Spectral Deferred Corrections... with less pain

Martin Schreiber

2023-07-24

Abstract

This document is written for applied mathematicians and HPC engineers who still need to become more familiar with spectral deferred correction (SDC) methods and who would like to avoid the frustration of SDC methods by understanding them, implementing them, and understanding their limitations. It provides both a mathematical perspective and an implementation perspective, which becomes particularly challenging once the SDC methods are extended to other, more generic formulations. This document is intended for teaching purposes rather than publication and is mainly a review, providing insight from different angles. Aspects such as numerical investigation, stability regions, etc., are not included since this is left to be either found in the publications or generated relatively straightforwardly.

If you spot any errors, please let the author know (schreiberx@gmail.com)!

Contents

1 Introduction

SDC methods have become increasingly popular over the last decade since they allow higher-order time integration by solely providing 1st order time integrators. SDC then allows iterative improvement of the solution.

Although they are of higher order, their applicability needs to be improved due to their large computational complexity. Also, in areas such as atmospheric simulations, their time step size is still limited compared to more sophisticated time integration methods such as semi-Lagrangian methods, and further research around this is currently under investigation.

The following notes have been strongly influenced by various publications. Although originally not directly related to SDC methods, we start with the collocation formulation [3] in Sec. 3.

Then, we continue with SDC (Sec. 4) where we will briefly explain the first paper about SDC methods [2] (Sec. 4.1). Here, the corrections have been explicitly expressed in terms or error and residual polynomials which is rather complicated. Alternative formulations have been found and investigated (see, e.g., [4, 1]) which provide a more convenient formulation and in particular avoid the explicit computation of the error polynomial which we call compact node-to-node formulation (Sec. 37). Originally intended for mathematical studies, another formulation (see [5]) allows to use a compact zero-to-node (Sec. 4.3) based on a matrix formulation.

Since the devil is in the detail, we continue by discussing details on how to implement the SDC methods in Sec. 5 if using explicit, implicit or arbitrary integration methods.

2 Problem

Given an ODE

$$u'(t) = f(u(t), s), \;\; u(t_0) = u_0 \tag{1}$$

to solve in the interval

$$t \in [0, \Delta t].$$

Solutions to this can be computed using the Picard formulation

$$u(t) = u(a) + \int_a^t f(u(s), s) ds. \tag{2}$$

3 Collocation method

We continue with the collocation method and introduce (quadrature) nodes τ_i which are given by

$$\{\tau_i | 1 \le i \le M\} \tag{3}$$

within the unit interval

$$\tau_i \in [0, 1]. \tag{4}$$

Quadrature-based nodes have two main advantageous: They don't suffer of the Runge phenomena and they allow to use a higher-order quadrature at then end (See [2]). We extend these points to

$$\{\tau_i | 0 \le i \le M + 1\} \tag{5}$$

by setting $\tau_0 = 0$ and $\tau_{M+1} = \Delta t$. Then, the subtimesteps are given by

$$\Delta \tau_m = \tau_m - \tau_{m-1} \;\; \text{for}\, m = 1, \ldots, M + 1. \tag{6}$$

The integral in Eq. (2) can be approximated by

$$u(\tau_m) = u_0 + \sum_{j=1}^{M} q_{m,j} f(u(\tau_j), \tau_j) \;\; \text{for}\, m = 1, \ldots, M. \tag{7}$$

The coefficients $q_{m,j}$ can be computed using the standard collocation method

$$q_{m,j} = \int_0^{\tau_m} l_j(s) ds \tag{8}$$

with l_j the j-th Lagrange polynomial.

3.1 Closure methods

Finally, we need to get a solution at $u(t = \Delta t)$.

1. **Direct method:**
   If the last quadrature point coincides with $t = \Delta t$, we could take the solution at this last quadrature point as the solution.

2. **Extrapolation:**
   We can use an extrapolation by computing an interpolation through all quadrature points and evaluating it at $t = \Delta t$.

