Inexact inner–outer Golub–Kahan bidiagonalization method: A relaxation strategy

Vincent Darrigrand, Andrei Dumitrasc, Carola Kruse, Ulrich Rüde

Inexact inner-outer Golub-Kahan bidiagonalization method: A relaxation strategy

Vincent Darrigrand [*1], Andrei Dumitrasc [†2], Carola Kruse[3], and Ulrich Ruede[2]

[1]IRIT, CNRS, Toulouse, France
[2]Chair for Computer Science 10 - System Simulation, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany
[3]Cerfacs, Toulouse, France

August 3, 2022

Abstract

We study an inexact inner-outer generalized Golub-Kahan algorithm for the solution of saddle-point problems with a two-times-two block structure. In each outer iteration, an inner system has to be solved which in theory has to be done exactly. Whenever the system is getting large, an inner exact solver is, however, no longer efficient or even feasible and iterative methods must be used. We focus this article on a numerical study showing the influence of the accuracy of an inner iterative solution on the accuracy of the solution of the block system. Emphasis is further given on reducing the computational cost, which is defined as the total number of inner iterations. We develop relaxation techniques intended to dynamically change the inner tolerance for each outer iteration to further minimize the total number of inner iterations. We illustrate our findings on a Stokes problem and validate them on a mixed formulation of the Poisson problem.

1 Introduction

Saddle-point systems can be found in a variety of application fields, such as, for example, mixed finite element methods in fluid dynamics or interior point methods in optimization. An extensive overview about application fields and

[*]EU Horizon 2020 Project Energy oriented Center of Excellence: toward exascale for energy (EoCoE-II), Project ID: 824158
[†]Bavarian Academic Center for Central, Eastern and Southeastern Europe (BAYHOST)

solution methods for this kind of problems is presented in the well-known article [6] by Benzi, Golub and Liesen. In our following study, we want to focus on an iterative solver based on the Golub-Kahan bidiagonalization: the generalized Golub-Kahan bidiagonalization (GKB) algorithm. This solver is designed for saddle-point systems, and was introduced by Arioli[1]. It belongs to the family of Krylov subspace methods and, as such, relies on specific orthogonality conditions, as we will review in more detail in Section 2. Enforcing these orthogonality conditions requires solving an *inner problem*, i.e. formally computing products with matrix inverses (as described in Algorithm 1). In practice, this computation is performed with a linear system solver. For this task, we will explore in this article the use of iterative methods to serve as replacement for direct methods that have been used within GKB so far. This is essential for very large problems, such as those coming from a discretized Partial Differential Equation (PDE) in 2D or 3D, when direct solvers may reach their limits.

Using an inner iterative solver might also be advantageous from another point of view as we motivate in the following. The solution of large linear systems is often the bottleneck in scientific computing. The computational cost and, consequently, the execution time and/or the energy consumption can become prohibitive. For the inner-outer iterative GKB solver in turn, the principal and costliest part is the solution of the inner system at each outer iteration. One approximate metric to measure the cost of the GKB solver is the aggregate sum of the number of inner iterations. For a given setup, the cost of the GKB method can hence be optimized by executing only a minimal number of inner iterations necessary for achieving a prescribed accuracy of the solution. To reduce this number, there are two possible steps to be taken into account. In a first step, for a given application it is often unnecessary to solve the linear system with the highest achievable accuracy. This could be the case, for example, in the solution of a discretized PDE, when the discretization already introduces an error. A precise solution of the linear system would not improve the numerical solution with respect to the analytic solution of the PDE any further than the discretization allows. Next, we come to the second step which will be the main point of the study in this paper. The solution of the inner linear system in the GKB method has to be exact, in theory. If we choose a rather low accuracy for the outer iterative solver, an inner exact solution might, however, no longer be necessary, as long as the inner error does not alter the chosen accuracy of the numerical solution. This strategy results in a further reduction of the number of inner iterations, since the inner solver will converge in fewer iterations when a less strict stopping tolerance is used.

In the following study, we address the case where the inner solver has a prescribed stopping tolerance and then how this limited accuracy affects the outer process and the quality of its iterates. We will show that, with the appropriate choice of parameters, it is possible to make use of inner iterative solvers without compromising the accuracy of the GKB result. As it can be seen immediately, the lower the accuracy for the inner solver, the less expensive the GKB method will be. Furthermore, we take advantage of the versatility of iterative methods by adapting the stopping tolerance of the inner solver dynamically. In other

words, we prescribe the tolerance of the inner solver according to some criteria determined at each outer iteration. This can lead to a reduction of the cost, since only a minimal number of inner iterations are executed. Typically, we will reduce the required accuracy for later instances of the inner solver, since later steps of the outer GKB-iteration may contribute less to the overall accuracy.

One particular advantage of our proposed method is its generality. The strategy is independent of other choices which are problem-specific, such as the preconditioner for a Krylov method. We perform most of our tests on a relatively small Stokes flow problem, to illustrate the salient features. We confirm our findings by one final test on a larger case of the mixed Poisson problem, including the use of the augmented Lagrangian method, to demonstrate the use in a realistic scenario.

Our study has a similar context as other works on inexact Krylov methods [8, 7], where these algorithms have been investigated from a numerical perspective. In these articles, the inexactness originates from a limited accuracy of the matrix-vector multiplication or that of the solution of a local sub-problem. Similar to what we have described above, it was found that the inner accuracy can be varied from step to step while still achieving convergence of the outer method. It was shown experimentally that the initial tolerance should be strict, then relaxed gradually, with the change being guided by the latest residual norm. Other works complemented the findings with theoretical insights, relevant to several algorithms of the Krylov family [27, 28, 30]. It was noted that, in some cases, unless a problem-dependent constant is included, the outer solver may fail to converge if the accuracy of the inner solution is adapted only based on the residual norm. This constant can be computed based on extreme singular values, as shown by Simoncini and Szyld [27]. Another source of inexactness can be the application of a preconditioner via an iterative method. Van den Eshof, Sleijpen and van Gijzen considered inexactness in Krylov methods originating both from matrix-vector products and variable preconditioning, using iterative methods from the GMRES family [31]. Similarly to earlier work, their analysis relies on the connection between the residual and the accuracy of the solution to the inner problem. Since applying the preconditioner has the same effect as a matrix-vector product, the same strategies can be applied to more complex, flexible algorithms, such as those involving variable preconditioning: FGMRES [26], GMRESR [32], etc. A flexible version of the Golub-Kahan bidiagonalization is employed by Chung and Gazzola to find regularized solutions to a problem of image deblurring [9]. In a more recent paper with the same application, Gazzola and Landman develop inexact Krylov methods as a way to deal with approximate knowledge of $\mathbf{A}$ and $\mathbf{A}^T$ [16]. Erlangga and Nabben construct a framework including nested Krylov solvers. They develop a multilevel approach to shift small eigenvalues, leading to a faster convergence of the linear solver [15]. In subsequent work related to multilevel Krylov methods, Kehl, Nabben and Szyld apply preconditioning in a flexible way, via an adaptive number of inner iterations [19]. Baumann and van Gijzen analyze solving shifted linear systems and, by applying flexible preconditioning, also develop nested Krylov solvers [5]. McInnes et al. consider hierarchical and nested Krylov methods

with a small number of vector inner products, with the goal of reducing the need for global synchronization in a parallel computing setting [23].

Other than solving linear systems, inexact Krylov methods have been studied when tackling eigenvalue problems, as in the paper by Golub, Zhang and Zha [17]. Although using different arguments, it was shown that the strategy of increasing the inner tolerance is successful for this kind of problem as well. Xu and Xue make use of an inexact rational Krylov method to solve nonsymmetric eigenvalue problems and observe that the accuracy of the inner solver (GMRES) can be relaxed in later outer steps, depending on the value of the eigenresidual [33]. Dax computes the smallest eigenvalues of a matrix via a restarted Krylov solver which includes inexact matrix inversion [10].

Our paper is structured as follows: in Section 2, we review the theory and properties of the GKB algorithm; in Section 3, we describe the specific problem we chose to use as test case for the numerical experiments; Section 4 is meant to illustrate the interactions between the accuracy of the inner solver and that of the outer one in a numerical test setting; Section 5 describes the link between the error of the outer solver and the perturbation induced by the use of an iterative inner solver. We describe and test our proposed strategy of using a variable tolerance parameter for the inner solver in Section 6. We explore the interaction between the method of the Augmented Lagrangian (AL) and our strategy in Section 7. The final section is devoted to concluding remarks.

2 Generalized Golub-Kahan algorithm

We are interested in saddle-point problems of the form

$$\begin{bmatrix} \mathbf{M} & \mathbf{A} \\ \mathbf{A}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \mathbf{r} \end{bmatrix} \tag{1}$$

with $\mathbf{M} \in \mathbb{R}^{m\times m}$ being a symmetric positive definite matrix and $\mathbf{A} \in \mathbb{R}^{m\times n}$ a full rank constraint matrix. The generalized GKB algorithm for the solution of a class of saddle-point systems was introduced by Arioli [1]. To apply it to the system (1), we first need to have the upper block of the right-hand side to be equal to 0. To this end, we use the transformation

$$\mathbf{u} = \mathbf{w} - \mathbf{M}^{-1}\mathbf{g}, \tag{2}$$

$$\mathbf{b} = \mathbf{r} - \mathbf{A}^T\mathbf{u}. \tag{3}$$

The resulting system is

$$\begin{bmatrix} \mathbf{M} & \mathbf{A} \\ \mathbf{A}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix}, \tag{4}$$

which is equivalent to that in Equation (1). We can recover the $\mathbf{w}$ variable as $\mathbf{w} = \mathbf{u} + \mathbf{M}^{-1}\mathbf{g}$.

Let $\mathbf{N} \in \mathbb{R}^{n \times n}$ be a symmetric positive definite matrix. To properly describe the GKB algorithm, we need to define the following norms

$$\|\mathbf{v}\|_{\mathbf{M}} = \sqrt{\mathbf{v}^T \mathbf{M} \mathbf{v}}; \qquad \|\mathbf{q}\|_{\mathbf{N}} = \sqrt{\mathbf{q}^T \mathbf{N} \mathbf{q}}; \qquad \|\mathbf{y}\|_{\mathbf{N}^{-1}} = \sqrt{\mathbf{y}^T \mathbf{N}^{-1} \mathbf{y}}. \tag{5}$$

Given the right-hand side vector $\mathbf{b} \in \mathbb{R}^n$, the first step of the bidiagonalization is

$$\beta_1 = \|\mathbf{b}\|_{\mathbf{N}^{-1}}, \quad \mathbf{q}_1 = \mathbf{N}^{-1}\mathbf{b}/\beta_1. \tag{6}$$

After k iterations, the partial bidiagonalization is given by

$$\begin{cases} \mathbf{A}\mathbf{Q}_k = \mathbf{M}\mathbf{V}_k\mathbf{B}_k, & \mathbf{V}_k^T\mathbf{M}\mathbf{V}_k = \mathbf{I}_k \\ \mathbf{A}^T\mathbf{V}_k = \mathbf{N}\mathbf{Q}_k\mathbf{B}_k^T + \beta_{k+1}\mathbf{q}_{k+1}\mathbf{e}_k^T, & \mathbf{Q}_k^T\mathbf{N}\mathbf{Q}_k = \mathbf{I}_k \end{cases}, \tag{7}$$

with the bidiagonal matrix

$$\mathbf{B}_k = \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \dots & 0 \\ 0 & \alpha_2 & \beta_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & \alpha_{k-1} & \beta_k \\ 0 & \dots & 0 & 0 & \alpha_k \end{bmatrix} \tag{8}$$

and the residual term $\beta_{k+1}\mathbf{q}_{k+1}\mathbf{e}_k^T$. The columns of $\mathbf{V}_k$ are orthonormal vectors with respect to the inner product and norm induced by $\mathbf{M}$, while the same holds for $\mathbf{Q}_k$ and $\mathbf{N}$ respectively

$$\begin{aligned} \mathbf{v}_i^T\mathbf{M}\mathbf{v}_j = 0, \forall i \neq j; \qquad \|\mathbf{v}_k\|_{\mathbf{M}} = 1; \\ \mathbf{q}_i^T\mathbf{N}\mathbf{q}_j = 0, \forall i \neq j; \qquad \|\mathbf{q}_k\|_{\mathbf{N}} = 1. \end{aligned} \tag{9}$$

Prior to the normalization leading to $\mathbf{v}_k$ and $\mathbf{q}_k$, the norms are stored as α_k for $\mathbf{v}_k$ and β_k for $\mathbf{q}_k$, as detailed in algorithm 1. Using $\mathbf{V}_k$, $\mathbf{Q}_k$ and the relations in Equation (7), we can transform the system from Equation (4) into a simpler form

$$\begin{bmatrix} \mathbf{I}_k & \mathbf{B}_k \\ \mathbf{B}_k^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{z}_k \\ \mathbf{y}_k \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{Q}_k^T\mathbf{b} \end{bmatrix}. \tag{10}$$

With the choice for $\mathbf{q}_1$ given in Equation (6), we have that $\mathbf{Q}_k^T\mathbf{b} = \beta_1\mathbf{e}_1$. The solution components to Equation (10) are then given by

$$\mathbf{z}_k = \beta_1\mathbf{B}_k^{-T}\mathbf{e}_1; \quad \mathbf{y}_k = -\mathbf{B}_k^{-1}\mathbf{z}_k, \tag{11}$$

where $\mathbf{B}_k^{-T}$ is the inverse of $\mathbf{B}_k^T$. We can build the k-th approximate solution to Equation (4) as

$$\mathbf{u}_k = \mathbf{V}_k\mathbf{z}_k; \quad \mathbf{p}_k = \mathbf{Q}_k\mathbf{y}_k. \tag{12}$$

In particular, after a number of $k = n$ steps and assuming exact arithmetic, we have $\mathbf{u}_k = \mathbf{u}$ and $\mathbf{p}_k = \mathbf{p}$, meaning we have found the exact solution to

Equation (4). A proof of why n terms are sufficient to find the exact solution is given in the introductory paper by Arioli [1]. This corresponds to a scenario where it is necessary to perform the n iterations, although, for specific problems with particular features, the solution may be found after fewer steps. As $k \to n$, the quality of the approximation improves ($\mathbf{u}_k \to \mathbf{u}$ and $\mathbf{p}_k \to \mathbf{p}$), with the bidiagonalization residual $\beta_{k+1}\mathbf{q}_{k+1}\mathbf{e}_k^T$ vanishing for $k = n$.

Given the structure of $\beta_1\mathbf{e}_1$ and $\mathbf{B}^T$, we find

$$\zeta_1 = \frac{\beta_1}{\alpha_1}, \quad \zeta_k = \zeta_{k-1}\frac{\beta_k}{\alpha_k}, \quad \mathbf{z}_k = \begin{bmatrix} \mathbf{z}_{k-1} \\ \zeta_k \end{bmatrix} \tag{13}$$

in a recursive manner. Then, $\mathbf{u}_k$ is computed as $\mathbf{u}_k = \mathbf{u}_{k-1} + \zeta_k\mathbf{v}_k$. In order to obtain a recursive formula for $\mathbf{p}$ as well, we introduce the vector

$$\mathbf{d}_k = \frac{\mathbf{q}_k - \beta_k\mathbf{d}_{k-1}}{\alpha_k}, \quad \mathbf{d}_1 = \frac{\mathbf{q}_1}{\alpha_1}. \tag{14}$$

Finally, the update formulas are

$$\mathbf{u}_k = \mathbf{u}_{k-1} + \zeta_k\mathbf{v}_k, \quad \mathbf{p}_k = \mathbf{p}_{k-1} - \zeta_k\mathbf{d}_k. \tag{15}$$

At step k of Algorithm 1, we have the following error in the energy norm.

$$\begin{aligned} \|\mathbf{e}_k\|_{\mathbf{M}}^2 = \|\mathbf{u}_k - \mathbf{u}\|_{\mathbf{M}}^2 &= \left\| \mathbf{V}_k\mathbf{z}_k - [\mathbf{V}_k\mathbf{V}_{n-k}] \begin{bmatrix} \mathbf{z}_k \\ \mathbf{z}_{n-k} \end{bmatrix} \right\|_{\mathbf{M}}^2 \\ &= \|\mathbf{V}_{n-k}\mathbf{z}_{n-k}\|_{\mathbf{M}}^2 = \|\mathbf{z}_{n-k}\|_2^2 = \sum_{i=k+1}^{n} \zeta_i^2 \end{aligned} \tag{16}$$

In the last line, we have made use of the $\mathbf{M}$-orthonormality of the $\mathbf{V}$ matrices. If we truncate the sum above to only its first d terms, we get a lower bound on the energy norm of the error. The subscript d stands for *delay*, because we can compute this lower bound corresponding to a given step k only after an additional d steps

$$\xi_{k,d}^2 = \sum_{i=k+1}^{k+d+1} \zeta_i^2 < \|\mathbf{e}_k\|_{\mathbf{M}}^2 . \tag{17}$$

With this bound for the absolute error, we can devise one for the relative error in Equation (18), which is then used as stopping criterion in Line 15 of Algorithm 1.

$$\bar{\xi}_{k,d}^2 = \frac{\sum_{i=k-d+1}^{k} \zeta_i^2}{\sum_{i=1}^{k} \zeta_i^2}. \tag{18}$$

The GKB algorithm has the following error minimization property. Let $\mathcal{V}_k = span\{\mathbf{v}_1, ..., \mathbf{v}_k\}$ and $\mathcal{Q}_k = span\{\mathbf{q}_1, ..., \mathbf{q}_k\}$. Then, for any arbitrary step k, we have that

$$\min_{\substack{\mathbf{u}_k \in \mathcal{V}_k, \\ (\mathbf{A}^T\mathbf{u}_k - \mathbf{b}) \perp \mathcal{Q}_k}} \|\mathbf{u} - \mathbf{u}_k\|_{\mathbf{M}} \tag{19}$$

is met for $\mathbf{u}_k$ as computed by Algorithm 1.

For brevity and because the GKB algorithm features this minimization property for the primal variable, our presentation will focus on the velocity for Stokes problems. The stopping criteria for our proposed algorithmic strategies rely on approximations of the velocity error norm. For all the numerical experiments that we have performed, the pressure error norm is close to that of the velocity (less than an order of magnitude apart). In the cases where we operate on a different subspace, as a result of preconditioning, we find that the pressure error norm is actually smaller than that for the velocity. In the case where the dual variable is equally important as the primal, one can use a monolithic approach, such as applying MINRES to the complete saddle-point system.

