The Riccati algorithm for eigenvalues and invariant subspaces of matrices with inexpensive action

Jan Brandts*

Mathematical Institute, Utrecht University, P.O. Box 80.010, 3508 TA, Utrecht, The Netherlands

Received 9 May 2001; accepted 15 April 2002

Abstract

We present new algorithms for the numerical approximation of eigenvalues and invariant subspaces of matrices with cheap action (for example, large but sparse). The methods work with inexact solutions of generalized algebraic Riccati equations. The simpler ones are variants of Subspace Iteration and Block Rayleigh Quotient Iteration in which updates orthogonal to current approximations are computed. Subspace acceleration leads to more sophisticated algorithms. Starting with a Block Jacobi Davidson algorithm, we move towards an algorithm that incorporates Galerkin projection of the non-linear Riccati equation directly, extending ideas of Hu and Reichel in the context of Sylvester equations. Numerical experiments show that this leads to a very competitive algorithm, which we will call the Riccati method, after J.F. Riccati (1676–1754).

1. Motivation and overview

In numerical methods, and among them those to tackle linear algebra problems, the stability of the problem is a first prerequisite [6,15,17,42]. If the solution of a problem changes drastically with small perturbations of the data, then a satisfactory
approximation method can, in general, not be found. It is well-known that invariant subspaces of a matrix tend to be more stable than eigenvectors. Simple matrices can be given [41] of which some eigenvectors are highly unstable, whereas the invariant subspace they span is much better conditioned. Therefore, it is of great importance to develop algorithms that identify and approximate stable invariant subspaces. In this paper, we concentrate on the approximation of so-called Lipschitz-stable invariant subspaces, which are described and classified completely in [15]. Lipschitz stable invariant subspaces are precisely the ones, that are isolated in the sense of the upcoming analysis in Section 2.2. This analysis is mostly due to Stewart [41].

Well-known methods to approximate invariant subspaces are for example Subspace Iteration and Block Rayleigh Quotient Iteration, and block versions of the Arnoldi and Lanczos methods. See [16,27] for details. For inexact versions of Subspace Iteration and Inexact Block Rayleigh Quotient iteration, see [12,24,37].

1.1. Riccati algorithms

In this paper we will concentrate on methods that we will classify as Riccati algorithms. By this we mean numerical methods that aim to compute an approximation to the error of a current invariant subspace approximation, to update, and to proceed iteratively. This error, once mathematically well-defined, can be seen to satisfy a so-called generalized algebraic Riccati equation, which looks as follows,

\[ BP - PM = PG^*P - C, \]

and in which \( C, G \) and \( P \) are \((N - k) \times k\) matrices, \( B \) is \((N - k) \times (N - k)\), and \( M \) is \( k \times k \). Here, \( k \) is the dimension of the invariant subspace that we are interested in, and \( N \) is the dimension of the total space. The matrix \( P \) is the unknown in this non-linear equation. We refer to [3,21,25,45] for details on Riccati equations, and in particular to [43,44] for perturbation theory.

1.1.1. The central role of Sylvester equations

The linear operator in the left-hand side of (1) that maps \( P \) to \( BP - PM \) will play an important role in the analysis. It is usually called a Sylvester operator and accordingly, any equation of the type \( AX - XB = C \) is referred to as a Sylvester equation. As observed by Higham [17,18], the backward stability properties of Sylvester equations are quite different from those of a standard linear system of type \( Ax = b \), so special attention needs to be paid to their numerical solution. For a comparison of numerical methods, we refer to Simoncini [34]. Motivated by experiences in [4], in this paper we choose to concentrate on the Hu–Reichel algorithm [20]. This method is not based on applying standard methods to the Kronecker product formulation of the Sylvester equation, but on the construction of a Kronecker product of Krylov subspaces belonging to the two matrices \( B \) and \( M \), respectively. In fact, the Hu–Reichel algorithm can easily be extended in order to reduce the com-
plete Riccati equation (1) itself, instead of merely its linear (Sylvester) part. The research in this paper is mainly focussed on that issue, and leads to interesting theoretical and practical observations.

1.1.2. Favorable properties of Riccati algorithms

The small though subtle difference between Riccati algorithms and strategies that merely try to (iteratively) solve equations in which the unknown is the invariant subspace itself (as opposed to computing a correction to a current approximation), leads to algorithms that can, asymptotically, be assumed to be numerically more stable. For instance, the (Block) Rayleigh Quotient Iteration (B)RQI may suffer from its own convergence in the sense that the conditioning of the Sylvester equation that needs to be solved per iteration step, may increase drastically as the eigenvalues converge. As pointed out by Parlett [27] for the single-vector case, this does not necessarily hurt the convergence of the actual eigenvalue iteration. Indeed, for normal matrices, it can be seen that the error lies almost entirely in a direction that is not dangerous for the actual convergence. However, the proof of this fact depends on the property that spectral complements are orthogonal—which, in general, does not hold for non-normal matrices. Moreover, as mentioned above, Sylvester equations have different backward stability properties than linear systems and it is not clear how to use Parlett’s argument in the context of blocks. Also in case linear correction equations (either of Sylvester type or not) are solved only to low accuracy, which happens often in practical computations, things remain unclear. One of the aims in this paper is to show that, also in the block non-normal case, Riccati algorithms automatically protect themselves against ill-conditioning of the Sylvester equations that need to be solved.

1.1.3. Examples of Riccati algorithms

An example of a Riccati algorithm that typically suffers less from the kind of instability of a linear system matrix mentioned above, is the Jacobi–Davidson (JD) algorithm [35], which is often interpreted as a stabilized and subspace accelerated version of RQI. Note that it is a vector algorithm and not a block algorithm. One of the by-products of our analysis is the formulation of a suitable (and flexible) block version of JD. Stressing again the special properties of block equations, this algorithm is a non-trivial generalization of the single vector case. The recent reports [28,29] consider a JD-like block algorithm in which the dimension of the Galerkin space is kept at $3k$. The authors interpret their method as a "Riccati-based preconditioner". The “Davidson” component [8] of their algorithm (the subspace acceleration part) is therefore rather small compared to the “Jacobi” component (the idea of using orthogonal corrections, which was already introduced in [22]). Nevertheless it perfectly classifies as a Riccati algorithm. The Jacobi–Davidson QR and QZ algorithms [13], although in themselves rather successful, do not classify as a block JD algorithm, since they build partial Schur forms in a vector-by-vector manner.
In [33], inexact solutions of Riccati equations were merely used to refine eigenvalue approximations produced by an Arnoldi process, rather than to produce independent approximations themselves. See also [9] for work related to [33].

In this paper we will show that inexact solution methods for the Riccati equation lead to a whole class of invariant subspace methods. As a matter of fact, when subspace acceleration is considered, approximating the solution \( P \) of (1) by \( C \) leads to the Block Arnoldi Algorithm, as we will show in Section 4.1.1.

1.1.4. Numerical solution of Riccati equations

Numerical methods for algebraic Riccati equations have been studied mostly in the context of optimal control. Unfortunately, the generalized equation that plays such an important role here in this paper, seems to be much less studied. In particular, there seems to be a lack of methods for generalized algebraic Riccati equations in the special case for which

- \( B \) is very large, but a matrix–vector multiplication with \( B \) only costs order \((N - k)m\) arithmetic operations (where \( m \ll N \)).
- \( M \) is a small to medium sized but generally dense matrix.

As a matter of fact, many current numerical methods for Riccati equations are designed for dense matrices of moderate size that arise in optimal control problems and are based on the numerical solution of the corresponding eigenproblem, which was in fact our starting point. See Chapter 7 of [3] and its 237 references. In this paper we propose to go exactly in the opposite direction.

1.2. Newton–Grassmann methods

In recent years, it has been more and more recognized that the (differential) geometry that underlies the theory of invariant subspaces should not be neglected in the design of numerical methods [10], nor in the derivation of perturbation bounds [11]. A natural setting for eigenvectors and invariant subspaces is the one of Grassmann manifolds \( G^k(\mathbb{R}^N) \), which allows unique representations of the objects as single points on differentiable manifolds formed as quotient spaces of orthogonal groups, as exposed to the non-specialist in Chapters 4 and 5 of [3]. As pointed out for the Hermitian case in [10], the Newton method applied to the function

\[
F : G^k(\mathbb{R}^N) \rightarrow \mathbb{R}, \quad Y \mapsto \frac{1}{2} \text{trace}(Y^* A Y),
\]

which has the invariant subspaces as stationary points, can be elegantly formulated without constraints and degeneracies. In order to be able to apply not only Newton’s method but also conjugate gradients on \( G^k(\mathbb{R}^N) \), the authors consider geodesics, parallel transport, and the computation of gradients and Hessians. Interestingly enough, the Newton update in their setting is defined as the solution \( Q \) of a Sylvester equation on the tangent space of the Grassmann manifold,
where \( \Pi \) projects onto this tangent space and \( \Pi AY \) is the gradient. After choosing suitable coordinates, the method that remains is equivalent to a method we propose further on in this paper. In a recent report [12], the Newton–Grassmann approach is compared with BRQI, in both the exact and the inexact setting. It should be stressed that for non-normal matrices, the invariant subspace problem, even when formulated on the Grassmann manifold, does not allow an equally elegant treatment. Moreover, subspace acceleration, like we will consider in Section 4, is not an issue in [10,12].

