SAND2022-6229C

Studying the Computational Performance of CSP Analysis on Heterogeneous Computing Platforms

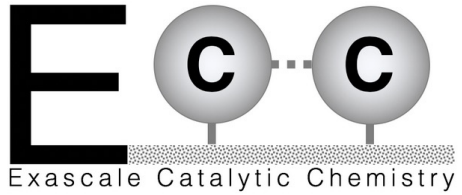
Oscar Diaz-Ibarra

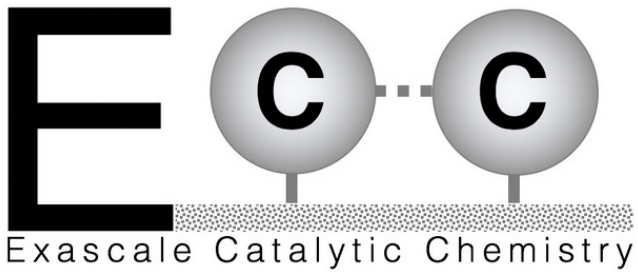
odiazib@sandia.gov

Kyungjoo Kim, Cosmin Safta, and Habib Najm

Sandia National Laboratories

18th International Conference on Numerical Combustion.
La Jolla, CA
05/08/2022 - 05/11/2022





https://ecc-project.sandia.gov/

Acknowledgement:
This work is supported as part of the Computational Chemical Sciences Program funded by the U.S. Department of Energy, Office of Science, Basic Energy Sciences, Chemical Sciences, Geosciences and Biosciences Division.

Award number: 0000232253

Motivation

- The computational singular perturbation (CSP) method was first introduced by Lam and Goussis (1988).
- This method is useful for dynamical systems with wide range of time scales.
- It has been applied in gas phase kinetic modeling, as well as biochemical modeling for diagnostics and model simplification.
- CSP analysis for gas-phase and surface chemistry models has been implemented in open-source code, CSPlib: https://github.com/sandialabs/CSPlib.

Computational Singular Perturbation (CSP) Analysis

- CSP basis vectors decouple the fast and slow subspaces.
- Eigenvectors of the Jacobian, i.e. $J_{ij} = \frac{\partial g_i(\boldsymbol{y})}{\partial y_j}$, provide first order decoupling of the fast and slow subspaces.
- For chemical systems, CSP indices measure contribution of reactions in the slow/fast subspace (Importance indices) and mode amplitudes (Participation index).

ODE:

$$\frac{d\boldsymbol{y}}{dt} = \boldsymbol{g}(\boldsymbol{y}) \qquad \boldsymbol{y} \in \mathbb{R}^N \quad \boldsymbol{y}(t=0) = \boldsymbol{y}_0$$

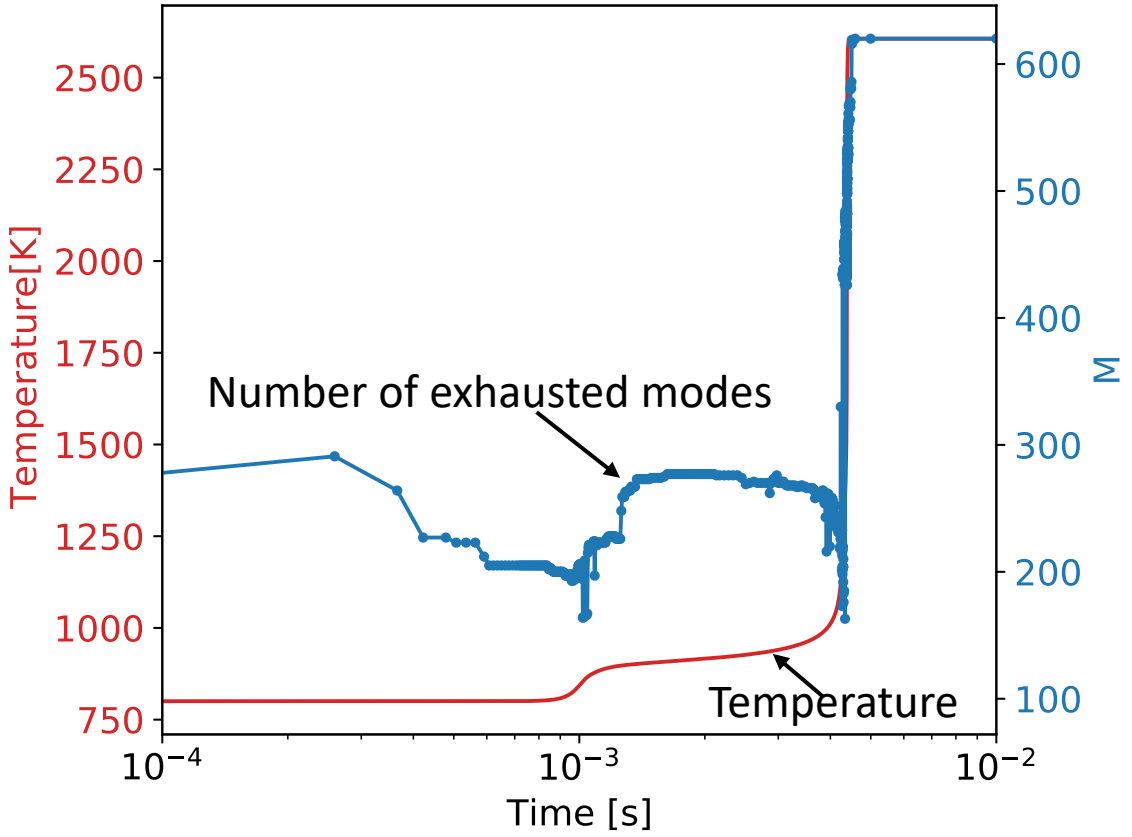
Expand RHS in CSP basis:

