An Exact Line Search Method for Solving Generalized Continuous-Time Algebraic Riccati Equations

Peter Benner and Ralph Byers

***Abstract*— We present a Newton-like method for solving algebraic Riccati equations that uses Exact Line Search to improve the sometimes erratic convergence behavior of Newton's method. It avoids the problem of a disastrously large first step and accelerates convergence when Newton steps are too small or too long. The additional work to perform the line search is small relative to the work needed to calculate the Newton step.**

I. Introduction

We study the generalized continuous-time algebraic Riccati equation (CARE)

$$\begin{aligned}0 &= R(X)\\ &= C^TQC + A^TXE + E^TXA\\ &\quad - (B^TXE + S^TC)^TR^{-1}(B^TXE + S^TC). \qquad (1)\end{aligned}$$

Here $A, E, X \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}, R = R^T \in \mathbb{R}^{m\times m}, Q = Q^T \in \mathbb{R}^{p\times p}, C \in \mathbb{R}^{p\times n}$, and $S \in \mathbb{R}^{p\times m}$. We will assume that E is nonsingular and $R > 0$, where $M > 0$ $(M \geq 0)$ denotes positive (semi-) definite matrices M. In principle, by inverting E, (1) may be reduced to the case $E = I$. This is convenient for studying convergence behavior of the numerical method presented here (see Section III). However, when E is ill-conditioned (i.e., nearly singular), this may introduce instability in numerical computations. Therefore, the algorithm derived here avoids inverting E.

Often, the desired solution X is *stabilizing* in the sense that the eigenvalues of the matrix pencil $E - \lambda(A - BR^{-1}(B^TXE + S^TC)$ have negative real parts. We denote this by $\lambda(E, A - BR^{-1}(B^TXE + S^TC)) \subset \mathbb{C}^-$. Assuming (E, A, B) strongly stabilizable and (E, A, C) strongly detectable, such a stabilizing solution exists and is unique [23]. Throughout this paper, we call the stabilizing solution X^*.

We also use the following notation. The *Frobenius* norm or *Euclidean* norm of a matrix M is defined by $\|M\|_F^2 = \mathrm{trace}(M^TM)$. For any symmetric matrix M, we have $\|M\|_F^2 = \mathrm{trace}(M^2)$, and for any two matrices M and $N, \mathrm{trace}(MN) = \mathrm{trace}(NM)$. Following [13], we define each floating point arithmetic operation together with the associated integer indexing as a *flop*.

The algebraic Riccati equation (1) is a nonlinear system of equations. One of the oldest, best studied numerical methods for solving (1) is Newton's method [9], [14], [18], [23], [26].

Manuscript received February 12, 1996. This work was supported in part by the National Science Foundation under Grants INT-8922444, CCR-9404425, and DMS-9628626 and the Kansas Institute for Theoretical and Computational Science.

P. Benner was with the University of Kansas, Lawrence, KS 66045 USA. He is now with Fakultät für Mathematik, Technische Universität, Chemnitz-Zwickau, 09107 Chemnitz, Germany.

R. Byers is with the Department of Mathematics, University of Kansas, Lawrence, KS 66045 USA.

Publisher Item Identifier S 0018-9286(98)01176-3.

Algorithm 1 (Newton's Method for Solving CARE)

```
1.  Choose some initial starting guess $X_0 = X_0^T$.
2.  FOR $j = 0, 1, 2, \cdots$
      2.1  $K_j \leftarrow R^{-1}(B^TX_jE + S^TC)$.
      2.2  Solve for $N_j$ in the Lyapunov equation
             $(A - BK_j)^TN_jE + E^TN_j(A - BK_j)$
             $= -R(X_j)$.
      2.3  $X_{j+1} \leftarrow X_j + N_j$.
    END FOR.
```

If (E, A, B) is strongly stabilizable, (E, A, C) is strongly detectable, and X_0 is stabilizing, then Algorithm 1 converges to the desired stabilizing solution X^* [18], [23], [26]. Ultimately, convergence is quadratic. At each step, $\lambda(E, A - BK_j) \subset \mathbb{C}^-$, and *after* the first step, convergence is monotone. (Algorithm 1 also provides all the ingredients for a condition estimate of CARE and N_j is an estimate of the error $X^* - X_j$ [7].)

