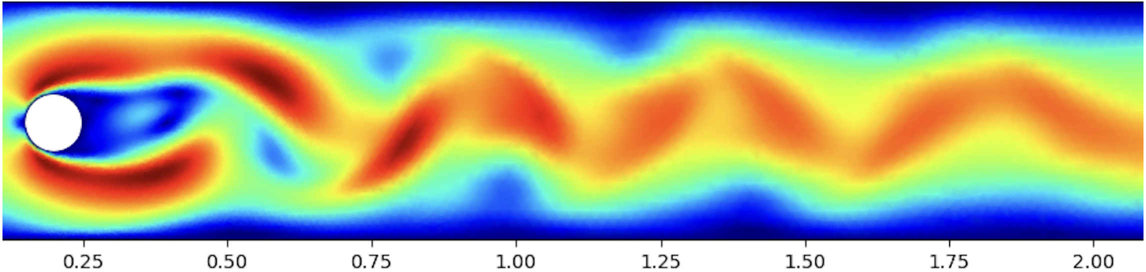


Spectral Deferred Corrections, Bacon-flavored

December 17, 2024 | Robert Speck & Thomas Baumann | Jülich Supercomputing Centre

Collocation Methods

Want to solve initial value problem in integral form:

$$[u_t(t) = f(t,u),\ u(t_0) = u_0] \iff \left[u(t) = u_0 + \int_{t_0}^{t} f(s,u)ds\right]$$

Discretize integral with quadrature rule:

- Discretize $[t_0, t_0 + \Delta t]$ at M quadrature nodes τ_m: $t_0 \leq \tau_m \leq t_0 + \Delta t$
- Approximate f by polynomial interpolation:

$$f(t,u) \approx \sum_{j=1}^{M} f(\tau_j, u(\tau_j))\, l_j^{\tau}(t)$$

  using Lagrange polynomials

$$l_j^{\tau}(t) = \frac{\prod_{k=1,k\neq j}^{M}(t-\tau_k)}{\prod_{k=1,k\neq j}^{M}(\tau_j-\tau_k)} \text{ with } l_j^{\tau}(\tau_k) = \begin{cases} 1, & j = k, \\ 0, & \text{otherwise} \end{cases}$$

Collocation Methods

Want to solve initial value problem in integral form:

$$\left[u_t(t) = f(t,u),\ u(t_0) = u_0\right] \iff \left[u(t) = u_0 + \int_{t_0}^{t} f(s,u)ds\right]$$

Discretize integral with quadrature rule:

- Discretize $[t_0, t_0 + \Delta t]$ at M quadrature nodes τ_m: $t_0 \leq \tau_m \leq t_0 + \Delta t$
- Approximate f by polynomial interpolation:

$$f(t,u) \approx \sum_{j=1}^{M} f(\tau_j, u(\tau_j))\, l_j^{\tau}(t)$$

using Lagrange polynomials

$$l_j^{\tau}(t) = \frac{\prod_{k=1,k\neq j}^{M}(t-\tau_k)}{\prod_{k=1,k\neq j}^{M}(\tau_j-\tau_k)} \text{ with } l_j^{\tau}(\tau_k) = \begin{cases} 1, & j = k, \\ 0, & \text{otherwise} \end{cases}$$

Collocation Methods

Want to solve initial value problem in integral form:

$$\left[u_t(t) = f(t,u),\ u(t_0) = u_0\right] \iff \left[u(t) = u_0 + \int_{t_0}^{t} f(s,u)ds\right]$$

Discretize integral with quadrature rule:

- Discretize $[t_0, t_0 + \Delta t]$ at M quadrature nodes τ_m: $t_0 \leq \tau_m \leq t_0 + \Delta t$
- Approximate f by polynomial interpolation:

$$f(t,u) \approx \sum_{j=1}^{M} f(\tau_j, u(\tau_j))\, l_j^{\tau}(t)$$

using Lagrange polynomials

$$l_j^{\tau}(t) = \frac{\prod_{k=1,k\neq j}^{M}(t - \tau_k)}{\prod_{k=1,k\neq j}^{M}(\tau_j - \tau_k)} \text{ with } l_j^{\tau}(\tau_k) = \begin{cases} 1, & j = k, \\ 0, & \text{otherwise} \end{cases}$$