$$\boldsymbol{g} = \boldsymbol{a}_1 f^1 + \cdots + \boldsymbol{a}_N f^N$$

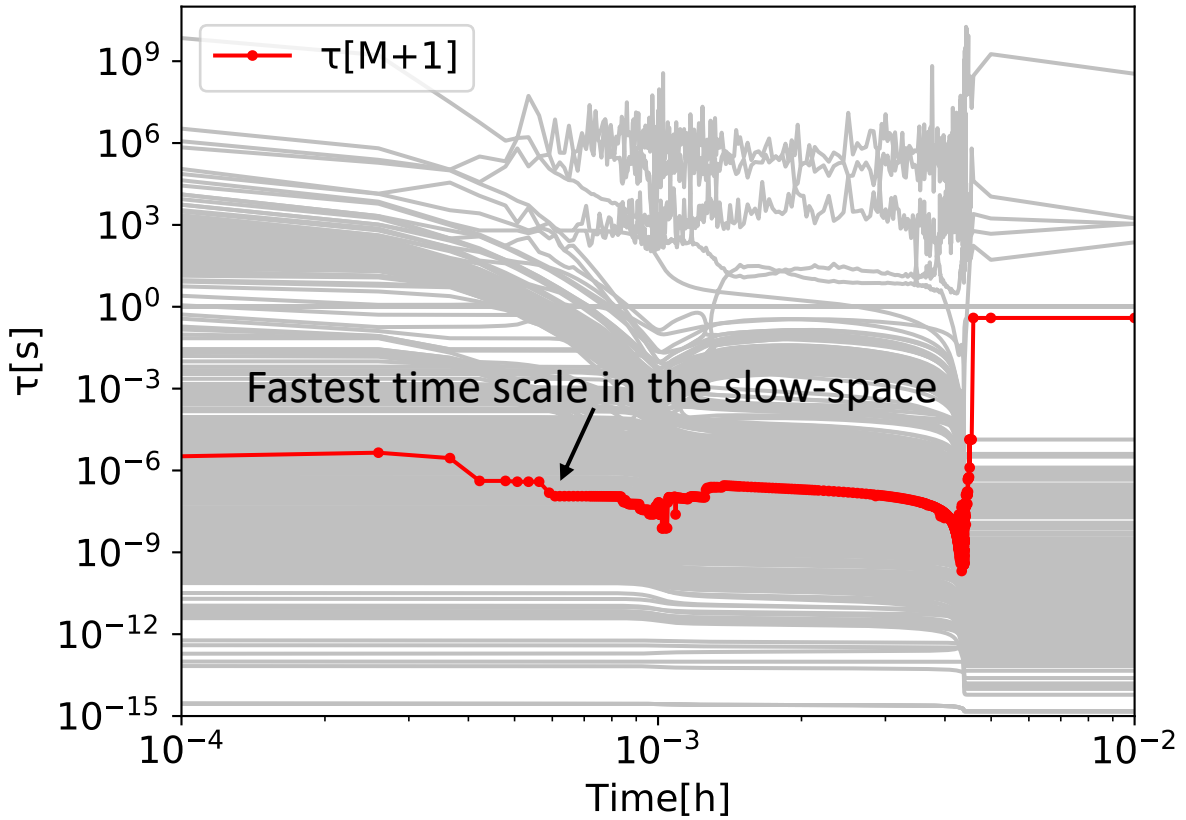
$$\boldsymbol{g} = \underbrace{\sum_{i=1}^{M} \boldsymbol{a}_i f^i}_{g_{\text{fast}} \approx 0} + \underbrace{\sum_{i=M+1}^{N} \boldsymbol{a}_i f^i}_{g_{\text{slow}}}$$