1.3. Outline of this paper

The organization of this paper is as follows. For a complete understanding of the ideas involved, we choose to repeat some important results rather than to refer to them. This gives us the opportunity to clarify interrelations between methods and ideas that might not have been easy to show otherwise.

First, in Section 2, we will concentrate on stability of invariant subspaces while deriving the Riccati Correction Equation, following the lines of Stewart in [41]. New detail is to include an operator splitting in the analysis that allows us to examine the convergence behavior of methods based on that splitting. In Section 3 we give some simple algorithms, that can be interpreted as Riccati type versions of Subspace Iteration and the Block Rayleigh Quotient Iteration. Section 3.3 contains the central new ideas. We show how to incorporate tensored Krylov subspaces of Hu–Reichel type [20]. This leads, via iterations that take place within those subspaces, to non-linear Hu–Reichel type Krylov subspace methods to approximate solutions of the Riccati Correction Equation. In Section 4, those approximate solutions will be used in the context of subspace acceleration, which means that Ritz–Galarkin test- and trialspaces will be formed from them. We show that also the Arnoldi and Jacobi–Davidson methods can be put in this framework, and present our new algorithms, of which the most promising can rightfully be called the Riccati method. We conclude in Section 5 with numerical experiments that show the success of the new methods compared to Jacobi–Davidson.

2. Invariant subspaces and algebraic Riccati equations

We will now recall some basic facts about invariant subspaces adopting the notations and results from [42], while putting the emphasis on algorithmical aspects.

Remark 2.1. Throughout the paper we will identify the columnspan of a matrix with the matrix itself, i.e. we talk about the matrix \( X \) as well as the subspace \( X \).
2.1. Derivation of the Riccati Correction Equation

Let $\hat{X}$ represent an invariant subspace of $A$, and let $\hat{Y}$ be such that $(\hat{X}|\hat{Y})$ is an $N \times N$ unitary matrix. Then, transformation of $A$ to the basis given by the columns of $(\hat{X}|\hat{Y})$ gives

$$A(\hat{X}|\hat{Y}) = (\hat{X}|\hat{Y}) \begin{bmatrix} \hat{M} & \hat{G}^* \\ 0 & \hat{B} \end{bmatrix}$$

for certain $\hat{M}$, $\hat{G}$ and $\hat{B}$. Now, let $X$ be a unitary matrix that is supposed to be an approximation of $\hat{X}$ and let $Y$ be such that $(X|Y)$ is unitary. Then, transformation of $A$ to the basis given by the columns of $(X|Y)$ results in the definition of the blocks $M, B, C$ and $G^*$ for which

$$A(X|Y) = (X|Y) \begin{bmatrix} M & G^* \\ C & B \end{bmatrix}.$$  

(5)

Note that by comparing columns $AX = XM + YC$. So, $C = Y^*R$, where $R := AX - XM$ is the residual for the approximation $X$. Since $X^*R = 0$, we moreover have that $\|C\| = \|R\|$. If $A$ is Hermitian, $\hat{G} = 0$ and $\hat{Y}$ is an invariant subspace as well. Also, $C = G^*$, and $Y$ is as good an approximation to $\hat{Y}$ as $X$ is to $\hat{X}$.

2.1.1. Unitary transformations mapping $(X|Y)$ to $(\hat{X}|\hat{Y})$

The matrix $(\hat{X}|\hat{Y})$ can be obtained from $X$ and $Y$ as follows. First assume that $H := X^*\hat{X}$ and $K := Y^*\hat{Y}$ are invertible (this assumption will later be reformulated as “the maximum angle between $\hat{X}$ and $X$ is not straight”). Then write

$$(\hat{X}|\hat{Y}) = (X|Y) \begin{bmatrix} X^*\hat{X} & X^*\hat{Y} \\ Y^*\hat{X} & Y^*\hat{Y} \end{bmatrix} =: (X|Y) \begin{bmatrix} H & QK \\ PH & K \end{bmatrix}.$$  

(6)

in which, clearly, $P = Y^*\hat{X}H^{-1}$ and $Q = X^*\hat{Y}K^{-1}$. As a product of two unitary matrices, the rightmost matrix in (6) is unitary as well, which leads to the relations

$$H^*(I + P^*P)H = I \quad \text{and} \quad K^*(I + Q^*Q)K = I,$$

(7)

$$H^*(P^* + Q)K = 0.$$  

(8)

Since we assumed $H$ and $K$ to be invertible, we can conclude from (8) that $Q = -P^*$, after which it appears from (7) that $H$ and $K$ are of the form $H = (I + P^*P)^{-1/2}U$ and $K = (I + P^*P)^{-1/2}V$ with $U$ and $V$ unitary matrices of the correct size. Summarizing, this gives

$$(\hat{X}|\hat{Y}) = (X|Y) \begin{bmatrix} I & -P^* \\ P^* & I \end{bmatrix} \begin{bmatrix} I + P^*P & 0 \\ 0 & I + PP^* \end{bmatrix}^{-1/2} \begin{bmatrix} U^* & 0 \\ 0 & V \end{bmatrix}.$$  

(9)

An important characterization of the matrix $P$ can be given by realizing that $(\hat{X}|\hat{Y})$ should accomplish the block Schur form (4). Substituting the result of (9) into
$\hat{Y^*A\hat{X}} = 0$ we find, in terms of the blocks $M$, $B$, $C$ and $G$ in (5), that this is (independent of $U$ and $V$) equivalent to the condition that $P$ satisfies the following generalized algebraic Riccati equation,

$$BP - PM = PG^*P - C.$$  \hfill (10)

This can, equivalently, be interpreted as that the block $P$ satisfies the following invariant subspace equation,

$$(X|Y)^*A(X|Y) \begin{bmatrix} I \\ P \end{bmatrix} = \begin{bmatrix} M & G^* \\ C & B \end{bmatrix} \begin{bmatrix} I \\ P \end{bmatrix} = \begin{bmatrix} I \\ P \end{bmatrix} (M + G^*P).$$ \hfill (11)

So given a matrix $P$ that solves (10) or (11), an invariant subspace $\hat{X}$ and its orthogonal complement $\hat{Y}$ can be computed through Eq. (9). Note that $X + YP$ already has the same columnspan as $\hat{X}$ and that the factor $(I + P^*P)^{-1/2}$ is an orthonormalization factor only. The presence of the arbitrary unitary transformations $U$ and $V$ reflects the non-uniqueness of orthonormal bases for a given subspace and can therefore be chosen as the identity for simplicity.

2.1.2. Orthogonal correction of $X$ into an invariant subspace

As we have seen above, in order to find the invariant subspace $\hat{X}$ from an initial approximation $X$ of $\hat{X}$, we need to compute the corresponding product $Q := YP$ and add it to $X$. Since each column of $YP$ is orthogonal to all columns of $X$, it makes sense to talk about finding the orthogonal correction to $X$. In (11), the same is expressed on a different basis: under the assumptions that $H$ and $K$ are invertible, the matrix $A$ transformed to the basis given by $(X|Y)$ has an invariant subspace that can be written as an orthogonal correction $P$ to the canonical embedding of $\mathbb{R}^k$ into $\mathbb{R}^N$. Also, the whole derivation leading to (9) shows that any next invariant subspace approximation that is constructed using approximations of $P$, is always obtained through (a sequence of) stable unitary similarity transformations, each of which being performed as the product of an orthogonal correction and an orthonormalization.

In case $v = X$ is a single vector that approximates an eigenvector $\hat{v} = \hat{X}$ of $A$, the orthogonal correction is illustrated in Fig. 1. As a non-linear equation, Eq. (10) may have several solutions. For example, in the case illustrated in Fig. 1, each eigenvector span that intersects the affine variety $v^\perp := \{w + v \mid w^*v = 0\}$, will give rise to a corresponding orthogonal correction. Note that this “intersection criterion” is the translation of the condition that $H := X^*\hat{X}$ should be invertible.

For obvious reasons, we will from now on refer to (10) as the Riccati Correction Equation.

