Simulating Re-entry flows with Eilmer 4

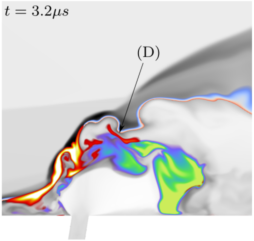
Nick N. Gibbons,
The University of Queensland, Brisbane, Queensland 4072, Australia

July 21, 2021

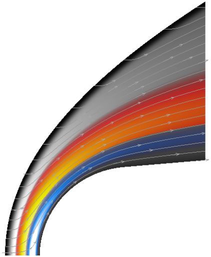
About Me!

- Undergraduate degree in Aerospace Engineering at UQ (2010-2013)
- PhD in Supersonic Turbulent Combustion also at UQ (2014-2019)
  - Supervised by Prof. Vincent Wheatley and Dr. Alexander Klimenko
- Postdoctoral Research Fellow (2020-present)
  - Supervised by Dr. Rowan Gollan
  - Linkage project with Lockheed Martin Australia on Electron Transpiration Cooling

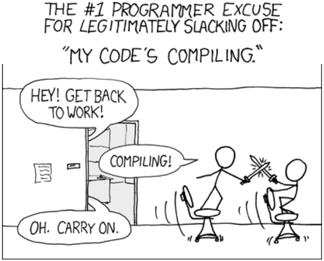
- Research Things I Do:



Supersonic Turbulent Combustion



Hypersonic Plasma Flows

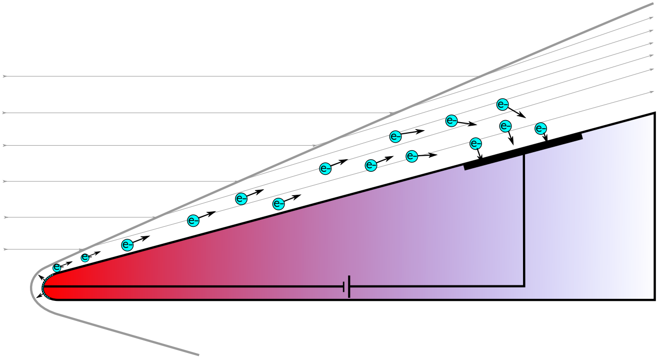


Parallel Programming for HPC

- You can follow me on Twitter! (@DrNickNGibbons)

Electron Transpiration Cooling

- Electrons take energy away from the surface and cool the emission point
- Surfaces can be sharp, compared to internal active cooling
- Need temperature/oxidation resistant material with low work function Φ



Eilmer4: An Open Source Compressible Fluid Solver

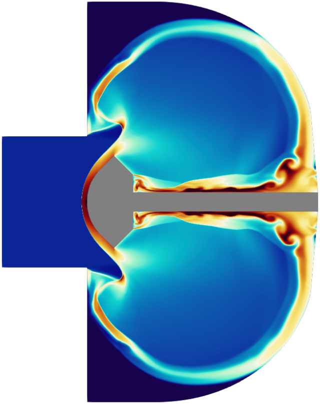
- Eilmer4 is our flagship compressible fluid dynamics code
- Freely available under the GPL 3.0 license (`https://gdtk.uqcloud.net/`)

Features:

- 2D/3D Structured or Unstructured grids
- Parallel scaling to thousands of cores
- RANS turbulence modelling
- Chemical reactions and multi-temperature flow
- Build your own grids or import them from common formats
- Extensively tested against data from hypersonics experiments

Experimental Features:

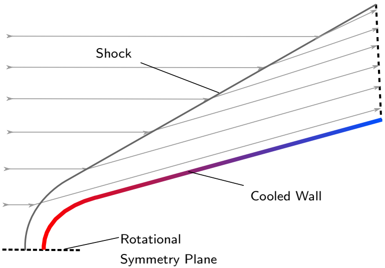
- Steady-state accelerator
- Low-dissipation fluxes for DNS/LES
- Lots of other things...