D. Goussis, H. Lam (1988--), M. Valorani ('90s--). Comb. Sci. Tech.; Comb. Flame; Comb. Theo. Mod.

CSP Helps Uncover the Dynamics of Complex Kinetic Models



Number of inactive models during the ignition for n-heptane.



Kinetic models exhibit a wide range of time scales.

https://combustion.llnl.gov/mechanisms/alkanes/n-heptane-detailed-mechanism-version-3
Number of species : 654 Number of reactions: 2827

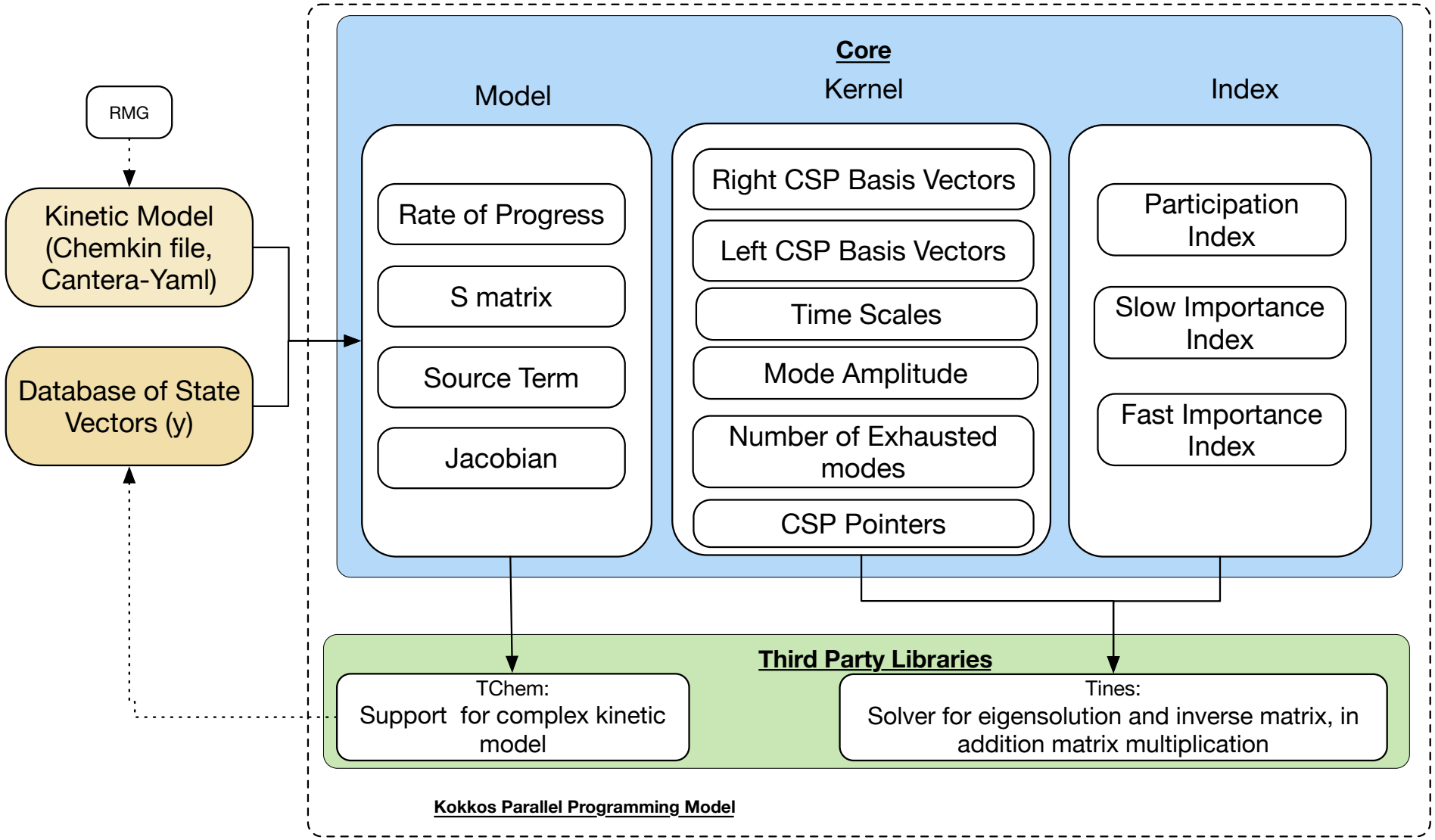
We Extend the CSP Method to Handle DAE Systems.

