Multiphysics time-integration for turbulent combustion at the exascale

Marc Day

Nicholas Wimer, Malik Hassanaly, Lucas Esclapez, Anne Felden, Julia Ream, Marc Henry de Frahan, Jon Rood

11 July 2024

SIAM Annual Meeting (AN24)

Spokane, WA

Pele Combustion Suite

Pele is the Exascale Computing Project's (ECP's) application suite for high-fidelity detailed simulations of **turbulent combustion.** Pele supports detailed physics and geometrical flexibility to evaluate design and operational characteristics of clean, efficient combustors for automotive, industrial, and aviation applications, including:

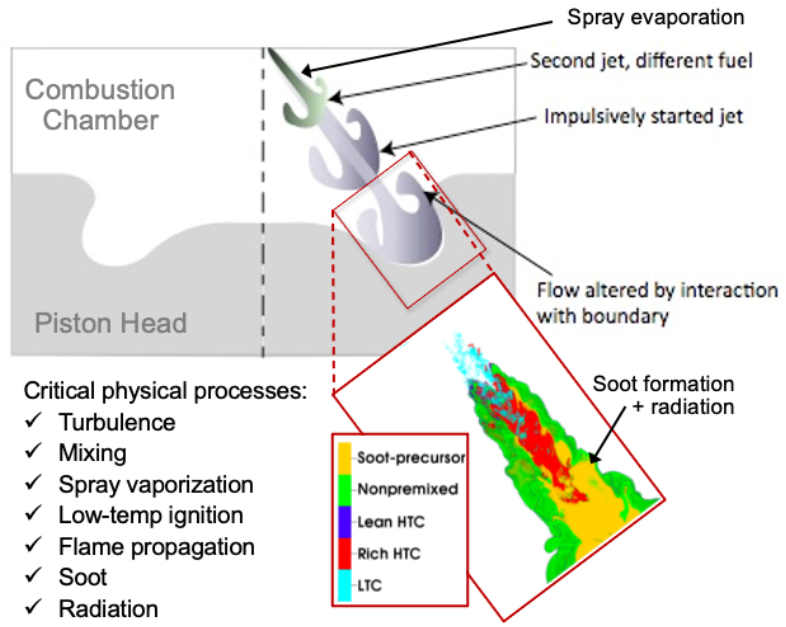
- Advanced internal combustion engines (e.g., RCCI)
- Novel supercritical CO_2 power cycles
- Rotating detonation engines
- Supersonic cavity flame holders
- Aviation combustors for sustainable drop-in JetA replacements

Pele combustion simulation and analysis suite:

- ***PeleC*** (compressible)
- ***PeleLMeX*** (low Mach) – replaces PeleLM
- ***PelePhysics*** (thermodynamics, transport, chemistry models)
- ***PeleAnalysis*** (in-situ, post-processing/analysis)
- ***PeleMP*** [multi-physics] (soot, radiation, Lagrangian sprays)
- ***PeleProduction*** (collaboration hub)

https://github.com/AMReX-Combustion

Pele's KPP2 Combustion Challenge Problem: RCCI



PeleC, PeleLMeX

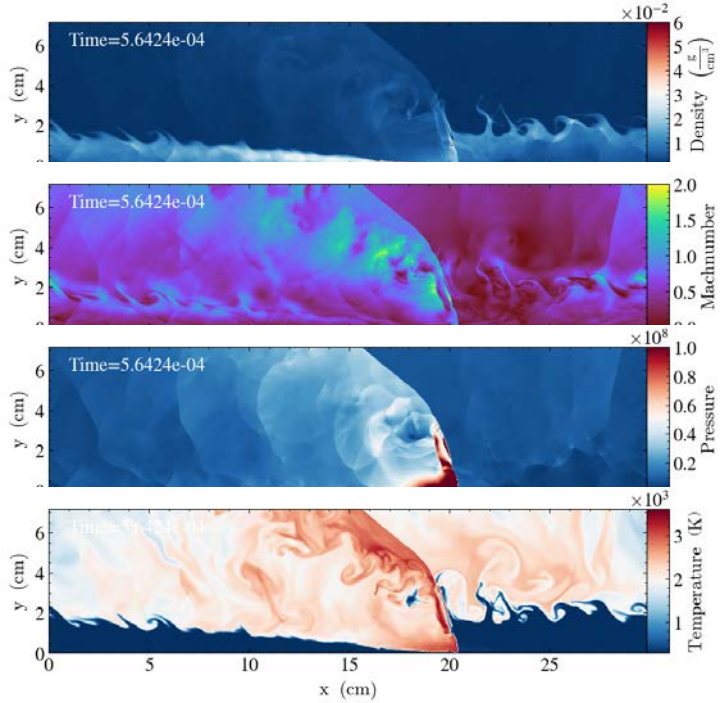
PelePhysics Transport, thermodynamics, finite-rate chemistry	**PeleMP** Multiphase (sprays) , Soot, Radiation	**AMReX-Hydro** Hydrodynamics and geometry	**SUNDIALS** Implicit/explicit ODE integrators **MAGMA** Batched linear solvers **HYPRE** Distributed linear solvers
AMReX Block-structured AMR library			

PeleC

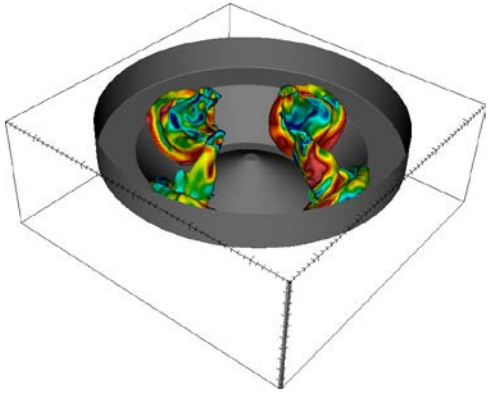
PeleC: Compressible reacting flow solver

- Conservation of species mass, momentum, total energy
- Embedded boundary capability for internal/external flows with complex geometries
- Time-explicit Runge-Kutta (RK)-based advance
  - Time-explicit RK variants for diffusion and hyperbolic (PPM, PLM, WENO, MOL)
  - SUNDIALS-driven ODE integration for finite-rate chemical kinetics (CVODE, ARKODE)
- **PelePhysics** provides finite-rate chemistry models, equations of state and transport properties, non-ideal thermodynamics and chemistry modifications, and tabulated lookup table and neural-net models available
- **PeleMP** provides access to optional multiphase (spray) fuel models via AMReX particle capability, in addition to soot and radiation models

https://github.com/AMReX-Combustion/PeleC



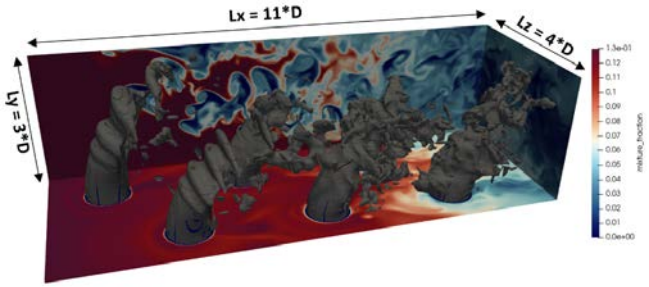
Rotating detonation engine
Sreejith NA et al., early results, 2022



Quad fuel jet into shaped piston bowl
Henry de Frahan et al., early results, 2022

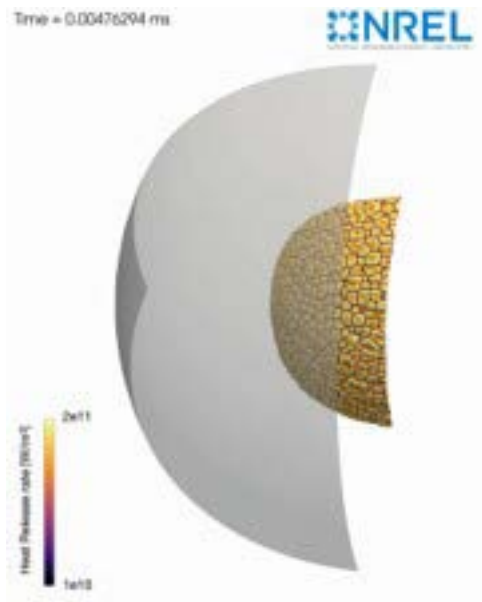
Supersonic cavity flame holder
Sitaraman et al., Combustion and Flame, 2021

