Noname manuscript No.
(will be inserted by the editor)

Parallelizing spectral deferred corrections across the method

Robert Speck

Received: date / Revised version: date

Abstract In this paper we present two strategies to enable "parallelization across the method" for spectral deferred corrections (SDC). Using standard low-order time-stepping methods in an iterative fashion, SDC can be seen as preconditioned Picard iteration for the collocation problem. Typically, a serial Gauß-Seidel-like preconditioner is used, computing updates for each collocation node one by one. The goal of this paper is to show how this process can be parallelized, so that all collocation nodes are updated simultaneously. The first strategy aims at finding parallel preconditioners for the Picard iteration and we test three choices using four different test problems. For the second strategy we diagonalize the quadrature matrix of the collocation problem directly. In order to integrate non-linear problems we employ simplified and inexact Newton methods. Here, we estimate the speed of convergence depending on the time-step size and verify our results using a non-linear diffusion problem.

Keywords Spectral deferred corrections, parallel-in-time integration, preconditioning, simplified Newton

1 Introduction

Implicit integration methods based on collocation are an attractive approach to solve initial value problems numerically. Depending on the choice of the collocation or quadrature nodes, they feature near-ideal or even ideal (for Gauß-Legendre nodes) convergence orders and typically have very advantageous stability properties. However, solving the dense and fully coupled collocation problem directly is prohibitively expensive in most cases: For M collocation nodes and an N-dimensional system of ordinary differential equations (ODEs), a system of size $MN \times MN$ has to be solved. Thus, an iterative strategy is favorable, where instead of the full system only M smaller systems of size $N \times N$ need to be solved for each iteration.

Such an approach is given by the so-called "spectral deferred correction methods" (SDC), introduced in [5]. After casting the initial value problem into its Picard form, a provisional solution to the integral problem is computed using a standard time-stepping method, typically the explicit or the implicit Euler scheme. Then, this provisional solution is corrected using a sequence of error integral equations, which are also solved using one of the standard methods. This way, a higher-order time-stepping method can be obtained simply by using low-order methods repeatedly. Xia et al. showed in [29] that each iteration or "sweep" of SDC can raise the order by one up to the order of the underlying collocation formula. The iterative structure of SDC has been proven to provide many opportunities for algorithmic and mathematical improvements: convergence can be accelerated by GMRES [11], IMEX splitting with high orders of accuracy is possible [19,21] and work can be shifted to coarser, less expensive levels to improve the efficiency of SDC [23]. In the last decade, SDC has been applied e.g. to gas dynamics and incompressible or reactive flows [1,17,20] as well as to fast-wave slow-wave problems [21] or particle dynamics [28].

One of the key features of such an iterative approach for time-stepping, though, is that these approaches can be used to enable efficient parallel-in-time integration. Using SDC, the "parallel full approximations scheme in

R. Speck, r.speck@fz-juelich.de
Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH

space and time" (PFASST) by Emmett and Minion [6] allows to integrate multiple time-steps simultaneously by using SDC sweeps on a space-time hierarchy. This "parallelization across the steps" approach [3] targets large-scale parallelization on top of saturated spatial parallelization of partial differential equations (PDEs), where parallelization in the temporal domain acts as a multiplier for standard parallelization techniques in space. In contrast, "parallelization across the method" approaches [3] try to parallelize the integration of each time-step individually. While this typically results in small-scale parallelization in the time-domain, parallel efficiency and applicability of these methods are often more favorable. Most notable, the "revisionist integral deferred correction method" (RIDC) by Christlieb et al. [4] makes use of integral deferred corrections (which are indeed closely related to SDC) in order to compute multiple iterations in a pipelined way. Also, in [13, 26, 10] parallel Runge-Kutta methods were investigated and we refer to [2] for more examples.

In this paper, we present two approaches to parallelize SDC across the method, allowing to compute the update for all collocation nodes simultaneously. First, we make use of parallel preconditioners by rewriting SDC as preconditioned Picard iteration, following the ideas of [11, 27, 21]. We explore ideas appearing in the context of Runge-Kutta methods [26, 10] and investigate three different choices, all of which enable parallelization across the nodes. Second, we diagonalize the quadrature matrix of the collocation problem and show how this idea can be extended to non-linear problems using simplified and inexact Newton methods [14]. We estimate the speed of convergence depending on the time-step size and verify our results using a non-linear diffusion problem. This second approach is closely related to the diagonalization-based parallelization strategy presented independently in [8], which uses this technique to enable larger-scale parallelization across the steps. We note that while methods like PFASST target distributed-memory parallelization, both approaches presented here are best implemented using shared-memory parallelization.

2 Spectral deferred corrections

For ease of notation we consider a scalar initial value problem

$$u_t = f(u), \quad u(0) = u_0$$

with $u(t), u_0, f(u) \in \mathbb{R}$. For an interval $[t_0, t_1]$, we rewrite this in Picard formulation as

$$u(t) = u_0 + \int_{t_0}^{t} f(u(s))ds, \ t \in [t_0, t_1],$$

Introducing M quadrature nodes $\tau_1, ..., \tau_M$ with $t_l \leq \tau_1 < ... < \tau_M = t_{l+1}$, we can approximate the integrals from t_l to these nodes τ_m using spectral quadrature like Gauß-Radau or Gauß-Lobatto quadrature, such that

$$u_m = u_0 + \sum_{j=1}^{M} q_{m,j} f(u_j), \quad m = 1, ..., M,$$

where $u_m \approx u(\tau_m)$, $\Delta t = t_1 - t_0$ and $q_{m,j}$ represent the quadrature weights for the interval $[t_0, \tau_m]$ with

$$\sum_{j=1}^{M} q_{m,j} f(u_j) \approx \int_{t_0}^{\tau_m} f(u(s))ds.$$

We can now combine these M equations into one system of linear or non-linear equations with

$$(\mathbf{I} - \Delta t \mathbf{Q} \boldsymbol{F})(\boldsymbol{u}) = \boldsymbol{u}_0 \tag{1}$$

where $\boldsymbol{u} = (u_1, ..., u_M)^T \approx (u(\tau_1), ..., u(\tau_M))^T \in \mathbb{R}^M$, $\boldsymbol{u}_0 = (u_0, ..., u_0)^T \in \mathbb{R}^M$, $\mathbf{Q} = (q_{ij})_{i,j} \in \mathbb{R}^{M \times M}$ is the matrix gathering the quadrature weights and the vector function $\boldsymbol{F}$ is given by $\boldsymbol{F}(\boldsymbol{u}) = (f(u_1), ..., f(u_M))^T \in \mathbb{R}^M$. This system of equations is called the "collocation problem" and it is equivalent to a fully implicit Runge-Kutta method, where the matrix $\mathbf{Q}$ contains the entries of the corresponding Butcher tableau. We note that for $f(u) \in \mathbb{R}^N$, we need to replace $\mathbf{Q}$ by $\mathbf{Q} \otimes \mathbf{I}_N$.

