Stage-parallel preconditioners for implicit Runge-Kutta methods of arbitrary high order. Linear problems

Owe Axelsson[1], Ivo Dravins[1], and Maya Neytcheva[1]

[1]Department of Information Technology, Uppsala University, Uppsala, Sweden

Keywords: Implicit Runge-Kutta methods, Radau quadrature, fully stage-parallel preconditioning

Abstract

Fully implicit Runge-Kutta methods offer the possibility to use high order accurate time discretization to match space discretization accuracy, an issue of significant importance for many large scale problems of current interest, where we may have fine space resolution with many millions of spatial degrees of freedom and long time intervals. In this work we consider strongly A-stable implicit Runge-Kutta methods of arbitrary order of accuracy, based on Radau quadratures. For the arising large algebraic systems we introduce an efficient preconditioner, that allows for fully stage-parallel solution. We analyse the spectrum of the corresponding preconditioned system and illustrate the performance of the solution method with numerical experiments using MPI. In this work we consider only linear problems.

1 Introduction

The availability of substantial computational power and high performance computing (HPC) resources has enabled high resolution, both in space and time, when performing numerical simulations of numerous problems of interest, modeling processes in physics, computational biology, financial and social processes, to name a few application areas.

Combining the requirements to have a fine, sometimes very fine, space resolution, high enough time resolution to balance the discretization error in space and time, fast and robust numerical simulations tilt the scales in favor of space discretization methods that easily handle complex geometries and adaptivity, together with higher order implicit time discretization methods. In this study we assume that the space discretization is done applying suitable finite element methods (FEM) and focus on a particular class of implicit

time discretization schemes, namely, the implicit Runge-Kutta (IRK) methods, based on Radau quadratures. To be specific, consider a system of evolutionary equations of the form

$$M\frac{\partial \boldsymbol{u}(t)}{\partial t} + K\boldsymbol{u}(t) = \boldsymbol{f}(t), \tag{1.1}$$

arising after semidiscretization in space of some non-stationary partial differential equation, say the heat equation $\frac{\partial u}{\partial t} = \Delta u + f(t)$, equipped with appropriate initial and boundary conditions. In (1.1), M is a mass matrix, K is a stiffness matrix and $\boldsymbol{u}(t)$, $\boldsymbol{f}(t)$ are vectors.

Evolution equations are by nature sequential. When solving evolution equations, standard time-stepping methods can be very costly and time-consuming. As is well known, when we use an explicit time-stepping method, to get a numerically stable solution, the time-step must be chosen sufficiently small and for ill-conditioned problems it must often be chosen unfeasibly small. Moreover, for strongly ill-conditioned problems, such methods may not even converge. In addition, even for the explicit methods, one must solve systems with the matrix M at each time-step. For the implicit methods, such as the backward Euler and trapezoidal methods one must solve systems with $M + \tau K$, or similar, where τ is the time-step. In this study we exclude from consideration the explicit time-step methods.

When we use a stable implicit time-integrator such as the backward Euler, the trapezoidal method or the Crank-Nicholson method (cf. [1]), the time-step must still be small to guarantee a sufficiently small time-integration error that balances a small space discretization error. Therefore, there are strong reasons to use higher order time-integration methods, which can enable the usage of much fewer time-steps and combine small total discretization error with fast integration in time.

As is also well known, see e.g [2], classical multistep methods can not have a higher order than two, otherwise they are not stable for all eigenvalues of the evolution operator $M^{-1}K$ with eigenvalues in the whole right half complex plane, that is, they are not A-stable. This can be a severe limitation because in many problems, such as network problems and time-harmonic Maxwell's equations there can appear rapidly changing oscillations, leading to the appearance of such widespread eigenvalues. On the other hand, in [3] see also [4], it is proven that there exist implicit Runge-Kutta methods of an arbitrary high order that are A-stable.

Implicit Runge-Kutta methods are first presented in [5], giving the methods the name 'IRK'. Independently, in [6] such methods are presented and it is shown that due to their high order of approximation and stability properties, they could be considered as global integration methods, that is, it could suffice to use just one or very few time-steps, that is, very large time-step intervals. In these original papers the A-stability property of the methods is not shown, but it is shown later in [7, 3].

There are several versions of IRK methods (see, e.g., [8, 9, 10, 11]). The most familiar methods are based on inner interval integration points being the zeros of some particular polynomials, namely,

(G) $\mathcal{P}_q(x)$, the Gauss integration method,

(R) $\mathcal{P}_q(x) - \mathcal{P}_{q-1}(x)$, the Radau or Gauss-Radau integration method,

(L) $\mathcal{P}_q(x) - \mathcal{P}_{q-2}(x)$, the Lobatto or Gauss-Lobatto integration method.

Here $\{\mathcal{P}_q\}$ denotes the set of Legendre polynomials of degree q, normalized at unity and transformed to the unit interval $[0, 1]$. We recall that all zeros of the Legendre polynomials are distinct. Further, as the polynomials are normalized at unity, $x = 1$ is a zero for the Radau and Lobatto methods. For the Lobatto method $x = 0$ is a zero too. More to mention, the approximation order (p) at the endpoints of each time-interval of methods (G), (R) and (L) is correspondingly $p = 2q$, $2q - 1$ and $2q - 2$. The approximation order at the interior integration points, however, is only of order q [5]. We stress, that the IRK methods are particularly important for large time intervals where we are somewhat less interested in the intermediate states of the solution but rather mainly in the final state of the underlying evolution systems.

The definition of the Legendre polynomials of degree q in the interval $[-1, 1]$ reads

$$L_q(x) = \frac{1}{2^q q!} \frac{d^q}{dx^q}(x^2 - 1)^q. \tag{1.2}$$

The polynomials satisfy the property $\int_{-1}^{1} L_p(x) L_q(x)\, dx = \delta_{pq}$. Formula (1.2) is recursive and is equivalent to

$$L_q(x) = \sum_{k=0}^{q} \binom{q}{k} \binom{q+k}{k} (-x)^{q+k}. \tag{1.3}$$

All methods (G), (R), (L) are A-stable, cf. e.g. [12]. Methods (G) and (L) are not strongly A-stable because for them the absolute value of the stability function converges to unity, implying, in the presence of large eigenvalues of K or the Jacobian matrix, at least a linear growth of rounding errors with increasing number of repeated time steps. In this work we advocate the Radau method which is the only strongly A-stable (also called strongly L-stable [13]) method, which means that the corresponding recursion stability factor converges to zero when the absolute values of the eigenvalues converge to infinity. Furthermore, the Radau method does not suffer from order reduction, that can occur for instance in the solution of systems of differential-algebraic equations, see e.g., [14, 15].

The high accuracy of the IRK methods makes them very attractive, in particular, in the case when the underlying problem requires a very fine space resolution and the time interval to integrate over is large. Then IRK allow to use large time steps and still balance the small space discretization error. Despite of that, the drawback with IRK is that in each timestep we have to solve the arising large algebraic system of order $qn \times qn$, where q is the stage order of the method, that is, equals the degree and number of zeros of the Legendre or the combination of Legendre polynomials and n is the size of M and K. For general IRK methods all stages are coupled and cannot be decomposed in some easier to handle form. The solution of this system can be costly and somewhat involved, which has been the major reason why IRK methods are more rarely used. In practice, the system must be solved by some preconditioned iterative method. Preferably, it should be possible to implement the method efficiently in a parallel computer environment.

We note, that aiming at avoiding the complications involved when applying implicit time-integration methods, there have been efforts in applying Richardson extrapolation and similar techniques in combination with explicit Runge-Kutta methods. The extrapolation still enables extension of the stability region, see, e.g. [16] and [17]. However, in addition to be forced to choose time-steps to make a stable recursion, the methods do not have the high order of discretization error, inherent in the IRK methods, see for instance, [9].

For an earlier discussion of solution techniques for IRK methods, see [18, 19], also [20] and [21], where diagonally implicit Runge-Kutta (DIRK) methods are presented. DIRK methods allow parallel implementation but have a much lower order of approximation, compared with the full IRK methods and, therefore, force the use of smaller time-steps. Since some time an alternative approach to solve time-dependent partial differential equations have been used, see, e.g., [22, 23, 24], based on combined time-space finite elements. This means that a 2D space partial differential operator is solved in a 3D space-time domain and a 3D space partial differential operator is solved in a 4D space-time finite element mesh. Clearly this complicates the implementation of the method, but such methods enable the use of adaptive mesh resolution methods in both time and space.

As stated above, in this paper we consider the solution of time-dependent partial differential equations. To discretize we apply finite elements only for the space domain. To do this, there are many different approaches, also various mesh refinement techniques can be used. Since the time interval is assumed to be quite large, the total number of degrees of freedom is expected not to be larger than for an adaptively constructed time-space mesh.

Nonlinear problems (although left out of the scope of the current paper) can be solved via some Newton iteration process, combined with a hierarchical mesh refinement process, that is a two-level or multilevel mesh method, see [25, 26, 27]. The hierarchical meshes are then used for the discretization of the space operator or for the combined time-space mesh. The idea is then to solve the nonlinear problem on each time-step by utilizing a coarse space mesh, where it is much cheaper to perform nonlinear iterations. The solution on the previous coarse mesh is interpolated to the next finer mesh and used as an initial approximation. Based on the norm of the local residuals it is then decided where the mesh must be locally refined. Since there are in general few additional mesh points, only few Newton nonlinear steps can be expected on each new mesh. This is easily understood since for a differentiable Jacobian matrix, the error arising from the Newton linearization is of second order, so even if we involve a coarse mesh discretization error, the iteration error will be of the same order as the fine mesh discretization error. Hence, it may even suffice to use just one Newton iteration step. The method is recursively repeated, intended to be the topic of a separate paper. Usage of adaptively constructed refined meshes has previously been applied in [28, 29]. The approach has been successfully applied also for non-differentiable Jacobians in the context of state-constrained optimal control problems, see [30].

Construction of efficient and well-parallelisable preconditioning methods for full IRK is the major topic of this paper. We use the result from [6] and [31] and analyse and implement in parallel methods which allow for full stage parallelism, that is, one can solve the arising q systems independently utilising q parallel processes or groups of processes.

The paper is structured as follows. In Section 2 we present the Radau type of IRK methods and various techniques to solve the arising block matrix linear system. Section 3 presents some versions of preconditioning methods which enable solving the full system efficiently and rapidly with parallelizable methods by use of proper iterative acceleration techniques. Section 4 contains some numerical experiments. Conclusions and outlook are given in Section 5. Apendix A contains details, related to the explicit form of the IRK matrices and Appendix B shows the algorithm to compute the eigenvectors of a triangular matrix. In Appendix C we briefly recollect some relevant characteristics of the Generalized Locally Toeplitz theory, used in analysing the spectrum of the preconditioned matrices. Appendix D provides additional insight on the spectrum of the preconditioned matrices for IRK with varying number of stages.

2 Implicit Runge-Kutta methods of Radau type

Time-integration methods are repeated on each time-step, whereby the end solution of the previous step is used as the initial solution for the next time interval.

Therefore, we do not use overlapping time-steps but instead advocate to utilize high order methods and very long time steps. To describe the method it suffices to consider just a single time-integration interval, $(0, \tau)$, $\tau > 0$, without any overlap.

In the subsequent derivations I_m denotes the identity matrix of order m. We utilize also the following tensor algebra identity with $\otimes$ being the matrix tensor product

$$(a \otimes b)(c \otimes d) = (ac) \otimes (bd). \tag{2.1}$$

2.1 Discrete tensor product matrix forms

To be specific, consider the differential equation (1.1), that is,

$$M\frac{d\boldsymbol{u}(t)}{dt} + K\boldsymbol{u}(t) = \boldsymbol{f}(t), \quad 0 < t \leq \tau, \quad \boldsymbol{u}(0) = \boldsymbol{u}_0.$$

This can be written

$$M\boldsymbol{v}(t) + K\int_0^t \boldsymbol{v}(s)ds = \boldsymbol{f}(t) - K\boldsymbol{u}_0, \tag{2.2}$$

where $\boldsymbol{v}(t) = \frac{d\boldsymbol{u}(t)}{dt}$, that is, $\boldsymbol{u}(t) = \int_0^t \boldsymbol{v}(s)ds + \boldsymbol{u}_0$.

Following [3], to solve (2.2) we use Radau quadrature, namely, we integrate from 0 to τc_i, $i = 1, \cdots, q$, where c_i are the zeros of $P_q(t) - P_{q-1}(t)$, $0 < t \leq 1$. Let

$$l_k(z) = \prod_{i=1, i\neq k}^{q} (z - c_i) \Big/ \prod_{i=1, i\neq k}^{q} (c_k - c_i)$$

be the Lagrange basic interpolation polynomial functions, $a_{ik} = \int_0^{c_i} l_k(z)dz, \quad i, k = 1, \ldots, q$, be the corresponding quadrature coefficients and let $\widetilde{\boldsymbol{v}}(t) = \sum_{k=1}^{q} \boldsymbol{v}_k l_k(t)$ be the corresponding interpolation polynomial. As shown in e.g., [32], the matrix $A_q = [a_{ik}]_{i,k=1}^{q}$ can be

presented in the form of a product of four matrices,

$$A_q = CVRV^{-1},$$

where $C = diag\{c_1, c_2, \cdots, c_q\}$, $R = diag\{1, 1/2, \cdots, 1/q\}$ and V is the Vandermonde matrix, generated by c_i, i.e.,

$$V = \begin{bmatrix} 1 & c_1 & c_1^2 & \cdots & c_1^{q-1} \\ 1 & c_2 & c_2^2 & \cdots & c_2^{q-1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & c_q & c_q^2 & \cdots & c_q^{q-1} \end{bmatrix}.$$

Since the zeros $\{c_i\}$ are distinct, V is nonsingular, thus, invertible. Clearly the IRK quadrature matrix A_q is nonsingular too.

