HPS Accelerated Spectral Solvers for Time Dependent Problems: Part II, Numerical Experiments

Tracy Babb, Per-Gunnar Martinsson, and Daniel Appelö

1 Introduction

In this chapter describes a highly computationally efficient solver for equations of the form

$$\kappa \frac{\partial u}{\partial t} = \mathscr{L}u(\boldsymbol{x}, t) + h(u, \boldsymbol{x}, t), \quad \boldsymbol{x} \in \Omega, t > 0, \tag{1}$$

with initial data $u(\boldsymbol{x}, 0) = u_0(\boldsymbol{x})$. Here $\mathscr{L}$ is an elliptic operator acting on a fixed domain Ω and h is lower order, possibly nonlinear terms. We take κ to be real or imaginary, allowing for parabolic and Schrödinger type equations. We desire the benefits that can be gained from an implicit solver, such as L-stability and stiff accuracy, which means that the computational bottleneck will be the solution of a sequence of elliptic equations set on Ω. In situations where the elliptic equation to be solved is the same in each time-step, it is highly advantageous to use a *direct* (as opposed to *iterative*) solver. In a direct solver, an approximate solution operator to the elliptic equation is built once. The cost to build it is typically higher than the cost required for a single elliptic solve using an iterative method such as multigrid, but the upside is that after it has been built, each subsequent solve is very fast. In this chapter, we argue that a particularly efficient direct solver to use in this context is a method obtained by combining a multidomain spectral collocation discretization (a

T. Babb · D. Appelö (✉)
University of Colorado, Boulder, CO, USA
e-mail: tracy.babb@colorado.edu; daniel.appelo@colorado.edu

P.-G. Martinsson
University of Texas, Austin, TX, USA
e-mail: pgm@ices.utexas.edu

S. J. Sherwin et al. (eds.), *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2018*, Lecture Notes in Computational Science and Engineering 134, https://doi.org/10.1007/978-3-030-39647-3_9

so-called "patching method", see e.g. Ch. 5.13 in [3]) with a nested dissection type solver. It has recently been demonstrated [1, 7, 12] that this combined scheme, which we refer to as a "Hierarchial Poincaré–Steklov (HPS)" solver, can be used with very high local discretization orders (up to $p = 20$ or higher) without jeopardizing either speed or stability, as compared to lower order methods.

In this chapter, we investigate the stability and accuracy that is obtained when combining high-order time-stepping schemes with the HPS method for solving elliptic equations. We restrict attention to relatively simple geometries (mostly rectangles). The method can without substantial difficulty be generalized to domains that can naturally be expressed as a union of rectangles, possibly mapped via curvilinear smooth parameter maps.

A longer version of this chapter with additional details is available at [2]. Also note that the conclusions are deferred to Part II of this paper (same issue).

2 The Hierarchical Poincaré–Steklov Method

In this section, we describe a computationally efficient and highly accurate technique for solving an elliptic PDE of the form

$$\begin{aligned} [Au](\boldsymbol{x}) &= g(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega, \\ u(\boldsymbol{x}) &= f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma, \end{aligned} \tag{2}$$

where Ω is a domain with boundary Γ, and where A is a variable coefficient elliptic differential operator

$$\begin{aligned} [Au](\boldsymbol{x}) = -c_{11}(\boldsymbol{x})[\partial_1^2 u](\boldsymbol{x}) - 2c_{12}(\boldsymbol{x})[\partial_1\partial_2 u](\boldsymbol{x}) - c_{22}(\boldsymbol{x})[\partial_2^2 u](\boldsymbol{x}) \\ + c_1(\boldsymbol{x})[\partial_1 u](\boldsymbol{x}) + c_2(\boldsymbol{x})[\partial_2 u](\boldsymbol{x}) + c(\boldsymbol{x})\, u(\boldsymbol{x}) \end{aligned}$$

with smooth coefficients. In the present context, (2) represents an elliptic solve that is required in an implicit time-descretization technique of a parabolic PDE, as discussed in Sect. 1. For simplicity, let us temporarily suppose that the domain Ω is rectangular; the extension to more general domains is discussed in Remark 1.

