EFFICIENT SOLUTION OF FULLY IMPLICIT RUNGE–KUTTA METHODS FOR LINEAR WAVE EQUATIONS[*]

AMAN RANI[†], PIETER GHYSELS[‡], VICTORIA HOWLE[†], KATHARINE LONG[†], AND MICHAL OUTRATA[§]

Abstract. We focus on time integration of time-dependent linear hyperbolic PDEs, where implicit Runge–Kutta (IRK) time stepping is an increasingly popular technique. However, this approach gives rise to linear block systems with a Kronecker product structure, involving the Butcher table and the mass and stiffness matrices. Taking the wave equation as the canonical example, the system arising from IRK time stepping is highly ill-conditioned but we can exploit the block structure. If N is the number of degrees of freedom for the discretization of the Laplace operator, then the resulting system matrix is a block $s \times s$ matrix where each block is of size $O(N) \times O(N)$, and s is the number of IRK stages. We reformulate the large $O(Ns) \times O(Ns)$ block structured linear system as a Sylvester matrix equation. This leads to s separate systems of order $O(N) \times O(N)$, these smaller systems are efficiently handled with the subsolves replaced by a single AMG V-cycle. We demonstrate the effectiveness of our approach on a 2D wave problems. Our experiments show that our approach not only reduces runtime but also requires fewer AMG V-cycles compared to traditional methods. As the number of Runge–Kutta stages increases and the mesh is refined, the Sylvester approach proves to be at least twice as fast as other existing methods, while also requiring fewer AMG V-cycles. We also introduce a block lower triangular preconditioner based on minimization of $\|L^{-1}A - I\|_2$ over the lower triangular matrices L (A being the Butcher table), which improves on an existing method based on minimization of $\kappa(L^{-1}A)$.

Key words. Preconditioning, Time Integrator, Implicit Runge–Kutta Methods, Hyperbolic PDEs

1. Introduction.

Explicit time steppers are a popular choice for hyperbolic PDEs but they put constraints on the time step. As we dive into scientific modeling and simulation, we encounter stiff systems arising from any spatial discretization (FEM, FDM, spectral, etc.) of the given hyperbolic/parabolic PDE. Accurately integrating these stiff equations demands very small time steps or the use of unconditionally stable time integrators. Implicit Runge–Kutta (IRK) methods stand out for handling such challenges as they avoid the Dahlquist barrier [10], enabling them to achieve higher-order accuracy without limitations. IRK methods, known for their stability and high order, have been less utilized for PDEs due to the challenge of handling large, strongly coupled linear systems. Recently, we have seen a renewed interest in tackling some of these challenges, focusing both on the practical implementation (see, among others, [1, 3, 9, 13, 22, 30, 32, 33, 38], as well as on the analysis (see, among others, [11, 14, 15]).

In [9, 28, 33, 39], the preconditioners introduced for these large systems are based on the diagonal or (more often) triangular approximation of the Butcher matrix A or its factors, aiming to overcome challenges associated with stage-coupled systems and to improve the scalability of the solution process. For an s-stage Runge–Kutta scheme and a spatial operator discretized with N degrees of freedom, the resulting

[*]Submitted to the editors 07/15/2024

Funding: This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) Program through the FASTMath Institute under Contract No. DE-AC02-05CH11231 at Lawrence Berkeley National Laboratory. This work was also supported in part by the PRIMUS/25/SCI/022 grant at Charles University.

[†]Texas Tech University. Lubbock, TX, USA. (aman.rani@ttu.edu)

[‡]Lawrence Berkeley National Laboratory. Berkeley, CA, USA.

[§]Charles University, Prague, Czechia.

linear system of stage unknowns is of size $O(sN) \times O(sN)$. Triangular approximation of the Butcher table leads to a block triangular linear system with s block-rows and block-columns with blocks of size $O(N)$, which can be solved using forward/backward substitution (for lower or upper triangular approximation respectively). This substitution phase is sequential, but each of the s subsolves can use standard solvers for mass and/or stiffness matrices. In this work, we reformulate the large linear system as a Sylvester matrix equation. This allows us to use well established linear algebra techniques to tackle this problem [36]. We use a transformation based on an eigendecomposition of the inverse transpose of the Butcher matrix to transform the large $O(sN)$ system into a block diagonal matrix, resulting in s separate smaller linear systems. Similar approaches were already proposed by Butcher [6] and Bickart [5], see also [18]. Here we focus on the efficient solution of the resulting shifted linear systems and how to deal with the complex arithmetic resulting from the eigendecomposition. When first introduced in [6], it was argued that this transformation to block diagonal form also allows for more parallelism, since the s subsolves can be performed concurrently. However, in order to exploit this, a costly matrix redistribution might be required. We recognize that the s linear systems can be solved efficiently, with standard solvers for mass/stiffness matrices, by fusing the matrix-vector products and the preconditioner applications and exploiting sparse matrix multiplication with multiple vectors (sparse matrix times dense matrix), instead of sparse matrix times vector multiplication. Due to its low arithmetic intensity, sparse matrix-vector multiplication is typically memory bandwidth limited. The sparse matrix times dense matrix kernel has a much higher arithmetic intensity — increasing with the number of vectors/stages s — and can achieve much higher performance on modern hardware. This is nicely illustrated by the roofline model [45]. Likewise, the preconditioner can be applied to multiple vectors at once. Unfortunately, not all state-of-the-art preconditioners support application to multiple vectors at once. Most sparse direct solvers, including SuperLU_Dist [23] do support this feature, but for instance pyAMG [4] does not.

A novel preconditioning approach for IRK systems was introduced in [38], with a follow-up paper [37] describing the non-linear setting. We have not yet compared our Sylvester reformulation against their approach; this comparison will be examined in a forthcoming paper.

Our main contributions in this work are:

- We show how the large block-structured linear systems from IRK methods can be reformulated as a Sylvester matrix equation, which can be solved with standard linear algebra techniques.
- We show how to solve the resulting set of smaller shifted linear systems efficiently by exploiting sparse matrix times dense matrix operations.
- We discuss the issues arising from complex arithmetic required to solve the Sylvester equation and how to deal with them in simulation frameworks that cannot handle complex-valued subsolves.
- We also present a new preconditioner based on a lower triangular approximation of the Butcher table called $\mathcal{P}_{TAI}$, which improves on those in [39, 33]. It is easier to implement compared to [33] and outperforms those in [39, 33], see section 4 for comparisons of $\mathcal{P}_{TAI}, \mathcal{P}_{LD}$ and $\mathcal{P}_{\kappa}$.

We focus on Gauss–Legendre IRK methods because these methods are A-stable, have an optimal order-to-stage ratio (order of accuracy $2s$) and also preserve quadratic invariants of the (spatially discretized) system [17]. An energy functional is conserved during the exact evolution of the wave equation, and it is desirable that a numerical method respect this conservation property. Upon spatial discretization the energy functional is approximated by a quadratic form acting on the solution vector. With a quadratic invariant preserving method such as Gauss–Legendre or Lobatto IIS, the value of this quadratic form remains constant (to within the inevitable roundoff error) during timestepping.

We use the following notation. Matrices are denoted by capital letters, lower case letters refer to scalars, while lower case bold letters refer to vectors (except when referring to columns of a matrix). The Kronecker product (sometimes called the matrix outer product) is denoted by $\otimes$ and the $\mathrm{vec}(\cdot)$ operator, defined as

$$\mathrm{vec}(K) = \mathrm{vec}\left(\begin{bmatrix}K_0 \dots K_s\end{bmatrix}\right) = \begin{bmatrix}K_0\\ \vdots \\ K_s\end{bmatrix} = \boldsymbol{k}\,, \tag{1.1}$$

where K_i denotes column i of matrix K, reshapes a matrix to a column vector by stacking the columns. Since we store matrices in column-major ordering, this reshaping can be done without memory copies.

The remainder of the paper is outlined as follows. In section 2 we recall the general $s-$stage implicit Runge–Kutta method, the Butcher table and the wave equation. Section 3 describes the transformation to block diagonal form using the Sylvester matrix reformulation and how to solve the resulting shifted linear systems. Section 4 shows the preconditioner analysis and performance results, including comparison with several other preconditioners. Finally, we conclude the paper in section 5.

2. Implicit Runge–Kutta Time Stepping for the Wave Equation.

The general $s-$stage IRK method for $\boldsymbol{u'} = \boldsymbol{\phi}(t, \boldsymbol{u})$ has the form

$$\begin{aligned} \boldsymbol{u_{n+1}} &= \boldsymbol{u_n} + h_t \sum_{i=1}^{s} b_i \boldsymbol{k_i}, \quad \text{with} \\ \boldsymbol{k_i} &= \boldsymbol{\phi}\big(t_n + c_i h_t, \boldsymbol{u_n} + h_t \sum_{j=1}^{s} a_{ij} \boldsymbol{k_j}\big), \qquad i = 1, \cdots, s, \end{aligned}$$

which is summarized by the Butcher table

$$\begin{array}{c|ccc} c_1 & a_{11} & \dots & a_{1s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & \dots & a_{ss} \\ \hline & b_1 & \dots & b_s \end{array} = \begin{array}{c|c} \boldsymbol{c} & A \\ \hline & \boldsymbol{b}^T \end{array}, \tag{2.1}$$

where $A \in \mathbb{R}^{s\times s}, \boldsymbol{b} \in \mathbb{R}^s, \boldsymbol{c} \in \mathbb{R}^s$. We use the wave equation

$$\begin{aligned} u_{tt} - c^2 \triangle u &= b(\boldsymbol{x}, t), && \text{in } \Omega \times (0, T], \\ \hat{\boldsymbol{n}} \cdot \nabla u &= 0, && \text{on } \partial\Omega, \\ u(\boldsymbol{x}, 0) &= f(\boldsymbol{x}), && \text{in } \Omega, \\ u_t(\boldsymbol{x}, 0) &= g(\boldsymbol{x}), && \text{in } \Omega, \end{aligned} \tag{2.2}$$

for the exposition of our method and the connected ideas, and explore more general setting in Section 4.4. In principle, the derivations below apply to any second-order linear hyperbolic PDE without time-dependent coefficients (even if, e.g., discretized with a Runge-Kutta-Nyström method). While the generalization for time-dependent coefficients and/or for nonlinear problems is practically important, it goes beyond the scope of this manuscript and will be treated in a separate manuscript.

Denoting $u_t = v$, we introduce the vector $\boldsymbol{w}$, concatenating u and v, and rewrite the above system (2.2) as a first order time dependent PDE

$$\begin{aligned} \boldsymbol{w}_t &= \mathcal{B}\boldsymbol{w} + \hat{\boldsymbol{b}}, && \text{in } \Omega \times (0, T], \\ \boldsymbol{c}^T \boldsymbol{w} &= \boldsymbol{0}, && \text{on } \partial\Omega, \\ \boldsymbol{w}(\boldsymbol{x}, 0) &= \boldsymbol{w_0}(\boldsymbol{x}), && \text{in } \Omega, \end{aligned}$$

where $\boldsymbol{w} = \begin{pmatrix} u \\ v \end{pmatrix}$, $\mathcal{B} = \begin{pmatrix} O & I \\ c^2\triangle & O \end{pmatrix}$, $\hat{\boldsymbol{b}} = \begin{pmatrix} 0 \\ b \end{pmatrix}$, $\boldsymbol{c}^T = \begin{pmatrix} \hat{\boldsymbol{n}} \cdot \nabla \\ 0 \end{pmatrix}$ and $\boldsymbol{w}_0 = \begin{pmatrix} f \\ g \end{pmatrix}$. Converting to weak form and discretizing using the standard finite element setting, we get the following linear system

$$\bar{M}\mathbf{w}_t(t) = B\mathbf{w}(t) + \mathbf{b}^{(ST)}(t), \tag{2.3}$$

where the vector $\mathbf{b}^{(ST)}$ aggregates the source terms stemming from the FEM discretization of the original functions $b(\boldsymbol{x}, t)$, $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ in (2.2) and $\bar{M} = \begin{bmatrix} M & O \\ O & M \end{bmatrix}$ and $B = \begin{bmatrix} O & M \\ -E & O \end{bmatrix}$. M and E are the mass and stiffness matrices given as