2.1.3. Computational aspects

The fact that we are interested in the correction $Q$ rather than in $P$ and $Y$ separately, has an important computational consequence. Instead of working with the matrix $Y$,
which could be quite large and therefore computationally expensive to form, we use
\[ B = Y^*AY, \ C = Y^*R, \ G^* = X^*AY \] and \( Y^*Y = I \) to transform (10) back to the original basis of \( \mathbb{R}^N \) as follows,

\[
BP - PM = PG^*P - C
\Rightarrow Y^*A(YP) - Y^*(YP)M = Y^*(YP)X^*A(YP) - Y^*R
\Rightarrow X^*Q = 0 \quad \text{and} \quad Y^*(AQ - QM) = Y^*(QX^*AQ - R). \quad (12)
\]

The orthogonality relation \( Y^*(AQ - QM - QX^*AQ + R) = 0 \) is basis-independent, so we can get rid of \( Y \) and replace it by \( II := I - XX^* \), since \( Y^*z = 0 \iff IIz = 0 \). Moreover, \( X^*R = X^*Q = 0 \). This transforms (12) into the equivalent equation

\[
X^*Q = 0 \quad \text{and} \quad IIAQ - QM = Q(X^*AQ - R). \quad (13)
\]

This equation only involves the given matrices \( A \) and \( X \), and the matrices \( M \) and \( R \), which, compared to \( B \) and \( Y \), are relatively cheap to compute.

**Observation 2.2.** It is interesting to see that this approach is in fact equivalent to a Ritz–Galerkin projection of the equation \( AQ - QM = QX^*AQ - R \) onto the orthogonal complement of \( X \). Indeed, we look for a block \( Q \) of the form \( YP \) such that the residual \( AQ - QM - QX^*AQ + R \) is orthogonal to the trial space. Since the exact solution is included in the trial space, this is a Ritz–Galerkin projection without loss of accuracy. Reason to perform it nevertheless, is that the projected equation is better conditioned.

We will see further on that the matrix \( II \) does not need to be formed when suitable iterative methods are used to solve (13). In theoretical parts of this paper, we will pre-
fer to work with (10). Not only because of its notational simplicity, but also because of the fact that avoiding $Y$ has led to a singular operator together with an orthogonality constraint to circumvent this singularity as in (13), which, though correct, seems somewhat cumbersome.

Observation 2.3. Eq. (13) can be interpreted as the non-linear version of the correction equation for linear systems of equations: if $X$ approximates the solution of $AX = B$ then, defining the residual $R$ in this setting as $R := AX - B$, the probably non-orthogonal, though unique correction $Q$ to $X$ that is needed to produce $\hat{X}$ from $X$ satisfies $AQ = -R$. Clearly, for linear systems, the correction equation is again a linear system. For invariant subspace problems, the correction equation is also an invariant subspace problem (11), though in (10) and (13) it is formulated as an equivalent generalized algebraic Riccati equation.

2.2. Convergence of approximations to invariant subspaces

In order to be able to talk about convergence of a sequence of approximations to an invariant subspace, we will use the following measure for the distance between two subspaces of the same dimension.

Definition 2.4. Let $U_1$ and $V_1$ be $N \times k$ matrices and $U_2$ and $V_2$ such that $(U_1|U_2)$ and $(V_1|V_2)$ are unitary. Define the gap $\theta(\cdot, \cdot)$ between $U_1$ and $V_1$ as

$$\theta(U_1, V_1) := \|U_1^*V_2\| = \|P_{U_1} - P_{V_1}\|.$$  \hspace{1cm} (14)

Here, $P_U$ and $P_V$ are the orthogonal projections on $U$ and $V$, respectively.

It is well-known [15] that $\theta(U_1, V_1)$ can be interpreted as the sine of the maximum angle between $U_1$ and $V_1$, and therefore, the condition from the previous section that $H := X^*\hat{X}$ should be invertible, is equivalent to the rather weak condition that this maximum angle between $X$ and $\hat{X}$ should not be straight.

2.2.1. Pin-pointing the invariant subspace closest to $X$

Each $k$-dimensional invariant subspace $\hat{X}$ of $A$ for which the maximum angle between $\hat{X}$ and the current approximation $X$ is not straight, corresponds to a solution $P$ of the Riccati Correction Equation (10). Using (9), we can derive that the gap between $X$ and $\hat{X}$ satisfies

$$\theta(\hat{X}, X) := \|\hat{X}^*Y\| = \|P(I + P^*P)^{-1/2}\| \leq \|P\|.$$  \hspace{1cm} (15)

For $k = 1$, the invariant subspace $\hat{X}$ for which $\theta(\hat{X}, X)$ is minimal, corresponds to the solution $P$ of (10) for which the norm is $\|P\|$ is minimal. Also in view of (11), we will therefore from now on concentrate on such minimal norm solution(s).
Definition 2.5. Define, on the space of \((N - k) \times k\) matrices, the linear Sylvester operator \(T\) associated with \(B\) and \(M\), and consequently the separation between the matrices \(B\) and \(M\) by

\[\text{T}: Q \mapsto BQ - QM, \quad \text{sep}(B, M) := \inf_{\|Q\|=1} \|T(Q)\|.\] (16)

Let \(T = R - S\) be a splitting of the operator \(T\). The remainder of this section reduces for \(S = 0\) to results by Stewart [41]. The reason to slightly complicate the analysis by introducing the splitting is to make some bounds easier to compute, and to be able to derive convergence bounds for fixed-point iterations for (10) based on this splitting. This will be the topic of Section 3.1.

Suppose that \(\|C\| \neq 0\) and that \(P\) satisfies (10). Then \(\|P\| \neq 0\) and

\[\inf_{\|Q\|=1} \|R(Q)\| \leq \frac{\|R(P)\|}{\|P\|} \leq \sup_{\|Q\|=1} \|S(Q)\| + \|G\| + \|C\| \|P\|.\] (17)

Writing \(\gamma := \|C\|, \chi := \|G\|, \sigma := \|S\|, \rho = \inf_{\|Q\|=1} \|R(Q)\|\) and \(\delta := \rho - \sigma\) we conclude from multiplying (17) with \(\|P\|\) that the norm \(p := \|P\|\) of any solution of (10) necessarily satisfies

\[\delta p \leq \chi p^2 + \gamma.\] (18)

If \(\delta^2 > 4\chi\gamma\) (Fig. 2(a)), then (18) holds for all \(p \geq 0\), which means that nothing can be concluded for \(p\) from this analysis. If, however, \(\delta^2 \leq 4\chi\gamma\) and \(\delta > 0\) (Fig. 2(b)), then both roots

---

![Fig. 2. Capturing an invariant subspace. (a) \(\delta^2 - 4\chi\gamma < 0\), (b) \(\delta^2 - 4\chi\gamma > 0\).](image-url)
\[ \tau_\ell := \frac{\delta}{2\chi} \left( 1 - \sqrt{1 - \frac{4\chi\gamma}{\delta^2}} \right) \quad \text{and} \quad \tau_r := \frac{\delta}{2\chi} \left( 1 + \sqrt{1 - \frac{4\chi\gamma}{\delta^2}} \right) \]  

(19)

are positive and real, and (18) holds everywhere except on the open interval \( \tau := (\tau_\ell, \tau_r) \). This means in particular that there exists no solution \( P \) of (10) such that \( \|P\| \in \tau \). There does, indeed, exist a unique solution \( P \) of (10) within the ball \( B(\tau_\ell) := \{ Q \mid \|Q\| \leq \tau_\ell \} \), which we will from now on denote by \( P^* \). This will be shown in Lemma 2.6.

**Lemma 2.6.** Suppose \( \delta^2 - 4\chi \gamma > 0 \) and \( \delta > 0 \). Then there exists exactly one solution \( P^* \) of (10) in \( B(\tau_\ell) \). This solution gives rise to an invariant subspace \( \hat{X} \) of \( A \) such that

\[ \theta(\hat{X}, X) \leq \|P^*\| \leq \tau_\ell \leq \frac{2\gamma}{\delta} \leq \frac{\delta}{2\chi} \]  

(20)

Moreover, there exists no solution \( P \) of (10) such that \( \tau_\ell < \|P\| < \tau_r \).