- CSP analysis for DAE/ODE of catalysis systems has been implemented in the open-source code, CSPlib: https://github.com/sandialabs/CSPlib.
- In CSPlib, computations can be executed on modern computing platforms, i.e, GPUs and many-core CPUs.

Oscar Diaz-Ibarra, Kyungjoo Kim, Cosmin Safta, Judit Zador & Habib N. Najm (2021) Using computational singular perturbation as a diagnostic tool in ODE and DAE systems: a case study in heterogeneous catalysis, Combustion Theory and Modelling, DOI: 10.1080/13647830.2021.2002417

Schematic of CSPlib

github.com/sandialabs/csplib



CSP Tasks in the Kernel/Index Class Are Linear Algebra Operations, E.G., Eigensolver, Matrix Inversion, Matrix-Matrix Multiplication.

CSPlib task	Type of operation	Library
Right CSP Basis Vectors	Right eigenvectors of Jacobian	Tines/LAPACK
Left CSP Basis Vectors	Matrix inverse of right eigenvectors	Tines/LAPACK
Time Scales	Multiplicative inverse of eigenvalues	CSPlib
Mode Amplitude	Dense matrix-vector multiplication	Tines
Number of Exhausted Modes	Iterative search	CSPlib
CSP Pointer	Element-wise matrix multiplication	CSPlib
Participation Index	Dense matrix-matrix multiplication	CSPlib/Tines
Slow Importance Index	Dense matrix-matrix multiplication	CSPlib/Tines
Fast Importance Index	Dense matrix-matrix multiplication	CSPlib/Tines

https://github.com/sandialabs/TChem

Rate of progress, S matrix, Source term, and Jacobian (TChem routines) are non-linear operations.

Tines

Time Integration, Newton and Eigensolver

https://github.com/sandialabs/Tines

CSPlib uses Kokkos as parallel programing model.

"Kokkos Core implements a programming model in C++ for writing performance portable applications targeting all major HPC platforms."

https://github.com/kokkos/kokkos

To effectively use modern computer platforms, CSPlib computes many CSP analyses in parallel, i.e., batched computation.

```
Kokkos::TeamPolicy<exec_space> policy(nBatch, Kokkos::AUTO());
Kokkos::parallel_for("UserCode", policy,
  KOKKOS_LAMBDA(const member_type& member) {
  const ordinal_type i = member.league_rank();
  // do something with ith work item
  // ...
  // compute model/kernel/index task
  compute_CSPlib_task(member, t, p, Y, ... );
});
```

Kokkos offers:

- Parallel computing patterns
- Data management (View)
- Backend: CUDA, HIP, SYCL, HPX, OpenMP and C++ threads

A single version of CSPlib is used for both CPU and GPU, mapping the code to OpenMP and CUDA backends respectively.

Case Study: Testbed Machine Specification

Processor	IBM Power9 2x20@3.6GHz	NVIDIA V100 80 SM@1.5GHz
# Threads	4 SMT/core	max 2048/SM
Cache	120 MB	6 MB L2
Memory	320 GB	16 GB
Compiler	GNU 7.2	NVCC 10.1
Exec. Space	OpenMP	CUDA