Because of its robustness in the presence of rounding errors, we prefer to calculate the Newton step explicitly as in Algorithm 1 rather than to use the mathematically equivalent formulation [9], [18], [23], [26]

$$\begin{aligned}(A - BK_j)^TX_{j+1}E + E^TX_{j+1}(A - BK_j)\\ = -C^T(Q - SR^{-1}S^T)C - E^TX_jBR^{-1}B^TX_jE\end{aligned}$$

which determines X_{j+1} directly. The coefficient matrices of the two Lyapunov equations are the same, but the right-hand sides are different. Loosely speaking, if the condition number of the coefficients permits us to solve the Lyapunov equation to (say) k correct significant digits, and X_{j+1} is calculated directly, then its accuracy is limited to k significant digits. However, in Algorithm 1, it is the rounding error corrupted Newton step N_j that is limited to k significant digits. The sum $X_j + N_j$ has roughly k more correct digits than X_j. The accuracy of Algorithm 1 is ultimately limited only by the accuracy to which $R(X_j)$ and the sum $X_j + N_j$ are calculated. Of the many methods for solving Riccati equations, Algorithm 1 usually squeezes out the maximum possible accuracy [2], [16], [17].

Algorithm 1 is potentially faster (and more accurate) than the widely used Schur vector method [20]. The break-even point is between six and eight iterations [9] (assuming that a Bartels–Stewart-like algorithm [3], [11] is used to solve the Lyapunov equation).

Although Algorithm 1 ultimately converges quadratically, rapid convergence occurs only in a neighborhood of X^*. Automatic stabilizing procedures like those proposed in [1], [27], and [28] may give choices of X_0 that lie far from the solution X^*. Sometimes the first Newton step N_0 is disastrously large and many iterations are needed to find the region of rapid convergence [16], [17]. If the Lyapunov equation is ill-conditioned it may be difficult to compute an accurate Newton step, and the exact-arithmetic convergence theory breaks down. (This signals an ill-conditioned algebraic Riccati equation [7].) Sometimes rounding errors or a poor X_0 cause Newton's method to converge to a nonstabilizing solution. For these reasons, Newton's method is often limited to defect correction or iterative refinement of an approximate solution obtained by a more robust method.

Example 1: This example is contrived to demonstrate a disastrous first step. (A similar example appears in [16] and [17].) For δ with $0 < \delta < 1$, let $A = S = 0, E = C = B = R = I_2$, and $Q = \mathrm{diag}(1, \sqrt{\delta})$. The stabilizing solution is $X^* = \mathrm{diag}(1, \delta^{1/4})$. Choosing $X_0 = \mathrm{diag}(1, \delta)$, we obtain $\|X^* - X_0\|_F \approx \sqrt{\delta}\|X^*\|_F$, but $\|N_0\|_F \approx 0.5\infty^{-(1/2)}$. For $\delta = 10^{-8}, \|R(X_0)\|_F \approx 10^{-4}$ and $\|R(X_1)\|_F \approx 10^7$. Newton's method then takes 20 iterations to reduce $\|R(X_j)\|_F$ back down to 10^{-4} where it reaches the region of quadratic convergence.

From the point of view of optimization theory, the Newton step gives a search direction along which $\|R(X_j)\|_F$ may be (at least approximately) minimized. The disastrous first step is a step in the search direction that is too long. The several subsequent steps that make limited progress are too short.

