JÜLICH Forschungszentrum

pySDC: Prototyping Spectral Deferred Corrections

Thomas Baumann[1,2], Thibaut Lunet[2], Daniel Ruprecht[2], Robert Speck[1], Lisa Wimmer[3]

Spectral Deferred Corrections (SDC)

Initial value problem in Picard form:

$$u(t) = u(t_0) + \int_{t_0}^{t} f(u(\tau))\, d\tau$$

Discretize with spectral quadrature:

$$\mathbf{u} = \mathbf{u}_0 + \Delta t Q F(\mathbf{u})$$

Preconditioning

- Picard iteration:

$$\mathbf{u}^{k+1} = \mathbf{u}^{k} + \left(\mathbf{u}_0 - (I - \Delta t Q F)\left(\mathbf{u}^{k}\right)\right)$$

- Precondition with simpler (lower triangular) quadrature rule Q_Δ:

$$\left(I - \Delta t Q_\Delta F\right)\left(\mathbf{u}^{k+1}\right) = \mathbf{u}_0 + \Delta t \left(Q - Q_\Delta\right) F\left(\mathbf{u}^{k}\right)$$

- Popular preconditioners: Implicit Euler or LU-decomposition for stiff problems

Properties

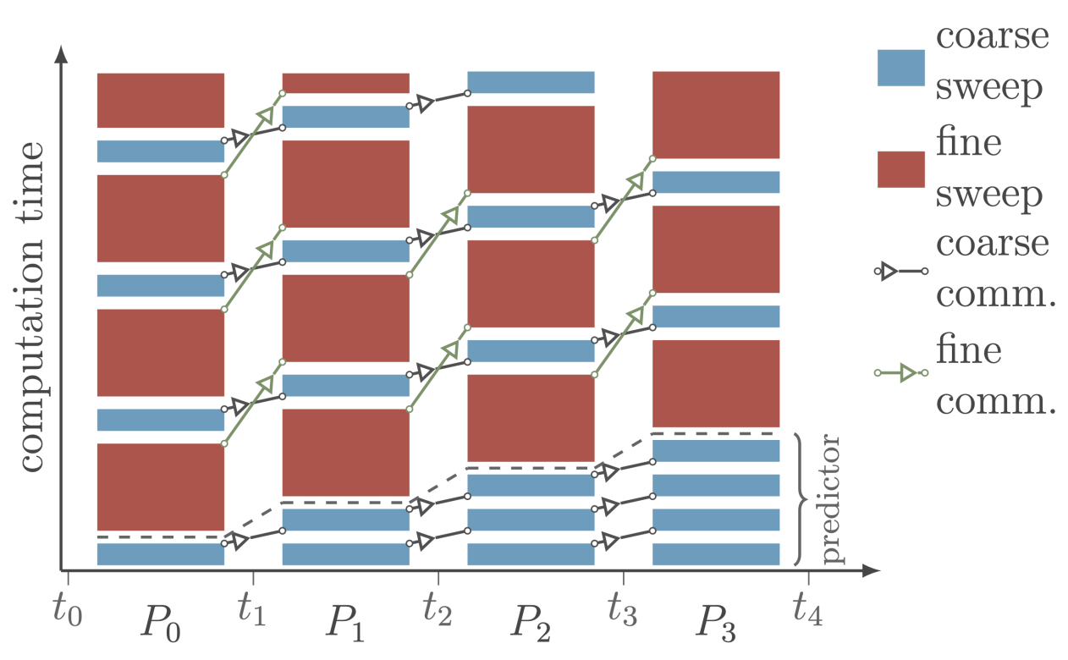
- Order can be equal to iteration count, depending on preconditioner
- Parallel-in-time extensions easy due to iterative nature
- Very malleable by choice of preconditioner(s): IMEX, Multi-implicit, Boris-SDC, …