3. **Quadrature:**
   A higher-order accuracy can be gained with a quadrature method using the corresponding weights w_j of some quadrature for the quadrature points τ_j to evaluate the integral of Eq. (2).

3.2 Iterative solution

We use the exponent k to describe the k-th iteration and use the notation

$$U = (u_1, u_2, \ldots, u_M) = (u(\tau_1), u(\tau_2), \ldots, u(\tau_M)) \tag{9}$$

$$U_0 = (u_0, u_0, \ldots, u_0) \tag{10}$$

$$\tau = (\tau_1, \ldots, \tau_M) \tag{11}$$

and

$$F(U, \tau) = (f(u_1, \tau_1), f(u_2, \tau_2), \ldots, f(u_M, \tau_M)). \tag{12}$$

We can then write the solution as

$$U(\tau) = U_0 + QF(U(\tau), \tau) \tag{13}$$

where Q is the spectral integration matrix given by Eq. (8) or as a fixed point iteration

$$\begin{aligned} U^{k+1}(\tau) &= U_0 + QF\left(U^k(\tau), \tau\right) \\ &= U^k + \left(U_0 - U^k + QF\left(U^k, \tau\right)\right) \end{aligned} \tag{14}$$

with

$$U = \left(u_1^k, u_2^k, \ldots, u_M^k\right) = \left(u^k(\tau_1), u^k(\tau_2), \ldots, u^k(\tau_M)\right). \tag{15}$$

4 SDC derivation

We will first derive SDC from Dutt et al.'s original formulation [2] and then relate it to a more modern formulation. We start with the residual in the k-th iteration given by

$$r^k(t) = u^k(a) + \int_a^t f\left(u^k(s), s\right) ds - u^k(t) \tag{16}$$

and the error

$$\epsilon^k(t) = u(t) - u^k(t) \tag{17}$$

$$\Leftrightarrow \quad u(t) = \epsilon^k(t) + u^k(t). \tag{18}$$

Using Eq. (18) in the Picard Eq. (2) yields

$$\epsilon^k(t) + u^k(t) = \epsilon^k(a) + u^k(a) + \int_a^t f(\epsilon^k(s) + u^k(s), s) ds. \tag{19}$$

Next, we can rewrite Eq. (16) as

$$u^k(a) = r^k(t) - \int_0^t f\left(u^k(s), s\right) ds + u^k(t) \tag{20}$$

and using this equation in Eq. (19) yields

$$\epsilon^k(t) + \cancel{u^k(t)} = \epsilon^k(a) + r^k(t) - \int_a^t f\left(u^k(s), s\right) ds + \cancel{u^k(t)} + \int_a^t f(\epsilon^k(s) + u^k(s), s) ds \tag{21}$$

$$\epsilon^k(t) = \epsilon^k(a) + r^k(t) - \int_a^t f\left(u^k(s), s\right) + f(\epsilon^k(s) + u^k(s), s) ds \tag{22}$$

Using the notation

$$G(u^k, \epsilon^k, s) = f(u^k(s) + \epsilon^k(s), s) - f\left(u^k(s), s\right) \tag{23}$$

we can obtain

$$\epsilon^k(t) - \epsilon^k(a) = \int_a^t G(u^k(s), s) ds + r^k(t). \tag{24}$$

4.1 Original SDC line-by-line

We arrived at a form where G only depends on the error (we know how to evaluate it at the current point) and where we need to approximate the residual somehow. Therefore, this equation can be solved using some off-the-shelf time integrator, which we denote by

$$I_a^t(u^k(s), \epsilon^k(s)) \approx \int_a^t G(u^k(s), \epsilon^k(s), s) ds \tag{25}$$

advancing $G(u^k, s)$ from a to t. Introducing the discrete points in time τ_i with $a = \tau_m$ and $t = \tau_{m+1}$ and the corresponding discrete error e_i^k and residual r_i^k we rearrange Eq. (24) to

$$\epsilon_{m+1}^k - \epsilon_m^k = I_0^{\tau_{m+1}}\left(u^k, \epsilon^k\right) - I_0^{\tau_m}\left(u^k, \epsilon^k\right) + r_{m+1}^k \tag{26}$$

$$\epsilon_{m+1}^k = \epsilon_m^k + I_{\tau_m}^{\tau_{m+1}}\left(u^k, \epsilon^k\right) + r_{m+1}^k \tag{27}$$

