Les Cahiers du GERAD ISSN: 0711–2440

Projected Krylov Methods for Unsymmetric Augmented Systems

D. Orban

G–2008–46

June 2008

Les textes publiés dans la série des rapports de recherche HEC n'engagent que la responsabilité de leurs auteurs. La publication de ces rapports de recherche bénéficie d'une subvention du Fonds québécois de la recherche sur la nature et les technologies.

Projected Krylov Methods for Unsymmetric Augmented Systems

Dominique Orban

GERAD and
École Polytechnique de Montréal
C.P. 6079, Succ. Centre-ville
Montréal (Québec) Canada, H3C 3A7

dominique.orban@gerad.ca

June 2008

Les Cahiers du GERAD

G–2008–46

Copyright © 2008 GERAD

Abstract

We propose a class of projected Krylov methods for the solution of unsymmetric augmented systems of equations such as those arising from the finite-element formulation of Navier-Stokes multi-fluid flow problems. The iterative methods only rely on matrix-vector products with the $(1,1)$ block of the augmented matrix—not with its transpose—and on a one-time symmetric indefinite factorization of a projection matrix. No computation of Schur complements or generalized inverses is necessary, nor is the computation of a nullspace or of a range-space basis. Numerical results coming from fluid dynamics examples illustrate the present approach and compare it to a direct application of a Krylov method to the augmented system, and to a direct factorization of the system – assuming that the latter is feasible.

Key Words: Unsymmetric systems, saddle-point systems, augmented systems, Krylov method, symmetric indefinite factorization.

Résumé

Nous proposons une classe de méthodes de Krylov projetées pour la résolution de systèmes linéaires augmentés non-symétriques tels que ceux qui surviennent dans la formulation par éléments finis de problèmes d'écoulement multi-fluides de Navier-Stokes. Ces méthodes itératives sont uniquement basées sur des produits matrice-vecteur avec le bloc $(1,1)$ de la matrice augmentée – et non avec la transposée – ainsi que sur la factorisation symétrique indéfinie d'une matrice de projection. Il n'est pas nécessaire de calculer de complément de Schur, d'inverse généralisé, ni de base du noyau ou de l'image d'une application linéaire. Des résultats numériques provenant de problèmes de dynamique des fluides illustrent notre approche et la compare à une application directe d'une méthode de Krylov au système augmenté ainsi qu'à une factorisation directe lorsque celle-ci est possible.

Acknowledgments: The author wishes to thank Steven Dufour and Alain Fidahoussen for discussions that sparked the development of the present research, for generating the test problems and for testing preliminary versions of the projected Bi-CGSTAB method in a finite-element code. Research partially supported by NSERC Discovery Grant 299010-04.

1 Introduction

We consider the iterative solution of systems of the form

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} b \\ d \end{bmatrix}, \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$ is square but not necessarily symmetric, $B \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^n$ and $d \in \mathbb{R}^m$. In a fluid flow context, systems such as (1.1) arise, for instance, as subproblems in Navier-Stokes iterations and must be solved to compute a correction (u, p) in the velocity and pressure fields. The large size of such problems often preclude a direct factorization of the coefficient matrix of (1.1). Iterative methods applied directly to (1.1) usually perform poorly as they are unable to take the rich structure of the system into account.

In single-fluid flows, and under usual assumptions, A is symmetric and positive definite and the structure of the augmented matrix in (1.1) is well known [33]. Direct or iterative methods taking advantage of symmetry may be used to solve the linear system for whenever A is symmetric and positive definite over the nullspace of B, (1.1) represents the first-order optimality conditions for the equality-constrained quadratic program

$$\underset{u\in\mathbb{R}^n}{\text{minimize}} \;\; -b^T u + \tfrac{1}{2} u^T A u \qquad \text{subject to} \;\; Bu = d, \tag{1.2}$$

where the variables p play the role of Lagrange multipliers. In this paper, we are interested in applications where the density of the coefficient matrix is mostly due to A as m is often substantially smaller than n and B is thus a "flat" matrix. This occurs, for instance, when considering the flow of two or more immiscible fluids through a cavity. Assuming B has full row rank, it is therefore often feasible to compute a factorization of the *projection matrix*

$$\begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix}, \tag{1.3}$$

where G is a sparse symmetric approximation to A that is positive definite over the nullspace of B, the simplest choice being $G = I$. A particularly efficient iterative method for (1.2) is then the projected preconditioned conjugate gradient algorithm [13, 31, 44] often used in nonlinear optimization contexts. Note that when $G = I$, the matrix (1.3) represents the orthogonal projector onto the nullspace of B. The projected preconditioned conjugate gradient method typically requires the factorization of a single projection matrix with $G = I$ and one matrix-vector product with A per iteration.

In this paper we examine similar iterations in the case where A is unsymmetric. The interpretation in terms of an optimization problem such as (1.2) is then lost. We are particularly interested in the case where A may not be assembled explicitly but rather, matrix-vector products with A may be obtained by calling a function. We will however assume that B is available explicitly. In the following, we examine families of efficient Krylov-type methods that can accomodate such a situation.

A Krylov-type iterative method applied to (1.1) usually requires matrix-vector products involving the augmented coefficient matrix of (1.1) and possibly also its transpose. Note that the augmented matrix is indefinite. In case A is symmetric but not positive definite, applicable Krylov methods include the minimum residual method MINRES and SYMMLQ [45]. When A is not symmetric, applicable methods include the conjugate gradient on the normal equations CGNE or CGNR [22, 25], the generalized minimum residual method GMRES and its

restarted variant GMRES(m) [51], the bi-conjugate gradient Bi-CG [25], the quasi-minimum residual method QMR [27] and its transpose-free variant TFQMR [29], the conjugate gradient squared method CGS [50], and the stabilized bi-conjugate gradient method Bi-CGSTAB [54]. All methods require a number of matrix-vector products with the coefficient matrix at each iteration. The methods Bi-CG and QMR also require a number of products with its transpose.

In fluid-flow contexts, obtaining A^T or products with A^T may entice discretizing a transpose differential operator or performing an additional sweep through the mesh. Since we may not want to incur this additional cost, especially repetitively as in Navier-Stokes iterations, we eliminate the Bi-CG and QMR methods. We also eliminate MINRES and SYMMLQ since we are interested in the case where A is not symmetric. We further eliminate GMRES and GMRES(m) because of their appetite for memory and the difficulty of determining an appropriate restart value. The methods CGNE, CGNR are applied to a system whose condition number is the square of that of A and thus often suffer from severe numerical cancellation effects, although some refinements were proposed in [6]. Finally, CGS often exhibits erratic convergence patterns and also suffers from cancellation effects. This leaves TFQMR and Bi-CGSTAB as candidates to tackle (1.1). Of course, we must keep in mind that in general, all Krylov methods exhibit satisfactory convergence provided they are appropriately preconditioned. However, we will show that the methods proposed in this paper perform well even when no preconditioner is used. We refer the reader to [5] for a description of most of the above methods along with numerous pointers to further references.

In the present context, a *projected Krylov method* results from formulating problem (1.1) as a reduced system—one where the iteration occurs in the nullspace of B—without recourse to computing a nullspace basis. This reduction is given in §1.4. An appropriately preconditioned Krylov method is next applied to the reduced system and, by means of a change of variable, we obtain a variant of the Krylov method in which only products with A (and possibly A^T, depending on the method) are necessary as well as a single factorization of (1.3) for some user-chosen preconditioner G.

In this paper, we describe the projection of the Bi-CGSTAB and TFQMR families of methods for unsymmetric linear systems and apply our results to (1.1). The methods we consider are akin to so-called projection methods [11], which are sometimes regarded as being too expensive and only effective on systems in which A is diagonally dominant [52]. We hope that the remainder of this paper will correct that reputation by showing that efficient projections combined with the appropriate Krylov iteration make for a very competitive numerical method.

The rest of this paper is organized as follows. Previous work on the topic at hand is reviewed in §1.1. We briefly recall some background on Krylov methods in §1.2 and give our working assumptions in §1.3. A general method for projecting a Krylov method is presented in §1.4 and applied to two major Krylov methods. The first, TFQMR is covered in §2 and §3. The second, Bi-CGSTAB is covered in §4 and §5. We discuss preconditioning issues in §6 and implementation and numerical results in §7. Some conclusions and future research directions appear in §8.

1.1 Related Work

An approach suggested to be numerically advantageous in [30] to solve (1.1) in the symmetric case is to regularize the $(1,1)$ block to obtain the mathematically equivalent system

$$\begin{bmatrix} A + \rho B^T B & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} b + \rho B^T d \\ d \end{bmatrix}, \tag{1.4}$$

where $\rho > 0$ is a regularization parameter chosen sufficiently large that $A + \rho B^T B$ is nonsingular. The system (1.4) gives the first-order optimality conditions of the quadratic program

$$\underset{u \in \mathbb{R}^n}{\text{minimize}} \; -(b + \rho B^T d)^T u + \tfrac{1}{2} u^T (A + \rho B^T B) u \qquad \text{subject to } \; Bu = d. \tag{1.5}$$

The authors of [30] observe that the conditioning of the $(1,1)$ block of the coefficient matrix of (1.4) decreases as ρ departs from zero, but increases again as ρ becomes large. However, the condition number of the whole matrix increases monotonically with ρ. Overall, appropriate values for ρ are difficult to estimate and the authors suggest the approximation $\rho = \|A\|/\|B\|^2$.

After pivoting about the $(1,1)$ block of the coefficient matrix of (1.4), we are left with

$$B(A + \rho B^T B)^{-1} B^T p = B(A + \rho B^T B)^{-1} b - d, \tag{1.6}$$

and u may be recovered from

$$(A + \rho B^T B) u = b - B^T p. \tag{1.7}$$

Note that the coefficient matrix of (1.6) is the (negative) Schur complement of $A + \rho B^T B$ in the coefficient matrix of (1.4). The coefficient matrix of (1.6) is likely dense and contains the inverse of another potentially dense matrix. In order to avoid forming this matrix explicitly, iterative methods are favored to obtain p. Upon noting that (1.6) is a positive definite system, it is interpreted as the optimality conditions of the unconstrained quadratic program

$$\underset{p \in \mathbb{R}^m}{\text{minimize}} \; (d - B(A + \rho B^T B)^{-1} b)^T p + \tfrac{1}{2} p^T (B(A + \rho B^T B)^{-1} B^T) p, \tag{1.8}$$

which is solved by means of a steepest-descent method—an approach known as Uzawa's method [3]. It is typically applied for a (large) fixed value of $\rho > 0$ in which case the algorithm simplifies. The computation of the steepest-descent direction involves solutions of systems with $(A + \rho B^T B)$ as coefficient matrix, which can be very ill conditioned for large values of ρ. Iterative methods thus must be appropriately preconditioned. Uzawa's method generalizes to the case where A is unsymmetric [24] but essentially suffers from the same shortcomings.