Our ambition here is merely to provide a high level description of the method; for implementation details, we refer to [1, 2, 7–9, 12, 13].

2.1 Discretization

We split the domain Ω into $n_1 \times n_2$ boxes, each of size $h \times h$. Then on each box, we place a $p \times p$ tensor product grid of Chebyshev nodes, as shown in Fig. 1. We use collocation to discretize the PDE (2). With $\{\boldsymbol{x}_i\}_{i=1}^N$ denoting the collocation points,

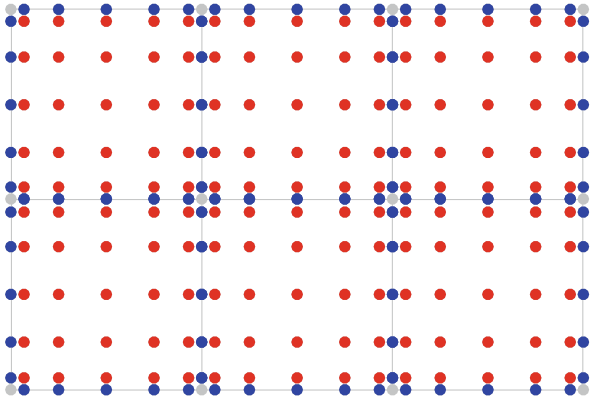

Fig. 1 The domain Ω is split into $n_1 \times n_2$ squares, each of size $h \times h$. In the figure, $n_1 = 3$ and $n_2 = 2$. Then on each box, a $p \times p$ tensor product grid of Chebyshev nodes is placed, shown for $p = 7$. At red nodes, the PDE (2) is enforced via collocation of the spectral differentiation matrix. At the blue nodes, we enforce continuity of the normal fluxes. Observe that the corner nodes (gray) are excluded from consideration

the vector u that represents our approximation to the solution u of (2) is given simply by $\mathsf{u}(i) \approx u(\boldsymbol{x}_i)$. We then discretize (2) as follows:

1. For each collocation node that is *internal* to a box (red nodes in Fig. 1), we enforce (2) by directly collocating the spectral differential operator supported on the box, as described in, e.g., Trefethen [15].
2. For each collocation node on an *edge* between two boxes (blue nodes in Fig. 1), we enforce that the normal fluxes across the edge be continuous. For instance, for a node $\boldsymbol{x}_i$ on a vertical line, we enforce that $\partial u/\partial x_1$ is continuous across the edge by equating the values for $\partial u/\partial x_1$ obtained by spectral differentiation of the boxes to the left and to the right of the edge. For an edge node that lies on the external boundary Γ, simply evaluate the normal derivative at the node, as obtained by spectral differentiation in the box that holds the node.
3. All corner nodes (gray in Fig. 1) are dropped from consideration. For an elliptic operator of the form (2) with $c_{12} = 0$, it turns out that these values do not contribute to any of the spectral derivatives on the interior nodes, which means that the method without corner nodes is mathematically equivalent to the method with corner nodes, see [5, Sec. 2.1] for details. When $c_{12} \neq 0$, one must in order to drop the corner nodes include an extrapolation operator when evaluating the terms involving the spectral representation of the mixed derivative $\partial^2 u/\partial x_1 \partial x_2$. This may lead to a slight drop in the order of convergence, but the difference is hardly noticeable in practice, and the exclusion of corner nodes greatly simplifies the implementation of the method.

Since we exclude the corner nodes from consideration, the total number of nodes in the grid equals $N = (p-2)\big(p\,n_1\,n_2 + n_1 + n_2\big) \approx p^2\,n_1\,n_2$. The discretization

procedure described then results in an $N \times N$ matrix A. For a node i, the value of $\mathsf{A}(i,:)\mathsf{u}$ depends on what type of node i is:

$$\mathsf{A}(i,:)\mathsf{u} \approx \begin{cases} [Au](\boldsymbol{x}_i) \text{ for any interior (red) node,} \\ 0 \text{ for any edge node (blue) not on } \Gamma, \\ \partial u/\partial n \text{ for any edge node (blue) on } \Gamma. \end{cases}$$

This matrix A can be used to solve BVPs with a variety of different boundary conditions, including Dirichlet, Neumann, Robin, and periodic [12].