This system of equations is dense and a direct solution is not advisable, in particular if the right-hand side of the ODE is non-linear. Using SDC, this problem can be solved iteratively and we follow [11, 27, 21] to present SDC as preconditioned Picard iteration for the collocation problem (1). Standard discretized Picard iteration is given by

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

for $k = 0, ...K$. This is simply a unmodified, non-linear Richardson iteration for (1) and for very small Δt, this indeed converges to the solution of (1). In order to increase range and speed of convergence, we now precondition this iteration. The standard approach to preconditioning is to define an operator $\mathbf{P}$ which is easy to invert but also close to the operator of the system. For SDC, we now choose a simpler quadrature rule for the preconditioner. In particular, the resulting matrix $\mathbf{Q}_\Delta$ gathering the weights of this rule is a lower triangular

matrix, such that solving the system can be easily done by forward substitution. We write

$$(\mathbf{I} - \Delta t \mathbf{Q}_\Delta \boldsymbol{F})\,(\boldsymbol{u}^{k+1}) = \boldsymbol{u}_0 + \Delta t(\mathbf{Q} - \mathbf{Q}_\Delta)\boldsymbol{F}(\boldsymbol{u}^k) \quad (2)$$

and the operator $\mathbf{I} - \Delta t \mathbf{Q}_\Delta \boldsymbol{F}$ is then called the SDC preconditioner. The matrix $\mathbf{Q}_\Delta$ is typically given by the implicit Euler method which corresponds to the right-hand side rule in terms of integration with

$$\mathbf{Q}_\Delta^{\mathrm{IE}} = \frac{1}{\Delta t}\begin{pmatrix} \Delta\tau_1 & & \\ \Delta\tau_1 & \Delta\tau_2 & \\ \vdots & \vdots & \\ \Delta\tau_1 & \Delta\tau_2 & \dots\ \Delta\tau_M \end{pmatrix},$$

where $\Delta\tau_m = \tau_m - \tau_{m-1}$ for $m = 2, ..., M$ and $\Delta\tau_1 = \tau_1 - t_0$ or, using the LU decomposition of $\mathbf{Q}^T$ [27], by

$$\mathbf{Q}_\Delta^{\mathrm{LU}} = \mathbf{U}^{\mathbf{T}} \quad \text{for} \quad \mathbf{Q}^T = \mathbf{L}\mathbf{U}.$$

This choice, sometimes also called the LU trick, is very well suited for stiff problems and has become the de-facto standard choice for $\mathbf{Q}_\Delta$.

Yet, common to these and most other choices is the fact that $\mathbf{Q}_\Delta$ is a lower triangular matrix, so that solving (2) can be done only in a serial, Gauß-Seidel-like way: first solve for u_1^{k+1} using the initial value u_0, then for u_2^{k+1} using u_1^{k+1} and so on. In order to introduce parallelism across the quadrature nodes, we investigate two strategies in the following: (A) choose a parallel preconditioner and (B) diagonalize the quadrature matrix.

3 Parallel preconditioning

The first idea to parallelize SDC over the quadrature nodes is quite obvious: instead of following a Gauß-Seidel-like approach, we try to find suitable matrices $\mathbf{Q}_\Delta$ which only have entries on the the diagonal, i.e. which allow to follow a Jacobian-like approach. To this end, we identify three candidates:

1. take the diagonal of $\mathbf{Q}$, i.e.

$$\mathbf{Q}_\Delta^{\mathrm{Qpar}} = \mathrm{diag}(q_{11}, ..., q_{mm}, ..., q_{MM}), \quad (3)$$

2. use Euler steps from t_0 to τ_m, i.e.

$$\mathbf{Q}_\Delta^{\mathrm{IEpar}} = \mathrm{diag}(\tau_1 - t_0, ..., \tau_m - t_0, ..., \tau_M - t_0), \quad (4)$$

3. minimize the spectral radius of $\mathbf{I} - \mathbf{Q}_\Delta^{-1}\mathbf{Q}$, i.e.

$$\mathbf{Q}_\Delta^{\mathrm{MIN}} = \mathrm{diag}(\hat{\boldsymbol{q}}) \quad (5)$$

with

$$\hat{\boldsymbol{q}} = \mathrm{argmin}_{\boldsymbol{q}\in\mathbb{R}^M}\ \rho(\mathbf{I} - \mathrm{diag}(\boldsymbol{q})\mathbf{Q})$$

While the first two approaches are obvious candidates and straightforward to compute, the third one is more involved. For the linear test problem $u_t = \lambda u$, SDC has an iteration matrix $\mathbf{K}$ with

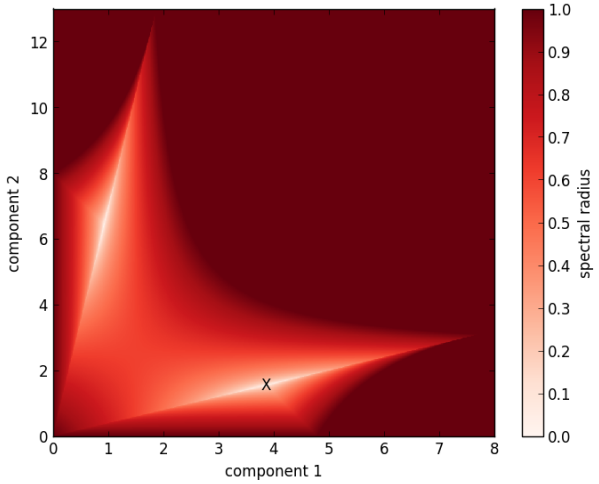
$$\mathbf{K} = \lambda\Delta t\mathbf{Q}_\Delta\left(\mathbf{I} - \lambda\Delta t\mathbf{Q}_\Delta\right)^{-1}\left(\mathbf{Q}_\Delta^{-1}\mathbf{Q} - \mathbf{I}\right),$$

see e.g. [21]. While the first factors all depend on the "space"-problem parameter λ the last factor does not and can therefore be modified independently of the spatial problem at hand. It also corresponds (up to the sign) to the stiff limit of the iteration matrix, i.e. for $|\lambda\Delta t| \to \infty$ we have $\mathbf{K} \to \mathbf{I} - \mathbf{Q}_\Delta^{-1}\mathbf{Q}$, see [21]. We choose to minimize the spectral radius, because the hope is that in this case strong damping of the stiff iteration error components is achieved, see [10] for more details on this matter.

