# Advanced Use of Eilmer for Hypersonic Flow Simulation

Rowan Gollan and Peter Jacobs

Day 2: 10 May 2018

The University of Queensland

# Short Course Outline, Day 2

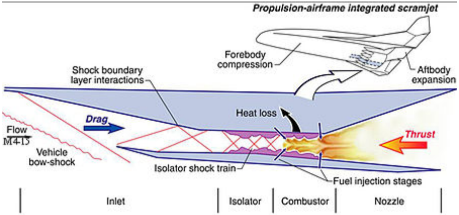
## Morning: presentation & discussion

- Eilmer for simulation of hypersonic flows
- Advanced thermochemistry
- Turbulence modelling
- Parallel computing: small & large scale
- Advanced/experimental features
    - user-defined BCs and source terms
    - moving grid
    - shock-fitting boundaries
    - block-marching mode
    - wall functions for turbulence
    - GPU-acceleration of reacting flows
    - steady-state solver
    - adjoint solver for optimisation
    - state-specific chemistry coupled to flow

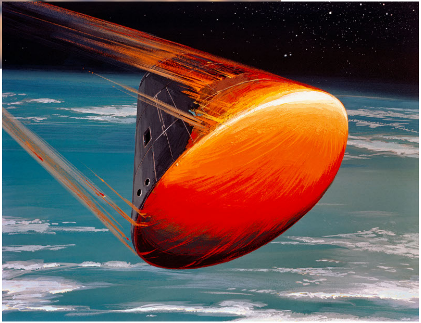
## Afternoon: hands-on activities

- Installing MPI version of Eilmer
- Working in 3D
    - importing grids from 3rd-party grid generation tools
    - partitioning structured and unstructured grids
- Fourth example: Reacting air flow over a sphere
- Fifth example: shock-wave boundary-layer interaction
- Tips & tricks:
    - running simulations in parallel
    - using the block-marching mode
- Bring-your-own examples

# ~~Why~~ How hypersonics?



Source: LaRC



# How hypersonics?

Conservation of mass:

$$\frac{\partial}{\partial t}\rho + \nabla \cdot \rho \mathbf{u} = 0 \tag{1}$$

Conservation of species mass:

$$\frac{\partial}{\partial t}\rho_i + \nabla \cdot \rho_i \mathbf{u} = -\left(\nabla \cdot \mathbf{J}_i\right) + \dot{\omega}_i \tag{2}$$

Conservation of momentum:

$$\frac{\partial}{\partial t}\rho \mathbf{u} + \nabla \cdot \rho \mathbf{u}\mathbf{u} = -\nabla p - \nabla \cdot \left\{-\mu(\nabla \mathbf{u} + (\nabla \mathbf{u})^{\dagger}) + \tfrac{2}{3}\mu(\nabla \cdot \mathbf{u})\boldsymbol{\delta}\right\} \tag{3}$$

Conservation of total energy:

$$\begin{aligned}\frac{\partial}{\partial t}\rho E + \nabla \cdot (e + \frac{p}{\rho})\mathbf{u} = \nabla \cdot [k\nabla T + \sum_{s=1}^{N_v} k_{v,s}\nabla T_{v,s}] + \nabla \cdot \left[\sum_{i=1}^{N_s} h_i \mathbf{J}_i\right] \\ - \left(\nabla \cdot \left[\left\{-\mu(\nabla \mathbf{u} + (\nabla \mathbf{u})^{\dagger}) + \tfrac{2}{3}\mu(\nabla \cdot \mathbf{u})\boldsymbol{\delta}\right\} \cdot \mathbf{u}\right]\right) - Q_{\text{rad}}\end{aligned} \tag{4}$$

Conservation of vibrational energy:

$$\frac{\partial}{\partial t}\rho_i e_{v,i} + \nabla \cdot \rho_i e_{v,i}\mathbf{u} = \nabla \cdot [k_{v,i}\nabla T_{v,i}] - \nabla \cdot e_{v,i}\mathbf{J}_i + Q_{T-V_i} + Q_{V-V_i} + Q_{\text{Chem}-V_i} - Q_{\text{rad}_i} \tag{5}$$

# Flows with chemical nonequilibrium

- Flow can be distinguished (and modelled) in three classifications:
  1. frozen flow (ideal gas model) – $t_{chem} \gg t_{flow}$
  2. nonequilbrium flow (finite-rate chemistry) – $t_{chem} \approx t_{flow}$
  3. equilibrium flow – $t_{chem} \ll t_{flow}$
- Model for rate of chemical change: Law of Mass Action

$$\frac{d[X_i]}{dt} = (\nu_i^{''} - \nu_i^{'}) \left\{ k_f \prod_i [X_i]^{\nu_i^{'}} - k_b \prod_i [X_i]^{\nu_i^{''}} \right\}$$

- Forward and reverse rates are computed from a modified Arrhenius form

$$k = AT^n e^{-T_a/T}$$

  alternatively, reverse rates may come from computing the equilibrium constant

- This ODE integrators available are:
  - Mott's $\alpha$ Quasi-Steady-State approach (for stiff systems, eg. combustion)
  - Runge-Kutta-Fehlberg (efficient for non-stiff systems, eg. endothermic chemistry)