Having computed A_q, by use of the numerical integration $\int_0^{c_i} \widetilde{\boldsymbol{v}}(s)\, ds = \sum_{k=1}^{q} a_{ik}\widetilde{\boldsymbol{v}}(c_k)$, $i = 1, 2, \cdots, q$, and using tensor products, we obtain the corresponding approximate algebraic form of (2.2),

$$(I_q \otimes M + \tau A_q \otimes K)\boldsymbol{v} = \boldsymbol{f} - (I_q \otimes K)(\boldsymbol{e}_q \otimes \boldsymbol{u}_0), \tag{2.3}$$

where $\boldsymbol{f} = [f_i]_{i=1}^q$, $f_i = f(\tau c_i)$ and $\boldsymbol{e}_q$ is a vector of length q with all components 1. The matrices M and K are real of size $n \times n$. The block vector $\boldsymbol{v}$ of length qn has block components $\boldsymbol{v}_i, i = 1, \cdots, q$ and each $\boldsymbol{v}_i$ is of length n. Note that

$$\boldsymbol{u}(\tau c_i) - \boldsymbol{u}_0 = \tau A_q \overline{\boldsymbol{v}}_i, \qquad i = 1, \cdots, q,$$

where $\overline{\boldsymbol{v}}_i = [v_i, v_{n+i}, \cdots, v_{(q-1)n+i}]$. Letting $\boldsymbol{w} = A_q \otimes I_n \boldsymbol{v}$, and utilising (2.1) we obtain the alternative transformed form of (2.3),

$$(A_q^{-1} \otimes M + \tau I_q \otimes K)\boldsymbol{w} = (A_q^{-1} \otimes I_n)\boldsymbol{f} - (A_q^{-1} \otimes K)(\boldsymbol{e}_q \otimes \boldsymbol{u}_0). \tag{2.4}$$

This transformation is suggested in [33], see also [34]. The systems (2.3) and (2.4) involve the $qn \times qn$ block matrices

$$\mathcal{A}_1 = I_q \otimes M + \tau A_q \otimes K \quad \text{and} \quad \mathcal{A}_2 = A_q^{-1} \otimes M + \tau I_q \otimes K, \tag{2.5}$$

respectively. To solve systems with those matrices, using an exact block matrix factorization method in a straightforward manner is unfeasible as it would lead to full matrices and would be clearly too expensive in computer time and memory demands. Therefore, to reduce the computer resource demands, instead of direct solution methods we must use a preconditioned iterative solution method, such as the generalized conjugate gradient method (GCG) ([35]) or the generalized minimum residual (GMRES) method ([11]), entailing the task to construct an efficient preconditioner.

To illustrate the IRK matrices for $q = 3$ or larger values of q it is convenient to use the so-called Butcher tableau ([36]), that is,

$$\left[\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{iq} \\ \vdots & & & \\ c_q & a_{q1} & \cdots & a_{qq} \\ \hline & b_1 & \cdots & b_q \end{array}\right] = \left[\begin{array}{c|c} \boldsymbol{c} & A_q \\ \hline & \boldsymbol{b} \end{array}\right].$$

However, we note, that for the Radau quadrature method $c_q = 1$, then $b_i = a_{qi}$, $i = 1, 2, \cdots, q$, so there is no need to use the vector $\boldsymbol{b}$. Note also that $c_i = \sum_{k=1}^{q} a_{i,k}$, that it, the rowsum of A equals the interpolation points used. For further comments on this, see [34].

We include here some examples of the quadrature matrices for the lowest order Radau methods. For stage order $q = 2$ we obtain

$$A_2 = \frac{1}{12}\begin{bmatrix} 5 & -1 \\ 9 & 3 \end{bmatrix} \quad \text{and} \quad A_2^{-1} = \frac{1}{2}\begin{bmatrix} 3 & 1 \\ -9 & 5 \end{bmatrix}.$$

As shown, e.g., in [3], the Butcher tableaux for $q = 3$ equals

$$\left[\begin{array}{c|ccc} \frac{2}{5} - \frac{\sqrt{6}}{10} & \left(\frac{11}{45} - \frac{7\sqrt{6}}{360}\right) & \left(\frac{37}{225} - \frac{169\sqrt{6}}{1800}\right) & \left(-\frac{2}{225} + \frac{\sqrt{6}}{75}\right) \\ \frac{2}{5} + \frac{\sqrt{6}}{10} & \left(\frac{37}{225} + \frac{169\sqrt{6}}{1800}\right) & \left(\frac{11}{45} + \frac{7\sqrt{6}}{360}\right) & \left(-\frac{2}{225} - \frac{\sqrt{6}}{75}\right) \\ 1 & \frac{4}{9} - \frac{\sqrt{6}}{36} & \frac{4}{9} + \frac{\sqrt{6}}{36} & \frac{1}{9} \end{array}\right].$$

A computation shows the exact form of A_3^{-1},

$$A_3^{-1} = \begin{bmatrix} \frac{\sqrt{6}}{2} + 2 & \frac{29\sqrt{6}}{30} - \frac{6}{5} & \frac{2}{5} - \frac{4\sqrt{6}}{15} \\ -\frac{29\sqrt{6}}{30} - \frac{6}{5} & 2 - \frac{\sqrt{6}}{2} & \frac{2}{5} + \frac{4\sqrt{6}}{15} \\ \frac{8\sqrt{6}}{3} - 1 & -\frac{8\sqrt{6}}{3} - 1 & 5 \end{bmatrix}.$$

Here the matrices are computed by exact integration using the symbolic Matlab tool and displayed in the above form as rational approximations using the Matlab command `rats` with relative accuracy $10^{-6}\|A\|$, respectively $10^{-6}\|A^{-1}\|$. One can notice that the signs in the subdiagonals of A^{-1} alternate, which is explained by A^{-1} being a sort of difference matrix. Further, the first upper diagonals are not that small, compared to the diagonal entries. The explicit forms of the matrix A_q for larger values of q are given in Appendix A.

3 Stage-parallel preconditioning methods

Note, that the second term in $\mathcal{A}_2$ in (2.5) involves a block-diagonal matrix and our aim is now to get a simpler form of the first matrix term also.

Based on the fact that A_q and, but to a lesser extent, also A_q^{-1} have a dominating lower-triangular part, one possibility to precondition $\mathcal{A}_1$ and $\mathcal{A}_2$ by the matrices

$$\mathcal{P}_1 = I_q \otimes M + \tau L_q^{(1)} \otimes K \quad \text{and} \quad \mathcal{P}_2 = L_q^{(2)} \otimes M + \tau I_q \otimes K,$$

respectively, where $L_q^{(1)}$ and $L_q^{(2)}$ are the lower-triangular factors in the exact LU-factorizations of A_q, respectively, of A_q^{-1}. This leads to achieving a block lower-triangular form of $\mathcal{P}_1$ and $\mathcal{P}_2$, however, the solution with them involves successively solving systems with the block-diagonal part $(I_q \otimes M + \tau(A_q \otimes K))_{ii}$ of $\mathcal{P}_1$ and correspondingly for $\mathcal{P}_2$, and does not allow for any parallelism between the stages of the method.

Clearly, a preconditioner based on directly using the lower-triangular part is mainly sequential, i.e., we must use a successive block-elimination method. We can achieve some parallelism by eliminating the lower block matrices in each block-column i, $i = 1, 2, \cdots, q$ in parallel. This gives parallelization factors $q-1$ for the first block matrix column, $q-2$ for the second one, etc., i.e. in the average the parallelization (speedup factor) is $q/2$.

For $\mathcal{P}_1$ the elimination requires solutions with matrices $(I_q \otimes M + \tau(A_q \otimes K))_{ii}$. When K is ill-conditioned then these block matrices are also ill-conditioned and we need a good preconditioner of these matrices too, for instance of algebraic multilevel (AMLI) type or of algebraic multigrid (AMG) type method [37, 38]. There exists various possibilities to employ parallel solvers for these inner systems, but we do not discuss this any further.

The same type of successive elimination method can be applied also for (2.4). Here the block-diagonal matrices equal $\mathcal{A}_{2,ii}$ which may have a much better form for the efficiency of the inner solution than $\mathcal{A}_{1,ii}$. Furthermore,

(i) the off-diagonal blocks arise now from the matrix term $A_q^{-1} \otimes M$, which is in general sparser than $A_q \otimes K$, in particular if M is a lumped mass matrix, hence, we save also in demand of memory;

(ii) in addition, any ill-conditioning of K does not harm the off-diagonal blocks.

Since solving (2.4) instead of (2.3) can significantly reduce the computational cost, it can be preferable and this is also the approach taken in this paper. In the next section we present some versions how to construct a preconditioner to the matrix $\mathcal{A}$ of a block-diagonal and enables stage-parallel solutions.

3.1 A summary of some earlier presented methods

Over many years, there have been various attempts to construct parallel solution methods for IRK problems, either by approximating the method by some lower order method on simpler form, by constructing a parallel preconditioner or by extending the method to a multistage method on several time steps. We list here some of these methods. There are three types of approaches that achieve some parallelism,

(i) across the method,

(ii) across the problem,

(iii) across the time-steps.

For a more general discussion of parallelization in Runge-Kutta methods, we refer to [39]. An early attempt was to use diagonally-implicit Runge-Kutta methods, also called DIRK methods, see e.g. [40], also [21]. It is straightforward to parallelize DIRK methods but, as already mentioned, they suffer from lower order approximation, that is, require smaller and, therefore, many more time steps. An early experience in using a diagonal matrix as a preconditioner to the IRK method is found in [41]. There, the linear systems

are decoupled into subsystems, which means that the cost of the full LU factorization of the global matrix is reduced to q factorizations of the diagonal blocks of size n.

As pointed out in [41], the construction of the preconditioner can be based on the Vandermonde matrix representation of the quadrature matrix, see Section (3.2) for details.

One strategy to construct a preconditioner is based on the spectral decomposition of the quadrature matrix A_q,

$$T^{-1}A_qT = \Lambda = diag\{\lambda_1, \lambda_2, \cdots, \lambda_q\}.$$

The matrix in the system (2.4) can then be transformed into a factorized form involving a block-diagonal matrix that decouples the stages, namely,

$$\mathcal{A}_1 = I_q \otimes M + \tau A_q \otimes K = (T^{-1} \otimes I_n)(I_q \otimes M + \tau\Lambda \otimes K)(T \otimes I_n). \tag{3.1}$$

However, the eigenvalues of A_q are reals and/or complex, which impairs the parallelism and increases the decomposition cost. Therefore, it is suggested to use the transformation of A_q

$$A_q = CVRV^{-1} = B^{-1}W^{-T}XW^{-1}, \tag{3.2}$$

where the entries of W are computed as $w_{ij} = P_{j-1}(c_i)$ with $P_i(x)$ being the shifted Legendre polynomial of degree i and $B = diag(b_1, b_2, \cdots, b_q)$. The matrix X is tridiagonal. The above W-transformation is first published in [8] and also advocated in [42]. In this way the complex arithmetic is avoided and the decoupling cost for the parallel implementation of this approach correspond to that for the implicit Euler method. In [42] the factorization (3.2) is used to obtain a preconditioned of block-tridiagonal form, which is then LU-factorized and simplified using a method parameter to be determined. In [43] standard preconditioners for low-order time discretizations are used to construct order-optimal diagonal block Jacobi preconditioners for high order discretizations. The convergence properties of the methods are improved in [44] by employing block Gauss-Seidel techniques.

In [45] optimal complex and real Schur-based preconditioners are compared, together with a block Jordan-form preconditioner and a near optimal singly diagonal approximate block real Schur decomposition is derived. The latter in particular has memory requirements and setup cost comparable to singly DIRK methods.

An example of a parallel across the problem method is the wave-front relaxation technique, see for instance, [46]. A drawback to mention here is that for stiff problems the convergence of the wave-front to the exact solution can be very slow. A similar technique is to use a time-harmonic expansion of the solution, which is a natural approach for problems with alternating source functions, such as in time-harmonic electromagnetic problems, cf. [47, 48] and the references therein. In such an approach the problem decouples into independent subproblems per angular frequency and in this way there is no need to use any time-integration method at all. Another approach to use Fourier expansions is to extend the interval $[0, \tau]$ to $[0, 2\tau]$ and use the backward form of the IRK method on the symmetrical interval $[\tau, 2\tau]$. Here the solution can be expanded in $e^{i\omega_k t}$ terms, which, again leads to uncoupled problems for each frequency ω_k and can be solved fully in parallel, cf. [48].

