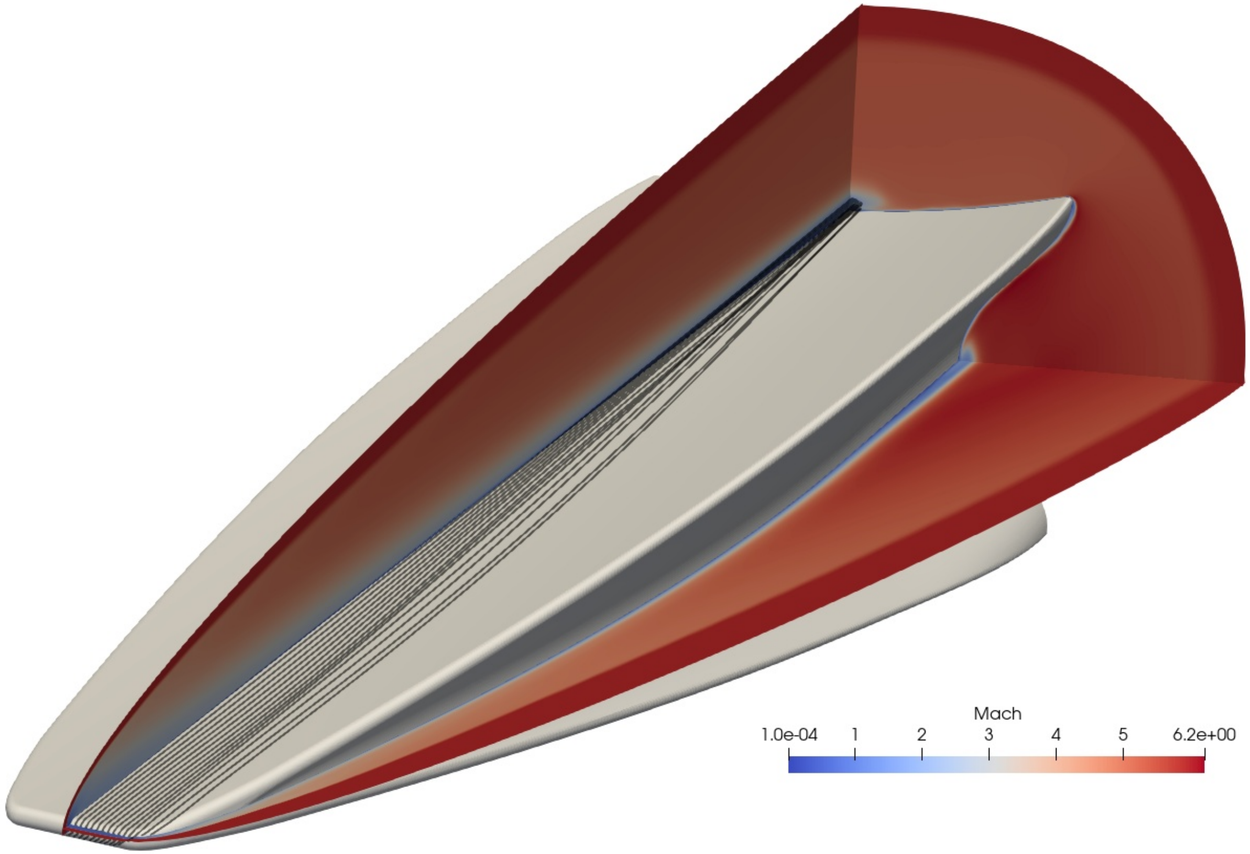
Eilmer Steady-State Accelerator: Investigation of Implicit Schemes

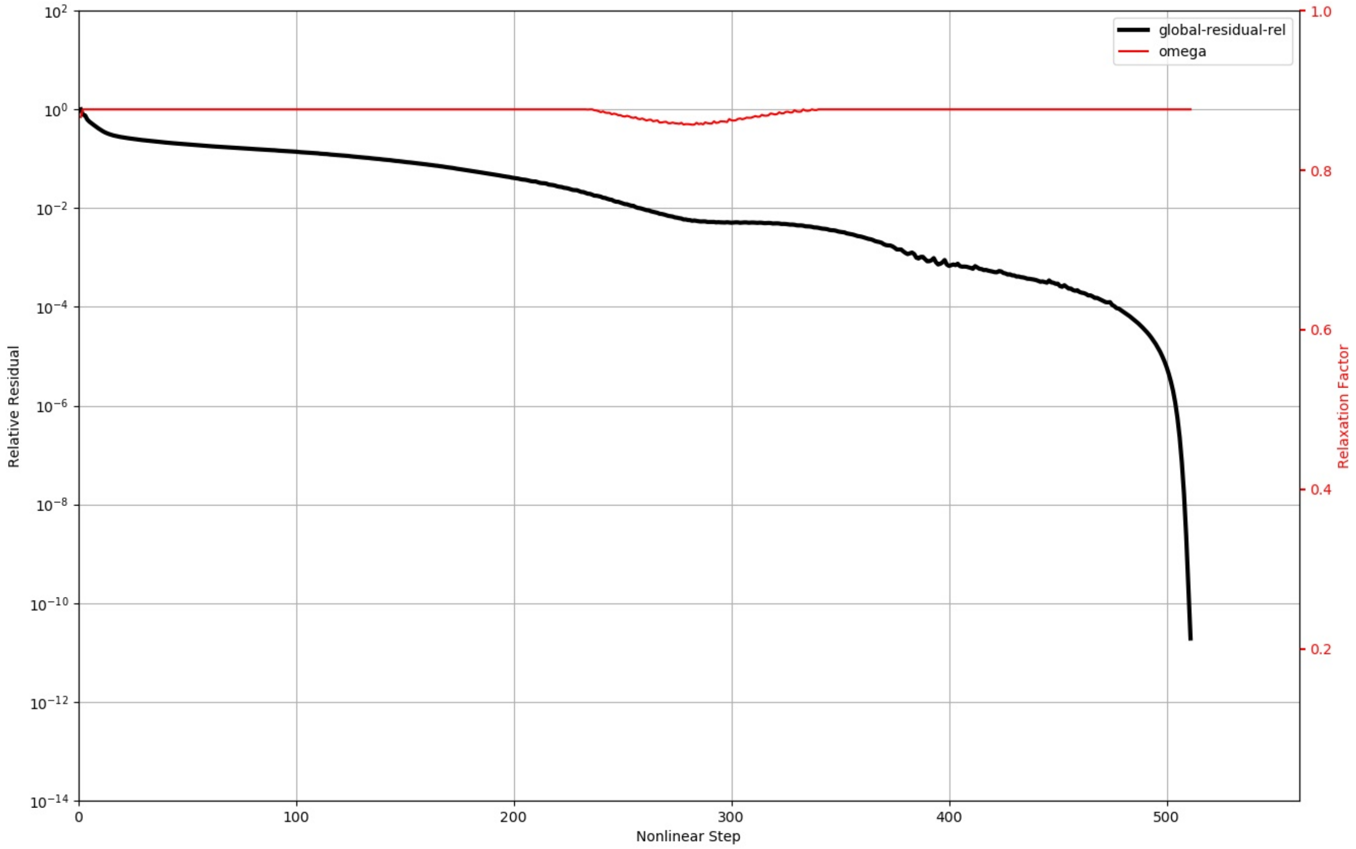
Kyle Damm, *Nick Gibbons, Rowan Gollan, Peter Jacobs*

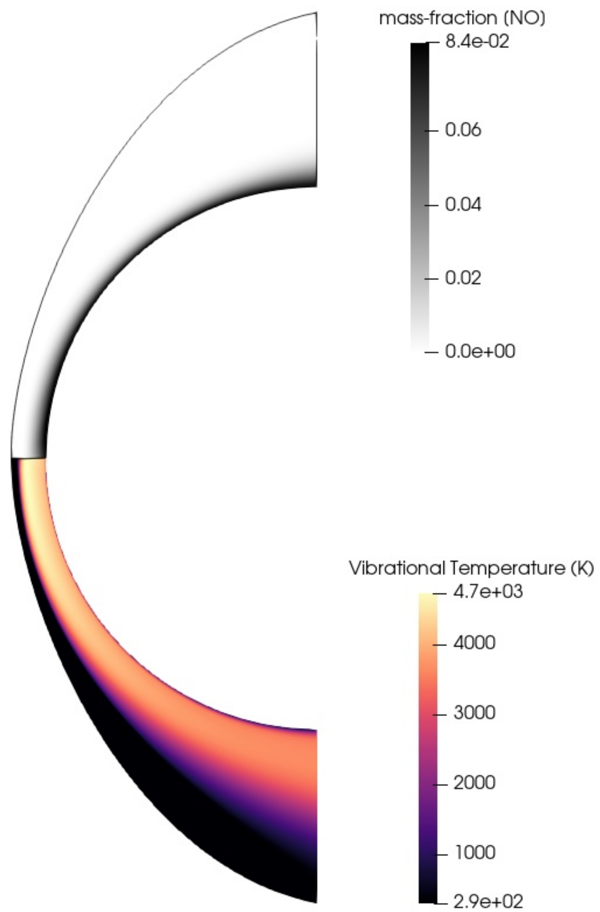


11th August 2022

Steady-State Solver Updates: Since 05-06-2021..