Collocation Methods Continued

- Recall polynomial approximation: $f(t,u) \approx \sum_{j=1}^{M} f(\tau_j, u(\tau_j))\, l_j^{\tau}(t)$
- Plug into continuous equation:

$$u(\tau_m) = u_0 + \int_{t_0}^{\tau_m} f(s,u)ds \approx u_0 + \int_{t_0}^{\tau_m} \sum_{j=1}^{M} f(\tau_j, u(\tau_j))\, l_j^{\tau}(s)ds \tag{1}$$

$$= u_0 + \sum_{j=1}^{M} f(\tau_j, u(\tau_j)) \int_{t_0}^{\tau_m} l_j^{\tau}(s)ds = u_0 + \sum_{j=1}^{M} q_{m,j} f(\tau_j, u(\tau_j)) \tag{2}$$

- Use quadrature rule Q from integrating Lagrange polynomials to approximate the integral!

Collocation Methods Continued

Use vector notation and rescale quadrature nodes from 0 to 1:

$$(\vec{u})_m = u_m \approx u(\tau_m),\ (\vec{\tau})_m = \tau_m,\ (\vec{u}_0)_m = u_0,\ (Q)_{m,j} = q_{m,j},\ (f(\vec{u}))_m = f(\tau_m, u_m)$$

$$\vec{u} = \vec{u}_0 + \Delta t Q f(\vec{u})$$

Recap:

- Approximate right-hand side by a degree M polynomial
- Use quadrature rule to integrate the polynomial exactly
- For special $\vec{\tau}$, the solution at $t + \Delta t$ has up to order $2M$
- Corresponds to fully implicit Runge-Kutta method, Butcher matrix Q
- Problem: Q is dense $\implies$ direct solve is very expensive!

Collocation Methods Continued

Use vector notation and rescale quadrature nodes from 0 to 1:

$$(\vec{u})_m = u_m \approx u(\tau_m),\ (\vec{\tau})_m = \tau_m,\ (\vec{u}_0)_m = u_0,\ (Q)_{m,j} = q_{m,j},\ (f(\vec{u}))_m = f(\tau_m, u_m)$$

$$\vec{u} = \vec{u}_0 + \Delta t Q f(\vec{u})$$

Recap:

- Approximate right-hand side by a degree M polynomial
- Use quadrature rule to integrate the polynomial exactly
- For special $\vec{\tau}$, the solution at $t + \Delta t$ has up to order $2M$
- Corresponds to fully implicit Runge-Kutta method, Butcher matrix Q
- Problem: Q is dense $\implies$ direct solve is very expensive!

Spectral Deferred Corrections

Basic idea

- Use spectral quadrature rule to get solutions of order $2M$ or $2M-1$ (or $2M-2$)
- Solve equation for the error with simple quadrature rule (originally Euler) and refine the solution
- Iterate

Key innovation: Apply deferred corrections to integral form of initial value problem

Error Equation

- Error at iteration k depends on unknown exact solution:

$$\delta^k(t) = u(t) - \vec{u}^k \vec{l}^\tau(t)$$

- Plugging error into initial value problem gives:

$$\delta^k(t) - \int_0^t \left(f\left(\vec{u}^k \vec{l}^\tau(s) + \delta^k(s)\right) - f\left(\vec{u}^k \vec{l}^\tau(s)\right)\right) ds = r^k(t)$$

- Residual depends only on available quantities:

$$r^k(t) = u_0 + \int_0^t \left(f(\vec{u}^k \vec{l}^\tau(s)) - \vec{u}^k \vec{l}^\tau(s)\right) ds$$