In many situations, a simple uniform mesh of the type shown in Fig. 1 is not optimal, since the regularity in the solution may vary greatly, due to corner singularities, localized loads, etc. The HPS method can easily be adapted to handle local refinement. The essential difficulty that arises is that when boxes of different sizes are joined, the collocation nodes along the joint boundary will not align. It is demonstrated in [1, 5] that this difficulty can stably and efficiently be handled by incorporating local interpolation operators.

2.2 A Hierarchical Direct Solver

A key observation in previous work on the HPS method is that the sparse linear system that results from the discretization technique described in Sect. 2.1 is particularly well suited for direct solvers, such as the well-known multifrontal solvers that compute an LU-factorization of a sparse matrix. The key is to minimize fill-in by using a so called nested dissection ordering [4, 6]. Such direct solvers are very powerful in a situation where a sequence of linear systems with the same coefficient matrix needs to be solved, since each solve is very fast once the coefficient matrix has been factorized. This is precisely the environment under consideration here. The particular advantage of combining the multidomain spectral collocation discretization described in Sect. 2.1 is that the time required for factorizing the matrix is *independent* of the local discretization order. As we will see in the numerical experiments, this enables us to attain both very high accuracy, and very high computational efficiency.

Remark 1 (General Domains) For simplicity we restrict attention to rectangular domains in this chapter. The extension to domains that can be mapped to a union of rectangles via smooth coordinate maps is relatively straight-forward, since the method can handle variable coefficient operators [12, Sec. 6.4]. Some care must be exercised since singularities may arise at intersections of parameter maps, which may require local refinement to maintain high accuracy.

The direct solver described exactly mimics the classical nested dissection method, and has the same asymptotic complexity of $O(N^{1.5})$ for the "build" (or "factorization") stage, and then $O(N \log N)$ cost for solving a system once the

coefficient matrix has been factorized. Storage requirements are also $O(N \log N)$. A more precise analysis of the complexity that takes into account the dependence on the order p of the local discretization shows [1] that $T_{\text{build}} \sim N\, p^4 + N^{1.5}$, and $T_{\text{solve}} \sim N\, p^2 + N \log N$.

3 Time-Stepping Methods

For high-order time-stepping of (1), we use the so called Explicit, Singly Diagonally Implicit Runge–Kutta (ESDIRK) methods. These methods have a Butcher diagram with a constant diagonal γ and are of the form

$$
\begin{array}{c|cccccc}
0 & 0 & & & & & \\
2\gamma & \gamma & \gamma & & & & \\
c_3 & a_{3,1} & a_{3,2} & \gamma & & & \\
\vdots & \vdots & \vdots & \ddots & \ddots & & \\
c_{s-1} & a_{s-1,1} & a_{s-1,2} & a_{s-1,3} & \cdots & \gamma & \\
1 & b_1 & b_2 & b_3 & \cdots & b_{s-1} & \gamma \\
\hline
 & b_1 & b_2 & b_3 & \cdots & b_{s-1} & \gamma
\end{array}
$$

ESDIRK methods offer the advantages of stiff accuracy and L-stability. They are particularly attractive when used in conjunction with direct solvers since the elliptic solve required in each stage involves the same coefficient matrix $(I - h\gamma\mathscr{L})$, where h is the time-step.

In general we split the right hand side of (1) into a stiff part, $F^{[1]}$, that will be treated implicitly using ESDIRK methods, and a part, $F^{[2]}$, that will be treated explicitly (with a Butcher table denoted $\hat{c}$, $\hat{A}$, and $\hat{b}$). Precisely we will use the Additive Runge–Kutta (ARK) methods by Carpenter and Kennedy [11], of order 3, 4 and 5.