For parabolic problems of the form

$$\boldsymbol{u}_t + K\boldsymbol{u} = \boldsymbol{0}, \boldsymbol{u}(0) = \boldsymbol{u}_0,$$

one can use the Laplace transform with parameter s. We then have

$$s\widehat{\boldsymbol{u}} + K\widehat{\boldsymbol{u}} = \boldsymbol{u}_0, \text{ which implies } \widehat{\boldsymbol{u}} = (sI + K)^{-1}\boldsymbol{u}_0.$$

The solution $\boldsymbol{u}(t)$ can be computed via the inverse Laplace transform

$$\boldsymbol{u}(t) = \frac{1}{2\pi\, i} \int_{\Gamma} e^{st}\widehat{\boldsymbol{u}}(s)\, ds,$$

where Γ is a contour in the right half plane, i.e., contains no eigenvalues of $sI + A$. Here the integral is approximated with a quadrature formula with nodes s_j. Then one only needs to compute $\widehat{\boldsymbol{u}}(s)$ at all s_j, which can be done fully in parallel. For details, see [49, 50].

Another possible parallelization method is the boundary value technique, that has appeared in [51]. Here the whole time interval is solved by computing the equation as arising in the implicit trapezoidal method with a backward midpoint rule at the endpoint. The whole system then becomes block tridiagonal and can be solved in various ways by common methods for two-point boundary value problems.

Since long, multigrid methods have successfully been used to solve space discretized problems and also have been shown to be efficiently parallelized. Parallel-in-time, i.e., parallelizable across time methods have also been developed, such as multiple-shooting techniques, cf. e.g. [52]. The Parareal framework has further renewed the interest in such methods, see [53, 54]. We claim that if one can solve the full IRK method in parallel in each time step, there is less need for such multiple timestep methods. However, these can be useful to obtain solution with higher accuracy also in interior integration points. A GPU implementation of the fully implicit IRK method is found in [55], applied to a system of ODEs. There, the matrices to be solved have the same structure as in (3.1) and the preconditioner has a tridiagonal structure.

In [33] a stage-parallel approach employing ILU-based preconditioners is presented. A study of a series LU-based preconditioning approaches is presented in [56], applied on the non-transformed equation (2.4). Related to IRK methods, however out of the context of parallel implementation, preconditioners for quadratic matrix polynomials are analysed in [57]. A recent work on preconditioning IRK is found in [58], in particular a bound on the condition number is obtained by an order one constant which is independent of spatial mesh and time-step size. There, various experiences in preconditioning the systems arising in IRK are also discussed with regard to their suitability for parallelization.

3.2 Preconditioners for the Radau tensor product system

As discussed in Section 2, the use of (2.4) with the inverse of the quadrature matrix has several important advantages over (2.3). Therefore, we consider only the form (2.4). To simplify the notations, we drop the subscript 2 in $\mathcal{A}_2$ and in $\mathcal{P}_2$.

Recall that $\mathcal{A} = A_q^{-1} \otimes M + \tau I_q \otimes K$. Let $A_q^{-1} = L_q U_q$, where it is assumed that the factor L_q is real-valued block lower-triangular and the upper-triangular factor U_q has a unit diagonal, thus, $U_q = I_q + \widehat{U}_q$ with $\widehat{U}_q$ strictly upper-triangular.

As shown in [6], the matrix A_q^{-1} has a dominating lower-triangular part. Therefore, in the construction of a preconditioner to the matrix $\mathcal{A}_1$ in (2.3) it can be efficient to use the lower triangular part L_q as an approximation of A_q^{-1}. Namely, consider the matrix

$$\mathcal{P} = L_q \otimes M + \tau I_q \otimes K \tag{3.3}$$

to be used as a preconditioner to $\mathcal{A}$, If applied straightforwardly, solutions of systems with $\mathcal{P}$ do not allow parallelism across stages. As earlier attempts towards overcoming this and aiming to obtain a fully stage-parallel preconditioner, we refer to [59, 60]. It is suggested to factorize A_q^{-1} as $T\Lambda T^{-1}$, where Λ is the diagonal matrix, formed by the eigenvalues of A_q^{-1}. However, as already mentioned, at least some of the eigenvalues appear as complex conjugate pairs and we are then forced to use complex arithmetic, leading to a less favourable computations and increased memory demands. This is the major reason why we use the spectral decomposition of L_q instead of A_q^{-1}. So, we assume that L_q is a good approximation to A_q^{-1} and construct the spectral decomposition $L_q = T_q \Lambda_q T_q^{-1}$. Here, T_q contains the eigenvectors of L_q and Λ_q is the diagonal part of L_q that contains the eigenvalues of L_q, all real. Using (2.1) we can now easily transform the solution of $\mathcal{P}\boldsymbol{v} = \boldsymbol{w}$ to a solution with a block-diagonal matrix. Namely,

$$\begin{aligned}
\mathcal{P} &= L_q \otimes M + \tau I_q \otimes K = T_q \Lambda_q T_q^{-1} \otimes M + \tau T_q T_q^{-1} \otimes K \\
&= (T_q \otimes I_n)(\Lambda_q \otimes M)(T_q^{-1} \otimes I_n) + \tau (T_q \otimes I_n)(I_q \otimes K)(T_q^{-1} \otimes I_n) \\
&= (T_q \otimes I_n)\left((\Lambda_q \otimes M) + \tau (I_q \otimes K)\right)(T_q^{-1} \otimes I_n) \\
&= (T_q \otimes I_n)\, \mathcal{P}_d\, (T_q^{-1} \otimes I_n),
\end{aligned} \tag{3.4}$$

where $\mathcal{P}_d = \Lambda_q \otimes M + \tau(I_q \otimes K)$. Hence, $\mathcal{P}_d$ is block diagonal! Then, instead of $\mathcal{P}\boldsymbol{v} = \boldsymbol{w}$, we solve $\mathcal{P}_d \boldsymbol{w} = (T_q^{-1} \otimes I_n)\boldsymbol{w}$ and recover $\boldsymbol{v}$ as $\boldsymbol{v} = (T_q \otimes I_n)\boldsymbol{w}$. The eigenvectors can be obtained by some available linear algebra software. Actually, as shown in Appendix B, this computation can be done by a simple recursion. We see first that

$$\begin{aligned}
\mathcal{P}^{-1}\mathcal{A} &= (L_q \otimes M + \tau I_q \otimes K)^{-1}(A_q^{-1} \otimes M + \tau I_q \otimes K) \\
&= (L_q \otimes M + \tau I_q \otimes K)^{-1}\left[(L_q \otimes M + \tau I_q \otimes K) + L_q \widehat{U}_q \otimes M\right] \\
&= I_{qn} + (L_q \otimes M + \tau I_q \otimes K)^{-1}(L_q \widehat{U}_q \otimes M),
\end{aligned} \tag{3.5}$$

where $\widehat{U}_q$ is the strictly upper triangular part of U_q. Hence,

$$\begin{aligned}
\|\mathcal{P}^{-1}\mathcal{A} - I_{qn}\| &= \|(L_q \otimes M + \tau I_q \otimes K)^{-1}(A_q^{-1} \otimes M + \tau I_q \otimes K) - I_{qn}\| \\
&\leq \|(L_q \otimes M + \tau I_q \otimes K)^{-1}(L_q \otimes M)\| \|\widehat{U}_q \otimes I_n\|,
\end{aligned}$$

which is predicted to be small since $\|\widehat{U}_q\| < 1$ (and can be reasonably small). Further, if K has a positive definite symmetric part, then $\|(L_q \otimes M + \tau I_q \otimes K)^{-1}(L_q \otimes M)\|$ is likely

smaller than unity and particularly small for large timesteps τ. To take an example, for $q = 2$, then

$$A_2^{-1} = \begin{bmatrix} \frac{3}{2} & 0 \\ -\frac{9}{2} & 4 \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{3} \\ 0 & 1 \end{bmatrix}, \text{ thus, } \widehat{U}_2 = \begin{bmatrix} 0 & \frac{1}{3} \\ 0 & 0 \end{bmatrix},$$

that is, $\|\widehat{U}_2\| = 1/3$. For $q = 3$ and $q = 4$ we have correspondingly, $\|\widehat{U}_3\| = 0.4098$, $\|\widehat{U}_4\| = 0.4779$ and

$$L_3 = \begin{bmatrix} 3.2247 & 0 & 0 \\ -3.5678 & 2.0673 & 0 \\ 5.5320 & -9.5354 & 9 \end{bmatrix}, \quad U_3 = \begin{bmatrix} 1 & 0.3621 & -0.0785 \\ 0 & 1 & 0.3739 \\ 0 & 0 & 1 \end{bmatrix},$$

$$L_4 = \begin{bmatrix} 5.6441 & 0 & 0 & 0 \\ -5.0492 & 2.9419 & 0 & 0 \\ 3.4925 & -5.1747 & 3.1618 & 0 \\ -6.9235 & 8.9548 & -16.6361 & 16 \end{bmatrix}, \quad U_4 = \begin{bmatrix} 1 & 0.3408 & -0.1038 & 0.0308 \\ 0 & 1 & 0.4183 & -0.0949 \\ 0 & 0 & 1 & 0.3869 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Next we analyse the spectrum of the preconditioned matrix $\mathcal{P}^{-1}\mathcal{A}$ in more detail. Consider the generalized eigenvalue problem

$$\mathcal{A}\boldsymbol{v} = \nu\mathcal{P}\boldsymbol{v}. \tag{3.6}$$

We transform it to

$$(\mathcal{A} - \mathcal{P})\boldsymbol{v} = (\nu - 1)\mathcal{P}\boldsymbol{v}. \tag{3.7}$$

Let $\mu = \nu - 1$ and observe from (3.5) that $\mathcal{A} - \mathcal{P} = L_q\widehat{U}_q \otimes M$. Hence, μ is an eigenvalue of

$$(L_q \otimes M + \tau I_q \otimes K)^{-1}(L_q\widehat{U}_q \otimes M)\boldsymbol{v} = \mu\boldsymbol{v}. \tag{3.8}$$

A straightforward computation shows that the matrix in (3.8) has one zero block-column of size $qn \times n$. Thus, n of the eigenvalues μ are equal to zero. To analyse the rest $(q-1)n$ eigenvalues, using the spectral decomposition of L_q, $L_q = T_q\Lambda_q T_q^{-1}$, we obtain the following,

$$\begin{aligned}
&(L_q \otimes M + \tau I_q \otimes K)^{-1}(L_q\widehat{U}_q \otimes M) \\
&= \left[(T_q \otimes I_n)(\Lambda_q \otimes M + \tau I_q \otimes K)(T_q^{-1} \otimes I_n)\right]^{-1}(T_q \otimes I_n)(\Lambda_q T_q^{-1}\widehat{U}_q \otimes M) \\
&= (T_q^{-1} \otimes I_n)^{-1}\left[\Lambda_q \otimes M + \tau I_q \otimes K\right]^{-1}(T_q \otimes I_n)^{-1}(T_q \otimes I_n)(\Lambda_q T_q^{-1}\widehat{U}_q \otimes M) \\
&= (T_q^{-1} \otimes I_n)^{-1}\left[I_{qn} + (\Lambda_q \otimes M)^{-1}(\tau I_q \otimes K)\right]^{-1}(\Lambda_q \otimes M)^{-1}(\Lambda_q T_q^{-1}\widehat{U}_q \otimes M) \\
&= (T_q^{-1} \otimes I_n)^{-1}\left[I_{qn} + (\Lambda_q \otimes M)^{-1}(\tau I_q \otimes K)\right]^{-1}(T_q^{-1}\widehat{U}_q \otimes I_n) \\
&= \underbrace{(T_q \otimes I_n)}_{\widetilde{T}}\underbrace{\left[I_{qn} + (\Lambda_q \otimes M)^{-1}(\tau I_q \otimes K)\right]^{-1}}_{W_1^{-1}}\underbrace{(T_q^{-1} \otimes I_n)}_{\widetilde{T}^{-1}}\underbrace{(\widehat{U}_q \otimes I_n)}_{W_2}
\end{aligned} \tag{3.9}$$

Due to the similarity transformation with $\widetilde{T}$ the eigenvalues of the matrix $\widetilde{T}W_1^{-1}\widetilde{T}^{-1}$ are equal to those of the matrix $W_1^{-1} = [I_{qn} + (\Lambda_q \otimes M)^{-1}(\tau I_q \otimes K)]^{-1}$. The matrix W_1^{-1} is

block-diagonal with blocks $\left(I_n + \frac{\tau}{\lambda_j} M^{-1}K\right)^{-1}$, $j = 1, \cdots, q$, λ_j are the diagonal entries of Λ_q, and all eigenvalues of W_2 are zero since $\widehat{U}_q$ is strictly upper triangular.

We need to estimate the bounds of the spectrum of $\widetilde{W} = \widetilde{T} W_1^{-1} \widetilde{T}^{-1} W_2$. The matrix $\widetilde{W}$ is a product of four matrices and is nonsymmetric. In such cases the standard tools to analyse its spectrum are rather limited. We derive an upper bound of its spectral radius, which shows that the spectrum can be considered nearly h- and τ-independent, and is improved for larger timesteps. To this end we use the following results. Let A and B denote generic square matrices, $\mathcal{S}(A)$ be the spectrum of A and $\mathcal{W}(A)$ its field of values, $\rho(A)$ and $\omega(A)$ be the spectral and the numerical radius of A, respectively. The following relations are known to hold:

(P1) $\rho(A) \leq \omega(A)$, where $\rho(A) = \max\{|\lambda : \ \lambda \in \text{spectrum of } A\}$,

(P2) $\frac{1}{2}\|A\| \leq \omega(A) \leq \|A\|$, where $\omega(A) = \sup\{(\boldsymbol{x}, A\boldsymbol{x}) : \|\boldsymbol{x}\| = 1\}$,

(P3) $\omega(AB) \leq 4\,\omega(A)\,\omega(B)$, c.f. [61],

(P4) $\|A \otimes B\| = \|A\|\,\|B\|$ and $\rho(A \otimes B) = \rho(A)\,\rho(B)$, c.f. [61],

(P5) **Theorem** ([62]) Let T be a Toeplitz matrix of size n and $f(z)$ be its symbol. Let $Co(\Omega)$, $\Omega \subset \mathbb{C}$ be the convex hull of Ω. Then

$$\mathcal{W}(T) \subseteq Co[f(\|z\| = 1)].$$

Moreover, the Hausdorff distance between the two sets goes to 0 with $n \to \infty$.