$$M_{ij} = \int \phi_i \phi_j \ , \qquad E_{ij} = \int \nabla\phi_i \cdot \nabla\phi_j \ , \tag{2.4}$$

where $\{\phi_j\}$ are the finite-element basis functions. For the above linear system (2.3), we use an IRK method (2.1), obtaining the stage equations

$$\bar{M}\boldsymbol{k_i} = B\boldsymbol{w_n} + h_t \sum_{j=1}^{s} a_{ij} B\boldsymbol{k_j} + \boldsymbol{b}_i^{(ST)}, \qquad i = 1, \dots, s,$$

where the vectors $\boldsymbol{b}_i^{(ST)}$ correspond to evaluations of $\mathbf{b}^{(ST)}$ at the corresponding time-points based on the chosen Runge-Kutta method. We rewrite this system using the block notation

$$\begin{bmatrix} \bar{M} - h_t a_{11} B & -h_t a_{12} B & \dots & -h_t a_{1s} B \\ -h_t a_{21} B & \bar{M} - h_t a_{22} B & \dots & -h_t a_{2s} B \\ \vdots & \vdots & \ddots & \vdots \\ -h_t a_{s1} B & -h_t a_{s2} B & \dots & \bar{M} - h_t a_{ss} B \end{bmatrix} \begin{bmatrix} \boldsymbol{k_1} \\ \boldsymbol{k_2} \\ \vdots \\ \boldsymbol{k_s} \end{bmatrix} = \begin{bmatrix} B\boldsymbol{w_n} + \boldsymbol{b}_1^{(ST)} \\ B\boldsymbol{w_n} + \boldsymbol{b}_2^{(ST)} \\ \vdots \\ B\boldsymbol{w_n} + \boldsymbol{b}_s^{(ST)} \end{bmatrix}, \tag{2.5}$$

and denote the vector of unknowns in (2.5) as $\boldsymbol{k} \in \mathbb{R}^{2Ns}$ and the right hand side vector as $\boldsymbol{f} \in \mathbb{R}^{2Ns}$. Using the Kronecker product notation, we further rewrite (2.5) as

$$\mathcal{A}\boldsymbol{k} = \boldsymbol{f}, \quad \text{with} \quad \mathcal{A} := I_s \otimes \bar{M} - h_t A \otimes B. \tag{2.6}$$

Note that we never need to (and essentially never should) form this large $2Ns \times 2Ns$ system explicitly. Instead, all operations can be written in terms of the stiffness and mass matrices. For instance, the system matrix in (2.6) can be applied to a vector $\boldsymbol{u}$ efficiently by taking linear combinations of the matrix products

$$\begin{aligned} &M\left[(\boldsymbol{u_1})_\mathbf{1} \quad (\boldsymbol{u_1})_\mathbf{2} \quad (\boldsymbol{u_2})_\mathbf{1} \quad (\boldsymbol{u_2})_\mathbf{2} \quad \dots \quad (\boldsymbol{u_s})_\mathbf{1} \quad (\boldsymbol{u_s})_\mathbf{2}\right] \\ &E\left[(\boldsymbol{u_1})_\mathbf{1} \quad (\boldsymbol{u_2})_\mathbf{1} \quad \dots \quad (\boldsymbol{u_s})_\mathbf{1}\right], \end{aligned} \tag{2.7}$$

where $\boldsymbol{u_i} = \left[(\boldsymbol{u_i})_\mathbf{1}^{\boldsymbol{T}} \quad (\boldsymbol{u_i})_\mathbf{2}^{\boldsymbol{T}}\right]^T$, with $(\boldsymbol{u_i})_\mathbf{1}$ and $(\boldsymbol{u_i})_\mathbf{2}$ both vectors of size N.

3. Sylvester Matrix Reformulation.

In the literature, the original structured system in (2.6) is often solved with preconditioned GMRES with a preconditioner of the general form $\mathcal{P} = I_s \otimes \bar{M} - h_t \tilde{A} \otimes B$ [9, 28, 33, 39]. But in our approach, we will take advantage of the Sylvester matrix system to break down the problem into solving s smaller systems of order $2N \times 2N$. We assume the invertibility of the Butcher coefficient matrix A but for the standard methods, such as Gauss–Legendre, Radau IA, Radau IIA, and Lobatto IIIC, the Butcher matrices are known to be regular, see [18]. Moreover, as mentioned in [18, page 368], the Butcher matrices for the Gauss–Legendre IRK methods have s distinct eigenvalues which come in $\left\lfloor \frac{s}{2} \right\rfloor$ complex conjugate pairs and possibly a single real eigenvalue (if s is odd). In particular, the matrices are diagonalizable. The diagonalizability of A has been observed numerically also for the other IRK methods we listed.

For the Sylvester matrix reformulation of (2.6), we factor out $A \otimes I_N$[*] and multiply both sides by $I_s \otimes \bar{M}^{-1}$, getting

$$\begin{aligned} (I_s \otimes \bar{M}^{-1})\left(A^{-1} \otimes \bar{M} - h_t I_s \otimes B\right)(A \otimes I_{2N})\,\boldsymbol{k} &= (I_s \otimes \bar{M}^{-1})\boldsymbol{f} \\ \left(A^{-1} \otimes I_{2N} + I_s \otimes \hat{B}\right)(A \otimes I_{2N})\,\boldsymbol{k} &= \hat{\boldsymbol{f}} \\ \left(A^{-1} \otimes I_{2N} + I_s \otimes \hat{B}\right)\hat{\boldsymbol{k}} &= \hat{\boldsymbol{f}} \end{aligned} \tag{3.1}$$

where $\hat{B} = -h_t \bar{M}^{-1} B$, $\hat{\boldsymbol{k}} = (A \otimes I_{2N})\,\boldsymbol{k}$ and $\hat{\boldsymbol{f}} = \left(I_s \otimes \bar{M}^{-1}\right)\boldsymbol{f}$. Now, the system in (3.1) can be reformulated as a Sylvester matrix equation

$$\hat{B}\hat{K} + \hat{K}A^{-T} = \hat{F} \tag{3.2}$$

where, $\mathrm{vec}(\hat{K}) = \hat{\boldsymbol{k}}$, and $\mathrm{vec}(\hat{F}) = \hat{\boldsymbol{f}}$ [42]. Considering the eigenvalue decomposition of the matrix A^{-T}

$$A^{-T} = WDW^{-1}, \quad \text{with } D = \mathrm{diag}(d_1, d_2, \dots, d_s),$$

(3.1) then becomes

$$\begin{aligned} \hat{B}\hat{K} + \hat{K}\left(WDW^{-1}\right) &= \hat{F} \\ \hat{B}\hat{K}W + \hat{K}WD &= \hat{F}W \\ \hat{B}\bar{K} + \bar{K}D &= \bar{F}, \end{aligned}$$

where $\bar{K} = \hat{K}W$ and $\bar{F} = \hat{F}W$. Since D is diagonal, this is simply a set of shifted

[*]This idea was first introduced by Butcher in [6].

linear systems

$$\begin{aligned}
\left(\hat{B} + d_i I\right)(\bar{K})_i &= (\bar{F})_i, \quad i = 1, \dots, s \\
\left(-h_t \bar{M}^{-1} B + d_i I\right)(\bar{K})_i &= (\hat{F}W)_i \\
\left(-h_t B + d_i \bar{M}\right)(\bar{K})_i &= (\bar{M}\hat{F}W)_i \\
\left(-h_t B + d_i \bar{M}\right)(\bar{K})_i &= (\bar{M}\bar{M}^{-1}FW)_i \\
\left(\bar{M} - \frac{h_t}{d_i} B\right)(\bar{K})_i &= \frac{1}{d_i}(FW)_i \,. \qquad (3.3)
\end{aligned}$$

As the eigenpairs of A^{-T} all come in complex conjugate pairs (and an extra real eigenpair if s is odd), we can order them so that

$$d_{2i} = \operatorname{conj}(d_{2i-1}), \qquad i = 1, 2, \dots, \left\lfloor \frac{s}{2} \right\rfloor,$$

and since B and $\bar{M}$ are real, we have

$$\bar{M} - \frac{h_t}{d_{2i}} B = \bar{M} - \frac{h_t}{\operatorname{conj}(d_{2i-1})} B = \operatorname{conj}\left(\bar{M} - \frac{h_t}{d_{2i-1}} B\right), \quad i = 1, 2, \dots, \left\lfloor \frac{s}{2} \right\rfloor,$$

and similarly

$$(FW)_{2i} = \operatorname{conj}((FW)_{2i-1}), \qquad i = 1, 2, \dots, \left\lfloor \frac{s}{2} \right\rfloor.$$

In words, computing the solution of the $(2i-1)$-th system, its complex conjugation gives us the solution of the $2i$-th system, i.e., we only need to solve $\lceil s/2 \rceil$ out of the s systems in (3.3). We shall take these corresponding to d_i with non-negative imaginary parts and refer to these as

$$\mathcal{A}_{Syl}[ii](\bar{K})_i = \frac{1}{d_i}(FW)_i, \quad i = 1, 2, \dots, \left\lceil \frac{s}{2} \right\rceil, \qquad (3.4)$$

where $\mathcal{A}_{Syl}$ denotes a $2N\lceil s/2 \rceil \times 2N\lceil s/2 \rceil$ block diagonal system with the (i,i) diagonal block denoted as $\mathcal{A}_{Syl}[ii] = \bar{M} - \frac{h_t}{d_i}B$. We discuss the solution of these in Section 3.1. Finally, we need to compute $\hat{K} = \bar{K}W^{-1}$, then $\hat{\boldsymbol{k}} = \operatorname{vec}(\hat{K})$, and

$$\boldsymbol{k} = (A \otimes I_{2N})^{-1}\hat{\boldsymbol{k}} = \left(A^{-1} \otimes I_{2N}\right)\operatorname{vec}(\hat{K}) = \operatorname{vec}(\hat{K}A^{-T}) = \operatorname{vec}(\bar{K}DW^{-1}). \qquad (3.5)$$

3.1. Solving Shifted Linear Systems.

In this section we discuss how to solve the $\lceil s/2 \rceil$ shifted linear systems from (3.3). We discuss two alternative approaches. In the first method, denoted as $\mathcal{P}^{I}_{Syl}$, the shifted systems are each solved separately with preconditioned GMRES. In the alternative approach, denoted as $\mathcal{P}^{II}_{Syl}$, the shifted systems are solved simultaneously with a single call to (preconditioned) GMRES.

The $\mathcal{P}^{I}_{Syl}$ approach. To solve each of the shifted linear systems separately, we use preconditioned GMRES, with the preconditioner based on the block LU decomposition of $\bar{M} - \frac{h_t}{d_i}B$,

$$\bar{M} - \frac{h_t}{d_i} B = \begin{bmatrix} M & O \\ \frac{h_t}{d_i}E & M + \left(\frac{h_t}{d_i}\right)^2 E \end{bmatrix} \begin{bmatrix} I & -\frac{h_t}{d_i} I \\ O & I \end{bmatrix}, \quad i = 1, \dots, \lceil s/2 \rceil. \qquad (3.6)$$

Proceeding with the standard block forward and backward substitution based on (3.6) requires a system solve with M and $M + (h_t/d_i)^2 E$. However, as we aim for a preconditioner, these solves can and should be only approximate. Also, as $d_i \in \mathbb{C}$, the system solve with $M + (h_t/d_i)^2 E$ is complex-valued.

Approximating the solve with M is the easier of the two tasks as it is a real matrix identical for all of the systems in (3.4). We considered replacing the solve with M with one or two iterations of the Gauss–Seidel method or with one V-cycle of AMG. Although, the use of two (one) iterations of Gauss–Seidel results in slightly (somewhat) larger number of GMRES iterations than with one AMG V-cycle, we observed faster runtimes and thus we stick to using two iterations of Gauss–Seidel as the approximate solver[†] for M.