We may choose to formulate the Runge–Kutta method in terms of either solving for slopes or solving for stage solutions. We denote these the k_i formulation and the u_i formulation, respectively. When solving for slopes the stage computation is

$$k_i^n = F^{[1]}(t_n + c_i \Delta t, u^n + \Delta t \sum_{j=1}^{s} a_{ij} k_j^n + \Delta t \sum_{j=1}^{s} \hat{a}_{ij} l_j^n), \quad i = 1, \ldots, s, \tag{3}$$

$$l_i^n = F^{[2]}(t_n + c_i \Delta t, u^n + \Delta t \sum_{j=1}^{s} a_{ij} k_j^n + \Delta t \sum_{j=1}^{s} \hat{a}_{ij} l_j^n), \quad i = 1, \ldots, s. \tag{4}$$

Note that the explicit nature of (4) is encoded in the fact that the elements on the diagonal and above in $\hat{A}$ are zero. Once the slopes have been computed the solution

at the next time-step is assembled as

$$u^{n+1} = u^n + \Delta t \sum_{j=1}^{s} b_j k_j^n + \Delta t \sum_{j=1}^{s} \hat{b}_j l_j^n. \tag{5}$$

If the method is instead formulated in terms of solving for the stage solutions the implicit solves take the form

$$u_i^n = u^n + \Delta t \sum_{j=1}^{s} \left(a_{ij} F^{[1]}(t_n + c_j \Delta t, u_j^n) + \hat{a}_{ij} F^{[2]}(t_n + c_j \Delta t, u_j^n) \right),$$

and the explicit update for u^{n+1} is given by

$$u^{n+1} = u^n + \Delta t \sum_{j=1}^{s} b_j (F^{[1]}(t_n + c_j \Delta t, u_j^n) + F^{[2]}(t_n + c_j \Delta t, u_j^n)).$$

The two formulations are algebraically equivalent but offer different advantages. For example, when working with the slopes we do not observe (see experiments presented in the second part of this paper) any order reduction due to time-dependent boundary conditions (see e.g. the analysis by Rosales et al. [14]). On the other hand and as discussed in some detail below, in solving for the slopes the HPS framework requires an additional step to enforce continuity.

We note that it is generally preferred to solve for the slopes when implementing implicit Runge–Kutta methods, particularly when solving very stiff problems where the influence of roundoff (or solver tolerance) errors can be magnified by the Lipschitz constant when solving for the stages directly.

Remark 2 The HPS method for elliptic solves was previously used in [10], which considered a linear hyperbolic equation

$$\frac{\partial u}{\partial t} = \mathscr{L} u(\boldsymbol{x}, t), \quad \boldsymbol{x} \in \Omega, t > 0,$$

where $\mathscr{L}$ is a skew-Hermitian operator. The evolution of the numerical solution can be performed by approximating the propagator $\exp(\tau \mathscr{L}) : L^2(\Omega) \to L^2(\Omega)$ via a rational approximation

$$\exp(\tau \mathscr{L}) \approx \sum_{m=-M}^{M} b_m (\tau \mathscr{L} - \alpha_m)^{-1}.$$

If application of $(\tau \mathscr{L} - \alpha_m)^{-1}$ to the current solution can be reduced to the solution of an elliptic-type PDE it is straightforward to apply the HPS scheme to each term in the approximation. A drawback with this approach is that multiple operators must

be formed and it is also slightly more convenient to time step non-linear equations using the Runge–Kutta methods we use here.

There are two modifications to the HPS algorithm that are necessitated by the use of ARK time integrators, we discuss these in the next two subsections.

3.1 Neumann Data Correction in the Slope Formulation

In the HPS algorithm the PDE is enforced on interior nodes and continuity of the normal derivative is enforced on the leaf boundary. Now, due to the structure of the update formula (5), if at some time u^n has an error component in the null space of the operator that is used to solve for a slope k_i, then this will remain throughout the solution process. Although this does not affect the stability of the method it may result in loss of relative accuracy as the solution evolves. As a concrete example consider the heat equation

$$u_t = u_{xx}, \quad x \in [0, 2], t > 0, \tag{6}$$

with the initial data $u(x, 0) = 1 - |x - 1|$, and with homogenous Dirichlet boundary conditions. We discretize this on two leaves which we denote by α and β.