# Chemistry input file

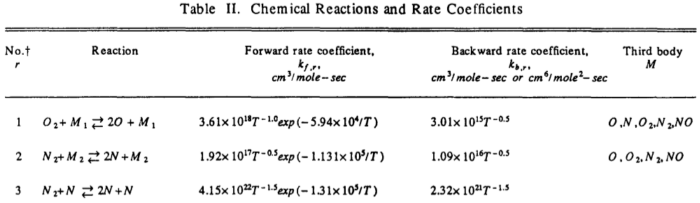


Table II. Chemical Reactions and Rate Coefficients

| No.† $r$ | Reaction | Forward rate coefficient, $k_{f,r}$, $cm^3/mole-sec$ | Backward rate coefficient, $k_{b,r}$, $cm^3/mole-sec$ or $cm^6/mole^2-sec$ | Third body $M$ |
|---|---|---|---|---|
| 1 | $O_2 + M_1 \rightleftarrows 2O + M_1$ | $3.61\times10^{18}T^{-1.0}exp(-5.94\times10^4/T)$ | $3.01\times10^{15}T^{-0.5}$ | $O, N, O_2, N_2, NO$ |
| 2 | $N_2 + M_2 \rightleftarrows 2N + M_2$ | $1.92\times10^{17}T^{-0.5}exp(-1.131\times10^5/T)$ | $1.09\times10^{16}T^{-0.5}$ | $O, O_2, N_2, NO$ |
| 3 | $N_2 + N \rightleftarrows 2N + N$ | $4.15\times10^{22}T^{-1.5}exp(-1.31\times10^5/T)$ | $2.32\times10^{21}T^{-1.5}$ | |

An extract of rate constants. **Source:** Gupta et al. (1990)

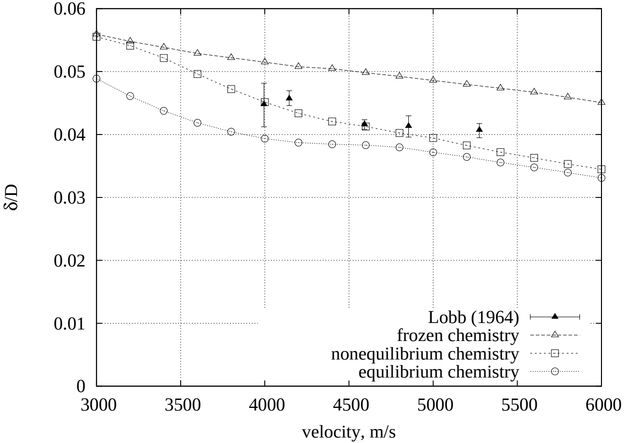
Eilmer input is designed to follow the typical layout of rate constants in the literature.[1]

```
Reaction{
   'O2 + M <=> O + O + M',
   fr={'Arrhenius', A=3.610e+18, n=-1.00, C=59400.00},
   br={'Arrhenius', A=3.010e+15, n=-0.50, C=0.0},
   label='r1',
   efficiencies={O2=9.0, N2=2.0, O=25.0, N=1.0, NO=1.0}
}
```

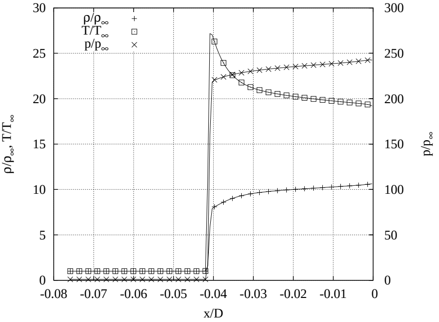
[1] And we provide a `chemkin2eilmer` converter tool.

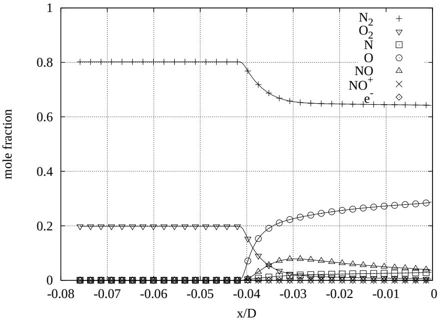
# Validation of reacting flow model for air

- Lobb (1964) performed a series of experiments to measure shock detachment distance on spheres fired into air at hypervelocities
- A 0.5 in nylon sphere was fired in a ballistic range into air at various speeds and pressures



# Stagnation streamline properties for reacting air over sphere





# Flows with thermal nonequilibrium

| molecules: | atoms: |
|---|---|
| translation | translation |
| rotation | |
| vibration | |
| electronic | electronic |

*Why isn't the vibrational energy in equilibrium?*

- it typically takes 5-10 collisions for the rotational mode to equilibriate at the translational temperature
- however, it can take on the order of $10^2$-$10^3$ collisions for vibrational modes to equilibriate

*A specific example...*

For a blunt body of nose radius 1 m with a velocity of 10 km/s at an altitude of 75 km:

$$\tau_{flow} \approx 10^{-4}s, \tau_{chem} \approx 10^{-3}s, \tau_{vib} \approx 10^{-5}$$

Thivet, Perrin and Candel (1991)

# Engineering model for flows with thermal nonequilibrium

$$\frac{\partial}{\partial t}\rho_i e_{v,i} + \nabla \cdot \rho_i e_{v,i}\mathbf{u} = \nabla \cdot [k_{v,i} \nabla T_{v,i}] - \nabla \cdot e_{v,i}\mathbf{J}_i + Q_{T-V_i} + Q_{V-V_i} + Q_{\text{Chem}-V_i} - Q_{\text{rad}_i}$$

