LA-UR-20-30412
Accepted Manuscript

Fast Solution of Fully Implicit Runge--Kutta and Discontinuous Galerkin in Time for Numerical PDEs, Part I: the Linear Setting

Southworth, Benjamin Scott
Krzysik, Oliver
Pazner, Will
De Sterck, Hans

Provided by the author(s) and the Los Alamos National Laboratory (2022-06-13).

To be published in: SIAM Journal on Scientific Computing

DOI to publisher's version: 10.1137/21M1389742

Permalink to record:
http://permalink.lanl.gov/object/view?what=info:lanl-repo/lareport/LA-UR-20-30412

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by Triad National Security, LLC for the National Nuclear Security Administration of U.S. Department of Energy under contract 89233218CNA000001. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

Fast solution of fully implicit Runge-Kutta and discontinuous Galerkin in time for numerical PDEs, Part I: the linear setting*

Ben S. Southworth† Oliver A. Krzysik‡ Will Pazner§
Hans De Sterck¶

October 5, 2021

Abstract

Fully implicit Runge-Kutta (IRK) methods have many desirable properties as time integration schemes in terms of accuracy and stability, but high-order IRK methods are not commonly used in practice with numerical PDEs due to the difficulty of solving the stage equations. This paper introduces a theoretical and algorithmic preconditioning framework for solving the systems of equations that arise from IRK methods applied to linear numerical PDEs (without algebraic constraints). This framework also naturally applies to discontinuous Galerkin discretizations in time. Under quite general assumptions on the spatial discretization that yield stable time integration, the preconditioned operator is proven to have condition number bounded by a small, order-one constant, independent of the spatial mesh and time-step size, and with only weak dependence on number of stages/polynomial order; for example, the preconditioned operator for 10th-order Gauss IRK has condition number less than two, *independent of the spatial discretization and time step*. The new method can be used with arbitrary existing preconditioners for backward Euler-type time stepping schemes, and is amenable to the use of three-term recursion Krylov methods when the underlying spatial discretization is symmetric. The new method is demonstrated to be effective on various high-order finite-difference and finite-element discretizations of linear parabolic and hyperbolic problems, demonstrating fast, scalable solution of up to 10th order accuracy. The new method consistently outperforms existing block preconditioning approaches, and in several cases, the new method can achieve 4th-order accuracy using Gauss integration with roughly half the number of preconditioner applications and wallclock time as required using standard diagonally implicit RK methods.

*BSS was supported by Lawrence Livermore National Laboratory under contract B639443, and as a Nicholas C. Metropolis Fellow under the Laboratory Directed Research and Development program of Los Alamos National Laboratory. OAK acknowledges the support of an Australian Government Research Training Program (RTP) Scholarship.

†Theoretical Division, Los Alamos National Laboratory, U.S.A. (southworth@lanl.gov), http://orcid.org/0000-0002-0283-4928

‡School of Mathematics, Monash University, Australia (oliver.krzysik@monash.edu), https://orcid.org/0000-0001-7880-6512

§Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, U.S.A. (pazner1@llnl.gov)

¶Department of Applied Mathematics, University of Waterloo, Waterloo, Canada (hdesterck@uwaterloo.ca)

1 Introduction

1.1 Fully implicit Runge-Kutta

Consider the method-of-lines approach to the numerical solution of linear partial differential equations (PDEs), where we discretize in space and arrive at a system of ordinary differential equations (ODEs) in time,

$$M\mathbf{u}'(t) = \mathcal{L}\mathbf{u} + \hat{\mathbf{f}}(t) \quad \text{in } (0,T], \quad \mathbf{u}(0) = \mathbf{u}_0,$$

where M is a mass matrix, $\mathcal{L} \in \mathbb{R}^{N\times N}$ a discrete linear operator, and $\hat{\mathbf{f}}(t)$ a time-dependent forcing function.[1] Then, consider time propagation using an s-stage Runge-Kutta scheme, characterized by the Butcher tableau

$$\begin{array}{c|c} \mathbf{c}_0 & A_0 \\ \hline & \mathbf{b}_0^T \end{array},$$

with Runge-Kutta matrix $A_0 = \{a_{ij}\} \in \mathbb{R}^{s\times s}$, weight vector $\mathbf{b}_0^T = (b_1, \ldots, b_s)^T$, and quadrature nodes $\mathbf{c}_0 = (c_1, \ldots, c_s)$.

Runge-Kutta methods update the solution using a sum over stage vectors,

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \delta t \sum_{i=1}^{s} b_i \mathbf{k}_i, \tag{1}$$

$$M\mathbf{k}_i = \mathcal{L}\left(\mathbf{u}_n + \delta t \sum_{j=1}^{s} a_{ij}\mathbf{k}_j\right) + \mathbf{f}(t_n + \delta t c_i). \tag{2}$$

The stage vectors $\{\mathbf{k}_i\}$ can then be expressed as the solution of the block linear system,

$$\left(\begin{bmatrix} M & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & M \end{bmatrix} - \delta t \begin{bmatrix} a_{11}\mathcal{L} & \ldots & a_{1s}\mathcal{L} \\ \vdots & \ddots & \vdots \\ a_{s1}\mathcal{L} & \ldots & a_{ss}\mathcal{L} \end{bmatrix}\right) \begin{bmatrix} \mathbf{k}_1 \\ \vdots \\ \mathbf{k}_s \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \vdots \\ \mathbf{f}_s \end{bmatrix}, \tag{3}$$

where $\mathbf{f}_i := \hat{\mathbf{f}}(t_n + \delta t c_i) + \mathcal{L}(t_n + \delta t c_i)\mathbf{u}_n$. Since $\mathcal{L}$ is the same at all stages (independent of time), (3) is often expressed in the equivalent compact Kronecker product form

$$(I \otimes M - \delta t A_0 \otimes \mathcal{L})\mathbf{k} = \mathbf{f}. \tag{4}$$

The difficulty in fully implicit Runge-Kutta methods (which we will denote IRK) lies in solving the $Ns \times Ns$ block linear system in (3). This paper focuses on the simulation of numerical PDEs, where N is typically very large and $\mathcal{L}$ is highly ill-conditioned. In such cases, direct solution techniques to solve (3) are not a viable option, and fast, parallel preconditioned iterative methods must be used (with an effective preconditioner being the key). However, higher-order IRK methods are not commonly employed in practice due to the difficulties of solving (3). Even for relatively simple parabolic PDEs where $-\mathcal{L}$ is symmetric positive definite (SPD), (3) is a large nonsymmetric matrix with significant block coupling. It is well-known in preconditioning fields, including (algebraic) multigrid, sparse approximate inverses, block preconditioning, etc., that nonsymmetric operators and/or systems with block structure (particularly more than two blocks/variables) generally introduce significant difficulties, and many methods do not extend well to nonsymmetric systems and/or block systems.

[1] Note, PDEs with an algebraic constraint, such as the divergence-free constraint in the Stokes equations, instead yield a differential algebraic equation (DAE), which requires separate careful treatment, and is addressed in a companion paper along with nonlinearities [50].

Remark 1 (Discontinuous Galerkin (DG) in time)**.** *DG-in-time discretizations of systems of linear ODEs give rise to linear algebraic systems of the form*

$$\left(\begin{bmatrix} \delta_{11} M & & \delta_{1s} M \\ & \ddots & \\ \delta_{s1} M & & \delta_{ss} M \end{bmatrix} - \delta t \begin{bmatrix} t_{11} \mathcal{L}_1 & \dots & t_{1s} \mathcal{L}_1 \\ \vdots & \ddots & \vdots \\ t_{s1} \mathcal{L}_s & \dots & t_{ss} \mathcal{L}_s \end{bmatrix} \right) \begin{bmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_s \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_s \end{bmatrix}. \tag{5}$$

The coefficients $T = \{t_{ij}\}$ correspond to a temporal mass matrix, the coefficients $\boldsymbol{\delta} = \{\delta_{ij}\}$ correspond to a DG weak derivative with upwind numerical flux, and the unknowns $\mathbf{u}_i$ are the coefficients of the polynomial expansion of the approximate solution (for example, see [1, 27, 30, 47]). Both of the coefficient matrices $T, \boldsymbol{\delta} \in \mathbb{R}^{s \times s}$ are invertible. It can be seen that the algebraic form of the DG in time discretization is closely related to the implicit Runge-Kutta system (3). In fact, in the case where $\mathcal{L}_i = \mathcal{L}$ for all i, the system (5) can be recast in the form of (3) using the invertibility of $\boldsymbol{\delta}$, with $A_0 = \boldsymbol{\delta}^{-1} T$. In particular, the degree-p DG method using $(p+1)$-point Radau quadrature, which is exact for polynomials of degree $2p$, is equivalent to the Radau IIA collocation method [30], which is used for many of the numerical results in Section 4. Thus, although the remainder of this paper focuses on fully implicit Runge-Kutta, the algorithms developed here can also be applied to DG discretizations in time on fixed slab-based meshes.

1.2 Outline

This paper develops a novel framework for the solution of fully implicit Runge-Kutta methods and DG discretizations in time for linear numerical PDEs, with theoretically guaranteed preconditioning, *independent of the spatial discretization.* The new method requires the preconditioning of s real-valued matrices of the form $\gamma M - \delta t \mathcal{L}$ for some $\gamma > 0$, analogous to the matrices that arise in backward Euler integration, and is easily implemented using existing preconditioners and parallel software libraries.

Section 2.1 provides background on why IRK methods are desirable over the simpler and more commonly used diagonally implicit Runge-Kutta (DIRK) methods, and also provides some historical context for the preconditioners developed in this work. Section 2.2 then briefly discusses stable integration from a method-of-lines perspective and introduces two key elements that will be used throughout the paper.

Section 3 introduces the new theoretical and algorithmic framework for solving for the IRK update in (1). Theory is developed in Section 3.2 that guarantees the condition number of the preconditioned operator is bounded by a small, order-one constant under basic assumptions on stability from Section 2.2. The condition number of the preconditioned system is *asymptotically optimal* in the sense that it is bounded independent of the spatial mesh and time step, and with only weak dependence on the number of stages or polynomial order. For example, the preconditioned operator for 10th-order Gauss IRK has condition number less than two. Moreover, the conditioning results are *independent of the underlying spatial discretization*, making the proposed method robust and, to-some-extent, black-box. To our knowledge, [24] is the only other work to prove guaranteed preconditioning or convergence for solving IRK methods applied to arbitrary linear PDEs.[2] In addition, in contrast to other works that have considered the preconditioning of (3), the proposed algorithm here (i) is amenable to short-term Krylov recursion (conjugate gradient (CG)/MINRES) if $\gamma M - \mathcal{L}$ is, and (ii) only operates on the solution, thus not requiring the storage of each stage vector.

[2]Some papers have considered bounds on the spectral radius of the preconditioned operator, e.g., [13], but for non-SPD spatial operators, such bounds can be a poor indicator of convergence, e.g., see [31].

Numerical results are provided in Section 4, demonstrating the new method for a variety of problems and corresponding preconditioners, including very high-order finite-difference and DG spatial discretizations of advection-diffusion equations, and matrix-free continuous Galerkin discretizations of diffusion equations. The method is shown to be fast and scalable up to 10th-order accuracy in time, effective on fully advective (hyperbolic) problems, and, for multiple examples, can obtain 4th-order accuracy with Gauss integration using roughly half as many preconditioning iterations and wallclock time as needed by standard 4th-order SDIRK schemes.

The methods are implemented with the MFEM [2] library and available at `https://github.com/bensworth/IRKIntegration`.

2 Background

2.1 Motivation and previous work

Diagonally implicit Runge-Kutta methods (DIRK), where A_0 is lower triangular, are commonly used in practice [25]. For such schemes, the solution of (3) using a block substitution algorithm requires only s linear solves of systems of the form $M - \delta t a_{ii}\mathcal{L}$. Unfortunately, DIRK schemes suffer from order reduction, where the order of accuracy observed in practice on stiff nonlinear PDEs or DAEs can be limited to $\approx \min\{p, q+1\}$ or q, respectively, for formal integration order p and stage-order q [18, 25]. The stage-order of a DIRK method is at most one (EDIRK methods, with one explicit stage, have a maximum stage order of two) and, thus, even a 6th-order DIRK method may only yield first- or second-order accuracy [9]. In contrast, IRK methods may have arbitrarily high stage order and, thus, formally high-order accuracy on stiff, nonlinear problems, and even index-2 DAEs [18]. Although the focus of this paper is linear PDEs without algebraic constraints, we want to highlight that the theory and framework developed here is fundamental to a companion paper on nonlinear PDEs and DAEs [50]. Furthermore, for less stiff problems, IRK methods can yield accuracy as high as order $2s$ for an s-stage method, compared with a maximum of s or $s+1$ for SDIRK methods with reasonable stability properties [18, Section IV.6],[25]. Multistep methods can overcome some of the accuracy constraints of SDIRK methods, but implicit multistep methods cannot be A-stable and greater than order two, which is limiting when considering advection-dominated or hyperbolic problems, where the field-of-values often push up against the imaginary axis. Furthermore, for problems where symplectic integration is desirable for conservation, neither linear multistep nor explicit methods can be generally symplectic (i.e., for non-separable problems) [19]. Although DIRK methods can be symplectic, they are limited to at most 4th order and, moreover, known methods above second order are impractical due to negative diagonal entries of A_0 (leading to a negative shift rather than positive shift of the spatial discretization) [25]. Thus, even moderate order symplectic integration requires IRK methods.

Many papers have considered the solution of (4), with Butcher [10] and Bickart's [6] being some of the earliest works, which develop ways to transform (4) to a simpler form.[3] There, and in many of the works that followed, the goal was to minimize the cost of LU decompositions used to solve (4), typically in the context of ODEs. For large-scale simulation of PDEs, particularly on modern computing architectures, LU decompositions (or other direct factorizations) are typically not feasible. In this vein, a number of people have considered preconditioning techniques for (4) or approximations

[3] In Kronecker form (4), SIRK methods [36] are also relatively straightforward to solve using existing preconditioning techniques. But, although SIRK methods offer some advantages over DIRK methods, they still lack the favorable stability and accuracy properties of IRK methods [8, 38].

to (4) on the nonlinear iteration level or time discretization level. Various block preconditioning/approximation techniques have been studied, primarily for parabolic problems [20, 21, 22, 23, 33, 35, 51], and multigrid methods for IRK and parabolic problems were developed in [28]. New ADI-type preconditioners for IRK methods were developed for parabolic problems in [12] with spectral radius shown to be < 1 under reasonable assumptions, and the method extended to the viscous wave equation in first-order form in [13]. More recently, block ILU preconditioners were successfully applied to a transformed version of (3) in [41] on more difficult nonlinear compressible fluids problems. A handful of works have also studied linear solvers for DG-in-time discretizations, primarily for parabolic problems, including block preconditioning approaches [5, 44, 49], and direct space-time multigrid methods [17]. In fact, some of the principles used in this paper are similar to those used in [5] for space-time DG discretizations of linear parabolic problems, and some of the theory derived therein is generalized to non-parabolic/non-SPD operators in this paper.

