A Review of Time Stepping Techniques and Preconditioning for Hyperbolic and Anisotropic Elliptic Problems

Technical Report 2068625-TN-03
Deliverables D1.1 and D2.1

Hussam Al Daas[*] Niall Bootland[*] Tyrone Rees[*] Sue Thorne[†]

March 2023

1 Introduction

Exascale targeted plasma modelling will require the efficient and accurate solution of systems of hyperbolic partial differential equations (PDEs), along with corresponding elliptic problems, in the presence of highly anisotropic dynamics. Furthermore, the techniques used must scale with the computing power available, and be robust enough to be portable to any emerging hardware that arises in the future. We are anticipating that high order finite elements/spectral elements will be used to discretize the equations, and we focus on methods amenable to such problems.

In Section 2, we investigate the current state of the art in time advance techniques, while in Section 3, we turn our attention to the state of the art for preconditioners of elliptic systems. We give a high level summary of our findings at the end of each section.

2 State-of-the-art time stepping techniques for hyperbolic PDEs

We consider the solution of hyperbolic systems of the form

$$\frac{\partial u}{\partial t} = \mathcal{A}(u,t),$$

together with appropriate initial and boundary conditions. Stability requirements mean that explicit methods (such as forward Euler) would require an unfeasibly small temporal step size. This problem, which is compounded by the fact that the step size for the *entire domain* is restricted by the finest mesh patch or wave velocity, generally makes explicit methods unsuitable for anisotropic hyperbolic equations.

We can make use of larger time steps with an implicit (or semi-implicit) method. While such schemes may be unconditionally stable, even in this case we must still restrict the size of the time step with a CFL-like condition to ensure that the solution is sufficiently accurate. Also, with any implicit method, there is the requirement to solve a large system of equations at each time step, and we must do this carefully to ensure performance on modern HPC systems.

2.1 Fully Implicit Methods

The Method of Lines [49] is a technique to transform a PDE into system of ordinary differential equations (ODEs) by applying a pre-determined discretization strategy to the spatial dimensions of the PDE. We may then solve the resulting ODE with an appropriate temporal scheme to the required accuracy.

[*]Scientific Computing Department, STFC Rutherford Appleton Laboratory, Harwell Campus, Didcot, OX11 0QX, UK.
[†]Hartree Centre, STFC Rutherford Appleton Laboratory, Harwell Campus, Didcot, OX11 0QX, UK. Email contact: `sue.thorne@stfc.ac.uk`

In the linear case, and without algebraic constraints, the ODE system takes the form

$$Mu'(t) = \mathcal{L}u + \hat{f}(t) \quad \text{in } (0, T], \quad u(0) = 0, \tag{1}$$

where $M \in \mathbb{R}^{n\times n}$ is a mass matrix, $\mathcal{L} \in \mathbb{R}^{n\times n}$ is a discrete linear operator, and $\hat{f}(t)$ a time-dependent forcing function.

One may solve the ODE (1) by applying an s-stage Runge–Kutta scheme:

$$u_{n+1} = u_n + \delta t \sum_{i=1}^{s} b_i k_i, \tag{2}$$

$$Mk_i = \mathcal{L}\left(u_n + \delta t \sum_{i=1}^{s} a_{ij} k_j\right) + f(t_n + \delta t c_i). \tag{3}$$

Such schemes are commonly expressed in terms of a Runge–Kutta matrix $A = \{a_{ij}\} \in \mathbb{R}^{s\times s}$, weight vector $b^T = (b_1, \dots, b_s)^T$, and quadrature nodes $c = (c_1, \dots, c_s)$, often presented in a Butcher tableau:

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array}.$$

The stage vectors $\{k_i\}$ are the solution of the block linear system,

$$\left(\begin{bmatrix} M & & \\ & \ddots & \\ & & M \end{bmatrix} - \delta t \begin{bmatrix} a_{11}\mathcal{L} & \cdots & a_{1s}\mathcal{L} \\ \vdots & \ddots & \vdots \\ a_{s1}\mathcal{L} & \cdots & a_{ss}\mathcal{L} \end{bmatrix}\right) \begin{bmatrix} k_1 \\ \vdots \\ k_s \end{bmatrix} = \begin{bmatrix} f_1 \\ \vdots \\ f_s \end{bmatrix}, \tag{4}$$

where $f_i := \hat{f}(t_n + \delta t c_i) + \mathcal{L}(t_n + \delta t c_i)u_n$. The difficulty in applying fully implicit Runge–Kutta methods lies in solving the $ns \times ns$ block linear system (4).

The solution of nonlinear systems requires solves with the form (4), but with $\mathcal{L}$ now being a linearized operator as determined by, say, a Newton or Picard iteration [82, 48].

We may also incorporate constraints (such as incompressibility): if the ‘Method of Lines’ applied to a PDE gives the differential algebraic equation (DAE)

$$\begin{aligned} Mu'(t) &= \mathcal{N}(u, w, t) \\ 0 &= \mathcal{G}(u, w, t), \end{aligned}$$

then an s-stage Runge–Kutta method applied to this gives iterates of the form

$$\begin{bmatrix} u_{n+1} \\ w_{n+1} \end{bmatrix} = \begin{bmatrix} u_n \\ w_n \end{bmatrix} + \delta t \sum_{i=1}^{s} b_i \begin{bmatrix} k_i \\ \ell_i \end{bmatrix}.$$

In the linear case we obtain the stage vectors k_i and ℓ_i by solving the system of equations

$$\left(\begin{bmatrix} \begin{bmatrix} M & \\ & 0 \end{bmatrix} & & \\ & \ddots & \\ & & \begin{bmatrix} M & \\ & 0 \end{bmatrix} \end{bmatrix} - \delta t \begin{bmatrix} a_{11}\begin{bmatrix} \mathcal{N}_u & \mathcal{N}_w \\ \mathcal{G}_u & \mathcal{G}_w \end{bmatrix} & \cdots & a_{1s}\begin{bmatrix} \mathcal{N}_u & \mathcal{N}_w \\ \mathcal{G}_u & \mathcal{G}_w \end{bmatrix} \\ \vdots & \ddots & \vdots \\ a_{s1}\begin{bmatrix} \mathcal{N}_u & \mathcal{N}_w \\ \mathcal{G}_u & \mathcal{G}_w \end{bmatrix} & \cdots & a_{ss}\begin{bmatrix} \mathcal{N}_u & \mathcal{N}_w \\ \mathcal{G}_u & \mathcal{G}_w \end{bmatrix} \end{bmatrix}\right) \begin{bmatrix} k_1 \\ \ell_1 \\ \vdots \\ k_s \\ \ell_s \end{bmatrix} = \begin{bmatrix} f_1 \\ g_1 \\ \vdots \\ f_s \\ g_s \end{bmatrix}. \tag{5}$$

Again, the nonlinear case is also possible, and results in a series of linearized systems of the form (5). For more details, see e.g., [9, Section 10.1.3], [82, Section 6]. In the following, for simplicity of exposition, we consider only the linear case without constraints (unless we state otherwise), but the ideas can also be applied in the more general case.

