Enabling next-generation combustion simulations by intelligent integration

Kyle Niemeyer
Assistant Professor
School of Mechanical, Industrial, & Manufacturing Engineering
Oregon State University

26 July 2017

kyle.niemeyer@oregonstate.edu
@kyleniemeyer
https://git.io/nrg

Today's talk

Today's talk

- Discuss challenges of incorporating detailed chemical kinetics models in multidimensional reacting flow simulations

Today's talk

- Discuss challenges of incorporating detailed chemical kinetics models in multidimensional reacting flow simulations

- Describe our efforts to reduce associated expense on modern computing architectures

Today's talk

- Discuss challenges of incorporating detailed chemical kinetics models in multidimensional reacting flow simulations
- Describe our efforts to reduce associated expense on modern computing architectures
- Summarize other current projects

Acknowledgements

Students

AJ Fillo

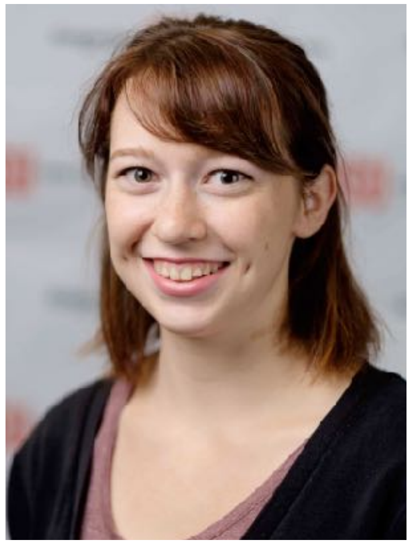
Matt Zaiger

Dan Magee

Luz Pacheco

Andrew Alferman

Tejas Mulky



Morgan Mayer

Phillip Mestas

Acknowledgements

Collaborators

University of Connecticut

Nick Curtis

Jackie Sung

Oregon State University

Shane Daly

Chris Hagen

Chris Stone
Computational Science & Eng. LLC

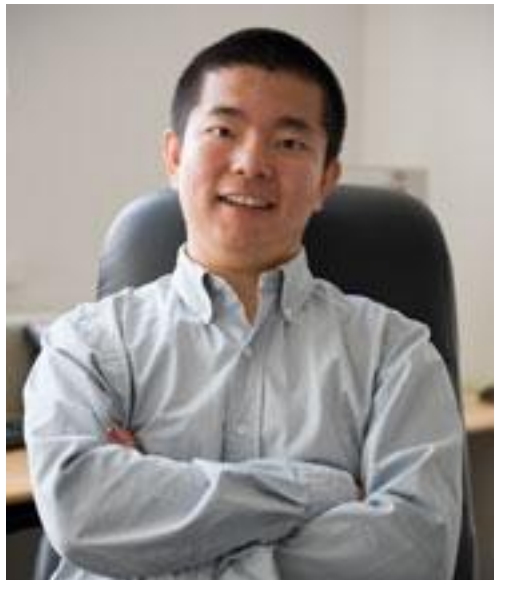
David Blunck

Acknowledgements

Collaborators

Bryan Weber, UConn

Peter Hamlington, CU Boulder



Qiqi Wang, MIT

Guillaume Blanquart, Caltech

Richard West, Northeastern

David Gleich, Purdue

Acknowledgements

Funders

SERDP
DOD · EPA · DOE

Chevron

NSF

DEPARTMENT OF ENERGY · UNITED STATES OF AMERICA ·

NETL

NASA

Challenge

Performing predictive simulations of reactive flows

Challenge

Performing predictive simulations of reactive flows
... in a **reasonable** amount of time

Challenge

Recent LES of diesel spray with 54-species *n*-dodecane model[1]:

[1]A. A. Moiz et al. *Combust. Flame* 173 (2016): 123–131. doi:10.1016/j.combustflame.2016.08.005

Challenge

Recent LES of diesel spray with 54-species *n*-dodecane model[1]:

48,000 CPU core-hours for 2 ms after start of injection

[1] A. A. Moiz et al. *Combust. Flame* 173 (2016): 123–131. doi:10.1016/j.combustflame.2016.08.005

What drives costs?

What drives costs?

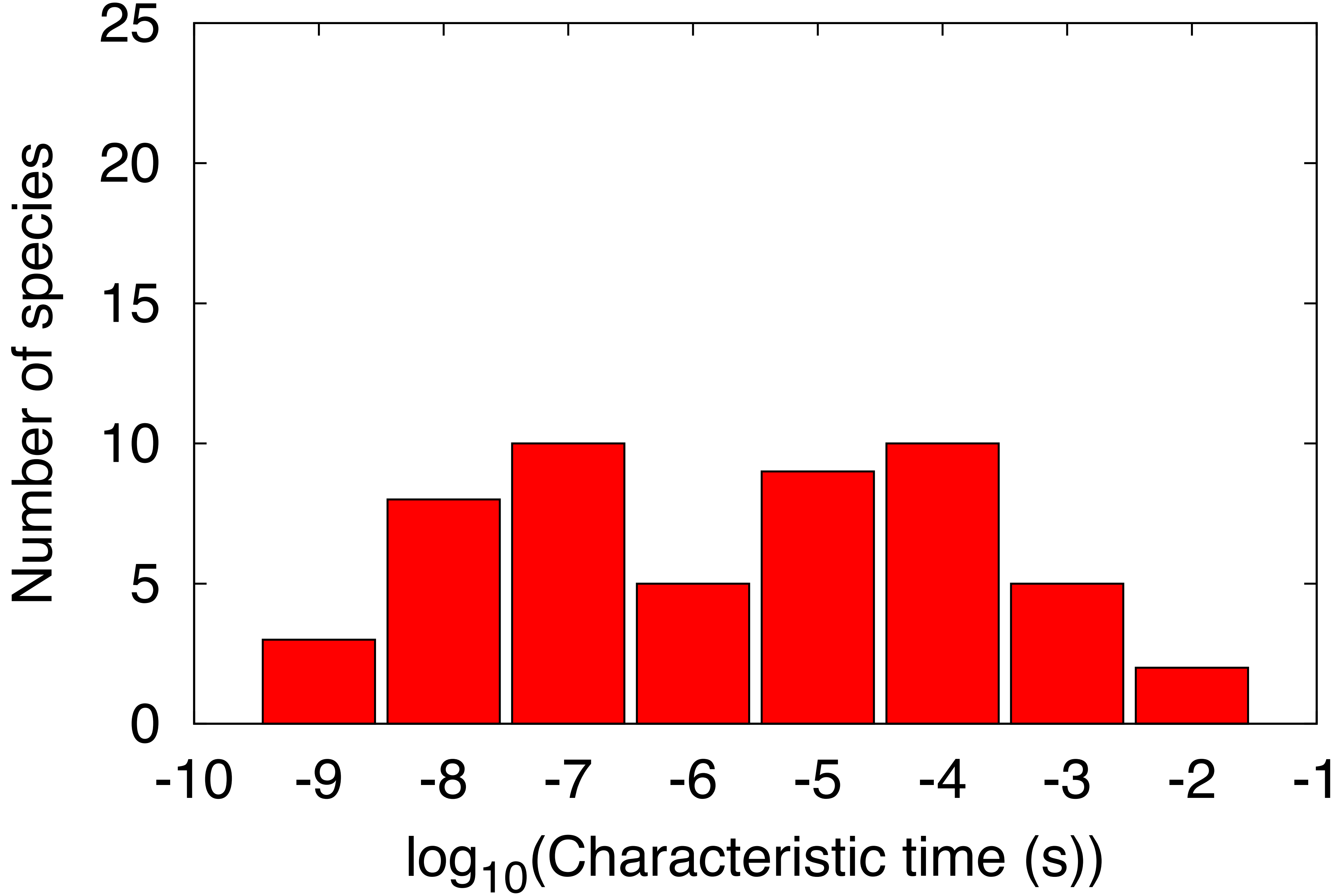
Stiffness

What drives costs?

Stiffness

Size

Kinetic models exhibit high stiffness:



Characteristic creation times of methane oxidation[2]

[2]K. E. Niemeyer, N. J. Curtis, & C. J. Sung. Fall 2015 Meeting of the West. States Sect. Combust. Inst. Provo, UT, USA, Oct. 2015. doi:10.6084/m9.figshare.2075515.v1

Stiffness

Stiffness

- Wide range of species/reaction time scales

Stiffness

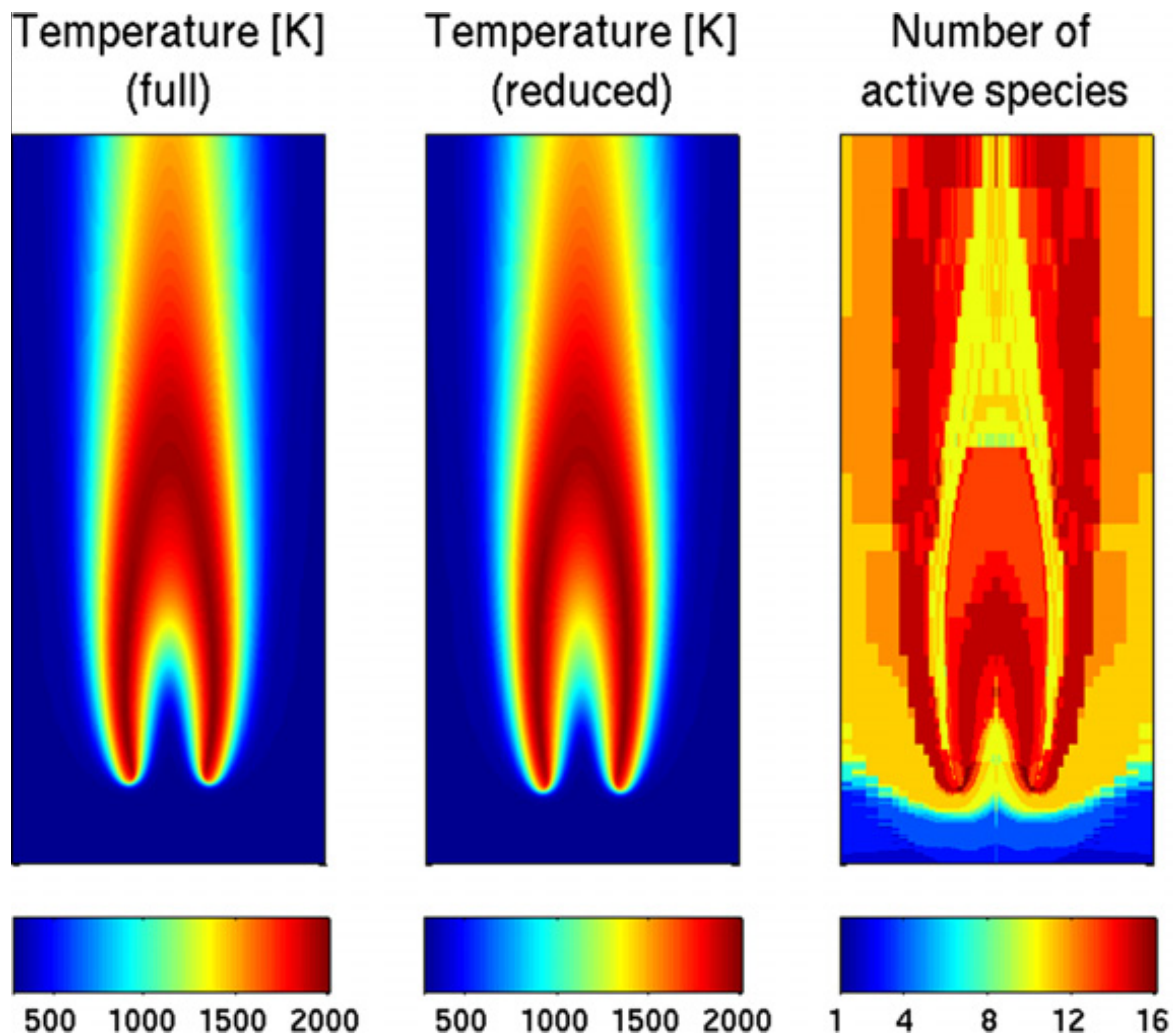
- Wide range of species/reaction time scales
- Rapidly depleting radical species, fast reversible reactions

Stiffness

- Wide range of species/reaction time scales
- Rapidly depleting radical species, fast reversible reactions
- Traditionally requires implicit integration algorithms

Are implicit integrators required *everywhere*?

Are implicit integrators required *everywhere*?



Dynamic adaptive chemistry approach of Tosatto et. al, studying a 2-D diluted JP-8 flame[3]

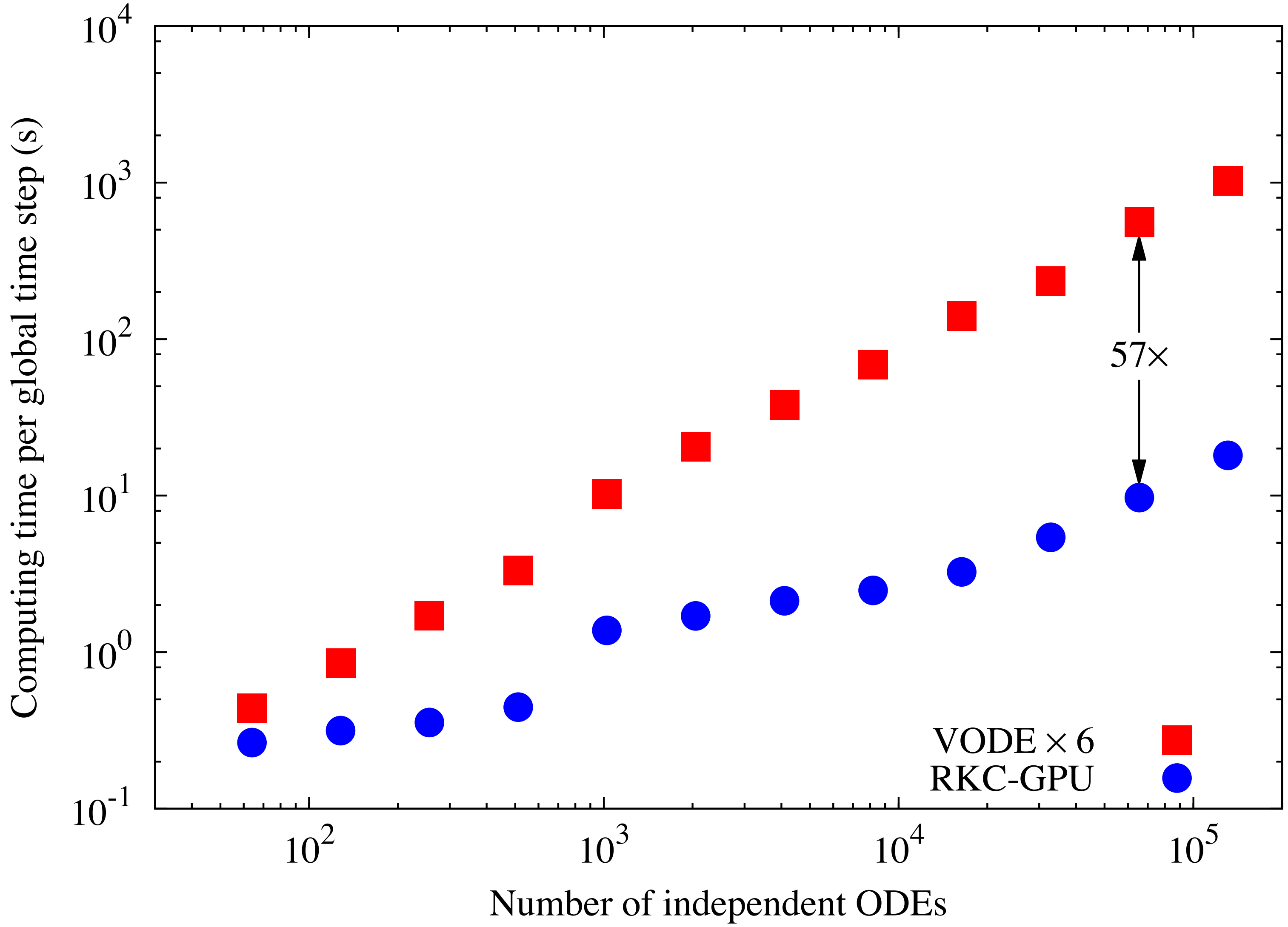
[3] L. Tosatto, B. Bennett, & M. Smooke. *Combust. Flame* 158.5 (2011):820–835. doi:10.1016/j.combustflame.2011.01.018

Are implicit integrators required *everywhere*?

Many areas of a reactive-flow simulation are non/weakly-reacting, or at chemical equilibrium:

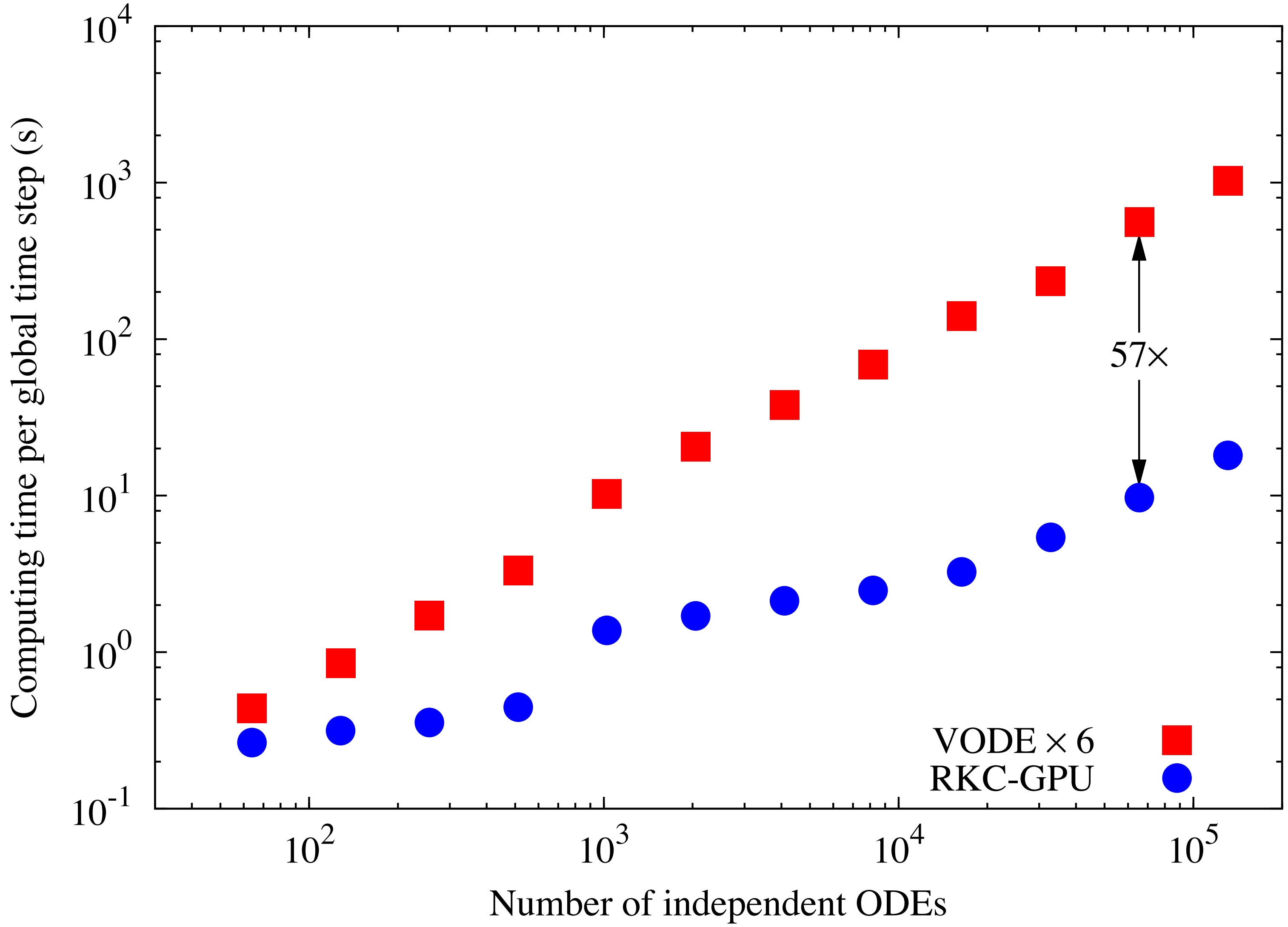
Are implicit integrators required *everywhere*?

Many areas of a reactive-flow simulation are non/weakly-reacting, or at chemical equilibrium:



Are implicit integrators required *everywhere*?

Many areas of a reactive-flow simulation are non/weakly-reacting, or at chemical equilibrium:



For less-stiff chemistry, stabilized-explicit or semi-implicit solvers may be **much** faster[4]

[4]K. E. Niemeyer & C. J. Sung. *J. Comput. Phys.* 256 (2014), pp. 854–871. doi:10.1016/j.jcp.2013.09.025

What drives costs?

What drives costs?

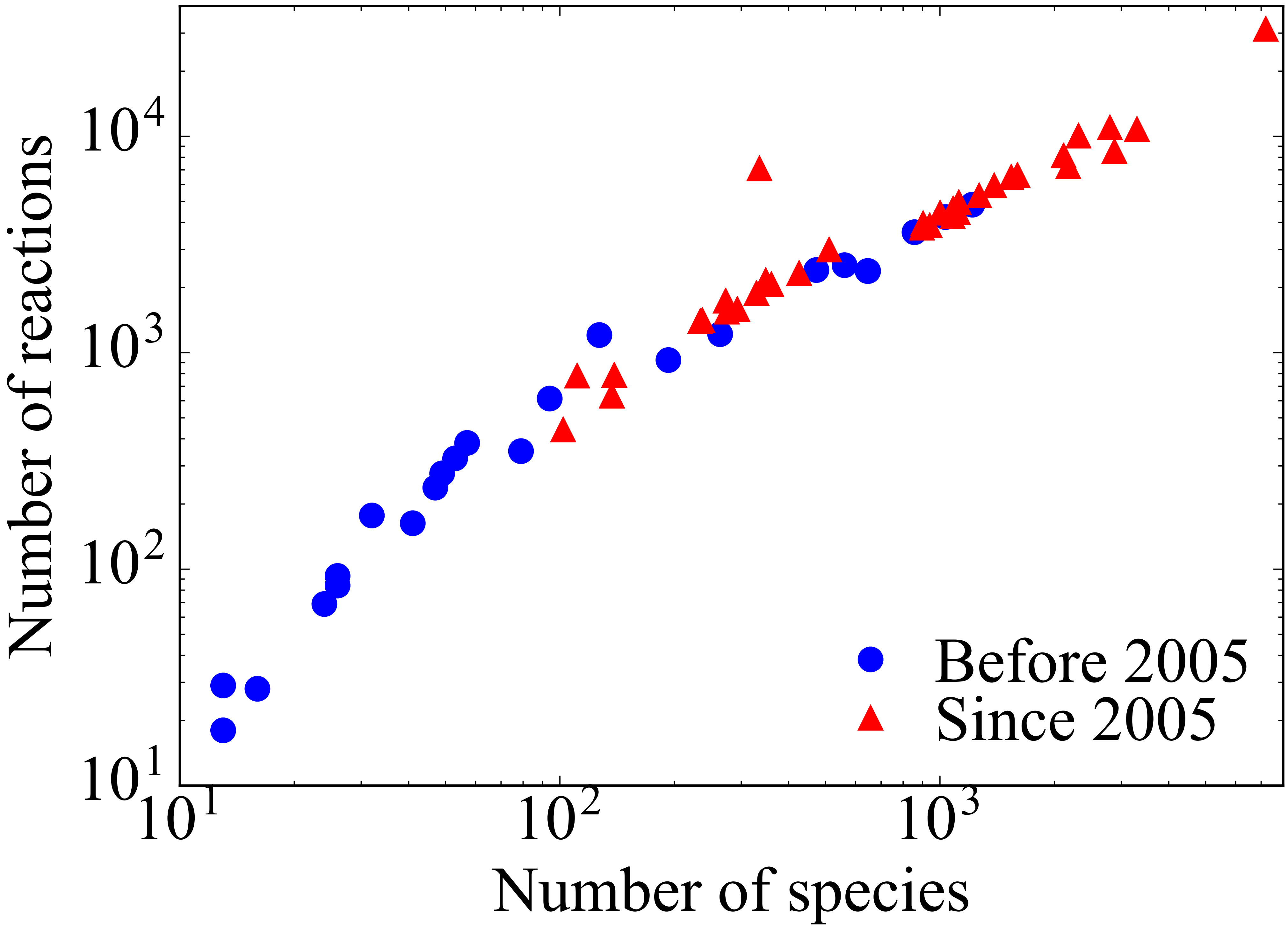
Stiffness

What drives costs?