- Significantly improved robustness for reacting flows via a physicality check
  - -- $\mathbf{U}^{n+1} = \mathbf{U}^{n} + \omega \Delta \mathbf{U}$
- Extended solver to include two-temperature modelling





Demonstrative Case: DLR Cylinder in HEG shock tunnel

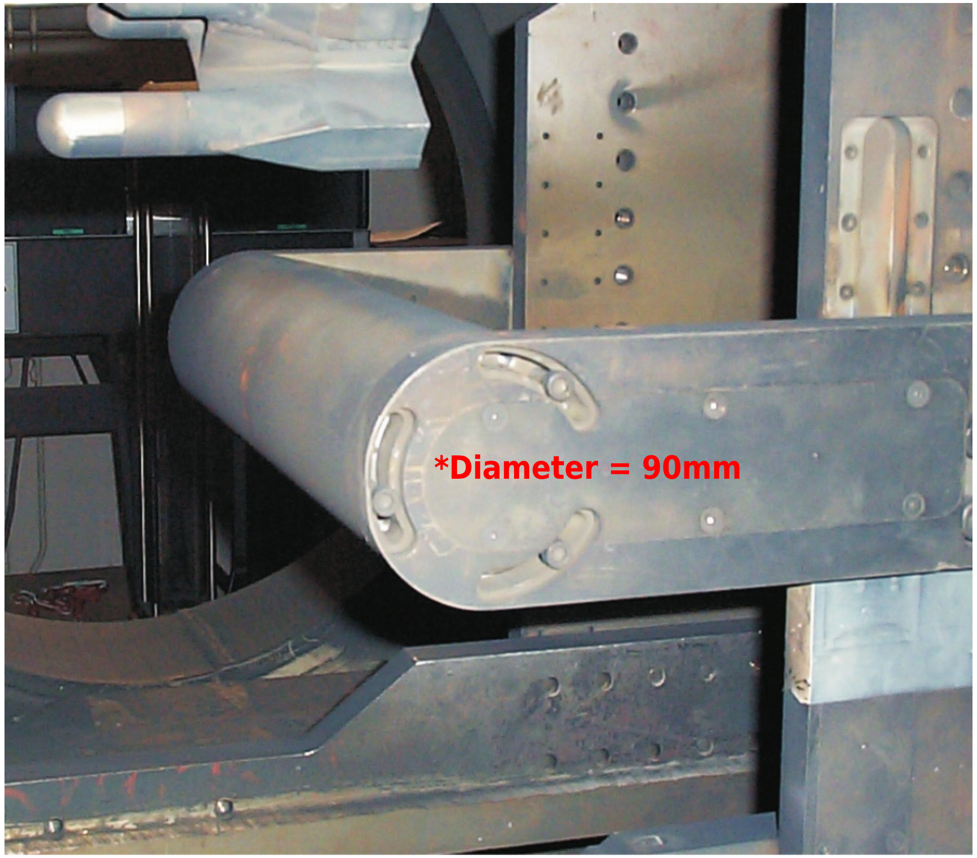


Image taken from Karl et al. 2003

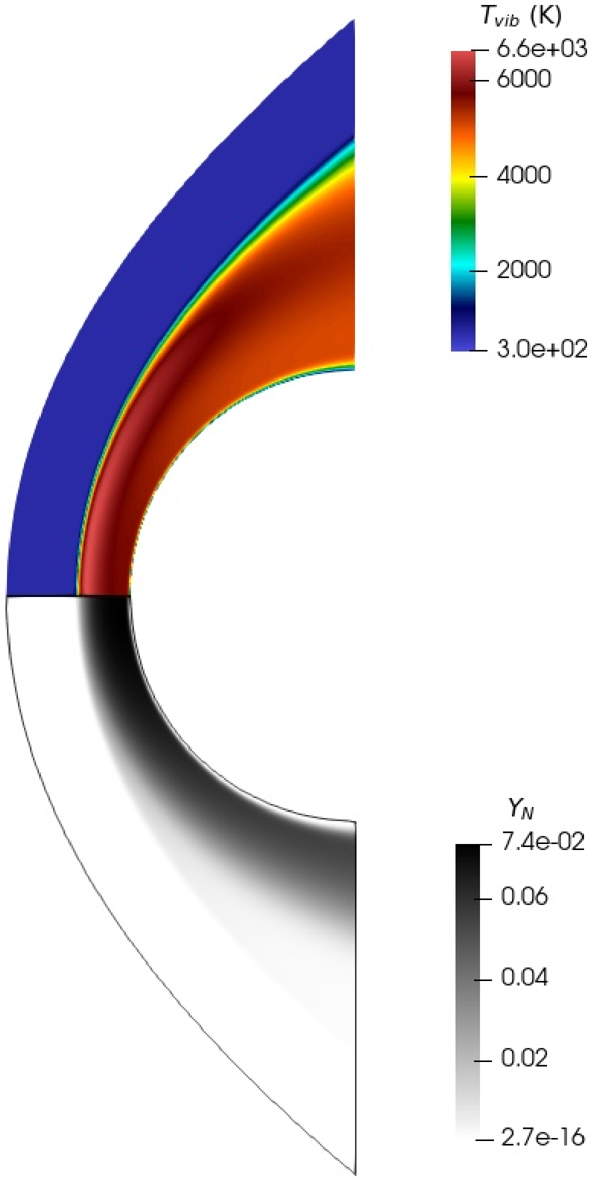
Freestream conditions:

- Mach 8.8 air flow (13.5 MJ/kg enthalpy)
- $Y_\infty = [Y_{N2}, Y_{O2}, Y_{NO}, Y_N, Y_O] = [0.7356, 0.1340, 0.0509, 0.0, 0.07095]$
- $T_\infty = T_{vib,\infty} = 694$ K
- $\rho_\infty = 3.26\text{e-}03$ kg/m^3

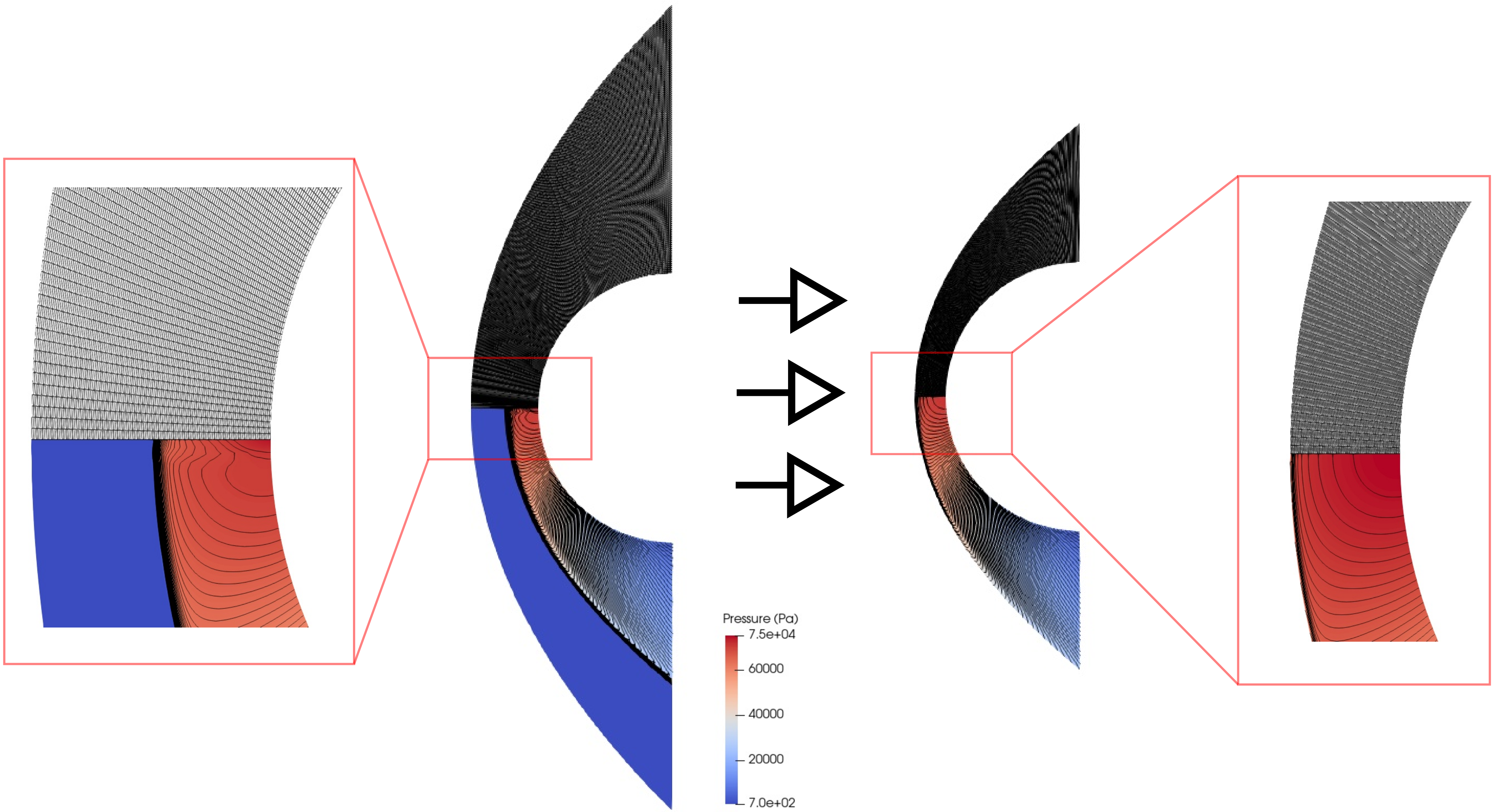
Demonstrative Case: DLR Cylinder in HEG shock tunnel

Simulation details:

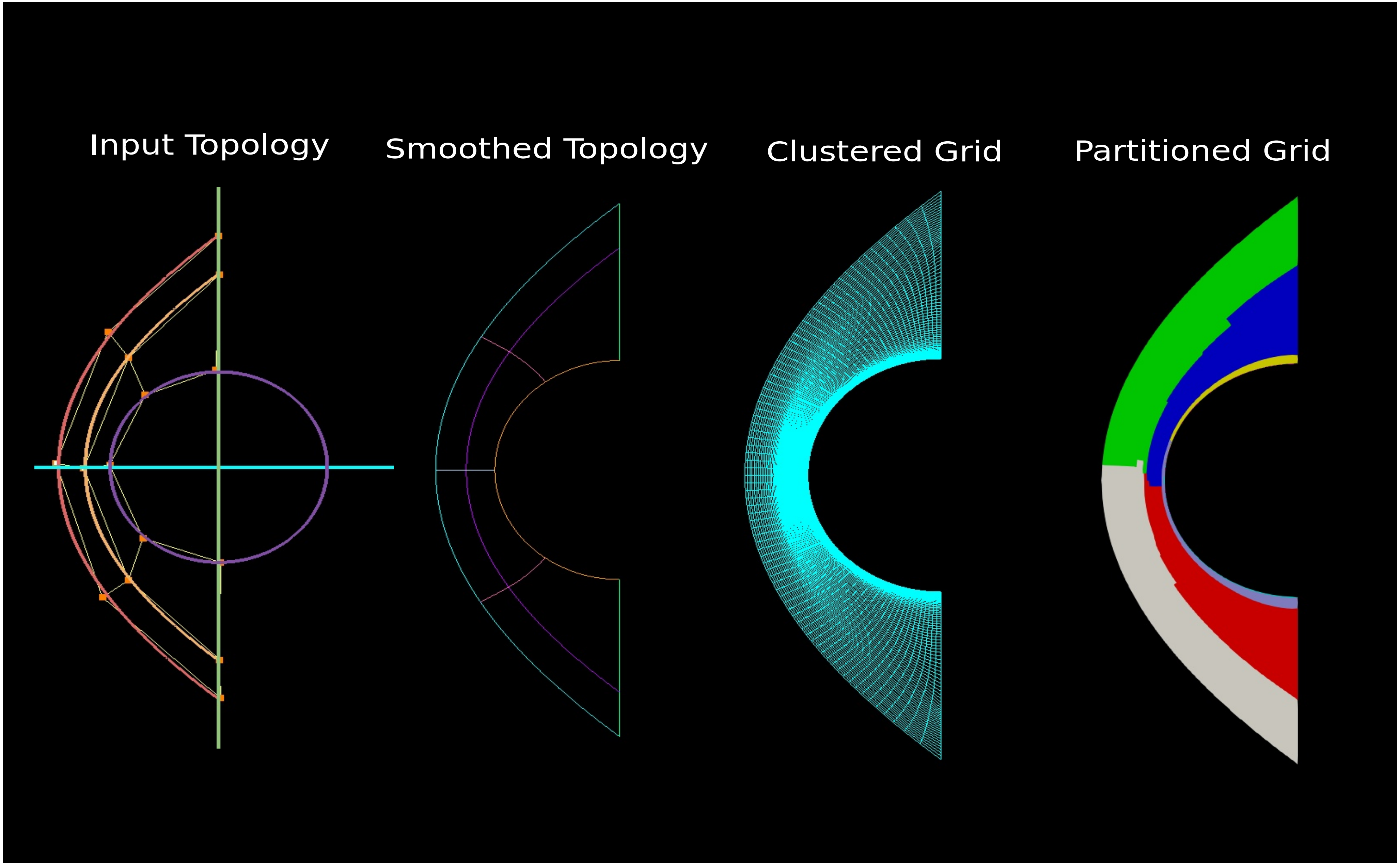
- 2D unstructured grid solver
- LSQ gradient reconstruction with hvenkat limiter
- Blended AUSMDV/Haenel flux scheme
- WLSQ gradients for viscous fluxes at cell-centers
- Thermochemical noneq. using Park (5s/5r) 2T model
- Fick's first law mass diffusion with binary diffusion coeffs.
- Noncatalytic wall boundary condition (T_wall = 300 K)



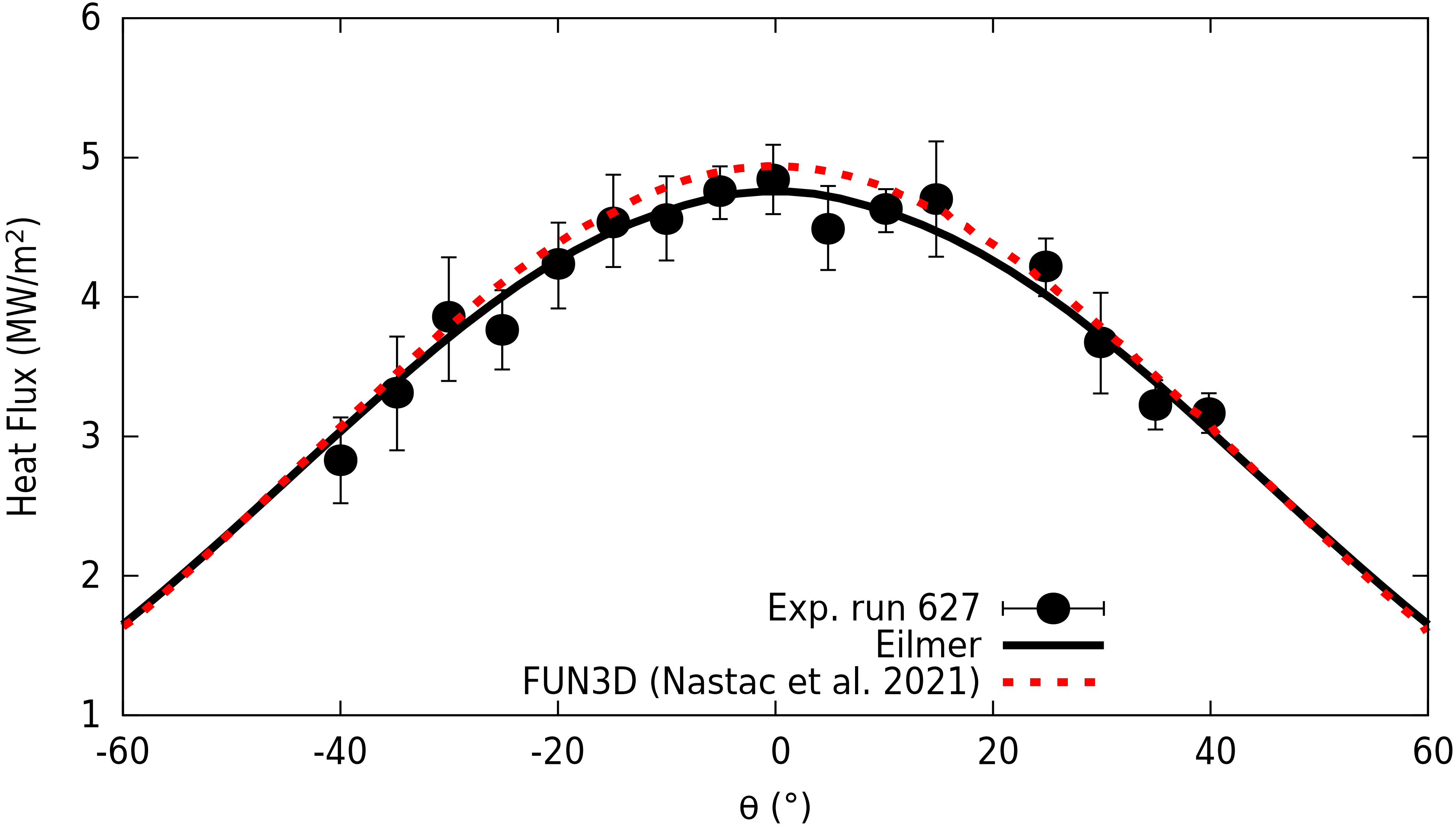
Grid Construction Method: Eilmer Shock-fitting Simulation



Grid Construction Method: GridPro (viscous) Grid



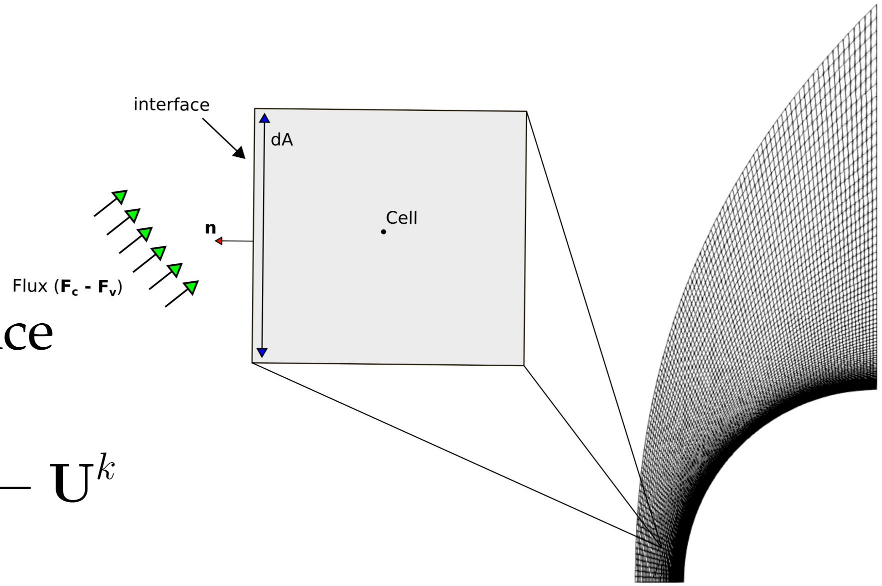
Grid Construction Method: Smooth Heat Flux Distribution