PeleLMeX



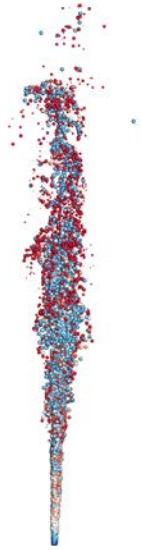
Gas turbine premixer, PeleLM
M. Vabre, B. Savard, et al.,
CICS Spring Technical Meeting, 2022

Spherical confined flame, PeleLMeX
L. Esclapez et al., in prep, 2022



PeleLMeX: Low Mach reacting flow solver

- Conservation of species mass, momentum, enthalpy
- Embedded boundary capability for internal/external flows with complex geometries
- Iterative/implicit SDC-variants for time-stepping tightly coupled ADR systems resulting from **large** dt (enabled by the low Mach algorithm)
  - Semi-implicit (Crank-Nicolson) diffusion, monotonic Godunov hyperbolics
  - SUNDIALS-driven implicit ODE integration for finite-rate chemical kinetics
- **PelePhysics** provides finite-rate chemistry models, equations of state and transport properties. Tabulated lookup table and neural-net-based models available
- **PeleMP** provides access to optional multiphase (spray) fuel models (Lagrangian, AMReX particles based), moment-based soot models, and radiation transport
- Available LES models include Smagorinsky, WALE, and Sigma
- ***PeleLMeX's non-subcycled integrator supports AMR with closed-chamber pressurization due to fueling and heat release***



Lagrangian fuel sprays in PeleLMeX
(droplets colored by T)
Ariente et al., in prep, 2022

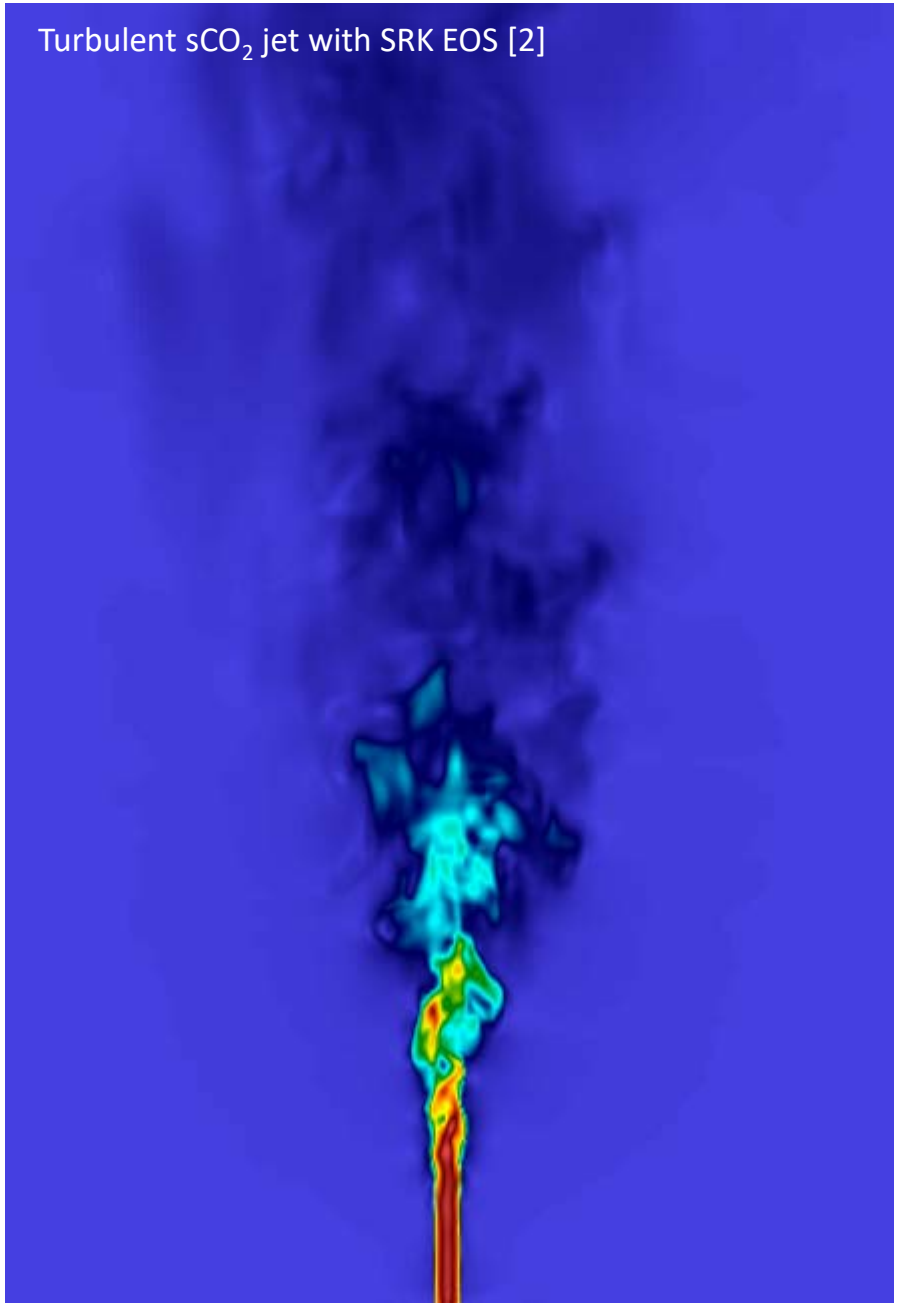
Quad n-dodecane jets into KPP PB
Wimer, Esclapez, et al., in prep, 2022

https://github.com/AMReX-Combustion/PeleLMeX

PelePhysics

An open-source combustion physics library

- https://github.com/AMReX-Combustion/PelePhysics
- EOS: ideal gas mixtures (CHEMKIN), Soave-Redlich-Kwong (SRK), and EOS lookup tables and/or NN's
- Models and parameters for thermodynamics
- Mixture-averaged and unity Le transport properties, including extensions for non-ideal gases
- Finite-rate chemistry integration via SUNDIALS
- Python-based C++ generator, CEPTR, converts combustion chemistry models into production rate and reaction Jacobian code for CPU/GPU evaluation



[2] Ream et al., SIAM CSE, 2021

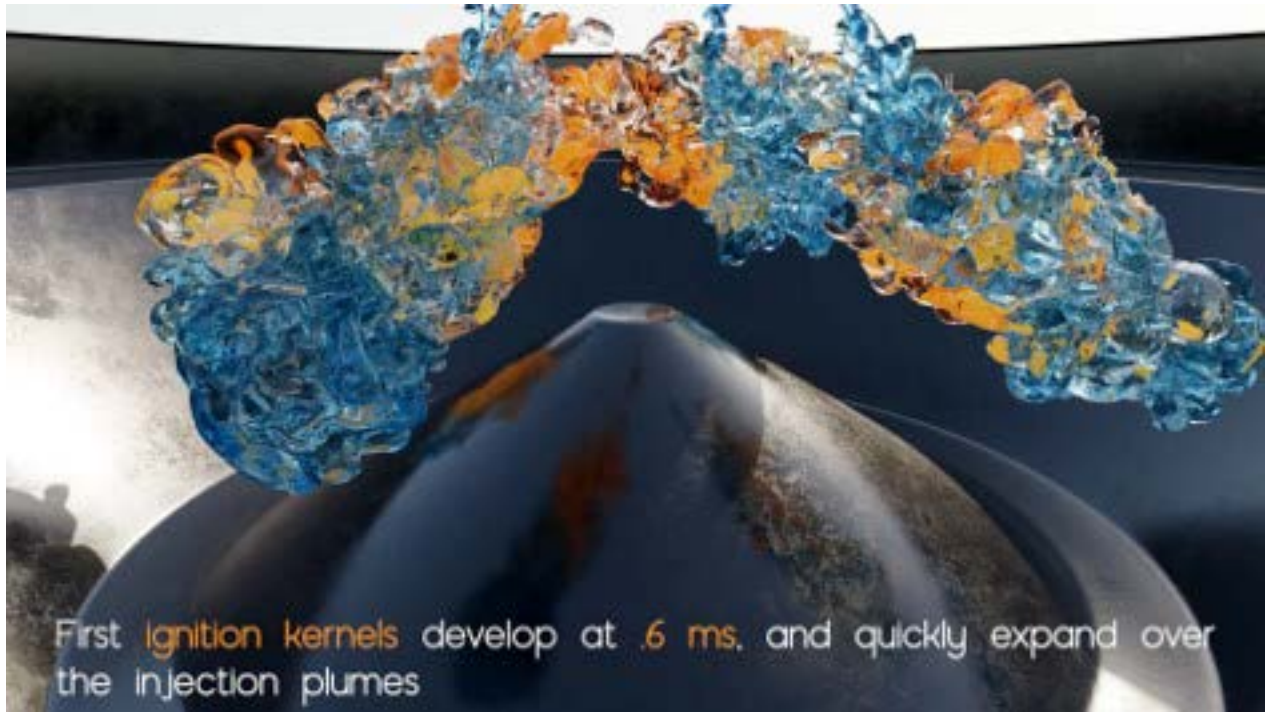
Pele Suite Exascale Applications

Reacting Flows CFD Challenge:

- CFD of real-world IC engines requires **high parallelism** and **significant computational resources**
- High-resolution reactivity-controlled combustion ignition (RCCI) engine
- Our exascale demo required over **20 billion grid cells** and was scaled over **56,000 AMD MI250X GPUs** using ORNL's Frontier supercomputer

Chemistry Integration is EXPENSIVE!

- The **majority of the computational cost** was used for the evaluation of the **53 species** chemical reaction mechanism
- Runs of this scale become intractable for design studies, parameter variations, or other *outer-loop* investigations



High-Resolution Pele simulation of an RCCI engine

Nicholas T. Wimer, Lucas Esclapez, Nicholas Brunhart-Lupo, Marc Henry de Frahan, Mohammad Rahimi, Malik Hassanaly, Jon Rood, Shashank Yellapantula, Hariswaran Sitaraman, Bruce Perry, Michael Martin, Olga Doronina, Sreejith N. Appukuttan, Martin Rieth, and Marc Day
Phys. Rev. Fluids **8**, 110511 – Published 17 November 2023

How can we speed up the chemical evaluator?

Pele: Discretization Notes

- Flames in low-speed flows (w/no detonations): $(\mathcal{D}+\mathcal{R})_{stiff}$ in a Lagrangian frame moving at speeds comparable to the flame speed, S_L
  - $\mathcal{R}$ creates radicals, $\mathcal{D}$ spreads them: $\tau_{stiff} \sim 10^{-12} - 10^{-7}$ s
  - Advective (e.g. turbulence), $\mathcal{A}$ processes: $\tau_{Adv} \geq 10^{-5}$ s
  - Pele codes explicitly timestep advection at Δt_{CFD} and advance $\mathcal{S}_{chem}(\mathcal{A},\mathcal{D})$ w/SUNDIALS

- Thermodynamic pressure stays spatially constant, $p_{th} = p_{th}(t)$
  - Evolving flow satisfies $\nabla \cdot U = S$. This is a constraint on the velocity that is NOT automatically satisfied by conservation.
  - Our finite-volume scheme advects mass, energy with predicted face- and time-centered centered **velocities that satisfy the divergence constraint**

PeleC: Compressible Conservation Equations (PeleLMeX similar, but low Mach)

$$\frac{\partial U}{\partial t} + \nabla \cdot \Gamma = S$$

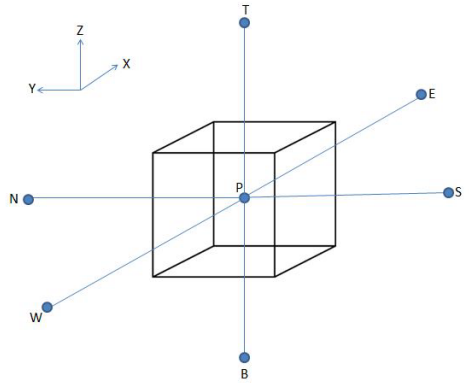
$$U = [\rho,\ \rho\vec{v},\ \rho E,\ \rho\mathcal{Y}_k,\ \rho\mathcal{A}_\ell,\ \rho\mathcal{B}_m]$$

Chemistry sources

$$\Gamma = [\rho\vec{v},\ \rho\vec{v}\vec{v} + P,\ \rho E\vec{v} + P\cdot\vec{v} + \mathcal{Q} + \mathcal{Q}_{rad},\ \rho\mathcal{Y}_k\vec{v} + \mathcal{F}_k,\ \rho\mathcal{A}_\ell\vec{v},\ \rho\mathcal{B}_m\vec{v}]$$

$$S = [S_\rho,\ \rho\vec{g} + S_{\rho\vec{v}},\ \rho\vec{v}\cdot\vec{g} + S_{\rho E},\ \rho\dot{\omega}_k + S_{\rho\mathcal{Y}_k},\ S_{\rho\mathcal{A}_\ell},\ S_{\rho\mathcal{B}_m}]$$

$P = p\mathcal{I} + \Pi$, $(\mathcal{F}_k, \Pi, \mathcal{Q})$: *diffusive transport*, S_*: *external sources (e.g. evaporated fuels)*



Finite Volume Discretization with AMR

$$\frac{\partial}{\partial t}\iiint_\Omega U dV + \iint_{\partial\Omega} \Gamma \cdot dA = \iiint_\Omega S dV$$

$$\rightarrow \quad \frac{\partial \mathcal{U}}{\partial t} = F_{AD} + \mathcal{S}_{chem}$$

PeleC Time Advance (PeleLMeX similar w/projections, implicit diffusion)

Second-order predictor-corrector split time-advance

- $\mathcal{U}' = \mathcal{U}^n + \Delta t\left(F_{AD}\left(\mathcal{U}^n\right) + \mathcal{S}^n_{chem}\right)$
- $\mathcal{U}'' = \mathcal{U}^n + \frac{\Delta t}{2}\left(F_{AD}\left(\mathcal{U}^n\right) + F_{AD}\left(\mathcal{U}'\right)\right) + \Delta t \cdot \mathcal{S}^n_{chem}$
- $F = \frac{1}{2}\left(F_{AD}\left(\mathcal{U}^n\right) + F_{AD}\left(\mathcal{U}''\right)\right)$
- $\mathcal{S}^{n+1}_{chem} = \mathcal{S}\left(\mathcal{U}^n, F\right)$ *Local, provided by SUNDIALS*
- $\mathcal{U}^{n+1} = \mathcal{U}^n + \Delta t\left(F + \mathcal{S}^{n+1}_{chem}\right)$

Pele -> SUNDIALS

Pele provides the following to SUNDIALS:

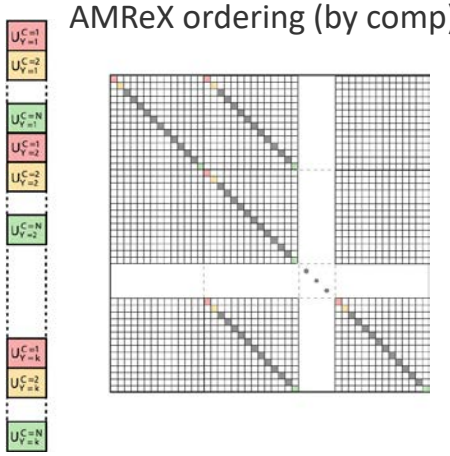
- $\mathcal{S}_{chem,k} = \rho\dot{\omega}_k = \sum_{n=1}^{N_R}\left(K_{Fn}\prod_{j=1}^{N_s}[X_j]^{\nu'_{kj}} - K_{Rn}\prod_{j=1}^{N_s}[X_j]^{\nu''_{kj}}\right)$
- State component scales via “typical values” (component-wise absolute and relative tolerance)

$$a_{\mathrm{tol,i}} = a_{\mathrm{tol}} \cdot \max\left(\frac{1}{2}(\min(y_i) + \max(y_i)), \tilde{y}_i\right)$$

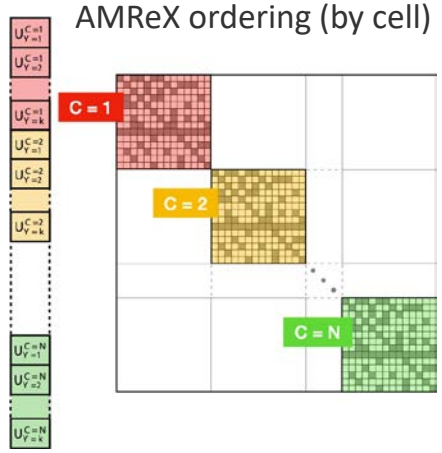
…and for implicit solves (the usual case):

- Jacobian, $\mathcal{J}_{ki} = \dfrac{\partial \mathcal{S}_{chem,k}}{\partial(\rho\mathcal{Y}_i)}$
- Preconditioner (approximate $\mathcal{J}_{ki}^{-1}$)

AMReX ordering (by comp)



AMReX ordering (by cell)



NOTE: Data ordering impacts sparsity pattern for Jacobian

Done. Right?