- Discretize error equation with "some" quadrature rule Q_Δ at the same nodes τ:

$$\vec{\delta}^k - \Delta t Q_\Delta \left(f\left(\vec{u}^k + \vec{\delta}^k\right) - f\left(\vec{u}^k\right)\right) = \vec{r}^k$$

- Solve this for $\vec{\delta}^k$ and update the solution:

$$\vec{u}^{k+1} = \vec{u}^k + \vec{\delta}^k$$

Error Equation

- Error at iteration k depends on unknown exact solution:

$$\delta^k(t) = u(t) - \vec{u}^k \vec{l}^\tau(t)$$

- Plugging error into initial value problem gives:

$$\delta^k(t) - \int_0^t \left(f\left(\vec{u}^k \vec{l}^\tau(s) + \delta^k(s)\right) - f\left(\vec{u}^k \vec{l}^\tau(s)\right)\right) ds = r^k(t)$$

- Residual depends only on available quantities:

$$r^k(t) = u_0 + \int_0^t \left(f(\vec{u}^k \vec{l}^\tau(s)) - \vec{u}^k \vec{l}^\tau(s)\right) ds$$

- Discretize error equation with "some" quadrature rule Q_Δ at the same nodes τ:

$$\vec{\delta}^k - \Delta t Q_\Delta \left(f\left(\vec{u}^k + \vec{\delta}^k\right) - f\left(\vec{u}^k\right)\right) = \vec{r}^k$$

- Solve this for $\vec{\delta}^k$ and update the solution:

$$\vec{u}^{k+1} = \vec{u}^k + \vec{\delta}^k$$

Error Equation Continued

What have we gained?
$\rightarrow$ Nothing, if we solve the error equation with the same $Q_\Delta = Q$ we used for the collocation problem!

Need simpler quadrature rule Q_Δ (called preconditioner) to solve for the error.
For instance, implicit Euler:

$$Q_\Delta = \begin{pmatrix} \tau_2 - \tau_1 & 0 & 0 & \dots & 0 \\ \tau_2 - \tau_1 & \tau_3 - \tau_2 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \dots & 0 \\ \tau_2 - \tau_1 & \tau_3 - \tau_2 & \dots & \dots & \tau_M - \tau_{M-1} \end{pmatrix}$$

Error Equation Continued

Resulting iteration after algebraic manipulation:

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

Compare to vanilla implicit Euler:

$$\left(1 - \Delta t f\right)\left(u\right) = u_0$$

Now did we gain something?

- We better choose Q_Δ lower triangular such we can solve with forward substitution
- We just need to solve implicit Euler steps with modified step size and right-hand side
- If we choose Q_Δ diagonal, we can solve for the nodes in parallel!
- Consider PDE with N degrees of freedom: Collocation problem is size $NM \times NM$, but if SDC convergences after K iterations, it requires KM solves of $N \times N$ systems

Modern Interpretation of SDC

- Consider fully implicit collocation problem: $(I - \Delta t Q f)(\vec{u}) = \vec{u}_0$
- Simplest iterative approach: Picard iteration:

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

$\rightarrow$ poor stability because it is explicit

- Precondition the Picard iteration with Q_Δ:

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

- Looks familiar! SDC = preconditioned Picard iteration

Modern Interpretation of SDC Continued

Construct SDC iteration matrix

- Consider linear test equation: $u_t = \lambda u$
- SDC iteration becomes:

$$\vec{u}^{k+1} = \underbrace{(I - \Delta t Q_\Delta \lambda)^{-1} \Delta t \left(Q - Q_\Delta\right) \lambda \vec{u}^k}_{G\vec{u}^k} + \underbrace{(I - \Delta t Q_\Delta \lambda)^{-1} \vec{u}_0}_{c}$$

- Convergence:
  - Look for Q_Δ with $\rho(G) < 1$
  - Look for Q_Δ with $||G|| < 1$
  - Look for Q_Δ that make G nilpotent

Q_Δ is now a preconditioner and not necessarily a quadrature rule!