The GKB (as implemented by Algorithm 1) is a nested iterative scheme in which each outer loop involves solving an inner linear system. According to the theory given in the paper by Arioli [1], the matrices $\mathbf{M}$ and $\mathbf{N}$ have to be inverted exactly in each iteration. We can choose $\mathbf{N} = \frac{1}{\eta}\mathbf{I}$, whose inversion reduces to a scalar multiplication. In the following sections, unless otherwise specified, we consider $\eta = 1$. On the other hand, the matrix $\mathbf{M}$ depends on the underlying differential equations or the problem setting in general. As long as the matrix $\mathbf{M}$ is of moderate size, a robust direct solver can be used. For large problems, however, a direct solution might no longer be possible and an iterative solver will be required. At this point, we face two problems. First, depending on the application, inverting $\mathbf{M}$ might be more or less costly. Second, to achieve a solution quality close to machine precision, an iterative solver might require a considerable number of iteration steps.

Algorithm 1 Golub-Kahan bidiagonalization algorithm

Require: $\mathbf{M}, \mathbf{A}, \mathbf{N}, \mathbf{b}$, maxit

$\beta_1 = \|\mathbf{b}\|_{\mathbf{N}^{-1}}$; $\mathbf{q}_1 = \mathbf{N}^{-1}\mathbf{b}/\beta_1$

$\mathbf{w} = \mathbf{M}^{-1}\mathbf{A}\mathbf{q}_1$; $\alpha_1 = \|\mathbf{w}\|_{\mathbf{M}}$; $\mathbf{v}_1 = \mathbf{w}/\alpha_1$

$\zeta_1 = \beta_1/\alpha_1$; $\mathbf{d}_1 = \mathbf{q}_1/\alpha_1$; $\mathbf{u}^{(1)} = \zeta_1\mathbf{v}_1$; $\mathbf{p}^{(1)} = -\zeta_1\mathbf{d}_1$;

$\bar{\xi}_{1,d} = 1$; $k = 1$;

while $\bar{\xi}_{k,d} >$ tolerance and $k <$ maxit **do**

  $\mathbf{g} = \mathbf{N}^{-1}\left(\mathbf{A}^T\mathbf{v}_k - \alpha_k\mathbf{N}\mathbf{q}_k\right)$; $\beta_{k+1} = \|\mathbf{g}\|_{\mathbf{N}}$

  $\mathbf{q}_{k+1} = \mathbf{g}/\beta_{k+1}$

  $\mathbf{w} = \mathbf{M}^{-1}\left(\mathbf{A}\mathbf{q}_{k+1} - \beta_{k+1}\mathbf{M}\mathbf{v}_k\right)$; $\alpha_{k+1} = \|\mathbf{w}\|_{\mathbf{M}}$

  $\mathbf{v}_{k+1} = \mathbf{w}/\alpha_{k+1}$

  $\zeta_{k+1} = -\dfrac{\beta_{k+1}}{\alpha_{k+1}}\zeta_k$

  $\mathbf{d}_{k+1} = \left(\mathbf{q}_{k+1} - \beta_{k+1}\mathbf{d}_k\right)/\alpha_{k+1}$

  $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \zeta_{k+1}\mathbf{v}_{k+1}$; $\mathbf{p}^{(k+1)} = \mathbf{p}^{(k)} - \zeta_{k+1}\mathbf{d}_{k+1}$

  $k = k + 1$

  if $k > d$ **then**

    $\bar{\xi}_{k,d} = \sqrt{\sum_{i=k-d+1}^{k}\zeta_i^2 \,/\, \sum_{i=1}^{k}\zeta_i^2}$

  end if

end while return $\mathbf{u}^{k+1}, \mathbf{p}^{k+1}$

In Line 8 of Algorithm 1, we have the application of $\mathbf{M}^{-1}$ to a vector, which represents what we call the *inner problem*. Typically, this is implemented as a call to a direct solver using the matrix $\mathbf{M}$ and the vector $\mathbf{A}\mathbf{q}_{k+1} - \beta_{k+1}\mathbf{M}\mathbf{v}_k$ as the right hand side. The main contribution of this work is a study of the behavior exhibited by Algorithm 1 when we replace the direct solver employed in Line 8 by an iterative one. In particular, for a target accuracy of the final GKB iterate, we want to minimize the total number of inner iterations.

Our choice for the inner solver is the unpreconditioned Conjugate Gradient (CG) algorithm, for its simplicity and relative generality. The strategies we propose in the subsequent sections do not rely on any specific feature of this inner solver, and are meant to be applicable regardless of this choice. We are interested in reducing the total number of inner iterations in a relative and general manner. This is why we do not take preconditioning for CG into account, which is usually problem-dependent. We measure the effectiveness of our methods based on the percentage of inner iterations saved when compared against a scenario to be described in more detail in the following sections.

3 Problem description

As test problem, we will use a 2D Stokes flow in a rectangular channel domain $\Omega = [-1, L] \times [-1, 1]$ given by

$$\begin{aligned} -\Delta\vec{u} + \nabla p &= 0 \\ \nabla \cdot \vec{u} &= 0, \end{aligned} \tag{20}$$

More specifically, we will consider the Poiseuille flow problem, i.e. a steady Stokes problem with the exact solution

$$\begin{cases} u_x = 1 - y^2, \\ u_y = 0, \\ p = -2x + \text{constant}. \end{cases} \tag{21}$$

The boundary conditions are given as Dirichlet condition on the inflow $\Gamma_{in} = \{-1\}\times[-1, 1]$ (left boundary) and no-slip conditions on the top and bottom walls $\Gamma_c = [-1, L]\times\{-1\}\cup[-1, L]\times\{1\}$. The outflow at the right $\Gamma_{out} = \{L\}\times[-1, 1]$ (right) is represented as a Neumann condition

$$\begin{aligned} \frac{\partial u_x}{\partial x} - p &= 0 \\ \frac{\partial u_y}{\partial x} &= 0. \end{aligned}$$

We use Q2-Q1 Finite Elements as discretization method. Our sample matrices are generated by the Incompressible Flow & Iterative Solver Software (IFISS)[1] package [13], see the book by Elman et al. [14] for a more detailed description of this reference Stokes problem.

[1] `http://www.cs.umd.edu/~elman/ifiss3.6/index.html`

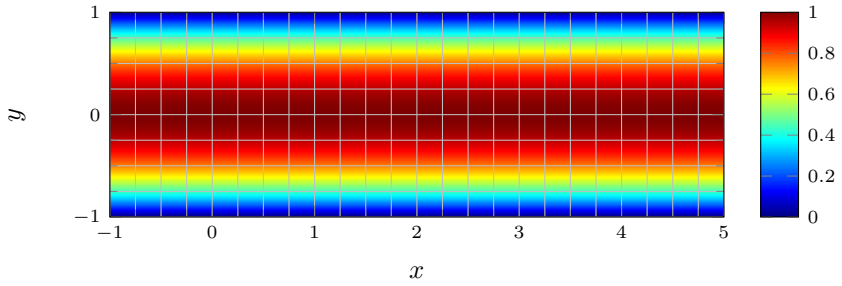

Figure 1: Exact solution to the Stokes problem in a channel of length 5. Plotted is the $1-y^2$ function, which represents the x direction velocity, overlaid with the mesh resulting from the domain discretization (Q2-Q1 Finite Elements Method).

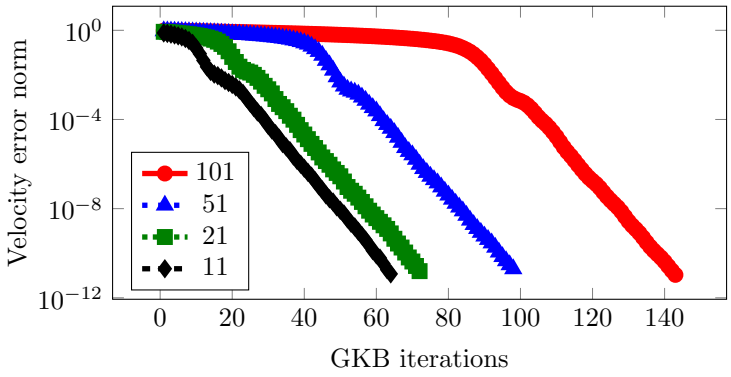

Figure 2: GKB convergence history for the IFISS channel problem. The length of each channel is given in the legend. Y-axis: Energy norm of the relative error for the velocity.

We first illustrate some particular features shown by GKB for this problem. We use a direct inner solver here, before discussing the influence of an iterative solver in subsequent sections. In Figure 2, we plot the convergence history for several channels of different lengths, which leads us to noticing the following details. The solver starts with a period of slow convergence, visually represented by a plateau, the length of which is proportional to the length of the channel. The rest of the convergence curve corresponds to a period of superlinear convergence, a phenomenon also known for other solvers of the Krylov family, such as CG. The presence of this plateau is especially relevant for our proposed strategies and, since it appears for each channel, we can conclude it is a significant feature of this class of channel problems. In the following numerical examples, we choose as boundary $L = 20$ and thus a domain of length 21 units.

4 Constant accuracy inner solver

Similar to what has been described by Golub et al. [17] for solving eigenvalue problems, we have observed that when using an iterative method as an inner solver, its accuracy has a clear effect on the overall accuracy of the outer solver (see Figure 3).

We solve the channel problem described in Section 3 with various configurations for the tolerance of the inner solver, and plot the resulting convergence curves in Figure 3. The outer solver is always GKB with a 10^{-7} tolerance. The cases we show are: a direct inner solver, three choices of constant inner solver tolerance (10^{-3}, 10^{-7} and 10^{-8}), and a final case using a low accuracy solver of (10^{-3}) only for the first two iterations, then a high accuracy one (10^{-14}).

The stopping criterion for the GKB algorithm is a delayed lower bound estimate for the energy norm of the primal variable (see Equation (17)). As such, GKB with a direct inner solver performs a few extra steps, achieving a higher accuracy than the one required, here around 10^{-8}.

Notice how the outer solver cannot achieve a higher accuracy than that of the inner solver. The outer solver stops reducing the error even before reaching the same accuracy as the inner solver. Replacing the exact inner solver by a CG method with a constant tolerance of 10^{-8} leads to a convergence process where the error norm eventually reaches a value just below the target accuracy of 10^{-7} and does not decrease further. This highlights the fact that the inner solver does not need to be exact in order to have GKB converge to the required solution. For this Poiseuille flow example, however, the inner solver must be least one order of magnitude more precise than the outer one.

In the last case examined here, we want to see if early imprecise iterations can be compensated later by others having a higher accuracy. This strategy of increasing accuracy has been found to work, e.g., in the case of the Newton method for nonlinear problems [11]. We tested the case when the first two iterations of GKB use an inner solver with tolerance 10^{-3}, with all the subsequent inner iterations employ a tolerance of 10^{-14}. The resulting curve shows a convergence history rather similar to the case where CG has a constant tolerance of 10^{-3}. The outer process cannot reduce the error norm below 10^{-3}, despite the fact that the bulk of the iterations employ a high-accuracy inner solver. This is in correspondence with which was observed by Golub et al. [17] for solving eigenvalue problems.

An interesting observation is that all the curves in Figure 3 overlap in their initial iterations, until they start straying from the apparent profile, eventually leveling off. In Section 5, we analyze the causes leading to these particular behaviors and link them to the accuracy of the inner solver.

5 Perturbation and error study

In this section we describe how the error associated with the iterates of Algorithm 1 behaves if we use an iterative solver for the systems involving $\mathbf{M}^{-1}$.

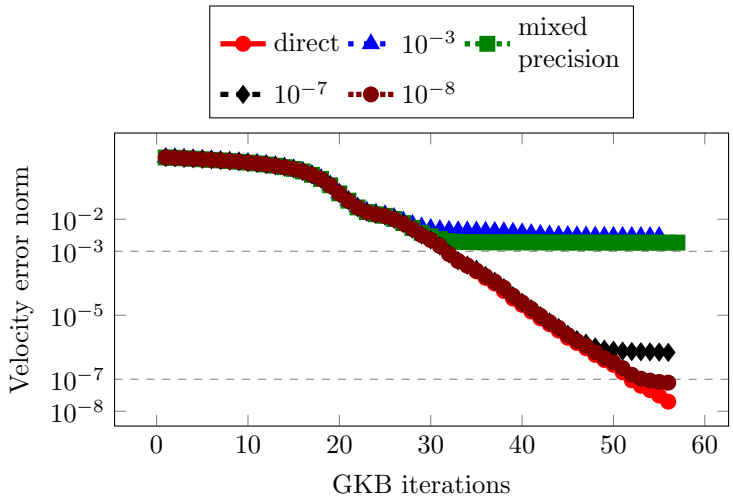

Figure 3: GKB convergence history for the IFISS Channel test case, depending on the CG tolerance (see legend). Y-axis: Energy norm of the relative error for the velocity. Target GKB tolerance 10^{-7}. Mixed precision: first two iterations 10^{-3}, afterwards 10^{-14}. The final value for each case with CG is: $3 \cdot 10^{-3}$, $7 \cdot 10^{-7}$, $8 \cdot 10^{-8}$, $2 \cdot 10^{-3}$. Only the cases and converge successfully, reducing the error norm below 10^{-7}.

We can think of the approximate solutions of these inner systems as perturbed versions of those we would get when using a direct solver. The error is then characterized in terms of this perturbation and the implications motivate our algorithmic strategies given in the subsequent sections. With this characterization, we can also explain the results in Section 4.

The use of an iterative inner solver directly affects the columns of the $\mathbf{V}$ matrix. In the following, $\mathbf{V}$ denotes the unperturbed matrix, with $\mathbf{E_V}$ being the associated perturbation matrix. In particular, we are interested in the $\mathbf{M}$ norm of the individual columns of $\mathbf{E_V}$, which gives us an idea of how far we are from the "ideal" columns of $\mathbf{V}$.

Changes in the $\mathbf{v}$ and $\mathbf{q}$ vectors also have an impact on their respective norms α and β, which shift away from the values they would normally have with a direct inner solver. In turn, these changes propagate to the coefficients ζ used to update the iterates $\mathbf{u}$ and $\mathbf{p}$. Our observations concern the $\mathbf{z}$ vector, its perturbation $\mathbf{e_z}$ and their effect on the error of the primal variable $\mathbf{u}$ measured in the $\mathbf{M}$ norm. The entries of $\mathbf{z}$ change sign every iteration, but we will only consider them in absolute value, as it is their magnitude which is important. In the following, we will denote perturbed quantities with a hat.

5.1 High initial accuracy followed by relaxation

In this subsection, we take a closer look at the interactions between the perturbation and the error. For us, perturbation is the result of using an inexact inner solver and represents a quantity which can prevent the outer solver from reducing the error below a certain value. The error itself needs to be precisely defined, as it may contain several components, each minimized by a different process. Because we focus on the difference between the perturbed and the unperturbed GKB, sources of error that affect both versions, such as the round-off error, are not included in the following discussion. According to the observations by Jiránek and Rozložník, the accuracy of the outer solver depends primarily on that of the inner solver, since the perturbations introduced by an iterative solver dominate those related to finite-precision arithmetic [18]. We take the exact solution $\mathbf{u}$ to be equal to $\mathbf{u}_n$, the n-th iterate of the unperturbed GKB with exact arithmetic.

At step k of the GKB, we have the error,

$$\|\mathbf{e}_k\|_{\mathbf{M}} = \|\hat{\mathbf{u}}_k - \mathbf{u}\|_{\mathbf{M}}\,, \tag{22}$$

where $\hat{\mathbf{u}}_k$ is the current approximate solution and $\mathbf{u}$ is the exact one. Both can be written as linear combinations of columns from $\mathbf{V}$ with coefficients from $\mathbf{z}$. Let $\hat{\mathbf{u}}_k$ come from an inexact version of Algorithm 1, where the solution of the inner problem (a matrix-vector product with $\mathbf{M}^{-1}$) includes perturbations. The term $\mathbf{u} = \mathbf{V}_n\mathbf{z}_n$ is available after n steps of Algorithm 1 in exact arithmetic, without perturbations. We separate the first k terms, which have been computed, from the remaining $(n-k)$.

$$\begin{aligned}
\|\mathbf{e}_k\|_{\mathbf{M}}^2 = \|\hat{\mathbf{u}}_k - \mathbf{u}\|_{\mathbf{M}}^2 &= \left\| (\mathbf{V}_k + \mathbf{E}_{\mathbf{V}})(\mathbf{z}_k + \mathbf{e}_{\mathbf{z}}) - [\mathbf{V}_k \mathbf{V}_{n-k}] \begin{bmatrix} \mathbf{z}_k \\ \mathbf{z}_{n-k} \end{bmatrix} \right\|_{\mathbf{M}}^2 \\
&= \|\mathbf{E}_{\mathbf{V}}\mathbf{z}_k + \mathbf{E}_{\mathbf{V}}\mathbf{e}_{\mathbf{z}} + \mathbf{V}_k\mathbf{e}_{\mathbf{z}} - \mathbf{V}_{n-k}\mathbf{z}_{n-k}\|_{\mathbf{M}}^2 \\
&\leq \|\mathbf{E}_{\mathbf{V}}\mathbf{z}_k\|_{\mathbf{M}}^2 + \|\mathbf{E}_{\mathbf{V}}\mathbf{e}_{\mathbf{z}}\|_{\mathbf{M}}^2 + \|\mathbf{e}_{\mathbf{z}}\|_2^2 + \|\mathbf{z}_{n-k}\|_2^2
\end{aligned} \tag{23}$$

In the last line, we have made use of the $\mathbf{M}$-orthonormality of the $\mathbf{V}$ matrices.

In the case of a direct inner solver, we can leave out the perturbation terms, recovering the result $\|\mathbf{e}_k\|_{\mathbf{M}}^2 = \|\mathbf{z}_{n-k}\|_2^2 = \sum_{i=k+1}^{n} \zeta_i^2$ given by Arioli [1]. This is simply the error coming from approximating $\mathbf{u}$ (a linear combination of n $\mathbf{M}$-orthogonal vectors) by $\mathbf{u}_k$ (a linear combination of only k $\mathbf{M}$-orthogonal vectors). This term decreases as we perform more steps of Algorithm 1 ($k \to n$). By truncating the sum $\sum_{i=k+1}^{n} \zeta_i^2$, we obtain a lower bound for the squared error.

The remaining three terms in Equation (23) include the perturbation coming from the inexact inner solution. Our goal is to minimize the total number of iterations of the inner solver, so we are interested in knowing how large can these terms be allowed to be, such that we still recover a final solution of the required accuracy. The answer is to keep them just below the final value of the fourth one, $\|\mathbf{z}_{n-k}\|_2$, below the acceptable algebraic error. If they are larger,

the final accuracy will suffer. If they are significantly smaller, then our inner solver is unnecessarily precise and expensive.