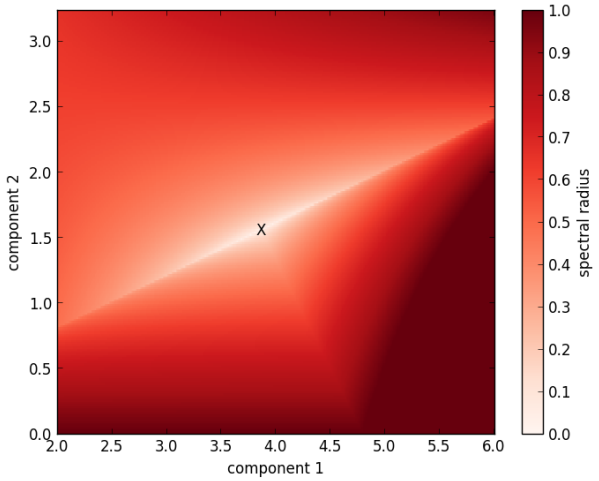
However, to the best of our knowledge there exists no analytic expression for the eigenvalues of $\mathbf{Q}$ or $\mathrm{diag}(\boldsymbol{q})\mathbf{Q}$, so that the computation of this minimizer has to be done numerically. We use the Nelder-Mead algorithm as implemented by SciPy v.0.18.1 [15] in the "optimize" package.

To get a first impression of this approach, we consider $M = 2$ Gauß-Radau nodes and investigate the values of $\rho(\mathbf{I} - \mathrm{diag}(\boldsymbol{q})\mathbf{Q})$ for different $\boldsymbol{q} = (q_1, q_2)^T$. In Figure 1 we let component 1 (i.e. q_1) vary between 0 and 8 and component 2 (i.e. q_2) between 0 and 13. Further experiments not shown here revealed that outside of this region the spectral radius is greater than 1. Figure 1a shows that there are two regions of interest shown in light colors, where a minimum can be expected. For Figure 1b, we zoom into the lower region, where the local minimum value computed by the Nelder-Mead optimization with starting value $\boldsymbol{q}_0 = (1, 1)^T$ is located. Note that for e.g. $\boldsymbol{q}_0 = (1, 2)^T$ the upper local minimum is found, which is about 2.5 times smaller than the one from the lower region ($2.6 \cdot 10^{-5}$ vs. $6.5 \cdot 10^{-5}$). Already for 2 nodes we can see that the regions of small spectral radii as well as the location of the minima are far from trivial.

Now, what is the best choice for $\mathbf{Q}_\Delta$? "Best" in our context means that for this choice the corresponding SDC iteration converges about as fast as the standard choices $\mathbf{Q}_\Delta^{\mathrm{IE}}$ and $\mathbf{Q}_\Delta^{\mathrm{LU}}$ for the problem at hand. Clearly, this is highly problem dependent, but even worse, the same argument which prevented us from finding the minimizer of the spectral radius analytically apply also to finding the best diagonal matrix $\mathbf{Q}_\Delta$, since there is no closed form of the eigenvalues or the norm of any matrix related to $\mathbf{Q}$. Thus, we choose four test problems, two linear and two non-linear, to quantify the impact of the three different $\mathbf{Q}_\Delta$:



(a) Full region



(b) Zoom in

Fig. 1: Spectral radius of $\mathbf{I}_2 - \mathbf{Q}_\Delta^{-1}\mathbf{Q}$ for a diagonal matrix $\mathbf{Q}_\Delta \in \mathbb{R}^{2\times 2}$ consisting of different components 1 (x-axis) and 2 (y-axis). The X indicates the result of the Nelder-Mead optimization. Note that both axes are scaled differently. Upper: Full domain, lower: zoom into a region of interest.

Problem A Heat equation with $\nu > 0$:

$$\begin{aligned} u_t &= \nu \Delta u \quad \text{on } [0,1]\times[0,T], \\ u(0,t) &= 0,\ u(1,t) = 0, \\ u(x,0) &= \sin(2\pi x) \end{aligned}$$

Problem B Advection equation with $c > 0$:

$$\begin{aligned} u_t &= c\nabla u \quad \text{on } [0,1]\times[0,T], \\ u(0,t) &= u(1,t), \\ u(x,0) &= \sin(2\pi x) \end{aligned}$$

Problem C Van der Pol oscillator with $\mu > 0$:

$$\begin{aligned} u_t = v,\ v_t = \mu(1-u^2)v - u \quad \text{on } [0,T], \\ u(0) = 2,\ v(0) = 0 \end{aligned}$$

Problem D Non-linear diffusion of Kolmogorov-Petrovskii-Piskunov type [7] with $\lambda_0 > 0$:

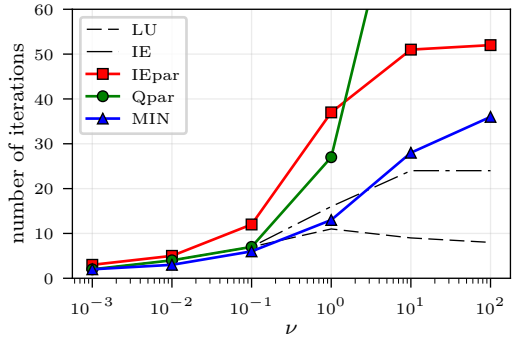
$$\begin{aligned} u_t &= \Delta u + \lambda_0^2 u(1-u^\nu) \quad \text{on } \mathbb{R}\times[0,T], \\ u(x,0) &= \left(1 + (2^{\nu/2}-1)e^{-(\nu/2)\delta x}\right)^{-\frac{2}{\nu}} \end{aligned} \tag{6}$$

and constants $\delta > 0$ and $\nu \in \mathbb{N}$ ($\nu = 1$ is used here, δ can be found in [7]).

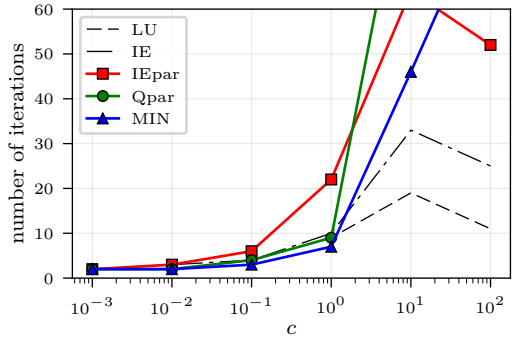
For all runs we choose $M = 3$ Gauß-Radau nodes, $T = \Delta t = 0.1$ and a residual tolerance of 10^{-8}. For Problems A and D we choose finite differences with $N = 63$ degrees of freedom and for B $N = 64$, again using finite differences.

In Figures 2a-2d we show the number of iterations for five different choices of $\mathbf{Q}_\Delta$: the implicit Euler and the LU trick as references as well as the three diagonal matrices defined in (3),(4) and (5). For each problem, we vary the parameter given in the description above to change the characteristics or stiffness of the problem. The first thing to notice is that in almost all cases $\mathbf{Q}_\Delta^{LU}$ is the best choice, in particular if for a problem the convergence of SDC is required to be roughly the same across all parameter values. For non-stiff problem, i.e. for small values of the parameters, however, the diagonal matrices work equally well. Especially $\mathbf{Q}_\Delta^{\text{MIN}}$ is capable of yielding convergence as fast or sometimes even faster than $\mathbf{Q}_\Delta^{\text{LU}}$, if the parameter is small enough. The other choices, namely $\mathbf{Q}_\Delta^{\text{IEpar}}$ and $\mathbf{Q}_\Delta^{\text{Qpar}}$ perform reasonably well for small parameters, too, but they lead to drastically increased numbers of iterations for larger parameters. In this regime, the LU trick shows its strength and, not surprisinlgy, this is precisely the way it was designed [27]. Only for the two non-linear problems and most notably for Problem D we see that $\mathbf{Q}_\Delta^{\text{MIN}}$ is a favorable choice. We finally note that results look very similar and in parts even slighty better for $M = 5$ nodes.