Despite many papers considering the efficient solution of IRK/DG-in-time methods, very little has been done in the development and analysis of preconditioning techniques for non-parabolic problems/non-SPD spatial operators, particularly methods that are amenable to combine with existing fast, parallel preconditioners (unlike, e.g., the block ILU approach in [40]). To our knowledge, no methods have been developed with theoretical guarantees of effectiveness/robustness for a wide range of problems. Here, we develop a preconditioning framework for linear PDEs that can be used with arbitrary existing preconditioners/linear solvers, and is guaranteed to provide effective preconditioning under only minor assumptions on the definiteness of the spatial operator (see Assumption 2). This provides a robust, almost-black-box method that can be quickly added to existing codes with implicit integration to support IRK integration for linear PDEs.

C++ Code for the IRK preconditioners developed here is built on the MFEM library [2], and available at `https://github.com/bensworth/IRKIntegration`.

Remark 2 (Growing interest in IRK)**.** *It is worth pointing out that while writing this paper, at least three preprints have been posted online studying the use of IRK methods for numerical PDEs. Two papers develop new block preconditioning techniques for parabolic PDEs [24, 42], and one focuses on a high-level numerical implementation of IRK methods with the Firedrake package [16].*

2.2 A preconditioning framework and stability

Throughout the paper, we use the reformulation used in, for example, [41], where we can pull an $A_0 \otimes I$ out of the fully implicit system in (3), yielding the equivalent problem

$$\left(A_0^{-1} \otimes M - \delta t I \otimes \mathcal{L}\right)(A_0 \otimes I)\mathbf{k} = \mathbf{f}. \tag{6}$$

The off-diagonal block coupling in (6) now consists of mass matrices rather than differential operators, which makes the analysis and solution more tractable. The algorithms developed here depend on the eigenvalues of A_0 and A_0^{-1}, leading to our first assumption.

Assumption 1. *Assume that all eigenvalues of A_0 (and equivalently A_0^{-1}) have positive real part.*

Recall that if an IRK method is A-stable, irreducible, and A_0 is invertible (which includes DIRK, Gauss, Radau IIA, and Lobatto IIIC methods, among others), then Assumption 1 holds [18]; that is, Assumption 1 is straightforward to satisfy in practice.

Stability must be taken into consideration when applying ODE solvers within a method-of-lines approach to numerical PDEs. The Dalhquist test problem extends naturally to this setting, where we are interested in the stability of the linear operator $\mathcal{L}$,

for the ODE(s) $\mathbf{u}'(t) = \mathcal{L}\mathbf{u}$, with solution $e^{t\mathcal{L}}\mathbf{u}$. A necessary condition for stability is that the eigenvalues of $\delta t\mathcal{L}$ lie within the region of stability for the Runge-Kutta scheme of choice (e.g., see [43]). Here we are interested in implicit schemes and, because most implicit Runge-Kutta schemes used in practice are A- or L-stable, an effectively necessary condition for stability is that the real part of eigenvalues of $\mathcal{L}$ be nonpositive. For normal matrices, this requirement ends up being a necessary and sufficient condition for stability.

For non-normal or non-diagonalizable operators, the analysis is more complicated. One of the best known works on the subject is by Reddy and Trefethen [43], where necessary and sufficient conditions for stability are derived as the ε pseudo-eigenvalues of $\mathcal{L}$ being within $\mathcal{O}(\varepsilon) + \mathcal{O}(\delta t)$ of the stability region as $\varepsilon, \delta t \to 0$. Here we relax this assumption to something that is more tractable to work with by noting that the ε pseudo-eigenvalues are contained within the field of values to $\mathcal{O}(\varepsilon)$ [52, Eq. (17.9)], where the field of values is defined as

$$W(\mathcal{L}) := \{\langle \mathcal{L}\mathbf{x}, \mathbf{x}\rangle \,:\, \|\mathbf{x}\| = 1\}\,. \tag{7}$$

This motivates the following assumption for the analysis done in this paper:

Assumption 2. *Let $\mathcal{L}$ be the linear spatial operator, and assume that $W(\mathcal{L}) \leq 0$ (that is, $W(\mathcal{L})$ is a subset of the left half plane (including imaginary axis)).*

Note that if $\mathcal{L}$ is normal, then Assumption 2 is equivalent to the real parts of the eigenvalues of $\mathcal{L}$ being in the closed left-half plane since $W(\mathcal{L})$ is the convex hull of the eigenvalues.

It should be noted that the field of values has an additional connection to stability. From [52, Theorem 17.1], we have that $\|e^{t\mathcal{L}}\| \leq 1$ for all $t \geq 0$ if and only if $W(\mathcal{L}) \leq 0$. This is analogous to the "strong stability" discussed by Leveque [29, Chapter 9.5], as opposed to the weaker (but still sufficient) condition $\|e^{t\mathcal{L}}\| \leq C$ for all $t \geq 0$ and some constant C. In practice, Assumption 2 often holds when simulating numerical PDEs, and in Section 3.2 it is proven that Assumptions 1 and 2 guarantee the methods proposed here yield a small, bounded condition number of the preconditioned operator. Specifically, conditioning depends on the eigenvalues of A_0, but the condition number of the preconditioned operator is < 2.5 for all Gauss, Radau, and Lobatto schemes tested here, up to 5 stages, and sees only slow growth in the total number of RK stages. It should also be noted that $\mathcal{L}$ need not be nonsingular.

3 Preconditioning the stage matrix

For ease of notation, let us scale both sides of (6) by a block diagonal operator, with diagonal blocks M^{-1}, and let

$$\widehat{\mathcal{L}} := \delta t M^{-1}\mathcal{L}, \tag{8}$$

for $i = 1, ..., s$. Now let α_{ij} denote the ij-element of A_0^{-1}.[4] Then, solving (6) can be effectively reduced to inverting the operator

$$\mathcal{M}_s := A_0^{-1} \otimes I - I \otimes \widehat{\mathcal{L}} = \begin{bmatrix} \alpha_{11}I - \widehat{\mathcal{L}} & \alpha_{12}I & \cdots & \alpha_{1s}I \\ \alpha_{21}I & \alpha_{22}I - \widehat{\mathcal{L}} & \cdots & \alpha_{2s}I \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{s1}I & \cdots & \cdots & \alpha_{ss}I - \widehat{\mathcal{L}} \end{bmatrix}. \tag{9}$$

[4]Note, there are methods with one explicit stage followed by several fully implicit stages [9]. In such cases, A_0 is not invertible, but the explicit stage can be eliminated from the system (by doing an explicit time step). The remaining operator can then be reformulated as in (6).

We proceed by deriving a closed form inverse of (9), demonstrating how the Runge-Kutta update in (1) can then be performed directly (without forming and saving each stage vector), and developing a preconditioning strategy to apply this update using existing preconditioners. Note, in practice we do *not* directly form $\widehat{\mathcal{L}}$, as M^{-1} is often a dense matrix. Rather, it is a theoretical tool to simplify notation; in practice mass-matrix inverses are applied via either preconditioned CG or direct inverse when feasible (e.g., for DG in space). See Algorithm 1 for a practical description of the final algorithm with mass matrices.

3.1 An inverse and update for commuting operators

This section introduces a result similar to Bickart's [6], but using a different framework. We consider $\mathcal{M}_s$ as a matrix over the commutative ring of linear combinations of $\{I, \widehat{\mathcal{L}}\}$, and the determinant and adjugate referred to in Lemma 1 are defined over matrix-valued elements rather than scalars. For the interested reader, see [7] for details on matrices and the corresponding linear algebra when matrix elements are defined over a space of commuting matrices.

Lemma 1. *Let α_{ij} denote the (i,j)th entry of A_0^{-1} and define $\mathcal{M}_s$ as in (9). Let $\det(\mathcal{M}_s)$ be the determinant of $\mathcal{M}_s$, $\mathrm{adj}(\mathcal{M}_s)$ be the adjugate of $\mathcal{M}_s$, and $P_s(x)$ be the characteristic polynomial of A_0^{-1}. Then, $\mathcal{M}_s$ is invertible if and only if $\det(\mathcal{M}_s)$ is not singular, and*

$$\mathcal{M}_s^{-1} = \big(I_s \otimes P_s(\widehat{\mathcal{L}})^{-1}\big)\mathrm{adj}(\mathcal{M}_s),$$

Proof. Notice in (9) that $\mathcal{M}_s$ is a matrix over the commutative ring of linear combinations of I and $\widehat{\mathcal{L}}$. A classical result in matrix analysis [7] tells us that

$$\mathrm{adj}(\mathcal{M}_s)\mathcal{M}_s = \mathcal{M}_s\mathrm{adj}(\mathcal{M}_s) = (I_s \otimes \det(\mathcal{M}_s))I.$$

Moreover, $\mathcal{M}_s$ is invertible if and only if the determinant of $\mathcal{M}_s$ is invertible, in which case $\mathcal{M}_s^{-1} = (I_s \otimes \det(\mathcal{M}_s)^{-1})\mathrm{adj}(\mathcal{M}_s)$ [7, Theorem 2.19 & Corollary 2.21]. Moreover, notice that $\mathcal{M}_s$ takes the form $A_0^{-1} - \widehat{\mathcal{L}}I$ over the commutative ring defined above. Analogous to a matrix defined over the real or complex numbers, the determinant of $A_0^{-1} - \widehat{\mathcal{L}}I$ is the characteristic polynomial of A_0^{-1} evaluated at $\widehat{\mathcal{L}}$, which completes the proof. □

Returning to (6), let $\hat{\mathbf{f}} = (I_s \otimes M^{-1})\mathbf{f}$, which applies the mass-matrix inverse in (8) to the right-hand side. Then, we can express the solution for the set of all stage vectors $\mathbf{k} = [\mathbf{k}_1; ...; \mathbf{k}_s]$ as

$$\mathbf{k} := \big(I_s \otimes \det(\mathcal{M}_s)^{-1}\big)\,(A_0^{-1} \otimes I)\mathrm{adj}(\mathcal{M}_s)(I_s \otimes M^{-1})\mathbf{f},$$

where $\mathbf{f} = [\mathbf{f}_1; ...; \mathbf{f}_s]$ (note that $A_0 \otimes I$ commutes with $(I_s \otimes \det(\mathcal{M}_s)^{-1})$). The Runge-Kutta update is then given by

$$\begin{aligned}\mathbf{u}_{n+1} &= \mathbf{u}_n + \delta t \sum_{i=1}^{s} b_i \mathbf{k}_i \\ &= \mathbf{u}_n + \delta t \det(\mathcal{M}_s)^{-1}(\mathbf{b}_0^T A_0^{-1} \otimes I)\mathrm{adj}(\mathcal{M}_s)(I_s \otimes M^{-1})\mathbf{f}. \end{aligned} \tag{10}$$

Remark 3 (Implementation & complexity)**.** *The adjugate consists of linear combinations of I and $\widehat{\mathcal{L}}$, and an analytical form can be derived for an arbitrary $s \times s$ matrix for small s. Applying its action requires a set of vector summations and matrix-vector*

multiplications. In particular, the diagonal elements of $\text{adj}(\mathcal{M}_s)$ *are monic polynomials in* $\widehat{\mathcal{L}}$ *of degree* $s-1$ *and off-diagonal terms are polynomials in* $\widehat{\mathcal{L}}$ *of degree* $s-2$.

Returning to (10), we consider two cases. First, if a given Runge-Kutta scheme is stiffly accurate (for example, Radau IIA methods), then $\mathbf{b}_0^T A_0^{-1} = [0, ..., 0, 1]$. *This yields the nice simplification that computing the update in (10) only requires applying the last row of* $\text{adj}(\mathcal{M}_s)$ *to* $\hat{\mathbf{f}}$ *(in a dot-product sense) and applying* $\det(\mathcal{M}_s)^{-1}$ *to the result. From the discussion above regarding the adjugate structure, applying the last row of* $\text{adj}(\mathcal{M}_s)$ *requires* $(s-2)(s-1)+(s-1) = (s-1)^2$ *matrix-vector multiplications. Because this only happens once, followed by the linear solve(s), these multiplications are typically of relatively marginal cost.*

In the more general case of non-stiffly accurate methods (for example, Gauss methods), one can obtain an analytical form for $(\mathbf{b}_0^T A_0^{-1} \otimes I)\text{adj}(\mathcal{M}_s)$. *Each element in this block* $1 \times s$ *matrix consists of polynomials in* $\widehat{\mathcal{L}}$ *of degree* $s-1$ *(although typically not monic). Compared with stiffly accurate schemes, this now requires* $(s-1)s$ *matrix-vector multiplications, which is* $s-1$ *more than for stiffly accurate schemes, but still typically of marginal overall computational cost. For more information, see Algorithm 1 and the discussion that follows it.*

3.2 Preconditioning by conjugate pairs

Following the discussion and algorithm developed in Section 3.1, the key outstanding point in computing $\mathbf{u}_{n+1}$ using the update (10) is inverting $P_s(\widehat{\mathcal{L}})$, where $P_s(x)$ is the characteristic polynomial of A_0^{-1} (see Lemma 1).

In contrast to much of the early work on solving IRK systems, where LU factorizations were the dominant cost and system sizes relatively small, explicitly forming and inverting $P_s(\widehat{\mathcal{L}})$ for numerical PDEs is typically not a viable option in high-performance simulation on modern computing architectures. Instead, by computing the eigenvalues $\{\lambda_i\}$ of A_0^{-1}, we can express $P_s(\widehat{\mathcal{L}})$ in a factored form,

$$P_s(\widehat{\mathcal{L}}) = \prod_{i=1}^{s} (\lambda_i I - \widehat{\mathcal{L}}), \tag{11}$$

and its inverse can then be computed by successive applications of $(\lambda_i I - \widehat{\mathcal{L}})^{-1}$, for $i = 1, ..., s$. Unfortunately, eigenvalues of A_0 and A_0^{-1} are often complex, and for real-valued matrices this makes the inverse of individual factors $(\lambda_i I - \widehat{\mathcal{L}})^{-1}$ more difficult and often impractical with standard preconditioners and existing software. Moving forward, let $\lambda := \eta + \mathrm{i}\beta$ denote an eigenvalue of A_0^{-1}, for $\eta, \beta \in \mathbb{R}$, with $\beta \geq 0$ and $\eta > 0$ under Assumption 1.