Stiffness

Size

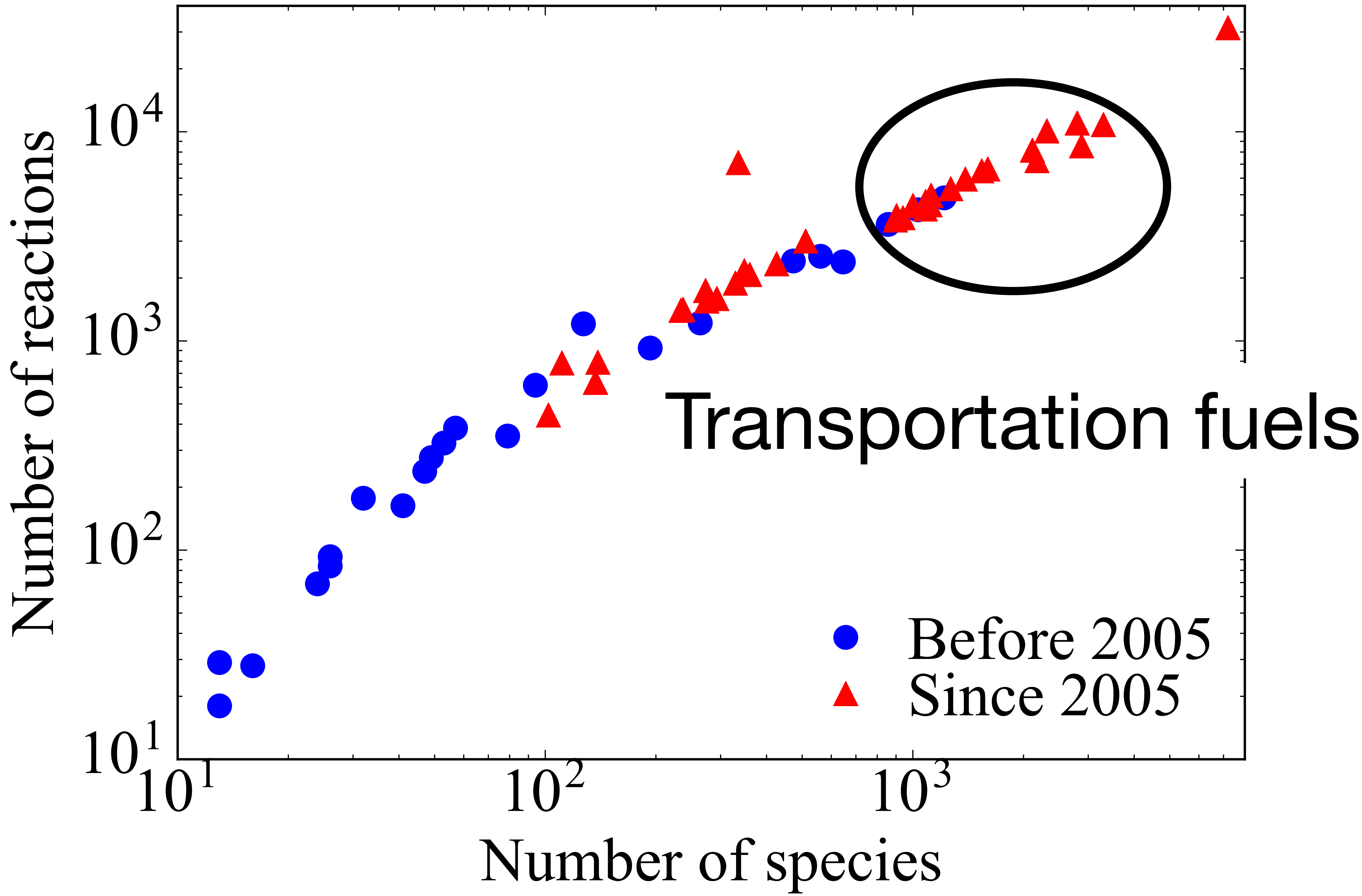
Kinetic model sizes have grown in recent years:



Chemical kinetic model size for hydrocarbon oxidation[5]

[5]K. Niemeyer. Hydrocarbon chemical kinetic model survey. figshare. 2016. doi:10.6084/m9.figshare.3792660.v1

Kinetic model sizes have grown in recent years:



Chemical kinetic model size for hydrocarbon oxidation[5]

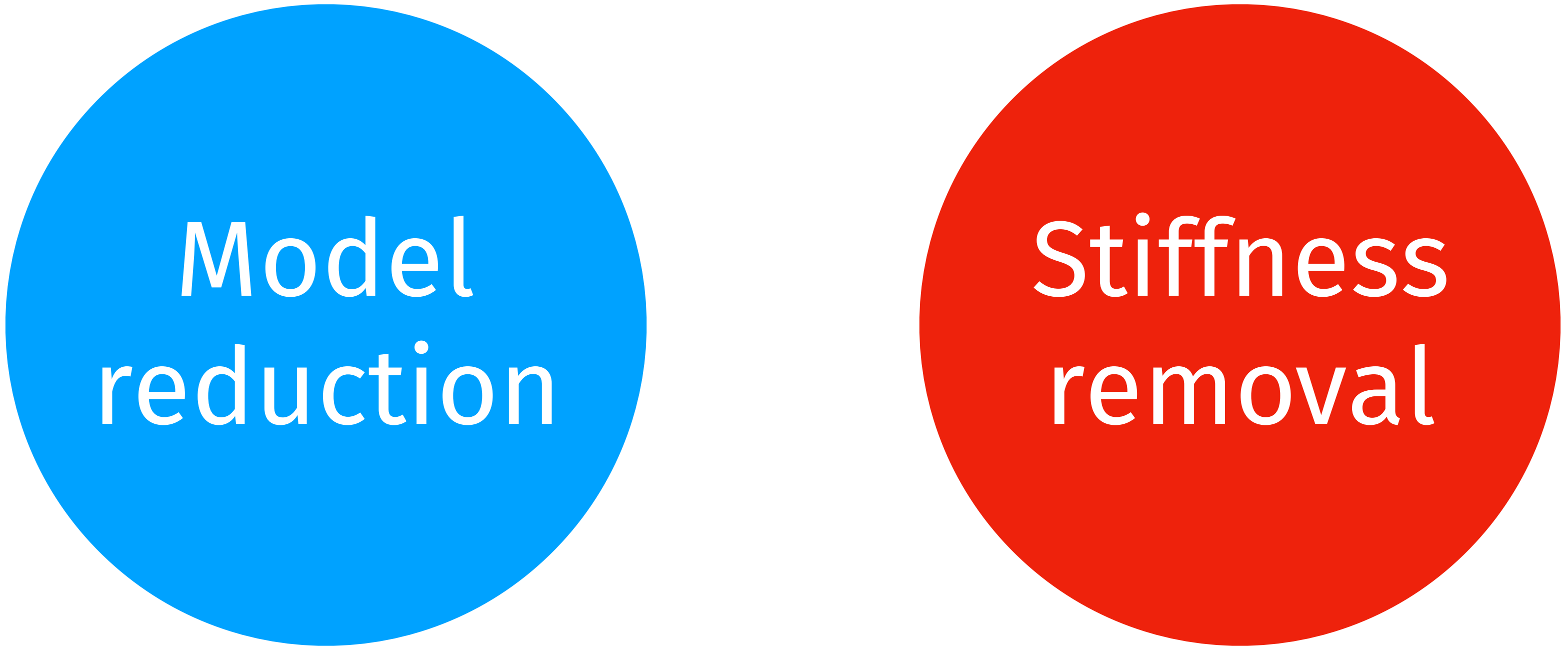
[5]K. Niemeyer. Hydrocarbon chemical kinetic model survey. figshare. 2016.
doi:10.6084/m9.figshare.3792660.v1

Cost reduction

Cost reduction

Model
reduction

Cost reduction



Cost reduction

Model reduction

Stiffness removal

Tabulation

Cost reduction

Model reduction

Stiffness removal

Tabulation

Integration algorithms

Cost reduction

Integration algorithms

Cost of integration

Cost of integration

Implicit algorithms require:

Cost of integration

Implicit algorithms require:

- Jacobian evaluation with finite differences: cost scales **quadratically** with number of species

Cost of integration

Implicit algorithms require:

- Jacobian evaluation with finite differences: cost scales **quadratically** with number of species
- (Dense) Jacobian factorization: cost scales **cubically** with number of species

Cost of integration

Implicit algorithms require:

- Jacobian evaluation with finite differences: cost scales **quadratically** with number of species

- (Dense) Jacobian factorization: cost scales **cubically** with number of species

Speedup may be achieved with a **sparse, analytical** Jacobian formulation

Parallelism

Parallelism

Distributed and multi-core parallelism are not enough…

Parallelism

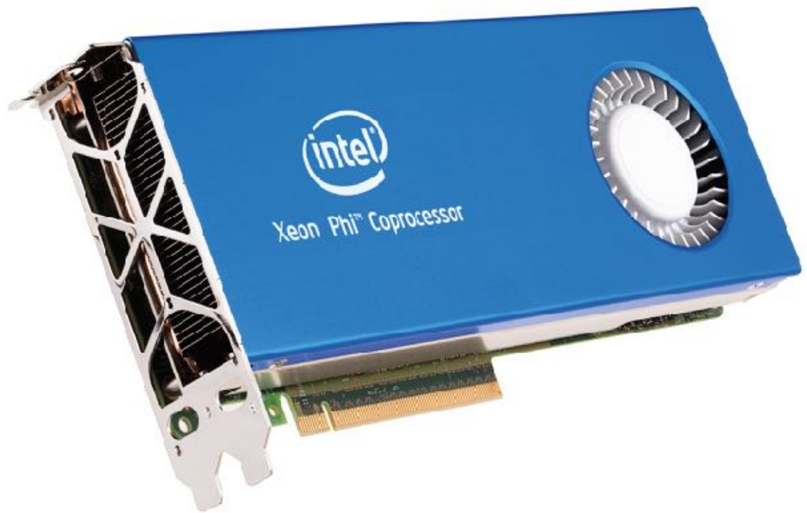
Distributed and multi-core parallelism are not enough…

SIMD-enabled and related single-instruction, multiple-thread (SIMT) processors have gained importance in scientific computing due to their increased FLOP throughput

Parallelism

Distributed and multi-core parallelism are not enough…

SIMD-enabled and related single-instruction, multiple-thread (SIMT) processors have gained importance in scientific computing due to their increased FLOP throughput



Project thrusts

Project thrusts

- Create sparse, analytical chemical kinetic Jacobian code to speed up existing programs, and power new ones

Project thrusts

- Create sparse, analytical chemical kinetic Jacobian code to speed up existing programs, and power new ones
- Develop library of vectorized solvers, usable on heterogeneous architectures (CPU, GPU, MIC, …)

Project thrusts

- Create sparse, analytical chemical kinetic Jacobian code to speed up existing programs, and power new ones
- Develop library of vectorized solvers, usable on heterogeneous architectures (CPU, GPU, MIC, …)
- Design scheduler for chemical kinetics ODEs based on stiffness metric to select appropriate integrators on available hardware

Project thrusts

- Create sparse, analytical chemical kinetic Jacobian code to speed up existing programs, and power new ones
- Develop library of vectorized solvers, usable on heterogeneous architectures (CPU, GPU, MIC, …)
- Design scheduler for chemical kinetics ODEs based on stiffness metric to select appropriate integrators on available hardware

http://slackha.github.io/

https://github.com/SLACKHA

`pyJac`: analytical chemical kinetic Jacobian generator

pyJac: analytical chemical kinetic Jacobian generator

pyJac[6]: open-source Python package that generates source code used to analytically calculate constant-pressure, mass-fraction based chemical kinetic Jacobian matrices. Currently supports:

https://github.com/SLACKHA/pyJac

[6]K. E. Niemeyer, N. J. Curtis, & C. J. Sung. *Comput. Phys. Comm.* 215 (2017):188–203. doi:10.1016/j.cpc.2017.02.004

pyJac: analytical chemical kinetic Jacobian generator

pyJac[6]: open-source Python package that generates source code used to analytically calculate constant-pressure, mass-fraction based chemical kinetic Jacobian matrices. Currently supports:

- Multi-threaded C or CUDA execution

https://github.com/SLACKHA/pyJac

[6]K. E. Niemeyer, N. J. Curtis, & C. J. Sung. *Comput. Phys. Comm.* 215 (2017):188–203. doi:10.1016/j.cpc.2017.02.004

`pyJac`: analytical chemical kinetic Jacobian generator

`pyJac`[6]: open-source Python package that generates source code used to analytically calculate constant-pressure, mass-fraction based chemical kinetic Jacobian matrices. Currently supports:

- Multi-threaded C or CUDA execution
- Built-in library generation for linking to external codes

`https://github.com/SLACKHA/pyJac`

[6]K. E. Niemeyer, N. J. Curtis, & C. J. Sung. *Comput. Phys. Comm.* 215 (2017):188–203. doi:10.1016/j.cpc.2017.02.004

pyJac: analytical chemical kinetic Jacobian generator

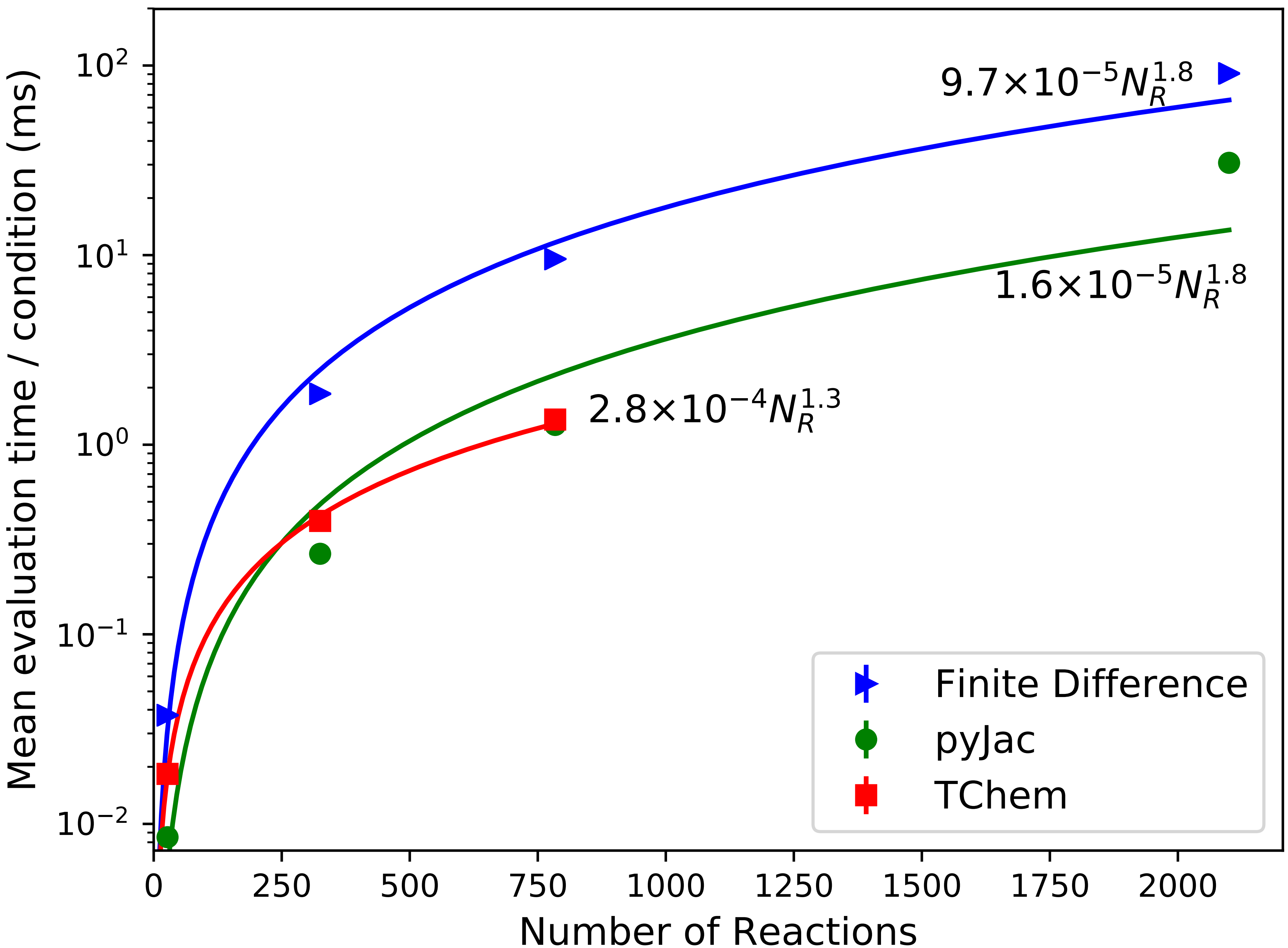
pyJac[6]: open-source Python package that generates source code used to analytically calculate constant-pressure, mass-fraction based chemical kinetic Jacobian matrices. Currently supports:

- Multi-threaded C or CUDA execution
- Built-in library generation for linking to external codes
- Python wrapper creation for (relatively) easy access

https://github.com/SLACKHA/pyJac

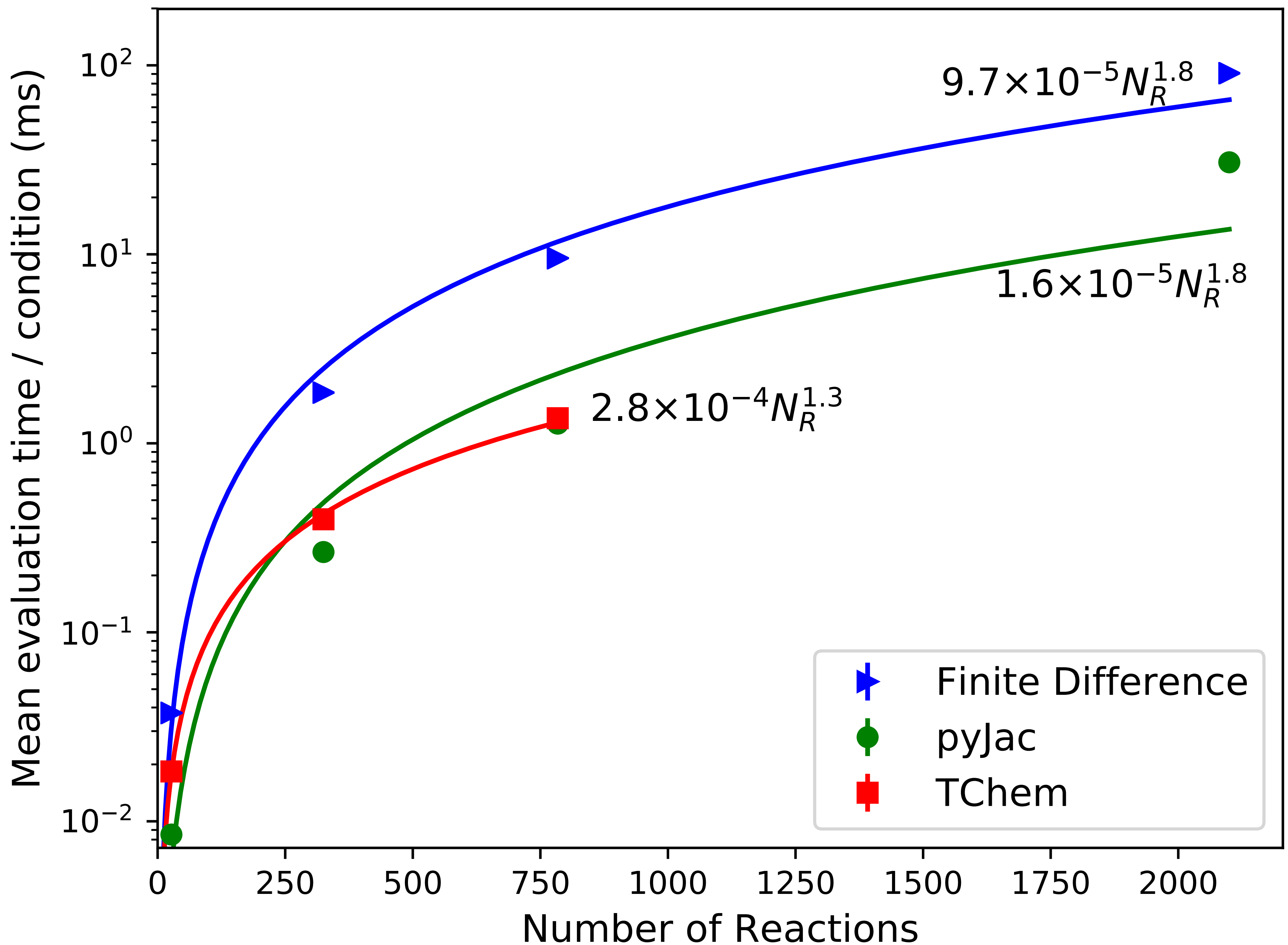
[6]K. E. Niemeyer, N. J. Curtis, & C. J. Sung. *Comput. Phys. Comm.* 215 (2017):188–203. doi:10.1016/j.cpc.2017.02.004

pyJac: analytical chemical kinetic Jacobian generator



Performance comparison with finite differences and TChem[6]

