# FAST ITERATIVE SOLVER FOR THE ALL-AT-ONCE RUNGE–KUTTA DISCRETIZATION

SANTOLO LEVEQUE[*], LUCA BERGAMASCHI[†], ÁNGELES MARTÍNEZ[‡], AND JOHN W. PEARSON[§]

**Abstract.** In this article, we derive fast and robust preconditioned iterative methods for the all-at-once linear systems arising upon discretization of time-dependent PDEs. The discretization we employ is based on a Runge–Kutta method in time, for which the development of robust solvers is an emerging research area in the literature of numerical methods for time-dependent PDEs. By making use of classical theory of block matrices, one is able to derive a preconditioner for the systems considered. An approximate inverse of the preconditioner so derived consists in a fixed number of linear solves for the system of the stages of the method. We thus propose a preconditioner for the latter system based on a singular value decomposition (SVD) of the (real) Runge–Kutta matrix $A_{\mathrm{RK}} = U\Sigma V^\top$. Supposing $A_{\mathrm{RK}}$ is invertible, we prove that the spectrum of the system for the stages preconditioned by our SVD-based preconditioner is contained within the right-half of the unit circle, under suitable assumptions on the matrix $U^\top V$ (which is well defined due to the polar decomposition of $A_{\mathrm{RK}}$). We show the numerical efficiency of our SVD-based preconditioner by solving the system of the stages arising from the discretization of the heat equation and the Stokes equations, with sequential time-stepping. Finally, we provide numerical results of the all-at-once approach for both problems.

**Key words.** Time-dependent problems, Parabolic PDE, Preconditioning, Saddle-point systems

**AMS subject classifications.** 65F08, 65F10, 65N22

## 1. Introduction.

Time-dependent partial differential equations (PDEs) arise very often in the sciences, from mechanics to thermodynamics, from biology to economics, from engineering to chemistry, just to name a few. In fact, many physical processes can be described by the relation of some physical quantities using a differential operator. As problems involving (either steady or unsteady) PDEs usually lack a closed form solution, numerical methods are employed in order to find an approximation of it. These methods are based on discretizations of the quantities involved. For time-dependent PDEs, the discretization has also to take into account the time derivative. Classical numerical approaches employed to solve time-dependent PDEs result in a sequence of linear systems to be solved sequentially, mimicking the evolution in time of the physical quantities involved.

In the last few decades, many researchers have devoted their effort to devising parallel-in-time methods for the numerical solution of time-dependent PDEs, leading to the development of the Parareal [20], the Parallel Full Approximation Scheme in Space and Time (PFASST) [6], and the Multigrid Reduction in Time (MGRIT) [7, 8] algorithms, for example. As opposed to the classical approach, for which in order to obtain an approximation of the solution of an initial boundary value problem at a time $t$ one has to find an approximation of the solution at all the previous times, parallel-in-time methods approximate the solution of the problem for all times *concurrently*. This

[*]CRM Ennio De Giorgi, Scuola Normale Superiore, Piazza dei Cavalieri 3, 56126, Pisa, Italy (santolo.leveque@sns.it)

[†]Department of Civil Environmental and Architectural Engineering, University of Padova, Via Marzolo 9, 35100, Padova, Italy (luca.bergamaschi@unipd.it)

[‡]Department of Mathematics and Geosciences, University of Trieste, Via Valerio 12/1, 34127, Trieste, Italy (amartinez@units.it)

[§]School of Mathematics, The University of Edinburgh, James Clerk Maxwell Building, The King's Buildings, Peter Guthrie Tait Road, Edinburgh, EH9 3FD, United Kingdom (j.pearson@ed.ac.uk)

in turns allows one to speed-up the convergence of the numerical solver by running the code on parallel architectures.

Among all the parallel-in-time approaches for solving time-dependent PDEs, increasing consideration has been given to the one introduced by Maday and Rønquist in 2008 [21], see, for example, [10, 23]. This approach is based on a *diagonalization* (ParaDiag) of the *all-at-once* linear system arising upon the discretization of the differential operator. The diagonalization of the all-at-once system can be performed in two ways. First, by employing a non-constant time-step, for instance by employing a geometrically increasing sequence $\tau_1, \tau_2, \ldots, \tau_{n_t}$ as in [21], one can prove that the discretized system is diagonalizable [9, 10]. Second, one can employ a constant time-step and approximate the block-Toeplitz matrix arising upon discretization by employing, for instance, a circulant approximation [23]. Either way, the diagonalization of the all-at-once linear system under examination allows one to devise a preconditioner that can be run in parallel, obtaining thus a substantial speed-up.

Despite the efficiency and robustness of the ParaDiag preconditioner applied to the all-at-once discretization of the time-dependent PDE studied, this approach has a drawback. In fact, the discretization employed is based on linear multistep methods. It is well known that this class of methods are (in general) not A-stable, a property that allows one to choose an arbitrary time-step for the integration. Specifically, an A-stable linear multistep method cannot have order of convergence greater than two, as stated by the second Dahlquist barrier, see for example [18, Theorem 6.6]. In contrast, it is well known that one can devise an A-stable (implicit) Runge–Kutta method of any given order. Further, implicit Runge–Kutta methods have better stability properties than the linear multistep methods, (e.g., L- or B-stability, see for instance [4, 12, 18]). For these reasons, in this work we present a preconditioner for the all-at-once linear system arising when a Runge–Kutta method is employed for the time discretization. To the best of the authors' knowledge, this is the first attempt that fully focuses on deriving such a preconditioner for the all-at-once Runge–Kutta discretization. In this regard, we would like to mention the work [17], where the authors derived two solvers for the linear systems arising from an all-at-once approach of space-time discretization of time-dependent PDEs. The first solver is based on the observation that the numerical solution can be written as the sum of the solution of a system involving an $\alpha$-circulant matrix with the solution of a Sylvester equations with a right-hand side of low rank. The second approach is based on an interpolation strategy: the authors observe that the numerical solution can be approximated (under suitable assumptions) by a linear combination of the solutions of systems involving $\alpha$-circulant matrices, where $\alpha$ is the $j$th root of unity, for some integer $j$. Although the authors mainly focused on multistep methods, they adapted the two strategies in order to tackle also the all-at-once systems obtained when employing a Runge–Kutta method in time. We would like to note that the preconditioner derived in our work does not exploit the block-Toeplitz structure of the system arising upon discretization, therefore it could be employed also with a non-constant time-step.

We would like to mention that, despite the fact that one can obtain better stability properties when employing Runge–Kutta methods, this comes to a price. In fact, this class of method results in very large linear systems with a very complex structure: in order to derive an approximation of the solution at a time $t$, one has to solve a linear system for the *stages* of the discretization. Of late, a great effort has been devoted to devising preconditioners for the numerical solution for the stages of a Runge–Kutta method, see for example [1, 2, 22, 27, 30, 31]. As we will show below, the preconditioner for the all-at-once Runge–Kutta discretization results in a block-diagonal solve for all the stages of all the time-steps, and a Schur complement whose inverse can be applied by solving again for the systems for the stages of the method. Since the most expensive task is to (approximately) invert the system for the stages, in this work we also introduce a new block-preconditioner for the stage solver. This preconditioner is based on an SVD of the Runge–Kutta coefficients matrix, and has many advantages as we will describe below. Alternatively, one can also employ the strategies described in [1, 2, 22, 30], for example.

This paper is structured as follows. In Section 2, we introduce Runge–Kutta methods and present the all-at-once system obtained upon discretization of a time-dependent differential equation by employing this class of methods in time. Then, we give specific details of the all-at-once system for the discretization of the heat equation and the Stokes equations. In Section 3, we present the proposed preconditioner for the all-at-once system together with the approximation of the system of the stages of the Runge–Kutta method, for all the problems considered in this work. In Section 4, we show the robustness of the proposed preconditioning strategy. Finally, conclusions and future work are given in Section 5.

## 2. Runge–Kutta methods.

In this section, we present the linear systems arising upon discretization when employing a Runge–Kutta method. In what follows, $I_m$ represents the identity matrix of dimension $m$.

For simplicity, we integrate the ordinary differential equation $v'(t) = f(v(t), t)$ between 0 and a final time $t_f > 0$, given the initial condition $v(0) = v_0$. After dividing the time interval $[0, t_f]$ into $n_t$ subintervals with constant time-step $\tau$, the discretization of an $s$-stages Runge–Kutta method applied to $v'(t) = f(v(t), t)$ reads as follows:

$$v_{n+1} = v_n + \tau \sum_{i=1}^{s} b_i k_{i,n}, \quad n = 0, \dots, n_t - 1,$$

where the stages $k_{i,n}$ are given by[1]

$$k_{i,n} = f\left(v_n + \tau \sum_{j=1}^{s} a_{i,j} k_{j,n}, t_n + c_i \tau\right), \qquad i = 1, \dots, s, \tag{2.1}$$

with $t_n = n\tau$. The Runge–Kutta method is uniquely defined by the coefficients $a_{i,j}$, the weights $b_i$, and the nodes $c_i$, for $i, j = 1, \dots, s$. For this reason, an $s$-stages Runge–Kutta method is defined by the following Butcher tableau:

$$\begin{array}{c|ccc} c_1 & a_{1,1} & \dots & a_{1,s} \\ \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s,1} & \dots & a_{s,s} \\ \hline & b_1 & \dots & b_s \end{array}$$

or in a more compact form

$$\begin{array}{c|c} \mathbf{c}_{\text{RK}} & A_{\text{RK}} \\ \hline & \mathbf{b}_{\text{RK}}^\top \end{array}$$

[1]Note that the stages $k_{i,n}$ for $i = 1, \dots, s$ represent an approximation of the time derivative of $v$.

A Runge–Kutta method is said to be explicit if $a_{i,j} = 0$ when $i \leq j$, otherwise it is called implicit. Note that for implicit Runge–Kutta method the stages are obtained by solving the non-linear equations (2.1).

In the following, we will employ Runge–Kutta methods as the time discretization for the time-dependent differential equations considered in this work. In particular, we will focus here on the Stokes equations. Since this system of equations is properly a differential-algebraic equation (DAE), we cannot employ the Runge–Kutta method as we have described above. In order to fix the notations, given a subset $\Omega \subset \mathbb{R}^d$, with $d = 1, 2, 3$, and a final time $t_f > 0$, we consider the following DAE:

$$\begin{cases} \frac{\partial v}{\partial t} + \mathcal{D}_1 v = f(\mathbf{x}, t) & \text{in } \Omega \times (0, t_f), \\ \mathcal{D}_2 v = g(\mathbf{x}, t) & \text{in } \Omega \times (0, t_f), \end{cases}$$

given some suitable initial and boundary conditions. Here, $\mathcal{D}_1$ and $\mathcal{D}_2$ are differential operators (only) in space. In addition, the variable $v$ may be a vector, and contains all the physical variables described by the DAE (e.g., the *temperature* for the heat equation, or the *velocity* of the fluid and the *kinematic pressure* for the Stokes equations). In what follows, we will suppose that the differential operators $\mathcal{D}_1$ and $\mathcal{D}_2$ are linear and time-independent. In addition, we will suppose that only Dirichlet boundary conditions are imposed.

Given suitable discretizations $\mathbf{D}_1$ and $\mathbf{D}_2$ of $\mathcal{D}_1$ and $\mathcal{D}_2$ respectively, after dividing the time interval $[0, t_f]$ into $n_t$ subintervals with constant time-step $\tau$, a Runge–Kutta discretization reads as follows:

$$\mathbf{M}\mathbf{v}_{n+1} = \mathbf{M}\mathbf{v}_n + \tau \mathbf{M} \sum_{i=1}^{s} b_i \mathbf{k}_{i,n}, \quad n = 0, \ldots, n_t - 1, \tag{2.2}$$

with a suitable discretization of the initial and boundary conditions. Here, $\mathbf{v}_n$ represents the discretization of $v$ at time $t_n$. The vectors $\mathbf{k}_{i,n}$ are defined as follows:

$$\begin{cases} \mathbf{M}\mathbf{k}_{i,n} + \mathbf{D}_1 \mathbf{v}_n + \tau \mathbf{D}_1 \displaystyle\sum_{j=1}^{s} a_{i,j} \mathbf{k}_{j,n} = \mathbf{f}_{i,n}, & i = 1, \ldots, s, \\ \mathbf{D}_2 \mathbf{v}_n + \tau \mathbf{D}_2 \displaystyle\sum_{j=1}^{s} a_{i,j} \mathbf{k}_{j,n} = \mathbf{g}_{i,n}, & i = 1, \ldots, s, \end{cases} \tag{2.3}$$

for $n = 0, \ldots, n_t - 1$, where $\mathbf{f}_{i,n}$ and $\mathbf{g}_{i,n}$ are discretizations of the functions $f$ and $g$ at the time $t_n + c_i \tau$, respectively. Finally, the matrix $\mathbf{M}$ is a suitable discretization of the identity operator. We recall that the stages $\mathbf{k}_{i,n}$ for $i = 1, \ldots, s$ represent an approximation of $\frac{\partial v}{\partial t}$; therefore, the (Dirichlet) boundary conditions on $\mathbf{k}_{i,n}$ are given by the time derivatives of the corresponding boundary conditions on $v$. Note that, if we relax the assumption of $\mathcal{D}_1$ and $\mathcal{D}_2$ being time-independent, (2.3) would be properly written as follows:

$$\begin{cases} \mathbf{M}\mathbf{k}_{i,n} + \mathbf{D}_1^{(n,i)} \mathbf{v}_n + \tau \mathbf{D}_1^{(n,i)} \displaystyle\sum_{j=1}^{s} a_{i,j} \mathbf{k}_{j,n} = \mathbf{f}_{i,n}, & i = 1, \ldots, s, \\ \mathbf{D}_2^{(n,i)} \mathbf{v}_n + \tau \mathbf{D}_2^{(n,i)} \displaystyle\sum_{j=1}^{s} a_{i,j} \mathbf{k}_{j,n} = \mathbf{g}_{i,n}, & i = 1, \ldots, s, \end{cases}$$

$\mathbf{D}_1^{(n,i)}$ and $\mathbf{D}_2^{(n,i)}$ being the discretizations of the differential operators $\mathcal{D}_1$ and $\mathcal{D}_2$ at time $t_n + c_i \tau$, respectively, for $n = 0, \ldots, n_t - 1$.