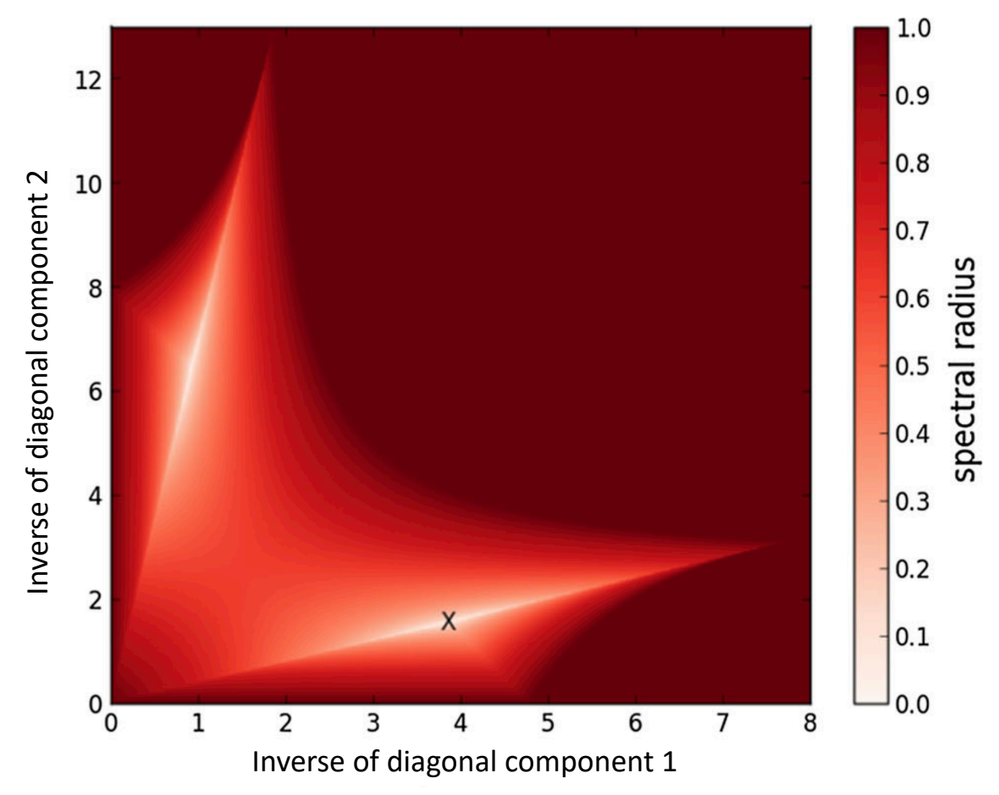
Parallel-in-Time

PFASST: SDC + Parareal + τ-Correction

- Assemble N steps into composite collocation problem
- Solve in parallel on fine grid
- Compute τ-correction on fine grid
- Solve serially on coarse grid, augmented by τ-correction
- Add coarse grid correction to fine solution

→ Space-time multigrid for the composite collocation problem





Diagonal SDC

- Diagonal preconditioner allows parallel update of collocation nodes in SDC iterations
- Options for generating diagonal preconditioners:
  - Diagonal elements of Q
  - Diagonal implicit Euler
  - Minimize spectral radius of SDC iteration matrix
  - …?

Why pySDC?

What is pySDC?

- Python implementation of various flavours of SDC, all the way to PFASST
- Implements only time stepping and leaves spatial part to NumPy, PETSc or FEniCS
- Intended for prototyping: Test algorithms before implementing them in production codes
- Actively developed and involved in many ongoing PinT projects