We reformulate the eigenvalue problem $\widetilde{T} W_1^{-1} \widetilde{T}^{-1} W_2 \boldsymbol{v} = \mu \boldsymbol{v}$ in (3.8) as

$$W_1^{-1} \widetilde{T}^{-1} W_2 \widetilde{T} \boldsymbol{w} = \mu \boldsymbol{w}, \text{or, equivalently, } W_1^{-1} \left((T_q^{-1} \widehat{U}_q T_q) \otimes I_n \right) \boldsymbol{w} = \mu \boldsymbol{w}, \tag{3.10}$$

where $\boldsymbol{w} = \widetilde{T}^{-1} \boldsymbol{v}$. We use that $\widetilde{T}^{-1} W_2 \widetilde{T} = (T_q^{-1} \otimes I_n)(\widehat{U}_q \otimes I_n)(T_q \otimes I_n)^{-1} = T_q^{-1} \widehat{U}_q T_q \otimes I_n$.

Then, we have

$$\begin{aligned} \rho(\widetilde{W}) \ &\leq \ \omega(\widetilde{W}) \leq 4\,\omega(W_1^{-1})\,\omega((T_q^{-1} \widehat{U}_q T_q \otimes I_n)) \leq 4\,\omega((T_q^{-1} \widehat{U}_q T_q)\,\omega(W_1^{-1}) \\ &= \ 4\|T_q^{-1} \widehat{U}_q T_q\|\,\omega(W_1^{-1}) = Const\,\omega(W_1^{-1}), \end{aligned} \tag{3.11}$$

where $Const$ is a constant that depends on q but does not depend on h and τ. The value of this constant is shown in Table 1.

It remains to estimate $\omega(W_1^{-1}) = \omega([I_{qn} + (\Lambda_q \otimes M^{-1})(\tau I_q \otimes K)]^{-1})$. The matrix sequence $\{W_1\}$, as a function of the increasing matrix size n, consists of block-diagonal matrices with blocks $I_n + \frac{\tau}{\lambda_i} M^{-1}K$. As $\{W_1\}$ and, respectively, $\{W_1^{-1}\}$ belong to the class of the generalized locally Toeplitz (GLT) matrices, their numerical radius can be estimated using property (P5). The theoretical justification to apply GLT in our case is

q	$\|T\|$	$\|T^{-1}\|$	$\|\widehat{U}\|$	$4\,\|T^{-1}\widehat{U}T\|$
2	1.369	2.819	0.33333	4.84
3	1.667	8.856	0.40983	8.32
4	1.922	$1.20\ 10^2$	0.47791	35.67
7	2.541	$3.48\ 10^4$	0.59606	$2.95\ 10^2$
9	2.882	$1.96\ 10^5$	0.64093	$4.74\ 10^3$

Table 1: Value of the factor *Const* in (3.11)

given in [63, 64, 65, 66]. Appendix C summarizes some major GLT properties, relevant to the current spectral analysis.

Consider next the matrix sequences $\{M\}$ and $\{K\}$ in terms of n. In 2D the mass matrix M is block-tridiagonal and each block has a tridiagonal structure. As shown in [67], for quadrilateral meshes the block-symbol of $\{M\}$, $f^M(\theta_1,\theta_2)$ is computed as follows,

$$f^M(\theta_1,\theta_2) \;=\; \frac{4h^2}{36}(2+\cos(\theta_1))(2+\cos(\theta_2)). \tag{3.12}$$

The function f^M/h^2 is the GLT symbol of $\{M/h^2\}$. The nonscaled matrix sequence $\{M\}$ is distributed as the constant 0 due to the factor h^2 in front of M, which goes to zero as the matrix size tends to infinity.
The stiffness matrix has also a block-tridiagonal structure and the block-symbol of $\{K\}$ is

$$f^L(\theta_1,\theta_2) \;=\; \frac{1}{3}\,(8-2\cos(\theta_1)-2\cos(\theta_2)(1+2\cos(\theta_1))). \tag{3.13}$$

Here, θ_1 and θ_2 are generic angles between $-\pi$ and π. Then, as the GLT matrices form an algebra, we straightforwardly derive the symbol of W_1^{-1}, $f^{W_1^{-1}}$ as follows,

$$f^{W_1^{-1}}(\theta_1,\theta_2) = \frac{1}{1+4\dfrac{\tau\,h^{-2}}{\lambda_i}\dfrac{8-2\cos(\theta_1)-2\cos(\theta_2)(1+2\cos(\theta_1))}{(2+\cos(\theta_1))(2+\cos(\theta_2))}}. \tag{3.14}$$

From Property (P5) we see that for $\theta_j=\pm\pi$, $j=1,2$ we have that

$$f^{W_1^{-1}} = \frac{\lambda_i}{\lambda_i+8\,\tau\,h^{-2}} < 1.$$

We add next the relations between τ and h to balance the global discretization error. For piece-wise linear discretization in space the space discretization error is $O(h^2)$, the time discretization error is $O(\tau^{2q-1})$ and for given q τ and h should be chosen so that both errors should be of the same order. Thus, $h=\tau^{\frac{2q-1}{2}}$ and $\tau\,h^{-2}=\tau^{2-2q}$. Table 2 shows the value of $\tau\,h^{-2}$ for some values of τ and q. It shows that $f^{W_1^{-1}}$ approaches zero very fast, which shows that $|\mu|$ approaches zero. We see that since $\omega(W_1^{-1})\in\mathcal{W}(W_1^{-1})\subseteq Co[f^{W_1^{-1}}(\|z\|=1)]$ and

$\tau \backslash q$	2	4	7	9
0.1	$1.0\,10^{1}$	$1.0\,10^{5}$	$1.0\,10^{11}$	$1.0\,10^{15}$
0.01	$1.0\,10^{4}$	$1.0\,10^{11}$	$1.0\,10^{23}$	$1.0\,10^{31}$
0.001	$1.0\,10^{6}$	$1.0\,10^{17}$	$1.0\,10^{35}$	$2.0\,10^{47}$

Table 2: Value of $\tau\, h^{-2}$

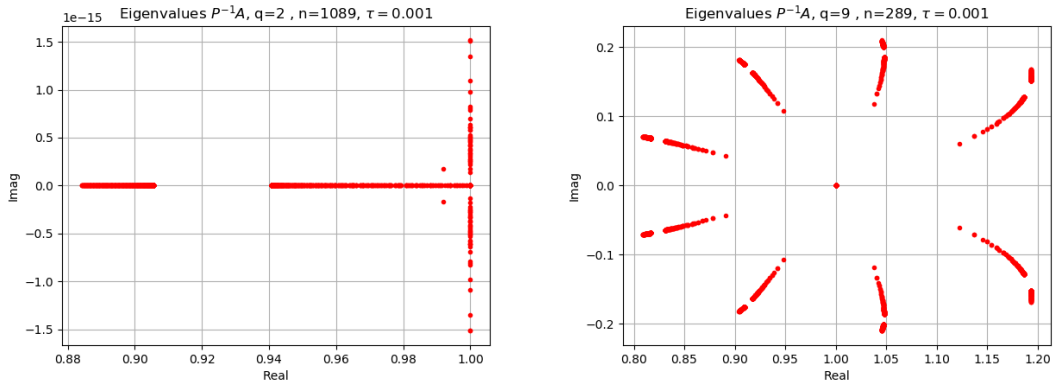

Figure 1: Problem 1: Eigenvalues of the preconditioned system, left $q = 2$, right $q = 9$

$f^{W_1^{-1}}(\|z\| = 1)$ decreases rapidly to zero with $h \to 0$ and τ appropriately chosen.

Figure 1 illustrates the spectrum of $\mathcal{P}^{-1}\mathcal{A}$.

Consider now an alternative for obtaining an LU factorization of A_q^{-1}, where L_q is real-valued and dominating over $\widetilde{U}_q$. We examine the particular factorization of A_q that is based on the Vandermonde matrix, as shown in (3.2), namely, $A_q = CVRV^{-1}$, where $C = diag(c_1, c_2, \cdots, c_q)$, $R = diag(1, 1/2, \cdots, 1/q)$ and V is the Vandermonde matrix, defined by the integration points $c_i, i = 1, 2, \cdots, q$. Then

$$A_q^{-1} = VR^{-1}V^{-1}C^{-1} \quad \text{and} \quad A_q^{-1}C = VR^{-1}V^{-1}.$$

Let factorize $VRV^{-1} = L_0U_0$, where U_0 has unit diagonal. Clearly, $VR^{-1}V^{-1} = U_0^{-1}L_0^{-1}$. Choose now as L_q the matrix

$$L_{q,0} = L_0^{-1}C^{-1}, \tag{3.15}$$

compute its spectral decomposition $L_{q,0} = T_{q,0}\Lambda_{q,0}T_{q,0}^{-1}$ and construct a preconditioner as in (3.3),

$$\mathcal{P} = L_{q,0} \otimes M + \tau I_q \otimes K. \tag{3.16}$$

As already mentioned, the eigenvectors can be computed by recursion as shown in Appendix B. Then, the following relations hold:

$$L_{q,0}^{-1}A_q^{-1} \;=\; CVRV^{-1}U_0VR^{-1}V^{-1}C^{-1} \;=\; ZU_0Z^{-1}$$

with $Z = CVRV^{-1}$. Thus, $L_{q,0}^{-1}A_q^{-1}$ is spectrally equivalent to U_0 which latter has all eigenvalues equal to one. The analysis of the preconditioned matrix $\mathcal{P}^{-1}\mathcal{A}$ with $L_{q,0}$ instead of L_q remains the same, the difference is in the effect of the matrices T_q.

As already mentioned, the systems (2.3) and (2.4) are in general very large scale, of order $(qn)^2$ but with the approach taken here we solve systems only of size n. In typical problems, in particular for there space dimensional problems, n itself is very large. We assume that we can either construct a feasible approximation of the diagonal blocks $\lambda_j M + \tau K$, $j = 1, 2, \cdots, q$ or can solve the inner systems with some efficient iterative solution method. A suitable approximation could for instance be based on a modified incomplete factorization, see [68, 69, 70] with sparse matrix factors. For the approach to solve systems with the diagonal blocks we can apply some algebraic multilevel technique, see e.g. [71, 72] and [73], or an algebraic multigrid (AMG) solver, which we utilize in this study, using library-provided AMG implementations.

3.3 Parallelization aspects

As the target solution method is a preconditioned Krylov Subspace iteration, the two major ingredients are the matrix-vector multiplication with the matrix $\mathcal{A}$ and the solution of linear systems with the matrix $\mathcal{P}$. The matrices $\mathcal{A}$ and $\mathcal{P}$ are of dimension qn. These matrices are never assembled but rather we utilize their structure to implement the above two operations efficiently in parallel.

We aim at distributed memory MPI-based implementation of the computations. In the chosen parallel computing environment there are two general strategies to administer the fully implicit IRK method and the preconditioner $\mathcal{P}$ in (3.4).

The first strategy is to distribute the space discretization mesh among a number of processes. This entails that the solution of each system on the diagonal of $\mathcal{P}_d$ is done in parallel in space, using well-known parallelizable methods, such as multilevel, multigrid, domain-decomposition techniques etc, implemented and optimized in some of the established scientific computing libraries. However, in this setting there is no parallelism across the stages. This is the implementation used for the numerical tests in this work.

The second strategy can be based on allocating q groups of processes and in order to use also parallelization in space, within each group replicate the matrices M and K. On the cost of a larger memory footprint and implementing the matrix-vector multiplication in a block-fashion, we can achieve full stage-parallel implementation of the algorithm. This implementation, its efficiency and comparison with the first strategy is a topic of another study, [75].

4 Numerical tests

The parallel tests are run on the Rackham cluster at the Uppsala Multidisciplinary Center for Advanced Computational Science (UPPMAX). Rackham consists of 486 nodes, each node having two 10-core Intel Xeon E5 2630 v4 at 2.20 GHz/core. We use nodes with

128 GB of memory. The test problems are implemented in the `deal.II` FEM library [76], interfacing with PETSc [77] to utilize the available preconditioned iterative methods.

Although implementing the IRK algorithm for time-dependent PDE of parabolic type is straightforward, for completeness we sketch it in Algorithm 1.

Algorithm 1 IRK: heat equation code outline

1. Setup system - assemble block matrices, create preconditioner, initialize multigrid block-solvers.

2. Time-stepping

Assemble initial state

for timesteps **do**

Assemble right-hand-side

Solve system (2.4) using GCR, preconditioned by $\mathcal{P}$ in (3.3), the block systems in $\mathcal{P}_d$ in (3.4) are solved with CG (P1)/GCR (P2) preconditioned by AMG.

We point out that in the tests, presented here, the block systems are solved one after another, parallelizing only in space, however the algorithm allows a stage-parallel implementation, which is the subject of [75].