- Internal energy modes are separable (Born-Oppenheimer approximation)
- Each separated mode is assumed to populate a Boltzmann distribution at a particular temperature
- Nonequilibrium occurs when these governing temperatures differ for different energy modes
- Depending on extent of nonequilbrium, each molecular species might have its own describing vibrational temperature
- Example: four-temperature air model has one temperature to describe tranlsational/rotational energy and one temperature each for vibrational energy of $N_2$, $O_2$ and NO

# Thermal nonequilibrium: energy exchange

$$\frac{\partial}{\partial t}\rho_i e_{v,i}+\nabla\cdot\rho_i e_{v,i}\mathbf{u}=\nabla\cdot[k_{v,i}\nabla T_{v,i}]-\nabla\cdot e_{v,i}\mathbf{J}_i+Q_{T-V_i}+Q_{V-V_i}+Q_{\mathrm{Chem}-V_i}-Q_{\mathrm{rad}_i}$$

*Translational-vibrational energy exchange*

$$Q_{T-V_i}=\sum_{c=1}^{n_s}x_c\frac{e^*_{v,p}-e_{v,p}}{(\tau_v^{p-c})_{V-T}}$$

*Vibrational-vibrational energy exchange*

$$Q_{V-V_i}=\sum_{q=1,q\neq p}^{n_v}\frac{x_q}{(\tau_v^{p-q})_{V-V}}\left(\frac{1-\exp(-\Theta_{v,p}/T)}{1-\exp(-\Theta_{v,q}/T)}\frac{e_{v,q}}{\hat{e}_{v,q}}(\bar{e}_{v,p}-e_{v,p})-\frac{e_{v,p}}{\hat{e}_{v,q}}(\bar{e}_{v,q}-e_{v,q})\right)$$

# Thermal nonequilibrium: chemistry-energy coupling

$$\frac{\partial}{\partial t}\rho_i e_{v,i} + \nabla\cdot\rho_i e_{v,i}\mathbf{u} = \nabla\cdot[k_{v,i}\nabla T_{v,i}] - \nabla\cdot e_{v,i}\mathbf{J}_i + Q_{T-V_i} + Q_{V-V_i} + Q_{\mathrm{Chem}-V_i} - Q_{\mathrm{rad}_i}$$

Chemistry-vibration coupling

- If the gas is in nonequilibrium, how are the reaction rates affected?

$$k = k^{EQ}(T)\Psi(T, T_{v,i})$$

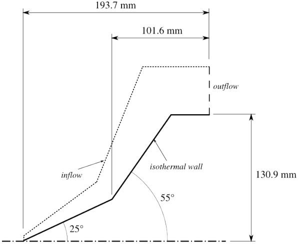
- When chemical reactions take place, how does this effect the average vibrational energy?
- A preferential model is implemented in Eilmer by Knab et al. This is a generalised and extended version of Treanor & Marrone's preferential dissociation model.

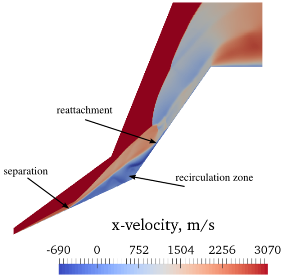
$$Q_{\mathrm{Chem}-V_i} = \sum_{j=1}^{Nr}\left[-G_{va,ij}\frac{1}{f_i}\left(\frac{df_i}{dt}\right)_f + G_{app,ij}\frac{1}{f_i}\left(\frac{df_i}{dt}\right)_r\right]$$

# Vibrational nonequilibrium validation: double cone experiments

CUBRC Run 80

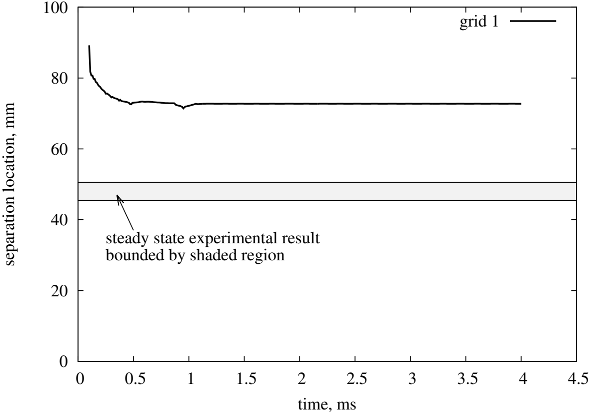
| Quantity | Value |
|---|---|
| $H_0$ (MJ/kg) | 5.28 |
| $p_0$ (MPa) | 8.48 |
| $T_0$ (K) | 4269 |
| $u_\infty$ (m/s) | 3066 |
| $p_\infty$ (Pa) | 63.6 |
| $\rho_\infty$ (kg/m$^3$) | $1.29 \times 10^{-3}$ |
| $T_\infty$ (K) | 166 |
| $T_{v,\infty}$ (K) | 2711 |
| mass fraction of $N_2$ | 0.9999 |
| mass fraction of $N$ | 0.0001 |
| $Re_\infty$ (m$^{-1}$) | $3.6 \times 10^5$ |