Modern Interpretation of SDC Continued

Look at stiff limit of SDC iteration matrix

Stiff limit $\lambda \to -\infty$, $\lambda \in \mathbb{R}$:

$$\vec{u}^{k+1} \approx \underbrace{\left(I - Q_\Delta^{-1} Q\right)}_{G} \vec{u}^k$$

- LU: $Q_\Delta = U^T$ with $LU = Q^T$ and $L_{ii} = 1$

  $$G = I - \left(U^T\right)^{-1} U^T L^T = I - L^T$$

  is strictly upper triangular and hence nilpotent

- MIN: Numerically minimize spectral radius of G with Q_Δ diagonal

Why bother with SDC?

- Special time-marching schemes easier to construct in low order, use SDC to get higher order
- Can accelerate SDC with inexactness, adaptive resolution between iterations, ...
- Parallel-in-Time (PinT) extensions

$\rightarrow$ Much greater flexibility than most other RK schemes

But: For non-stiff problems, explicit RK methods are very hard to beat with SDC

SDC examples

Implicit-Explicit Splitting, with Daniel Ruprecht

- Replace $Q_\Delta f$ with $Q_{\Delta,I} f_I + Q_{\Delta,E} f_E$ in SDC iteration
- Choose $Q_{\Delta,E}$ strictly lower triangular for explicit integration
- Done

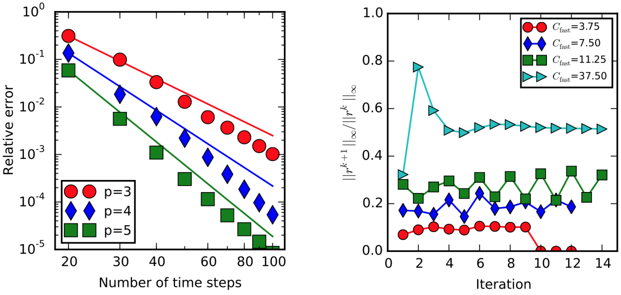
Methods and Algorithms for Scientific Computing

Spectral Deferred Corrections with Fast-wave Slow-wave Splitting

Authors: Daniel Ruprecht and Robert Speck | AUTHORS INFO & AFFILIATIONS

https://doi.org/10.1137/16M1060078

GET ACCESS BibTeX Tools



SDC examples

Boris SDC, with Mathias Winkel and Daniel Ruprecht

- Apply SDC to second-order equations of motion
- Choose Q_Δs to mimick velocity-Verlet integration
- Apply Boris trick for the (seemingly) implicit part
- Done

ELSEVIER

Contents lists available at ScienceDirect

Journal of Computational Physics

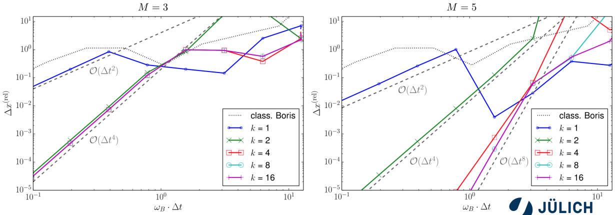
www.elsevier.com/locate/jcp

A high-order Boris integrator

Mathias Winkel [a,*], Robert Speck [b,a], Daniel Ruprecht [a]

[a] Institute of Computational Science, University of Lugano, Switzerland

[b] Jülich Supercomputing Centre, Forschungszentrum Jülich, Germany



SDC examples

Adaptive SDC, with Thomas Baumann and the TUHH crew

Numerical Algorithms
https://doi.org/10.1007/s11075-024-01964-z

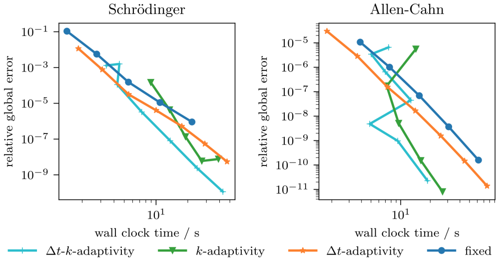
RESEARCH

Adaptive time step selection for spectral deferred correction

Thomas Baumann [1,2] · Sebastian Götschel [2] · Thibaut Lunet [2] · Daniel Ruprecht [2] · Robert Speck [1]