In this paper we show how to minimize $\|R(X)\|_F$ along the search direction at little additional cost. This avoids a disastrously large first step, accelerates convergence when Newton steps are too small or too long, and restores some robustness to Newton's method. The idea is to choose $t_j > 0$ to minimize $\|R(X_{j+1})\|_F = \|R(X_j + t_j N_j)\|_F$, i.e., to use an *Exact Line Search* along the Newton direction. Line searches along conjugate gradient directions were used in [10] and [12] to solve (1). Line searches were also used in the Fletcher–Powell/Davidon's method proposed in [21]. Section II shows that the extra cost of doing an Exact Line Search is little more than the cost of calculating the Newton step N_j in Algorithm 1. In Section III we prove that the Exact Line Search along the Newton direction converges quadratically to the stabilizing solution, if the starting guess X_0 is stabilizing. Numerical examples in Section IV demonstrate that step-size control often saves enough iterations to be competitive with the Schur vector method. Some final remarks and conclusions appear in Section V.

II. Step-Size Control by Exact Line Search

Line searches are a well-understood technique in optimization [8]. The approach is to replace Step 2.3 in Algorithm 1 by $X_{j+1} = X_j + t_j N_j$, where t_j is a real scalar "step length" in the direction of N_j. The step length is chosen to minimize or approximately minimize an objective function which, in our case, is $\|R(X_j + t_j N_j)\|_F^2$. The line search is said to be *exact* if t_j is an exact (as opposed to approximate) minimizer.

From (1), we obtain

$$\begin{aligned} R(X_j + tN_j) &= R(X_j) + t((A - BK_j)^T N_j E + E^T N_j (A - BK_j)) \\ &\quad - t^2 E^T N_j B R^{-1} B^T N_j E. \end{aligned} \tag{2}$$

If $V_j = E^T N_j B R^{-1} B^T N_j E$ and N_j is as in Step 2.2 of Algorithm 1, then

$$R(X_j + tN_j) = (1 - t)R(X_j) - t^2 V_j. \tag{3}$$

So, finding t_j to minimize $\|R(X_{j+1})\|_F$ is equivalent to minimizing the quartic polynomial

$$\begin{aligned} f_j(t) &= \mathrm{trace}(R(X_j + tN_j)^2) \\ &= \alpha_j (1-t)^2 - 2\beta_j (1-t)t^2 + \gamma_j t^4 \end{aligned} \tag{4}$$

where $\alpha_j = \mathrm{trace}(R(X_j)^2), \beta_j = \mathrm{trace}(R(X_j)V_j)$, and $\gamma_j = \mathrm{trace}(V_j^2)$. If $\gamma_j \neq 0$, then $f_j(t)$ has at most two local minima, one of which is the global minimum. If $\gamma_j = 0$, then $f_j(t)$ attains its global minimum value (zero) at $t_j = 1$. Differentiating f_j and using (3), we obtain

$$\begin{aligned} f_j'(t) &= -2 \cdot \mathrm{trace}((R(X_j) + 2tV_j)R(X_j + tN_j)) \\ &= -2 \cdot \mathrm{trace}((R(X_j) + 2tV_j)((1-t)R(X_j) - t^2 V_j)). \end{aligned} \tag{5}$$

Remark 1: There exists a local minimum of f_j at some value of $t_j \in [0, 2]$, since $f_j'(0) = -2 \cdot \mathrm{trace}(R(X_j)^2) \leq 0$, and $f_j'(2) = 2 \cdot \mathrm{trace}((R(X_j) + 4V_j)^2) \geq 0$. If $R(X_j) \neq 0$, i.e., if X_j is not a solution of (1), then $f_j'(0) < 0$ and the Newton step is a descent direction of $\|R(X_j + tN_j)\|_F$. It follows that for the minimizing $t_j \in [0, 2]$, we have $\|R(X_j + t_j N_j)\|_F \leq \|R(X_j)\|_F$ and $\|R(X_j + t_j N_j)\|_F = \|R(X_j)\|_F$ if and only if $R(X_j) = 0$.

Remark 1 suggests that we modify Algorithm 1 as follows.