For general properties of such methods, we refer the reader to Ascher and Petzold [9, Section 4]. In the specific case that interests us, the algebraic block is very large and ill conditioned, being the spatial discretization of a partial differential equation. We can expect standard implementations of algorithms for solving ODEs to fail, as direct methods would struggle to solve even a single system with $\mathcal{L}$. Below we describe the state of the art.

2.1.1 Diagonally-implicit Runge–Kutta (DIRK) methods

Diagonally-implicit Runge–Kutta (DIRK) methods have a lower triangular Runge–Kutta matrix, A. This simplifies the solution of (4) considerably, as the implicit solve requires only a series of $n \times n$ systems, rather than one $ns \times ns$ system solve. Kennedy and Carpenter [45] performed a comprehensive survey of DIRK methods for NASA, followed up by the development of several new methods [46]. We recommend these papers, and the references therein, for more detail on this class of methods.

DIRK methods can be further divided into a series of subclasses: SDIRK methods are DIRK methods where A has a constant diagonal; EDIRK methods are DIRK methods with an explicit first stage (so $a_{1,1} = 0$); ESDIRK methods [50] are EDIRK methods where the non-zeros on the diagonal are constant. Nektar++ implements SDIRK with two and three stages, and an ESDIRK method with six stages; Yan et al. [90] gives a preliminary comparison of the performance of these methods against each other and against explicit methods.

Pan et al. [66] point out that one must be careful when choosing the time step, as making a naive choice may lead to extra work without any gain in accuracy (see, e.g., [62]). They observe that, when solving Navier–Stokes equations using a Discontinuous Galerkin discretization in the spatial domain, an ESDIRK method in temporal domain, and solving the nonlinear system using a Jacobian Free-Newton Krylov method (JFNK) [48], then the error in one time step is the sum of the local temporal error (from ESDIRK), and the averaged spatial error (from DG) and the averaged algebraic JFNK error (from preconditioned GMRES). They therefore propose a system, implemented in Nektar++ [90], which uses explicit formulae for the spatial truncation error and temporal error to estimate the errors in these components. These are then used to choose *a priori* a time step and a Newton tolerance such that the temporal and iterative errors are smaller than the spatial errors, thus ensuring that the accuracy is enough to ensure sufficient convergence, but not too much to waste CPU time. Since this method is based on local errors, there is no guarantee on the global errors; however, it suggests a reasonable heuristic which has been shown to be effective on the isentropic vortex, Taylor-Green vortex, flat plate boundary layer, and turbulent flow over a cylinder model problems. While this approach provides an upper bound for the time step, this may not be enough to maintain stability for challenging problems, yet the method is a promising approach for automatic simulation pipelines.

2.1.2 High order implicit Runge–Kutta (IRK) method

While DIRK schemes are attractive, since they simplify the structure of the linear systems, they come with a number of drawbacks, as outlined in [45]. For a DIRK method with formal integration order p and stage-order q, the order of accuracy observed in practice for stiff nonlinear PDEs or DAEs is approximately $\min\{p, q + 1\}$. Stable DIRK methods have a maximum order of $p = s + 1$, and the stage-order, q, is usually 1, although can be 2 for DIRK methods with an explicit first stage. However, DIRK methods cannot have a stage-order larger than 2. Symplectic DIRK methods can be at most 4th order, and have the additional restriction that documented methods above order two have Runge–Kutta matrices with negatives on the diagonal, which usually leads to more difficult linear systems to solve. Overall, DIRK methods have limited accuracy and, while this is often good enough for the application at hand, it can be an issue for difficult modelling problems.

Fully implicit Runge–Kutta (IRK) methods, in contrast, can have high-order accuracy, since they may have any stage-order: an s-stage IRK method may have accuracy of order $2s$. However, the linear system solved at each step (4) is formidable, and is intractable for realistic sized problems without careful handling of the linear algebra. Common IRK methods include the Gauss–Legendre, Gauss–Lobatto and Gauss–Radau families, which contain classical techniques such as backward Euler (RadauIIA), and Crank–Nicolson (LobattoIIIA).

Farrell et al. [30] have developed a high-level library called Irksome for manipulating UFL (Unified Form Language) expressions of semidiscrete variational forms to obtain UFL expressions for the coupled Runge–Kutta stage equations (3) at each time step. Irksome works with the Firedrake package to enable the efficient solution of the resulting coupled algebraic systems, which are solved matrix-free. Irksome solves the matrix (4) using a Krylov method with preconditioner

$$\mathrm{blkdiag}\left(M - a_{1,1}\delta t\mathcal{L}, \cdots, M - a_{s,s}\delta t\mathcal{L}\right),$$

which has been shown [55] to be a good preconditioner for parabolic problems.

The system (4) can be re-written in Kronecker product form as

$$(I \otimes M - \delta t A \otimes \mathcal{L})k = f, \tag{6}$$

which is the Sylvester matrix equation. Such systems arise commonly in model order reduction and control applications, and we refer the reader to Simoncini's survey [80] on techniques for solving matrix equations. This connection has led to a number of interesting approaches for the solution of (4) using fully-implicit IRK methods over the past few years [83, 82, 42, 56, 71].

Southworth et al. introduce a theoretical and algorithmic preconditioning framework for solving (4) in the linear [83] and nonlinear [82] setting. This framework also naturally applies to discontinuous Galerkin discretizations in time. Under quite general assumptions on the spatial discretization that yield stable time integration, they prove that the preconditioned operator has a condition number bounded by a small, order-one constant, independent of the spatial mesh and time-step size, and with only weak dependence on number of stages/polynomial order; for example, the preconditioned operator for 10th-order Gauss IRK has a condition number less than two, independent of the spatial discretization and time step.

The proposed method can be used with arbitrary existing preconditioners for backward Euler-type time-stepping schemes and is amenable to the use of three-term recursion Krylov methods when the underlying spatial discretization is symmetric. In [83], the authors apply their method to various high-order finite difference and finite element discretizations of linear parabolic and hyperbolic problems, demonstrating fast, scalable solution with up to 10th-order accuracy. The proposed method, in several cases, can achieve 4th-order accuracy using Gauss integration with roughly half the number of preconditioner applications and wall-clock time than as is required using standard DIRK methods. In [82] the authors treat the nonlinear case along with the problem of DAEs, applying the method to the nonlinear Navier–Stokes equation. Again, the method only requires an efficient preconditioner for matrices arising from the classical backward Euler scheme.

The numerical experiments in [83, 82] used the software MFEM [8], and employs multigrid preconditioners that are available within HYPRE (see Section 3). We highlight that Masud et al. [56] presented a similar approach to that of Southworth et al., treating the parabolic case.