Here, we combine conjugate eigenvalues into quadratic polynomials that we must precondition, which take the form

$$\begin{aligned} \mathcal{Q}_\eta :&= ((\eta + \mathrm{i}\beta)I - \widehat{\mathcal{L}})((\eta - \mathrm{i}\beta)I - \widehat{\mathcal{L}}) \\ &= (\eta^2 + \beta^2)I - 2\eta\widehat{\mathcal{L}} + \widehat{\mathcal{L}}^2 = (\eta I - \widehat{\mathcal{L}})^2 + \beta^2 I. \end{aligned} \tag{12}$$

We then express (11) as a product of $\mathcal{Q}_{\eta_j}$ (12), for $j = 1, ..., s/2$, $P_s(\widehat{\mathcal{L}}) = \prod_{j=1}^{s/2} \mathcal{Q}_{\eta_j}$, and solve each successive quadratic operator $\mathcal{Q}_{\eta_j}$ (note, for odd s there will also be a term $(\lambda_i I - \widehat{\mathcal{L}})$ corresponding to the real eigenvalue of the Butcher tableau). In practice, we typically do not want to directly form or precondition a quadratic operator like (12), due to (i) the overhead cost of large parallel matrix multiplication, (ii) the fact that many fast parallel methods such as multigrid are not well-suited for solving a polynomial in $\widehat{\mathcal{L}}$, and (iii) it is increasingly common that even $\mathcal{L}$ is only available as

a partially-assembled/matrix-free operator. The point of (12) is that by considering conjugate pairs of eigenvalues, the resulting operator is real-valued. To invert (11), we fully resolve the inverse for one conjugate pair of eigenvalues (12) before moving onto the next; this avoids potentially compounding condition numbers if we tried to invert the full polynomial (11) all-at-once. Moreover, then we only need to store a solver for one pair of eigenvalues at a time.

3.3 Condition-number optimal conjugate preconditioning

This section develops a preconditioner for $\mathcal{Q}_\eta$ such that the condition number of the preconditioned operator is bounded by a small, order-one constant, *independent of* $\widehat{\mathcal{L}}$. The preconditioner is optimal over the space of general preconditioners $(\delta I - \widehat{\mathcal{L}})^{-1}(\gamma I - \widehat{\mathcal{L}})^{-1}$, for $\delta, \gamma \in (0, \infty)$, in terms of minimizing the maximum condition number over all $\widehat{\mathcal{L}}$. Furthermore, the condition number of the preconditioned system is asymptotically optimal in the sense that it is bounded independent of δt and spatial mesh spacing, h, and has only weak dependence on the order of time integration. The analysis derived herein is based on the assumption that a small bounded condition number corresponds to better preconditioners for nonsymmetric matrices.

Given that (12) is a quadratic polynomial in $\widehat{\mathcal{L}}$, consider defining a preconditioner as a *factored* quadratic polynomial in $\widehat{\mathcal{L}}$, $[(\delta I - \widehat{\mathcal{L}})(\gamma I - \widehat{\mathcal{L}})]^{-1}$, for $\gamma, \delta > 0$, where we can invert the two factors separately. The preconditioned operator then takes the form

$$\mathcal{P}_{\delta,\gamma} \coloneqq (\delta I - \widehat{\mathcal{L}})^{-1}(\gamma I - \widehat{\mathcal{L}})^{-1}\Big[(\eta I - \widehat{\mathcal{L}})^2 + \beta^2 I\Big]. \tag{13}$$

Such an approach was proven effective for symmetric definite spatial matrices in [5], where it is assumed $\gamma = \delta$, and the constant $\gamma = \gamma_* = \sqrt{\eta^2 + \beta^2}$ is derived to be optimal in a certain sense. Theorem 1 in Appendix A derives tight bounds on the maximum condition number of $\mathcal{P}_{\delta,\gamma}$ (13) over all $\widehat{\mathcal{L}}$ that satisfy Assumption 2, and further derives $\delta, \gamma \in (0, \infty)$ that minimize this upper bound. Corollary 1 below shows that the optimal factored quadratic preconditioner (14) over all $\delta, \gamma \in (0, \infty)$, in terms of minimizing the maximum ℓ^2-condition number over all $\widehat{\mathcal{L}}$ that satisfy Assumption 2, is obtained by setting $\delta = \gamma = \gamma_* \coloneqq \sqrt{\eta^2 + \beta^2}$. The preconditioned operator then takes the form[5]

$$\mathcal{P}_{\gamma_*} \coloneqq (\gamma_* I - \widehat{\mathcal{L}})^{-2}\Big[(\eta I - \widehat{\mathcal{L}})^2 + \beta^2 I\Big]. \tag{14}$$

Corollary 1 (Condition-number bounds, independent of $\widehat{\mathcal{L}}$)**.** *The maximum ℓ^2 condition number of $\mathcal{P}_{\delta,\gamma}$ (13) over* all *$\widehat{\mathcal{L}}$ that satisfy Assumption 2 is minimized over $\delta, \gamma \in (0, \infty)$ when $\delta = \gamma = \gamma_*$, with*

$$\gamma_* = \sqrt{\eta^2 + \beta^2}. \tag{15}$$

That is to say, the preconditioner $(\gamma_* I - \widehat{\mathcal{L}})^{-2}$ is optimal over the space of general preconditioners $(\delta I - \widehat{\mathcal{L}})^{-1}(\gamma I - \widehat{\mathcal{L}})^{-1}$, for $\delta, \gamma \in (0, \infty)$, in terms of minimizing the maximum condition number over all $\widehat{\mathcal{L}}$. Furthermore, the condition number of the preconditioned operator $\mathcal{P}_{\gamma_*}$ (14) is bounded for all $\widehat{\mathcal{L}}$ via

$$\kappa(\mathcal{P}_{\gamma_*}) \leq \sqrt{1 + \frac{\beta^2}{\eta^2}}, \tag{16}$$

and $\exists$ some $\widehat{\mathcal{L}}$ such that (16) is satisfied with equality.

[5]Note that in (13), the preconditioner $((\delta I - \widehat{\mathcal{L}})^{-1}(\gamma I - \widehat{\mathcal{L}})^{-1})$ and the operator $((\eta I - \widehat{\mathcal{L}})^2 + \beta^2 I)$ commute, and so left and right preconditioning are equivalent.

Proof. See Appendix A. □

Remark 4 (Three-term recursion)**.** *Note that for a given conjugate pair of eigenvalue, suppose* $(\eta I - \widehat{\mathcal{L}})$ *is SPD and* $(\gamma_* I - \widehat{\mathcal{L}})^{-1}$ *some SPD preconditioner (with modified constant, which should not affect definiteness). Then, the associated quadratic operator* $\mathcal{Q}_\eta$ (12) *and preconditioner* $(\gamma_* I - \widehat{\mathcal{L}})^{-2}$ *are also both SPD. It follows that if CG/MINRES can be applied to backward Euler or SDIRK schemes, it can also be applied to the quadratic operators arising here.*

Remark 5 (Mass matrices)**.** *Recall in the finite element context where mass matrices are involved, we defined* $\widehat{\mathcal{L}} := \delta t M^{-1}\mathcal{L}$ *as a theoretical tool. In practice, we do not form* $\widehat{\mathcal{L}}$ *directly. Factoring the* M^{-1} *off of the left, the quadratic polynomial* (12) *for a given conjugate pair of eigenvalues can be expressed as*

$$\mathcal{Q}_\eta = M^{-1}(\eta M - \delta t\mathcal{L})M^{-1}(\eta M - \delta t\mathcal{L}) + \beta^2 I \tag{17}$$

To invert $\mathcal{Q}_\eta$ *iteratively, it is best to first scale both sides of the linear system by* M*, so we iterate on* $M\mathcal{Q}_\eta$ *(see also Algorithm 1). Each Krylov or fixed-point iteration requires applying the operator to compute a residual, and applying* $M\mathcal{Q}_\eta$ *only requires computing* M^{-1} *once, while applying* $\mathcal{Q}_\eta$ *requires computing* M^{-1} *twice, thus halving the number of times* M^{-1} *must be applied each iteration. Moreover,* $\mathcal{Q}_\eta$ *is not SPD, but if* M *and* $\mathcal{L}$ *are Hermitian,* $M\mathcal{Q}_\eta$ is *SPD, as is the preconditioner* $(\gamma_* I - \delta t\mathcal{L})^{-1}M(\gamma_* I - \delta t\mathcal{L})^{-1}$*, thus allowing the use of CG or MINRES acceleration analogous to Remark 4.*

Table 1 provides condition number bounds from Corollary 1 and (16) for Gauss, Radau IIA, and Lobatto IIIC Runge-Kutta methods. Note that γ_* is different for each conjugate eigenvalue pair and each IRK method. It also should be pointed out that the formally optimal γ in terms of minimizing condition number is not the same for all $\widehat{\mathcal{L}}$; rather, here we develop a constant γ_* that is *robust and effective* for all $\widehat{\mathcal{L}}$, and does not require additional analysis (analytical or numerical) as would be necessary to tune γ to a specific operator.

Stages	2	3		4		5		
	$\lambda^{\pm}_{1,2}$	λ_1	$\lambda^{\pm}_{2,3}$	$\lambda^{\pm}_{1,2}$	$\lambda^{\pm}_{3,4}$	λ_1	$\lambda^{\pm}_{2,3}$	$\lambda^{\pm}_{4,5}$
Gauss	1.15	1.00	1.38	1.61	1.04	1.00	1.83	1.13
Radau IIA	1.22	1.00	1.51	1.79	1.05	1.00	2.05	1.15
Lobatto IIIC	1.41	1.00	1.79	2.12	1.06	1.00	2.42	1.17

Table 1: Bounds on $\kappa(\mathcal{P}_{\gamma_*})$ from Corollary 1 and (16) for Gauss, Radau IIA, and Lobatto IIIC integration, with 2–5 stages. Each column within a given set of stages corresponds to either a real eigenvalue, $\lambda_1 = \eta$, or a conjugate pair of eigenvalues, e.g., $\lambda^{\pm}_{2,3} = \eta \pm \mathrm{i}\beta$, of A_0^{-1}.

Corollary 1 introduces a modified constant for preconditioning. To compare with a naive approach of $\delta = \gamma = \eta$ (i.e., preconditioning by ignoring the β^2 term in $\mathcal{Q}_\eta$ (12)), one can derive a worst-case condition number of $1 + \beta^2/\eta^2$ for SPD operators and $\approx (1+\beta^2/\eta^2)^{3/2}$ for skew symmetric operators, squaring and cubing the worst-case condition number derived for γ_* in Corollary 1, respectively. Numerical tests indicate using the modified constant γ_* as opposed to η is particularly important for hyperbolic-type problems, which tend to have dominant imaginary eigenvalues, even if the spatial discretization is not skew symmetric. Indeed, one example in Section 4.2 demonstrates an almost 6× reduction in iteration count achieved by using γ_* instead of η.

Remark 6 (Inexact preconditioning)**.** *In practice, fully converging* $(\gamma_* I - \widehat{\mathcal{L}})^{-1}$ *each iteration as a preconditioner is often not desirable due to the cost of performing a full linear solve. Here, we propose applying a Krylov method to* $\mathcal{Q}_\eta := (\eta^2+\beta^2)I - 2\eta\widehat{\mathcal{L}} + \widehat{\mathcal{L}}^2$ *by computing the operator's action (that is, not fully constructing it), and preconditioning each Krylov iteration with two applications of a sparse parallel preconditioner for* $(\gamma_* I - \widehat{\mathcal{L}})$*, approximating the action of* $(\gamma_* I - \widehat{\mathcal{L}})^{-2}$*.*

Analogous to standard block-preconditioning techniques, this approximate inverse approach is often (but not always) more efficient than computing a full inverse each iteration. However, it is important that the underlying preconditioner provides a good approximation. Fortunately, for difficult problems without highly effective preconditioners, it is straightforward to apply either multiple inner fixed-point iterations or an inner Krylov iteration (wrapped with a flexible outer Krylov method [37, 46]) to ensure robust (outer) iterations. In Section 4.2.3, a numerical example is shown where the proposed method diverges using a single inner fixed-point iteration as a preconditioner for $(\gamma_* I - \widehat{\mathcal{L}})$*, but three (or more) inner fixed-point iterations yields fast, stable convergence.*

3.4 Algorithm

We conclude this Section by providing a step-by-step description of the new method in Algorithm 1, which computes the solution $\mathbf{u}_{n+1}$ at time t_{n+1} using the update formula in (10). Bullets provide additional discussion on some nuances of the implementation.

Algorithm 1 Advance $\mathbf{u}_n$ to $\mathbf{u}_{n+1}$. Assume even s, and A_0^{-1} has $s/2$ complex-conjugate eigenvalue pairs $(\eta_i \pm \mathrm{i}\beta_i)_{i=1}^{s/2}$.

1: Evaluate $\mathbf{f} \equiv \mathbf{f}(\mathbf{u}_n, t_n)$ ▷ RHS of $Ns \times Ns$ system (3)

// Form RHS of linear system: $\mathbf{z} = [(\mathbf{b}_0^T A_0^{-1} \otimes I_N)\,\mathrm{adj}(\mathcal{M}_s)][(I_s \otimes M^{-1})\mathbf{f}]$

2: $\mathbf{z} \leftarrow 0$

3: **for** $i = 1 \to s$ **do**

4: $\quad \mathbf{z} \leftarrow \mathbf{z} + R_i(\widehat{\mathcal{L}})(M^{-1}\mathbf{f}_i)$

// Solve linear system: $P_s(\widehat{\mathcal{L}})\mathbf{y} = \prod_{i=1}^{s/2} \mathcal{Q}_{\eta_i}\mathbf{y} = \mathbf{z}$, where $\mathcal{Q}_{\eta_i} := (\eta_i I - \widehat{\mathcal{L}})^2 + \beta_i^2 I$

5: **for** $i = s/2 \to 1$ **do** ▷ Solve $\mathcal{Q}_{\eta_i}\mathbf{y} = \mathbf{z}$

6: $\quad \mathbf{z} \leftarrow M\mathbf{z}$ ▷ Scale $\mathcal{Q}_{\eta_i}\mathbf{y} = \mathbf{z}$ by M

7: $\quad \mathbf{y} \leftarrow \mathrm{Krylov}\big(M\mathcal{Q}_{\eta_i}, \mathbf{z}, \mathcal{P}M^{-1}\mathcal{P}\big)$ ▷ Inner preconditioner $\mathcal{P} \sim (\gamma_* M - \delta t \mathcal{L})^{-1}$

8: $\quad \mathbf{z} \leftarrow \mathbf{y}$ ▷ Set RHS for next i

// Get IRK solution at new time: $\mathbf{u}_{n+1} = \mathbf{u}_n + \delta t P_s(\widehat{\mathcal{L}})^{-1}\mathbf{z}$

9: $\mathbf{u}_{n+1} \leftarrow \mathbf{u}_n + \delta t \mathbf{y}$ ▷ IRK solution at t_{n+1}

- Line 4 of Algorithm 1: The RHS vector in the linear system of (10) can be expressed as $\mathbf{z} = [(\mathbf{b}_0^T A_0^{-1} \otimes I_N)\,\mathrm{adj}(\mathcal{M}_s)][(I_s \otimes M^{-1})\mathbf{f}] = \sum_{i=1}^{s} R_i(\widehat{\mathcal{L}})(M^{-1}\mathbf{f}_i)$. Here R_i is a polynomial of degree s that results from taking the inner product of $\mathbf{b}_0^T A_0^{-1}$ with the ith column of the matrix $\mathrm{adj}(A_0^{-1} - xI)$, where x is a scalar variable. The coefficients of R_i are precomputed with high precision (e.g., in Mathematica), and after forming the vector $M^{-1}\mathbf{f}_i$, the action of $R_i(\widehat{\mathcal{L}})$ is applied using a Horner-like scheme, which only requires computing the action of $\widehat{\mathcal{L}}$ s times. Recall that the (potentially dense) matrix $\widehat{\mathcal{L}} = \delta t M^{-1}\mathcal{L}$ is not formed, but its action is computed using that of M^{-1} and $\mathcal{L}$.
- Line 7 of Algorithm 1: $\mathbf{x} \leftarrow \mathrm{Krylov}(A, \mathbf{b}, B)$ means apply a Krylov method to solve $A\mathbf{x} = \mathbf{b}$, with left or right preconditioning $B \approx A^{-1}$. In Line 7, the inner

preconditioner $\mathcal{P}$ is some approximation to $(\gamma_* M - \delta t \mathcal{L})^{-1}$, such as one multigrid iteration (for example, see Remark 6), and the full preconditioner consists of applying $\mathcal{P} M^{-1} \mathcal{P}$. In each Krylov iteration, the operator $M \mathcal{Q}_{\eta_i}$ is not formed, but its action is computed using a Horner-like scheme.