The performance of the IRK method and the proposed preconditioner are illustrated using the following two test problems.

Problem 1. *Heat equation with a discontinuous initial guess*
Consider the equation

$$\begin{aligned} \frac{\partial u(x,y,t)}{\partial t} - \Delta u(x,y,t) &= f(x,y,t) \quad \text{in } \Omega = (0,1)^2,\ t \in [0,T] \\ u(x,y,0) &= u_{ex}(x,y,0), \quad (x,y) \in \Omega \\ u(x,y,t) &= u_{ex}(x,y,t), \quad (x,y) \in \partial\Omega \end{aligned}$$

and choose

$$\begin{aligned} f(x,y,t) = &\sin(a_x \pi x)\sin(a_y \pi y)(\pi\cos(\pi t) - a_t(1+\sin(\pi t)))\exp(-a_t t) + \\ &(a_x^2 + a_y^2)\pi^2 \sin(a_x \pi x)\sin(a_y \pi y)(1+\sin(\pi t))\exp(-a_t t). \end{aligned}$$

The function $f(x,y,t)$ corresponds to the analytical solution

$$u_{ex}(x,y,t) = \sin(a_x \pi x)\sin(a_y \pi y)(1+\sin(\pi t))e^{-a_t t}.$$

For the tests we choose $a_x = a_y = 2$ and $a_t = 0.5$. For the outer solver we use GCR with stopping criteria $\|P^{-1}(Ax-b)\|_2 < 10^{-8} \cdot q \cdot n_{dofs}$. For the q block-systems in the preconditioner we solve using CG preconditioned by an AMG. We fix $\tau = 0.1$ and take five timesteps. We run tests with the two stage ($q = 2$) and the nine stage method ($q = 9$).

Problem 2. *[Similar to [57]] Consider the two-dimensional time-dependent convection-diffusion equation*

$$\frac{\partial u(x,y,t)}{\partial t} + \sigma(t)(-\Delta u + \boldsymbol{b}\cdot\nabla u - f) = 0 \quad in \;\; \Omega = (0,1)^2, \; t > 0, \tag{4.1}$$

with initial condition $u(x,y,0) = u_0$ *to be the pyramid function shown in Figure 2, boundary conditions*

$$\begin{cases} u = 0 & on\ y = 0, y = 1 \\ u = g(x,y,t), \ell > 1 & on\ x = 1, \\ \dfrac{\partial u}{\partial n} = 0 & on\ x = 0, \end{cases}$$

$\boldsymbol{b}(x,y) = [-\ell, 0]$, $\sigma(t) = 1 + \frac{2}{5}\sin(k\pi t)$, $g(x,y,t) = 2\ell y(1-y)\sin(k\pi t)$ *and* $f(x,y,t) = 2e^{-\ell x}$.

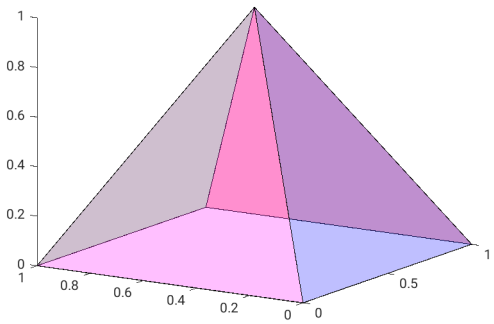

Figure 2: Problem 2: Initial condition

As we deal with nonhomogeneous and time-dependent boundary conditions, we follow the general practice to construct a partial solution $\psi(x,y,t)$ *that satisfies the boundary conditions and then reformulate the problem to find* $v = u - \psi$. *In order to homogenize the boundary conditions we choose*

$$\psi(x,y,t) = e^{-\ell x}x^2 y(1-y)sin(k\pi t).$$

We find then that the function $v(x,y,t) = u(x,y,t) - \psi(x,y,t)$ *is the solution of the initial boundary problem*

$$\frac{\partial v(x,y,t)}{\partial t} + \sigma(t)(-\Delta v + \boldsymbol{b}\cdot\nabla v - \widetilde{f}) = 0 \quad in \;\; \Omega = (0,1)^2, \; t > 0,$$

with boundary conditions

$$\begin{cases} v = 0 & on\ y = 0, y = 1, x = 1, \\ \dfrac{\partial v}{\partial n} = 0 & on\ x = 0, \end{cases}$$

initial condition

$$v(x,y,0) = u_0(x,y) - \psi(x,y,0) = u_0(x,y).$$

and

$$\widetilde{f} = f + \frac{1}{\sigma(t)}\frac{\partial \psi}{\partial t} - \Delta\psi + b\cdot\nabla\psi, \quad \textit{in detail,}$$

$$\widetilde{f}(x,y,t) = -2e^{-\ell x} + \frac{k\pi e^{-\ell x}x^2 y(1-y)\cos(k\pi t)}{1+\frac{2}{5}\sin(k\pi t)} - \left[(\ell^2x^2 - 4\ell x + 2)y(1-y) - 2x^2\right]e^{-\ell x}\sin(k\pi t).$$

The additional complication with the σ-scaling is taken into account when constructing the preconditioner as follows. The system equation, analogous to (2.3)*, now reads*

$$(I_q \otimes M + \tau\Sigma_q A_q \otimes K)k = \widehat{\boldsymbol{f}},$$

where Σ_q is a diagonal matrix, containing the values of σ in the integration points. Multiplying the latter equation by $A_q^{-1}\Sigma_q^{-1}$ from the left we obtain

$$(A_q^{-1}\Sigma_q^{-1} \otimes M + \tau I_q \otimes K)\boldsymbol{k} = (A_q^{-1}\Sigma_q^{-1} \otimes I_n)\widehat{\boldsymbol{f}}.$$

As for Problem 1, we then select the preconditioner to be

$$\mathcal{P} := (L\Sigma_q^{-1} \otimes M + \tau I_q \otimes K).$$

Finally we decompose $L_q\Sigma_q^{-1} = T\Lambda T^{-1}$ and obtain an analogous form of the preconditioner used in Problem 1 with the difference that the eigenvalue decomposition is computed in every time step. The cost of this is low, however, as the matrix in question is of size $q \times q$.

The parallel performance results for Problem 1 for $q = 2$ are presented in Table 3 and for $q = 9$ - in Table 4. The corresponding results for Problem 2 for $q = 2$ are shown in Table 5.

From the timing results for $q = 2$ in Table 3, column 2, we see that the fixed-size speedup is found to be $139/75 = 1.85$ close to the ideal factor 2 and and $139/62 = 2.24$ instead of the ideal factor 3.

From the timing results for $q = 9$ in Table 4, column 2, we observe that the fixed-size speedups is $1613/824 \approx 1.96$ which is very close to the ideal factor 2 and $1613/636 \approx 2.54$ instead of the ideal factor 3.

5 Concluding remarks

Given their fundamental significance in many branches of science, solving time-dependent partial differential equations has been an important question for centuries and it remains an issue with high impact in many scientific computing applications. When handling such problems, the use of high order accurate implicit Runge-Kutta methods of Radau type can be numerically very efficient since the methods are strongly A-stable, enable use of large time-steps and can handle highly ill-conditioned matrices with a widespread spectrum.

No CPUs	Total runtime	Av.out. Aiter.	Av.in. iter.	$\|u_c(\tau_F) - u_a(\tau_F)\|_\infty$	$\|u_c(\tau_F) - u_a(\tau_F)\|_2/n_{dofs}$
Block dimension **16 641**, full system dimension **33 282**					
20	1	2	5	$2.395 \cdot 10^{-4}$	$9.209 \cdot 10^{-7}$
Block dimension **1 050 625**, full system dimension **2 101 250**					
20	11	2	5	$7.014 \cdot 10^{-5}$	$3.418 \cdot 10^{-8}$
Block-dimension **16 785 409**, full system dimension **33 570 818**					
100	37	2	5	$7.461 \cdot 10^{-5}$	$9.103 \cdot 10^{-9}$
Block dimension **67 125 249**, full system dimension **134 250 498**					
100	139	2	5	$7.463 \cdot 10^{-5}$	$4.553 \cdot 10^{-9}$
200	75	2	5	$7.397 \cdot 10^{-5}$	$4.513 \cdot 10^{-9}$
300	62	2	5	$7.330 \cdot 10^{-5}$	$4.470 \cdot 10^{-9}$
Block dimension **268 468 225**, full system dimension **536 936 450**					
600	125	2	6	$7.301 \cdot 10^{-5}$	$2.227 \cdot 10^{-9}$

Table 3: Problem 1: Parallel run times and errors for $q = 2$, solving for 5 time steps with $\tau = 0.1$. Errors evaluated at $\tau_F = 0.5$, parallelism only in space variables

It has been shown that for linear problems the arising globally coupled linear systems on each time-step can be also solved efficiently by a preconditioned iterative method with a high quality stage-parallel (block-diagonal) preconditioner. To construct the preconditioner, we discuss two ways to approximate the inverse of the quadrature matrix by a lower-triangular matrix with real eigenvalues, in this way fully avoiding complex arithmetic.

To solve nonlinear problems, one can use hierarchical basis functions in space and a lower order Radau method for the time-discretization on the coarse mesh and interpolate the solution to the fine mesh. This is planned to be presented in a separate paper. To further increase the efficiency, one can also use adaptive space mesh refinement methods and one can also use lower order Radau methods and coarser meshes to obtain an already quite accurate approximation for the higher order Radau methods and finer mesh space discretizations.

Acknowledgments

The work of the second author (fully) and the third author (partly) is supported by Research Grant VR-2017-03749, '*Mathematics and numerics in PDE-constrained optimization problems with state and control constraints*', financed by the Swedish Research Council. Gratitude is extended to Peter Munch for his help in navigating deal.II.

The computations have been enabled by resources in project SNIC 2021/22-633 provided by the Swedish National Infrastructure for Computing (SNIC) at UPPMAX, partially funded by the Swedish Research Council through grant agreement no. 2018-05973".

No. CPUs	Total runtime	Av.out. iter.	Av.in. iter.	$\|u_c(\tau_F) - u_a(\tau_F)\|_\infty$	$\|u_c(\tau_F) - u_a(\tau_F)\|_2/n_{dofs}$
Block dimension **16 641**, full system dimension **149 769**					
20	6	9	6	$3.144 \cdot 10^{-4}$	$1.209 \cdot 10^{-6}$
Block dimension **1 050 625**, full system dimension **9 455 625**					
20	109	7	6	$2.365 \cdot 10^{-6}$	$1.151 \cdot 10^{-9}$
Block-dimension **16 785 409**, full system dimension **151 068 681**					
300	204	7	7	$5.934 \cdot 10^{-6}$	$7.236 \cdot 10^{-10}$
Block dimension **67 125 249**, full system dimension **604 127 241**					
100	1613	6	7	$2.251 \cdot 10^{-5}$	$1.373 \cdot 10^{-9}$
200	824	6	8	$2.030 \cdot 10^{-5}$	$1.239 \cdot 10^{-9}$
300	636	6	7	$1.686 \cdot 10^{-5}$	$1.029 \cdot 10^{-9}$
Block dimension **268 468 225**, full system dimension **2 416 214 025**					
600	1467	6	8	$1.824 \cdot 10^{-5}$	$5.566 \cdot 10^{-10}$

Table 4: Problem 1: Parallel run times and errors for $q = 9$, solving for 5 time steps with $\tau = 0.1$. Errors evaluated at $\tau_F = 0.5$, parallelism only in space variables

Conflicts of interest

There are no conflicts of interest related to the study, presented in this paper.

References

[1] J. Crank, P.A. Nicolson, A practical method for numerical evaluation of partial differential equations of the heat conduction type, *Mathematical Proceedings of the Cambridge Philosophical Society*, 1 (1947), 50–67.

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

[3] O. Axelsson, A class of A-stable methods. *Nordisk Tidskrift for Informationsbehandling (BIT)* 9 (1969), 185–199.

[4] B.L. Ehle, High order A-stable methods for the numerical solution of D.E.s, *BIT* 8 (1968), 276–278.

[5] J.C. Butcher, Implicit Runge-Kutta processes, *Mathematics of Computation*, 18 (1964), 50–64.

[6] O. Axelsson, Global integration of differential equations through Lobatto quadrature. *Nordisk Tidskrift for Informationsbehandling*, 4 (1964), 69–86.

No. CPUs	Total runtime	Av.out. iter.	Av.in. iter.	No. CPUs	Total runtime	Av.out. iter.	Av.in. iter.
$q = 2, l = 2,\ k = 5, \tau = 0.1$				$q = 4, l = 10,\ k = 10, \tau = 0.001$			
Block dim. **4 198 401**				Block dim. **4 198 401**			
20	64	3	6	20	210	7	6
Block dim. **16 785 409**				Block dim. **16 785 409**			
100	69	3	8	100	191	7	7
Block dim. **67 125 249**				Block dim. **67 125 249**			
100	276	3	8	100	764	6	7
200	143	3	9	200	393	6	7
300	106	3	9	300	288	6	7
Block dim. **268 468 225**				Block dim. **268 468 225**			
600	245	3	10	600	664	6	8

Table 5: Problem 2: Parallel run times and iteration counts, parallelism only in space variables

[7] B.L. Ehle, A-stable methods and Padé approximations to the exponential, *SIAM Journal on Mathematical Analysis* 4 (1973), 671–680.

