was tuned

5 SPECIES State-to-State (StS) MODEL[2]

- Detailed vibrational kinetics of molecules.
- 68 and 47 vibrational levels for N_2 and O_2 respectively
- Thousands of elementary processes → High accuracy but huge computational cost

[1] C. Park, Nonequilibrium Hypersonic Aerothermodynamics, Wiley, New York, 1990

[2] M. Capitelli et al., Fundamentals Aspects of Plasma Chemical Physics: Kinetics, Springer Science & Business Media, 2015

Multi-temperature 5 species Park model

REACTIONS:

Dissociation

$N_2+X \leftarrow \rightarrow 2N + X$

$O_2+X \leftarrow \rightarrow 2O + X$

$NO+X \leftarrow \rightarrow N + O + X$

with $X=N_2, O_2, NO, N, O$

Zeldovich exchange reactions

$N_2+O \leftarrow \rightarrow N + NO$

$O_2+N \leftarrow \rightarrow NO + O$

$$\sum_{k=1}^{K} \upsilon'_{ki}\,\chi_k \Leftrightarrow \sum_{k=1}^{K} \upsilon''_{ki}\,\chi_k$$

Generic *i*

$$\dot{\omega}_k = M_k \sum_{i=1}^{I} \upsilon_{ki} q_i$$

Chemical production rate of the *k*th species

$$\upsilon_{ki} = \upsilon''_{ki} - \upsilon'_{ki}$$

Net stoichiometric coefficient

$$q_i = k_{fi} \prod_{k=1}^{K} [X_k]^{\upsilon'_{ki}} - k_{ri} \prod_{k=1}^{K} [X_k]^{\upsilon''_{ki}}$$

Rate of progress of the *i*th reaction

Multi-temperature 5 species Park model

The two-temperature Park model assumes that the Arrhenius rate constants are functions of a geometrically averaged between the translational-rotational temperature (T) and the vibrational temperature (T_v) in the form:

$$T_a = T_v^q T^{1-q}$$

with q between 0.3 and 0.7

$$k_{fi} = A_i T^{\beta_i} \exp\left(\frac{-E_i}{R_c T_a} \right)$$

Arrhenius forward rate constant

$$k_{ri} = \frac{k_{fi}(T_a)}{K_{C_i}(T_a)}$$

reverse rate constant

$$\dot{\omega}_{LT,m} = \rho_m \frac{e_{vib,m}(T) - e_{vib,m}(T_{V,m})}{\tau_m}$$

Landau Teller evolution of the vibrational energy

τ_m **Vibrational energy relaxation time (Millikan-White expression)**

5 species State-to-State (StS) model

The State-to-State approach write a relaxation equation for each vibrational level so that it is possible to calculate the distribution of internal states when it departs from the Boltzmann one.

Pure N_2

$N_2(v)+N_2 \leftrightarrow N_2(v-1)+N_2$	vTm
$N_2(v)+N \leftrightarrow N_2(v-\Delta v)+N$	vTa
$N_2(v)+N_2(w-1) \leftrightarrow N_2(v-1)+N_2(w)$	vv
$N_2(v)+N_2 \leftrightarrow 2N+N_2$	drm
$N_2(v)+N \leftrightarrow 2N+N$	dra

Pure O_2

$O_2(v)+O_2 \leftrightarrow O_2(v-1)+O_2$	vTm
$O_2(v)+O \leftrightarrow O_2(v-\Delta v)+O$	vTa
$O_2(v)+O_2(w-1) \leftrightarrow O_2(v-1)+O_2(w)$	vv
$O_2(v)+O_2 \leftrightarrow 2O+O_2$	drm
$O_2(v)+O \leftrightarrow 2O+O$	dra

Mixed N_2

$N_2(v)+O_2 \leftrightarrow N_2(v-1)+O_2$	vTm
$N_2(v_{\max})+O_2 \leftrightarrow 2N+O_2$	LC
$N_2(v)+O \leftrightarrow N_2(v-1)+O$	vTa
$N_2(v_{\max})+O_2 \leftrightarrow 2N+O_2$	LC

Mixed O_2

$O_2(v)+N_2 \leftrightarrow O_2(v-1)+N_2$	vTm
$O_2(v_{\max})+N_2 \leftrightarrow 2O+N_2$	LC
$O_2(v)+N \leftrightarrow O_2(v-1)+N$	vTa
$O_2(v_{\max})+N \leftrightarrow 2O+N$	LC

$O_2(v)+N_2(w-1) \leftrightarrow O_2(v-2)+O_2(w)$	vv

Zeldovich exchange reactions

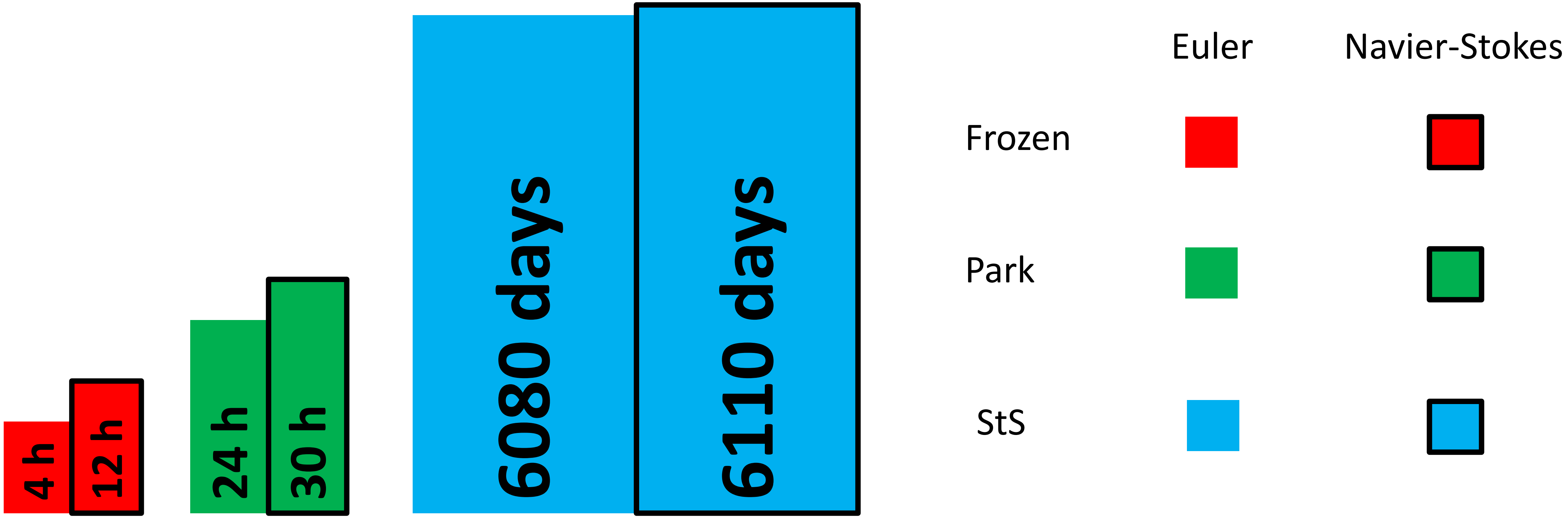
$$O_2(v)+N \leftrightarrow NO+O$$

$$N_2(v)+O \leftrightarrow NO+N$$

vTm/vTa:vibrational translational energy exchange with molecules/atoms;
vv: vibrational vibrational energy exchange
drm/dra: dissociation-recombination with molecules/atoms
LC: ladder climbing

Does StS need parallel computing? YES

Single core CPU computational time to complete a simulation for a hypersonic flow of a 5 species neutral air mixture over a sphere: 2D 512x256 mesh



Parallel computing approach

Multi-GPUs parallelization by using an MPI-CUDA approach

Why GPU for HPC? Why CUDA? Why Message Passing Interface (MPI)?

GPUs:

- **Many-core chips**
- **Huge amount of Flops**
- **High memory bandwidth**
- **High energy efficiency**

CUDA:

- the NVIDIA CUDA architecture was released in November 2006. It is not only a new hardware architecture but above all it provides a programming language (C / C ++ extension) that allows easy use of GPUs for general purpose computing

MPI:

- **It allows to scale the application across a multiple-nodes GPU cluster**

J. Sanders, E. Kandrot, CUDA by example, Addison-Wesley, New-York, 2011.

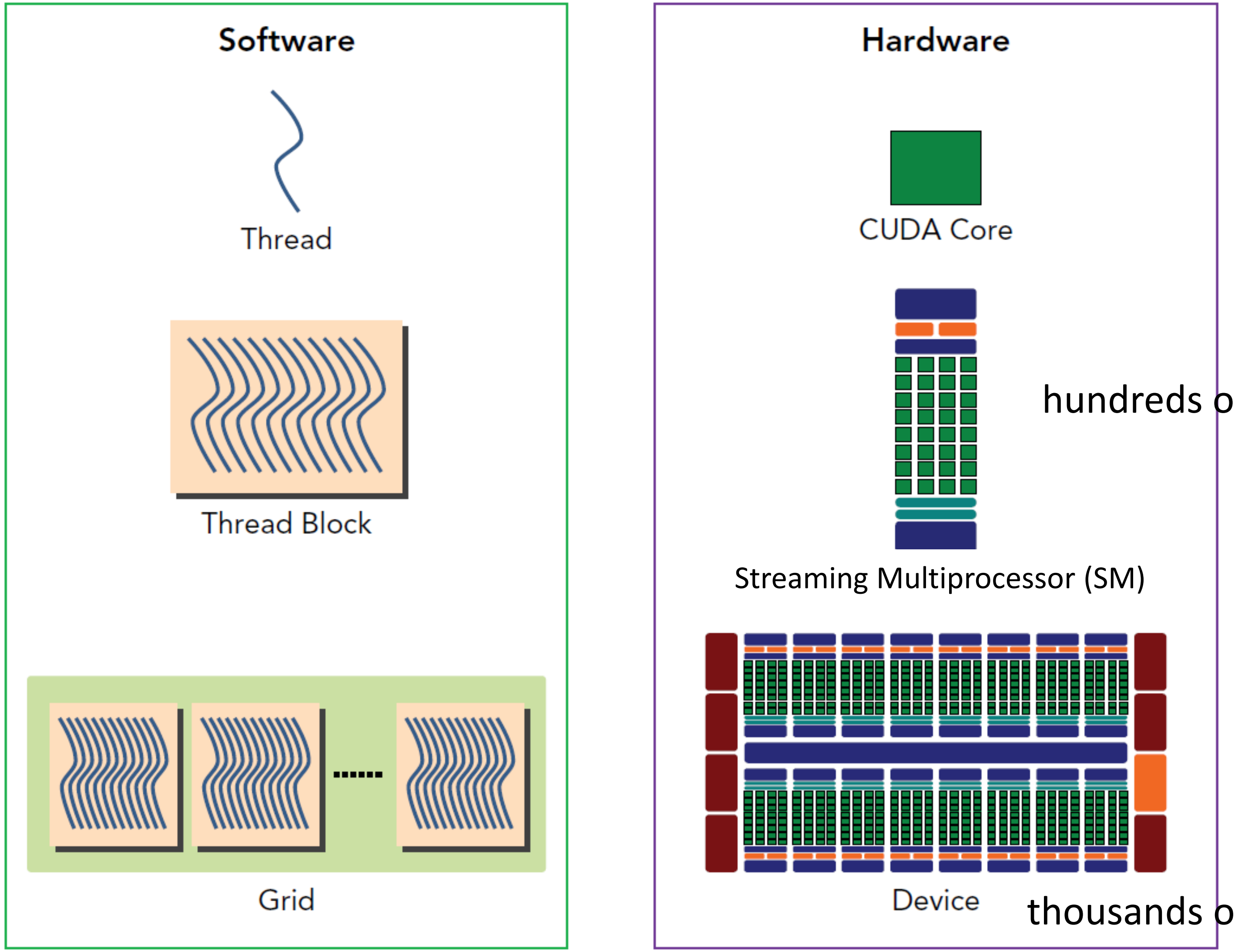
GPU vs CPU performance

	CPU 2016	NVIDIA Tesla P100
Theoretical GFLOP/s double precision	700	4700 - 5300
Theoretical Peak Memory Bandwidth GB/s	80	732
Theoretical GFLOP/s per Watt double precision	6	17.7 – 18.8

