Numerical Solution of Special Algebraic Riccati Equations via an Exact Line Search Method

Peter Benner

Zentrum für Technomathematik
Fachbereich 3 – Mathematik und Informatik
Universität Bremen
D-28334 Bremen (FRG).
Fax : +49 421 2184863 and e-mail: benner@numerik.uni-bremen.de

Abstract

We study the numerical solution of a class of algebraic Riccati equations arising in spectral factorization and $\mathcal{H}_\infty$ optimal control problems. This type of matrix Riccati equations differs from the standard type usually considered in linear-quadratic regulator problems in that both the quadratic and constant term are positive semidefinite. We show that the exact line search method derived for the standard "continuous-time" algebraic Riccati equation can be applied to this type of equation as well. The method can either be used to solve the Riccati equation iteratively or to improve a solution computed by any other method via iterative refinement.

Key words: Numerical Methods, Optimal Control, $\mathcal{H}_\infty/\mathcal{L}_\infty$.

1 Introduction

In this paper, we consider the algebraic Riccati equation (ARE)

$$0 = R(X) := Q + A^T X + XA + XGX \qquad (1)$$

where $A, G, Q, X \in \mathbb{R}^{n\times n}$. Here, A is *stable*, i.e., has all its eigenvalues in the open left half plane, G and Q are symmetric positive semidefinite, and X is the symmetric sought-after solution of (1). The above equation differs from the continuous-time algebraic Riccati equation (CARE) arising in the linear-quadratic regulator problem in the sign of the quadratic term, or, in other words, the CARE can be given as in (1) assuming G negative semidefinite and Q positive semidefinite (without the additional assumption of A being stable).

The ARE (1) arises, e.g., in spectral factorization and $\mathcal{H}_\infty$ optimal control problems (see, e.g., [9] and the references given therein), and related model reduction problems (see, e.g., [14] and the references given therein). As for the CARE, usually the desired solution X_s of (1) is *stabilizing* in the sense that the matrix $A + GX_s$ is stable.

The *Hamiltonian* matrix associated with (1) is

$$H = \begin{bmatrix} A & G \\ -Q & -A^T \end{bmatrix}. \qquad (2)$$

As a consequence of the *real bounded lemma* (see, e.g., [9] and many other references given therein), the ARE (1) with A stable has a unique symmetric positive semidefinite stabilizing solution X_s if and only if H has no eigenvalues on the imaginary axis. Throughout this paper, we will assume the existence of such a stabilizing solution X_s. For a discussion of other solutions of (1) see also [7].

The stabilizing solution of the ARE (1) can therefore be determined analogously to the stabilizing solution of the CARE by computing a certain invariant subspace of the Hamiltonian matrix H given in (2): if we assume that H has no eigenvalues on the imaginary axis, then H has exactly n eigenvalues in the open left half plane. Thus, there exists an n–dimensional H–invariant subspace corresponding to these eigenvalues — the *stable* invariant subspace of H. Let this invariant subspace be spanned by the columns of $\begin{bmatrix} U \\ V \end{bmatrix}$, $U, V \in \mathbb{R}^{n\times n}$, and assume that U is nonsingular. Then $X_s = VU^{-1}$. Therefore, any numerical method solving the CARE by means of the stable H-invariant subspace can also be used to solve the ARE (1).

Another approach used to solve the CARE is Newton's method, first proposed by Kleinman [8]. Since the proof of convergence of Newton's method to the stabilizing solution of the CARE relies upon the definiteness assumptions on the coefficient matrices G and Q it is not straightforward to apply this method to the ARE (1). However, in [13] it is proved that Kleinman's method applied to the ARE (1) has the same convergence properties as for the CARE.