The following observations rely on the behavior of the $\mathbf{z}$ vector. At each iteration, this vector gains an additional entry, while leaving the previous ones unchanged. These entries form a (mostly) decreasing sequence and have a magnitude below 1 when reaching the superlinear convergence phase. Unfortunately, we cannot yet provide a formal proof of these properties, but having seen them consistently reappear in our numerical experiments encourages us to consider them for motivating our approach. These properties appear in both cases, with and without perturbation.

The decrease in the entries of the coefficient vector used to build the approximation has also been observed and described for other Krylov methods (see references [30, 27, 28]). Their context is that of inexact matrix-vector products, which is another way of viewing our case. The fact that new entries of $\mathbf{z}$ are simply appended to the old ones and that they are smaller than one is linked to the particular construction specific to GKB.

Back to Equation (23), let us assume the perturbation at each iteration is constant, i.e. the $\mathbf{M}$ norm of each column of $\mathbf{E_V}$ is equal to the same constant. Then, the vector $\mathbf{E_V z}_k$ will be a linear combination of perturbation vectors with coefficients from $\mathbf{z}_k$. Following our observations concerning the entries of $\mathbf{z}_k$, the first terms of the linear combination will be the dominant ones, with later terms contributing less and less to the sum. If the perturbation of the first $\mathbf{v}$ has an $\mathbf{M}$ norm below our target accuracy, the term $\|\mathbf{E_V z}_k\|_\mathbf{M}$ will never contribute to the error. We can allow the $\mathbf{M}$ norm of the columns of $\mathbf{E_V}$ to increase, knowing the effect of the perturbation will be reduced by the entries of $\mathbf{z}$, which are decreasing and less than one. The GKB solution can be computed in a less expensive way, as long as the term $\|\mathbf{E_V z}_k\|_\mathbf{M}$ is kept below our target accuracy. The perturbation should initially be small, then allowed to increase proportionally to the decrease of the entries in $\mathbf{z}$.

Next, we describe the terms including $\mathbf{e_z}$. Let the following define the perturbed entries of $\hat{\mathbf{z}}$

$$\hat{\zeta}_k = -\hat{\zeta}_{k-1}\frac{\hat{\beta}_k}{\hat{\alpha}_k} = -\hat{\zeta}_{k-1}(\frac{\beta_k}{\alpha_k} + \epsilon_k).$$

The term ϵ_k is the perturbation introduced at iteration k, coming from the shifted norms associated with $\mathbf{q}_k$ and $\mathbf{v}_k$. This term is then multiplied by $\hat{\zeta}_{k-1}$ which, according to our empirical observations, decreases at (almost) every step. If we assume ϵ_k is constant, the entries of $\mathbf{e_z}$ decrease in magnitude and the norm $\|\mathbf{e_z}\|_2$ is mostly dominated by the first vector entry. The strategy described for the term $\|\mathbf{E_V z}_k\|_\mathbf{M}$ also keeps $\|\mathbf{e_z}\|_2$ small. We start with a perturbation norm below the target accuracy, to ensure the quality of the final iterate. Gradually, we allow an increase in the perturbation norm proportional to the decrease of $\hat{\zeta}_k$ to reduce the costs of the inner solver. Finally, since the vector $\mathbf{e_z}$ decreases similarly to $\mathbf{z}$, the term $\|\mathbf{E_V e_z}\|_\mathbf{M}$ can be described in the same way as $\|\mathbf{E_V z}_k\|_\mathbf{M}$.

We close this section by emphasizing the important role played by the first iterations and how the initial perturbations can affect the accuracy of the solution. Notice that the perturbation terms included refer to all the k steps, not just the latest one. Relaxation strategies that start with a low accuracy and gradually increase it are unlikely to work for GKB and other algorithms with similar error minimization properties. Since the first vectors computed are the ones that contribute the most to reducing the error, they should be determined as precisely as possible. Even if we follow a perturbed iteration exclusively by very accurate ones, this will not prevent the perturbation from being transmitted to all the subsequent vectors, and potentially be amplified by multiplication with matrices and floating-point error. With these observations in mind, we can understand the results in Section 4.

These findings are in line with those concerning other Kylov methods in the presence of inexactness (see Section 11 of the survey by Simoncini and Szyld [29] and the references therein). GKB is not the only method which benefits from lowering the accuracy of the inner process, and the reason why this is possible is linked to the decreasing entries of the coefficient vector.

6 Relaxation strategy choices

We have seen in Section 5.1 that we can allow the perturbation norm to increase in a safe way, as long as the process is guided by the decrease of $\left|\hat{\zeta}\right|$. This means that we can adapt the tolerance of the inner solver, such that each call is increasingly cheaper, without compromising the accuracy of the final GKB iterate. Then, at step k we can call the inner solver with a tolerance equal to $\tau/f(\zeta)$. The scalar τ represents a constant chosen as either the target accuracy for the final GKB solution, or something stricter, to counteract possible losses coming from floating-point arithmetic. The function f is chosen based on the considerations described below, with the goal of minimizing the number of inner iterations.

A similar relaxation strategy was used in a numerical study by Bouras and Frayssé [7] to control the magnitude of the perturbation introduced by performing inexact matrix-vector products. They employ Krylov methods with a residual norm minimization property, so the proposed criterion divides the target accuracy by the latest residual norm. In our case, because of the minimization property in Equation (19), we need to use the error norm instead of the residual, since it is the only quantity which is strictly decreasing. Due to the actual error norm being unknown, we rely on approximations found via ζ.

Considering the error characterization of the unperturbed process $\|\mathbf{e}_k\|_{\mathbf{M}}^2 = \sum_{i=k+1}^{n} \zeta_i^2$, we can approximate the error by the first term of the sum, which is the dominant one. However, when starting iteration k we do not know ζ_{k+1}, not even ζ_k, so we cannot choose a tolerance for the inner solver required to compute $\mathbf{u}_k$ based on these. What we can do is predict these values via extrapolation, using information from the known values ζ_{k-1} and ζ_{k-2}. We know that in general $\frac{\beta_k}{\alpha_k} = \frac{\zeta_k}{\zeta_{k-1}}$ acts as a local convergence factor for the $|\zeta|$ sequence. We

approximate the one for step k by using the previous one $\frac{\zeta_{k-1}}{\zeta_{k-2}}$. Then, we can compute the prediction $\tilde{\zeta}_k := \zeta_{k-1}\frac{\zeta_{k-1}}{\zeta_{k-2}}$. By squaring the local convergence factor, we get an approximation for ζ_{k+1} as $\tilde{\zeta}_{k+1} := \zeta_{k-1}\left(\frac{\zeta_{k-1}}{\zeta_{k-2}}\right)^2$, which we can use to approximate $\|\mathbf{e}_k\|_{\mathbf{M}}$ and adapt the tolerance of the inner solver.

In practice, we only consider processes which include perturbation, and assume we have no knowledge of the unperturbed values $|\zeta|$. As such, for better readability, we drop the hat notation with the implicit convention that we are referring to values which do include perturbation and use them in the extrapolation rule above.

For some isolated iterations, it is possible that $|\zeta_k| \geq |\zeta_{k-1}|$. This behavior is then amplified through extrapolation, potentially leading to even larger values. In turn, this can cause an increase in the accuracy of the inner solver, following a stricter value for the tolerance parameter $\tau/f(\zeta)$. In Section 5.1, we have shown that there is no benefit in increasing this accuracy. The new perturbation would be smaller in norm, but the error $\|\mathbf{e}_k\|_{\mathbf{M}}$ would be dominated by the previous, larger perturbation. As such, we propose computing several candidate values for the stopping tolerance of the inner solver, and choose the one with maximum value. Since these are only scalar quantities, the associated computational effort is negligible, but the impact of a well-chosen tolerance sequence can lead to significant savings in the total number of inner iterations. The candidate values are:

$$\begin{cases} \text{the value at the previous step,} \\ \tau/\left|\zeta_{k-1}\right|, \\ \tau/\left|\tilde{\zeta}_k\right|, \\ \tau/\left|\tilde{\zeta}_{k+1}\right|. \end{cases} \tag{24}$$

To prevent a limitless growth of the tolerance parameter, we impose a maximum value of 0.1. All these choices are safe in the sense that they do not lead to introduction of perturbations which prevent the outer solver from reaching the target accuracy.

We proceed by testing these relaxations strategies on the problem described in Section 3. The initial tolerance for CG is set to $\tau = 10^{-8}$, one order of magnitude more precise than the one set of GKB. As a baseline for comparison, we first keep the tolerance constant, equal to τ. Then, we introduce adaptivity using $\tau/\left|\zeta_{k-1}\right|$. The third case changes the tolerance according to $\tau/\left|\tilde{\zeta}_{k+1}\right|$, the latter term being a predicted approximation of the current error. Finally, we employ a hybrid approach, where all candidate values in Equation (24) are computed, but only the largest one is used. In the legends of the following plots, these four cases are labeled `Constant`, `Adaptive`, `Predicted`, and `Hybrid`, respectively. To monitor GKB convergence, we track the lower bound for the energy norm of the error corresponding to the primal variable given in Equation (17). For easy reference, all the choices used and their respective labels are given below. We

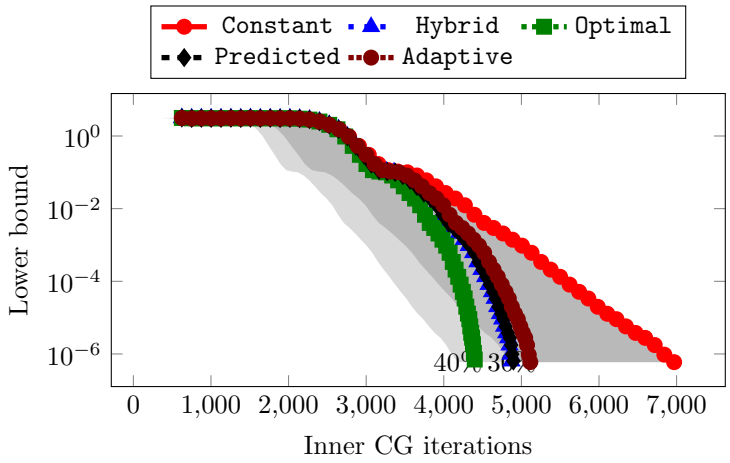

Figure 4: Lower bound (Equation (17)) for the error norm associated with the GKB iterates versus the cumulative number of inner CG iterations when solving the original problem from Section 3. The parameter used in `Optimal` is 0.05. See Equations (25) to (29) for the strategies denoted by the labels.

define $\tau = 10^{-8}$.

$$(\texttt{Constant}) : \tau, \tag{25}$$

$$(\texttt{Adaptive}) : \tau/|\zeta_{k-1}|, \tag{26}$$

$$(\texttt{Predicted}) : \tau/|\tilde{\zeta}_{k+1}|, \tag{27}$$

$$(\texttt{Hybrid}) : \max\left\{\tau/|\zeta_{k-1}|, \tau/|\tilde{\zeta}_{k}|, \tau/|\tilde{\zeta}_{k+1}|, \text{previous value}\right\}. \tag{28}$$

$$(\texttt{Optimal}) : \tau/(parameter\cdot|\zeta_{k-1}|), \tag{29}$$

Only the last scenario above, `Optimal`, is left to explain. To see if the parameter-free choices can be improved, we run one more case which includes adaptivity by using $|\zeta_{k-1}|$, but also one constant parameter tuned experimentally. This is motivated by the fact that the considerations leading to Equation (24) rely mostly on approximations and inequalities, which means we have an over-estimate of the error. It may be possible to reduce the total number of iterations further, by including an (almost) optimal, problem-dependent constant. The goal is to find a sequence of tolerance parameters with terms that are as large as possible, while guaranteeing the accuracy of the final GKB iterate.

All the results are given in Table 1 and Figure 4. `Hybrid` offers the highest savings among the parameter-free choices (30%), but `Optimal`, the test with the problem-dependent constant, reveals that we can still improve this performance by about 6%.

Table 1: Reduction of the total number of CG iterations. The CG tolerance is relaxed according to Equations (25) to (29). The parameter in `Optimal` is 0.05.

CG tolerance	`Constant`	`Adaptive`	`Predicted`	`Hybrid`	`Optimal`
CG iterations	6963	5115	4897	4873	4399
Savings %	-	26.54	29.67	30.02	36.82

6.1 Increasing the savings by working on a simplified problem

Considering the observations in Section 5.1 and the results plotted in Figure 4, we can significantly reduce the accuracy of the inner solver only when the outer solver is in a superlinear convergence phase, when the $|\zeta|$ sequence decreases rapidly. How much we can relax depends on the slope of the convergence curve. As such, to get the maximum reduction of the total number of iterations, the problem needs to be simplified, such that the convergence curve is as steep as possible and has no plateau. It is common to pair Krylov methods with other strategies, such as preconditioning, in order to improve their convergence behavior. The literature on these kinds of approaches is rich [21, 22, 6, 24]. The following tests quantify how beneficial is the interaction between our proposed relaxation scheme and these other strategies.

It has been shown by Arioli and Orban that the GKB applied to the saddle-point system is equivalent to the CG algorithm applied to the Schur complement equation [25, Chapter 5]. As such, the first step towards accelerating GKB is to consider the Schur complement, defined as $\mathbf{S} := \mathbf{A}^T\mathbf{M}_{-1}\mathbf{A}$, especially its spectrum. Ideally, a spectrum with tightly clustered values and no outliers leads to rapid GKB convergence [20]. To get as close as possible to this clustering we use the following two methods to induce positive changes in the spectrum: preconditioning with the Least Squares Commutator (LSC) [12] and eigenvalue deflation. Each of them operates differently and leads to convergence curves with different traits.

In Figure 5, we plot the GKB convergence curve for each of these, using a direct inner solver. The LSC aligns the small values in the spectrum with the main cluster and brings everything closer together. The corresponding GKB convergence curve has no plateau and is much steeper than the curve for the unpreconditioned case. Using deflation, we remove the five smallest values from the spectrum, which constitute outliers with the respect to the main cluster. The other values remain unchanged. As such, its convergence curve no longer has the initial plateau, but is otherwise the same as in the original problem.

For both of these cases we apply the same strategies of relaxing the inner tolerance, to see how many total CG iterations we can save. The rest of the set-up is identical to that described for Table 1. We tabulate the results in Tables 2 and 3 and plot them in Figures 6 and 7. They highlight that the best parameter-free results are obtained when using `Hybrid`, which leads to savings of about 50%, depending on the specific case. When comparing this parameter-

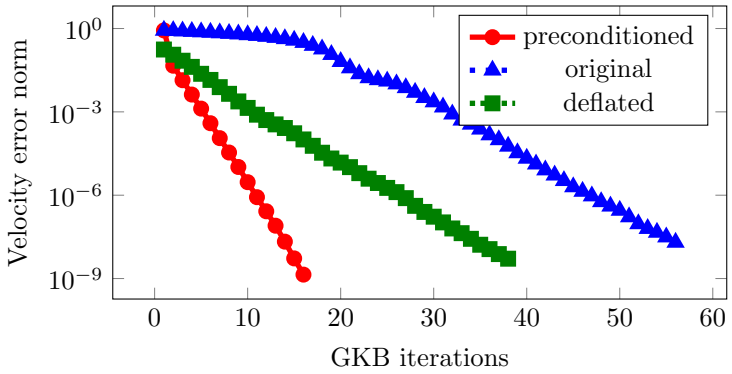

Figure 5: GKB convergence curves for the IFISS channel test case before and after spectral clustering. Y-axis: Energy norm of the relative error for the velocity. Target GKB tolerance 10^{-7}. Using the Least Squares Commutator preconditioner or deflation of the smallest five spectral outliers.

Table 2: Reduction of the total number of CG iterations after using the LSC preconditioner. The CG tolerance is relaxed according to Equations (25) to (29). The parameter used in `Optimal` is 0.007.

CG tolerance	`Constant`	`Adaptive`	`Predicted`	`Hybrid`	`Optimal`
CG iterations	2052	1301	1073	1046	919
Savings %	-	36.60	47.71	49.03	55.21

free approach to `Optimal`, which includes an experimental constant, we find that the hybrid approach can still be improved. Nonetheless, the difference in CG iterations savings is not very high (up to 6%), which supports the idea that our proposed strategy is efficient in a general-use setting. An additional observation pertaining to the plots is that even if convergence is relatively fast (Figure 6) or slow (Figure 7), the final savings are still around 50%, as long as there is no plateau.

Table 3: Reduction of the total number of CG iterations after using deflation. The CG tolerance is relaxed according to Equations (25) to (29). The parameter used in `Optimal` is 0.09.

CG tolerance	`Constant`	`Adaptive`	`Predicted`	`Hybrid`	`Optimal`
CG iterations	4830	2625	2416	2411	2110
Savings %	-	45.65	49.98	50.08	56.31

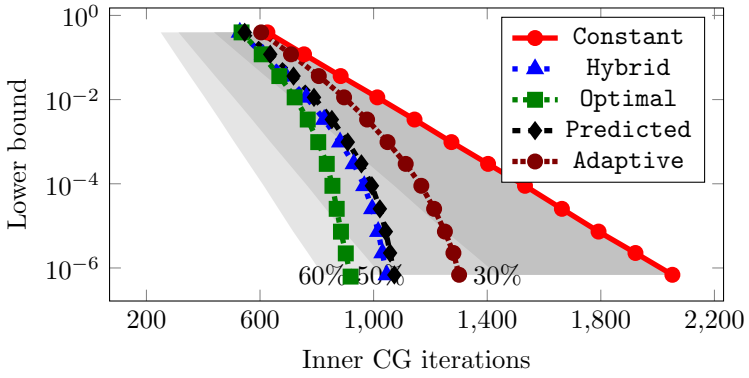

Figure 6: Lower bound (Equation (17)) for the error norm associated with the GKB iterates versus the cumulative number of inner CG iterations when solving the problem from Section 3. The problem includes preconditioning with the LSC. The parameter used in `Optimal` is 0.007. See Equations (25) to (29) for the strategies denoted by the labels.