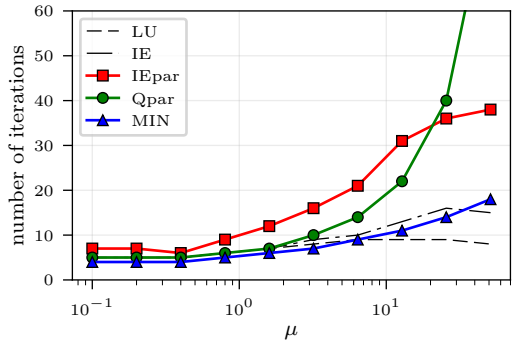
The numerical comparison of the five different candidates for $\mathbf{Q}_\Delta$, two leading to serial and three to parallel SDC iterations, shows no clear result. The minimization-based approach $\mathbf{Q}_\Delta^{\text{MIN}}$ seems to be a reliable and sometimes even favorable choice, but only for the scenarios tested here. Due to the intricate structure of $\mathbf{Q}$ there is no adequate mathematical theory to guide the selection of the $\mathbf{Q}_\Delta$ matrix. Even for a particular choice, estimating convergence speed with reasonable



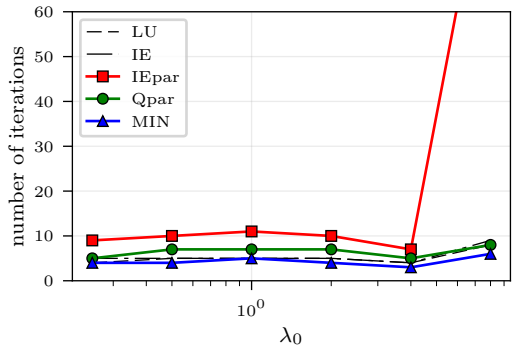
(a) Heat equation



(b) Advection equation



(c) Van der Pol oscillator



(d) Non-linear diffusion

Fig. 2: Number of iterations for five different choices of $\mathbf{Q}_\Delta$ for the four test problems A-D and varying parameters. The serial Gauß-Seidel-type matrices for the implicit Euler and the LU trick are shown as reference.

sharp bounds is not straightforward. However, this section shows that in many cases parallel SDC iterations are indeed possible and can be implemented without any overhead. Note that this strategy is only reasonable to use for shared-memory parallelization: in each iteration, the system (2) is solved in parallel, but in order to compute the right-hand side, each compute unit (core, threads etc.) needs to have data from all M quadrature nodes. If the collocation problem is large, e.g. due to a large amount of degrees-of-freedom N in space, each unit has to receive the full vector $\boldsymbol{u}^k$, consisting of MN variables. If distributed-memory parallelization were used here, communication costs would be prohibitively high.

4 Diagonalization of Q

While choosing a parallel preconditioner is a simple but rather heuristical idea, the second approach we describe here is more intricate. In order to introduce parallelism across the nodes we now look at the diagonalization of the quadrature matrix $\mathbf{Q}$. To this end, we write

$$\mathbf{Q} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1}$$

where $\mathbf{\Lambda} = \text{diag}(\lambda_1(\mathbf{Q}), ..., \lambda_M(\mathbf{Q}))^T$ is a diagonal matrix with eigenvalues $\lambda_i(\mathbf{Q}) \in \mathbb{C}$ of $\mathbf{Q}$ on the diagonal and $\mathbf{V}$ contains the eigenvectors of $\mathbf{Q}$. For non-symmetric quadrature rules such as Gauß-Radau, this diagonalization of $\mathbf{Q}$ is possible, since all eigenvalues are distinct.

If f is linear, i.e. if $f(u) = au$, then this leads to a parallel direct solver of the collocation problem (1). We have

$$\begin{aligned}(\mathbf{I} - \Delta t \mathbf{Q}\boldsymbol{F})(\boldsymbol{u}) &= (\mathbf{I} - a\Delta t\mathbf{Q})\,\boldsymbol{u} \\ &= \mathbf{V}(\mathbf{I} - a\Delta t\mathbf{\Lambda})\,\mathbf{V}^{-1}\boldsymbol{u}\end{aligned} \tag{7}$$

so that (1) can be solved in three simple steps:

1. replace $\boldsymbol{u}_0$ by $\tilde{\boldsymbol{u}}_0 = \mathbf{V}^{-1}\boldsymbol{u}_0$ (serial)
2. solve $(\mathbf{I} - a\Delta t\mathbf{\Lambda})\,\tilde{\boldsymbol{u}} = \tilde{\boldsymbol{u}}_0$ (parallel in M)
3. replace $\tilde{\boldsymbol{u}}$ by $\boldsymbol{u} = \mathbf{V}\tilde{\boldsymbol{u}}$ (serial)

Since $\mathbf{\Lambda}$ is a diagonal matrix, step 2 can be done in parallel for all M quadrature nodes at once. Steps 1 and 3, in contrast, require global communication: While $\mathbf{V}$ and $\mathbf{V}^{-1}$ can be precomputed, the multiplications with these dense matrices require all-to-all exchanges of M-times the spatial data. Thus, for problems with many degrees-of-freedom in space (i.e. for N large), this results in significant communication costs. We therefore consider this strategy only suitable for shared-memory parallelization as well, which plays nicely with the rather

small number M of quadrature nodes used in typical applications. The overhead of this approach then merely consists of the two multiplications with $\mathbf{V}$ and $\mathbf{V}^{-1}$, which is negligible compared to evaluations of right-hand sides and solutions of implicit systems.