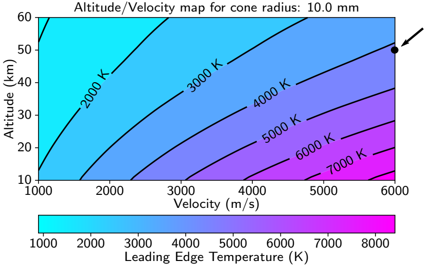


Eilmer4 solution of expansion tube over the Muses-C aeroshell, by Peter Jacobs.

Example Simulation: Uniform

- Work-in-progress simulations for AIAA ASCEND conference in November
- Blunt cone ($r = 10mm$) with cooled wall at $v = 6000m/s$ and $h = 50km$
- Compute leading edge temperatures for Radiation only versus Radiation+ETC
- Electron Transpiration Cooling model from Alkandry, Hanquist, and Boyd, 2014
- Convective heating estimates from correlation by Brandis and Johnston, 2014

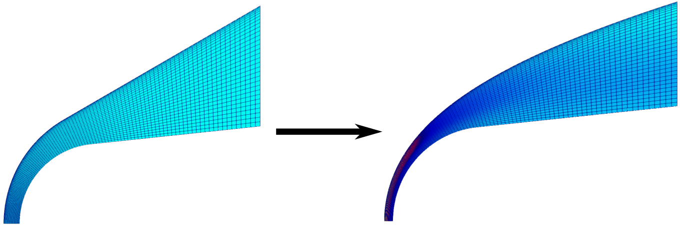




Solving Reentry Flows

- Workflow:
  1. Inviscid, shock fitting simulation to establish shock shape
  2. Extract shock boundary and regrid with clustering
  3. Coarse viscous solution with steady state solver
  4. Fine viscous solution with steady state solver

Shock Fitting Procedure:



Example Script: Core Simulation Parameters

Contents of input file etc.lua:

```lua
-- Simulation of ETC cooled cones phase 2: e4-nk-dist viscous solve
--
-- @author: Nick N. Gibbons
-- 2021-06-25

job_title = "Blunt Cone Shock Fitting"
config.dimensions = 2
config.axisymmetric = true
config.viscous = true

config.flux_calculator = "hanel"
config.spatial_deriv_locn = "cells"
config.spatial_deriv_calc = "least_squares"

config.mass_diffusion_model = "ficks_first_law"
config.diffusion_coefficient_type = "binary_diffusion"
...
```

Example Script: Numerical Method Options

```
-- Simulation of ETC cooled cones phase 2: e4-nk-dist viscous solve
--
-- @author: Nick N. Gibbons
-- 2021-06-25

job_title = "Blunt Cone Shock Fitting"
config.dimensions = 2
config.axisymmetric = true
config.viscous = true

config.flux_calculator = "hanel"
config.spatial_deriv_locn = "cells"
config.spatial_deriv_calc = "least_squares"

config.mass_diffusion_model = "ficks_first_law"
config.diffusion_coefficient_type = "binary_diffusion"
...
```

Example Script: Species Diffusion Options

```
config.flux_calculator = "hanel"
config.spatial_deriv_locn = "cells"
config.spatial_deriv_calc = "least_squares"

config.mass_diffusion_model = "ficks_first_law"
config.diffusion_coefficient_type = "binary_diffusion"

nsp, nmodes, gmodel = setGasModel('gm-air11-2T.lua')
config.reacting = true
config.reactions_file = 'rr-kim-air11-2T.lua'
config.energy_exchange_file = 'ee-kim-air11-2T.lua'
...

```

$$\frac{\partial \rho_s}{\partial t} + \frac{\partial}{\partial x_j}(\rho_s u_j) + \frac{\partial}{\partial x_j} D_s \frac{\partial Y_s}{\partial x_j} = \dot{\omega}_s \qquad (1)$$

Example Script: Gas, Chemistry, and Energy Exchange

```
config.mass_diffusion_model = "ficks_first_law"
config.diffusion_coefficient_type = "binary_diffusion"

nsp, nmodes, gmodel = setGasModel('gm-air11-2T.lua')
config.reacting = true
config.reactions_file = 'rr-kim-air11-2T.lua'
config.energy_exchange_file = 'ee-kim-air11-2T.lua'

T_inf = 270.65       -- K
rho_inf = 0.000977521752421559  -- kg/m^3
mass_fraction = {N2=0.767, O2=0.233}
u_inf = 6000.0     -- m/s
...
```