4 Numerical results

Numerical results consider Gauss, RadauIIA, and LobattoIIIC IRK methods, as well as several SDIRK methods for comparison: 2-stage, 2nd-order L-stable SDIRK [25, Eq. 221] ($\gamma = (2-\sqrt{2})/2$), 2-stage, 3rd-order A-stable SDIRK [25, Eq. 223] ($\gamma = (3+\sqrt{3})/3$), 3-stage, 3rd-order L-stable SDIRK [25, Eq. 229], 3-stage, 4th-order A-stable SDIRK [18, Eq. (6.18)], and 5-stage, 4th-order L-stable SDIRK [18, Table 6.5].

For some problems, runtime comparisons are made between the current IRK algorithm and those proposed in [51] and [42]. The algorithms from [51] and [42] solve the stage equations (3) with an iterative solver, such as GMRES, for example, using a block preconditioner based on the Butcher matrix A_0. The block preconditioners are chosen to have a block triangular structure, such that they can be applied via forward/backward substitution. During the application of the preconditioners, exact inverses of the diagonal blocks are approximated with an inexpensive iterative method, such as a single multigrid cycle, for example. Of all the preconditioners proposed in Staff et al. [51], we show results for the one that uses a lower-triangular splitting of A_0, which we refer to as "GSL", since we find it has the smallest runtime. From Rana et al. [42] we compare with the LD preconditioner, which uses a preconditioner based on the LD component of an LDU factorization of A_0, which we refer to as "LD."

4.1 Finite-difference advection-diffusion

In this section, we consider a constant-coefficient advection-diffusion problem discretized in space with high-order finite-differences. An exact solution to this problem is used to demonstrate the high-order accuracy of the IRK methods, and the robustness of the algorithms developed in the previous section with respect to mesh resolution. Specifically, we solve the PDE

$$u_t + 0.85u_x + u_y = 0.3u_{xx} + 0.25u_{yy} + s(x,y,t), \quad (x,y,t) \in (-1,1)^2 \times (0,2], \tag{18}$$

on a periodic spatial domain. The source term $s(x,y,t)$ is chosen such that the solution of the PDE is $u(x,y,t) = \sin^4(\pi/2[x-1-0.85t]) \sin^4(\pi/2[y-1-t]) \exp(-[0.3+0.25]t)$.

We consider tests using IRK methods of orders three, four, seven, and eight. The 3rd- and 4th-order IRK methods are paired with 4th-order central-finite-differences in space, and the 7th- and 8th-order methods with 8th-order central-finite-differences in space. In all cases, a time-step of $\delta t = 2h$ is used, with h denoting the spatial mesh size, and results are run on four cores. Due to the diffusive, but non-SPD nature of the spatial discretization, we apply GMRES(30) preconditioned by a classical algebraic multigrid (AMG) method in the *hypre* library [15]. Specifically, we use classical interpolation (type 0), Falgout coarsening (type 6) with a strength tolerance $\theta_C = 0.25$, zero levels of aggressive coarsening, and L_1-Gauss–Seidel relaxation (type 8), with a relative stopping tolerance of 10^{-13}. A single iteration of AMG is applied to approximate $(\gamma_* I - \delta t \mathcal{L})^{-1}$.

In Figure 1, discretization errors are shown for different IRK methods, alongside the average number of AMG iterations needed per time step. The expected asymptotic convergence rates (black dashed lines in the left panel) are observed for all discretizations.[6]

[6] An exception here is A–SDIRK(4), which appears to be converging with a rate closer to three than four; however, further decreasing δt (not shown here) confirms 4th-order convergence is achieved eventually.

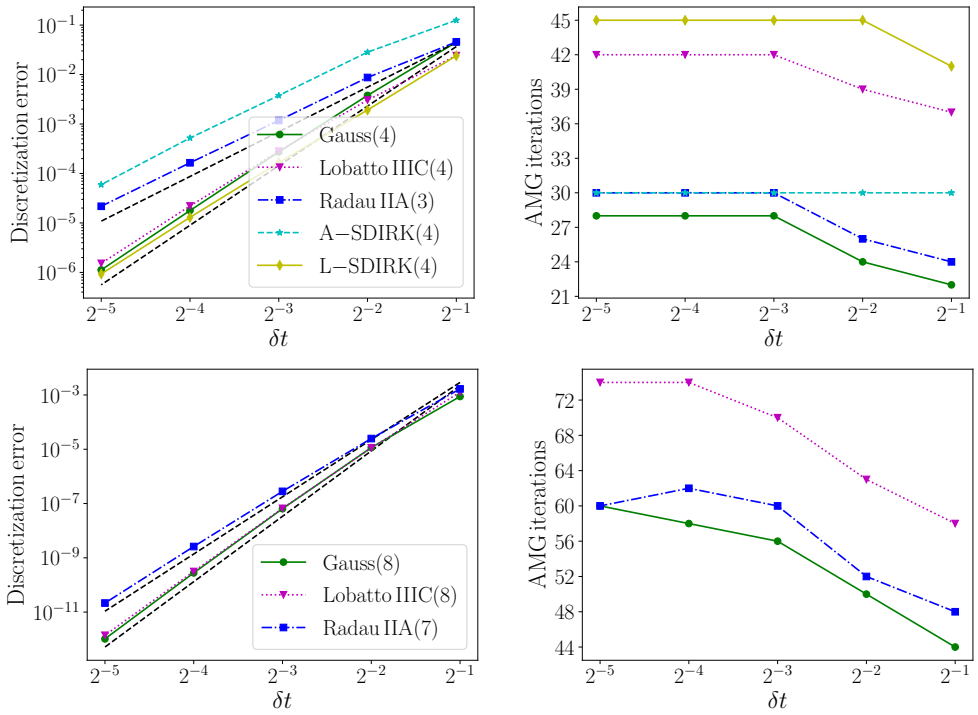


Figure 1: Finite-difference advection-diffusion problem (18). L_∞-discretization errors at $t = 2$ as a function of time-step δt are shown on the left for various discretizations of approximately 4th order (top) and 8th order (bottom). Black, dashed lines with slopes of three and four are shown (top), as are those with slopes of seven and eight (bottom). Plots on the right show the average number of AMG iterations per time step. For time-step size $\delta t = 2^{-\ell}$, the linear systems are of size $n_x \times n_y = 2^{\ell+2} \times 2^{\ell+2}$.

The preconditioner appears robust with respect to mesh and problem size, since the average number of AMG iterations per time step (which is a proxy for the number of GMRES iterations) remains roughly constant as the the mesh is refined. Of the fully implicit methods, the Gauss methods require the fewest AMG iterations, closely followed by Radau IIA methods, with the Lobatto IIIC methods requiring the most AMG iterations. This is consistent with the theoretical estimates in Table 1. Note that while Gauss and Radau IIA methods have very similar iteration counts, Gauss converges at one order faster, which can be seen in the left-hand panel of the figure.

Considering the lower-order methods in the top row of Figure 1, L–SDIRK(4) (see [18, Table 6.5]), a 5-stage, 4th-order, L-stable SDIRK method requires the most AMG iterations of all methods. A–SDIRK(4) (see [18, eq. (6.18)]), a 3-stage, 4th-order, A-stable SDIRK method, requires far fewer AMG iterations than L–SDIRK(4). However, A–SDIRK(4) yields a significantly larger discretization error than the other 4th-order schemes, and takes longer to reach its asymptotic convergence rate. Thus, in terms of solution accuracy as a function of computational work, the new preconditioner with 4th-order Gauss integration is the clear winner for this particular test problem, requiring roughly half the AMG iterations of the commonly used L-stable SDIRK4 scheme.

Table 2 shows the runtime of the block-preconditioning approaches of GSL [51] and LD [42] (see the introduction of Section 4) relative to the current approach that uses complex-conjugate preconditioning. For the GSL and *LD* solves, GMRES(30) is used as the solver with a relative stopping tolerance of 10^{-13}, and a single iteration of AMG

is used to approximate the inverses of the diagonal blocks in the block lower triangular preconditioners. These AMG methods use the same settings as those described above. In all cases, the runtime of the current IRK algorithm is smaller than those using block preconditioning, demonstrating the competitiveness of our approach with existing ones, and the advantages of using an optimized preconditioner.

	Runtime relative to current approach		
	Current	GSL [51]	*LD* [42]
Gauss(4)	1	1.24	1.21
Radau IIA(3)	1	1.44	1.16
Lobatto IIIC(4)	1	2.09	1.86
Gauss(8)	1	1.64	1.57
Radau IIA(7)	1	1.92	1.64
Lobatto IIIC(8)	1	2.97	2.22

Table 2: Finite-difference advection-diffusion problem (18). Runtime of block-preconditioning IRK algorithms GSL [51] and *LD* [42] relative to the runtime of the current algorithm that uses complex-conjugate preconditioning. Each relative runtime measurement has been calculated as the mean of the relative runtimes for solving each of the problem sizes shown in Figure 1.

4.2 DG in space advection-diffusion

Here we consider a more difficult advection-diffusion problem, discretized using high-order DG finite elements in space (independent of the time discretization; i.e., no relation to Remark 1). We demonstrate the effectiveness of the new preconditioning and "optimal γ_*" on more complex flows (Section 4.2.2), examine order reduction in DIRK and IRK methods (Section 4.2.1), study the use of multiple "inner" preconditioning iterations to approximate $(\gamma_* M - \delta t \mathcal{L})^{-1}$ (or even inner Krylov acceleration; Section 4.2.3), and finally make a comparison with other state-of-the-art IRK solvers from [41, 42, 51] (Section 4.2.4).

The governing equations in spatial domain $\Omega = [0,1] \times [0,1]$ are given by

$$u_t + \nabla \cdot (\boldsymbol{\beta} u - \varepsilon \nabla u) = f \tag{19}$$

where $\boldsymbol{\beta}(x,y) := (\cos(4\pi y), \sin(2\pi x))^T$ is the prescribed velocity field and ε the diffusion coefficient. Dirichlet boundary conditions are weakly enforced on $\partial\Omega$, and (19) is discretized with an upwind DG method [14], where diffusion terms are treated with the symmetric interior penalty method [3, 4]. The resulting finite element problem is to find $u_h \in V_h$ such that, for all $v_h \in V_h$,

$$\begin{aligned}\int_\Omega \partial_t(u_h) v_h \, dx - \int_\Omega u_h \boldsymbol{\beta} \cdot \nabla_h v_h \, dx + \int_\Gamma \widehat{u_h} \boldsymbol{\beta} \cdot [\![v_h]\!] \, ds + \int_\Omega \nabla_h u_h \cdot \nabla_h v_h \, dx \\ - \int_\Gamma \{\nabla_h u_h\} \cdot [\![v_h]\!] \, ds - \int_\Gamma \{\nabla_h v_h\} \cdot [\![u_h]\!] \, ds + \int_\Gamma \sigma [\![u_h]\!] \cdot [\![v_h]\!] \, ds = \int_\Omega f v_h \, dx,\end{aligned}$$

where V_h is the DG finite element space consisting of piecewise polynomials of degree p defined on elements of the computational mesh $\mathcal{T}$ of the spatial domain Ω. No continuity is enforced between mesh elements. Here, ∇_h is the broken gradient, Γ denotes the skeleton of the mesh, and $\{\cdot\}$ and $[\![\cdot]\!]$ denote the average and jump of a function across a mesh interface. $\widehat{u_h}$ is used to denote the upwind numerical flux. The parameter σ

is the *interior penalty parameter*, which must be chosen sufficiently large to obtain a stable discretization [4]. In particular, we choose $\sigma \sim p^2/h$; see also [48] for an explicit expression for this parameter. This discretization has been implemented in the MFEM finite element framework [2], and uses AMG preconditioning with approximate ideal restriction (AIR) [31, 32] (after first scaling by the inverse of the block-diagonal mass matrix).

4.2.1 Order reduction and wall-clock-time

We begin by considering a manufactured solution. We set diffusion coefficient $\epsilon = 10^{-4}$ and choose initial conditions, Dirichlet boundary conditions, and time-dependent forcing function such that (19) with constant coefficient advection, $\boldsymbol{\beta} = [1, 1]$ satisfies the exact space-time solution $u_* = \sin(2\pi(x-t))\sin(2\pi(y-t))$. This results in weakly imposing time-dependent Dirichlet boundary conditions, a constraint known to cause order reduction in DIRK methods due to low stage order [45]. 4th-order elements are used on a mesh with $h \approx 0.0078$, which results in finite element approximation of the initial condition with ℓ^2-error $3{\cdot}10^{-12}$) (i.e., this is roughly the expected limit of accuracy that can be obtained after time integration).

We first consider the order of convergence for L-stable SDIRK methods compared with RadauIIA and Gauss IRK methods. Error is measured at time $t = 2$ against the exact solution in the ℓ^2-norm and a broken $\mathcal{H}^1$-norm, and results are shown in Figure 2. Order reduction is most pronounced for 4th-order L-stable SDIRK, achieving only second order in the broken $\mathcal{H}^1$-norm and order 2.7 in the ℓ^2-norm as $\delta t \to 0$. 3rd-order L-stable SDIRK achieves the expected order in ℓ^2-norm, but only 2nd-order in broken $\mathcal{H}^1$-norm. In contrast, 3rd order Radau achieves just under 3rd order in the $\mathcal{H}^1$-norm and 3rd order in ℓ^2, and 4th-order Gauss achieves 4th order in both norms for sufficiently small δt. Most higher-order Radau and Gauss methods observe some order reduction in both norms, achieving accuracy somewhere between the stage-order [18, Section IV.5] (s) and full order ($2s-1$ and $2s$, respectively), but still obtain significantly smaller error than the lower-order schemes.

