TIME-PARALLEL IMEX TIME INTEGRATION BASED ON SPECTRAL DEFERRED CORRECTION

Thibaut Lunet, Gayatri Čaklović, Thomas Saupe (né Baumann),
Daniel Ruprecht, Sebastian Götschel, Robert Speck, Philip Freese, Martin Schreiber

Institute for Autonomous Cyber-Physical Systems (E-24)
Hamburg University of Technology

Go20, Malta

Mai 18th. 2026

Usual dilemma with time-integration

TUHH
Hamburg University of Technology

Choice between :

Explicit time integration

- simple to implement, but limited time-step size for numerical stability
- good old RK4 quite hard to beat $\rightarrow$ limited potential for improvement

$\Rightarrow$ usually more adapted to (non-linear) advection-dominated problems

Usual dilemma with time-integration

TUHH
Hamburg University of Technology

Choice between :

Explicit time integration

- simple to implement, but limited time-step size for numerical stability
- good old RK4 quite hard to beat $\rightarrow$ limited potential for improvement

$\Rightarrow$ usually more adapted to (non-linear) advection-dominated problems

Implicit time integration

- unconditionally stable, but harder to implement (and parallelize ...)
- require a (non-)linear solve for each time-step

$\Rightarrow$ usually more adapted to (linear) diffusion-dominated problems

Usual dilemma with time-integration

TUHH
Hamburg University of Technology

Choice between :

Explicit time integration

- simple to implement, but limited time-step size for numerical stability
- good old RK4 quite hard to beat $\rightarrow$ limited potential for improvement

$\Rightarrow$ usually more adapted to (non-linear) advection-dominated problems

Implicit time integration

- unconditionally stable, but harder to implement (and parallelize ...)
- require a (non-)linear solve for each time-step

$\Rightarrow$ usually more adapted to (linear) diffusion-dominated problems

When you can't choose, simply take both : Implicit-Explicit methods (IMEX)

$$M\frac{\partial U}{\partial t} - L.U(t) = \mathcal{N}\left(U(t), t\right), \quad U(0) = U_0 \in \mathbb{C}^{N_{DoF}}$$

- L : linear operator (diffusive part), solved **implicitly**
- $\mathcal{N}$: non-linear operator (advective part), solved **explicitly**
- M : mass matrix, (singular) **such that** $M - \alpha L$ is invertible for any $\alpha \in]0, +\infty[$

$$M\frac{\partial U}{\partial t} - L.U(t) = \mathcal{N}\left(U(t), t\right), \quad U(0) = U_0 \in \mathbb{C}^{N_{DoF}}$$

- L : linear operator (diffusive part), solved **implicitly**
- $\mathcal{N}$: non-linear operator (advective part), solved **explicitly**
- M : mass matrix, (singular) **such that** $M - \alpha L$ is invertible for any $\alpha \in]0, +\infty[$

Most IMEX implementations require 3 main routines :

- `evalL(U)` : evaluate L on a solution U
- `evalN(U,t)` : evaluate $\mathcal{N}$ on a solution U at time t
- `linSolve(alpha,rhs)` : solve $(M - \alpha L)U = rhs$ for any rhs and α

$$M\frac{\partial U}{\partial t} - L.U(t) = \mathcal{N}\left(U(t), t\right), \quad U(0) = U_0 \in \mathbb{C}^{N_{DoF}}$$

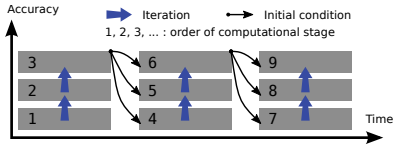
- L : linear operator (diffusive part), solved **implicitly**
- $\mathcal{N}$: non-linear operator (advective part), solved **explicitly**
- M : mass matrix, (singular) **such that** $M - \alpha L$ is invertible for any $\alpha \in]0, +\infty[$

Most IMEX implementations require 3 main routines :

- `evalL(U)` : evaluate L on a solution U
- `evalN(U,t)` : evaluate $\mathcal{N}$ on a solution U at time t
- `linSolve(alpha,rhs)` : solve $(M - \alpha L)U = rhs$ for any *rhs* and α

Spectral Deferred Correction : very good challenger to current state-of-the art for IMEX

Once upon a PinT ... Spectral Deferred Corrections



SDC[1] was introduced in 2000 on the basis of Deferred Correction[2]

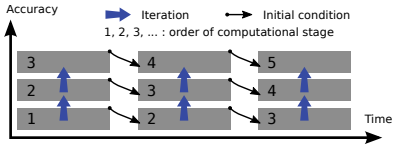
[1] Dutt, Greengard, and Rokhlin, 2000. *"Spectral deferred correction methods for ordinary differential equations"*.

[2] Fox, 1947. *"Some improvements in the use of relaxation methods for the solution of ordinary and partial differential equations"*.

[3] Guibert and Tromeur-Dervout, 2007. *"Parallel deferred correction method for CFD problems"*.

[4] Christlieb, Macdonald, and Ong, 2010. *"Parallel high-order integrators"*.

[5] Emmett and Minion, 2012. *"Toward an efficient parallel in time method for partial differential equations"*.



SDC[1] was introduced in 2000 on the basis of Deferred Correction[2]
$\Rightarrow$ induced many contributions to **parallel in time (PinT)** since then :

- 2007 : pipeline parallel sweeps across the steps[3], *a.k.a* (Block) Gauss Seidel

[1] Dutt, Greengard, and Rokhlin, 2000. *"Spectral deferred correction methods for ordinary differential equations"*.

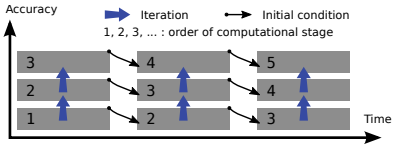
[2] Fox, 1947. *"Some improvements in the use of relaxation methods for the solution of ordinary and partial differential equations"*.

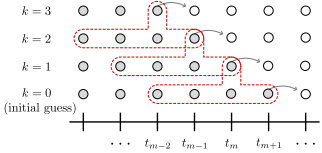
[3] Guibert and Tromeur-Dervout, 2007. *"Parallel deferred correction method for CFD problems"*.

[4] Christlieb, Macdonald, and Ong, 2010. *"Parallel high-order integrators"*.

[5] Emmett and Minion, 2012. *"Toward an efficient parallel in time method for partial differential equations"*.