A related approach consists in formulating the augmented-Lagrangian problem associated to (1.2) [24], resulting in the unconstrained quadratic program

$$\underset{u \in \mathbb{R}^n}{\text{minimize}} \; -(b + B^T(\rho_k d - p_k))^T u + \tfrac{1}{2} u^T (A + \rho_k B B^T) u, \tag{1.9}$$

where p_k is the current pressure—i.e., Lagrange multiplier—estimate and $\rho_k > 0$ is the current penalty parameter [7]. Note the difference between (1.5), which is constrained, and (1.9), which is unconstrained and differs in the linear term of the objective. Again, the matrix BB^T is likely to be (nearly) dense and to contribute adversely to the conditioning of the problem. The augmented-Lagrangian algorithm starts with an initial penalty parameter $\rho_0 > 0$ and initial estimates $u_0 \in \mathbb{R}^n$ and $p_0 \in \mathbb{R}^m$. At iteration k, u_{k+1} is obtained as a stationary point of (1.9), i.e.,

$$(A + \rho_k B^T B) u = b - B^T (p_k - \rho_k d)$$

and the estimate p_k is updated using

$$p_{k+1} = p_k + \rho_k (B u_{k+1} - d).$$

The parameter ρ_k is then increased and the process repeats.

Saddle-point systems such as (1.1) have generated a substantial amount of publications in the last decade, particularly in relation to preconditioning techniques, and we can only name a few here. Whenever A is nonsingular, some direct and iterative techniques are based on the splitting

$$\begin{bmatrix} I & 0 \\ B^T A^{-1} & I \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & -S \end{bmatrix} \begin{bmatrix} I & A^{-1}B \\ 0 & I \end{bmatrix}$$

of (1.1), where $S = -B^T A^{-1} B$ is the Schur complement of A [8]. A drawback of some iterative methods is their requirement for a symmetric positive definite preconditioner. To remedy this situation, [28] devise a specialized QMR iteration which accomodates arbitrary symmetric preconditioners. In [37] a successive over-relaxation scheme is developed. A primal regularization approach is taken in [30], where A is replaced by $A + BWB^T$ for some square matrix W which is typically a multiple of the identity. In the non-symmetric case, but under the assumption that the symmetric part of A is positive definite, [36] consider symmetric indefinite preconditioners for iterative methods applied to (1.1). Related preconditioning techniques also appear in [23]. In the symmetric case, [46], use the projection operator used in the present paper as an idefinite preconditioner. The definite reference regarding properties, applications and numerical methods for saddle-point systems is [8].

In the symmetric case, Bramley [11] proposes the *projection method* in terms of the orthogonal projection onto the nullspace of B written as $P = I - B^T(B^T)^\dagger$ where $(B^T)^\dagger$ is the Moore-Penrose pseudo-inverse of B^T. When $d = 0$ in (1.1), then (Pu, p) solves (1.1) if and only if $PAPu = Pb$ and $p = (B^T)^\dagger(b - APu) + v$ for some v such that $B^T v = 0$. Note that whenever B has full row rank, $P = I - B^T(BB^T)^{-1}B$. The resulting algorithm requires a Cholesky factorization of BB^T which is impractical in many cases given that the latter matrix may be dense. The author notes that the singular values satisfy $\sigma_{\min}(A) \leq \sigma_{\min}(PAP) \leq \sigma_{\max}(PAP) \leq \sigma_{\max}(A)$ so that the matrix PAP cannot have worse conditioning that A itself. Related numerical results also appear in [10].

The authors of [4] solve (1.1) when A is symmetric and positive definite and B is *totally unimodular*[1] by explicitly computing a nullspace basis Z and forming the reduced system. This strong structure allows to predict the sparsity pattern of $Z^T AZ$. The reduced system is then solved using a Cholesky factorization or a conjugate gradient iteration.

The idea of working in the nullspace of B is not new and can be traced back to at least 1968 [12]. The methods proposed here are related to the projection method but avoid its pitfalls by working with augmented systems. They fit in a general framework allowing to derive projected Krylov algorithms by inserting a projection step at a few places in a classic Krylov algorithm. The same goes for the implementation—the methods proposed here can be easily implemented by adding a few lines to an existing implementation of a Krylov method. The projected variants of the algorithms have a number of advantages over the direct application of a Krylov method to the augmented system, over a factorization of the full augmented system and over reduced approaches such as Uzawa's method and Bramley's projection method. Among them, we mention that only a very sparse projection matrix need be factorized, and matrix-vector products with the $(1, 1)$ block only are required. As a consequence, the memory requirements are modest and this approach lends itself to efficient implementation since in many applications, the matrix A need not be formed. For instance, it may be kept in memory in finite-element format and matrix-vector products can be assembled whenever necessary.

[1] A matrix is totally unimodular if it has full row rank and its entries are 0 and ± 1.

More recently, [38] considered the solution of fully non-symmetric saddle point problems, i.e., whose coefficient matrix has the form

$$\begin{bmatrix} A & B_2^T \\ B_1 & 0 \end{bmatrix},$$

where A is potentially non-symmetric. They propose a method based on a generalized Schur complement which reduces the problem to another smaller-scale non-symmetric saddle point problem and apply a method that they also name the projected Bi-CGSTAB method to the latter. The method is however costly as it requires computation of matrix-vector product with a Schur-complement-like matrix involving a generalized inverse, the calculation and storage of $(BB^T)^{-1}$ and the computation of both nullspace and a range-space bases. Incidentally, they only consider orthogonal projectors, using a multigrid scheme as preconditioning technique which turns out to be quite efficient in the fictitious domains applications that they set out to tackle. In this sense, the approach of [38] bears some resemblance to the projection method of [11].

We finally note that numerical algorithms for (1.1) can take advantage of the fact that in some applications, $d = 0$, especially when A is symmetric and positive definite. This is particularly important because the inverse of the coefficient matrix of (1.1), when it exists, may have very ill-conditioned blocks. However, those blocks become irrelevant because $d = 0$. We refer the reader to [8, §3.5] and references therein. We do not assume that $d = 0$ in this paper.

In the sequel, we present projected Krylov algorithms based on possibly oblique projectors, which serve the purpose of preconditioning the iteration. More importantly, the projected Krylov methods presented below require no computation of nullspace bases, no Schur complement and generate iterates lying in the nullspace of B naturally.

1.2 Background

For a given $n \times n$ nonsingular system $Mx = b$ of linear equations and x_k, an approximation to the solution $x^* = M^{-1}b$, we denote by $r_k = b - Mx_k$ the residual associated to x_k. A Krylov method is an iterative procedure that, at iteration k, approximates the solution x^* by x_k lying in the affine space

$$x_0 + \mathcal{K}_k(r_0, M),$$

where $x_0 \in \mathbb{R}^n$ is the initial guess and $\mathcal{K}_k(r_0, M)$ is the k-th Krylov subspace generated by M and r_0, defined by

$$\mathcal{K}_k(r_0, M) = \text{span}\{r_0, Mr_0, M^2r_0, \ldots, M^{k-1}r_0\}.$$

Whenever M is symmetric and positive definite, x_k may be defined by minimizing $\|x - x^*\|_M$ over $x_0 + \mathcal{K}_k(r_0, M)$, or, equivalently, by minimizing $-b^Tx + \frac{1}{2}x^TMx$ over the same subspace. The prime example of this class of methods is the conjugate gradient algorithm. When M is nonsymmetric, there is no such interpretation in terms of quadratic programming and Krylov methods seek to minimize some norm of the residual $\|r_k\| = \|b - Mx_k\|$ over $x_0 + \mathcal{K}_k(r_0, M)$—as in the GMRES method [51]—to minimize a surrogate residual—as in the QMR or TFQMR methods [29, 27]—or to attain a so-called Galerkin property $r_k \perp \mathcal{K}_k(\bar{r}_0, M^T)$ for some $\bar{r}_0 \in \mathbb{R}^n$ such that $r_0^T\bar{r}_0 \neq 0$—as in the bi-conjugate gradient algorithm [26], nicknamed Bi-CG. In all cases, the residual may be expressed as

$$r_k = p_k(M)r_0,$$

where $p_k(z)$ is a *residual* polynomial—a polynomial of degree k satisfying $p(0) = 1$. The above methods provably terminate in at most n iterations in exact arithmetic.

By construction, the bi-conjugate gradient involves related linear systems with M^T as coefficient matrix, and the corresponding residuals satisfy

$$\bar{r}_k = p_k(M^T)\bar{r}_0,$$

with the same residual polynomial $p_k(z)$. As this last identity suggests, Bi-CG requires matrix-vector products with M^T, which is a serious drawback for the applications we have in mind. In particular, Bi-CG requires products of the form $r_k^T\bar{r}_k$. However, since

$$r_k^T\bar{r}_k = (p_k(M)r_0)^T(p_k(M^T)\bar{r}_0) = (p_k(M)^2r_0)^T\bar{r}_0,$$

products with M^T can all be eliminated from the algorithm, and the result is an iteration in which

$$r_k = p_k(M)^2r_0, \tag{1.10}$$

nicknamed the *conjugate gradient squared*, or CGS, iteration [50]. A feature shared by all Krylov methods is that their convergence behavior is largely governed by the spectral features of the residual polynomial. Hence, a major disadvantage of CGS is that the condition number of the residual polynomial appears squared. In practice, this often translates into slow and erratic convergence patterns.

In [54], an attempt is made to smooth over this erratic convergence by replacing (1.10) by

$$r_k = q_k(M)p_k(M)r_0, \tag{1.11}$$

where the residual polynomial $q_k(z)$ is expressed as

$$q_k(z) = \prod_{i=1}^{k}(1 - \omega_i z).$$

For each index i, the parameter ω_i is selected so as to minimize r_i as a function of ω_i. The result is the *stabilized* Bi-CG method, nicknamed Bi-CGSTAB.

The CGS method is seldom used because of its rather unpredictable convergence properties. However, Bi-CGSTAB is probably the most widely used method of its class—the class of Krylov methods based on a Galerkin condition. There is unfortunately no convergence theory for Bi-CGSTAB. It nevertheless often outperforms GMRES in practice, especially when the dimension of the Krylov space becomes large and GMRES requires substantial storage, and has the advantages of having bounded storage requirements and rather low cost per iteration. We will use Bi-CGSTAB as one of our methods of choice in the next sections, when we construct projected Krylov methods. We recall the Bi-CGSTAB algorithm and taylor it to the solution of (1.1) in §4.