Considering $M + (h_t/d_i)^2 E$, we get $\lfloor s/2 \rfloor$ complex-valued matrices (for s odd, we get an extra real-valued one), hence using one AMG V-cycle would require separate set-up of AMG in complex-valued arithmetic (see [27]), a *non-trivial time investment*. We instead replace the complex eigenvalues $d_1, \dots, d_{\lfloor s/2 \rfloor}$ with a single scalar d and take $d = d_{\text{avg}} := (d_1 + \ldots + d_s)/s$, replacing $M + (h_t/d_i)^2 E$ with $M + (h_t/d_{\text{avg}})^2 E$. This way we deal with the same matrix solve for all $\lceil s/2 \rceil$ forward substitutions (when applying the preconditioner for (3.4)). Moreover, since the eigenvalues appear in complex conjugate pairs, d_{avg} is always real and hence the same is true for $M + (h_t/d_{\text{avg}})^2 E$. Apart from being always real, d_{avg} is also the minimizer of $\|D - dI\|_F$ over all d, i.e., the best scalar approximation of D. We also did extensive numerical testing, trying to optimize for the choice of $d \in \mathbb{R}$ to improve the quality of the preconditioner and observed that the choice $d = d_{\text{avg}}$ leads to (close to) the best performing one[‡], regardless of s and N. Altogether, we now call the preconditioner setup phase (for AMG or even SuperLU) only once the matrix $M + (h_t/d_{\text{avg}})^2 E$ – instead of $\lceil s/2 \rceil$ times, reducing the preconditioner setup time.

The $\mathcal{P}_{Syl}^{II}$ approach. Alternatively to solving the systems in (3.4) independently, we can stack the systems in (3.4) into a single larger block diagonal system

$$(3.7)\qquad \begin{bmatrix} \bar{M} - \dfrac{h_t}{d_1} B & & \\ & \ddots & \\ & & \bar{M} - \dfrac{h_t}{d_{\lceil s/2 \rceil}} B \end{bmatrix} \begin{bmatrix} (\bar{K})_1 \\ \vdots \\ (\bar{K})_{\lceil s/2 \rceil} \end{bmatrix} = \begin{bmatrix} \dfrac{1}{d_1}(FW)_1 \\ \vdots \\ \dfrac{1}{d_{\lceil s/2 \rceil}}(FW)_{\lceil s/2 \rceil} \end{bmatrix},$$

and solve (3.7) with preconditioned GMRES, where the preconditioner corresponds to stacking the preconditioners for the $\mathcal{P}_{Syl}^{I}$ approach. Naturally, neither of these large matrices should be assembled and instead we formulate all their applications in terms of the matrix-vector products/solves, analogously to (2.7). The main benefit of solving (3.7) over using $\lceil s/2 \rceil$ separate GMRES instances for (3.4) is that now the matrix-vector products with the $\lceil s/2 \rceil$ diagonal blocks can be batched together, in the sense of (2.7) – for both the system matrix as well as the preconditioner application. For instance, for the matrix-vector multiplication with the system matrix in (3.7)

[†]We note that *mass lumping* can also be a useful approach, particularly in the context of higher-order FEM discretizations, see, e.g., [44] and the references therein.

[‡]We aimed to minimize the number of GMRES iterations for the slowest of the systems (3.3) using the preconditioner (3.6) with d_i replaced with some $d \in \mathbb{R}$. Convergence of some of the systems can be further accelerated by a different choice of $d \neq d_{\text{avg}}$ but not the slowest, numerically confirming $d = d_{\text{avg}}$ as a sensible choice.

with a vector $\boldsymbol{u}$ of size $2N\lceil s/2\rceil$, we only need to apply the matrices M and E once to $2\lceil s/2\rceil$ vectors and then consider appropriate linear combinations (including the complex shifts corresponding to d_i). This is done by reshaping the vector $\boldsymbol{u}$ into a matrix $U \in \mathbb{R}^{N\times 2\lceil s/2\rceil}$ such that $\operatorname{vec}(U) = \boldsymbol{u}$, and only then applying either M or E.

For sparse matrices, product with dense matrices has a much higher arithmetic intensity than a sequence of products with vectors and thus runs more efficiently [45]. The same is true when solving linear systems with sparse matrices and single/multiple dense right-hand sides. This clearly presents the upside of the $\mathcal{P}^{II}_{Syl}$ approach.

However, there are also drawbacks. The $\mathcal{P}^{II}_{Syl}$ approach introduces a synchronization point in the solution process of (3.4), which limits the parallelization of the solution process. To asses whether it is more efficient to prioritize the arithmetic density or the completely parallel set-up is highly machine-dependent and we don't comment on this any further.

Another drawback of the $\mathcal{P}^{II}_{Syl}$ approach is that the number of preconditioned GMRES iterations for (3.7) is usually somewhat higher than the average number of preconditioned GMRES iterations for the independent shifted systems in (3.4). Notice that this is not easily explainable by the standard GMRES bounding techniques using the so-called ideal GMRES bound (see [24, Sec. 5.7.3, (5.7.13)]), which focuses on the matrix polynomial norm estimation, see [12, 16, 24]. Indeed, the eigenpairs of the preconditioned system for the $\mathcal{P}^{II}_{Syl}$ approach are easily calculated based on those of the preconditioned systems $\mathcal{P}^{I}_{Syl}$ approach, e.g., eigenvalues in $\mathcal{P}^{II}_{Syl}$ correspond to the union of the eigenvalues for $\mathcal{P}^{I}_{Syl}$. Similarly, FoV $\mathcal{P}^{II}_{Syl}$ is the convex hull of the FoVs for $\mathcal{P}^{I}_{Syl}$ (see [20, Property 1.2.10, p.12]) and similar calculation also holds for the pseudospectra (see [40, Theorem 2.4]). In fact, the difference in the number of GMRES iterations is due to the interaction of the right-hand sides with the preconditioned system matrices. Although this interaction can be crucial for GMRES behavior, see [24, Sec. 5.7.5], in our case the right-hand side vectors are coming from (2.5) and as such we expect these to vary smoothly across the Runge–Kutta stages. In other words, we do not expect large changes between the average number of preconditioned GMRES iterations for (3.4) and the number of preconditioned GMRES iterations for (3.7).

Since the systems in (3.6) and (3.7) are complex, we use a complex GMRES solver. The block triangular solves in (3.6) are also performed in complex arithmetic. Only the calls to the AMG or SuperLU solvers are done in real arithmetic, which is possible because d_{avg} is real. Solving a linear system with a real matrix but complex right-hand side can be done with two real-valued solves, one for the real and one for the imaginary part. We could also use a real-valued GMRES by explicitly solving for the real and complex parts of $\bar{K}$. A complex system $Ax = b$ or $(\Re(A) + i\Im(A))(\Re(x) + i\Im(x)) = \Re(b) + i\Im(b)$ can be solved using real arithmetic by solving the system

$$\begin{bmatrix} \Re(A) & -\Im(A) \\ \Im(A) & \Re(A) \end{bmatrix} \begin{bmatrix} \Re(x) \\ \Im(x) \end{bmatrix} = \begin{bmatrix} \Re(b) \\ \Im(b) \end{bmatrix}. \tag{3.8}$$

However, since the system is now twice as large, we expected this system to have slower solution process (in terms of the overall runtime) and haven't investigated this direction any further.

4. Numerical Results.

In our experiments, we use first-order Lagrange basis functions ($p = 1$) on a 2D triangular mesh for $\Omega = [0, 1]^2$ with the standard Galerkin finite elements spatial discretization and the initial condition $f(\boldsymbol{x}) = \cos(\pi x_1)\cos(\pi x_2)$.

The temporal step size h_t was chosen to balance spatial and temporal errors, depending on the spatial step size h. Specifically, for an s-stage IRK method of order q, we set $h_t = h^{\frac{p+1}{q}}$, where p is the degree of the basis polynomials. For example, for the wave equation with s-stage Gauss–Legendre methods of order $q = 2s$, we chose $h_t = h^{\frac{1}{s}}$.

We use the method of manufactured solutions to facilitate the calculation of relative errors. The Sundance package [26] from the Trilinos project [41] is used to generate the mass and stiffness matrices required for our simulations. Our code is implemented in python, leveraging open-source packages such as NumPy [43], SciPy [43], and PyAMG [4]. Specifically, we use the Ruge-Stüben algebraic multigrid solver from the PyAMG library to perform our experiments efficiently. Additionally, we used MATLAB for creating a few of the plots.

We compare our method with some of the existing methods [8, 9, 28, 33, 39], where the original system (2.6) is preconditioned with a block preconditioner $\mathcal{P}$ having the general form

$$\mathcal{P} = I_s \otimes \bar{M} - h_t \tilde{A} \otimes B, \tag{4.1}$$

where $\tilde{A}$ forms a good preconditioner of the Butcher coefficient matrix A – we consider three specific choices:

- **LD preconditioner ($\boldsymbol{\mathcal{P}_{LD}}$)** [33]: $\tilde{A} = LD$, where $A = LDU$ with L unit lower triangular, D diagonal and U unit upper triangular matrices.
- **$\boldsymbol{\kappa}$-optimal preconditioner ($\boldsymbol{\mathcal{P}_{\kappa}}$)** [39]: $\tilde{A} = L$, with L chosen to minimize $\kappa(L^{-1}A)$ over the space of lower triangular matrices. We carry out the optimization using the Nelder-Mead optimization algorithm [31].
- **Triangular approximate inverse preconditioner ($\boldsymbol{\mathcal{P}_{TAI}}$)**: $\tilde{A} = L$, with L chosen to minimize $\left\|L^{-1}A - I\right\|_2$ over the space of lower triangular matrices.

The κ-optimal and LD preconditioners were initially studied in the context of parabolic equations [28, 39, 33], but have also been shown to be effective when applied to the wave equation with Runge–Kutta–Nystrom timestepping [9, 8].

We propose the $\boldsymbol{\mathcal{P}_{TAI}}$ preconditioner as a new alternative to $\boldsymbol{\mathcal{P}_{\kappa}}$ with a similar idea but a simpler implementation – minimization of $\left\|L^{-1}A - I\right\|_2$ amounts to solving s small least-squares problems as opposed to the minimization of $\kappa(L^{-1}A)$, which requires an optimization algorithm such as Nelder-Mead.

4.1. Preconditioner Application.

Since the preconditioners $\boldsymbol{\mathcal{P}_{LD}}, \boldsymbol{\mathcal{P}_{\kappa}}$ and $\boldsymbol{\mathcal{P}_{TAI}}$ take $\tilde{A}$ as a lower triangular matrix, we put $\tilde{A} = [l_{ij}]$, with $l_{ij} = 0$ for $i < j$. The application of these preconditioners amounts to solving the system

$$\begin{bmatrix} \bar{M} - h_t l_{11} B & & & \\ -h_t l_{21} B & \bar{M} - h_t l_{22} B & & \\ \vdots & & \ddots & \\ -h_t l_{s1} B & \dots & & \bar{M} - h_t l_{ss} B \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \vdots \\ \boldsymbol{v}_s \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_s \end{bmatrix}, \tag{4.2}$$

which we do with a single block forward substitution, requiring s subsolves, namely

$$(\bar{M} - h_t l_{ii} B)\boldsymbol{v}_i = \boldsymbol{b}_i + \sum_{j=1}^{i-1} h_t l_{ij} B \boldsymbol{v}_j \,. \tag{4.3}$$

We again take advantage of the 2×2 block structure of $\bar{M} - h_t l_{ii} B$, i.e., having the block LU factorization

$$\bar{M} - h_t l_{ii} B = \begin{bmatrix} M & 0 \\ h_t l_{ii} E & M + h_t^2 l_{ii}^2 E \end{bmatrix} \begin{bmatrix} I & -h_t l_{ii} I \\ 0 & I \end{bmatrix}.$$

We solve

$$\begin{bmatrix} M & 0 \\ h_t l_{ii} E & M + h_t^2 l_{ii}^2 E \end{bmatrix} \begin{bmatrix} I & -h_t l_{ii} I \\ 0 & I \end{bmatrix} \boldsymbol{v}_i = \boldsymbol{b}_i$$

using block forward and backward substitution. This way, each solve requires one solve with M and one solve with $M + h_t^2 l_{ii}^2 E$ and analogously to the approaches $\mathcal{P}_{Syl}^{I}, \mathcal{P}_{Syl}^{II}$ we replace the solves with M with two iterations of Gauss-Seidel and the solves with $M + h_t^2 l_{ii}^2 E$ with a single V-cycle of AMG.