pySDC offers

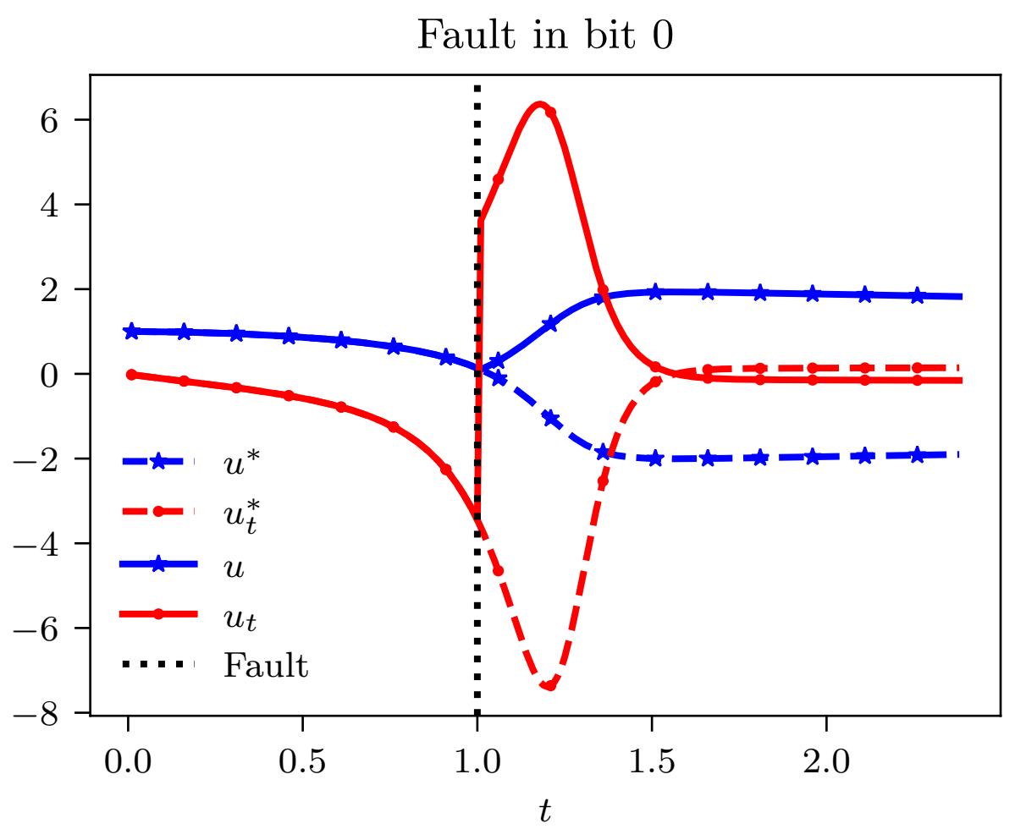
- Comprehensive tutorials from running examples to implementing new algorithms
- Many example problems: ND heat equation, Allen-Cahn, Van der Pol, Penning Trap, …
- Parallel algorithms available in MPI and pseudo-parallel implementations
- Well-documented and well-tested core library and projects
- Separation of concerns: Work on your method or problem without awareness of the entire code