The transpose free variant of the quasi-minimum residual method, TFQMR [29] originates in fact not from the quasi-residual method QMR [27] but from CGS. Instead of minimizing the norm of the residual r_k, quasi-minimum residual methods minimize instead a full-rank linear transformation of r_k. In TFQMR, the residual is expressed in matrix form as

$$r_k = W_{k+1}\Omega_{k+1}^{-1}(f_{k+1} - H_k z),$$

where the columns of $W_{k+1} \in \mathbb{R}^{n\times(k+1)}$ are vectors of the form $p_i(A)^2 r_0$ and $p_i(A)p_{i-1}(A)r_0$, $\Omega_{k+1} = \text{diag}(\omega_1, \ldots, \omega_{k+1})$ is a diagonal scaling matrix, $f_{k+1} = \Omega_{k+1}e_1$ where e_1 is the first vector of the canonical basis, and H_k is an upper Hessenberg matrix. The scalars ω_i are positive *weights* and the method seeks a z so as to

$$\underset{z\in\mathbb{R}^k}{\text{minimize}} \ \|f_{k+1} - H_k z\|_2, \tag{1.12}$$

in the Euclidian norm. The objective of the least-squares problem (1.12) is called the quasi residual, hence the name *quasi minimization*. A solution z_k to (1.12) then yields a next iterate x_k. An appropriate choice of the weights ω_i yields a useful lower bound on $\|r_k\|$ as a function of the norm of the quasi residual and thus yields a natural stopping criterion. We recall and build upon the TFQMR algorithm in §2.

For a more complete discussion of Krylov methods, we refer the reader to the sources cited above and to the more general texts [5, 40, 48, 53].

1.3 Notation, Basic Results and Assumptions

In this section, we recall basic properties of systems of the form (1.1) and summarize our working assumptions for the remainder of the paper.

For a given matrix $M \in \mathbb{R}^{q\times r}$, let $\mathcal{N}(M) \subseteq \mathbb{R}^r$ and $\mathcal{R}(M) \subseteq \mathbb{R}^q$ denote the nullspace and the range space of M respectively. If $\mathbb{L} \subseteq \mathbb{R}^r$ is a subspace of $\mathbb{R}^r$, we use the notation

$$\mathcal{R}(M \mid \mathbb{L}) = \{u \in \mathbb{R}^q \mid u = Mv \text{ for some } v \in \mathbb{L}\}.$$

In other words, $\mathcal{R}(M \mid \mathbb{L})$ is the range space of the restriction of the operator M to the subspace $\mathbb{L}$. If the columns of the matrix N form a basis for $\mathbb{L}$, then $\mathcal{R}(M \mid \mathbb{L}) = \mathcal{R}(N^T M N)$.

We will consider several matrices of the form

$$K_M \equiv \begin{bmatrix} M & B^T \\ B & 0 \end{bmatrix}$$

for some matrix $M \in \mathbb{R}^{n\times n}$. In particular, K_A is the coefficient matrix of (1.1), K_I has the identity matrix in the $(1,1)$ block and we will refer to K_G for preconditioning systems where G satisfies appropriate properties.

Our basic working assumptions are stated in Assumption 1.1.

Assumption 1.1 *The matrices $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$ appearing in the coefficient matrix K_A are such that*

1. *B has full row rank,*
2. *$\mathcal{N}(A) \cap \mathcal{N}(B) = \{0\}$,*
3. *$\mathcal{R}(A \mid \mathcal{N}(B)) \cap \mathcal{R}(B^T) = \{0\}$.*

For the system (1.1) to be consistent, its coefficient matrix K should be nonsingular. The following result states that our basic assumption covers all possibilities.

Theorem 1.1 *The coefficient matrix of (1.1) is nonsingular if and only if Assumption 1.1 is satisfied.*

Proof. The proof is a special case of [38, Theorem 3.1]. □

Note that conditions 2 and 3 of Assumption 1.1 have meaningful implications for the numerical method that we have in mind. Condition 2 implies that the restriction of A to the nullspace of B is nonsingular. Therefore, it makes sense to work in $\mathcal{N}(B)$ and identify the component u of the solution to (1.1) in that subspace. This justifies the approach to be suggested in §1.4. In particular, A itself could be singular, but its restriction to $\mathcal{N}(B)$ must not. Condition 3 implies that there exist no nonzero $v \in \mathcal{N}(B)$ such that $Av = B^T p$ for some $p \in \mathbb{R}^m$. Finally, the full row-rank assumption on B is classical in this context.

We say that a matrix M is positive definite over a subspace $\mathbb{L}$ if $w^T M w > 0$ for all nonzero $w \in \mathbb{L}$. The following corollary will be useful in the sequel of the paper in terms of preconditioners.

Corollary 1.1 *If the matrix $M \in \mathbb{R}^{n\times n}$ is positive definite over the nullspace of B and if B has full row rank, then K_M is nonsingular.*

Proof. The first part of Assumption 1.1 is satisfied by assumption. The second part is satisfied since M is nonsingular over $\mathcal{N}(B)$. To verify the third part, suppose $z = Mw$ with $Bw = 0$. If it is also possible to write $z = B^T y$, then $w^T M w = w^T B^T y = 0$ so that $w = 0$ and $z = 0$. □

1.4 A Nullspace Approach

Let us now consider the augmented system (1.1). Let $Z \in \mathbb{R}^{q\times n}$ be a matrix whose rows form a basis for the nullspace of B. Any vector of $\mathbb{R}^n$, and in particular any solution u^* to (1.1) may be written

$$u^* = Z u_{\mathrm{Z}}^* + B^T u_{\mathrm{B}}^*, \tag{1.13}$$

so that the first block of equations of (1.1) turns into

$$AZ u_{\mathrm{Z}}^* + AB^T u_{\mathrm{B}}^* + B^T p = b. \tag{1.14}$$

Assuming that B has full row rank, premultiplying (1.13) by B and using the second block of equations of (1.1) yields

$$BB^T u_{\mathrm{B}}^* = d, \tag{1.15}$$

which uniquely determines u_{B}^*. By substituting into (1.14) and premultiplying with Z^T, we have the following relation,

$$Z^T AZ u_{\mathrm{Z}}^* = Z^T (b - AB^T u_{\mathrm{B}}^*), \tag{1.16}$$

in the unknown u_{Z}^*. Note that in (1.16), the coefficient matrix $Z^T AZ$ may be unsymmetric. Such a system might be solved iteratively by an appropriately preconditioned Krylov method. For computational reasons, we wish however to avoid the burden of computing a nullspace matrix Z, particularly as in Navier-Stokes problems, this would have to be done at each outer iteration.

In the following, we investigate modifications of the preconditioned TFQMR and Bi-CGSTAB methods using projection matrices of the form (1.3) and solving (1.16) without recourse to computing Z. The procedure is identical for both methods and can be applied to obtain a projected variant of any Krylov method. First, we write out the preconditioned Krylov algorithm for (1.16) with a preconditioner of the form $M = Z^T G Z$ where G is symmetric and such that M is positive definite. Next, we apply changes of variables by means of Z and Z^T. Preconditioning systems are replaced with the calculation of projection, which are carried out by solving systems with (1.3).

Our projected algorithm will involve solving linear systems with coefficient matrix K_Q for $Q = I$ and $Q = G$. We will write $\hat{g} = P_Q(g)$ when there exists a vector $\hat{h}$ such that

$$\begin{bmatrix} Q & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \hat{g} \\ \hat{h} \end{bmatrix} = \begin{bmatrix} g \\ 0 \end{bmatrix}. \tag{1.17}$$

With $Q = I$, P_I is the orthogonal projector onto the nullspace of B, whereas P_G is an oblique projection operator onto the nullspace of B, defined by the matrix G. Note that those projectors are equivalently described by the matrices

$$Z(Z^TZ)^{-1}Z^T \quad \text{and} \quad Z(Z^TGZ)^{-1}Z^T,$$

respectively. For the above expressions to represent projectors, they must be symmetric, which explains why G must be a symmetric approximation to A.

Note that although it is feasible to compute a sparse basis Z in practice, projected methods require the solution of linear systems with coefficient matrix $Z(Z^TZ)^{-1}Z^T$ or $Z(Z^TGZ)^{-1}Z^T$. In general, the latter matrices can be considerably denser than (1.3) and must be factorized because accurate projections are crucial in a practical implementation, as we discuss in §6.

2 The Preconditioned TFQMR Algorithm

In this section, we review the preconditioned TFQMR algorithm. We consider a generic linear system

$$Ax = b, \tag{2.1}$$

where $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$. It is well known that in many cases, it is far more efficient numerically to precondition this linear system before solving it. A two-sided preconditioner is a pair $\{R_1, R_2\}$ of nonsingular matrices which is used to transform (2.1) into the equivalent system

$$R_1^{-T}AR_2^{-1}x = R_1^{-T}b, \tag{2.2}$$

where we used the notation $R_1^{-T} = (R_1^{-1})^T$. The TFQMR method [29] applied to (2.2) is summarized in Algorithm 2.1. To distinguish the iterates generated by this algorithm from those of subsequent versions of it, we denote them with a bar.

It will be useful to derive a practical version of Algorithm 2.1 in terms of a preconditioner composed of R_1 and R_2. Indeed, we will subsequently use the same type of manipulations to derive a projected variant of this method.

In Algorithm 2.1, define the new variables $\bar{x} = R_2x$, $\bar{d} = R_2d$, $\bar{y} = R_2y$, $\bar{w} = R_1^{-T}w$, $\bar{v} = R_1^{-T}v$, and $\bar{r} = R_1^{-T}r$, and pick the vector $\tilde{r}_0$ as a vector of the form $R_2^{-T}\hat{r}_0$. A few elementary algebraic manipulations show that Algorithm 2.1 may be rephrased as Algorithm 2.2.

In Algorithm 2.2, it is important to remark that the notation M^{-1} does not by any means suggest that one should compute the inverse of M, merely that one must solve a linear system of equations with M as coefficient matrix. Note also that at each iteration, we must compute the preconditioned vectors $M^{-1}v_k$ and $M^{-1}w_k$ where $M = R_1^TR_2$ is the preconditioner. It appears however that the whole algorithm cannot be formulated entirely in terms of M because of the computation of τ_0 and θ_j. A similar observation had already been made for the Bi-CGSTAB method in [54]—to which we come back in §4—and we resolve it here in a similar way. We approximate $\tau_0 = \|R_1^{-T}r_0\|$ by $\|r_0\|$, which amounts to computing the residual norm in the non-preconditioned space. Similarly, θ_j is computed as $\|w_{j+1}\|/\tau_{j-1}$. These are, of course, approximations, but they have proved satisfactory in the context of Bi-CGSTAB. As we will see in §7, they yield good numerical performance in the context of TFQMR as well.

Algorithm 2.1 Preconditioned TFQMR (version 1)

Step 0. Choose $\bar{x}_0 \in \mathbb{R}^n$. Let $k = 0$ and

1. $\bar{w}_1 = \bar{y}_1 = \bar{r}_0 = R_1^{-T}(b - AR_2^{-1}\bar{x}_0)$,
2. $\bar{v}_0 = R_1^{-T}AR_2^{-1}\bar{y}_1$, $\bar{d}_0 = 0$, $\bar{\tau}_0 = \|\bar{r}_0\|$, $\bar{\theta}_0 = 0$, and $\bar{\eta}_0 = 0$,
3. Choose $\tilde{r}_0 \in \mathbb{R}^n$ such that $\bar{\rho}_0 \equiv \tilde{r}_0^T\bar{r}_0 \neq 0$.