Yet, the more severe restriction comes from the linearity of the right-hand side f. If f is non-linear, we cannot follow (7) to solve the collocation problem in parallel. This is due to the coupling of the non-linear $\boldsymbol{F}(\boldsymbol{u})$ with the matrix $\mathbf{Q}$, which does not allow the $\mathbf{V}$ to be extracted. The obvious way to proceed is to linearize the problem using Newton's method. We define

$$\boldsymbol{G}(\boldsymbol{u}) = \boldsymbol{u} - \Delta t \mathbf{Q}\boldsymbol{F}(\boldsymbol{u}) - \boldsymbol{u}_0$$

so that solving (1) is equivalent to finding a root of $\boldsymbol{G}(\boldsymbol{u})$. For Newton iterations, we need the Jacobian $\boldsymbol{J_G}$ of $\boldsymbol{G}$, which is given by

$$\boldsymbol{J_G}(\boldsymbol{u}) = \mathbf{I} - \Delta t \mathbf{Q}\boldsymbol{J_F}(\boldsymbol{u})$$

with

$$\boldsymbol{J_F}(\boldsymbol{u}) = \mathrm{diag}\left(f'(u_1), ..., f'(u_M)\right) \in \mathbb{R}^{M\times N}. \tag{8}$$

Then, the Newton iteration is given by

$$\begin{aligned} \boldsymbol{J_G}(\boldsymbol{u}^k)\boldsymbol{e}^k &= -\boldsymbol{G}(\boldsymbol{u}^k), \\ \boldsymbol{u}^{k+1} &= \boldsymbol{u}^k + \boldsymbol{e}^k. \end{aligned} \tag{9}$$

The matrix $\boldsymbol{J_G}(\boldsymbol{u}^k)$ looks very much like the original operator of the collocation problem (1) for a linear function. However, the matrix $\boldsymbol{J_F}(\boldsymbol{u}^k)$ which appears in this problem still does not decouple from the matrix $\mathbf{Q}$, since for each quadrature node a different entry is given. Again, $\mathbf{V}$ cannot be extracted and we need another step to obtain parallelism via diagonalization.

We note that for $k = 0$ we have

$$\begin{aligned} \boldsymbol{J_F}(\boldsymbol{u}^0) &= \mathrm{diag}\left(f'(u_1^0), ..., f'(u_M^0)\right) \\ &= \mathrm{diag}\left(f'(u_0^0), ..., f'(u_0^0)\right) = f'(u_0)\mathbf{I}_M \end{aligned}$$

if the iteration is started with $\boldsymbol{u}_0$ as initial guess. This directly leads to a simplified Newton method with

$$\begin{aligned} \boldsymbol{J_G}(\boldsymbol{u}^0)\boldsymbol{e}^k &= -\boldsymbol{G}(\boldsymbol{u}^k), \\ \boldsymbol{u}^{k+1} &= \boldsymbol{u}^k + \boldsymbol{e}^k. \end{aligned} \tag{10}$$

where

$$\boldsymbol{J_G}(\boldsymbol{u}^0) = \mathbf{I} - f'(u_0)\Delta t\mathbf{Q} \tag{11}$$

and here we indeed can use diagonalization of $\mathbf{Q}$ for parallelization across M nodes. For each iteration k with a given iterate $\boldsymbol{u}^k$, the algorithm consists of these four steps:

1. replace $\boldsymbol{r}^k = -\boldsymbol{G}(\boldsymbol{u}^k)$ by $\tilde{\boldsymbol{r}}^k = -\mathbf{V}^{-1}\boldsymbol{G}(\boldsymbol{u}^k)$ (serial)
2. solve $(\mathbf{I} - f'(u_0)\Delta t\boldsymbol{\Lambda})\,\tilde{\boldsymbol{e}}^k = \tilde{\boldsymbol{r}}^k$ (parallel in M)
3. replace $\tilde{\boldsymbol{e}}^k$ by $\boldsymbol{e}^k = \mathbf{V}\tilde{\boldsymbol{e}}^k$ (serial)
4. set $\boldsymbol{u}^{k+1} = \boldsymbol{u}^k + \boldsymbol{e}^k$ (parallel in M)

We note that using simplified Newton methods for fully-implicit Runge-Kutta schemes like (1) is a standard way of solving these systems, see e.g. Section IV.8 in [9].

The price for using diagonalization of $\mathbf{Q}$ for parallelization is therefore the re-introduction of an iterative process as well as the need for the Jacobian of the right-hand side. The latter, at least, has to be computed only once per time-step. While for linear problems we obtain a parallel direct solver, a simplified Newton approach is required to obtain the same level of parallelism for non-linear problems. The question now is, how much the approximation of the Jacobian $\boldsymbol{J_G}(\boldsymbol{u}^k)$ by $\boldsymbol{J_G}(\boldsymbol{u}^0)$ affects the convergence of the method and how this compares to standard SDC iterations. In contrast to the previous section, we are now able to investigate this not only numerically but also on an analytic level. It is well known that for suitable right-hand sides and initial guesses the standard, unmodified Newton method converges quadratically while the simplified Newton method only shows linear convergence, see e.g. [16, 12]. Yet, we are also interested in the constants and their dependence on the time-step size Δt. More precisely, we can show the following result.

Theorem 1 *Let f' be Lipschitz continuous with constant γ_f and $\boldsymbol{u}_0, \boldsymbol{u}^k \in B(\Delta t) = \{\boldsymbol{u} \in \mathbb{R}^M : \|\boldsymbol{u} - \boldsymbol{u}^*\| \leq c_1\Delta t\}$ for the exact solution $\boldsymbol{u}^*$ of the collocation problem* (1) *and the current iterate $\boldsymbol{u}^k$. Furthermore, assume that $\boldsymbol{J_G}(\boldsymbol{u}^*)$ is non-singular. Then the simplified Newton iteration* (10) *converges with*

$$\|\boldsymbol{e}^{k+1}\|_\infty \leq c\Delta t^2 \|\boldsymbol{e}^k\|_\infty$$

if Δt is small enough.

Proof It is tempting to simply follow Theorem 5.4.2 in [16] for this proof: this states that the "chord" or simplified Newton method converges linearly to the exact solution $\boldsymbol{u}^*$, if the standard assumptions are satisfied (see 4.3 in [16]), namely if $\boldsymbol{u}^*$ is a solution of $\boldsymbol{G}(\boldsymbol{u}) = 0$, $\boldsymbol{J_G}$ is Lipschitz continuous and $\boldsymbol{J_G}(\boldsymbol{u}^*)$ is non-singular. We first note that these standard assumptions are satisfied by the conditions we require here. In particular, $\boldsymbol{J_G}$ is Lipschitz continuous with constant $\Delta t\gamma_f\|\mathbf{Q}\|_\infty$, because

$$\begin{aligned} \|\boldsymbol{J_G}(\boldsymbol{u}) - \boldsymbol{J_G}(\boldsymbol{v})\|_\infty &\leq \Delta t\|\mathbf{Q}\|_\infty\|\boldsymbol{J_F}(\boldsymbol{u}) - \boldsymbol{J_F}(\boldsymbol{v})\|_\infty \\ &\leq \Delta t\gamma_f\|\mathbf{Q}\|_\infty\|\boldsymbol{u} - \boldsymbol{v}\|_\infty. \end{aligned} \tag{12}$$

For the last inequality we note that

$$\begin{aligned}&\|\boldsymbol{J_F}(\boldsymbol{u}) - \boldsymbol{J_F}(\boldsymbol{v})\|_\infty \\ =& \max_{m=1,\dots,M} \left| \frac{\partial f}{\partial u_m}(u_m) - \frac{\partial f}{\partial u_m}(v_m) \right| \\ \leq& \gamma_f \max_{m=1,\dots,M} |u_m - v_m| = \gamma_f \|\boldsymbol{u} - \boldsymbol{v}\|_\infty.\end{aligned}$$