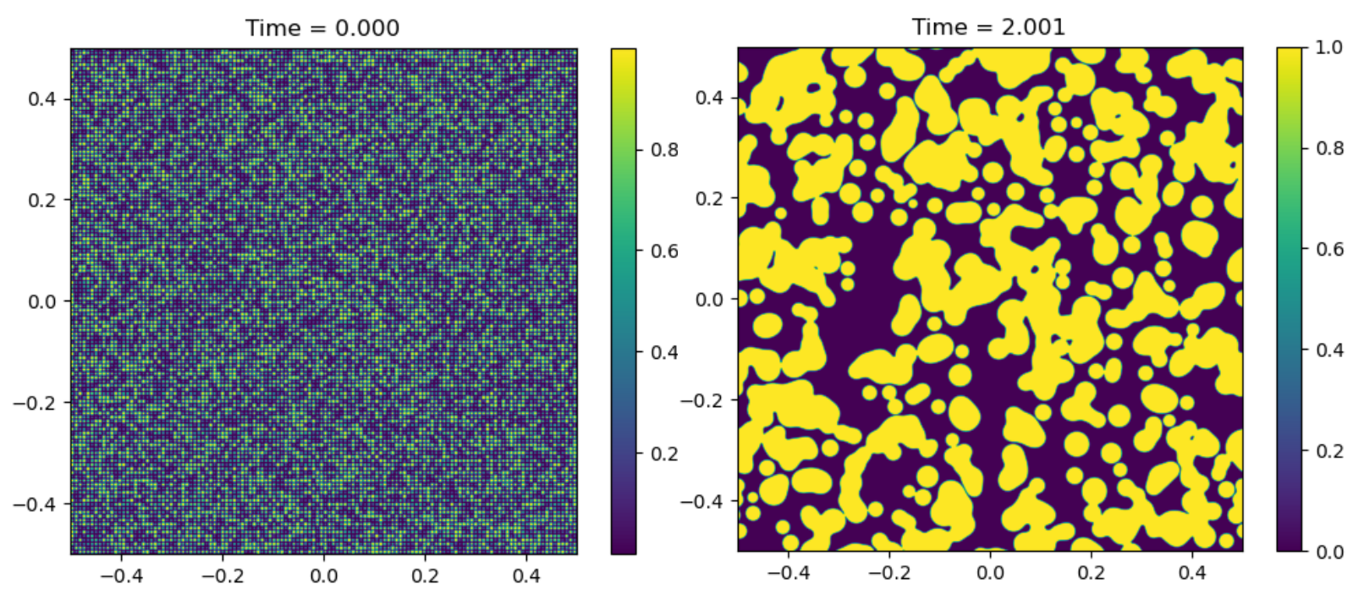
$\Longrightarrow$ Time-to-solution: 👎, but time-to-simulation: 👍

Ongoing Projects

Resilience and Adaptivity in SDC

- Transfer concepts from embedded Runge-Kutta to SDC
- Gain computational efficiency by adaptive resolution in time
- Gain resilience against soft faults by controlling the local error
- Also works in multi-step Block Gauß-Seidel SDC
- Image: Fault in sign bit sends van der Pol off its trajectory

