DEFLATION FOR THE OFF-DIAGONAL BLOCK IN SYMMETRIC SADDLE POINT SYSTEMS *

A. DUMITRASC†, C. KRUSE‡, AND U. RÜDE*†

Abstract. Deflation techniques are typically used to shift isolated clusters of small eigenvalues in order to obtain a tighter distribution and a smaller condition number. Such changes induce a positive effect in the convergence behavior of Krylov subspace methods, which are among the most popular iterative solvers for large sparse linear systems. We develop a deflation strategy for symmetric saddle point matrices by taking advantage of their underlying block structure. The vectors used for deflation come from an elliptic singular value decomposition relying on the generalized Golub-Kahan bidiagonalization process. The block targeted by deflation is the off-diagonal one since it features a problematic singular value distribution for certain applications. One example is the Stokes flow in elongated channels, where the off-diagonal block has several small, isolated singular values, depending on the length of the channel. Applying deflation to specific parts of the saddle point system is important when using solvers such as CRAIG, which operates on individual blocks rather than the whole system. The theory is developed by extending the existing framework for deflating square matrices before applying a Krylov subspace method like MINRES. Numerical experiments confirm the merits of our strategy and lead to interesting questions about using approximate vectors for deflation.

Key words. Golub-Kahan bidiagonalization, eigenvalue deflation, singular value decomposition, saddle point problems, Stokes equation

MSC codes. 15A18, 35P15, 65F10, 65F15, 65N22

1. Introduction. Advanced simulation tasks often require the numerical solution of partial differential equations (PDE) with fine resolution such that the discretization leads to very large, sparse linear systems. For many practical applications, the computational cost of the linear solvers is among the critical factors, since it can be dominating and a bottleneck in the overall cost of the simulation and thus its speed. While direct methods with their generality and high accuracy often represent the first choice, they may not be efficient or their application may even become unfeasible due to their high computational complexity. As alternative, iterative solvers that successively improve an initial guess of the solution vector can be used. These solvers can be significantly cheaper, also as the iteration can be stopped at any time, when only a moderately precise solution is needed. However, they rarely work well in a black-box manner and have to be tailored to the given matrix properties that may be hard to identify.

One widely used class of iterative solvers are Krylov subspace methods. In the case of algorithms designed for symmetric problems, their speed of convergence is linked to the conditioning or, more precisely, the spectral properties of the system matrix [39, 7]. To obtain fast convergence, it is desirable to have clustered eigenvalues that are bounded away from zero. When this is not the case, the method converges slowly or can even stagnate [30]. The general strategy is then to help the iterative solver by manipulating the linear system, for example by preconditioning. Preconditioning aims to improve the spectral properties of the matrix, such that the linear system becomes easier to solve [7]. Depending on how much one wishes to invest, the preconditioner can be as simple as a diagonal matrix, or as sophisticated as a matrix representing the eigenvectors associated with the smallest eigenvalues. An alternative technique to preconditioning is known as deflation. It is similar, in the sense that it affects the spectrum of the matrix and makes the system easier to solve, by leading to a better clustering of the eigen-

---

†Chair for Computer Science 10 - System Simulation, Friedrich-Alexander University Erlangen-Nuremberg, 91058, Erlangen, Germany (andrei.dumitrasc@fau.de, ulrich.ruede@fau.de).

‡CERFACS, 31057, Toulouse, France (carola.kruse@cerfacs.fr).

values. Unlike preconditioning, it has a more targeted effect, shifting only the outliers such that they are either zero or closer to the main cluster. The rest of the spectrum is left unchanged. Note that this is not the only definition associated with the idea of deflation. For a concise but informative chronological overview of the developments related to deflation we refer to [24]. A notable work describing deflation and augmentation methods under the wider umbrella of recycling is the more recent survey [42], where significant attention is also given to sequences of (shifted) systems. An example of such a Krylov method with recycling is RMINRES, used successfully in an application of electrical impedance tomography (see [31]).

In this paper, we focus on iterative solvers for matrices in saddle point form, as they appear, for example, in the discretization of PDEs in incompressible fluid flow, which is a longstanding research focus in our group [27, 28, 29, 12]. In particular, we focus on fluid flow in a thin-elongated channel domain, where the ratio of length and the height is large, as described by the Stokes equation

$$-\Delta \vec{u} + \nabla p = \vec{f}$$
$$\nabla \cdot \vec{u} = 0.$$

The exact definition, boundary conditions and special properties of our test problem are given in Section 3. Here, we only want to sketch the behavior and difficulty of the chosen iterative solver and the aim of our work. When discretizing the above equation by, e.g., finite differences or a stable class of finite elements, we obtain a linear system with a matrix of 2×2 block structure

$$(1.1) \qquad \begin{bmatrix} \mathbf{W} & \mathbf{A} \\ \mathbf{A}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \mathbf{r} \end{bmatrix},$$

with $\mathbf{W} \in \mathbb{R}^{m \times m}$ being symmetric and positive definite (SPD) and $\mathbf{A} \in \mathbb{R}^{m \times n}, (n < m)$ having full column rank. The vectors are $\mathbf{u} \in \mathbb{R}^m, \mathbf{p} \in \mathbb{R}^n, \mathbf{g} \in \mathbb{R}^m$ and $\mathbf{r} \in \mathbb{R}^n$, with $\mathbf{u}$ and $\mathbf{p}$ representing velocity and pressure, respectively. This saddle point system matrix is indefinite and thus has positive and negative eigenvalues. An overview about solution methods, including *segregated methods* that exploit the block structure of the system and *monolithic methods* that are applied to the system as a whole, is given in [8]. While it is known that, e.g., monolithic multigrid methods can achieve excellent performance for special saddle point problems [26], a block version offers the possibility to re-use existing solvers. In the following work, we are interested in solving the above linear system by Krylov subspace methods. Since the system is symmetric, we could apply as a first choice the well-known *minimum residual* (MINRES) method monolithically. Its convergence curve is characterized by a plateau, i.e., a stage where the residual we monitor at each iteration does not decrease significantly over a certain number of iterations depending on the channel length.

The issue of the plateau has been observed by other authors as well, even for other kinds of Stokes problems (see [34]). In particular, it has been shown that in practice, preconditioning based on the pressure mass matrix clusters most eigenvalues, but not the small outliers. The condition number and solver convergence are visibly improved, but the plateau remains. To tackle this specific issue, the targeted effect of deflation is especially well suited. Effective preconditioners for MINRES can be built by combining ideas from deflation, multigrid and reusing spectral information (see [37]), assuming a setting where the leading block of the matrix is ill-conditioned.

The same tendency to stagnate (see Figure 2a) appears when we use the CRAIG version of the generalized Golub-Kahan bidiagonalization process [2], our solver of choice. This segregated method relies on exploiting the block structure of the indefinite system. In a series of papers [27, 28, 29], the authors have shown that this solver exhibits fast convergence when applied to

certain saddle point systems and in particular to the Stokes problem on the unit square and a rectangular domain. In an attempt to explain however the plateau for the thin-elongated channel flow problem and thus increasingly slow convergence, we observe that the off-diagonal block $\mathbf{A}$ becomes increasingly ill-conditioned. In particular, the conditioning varies with certain physical parameters, such as the ratio between the length and width of the channel. A related analysis shows that this ratio impacts the condition number of the Schur complement [9].

The contribution of this article is the definition of a deflation strategy for the CRAIG iterative solver, that extends the framework in [17]. It has been shown that the CRAIG solver is equivalent to the *Conjugate Gradient* (CG) method on the Schur complement equation (see [35, Chapter 5]). As such, we draw from the existing knowledge about CG and always consider the effects of our procedures on the Schur complement, either implicitly or explicitly. As pointed out above, we can focus on Stokes problems where the condition number itself may not seem problematic, since this can be improved even with simple preconditioners. Instead, we restrict our attention to the presence of outliers in the spectrum, to better isolate the impact of deflation.

Following [17], we define the operators necessary for deflation and the subsequent correction. Through projection, a certain subspace can be removed from the range of the matrix. This subspace is chosen to be the one that the iterative solver has difficulties operating with. The resulting deflated system is solved quickly, yielding a part of the solution to the initial problem. The complete solution is found through a correction, by which we reintroduce the information initially removed by deflation. The basis for the deflated subspace is typically the set of eigenvectors corresponding to the outliers in the spectrum. The basis itself can be different, so long as the vectors span the required subspace. As such, the projectors can also be built by other means, such as in [33], where multigrid and domain decomposition approaches are discussed.

Our aim is now to deflate the Schur complement implicitly, without constructing it explicitly. As such, we will not employ its eigendecomposition, but rather the related elliptic singular value decomposition of the off-diagonal block (see [2]). Although we define deflation using exact information about the decomposition, in practice it is rarely the case that this is available. As such, we also study the implications of using approximate vectors. For computing this information in an iterative way, we extend two already existing algorithms. The first is the Augmented Lanczos Bidiagonalization by Baglama and Reichel [4]. Our extension relies primarily on replacing the original, standard inner-products such that the resulting algorithm realizes the generalized Golub-Kahan bidiagonalization [2]. Following this change, the output is a set of approximate elliptic singular triplets. For an augmented (not generalized) Golub-Kahan bidiagonalization in the context of the LSQR solver where deflation is carried out using harmonic Ritz values, see [5]. The second algorithm we extend is inspired by eigCG, which was defined by Stathopoulos and Orginos [43]. Their version solves a linear system with CG and, at the same time, uses the vectors that the solver builds to compute approximate eigenpairs (also known as Ritz pairs). With the same objectives, we adapt the CRAIG algorithm of Arioli [2] such that, as it solves a saddle point system, it also computes approximate elliptic singular triplets.

One alternative choice for tackling saddle point systems is to treat the matrix as a single unit and solve the associated system with MINRES [8]. It has been shown that CRAIG needs two times less iterations to converge than MINRES, when the latter is paired with a particular block diagonal preconditioner, which makes the two algorithms comparable in terms of cost per iteration [2]. We perform numerical experiments to test whether this 2:1 ratio still holds if deflation is applied to both algorithms.

Throughout the paper, we will illustrate theoretical and practical aspects by making use of two test problems chosen for their simplicity and their specific features. Both represent a kind of Stokes flow in a long and thin rectangular channel, but with different definitions and discretizations. One of them is designed by us for the purpose of exposing the algorithmic

features, while the other is created with the software package Incompressible Flow & Iterative Solver Software [1] (IFISS).

The paper is structured as follows. In Section 2, we extend the deflation framework presented in [17] to the case of saddle point systems, operating with the individual blocks of the matrix. This deflation is achieved using exact spectral information, following an elliptic singular value decomposition. Next, in Section 3, we introduce our test problems, which we use in the remainder of the article for exemplification. We consider some theoretical consequences of deflation using approximate spectral information in Section 4. Sections 5 and 6 are devoted to algorithms for computing the elliptic singular value decomposition. In Section 7, we numerically compare deflated MINRES and deflated CRAIG. We also briefly compare deflation and preconditioning using the pressure mass matrix in Section 8, before closing with the conclusions in Section 9.

2. Deflation definition. In this section, we introduce the steps to be taken when applying deflation to a saddle point problem, as well as the necessary prerequisites. In addition to the description of the two main steps of projection and correction, we now provide the formal setting and definition. The main tool we will need is the elliptic singular value decomposition (see [2]). For the time being, we assume the case where this decomposition provides us with exact information, which is subsequently applied via deflation.

Since we require $\mathbf{A}$ to have full column rank, the system in Equation (1.1) has a unique solution. Our solver of choice is CRAIG (given in Appendix A), which relies on the generalized Golub-Kahan bidiagonalization introduced by Arioli [2]. In particular, we are interested in accelerating its convergence for a class of problems to be described in Section 3. As noted in [35, Chapter 5], there is an equivalence between using CRAIG on Equation (1.1) and using CG on the Schur complement equation

$$(2.1) \qquad \mathbf{A}^T \mathbf{W}^{-1} \mathbf{A} \mathbf{p} = \mathbf{A}^T \mathbf{W}^{-1} \mathbf{g} - \mathbf{r},$$

with $\mathbf{S} = \mathbf{A}^T \mathbf{W}^{-1} \mathbf{A}$ being the Schur complement. According to the literature (see e.g. [3]), CG's convergence depends on the eigenvalue distribution of the matrix it operates on, in this case $\mathbf{S}$. Clustered values are associated with fast convergence, whereas scattered values, particularly the presence of outliers, lead to slow convergence. Removing these small outliers from the spectrum via deflation can improve the convergence behavior, as seen in [33]. This represents our core strategy: we employ deflation to achieve a better eigenvalue clustering for the Schur complement, which translates to faster CRAIG convergence. The challenge is to do this without building $\mathbf{S}$ explicitly.

One way to achieve this goal is as follows. By performing a Cholesky decomposition of the (1,1)-block, we have $\mathbf{W} = \mathbf{L}\mathbf{L}^T$, $\mathbf{W}^{-1} = \mathbf{L}^{-T}\mathbf{L}^{-1}$ and define

$$(2.2) \qquad \tilde{\mathbf{A}} = \mathbf{L}^{-1}\mathbf{A}, \quad \mathbf{y} = \mathbf{L}^T \mathbf{u}, \quad \mathbf{f} = \mathbf{L}^{-1}\mathbf{g}.$$

We can express the Schur complement equation (2.1) as a normal equation

$$(2.3) \qquad \tilde{\mathbf{A}}^T \tilde{\mathbf{A}} \mathbf{p} = \tilde{\mathbf{A}}^T \mathbf{f} - \mathbf{r}.$$

There is a connection between the singular value decomposition of $\tilde{\mathbf{A}}$ and the eigendecomposition of $\mathbf{S}$: the right singular vectors of $\tilde{\mathbf{A}}$ are the eigenvectors of $\mathbf{S}$ and the singular values are the square roots of the eigenvalues. Then, we can implicitly deflate $\mathbf{S}$ by relying on $\tilde{\mathbf{A}}$.

An alternative to using the SVD of $\tilde{\mathbf{A}}$ follows an equivalence given in [2]. Note that the author considers a more general definition of $\tilde{\mathbf{A}}$ as $\tilde{\mathbf{A}} = \mathbf{W}^{-\frac{1}{2}}\mathbf{A}\mathbf{N}^{-\frac{1}{2}}$, for a symmetric and positive definite matrix $\mathbf{N} \in \mathbb{R}^{n \times n}$.