Common to all the approaches described here are fast methods for the solution of a sequence of linear systems of the form

$$(\alpha_i M - \mathcal{L})x = f_i \text{ or } \begin{bmatrix} \alpha_i M - \mathcal{L} & B^T \\ B & C \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} f_{i1} \\ f_{i2} \end{bmatrix}, \tag{7}$$

depending on if constraints are present or not, where M is a mass matrix, $\mathcal{L}$ is a linear differential operator, and B, B^T and C represent the differential algebraic constraints. The parameters α_i depend on the values in the Runge–Kutta matrix A and can be assumed to be positive. The symmetry and definiteness of (7) therefore depends on the linear differential operator $\mathcal{L}$. We point to Section 3 for a description of recent advances in preconditioning (7) for both the symmetric and nonsymmetric case.

2.2 Linear Multistep Methods

Linear multistep time integration techniques applied to (1) take the form

$$\sum_{j=-k}^{0} \alpha_{k+j} M u_{i+j} = \delta t \sum_{j=-k}^{0} \beta_{k+j} \left(\mathcal{L}(u_{i+j}) + f(t_{i+j})\right),$$

for a given set of α_i and β_j. Methods with $\beta_k = 0$ are explicit, for example the Adams–Bashforth (AB) family, while non-zero β_k give implicit methods, for example the Adams–Moulton (AM) and backward differentiation formulae (BDF) families).

There are, however, a number of drawbacks which limit their application to hyperbolic PDEs. Although they do not suffer from the same accuracy issues of DIRK methods, there are no implicit multistep methods which are A-stable and of order greater than two. There are also no generally symplectic multistep methods. By the nature of multistep methods, which build approximations to the solution using solutions at previous points in time, multistep methods are more memory intensive than Runge–Kutta methods and so may not be so applicable for large simulations at exascale.

2.3 Implicit–Explicit (IMEX) methods

It is often the case that time-dependent PDEs take the form

$$\frac{\partial u}{\partial t} = \mathcal{B}(u,t) + \mathcal{C}(u,t),$$

where the operators $\mathcal{B}(u,t)$ and $\mathcal{C}(u,t)$ model phenomena present at different time-scales; examples include a non-stationary convection–diffusion equation where we follow the convected time-scale, where $\mathcal{B}(u,t)$ and $\mathcal{C}(u,t)$ may represent the diffusive and convective terms, respectively, or coupled multi-physics problems, where mixed fluids may have different behaviours.

The presence of the fast term precludes the use of fully explicit integrators, as we would require a prohibitively small time-step to ensure convergence. On the other hand, using a fully implicit method may ask too much of the linear (or nonlinear) solvers. Implicit–Explicit (IMEX) time integration schemes exploit the structure of the problem by solving for the slow term explicitly, where we may safely take larger time steps, and solving for the fast term implicitly. The split between the two different regimes is not always clear, and schemes may leak stiffness. IMEX methods also may suffer from accuracy issues, and fully implicit methods may need to be used if a tight tolerance is required.

A number of IMEX schemes have recently been proposed for general hyperbolic problems[38, 20, 67], for multi-physics problems (see [47, Section 3.2.2] and the references therein), and for plasma modelling [59]. Nektar++ implements a scheme by Karniadakis et al.[43] for thermal convection problems[41]; we note that Nektar++ uses an equivalence between IMEX methods and general linear methods to ease the switch between various implicit, explicit and IMEX methods [87]. The PETSc TS package [12, Section 6] provides interfaces to the IMEX methods proposed in [10, 20, 37, 44, 67] (mostly using an SDIRK implicit solver). We would like to highlight that IMEX methods (together with bespoke preconditioners for the implicit solves) have recently been successfully applied at scale as the integrator in the Unified Model [57, 16].

2.4 Parallel-in-time

As the parallel performance of spatial solvers has become saturated, there has been an increasing interest in parallel in time (PinT) methods; see the survey by Gander [32]. The development of the parareal method by Lions, Maday and Turinici in 2001 [51] was the catalyst for much of this activity. The parareal algorithm combines two solvers; a cheap, global 'coarse' solver, and a more accurate 'fine' solver. The basic concept is simple: we employ a fine solver in parallel, which constitutes the bulk of the computational effort, and then combine the results into a global solution using the coarse solver. While most results in the literature apply this method to fairly simple model problems and, in particular, those of a diffusive nature, there has been some success in applying the parareal paradigm to real scientific applications with a hyperbolic PDE, most notably in the work of Samaddar et al. on tokamak edge plasma simulations [79, 78, 77, 15, 75, 27, 74, 76]. Key in this work is the selection of an appropriate coarse grid solver, and unfortunately there is currently little (if any) theoretical guidance on which schemes may prove successful for other problems. Nevertheless, these studies report a speed up of roughly a factor of ten by using parareal over conventional time-stepping techniques.

An alternative approach to parareal is multigrid reduction in time (MGRIT) [29], a method built on the equivalence between a traditional time integration method and a block lower triangular system of equations. The parareal method is equivalent to a two grid (in time) multigrid method [33] and MGRIT is, in some ways, the natural extension, considering a hierarchy of grids. XBraid software [1] is an implementation of MGRIT, and has been shown to give speed-ups of up to a factor of fifty over sequential time-stepping.

Both parareal and MGRIT struggle on hyperbolic problems. Southworth et al. [81] recently provided a convergence analysis detailing whether or not parareal or two-level MGRIT will converge on a given problem, as well as what spatial or temporal discretizations are applicable with MGRIT. Recent work by Wathen and collaborators [58, 24] and Gander et al. [34], based on the development of block preconditioners for the large block lower-triangular time-stepping matrix, has potential to overcome this limitation.

We highlight there is an ExCALIBUR project to compare the performance of parallel in time methods for exascale use, the findings of which will be relevant here. `https://excalibur.ac.uk/projects/exposing-parallelism-parallel-in-time/`

Summary

- DIRK time-stepping methods, where applicable, are well developed and the review [45] provides a nice summary.
- Fully implicit Runge–Kutta methods were largely dismissed as impractical for hyperbolic problems, but recent advances in preconditioning techniques may mean this is no-longer the case; Southworth et al. [83, 82] describes (and advances) the state of the art here.
- The theory of IMEX methods is also well developed and should be used where appropriate.
- Recent work on parallel-in-time methods may help break the inherently sequential nature of traditional time integration algorithms, which rely exclusively on the solution of the spatial discretization for parallelism.
- All approaches for solving implicit time-stepping problems require good methods for solving (variations of) the stationary system; see Section 3.

3 State-of-the-art approaches for preconditioning elliptic problems discretized with high-order methods

Existing and emerging computing architectures suffer from an exponentially growing gap between the time necessary to perform a floating point operation and the time to move data across the network on a distributed memory environment. High-order numerical methods provide higher accuracy with fewer degrees of freedom, at the cost of more arithmetic operations performed per degree of freedom. Because of their high arithmetic intensity, high-order methods are promising candidates to get the best performance on exascale systems.