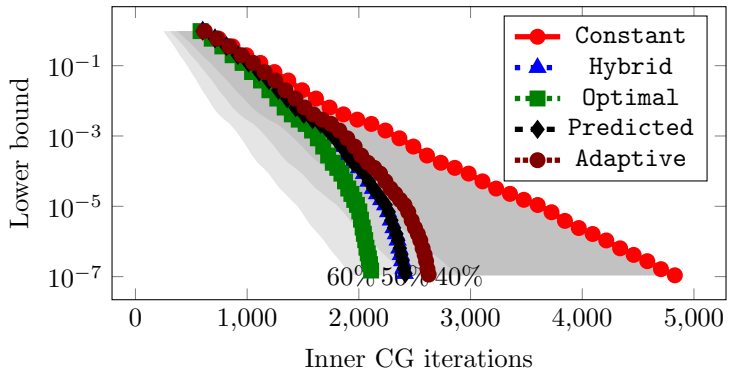

Figure 7: Lower bound (Equation (17)) for the error norm associated with the GKB iterates versus the cumulative number of inner CG iterations when solving the problem from Section 3. The problem includes deflation of the five smallest spectral outliers. The parameter used in `Optimal` is 0.09. See Equations (25) to (29) for the strategies denoted by the labels.

Table 4: Reduction of the total number of CG iterations after using the AL ($\eta = 1000$). The CG tolerance is relaxed according to Equations (25) to (29). The parameter used in `Optimal` is 0.005.

CG tolerance	`Constant`	`Adaptive`	`Predicted`	`Hybrid`	`Optimal`
CG iterations	2601	1886	1707	1661	1647
Savings %	-	27.49	34.37	36.14	36.68

7 GKB with the augmented Lagrangian approach

The method of the AL has been used successfully to speed up the convergence of the GKB algorithm [20], with this effect being theoretically explained by Arioli et al. [20]. Maybe most striking is the potential to reach mesh-independent convergence, provided that the augmentation parameter is large enough. Another use of the AL method is to transform the (1,1)-block of a saddle-point system, say $\mathbf{W}$, from a positive semi-definite matrix to a positive definite one. However, this can happen only if the off-diagonal block $\mathbf{A}$ is full rank or, more generally, if $\ker(\mathbf{W}) \cap \ker(\mathbf{A}^T) = \{\mathbf{0}\}$.

Let $\mathbf{N} \in \mathbb{R}^{n \times n}$ be a symmetric, positive definite matrix. For a given symmetric, positive semi-definite matrix $\mathbf{W} \in \mathbb{R}^{m \times m}$, we can transform it into a positive-definite one by

$$\mathbf{M} := \mathbf{W} + \mathbf{A}\mathbf{N}^{-1}\mathbf{A}^T. \tag{30}$$

The upper right-hand side term $\mathbf{g}$ then becomes

$$\mathbf{g} := \mathbf{g} + \mathbf{A}\mathbf{N}^{-1}\mathbf{r}. \tag{31}$$

With these changes in place, we can proceed to using the GKB algorithm, as described in Section 2.

Note that if the matrix $\mathbf{W}$ is already symmetric positive-definite, the transformation of the (1,1)-block is not necessary for using the GKB method. However, the application of the AL approach does lead to a better conditioning of the Schur complement, which significantly improves convergence speed [20]. As in Section 2, we choose $\mathbf{N} = \frac{1}{\eta}\mathbf{I}$. There is as usual no free lunch: depending on the conditioning of the matrix $\mathbf{A}$ and the magnitude of η, the AL can also degrade the conditioning of the $\mathbf{M}$ matrix as a side-effect.

We test whether the augmentation interacts with the strategies we propose in Section 6, namely if we can still achieve about 50% savings in the total number of inner iterations. The strategies are applied when solving the problem described in Section 3 after an augmentation with a parameter $\eta = 1000$, with the results being given in Table 4 and plotted in Figure 8. Comparing the percentage of iterations saved in this case to those obtained in Section 6, it is clear that, when combined with the AL method, the strategy of variable inner tolerance does help reducing the total number of inner iterations, but by a lower percentage.

Since the AL method modifies the (1,1)-block of the saddle-point system, it changes the difficulty of the inner problem and how many iterations the inner

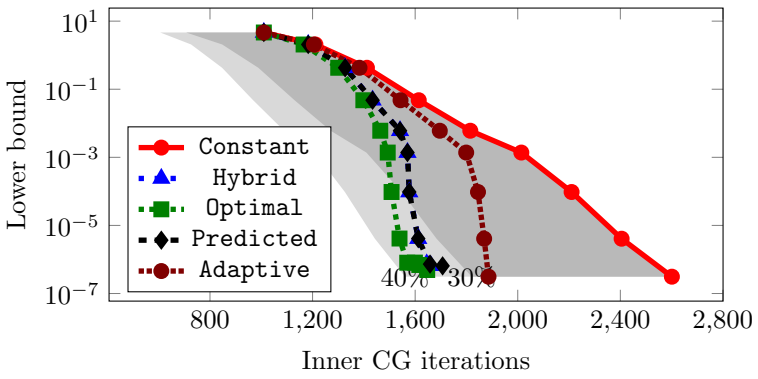

Figure 8: Lower bound (Equation (17)) for the error norm associated with the GKB iterates versus the cumulative number of inner CG iterations when solving the problem from Section 3. The problem includes the AL ($\eta = 1000$). The parameter used in `Optimal` is 0.005. See Equations (25) to (29) for the strategies denoted by the labels.

solver needs to perform. As such, a global comparison in terms of number of inner iterations, among all the scenarios we studied (original, preconditioned, deflated, including the AL) is not fair unless the inner problem has the same degree of difficulty for all the cases.

To verify the generality of our method, we also apply it in a different context than that described in Section 3. Let us consider a Mixed Poisson problem. We solve the Poisson equation $-\Delta u = f$ on the unit square $(0,1)^2$ using a mixed formulation. We introduce the vector variable $\vec{\sigma} = \nabla u$. Find $(\vec{\sigma}, u) \in \Sigma \times W$ such that

$$\vec{\sigma} - \nabla u = 0 \tag{32}$$

$$-\mathrm{div}(\vec{\sigma}) = f. \tag{33}$$

where homogeneous Dirichlet boundary conditions are imposed for u at all walls. The forcing term f is random and uniformly drawn in $(0,1)$. The discretization is done with a lowest order Raviart-Thomas space $\Sigma^h \subset \Sigma$, and a space $W^h \subset W$ containing piece-wise constant basis functions. We used the finite element package Firedrake[2] coupled with a PETSc [3, 2, 4] implementation of GKB [3], adapted to include dynamical relaxation, to produce the following numerical results. We used the implementation provided by Firedrake[4]. The test case has 328 192 degrees of freedom, of which 197 120 are associated with the (1,1)-block. The GKB delay parameter is set to 3. The augmentation parameter η is set to 500 and the tolerance for the GKB set to 10^{-5}. The results are presented in Figure 9. We confirm the results presented above with a reduction of over

[2] `www.firedrakeproject.org`
[3] `https://petsc.org/release/docs/manualpages/PC/PCFIELDSPLIT.html#PCFIELDSPLIT`
[4] `https://www.firedrakeproject.org/demos/saddle_point_systems.py.html`

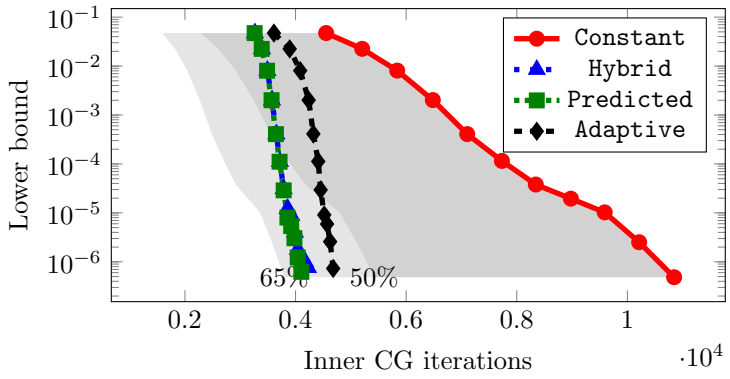

Figure 9: Lower bound (Equation (17)) for the error norm associated with the GKB iterates versus the cumulative number of inner CG iterations when solving the Mixed Poisson problem. We also use the AL ($\eta = 500$). See Equations (25) to (28) for the strategies denoted by the labels.

Table 5: Reduction of the total number of CG iterations after using the AL ($\eta = 500$) on the Mixed Poisson problem. The CG tolerance is relaxed according to Equations (25) to (28).

CG tolerance	`Constant`	`Adaptive`	`Predicted`	`Hybrid`
CG iterations	10845	4680	4105	4225
Savings %	-	56.84	62.15	61.04

60% in the total number of inner CG iterations with respect to the constant accuracy set up.

8 Conclusions

We have studied the behavior of the GKB algorithm in the case where the inner problem, i.e. the solution of a linear system, is performed iteratively. We have found that the inner solver does not need to be as precise as a direct one in order to achieve a GKB solution of a predefined accuracy.

Furthermore, we have proposed algorithmic strategies that reduce the cost of the inner solver, quantified as the cumulative number of inner iterations. This is possible by selecting criteria to change the stopping tolerance. To motivate these choices, we have studied the perturbation generated by the inexact inner solver. The findings show that the perturbation introduced in early iterations has a higher impact on the accuracy of the solution compared to later ones.

We devised a dynamic way of adapting the accuracy of the inner solver at each call to minimize its cost. The initial, high accuracy is gradually reduced, maintaining the resulting perturbation under control.

Our relaxation strategy is inexpensive, easy to implement, and has reduced

the total number of inner iterations by 33-63% in our tests. The experiments also show that including methods such as deflation, preconditioning and the augmented Lagrangian has no negative impact and can lead to a higher percentage of savings. Another advantage is that our method does not rely on additional parameters and is thus usable in a black-box fashion.

Acknowledgments The authors thank Mario Arioli for many inspiring discussions and advice.

References

[1] M. Arioli. Generalized Golub–Kahan bidiagonalization and stopping criteria. *SIAM Journal on Matrix Analysis and Applications*, 34(2):571–592, 2013.

[2] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Steven Benson, Jed Brown, Peter Brune, Kris Buschelman, Emil Constantinescu, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Václav Hapla, Tobin Isaac, Pierre Jolivet, Dmitry Karpeev, Dinesh Kaushik, Matthew G. Knepley, Fande Kong, Scott Kruger, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Lawrence Mitchell, Todd Munson, Jose E. Roman, Karl Rupp, Patrick Sanan, Jason Sarich, Barry F. Smith, Stefano Zampini, Hong Zhang, Hong Zhang, and Junchao Zhang. PETSc/TAO users manual. Technical Report ANL-21/39 - Revision 3.16, Argonne National Laboratory, 2021.

[3] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Steven Benson, Jed Brown, Peter Brune, Kris Buschelman, Emil M. Constantinescu, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Václav Hapla, Tobin Isaac, Pierre Jolivet, Dmitry Karpeev, Dinesh Kaushik, Matthew G. Knepley, Fande Kong, Scott Kruger, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Lawrence Mitchell, Todd Munson, Jose E. Roman, Karl Rupp, Patrick Sanan, Jason Sarich, Barry F. Smith, Stefano Zampini, Hong Zhang, Hong Zhang, and Junchao Zhang. PETSc Web page. `https://petsc.org/`, 2021.

[4] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[5] Manuel Baumann and Martin B Van Gijzen. Nested krylov methods for shifted linear systems. *SIAM Journal on Scientific Computing*, 37(5):S90–S112, 2015.

[6] Michele Benzi, Gene H. Golub, and Jörg Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[7] Amina Bouras and Valérie Frayssé. Inexact matrix-vector products in krylov methods for solving linear systems: a relaxation strategy. *SIAM Journal on Matrix Analysis and Applications*, 26(3):660–678, 2005.

[8] Amina Bouras, Valérie Frayssé, and Luc Giraud. A relaxation strategy for inner-outer linear solvers in domain decomposition methods, 2000. Technical Report 17.

[9] Julianne Chung and Silvia Gazzola. Flexible krylov methods for ℓ_p regularization. *SIAM Journal on Scientific Computing*, 41(5):S149–S171, 2019.

[10] Achiya Dax. A restarted krylov method with inexact inversions. *Numerical Linear Algebra with Applications*, 26(1):e2213, 2019.

[11] Ron S Dembo, Stanley C Eisenstat, and Trond Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400–408, 1982.

[12] Howard Elman, Victoria E Howle, John Shadid, Robert Shuttleworth, and Ray Tuminaro. Block preconditioners based on approximate commutators. *SIAM Journal on Scientific Computing*, 27(5):1651–1668, 2006.

[13] Howard Elman, Alison Ramage, and David Silvester. Algorithm 866: IFISS, a Matlab toolbox for modelling incompressible flow. *ACM Trans. Math. Softw.*, 33:2–14, 2007.

[14] Howard C Elman, David J Silvester, and Andrew J Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Numerical Mathematics and Scie, 2014.

[15] Yogi A Erlangga and Reinhard Nabben. Multilevel projection-based nested krylov iteration for boundary value problems. *SIAM Journal on Scientific Computing*, 30(3):1572–1595, 2008.

[16] Silvia Gazzola and Malena Sabate Landman. Regularization by inexact krylov methods with applications to blind deblurring. *SIAM Journal on Matrix Analysis and Applications*, 42(4):1528–1552, 2021.

[17] Gene H. Golub, Zhenyue Zhang, and Hongyuan Zha. Large sparse symmetric eigenvalue problems with homogeneous linear constraints: the lanczos process with inner–outer iterations. *Linear Algebra and its Applications*, 309(1):289 – 306, 2000.

[18] Pavel Jiránek and Miroslav Rozložník. Maximum attainable accuracy of inexact saddle point solvers. *SIAM journal on matrix analysis and applications*, 29(4):1297–1321, 2008.

[19] René Kehl, Reinhard Nabben, and Daniel B Szyld. Adaptive multilevel krylov methods. *Electronic Transactions on Numerical Analysis*, 51, 2019.

[20] C. Kruse, V. Darrigrand, N. Tardieu, M. Arioli, and U. Rüde. Application of an iterative golub-kahan algorithm to structural mechanics problems with multi-point constraints. *Adv. Model. and Simul. in Eng. Sci*, 7, 2020.

[21] Daniel Loghin and Andrew J Wathen. Schur complement preconditioning for elliptic systems of partial differential equations. *Numerical linear algebra with applications*, 10(5-6):423–443, 2003.

[22] Daniel Loghin and Andrew J Wathen. Analysis of preconditioners for saddle-point problems. *SIAM Journal on Scientific Computing*, 25(6):2029–2049, 2004.

[23] Lois Curfman McInnes, Barry Smith, Hong Zhang, and Richard Tran Mills. Hierarchical krylov and nested krylov methods for extreme-scale computing. *Parallel Computing*, 40(1):17–31, 2014.

[24] Maxim A Olshanskii and Valeria Simoncini. Acquired clustering properties and solution of certain saddle point systems. *SIAM journal on matrix analysis and applications*, 31(5):2754–2768, 2010.

[25] Dominique Orban and Mario Arioli. *Iterative solution of symmetric quasi-definite linear systems*. SIAM, 2017.

[26] Youcef Saad. A flexible inner-outer preconditioned gmres algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469, 1993.

[27] Valeria Simoncini and Daniel B Szyld. Theory of inexact krylov subspace methods and applications to scientific computing. *SIAM Journal on Scientific Computing*, 25(2):454–477, 2003.

[28] Valeria Simoncini and Daniel B Szyld. Relaxed krylov subspace approximation. In *PAMM: Proceedings in Applied Mathematics and Mechanics*, volume 5, pages 797–800. Wiley Online Library, 2005.

[29] Valeria Simoncini and Daniel B Szyld. Recent computational developments in krylov subspace methods for linear systems. *Numerical Linear Algebra with Applications*, 14(1):1–59, 2007.

[30] Jasper Van Den Eshof and Gerard LG Sleijpen. Inexact krylov subspace methods for linear systems. *SIAM Journal on Matrix Analysis and Applications*, 26(1):125–153, 2004.

[31] Jasper Van Den Eshof, Gerard LG Sleijpen, and Martin B van Gijzen. Relaxation strategies for nested krylov methods. *Journal of Computational and Applied Mathematics*, 177(2):347–365, 2005.

[32] Henk A Van der Vorst and Cornelis Vuik. Gmresr: a family of nested gmres methods. *Numerical Linear Algebra with Applications*, 1(4):369–386, 1994.

[33] Shengjie Xu and Fei Xue. Inexact rational krylov subspace method for eigenvalue problems. *Numerical Linear Algebra with Applications*, page e2437, 2022.

Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

Inexact inner-outer Golub-Kahan bidiagonalization method: A relaxation strategy

Vincent Darrigrand[1] | Andrei Dumitrasc[2] | Carola Kruse[3] | Ulrich Rüde[2,3]

[1]IRIT-CNRS, Toulouse, France
[2]Chair for Computer Science 10 - System Simulation, Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany
[3]Cerfacs, Toulouse, France

Correspondence
*Andrei Dumitrasc, Friedrich-Alexander-Universität Erlangen-Nürnberg, Cauerstraße 11, 91058 Erlangen, Germany
Email: andrei.dumitrasc@fau.de

Funding Information
This research was supported by the EU Horizon 2020 Project Energy oriented Center of Excellence: toward exascale for energy (EoCoE-II), Project ID: 824158 and the Bavarian Academic Center for Central, Eastern and Southeastern Europe (BAYHOST)

Summary

We study an inexact inner-outer generalized Golub-Kahan algorithm for the solution of saddle-point problems with a two-times-two block structure. In each outer iteration, an inner system has to be solved which in theory has to be done exactly. Whenever the system is getting large, an inner exact solver is, however, no longer efficient or even feasible and iterative methods must be used. We focus this article on a numerical study showing the influence of the accuracy of an inner iterative solution on the accuracy of the solution of the block system. Emphasis is further given on reducing the computational cost, which is defined as the total number of inner iterations. We develop relaxation techniques intended to dynamically change the inner tolerance for each outer iteration to further minimize the total number of inner iterations. We illustrate our findings on a Stokes problem and validate them on a mixed formulation of the Poisson problem.

KEYWORDS:
Inner-outer iterative methods, Golub-Kahan bidiagonalization, Saddle-point problems, Stokes equation

1 | INTRODUCTION

Saddle-point systems can be found in a variety of application fields, such as, for example, mixed finite element methods in fluid dynamics or interior point methods in optimization. An extensive overview about application fields and solution methods for this kind of problems is presented in the well-known article[?] by Benzi, Golub and Liesen. In our following study, we want to focus on an iterative solver based on the Golub-Kahan bidiagonalization: the generalized Golub-Kahan bidiagonalization (GKB) algorithm. This solver is designed for saddle-point systems, and was introduced by Arioli[?]. It belongs to the family of Krylov subspace methods and, as such, relies on specific orthogonality conditions, as we will review in more detail in Section 2. Enforcing these orthogonality conditions requires solving an *inner problem*, i.e. formally computing products with matrix inverses (as described in Algorithm 1). In practice, this computation is performed with a linear system solver. For this task, we will explore in this article the use of iterative methods to serve as replacement for direct methods that have been used within GKB so far. This is essential for very large problems, such as those coming from a discretized Partial Differential Equation (PDE) in 2D or 3D, when direct solvers may reach their limits.