Altogether, the preconditioners $\mathcal{P}_{LD}, \mathcal{P}_{\kappa}$, and $\mathcal{P}_{TAI}$ require s AMG setups initially and then s V-cycles and $2s$ Gauss-Seidel iterations per application (we assume that the Gauss-Seidel set-up time is negligible compared to that of the AMG). For both $\mathcal{P}_{Syl}^{I}, \mathcal{P}_{Syl}^{II}$, we can do with only 1 AMG setup and then $\lceil s/2 \rceil$ V-cycles and $2s$ Gauss-Seidel iterations per application. This comparison invites the idea of replacing the AMG V-cycles for the s different matrices $M + h_t^2 l_{ii}^2 E$ with an averaged AMG V-cycle for $M + h_t^2 l_{avg}^2 E$ to obtain both, the speed-up in the set-up phase as well as improved arithmetic density, analogously to $\mathcal{P}_{Syl}^{I,II}$. Since we consider the above preconditioners mainly for the sake of comparison, we do not explore this direction any further here but this seems to us as an interesting option for improving the efficiency of these preconditioners.

4.2. Analysis of the preconditioners. To predict a GMRES preconditioner quality we commonly try to calculate or bound some key properties of the preconditioned system, e.g., its spectrum (and its clustering) and conditioning of its eigenbasis, its field of values (FoV) or its pseudospectrum. A favorable results, such as these being well-separated from the origin and/or tightly clustered can provide valuable insights into the convergence behavior of GMRES, see [24, Section 5.7]. Here we illustrate some of these properties for the preconditioned system with all of the considered preconditioners, with the mesh size[§] $h = 2^{-3}$.

First, in Figure 4.1, we show the "benchmark" eigenvalue and FoV plots for both the unpreconditioned and the left preconditioned systems using $\mathcal{P}_{LD}, \mathcal{P}_{\kappa}$, and $\mathcal{P}_{TAI}$ and constructing the preconditioners exactly. Notably, the eigenvalues of each of the preconditioned systems are much better separated from the origin comapred to the unpreconditioned case, with $\mathcal{P}_{TAI}$ achieving the most favorable properties of the ones plotted. However, increasing the number of stages results in the a shift of the eigenvalues closer to the origin and fast expansion of FoV.

Next, we examine the analogous quantities also for $\mathcal{P}_{Syl}^{I}$ and $\mathcal{P}_{Syl}^{II}$. As in section 3.1, we only consider the systems associated with the eigenvalues d_i of A^{-T} with non-negative imaginary part, see (3.4).

As for the preconditioners, we recall the block LU factorization of $\mathcal{A}_{Syl}[ii]$ in (3.4) and replace d_i with d_{avg} in the diagonal blocks of the block lower triangular factor.

[§]Note that later we show performance results for $h = 2^{-9}$. Both computation and visualization of spectra and FoV is demanding for fine mesh size (and larger s). The effect of the mesh scaling has been considered in [15, 11] for parabolic problems and can be addressed in similarly also here.

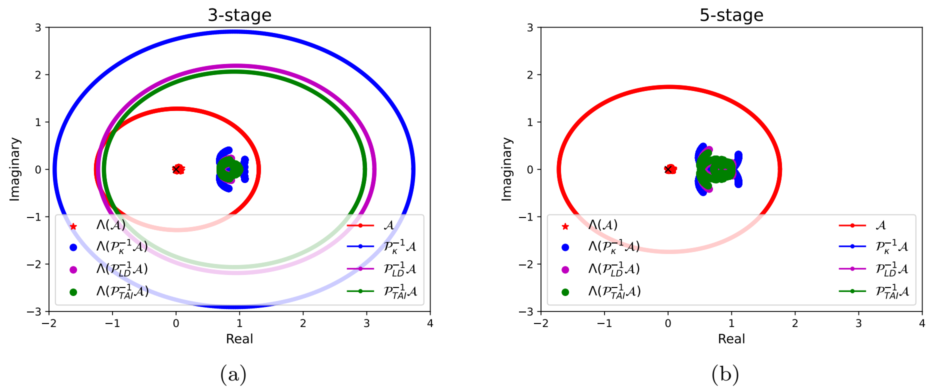


Fig. 4.1: Eigenvalue and FoV plots of the left-preconditioned 2D wave problem with $h = 2^{-3}$ using the Gauss–Legendre IRK method for $s = 3$ (a) and $s = 5$ (b), preconditioned with $\mathcal{P}_{\kappa}, \mathcal{P}_{LD}$ and $\mathcal{P}_{TAI}$. The spectrum of the unpreconditioned systems in red are shown together with the spectrum of the preconditioned systems as a reference. Preconditioners are constructed exactly. We note that the FoV for the preconditioned systems for $s = 5$ all contain the rectangle $(-2, 4) \times (-3i, 3i)$ and as such are not visible in the plot.

Multiplying back, the exact preconditioner for the i-th system becomes

$$(4.4) \qquad \begin{bmatrix} M & -\dfrac{h_t}{d_i}M \\ \dfrac{h_t}{d_i}E & h_t^2\left(\dfrac{d_i^2 - d_{\text{avg}}^2}{d_i^2 d_{\text{avg}}^2}\right)E + M \end{bmatrix}, \quad \text{for } i = 1, \dots, \left\lceil \frac{s}{2} \right\rceil .$$

In Figure 4.2 we show the spectra and FoV with the preconditioners constructed exactly (i.e., corresponding to (4.4)) and in Figure 4.3 we consider the "realistic" preconditioners, where the block-solves are replaced with two Gauss-Seidel iterations or one AMG V-cycle, see section 3.1.

In both cases, the eigenvalues of the preconditioned systems $\mathcal{P}_{Syl}^{-1}\mathcal{A}_{Syl}[ii]$ are well separated away from zero for $i = 1, \dots, \lceil s/2 \rceil$. While the plots in Figures 4.2 and 4.3 resemble each other closely, differences can be observed, e.g., by comparing the subplots for $s = 2$ and $s = 4$. We also see that the FoV plots tell a similar story with respect to increasing s – larger s results in larger FoVs that are, moreover, closer to the origin and thereby worsening the classical GMRES bound based on FoVs. Nonetheless, we see a quantitative improvement in the FoV compared to the other preconditioners.

Finally, we present the condition numbers of the eigenbasis of the preconditioned systems in Table 4.1. First, we see that for both $\mathcal{P}_{Syl}^{I}$ and $\mathcal{P}_{Syl}^{II}$ we obtain truly well-conditioned eigenbasis, suggesting that the eigenvalues indeed govern the GMRES convergence, i.e., the study and understanding of Figure 4.2 becomes decisive. Seemingly, the same cannot be said for $\mathcal{P}_{LD}, \mathcal{P}_{\kappa}$ and $\mathcal{P}_{TAI}$, looking at the numbers before parenthesis in Table 4.1. However, this can be ascribed to large extend to the singularity of E (due to the Neumann BC in (2.2)). As this results in only one

s	$\mathcal{P}_{LD}$	$\mathcal{P}_{\kappa}$	$\mathcal{P}_{TAI}$	$\mathcal{P}^{I}_{Syl}$	$\mathcal{P}^{II}_{Syl}$
2	7.7e+7 (1.4e+2)	5.8e+7 (1.1e+2)	7.2e+7 (1.3e+2)	[3.6]	3.6
3	8.2e+7 (2.1e+2)	8.1e+7 (2.1e+2)	6.5e+7 (1.7e+2)	[3.6, 3.9]	3.9
4	7.9e+7 (2.4e+2)	1.2e+8 (3.8e+2)	8.0e+7 (2.4e+2)	[3.6, 3.7]	3.7
5	1.0e+8 (3.6e+2)	1.4e+8 (4.7e+2)	1.6e+8 (5.4e+2)	[3.6, 3.6, 3.5]	3.6

Table 4.1: We show the condition numbers of the eigenbasis of the preconditioned systems (for the exact preconditioners) for different s with the mesh size $h = 2^{-3}$. For $\mathcal{P}_{LD}, \mathcal{P}_{\kappa}$ and $\mathcal{P}_{TAI}$ we observed that the singularity of E is the main culprit and we also show (in parantheses) the conditioning when this is artificially removed. For $\mathcal{P}^{I}_{Syl}$ we present the conditioning of the eigenbasis for each of the $\lceil s/2 \rceil$ systems. We want to emphasize that to obtain these results accurately, requires a careful reformulation of the calculation, refer to the text for more details.

eigenvector that corresponds to the zero eigenvalue, we can look on the conditioning of the eigenbasis without this mode (corresponding to a constant function) and we see a significant improvement (shown in the parentheses in Table 4.1). Let us note that the results in Table 4.1 are not identical to numbers we obtain when we compute the condition numbers of the eigenbasis naively, using routines `solve(), eig(), cond()` from the library `numpy.linalg` (or their other equivalents) – quite on the contrary. Due to the high density of the eigenvalues shown in Figures 4.1–4.3, the standard approach is susceptible to numerical instabilities and gives (sometimes hugely) inaccurate results. However, these issues can be addressed by a direct reformulation to obtain correct values.[¶]

4.3. Performance Comparison: Wave Equation.

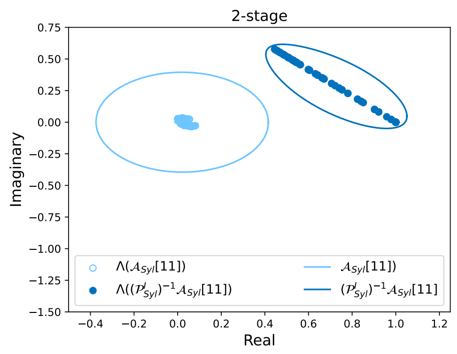
In this section, we compare the performance of the different preconditioning methodologies analyzed in the previous section. Our comparison focuses on AMG setup time, total GMRES solve time, GMRES iteration counts. We also verify the code and its calculations using the method of manufactured solutions, see [35] and the references therein.

Number of solver calls: For $\mathcal{P}^{I}_{Syl}$, we require one real AMG set-up (as discussed in Section 3.1) and each preconditioner application requires two V-cycles (since the right-hand side is complex-valued we compute separately the real and the imaginary parts) and two iterations of Gauss-Seidel (replacing the solve with M with two Gauss-Seidel iterations). Denoting the number of GMRES iteration for each of the $\lceil s/2 \rceil$ systems by $itr_1, \ldots, itr_{\lceil s/2 \rceil}$, the total number of AMG V-cycles and Gauss-Seidel calls becomes

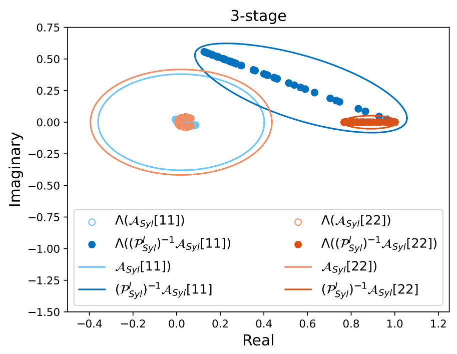
$$\#\ \text{AMG V-cycles} : \sum_{j=1}^{\lceil s/2 \rceil} 2 \times itr_j, \qquad \#\ \text{Gauss-Seidel iterations} : \sum_{j=1}^{\lceil s/2 \rceil} 2 \times itr_j.$$

[¶]These statements are not obvious but can be derived using the framework in [15] and will be explained and proved in detail in the upcoming manuscript focusing on the spectral analysis of these preconditioners. However we feel that these should be mentioned here to show the full picture of the preconditioned systems for GMRES, and eigenvalues alone cannot do that.

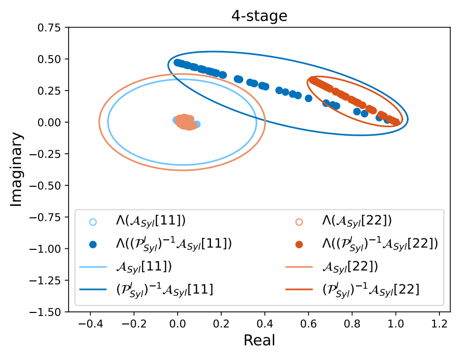
In [11] and [15], the authors propose new techniques for spectral analysis of similar systems arising from parabolic PDEs – these can be also adapted to the hyperbolic case as well as extended for the FoVs and pseudospectral GMRES bounds. This is already a work in progress and will be treated separately in our upcoming paper, together with theoretical justification of the above observations. Next, we discuss the resulting GMRES performance.