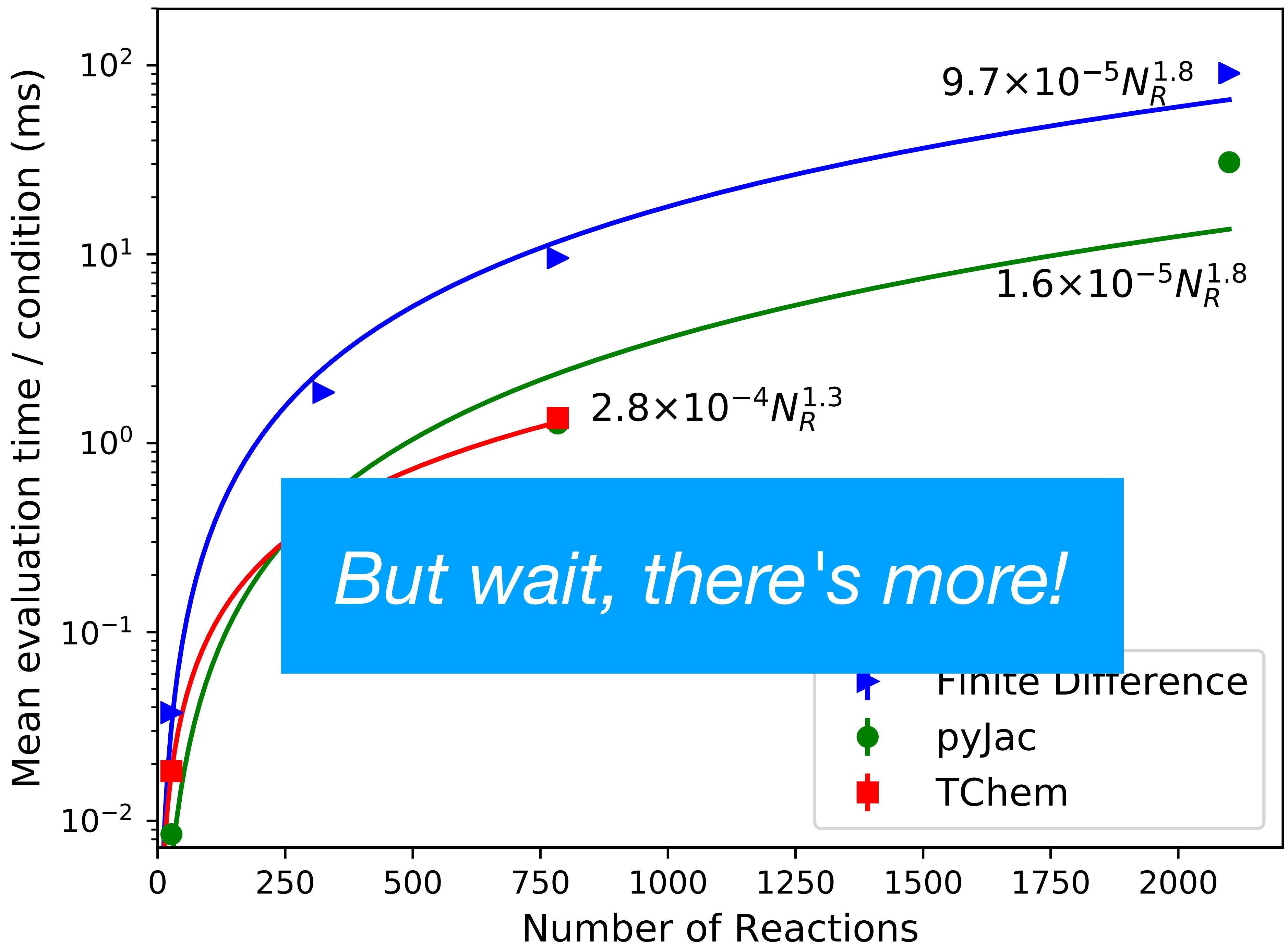
pyJac: analytical chemical kinetic Jacobian generator



Performance comparison with finite differences and TChem[6]

[6]K. E. Niemeyer, N. J. Curtis, & C. J. Sung. *Comput. Phys. Comm.* 215 (2017):188–203. doi:10.1016/j.cpc.2017.02.004

`pyJac`: analytical chemical kinetic Jacobian generator



Performance comparison with finite differences and TChem[6]

[6]K. E. Niemeyer, N. J. Curtis, & C. J. Sung. *Comput. Phys. Comm.* 215 (2017):188–203. doi:10.1016/j.cpc.2017.02.004

`pyJac`: analytical chemical kinetic Jacobian generator

pyJac: analytical chemical kinetic Jacobian generator

pyJac v2 currently under development, targeting SIMD/SIMT vectorization on CPUs, GPUs and MICs

https://github.com/SLACKHA/pyJac

pyJac: analytical chemical kinetic Jacobian generator

pyJac v2 currently under development, targeting SIMD/SIMT vectorization on CPUs, GPUs and MICs

- Change of system to concentration-based equations to increase sparsity

https://github.com/SLACKHA/pyJac

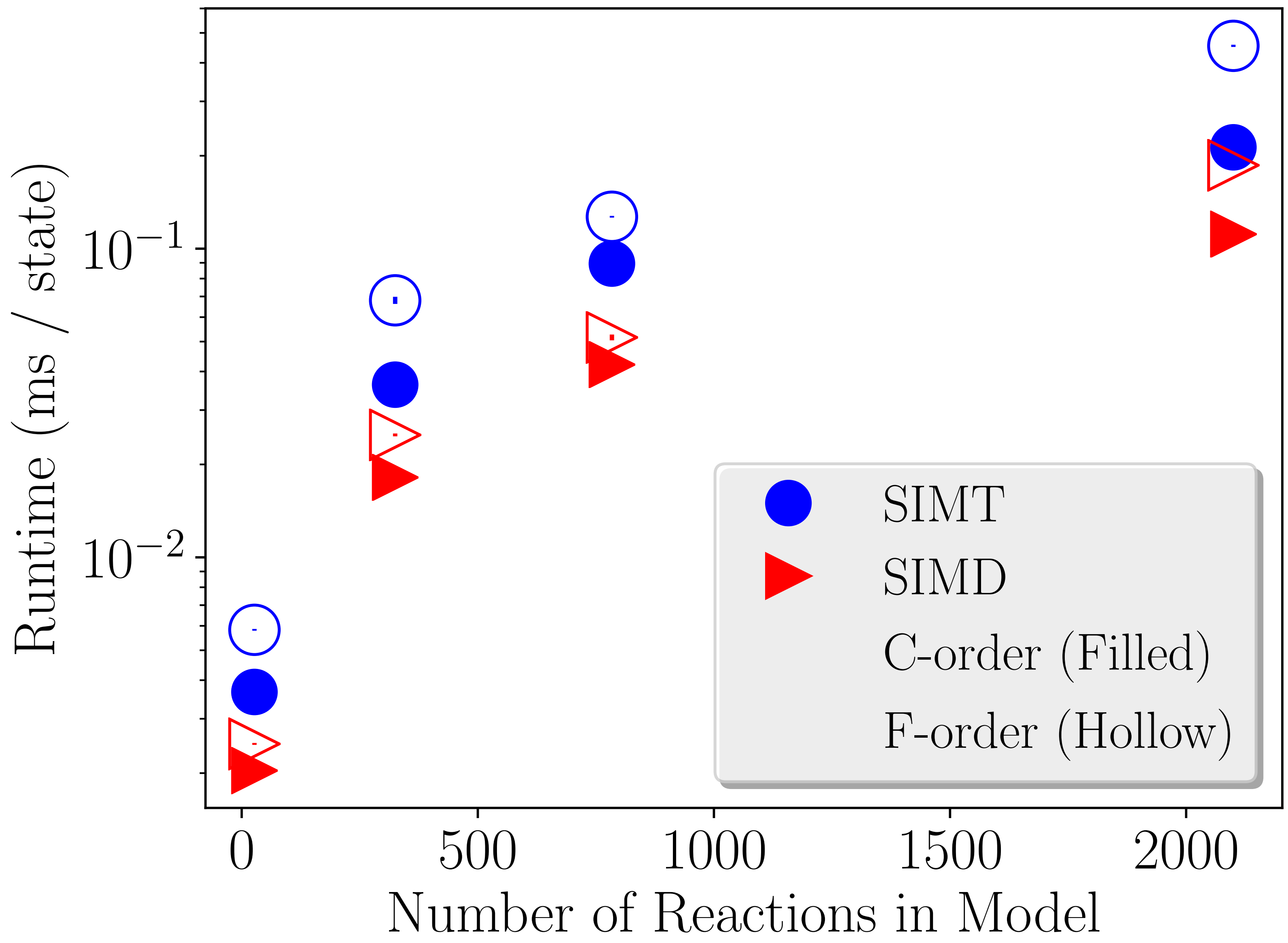
`pyJac`: analytical chemical kinetic Jacobian generator

`pyJac` v2 currently under development, targeting SIMD/SIMT vectorization on CPUs, GPUs and MICs

- Change of system to concentration-based equations to increase sparsity
- Both wide ("per-thread") and deep ("per-block") vectorization pursued to provide more flexible options for ODE integration

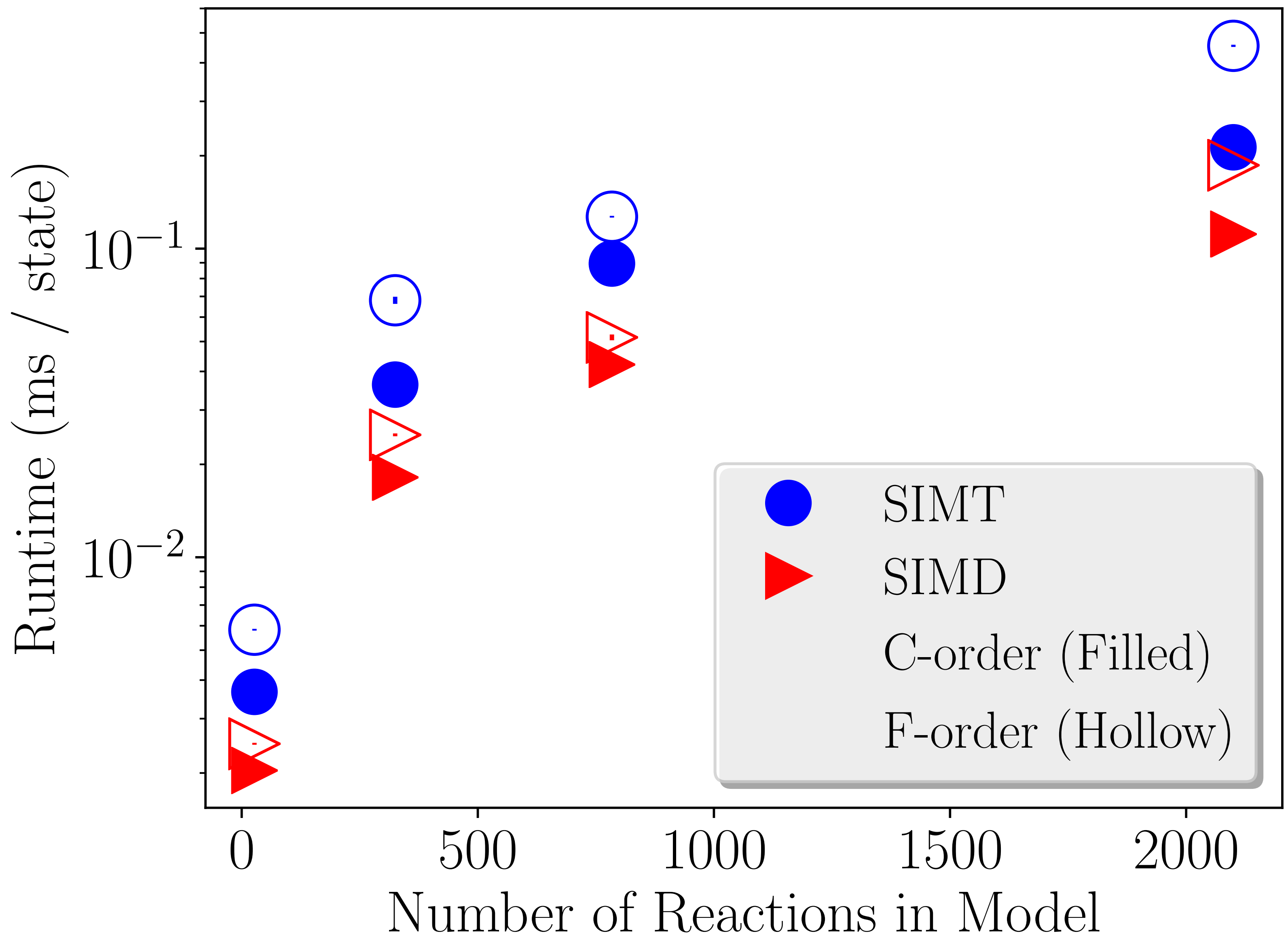
`https://github.com/SLACKHA/pyJac`

pyJac v2 vectorized rate evaluation



Runtimes of wide SIMD-vectorized species/temperature rates compared to a non-vectorized (SIMT) baseline on a single core of Intel Xeon X5650 CPU[7]

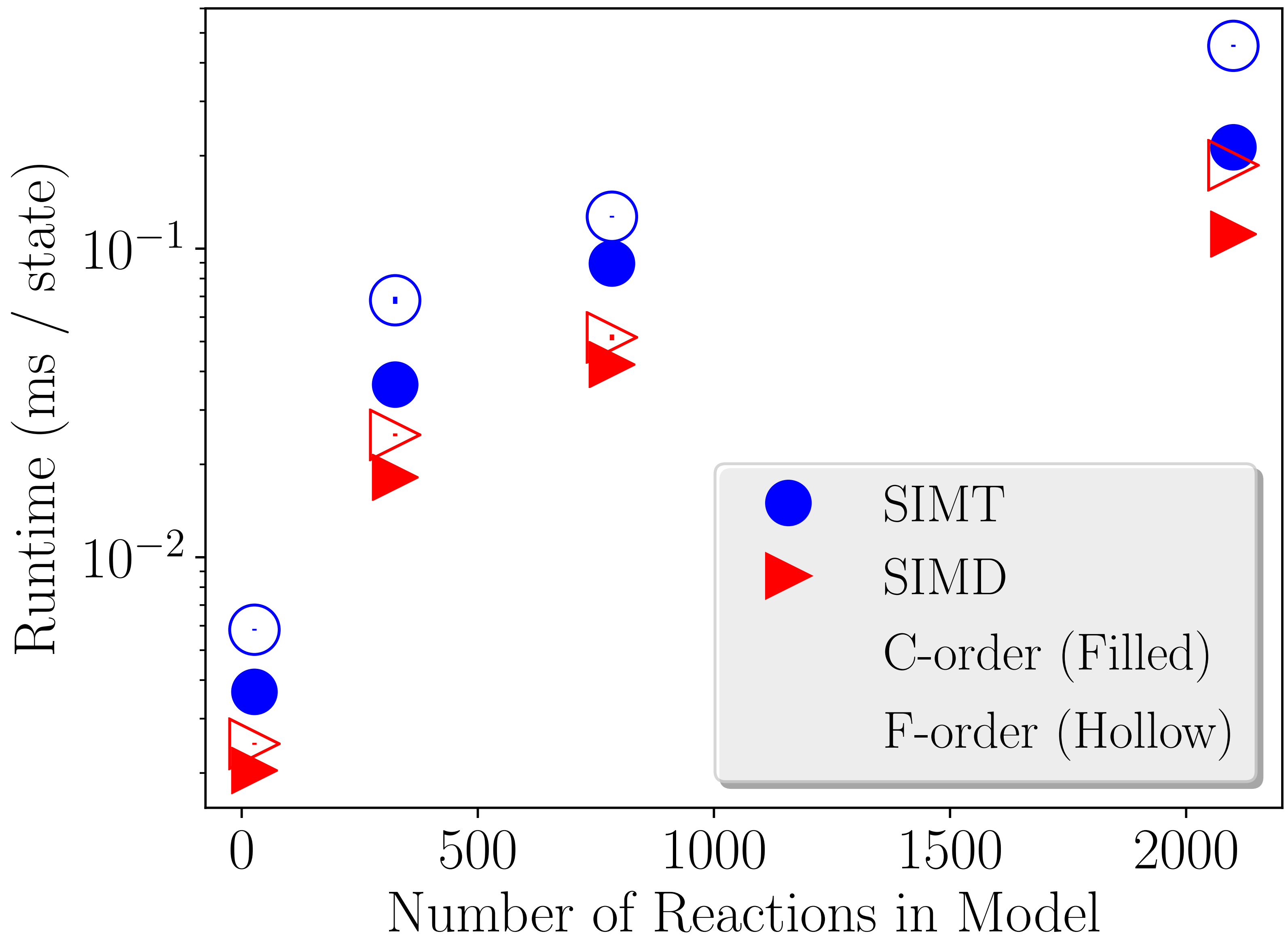
pyJac v2 vectorized rate evaluation



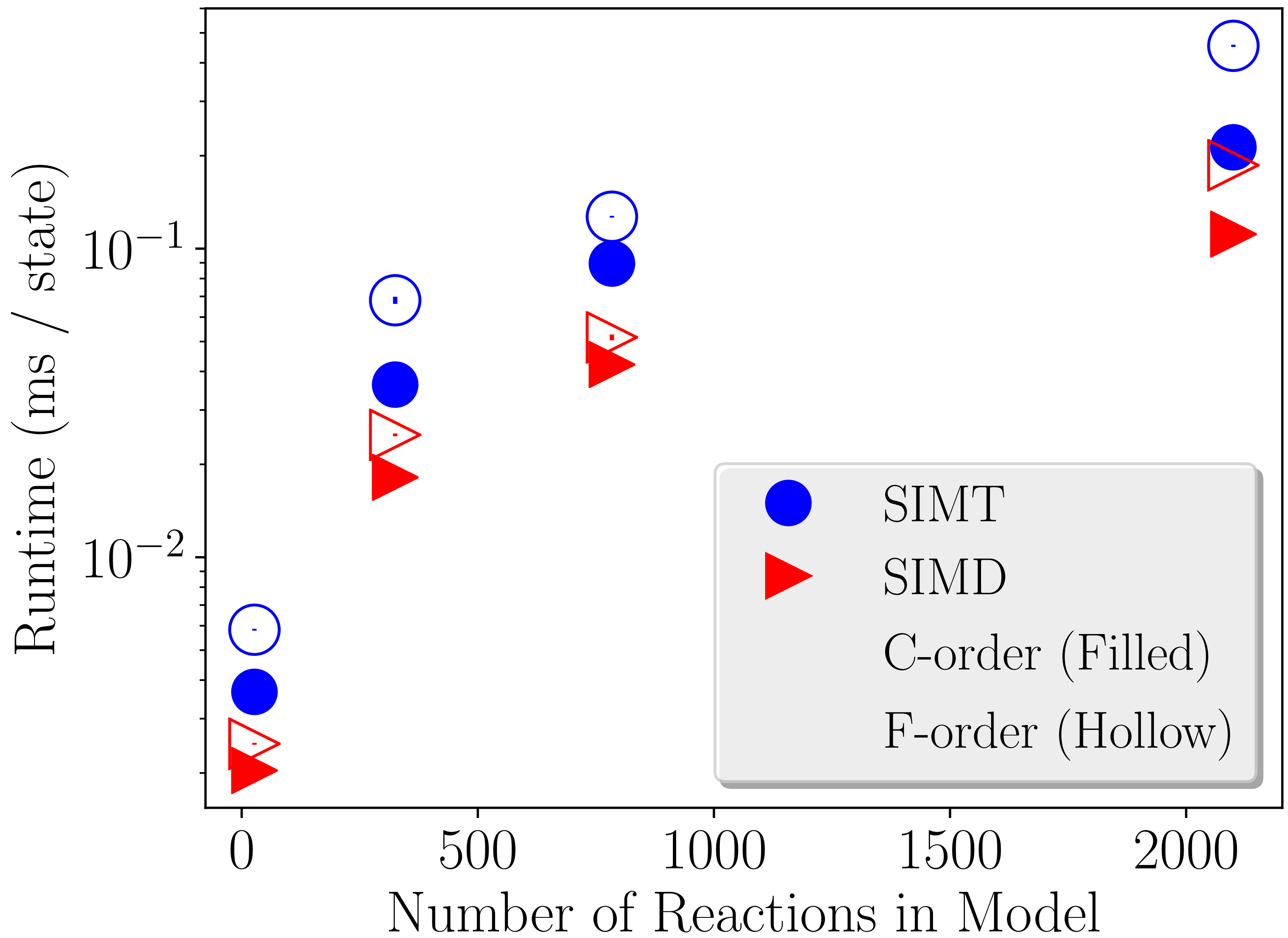
Runtimes of wide SIMD-vectorized species/temperature rates compared to a non-vectorized (SIMT) baseline on a single core of Intel Xeon X5650 CPU[7]

[7]N. J. Curtis & C. J. Sung. "SIMD-vectorized Chemical Source Term Evaluation", 10th U.S. National Combustion Meeting, College Park, MD.