Algorithm 2 (Exact Line Search)

```
1.  Choose some initial starting guess $X_0 = X_0^T$.
2.  FOR $j = 0, 1, 2, \cdots$
      2.1  $K_j \leftarrow R^{-1}(B^T X_j E + S^T C)$.
      2.2  Solve for $N_j$ in the Lyapunov equation
             $(A - BK_j)^T N_j E + E^T N_j (A - BK_j)$
             $= -R(X_j)$.
      2.3  $V_j \leftarrow E^T N_j B R^{-1} B^T N_j E$.
      2.4  Find a local minimizer $t_j \in [0, 2]$ of
             $f_j(t)$ using (4).
      2.5  $X_{j+1} \leftarrow X_j + t_j N_j$.
    END FOR.
```

Remark 2: Algorithm 2 finds the solution of scalar Riccati equations in the first step. Applied to Example 1, one step of the Exact Line Search reduces $\|R(X_j)\|_F$ by as much as 24 steps of Newton's method.

In addition to the work in Algorithm 1, at each iteration, Algorithm 2 must compute the symmetric matrix $V_j = E^T N_j B R^{-1} B^T N_j E$. One way to compute V_j efficiently is as follows. Before starting the iteration, we compute a Cholesky factorization of $R, R = L^T L$ and store the product $\hat{B} = BL^{-1}$. Using $\hat{B}$, we can obtain V_j from $V_j = (E^T N_j \hat{B})(E^T N_j \hat{B})^T$ which requires $5n^2 m + nm$ flops. In case $E = I$, this reduces to $3n^2 m + nm$ flops. In many applications, $m \ll n$, in which case the computation of this matrix is cheap relative to the cost of the Newton step N_j. Computing the coefficients $\alpha_j, \beta_j, \gamma_j$ of f_j and finding the minimizing t_j contributes $3n$ inner products and some scalar operations, which is negligible compared to the $O(n^3)$ flops used by matrix multiplications and Lyapunov equation solutions. Using work estimates from [11] and [13] for solving the Lyapunov equation, we can conclude that for $m = n$, each iteration step of Algorithm 2 does less than 10% more work if $E = I$ and less than 5% more work if $E \neq I$. This comparison becomes more favorable as m decreases relative to n.

III. Convergence

Algorithm 2 casts the nonlinear equation (1) as a nonlinear least squares problem. The convergence theory for this approach is well known and largely satisfactory (for example, see [8, Sec. 6.5]). However, convergence—even convergence to a solution—is not sufficient. Often it is the symmetric stabilizing solution that is required. Other solutions can be transformed back to the stabilizing solution through a process of eigenvalue ordering [9], but it is preferable to get the stabilizing solution in the first place. In this section, we show that under certain assumptions, Algorithm 2 has guaranteed quadratic convergence from a stabilizing starting guess to the stabilizing solution.

By assumption, E is nonsingular, so we may rewrite (1) as

$$R(X) = \tilde{R}(\tilde{X}) = \tilde{F} + \tilde{A}^T \tilde{X} + \tilde{X}\tilde{A} - \tilde{X}\tilde{G}\tilde{X} \tag{6}$$

where

$$\tilde{A} = E^{-1}(A - BR^{-1}S^TC), \qquad \tilde{F} = C^T(Q - SR^{-1}S^T)C$$
$$\tilde{B} = E^{-1}B, \qquad \tilde{G} = \tilde{B}R^{-1}\tilde{B}^T, \qquad \tilde{X} = E^TXE.$$