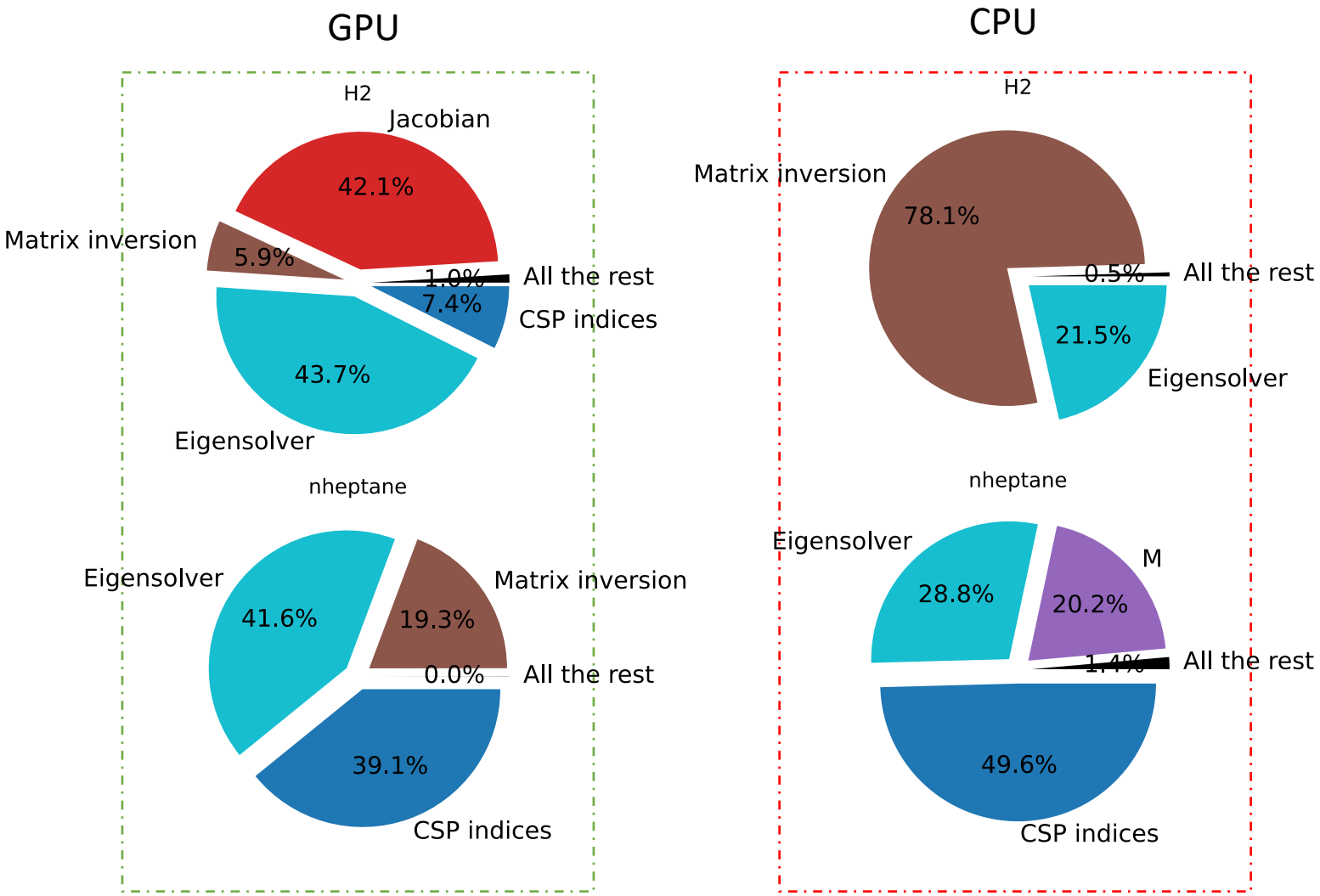
CSPlib copies database from host to device when the CSP analysis starts.

Case Study: Chemical Kinetic Models

Model	# Species	# Reactions	Reference
H_2	8	19	Yetter et al. (1991)
CO	14	33	Ranzi et al. (2012)
GRIMech 3.0	53	325	GRI-Mech v3.0 (2011)
n-dodecane	106	420	Luo et al. (2014)
n-heptane	654	2827	Mehl et al. (2011)

- Constant pressure ignition reactor configuration (from TChem)
- Batch parallel computation (vary the number of input states)

Eigensolver is the most expensive task in the CSP analysis, as expected.



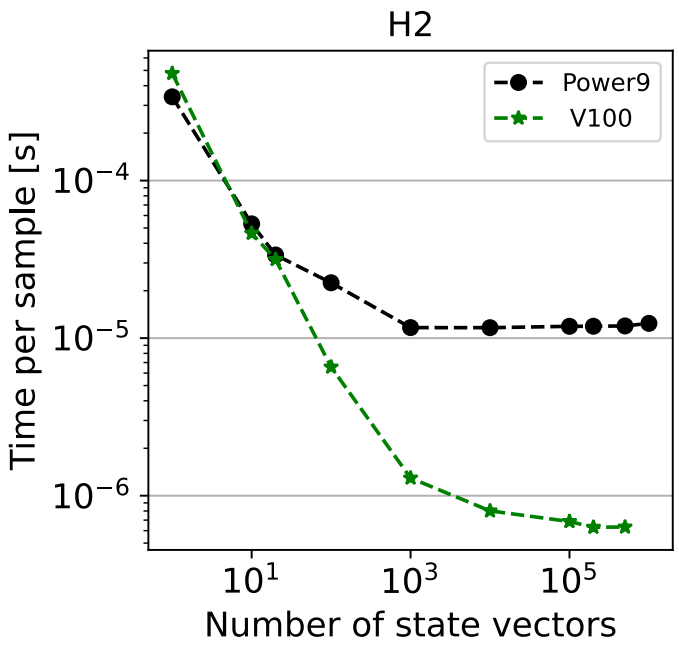
"All the rest": tasks with less than 0.5 % w.r.t. total time.