Now in the k_i formulation, we solve several PDEs for the k_i values and update the solution as

$$u^{n+1} = u^n + \Delta t \sum_{j=1}^{s} b_j k_j^n.$$

Here, even though the individual slopes have continuous derivatives the kink in u^n will be propagated to u^{n+1}. In this particular example we would end up with the incorrect steady state solution $u(x, t) = 1 - |x - 1|$.

Fortunately, this can easily be mitigated by adding a consistent penalization of the jump in the derivative of the solution during the merging of two leaves (for details see Section 4 in [1]). That is, if we denote the jump by $[[\cdot]]$ we replace the condition $0 = [[Tk + h^k]]$ where Tk is the derivative from the homogenous part and h^k is the derivative for the particular solution (of the slope) by the condition $[[Tk + h^k - \Delta t^{-1} h^u]] = 0$. In comparison to [1] we get the slightly modified merge formula

$$\mathsf{k}_{i,3} = \left(\mathsf{T}^{\alpha}_{3,3} - \mathsf{T}^{\beta}_{3,3}\right)^{-1}\left(\mathsf{T}^{\beta}_{3,2}\mathsf{k}_{i,2} - \mathsf{T}^{\alpha}_{3,1}\mathsf{k}_{i,1} + \mathsf{h}^{k,\beta}_{3} - \mathsf{h}^{k,\alpha}_{3} - \frac{1}{\Delta t}(\mathsf{h}^{u,\alpha}_{3} - \mathsf{h}^{u,\beta}_{3})\right),$$

along with the modified equation for the fluxes of the particular solution on the parent box

$$\begin{bmatrix} \mathsf{v}_1 \\ \mathsf{v}_2 \end{bmatrix} = \left(\begin{bmatrix} \mathsf{T}^{\alpha}_{1,1} & 0 \\ 0 & \mathsf{T}^{\beta}_{2,2} \end{bmatrix} + \begin{bmatrix} \mathsf{T}^{\alpha}_{1,3} \\ \mathsf{T}^{\beta}_{2,3} \end{bmatrix} \left(\mathsf{T}^{\alpha}_{3,3} - \mathsf{T}^{\beta}_{3,3}\right)^{-1} \left[-\mathsf{T}^{\alpha}_{3,1} \mid \mathsf{T}^{\beta}_{3,2} \right] \right) \begin{bmatrix} \mathsf{k}_{i,1} \\ \mathsf{k}_{i,2} \end{bmatrix} + \begin{bmatrix} \mathsf{h}^{k,\alpha}_{1} \\ \mathsf{h}^{k,\beta}_{2} \end{bmatrix} + \begin{bmatrix} \mathsf{T}^{\alpha}_{1,3} \\ \mathsf{T}^{\beta}_{2,3} \end{bmatrix} \left(\mathsf{T}^{\alpha}_{3,3} - \mathsf{T}^{\beta}_{3,3}\right)^{-1} \left(\mathsf{h}^{\beta}_{3} - \mathsf{h}^{\alpha}_{3} - \frac{1}{\Delta t}(\mathsf{h}^{u,\alpha}_{3} - \mathsf{h}^{u,\beta}_{3})\right).$$

Due to space we must refer to [1] for a detailed discussion of these equations. Briefly, $h^{k,\alpha}$ and $h^{k,\beta}$ above denote the spectral derivative on each child's boundary for the particular solution to the PDE for k_i and are already present in [1]. However, $h^{u,\alpha}$ and $h^{u,\beta}$, which denote the spectral derivative of u^n on the boundary from each child box, are new additions.

