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

RESEARCH

Adaptive time step selection for spectral deferred correction

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

Received: 20 March 2024 / Accepted: 17 October 2024

Abstract
Spectral Deferred Correction (SDC) is an iterative method for the numerical solution of ordinary differential equations. It works by refining the numerical solution for an initial value problem by approximately solving differential equations for the error, and can be interpreted as a preconditioned fixed-point iteration for solving the fully implicit collocation problem. We adopt techniques from embedded Runge-Kutta Methods (RKM) to SDC in order to provide a mechanism for adaptive time step size selection and thus increase computational efficiency of SDC. We propose two SDC-specific estimates of the local error that are generic and do not rely on problem specific quantities. We demonstrate a gain in efficiency over standard SDC with fixed step size and compare efficiency favorably against state-of-the-art adaptive RKM.

Keywords Parallel-in-time · Spectral deferred correction · Adaptivity

1 Introduction

Spectral Deferred Corrections (SDC) were introduced by Dutt et al. [1] as a more stable variant of the classical deferred correction approach for solving ordinary differential equations (ODEs). Deferred correction methods iteratively refine the numerical solution for an initial value problem by approximately solving differential equations for the error.

✉ Thomas Baumann
t.baumann@fz-juelich.de

1 Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Wilhelm-Johnen-Straße, Jülich 52428, Germany

2 Chair Computational Mathematics, Institute of Mathematics, Hamburg University of Technology, Am Schwarzenberg-Campus 3, Hamburg 21073, Germany

Published online: 28 October 2024

While SDC often requires more work per time step than classical Runge-Kutta Methods (RKM)[1] to reach a given order, it is more flexible due to the low order iterative solves, and attaining high order of accuracy is simple. This can make SDC computationally competitive when medium to high accuracy is desired [2]. SDC has been successfully applied to complex problems that benefit from operator splitting [3–5] and problems where only low order solvers are available [6, 7]. The recent review paper by Ong and Spiteri [8] provides an overview of deferred correction methods.

While adaptive time step selection for SDC has already been discussed in the original SDC paper [1], it has not yet been widely explored despite its potential to improve computational efficiency. Selecting time steps in SDC based on conserved quantities has been shown to mitigate order reduction [9, 10]. An algorithm similar to what we propose here, based on comparing the high order SDC solution to a low order secondary solution, was shown to work well [6]. However, both these approaches were tailored to specific problems.

This paper adopts techniques for adaptive step size selection from explicit and diagonally implicit RKM to SDC methods. What is new in this paper is that we use generic error estimates and combine a generic step size update equation known from embedded RKM with SDC. We show for four different nonlinear problems that adaptive step size selection for SDC improves computational efficiency with little problem-specific tweaking in a way that is familiar from embedded RKM. The step size selection requires only one additional parameter compared to standard SDC, a tolerance that is used to control an estimate of the local error.

SDC has gained popularity in the parallel-in-time (PinT) integration community due to its iterative and highly flexible nature [11]. SDC-based PinT algorithms have been devised in both parallel-across-the-method [12] as well as parallel-across-the-steps [13] fashion. We investigate adaptive step size selection for the parallel-across-the-steps Block Gauß-Seidel SDC algorithm and the parallel-across-the-method diagonal SDC approach. So far, there are only very few studies that explore adaptive step size selection for the Parareal PinT method [14–16] and none for SDC-flavored PinT algorithms.

2 Methods and background

This section reviews the SDC algorithm, including its parallelizable variants diagonal SDC and Block Gauß-Seidel SDC. It outlines adaptive step size selection strategies known from embedded RKM and proposes two ways of applying them to SDC.

2.1 Spectral deferred correction

SDC methods perform numerical integration of initial value problems (IVP)

$$u_t = f(u), \quad u(t_0) = u_0, \tag{1}$$

[1] Note that SDC with a fixed number of iterations can be interpreted as a Runge-Kutta method itself. However, throughout this article, we use the acronym RKM exclusively to refer to the diagonally implicit or explicit Runge-Kutta methods against which we compare adaptive SDC.

consisting of an ODE and initial conditions, where u is the solution, f the right-hand side function, and the subscript t indicates a derivative with respect to time. For ease of notation we restrict the discussion to autonomous scalar problems, i.e. $u(t), u_0, f(u) \in \mathbb{R}$ and solve a single time step from $t_0 = 0$ to $t_1 = \Delta t$. All derivations and results shown here can be transferred to higher dimensions, albeit with a more cumbersome notation.

We recast (1) in Picard form

$$u(t) = u_0 + \int_0^t f\,(u(s))\,ds, \quad t \in [0, \Delta t], \tag{2}$$

by integrating with respect to time. Next, the integral is approximated with a spectral quadrature rule using M collocation nodes $0 \leq \tau_m \leq 1$, $m = 1, ..., M$, rescaled by Δt to cover the interval $[0, \Delta t]$. The aim is to obtain values of the solution at the quadrature nodes and to use polynomial interpolation to approximate the continuous solution. Using Lagrange polynomials [17]

$$l_j^\tau(t) = \frac{\prod_{i=1, i\neq j}^{M}(t - \tau_i)}{\prod_{i=1, i\neq j}^{M}(\tau_j - \tau_i)}. \tag{3}$$

to approximate the right-hand side function yields

$$f(u(t)) \approx \sum_{j=1}^{M} f(u(\Delta t \tau_j)) l_j^\tau(t/\Delta t). \tag{4}$$

By defining quadrature weights

$$q_{mj} = \int_0^{\tau_m} l_j^\tau(s) ds, \tag{5}$$

we can approximate the solution at the quadrature nodes by

$$u_m := u_0 + \Delta t \sum_{j=1}^{M} q_{mj} f(u(\Delta t \tau_j)) \approx u(\Delta t \tau_m). \tag{6}$$

We call (6) for $m = 1, \ldots, M$ the collocation problem. To streamline notation, we write the collocation problem in vector form

$$\vec{u} = \vec{u}_0 + \Delta t\, Q F(\vec{u}), \tag{7}$$

where $Q \in \mathbb{R}^{M\times M}$ is the quadrature matrix containing the weights q_{mj}, $F(\vec{u}) = (f(u_1), ..., f(u_M))^T \in \mathbb{R}^M$ is a vector function for the temporal evolution, $\vec{u} = (u_1, ..., u_M)^T \in \mathbb{R}^M$ is the vector carrying the approximate solutions at the quadrature nodes and $\vec{u}_0 = (u_0, ..., u_0)^T \in \mathbb{R}^M$ the initial conditions.

This collocation problem corresponds to a fully implicit Runge-Kutta method. It can be solved directly, but doing so is expensive because Q is dense and the equations for the stages (6) are all coupled. Certain types of nodes, such as Gauß-Legendre, Gauß-Lobatto and Gauß-Radau can achieve super-convergence up to order $2M$ at the right boundary with only M nodes [18, Theorems 5.2, 5.3 and 5.5].

Using Picard iteration to solve the collocation problem provides a simple iterative scheme

$$\vec{u}^{k+1} = \vec{u}^k + \vec{r}^k \tag{8}$$

$$\vec{r}^k := \vec{u}_0 + \Delta t\, Q F\left(\vec{u}^k\right) - \vec{u}^k \tag{9}$$

where the solution in iteration k is improved by adding the residual $\vec{r}^k$. However, this method only converges for small Δt or (very) non-stiff ODEs. SDC preconditions Picard iterations with a low order method to achieve convergence also for large time step size. As we will now illustrate, this can be seen as integrating an equation for the error with said low order method.

The quadrature rule integrates the polynomial approximation exactly, i.e. $\int_0^t F(\vec{u})\vec{l}^\tau(s/\Delta t)ds = \Delta t(QF(\vec{u}))\vec{l}^\tau(t/\Delta t)$. Thus, if the error of the current polynomial approximation $\delta^k(t) = u(t) - \vec{u}_m^k \vec{l}^\tau(t/\Delta t)$ is plugged into (2), we get

$$\delta^k(t) - \int_0^t \left(f\left(\vec{u}^k\vec{l}^\tau(s/\Delta t) + \delta^k(s)\right) - F\left(\vec{u}^k\right)\vec{l}^\tau(s/\Delta t)\right) ds = \vec{r}^k\vec{l}^\tau(t/\Delta t). \tag{10}$$

The integral is approximated by a simpler quadrature rule Q_Δ, typically referred to as preconditioner, and the resulting nonlinear system

$$\vec{\delta}^k - \Delta t\, Q_\Delta \left(F\left(\vec{u}^k + \vec{\delta}^k\right) - F\left(\vec{u}^k\right)\right) = \vec{r}^k. \tag{11}$$

has to be solved in each iteration. The solution is then updated by adding the correction, $\vec{u}^{k+1} = \vec{u}^k + \vec{\delta}^k$. By expanding the residual and plugging in the refinement equation, we can eliminate the defect and simplify the SDC iteration to

$$\left(I_M - \Delta t\, Q_\Delta F\right)\left(\vec{u}^{k+1}\right) = \vec{u}_0 + \Delta t\left(Q - Q_\Delta\right) F\left(\vec{u}^k\right), \tag{12}$$

with I_M the $M \times M$ identity matrix.

The preconditioner Q_Δ is typically chosen to be a lower triangular matrix such that the system can be solved with forward substitution. In the context of partial differential equations with N degrees of freedom, the collocation problem is a $NM \times NM$ system and iterating K times with forward substitution allows to instead solve KM-many $N \times N$ systems. Since the algorithm proceeds from one line of the system to the next, SDC iterations are often referred to as "sweeps".