In compact form, we can rewrite (2.3) as follows:

$$\begin{cases} (I_s \otimes \mathbf{M})\mathbf{k}_n + (\mathbf{e} \otimes \mathbf{D}_1)\mathbf{v}_n + \tau(A_{\mathrm{RK}} \otimes \mathbf{D}_1)\mathbf{k}_n = \mathbf{f}_n, \\ (\mathbf{e} \otimes \mathbf{D}_2)\mathbf{v}_n + \tau(A_{\mathrm{RK}} \otimes \mathbf{D}_2)\mathbf{k}_n = \mathbf{g}_n, \end{cases}$$

where $\mathbf{e} \in \mathbb{R}^s$ is the column vector of all ones. Here, we set $\mathbf{k}_n = [\mathbf{k}_{1,n}^\top, \ldots, \mathbf{k}_{s,n}^\top]^\top$, $\mathbf{f}_n = [\mathbf{f}_{1,n}^\top, \ldots, \mathbf{f}_{s,n}^\top]^\top$, and $\mathbf{g}_n = [\mathbf{g}_{1,n}^\top, \ldots, \mathbf{g}_{s,n}^\top]^\top$. Further, we may rewrite (2.2) as

$$\mathbf{M}\mathbf{v}_{n+1} = \mathbf{M}\mathbf{v}_n + \tau(\mathbf{b}_{\mathrm{RK}}^\top \otimes \mathbf{M})\mathbf{k}_n.$$

We are now able to write the all-at-once system for the Runge–Kutta discretization in time. By setting $\mathbf{v} = [\mathbf{v}_0^\top, \ldots, \mathbf{v}_{n_t}^\top]^\top$ and $\mathbf{k} = [\mathbf{k}_0^\top, \ldots, \mathbf{k}_{n_t-1}^\top]^\top$, we can rewrite (2.2)–(2.3) in matrix form as

$$\underbrace{\begin{bmatrix} \Phi & \Psi_1 \\ \Psi_2 & \Theta \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} \mathbf{v} \\ \mathbf{k} \end{bmatrix} = \mathbf{b}. \tag{2.4}$$

The blocks of the matrix $\mathcal{A}$ are given by

$$\Phi = \begin{bmatrix} \mathbf{M} & & & \\ -\mathbf{M} & \ddots & & \\ & \ddots & \ddots & \\ & & -\mathbf{M} & \mathbf{M} \end{bmatrix}, \qquad \Psi_1 = -\begin{bmatrix} 0 & & \\ \tau\mathbf{b}_{\mathrm{RK}}^\top \otimes \mathbf{M} & & \\ & \ddots & \\ & & \tau\mathbf{b}_{\mathrm{RK}}^\top \otimes \mathbf{M} \end{bmatrix},$$
$$\Psi_2 = \begin{bmatrix} \mathbf{e} \otimes \mathbf{D}_1 & & & \\ \mathbf{e} \otimes \mathbf{D}_2 & & & \\ & \ddots & & \\ & & \mathbf{e} \otimes \mathbf{D}_1 & 0 \\ & & \mathbf{e} \otimes \mathbf{D}_2 & 0 \end{bmatrix}, \quad \Theta = I_{n_t} \otimes \begin{bmatrix} I_s \otimes \mathbf{M} + \tau A_{\mathrm{RK}} \otimes \mathbf{D}_1 \\ \tau A_{\mathrm{RK}} \otimes \mathbf{D}_2 \end{bmatrix}. \tag{2.5}$$

Note that $\Theta$ is block-diagonal.

In what follows, we will present the all-at-once system (2.4) for the specific cases of the heat equation and the Stokes equations.

**2.1. Heat equation.** Given a domain $\Omega \subset \mathbb{R}^d$, with $d = 1, 2, 3$, and a final time $t_f > 0$, we consider the following heat equation:

$$\begin{cases} \frac{\partial v}{\partial t} - \nabla^2 v = f(\mathbf{x}, t) & \text{in } \Omega \times (0, t_f), \\ v(\mathbf{x}, t) = g(\mathbf{x}, t) & \text{on } \partial\Omega \times (0, t_f), \\ v(\mathbf{x}, 0) = v_0(\mathbf{x}) & \text{in } \Omega, \end{cases} \tag{2.6}$$

where the functions $f$ and $g$ are known. In addition, the initial condition $v_0(\mathbf{x})$ is also given.

After dividing the time interval $[0, t_f]$ into $n_t$ subintervals, the discretization of (2.6) by employing a Runge–Kutta method reads as follows:

$$M\mathbf{v}_{n+1} = M\mathbf{v}_n + \tau M \sum_{i=1}^{s} b_i \mathbf{k}_{i,n}, \quad n = 0, \ldots, n_t - 1, \tag{2.7}$$

with $M\mathbf{v}_0 = M\mathbf{v}^0$ a suitable discretization of the initial condition. The stages $\mathbf{k}_{i,n}$ are defined as follows:

$$M\mathbf{k}_{i,n} + K\mathbf{v}_n + \tau K \textstyle\sum_{j=1}^{s} a_{i,j}\mathbf{k}_{j,n} = \mathbf{f}_{i,n}, \qquad i = 1, \dots, s, \; n = 0, \dots, n_t - 1, \tag{2.8}$$

where

$$(\mathbf{f}_{i,n})_m = \int_\Omega f(\mathbf{x}, t_n + c_i\tau)\phi_m \, \mathrm{d}\Omega, \quad i = 1, \dots, s.$$

Here, $K$ and $M$ are the *stiffness* and *mass* matrix respectively. For a Dirichlet problem, both matrices are symmetric positive definite (s.p.d.). As we mentioned above, the boundary conditions on the stages $\mathbf{k}_{i,n}$ are given by the time derivatives of the corresponding boundary conditions on $v$. Specifically, we have

$$(\mathbf{k}_{i,n})|_{\partial\Omega} = \tfrac{\partial g}{\partial t}(\cdot, t_n + c_i\tau).$$

By adopting an all-at-once approach, we can rewrite the system (2.7)–(2.8) as follows:

$$\begin{cases} M\mathbf{v}_0 = M\mathbf{v}^0, & \\ M\mathbf{v}_{n+1} - M\mathbf{v}_n - \tau M \textstyle\sum_{j=1}^{s} b_j\mathbf{k}_{j,n} = \mathbf{0}, & n = 0, \dots, n_t - 1, \\ M\mathbf{k}_{i,n} + K\mathbf{v}_n + \tau K \textstyle\sum_{j=1}^{s} a_{i,j}\mathbf{k}_{j,n} = \mathbf{f}_{i,n}, & i = 1, \dots, s, \; n = 0, \dots, n_t - 1. \end{cases}$$

In matrix form, we have

$$\underbrace{\begin{bmatrix} \Phi & \Psi_1 \\ \Psi_2 & \Theta \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \vdots \\ \mathbf{v}_{n_t} \\ \mathbf{k}_0 \\ \vdots \\ \mathbf{k}_{n_t-1} \end{bmatrix} = \begin{bmatrix} \mathbf{v}^0 \\ \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_{n_t} \\ \mathbf{f}_0 \\ \vdots \\ \mathbf{f}_{n_t-1} \end{bmatrix}, \tag{2.9}$$

where the vectors $\mathbf{b}_n$, $n = 1, \dots, n_t$, contain information about the boundary conditions. The blocks of the matrix $\mathcal{A}$ are given by

$$\Phi = \begin{bmatrix} M & & & \\ -M & \ddots & & \\ & \ddots & \ddots & \\ & & -M & M \end{bmatrix}, \qquad \Psi_1 = -\begin{bmatrix} & 0 & \\ \tau\mathbf{b}_{\mathrm{RK}}^\top \otimes M & & \\ & \ddots & \\ & & \tau\mathbf{b}_{\mathrm{RK}}^\top \otimes M \end{bmatrix},$$

$$\Psi_2 = \begin{bmatrix} \mathbf{e} \otimes K & & \\ & \ddots & \\ & \mathbf{e} \otimes K & 0 \end{bmatrix}, \quad \Theta = I_{n_t} \otimes (I_s \otimes M + \tau A_{\mathrm{RK}} \otimes K).$$

**2.2. Stokes equations.** Given a domain $\Omega \subset \mathbb{R}^d$, with $d = 2, 3$, and a final time $t_f > 0$, we consider the following Stokes equations:

$$\begin{cases} \frac{\partial \vec{v}}{\partial t} - \nabla^2 \vec{v} + \nabla p = \vec{f}(\mathbf{x}, t) & \text{in } \Omega \times (0, t_f), \\ -\nabla \cdot \vec{v} = 0 & \text{in } \Omega \times (0, t_f), \\ \vec{v}(\mathbf{x}, t) = \vec{g}(\mathbf{x}, t) & \text{on } \partial\Omega \times (0, t_f), \\ \vec{v}(\mathbf{x}, 0) = \vec{v}_0(\mathbf{x}) & \text{in } \Omega. \end{cases} \tag{2.10}$$

As above, the functions $\vec{f}$ and $\vec{g}$ as well as the initial condition $\vec{v}_0(\mathbf{x})$ are known.

After dividing the time interval $[0, t_f]$ into $n_t$ subintervals, the discretization of (2.10) by a Runge–Kutta method reads as follows:

$$\begin{aligned} M_v\mathbf{v}_{n+1} &= M_v\mathbf{v}_n + \tau M_v \textstyle\sum_{i=1}^{s} b_i\mathbf{k}_{i,n}^v, \qquad n = 0,\ldots,n_t-1,\\ M_p\mathbf{p}_{n+1} &= M_p\mathbf{p}_n + \tau M_p \textstyle\sum_{i=1}^{s} b_i\mathbf{k}_{i,n}^p, \qquad n = 0,\ldots,n_t-1, \end{aligned} \tag{2.11}$$

with $M_v\mathbf{v}_0 = M_v\mathbf{v}^0$ a suitable discretization of the initial condition, and $M_p\mathbf{p}_0 = M_p\mathbf{p}^0$ a suitable approximation of the pressure $p$ at time $t = 0$. The stages $\mathbf{k}_{i,n}^v$ and $\mathbf{k}_{i,n}^p$ are defined as follows:

$$\begin{aligned} &M_v\mathbf{k}_{i,n}^v + K_v\mathbf{v}_n + \tau K_v \textstyle\sum_{j=1}^{s} a_{i,j}\mathbf{k}_{j,n}^v + B^\top\mathbf{p}_n + \tau B^\top \textstyle\sum_{j=1}^{s} a_{i,j}\mathbf{k}_{j,n}^p = \mathbf{f}_{i,n},\\ &B\mathbf{v}_n + \tau B \textstyle\sum_{j=1}^{s} a_{i,j}\mathbf{k}_{j,n}^v = \mathbf{0}, \end{aligned} \tag{2.12}$$

for $i = 1,\ldots,s$, $n = 0,\ldots,n_t-1$, where

$$(\mathbf{f}_{i,n})_m = \int_\Omega \vec{f}(\mathbf{x}, t_n + c_i\tau)\cdot\vec{\phi}_m \, \mathrm{d}\Omega, \quad i = 1,\ldots,s.$$

Here, $K_v$ and $M_v$ (resp., $K_p$ and $M_p$) are the *vector-stiffness* and *vector-mass* matrix (resp., stiffness and mass), respectively. Finally, as above, the boundary conditions on $\mathbf{k}_{i,n}^v$ are given by

$$(\mathbf{k}_{i,n}^v)\big|_{\partial\Omega} = \tfrac{\partial\vec{g}}{\partial t}(\cdot, t_n + c_i\tau).$$