For several reasons, Newton's method is mainly used for iterative refinement of Riccati solutions computed by some other method rather than for directly solving the CARE. One of these reasons is that to converge to the stabilizing solution, the starting point has to be stabilizing. There exist stabilization procedures that provide a stabilizing initial guess (e.g., [12]), but often these lead far away from the desired solution X_s. In the case considered here, the matrix A of (1) is stable, and thus $X_0 = 0$ is a simple stabilizing starting guess. Still, Newton's method may be too expensive from the point of view of computational cost. The most commonly used method for solving the CARE during the last 15 years is the Schur vector method [11] which essentially computes the stable H-invariant subspace by applying the standard QR algorithm (e.g., [6]) to the Hamiltonian matrix H and then reordering the eigenvalues. Solving the CARE by this method requires roughly the same computation time as seven steps of Newton's method. Usually, the solution computed by the Schur vector method is improved by iterative refinement. Often, when used as an iterative solver of the ARE (1) or the CARE, the number of iterations required by Newton's method to converge is much higher than seven. On the other hand, used as defect correction or iterative refinement method, usually only one or two Newton iterations are required to obtain the maximum possible accuracy. Having this in mind, we can conclude that in order to be competitive with the Schur vector approach, Newton's method should terminate in at most nine to ten iterations.

In [3], an exact line search method is introduced which improves Newton's method for the numerical solution of the CARE in several aspects. There, the Newton iteration step is modified by computing an optimal step size along the Newton direction in order to minimize the Frobenius norm of the next residual. This is achieved by little extra cost such that compared to a standard Newton step, an exact line search Newton step requires only about 5–10% more work. Numerical examples suggest that often, this exact line search method computes the solution of the CARE at a computational cost competitive to the Schur vector method. Moreover, the method has the same pleasant properties as Newton's method when used for iterative refinement. Sometimes, it even then compares favorably to Newton's method when the CARE is ill-conditioned.

In the next section, we will introduce the exact line search method for the numerical solution of the ARE (1). A numerical example will be given in Section 3.

2 Exact Line Search

For the ease of notation, we will introduce the following *Lyapunov operators*:

$$\Omega_j(Z) = (A + GX_j)^T Z + Z(A + GX_j),$$

where $Z \in \mathbb{R}^{n\times n}$ and $X_j = X_j^T \in \mathbb{R}^{n\times n}$, $j = 0, 1, \ldots$, will be the iterates generated by the following algorithm. (It is assumed that Ω_j is a nonsingular operator for all j.)

A general framework for the algorithms considered here is given in the following algorithm.

Algorithm 1

```
1. Set $X_0 = 0$.
2. FOR $j = 0, 1, 2, \ldots$
     2.1 $N_j = -\Omega_j^{-1}(R(X_j))$.
     2.2 $X_{j+1} = X_j + t_j N_j$.
   END FOR
```

In case $t_j = 1$ for all j, this is Newton's method for the ARE (1). Note that this formulation differs from the one given in [13],

$$\begin{aligned} X_0 &= 0, \\ X_{j+1} &= \Omega_j^{-1}(X_j G X_j - Q), \quad j \geq 0, \end{aligned} \tag{3}$$

in that it does not compute X_{j+1} directly from the Lyapunov equation but computes first the *Newton step* N_j and then updates X_j by adding the Newton step. Our formulation is analogous to the usual formulation of Newton's method for the solution of nonlinear equations as given in [5, Algorithm 5.1.1] while the formulation (3) is based on the original formulation of Newton's method for the CARE by Kleinman [8]. While mathematically equivalent, (3) has the disadvantage that a possible ill-conditioned Lyapunov operator Ω_j directly affects the accuracy of the approximation X_{j+1}. In Algorithm 1, the same Lyapunov operator is used to solve the Lyapunov equation, but an ill-conditioning only affects the accuracy of the step N_j. However, in general, it is sufficient that the first significant digits of N_j are accurate in order to correct the most significant wrong digits of X_j and thereby obtain a more accurate approximation $X_{j+1} = X_j + t_j N_j$.

Our aim here is to choose the parameter t_j in Algorithm 1 in order to minimize some measure of the error for the next approximation X_{j+1}. In optimization theory, this is called a *line search* where the search direction is the Newton direction, given by the step N_j. A line search is said to be *exact* (as opposed to *approximate*) if t_j is an exact minimizer. Here, we choose the Frobenius norm of the next residual as an error measure, i.e., we try to find $t = t_j$ minimizing

$$f_j(t) = \|R(X_j + tN_j)\|_F^2 \tag{4}$$

which is equivalent to minimizing $\|R(X_j + tN_j)\|_F$. We will see that such a t_j can be computed without adding a prohibitive extra cost to the Newton step.