Preconditioners and diagonal SDC

In the original derivation of SDC, implicit or explicit Euler were proposed for solving the error equations [1]. These are first order quadrature rules, integrating from node

to node. This means they increase the order of accuracy by one up to the order of the underlying collocation problem, provided they converge at all. The preconditioner corresponding to implicit Euler, for example, reads:

$$Q_{\Delta}^{\mathrm{IE}} = \begin{pmatrix} \tau_2 - \tau_1 & 0 & 0 & \dots & 0 \\ \tau_2 - \tau_1 & \tau_3 - \tau_2 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \dots & 0 \\ \tau_2 - \tau_1 & \tau_3 - \tau_2 & \dots & \dots & \tau_M - \tau_{M-1} \end{pmatrix}. \tag{13}$$

Higher order preconditioners such as RKM can sometimes increase the order of accuracy by more than one with each iteration [19]. However, lack of smoothness in the error can limit gains to one order per iteration regardless of the order of the preconditioner, particularly when non-equidistant nodes are used [20].

Other interpretations of SDC do not rely on the preconditioner being consistent with an integration rule [21]. The LU preconditioner [22], where $Q_{\Delta} = U^{\mathrm{T}}$, with $LU = Q^{\mathrm{T}}$, is very effective for stiff problems. Be aware that most of these preconditioners cannot be interpreted as A-stable time marching schemes and usually pose some restrictions on the domain of convergence of the collocation problem.

Another promising class of preconditioners use a diagonal Q_{Δ} and allow updating all nodes in parallel [12]. This is a small-scale PinT algorithm since the number of nodes is typically small, but it can be combined with other PinT algorithms that solve multiple steps concurrently. Numerical experiments suggest that best parallel efficiency is obtained when maximizing parallelism across-the-nodes and spending the remaining computational resources on parallelizing across-the-steps [23].

Good diagonal preconditioners can be derived, for instance, by minimizing the spectral radius of the SDC iteration matrix in the stiff or non-stiff limit of the test equation. When running diagonal SDC, we use preconditioners derived in this way [24], namely MIN-SR-S, which is suitable for stiff problems. The performance of diagonal SDC variants combined with space-time parallelization has also been investigated [25].

Since the LU and diagonal preconditioner cannot guarantee an increase by exactly one order per sweep [24, 26], we stick to the implicit Euler preconditioner for most simulations.

Inexact SDC iterations

Performing the implicit solves inside the SDC iteration to full accuracy often comes at no benefit to overall accuracy because the preconditioner itself is only approximate. Reducing tolerances or strict limits on the number of allowed iterations for the implicit solver can be used to avoid over-solving and to improve overall computational efficiency of SDC [27]. Optimal tolerances can be derived, but require realistic work and error models [28]. Still, efficiency can be gained even with sub-optimal tolerances when adjusting the tolerance of an inner solver based on the outer residual or when allowing only few iterations of an inner solver. Because SDC provides good initial guesses for the nonlinear solver, this can lead to very efficient schemes [27].

For simplicity, we fix the ratio of inner tolerance to outer residual based on heuristics to some fixed value. Additionally, we can only employ inexactness if knowing the order

after every iteration is not required because otherwise we cannot guarantee that the inner solver is accurate enough to increase the order by one.

Implicit-explicit splitting

When solving problems with stiff and non-stiff components it is possible to treat only the stiff part implicitly and the remainder explicitly. This is called implicit-explicit (IMEX) splitting and can easily be used in SDC [2, 5]. The IMEX-SDC iteration reads

$$\begin{aligned}\left(1-\Delta t\tilde{q}^I_{m+1,m+1}f^I\right)\left(u^{k+1}_{m+1}\right) &= u_0+\Delta t\sum_{j=1}^{m}\left(\tilde{q}^I_{m+1,j}f^I+\tilde{q}^E_{m+1,j}f^E\right)\left(u^{k+1}_j\right)\\ &\quad-\Delta t\sum_{j=1}^{m+1}\left(\tilde{q}^I_{m+1,j}f^I+\tilde{q}^E_{m+1,j}f^E\right)\left(u^{k}_j\right)\\ &\quad+\Delta t\sum_{j=1}^{M}q_{m+1,j}\left(f^I+f^E\right)\left(u^{k}_j\right).\end{aligned}$$

with superscripts I and E referring to the implicit and explicit part and $\tilde{q}$ being the entries of the preconditioners. The scheme typically has the same order of accuracy as the non-split version, although order reduction may occur [5, 29]. IMEX-SDC has been shown to outperform DIRK-based IMEX Runge-Kutta methods for incompressible flow simulations in wall-time measurements [29].

Dense output

As the solution of the collocation problem is an approximation by a polynomial of degree M, a natural continuous extension is suggested by evaluating this polynomial anywhere within the interval. We refer to this as "dense output" property of the collocation problem [30, Sect. II.6]. Keep in mind that, while the solution at the boundary is up to order $2M$ accurate for M nodes, the accuracy inside the interval is order M.

2.2 Adaptive step size selection for SDC

Adaptive selection of the time step size is useful when the rate of change of the solution is not uniform across the computational domain. We transfer well-known concepts from embedded Runge-Kutta methods [30] to SDC. First, we need to estimate the local error, which we then aim to control by choosing an appropriate step size.

The error estimation works by computing two solutions to the same initial value problem with a different order of accuracy. The difference between the solutions is a reasonable estimate of the local error of the less accurate method

$$\begin{aligned}\epsilon &= \|u^{(p)}-u^{(q)}\|_\infty\\ &= \|\left(u^{(p)}-u^*\right)-\left(u^{(q)}-u^*\right)\|_\infty\\ &= \|\delta^{(p)}-\delta^{(q)}\|_\infty = \|\delta^{(p)}\|_\infty+\mathcal{O}\left(\Delta t^{q+1}\right),\end{aligned} \tag{14}$$

Algorithm 1 SDC with Δt-adaptivity

$u^0 \leftarrow u_0$
$k \leftarrow 1$
while $k \leq k_{\max}$ **do**
  $u^k \leftarrow$ SDC iteration applied to u^{k-1}
  $k \leftarrow k + 1$
end while
$\epsilon \leftarrow \|u^{k_{\max}} - u^{k_{\max}-1}\|$
$\Delta t \leftarrow \beta \Delta t \left(\frac{\epsilon_{\text{TOL}}}{\epsilon}\right)^{1/k_{\max}}$
if $\epsilon > \epsilon_{\text{TOL}}$ **then**
  Restart current step with u_0
else
  Move on to next step with $u^{k_{\max}}$
end if

where $u^{(p)}, u^{(q)}$ are the solutions at the end of the time interval, obtained by integration schemes of order p and q with $q > p$. u^* marks the exact solution, δ denotes the local error with analogous meaning of the superscript and ϵ is the estimate of the local error. Once ϵ is known, an optimal step size

$$\Delta t_{\text{opt}} = \beta \Delta t \left(\frac{\epsilon_{\text{TOL}}}{\epsilon}\right)^{1/(p+1)} \tag{15}$$

can be estimated such that $\epsilon \approx \epsilon_{\text{TOL}}$. Here, ϵ_{TOL} is the user-defined tolerance for the local error and β is a safety factor, usually $\beta = 0.9$. This update equation is based on the order of accuracy p of the time-marching scheme [30, 31]. As is also common in embedded RKM, we use "local extrapolation", meaning we advance using the higher order solution, even though we control the error of the lower order one. Crucially, we check if the local error estimate falls below the desired accuracy and we move on to the next step with Δt_{opt} only if it does. If we fail to satisfy the accuracy requirements, we recompute the current step with Δt_{opt}. While the time-scale of the problem may have changed in the next step, heuristically, Δt_{opt} often appears to be a good guess for the optimal step size for the next step as well.

Adaptive selection of Δt

Since the order increases by one with each SDC iteration (up to the order of the underlying collocation problem), the increment can be used directly as the error estimate. While adaptivity based on the increment was already proposed in the original SDC paper [1], they employ a simpler step size update equation based only on doubling or halving. A similar approach, although not yet called SDC, was studied by van der Houwen and Sommeijer [32, 33] for other preconditioners.

Algorithm 1 shows in pseudo-code the algorithm resulting from combining an increment based error estimation with (15).

Since this approach modifies only Δt but keeps the number of iterations constant, we refer to it as Δt-adaptivity. Order reduction may be observed for very stiff problems, requiring some extra care [34, 35].

Adaptive selection of Δt and k

By using the dense output property of the converged collocation problem, we can design an approach to choose both the time step Δt and the number of iterations k adaptively. For the fully converged collocation solution, the $M+1$ values $u_0, u_1, \ldots, u_M$ at the collocation points $\tau = \{\tau_i : i = 0, \ldots, M\}$ define a polynomial on $[0, \Delta t]$ that provides an M-th order accurate approximation at any point $t \in [0, \Delta t]$. By removing the collocation point τ_{M-1} and using a set of nodes

$$\tau^* = \{\tau_i : i = 0, \ldots, M-2, M\}, \tag{16}$$

we can construct a polynomial of degree $M-1$ and evaluate it at τ_{M-1} to produce an $M-1$-st order accurate approximation

$$u_{M-1}^{(M-1)} = \sum_{i=0}^{M-2} u_i^k l_i^{\tau^*}(\tau_{M-1}) + u_M^k l_{M-1}^{\tau^*}(\tau_{M-1}) \tag{17}$$

of the solution at the removed point. Here, $l_i^{\tau^*}$ are the Lagrange polynomials of the i-th node in the set τ^*. We can then use the difference

$$\epsilon = \|u_{M-1}^{(M-1)} - u_{M-1}^k\|_\infty, \tag{18}$$

as an error estimate. Tests not documented here suggest that the node at which to compute the error can be chosen more or less arbitrarily. We choose node $M-1$ in our examples since this performed well and leave a more detailed mathematical investigation for future work.