Implicit Schemes: Derivation

Residual function defined as

$$\frac{d\mathbf{U}}{dt} = \mathbf{R}(\mathbf{U}) = -\frac{1}{V}\sum_{faces}(\overline{F_c} - \overline{F_v}) \cdot \hat{n}\, dA + \mathbf{S}$$



Fully discrete form written using a backward difference

$$\frac{\Delta\mathbf{U}^k}{\Delta t} = \mathbf{R}(\mathbf{U}^{k+1}), \quad \Delta\mathbf{U}^k = \mathbf{U}^{k+1} - \mathbf{U}^k$$

Since we don't know $\mathbf{R}(\mathbf{U}^{k+1})$, we linearise in time

$$\frac{\Delta\mathbf{U}^k}{\Delta t} = \mathbf{R}(\mathbf{U}^k) + \frac{\partial\mathbf{R}(\mathbf{U}^k)}{\partial\mathbf{U}^k}\Delta\mathbf{U}^k$$

This is then rearranged to recover the implicit-Euler time marching iterate

$$\mathbf{J}(\mathbf{U}^k)\Delta\mathbf{U}^k = \left[\frac{1}{\Delta t}\mathbf{I} - \frac{\partial\mathbf{R}(\mathbf{U}^k)}{\partial\mathbf{U}^k}\right]\Delta\mathbf{U}^k = \mathbf{R}(\mathbf{U}^k), \quad \mathbf{U}^{k+1} = \mathbf{U}^k + \Delta\mathbf{U}^k$$

Note: as $\frac{1}{\Delta t}$ approaches 0, Newton's method is recovered

Implicit Schemes: Definitions

If we let,

$$A = \mathbf{J}(\mathbf{U}^k), \;\; \mathbf{x} = \Delta\mathbf{U}^k, \;\; \mathbf{b} = \mathbf{R}(\mathbf{U}^k)$$

Then we see that this is a standard linear system,

$$A\mathbf{x} = \mathbf{b}, \quad \text{where} \quad A = L + D + U$$

$$\mathsf{L} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}. \qquad \mathsf{D} = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}. \qquad \mathsf{U} = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Implicit Schemes: Defect-Correction

- ***Approximate*** Jacobian (e.g. first order reconstruction) for the L.H.S. implicit operator (A)
- Solve system using a textbook iterative method (e.g. Jacobi, Gauss-Seidel)
- ***Approximate*** Jacobian ***limits*** maximum achievable ***CFL***
- We will consider...

Diagonal:

$$\mathbf{x}^{n+1} = D^{-1}\mathbf{b}$$

Jacobi:

$$\mathbf{x}^{n+1} = D^{-1}\big(\mathbf{b} - L\mathbf{x}^{n} - U\mathbf{x}^{n}\big)$$

SGS:

$$\mathbf{x}^{n+\frac{1}{2}} = D^{-1}\big(\mathbf{b} - U\mathbf{x}^{n} - L\mathbf{x}^{n+\frac{1}{2}}\big)$$

$$\mathbf{x}^{n+1} = D^{-1}\big(\mathbf{b} - L\mathbf{x}^{n+\frac{1}{2}} - U\mathbf{x}^{n+1}\big)$$

Implicit Schemes: Newton's Method

- ***Uses exact*** Jacobian (e.g. no approximations) for the L.H.S. implicit operator (A)
- Solve system using a Krylov iterative method (e.g. GMRES) --> use Frechet derivative
- Theoretically ***no limits*** on maximum achievable ***CFL***
- We will consider...

Jacobian-Free Newton-Krylov:

ALGORITHM **6.9** *GMRES*

1. Compute $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$, and $v_1 := r_0/\beta$
2. For $j = 1, 2, \ldots, m$ Do:
3. Compute $w_j := Av_j$
4. For $i = 1, \ldots, j$ Do:
5. $h_{ij} := (w_j, v_i)$
6. $w_j := w_j - h_{ij}v_i$
7. EndDo
8. $h_{j+1,j} = \|w_j\|_2$. If $h_{j+1,j} = 0$ set $m := j$ and go to 11
9. $v_{j+1} = w_j/h_{j+1,j}$
10. EndDo
11. Define the $(m+1) \times m$ Hessenberg matrix $\bar{H}_m = \{h_{ij}\}_{1 \le i \le m+1, 1 \le j \le m}$.
12. Compute y_m the minimizer of $\|\beta e_1 - \bar{H}_m y\|_2$ and $x_m = x_0 + V_m y_m$.

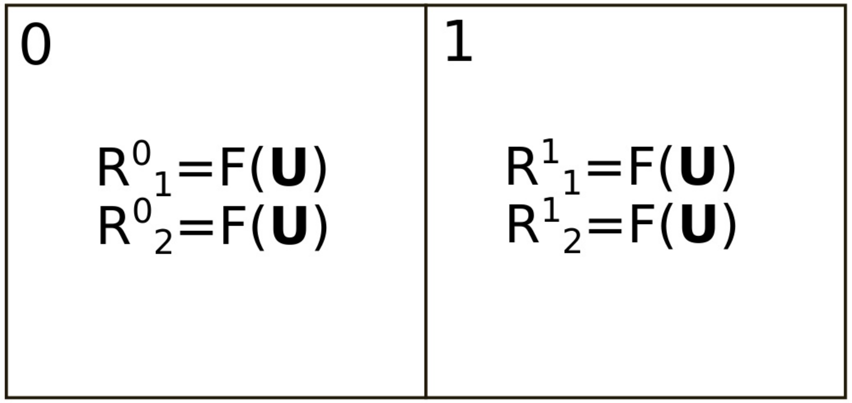
Algorithm taken from Saad (2003)

$$\mathbf{Jv} = \left[\mathbf{R}(\mathbf{U} + \epsilon\mathbf{v}) - \mathbf{R}(\mathbf{U})\right]/\epsilon$$