Using an inner iterative solver might also be advantageous from another point of view as we motivate in the following. The solution of large linear systems is often the bottleneck in scientific computing. The computational cost and, consequently, the execution time and/or the energy consumption can become prohibitive. For the inner-outer iterative GKB solver in turn, the principal and costliest part is the solution of the inner system at each outer iteration. One approximate metric to measure the cost of the GKB solver is the aggregate sum of the number of inner iterations. For a given setup, the cost of the GKB method can

hence be optimized by executing only a minimal number of inner iterations necessary for achieving a prescribed accuracy of the solution. To reduce this number, there are two possible steps to be taken into account. In a first step, for a given application it is often unnecessary to solve the linear system with the highest achievable accuracy. This could be the case, for example, in the solution of a discretized PDE, when the discretization already introduces an error. A precise solution of the linear system would not improve the numerical solution with respect to the analytic solution of the PDE any further than the discretization allows. Next, we come to the second step which will be the main point of the study in this paper. The solution of the inner linear system in the GKB method has to be exact, in theory. If we choose a rather low accuracy for the outer iterative solver, an inner exact solution might, however, no longer be necessary, as long as the inner error does not alter the chosen accuracy of the numerical solution. This strategy results in a further reduction of the number of inner iterations, since the inner solver will converge in fewer iterations when a less strict stopping tolerance is used.

In the following study, we address the case where the inner solver has a prescribed stopping tolerance and then how this limited accuracy affects the outer process and the quality of its iterates. We will show that, with the appropriate choice of parameters, it is possible to make use of inner iterative solvers without compromising the accuracy of the GKB result. As it can be seen immediately, the lower the accuracy for the inner solver, the less expensive the GKB method will be. Furthermore, we take advantage of the versatility of iterative methods by adapting the stopping tolerance of the inner solver dynamically. In other words, we prescribe the tolerance of the inner solver according to some criteria determined at each outer iteration. This can lead to a reduction of the cost, since only a minimal number of inner iterations are executed. Typically, we will reduce the required accuracy for later instances of the inner solver, since later steps of the outer GKB-iteration may contribute less to the overall accuracy.

One particular advantage of our proposed method is its generality. The strategy is independent of other choices which are problem-specific, such as the preconditioner for a Krylov method. We perform most of our tests on a relatively small Stokes flow problem, to illustrate the salient features. We confirm our findings by one final test on a larger case of the mixed Poisson problem, including the use of the augmented Lagrangian method, to demonstrate the use in a realistic scenario.

Our study has a similar context as other works on inexact Krylov methods[??], where these algorithms have been investigated from a numerical perspective. In these articles, the inexactness originates from a limited accuracy of the matrix-vector multiplication or that of the solution of a local sub-problem. Similar to what we have described above, it was found that the inner accuracy can be varied from step to step while still achieving convergence of the outer method. It was shown experimentally that the initial tolerance should be strict, then relaxed gradually, with the change being guided by the latest residual norm. Other works complemented the findings with theoretical insights, relevant to several algorithms of the Krylov family[???]. It was noted that, in some cases, unless a problem-dependent constant is included, the outer solver may fail to converge if the accuracy of the inner solution is adapted only based on the residual norm. This constant can be computed based on extreme singular values, as shown by Simoncini and Szyld[?]. Another source of inexactness can be the application of a preconditioner via an iterative method. Van den Eshof, Sleijpen and van Gijzen considered inexactness in Krylov methods originating both from matrix-vector products and variable preconditioning, using iterative methods from the GMRES family[?]. Similarly to earlier work, their analysis relies on the connection between the residual and the accuracy of the solution to the inner problem. Since applying the preconditioner has the same effect as a matrix-vector product, the same strategies can be applied to more complex, flexible algorithms, such as those involving variable preconditioning: FGMRES[?], GMRESR[?], etc. A flexible version of the Golub-Kahan bidiagonalization is employed by Chung and Gazzola to find regularized solutions to a problem of image deblurring[?]. In a more recent paper with the same application, Gazzola and Landman develop inexact Krylov methods as a way to deal with approximate knowledge of $\mathbf{A}$ and $\mathbf{A}^T$[?]. Erlangga and Nabben construct a framework including nested Krylov solvers. They develop a multilevel approach to shift small eigenvalues, leading to a faster convergence of the linear solver[?]. In subsequent work related to multilevel Krylov methods, Kehl, Nabben and Szyld apply preconditioning in a flexible way, via an adaptive number of inner iterations[?]. Baumann and van Gijzen analyze solving shifted linear systems and, by applying flexible preconditioning, also develop nested Krylov solvers[?]. McInnes et al. consider hierarchical and nested Krylov methods with a small number of vector inner products, with the goal of reducing the need for global synchronization in a parallel computing setting[?].

Other than solving linear systems, inexact Krylov methods have been studied when tackling eigenvalue problems, as in the paper by Golub, Zhang and Zha[?]. Although using different arguments, it was shown that the strategy of increasing the inner tolerance is successful for this kind of problem as well. Xu and Xue make use of an inexact rational Krylov method to solve nonsymmetric eigenvalue problems and observe that the accuracy of the inner solver (GMRES) can be relaxed in later outer steps, depending on the value of the eigenresidual[?]. Dax computes the smallest eigenvalues of a matrix via a restarted Krylov solver which includes inexact matrix inversion[?].

Our paper is structured as follows: in Section 2, we review the theory and properties of the GKB algorithm; in Section 3, we describe the specific problem we chose to use as test case for the numerical experiments; Section 4 is meant to illustrate the interactions between the accuracy of the inner solver and that of the outer one in a numerical test setting; Section 5 describes the link between the error of the outer solver and the perturbation induced by the use of an iterative inner solver. We describe and test our proposed strategy of using a variable tolerance parameter for the inner solver in Section 6. We explore the interaction between the method of the Augmented Lagrangian (AL) and our strategy in Section 7. The final section is devoted to concluding remarks.

2 | GENERALIZED GOLUB-KAHAN ALGORITHM

We are interested in saddle-point problems of the form

$$\begin{bmatrix} \mathbf{M} & \mathbf{A} \\ \mathbf{A}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \mathbf{r} \end{bmatrix} \tag{1}$$

with $\mathbf{M} \in \mathbb{R}^{m\times m}$ being a symmetric positive definite matrix and $\mathbf{A} \in \mathbb{R}^{m\times n}$ a full rank constraint matrix. The generalized GKB algorithm for the solution of a class of saddle-point systems was introduced by Arioli[?]. To apply it to the system (1), we first need to have the upper block of the right-hand side to be equal to 0. To this end, we use the transformation

$$\mathbf{u} = \mathbf{w} - \mathbf{M}^{-1}\mathbf{g}, \tag{2}$$

$$\mathbf{b} = \mathbf{r} - \mathbf{A}^T\mathbf{u}. \tag{3}$$

The resulting system is

$$\begin{bmatrix} \mathbf{M} & \mathbf{A} \\ \mathbf{A}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{b} \end{bmatrix}, \tag{4}$$

which is equivalent to that in Equation (1). We can recover the $\mathbf{w}$ variable as $\mathbf{w} = \mathbf{u} + \mathbf{M}^{-1}\mathbf{g}$.

Let $\mathbf{N} \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix. To properly describe the GKB algorithm, we need to define the following norms

$$\|\mathbf{v}\|_{\mathbf{M}} = \sqrt{\mathbf{v}^T\mathbf{M}\mathbf{v}}; \qquad \|\mathbf{q}\|_{\mathbf{N}} = \sqrt{\mathbf{q}^T\mathbf{N}\mathbf{q}}; \qquad \|\mathbf{y}\|_{\mathbf{N}^{-1}} = \sqrt{\mathbf{y}^T\mathbf{N}^{-1}\mathbf{y}}. \tag{5}$$

Given the right-hand side vector $\mathbf{b} \in \mathbb{R}^n$, the first step of the bidiagonalization is

$$\beta_1 = \|\mathbf{b}\|_{\mathbf{N}^{-1}}, \quad \mathbf{q}_1 = \mathbf{N}^{-1}\mathbf{b}/\beta_1. \tag{6}$$

After k iterations, the partial bidiagonalization is given by

$$\begin{cases} \mathbf{A}\mathbf{Q}_k = \mathbf{M}\mathbf{V}_k\mathbf{B}_k, & \mathbf{V}_k^T\mathbf{M}\mathbf{V}_k = \mathbf{I}_k \\ \mathbf{A}^T\mathbf{V}_k = \mathbf{N}\mathbf{Q}_k\mathbf{B}_k^T + \beta_{k+1}\mathbf{q}_{k+1}\mathbf{e}_k^T, & \mathbf{Q}_k^T\mathbf{N}\mathbf{Q}_k = \mathbf{I}_k \end{cases}, \tag{7}$$

with the bidiagonal matrix

$$\mathbf{B}_k = \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \dots & 0 \\ 0 & \alpha_2 & \beta_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & \alpha_{k-1} & \beta_k \\ 0 & \dots & 0 & 0 & \alpha_k \end{bmatrix} \tag{8}$$

and the residual term $\beta_{k+1}\mathbf{q}_{k+1}\mathbf{e}_k^T$. The columns of $\mathbf{V}_k$ are orthonormal vectors with respect to the inner product and norm induced by $\mathbf{M}$, while the same holds for $\mathbf{Q}_k$ and $\mathbf{N}$ respectively

$$\begin{aligned} \mathbf{v}_i^T\mathbf{M}\mathbf{v}_j = 0, \forall i \neq j; \qquad \|\mathbf{v}_k\|_{\mathbf{M}} = 1; \\ \mathbf{q}_i^T\mathbf{N}\mathbf{q}_j = 0, \forall i \neq j; \qquad \|\mathbf{q}_k\|_{\mathbf{N}} = 1. \end{aligned} \tag{9}$$

Prior to the normalization leading to $\mathbf{v}_k$ and $\mathbf{q}_k$, the norms are stored as α_k for $\mathbf{v}_k$ and β_k for $\mathbf{q}_k$, as detailed in algorithm 1. Using $\mathbf{V}_k$, $\mathbf{Q}_k$ and the relations in Equation (7), we can transform the system from Equation (4) into a simpler form

$$\begin{bmatrix} \mathbf{I}_k & \mathbf{B}_k \\ \mathbf{B}_k^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{z}_k \\ \mathbf{y}_k \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{Q}_k^T\mathbf{b} \end{bmatrix}. \tag{10}$$

With the choice for $\mathbf{q}_1$ given in Equation (6), we have that $\mathbf{Q}_k^T\mathbf{b} = \beta_1\mathbf{e}_1$. The solution components to Equation (10) are then given by

$$\mathbf{z}_k = \beta_1\mathbf{B}_k^{-T}\mathbf{e}_1; \quad \mathbf{y}_k = -\mathbf{B}_k^{-1}\mathbf{z}_k, \tag{11}$$

where $\mathbf{B}_k^{-T}$ is the inverse of $\mathbf{B}_k^T$. We can build the k-th approximate solution to Equation (4) as

$$\mathbf{u}_k = \mathbf{V}_k\mathbf{z}_k; \quad \mathbf{p}_k = \mathbf{Q}_k\mathbf{y}_k. \tag{12}$$

In particular, after a number of $k = n$ steps and assuming exact arithmetic, we have $\mathbf{u}_k = \mathbf{u}$ and $\mathbf{p}_k = \mathbf{p}$, meaning we have found the exact solution to Equation (4). A proof of why n terms are sufficient to find the exact solution is given in the introductory paper by Arioli[?] . This corresponds to a scenario where it is necessary to perform the n iterations, although, for specific problems with particular features, the solution may be found after fewer steps. As $k \to n$, the quality of the approximation improves ($\mathbf{u}_k \to \mathbf{u}$ and $\mathbf{p}_k \to \mathbf{p}$), with the bidiagonalization residual $\beta_{k+1}\mathbf{q}_{k+1}\mathbf{e}_k^T$ vanishing for $k = n$.

Given the structure of $\beta_1\mathbf{e}_1$ and $\mathbf{B}^T$, we find

$$\zeta_1 = \frac{\beta_1}{\alpha_1}, \quad \zeta_k = \zeta_{k-1}\frac{\beta_k}{\alpha_k}, \quad \mathbf{z}_k = \begin{bmatrix}\mathbf{z}_{k-1}\\ \zeta_k\end{bmatrix} \tag{13}$$

in a recursive manner. Then, $\mathbf{u}_k$ is computed as $\mathbf{u}_k = \mathbf{u}_{k-1} + \zeta_k\mathbf{v}_k$. In order to obtain a recursive formula for $\mathbf{p}$ as well, we introduce the vector

$$\mathbf{d}_k = \frac{\mathbf{q}_k - \beta_k\mathbf{d}_{k-1}}{\alpha_k}, \quad \mathbf{d}_1 = \frac{\mathbf{q}_1}{\alpha_1}. \tag{14}$$

Finally, the update formulas are

$$\mathbf{u}_k = \mathbf{u}_{k-1} + \zeta_k\mathbf{v}_k, \quad \mathbf{p}_k = \mathbf{p}_{k-1} - \zeta_k\mathbf{d}_k. \tag{15}$$

At step k of Algorithm 1, we have the following error in the energy norm.

$$\begin{aligned}\|\mathbf{e}_k\|_{\mathbf{M}}^2 = \|\mathbf{u}_k - \mathbf{u}\|_{\mathbf{M}}^2 &= \left\|\mathbf{V}_k\mathbf{z}_k - [\mathbf{V}_k\mathbf{V}_{n-k}]\begin{bmatrix}\mathbf{z}_k\\ \mathbf{z}_{n-k}\end{bmatrix}\right\|_{\mathbf{M}}^2\\ &= \|\mathbf{V}_{n-k}\mathbf{z}_{n-k}\|_{\mathbf{M}}^2 = \|\mathbf{z}_{n-k}\|_2^2 = \sum_{i=k+1}^{n}\zeta_i^2\end{aligned} \tag{16}$$

In the last line, we have made use of the $\mathbf{M}$-orthonormality of the $\mathbf{V}$ matrices. If we truncate the sum above to only its first d terms, we get a lower bound on the energy norm of the error. The subscript d stands for *delay*, because we can compute this lower bound corresponding to a given step k only after an additional d steps

$$\xi_{k,d}^2 = \sum_{i=k+1}^{k+d+1}\zeta_i^2 < \|\mathbf{e}_k\|_{\mathbf{M}}^2. \tag{17}$$

With this bound for the absolute error, we can devise one for the relative error in Equation (18), which is then used as stopping criterion in Line 15 of Algorithm 1.

$$\bar{\xi}_{k,d}^2 = \frac{\sum_{i=k-d+1}^{k}\zeta_i^2}{\sum_{i=1}^{k}\zeta_i^2}. \tag{18}$$

The GKB algorithm has the following error minimization property. Let $\mathcal{V}_k = span\{\mathbf{v}_1, ..., \mathbf{v}_k\}$ and $\mathcal{Q}_k = span\{\mathbf{q}_1, ..., \mathbf{q}_k\}$. Then, for any arbitrary step k, we have that

$$\min_{\substack{\mathbf{u}_k\in\mathcal{V}_k,\\ (\mathbf{A}^T\mathbf{u}_k-\mathbf{b})\perp\mathcal{Q}_k}} \|\mathbf{u} - \mathbf{u}_k\|_{\mathbf{M}} \tag{19}$$

is met for $\mathbf{u}_k$ as computed by Algorithm 1.

For brevity and because the GKB algorithm features this minimization property for the primal variable, our presentation will focus on the velocity for Stokes problems. The stopping criteria for our proposed algorithmic strategies rely on approximations of the velocity error norm. For all the numerical experiments that we have performed, the pressure error norm is close to that of the velocity (less than an order of magnitude apart). In the cases where we operate on a different subspace, as a result of preconditioning, we find that the pressure error norm is actually smaller than that for the velocity. In the case where the dual variable is equally important as the primal, one can use a monolithic approach, such as applying MINRES to the complete saddle-point system.

The GKB (as implemented by Algorithm 1) is a nested iterative scheme in which each outer loop involves solving an inner linear system. According to the theory given in the paper by Arioli[?] , the matrices $\mathbf{M}$ and $\mathbf{N}$ have to be inverted exactly in each

iteration. We can choose $\mathbf{N} = \frac{1}{\eta}\mathbf{I}$, whose inversion reduces to a scalar multiplication. In the following sections, unless otherwise specified, we consider $\eta = 1$. On the other hand, the matrix $\mathbf{M}$ depends on the underlying differential equations or the problem setting in general. As long as the matrix $\mathbf{M}$ is of moderate size, a robust direct solver can be used. For large problems, however, a direct solution might no longer be possible and an iterative solver will be required. At this point, we face two problems. First, depending on the application, inverting $\mathbf{M}$ might be more or less costly. Second, to achieve a solution quality close to machine precision, an iterative solver might require a considerable number of iteration steps.