Once upon a PinT ... Spectral Deferred Corrections





SDC[1] was introduced in 2000 on the basis of Deferred Correction[2]

$\Rightarrow$ induced many contributions to **parallel in time (PinT)** since then :

- 2007 : pipeline parallel sweeps across the steps[3], *a.k.a* (Block) Gauss Seidel
- 2010 : pipeline parallel sweeps across many nodes (RIDC)[4]

[1] Dutt, Greengard, and Rokhlin, 2000. *"Spectral deferred correction methods for ordinary differential equations"*.

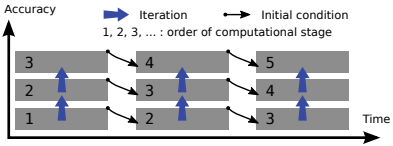
[2] Fox, 1947. *"Some improvements in the use of relaxation methods for the solution of ordinary and partial differential equations"*.

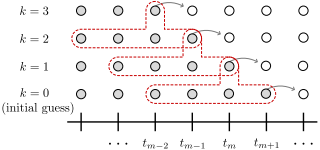
[3] Guibert and Tromeur-Dervout, 2007. *"Parallel deferred correction method for CFD problems"*.

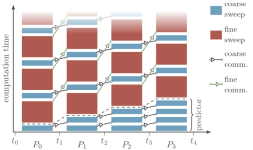
[4] Christlieb, Macdonald, and Ong, 2010. *"Parallel high-order integrators"*.

[5] Emmett and Minion, 2012. *"Toward an efficient parallel in time method for partial differential equations"*.

Once upon a PinT ... Spectral Deferred Corrections







SDC[1] was introduced in 2000 on the basis of Deferred Correction[2]

$\Rightarrow$ induced many contributions to **parallel in time (PinT)** since then :

- 2007 : pipeline parallel sweeps across the steps[3], *a.k.a* (Block) Gauss Seidel
- 2010 : pipeline parallel sweeps across many nodes (RIDC)[4]
- 2012 : multi-level sweeps with parallel fine sweep across the steps (PFASST)[5]

[1] Dutt, Greengard, and Rokhlin, 2000. *"Spectral deferred correction methods for ordinary differential equations"*.

[2] Fox, 1947. *"Some improvements in the use of relaxation methods for the solution of ordinary and partial differential equations"*.

[3] Guibert and Tromeur-Dervout, 2007. *"Parallel deferred correction method for CFD problems"*.

[4] Christlieb, Macdonald, and Ong, 2010. *"Parallel high-order integrators"*.

[5] Emmett and Minion, 2012. *"Toward an efficient parallel in time method for partial differential equations"*.

1. consider the ODE system :

$$\frac{du}{dt} = f(u,t) \qquad t \in [0, \Delta t], \quad u(0) = u_0$$

2. write it into Picard form :

$$u(t) = u_0 + \int_0^t f(u(s), s)\, \mathrm{d}s, \qquad 0 \leq t \leq \Delta t$$

1. consider the ODE system :

$$\frac{du}{dt} = f(u,t) \qquad t \in [0, \Delta t], \quad u(0) = u_0$$

2. write it into Picard form :

$$u(t) = u_0 + \int_0^t f(u(s), s)\, \mathrm{d}s, \qquad 0 \leq t \leq \Delta t$$

3. use a given quadrature rule to approximate the integral on M **nodes** τ_m :

$$u_m = u_n + \Delta t \sum_{j=1}^{M} q_{m,j} f(u_j, \tau_j), \quad m = 1, \ldots, M, \quad \tau_m \in [0,1]$$

$$\Rightarrow \textbf{Collocation method}$$

Fully implicit M-stage Runge-Kutta methods

$$\begin{array}{c|ccc} \tau_1 & q_{1,1} & \dots & q_{1,M} \\ \vdots & \vdots & & \vdots \\ \tau_M & q_{1,M} & \dots & q_{M,M} \\ \hline & \omega_1 & \dots & \omega_M \end{array}$$

Node solutions computation can be seen as a **all-at-once system** :

$$\mathbf{u} - \Delta t Q f(\mathbf{u}) = \mathbf{u}_0, \quad \mathbf{u} = [.., u_m, ..], \quad \mathbf{u}_0 = [u_0, .., u_0]$$

Fully implicit M-stage Runge-Kutta methods

$$\begin{array}{c|ccc} \tau_1 & q_{1,1} & \dots & q_{1,M} \\ \vdots & \vdots & & \vdots \\ \tau_M & q_{1,M} & \dots & q_{M,M} \\ \hline & \omega_1 & \dots & \omega_M \end{array}$$

Node solutions computation can be seen as a **all-at-once system** :

$$\mathbf{u} - \Delta t Q f(\mathbf{u}) = \mathbf{u}_0, \quad \mathbf{u} = [.., u_m, ..], \quad \mathbf{u}_0 = [u_0, .., u_0]$$

Once node solutions u_m (stages) are known, we perform (or not) the **step update** :

$$u_{n+1} = u_n + \Delta t \sum_{j=1}^{M} \omega_j f(u_j, \tau_j)$$

SDC $\Leftrightarrow$ *approximate the all-at-once with an* **iterative method** *(iterative Runge-Kutta)*

Solve iteratively the all-at-once system : $(I - \Delta t Q f)\mathbf{u} = \mathbf{u}_0$

4. consider the Picard iteration :

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

Solve iteratively the all-at-once system : $(I - \Delta t Q f)\mathbf{u} = \mathbf{u}_0$

4. consider the Picard iteration :

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

5. improve it with a preconditioned iteration :

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

Solve iteratively the all-at-once system : $(I - \Delta t Q f)\mathbf{u} = \mathbf{u}_0$

4. consider the Picard iteration :

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

5. improve it with a preconditioned iteration :

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

6. build P with a **lower triangular approximation** of Q :

$$Q_\Delta \simeq Q \quad \Rightarrow \quad P := (I - \Delta t Q_\Delta f) \quad \Rightarrow \text{direct solution with backward substitution}$$

$\rightarrow$ *repeat* K *times the SDC iteration, a.k.a* **sweep**

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

Backward Euler based SDC (implicit) :

$$\mathbf{Q}_\Delta^{BE} = \begin{pmatrix} \Delta\tau_1 & & & \\ \Delta\tau_1 & \Delta\tau_2 & & \\ \vdots & & \ddots & \\ \Delta\tau_1 & \dots & \dots & \Delta\tau_M \end{pmatrix}$$

$$u_m^{k+1} = u_n + \Delta t \sum_{j=1}^{M} q_{m,j} f(u_j^k, \tau_j) + \Delta t \sum_{j=1}^{m} \Delta\tau_j [f(u_j^{k+1}, \tau_j) - f(u_j^k, \tau_j)]$$