- Kinetics models size affects the performance of CSPlib in GPUs/CPUs.
- The computation of CSP indices is significant for large kinetic models because of the number of species and reactions.
  - H2 : 500, 000 state vectors
  - n-heptane : 100 state vectors
  - Read/write files is not included.

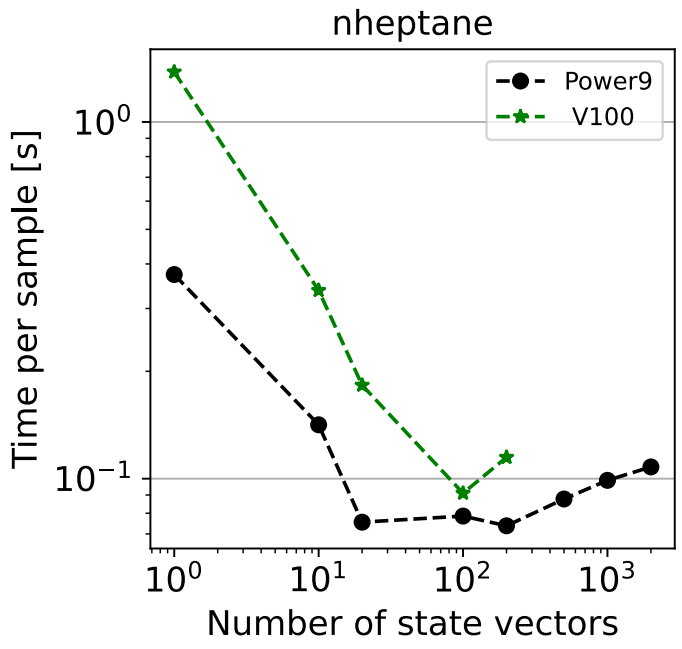
https://github.com/kokkos/kokkos-tools/tree/master/profiling/simple-kernel-timer-json

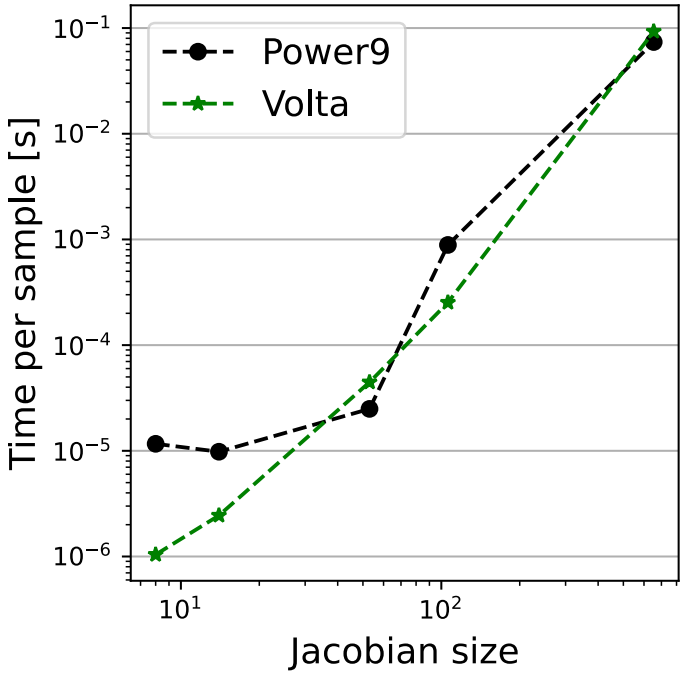
GPU Eigensolver Is Faster Than the LAPACK Routine for Small Kinetic Models.