For very large step sizes the SDC iteration may not converge with arbitrary preconditioner. To mitigate, we introduce a relative limit $\gamma = 4$ on how much the step size is allowed to increase and set $k_{\max} = 16$, so that we can return to the last step size that allowed convergence if the desired residual tolerance was not achieved after $k_{\max}$ iterations.

Since this approach does not require a fixed k, we can simply keep iterating until convergence and thus let SDC choose the number of iterations as well as the step size. We therefore refer to this approach as Δt-k-adaptivity. Algorithm 2 sketches the algorithm in pseudo-code.

Note that the residual tolerance controls how accurately we solve the collocation problem while the error tolerance controls how close we are to the continuous solution of the ODE. Therefore, some care must be taken to reasonably balance the two - a very small residual with a large error tolerance, for example, will lead to a very good approximation of the collocation polynomial that is a poor approximation of the continuous solution. In our numerical examples, we choose a residual tolerance r_{TOL} a few orders of magnitude smaller than ϵ_{TOL}.

Mitigating the cost of restarts Restarting steps from scratch in Algorithms 1 and 2 is expensive but SDC offers a unique way to reduce this overhead: Using the dense output property, we can evaluate the polynomial of a restarted step at the new collocation

Algorithm 2 SDC with Δt-k-adaptivity

$\vec{u}^0 \leftarrow \vec{u}_0$
$k \leftarrow 1$
$r \leftarrow \|\vec{u}_0 + \Delta t\, Q F\left(\vec{u}^0\right) - \vec{u}^0\|_\infty$
$r_{\text{prev}} \leftarrow \infty$
$r_{\max} \leftarrow 10^9$
$no\_convergence \leftarrow False$
while $r > r_{\text{tol}}$ and not $no\_convergence$ **do**
  $\vec{u}^k \leftarrow$ inexact SDC iteration applied to $\vec{u}^{k-1}$
  $r \leftarrow \|\vec{u}_0 + \Delta t\, Q F\left(\vec{u}^k\right) - \vec{u}^k\|_\infty$
  if $r > r_{\max}$ or $r > r_{\text{prev}}$ or $k = k_{\max}$ **then**
    $no\_convergence \leftarrow True$
  end if
  $r_{\text{prev}} \leftarrow r$
  $k \leftarrow k + 1$
end while
if $no\_convergence$ **then**
  $\Delta t \leftarrow \Delta t / \gamma$
  Restart current step with u_0
else
  $\tau^* \leftarrow \{\tau_i, i \neq M-1\}$
  $\epsilon \leftarrow \| \sum_{i=0}^{M-2} u_i^k l_i^{\tau^*}(\tau_{M-1}) + u_M^k l_{M-1}^{\tau^*}(\tau_{M-1}) - u_{M-1}^k \|_\infty$
    ▷ Note that $l_{M-1}^{\tau^*}$ is associated with τ_M, i.e. $l_{M-1}^{\tau^*}(\tau_M) = 1$
  $\Delta t \leftarrow \max\left(\gamma, \beta\left(\frac{\epsilon_{\text{TOL}}}{\epsilon}\right)^{1/M}\right)\Delta t$
  if $\epsilon > \epsilon_{\text{TOL}}$ **then**
    Restart current step with u_0
  else
    Move on to next step with u^k
  end if
end if

nodes resulting from shorter step size, thus re-using the previously computed solution. Despite being not accurate enough to satisfy the prescribed tolerance, this is a good initial guess, and the SDC iteration is expected to converge in fewer iterations. This is sensible to do in the Δt-k-adaptivity strategy, but only when the collocation problem of the restarted step has converged, i.e., the SDC residual is small enough, as otherwise the solution is not a useful approximation. However, with the Δt-adaptivity strategy, a gain in efficiency is unlikely as the step size update equation will overestimate the optimal step size due to a one-time exceptionally good initial guess which the next step will not have access to.

Advantages of Δt- and Δt-k-adaptivity over k-adaptivity

Even without local error estimation, SDC can be used adaptively with a fixed step size by simply iterating until the SDC residual is below a set tolerance. We call this approach k-adaptivity. Our newly introduced Δt- and Δt-k-adaptivity approaches have two major advantages over k-adaptivity. First, the step size is a floating point number, which can be finely adjusted, whereas the number of iterations is an integer allowing the scheme much less control. Second, the local error estimate, (14), measures the error with respect to the continuous solution. By contrast, the SDC residual is only a measure of the iteration error with respect to the discrete collocation solution. While a

low residual indicates that the collocation problem has been solved to high accuracy, the truncation error of the method might still be large.

2.3 Pipeline-based parallelism: Block Gauß-Seidel SDC

A relatively straightforward pipeline-based parallel-in-time variant of SDC can be constructed by solving multiple time steps simultaneously in block Gauß-Seidel fashion [36]: Instead of waiting for the previous step to converge to full accuracy before starting a new step, we begin solving a new step as soon as a single iteration has been performed on the previous step in the block and keep refining the initial conditions with the iterates from the previous step between iterations.

Figure 1 illustrates both sequential SDC one the left and parallel-across-the-steps Block Gauß-Seidel SDC (GSSDC) on the right. A very similar parallel-in-time (PinT) algorithm based on pipelining more general deferred corrections, Revisionist Integral Deferred Correction (RIDC) [37], has been shown to provide good speedup. We focus on GSSDC as it is at the heart of the large-scale PinT algorithm PFASST [13]. In contrast to Guibert et al. [36], however, we do not use any overlap between the collocation nodes of different time steps. When using adaptive step size selection, we use a single Δt for all steps in the block.

It has been demonstrated that the convergence order is maintained when increasing the number of steps N in a block of GSSDC [38]. In particular, it is the same as for single step SDC, which means we can employ the adaptive step size controller with no modification. However, the results will not be identical to serial SDC due to the inexactness in the initial conditions.

While we can still estimate the local error by means of the increment, its interpretation is different in the multi-step version. Within a block, we are solving all steps with a first order accurate method, then we solve all steps with a second order accurate method and so on. This means that the increment is an estimate of the global error within the block. As the whole problem is divided into multiple blocks, we should still view this error as local in the context of the global time domain, but we need to be aware that the same local tolerance ϵ_{tol} applied to a block of multiple steps should result in smaller local errors in each step inside the block.

When the error estimate exceeds the prescribed tolerance, we have two choices for restarting: We can restart from the initial conditions of the first step in the block, or we can restart from the initial conditions of the first step in the block where the

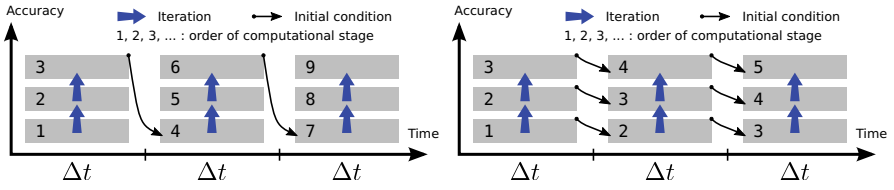

Fig. 1 Left: Sequential SDC. Right: Block Gauß-Seidel SDC. Sequential time stepping uses the converged solution of the last step as initial conditions, whereas the parallel version receives refined initial conditions between iterations

error estimate exceeds the threshold. The first is the more rigorous strategy while the second strategy is heuristic but can improve performance at a slight cost to accuracy. In the tests included here, we show only the second strategy because it performed consistently better.

3 Benchmark problems

We use one nonlinear ODE and three nonlinear PDEs as benchmarks to test the performance of our adaptive strategies. Each benchmark problem poses different challenges for numerical time-integration. All implementations are publicly available on GitHub[2] as part of the pySDC library [39]. Wall-clock times are measured on the JUSUF supercomputer [40] at Forschungszentrum Jülich.

3.1 Van der Pol

The van der Pol equation

$$u_{tt} - \mu\left(1 - u^2\right) u_t + u = 0, \tag{19}$$

$$u(t=0) = u_0, \quad u_t(t=0) = u_0', \tag{20}$$

is named after a Dutch electrical engineer who used the equation to study the behavior of vacuum tubes in radios [41]. Here, μ is a parameter controlling the nonlinearity, u is the solution and the subscript t marks a derivative with respect to time. In our tests, we set $u_0 = 1.1$ and $u_0' = 0$, $\mu = 1000$ and solve up to $t = 20$. In the pySDC implementation, we introduce $v(t) = u_t(t)$, rewrite the van der Pol equation as a first order system and use a Newton scheme to solve the nonlinear systems within the SDC sweeps.

For $\mu = 0$, we recover the harmonic oscillator, but with increasing μ the problem becomes increasingly stiff. Van der Pol describes the problem with $\|\mu\| \ll 1$ as modelling free oscillations of a triode oscillator, whereas $\|\mu\| \gg 1$ models a free relaxation oscillation [42]. This is a useful test problem for adaptive step size control as the nonlinear damping introduces a second time-scale to the oscillation, see Fig. 2.

We use the SciPy [43] method SOLVE_IVP from the INTEGRATE package with an explicit embedded Runge-Kutta method of order 5(4) [44] with tolerances close to machine precision to obtain reference solutions. When using Δt-k-adaptivity, we select $r_{\text{TOL}} = 10^{-5}\epsilon_{\text{TOL}}$, and stop the Newton scheme at a tolerance of $10^{-5}r$, with r the current SDC residual, or after a maximum of 9 iterations.