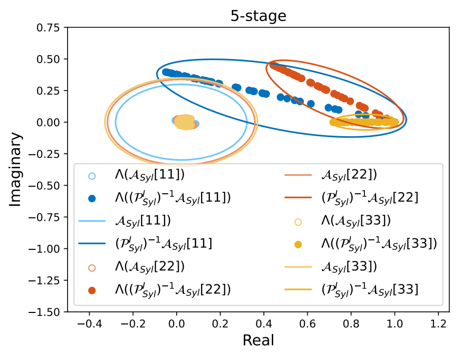
(a)



(b)



(c)



(d)

Fig. 4.2: Eigenvalue and FoV plots of $\mathcal{A}_{Syl}[ii]$ and $(\mathcal{P}^I_{Syl})^{-1}\mathcal{A}_{Syl}[ii]$ of a 2D wave equation with $h = 2^{-3}$ using Gauss–Legendre for $s = 2$ to $s = 5$. Preconditioners are constructed exactly.

For the preconditioners $\mathcal{P}_{LD}$, $\mathcal{P}_{\kappa}$, and $\mathcal{P}_{TAI}$, we require s real AMG set-ups (as discussed in Section 3.1) and each preconditioner application requires s AMG V-cycles and $2s$ Gauss-Seidel iterations. Denoting the number of GMRES iterations by itr_{LD}, itr_{κ} and itr_{TAI}, the total number of AMG V-cycles and Gauss-Seidel iterations for each of these preconditioners becomes

$$\#\text{ AMG V-cycles}: \; s \times itr_{LD,\kappa,TAI}, \quad \#\text{ Gauss-Seidel iterations}: \; 2s \times itr_{LD,\kappa,TAI}.$$

Figure 4.4b compares the total number of AMG V-cycles required by various preconditioners for the mesh size $h = 2^{-9}$. For $s \geq 4$, $\mathcal{P}^I_{Syl}$ requires fewer V-cycles compared to the other preconditioners.

Runtime: In order to compare the runtime of the preconditioners, we clock the set-up times and the times until GMRES converges to the relative residual smaller than $1e-8$[‖] with randomized right-hand side vectors (of appropriate sizes) for the

[‖]As we focus here on comparing the preconditioners, we set identical GMRES tolerance for all

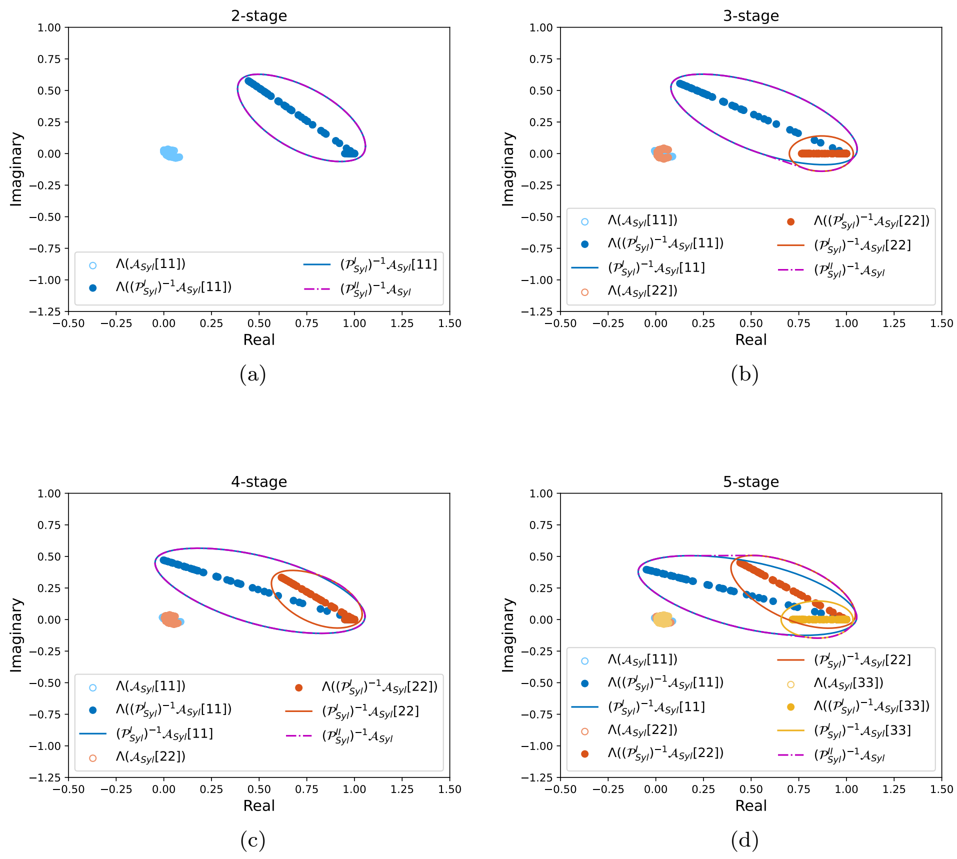


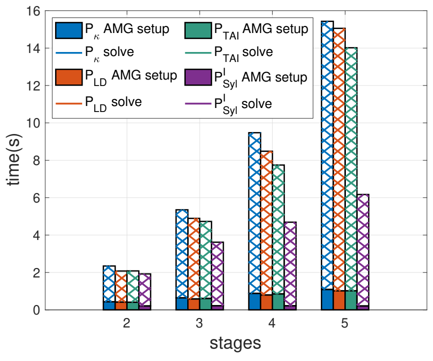
Fig. 4.3: Eigenvalue and FoV plots of $\mathcal{A}_{Syl}[ii]$ and $(\mathcal{P}^I_{Syl})^{-1}\mathcal{A}_{Syl}[ii]$ of a 2D wave equation with $h = 2^{-3}$ using Gauss–Legendre for $s = 2$ to $s = 5$. The preconditioners use instead of exact solves 2 iterations of Gauss-Seidel and 1 AMG V-cycle.

preconditioned systems $\mathcal{P}_{LD}^{-1}\mathcal{A}$, $\mathcal{P}_{\kappa}^{-1}\mathcal{A}, \mathcal{P}_{TAI}^{-1}\mathcal{A}$ and for the approach $\mathcal{P}^I_{Syl}$ (adding together all of the $\lceil s/2 \rceil$ system timings).

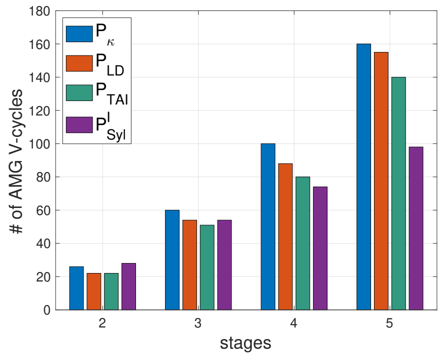
In Figure 4.4a, we compare both the solve time (cross pattern) and AMG setup time (solid color) for different preconditioners for various number of stages for the mesh size $h = 2^{-9}$ (finest considered). We begin to notice differences in performance for $s = 3$, while for $s = 5$ $\mathcal{P}^I_{Syl}$ showcases a significant advantage, being twice as fast as the next best method, $\mathcal{P}_{TAI}$.

In Figure 4.5, we again present both the solve time (cross pattern) and AMG setup time (solid color) for different preconditioners, now with fixed number of stages ($s = 5$) and for refining mesh. The $\mathcal{P}^I_{Syl}$ approach consistently outperforms the other preconditioners throughout the refinement process, again requiring only half of the total runtime of the next best method, $\mathcal{P}_{TAI}$.

mesh sizes but note that in practice we usually aim for balancing the GMRES tolerance against the (expected) discretization error, similarly to balancing the temporal and spatial discretization errors.

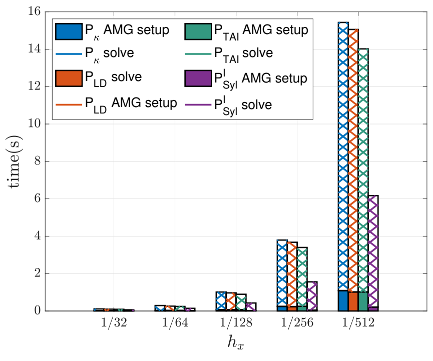


(a) runtime

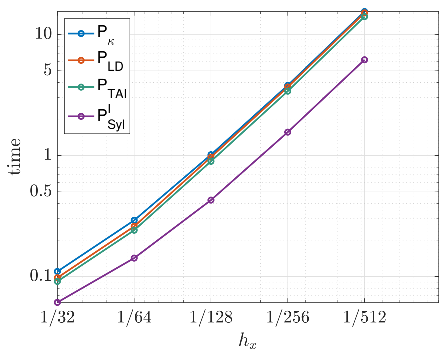


(b) number of AMG V-cycles

Fig. 4.4: Comparison of the Sylvester formulation $\mathcal{P}^I_{Syl}$ (separate $\lceil s/2 \rceil$ GMRES systems) with other preconditioners for $h = 2^{-9}$, corresponding to one preconditioned GMRES call (or to the sum of the $\lceil s/2 \rceil$ calls for $\mathcal{P}^I_{Syl}$), including the set-up time.



(a)

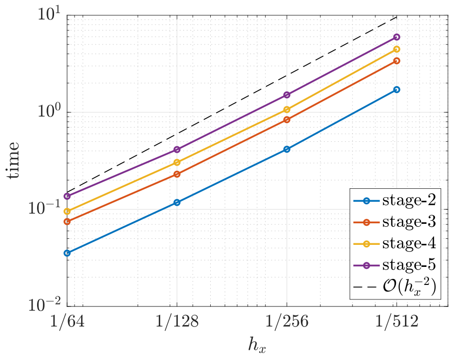


(b)

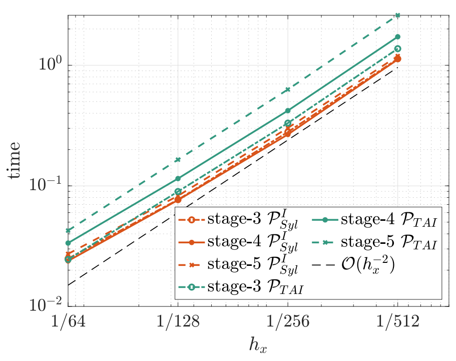
Fig. 4.5: Runtime comparisons (including the set-up time) in linear (a) and logarithmic (right) scale of the Sylvester formulation $\mathcal{P}^I_{Syl}$ (separate $\lceil s/2 \rceil$ GMRES systems) with other preconditioners as mesh refines for $s = 5$. The timings correspond to one preconditioned GMRES call (or to the sum of the $\lceil s/2 \rceil$ calls for $\mathcal{P}^I_{Syl}$).

The superior performance of $\mathcal{P}^I_{Syl}$ can be attributed to its efficient handling of the Sylvester reformulation, which reduces the number of AMG V-cycles needed for each subsolve. This efficiency becomes more pronounced as the stage number increases and the mesh is refined, highlighting the robustness and scalability of $\mathcal{P}^I_{Syl}$. On the other hand, preconditioners such as $\mathcal{P}_{LD}$ and $\mathcal{P}_\kappa$ require additional AMG V-cycles and hence more time, especially for finer meshes and/or larger s. Comprehensive tabulated data, including solve time, AMG setup time, and iteration count, can be found in [34].

Scalability: In Figure 4.6, we present the scalability of $\mathcal{P}^I_{Syl}$ with respect to



(a)



(b)

Fig. 4.6: Linear scaling of the solution time of $\mathcal{P}^I_{Syl}$ with respect to N (the number of mesh points) and s is shown in (a) and (b) respectively. In (b), to see the linear scaling in s, we divide the runtime by s, and compare with $\mathcal{P}_{TAI}$. The timings correspond to one preconditioned GMRES call (or to the sum of the $\lceil s/2 \rceil$ calls for $\mathcal{P}^I_{Syl}$) without the AMG set-up time.

