Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2021-0182

Vol. **32**, No. 1, pp. 156-188
July 2022

Reliability Investigation of BiCGStab and IDR Solvers for the Advection-Diffusion-Reaction Equation

Chris Schoutrop[1,*], Jan ten Thije Boonkkamp[2] and Jan van Dijk[1]

[1] *Department of Applied Physics, Eindhoven University of Technology, The Netherlands.*
[2] *Department of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands.*

Received 8 September 2021; Accepted (in revised version) 14 April 2022

Abstract. The reliability of BiCGStab and IDR solvers for the exponential scheme discretization of the advection-diffusion-reaction equation is investigated. The resulting discretization matrices have real eigenvalues. We consider BiCGStab, IDR(S), BiCGStab(L) and various modifications of BiCGStab, where S denotes the dimension of the shadow space and L the degree of the polynomial used in the polynomial part. Several implementations of BiCGStab exist which are equivalent in exact arithmetic, however, not in finite precision arithmetic. The modifications of BiCGStab we consider are; choosing a random shadow vector, a reliable updating scheme, and storing the best intermediate solution. It is shown that the Local Minimal Residual algorithm, a method similar to the ''minimize residual" step of BiCGStab, can be interpreted in terms of a time-dependent advection-diffusion-reaction equation with homogeneous Dirichlet boundary conditions for the residual, which plays a key role in the convergence analysis. Due to the real eigenvalues, the benefit of BiCGStab(L) compared to BiCGStab is shown to be modest in numerical experiments. Non-sparse (e.g. uniform random) shadow residual turns out to be essential for the reliability of BiCGStab. The reliable updating scheme ensures the required tolerance is truly achieved. Keeping the best intermediate solution has no significant effect. Recommendation is to modify BiCGStab with a random shadow residual and the reliable updating scheme, especially in the regime of large Péclet and small Damköhler numbers. An alternative option is IDR(S), which outperforms BiCGStab for problems with strong advection in terms of the number of matrix-vector products. The MATLAB code used in the numerical experiments is available on GitLab: `https://gitlab.com/ChrisSchoutrop/krylov-adr`, a C++ implementation of IDR(S) is available in the Eigen linear algebra library: `http://eigen.tuxfamily.org`.

AMS subject classifications: 00A79, 70-08, 15A06, 15B05, 15A18

*Corresponding author. *Email addresses:* `c.e.m.schoutrop@tue.nl` (C. Schoutrop), `j.h.m.tenthijeboonkkamp@tue.nl` (J. ten Thije Boonkkamp), `j.v.dijk@tue.nl` (J. van Dijk)

Key words: BiCGStab, IDR, shadow residual, advection-diffusion-reaction equation.

1 Introduction

Advection-diffusion-reaction (ADR) equations and their discrete approximations are ubiquitous in the modeling of physical systems [1–5]. A wide variety of discretization schemes for the ADR equation, such as finite difference, (pseudo)spectral, finite element and finite volume methods exist. For the solution to be representative, the discretization scheme must not only be convergent in the limit of infinitesimally fine grids, but also yield representative results for more pragmatic grid sizes. A counterexample, is the discretization of the ADR equation in the presence of strong advection where the central differencing scheme yields spurious oscillations if the grid is too coarse [6, p. 83]. This discrepancy is reflected by the eigenvalues of the exact ADR-operator and the discretized operator, i.e., the exact operator has *real* eigenvalues, whereas the discretized version has *complex* eigenvalues. However, for the exponential discretization scheme of [6, 7] which we use here, the discretized version also has real eigenvalues.

After the discretization step a linear system is obtained which must be solved to obtain the approximate solution. Such linear systems are of the type $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$, with $\boldsymbol{A}$ a generally sparse, asymmetric, but invertible matrix of size $N\times N$. In this paper we mainly consider 3D equations. Note that even with a modest $M=102$ grid points per direction this results in a linear system with $N=10^6$ unknowns. A robust method for solving such linear systems is by factorizing $\boldsymbol{A}$ into a pair of lower and upper triangular matrices using the well-known LU decomposition [8, p. 96], and subsequently computing $\boldsymbol{x}$ by solving two triangular systems using backward and forward substitution. The main downside of LU decomposition is that for a sparse system matrix there can be significant fill-in; the factors $\boldsymbol{L}$ and $\boldsymbol{U}$ are not guaranteed to be sparse. As a result the time complexity of factorizing the resulting $\boldsymbol{A}$ for discretized three-dimensional ADR-equations is in general $\mathcal{O}(N^{7/3})$ [9].

Another approach are iterative methods, most commonly the Krylov subspace methods such as the Conjugate Gradient (CG) method. Such methods seek successive approximations to the solution as a projection on the linear subspace $\mathcal{K}_k(\boldsymbol{A},\boldsymbol{r}_0)=\text{span}\{\boldsymbol{r}_0,\boldsymbol{A}\boldsymbol{r}_0,\boldsymbol{A}^2\boldsymbol{r}_0,\cdots,\boldsymbol{A}^{k-1}\boldsymbol{r}_0\}$ with $\boldsymbol{r}_0:=\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}_0$ the residual generated by some initial guess $\boldsymbol{x}_0$. For linear systems obtained from discretizing a second order PDE with a grid spacing proportional to $N^{-1/3}$, the CG method requires $\mathcal{O}(N^{4/3})$ flops to reach a given tolerance ϵ [9], saving a factor N compared to the LU decomposition. A second benefit is that the memory requirement of $\mathcal{O}(N)$ is modest, as only the matrix itself and a constant number of vectors of size N have to be stored. Additionally, CG is an optimal Krylov method in the sense that in each iteration the error is minimized over the $\boldsymbol{A}$-norm [10].

The main drawback of CG is the requirement of a symmetric, positive definite matrix $\boldsymbol{A}$. For more general invertible matrices the Faber-Manteuffel theorem [11] states

that there is no Krylov subspace method using $\mathcal{K}_k(\boldsymbol{A},\boldsymbol{r}_0)$ that has short recurrences and optimality [12]. Short recurrence methods express the next residual and solution approximation in terms of a fixed (practically small) number of previous residuals and solutions. However, in long recurrence methods the number of terms in the recurrence grows with the iteration count. Optimality here refers to the minimization of the error $\boldsymbol{e}_k := \boldsymbol{x}-\boldsymbol{x}_k \in \text{span}\{\boldsymbol{x}-\boldsymbol{x}_0\}+\mathcal{K}_k(\boldsymbol{A},\boldsymbol{r}_0)$ in some norm. Therefore, for a general invertible matrix $\boldsymbol{A}$ one has to either, drop the optimality requirement, use long recurrences, use a space different than $\mathcal{K}_k(\boldsymbol{A},\boldsymbol{r}_0)$, or settle for a compromise. This leads to a wide variety of Krylov methods, each with different tradeoffs.

One possible extension of CG to solve non-symmetric systems is the BiCG algorithm originally described in [13]. The idea of BiCG is to use a second Krylov subspace denoted as the ''shadow space''; $\mathcal{K}_k(\boldsymbol{A}^{\mathrm{T}},\widetilde{\boldsymbol{r}}_0)$, where the vector $\widetilde{\boldsymbol{r}}_0$ is arbitrary but commonly chosen as $\boldsymbol{r}_0$. Unfortunately, the BiCG algorithm has several drawbacks as well. First, unlike CG the residuals produced by the BiCG algorithm are not guaranteed to decrease each iteration in some norm. Second, additional computational effort is needed in matrix vector products involving $\boldsymbol{A}^{\mathrm{T}}$ [8, p. 247]. Third, matrix-vector products of the form $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{v}$ may be prohibitively expensive to compute, making the BiCG method unsuitable for some applications such as Newton-Krylov methods [2, 8, p. 241].

To alleviate these issues the well-known BiCGStab algorithm was developed. This method was originally described in [14] and has since then been incorporated in many popular linear algebra packages such as MATLAB [15] and Eigen [16], and in simulation software such as PLASIMO [17] and COMSOL [18]. BiCGStab combines one iteration of BiCG with a Local Minimal Residual (LMR) step [19]; this step can also be seen as performing one iteration of the GMRES method [20]. LMR chooses the current residual as a search direction, and takes a step in this direction such that the next residual is minimized, which results in a smoother decrease of the residual norm of BiCGStab compared to BiCG. Additionally, BiCGStab does not require operations involving $\boldsymbol{A}^{\mathrm{T}}$. However, this combination of LMR and BiCG leads to several subtleties which are discussed in Section 2.

To compare BiCGStab with other Krylov methods, we also include BiCGStab(L) and IDR(S). The BiCGStab(L) method, combines L steps of BiCG with a GMRES(L) step [21]. Another method we include is the recent IDR(S) algorithm, which can be seen as using S different shadow spaces; $\mathcal{K}_k(\boldsymbol{A},\widetilde{\boldsymbol{R}})$ with $\widetilde{\boldsymbol{R}}$ an $N\times S$ matrix. IDR(S) is constructed such that the residuals are forced into a subspace that decreases in dimension each iteration [22]. Even though not as widespread as BiCGStab, IDR(S) outperforms BiCGStab for a wide class of problems [22, 23]. We focus on BiCGStab, BiCGStab(L), IDR(S), and most importantly the differences in reliability arising in the implementations of the BiCGStab algorithm.

In this paper we focus on specific implementations of the BiCGStab algorithm, and show that small differences in these implementations can have a significant impact on the reliability of this solver. Here we place the effects of these differences in a physical context by considering the advection-diffusion-reaction equation. Furthermore, we do

not use the central differencing discretization scheme, but apply the exponential scheme for which the common critique that BiCGStab cannot handle complex eigenvalues does not apply. We then show that even though the eigenvalues are real, there are still pitfalls for a widely used BiCGStab implementation, and we present possible mitigations. Additionally we show that even for preconditioned systems these modifications are still applicable. To summarize, the objective and novelty of this paper is:

1. Investigate the reliability of BiCGStab, BiCGStab(L) and IDR(S) for advection-diffusion-reaction equations discretized with the exponential scheme, resulting in matrices with real eigenvalues.
2. Discuss the subtleties in different BiCGStab implementations for model linear systems.
3. Discuss pitfalls and remedies for several BiCGStab implementations.
4. Highlight the critical effect of the choice of a shadow residual $\widetilde{\boldsymbol{r}}$ in BiCGStab.
5. Illustrate that for the LMR method, the residual propagates similarly to the solution of a time-dependent advection-diffusion equation. This phenomenon is used to obtain insight into the convergence of BiCGStab.
6. Present numerical results suggesting that the residual for BiCGStab is transported similarly to the solution of a time-dependent advection-diffusion problem to support the observation regarding the shadow residual $\widetilde{\boldsymbol{r}}$.
7. Show that even for preconditioned systems the proposed modifications are still relevant.

Typically, large sparse linear systems are solved using preconditioned iterations. We investigate the effect of Jacobi, incomplete LU and geometric multigrid preconditioners for the MATLAB implementation of BiCGStab. Such preconditioners do indeed significantly reduce the number of required matrix-vector products, however we show that it does not alleviate all shortcomings. To simplify the analysis, and work toward a robust BiCGStab variant that converges for a broad range of systems we mainly consider unpreconditioned systems in the coming sections.

The contents of this paper are the following. First, the BiCGStab algorithm and different implementations are discussed in Section 2. Second, the model ADR-system along with the discretization and resulting eigenvalues are introduced in Section 3. Third, several numerical experiments are presented and discussed in Section 4. Finally, we end with concluding remarks in Section 5.

In conclusion, we recommend to use BiCGStab with a random shadow residual $\widetilde{\boldsymbol{r}}$ and reliable updating scheme, or alternatively the efficient IDR(S) algorithm. The MATLAB code used in the numerical experiments is available on GitLab: `https://gitlab.com/ChrisSchoutrop/krylov-adr` a C++ implementation of IDR(S) is available in the Eigen library [16].

2 Solver implementation

One of the main goals of this paper is to compare BiCGStab and IDR. However, this is not as straightforward as one might expect. For example, the BiCGStab algorithm has several variants, which are not identical for finite precision arithmetic. A common alternative to BiCGStab is BiCGStab(L), which modifies BiCGStab to include a higher order stabilizing polynomial step and reduces to BiCGStab for $L=1$. The higher order polynomial step makes BiCGStab(L) more robust for matrices having eigenvalues with large imaginary parts [21]. It is shown in [24] that BiCGStab(L) can be implemented in different ways, affecting both stability and computational complexity. There also exist other algorithms which reduce to BiCGStab for specific parameters, such as IDR(S) [22] and IDR(S)Stab(L) [23, 25]. In addition there are several variants for specific situations, such as a version of BiCGStab geared specifically for parallel computers; see e.g. [26] and [27], a block-version [28] specialized for solving systems with multiple right-hand sides, and a recycling version [29] optimized for solving a sequence of linear systems.

The BiCGStab algorithm which we list here for the sake of completeness is based on Algorithm 1 of [30] with $L=1$, given here as the baseline BiCGStab Algorithm 1. In this paper $\langle \boldsymbol{a},\boldsymbol{b}\rangle := \boldsymbol{a}^{\mathrm{T}}\boldsymbol{b}$ denotes the inner product of the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. In Algorithm 1 the MATLAB notation is used, i.e., $\widehat{\boldsymbol{r}}_{:,1}$ denotes the entire first column of the matrix $\widehat{\boldsymbol{r}}$. Unless mentioned explicitly all arithmetic is conducted over the real numbers.

As mentioned in the introduction, BiCGStab combines BiCG with LMR. In Algorithm 1, specifically in Lines 2−6, the initial residual $\widehat{\boldsymbol{r}}_{:,1}$, the search directions $\widehat{\boldsymbol{u}}$ and the shadow residual $\widetilde{\boldsymbol{r}}$ are initialized. The next part (lines 8−15) performs one BiCG iteration, and the lines 16−19 perform the LMR-step. The LMR-step can be seen as choosing the search direction equal to the residual $\widehat{\boldsymbol{r}}_{:,1}$, and computing ω such that the norm of the next residual, $\|\widehat{\boldsymbol{r}}_{:,1}-\omega\widehat{\boldsymbol{r}}_{:,2}\|_2$, is minimized. Note that $\widehat{\boldsymbol{r}}_{:,1}$ contains the residual and $\widehat{\boldsymbol{r}}_{:,2}$ is an auxiliary vector used in the LMR step.