pyJac v2 vectorized rate evaluation

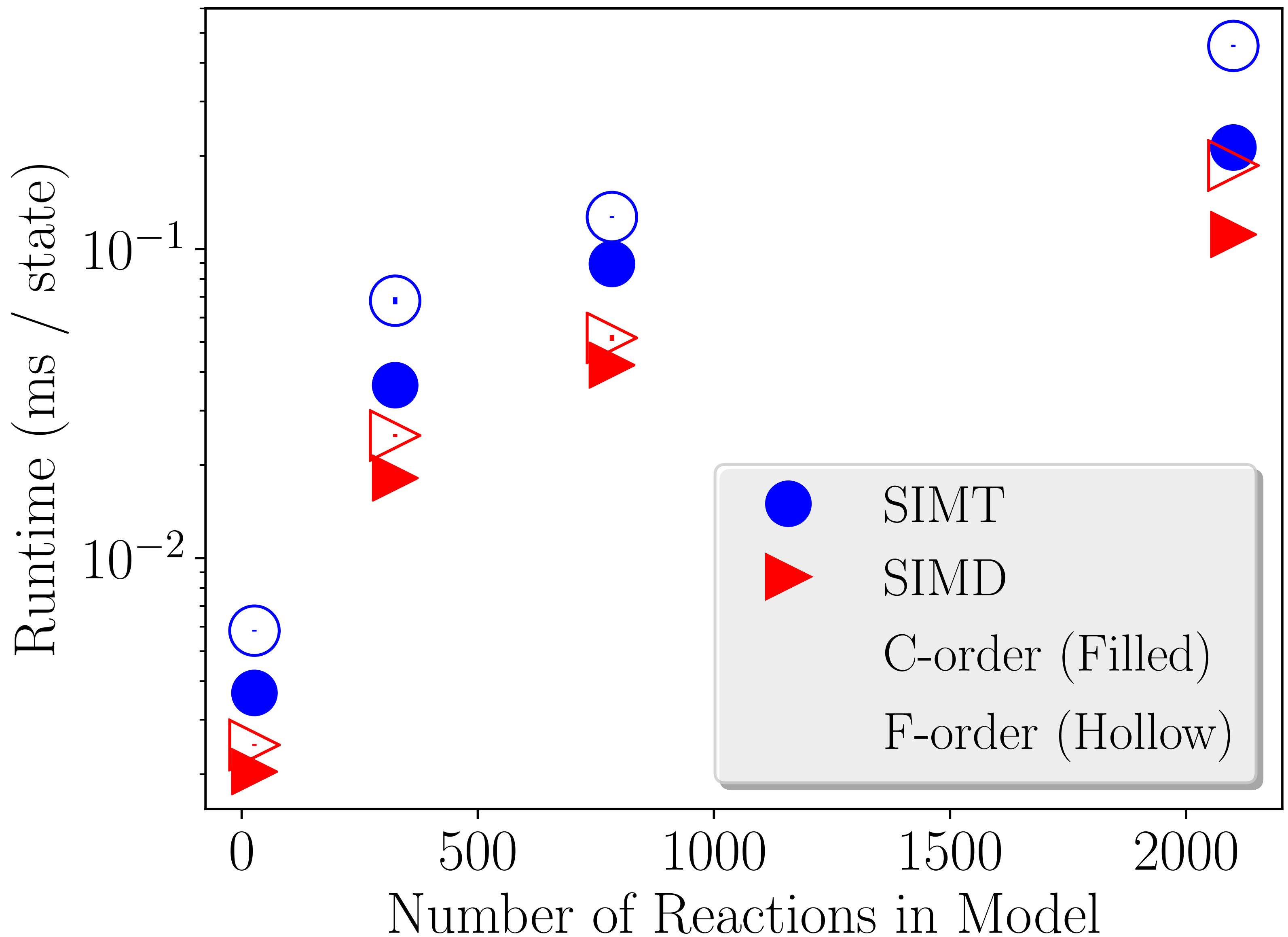


`pyJac` v2 vectorized rate evaluation



- “C”-ordered (row-major) data up to 1.67–2.13× faster than “F”-ordered (column-major)

pyJac v2 vectorized rate evaluation



- “C”-ordered (row-major) data up to 1.67–2.13× faster than “F”-ordered (column-major)

- SIMD-vectorized code up to 1.99–2.72 × faster than non-vectorized baseline

`accelerInt`: integrators for hybrid architectures

`accelerInt`: integrators for hybrid architectures

`accelerInt`: collection of validated[4,8] stiff and non-stiff integrators for the CPU and GPU

`https://github.com/SLACKHA/accelerInt`

`accelerInt`: integrators for hybrid architectures

`accelerInt`: collection of validated[4,8] stiff and non-stiff integrators for the CPU and GPU

- Multithreaded CPU/wide-vectorized GPU solvers

`https://github.com/SLACKHA/accelerInt`

`accelerInt`: integrators for hybrid architectures

`accelerInt`: collection of validated[4,8] stiff and non-stiff integrators for the CPU and GPU

- Multithreaded CPU/wide-vectorized GPU solvers
- Built-in compatibility with `pyJac` (but other ODE systems are possible too)

`https://github.com/SLACKHA/accelerInt`

`accelerInt`: integrators for hybrid architectures

`accelerInt`: collection of validated[4,8] stiff and non-stiff integrators for the CPU and GPU

- Multithreaded CPU/wide-vectorized GPU solvers
- Built-in compatibility with `pyJac` (but other ODE systems are possible too)
- Library interface available for use with external code

`https://github.com/SLACKHA/accelerInt`

[8]N. J. Curtis, K. E. Niemeyer, & C. J. Sung. *Combust. Flame* 179 (2017):312–324. doi:10.1016/j.combustflame.2017.02.005

`accelerInt`: available solvers

Integrator	Type	Order	CPU	GPU
CVODE[9]	Variable-order BDF	Variable (max 5th)	×	–
Radau-IIa[10]	Implicit RK	5th	×	×
EXP4[11]	Semi-implicit exponential	Nominally 4th	×	×
EXPRB43[12]	Semi-implicit exponential	4th	×	×

`accelerInt`: available solvers

Integrator	Type	Order	CPU	GPU
CVODE[9]	Variable-order BDF	Variable (max 5th)	×	–
Radau-IIa[10]	Implicit RK	5th	×	×
EXP4[11]	Semi-implicit exponential	Nominally 4th	×	×
EXPRB43[12]	Semi-implicit exponential	4th	×	×

[9] P. N. Brown, G. D. Byrne, & A. C. Hindmarsh. *SIAM J. Sci. Stat. Comput.* 10.5 (1989):1038–1051. doi:10.1137/0910062

[10] G. Wanner & E. Hairer. *Solving Ordinary Differential Equations II*. 2nd ed. Springer-Verlag, Berlin, 1996. doi:10.1007/978-3-642-05221-7

[11] M. Hochbruck, C. Lubich, & H. Selhofer. *SIAM J. Sci. Comput*. 19.5 (1998):1552–1574. doi:10.1137/S1064827595295337

[12] M. Hochbruck, A. Ostermann, & J. Schweitzer. *SIAM J. Numer. Anal*. 47.1 (2009):786–803. doi:10.1137/080717717

`accelerInt`: planned additions

`accelerInt`: planned additions

- Addition of 5th-order explicit Runge–Kutta–Cash–Karp and stabilized explicit second-order Runge–Kutta–Chebyshev solvers[4]

`accelerInt`: planned additions

- Addition of 5th-order explicit Runge–Kutta–Cash–Karp and stabilized explicit second-order Runge–Kutta–Chebyshev solvers[4]
- Update for new vectorized version of `pyJac`

`accelerInt`: planned additions

- Addition of 5th-order explicit Runge–Kutta–Cash–Karp and stabilized explicit second-order Runge–Kutta–Chebyshev solvers[4]
- Update for new vectorized version of `pyJac`
- Addition of linearly-implicit methods (Rosenbrock) and (potentially) hybrid implicit/explicit solvers

Stiffness characterization

Stiffness characterization

- **Goal**: reliable stiffness metric to switch between integration algorithms based on state & hardware

Stiffness characterization

- **Goal**: reliable stiffness metric to switch between integration algorithms based on state & hardware
- Currently: evaluate existing stiffness metrics using realistic, sampled PaSR state data

Stiffness ratio

$$\text{ratio} = \frac{\max|\lambda_p|}{\min|\lambda_p|}$$

Stiffness ratio

$$\text{ratio} = \frac{\max|\lambda_p|}{\min|\lambda_p|}$$

LeVeque, R. J. *Finite Difference Methods for Ordinary and Partial Differential Equations*. (2007)
doi:10.1137/1.9780898717839

Stiffness ratio

$$\text{ratio} = \frac{\max|\lambda_p|}{\min|\lambda_p|}$$

$$\lambda_p = \text{eigenvalue of Jacobian}$$

LeVeque, R. J. *Finite Difference Methods for Ordinary and Partial Differential Equations*. (2007)
doi:10.1137/1.9780898717839

Shampine stiffness index

$$\text{index} = \rho[f_y(x_n, y(x_n))]\|y^{(p+1)}(x_n)\|^{-1/(p+1)}$$

Shampine stiffness index

$$\text{index} = \rho[f_y(x_n, y(x_n))] \|y^{(p+1)}(x_n)\|^{-1/(p+1)}$$

L. F. Shampine, *Mathematics of Computation* 39 (1982):109–123.
doi:10.1090/S0025-5718-1982-0658216-2

Shampine stiffness index

$$\text{index} = \underline{\rho[f_y(x_n, y(x_n))]}\|y^{(p+1)}(x_n)\|^{-1/(p+1)}$$

spectral radius
of Jacobian

L. F. Shampine, *Mathematics of Computation* 39 (1982):109–123.
doi:10.1090/S0025-5718-1982-0658216-2

Shampine stiffness index

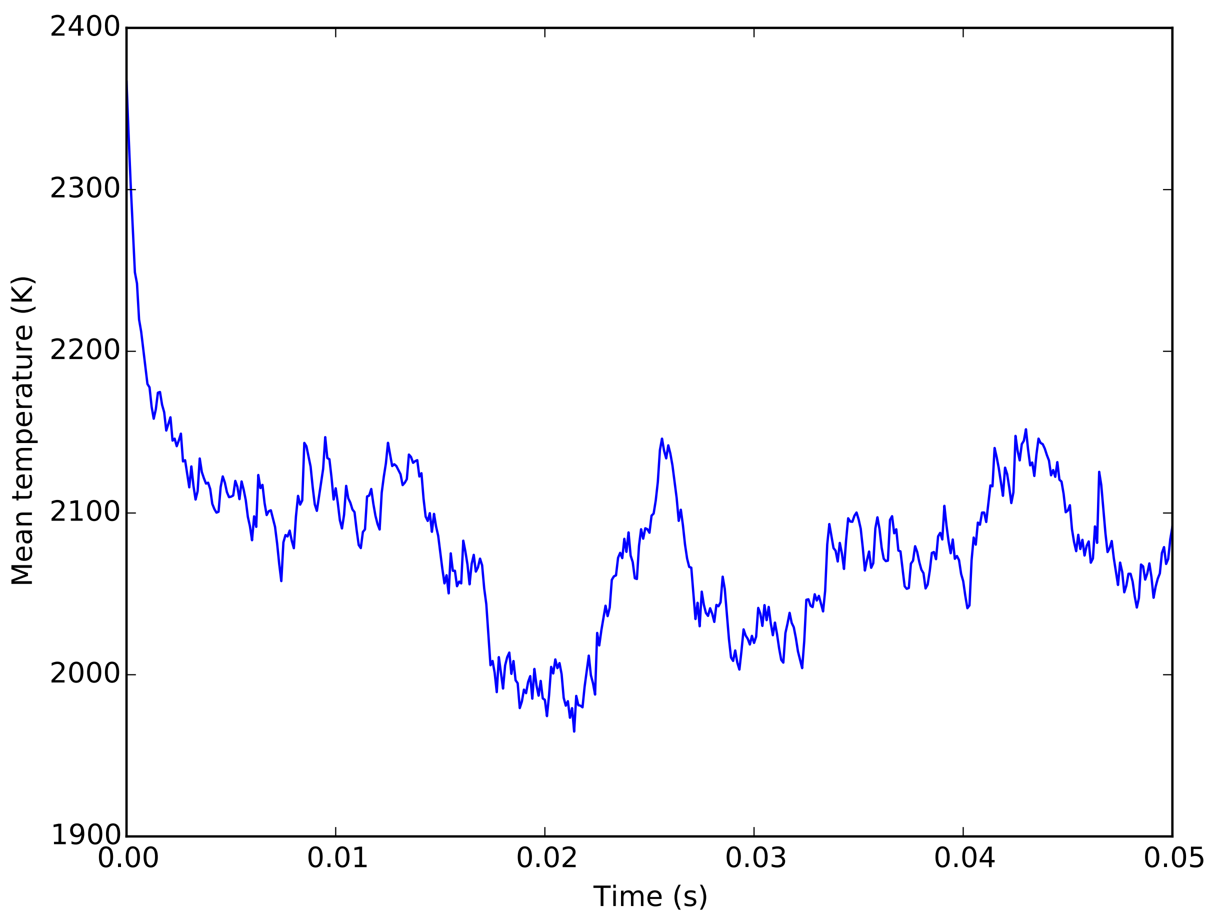
$$\text{index} = \rho[f_y(x_n, y(x_n))] \| y^{(p+1)}(x_n) \|^{-1/(p+1)}$$

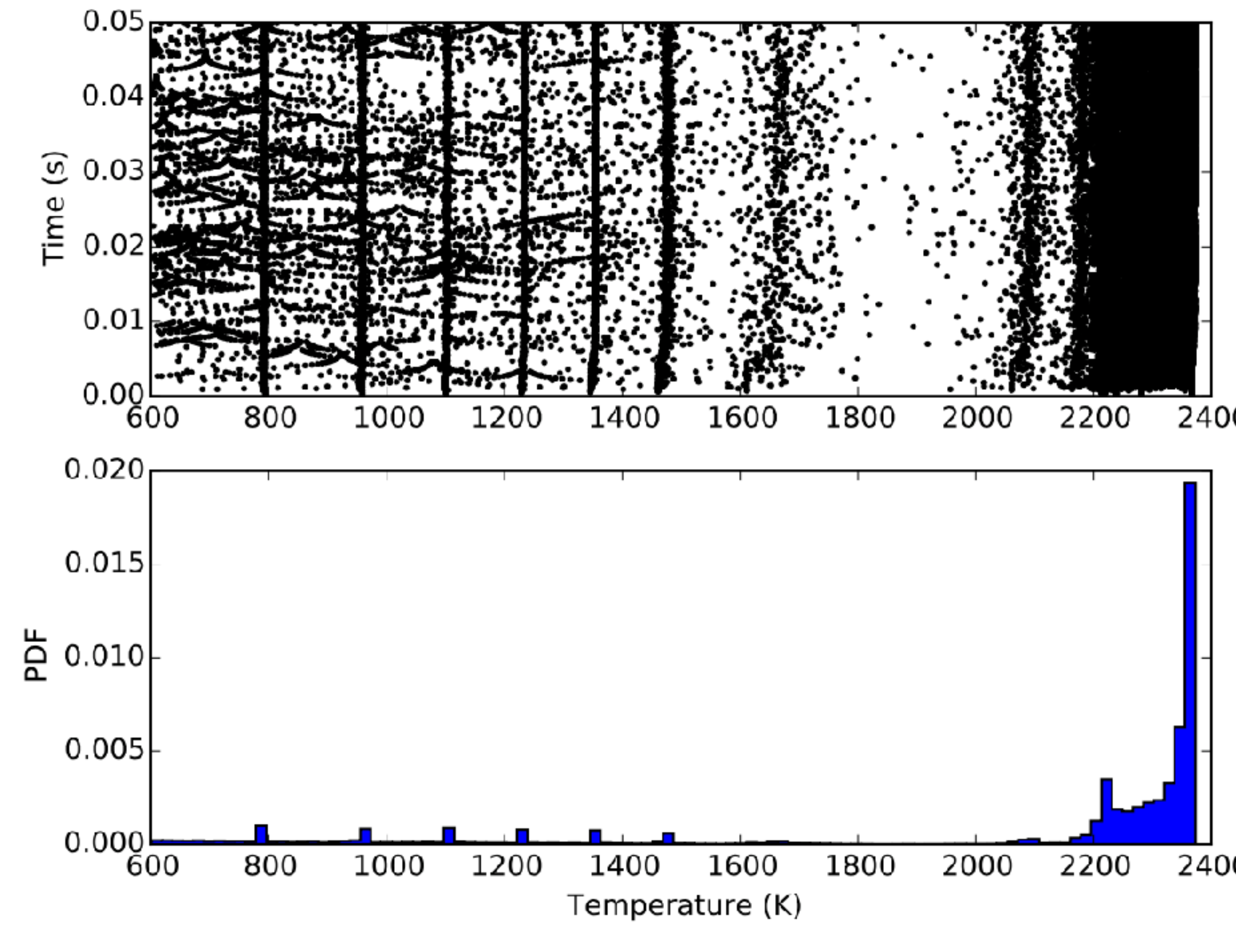
spectral radius
of Jacobian

p+1 derivative
of solution vector

L. F. Shampine, *Mathematics of Computation* 39 (1982):109–123.
doi:10.1090/S0025-5718-1982-0658216-2

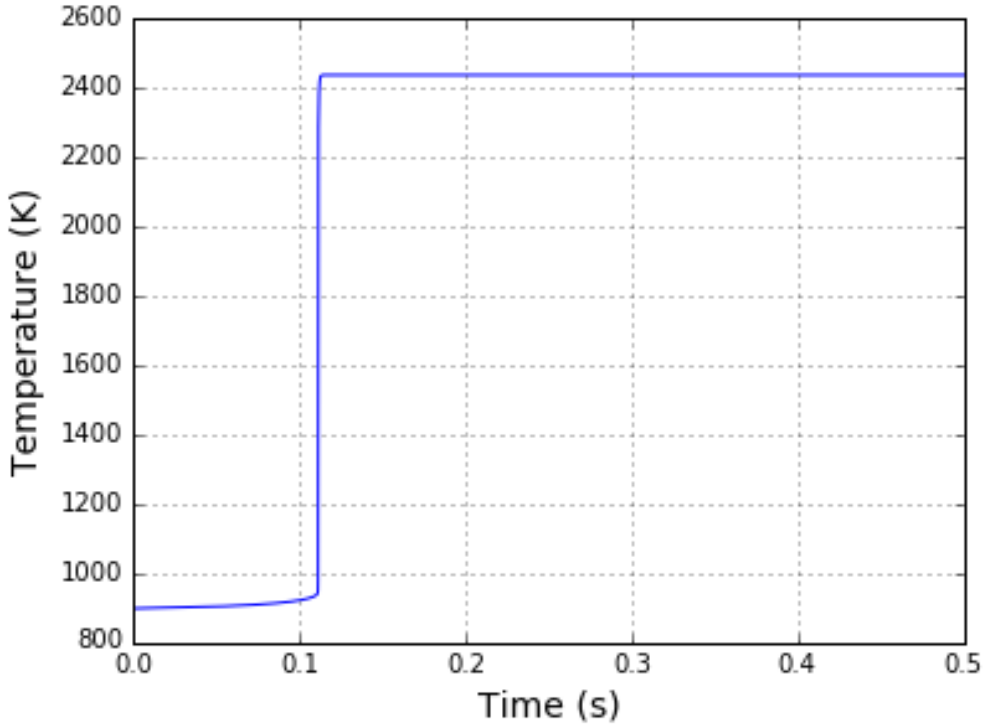
Partially Stirred Reactor (PaSR)



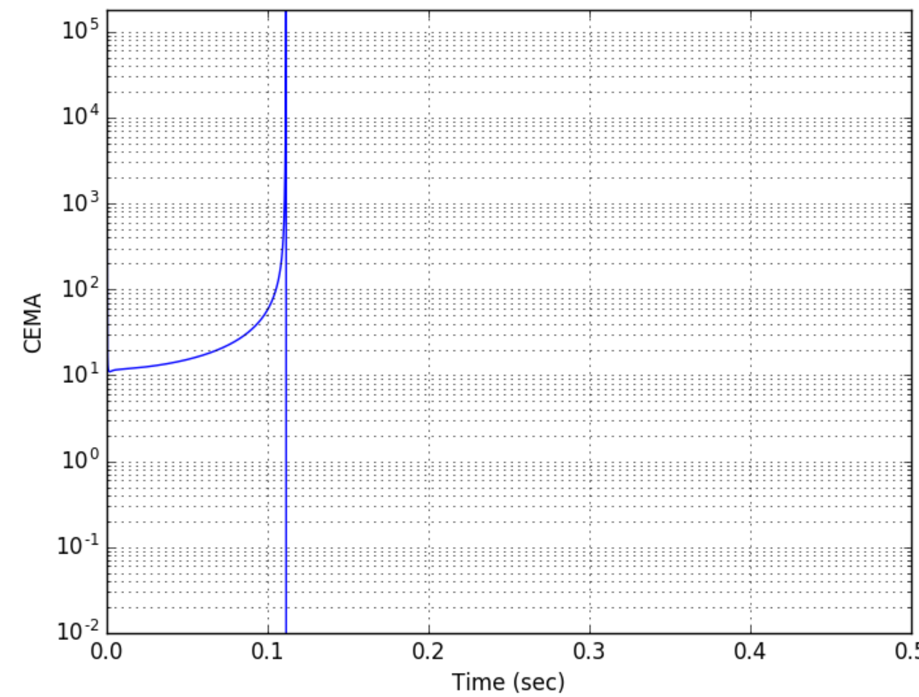


- Cantera-based PaSR implementation; premixed combustion with fresh fuel/air mixture & pilot streams
- Pairwise mixing, reaction fractional steps, inflow/outflow events