It is easy to verify that $\tilde{X}_j$ and $\tilde{R}(\tilde{X}_j)$ are the sequences of approximate solutions and residuals produced by Algorithm 2 applied to (6) with starting guess $\tilde{X}_0$. Note that because E is nonsingular, the boundedness, convergence (or lack of it), and rate of convergence of the two sequences X_j and $\tilde{X}_j$ are identical and $\lambda(E, A - BK_j) \subset \mathbb{C}^-$ if and only if $\tilde{A} - \tilde{G}\tilde{X}_j$ is stable. The residual satisfies $\tilde{R}(\tilde{X}_j) = R(X_j)$, and X^* satisfies $R(X^*) = 0$ if and only if $\tilde{X}^* = E^TX^*E$ satisfies $\tilde{R}(\tilde{X}^*) = 0$. Note further that the sequence of step sizes t_j produced by Algorithm 2 is equal in both cases. The coefficient matrix $\tilde{G} = \tilde{B}R^{-1}\tilde{B}^T$ is symmetric positive semidefinite because by assumption, R is symmetric positive definite.

In Remark 1, it was observed that there exists a local minimizer of $\|R(X_j + tN_j)\|_F$ in the interval $[0, 2]$. The following lemma shows that the iterates $\tilde{X}_j + t_j\tilde{N}_j$ are stabilizing if the starting guess $\tilde{X}_0$ is stabilizing, and $t_j \in [0, 2]$. We can thus consider $[0, 2]$ to be the "canonical" search interval.

Lemma 3: If $\tilde{G} \geq 0$ and $\tilde{A} - \tilde{G}\tilde{X}_j$ is stable, then for all $t \in [0, 2], \tilde{A} - \tilde{G}(\tilde{X}_j + t\tilde{N}_j)$ is also stable.

Proof: The Newton Step $\tilde{N}_j$ is determined by

$$\begin{aligned}(\tilde{A} - \tilde{G}\tilde{X}_j)^T(\tilde{X}_j + \tilde{N}_j) + (\tilde{X}_j + \tilde{N}_j)(\tilde{A} - \tilde{G}\tilde{X}_j)\\ = -\tilde{F} - \tilde{X}_j\tilde{G}\tilde{X}_j.\end{aligned}$$

Subtracting this from $\tilde{R}(\tilde{X}^*) = 0$ and subtracting $\tilde{X}_j\tilde{G}\tilde{X}^* + \tilde{X}^*\tilde{G}\tilde{X}_j$ on both sides yields

$$\begin{aligned}(\tilde{A} - \tilde{G}\tilde{X}_j)^T\Big(\tilde{X}^* - (\tilde{X}_j + \tilde{N}_j)\Big) + \Big(\tilde{X}^* - (\tilde{X}_j + \tilde{N}_j)\Big)(\tilde{A} - \tilde{G}\tilde{X}_j)\\ = (\tilde{X}^* - \tilde{X}_j)\tilde{G}(\tilde{X}^* - \tilde{X}_j). \qquad (7)\end{aligned}$$

Using a modified version of Lyapunov's theorem [19, p. 447] (7) together with the stability of $\tilde{A} - \tilde{G}\tilde{X}_j$ and $\tilde{G} \geq 0$ implies $\tilde{X}^* - (\tilde{X}_j + \tilde{N}_j) \leq 0$. Rearranging (7), we obtain

$$\begin{aligned}&(\tilde{A} - \tilde{G}(\tilde{X}_j + t\tilde{N}_j))^T(\tilde{X}^* - (\tilde{X}_j + \tilde{N}_j))\\ &\quad + (\tilde{X}^* - (\tilde{X}_j + \tilde{N}_j))(\tilde{A} - \tilde{G}(\tilde{X}_j + t\tilde{N}_j))\\ &= (\tilde{X}^* - (\tilde{X}_j + t\tilde{N}_j))\tilde{G}(\tilde{X}^* - (\tilde{X}_j + t\tilde{N}_j))\\ &\quad + t(2 - t)\tilde{N}_j\tilde{G}\tilde{N}_j =: W. \qquad (8)\end{aligned}$$

Since $t \in [0, 2]$, the right-hand side W in (8) is positive semidefinite.