Step 1. Compute $\bar{\alpha}_{k-1} = \dfrac{\bar{\rho}_{k-1}}{\tilde{r}_0^T\bar{v}_{k-1}}$,, and set $\bar{y}_{2k} = \bar{y}_{2k-1} - \bar{\alpha}_{k-1}\bar{v}_{k-1}$,

Step 2. For $j = 2k-1$ and $j = 2k$, set

1. $\bar{w}_{j+1} = \bar{w}_j - \bar{\alpha}_{k-1}R_1^{-T}AR_2^{-1}\bar{y}_j$,
2. $\bar{\theta}_j = \|\bar{w}_{j+1}\|/\bar{\tau}_{j-1}$, $\quad \bar{c}_j = 1/\sqrt{1+\bar{\theta}_j^2}$, $\quad \bar{\tau}_j = \bar{\tau}_{j-1}\bar{\theta}_j\bar{c}_j$, $\quad \bar{\eta}_j = \bar{c}_j^2\bar{\alpha}_{k-1}$,
3. $\bar{d}_j = \bar{y}_j + \dfrac{\bar{\theta}_{j-1}^2\bar{\eta}_{j-1}}{\bar{\alpha}_{k-1}}\bar{d}_{j-1}$,
4. $\bar{x}_j = \bar{x}_{j-1} + \bar{\eta}_j\bar{d}_j$.

Step 3. Set

1. $\bar{\rho}_k = \tilde{r}_0^T\bar{w}_{2k+1}$, and $\bar{\beta}_k = \bar{\rho}_k/\bar{\rho}_{k-1}$,
2. $\bar{y}_{2k+1} = \bar{w}_{2k+1} + \bar{\beta}_k\bar{y}_{2k}$,
3. $\bar{v}_k = R_1^{-T}AR_2^{-1}\bar{y}_{2k+1} + \bar{\beta}_k\left(R_1^{-T}AR_2^{-1}\bar{y}_{2k} + \bar{\beta}_k\bar{v}_{k-1}\right)$,
4. $k \leftarrow k+1$. Return to Step 1.

3 The Projected TFQMR Algorithm

We now wish to apply Algorithm 2.2 to (1.16) with the approximations given at the end of §2. It appears natural in this case to choose a preconditioner of the form $M = Z^TGZ$ where $G \in \mathbb{R}^{n\times n}$ is a symmetric appoximation to A such that M is positive definite. For instance, we might select $G = I$, the identity matrix of appropriate dimension.

We now rewrite Algorithm 2.2 with $M = Z^TGZ$ and denote all vectors with a superscript Z: r^{Z}, y^{Z}, w^{Z}, and so forth. To recover the notation of (1.1), we use the notation u instead of x and define the vectors

$$r_k^{\text{Z}} = Z^Tr_k,\ v_k^{\text{Z}} = Z^Tv_k,\ w_k^{\text{Z}} = Z^Tw_k, \quad \text{and} \quad Zy_k^{\text{Z}} = y_k,\ Zd_k^{\text{Z}} = d_k,\ u_k = Zu_k^{\text{Z}} + B^Tu_{\text{B}}^*. \quad (3.1)$$

Upon selecting an initial $u_0^{\text{Z}} \in \mathbb{R}^p$ where p is the dimension of the nullspace of B, the initial residual is given by

$$r_0^{\text{Z}} = Z^T(b - AB^Tu_{\text{B}}^*) - Z^TAZu_0^{\text{Z}} = Z^T\left(b - A(Zu_0^{\text{Z}} + B^Tu_{\text{B}}^*)\right) = Z^T(b - Au_0).$$

For consistency with (1.1), we thus simply state that instead of u_0^{Z}, we pick an initial $u_0 \in \mathbb{R}^n$, compute the initial residual $r_0 = b - Au_0$ and transform residuals using $r_0^{\text{Z}} = Z^Tr_0$.

Stage 4 of Step 2 in Algorithm 2.2 becomes

$$u_j^{\text{Z}} = u_{j-1}^{\text{Z}} + \eta_jd_j^{\text{Z}}.$$

Algorithm 2.2 Preconditioned TFQMR (version 2)

Step 0. Choose $x_0 \in \mathbb{R}^n$. Let $k = 0$ and

1. $r_0 = b - Ax_0$, $y_1 = M^{-1}r_0$, $w_1 = r_0$,
2. $v_0 = Ay_1$, $d_0 = 0$, $\tau_0 = \|R_1^{-T}r_0\|$, $\theta_0 = 0$, and $\eta_0 = 0$,
3. choose $\hat{r}_0 \in \mathbb{R}^n$ such that $\rho_0 \equiv \hat{r}_0^T r_0 \neq 0$.

Step 1. Compute $\alpha_{k-1} = \dfrac{\rho_{k-1}}{\hat{r}_0^T M^{-1} v_{k-1}}$, and set $y_{2k} = y_{2k-1} - \alpha_{k-1} M^{-1} v_{k-1}$,

Step 2. For $j = 2k-1$ and $j = 2k$, set

1. $w_{j+1} = w_j - \alpha_{k-1} A y_j$,
2. $\theta_j = \|R_1^{-T} w_{j+1}\|/\tau_{j-1}$, $\quad c_j = 1/\sqrt{1+\theta_j^2}$, $\quad \tau_j = \tau_{j-1}\theta_j c_j$, $\quad \eta_j = c_j^2 \alpha_{k-1}$,
3. $d_j = y_j + \frac{\theta_{j-1}^2 \eta_{j-1}}{\alpha_{k-1}} d_{j-1}$,
4. $x_j = x_{j-1} + \eta_j d_j$.

Step 3. Set

1. $\rho_k = \hat{r}_0^T M^{-1} w_{2k+1}$, and $\beta_k = \rho_k/\rho_{k-1}$,
2. $y_{2k+1} = M^{-1} w_{2k+1} + \beta_k y_{2k}$,
3. $v_k = Ay_{2k+1} + \beta_k \left(Ay_{2k} + \beta_k v_{k-1}\right)$,
4. $k \leftarrow k+1$. Return to Step 1.

From (3.1), we premultiply both sides of the latter equality by Z and add $B^T u_{\mathrm{B}}^*$ to both sides to obtain

$$u_j = u_{j-1} + \eta_j d_j.$$

The remaining transformations are similar. After the above change of notation and reworking of Algorithm 2.2, we obtain Algorithm 3.1.

A few comments on Algorithm 3.1 are in order. Firstly, note that Algorithm 3.1 is not directly applicable since Z occurs explicitly at two places, namely the computation of τ_0 and of θ_j. However, since the choice of Z is arbitrary, we may assume that Z was chosen to have orthonormal columns so that $Z^T Z = I$ and ZZ^T is the orthogonal projection onto the nullspace of B. As a consequence, we have

$$\tau_0 = \|Z^T r_0\| = \left(r_0^T Z Z^T r_0\right)^{\frac{1}{2}} = \left(r_0^T P_I(r_0)\right)^{\frac{1}{2}} = \|P_I(r_0)\|, \tag{3.2}$$

and similarly,

$$\theta_j = \|P_I(w_{j+1})\|/\tau_{j-1}. \tag{3.3}$$

Note that this is akin to our approximating τ_0 and θ_j in unpreconditioned space in the previous section.

Secondly, the particular choice for the fixed vector $\hat{r}_0$ allows to write

$$\rho_0 = \hat{r}_0^T r_0^{\mathrm{Z}} = \hat{r}_0^T Z^T r_0 = \tilde{r}_0^T Z Z^T r_0 = \tilde{r}_0^T P_I(r_0).$$

Algorithm 3.1 Projected Preconditioned TFQMR

Step 0. Choose $u_0 \in \mathbb{R}^n$. Let $k = 0$ and

1. $r_0 = b - Au_0, \quad y_1 = P_G(r_0), \quad w_1 = r_0,$
2. $v_0 = Ay_1, \quad d_0 = 0, \quad \tau_0 = \|Z^T r_0\|, \quad \theta_0 = 0, \quad$ and $\eta_0 = 0,$
3. choose $\hat{r}_0 = Z^T \tilde{r}_0$ for some $\tilde{r}_0 \neq 0$ so that $\rho_0 \equiv \tilde{r}_0^T P_I(r_0)$.

Step 1. Compute $\hat{v}_{k-1} = P_G(v_{k-1})$, $\alpha_{k-1} = \dfrac{\rho_{k-1}}{\tilde{r}_0^T \hat{v}_{k-1}}$, and set $y_{2k} = y_{2k-1} - \alpha_{k-1}\hat{v}_{k-1}$,

Step 2. For $j = 2k-1$ and $j = 2k$, set

1. $w_{j+1} = w_j - \alpha_{k-1} A y_j$, and compute $\hat{w}_{j+1} = P_G(w_{j+1})$,
2. $\theta_j = \|Z^T w_{j+1}\|/\tau_{j-1}, \quad c_j = 1/\sqrt{1+\theta_j^2}, \quad \tau_j = \tau_{j-1}\theta_j c_j, \quad \eta_j = c_j^2 \alpha_{k-1},$
3. $d_j = y_j + \dfrac{\theta_{j-1}^2 \eta_{j-1}}{\alpha_{k-1}} d_{j-1},$
4. $u_j = u_{j-1} + \eta_j d_j.$

Step 3. Set

1. $\rho_k = \tilde{r}_0^T \hat{w}_{2k+1}$, and $\beta_k = \rho_k/\rho_{k-1}$,
2. $y_{2k+1} = \hat{w}_{2k+1} + \beta_k y_{2k}$,
3. $v_k = Ay_{2k+1} + \beta_k \left(Ay_{2k} + \beta_k v_{k-1}\right)$,
4. $k \leftarrow k+1$.

A practical choice for the vector $\tilde{r}_0$ might be to simply set $\tilde{r}_0 = r_0$ for then we obtain $\rho_0 = \|P_I(r_0)\|^2$. If the latter were to vanish, then $r_0 = b - Au_0 \in \mathcal{N}(B)^\perp = \mathcal{R}(B^T)$ and u_0 would be a solution. Another possibility would be to select $\bar{r}_0 = (Z^T G Z)^{-1} Z^T \tilde{r}_0$, for in this case, $\rho_0 = \tilde{r}_0^T P_G(r_0)$. With $\tilde{r}_0 = r_0$, we are then requiring that r_0 not be orthogonal to $\mathcal{N}(B)$ in the preconditioned sense. Without loss of generality, we may thus assume that $\rho_0 \neq 0$.

Finally, because we defined the vectors w such that $w^z = Z^T w$, we have, in Step 3,

$$\rho_k = \hat{r}_0^T (Z^T G Z)^{-1} Z^T w_{2k+1} = \tilde{r}_0^T Z (Z^T G Z)^{-1} Z^T w_{2k+1} = \tilde{r}_0^T P_G(w_{2k+1}).$$

The above suggests that both K_I and K_G must be factorized whenever $G \neq I$, depending on the choice of $\tilde{r}_0$. After the factorizations, the additional cost of Algorithm 3.1 is then one factorization of the projection matrix K_G, one backsolve during initialization and three backsolves at each iteration. As K_G is symmetric and indefinite, a Bunch-Parlett-type factorization is appropriate and we use the multifrontal methods implemented in MA27 and MA57, part of the Harwell Subroutine Library [39]. Note that the freedom of choosing G may be also used so the factors of K_G are as sparse as possible. In particular, when $G = I$, the factors of K_I are potentially much sparser than the factors of the coefficient matrix of (1.1). In fact, their sparsity is entirely determined by that of B.