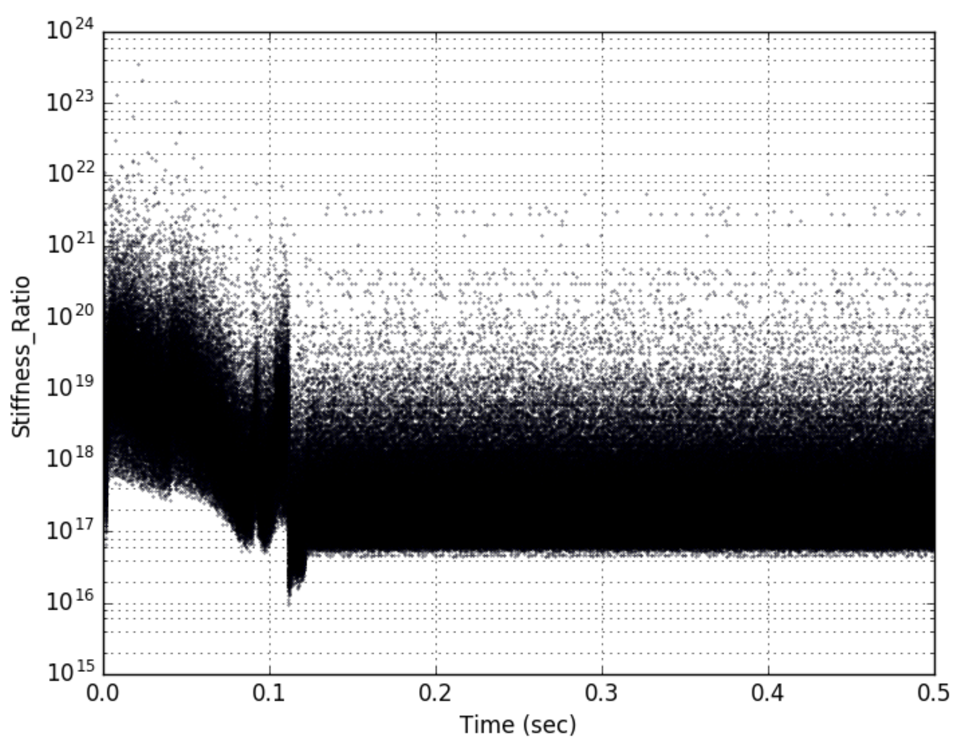
https://github.com/SLACKHA/pyJac



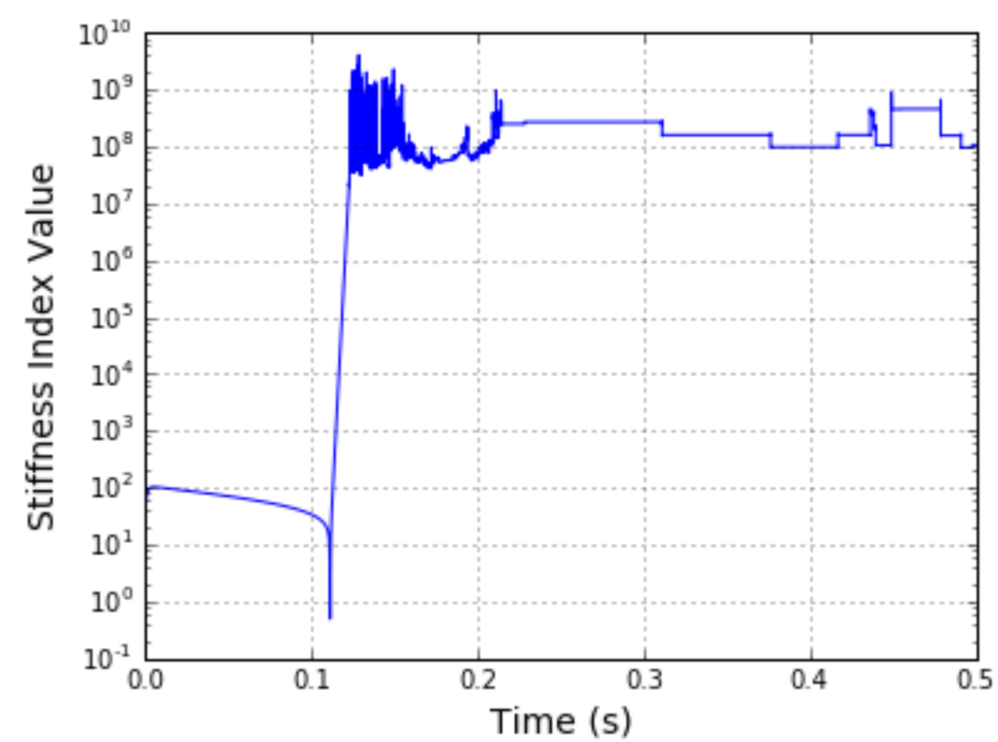
Temperature vs. time



Positive eigenvalue/CEMA

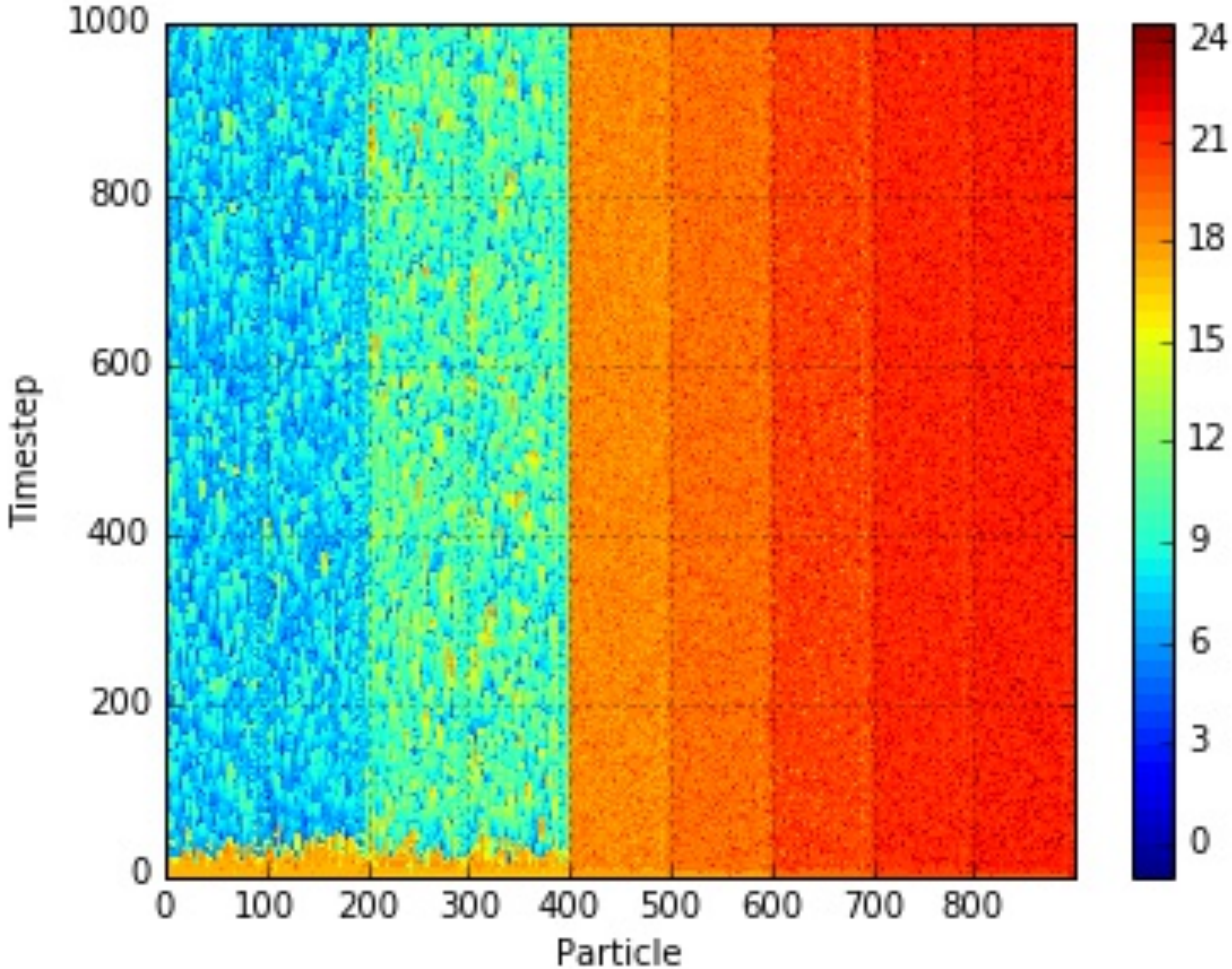


Stiffness ratio vs. time

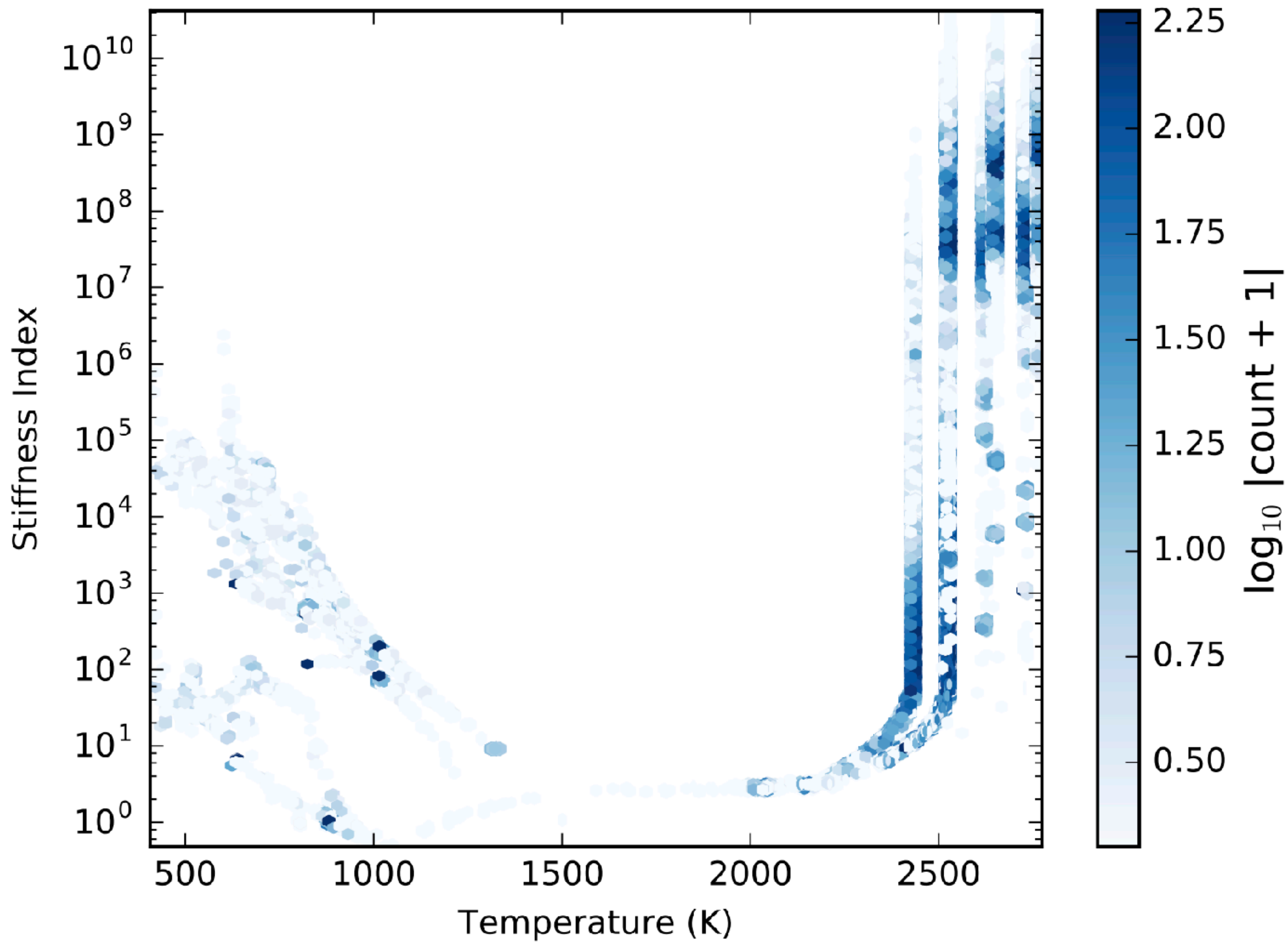


Stiffness index vs. time

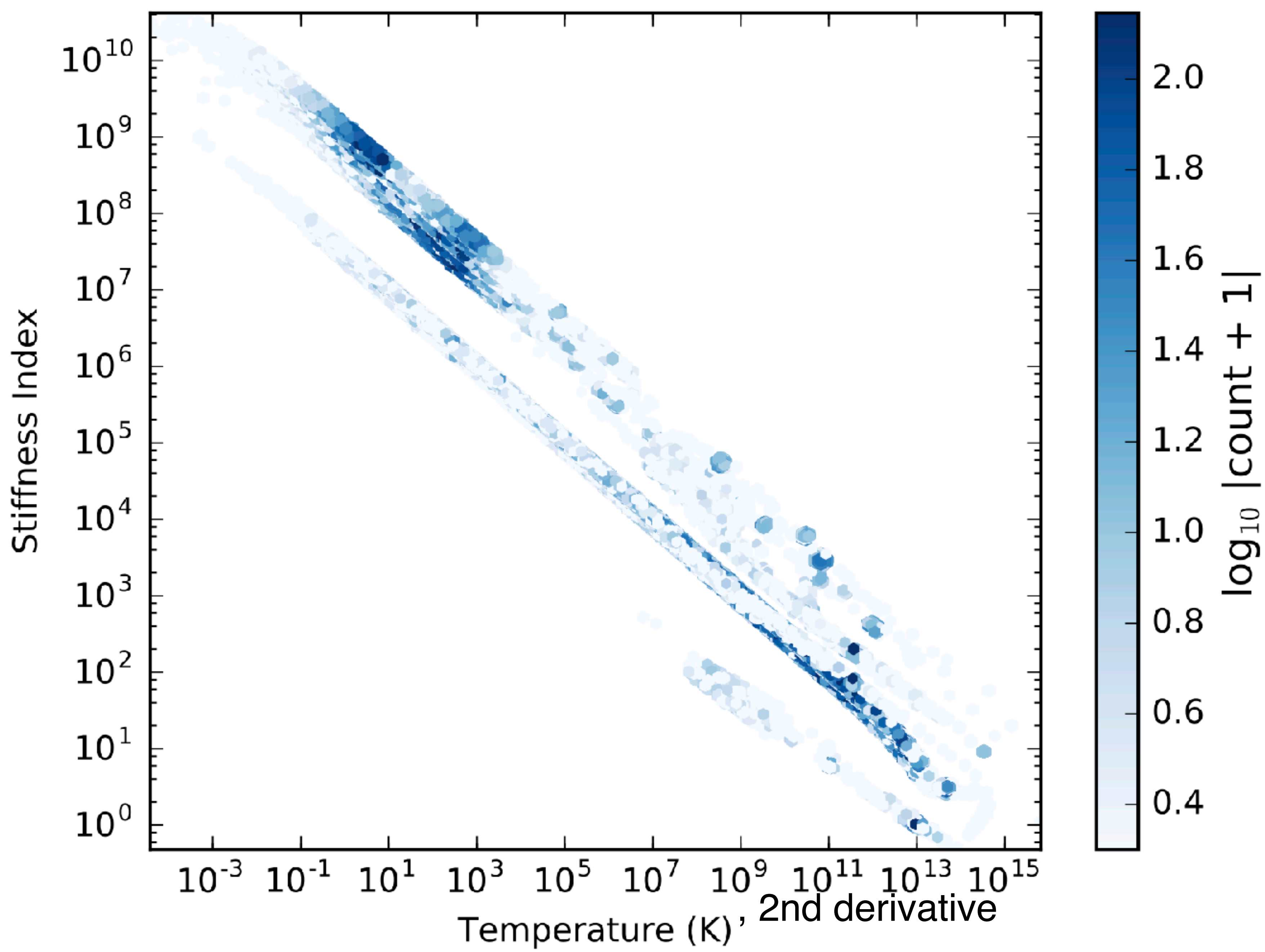
Sampled data



Stiffness index vs. temperature



Stiffness index vs. temperature 2nd derivative



Stiffness characterization: future work

Stiffness characterization: future work

- Investigating additional metrics (e.g., “stiffness indicator”)

Stiffness characterization: future work

- Investigating additional metrics (e.g., “stiffness indicator”)
- Also comparing stiffness prediction with “actual” stiffness: computational cost

Stiffness characterization: future work

- Investigating additional metrics (e.g., “stiffness indicator”)
- Also comparing stiffness prediction with “actual” stiffness: computational cost
- Next steps: use metrics to switch integrators, and evaluate improvement in performance

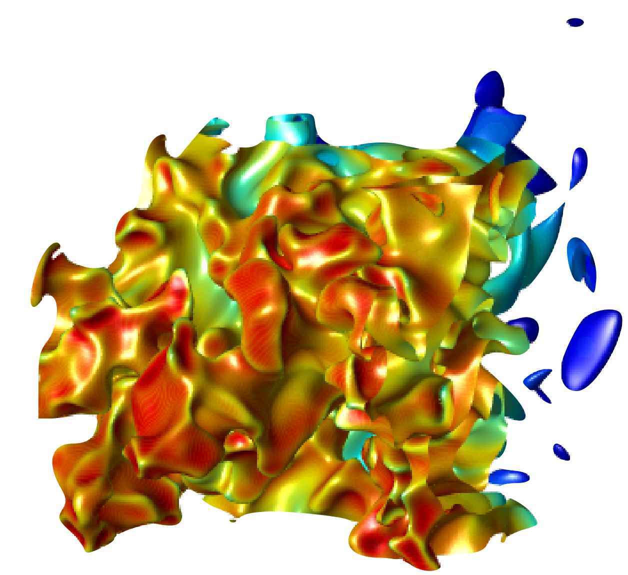
Other ongoing efforts

Group themes

- Combustion/reactive flow modeling
- Numerical methods for CFD
- Ocean biogeochemistry/turbulence
- Smoldering combustion
- Open science

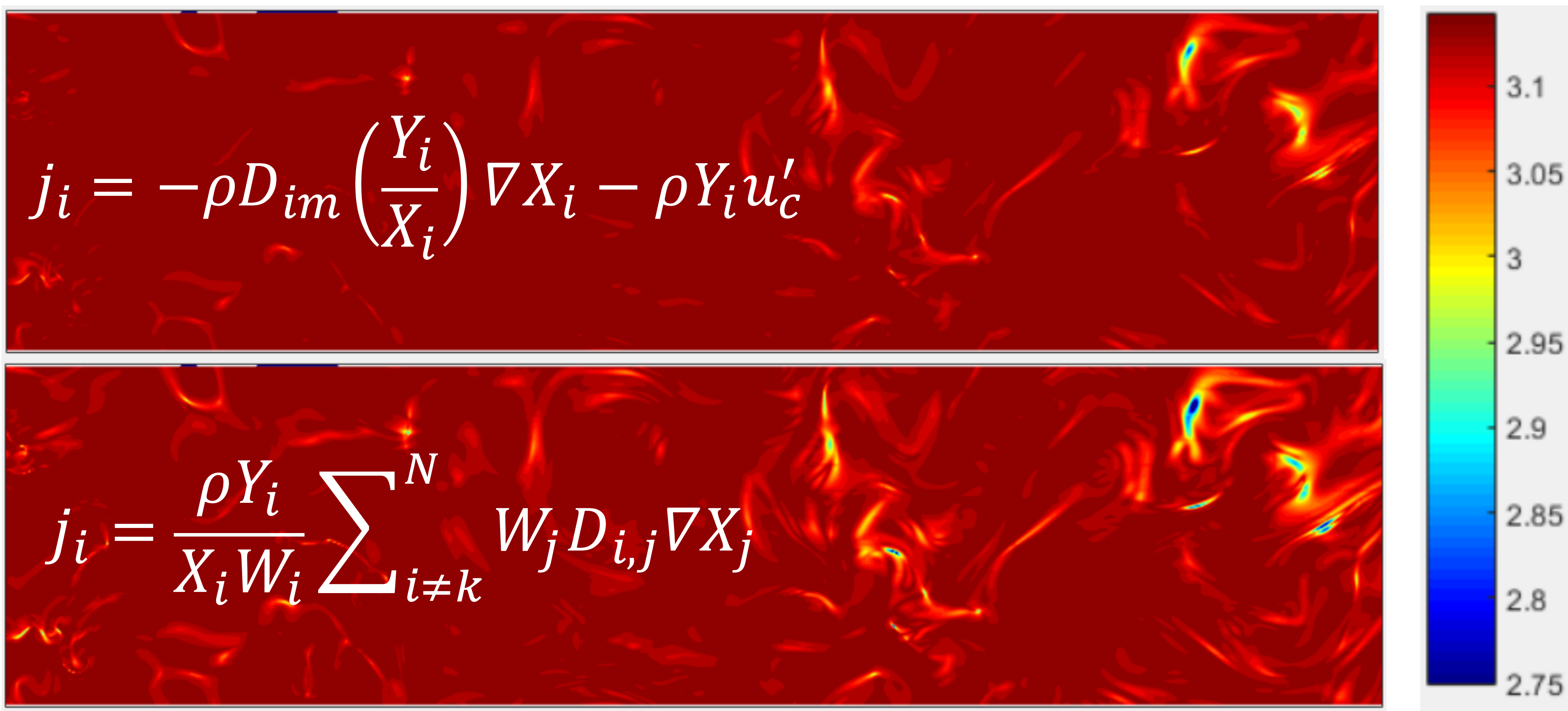
Importance of multicomponent diffusion in turbulent flames

- Student: AJ Fillo; collaboration with Prof. Guillaume Blanquart at CalTech
- DNS supposedly "model-free", but community relies on mixture-averaged (or simpler) approximation for diffusion
- Differences in *laminar* flames have been observed, and recent studies pointed out affect of differential diffusion on *turbulent* flame speed/structure