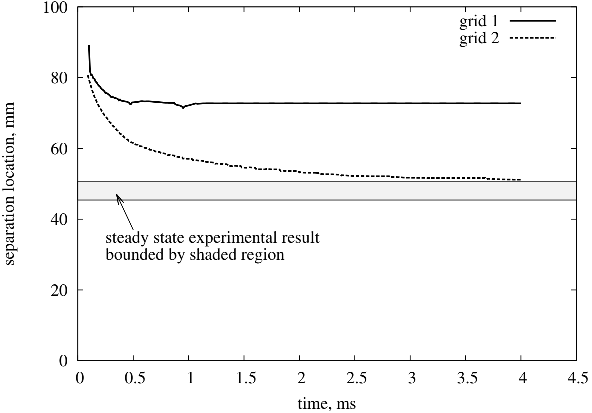
# Double cone: grid resolution study

| Grid | no. cells normal to wall | no. cells along first ramp | no. cells along second ramp | no. cells along shoulder | total no. cells |
|---|---|---|---|---|---|
| 1 | 40 | 120 | 120 | 60 | 12 000 |



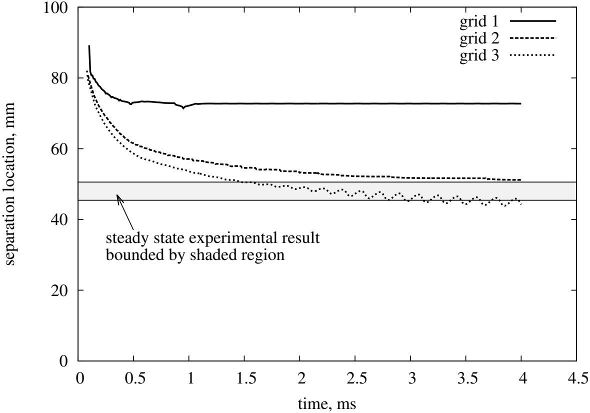
# Double cone: grid resolution study

| Grid | no. cells<br>normal to wall | no. cells<br>along first ramp | no. cells<br>along second ramp | no. cells<br>along shoulder | total no. cells |
|---|---|---|---|---|---|
| 1 | 40 | 120 | 120 | 60 | 12 000 |
| 2 | 80 | 240 | 240 | 120 | 48 000 |



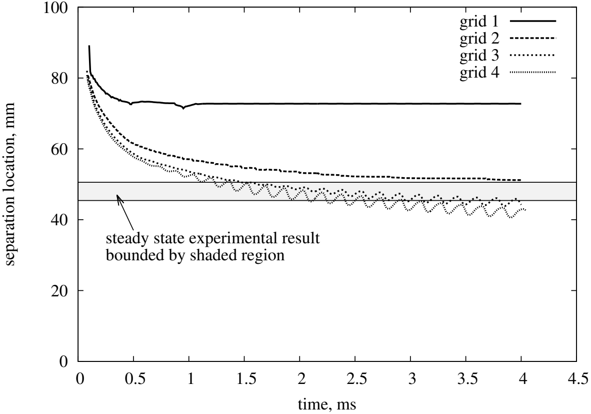
# Double cone: grid resolution study

| Grid | no. cells<br>normal to wall | no. cells<br>along first ramp | no. cells<br>along second ramp | no. cells<br>along shoulder | total no. cells |
|---|---|---|---|---|---|
| 1 | 40 | 120 | 120 | 60 | 12 000 |
| 2 | 80 | 240 | 240 | 120 | 48 000 |
| 3 | 120 | 360 | 360 | 180 | 108 000 |



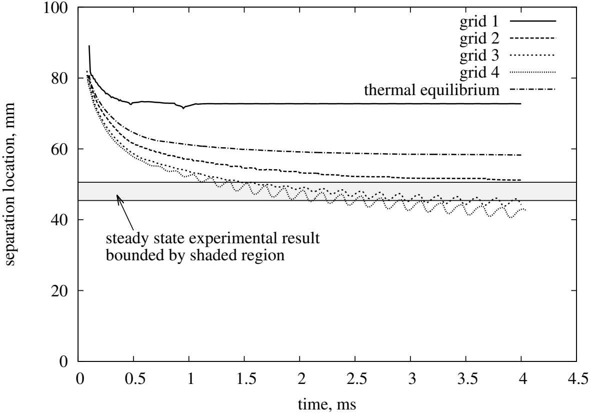
# Double cone: grid resolution study

| Grid | no. cells<br>normal to wall | no. cells<br>along first ramp | no. cells<br>along second ramp | no. cells<br>along shoulder | total no. cells |
|---|---|---|---|---|---|
| 1 | 40 | 120 | 120 | 60 | 12 000 |
| 2 | 80 | 240 | 240 | 120 | 48 000 |
| 3 | 120 | 360 | 360 | 180 | 108 000 |
| 4 | 160 | 480 | 480 | 240 | 192 000 |