To illustrate some of the differences between several BiCGStab implementations, we focus on three specific modifications that require little alteration to the baseline BiCGStab algorithm, i.e.,

1. Random shadow residual, $\widetilde{\boldsymbol{r}}$, Section 2.1.
2. Reliable update scheme from [31], Section 2.2.
3. Storing solution with best residual thus far, Section 2.3.

In the next sections we illustrate shortcomings of baseline BiCGStab using small model problems, and comment on the effect of these three modifications. In Section 2.4 it is shown how the initial residual is propagated similarly to the solution of a time-dependent advection-diffusion equation, which is argued to be the main reason for the effectiveness of choosing a random shadow residual later in this paper.

Algorithm 1 Baseline BiCGStab

1: **function** BASELINE BICGSTAB($\boldsymbol{A},\boldsymbol{b},\boldsymbol{x},tol$)
2: $\rho_0 \leftarrow 1, \alpha \leftarrow 1, \omega \leftarrow 1, \widehat{\boldsymbol{r}} \leftarrow \boldsymbol{0} \in \mathbb{R}^{N\times 2}, \widehat{\boldsymbol{u}} \leftarrow \boldsymbol{0} \in \mathbb{R}^{N\times 2}$
3: $\widehat{\boldsymbol{r}}_{:,1} \leftarrow \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$
4: $\widetilde{\boldsymbol{r}} \leftarrow \widehat{\boldsymbol{r}}_{:,1}$ ▷ Arbitrary, such that $\langle \widetilde{\boldsymbol{r}}, \widehat{\boldsymbol{r}}_{:,1} \rangle \neq 0$
5: $\boldsymbol{x}' \leftarrow \boldsymbol{x}, \quad \boldsymbol{x} \leftarrow \boldsymbol{0}$
6: $\zeta \leftarrow \|\widehat{\boldsymbol{r}}_{:,1}\|$
7: **while** $\zeta > tol\|\boldsymbol{b}\|$ **do**
8: $\rho_0 \leftarrow -\rho_0\omega$
9: $\rho_1 \leftarrow \langle \widetilde{\boldsymbol{r}}, \widehat{\boldsymbol{r}}_{:,1} \rangle$
10: $\beta \leftarrow \alpha(\rho_1/\rho_0), \quad \rho_0 \leftarrow \rho_1$
11: $\widehat{\boldsymbol{u}}_{:,1} \leftarrow \widehat{\boldsymbol{r}}_{:,1} - \beta\widehat{\boldsymbol{u}}_{:,1}$
12: $\widehat{\boldsymbol{u}}_{:,2} \leftarrow \boldsymbol{A}\widehat{\boldsymbol{u}}_{:,1}$
13: $\alpha \leftarrow \rho_1 / \langle \widetilde{\boldsymbol{r}}, \widehat{\boldsymbol{u}}_{:,2} \rangle$
14: $\boldsymbol{x} \leftarrow \boldsymbol{x} + \alpha\widehat{\boldsymbol{u}}_{:,1}$
15: $\widehat{\boldsymbol{r}}_{:,1} \leftarrow \widehat{\boldsymbol{r}}_{:,1} - \alpha\widehat{\boldsymbol{u}}_{:,2}$
16: $\widehat{\boldsymbol{r}}_{:,2} \leftarrow \boldsymbol{A}\widehat{\boldsymbol{r}}_{:,1}$
17: $\omega \leftarrow \text{argmin}_\omega \|\widehat{\boldsymbol{r}}_{:,1} - \omega\widehat{\boldsymbol{r}}_{:,2}\| = \langle \widehat{\boldsymbol{r}}_{:,2}, \widehat{\boldsymbol{r}}_{:,1} \rangle / \langle \widehat{\boldsymbol{r}}_{:,2}, \widehat{\boldsymbol{r}}_{:,2} \rangle$
18: $\boldsymbol{x} \leftarrow \boldsymbol{x} + \omega\widehat{\boldsymbol{r}}_{:,1}$
19: $\widehat{\boldsymbol{r}}_{:,1} \leftarrow \widehat{\boldsymbol{r}}_{:,1} - \omega\widehat{\boldsymbol{r}}_{:,2}$
20: $\widehat{\boldsymbol{u}}_{:,1} \leftarrow \widehat{\boldsymbol{u}}_{:,1} - \omega\widehat{\boldsymbol{u}}_{:,2}$
21: $\zeta \leftarrow \|\widehat{\boldsymbol{r}}_{:,1}\|$
22: **end while**
23: $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{x}'$ **return** $\boldsymbol{x}, \zeta/\|\boldsymbol{b}\|$
24: **end function**

2.1 Choice of shadow residual

There is quite some freedom in the implementation of BiCGStab, one example is the choice for $\widetilde{\boldsymbol{r}}$ in Algorithm 1. This vector of the BiCGStab algorithm is the so-called ''shadow residual'' and can be chosen almost arbitrarily. By default $\widetilde{\boldsymbol{r}}$ is chosen as the first residual $\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$. However, as will be demonstrated for several model systems, the specific choice of $\widetilde{\boldsymbol{r}}$ can strongly influence the convergence of BiCGStab.

An interesting alternative choice for $\widetilde{\boldsymbol{r}}$ stems from the IDR(S) algorithm in [22], i.e., choosing $\widetilde{\boldsymbol{r}}$ random and *complex*. To illustrate this, consider linear systems involving a matrix with eigenvalues that have a large imaginary part compared to the real part. For such systems BiCGStab breaks down as a result of $|\omega| \ll 1$ in line 17. Such systems were one of the main motivations to develop BiCGStab(L) [21]. The simplest system for which BiCGStab will stagnate due to $\omega = 0$ is a system involving a $\pi/2$-rotation matrix;

$$\boldsymbol{A} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \boldsymbol{x}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{2.1}$$

The matrix in (2.1) has purely imaginary eigenvalues $\pm i$. It is straightforward to show that for any real-valued $\widehat{\boldsymbol{r}}_{:,1}$, the operation $\widehat{\boldsymbol{r}}_{:,2} \leftarrow \boldsymbol{A}\widehat{\boldsymbol{r}}_{:,1}$ results in $\langle \widehat{\boldsymbol{r}}_{:,1}, \widehat{\boldsymbol{r}}_{:,2} \rangle = 0$, since $\widehat{\boldsymbol{r}}_{:,1} \perp \widehat{\boldsymbol{r}}_{:,2}$ and thus $\omega = 0$. One way to avoid this is to use a version of BiCGStab with higher-order stabilizing polynomials, as is the idea behind BiCGStab(L) with $L > 1$. Alternatively, note that this issue does not arise in complex arithmetic. This is achieved by choosing $\widetilde{\boldsymbol{r}}$ random and complex, since this will also result in a complex-valued $\widehat{\boldsymbol{r}}_{:,1}$. Consequently, as the inner product to determine ω also involves complex conjugation, it can be shown that $\langle \widehat{\boldsymbol{r}}_{:,1}, \widehat{\boldsymbol{r}}_{:,2} \rangle$ is purely imaginary. In this case BiCGStab does most likely not break down, however, choosing complex $\widetilde{\boldsymbol{r}}$ comes at the cost of replacing real by complex arithmetic.

But, there also exist systems with *real* eigenvalues for which the default choice of $\widetilde{\boldsymbol{r}}$ will cause BiCGStab to break down. Surprisingly, this occurs for the following system

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \boldsymbol{x}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{2.2}$$

It can be verified that the default choice of $\widetilde{\boldsymbol{r}}$ leads to $\langle \widetilde{\boldsymbol{r}}, \widehat{\boldsymbol{u}}_{:,2} \rangle = 0$ and the algorithm stalls while not having updated $\boldsymbol{x}$ even once. The idea of choosing a random $\widetilde{\boldsymbol{r}}$ is to avoid any correlation between the generated residuals $\widehat{\boldsymbol{r}}$ and search directions $\widehat{\boldsymbol{u}}$, making it exceedingly unlikely for BiCGStab to break down or stagnate due to any of these quantities having a vanishingly small inner product with $\widetilde{\boldsymbol{r}}$. This can be achieved by choosing every element of $\widetilde{\boldsymbol{r}}$ in the interval $(0,1)$ from a uniform random distribution, i.e., $\widetilde{r}_i \in U(0,1)$.

Inspired by the system presented in Example 3.3 of [32], another system for which BiCGStab will break down, and which can be avoided by a different choice of $\widetilde{\boldsymbol{r}}$ reads

$$\boldsymbol{A} = \begin{bmatrix} \lambda & 0 & 0 \\ -\lambda & \lambda & 0 \\ 0 & -\lambda & \lambda \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{x}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \lambda > 0, \tag{2.3}$$

where again all eigenvalues are real, and are equal to λ. This system can be seen as the discretization with the upwind scheme of a one-dimensional problem on a grid with only a few grid cells. Note that for problems involving strong advection the exponential scheme reduces to the upwind discretization scheme. Note that each iteration the boundary value on the left is propagated one grid cell. For the default choice of $\widetilde{\boldsymbol{r}} = \boldsymbol{r}_0$ this is fatal, since $\boldsymbol{r}_0$ only has the first element non-zero, however, the later residuals produced have a vanishing first element, leading to $\rho_1 = \langle \widetilde{\boldsymbol{r}}, \widehat{\boldsymbol{r}}_{:,1} \rangle = 0$ on line 9 in Algorithm 1. Because of this, combined with the property that $\boldsymbol{A}\boldsymbol{e}_i = \lambda(\boldsymbol{e}_i - \boldsymbol{e}_{i+1})$ for $i < 3$, α on line 13 becomes undefined due to a zero by zero division. A straightforward calculation shows that α becomes undefined in the second iteration of Algorithm 1. It is shown in Section 2.4 that $\langle \boldsymbol{r}_0, \boldsymbol{r}_k \rangle$ decreases exponentially for the LMR method; it is also observed later in Section 4.4 that a similar phenomenon occurs for BiCGStab and LMR when applied to discretization matrices obtained from two-dimensional ADR problems.

The idea of choosing a random $\widetilde{\boldsymbol{r}}$ is inspired by several sources. First, the effect of choosing a random initial guess (and thus a random $\widetilde{\boldsymbol{r}}$) was shown in [33] for BiCG algorithms to significantly impact which model problems could be solved. Second, similar

to BiCGStab, IDR(S) uses an S-dimensional shadow space. In [22] in the context of the IDR(S) algorithm it was noted that choosing *all* S vectors at random is essential for robustness. Note that IDR(1) is equivalent to BiCGStab. Finally, it was shown in two numerical experiments of acoustics problems that choosing a random $\widetilde{\boldsymbol{r}}$ improved convergence [34] for CGS, and in [35] for both CGS and BiCGStab. Here CGS is the well-known Conjugate Gradient Squared method, a derivative of BiCG that does not require operations with $\boldsymbol{A}^T$.

2.2 Reliable update scheme

Note that in baseline BiCGStab, the residual $\boldsymbol{r} := \widehat{\boldsymbol{r}}_{:,1}$ is only directly computed from the definition $\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$ at the initialization step in line 3. For all subsequent iterations both $\boldsymbol{r}$ and $\boldsymbol{x}$ are computed recursively from the results of previous iterations. This opens an avenue through which finite precision arithmetic can lead to discrepancies between the true residual $\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$ and the recursively computed residual $\boldsymbol{r}$. This discrepancy between the true residual and the recursively computed residual is called the *residual gap*.

Assuming the only error is introduced by k updates of the form

$$\boldsymbol{x}_{j+1} = \boldsymbol{x}_j + \boldsymbol{w}_{j+1}, \quad \boldsymbol{r}_{j+1} = \boldsymbol{r}_j - \boldsymbol{A}\boldsymbol{w}_{j+1}, \tag{2.4}$$

it is shown in [24] that the maximum residual gap due to accumulating rounding errors is bounded by

$$\big| \|\boldsymbol{r}_k\| - \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k\| \big| \leq 2k n_{\mathrm{A}} \epsilon_{\mathrm{m}} \| |\boldsymbol{A}| \| \|\boldsymbol{A}^{-1}\| \max_{j \leq k} \|\boldsymbol{r}_j\|. \tag{2.5}$$

Here n_A is the maximum number of non-zeros per row and ϵ_{m} the machine precision, $\| |\boldsymbol{A}| \|$ is the 2-norm of the element-wise absolute value of $\boldsymbol{A}$. An example of a problematic system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is given by

$$\boldsymbol{A} = \begin{bmatrix} 1 & -\gamma & 0 \\ -\gamma & 1 & -\gamma \\ 0 & -\gamma & 1 \end{bmatrix}, \quad \boldsymbol{x}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}. \tag{2.6}$$

For this system it can be shown, in particular if $\gamma \geq \sqrt{2}$, that

$$\| |\boldsymbol{A}| \| = 1 + \sqrt{2}\gamma, \quad \|\boldsymbol{A}^{-1}\| = 1, \tag{2.7}$$

where the matrix-norms are the 2-norm. After the first update to the residual we obtain

$$\widehat{\boldsymbol{r}}_{:,1} = \begin{bmatrix} 0 \\ 2\gamma \\ 0 \end{bmatrix}, \tag{2.8}$$

therefore the term $\max_{j \leq k} \|\boldsymbol{r}_j\|$ in (2.5) is at least 2γ. Using this result it follows that a conservative estimate for the right hand side in (2.5) is γ^2, more specifically it scales as

$\Omega(\gamma^2)$ in the limit $\gamma \to \infty$ for this system. Therefore, depending on γ the upper bound for the residual gap can be arbitrarily large.

By computing the true residual from the definition $\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$ the gap is reduced to zero. However, this is expensive as it costs an additional Matrix-Vector product (MV). In [31] it is suggested to compute the true residual if one of the following conditions holds:

1. $\|\widehat{\boldsymbol{r}}_{:,1}\| < 0.01\|\boldsymbol{b}\|$ and $\|\boldsymbol{b}\| \leq \max_j \|\widehat{\boldsymbol{r}}^j_{:,1}\|$,

2. $\|\boldsymbol{b}\| \leq 0.01 \max_j \|\widehat{\boldsymbol{r}}^j_{:,1}\|$ and $\|\widehat{\boldsymbol{r}}_{:,1}\| < \max_j \|\widehat{\boldsymbol{r}}^j_{:,1}\|$,

where the maximum is taken over all residuals since the last computation of the true residual. By computing the true residual the residual gap can be significantly decreased with only a few extra MV. Similarly, $\boldsymbol{x}$ is updated in a ''group wise'' manner, collecting all the updates into a temporary vector then applying several updates at once to $\boldsymbol{x}$ in order to minimize the effects of round-off errors. For more details regarding the derivation of the reliable updating strategy and the precise conditions for the groupwise updates, the reader is referred to [24, 31].