3.2 Quench

This is a simplified model of temperature leaks in superconducting magnets provided by Erik Schnaubelt [45, Section 4.3]. Once the temperature exceeds a certain threshold,

[2] https://github.com/Parallel-in-Time/pySDC

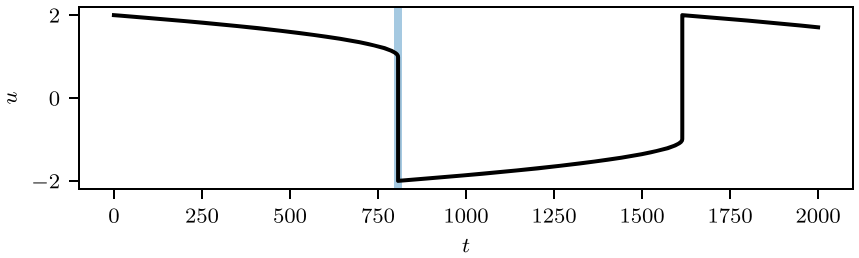

Fig. 2 Solution of a van der Pol problem for $\mu = 1000$ over time. The solution is oscillating on two time-scales. In order not to over-resolve the slow parts, the resolution has to be adjusted during runtime. In numerical tests, we solve only the shaded transition at $t \approx 800$ as this is already very expensive with fixed step size schemes. Note that at this high value of μ, the problem is extremely stiff

superconductivity ceases and runaway heating of the magnet sets in. This effect has led to the explosion of large magnets at the Large Hadron Collider [46].

The model consists of a one-dimensional heat equation with a nonlinear source term heating parts of the domain. For the boundary conditions, we choose Neumann-zero to treat the magnet as completely isolated from the environment, except for the leak. Due to superconductivity, the diffusivity is high, making the problem very stiff and prohibiting the use of explicit time-stepping schemes. The equation reads

$$C_V u_t - \kappa \Delta u = Q(u), \tag{21}$$

$$Q(u) = Q_{\max} \times \begin{cases} 1, & x \in (0.45, 0.55), \\ f(u), & \text{else}, \end{cases} \tag{22}$$

$$f(u) = \begin{cases} 0, & u < T_{\text{thresh}}, \\ \frac{u - T_{\text{thresh}}}{T_{\max} - T_{\text{thresh}}}, & T_{\text{thresh}} \le u < T_{\max}, \\ 1, & T_{\max} \le u, \end{cases} \tag{23}$$

$$\Omega \in]0, 1[, \tag{24}$$

$$u_x = 0, \ x \in \partial\Omega, \tag{25}$$

$$u(t = 0) = 0, \tag{26}$$

with the Laplacian Δ and parameters

$$\begin{aligned} C_V &= 1000, \\ \kappa &= 1000, \\ T_{\text{thresh}} &= 10^{-2}, \\ T_{\max} &= 2 \times 10^{-2}, \\ Q_{\max} &= 1. \end{aligned}$$

We solve until $t = 500$ and use a Newton scheme for implicit solves. Figure 3 shows the solution over time.

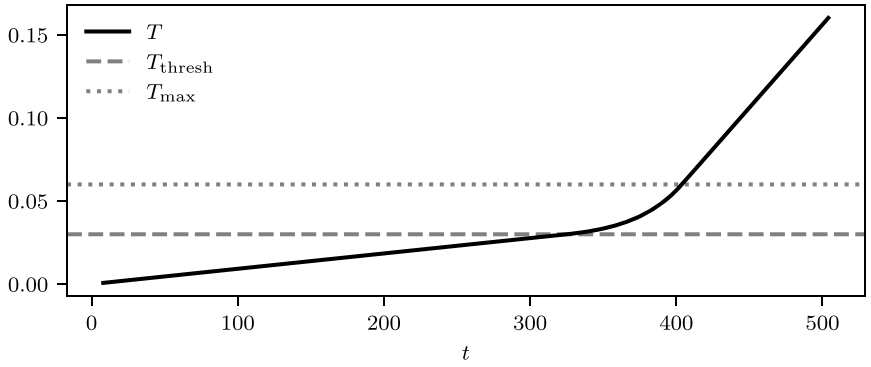

Fig. 3 Solution of the Quench problem. Shown is the maximal temperature $T(t) = \|u(x,t)\|_{L^\infty(\Omega)}$ across the spatial domain over time. We see a slow heating up to about $t = 320$, after which the temperature in some parts of the domain exceeds the threshold value and a linear transition towards runaway heating is entered. Physically, this means the magnet stops being superconducting, which can have catastrophic effects in particle accelerators

The behavior is fairly simple, except for the transition from superconductivity to runaway heating, which is challenging to resolve accurately.

We again rely on the SciPy method SOLVE_IVP to generate reference solutions, but, as the problem is very stiff, use an implicit backward differentiation formula [47]. When using Δt-k-adaptivity, we select $r_{\text{TOL}} = 10^{-1}\epsilon_{\text{TOL}}$, and stop the Newton scheme at a tolerance of $10^{-1}r$ or after a maximum of 5 iterations.

3.3 Nonlinear schrödinger

The focusing nonlinear Schrödinger equation is a wave-type equation that describes problems such as signal propagation in optical fibers [48]. The formulation we solve can be written as

$$u_t = i\,\Delta u + 2i\,\|u\|^2 u, \tag{27}$$

$$u(t=0) = \frac{1}{\sqrt{2}}\left(\frac{1}{1-\cos(x+y)/\sqrt{2}} - 1\right), \tag{28}$$

with i being the imaginary unit. We solve (27) on a two-dimensional spatial domain with periodicity 2π up to $t = 1$ using fast Fourier transforms. We use implicit-explicit (IMEX) splitting to integrate the Laplacian implicitly and the nonlinear term explicitly. The global error is computed with respect to the analytic solution [49, Equation (39)] See Fig. 4 for a visualization of the solution. When using Δt-k-adaptivity, we select $r_{\text{TOL}} = 10^{-4}\epsilon_{\text{TOL}}$

3.4 Allen-Cahn

The Allen-Cahn variant considered here is a two-dimensional reaction-diffusion equation with periodic boundary conditions that can be used to model transitions between

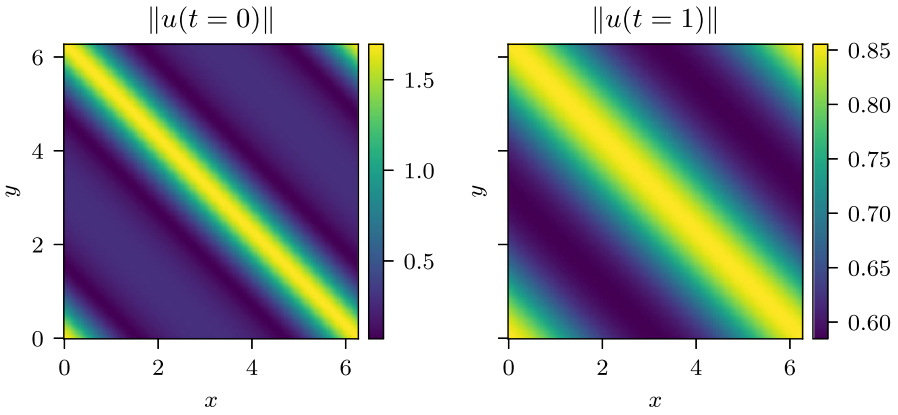

Fig. 4 Plot of the initial condition for the Schrödinger example (left) and the solution at the end of the interval under consideration (right). The initial conditions are purely real, but the solution shifts to the complex domain as the simulation progresses

two phases. We choose initial conditions representing a circle of one phase embedded in the other phase. We add time-dependent forcing, such that the circle alternates between growing and shrinking.

$$u_t = \Delta u - \frac{2}{\epsilon^2} u(1-u)(1-2u) - 6u(1-u) f(u,t), \tag{29}$$

$$f(u,t) = \frac{\sum \left(\Delta u - \frac{2}{\epsilon^2} u(1-u)(1-2u) \right)}{\sum 6u(1-u)} \left(1 - \sin\left(4\pi \frac{t}{0.032} \right) \times 10^{-2} \right) \tag{30}$$

$$u_0(x) = \tanh\left(\frac{R_0 \|x\|}{\sqrt{2}\epsilon} \right), \tag{31}$$

$$x \in [-0.5, 0.5[^2 \tag{32}$$

$$\epsilon = 0.04, \tag{33}$$

$$R_0 = 0.25, \tag{34}$$

Figure 5 shows the initial condition and the evolution of the radius over time. Similar to our approach for the Schrödinger problem, we use IMEX splitting, integrating the Laplacian discretized with a spectral method implicitly while treating the nonlinear term explicitly. The IMEX scheme is not unconditionally stable, providing an upper limit on the step size and adding to the challenges for adaptive step size selection. We again compute the error with respect to the SciPy method SOLVE_IVP. When using Δt-k-adaptivity, we select $r_{\mathrm{TOL}} = 10^{-3} \epsilon_{\mathrm{TOL}}$

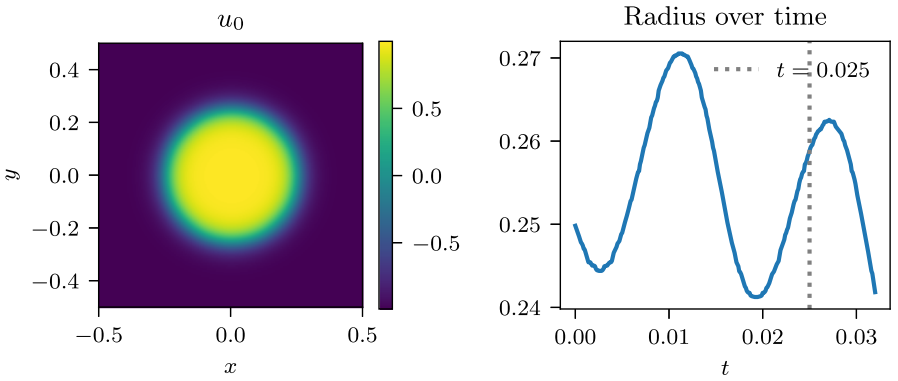

Fig. 5 Left: Initial condition consisting of a circle of high phase embedded in the low phase. Right: Evolution of the radius of the circle over time. The simulation is terminated at $t = 0.025$

4 Numerical results

We investigate performance of the methods from Section 2 applied to the problems from Section 3 with respect to computational efficiency and then compare them to established RKM.