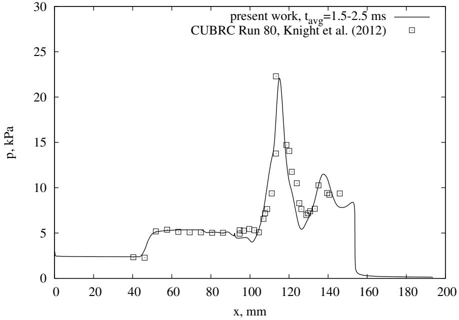


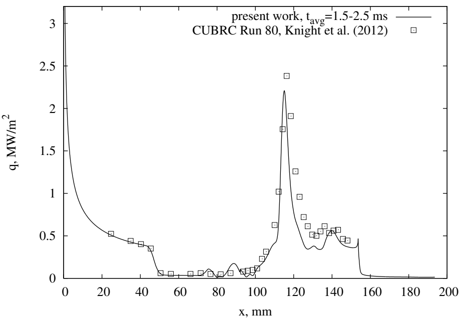
# Double cone: thermal equilibrium assumption

| Grid | no. cells<br>normal to wall | no. cells<br>along first ramp | no. cells<br>along second ramp | no. cells<br>along shoulder | total no. cells |
|---|---|---|---|---|---|
| 1 | 40 | 120 | 120 | 60 | 12 000 |
| 2 | 80 | 240 | 240 | 120 | 48 000 |
| 3 | 120 | 360 | 360 | 180 | 108 000 |
| 4 | 160 | 480 | 480 | 240 | 192 000 |



# Double cone: comparison to experiment

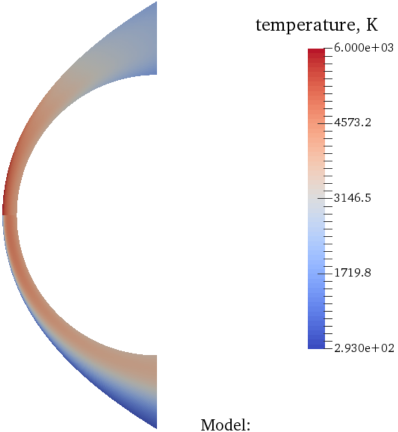




# Validation: 2-temperature reacting air flow over a sphere

Nonaka experiment:
vel.x = 3490 m/s
p_inf = 4850 Pa
T_inf = 293 K
f_N2 = 0.767
f_O2 = 0.233

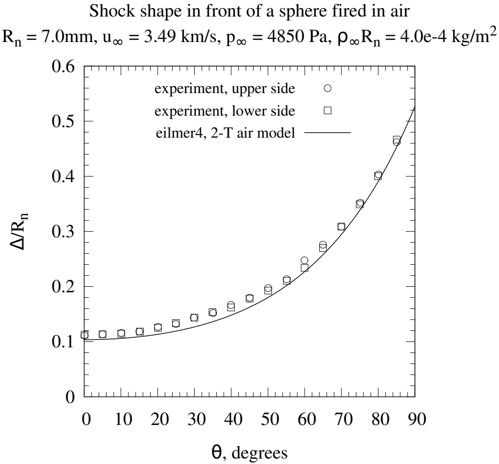
Top half: transrotaional temperature
Bottom half: vibroelectronic temperature



t = 60.172 micro s

Model:
5-species, 2-temperature air
Gupta et al reaction rates
Park 2-T modifications for rate constants

# Validation: 2-temperature reacting air flow over a sphere

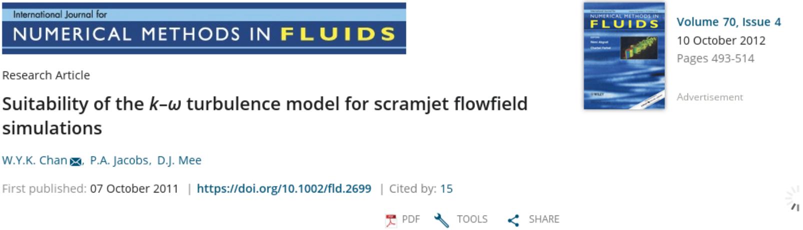


# Turbulence modelling

When the turbulence model is activated, Eilmer solves the Reynolds-Averaged Navier-Stokes (RANS) equations.

The turbulence model in Eilmer is the Wilcox-2006 $k - \omega$ model.

For more details, see:



International Journal for
NUMERICAL METHODS IN FLUIDS

Research Article

**Suitability of the *k*–*ω* turbulence model for scramjet flowfield simulations**

W.Y.K. Chan, P.A. Jacobs, D.J. Mee

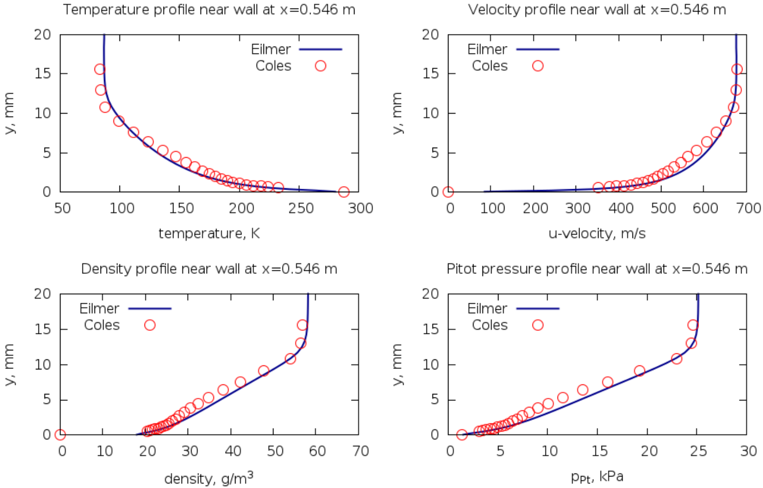
First published: 07 October 2011 | https://doi.org/10.1002/fld.2699 | Cited by: 15

PDF TOOLS SHARE

Volume 70, Issue 4
10 October 2012
Pages 493-514

Advertisement

# Validation: $k - \omega$ turbulence model


Temperature profile near wall at x=0.546 m

Velocity profile near wall at x=0.546 m

Density profile near wall at x=0.546 m

Pitot pressure profile near wall at x=0.546 m

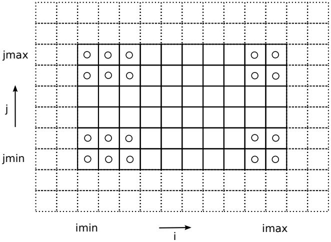
Mach 3.7 flow over a flat plate, adiabatic wall condition.