Algorithm 1 Golub-Kahan bidiagonalization algorithm

Require: $\mathbf{M}, \mathbf{A}, \mathbf{N}, \mathbf{b}$, maxit

1: $\beta_1 = \|\mathbf{b}\|_{\mathbf{N}^{-1}}$; $\mathbf{q}_1 = \mathbf{N}^{-1}\mathbf{b}/\beta_1$
2: $\mathbf{w} = \mathbf{M}^{-1}\mathbf{A}\mathbf{q}_1$; $\alpha_1 = \|\mathbf{w}\|_{\mathbf{M}}$; $\mathbf{v}_1 = \mathbf{w}/\alpha_1$
3: $\zeta_1 = \beta_1/\alpha_1$; $\mathbf{d}_1 = \mathbf{q}_1/\alpha_1$; $\mathbf{u}^{(1)} = \zeta_1\mathbf{v}_1$; $\mathbf{p}^{(1)} = -\zeta_1\mathbf{d}_1$;
4: $\bar{\xi}_{1,d} = 1$; $k = 1$;
5: **while** $\bar{\xi}_{k,d} >$ tolerance and $k <$ maxit **do**
6: $\quad \mathbf{g} = \mathbf{N}^{-1}\left(\mathbf{A}^T\mathbf{v}_k - \alpha_k\mathbf{N}\mathbf{q}_k\right)$; $\beta_{k+1} = \|\mathbf{g}\|_{\mathbf{N}}$
7: $\quad \mathbf{q}_{k+1} = \mathbf{g}/\beta_{k+1}$
8: $\quad \mathbf{w} = \mathbf{M}^{-1}\left(\mathbf{A}\mathbf{q}_{k+1} - \beta_{k+1}\mathbf{M}\mathbf{v}_k\right)$; $\alpha_{k+1} = \|\mathbf{w}\|_{\mathbf{M}}$
9: $\quad \mathbf{v}_{k+1} = \mathbf{w}/\alpha_{k+1}$
10: $\quad \zeta_{k+1} = -\dfrac{\beta_{k+1}}{\alpha_{k+1}}\zeta_k$
11: $\quad \mathbf{d}_{k+1} = \left(\mathbf{q}_{k+1} - \beta_{k+1}\mathbf{d}_k\right)/\alpha_{k+1}$
12: $\quad \mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \zeta_{k+1}\mathbf{v}_{k+1}$; $\mathbf{p}^{(k+1)} = \mathbf{p}^{(k)} - \zeta_{k+1}\mathbf{d}_{k+1}$
13: $\quad k = k + 1$
14: $\quad$ **if** $k > d$ **then**
15: $\quad\quad \bar{\xi}_{k,d} = \sqrt{\sum_{i=k-d+1}^{k}\zeta_i^2 / \sum_{i=1}^{k}\zeta_i^2}$
16: $\quad$ **end if**
17: **end while return** $\mathbf{u}^{k+1}, \mathbf{p}^{k+1}$

In Line 8 of Algorithm 1, we have the application of $\mathbf{M}^{-1}$ to a vector, which represents what we call the *inner problem*. Typically, this is implemented as a call to a direct solver using the matrix $\mathbf{M}$ and the vector $\mathbf{A}\mathbf{q}_{k+1} - \beta_{k+1}\mathbf{M}\mathbf{v}_k$ as the right hand side. The main contribution of this work is a study of the behavior exhibited by Algorithm 1 when we replace the direct solver employed in Line 8 by an iterative one. In particular, for a target accuracy of the final GKB iterate, we want to minimize the total number of inner iterations.

Our choice for the inner solver is the unpreconditioned Conjugate Gradient (CG) algorithm, for its simplicity and relative generality. The strategies we propose in the subsequent sections do not rely on any specific feature of this inner solver, and are meant to be applicable regardless of this choice. We are interested in reducing the total number of inner iterations in a relative and general manner. This is why we do not take preconditioning for CG into account, which is usually problem-dependent. We measure the effectiveness of our methods based on the percentage of inner iterations saved when compared against a scenario to be described in more detail in the following sections.

3 | PROBLEM DESCRIPTION

As test problem, we will use a 2D Stokes flow in a rectangular channel domain $\Omega = [-1, L] \times [-1, 1]$ given by

$$\begin{aligned} -\Delta\vec{u} + \nabla p &= 0 \\ \nabla \cdot \vec{u} &= 0, \end{aligned} \tag{20}$$

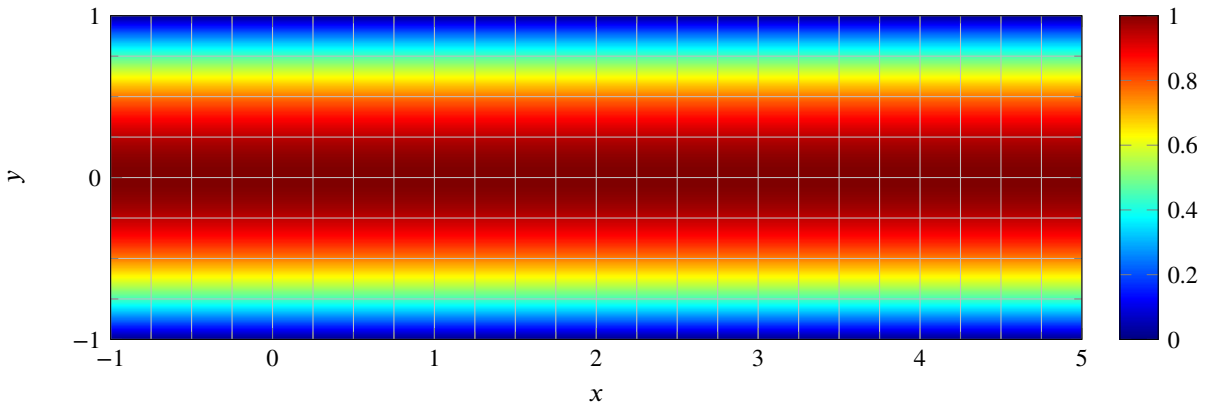

FIGURE 1 Exact solution to the Stokes problem in a channel of length 5. Plotted is the $1 - y^2$ function, which represents the x direction velocity, overlaid with the mesh resulting from the domain discretization (Q2-Q1 Finite Elements Method).

More specifically, we will consider the Poiseuille flow problem, i.e. a steady Stokes problem with the exact solution

$$\begin{cases} u_x = 1 - y^2, \\ u_y = 0, \\ p = -2x + \text{constant}. \end{cases} \tag{21}$$

The boundary conditions are given as Dirichlet condition on the inflow $\Gamma_{in} = \{-1\} \times [-1, 1]$ (left boundary) and no-slip conditions on the top and bottom walls $\Gamma_c = [-1, L] \times \{-1\} \cup [-1, L] \times \{1\}$. The outflow at the right $\Gamma_{out} = \{L\} \times [-1, 1]$ (right) is represented as a Neumann condition

$$\frac{\partial u_x}{\partial x} - p = 0$$
$$\frac{\partial u_y}{\partial x} = 0.$$

We use Q2-Q1 Finite Elements as discretization method. Our sample matrices are generated by the Incompressible Flow & Iterative Solver Software (IFISS)[1] package[?], see the book by Elman et al.[?] for a more detailed description of this reference Stokes problem.

We first illustrate some particular features shown by GKB for this problem. We use a direct inner solver here, before discussing the influence of an iterative solver in subsequent sections. In Figure 2, we plot the convergence history for several channels of different lengths, which leads us to noticing the following details. The solver starts with a period of slow convergence, visually represented by a plateau, the length of which is proportional to the length of the channel. The rest of the convergence curve corresponds to a period of superlinear convergence, a phenomenon also known for other solvers of the Krylov family, such as CG. The presence of this plateau is especially relevant for our proposed strategies and, since it appears for each channel, we can conclude it is a significant feature of this class of channel problems. In the following numerical examples, we choose as boundary $L = 20$ and thus a domain of length 21 units.

4 | CONSTANT ACCURACY INNER SOLVER

Similar to what has been described by Golub et al.[?] for solving eigenvalue problems, we have observed that when using an iterative method as an inner solver, its accuracy has a clear effect on the overall accuracy of the outer solver (see Figure 3).

We solve the channel problem described in Section 3 with various configurations for the tolerance of the inner solver, and plot the resulting convergence curves in Figure 3. The outer solver is always GKB with a 10^{-7} tolerance. The cases we show are: a direct inner solver, three choices of constant inner solver tolerance (10^{-3}, 10^{-7} and 10^{-8}), and a final case using a low accuracy solver of (10^{-3}) only for the first two iterations, then a high accuracy one (10^{-14}).

[1] http://www.cs.umd.edu/~elman/ifiss3.6/index.html

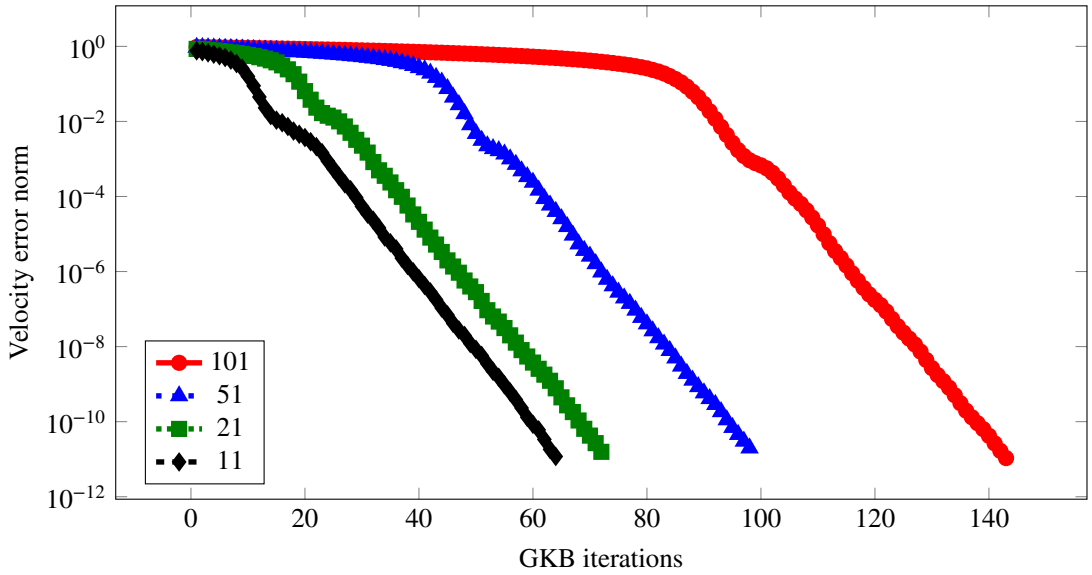

FIGURE 2 GKB convergence history for the IFISS channel problem. The length of each channel is given in the legend. Y-axis: Energy norm of the relative error for the velocity.

The stopping criterion for the GKB algorithm is a delayed lower bound estimate for the energy norm of the primal variable (see Equation (17)). As such, GKB with a direct inner solver performs a few extra steps, achieving a higher accuracy than the one required, here around 10^{-8}.

Notice how the outer solver cannot achieve a higher accuracy than that of the inner solver. The outer solver stops reducing the error even before reaching the same accuracy as the inner solver. Replacing the exact inner solver by a CG method with a constant tolerance of 10^{-8} leads to a convergence process where the error norm eventually reaches a value just below the target accuracy of 10^{-7} and does not decrease further. This highlights the fact that the inner solver does not need to be exact in order to have GKB converge to the required solution. For this Poiseuille flow example, however, the inner solver must be least one order of magnitude more precise than the outer one.

In the last case examined here, we want to see if early imprecise iterations can be compensated later by others having a higher accuracy. This strategy of increasing accuracy has been found to work, e.g., in the case of the Newton method for nonlinear problems[?]. We tested the case when the first two iterations of GKB use an inner solver with tolerance 10^{-3}, with all the subsequent inner iterations employ a tolerance of 10^{-14}. The resulting curve shows a convergence history rather similar to the case where CG has a constant tolerance of 10^{-3}. The outer process cannot reduce the error norm below 10^{-3}, despite the fact that the bulk of the iterations employ a high-accuracy inner solver. This is in correspondence with which was observed by Golub et al.[?] for solving eigenvalue problems.

An interesting observation is that all the curves in Figure 3 overlap in their initial iterations, until they start straying from the apparent profile, eventually leveling off. In Section 5, we analyze the causes leading to these particular behaviors and link them to the accuracy of the inner solver.

5 | PERTURBATION AND ERROR STUDY

In this section we describe how the error associated with the iterates of Algorithm 1 behaves if we use an iterative solver for the systems involving $\mathbf{M}^{-1}$. We can think of the approximate solutions of these inner systems as perturbed versions of those we would get when using a direct solver. The error is then characterized in terms of this perturbation and the implications motivate our algorithmic strategies given in the subsequent sections. With this characterization, we can also explain the results in Section 4.

The use of an iterative inner solver directly affects the columns of the $\mathbf{V}$ matrix. In the following, $\mathbf{V}$ denotes the unperturbed matrix, with $\mathbf{E_V}$ being the associated perturbation matrix. In particular, we are interested in the $\mathbf{M}$ norm of the individual columns of $\mathbf{E_V}$, which gives us an idea of how far we are from the "ideal" columns of $\mathbf{V}$.

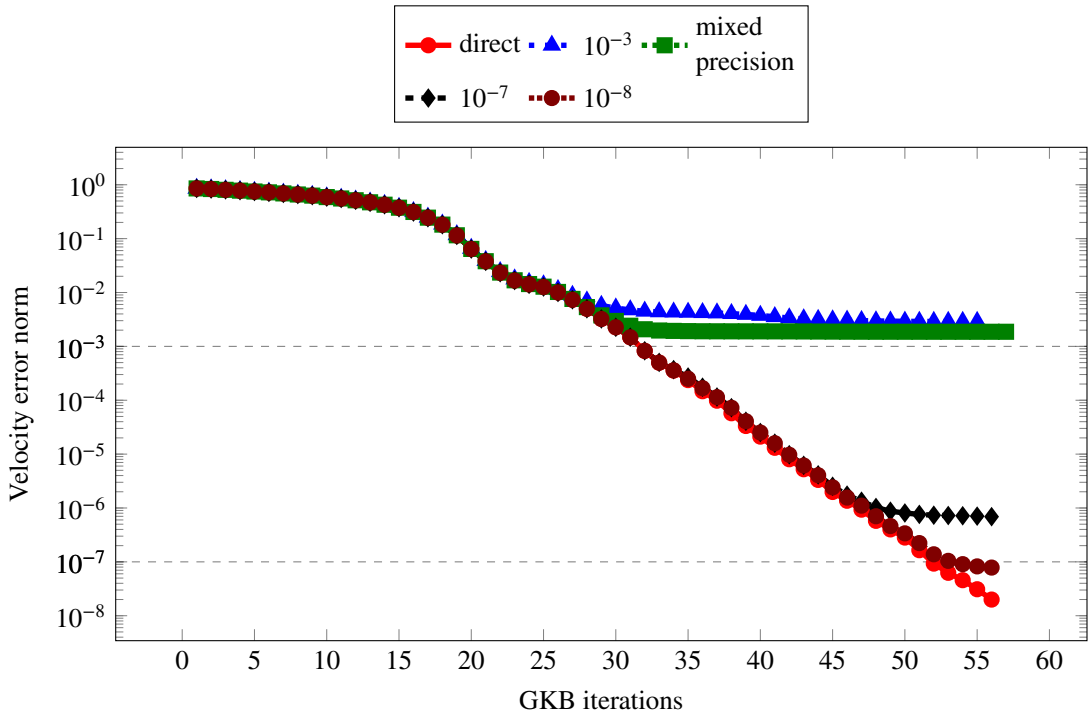

FIGURE 3 GKB convergence history for the IFISS Channel test case, depending on the CG tolerance (see legend). Y-axis: Energy norm of the relative error for the velocity. Target GKB tolerance 10^{-7}. Mixed precision: first two iterations 10^{-3}, afterwards 10^{-14}. The final value for each case with CG is: $3 \cdot 10^{-3}$, $7 \cdot 10^{-7}$, $8 \cdot 10^{-8}$, $2 \cdot 10^{-3}$. Only the cases and converge successfully, reducing the error norm below 10^{-7}.

Changes in the $\mathbf{v}$ and $\mathbf{q}$ vectors also have an impact on their respective norms α and β, which shift away from the values they would normally have with a direct inner solver. In turn, these changes propagate to the coefficients ζ used to update the iterates $\mathbf{u}$ and $\mathbf{p}$. Our observations concern the $\mathbf{z}$ vector, its perturbation $\mathbf{e}_\mathbf{z}$ and their effect on the error of the primal variable $\mathbf{u}$ measured in the $\mathbf{M}$ norm. The entries of $\mathbf{z}$ change sign every iteration, but we will only consider them in absolute value, as it is their magnitude which is important. In the following, we will denote perturbed quantities with a hat.

5.1 | High initial accuracy followed by relaxation

In this subsection, we take a closer look at the interactions between the perturbation and the error. For us, perturbation is the result of using an inexact inner solver and represents a quantity which can prevent the outer solver from reducing the error below a certain value. The error itself needs to be precisely defined, as it may contain several components, each minimized by a different process. Because we focus on the difference between the perturbed and the unperturbed GKB, sources of error that affect both versions, such as the round-off error, are not included in the following discussion. According to the observations by Jiránek and Rozložník, the accuracy of the outer solver depends primarily on that of the inner solver, since the perturbations introduced by an iterative solver dominate those related to finite-precision arithmetic[?]. We take the exact solution $\mathbf{u}$ to be equal to $\mathbf{u}_n$, the n-th iterate of the unperturbed GKB with exact arithmetic.

At step k of the GKB, we have the error,

$$\|\mathbf{e}_k\|_\mathbf{M} = \|\hat{\mathbf{u}}_k - \mathbf{u}\|_\mathbf{M}, \tag{22}$$

where $\hat{\mathbf{u}}_k$ is the current approximate solution and $\mathbf{u}$ is the exact one. Both can be written as linear combinations of columns from $\mathbf{V}$ with coefficients from $\mathbf{z}$. Let $\hat{\mathbf{u}}_k$ come from an inexact version of Algorithm 1, where the solution of the inner problem

(a matrix-vector product with $\mathbf{M}^{-1}$) includes perturbations. The term $\mathbf{u} = \mathbf{V}_n\mathbf{z}_n$ is available after n steps of Algorithm 1 in exact arithmetic, without perturbations. We separate the first k terms, which have been computed, from the remaining $(n-k)$.

$$\begin{aligned}\left\|\mathbf{e}_k\right\|_{\mathbf{M}}^2 = \left\|\hat{\mathbf{u}}_k - \mathbf{u}\right\|_{\mathbf{M}}^2 &= \left\|(\mathbf{V}_k + \mathbf{E}_{\mathbf{V}})(\mathbf{z}_k + \mathbf{e}_{\mathbf{z}}) - [\mathbf{V}_k\mathbf{V}_{n-k}]\begin{bmatrix}\mathbf{z}_k\\ \mathbf{z}_{n-k}\end{bmatrix}\right\|_{\mathbf{M}}^2 \\ &= \left\|\mathbf{E}_{\mathbf{V}}\mathbf{z}_k + \mathbf{E}_{\mathbf{V}}\mathbf{e}_{\mathbf{z}} + \mathbf{V}_k\mathbf{e}_{\mathbf{z}} - \mathbf{V}_{n-k}\mathbf{z}_{n-k}\right\|_{\mathbf{M}}^2 \\ &\leq \left\|\mathbf{E}_{\mathbf{V}}\mathbf{z}_k\right\|_{\mathbf{M}}^2 + \left\|\mathbf{E}_{\mathbf{V}}\mathbf{e}_{\mathbf{z}}\right\|_{\mathbf{M}}^2 + \left\|\mathbf{e}_{\mathbf{z}}\right\|_2^2 + \left\|\mathbf{z}_{n-k}\right\|_2^2\end{aligned} \tag{23}$$

In the last line, we have made use of the $\mathbf{M}$-orthonormality of the $\mathbf{V}$ matrices.