Let the following be a partial elliptic (generalized) SVD of $\mathbf{A}$:

(2.4)
$$\mathbf{AV} = \mathbf{WU\Sigma}, \quad \mathbf{A}^T\mathbf{U} = \mathbf{V\Sigma}, \quad \mathbf{U}^T\mathbf{WU} = \mathbf{I}_k, \quad \mathbf{V}^T\mathbf{V} = \mathbf{I}_k,$$

with $k < n$ triplets, and $\mathbf{I}_k$ being the $k \times k$ identity matrix. Additionally, $\mathbf{A} = \mathbf{WU\Sigma V}^T$ only when $k = n$. The columns of $\mathbf{V} \in \mathbb{R}^{n \times k}$ are the right elliptic singular vectors, the columns of $\mathbf{U} \in \mathbb{R}^{m \times k}$ are the left elliptic singular vectors, and the entries of the diagonal matrix $\mathbf{\Sigma} \in \mathbb{R}^{k \times k}$ are the elliptic singular values. Note that the left vectors are not orthonormal in the Euclidean sense, but with respect to an inner product and norm defined by $\mathbf{W}$

$$\langle \mathbf{u}_i, \mathbf{u}_j \rangle_{\mathbf{W}} = \mathbf{u}_i^T \mathbf{W} \mathbf{u}_j = 0, \quad \forall i \neq j,$$
$$\|\mathbf{u}\|_{\mathbf{W}} = \sqrt{\langle \mathbf{u}_i, \mathbf{u}_j \rangle_{\mathbf{W}}} = 1.$$

The equivalence given in [2] states that the elliptic SVD (ESVD) of $\mathbf{A}$ is linked to the SVD of $\tilde{\mathbf{A}}$: the values and the right vectors of both decompositions coincide. For the left vectors we have $\mathbf{U} = \mathbf{L}^{-T}\tilde{\mathbf{U}}$, with $\mathbf{U}$ being the left elliptic singular vectors of $\mathbf{A}$ and $\tilde{\mathbf{U}}$ being the left singular vectors of $\tilde{\mathbf{A}}$. Finally, considering this equivalence with that concerning $\tilde{\mathbf{A}}$ and $\mathbf{S}$, it follows that we can deflate the Schur complement implicitly, by referring to the ESVD of $\mathbf{A}$.

Given Equation (2.4), a partial ESVD of $\mathbf{A}$ with k triplets, we define the approximate left-inverse of $\mathbf{A}$ as

(2.5)
$$\mathbf{M} = \mathbf{V\Sigma}^{-1}\mathbf{U}^T,$$

the $\mathbf{W}-$orthogonal projector

(2.6)
$$\mathbf{P} = \mathbf{I}_m - \mathbf{AM},$$

and the (Euclidean) orthogonal projector

(2.7)
$$\mathbf{Q} = \mathbf{I}_n - \mathbf{MA}.$$

Then, we define the deflated system as

(2.8)
$$\begin{bmatrix} \mathbf{W} & \mathbf{AQ} \\ \mathbf{Q}^T\mathbf{A}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{u}} \\ \hat{\mathbf{p}} \end{bmatrix} = \begin{bmatrix} \mathbf{g} \\ \mathbf{Q}^T\mathbf{r} \end{bmatrix},$$

whose solution we correct by

(2.9)
$$\mathbf{p} = \mathbf{Q}\hat{\mathbf{p}} + \mathbf{Mg} - \mathbf{MWM}^T\mathbf{r}, \quad \mathbf{u} = \mathbf{P}^T\hat{\mathbf{u}} + \mathbf{M}^T\mathbf{r},$$

to retrieve the solution of the original problem in Equation (1.1).

Deflation differs from preconditioning in one important way. Since $rank(\mathbf{Q}) = n - k$, applying $\mathbf{Q}$ does not preserve the solution set, i.e. the systems in Equation (2.8) and Equation (1.1) are not equivalent, making the correction in Equation (2.9) necessary. The correction terms add information complementary to that found by solving the deflated system.

It is also possible to define a deflated system similar to that in Equation (2.8) by including the action of $\mathbf{P}$, but it can be shown that the resulting Schur complement is the same as that of Equation (2.8). With $\mathbf{P} \in \mathbb{R}^{m \times m}, \mathbf{Q} \in \mathbb{R}^{n \times n}$ and $m > n$, it is preferable to avoid applying $\mathbf{P}$ whenever possible, due to its size.

THEOREM 2.1. *Applying deflation as in Equation* (2.8) *and correction as in Equation* (2.9) *leads to the solution of the initial saddle point system Equation* (1.1).

Proof. The proof follows by inserting the corrected solution from Equation (2.9) into the initial saddle point system in Equation (1.1) and making use of the equalities in Equation (2.8), the definitions in Equations (2.5)–(2.7) and the properties of the projectors

$$(2.10) \qquad \mathbf{PW} = \mathbf{WP}^T; \quad \mathbf{PA} = \mathbf{AQ}; \quad \mathbf{QQ} = \mathbf{Q}. \qquad \square$$

We conclude this section with a note on alternative choices of information to be used in defining deflation. The intuitive one is to leverage information about the spectrum via a decomposition, as we have shown above. Another choice would be using multigrid operators (see [33]). If we have a left basis $\mathbf{U}$ ($\mathbf{W}$-orthogonal) and a right one $\mathbf{V}$ orthogonal) which are not sets of elliptic singular vectors, then we set $\boldsymbol{\Sigma} = \mathbf{U}^T \mathbf{A} \mathbf{V}$. If we additionally require that $\mathbf{AV} = \mathbf{WU}\boldsymbol{\Sigma}$, then the resulting $\mathbf{M}, \mathbf{P}$ and $\mathbf{Q}$ will satisfy the conditions in Equation (2.10). As such, following Theorem 2.1, deflation and correction will find the solution of the initial saddle point system. The choice of $\mathbf{U}$ and $\mathbf{V}$ is free, as long as their columns span the intended subspaces, i.e. those spanned by the elliptic singular vectors corresponding to the values that we intend to deflate.

3. Deflation effects on two test problems. In this section, we introduce the saddle point test cases in order to motivate our choices regarding deflation. Our focus is on cases that exhibit slow convergence, such as systems arising from the discretization of incompressible flow problems in elongated domains. It is known that increasing the ratio between the width and length of the domain degrades the conditioning of the Schur complement [9]. We will see that this is due to more small outliers that appear in the spectrum, which also drift away from the main cluster of values. We explore different kinds of discretization and geometries. We analyze the spectra associated with these problems and see how deflation leads to faster solver convergence.

3.1. A simplified model of flow in a 1D channel. The Stokes system in its continuous form is given by

$$(3.1) \qquad -\Delta \vec{u} + \nabla p = \vec{f}$$
$$(3.2) \qquad \nabla \cdot \vec{u} = 0.$$

The first problem is motivated by Stokes flow, but is deliberately constructed to highlight the phenomenon of stagnation in a Krylov method via a simple model system. To this end, we consider transport in a long and thin channel, as illustrated in Figure 1. The discrete system results from the classical Marker-and-Cell (MAC) finite difference scheme [25]. Here the flow enters a channel from the left and leaves at the right. The channel is modeled by a collection of n cells in the horizontal direction, but only two cells vertically. As illustrated in Figure 1, only the horizontal velocities v_i^t and v_i^b, $i = 0, 1, \ldots, n$ are used. They represent the velocities at the mid-edge position of the top and bottom cells, respectively.

The boundary conditions are set to represent an inflow on the left end. Specifically, we impose an inflow only through the upper half of the boundary, $v_0^t = 1, v_0^b = 0$. Similarly, outflow is enforced on the right channel end, but only through the lower cell $v_n^t = 0, v_n^b = 1$. This setup is chosen to avoid trivial solutions. The velocity values corresponding to these Dirichlet type boundary conditions can be eliminated and will then appear in the right hand side vector of the reduced linear system. The discrete equations represent a simplified form of momentum conservation taking the form

$$(3.3) \qquad \left. \begin{array}{rcl} -v_{i-1}^t + 4v_i^t - v_{i+1}^t - v_i^b & = & 0 \\ -v_{i-1}^b + 4v_i^b - v_{i+1}^b - v_i^t & = & 0 \end{array} \right\}, \text{ for } i = 1, \ldots, n-1$$

Note that this setup assumes implicitly that the top and bottom walls are characterized by a no-slip boundary condition. Consequently, the resulting velocity block $\mathbf{W}$ of the saddle point
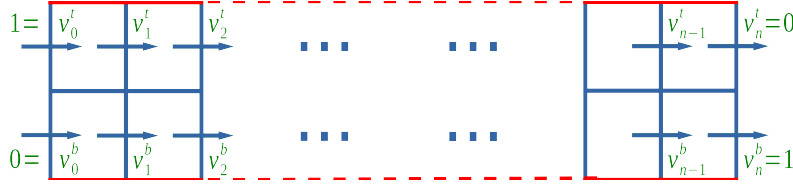
Fig. 1: 1D model of an elongated channel, with inflow only through the upper-left cell (v_0^t) and outflow only through the lower-right cell (v_n^b).

system in Equation (1.1) becomes strictly diagonally dominant and well-conditioned. Using the theorem of Gerschgorin, it is easy to see that its eigenvalues are bounded below by 1, and its condition number is better than 7, independent of the number of equations n.

Next, the lower block row ($[\mathbf{A}^T \ \mathbf{0}]$ in Equation (1.1)) enforces mass conservation for each pair of vertical cells, by requiring

$$(3.4) \qquad v_i^t - v_{i-1}^t + v_i^b - v_{i-1}^b \ = \ 0, \quad \text{for } i = 1, \ldots, n$$

The resulting system is as in Equation (1.1). Given this construction, $\mathbf{A}$ would be rank-deficient, reflecting the fact that the pressure is determined only up to a constant. To remedy this in the discrete system, we additionally replace the first constraint (on $i = 1$) by a convex combination between itself and the last constraint and the latter is dropped from the system. Once we include the treatment of the boundary conditions and constraints as described above, and we take a row-wise ordering of the unknowns, the entries of the system from Equation (1.1) are

$$(3.5) \qquad \mathbf{W} = \left[\begin{array}{cccc|cccc}
4 & -1 & & & -1 & & & \\
-1 & 4 & -1 & & & -1 & & \\
& -1 & 4 & \ddots & & & -1 & \\
& & \ddots & \ddots & & & & \ddots \\
\hline
-1 & & & & 4 & -1 & & \\
& -1 & & & -1 & 4 & -1 & \\
& & -1 & & & -1 & 4 & \ddots \\
& & & \ddots & & & \ddots & \ddots
\end{array}\right], \quad
\mathbf{A} = \left[\begin{array}{cccc|cccc}
0.5 & -1 & & & & & & \\
& 1 & -1 & & & & & \\
& & 1 & \ddots & & & & \\
-0.5 & & & \ddots & & & & \\
\hline
0.5 & -1 & & & & & & \\
& 1 & -1 & & & & & \\
& & 1 & \ddots & & & & \\
-0.5 & & & \ddots & & & &
\end{array}\right],$$

$$(3.6) \qquad \mathbf{u} = \begin{bmatrix} v_1^t \\ \vdots \\ v_{n-1}^t \\ \hline v_1^b \\ \vdots \\ v_{n-1}^b \end{bmatrix}, \quad
\mathbf{p} = \begin{bmatrix} p_1 \\ \vdots \\ p_{n-1} \end{bmatrix}, \quad
\mathbf{g} = \begin{bmatrix} v_0^t = 1 \\ \vdots \\ v_n^t = 0 \\ \hline v_0^b = 0 \\ \vdots \\ v_n^b = 1 \end{bmatrix}, \quad
\mathbf{r} = \begin{bmatrix} 1 \\ \vdots \\ 0 \end{bmatrix}.$$

In the limit $n \to \infty$, the channel grows to infinite length, while its width stays constant, or, alternatively, it can be interpreted as staying at constant length but its width shrinking to zero. Thus, the linear system is not a discretization of a differential equation on a fixed domain. However, such a change of the domain geometry has an impact on the *inf-sup* condition or equivalently, on the smallest eigenvalue of the Schur complement (see [9]), leading it to approach zero.
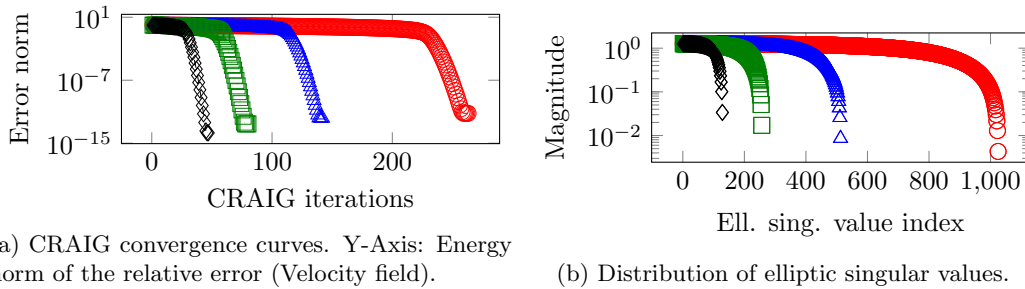
(a) CRAIG convergence curves. Y-Axis: Energy norm of the relative error (Velocity field).

(b) Distribution of elliptic singular values.

Fig. 2: Corresponding spectra and CRAIG convergence curves for four 1D channels of lengths 128 - ◇ -, 256 □ , 512 △ and 1024 –⊖–.

Note also that the solution of this particular saddle point system becomes trivial with a change of variables, involving the sum and difference of each vertical pair of velocity variables, as a simple analogue to the stream-function-vorticity formulation of the Stokes problem (see [23]).

Here we do not transform the variables accordingly, but use the original definition of the variables to exhibit and illustrate a typical CRAIG convergence pattern. When used to solve the initial system from Equation (1.1), a stagnation in error reduction occurs. This convergence plateau lasts for a number of iterations roughly equal to a quarter of the channel's length, as shown in Figure 2a. Only when the end of the plateau is reached, fast convergence occurs to then reach the solution quickly. As such, simply by modifying the length of the channel, we can adapt the difficulty of the problem.

