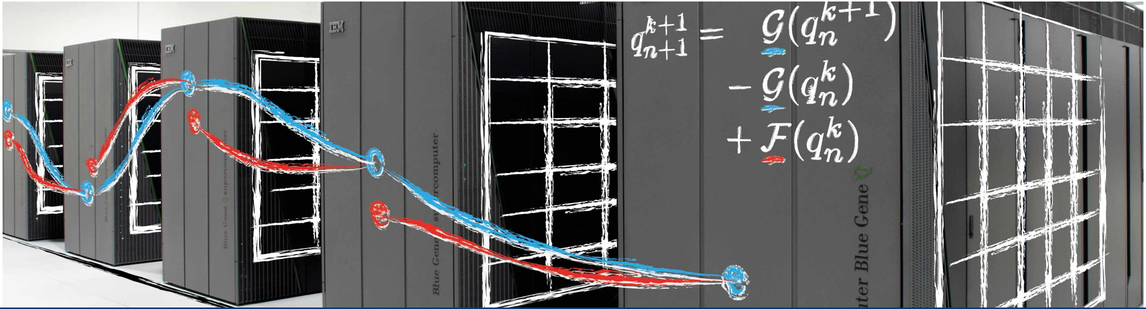


Parallelizing SDC Across the Method

November 1, 2022 | Thomas Baumann, Ruth Schöbel, Robert Speck | Jülich Supercomputing Centre

Outline

Are you asking yourself any of the following questions?

1. What is SDC?
2. How could we parallelize this across the method?
3. What have Robert and Ruth published about this already?
4. What are Ruth and me up to in this area?
5. Audience participation: What are you doing like this?

Then today is your lucky day!

Actually, it's a boatload of equations, so no. It's not your lucky day...

The Collocation Problem

Consider the Picard form of an initial value problem on $[T_0, T_1]$

$$u(t) = u_0 + \int_{T_0}^{t} f(u(s))ds,$$

discretized using spectral quadrature rules with nodes t_m:

$$u_m = u_0 + \Delta t \sum_{l=1}^{M} q_{m,l} f(u_l) \approx u_0 + \int_{T_0}^{t_m} f(u(s))ds,$$

$\Rightarrow$ corresponds to a fully implicit Runge-Kutta method on $[T_0, T_1]$.

The Collocation Problem

Consider the Picard form of an initial value problem on $[T_0, T_1]$

$$u(t) = u_0 + \int_{T_0}^{t} f(u(s))ds,$$

discretized using spectral quadrature rules with nodes t_m:

$$(I - \Delta t Q F)(\vec{u}) = \vec{u}_0$$

$\Rightarrow$ corresponds to a fully implicit Runge-Kutta method on $[T_0, T_1]$.

Q is typically dense, so solving this system directly is very expensive!

Solving the Collocation Problem

A few popular approaches...

1. Serial in time
   1. Diagonally implicit Runge-Kutta (DIRK): Larger, but lower triangular Q
   2. Explicit Runge-Kutta: Larger, but strictly lower triangular Q
   3. Spectral deferred corrections (SDC): Iterate with lower triangular preconditioner Q_Δ
2. Parallel across the method
   1. Diagonalize Q before solving: Parallel computation at the expense of extra work
   2. SDC with diagonal preconditioner: Parallel computation at the cost of possibly more iterations

Solving the Collocation Problem

A few popular approaches...

1. Serial in time
   1. Diagonally implicit Runge-Kutta (DIRK): Larger, but lower triangular Q
   2. Explicit Runge-Kutta: Larger, but strictly lower triangular Q
   3. Spectral deferred corrections (SDC): Iterate with lower triangular preconditioner Q_Δ

2. Parallel across the method
   1. Diagonalize Q before solving: Parallel computation at the expense of extra work
   2. SDC with diagonal preconditioner: Parallel computation at the cost of possibly more iterations

Spectral Deferred Corrections

- Standard Picard iteration is Richardson for $(I - \Delta t QF)(\vec{u}) = \vec{u}_0$, i.e.

$$\vec{u}^{k+1} = \vec{u}^k + \underbrace{\left(\vec{u}_0 - (I - \Delta t QF)(\vec{u}^k)\right)}_{\text{residual } \vec{r}^k}$$

- Preconditioning: use simpler (lower triangular) integration rule Q_Δ with

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

This corresponds to **spectral deferred corrections (SDC)**!