A finite element or discontinuous Galerkin method of polynomial degree p in d spatial dimensions will have $\mathcal{O}(p^d)$ degrees of freedom per element, and hence the system matrix will have $\mathcal{O}(p^{2d})$ non-zero entries. For this reason it is often the case that the coefficient matrix is not assembled and associated operations are carried out in a matrix-free fashion. A matrix-free method reduces the memory requirements to $\mathcal{O}(p^d)$ and, making use of sum factorization techniques, the cost of a matrix–vector product can be reduced to $\mathcal{O}(dp^{d+1})$[65].

The condition number of the discretized matrix increases quadratically with the polynomial degree. Due to the large bandwidth, direct methods for solving linear systems are not an option. We therefore must turn to iterative solvers and it is vital that we pair the Krylov subspace solver with an appropriate preconditioner to get good performance.

In this section, we summarize the state-of-the-art in techniques proposed to precondition linear systems arising from the discretization of elliptic PDEs with high-order methods. We point the reader to the NEPTUNE report [7] for an overview of general-purpose preconditioners, as well as the excellent review by Wathen[88].

3.1 Preconditioning high order finite element matrices

There is a good body of work on preconditioners for low- to moderate-order finite elements and there are some excellent implementations of algorithms which have been used in a wide variety of practical situations; see Section 3.2. However, it has been observed that such methods often give less than desirable performance when applied to high-order discretization (however, see Heys et al. [40], who show that AMG can be applied successfully—albeit with a mild p-dependence—through minor modifications).

One technique that has proved successful, first proposed by Orszag in 1980 [65], is to exploit the spectral equivalence between low-order and high-order operators, known as FEM–SEM equivalence; see the review by Canuto, Gervasio and Quarteroni[22] and the references therein. This approach allows us to use a low-order discretization to precondition the high-order problem, allowing the use of methods outlined in Section 3.2 as a fast approximation to this ‘ideal’, low-order, preconditioner.

In practice, the tensor-product of the Gauss–Lobatto–Legendre points used to generate the grid for spectral element methods result in anisotropic finite element meshes, which challenge basic multigrid and domain decomposition methods[52]. Nevertheless, there has been considerable success in the development of such methods in recent years: Pazner and Kolev [69, 70] develop a multigrid preconditioner for high-order continuous and discontinuous Galerkin methods with hp-refinement; Chalmers and Warburton[23] and Olson[63] apply this idea to simplexes in two and three dimensions; Bello-Maldonado and Fischer[14] create a scheme that uses a preconditioner based on $P1$-elements, using a meshing technique for rectangular and hexahedral elements; Pazner, Dohrmann and Kolev [72, 25] show this can be applied to finite element problems on $H(\text{curl})$ and $H(\text{div})$ spaces (using Nédélec and Raivart–Thomas elements, respectively). These methods are typically independent of the polynomial degree, mesh size and, for DG methods, the penalty parameter. A comparison by Sundar, Stadler and Biros [85] concludes that this approach is more advantageous than h- or p-multigrid.

We particularly highlight the work of Pazner, Dohrmann and Kolev [72], who present numerical experiments using the finite element library MFEM, with which the construction of such preconditioners requires only one or two lines of code. These tests corroborate the theoretical properties of the proposed preconditioners, and demonstrate the flexibility and scalability of the method on a range of challenging three-dimensional problems. These new solvers are flexible and easy to use; any black-box preconditioner for low-order problems can be used to create an effective and efficient preconditioner for the corresponding high-order problem. The `lor_solvers` miniapp, and its parallel counterpart `plor_solvers`, illustrate the construction of low-order-refined (LOR) discretizations and solvers, and come distributed within the MFEM source code, available at `https://github.com/mfem/mfem`.

An alternative approach is to use the observation of Pavarino[68] that additive Schwartz, together with an additive coarse space of order one and a vertex-centred space decomposition, is robust with respect to both p and h for symmetric and coercive problems. The local solves for such problems are dense and solved with a direct method. Furthermore, the coarse-grid operator is also fairly dense and can quickly become expensive. However, in recent years good methods for solving this system have become available; see, e.g., Lottes and Fischer[52].

Recently Brubeck and Farrell[21] introduced a p-robust preconditioner which uses the additive Schwarz method with vertex patches combined with a low-order coarse space as a solver for symmetric and coercive problems. By constructing a tensor product basis that diagonalizes the blocks in the stiffness matrix for the internal degrees of freedom of each individual cell, they show that the patch problem is as sparse as a low-order finite difference discretization and having a sparse Cholesky factorization allows them to scale to large polynomial degree p and afford the assembly and factorization of the matrices in the vertex-patch problems. They successfully apply their method to the Poisson equation and the mixed formulation of linear elasticity both with constant coefficient problems and claim that the theory of [11] suggests that it would remain effective for spatially varying coefficients. In their conclusion, they state that the downsides of their approach: (1) its narrow applicability: it will not be effective on more general problems (tested on Poisson and mixed formulation of linear elasticity), especially for those where the dominant terms include mixed derivatives and mixed vector components; (2) their method relies on having a good quality mesh, with the performance depending on the minimal angle.

Finally, we highlight that even explicit time integration methods require the solution of a mass matrix at each time step. While this is trivial for low-order discretization, it is less obviously the case for high-order problems, as the condition number of the mass matrix grows algebraically with the polynomial order. Ainsworth and Jiang[2, 3] describe an efficient preconditioner for the mass matrix, independent of h and p, based on a specific choice of hierarchical basis, which involves only diagonal solves.

3.2 Preconditioners for low order problems

The methods described above depend on a robust and efficient linear solver for low-order finite element systems. Outside of basic PDEs on uniform grids, where fast multipole methods and fast Fourier transforms may be applied successfully[36], the main choice is between multigrid methods and domain decomposition methods.

3.2.1 Multigrid methods

Multigrid (MG) methods[86] are split into algebraic (AMG) and geometric (GMG) variants. AMG methods use the algebraic properties of the assembled matrix to restrict to coarser grids, whereas GMG solvers use information about the underlying mesh connectivity. GMG solvers can be over an order of magnitude faster than the best AMG solvers [36], since they can be used matrix-free, and operators can be modified on different levels, which can be advantageous when solving, e.g., advection–diffusion problems[73], or when needing to accommodate non-standard boundary conditions. However, there are excellent robust AMG implementations available which work well off-the-shelf for a wide range of problems, which we describe below, and AMG solvers and preconditioners are some of the fastest black-box numerical methods to solve linear systems.