An alternative strategy is applied in MATLAB's implementation of BiCGStab. If $\|\boldsymbol{r}\|/\|\boldsymbol{b}\|$ has decreased below the specified tolerance, the recursive residual is replaced by the true residual. This has the benefit of ensuring the residual is accurate, however, replacing the recursively computed residual by the true residual can destroy the BiCG process [24]. We will indeed show in numerical experiments that MATLAB's implementation can break down after this residual replacement.

2.3 Keeping the lowest residual solution

A final idea is applied in MATLAB's variant of BiCGStab [15]. One major difference between the baseline and MATLAB's variant of BiCGStab is keeping track of the $\boldsymbol{x}$ with the lowest residual. After every update of $\boldsymbol{x}$ and $\widehat{\boldsymbol{r}}_{:,1}$ the new $\boldsymbol{x}$ is stored separately until a solution is computed with an even lower residual. This idea has its merits, since the convergence of BiCGStab is in general not monotonic. If the method breaks down, the solution $\boldsymbol{x}$ computed last may be worse than a previously computed solution. Nevertheless it is important to keep in mind that, due to the residual gap, a previously computed $\boldsymbol{x}$ may not actually have a smallest true residual, even though it could have the smallest recursively computed residual as pointed out in Section 2.2. The effects of this modification are discussed in more detail in Section 4.2. In the next section we analyze the way the initial residual of the LMR method changes with each iteration.

2.4 Propagation of the initial residual

To illustrate how the initial residual is affected by the BiCGStab process we investigate a simpler Krylov subspace method. The LMR algorithm as presented in [19] is given in

Algorithm 2. In essence this method makes a step in the direction of the residual, and determines a step size such that the norm of the next residual is minimal. This method can also be interpreted as GMRES(1), or the second half of BiCGStab (Line 16-19 in baseline BiCGStab).

Algorithm 2 LMR algorithm

1: **function** LMR($\boldsymbol{A}$,$\boldsymbol{b}$,$\boldsymbol{x}$,tol)
2: $\quad\boldsymbol{r}\leftarrow\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}$
3: $\quad$**while** $\|\boldsymbol{r}\|_2 > tol$ **do**
4: $\quad\quad\omega\leftarrow\frac{\langle\boldsymbol{A}\boldsymbol{r},\boldsymbol{r}\rangle}{\langle\boldsymbol{A}\boldsymbol{r},\boldsymbol{A}\boldsymbol{r}\rangle}$
5: $\quad\quad\boldsymbol{x}\leftarrow\boldsymbol{x}+\omega\boldsymbol{r}$
6: $\quad\quad\boldsymbol{r}\leftarrow\boldsymbol{r}-\omega\boldsymbol{A}\boldsymbol{r}$
7: $\quad$**end while**
8: $\quad$**return** $\boldsymbol{x}$
9: **end function**

To illustrate the LMR process, we consider a system of arbitrary size similar to the example shown in (2.3) with $\lambda=1$. It is shown in Appendix A that for a linear system involving a bidiagonal Toeplitz matrix $\boldsymbol{A}$ of size $N\times N$, with -1 on the first sub-diagonal and 1 on the diagonal the residual propagates toward the right. Such a bidiagonal Toeplitz matrix is a model for a one-dimensional advection equation with an upwind discretization. More specifically, given a right hand side $\boldsymbol{e}_1$ and initial guess $\boldsymbol{x}=\boldsymbol{0}$ the k-th LMR residual has elements that follow a binomial distribution;

$$r_i^k=\binom{k}{i-1}(\tfrac{1}{2})^k,\quad i=1,2,\cdots,k+1<N, \tag{2.9}$$

with $\boldsymbol{r}^0=\boldsymbol{e}_1$. Importantly, note that

$$\langle\boldsymbol{r}^0,\boldsymbol{r}^k\rangle=\frac{1}{2^k}, \tag{2.10}$$

i.e., the inner product of the first and the k-th residual decreases *exponentially* with k.

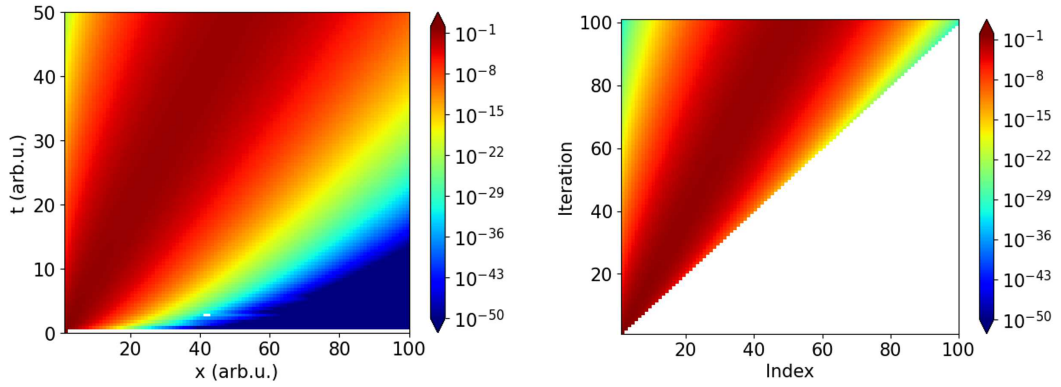
Another interpretation of the LMR algorithm exists if $\boldsymbol{A}$ is positive definite, since for such matrices $\omega>0$. Line 6 of Algorithm 2 can then be written as

$$\frac{\boldsymbol{r}^{k+1}-\boldsymbol{r}^k}{\omega}=-\boldsymbol{A}\boldsymbol{r}^k, \tag{2.11}$$

which is the forward Euler discretization with step size ω of the ODE system

$$\frac{\mathrm{d}\boldsymbol{r}}{\mathrm{d}t}=-\boldsymbol{A}\boldsymbol{r}. \tag{2.12}$$

The LMR algorithm can be seen as time integration method for a linear ODE system. For ADR problems the matrix $\boldsymbol{A}$ represents a discretized ADR operator, and $\boldsymbol{r}$ a discrete



(a) Exact solution of (2.12) (b) LMR residual

Figure 1: Comparison of the exact solution of the time dependent ODE system (2.12) and the LMR solution (forward Euler approximation) for the bidiagonal Toeplitz matrix of size 100. Note that for the first 100 iterations $\omega = \frac{1}{2}$. Importantly, this shows that the residual is advected throughout the domain, one grid cell per iteration.

approximation to a boundary value problem with homogeneous Dirichlet boundary conditions. The exact solution to (2.12) is compared with the LMR residual in Fig. 1 for the model upwind discretization resulting in a bidiagonal Toeplitz matrix of size 100. It is shown in Appendix B that the discretization matrices obtained later on in this paper are indeed positive definite. From Fig. 1 it can be concluded that the residual starts primarily on the left and propagates toward the interior of the domain. Additionally, it can be seen that the LMR residual indeed resembles the exact solution of the ODE system.

3 The discrete advection-diffusion-reaction equation

To investigate the performance of baseline BiCGStab and the modifications presented in Section 2, we introduce the advection-diffusion-reaction equation as a model problem. First, the scalar advection-diffusion-reaction equation is converted to dimensionless form, to investigate the iterative solvers for the entire parameter space. Second, the discretization scheme is outlined to obtain a set of difference equations, yielding a linear system. Third, the eigenvalues of the resulting linear system are obtained to show that the obtained matrix has real, positive eigenvalues.

3.1 Dimensionless form of the ADR equation

The scalar advection-diffusion-reaction equation, used here as a test problem, is relevant for a wide variety of physical systems, describing for example Fickian diffusion in a binary mixture or describing a temperature distribution. Starting from the general ADR

equation given by Eq. (2.13) in Ref. [6], with a linearized source term the equation reads

$$\frac{\partial \xi \varphi}{\partial t} + \nabla \cdot (\vec{u}\varphi - \epsilon \nabla \varphi) = s_C - s_P \varphi. \tag{3.1}$$

Here $\vec{u}$ is the velocity field, $\epsilon > 0$ a diffusion coefficient, s_C and $s_P \varphi$ a constant and linear reaction term, respectively, and φ the quantity of interest. For clarity $\vec{u}$, ϵ, s_C and s_P are taken to be constant, furthermore $s_C, s_P > 0$. In absence of transport, $s_C - s_P \varphi_{eq} = 0$ where φ_{eq} is the (chemical) equilibrium value of φ. Then after the substituting $\varphi_{eq} := s_C / s_P$, the resulting model equation is given by

$$\frac{\partial \xi \varphi}{\partial t} + \nabla \cdot (\vec{u}\varphi - \epsilon \nabla \varphi) = -s_P(\varphi - \varphi_{eq}). \tag{3.2}$$

The coefficient $\xi > 0$ denotes the responsiveness of the system; for small ξ the solution can vary more rapidly compared to large values of ξ. Physically ξ can, for example, take the role of a specific heat capacity or mass density.

Consider a Cartesian three-dimensional coordinate system, then Eq. (3.2) can be expanded as follows

$$\frac{\partial \xi \varphi}{\partial t} + \sum_{i=1}^{3} \frac{\partial}{\partial x_i} \left(u_i \varphi - \epsilon \frac{\partial \varphi}{\partial x_i} \right) = -s_P(\varphi - \varphi_{eq}), \tag{3.3}$$

with x_i the i-th Cartesian coordinate, and u_i the corresponding component of the advection velocity. Next, we aim to make (3.3) dimensionless. To commence, a scaling is performed; we scale the time t by a characteristic timescale T, and the coordinates x_i are scaled by the length scales L_i for the i-th direction;

$$t^* := t/T, \quad x_i^* := (x_i - x_{i,\min})/L_i. \tag{3.4}$$

The values of the characteristic variables are to be determined later. Using the newly introduced quantities T and L_i, the partial derivatives can be written as

$$\frac{\partial}{\partial t} = \frac{1}{T}\frac{\partial}{\partial t^*}, \quad \frac{\partial}{\partial x_i} = \frac{1}{L_i}\frac{\partial}{\partial x_i^*}. \tag{3.5}$$

Assuming constant ξ, and combining with the differentiation rules from (3.5), Eq. (3.3) can be simplified to

$$\frac{\xi}{T}\frac{\partial \varphi}{\partial t^*} + \epsilon \sum_{i=1}^{3} \frac{1}{L_i^2}\frac{\partial}{\partial x_i^*} \left(\frac{u_i L_i}{\epsilon}\varphi - \frac{\partial \varphi}{\partial x_i^*} \right) = -s_P(\varphi - \varphi_{eq}). \tag{3.6}$$

Furthermore, by introducing Péclet numbers for each of the Cartesian directions

$$\mathrm{Pe}_i := \frac{u_i L_i}{\epsilon}, \tag{3.7}$$

we can simplify the notation. These Péclet numbers can be interpreted as the relative strength of advection in the direction $\vec{e}_i$ compared to diffusion. After some algebraic manipulations the advection-diffusion-reaction equation becomes

$$\frac{\partial\varphi}{\partial t^*}+\frac{T\epsilon}{\xi}\sum_{i=1}^{3}\frac{1}{L_i^2}\frac{\partial}{\partial x_i^*}\left(\mathrm{Pe}_i\varphi-\frac{\partial\varphi}{\partial x_i^*}\right)=-s_\mathrm{P}(\varphi-\varphi_\mathrm{eq})\frac{T}{\xi}. \tag{3.8}$$

We introduce the transport frequencies $\nu_{\mathrm{t},i}$ and the reaction frequency ν_r, defined as

$$\nu_{\mathrm{t},i}:=\frac{\epsilon}{\xi L_i^2},\quad \nu_\mathrm{r}:=\frac{s_\mathrm{P}}{\xi}. \tag{3.9}$$

We then obtain the advection-diffusion-reaction equation in terms of frequencies $\nu_{\mathrm{t},i}$ and ν_r as;

$$\frac{\partial\varphi}{\partial t^*}+T\sum_{i=1}^{3}\nu_{\mathrm{t},i}\frac{\partial}{\partial x_i^*}\left(\mathrm{Pe}_i\varphi-\frac{\partial\varphi}{\partial x_i^*}\right)=-\nu_\mathrm{r}T(\varphi-\varphi_\mathrm{eq}). \tag{3.10}$$

Next, by taking T such that $\nu_\mathrm{r}T=1$ the Damköhler numbers Da_i can be introduced as follows,

$$\nu_{\mathrm{t},i}T=\frac{\nu_{\mathrm{t},i}}{\nu_\mathrm{r}}=:\frac{1}{\mathrm{Da}_i},\quad \mathrm{Da}_i=\frac{s_\mathrm{P}L_i^2}{\epsilon}. \tag{3.11}$$

Finally, the dimensionless time-dependent advection-diffusion-reaction equation is obtained by dividing both sides by a reference value φ_ref, such that $\varphi^*:=\varphi/\varphi_\mathrm{ref}$, which results in

$$\frac{\partial\varphi^*}{\partial t^*}+\sum_{i=1}^{3}\frac{1}{\mathrm{Da}_i}\frac{\partial}{\partial x_i^*}\left(\mathrm{Pe}_i\varphi^*-\frac{\partial\varphi^*}{\partial x_i^*}\right)=-(\varphi^*-\varphi_\mathrm{eq}^*). \tag{3.12}$$

If the characteristic length scales $L_i=L$ are the same in each of the Cartesian directions, and assuming a stationary solution, Eq. (3.12) reduces to

$$\sum_{i=1}^{3}\frac{\partial}{\partial x_i^*}\left(\mathrm{Pe}_i\varphi^*-\frac{\partial\varphi^*}{\partial x_i^*}\right)=-\mathrm{Da}(\varphi^*-\varphi_\mathrm{eq}^*), \tag{3.13}$$

where now all Damköhler numbers are the same; Da. For ease of notation, * is omitted in the following sections. In the next section we discretize Eq. (3.13) and obtain a linear system which can be used to approximate the exact solution of (3.13).