# Advanced and experimental features

## User-customisation: boundary conditions and source terms

- Sometimes users would like to do a special one-time modelling job that requires special boundary conditions or special source terms.
- Another use for the user-defined BCs and source terms is when one wants to test or prototype a boundary condition before it gets converted to D code and included in the main simulation program.
- Eilmer allows programmable boundary conditions and source terms by allowing the user to write small Lua programs.

# User-customisation: boundary conditions and source terms



```
function ghostCells(args)
   ghost = {}
   ghost.p = 95.84e3 -- pressure, Pa
   ghost.T = 1103.0  -- temperatures, K (as a table)
   ghost.massf = {air=1.0} -- mass fractions to be provided as a table
   ghost.velx = 1000.0  -- x-velocity, m/s
   ghost.vely = 0.0     -- y-velocity, m/s
   ghost.velz = 0.0
   return ghost, ghost
end
```

# Moving grid capability

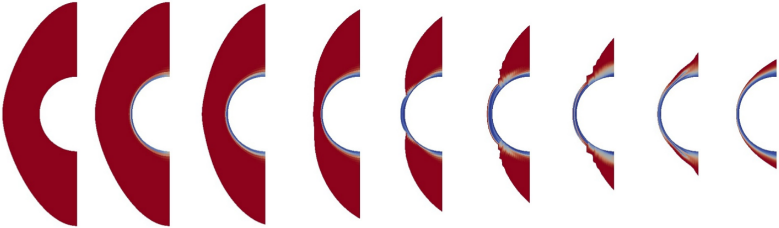
- Moving grids and moving boundaries are also controlled using small Lua programs built by the user.
- It is hard to predict all the types of motion that users would want to simulate, hence the user-built programs to control the motion.
- The user needs to set the velocity of the grid points and the simulation code takes care of updating to the new positions.
- With enoug creativity, this can be used to simulate fluid-structure interation problems.

# Moving grid capability: grid motion program

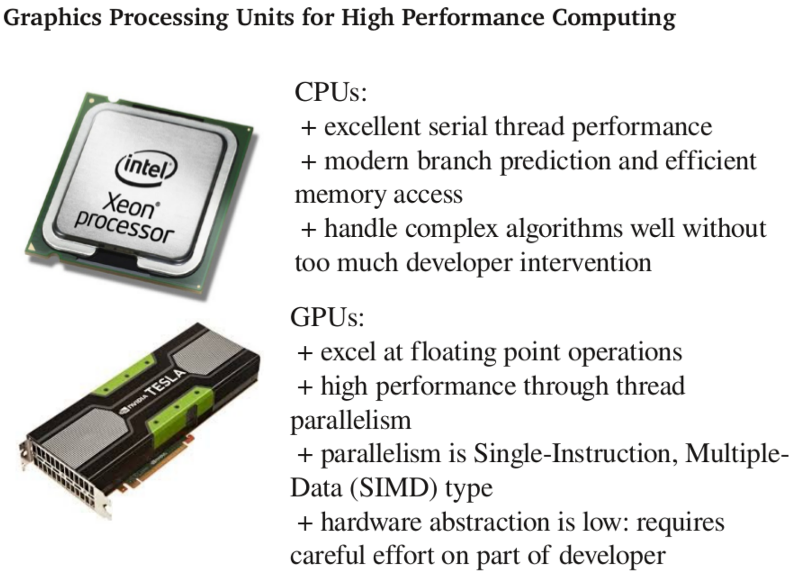
```
pSpeed = 293.5 -- m/s
L = 0.5 -- m
H = 0.1 -- m
endDomain = L

function assignVtxVelocities(sim_time, dt)
   pPos = pSpeed * sim_time
   L = endDomain - pPos
   imin = blockData[0].vtxImin
   imax = blockData[0].vtxImax
   jmin = blockData[0].vtxJmin
   jmax = blockData[0].vtxJmax
   for j=jmin,jmax do
      for i=imin,imax do
         pos = getVtxPosition(0, i, j, 0)
         vtxSpeed = ((endDomain - pos.x)/L)*pSpeed
         setVtxVelocity(Vector3:new{x=vtxSpeed}, 0, i, j)
      end
   end
end
```

# Shock-fitting boundaries

- For blunt body flows, it is critical to have nicely aligned grids with the bow shock.
- One solution is to do direct shock-fitting at the boundaries.
- Eilmer uses moving grid capability and a specialised shock-fitting boundary condition to allow shock-fitting at inflow boundaries.