***We use a complex step variant**

- ***Note:*** GMRES still needs an approximate Jacobian for preconditioning step

Implicit Schemes: Constructing the approximate Jacobian

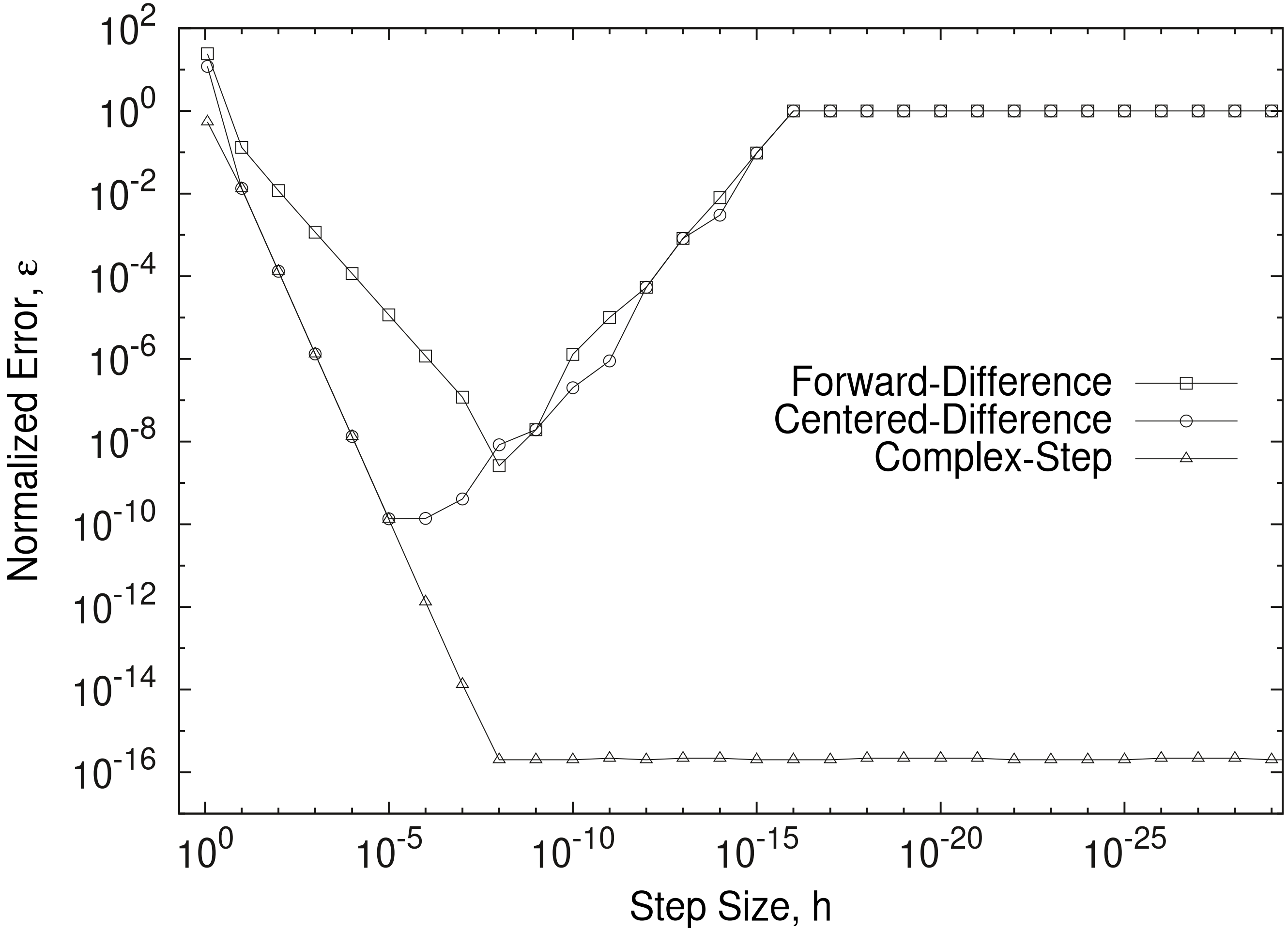


$\mathbf{U} = [U^0_1, U^0_2, U^1_1, U^1_2]$

$$\frac{\partial \mathbf{R}}{\partial \mathbf{U}} = \begin{bmatrix} \frac{\partial \mathbf{R}^0_1}{\partial \mathbf{U}^0_1} & \frac{\partial \mathbf{R}^0_1}{\partial \mathbf{U}^0_2} & \frac{\partial \mathbf{R}^0_1}{\partial \mathbf{U}^1_1} & \frac{\partial \mathbf{R}^0_1}{\partial \mathbf{U}^1_2} \\ \frac{\partial \mathbf{R}^0_2}{\partial \mathbf{U}^0_1} & \frac{\partial \mathbf{R}^0_2}{\partial \mathbf{U}^0_2} & \frac{\partial \mathbf{R}^0_2}{\partial \mathbf{U}^1_1} & \frac{\partial \mathbf{R}^0_2}{\partial \mathbf{U}^1_2} \\ \frac{\partial \mathbf{R}^1_1}{\partial \mathbf{U}^0_1} & \frac{\partial \mathbf{R}^1_1}{\partial \mathbf{U}^0_2} & \frac{\partial \mathbf{R}^1_1}{\partial \mathbf{U}^1_1} & \frac{\partial \mathbf{R}^1_1}{\partial \mathbf{U}^1_2} \\ \frac{\partial \mathbf{R}^1_2}{\partial \mathbf{U}^0_1} & \frac{\partial \mathbf{R}^1_2}{\partial \mathbf{U}^0_2} & \frac{\partial \mathbf{R}^1_2}{\partial \mathbf{U}^1_1} & \frac{\partial \mathbf{R}^1_2}{\partial \mathbf{U}^1_2} \end{bmatrix}$$

$$\frac{\partial \mathbf{R}_k}{\partial \mathbf{U}_j} = \frac{Im[\mathbf{R}_k(\mathbf{U}_j + ih)]}{h}$$

Implicit Schemes: Why use complex numbers?



Function:

$$J(x) = \frac{e^x}{\sqrt{sin^3(x) + cos^3(x)}}$$

Finite difference:

$$\frac{\partial J}{\partial x}_{forward} \approx \frac{J(x+h) - J(x)}{h}$$

$$\frac{\partial J}{\partial x}_{central} \approx \frac{J(x+h) - J(x-h)}{2h}$$

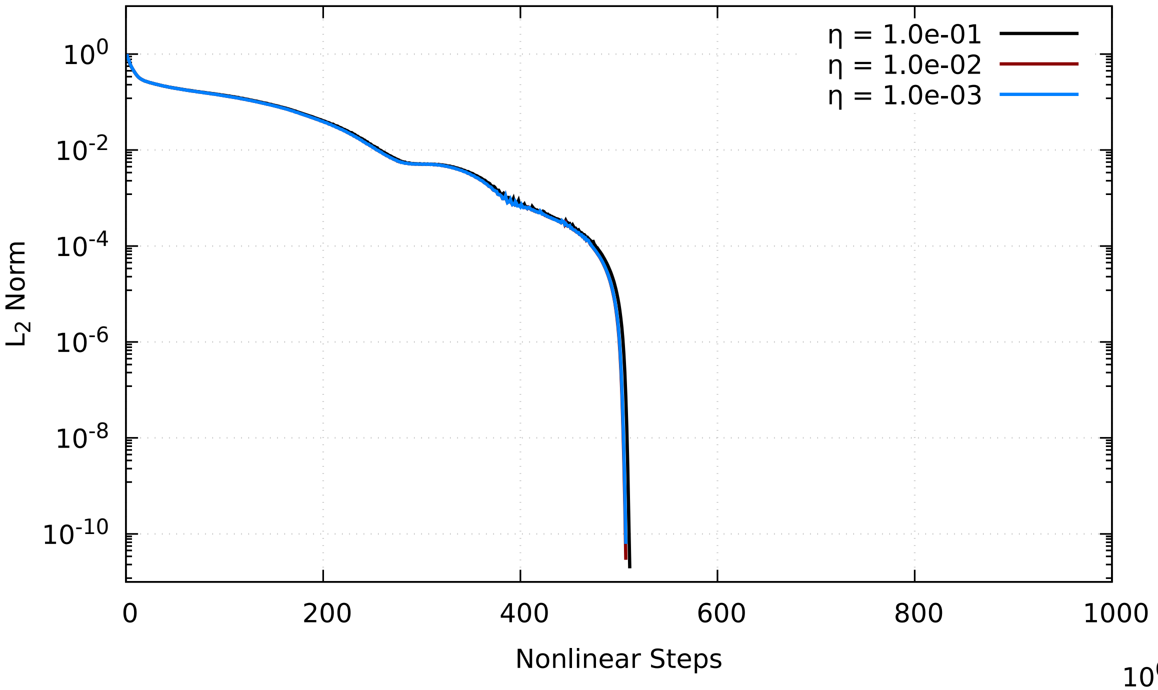
Complex-step derivative approximation

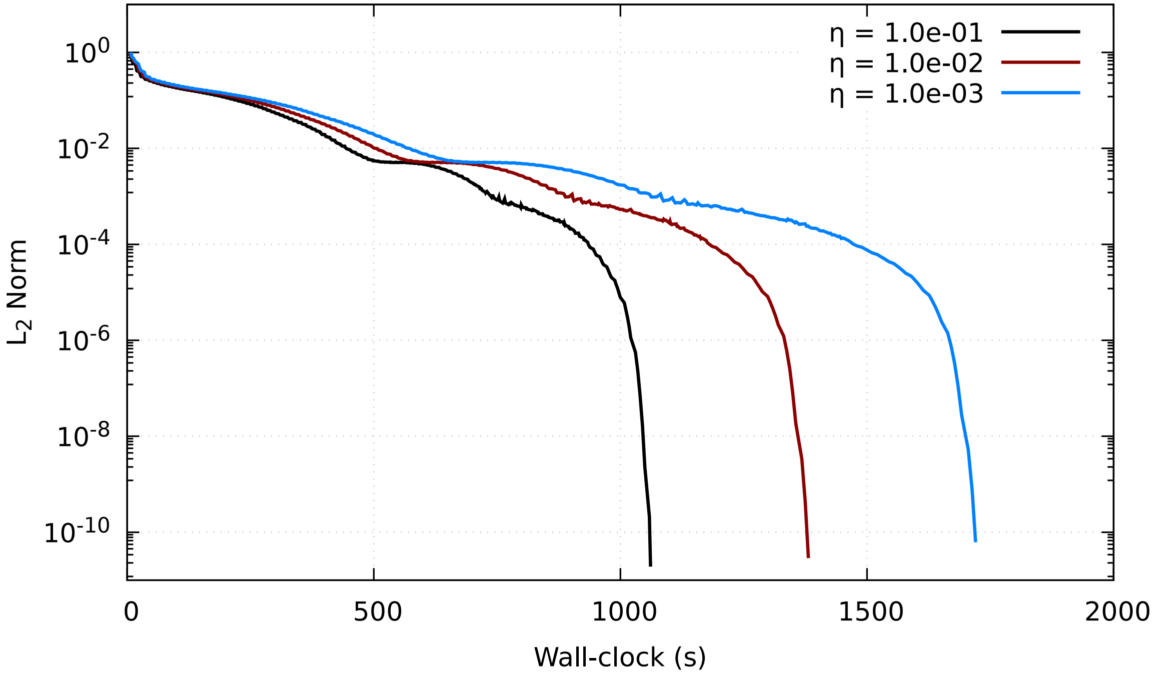
$$\frac{\partial J}{\partial x}_{complex} \approx \frac{Im[J(x+ih)]}{h}$$

Results: Method

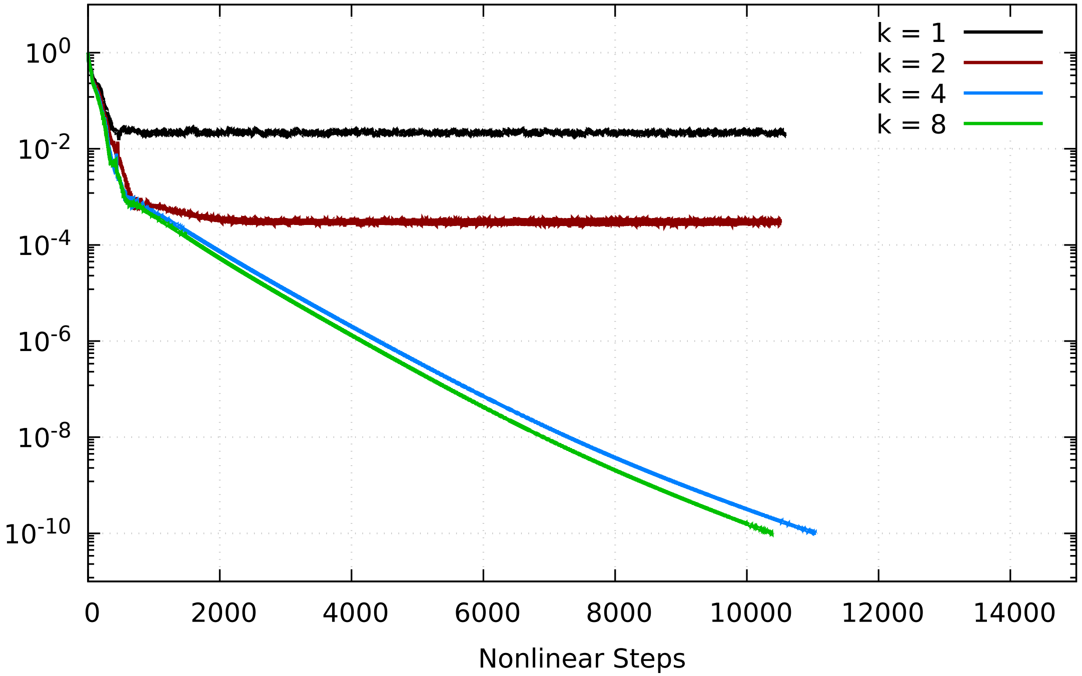
- All simulations use local time-stepping
- Defect-correction schemes use a bespoke prescribed CFL schedule
    -- $CFL_{min} = 0.01$ and $CFL_{max} <= 100.0$
- Newton-Krylov scheme uses an automated CFL growth algorithm
    -- $CFL_{min} = 1.0$ and $CFL_{max} = 1{,}000{,}000$
- Explicit Euler example uses a fixed CFL of 0.05
- Approximate Jacobian matrix evaluated every 5 nonlinear steps
- Convergence tolerance of 10 orders of magnitude drop in the global residual