3.2 Discretization

3.2.1 One-dimensional scheme

To discretize (3.13), for ease of exposition, we start with the one-dimensional equivalent with $\varphi_\mathrm{eq}=0$, since φ_eq will only affect the right hand side of the resulting linear system. The ADR-equation can then be written as

$$\frac{\mathrm{d}}{\mathrm{d}x}\Gamma(\varphi(x))=-\mathrm{Da}\,\varphi(x), \tag{3.14a}$$

where Γ is the flux, given by

$$\Gamma(\varphi) := \mathrm{Pe}\,\varphi - \frac{\mathrm{d}\varphi}{\mathrm{d}x}. \tag{3.14b}$$

To discretize (3.14a), we use the Finite Volume Method (FVM). In the FVM method the domain is subdivided into a finite number of disjunct intervals; referred to as control volumes. Here a cell-centered approach is applied on a uniform grid; in such a configuration φ has to be computed at the nodal points x_i. Control volumes are defined around these nodal points; the i-th control volume extends over $[x_{i-1/2}, x_{i+1/2}]$, where $x_{i\pm1/2} := \frac{1}{2}(x_i + x_{i\pm1})$ and the width of this volume is defined as the grid size $\Delta x := x_{i+1/2} - x_{i-1/2}$.

Previously, the Péclet and Damköhler numbers were defined with respect to a length scale L, in the following we take the length of each grid cell as the respective length scale, that is;

$$L = \Delta x =: h, \tag{3.15}$$

defining them as the grid Péclet and Damköhler numbers. Integrating (3.14a) over a control volume and approximating the integral over the source term with the midpoint rule we obtain the discrete conservation law for each interval;

$$F_{i+1/2} - F_{i-1/2} = -h\mathrm{Da}\,\varphi_i, \tag{3.16}$$

with φ_i a numerical approximation of $\varphi(x_i)$. Several expressions for the numerical flux $F_{i+1/2}$ at $x_{i+1/2}$ exist, depending on the discretization scheme used. Here we use the exponential flux given in [6, p. 86];

$$F_{i+1/2} = \frac{1}{h}(\mathrm{B}(-\mathrm{Pe})\varphi_i - \mathrm{B}(\mathrm{Pe})\varphi_{i+1}), \tag{3.17}$$

where $\mathrm{B}(z)$ is the generating function for the Bernoulli numbers; in short the Bernoulli function, defined as [36, p. 40] [37, p. 804]

$$\mathrm{B}(z) := \begin{cases} \frac{z}{\exp(z)-1}, & z \neq 0, \\ 1, & z = 0. \end{cases} \tag{3.18}$$

The resulting exponential scheme is obtained from the discrete conservation law (3.16) resulting in

$$\frac{1}{h}(-\mathrm{B}(\mathrm{Pe})\varphi_{i+1} + (\mathrm{B}(-\mathrm{Pe}) + \mathrm{B}(\mathrm{Pe}))\varphi_i - \mathrm{B}(-\mathrm{Pe})\varphi_{i-1}) = -h\mathrm{Da}\,\varphi_i. \tag{3.19}$$

Assuming Dirichlet boundary conditions, the discretization matrix without linear source term, is given by a tridiagonal Toeplitz matrix. For the exponential scheme the following discretization matrix is obtained;

$$\boldsymbol{A}_x = \frac{1}{h}\mathrm{tridiag}(-\mathrm{B}(-\mathrm{Pe}), \mathrm{B}(-\mathrm{Pe}) + \mathrm{B}(\mathrm{Pe}), -\mathrm{B}(\mathrm{Pe})), \tag{3.20}$$

with tridiag(a,b,c) indicating a tridiagonal Toeplitz matrix with a on the sub-diagonal, b on the diagonal and c on the super-diagonal. Note that for strong positive advection ($\mathrm{Pe}\to\infty$) the matrix $\boldsymbol{A}_x$ becomes bi-diagonal, similar to the small model system shown in Eq. (2.3), and the problem discussed in Section 2.4. Since this is a tridiagonal Toeplitz matrix a closed form expression for the eigenvalues is known [38], namely they are given by

$$\lambda_k(\boldsymbol{A}_x)=\frac{1}{h}\Big(\mathrm{B}(-\mathrm{Pe})+\mathrm{B}(\mathrm{Pe})+2\sqrt{\mathrm{B}(\mathrm{Pe})\mathrm{B}(-\mathrm{Pe})}\cos\Big(\frac{k\pi}{M-1}\Big)\Big),\quad k=1,2,\cdots,M-2, \tag{3.21}$$

with M the number of grid points. Note that since $\mathrm{B}(\mathrm{Pe})>0$ and the cosine factor is greater than -1, we can obtain a strict lower bound, as

$$\lambda_k>\frac{1}{h}\Big(\mathrm{B}(-\mathrm{Pe})+\mathrm{B}(\mathrm{Pe})-2\sqrt{\mathrm{B}(\mathrm{Pe})\mathrm{B}(-\mathrm{Pe})}\Big)=\frac{1}{h}\Big(\sqrt{\mathrm{B}(-\mathrm{Pe})}-\sqrt{\mathrm{B}(\mathrm{Pe})}\Big)^2\geq 0. \tag{3.22}$$

3.2.2 Three-dimensional scheme

To extend the discussion of Section 3.2.1 to three-dimensional problems, the Kronecker sum is used. The Kronecker sum $\oplus$, as defined is given by [39, p. 268]

$$\boldsymbol{P}\oplus\boldsymbol{Q}:=\boldsymbol{I}_n\otimes\boldsymbol{P}+\boldsymbol{Q}\otimes\boldsymbol{I}_m, \tag{3.23}$$

with $\boldsymbol{P}\in\mathbb{R}^{m\times m}$, $\boldsymbol{Q}\in\mathbb{R}^{n\times n}$ and $\boldsymbol{I}_m$, $\boldsymbol{I}_n$ the identity matrices of size $m\times m$ and $n\times n$, respectively. Here $\otimes$ is the well-known Kronecker product (tensor product) defined for two matrices $\boldsymbol{P}\in\mathbb{R}^{N\times N}$ and $\boldsymbol{Q}$ as

$$\boldsymbol{P}\otimes\boldsymbol{Q}=\begin{bmatrix} p_{1,1}\boldsymbol{Q} & \cdots & p_{1,N}\boldsymbol{Q} \\ \vdots & \ddots & \vdots \\ p_{N,1}\boldsymbol{Q} & \cdots & p_{N,N}\boldsymbol{Q}\end{bmatrix}. \tag{3.24}$$

It is then possible to construct the matrix $\boldsymbol{A}_{3\mathrm{D}}$ from three one-dimensional discretization matrices using the Kronecker sum, similar to the procedure in [40].

For a three-dimensional problem on a cube of sides $(M-1)h$ with constant coefficients and cubical cells with sides of length h, the discretization matrix excluding the source term can be written as

$$\boldsymbol{A}_{3\mathrm{D}}=(\boldsymbol{A}_x\oplus\boldsymbol{A}_y\oplus\boldsymbol{A}_z)h^2, \tag{3.25}$$

where $\boldsymbol{A}_x$, $\boldsymbol{A}_y$ and $\boldsymbol{A}_z$ are the discretization matrices corresponding to the one-dimensional problem for each direction as laid out in Section 3.2.1. Applying (3.23) twice, and using the associative property of the Kronecker sum it can be shown that

$$\boldsymbol{A}_x\oplus\boldsymbol{A}_y\oplus\boldsymbol{A}_z=\boldsymbol{I}\otimes\boldsymbol{I}\otimes\boldsymbol{A}_x+\boldsymbol{I}\otimes\boldsymbol{A}_y\otimes\boldsymbol{I}+\boldsymbol{A}_z\otimes\boldsymbol{I}\otimes\boldsymbol{I}, \tag{3.26}$$

where each of the identity matrices is of size $M-2$, the number of grid cells in each direction.

Next, we show how the eigenvalues of $\boldsymbol{A}_{3\mathrm{D}}$ can be obtained. Consider a vector $\boldsymbol{v}$ as right-eigenvector of $\boldsymbol{P}$ with eigenvalue λ and $\boldsymbol{w}$ a right-eigenvector of $\boldsymbol{Q}$ with eigenvalue μ, then by using the mixed-product property of tensor products it follows that [39, p. 244]

$$\begin{aligned}(\boldsymbol{I}_n\otimes\boldsymbol{P}+\boldsymbol{Q}\otimes\boldsymbol{I}_m)(\boldsymbol{w}\otimes\boldsymbol{v}) &= \boldsymbol{I}_n\boldsymbol{w}\otimes\boldsymbol{P}\boldsymbol{v}+\boldsymbol{Q}\boldsymbol{w}\otimes\boldsymbol{I}_m\boldsymbol{v}\\ &= \boldsymbol{w}\otimes\lambda\boldsymbol{v}+\mu\boldsymbol{w}\otimes\boldsymbol{v}\\ &= (\lambda+\mu)(\boldsymbol{w}\otimes\boldsymbol{v}).\end{aligned} \tag{3.27}$$

Thus $\boldsymbol{w}\otimes\boldsymbol{v}$ is an eigenvector of $\boldsymbol{P}\oplus\boldsymbol{Q}$ with eigenvalue $\lambda+\mu$, conform Theorem 4.4.5 in [39, p. 268]. Consequently, if λ_x is an eigenvalue of $\boldsymbol{A}_x$, λ_y an eigenvalue of $\boldsymbol{A}_y$ and λ_z an eigenvalue of $\boldsymbol{A}_z$, then $\lambda_x+\lambda_y+\lambda_z$ is an eigenvalue of $\boldsymbol{A}_{3\mathrm{D}}$, resulting in an expression for the eigenvalues of $\boldsymbol{A}_{3\mathrm{D}}$;

$$\lambda_{(i,j,k)}(\boldsymbol{A}_{3\mathrm{D}}) = (\lambda_i(\boldsymbol{A}_x)+\lambda_j(\boldsymbol{A}_y)+\lambda_k(\boldsymbol{A}_z))h^2, \quad i,j,k=1,2,\cdots,M-2. \tag{3.28}$$

Finally, the effect of the linear source term has to be taken into account, namely a shift of the eigenvalues by $h^3\mathrm{Da}$ resulting in the final set of eigenvalues for the discretization matrix;

$$\lambda_{(i,j,k)}(\boldsymbol{A}) = (\lambda_i(\boldsymbol{A}_x)+\lambda_j(\boldsymbol{A}_y)+\lambda_k(\boldsymbol{A}_z))h^2+h^3\mathrm{Da}, \quad i,j,k=1,2,\cdots,M-2. \tag{3.29}$$

This shows that all eigenvalues of $\boldsymbol{A}$ are positive and real for $\mathrm{Da}\geq 0$. As a result, for all values of Pe and Da which will be used in the numerical experiments later, the linear system is invertible and only has positive, real eigenvalues. Furthermore, in Appendix B it is shown that $\boldsymbol{A}$ is also positive definite.

In [21] an example was shown involving an advection problem where the system matrix following from discretization with the central differencing scheme contains eigenvalues with large imaginary parts. In such a case BiCGStab($L>1$) is shown to outperform BiCGStab. However, it should also be noted that for problems with strong advection the central difference scheme yields spurious oscillations in the solution. Therefore, in our experiments we consider the exponential scheme which does not result in unphysical behavior for strong advection.

In the numerical experiments of Section 4 we illustrate that both of MATLAB's versions of BiCGStab and BiCGStab(2) do not converge for advection dominated problems with the exponential scheme used here. Moreover, we also investigate modifications to the baseline BiCGStab algorithm and show a potential mitigation by using a different choice of $\widetilde{\boldsymbol{r}}$.

4 Results and discussion

In this section we investigate the reliability of BiCGStab, IDR and their various implementations. To do this an ADR-equation is discretized in 3D and the true relative residuals $\|\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}_k\|/\|\boldsymbol{b}\|$ are compared over the entire range of Péclet and Damköhler numbers. Next, convergence as function of the number of Matrix-Vector products (MV) is

compared for four specific combinations of Péclet and Damköhler numbers. Finally, a numerical experiment is conducted to elaborate on the effect of the residual propagating through the domain, and the effect of choosing a random $\tilde{\boldsymbol{r}}$ in BiCGStab.

4.1 Convergence as function of Péclet and Damköhler numbers

To investigate the convergence of BiCGStab and IDR, a model problem is set up to investigate the entire range of Péclet and Damköhler numbers. We solve (3.13) for φ on the unit cube. To investigate the entire range of Péclet and Damköhler numbers we choose

$$\begin{aligned} &\vec{u}=\frac{u}{\sqrt{3}}(\vec{e}_x+\vec{e}_y+\vec{e}_z), \quad \epsilon=1, \quad \varphi_{\text{eq}}=0, \\ &\varphi(0,y,z)=\varphi(x,1,z)=\varphi(x,y,1)=1, \\ &\varphi(1,y,z)=\varphi(x,0,z)=\varphi(x,y,0)=0, \end{aligned} \tag{4.1}$$

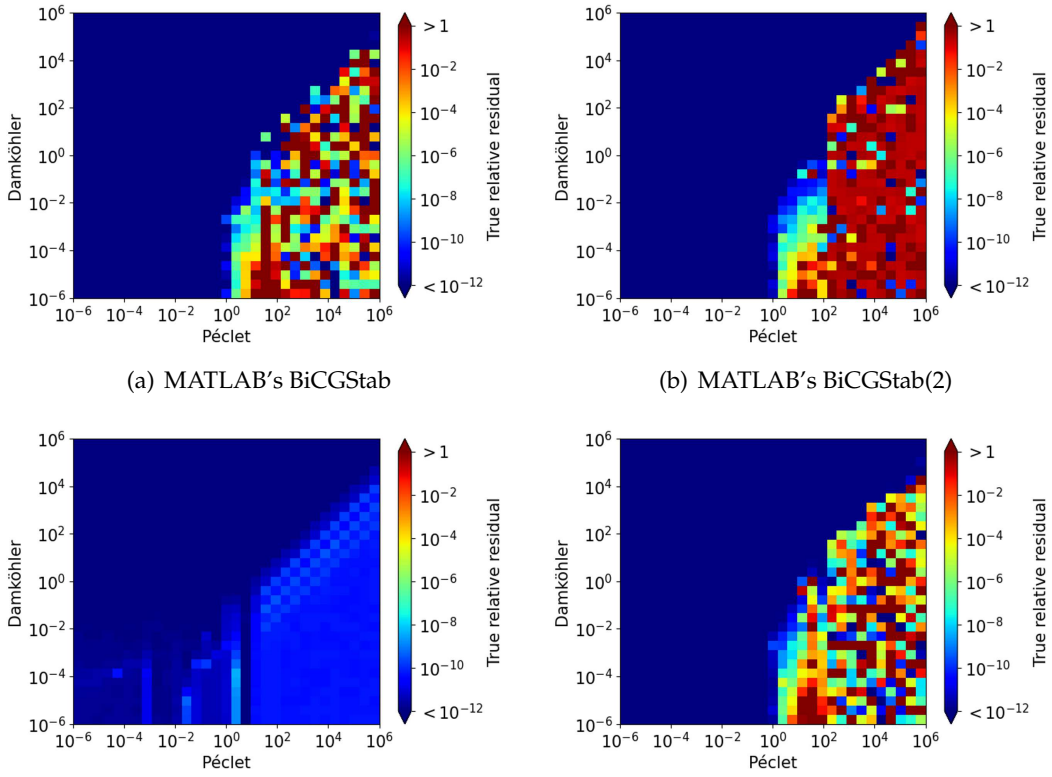
for varying u, and s_{P}. The grid contains $M=101$ grid cells for each of the three Cartesian directions such that h=1/100. After discretization with the exponential scheme described in Section 3.2 a linear system with 99^3=970,299 unknowns is obtained. No preconditioner is used and the initial guess is set to zero. Both Pe and Da use the grid spacing h as the characteristic length scales. The grid Péclet and Damköhler numbers in (3.13) are varied over the range $[10^{-6},10^{6}]$.