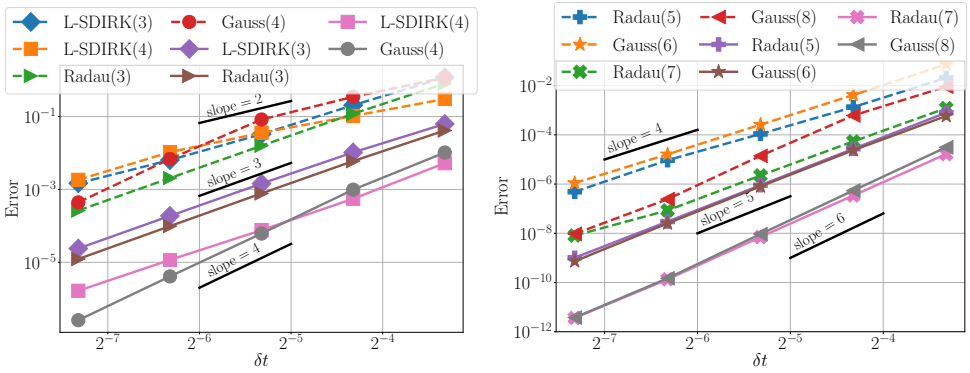


Figure 2: ℓ^2-error (solid lines) and broken $\mathcal{H}^1$-error (dashed lines) at time $t = 2$.

We now compare accuracy as a function of wallclock time of standard A-stable and L-stable SDIRK methods with IRK methods using the preconditioners developed here. *For a given wall-clock time, the IRK methods of a given order yield smaller error than equivalent-order SDIRK methods in all cases considered.* Moreover, for a fixed wallclock time, the high-order IRK methods can achieve as much as two orders of magnitude reduction in error compared with lower-order schemes.

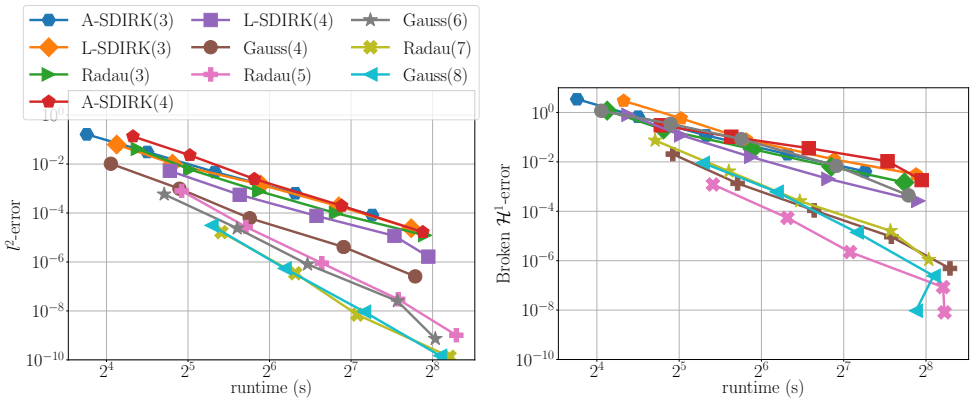
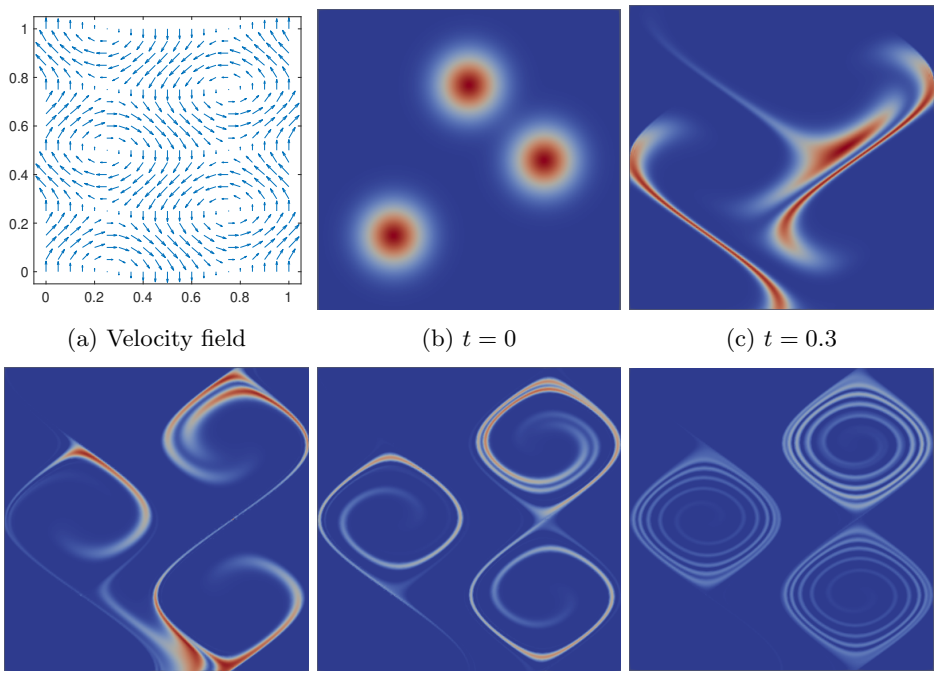


Figure 3: ℓ^2-error (right) and broken $\mathcal{H}^1$-error (left) at time $t = 2$ as a function of wallclock time in seconds.

4.2.2 A hyperbolic example, and preconditioning with η vs. γ_*

The DG method is particularly well-suited for advection-dominated problems. In the following subsections we vary ε from 0 (purely advective) to 0.01. The velocity field, initial condition, and numerical solution for $\varepsilon = 10^{-6}$ are shown in Figure 4.



(a) Velocity field (b) $t = 0$ (c) $t = 0.3$

(d) $t = 0.8$ (e) $t = 2.0$ (f) $t = 7.0$

Figure 4: DG advection-diffusion problem with velocity field shown in subplot (a) and the solution plotted for various time points from $t = 0$ to $t = 7.0$ in subplots (b-f). Heatmap indicates solution in the 2d domain, with blue $\mapsto 0$ and red $\mapsto 1$.

First, we demonstrate the effectiveness of using γ_* (Section 3.3) instead of η in the preconditioner, as well as the robustness of the proposed method on a fully hyperbolic

problem, where most papers have only discussed parabolic PDEs. Thus, we set the diffusion coefficient $\varepsilon = 0$ and apply AIR as a preconditioner for individual systems $(\gamma M - \delta t\mathcal{L})$.

AIR was originally designed for upwind DG discretizations of advection and is well-suited for this problem. We use the *hypre* implementation, with distance 1.5 restriction with strength tolerance $\theta_R = 0.01$, one-point interpolation (type 100), Falgout coarsening (type 6) with strength tolerance $\theta_C = 0.1$, no pre-relaxation, and forward Gauss Seidel post-relaxation (type 3), first on F-points followed by a second sweep on all points. The domain is discretized using 4th-order finite elements on a structured mesh, and the time step for each integration scheme is chosen such that the spatial and temporal orders of accuracy match; for example, for 8th-order integration we choose $\delta t = \sqrt{h}$, for mesh spacing h, so that $\delta t^8 = h^4$. All linear systems are solved to a relative tolerance of 10^{-12}. There are a total of 1,638,400 spatial degrees-of-freedom (DOFs), and the simulations are run on 288 cores on the Quartz machine at Lawrence Livermore National Lab, resulting in ∼5600 DOFs/processor.

Table 3 shows the average number of AIR iterations to solve for each pair of stages of an IRK method using η and γ_* as the preconditioning constants. Iteration counts are shown for Gauss, Radau IIA, and Lobatto IIIC integration, with 2–5 stages, and the (factor of) reduction in iteration count achieved using γ_* vs. η is also shown. For 5-stage Lobatto IIIC integration, γ_* yields almost a 6× reduction in total inner AIR iterations to solve for the "hard" stage ($\beta > \eta$), while in no cases is there an increase in iteration count when using γ_*.

Gauss								
Stages/Order	2/4	3/6		4/8		5/10		
Iterations(η)	17	6	30	11	47	8	16	70
Iterations(γ_*)	11	6	15	10	19	8	13	23
Speedup	1.5	1.0	2.0	1.1	2.5	1.0	1.2	3.0
Radau IIA								
Stages/Order	2/3	3/5		4/7		5/9		
Iterations(η)	12	5	39	11	64	8	16	97
Iterations(γ_*)	12	5	18	9	21	8	12	25
Speedup	1.0	1.0	2.2	1.2	3.0	1.0	1.3	3.9
Lobatto IIIC								
Stages/Order	2/2	3/4		4/6		5/8		
Iterations(η)	8	3	67	11	113	7	17	175
Iterations(γ_*)	8	3	22	9	26	7	12	30
Speedup	1.0	1.0	3.0	1.2	4.3	1.0	1.4	5.8

Table 3: Average AIR iterations to solve for each stage in an implicit Runge-Kutta method using preconditioners $(\eta M - \delta t\mathcal{L})^{-2}$ and $(\gamma_* M - \delta t\mathcal{L})^{-2}$, with γ_* defined in (15). The ratio of iterations(η)/iterations(γ_*) is shown in the "Speedup" rows.

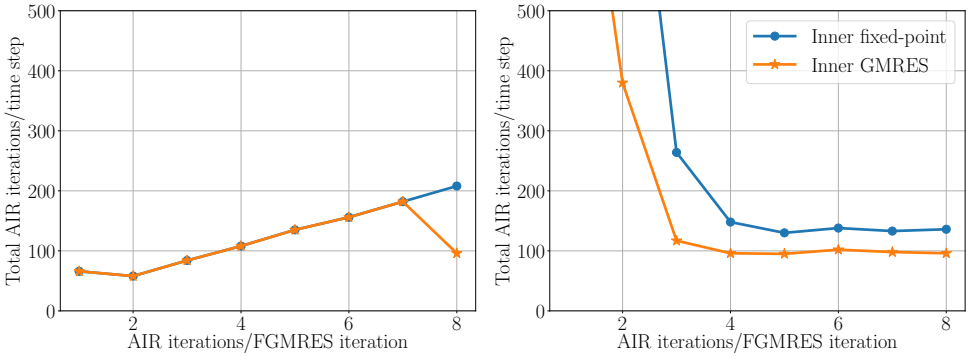
4.2.3 Diffusive problems and inner Krylov

In [31], AIR was shown to be effective on some DG advection-diffusion problems, and classical AMG is known to be effective on diffusion-dominated problems. However, the region of comparable levels of advection and diffusion remains the most difficult from

a multigrid perspective. We use this to demonstrate how methods developed here require a "good" preconditioner for a backward Euler time step, $(\gamma M - \delta t \mathcal{L})^{-1}$, in order to converge on more general IRK methods. Fortunately, ensuring a preconditioner is sufficiently good can be resolved by appealing to standard block preconditioning techniques, where an inner iteration is used that applies multiple AIR iterations as a single preconditioner.

Here we consider an analogous problem to above, but set the diffusion coefficient to $\varepsilon = 0.01$. We use a mesh with spacing $h \approx 0.001$, 2nd-order DG finite elements, a time step of $\delta t = 0.1$, and three-stage 6th-order Gauss integration. Altogether, this yields equal orders of accuracy, with time and space error $\sim 10^{-6}$. FGMRES [46] is used for the outer iteration, which allows for GMRES to be applied in an inner iteration as a preconditioner for $(\gamma_* M - \delta t \mathcal{L})$. Figure 5 plots the total number of AIR iterations per time step as a function of the number of AIR iterations applied for each application of the preconditioner, using an inner GMRES or an inner fixed-point (Richardson) iteration. An advection-dominated problem with $\varepsilon = 10^{-6}$ is also shown for comparison.

Recall we have three stages, one of which is a single linear system corresponding to a real eigenvalue, and the other corresponding to a pair of complex conjugate eigenvalues, which we precondition as in Section 3. The latter ends up being the more difficult problem to solve – for $\varepsilon = 0.01$ (Figure 5b), the outer FGMRES iteration for the complex conjugate quadratic does not converge in 1000 iterations when using one AIR iteration as a preconditioner. If two AIR iterations with GMRES are used as a preconditioner, the FGMRES iteration converges in approximately 130 iterations, each of which requires two applications of GMRES preconditioned with two AIR iterations, yielding just over 500 total AIR iterations to converge. Further increasing the number of AIR iterations per preconditioning yields nice convergence using inner fixed-point or GMRES, with 150 and 112 total AIR iterations per time step, respectively. In contrast, Figure 5a shows that additional AIR iterations for the advection-dominated case are generally detrimental to overall computational cost (although the outer iteration converges slightly faster, it does not make up for the additional linear solves/iteration).



(a) $\varepsilon = 10^{-6}$. (b) $\varepsilon = 0.01$.

Figure 5: Total AIR iterations per time step as a function of the number of inner AIR iterations applied during each application of the preconditioner, for diffusion coefficient ε.

4.2.4 Comparison of preconditioners

We conclude this section by performing a detailed comparison with other IRK solvers. We consider the three-stage and five-stage Gauss, Radau, and Lobatto IRK integration schemes, with the velocity field shown in Figure 4, and diffusion coefficients $\epsilon \in$

$\{0, 10^{-4}, 0.01\}$. For these tests, we (somewhat arbitrarily) choose $\delta t = 0.1$, $h \approx 0.002$, and second-order finite elements, resulting in 2,359,296 DOFs. All simulations are run on 256 cores on the Quartz supercomputer at Lawrence Livermore National Lab, with 9,360 DOFs/core. We compare the IRK methods developed here (denoted IRK), with the GSL [51] and LD [42] block preconditioners introduced earier, as well as a block ILU preconditioner as in [41]. As in Section 4.2.3, for the case of $\epsilon = 0.01$, we use three inner AIR iterations for the IRK solver. We only use one inner AIR iteration for the GSL and LD block preconditioners, as numerical tests indicated that they neither need nor benefit from inner iterations. Table 4 presents the wallclock time and average iterations per time step to take five time steps with each of the methods. Iterations are normalized as one preconditioner applied to each or all stages, to account for the difference between ILU applied to the full IRK operator and IRK/GSL/LD applied to individual stages, as well as the use of three vs. one inner iteration.

We highlight several observations:

- In general, the method proposed here is faster than ILU in all cases but one (in which the times are very close), while offering as much as a $7\times$ speedup in one example (Radau(9) $\epsilon = 0.01$). In general, ILU will be more competitive on a smaller number of MPI processes and less competitive on more MPI processes, due to its general degradation in parallel.
- The method proposed here is particularly effective on higher-order integrators (in this case, the 5-stage examples) and advection-dominated problems, in many cases offering a $10-15\times$ speedup over the block preconditioning from [51]. The very new method from [42] is more competitive, typically only $2-5\times$ slower on the more advective problems, while performing better than the method developed here for all cases of $\epsilon = 0.01$. The better performance for $\epsilon = 0.01$ is because the method from [42] appears more robust to not having an accurate inner inverse, whereas IRK requires three inner AIR iterations for good performance on $\epsilon = 0.01$ (thus, costing $3\times$ as much for one true iteration). However, this property is a blessing and a curse: numerical tests also indicated that additional inner iterations did *not* improve convergence of [42, 51] on advective problems, thus limiting their performance to that of the outer block preconditioner, which currently lacks robust theoretical support. In contrast, the IRK solvers developed here are *guaranteed* to be robust, as long as one provides a reasonably accurate inner inverse.