Of course, the present approach will only be advantageous if solving several systems of the form (1.17) is substantially easier than solving (1.1), which is often the case.

4 The Preconditioned Bi-CGSTAB Algorithm

The preconditioned Bi-CGSTAB method for (2.1) solves the equivalent system (2.2) by means of Algorithm 4.1 where the preconditioner is defined as $M = R_1^T R_2$ [54].

Algorithm 4.1 Preconditioned Bi-CGSTAB

Step 0. Choose $x_0 \in \mathbb{R}^n$. Compute $r_0 = b - Ax_0$ and choose $\bar{r}_0 \in \mathbb{R}^n$ such that $\bar{r}_0^T r_0 \neq 0$. Let $d_0 = r_0$.

Step 1. For $k = 0, 1, 2, \ldots,$

1. Solve $M\bar{d}_k = d_k$,
2. $\alpha_k = \dfrac{\bar{r}_0^T r_k}{\bar{r}_0^T A\bar{d}_k}$,
3. $s_k = r_k - \alpha_k A\bar{d}_k$,
4. Solve $M\bar{s}_k = s_k$,
5. $\omega_k = \dfrac{(R_1^{-T} s_k)^T (R_1^{-T} A\bar{s}_k)}{\|R_1^{-T} A\bar{s}_k\|_2^2}$,
6. $x_{k+1} = x_k + \alpha_k \bar{d}_k + \omega_k \bar{s}_k$,
7. $r_{k+1} = s_k - \omega_k A\bar{s}_k$,
8. $\beta_k = \dfrac{\alpha_k}{\omega_k} \dfrac{\bar{r}_0^T r_{k+1}}{\bar{r}_0^T r_k}$,
9. $d_{k+1} = r_{k+1} + \beta_k (d_k - \omega_k A\bar{d}_k)$.

As in Algorithm 2.2, the computation of ω_k in Algorithm 4.1 cannot be stated in terms of the preconditioner M only but requires information about how M is split. Since this is inconvenient, [54] suggests to use instead

$$\omega_k = \frac{s_k^T A\bar{s}_k}{\|A\bar{s}_k\|_2^2}, \tag{4.1}$$

and notes that in computational experiments, the latter formula behaves very similarly to the one involving R_1 as a split preconditioner. Once again, this inconvenience with Bi-CGSTAB also occurs in the projected version of the algorithm, given in the next section.

5 The Projected Bi-CGSTAB Algorithm

We apply Algorithm 4.1 with the modification (4.1) to the system (1.16) with a preconditioner of the form $M = Z^T G Z$ where G is such that M is positive definite. Let the vectors of the resulting formulation of Algorithm 4.1 be denoted u_k^{Z}, r_k^{Z}, d_k^{Z}, $\bar{d}_k^{\mathrm{Z}}$, and so forth. Upon defining vectors u_k, r_k, d_k, $\bar{d}_k$, s_k and $\bar{s}_k$ such that

$$r_k^{\mathrm{Z}} = Z^T r_k,\ d_k^{\mathrm{Z}} = Z^T d_k,\ s_k^{\mathrm{Z}} = Z^T s_k, \quad \text{and} \quad Z\bar{d}_k^{\mathrm{Z}} = \bar{d}_k,\ Z\bar{s}_k^{\mathrm{Z}} = \bar{s}_k,\ u_k = Zu_k^{\mathrm{Z}} + B^T u_{\mathrm{B}}^*, \tag{5.1}$$

Algorithm 5.1 Projected Preconditioned Bi-CGSTAB

Step 0. Choose $u_0 \in \mathbb{R}^n$. Compute $r_0 = b - Au_0$ and choose $\bar{r}_0 \in \mathbb{R}^n$ such that $(Z\bar{r}_0)^T r_0 \neq 0$. Let $d_0 = r_0$.

Step 1. For $k = 0, 1, 2, \ldots,$

1. Compute $\bar{d}_k = P_G(d_k)$,
2. $\alpha_k = \dfrac{(Z\bar{r}_0)^T r_k}{(Z\bar{r}_0)^T A\bar{d}_k}$,
3. $s_k = r_k - \alpha_k A\bar{d}_k$,
4. Compute $\bar{s}_k = P_G(s_k)$,
5. $\omega_k = \dfrac{(ZZ^T s_k)^T A\bar{s}_k}{\|Z^T A\bar{s}_k\|_2^2}$,
6. $u_{k+1} = u_k + \alpha_k \bar{d}_k + \omega_k \bar{s}_k$,
7. $r_{k+1} = s_k - \omega_k A\bar{s}_k$,
8. $\beta_k = \dfrac{\alpha_k}{\omega_k} \dfrac{(Z\bar{r}_0)^T r_{k+1}}{(Z\bar{r}_0)^T r_k}$,
9. $d_{k+1} = r_{k+1} + \beta_k (d_k - \omega_k A\bar{d}_k)$.

and deferring momentarily the choice of the fixed vector $\bar{r}_0$, we obtain Algorithm 5.1.

As in Algorithm 3.1, the computation of ω_k in Algorithm 5.1 still involves Z explicitly. Again, it may be simplified since without loss of generality, we may assume that the columns of Z are chosen to form an orthonormal basis for the nullspace of B, so that $Z^T Z = I$ and $ZZ^T = P_I$ is the orthogonal projection onto the nullspace of B. In this case,

$$\|Z^T A\bar{s}_k\|_2^2 = \bar{s}_k^T A^T ZZ^T A\bar{s}_k = (A\bar{s}_k)^T P_I (A\bar{s}_k) = \|P_I(A\bar{s}_k)\|_2^2,$$

so that the formula for ω_k simplifies to

$$\omega_k = \frac{P_I(s_k)^T A\bar{s}_k}{\|P_I(A\bar{s}_k)\|_2^2}. \tag{5.2}$$

When $G = I$, the computation of ω_k comes at the cost of one extra pair of forward and back solves. However, for more general preconditioners G, it appears that the computation of ω_k comes at the extra cost of factorizing K_I in addition to K_G. This drawback, shared with Algorithm 3.1, it at variance with the projected conjugate gradient algrithm [31]. We will see however that Algorithm 5.1 is very successful in practice and the above drawback is quickly overshadowed by its performance, even more so than in the case of Algorithm 3.1.

The only element of Algorithm 5.1 that has not been explained so far is the choice for the fixed vector $\bar{r}_0$. Note that the latter only appears in products of the form $(Z\bar{r}_0)^T v$ for various values of the vector v. Two particular choices are of interest in the projected algorithm. The first one is

$$\bar{r}_0 = Z^T \tilde{r}_0 \tag{5.3}$$

for some $\tilde{r}_0 \in \mathbb{R}^n$, for then

$$(Z\bar{r}_0)^T v = (ZZ^T\tilde{r}_0)^T v = P_I(\tilde{r}_0)^T v.$$

The initial requirement on $\bar{r}_0$ thus becomes $P_I(\tilde{r}_0)^T r_0 \neq 0$. Upon selecting $\tilde{r}_0 = r_0$, this requirement becomes equivalent to r_0 not being orthogonal to the nullspace of B, a situation which would indicate that u_0 is already a solution.

The second choice is

$$\bar{r}_0 = (Z^T G Z)^{-1} Z^T \tilde{r}_0 \tag{5.4}$$

for some $\tilde{r}_0 \in \mathbb{R}^n$. In this case,

$$(Z\bar{r}_0)^T v = \left(Z(Z^T G Z)^{-1} Z^T \tilde{r}_0\right)^T v = P_G(\tilde{r}_0)^T v.$$

The requirement $(Z\bar{r}_0)^T r_0 \neq 0$ becomes $P_G(\tilde{r}_0)^T r_0 \neq 0$ and the choice $\tilde{r}_0 = r_0$ imposes a lack of orthogonality to the nullspace of B in the preconditioned sense. The above discussion thus eliminates all occurences of Z in Algorithm 5.1.

We now examine under what conditions the projected variant of Bi-CGSTAB may break down. A breakdown could occur if the denominator of α_k vanishes, the denominator of ω_k or ω_k itself vanishes. Finally, it could occur if $\bar{r}_0^T r_k = 0$ in the denominator of β_k. The second possibility will not occur, since $A\bar{s}_k \in \mathcal{R}(A \mid \mathcal{N}(B))$ and by Assumption 1.1, this vector cannot be orthogonal to $\mathcal{N}(B)$ and therefore, its projection cannot vanish. Thus the denominator of ω_k in (5.2) cannot vanish. We have little control over the terms containing $\bar{r}_0$. If those cause a breakdown, a usual solution is to restart the iterations from the current point with a new value for $\bar{r}_0$. Finally, ω_k will vanish if $\bar{s}_k^T A\bar{s}_k = 0$, but, as above, this cannot be due to $A\bar{s}_k$ vanishing.

6 Preconditioning, Pressure Calculation and Cancellation

Algorithms 3.1 and 5.1 identify a solution u to (1.1) but do not identify the pressure component p of a solution. Once u has been found, we see from (1.1) that p is a solution to

$$B^T p = b - Au,$$

or, in other words, is uniquely determined by the equation

$$BB^T p = B(b - Au), \tag{6.1}$$

which can be interpreted as the first-order optimality conditions of the linear least-squares problem

$$p = \operatorname*{argmin}_{\lambda \in \mathbb{R}^m} \ \tfrac{1}{2}\|B^T\lambda + b - Au\|_2^2, \tag{6.2}$$

in Euclidian norm. One might go about solving (6.2) iteratively but in our context, it is useful to notice that (6.1) may be equivalenty expressed in terms of the augmented system

$$\begin{bmatrix} I & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} w \\ p \end{bmatrix} = \begin{bmatrix} b - Au \\ 0 \end{bmatrix} \tag{6.3}$$

for some auxilliary vector w. Therefore, if either Algorithm 3.1 or 5.1 is applied, the latter calculation of p costs one extra matrix-vector product, one extra forward solve and one extra backsolve since the augmented matrix has already been factorized.

Any preconditioner G that is positive definite on $\mathcal{N}(B)$ induces a semi-norm $\|g\|_G$ defined by

$$\|g\|_G^2 = w^T g,$$

where w solves the system

$$\begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} w \\ y \end{bmatrix} = \begin{bmatrix} g \\ 0 \end{bmatrix},$$

i.e., $w = P_G(g)$. In other words, $\|g\|_G = \|P_G(g)\|$. This induced semi-norm measures deviation from $\mathcal{R}(B^T)$ and vanishes if and only if $g \in \mathcal{R}(B^T)$. Effectively, it acts as a norm on $\mathcal{N}(B)$. For any $g \in \mathbb{R}^n$ and any $p \in \mathbb{R}^m$, it satisfies $\|g\|_G = \|g + B^T p\|_G$.