Each of the iterative solvers is given a maximum of 10^4 MV, and the tolerance is set such that the relative residual $\|\boldsymbol{r}\|/\|\boldsymbol{b}\|$ is smaller than 10^{-12}. We will show that the most efficient solver in terms of MV requires less than 10% of this maximum to converge. I.e., if a solver does not converge, increasing the maximum number of iterations in an attempt to enforce convergence would be inefficient, and may not resolve the issue either. After completion of the iteration we compare the true residual for all four methods in Fig. 2.

It can be seen in Figs. 2(a), 2(b) and 2(d) that none of the BiCGStab variants are successful in solving the discretized advection-diffusion-reaction problem for all Péclet and Damköhler numbers. However, Fig. 2(c) shows that IDR(4) is close to achieving the prescribed tolerance for a wide range of Péclet and Damköhler numbers.

BiCGStab(2) does not show improved reliability compared to BiCGStab. This is expected based on the eigenvalues, since the eigenvalues of the discretization matrix are real as pointed out in Section 3.1. The ability to handle matrices with complex eigenvalues efficiently is one of the main advantages of BiCGStab(2) over BiCGStab. However, for real eigenvalues this advantage is not relevant. As a result there is little difference between the regions where MATLAB's implementations of BiCGStab and BiCGStab(2) converge.

In addition, it can be seen that MATLAB's implementation of BiCGStab performs similarly to the baseline BiCGStab algorithm. This is due to a common issue related to propagation of the initial residual. Both of MATLAB's implementation and the baseline algorithm share this issue, which is discussed in more detail in Section 4.4. Most



(a) MATLAB's BiCGStab

(b) MATLAB's BiCGStab(2)

(c) IDR(4)

(d) Baseline BiCGStab

Figure 2: True residual of the boundary value problem given by (3.13) and (4.1). Comparison between several solvers.

importantly, neither solver converges reliably for problems with dominant advection, especially when only a weak source term is present.

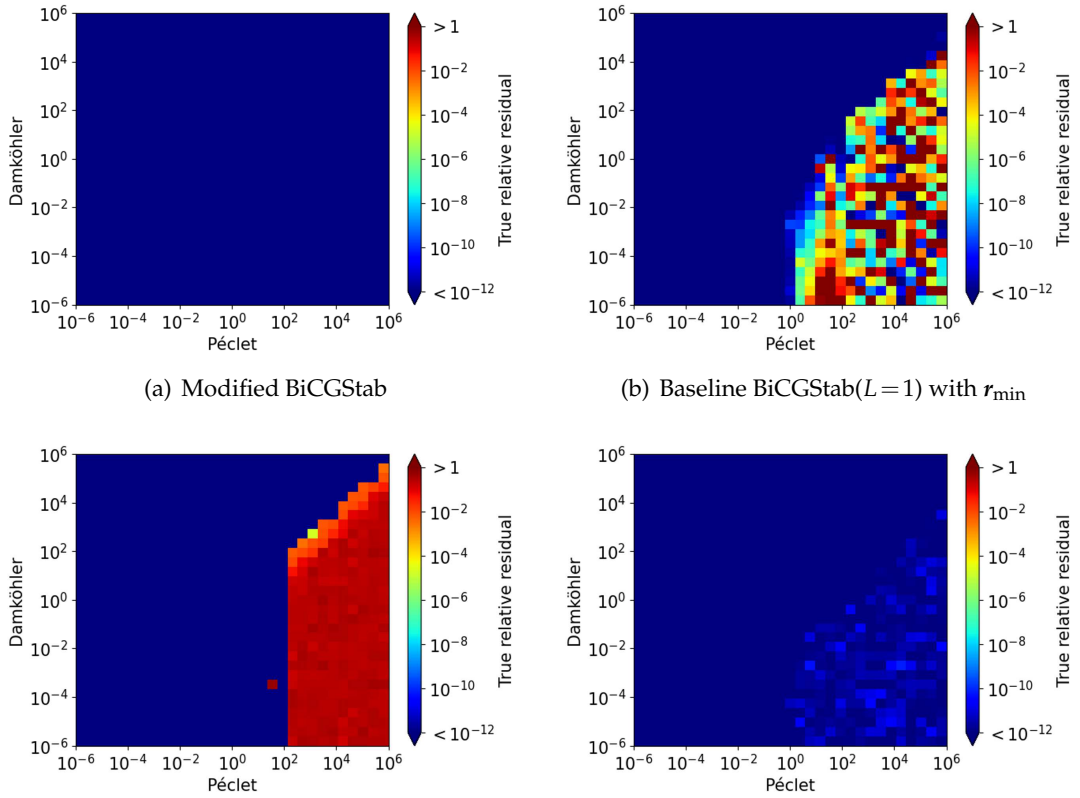
In the next section we show the effects of the modifications presented in Section 2 when applied to the baseline BiCGStab algorithm.

4.2 Comparing modified solvers

To show the effects of the modifications to BiCGStab as discussed in Section 2, we first show a version of BiCGStab with all modifications enabled;

- random shadow residual $\widetilde{\boldsymbol{r}}$,
- reliable updating scheme,
- keeping the best intermediate solution.

The true residual computed from the solutions returned by the modified BiCGStab algorithm with all three modifications *enabled* is shown in Fig. 3(a). It can be seen in this figure



(a) Modified BiCGStab

(b) Baseline BiCGStab($L=1$) with $\boldsymbol{r}_{\min}$

(c) Modified BiCGStab without random $\widetilde{\boldsymbol{r}}$

(d) Modified BiCGStab without reliable update

Figure 3: True residual of the system boundary value problem given by (3.13) and (4.1). Comparison between BiCGStab and several modified variants.

that for all Péclet and Damköhler numbers the modified version of BiCGStab converges to the prescribed tolerance. Next, we *enable* or *disable* one of the modifications in separate figures to see their effect on the final residual.

Fig. 3(b) shows that the effect of storing the solution corresponding to the lowest residual thus far, $\boldsymbol{r}_{\min}$, compared with the baseline algorithm shown in Fig. 2(d) is negligible. Furthermore, if the solver were to converge, the benefit of this modification is limited as well. If the solver converges, modified BiCGStab always returns the solution computed in the last iteration. Thus if the recursively computed residual is accurate, and the solver does not stagnate, then keeping track of the best intermediate solution is redundant. However, this may have an unwanted effect if the maximum number of iterations set is small. If there is no improvement over the initial residual, BiCGStab would return the initial guess as a solution.

In Fig. 3(c) we *disable* the use of a random $\widetilde{\boldsymbol{r}}$, and use the default choice $\widetilde{\boldsymbol{r}}=\boldsymbol{r}_0=\boldsymbol{b}-\boldsymbol{A}\boldsymbol{x}_0$. Here it can be observed that the choice of $\widetilde{\boldsymbol{r}}$ has a substantial effect on the final residual

for the region roughly given by $\mathrm{Pe} > 10^2$ and $\mathrm{Da} < \mathrm{Pe}$. Compared to Fig. 3(a) it is clear that for these problems the choice of $\widetilde{\boldsymbol{r}}$ plays a significant role.

The cause of the convergence issues when using the default choice of $\widetilde{\boldsymbol{r}} = \boldsymbol{r}_0$ is a manifestation of the three-dimensional equivalent of the scenario presented in Eq. (2.3). More specifically, the residual starts near one of the boundaries and propagates in a wave-like manner through the domain. The important consequence of this is that the initial residual only contains non-zero elements near one of the boundaries and gradually these non-zero elements decrease in further iterations. This leads to the issue that $\langle \widetilde{\boldsymbol{r}}, \widehat{\boldsymbol{r}}_{:,1} \rangle \to 0$ and indirectly $\langle \widetilde{\boldsymbol{r}}, \widehat{\boldsymbol{u}}_{:,2} \rangle \to 0$. Reliable updating offers no improvement here, as the issue occurs even with accurate residuals. This effect is shown in more detail in Section 4.4.

The effect of the reliable updating scheme is shown in Fig. 3(d). In this figure we show BiCGStab with the modifications of random $\widetilde{\boldsymbol{r}}$ and keeping the best intermediate solution *enabled*, and reliable updating *disabled*. For a wide range of the Péclet and Damköhler numbers the solver converges to the required tolerance of 10^{-12}, except for some cases in the region $\mathrm{Pe} > 1$ and $\mathrm{Da} < 10^3$. Even though the effect of reliable updating is not as dramatic as choosing a different $\widetilde{\boldsymbol{r}}$, a good solver should not stop iterating prematurely. Therefore, the reliable updating scheme is nevertheless a beneficial addition to ensure the required tolerance has truly been achieved.

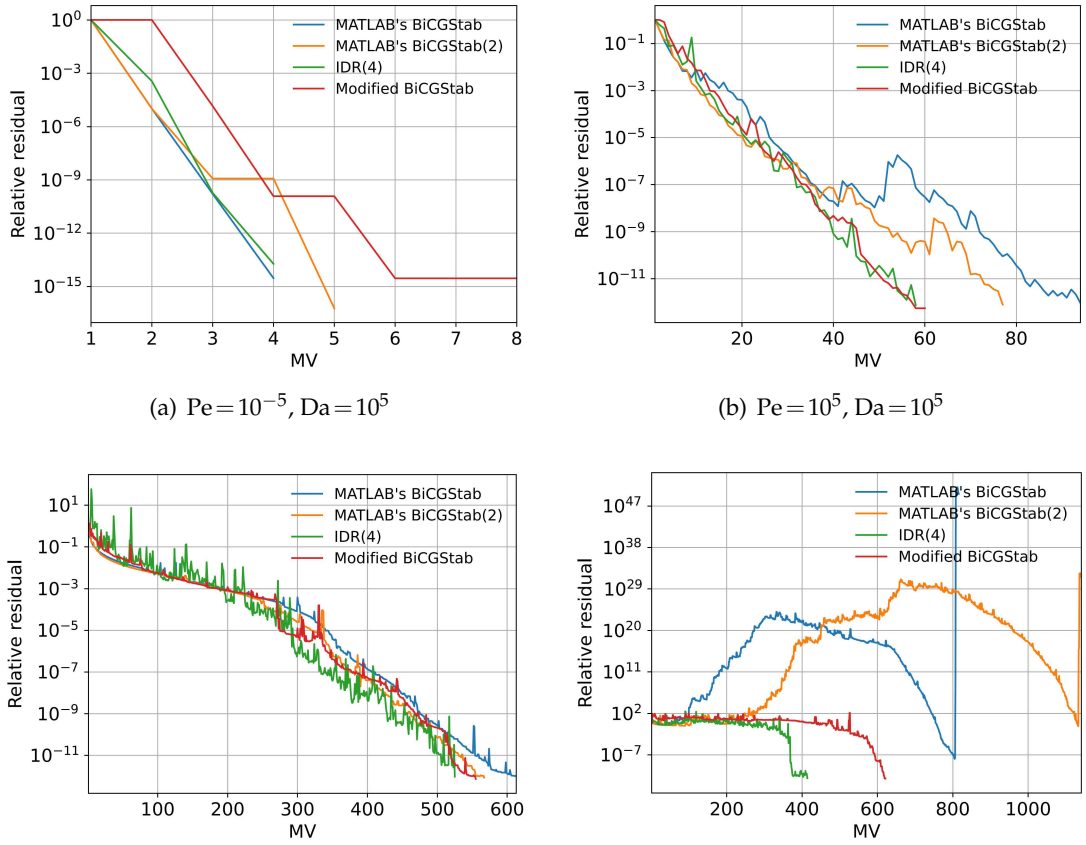
In the next section we investigate the evolution of the residual as function of the number of MV, for four selected combinations of Péclet and Damköhler numbers.

4.3 Comparing performance for specific Péclet and Damköhler numbers

The convergence as function of the number of matrix-vector products for four combinations of Péclet and Damköhler numbers is investigated. Fig. 4 shows the relative residual (as computed by the solver) $\|\boldsymbol{r}\| / \|\boldsymbol{b}\|$ as function of number of matrix-vector products. Here the same model problem is used as in the previous sections.

For large Damköhler and small Péclet numbers it can be seen in Fig. 4(a) that all methods converge within a few MV. When both the Péclet and Damköhler numbers are large it can be seen in Fig. 4(b) that the methods perform similarly for the first 40 MV, however after this point MATLAB's solvers start to slow down. For a problem with dominant diffusion, Fig. 4(c), all solvers converge smoothly and perform about the same. However, the number of MV required is much larger than the case with negligible diffusion shown in Fig. 4(b).