- Reactions and energy exchange models from Kim and Jo, 2021
- Ionisation rates are based on the classic Park, 1993 model
- Includes chemical/exchange coupling from Knab, Fruhauf, and Messerschmid, 1995

Example Script: Inflow Conditions

```
nsp, nmodes, gmodel = setGasModel('gm-air11-2T.lua')
config.reacting = true
config.reactions_file = 'rr-kim-air11-2T.lua'
config.energy_exchange_file = 'ee-kim-air11-2T.lua'

T_inf = 270.65       -- K
rho_inf = 0.0009775217524215̈59  -- kg/m^3
mass_fraction = {N2=0.767, O2=0.233}
u_inf = 6000.0     -- m/s

-- Compute full inflow state from T,rho, and u
Q = GasState:new{gmodel}
Q.T = T_inf;
Q.rho = rho_inf;
Q.massf = mass_fraction
Q.T_modes = {T_inf}
gmodel:updateThermoFromRHOT(Q);
p_inf = Q.p
inflow = FlowState:new{p=p_inf, T=T_inf, velx=u_inf, vely=0.0,
                       massf=mass_fraction, T_modes={T_inf}}

initial=FlowSolution:new{jobName="etc",dir="../one",tindx=40,nBlocks=16}
...
```

Example Script: Initial conditions

```
T_inf = 270.65       -- K
rho_inf = 0.0009775217752421559  -- kg/m^3
mass_fraction = {N2=0.767, O2=0.233}
u_inf = 6000.0     -- m/s

-- Compute full inflow state from T,rho, and u
Q = GasState:new{gmodel}
Q.T = T_inf;
Q.rho = rho_inf;
Q.massf = mass_fraction
Q.T_modes = {T_inf}
gmodel:updateThermoFromRHOT(Q);
p_inf = Q.p
inflow = FlowState:new{p=p_inf, T=T_inf, velx=u_inf, vely=0.0,
                       massf=mass_fraction, T_modes={T_inf}}

initial=FlowSolution:new{jobName="etc",dir="../one",tindx=40,nBlocks=16}

 -- Grid Geometry Specs  --            bo    ___---* co
 ri = 0.010              --        _-- *---/      |
 thetai = math.rad(6.0)  --       /               |
 ro = 0.015              --      /     _-*---------* ci
 diffo = 0.003           --     /     /  bi
 thetao = math.rad(30.0) --     |     |
 L = 0.0202+ri           --  ao *----*ai *oi *oo
...
```

Example Script: Gridding

```
initial=FlowSolution:new{jobName="etc",dir="../one",tindx=40,nBlocks=16}

 -- Grid Geometry Specs   --            bo   ___---* co
 ri = 0.010               --        _-- *---/      |
 thetai = math.rad(6.0)   --       /               |
 ro = 0.015               --      /    _-*---------* ci
 diffo = 0.003            --     /    /  bi
 thetao = math.rad(30.0)  --     |    |
 L = 0.0202+ri            --  ao *----*ai *oi *oo

oi = Vector3:new{x=ri, y=0.0}
ai = oi + ri*Vector3:new{x=-1.0, y=0.0}
bi = oi + ri*Vector3:new{x=-math.sin(thetai), y=math.cos(thetai)}
dxci = L - bi.x
dyci = dxci*math.tan(thetai)
ci = Vector3:new{x=bi.x + dxci, y=bi.y+dyci}
aibi = Arc:new{p0=ai, p1=bi, centre=oi}
bici = Line:new{p0=bi, p1=ci}
aici = Polyline:new{segments={aibi, bici}}

aoco = Spline2:new{filename="../one/shock_shape.dat"}
...
```