4.1 Computational efficiency

We demonstrate the adaptive resolution capabilities in detail for the van der Pol problem first, as it easy to visualize. Figure 6 shows the solution (upper), local error (middle) and computational work, measured in total number of required Newton iterations (lower) for the van der Pol equation.

Resolving the transitions requires a very small step size due to the high stiffness of the problem, but in between transitions, a much larger step size suffices. We select parameters such the maximal local error during the transition is on the order of 10^{-5}. For fixed step size schemes, the resulting step size is so small, that we solve to machine precision outside of the transition. The associated computational effort to cover this short simulation time is approximately 70 times of what is required by step size adaptive strategies to meet the same accuracy requirements. In tests not shown here, we found similar, although less pronounced, trends for smaller values of μ.

Figure 7 shows error versus wall-clock time for the four problems from Section 3. Choosing the step size adaptively is beneficial in all cases, but it proves particularly essential for the van der Pol case. Δt-k-adaptive SDC is particularly efficient for the PDE examples. However, the IMEX-scheme used in Allen-Cahn becomes unstable for large step sizes, causing around 100 restarts for loose tolerances and decreasing the efficiency of Δt-k-adaptive SDC. A remedy is to limit the step size, which proved to increase efficiency in these cases in tests not shown here. Note that even though efficiency is decreased, Δt-k-adaptivity provides physical solutions for any tolerance here.

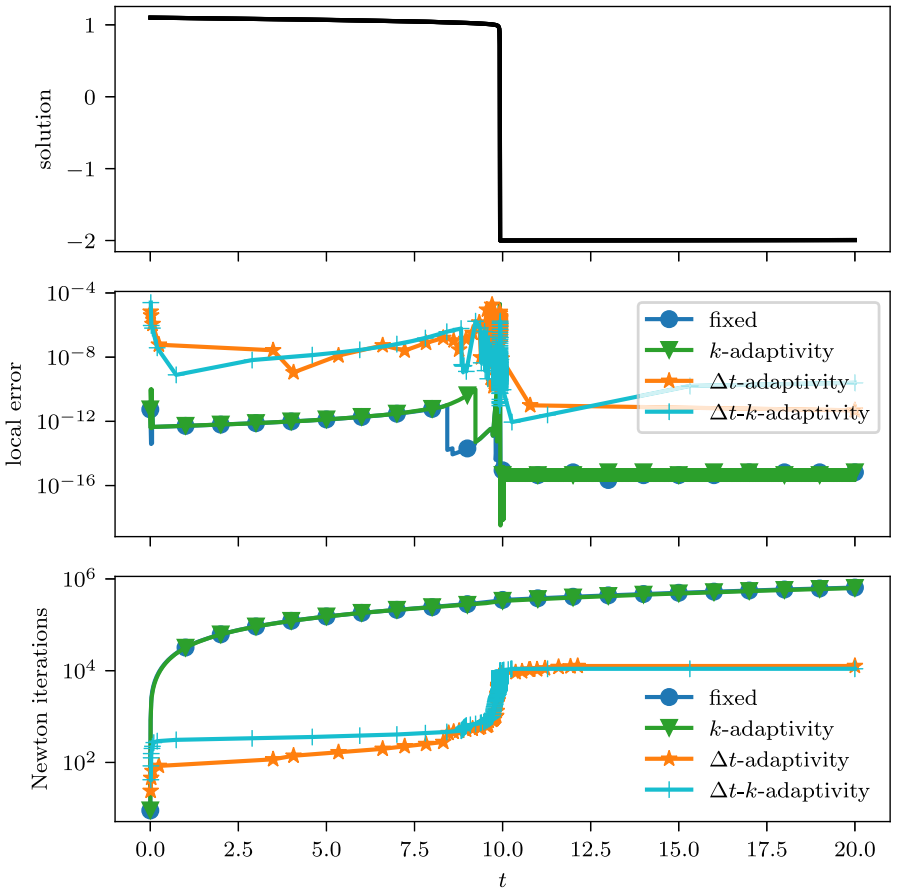

Fig. 6 SDC for the van der Pol problem with $\mu = 1000$, solved with four different strategies. The top panel shows the solution, the middle panel shows the resolution via the local error compared to a reference solution and the bottom panel shows how many Newton iterations are needed to reach the respective time, which is a good indicator of computational cost. We show only a fast transition which requires very small step sizes due to extreme stiffness. The solution is periodic with the next transition appearing at around $t = 600$ (see Fig. 2). Even in this short time period, schemes with fixed step size need about 70 times as many Newton iterations as adaptive ones, with the difference only becoming larger during the long period of little action until the next transition. Note that each marker represents a single step for methods with adaptive Δt or 10000 for the fixed or k-adaptive methods

We investigate the relationship between ϵ_{TOL} and the resulting error in Fig. 8. We select the step size by controlling the local error of a lower order method, see (14) and (15). The resulting scaling for the global error is $e \propto \epsilon_{\text{TOL}}^{q/p+1}$, with p and q as in (14). That means we expect a linear dependence of global error on step size for Δt-adaptivity for any q. For Δt-k-adaptivity, on the other hand, the scaling depends on the choice of quadrature; in our case the result is $e \propto \epsilon_{\text{TOL}}^{5/4}$. We observe this scaling in practice to a reasonable degree. Some deviations due to stiffness or stability restrictions are not unexpected and occur also in embedded RKM.

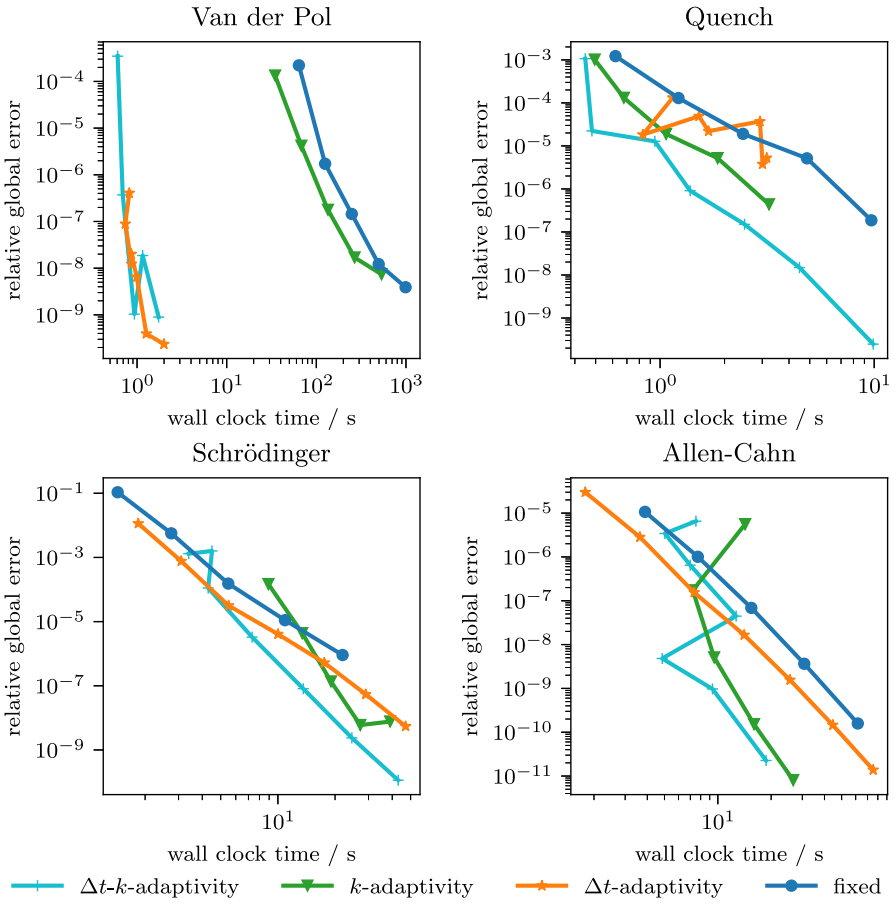

Fig. 7 Wall-clock time versus the global error relative to the magnitude of the solution. For Δt- and Δt-k-adaptivity, the tolerance for the local error is adjusted, whereas the step size controls the accuracy of the other strategies. Note that for the Quench problem in Δt-adaptivity, the global error is not necessarily reduced when choosing a smaller tolerance because the error depends sensitively on the way the transition to runaway heating is resolved. Adaptive resolution vastly enhances efficiency in all cases

To mitigate the cost of restarts, we use interpolation of the polynomial to obtain an initial guess after the collocation problem has converged with too large a step size in Δt-k-adaptivity, as discussed in Section 2.2. This indeed reduces the number of iterations in our numerical tests, but in the case of diagonally preconditioned SDC requires all-to-all communication, incurring additional cost. However, the number of such restarts is small for all problems under consideration, such that the overall computational cost changes only little. For problems with dynamics that lead to more restarts, on the other hand, interpolation can reduce the computational cost more significantly.