3.1.1. Applying deflation on the 1D channel problem. Since the CRAIG algorithm is equivalent to that of the Conjugate Gradient (CG) applied to the Schur complement equation, we can learn more about CRAIG's convergence behavior by analogy with CG. We know from [21] that CG's convergence rate depends on the condition number of the system matrix. For us, this means studying the eigenvalue distribution of the Schur complement or $\mathbf{A}$'s elliptic singular values, as defined in Section 2.

In Figure 2b, we illustrate the elliptic spectra of a 1D channel problem as described above, where the channel length varies between 128 and 1024. The largest values are well clustered and away from the origin, but the smallest stray from this cluster. These outliers get more numerous and closer to zero as the channel length increases. In the following considerations, we focus on the case where this length is 512. If the smallest ten values were deflated, then the effective condition number, which uses only the nonzero values, would be smaller and CRAIG would converge in fewer steps. We consider this strategy analogously to the known behavior of CG [33].

A more detailed look into the problem comes from analyzing the elliptic singular vectors as well, answering the question "Which values and vectors do we really need to deflate?". It has been shown in [40] that, at each iteration, CRAIG seeks to minimize the error norm of the primal variable $\mathbf{u}$ in Equation (1.1), which also holds for the generalized version we use (see [2]). We consider the initial error $\tilde{\mathbf{e}}_1$, i.e. the difference between the true solution and the first iterate computed by CRAIG. By expressing this initial error as a linear combination of the left elliptic singular vectors $\mathbf{U}$ of $\mathbf{A}$,

$$(3.7) \qquad\qquad\qquad \mathbf{U}\mathbf{z} = \tilde{\mathbf{e}}_1,$$

(a) All entries of **z**.

(b) Only entries with magnitudes above 10^{-7}.

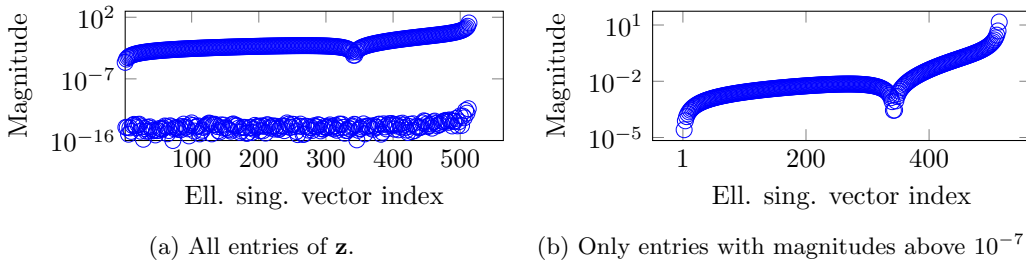Fig. 3: Entries of the **z** vector (see Equation (3.7)), showing how much each left elliptic singular vector contributes to the initial error.1D channel problem (length=512). Vectors listed in descending order of the corresponding elliptic singular values.

we can see which of them contribute to this error and how many of them have a large coefficient in **z**. These vectors are then good candidates for deflation, since removing them from the problem reduces the error significantly. Deflating vectors which have small (near zero) coefficients in **z** represents an unnecessary expense, as we argue and illustrate below. See [30] for a similar analysis, but regarding the eigencomponents of the initial error for CG.

We return to our 1D channel problem, apply the ideas described above and plot the entries of **z** in Figure 3. Note that in the following examples, we take a descending ordering of the elliptic singular values and of their vectors. As such, the first entries correspond to the most dominant vectors, i.e. those associated with the largest elliptic singular values. The rightmost entries correspond to the vectors associated with the smallest elliptic singular values.

Figure 3a shows that only odd-numbered left elliptic singular vectors are part of the error, while the coefficients of even-numbered ones are close to machine-zero. A black-box eigensolver typically delivers vectors corresponding to consecutive values. As such, if we tried to deflate a group of k vectors from this problem, only every second of them would actually be necessary.

In Figure 3b, we only look at the meaningful entries of **z**, those above a threshold of 10^{-7}. It is clear that the highest entries are to the right, which implies that the vectors contributing the most to the error belong to the smallest elliptic singular values. If we choose to deflate these vectors and their corresponding elliptic singular values, CRAIG is certain to converge faster. While it is common in the literature to focus on deflating the vectors associated with the smallest values, we stress that it is important to first analyze whether these vectors have a significant contribution to the error. That is, one could express the initial error or residual as in Equation (3.7), and check whether the corresponding entries in **z** are the largest, as we have seen was the case for the problem considered above (see Figure 3b). If the error components mentioned above are not the largest, the plateau will appear at a later moment, after an initial stage of faster convergence. For such problems, deflation may not be necessary, if we only need solutions of limited accuracy, to which we can converge before the onset of the plateau stage.

We conclude this subsection with a convergence plot (Figure 4) showing how deflation helps in the case of this 1D channel test case. We compare deflating the smallest 10, 50 and 100 values and take the undeflated problem as reference. Unsurprisingly, deflating more values leads to faster convergence. While it is unrealistic to deflate so many of them, the performance is predictable once we look at the spectrum in Figure 2b and see how the remaining values are distributed. We note that even deflating ten values can be very helpful, especially if we only need a solution of moderate precision.
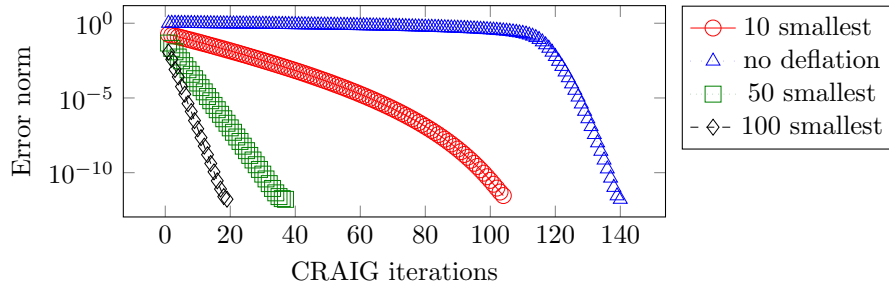
Fig. 4: Convergence curves of the CRAIG algorithm for the 1D channel problem after having deflated a number of elliptic singular values of $\mathbf{A}$. Y-Axis: Energy norm of the relative error (Velocity field).

3.2. IFISS channel problem. This test case is generated using the Incompressible Flow & Iterative Solver Software [2] (IFISS) package. The particular scenario that we construct is once again that of a long and thin channel domain $\Omega = [-1, L] \times [-1, 1]$, with $L > 1$, potentially $L \gg 1$. This time, however, we employ the Finite Elements discretization Q2-Q1.

The steady Stokes problem here is the Poiseuille flow, with the following boundary conditions: a Dirichlet condition on the inflow (left boundary, $\Gamma_{in} = \{-1\} \times [-1, 1]$), a characteristic boundary with zero velocity above and below, also known as no-slip condition ($\Gamma_c = [-1, L] \times \{-1\} \cup [-1, L] \times \{1\}$), while the outflow (right, $\Gamma_{out} = \{L\} \times [-1, 1]$) is described by the Neumann condition

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

$$\frac{\partial u_y}{\partial x} = 0.$$

For this particular problem definition, we have the analytical solution

(3.8)
$$\begin{cases} u_x = 1 - y^2, \\ u_y = 0, \\ p = -2x + \text{constant}. \end{cases}$$

See [13] for a more detailed description of this reference Stokes problem.

3.2.1. Applying deflation on the IFISS channel problem. As in Subsection 3.1, when solving this problem with CRAIG, we can see in Figure 5a how the convergence curve starts with a plateau having a length depending on the length of the channel.

The elliptic spectrum in Figure 5b shows how, for different channel lengths, the main cluster of values is contained within the same interval. Only the smallest values become outliers, getting more numerous and further away from the main cluster. This latter feature explains why the problem gets increasingly more difficult to solve, with a longer plateau. The first feature, concerning the cluster, explains why the convergence curve after the plateau is the same in all three cases. We focus only on the channel of length 20 and will consider the five smallest values as candidates for deflation.

We also take a look at the elliptic singular vectors, and how present they are in the initial error for this particular problem. Their coefficients in the linear combination (Equation (3.7))
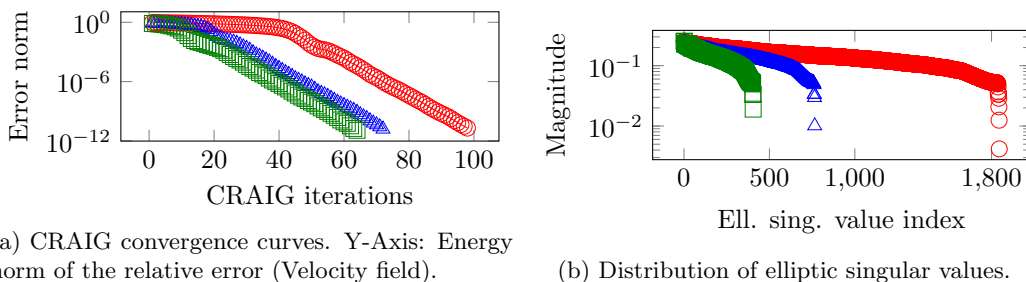
(a) CRAIG convergence curves. Y-Axis: Energy norm of the relative error (Velocity field).

(b) Distribution of elliptic singular values.

Fig. 5: Corresponding spectra and CRAIG convergence curves for three IFISS channels of lengths 10 □ , 20 △ and 50 ─⊖─.

(a) All entries of **z**.
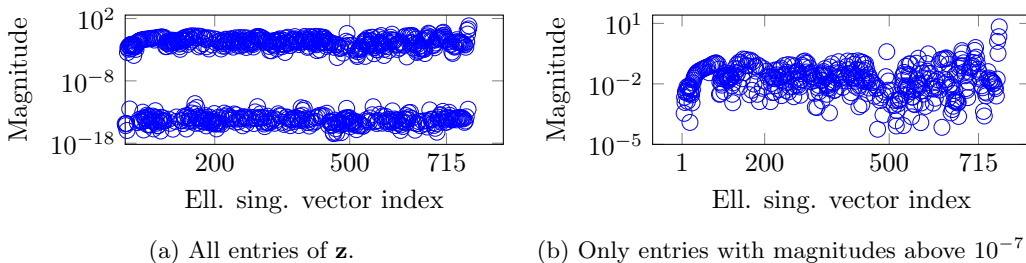
(b) Only entries with magnitudes above 10^{-7}.

Fig. 6: Entries of the **z** vector (see Equation (3.7)), showing how much each left elliptic singular vector contributes to the initial error. IFISS channel problem (length=20). Vectors listed in descending order of the corresponding elliptic singular values.

are plotted in Figure 6. We can see in Figure 6a that many coefficients are close to machine-zero, including that corresponding to the vector associated with the fourth smallest elliptic singular value. In Figure 6b, where we focus only on the entries greater than 10^{-7}, we notice how the rightmost coefficients are larger than the rest, supporting the idea of deflating the vectors of the smallest elliptic singular values.

Simply from analyzing the spectrum in Figure 5b and the coefficients in Figure 6b, we can tell that removing the five smallest values should reduce the condition number significantly, accelerating CRAIG's convergence. Deflating more values however, is not expected to bring much additional improvement, since the rest of the values are already well clustered. We test these assumptions and deflate first five, then 50 values, plotting CRAIG's convergence history in Figure 7. Deflating five values completely removed the plateau. Deflating 50 reduced the number of iterations more than the case of five, but not by much. This is understandable if we consider that in Figure 6a, more than half of the values after index 715 are already 0, as if they had already been deflated. The rest are nonzeros, but are small nonetheless, not contributing much to the error. Deflation sets them to zero and leads to the additional gains shown in Figure 7.

4. Deflation using approximate elliptic singular triplets. We have seen how deflation using exact information about the spectrum of the problem can be beneficial by leading to a better spectral clustering and reducing the number of iterations necessary for convergence.
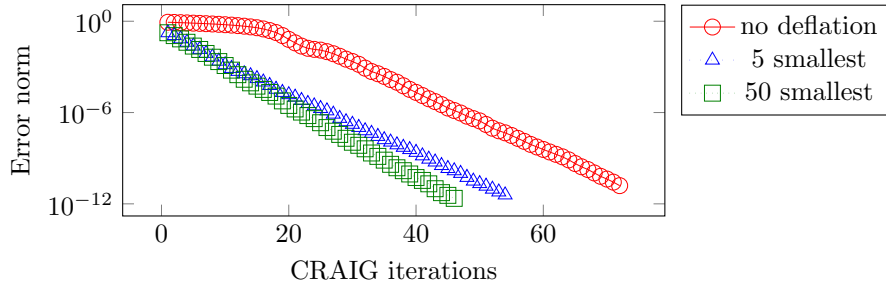
Fig. 7: Convergence curves of the CRAIG algorithm for the IFISS channel problem after having deflated a number of elliptic singular values. Y-Axis: Energy norm of the relative error (Velocity field).

Finding such exact information, however, can prove to be prohibitively expensive. It is often the case that one seeks a solution of moderate accuracy and uses an iterative method with a fairly high tolerance parameter. It seems unreasonable then to invest so much in finding exact spectral information, only to accelerate finding an inexact solution to the linear system. Luckily, we can show that this is not necessary and that satisfactory results can be achieved even with approximate knowledge of the deflation vectors. We analyze the spectrum of the deflated matrix to see how it changes when we vary the accuracy of the deflation vectors. Then, we compare the numerical performance of this approximate deflation with that of its exact counterpart.

In the literature, the effects of using approximate deflation have been studied for particular definitions of the deflation operator and choice of linear solver. The spectrum of the deflated matrix and its influence on the convergence behavior of GMRES has been analyzed in [41]. In [18], the authors examine spectral preconditioners, including deflation and coarse grid correction, and how the resulting spectrum changes as a function of perturbation. Subdomain deflation combined with variants of the Incomplete Cholesky decomposition as preconditioner and their effect on the spectrum are discussed in [45].

In this paper, we are interested in how approximate spectral information affects the theory presented in Section 2. To understand where the inexactness originates from, let us review the stages of computing the SVD, assuming exact arithmetic.