- quadrature terms
- correction terms

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

Forward Euler based SDC (explicit) :

$$\mathbf{Q}_\Delta^{FE} = \begin{pmatrix} 0 & & & \\ \Delta\tau_2 & 0 & & \\ \vdots & & \ddots & \\ \Delta\tau_2 & \dots & \Delta\tau_M & 0 \end{pmatrix}$$

$$u_m^{k+1} = u_n + \Delta t \sum_{j=1}^{M} q_{m,j} f(u_j^k, \tau_j) + \Delta t \sum_{j=1}^{m-1} \Delta\tau_{j+1} [f(u_j^{k+1}, \tau_j) - f(u_{j-1}^k, \tau_{j-1})]$$

- quadrature terms
- correction terms

More expensive than classical methods (RK, Multistep, ...) **but** several advantages :

1. order of accuracy $\Leftrightarrow$ number of sweeps $K \rightarrow$ high order is easy to reach

More expensive than classical methods (RK, Multistep, ...) **but** several advantages :

1. order of accuracy $\Leftrightarrow$ number of sweeps $K \rightarrow$ high order is easy to reach
2. easily generalizable to IMEX splitting[6]

[6] Minion, 2003. *"Semi-implicit spectral deferred correction methods for ordinary differential equations"*.

More expensive than classical methods (RK, Multistep, ...) **but** several advantages :

1. order of accuracy $\Leftrightarrow$ number of sweeps $K \rightarrow$ high order is easy to reach
2. easily generalizable to IMEX splitting[6]
3. simple way to avoid the step update (stiffly accurate methods)

[6] Minion, 2003. *"Semi-implicit spectral deferred correction methods for ordinary differential equations"*.

More expensive than classical methods (RK, Multistep, ...) **but** several advantages :

1. order of accuracy $\Leftrightarrow$ number of sweeps $K \rightarrow$ high order is easy to reach
2. easily generalizable to IMEX splitting[6]
3. simple way to avoid the step update (stiffly accurate methods)
4. high numerical stability with explicit forms

[6] Minion, 2003. *"Semi-implicit spectral deferred correction methods for ordinary differential equations"*.

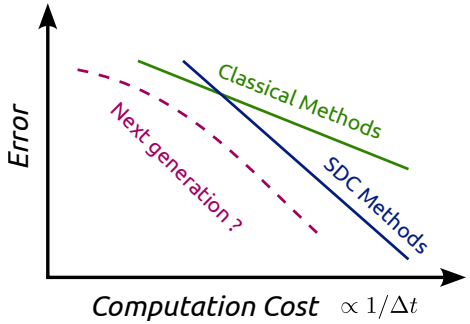
More expensive than classical methods (RK, Multistep, ...) **but** several advantages :

1. order of accuracy $\Leftrightarrow$ number of sweeps $K \rightarrow$ high order is easy to reach
2. easily generalizable to IMEX splitting[6]
3. simple way to avoid the step update (stiffly accurate methods)
4. high numerical stability with explicit forms
5. very high modularity (quadrature nodes, K, Q_Δ coefficients, ...)

$\Rightarrow$ very good **design framework** for the next generation time-integration methods

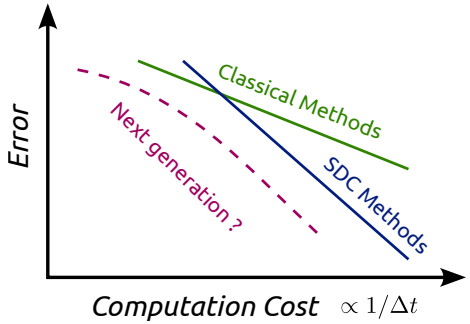
[6] Minion, 2003. *"Semi-implicit spectral deferred correction methods for ordinary differential equations"*.

Better error *vs.* cost ratio

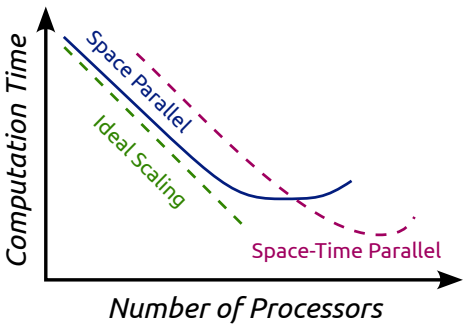


Main objectives

Better error *vs.* cost ratio

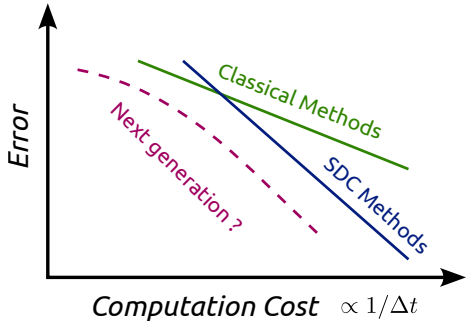


Better parallel efficiency

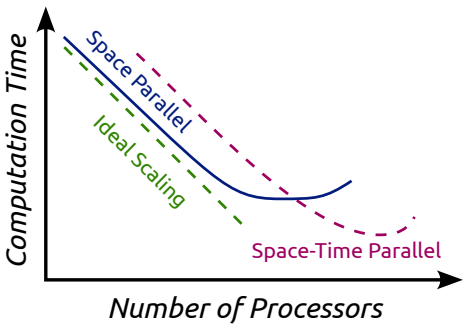


Main objectives

Better error *vs.* cost ratio



Better parallel efficiency



$\rightarrow$ *here is a first glitch of what can be done with SDC ...*

Parallel Diagonal SDC[7,8] : base idea

Hamburg University of Technology

Consider a diagonal $\mathbf{Q}_\Delta$ matrix :
$\Rightarrow$ time-parallel sweeps across nodes !

... *how to choose the coefficients* ?

$$\mathbf{Q}_\Delta^{diag} = \begin{pmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_M \end{pmatrix}$$

[7] Houwen and Sommeijer, 1991. *"Iterated Runge–Kutta methods on parallel computers"*.

[8] Speck, 2018. *"Parallelizing spectral deferred corrections across the method"*.