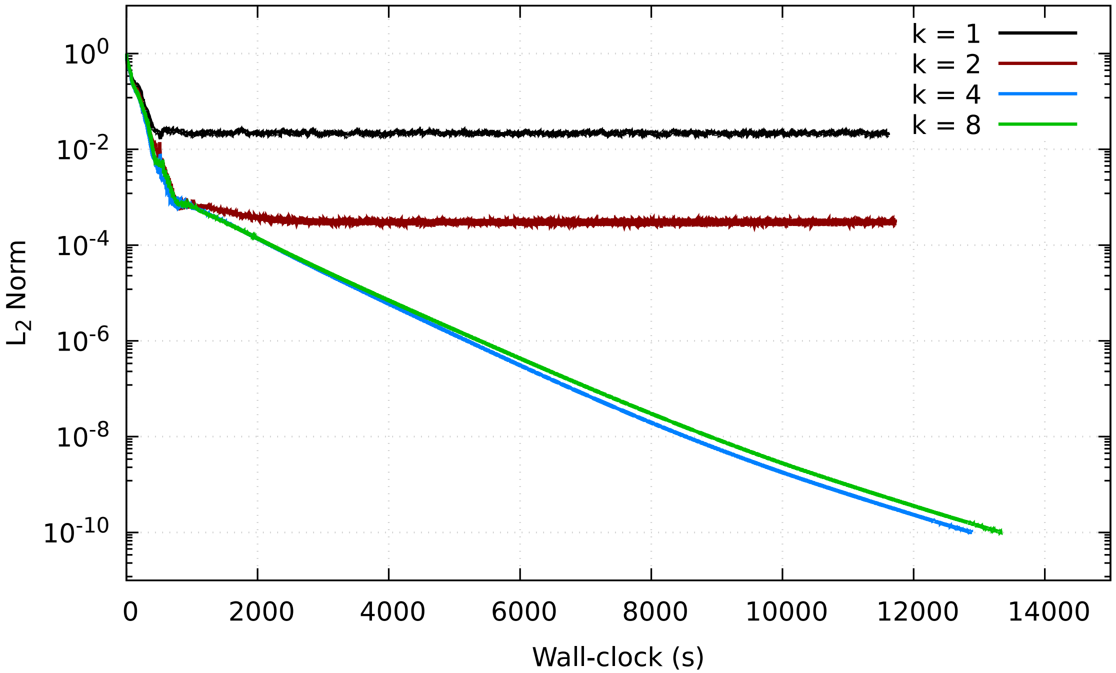
Results: JFNK Convergence History



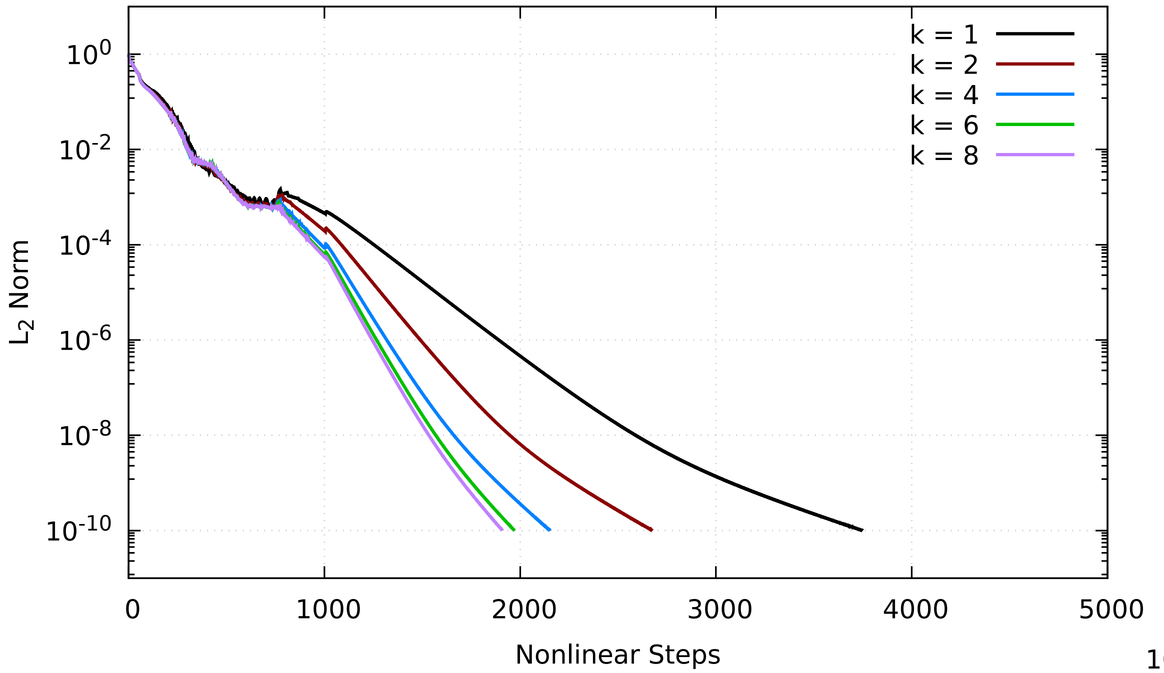


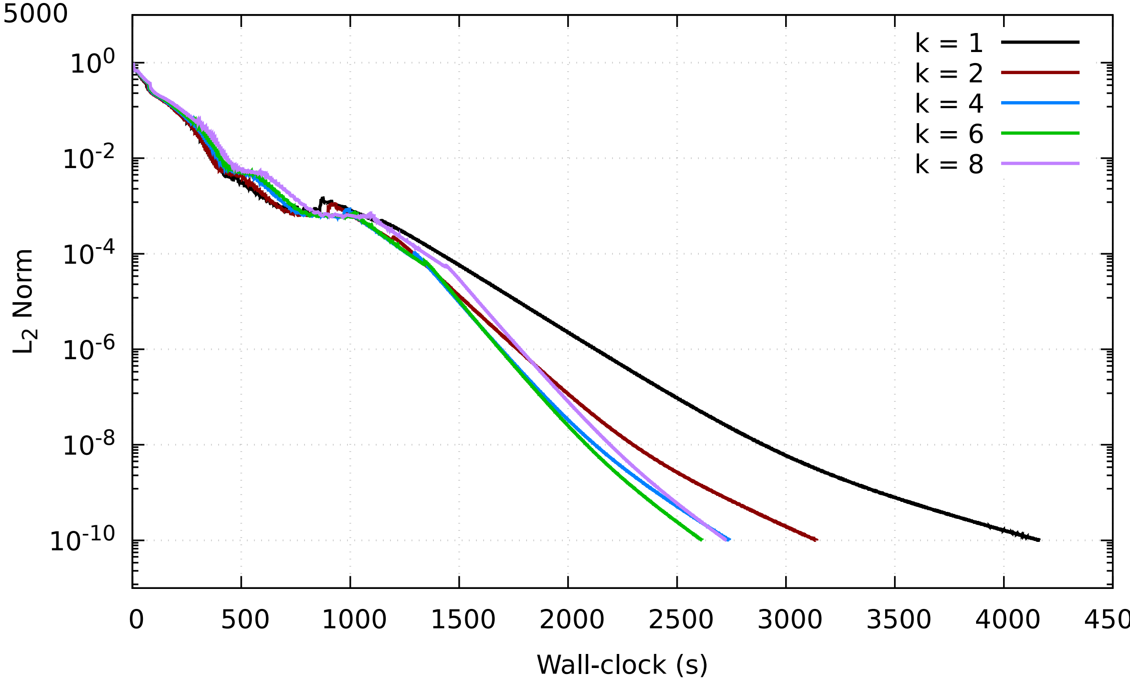
Results: Jacobi Convergence History



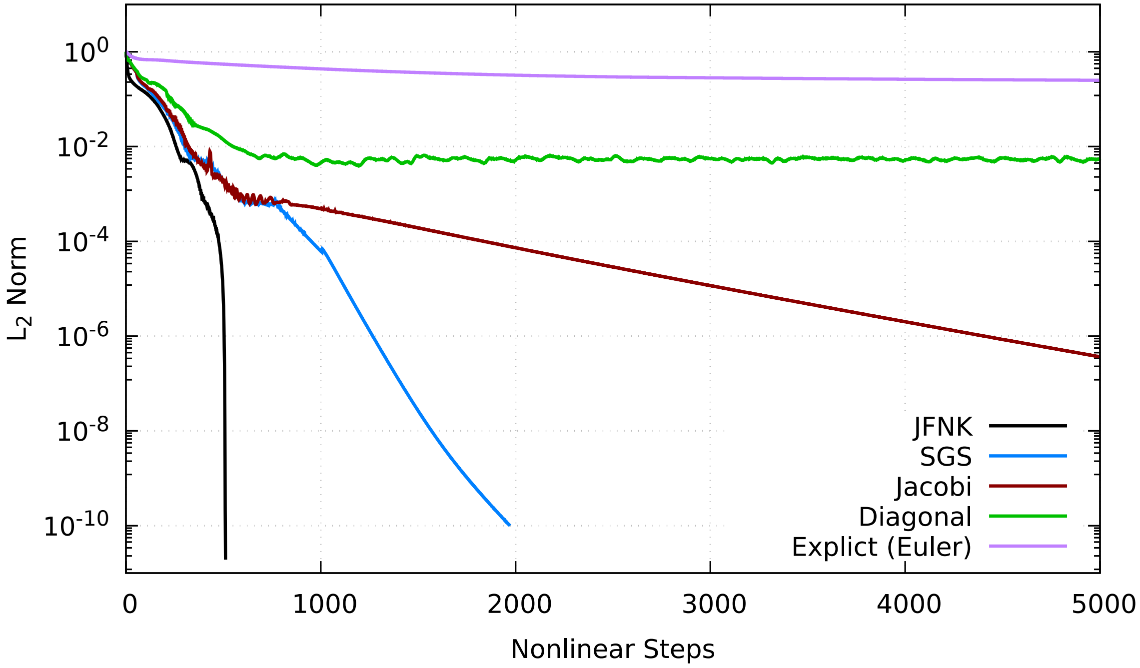


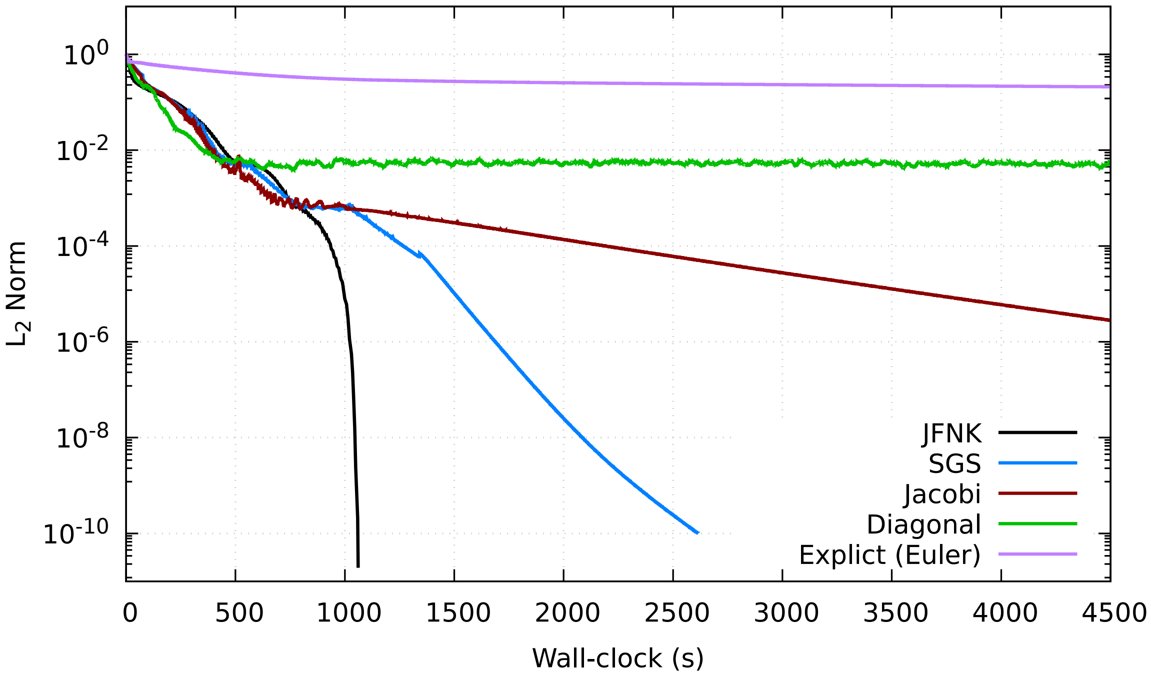
Results: SGS Convergence History





Results: Comparison of Convergence History

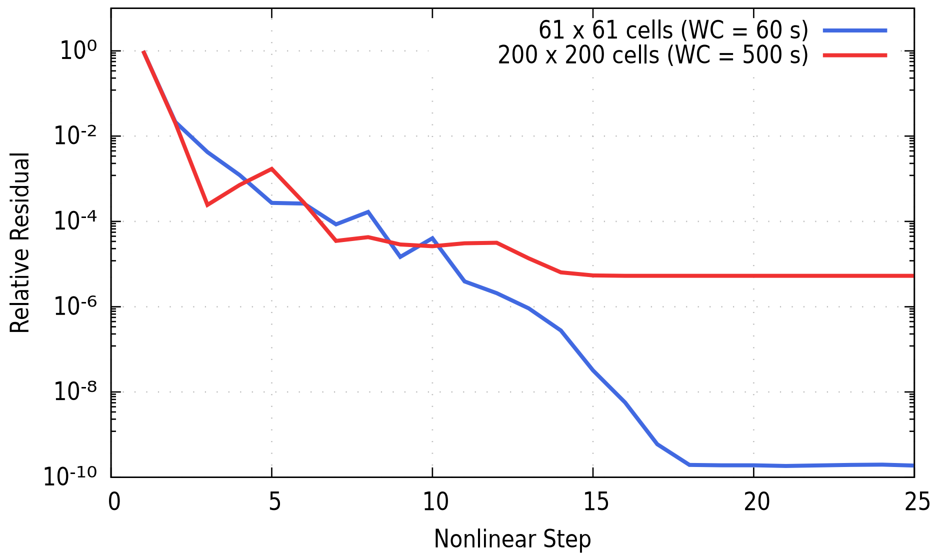


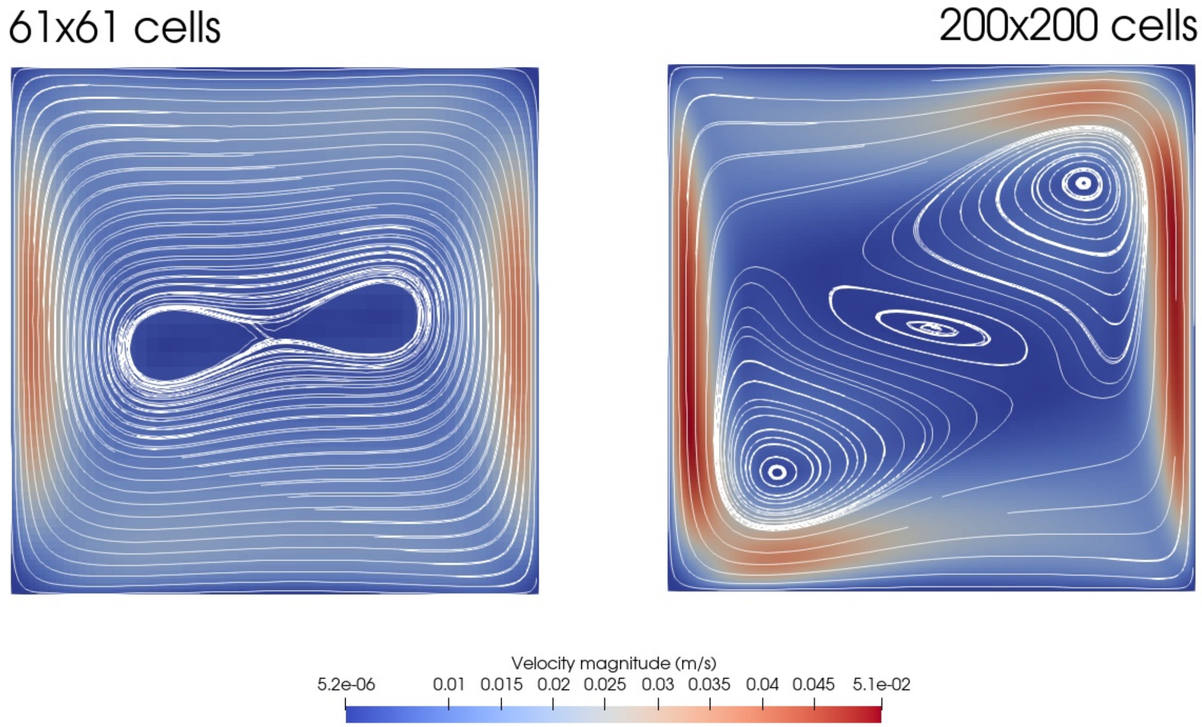


Recent JFNK examples: Natural Convection

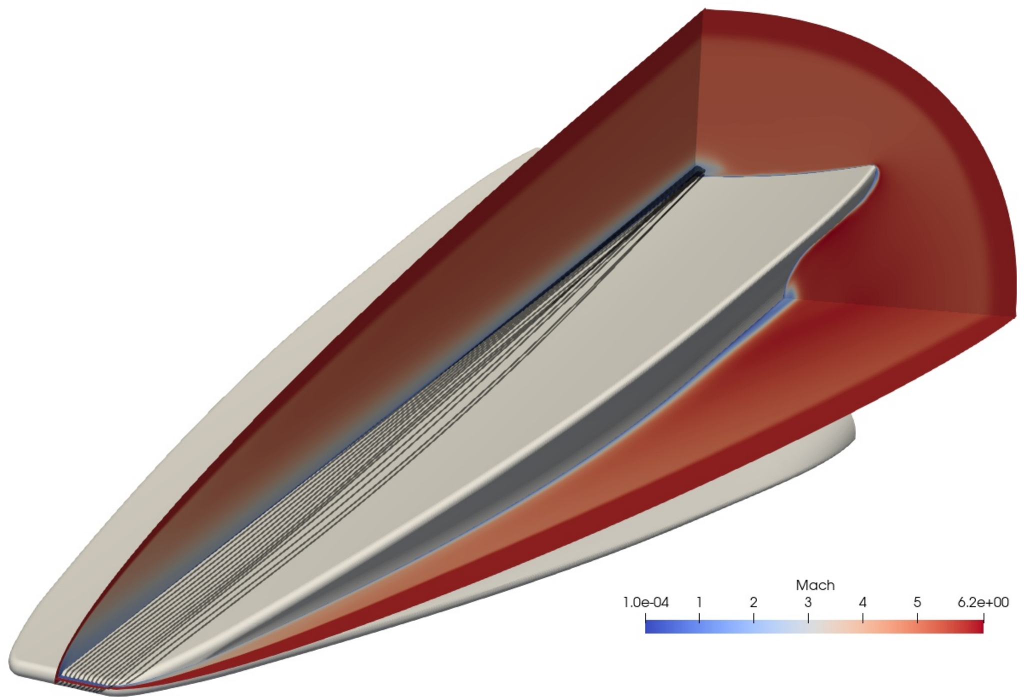
Simulation details:

- 2D structured grid solver
- AUSMDV flux scheme
- CFL_0 = 1,000,000
- Employed PJ's new gravitational term
- No Slip BCs:
  - -- North/South = Adiabatic
  - -- East = Fixed 293 K
  - -- West = Fixed 283 K



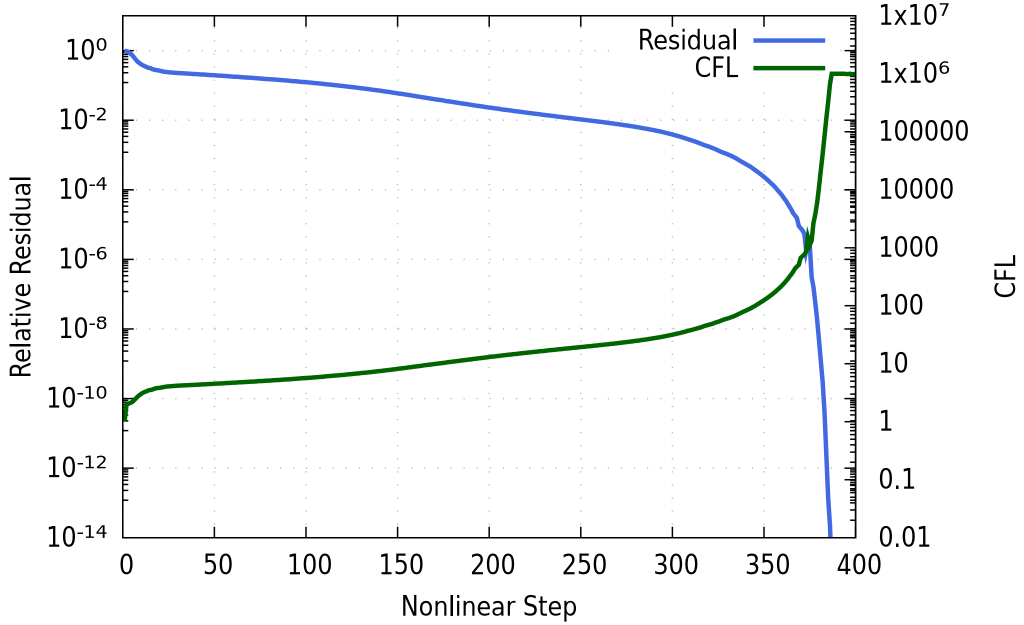


Recent JFNK examples: BoLT-II T4 model simulation

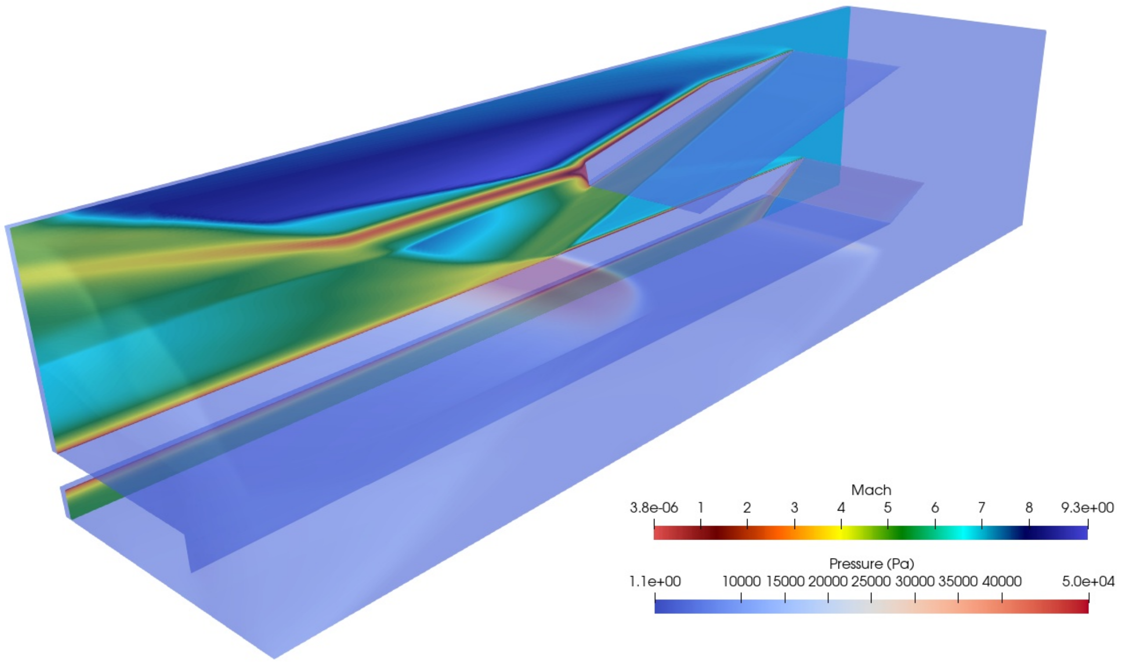


Simulation details:

- 3D unstructured grid solver
- LDFSS2 flux scheme
- Park heuristic limiter
- SA turbulence model (fully turbulent)
- Approx. 1 million cells
- Solved in ~1 hour on 32 core workstation