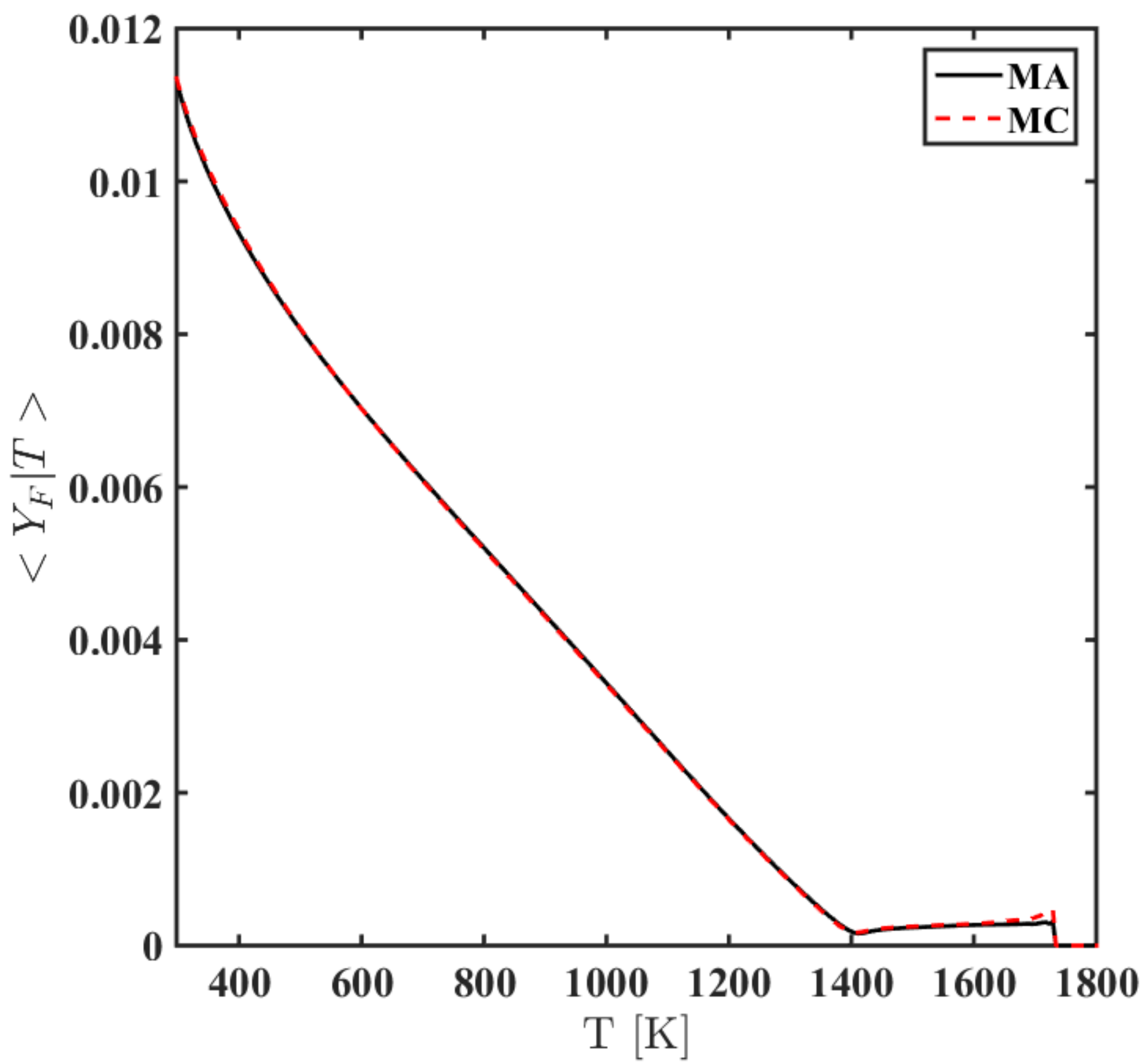
Flux angle contours

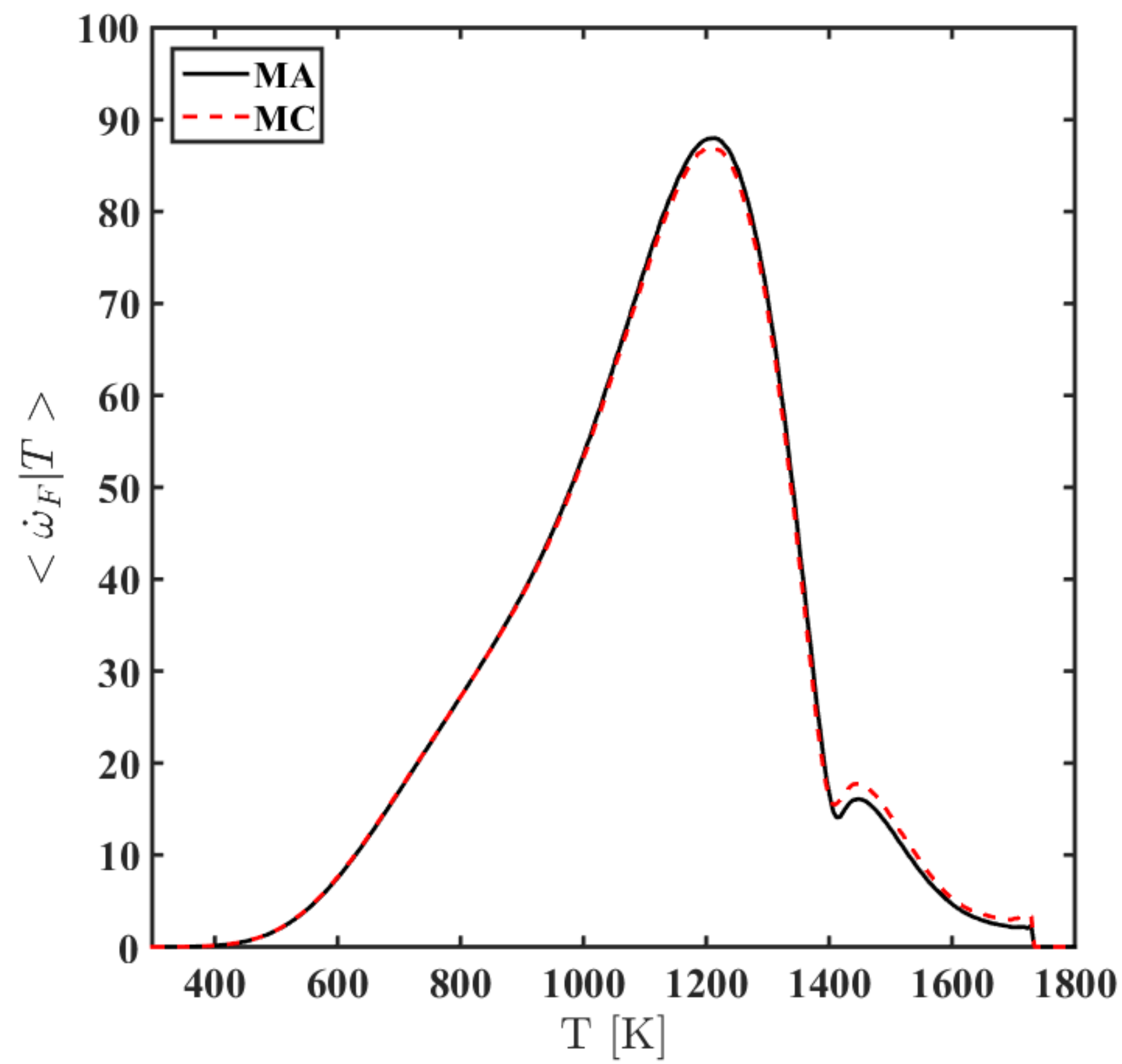
mixture-averaged



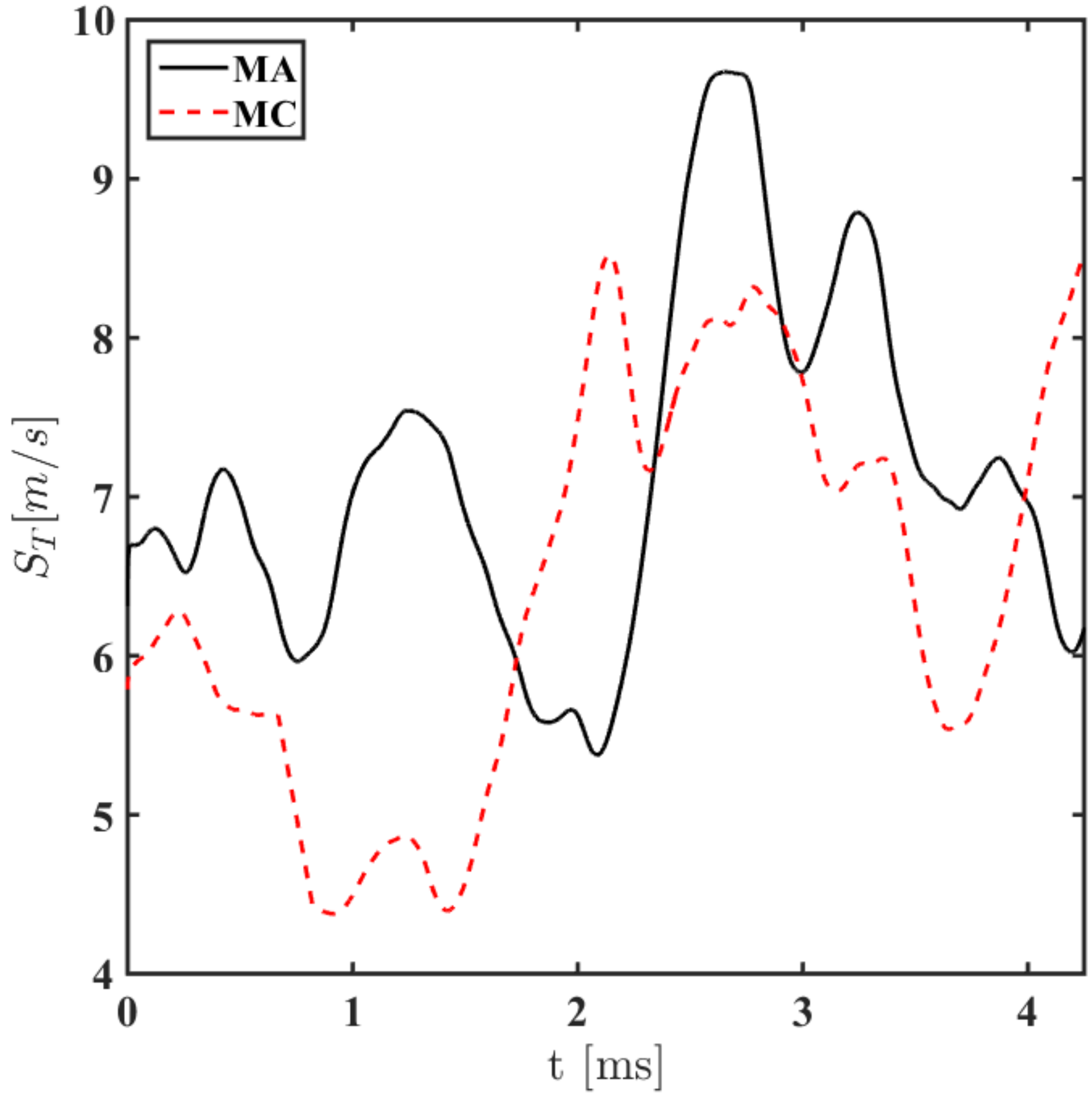
multicomponent

Conditional means

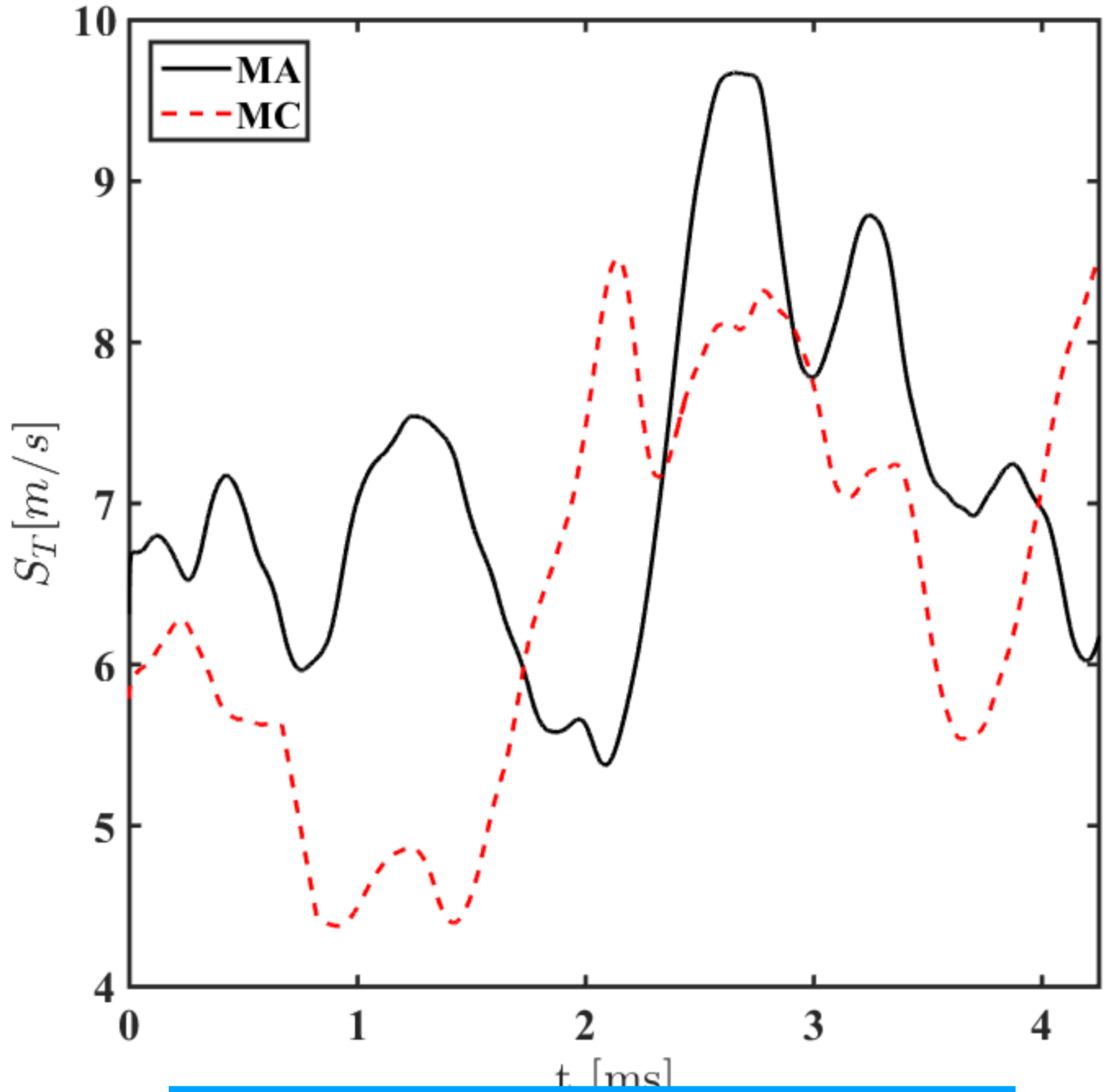




Turbulent flame speed



Turbulent flame speed



~8% difference

Effect of kinetic model reduction on turbulent flame characteristics

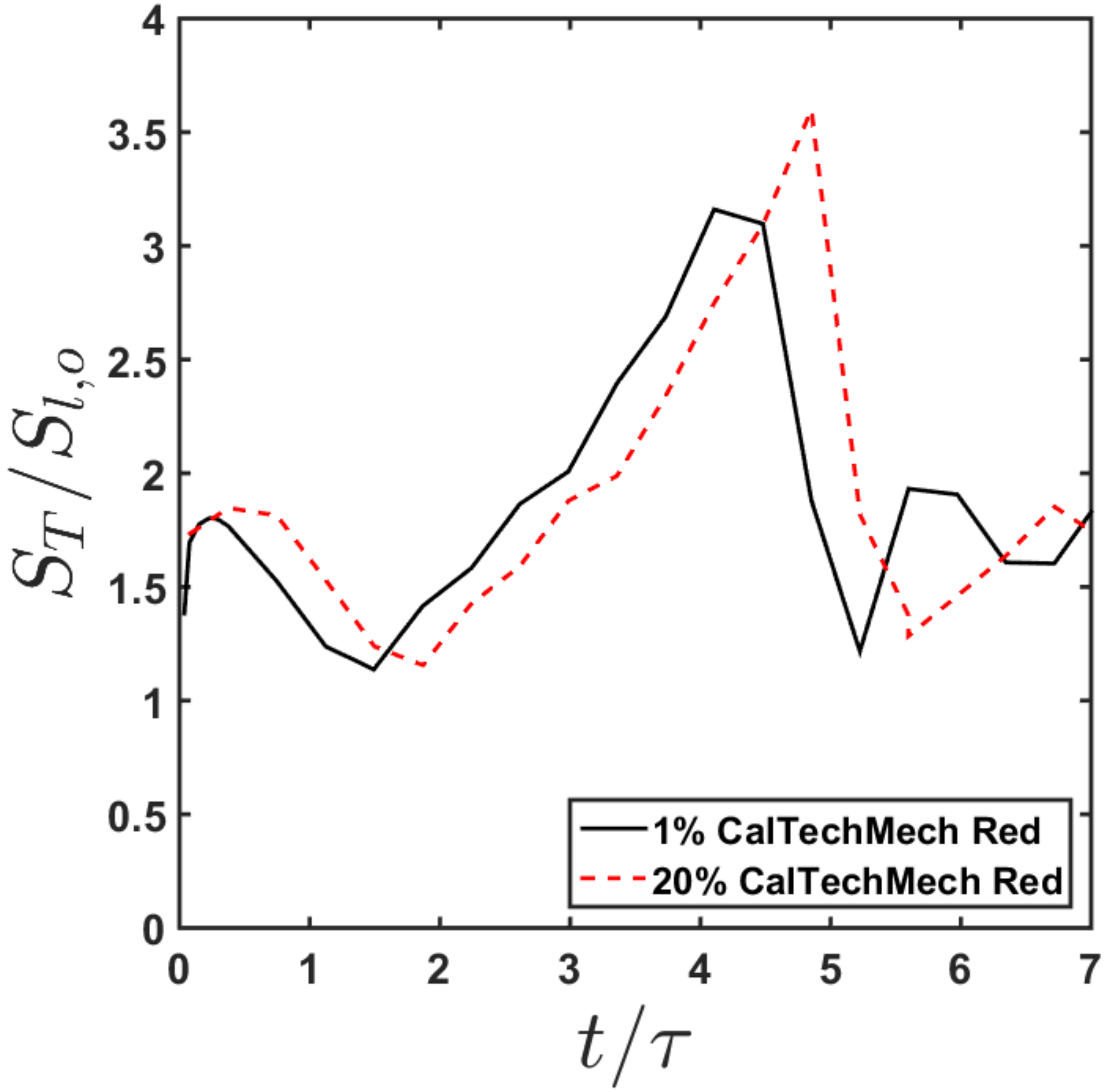
- Student: AJ Fillo

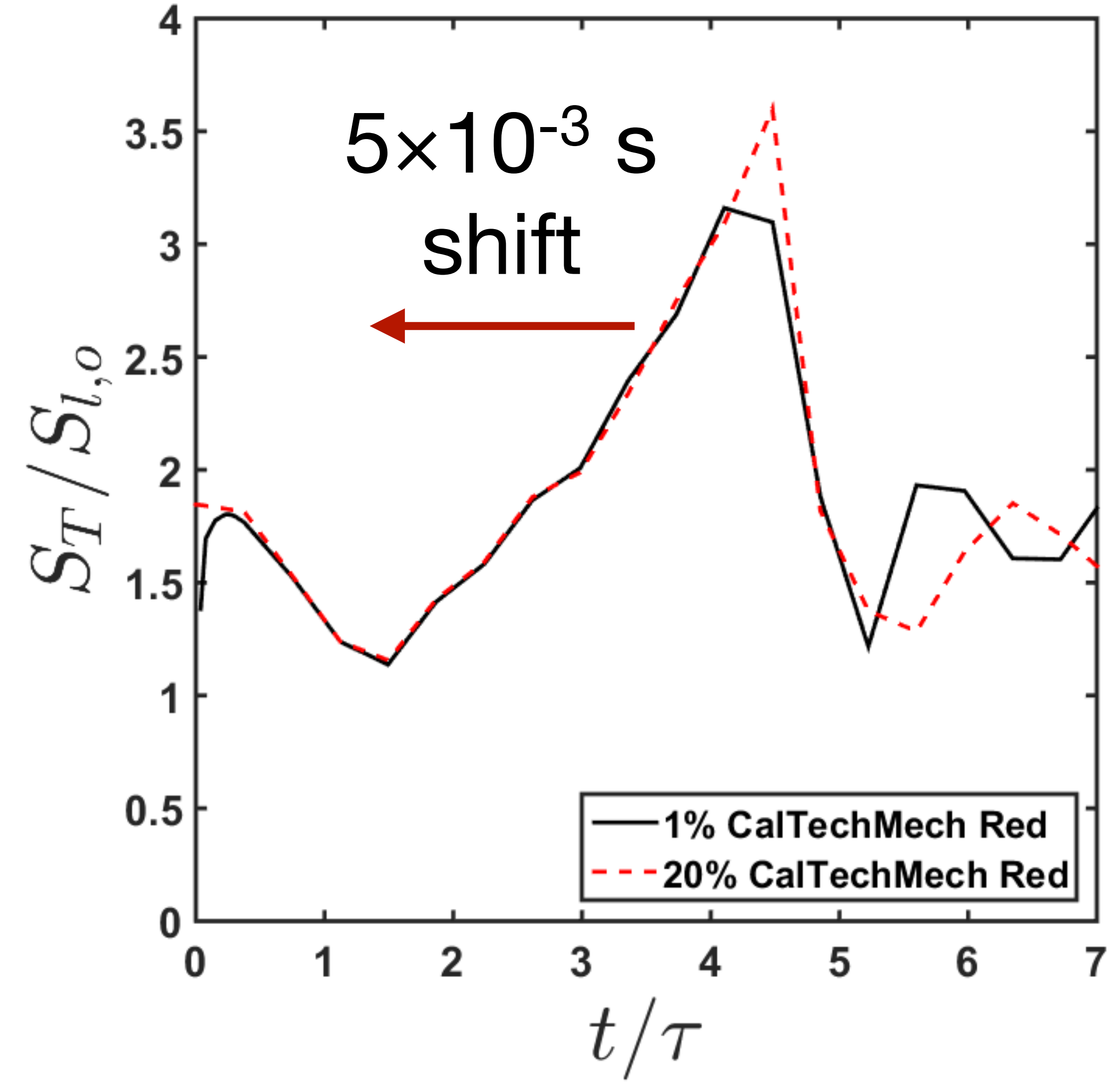
- Common to perform chemical kinetic model reduction and validate against detailed model using homogeneous or laminar phenomena: autoignition, PSR, laminar flame speed

- Assumed that “good” comparison in these implies “good” performance in unsteady, turbulent flames—but this has not been confirmed

- Our work: compare detailed *n*-heptane model (174 species) with reduced models in premixed turbulent flames