**Proof.** Note that \( \delta > 0 \) implies \( \rho > 0 \) so \( R \) is invertible and \( \|R^{-1}\|^{-1} = \rho \). Consider the mapping \( W \) defined by

\[ W(Q) = R^{-1}(S(Q) + QG^*Q - C). \]  

(21)

One can verify that \( W \) maps \( B(\tau_\ell) \) into itself. Now, let \( U, V \in B(\tau_\ell) \), then an easy algebraic manipulation yields the following equality,

\[ W(U) - W(V) = R^{-1} \left( S(U - V) + UG^*(U - V) + (U - V)G^*V \right). \]  

(22)

After taking norms and using that \( \chi(\tau_\ell + \tau_r) = \rho - \sigma \), we obtain

\[ \|W(U) - W(V)\| \leq \frac{\sigma}{\chi} + \frac{\tau_\ell + \tau_r}{\sigma/\chi + \tau_\ell + \tau_r} \|U - V\|, \]  

(23)

so \( W \) is a contraction on \( B(\tau_\ell) \), establishing the existence and uniqueness of a fixed point \( P^* \) of \( W \) in \( B(\tau_\ell) \). Clearly, \( P^* \) solves (10). Combining (15) and the discriminant condition \( \delta^2 - 4\chi \gamma > 0 \) leads to (20). \( \Box \)

**2.2.2. Discussion**

The operator splitting introduces a parameter \( \sigma \) in the analysis that gives optimal results in Lemma 2.6 for \( \sigma = 0 \). First, of course, the conditions \( \delta^2 - 4\chi \gamma > 0 \) and \( \delta > 0 \) are the easiest to satisfy when no splitting is used at all. Apart from that, Fig. 2 clearly shows that \( \tau_\ell \) moves to the left and \( \tau_r \) to the right for increasing slope \( \delta \) of the straight line. So \( P^* \) can a priori be known to be better isolated for larger values of \( \delta \). And finally, the contraction number in (23) is smallest for \( \sigma = 0 \). Therefore, the
question may arise why to include the analysis for non-trivial splittings. The main reasons are of practical nature,

- it often appears much harder to compute (or to estimate) the smallest singular value $\delta$ of $T$ than $\rho$ and $\sigma$ for some splitting,
- in numerical algorithms, (see Section 3) solving Sylvester systems $T(U) = V$ may be hard, and splittings could be used that still lead to convergence.

In such cases, Lemma 2.6 helps to formulate convergence conditions for iterations based on such a splitting as well as a contraction number corresponding to such iterations. We will study those in Section 3.

3. Approximation of the Riccati Correction Equation

Suppose that the conditions of Lemma 2.6 are satisfied. This implies that a unique minimum-norm solution $P^*$ of the Riccati Correction Equation is guaranteed inside a non-empty ball $B(\tau)$. In order to compute $P^*$, one may use standard iteration methods based on the contractivity within this ball, like simple Picard iterations or the Newton method. We will outline those in Section 3.1.

In practice however, the most difficult part is to get into the situation that the conditions of Lemma 2.6 are satisfied. Numerical experiments strongly suggest that this is actually of vital importance for convergence. To overcome this problem, in Section 4 we will build a Ritz–Galerkin space of approximate solutions of the Riccati Correction Equation in the hope that this brings us close enough to an invariant subspace for the conditions to hold. So, apart from solving (10) to high accuracy once $\delta^2 - 4\gamma \chi > 0$, we are also interested in heuristics on what to do if this is not yet the case.

3.1. Basic Picard iterations and Newton’s method

The Picard iterations that are easiest to study are the ones that treat the quadratic term $PG^*P$ in (10) explicitly,

given $P_0 = 0$, iterate $R(P_{n+1}) = S(P_n) + P_nG^*P_n - C$. (24)

The effect of the splitting on the convergence of the Picard iteration is given in the following theorem. Note that also its conditions depend on the splitting.

**Theorem 3.1.** Assume that $\delta^2 - 4\gamma \chi > 0$ and $\delta > 0$. Then the explicit Picard iteration (24) converges to $P^*$. Moreover,

$$\frac{\|P^* - P_n\|}{\|P^*\|} \leq \left(\frac{\sigma/\chi + \tau e + \tau r}{\sigma/\chi + \tau e + \tau r}\right)^n = \left(1 - \frac{\tau r - \tau e}{\sigma/\chi + \tau e + \tau r}\right)^n. \quad (25)$$

**Proof.** Using $P^* - P_n = W(P^*) - W(P_{n-1})$ and the contraction property (23), we obtain the result by induction and the fact that $\|P^* - P_0\| = \|P^*\|$. □
Unfortunately, these Picard iterations are only linearly convergent, which may seem less interesting than for instance Newton-like methods. Defining a function \( f \) by \( f(P) := BP - PM - PG^*P + C \), we note that

\[
Df(P)(H) = f(P + H) - f(P) + HG^*H = (B - PG^*)H - H(M + G^*P),
\]

(26)

so that the following scheme is the classical Newton method applied to (10).

**Algorithm 3.1.** Newton method for the Riccati Correction Equation

```plaintext
input: B, M, G, C, ε
n = 0, P_0 = 0, S_0 = C
while \|S_n\| > ε \|S_0\|

  Df(P_n)(Q) = S_n (solving for Q)
  P_{n+1} = P_n - Q
  S_{n+1} = f(P_n)
  n = n + 1
end (while)
```

The Newton method converges quadratically when close enough to the exact solution [9]. The advantage of the simple Picard iteration however, is that the linear operator with which equations are to be solved, is the same for all iterations. We will try to exploit this further on. First, we will describe how to accelerate the Picard iterations.

### 3.2. A note on accelerated Picard iterations

Since the matrix coefficients \( M, B, C \) and \( G \) of the Riccati Correction Equation (10) determine the convergence properties of the Picard iterations, it may be well worthwhile to replace them once better coefficients are available. In view of (9), we can, for any \( P_n \), define new though intermediate approximations of the invariant subspace by

\[
(X_n|Y_n) = (X|Y) \left[ \begin{array}{cc} I & -P_n^* \\ P_n & I \end{array} \right] \left[ \begin{array}{cc} I + P_n^*P_n & 0 \\ 0 & I + P_nP_n^* \end{array} \right]^{-1/2}
\]

(27)

so that with \( P_0 = 0 \) we have \( X_0 := X \) and \( Y_0 := Y \), and defining

\( B_n := Y_n^*AY_n, \quad M_n := X_n^*AX_n, \quad C_n := Y_n^*AX_n \) and \( G_n := X_n^*AY_n \),

we obtain an updated Riccati Correction Equation,

\[
B_nP - PM_n = PG_n^*P - C_n,
\]

(28)

for which the correspondingly defined quantity \( \delta_n^2 = 4\gamma_nX_n \) might be larger than before, leading to faster convergence. As an example of how to implement this, see
Algorithm 3.2. By $T_n$ we denote the Sylvester operator belonging to $B_n$ and $M_n$, and $W_n$ denotes a successive substitution process corresponding to a splitting of $T_n$. The inner iteration is, in this example, limited to $\ell$ steps, although it can also be terminated by a sufficient reduction of the residual $S_k$ (see below) for the Riccati Correction Equation.

Algorithm 3.2. Accelerated Picard iteration

**input:** $A$, $X_0$, $\varepsilon_1$, $\varepsilon_2$, $L$

$n = 0$

choose $Y_0$ and compute $B_0$, $M_0$, $C_0$, $G_0$

while $\|C_n\| > \varepsilon_1 \|C_0\|$

$k = 0$

$P_0 = 0$

$S_0 = T_n(P_0) - P_0 G_n^* P_0 + C_n$

while $\|S_k\| > \varepsilon_2 \|S_0\|$ and $k \leq \ell$

$k = k + 1$

$S_k = T_n(P_k) - P_k G_n^* P_k + C_n$

end (while)

$X_{n+1} = \text{qr}(X_n + Y_n P_k)$ (orthogonal factor of qr-factorization)

choose $Y_{n+1}$ and compute $B_{n+1}$, $M_{n+1}$, $C_{n+1}$, $G_{n+1}$

$n = n + 1$

end (while)

Accelerating the Picard iterations in this manner leads to iterations schemes that are quite similar to the Newton Method. The main difference is that in the Newton Method, the coefficient matrices of the Sylvester equation change in every iteration step, whereas in Algorithm 3.2, a number of steps (bounded by $\ell$) with the same operator is done. We will now show how this might be exploited.

3.3. Tensor product Krylov subspace methods

As mentioned in Section 1, solving Sylvester equations is of central importance in the approximation methods of this paper. Here we will start with studying the inner loop of Algorithm 3.2, in which $k \leq \ell$ Picard iterations are performed. Without splitting, and suppressing the indices of the matrices $B$, $M$, $G$ and $C$, this boils down to

Given $P_0 = 0$, iterate $B P_{k+1} - P_{k+1} M = P_k G^* P_k - C$. \hspace{1cm} (29)

This means that $k \leq \ell$ Sylvester equations, all with the same operator but with different right-hand sides, need to be solved. However, if the sequence $P_k$ converges, the right-hand sides will not differ much. In that case, a Krylov subspace that is used to approximate $P_1$ from $P_0$, may be well suitable to approximate $P_{k+1}$ from $P_k$. 

also for larger \( k \). We will illustrate this in Section 3.3.2. First however, we present a brief review of the Hu–Reichel algorithm [20] for building a tensor product Krylov subspace that is particularly suitable for Sylvester equations in Section 3.3.1. Then, in Section 3.3.3, we give an interpretation of this approach in terms of Ritz–Galerkin projection of the Riccati Correction Equation. From now on, we will denote the block Krylov subspace span\{X, AX, \ldots, A^{m-1}X\} by \( K^m(A, X) \).