However, using the estimate given in this theorem with the constants we have here would provide an estimate which is only linear in Δt and therefore too pessimistic. To overcome this, we look at the more technical Theorem 5.4.1 in [16], stating that for inaccurately computed $\boldsymbol{G}$ and $\boldsymbol{J_G}$ the iteration error $\boldsymbol{e}^{k+1}$ can be estimated by a linear combination of the previous error $\boldsymbol{e}^k$, the error in the Jacobian $\boldsymbol{J_G}$ as well as the error in the function $\boldsymbol{G}$. Here, the error in the Jacobian is simply the difference between $\boldsymbol{J_G}(\boldsymbol{u}^k)$ and $\boldsymbol{J_G}(\boldsymbol{u}^0)$ and we have with (12)

$$\begin{aligned}\left\|\boldsymbol{J_G}(\boldsymbol{u}^0) - \boldsymbol{J_G}(\boldsymbol{u}^k)\right\|_\infty &\leq \Delta t \gamma_f \|\mathbf{Q}\|_\infty \|\boldsymbol{u}_0 - \boldsymbol{u}^k\|_\infty \\ &\leq c_1 \Delta t^2,\end{aligned} \tag{13}$$

since $\boldsymbol{u}^k$ and $\boldsymbol{u}^0$ are both in $B(\Delta t)$. The function $\boldsymbol{G}$ is evaluated exactly, so that the error in $\boldsymbol{G}$ is just zero. Then, looking at the proof of Theorem 5.4.1, the last estimate reads in our notation

$$\|\boldsymbol{e}^{k+1}\|_\infty \leq \gamma_f \Delta t \left\|\boldsymbol{e}^k\right\|_\infty^2 + 16P \left\|\boldsymbol{e}^k\right\|_\infty + 0. \tag{14}$$

for

$$\begin{aligned}P = \left\|\boldsymbol{J_G}(\boldsymbol{u}^*)^{-1}\right\|_\infty^2 \|\boldsymbol{J_G}(\boldsymbol{u}^*)\|_\infty \cdot \\ \left\|\boldsymbol{J_G}(\boldsymbol{u}^0) - \boldsymbol{J_G}(\boldsymbol{u}^k)\right\|_\infty.\end{aligned}$$

For Δt small enough and by assumption, we can bound both $\left\|\boldsymbol{J_G}(\boldsymbol{u}^*)^{-1}\right\|_\infty^2$ and $\|\boldsymbol{J_G}(\boldsymbol{u}^*)\|_\infty$ by some constant c_2. Also, we note that

$$\left\|\boldsymbol{e}^k\right\|_\infty^2 \leq \Delta t \left\|\boldsymbol{e}^k\right\|_\infty.$$

Then, putting all the results together Inequality (14) yields

$$\begin{aligned}\|\boldsymbol{e}^{k+1}\|_\infty &\leq \gamma_f \Delta t^2 \left\|\boldsymbol{e}^k\right\|_\infty + 16 c_1 c_2 \Delta t^2 \left\|\boldsymbol{e}^k\right\|_\infty \\ &= c \Delta t^2 \left\|\boldsymbol{e}^k\right\|_\infty\end{aligned}$$

which concludes the proof. □

This shows that the simplified Newton iteration converges linearly with a contraction factor of the order of $\mathcal{O}(\Delta t^2)$.

An objection one might raise is that the eigenvalues of $\mathbf{Q}$ and thus the entries of $\mathbf{\Lambda}$ are complex, making implementations slightly more cumbersome. To avoid this, we can "borrow" the preconditioning idea from SDC, i.e. instead of using the simplified Newton method with Eq. (11), we use

$$\boldsymbol{J}_{\boldsymbol{G}}^{\Delta}(\boldsymbol{u}^0) = \mathbf{I} - f'(u_0)\Delta t \mathbf{Q}_\Delta$$

so that the simplified Newton method becomes an inexact simplified Newton method:

$$\begin{aligned}\boldsymbol{J}_{\boldsymbol{G}}^{\Delta}(\boldsymbol{u}^0)\boldsymbol{e}^k &= -\boldsymbol{G}(\boldsymbol{u}^k), \\ \boldsymbol{u}^{k+1} &= \boldsymbol{u}^k + \boldsymbol{e}^k.\end{aligned} \tag{15}$$

Then, by diagonalizing $\mathbf{Q}_\Delta$ instead of $\mathbf{Q}$, we can parallelize each of these iterations across the nodes, too. Note that the diagonal part of $\mathbf{Q}_\Delta$ only has real entries, since all eigenvalues of the diagonal matrix $\mathbf{Q}_\Delta$ are real (and distinct), at least for typical choices of $\mathbf{Q}_\Delta$.

Theorem 2 *Let f' be Lipschitz continuous with constant γ_f and $\boldsymbol{u}_0, \boldsymbol{u}^k \in B(\Delta t) = \{\boldsymbol{u} \in \mathbb{R}^M : \|\boldsymbol{u} - \boldsymbol{u}^*\| \leq c_1 \Delta t\}$ for the exact solution $\boldsymbol{u}^*$ of the collocation problem* (1)*. Furthermore, assume that $\boldsymbol{J_G}(\boldsymbol{u}^*)$ and $\boldsymbol{J}_{\boldsymbol{G}}^{\Delta}(\boldsymbol{u}^0)$ are non-singular. Then the inexact simplified Newton iteration* (15) *converges with*

$$\|\boldsymbol{e}^{k+1}\|_\infty \leq c \Delta t \|\boldsymbol{e}^k\|_\infty$$

if Δt is small enough.

Proof We use again the final estimate of the proof of Theorem 5.4.1 in [16], which for

$$\tilde{P} = \left\|\boldsymbol{J_G}(\boldsymbol{u}^*)^{-1}\right\|_\infty^2 \|\boldsymbol{J_G}(\boldsymbol{u}^*)\|_\infty \left\|\boldsymbol{J}_{\boldsymbol{G}}^{\Delta}(\boldsymbol{u}^0) - \boldsymbol{J_G}(\boldsymbol{u}^k)\right\|_\infty$$

now reads

$$\|\boldsymbol{e}^{k+1}\|_\infty \leq \gamma_f \Delta t \left\|\boldsymbol{e}^k\right\|_\infty^2 + 16\tilde{P} \left\|\boldsymbol{e}^k\right\|_\infty + 0,$$

i.e. we simply replaced $\boldsymbol{J_G}(\boldsymbol{u}_0)$ by $\boldsymbol{J}_{\boldsymbol{G}}^{\Delta}(\boldsymbol{u}^0)$ in (14). Then, we have