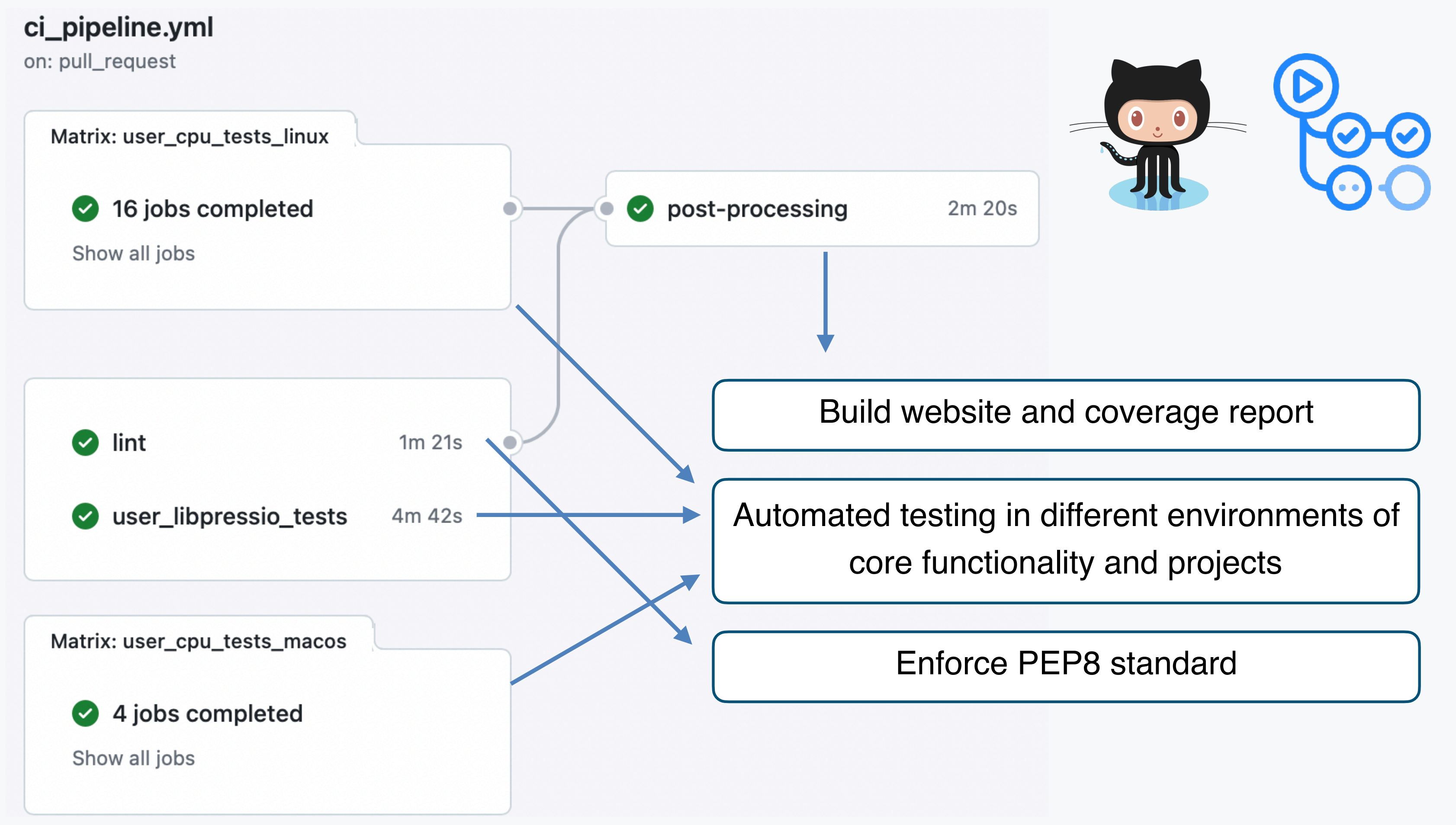




GPU Acceleration of pySDC

- Replace NumPy with CuPy for spatial solvers
- Measured speedup ≈ 100 on NVIDIA Tesla V100 vs. AMD EPYC 7742
- Enables solving very large problems
- So far only tested single GPU and time-serial
- Image: High resolution 2D Allen-Cahn problem

Continuous Integration



ci_pipeline.yml
on: pull_request

Matrix: user_cpu_tests_linux
✓ 16 jobs completed
Show all jobs

✓ post-processing 2m 20s

✓ lint 1m 21s
✓ user_libpressio_tests 4m 42s

Matrix: user_cpu_tests_macos
✓ 4 jobs completed
Show all jobs

Build website and coverage report

Automated testing in different environments of core functionality and projects

Enforce PEP8 standard

CI pipeline for pySDC passing | openssf best practices in progress 91% | codecov 73% | DOI 10.5281/zenodo.7766942

Become a Collaborator!

Test your SDC-related method with various available problems
or
test your problem with various available SDC-related methods!

SCAN ME

Ideas for projects

- Implement ParaDiag for single-level PinT using diagonalization
- Add multi-GPU support in space and space-time GPU capabilities
- Enhance PETSc and FEniCS interfaces and add more

Always open for Bachelor, Master, … theses!

Website: https://parallel-in-time.org/pySDC **GitHub:** https://github.com/Parallel-in-Time/pySDC

EuroHPC Joint Undertaking HIRSE_PS TIME-X SPONSORED BY THE Federal Ministry of Education and Research

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955701. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Belgium, France, Germany, and Switzerland. This project also received funding from the German Federal Ministry of Education and Research (BMBF) grant 16HPC048.

Contact: r.speck@fz-juelich.de, t.baumann@fz-juelich.de
[1]Jülich Supercomputing Centre, Forschungszentrum Jülich, Germany. [2]Hamburg University of Technology, Institute of Mathematics, Chair Computational Mathematics, Germany. [3]Bergische Universität Wuppertal, Germany

Member of the Helmholtz Association