The convergence of AMG solvers is well established for symmetric positive definite (SPD) linear systems resulting from the discretization of general elliptic PDEs or the spatial discretization of parabolic PDEs. Hyperbolic PDEs remain a challenge for AMG, as well as other fast linear solvers, in part because the resulting linear systems are often highly nonsymmetric. Nevertheless, modern AMG implementations can perform well on a wide range of problems.

The HYPRE library[28], originating from the Lawrence Livermore National Laboratories, contains a number of high quality implementations of multigrid preconditioners. BoomerAMG [39] is a parallel implementation of the classical Ruge and Stüben AMG method, and offers a range of coarsening, interpolation, and smoothing options.

The ML preconditioner within Trilinos [35], developed by Sandia National Laboratory, is another fully featured algebraic multigrid implementation, offering a range of multilevel schemes, smoothers, and coarse solvers.

The GAMG preconditioner within PETSc [12, Section 4.4.5] uses a smoothed aggregation coarsening strategy, and offers a reference implementation of the classical Ruge and Stüben method. It also provides unsmoothed aggregation, which can be useful for nonsymmetric problems. Note that both HYPRE and ML are also available via the PETSc interface.

The AGMG method of Notay [61, 60] is another aggregation-based algebraic multigrid method which can be applied to any system matrix that has positive diagonal entries. The implementation can use MPI, multithreading, or both.

A variety of multigrid relaxation methods, based on a topological construction, are incorporated into a unifying software abstraction called PCPATCH [31], implemented in PETSc. This allows a range of schemes to be available through simple manipulation of solver options at runtime and so enables ready exploration of suitable multigrid methods for challenging problems. Various parameter-robust preconditioners for physical applications are discussed which fit within this framework and a Firedrake example code is provided.

It is well known that AMG methods do not perform well 'out-of-the-box' for anisotropic problems, and may require some problem-specific tuning; the PETSc documentation [12, Section 4.4.5] gives some advice for this.

Manteuffel et al. [53] present a new variation on classical AMG for nonsymmetric matrices (denoted ℓ-AIR), based on a local approximation to the ideal restriction operator, coupled with F-relaxation. They demonstrate the efficacy of the proposed preconditioner on systems arising from the discrete form of the advection–diffusion–reaction equation. The ℓ-AIR approach is shown to be a robust solver for various discretizations of the advection–diffusion–reaction equation, including time-dependent and steady-state, from purely advective to purely diffusive. Convergence is robust for discretizations on unstructured meshes and using higher-order finite elements, and is particularly effective on upwind discontinuous Galerkin discretizations. We note that ℓ-AIR is available in PyAMG [64], and an implementation through HYPRE is underway.

In a related work, Manteuffel et al.[54] present a reduction-based AMG method developed for upwind discretizations of hyperbolic PDEs, based on the concept of a Neumann approximation to ideal restriction (n-AIR). The n-AIR approach can be seen as a variation of local AIR (ℓ-AIR) specifically targeting matrices with triangular structure. Although less versatile than ℓ-AIR, setup times for n-AIR can be substantially faster for problems with high connectivity. The n-AIR algorithm is shown to be an effective and scalable solver of steady state transport for discontinuous, upwind discretizations, with unstructured meshes, and up to 6th-order finite elements, offering a significant improvement over existing AMG methods. It is also shown to be effective on several classes of nearly triangular matrices resulting from curvilinear finite elements and artificial diffusion.

In a very recent preprint, Wimmer et al. [89] propose a solver for anisotropic heat flux tailored towards applications in magnetic confinement fusion plasmas where there is strong anisotropy. This is a combination of a new finite element discretization of the problem, using a discontinuous Galerkin approach on a mixed formulation which introduces heat flux, and an algebraic multigrid method based on the AIR paradigm. They show fast convergence even in the highly anisotropic case where diffusion-based AMG methods typically struggle.

3.2.2 Domain decomposition methods

Domain decomposition methods are among the most efficient for solving sparse linear systems of equations on massively parallel architectures; see, e.g., the book by Dolean, Jolivet and Nataf [26] for a recent overview of the field. Their effectiveness relies on a judiciously chosen coarse space. Spillane et al. introduced GenEO, a spectral coarse space, in [84]. GenEO proved to be very attractive as it can be constructed efficiently and adapts to the difficulties posed in the underlying problem with a minimum amount of effort from the user perspective. This method is theoretically proved to be efficient and robust with respect to the problem size, mesh, discretization type, and order for elliptic PDEs [4, 13], along with any problem parameters. That is, the preconditioner can be constructed efficiently such that the condition number of the preconditioned matrix is bounded from above by a user defined number.

Further, a theory has recently been developed to analyse a GenEO-type coarse space for indefinite and non-self-adjoint problems [18], in particular considering the convection–diffusion–reaction equation. A related approach is also shown to be effective for heterogeneous Helmholtz problems at moderately high frequencies [17, 19].

Al Daas et al. [6] have recently extended GenEO to be applicable as a black-box method to precondition general sparse linear system. They assessed the efficiency and scalability of the algebraic GenEO method using a variety of very challenging problems arising from a wide range of applications and including highly nonsymmetric problems such as the advection dominated advection–diffusion equation. Comparisons against state-of-the-art multigrid preconditioners illustrated the robustness and efficiency of algebraic GenEO.

Al Daas et al. [5] recently introduced an algebraic extension of GenEO for the diagonally weighted normal equation matrix. Their motivation was to precondition iterative methods for solving linear least-squares problems. The weighted normal equation matrix also arises in block preconditioning where a preconditioner is required for the (approximate) Schur complement matrix. The preconditioner proposed in [5] can be directly employed within a block preconditioner resulting in a robust, efficient and scalable preconditioner for the (approximate) Schur complement matrix.

Implementations of the algebraic GenEO methods proposed in [6, 5] are available through the PETSc preconditioner PCHPDDM, and only require the coefficient matrix.

Summary

- The most promising method for solving matrices stemming from high-order elliptic PDEs is to precondition with a low-order finite element operator; low-order solvers have become robust enough to solve the resulting (challenging) linear systems [72].
- Multigrid and domain decomposition based preconditioners are the leading contenders for performant parallel iterative solves on anisotropic elliptic problems of low order.
- There are several excellent algebraic multigrid packages available that are highly parallel. However, to get good performance it is vital to tune the parameters for a given problem. A well-implemented geometric multigrid method would give superior performance to an algebraic multigrid.
- The GenEO method [84] is the domain decomposition method with most promise, with a good trade-off between performance and ease of setup; the recent development of fully algebraic variants [6, 5] make this even more the case.

References

[1] XBraid: Parallel multigrid in time. `http://llnl.gov/casc/xbraid`.

[2] M. Ainsworth and S. Jiang. Preconditioning the mass matrix for high order finite element approximation on tetrahedra. *SIAM Journal on Scientific Computing*, 43(1):A384–A414, 2021.