If Algorithm 3.1 is applied with a preconditioner G, the calculation of the final value of p need not be at the cost of a factorization of P_I. Indeed, instead of (6.3), we use

$$\begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} w \\ p \end{bmatrix} = \begin{bmatrix} b - Au \\ 0 \end{bmatrix}, \tag{6.4}$$

which yields p as the solution to the "weighted" linear least-squares problem

$$p = \operatorname*{argmin}_{\lambda \in \mathbb{R}^m} \tfrac{1}{2}\|B^T\lambda + b - Au\|_{G^{-1}}^2, \tag{6.5}$$

in the G^{-1} seminorm, the dual seminorm to $\|\cdot\|_G$. Indeed, (6.4) yields

$$w + G^{-1}B^T p = G^{-1}(b - Au), \qquad \text{and} \qquad Bw = 0,$$

and therefore

$$BG^{-1}B^T p = BG^{-1}(b - Au),$$

which are the first-order optimality conditions of (6.5).

Similarly, the value u_{B}^* satisfying (1.15) can be computed using

$$\begin{bmatrix} I & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} w \\ -u_{\mathrm{B}}^* \end{bmatrix} = \begin{bmatrix} 0 \\ d \end{bmatrix}$$

since the latter system yields $w = B^T u_{\mathrm{B}}^*$ and $Bw = d$.

As observed in [31] in the case of the projected conjugate gradient algorithm, severe numerical cancellation may occur during the projection computations. Indeed, as convergence occurs, g will align itself more and more with $\mathcal{R}(B^T)$, i.e., become orthogonal to $\mathcal{N}(B)$ and therefore, in (1.17), $\hat{g}$ will converge to zero. However, g itself will often remain substantial and therefore $\hat{h}$ will stay bounded away from zero. Suppose for simplicity that $Q = G = I$. Then (1.17) yields

$$\hat{g} = g - B^T\hat{h},$$

which suggests why numerical cancellation is likely to occur. The backward error analysis in [9] shows that

$$\frac{\|\hat{g} - \hat{g}_x\|}{\|\hat{g}_x\|} \leq \eta\epsilon_m(\sigma_{\max}(B) + \kappa(B))\frac{\|\hat{h}_x\|}{\|\hat{g}_x\|},$$

where η is the product of a low-degree polynomial in $n + m$ with the growth factor from a Bunch-Parlett factorization of K_I, ϵ_M is the machine epsilon, $\sigma_{\max}(B)$ is the largest singular value of B, $\kappa(B)$ is the condition number of B, and quantities with a subscript x denote

the *exact* value, as would result from exact arithmetic, while non-subscripted quantities are computed values. Therefore, whenever $\kappa(B)$ is large or $\|\hat{h}_x\| \approx \|g\|$, i.e., $\|\hat{g}_x\| \ll \|\hat{h}_x\|$, the relative error in the computed projection could potentially be large.

Different strategies are possible to improve numerical stability. On the one hand, one may attempt to improve the quality of the projection with iterative refinement. One the other hand, the inaccurate $\hat{g}$ could in turn be projected as suggested in [14, 31]. However, the former may not converge if $\kappa(K(G))$ is too large and costs extra matrix-vector products, and the both cost extra backsolves. As it turns out, there is a cheaper remedy to the cancellation issue suggested in [31], which consists in a residual update strategy possibly followed by one or more steps of iterative refinement. The main idea is that in (1.17), in exact arithmetic, $\hat{g}_x$ is unaffected if g is replaced by $g - B^T\lambda$ for some $\lambda \in \mathbb{R}^m$. It thus suffices to pick a λ such that $\|\hat{g}\|$ is closer to $\|g\|$. A convenient approximation is obtained by solving the linear least-squares problem

$$\underset{\lambda\in\mathbb{R}^m}{\text{minimize}} \;\; \tfrac{1}{2}\|g - B^T\lambda\|^2_{G^{-1}}. \tag{6.6}$$

since the solution to (6.6) is precisely $\lambda = \hat{h}$. Unfortunately, this value is obtained after the fact and in order to use it, one would have to project a second time, with g replaced by $g - B^T\hat{h}$. This inefficiency can be remedied cheaply by using slightly outdated information.

We now examine how the above discussion applied to Algorithms 3.1 and 5.1.

6.1 Stabilizing the Projected TFQMR Algorithm

In Algorithm 2.2, the vector w_{2k+1} in Step 3 is the kth residual from the CGS process [29], i.e.,

$$w_{2k+1} = r_k^{\text{CGS}}.$$

Therefore, in Algorithm 3.1, we expect $P_G(w_{2k+1})$ to become small and following the discussion of the above section, it is the latter projection that is subject to numerical errors. We thus consider w_{2k+1} in the role of g and $\hat{w}_{2k+1}$ in the role of $\hat{g}$, and should redefine w_{2k+1} as $w_{2k+1} - B^T\lambda$ for some $\lambda \in \mathbb{R}^m$ prior to projection. Let the projection of w_{2k+1} be given by

$$\begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \hat{w}_{2k+1} \\ h_{2k+1} \end{bmatrix} = \begin{bmatrix} w_{2k+1} \\ 0 \end{bmatrix}. \tag{6.7}$$

Instead of redefining w_{2k+1} as the ideal $w_{2k+1} - B^T h_{2k+1}$ when solving (6.7), we redefine it as $w_{2k+1} - B^T h_{2k-1}$, where h_{2k-1} was obtained as a by-product of the projection of w_{2k-1}. In the numerical implementation, this stabilization step turned out to be crucial.

6.2 Stabilizing the Projected Bi-CGSTAB Algorithm

In Algorithm 5.1, we consider s_k in the role of g and $\bar{s}_k = P_G(s_k)$ in the role of $\hat{g}$. From the previous discussion, we should redefine s_k as $s_k - B^T\lambda$ for some $\lambda \in \mathbb{R}^m$ before computing $\bar{s}_k$ at stage 4. As in §6.1, let the projection of s_k be given by

$$\begin{bmatrix} G & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \bar{s}_k \\ h_k \end{bmatrix} = \begin{bmatrix} s_k \\ 0 \end{bmatrix}. \tag{6.8}$$

Again, instead of redefining s_k as the ideal $s_k - B^T h_k$ when solving (6.8), we redefine it as $s_k - B^T h_{k-1}$, where h_{k-1} was obtained as a by-product of the projection of s_{k-1} and thus comes at no extra cost. At variance with §6.1, this stabilization of the algorithm greatly enhances its numerical performance and in our experience, its net effect is to have the algorithm terminate in fewer iterations.

7 Implementation and Numerical Results

Algorithms 3.1 and 5.1 are implemented as flexible Fortran 95 modules named PTFQMR and PBCGSTAB in the forthcoming projected Krylov solvers library [43]. The modules crucially rely upon either of the HSL [39] packages MA47 [19, 20] and MA57 [21] to factorize the projection matrices P_I and P_G. The MA47 and MA57 packages are evolutions of the earlier MA27 [17, 18], which implements a stable multi-frontal symmetric indefinite factorization. Both can take advantage of the augmented structure of P_I and P_G—namely the presence of a zero diagonal block—via their pivoting strategy and in particular their use of so-called *oxo* pivots [20]. Note that all three packages are freely available for academic use as part of the *Harwell Subroutine Library*, accessible from `hsl.rl.ac.uk/hsl2007/hsl20074researchers.html`. The ordering scheme used by MA57 to promote sparsity of the factors is one of the approximate minimum degree ordering (AMD) [1, 2], or the multilevel recursive-bisection and k-way partitioning routine METIS [41, 42]. Depending on the matrix size and column density, MA57 automatically selects an appropriate ordering. For numerical experience supporting the decision criteria, see [32]. In the tests below, the matrix A is formed but is only used for matrix-vector products. In a practical setting, A may be held in finite-element format and never be assembled.

In the default implementation, we choose $u_0 = 0$ so that $r_0 = b$. The choice of $\bar{r}_0$ is guided by the transformation used to convert Algorithm 2.2 into Algorithm 3.1 and Algorithm 4.1 into Algorithm 5.1.

The stopping test implemented by default in PTFQMR is classical. The stopping tolerance is fixed at $\epsilon = \epsilon_a + \epsilon_r \tau_0$, where ϵ_a and ϵ_r are absolute and relative stopping tolerances respectively. The quasi residual τ_k is monitored along the iterations and we declare convergence if $\sqrt{2k}\,\tau_k \leq \epsilon$ (or $\sqrt{2k+1}\,\tau_k \leq \epsilon$, depending on the index of the inner loop of Algorithm 3.1) or $\|P_G(w_k)\|_2 \leq \epsilon$.

The stopping test implemented by default in PBCGSTAB can be triggered by the Bi-CG residual vector s_k or by the residual vector r_k. On the one hand, the iteration stops whenever

$$(s_k^T P_G(s_k))^{\frac{1}{2}} = \|P_G(s_k)\| \leq \epsilon_a + \epsilon_r (\bar{r}_0^T r_0)^{\frac{1}{2}}.$$

In this case, the iterate u_k must still be updated using

$$u_{k+1} = u_k + \alpha_k \bar{d}_k.$$

On the other hand, we also stop the iteration whenever the Galerkin-like condition

$$\bar{r}_0^T r_k < 10^{-12}\ \bar{r}_0^T r_0 \quad \text{and} \quad (r_k^T P_G(r_k))^{\frac{1}{2}} \leq \epsilon_a + \epsilon_r (\bar{r}_0^T r_0)^{\frac{1}{2}}$$

is satisfied.

By default, ϵ_a and ϵ_r are set in both implementations to 10^{-6} but can of course be ajusted by the user.

Finally, in both algorithms, the iteration terminates if the total number of matrix-vector products exceeds $q\ n_A$, where n_A is the order of the $(1,1)$ block matrix A. In our tests, we chose $q = 3$ in PTFQMR while $q = 2$ in PBCGSTAB.

We compare the projected Bi-CGSTAB approach with a direct LU factorization of (1.1). The comparison is based on the total solution time and memory requirements. The LU factorization is realized by means of the UMFPACK package [15].

Since the present study is not a study of preconditioners, we choose $G = I$ in all our tests and defer the examination of appropriate preconditioners to future research. All tests below were performed with the Intel compiler 10.0 on a 2.4 GHz Intel Core 2 Duo Apple laptop with 4 GB of memory.

The test problems used below arise from the discretization of the Stokes or Navier-Stokes equations describing the flow of an incompressible viscous Newtonian fluid in a domain Ω which contains a (potentially) moving subdomain Ω^*, i.e.,

$$\left(\frac{\partial \vec{v}}{\partial t} + \vec{v} \cdot \text{grad } \vec{v}\right) + \mu \nabla^2 \vec{v} + \text{grad } p = \vec{f} \qquad \text{in } \Omega, \tag{7.1a}$$

$$\text{div } \vec{v} = 0 \qquad \text{in } \Omega, \tag{7.1b}$$

$$\vec{v} = \vec{v^*} \qquad \text{on } \Gamma^*, \tag{7.1c}$$

where $\vec{v}$ is the velocity field, p is the pressure, μ is the viscosity, ∇^2 is the Laplacian operator, $\vec{f}$ is a body force, Γ^* is the boundary of Ω^*, and $\vec{v^*}$ is the prescribed velocity on Γ^*. In our problems, Ω^* represents an obstacle modeled by means of the fictitious domains method—see, e.g., [34].