Parallel variants of SDC

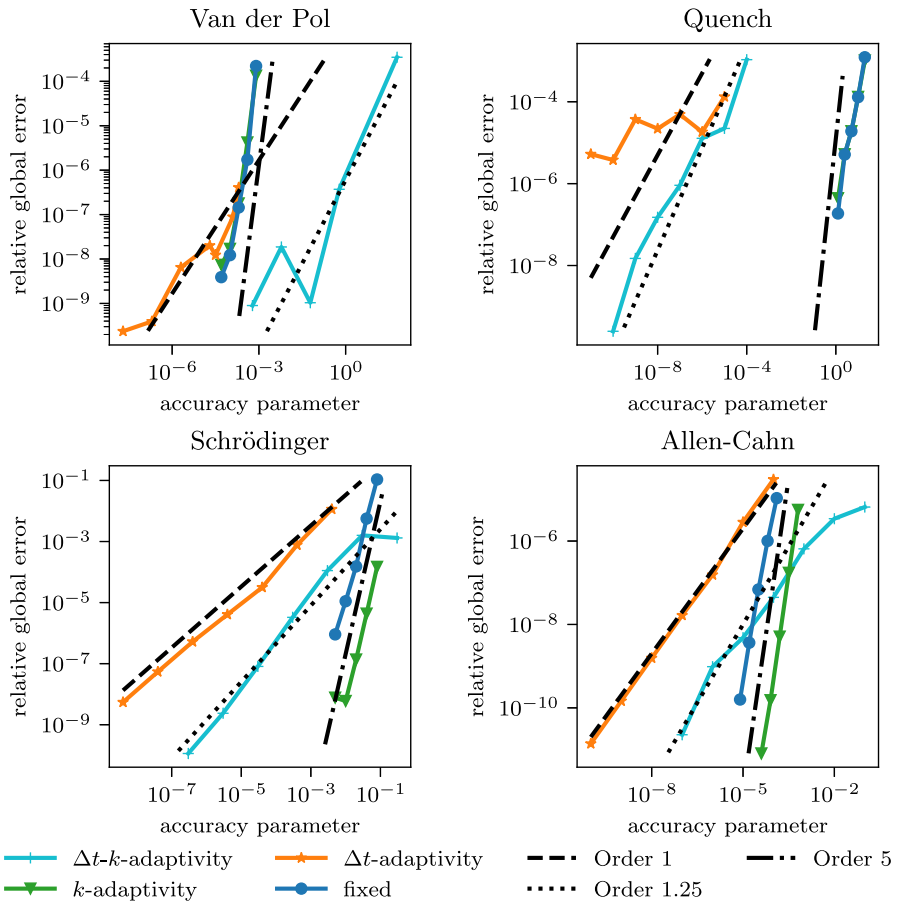

Fig. 8 Accuracy versus the parameter used to control it. For the fixed and k-adaptivity schemes this means step size and ϵ_{TOL} for step size-adaptive variants. The methods all have a global order of accuracy of 5 with respect to step size. We expect $e \propto \epsilon_{\text{TOL}}$ for Δt-adaptivity and $e \propto \epsilon_{\text{TOL}}^{1.25}$ for Δt-k-adaptivity, which we also observe in practice. Only in the combination of Δt-adaptivity and Quench, do we see significant deviations due to the complicating effect of the transition to runaway heating

Figure 9 shows error against wall clock time for two serial adaptive variants of SDC, a Δt-adaptive block parallel variant of SDC, a Δt-k-adaptive variant with diagonal preconditioner and a Δt-k-adaptive combination of the latter two.

For the three PDE examples, the Δt-k-adaptive SDC with diagonal preconditioner ("parallel-across-the-method") [24] is the most efficient variant. For the ODE and Allen-Cahn examples, the Δt-adaptive block parallel GSSDC variant also shows good speedup. Since the implicit solvers are relatively expensive in the Quench problem, which uses a Newton scheme and finite difference discretization, parallel efficiency of diagonal SDC is especially good. Note that the serial comparison runs for diagonal SDC were also performed with diagonal preconditioners as these were found to give the best performance in our tests.

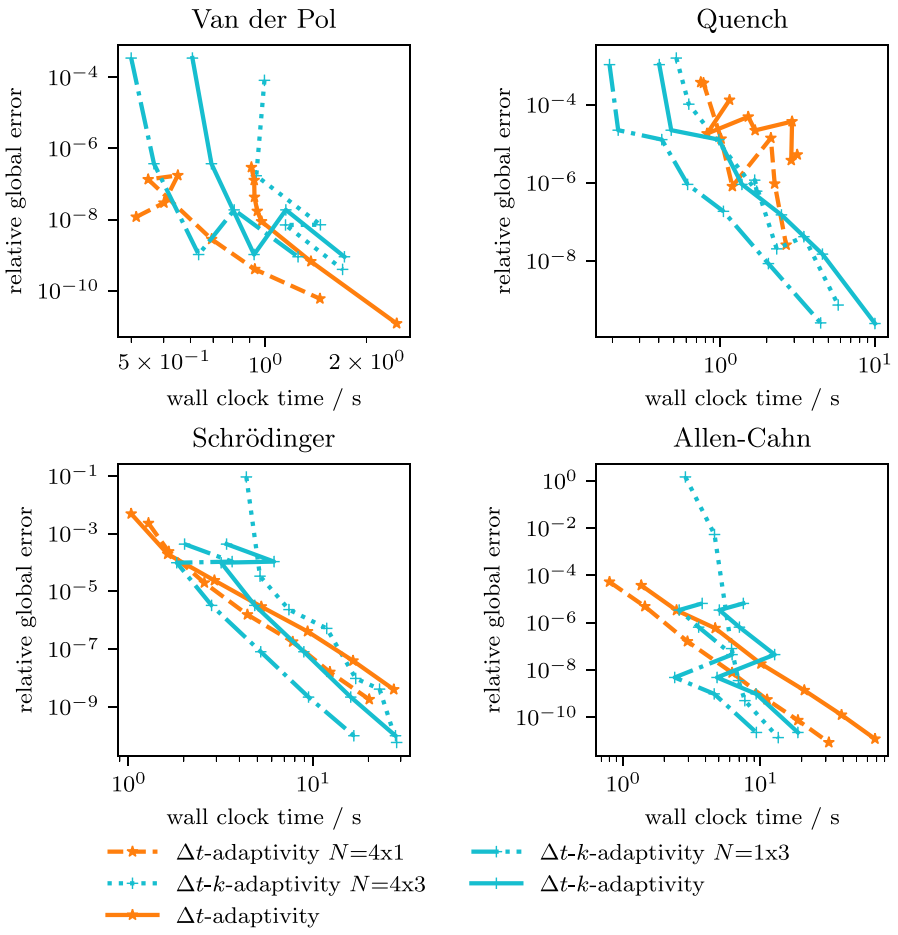

Fig. 9 Timings for parallel flavors of step size adaptive SDC. The number of processes is denoted, if larger than one, as N=[number of processes for GSSDC]x[number of processes for diagonal SDC]. For van der Pol and Allen-Cahn, GSSDC gives decent parallel efficiency, whereas better parallel efficiency is observed with diagonal SDC for the other problems. Wall time is roughly halved when using three processes in diagonal SDC. Stability issues in the IMEX solver of Allen-Cahn can lead to very large errors or crashes when combining diagonal SDC and GSSDC

Unfortunately, combining both diagonal SDC and GSSDC does not further improve performance. The reason seems to be that the performance of GSSDC is highly sensitive to the preconditioner and that it works best with implicit Euler for reasons not yet well understood. Switching to a diagonal preconditioner in GSSDC significantly increases the number of iterations required which negates any performance gains from parallelization. A deeper investigation of this issue is left for future work.

4.2 Comparing efficiency against Runge-Kutta methods

After establishing that adaptive SDC is generally more efficient than SDC with fixed Δt and k, we now compare the run-times of Δt- and Δt-k-adaptive SDC against state-of-the-art embedded RKM of the same order. Figure 10 shows error against wall clock time for two variants of SDC against different embedded RKM for our four benchmark problems. For van der Pol and Quench, we compare against ESDIRK5(3) [50], a singly diagonally implicit, stiffly accurate pair of orders 3 and 5 with an explicit first stage. For Schrödinger and Allen-Cahn, we compare against ARK5(4) [51], an additive pair of a singly diagonally implicit stiffly accurate L-stable embedded method of orders 4 and 5 and an explicit embedded method of orders 4 and 5. All problems are sufficiently stiff such that explicit RKM were encountering stability issues, produced unphysical

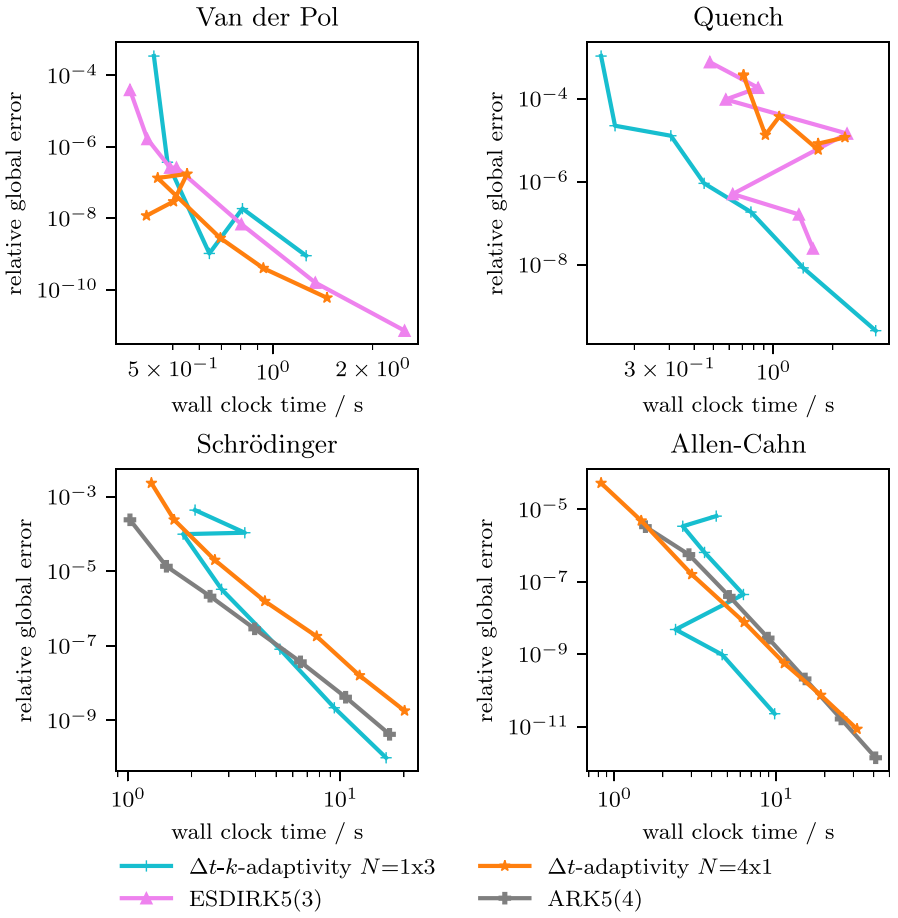

Fig. 10 Comparison of embedded RKM with parallel versions of Δt- and Δt-k-adaptive SDC from Fig. 9. All methods are of order 5. We find that we can outperform the RKM with one version of step size adaptive and parallel SDC for all problems

solutions or no solutions at all within a reasonable time frame relative to the implicit methods.