In the first 15 positions of the June 2017 green 500 list 13 clusters are powered with NVIDIA GPUs

CUDA C PROGRAMMING GUIDE PG-02829-001_v9.0 | September 2017
Tesla P100 | Data Sheet | Oct16
Tesla P100 PCle | Data Sheet | Oct 16
https://www.karlrupp.net/2013/06/cpu-gpu-and-mic-hardware-characteristics-over-time/
https://www.top500.org/green500/list/2017/06/

GPU



hundreds of threads

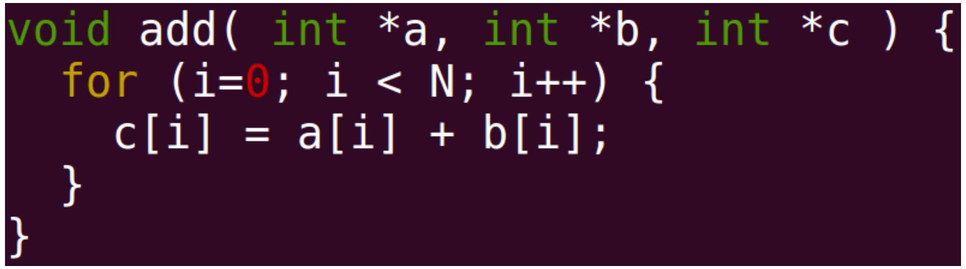
thousands of threads

Software implementation tries to be a mirror of the hardware structure

Figure taken from: J. Cheng ,M. Grossman ,T. McKercher, Professional CUDA® C Programming, John Wiley & Sons, Inc.

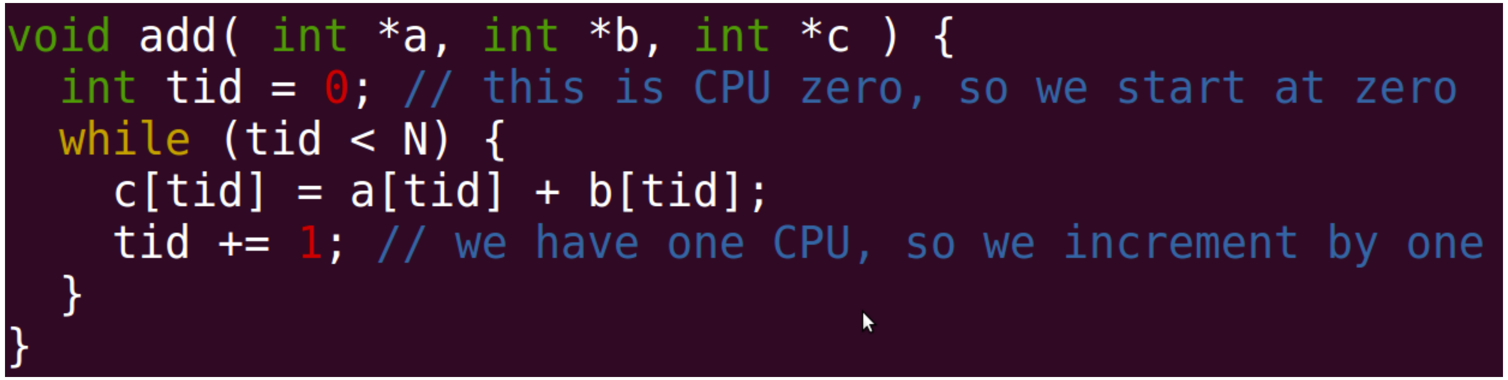
CUDA C parallel programming example: vectors sum

Task: sum vectors a and b (with N components) in a third vector c



```
void add( int *a, int *b, int *c ) {
  for (i=0; i < N; i++) {
    c[i] = a[i] + b[i];
  }
}
```

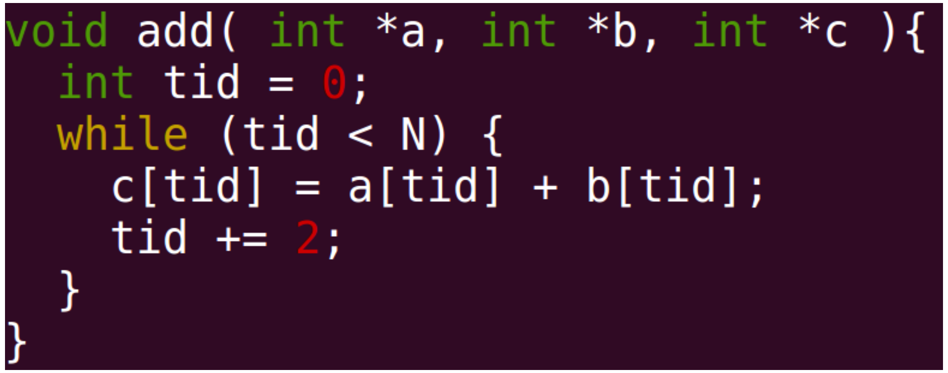
Serial CPU code



```
void add( int *a, int *b, int *c ) {
  int tid = 0; // this is CPU zero, so we start at zero
  while (tid < N) {
    c[tid] = a[tid] + b[tid];
    tid += 1; // we have one CPU, so we increment by one
  }
}
```

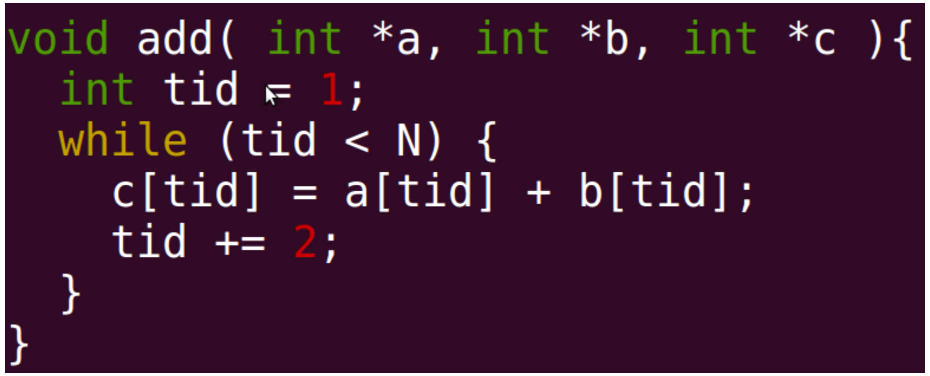
An easy trick to write a parallel code

CPU 1



```
void add( int *a, int *b, int *c ){
  int tid = 0;
  while (tid < N) {
    c[tid] = a[tid] + b[tid];
    tid += 2;
  }
}
```

CPU 2



```
void add( int *a, int *b, int *c ){
  int tid = 1;
  while (tid < N) {
    c[tid] = a[tid] + b[tid];
    tid += 2;
  }
}
```

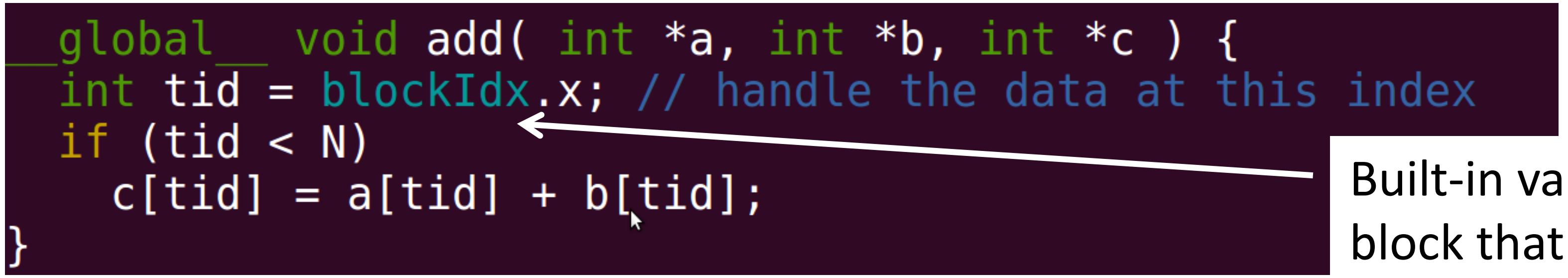
J. Sanders, E. Kandrot, CUDA by example, Addison-Wesley, New-York, 2011.