Spectral Deferred Corrections

- Standard Picard iteration is Richardson for $(I - \Delta t Q F)(\vec{u}) = \vec{u}_0$, i.e.

$$\vec{u}^{k+1} = \vec{u}^k + \underbrace{\left(\vec{u}_0 - (I - \Delta t Q F)(\vec{u}^k)\right)}_{\text{residual } \vec{r}^k}$$

- Preconditioning: use simpler (lower triangular) integration rule Q_Δ with

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

This corresponds to **spectral deferred corrections (SDC)**!

Spectral Deferred Corrections: Role of the Preconditioner

- Solve defect equation using Q_Δ:

$$\vec{\delta}^{k+1} - \Delta t Q_\Delta F(\vec{u}^k + \vec{\delta}^{k+1}) = \vec{r}^k - \Delta t Q_\Delta F(\vec{u}^k)$$

- On right hand side: residual $\vec{r}^k$ computed with full Q:

$$\vec{r}^k = \vec{u}_0 + \Delta t Q F(\vec{u}^k) - \vec{u}^k$$

- Refine the solution with defect:

$$\vec{u}^{k+1} = \vec{u}^k + \vec{\delta}^{k+1} = \vec{u}^k + \underbrace{\vec{u}_0 + \Delta t Q F(\vec{u}^k) - \vec{u}^k}_{\vec{r}^k} + \Delta t Q_\Delta (F(\underbrace{\vec{u}^k + \vec{\delta}^{k+1}}_{\vec{u}^{k+1}}) - F(\vec{u}^k))$$

- Simplifies to the familiar:

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

SDC with Diagonal Preconditioner

Diagonalize existing preconditioners

Want to solve: $(I - \Delta t Q_\Delta F)(\vec{u}^{k+1}) = rhs$

- For linear problems: $Q_\Delta F = Q_\Delta \otimes A$, $Q_\Delta \in \mathbb{R}^{M\times M}$, $A \in \mathbb{R}^{N\times N}$
- Diagonalize:

$$Q_\Delta \otimes A = (V_{Q_\Delta} \otimes I_N)(I_M \otimes I_N - \Delta t \Lambda_{Q_\Delta} \otimes A)(V_{Q_\Delta} \otimes I_N)^{-1}$$

- Multiply by $(V_{Q_\Delta} \otimes I_N)^{-1}$ to get

$$\underbrace{(I_M \otimes I_N - \Delta t \Lambda_{Q_\Delta} \otimes A)}_{\text{block diagonal}} \tilde{\vec{u}}^{k+1} = \tilde{rhs}$$

- Solve and multiply by $(V_{Q_\Delta} \otimes I_N)$ to obtain $\vec{u}^{k+1}$
- $(V_{Q_\Delta} \otimes I_N)$ is dense $\implies$ all-to-all communication $\implies$ best for shared memory parallelization

SDC with Diagonal Preconditioner

Quasi-Newton scheme for non-linear problems

- Define:

$$G = (I - \Delta t Q_\Delta F)(\vec{u}^k) - rhs$$

- Build Jacobian of G:

$$J_G = I - \Delta t Q_\Delta J_F(\vec{u}^k),$$

with

$$J_F(\vec{u}^k) = \mathrm{diag}(f'(u_1),\ldots,f'(u_M)) \in \mathbb{R}^{MN\times MN}$$

- Newton iteration:

$$J_G(\vec{u}^k)\vec{e}^j = -G(\vec{u}^k), \quad \vec{u}^{k+1} = \vec{u}^k + \vec{e}^j$$

- Diagonalize Q_Δ:

$$((V_{Q_\Delta} \otimes I)^{-1} - \Delta t(\Lambda_{Q_\Delta} \otimes I_N)\underbrace{(V_{Q_\Delta} \otimes I_N)J_F(\vec{u}^k)}_{\text{dense}})\vec{e}^j = -\tilde{G}(\vec{u}^k)$$