Example 7.1 In this test problem, the Newton system encountered at iterations 5, 7 and 8 of a Navier-Stokes solver was output to file. The continuous problems concerns the study of the flow of two immiscible fluids in a cavity. The sparsity pattern of the coefficient matrix is show in the leftmost plot of Figure 7.1, where it is clear that the $(1,1)$ block contains a number of rows that make it unsymmetric. The three systems have nearly identical properties. For instance, in the system encountered at iteration 8, the $(1,1)$ block is of order $7,092$ with $168,054$ nonzero elements—making it 0.28% dense in K_A. The $(1,2)$ block has size 669×7092 with $21,687$ nonzero elements—making it 0.036% dense in K_A. We thus see that it is A that is responsible for most of the density of K_A. The total number of nonzero elements to keep in memory for this matrix of order $7,761$ is thus $211,428$—an overall density of 0.351%. On the other hand, the projection matrix P_I, being symmetric, we only store its lower triangle, which amounts to $28,779$ nonzeros, a density of 0.048%. Its factors, as computed by MA57, have $55,316$ nonzeros—a density of 0.092%—a minor example of fill-in. The factorization of P_I was realized in 0.10 seconds.

In the LU factorization, the L factor was found to have $568,780$ entries and U was found to have $543,144$, densities of 0.944% and 0.902% respectively, a substantially more important fill-in. In this small-scale example, the LU factorization was performed in a mere 0.33 seconds.

Algorithm 3.1 performed $13,037$ matrix-vector products before declaring convergence in 60.22 seconds, for a total running time of 60.32 seconds. Algorithm 5.1 performed $2,464$ matrix-vector products before reaching convergence in 13.18 seconds, for a total running time of 13.28 seconds. The bottom plots of Figure 7.1 show the quasi-residual history of PTFQMR (left) and the residual history of PBCGTAB (right) on this linear system. The plot of Figure 7.1 are representative of the residual history on all problems that we tested.

At the end of the iterations, we record the norm of the PTFQMR quasi residual, the norm of the recurred PBCGSTAB residual vector, the relative residual of (1.1), defined by

$$\frac{\left\| \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u^* \\ p^* \end{bmatrix} - \begin{bmatrix} b \\ d \end{bmatrix} \right\|}{\left\| \begin{bmatrix} b \\ d \end{bmatrix} \right\|},$$

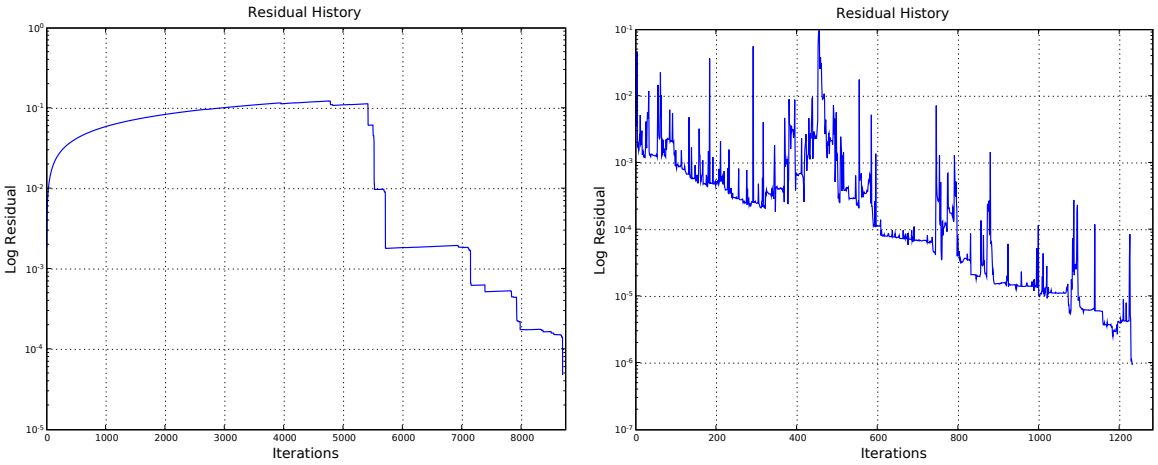

Figure 7.1: History of the PTFQMR quasi-residual (left) and of the PBCGTAB recurred residual (right).

and the relative error with the solution found with the *LU* factorization. This data is reported in Tables 7.1 and 7.2 for the three problems.

Although the relative errors with the solution found by the LU factorization only have 3 or 4 significant digits in Tables 7.1 and 7.2, we wish to stress that it is possible to improve the estimates in PBCGSTAB. For instance, upon setting $\epsilon_a = 10^{-12}$ and $\epsilon_r = 10^{-11}$, we obtained a relative error of 8.7e−07 in 5,332 matrix-vector products, which was realized in 29.59 seconds on the iter-8 problem. We were not able to attain the same accuracy with PTFQMR. It is imporant to realize however that on the one hand, no preconditioner was used in the above experiments, and on the other hand, these iterative solvers are meant to be included in a Newton-Krylov process. In such a process, the stopping tolerance to which the linear systems are solved decreases as the iterations proceed. For instance, results obtained in a Newton-Krylov process based on PBCGSTAB yield much improvement over a Newton-LU process with

Table 7.1: Statistics on the PTFQMR runs on the three problems arising at iterations 5, 7 and 8 in a Navier-Stokes solver. The factorization and solve times are in seconds.

Instance	nnz(A)	nnz(B)	Matvecs	Quasi Res.	Rel. Res.	Rel. Err.	Fact	Solve
iter-5	168,104	21,688	6,483	1.7e−05	2.4e−06	1.1e−04	0.11	29.96
iter-7	168,031	21,691	12,054	6.1e−05	2.9e−05	4.3e−03	0.11	55.93
iter-8	168,055	21,687	13,037	4.9e−05	1.6e−05	2.1e−04	0.11	60.22

Table 7.2: Statistics on the PBCGSTAB runs on the three problems arising at iterations 5, 7 and 8 in a Navier-Stokes solver. The factorization and solve times are in seconds.

Instance	nnz(A)	nnz(B)	Matvecs	Recur. Res.	Rel. Res.	Rel. Err.	Fact	Solve
iter-5	168,104	21,688	4,200	9.7e−07	2.7e−03	1.9e−03	0.11	22.54
iter-7	168,031	21,691	2,710	6.7e−07	1.1e−05	3.9e−03	0.11	14.56
iter-8	168,055	21,687	2,464	9.5e−07	2.7e−05	3.7e−03	0.10	13.18

an adaptive stopping tolerance defined by $\min\{0.1\|r_k\|, \|r_k\|^{1.5}\}$ at iteration k, where r_k is the right-hand side of (1.1). This is however the subject of ongoing research [16].

For comparison purposes, we ran the iterative Krylov methods as implemented in SPARSKIT [47] directly on the augmented system, without preconditioning. The results are summarized in Table 7.3. In each case, the iteration limit was set to $6n$ where n is the order of the system. In the present example, $6n = 46566$. We refer the reader to [5, 47] for more details on the Krylov algorithms and the various stopping criteria implemented in SPARSKIT. All methods reached the iteration limit in 21.95 seconds or more and the smallest residual has a magnitude of about 1.53e−05.

Example 7.2 The same problem as in Example 7.1 was discretized more finely to yield linear systems of order 32,965. Similar experiments were performed and results are reported in Tables 7.4 and 7.5. We were not able to obtain the desired accuracy on the iter-2 problem with PTFQMR. The quasi residual lingers at a value around 10^{-4} early on in the iterations and slowly increases.

Table 7.3: Summary results of the Krylov methods implemented in SPARSKIT on the linear system occurring at iteration 8 of a Navier-Stokes process. Residuals are absolute, as recurred by the method. Workspace is the total number of double precision values to be kept in memory for each solver. All solvers reach the maximal allowed number of iterations, in this case, $6n$.

Solver	Iterations	Residual	Workspace	Time (s)
BCG	46566	0.1329880e+02	54327	21.95
DBCG	46567	0.4489614e+01	85371	23.02
CGNR	46567	0.1537764e−04	38805	21.25
BCGSTAB	46567	0.3111281e−04	62088	21.95
TFQMR	46567	0.8147863e−04	85371	23.11
FOM	46566	0.4278482e101	132108	28.89
GMRES	46566	0.6374488e+97	132108	28.33
FGMRES	46566	0.3499084e+96	248519	28.63
DQGMRES	46566	0.1387398e+95	256177	44.45

Table 7.4: Statistics on the PTFQMR runs on the four problems arising at iterations 2 and 6 in a Navier-Stokes solver. The factorization and solve times are in seconds. Horizontal bars indicate a failure.

Instance	nnz(A)	nnz(B)	Matvecs	Quasi Res.	Rel. Res.	Rel. Err.	Fact	Solve
iter-2	541,545	86,083	—	—	—	—	0.60	—
iter-6	795,404	86,063	2,723	1.7e−05	7.8e−07	5.1e−05	0.60	58.40

Table 7.5: Statistics on the PBCGSTAB runs on the three problems arising at iterations 2 and 6 in a Navier-Stokes solver. The factorization and solve times are in seconds.

Instance	nnz(A)	nnz(B)	Matvecs	Recur. Res.	Rel. Res.	Rel. Err.	Fact	Solve
iter-2	541,545	86,083	2,786	1.1e−06	9.0e−01	2.0e−03	0.60	59.16
iter-6	795,404	86,063	976	9.0e−07	1.7e−04	2.5e−04	0.61	24.11

Our last test problem is the Von Karmann vortices problem: two immiscible fluids flow in a rectangular cavity around a fixed circular obstacle. The latter is entirely immersed in one of the fluids. The obstacle is modeled by means of the fictitious domains method with 200 control points. The $(1,1)$ block has size $87,196$ and $2,084,752$ nonzero elements, and the $(2,1)$ block has $7,579$ rows with $274,087$ nonzeros. Factorizing the augmented projection matrix required 3.00 seconds and the triangular factor has $587,336$ nonzero elements. The relatively high number of control points used in the fictitious domains method introduced near rank deficiency in the $(2,2)$ block. For this reason, we could not compare our solution with that of an LU factorization since the latter was unable to determine the p component of the solution—UMFPACK was however able to return a complete solution vector u. The PBCGSTAB method converged after 66 matrix-vector products with residual 7.7e−07 in 5.21 seconds to a solution whose u component was within 1.0e−04 of that of the partial factorization.

The PTFQMR method converged in 492 matrix-vector products in 36.18 seconds with a quasi residual of 1.2e−05 to a solution whose u component was within 6.8e−05 of that of the partial factorization.