Received: 20 March 2024 / Accepted: 17 October 2024
© The Author(s) 2024

- Use update formula to get an error estimate (or be even more clever)
- Use standard RK techniques to find new Δt
- Done



Parallel-in-Time extensions

Four approaches

Following Kevin Burrage's terminology:

1. "Parallelization across the method": computation of the solution at all M stages at once
   1. using diagonalization of Q
   2. using diagonal preconditioners
2. "Parallelization across the steps": computation of the solution at multiple steps at once
   1. using multilevel/multigrid techniques
   2. using diagonalization techniques

Parallel-in-Time extensions

Four approaches

Following Kevin Burrage's terminology:

1. "Parallelization across the method": computation of the solution at all M stages at once
   1. using diagonalization of Q
   2. using diagonal preconditioners
2. "Parallelization across the steps": computation of the solution at multiple steps at once
   1. using multilevel/multigrid techniques
   2. ~~using diagonalization techniques~~ - not this talk

Parallelization across the method I

Diagonalization

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

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

with diagonal matrix D_Q.

Remarks:

- 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: D_Q has complex entries!

Parallelization across the method II

Parallel SDC, with Ruth Schöbel, Daniel Ruprecht, Thibaut Lunet, Gayatri Caklovic and others

Idea: use diagonal Q_Δ to compute updates simultaneously for all collocation nodes

How to find a suitable Q_Δ?

1. Standard tricks like the diagonal of Q (don't work well)
2. Minimize $\rho(I - Q_\Delta^{-1} Q)$ to tune the iteration for the stiff limit (works well for stiff problems)
3. Use machine/reinforcement learning to find the "optimal" entries of Q_Δ for a given problem class

Parallelization across the method II

Parallel SDC, with Ruth Schöbel, Daniel Ruprecht, Thibaut Lunet, Gayatri Caklovic and others

Idea: use diagonal Q_Δ to compute updates simultaneously for all collocation nodes

How to find a suitable Q_Δ?

1. Standard tricks like the diagonal of Q (don't work well)
2. Minimize $\rho(I - Q_\Delta^{-1}Q)$ to tune the iteration for the stiff limit (works well for stiff problems)
3. Use machine/reinforcement learning to find the "optimal" entries of Q_Δ for a given problem class

arXiv > math > arXiv:2403.18641

Mathematics > Numerical Analysis

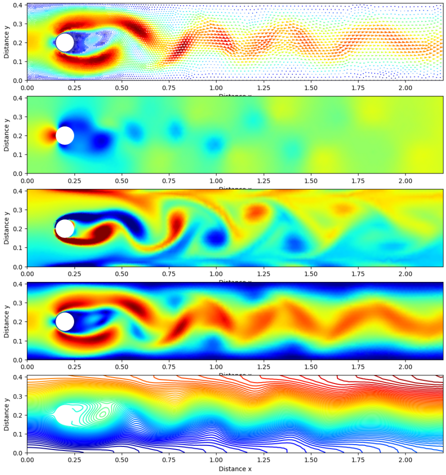
[Submitted on 27 Mar 2024]

Improving Efficiency of Parallel Across the Method Spectral Deferred Corrections

Gayatri Čaklović, Thibaut Lunet, Sebastian Götschel, Daniel Ruprecht

Parallel SDC for Navier-Stokes equations

Monolithic SDC with diagonal preconditioners, with Abdelouahed Ouardghi



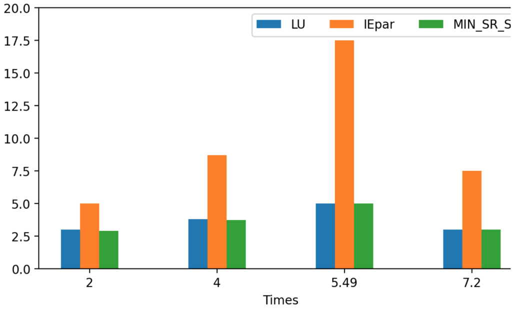


Figure: Left: Flow around the cylinder, DFG95 benchmark. Top: Number of iterations for different SDC preconditioners at selected time-steps. Smaller is better, blue is serial reference.