Using the identity given by the Lyapunov equation in Step 2.1 of Algorithm 1, it is easy to see that [3]

$$R(X_j + tN_j) = (1-t)R(X_j) + t^2 N_j G N_j. \tag{5}$$

Using the well-known property of the Frobenius norm that $\|M\|_F^2 = \mathrm{trace}\left(\mathrm{M^T M}\right)$, we have

$$f_j(t) = \alpha_j(1-t)^2 + 2\beta_j(1-t)t^2 + \gamma_j t^4, \tag{6}$$

where

$$\alpha_j \;=\; \mathrm{trace}\left(\mathrm{R(X_j)^2}\right),$$

$$\begin{aligned}\beta_j &= \mathrm{trace}\left(\mathrm{R}(\mathrm{X_j})\mathrm{N_j}\mathrm{G}\mathrm{N_j}\right),\\ \gamma_j &= \mathrm{trace}\left((\mathrm{N_j}\mathrm{G}\mathrm{N_j})^2\right).\end{aligned}$$

Thus, f_j is a polynomial of degree at most four. If $\gamma_j = 0$, $t_j = 1$ minimizes f_j, and $X_j + t_j N_j$ is a solution of (1). Therefore, in the sequel we will assume that $\gamma_j > 0$. This implies that f_j has either one local minimum and no local maxima or two local minima and one local maximum. Since $\lim_{t\to\pm\infty} f_j(t) = \infty$, one of these local minima is also the global minimum.

Theorem 1 *There is a local minimum at some value $t_j \in [\,0,\ 2\,]$.*

Proof: Differentiating (4), using (5), and defining

$$V_j := N_j G N_j,$$

we obtain

$$f_j'(t) = -2\cdot \mathrm{trace}\Big(\left(\mathrm{R}(\mathrm{X_j}) - 2\mathrm{t}\mathrm{V_j}\right)\left((1-\mathrm{t})\mathrm{R}(\mathrm{X_j}) + \mathrm{t}^2\mathrm{V_j}\right)\Big). \quad (7)$$

Thus,

$$f_j'(0) = -2\cdot\mathrm{trace}\left(\mathrm{R}(\mathrm{X_j})^2\right) \le 0,$$

and

$$f_j'(2) = 2\cdot\mathrm{trace}\left((\mathrm{R}(\mathrm{X_j}) - 4\mathrm{V_j})^2\right) \ge 0.$$

If $f_j'(0) = 0$, then $\|R(X_j)\|_F = 0$ and $t_j = 0$. Otherwise, $f_j'(0) < 0$ and there must be a local minimum at some value $t_j \in (\,0,\ 2\,]$. □

From the proof of Theorem 1 we immediately obtain the following result.

Corollary 2 *a) $t_j = 0$ if and only if $R(X_j) = 0$, i.e., if X_j is a solution of (1).*
b) $\|R(X_j)\|_F \le \|R(X_j + t_j N_j)\|_F$ and equality holds if and only if $R(X_j) = 0$.

Note that Theorem 1 and Corollary 2 do not require any assumptions on the coefficients of the ARE (1) except G and Q being symmetric. Thus, they both hold for general AREs of the form (1).

In the following we will see that the interval $[\,0,\ 2\,]$ turns out to be the obvious search interval for the suggested line search procedure.

Lemma 3 *If $A + GX_j$ is stable, $t \in [\,0,\ 2\,]$, and*

$$N_j = -\Omega_j^{-1}(R(X_j)), \quad (8)$$

then $A + G(X_j + tN_j)$ is stable.

Proof: The identity (8) is equivalent to

$$\Omega_j\,(X_j + N_j) = -Q + X_j G X_j. \quad (9)$$

Subtracting (9) from $R(X_s) = 0$ and adding $X_j G X_s + X_s G X_j$ on both sides yields

$$\Omega_j\,(X_s - (X_j + N_j)) = -(X_s - X_j)G(X_s - X_j). \quad (10)$$

Using a modified version of Lyapunov's theorem (see, e.g., [10, Chapter 13, Proposition 1]), the stability of $A + GX_j$ implies $X_s - (X_j + N_j) \ge 0$. Now adding

$$tN_j G\big(X_s - (X_j + N_j)\big) + \big(X_s - (X_j + N_j)\big)GtN_j$$

on both sides of (10), we obtain

$$\begin{aligned}&(A + G(X_j + tN_j))^T\,(X_s - (X_j + N_j)) + \\ &\quad + \ (X_s - (X_j + N_j))\,(A + G(X_j + tN_j))\\ = &\quad - \ (X_s - (X_j + tN_j))\,G\,(X_s - (X_j + tN_j))\\ &\quad - \ t(2-t)N_j G N_j \qquad (11)\\ =: &\quad W.\end{aligned}$$

Since $t \in [\,0,\ 2\,]$, the right-hand side W in (11) is negative semidefinite.