Issues:

1. In combustion, we are changing fuels all the time. New models require new code for the production rates, Jacobians, preconditioners, etc. Producing these routines by hand is tedious and error-prone
2. The robustness of Newton-Krylov schemes depends heavily on the quality of the preconditioner used. Forming a good preconditioner requires much of the same infrastructure as forming the Jacobian itself.
3. The QSSA (discussed next) provides a rigorous approach to reducing the size of models, at the expense of dramatically complicating the coding tasks
4. We have found numerical Jacobians are a viable path in general, but they tend to be extremely sensitive to variable scaling issues, and difficult to manage generically

Quasi-Steady-State Approximation (QSSA)

- Chemical evolution timescales can vary many orders of magnitude across species
  - Timescales of fuel consumption, product generation typically match those of the flow
  - Production/destruction of intermediates can be ten (or more) orders of magnitude faster
  - Robust/accurate integration of such *stiff* systems is not practical
- The **Quasi-Steady State Approximation (QSSA)** assumes species, *K*, with short characteristic timescales and low concentrations at always at **steady-state:** $S_{chem,K} \approx 0$
  - QSSA species are NOT transported in Pele
  - Cellwise, QSSA species concentrations become algebraic expressions/constraints as a function of the rest of the state as it evolves
- With QSSA the size of the chemical reaction mechanism (number of species and reactions) and stiffness can be **drastically reduced**
- **"Science" Research Questions:**
  - What is the benefit of explicitly forming the Jacobian (vs using a Newton-Krylov type strategy)?
  - What is the impact of the QSSA on the accuracy of predictions in our demo problem?

CEPTR Jacobian Implementation

The second term is difficult due to chain rule terms and long expressions

$$J_{i,j} = \left(\frac{\partial \dot{\omega}_i}{\partial C_j}\right)_{CQSS,k} + \left(\frac{\partial \dot{\omega}}{\partial C_{QSS,k}}\right)_{C_j} \frac{dC_{QSS,k}}{dC_j}$$

- Chemical mechanisms are large and stiff systems
- Integration is typically implicit by means of a **reaction Jacobian**

- Non-QSSA mechanisms can simply accumulate contributions to the Jacobian one reaction at a time
- With QSSA, there is a set of **non-QSSA species** (transported) and a set of **QSSA species** (algebraically related to the transported species)
- Use **SymPy** to symbolically track the full derivatives of each of the QSS and non-QSS species interactions
- The final Jacobian contains **many repeated calculations** from the chain rule nature of interactions

```
// d()/d[NC12H26]
dqdci = +k_f * sc[9];
J[0] -= dqdci;  // dwdot[NC12H26]/d[NC12H26]
J[9] -= dqdci;  // dwdot[CH3]/d[NC12H26]
J[10] += dqdci; // dwdot[CH4]/d[NC12H26]
// d()/d[CH3]
(-x1296 * x555 - x2939 * x555 - x555 * x771 - x555 * x833 - x555 * x964 -
```

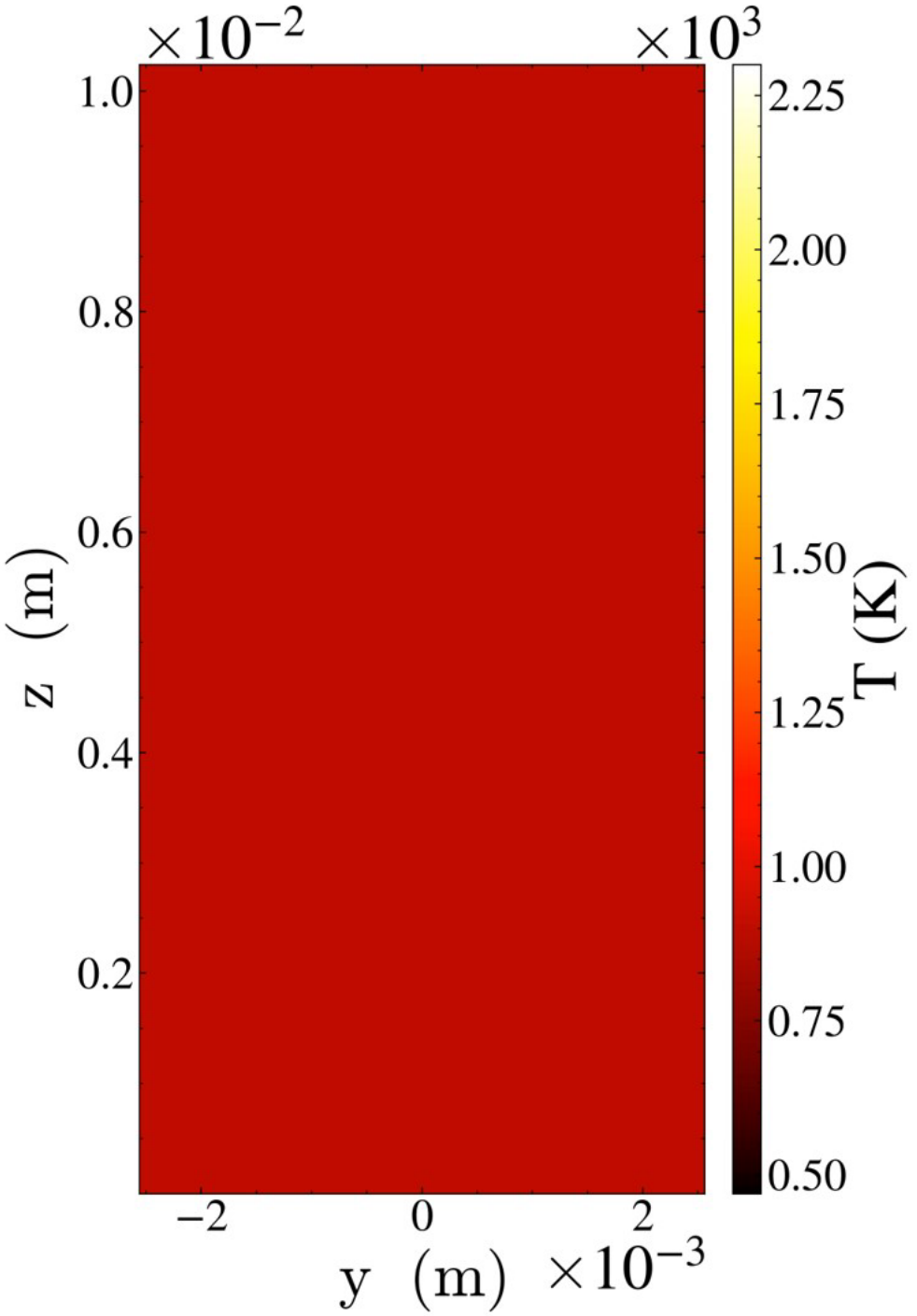
Computational Setup

- We set up a **multi-pulse n-dodecane jet** that is reminiscent of the RCCI conditions
- Skeletal mechanism is **53 species** and QSSA mechanism is **35 species**
- Domain is filled with **methane/air** mixture at **900 K** and 60 atm
- Fuel jet is **n-dodecane** at **470 K**
- Two pulses of fuel each for 0.5 ms with a 0.5 ms dwell time in between

Mechanism Sweep:

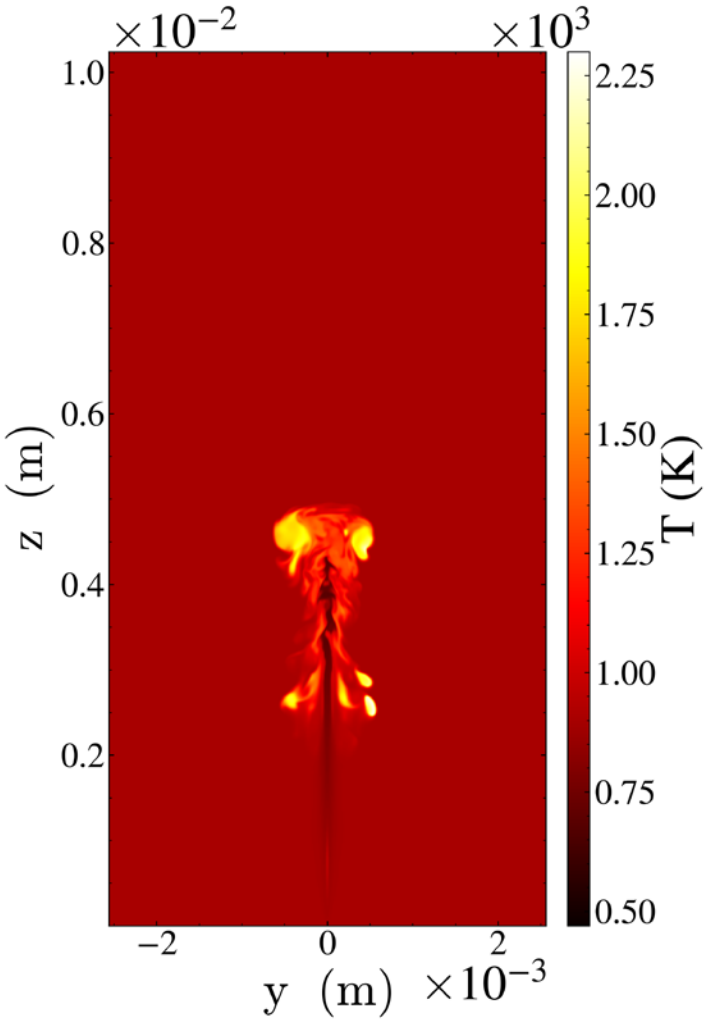
- Skeletal mechanism using traditional Newton-Krylov method
- Reduced QSS mechanism using traditional NK
- Reduced QSS mechanism using analytic Jacobian formulation
  - 96 different iterations on the pre-computed terms for memory optimization