By adopting an all-at-once approach, we can rewrite the system (2.11)–(2.12) as follows:

$$\begin{cases} M_v\mathbf{v}_0 = M_v\mathbf{v}^0, & \\ M_p\mathbf{p}_0 = M_p\mathbf{p}^0, & \\ M_v\mathbf{v}_{n+1} - M_v\mathbf{v}_n - \tau M_v \sum_{j=1}^{s} b_j\mathbf{k}_{j,n}^v = \mathbf{0} & \\ M_p\mathbf{p}_{n+1} - M_p\mathbf{p}_n - \tau M_p \sum_{i=1}^{s} b_i\mathbf{k}_{i,n}^p = \mathbf{0} & \\ M_v\mathbf{k}_{i,n}^v + K_v(\mathbf{v}_n + \tau\sum_{j=1}^{s} a_{i,j}\mathbf{k}_{j,n}^v) + B^\top(\mathbf{p}_n + \tau\sum_{j=1}^{s} a_{i,j}\mathbf{k}_{j,n}^p) = \mathbf{f}_{i,n}, & i = 1,\ldots,s,\\ B\mathbf{v}_n + \tau B\sum_{j=1}^{s} a_{i,j}\mathbf{k}_{j,n}^v = \mathbf{0}, & i = 1,\ldots,s, \end{cases}$$

for $n = 0,\ldots,n_t-1$. In matrix form, the system is of the form (2.4), with

$$\begin{aligned} \mathbf{v} &= [\mathbf{v}_0^\top, \mathbf{p}_0^\top, \ldots, \mathbf{v}_{n_t}^\top, \mathbf{p}_{n_t}^\top]^\top,\\ \mathbf{k} &= [(\mathbf{k}_0^v)^\top, (\mathbf{k}_0^p)^\top, \ldots, (\mathbf{k}_{n_t-1}^v)^\top, \ldots, (\mathbf{k}_{n_t-1}^p)^\top]^\top, \end{aligned}$$

where

$$\mathbf{k}_n^v = [(\mathbf{k}_{n,1}^v)^\top, \ldots, (\mathbf{k}_{n,s}^v)^\top]^\top, \quad \mathbf{k}_n^p = [(\mathbf{k}_{n,1}^p)^\top, \ldots, (\mathbf{k}_{n,s}^p)^\top]^\top, \quad n = 0,\ldots,n_t-1.$$

Further, the blocks of the matrix $\mathcal{A}$ are as in (2.5), with $\Theta = I_{n_t}\otimes\widehat{\Theta}$, and

$$\Psi_2 = \begin{bmatrix} \widehat{\Psi}_2 & & \\ & \ddots & \\ & & \widehat{\Psi}_2 & 0 \end{bmatrix}, \qquad \mathbf{M} = \begin{bmatrix} M_v & 0 \\ 0 & M_p \end{bmatrix}.$$

Here, the blocks are given by

$$\widehat{\Psi}_2 = \begin{bmatrix} \mathbf{e}\otimes K_v & \mathbf{e}\otimes B^\top \\ \mathbf{e}\otimes B & 0 \end{bmatrix}, \quad \widehat{\Theta} = \begin{bmatrix} I_s\otimes M_v + \tau A_{\mathrm{RK}}\otimes K_v & \tau A_{\mathrm{RK}}\otimes B^\top \\ \tau A_{\mathrm{RK}}\otimes B & 0 \end{bmatrix}. \tag{2.13}$$

Note that we need an approximation of the pressure $p$ at time $t = 0$. In our tests, before solving for the all-at-once system we integrate the problem between $(0, \epsilon_{\mathrm{BE}})$ employing one step of backward Euler. The approximations of the velocity $\vec{v}$ and pressure $p$ at time $t = \epsilon_{\mathrm{BE}}$ are then employed as initial conditions for solving the problem in $(\epsilon_{\mathrm{BE}}, t_f)$. In our tests, we choose $\epsilon_{\mathrm{BE}} = h^{2.5}$, with $h$ the mesh size in space. This choice has been made as a trade-off between having an accurate enough solution at time $t = \epsilon_{\mathrm{BE}}$ and a fast solver for the backward Euler discretization. We would like to mention that finding suitable initial conditions for the problem (2.11)–(2.12) is beyond the scope of this work, and one can employ other approaches. For instance, one can employ a *solenoidal projection*, as done in [14].

## 3. Preconditioner.

Supposing that $\Theta$ is invertible, we consider as a preconditioner for the system (2.4) the following matrix:

$$\mathcal{P} = \begin{bmatrix} S & \Psi_1 \\ 0 & \Theta \end{bmatrix}, \tag{3.1}$$

where $S = -\Phi - \Psi_1 \Theta^{-1} \Psi_2$ is the Schur complement, with $\Phi$, $\Psi_1$, $\Psi_2$, and $\Theta$ defined as in (2.5). Specifically, we have

$$S = - \begin{bmatrix} \mathbf{M} & & & \\ -\mathbf{M} + X & \ddots & & \\ & \ddots & \ddots & \\ & & -\mathbf{M} + X & \mathbf{M} \end{bmatrix},$$

where

$$X = \tau \begin{bmatrix} b_1 \mathbf{M} & \dots & b_s \mathbf{M} \end{bmatrix} \begin{bmatrix} I_s \otimes \mathbf{M} + \tau A_{\mathrm{RK}} \otimes \mathbf{D}_1 \\ \tau A_{\mathrm{RK}} \otimes \mathbf{D}_2 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{e} \otimes \mathbf{D}_1 \\ \mathbf{e} \otimes \mathbf{D}_2 \end{bmatrix}.$$

The preconditioner $\mathcal{P}$ given in (3.1) is optimal. In fact, supposing also that $S$ is invertible, one can prove that $\lambda(\mathcal{P}^{-1}\mathcal{A}) = \{1\}$, and the minimal polynomial of the preconditioned matrix has degree 2, see, for instance, [16, 24]. For this reason, when employing the preconditioner $\mathcal{P}$, an appropriate iterative method should converge in at most two iterations (in exact arithmetic). However, in practical applications even forming the Schur complement $S$ may be unfeasible due to the large dimensions of the system. Besides, the block $\Theta$ may be singular, in which case not only is the Schur complement $S$ not well defined, but also we cannot apply the inverse of $\mathcal{P}$. For this reason, rather than solving for the matrix $\mathcal{P}$, we favour finding a cheap invertible approximation $\widetilde{\mathcal{P}}$ of $\mathcal{P}$, in which the block $\Theta$ is replaced by an invertible $\widetilde{\Theta}$ and the Schur complement $S$ is approximated by $\widetilde{S} \approx -\Phi - \Psi_1 \widetilde{\Theta}^{-1} \Psi_2$, respectively. In what follows, we will find approximations of the main blocks of $\mathcal{P}$.

Clearly, the matrix $\Theta$ defined in (2.5) is block-diagonal, with each diagonal-block given by the system for the stages. Therefore, a cheap method for approximately inverting the system for the stages gives also a cheap way for approximately inverting the block $\Theta$.

We now focus on an approximation of the Schur complement $S$. The latter may be factorized as follows:

$$S = -\begin{bmatrix} \mathbf{M} & & \\ & \ddots & \\ & & \mathbf{M} \end{bmatrix} \underbrace{\begin{bmatrix} I_{n_x} & & & \\ -I_{n_x}+\widehat{X} & \ddots & & \\ & \ddots & \ddots & \\ & & -I_{n_x}+\widehat{X} & I_{n_x} \end{bmatrix}}_{\widehat{S}}, \tag{3.2}$$

where

$$\widehat{X} = \tau \begin{bmatrix} b_1 I_{n_x} & \dots & b_s I_{n_x} \end{bmatrix} \begin{bmatrix} I_s \otimes \mathbf{M} + \tau A_{\mathrm{RK}} \otimes \mathbf{D}_1 \\ \tau A_{\mathrm{RK}} \otimes \mathbf{D}_2 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{e} \otimes \mathbf{D}_1 \\ \mathbf{e} \otimes \mathbf{D}_2 \end{bmatrix}. \tag{3.3}$$

From here, in order to approximately invert the Schur complement $S$ one has to employ a block-forward substitution, applying the inverse of the system for the stages inexactly.

As we mentioned, the main computational task is to (approximately) solve for the linear system of the stages. In the following, we present the strategy adopted in this work. Again, we would like to mention that one may also employ other solvers for the stages, providing their optimality.

In what follows, we will assume that the matrix $A_{\mathrm{RK}}$ is invertible.

**3.1. Preconditioner for the stages.** As we mentioned above, an all-at-once solve for a Runge–Kutta discretization in time may be performed only if one has an optimal preconditioner for the system of the stages

$$\underbrace{\begin{bmatrix} I_s \otimes \mathbf{M} + \tau A_{\mathrm{RK}} \otimes \mathbf{D}_1 \\ \tau A_{\mathrm{RK}} \otimes \mathbf{D}_2 \end{bmatrix}}_{\widehat{\Theta}} \begin{bmatrix} \mathbf{k}_{1,n} \\ \vdots \\ \mathbf{k}_{s,n} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_s \end{bmatrix}.$$

In order to derive a preconditioner for the matrix $\widehat{\Theta}$, we consider an SVD decomposition of the matrix $A_{\mathrm{RK}} = U\Sigma V^*$, where $U$ and $V$ are unitary matrices whose columns are the left and right singular vectors of $A_{\mathrm{RK}}$, respectively, and $\Sigma$ is a diagonal matrix with entries equal to the singular values of $A_{\mathrm{RK}}$. Note that this decomposition is not unique. Note also that since the matrix $A_{\mathrm{RK}}$ is real, the matrices $U$ and $V$ can be chosen to be real [11, Section 2.4], therefore they are properly orthogonal matrices. For this reason, we can write $A_{\mathrm{RK}} = U\Sigma V^\top$. From here, we can write

$$\begin{aligned} I_s \otimes \mathbf{M} + \tau A_{\mathrm{RK}} \otimes \mathbf{D}_1 &= I_s \otimes \mathbf{M} + \tau(U\Sigma V^\top) \otimes \mathbf{D}_1 \\ &= (U \otimes I_{n_x})[(U^\top V) \otimes \mathbf{M} + \tau\Sigma \otimes \mathbf{D}_1](V^\top \otimes I_{n_x}). \end{aligned}$$

Note that, since the matrices $U$ and $V$ are orthogonal, the same holds for the matrix $U^\top V$. In particular, the eigenvalues of the matrix $U^\top V$ all lie on the unit circle centered at the origin of the complex plane, and its eigenvectors are mutually orthogonal. Since all the eigenvalues have absolute value equal to 1, we can derive the following approximation:

$$\mathcal{P}_{\mathrm{RK}} := \begin{bmatrix} U \otimes I_{n_x} \\ U \otimes I_{n_x} \end{bmatrix} \begin{bmatrix} I_s \otimes \mathbf{M} + \tau\Sigma \otimes \mathbf{D}_1 \\ \tau\Sigma \otimes \mathbf{D}_2 \end{bmatrix} \begin{bmatrix} V^\top \otimes I_{n_x} \end{bmatrix} \approx \widehat{\Theta}.$$

This approximation can be employed as a preconditioner for the matrix $\widehat{\Theta}$ within the GMRES algorithm derived in [29]. Note that, excluding the effect of the inverses of the matrices $U \otimes I_{n_x}$ and $V^\top \otimes I_{n_x}$ (which require only a matrix–vector product), the $(1,1)$-block of the preconditioner is block-diagonal. Further, the matrices are all real, and we are not forced to work in complex arithmetic. Finally, we would like to mention that, compared to an eigendecomposition, by employing this strategy one is able to avoid possibly ill-conditioned matrices arising from the eigenvectors, for example. Despite the above properties holding, one cannot expect the approximation $\mathcal{P}_{\mathrm{RK}}$ to be completely robust. In fact, the matrix $U^\top V$ has $s$ distinct eigenvalues, therefore we expect the preconditioned matrix $\mathcal{P}_{\mathrm{RK}}^{-1}\widehat{\Theta}$ to have $s$ clusters of eigenvalues. Nonetheless, we expect the preconditioner to work robustly at least with respect to the mesh size $h$.

Below, we will show how to employ the preconditioner $\mathcal{P}_{\mathrm{RK}}$ for solving for the matrix $\widehat{\Theta}$.

**3.1.1. Heat equation.** Before specifying our strategy for the heat equation, we would like to introduce other preconditioners employed for solving for the stages of a Runge–Kutta discretization for this problem.

In [22], the authors approximate the matrix $I_s \otimes M + \tau A_{\mathrm{RK}} \otimes K$ with a fixed number of GMRES iteration, preconditioned with the following matrix:

$$\mathcal{P}_{\mathrm{MNS}} = \begin{bmatrix} M + \tau a_{1,1} K & & \\ & \ddots & \\ & & M + \tau a_{s,s} K \end{bmatrix}.$$

The preconditioner $\mathcal{P}_{\mathrm{MNS}}$ is optimal, in the sense that it can be proved that the condition number of the preconditioned system $\mathcal{P}_{\mathrm{MNS}}^{-1}(I_s \otimes M + \tau A_{\mathrm{RK}} \otimes K)$ is independent of the time-step $\tau$ and the mesh size $h$, see [22]. However, numerical experiments show that the condition number may be dependent on the number of stages $s$, see [22]. We would like to mention that other approaches may be employed, since the solver we propose for the all-at-once system is mainly based on a solver for the system of the stages of a Runge–Kutta method. For instance, in [31] the authors employ as a preconditioner the block-lower triangular part of the matrix $I_s \otimes M + \tau A_{\mathrm{RK}} \otimes K$, obtaining more robustness with respect to the number of stages $s$. Alternatively, one may employ the strategies described, for instance, in [1, 2, 27] as a preconditioner for the linear system considered.