Now suppose $A + G(X_j + tN_j)$ has an eigenvalue λ with $\mathrm{Re}(\lambda) \ge 0$ and corresponding eigenvector $z \ne 0$. Then we have

$$(A + G(X_j + tGN_j))\,z = \lambda z. \quad (12)$$

Multiply (11) from the left by z^H and from the right by z. Then we obtain

$$2\cdot\mathrm{Re}(\lambda) z^H\,(X_s - (X_j + N_j))\,z = z^H W z. \quad (13)$$

The left-hand side of (13) is nonnegative since

$$X_s - (X_j + N_j) \ge 0$$

and $\mathrm{Re}(\lambda) \ge 0$. As W is negative semidefinite, the right-hand side of (13) is nonpositive and it follows that

$$z^H W z = 0.$$

Thus,

$$z^H\,(X_s - (X_j + tN_j))\,G\,(X_s - (X_j + tN_j))\,z = 0$$

and since $G \ge 0$, this implies

$$G\,(X_s - (X_j + tN_j))\,z = 0,$$

or, equivalently,

$$GX_s z = G(X_j + tN_j)z. \quad (14)$$

From (12) and (14) we obtain

$$\begin{aligned}\lambda z &= (A + G(X_j + tN_j))\,z\\ &= (A + GX_s)\,z.\end{aligned}$$

Hence, λ is an eigenvalue of $A + GX_s$ which contradicts the stability of $A + GX_s$. □

Theorem 1 and Lemma 3 show that in each step of Algorithm 1, we can find a $t_j \in [\,0,\ 2\,]$ minimizing $\|R(X_{j+1})\|_F$ and that all iterates are stabilizing as long as X_0 was chosen to be stabilizing. Since A is stable, $X_0 = 0$ satisfies this last demand. Lemma 3 also shows that if t_j is chosen from $[0, 2]$, Algorithm 1 cannot fail due to a singular Lyapunov operator Ω_j.

Even if limited to $[0, 2]$, there is some ambiguity in the choice of t_j. As we have seen before, f_j may have two local minima. Because of Lemma 3, we only have a choice if both of them are contained in $[0, 2]$. In that case, we can choose the one defining the global minimum. In practise, this is very seldom observed. In most cases, there is only one local minimum (which then is the global minimum). Two local minima in $[0, 2]$ occured only twice while testing Algorithm 1 with exact line search for some hundred examples.

Let us briefly discuss the additional computational cost necessary to perform an exact line search. Usually, the Lyapunov equation in Step 2.1 is solved using the *Bartels-Stewart* algorithm [2]. Thus, using flop estimates from [6], performing one iteration step of Algorithm 1 requires about $36n^3$ flops. (Following [6], we define each floating point arithmetic operation together with the associated integer indexing as a *flop*.) The computation of the symmetric matrix $N_j G N_j$ can be done using $3n^3$ flops while the computation of the coefficients α_j, β_j, and γ_j of f_j requires $3n$ inner products of length n, i.e., $6n^2$ flops. The computation of the minimizing t_j is an $O(1)$ operation negligible compared to the $O(n^3)$ cost of the iteration step. We can conclude that an exact line search does increase the computational cost of one iteration step by no more than 10%. In other words, the extra work is amortized if we can save one out of ten iteration steps. In many control applications, the coefficient matrix G of the quadratic term in (1) is given as $G = BB^T$ where $B \in \mathbb{R}^{n\times m}$. If $m \ll n$, then this comparison becomes even more favourable. For instance, if $m = 1$, the additional cost is about $10n^2$, i.e., there is no significant extra cost for an exact line search.

In [3], a convergence theory for Algorithm 1 with exact line search iss derived. The same results can be obtained for the ARE (1). Since the proofs only differ marginally from those given in [3] for the CARE, we omit them here and only summarize the results in the following theorem.