SDC with Diagonal Preconditioner

Quasi-Newton scheme for non-linear problems

- Define:

$$G = (I - \Delta t Q_\Delta F)(\vec{u}^k) - rhs$$

- Build approximate Jacobian of G:

$$J_G = I - \Delta t Q_\Delta J_F(\vec{u}_0),$$

with

$$J_F(\vec{u}_0) = \mathrm{diag}(f'(u_0),\ldots,f'(u_0)) = I_M \otimes f'(u_0)$$

- Quasi Newton iteration:

$$J_G(\vec{u}_0)\vec{e}^j = -G(\vec{u}^k), \quad \vec{u}^{k+1} = \vec{u}^k + \vec{e}^j$$

- Diagonalize Q_Δ:

$$\underbrace{\left(1_M \otimes 1_N - \Delta t \Lambda_{Q_\Delta} \otimes f'(u_0)\right)}_{\text{block diagonal}} \tilde{\vec{e}}^j = -\tilde{G}(\vec{u}^k)$$

- Regular Newton converges quadratically, but quasi-Newton only linearly!

Diagonalize the Quadrature Matrix

Both a Runge-Kutta method and an SDC method

For suitable choices of the M collocation nodes, Q can be diagonalized, i.e. for linear problems

$$(I - \Delta t Q F)(\vec{u}) = (I - \Delta t Q \otimes A)\vec{u} = (V_Q \otimes I)(I - \Delta t \Lambda_Q \otimes A)(V_Q \otimes I)^{-1}\vec{u}$$

Remarks:

- Equivalent to diagonalizing Q_Δ if $Q_\Delta = Q$
- This is a direct solver for linear problems
- Extension to nonlinear problems via inexact Newton
- Classical approach to deal with fully-implicit RK methods
- Beware: Λ_Q has complex entries!

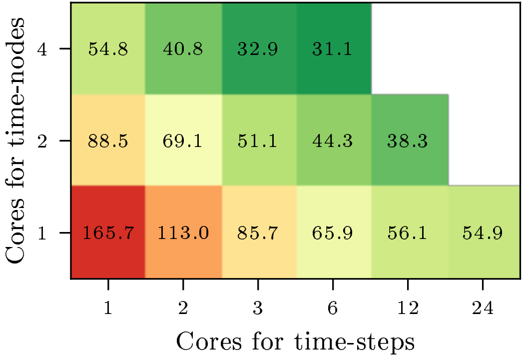
XXXtra Parallel: Combine PFASST with Parallel SDC

Ruth and Robert in "PFASST-ER: Combining the Parallel Full Approximation Scheme in Space and Time with parallelization across the method" (2020)

Idea: Use parallel SDC sweeps within parallel time-steps.

Example: 2D Allen-Cahn, fully-implicit, 256x256 DOFs in space, up to 24 available cores.

Best parallel efficiency: Saturate node-parallelism before doing step-parallelism.



SDC with Diagonal Preconditioner

What if the preconditioner was diagonal to begin with?

Want to solve: $(I - \Delta t Q_\Delta F)(\vec{u}^{k+1}) = rhs$

- Q_Δ is already diagonal: $Q_\Delta = \Lambda$
- We get $(I - \Delta t \Lambda F)(\vec{u}^{k+1}) = rhs$
- This decouples to $(1 - \Delta t \lambda_i f)(u_i^{k+1}) = rhs_i$
- Parallel sweeps with standard Newton scheme for non-linear problems!

JÜLICH Forschungszentrum

Design Diagonal Preconditioners

Robert in “Parallelizing spectral deferred corrections across the method” (2018)

1. Diagonal elements of the full quadrature matrix
2. Diagonal implicit Euler
3. Minimize the spectral radius

Design Diagonal Preconditioners

Robert in “Parallelizing spectral deferred corrections across the method” (2018)

1. Diagonal elements of the full quadrature matrix

$$Q_\Delta^{Q_{\text{par}}} = \text{diag}(q_{ii}),$$

with q_{ii} the diagonal elements of Q.
Implemented as “Qpar” in pySDC.