# GPU acceleration for chemically reacting flows



## Graphics Processing Units for High Performance Computing

CPUs:

+ excellent serial thread performance

+ modern branch prediction and efficient memory access

+ handle complex algorithms well without too much developer intervention

GPUs:

+ excel at floating point operations

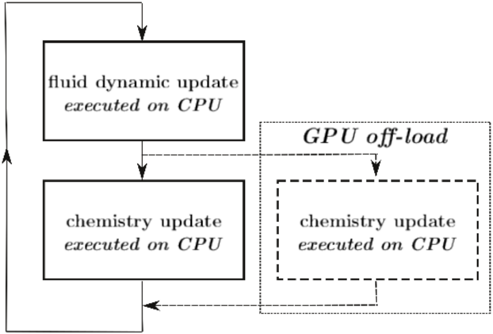
+ high performance through thread parallelism

+ parallelism is Single-Instruction, Multiple-Data (SIMD) type

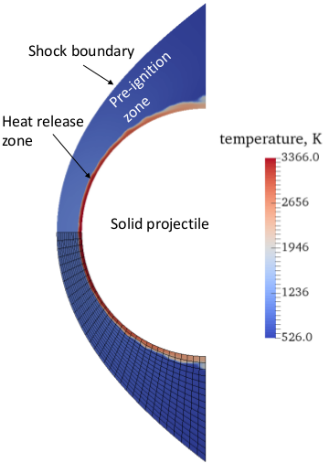
+ hardware abstraction is low: requires careful effort on part of developer

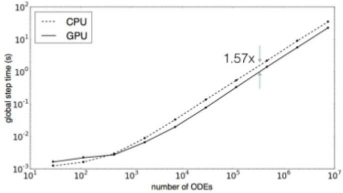
# GPU acceleration: Eilmer implementation

+ operator-splitting algorithm

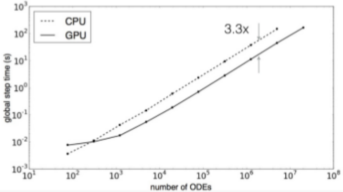

fluid dynamic update *executed on CPU*
*GPU off-load*
chemistry update *executed on CPU*
chemistry update *executed on CPU*

# GPU acceleration: proof-of-concept, early results



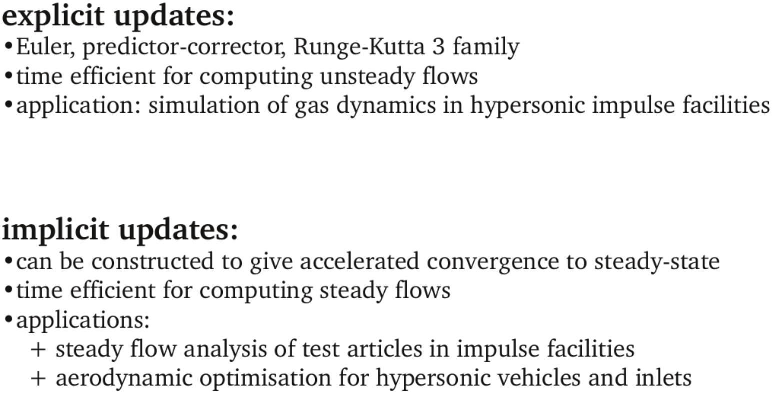


Hydrogen combustion



Methane combustion

# Steady-state solver: update schemes in Eilmer



**explicit updates:**

- Euler, predictor-corrector, Runge-Kutta 3 family
- time efficient for computing unsteady flows
- application: simulation of gas dynamics in hypersonic impulse facilities

**implicit updates:**

- can be constructed to give accelerated convergence to steady-state
- time efficient for computing steady flows
- applications:
  - + steady flow analysis of test articles in impulse facilities
  - + aerodynamic optimisation for hypersonic vehicles and inlets

# Steady-state solver: outer iterations

- Using Newton's method to solve the zero equation, F(u) = 0.
- These steps are called the outer iterations.

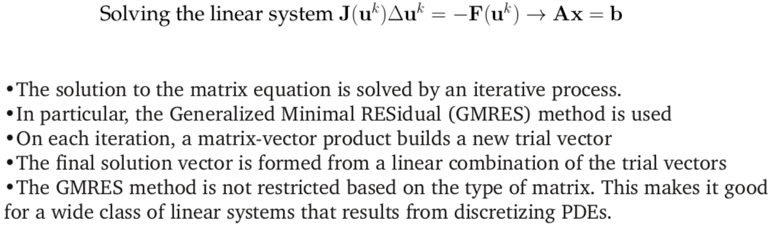
$$\text{Taylor expansion: } \mathbf{F}(\mathbf{u}^{k+1}) = \mathbf{F}(\mathbf{u}) + \mathbf{F}'(\mathbf{u}^k)\left[\mathbf{u}^{k+1} - \mathbf{u}^k\right] + \text{h.o.t}$$

- Set $\mathbf{F}(\mathbf{u}^{k+1}) = 0$
- Ignore higher order terms
- Notation substitution...

$$\mathbf{J}(\mathbf{u}^k)\Delta\mathbf{u}^k = -\mathbf{F}(\mathbf{u}^k), \quad \mathbf{u}^{k+1} = \mathbf{u}^k + \Delta\mathbf{u}^k$$

```
0. Given $\mathbf{u}^0$
1. Repeat for $k = 0,...,$ good enough:

    1a. Solve $\mathbf{J}(\mathbf{u}^k)\Delta\mathbf{u}^k = -\mathbf{F}(\mathbf{u}^k)$
    1b. $\mathbf{u}^{k+1} = \mathbf{u}^k + \Delta\mathbf{u}^k$
```