Theorem 4 *Assume that A is stable and (A, G) defines a controllable matrix pair. Let X_j denote the iterates produced by Algorithm 1 where in each step the t_j are chosen to minimize $\|R(X_j + tN_j)\|_F$ in $[0, 2]$. If $t_j > \varepsilon$ for all j and some $\varepsilon > 0$, then*

$$\lim_{j\to\infty} X_j = X_s.$$

Moreover, convergence is quadratic in a neighbourhood of X_s.

The above convergence result relies on the fact that $t_j > \varepsilon$ for all j and a given constant $\varepsilon > 0$. We can modify Algorithm 1 such that the step size is set to one if t_j drops below a prescribed (small) constant. We can then "re-start" Algorithm 1 with the new "starting guess" $X_j + N_j$ which is stabilizing by Lemma 3. In our numerical experiments, very small step sizes occured only at the very beginning of the iteration if the starting guess already yielded a residual norm within the order of the limiting accuracy. In such a case, neither Newton's method nor exact line search can be expected to improve the accuracy of the approximate solution of (1) any further.

Furthermore, Theorem 1 relies on the controllability of (A, G). This assumption is needed in parts of the proofs in [3]. Numerical experiments with uncontrollable data (as, for instance, the example given in Section 3) suggest that this assumption can be removed. The iterates produced by Algorithm 1 with exact line search always converged to the stabilizing solution X_s.

3 Numerical Example

We compared Newton's method (NWT) as given by Algorithm 1 with $t_j = 1$ for all j, and Algorithm 1 with exact line search (ELS), i.e., in each step the t_j are computed to minimize $\|R(X_j + tN_j)\|_F$ for the following example given in [14, 13].

Consider a transfer function

$$G(s) = C(sI_n - A)^{-1}B + D$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, and $D \in \mathbb{R}^{p\times m}$. Let (A, B, C, D) be a stable minimal state-space realization of $G(s)$. Define the power spectrum matrix $\Phi(s) = G(s)G^T(-s)$, and let W be a *square minimum phase right spectral factor* of Φ, satisfying $\Phi(s) = W^T(-s)W(s)$. If we assume that D has full row rank, then $R = DD^T$ is positive definite and a minimal state space realization (A_W, B_W, C_W, D_W) of W is given by (see [1])

$$\begin{aligned} A_W &= A, \\ B_W &= BD^T + P_W C^T, \\ C_W &= R^{-\frac{1}{2}}(C - B_W^T X_W), \\ D_W &= R^{\frac{1}{2}}. \end{aligned}$$

Here, P_W is the solution of

$$AP + PA^T = -BB^T$$

and X_W is the stabilizing solution of the ARE

$$\begin{aligned} (A - B_W R^{-1} C)^T X &+ X(A - B_W R^{-1}C) + \\ X B_W R^{-1} B_W^T X &+ C^T R^{-1} C = 0. \end{aligned} \qquad (15)$$

In [14], it is observed that $X_W > 0$ as it is the observability Gramian of W and thus, using a version of Lyapunov's theorem [10, p.447], $A - B_W R^{-1} C$ is stable which shows that (15) is of the form (1).

In [14], the computation of X_W (and P_W) is part of a *square-root balancing-free stochastic truncation model reduction* algorithm. This algorithm does not rely upon the minimality of (A, B, C, D).

As an example consider a transfer function $G(s)$ having a tenth order non-minimal state-space realization (neither controllable nor observable) (A,B,C,D) where (see [13, 14])

$$A = \begin{bmatrix} -6 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -10 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -8 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -13 & -3 & 9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -8 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -14 & -9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 \end{bmatrix},$$

$$B^T = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 10^{-3} \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 10^{-3} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$C = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 5 \cdot 10^{-5} \\ 0 & 0 & 0 & 0 & 0 & 0 & -6 & 1 & -2 & 5 \cdot 10^{-5} \end{bmatrix},$$

$$D = 10^{-\alpha} \times \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The two methods NWT and ELS were implemented in MATLAB 4.2c[1] and tested by solving (15) for the data given above (and many other examples). Table 1 shows the number of iterations (NIT) and the final residual $\|R(X_{NIT})\|_F$ for each of the two methods when used to solve (15) for varying matrices D ($\alpha = 0, 1, \ldots, 6$).