Fig. 4(d) highlights one of the points of interest with MATLAB's implementations. First, it can be seen that the residual *increases* significantly above the starting value for MATLAB's solvers. Second, the residual can be seen to sharply increase at approximately 800 MV and 1200 MV for MATLAB's versions of BiCGStab and BiCGStab(L), respectively. The cause of this is that after a certain number of iterations the recursively computed residual does decrease, however it strongly deviates from the true residual. When MATLAB's implementation detects stagnation, or when the recursively computed residual norm is below the tolerance, the true residual is computed. After this step the recursive



(a) $\mathrm{Pe}=10^{-5}$, $\mathrm{Da}=10^{5}$ (b) $\mathrm{Pe}=10^{5}$, $\mathrm{Da}=10^{5}$

(c) $\mathrm{Pe}=10^{-5}$, $\mathrm{Da}=10^{-5}$ (d) $\mathrm{Pe}=10^{5}$, $\mathrm{Da}=10^{-5}$

Figure 4: Relative residual of the boundary value problem given by (3.13) and (4.1) as function of the number of matrix-vector products (MV). Note that this figure shows the convergence behavior corresponding to four locations shown in Figs. 2 and 3.

residual is replaced by the true residual. However, shortly after this replacement of the recursively computed residual by the true residual, the solvers stagnate.

It can be seen in Fig. 4(d) that IDR(4) converges in the least number of matrix-vector products. Next, note that for the discretized problem, it takes at least 300 MV to propagate the boundary conditions throughout the domain if every MV only spreads the solution to neighbouring cells, similar to what was discussed for LMR in Section 2.4. Interestingly, the convergence graph for IDR(4) shows a plateau for the first approximately 300 MV. Similarly, it can be seen in Fig. 4(c) that there appears to be a speedup in convergence around 300 MV. For Fig. 4(a) it should be noted that the system matrix is strongly diagonally dominant, nearly becoming a multiple of the identity matrix, causing the fast convergence. Fig. 4(b) does not show a plateau region, presumably since the initial guess of zeros is already a good approximation throughout the domain.

4.4 Propagation of the residual for LMR and BiCGStab

In this section we investigate the hypothesis on the significant impact of replacing the default choice of $\widetilde{\boldsymbol{r}} = \boldsymbol{r}_0$ with a random $\widetilde{\boldsymbol{r}}$. It is conjectured that, similar to the model problem discussed in Section 2.4, for 2D and 3D problems the residual propagates into the interior domain, following the advection direction.

To show the effect of the residual propagating through the domain, as was suggested by Eq. (2.12), we consider a two-dimensional ADR problem on the unit square with the following parameters;

$$\begin{aligned} &\vec{u}\frac{u}{\sqrt{2}}(\vec{e}_x+\vec{e}_y), \quad \epsilon=1, \quad \varphi_{\text{eq}}=0 \\ &\varphi(0,y)=\varphi(x,1)=1, \\ &\varphi(1,y)=\varphi(x,0)=0, \end{aligned} \tag{4.2}$$

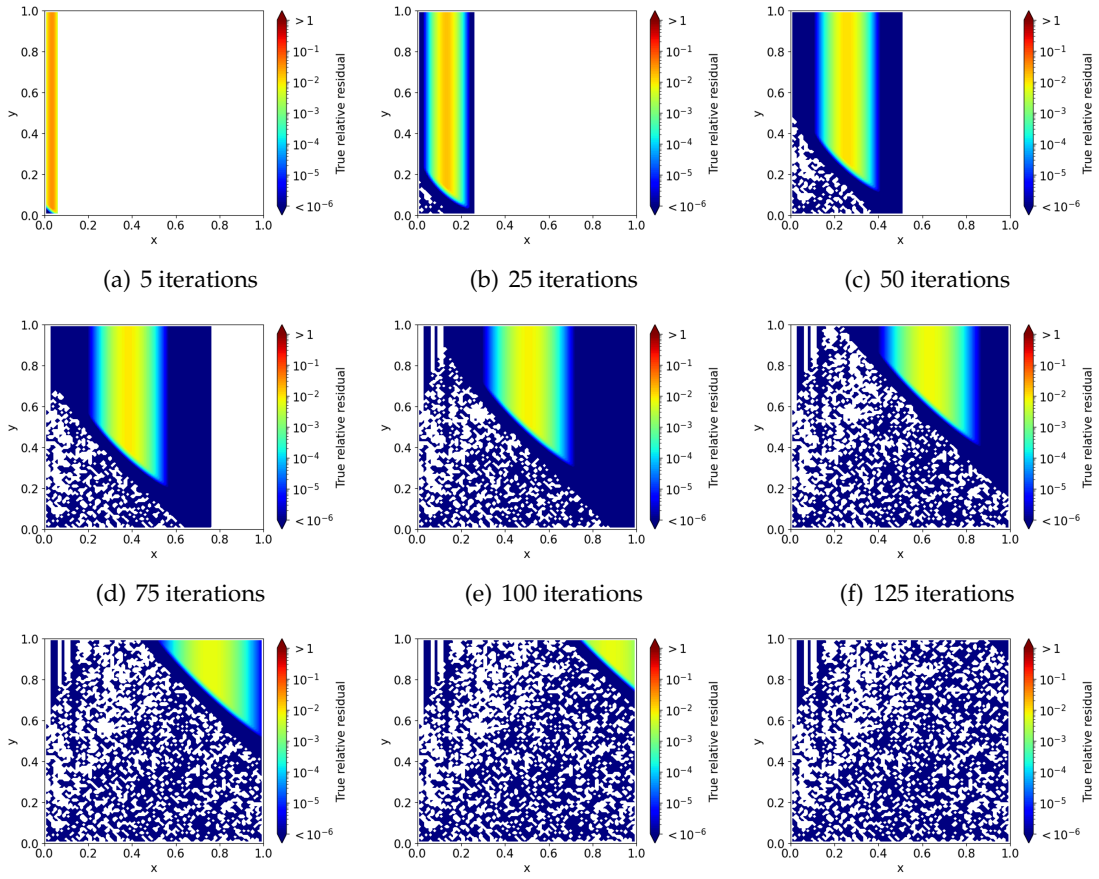
where again both u, and s_{P} are varied to investigate specific Péclet and Damköhler numbers. The system is discretized using the exponential scheme using 101 grid points per Cartesian direction. The Péclet number is set to 10^5 and the Damköhler number to 10^{-5}; the advection dominated regime. The resulting linear system is solved using a zero initial guess for both LMR and baseline BiCGStab, modified to include a random $\widetilde{\boldsymbol{r}}$.

In Fig. 5 two main effects can be seen. First, the non-zero residual is propagating in the x-direction. Second, a wave-like structure traveling in the $\vec{e}_x+\vec{e}_y$ direction. Since in the two-dimensional discretization scheme cells are coupled using a 5-point stencil, every element in the residual vector can affect at most 4 other elements per MV, viz. their direct neighbors. Therefore to have the grid cell representing the residual at $x=0$, $y=0$ affect the grid cell at $x=1$, $y=0$ would take $M-2$ MV. Similarly, it would take $2(M-1)$ MV to have the grid cell at $x=0$, $y=0$ affect the one at $x=1$, $y=1$, since the propagation follows a staircase pattern.

It can be seen in Fig. 5 that for LMR the initial residual is non-zero close to $x=0$ and zero throughout the rest of the domain, similar to the one-dimensional case shown in Fig. 1. For consecutive iterations the residual can be seen to propagate similar to a wave traveling in the $\vec{e}_x+\vec{e}_y$ direction. This is consistent with the interpretation of LMR solving a time-dependent version of the advection equation, as suggested by Eq. (2.12).

For BiCGStab a similar effect is observed, however, a dispersion-like phenomenon is also present; see Fig. 6. The residual is pushed into the direction $\vec{e}_x+\vec{e}_y$ as well, however, unlike LMR, the wavefront is not as sharp. It can be seen that the residual propagates in a wave-like manner through the domain prone to dispersion. Note that the initial residual is only non-zero on the left-most grid cells.

Importantly, since the residual propagates into the interior domain this results in $\langle \boldsymbol{r}_0, \boldsymbol{r}_k \rangle \to 0$ on line 9 of Algorithm 1. Consequently $\beta \to 0$, $\widehat{\boldsymbol{u}}_{:,1} \to \widehat{\boldsymbol{r}}_{:,1}$. Additional multiplications by $\boldsymbol{A}$ only shift the residual even more, $\langle \widetilde{\boldsymbol{r}}, \boldsymbol{A}\widehat{\boldsymbol{u}}_{:,1} \rangle \to 0$ and α becomes undetermined on line 13. Note, however, that since BiCGStab performs 2 MV per iteration, one might expect the wave traveling in the $\vec{e}_x+\vec{e}_y$ to have crossed through the entire domain after



(a) 5 iterations (b) 25 iterations (c) 50 iterations

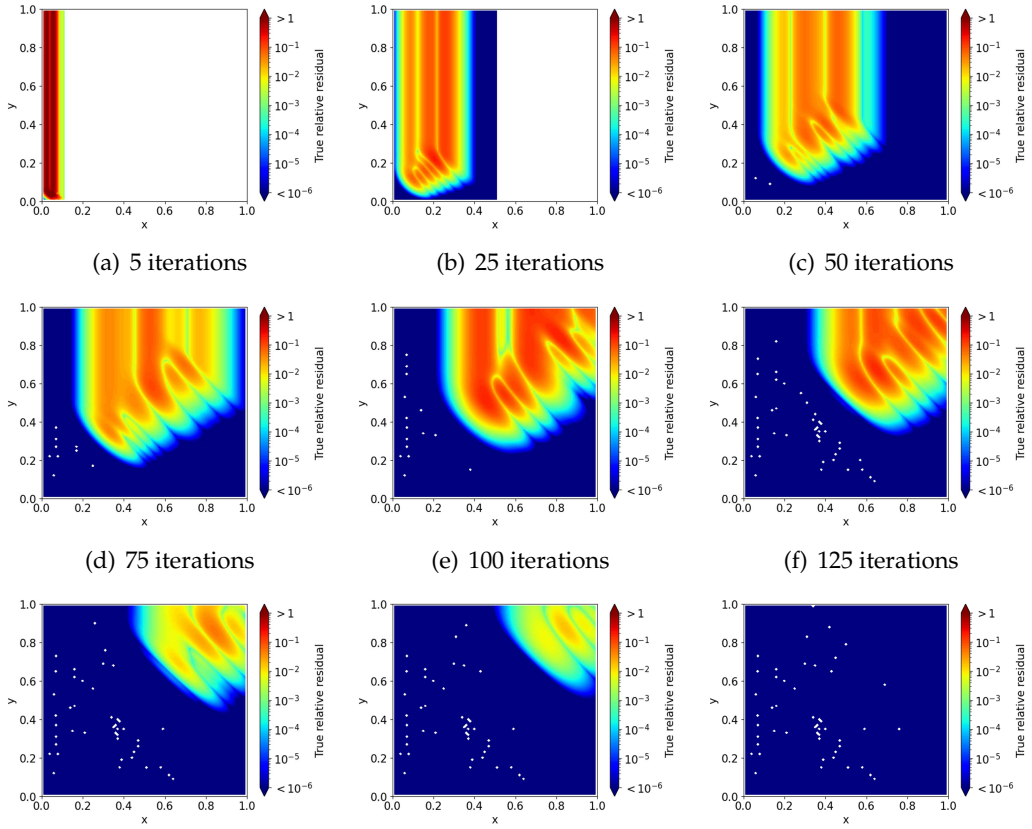
(d) 75 iterations (e) 100 iterations (f) 125 iterations

(g) 150 iterations (h) 175 iterations (i) 200 iterations

Figure 5: Relative residuals for LMR plotted in the $x-y$ plane for the model problem given by the parameters in (4.2). White in these figures indicates a residual of exactly 0 at a given position. Here it can be seen that the residual propagates in the direction of $\vec{u}$.

50 iterations. This does not seem to be the case, suggesting that only the LMR-part is responsible for the wave propagation.

The effect of propagating residuals is the suspected cause of BiCGStab failing to converge without a random $\tilde{\boldsymbol{r}}$, which can be seen by comparing Fig. 3(a) and Fig. 3(c). Note that this reason is completely different from the issue due to BiCGStab failing for complex eigenvalues which result from central difference discretization. Additionally, the propagating residuals suggest that the LMR method needs $\mathcal{O}(M)$ iterations to get through the first phase of convergence (the plateau in Fig. 7). It is suspected that this effect also relates to the conjecture of [41] which states the first phase of convergence is determined by the longest streamline for residual-minimizing Krylov subspace methods, in this case $2(M-2)$ cells. As it takes $2(M-2)$ iterations for the residual to propagate from $x=0, y=0$ to $x=1, y=1$, following only the grid cells in the direction of the advection velocity.



(a) 5 iterations (b) 25 iterations (c) 50 iterations

(d) 75 iterations (e) 100 iterations (f) 125 iterations

(g) 150 iterations (h) 175 iterations (i) 200 iterations

Figure 6: Relative residuals plotted in the $x-y$ plane for baseline BiCGStab with random $\tilde{r}$ for the problem described in (4.2) ($\tilde{r}=r_0$ breaks down). White in these figures indicates a residual of exactly 0 at a given position.

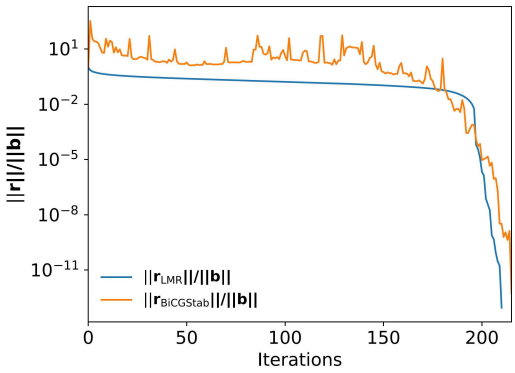


Figure 7: Relative residuals for LMR and baseline BiCGStab with random $\tilde{r}$ for the problem described in (3.13) and (4.2) as function of the number of iterations. Note that BiCGStab performs 2 MV per iteration, one in the BiCG part, one in the LMR part.

4.5 Preconditioned BiCGStab

Since linear systems are typically solved using preconditioned iterations, several numerical experiments using MATLAB's BiCGStab, have been performed for a number of preconditioners.

Specifically, we investigate if the use of a preconditioner can alleviate the convergence issues of MATLAB's BiCGStab, displayed in Fig. 2, in the regime of large Péclet and small Damköhler numbers. MATLAB's BiCGStab was chosen as it supports the use of functions as a preconditioner, and has a larger user base than our modified implementation of BiGCStab. Furthermore, our modified implementation converges for all Péclet and Damköhler numbers without requiring a preconditioner.