Consider a diagonal $\mathbf{Q}_\Delta$ matrix :
$\Rightarrow$ time-parallel sweeps across nodes !

$$\mathbf{Q}_\Delta^{diag} = \begin{pmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_M \end{pmatrix}$$

... *how to choose the coefficients* ?
$\rightarrow$ write the SDC iteration matrix on Dahlquist problem :

$$\boldsymbol{e}^{k+1} = \mathbf{K}(\lambda\Delta t)\boldsymbol{e}^k \quad \Rightarrow \quad \mathbf{K}(\lambda\Delta t) = \lambda\Delta t(\mathbf{I} - \lambda\Delta t\mathbf{Q}_\Delta)^{-1}(\mathbf{Q} - \mathbf{Q}_\Delta)$$

$\rightarrow$ find the d_m coefficients minimizing the spectral radius of :

1. $\mathbf{I} - \mathbf{Q}_\Delta^{-1}\mathbf{Q} = \lim_{|\lambda\Delta t|\rightarrow\infty} \mathbf{K}(\lambda\Delta t)$ (stiff limit)
2. $\mathbf{Q} - \mathbf{Q}_\Delta = \lim_{|\lambda\Delta t|\rightarrow 0} \frac{\mathbf{K}(\lambda\Delta t)}{\lambda\Delta t}$ (non-stiff limit)

[7] Houwen and Sommeijer, 1991. *"Iterated Runge–Kutta methods on parallel computers"*.

[8] Speck, 2018. *"Parallelizing spectral deferred corrections across the method"*.

Parallel Diagonal SDC[9] : proposed solution

Look at **nilpotency** (and polynomials) rather than spectral radius (ill-conditioned)

[9] Čaklović et al., 2025. *"Improving Efficiency of Parallel Across the Method Spectral Deferred Corrections"*.

Look at **nilpotency** (and polynomials) rather than spectral radius (ill-conditioned)
$\Rightarrow$ three candidates :

1. `MIN-SR-NS` : $\mathbf{Q}_\Delta = diag(\tau_1/M, .., \tau_M/M)$ makes $\mathbf{Q} - \mathbf{Q}_\Delta$ nilpotent
2. `MIN-SR-S` : $\mathbf{Q}_\Delta$ that makes $\mathbf{I} - \mathbf{Q}_\Delta^{-1}\mathbf{Q}$ nilpotent, solving

$$\det((1-z)\mathbf{I} + z\mathbf{Q}_\Delta^{-1}\mathbf{Q}) = 1, \quad z \in \{\tau_1, ..., \tau_M\}$$

3. `MIN-SR-FLEX` : change $\mathbf{Q}_\Delta$ along iterations to make this matrix

$$(\mathbf{I} - \mathbf{Q}_\Delta^{-1}(M)\mathbf{Q}) \dots (\mathbf{I} - \mathbf{Q}_\Delta^{-1}(2)\mathbf{Q})(\mathbf{I} - \mathbf{Q}_\Delta^{-1}(1)\mathbf{Q})$$

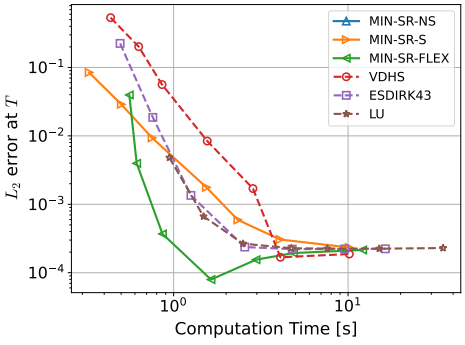
nilpotent, using $\mathbf{Q}_\Delta(m) = diag(\tau_1/m, .., \tau_M/m)$

[9] Čaklović et al., 2025. *"Improving Efficiency of Parallel Across the Method Spectral Deferred Corrections"*.

TUHH
Hamburg University of Technology

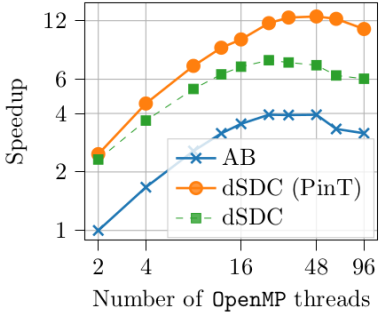
Implementation and evaluation of Parallel SDC sweeps :

1D non-linear Allen Cahn[a]



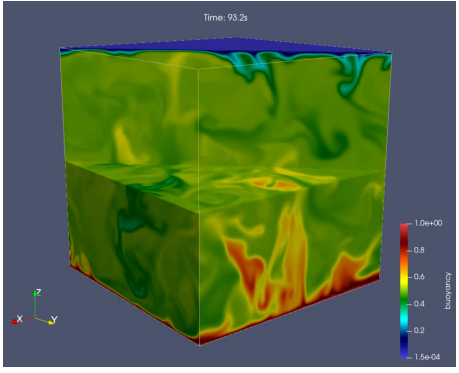
[a] Čaklović et al., 2025. *"Improving Efficiency of Parallel Across the Method Spectral Deferred Corrections"*.

2D shallow-water equations (sphere)[a]



[a] Freese et al., 2024. *"Parallel performance of shared memory parallel spectral deferred corrections"*.

Rayleigh-Bénard Convection in 3D



- turbulence induced by temperature gradient
- direct numerical simulation
- periodic boundary condition in x and y
- Dirichlet boundary condition in z
- velocities u_x, u_y, u_z, buoyancy b, pressure p

Depends on two dimensionless numbers :

1. Rayleigh Ra (turbulence)
2. Prandtl Pr (thermodynamic)

Normalized system of equations for the incompressible flow

$$\frac{\partial \mathbf{u}}{\partial t} - \sqrt{\frac{Pr}{Ra}}\Delta \mathbf{u} + \nabla p - b\mathbf{e}_z = \mathbf{u} \cdot \nabla \mathbf{u} \tag{1}$$

$$\frac{\partial b}{\partial t} - \frac{1}{\sqrt{RaPr}}\Delta b = \mathbf{u} \cdot \nabla b \tag{2}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{3}$$

Normalized system of equations for the incompressible flow

$$\frac{\partial \mathbf{u}}{\partial t} - \sqrt{\frac{Pr}{Ra}}\Delta \mathbf{u} + \nabla p - b\mathbf{e}_z = \mathbf{u} \cdot \nabla \mathbf{u} \tag{1}$$

$$\frac{\partial b}{\partial t} - \frac{1}{\sqrt{RaPr}}\Delta b = \mathbf{u} \cdot \nabla b \tag{2}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{3}$$

Pseudo-spectral space discretization (FFT-based) that solves the generic IMEX system :

$$M\frac{\partial U}{\partial t} - L.U(t) = \mathcal{N}\left(U(t), t\right), \quad U(0) = U_0 \in \mathbb{C}^{N_{DoF}}, \quad \text{with } M \text{ singular !}$$