Now suppose $\tilde{A} - \tilde{G}(\tilde{X}_j + t\tilde{N}_j)$ has an eigenvalue λ with $\operatorname{Re}(\lambda) \geq 0$ and corresponding eigenvector $z \neq 0$, i.e.,

$$(\tilde{A} - \tilde{G}(\tilde{X}_j + t\tilde{G}\tilde{N}_j))z = \lambda z. \qquad (9)$$

Multiply (8) from the left by z^H and from the right by z. Then we obtain

$$2 \cdot \operatorname{Re}(\lambda)z^H(\tilde{X}^* - (\tilde{X}_j + \tilde{N}_j))z = z^HWz. \qquad (10)$$

The left-hand side of (10) is nonpositive since $\tilde{X}^* - (\tilde{X}_j + \tilde{N}_j) \leq 0$ and $\operatorname{Re}(\lambda) \geq 0$. As W is positive semidefinite, the right-hand side of (10) is nonnegative and it follows that $z^HWz = 0$. Thus

$$z^H(\tilde{X}^* - (\tilde{X}_j + t\tilde{N}_j))\tilde{G}(\tilde{X}^* - (\tilde{X}_j + t\tilde{N}_j))z = 0$$

and since $\tilde{G} \geq 0$, this implies $\tilde{G}(\tilde{X}^* - (\tilde{X}_j + t\tilde{N}_j))z = 0$, or, equivalently, $\tilde{G}\tilde{X}^*z = \tilde{G}(\tilde{X}_j + t\tilde{N}_j)z$. From (9) we therefore obtain $\lambda z = (\tilde{A} - \tilde{G}(\tilde{X}_j + t\tilde{N}_j))z = (\tilde{A} - \tilde{G}\tilde{X}^*)z$. Hence, λ is an eigenvalue of $\tilde{A} - \tilde{G}\tilde{X}^*$ which contradicts the stability of $\tilde{A} - \tilde{G}\tilde{X}^*$. □

The *Lyapunov operator* corresponding to the Lyapunov equations in Step 2.2 of Algorithm 2 is defined by $\tilde{\Omega}_j(Z) = (\tilde{A} - \tilde{G}\tilde{X}_j)^TZ + Z(\tilde{A} - \tilde{G}\tilde{X}_j)$ for $Z \in \mathbb{R}^{n \times n}$ and $j = 1, 2, \cdots$. A corollary of Lemma 3 is that with a stabilizing starting guess, Algorithm 2 cannot fail due to a singular Lyapunov operator.

Corollary 4: If $\tilde{X}_0$ is stabilizing, and Algorithm 2 is applied to (6), then the Lyapunov operator $\tilde{\Omega}_j$ in Step 2.2 is nonsingular for all j, and the sequence of approximate solutions $\tilde{X}_j$ is well defined.

We will also need the following technical characterization of controllability.

Lemma 5: Suppose that $A \in \mathbb{R}^{n \times n}, B \in \mathbb{R}^{n \times m}, R \in \mathbb{R}^{m \times m}$, and R is symmetric positive definite. The pair (A, B) is controllable if and only if the only matrix $Y = Y^H$ satisfying $YBR^{-1}B^TY = 0$ and $A^TY + YA \geq 0$ is $Y = 0$.

Proof: We will prove the contrapositive of the statement in Lemma 5: the pair (A, B) is uncontrollable if and only if there exists $Y = Y^H \neq 0$ such that $YBR^{-1}B^TY = 0$ and $A^TY + YA \geq 0$.

If (A, B) is uncontrollable, then there exists a left eigenvector w of A that lies in the left null space of B. Let λ_r be the real part of the corresponding eigenvalue of A. If $Y = \operatorname{sign}(\lambda_r)ww^H$, then $YBR^{-1}B^TY = ww^HBR^{-1}B^Tww^H = 0$ and $A^TY + YA = 2|\lambda_r|Y = 2|\lambda_r|ww^H$ is positive semidefinite.

For the converse, assume that there exists a symmetric matrix $Y \neq 0$ such that $A^TY + YA \geq 0$ and $YBR^{-1}B^TY = 0$. We will show that (A, B) is uncontrollable by constructing a left eigenvector of A belonging to the left null space of B.