4.3 High-order matrix-free discretization of diffusion

In this example, we illustrate the use of high-order IRK methods coupled with high-order finite element spatial discretizations. It is well-known that matrix assembly becomes prohibitively expensive for high-order finite elements. Naive algorithms typically require $\mathcal{O}(p^{3d})$ operations to assemble the resulting system matrix, where p is the polynomial degree and d is the spatial dimension. Techniques such as sum factorization can reduce this cost on tensor-product elements to $\mathcal{O}(p^{2d+1})$, however this cost can still be prohibitive for large values of p [34]. On the other hand, matrix-free operator evaluation on tensor-product meshes can be performed in $\mathcal{O}(p^{d+1})$ operations [39], motivating the development of solvers and preconditioners that can be constructed and applied without access to the assembled system matrix [26].

We consider a high-order finite element discretization of the linear heat equation on spatial domain Ω,

$$\int_\Omega \partial_t(u_h) v_h \, dx + \int_\Omega \nabla u_h \cdot \nabla v_h \, dx = \int_\Omega f v_h \, dx,$$

where $u_h, v_h \in V_h$, and V_h denotes the degree-p H^1-conforming finite element space

3 stages	**Gauss(6)**	GSL [51]	LD [42]	ILU [40]	IRK
	$\epsilon = 0$	7.76 (66)	4.46 (27)	3.36 (45)	**3.24** (24)
	$\epsilon = 10^{-4}$	16.4 (67)	8.85 (28)	**5.76** (73)	6.02 (20)
	$\epsilon = 0.01$	10.5 (51)	**7.48** (28)	85 (491)	14.7 (44)
	Radau(5)	GSL [51]	LD [42]	ILU [40]	IRK
	$\epsilon = 0$	11.9 (101)	5.35 (32)	5.20 (68)	**3.81** (27)
	$\epsilon = 10^{-4}$	25.1 (101)	10.62 (34)	7.11 (87)	**6.42** (22)
	$\epsilon = 0.01$	17.6 (85)	**10.32** (41)	164 (716)	31.0 (94)
	Lobatto(4)	GSL [51]	LD [42]	ILU [40]	IRK
	$\epsilon = 0$	19.3 (168)	7.86 (47.2)	10.4 (118.6)	**4.89** (36)
	$\epsilon = 10^{-4}$	39.2 (167)	15.2 (49.2)	9.43 (109.2)	**7.91** (25)
	$\epsilon = 0.01$	26.6 (133)	**14.3** (57)	275 (955.8)	38.9 (115)
5 stages	**Gauss(10)**	GSL [51]	LD [42]	ILU [40]	IRK
	$\epsilon = 0$	24.9 (151)	11.5 (43)	12.7 (78)	**4.40** (21)
	$\epsilon = 10^{-4}$	55.9 (153)	25.8 (52)	11.6 (73)	**10.1** (22)
	$\epsilon = 0.01$	28.4 (91)	**17.9** (41)	134 (449)	22.6 (42)
	Lobatto(8)	GSL [51]	LD [42]	ILU [40]	IRK
	$\epsilon = 0$	70.6 (435)	21.7 (83)	113 (403)	**5.41** (44)
	$\epsilon = 10^{-4}$	161 (444)	47.7 (99)	15.9 (94)	**11.2** (25)
	$\epsilon = 0.01$	73.0 (242)	**34.3** (84)	281 (703)	52.3 (100)
	Radau(9)	GSL [51]	LD [42]	ILU [40]	IRK
	$\epsilon = 0$	40.2 (241)	14.6 (53)	25.0 (140)	**4.72** (23)
	$\epsilon = 10^{-4}$	89.5 (243)	31.3 (64)	13.8 (84)	**10.3** (23)
	$\epsilon = 0.01$	47.8 (153)	**25.6** (61)	220 (609)	33.4 (64)

Table 4: Comparison of wallclock time (left) to take 5 time steps, and average iteration count per time step (normalized as one preconditioner applied to all stages, right in $(\cdot)$) for various IRK preconditioners. Fastest wallclock times for each fixed ϵ are shown in bold.

defined on a mesh $\mathcal{T}$ consisting of tensor-product elements (i.e. quadrilaterals or hexahedra). The matrix-free action of the corresponding operator is computed in $\mathcal{O}(p^{d+1})$ operations using the *partial assembly* features of the MFEM finite element library [2]. In order to precondition the resulting system, we make use of a low-order refined preconditioner, whereby the high-order system is preconditioned using a spectrally equivalent low-order finite element discretization computed on a refined mesh [11]. The low-order refined discretization can be assembled in $\mathcal{O}(p^d)$ time, thereby avoiding the prohibitive costs of high-order matrix assembly. We make use of the uniform preconditioners for the low-order refined problem based on subspace corrections, developed in [40].

For this test case, take the spatial domain to be $\Omega = [0,1] \times [0,1]$, with periodic boundary conditions. We choose the forcing term

$$f(x,y,t) = \sin(2\pi x)\cos(2\pi y)\left(\cos(t) + 8\pi^2(2+\sin(t))\right),$$

which corresponds to the exact solution

$$u(x,y,t) = \sin(2\pi x)\cos(2\pi y)(2+\sin(t)).$$

We begin with a very coarse 3×3 mesh, and integrate in time until $t = 0.1$ using the Gauss and Radau IIA methods of orders 2 through 10. For each test case, the finite

element polynomial degree is set to $k-1$, where k is the order of accuracy of the time integration method, resulting in kth order convergence in both space and time. The mesh and time step are refined by factors of two to confirm the high-order convergence in space and time of the method. The relative L^2 error, obtained by comparing against the exact solution, is shown in Figure 6.

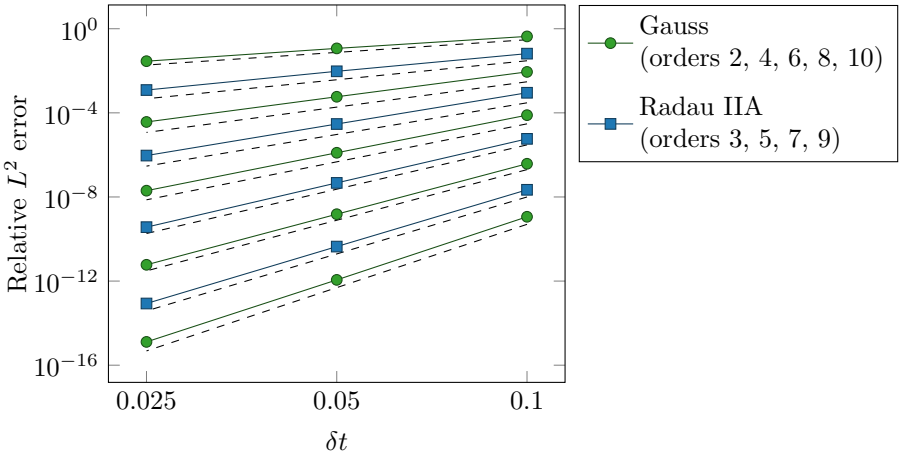


Figure 6: High-order convergence in space and time for the matrix-free diffusion problem. Gauss and Radau IIA methods of orders 2 through 10 are used. The dashed lines indicate the expected rates of convergence for each method.

We also use this test case to study the effect of inner iterations on the convergence of the iteration solver. As discussed in Remark 6, it is important that the underlying preconditioner provides a good approximation of the inverse of the operator. For that reason, we consider the use of an inner Krylov solver at every iteration. Since this corresponds to using a variable preconditioner at each iteration, a flexible Krylov method may have to be used for the outer iteration, although in practice good convergence is often still observed using the standard CG method [37]. In particular, we compare the total number of preconditioner applications required to converge the outer iteration to a relative tolerance of 10^{-10}, both with and without an inner Krylov solver. For the inner Krylov solver, we use a CG iteration with the same relative tolerance as the outer iteration in order to give a good approximation to the inverse of the operator. The iteration counts are displayed in Figure 7. We note that for the fully implicit IRK methods, using an inner Krylov solver can reduce the total number of preconditioner applications by about a factor of 1.5, although this depends on the type of method and order of accuracy. As expected, the use of inner iterations does not reduce the total number of preconditioner applications for DIRK methods. In addition, for this test case, the total number of preconditioner applications required for the second and fourth order Gauss IRK methods is between 1.3 and 2 times smaller than those required for the corresponding equal-order DIRK methods.

5 Conclusions

This paper introduces a theoretical and algorithmic framework for the fast, parallel solution of fully implicit Runge-Kutta and DG discretizations in time for linear numerical PDEs. Theory is developed to guarantee the preconditioned condition number is bounded by a small, order-one constant under fairly general assumptions on the spatial discretization that yield stable time integration. Numerical results demonstrate the

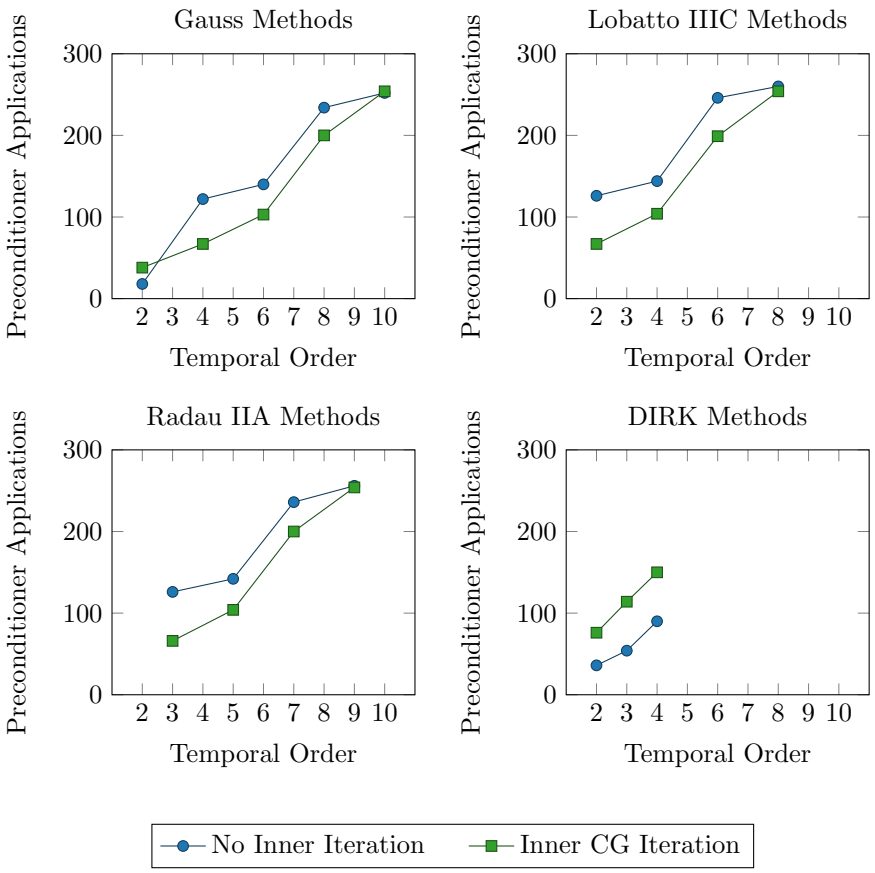


Figure 7: Comparison of total number of preconditioner applications with and without inner iterations. Both the outer iteration and the inner CG iteration are converged to a relative tolerance of 10^{-10}.

new method on various high-order finite-difference and finite-element discretizations of linear parabolic and hyperbolic problems, demonstrating fast, scalable solution of up to 10th order accuracy. In several cases, the new method can achieve 4th-order accuracy using Gauss integration with roughly half the number of preconditioner applications as required using standard SDIRK techniques, and in many cases the method outperforms previous state of the art. Ongoing work involves addressing fully nonlinear problems and algebraic constraints, in particular, without assuming that the linear system (3) can be expressed in Kronecker-product form (thus allowing for a true Newton or better Newton-like method compared with the commonly used/analyzed simplified Newton approach).

A Proofs

Theorem 1. *Suppose Assumptions 1 and 2 hold, that is, $\eta > 0$ and $W(\widehat{\mathcal{L}}) \leq 0$, and suppose $\widehat{\mathcal{L}}$ is real-valued. Let $\mathcal{P}_{\delta,\gamma}$ denote the preconditioned operator as in (13), where $[(\eta I - \widehat{\mathcal{L}})^2 + \beta^2 I]$ is preconditioned with $(\delta I - \widehat{\mathcal{L}})^{-1}(\gamma I - \widehat{\mathcal{L}})^{-1}$, for $\delta, \gamma \in (0, \infty)$. Let*

$\kappa(\mathcal{P}_{\delta,\gamma})$ denote the two-norm condition number of $\mathcal{P}_{\delta,\gamma}$, and define γ_* by

$$\gamma_* := \frac{\eta^2 + \beta^2}{\delta}. \tag{20}$$

Then

$$\kappa(\mathcal{P}_{\delta,\gamma_*}) \leq \frac{1}{2\eta}\left(\delta + \frac{\eta^2+\beta^2}{\delta}\right). \tag{21}$$

Moreover, (i) bound (21) is tight when considered over all $\widehat{\mathcal{L}}$ that satisfy Assumption 2 in the sense that $\exists\ \widehat{\mathcal{L}}$ such that (21) holds with equality, and (ii) $\gamma = \gamma_*$ is optimal in the sense that, without further assumptions on $\widehat{\mathcal{L}}$, γ_* minimizes a tight upper bound on $\kappa(\mathcal{P}_{\delta,\gamma})$, with $\gamma_* = \operatorname{argmin}_{\gamma\in(0,\infty)} \max_{\widehat{\mathcal{L}}} \kappa(\mathcal{P}_{\delta,\gamma})$.

Proof. We prove this theorem via a sequence of three lemmas, successively proving the upper bound in (21), followed by the tightness of this bound, followed by the optimality of γ_*.