Example Script: More Gridding

```
oi = Vector3:new{x=ri, y=0.0}
ai = oi + ri*Vector3:new{x=-1.0, y=0.0}
bi = oi + ri*Vector3:new{x=-math.sin(thetai), y=math.cos(thetai)}
dxci = L - bi.x
dyci = dxci*math.tan(thetai)
ci = Vector3:new{x=bi.x + dxci, y=bi.y+dyci}
aibi = Arc:new{p0=ai, p1=bi, centre=oi}
bici = Line:new{p0=bi, p1=ci}
aici = Polyline:new{segments={aibi, bici}}

aoco = Spline2:new{filename="../one/shock_shape.dat"}
...

grid = StructuredGrid:new{psurface=surface, niv=niv, njv=njv, cfList=cluste

blocks = FBArray:new{grid=grid,
                        initialState=initial,
                        bcList={
                           north=OutFlowBC_Simple:new{},
                           west=InFlowBC_Supersonic:new{flowState=inflow},
                           east=WallBC_NoSlip_FixedT:new{Twall=2000.0, grou
                        nib=2, njb=8}

...
```

Example Script: Blocks and Boundary Conditions

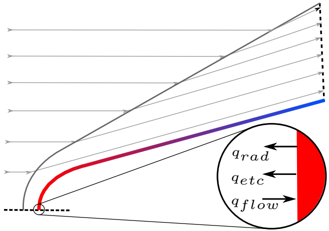
```
aoco = Spline2:new{filename="../one/shock_shape.dat"}
...
grid = StructuredGrid:new{psurface=surface, niv=niv, njv=njv, cfList=clusterlist}

blocks = FBArray:new{grid=grid,
                        initialState=initial,
                        bcList={
                           north=OutFlowBC_Simple:new{},
                           west=InFlowBC_Supersonic:new{flowState=inflow},
                           east=WallBC_ThermionicEmission:new{emissivity=0.89,
                                                               ThermionicEmissionActive=0,
                                                               group="wall"}},
                        nib=2, njb=8}

identifyBlockConnections()

config.write_loads = true
config.boundary_groups_for_loads = "wall"
...

```

$$0 = q_{flow} - q_{rad} - q_{etc} \qquad (2)$$

$$q_{flow} = \kappa \frac{\partial T}{\partial x} - \sum_s h_s D_s \frac{\partial Y_s}{\partial x}$$

$$q_{rad} = \epsilon \sigma_{SB} T^4$$

$$q_{etc} = \frac{A_r T^2 exp\left(\frac{-\Phi}{k_B T}\right)}{Q_e}\left(\Phi + 2k_B T\right)$$



Example Script: Loads Output Options

```
identifyBlockConnections()

config.write_loads = true
config.boundary_groups_for_loads = "wall"

SteadyStateSolver{
   precondition_matrix_type = "ilu",
   frozen_preconditioner_count = 100,
...
```

Interlude: The Steady State Solver

- At each iteration the steady state solver solves a matrix problem:

$$A\mathbf{x} = \mathbf{b}$$

- Most important takeaways:
  - The matrix is solved iteratively
  - The solution is built out of a linear combination of vectors
  - We use a preconditioner to make the matrix easier to solve

$$\mathbf{x}_n = \mathbf{x}_0 + Q_n \mathbf{y}_n$$

$$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_j \end{bmatrix} = \begin{bmatrix} Q_{00} \\ Q_{10} \\ \vdots \\ Q_{j0} \end{bmatrix} \begin{bmatrix} y_0 \end{bmatrix} \qquad \rightarrow \qquad \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_j \end{bmatrix} = \begin{bmatrix} Q_{00} & Q_{01} \\ Q_{10} & Q_{11} \\ \vdots & \vdots \\ Q_{j0} & Q_{j1} \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}$$

Example Script: Steady State Solver Preconditioner

```
config.write_loads = true
config.boundary_groups_for_loads = "wall"

SteadyStateSolver{
   precondition_matrix_type = "ilu",
   frozen_preconditioner_count = 100,

   max_outer_iterations = 40,
   max_restarts = 10,
   use_complex_matvec_eval = true,
...
```