These experiments use the same parameters as presented in Section 4.1, however with $M=256$. Several convergence plots similar to the ones shown in Fig. 4 are presented. The preconditioners used here are Jacobi, ILU(0) and a V-cycle geometric multigrid preconditioner with a depth of 4 and Gauss-Seidel smoothing in the direction of the advection [42, 95]. The convergence plots of which are shown in Fig. 8. The results show that the multigrid is most efficient in terms of matrix-vector products, followed closely by ILU(0). Both multigrid and ILU(0) converge especially quickly for strong advection, since in this limit the system matrix becomes approximately lower-triangular. The Jacobi preconditioner gives no significant speedup, if any.

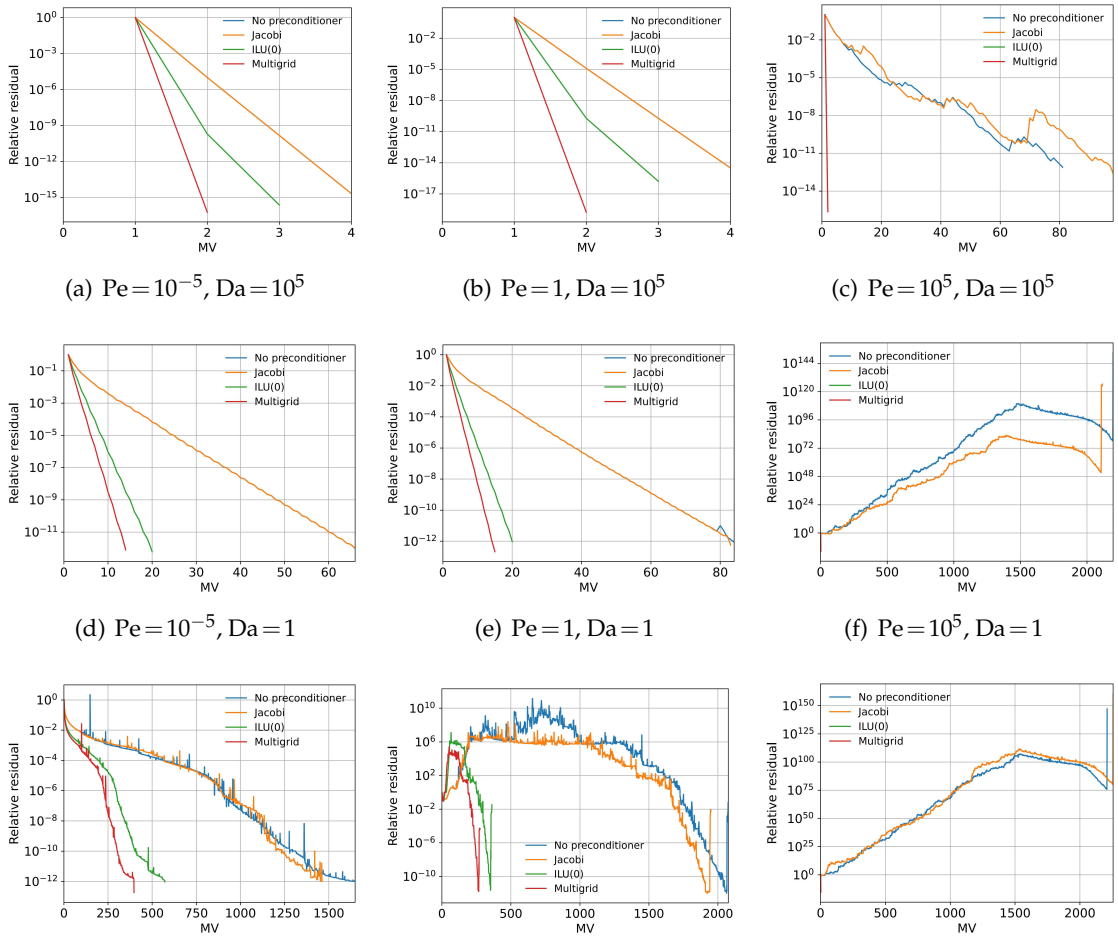
Note, however, that the issues with MATLAB's BiCGStab are not completely resolved by adding a preconditioner. As can be seen in Fig. 8(h), the issue related to the deviation between the estimated and the true residuals is still present, as can be seen by the sudden increase in residual after the estimated residual has reached the tolerance of 10^{-12}.

Even though a good preconditioner does significantly lower the required amount of matrix-vector products as expected, it does not provide a remedy for all convergence issues. Both a good preconditioner *and* a robust iterative method are required for efficient and reliable convergence.

5 Conclusions

As mentioned in the original reference of BiCGStab [14], there are many possible variants of BiCGStab which are all equivalent in exact arithmetic, but may have different behavior in finite precision arithmetic. In this paper we have reported on the reliability of different implementations of BiCGStab and IDR for advection-diffusion-reaction (ADR) problems with real eigenvalues of the discretization matrix.

First, a baseline BiCGStab implementation has been presented, for which it has been shown that even for small matrices this variant can exhibit complications in converging to a solution. Three modifications to the baseline BiCGStab algorithm have been implemented, i.e., the choice of shadow residual $\widetilde{\boldsymbol{r}}$, the reliable updating scheme and keeping the solution with the smallest residual thus far.



(a) $\mathrm{Pe}=10^{-5}$, $\mathrm{Da}=10^{5}$ (b) $\mathrm{Pe}=1$, $\mathrm{Da}=10^{5}$ (c) $\mathrm{Pe}=10^{5}$, $\mathrm{Da}=10^{5}$

(d) $\mathrm{Pe}=10^{-5}$, $\mathrm{Da}=1$ (e) $\mathrm{Pe}=1$, $\mathrm{Da}=1$ (f) $\mathrm{Pe}=10^{5}$, $\mathrm{Da}=1$

(g) $\mathrm{Pe}=10^{-5}$, $\mathrm{Da}=10^{-5}$ (h) $\mathrm{Pe}=1$, $\mathrm{Da}=10^{-5}$ (i) $\mathrm{Pe}=10^{5}$, $\mathrm{Da}=10^{-5}$

Figure 8: Comparison of the estimated residual as function of number of MV for MATLAB's BiCGStab with various preconditioners with $M=256$. For $\mathrm{Pe}=10^{5}$ ILU(0) and Multigrid both converge in less than 5 MV.

A model ADR problem has been set up and converted into dimensionless form. Using this dimensionless form the entire parameter space of Péclet and Damköhler numbers can be investigated. The ADR-equation has been discretized using the Finite Volume Method in combination with the exponential scheme of [6]. Unlike the central difference scheme, the exponential scheme yields a discretization matrix with *real* eigenvalues, even for strong advection.

Since the eigenvalues are real, there is no improvement in reliability of BiCGStab(2) compared to BiCGStab. MATLAB's implementations of BiCGStab and BiCGStab(2) performed no better than the baseline BiCGStab algorithm. IDR(4) performed very well for practically all Péclet and Damköhler numbers, clearly outperforming the modified

BiCGStab implementations, in terms of the number of matrix-vector products for problems with large Péclet number. For roughly $\mathrm{Pe} > 1$ and $\mathrm{Da} < \mathrm{Pe}$ MATLAB's solvers and the baseline BiCGStab method do not converge. Choosing a random shadow residual, $\widetilde{\boldsymbol{r}}$, is essential in the numerical experiments for the reliability of BiCGStab starting from a zero initial guess. It is hypothesized that this is due to the residual moving through the domain, eventually leading to $\langle \widetilde{\boldsymbol{r}}, \boldsymbol{r} \rangle \to 0$ and a breakdown of the BiCG part.

It is shown for matrices resulting from discretized ADR equations that the LMR method can be interpreted as a time integration method for the advection equation subject to homogeneous Dirichlet boundary conditions. This is explicitly shown for a one-dimensional advection equation in Section 2.4. However, since the investigated discretization matrices are positive definite implying $\omega > 0$, this time stepping property also holds for these problems. Furthermore, it is demonstrated that BiCGStab shows similar behavior in numerical experiments. It is suspected that the effect of the residual moving as a wave in the advection direction is related to the conjecture in [41], where it was argued that the first phase (initial plateau where the residual remains relatively constant) lasts as long as the longest streamline takes to traverse the grid with the flow for residual-minimizing Krylov subspace methods.

The reliable updating scheme included in BiCGStab adds an extra step to ensure the computed residual truly achieves the prescribed tolerance. As long as the solver converges to the tolerance required, there is no benefit in storing the best solution thus far. If the method stalls the effect is still not significant in our experiments.

We recommend to modify BiCGStab with a random shadow residual, in conjunction with a reliable updating scheme. Most notably if the initial residual is sparse, for example when it is only non-zero near the boundaries of the domain. Moreover IDR(4) is an excellent candidate, achieving the tolerance for a large number of cases. Additionally, for strong advection, IDR(4) uses significantly less matrix-vector products compared to BiCGStab.

In practice one cannot know beforehand which linear solver is optimal for a given problem. We have shown in this paper that even specific implementations of BiCGStab have a significant impact on convergence. For example, the seemingly arbitrary parameter $\widetilde{\boldsymbol{r}}$ has a major impact on the robustness of the solver in our experiments. This observation complicates the applicability of specific iterative methods even further. Nevertheless, we conclude, based on the numerical results, that modified BiCGStab and IDR(S) would be preferable over standard BiCGStab for ADR problems. A practical benefit of the modified BiCGStab variant presented here is that choosing a different $\widetilde{\boldsymbol{r}}$ is trivial to implement in existing codes and should be sufficient to largely mitigate issues relating to the vanishing of $\langle \widetilde{\boldsymbol{r}}, \widehat{\boldsymbol{r}}_{:,1} \rangle$.

In this paper, the analysis is only applied to the linear advection-diffusion-reaction equation. However, for nonlinear problems such as the nonlinear ADR equation and the Navier-Stokes equations a linear system is still obtained after linearization using Newton or Picard iteration. Alternatively, one may choose to linearize before discretizing, for example by linearizing the source term of the ADR equation which was considered in

this paper. We believe that even for nonlinear PDEs the results shown here can still be relevant.

Acknowledgments

This research is supported by the Netherlands Organisation for Scientific Research (NWO), which is partly funded by the Ministry of Economic Affairs, and The Netherlands eScience Center in the framework of the JCER program. The Netherlands eScience Center is funded by NWO and SURF.

Appendices

A Derivation LMR stepsize and residual

In this appendix we show that the stepsize $\omega = \frac{1}{2}$ under certain conditions; additionally we derive an expression for the k-th residual generated by the LMR algorithm for the problem given in Section 2.4.

First, the stepsize ω in the LMR algorithm is given by

$$\omega := \frac{\langle \boldsymbol{Ar}, \boldsymbol{r} \rangle}{\langle \boldsymbol{Ar}, \boldsymbol{Ar} \rangle}, \tag{A.1}$$

where the matrix $\boldsymbol{A}$ is the tridiagonal Toeplitz matrix

$$\boldsymbol{A} := \text{tridiag}(-1, 1, 0). \tag{A.2}$$

With these definitions, we will show that $\omega = \frac{1}{2}$ for any vector $\boldsymbol{r}$ with last element equal to zero given this specific matrix $\boldsymbol{A}$. We start by computing the numerator in (A.1);

$$\langle \boldsymbol{Ar}, \boldsymbol{r} \rangle = \sum_{i=1}^{N} (\boldsymbol{Ar})_i r_i = \sum_{i=1}^{N} \Big(\sum_{j=1}^{N} a_{ij} r_j \Big) r_i, \tag{A.3}$$

then by splitting off the first term of the sum over i, and substituting the values for the elements of $\boldsymbol{A}$ we arrive at

$$\sum_{j=1}^{N} a_{1j} r_j r_1 + \sum_{i=2}^{N} \Big(\sum_{j=1}^{N} a_{ij} r_j \Big) r_i = r_1^2 + \sum_{i=2}^{N} (-r_{i-1} + r_i) r_i. \tag{A.4}$$

The expression in (A.4) can be seen as the squared 2-norm of $\boldsymbol{r}$ with a rest term, since after expanding the brackets it is equal to

$$\sum_{i=1}^{N} r_i^2 - \sum_{i=2}^{N} r_{i-1} r_i = \|\boldsymbol{r}\|^2 - \sum_{i=1}^{N-1} r_i r_{i+1}. \tag{A.5}$$

The denominator in (A.1) is computed via a similar procedure;

$$\langle \boldsymbol{Ar}, \boldsymbol{Ar} \rangle = \sum_{i=1}^{N} (\boldsymbol{Ar})_i^2 = \sum_{i=1}^{N} \Big(\sum_{j=1}^{N} a_{ij} r_j \Big)^2, \tag{A.6}$$

after again splitting off the first term of the i-sum and substituting the values for $\boldsymbol{A}$ we obtain

$$\Big(\sum_{j=1}^{N} a_{1j} r_j \Big)^2 + \sum_{i=2}^{N} \Big(\sum_{j=1}^{N} a_{ij} r_j \Big)^2 = r_1^2 + \sum_{i=2}^{N} (-r_{i-1} + r_i)^2. \tag{A.7}$$

Then, expanding the squared term in the summation and relabeling the index, the denominator becomes

$$r_1^2 + \sum_{i=2}^{N} (r_i^2 + r_{i-1}^2 - 2 r_{i-1} r_i) = \|\boldsymbol{r}\|^2 + \sum_{i=2}^{N} r_{i-1}^2 - 2 \sum_{i=2}^{N} r_{i-1} r_i = 2\|\boldsymbol{r}\|^2 - r_N^2 - 2 \sum_{i=1}^{N-1} r_i r_{i+1}. \tag{A.8}$$

By dividing (A.5) and (A.8) it is clear that

$$\omega = \tfrac{1}{2}, \quad \text{if} \quad r_N = 0. \tag{A.9}$$

Second, we derive an expression for the k-th residual produced by LMR starting from an initial residual of $\boldsymbol{e}_1$. The LMR recurrence for the residual, given $\omega = \frac{1}{2}$, is the following

$$\boldsymbol{r}^{k+1} = (\boldsymbol{I} - \omega \boldsymbol{A}) \boldsymbol{r}^k = (\boldsymbol{I} - \tfrac{1}{2} \boldsymbol{A}) \boldsymbol{r}^k. \tag{A.10}$$

Next, the matrix $\boldsymbol{B}$ is introduced as

$$\boldsymbol{B} := (\boldsymbol{I} - \tfrac{1}{2} \boldsymbol{A}) = \tfrac{1}{2} \text{tridiag}(1,1,0). \tag{A.11}$$

Note that the matrix $\boldsymbol{B}$ can be interpreted as a smoothing matrix, taking two neighbouring elements of a vector and computing the average value. We then define

$$r_0^k := 0 \quad \text{for all} \quad k, \tag{A.12}$$

and a special case of the binomial coefficients as

$$\binom{0}{0} := 1. \tag{A.13}$$

Then for any vector $\boldsymbol{r}^k$ the LMR recurrence (A.10) gives the recurrence

$$r_i^{k+1} = \tfrac{1}{2} (r_i^k + r_{i-1}^k), \quad i = 1,2,\cdots,N. \tag{A.14}$$

We will prove that the k-th residual

$$r_i^k = \binom{k}{i-1} (\tfrac{1}{2})^k, \quad i = 1,2,\cdots,k+1 < N, \tag{A.15}$$

with the other elements given by

$$r_i^k = 0, \quad i > k+1. \tag{A.16}$$

To prove (A.15) holds for any k, we use a proof by induction; we start by showing the base case $k=0$, then for a given k we derive the case for $k+1$.