In the numerical tests below, we compare our preconditioner $\mathcal{P}_{\mathrm{RK}}$ only with the preconditioner $\mathcal{P}_{\mathrm{MNS}}$. This is done for various reasons. In fact, although the methods presented in [31, 27] are robust, the preconditioners require one to solve for a block-lower triangular matrix. On the other hand, the robust preconditioner employed in [2] makes use of complex arithmetic, therefore the strategy requires one to solve for systems twice the dimension of each block in order to work with real arithmetic.

We can now describe the preconditioner $\mathcal{P}_{\mathrm{RK}}$ employed for solving for the stages of the discretization of the heat equation. The system for the stages is given by

$$I_s \otimes M + \tau A_{\mathrm{RK}} \otimes K.$$

In order to solve for this matrix, we employ GMRES with the following preconditioner:

$$\mathcal{P}_{\mathrm{RK}} = (U \otimes I_{n_x})(I_s \otimes M + \tau \Sigma \otimes K)(V^\top \otimes I_{n_x}). \tag{3.4}$$

The following theorem gives the optimality of the proposed preconditioner, under reasonable assumptions. More specifically, we require that the real part of the Rayleigh quotient $\frac{\mathbf{x}^*(U^\top V)\mathbf{x}}{\mathbf{x}^*\mathbf{x}}$ is positive, for any $\mathbf{x} \in \mathbb{C}^s$ with $\mathbf{x} \neq \mathbf{0}$. Note that, since the matrix $U^\top V$ is orthogonal (in particular, it is normal), this is equivalent to saying that the real part of the eigenvalues of the matrix $U^\top V$ is positive, since the set $\left\{\frac{\mathbf{x}^*(U^\top V)\mathbf{x}}{\mathbf{x}^*\mathbf{x}} | \mathbf{x} \in \mathbb{C}^s,\, \mathbf{x} \neq \mathbf{0}\right\}$ describes the *field of values* of the matrix $U^\top V$, and it represents the convex hull that contains the eigenvalues of this matrix. Note also that, again with $U^\top V$ orthogonal, our assumption is equivalent to saying that the ratio $\frac{\mathbf{x}^*(U^\top V)\mathbf{x}}{\mathbf{x}^*\mathbf{x}}$, with $\mathbf{x} \in \mathbb{C}^s \setminus \{\mathbf{0}\}$, is contained within the right-half of the unit circle centered at the origin of the complex plane. For these as well as other results on the field of values, we recommend the book [15].

REMARK 1. *Before moving to the statement and the proof of the eigenvalue result for the preconditioner we adopt, we would like to discuss the assumption we make. As we mentioned above, the SVD is not unique. However, under the assumption of $A_{\mathrm{RK}}$ being invertible, the product $U^\top V$ of the matrices containing the singular vectors is uniquely defined. In fact, an invertible matrix $A_{\mathrm{RK}}$ has a unique polar decomposition $A_{\mathrm{RK}} = \widehat{U}P$, with $\widehat{U}$ unitary and $P$ Hermitian positive-definite, see [13, Theorem 2.17]. Therefore, given $A_{\mathrm{RK}} = U\Sigma V^\top$ as an SVD of the matrix $A_{\mathrm{RK}}$, we clearly have $\widehat{U} = UV^\top$ and $P = V\Sigma V^\top$. From the uniqueness of the matrix $\widehat{U}$, we can derive that the product $U^\top V$ is uniquely defined.*

THEOREM 3.1. *Let be $A_{\mathrm{RK}}$ be the matrix representing the coefficients of a Runge–Kutta method. Let $A_{\mathrm{RK}} = U\Sigma V^\top$ be an SVD of the matrix $A_{\mathrm{RK}}$. Suppose that the real part of Rayleigh quotient $\frac{\mathbf{x}^*(U^\top V)\mathbf{x}}{\mathbf{x}^*\mathbf{x}}$ is positive, for any $\mathbf{x} \in \mathbb{C}^s \setminus \{\mathbf{0}\}$. Then, the eigenvalues of the matrix $\mathcal{P}_{\mathrm{RK}}^{-1}(I_s \otimes M + \tau A_{\mathrm{RK}} \otimes K)$ all lie in the right-half of the unit circle centered at the origin of the complex plane.*

*Proof.* Let $\lambda$ be an eigenvalue of the matrix $\mathcal{P}_{\mathrm{RK}}^{-1}(I_s \otimes M + \tau A_{\mathrm{RK}} \otimes K)$, with $\mathbf{x}$ the corresponding eigenvector. Then, we have

$$(I_s \otimes M + \tau A_{\mathrm{RK}} \otimes K)\mathbf{x} = \lambda \mathcal{P}_{\mathrm{RK}}\mathbf{x}.$$

By employing the SVD of the matrix $A_{\mathrm{RK}}$ and a well known property of the Kronecker product, we can write

$$(U \otimes I_{n_x})((U^\top V) \otimes M + \tau\Sigma \otimes K)(V^\top \otimes I_{n_x})\mathbf{x} = \lambda \mathcal{P}_{\mathrm{RK}}\mathbf{x}.$$

From (3.4), by setting $\mathbf{y} = (V^\top \otimes I_{n_x})\mathbf{x}$, the previous expression is equivalent to

$$((U^\top V) \otimes M + \tau\Sigma \otimes K)\mathbf{y} = \lambda(I_s \otimes M + \tau\Sigma \otimes K)\mathbf{y}.$$

Recalling that the matrix $M$ is s.p.d., we can write $M^{\frac{1}{2}}$. Then, we have

$$((U^\top V) \otimes I_{n_x} + \tau\Sigma \otimes (M^{-\frac{1}{2}}KM^{-\frac{1}{2}}))\mathbf{z} = \lambda(I_s \otimes I_{n_x} + \tau\Sigma \otimes (M^{-\frac{1}{2}}KM^{-\frac{1}{2}}))\mathbf{z},$$

where $\mathbf{z} = (I_s \otimes M^{\frac{1}{2}})\mathbf{y}$.

Since $\Sigma$ is s.p.d. and $K$ is symmetric positive semi-definite, we have that the matrix $I_s \otimes I_{n_x} + \tau\Sigma \otimes (M^{-\frac{1}{2}}KM^{-\frac{1}{2}})$ is s.p.d., therefore invertible. Thus, we can consider the generalized Rayleigh quotient

$$\begin{aligned}\lambda &= \frac{\mathbf{z}^*((U^\top V) \otimes I_{n_x} + \tau\Sigma \otimes (M^{-\frac{1}{2}}KM^{-\frac{1}{2}}))\mathbf{z}}{\mathbf{z}^*(I_s \otimes I_{n_x} + \tau\Sigma \otimes (M^{-\frac{1}{2}}KM^{-\frac{1}{2}}))\mathbf{z}} \\ &= \frac{1}{1+\tau\hat{\lambda}}\left(\frac{\mathbf{z}^*((U^\top V) \otimes I_{n_x})\mathbf{z}}{\mathbf{z}^*\mathbf{z}} + \tau\hat{\lambda}\right), \end{aligned} \tag{3.5}$$

with $\hat{\lambda} = \frac{\mathbf{z}^*(\Sigma \otimes (M^{-\frac{1}{2}} K M^{-\frac{1}{2}}))\mathbf{z}}{\mathbf{z}^*\mathbf{z}}$.

Again, since the matrix $\Sigma \otimes (M^{-\frac{1}{2}} K M^{-\frac{1}{2}})$ is symmetric positive semi-definite, the value $\hat{\lambda}$ is real and non-negative. Then, since $\tau > 0$, we have $1 + \tau\hat{\lambda} \geq 1$, and $\frac{\tau\hat{\lambda}}{1+\tau\hat{\lambda}} \leq 1$.

Denoting here with $i$ the imaginary unit, under our assumption we can write

$$\frac{\mathbf{z}^*((U^\top V) \otimes I_{n_x})\mathbf{z}}{\mathbf{z}^*\mathbf{z}} = a + ib,$$

with $0 < a \leq 1$, and $b \in [-1, 1]$ such that $a^2 + b^2 \leq 1$, due to $U^\top V$ being orthogonal. In fact, we have

$$\frac{\mathbf{z}^*((U^\top V) \otimes I_{n_x})\mathbf{z}}{\mathbf{z}^*\mathbf{z}} = \frac{\hat{\mathbf{z}}^*(I_{n_x} \otimes (U^\top V))\hat{\mathbf{z}}}{\hat{\mathbf{z}}^*\hat{\mathbf{z}}},$$

with $\hat{\mathbf{z}} = P\mathbf{z}$ for a suitable permutation $P$. In particular, we have that the fields of values $\left\{\frac{\mathbf{x}^*(U^\top V)\mathbf{x}}{\mathbf{x}^*\mathbf{x}} | \mathbf{x} \in \mathbb{C}^s,\, \mathbf{x} \neq \mathbf{0}\right\}$ and $\left\{\frac{\mathbf{z}^*((U^\top V)\otimes I_{n_x})\mathbf{z}}{\mathbf{z}^*\mathbf{z}} | \mathbf{z} \in \mathbb{C}^{sn_x},\, \mathbf{z} \neq \mathbf{0}\right\}$ describe the same subset of the complex plane. Since $a > 0$, from (3.5) we can say that the real part of $\lambda$ is greater than 0. Finally, from $a^2 + b^2 \leq 1$ and (3.5) we can derive that the absolute value of $\lambda$ is less or equal than 1, that is, $\lambda$ lies in the right-half of the unit circle centered at the origin of the complex plane. In fact, we have

$$\begin{aligned}
|\lambda|^2 = \lambda\lambda^* &= \frac{1}{(1+\tau\hat{\lambda})^2}\left[(a+\tau\hat{\lambda})^2 + b^2\right] = \frac{1}{(1+\tau\hat{\lambda})^2}\left[a^2 + b^2 + (\tau\hat{\lambda})^2 + 2a\tau\hat{\lambda}\right] \\
&\leq \frac{1}{(1+\tau\hat{\lambda})^2}\left[1 + (\tau\hat{\lambda})^2 + 2a\tau\hat{\lambda}\right] \leq \frac{1}{(1+\tau\hat{\lambda})^2}\left[1 + (\tau\hat{\lambda})^2 + 2\tau\hat{\lambda}\right] = 1.
\end{aligned}$$

Since $\lambda$ is an eigenvalue of the matrix $\mathcal{P}_{\mathrm{RK}}^{-1}(I_s \otimes M + \tau A_{\mathrm{RK}} \otimes K)$, the above gives the desired result.
□

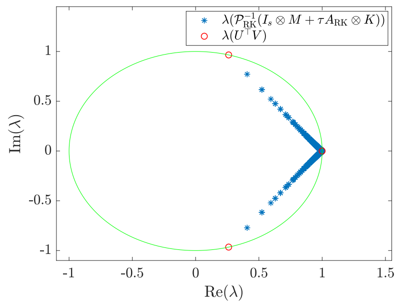
In Figure 3.1, we report the eigenvalue distributions of the matrices $\mathcal{P}_{\mathrm{RK}}^{-1}(I_s \otimes M + \tau A_{\mathrm{RK}} \otimes K)$ and $U^\top V$, employing $\mathbf{Q}_2$ elements, for 3-stages Gauss, 4-stages Lobatto IIIC, 5-stages Radau IIA, and 9-stages Radau IIA methods, with $\tau = 0.2$ and level of refinement $\mathtt{l} = 4$. Here, $\mathtt{l}$ represents a spatial uniform grid of mesh size $h = 2^{-1}$, in each dimension. Further, in green we plot the unit circle centered at the origin of the complex plane.

Interestingly, as observed in Figure 3.1, the eigenvalues of $\mathcal{P}_{\mathrm{RK}}^{-1}(I_s \otimes M + \tau A_{\mathrm{RK}} \otimes K)$ all lie in, or near, segments that join 1 to an eigenvalue of $U^\top V$. We are not able to explain this behavior using the analysis of Theorem 3.1. However, we would like to note that, in practice, the eigenvalues seem to locate away from 0. In particular, we cannot expect 0 to be an eigenvalue of $\mathcal{P}_{\mathrm{RK}}^{-1}(I_s \otimes M + \tau A_{\mathrm{RK}} \otimes K)$, since both $\mathcal{P}_{\mathrm{RK}}$ and $I_s \otimes M + \tau A_{\mathrm{RK}} \otimes K$ are invertible.

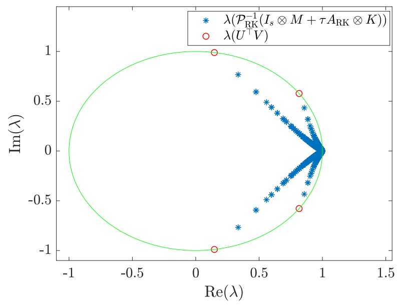
We would like to mention that, for all the Runge–Kutta methods we employ (excluding the 5-stages Radau IIA), the real part of the eigenvalues of the matrix $U^\top V$ was positive, thus the assumption of Theorem 3.1 is not excessively restrictive[2]. Further, following the sketch of the proof above, one can understand that we are able to derive this result about the preconditioner only because the matrix $K$ is symmetric positive semi-definite. For this reason, we cannot expect this property of the preconditioner to hold for more general problems.

[2]For the 5-stages Radau IIA method, the eigenvalues of the matrix $U^\top V$ are negative, but very close to 0, as seen from Figure 3.1.