[3] M. Ainsworth and S. Jiang. A systematic approach to constructing preconditioners for the *hp*-version mass matrix on unstructured and hybrid finite element meshes. *SIAM Journal on Scientific Computing*, 44(2):A901–A934, 2022.

[4] H. Al Daas, L. Grigori, P. Jolivet, and P.-H. Tournier. A multilevel Schwarz preconditioner based on a hierarchy of robust coarse spaces. *SIAM Journal on Scientific Computing*, 43(3):A1907–A1928, 2021.

[5] H. Al Daas, P. Jolivet, and J. A. Scott. A robust algebraic domain decomposition preconditioner for sparse normal equations. *SIAM Journal on Scientific Computing*, 44(3):A1047–A1068, 2022.

[6] H. Al Daas, P. Jolivet, and T. Rees. Efficient algebraic two-level Schwarz preconditioner for sparse matrices. Accepted in *SIAM Journal on Scientific Computing*, 2023.

[7] V. Alexandrov, A. Lebedev, E. Sahin, and S. Thorne. Linear systems of equations and preconditioners relating to the NEPTUNE Programme. NEPTUNE Technical Report 2047353-TN-04, CCFE Culham, 2021.

[8] R. Anderson, J. Andrej, A. Barker, J. Bramwell, J.-S. Camier, J. C. V. Dobrev, Y. Dudouit, A. Fisher, T. Kolev, W. Pazner, M. Stowell, V. Tomov, I. Akkerman, J. Dahm, D. Medina, and S. Zampini. MFEM: A modular finite element methods library. *Computers & Mathematics with Applications*, 81:42–74, 2021.

[9] U. M. Ascher and L. R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM, 1998.

[10] U. M. Ascher, S. J. Ruuth, and R. J. Spiteri. Implicit-Explicit Runge-Kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics*, 25(2):151–167, 1997.

[11] O. Axelsson and J. Karátson. Equivalent operator preconditioning for elliptic problems. *Numerical Algorithms*, 50(3):297–380, 2009.

[12] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. Gropp, D. Karpeyev, D. Kaushik, M. Knepley, D. May, L. Curfman McInnes, R. Mills, T. Munson, K. Rupp, P. Sanan, B. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc Users Manual. 2019.

[13] P. Bastian, R. Scheichl, L. Seelinger, and A. Strehlow. Multilevel spectral domain decomposition. *SIAM Journal on Scientific Computing*, pages S1–S26, 2022. doi: 10.1137/21M1427231. Published online in advance.

[14] P. D. Bello-Maldonado and P. F. Fischer. Scalable low-order finite element preconditioners for high-order spectral element Poisson solvers. *SIAM Journal on Scientific Computing*, 41(5):S2–S18, 2019.

[15] L. A. Berry, W. Elwasif, J. M. Reynolds-Barredo, D. Samaddar, R. Sanchez, and D. E. Newman. Event-based parareal: A data-flow based implementation of parareal. *Journal of Computational Physics*, 231(17):5945–5954, 2012.

[16] J. Betteridge, T. H. Gibson, I. G. Graham, and E. H. Müller. Multigrid preconditioners for the hybridised discontinuous Galerkin discretisation of the shallow water equations. *Journal of Computational Physics*, 426:109948, 2021.

[17] N. Bootland and V. Dolean. Can DtN and GenEO coarse spaces be sufficiently robust for heterogeneous Helmholtz problems? *Mathematical and Computational Applications*, 27(3), 2022.

[18] N. Bootland, V. Dolean, I. G. Graham, C. Ma, and R. Scheichl. Overlapping Schwarz methods with GenEO coarse spaces for indefinite and nonself-adjoint problems. *IMA Journal of Numerical Analysis*, 2022. doi: 10.1093/imanum/drac036. Published online in advance, drac036.

[19] N. Bootland, V. Dolean, I. G. Graham, C. Ma, and R. Scheichl. GenEO coarse spaces for heterogeneous indefinite elliptic problems. In S. C. Brenner, E. Chung, A. Klawonn, F. Kwok, J. Xu, and J. Zou, editors, *Domain Decomposition Methods in Science and Engineering XXVI*, pages 115–122, Cham, Switzerland, 2023. Springer.

[20] S. Boscarino, L. Pareschi, and G. Russo. Implicit-explicit Runge–Kutta schemes for hyperbolic systems and kinetic equations in the diffusion limit. *SIAM Journal on Scientific Computing*, 35(1): A22–A51, 2013.

[21] P. D. Brubeck and P. E. Farrell. A scalable and robust vertex-star relaxation for high-order FEM. *SIAM Journal on Scientific Computing*, 44(5):A2991–A3017, 2022.

[22] C. Canuto, P. Gervasio, and A. Quarteroni. Finite-element preconditioning of G-NI spectral methods. *SIAM Journal on Scientific Computing*, 31(6):4422–4451, 2010.

[23] N. Chalmers and T. Warburton. Low-order preconditioning of high-order triangular finite elements. *SIAM Journal on Scientific Computing*, 40(6):A4040–A4059, 2018.

[24] F. Danieli, B. S. Southworth, and A. J. Wathen. Space-time block preconditioning for incompressible flow. *SIAM Journal on Scientific Computing*, 44(1):A337–A363, 2022.

[25] C. R. Dohrmann. Spectral equivalence of low-order discretizations for high-order H(curl) and H(div) spaces. *SIAM Journal on Scientific Computing*, 43(6):A3992–A4014, 2021.

[26] V. Dolean, P. Jolivet, and F. Nataf. *An Introduction to Domain Decomposition Methods: Algorithms, Theory, and Parallel Implementation.* SIAM, 2015.

[27] W. R. Elwasif, S. S. Foley, D. E. Bernholdt, L. A. Berry, D. Samaddar, D. E. Newman, and R. Sanchez. A dependency-driven formulation of parareal: Parallel-in-time solution of PDEs as a many-task application. In *Proceedings of the 2011 ACM International Workshop on Many Task Computing on Grids and Supercomputers*, pages 15–24. Association for Computing Machinery, New York, NY, USA, 2011.

[28] R. D. Falgout and U. M. Yang. *hypre*: A library of high performance preconditioners. In P. M. A. Sloot, A. G. Hoekstra, C. J. K. Tan, and J. J. Dongarra, editors, *Computational Science — ICCS 2002*, pages 632–641, Berlin, Heidelberg, 2002. Springer.

[29] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, and J. B. Schroder. Parallel time integration with multigrid. *SIAM Journal on Scientific Computing*, 36(6):C635–C661, 2014.

[30] P. E. Farrell, R. C. Kirby, and J. Marchena-Menéndez. Irksome: Automating Runge–Kutta time-stepping for finite element methods. *ACM Transactions on Mathematical Software.*, 47(4), 2021.

[31] P. E. Farrell, M. G. Knepley, L. Mitchell, and F. Wechsung. PCPATCH: Software for the topological construction of multigrid relaxation methods. *ACM Transactions on Mathematical Software*, 47(3), 2021.