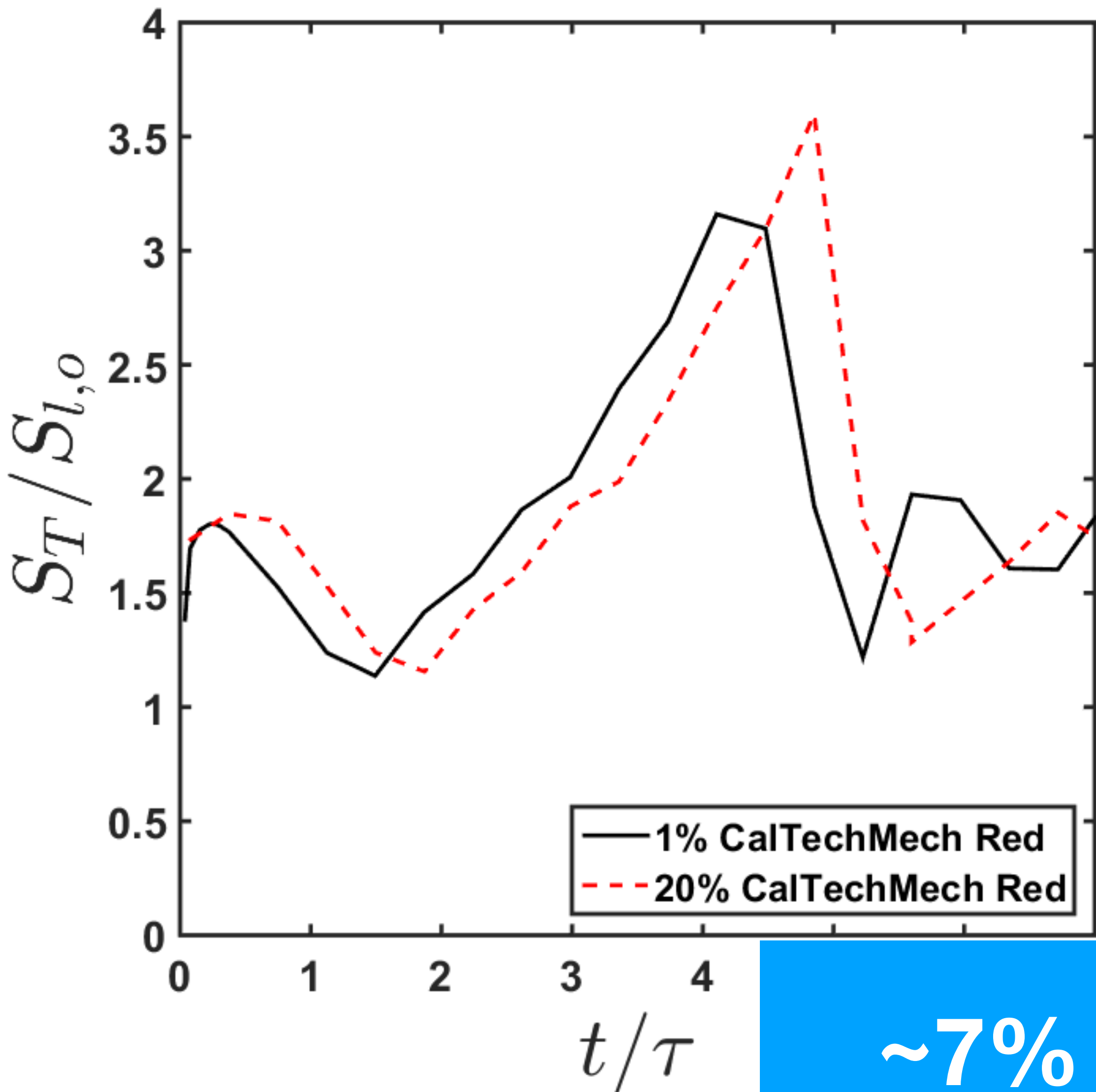
Turbulent flame speed

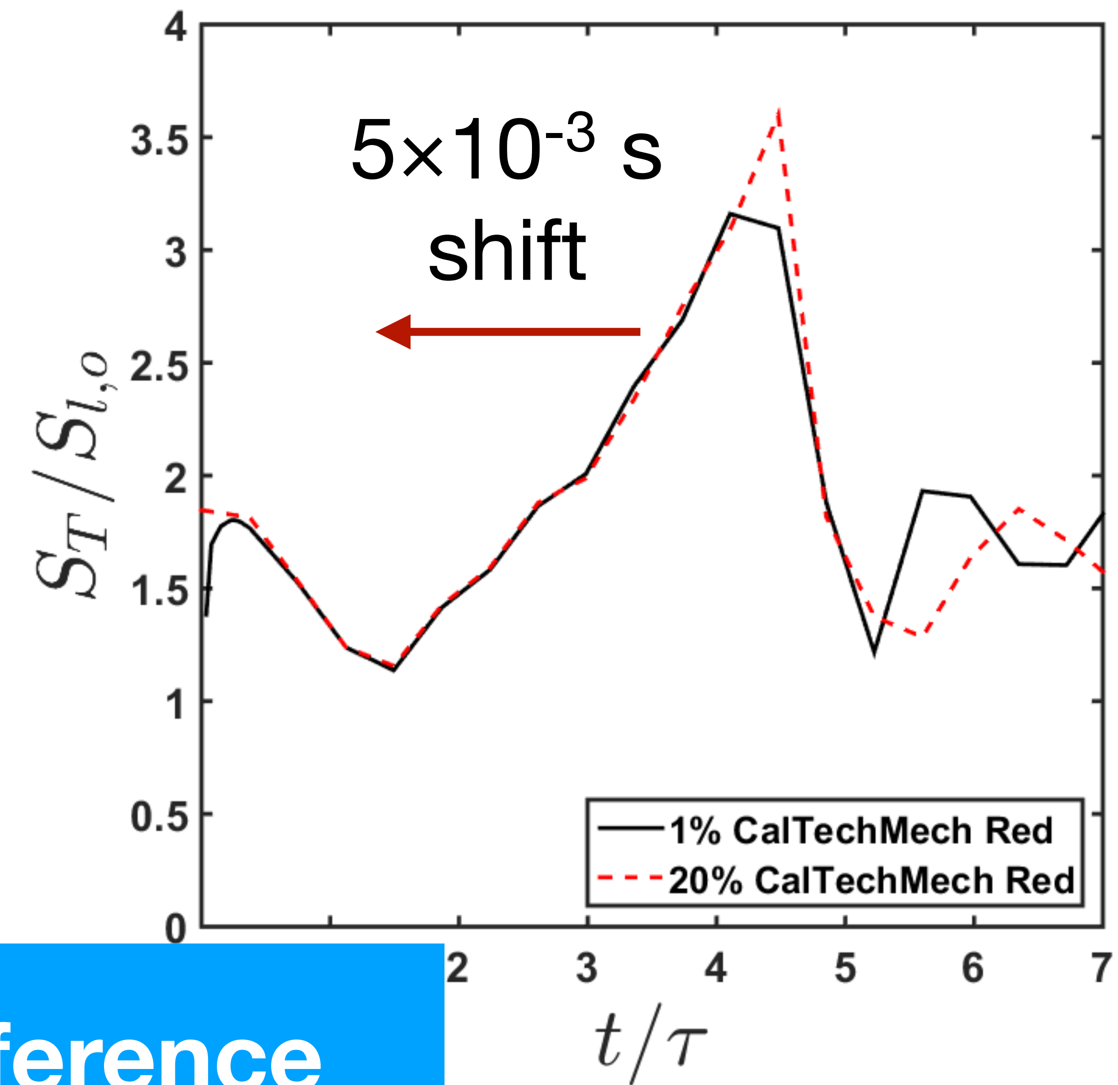




	1% reduction	20% reduction
$S_{L,0}$	37.5 cm/s	36.7 cm/s
S_T	69.3 cm/s	74.04 cm/s

Turbulent flame speed

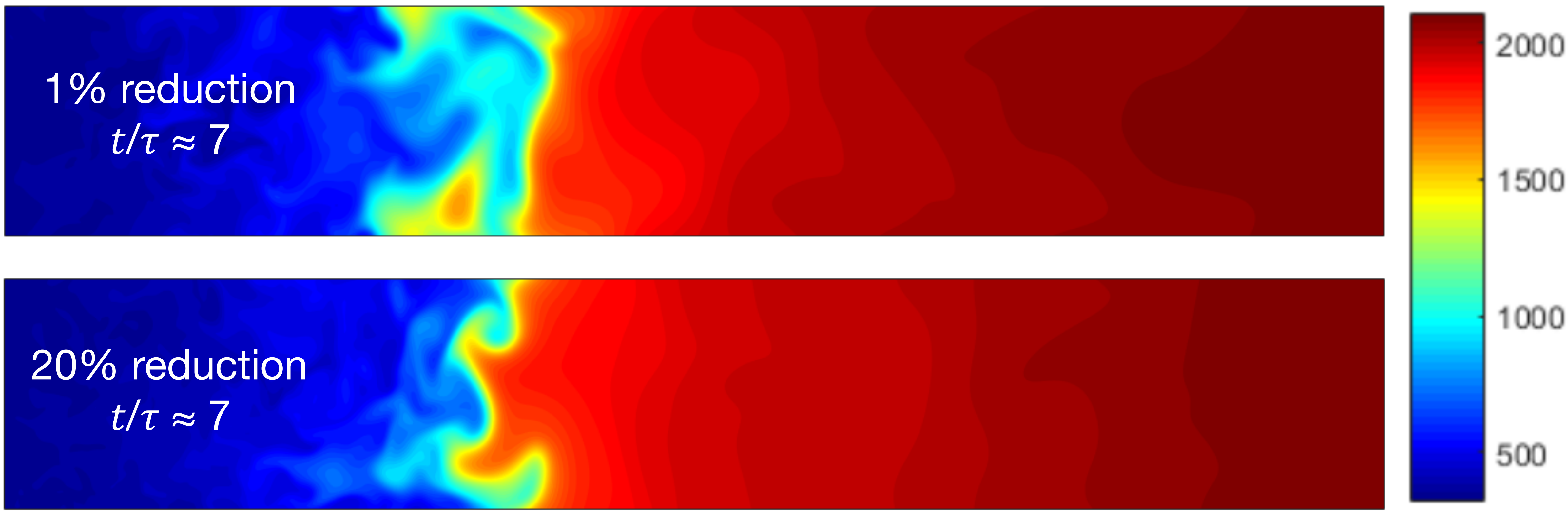




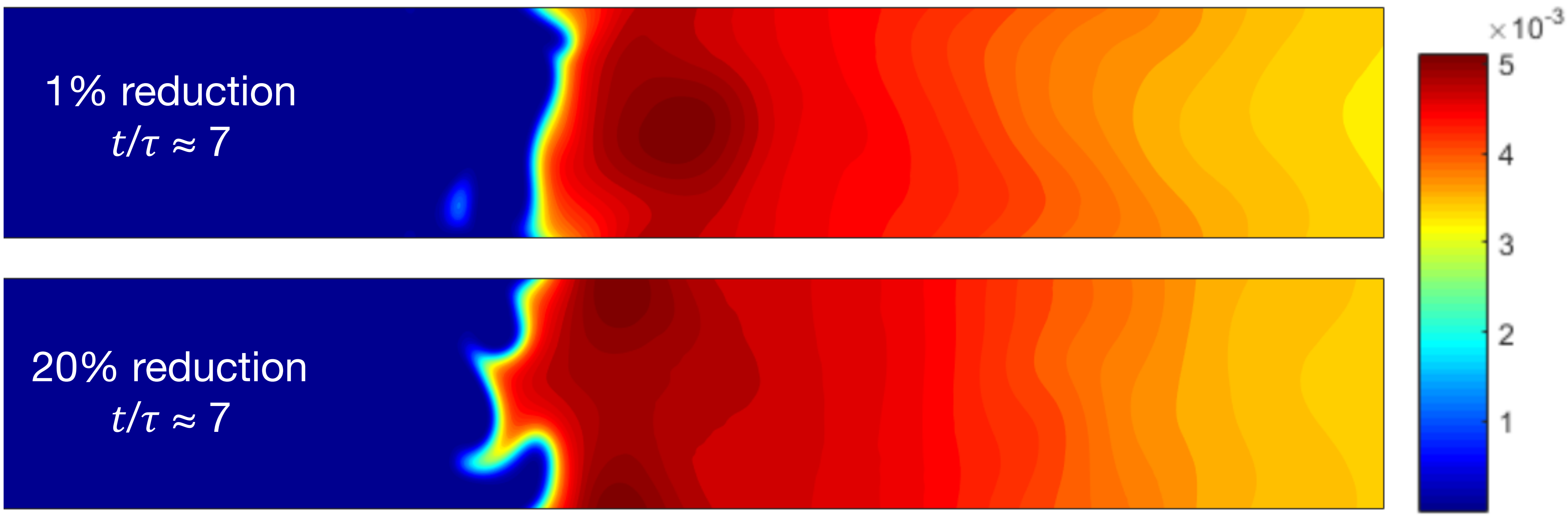
~7% difference

	1% reduction	20% reduction
$S_{L,0}$	37.5 cm/s	36.7 cm/s
S_T	69.3 cm/s	74.04 cm/s

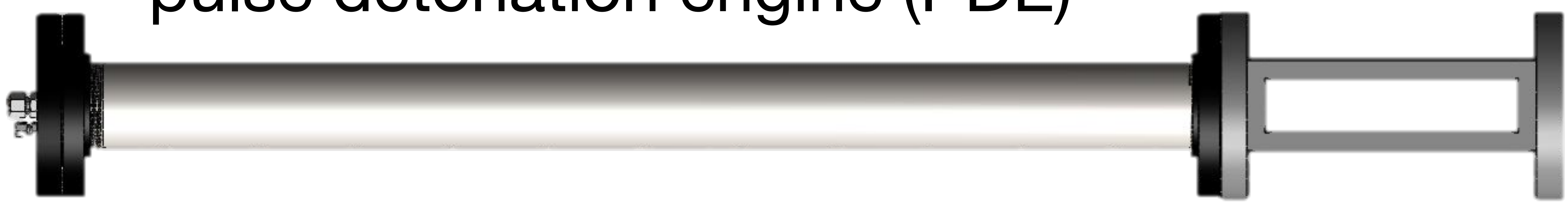
Temperature contour



OH mass fraction contour



PDE-MHD for oxycoal power generation



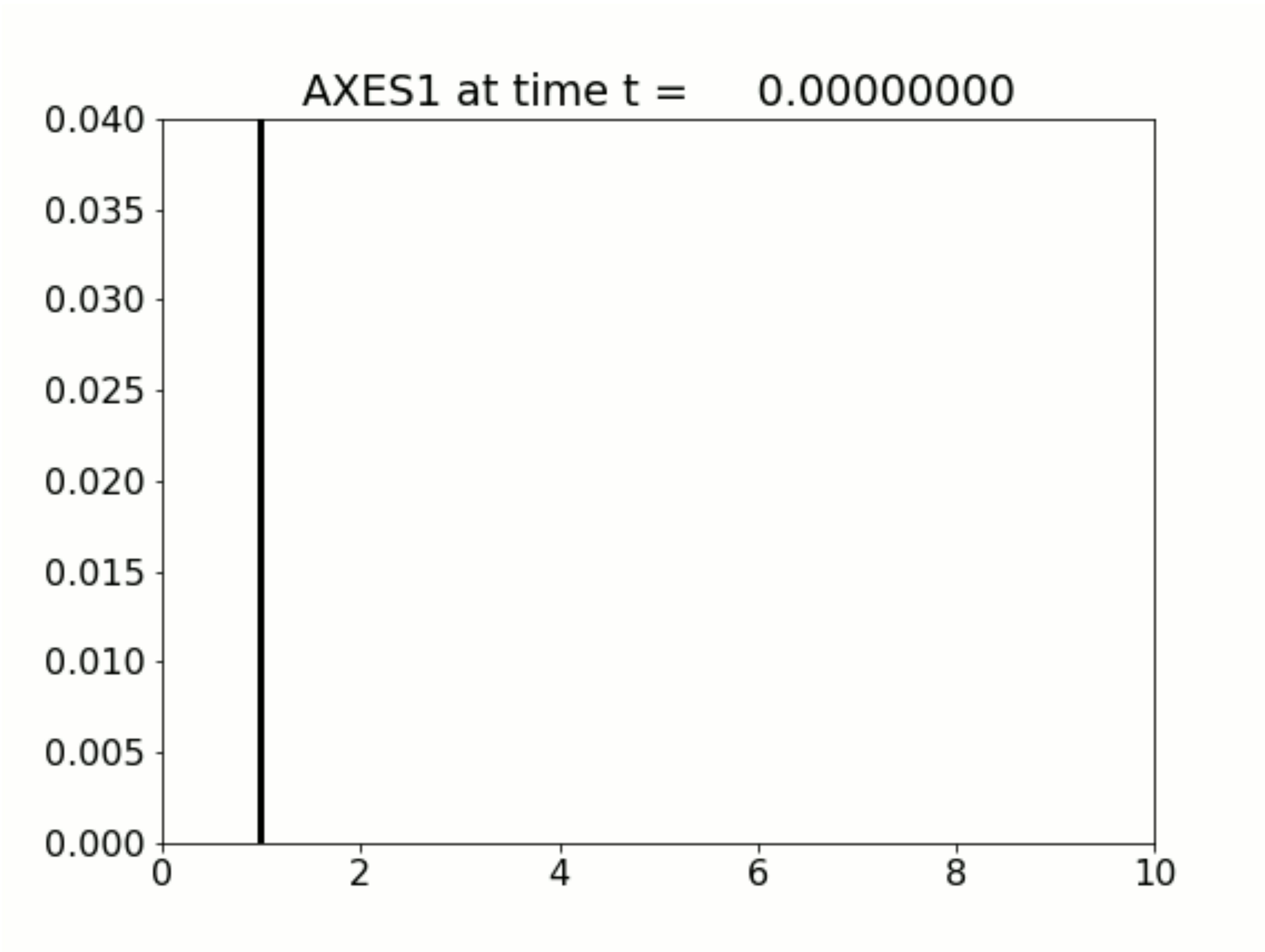
- PDE-MHD potential for oxycoal combustion: high efficiency (topping cycle) & direct power extraction—no moving parts

- Questions about interaction between detonation and MHD/ seed particle ionization, and potential power generation

PDE-MHD for oxycoal power generation

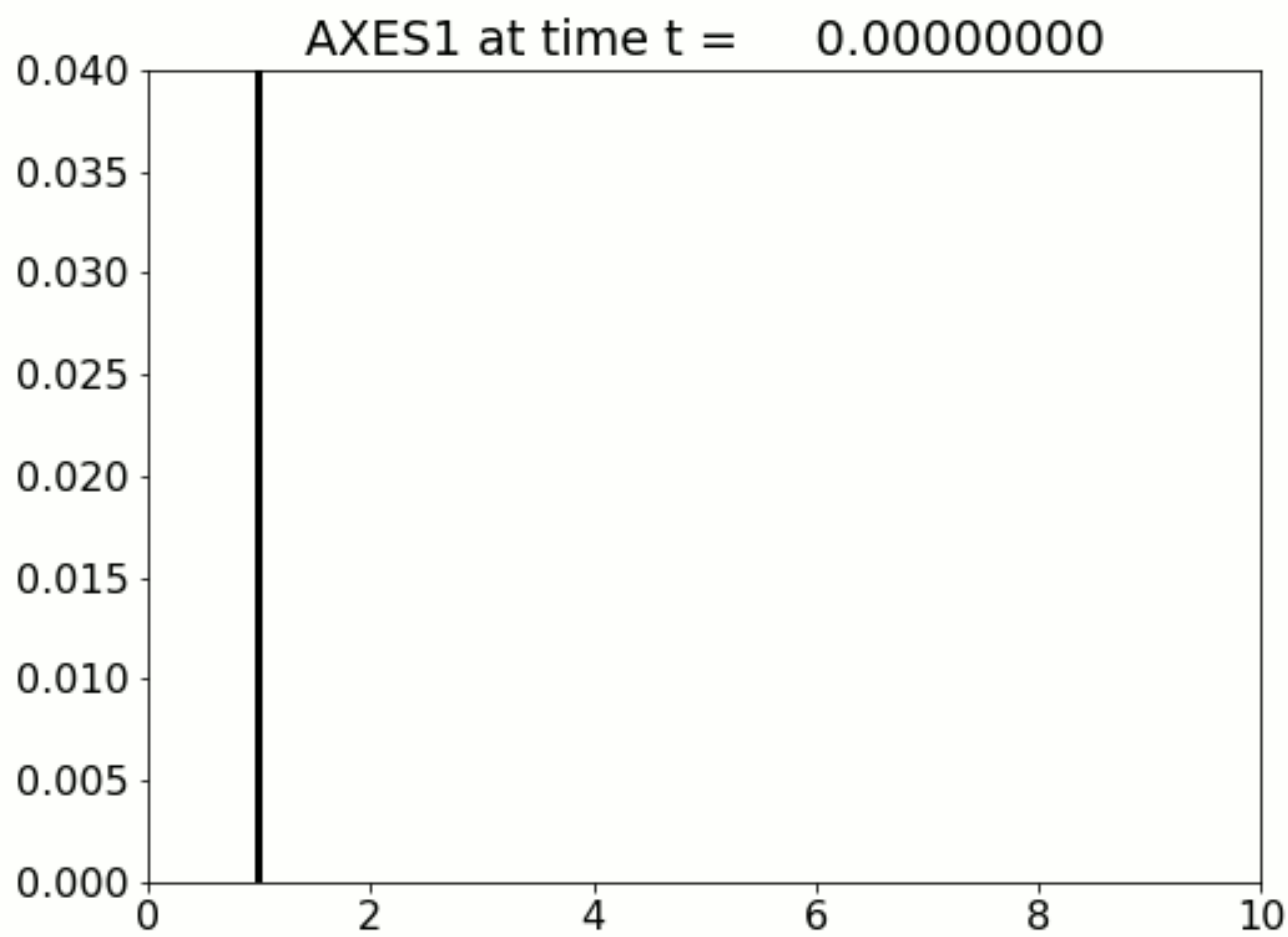
- Student: Matt Zaiger, collaboration with Prof. David Blunck at Oregon State
- Method: use CLAWpack + Cantera to solve reactive Euler equations

PDE: H_2+O_2



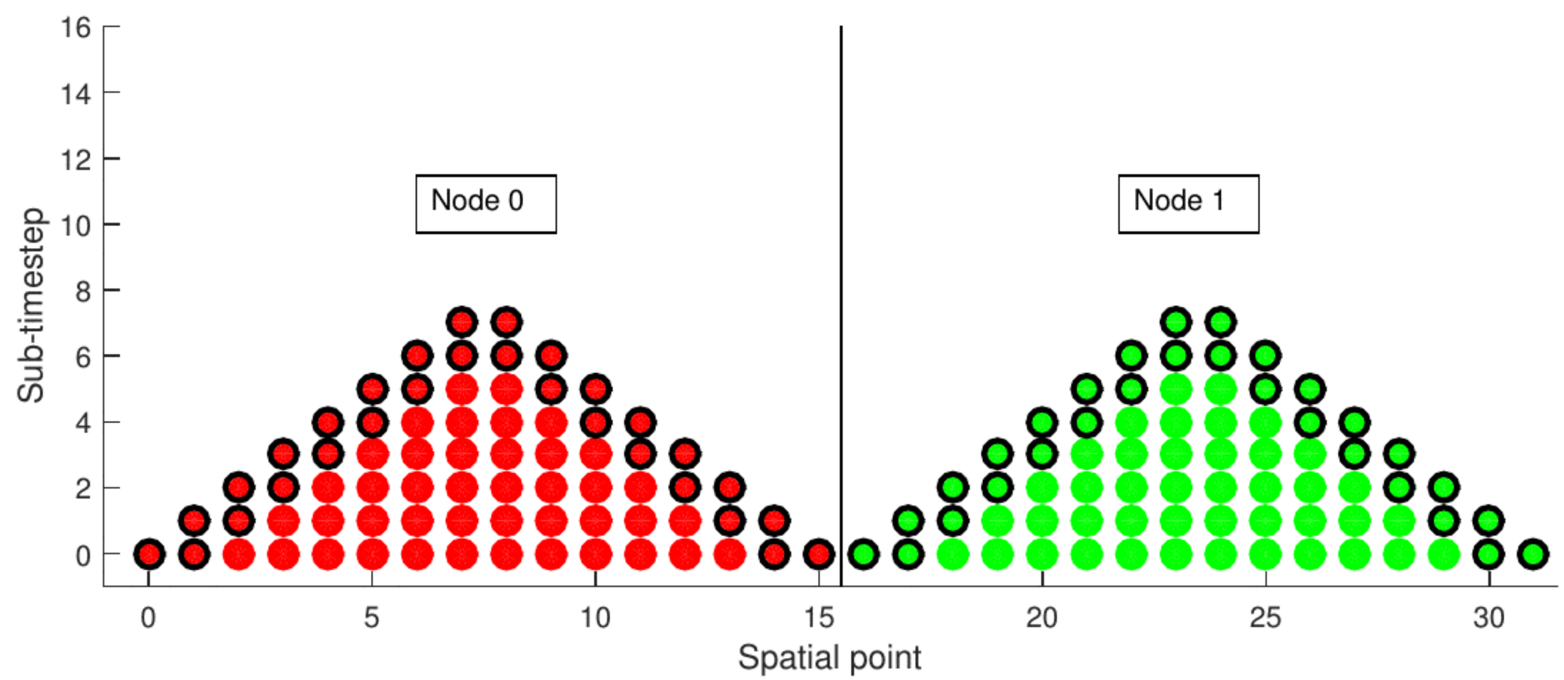
AXES1 at time t = 0.00000000
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
2
4
6
8
10

PDE: H_2+O_2


AXES1 at time t = 0.00000000
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
2
4
6
8
10