To find the SVD of a general matrix, three steps are required: a reduction to bidiagonal form, an SVD (diagonalization) of the resulting bidiagonal matrix and a multiplication of the matrices from the two previous steps. One algorithm used for the first step is that of Lanczos [4]. If we need the singular vectors to be orthogonal in a non-Euclidean way, we need to replace Lanczos's algorithm with the generalized Golub-Kahan bidiagonalization [2], which redefines the inner products by using SPD matrices other than the identity. In particular, we require $\mathbf{W}$-orthogonal left vectors. With this change, what we find are the elliptic singular triplets [2].

In the following relations, we will use the subscript b to denote matrices coming from the first stage (the bidiagonalization), and the subscript s for those coming from the second one (the SVD). The final matrices, used to build the deflation operators $\mathbf{P}$, $\mathbf{Q}$ and $\mathbf{M}$ have no subscript.

We focus on approximate elliptic singular triplets resulting from a decomposition which includes a partial generalized Golub-Kahan bidiagonalization. Assume we stop the bidiagonalization after $\eta < n$ steps. Then, for $\mathbf{A} \in \mathbb{R}^{m \times n}$, we have $\mathbf{U}_b \in \mathbb{R}^{m \times \eta}$, $\mathbf{V}_b \in \mathbb{R}^{n \times \eta}$, $\mathbf{B} \in \mathbb{R}^{\eta \times \eta}$, with

$$(4.1) \qquad \mathbf{A}\mathbf{V}_b = \mathbf{W}\mathbf{U}_b\mathbf{B}, \quad \mathbf{A}^T\mathbf{U}_b = \mathbf{V}_b\mathbf{B}^T + \mathcal{E}, \quad \mathbf{U}_b^T\mathbf{W}\mathbf{U}_b = \mathbf{I}_\eta, \quad \mathbf{V}_b^T\mathbf{V}_b = \mathbf{I}_\eta.$$

We refer to $\mathcal{E}$ as the residual term, which vanishes only when $\eta = n$ (and in exact arithmetic), leading to $\mathbf{A} = \mathbf{W}\mathbf{U}_b\mathbf{B}\mathbf{V}_b^T$. More importantly, when $\eta = n$ the spectrum of $\mathbf{B}$ is the same as that of $\mathbf{A}$. However, if $\eta < n$, then it is only an approximation which improves as $\eta \to n$ and/or if the bidiagonalization includes a restarting mechanism. Following [4], we can further describe this term as

$$(4.2) \qquad \mathcal{E} = \mathbf{r}_\eta \mathbf{e}_\eta^T,$$

where $\mathbf{r}_\eta$ is the residual vector used to continue the bidiagonalization: $\beta_{\eta+1} = \|\mathbf{r}_\eta\|_2$, $\mathbf{v}_{\eta+1} = \mathbf{r}_\eta/\beta_{\eta+1}$ and $\mathbf{e}_\eta \in \mathbb{R}^\eta, \mathbf{e}_\eta = \{0, ..., 0, 1\}^T$; i.e., $\mathbf{e}_\eta$ is taken to be the last vector of the standard basis for $\mathbb{R}^{\eta \times \eta}$.

As stated before, to get our approximate elliptic singular triplets, we perform the SVD of $\mathbf{B}$, keep only the k *target* triplets

$$(4.3) \qquad \mathbf{B}\mathbf{V}_s = \mathbf{U}_s\mathbf{\Sigma}, \quad \mathbf{V}_s^T\mathbf{V}_s = \mathbf{I}_k, \quad \mathbf{U}_s^T\mathbf{U}_s = \mathbf{I}_k.$$

Above, *target* denotes which triplets we aim to deflate: either the first (largest elliptic singular values) or the last (smallest).

The SVD matrices are then multiplied with those from the partial bidiagonalization

$$(4.4) \qquad \mathbf{U} = \mathbf{U}_b\mathbf{U}_s, \quad \mathbf{V} = \mathbf{V}_b\mathbf{V}_s.$$

The approximate elliptic SVD (ESVD) is then

$$(4.5) \qquad \mathbf{A}\mathbf{V} = \mathbf{W}\mathbf{U}\mathbf{\Sigma}, \quad \mathbf{A}^T\mathbf{U} = \mathbf{V}\mathbf{\Sigma} + \mathcal{E}\mathbf{U}_s, \quad \mathbf{V}^T\mathbf{V} = \mathbf{I}_k, \quad \mathbf{U}^T\mathbf{W}\mathbf{U} = \mathbf{I}_k.$$

As a side note, we remark that it is also possible to have an error term $\mathbf{E}$ such that $\mathbf{A}\mathbf{V} = \mathbf{W}\mathbf{U}\mathbf{\Sigma} + \mathbf{E}$, if we consider a different kind of partial bidiagonalization, or simply other sources of error. While they are not considered in this present work, such cases represent interesting directions for further research.

For the residual term in Equation (4.5), we obtain

$$(4.6) \qquad \mathcal{E}\mathbf{U}_s = \beta_{\eta+1}\mathbf{v}_{\eta+1}\mathbf{e}_\eta^T\mathbf{U}_s.$$

It is known that the last components of the eigenvectors of the Lanczos tridiagonal matrix can be used to track progress when approximating eigenpairs of a symmetric, positive definite matrix (see [22, 36]). Given the link with the bidiagonalization, this observation can be used for the SVD as well, as done in [4], leading to a stopping criterion for the iterative procedure needed to compute such approximate triplets. The GK bidiagonalization we use here is a generalization, but as seen above, it is still true that the residual term depends on the last row of $\mathbf{U}_s$. With more restarts and/or a larger subspace generated by the GK bidiagonalization, this residual term diminishes.

We have seen how a partial bidiagonalization leads to a residual term in the ESVD. We turn now to the deflated matrix $\mathbf{P}\mathbf{A}$, where $\mathbf{P} = \mathbf{I}_m - \mathbf{W}\mathbf{U}\mathbf{U}^T$, to see how it changes when relying on approximate spectral information.

$$(4.7) \qquad \mathbf{P}\mathbf{A} = \mathbf{A} - \mathbf{W}\mathbf{U}\mathbf{U}^T\mathbf{A} = \mathbf{A} - \mathbf{W}\mathbf{U}(\mathbf{\Sigma}\mathbf{V}^T + \mathbf{U}_s^T\mathcal{E}^T)$$

$$(4.8) \qquad = \mathbf{A} - \mathbf{W}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T - \mathbf{W}\mathbf{U}\mathbf{U}_s^T\mathcal{E}^T$$

The partial bidiagonalization also affects the term $\mathbf{W}\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$. If the deflation vectors were exact, we would shift k *target* values to 0, while leaving the rest as they were. Neither effect can be counted on in the presence of inexactness, here due to approximating the elliptic singular

triplets of $\mathbf{A}$ from the small bidiagonal matrix $\mathbf{B}$. The spectrum will include k zeros, but not necessarily from deflating the intended values. Also, the other values may suffer perturbations. Both effects diminish as the accuracy of the deflation vectors increases. If $\eta = n$ and we assume exact arithmetic, we do indeed get the intended triplets. In general, neither condition holds and we only get an approximation that improves as $\eta \to n$, or via a restarted algorithm. These changes to the spectrum then propagate, leading linear solvers to converge differently as we will explore numerically in Section 5.

5. Computing the elliptic singular triplets. In this section, we describe the algorithms by which one can find an approximate elliptic SVD (ESVD) and provide numerical illustration of the points brought up in Section 4 regarding the influence of the eigensolver on the linear solver with respect to accuracy. The first approach to computing the ESVD (Algorithm 5.1) uses already existing tools while the second approach (Algorithm 5.2) relies on a reformulation of the generalized Golub-Kahan bidiagonalization algorithm, not as part of a linear solver, but as part of an eigensolver.

In principle, the ESVD of $\mathbf{A}$ differs from a regular SVD only in the requirement that the left singular vectors should be $\mathbf{W}$-orthogonal, instead of Euclidean. As such, we can "integrate" this target space of orthogonality into the problem, and then compute the SVD. According to the description in Section 2, we have the following Algorithm 5.1.

Algorithm 5.1 Elliptic Singular Value Decomposition 1

Input: $\mathbf{L}$ - Cholesky factor of $\mathbf{W}$, lower triangular matrix; $\mathbf{A}$ - rectangular matrix.
Output: $\mathbf{U}$ - left elliptic singular vectors; $\mathbf{\Sigma}$ - elliptic singular values; $\mathbf{V}$ - right elliptic singular.

1: Compute a (partial) SVD of $\mathbf{L}^{-1}\mathbf{A}$: get left singular vectors $\tilde{\mathbf{U}}$, singular values $\mathbf{\Sigma}$ and right singular vectors $\mathbf{V}$.
2: Transform: $\mathbf{U} = \mathbf{L}^{-T}\tilde{\mathbf{U}}$

This algorithm relies on the ability to factorize $\mathbf{W}$. If that is the case, it enables us to find the ESVD, and also reuse these factors when we call CRAIG to solve the linear system. If the Cholesky decomposition cannot be performed, Algorithm 5.3 is a viable alternative.

The second approach of finding a partial, approximate ESVD follows the stages described in Section 4. It is a nested iteration, where the outer algorithm (Algorithm 5.2) carries out a partial bidiagonalization (Algorithm 5.3) at every step. As seen in Section 4, the accuracy of the ESVD depends to a significant extent on the bidiagonalization, i.e. on how well the spectrum of $\mathbf{B}$ approximates that of $\mathbf{A}$. Due to memory limitations, the bidiagonalization is not performed only once with n steps, but rather several times, with $\eta < n$ steps, in a restarted fashion. The key here is that with each restart, the bidiagonalization improves the results of the previous one via an augmentation mechanism, which allows it to explore a new portion of the search space.

To define our algorithms, we extend the developments presented in [4]. The strategy presented there is to augment the bidiagonalization search space at each restart with the current approximation of the k elliptic singular triplets. From the second call on, the bidiagonalization generates $\eta - k - 1$ triplets, from which information is extracted by a subsequent SVD to refine the existing k elliptic singular triplets. The presentation in [4] uses the Lanczos bidiagonalization, while we need the generalized GK algorithm, in order to have $\mathbf{W}$- orthogonal left vectors.

We note that, owing to the symmetry of $\mathbf{W}$ and $\mathbf{I}$, it is possible for the bidiagonalization to build sets of vectors that are $\mathbf{W}$- and Euclidean orthogonal using short recurrences. As such, after each restart, only the first computed vector $\mathbf{u}$ needs to be explicitly orthogonalized against all those in $\mathbf{U}$ (Line 9). The subsequent ones then reference only the immediate predecessor,

---

Algorithm 5.2 Elliptic Singular Value Decomposition 2

Input: $\mathbf{A}$ - rectangular matrix, $\mathbf{W}$ - SPD matrix, $\mathbf{v}$ - starting vector, η - maximum search subspace dimension, k - number of desired triplets, *target* - choose between *smallest* or *largest* triplets, *tol* - tolerance, *max_iter* - maximum number of iterations.
Output: $\mathbf{U}$ - approximate left elliptic singular vectors, $\mathbf{\Sigma}$ - approximate elliptic singular values, $\mathbf{V}$ - approximate elliptic singular right vectors.

1: Call Algorithm 5.3 with a starting vector $\mathbf{v}$, $\mathbf{u}_0 = \mathbf{0}$, and passing $\mathbf{A}, \mathbf{W}$, and η. Get $\mathbf{U}_b, \mathbf{B}, \mathbf{V}_b$ and residual $\mathbf{r}$.
2: **for** $i = 1 : max\_iter$ **do**
3:    Compute the SVD of $\mathbf{B}$ for the k *target* triplets; store them in $\mathbf{U}_s, \mathbf{\Sigma}_i$ and $\mathbf{V}_s$.
4:    $\mathbf{U}_i = \mathbf{U}_b \mathbf{U}_s;$ $\mathbf{V}_i = \mathbf{V}_b \mathbf{V}_s;$ $\beta = \|\mathbf{r}\|_2;$
5:    $\hat{\mathbf{u}} = $ the last row of $\mathbf{U}_s;$ $\mu = \max(|\hat{\mathbf{u}}|);$ $\sigma = $ the first entry of $\mathbf{\Sigma}_i;$
6:    **if** $(\beta\mu)/\sigma < tol$ OR $i = max\_iter$ **then**
7:       **break**
8:    **end if**
9:    $\mathbf{v} = \mathbf{r}/\beta;$ $\mathbf{V}_i = [\mathbf{V}_i \ \mathbf{v}];$ $\mathbf{u} = \mathbf{A}\mathbf{v};$ $\mathbf{d} = \mathbf{U}_i^T \mathbf{u};$ $\mathbf{u} = \mathbf{W}^{-1}\mathbf{u} - \mathbf{U}_i \mathbf{d};$ $\mathbf{\Sigma}_i = [\mathbf{\Sigma}_i \ \mathbf{d}]$
   $\alpha = \|\mathbf{u}\|_{\mathbf{W}};$ $\mathbf{\Sigma}_i = \begin{bmatrix} \mathbf{\Sigma}_i \\ \alpha\mathbf{e}_{k+1}^T \end{bmatrix};$ $\mathbf{u} = \mathbf{u}/\alpha;$ $\mathbf{U}_i = [\mathbf{U}_i \ \mathbf{u}].$
10:    Compute residual $\mathbf{v} = \mathbf{A}^T\mathbf{u} - \alpha\mathbf{v};$
11:    Reorthogonalization: $\mathbf{v} = \mathbf{v} - \mathbf{V}_i \mathbf{V}_i^T \mathbf{v};$
12:    $\beta = \|\mathbf{v}\|_2;$ $\mathbf{u}_0 = $ the last column of $\mathbf{U}_i;$
13:    Call Algorithm 5.3 with starting vector $\mathbf{v}$, $\eta - k - 1$ steps and passing $\mathbf{A}, \mathbf{W}$ and $\mathbf{u}_0$. Optionally, pass $\mathbf{U}_i$ and $\mathbf{V}_i$ for additional reorthogonalization. Get $\mathbf{U}_b, \mathbf{B}, \mathbf{V}_b$ and residual $\mathbf{r}$.
14:    Extend $\mathbf{U}_b = [\mathbf{U}_i \ \mathbf{U}_b]$, $\mathbf{V}_b = [\mathbf{V}_i \ \mathbf{V}_b]$ and $\mathbf{B} = \begin{bmatrix} \mathbf{\Sigma}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{bmatrix}$. Set $\mathbf{B}_{k+1,k+2} = \beta$.
15: **end for**
16: $\mathbf{U} = \mathbf{U}_i;$ $\mathbf{V} = \mathbf{V}_i;$ $\mathbf{\Sigma} = \mathbf{\Sigma}_i;$

---

while staying orthogonal to all the previous ones.