3.3.1. The Hu–Reichel algorithm for Sylvester equations

Generally, assume that \( U \) is an orthogonal \( n \times p \) matrix and \( V \) an orthogonal \( k \times q \) matrix. Then, a Ritz–Galerkin approximation \( UZV^* \) of the solution \( P_1 \) of the first iteration step \( BP_1 - P_1 M = -C \) of (29), can be obtained by identifying the \( p \times q \) matrix \( Z \) such that

\[
U^*(BUZV^* - UZV^* M + C)V = 0.
\]

(30)

Note that if \( U, Z \) and \( V \) are full rank matrices, the column span of \( UZV^* \) is equal to the column space of \( U \), whereas its row span equals that of \( V^* \). So, \( UZV^* \) represents a tensor product of the subspaces \( U \) and \( V \).

Hu and Reichel [20] propose to choose for \( U \) and \( V \) orthonormal bases for the block Krylov subspaces associated to \( B \) and to \( M^* \), respectively, with respective starting blocks full rank matrices \( CB \) and \( CM \) such that \( C = CBC^* \). Then, (30) can be written as

\[
H_B Z_1 - Z_1 M = -(U^*CB)(V^*CM)^*,
\]

(31)

where \( H_B := U^*BU \) is \( p \times p \) upper Hessenberg, \( H_M := V^*M^*V \) is \( q \times q \) upper Hessenberg, and both \( U^*CB \) and \( V^*CM \) tall upper triangular matrices. As observed by Simoncini [34], this Galerkin method is equivalent to a truncation of a series representation of the exact solution in terms of block Krylov matrices and minimal polynomials. Hu and Reichel also present a minimal residual method based on the same idea.

Remark 3.2. Since \( k \) is assumed small, we will, alternatively, choose \( V = I \), the \( k \times k \) identity matrix. This means that the action of \( M \) is used exactly. We will write the resulting projected equation as

\[
H_B Z_1 - Z_1 M = -U^*C,
\]

(32)

where \( Z_1 \) is \( p \times k \) with \( p = mk \) for some \( m \). Any efficient direct method can be used to solve the small projected system.

Looking ahead to the next section, we suggest to compute Schur decompositions of \( H_B \) and \( M \) in (32) and to apply the Bartels–Stewart algorithm [2] to the resulting transformed Sylvester equation with two upper triangular matrices.
3.3.2. Further iteration in the same Krylov subspace

Suppose that the Hu–Reichel algorithm has provided us with an approximation \( \hat{P}_1 = UZ_1 \) of \( P_1 \). Iterating further on (29) means that an approximation \( \tilde{P}_2 \) of \( P_2 \) should be found from

\[
B \tilde{P}_2 - \tilde{P}_2T = \hat{P}_1 G^* \hat{P}_1 - C. \tag{33}
\]

This can, in theory, be done in the same Krylov subspace that was already built in the first iteration step, simply by replacing \(-C\) in (32) by \( \hat{P}_1 G^* \hat{P}_1 - C \), leading to

\[
H_B Z_2 - Z_2 M = U^*(\hat{P}_1 G^* \hat{P}_1 - C) = Z_1 (G^*U)Z_1 - U^*C. \tag{34}
\]

This Sylvester equation can be solved very cheaply. Not only because it is small, but in particular since left-overs from the solution process for (32) can be re-used, like the Schur decompositions of \( H_B \) and \( M \).

Clearly, \( Z_2 \) gives rise to an approximation \( \hat{P}_2 \) of \( \tilde{P}_2 \), and the process can be repeated. The sequence \( \hat{P}_k = UZ_k \) that is defined this way, approximates the original iterates \( P_k \) and lies completely in the \( mk \)-dimensional block Krylov subspace for \( B \) with starting block \( C \) that was constructed to approximate the solution of the first iteration step of (29). We will now show that there is another interpretation to this procedure.

3.3.3. The Hu–Reichel algorithm for the Riccati Correction Equation

We can apply the Ritz–Galerkin idea not just to each of the Sylvester equations, but a matter of fact, also to the Riccati Correction Equation (10) itself. We look for an approximate solution of the form \( UZV^* \) and demand that the residual is orthogonal to the tensor product of \( U \) and \( V \), interpreted as subspaces, by

\[
U^*(BUZV^* - UZV^*M - UZV^*G^*UV^* + C)V = 0, \tag{35}
\]

which, after some cleaning up, leads to

\[
(U^*BU)Z - Z(V^*MV) = Z(V^*G^*U)Z - U^*CV. \tag{36}
\]

This is a small generalized Riccati equation for \( Z \), and one may check that using Picard iterations to approximate its smallest norm solution, starting with \( Z_0 = 0 \), gives back the sequence \( Z_k \) from the previous section. On the other hand, comparing (36) with (10) and (11), we know that it is equivalent to the invariant subspace problem,

\[
\begin{bmatrix}
V^*MV & V^*G^*U \\
U^*CV & U^*BU
\end{bmatrix}
\begin{bmatrix}
I \\
Z
\end{bmatrix} = \begin{bmatrix}
I \\
Z
\end{bmatrix} (V^*MV + V^*G^*UZ), \tag{37}
\]

and we can write the matrix in the left-hand side of (37) as

\[
\begin{bmatrix}
V^*MV & V^*G^*U \\
U^*CV & U^*BU
\end{bmatrix} = \begin{bmatrix}
V & 0 \\
0 & U^*
\end{bmatrix} \begin{bmatrix}
X|Y)^*A(X|Y) & V \\
0 & U
\end{bmatrix}. \tag{38}
\]
This shows that a general tensor product Ritz–Galerkin method applied to the Riccati Correction Equation, directly corresponds to a special type of Ritz–Galerkin projection of the original matrix \( A \) on the space defined by the matrix \( (XV | YU) \).

**Observation 3.3.** If \( U \) contains an orthonormal basis for the block Krylov subspace for \( B \) with start block \( C \), and if \( V = I \), the matrix \( A \) projected on \( (X|YU) \) is block upper Hessenberg. This follows directly from (38). The block version of the Implicit Q-theorem [16] then shows, that \( (X|YU) \) contains, in fact, an orthonormal basis for the block Krylov subspace for \( A \) with start block \( X \).

**Theorem 3.4.** The Hu–Reichel method applied to the Riccati Correction Equation results in the Block Arnoldi method for the matrix \( A \) with start block \( X \).

This equivalence is valid in exact arithmetic, and it shows that also the (Block) Arnoldi method [1,32] can be interpreted as a Riccati-type algorithm that works with orthogonal corrections to a given approximation \( X \) of an invariant subspace. In numerical practice however, there is a subtle difference: an implementation based on the Riccati Correction Equation in the original coordinates (13) forces an additional re-orthogonalization within this Block Arnoldi method. This is expressed in the following Lemma, formulated generally, that connects the implementation through (13) with the standard implementation of the Block Arnoldi method.

**Lemma 3.5.** Define for a given \( N \times k \) orthogonal matrix \( X \) the blocks

\[
M = X^*AX, \quad R = AX - XM \quad \text{and} \quad S^* = X^*A - MX^*.
\]

Let \( Q_1R_1 := R \) denote a QR-decomposition of \( R \) and write \( Q \) for the orthonormal basis of \( K^m(IIAII, Q_1) \) generated by \( m \) steps of the Block Arnoldi algorithm. Set \( H = Q^*AQ \). Then \( (X|Q) \) is the Block Arnoldi basis for \( K^{m+1}(A, X) \), and

\[
H_+ := (X|Q)^*A(X|Q) = \begin{pmatrix} M & S^*Q \\ Q^*R & H \end{pmatrix}.
\] (39)

**Remark 3.6.** The equivalence should be properly taken in the light of the Implicit Q-theorem [16] and modulo non-uniqueness of \( QR \)-factorizations. In the proof below we will silently assume this and refrain from expressing it explicitly.