[8] E. Hairer, G. Wanner, Algebraically stable and implementable Runge-Kutta methods of higher order. *SIAM Journal on Numerical Analysis* 18 (1981), 1098–1108.

[9] E. Hairer, G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, Springer-Verlag, 1991.

[10] E. Hairer, G. Wanner, Stiff differential equations solved by Radau methods, *Journal of Computational and Applied Mathematics* 111 (1999), 93–111.

[11] Y. Saad, *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, PA, 2003.

[12] W. Hundsdorfer, J. Verwer, *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*, Springer-Verlag Berlin Heidelberg, 2003.

[13] J.D. Lambert, *Numerical Methods for Ordinary Differential Systems*, Wiley, New York, 1992.

[14] L. Petzold, Order results for implicit Runge-Kutta methods, applied to differential-algebraic systems, *SIAM Journal on Numerical Analysis*, 23 (1986), 837–852.

[15] O. Axelsson, R. Blaheta, R. Kohut, Preconditioning methods for high-order strongly stable time integration methods with an application for a DAE problem, *Numerical Linear Algebra with Applications* 22 (2015), 930–949.

[16] S. P. Nørsett, Semi-explicit Runge-Kutta Methods, Report Mathematics & Computation, 1974, Department of Mathematics, University of Trondheim, Norway.

[17] Z. Zlatev, Modified diagonally implicit Runge-Kutta methods. *SIAM Journal on Scientific and Statistical Computing* 2 (1981), 321–334.

[18] P.J. van der Houwen, Parallel step-by-step methods. Parallel methods for ordinary differential equations (Grado, 1991). *Applied Numerical Mathematics* 11 (1993), no. 1-3, 69–81.

[19] P.J. van der Houwen, B.P. Sommeijer, Analysis of parallel diagonally implicit iteration of Runge-Kutta methods. Parallel methods for ordinary differential equations (Grado, 1991). *Applied Numerical Mathematics* 11 (1993), 169–188.

[20] O. Axelsson, On the efficiency of a class of A-stable methods. *Nordisk Tidskrift for Informationsbehandling (BIT)* 14 (1974), 279–287.

[21] J.C. Butcher, Diagonally-implicit multi-stage integration methods, *Applied Numerical Mathematics*, 11 (1993), 347–363.

[22] T.J.R. Hughes, G.M. Hulbert. Space-time finite element methods for elastodynamics: Formulations and error estimates. *Computational Methods in Applied Mathematics*, 66 (1988), 339–363.

[23] K. Eriksson, C. Johnson, Adaptive finite element methods for parabolic problems I: A linear model problem. *SIAM Journal on Numerical Analysis* 28 (1991), 43–77.

[24] O. Steinbach, H. Yang, Comparison of algebraic multigrid methods for an adaptive space-time finite-element discretization of the heat equation in 3D and 4D. *Numerical Linear Algebra with Applications* 25 (2018), e2143.

[25] J. Xu, A novel two-grid method for semilinear elliptic equations. *SIAM Journal on Scientific Computing*, 15 (1994), 231–237.

[26] O. Axelsson, On mesh independence and Newton type methods. Proc. International Symposium on Numerical Analyses (ISNA92), Prague, 1992, *Applied Mathematics* 38 (1993), 249–265.

[27] O. Axelsson, W. Layton, A Two–Level Method for the Discretization of Nonlinear Boundary Value Problems. *SIAM Journal on Numerical Analysis*, 33 (1996), 2359–2374.

[28] O. Axelsson, I. Kaporin, On a class of nonlinear equation solvers based on the residual norm reduction over a sequence of affine subspaces. *SIAM Journal on Scientific Computing* 16 (1995), 228–249.

[29] O. Axelsson, S. Sysala, An adaptive Newton method for solving nonlinear partial differential equations. In cooperation with and technically co-sponsored by IEEE PS Computer Society Chapter (IEEE) (2018): 89.

[30] O. Axelsson, M. Neytcheva, A. Ström, An efficient preconditioning method for state box-constrained optimal control problems, *Journal of Numerical Mathematics* 26 (2018), 185–207.

[31] O. Axelsson, M. Neytcheva, Numerical solution methods for implicit Runge-Kutta methods of arbitrarily high order. In P. Frolkovič, K. Mikula, D. Ševčovič, Proceedings of the conference *Algoritmy 2020*, (2020), pp 11-20. `http://www.iam.fmph.uniba.sk/amuc/ojs/index.php/algoritmy/article/view/1593/842`

[32] W. Hoffman, J.J.B de Swart Approximating Runge-Kutta matrices by triangular matrices, *BIT Numerical Mathematics* 37 (1997), 346–354.

[33] W. Pazner, P.-O. Persson, Stage-parallel fully implicit Runge-Kutta solvers for discontinuous Galerkin fluid simulations. *Journal of Computational Physics* 335 (2017), 700–717.

[34] Ernst Hairer, Gerhard Wanner, Construction of Implicit Runge-Kutta Methods. In Solving Ordinary Differential Equations II, 71–90. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.

[35] O. Axelsson. A Generalized Conjugate Gradient, Least Square Method, *Numerische Mathematik*, 51 (1987) 209–227.

[36] J.C. Butcher, *The Numerical Analysis of Ordinary Differential Equations*, Wiley, Chichester (1987).

[37] Y. Notay An aggregation-based algebraic multigrid method. *Electronic Transactions on Numerical Analysis*, 37 (2010), 123–146.

[38] P.S. Vassilevski, *Multilevel Block Factorization Preconditioners*. Springer-Verlag, New York, 2008.

[39] K.R. Jackson, S.P. Nørsett, The potential for parallelism in Runge-Kutta methods. Part I. RK formulas in standard form, *SIAM Journal on Numerical Analysis*, 32 (1995), 49-82.

[40] R. Alexander, Diagonally implicit Runge-Kutta methods for stiff ODEs, *SIAM Journal on Numerical Analysis*, 14 (1977), 1006–1021.

[41] P. van der Houwen, J.J.B. de Swart, Parallel linear system solver for Runge-Kutta methods, *Advances in Computational Mathematics*, 7 (1997), 157–181.

[42] L.O. Jay, T. Braconnier, A parallelizable preconditioner for the iterative solution of implicit Runge-Kutta type method, *Journal of Computational and Applied Mathematics*, 111 (1999), 63–76.

[43] K.-A. Mardal, T. K. Nilssen, G. A. Staff, Order optimal preconditioners for implicit Runge-Kutta schemes applied to parabolic PDEs, *SIAM Journal in Scientific Computing* 29 (2007), 361–375.

[44] G. A. Staff, K.-A. Mardal, T. K. Nilssen, Preconditioning of fully implicit Runge-Kutta schemes for parabolic PDEs, *Model Identification Control* 27 (2006), 109–123.

[45] Xiangmin Jiao, Xuebin Wang, Qiao Chen, Optimal and low-memory near-optimal preconditioning of fully implicit Runge-Kutta schemes for parabolic PDEs, *SIAM J. Sci. Comput.*, 2021 43 (2021), A3527–A3551.

[46] P.J. van der Houwen, W.A. van der Veen, Waveform relaxation methods for implicit differential equations. Parallel methods for ODEs. *Advances in Computational Mathematics* 7 (1997), 183–197.

[47] O. Axelsson, D. Lukáš, Preconditioners for time-harmonic optimal control eddy-current problems. Large-scale scientific computing, 47–54, Lecture Notes in Comput. Sci., 10665, Springer, Cham, 2018.

[48] O. Axelsson, M. Neytcheva, Z.-Z. Liang, Parallel solution methods and preconditioners for evolution equations. *Mathematical Modelling and Analysis*, 23 (2018), 287–308.

[49] D. Sheen, I.H. Sloan, V. Thomée, A parallel method for time-discretization of parabolic problems based on contour integral representation and quadrature, *Mathematics of Computation*, 69 (1999), 177–195.

[50] D. Sheen, I.H. Sloan, V. Thomée, A parallel method for time discretization of parabolic equations based on Laplace transformation and quadrature, *IMA Journal of Numerical Analysis*, 23 (2003), 269–299.

[51] O. Axelsson, J. Verwer, Boundary value techniques for initial value problems in ordinary differential equations, *Mathematics of Computation*, 45 (1985), 153–171.

[52] A. Bellen, M. Zennaro, Parallel algorithms for initial-value problems for difference and differential equations, *Journal of Computational and Applied Mathematics*, 25 (1989), 341–350.

[53] J.-L. Lions, Y. Maday, G. Turinici, A "parareal" in time discretization of PDEs, *Comptes Rendus de l Académie des Sciences* - Series I - Mathematics 332 (2001), 661–668.

[54] M. Bolten, D. Moser, R. Spech, A multigrid perspective on the parallel full approximation scheme in space and time, *Numerical Linear Algebra with Applications* 24 (2017), e2110.

[55] Kaipei Liu, Xiaobing Liao, Yuye Li, Parallel simulation of power systems transient stability based on implicit Runge–Kutta methods and W-transformation, *Electric Power Components and Systems*, 47 (2017), 2246–2256.

[56] M. Rana, V. Howle, K. Long, A. Meek, W. Milestone, A New Block Preconditioner for Implicit Runge-Kutta Methods for Parabolic PDE Problems. `https://arxiv.org/abs/2010.11377`, 2021.

[57] O. Axelsson, R. Blaheta, S. Sysala, B. Ahmad, On the solution of high order stable time integration methods, *Boundary Value Problems* 108 (2013), 22 p.

[58] Ben S. Southworth, Oliver A. Krzysik, Will Pazner, Hans 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.*, Vol 44, No.1, (2022) A416–A443.

[59] J.C. Butcher, On the implementation of implicit Runge-Kutta methods, *BIT Numerical Mathematics*, 16 (1976), 237—240.

[60] T.A. Bickart, An Efficient Solution Process for Implicit Runge–Kutta Methods, *SIAM Journal on Numerical Analysis*, 14 (1977), 1022—1027.

[61] R. Bhatia, *Matrix Analysis*, Berlin. Springer-Verlag New York Berlin Heidelberg, 1997.

[62] M. Eiermann, Field of values and iterative methods, *Linear Alg. Appl.*, 180 (1993), 167–197.

[63] C. Garoni, S. Serra-Capizzano, *Generalized Locally Toeplitz Sequences: Theory and Applications*, Vol. I, Springer, Cham, 2017.

[64] C. Garoni, S. Serra-Capizzano, *Generalized locally Toeplitz sequences: Theory and Applications*, Vol. II. Springer, Cham, 2018.

[65] G. Barbarino, C. Garoni, S. Serra-Capizzano, Block generalized locally Toeplitz sequences: theory and applications in the unidimensional case. *Electron. Trans. Numer. Anal.* 53 (2020), 28–112.

[66] G. Barbarino, C. Garoni, S. Serra-Capizzano, Block generalized locally Toeplitz sequences: theory and applications in the multidimensional case. *Electron. Trans. Numer. Anal.* 53 (2020), 113–216.

[67] Ali Dorostkar, Maya Neytcheva, Stefano Serra-Capizzano, Spectral analysis of coupled PDEs and of their Schur complements via the notion of Generalized Locally Toeplitz sequences, *Computer Methods in Applied Mechanics and Engineering*, 309 (2016), 74-105.

[68] I. Gustafsson, Modified incomplete Cholesky (MIC) methods. *Preconditioning methods: analysis and applications*, 265–293, Topics in Comput. Math., 1, Gordon & Breach, New York, 1983.

[69] O. Axelsson, A general incomplete block-matrix factorization method, *Linear Algebra and its Applications*, 74 (1986), 179–190.

[70] R. Beauwens, Modified incomplete factorization strategies. In: Axelsson O., Kolotilina L.Y. (eds) Preconditioned Conjugate Gradient Methods. Lecture Notes in Mathematics, vol 1457. (1990), Springer, Berlin, Heidelberg.

[71] O. Axelsson, P.S. Vassilevski, Algebraic multilevel preconditioning methods. I. *Numerische Mathematik* 56 (1989), 157–177.

[72] O. Axelsson, M. Neytcheva, Algebraic multilevel iteration method for Stieltjes matrices, *Numer. Linear Algebra Appl.* 1 (1994), 213–236.

[73] O. Axelsson, *Iterative Solution Methods*. Cambridge University Press, Cambridge 1994.

[74] Hao Chen, Kronecker product splitting preconditioners for implicit Runge-Kutta discretizations of viscous wave equations, *Applied Mathematical Modelling*, 40 (2016), 4429–4440.

[75] P. Munch, I. Dravins, Performance study of a stage-parallel implementation of a fully implicit Runge-Kutta method. *Manuscript in progress.*

[76] D. Arndt, W. Bangerth, B. Blais, M. Fehling, R. Gassmöller, T. Heister, L. Heltai, U. Köcher, M. Kronbichler, M. Maier, P. Munch, J. Pelteret, S. Proell, K. Simon, B. Turcksin, D. Wells, J. Zhang, The deal.II Library, Version 9.3, *Journal of Numerical Mathematics*, vol. 29, no. 3, (2021) 171–186.

[77] S. Balay et. al., PETSc/TAO Users Manual, textitArgonne National Laboratory, ANL-21/39 - Revision 3.16 (2021).