[32] M. J. Gander. 50 years of time parallel time integration. In T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, editors, *Multiple Shooting and Time Domain Decomposition Methods*, pages 69–113, Cham, 2015. Springer International Publishing.

[33] M. J. Gander and S. Vandewalle. Analysis of the parareal time-parallel time-integration method. *SIAM Journal on Scientific Computing*, 29(2):556–578, 2007.

[34] M. J. Gander, J. Liu, S.-L. Wu, X. Yue, and T. Zhou. ParaDiag: Parallel-in-time algorithms based on the diagonalization technique. *arXiv:2005.09158*, 2021.

[35] M. Gee, C. Siefert, J. Hu, R. Tuminaro, and M. Sala. Ml 5.0 smoothed aggregation user's guide. Technical Report SAND2006-2649, Sandia National Laboratories, 2006.

[36] A. Gholami, D. Malhotra, H. Sundar, and G. Biros. FFT, FMM, or multigrid? A comparative study of state-of-the-art Poisson solvers for uniform and nonuniform grids in the unit cube. *SIAM Journal on Scientific Computing*, 38(3):C280–C306, 2016.

[37] F. X. Giraldo, J. F. Kelly, and E. M. Constantinescu. Implicit-explicit formulations of a three-dimensional nonhydrostatic unified model of the atmosphere (NUMA). *SIAM Journal on Scientific Computing*, 35(5):B1162–B1194, 2013.

[38] S. Gottlieb, Z. J. Grant, J. Hu, and R. Shu. High order strong stability preserving multiderivative implicit and IMEX Runge–Kutta methods with asymptotic preserving properties. *SIAM Journal on Numerical Analysis*, 60(1):423–449, 2022.

[39] V. E. Henson and U. M. Yang. BoomerAMG: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41(1):155–177, 2002.

[40] J. J. Heys, T. A. Manteuffel, S. F. McCormick, and L. N. Olson. Algebraic multigrid for higher-order finite elements. *Journal of Computational Physics*, 204(2):520–532, 2005.

[41] M. Z. Hossain, C. D. Cantwell, and S. J. Sherwin. A spectral/*hp* element method for thermal convection. *International Journal for Numerical Methods in Fluids*, 93(7):2380–2395, 2021.

[42] X. Jiao, X. Wang, and Q. Chen. Optimal and low-memory near-optimal preconditioning of fully implicit Runge–Kutta schemes for parabolic PDEs. *SIAM Journal on Scientific Computing*, 43(5): A3527–A3551, 2021.

[43] G. E. Karniadakis, M. Israeli, and S. A. Orszag. High-order splitting methods for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 97(2):414–443, 1991.

[44] C. A. Kennedy and M. H. Carpenter. Additive Runge–Kutta schemes for convection–diffusion–reaction equations. *Applied Numerical Mathematics*, 44(1):139–181, 2003.

[45] C. A. Kennedy and M. H. Carpenter. Diagonally Implicit Runge-Kutta Methods for Ordinary Differential Equations. A Review. Technical Report NF1676L-19716, 2016.

[46] C. A. Kennedy and M. H. Carpenter. Diagonally implicit Runge–Kutta methods for stiff ODEs. *Applied Numerical Mathematics*, 146:221–244, 2019.

[47] D. E. Keyes, L. C. McInnes, C. Woodward, W. Gropp, E. Myra, M. Pernice, J. Bell, J. Brown, A. Clo, J. Connors, E. Constantinescu, D. Estep, K. Evans, C. Farhat, A. Hakim, G. Hammond, G. Hansen, J. Hill, T. Isaac, X. Jiao, K. Jordan, D. Kaushik, E. Kaxiras, A. Koniges, K. Lee, A. Lott, Q. Lu, J. Magerlein, R. Maxwell, M. McCourt, M. Mehl, R. Pawlowski, A. P. Randles, D. Reynolds, B. Rivière, U. Rüde, T. Scheibe, J. Shadid, B. Sheehan, M. Shephard, A. Siegel, B. Smith, X. Tang, C. Wilson, and B. Wohlmuth. Multiphysics simulations: Challenges and opportunities. *The International Journal of High Performance Computing Applications*, 27(1):4–83, 2013.

[48] D. A. Knoll and D. E. Keyes. Jacobian-free Newton–Krylov methods: A survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

[49] H.-O. Kreiss and G. Scherer. Method of lines for hyperbolic differential equations. *SIAM Journal on Numerical Analysis*, 29(3):640–646, 1992.

[50] A. Kværnø. Singly diagonally implicit Runge–Kutta methods with an explicit first stage. *BIT Numerical Mathematics*, 44(3):489–502, 2004.

[51] J.-L. Lions, Y. Maday, and G. Turinici. A "parareal" in time discretization of PDE's. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 332:661–668, 2001.

[52] J. W. Lottes and P. F. Fischer. Hybrid multigrid/Schwarz algorithms for the spectral element method. *Journal of Scientific Computing*, 24(1):45–78, 2005.

[53] T. A. Manteuffel, J. Ruge, and B. S. Southworth. Nonsymmetric algebraic multigrid based on local approximate ideal restriction (ℓAIR). *SIAM Journal on Scientific Computing*, 40(6):A4105–A4130, 2018.

[54] T. A. Manteuffel, S. Münzenmaier, J. Ruge, and B. S. Southworth. Nonsymmetric reduction-based algebraic multigrid. *SIAM Journal on Scientific Computing*, 41(5):S242–S268, 2019.

[55] K.-A. Mardal, T. K. Nilssen, and G. A. Staff. Order-optimal preconditioners for implicit Runge–Kutta schemes applied to parabolic PDEs. *SIAM Journal on Scientific Computing*, 29(1):361–375, 2007.

[56] M. Masud Rana, V. E. Howle, K. Long, A. Meek, and W. Milestone. A new block preconditioner for implicit Runge–Kutta methods for parabolic PDE problems. *SIAM Journal on Scientific Computing*, 43(5):S475–S495, 2021.

[57] C. Maynard, T. Melvin, and E. H. Müller. Multigrid preconditioners for the mixed finite element dynamical core of the LFRic atmospheric model. *Quarterly Journal of the Royal Meteorological Society*, 146(733):3917–3936, 2020.

[58] E. McDonald, J. Pestana, and A. Wathen. Preconditioning and iterative solution of all-at-once systems for evolutionary partial differential equations. *SIAM Journal on Scientific Computing*, 40 (2):A1012–A1033, 2018.

[59] S. T. Miller, E. C. Cyr, J. N. Shadid, R. M. J. Kramer, E. G. Phillips, S. Conde, and R. P. Pawlowski. IMEX and exact sequence discretization of the multi-fluid plasma model. *Journal of Computational Physics*, 397:108806, 2019.