CUDA C parallel programming example: vectors sum

add<<N, 1>>(dev_a, dev_c, dev_d)

Blocks Threads per block

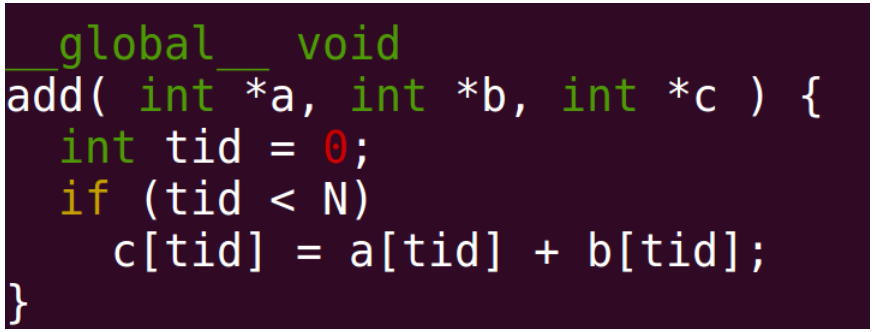
GPU kernel: N parallel blocks are launched



```
__global__ void add( int *a, int *b, int *c ) {
  int tid = blockIdx.x; // handle the data at this index
  if (tid < N)
    c[tid] = a[tid] + b[tid];
}
```

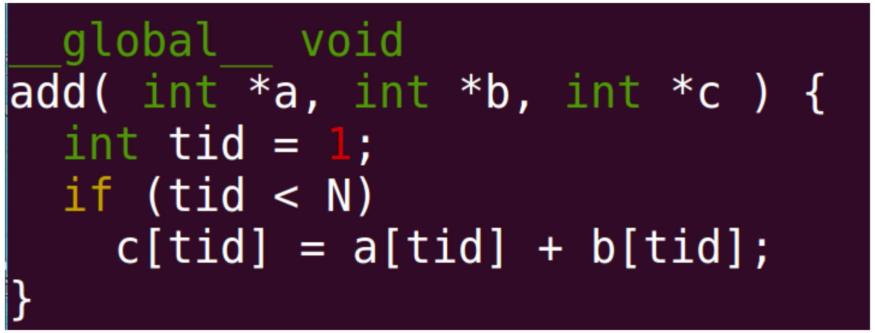
Built-in variable that gives the number of block that is running

BLOCK 1



```
__global__ void
add( int *a, int *b, int *c ) {
  int tid = 0;
  if (tid < N)
    c[tid] = a[tid] + b[tid];
}
```

BLOCK 2



```
__global__ void
add( int *a, int *b, int *c ) {
  int tid = 1;
  if (tid < N)
    c[tid] = a[tid] + b[tid];
}
```

this is what happens at runtime in the two blocks after the software substitutes the appropriate block index for blockIdx.x:

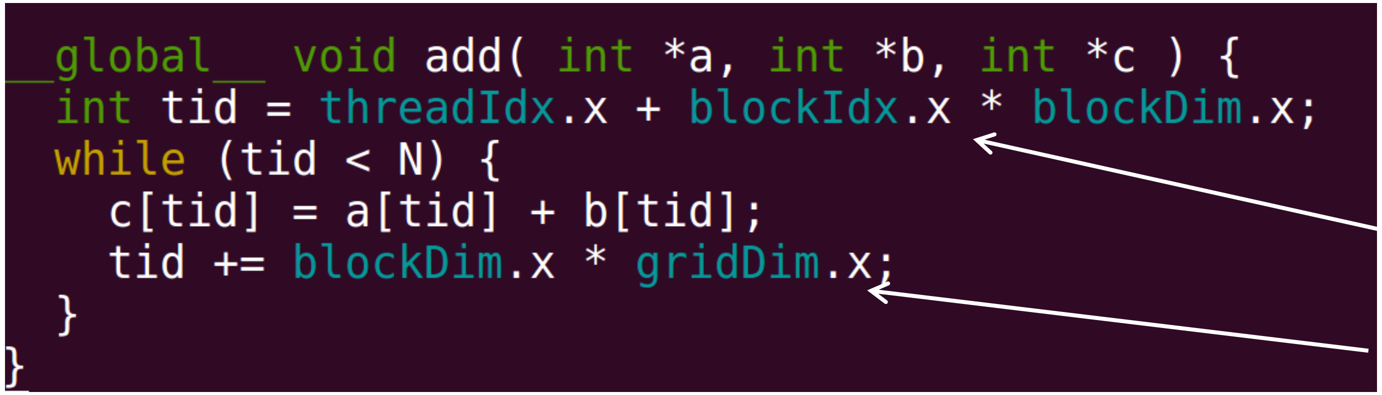
J. Sanders, E. Kandrot, CUDA by example, Addison-Wesley, New-York, 2011.

CUDA C parallel programming example: vectors sum

Splitting parallel blocks: needed to exploit all the GPU capacities

add<<B, T>>(dev_a, dev_c, dev_d)

BxT total number of threads; B blocks; T threads per block



```
__global__ void add( int *a, int *b, int *c ) {
  int tid = threadIdx.x + blockIdx.x * blockDim.x;
  while (tid < N) {
    c[tid] = a[tid] + b[tid];
    tid += blockDim.x * gridDim.x;
  }
}
```

Linear mapping

Needed if BxT<N

Block 0	Thread 0	Thread 1	Thread 2	Thread 3
Block 1	Thread 0	Thread 1	Thread 2	Thread 3

Example of 2 blocks with 4 threads per block:
blockDim.x=4
gridDim.x=2

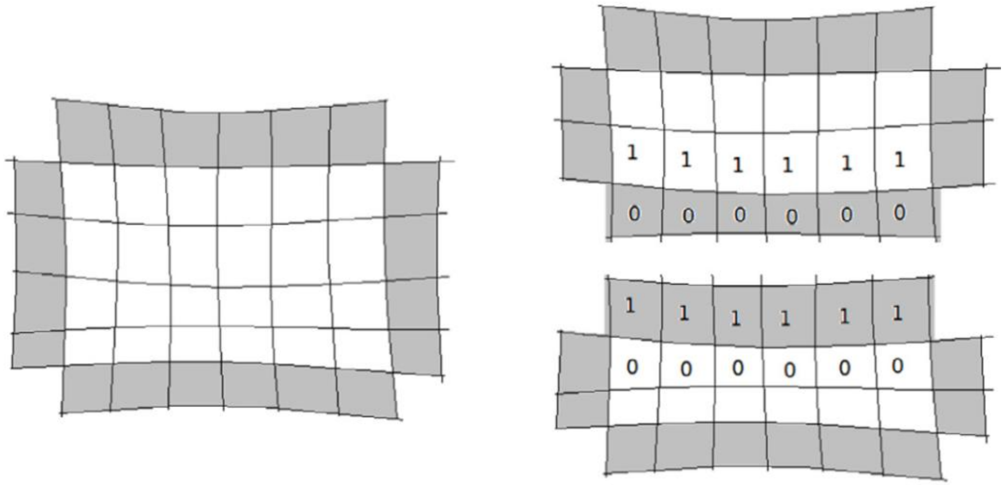
J. Sanders, E. Kandrot, CUDA by example, Addison-Wesley, New-York, 2011.

Multi-GPU: MPI-CUDA

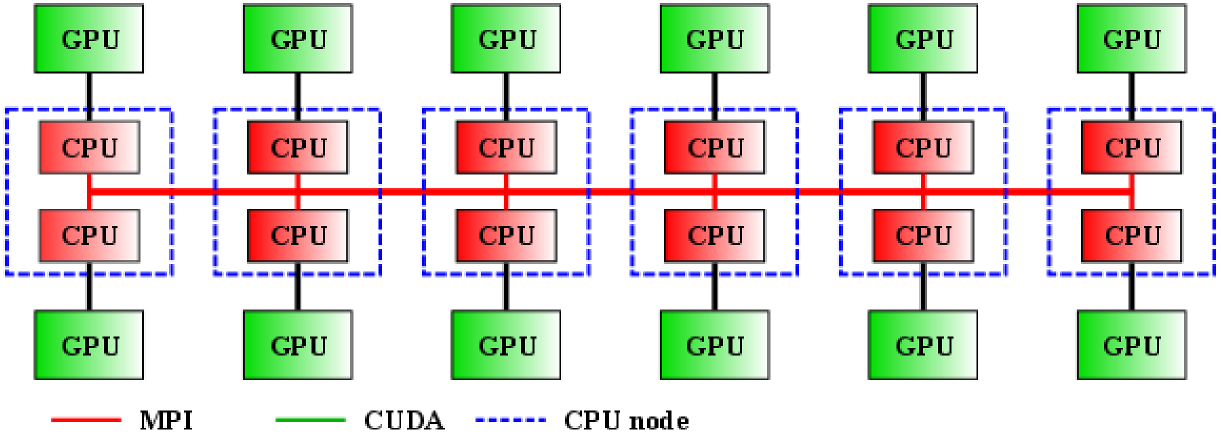
Initialize MPI environment	MPI_Init MPI_Comm_rank MPI_Comm_size
Creation of a 2D topology with neighbor relations	MPI_Cart_create MPI_Cart_coords MPI_Cart_shift

Associate each MPI process to a single GPU	cudaGetDeviceCount(&devCount)
	cudaSetDevice(myrank%devcount)

Creation of derived datatypes for transfers	MPI_Type_indexed MPI_Type_contiguous
Allocate arrays	malloc, cudaMalloc
Set input parameters on CPUs	init()
COPY arrays from CPUs to GPUs	cudaMemcpy
	START time integration loop
Frozen+kinetic step	generic_routines()
BC + data transfer	bc() cudaMemcpy MPI_Sendrecv cudaMemcpy
	END time integration loop



Classical domain decomposition approach

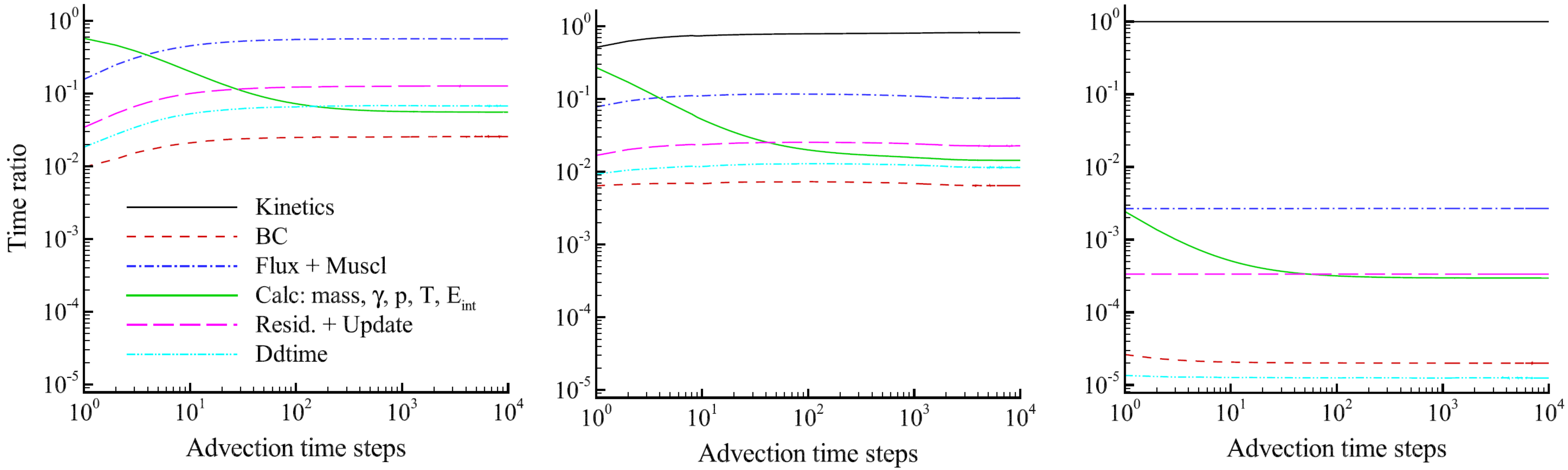


Poliba GPU cluster scheme

━ GPU-CPU communications

━ MPI infiniband communications among nodes

Code profiling (Euler eq.)



Frozen Park StS

Code profiling on a NVIDIA Tesla K40: Mach 6 **AIR** flow over a sphere; 256x128 computational cells; 4 chemical sub-step; 8 Gauss-Seidel inner iterations.