A Appendix: Construction of IRM methods

The standard Legendre polynomials of degree q are defined in (1.2), for convenience included once again:

$$L_q(x) = \frac{1}{2^q q!}\frac{d^q}{dx^q}(x^2-1)^q, \tag{A.1}$$

which yields the following expansion, used in the numerical computations:

$$L_q(x) = \frac{1}{2^q}\sum_{k=0}^{\lfloor \frac{q}{2} \rfloor} \frac{(-1)^k(2q-2k)!}{k!(q-k)!(q-2k)!}x^{q-2k}.$$

To exemplify, the polynomials, generated by the above formula are as follows:

q	$L_q(x)$
1	x
2	$\frac{1}{2}(3x^2-1)$
3	$\frac{1}{2}(5x^3-3x)$
4	$\frac{1}{8}(35x^4-30x^2+3)$
5	$\frac{1}{8}(63x^5-70x^3+15x)$
6	$\frac{1}{16}(231x^6-315x^4+105x^2-5)$
7	$\frac{1}{16}(429x^7-693x^5+315x^3-35x)$
8	$\frac{1}{128}(6435x^8-12012x^6+6930x^4-1260x^4+35)$
9	$\frac{1}{128}(12155x^9-25740x^7+18018x^5-4620x^3+315x)$
10	$\frac{1}{256}(46189x^{10}-109395x^8+90090x^6-30030x^4+3465x^2-63)$
11	$\frac{1}{256}(88179x^{11}-230945x^9+218790x^7-90090x^5+15015x^3-693x)$

As is readily seen, these polynomials are normalized at 1.

The shifted Legendre polynomials can be defined, for instance, by the substitution $x = 2y-1, x \in [-1,1],\ y \in [0,1]$:

$$L_q^s(x) = L_{q-1}(2x-1). \tag{A.2}$$

This leads to the following (shifted) polynomials:

q	$L_q^s(x)$
0	1
1	$2x-1$
2	$6x^2-6x+1$
3	$20x^3-30x^2+12x-1$
4	$70x^4-140x^3+90x^2+20x+1$
5	$252x^5-630x^4+560x^3-210x^2+30x-1$
6	$924x^6-2772x^5+3150x^4-1680x^3+420x^2-42x+1$
7	$3432x^7-12012x^6+16632x^5-11550x^4+4200x^3-756x^2+56*x-1$
8	$12870x^8-51480x^7+84084x^6-72072x^5+34650x^4-9240x^3-1260x^2-70x+1$
9	$48620x^9-218790x^8+411840x^7-420420x^6+252252x^5-90090x^4+18480x^3-1980x^2+90x-1$

These polynomials are also normalized at 1. The above definition is equivalent to

$$L_q^s(x) = (-1)^q \sum_{k=0}^{q} \binom{q}{k} \binom{q+k}{k} (-x)^k. \tag{A.3}$$

We form next $P_q(x) = L_q^s(x) - L_{q-1}^s(x)$ and compute the roots of $P_q(x)$ For instance, consider $P_2(x) = 3x^2-4x+1$ and its roots, which are $1/3, 1$. Then we have:

$$A = CVRV^{-1} = \begin{bmatrix} 1 & \\ & 1/3 \end{bmatrix} \begin{bmatrix} 1 & 1/3 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & \\ & 1/2 \end{bmatrix} \begin{bmatrix} 3/2 & -1/2 \\ -3/2 & 3/2 \end{bmatrix} = \begin{bmatrix} 5/12 & -1/12 \\ 3/4 & 1/4 \end{bmatrix}$$

In this paper, to generate the shifted Legendre polynomials we use the formula (A.3).

Based on (A.3), the IRK matrix A_q and its exact inverse can be computed exactly, using symbolic computations, for instance till $q = 12$. We show below the matrices for various q (in rational form):

$$A_3 = \begin{bmatrix} 445/2261 & -112/1709 & 323/13588 \\ 764/1937 & 748/2561 & -323/7774 \\ 1509/4009 & 903/1762 & 1/9 \end{bmatrix}$$

$$A_3^{-1} = \begin{bmatrix} 1277/396 & 995/852 & -1960/7741 \\ -710/199 & 683/881 & 1960/1861 \\ 2163/391 & -2945/391 & 5 \end{bmatrix}$$

$$A_4 = \begin{bmatrix} 246/2177 & -73/1811 & 41/1589 & -80/8077 \\ 409/1745 & 1795/8676 & -67/1400 & 111/6917 \\ 239/1103 & 1101/2711 & 569/3010 & -1227/50740 \\ 334/1515 & 217/559 & 293/891 & 1/16 \end{bmatrix}$$

$$A_4^{-1} = \begin{bmatrix} 2252/399 & 1031/536 & -1654/2823 & 217/1248 \\ -2565/508 & 1088/891 & 937/534 & -2137/4915 \\ 4404/1261 & -1287/323 & 551/868 & 2387/1310 \\ -3891/562 & 277/42 & -4394/361 & 17/2 \end{bmatrix}$$

$$A_5 = \begin{bmatrix} 259/3548 & -109/4077 & 131/7014 & -31/2407 & 103/20425 \\ 167/1086 & 479/3276 & -262/7189 & 83/3909 & -68/8569 \\ 222/1585 & 1592/5325 & 517/3085 & -244/7183 & 78/7127 \\ 315/2174 & 553/2000 & 245/752 & 407/3161 & -117/7448 \\ 863/6005 & 83/295 & 2344/7517 & 253/1134 & 1/25 \end{bmatrix}$$

$$A_5^{-1} = \begin{bmatrix} 3695/422 & 1579/546 & -589/673 & 271/678 & -263/1967 \\ -2485/347 & 1248/691 & 1345/569 & -226/261 & 1347/4910 \\ 2362/573 & -2257/502 & 1621/1892 & 3780/1501 & -935/1423 \\ -927/239 & 2874/847 & -1157/223 & 415/714 & 1745/621 \\ 1489/177 & -1875/269 & 1457/166 & -2077/114 & 13 \end{bmatrix}$$

The roots for $q = 9$ are

$$\begin{array}{lll} 0.017779915147363 & 0.091323607899794 & 0.214308479395631 \\ 0.371932164583272 & 0.545186684803427 & 0.713175242855569 \\ 0.855633742957854 & 0.955366044710030 & 1.000000000000000 \end{array}$$

and the matrices A_9 and A_9^{-1} are shown in Table 6. It is straightforward to check that the row sum of the matrices is equal to the vector of the interpolation points.

B Appendix: Computing eigenvectors of lower-triangular matrices

```
function [Lan,V] = Lower_trian_eigenvectors(L);
q = size(L,1);
V=eye(q,q);
for k=1:q
    for l=k+1:q
        for j=k:l-1
            dd = 1/(L(l,l)-L(k,k));
            if dd>0
                V(l,k)=V(l,k)-L(l,j)*V(j,k)*dd;
            else
                V(l,k)=V(l,k)+L(l,j)*V(j,k)*(-dd);
            end
        end
    end
end
```

$$A_9 = \begin{bmatrix} 178/7811 & -31/3609 & 71/11006 & -59/11222 & 47/10709 & -59/16159 & 88/29927 & -150/69791 & 14/16301 \\ 365/7463 & 65/1282 & -71/5250 & 147/15962 & -163/22779 & 275/47849 & -36/7925 & 33/10036 & -89/67987 \\ 151/3452 & 287/2650 & 262/3593 & -95/5628 & 55/5138 & -49/6201 & 53/8846 & -57/13418 & 19/11322 \\ 455/9838 & 1109/11485 & 472/3059 & 318/3667 & -281/15229 & 28/2537 & -152/19809 & 141/26969 & -11/5403 \\ 677/15100 & 102/997 & 183/1324 & 327/1804 & 536/5927 & -452/24993 & 107/10497 & -74/11553 & 406/167273 \\ 498/10907 & 963/9713 & 317/2175 & 253/1546 & 1188/6389 & 224/2679 & -183/11575 & 81/9953 & -63/21646 \\ 89/1969 & 189/1874 & 133/937 & 675/3943 & 345/2032 & 533/3177 & 141/2102 & -126/10685 & 136/37681 \\ 217/4778 & 497/4967 & 619/4309 & 352/2095 & 227/1293 & 241/1546 & 571/4430 & 198/4625 & -89/18036 \\ 106/2337 & 145/1446 & 365/2549 & 555/3287 & 237/1361 & 253/1597 & 1704/13787 & 332/4497 & 1/81 \end{bmatrix}$$

$$A_9^{-1} = \begin{bmatrix} 2081/74 & 3943/442 & -2414/927 & 673/565 & -579/863 & 502/1183 & -309/1088 & 323/1740 & -151/2132 \\ -1057/51 & 3838/701 & 3793/598 & -3153/1376 & 978/823 & -413/571 & 261/550 & -712/2319 & 467/4002 \\ 4315/434 & -886/85 & 3138/1345 & 2348/449 & -723/346 & 1046/905 & -5067/6989 & 430/937 & -714/4127 \\ -2577/385 & 2422/437 & -939/122 & 652/485 & 1496/309 & -945/461 & 1186/1017 & -326/461 & 559/2129 \\ 4335/809 & -3501/857 & 5617/1285 & -2257/328 & 509/555 & 21967/4425 & -756/349 & 855/713 & -635/1469 \\ 13455/2761 & 2113/591 & -1408/405 & 1018/243 & -2791/391 & 1222/1743 & 7087/1252 & -1186/487 & 1957/2381 \\ 2232/445 & -790/219 & 939/280 & -1521/415 & 3899/814 & -15547/1786 & 142/243 & 4899/668 & -970/479 \\ -1244/203 & 3224/739 & -2972/749 & 673/162 & -14929/3012 & 13994/1999 & -1412/103 & 997/1905 & 4585/512 \\ 3849/263 & -3892/375 & 3221/344 & -1651/171 & 2863/256 & -1331/90 & 2630/111 & -6334/113 & 41 \end{bmatrix}$$

Table 6: Matrices A_9 and A_9^{-1}

C Appendix: Briefly on the Generalized Locally Toeplitz theory

For clarity of the presentation we include some details regarding the Generalized Locally Toeplitz theory (GLT). Most discretization techniques used to approximate PDEs, such as the Finite Difference method (FDM), the Finite Element method (FEM), the Isogeometric Analysis (IgA) lead to sequences of sparse matrices that admit either Toeplitz or (block) multilevel Toeplitz structure. As is well studied, under quite general assumptions, sequences of (block) Toeplitz matrices can be associated with analytic functions, referred to as 'matrix symbols'. The Generalized Locally Toeplitz theory broadens the applicability of the matrix symbols in the cases for variable coefficients, non-uniform gridding and more. Applying the theory of block multilevel Toeplitz matrix sequences could give additional and deeper understanding of the spectral behavior of the so-arising large matrices, some functions of these matrices and in particular, their preconditioned form, when analysing the properties of various preconditioning techniques.

Five main features of the GLT class of matrices are most relevant and used in the analysis of the preconditioned systems, considered in this paper. For details and historical developments of the technique see [63, 64] and the references therein.

GLT1 Each GLT sequence has a symbol f. If the sequence is Hermitian then f describes the asymptotic behaviour of its eigenvalues (up to a finite number of outliers). Otherwise $|f| = (f^*f)^{1/2}$ describes the asymptotic behaviour of the singular values.

GLT2 The set of GLT sequences form a $*$-algebra (close under linear combinations, conjugation, products, inversion (whenever the symbol vanishes, at most, in a set of zero Lebesgue measure)). Hence, the sequence obtained via algebraic operations on a finite set of input GLT sequences is still a GLT sequence and its symbol is obtained by the same algebraic manipulations on the corresponding symbols of the input GLT sequences.

GLT3 Every Toeplitz sequence generated by a L^1 function f is a GLT sequence and its symbol is f, possessing the properties from in item **GLT1**.

GLT4 The approximation of PDEs with non-constant coefficients, general domains, nonuniform gridding by local methods (FDM, FEM, IgA etc), under very mild assumptions leads also to GLT sequences.

GLT5 GLT structures are encountered for certain matrix sequences, related to preconditioners, based on approximations of PDEs by local methods. Moreover, the symbol includes information about the coefficients and the domain of the PDE, as well as information on the discretization schemes for the derivatives including the used meshes, which have to be described, at least asymptotically, as a map of a reference equispaced mesh, developed for multi-dimensional settings. Furthermore, also in presence of non-dominating advection terms the distribution result for the eigenvalues can be recovered.