In the case of a direct inner solver, we can leave out the perturbation terms, recovering the result $\left\|\mathbf{e}_k\right\|_{\mathbf{M}}^2 = \left\|\mathbf{z}_{n-k}\right\|_2^2 = \sum_{i=k+1}^{n}\zeta_i^2$ given by Arioli[?]. This is simply the error coming from approximating $\mathbf{u}$ (a linear combination of n $\mathbf{M}$-orthogonal vectors) by $\mathbf{u}_k$ (a linear combination of only k $\mathbf{M}$-orthogonal vectors). This term decreases as we perform more steps of Algorithm 1 ($k \to n$). By truncating the sum $\sum_{i=k+1}^{n}\zeta_i^2$, we obtain a lower bound for the squared error.

The remaining three terms in Equation (23) include the perturbation coming from the inexact inner solution. Our goal is to minimize the total number of iterations of the inner solver, so we are interested in knowing how large can these terms be allowed to be, such that we still recover a final solution of the required accuracy. The answer is to keep them just below the final value of the fourth one, $\left\|\mathbf{z}_{n-k}\right\|_2$, below the acceptable algebraic error. If they are larger, the final accuracy will suffer. If they are significantly smaller, then our inner solver is unnecessarily precise and expensive.

The following observations rely on the behavior of the $\mathbf{z}$ vector. At each iteration, this vector gains an additional entry, while leaving the previous ones unchanged. These entries form a (mostly) decreasing sequence and have a magnitude below 1 when reaching the superlinear convergence phase. Unfortunately, we cannot yet provide a formal proof of these properties, but having seen them consistently reappear in our numerical experiments encourages us to consider them for motivating our approach. These properties appear in both cases, with and without perturbation.

The decrease in the entries of the coefficient vector used to build the approximation has also been observed and described for other Krylov methods (see references[???]). Their context is that of inexact matrix-vector products, which is another way of viewing our case. The fact that new entries of $\mathbf{z}$ are simply appended to the old ones and that they are smaller than one is linked to the particular construction specific to GKB.

Back to Equation (23), let us assume the perturbation at each iteration is constant, i.e. the $\mathbf{M}$ norm of each column of $\mathbf{E}_{\mathbf{V}}$ is equal to the same constant. Then, the vector $\mathbf{E}_{\mathbf{V}}\mathbf{z}_k$ will be a linear combination of perturbation vectors with coefficients from $\mathbf{z}_k$. Following our observations concerning the entries of $\mathbf{z}_k$, the first terms of the linear combination will be the dominant ones, with later terms contributing less and less to the sum. If the perturbation of the first $\mathbf{v}$ has an $\mathbf{M}$ norm below our target accuracy, the term $\left\|\mathbf{E}_{\mathbf{V}}\mathbf{z}_k\right\|_{\mathbf{M}}$ will never contribute to the error. We can allow the $\mathbf{M}$ norm of the columns of $\mathbf{E}_{\mathbf{V}}$ to increase, knowing the effect of the perturbation will be reduced by the entries of $\mathbf{z}$, which are decreasing and less than one. The GKB solution can be computed in a less expensive way, as long as the term $\left\|\mathbf{E}_{\mathbf{V}}\mathbf{z}_k\right\|_{\mathbf{M}}$ is kept below our target accuracy. The perturbation should initially be small, then allowed to increase proportionally to the decrease of the entries in $\mathbf{z}$.

Next, we describe the terms including $\mathbf{e}_{\mathbf{z}}$. Let the following define the perturbed entries of $\hat{\mathbf{z}}$

$$\hat{\zeta}_k = -\hat{\zeta}_{k-1}\frac{\hat{\beta}_k}{\hat{\alpha}_k} = -\hat{\zeta}_{k-1}(\frac{\beta_k}{\alpha_k} + \epsilon_k).$$

The term ϵ_k is the perturbation introduced at iteration k, coming from the shifted norms associated with $\mathbf{q}_k$ and $\mathbf{v}_k$. This term is then multiplied by $\hat{\zeta}_{k-1}$ which, according to our empirical observations, decreases at (almost) every step. If we assume ϵ_k is constant, the entries of $\mathbf{e}_{\mathbf{z}}$ decrease in magnitude and the norm $\left\|\mathbf{e}_{\mathbf{z}}\right\|_2$ is mostly dominated by the first vector entry. The strategy described for the term $\left\|\mathbf{E}_{\mathbf{V}}\mathbf{z}_k\right\|_{\mathbf{M}}$ also keeps $\left\|\mathbf{e}_{\mathbf{z}}\right\|_2$ small. We start with a perturbation norm below the target accuracy, to ensure the quality of the final iterate. Gradually, we allow an increase in the perturbation norm proportional to the decrease of $\hat{\zeta}_k$ to reduce the costs of the inner solver. Finally, since the vector $\mathbf{e}_{\mathbf{z}}$ decreases similarly to $\mathbf{z}$, the term $\left\|\mathbf{E}_{\mathbf{V}}\mathbf{e}_{\mathbf{z}}\right\|_{\mathbf{M}}$ can be described in the same way as $\left\|\mathbf{E}_{\mathbf{V}}\mathbf{z}_k\right\|_{\mathbf{M}}$.

We close this section by emphasizing the important role played by the first iterations and how the initial perturbations can affect the accuracy of the solution. Notice that the perturbation terms included refer to all the k steps, not just the latest one. Relaxation strategies that start with a low accuracy and gradually increase it are unlikely to work for GKB and other algorithms with similar error minimization properties. Since the first vectors computed are the ones that contribute the most to reducing the error, they should be determined as precisely as possible. Even if we follow a perturbed iteration exclusively by very accurate

ones, this will not prevent the perturbation from being transmitted to all the subsequent vectors, and potentially be amplified by multiplication with matrices and floating-point error. With these observations in mind, we can understand the results in Section 4.

These findings are in line with those concerning other Kylov methods in the presence of inexactness (see Section 11 of the survey by Simoncini and Szyld[?] and the references therein). GKB is not the only method which benefits from lowering the accuracy of the inner process, and the reason why this is possible is linked to the decreasing entries of the coefficient vector.

6 | RELAXATION STRATEGY CHOICES

We have seen in Section 5.1 that we can allow the perturbation norm to increase in a safe way, as long as the process is guided by the decrease of $\left|\hat{\zeta}\right|$. This means that we can adapt the tolerance of the inner solver, such that each call is increasingly cheaper, without compromising the accuracy of the final GKB iterate. Then, at step k we can call the inner solver with a tolerance equal to $\tau/f(\zeta)$. The scalar τ represents a constant chosen as either the target accuracy for the final GKB solution, or something stricter, to counteract possible losses coming from floating-point arithmetic. The function f is chosen based on the considerations described below, with the goal of minimizing the number of inner iterations.

A similar relaxation strategy was used in a numerical study by Bouras and Frayssé[?] to control the magnitude of the perturbation introduced by performing inexact matrix-vector products. They employ Krylov methods with a residual norm minimization property, so the proposed criterion divides the target accuracy by the latest residual norm. In our case, because of the minimization property in Equation (19), we need to use the error norm instead of the residual, since it is the only quantity which is strictly decreasing. Due to the actual error norm being unknown, we rely on approximations found via ζ.

Considering the error characterization of the unperturbed process $\|\mathbf{e}_k\|_{\mathbf{M}}^2 = \sum_{i=k+1}^{n} \zeta_i^2$, we can approximate the error by the first term of the sum, which is the dominant one. However, when starting iteration k we do not know ζ_{k+1}, not even ζ_k, so we cannot choose a tolerance for the inner solver required to compute $\mathbf{u}_k$ based on these. What we can do is predict these values via extrapolation, using information from the known values ζ_{k-1} and ζ_{k-2}. We know that in general $\frac{\beta_k}{\alpha_k} = \frac{\zeta_k}{\zeta_{k-1}}$ acts as a local convergence factor for the $|\zeta|$ sequence. We approximate the one for step k by using the previous one $\frac{\zeta_{k-1}}{\zeta_{k-2}}$. Then, we can compute the prediction $\tilde{\zeta}_k := \zeta_{k-1}\frac{\zeta_{k-1}}{\zeta_{k-2}}$. By squaring the local convergence factor, we get an approximation for ζ_{k+1} as $\tilde{\zeta}_{k+1} := \zeta_{k-1}\left(\frac{\zeta_{k-1}}{\zeta_{k-2}}\right)^2$, which we can use to approximate $\|\mathbf{e}_k\|_{\mathbf{M}}$ and adapt the tolerance of the inner solver.

In practice, we only consider processes which include perturbation, and assume we have no knowledge of the unperturbed values $|\zeta|$. As such, for better readability, we drop the hat notation with the implicit convention that we are referring to values which do include perturbation and use them in the extrapolation rule above.

For some isolated iterations, it is possible that $|\zeta_k| \geq |\zeta_{k-1}|$. This behavior is then amplified through extrapolation, potentially leading to even larger values. In turn, this can cause an increase in the accuracy of the inner solver, following a stricter value for the tolerance parameter $\tau/f(\zeta)$. In Section 5.1, we have shown that there is no benefit in increasing this accuracy. The new perturbation would be smaller in norm, but the error $\|\mathbf{e}_k\|_{\mathbf{M}}$ would be dominated by the previous, larger perturbation. As such, we propose computing several candidate values for the stopping tolerance of the inner solver, and choose the one with maximum value. Since these are only scalar quantities, the associated computational effort is negligible, but the impact of a well-chosen tolerance sequence can lead to significant savings in the total number of inner iterations. The candidate values are:

$$\begin{cases} \text{the value at the previous step,} \\ \tau/\left|\zeta_{k-1}\right|, \\ \tau/\left|\tilde{\zeta}_k\right|, \\ \tau/\left|\tilde{\zeta}_{k+1}\right|. \end{cases} \tag{24}$$

To prevent a limitless growth of the tolerance parameter, we impose a maximum value of 0.1. All these choices are safe in the sense that they do not lead to introduction of perturbations which prevent the outer solver from reaching the target accuracy.

We proceed by testing these relaxations strategies on the problem described in Section 3. The initial tolerance for CG is set to $\tau = 10^{-8}$, one order of magnitude more precise than the one set of GKB. As a baseline for comparison, we first keep the tolerance constant, equal to τ. Then, we introduce adaptivity using $\tau/\left|\zeta_{k-1}\right|$. The third case changes the tolerance according to $\tau/\left|\tilde{\zeta}_{k+1}\right|$, the latter term being a predicted approximation of the current error. Finally, we employ a hybrid approach, where all candidate values in Equation (24) are computed, but only the largest one is used. In the legends of the following plots, these four cases are labeled `Constant`, `Adaptive`, `Predicted`, and `Hybrid`, respectively. To monitor GKB convergence, we track the

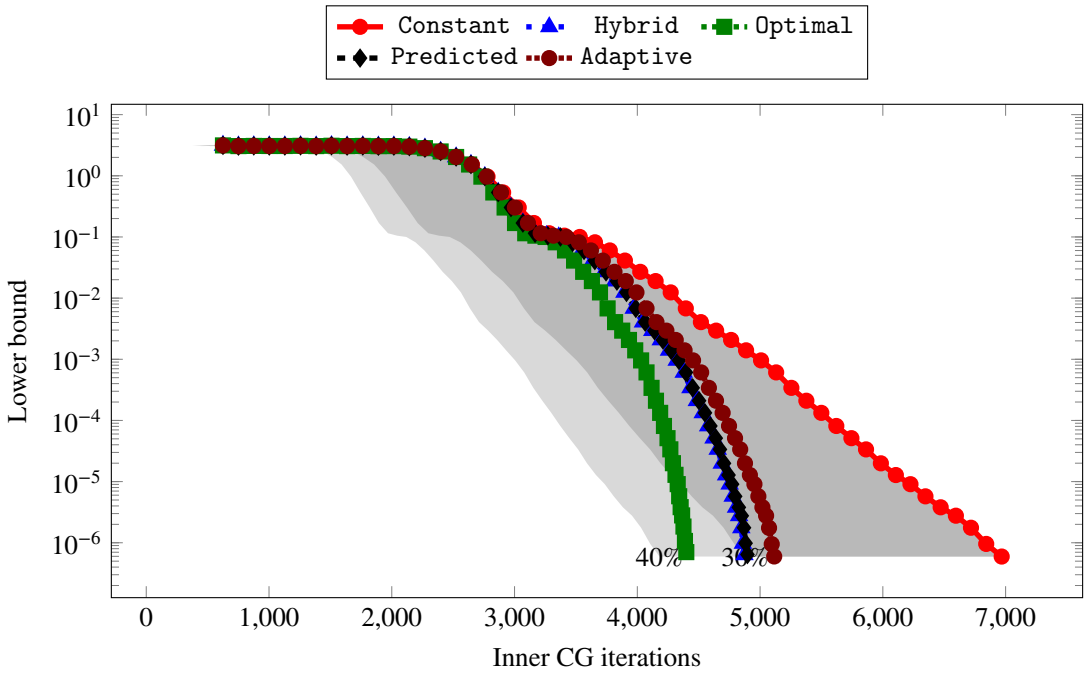

FIGURE 4 Lower bound (Equation (17)) for the error norm associated with the GKB iterates versus the cumulative number of inner CG iterations when solving the original problem from Section 3. The parameter used in `Optimal` is 0.05. See Equations (25) to (29) for the strategies denoted by the labels.

TABLE 1 Reduction of the total number of CG iterations. The CG tolerance is relaxed according to Equations (25) to (29). The parameter in `Optimal` is 0.05.

CG tolerance	`Constant`	`Adaptive`	`Predicted`	`Hybrid`	`Optimal`
CG iterations	6963	5115	4897	4873	4399
Savings %	-	26.54	29.67	30.02	36.82

lower bound for the energy norm of the error corresponding to the primal variable given in Equation (17). For easy reference, all the choices used and their respective labels are given below. We define $\tau = 10^{-8}$.

$$(\texttt{Constant}) : \tau, \tag{25}$$

$$(\texttt{Adaptive}) : \tau/|\zeta_{k-1}|, \tag{26}$$

$$(\texttt{Predicted}) : \tau/|\tilde{\zeta}_{k+1}|, \tag{27}$$

$$(\texttt{Hybrid}) : \max\left\{\tau/|\zeta_{k-1}|, \tau/|\tilde{\zeta}_{k}|, \tau/|\tilde{\zeta}_{k+1}|, \text{previous value}\right\}. \tag{28}$$

$$(\texttt{Optimal}) : \tau/(\textit{parameter}\cdot|\zeta_{k-1}|), \tag{29}$$

Only the last scenario above, `Optimal`, is left to explain. To see if the parameter-free choices can be improved, we run one more case which includes adaptivity by using $|\zeta_{k-1}|$, but also one constant parameter tuned experimentally. This is motivated by the fact that the considerations leading to Equation (24) rely mostly on approximations and inequalities, which means we have an over-estimate of the error. It may be possible to reduce the total number of iterations further, by including an (almost) optimal, problem-dependent constant. The goal is to find a sequence of tolerance parameters with terms that are as large as possible, while guaranteeing the accuracy of the final GKB iterate.

All the results are given in Table 1 and Figure 4. `Hybrid` offers the highest savings among the parameter-free choices (30%), but `Optimal`, the test with the problem-dependent constant, reveals that we can still improve this performance by about 6%.

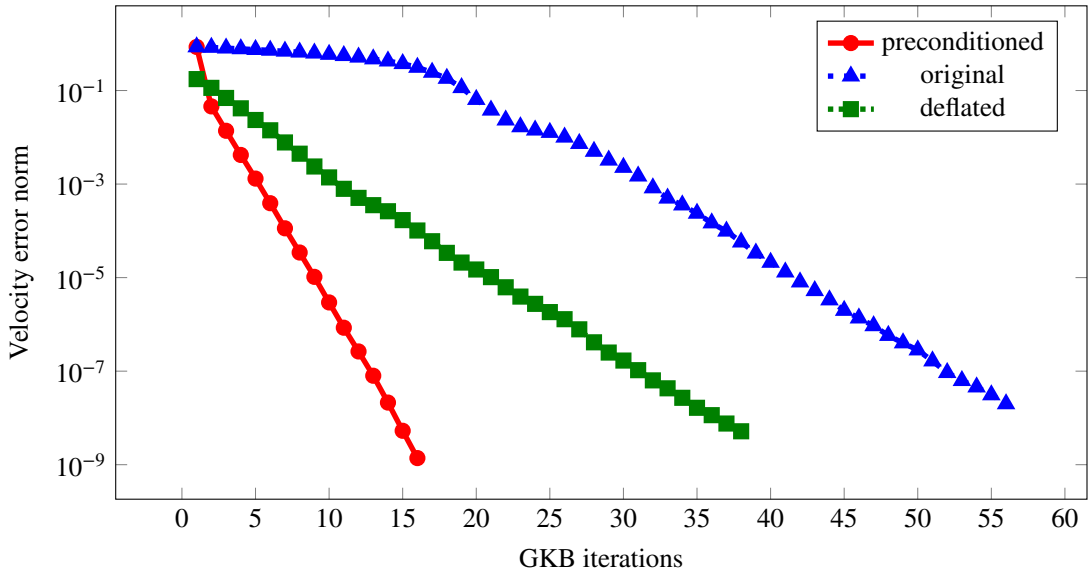

FIGURE 5 GKB convergence curves for the IFISS channel test case before and after spectral clustering. Y-axis: Energy norm of the relative error for the velocity. Target GKB tolerance 10^{-7}. Using the Least Squares Commutator preconditioner or deflation of the smallest five spectral outliers.

6.1 | Increasing the savings by working on a simplified problem

Considering the observations in Section 5.1 and the results plotted in Figure 4, we can significantly reduce the accuracy of the inner solver only when the outer solver is in a superlinear convergence phase, when the $|\zeta|$ sequence decreases rapidly. How much we can relax depends on the slope of the convergence curve. As such, to get the maximum reduction of the total number of iterations, the problem needs to be simplified, such that the convergence curve is as steep as possible and has no plateau. It is common to pair Krylov methods with other strategies, such as preconditioning, in order to improve their convergence behavior. The literature on these kinds of approaches is rich[?????]. The following tests quantify how beneficial is the interaction between our proposed relaxation scheme and these other strategies.