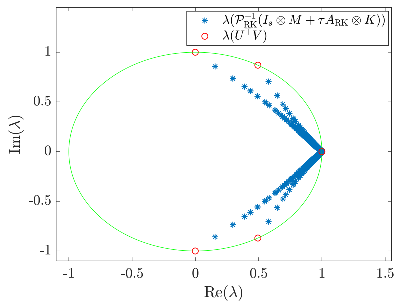
Fig. 3.1: Eigenvalue distributions of $\mathcal{P}_{\mathrm{RK}}^{-1}(I_s \otimes M + \tau A_{\mathrm{RK}} \otimes K)$ and of $U^{\top}V$, for 3-stages Gauss, 4-stages Lobatto IIIC, 5-stages Radau IIA, and 9-stages Radau IIA methods, with $\tau = 0.2$, and $\mathtt{l} = 4$. In green, we plot the unit circle centered at the origin of the complex plane.



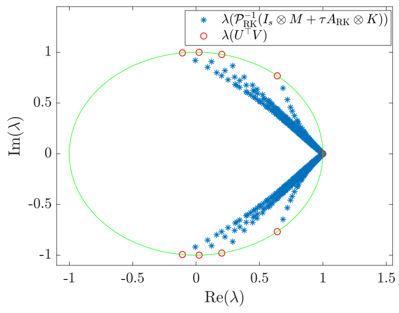
(a) 3-stages Gauss.



(b) 4-stages Lobatto IIIC.



(c) 5-stages Radau IIA.



(d) 9-stages Radau IIA.

**3.1.2. Stokes equations.** In this section we derive a preconditioner for the block $\widehat{\Theta}$ defined in (2.13).

We recall that the matrix $\widehat{\Theta}$ is given by

$$\widehat{\Theta} = \begin{bmatrix} I_s \otimes M_v + \tau A_{\mathrm{RK}} \otimes K_v & \tau A_{\mathrm{RK}} \otimes B^{\top} \\ \tau A_{\mathrm{RK}} \otimes B & 0 \end{bmatrix}.$$

In order to solve for this system, we employ as a preconditioner

$$\mathcal{P}_{\mathrm{RK}} = \begin{bmatrix} I_s \otimes M_v + \tau A_{\mathrm{RK}} \otimes K_v & 0 \\ \tau A_{\mathrm{RK}} \otimes B & S_{\mathrm{RK}} \end{bmatrix},$$

where

$$S_{\mathrm{RK}} = -\tau^2 (A_{\mathrm{RK}} \otimes B)(I_s \otimes M_v + \tau A_{\mathrm{RK}} \otimes K_v)^{-1}(A_{\mathrm{RK}} \otimes B^{\top}). \tag{3.6}$$

The $(1,1)$-block $I_s \otimes M_v + \tau A_{\mathrm{RK}} \otimes K_v$ can be dealt with as for the heat equation. More specifically, we approximate it with $(U \otimes I_{n_v})(I_s \otimes M_v + \tau \Sigma \otimes K_v)(V^{\top} \otimes I_{n_v})$.

In order to find a suitable approximation of the Schur complement $S_{\mathrm{RK}}$, we observe that

$$A_{\mathrm{RK}} \otimes B = (A_{\mathrm{RK}} \otimes I_{n_p})(I_s \otimes B),$$
$$A_{\mathrm{RK}} \otimes B^\top = (I_s \otimes B^\top)(A_{\mathrm{RK}} \otimes I_{n_p}).$$

It is clear that if we find a suitable approximation of the following matrix:

$$\widetilde{S}_{\mathrm{int}} \approx S_{\mathrm{int}} = (I_s \otimes B)(I_s \otimes M_v + \tau A_{\mathrm{RK}} \otimes K_v)^{-1}(I_s \otimes B^\top),$$

then a suitable approximation of the Schur complement $S_{\mathrm{RK}}$ is given by

$$\widetilde{S}_{\mathrm{RK}} := -\tau^2 (A_{\mathrm{RK}} \otimes I_{n_p})\widetilde{S}_{\mathrm{int}}(A_{\mathrm{RK}} \otimes I_{n_p}). \tag{3.7}$$

Note that the matrix $(A_{\mathrm{RK}} \otimes I_{n_p})$ can be easily inverted by making use of the SVD of the matrix $A_{\mathrm{RK}} = U\Sigma V^\top$.

In order to derive a suitable approximation of the matrix $S_{\mathrm{int}}$, we employ the block-commutator argument derived in [19]. We would like to mention that a similar approach has been derived independently and employed for a parallel-in-time solver for the incompressible Navier–Stokes equations by the authors in [5]. The approximation we employ is given by

$$\widetilde{S}_{\mathrm{int}} := (I_s \otimes K_p)(I_s \otimes M_p + \tau A_{\mathrm{RK}} \otimes K_p)^{-1}(I_s \otimes M_p).$$

Then, our approximation of the Schur complement $S_{\mathrm{RK}}$ is given by (3.7) with this choice of $\widetilde{S}_{\mathrm{int}}$. For details on the derivation of the approximation, we refer to [19].

In Figure 3.2, we report the eigenvalues of the matrix $\widetilde{S}_{\mathrm{RK}}^{-1} S_{\mathrm{RK}}$, for 3-stages Gauss, 3-stages Lobatto IIIC, and 3-stages Radau IIA methods, with $\tau = 0.2$, and level of refinement $\mathtt{l} = 5$. Here, $\mathtt{l}$ represents a (spatial) uniform grid of mesh size $h = 2^{1-\mathtt{l}}$ for $\mathbf{Q}_1$ basis functions, and $h = 2^{-\mathtt{l}}$ for $\mathbf{Q}_2$ elements, in each dimension. Since for the problem we are considering the matrix $K_p$ is not invertible, we derive an invertible approximation $\widetilde{S}_{\mathrm{RK}}$ by "pinning" the value of one of the nodes of the matrix $K_p$, for each $K_p$ within the definition of $\widetilde{S}_{\mathrm{int}}$.

Finally, we can present our approximation $\widetilde{\mathcal{P}}_{\mathrm{RK}}$ of the preconditioner $\mathcal{P}_{\mathrm{RK}}$ for the matrix $\widehat{\Theta}$. By employing a well known property of the Kronecker product, the approximation of the preconditioner $\mathcal{P}_{\mathrm{RK}}$ is given by the following

$$\widetilde{\mathcal{P}}_{\mathrm{RK}} = \begin{bmatrix} U \otimes I_{n_v} & 0 \\ 0 & U \otimes I_{n_p} \end{bmatrix} \widetilde{\mathcal{P}}_{\mathrm{int}} \begin{bmatrix} V^\top \otimes I_{n_v} & 0 \\ 0 & V^\top \otimes I_{n_p} \end{bmatrix}, \tag{3.8}$$

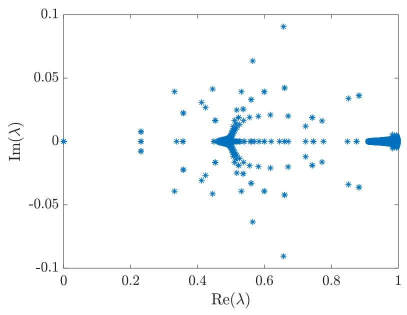
with

$$\widetilde{\mathcal{P}}_{\mathrm{int}} = \begin{bmatrix} I_s \otimes M + \tau\Sigma \otimes K & 0 \\ \tau\Sigma \otimes B & -\tau^2\left((\Sigma V^\top) \otimes I_{n_p}\right)\widetilde{S}_{\mathrm{int}}\left((U\Sigma) \otimes I_{n_p}\right) \end{bmatrix}.$$

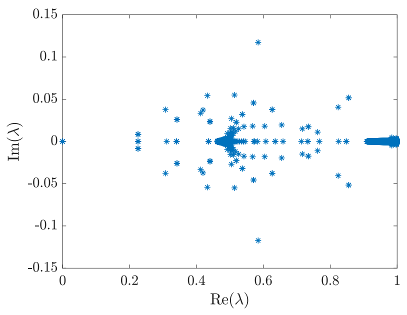
Before completing this section, we would like to mention that the matrix $\widehat{\Theta}$ is not invertible. For this reason, the block $\Theta = I_{n_t} \otimes \widehat{\Theta}$ is not invertible, and one cannot employ the preconditioner defined in (3.1) as it is. We thus rather employ as $(2,2)$-block the matrix $\widetilde{\Theta} = I_{n_t} \otimes \widetilde{\Theta}$, with $\widetilde{\Theta}$ given by the following invertible perturbation of $\widehat{\Theta}$:

$$\widetilde{\Theta} = \begin{bmatrix} I_s \otimes M_v + \tau A_{\mathrm{RK}} \otimes K_v & \tau A_{\mathrm{RK}} \otimes B^\top \\ \tau A_{\mathrm{RK}} \otimes B & -\tau^2\gamma(A_{\mathrm{RK}}^2 \otimes M_p) \end{bmatrix},$$

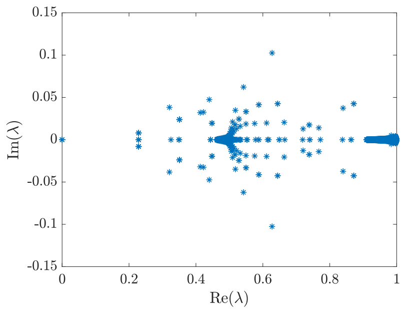
Fig. 3.2: Commutator approximation for the Stokes equations. Eigenvalues of $\widetilde{S}_{\text{RK}}^{-1} S_{\text{RK}}$, for 3-stages Gauss, 3-stages Lobatto IIIC, and 3-stages Radau IIA methods, with $\tau = 0.2$, and $\mathtt{l} = 5$.



(a) 3-stages Gauss.



(b) 3-stages Lobatto IIIC.



(c) 3-stages Radau IIA.

where $0 < \gamma \ll 1$. The parameter $\gamma$ has to be chosen such that the perturbed matrix $\widetilde{\Theta}$ is invertible, but "close" enough to the original $\widehat{\Theta}$. In our test, we chose $\gamma = 10^{-4}$. Note that we chose this perturbation of the $(2,2)$-block for the expression of the Schur complement $S_{\text{RK}}$ given in (3.6). In fact, it is easy to see that

$$-\tau^2\gamma(A_{\text{RK}}^2 \otimes M_p) = -\tau^2(A_{\text{RK}} \otimes I_{n_p})(\gamma I_s \otimes M_p)(A_{\text{RK}} \otimes I_{n_p}),$$

implying that we perturb $S_{\text{int}}$ only by the matrix $\gamma I_s \otimes M_p$.

**4. Numerical results.** We now provide numerical evidence of the effectiveness of our preconditioning strategy. All tests are run in sequential on MATLAB R2018b, using a 1.70GHz Intel quad-core i5 processor and 8 GB RAM on an Ubuntu 18.04.1 LTS operating system. A future version of this work will consider a parallel implementation of the proposed solver.

In all our tests, $d = 2$ (that is, $\mathbf{x} = (x_1, x_2)$), and $\Omega = (-1, 1)^2$. We employ $\mathbf{Q}_1$ and $\mathbf{Q}_2$ finite element basis functions in the spatial dimensions for the heat equation, while employing inf–sup stable Taylor–Hood $\mathbf{Q}_2$–$\mathbf{Q}_1$ for the Stokes equations. In all our tests, the level of refinement $l$ represents a (spatial) uniform grid of mesh size $h = 2^{1-l}$ for $\mathbf{Q}_1$ basis functions, and $h = 2^{-l}$ for $\mathbf{Q}_2$ elements, in each dimension. Regarding the time grid, denoting with $q_{\text{FE}}$ the order of the finite element approximations and with $q_{\mathrm{RK}}$ the order of the Runge–Kutta method we employ, we chose as number of intervals in time $n_t$ the closest integer such that $\tau \leq h^{q_{\mathrm{FE}}/q_{\mathrm{RK}}}$, where $\tau = t_f/n_t$ is the time step.

In all our tests, we apply 20 steps of Chebyshev semi-iteration to mass matrices, while employing 2 V-cycles (with 2 symmetric Gauss–Seidel iterations for pre-/post-smoothing) of the `HSL_MI20` solver [3] for other matrices.

All CPU times below are reported in seconds.

**4.1. Heat equation: sequential time-stepping.** In this section, we compare our SVD-based preconditioner with the block-diagonal preconditioner $\mathcal{P}_{\mathrm{MNS}}$ derived in [22]. We refer the reader to Section 3.1.1 for this choice of preconditioner. We evaluate the numerical solution in a classical way, solving sequentially for the time step $n$, that is, we solve for the system of the stages given in (2.8) and then update the solution at time $t_{n+1}$ with (2.7). We compare the number of iterations required for convergence up to a tolerance $10^{-8}$ on the relative residual for both methods. We employ GMRES as the Krylov solver, with a restart every 10 iterations.