$\Rightarrow$ *requires adapted time-integration schemes ...*

Time-integration using IMEX Runge-Kutta[10]

Stiffly accurate IMEX Butcher tables A and $\hat{A}$ with common stages $\mathbf{c}$

$$\begin{array}{c|cccc} 0 & 0 & 0 & \dots & 0 \\ c_1 & 0 & a_{1,1} & \dots & 0 \\ c_2 & 0 & a_{2,1} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & a_{M,1} & \dots & a_{M,M} \\ \hline & 0 & a_{M,1} & \dots & a_{M,M} \end{array} \qquad \begin{array}{c|cccc} 0 & 0 & 0 & \dots & 0 \\ c_1 & \hat{a}_{2,1} & 0 & \dots & 0 \\ c_2 & \hat{a}_{3,1} & \hat{a}_{3,2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \hat{a}_{s,1} & \hat{a}_{s,2} & \dots & 0 \\ \hline & \hat{a}_{s,1} & \hat{a}_{s,2} & \dots & 0 \end{array}$$

Time-stepping solves the **all-at-once system** to get $U_S = [U_1, \dots, U_s]^T$:

$$[I \otimes M - A \otimes L]\, U_S - \hat{A} \otimes \mathcal{F}(U_S, \mathbf{c}) = \mathbb{1} \otimes U_0$$

[10] Ascher, Ruuth, and Spiteri, 1997. *"Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations"*.

Time-integration using IMEX Runge-Kutta[10]

TUHH
Hamburg University of Technology

Stiffly accurate IMEX Butcher tables A and $\hat{A}$ with common stages $\mathbf{c}$

$$\begin{array}{c|cccc} 0 & 0 & 0 & \dots & 0 \\ c_1 & 0 & a_{1,1} & \dots & 0 \\ c_2 & 0 & a_{2,1} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & a_{M,1} & \dots & a_{M,M} \\ \hline & 0 & a_{M,1} & \dots & a_{M,M} \end{array} \qquad \begin{array}{c|cccc} 0 & 0 & 0 & \dots & 0 \\ c_1 & \hat{a}_{2,1} & 0 & \dots & 0 \\ c_2 & \hat{a}_{3,1} & \hat{a}_{3,2} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \hat{a}_{s,1} & \hat{a}_{s,2} & \dots & 0 \\ \hline & \hat{a}_{s,1} & \hat{a}_{s,2} & \dots & 0 \end{array}$$

Time-stepping solves the **all-at-once system** to get $U_S = [U_1, \dots, U_s]^T$:

$$[I \otimes M - A \otimes L]\, U_S - \hat{A} \otimes \mathcal{F}(U_S, \mathbf{c}) = \mathbb{1} \otimes U_0$$

$\Rightarrow$ stage-by-stage with backward substitution + **last stage as next step solution**

[10] Ascher, Ruuth, and Spiteri, 1997. *"Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations"*.

Collocation matrix Q on the nodes τ (stages) to build the all-at-once problem :

$$\begin{array}{c|ccc} \tau_1 & q_{1,1} & \cdots & q_{1,s} \\ \vdots & \vdots & & \vdots \\ 1 & q_{1,s} & \cdots & q_{s,s} \\ \hline & q_{1,s} & \cdots & q_{s,s} \end{array}$$

$$[I \otimes M - Q \otimes L]\, U_S - Q \otimes \mathcal{F}(U_S, \mathbf{c}) = \mathbb{1} \otimes U_0$$

Collocation matrix Q on the nodes τ (stages) to build the all-at-once problem :

$$\begin{array}{c|ccc} \tau_1 & q_{1,1} & \dots & q_{1,s} \\ \vdots & \vdots & & \vdots \\ 1 & q_{1,s} & \dots & q_{s,s} \\ \hline & q_{1,s} & \dots & q_{s,s} \end{array}$$

$$[I \otimes M - Q \otimes L]\, U_S - Q \otimes \mathcal{F}(U_S, \mathbf{c}) = \mathbb{1} \otimes U_0$$

Use K IMEX sweeps on $U_S^k = [U_1^k, \dots, U_s^k]^T$ (solved with backward substitution)

$$[I \otimes M - Q_\Delta \otimes L]\, U_S^{k+1} = \mathbb{1} \otimes U_0 + Q \otimes \left[L U_S^k + \mathcal{F}(U_S^k, \mathbf{c}) \right] - Q_\Delta \otimes L U_S^k$$

Collocation matrix Q on the nodes τ (stages) to build the all-at-once problem :

$$\begin{array}{c|ccc} \tau_1 & q_{1,1} & \dots & q_{1,s} \\ \vdots & \vdots & & \vdots \\ 1 & q_{1,s} & \dots & q_{s,s} \\ \hline & q_{1,s} & \dots & q_{s,s} \end{array}$$

$$[I \otimes M - Q \otimes L]\, U_S - Q \otimes \mathcal{F}(U_S, \mathbf{c}) = \mathbb{1} \otimes U_0$$

Use K IMEX sweeps on $U_S^k = [U_1^k, \dots, U_s^k]^T$ (solved with backward substitution)

$$[I \otimes M - Q_\Delta \otimes L]\, U_S^{k+1} = \mathbb{1} \otimes U_0 + Q \otimes \left[L U_S^k + \mathcal{F}(U_S^k, \mathbf{c}) \right] - Q_\Delta \otimes L U_S^k$$

Use a diagonal "approximation" $Q_\Delta \Rightarrow$ **parallel IMEX sweeps across the stages**

Comparison with state of the art

Considering the same number of stages $s = 4$, $K = 4$ for SDC and some $\mathbf{Q}_\Delta$ ...

Runge-Kutta (RK443)

- third order accuracy
- maximum CFL at 1.57
- computational cost $\simeq N_{stages}$
- sequential stage computation

PIMEX SDC

- fourth order accuracy
- maximum CFL at 2.83
- computational cost $\simeq M \times K$
- **parallel stage computation**

Comparison with state of the art

Considering the same number of stages $s = 4$, $K = 4$ for SDC and some $\mathbf{Q}_\Delta$ ...