- Approximately **150 million grid cells**

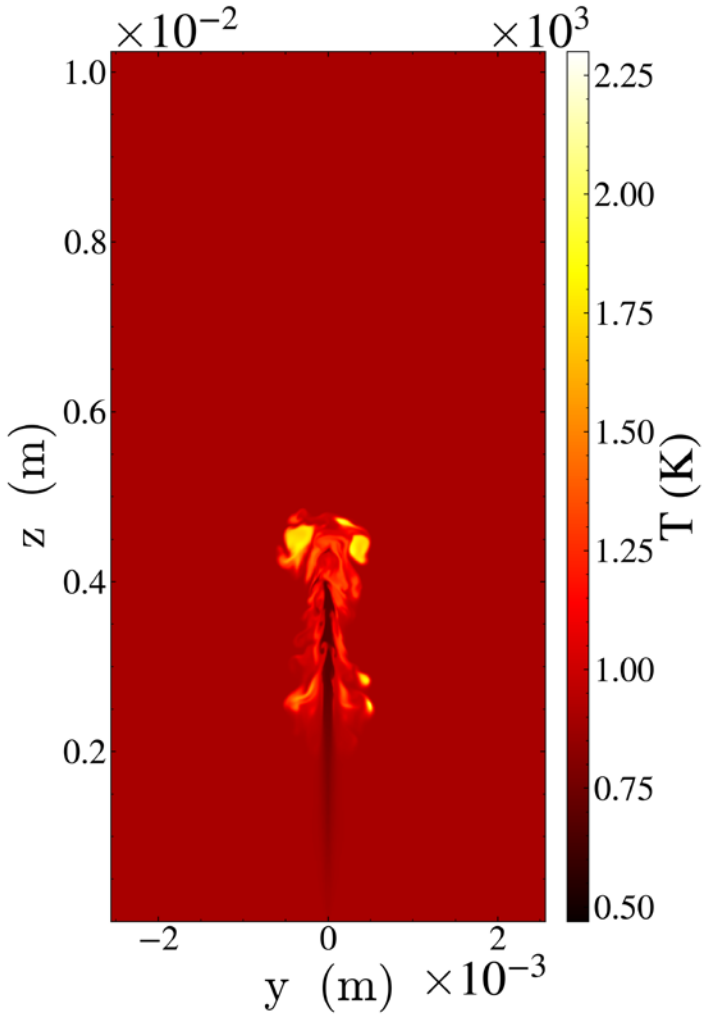


Qualitative Comparisons

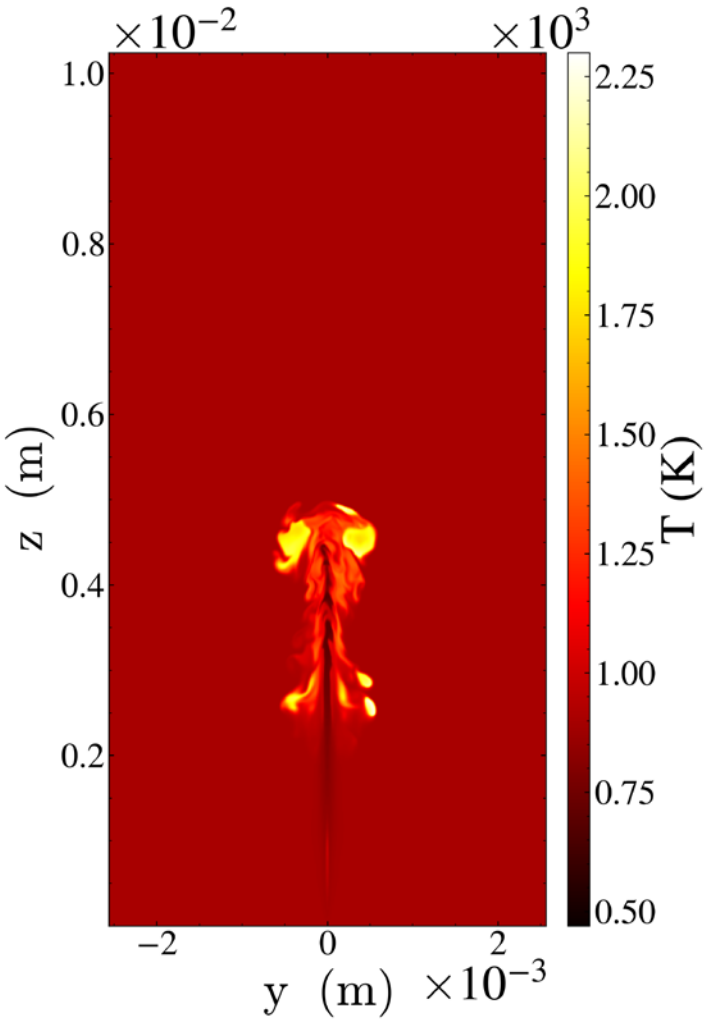
time: 0.75 ms



Skeletal: 162 s per timestep



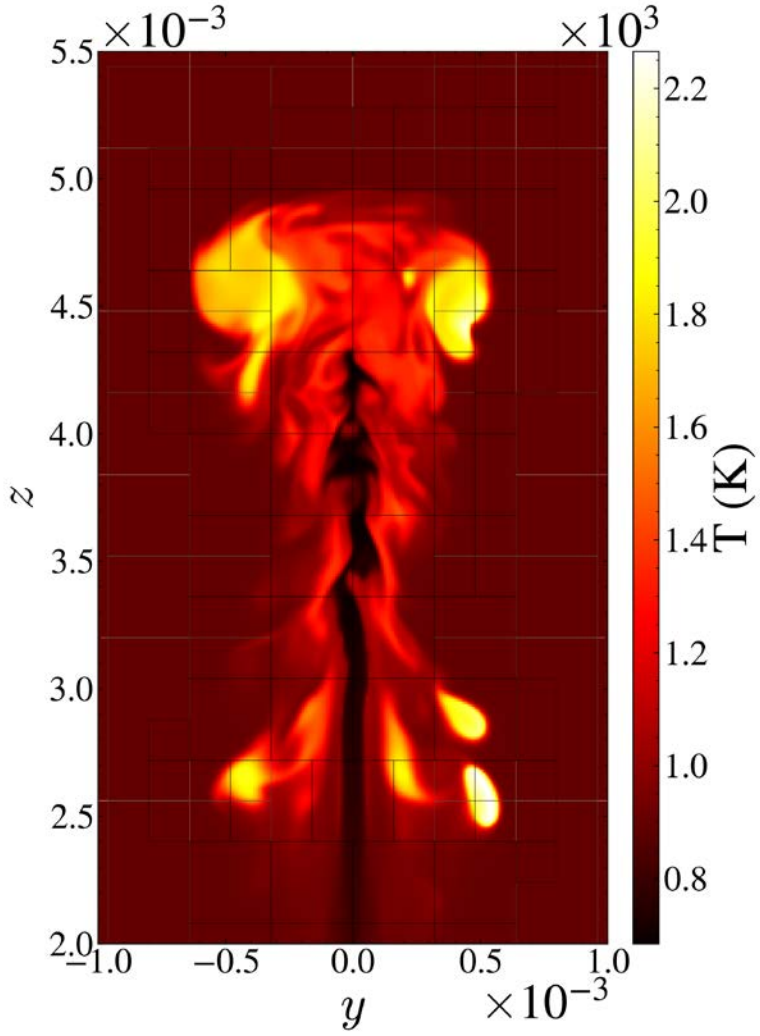
QSS NK: 80 s per timestep



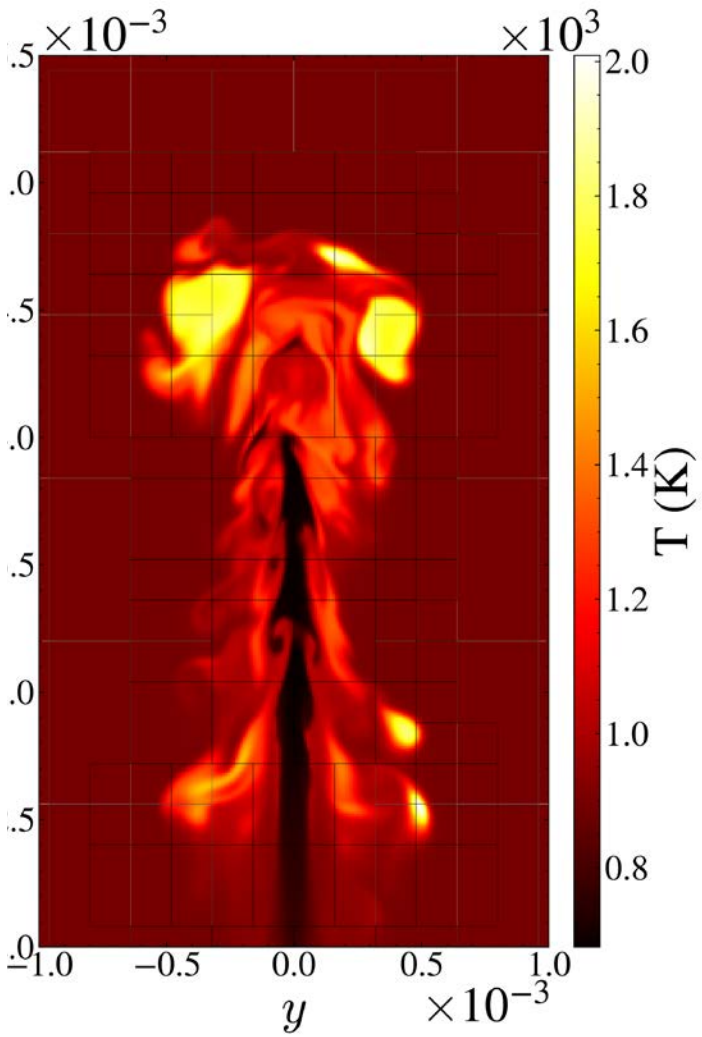
QSS AJ: 44 s per timestep

Qualitative Comparisons

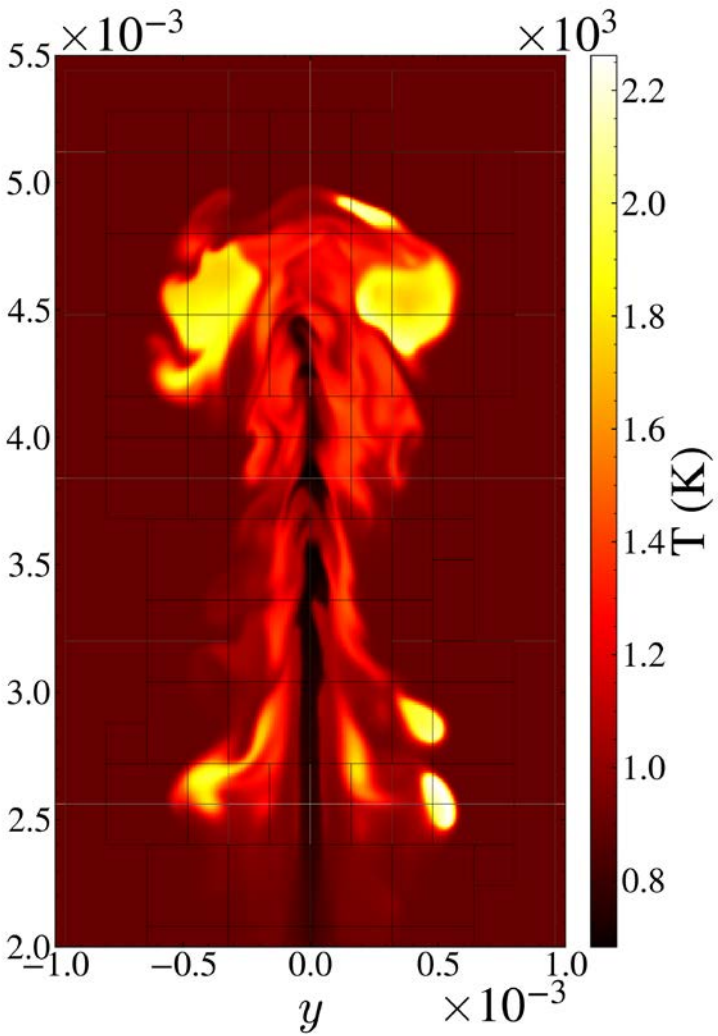
time: 0.75 ms



Skeletal: 162 s per timestep