Lemma 2 (Upper bound)**.** *Under the assumptions of Theorem 1, $\kappa(\mathcal{P}_{\delta,\gamma_*}) \leq \frac{1}{2\eta}\left(\delta + \frac{\eta^2+\beta^2}{\delta}\right)$ (21).*

Proof. The square of the condition number of $\mathcal{P}_{\delta,\gamma}$ is given by

$$\kappa^2(\mathcal{P}_{\delta,\gamma}) = \|\mathcal{P}_{\delta,\gamma}\|^2\|\mathcal{P}_{\delta,\gamma}^{-1}\|^2 = \max_{\boldsymbol{v}\neq 0}\frac{\|\mathcal{P}_{\delta,\gamma}\boldsymbol{v}\|^2}{\|\boldsymbol{v}\|^2}\frac{1}{\min\limits_{\boldsymbol{v}\neq 0}\dfrac{\|\mathcal{P}_{\delta,\gamma}\boldsymbol{v}\|^2}{\|\boldsymbol{v}\|^2}}, \tag{22}$$

where for real-valued $\widehat{\mathcal{L}}$, the max and min can be obtained by restricting ourselves to real-valued $\boldsymbol{v}$. The key step in establishing (21) is bounding $\|\mathcal{P}_{\delta,\gamma}\|^2$ and $\|\mathcal{P}_{\delta,\gamma}^{-1}\|^2$ from above by bounding $\|\mathcal{P}_{\delta,\gamma}\boldsymbol{v}\|^2/\|\boldsymbol{v}\|^2$ from above and below, respectively. Consider the form of the preconditioned operator $\mathcal{P}_{\delta,\gamma}$ in (13) and make the substitution $\boldsymbol{v} \mapsto (\gamma I - \widehat{\mathcal{L}})(\delta I - \widehat{\mathcal{L}})\boldsymbol{w}$. Using the fact that rational functions of $\mathcal{L}$ commute, $\|\mathcal{P}_{\delta,\gamma}\boldsymbol{v}\|^2$ can be expanded for real-valued $\boldsymbol{v}$ (and, thus, real-valued $\boldsymbol{w}$) as

$$\begin{aligned}
\|\mathcal{P}_{\delta,\gamma}\boldsymbol{v}\|^2 &= \left\|[(\eta I - \widehat{\mathcal{L}})^2 + \beta^2]\boldsymbol{w}\right\|^2, \\
&= \left\|[(\eta^2+\beta^2)\boldsymbol{w} - 2\eta\,\widehat{\mathcal{L}}\,\boldsymbol{w} + \widehat{\mathcal{L}}^2\,\boldsymbol{w}\right\|^2 \\
&= \left\|(\eta^2+\beta^2)\boldsymbol{w} + \widehat{\mathcal{L}}^2\,\boldsymbol{w}\right\|^2 - 4\eta(\eta^2+\beta^2)\langle\widehat{\mathcal{L}}\,\boldsymbol{w}, \boldsymbol{w}\rangle \\
&\qquad - 4\eta\langle\widehat{\mathcal{L}}(\widehat{\mathcal{L}}\,\boldsymbol{w}), \widehat{\mathcal{L}}\,\boldsymbol{w}\rangle + 4\eta^2\left\|\,\widehat{\mathcal{L}}\,\boldsymbol{w}\right\|^2.
\end{aligned} \tag{23}$$

Similarly, expanding $\|\boldsymbol{v}\|^2$ yields

$$\begin{aligned}
\|\boldsymbol{v}\|^2 &= \left\|(\gamma I - \widehat{\mathcal{L}})(\delta I - \widehat{\mathcal{L}})\boldsymbol{w}\right\|^2, \\
&= \left\|\,\delta\gamma\boldsymbol{w} - (\delta+\gamma)\,\widehat{\mathcal{L}}\,\boldsymbol{w} + \widehat{\mathcal{L}}^2\,\boldsymbol{w}\right\|^2, \\
&= \left\|\,\delta\gamma\boldsymbol{w} + \widehat{\mathcal{L}}^2\,\boldsymbol{w}\right\|^2 - 2\delta\gamma(\delta+\gamma)\langle\widehat{\mathcal{L}}\,\boldsymbol{w}, \boldsymbol{w}\rangle \\
&\qquad - 2(\delta+\gamma)\langle\widehat{\mathcal{L}}(\widehat{\mathcal{L}}\,\boldsymbol{w}), \widehat{\mathcal{L}}\,\boldsymbol{w}\rangle + (\delta+\gamma)^2\left\|\,\widehat{\mathcal{L}}\,\boldsymbol{w}\right\|^2.
\end{aligned} \tag{24}$$

Thus, the key ratio in (22) takes the form

$$\frac{\|\mathcal{P}_{\delta,\gamma}\boldsymbol{v}\|^2}{\|\boldsymbol{v}\|^2} = \frac{c_0(\boldsymbol{w})f_0(\boldsymbol{w}) + c_1 f_1(\boldsymbol{w}) + c_2 f_2(\boldsymbol{w}) + c_3 f_3(\boldsymbol{w})}{f_0(\boldsymbol{w}) + f_1(\boldsymbol{w}) + f_2(\boldsymbol{w}) + f_3(\boldsymbol{w})}, \tag{25}$$

where for $\delta, \gamma > 0$, we have defined the functions and constants

$$\begin{aligned}
f_0 &:= \left\| \delta\gamma \boldsymbol{w} + \widehat{\mathcal{L}}^2 \boldsymbol{w} \right\|^2 \geq 0, & c_0 &:= \frac{\left\|(\eta^2+\beta^2)\boldsymbol{w} + \widehat{\mathcal{L}}^2 \boldsymbol{w}\right\|^2}{\left\| \delta\gamma \boldsymbol{w} + \widehat{\mathcal{L}}^2 \boldsymbol{w} \right\|^2} \geq 0, \\
f_1 &:= -2\delta\gamma(\delta+\gamma)\langle \widehat{\mathcal{L}} \boldsymbol{w}, \boldsymbol{w}\rangle \geq 0, & c_1 &:= \frac{\eta^2+\beta^2}{\delta\gamma}\frac{2\eta}{\delta+\gamma} > 0, \\
f_2 &:= -2(\delta+\gamma)\langle \widehat{\mathcal{L}}(\widehat{\mathcal{L}} \boldsymbol{w}), \widehat{\mathcal{L}} \boldsymbol{w}\rangle \geq 0, & c_2 &:= \frac{2\eta}{\delta+\gamma} > 0, \\
f_3 &:= (\delta+\gamma)^2 \left\| \widehat{\mathcal{L}} \boldsymbol{w} \right\|^2 \geq 0, & c_3 &:= \left(\frac{2\eta}{\delta+\gamma}\right)^2 > 0.
\end{aligned} \tag{26}$$

Note that functions f_1 and f_2 are non-negative by assumption of $W(\widehat{\mathcal{L}}) \leq 0$, while for all $\boldsymbol{w} \neq \boldsymbol{0}$, it must hold that either $c_0 f_0 > 0$ or $c_3 f_3 > 0$ (or both, because $c_3 f_3 = 0$ i.f.f. $\widehat{\mathcal{L}} \boldsymbol{w} = \boldsymbol{0}$, which implies $c_0 f_0 > 0$ for $\boldsymbol{w} \neq \boldsymbol{0}$).

Since all of the addends in the numerator and denominator of (25) are non-negative, and at least one addend in each is positive, (25) can be bounded as

$$\min\{c_0, c_1, c_2, c_3\} =: c_{\min} \leq \frac{\|\mathcal{P}_{\delta,\gamma}\boldsymbol{v}\|^2}{\|\boldsymbol{v}\|^2} \leq c_{\max} := \max\{c_0, c_1, c_2, c_3\}.$$

Applying these bounds to the norms in (22) yields

$$\|\mathcal{P}_{\delta,\gamma}\| \leq \sqrt{c_{\max}}, \quad \|\mathcal{P}_{\delta,\gamma}^{-1}\| \leq \frac{1}{\sqrt{c_{\min}}}. \tag{27}$$

Bounding c_0 for general γ, and hence $c_{\min}$ and $c_{\max}$, is difficult because the sign of $\langle \widehat{\mathcal{L}}^2 \boldsymbol{w}, \boldsymbol{w}\rangle$ (which appears in expanding $\left\| \delta\gamma \boldsymbol{w} + \widehat{\mathcal{L}}^2 \boldsymbol{w} \right\|^2$) is not known for general $\widehat{\mathcal{L}}$, noting that the sign of $W(\widehat{\mathcal{L}})$ does not determine that of $W(\widehat{\mathcal{L}}^2)$. However, observe from (26) that the judicious choice of $\gamma = \gamma_* := (\eta^2+\beta^2)/\delta$ yields $c_0(\boldsymbol{w}) = 1$. Moreover, in the final part of this proof we demonstrate that $\gamma = \gamma_*$ is optimal, and, as such, moving forward we only consider the case $\gamma = \gamma_*$.

Letting $\gamma = \gamma_* := (\eta^2+\beta^2)/\delta$, from (26) one has $c_0 = 1 \geq c_1 = c_2 = \sqrt{c_3} = 2\eta/(\delta+\gamma_*)$, where the inequality $1 \geq 2\eta/(\delta+\gamma_*)$ follows by noting the equivalent relation $\delta^2 - 2\eta\delta + \eta^2 + \beta^2 \geq 0$ for all $\eta, \delta > 0$. Thus, for $\gamma = \gamma_*$, the bounds in (27) are given by

$$\|\mathcal{P}_{\delta,\gamma_*}\| \leq 1, \quad \|\mathcal{P}_{\delta,\gamma_*}^{-1}\| \leq \frac{\delta+\gamma_*}{2\eta} = \frac{1}{2\eta}\left(\delta + \frac{\eta^2+\beta^2}{\delta}\right) \tag{28}$$

Applying these bounds to the condition number (22) yields the upper bound in (21). □

We now show that bound (21) is tight. We do so by construction, showing that the bound in (21) is achieved for certain matrices that satisfy Assumption 2.

Lemma 3 (Tightness)**.** $\exists\ \widehat{\mathcal{L}}$ *such that* (21) *holds with equality.*

Proof. Note that the min/max of $\|\mathcal{P}_{\delta,\gamma}\boldsymbol{v}\|^2/\|\boldsymbol{v}\|^2$ over $\boldsymbol{v}$ for real-valued $\mathcal{P}_{\delta,\gamma}$ is equivalent when minimizing over real or complex $\boldsymbol{v}$; we now consider complex $\boldsymbol{v}$ for theoretical purposes. To that end, let $\boldsymbol{v} = (\gamma I - \widehat{\mathcal{L}})(\delta I - \widehat{\mathcal{L}})\boldsymbol{w}$, but suppose that $(\mathrm{i}\xi, \boldsymbol{w})$ is an eigenpair of $\widehat{\mathcal{L}}$, with ξ a real number and $\boldsymbol{w}$ a complex eigenvector. Plugging into $\|\mathcal{P}_{\delta,\gamma}\boldsymbol{v}\|^2$ (23) and $\|\boldsymbol{v}\|^2$ (24), and taking the ratio as in (25), define the following function of ξ:

$$\mathcal{H}_{\delta,\gamma}(\xi) := \left.\frac{\|\mathcal{P}_{\delta,\gamma}\boldsymbol{v}\|^2}{\|\boldsymbol{v}\|^2}\right|_{\widehat{\mathcal{L}}\boldsymbol{w} = \mathrm{i}\xi\boldsymbol{w}} = \frac{|(\eta - \mathrm{i}\xi)^2 + \beta^2|^2}{|(\delta\gamma - \xi^2 - \mathrm{i}(\delta+\gamma)\xi|^2} = \frac{(\delta\gamma_* - \xi^2)^2 + (2\eta\xi)^2}{(\delta\gamma - \xi^2)^2 + [\xi(\delta+\gamma)]^2}, \tag{29}$$

where we have made use of $\delta\gamma_* = \eta^2 + \beta^2$. By virtue of restricting that $\boldsymbol{w}$ be an eigenvector, from (22) we have

$$\frac{1}{\|\mathcal{P}_{\delta,\gamma}^{-1}\|^2} = \min_{\boldsymbol{v}\neq 0} \frac{\|\mathcal{P}_{\delta,\gamma}\boldsymbol{v}\|^2}{\|\boldsymbol{v}\|^2} \leq \mathcal{H}_{\delta,\gamma}(\xi) \leq \max_{\boldsymbol{v}\neq 0} \frac{\|\mathcal{P}_{\delta,\gamma}\boldsymbol{v}\|^2}{\|\boldsymbol{v}\|^2} = \|\mathcal{P}_{\delta,\gamma}\|^2. \tag{30}$$

That is, any value of $1/\mathcal{H}_{\delta,\gamma}(\xi)$ serves as a lower bound on $\|\mathcal{P}_{\delta,\gamma}^{-1}\|^2$, while any value of $\mathcal{H}_{\delta,\gamma}(\xi)$ serves as a lower bound on $\|\mathcal{P}_{\delta,\gamma}\|^2$. Therefore, the ratio of any two values of $\mathcal{H}_{\delta,\gamma}(\xi)$ provides a lower bound on $\kappa^2(\mathcal{P}_{\delta,\gamma})$.

We now show that bound (21) on $\kappa(\mathcal{P}_{\delta,\gamma_*})$ is tight. Considering (29) at the judiciously chosen eigenvalues of $\mathrm{i}\xi = \{0, \pm\mathrm{i}\sqrt{\delta\gamma_*}\}$, we have

$$\mathcal{H}_{\delta,\gamma}(0) = \frac{\gamma_*^2}{\gamma^2}, \qquad \mathcal{H}_{\delta,\gamma}(\pm\sqrt{\delta\gamma_*}) = \frac{(2\eta)^2\gamma_*}{\delta(\gamma-\gamma_*)^2 + \gamma_*(\delta+\gamma)^2}. \tag{31}$$

First observe from (30) and (31) that $\|\mathcal{P}_{\delta,\gamma_*}\|^2 \geq \mathcal{H}_{\delta,\gamma_*}(0) = 1$, and thus the upper bound on $\|\mathcal{P}_{\delta,\gamma_*}\|$ from (28) achieves equality for a matrix $\widehat{\mathcal{L}}$ having an eigenvalue of $\xi = 0$. Secondly, observe from (30) and (31) that $\|\mathcal{P}_{\delta,\gamma_*}^{-1}\|^2 \geq 1/\mathcal{H}_{\delta,\gamma_*}(\pm\sqrt{\delta\gamma_*}) = [(\delta+\gamma_*)/(2\eta)]^2$, and thus the upper bound on $\|\mathcal{P}_{\delta,\gamma_*}^{-1}\|$ from (28) achieves equality for a matrix $\widehat{\mathcal{L}}$ having eigenvalues $\mathrm{i}\xi = \pm\mathrm{i}\sqrt{\delta\gamma_*}$. Therefore, bound (21) on $\kappa(\mathcal{P}_{\delta,\gamma_*})$ achieves equality for any matrix $\widehat{\mathcal{L}}$ having eigenvalues $\{0, \pm\mathrm{i}\sqrt{\delta\gamma_*}\}$.[7] □

Last, having shown that (21) is tight, we now show that $\gamma = \gamma_*$ is optimal in terms of minimizing the maximum condition number of all $\widehat{\mathcal{L}}$ that satisfy Assumption 2, by showing that for $\gamma \neq \gamma_*$, $\exists$ matrices $\widehat{\mathcal{L}}$ for which $\kappa(\mathcal{P}_{\delta,\gamma}) > (\delta+\gamma_*)/2\eta$.