Runge-Kutta (RK443)

- third order accuracy
- maximum CFL at 1.57
- computational cost $\simeq N_{stages}$
- sequential stage computation

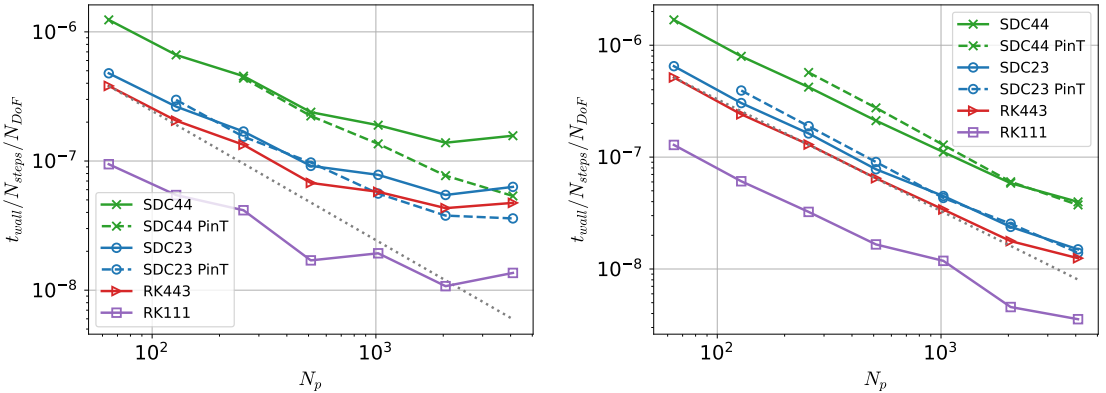
PIMEX SDC

- fourth order accuracy
- maximum CFL at 2.83
- computational cost $\simeq M \times K$
- **parallel stage computation**

Faster time-to-solution for SDC with parallel-in-time (PinT) **and** *time-step increase ?*

Strong scaling tests for single time-step

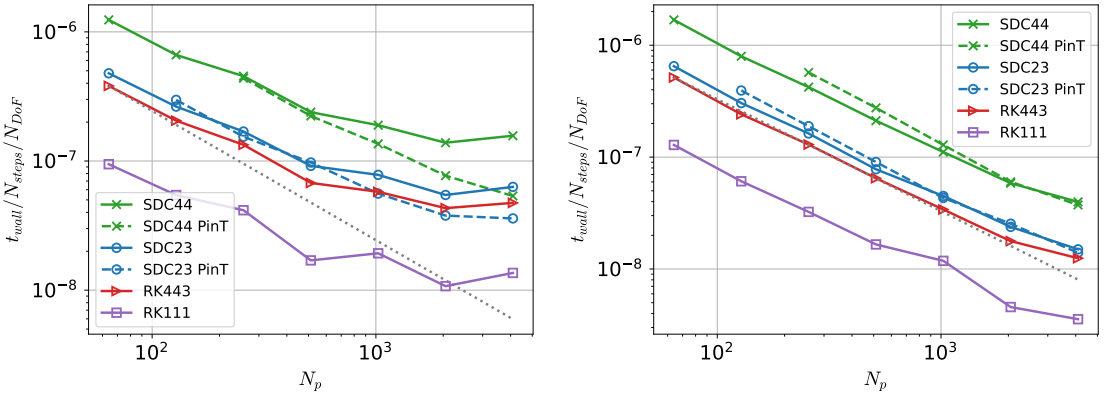
RBC on a 32×128^2 (left) and 64×256^2 (right) grid, 50% CPU node load (Jusuf, JSC)



Strong scaling tests for single time-step

TUHH
Hamburg University of Technology

RBC on a 32×128^2 (left) and 64×256^2 (right) grid, 50% CPU node load (Jusuf, JSC)



$\Rightarrow$ PIMEX SDC may win by a small margin, but **time-step size increase is crucial**

Numerical stability depending on Rayleigh number

Maximum time-step size and multiplicative factor vs RK111 :

Method	$Ra = 10^5$ $(128^2 \times 32)$	$Ra = 10^6$ $(256^2 \times 64)$	$Ra = 10^7$ $(512^2 \times 128)$
RK111	0.0526	0.00568	0.00144
RK443	0.0714 ($\times$1.4)	0.0147 ($\times$2.6)	0.0050 ($\times$3.5)
PIMEX SDC23	0.1000 ($\times$1.9)	0.0167 ($\times$2.9)	0.0056 ($\times$3.9)
PIMEX SDC44	0.0833 ($\times$1.6)	0.0192 ($\times$3.4)	0.0071 ($\times$4.9)

- stability advantages of PIMEX SDC increases with Ra
- projected computation time reduction using PinT around -20% for high Ra

Numerical stability depending on Rayleigh number

Maximum time-step size and multiplicative factor vs RK111 :

Method	$Ra = 10^5$ $(128^2 \times 32)$	$Ra = 10^6$ $(256^2 \times 64)$	$Ra = 10^7$ $(512^2 \times 128)$
RK111	0.0526	0.00568	0.00144
RK443	0.0714 ($\times$1.4)	0.0147 ($\times$2.6)	0.0050 ($\times$3.5)
PIMEX SDC23	0.1000 ($\times$1.9)	0.0167 ($\times$2.9)	0.0056 ($\times$3.9)
PIMEX SDC44	0.0833 ($\times$1.6)	0.0192 ($\times$3.4)	0.0071 ($\times$4.9)

- stability advantages of PIMEX SDC increases with *Ra*
- projected computation time reduction using PinT around -20% for high *Ra*

... now we just need access to a larger HPC cluster to demonstrate that

In the meantime :
can this stability improvement be demonstrated (or improved) with analysis ?

Dahlquist problem : $\frac{du}{dt} = (z_I + z_E)u$, $\Delta t = 1$, stage solution vector $\mathbf{u} = [u_1, ..., u_s]^T$

- IMEX RK step (or general collocation) :

$$\left(\mathbf{I} - z_I A - z_E \hat{A}\right)\mathbf{u} = \mathbf{u}_0 \qquad (\mathbf{I} - (z_I - z_E)\mathbf{Q})\,\mathbf{u} = \mathbf{u}_0$$

Dahlquist problem : $\frac{du}{dt} = (z_I + z_E)u$, $\Delta t = 1$, stage solution vector $\mathbf{u} = [u_1, ..., u_s]^T$

- IMEX RK step (or general collocation) :

$$\left(\mathbf{I} - z_I A - z_E \hat{A}\right)\mathbf{u} = \mathbf{u}_0 \qquad (\mathbf{I} - (z_I - z_E)\mathbf{Q})\,\mathbf{u} = \mathbf{u}_0$$

- PIMEX SDC sweep :

$$(\mathbf{I} - z_I \mathbf{Q}_\Delta)\,\mathbf{u}^{k+1} = \mathbf{u}_0 + (z_I + z_E)Q\mathbf{u}^k - z_I \mathbf{Q}_\Delta \mathbf{u}^k$$