Example Script: Newton-Krylov Configuration

```
SteadyStateSolver{
   precondition_matrix_type = "ilu",
   frozen_preconditioner_count = 100,

   max_outer_iterations = 40,
   max_restarts = 10,
   use_complex_matvec_eval = true,

   number_total_steps = 6000,
   stop_on_relative_global_residual = 1e-6,
...
```

$$= \begin{bmatrix} Q_{00} & Q_{01} & \cdots & Q_{0,40} \\ Q_{10} & Q_{11} & \cdots & Q_{1,40} \\ \vdots & \vdots & \ldots & \vdots \\ Q_{j0} & Q_{j1} & \cdots & Q_{j,40} \end{bmatrix}$$

Example Script: Stopping Criteria

```
    max_outer_iterations = 40,
    max_restarts = 10,
    use_complex_matvec_eval = true,

    number_total_steps = 6000,
    stop_on_relative_global_residual = 1e-6,

    residual_based_cfl_scheduling = false,
    cfl_schedule_length = 12,
    cfl_schedule_value_list = {
       0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0, 20.0, 50.0, 80.0, 100.0, 200.
    },
    cfl_schedule_iter_list = {
       1, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100
    },
...
```

Example Script: Timestep Scheduling

```
number_total_steps = 6000,
stop_on_relative_global_residual = 1e-6,

residual_based_cfl_scheduling = false,
cfl_schedule_length = 12,
cfl_schedule_value_list = {
   0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0, 20.0, 50.0, 80.0, 100.0, 200.0
},
cfl_schedule_iter_list = {
   1, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100
},

-- Settings for start-up phase
number_start_up_steps = 0,
cfl0 = 0.1,
eta0 = 0.008,
sigma0 = 1.0e-30,
...
```

Example Script: Start-up Phase Configuration

```
    residual_based_cfl_scheduling = false,
    cfl_schedule_length = 12,
    cfl_schedule_value_list = {
       0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0, 20.0, 50.0, 80.0, 100.0, 200.
    },
    cfl_schedule_iter_list = {
       1, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100
    },

    -- Settings for start-up phase
    number_start_up_steps = 0,
    cfl0 = 0.1,
    eta0 = 0.008,
    sigma0 = 1.0e-30,

    -- Settings for main solution phase
    cfl1 = 0.1,
    eta1 = 0.001,
    sigma1 = 1.0e-30,
    eta_strategy = "constant",
...
```

Example Script: Main Phase Configuration

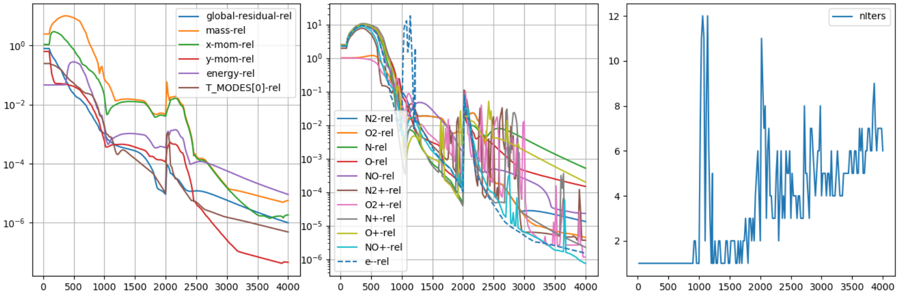
```
    -- Settings for start-up phase
    number_start_up_steps = 0,
    cfl0 = 0.1,
    eta0 = 0.008,
    sigma0 = 1.0e-30,

    -- Settings for main solution phase
    cfl1 = 0.1,
    eta1 = 0.001,
    sigma1 = 1.0e-30,
    eta_strategy = "constant",

    -- Settings control write-out
    snapshots_count = 100,
    number_total_snapshots = 1000,
    write_diagnostics_count = 20,
    write_loads_count = 100,
}
```