What's left is to find an approximation of the residual integral. We can use again a spectral integration where we obtain

$$r_{m+1} = u_m + \int_{\tau_m}^{\tau_{m+1}} f\left(u^k(s), s\right) ds - u_{m+1}^k \tag{28}$$

$$= \int_{\tau_m}^{\tau_{m+1}} f\left(u^k(s), s\right) ds - u_{m+1}^k + u_m^k \tag{29}$$

$$\approx \sum_{1 \le i \le M} s_{m,i} f\left(u_i^k, \tau_i\right) ds - u_{m+1}^k + u_m^k \tag{30}$$

with the coefficients given by

$$s_{m,j} = \int_{\tau_m}^{\tau_{m+1}} l_j(s) ds. \tag{31}$$

This reconstruction always includes all quadrature points. Finally we get our SDC iteration by

$$\boxed{\begin{aligned} \epsilon_{m+1}^k &= \epsilon_m^k + I_{\tau_m}^{\tau_{m+1}}\left(u_m^k, \epsilon_m^k, s\right) + \sum_{1 \le i \le M} s_{m,i} f\left(u_i^k, \tau_i\right) ds - u_{m+1}^k + u_m^k. && (32) \\ u_m^{k+1} &= u_m^k + \epsilon_m^k. && (33) \end{aligned}}$$

The difference to the original SDC formulation is that we didn't yet specialize this form on either explicit or implicit methods, but kept the general integrator R in it which could be replace by whatever integrator.

4.2 Compact node-to-node

The next steps we describe here provides a by far more elegant formulation and in particular compact formulation of the SDC method which avoids explicitly setting up the error and residual.

Using the 2nd equation $\epsilon_m^k = u_m^{k+1} - u_m^k$ in the 1st one provides a more compact form (see [4, 1]) where we obtain

$$u_{m+1}^{k+1} \cancel{- u_{m+1}^k} = u_m^{k+1} \cancel{- u_m^k} + I_{\tau_m}^{\tau_{m+1}}\left(u_m^k, u_m^{k+1} - u_m^k\right) + \sum_{1 \le i \le M} s_{m,i} f\left(u_i^k\left(\tau_i\right), \tau_i\right) ds \cancel{- u_{m+1}^k} + \cancel{u_m^k}.$$

Then, we can write the time integrator even more compactly as

$$\begin{aligned}
I_{\tau_m}^{\tau_{m+1}}\left(u^k, u^{k+1}-u^k\right) &\approx \int_{\tau_m}^{\tau_{m+1}} G(u^k, u^{k+1}-u^k, s)ds && (34)\\
&= \int_{\tau_m}^{\tau_{m+1}} f(u^k + u^{k+1} - u^k, s) - f\left(u^k(s), s\right) ds && (35)\\
&= \int_{\tau_m}^{\tau_{m+1}} f(u^{k+1}, s) - f\left(u^k(s), s\right) ds. && (36)
\end{aligned}$$

Note that we didn't use any discretization on u, yet, since this will depend on the time integrator. Then, we obtain the compact notation which is also known as **node-to-node** formulation

$$\boxed{u_{m+1}^{k+1} = u_m^{k+1} + I_{\tau_m}^{\tau_{m+1}}\left[f(u^{k+1}, s)\right] - I_{\tau_m}^{\tau_{m+1}}\left[f\left(u^k, s\right)\right] + \sum_{1\le i\le M} s_{m,i} f\left(u_i^k, \tau_i\right). \qquad (37)}$$