Time per iterations:

Frozen = 4.72*10^{-3} s

Park = 2.78*10^{-2} s

StS = 51.1 s

Iterations required for a full simulation 10000-20000 ---> 6-12 days for StS (for 512x256 cells 48-96 days)

F. Bonelli, M. Tuttafesta, G. Colonna, L. Cutrone, G. Pascazio, An MPI-CUDA approach for hypersonic flows with detailed state-to-state air kinetics using a GPU cluster, Comput. Phys. Comm., 219, pp. 178-195, 2017; M. Tuttafesta, G. Colonna, G. Pascazio, Comput. Phys. Comm. 184 (6) (2013) 1497–1510.

MPI-CUDA: GPU vs CPU computational performance

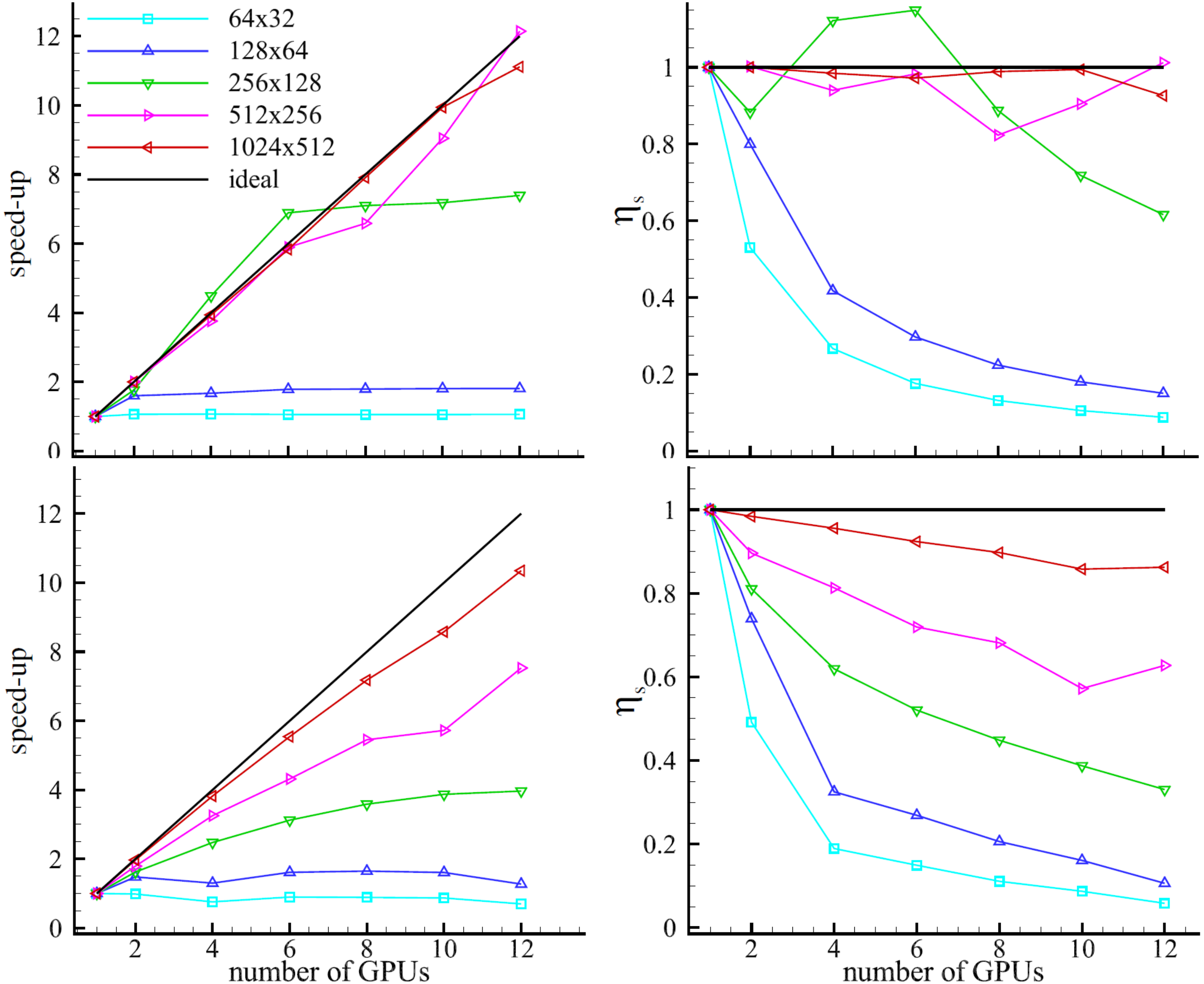
NVIDIA Tesla K40 (235 W) VS Intel Xeon CPU E5-2630 (6 cores) v2 2.60 GHz (80 W)

StS	Fluid cells	12 GPUs --Time per iteration (s) (Energy (J))	12 CPUs -- Time per iteration (s) (Energy(J))	Speed up (1 GPU vs 1 core)
	64x32	6.33 ($1.8*10^4$)	8.17 ($7.8*10^3$)	1.29 (7.7)
	128x64	6.36 ($1.8*10^4$)	26.71 ($2.56*10^4$)	4.2 (25.2)
	256x128	6.90 ($1.9*10^4$)	105.9 ($10.2*10^4$)	15.3 (91.8)
	512x256	15.91 ($4.5*10^4$)	419.5 ($40.3*10^4$)	26.4 (158.4)
	1024x512	68.72 ($19.4*10^4$)	1702.1 ($163.4*10^4$)	24.8 (148.8)
Park				
	64x32	$7.50*10^{-3}$ (21)	$1.59*10^{-3}$ (1.5)	0.21 (1.3)
	128x64	$7.77*10^{-3}$ (22)	$4.55*10^{-3}$ (4.3)	0.59 (3.5)
	256x128	$7.24*10^{-3}$ (20)	$1.68*10^{-2}$ (16)	2.32 (13.9)
	512x256	$1.36*10^{-2}$ (38)	$6.53*10^{-2}$ (63)	4.8 (28.8)
	1024x512	$3.48*10^{-2}$ (98)	$2.46*10^{-1}$ (236)	7.1 (42.6)

MPI-CUDA strong scaling

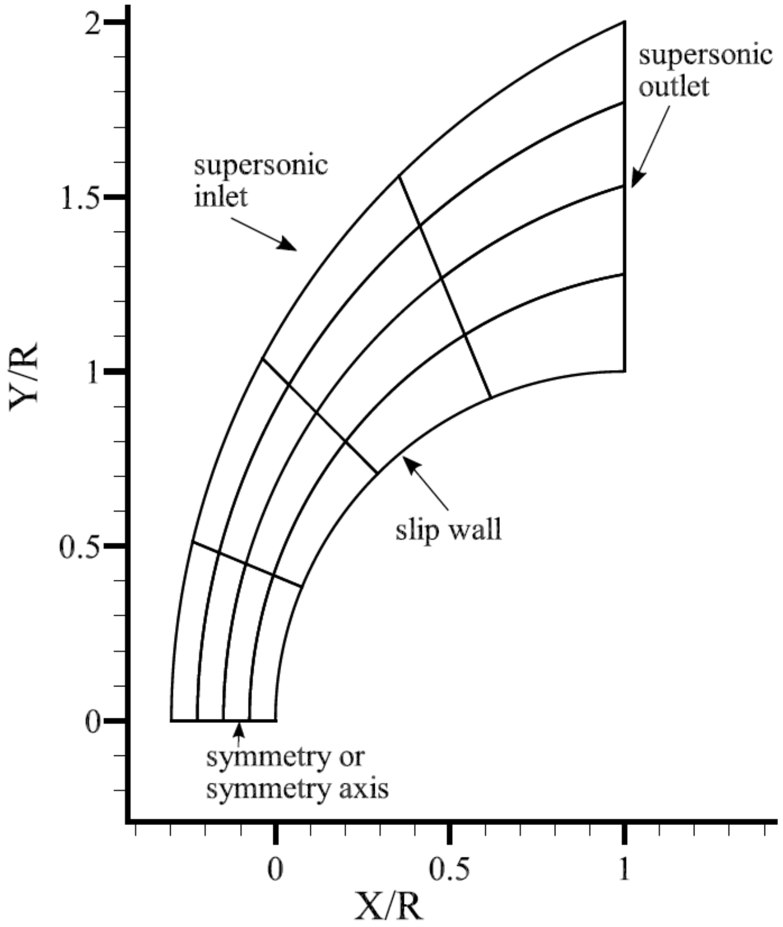
StS

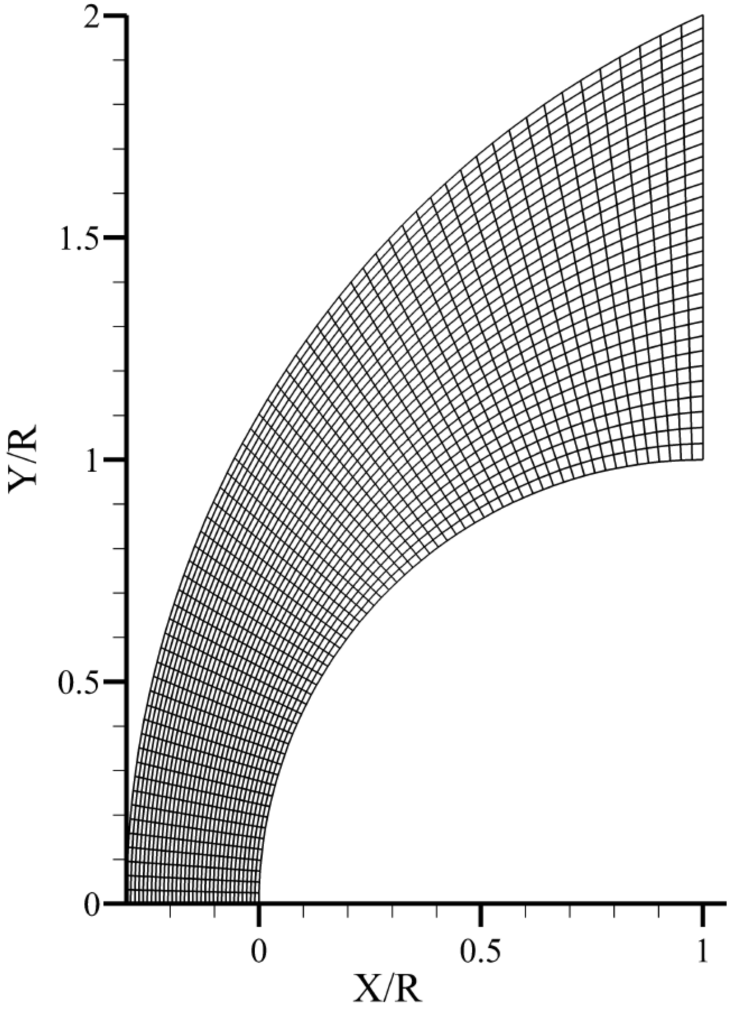
Park



F. Bonelli, M. Tuttafesta, G. Colonna, L. Cutrone, G. Pascazio, *An MPI-CUDA approach for hypersonic flows with detailed state-to-state air kinetics using a GPU cluster*, Computer Physics Communications, 219, pp. 178-195, 2017

Flow past a sphere: Nonaka[4] test case (Euler eqs.)