**Proof of Lemma 3.5.** Recall the construction of the Block Arnoldi basis for \( K^{m+1} \) \((A, X)\). Starting with \( X \), the next basis block is obtained by applying \( A \) to \( X \), orthogonalization of the result to \( X \), and picking out its orthogonal \( QR \)-factor. Since \((I - XX^*)AX = R\), this orthogonal \( QR \)-factor equals \( Q_1 \). To compute the next block-column, \( AQ_1 \) needs to be orthogonalized against \( X \) and \( Q_1 \). But since \( IIQ_1 = Q_1 \), this is the same as orthogonalizing \( IIQ_1 \) to \( X \) and \( Q_1 \). The orthogonalization against \( X \) results in \( IIQ_1 \) which then needs to be orthogonalized against \( Q_1 \).
Proceeding inductively, we conclude the equivalence, and compute the projected matrix \( A \) on \( K_1^{m+1}(A, X) \) as

\[
H_+ = \begin{pmatrix}
X^*AX & X^*AQ \\
Q^*AX & Q^*AQ
\end{pmatrix} = \begin{pmatrix}
X^*AX & (X^*A - MX^*)Q \\
Q^*(AX - XM) & Q^*AQ
\end{pmatrix}.
\] (40)

This completes the proof by definition of \( R, S, M, H \). \( \square \)

In the presence of rounding errors, which may be substantial if \( X \) is close to an invariant subspace, the computed residual might be far from orthogonal to \( X \). If so, then for the computed blocks \( Q_j \), denoting them by \( \tilde{Q}_j \), we have \( A\tilde{Q}_j \neq A\Pi\tilde{Q}_j \). In this case, the re-orthogonalization forced by working with \( A\Pi\Pi \) may result in a more stable method.

### 3.3.4. Concluding remarks

We will end this section with some concluding remarks that aim to rephrase and expose the cross-links between numerical methods that were derived in different settings, but which may have turned out to be more similar than expected on beforehand.

- At each point during a Block Arnoldi method, the method may of course be fully restarted, using as restart the best approximating Ritz block. This may, compared to the Implicitly Restarted Arnoldi method, not be a very sophisticated way to restart, but seen in the light of Section 3.2, when close enough to an exact invariant subspace, this restart, together with explicit re-orthogonalization, might substantially improve the convergence.
- Simoncini and Sadkane suggest in their paper [33] to refine an approximation \( X \) of an invariant subspace obtained by the Arnoldi method, by writing down a Riccati Correction Equation for \( X \) and doing one or more Picard iterations. From the analysis above, it shows that instead, the Arnoldi method might have been restarted with \( X \) and using explicit re-orthogonalization to \( X \) in the construction of the Krylov Subspace from that point on.

### 4. Riccati algorithms with subspace acceleration

All previous algorithms are Ritz–Galerkin methods in which each Ritz–Galerkin projection is done on a space having the same dimension as the object (eigenvector, invariant subspace) that we wish to approximate. The next subspace to project upon simply replaces the previous, and is either generated by some number of Picard iteration steps like in Algorithm 3.2 or, as proposed in Section 3.3.3, by a Hu–Reichel Galerkin approach that could be interpreted as a Block Arnoldi method with re-orthogonalization. Unfortunately, satisfying the conditions under which the Picard iterations would converge is not trivial, nor are they easy to check. As a matter of
fact, numerical experiments strongly suggest that if the conditions are not satisfied, there is no convergence.

4.1. Ritz–Galerkin methods with an expanding subspace

In order to overcome these problems, which clearly occur if the initial guess is too far away from the solution, it will be important to include the possibility to expand the Galerkin subspace instead of merely replacing it. Here, we will expand the subspace by approximate solutions of Riccati Correction Equations. Expanding the space in which approximations for the object are sought, is a systematic and straightforward way to find one such, that the conditions for convergence of the Picard iterations and their accelerations are satisfied. From that point on, no further subspace expansion would be needed, although obviously in practice one prefers to do so. In Algorithm 4.1 below we sketch a general frame, in which the dimension of the subspace (and hence of the projected problem) grows in each step by $k$.

If the dimension of the subspace becomes too large for efficient computations, restart strategies should be applied. We will not go into detail here, but mention that for algorithms with expanding subspaces like the Arnoldi [1] and Jacobi–Davidson [35] methods, restart strategies have been considered in for example [13,26,38].

Algorithm 4.1. Ritz–Galerkin method with inexact Riccati expansion

\begin{verbatim}
input: A, V, ε
W = AV;
M = V*W;  projected matrix from initial approximation V
R = S = W − VM;  initial residual

while ∥S∥ > ε∥R∥
  Q = approximate solution of the Riccati Correction Equation;
  Ŷ = (I − VV*)Q;  orthogonalization of columns of Q against V
  Ỹ = qr(Ŷ);  Ŷ is the orthonormal factor of Ŷ’s QR-decomposition
  Ŵ = AV;
  M = (M V*Ŵ V*Ŵ)  ;  efficient implementation of projection
      (V*Ŵ V*Ŵ) ;  M = (V|Ŵ)(Ŵ|Ŵ) using previous M
  V = (V|Ŷ);  expansion of Galerkin subspace
  W = (W|Ŵ);  updating the matrix W = AV
  S = residual of a new object approximation derived from M and V;
end (while)
\end{verbatim}

Remark 4.1. From an expanded subspace, important indicative information could be obtained about the stability of the object of interest by looking at the approximations of nearby objects that are present in the expanded space. At each projection step it may be decided to select from the subspace a new object of different dimension (for example, by including a Ritz vector belonging to a very near Ritz value) and iterate further with this new object to a hopefully more stable goal.
4.1.1. The (block) Arnoldi method

The (block) Arnoldi method [1,32] can easily be put in the framework given by Algorithm 4.1. This can be seen from the Riccati Correction Equation (13). Taking \(-R\) as a (very rough) approximation to \(Q\), makes that in Algorithm 4.1 a Krylov subspace is built, and an orthonormal basis of it is stored in the matrix \(V\). Note that this approximation \(-R\) to \(Q\) is obtained when one step of Picard iteration (24) is applied using the operator splitting \(T = S - R\) with \(R(Q) = -QM\). As such, the Arnoldi method seems a very unsophisticated method. However, as a numerical method it is very efficient for several reasons. Firstly, the residual is available without any extra computation. Secondly, by intrinsic orthogonality of \(R\) to the subspace, the orthogonalization of the columns of \(Q\) to \(V\) is superfluous, and the matrix \(M\) is the usual block upper Hessenberg matrix of the Arnoldi method. Together with restart strategies of the Arnoldi algorithm like the one developed by Sorensen in [38], this explains the popularity of the method.

4.1.2. The (block) Jacobi–Davidson algorithm

The Jacobi–Davidson algorithm, as originally developed by Sleijpen and Van der Vorst in [35] aims to approximate an eigenvalue \(\lambda\) and eigenvector \(\hat{v}\) of a matrix \(A\). As such, it fits exactly in the above framework: an initial guess \(v\) for \(\hat{v}\) is used for a Ritz–Galerkin projection, which yields a Ritz-value \(\mu\). Then the authors propose to expand the current subspace (which at this moment only consists of \(v\)) by \(\hat{q}\), where \(\hat{q}\) is the solution of the Jacobi–Davidson Correction Equation, which reads as

\[
(I - vv^*)(A - \mu I)(I - vv^*)\hat{q} = -r.
\]

Then, a new Ritz–Galerkin projection takes place, now on the span of \(v\) and \(v + \hat{q}\) and a new Ritz pair is selected to formulate a new correction equation as (41). This process is repeated.

Remark 4.2. Clearly, a Block Jacobi–Davidson method results from Algorithm 4.1 if for the expansion vectors \(\hat{Q}\) we use (approximate) solutions orthogonal to \(X\) of the Sylvester equation \(\Pi A\Pi \hat{Q} - \hat{Q}M = -R\). We did not yet find this algorithm in the research literature.

The solution \(\hat{q}\) of the Jacobi–Davidson Correction Equation can also be interpreted as the result of one Picard iteration step with start value \(q_0 = 0\) applied to the vector version of the Riccati Correction Equation (13),

\[
(I - vv^*)(A - \mu I)(I - vv^*)q = q(v^*A)q - r.
\]

So, the Jacobi–Davidson method uses a better approximation of the solution of the Riccati Correction Equation than the Arnoldi method does, namely \(\hat{q}\) from (41) instead of \(-r\). This correction is computationally more expensive to obtain and is not automatically orthogonal to the subspace built so far. Nor is the projected matrix \(M\) in Algorithm 4.1 upper Hessenberg. So it is not clear which of the two is the better method, and most probably the answer differs for each application. However, once it
has been decided to use Jacobi–Davidson, and to approximate the solution \( \hat{q} \) of the correction equation by means of a Krylov subspace method, it may be worthwhile to consider the efficient methods from Section 3.3 to obtain even better approximations to the solution \( q \) of (42) by re-using Krylov subspaces that were built to approximate the solution \( \hat{q} \) of (41). The main idea of this paper is to use those approximations as expansion vectors instead of the Jacobi–Davidson expansion vectors \( \hat{q} \) from (41).