D Appendix: Eigenvalue examples

We include additional examples of eigenvalues of the preconditioned block system. Figures 3-4 illustrate the eigenvalues for nine different choices of τ for the four stage method, the plots illustrate a typical behavior where the spectrum is close to one for small and large values of τ while displaying a wider range in between. Note however that in observed spectra the magnitude of complex part remains smaller than 0.2 while the real part remains in the interval $[0.6, 1.1]$. For practical applications one may choose the value in τ such

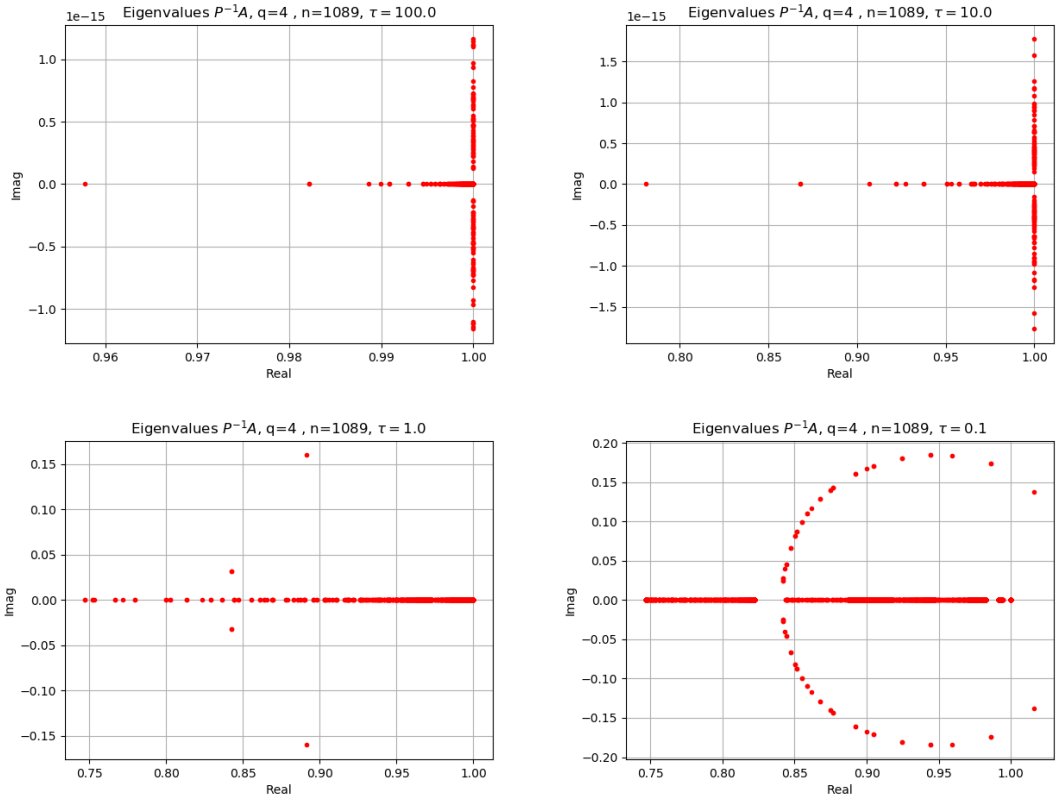

Figure 3: Problem 1, $q = 4$: Eigenvalues of $\mathcal{P}^{-1}\mathcal{A}$ for various values of τ

that the time-error matches the spacial error. In [56] the relation

$$\tau = h^{\frac{p+1}{2q-1}}$$

is used. Here p denotes the degree of the Lagrange polynomial used in the discretization, in all cases considered in this work we have $p = 1$. Using this choice of τ we plot the eigenvalues of the preconditioned system for $q = 1, 2, ..., 10$. In Figures 5-6 we see how both the real and complex ranges of eigenvalues increase with the number of stages.

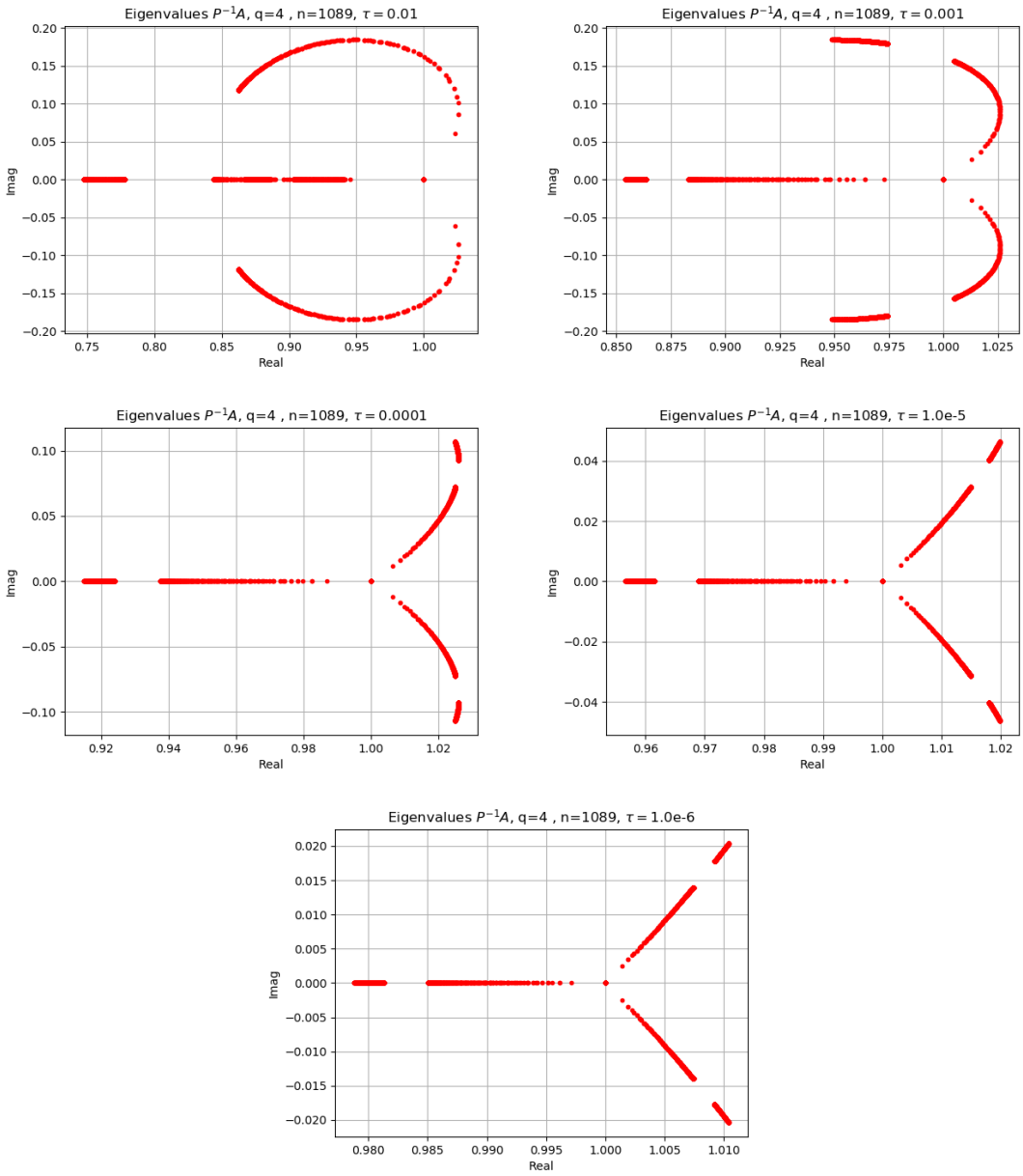

Figure 4: Problem 1, $q = 4$: Eigenvalues of $\mathcal{P}^{-1}\mathcal{A}$ for various values of τ

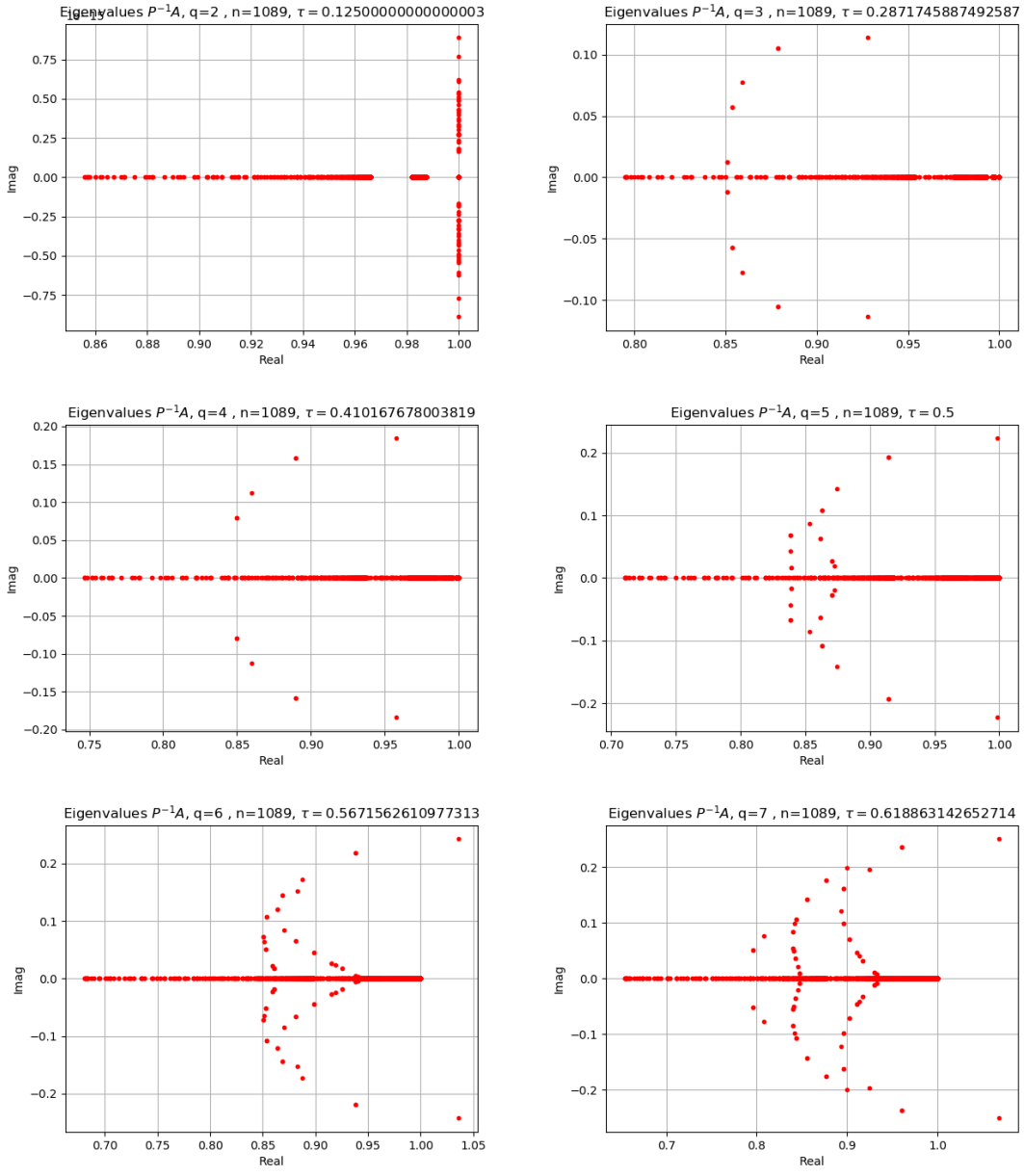

Figure 5: Problem 1,$\tau = h^{\frac{p+1}{2q-1}}$: Eigenvalues of $\mathcal{P}^{-1}\mathcal{A}$ for various values of q

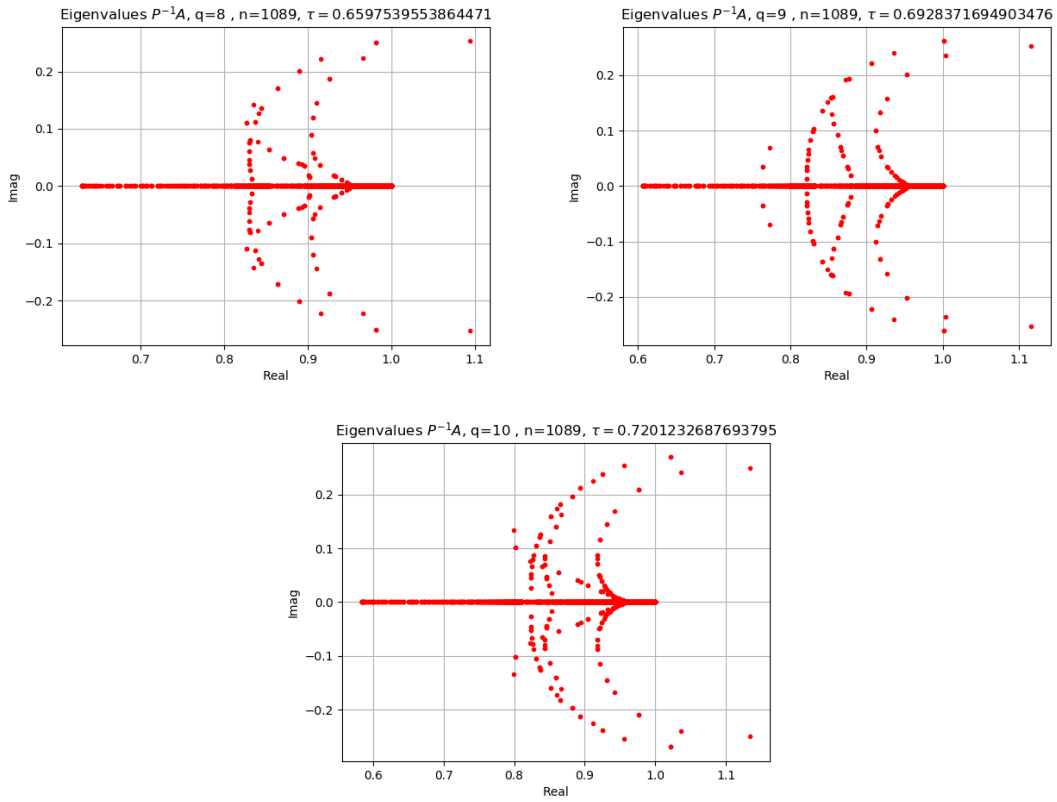

Figure 6: Problem 1,$\tau = h^{\dfrac{p+1}{2q-1}}$: Eigenvalues of $\mathcal{P}^{-1}\mathcal{A}$ for various values of q