[4]R= 7mm;

u_∞=3490 m/s

T_∞=293 K

P_∞= 4825 Pa

Y_{N2}=0.767

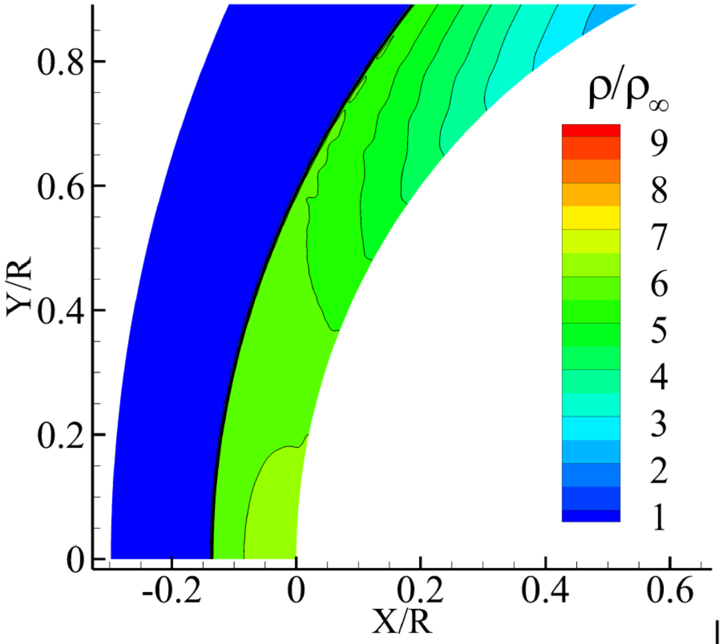
Y_{O2}=0.233

Computational domain, with an example of 4 x4 MPI partitioning, along with boundary conditions (left). 228x392 computational grid shown every 10 grid points (right).

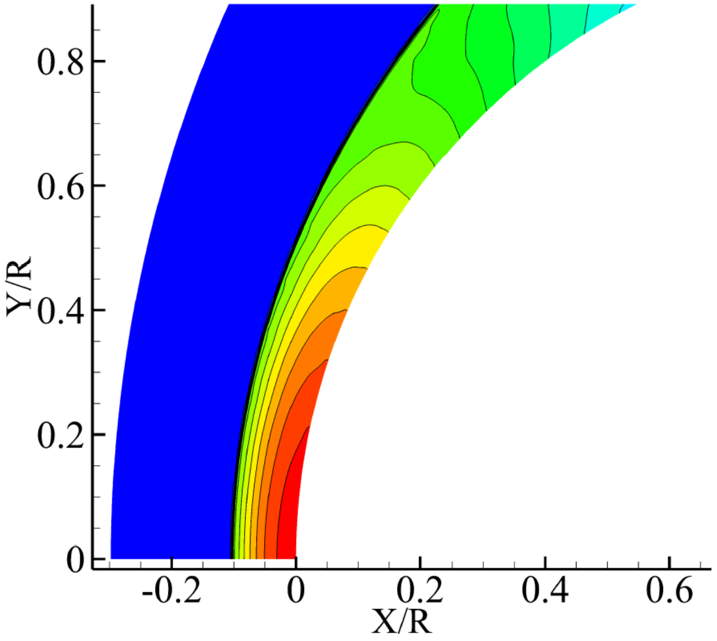
[4]S. Nonaka et al. ,JTHT 14 (2), pp. 225-229, 2000

Nonaka[4] test case (Euler eqs.)

F. Bonelli, M. Tuttafesta, G. Colonna, L. Cutrone, G. Pascazio, *An MPI-CUDA approach for hypersonic flows with detailed state-to-state air kinetics using a GPU cluster*, Computer Physics Communications, 219, pp. 178-195, 2017