We take $t_f = 2$, and solve the problem (2.6) with solution given by

$$v(\mathbf{x}, t) = e^{t_f - t} \cos\left(\frac{\pi x_1}{2}\right) \cos\left(\frac{\pi x_2}{2}\right) + 1,$$

with initial and boundary conditions obtained from this $v$. In Tables 4.1–4.4, we report the average number of GMRES iterations and the total CPU times for solving all the linear systems together with the numerical errors $v_{\mathrm{error}}$ on the solution, for different Runge–Kutta methods. We evaluate the numerical error $v_{\mathrm{error}}$ in the scaled vector $\ell^\infty$-norm, defined as

$$v_{\mathrm{error}} = \max_n \left\{ \frac{|v_{\mathtt{j},n} - v^{\mathrm{sol}}_{\mathtt{j},n}|}{|v^{\mathrm{sol}}_{\mathtt{j},n}|}, \text{ with } \mathtt{j} = \arg\max_j |v_{j,n} - v^{\mathrm{sol}}_{j,n}| \right\},$$

where $v_{j,n}$ and $v^{\mathrm{sol}}_{j,n}$ are the entries of the computed solution $\mathbf{v}$ and the (discretized) exact solution for $v$ at time $t = t_n$. Further, in Tables 4.3–4.4 we report the dimension of the system solved for Radau IIA methods when employing $\mathbf{Q}_1$ and $\mathbf{Q}_2$ finite elements, respectively. We would like to note that those values coincide with the dimension of the system arising when employing the corresponding level of refinement of an $s$-stages Gauss or Lobatto IIIC method.

Tables 4.1–4.4 show a very mild dependence of our SVD-based preconditioner with respect to the mesh size $h$ and the number of stages $s$. Nonetheless, the preconditioner we propose is able to reach convergence in less than 25 iterations. Although the block-diagonal preconditioner $\mathcal{P}_{\mathrm{MNS}}$ is optimal with respect to the discretization parameters, the number of iterations required to reach the prescribed tolerance is not robust with respect to the number of stages $s$. For instance, in order to reach convergence the solver needs almost 60 iterations for the 5-stages Lobatto IIIC method. On the other hand, our solver does not suffer (drastically) from this dependence. In fact, when employing a high number of stages, our solver can be between 2 and 3 times faster. Finally, we would like to note that the error behaves as predicted, that is, the method behaves as a second- and third-order method with $\mathbf{Q}_1$ and $\mathbf{Q}_2$ finite elements, respectively. In particular, we do not experience any order reduction for the numerical tests carried out in this work, aside the pollution of the linear residual on the numerical solution.

Table 4.1: Sequential solve of the heat equation: average GMRES iterations, total CPU times, and resulting relative errors in $v$ for Gauss methods, employing $\mathbf{Q}_1$ and $\mathbf{Q}_2$ finite elements.

| | | $\mathbf{Q}_1$ | | | | | | $\mathbf{Q}_2$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\mathcal{P}_{\text{MNS}}$ | | | $\mathcal{P}_{\text{RK}}$ | | | $\mathcal{P}_{\text{MNS}}$ | | | $\mathcal{P}_{\text{RK}}$ | | |
| $s$ | `l` | `it` | CPU | $v_{\text{error}}$ | `it` | CPU | $v_{\text{error}}$ | `it` | CPU | $v_{\text{error}}$ | `it` | CPU | $v_{\text{error}}$ |
| 2 | 3 | 10 | 0.05 | 6.35e-03 | 8 | 0.05 | 6.35e-03 | 10 | 0.1 | 1.33e-04 | 8 | 0.1 | 1.33e-04 |
| | 4 | 10 | 0.1 | 1.69e-03 | 8 | 0.09 | 1.69e-03 | 10 | 0.4 | 1.62e-05 | 9 | 0.4 | 1.62e-05 |
| | 5 | 10 | 0.3 | 4.55e-04 | 8 | 0.2 | 4.55e-04 | 10 | 2.2 | 2.39e-06 | 10 | 2.2 | 2.39e-06 |
| | 6 | 10 | 1.3 | 1.14e-04 | 9 | 1.2 | 1.14e-04 | 10 | 16 | 2.91e-07 | 12 | 19 | 2.91e-07 |
| | 7 | 10 | 6.7 | 2.95e-05 | 10 | 6.8 | 2.95e-05 | 10 | 116 | 3.44e-08 | 13 | 167 | 3.45e-08 |
| 3 | 3 | 18 | 0.2 | 5.45e-03 | 12 | 0.1 | 5.45e-03 | 18 | 0.2 | 5.10e-05 | 11 | 0.2 | 5.10e-05 |
| | 4 | 17 | 0.2 | 1.53e-03 | 11 | 0.1 | 1.53e-03 | 18 | 0.6 | 3.43e-06 | 13 | 0.5 | 3.43e-06 |
| | 5 | 16 | 0.5 | 4.17e-04 | 12 | 0.4 | 4.17e-04 | 18 | 3.1 | 2.36e-07 | 15 | 2.5 | 2.36e-07 |
| | 6 | 16 | 1.8 | 1.07e-04 | 12 | 1.6 | 1.07e-04 | 17 | 20 | 1.69e-08 | 17 | 18 | 1.66e-08 |
| | 7 | 16 | 8.0 | 2.71e-05 | 13 | 7.0 | 2.71e-05 | 17 | 107 | 1.95e-09 | 18 | 122 | 1.78e-09 |

Table 4.2: Sequential solve of the heat equation: average GMRES iterations, total CPU times, and resulting relative errors in $v$ for Lobatto IIIC methods, employing $\mathbf{Q}_1$ and $\mathbf{Q}_2$ finite elements.

| | | $\mathbf{Q}_1$ | | | | | | $\mathbf{Q}_2$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\mathcal{P}_{\text{MNS}}$ | | | $\mathcal{P}_{\text{RK}}$ | | | $\mathcal{P}_{\text{MNS}}$ | | | $\mathcal{P}_{\text{RK}}$ | | |
| $s$ | `l` | `it` | CPU | $v_{\text{error}}$ | `it` | CPU | $v_{\text{error}}$ | `it` | CPU | $v_{\text{error}}$ | `it` | CPU | $v_{\text{error}}$ |
| 2 | 3 | 12 | 0.1 | 1.36e-02 | 7 | 0.09 | 1.36e-02 | 13 | 0.5 | 2.55e-03 | 9 | 0.3 | 2.55e-03 |
| | 4 | 12 | 0.3 | 4.10e-03 | 8 | 0.2 | 4.10e-03 | 14 | 2.8 | 3.66e-04 | 10 | 1.9 | 3.66e-04 |
| | 5 | 12 | 1.4 | 1.14e-03 | 9 | 0.9 | 1.14e-03 | 13 | 23 | 5.06e-05 | 12 | 22 | 5.06e-05 |
| | 6 | 12 | 8.7 | 3.01e-04 | 10 | 7.1 | 3.01e-04 | 13 | 276 | 6.45e-06 | 12 | 263 | 6.45e-06 |
| | 7 | 12 | 66 | 7.77e-05 | 12 | 68 | 7.77e-05 | 12 | 3226 | 8.18e-07 | 12 | 3377 | 8.17e-07 |
| 3 | 3 | 26 | 0.2 | 5.42e-03 | 8 | 0.07 | 5.42e-03 | 25 | 0.5 | 1.15e-04 | 10 | 0.2 | 1.15e-04 |
| | 4 | 25 | 0.3 | 1.48e-03 | 9 | 0.1 | 1.48e-03 | 26 | 1.6 | 1.75e-05 | 11 | 0.7 | 1.75e-05 |
| | 5 | 25 | 0.9 | 3.79e-04 | 10 | 0.4 | 3.79e-04 | 29 | 9.5 | 2.95e-06 | 12 | 4.1 | 2.95e-06 |
| | 6 | 25 | 4.7 | 9.82e-05 | 11 | 2.2 | 9.82e-05 | 28 | 64 | 3.90e-07 | 15 | 37 | 3.89e-07 |
| | 7 | 26 | 25 | 2.43e-05 | 12 | 13 | 2.43e-05 | 28 | 490 | 4.82e-08 | 17 | 330 | 4.81e-08 |
| 4 | 3 | 40 | 0.4 | 5.75e-03 | 12 | 0.1 | 5.75e-03 | 41 | 0.6 | 2.37e-05 | 13 | 0.2 | 2.37e-05 |
| | 4 | 37 | 0.4 | 1.55e-03 | 11 | 0.1 | 1.55e-03 | 42 | 2.0 | 1.65e-06 | 15 | 0.7 | 1.66e-06 |
| | 5 | 37 | 1.4 | 4.18e-04 | 13 | 0.5 | 4.18e-04 | 44 | 9.2 | 1.63e-07 | 16 | 3.5 | 1.61e-07 |
| | 6 | 38 | 5.4 | 1.07e-04 | 15 | 2.4 | 1.07e-04 | 44 | 60 | 1.33e-08 | 18 | 27 | 1.28e-08 |
| | 7 | 39 | 25 | 2.73e-05 | 16 | 11 | 2.73e-05 | 43 | 368 | 2.18e-09 | 18 | 171 | 2.32e-09 |
| 5 | 3 | 56 | 0.5 | 5.91e-03 | 15 | 0.2 | 5.91e-03 | 55 | 1.0 | 1.94e-05 | 15 | 0.3 | 1.94e-05 |
| | 4 | 52 | 0.8 | 1.55e-03 | 15 | 0.2 | 1.55e-03 | 56 | 2.8 | 1.19e-06 | 17 | 0.9 | 1.19e-06 |
| | 5 | 51 | 1.8 | 4.07e-04 | 14 | 0.5 | 4.07e-04 | 59 | 12 | 7.31e-08 | 18 | 3.7 | 7.39e-08 |
| | 6 | 53 | 6.9 | 1.07e-04 | 16 | 2.3 | 1.07e-04 | 57 | 67 | 4.93e-09 | 21 | 27 | 3.95e-09 |
| | 7 | 54 | 34 | 2.71e-05 | 17 | 11 | 2.71e-05 | 57 | 389 | 9.56e-10 | 22 | 162 | 7.53e-09 |

**4.2. Heat equation: sequential all-at-once solve.** In this section, we present the numerical results for the all-at-once Runge–Kutta discretization of the heat equation given in (2.9). The problem we consider here is the same as the one in Section 4.1. We employ as a preconditioner the matrix $\mathcal{P}$ given in (3.1), with the matrix $\widehat{S}$ defined in (3.2) approximated with a block-forward substitution. Within this process, the matrix $\widehat{X}$ defined in (3.3) is applied inexactly through an approximate inversion of the block for the stages. More specifically, given the results of the previous section, we approximately invert each block for the stages with 5 GMRES iterations precon-

Table 4.3: Sequential solve of the heat equation: degrees of freedom (DoF), average GMRES iterations, total CPU times, and resulting relative errors in $v$ for Radau IIA methods, employing $\mathbf{Q}_1$ finite elements.

| | | | $\mathcal{P}_{\text{MNS}}$ | | | $\mathcal{P}_{\text{RK}}$ | | |
|---|---|---|---|---|---|---|---|---|
| $s$ | `l` | DoF | `it` | CPU | $v_{\text{error}}$ | `it` | CPU | $v_{\text{error}}$ |
| 2 | 3 | 98 | 11 | 0.09 | 5.48e-03 | 8 | 0.08 | 5.48e-03 |
| | 4 | 450 | 10 | 0.1 | 1.39e-03 | 8 | 0.08 | 1.39e-03 |
| | 5 | 1922 | 12 | 0.5 | 3.67e-04 | 9 | 0.4 | 3.67e-04 |
| | 6 | 7938 | 12 | 2.9 | 9.44e-05 | 10 | 2.3 | 9.44e-05 |
| | 7 | 32,258 | 12 | 17 | 2.35e-05 | 11 | 16 | 2.34e-05 |
| 3 | 3 | 147 | 21 | 0.1 | 5.71e-03 | 10 | 0.07 | 5.71e-03 |
| | 4 | 675 | 19 | 0.2 | 1.60e-03 | 10 | 0.1 | 1.60e-03 |
| | 5 | 2883 | 19 | 0.6 | 4.17e-04 | 11 | 0.4 | 4.17e-04 |
| | 6 | 11,907 | 20 | 2.6 | 1.08e-04 | 12 | 1.6 | 1.08e-04 |
| | 7 | 48,387 | 20 | 14 | 2.74e-05 | 13 | 9.3 | 2.74e-05 |
| 4 | 3 | 196 | 30 | 0.2 | 5.59e-03 | 12 | 0.08 | 5.59e-03 |
| | 4 | 900 | 29 | 0.4 | 1.55e-03 | 12 | 0.2 | 1.55e-03 |
| | 5 | 3844 | 27 | 1.0 | 4.18e-04 | 15 | 0.5 | 4.18e-04 |
| | 6 | 15,876 | 27 | 3.6 | 1.07e-04 | 16 | 2.1 | 1.07e-04 |
| | 7 | 64,516 | 28 | 17 | 2.70e-05 | 17 | 10 | 2.70e-05 |
| 5 | 3 | 245 | 40 | 0.4 | 5.91e-03 | 16 | 0.1 | 5.91e-03 |
| | 4 | 1125 | 38 | 0.6 | 1.55e-03 | 16 | 0.2 | 1.55e-03 |
| | 5 | 4805 | 36 | 1.2 | 4.07e-04 | 15 | 0.5 | 4.07e-04 |
| | 6 | 19,845 | 36 | 4.7 | 1.07e-04 | 17 | 2.2 | 1.07e-04 |
| | 7 | 80,645 | 37 | 25 | 2.71e-05 | 18 | 12 | 2.71e-05 |

Table 4.4: Sequential solve of the heat equation: degrees of freedom (DoF), average GMRES iterations, total CPU times, and resulting relative errors in $v$ for Radau IIA methods, employing $\mathbf{Q}_2$ finite elements.