The above initial data is of course extreme but we note that the problem persists for any non-polynomial initial data with the size of the (stationary) error depending on resolution of the simulation. We further note that the described penalization removes this problem without affecting the accuracy or performance of the overall algorithm.

Remark 3 Although for linear constant coefficient PDE it may be possible to project the initial data in a way so that interior affine functions do not cause the difficulty above, for greater generality, we have chosen to enforce the extra penalization throughout the time evolution.

Remark 4 When utilizing the u_i formulation in a purely implicit problem we do not encounter the difficulty described above. This is because we enforce continuity of the derivative in u^n_s when solving

$$(I - \Delta t \gamma \mathscr{L}) u^n_s = u^n + \Delta t \mathscr{L}\Big(\sum_{j=1}^{s-1} a_{sj} u^n_j \Big) + \Delta t \sum_{j=1}^{s-1} a_{sj} g(x, t_n + c_j \Delta t),$$

followed by the update $u^{n+1} = u^n_s$.

3.2 Enforcing Continuity in the Explicit Stage

The second modification is to the first explicit stage in the k_i formulation. Solving a problem with no forcing this stage is simply

$$k^n_1 = \mathscr{L}(u_n).$$

When, for example, $\mathscr{L}$ is the Laplacian, we must evaluate it on all nodes on the interior of the physical domain. This includes the nodes on the boundary between two leafs where the spectral approximation to the Laplacian can be different if we use values from different leaves. The seemingly obvious choice, replacing the Laplacian on the leaf boundary by the average, leads to instability. However, stability can be restored if we enforce $k_1^n = \mathscr{L}(u_n)$ on the interior of each leaf and continuity of the derivative across each leaf boundary. Algorithmically, this is straightforward as these are the same conditions that are enforced in the regular HPS algorithm, except in this case we simply have an identity equation for k_1 on the interior nodes instead of a full PDE.

Although it is convenient to enforce continuity of the derivative using the regular HPS algorithm it can be done in a more efficient fashion by forming a separate system of equations involving only data on the leaf boundary nodes. In a single dimension on a discretization with n leafs this reduces the work associated with enforcing continuity of the derivative across leaf boundary nodes from solving $n \times (p-1) - 1$ equations for $n \times (p-1) - 1$ unknowns to solving a tridiagonal system of equations $n - 1$ equations for $n - 1$ unknowns.

In two dimensions the system is slightly different, but if we have $n \times n$ leafs with $p \times p$ Chebyshev nodes on each leaf then eliminating the explicit equations for the interior nodes reduces the system to $(p-2) \times 2n$ independent tridiagonal systems of $n-1$ equations with $n-1$ unknowns for a total of $(p-2) \times 2n \times (n-1)$ equations with $(p-2) \times 2n \times (n-1)$ unknowns.

When the u_i formulation is used for a fully implicit problem the intermediate stage values still requires us to evaluate $\mathscr{L}u^n$, but this quantity only enters through the body load in the intermediate stage PDEs. The explicit first stage in this formulation is simply $u_1^n = u^n$. Furthermore, while we must calculate

$$u^{n+1} = u^n + \Delta t \mathscr{L}\Big(\sum_{j=1}^{s} a_{sj} u_j^n\Big),$$

this is equivalent to u_s^n since $b_j = a_{sj}$ and we simply take $u^{n+1} = u_s^n$.

When both explicit and implicit terms are present, we proceed differently. Now, the values of u_i^n look almost identical to the implicit case and we still avoid the problem of an explicit "solve" in u_1^n, but we also have

$$u^{n+1} = u^n + \Delta t \sum_{j=1}^{s} b_j (F^{[1]}(t_n + c_j \Delta t, u_j^n) + F^{[2]}(t_n + c_j \Delta t, u_j^n))$$

The ESDIRK method has the property that $b_j = a_{sj}$, but for the explicit Runge–Kutta method we have $b_j \neq \hat{a}_{sj}$. When the explicit operator $F^{[2]}$ does not contain partial derivatives we need not enforce continuity of the derivative and can simply

reformulate the method as

$$u^{n+1} = u_s^n + \Delta t \sum_{j=1}^{s} (a_{sj} - \hat{a}_{sj}) F^{[2]}(t_n + c_j \Delta t, u_j^n)$$

4 Boundary Conditions

The above description for Runge–Kutta methods does not address how to impose boundary conditions for a system of ODEs resulting from a discretization of a PDE. In particular, the different formulations incorporate boundary conditions in slightly different ways.