4.3 Compact zero-to-node

We can use this recursively to obtain the **zero-to-node** formulation

$$u_m^{k+1} = u_0 + \sum_{1\le j\le m-1} I_{\tau_j}^{\tau_{j+1}}\left[f(u^{k+1}, s)\right] - I_{\tau_j}^{\tau_{j+1}}\left[f\left(u^k, s\right)\right] + \sum_{1\le i\le M} q_{m,i} f\left(u_i^k, \tau_i\right) \tag{38}$$

with the matrix Q entries given by

$$q_{m,i} = \sum_{1\le j\le m-1} s_{j,i} \tag{39}$$

matrix E elements

$$E = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix} \tag{40}$$

where

$$I\left[\cdot\right] = \left(I_{\tau_0}^{\tau_1}\left[\cdot\right], I_{\tau_1}^{\tau_2}\left[\cdot\right], ..., I_{\tau_{M-1}}^{\tau_M}\left[\cdot\right]\right) \tag{41}$$

we can obtainand finally also the **matrix-based zero-to-node formulation**

$$\boxed{U^{k+1} = U_0 + E\left(I\left[F(U^{k+1}, \tau)\right] - I\left[F(U^k, \tau)\right]\right) + QF\left(U^k\right). \qquad (42)}$$

Again, we like to emphasize that the integrator I is so far not specified and we will introduce it in the next section.

4.4 Closure methods

The closure methods for SDC methods are the same as for the collocation method and we refer to (3.1) for possible solutions.

5 Realization with compact node-to-node

Next, we discuss different approaches to specialize the general SDC method based on the compact node-to-node form (37) introduced before to particular time integrators. This basically replaces the integrator I with explicit, implicit and arbitrary time integrators either to the entire tendencies or by splitting it into subterms of the form

$$f(u, s) = f_{\text{explicit}}(u, s) + f_{\text{implicit}}(u, s) + f_{\text{integration}}(u, s)$$

where each term refers to the method how this term should be integrated.

We use the nomenclature "sweep" which has been introduced to refer to one iteration of the SDC method from U^k to U^{k+1} and to differentiate to the iteration from one time node U_m to U_{m+1}.

5.1 Initialization sweep

The first sweep can be an arbitrary time integrator. We can also take the perspective that this should be a good initial guess to start the SDCs themselves. The initial sweep starts at $\tau_0 = 0$ and uses M iterations to time integrate solutions to τ_M.

Depending on which particular time integrators will be used, some parts have to be cached which will be discussed in the following individual sections.

5.2 SDC iteration

For sake of better discussion, we give a recap of equation (37) here which we will follow:

$$u_{m+1}^{k+1} = u_m^{k+1} + \underbrace{I_{\tau_m}^{\tau_{m+1}}\left[f(u^{k+1},s)\right] - I_{\tau_m}^{\tau_{m+1}}\left[f\left(u^k,s\right)\right]}_{\text{Integrator correction}} + \underbrace{\sum_{1\leq i\leq M} s_{m,i} f\left(u_i^k,\tau_i\right)}_{\text{Residual correction}}.$$

We split this into two main parts, the integrator correction which solely depends on time integrators and the residual correction based on time tendencies.

Implementation remarks:

- For the residual correction, time tendency evaluations $f(u)$ are obviously required. Depending on which integrator correction is used, they are already to be computed already for the integration correction itself, but might also have to be computed just for this residual evaluation, depending on which discretization of the integrator is used.
- Two buffers are used to read the tendency terms $f(u^k,s)$ computed in the previous iteration and $f(u^{k+1},s)$ to store the tendencies for the current iteration.
- The first time step goes from $t = \tau_0 = 0$ to $t = \tau_1$, but not with a regular time integrator as we one would expect. This is described in other implementation remarks below.