# Steady-state solver: inner iterations



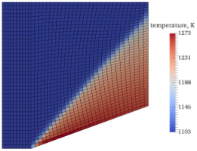
Solving the linear system $\mathbf{J}(\mathbf{u}^k)\Delta\mathbf{u}^k = -\mathbf{F}(\mathbf{u}^k) \rightarrow \mathbf{A}\mathbf{x} = \mathbf{b}$

- The solution to the matrix equation is solved by an iterative process.
- In particular, the Generalized Minimal RESidual (GMRES) method is used
- On each iteration, a matrix-vector product builds a new trial vector
- The final solution vector is formed from a linear combination of the trial vectors
- The GMRES method is not restricted based on the type of matrix. This makes it good for a wide class of linear systems that results from discretizing PDEs.

# Steady-state solver: some test cases

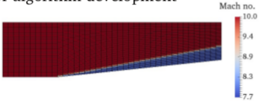
### Mach 1.5 flow over a 20-deg cone

- axisymmetric, inviscid, low supersonic Mach number
- the classic "intro to eilmer" test case



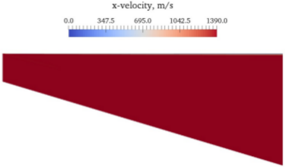
### Mach 10 flow over a 6-deg wedge

- 2D planar, inviscid, high Mach number
- conditions relevant to SPARTAN flight
- principal test case for algorithm development



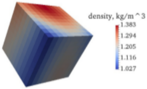
### Mach 4 laminar flow over a flat plate

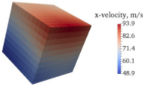
- 2D planar, viscous
- high aspect ratio cells



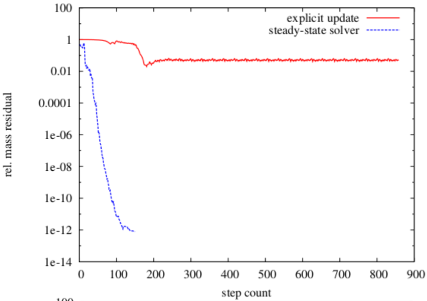
### 3D Method of Manufactured Solutions

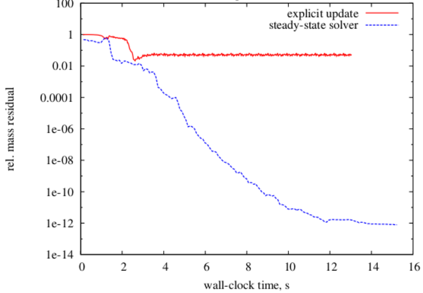
- unstructured grid in 3D
- multiple block
- parallel
- viscous terms
- user-defined B.C.s
- user-defined source terms





# Steady-state solver

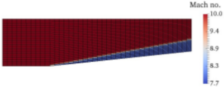




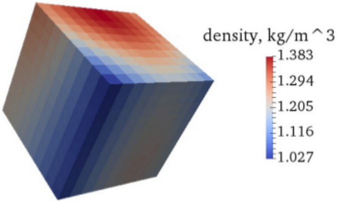
## Mach 10 flow over a 6 deg wedge

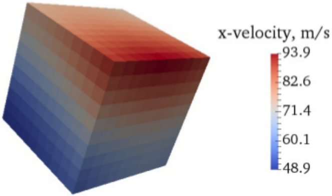
**Notes:**

- Flux calculator: ausmdv
- Geometry: 2D planar
- Steady-state solver can achieve 12-orders of magnitude residual drop on relatively poor grid
- Explict updates stall early
- Compute core: 1 x Intel Core i3-4170



# Steady-state solver

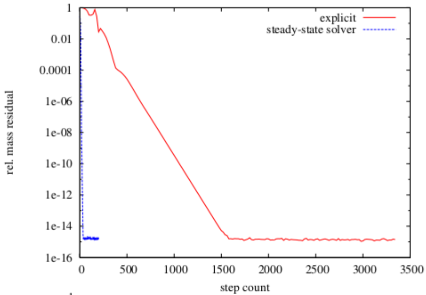


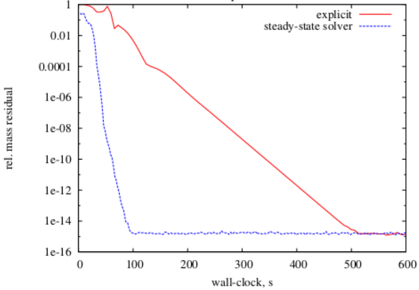


**Throwing in the kitchen sink...**

- unstructured grid in 3D
- multiple block
- parallel
- viscous terms
- user-defined B.C.s
- user-defined source terms

# Steady-state solver

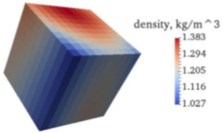


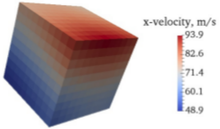


## 3D Method of Manufactured Solutions

**Notes:**

- Compute cores: 8 x AMD Opteron 6378


density, kg/m^3
1.383
1.294
1.205
1.116
1.027


x-velocity, m/s
93.9
82.6
71.4
60.1
48.9