| | | | $\mathcal{P}_{\text{MNS}}$ | | | $\mathcal{P}_{\text{RK}}$ | | |
|---|---|---|---|---|---|---|---|---|
| $s$ | `l` | DoF | `it` | CPU | $v_{\text{error}}$ | `it` | CPU | $v_{\text{error}}$ |
| 2 | 3 | 450 | 12 | 0.2 | 2.10e-04 | 8 | 0.2 | 2.10e-04 |
| | 4 | 1922 | 12 | 0.8 | 2.86e-05 | 10 | 0.6 | 2.86e-05 |
| | 5 | 7938 | 12 | 5.4 | 3.76e-06 | 11 | 5.1 | 3.76e-06 |
| | 6 | 32,258 | 12 | 45 | 4.82e-07 | 14 | 52 | 4.82e-07 |
| | 7 | 130,050 | 12 | 400 | 6.11e-08 | 14 | 496 | 6.11e-08 |
| 3 | 3 | 675 | 20 | 0.3 | 2.62e-05 | 11 | 0.1 | 2.62e-05 |
| | 4 | 2883 | 23 | 0.9 | 2.61e-06 | 12 | 0.5 | 2.61e-06 |
| | 5 | 11,907 | 22 | 4.7 | 2.31e-07 | 15 | 3.3 | 2.31e-07 |
| | 6 | 48,387 | 23 | 32 | 2.94e-08 | 17 | 24 | 2.96e-08 |
| | 7 | 195,075 | 22 | 215 | 2.95e-09 | 18 | 183 | 2.85e-09 |
| 4 | 3 | 900 | 28 | 0.4 | 1.94e-05 | 15 | 0.2 | 1.94e-05 |
| | 4 | 3844 | 30 | 1.1 | 1.17e-06 | 16 | 0.6 | 1.17e-06 |
| | 5 | 15,876 | 30 | 5.3 | 7.25e-08 | 17 | 3.2 | 7.14e-08 |
| | 6 | 64,516 | 30 | 31 | 5.29e-09 | 18 | 19 | 3.43e-09 |
| | 7 | 260,100 | 30 | 203 | 9.39e-10 | 19 | 126 | 1.20e-09 |
| 5 | 3 | 1125 | 38 | 0.7 | 1.92e-05 | 16 | 0.3 | 1.91e-05 |
| | 4 | 4805 | 40 | 1.6 | 1.20e-06 | 16 | 0.7 | 1.20e-06 |
| | 5 | 19,845 | 42 | 8.3 | 7.37e-08 | 19 | 3.9 | 7.39e-08 |
| | 6 | 80,645 | 40 | 41 | 4.67e-09 | 21 | 23 | 4.05e-09 |
| | 7 | 325,125 | 40 | 235 | 1.15e-09 | 22 | 130 | 7.42e-09 |

ditioned with the matrix $\mathcal{P}_{\text{RK}}$ defined in (3.4). This process is also employed for approximating the $(2, 2)$-block $\Theta$. Since the inner iteration is based on a fixed number of GMRES iterations, we have to employ the flexible version of GMRES derived in [28] as an outer solver. As above, we restart FGMRES after every 10 iterations. Our implementation is based on the flexible GMRES routine in the TT-Toolbox [26]. The solver is run until a relative tolerance of $10^{-8}$ is achieved.

In Tables 4.5–4.7, we report the number of FGMRES iterations and the CPU times, for different Runge–Kutta methods. Further, we report the numerical error $v_{\text{error}}$ in the scaled vector $\ell^{\infty}$-norm, defined as in the previous section, together with the dimension of the system solved for each method.

Table 4.5: All-at-once solve of the heat equation: degrees of freedom (DoF), FGMRES iterations, CPU times, and resulting relative errors in $v$ for Gauss methods, employing $\mathbf{Q}_1$ and $\mathbf{Q}_2$ finite elements.

| | | $\mathbf{Q}_1$ | | | | $\mathbf{Q}_2$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| $s$ | `l` | DoF | `it` | CPU | $v_{\text{error}}$ | DoF | `it` | CPU | $v_{\text{error}}$ |
| 2 | 3 | 637 | 6 | 0.3 | 6.06e-03 | 4275 | 6 | 0.8 | 1.08e-04 |
| | 4 | 4275 | 6 | 0.7 | 1.45e-03 | 29,791 | 8 | 4.0 | 1.44e-05 |
| | 5 | 24,025 | 6 | 1.8 | 4.38e-04 | 194,481 | 8 | 20 | 2.07e-06 |
| | 6 | 146,853 | 8 | 13 | 9.62e-05 | 1,322,578 | 8 | 148 | 1.73e-07 |
| 3 | 3 | 833 | 9 | 0.8 | 5.40e-03 | 3825 | 8 | 1.1 | 3.57e-05 |
| | 4 | 3825 | 8 | 1.0 | 1.51e-03 | 24,025 | 9 | 4.0 | 1.98e-06 |
| | 5 | 24,025 | 9 | 3.2 | 3.52e-04 | 130,977 | 12 | 25 | 1.02e-07 |
| | 6 | 115,101 | 12 | 16 | 8.37e-05 | 790,321 | 13 | 167 | 1.57e-08 |

Table 4.6: All-at-once solve of the heat equation: degrees of freedom (DoF), FGMRES iterations, CPU times, and resulting relative errors in $v$ for Lobatto IIIC methods, employing $\mathbf{Q}_1$ and $\mathbf{Q}_2$ finite elements.

| | | $\mathbf{Q}_1$ | | | | $\mathbf{Q}_2$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| $s$ | `l` | DoF | `it` | CPU | $v_{\text{error}}$ | DoF | `it` | CPU | $v_{\text{error}}$ |
| 2 | 3 | 1225 | 5 | 0.5 | 1.28e-02 | 11,025 | 6 | 2.1 | 2.20e-03 |
| | 4 | 11,025 | 6 | 1.7 | 3.56e-03 | 133,579 | 8 | 18 | 3.25e-04 |
| | 5 | 93,217 | 7 | 8.2 | 1.05e-03 | 1,528,065 | 8 | 163 | 4.56e-05 |
| | 6 | 766,017 | 8 | 65 | 2.71e-04 | 17,580,610 | 10 | 2512 | 5.81e-06 |
| 3 | 3 | 833 | 6 | 0.5 | 5.42e-03 | 5625 | 7 | 1.3 | 9.82e-05 |
| | 4 | 5625 | 7 | 1.1 | 1.28e-03 | 39,401 | 7 | 5.1 | 1.58e-05 |
| | 5 | 31,713 | 7 | 3.3 | 3.63e-04 | 257,985 | 9 | 34 | 2.59e-06 |
| | 6 | 194,481 | 7 | 16 | 8.27e-05 | 1,758,061 | 10 | 292 | 3.68e-07 |
| 4 | 3 | 1029 | 9 | 1.0 | 5.75e-03 | 4725 | 9 | 1.7 | 2.37e-05 |
| | 4 | 4725 | 9 | 1.4 | 1.54e-03 | 29,791 | 12 | 7.3 | 1.57e-06 |
| | 5 | 29,791 | 12 | 5.9 | 3.53e-04 | 162,729 | 12 | 31 | 1.38e-07 |
| | 6 | 142,884 | 12 | 22 | 8.40e-05 | 983,869 | 13 | 231 | 1.75e-07 |
| 5 | 3 | 931 | 10 | 1.1 | 5.91e-03 | 5625 | 12 | 2.8 | 1.94e-05 |
| | 4 | 5625 | 12 | 2.4 | 1.54e-03 | 29,791 | 13 | 8.6 | 1.19e-06 |
| | 5 | 24,025 | 12 | 4.9 | 4.01e-04 | 146,853 | 13 | 33 | 2.40e-07 |
| | 6 | 123,039 | 12 | 20 | 9.64e-05 | 790,321 | 15 | 232 | 4.85e-08 |

Tables 4.5–4.7 show the parameter robustness of our preconditioner, with the linear solver converging in at most 15 iterations for all the methods and the parameters chosen. We are able to observe almost linear scalability of the solver with respect to the dimension of the problem. We would like to note that, when employing the finest grid for $\mathbf{Q}_2$ elements for the 2-stages Lobatto IIIC method, the all-at-once discretization results in a system of 17,580,610 unknowns to be solved on a laptop.

Table 4.7: All-at-once solve of the heat equation: degrees of freedom (DoF), FGMRES iterations, CPU times, and resulting relative errors in $v$ for Radau IIA methods, employing $\mathbf{Q}_1$ and $\mathbf{Q}_2$ finite elements.

| | | $\mathbf{Q}_1$ | | | | $\mathbf{Q}_2$ | | | |
|---|---|---|---|---|---|---|---|---|---|
| $s$ | `l` | DoF | `it` | CPU | $v_{\text{error}}$ | DoF | `it` | CPU | $v_{\text{error}}$ |
| 2 | 3 | 931 | 6 | 0.5 | 5.09e-03 | 5625 | 6 | 1.0 | 2.02e-04 |
| | 4 | 5625 | 6 | 0.9 | 1.35e-03 | 47,089 | 7 | 5.3 | 2.51e-05 |
| | 5 | 38,440 | 6 | 3.1 | 3.46e-04 | 384,993 | 8 | 41 | 3.41e-06 |
| | 6 | 254,016 | 8 | 21 | 8.35e-05 | 3,112,897 | 8 | 351 | 5.71e-07 |
| 3 | 3 | 833 | 7 | 0.5 | 5.71e-03 | 4725 | 8 | 1.2 | 2.57e-05 |
| | 4 | 4725 | 7 | 1.0 | 1.46e-03 | 27,869 | 9 | 4.4 | 2.28e-06 |
| | 5 | 27,869 | 9 | 3.7 | 3.30e-04 | 178,605 | 10 | 26 | 2.24e-07 |
| | 6 | 130,977 | 9 | 14 | 1.03e-04 | 1,048,385 | 12 | 204 | 2.68e-08 |
| 4 | 3 | 784 | 9 | 0.7 | 5.59e-03 | 4725 | 10 | 1.7 | 1.94e-05 |
| | 4 | 4725 | 10 | 1.5 | 1.54e-03 | 24,986 | 10 | 4.8 | 1.12e-06 |
| | 5 | 24,986 | 10 | 4.0 | 3.77e-04 | 142,884 | 12 | 27 | 7.56e-08 |
| | 6 | 123,039 | 12 | 18 | 9.00e-05 | 741,934 | 15 | 196 | 6.63e-07 |
| 5 | 3 | 931 | 12 | 1.2 | 5.91e-03 | 5625 | 14 | 3.2 | 1.92e-05 |
| | 4 | 5625 | 14 | 2.7 | 1.54e-03 | 24,025 | 13 | 6.4 | 1.16e-06 |
| | 5 | 24,025 | 14 | 5.7 | 4.01e-04 | 146,853 | 14 | 35 | 5.18e-08 |
| | 6 | 123,039 | 14 | 23 | 9.65e-05 | 693,547 | 13 | 171 | 1.32e-07 |

Finally, we also observe the predicted second- and third-order convergence for $\mathbf{Q}_1$ and $\mathbf{Q}_2$ discretization, respectively, until the linear tolerance causes slight pollution of the discretized solution for the finest grid of $\mathbf{Q}_2$ discretization (see, for instance, level `l` $= 6$ for the 5-stages Radau IIA method in Table 4.7).

**4.3. Stokes equations: sequential time-stepping.** In this section, we provide numerical results of the efficiency of the preconditioner $\widetilde{\mathcal{P}}_{\text{RK}}$ defined in (3.8) for the system of the stages for the Stokes problem. We run preconditioned GMRES up to a tolerance of $10^{-8}$ on the relative residual, with a restart after every 10 iterations.

We present results for a sequential time-stepping. We take $t_f = 2$, and solve the problem (2.10), testing our solver against the following exact solution:

$$\begin{aligned}\vec{v}(\mathbf{x},t) &= e^{t_f-t}[20x_1x_2^3, 5x_1^4 - 5x_2^4]^\top,\\ p(\mathbf{x},t) &= e^{10t_f-t}\left(60x_1^2x_2 - 20x_2^3\right) + \text{constant},\end{aligned}$$

with initial and boundary conditions obtained from this $\vec{v}$. The previous is an example of time-dependent *colliding flow*. A time-independent version of this flow has been used in [25, Section 3.1]. We evaluate the errors in the $L^\infty(\mathcal{H}_0^1(\Omega)^d)$ norm for the velocity and in the $L^\infty(L^2(\Omega))$ norm for the pressure, defined respectively as follows:

$$\begin{aligned}v_{\text{error}} &= \max_n \left[(\mathbf{v}_n - \mathbf{v}_n^{\text{sol}})^\top K_v(\mathbf{v}_n - \mathbf{v}_n^{\text{sol}})\right]^{1/2},\\ p_{\text{error}} &= \max_n \left[(\mathbf{p}_n - \mathbf{p}_n^{\text{sol}})^\top M_p(\mathbf{p}_n - \mathbf{p}_n^{\text{sol}})\right]^{1/2}.\end{aligned}$$

In the previous, $\mathbf{v}_n^{\text{sol}}$ (resp., $\mathbf{p}_n^{\text{sol}}$) is the discretized exact solution for $\vec{v}$ (resp., $p$) at time $t_n$.