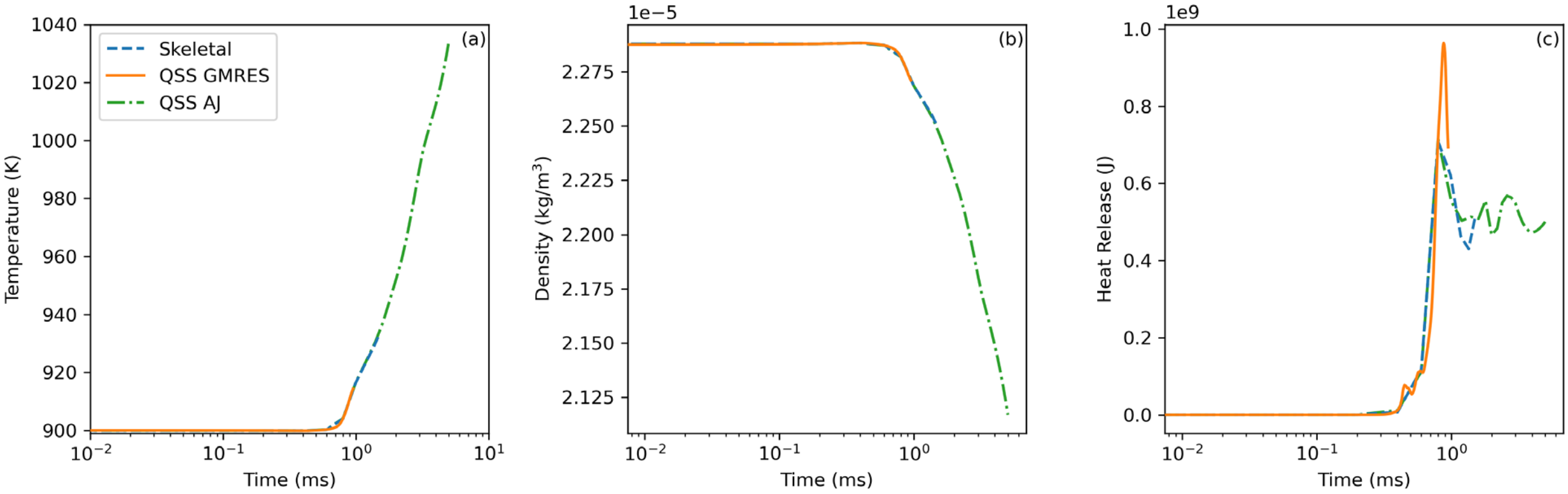


QSS NK: 80 s per timestep



QSS AJ: 44 s per timestep

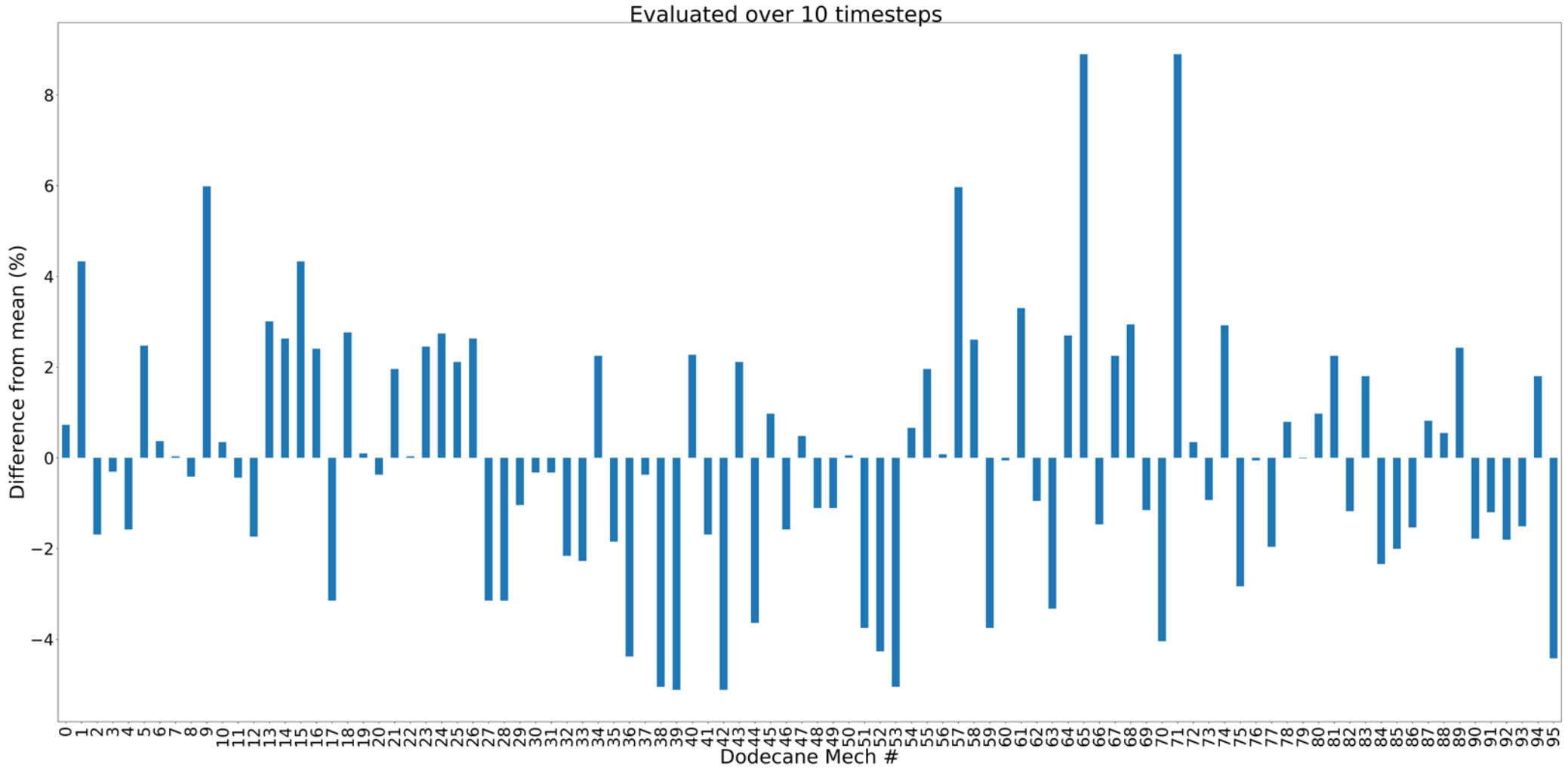
Volume Average Comparisons



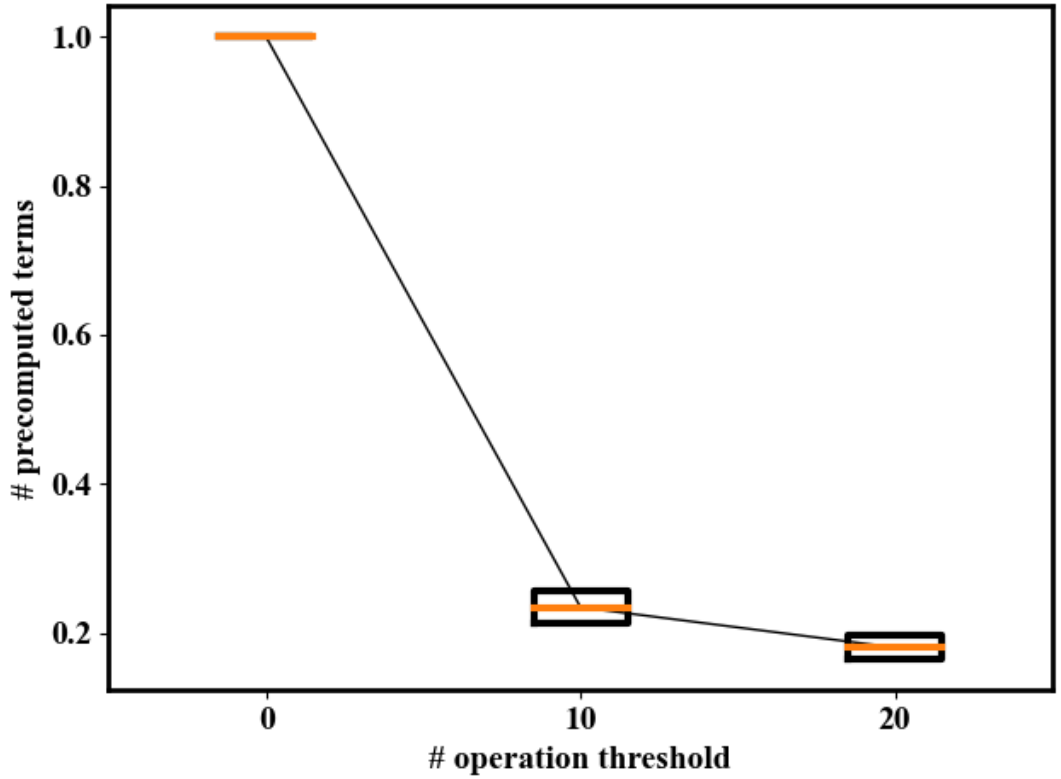
- Temperature and Density are very consistent
- Heat Release Rate shows the largest differences with **QSS NK as the main outlier**
- The QSS with **analytical Jacobian formulation** is in close agreement to the **Skeletal mechanism**