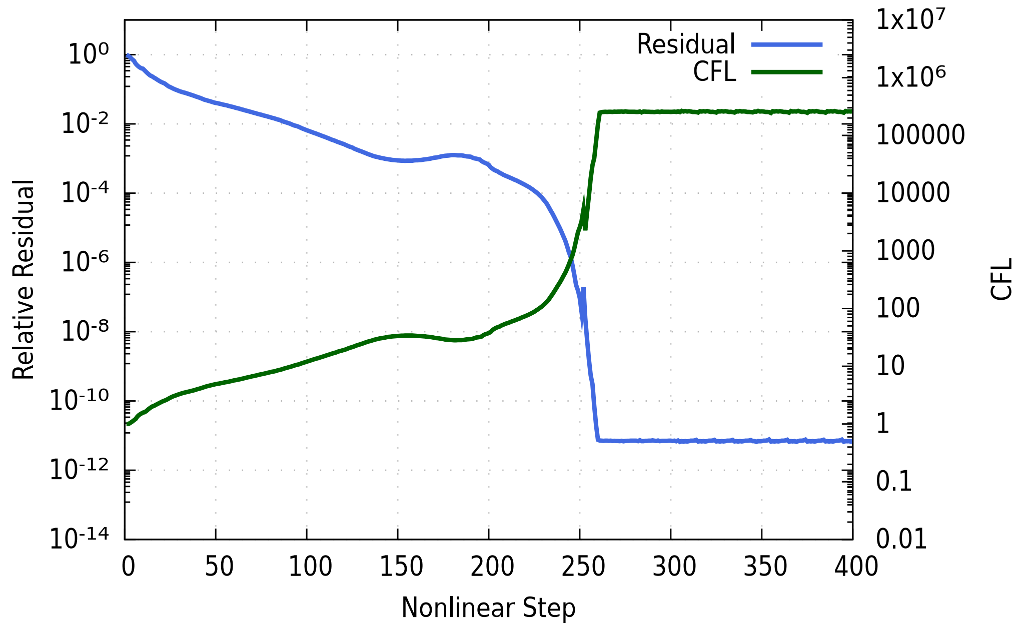


Recent JFNK examples: SWTBLI T4 model simulation



Simulation details:

- 3D unstructured grid solver
- HLLC flux scheme
- hvenkat limiter
- SA turbulence model (fixed transition)
- Approx. 7.5 million cells
- Solved in ~1.5 hours on 240 cores



Summary: Take home message

- All methods covered today give decent ***engineering level*** convergence
- Newton-Krylov method is superior for deep convergence (e.g. needed for adjoint)
- Automatic CFL growth achieved via Newton-Krylov method is valuable

Questions?