In Tables 4.8–4.9, we report the average number of GMRES iterations and the average CPU times together with the numerical errors on the solutions, for different Runge–Kutta methods. Further, in Table 4.9 we report the dimensions of the system solved for Lobatto IIIC and Radau IIA methods. As for the heat equation, those values coincide with the dimensions of the system arising when employing the corresponding level of refinement in an $s$-stages Gauss method.

Table 4.8: Sequential solve of the Stokes equations: average GMRES iterations, average CPU times, and resulting errors in $\vec{v}$ and $p$, for Gauss methods.

| $s$ | `l` | `it` | CPU | $v_{\text{error}}$ | $p_{\text{error}}$ |
|---|---|---|---|---|---|
| 2 | 3 | 36 | 0.15 | 1.29e+00 | 9.73e-01 |
|  | 4 | 37 | 0.34 | 1.66e-01 | 4.86e-01 |
|  | 5 | 42 | 1.5 | 2.08e-02 | 2.15e-01 |
|  | 6 | 47 | 6.8 | 2.57e-03 | 1.20e-01 |
|  | 7 | 50 | 32 | 3.12e-04 | 5.94e-02 |
| 3 | 3 | 54 | 0.36 | 1.42e+00 | 7.19e-01 |
|  | 4 | 61 | 0.97 | 1.85e-01 | 1.20e-01 |
|  | 5 | 68 | 3.3 | 2.39e-02 | 3.20e-02 |
|  | 6 | 75 | 17 | 3.07e-03 | 1.10e-02 |
|  | 7 | 83 | 85 | 3.96e-04 | 3.87e-03 |

Table 4.9: Sequential solve of the Stokes equations: degrees of freedom (DoF), average GMRES iterations, average CPU times, and resulting errors in $\vec{v}$ and $p$, for Lobatto IIIC and Radau IIA methods.

|  |  |  | Lobatto IIIC |  |  |  | Radau IIA |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|
| $s$ | `l` | DoF | `it` | CPU | $v_{\text{error}}$ | $p_{\text{error}}$ | `it` | CPU | $v_{\text{error}}$ | $p_{\text{error}}$ |
| 2 | 3 | 1062 | 33 | 0.14 | 1.41e+00 | 5.67e-01 | 35 | 0.15 | 1.37e+00 | 5.07e-01 |
|  | 4 | 4422 | 39 | 0.36 | 2.04e-01 | 5.19e-02 | 38 | 0.37 | 1.77e-01 | 4.20e-02 |
|  | 5 | 18,054 | 46 | 1.5 | 3.19e-02 | 5.11e-03 | 44 | 1.4 | 2.27e-02 | 3.53e-03 |
|  | 6 | 72,966 | 50 | 6.8 | 5.25e-03 | 5.45e-04 | 49 | 6.9 | 2.91e-03 | 3.13e-04 |
|  | 7 | 293,382 | 54 | 34 | 8.95e-04 | 6.20e-05 | 54 | 36 | 3.90e-04 | 3.36e-05 |
| 3 | 3 | 1593 | 42 | 0.26 | 1.36e+00 | 5.06e-01 | 48 | 0.28 | 1.35e+00 | 5.01e-01 |
|  | 4 | 6633 | 44 | 0.64 | 1.76e-01 | 4.18e-02 | 55 | 0.79 | 1.75e-01 | 4.16e-02 |
|  | 5 | 27,081 | 54 | 2.7 | 2.24e-02 | 3.54e-03 | 64 | 3.1 | 2.23e-02 | 3.53e-03 |
|  | 6 | 109,449 | 62 | 14 | 2.82e-03 | 3.04e-04 | 74 | 17 | 2.82e-03 | 3.02e-04 |
|  | 7 | 440,073 | 70 | 73 | 3.55e-04 | 2.64e-05 | 78 | 85 | 3.54e-04 | 2.61e-05 |
| 4 | 3 | 2124 | 56 | 0.46 | 1.35e+00 | 4.99e-01 | 67 | 0.54 | 1.36e+00 | 4.97e-01 |
|  | 4 | 8844 | 67 | 1.3 | 1.75e-01 | 4.14e-02 | 69 | 1.4 | 1.75e-01 | 4.07e-02 |
|  | 5 | 36,108 | 75 | 5.2 | 2.23e-02 | 3.49e-03 | 79 | 5.4 | 2.22e-02 | 3.45e-03 |
|  | 6 | 145,932 | 80 | 25 | 2.81e-03 | 2.99e-04 | 87 | 28 | 2.80e-03 | 2.96e-04 |
|  | 7 | 586,764 | 88 | 131 | 3.53e-04 | 2.59e-05 | 91 | 135 | 3.52e-04 | 2.56e-05 |
| 5 | 3 | 2655 | 68 | 0.74 | 1.34e+00 | 4.88e-01 | 76 | 0.83 | 1.34e+00 | 4.89e-01 |
|  | 4 | 11,055 | 82 | 2.2 | 1.75e-01 | 4.08e-02 | 90 | 2.3 | 1.74e-01 | 4.10e-02 |
|  | 5 | 45,135 | 96 | 8.9 | 2.22e-02 | 3.46e-03 | 93 | 8.3 | 2.21e-02 | 3.43e-03 |
|  | 6 | 182,415 | 99 | 43 | 2.80e-03 | 2.94e-04 | 105 | 45 | 2.80e-03 | 2.95e-04 |
|  | 7 | 733,455 | 110 | 242 | 3.52e-04 | 2.55e-05 | 110 | 247 | 3.52e-04 | 2.55e-05 |

From Tables 4.8–4.9, we can observe that the mesh size $h$ is slightly affecting the average number of GMRES iterations[3]. Further, we can observe a more invasive dependence on the number of stages $s$. Nonetheless, the CPU times scale almost linearly with respect to the dimension of the system solved. Finally, we can observe

[3]From the authors' experience, this is because we are simply approximating the $(1,1)$-block. In fact, the block-commutator preconditioner would have been more effective if one employed a fixed number of GMRES iterations to approximately invert the $(1,1)$-block. However, having a further "layer" of preconditioner would have been non-optimal in the all-at-once approach.

that the error on the velocity and on the pressure behaves (roughly) as second order for every method but the 2-stages Gauss method, for which we observe a first-order convergence for the error on the pressure.

**5. Conclusions.** In this work, we presented a robust preconditioner for the numerical solution of the all-at-once linear system arising when employing a Runge–Kutta method in time. The proposed preconditioner consists of a block diagonal solve for the systems for the stages of the method, and a block-forward substitution for the Schur complement. Since the preconditioner requires a solver for the system for the stages, we proposed a preconditioner for the latter system based on the SVD of the Runge–Kutta coefficient matrix $A_{\mathrm{RK}}$. Numerical results showed the efficiency and robustness of the proposed preconditioner with respect to the discretization parameters and the number of stages of the Runge–Kutta method employed for a number of test problems.

In the next version of this report we will present results from a fully parallel implementation.

**Acknowledgements.** SL would like to thank Michele Benzi for useful discussions on Theorem 3.1 and on the preconditioner for the Stokes equations. JWP gratefully acknowledges financial support from the Engineering and Physical Sciences Research Council (EPSRC) UK grant EP/S027785/1. LB and AM gratefully acknowledge the support of the "INdAM – GNCS Project", CUP_E53C22001930001.

## REFERENCES

[1] R. Abu-Labdeh, S. MacLachlan, and P. E. Farrell, *Monolithic multigrid for implicit Runge–Kutta discretizations of incompressible fluid flow*, arXiv:2202.07381, 2022.

[2] O. Axelsson, I. Dravins, and M. Neytcheva, *Stage-parallel preconditioners for implicit Runge–Kutta methods of arbitrary high order. Linear problems*, Uppsala University, technical reports from the Department of Information Technology 2022-004, 2022.

[3] J. Boyle, M. Mihajlović, and J. Scott, *HSL_MI20: an efficient AMG preconditioner for finite element problems in 3D*, Int. J. Numer. Meth. Eng., 82, pp. 64–98, 2010.

[4] J. C. Butcher, *Numerical methods for ordinary differential equations*, Third Edition, John Wiley & Sons, Ltd, 2016.

[5] F. Danieli, B. S. Southworth, and A. J. Wathen, *Space-time block preconditioning for incompressible flow*, SIAM J. Sci. Comput., 44, pp. A337–A363, 2022.

[6] M. Emmett and M. Minion, *Toward an efficient parallel in time method for partial differential equations*, Comm. App. Math. Comput. Sci., 7, pp. 105–132, 2012.

[7] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, and J. B. Schroder, *Parallel time integration with multigrid*, SIAM J. Sci. Comput., 36, pp. C635–C661, 2014.

[8] S. Friedhoff, R. D. Falgout, T. V. Kolev, S. P. MacLachlan, and J. B. Schroder, *A multigrid-in-time algorithm for solving evolution equations in parallel*, in Sixteenth Copper Mountain Conference on Multigrid Methods, Copper Mountain, CO, United States, 2013.

[9] M. J. Gander, L. Halpern, J. Rannou, and J. Ryan, *A direct solver for time parallelization*, in: Domain Decomposition Methods in Science and Engineering XXII. Springer, pp. 491–499, 2016.

[10] M. J. Gander, L. Halpern, J. Rannou, and J. Ryan, *A direct time parallel solver by diagonalization for the wave equation*, SIAM J. Sci. Comput., 41, pp. A220–A245, 2019.

[11] G. H. Golub and C. F. van Loan, *Matrix computations*, 4th edition, The Johns Hopkins University Press, 1996.

[12] E. Hairer and G. Wanner, *Solving ordinary differential equations II: stiff and differential-algebraic problems*, 2nd edition, Springer, Berlin, 1996.

[13] B. C. Hall, *Lie groups, Lie algebras, and representations: an elementary introduction*, 2nd edition, Springer, 2015.

[14] M. Hinze, M. Köster, and S. Turek: *A space–time multigrid method for optimal flow control.* In: Leugering G., Engell S., Griewank A., Hinze M., Rannacher R., Schulz V., Ulbrich M., Ulbrich S. (eds.), *Constrained Optimization and Optimal Control for Partial Differential Equations*, pp. 147–170, Springer, Basel, 2012.

[15] R. A. Horn and C. A. Johnson: *Topics in matrix analysis*, Cambridge University Press, Cambridge, 1991.

[16] I. C. F. Ipsen: *A note on preconditioning nonsymmetric matrices*, SIAM J. Sci. Comput., 23, pp. 1050–1051, 2001.

[17] D. Kressner, S. Massei, and J. Zhu: *Improved parallel-in-time integration via low-rank updates and interpolation*, arXiv:2204.03073, 2022.

[18] J. D. Lambert, *Numerical method for ordinary differential systems: the initial value problem*, John Wiley & Sons, Ltd, New York, 1991.

[19] S. Leveque and J. W. Pearson, *Parameter-robust preconditioning for Oseen iteration applied to stationary and instationary Navier–Stokes control*, SIAM J. Sci. Comput., 44, pp. B694–B722, 2022.

[20] J. L. Lions, Y. Maday, and G. Turinici, *A "Parareal" in time discretization of PDE's*, C. R. Math. Acad. Sci. Paris, 332, pp. 661–668, 2001.

[21] Y. Maday and E. M. Rønquist, *Parallelization in time through tensor-product space-time solvers*, Comptes Rendus Mathematique, 346, pp. 113–118, 2008.

[22] K.-A. Mardal, T. K. Nilssen, and G. A. Staff, *Order-optimal preconditioners for implicit Runge–Kutta schemes applied to parabolic PDEs*, SIAM J. Sci. Comput., 29, pp. 361–375, 2007.

[23] E. McDonald, J. Pestana, and A. Wathen, *Preconditioning and iterative solution of all-at-once systems for evolutionary partial differential equations*, SIAM J. Sci. Comput., 40, pp. A1012–A1033, 2018.

[24] M. F. Murphy, G. H. Golub, and A. J. Wathen: *A note on preconditioning for indefinite linear systems*, SIAM J. Sci. Comput., 21, pp. 1969–1972, 2000.

[25] S. Norburn and D. J. Silvester: *Stabilised vs. stable mixed methods for incompressible flow*, Comput. Methods Appl. Mech. Eng., 166, pp. 131–141, 1998.

[26] I. V. Oseledets *et al.*: *TT-Toolbox Software*; see `https://github.com/oseledets/TT-Toolbox`.

[27] Md. M. Rana, V. E. Howle, K. Long, A. Meek, and W. Milestone, *A new block preconditioner for implicit Runge–Kutta methods for parabolic PDE problems*, SIAM J. Sci. Comput., 43, pp. S475–S495, 2021.

[28] Y. Saad: *A flexible inner–outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14, pp. 461–469, 1993.

[29] Y. Saad and M. H. Schultz: *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7, pp. 856–869, 1986.

[30] B. S. Southworth, O. Krzysik, W. Pazner, and H. De Sterck, *Fast solution of fully implicit Runge–Kutta and discontinuous Galerkin in time for numerical PDEs, part I: the linear setting*, SIAM J. Sci. Comput., 44, pp. A416–A443, 2022.

[31] G. A. Staff, K.-A. Mardal, and T. K. Nilssen, *Preconditioning of fully implicit Runge-Kutta schemes for parabolic PDEs*, Model. Identif. Control, 27, pp. 109–123, 2006.