4.2. New algorithms

Here we advertise two new Ritz–Galerkin methods with expanding subspaces to approximate eigenpairs or invariant subspaces. The philosophy behind them, is that once you have built a Krylov subspace on the start vector \( r \) in (41) in which to approximate \( \hat{q} \), you might as well try to find a better expansion vector from this same subspace. Especially in the case that the Hu–Reichel approach of Section 3.3.3 is used, this leads to an interesting interpretation.

4.2.1. A refined Block Jacobi–Davidson approach

Given the Block Jacobi–Davidson Correction Equation in Remark 4.2, we can solve it by the Hu–Reichel method (see Section 3.3.3) and build an orthonormal basis \( U \) for the block Krylov subspace \( K^m(IIH, R) \), where as before, \( H = I - XX^* \).

Denoting the block upper Hessenberg representation of \( H^{m}(IIH, R) \) by \( H_A \), we can approximate \( \tilde{Q} \) from Remark 4.2 by \( \tilde{Q}_1 = UZ_1 \), where \( Z_1 \) solves the projected Sylvester equation

\[
H_AZ_1 - Z_1M = -U^*R.
\]

Instead of using \( \tilde{Q}_1 \) for expansion of the subspace, we can perform some consecutive iteration steps on the Riccati Correction Equation (13) almost free of charge,

\[
H_AZ_{k+1} - Z_{k+1}M = Z_k(X^*AU)Z_k - U^*R,
\]

if in the first iteration step suitable decompositions of \( H_A \) and \( M \) were computed and stored. We propose to use, for some modest value of \( k > 1 \), \( \tilde{Q}_k := UZ_k \) as expansion block instead, in particular in case Picard iteration on \( H_AZ - ZM = Z(X^*AU)Z - U^*R \) is convergent (see Theorem 3.1).

4.2.2. Block Arnoldi as inner iteration

As mentioned in Section 3.3.3, the iteration in (44) corresponds to computing an eigenvector of a small eigenproblem. Indeed, given the Riccati Correction Equation (13) and the Krylov subspace \( K^m(IIH, R) \) we can form the relatively small projected matrix \( H_A \) from Lemma 3.5, in which \( H \) is the matrix \( H_A \) from Section 4.2.1 above. We may solve for the invariant subspace of \( H_A \) that is in the Frobenius norm closest to \( (I|0)^* \), and use the approximation of \( Q \) obtained this way to expand the current subspace. If the interest is not in the nearest subspace, and when a target is set in the complex plane, one should choose the invariant subspace of \( H_A \) giving (a)
Ritz value(s) closest to that target instead. Given the results of Section 3.3.3 this actually means that, starting with a Galerkin subspace \( V = X \), we:

- perform \( m + 1 \) steps of the Block Arnoldi algorithm applied to \( A \) with start block \( X \) and explicit re-orthogonalization to \( X \) during the process (cf. Lemma 3.5),
- the Arnoldi Ritz block \( \tilde{X} \) giving rise to the favored Rayleigh quotient \( M \) is used to expand the Galerkin space \( V \).

This is repeated until convergence. Clearly, the result is a Block Arnoldi method as inner iteration inside a Jacobi–Davidson-type outer iteration. In fact, Jacobi–Davidson can be interpreted as applying merely one step of inexact inverse iteration to the inner eigenvalue problem, instead of applying Block Arnoldi.

4.2.3. The advantage of a non-linear projected correction equation

Consider once more the vector case of the Riccati Correction Equation,

\[
(I - vv^*)(A - \mu I)(I - vv^*)q = q(v^*A)q - r.
\] (45)

In case \( v \) is close to the eigenvector of interest, the correction \( q \) that is needed will be small in norm, and the quadratic term in \( q \) will have relatively small to no influence. In Section 3 we have seen that the Newton method and other methods will in that case even be rapidly convergent without subspace expansion. Solving small eigenproblems to find an expansion vector seems to be overkill, and the Jacobi–Davidson correction will most probably do the job.

On the other hand, consider the case that the approximation \( v \) is not very good yet, which will of course happen quite often in practical applications. It is important to observe that the correction \( q \) that we are after, still solves (45). Overruling the linearization that leads to the Jacobi–Davidson Correction Equation, and setting up the inner-loop small eigenvalue problem instead, will give at least several possible corrections to choose from, as opposed to the unique correction given by the Jacobi–Davidson correction equation (41). Choosing from the eigenvectors of the small problem the correction that brings \( v \) closest to a given target, seems to be a logical option. One may, alternatively, choose to apply a harmonic Rayleigh–Ritz approach for approximating the inner-loop eigenvalue problem in early stages, as we will explain now.

4.3. Interior eigenvalues

For eigenvalues in the interior of the spectrum, one often prefers the use of harmonic Ritz values instead of Ritz-values. This can, without any complication be implemented, since this concept is independent of the way in which the Galerkin space in Algorithm 4.1 is expanded. We will briefly explain the idea behind it.

---

1 I would like to thank Jasper van den Eshof who pointed this out to me, which resulted in the reparation of an error in the computer code used in the experiments of Section 5.
4.3.1. Harmonic Ritz values and vectors

Consider, very generally, Ritz–Galerkin projection: an eigenvalue problem for a matrix $A$ is approximated by projecting it on a subspace $V$, leading to the Galerkin orthogonality relation

$$V^*(AVy - \theta Vy) = 0.$$ \hspace{1cm} (46)

The success of this approach is optimal if $V$ is an invariant subspace of $A$. In practice of course, we do not have invariant subspaces available, so we hope to project on subspaces that are close to invariant. An interesting observation is the following.

**Observation 4.3.** The space $AV$ is, in an obvious sense, as good an approximation for an invariant subspace of $A^{-1}$ as $V$ is to an invariant subspace of $A$.

This suggests to perform a Ritz–Galerkin projection of $A^{-1}$ on the space $AV$. This would lead to Ritz-values $\theta$ and vectors $AVy$ for $A^{-1}$ by solving

$$(AV)^* \left( A^{-1}(AVy) - \theta(AVy) \right) = 0 \quad \text{or} \quad (AV)^* \left( AVy \frac{1}{\theta} Vy \right) = 0.$$ \hspace{1cm} (47)

Clearly, the eigenvalues of $A^{-1}$ are the reciprocals of the eigenvalues of $A$, while their eigenvectors are the same. This means that the reciprocals of $\theta$ and the vectors $AVy$ can be considered as approximations to eigenpairs of $A$.

**Definition 4.4.** The pairs $(\theta^{-1}, AVy)$ from (47) are called the **harmonic Ritz-pairs** for $A$ with respect to the subspace $V$. Consequently, $\theta^{-1}$ is a **harmonic Ritz value** and $AVy$ a **harmonic Ritz vector**.

The importance of harmonic Ritz pairs stems from the fact that in many Galerkin methods, convergence of the standard Ritz pairs towards extremal eigenvalues is much more pronounced (faster, more regular, even monotone in case $A = A^*$) than to interior eigenvalues. From the construction above it follows that for harmonic Ritz values we may expect that the convergence towards interior eigenvalues is like that. See also [35] for the use of harmonic Ritz values within the Jacobi–Davidson framework.

4.4. Preconditioning

The success of the Jacobi–Davidson method often depends very much on a good preconditioner for the correction equation, and several strategies have already been investigated in the literature. We expect the algorithm in Section 4.2.1 to profit from preconditioning in the same way. Also if other strategies are used than Krylov subspace methods to solve the Jacobi–Davidson Correction Equation (such as incomplete factorizations [36] or domain decompositions [14]), further Picard iterations on the Riccati Correction Equation could make use of the very same preconditioning. Note
that with these other preconditioners, every Picard iteration step will be relatively more expensive, since it does not take place in a small (Krylov) subspace anymore. The advantage that no new factorizations are needed still remains.

5. Numerical experiments

In the previous section, we have discussed several iterative methods for the approximation of invariant subspaces or eigenvectors of large matrices with cheap action. We will now do some numerical tests to find out if the algorithms behave as expected.

5.1. Comparing plain Block Arnoldi and Block Jacobi–Davidson

We have introduced a Block Jacobi–Davidson (BJD) algorithm in Section 4.1.2 and proposed to solve the occurring Sylvester equations by the Hu–Reichel method. The BJD method is a straightforward generalization of the vector-version, but we did not yet find it as such in the research literature. We will now briefly compare BJD with the Block Arnoldi (BA) method numerically. The expectation is, that due to the Newton character of Jacobi–Davidson, BJD will be more efficient than BA, especially when the start vector is close enough to the exact solution.

Our experiments were performed with the following two matrices, where $D$ is the diagonal matrix with eigenvalues $1, 2, \ldots, 100$.