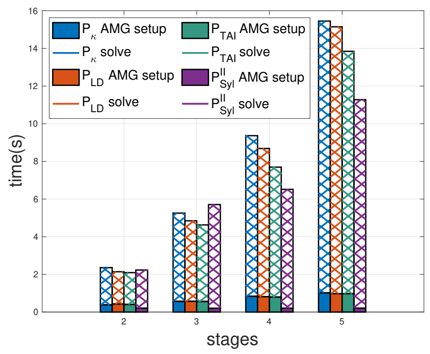
N (the number of mesh points, $N = 2(1 + h_x^{-1})^2$) and s (the number of stages) corresponding to one preconditioned GMRES call (or to the sum of the $\lceil s/2, \rceil$ calls for $\mathcal{P}^I_{Syl}$). Figure 4.6a demonstrates the linear scalability of $\mathcal{P}^I_{Syl}$ in terms of N, indicating that Sylvester preconditioning achieves optimal scaling with mesh size, exhibiting a runtime of $O(N)$.

Our method is highly scalable, showing a time complexity of $O(sN)$ compared to $O(s^2N)$ for other preconditioners, as illustrated in Figure 4.6b. To demonstrate this, we consider the time per stage and compare it with $\mathcal{P}_{TAI}$, shown in green, which has a time complexity of $O(s^2N)$. We expect the $\mathcal{P}^I_{Syl}$ plots (in orange in Figure 4.6b) corresponding to different times per stage to overlap, which is consistent with our observations. In contrast, the $\mathcal{P}_{TAI}$ plots exhibit vertical shifts as the stage number increases, confirming the expected $O(s^2N)$ complexity.

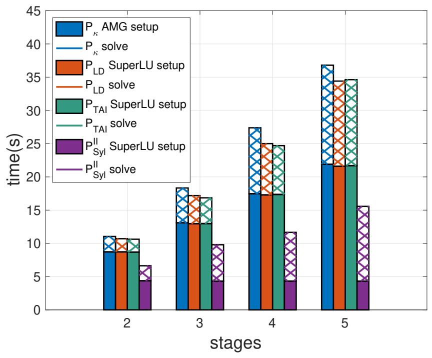
However, it is worth noting that due to the transformation with W, our method technically remains $O(s^2N)$. Despite this, the transformation step is highly efficient and does not dominate the overall timing, ensuring that $\mathcal{P}^I_{Syl}$ remains an optimal and scalable preconditioning strategy.

$\boldsymbol{\mathcal{P}^{II}_{Syl}}$ approach: So far, we have discussed the performance of $\mathcal{P}^I_{Syl}$ and compared it with other preconditioners $\mathcal{P}_\kappa$, $\mathcal{P}_{LD}$ and $\mathcal{P}_{TAI}$. Next we will discuss the performance of $\mathcal{P}^{II}_{Syl}$ approach, based on construction of a diagonal block matrix, referred to as $\mathcal{A}_{Syl}[ii]$, see (3.4). This allows us to batch linear systems solves with $M-(h_t/d_{avg})^2E$ during the preconditioner application into a single solve applied to $\lceil s/2, \rceil$ right-hand sides as well as also harvest the gains of lower set-up times of $\mathcal{P}^I_{Syl}$.

Figure 4.7 shows the runtime performance of $\mathcal{P}^{II}_{Syl}$ compared with other preconditioners ($\mathcal{P}_{LD}$, $\mathcal{P}_\kappa$, and $\mathcal{P}_{TAI}$) using two different solvers for the problems with the system matrices $M-(h_t/d_{avg})^2E$ and $M - h_t^2 l_{ii}^2 E$ – one AMG V-cycle in Figure 4.7a or the `SuperLU` in Figure 4.7b. As above, the cross pattern represents the solve time, while the solid color indicates the AMG/SuperLU setup time.



(a) AMG



(b) SuperLU

Fig. 4.7: Runtime of the Sylvester formulation $\mathcal{P}^{II}_{Syl}$ (GMRES applied to (3.7)) using (a) AMG (without multiple right-hand side application) and (b) SuperLU (with multiple right-hand side application), with other preconditioners with $h = 2^{-9}$ for one preconditioned GMRES call.

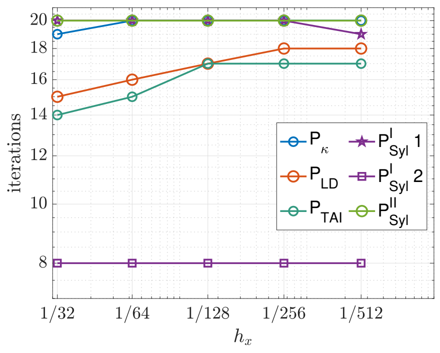
The reason for comparing these two solvers is the following: while generally slower for large systems and more expensive in set-up time SuperLU allows for multiple right-hand side whereas the PyAMG implementation of AMG V-cycle does not (using a different AMG implementation that does allow for multiple right-hand sides, e.g., the `hypre` package, is part of the ongoing work). Hence, the possible gains from batching the solves together can be inferred by comparing these two.

In Figure 4.7a, we see that $\mathcal{P}^{II}_{Syl}$ performs well compared to the other preconditioners when using AMG for the subsolve, for $s \geq 4$. In Figure 4.7b, $\mathcal{P}^{II}_{Syl}$ consistently outperforms the other preconditioners in terms of runtime when using SuperLU for the subsolve.

Although the ability to combine operations efficiently is theoretically crucial for maintaining scalability and reducing computational overhead, and particularly so in large-scale simulations, we have seen only modest gains in the runtime improvement of $\mathcal{P}^{II}_{Syl}$ compared to the other preconditioners when comparing SuperLU and the AMG. Naturally, the set-up time improvement is decisive but that is to be expected, especially for larger systems, such as those for $s = 4, 5$. That being said, we believe that further investigation and code improvement in the direction of the $\mathcal{P}^{II}_{Syl}$ approach could and will lead to meaningful improvements. Especially as we continue to refine our preconditioning strategies, integrating more advanced and apt software solutions will be essential for achieving optimal performance.

GMRES iterations: Figure 4.8 compares the iteration counts of $\mathcal{P}_{LD}$, $\mathcal{P}_{\kappa}$, $\mathcal{P}_{TAI}$, $\mathcal{P}^{I}_{Syl}$, and $\mathcal{P}^{II}_{Syl}$. We see that the total number of iterations for $\mathcal{P}^{II}_{Syl}$ is lower than the sum of the number of iterations for $\mathcal{P}^{I}_{Syl}$ for all $i = 1, \ldots, s$. Admittedly, this is not a fair comparison, since each iteration for $\mathcal{P}^{II}_{Syl}$ is more expensive than those for $\mathcal{P}^{I}_{Syl}$[**]. Nonetheless, Figure 4.8 highlights our experience, where only one

[**]In fact, even the GMRES iterations for the different systems with $\mathcal{P}^{I}_{Syl}$ can have different costs – if s is odd, then one of the systems will be real. This highlights that the timings might give a


(a) stage-3

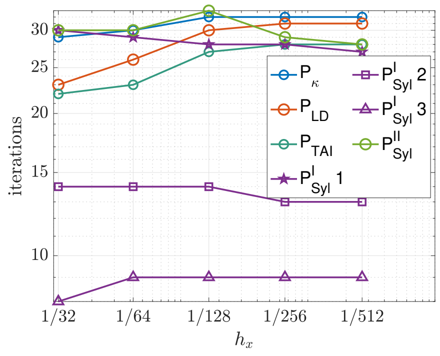
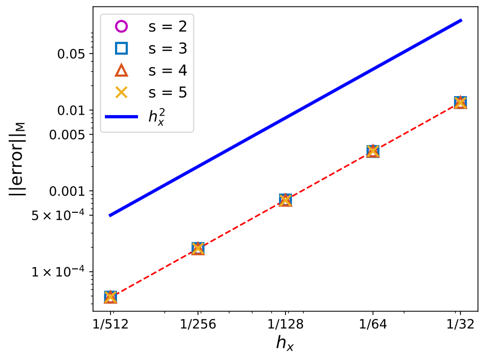

(b) stage-5

Fig. 4.8: The number of preconditioned GMRES iterations for a single GMRES call (or for each of the $\lceil s/2 \rceil$ calls for $\mathcal{P}_{Syl}^{I}$) as a function of the mesh-size h_x.


(a) $\epsilon_1, \epsilon_2, \epsilon_3$ = 0.75,0,0

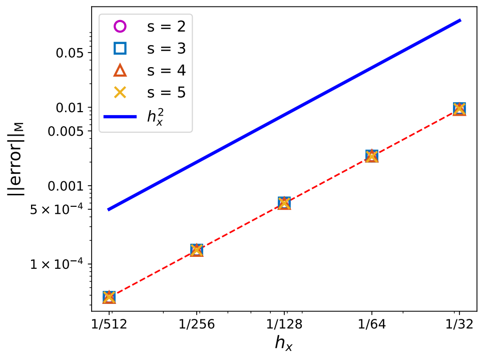

(b) $\epsilon_1, \epsilon_2, \epsilon_3$ = 0.75,0.75,0.375

Fig. 4.9: Mesh dependence of the relative errors for the generalized wave equation with the parameters choice $[\epsilon_1, \epsilon_2, \epsilon_3] = [0.75, 0, 0]$ (a) and the parameters choice $[\epsilon_1, \epsilon_2, \epsilon_3] = [0.75, 0.75, 0.375]$ (b). The errors are $O\left(h_x^2\right)$ at all stages; recall that the timestep h_t was chosen so that $h_x^{p+1} = h_t^{2s}$ accuracy, so we are obtaining the expected convergence rate in both space and time with no order reduction.

of the $\mathcal{P}_{Syl}^{I}$ systems required comparable number of GMRES iterations to the other considered preconditioners while the rest of the systems converged in significantly fewer iterations.

Computation verification: To verify the computed solutions and the code, we look at the discrete L^2 norm of the relative errors in space along the entire time-stepping process and we considered the largest one, i.e., we focused on the discrete L^∞ norm in time of the discrete L^2 norms in space. The GMRES solve(s) were run at each timestep until the relative residual decreased below $1e-8$. The (relative) L^2 norm

better idea of the efficiency than the iteration counts in our set-up.

of the sptaial errors are generally of similar magnitude and we have not observed any order reduction in these as the timestepping progressed. As the wave equation in the simple form plausibly doesn't pose enough of a challenge, we carried out the same experiments also for the generalized wave equation problem described in Section 4.4 and we show the results in Figure 4.9a (for the parameters choice $[\epsilon_1, \epsilon_2, \epsilon_3] = [0.75, 0, 0]$) and Figure 4.9b (for the parameters choice $[\epsilon_1, \epsilon_2, \epsilon_3] = [0.75, 0.75, 0.375]$). Same as for the simple wave equation, we have not observed any order reduction.

In conclusion, while $\mathcal{P}^{I}_{Syl}$ already demonstrates excellent performance, there is potential for improving the performance of $\mathcal{P}^{II}_{Syl}$ with AMG if we adopt software that supports combined operations. This adaptation would, based on our experience, lead to meaningful runtime reductions, see [25].

4.4. Performance Comparison: Generalized settings.

We consider a generalized version of (2.2), namely

$$\begin{aligned}
u_{tt} &= \nabla \cdot (\kappa(\boldsymbol{x})\nabla u) - \beta(\boldsymbol{x})u + b(\boldsymbol{x}, t), && \text{in } \Omega \times (0, T],\\
\hat{\boldsymbol{n}} \cdot \nabla u &= 0, && \text{on } \partial\Omega,\\
u(\boldsymbol{x}, 0) &= \psi(\boldsymbol{x}) && \text{in } \Omega,\\
u_t(\boldsymbol{x}, 0) &= 0 && \text{in } \Omega,
\end{aligned} \tag{4.5}$$

with the particular choices

$$\kappa(\boldsymbol{x}) = 1 + \epsilon_1 \cos\left(\sqrt{2}x_1 + \frac{x_2}{\sqrt{3}}\right), \quad \beta(\boldsymbol{x}) = \epsilon_2 + \epsilon_3 \sin\left(\frac{x_1}{\sqrt{6}} - \sqrt{5}x_2\right),$$
$$\psi(\boldsymbol{x}) = \cos(x_1)\cos(2x_2),$$

where $\boldsymbol{x} = [x_1, x_2]^T \in \Omega = [0, \pi]^2$ and the coefficients are constrained as $|\epsilon_1| < 1$, $|\epsilon_2| \geq 0$, $|\epsilon_3| \leq |\epsilon_2|$ so that both $\kappa(\boldsymbol{x})$ and $\beta(\boldsymbol{x})$ are positive functions. We again use the method of manufactured solutions and take $b(\boldsymbol{x}, t)$ so that the function $u(\boldsymbol{x}, t) := \psi(\boldsymbol{x})\cos(t)$ satisfies the equation (4.5). The transformation and weak formulation carry through analogously to Section 2, arriving at the same problem (2.6), only now the stiffness matrix entries are given as

$$E_{ij} = \int_{\Omega} \kappa \nabla\phi_i \cdot \nabla\phi_j + \beta\phi_i\phi_j \mathrm{d}\boldsymbol{x},$$

instead of (2.4).