The bidiagonalization is augmented in Line 3. The first time, Algorithm 5.3 is called with $\mathbf{u}_0 = \mathbf{0}$ (Line 1), since there is no information to augment the bidiagonalization with. Subsequent calls also send the last column of the matrix $\mathbf{U}$ (the current approximate left singular vectors), as $\mathbf{u}_0$ which implicitly represents the augmentation. Thanks to the short recurrences mentioned above, we do not need the entire matrix, and even the last column is only referenced once. This way, the augmentation in Algorithm 5.3 has only a negligible cost, while ensuring that each call finds new information used to refine $\mathbf{U}, \mathbf{\Sigma}$ and $\mathbf{V}$ in the outer iteration (Algorithm 5.2).

The stopping condition in Algorithm 5.2 depends on the term $\mathcal{E}\mathbf{U}_s$ from (4.5), computed in a simpler way, based on scalars, as in [4]. The quantity $\beta\mu \to 0$ as the iterations progress and σ is used in order to achieve a relative, problem-specific value, since it approximates the largest elliptic singular value. This corresponds to what in [46] is referred to as eigenresidual. Unfortunately, our tests, like those in [43] and [46], show that the eigenresidual's Euclidean norm does not decrease monotonically, but rather in an irregular way. This happens predominantly in the beginning of the iterative process, stabilizing afterwards. Such irregularities would make us think that the approximations do not improve in quality every iteration, making it difficult to know when to stop the algorithm if we only need rough approximations. However, the theory about the convergence of the restarted Lanczos process in [1] shows that the error decreases monotonically. Our tests, as well as those in [43] and [46] support this finding. The implication is

Algorithm 5.3 Restarted Augmented Golub-Kahan Bidiagonalization

Input: $\mathbf{A}$ - rectangular matrix; $\mathbf{W}$ - SPD matrix; $\mathbf{v}$ - right starting vector; $\mathbf{u}_0$ - left starting vector; η - number of steps.

   Optional, for additional reorthogonalization: $\mathbf{U}$ - approximate left elliptic singular vectors; $\mathbf{V}$ - approximate right elliptic singular vectors.

Output: $\mathbf{U}_b$ - left vectors ($\mathbf{W}$-orthogonal); $\mathbf{B}$ - upper bidiagonal matrix; $\mathbf{V}_b$ - right vectors (orthogonal); $\mathbf{r}$ - residual vector

1: $\beta_1 = \|\mathbf{v}\|_2;$ $\mathbf{v}_1 = \mathbf{v}/\beta_1;$ $\mathbf{V}_1 = \mathbf{v}_1;$ $\mathbf{u} = \mathbf{W}^{-1}\mathbf{A}\mathbf{v}_1;$
2: **if** $\mathbf{u}_0 \neq \mathbf{0}$ **then**
3:    $\mathbf{W}$-orthogonalization: $\mathbf{u} = \mathbf{u} - \beta_1\mathbf{u}_0;$
4: **end if**
5: **if** $\mathbf{U}$ is given **then**
6:    $\mathbf{W}$-reorthogonalization: $\mathbf{u} = \mathbf{u} - \mathbf{U}\mathbf{U}^T\mathbf{W}\mathbf{u};$
7: **end if**
8: $\alpha_1 = \|\mathbf{u}\|_{\mathbf{W}};$ $\mathbf{B}_1 = \alpha_1;$ $\mathbf{u}_1 = \mathbf{u}/\alpha_1;$ $\mathbf{U}_1 = \mathbf{u}_1;$
9: **for** $j = 1 : \eta$ **do**
10:    $\mathbf{v} = \mathbf{A}^T\mathbf{u}_j - \alpha_j\mathbf{v}_j;$
11:    Reorthogonalization: $\mathbf{v} = \mathbf{v} - \mathbf{V}_j\mathbf{V}_j^T\mathbf{v};$
12:    **if** $\mathbf{V}$ is given **then**
13:       Reorthogonalization: $\mathbf{v} = \mathbf{v} - \mathbf{V}\mathbf{V}^T\mathbf{v};$
14:    **end if**
15:    **if** $j < \eta$ **then**
16:       $\beta_{j+1} = \|\mathbf{v}\|_2;$ $\mathbf{v}_{j+1} = \mathbf{v}/\beta_{j+1};$ $\mathbf{V}_{j+1} = [\mathbf{V}_j \; \mathbf{v}_{j+1}];$
17:       $\mathbf{B}_{j+1} = [\mathbf{B}_j \; \beta_{j+1}\mathbf{e}_j]$ $\mathbf{u} = \mathbf{W}^{-1}\mathbf{A}\mathbf{v}_{j+1} - \beta_{j+1}\mathbf{u}_j;$
18:       $\mathbf{W}$-reorthogonalization: $\mathbf{u} = \mathbf{u} - \mathbf{U}_j\mathbf{U}_j^T\mathbf{W}\mathbf{u};$
19:       **if** $\mathbf{U}$ is given **then**
20:          $\mathbf{W}$-reorthogonalization: $\mathbf{u} = \mathbf{u} - \mathbf{U}\mathbf{U}^T\mathbf{W}\mathbf{u};$
21:       **end if**
22:       $\alpha_{j+1} = \|\mathbf{u}\|_{\mathbf{W}};$ $\mathbf{u}_{j+1} = \mathbf{u}/\alpha_{j+1};$ $\mathbf{U}_{j+1} = [\mathbf{U}_j \; \mathbf{u}_{j+1}];$ $\mathbf{B}_{j+1} = \begin{bmatrix} \mathbf{B}_{j+1} \\ \alpha_{j+1}\mathbf{e}_{j+1}^T \end{bmatrix};$
23:    **end if**
24: **end for**
25: $\mathbf{r} = \mathbf{v};$ $\mathbf{V}_b = \mathbf{V}_j;$ $\mathbf{U}_b = \mathbf{U}_j;$ $\mathbf{B} = \mathbf{B}_j;$

that the eigenresidual's Euclidean norm may not be the best stopping criterion for Algorithm 5.2 and that it may not accurately reflect the quality of the approximations.

The reorthogonalization steps in Algorithm 5.3 and Algorithm 5.2 could be necessary in practice, as a result of finite-precision computations. In Algorithm 5.3, we can reorthogonalize with respect to local vectors ($\mathbf{U}_j$ and $\mathbf{V}_j$) computed during that call and/or those computed previously, representing the current approximations for the elliptic singular vectors ($\mathbf{U}$ and $\mathbf{V}$). How this is carried out exactly is a problem dependent choice, but the authors of [4] and [6] provide useful suggestions regarding partial or full reorthogonalization. We can choose to work with $\mathbf{U}_j$ and/or $\mathbf{V}_j$ or a subset of their columns, in order to fine-tune the relationship between them. An important observation is that, when targeting the smallest values, the computations are more sensitive to round-off errors, so full reorthogonalization is the safest choice, in order to maximize the accuracy of the results. For simplicity, we used the Gram-Schmidt procedure, but more robust choices are also possible. As we will see in the numerical tests, deflation does not need exact information in order to be effective. Thus, if we compute our deflation vectors

Table 1: Errors of the approximated elliptic singular triplets and perturbations of the spectrum after deflation with triplets from Algorithm 5.2.

ESVD iterations	median and max error (targeted vectors)	median and max relative error (targeted values)	median and max relative perturbation (other values)
10	[5E-1, 1]	[1, 2]	[1E-5, 4E-2]
26	[2E-4, 1E-2]	[1E-6, 7E-2]	[4E-12, 4E-2]
30	[2E-8, 6E-7]	[1E-15, 1E-10]	[4E-16, 8E-11]

via Algorithm 5.2, we may skip some or all of the reorthogonalization steps, provided that the problem is not very ill-conditioned. See [19] for a study comparing the loss of orthogonality between vectors computed by several variants of the Gram-Schmidt method, as well as [38], where this behavior is considered for non-Euclidean orthogonality.

Next, we analyze one of the test cases given in Section 3, to see how the spectrum looks like after deflation with approximate vectors and how this changes the way our linear solver, CRAIG, converges. We use the problem of the 1D channel to illustrate the following points. We run Algorithm 5.2 for 10, 26 and 30 iterations and use the resulting triplets (the 10 smallest) for deflation. The error and perturbation ranges are given in Table 1. Let $\mathcal{I}$ be the set of indices denoting the values we want to deflate and $\overline{\mathcal{I}}$ its complement. We use the superscript $*$ to refer to exact triplets. Then, the error for the deflated vectors is computed as $\left\| \mathbf{v}_i^* - \mathbf{V}\mathbf{V}^T \mathbf{v}_i^* \right\|_2, i \in \mathcal{I}$, the relative error for the deflated values is $|(\sigma_i - \sigma_i^*)/\sigma_i^*|, i \in \mathcal{I}$, and the relative perturbation of the undeflated values is $|(\sigma_i - \sigma_i^*)/\sigma_i^*|, i \in \overline{\mathcal{I}}$.

By construction, the deflated matrix will have a spectrum containing $k = 10$ zeros. Depending on the quality of the triplets, these zeros come either from deflating our targeted values or others. By zooming in on the portion of the spectrum that we wanted to deflate (see Figure 8b), we get a detailed view. In the least precise case (10 iterations), we have deflated only some of our targeted values, and since the smallest one is still in place, we cannot expect a smaller condition number or a faster CRAIG convergence. Additionally, the values include a significant perturbation (see Table 1). These two effects explain why CRAIG actually needs more iterations in Figure 8a. For the case with 26 iterations, all the targeted values get deflated, but there are also a few highly perturbed ones among the rest. The corresponding convergence curve in Figure 8a resembles that of exact deflation, until the perturbed values come into play and lead it to deviate. If we run Algorithm 5.2 for 30 iterations, the resulting information is precise enough for our problem. That is to say, there is no difference between exact and approximate deflation, at least when comparing CRAIG's convergence curves. As for the case with 26 iterations, all the targeted values were successfully deflated, but additionally, the perturbation on the rest approaches machine zero (see Table 1).

How precise should the deflation vectors be? The answer is problem dependent, which means we should think about the matrix spectrum and the coefficients in Equation (3.7) related to the initial error. For the 1D channel problem, we have seen in Section 3 that only every second elliptic singular vector is part of the error. However, the smallest elliptic singular values occur in pairs with (almost) equal values. Then, Algorithm 5.2 needs some time to correctly identify the vectors, otherwise it may deliver approximations to those who are not part of our problem, while perturbing those that are. An alternative would be to adapt the block Lanczos algorithm (see [20]), which is better suited for problems where the values are not simple. Our second test case, the IFISS channel, has a spectrum with outliers that are better separated from the main cluster and from each other (see Figure 5b). There, we have found that approximate deflation is as effective as the exact one even with less precise vectors.
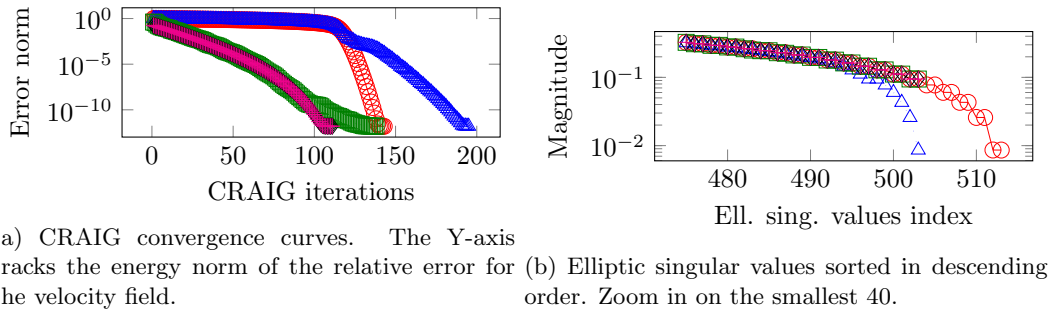
(a) CRAIG convergence curves. The Y-axis tracks the energy norm of the relative error for the velocity field.

(b) Elliptic singular values sorted in descending order. Zoom in on the smallest 40.

Fig. 8: Comparison of deflation effects on the 1D channel problem. We consider the case without deflation –⊖–, three cases with inexact deflation using vectors found after a given number of ESVD iterations (10 △ , 26 □ and 30 -◇-) and the case with exact deflation + .

6. Approximating elliptic singular triplets from a previous CRAIG solve. Obtaining approximate information about the spectrum can be done in several ways. One approach is that of finding this information prior to solving the linear system. In this case, we can first use the algorithms given in Section 5 to find the deflation vectors, then proceed to solving the deflated linear system. Another approach becomes available if one considers the case of solving several linear systems, where the right-hand side changes, but the matrix stays the same. An example of this scenario is solving time-dependent problems. It is possible then to start solving immediately, without deflation, then using the vectors resulting from the iterations to extract some spectral information which can be subsequently used to deflate the next linear systems, leading to a faster convergence. Below, we show how this second strategy of finding deflation vectors can be used in our case, by solving saddle point problems with CRAIG [2].

Our approach is inspired by the use of Ritz values and vectors associated with CG, more precisely with the Lanczos process with which CG is connected. The Ritz values are the eigenvalues of the tridiagonal matrix that can be constructed using the scalars computed by CG, and are approximations of the eigenvalues of the system matrix (see [44, Chapter 5]). The approximate eigenvectors associated with these approximate eigenvalues are known as Ritz vectors and they constitute the basis used in the deflation process. For deflation using Ritz vectors, harmonic Ritz vectors and others in the context of solving with GMRES, we refer to [32, 41, 11].