- for $A_1$ we took $D$ and added ones on the two co-diagonals, so $A_1(j, j + 1) = A_1(j + 1, j) = 1$. Apart from that, we set $A_1(100, 1) = A_1(1, 100) = 1/2$. We deliberately took this matrix because it was also used in tests in [7,35].
- $A_2$ is the Sherman 4 matrix from the Harwell–Boeing collection, which is real unsymmetric and $1104 \times 1104$.

For both matrices, we aimed for the invariant subspace belonging to the two largest eigenvalues in magnitude. We started both with an initial subspace chosen randomly, and with one that was within a relative perturbation of ten percent from the target space. In BJD, we used the Hu–Reichel algorithm with a 10-block Krylov subspace to approximate the solution of the correction equation (Figs. 3 and 4).

We depicted the convergence histories of both methods by putting the iteration number on the horizontal axis, and the 10-log of the residual Frobenius norm on the vertical axis. Just as in the non-blocked version of both methods, here too it holds that one BJD iteration is much more expensive than one BA iteration because a Krylov subspace is being built to solve for a correction, and orthogonalization in the outer iteration is needed.

Both experiments confirm what is known for the non-block versions, which is that close to the solution, BJD converges much more rapidly and will most probably also be cheaper than BA even though each iteration is much more expensive. If,
however, the start space is not close to the target, the situation is less clear. BJD needs an initial phase in which it expands the Galerkin space in order to get close to the solution. Once that has been done, convergence is fast again, but the total computational costs may be higher than for BA. We will now show that one of our new methods drastically reduces the initial phase for (B)JD.

5.2. The new methods

We have introduced two possible refinements of the (B)JD method in Section 4.2. The first method we called Refined (B)JD and will be abbreviated by R(B)JD. It involves additional Picard iteration on the Riccati Correction Equation in the same subspace that was used to find an approximation of the (B)JD correction equation. The main idea of the second method is not to linearize the Riccati Correction Equation, but to replace it by a low dimensional Riccati Equation by projection on a subspace. This subspace may be chosen as the same subspace that would normally have
been built for the (B)JD correction equation. By non-linearity of the projected Riccati equation, this gives a small number of possible corrections to choose from, instead of the unique correction resulting from linearization in (B)JD. Since the (B)JD correction is designed to point in the direction of the closest invariant subspace, we expect that (B)JD will be less efficient than our new algorithm whenever the closest invariant subspace is not the one that we are after. We already explained that new algorithm can be seen as a (B)JD method in the outer iteration combined with (B)A in the inner iteration.

**Definition 5.1 (Riccati method).** Looking ahead at its success, instead of calling it (B)JD(B)A we would like to refer to the man who introduced the Riccati equation, J.F. Riccati (1676–1754). So we will call it from now on the Riccati method.

Clearly, both methods can be implemented on top of a (B)JD code in which the correction equation is solved by a subspace method. As long as the subspace has relatively low dimension, the costs of solving the extra Picard iteration(s) of the first method, as well as the costs for solving the low dimensional Riccati equation in the second method, are rather low. In our numerical tests, we will only work with Krylov subspaces.

5.3. A typical experiment

To start, we took the matrix $A_1$ from the previous experiments and compared JD, RJD and Riccati. For computational simplicity, we aimed for eigenvectors rather than invariant subspaces, moreover since it showed that the Block versions behaved similarly. We took $m = 10$ for each method (Fig. 5).

This experiment turns out to be typical. RJD usually does not differ much from the JD since the refinement still aims for the closest eigenvector. But Riccati, especially when far away from the solution, exploits the whole Krylov subspace that was built

![Fig. 5. Comparing JD (asterisks) and RJD (diamonds) and Riccati (circles) in convergence behaviour for the matrix $A_1$. In (a) we started close to the solution, whereas in (b) we started randomly.](image-url)
for solving the JD correction equation. Not with the aim to solve this equation, but to get close to the solution quickly. The dependence on \( m \) is as follows. Taking the same random start for each experiment, we get typically:

<table>
<thead>
<tr>
<th>( m )</th>
<th>( 3 )</th>
<th>( 5 )</th>
<th>( 10 )</th>
<th>( 20 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cpu time speed-up factor, Ric/JD</td>
<td>1.78</td>
<td>0.52</td>
<td>0.18</td>
<td>0.05</td>
</tr>
<tr>
<td>Number of iterations speed-up factor, Ric/JD</td>
<td>0.97</td>
<td>0.75</td>
<td>0.46</td>
<td>0.20</td>
</tr>
</tbody>
</table>

Here, the number 0.18 for \( m = 10 \) indicates that Riccati needed only 0.18 times the cpu-time that JD needed to get the same accuracy with the same start vector, whereas the number 0.46 is the fraction of iterations of JD that Riccati needed to do so. Since R(B)JD never did not seem to be structurally much better than (B)JD, we will from now on only experiment with JD versus the Riccati method, which seems to do very well.

5.4. Jacobi–Davidson versus Riccati for large matrices

First, we repeated the experiment of Section 5.3 with the Sherman 4 matrix, but only with JD and its best competitor, Riccati. In Fig. 6, the convergence histories are depicted for \( m = 10 \), starting both close to the solution (a) and randomly (b). The behaviour seems the same as in Section 5.3.

The corresponding relative speed-up factors are present as a part of the following tabular, in which we also give results for some other matrices from the Harwell Boeing collection. We always started with a random vector, but kept it the same for all experiments with the particular matrix. Dashes mean that no experiment was done. We repeated the experiment in case of misconvergence, which happened once for JD and \( m = 20 \) in lshp3466.

![Fig. 6. Comparing JD (asterisks) and Riccati (circles) in convergence behaviour for the matrix Sherman 4. In (a) we started close to the solution, whereas in (b) we started randomly.](image-url)
To illustrate the success of the Riccati method over plain Jacobi–Davidson once more, we presented the two convergence graphs corresponding to the matrices memplus for $m = 5$ and bcstk32 for $m = 10$ in Fig. 7. Note that the extra costs for the Riccati method per iteration step due to solving $5 \times 5$ and $10 \times 10$ eigenvalue problems, are negligible.

### 5.5. Hardware and software and real arithmetic

Our experiments were carried out on a SunBlade 100 workstation using MatLab 6, in which the flop-count is unfortunately no longer available. Instead, we used the cpu-time count, but note that this is not always reliable.
We think it is natural to work only with real invariant subspaces when real matrices are used in experiments. Therefore, in the actual implementation, all arithmetic was kept real. If complex conjugate eigenpairs were involved, we selected real invariant subspaces of dimension two rather than complex eigenvectors. This could easily be implemented since our algorithms are block algorithms anyway.

MatLab’s Schur decomposition is a real Schur decomposition, and we used the MatLab code developed in [5] to sort the real Schur forms with respect to targets, and to find the two-dimensional real invariant subspaces belonging to complex conjugate pairs. Also for finding a single real eigenvector, the combination of computing the real Schur form and the sorting algorithm was used. The dominant eigenvalue of nncl1374 is in fact a complex conjugate pair. It was returned by all methods as a two-by-two real matrix together with the corresponding $1374 \times 2$ real invariant subspace.

5.6. Discussion

The experiments show that the Riccati method, which has several potential corrections per iteration step, as opposed to only one in Jacobi–Davidson, generally performs much better than Jacobi–Davidson. Of course, the comparison is not always fair. If it is known that the start vector is possibly far from the solution, there is not much sense in solving the Jacobi–Davidson correction equation more and more accurately, since even the exact solution would not give a good correction. So, also the best preconditioner would not help Jacobi–Davidson in that case. A plausible alternative would be to apply first the Arnoldi method until one is close enough to the target solution, and then switch to Jacobi–Davidson. But this, in fact, already very much resembles the Riccati method, given its interpretation in Section 4.2.2. It would be interesting to find a heuristic for a flexible value of $m$ in relation to the size of the residual of the current approximation.

Note that the number of matrix–vector multiplications with $A$ is the same for both methods, and that the difference in costs is the solution of small eigenvalue problems in each iteration of the Riccati method. In our program, this is done by brute force (computing all eigendata and choosing the best correction), but one could think of a recursive strategy in which the Riccati method itself is used. In that case, it would pay off to consider the harmonic approach to favor convergence towards eigenvectors belonging to eigenvalues in the interior of the spectrum.

Acknowledgements

The author wishes to thank Jasper van den Eshof, Jaroslav Kautsky, Michal Křížek and Gerard Sleijpen for valuable discussions, and Rich Lehoucq moreover for pointing out reference [33]. The research has been made possible through a Fellowship of the Royal Netherlands Academy of Arts and Sciences (KNAW).
References


[34] G.L.G. Sleijpen, F.W. Wubs, Effective preconditioning techniques for eigenvalue problems, Preprint 1117, Mathematics Department, University Utrecht, Netherlands, 1999.