We find that at least one type of step size adaptive parallel SDC can outperform the RKM for all examples. In our testing, the RKM is only faster for Schrödinger when low accuracy is sufficient. Therefore, adaptive SDC is not only more efficient than standard SDC but also competitive with state-of-the-art adaptive RKM. SDC also offers easy tuning of the order, simply by changing parameters. By contrast, ARK5(4) is the highest order additive RKM we are aware of. In experiments not shown here, we repeated the comparison with third order accurate methods. We found the same trends as in Fig. 10 with at least one SDC method outperforming the RKM, even though lower order SDC methods allow for less concurrency. Note that there exist also stage-parallel RKM, which allow parallelism with a similar idea as diagonal SDC, see e.g. [52]. However, they often have worse stability than their serial counterparts and we were unable to find an embedded method of order higher than 4 in the literature.

5 Summary

The paper adopts concepts from adaptive embedded Runge-Kutta methods to spectral deferred correction. We propose procedures to control the step size Δt in SDC, the iteration number k or both. Our adaptive techniques can also be used for variants of SDC that are parallelizable, either in a Gauss-Seidel "parallel-across-the-steps" fashion (GSSDC) or in a "parallel-across-the-method" way by using a diagonal preconditioner. Numerical examples demonstrate that adaptive SDC is more efficient than SDC with fixed step size and iteration number and that adaptive parallel SDC can be competitive with embedded Runge-Kutta methods for the integration of four complex, nonlinear time-dependent problems.

One advantage of SDC is flexibility due to its iterative nature, making it easy to include techniques like splitting. The Δt-k-adaptive algorithm, in particular, retains much of the flexibility of SDC. Preconditioners or reduced accuracy spatial solves tailored to specific problems can be used. Also, spatial adaptivity may be leveraged in unique ways [53–55].

Adaptive time stepping has been explored for revisionist integral deferred correction (RIDC) [56], which is similar to GSSDC. However, they find that the increment is a poor choice for an error estimate, because of the accumulation of local errors from different steps in the increment. The reason why it works well for GSSDC is that instead of allowing maximal pipelining, GSSDC solves only fixed size blocks of steps at a time, each with the same step size. While sacrificing flexibility compared to RIDC, this allows us to view the accumulated errors inside the blocks as a local error of the block and enables efficient step size selection in GSSDC. More elaborate step size selection for GSSDC that allow different step sizes for each step in the blocks like in RIDC are left for future work.

The similarity of our methods to embedded RKM means any step size controller used in that context can also be used in SDC. It is well known that ideas like limiting the step size can further boost efficiency on a problem specific basis. So while our method only uses a single tolerance parameter, more elaborate step size update equations can

readily be employed by domain scientists with intimate knowledge of the problem at hand.

Acknowledgements The authors gratefully acknowledge computing time granted on JUSUF through the CSTMA project at Jülich Supercomputing Centre. The authors also gratefully acknowledge the helpful comments from the reviewers.

Author Contributions T.B. wrote the main manuscript text and carried out the implementations. T.L. prepared Fig. 1. All authors, T.B., S.G., T.L., D.R. and R.S., reviewed the manuscript and considerably contributed to both text and results.

Funding Open Access funding enabled and organized by Projekt DEAL. We thankfully acknowledge funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 955701. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Belgium, France, Germany, and Switzerland. We also thankfully acknowledge funding from the German Federal Ministry of Education and Research (BMBF) grant 16HPC048.

Data Availability For instructions on how to reproduce the results shown in this paper, please consult the project section in the documentation of the pySDC code at https://parallel-in-time.org/pySDC/projects/Resilience.html.

Declarations

Competing Interests The authors declare no competing interests.

Ethical Approval Not applicable

References

1. Dutt, A., Greengard, L., Rokhlin, V.: Spectral deferred correction methods for ordinary differential equations. BIT Numerical Math. **40**(2), 241–266 (2000). https://doi.org/10.1023/A:1022338906936
2. Ruprecht, D., Speck, R.: Spectral deferred corrections with fast-wave slow-wave splitting. SIAM J. Scientific Comput. **38**(4), 2535–2557 (2016). https://doi.org/10.1137/16M1060078
3. Bourlioux, A., Layton, A.T., Minion, M.L.: High-order multi-implicit spectral deferred correction methods for problems of reactive flow. J. Comput. Phys. **189**(2), 651–675 (2003). https://doi.org/10.1016/S0021-9991(03)00251-1
4. Layton, A.T., Minion, M.L.: Conservative multi-implicit spectral deferred correction methods for reacting gas dynamics. J. Comput. Phys. **194**(2), 697–715 (2004). https://doi.org/10.1016/j.jcp.2003.09.010
5. Minion, M.L.: Semi-implicit spectral deferred correction methods for ordinary differential equations. Commun. Math. Sci. **1**(3), 471–500 (2003). https://doi.org/10.4310/CMS.2003.v1.n3.a6
6. Guo, R., Xu, Y.: High order numerical simulations for the binary fluid-surfactant system using the discontinuous Galerkin and spectral deferred correction methods. SIAM Journal on Scientific Computing. **42**(2), 353–378 (2020). https://doi.org/10.1137/18M1235405

7. Feng, X., Tang, T., Yang, J.: Long time numerical simulations for phase-field problems using p-adaptive spectral deferred correction methods. SIAM J. Scientific Comput. **37**(1), 271–294 (2015). https://doi.org/10.1137/130928662
8. Ong, B.W., Spiteri, R.J.: Deferred Correction Methods for Ordinary Differential Equations. J. Scientific Comput. **83**(3), 60 (2020). https://doi.org/10.1007/s10915-020-01235-8
9. Quaife, B., Biros, G.: Adaptive time stepping for vesicle suspensions. J. Comput. Phys. **306**, 478–499 (2016). https://doi.org/10.1016/j.jcp.2015.11.050
10. Guo, R., Xu, Y.: A high order adaptive time-stepping strategy and local discontinuous Galerkin method for the modified phase field crystal equation. Commun. Comput. Phys. **24**(1), 123–151 (2018). https://doi.org/10.4208/cicp.OA-2017-0074
11. Gander, M.J.: 50 years of time parallel time integration. In: Carraro, T., Geiger, M., Körkel, S., Rannacher, R. (eds.) Multiple Shooting and Time Domain Decomposition Methods vol. 9, pp. 69–113. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-23321-5_3 . Series Title: Contributions in Mathematical and Computational Sciences
12. Speck, R.: Parallelizing spectral deferred corrections across the method. Comput. Visualization Sci. **19**(3), 75–83 (2018). https://doi.org/10.1007/s00791-018-0298-x
13. Emmett, M., Minion, M.: Toward an efficient parallel in time method for partial differential equations. Commun. Appl. Math. Comput. Sci. **7**(1), 105–132 (2012). https://doi.org/10.2140/camcos.2012.7.105
14. Maday, Y., Mula, O.: An adaptive parareal algorithm. J. Comput. Appl. Math. **377**, 112915 (2020). https://doi.org/10.1016/j.cam.2020.112915
15. Legoll, F., Lelièvre, T., Sharma, U.: An adaptive Parareal algorithm: Application to the simulation of molecular dynamics trajectories. SIAM J. Scientific Comput. **44**(1), 146–176 (2022). https://doi.org/10.1137/21m1412979
16. Kazakov, E., Efremenko, D., Zemlyakov, V., Gao, J.: A time-parallel ordinary differential equation solver with an adaptive step size: Performance assessment. In: Voevodin, V., Sobolev, S., Yakobovskiy, M., Shagaliev, R. (eds.) Supercomputing, pp. 3–17. Springer, Cham (2022)
17. Berrut, J.-P., Trefethen, L.N.: Barycentric Lagrange interpolation. SIAM Rev. **46**(3), 501–517 (2004). https://doi.org/10.1137/S0036144502417715
18. Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II. Springer Series in Computational Mathematics, vol. 14. Springer, Berlin, Heidelberg (1996). https://doi.org/10.1007/978-3-642-05221-7. http://link.springer.com/10.1007/978-3-642-05221-7
19. Christlieb, A., Ong, B., Qiu, J.-M.: Integral deferred correction methods constructed with high order Runge-Kutta integrators. Math. Comput. **79**(270), 761–783 (2009). https://doi.org/10.1090/S0025-5718-09-02276-5
20. Christlieb, A., Ong, B., Qiu, J.-M.: Comments on high-order integrators embedded within integral deferred correction methods. Commun. Appl. Math. Comput. Sci. **4**(1), 27–56 (2009). https://doi.org/10.2140/camcos.2009.4.27
21. Causley, M., Seal, D.: On the convergence of spectral deferred correction methods. Commun. Appl. Math. Comput. Sci. **14**(1), 33–64 (2019). https://doi.org/10.2140/camcos.2019.14.33
22. Weiser, M.: Faster SDC convergence on non-equidistant grids by DIRK sweeps. BIT Numerical Math. **55**(4), 1219–1241 (2015). https://doi.org/10.1007/s10543-014-0540-y
23. Schöbel, R., Speck, R.: PFASST-ER: combining the parallel full approximation scheme in space and time with parallelization across the method. Comput. Visualization Sci. **23**(1–4), 12 (2020). https://doi.org/10.1007/s00791-020-00330-5
24. Čaklović, G., Lunet, T., Götschel, S., Ruprecht, D.: Improving Efficiency of Parallel Across the Method Spectral Deferred Corrections. SIAM Journal of Scientific Computing (2024). arXiv:2403.18641
25. Freese, P., Götschel, S., Lunet, T., Ruprecht, D., Schreiber, M.: Parallel performance of shared memory parallel spectral deferred corrections (2024). arXiv:2403.20135
26. Kremling, G., Speck, R.: Convergence of multilevel spectral deferred corrections. Commun. Appl. Math. Comput. Sci. **16**(2), 227–265 (2021)
27. Speck, R., Ruprecht, D., Minion, M., Emmett, M., Krause, R.: Inexact spectral deferred corrections. In: Dickopf, T., Gander, M.J., Halpern, L., Krause, R., Pavarino, L.F. (eds.) Domain Decomposition Methods in Science and Engineering XXII, pp. 389–396. Springer, Cham (2016)
28. Weiser, M., Ghosh, S.: Theoretically optimal inexact spectral deferred correction methods. Commun. Appl. Math. Comput. Sci. **13**(1), 53–86 (2018). https://doi.org/10.2140/camcos.2018.13.53