The particular strategy that we build on was presented in [43] for computing Ritz vectors from CG runs. The key feature of their algorithm, called *eigCG*, lies in the quick convergence of the Ritz vectors to the eigenvectors, without affecting the convergence of the linear solver. Even if an eigensolver and a linear solver may rely on the same algorithm (in this case, the Lanczos tridiagonalization), it is used differently in each case, which typically means one has to choose which objective gets prioritized: either the linear solver or the eigensolver. The authors of [43] show that their method can achieve both goals quickly. The success of their algorithm is partly due to the use of two consecutive tridiagonal matrices built from the output of CG. The mechanism allows them to obtain spectral information at a rate comparable to the more expensive full and reorthogonalized Lanczos process. The eigenvalue problem is solved relying on the linear solver, but not influencing it.

We follow the strategy given in [43] for computing Ritz vectors from CG runs and adapt it such that it fits our CRAIG solver and the underlying bidiagonalization process. The changes that we have to make stem from the fact that the generalized Golub-Kahan process works with

two sets of orthogonal vectors (instead of one for CG) and the fact that we are looking for elliptic singular vectors, rather than eigenvectors. We give our interpretation in Algorithm 6.1.

Algorithm 6.1 Elliptic Singular Value Decomposition 3

Input: $\mathbf{A}$ - rectangular matrix; k - number of desired triplets;   *target* - choose between *smallest* or *largest* triplets; $\mathbf{U}_b, \mathbf{B}, \mathbf{V}_b$ from the bidiagonalization; $\mathbf{U}, \boldsymbol{\Sigma}, \mathbf{V}$ previous approximate elliptic singular triplets.
Output: $\mathbf{U}, \boldsymbol{\Sigma}, \mathbf{V}$ new approximate elliptic singular triplets.

1: **if** $\mathbf{U}, \boldsymbol{\Sigma}, \mathbf{V}$ are given **then**
2:     $\mathbf{v}$ = first column of $\mathbf{V}_b$; $\mathbf{r} = \mathbf{U}^T \mathbf{A} \mathbf{v}$; $\mathbf{B} = [\begin{smallmatrix} \boldsymbol{\Sigma} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} \end{smallmatrix}]$; $\mathbf{B}_{1:2k,2k+1} = \mathbf{r}$;
3:     $\mathbf{U}_b = [\mathbf{U}\ \mathbf{U}_b]$; $\mathbf{V}_b = [\mathbf{V}\ \mathbf{V}_b]$;
4: **end if**
5: Compute the SVD of $\mathbf{B}$ and keep the k *target* triplets in $\mathbf{U}_{s1}, \boldsymbol{\Sigma}_{s1}$ and $\mathbf{V}_{s1}$;
6: Compute the SVD of $\mathbf{B}_{1:\eta-1,1:\eta-1}$ and keep the k *target* triplets in $\mathbf{U}_{s2}, \boldsymbol{\Sigma}_{s2}$ and $\mathbf{V}_{s2}$;
7: Append a row of zeros to $\mathbf{U}_{s2}$ and $\mathbf{V}_{s2}$ so that they also have η rows;
8: Concatenate and orthogonalize the left vectors $\mathbf{U}_{so} = ortho([\mathbf{U}_{s1}\ \mathbf{U}_{s2}])$;
9: Concatenate and orthogonalize the right vectors $\mathbf{V}_{so} = ortho([\mathbf{V}_{s1}\ \mathbf{V}_{s2}])$;
10: $\mathbf{H} = \mathbf{U}_{so}{}^T \mathbf{B} \mathbf{V}_{so}$;
11: Compute the SVD of $\mathbf{H}$ and store the triplets in $\mathbf{U}_{s3}, \boldsymbol{\Sigma}_{s3}$ and $\mathbf{V}_{s3}$;
12: $\mathbf{U} = \mathbf{U}_b \mathbf{U}_{so} \mathbf{U}_{s3}$; $\mathbf{V} = \mathbf{V}_b \mathbf{V}_{so} \mathbf{V}_{s3}$; $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{s3}$;

We describe the CRAIG solver in Appendix A and Algorithm A.1. In particular, we highlight where Algorithm 6.1 can be integrated in order to obtain approximations to the elliptic singular triplets from the information computed by the solver. Algorithm 6.1 is to be called repetitively and proceeds in the following way. The first time, we have $\mathbf{B} \in \mathbb{R}^{\eta \times \eta}$ with $\mathbf{U}_b \in \mathbb{R}^{m \times \eta}$ and $\mathbf{V}_b \in \mathbb{R}^{n \times \eta}$ from the bidiagonalization, similar to the first step of Algorithm 5.2. Three SVDs follow, with their results subscripted by $s1$, $s2$ and $s3$ respectively. Since the SVDs operate on small matrices, they are not particularly expensive. The first SVD is applied to $\mathbf{B}$, the second to $\mathbf{B}_{1:\eta-1,1:\eta-1}$, and the third to the combined result of the previous two. For this combination to be possible, $\mathbf{U}_{s2}$ and $\mathbf{V}_{s2}$ need to be extended and orthogonalized to $\mathbf{U}_{s1}$ and $\mathbf{V}_{s1}$. Collecting the results of the three SVDs, we have $\mathbf{U}, \boldsymbol{\Sigma}, \mathbf{V}$, a set of $2k$ approximate elliptic singular triplets, which concludes one step of Algorithm 6.1. Subsequent steps refine these $2k$ triplets from an additional $\eta-2k$ provided by the bidiagonalization. Once the CRAIG solver concludes, it returns the k triplets that we are interested in along with the solution of the saddle point system.

There are a few notable differences between Algorithm 5.2 and Algorithm 6.1. The latter makes use of the vectors resulting from the GK bidiagonalization (part of the CRAIG solver), but does not influence it at all, since it does not include an augmentation. The bidiagonalization runs only once, without restarts, in contrast to the way Algorithm 5.2 uses it. Another point is that instead of keeping only k approximations using the most recent bidiagonal matrix, Algorithm 6.1 keeps an extra set of k from the previous one. Combining these and refining them with further $\eta-2k$ vectors from the bidiagonalization leads to rapid convergence of the spectral information.

In Figure 9 and Figure 10, we plot the convergence history for the two test cases described in Section 3 in the following three scenarios: no deflation, deflation using approximate vectors and deflation using exact vectors. The approximations were obtained using Algorithm 6.1 based on vectors coming from a previous linear solve of the same system. We can see how, in both cases, approximate vectors are just as effective as their exact counterparts in the process of deflation.

In Table 2 we give the errors for the approximate elliptic singular values and vectors. If we denote the exact vectors by $\mathbf{v}_i^*$ and the approximate ones by $\mathbf{V}$, the quantities we report are
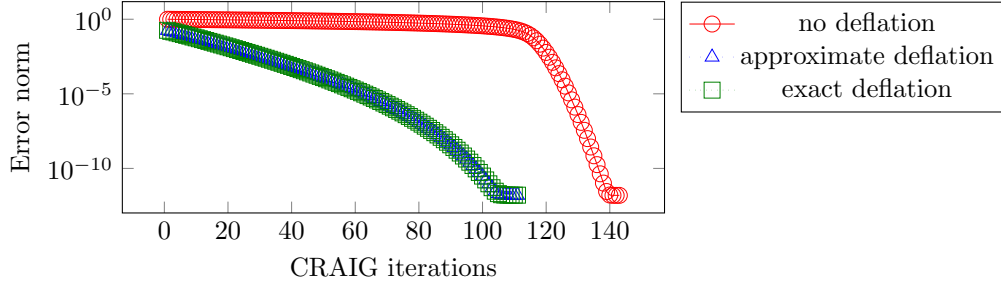
Fig. 9: Convergence curves of the CRAIG algorithm for the deflated 1D channel problem. Y-Axis: Energy norm of the relative error (Velocity field).
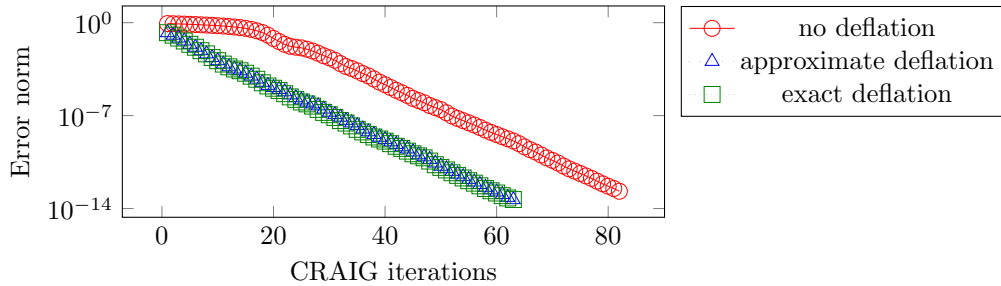

Fig. 10: Convergence curves of the CRAIG algorithm for the deflated IFISS channel problem. Y-Axis: Energy norm of the relative error (Velocity field).

$\left\| \mathbf{v}_i^* - \mathbf{V}\mathbf{V}^T\mathbf{v}_i^* \right\|_2$, with $|(\sigma_i - \sigma_i^*)/\sigma_i^*|$ being the relative error for the values. We can see how recycling information from a linear solve can lead to fairly precise approximations. However, it may be the case that this level of accuracy cannot be reached after just one solve. This is especially true when attempting to identify many vectors. For this scenario, the authors of [43] present an algorithm to refine the set using information from several linear solves.

7. Comparison with deflated MINRES. A popular choice for solving saddle point problems is the MINRES algorithm, applied monolithically and paired with an appropriately defined deflation to reduce convergence plateaus. In this section, we are interested in comparing the number of iterations performed by the linear solver following deflation, considering the two approaches: monolithic and segregated, with their respective solver MINRES and CRAIG.

In [2], an important link between the generalized CRAIG algorithm and MINRES has been drawn. If the system in Equation (1.1) is first preconditioned by a block-diagonal matrix with $\mathbf{W}$ and $\mathbf{I}$ and then solved using MINRES, the required number of iterations will be double compared to that of generalized CRAIG. This is based on existing results regarding unpreconditioned MINRES and the CRAIG algorithm without the generalization, such as those in [14] (Theorem 6.9.9). The same theorem highlights a situation where MINRES only reduces the residual every two iterations, by producing the same iterate twice in a row. This phenomenon occurs when the spectrum of the matrix is symmetric. The preconditioned saddle point matrix is

$$\begin{bmatrix} \mathbf{L}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{W} & \mathbf{A} \\ \mathbf{A}^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{L}^{-T} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \tilde{\mathbf{A}} \\ \tilde{\mathbf{A}}^T & \mathbf{0} \end{bmatrix},$$

Table 2: Relative errors for the smallest five approximate elliptic singular vectors and values (ordered descendingly) obtained from Algorithm 6.1.

triplet index	5th	4th	3rd	2nd	1st
1D Channel problem					
vectors	5.18E-06	3.52E-07	7.56E-08	3.23E-08	1.17E-08
values	3.78E-11	4.07E-13	4.19E-14	1.48E-15	1.29E-13
IFISS Channel problem					
vectors	4.56E-02	5.95E-05	7.02E-09	5.64E-08	1.02E-05
values	1.52E-04	5.37E-10	4.12E-15	5.94E-15	1.23E-13

where we use a Cholesky decomposition of the (1,1)-block $\mathbf{W} = \mathbf{L}\mathbf{L}^T$, $\mathbf{W}^{-1} = \mathbf{L}^{-T}\mathbf{L}^{-1}$ and define $\tilde{\mathbf{A}} = \mathbf{L}^{-1}\mathbf{A}$ as in Section 2. We can describe its spectrum by adapting Result 1 from [40] to our context:

$$\begin{cases} 1, & \text{m-n times,} \\ \frac{1}{2} + \sqrt{\sigma_i^2 + \frac{1}{4}}, & \text{n times,} \\ \frac{1}{2} - \sqrt{\sigma_i^2 + \frac{1}{4}}, & \text{n times,} \end{cases}$$

where σ_i are the n elliptic singular values of $\mathbf{A}$ (singular values of $\tilde{\mathbf{A}}$). If we ignore the cluster of values equal to one, which do not impact the convergence of MINRES, the rest of the spectrum is symmetric around the value 0.5. This symmetry then leads to redundant iterations in MINRES and GMRES as well, as proven theoretically in [15]. The authors do note that in numerical experiments, the two consecutive iterates may not necessarily be identical, but are very close nonetheless.

We can observe this effect in our context as well, when comparing generalized CRAIG and preconditioned MINRES. In Figure 11, we show the convergence curves of the two algorithms when solving the 1D channel problem described in Section 3. Notice how every second MINRES iteration does not visibly contribute to the reduction of the error. With the exception of this feature, the two algorithms converge in the same way, featuring the same two phases of slow initial convergence (plateau) and subsequent superlinear convergence.

We proceed to test whether this 1:2 iteration ratio is preserved when applying deflation. When using MINRES, we consider the saddle point matrix as a whole and deflate it using the procedure described in [17], relying on the eigenpairs. When using CRAIG, we make use of the elliptic singular triplets and apply deflation following Section 2. Both approaches eliminate the same number of values from the spectrum of their respective problem. The convergence history of both algorithms is given in Figure 11, for the 1D channel test case. We also compare the two scenarios on the IFISS channel problem presented in Section 3, with the results plotted in Figure 12. As we can see, the same relationship between the two algorithms holds, both with and without deflation. It should be noted that the respective curves are not identical (aside from the 1:2 scale) partly because the two algorithms under consideration have different minimization goals. For MINRES, this is the residual norm of the preconditioned saddle point system, whereas CRAIG minimizes the energy norm of the error for the primal variable $\mathbf{u}$.

7.1. Augmented CRAIG. Above, we have compared the two algorithms only with respect to the number of iterations. If we are interested in memory use or time, we can consider augmented MINRES [34] and augmented CRAIG, the latter being described below.

Augmentation can be seen as an efficient implementation of deflation. First, given the
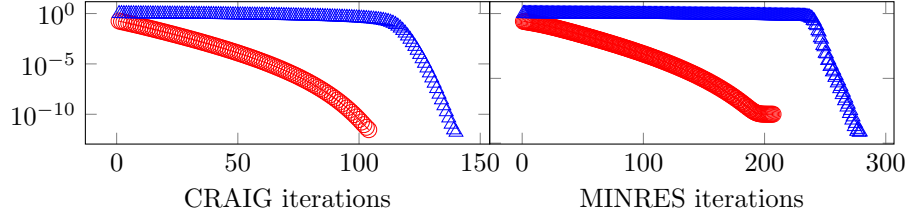
Energy norm of relative error (Velocity field)

Fig. 11: Convergence curves of the CRAIG algorithm (Left) and preconditioned MINRES (Right) for the 1D channel problem. Undeflated in △ , exact deflation in ⊖.