5.2.1 Explicit time integrators

Using an explicit time integrator uses an approximation replaces the time integrator with an explicit one, hence

$$I_{\tau_m}^{\tau_{m+1}}\left[f_{\text{explicit}}(u^{k+1},s)\right] - I_{\tau_m}^{\tau_{m+1}}\left[f_{\text{explicit}}\left(u^k,s\right)\right] = \Delta\tau_m f_{\text{explicit}}\left(u_m^{k+1},\tau_m\right) - \Delta\tau_m f_{\text{explicit}}\left(u_m^k,\tau_m\right)$$

This can be computed straight-forward since the tendencies $f_{\text{explicit}}\left(u_m^k,\tau_m\right)$ have been computed in the previous iteration and $f_{\text{explicit}}\left(u_m^{k+1},\tau_m\right)$ can be evaluated since u_m^{k+1} is known.

*Implementation remark*s:

- For the first sweep and for efficiency reasons, some time tendencies were computed (and buffered) in the initial sweep. Then, during each sweep, all tendency evaluations are also buffered for the next sweep.
- For the first iteration, computing the correction for u_1, **no evaluation (=correction) is required** since both tendencies are evaluate at u_0 which simply cancel out.
- No special attention needs to be paid to the non-autonomous evaluations here since the functions are always evaluated at τ_m.

5.2.2 Implicit time integrators

Implicit time integrators simply use the solution at the next point in time, hence to integrate to τ_{m+1}, they use the solution at u_{m+1} which leads to this scheme:

$$I_{\tau_m}^{\tau_{m+1}}\left[f_{\text{implicit}}(u^{k+1},s)\right] - I_{\tau_m}^{\tau_{m+1}}\left[f_{\text{implicit}}\left(u^k,s\right)\right] = \Delta\tau_m f_{\text{implicit}}\left(u_{m+1}^{k+1},\tau_{m+1}\right) - \Delta\tau_m f_{\text{implicit}}\left(u_{m+1}^k,\tau_{m+1}\right).$$

Although $f_{\text{implicit}}(u_{m+1}^k)$ seems to require an implicit evaluation at the next time step τ_{m+1}, it is already known from the previous iteration k. In particular, it is also required to be computed as part of the residual corrections, hence, we can directly use it from there without any additional evaluations required. The only term requiring an implicit solve is then $f_{\text{implicit}}\left(u^{k+1}_{m+1}, \tau_{m+1}\right)$ which we solve by rearranging the entire equation (including the residual corrections) and solve it to obtain u^{k+1}_{m+1}.

Implementation remarks:

- Attention needs to be paid to non-autonomous functions since the implicit time integration as well as the tendencies are evaluated at τ_{m+1}.

5.2.3 IMEX time integrators

In what follows, we will describe one particular IMEX time integrator. Different IMEX methods are possible and we will simply follow the approach to treat them separately as

$$\begin{aligned} I_{\tau_m}^{\tau_{m+1}}\left[f(u^{k+1}, s)\right] - I_{\tau_m}^{\tau_{m+1}}\left[f\left(u^k, s\right)\right] &= \Delta\tau_m f_{\text{implicit}}\left(u^{k+1}_{m+1}, \tau_{m+1}\right) - \Delta\tau_m f_{\text{implicit}}\left(u^{k}_{m+1}, \tau_{m+1}\right) \\ &\quad + \Delta\tau_m f_{\text{explicit}}\left(u^{k+1}_{m}, \tau_{m}\right) - \Delta\tau_m f_{\text{explicit}}\left(u^{k}_{m}, \tau_{m}\right). \end{aligned}$$

Solving for this is straightforward by first adding and subtracting the explicit time tendencies, then the one for the implicit part followed by applying the implicit solver for the term $f_{\text{implicit}}\left(u^{k+1}_{m+1}, s\right)$.

Note, that this is just one particular implementation of an IMEX scheme and this could be, e.g., part of the general time integrator discussed next. We like to point out the compact matrix-based notation for this formulation following [5], given by

$$U^{k+1} = U_0 + Q_\Delta^{\text{explicit}}\left(F_{\text{explicit}}(U^{k+1}, \tau) - F_{\text{explicit}}(U^k, \tau)\right) + Q_\Delta^{\text{implicit}}\left(\underbrace{F_{\text{implicit}}(U^{k+1}, \tau)}_{\text{Unknown } U^{k+1} \text{ for next time steps}} - F_{\text{implicit}}(U^k, \tau)\right) + QF\left(U^k\right)$$