	NWT		ELS	
α	NIT	$\|R(X_{NIT})\|_F$	NIT	$\|R(X_{NIT})\|_F$
0	2	1.5×10^{-14}	2	8.2×10^{-15}
1	3	1.4×10^{-12}	3	1.6×10^{-13}
2	6	7.4×10^{-11}	5	6.5×10^{-11}
3	10	9.2×10^{-9}	6	8.6×10^{-9}
4	14	1.9×10^{-6}	7	1.8×10^{-6}
5	18	4.4×10^{-4}	8	2.7×10^{-4}
6	22	7.1×10^{-2}	8	8.8×10^{-2}

Table 1: Comparison of NWT and ELS

As expected, Table 1 shows that the final residuals signal the same attained accuracy for both methods which is the limiting accuracy to be expected from estimates of the condition number K_{ARE} of the ARE as proposed in [4] ($K_{ARE} = O(10^{2\alpha})$). But while ELS stays in the region of computational cost of the Schur vector method, Newton's method becomes prohibitively expensive for smaller values of α. Note further that for all considered values of α, the solution of the Schur vector method could be improved by one step of either Newton's method or ELS to attain the limiting accuracy as given by the residuals in the table.

[1]MATLAB is a registered trademark by the MathWorks, Inc.

4 Conclusions

An exact line search method for the numerical solution of special algebraic Riccati equations as they arise in spectral factorization and $\mathcal{H}_\infty$ optimal control problems has been proposed. This method can be used as an iterative solution method for this class of equations or as an iterative refinement method. Used as an iterative refinement method, it shows the same fast convergence behaviour as Newton's method, while it is often competitive as an iterative solver when the cost for Newton's method used as an iterative solver for (1) is prohibitive. The method can be applied whenever a maximum-accuracy solution of the ARE (1) is required.

References

[1] B. D. O. Anderson, *A system theory criterion for positive real matrices*, SIAM J. Cont., 5 (1967), pp. 171–182.

[2] R. Bartels and G. Stewart, *Solution of the matrix equation $AX + XB = C$: Algorithm 432*, Comm. ACM, 15 (1972), pp. 820–826.

[3] P. Benner and R. Byers, *An exact line search method for solving generalized continuous-time algebraic Riccati equations*, IEEE Trans. Automat. Control, *to appear 1997*.

[4] R. Byers, *Numerical condition of the algebraic Riccati equation*, Contemp. Math., 47 (1985), pp. 35–49.

[5] J. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice Hall, Englewood Cliffs, New Jersey, 1983.

[6] G. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, second ed., 1989.

[7] H. Kano and T. Nishimura, *Nonnegative-definite solutions of algebraic matrix Riccati equations with nonnegative-definite quadratic and constant terms*, in Systems and Networks: Mathematical Theory and Applications, U. Helmke, R. Mennicken, and J. Saurer, eds., vol. II, Akademie Verlag, 1994, pp. 265–268.

[8] D. L. Kleinman, *On an iterative technique for Riccati equation computations*, IEEE Trans. Automat. Control, AC-13 (1968), pp. 114–115.

[9] P. Lancaster and L. Rodman, *The Algebraic Riccati Equation*, Oxford University Press, Oxford, 1995.

[10] P. Lancaster and M. Tismenetsky, *The Theory of Matrices*, Academic Press, Orlando, 2nd ed., 1985.

[11] A. Laub, *A Schur method for solving algebraic Riccati equations*, IEEE Trans. Automat. Control, AC-24 (1979), pp. 913–921. (see also *Proc. 1978 CDC (Jan. 1979)*, pp. 60-65).

[12] V. Sima, *An efficient Schur method to solve the stabilization problem*, IEEE Trans. Automat. Control, AC-26 (1981), pp. 724–725.

[13] A. Varga, *On computing high accuracy solutions of a class of Riccati equations*, Control–Theory and Advanced Technology, 10 (1995), pp. 2005–2016.

[14] A. Varga and K. H. Fasol, *A new square–root balancing–free stochastic truncation model reduction algorithm*, in Prepr. 12th IFAC World Congress, vol. 7, Sydney, Australia, 1993, pp. 153–156.