It has been shown by Arioli and Orban that the GKB applied to the saddle-point system is equivalent to the CG algorithm applied to the Schur complement equation[?, Chapter 5]. As such, the first step towards accelerating GKB is to consider the Schur complement, defined as $\mathbf{S} := \mathbf{A}^T\mathbf{M}_{-1}\mathbf{A}$, especially its spectrum. Ideally, a spectrum with tightly clustered values and no outliers leads to rapid GKB convergence[?]. To get as close as possible to this clustering we use the following two methods to induce positive changes in the spectrum: preconditioning with the Least Squares Commutator (LSC)[?] and eigenvalue deflation. Each of them operates differently and leads to convergence curves with different traits.

In Figure 5, we plot the GKB convergence curve for each of these, using a direct inner solver. The LSC aligns the small values in the spectrum with the main cluster and brings everything closer together. The corresponding GKB convergence curve has no plateau and is much steeper than the curve for the unpreconditioned case. Using deflation, we remove the five smallest values from the spectrum, which constitute outliers with the respect to the main cluster. The other values remain unchanged. As such, its convergence curve no longer has the initial plateau, but is otherwise the same as in the original problem.

For both of these cases we apply the same strategies of relaxing the inner tolerance, to see how many total CG iterations we can save. The rest of the set-up is identical to that described for Table 1. We tabulate the results in Tables 2 and 3 and plot them in Figures 6 and 7. They highlight that the best parameter-free results are obtained when using `Hybrid`, which leads to savings of about 50%, depending on the specific case. When comparing this parameter-free approach to `Optimal`, which includes an experimental constant, we find that the hybrid approach can still be improved. Nonetheless, the difference in CG iterations savings is not very high (up to 6%), which supports the idea that our proposed strategy is efficient in a general-use setting. An additional observation pertaining to the plots is that even if convergence is relatively fast (Figure 6) or slow (Figure 7), the final savings are still around 50%, as long as there is no plateau.

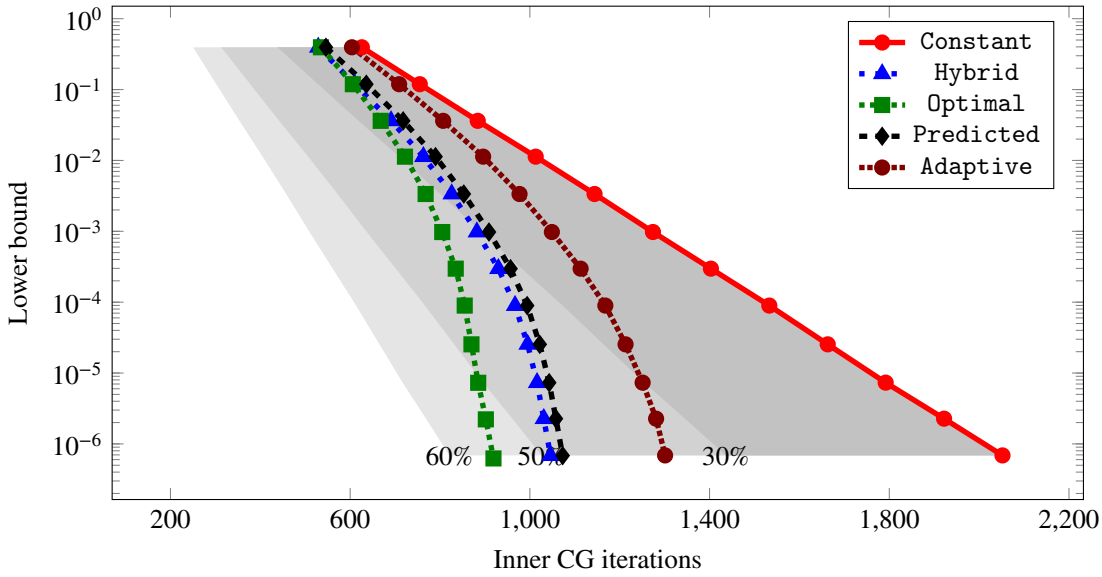

FIGURE 6 Lower bound (Equation (17)) for the error norm associated with the GKB iterates versus the cumulative number of inner CG iterations when solving the problem from Section 3. The problem includes preconditioning with the LSC. The parameter used in `Optimal` is 0.007. See Equations (25) to (29) for the strategies denoted by the labels.

TABLE 2 Reduction of the total number of CG iterations after using the LSC preconditioner. The CG tolerance is relaxed according to Equations (25) to (29). The parameter used in `Optimal` is 0.007.

CG tolerance	`Constant`	`Adaptive`	`Predicted`	`Hybrid`	`Optimal`
CG iterations	2052	1301	1073	1046	919
Savings %	-	36.60	47.71	49.03	55.21

TABLE 3 Reduction of the total number of CG iterations after using deflation. The CG tolerance is relaxed according to Equations (25) to (29). The parameter used in `Optimal` is 0.09.

CG tolerance	`Constant`	`Adaptive`	`Predicted`	`Hybrid`	`Optimal`
CG iterations	4830	2625	2416	2411	2110
Savings %	-	45.65	49.98	50.08	56.31

7 | GKB WITH THE AUGMENTED LAGRANGIAN APPROACH

The method of the AL has been used successfully to speed up the convergence of the GKB algorithm[?], with this effect being theoretically explained by Arioli et al.[?]. Maybe most striking is the potential to reach mesh-independent convergence, provided that the augmentation parameter is large enough. Another use of the AL method is to transform the (1,1)-block of a saddle-point system, say $\mathbf{W}$, from a positive semi-definite matrix to a positive definite one. However, this can happen only if the off-diagonal block $\mathbf{A}$ is full rank or, more generally, if $\ker(\mathbf{W}) \cap \ker(\mathbf{A}^T) = \{\mathbf{0}\}$.

Let $\mathbf{N} \in \mathbb{R}^{n\times n}$ be a symmetric, positive definite matrix. For a given symmetric, positive semi-definite matrix $\mathbf{W} \in \mathbb{R}^{m\times m}$, we can transform it into a positive-definite one by

$$\mathbf{M} := \mathbf{W} + \mathbf{A}\mathbf{N}^{-1}\mathbf{A}^T. \tag{30}$$

The upper right-hand side term $\mathbf{g}$ then becomes

$$\mathbf{g} := \mathbf{g} + \mathbf{A}\mathbf{N}^{-1}\mathbf{r}. \tag{31}$$

With these changes in place, we can proceed to using the GKB algorithm, as described in Section 2.

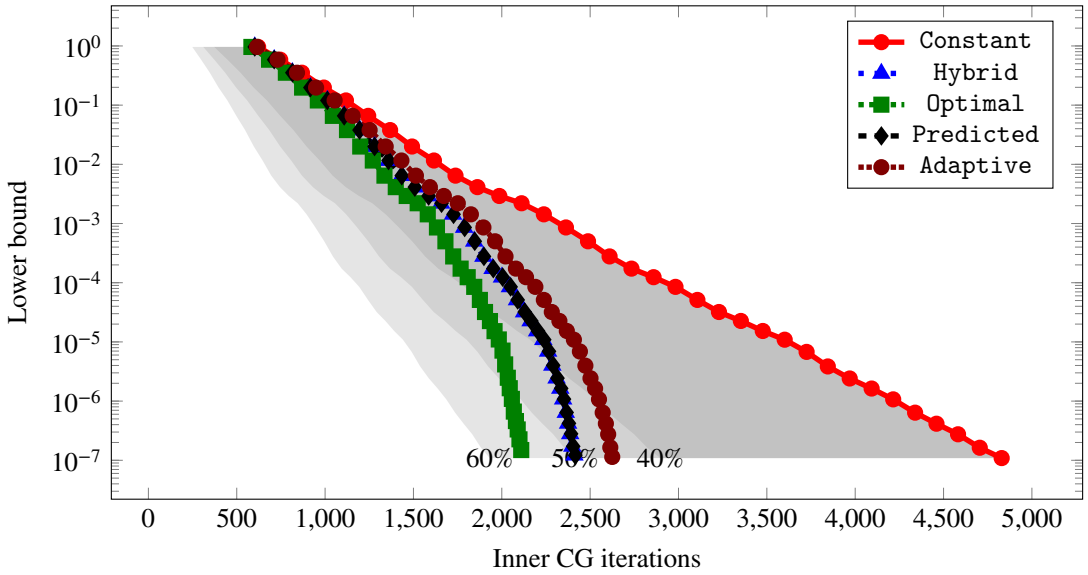

FIGURE 7 Lower bound (Equation (17)) for the error norm associated with the GKB iterates versus the cumulative number of inner CG iterations when solving the problem from Section 3. The problem includes deflation of the five smallest spectral outliers. The parameter used in `Optimal` is 0.09. See Equations (25) to (29) for the strategies denoted by the labels.

TABLE 4 Reduction of the total number of CG iterations after using the AL ($\eta = 1000$). The CG tolerance is relaxed according to Equations (25) to (29). The parameter used in `Optimal` is 0.005.

CG tolerance	`Constant`	`Adaptive`	`Predicted`	`Hybrid`	`Optimal`
CG iterations	2601	1886	1707	1661	1647
Savings %	-	27.49	34.37	36.14	36.68

Note that if the matrix $\mathbf{W}$ is already symmetric positive-definite, the transformation of the (1,1)-block is not necessary for using the GKB method. However, the application of the AL approach does lead to a better conditioning of the Schur complement, which significantly improves convergence speed[?] . As in Section 2, we choose $\mathbf{N} = \frac{1}{\eta}\mathbf{I}$. There is as usual no free lunch: depending on the conditioning of the matrix $\mathbf{A}$ and the magnitude of η, the AL can also degrade the conditioning of the $\mathbf{M}$ matrix as a side-effect.

We test whether the augmentation interacts with the strategies we propose in Section 6, namely if we can still achieve about 50% savings in the total number of inner iterations. The strategies are applied when solving the problem described in Section 3 after an augmentation with a parameter $\eta = 1000$, with the results being given in Table 4 and plotted in Figure 8. Comparing the percentage of iterations saved in this case to those obtained in Section 6, it is clear that, when combined with the AL method, the strategy of variable inner tolerance does help reducing the total number of inner iterations, but by a lower percentage.

Since the AL method modifies the (1,1)-block of the saddle-point system, it changes the difficulty of the inner problem and how many iterations the inner solver needs to perform. As such, a global comparison in terms of number of inner iterations, among all the scenarios we studied (original, preconditioned, deflated, including the AL) is not fair unless the inner problem has the same degree of difficulty for all the cases.

To verify the generality of our method, we also apply it in a different context than that described in Section 3. Let us consider a Mixed Poisson problem. We solve the Poisson equation $-\Delta u = f$ on the unit square $(0,1)^2$ using a mixed formulation. We introduce the vector variable $\vec{\sigma} = \nabla u$. Find $(\vec{\sigma}, u) \in \Sigma \times W$ such that

$$\vec{\sigma} - \nabla u = 0 \tag{32}$$

$$-\mathrm{div}(\vec{\sigma}) = f. \tag{33}$$

where homogeneous Dirichlet boundary conditions are imposed for u at all walls. The forcing term f is random and uniformly drawn in (0, 1). The discretization is done with a lowest order Raviart-Thomas space $\Sigma^h \subset \Sigma$, and a space $W^h \subset W$ containing

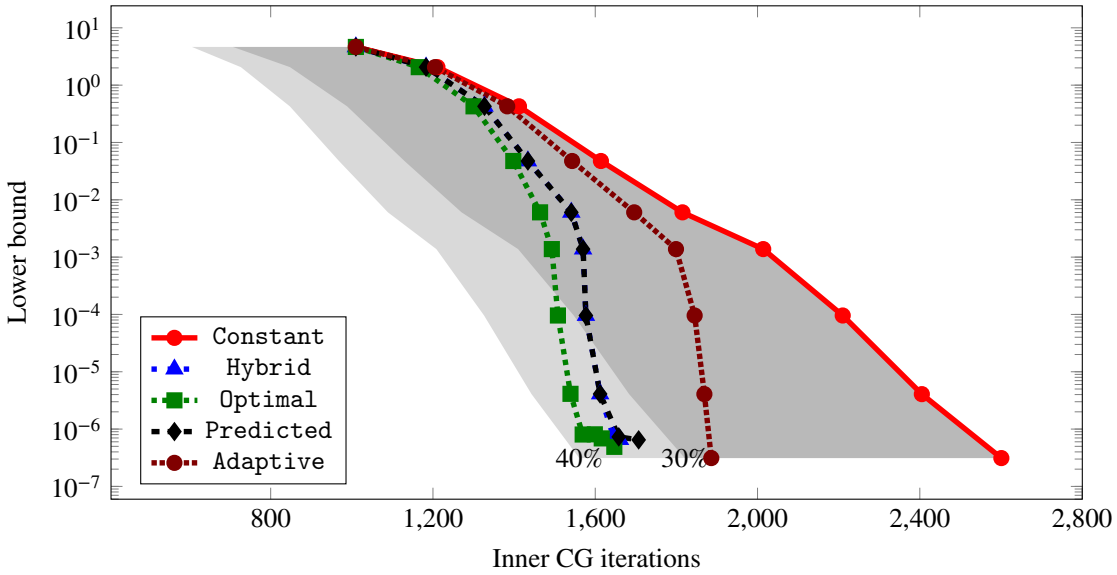

FIGURE 8 Lower bound (Equation (17)) for the error norm associated with the GKB iterates versus the cumulative number of inner CG iterations when solving the problem from Section 3. The problem includes the AL ($\eta = 1000$). The parameter used in `Optimal` is 0.005. See Equations (25) to (29) for the strategies denoted by the labels.

TABLE 5 Reduction of the total number of CG iterations after using the AL ($\eta = 500$) on the Mixed Poisson problem. The CG tolerance is relaxed according to Equations (25) to (28).

CG tolerance	`Constant`	`Adaptive`	`Predicted`	`Hybrid`
CG iterations	10845	4680	4105	4225
Savings %	-	56.84	62.15	61.04

piece-wise constant basis functions. We used the finite element package Firedrake[2] coupled with a PETSc[???] implementation of GKB [3], adapted to include dynamical relaxation, to produce the following numerical results. We used the implementation provided by Firedrake[4]. The test case has 328 192 degrees of freedom, of which 197 120 are associated with the (1,1)-block. The GKB delay parameter is set to 3. The augmentation parameter η is set to 500 and the tolerance for the GKB set to 10^{-5}. The results are presented in Figure 9. We confirm the results presented above with a reduction of over 60% in the total number of inner CG iterations with respect to the constant accuracy set up.

8 | CONCLUSIONS

We have studied the behavior of the GKB algorithm in the case where the inner problem, i.e. the solution of a linear system, is performed iteratively. We have found that the inner solver does not need to be as precise as a direct one in order to achieve a GKB solution of a predefined accuracy.

Furthermore, we have proposed algorithmic strategies that reduce the cost of the inner solver, quantified as the cumulative number of inner iterations. This is possible by selecting criteria to change the stopping tolerance. To motivate these choices, we have studied the perturbation generated by the inexact inner solver. The findings show that the perturbation introduced in early iterations has a higher impact on the accuracy of the solution compared to later ones. We devised a dynamic way of adapting the accuracy of the inner solver at each call to minimize its cost. The initial, high accuracy is gradually reduced, maintaining the resulting perturbation under control.

[2] www.firedrakeproject.org
[3] https://petsc.org/release/docs/manualpages/PC/PCFIELDSPLIT.html#PCFIELDSPLIT
[4] https://www.firedrakeproject.org/demos/saddle_point_systems.py.html

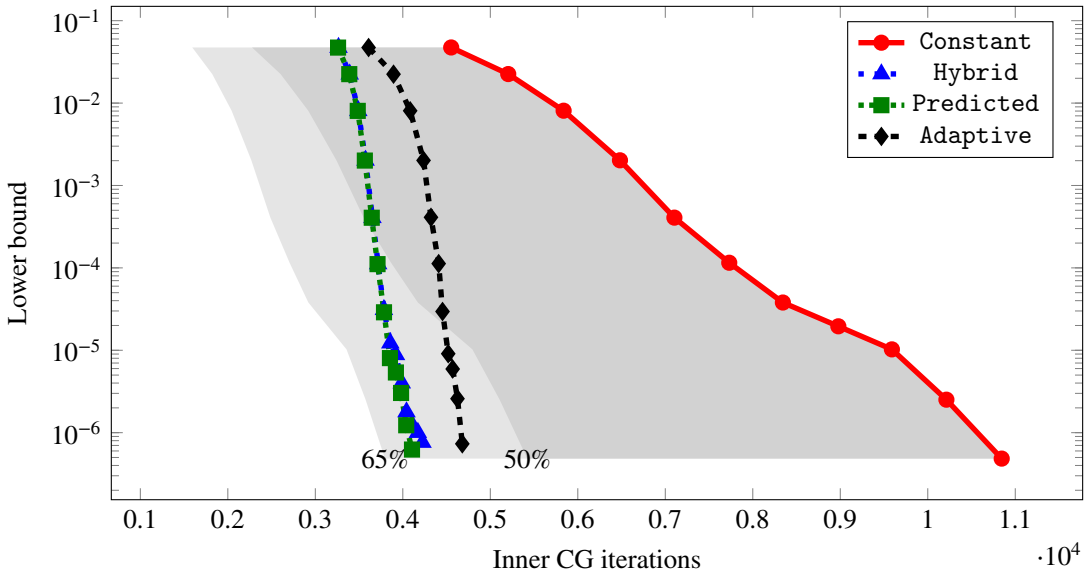

FIGURE 9 Lower bound (Equation (17)) for the error norm associated with the GKB iterates versus the cumulative number of inner CG iterations when solving the Mixed Poisson problem. We also use the AL ($\eta = 500$). See Equations (25) to (28) for the strategies denoted by the labels.

Our relaxation strategy is inexpensive, easy to implement, and has reduced the total number of inner iterations by 33-63% in our tests. The experiments also show that including methods such as deflation, preconditioning and the augmented Lagrangian has no negative impact and can lead to a higher percentage of savings. Another advantage is that our method does not rely on additional parameters and is thus usable in a black-box fashion.

Acknowledgments

The authors thank Mario Arioli for many inspiring discussions and advice.

How to cite this article: V. Darrigrand, A. Dumitrasc, C. Kruse, and U. Rüde (2022), Inexact inner-outer Golub-Kahan bidiagonalization method: a relaxation strategy, *Numerical Linear Algebra with Applications*, *2017;00:1–6*.