$$\begin{aligned}\left\|\boldsymbol{J}_{\boldsymbol{G}}^{\Delta}(\boldsymbol{u}^0) - \boldsymbol{J_G}(\boldsymbol{u}^k)\right\|_\infty \leq& \left\|\boldsymbol{J}_{\boldsymbol{G}}^{\Delta}(\boldsymbol{u}^0) - \boldsymbol{J_G}(\boldsymbol{u}^0)\right\|_\infty \\ &+ \left\|\boldsymbol{J_G}(\boldsymbol{u}^0) - \boldsymbol{J_G}(\boldsymbol{u}^k)\right\|_\infty\end{aligned}$$

and we note that the second term is the same we had in (13). For the first term it is

$$\left\|\boldsymbol{J}_{\boldsymbol{G}}^{\Delta}(\boldsymbol{u}^0) - \boldsymbol{J_G}(\boldsymbol{u}^0)\right\|_\infty \leq \Delta t |f'(u_0)| \|\mathbf{Q} - \mathbf{Q}_\Delta\|_\infty$$

so that

$$\begin{aligned}\left\|\boldsymbol{J}_{\boldsymbol{G}}^{\Delta}(\boldsymbol{u}^0) - \boldsymbol{J_G}(\boldsymbol{u}^k)\right\|_\infty \leq& \Delta t |f'(u_0)| \|\mathbf{Q} - \mathbf{Q}_\Delta\|_\infty \\ &+ c_1 \Delta t^2,\end{aligned}$$

see (13). Therefore, we obtain

$$\begin{aligned}\|\boldsymbol{e}^{k+1}\|_\infty \leq& \gamma_f \Delta t^2 \left\|\boldsymbol{e}^k\right\|_\infty + 16 c_1 c_2 \Delta t^2 \left\|\boldsymbol{e}^k\right\|_\infty \\ &+ 16 c_2 c_3 \Delta t \|\mathbf{Q} - \mathbf{Q}_\Delta\|_\infty \left\|\boldsymbol{e}^k\right\|_\infty\end{aligned}$$

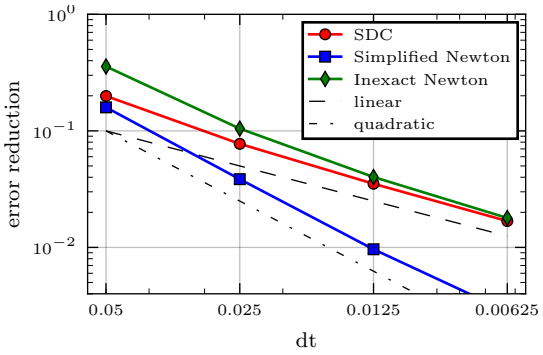


Fig. 3: Convergence rates of SDC, simplified Newton and inexact Newton for the non-linear diffusion problem D. Difference between the error at iteration 2 and 3 for the last time-step is shown.

so that in summary

$$\|e^{k+1}\|_\infty \leq c\Delta t \|e^k\|_\infty$$

which concludes the proof. □

Aside from constants, this is the same rate as the classical SDC convergence rate [25,21]. However, we can see in the last estimate of this proof that with the introduction of $\mathbf{Q}_\Delta$ the order of convergence in Δt becomes linear only because of the factor $\|\mathbf{Q} - \mathbf{Q}_\Delta\|_\infty$ which turns out to be small in many cases.

We test both theorems using the the non-linear diffusion equation, see Problem D in the previous section. For particular choices of δ and λ_0 this problem has an exact solution which we can use to evaluate the error. Note that instead of having boundaries at $\pm\infty$ we use the exact solution at the boundaries on a fixed interval $[a, b] = [-5, 5]$. We use $M = 5$ Gauß-Radau nodes with the LU trick of [27] as preconditioner $\mathbf{Q}_\Delta$, $T = 0.1$ with $2, ..., 16$ time-steps, finite differences with $N = 8191$ degrees-of-freedom in space and $\nu = 1$, $\lambda_0 = 5$ and δ given by the relation in [7]. For SDC, the implicit systems at each node are solved using a standard (spatial) Newton method with tolerance 10^{-12} while the linear systems for the Newton-like approaches are solved directly. In order to compute the rate of error reduction, we compute the ratio between the error on all quadrature nodes before and after iteration 2 at the last time-step. For SDC, the simplified Newton method as well as the inexact Newton method, these ratios are shown in Figure 3 for different time-step sizes Δt.

We can nicely observe how SDC converges linearly with Δt, while the simplified Newton method shows a quadratic dependency on the time step size. The inexact Newton method converges slightly faster than linear in Δt, but is still far away from a quadratic dependency. It has the largest ratio of all three methods, suggesting that the constants are higher than for the other methods. We note that in this case we have $\|\mathbf{Q} - \mathbf{Q}_\Delta\|_\infty \approx 0.265$. We also observe that the simplified Newton method does not only have better convergence rates but also has the smallest ratios of all methods for all Δt. Thus, if complex arithmetic is not a problem (or circumvented in other ways) and if the Jacobian of the right-hand side is available, this approach seems to be preferable.

5 Conclusion and outlook

In this paper we have introduced, discussed and analyzed two different approaches for parallelizing iterations of implicit spectral deferred corrections across the quadrature nodes. While parallel-in-time algorithms like PFASST allow to compute multiple time-steps simultaneously and target large-scale parallelism in time, the ideas presented here focus on the small-scale parallelization of a single time-step. In the sense of [3], these ideas therefore realize "parallelization across the method" for SDC.

The first approach allows simultaneous evaluation of SDC updates on all quadrature nodes by using a diagonal preconditioning matrix $\mathbf{Q}_\Delta$ instead of standard Gauß-Seidel-like choices. With $\mathbf{Q}_\Delta$ being diagonal, all M "stages" of SDC, i.e. updates for the solution at each quadrature node, can be computed using M processes. This includes solving an implicit system at the nodes as well as evaluation of the right-hand side of the initial value problem, all of which can now be done in parallel. We chose three different diagonal matrices and analyzed their impact on the convergence of SDC using four different test problems. While for non-stiff cases all candidates performed rather well, only the third alternative using a minimization approach was able to work about as good as the standard SDC preconditioners for stiff problems, too. Yet, so far no conclusive theory exists to estimate the impact of the choice of $\mathbf{Q}_\Delta$ in the convergence of SDC, let alone for the derivation of optimal (serial or parallel) preconditioners. For this approach, generic and rather obvious choices of $\mathbf{Q}_\Delta$ were considered and it can be expected that better candidates might exists, in particular if the problem at hand is taken into account.