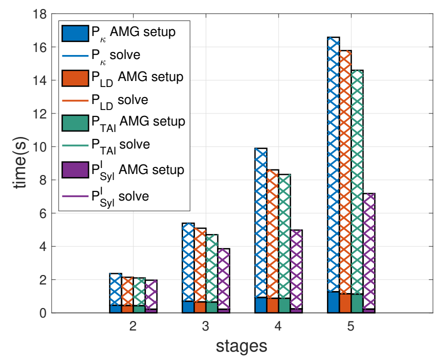
In our experience, the results for this problem generally follow the key features pointed out in the previous section. We consider two particular setting:

The space-variable wave equation We take $[\epsilon_1, \epsilon_2, \epsilon_3] = [0.75, 0, 0]$ so that $\beta(\boldsymbol{x}) \equiv 0$. We show the runtimes and AMG V-cycles count in Figure 4.10 and their scaling in Figures 4.11 and 4.12.

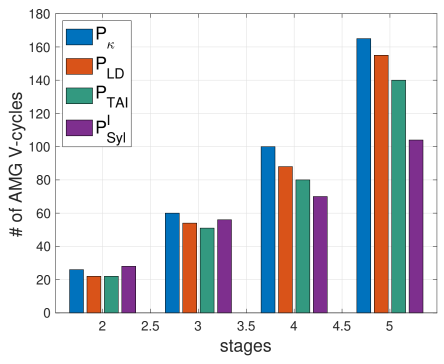
The Klein-Gordon equation We take $[\epsilon_1, \epsilon_2, \epsilon_3] = [0.75, 0.75, 0.375]$. We show the runtimes in Figure 4.13 and their scaling in Figures 4.14 and 4.15.

5. Conclusion.

We have presented a reformulation of the large, structured linear system from the IRK time integration of hyperbolic PDEs as an equivalent Sylvester matrix equation. We then reduced the problem to a series of s separate smaller linear systems, which we can solve efficiently with preconditioned GMRES. The resulting, new method stands proves to be twice as fast as other existing approaches when increasing the number of Runge–Kutta stages and refining the mesh,

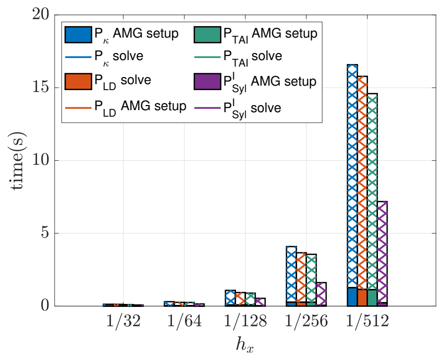


(a) runtime

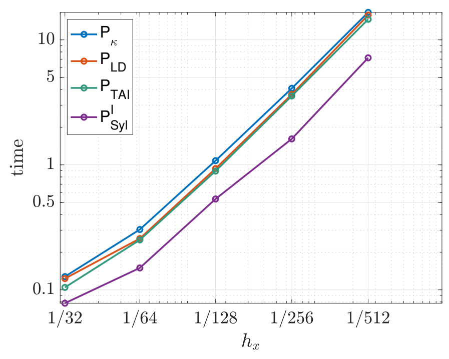


(b) number of AMG V-cycles

Fig. 4.10: Comparison of the Sylvester formulation $\mathcal{P}^I_{Syl}$ (separate $\lceil s/2 \rceil$ GMRES systems) with other preconditioners for $h = 2^{-9}$, corresponding to one preconditioned GMRES call (or to the sum of the $\lceil s/2 \rceil$ calls for $\mathcal{P}^I_{Syl}$), including the set-up time. The parameters are taken as $[\epsilon_1, \epsilon_2, \epsilon_3] = [0.75, 0, 0]$.



(a)

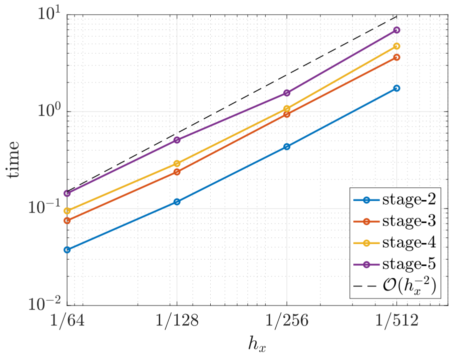


(b)

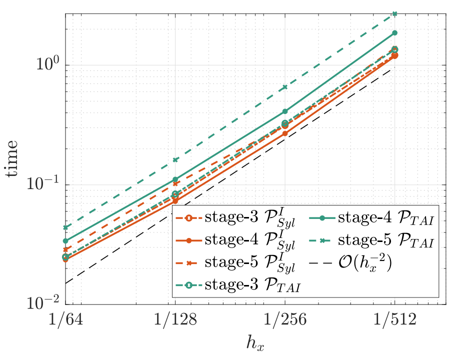
Fig. 4.11: Runtime comparisons (including the set-up time) in linear (a) and logarithmic (right) scale of the Sylvester formulation $\mathcal{P}^I_{Syl}$ (separate $\lceil s/2 \rceil$ GMRES systems) with other preconditioners as mesh refines for $s = 5$. The timings correspond to one preconditioned GMRES call (or to the sum of the $\lceil s/2 \rceil$ calls for $\mathcal{P}^I_{Syl}$). The parameters are taken as $[\epsilon_1, \epsilon_2, \epsilon_3] = [0.75, 0, 0]$.

while requiring fewer AMG V-cycles. Our experiments show that our method outperforms other commonly used preconditioners, with the improvement becoming more pronounced as s is increased and persistently doing better, twice as fast with refined spatial discretization.

The preconditioner $\mathcal{P}^I_{Syl}$ reduces both the solve time but also the AMG setup time, compared to the other preconditioenrs. Notably, $\mathcal{P}^I_{Syl}$ requires only one AMG setup, while the other preconditioners require s set-ups, making also the set-up phase

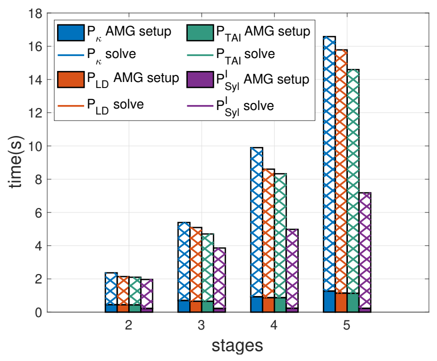


(a)

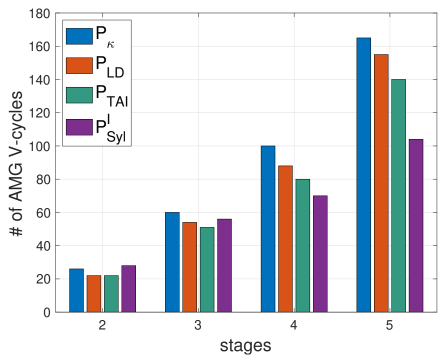


(b)

Fig. 4.12: Linear scaling of the solution time of $\mathcal{P}^I_{Syl}$ with respect to N (the number of mesh points) and s is shown in (a) and (b) respectively. In (b), to see the linear scaling in s, we divide the runtime by s, and compare with $\mathcal{P}_{TAI}$. The timings correspond to one preconditioned GMRES call (or to the sum of the $\lceil s/2 \rceil$ calls for $\mathcal{P}^I_{Syl}$) without the AMG set-up time. The parameters are taken as $[\epsilon_1, \epsilon_2, \epsilon_3] = [0.75, 0, 0]$.



(a) runtime



(b) number of AMG V-cycles

Fig. 4.13: Comparison of the Sylvester formulation $\mathcal{P}^I_{Syl}$ (separate $\lceil s/2 \rceil$ GMRES systems) with other preconditioners for $h = 2^{-9}$, corresponding to one preconditioned GMRES call (or to the sum of the $\lceil s/2 \rceil$ calls for $\mathcal{P}^I_{Syl}$), including the set-up time. The parameters are taken as $[\epsilon_1, \epsilon_2, \epsilon_3] = [0.75, 0.75, 0.375]$.

notably more efficient. Although $\mathcal{P}^{II}_{Syl}$ did not perform to its full potential in its current implementation, we explained its potential based on improved efficiency of combined matrix-vector multiplication operations.

We are considering integrating our proposed method in MFEM (Modular Finite Element Method) [2, 29] and/or SUNDIALS [19], and we plan to extend the framework to other preconditioners and other time-dependent PDEs. In order to take full advantage of the new formulation, we intend to use the hypre algebraic multigrid solver [21]

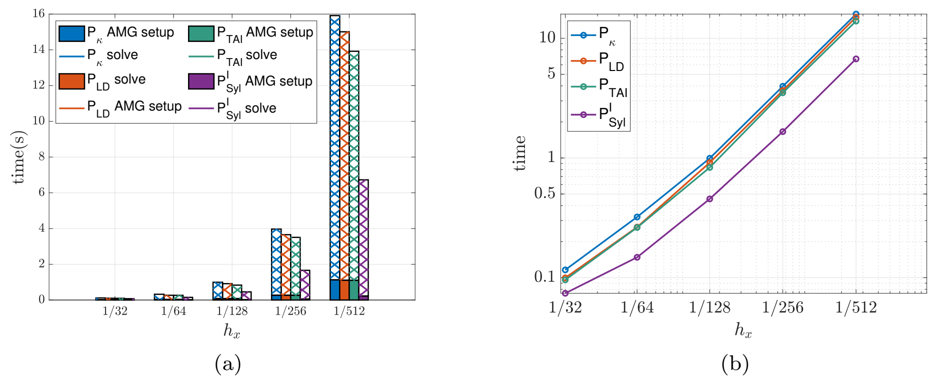


Fig. 4.14: Runtime comparisons (including the set-up time) in linear (a) and logarithmic (right) scale of the Sylvester formulation $\mathcal{P}^I_{Syl}$ (separate $\lceil s/2 \rceil$ GMRES systems) with other preconditioners as mesh refines for $s = 5$. The timings correspond to one preconditioned GMRES call (or to the sum of the $\lceil s/2 \rceil$ calls for $\mathcal{P}^I_{Syl}$). The parameters are taken as $[\epsilon_1, \epsilon_2, \epsilon_3] = [0.75, 0.75, 0.375]$.