Starting from the first residual vector $\boldsymbol{r}^0 = \boldsymbol{e}_1$;

$$\boldsymbol{r}_1^0 = 1, \quad r_i^0 = 0, \quad i > 1. \tag{A.17}$$

The LMR recurrence yields the next residual as

$$\boldsymbol{r}^1 = \boldsymbol{B}\boldsymbol{e}_1 = \frac{1}{2}\begin{bmatrix}1 & 1 & 0 & \dots & 0\end{bmatrix}^T, \tag{A.18}$$

which agrees with (A.15). Next, we have to show that for any k,i

$$r_i^{k+1} = \binom{k+1}{i-1}(\tfrac{1}{2})^{k+1}, \quad i = 1,2,\cdots,k+2 < N. \tag{A.19}$$

Substituting r_i^k from (A.15) into the recurrence (A.14) yields

$$r_i^{k+1} = \tfrac{1}{2}(r_i^k + r_{i-1}^k) = \tfrac{1}{2}\left[\binom{k}{i-1}(\tfrac{1}{2})^k + \binom{k}{i-2}(\tfrac{1}{2})^k\right], \quad i = 1,2,\cdots,k+2 < N. \tag{A.20}$$

This can be rewritten to

$$r_i^{k+1} = \left[\binom{k}{i-1} + \binom{k}{i-2}\right](\tfrac{1}{2})^k = \binom{k+1}{i-1}(\tfrac{1}{2})^{k+1}. \tag{A.21}$$

Since (A.21) is equal to (A.15) after relabeling the index, it follows by induction that for any k Eq. (A.15) satisfies the relation (A.10) completing the proof.

B Positive definite discretization matrices

A positive definite (PD) matrix has the property that $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} > 0$ for all non-zero vectors $\boldsymbol{x}$ [43, p. 140]. First, we start by showing if $\boldsymbol{x}^T(\boldsymbol{A}^T + \boldsymbol{A})\boldsymbol{x} > 0$ then $\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} > 0$. Second, we show that the discretization matrices obtained for the one-dimensional ADR problem are PD. Finally, we show that the two-dimensional and three-dimensional ADR problems also yield PD discretization matrices.

First, note that

$$\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} = (\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x})^T = \boldsymbol{x}^T\boldsymbol{A}^T\boldsymbol{x}, \tag{B.1}$$

and consequently

$$\boldsymbol{x}^T(\boldsymbol{A} + \boldsymbol{A}^T)\boldsymbol{x} = 2\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}, \tag{B.2}$$

therefore if

$$\boldsymbol{x}^T(\boldsymbol{A}+\boldsymbol{A}^T)\boldsymbol{x}>0, \quad \text{then,} \quad \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}>0. \tag{B.3}$$

Second, we show that $\boldsymbol{A}_x+\boldsymbol{A}_x^T$ with $\boldsymbol{A}_x$ given by (B.4) later, is PD. To do this, note that the matrix $\boldsymbol{A}_x+\boldsymbol{A}_x^T$ is symmetric, therefore if all eigenvalues are positive then this matrix is PD [39, p.246]. The matrix $\boldsymbol{A}_x+\boldsymbol{A}_x^T$ is given by

$$\boldsymbol{A}_x+\boldsymbol{A}_x^T=\frac{1}{h}\text{tridiag}(-\text{B}(\text{Pe})-\text{B}(-\text{Pe}),2\text{B}(-\text{Pe})+2\text{B}(\text{Pe}),-\text{B}(-\text{Pe})-\text{B}(\text{Pe})), \tag{B.4}$$

which can be written as

$$\boldsymbol{A}_x+\boldsymbol{A}_x^T=\frac{1}{h}(\text{B}(\text{Pe})+\text{B}(-\text{Pe}))\text{tridiag}(-1,2,-1). \tag{B.5}$$

Since the eigenvalues of $\text{tridiag}(-1,2,-1)$ are positive for any number of grid points, $\boldsymbol{A}_x+\boldsymbol{A}_x^T$ is PD. It can be shown in a similar fashion that also for non-zero Damköhler numbers discretization matrix is PD, since the addition of a non-zero Damköhler number term only shifts all eigenvalues toward the right on the real axis.

To extend this derivation to the two-dimensional ADR problems, note that

$$(\boldsymbol{A}_x\oplus\boldsymbol{A}_y)^T=(\boldsymbol{I}\otimes\boldsymbol{A}_x+\boldsymbol{A}_y\otimes\boldsymbol{I})^T=(\boldsymbol{I}\otimes\boldsymbol{A}_x)^T+(\boldsymbol{A}_y\otimes\boldsymbol{I})^T, \tag{B.6}$$

and using the property $(\boldsymbol{I}\otimes\boldsymbol{A})^T=\boldsymbol{I}\otimes\boldsymbol{A}^T$ [44, p. 40] it follows that

$$(\boldsymbol{A}_x\oplus\boldsymbol{A}_y)+(\boldsymbol{A}_x\oplus\boldsymbol{A}_y)^T=\boldsymbol{I}\otimes(\boldsymbol{A}_x+\boldsymbol{A}_x^T)+(\boldsymbol{A}_y+\boldsymbol{A}_y^T)\otimes\boldsymbol{I}. \tag{B.7}$$

By using corollary 4.2.13 of [39, p.246] it is known that if $(\boldsymbol{A}_x+\boldsymbol{A}_x^T)$ is symmetric positive definite (SPD) then $\boldsymbol{I}\otimes(\boldsymbol{A}_x+\boldsymbol{A}_x^T)$ is also SPD. Via a similar argument it follows that $(\boldsymbol{A}_y+\boldsymbol{A}_y^T)\otimes\boldsymbol{I}$ is also SPD. Since the element-wise sum of two SPD matrices yields another SPD matrix, we conclude that $(\boldsymbol{A}_x\oplus\boldsymbol{A}_y)$ is PD.

The matrices obtained from the three-dimensional discretization of the ADR equation with the exponential scheme can be shown to be PD in a similar way.

References

[1] L. Martinez, A. Dhruv, L. Lin, E. Balaras, M. Keidar, Interaction between a helium atmospheric plasma jet and targets and dynamics of the interface, Plasma Sources Science and Technology, 28 (2019):115002.

[2] D.A. Knoll, P.R. McHugh, Newton-Krylov methods applied to a system of convection-diffusion-reaction equations, Computer Physics Communications, 88 (1995):141–160.

[3] J. Lin, S. Reutskiy, C. Chen, J. Lu, A novel method for solving time-dependent 2D advection-diffusion-reaction equations to model transfer in nonlinear anisotropic media, Communications in Computational Physics, 26 (2019):233–264.

[4] B. Fischer, A. Ramage, D.J. Silvester, A.J. Wathen, On parameter choice and iterative convergence for stabilised discretisations of advection-diffusion problems, Computer Methods in Applied Mechanics and Engineering, 179 (1999):179–195.

[5] A. Singh, S. Das, S.H. Ong, H. Jafari, Numerical solution of nonlinear reaction-advection-diffusion equation, Journal of Computational and Nonlinear Dynamics, 14 (2019).
[6] S. Patankar, Numerical Heat Transfer and Fluid Flow, Taylor & Francis, 2018.
[7] D.L. Scharfetter, H.K. Gummel, Large-signal analysis of a silicon read diode oscillator, IEEE Transactions on Electron Devices, 16 (1969):64–77.
[8] Y. Saad, Iterative Methods for Sparse Linear Systems: Second Edition, SIAM, 2003.
[9] H.A. van der Vorst, Iterative Krylov Methods for Large Linear Systems, volume 13, Cambridge University Press, 2003.
[10] J.R. Shewchuk, An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, Technical report, Carnegie Mellon University, 1994.
[11] V. Faber, T. Manteuffel, Necessary and sufficient conditions for the existence of a conjugate gradient method, SIAM Journal on Numerical Analysis, 21 (1984):352–362.
[12] J. Liesen, Z. Strakoˇs, On optimal short recurrences for generating orthogonal Krylov subspace bases, SIAM Review, 50 (2008):485–503.
[13] C. Lanczos, Solution of systems of linear equations by minimized iterations, J. Res. Nat. Bur. Standards, 49 (1952):33–53.
[14] H.A. van der Vorst, Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing, 13 (1992):631–644.
[15] MATLAB, R2019a, The MathWorks Inc., 2019.
[16] G. Guennebaud, B. Jacob, et al., Eigen, URl: http://eigen.tuxfamily.org, (2010).
[17] J. Van Dijk, K. Peerenboom, M. Jimenez, D. Mihailova, J. Van der Mullen, The plasma modelling toolkit PLASIMO, Journal of Physics D: Applied Physics, 42 (2009):194012.
[18] COMSOL Multiphysics v. 5.6. www.comsol.com. COMSOL AB, Stockholm, Sweden.
[19] G.L.G. Sleijpen, H.A. Van der Vorst, Optimal iteration methods for large linear systems of equations, Notes on Numerical Fluid Mechanics, 45 (1993):291–291.
[20] Y. Saad, M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM Journal on Scientific and Statistical Computing, 7 (1986):856–869.
[21] G.L.G. Sleijpen, H.A. van der Vorst, D.R. Fokkema, BiCGstab (ell) for linear equations involving unsymmetric matrices with complex spectrum, Electronic Transactions on Numerical Analysis, 1 (1993):11–32.
[22] P. Sonneveld, M.B. van Gijzen, IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations, SIAM Journal on Scientific Computing, 31 (2008):1035–1062.
[23] G.L.G. Sleijpen, M.B. van Gijzen, Exploiting BiCGstab (l) strategies to induce dimension reduction, SIAM Journal on Scientific Computing, 32 (2010):2687–2709.
[24] G.L.G. Sleijpen, H.A. van der Vorst, D.R. Fokkema, BiCGstab (l) and other hybrid Bi-CG methods, Numerical Algorithms, 7 (1994):75–109.
[25] K. Aihara, K. Abe, E. Ishiwata, A variant of IDRstab with reliable update strategies for solving sparse linear systems, Journal of Computational and Applied Mathematics, 259 (2014):244–258.
[26] L.T. Yang, R.P. Brent, The improved BiCGStab method for large and sparse unsymmetric linear systems on parallel distributed memory architectures, in Fifth International Conference on Algorithms and Architectures for Parallel Processing, 2002. Proceedings, IEEE, 2002 324–328.
[27] S. Cools, W. Vanroose, The communication-hiding pipelined BiCGStab method for the par-

allel solution of large unsymmetric linear systems, Parallel Computing, 65 (2017):1–20.
[28] A. El Guennouni, K. Jbilou, H. Sadok, A block version of BiCGSTAB for linear systems with multiple right-hand sides, Electronic Transactions on Numerical Analysis, 16 (2003):2.
[29] K. Ahuja, P. Benner, E. de Sturler, L. Feng, Recycling BiCGSTAB with an application to parametric model order reduction, SIAM Journal on Scientific Computing, 37 (2015):429–446.
[30] D.R. Fokkema, Enhanced Implementation of BiCGstab (l) for Solving Linear Systems of Equations, 1996.
[31] G.L.G. Sleijpen and H.A. van der Vorst, Reliable updated residuals in hybrid Bi-CG methods, Computing, 56 (1996):141–163.
[32] J. Liesen, P. Tichỳ, Convergence analysis of Krylov subspace methods, GAMM-Mitteilungen, 27 (2004):153–173.
[33] W. Joubert, Lanczos methods for the solution of nonsymmetric systems of linear equations, SIAM Journal on Matrix Analysis and Applications, 13 (1992):926–943.
[34] Y. Yasuda. S. Sakamoto, Y. Kosaka, T. Sakuma, N. Okamoto, T. Oshima, Numerical analysis of large-scale sound fields using iterative methods part I: Application of Krylov subspace methods to boundary element analysis, Journal of Computational Acoustics, 15 (2007):449–471.
[35] N. Okamoto, R. Tomiku, T. Otsuru, Y. Yasuda, Numerical analysis of large-scale sound fields using iterative methods part II: Application of Krylov subspace methods to finite element analysis, Journal of Computational Acoustics, 15 (2007):473–493.
[36] J. Spanier, K.B. Oldham, An Atlas of Functions, Hemisphere Publishing Corporation New York, 1987.
[37] M. Abramowitz and I.A. Stegun, Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, volume 55, US Government printing office, 1964.
[38] S. Noschese, L. Pasquini, L. Reichel, Tridiagonal Toeplitz matrices: Properties and novel applications, Numerical Linear Algebra with Applications, 20 (2013):302–326.
[39] R.A. Horn, C.R. Johnson, Topics in Matrix Analysis, 1991, Cambridge University Presss, Cambridge, 37 (1991):39.
[40] V. Khoromskaia, B.N. Khoromskij, F. Otto, Numerical study in stochastic homogenization for elliptic partial differential equations: Convergence rate in the size of representative volume elements, Numerical Linear Algebra with Applications, 27 (2020):2296.
[41] O.G. Ernst, Residual-minimizing Krylov subspace methods for stabilized discretizations of convection-diffusion equations, SIAM Journal on Matrix Analysis and Applications, 21 (2000):1079–1101.
[42] H.C. Elman, D.J. Silvester and A.J. Wathen, Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics, Oxford University Press, USA, 2014.
[43] C.F. van Loan, G.H. Golub, Matrix Computations, Johns Hopkins University Press, 1983.
[44] F. Hiai, D. Petz, Introduction to Matrix Analysis and Applications, Springer Science & Business Media, 2014.