Energy norm of relative error (Velocity field)

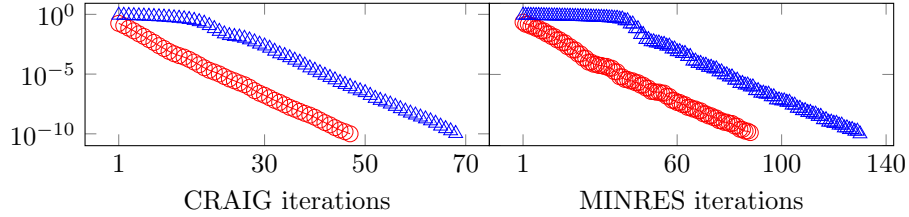Fig. 12: Convergence curves of the CRAIG algorithm (Left) and preconditioned MINRES (Right) for the IFISS channel problem. Undeflated in △ , exact deflation in ⊖.

definition of the operators $\mathbf{M}$, $\mathbf{P}$ and $\mathbf{Q}$ in Equations (2.5)–(2.7), they will most likely be dense matrices if explicitly computed and stored. Then, the deflated matrix, whether computed as $\mathbf{PA}$ or $\mathbf{AQ}$, will also be dense. However, in many applications, the matrices that result from the discretization are (very) large and sparse. Losing sparsity by a naive deflation might make the system difficult or impossible to solve due to excessive requirements concerning memory and time.

We have defined deflation to be paired with the CRAIG solver [2], which only uses the matrix $\mathbf{A}$ as part of matrix-vector products $\mathbf{Ax}$, for some vector $\mathbf{x}$. To avoid the difficulties mentioned above, these products should be reformulated to include the effect of deflation in an implicit manner. Applying the definition of the deflation operators we get $\mathbf{Ax} - \mathbf{AV\Sigma}^{-1}\mathbf{U}^T\mathbf{Ax}$, which has the advantage of manipulating matrices that are either sparse or small (with only k columns). Suitable alternatives in the same style can be defined to replace the operations in Equation (2.9), avoiding explicit use of $\mathbf{P}, \mathbf{Q}$, and $\mathbf{M}$. This strategy corresponds to an *augmented* CRAIG solver, similar to the augmented MINRES described in [34], where the basis for deflation is passed to the solver and applied whenever needed. Here, we note that MINRES operates monolithically, with a basis $\mathbf{Y} \in \mathbb{R}^{(m+n)\times k}$ applied repeatedly, whereas augmented CRAIG uses either $\mathbf{U} \in \mathbb{R}^{m\times k}$ or $\mathbf{V} \in \mathbb{R}^{n\times k}$. Thus, adding augmentation to CRAIG is less expensive than to MINRES. Although CRAIG needs matrix-vector products with both $\mathbf{A}$ and $\mathbf{A}^T$, it is sufficient to reformulate only that with $\mathbf{A}$. It can be shown using the properties

from Equation (2.10) that the Schur complement $\mathbf{S} = \mathbf{A}^T\mathbf{W}^{-1}\mathbf{A}$ is correctly deflated even with this "one-sided" augmentation, which leads CRAIG to converge quickly to the solution of Equation (2.8). For a more detailed comparison of augmentation and deflation, we refer to [10, 16].

REMARK 1. *Further savings compared to augmented MINRES are possible if we have exact elliptic singular triplets. Then, it is true that* $\mathbf{V}\mathbf{V}^T = \mathbf{M}\mathbf{A}$, *which we use to define a simpler and quicker matrix-vector product:* $\mathbf{A}\mathbf{x} - \mathbf{A}\mathbf{V}\mathbf{V}^T\mathbf{x}$, *circumventing matrices of order m. This simplification can still be applied if we have approximate instead of exact triplets, provided we are only interested in a solution of moderate accuracy, as is the case in certain applications. One such example is solving linear systems as part of a nonlinear solver, as shown in [11], where the linear solver needs to reduce the residual norm only by* 10^{-1}. *Using the simpler* $\mathbf{Q} := \mathbf{I}_n - \mathbf{V}\mathbf{V}^T$ *in the deflation and correction stages we described in Section 2 introduces a perturbation proportional to the accuracy of the elliptic singular vectors. This then limits the accuracy of the solution we seek for the initial saddle point system. Such a behavior is similar to a different setting, where we approximate the action of* $\mathbf{W}^{-1}$ *in the CRAIG solver by using an iterative solver instead [12].*

8. Deflation and preconditioning. We have seen how targeted, local changes in the spectrum can be induced by means of deflation. Preconditioning, on the other hand, typically aims to change the spectrum as a whole, clustering the values away from zero. For saddle point problems, the following block diagonal preconditioner is considered ideal

$$\mathcal{P} = \begin{bmatrix} \mathbf{W} & \mathbf{0} \\ \mathbf{0} & \mathbf{S} \end{bmatrix}.$$

It is ideal in the sense that depending on the solver used, when applied as $\mathcal{P}^{-1}$ to a system like the one in Equation (1.1), three or fewer iterations will be sufficient to converge to the exact solution. Since $\mathbf{S}^{-1}$ is difficult to obtain, an approximation is used instead, trying to balance the cost of building and applying it against the improvement in convergence speed that it brings. The preconditioning strategy used in the remainder of this section, but also other choices for saddle point problems are described in [8, Section 10].

For Stokes flow problems, the pressure mass matrix $\mathbf{Q}_p$ is commonly employed to achieve such an approximation, exploiting the spectral equivalence it has to the Schur complement (see [13, Chapter 3]). It is formally defined as

$$\mathbf{Q}_p = [\mathbf{q}_{ij}], \quad \mathbf{q}_{ij} = \int_\Omega \psi_i\psi_j, \quad i,j = 1,...,n,$$

where Ω is the domain, $\{\psi\}_{i=1}^n$ are basis functions for the finite-dimensional space $\mathcal{M}_n \in \mathcal{M}$, with $\mathcal{M}$ being the Hilbert space for the pressure. To further reduce the cost of applying the preconditioner, it is simplified to a diagonal matrix, for example by lumping, i.e. placing each row-sum on the diagonal.

We proceed to apply this kind of preconditioning to our IFISS channel problem and compare its behavior with that of deflation. As Figure 13a illustrates, CRAIG converges faster in the presence of this preconditioner, but the initial plateau phase is mostly unaffected. Deflation of the five smallest elliptic singular values removes the plateau, enabling a faster convergence to solutions of moderate accuracy (error norm $> 10^{-7}$). However, this deflation does not improve the second stage of the convergence process, which is important if we aim for a very accurate solution. The complementary action of the preconditioner and deflation motivates the third scenario, in which both are applied, leading to the best result. A similar combined approach was employed in [34] for a different type of Stokes problem solved with MINRES.
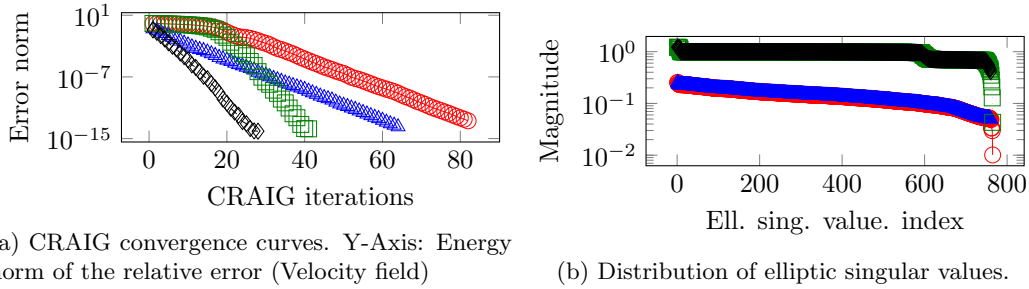
(a) CRAIG convergence curves. Y-Axis: Energy
norm of the relative error (Velocity field)

(b) Distribution of elliptic singular values.

Fig. 13: Corresponding CRAIG convergence curves and Schur complement spectra for the IFISS channel of length 20. We consider the original case —⊖—, the deflated case (five smallest elliptic singular values) △ , the preconditioned case (lumped pressure mass matrix) □ , and the case with both deflation and preconditioning - ⬦- .

To understand the effect of the pressure mass matrix preconditioner, we plot in Figure 13b the elliptic spectra for the four cases illustrated in Figure 13a. The most visible effect is how almost 600 of the largest values cluster around 1, while a second cluster forms around 0.7. However, the small outliers are not brought closer to the rest of the values, which is reflected in the persistence of the plateau visible in Figure 13a.

Stokes problems on elongated domains represent a special case, where the spectrum of the Schur complement changes also as a function of the anisotropy. More precisely, the smallest elliptic singular value is bounded (see [9])

$$\frac{1}{2\sqrt{15}}l^{-1} \leq \sigma_n \leq \frac{\pi}{2\sqrt{3}}l^{-1},$$

where $l > 1$ is the ratio between the sides of the channel. As this ratio increases, the smallest elliptic singular value approaches zero. Additionally, as seen in Figure 5b, more and more small outliers appear. Deflation can then be used to manage the outliers' effect, while the pressure mass matrix as a preconditioner clusters the other elliptic singular values.

9. Conclusions. In this paper, we have defined a particular kind of deflation to be used with CRAIG, a segregated linear solver for symmetric saddle point problems. The motivation for this development was to accelerate convergence for problems with a spectrum which contains outliers, leading to stagnation periods in the convergence process. As an example of such a problem, we have considered two kinds of Stokes flow in thin, elongated channels. Our definition of deflation uses bases from an elliptic singular value decomposition of the off-diagonal block **A** as an alternative to explicitly deflating the Schur complement, which could pose numerical difficulties. With this deflation mechanism, stagnation periods can be shortened or removed, even if we only have bases with inexact spectral information. We have extended two iterative algorithms, such that their output is a partial approximate elliptic singular value decomposition which can be employed in deflation. Following a comparison with MINRES, we have seen that our deflation for a segregated solver performs just as well as that for a monolithic one, removing the plateau in convergence. The ability to exploit block structure and the lack of redundant iterations in CRAIG highlights the advantage and need of having a segregated deflation strategy. Additionally, the augmentation procedure we have described for CRAIG operates with smaller matrices than MINRES, leading to faster iterations. As future work, a promising avenue is that of using multigrid transfer operators instead of the ESVD as a way to obtain bases for deflation.

REFERENCES

[1] K. Aishima, *Global convergence of the restarted Lanczos and Jacobi–Davidson methods for symmetric eigenvalue problems*, Numer. Math., 131 (2015), pp. 405–423.

[2] M. Arioli, *Generalized Golub–Kahan bidiagonalization and stopping criteria*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 571–592.

[3] O. Axelsson and J. Karátson, *Reaching the superlinear convergence phase of the CG method*, J. Comput. Appl. Math., 260 (2014), pp. 244–257.

[4] J. Baglama and L. Reichel, *Augmented implicitly restarted Lanczos bidiagonalization methods*, SIAM J. Sci. Comput., 27 (2005), pp. 19–42.

[5] J. Baglama, L. Reichel, and D. Richmond, *An augmented LSQR method*, Numer. Algorithms, 64 (2013), pp. 263–293.

[6] J. L. Barlow, *Reorthogonalization for the Golub–Kahan–Lanczos bidiagonal reduction*, Numer. Math., 124 (2013), pp. 237–278.

[7] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst, *Templates for the solution of linear systems: building blocks for iterative methods*, SIAM, 1994.

[8] M. Benzi, G. H. Golub, and J. Liesen, *Numerical solution of saddle point problems*, Acta Numer., 14 (2005), pp. 1–137.

[9] E. V. Chizhonkov and M. A. Olshanskii, *On the domain geometry dependence of the LBB condition*, ESAIM Math. Model. Numer. Anal., 34 (2000), pp. 935–951.

[10] O. Coulaud, L. Giraud, P. Ramet, and X. Vasseur, *Deflation and augmentation techniques in Krylov subspace methods for the solution of linear systems*, arXiv preprint arXiv:1303.5692, (2013).

[11] H. A. Daas, L. Grigori, P. Hénon, and P. Ricoux, *Recycling Krylov Subspaces and Truncating Deflation Subspaces for Solving Sequence of Linear Systems*, ACM Trans. Math. Software, 47 (2021), pp. 1–30.

[12] V. Darrigrand, A. Dumitrasc, C. Kruse, and U. Rüde, *Inexact inner–outer Golub–Kahan bidiagonalization method: A relaxation strategy*, Numer. Linear Algebra Appl., (2022), p. e2484.

[13] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*, Numer. Math. Sci. Comput., 2014.

[14] B. Fischer, *Polynomial based iteration methods for symmetric linear systems*, SIAM, 2011.

[15] B. Fischer, A. Ramage, D. J. Silvester, and A. J. Wathen, *Minimum residual methods for augmented systems*, BIT, 38 (1998), pp. 527–543.

[16] A. Gaul, M. H. Gutknecht, J. Liesen, and R. Nabben, *Deflated and augmented Krylov subspace methods: Basic facts and a breakdown-free deflated MINRES*, tech. report, Technical University Berlin, 2011. MATHEON Preprint 759.

[17] A. Gaul, M. H. Gutknecht, J. Liesen, and R. Nabben, *A framework for deflated and augmented Krylov subspace methods*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 495–518.

[18] L. Giraud and S. Gratton, *On the sensitivity of some spectral preconditioners*, SIAM J. Matrix Anal. Appl., 27 (2006), pp. 1089–1105.

[19] L. Giraud, J. Langou, and M. Rozložník, *The loss of orthogonality in the Gram-Schmidt orthogonalization process*, Comput. Math. Appl., 50 (2005), pp. 1069–1075.

[20] G. H. Golub and R. Underwood, *The block Lanczos method for computing eigenvalues*, in Mathematical software, Elsevier, 1977, pp. 361–377.

[21] G. H. Golub and C. F. Van Loan, *Matrix computations*, Johns Hopkins studies in the mathematical sciences, Johns Hopkins University Press, Baltimore, MD, 1996.