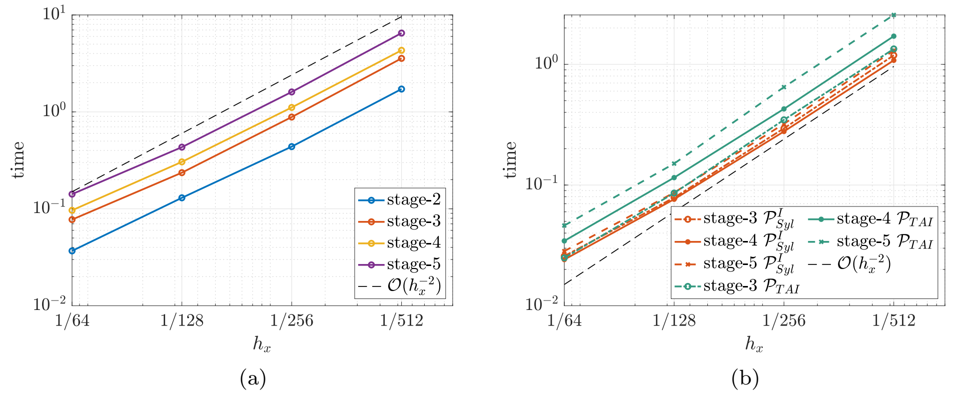


Fig. 4.15: Linear scaling of the solution time of $\mathcal{P}^I_{Syl}$ with respect to N (the number of mesh points) and s is shown in (a) and (b) respectively. In (b), to see the linear scaling in s, we divide the runtime by s, and compare with $\mathcal{P}_{TAI}$. The timings correspond to one preconditioned GMRES call (or to the sum of the $\lceil s/2 \rceil$ calls for $\mathcal{P}^I_{Syl}$) without the set-up time. The parameters are taken as $[\epsilon_1, \epsilon_2, \epsilon_3] = [0.75, 0.75, 0.375]$.

instead of pyAMG as it supports multiple right-hand side vectors. Alternatively, we will look at using the STRUMPACK [7] sparse solver and preconditioning library. The STRUMPACK preconditioners are based on sparse LU factorization with rank-structured compression and can achieve near-linear complexity for a range of PDE problems, and support multiple right-hand sides. The STRUMPACK preconditioner accuracy can be tuned with the low-rank compression tolerance, offering a trade-off between compression tolerance (and thus set-up costs) and the approximation accuracy (presumably leading to a better preconditioner and fewer GMRES iterations). Having many time steps further emphasizes the importance of this trade-off. We also plan to study extensions to the non-linear case. We also continue investigating the field of values and spectral analysis as part of our ongoing research project.

To conclude, $\mathcal{P}^{I}_{Syl}$ has clearly outperformed the other considered preconditioners for the considered problems, especially at higher stages and with finer mesh sizes. The results indicate that continuing to develop and integrate advanced preconditioning techniques can make these methods even more efficient and scalable for large-scale computational problems.

REFERENCES

[1] R. Abu-Labdeh, S. MacLachlan, and P. E. Farrell, *Monolithic multigrid for implicit Runge–Kutta discretizations of incompressible fluid flow*, Journal of Computational Physics, 478 (2023), p. 111961, https://doi.org/https://doi.org/10.1016/j.jcp.2023.111961.

[2] R. Anderson, J. Andrej, A. Barker, J. Bramwell, J.-S. Camier, J. Cerveny, V. Dobrev, Y. Dudouit, A. Fisher, T. Kolev, W. Pazner, M. Stowell, V. Tomov, I. Akkerman, J. Dahm, D. Medina, and S. Zampini, *MFEM: A modular finite element methods library*, Computers & Mathematics with Applications, 81 (2021), pp. 42–74, https://doi.org/10.1016/j.camwa.2020.06.009.

[3] O. Axelsson, I. Dravins, and M. Neytcheva, *Stage-parallel preconditioners for implicit Runge–Kutta methods of arbitrarily high order, linear problems*, Numerical Linear Algebra with Applications, 31 (2024), p. e2532, https://doi.org/https://doi.org/10.1002/nla.2532.

[4] N. Bell, L. N. Olson, and J. Schroder, *PyAMG: Algebraic multigrid solvers in python*, Journal of Open Source Software, 7 (2022), p. 4142.

[5] T. A. Bickart, *An efficient solution process for implicit Runge–Kutta methods*, SIAM Journal on Numerical Analysis, 14 (1977), pp. 1022–1027.

[6] J. C. Butcher, *On the implementation of implicit Runge–Kutta methods*, BIT Numerical Mathematics, 16 (1976), pp. 237–240, https://doi.org/10.1007/BF01932265.

[7] L. Claus, P. Ghysels, Y. Liu, T. A. Nhan, R. Thirumalaisamy, A. P. S. Bhalla, and S. Li, *Sparse approximate multifrontal factorization with composite compression methods*, ACM Transactions on Mathematical Software, 49 (2023), pp. 1–28.

[8] M. R. Clines, *Optimality Theorems and Numerical Results on Block Preconditioners for Implicit Runge–Kutta Methods Applied to PDEs in Engineering and Biophysics*, PhD thesis, Texas Tech University, 2022.

[9] M. R. Clines, V. E. Howle, and K. R. Long, *Efficient Order-Optimal Preconditioners for Implicit Runge–Kutta and Runge–Kutta–Nyström Methods Applicable to a Large Class of Parabolic and Hyperbolic PDEs*, arXiv preprint arXiv:2206.08991, (2022).

[10] G. G. Dahlquist, *A special stability problem for linear multistep methods*, BIT Numerical Mathematics, 3 (1963), pp. 27–43.

[11] I. Dravins, S. Serra-Capizzano, and M. Neytcheva, *Spectral analysis of preconditioned matrices arising from stage-parallel implicit Runge–Kutta methods of arbitrarily high order*, SIAM Journal on Matrix Analysis and Applications, 45 (2024), pp. 1007–1034, https://doi.org/10.1137/23M1552498.

[12] M. Embree, *How descriptive are GMRES convergence bounds?*, 2023, https://arxiv.org/pdf/2209.01231.pdf. arXiv preprint: 2209.01231.

[13] P. E. Farrell, R. C. Kirby, and J. Marchena-Menéndez, *Irksome: Automating Runge–Kutta time-stepping for finite element methods*, ACM Transactions on Mathematical Software, 47 (2021), https://doi.org/10.1145/3466168.

[14] M. J. Gander and M. Outrata, *Spectral analysis of implicit 2-stage block Runge–Kutta preconditioners*, Linear Algebra and its Applications, (2023), https://doi.org/https://doi.org/10.1016/j.laa.2023.07.008.

[15] M. J. Gander and M. Outrata, *Spectral analysis of implicit s-stage block Runge–Kutta preconditioners*, SIAM Journal on Scientific Computing, in press, 46 (2024), pp. A2047–A2072, https://doi.org/10.1137/23M1604266, https://doi.org/10.1137/23M1604266, https://arxiv.org/abs/https://doi.org/10.1137/23M1604266.

[16] A. Greenbaum, V. Pták, and Z. Strakoš, *Any nonincreasing convergence curve is possible for GMRES*, SIAM Journal on Matrix Analysis and Applications, 17 (1996), pp. 465–469.

[17] E. Hairer, C. Lubich, and G. Wanner, *Geometric numerical integration*, Springer, 2000.

[18] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, vol. 14 of Springer Series in Computational Mathematics, Springer Berlin, Heidelberg, 1996, https://doi.org/https://doi.org/10.1007/978-3-642-05221-7.

[19] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward, *SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers*, ACM Transactions on Mathematical Software (TOMS), 31 (2005), pp. 363–396, https://doi.org/10.1145/1089014.1089020.

[20] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, 1994.

[21] *hypre: High performance preconditioners*. https://llnl.gov/casc/hypre, https://github.com/hypre-space/hypre.

[22] S. Leveque, L. Bergamaschi, A. Martínez, and J. W. Pearson, *Parallel-in-time solver for the all-at-once Runge–Kutta discretization*, SIAM Journal on Matrix Analysis and Applications, 45 (2024), pp. 1902–1928, https://doi.org/10.1137/23M1567862.

[23] X. S. Li and J. W. Demmel, *SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems*, ACM Transactions on Mathematical Software (TOMS), 29 (2003), pp. 110–140.

[24] J. Liesen and Z. Strakoš, *Krylov Subspace Methods: Principles and Analysis*, Oxford University Press, Oxford, 2013.

[25] Y. Liu, M. Jacquelin, P. Ghysels, and X. S. Li, *Highly scalable distributed-memory sparse triangular solution algorithms*, SIAM, Philadelphia, PA, USA, 2018, pp. 87–96, https://doi.org/10.1137/1.9781611975215.9.

[26] K. Long, P. T. Boggs, and B. G. van Bloemen Waanders, *Sundance: High-level software for PDE-constrained optimization*, Scientific Programming, 20 (2012), pp. 293–310.

[27] S. P. MacLachlan and C. W. Oosterlee, *Algebraic multigrid solvers for complex-valued matrices*, SIAM Journal on Scientific Computing, 30 (2008), pp. 1548–1571, https://doi.org/10.1137/070687232.

[28] K.-A. Mardal, T. K. Nilssen, and G. A. Staff, *Order-optimal Preconditioners for Implicit Runge–Kutta Schemes Applied to Parabolic PDEs*, SIAM Journal on Scientific Computing, 29 (2007), pp. 361–375.

[29] *MFEM: Modular finite element methods [Software]*. mfem.org, https://doi.org/10.11578/dc.20171025.1248.

[30] P. Munch, I. Dravins, M. Kronbichler, and M. Neytcheva, *Stage-parallel fully implicit Runge–Kutta implementations with optimal multilevel preconditioners at the scaling limit*, SIAM Journal on Scientific Computing, 46 (2024), pp. S71–S96, https://doi.org/10.1137/22M1503270, https://doi.org/10.1137/22M1503270, https://arxiv.org/abs/https://doi.org/10.1137/22M1503270.

[31] J. A. Nelder and R. Mead, *A simplex method for function minimization*, The computer journal, 7 (1965), pp. 308–313.

[32] W. Pazner and P.-O. Persson, *Stage-parallel fully implicit Runge–Kutta solvers for discontinuous Galerkin fluid simulations*, Journal of Computational Physics, 335 (2017), pp. 700–717, https://doi.org/https://doi.org/10.1016/j.jcp.2017.01.050.

[33] M. M. Rana, V. E. Howle, K. Long, A. Meek, and W. Milestone, *A New Block Preconditioner for Implicit Runge–Kutta Methods for Parabolic PDE Problems*, SIAM Journal on Scientific Computing, 43 (2021), pp. S475–S495.

[34] A. Rani, *Scalable Preconditioners for Implicit Runge–Kutta Methods: Sylvester Equations and Triangular Approximate Inverses.*, PhD thesis, Texas Tech University, 2024.

[35] P. J. Roache, *Building PDE codes to be verifiable and validatable*, Computing in Science & Engineering, 6 (2004), pp. 30–38, https://doi.org/10.1109/MCSE.2004.33.

[36] V. Simoncini, *Computational Methods for Linear Matrix Equations*, SIAM Review, 58 (2016), pp. 377–441, https://doi.org/10.1137/130912839.

[37] B. S. Southworth, O. Krzysik, and W. Pazner, *Fast solution of fully implicit Runge–Kutta and discontinuous Galerkin in time for numerical PDEs, part II: nonlinearities and daes*, SIAM Journal on Scientific Computing, 44 (2022), pp. A636–A663.

[38] B. S. Southworth, O. Krzysik, W. Pazner, and H. De Sterck, *Fast Solution of Fully Implicit Runge–Kutta and Discontinuous Galerkin in Time for Numerical PDEs, Part I: the Linear Setting*, SIAM Journal on Scientific Computing, 44 (2022), pp. A416–A443.

[39] G. A. Staff, K.-A. Mardal, and T. K. Nilssen, *Preconditioning of fully implicit Runge–Kutta schemes for parabolic PDEs*, Modeling, Identification and Control, 27 (2006), pp. 109–123, https://doi.org/10.4173/mic.2006.2.3.

[40] L. N. Trefethen and M. Embree, *Spectra and Pseudospectra: The Behaviour of Non-Normal Matrices and Operators*, Princeton University Press, Princeton, New Jersey, 2005.

[41] T. Trilinos Project Team, *The Trilinos Project Website*.

[42] C. F. Van Loan, *The ubiquitous Kronecker product*, Journal of Computational and Applied Mathematics, 123 (2000), pp. 85–100.

[43] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, *SciPy 1.0: Fundamental algorithms for scientific computing in python*, Nature Methods, 17 (2020), pp. 261–272, https://doi.org/10.1038/s41592-019-0686-2.

[44] Y. Voet, E. Sande, and A. Buffa, *A mathematical theory for mass lumping and its generalization with applications to isogeometric analysis*, Computer Methods in Applied Mechanics and Engineering, 410 (2023), p. 116033, https://doi.org/https://doi.org/10.1016/j.cma.2023.116033.

[45] S. Williams, A. Waterman, and D. Patterson, *Roofline: an insightful visual performance model for multicore architectures*, Communications of the ACM, 52 (2009), pp. 65–76.