The second approach uses diagonalization of the quadrature matrix $\mathbf{Q}$ to achieve parallelism across the nodes. For linear problems and suitable choices of quadrature rules, this yields a direct parallel solver of the collocation problem. For non-linear problem, though, this

is not applicable and linearization via Newton's method is needed. The resulting linear systems for each Newton iteration looks similar to a linear collocation problem, but only when the full Jacobian is frozen at the first node the diagonalization technique is applicable. This simplified Newton method shows quite remarkable convergence properties when compared to standard SDC. Yet, complex arithmetic is needed due to the complex eigenvalues of $\mathbf{Q}$. If this is an issue, the simplified Newton method can be further extended to an inexact simplified Newton method by diagonalizing $\mathbf{Q}_\Delta$ instead of $\mathbf{Q}$. For these two methods, simplified and inexact simplified Newton, we were able to prove linear convergence and show that the constants depend quadratically in the first and linearly in the second case on Δt. This makes the convergence of the inexact method about as fast as standard SDC, which we could also observe numerically using a non-linear test problem.

Both approaches are rather easy to implement and the code used for generating the numerical results of this paper can be found online within the `pySDC` framework [22]. We firmly believe that there are more ways to achieve parallelism within SDC, either across the nodes or even across the iterations. One possibility is to apply methods for parallelization across the step like Parareal [18] for the preconditioner itself. When choosing the implicit Euler for $\mathbf{Q}_\Delta$, each iteration of SDC is just a sequence of implicit Euler steps with a modified right-hand side. Thus, if a method is able to parallelize M implicit Euler steps, we could use it within SDC as parallel preconditioner. If this method is an iterative method itself, then with SDC being the outer iteration ideas like inexact SDC [24] could further speed up the algorithm.

References

1. Bouzarth, E.L., Minion, M.L.: A multirate time integrator for regularized Stokeslets. Journal of Computational Physics **229**(11), 4208 – 4224 (2010)
2. Burrage, K.: Parallel methods for initial value problems. Applied Numerical Mathematics **11**(1–3), 5–25 (1993)
3. Burrage, K.: Parallel methods for ODEs. Advances in Computational Mathematics **7**, 1–3 (1997)
4. Christlieb, A.J., Macdonald, C.B., Ong, B.W.: Parallel high-order integrators. SIAM Journal on Scientific Computing **32**(2), 818–835 (2010)
5. Dutt, A., Greengard, L., Rokhlin, V.: Spectral deferred correction methods for ordinary differential equations. BIT Numerical Mathematics **40**(2), 241–266 (2000)
6. Emmett, M., Minion, M.L.: Toward an Efficient Parallel in Time Method for Partial Differential Equations. Communications in Applied Mathematics and Computational Science **7**, 105–132 (2012)
7. Feng, Z.: Traveling wave behavior for a generalized fisher equation. Chaos, Solitons & Fractals **38**(2), 481 – 488 (2008)
8. Gander, M.J., Halpern, L., Ryan, J., Tran, T.T.B.: A Direct Solver for Time Parallelization, pp. 491–499. Springer International Publishing, Cham (2016)
9. Hairer, E., Wanner, G.: Solving ordinary differential equations. II. Stiff and differential-algebraic problems. Springer series in computational mathematics. Springer, Heidelberg, New York (2010)
10. van der Houwen, P., Sommeijer, B.: Analysis of parallel diagonally implicit iteration of Runge-Kutta methods. Applied Numerical Mathematics **11**(1), 169–188 (1993)
11. Huang, J., Jia, J., Minion, M.: Accelerating the convergence of spectral deferred correction methods. Journal of Computational Physics **214**(2), 633 – 656 (2006)
12. Jackson, K.R., Kværnø, A., Nørsett, S.P.: The Use of Butcher Series in the Analysis of Newton-like Iterations in Runge-Kutta Formulas. Appl. Numer. Math. **15**(3), 341–356 (1994)
13. Jackson, K.R., Nørsett, S.P.: The potential for parallelism in rungekutta methods. part 1: Rk formulas in standard form. SIAM Journal on Numerical Analysis **32**(1), 49–82 (1995)
14. Jay, L.O.: Inexact simplified newton iterations for implicit runge-kutta methods. SIAM J. Numer. Anal. **38**(4), 1369–1388 (2000)
15. Jones, E., Oliphant, T., Peterson, P., et al.: SciPy: Open source scientific tools for Python (2001–). URL http://www.scipy.org/. [Online; accessed 2017-02-02]
16. Kelley, C.T.: Iterative Methods for Linear and Nonlinear Equations. No. 16 in Frontiers in Applied Mathematics. SIAM (1995)
17. Layton, A.T., Minion, M.L.: Conservative multi-implicit spectral deferred correction methods for reacting gas dynamics. Journal of Computational Physics **194**(2), 697 – 715 (2004)
18. Lions, J.L., Maday, Y., Turinici, G.: A "parareal" in time discretization of PDE's. Comptes Rendus de l'Acadmie des Sciences - Series I - Mathematics **332**, 661–668 (2001)
19. Minion, M.L.: Semi-implicit spectral deferred correction methods for ordinary differential equations. Communications in Mathematical Sciences **1**(3), 471–500 (2003)
20. Minion, M.L.: Semi-implicit projection methods for incompressible flow based on spectral deferred corrections. Applied Numerical Mathematics **48**(3–4), 369 – 387 (2004). Workshop on Innovative Time Integrators for PDEs
21. Ruprecht, D., Speck, R.: Spectral deferred corrections with fast-wave slow-wave splitting. SIAM Journal on Scientific Computing **38**(4), A2535–A2557 (2016)
22. Speck, R.: Parallel-in-Time/pySDC: Parallel SDC (2017). DOI 10.5281/zenodo.376982. URL http://www.parallelintime.org/pySDC/
23. Speck, R., Ruprecht, D., Emmett, M., Minion, M., Bolten, M., Krause, R.: A multi-level spectral deferred correction method. BIT Numerical Mathematics **55**(3), 843–867 (2015)
24. Speck, R., Ruprecht, D., Minion, M., Emmett, M., Krause, R.: Inexact spectral deferred corrections. In: T. Dickopf, J.M. Gander, L. Halpern, R. Krause, F.L. Pavarino (eds.) Domain Decomposition Methods in Science and Engineering XXII, pp. 389–396. Springer International Publishing (2016)
25. Tang, T., Xie, H., Yin, X.: High-order convergence of spectral deferred correction methods on general quadrature nodes. Journal of Scientific Computing **56**(1), 1–13 (2013)

26. Van Der Houwen, P., Sommeijer, B.: Parallel iteration of high-order Runge-Kutta methods with stepsize control. Journal of Computational and Applied Mathematics **29**(1), 111 – 127 (1990)
27. Weiser, M.: Faster SDC convergence on non-equidistant grids by DIRK sweeps. BIT Numerical Mathematics **55**(4), 1219–1241 (2014)
28. Winkel, M., Speck, R., Ruprecht, D.: A high-order Boris integrator. Journal of Computational Physics **295**, 456–474 (2015)
29. Xia, Y., Xu, Y., Shu, C.W.: Efficient time discretization for local discontinuous Galerkin methods. Discrete and Continuous Dynamical Systems – Series B **8**(3), 677 – 693 (2007)