H2: 9 variables



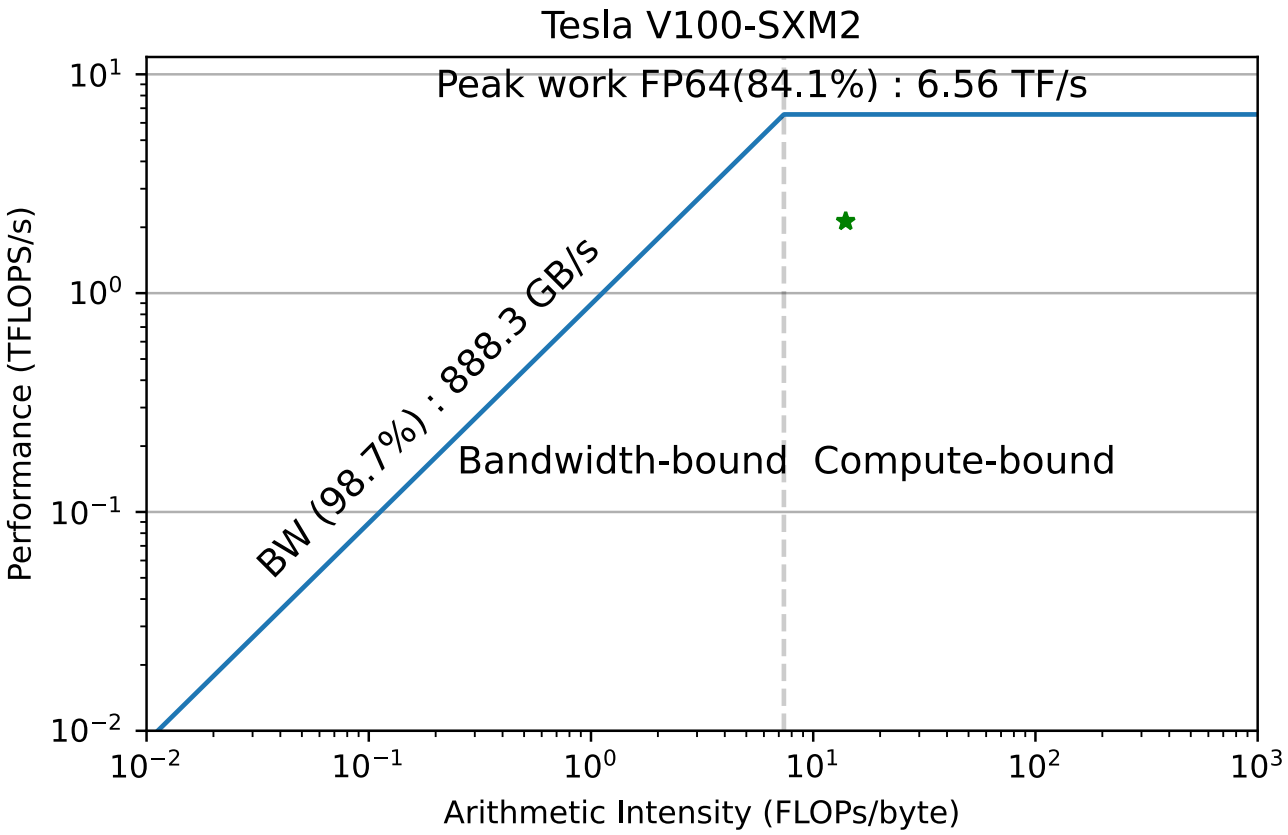
N-heptane: 655 variables





Due to limitations of GPU memory, the number of state vectors for large kinetic model cannot be increased.

How Does CSPlib Compare with Peak Performances in GPUs?



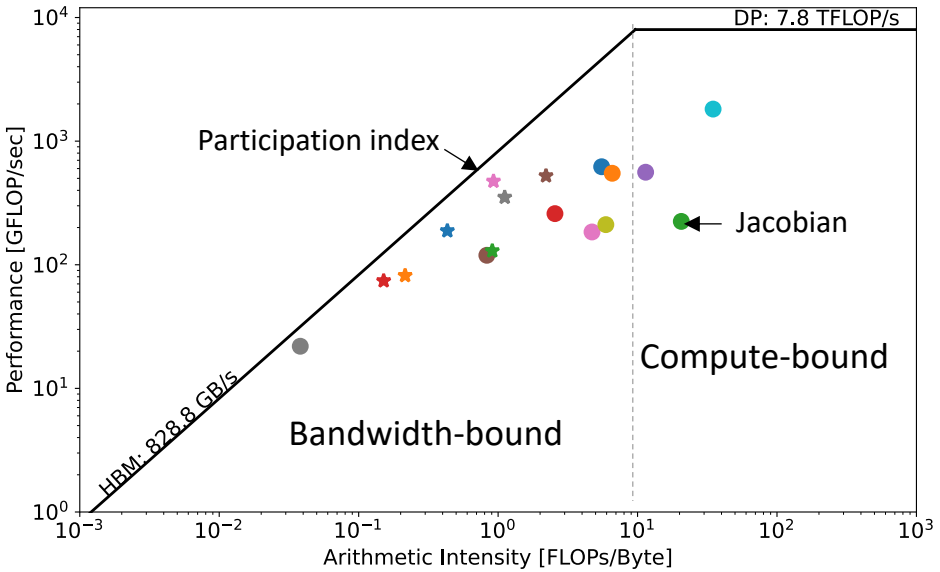
- Arithmetic Intensity (AI): ratio of total floating-point operations (FLOPs) to the total data movement (Bytes).
- Performance FLOPs/s : computational throughput.