Dahlquist problem : $\frac{du}{dt} = (z_I + z_E)u$, $\Delta t = 1$, stage solution vector $\mathbf{u} = [u_1, ..., u_s]^T$

- IMEX RK step (or general collocation) :

$$\left(\mathbf{I} - z_I A - z_E \hat{A}\right) \mathbf{u} = \mathbf{u}_0 \qquad (\mathbf{I} - (z_I - z_E)\mathbf{Q})\, \mathbf{u} = \mathbf{u}_0$$

- PIMEX SDC sweep :

$$(\mathbf{I} - z_I \mathbf{Q}_\Delta)\, \mathbf{u}^{k+1} = \mathbf{u}_0 + (z_I + z_E) Q \mathbf{u}^k - z_I \mathbf{Q}_\Delta \mathbf{u}^k$$

- generic IMEX SDC sweep :

$$\left(\mathbf{I} - z_I \mathbf{Q}_\Delta^I - z_E \mathbf{Q}_\Delta^E\right) \mathbf{u}^{k+1} = \mathbf{u}_0 + (z_I + z_E) Q \mathbf{u}^k - z_I \mathbf{Q}_\Delta \mathbf{u}^k - z_E \mathbf{Q}_\Delta^E \mathbf{u}^k$$

PIMEX SDC $\Leftrightarrow$ Picard for explicit ($\mathbf{Q}_\Delta^E = 0$), diagonal $\mathbf{Q}_\Delta$ coefficients for implicit

Advection-diffusion splitting

$$\frac{du}{dt} = (\lambda_I + i\lambda_E)u, \quad u_0 = 1, \quad \sqrt{i} = -1, \quad \lambda_I \in]-\infty, 0], \quad \lambda_E \in [0, +\infty]$$

$\rightarrow$ compute the numerical solution u_1 after one time-step $\Delta t = 1$, and look at :

1. amplification factor $|u_1|$: larger than 1 $\Rightarrow$ numerical instability
2. accuracy $|u_1 - e^{\lambda_I + i\lambda_E}|$: how accurate are we on each eigenvalue

[11] Ascher, Ruuth, and Spiteri, 1997. *"Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations"*.

Advection-diffusion splitting

$$\frac{du}{dt} = (\lambda_I + i\lambda_E)u, \quad u_0 = 1, \quad \sqrt{i} = -1, \quad \lambda_I \in]-\infty, 0], \quad \lambda_E \in [0, +\infty]$$

$\rightarrow$ compute the numerical solution u_1 after one time-step $\Delta t = 1$, and look at :

1. amplification factor $|u_1|$: larger than 1 $\Rightarrow$ numerical instability
2. accuracy $|u_1 - e^{\lambda_I + i\lambda_E}|$: how accurate are we on each eigenvalue

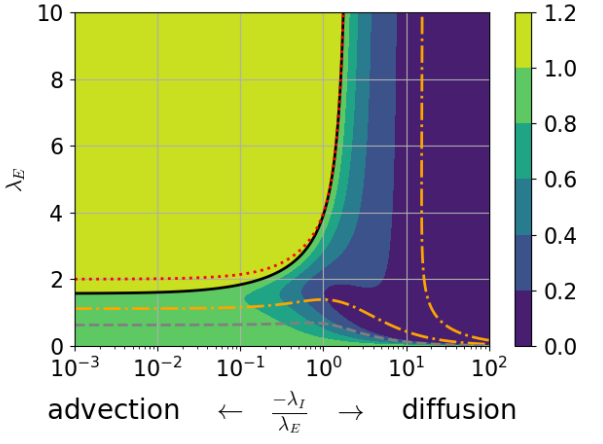
$\rightarrow$ contour plot[11] of those quantities in the $(-\lambda_I/\lambda_E, \lambda_E)$ plane ...

[11] Ascher, Ruuth, and Spiteri, 1997. *"Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations"*.

Maximum CFL for $-\lambda_I/\lambda_E \to 0$
$\Rightarrow$ $CFL_{max} = 1.57$

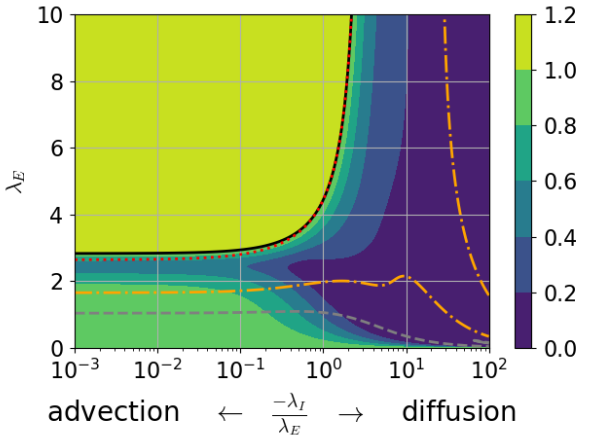
Accuracy contours

- gray dashes : 0.01
- orange dots-dashes : 0.1
- red dots : 1



PIMEX SDC44

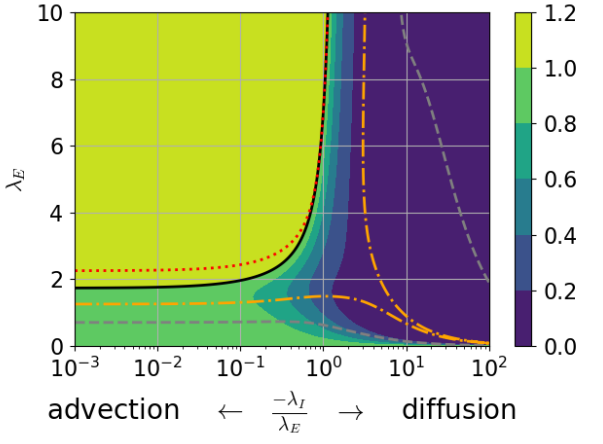
Maximum CFL for $-\lambda_I/\lambda_E \to 0$
$\Rightarrow$ $CFL_{max} = 2.83$ ($\times$ 1.8 vs RK443)



PIMEX SDC stability

PIMEX SDC23

Maximum CFL for $-\lambda_I/\lambda_E \to 0$
$\Rightarrow$ $CFL_{max} = 1.73$ ($\times$ 1.1 vs RK443)



Current limitations of this analysis

We don't really look at the effective values for λ_E and λ_I/λ_E ... we would also need :

- linearization of the governing equations for one time-steps
- expression depending on effective CFL and Ra
- taking into account 3D

Current limitations of this analysis

We don't really look at the effective values for λ_E and λ_I/λ_E ... we would also need :

- linearization of the governing equations for one time-steps
- expression depending on effective CFL and *Ra*
- taking into account 3D

But enough to get stability estimation and investigate other SDC variants ...