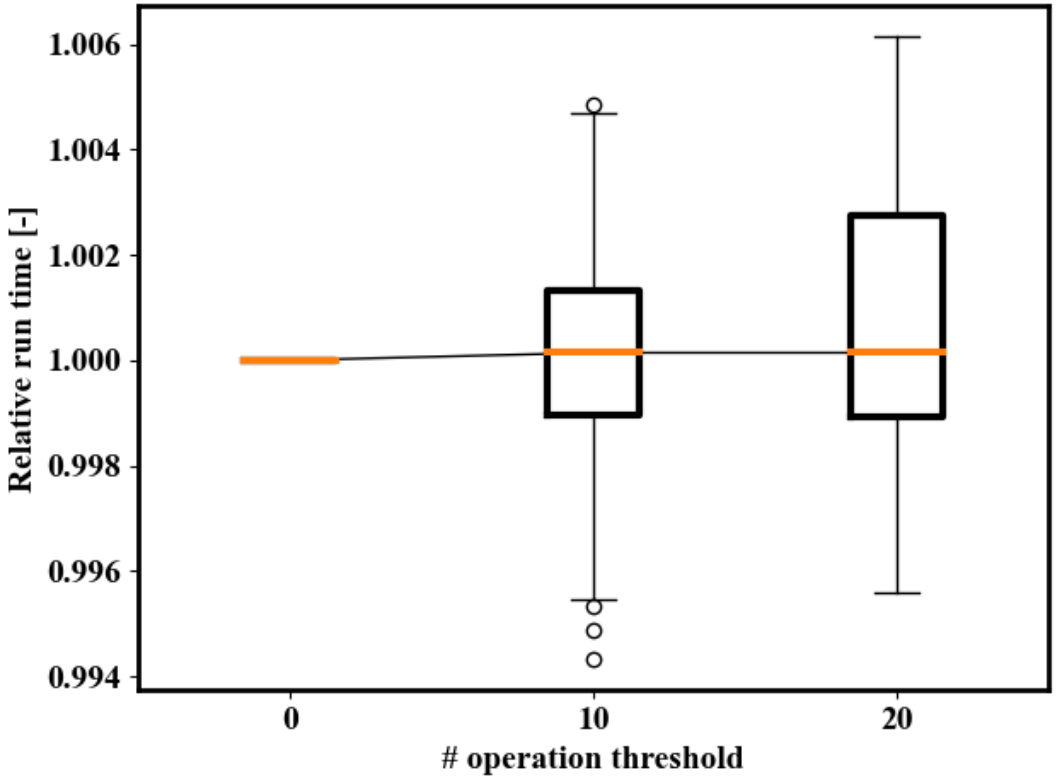
Chemical Mechanism Optimizations

- 96 different mechanisms with different levels of optimizations
- Compared to the mean computation time there can be variation of about +8%/-4.5%



Relative Performance

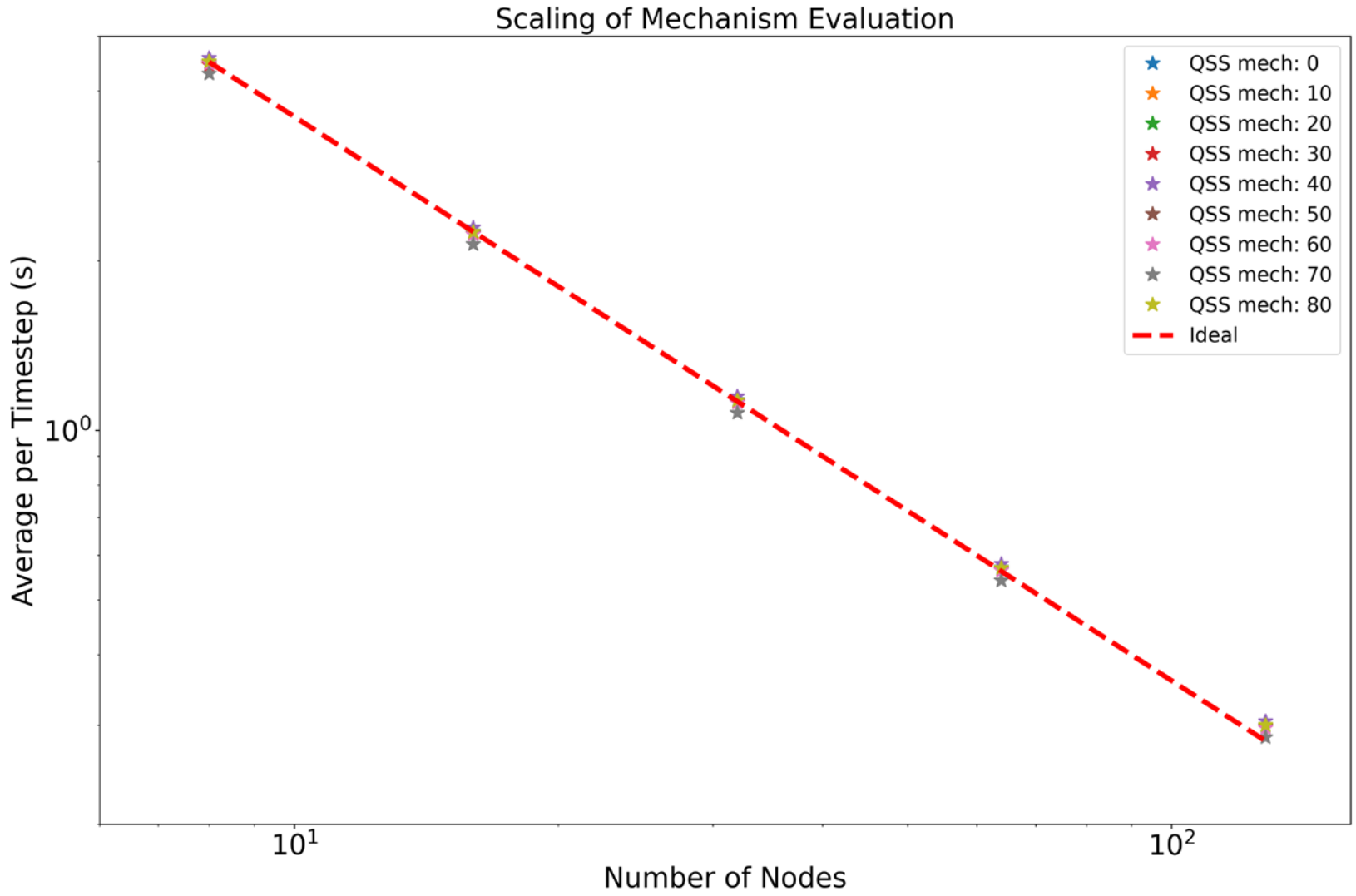




- **Consistent results** over a wide range of precomputations
- Therefore, we can make liberal **GPU optimizations without sacrificing performance** of the chemical mechanism evaluation

Strong Scaling of Chemistry Evaluation

- The simulation was run on a range of nodes from **8 – 128 (64 – 1024 GPUs)**
- The evaluation of the reaction mechanism timing is consistent with the **expected ideal**
- The reaction mechanism formulations **do not impact the scalability** of the Pele suite



Conclusions

- **Implementation of automated QSSA mechanism generator**
  - Using CEPTR in PelePhysics
  - We formulate an entirely **symbolic analytic Jacobian** using SymPy/SymEngine
  - We implement a series of optimizations to reduce memory pressure for **GPU optimizations** with little/no penalty in computational cost
- **Evaluated on multi-pulse, high-pressure jet**
  - Approximately 4x speed improvement with respect to skeletal and **2x with respect to Newton-Krylov**
  - Best results between full skeletal mechanism and QSS analytic Jacobian
  - Improved stability and runtime with analytic Jacobian (noted across a broad range of models/configurations now)

https://github.com/AMReX-Combustion

Thank you!

www.nrel.gov

NREL/PR-2C00-90542

This work was authored by the National Renewable Energy Laboratory, operated by Alliance for Sustainable Energy, LLC, for the U.S. Department of Energy (DOE) under Contract No. DE-AC36-08GO28308. Funding provided by U.S. Department of Energy Office of Science and National Nuclear Security Administration. The views expressed in the article do not necessarily represent the views of the DOE or the U.S. Government. The U.S. Government retains and the publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for U.S. Government purposes. This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. A portion of the research was performed using computational resources sponsored by the Department of Energy's Office of Energy Efficiency and Renewable Energy and located at the National Renewable Energy Laboratory. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.