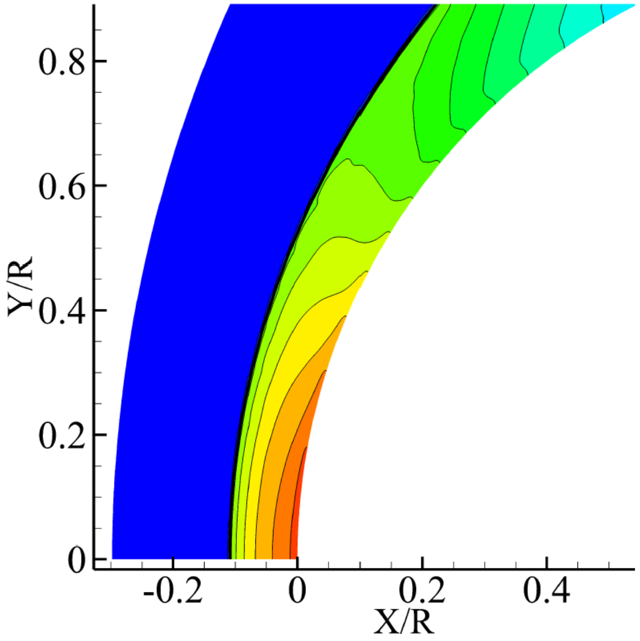


Frozen

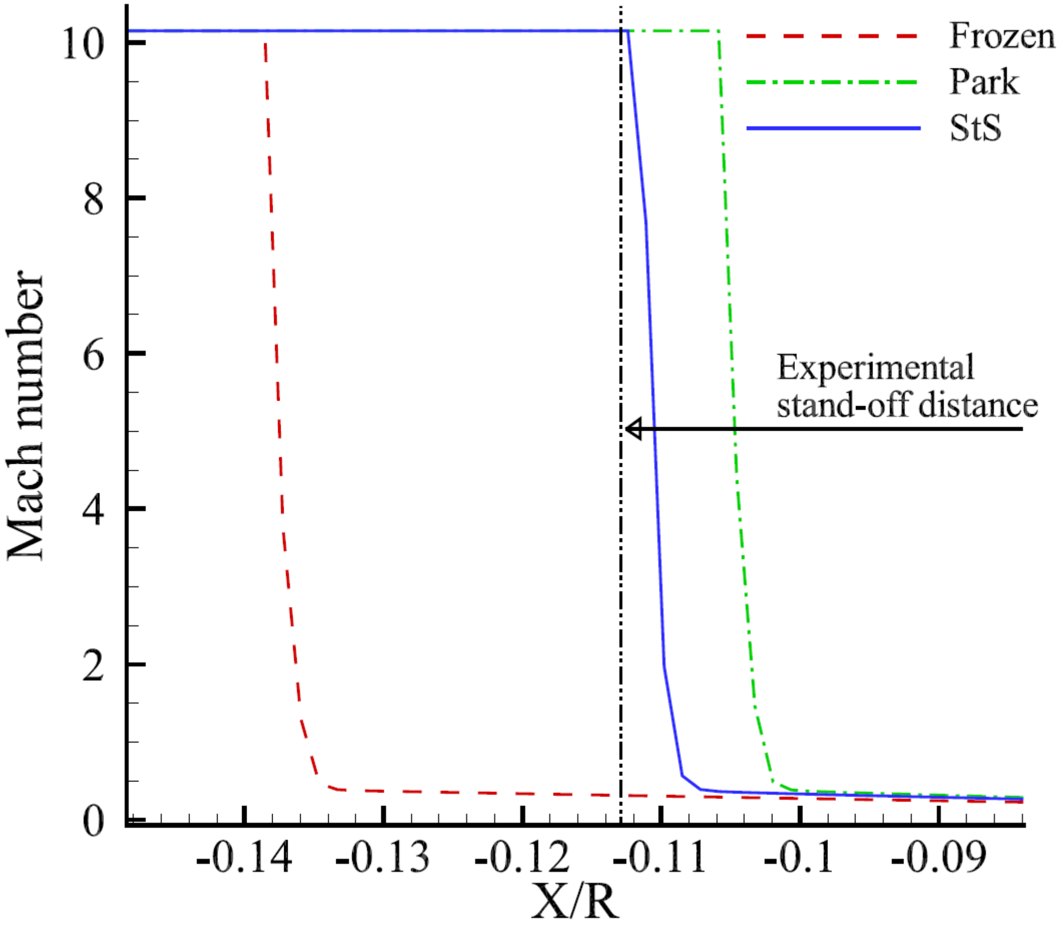


Park

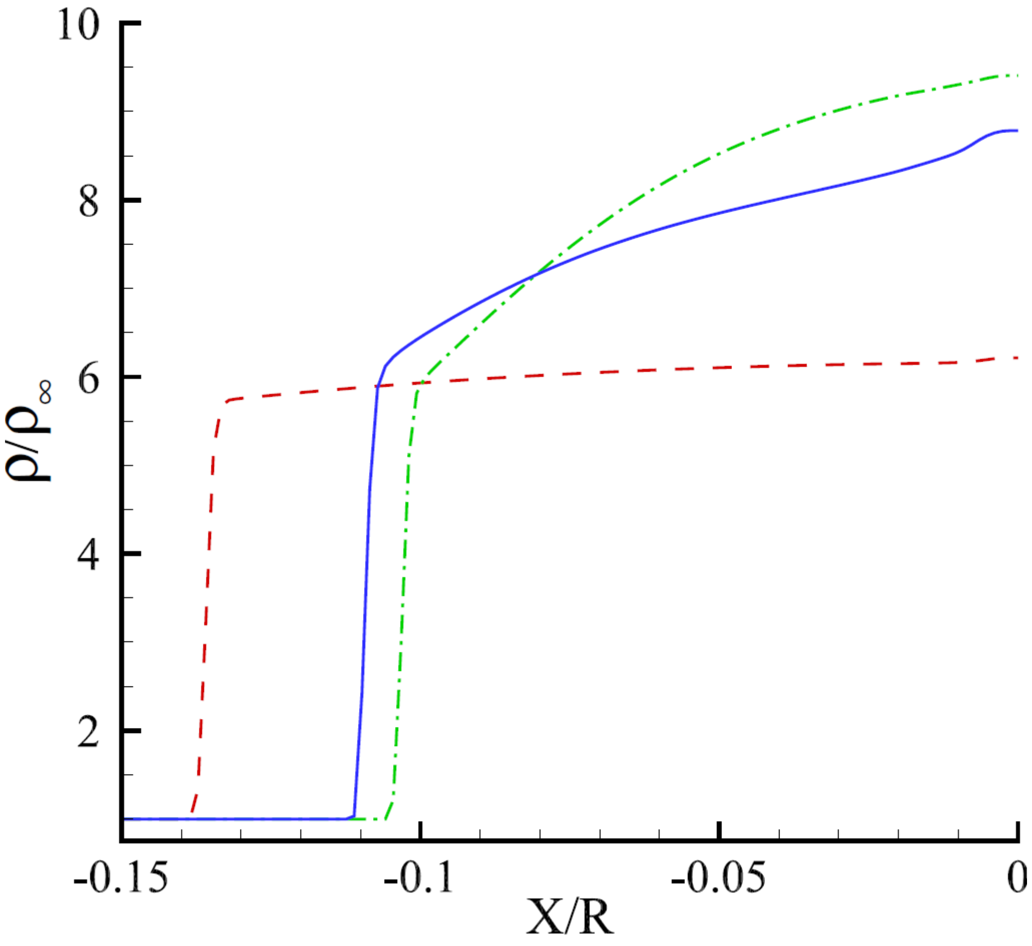
StS



Nonaka[4] test case (Euler eqs.): stagnation line profiles



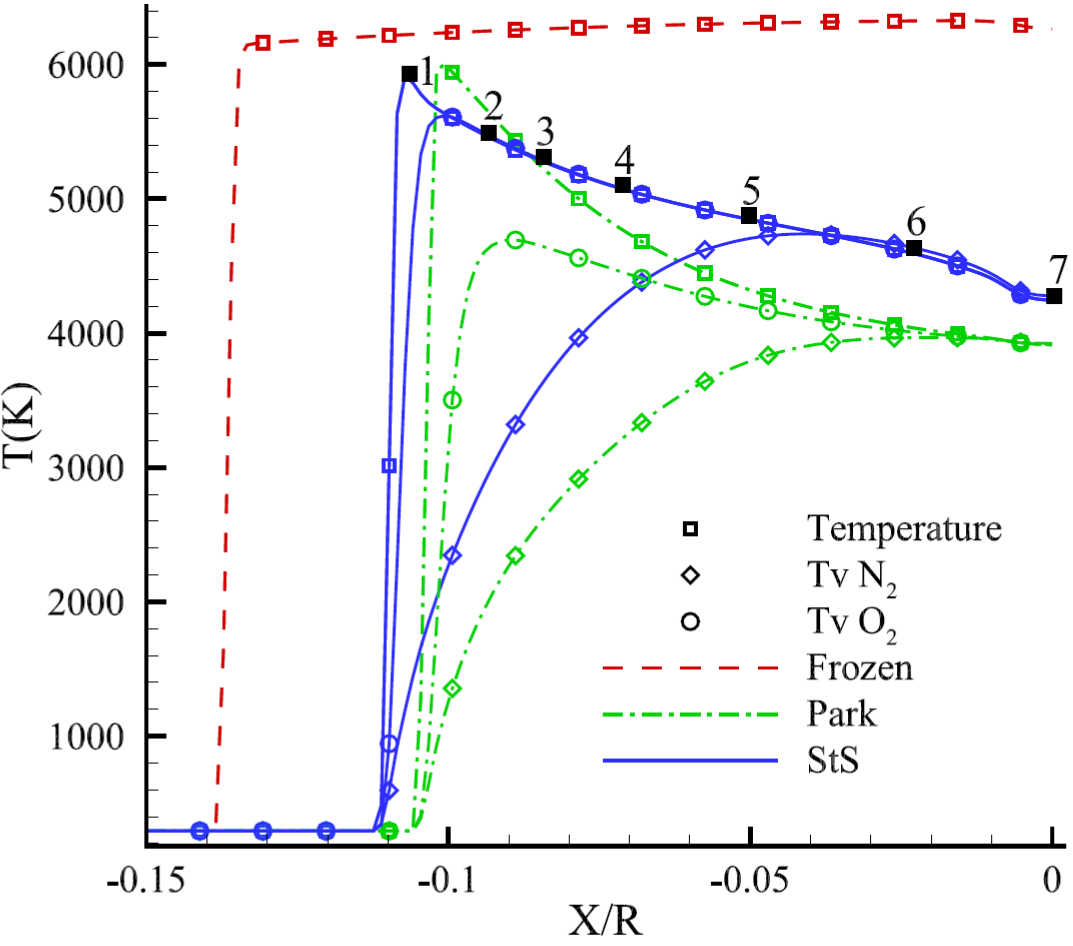
Mach Number



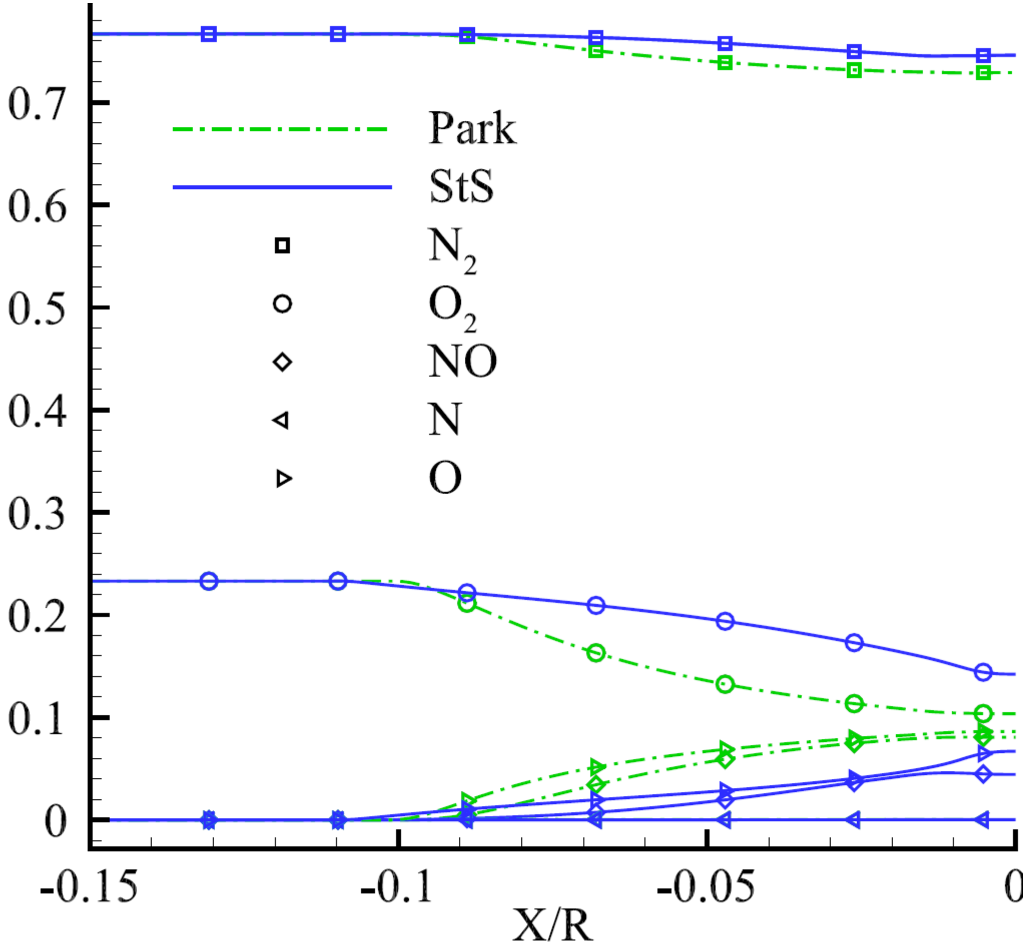
Normalized density

F. Bonelli, M. Tuttafesta, G. Colonna, L. Cutrone, G. Pascazio, An MPI-CUDA approach for hypersonic flows with detailed state-to-state air kinetics using a GPU cluster, Computer Physics Communications, 219, pp. 178-195, 2017

Nonaka[4] test case (Euler eqs.): stagnation line profiles



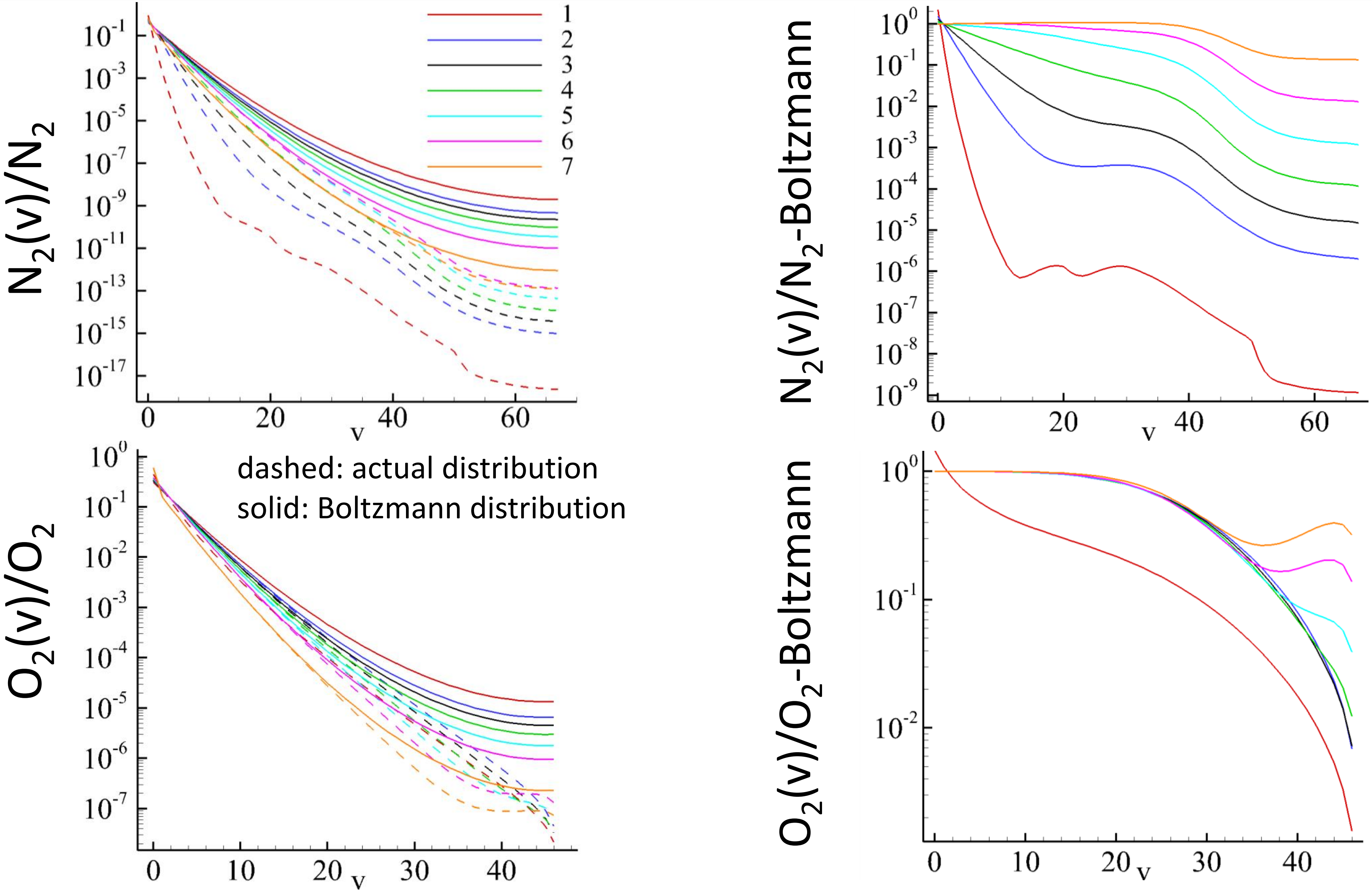
Temperatures



Mass fractions

F. Bonelli, M. Tuttafesta, G. Colonna, L. Cutrone, G. Pascazio, An MPI-CUDA approach for hypersonic flows with detailed state-to-state air kinetics using a GPU cluster, Computer Physics Communications, 219, pp. 178-195, 2017