- To compare application performance vs machine capabilities.
- To track code progress.
- Information: run time, the total number of FLOPs performed and bytes moved.
- Profiling data is produced by Nsight Compute (NVIDIA) to produce roofline charts.

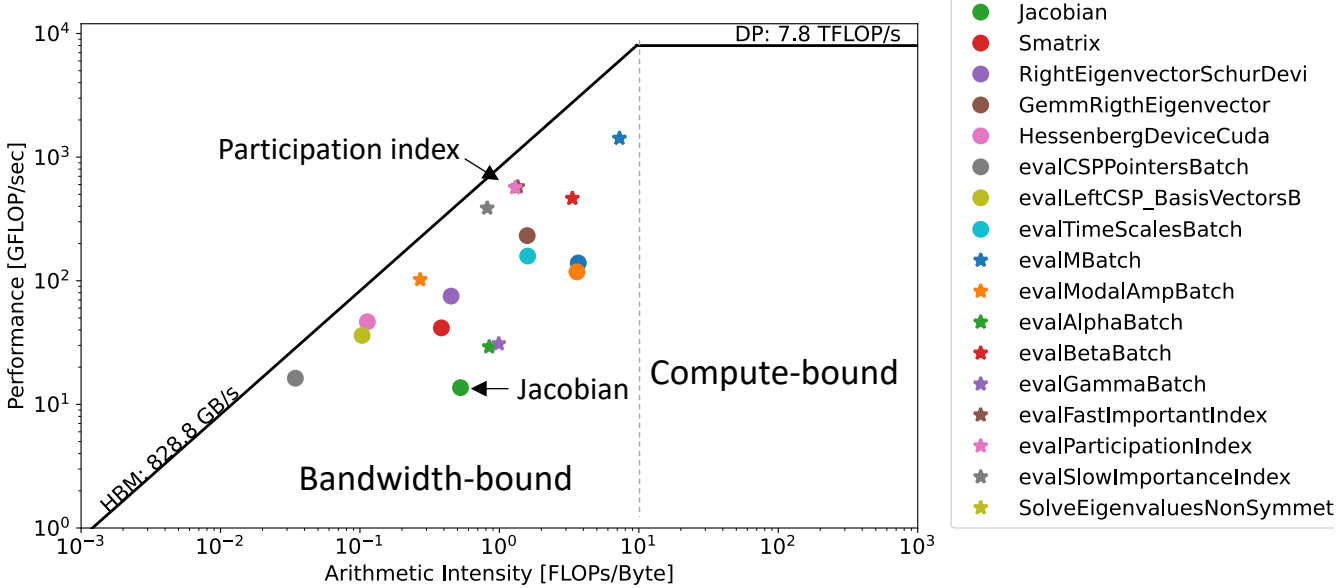
Documentation:
https://docs.nersc.gov/tools/performance/roofline/

https://gitlab.com/NERSC/roofline-on-nvidia-gpus

According to Roofline Model, Most of The CSP Tasks Are Considered to Be Bandwidth-bound.



H2: 9 variables/19 reactions
Batch size : 500,000 samples



N-heptane: 655 variables/ 2827 reactions
Batch size : 100 samples

Many tasks are close to peak performance, e.g, Participation index.

Conclusions

- We developed an open-source code, CSPlib, and analyzed its performance on modern computing platforms using wall-clock time profiles and roofline charts.
- The most expensive tasks in CSPlib are: eigensolver, matrix inversion, and Jacobian evaluation; which are expected in the CSP analysis.
- According to Roofline model, most of the CSP tasks are considered to be bandwidth-bound.
- GPU implementations of CSPlib (TChem/Tines) are faster that the CPU implementation for small kinetic models (H2, CO, GRI3).