Lemma 4 (Optimal γ_*)**.** $\gamma_* = \mathrm{argmin}_{\gamma\in(0,\infty)} \max_{\widehat{\mathcal{L}}} \kappa(\mathcal{P}_{\delta,\gamma})$

Proof. Once again, consider the matrix $\widehat{\mathcal{L}}$ from above with eigenvalues $\{0, \pm\mathrm{i}\sqrt{\delta\gamma_*}\}$, such that $\kappa(\mathcal{P}_{\delta,\gamma_*}) = (\delta+\gamma_*)/(2\eta)$. Now, observe from this, (30), and (31), that for $0 < \gamma < \gamma_*$,

$$\kappa^2(\mathcal{P}_{\delta,\gamma_*}) < \frac{\gamma_*[\delta(\gamma-\gamma_*)^2 + \gamma_*(\delta+\gamma)^2]}{(2\eta\gamma)^2} = \frac{\mathcal{H}_{\delta,\gamma}(0)}{\mathcal{H}_{\delta,\gamma}(\pm\sqrt{\delta\gamma_*})} \leq \kappa^2(\mathcal{P}_{\delta,\gamma}), \tag{32}$$

by noting that the first inequality in (32) is equivalent to $\gamma_*(\gamma_*-\gamma)^2 + (\gamma_*-\gamma)[2\gamma_*\gamma + \delta(\gamma_*+\gamma)] > 0$, which is clearly true when $0 < \gamma < \gamma_*$. Now suppose that $\widehat{\mathcal{L}}$ has eigenvalues $\mathrm{i}\xi \to \pm\mathrm{i}\infty$, which, when substituted into (29), yields $\lim_{\xi\to\pm\infty} \mathcal{H}_{\delta,\gamma}(\xi) = 1$. Combining with (30) and (31), we have for $\gamma_* < \gamma < \infty$,

$$\kappa^2(\mathcal{P}_{\delta,\gamma_*}) < \frac{\delta(\gamma-\gamma_*)^2 + \gamma_*(\delta+\gamma)^2}{(2\eta)^2\gamma_*} = \frac{\mathcal{H}_{\delta,\gamma}(\pm\infty)}{\mathcal{H}_{\delta,\gamma}(\pm\sqrt{\delta\gamma_*})} \leq \kappa^2(\mathcal{P}_{\delta,\gamma}), \tag{33}$$

by noting that the first inequality in (33) is equivalent to $\delta(\gamma-\gamma_*)^2 + \gamma_*(\gamma-\gamma_*)(2\delta + \gamma_* + \gamma) > 0$, which is clearly satisfied for $\gamma > \gamma_*$.

By construction in (32) and (33), we have shown that for all $\gamma \in (0,\infty)\setminus\gamma_*$, there exist matrices $\widehat{\mathcal{L}}$ such that $\kappa(\mathcal{P}_{\delta,\gamma}) > \kappa(\mathcal{P}_{\delta,\gamma_*}) = (\delta+\gamma_*)/(2\eta)$. It therefore holds for general $\widehat{\mathcal{L}}$ satisfying Assumption 2 that a tight upper bound on $\kappa(\mathcal{P}_{\delta,\gamma})$ for $\gamma \in (0,\infty)\setminus\gamma_*$ must be larger than the tight upper bound of $\kappa(\mathcal{P}_{\delta,\gamma_*}) \leq (\delta+\gamma_*)/(2\eta)$. Hence $\gamma = \gamma_*$ is the minimizer over $\gamma \in (0,\infty)$ of a tight upper bound on $\kappa(\mathcal{P}_{\delta,\gamma})$. □

[7] By nature of the continuity of eigenvalues and continuity of $\mathcal{H}_{\delta,\gamma}(\xi)$ at $\xi = 0$, there also exist nonsingular matrices with condition number within ϵ of (21) for any $\epsilon > 0$.

□

Proof of Corollary 1. From Theorem 1, a tight upper bound on the condition number of $\mathcal{P}_{\delta,\gamma}$ (13) over all $\widehat{\mathcal{L}}$ is minimized with respect to γ when $\gamma = \gamma_*$ (20), with its minimum value given by (21). To minimize bound (21) with respect to δ, we differentiate it and observe for $\delta > 0$ that there is only one critical point at $\delta = \sqrt{\eta^2 + \beta^2}$. Since this function is increasing as $\delta \to 0^+$ and $\delta \to \infty$, this critical point must be a local minimum. Therefore, the tight upper bound (21) is minimized when $\delta = \sqrt{\eta^2 + \beta^2}$. Substituting $\delta = \sqrt{\eta^2 + \beta^2}$ into (20) yields $\gamma_* = \sqrt{\eta^2 + \beta^2}$. Finally, substituting $\delta = \gamma_* = \sqrt{\eta^2 + \beta^2}$ into (21) and noting that $\mathcal{P}_{\gamma,\gamma}$ (13) is equivalent to $\mathcal{P}_\gamma$ (14) yields bound (16). □

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-JRNL-817946). Los Alamos National Laboratory report number LA-UR-20-30412. This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes.

References

[1] G. Akrivis, C. Makridakis, and R. H. Nochetto, *Galerkin and Runge-Kutta methods: unified formulation, a posteriori error estimates and nodal superconvergence*, Numerische Mathematik, 118 (2011), pp. 429–456, doi:10.1007/s00211-011-0363-6.

[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, J. Dahm, D. Medina, and S. Zampini, *MFEM: a modular finite element methods library*, Computers & Mathematics with Applications, (2020), doi:10.1016/j.camwa.2020.06.009.

[3] D. N. Arnold, *An interior penalty finite element method with discontinuous elements*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 742–760, doi:10.1137/0719052.

[4] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini, *Unified analysis of discontinuous Galerkin methods for elliptic problems*, SIAM Journal on Numerical Analysis, 39 (2002), pp. 1749–1779, doi:10.1137/S0036142901384162.

[5] S. Basting and E. Bänsch, *Preconditioners for the Discontinuous Galerkin time-stepping method of arbitrary order*, ESAIM: Mathematical Modelling and Numerical Analysis, 51 (2017), pp. 1173–1195, doi:10.1051/m2an/2016055.

[6] T. A. Bickart, *An Efficient Solution Process for Implicit Runge–Kutta Methods*, SIAM Journal on Numerical Analysis, 14 (1977), pp. 1022–1027, doi:10.1137/0714069.

[7] W. C. Brown, *Matrices over commutative rings*, Marcel Dekker, Inc., 1993.

[8] K. Burrage, *Efficiently Implementable Algebraically Stable Runge–Kutta Methods*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 245–258, doi:10.1137/0719015.

[9] J. Butcher and D. Chen, *A new type of singly-implicit Runge–Kutta method*, Applied Numerical Mathematics, 34 (2000), pp. 179–188, doi:10.1016/s0168-9274(99)00126-9.

[10] J. C. Butcher, *On the implementation of implicit Runge-Kutta methods*, BIT Numerical Mathematics, 16 (1976), pp. 237–240, doi:10.1007/bf01932265.

[11] C. Canuto, P. Gervasio, and A. Quarteroni, *Finite-element preconditioning of G-NI spectral methods*, SIAM Journal on Scientific Computing, 31 (2010), pp. 4422–4451, doi:10.1137/090746367.

[12] H. Chen, *A splitting preconditioner for the iterative solution of implicit Runge-Kutta and boundary value methods*, BIT Numerical Mathematics, 54 (2014), pp. 607–621, doi:10.1007/s10543-014-0467-3.

[13] H. Chen, *Kronecker product splitting preconditioners for implicit Runge-Kutta discretizations of viscous wave equations*, Applied Mathematical Modelling, 40 (2016), pp. 4429–4440, doi:10.1016/j.apm.2015.11.037.

[14] B. Cockburn and C.-W. Shu, *Runge-Kutta discontinuous Galerkin methods for convection-dominated problems*, Journal of Scientific Computing, 16 (2001), pp. 173–261, doi:10.1023/a:1012873910884.

[15] R. D. Falgout and U. M. Yang, *hypre: A library of high performance preconditioners*, European Conference on Parallel Processing, 2331 LNCS (2002), pp. 632–641.

[16] P. E. Farrell, R. C. Kirby, and J. Marchena-Menendez, *Irksome: Automating runge–kutta time-stepping for finite element methods*, arXiv preprint arXiv:2006.16282, (2020).

[17] M. J. Gander and M. Neumuller, *Analysis of a new space-time parallel multigrid algorithm for parabolic problems*, SIAM Journal on Scientific Computing, 38 (2016), pp. A2173–A2208.

[18] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems*, Springer Series in Computational Mathematics, Springer Berlin Heidelberg, 1996.

[19] E. Hairer, G. Wanner, and C. Lubich, *Geometric Numerical Integration, Structure-Preserving Algorithms for Ordinary Differential Equations*, (2002), doi:10.1007/978-3-662-05018-7.

[20] W. Hoffmann and J. J. B. D. Swart, *Approximating Runge-Kutta matrices by triangular matrices*, BIT Numerical Mathematics, 37 (1997), pp. 346–354, doi:10.1007/bf02510217.

[21] P. J. v. d. Houwen and J. J. B. d. Swart, *Parallel linear system solvers for Runge-Kutta methods*, Advances in Computational Mathematics, 7 (1997), pp. 157–181, doi:10.1023/a:1018990601750.

[22] P. J. v. d. Houwen and J. J. B. d. Swart, *Triangularly Implicit Iteration Methods for ODE-IVP Solvers*, SIAM Journal on Scientific Computing, 18 (1997), pp. 41–55, doi:10.1137/s1064827595287456.

[23] L. O. Jay, *Inexact Simplified Newton Iterations for Implicit Runge-Kutta Methods*, SIAM Journal on Numerical Analysis, 38 (2000), pp. 1369–1388, doi:10.1137/s0036142999360573.

[24] X. Jiao, X. Wang, and Q. Chen, *Optimal and low-memory near-optimal preconditioning of fully implicit runge-kutta schemes for parabolic pdes*, arXiv preprint arXiv:2012.12779, (2020).

[25] C. Kennedy and M. H. Carpenter, *Diagonally Implicit Runge-Kutta Methods for Ordinary Differential Equations. A Review*, tech. report, 2016.

[26] M. Kronbichler and K. Ljungkvist, *Multigrid for matrix-free high-order finite element computations on graphics processors*, ACM Transactions on Parallel Computing, 6 (2019), pp. 1–32, doi:10.1145/3322813.

[27] P. Lasaint and P. Raviart, *On a finite element method for solving the neutron transport equation*, Mathematical Aspects of Finite Elements in Partial Differential Equations, (1974), pp. 89–123, doi:10.1016/b978-0-12-208350-1.50008-x.

[28] J. V. Lent and S. Vandewalle, *Multigrid Methods for Implicit Runge–Kutta and Boundary Value Method Discretizations of Parabolic PDEs*, SIAM Journal on Scientific Computing, 27 (2005), pp. 67–92, doi:10.1137/030601144.

[29] R. J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*, vol. 98, Siam, 2007.

[30] C. Makridakis and R. H. Nochetto, *A posteriori error analysis for higher order dissipative methods for evolution problems*, Numerische Mathematik, 104 (2006), pp. 489–514, doi:10.1007/s00211-006-0013-6.

[31] T. A. Manteuffel, S. Münzenmaier, J. Ruge, and B. S. Southworth, *Nonsymmetric reduction-based algebraic multigrid*, SIAM J. Sci. Comput., 41 (2019), pp. S242–S268, doi:10.1137/18M1193761.

[32] T. A. Manteuffel, J. Ruge, and B. S. Southworth, *Nonsymmetric algebraic multigrid based on local approximate ideal restriction (ℓAIR)*, SIAM J. Sci. Comput., 40 (2018), pp. A4105–A4130, doi:10.1137/17M1144350.

[33] 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, doi:10.1137/05064093x.

[34] J. Melenk, K. Gerdes, and C. Schwab, *Fully discrete hp-finite elements: fast quadrature*, Computer Methods in Applied Mechanics and Engineering, 190 (2001), pp. 4339–4364, doi:10.1016/s0045-7825(00)00322-4.

[35] T. K. Nilssen, G. A. Staff, and K. Mardal, *Order optimal preconditioners for fully implicit Runge-Kutta schemes applied to the bidomain equations*, Numerical Methods for Partial Differential Equations, 27 (2011), pp. 1290–1312, doi:10.1002/num.20582.

[36] S. P. Nørsett, *Runge-kutta methods with a multiple real eigenvalue only*, BIT Numerical Mathematics, 16 (1976), pp. 388–393.

[37] Y. Notay, *Flexible conjugate gradients*, SIAM Journal on Scientific Computing, 22 (2000), pp. 1444–1460, doi:10.1137/s1064827599362314, https://doi.org/10.1137%2Fs1064827599362314.

[38] B. Orel, *Real pole approximations to the exponential function*, BIT, 31 (1991), pp. 144–159, doi:10.1007/bf01952790.

[39] S. A. Orszag, *Spectral methods for problems in complex geometries*, Journal of Computational Physics, 37 (1980), pp. 70–92, doi:10.1016/0021-9991(80)90005-4.

[40] W. Pazner, *Efficient low-order refined preconditioners for high-order matrix-free continuous and discontinuous Galerkin methods*, SIAM Journal on Scientific Computing (In Press), (2020).

[41] 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, doi:10.1016/j.jcp.2017.01.050.

[42] 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*, arXiv preprint arXiv:2010.11377, (2020).

[43] S. C. Reddy and L. N. Trefethen, *Stability of the method of lines*, Numerische Mathematik, 62 (1992), pp. 235–267, doi:10.1007/bf01396228.

[44] T. Richter, A. Springer, and B. Vexler, *Efficient numerical realization of discontinuous Galerkin methods for temporal discretization of parabolic problems*, Numerische Mathematik, 124 (2013), pp. 151–182, doi:10.1007/s00211-012-0511-7.

[45] R. R. Rosales, B. Seibold, D. Shirokoff, and D. Zhou, *Spatial manifestations of order reduction in runge-kutta methods for initial boundary value problems*, arXiv preprint arXiv:1712.00897, (2017).

[46] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM Journal on Scientific Computing, 14 (1993), pp. 461–469.

[47] D. Schötzau and C. Schwab, *Time Discretization of Parabolic Problems by the HP-Version of the Discontinuous Galerkin Finite Element Method*, SIAM Journal on Numerical Analysis, 38 (2000), pp. 837–875, doi:10.1137/s0036142999352394.

[48] K. Shahbazi, *An explicit expression for the penalty parameter of the interior penalty method*, Journal of Computational Physics, 205 (2005), pp. 401–407.

[49] I. Smears, *Robust and efficient preconditioners for the discontinuous Galerkin time-stepping method*, IMA Journal of Numerical Analysis, (2016), p. drw050, doi:10.1093/imanum/drw050.

[50] B. S. Southworth, O. A. Krzysik, and W. Pazner, *Fast solution of fully implicit Runge-Kutta and discontinuous Galerkin in time for numerical PDEs, part II: nonlinearities and DAEs*, arXiv preprint arXiv:2101.01776, (2021).

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

[52] L. N. Trefethen and M. Embree, *Spectra and pseudospectra: the behavior of nonnormal matrices and operators*, Princeton University Press, 2005.