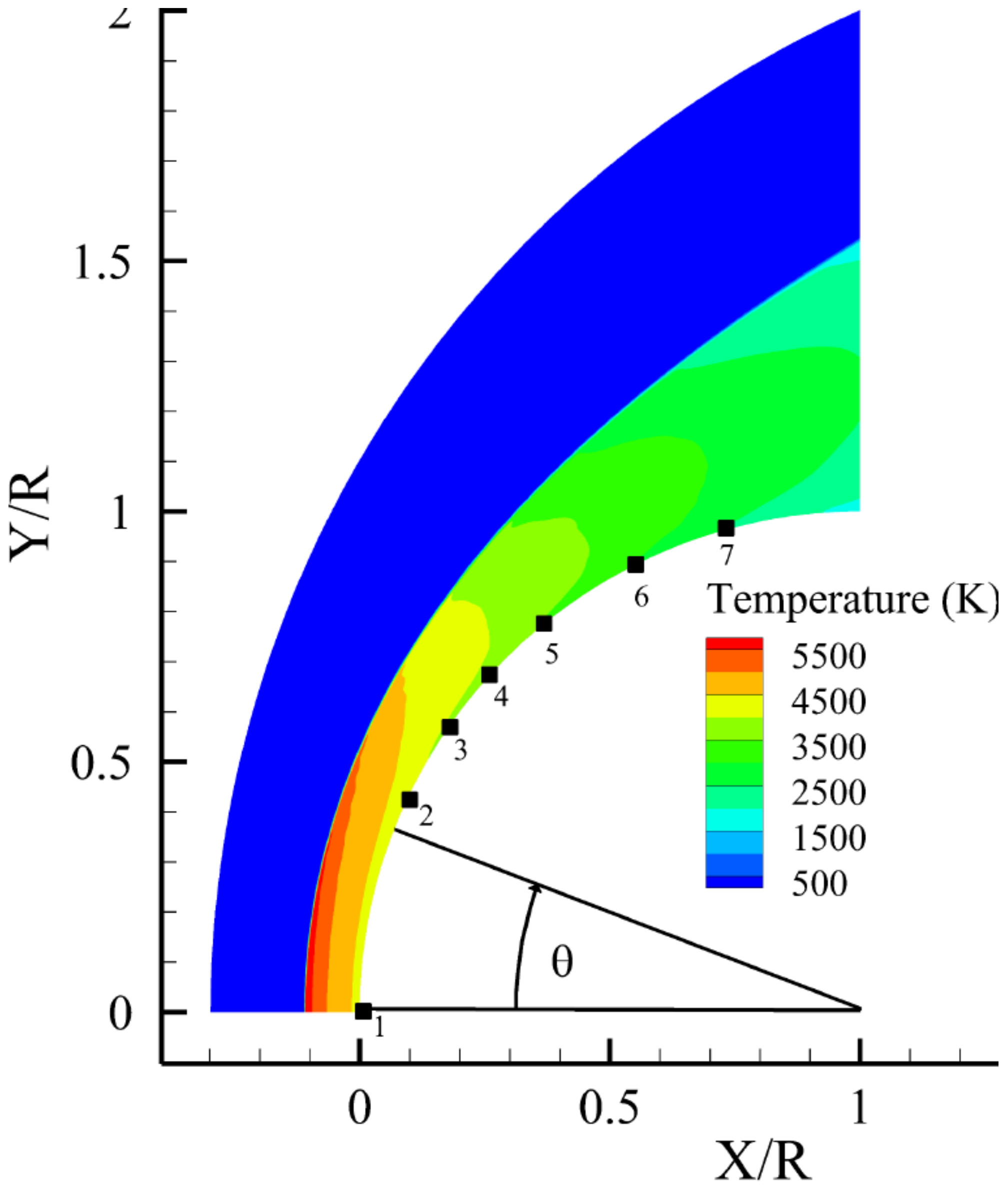
Nonaka[4] test case (Euler eqs.): vibrational distributions along stagnation line