Parallelization across the steps

Multigrid for the composite collocation problem

We now glue L time-steps together, using N to transfer information from step l to step $l+1$. We get the composite collocation problem:

$$\begin{pmatrix} I-\Delta tQF & & & \\ -N & I-\Delta tQF & & \\ & \ddots & \ddots & \\ & & -N & I-\Delta tQF \end{pmatrix} \begin{pmatrix} \vec{u}_1 \\ \vec{u}_2 \\ \vdots \\ \vec{u}_L \end{pmatrix} = \begin{pmatrix} \vec{u}_0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

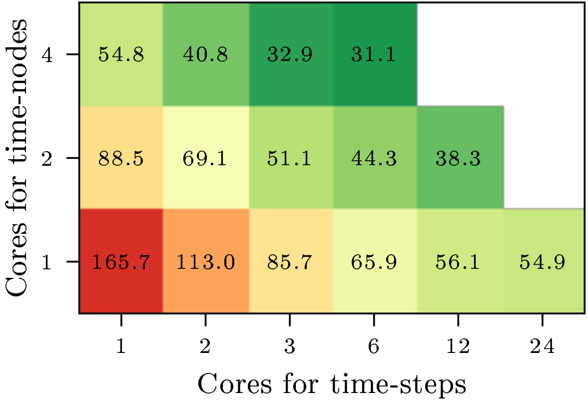
Parallel Full Approximation Scheme in Space and Time (PFASST, Minion and Emmett, 2012):

- use (linear/FAS) multigrid to solve this system iteratively
- smoother: parallel block-wise Jacobi with SDC in the blocks
- coarse-level solver: serial block-wise Gauß-Seidel with SDC in the blocks
- exploit cheapest coarse level to quickly propagate information forward in time

One step further: PFASST-ER

PFASST + parallel SDC, with Ruth Schöbel

Idea: Use parallel SDC sweeps within parallel time-steps
Example: 2D Allen-Cahn, fully-implicit, 256x256 DOFs in space, up to 24 available cores.



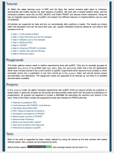
<commercial>

pySDC - Prototyping Spectral Deferred Corrections

Test before you invest at `https://parallel-in-time.org/pySDC`

Tutorials and examples

- Ships with a lot of examples
- Many SDC flavors up to PFASST
- Problems beyond heat equation



Parallel and serial

- Serial algorithms
- Pseudo-parallel algorithms
- Time-parallel algorithms
- Space-time parallel algorithms

Python

- Interface compiled code for expensive spatial solves
- Implementation close to formulas

CI/CD/CT

- Well documented
- Well tested
- Works ~~on my machine~~ anywhere
- Reproduce paper results

JÜLICH
Forschungszentrum

Code separated into modules

Problem

- implicit Euler like solves
- evaluate right hand side
- initial conditions, maybe exact solution
- use your own datatype

Sweeper: Timestepping

- assembles and calls solves in problem class
- administers right hand side evaluations
- takes care of Q_Δ, splitting etc.
- DIRK methods available as sweepers

Callbacks: Modify anything at any time

- solution
- step size
- sweeper
- ...

Hooks: Extract anything at any time

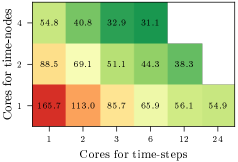
- Newton / SDC iterations and f evaluations
- wall time
- error
- ...

</commercial>

Three takeaways

Spectral Deferred Corrections (SDC) are a great playground for research on time integration methods

Lots of SDC variants and their combination can lead to highly competitive time integration methods (and are a lot of fun)



Prototyping ideas, with real code, on real (parallel) machines, is crucial to find out about potential and limitations