[22] G. H. Golub, Z. Zhang, and H. Zha, *Large sparse symmetric eigenvalue problems with homogeneous linear constraints: the Lanczos process with inner–outer iterations*, Linear Algebra Appl., 309 (2000), pp. 289–306.

[23] M. Griebel, T. Dornseifer, and T. Neunhoeffer, *Numerical simulation in fluid dynamics: a practical introduction*, SIAM, 1998.

[24] M. H. Gutknecht, *Spectral deflation in Krylov solvers: A theory of coordinate space based methods*, Electron. Trans. Numer. Anal, 39 (2012), pp. 156–185.

[25] F. H. Harlow and J. E. Welch, *Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface*, Phys. Fluids, 8 (1965), pp. 2182–2189.

[26] N. Kohl and U. Rüde, *Textbook efficiency: massively parallel matrix-free multigrid for the Stokes system*, arXiv preprint arXiv:2010.13513, (2020).

[27] 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).

[28] C. Kruse, M. Sosonkina, M. Arioli, N. Tardieu, and U. Rüde, *Parallel performance of an iterative solver based on the Golub-Kahan bidiagonalization.*, In: Wyrzykowski R., Deelman E., Dongarra J., Karczewski K. (eds) Parallel Processing and Applied Mathematics. PPAM 2019. Lect. Notes Comp.

Sci., 12043 (2020).

[29] C. Kruse, M. Sosonkina, M. Arioli, N. Tardieu, and U. Rüde, *Parallel solution of saddle point systems with nested iterative solvers based on the Golub-Kahan bidiagonalization*, Concurr. Comput., 33 (2021), p. e5914.

[30] J. Liesen and P. Tichỳ, *Convergence analysis of Krylov subspace methods*, GAMM-Mitt., 27 (2004), pp. 153–173.

[31] L. A. M. Mello, E. De Sturler, G. H. Paulino, and E. C. N. Silva, *Recycling Krylov subspaces for efficient large-scale electrical impedance tomography*, Comput. Methods Appl. Mech. Eng., 199 (2010), pp. 3101–3110.

[32] S. Mercier, S. Gratton, N. Tardieu, and X. Vasseur, *A new preconditioner update strategy for the solution of sequences of linear systems in structural mechanics: application to saddle point problems in elasticity*, Comput. Mech., 60 (2017), pp. 969–982.

[33] R. Nabben and C. Vuik, *A comparison of deflation and coarse grid correction applied to porous media flow*, SIAM J. Numer. Anal., 42 (2004), pp. 1631–1647.

[34] M. A. Olshanskii and V. Simoncini, *Acquired clustering properties and solution of certain saddle point systems*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2754–2768.

[35] D. Orban and M. Arioli, *Iterative solution of symmetric quasi-definite linear systems*, SIAM, 2017.

[36] C. C. Paige, *Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem*, Linear Algebra Appl., 34 (1980), pp. 235–258.

[37] A. Ramage, D. Ruiz, A. Sartenaer, and C. Tannier, *Using partial spectral information for block diagonal preconditioning of saddle-point systems*, Comput. Optim. Appl., 78 (2021), pp. 353–375.

[38] M. Rozložník, M. Tůma, A. Smoktunowicz, and J. Kopal, *Numerical stability of orthogonalization methods with a non-standard inner product*, BIT, 52 (2012), pp. 1035–1058.

[39] Y. Saad, *Iterative methods for sparse linear systems*, SIAM, 2003.

[40] M. A. Saunders, *Solution of sparse rectangular systems using LSQR and CRAIG*, BIT, 35 (1995), pp. 588–604.

[41] J. A. Sifuentes, M. Embree, and R. B. Morgan, *GMRES convergence for perturbed coefficient matrices, with application to approximate deflation preconditioning*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1066–1088.

[42] K. M. Soodhalter, E. de Sturler, and M. E. Kilmer, *A survey of subspace recycling iterative methods*, GAMM-Mitt., 43 (2020), p. e202000016.

[43] A. Stathopoulos and K. Orginos, *Computing and deflating eigenvalues while solving multiple right-hand side linear systems with an application to quantum chromodynamics*, SIAM J. Sci. Comput., 32 (2010), pp. 439–462.

[44] H. A. Van der Vorst, *Iterative Krylov methods for large linear systems*, no. 13 in Cambridge monographs on applied and computational mathematics, Cambridge University Press, 2003.

[45] E. van't Wout, M. B. van Gijzen, A. Ditzel, A. van der Ploeg, and C. Vuik, *The Deflated Relaxed Incomplete Cholesky CG method for use in a real-time ship simulator*, Proc. Comput. Sci., 1 (2010), pp. 249–257.

[46] N. Venkovic, P. Mycek, L. Giraud, and O. Le Maitre, *Comparative study of harmonic and Rayleigh-Ritz procedures with applications to deflated conjugate gradients*, tech. report, CERFACS, 2020. TR/PA/20-3.

Appendix A. The CRAIG algorithm.

The CRAIG algorithm iteratively solves the saddle point system in Equation (1.1) in the following way. It computes the generalized Golub-Kahan bidiagonalization

$$(A.1) \qquad \mathbf{A}\mathbf{V}_i = \mathbf{W}\mathbf{U}_i\mathbf{B}_i, \quad \mathbf{U}_i^T\mathbf{W}\mathbf{U}_i = \mathbf{I}_i, \quad \mathbf{A}^T\mathbf{U}_i = \mathbf{V}_i\mathbf{B}_i^T + \beta_{i+1}\mathbf{q}_{i+1}\mathbf{e}_i^T, \quad \mathbf{V}_i^T\mathbf{V}_i = \mathbf{I}_i,$$

with the bidiagonal matrix

$$(A.2) \qquad \mathbf{B}_i = \begin{bmatrix} \alpha_1 & \beta_2 & 0 & \dots & 0 \\ 0 & \alpha_2 & \beta_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & \alpha_{i-1} & \beta_i \\ 0 & \dots & 0 & 0 & \alpha_i \end{bmatrix}.$$

Using $\mathbf{U}_i$ and $\mathbf{V}_i$ it transforms Equation (1.1) into

$$(A.3) \qquad \begin{bmatrix} \mathbf{I}_i & \mathbf{B}_i \\ \mathbf{B}_i^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{z}_i \\ \mathbf{y}_i \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{V}_i^T\mathbf{b} \end{bmatrix},$$

with the solution

$$(A.4) \qquad \mathbf{z}_i = \beta_1 \mathbf{B}_i^{-T}\mathbf{e}_1; \quad \mathbf{y}_i = -\mathbf{B}_i^{-1}\mathbf{z}_i,$$

where $\mathbf{e}_1$ is the first column of the identity matrix $\mathbf{I}_i \in \mathbb{R}^{i \times i}$. The approximate solution of the initial problem is then given by

$$(A.5) \qquad \mathbf{u}^{(i)} = \mathbf{U}_i\mathbf{z}_i; \quad \mathbf{p}^{(i)} = \mathbf{V}_i\mathbf{y}_i.$$

Note: $\mathbf{u}^{(i)}$ is the i-th approximate solution, while $\mathbf{u}_i$ is the i-th left vector.

It is also possible to define updates for $\mathbf{u}^{(i)}$ and $\mathbf{p}^{(i)}$ in a recursive manner. To this end, we will need only ζ_i (the last entry of $\mathbf{z}_i$) and an auxiliary vector $\mathbf{d}$ for $\mathbf{p}^{(i)}$ (see [2] for details). The advantage of a recursive approach is that each update needs only the latest pair of left and right vectors.

In Algorithm A.1, we give a description of the CRAIG solver, and how it can interact with Algorithm 6.1. An efficient implementation has been given in [2], making use of the simple structure of $\mathbf{B}$ and short term recurrences. The author also presents error bounds useful for monitoring convergence.

An important point is that the use of short term recurrences implies there is no need to store the left and right vectors in order to compute the approximate solutions. However, for our purposes in this work they are necessary in order to extract information about the elliptic singular triplets from them using Algorithm 6.1. After a group of η left and right bidiagonalization vectors have been processed, they can be discarded. Additionally, while the iterative linear solver can deliver satisfactory approximate solutions even if the bidiagonalization vectors gradually lose orthogonality, the same may not necessarily hold in our extended context. This leads to the potential need for reorthogonalization, with similar remarks as in Section 5.

We present in Algorithm A.1 how the original Craig solver is extended to make use of Algorithm 6.1. We mark the additional statements needed for Algorithm 6.1 with special line numbers, and the associated inputs and outputs with a ●.

Algorithm A.1 CRAIG algorithm

Input: $\mathbf{A} \in \mathbb{R}^{m \times n}, (n < m); \quad \mathbf{W} \in \mathbb{R}^{m \times m}$, SPD; $\quad \mathbf{g} \in \mathbb{R}^m; \quad \mathbf{r} \in \mathbb{R}^n; \quad maxiter \in \mathbb{R};$
- $k \in \mathbb{R}$ - number of elliptic singular triplets;
- $\eta \in \mathbb{R}$ - maximum search subspace dimension;
- *target* - choose between smallest or largest triplets;

Output: $\mathbf{u} \in \mathbb{R}^m; \quad \mathbf{p} \in \mathbb{R}^n;$
- $\mathbf{U} \in \mathbb{R}^{m \times k}$ - approximate left elliptic singular vectors;
- $\mathbf{\Sigma} \in \mathbb{R}^{k \times k}$ - approximate elliptic singular values;
- $\mathbf{V} \in \mathbb{R}^{n \times k}$ - approximate right elliptic singular vectors;

1: $\mathbf{u}^{(0)} = \mathbf{W}^{-1}\mathbf{g}; \qquad \mathbf{b} = \mathbf{r} - \mathbf{A}^T\mathbf{u}^{(0)}; \quad \beta_1 = \|\mathbf{b}\|_2; \quad \mathbf{v}_1 = \mathbf{b}/\beta_1;$
2: $\mathbf{w} = \mathbf{W}^{-1}\mathbf{A}\mathbf{v}_1; \quad \alpha_1 = \|\mathbf{w}\|_\mathbf{W}; \quad \mathbf{u}_1 = \mathbf{w}/\alpha_1; \quad \zeta_1 = \beta_1/\alpha_1; \quad \mathbf{d}_1 = \mathbf{v}_1/\alpha_1;$
3: $\mathbf{u}^{(1)} = \mathbf{u}^{(0)} + \zeta_1\mathbf{u}_1; \quad \mathbf{p}^{(1)} = -\zeta_1\mathbf{d}_1; \quad i = 2;$
3.1: $\mathbf{B}_1 = \alpha_1; \quad \mathbf{V}_1 = \mathbf{v}_1; \quad \mathbf{U}_1 = \mathbf{u}_1;$
4: **while** not converged AND $i < maxiter$ **do**
5: $\quad \mathbf{v} = \mathbf{A}^T\mathbf{u}_{i-1} - \alpha_{i-1}\mathbf{v}_{i-1};$
5.1: $\quad$ Reorthogonalization: $\mathbf{v} = \mathbf{v} - \mathbf{V}_i\mathbf{V}_i^T\mathbf{v};$
6: $\quad \beta_i = \|\mathbf{v}\|_2; \quad \mathbf{v}_i = \mathbf{v}/\beta_i;$
7: $\quad \mathbf{w} = \mathbf{W}^{-1}\left(\mathbf{A}\mathbf{v}_i - \beta_i\mathbf{W}\mathbf{u}_{i-1}\right);$
7.1: $\quad$ **W**-reorthogonalization: $\mathbf{w} = \mathbf{w} - \mathbf{U}_i\mathbf{U}_i^T\mathbf{W}\mathbf{w};$
8: $\quad \alpha_i = \|\mathbf{w}\|_\mathbf{W}; \quad \mathbf{u}_i = \mathbf{w}/\alpha_i;$
8.1: $\quad \mathbf{V}_i = [\mathbf{V}_{i-1}\ \mathbf{v}_i]; \quad \mathbf{U}_i = [\mathbf{U}_{i-1}\ \mathbf{u}_i]; \quad \mathbf{B}_i = [\mathbf{B}_{i-1}\ \beta_i\mathbf{e}_{i-1}]; \quad \mathbf{B}_i = \begin{bmatrix} \mathbf{B}_i \\ \alpha_i\mathbf{e}_i^T \end{bmatrix};$
8.2: $\quad$ **if** $mod(i, \eta) = 0$ **then**
8.3: $\quad\quad \mathbf{B}_\eta = $ lower right $\eta \times \eta$ submatrix of $\mathbf{B}_i;$
8.4: $\quad\quad$ **if** $i = \eta$ **then**
8.5: $\quad\quad\quad$ Call Algorithm 6.1 with $\mathbf{A}, k, \mathbf{U}_b = \mathbf{U}_i, \mathbf{B} = \mathbf{B}_\eta, \mathbf{V}_b = \mathbf{V}_i$. Get $2k$ approximate triplets in $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}$.
8.6: $\quad\quad\quad \eta = \eta - 2k;$
8.7: $\quad\quad$ **else**
8.8: $\quad\quad\quad$ Call Algorithm 6.1 with $\mathbf{A}, k, \mathbf{U}_b = \mathbf{U}_i, \mathbf{B} = \mathbf{B}_\eta, \mathbf{V}_b = \mathbf{V}_i$ and the current $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}$. Get $2k$ updated approximate triplets in $\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}$.
8.9: $\quad\quad$ **end if**
8.10: $\quad\quad$ Clear $\mathbf{V}_i$ and $\mathbf{U}_i$.
8.11: $\quad$ **end if**
9: $\quad \zeta_i = -\frac{\beta_i}{\alpha_i}\zeta_{i-1}; \quad \mathbf{d}_i = (\mathbf{v}_i - \beta_i\mathbf{d}_{i-1})/\alpha_i;$
10: $\quad \mathbf{u}^{(i)} = \mathbf{u}^{(i-1)} + \zeta_i\mathbf{u}_i; \quad \mathbf{p}^{(i)} = \mathbf{p}^{(i-1)} - \zeta_i\mathbf{d}_i;$
11: $\quad i = i + 1;$
12: **end while**
13: $\mathbf{u} = \mathbf{u}^{(i)}; \quad \mathbf{p} = \mathbf{p}^{(i)};$
13.1: Choose the k *target* triplets from $\mathbf{U}, \mathbf{\Sigma}$ and $\mathbf{V}$.