In order to cope with the near rank deficiency of B, the $(2,2)$ block of the projection matrix P_I was set to $-\delta I$ for some small value of $\delta > 0$. In the present experiment, we selected $\delta = 10^{-8}$. In the context of Newton-Krylov solvers, this regularization strategy has proved beneficial on most problems using the fictitious domains method.

8 Conclusion

We have presented a general framework to project a Krylov method, where the projection is understood in the sense of solving an augmented system of the form (1.1) while only performing matrix-vector products with the $(1,1)$ block. The framework requires that the preconditioner be symmetric even if the original system is not. The method can be implemented by making very few changes to an existing implementation of the TFQMR, Bi-CGSTAB, or any other Krylov iterative method and by interacting with a package for symmetric indefinite factorization to solve systems of the form (1.17). The projected TFQMR and Bi-CGSTAB algorithms were implemented as part of the upcoming library of projected Krylov methods [43]. In our tests, PTFQMR appeared substantially slower and less robust that PBCGSTAB. In future research, we will investigate projected variants of GMRES and its restarted variants for inclusion in the library.

The projected Bi-CGSTAB implementation has been used to solve fluid flow problems with moving obstacles when embedded in a Newton-Krylov algorithm. A complete analysis of the general framework for nonlinear systems is deferred to [16].

Refinements of our implementation are possible, based on modified versions of Bi-CGSTAB for unsymmetric matrices with complex spectrum and for complex matrices [35, 49]. Moreover, our assumption that B has full row rank does not necessarily hold in practice. In this case, regularization methods, which perturb the zero $(2,2)$ block of $K(A)$, are natural, but conflict with the projection property. We are currently investigating this avenue.

Much remains to be done to take into account the case where B is either too dense or too expensive to form in order to be able to afford a symmetric indefinite factorization of $K(I)$ or $K(G)$. The investigation of matrix-free methods for the Navier-Stokes problem in that context will be the subject of future research.

As an additional extension of the present work, we will mention the solution of fully non-symmetric saddle-point problems by means of projected Krylov methods.

References

[1] P. R. Amestoy, T. A. Davis, and I. S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17:886–905, 1996.

[2] P. R. Amestoy, T. A. Davis, and I. S. Duff. AMD, an approximate minimum degree ordering algorithm. *Transactions of the ACM on Mathematical Software*, 30(3):381–388, 2004.

[3] K. Arrow, L. Hurwicz, and H. Uzawa. *Studies in Nonlinear Programming*. Stanford University Press, Stanford, CA, 1958.

[4] M. Arioli and G. Manzini. Null space algorithms for solving augmented systems arising in the the mixed finite element approximation of saddle point problems. In N. E. Mastorakis, editor, *CSCC 2000, MCP 2000, MCME 2000 Proceedings*, pages 2851–2856, 2000.

[5] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. M. 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, Philadelphia, USA, 2nd edition, 1993.

[6] Å. Björck and T. Elfving. Accelerated projection methods for computing pseudo-inverse solutions of systems of linear equations. *BIT*, 19:145–163, 1979.

[7] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, London, 1982.

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

[9] Å. Björck. Pivoting and stability in augmented systems. In D. F. Griffiths and G. A. Watson, editors, *Numerical Analysis 1991*, number 260 in Pitman Res. Notes Math. Ser., pages 1–16. Longman Scientific and Technical, Harlow, UK, 1992.

[10] R. Bramley and D. Pelletier. Iterative methods and formulations for incompressible FEM flow solvers. In *34th Aerospace Sciences Meeting and Exhibit*, pages 15–18, Reno, NV, 1996. AIAA.

[11] R. Bramley. An orthogonal projection algorithm for linear stokes problems. Technical Report 1190, CSRD, Univ. of Illinois, Urbana Champain IL, USA, 1992.

[12] A. Chorin. Numerical solution of the navier-stokes equations. *Mathematics of Computation*, 22:745–762, 1968.

[13] T. F. Coleman. Linearly constrained optimization and projected preconditioned conjugate gradients. In J. Lewis, editor, *Proceedings of the Fifth SIAM conference on Applied Linear Algebra*, pages 118–122, Philadelphia, USA, 1994. SIAM.

[14] T. F. Coleman and A. Verma. A preconditioned conjugate gradient approach to linear equality constrained optimization. *Computational Optimization and Applications*, 20:61–72, 2000.

[15] T. A. Davis. Algorithm 832: UMFPACK, an unsymmetric-pattern multifrontal method. *Transactions of the ACM on Mathematical Software*, 30(2):196–199, 2004.

[16] S. Dufour, A. Fidahoussen, and D. Orban. Iterative solvers for the modeling of free-surface flows with moving boundaries. Cahier du GERAD, GERAD, Montréal, Québec, Canada, 2008. In preparation.

[17] I. S. Duff and J. K. Reid. MA27—a set of Fortran subroutines for solving sparse symmetric sets of linear equations. Report AERE R10533, HMSO, London, UK, 1982.

[18] I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear systems. *Transactions of the ACM on Mathematical Software*, 9:302–325, 1983.

[19] I. S. Duff and J. K. Reid. MA47, a Fortran code for direct solution of indefinite sparse symmetric linear systems. Technical Report RAL-95-001, Rutherford Appleton Laboratory, Chilton, Oxfordshire, UK, 1995.

[20] I. S. Duff and J. K. Reid. Exploiting zeros on the diagonal in the direct solution of indefinite sparse symmetric linear systems. *Transactions of the ACM on Mathematical Software*, 22(2):227–257, 1996.

[21] I. S. Duff. MA57—a code for the solution of sparse Symmetric Definite and Indefinite Systems. *Transactions of the ACM on Mathematical Software*, 30(2):118–144, 2004.

[22] H. C. Elman. *Iterative Methods for Large, Sparse, Nonsymmetric Systems of Linear Equations*. Ph.D. thesis, Yale University, New Haven, CT, 1982.

[23] H. Elman and D. Silvester. Fast nonsymmetric iterations and preconditioning for navier-stokes equations. *SIAM Journal on Scientific Computing*, 17(1):33–46, 1996.

[24] M. Fortin and R. Glowinski. *Augmented Lagrangian Methods: Applications to the Numerical Solution of Boundary-Value Problems*. North-Holland, Amsterdam, 1983.

[25] R. W. Freund, G. H. Golub, and N. M. Nachtigal. Iterative solution of linear systems. *Acta Numerica*, 1:57–100, 1991.

[26] R. Fletcher. Conjugate gradient methods for indefinite systems. In G. Watson, editor, *Numerical Analysis Dundee 1975*, pages 73–89. Springer Verlag, Berlin, New-York, 1976.

[27] R. W. Freund and N. M. Nachitgal. QMR: a quasi-minimal residual method for non-hermitian linear systems. *Numerische Mathematik*, 60:315–339, 1991.

[28] R. W. Freund and N. M. Nachtigal. A new krylov-subspace method for symmetric indefinite linear systems. Technical Report ORNL/TM-12754, Oak Ridge National Laboratory, Oak Ridge, TN, 1994.

[29] R. W. Freund. A transpose-free quasi-minimal residual method for non-hermitian linear systems. *SIAM Journal on Scientific Computing*, 14(2):470–482, 1993.

[30] G. H. Golub and C. Greif. On solving block-structured indefinite linear systems. *SIAM Journal on Scientific Computing*, 24(6):2076–2092, 2003.

[31] N. I. M. Gould, M. E. Hribar, and J. Nocedal. On the solution of equality constrained quadratic programming problems arising in optimization. *SIAM Journal on Scientific Computing*, 23(4):1376–1395, 2001.

[32] N. I. M. Gould, Y. Hu, and J. A. Scott. A numerical evaluation of sparse direct symmetric solvers for the solution of large sparse, symmetric linear systems of equations. Technical Report RAL-TR-2005-005, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2005.

[33] N. I. M. Gould. On practical conditions for the existence and uniqueness of solutions to the general equality quadratic-programming problem. *Mathematical Programming*, 32(1):90–99, 1985.

[34] R. Glowinski, T. Pan, and J. Périaux. A fictitious domain method for dirichlet problem and applications. *Computer Methods in Applied Mechanics and Engineering*, 111(3–4):283–303, 1994.

[35] M. H. Gutknecht. Variants of BICGSTAB for matrices with complex spectrum. *SIAM Journal on Scientific Computing*, 14(5):1020–1033, 1993.

[36] G. H. Golub and A. J. Wathen. An iterations for indefinite systems and its applications to the navier-stokes equations. *SIAM Journal on Scientific Computing*, 19(2):530–539, 1998.

[37] G. H. Golub, X. Wu, and J.-Y. Yuan. SOR-like methods for augmented systems. *BIT*, 41(1):71–85, 2001.

[38] J. Haslinger, T. Kozubek, R. Kučera, and G. Peichl. Projected schur complement method for solving non-symmetric systems arising from a smooth fictitious domain approach. *Numerical Linear Algebra with Applications*, 14:713–739, 2007.

[39] Harwell Subroutine Library. *A collection of Fortran codes for large-scale scientific computation*. AERE Harwell Laboratory, www.cse.clrc.ac.uk/nag/hsl, 2007.

[40] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. SIAM, Philadelphia, PA, 1995.

[41] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1):96–129, 1998.

[42] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1999.

[43] D. Orban. A library of projected krylov methods. Cahier du GERAD, GERAD, Montréal, Québec, Canada, 2008. In preparation.

[44] B. T. Polyak. The conjugate gradient method in extremal problems. *USSR Computational Mathematics and Mathematical Physics*, 9:94–112, 1969.

[45] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12:617–629, 1975.

[46] M. Rozložnik and V. Simoncini. Krylov subspace methods for saddle point problems with indefinite preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 24(2):368–391, 2002.

[47] Y. Saad. SPARSKIT: a basic tool kit for sparse matrix computations. Report RIACS-90-20, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffett Field, CA, USA, 1990.

[48] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, USA, 2nd edition, 2003.

[49] G. L. G. Sleijpen and D. R. Fokkema. BICGSTAB(l) for linear equations involving unsymmetric matrices with complex spectrum. *Electronics Transactions on Numerical Analysis*, 1:11–32, 1993.

[50] P. Sonneveld. CGS, a fast lanczos-type solver for nonsymmetric linear equations. *SIAM Journal on Scientific and Statistical Computing*, 10:36–52, 1989.

[51] Y. Saad and M. Schultz. GMRES, a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7:856–869, 1986.

[52] V. Sarin and A. Sameh. An efficient iterative method for the generalized stokes problem. *SIAM Journal on Scientific Computing*, 19(1):206–226, 1998.

[53] V. Simoncini and D. B. Szyld. Recent computational developments in krylov subspace methods for linear systems. *Numerical Linear Algebra with Applications*, 14(1):1–59, 2006.

[54] H. A. Van der Vorst. Bi-CGSTAB: A fast and smoothly converging variant to Bi-CG for the solution of nonsymmetric systems. *SIAM Journal on Scientific and Statistical Computing*, 13:631–644, 1992.