Swept time-space domain decomposition

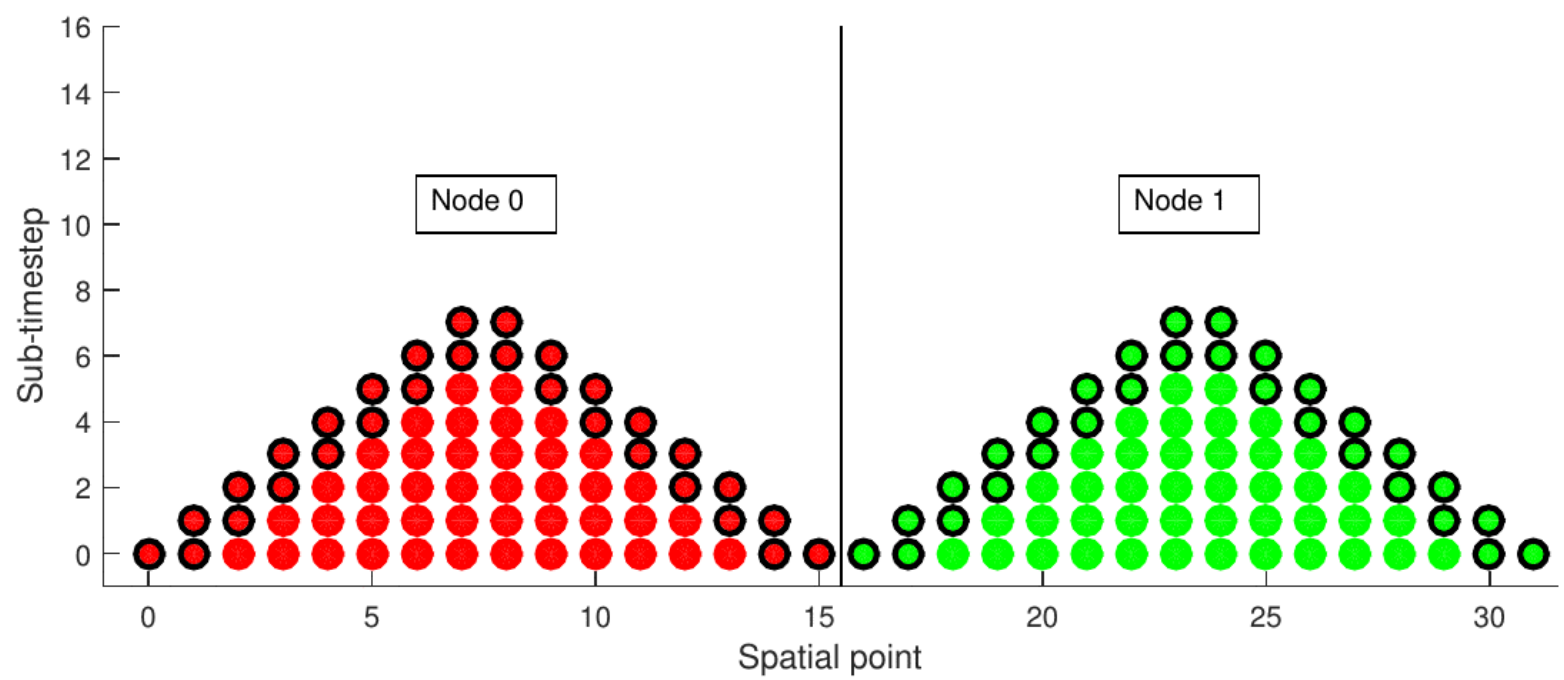
- Student: Daniel Magee; collaboration with Qiqi Wang (MIT) and David Gleich (Purdue)
- Main idea: reduce communication in distributed parallel PDE solution by performing all possible calculations in subdomain
- Our work: designed GPU-capable version of algorithm, tested with various 1D PDEs



D. J. Magee & K. E. Niemeyer. "Accelerating solutions of PDEs with GPU-based swept time-space decomposition."Under review, 2017. arXiv:1705.03162 [physics.comp-ph]

Swept time-space domain decomposition

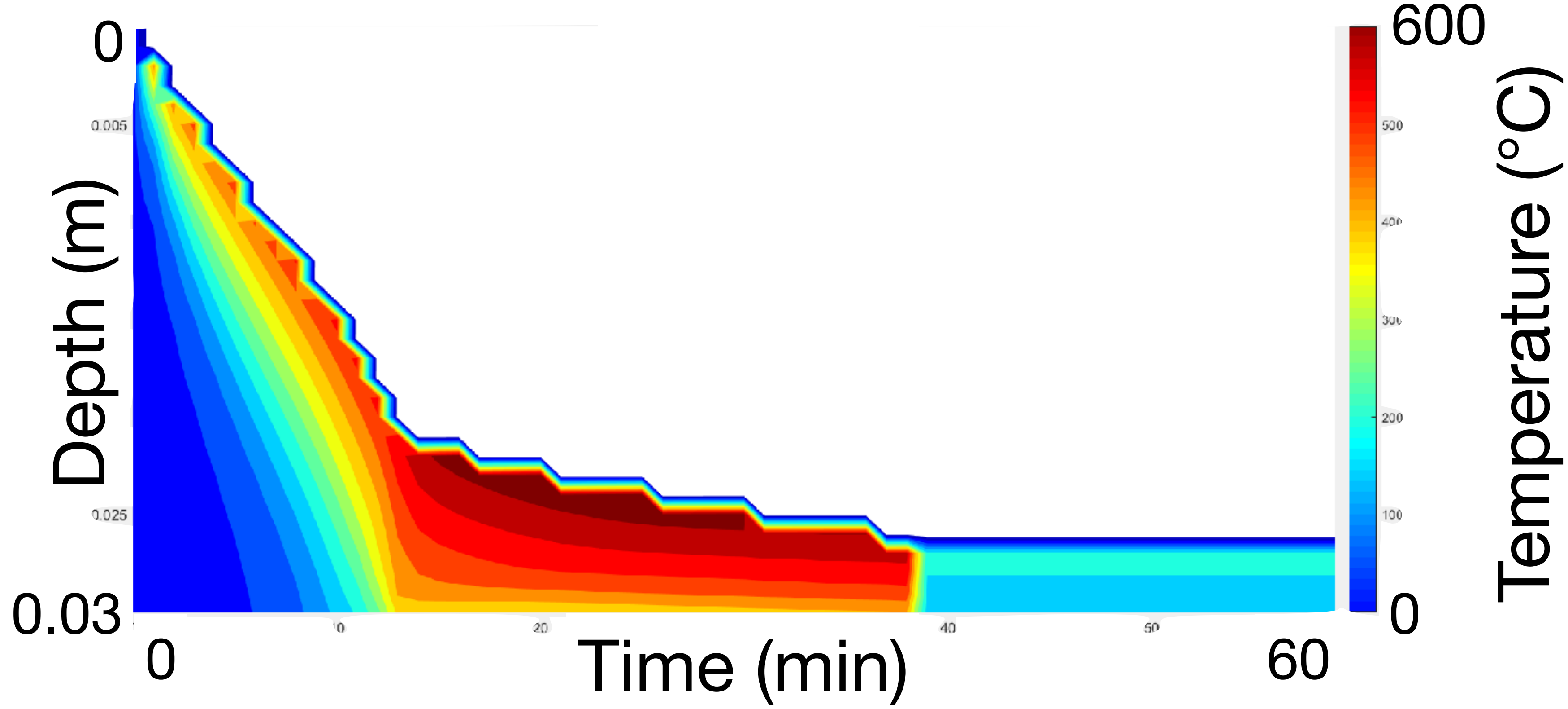
- Student: Daniel Magee; collaboration with Qiqi Wang (MIT) and David Gleich (Purdue)
- Main idea: reduce communication in distributed parallel PDE solution by performing all possible calculations in subdomain
- Our work: designed GPU-capable version of algorithm, tested with various 1D PDEs



D. J. Magee & K. E. Niemeyer. "Accelerating solutions of PDEs with GPU-based swept time-space decomposition."Under review, 2017. arXiv:1705.03162 [physics.comp-ph]

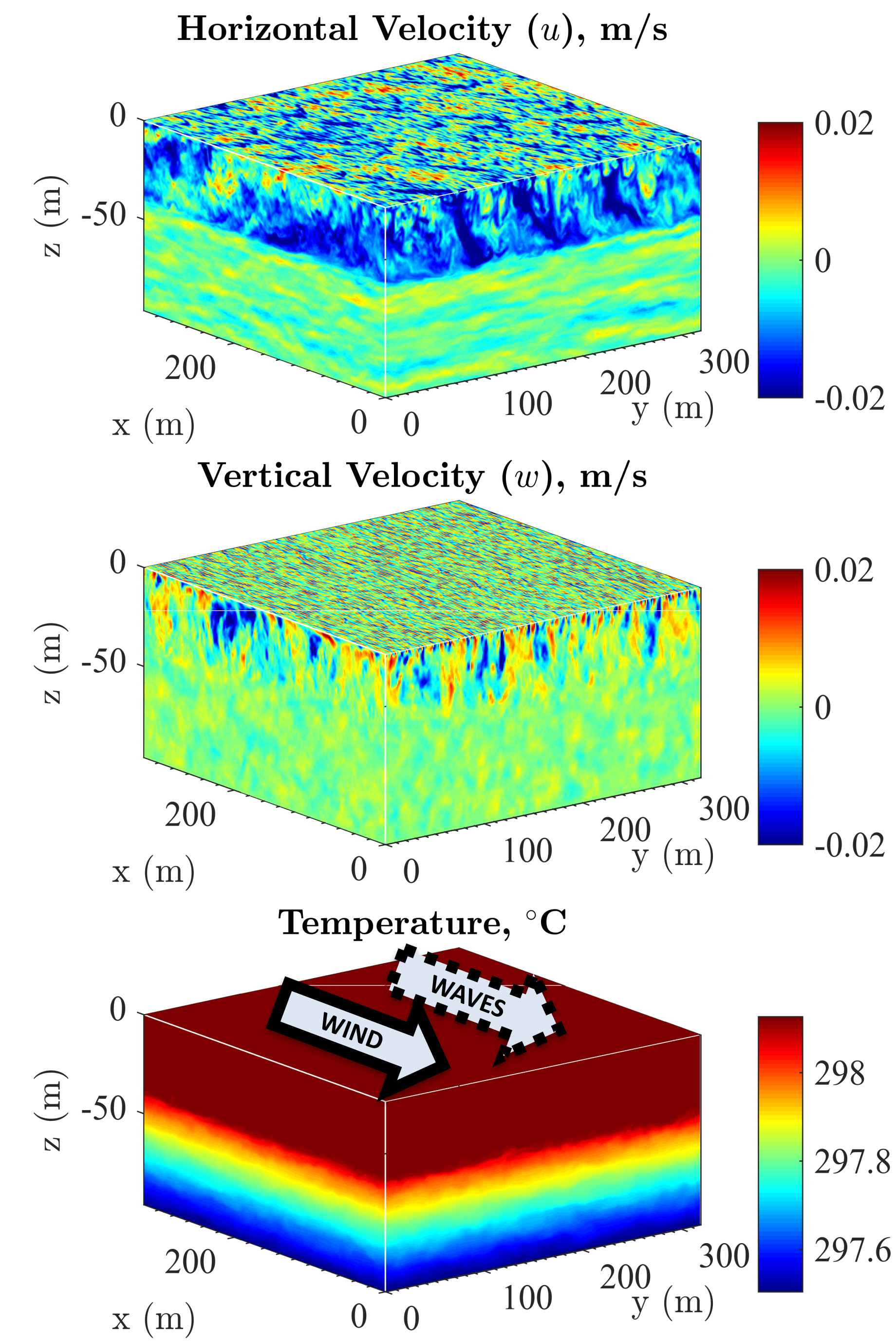
Smoldering combustion of wood fuels

- Smoldering combustion of wood fuels not well understood—what parameters control ignition & propagation?
- Student: Tejas Mulky; collaboration with Prof. David Blunck @ Oregon State
- Fuels of interest: wood-like combinations of cellulose, hemicellulose, & lignin
- Peat smoldering propagation:



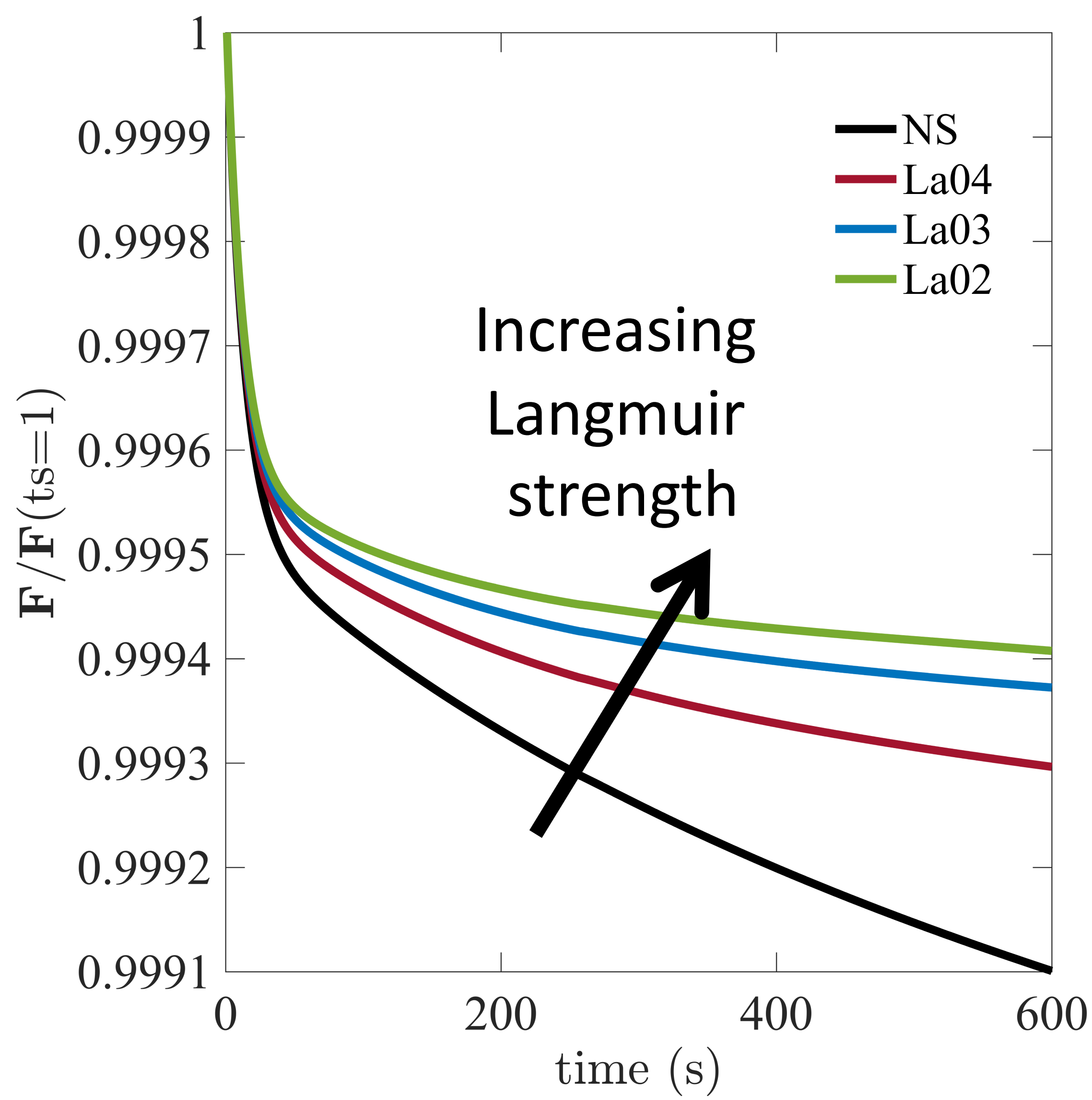
Interaction between ocean biogeochemistry and turbulence

- Student: Luz Pacheco; collaboration with Katherine Smith & Prof. Peter Hamlington @ CU Boulder
- Much like in combustion, in the ocean strong interactions occur between (biogeo)chemistry and turbulence
- Currently: interaction between wave-driven Langmuir turbulence and carbonate chemistry.
- Developing new solver based on FEniCS



K Smith, P Hamlington, K Niemeyer, B Fox-Kemper, & N Lovenduski. "Effects of Langmuir Turbulence on Upper Ocean Carbonate Chemistry", presented at 21st Conference on Atmospheric and Oceanic Fluid Dynamics (2017), Portland OR

Interaction between ocean biogeochemistry and turbulence



Flux rate of CO2 across the air-sea interface as a function of time normalized by initial flux rate

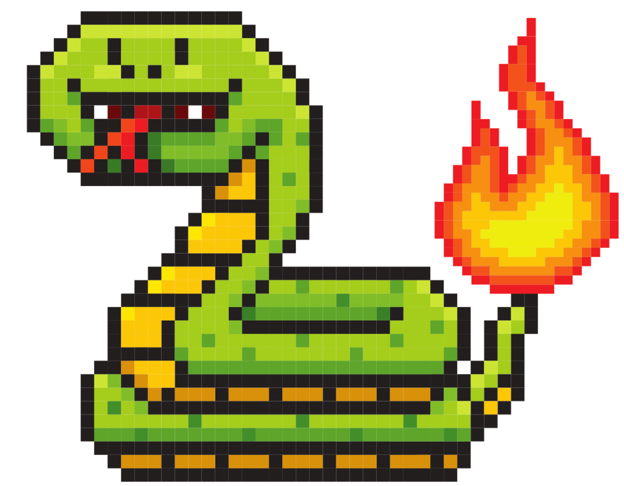
K Smith, P Hamlington, K Niemeyer, B Fox-Kemper, & N Lovenduski. "Effects of Langmuir Turbulence on Upper Ocean Carbonate Chemistry", presented at 21st Conference on Atmospheric and Oceanic Fluid Dynamics (2017), Portland OR

pyMARS: chemical kinetic model reduction software

- Students: Phillip Mestas, Parker Clayton
- **Under development**: Python & Cantera-based, open-source version of MARS for automatically reducing chemical kinetic models
- Currently supports directed relation graph (DRG) method; DRG with error propagation and sensitivity analysis being added

https://github.com/Niemeyer-Research-Group/pyMARS

ChemKED: data format for fundamental combustion measurements



- Student: Morgan Mayer; collaboration with Dr. Bryan Weber @ Univ. Connecticut
- Human- and machine-readable, open standard for describing fundamental combustion experiments—currently, autoignition
- PyKED: Python-based software for validating and interacting with ChemKED files
- Also building database of files: Prometheus

https://github.com/pr-omethe-us/PyKED

JOSS: Journal of Open Source Software

- JOSS publishes (short) software articles
- Peers review article, software, and associated artifacts
- JOSS has an ISSN (2475-9066) and software articles receive Crossref DOI upon publication
- JOSS celebrated its first birthday in May 🎂
- 111 articles published in first year
- Now: 123 published articles and 68 submitted

`http://joss.theoj.org/`

`http://bit.ly/joss-scipy2017`

Group themes

- Combustion/reactive flow modeling
- Numerical methods for CFD
- Ocean biogeochemistry/turbulence
- Smoldering combustion
- Open science

Group themes

- Combustion/reactive flow modeling 🔥
- Numerical methods for CFD
- Ocean biogeochemistry/turbulence
- Smoldering combustion
- Open science

Group themes

- Combustion/reactive flow modeling 🔥
- Numerical methods for CFD 💨
- Ocean biogeochemistry/turbulence
- Smoldering combustion
- Open science

Group themes

- Combustion/reactive flow modeling 🔥
- Numerical methods for CFD 💨
- Ocean biogeochemistry/turbulence 💧
- Smoldering combustion
- Open science

Group themes

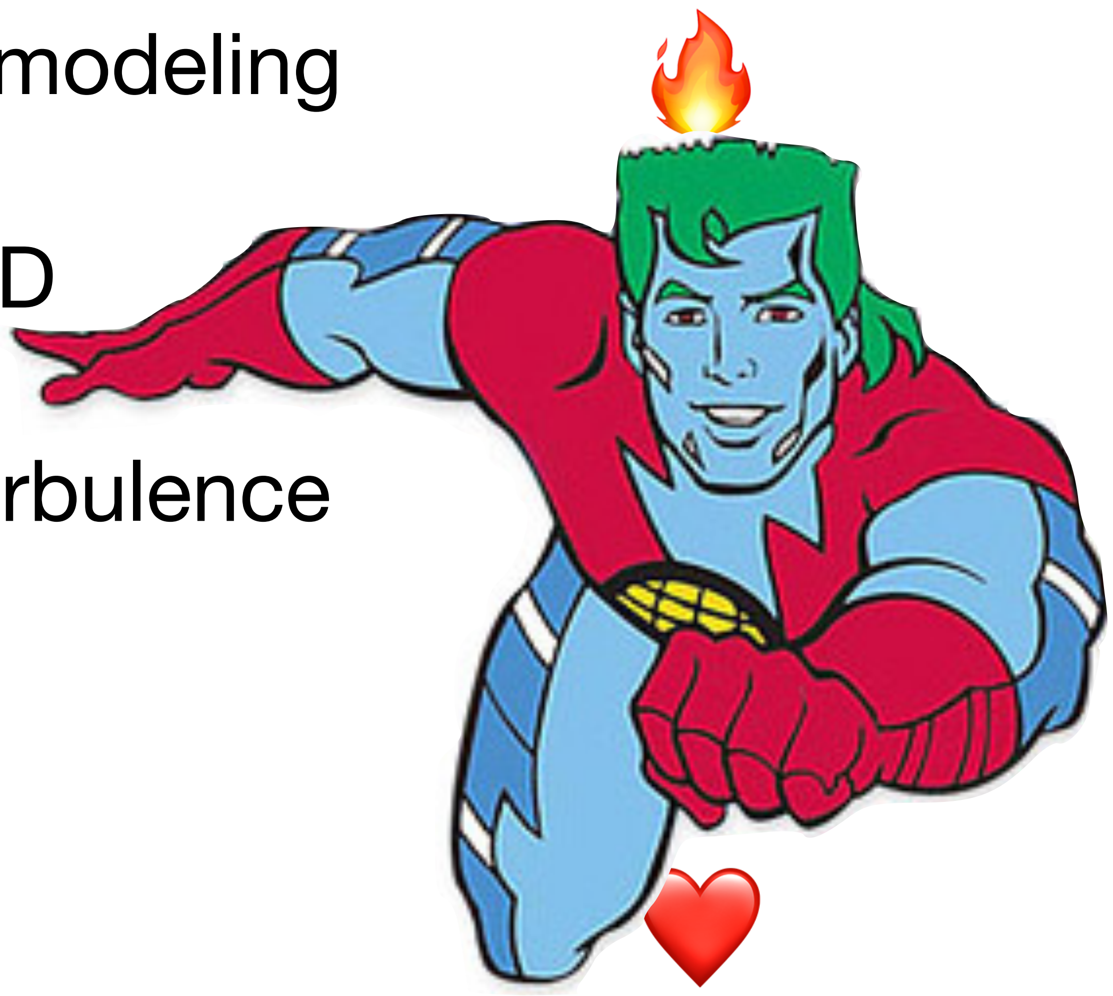
- Combustion/reactive flow modeling 🔥
- Numerical methods for CFD 💨
- Ocean biogeochemistry/turbulence 💧
- Smoldering combustion 🌎
- Open science

Group themes

- Combustion/reactive flow modeling 🔥
- Numerical methods for CFD 💨
- Ocean biogeochemistry/turbulence 💧
- Smoldering combustion 🌎
- Open science ❤️

Group themes

- Combustion/reactive flow modeling
- Numerical methods for CFD
- Ocean biogeochemistry/turbulence
- Smoldering combustion
- Open science



Thank you!
Questions?

https://git.io/nrg

?

Thank you!
Questions?

https://git.io/nrg