$$\mathbf{x}_n = \mathbf{x}_0 + Q_n \mathbf{y}_n$$

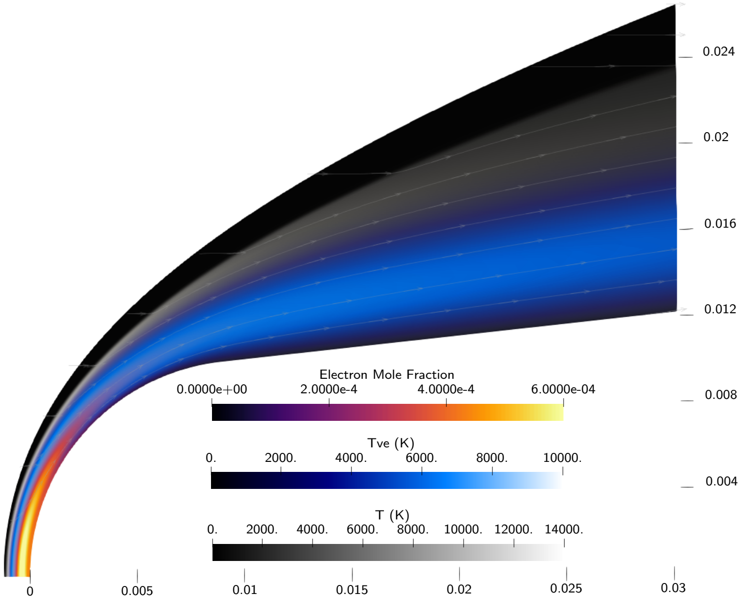
Example Script: Output Configuration

```
    -- Settings for main solution phase
    cfl1 = 0.1,
    eta1 = 0.001,
    sigma1 = 1.0e-30,
    eta_strategy = "constant",

    -- Settings control write-out
    snapshots_count = 100,
    number_total_snapshots = 1000,
    write_diagnostics_count = 20,
    write_loads_count = 100,
}
```

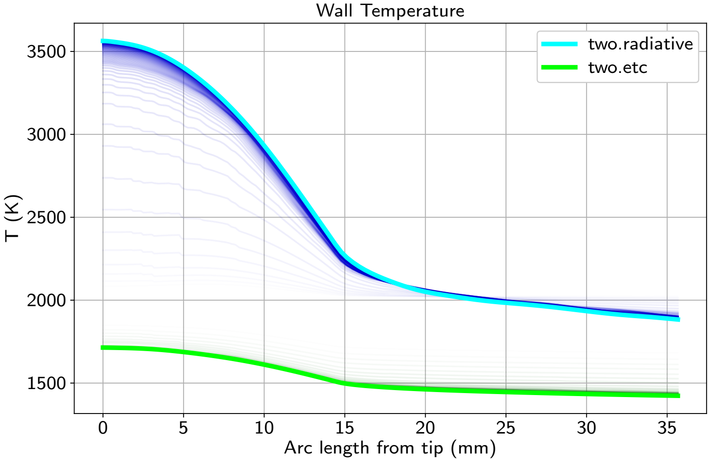
Results: Residual Drop



Results: Flow Solution



Results: Wall Temperature



QA

Thanks for listening!

References:

Hicham Alkandry, Kyle M. Hanquist, and Iain D. Boyd. "Conceptual Analysis of Electron Transpiration Cooling for the Leading Edges of Hypersonic Vehicles". In: 11th AIAA/ASME Joint Thermophysics and Heat Transfer Conference. 2674. Atlanta, GA, 2014.

A. M. Brandis and C. O. Johnston. "Characterisation of Stagnation-Point Heat Flux for Earth Entry". In: 45th AIAA Plasmadynamics and Lasers Conference. 2374. Atlanta, GA, 2014.

Jae Gang Kim and Sung Min Jo. "Modification of chemical-kinetic parameters for 11-air species in re-entry flows". In: International Journal of Heat and Mass Transfer 169.120950 (2021). DOI: doi.org/10.1016/j.ijheatmasstransfer.2021.120950.

O. Knab, H. H¿ Fruhauf, and E. W. Messerschmid. "Theory and Validation of a Physically Consistent Coupled Vibration-Chemistry-Vibration Model". In: Journal of Thermophysics and Heat Transfer 9.2 (1995).

Chul Park. "Review of Chemical-Kinetic Problems of Future NASA Missions, I: Earth Entries". In: Journal of Thermophysics and Heat Transfer 7.3 (1993).