with

$$Q_\Delta^{\text{fast}} = \begin{bmatrix} \Delta\tau_1 & & & & \\ \Delta\tau_1 & \Delta\tau_2 & & & \\ \Delta\tau_1 & \Delta\tau_2 & \Delta\tau_3 & & \\ \vdots & \vdots & \vdots & \ddots & \\ \Delta\tau_1 & \Delta\tau_2 & \Delta\tau_3 & \cdots & \Delta\tau_M \end{bmatrix} \quad Q_\Delta^{\text{slow}} = \begin{bmatrix} 0 & & & & \\ \Delta\tau_1 & 0 & & & \\ \Delta\tau_1 & \Delta\tau_2 & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ \Delta\tau_1 & \Delta\tau_2 & \Delta\tau_3 & \cdots & 0 \end{bmatrix}.$$

5.2.4 General time integrators

Using an arbitrary integrator requires just a few changes. In principle, we simply need to follow the formulation

$$I_{\tau_m}^{\tau_{m+1}}\left[f(u^{k+1}, s)\right] - I_{\tau_m}^{\tau_{m+1}}\left[f\left(u^k, s\right)\right] \tag{43}$$

and replace each I with the corresponding time integration. That's actually it if there wouldn't be some peculiarities for the implementation.

Implementation remarks:

- Each time integrator needs to be applied directly on the solution u^k without first applying the residual correction.
- The term $I_{\tau_m}^{\tau_{m+1}}\left[f\left(u, s\right)\right]$ only refers to the update to the solution u_m, not to the estimated solution u_{m+1}. Using some arbitrary time integrator computing u_{m+1}, we then get $I_{\tau_m}^{\tau_{m+1}}\left[f\left(u^k, s\right)\right] = u_{m+1} - u_m$.
- For sake of efficiency, the integration $I_{\tau_m}^{\tau_{m+1}}\left[f\left(u^k, s\right)\right]$ itself is stored in a buffer to be reused in the next iteration. Comparing the computational complexity of this method with the implicit one requires just one additional subtract operation.

5.2.5 All-in-one

The all-in-one method can finally assemble all components in different orders. We recommend the following order:

1. Partial evaluation of explicit time integrator: If an explicit time integrator is used, we first need to evaluate the tendencies on u^k.

2. General time integrator: We evaluate this first because we need the solution u^k to evaluate it first.
3. Residual corrections: Then, we apply the residual corrections.
4. Explicit time integrator: The explicit time integrator can then be applied based on the tendencies of the previous iteration and with results from step 1.
5. Implicit time integrator: This is applied in the same way as before.

6 Preconditioner-based SDC

Taking the perspective of a fixed point iteration from Eq. 14 we can introduce a preconditioner (see [6]). Assuming that U^k approaches the exact solution, we know that the residual $r^k = U_0 - (I - \Delta t Q F) U^k$ approaches zero. Therefore, we could try to use a preconditioner P to get

$$U^{k+1} = U^k + P \underbrace{\left(U_0 - (I - \Delta t Q F) U^k\right)}_{=\text{residual}} \tag{44}$$

which would target accelerating the convergence speed of the iterations.

6.1 Derivation

We can now choose a particular preconditioner, namely one which is based on a particular matrix shape such as

$$P = (I - \Delta t Q_\Delta F)^{-1} \tag{45}$$

where Q_Δ is of lower diagonal form. Note, that later on we could replace $Q_\Delta F$ with all kinds of different split preconditioners, e.g., $Q_\Delta F = \left(Q_\Delta^{\text{fast}} F^{\text{fast}} + Q_\Delta^{\text{slow}} F^{\text{slow}}\right)$. If we apply this preconditioner we obtain

$$\begin{aligned}
U^{k+1} &= U^k + (I - \Delta t Q_\Delta F)^{-1} \left(U_0 - (I - \Delta t Q F) U^k\right) && (46)\\
(I - \Delta t Q_\Delta F) U^{k+1} &= (I - \Delta t Q_\Delta F) U^k + \left(U_0 - (I - \Delta t Q F) U^k\right) && (47)\\
&= U^k - \Delta t Q_\Delta F U^k + U_0 - U^k + \Delta t Q F U^k && (48)\\
&= -\Delta t Q_\Delta F U^k + U_0 + \Delta t Q F U^k && (49)\\
(I - \Delta t Q_\Delta F) U^{k+1} &= U_0 + \Delta t (Q - Q_\Delta) F U^k. && (50)
\end{aligned}$$