29. Guesmi, M., Grotteschi, M., Stiller, J.: Assessment of high-order IMEX methods for incompressible flow. Int. J. Numerical Methods in Fluids. **95**(6), 954–978 (2023). https://doi.org/10.1002/fld.5177 https://arxiv.org/abs/https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.5177
30. Hairer, E., Wanner, G., Nørsett, S.P.: Solving Ordinary Differential Equations I. Springer Series in Computational Mathematics, vol. 8. Springer, Berlin, Heidelberg (1993). https://doi.org/10.1007/978-3-540-78862-1
31. Tsitouras, C.: Runge-Kutta pairs of order 5(4) satisfying only the first column simplifying assumption. Comput. Math. Appl. **62**(2), 770–775 (2011). https://doi.org/10.1016/j.camwa.2011.06.002
32. Houwen, P.J., Sommeijer, B.P., Couzy, W.: Embedded diagonally implicit runge-kutta algorithms on parallel computers. Math. Comput. **58**(197), 135–159 (1992)
33. Van Der Houwen, P.J., Sommeijer, B.P.: Parallel iteration of high-order runge-kutta methods with stepsize control. J. Comput. Appl. Math. **29**(1), 111–127 (1990). https://doi.org/10.1016/0377-0427(90)90200-J
34. Huang, J., Jia, J., Minion, M.: Accelerating the convergence of spectral deferred correction methods. J. Comput. Phys. **214**(2), 633–656 (2006). https://doi.org/10.1016/j.jcp.2005.10.004
35. Layton, A.T., Minion, M.L.: Implications of the choice of quadrature nodes for Picard integral deferred corrections methods for ordinary differential equations. BIT Numerical Math. **45**(2), 341–373 (2005). https://doi.org/10.1007/s10543-005-0016-1
36. Guibert, D., Tromeur-Dervout, D.: Parallel deferred correction method for CFD problems. In: Kwon, J.H., Ecer, A., Satofuka, N., Periaux, J., Fox, P. (eds.) Parallel Computational Fluid Dynamics 2006, pp. 131–138. Elsevier Science B.V., Amsterdam (2007). https://doi.org/10.1016/B978-044453035-6/50019-5
37. Christlieb, A.J., Macdonald, C.B., Ong, B.W.: Parallel high-order integrators. SIAM J. Scientific Comput. **32**(2), 818–835 (2010). https://doi.org/10.1137/09075740X
38. Gander, M.J., Lunet, T., Ruprecht, D., Speck, R.: A unified analysis framework for iterative parallel-in-time algorithms. SIAM J. Scientific Comput. **45**(5), 2275–2303 (2023). https://doi.org/10.1137/22M1487163
39. Speck, R.: Algorithm 997: pySDC—Prototyping spectral deferred corrections. ACM Trans. Math. Softw. **45**(3) (2019). https://doi.org/10.1145/3310410
40. Von St. Vieth, B.: JUSUF: Modular tier-2 supercomputing and cloud infrastructure at Jülich Supercomputing Centre. J. Large-scale Res Facilities JLSRF. **7**, 179 (2021). https://doi.org/10.17815/jlsrf-7-179
41. Van Der Pol, B.: LXXXVIII. *On "relaxation-oscillations"*. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science. **2**(11), 978–992 (1926). https://doi.org/10.1080/14786442608564127
42. Pol, B.: The nonlinear theory of electric oscillations. Proceedings of the Institute of Radio Engineers. **22**(9), 1051–1086 (1934). https://doi.org/10.1109/JRPROC.1934.226781
43. Virtanen, P., Gommers, R., Oliphant, T.E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S.J., Brett, M., Wilson, J., Millman, K.J., Mayorov, N., Nelson, A.R.J., Jones, E., Kern, R., Larson, E., Carey, C.J., Polat, İ., Feng, Y., Moore, E.W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E.A., Harris, C.R., Archibald, A.M., Ribeiro, A.H., Pedregosa, F., van Mulbregt, P., SciPy 1.0 Contributors: SciPy 1.0: Fundamental algorithms for scientific computing in Python. Nature Meth. **17**, 261–272 (2020). https://doi.org/10.1038/s41592-019-0686-2
44. Dormand, J.R., Prince, P.J.: A family of embedded Runge-Kutta formulae. J. Comput. Appl. Math. **6**(1), 19–26 (1980). https://doi.org/10.1016/0771-050X(80)90013-3
45. Schnaubelt, E., Wozniak, M., Schöps, S.: Thermal thin shell approximation towards finite element quench simulation. Superconductor Sci. Technol. **36**(4), 044004 (2023). https://doi.org/10.1088/1361-6668/acbeea
46. Bajko, M., Bertinelli, F., Catalan-Lasheras, N., Claudet, S., Cruikshank, P., Dahlerup-Petersen, K., Denz, R., Fessia, P., Garion, C., Jimenez, J., Kirby, G., Lebrun, P., Le Naour, S., Mess, K.-H., Modena, M., Montabonnet, V., Nunes, R., Parma, V., Perin, A., Rijk, G., Rijllart, A., Rossi, L., Schmidt, R., Siemko, A., Strubin, P., Tavian, L., Thiesen, H., Tock, J., Todesco, E., Veness, R., Verweij, A., Walckiers, L., Van Weelderen, R., Wolf, R., Fehér, S., Flora, R., Koratzinos, M., Limon, P., Strait, J.: Report of the task force on the incident of 19th September 2008 at the LHC. Technical report, CERN, Geneva (2009). https://cds.cern.ch/record/1168025

47. Shampine, L.F., Reichelt, M.W.: The MATLAB ODE suite. SIAM J. Scientific Comput. **18**(1), 1–22 (1997). https://doi.org/10.1137/S1064827594276424
48. Ablowitz, M.J., Prinari, B., Trubatch, A.D.: Discrete and Continuous Nonlinear Schrödinger Systems. London Mathematical Society Lecture Note Series. Cambridge University Press, Cambridge (2003). https://doi.org/10.1017/CBO9780511546709
49. Akhmediev, N.N., Eleonskii, V.M., Kulagin, N.E.: Exact first-order solutions of the nonlinear Schrödinger equation. Theoretical Math. Phys. **72**(2), 809–818 (1987). https://doi.org/10.1007/BF01017105
50. Kennedy, C.A., Carpenter, M.H.: Diagonally Implicit Runge-Kutta Methods for Ordinary Differential Equations. A Review. National Aeronautics and Space Administration (2016). https://ntrs.nasa.gov/api/citations/20160005923/downloads/20160005923.pdf
51. Kennedy, C.A., Carpenter, M.H.: Higher-order additive Runge-Kutta schemes for ordinary differential equations. Appl. Numerical Math. **136**, 183–205 (2019). https://doi.org/10.1016/j.apnum.2018.10.007
52. Jackson, K.R.: A survey of parallel numerical methods for initial value problems for ordinary differential equations. IEEE Trans. Magnetics. **27**(5), 3792–3797 (1991). https://doi.org/10.1109/20.104928
53. Weiser, M., Scacchi, S.: Spectral deferred correction methods for adaptive electro-mechanical coupling in cardiac simulation. In: Russo, G., Capasso, V., Nicosia, G., Romano, V. (eds.) Progress in Industrial Mathematics at ECMI 2014, pp. 321–328. Springer, Cham (2016)
54. Weiser, M., Chegini, F.: Adaptive multirate integration of cardiac electrophysiology with spectral deferred correction methods. In: CMBE22 - 7th International Conference on Computational & Mathematical Biomedical Engineering, pp. 528–531 (2022)
55. Chegini, F., Steinke, T., Weiser, M.: Efficient adaptivity for simulating cardiac electrophysiology with spectral deferred correction methods. arXiv e-prints, 2311–07206 (2023). https://doi.org/10.48550/arXiv.2311.07206. arXiv:2311.07206
56. Christlieb, A., Macdonald, C., Ong, B., Spiteri, R.: Revisionist integral deferred correction with adaptive step-size control. Commun. Appl. Math. Comput. Sci. **10**(1), 1–25 (2015). https://doi.org/10.2140/camcos.2015.10.1