2. Diagonal implicit Euler
3. Minimize the spectral radius

Design Diagonal Preconditioners

Robert in “Parallelizing spectral deferred corrections across the method” (2018)

1. Diagonal elements of the full quadrature matrix
2. Diagonal implicit Euler

$$Q_\Delta^{\mathsf{IEpar}} = \mathrm{diag}(\tau_i),$$

with τ_i the nodes of the quadrature rule.
Implemented as “IEpar” in pySDC.
Note: Standart implicit Euler integrates from node to node:

$$q_{\Delta ij}^{\mathsf{IE}} = \begin{cases} \tau_j - \tau_{j-1}, & 1 < j \leq i \\ \tau_j, & j = 1 \\ 0, & \text{otherwise} \end{cases}$$

3. Minimize the spectral radius

Design Diagonal Preconditioners

Robert in “Parallelizing spectral deferred corrections across the method” (2018)

1. Diagonal elements of the full quadrature matrix
2. Diagonal implicit Euler
3. Minimize the spectral radius
   Implemented as “MIN” in pySDC.
   - Dahlquist equation: $u_t = \lambda u$ (linear ODE)
   - SDC iteration matrix:

$$K = \lambda \Delta t Q_\Delta (I - \lambda \Delta t Q_\Delta)^{-1} \left(Q_\Delta^{-1} Q - I\right) \;\rightarrow\; \vec{u}^{k+1} = K \vec{u}^k$$

   - Stiff limit:

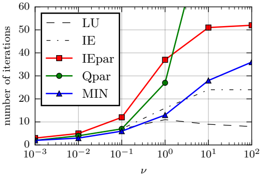
$$|\lambda \Delta t| \rightarrow \infty\text{: } K \rightarrow I - Q_\Delta^{-1} Q := K_\infty$$

     independent of λ
   - Minimize spectral radius $\rho(K_\infty)$ by choice of diagonal Q_Δ using SCIPY

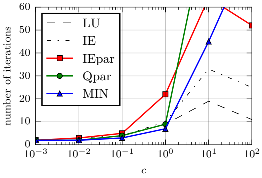
Design Diagonal Preconditioners

Robert in “Parallelizing spectral deferred corrections across the method” (2018)

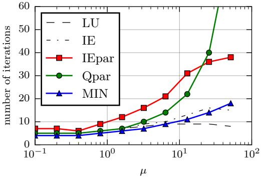
- For small parameters i.e. non-stiff problems, all approaches work as well as popular LU and IE preconditioners
- For stiff problems, only MIN works sometimes
- Same iteration count means lower execution time because of parallelism



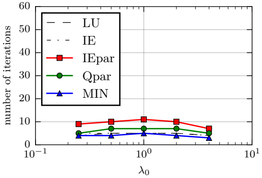
(a) Heat equation



(b) Advection



(c) Van der Pol



(d) Non-linear diffusion

JÜLICH Forschungszentrum

Find New Diagonal Preconditioners for SDC

Brought to you by Ruth and the AI gang

Go from minimizing $\rho(K_\infty)$ to minimizing range of $\rho(K_\lambda)$

- Optimize preconditioner for Dahlquist problem with specific λ
- Precompute a range of preconditioners for various λ
- Use FFT for space-discretization to obtain a system of Dahlquist problems[1]
- Solve each Dahlquist problem with its optimal preconditioner
- Option: Use reinforcement learning

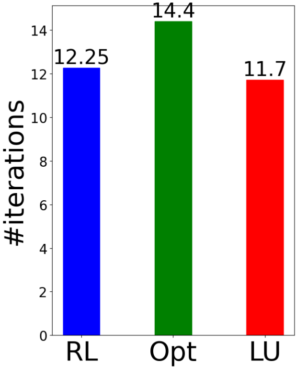
[1] Derivatives are multiplications in Fourier space

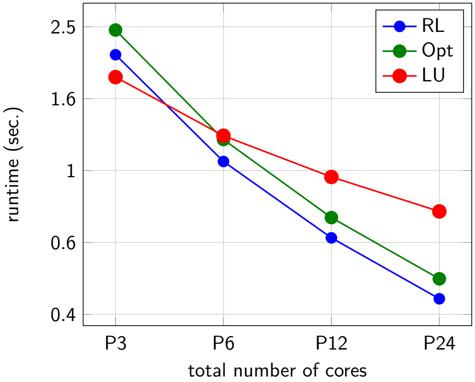
Find New Diagonal Preconditioners for SDC with AI

Ruth and the AI gang

Example:

Schrödinger equation: $u_t = (\Delta u - 6u|u|^2)i$ on $[0,2\pi]^3$, space-parallel LU-based SDC vs. parallel SDC





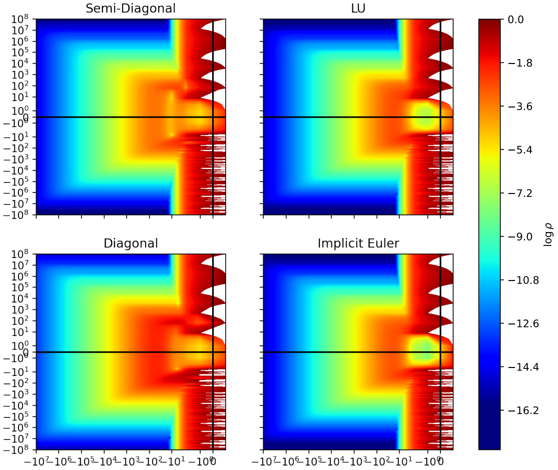
Generate Diagonal Preconditioners Using Adaptivity

Optimization problem:

- Adaptivity controls the step size
- Solve a reference problem over fixed interval in time
- Count iterations and minimize with diagonal elements as input

Q_Δ does not need to be diagonal!

- In pySDC: First column of Q_Δ corresponds to initial conditions
- Initial conditions are known on all ranks
- Can do parallel midpoint method for instance



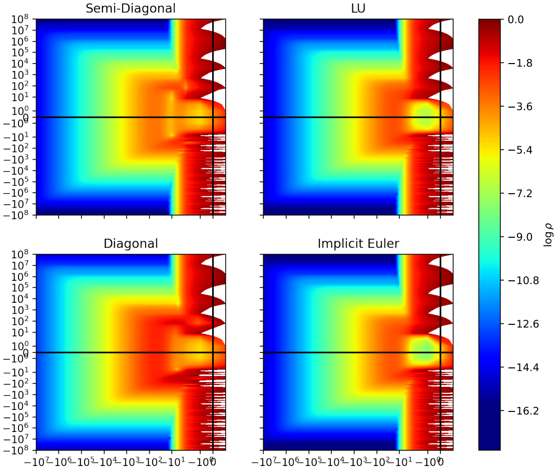
Generate Diagonal Preconditioners Using Adaptivity

Optimization problem:

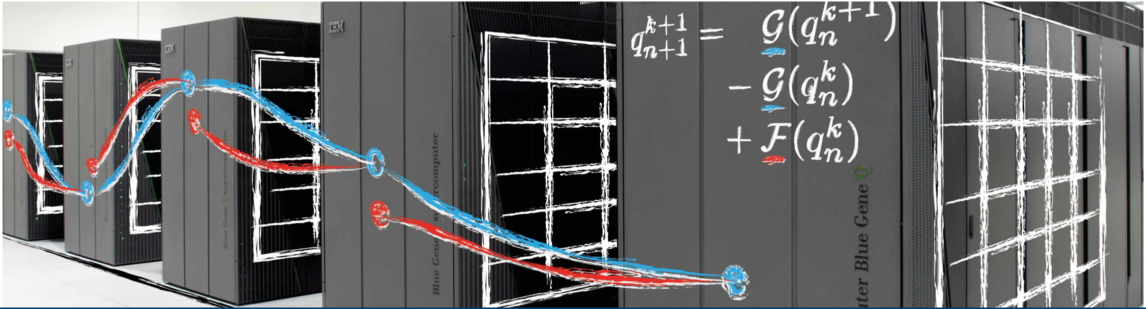
- Adaptivity controls the step size
- Solve a reference problem over fixed interval in time
- Count iterations and minimize with diagonal elements as input

Q_Δ does not need to be diagonal!

- In pySDC: First column of Q_Δ corresponds to initial conditions
- Initial conditions are known on all ranks
- Can do parallel midpoint method for instance




I WANT YOUR
PRECONDITIONER
...so I can share it with corporations
and then store it in plain text
where anyone can steal it.



Parallelizing SDC Across the Method

November 1, 2022 | Thomas Baumann, Ruth Schöbel, Robert Speck | Jülich Supercomputing Centre