Note the lower diagonal form of Q_Δ allowing us to solve for U^{k+1} more conveniently.

6.2 Convergence speed

We can investigate its convergence speed by assuming linearity of $F = \lambda$, using its error $\epsilon^k = u^k - u^*$ with u^* the exact numerical solution given by $u_0 = (I - \Delta t Q \lambda) u^*$. We then obtain

$$\begin{aligned}
(I - \Delta t Q_\Delta \lambda) u^{k+1} &= u_0 + \Delta t (Q - Q_\Delta) \lambda u^k && (51)\\
(I - \Delta t Q_\Delta \lambda) u^{k+1} - (I - \Delta t Q \lambda) u^* &= \Delta t (Q - Q_\Delta) \lambda u^k && (52)\\
(I - \Delta t Q_\Delta \lambda) u^{k+1} - (I - \Delta t Q_\Delta \lambda) u^* + (I - \Delta t Q_\Delta \lambda) u^* - (I - \Delta t Q \lambda) u^* &= \Delta t (Q - Q_\Delta) \lambda u^k && (53)\\
(I - \Delta t Q_\Delta \lambda) e^{k+1} + \Delta t (Q - Q_\Delta) \lambda u^* &= \Delta t (Q - Q_\Delta) \lambda u^k && (54)\\
(I - \Delta t Q_\Delta \lambda) e^{k+1} &= \Delta t (Q - Q_\Delta) \lambda \epsilon^k && (55)\\
e^{k+1} &= \underbrace{(I - \Delta t \lambda Q_\Delta)^{-1} \Delta t \lambda (Q - Q_\Delta)}_{\text{iteration matrix } K} \epsilon^k. && (56)
\end{aligned}$$

6.3 Preconditioner choices

There are different choices of preconditioners.

- Implicit Euler

$$Q_\Delta = \begin{bmatrix} \Delta\tau_1 & & & \\ \Delta\tau_1 & \Delta\tau_2 & & \\ \Delta\tau_1 & \Delta\tau_2 & \Delta\tau_3 & \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix} \tag{57}$$

- LU trick

$$Q_\Delta = U^T \tag{58}$$

  for

$$Q^T = LU. \tag{59}$$

- Parallel SDC:

$$Q_\Delta = diag(\hat{q}) \tag{60}$$

References

[1] Tommaso Buvoli. A Class of Exponential Integrators Based on Spectral Deferred Correction. pages 1–22, 2015. _eprint: 1504.05543.

[2] Alok Dutt, Leslie Greengard, and Vladimir Rokhlin. Spectral Deferred Correction Methods for Ordinary. 40(2):1–26, 1998.

[3] Ernst Hairer, S.P. Norsett, and Gerhard Wanner. *Solving ordinary differential equations I: Nonstiff problems.* 1987.

[4] Michael L. Minion. Semi-implicit spectral deferred correction methods for ordinary differential equations. *Communications in Mathematical Sciences*, 1(3):471–500, 2003. ISBN: 978-960-474-071-0.

[5] Daniel Ruprecht and Robert Speck. Spectral Deferred Corrections with Fast-wave Slow-wave Splitting. *SIAM Journal on Scientific Computing*, 38(4):A2535–A2557, 2016. ISBN: 0001405101 _eprint: arXiv:1302.5877.

[6] Robert Speck. Parallelizing spectral deferred corrections across the method. *Computing and Visualization in Science*, March 2017. _eprint: 1703.08079.