Idea : modify the implicit operator in SDC to increase stability[12] ... *because we can !*

$$\left(\mathbf{I} - (z_I - \frac{\theta}{2} z_E^2)\mathbf{Q}_\Delta\right)\mathbf{u}^{k+1} = \mathbf{u}_0 + (z_I + z_E)Q\mathbf{u}^k - (z_I - \frac{\theta}{2} z_E^2)\mathbf{Q}_\Delta \mathbf{u}^k$$

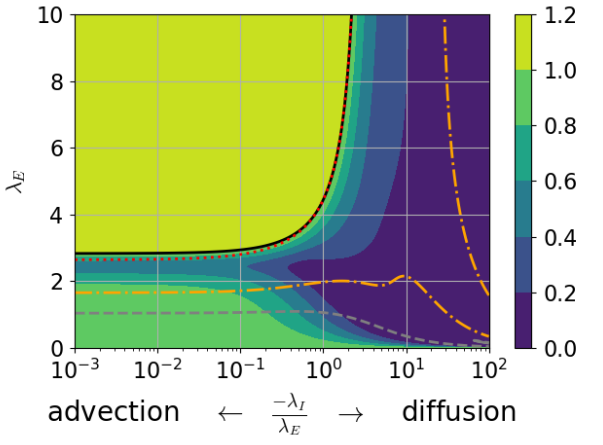
- used one major advantage of SDC : you can put **anything** in the preconditionner
- corresponds to subtracting a part of the advective term into the diffusive term
- loose a bit on accuracy, but gain a lot on numerical stability

[12] Stiller, 2025. *"Stable semi-implicit SDC methods for conservation laws"*.

Stabilized PIMEX SDC stability

SPIMEX SDC44, $\theta = 0.0$

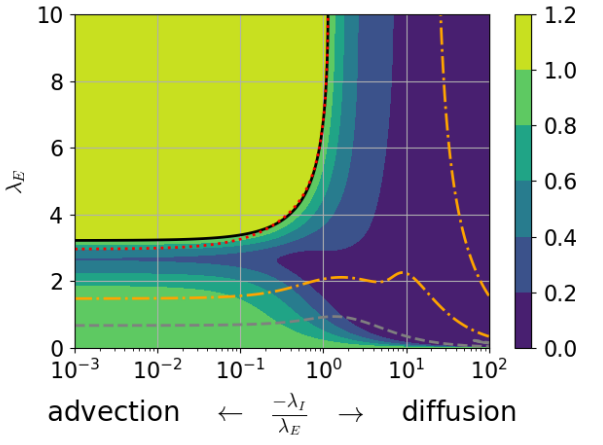
Maximum CFL for $-\lambda_I/\lambda_E \to 0$
$\Rightarrow$ $CFL_{max} = 2.83$ ($\times$ 1.8 vs RK443)



Stabilized PIMEX SDC stability

SPIMEX SDC44, $\theta = 0.1$

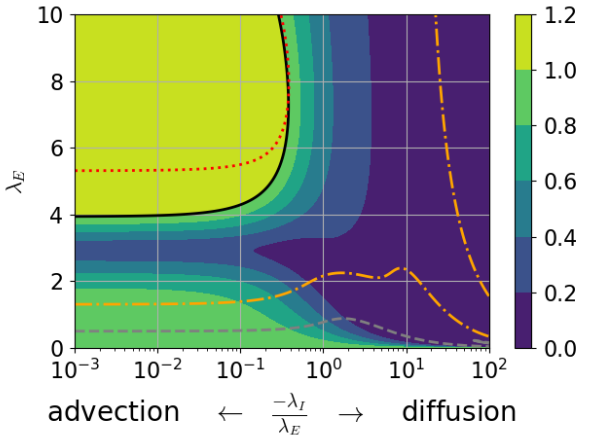
Maximum CFL for $-\lambda_I/\lambda_E \to 0$
$\Rightarrow$ $CFL_{max} = 3.22$ ($\times$ 2.1 vs RK443)



Stabilized PIMEX SDC stability

SPIMEX SDC44, $\theta = 0.2$

Maximum CFL for $-\lambda_I/\lambda_E \to 0$
$\Rightarrow$ $CFL_{max} = 3.94$ ($\times$ 2.5 vs RK443)



But why stop there ... we still can

- optimize diagonal coefficients for IMEX
- combine with multi-level or PFASST based approaches (second PinT layer)
- try other crazy ideas ...

Pandora's box still not fully opened yet !

Conclusions

TUHH
Hamburg University of Technology

SDC is a generic framework to solve ODEs, with

1. high design flexibility and combination potential (multi-level / multi-scale, ...)
2. higher cost for better accuracy and numerical stability
3. natural direct time-parallelism with problem-independent speedup

Conclusions

TUHH
Hamburg University of Technology

SDC is a generic framework to solve ODEs, with

1. high design flexibility and combination potential (multi-level / multi-scale, ...)
2. higher cost for better accuracy and numerical stability
3. natural direct time-parallelism with problem-independent speedup

But finding a new rare gem requires :

- more people knowing how it works, with some time to spend on it
- proper analysis to determine optimal SDC coefficients for different cases
- efficient generic implementations to show off on large scale non-linear problems

$\Rightarrow$ we need developments in Applied Mathematics & HPC & RSE

Conclusions

TUHH
Hamburg University of Technology

SDC is a generic framework to solve ODEs, with

1. high design flexibility and combination potential (multi-level / multi-scale, ...)
2. higher cost for better accuracy and numerical stability
3. natural direct time-parallelism with problem-independent speedup

But finding a new rare gem requires :

- more people knowing how it works, with some time to spend on it
- proper analysis to determine optimal SDC coefficients for different cases
- efficient generic implementations to show off on large scale non-linear problems

$\Rightarrow$ we need developments in Applied Mathematics & HPC & RSE

as Gear / Butcher said : lot more to do ...

SDC Days 2026 : gather people who are interested in SDC-based methods, or related time-integration in general ...

- 2022 : Hamburg (TUHH)
- 2023 : Jülich Supercomputing Center
- 2024 : Grenoble (INRIA)
- 2025 : Dresden (TUD)

$\Rightarrow$ next one around **16th December 2026 at TUHH**

more details to come ...