[60] Y. Notay. An aggregation-based algebraic multigrid method. *Electronic Transactions on Numerical Analysis*, 37(6):123–146, 2010.

[61] Y. Notay. Aggregation-based algebraic multigrid for convection-diffusion equations. *SIAM Journal on Scientific Computing*, 34(4):A2288–A2316, 2012.

[62] G. Noventa, F. Massa, F. Bassi, A. Colombo, N. Franchina, and A. Ghidoni. A high-order discontinuous Galerkin solver for unsteady incompressible turbulent flows. *Computers & Fluids*, 139:248–260, 2016.

[63] L. Olson. Algebraic multigrid preconditioning of high-order spectral elements for elliptic problems on a simplicial mesh. *SIAM Journal on Scientific Computing*, 29(5):2189–2209, 2007.

[64] L. N. Olson and J. B. Schroder. PyAMG: Algebraic multigrid solvers in Python v4.0, 2018. URL `https://github.com/pyamg/pyamg`. Release 4.0.

[65] S. A. Orszag. Spectral methods for problems in complex geometries. *Journal of Computational Physics*, 37(1):70–92, 1980.

[66] Y. Pan, Z.-G. Yan, J. Peiró, and S. J. Sherwin. Development of a balanced adaptive time-stepping strategy based on an implicit JFNK-DG compressible flow solver. *Communications on Applied Mathematics and Computation*, 2021.

[67] L. Pareschi and G. Russo. Implicit–explicit Runge–Kutta schemes and applications to hyperbolic systems with relaxation. *Journal of Scientific Computing*, 25(1):129–155, 2005.

[68] L. F. Pavarino. Additive Schwarz methods for the p-version finite element method. *Numerische Mathematik*, 66(1):493–515, 1993.

[69] W. Pazner. Efficient low-order refined preconditioners for high-order matrix-free continuous and discontinuous Galerkin methods. *SIAM Journal on Scientific Computing*, 42(5):A3055–A3083, 2020.

[70] W. Pazner and T. Kolev. Uniform subspace correction preconditioners for discontinuous Galerkin methods with hp-refinement. *Communications on Applied Mathematics and Computation*, 4(2): 697–727, 2022.

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

[72] W. Pazner, T. Kolev, and C. Dohrmann. Low-order preconditioning for the high-order finite element de Rham complex. *SIAM Journal on Scientific Computing*, 45(2):A675–A702, 2023.

[73] A. Ramage. A multigrid preconditioner for stabilised discretisations of advection–diffusion problems. *Journal of Computational and Applied Mathematics*, 110(1):187–203, 1999.

[74] J. M. Reynolds Barredo, D. E. Newman, R. Sanchez, D. Samaddar, L. A. Berry, and W. Elwasif. Toward the application of the Parareal algorithm to 5D gyrokinetic plasma turbulence. In *53rd Annual Meeting of the APS Division of Plasma Physics*, page TP9.053, 2011.

[75] J. M. Reynolds-Barredo, D. E. Newman, R. Sanchez, D. Samaddar, L. A. Berry, and W. R. Elwasif. Mechanisms for the convergence of time-parallelized, parareal turbulent plasma simulations. *Journal of Computational Physics*, 231(23):7851–7867, 2012.

[76] D. Samaddar, D. E. Newman, and R. Sánchez. Parallelization in time of numerical simulations of fully-developed plasma turbulence using the parareal algorithm. *Journal of Computational Physics*, 229(18):6558–6573, 2010.

[77] D. Samaddar, T. A. Casper, S. H. Kim, L. A. Berry, W. R. Elwasif, D. Batchelor, and W. A. Houlberg. Time parallelization of advanced operation scenario simulations of ITER plasma. *Journal of Physics: Conference Series*, 410:012032, 2013.

[78] D. Samaddar, D. P. Coster, X. Bonnin, C. Bergmeister, E. Havlíčková, L. A. Berry, W. R. Elwasif, and D. B. Batchelor. Temporal parallelization of edge plasma simulations using the parareal algorithm and the SOLPS code. *Computer Physics Communications*, 221:19–27, 2017.

[79] D. Samaddar, D. P. Coster, X. Bonnin, L. A. Berry, W. R. Elwasif, and D. B. Batchelor. Application of the parareal algorithm to simulations of ELMs in ITER plasma. *Computer Physics Communications*, 235:246–257, 2019.

[80] V. Simoncini. Computational methods for linear matrix equations. *SIAM Review*, 58(3):377–441, 2016.

[81] B. S. Southworth, W. Mitchell, A. Hessenthaler, and F. Danieli. Tight two-level convergence of Linear Parareal and MGRIT: Extensions and implications in practice. In B. Ong, J. Schroder, J. Shipton, and S. Friedhoff, editors, *Parallel-in-Time Integration Methods*, pages 1–31, Cham, 2021. Springer International Publishing.

[82] B. S. Southworth, O. Krzysik, and W. Pazner. Fast solution of fully implicit Runge–Kutta and discontinuous Galerkin in time for numerical PDEs, Part II: nonlinearities and DAEs. *SIAM Journal on Scientific Computing*, 44(2):A636–A663, 2022.

[83] B. S. Southworth, O. Krzysik, W. Pazner, and H. D. Sterck. Fast solution of fully implicit Runge–Kutta and discontinuous Galerkin in time for numerical PDEs, Part I: the linear setting. *SIAM Journal on Scientific Computing*, 44(1):A416–A443, 2022.

[84] N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, and R. Scheichl. Abstract robust coarse spaces for systems of PDEs via generalized eigenproblems in the overlaps. *Numerische Mathematik*, 126(4):741–770, 2014.

[85] H. Sundar, G. Stadler, and G. Biros. Comparison of multigrid algorithms for high-order continuous finite element discretizations. *Numerical Linear Algebra with Applications*, 22(4):664–680, 2015.

[86] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, 2000.

[87] P. E. Vos, C. Eskilsson, A. Bolis, S. Chun, R. M. Kirby, and S. J. Sherwin. A generic framework for time-stepping partial differential equations (PDEs): General linear methods, object-oriented implementation and application to fluid problems. *International Journal of Computational Fluid Dynamics*, 25(3):107–125, 2011.

[88] A. J. Wathen. Preconditioning. *Acta Numerica*, 24:329–376, 2015.

[89] G. A. Wimmer, B. S. Southworth, T. J. Gregory, and X. Tang. A fast algebraic multigrid solver and accurate discretization for highly anisotropic heat flux I: open field lines. *arXiv:2301.13351*, 2023.

[90] Z.-G. Yan, Y. Pan, G. Castiglioni, K. Hillewaert, J. Peiró, D. Moxey, and S. J. Sherwin. Nektar++: Design and implementation of an implicit, spectral/*hp* element, compressible flow solver using a Jacobian-free Newton Krylov approach. *Computers & Mathematics with Applications*, 81:351–372, 2021.