In this work we consider Dirichlet, Neumann, and periodic boundary conditions. For periodic boundary conditions the intermediate stage boundary conditions are enforced to be periodic for both formulations. As the k_i stage values are approximations to the time derivative of u, the imposed Dirichlet boundary conditions for $x \in \Gamma$ are $k_i^n = u_t(x, t_n + c_i \Delta t)$. When solving for u_i one may attempt to enforce boundary conditions using $u_i = u(x, t + c_i \Delta t), x \in \Gamma$. However, as demonstrated in part two of this series and discussed in detail in [14], this results in order reduction for time dependent boundary conditions.

In the HPS algorithm, Neumann or Robin boundary conditions are mapped to Dirichlet boundary conditions using the linear Dirichlet to Neumann operator as discussed for example in [1].

References

1. Babb, T., Gillman, A., Hao, S., Martinsson, P.: An accelerated Poisson solver based on multidomain spectral discretization. BIT Numer. Math. **58**, 851–879 (2018)
2. Babb, T., Martinsson, P.-G., Appelö, D.: HPS accelerated spectral solvers for time dependent problems (2018). arXiv:1811.04555
3. Canuto, C., Hussaini, M.Y., Quarteroni, A., Zang, T.A.: Spectral Methods: Evolution to Complex Geometries and Applications to Fluid Dynamics. Springer, Berlin (2007)
4. Duff, I., Erisman, A., Reid, J.: Direct Methods for Sparse Matrices. Oxford University Press, Oxford (1989)
5. Geldermans, P., Gillman, A.: An adaptive high order direct solution technique for elliptic boundary value problems. SIAM J. Sci. Comput. **41**(1), A292–A315 (2019). arXiv:1710.08787
6. George, A.: Nested dissection of a regular finite element mesh. SIAM J. Numer. Anal. **10**, 345–363 (1973)
7. Gillman, A., Martinsson, P.: A direct solver with $\mathcal{O}(N)$ complexity for variable coefficient elliptic PDEs discretized via a high-order composite spectral collocation method. SIAM J. Sci. Comput. **36**, A2023–A2046 (2014). arXiv:1307.2665
8. Gillman, A., Barnett, A., Martinsson, P.-G.: A spectrally accurate direct solution technique for frequency-domain scattering problems with variable media. BIT Numer. Math. **55**, 141–170 (2015)

9. Hao, S., Martinsson, P.: A direct solver for elliptic PDEs in three dimensions based on hierarchical merging of Poincaré-Steklov operators. J. Comput. Appl. Math. **308**, 419–434 (2016)
10. Haut, T., Babb, T., Martinsson, P., Wingate, B.: A high-order scheme for solving wave propagation problems via the direct construction of an approximate time-evolution operator. IMA J. Numer. Anal. **36**, 688–716 (2016)
11. Kennedy, C., Carpenter, M.: Additive Runge–Kutta schemes for convection-diffusion-reaction equations. Appl. Numer. Math. 44, 139–181 (2003)
12. Martinsson, P.: A direct solver for variable coefficient elliptic PDEs discretized via a composite spectral collocation method. J. Comput. Phys. **242**, 460–479 (2013)
13. Martinsson, P.: The hierarchical Poincaré-Steklov (HPS) solver for elliptic PDEs: a tutorial (2015). arXiv:1506.01308
14. Rosales, R., Seibold, B., Shirokoff, D., Zhou, D.: Order reduction in high-order Runge–Kutta methods for initial boundary value problems (2017). arXiv:1712.00897
15. Trefethen, L.: Spectral Methods in Matlab. SIAM, Philadelphia (2000)