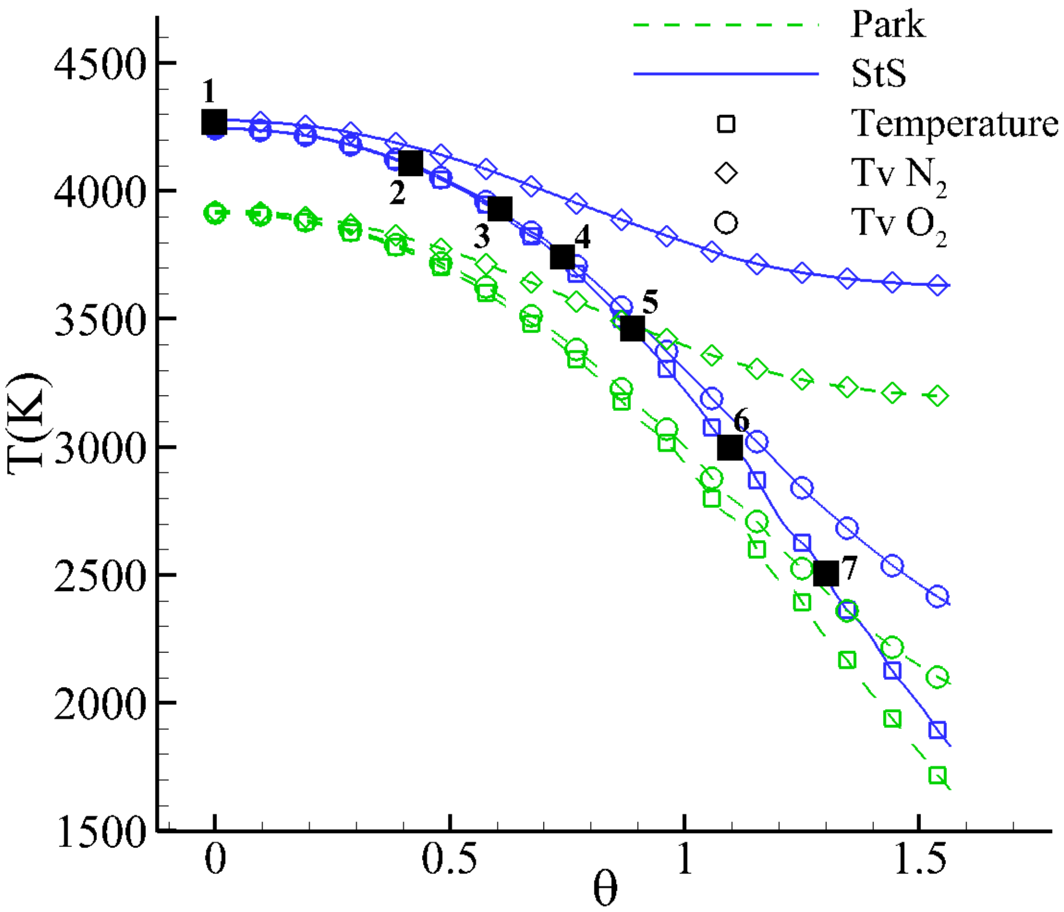
dashed: actual distribution

solid: Boltzmann distribution

Nonaka[4] test case (Euler eqs.): temperature wall line profiles

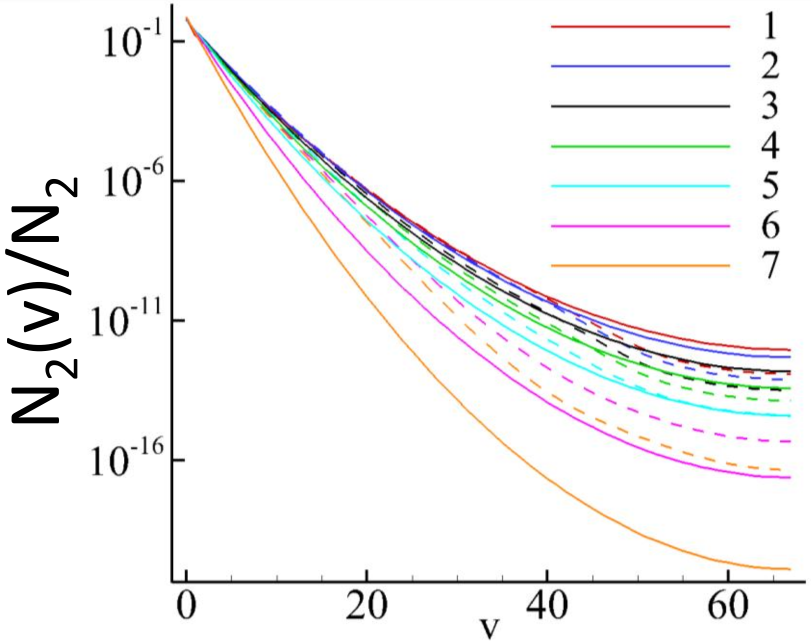


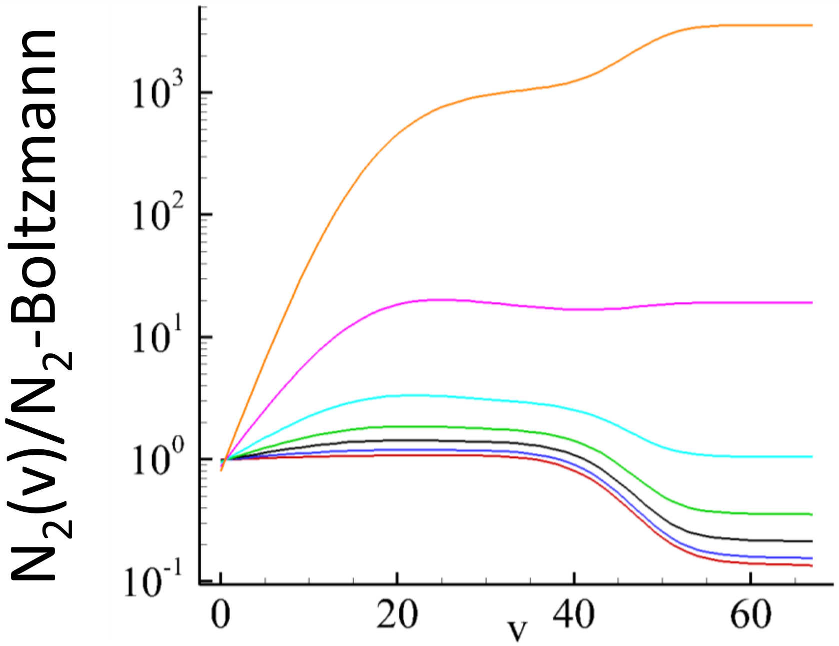
Translational temperature



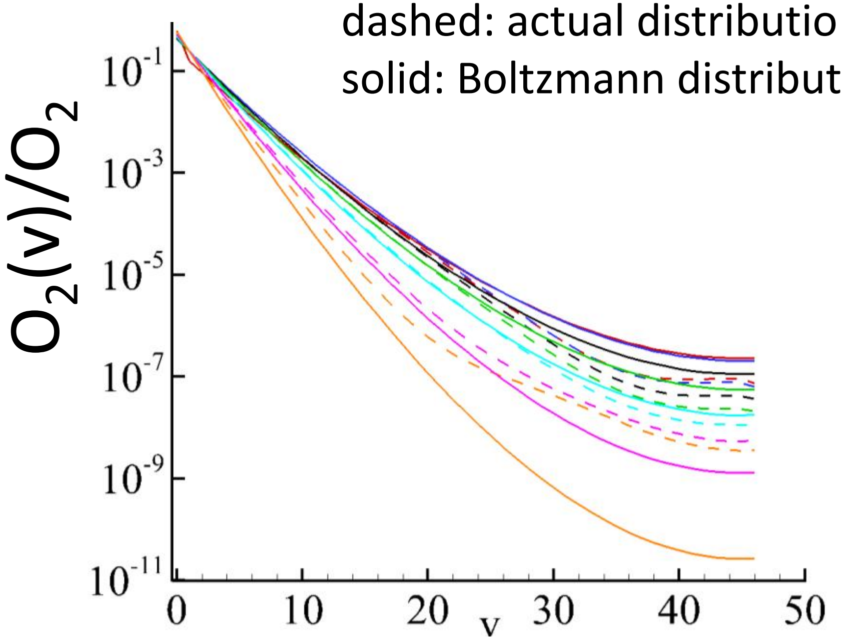
Temperature profiles along the wall

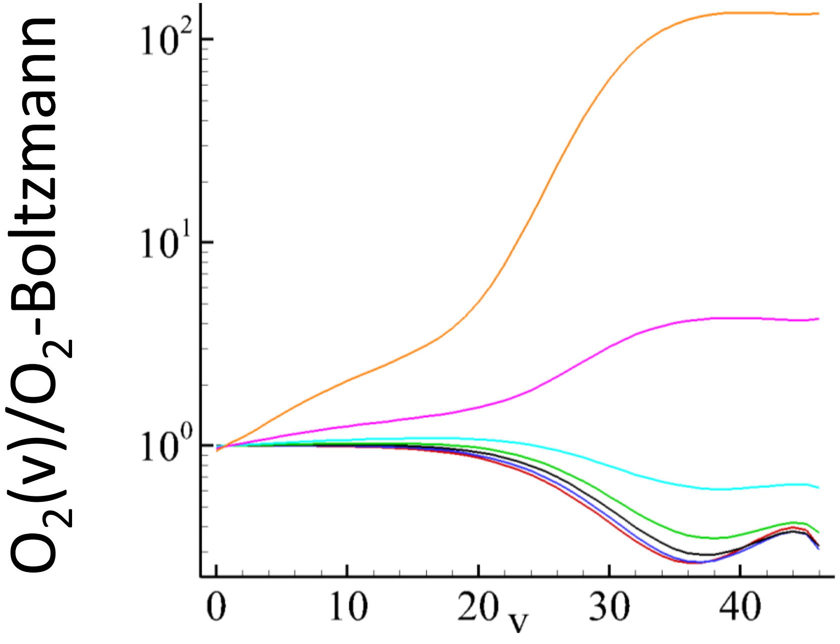
Nonaka[4] test case (Euler eqs.): vibrational distributions wall profiles



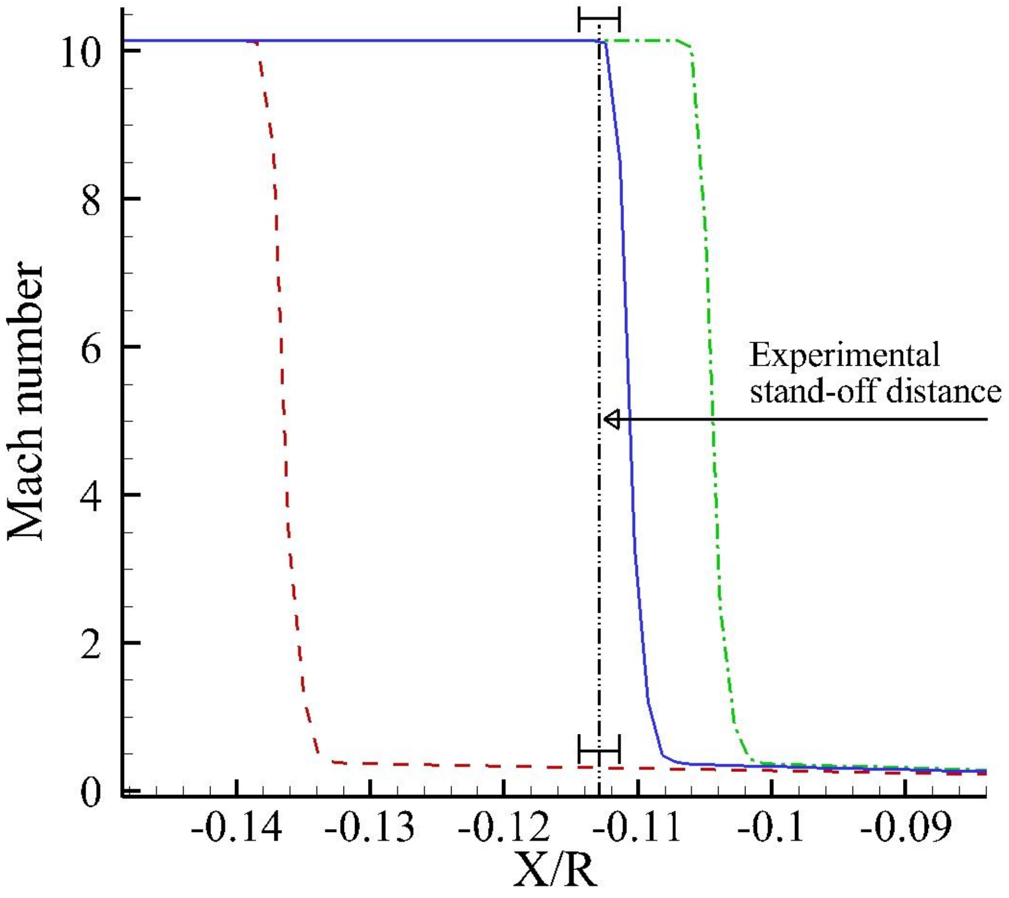


dashed: actual distribution
solid: Boltzmann distribution

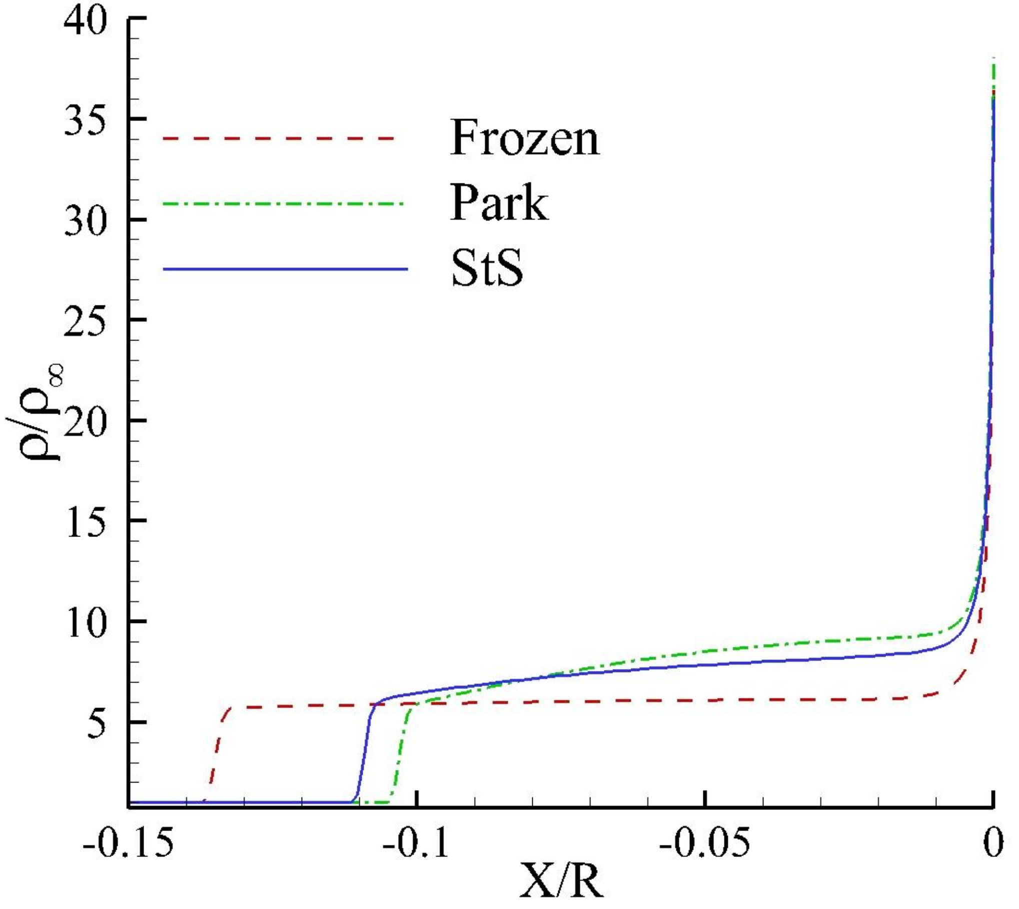




Nonaka[4] test case (Navier-Stokes): stagnation line profiles



Mach Number



Normalized density

Conclusions

- We developed an efficient multi-GPU code for two-dimensional fluid dynamics
- A second-order accurate finite-volume space discretization scheme has been used, in conjunction with an explicit Runge-Kutta time integration scheme and an operator-splitting approach with implicit chemical source term treatment
- We demonstrated the accuracy and the feasibility of fluid dynamic computations of thermochemical non-equilibrium flows by means of detailed state-to-state (StS) vibrationally resolved air kinetics
- The MPI-CUDA approach allowed us to efficiently scale the code across a multiple-nodes GPU cluster with good scalability performance: comparing the single GPU against the single core CPU performance speed-up values up to 150 were found.

Current and future work

- Extensive validation of the Navier-Stokes solver with StS model;
- Extension to 3D with Immersed Boundary method
- Introduction of ionized species
- Flow-wall boundary treatment: models for catalysis and ablation