By choosing an appropriate orthonormal basis, we may arrange that A, Y, and $BR^{-1}B^T$ take the form

$$BR^{-1}B^T = \begin{bmatrix} G_{11} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad Y = \begin{bmatrix} 0 & 0 & 0 \\ 0 & Y_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$
$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$

where $G_{11} \in \mathbb{R}^{h \times h}$ and $Y_{22} \in \mathbb{R}^{k \times k}$ are nonsingular. The assumption $Y \neq 0$ implies that $k > 0$. However, it is possible that either $h = 0$ or $n - h - k = 0$, in which case the corresponding rows and columns do not appear. In this basis, $A^TY + YA$ takes the form

$$A^TY + YA = \begin{bmatrix} 0 & A_{21}^TY_{22} & 0 \\ Y_{22}A_{21} & A_{22}^TY_{22} + Y_{22}A_{22} & Y_{22}A_{23} \\ 0 & A_{23}^TY_{22} & 0 \end{bmatrix}.$$

By hypothesis, this matrix is positive semidefinite, so $Y_{22}A_{21} = 0$ and $Y_{22}A_{23} = 0$. It follows from the nonsingularity of Y_{22} that $A_{21} = 0$ and $A_{23} = 0$.

Let $w_2 \in \mathbb{C}^k$ be a left eigenvector of A_{22}. Define $w \in \mathbb{C}^n$ as $w = [w_1, w_2, w_3]$ where $w_1 = 0 \in \mathbb{C}^h$ and $w_3 = 0 \in \mathbb{R}^{n-h-k}$. The vector w is a left eigenvector of A belonging to the left null space of B. □

As seen in Remark 1, the sequence of residuals $\tilde{R}(\tilde{X}_j)$ produced by Algorithm 2 is monotonically decreasing and, in particular, bounded. The next lemma shows that boundedness carries over to the iterates $\tilde{X}_j$ also.

Lemma 6: Suppose that $\tilde{X}_j, j = 1, 2, 3, \cdots$ is a sequence of symmetric n-by-n matrices such that $\tilde{R}(\tilde{X}_j)$ is bounded. If $(\tilde{A}, \tilde{B})$ is a controllable pair, then the sequence $\tilde{X}_j$ is bounded.

Proof: We will prove the contrapositive: if $\tilde{X}_j$ is unbounded, then $(\tilde{A}, \tilde{B})$ is not controllable. Without loss of generality we may assume that $\lim_{j\to\infty} \|\tilde{X}_j\|_F = \infty$. (If not, we may consider a subsequence for which this assertion holds.) Define $\xi_j = \|\tilde{X}_j\|_F$ and $\tilde{Y}_j = \tilde{X}_j/\xi_j$. The $\tilde{Y}_j$'s are bounded, so there is a convergent subsequence which we may assume without loss of generality is the whole sequence. Let $\tilde{Y} = \lim_{j\to\infty} \tilde{Y}_j$. Note that $\tilde{Y} \neq 0$. From definition (6), we have

$$\frac{1}{\xi_j}(\tilde{F} - \tilde{R}(\tilde{X}_j)) + \tilde{A}^T\tilde{Y}_j + \tilde{Y}_j\tilde{A} = \xi_j\tilde{Y}_j\tilde{B}R^{-1}\tilde{B}^T\tilde{Y}_j. \qquad (11)$$

Because $\tilde{R}(\tilde{X}_j)$ is bounded, the first term on the left-hand side of (11) tends to zero as $j \to \infty$. The second term approaches the finite limit $\tilde{A}^T\tilde{Y} + \tilde{Y}\tilde{A}$. From the right-hand side, it is clear that this is a limit of positive semidefinite matrices and hence is positive semidefinite. Dividing (11) by ξ_j and letting $j \to \infty$ gives $\tilde{Y}\tilde{B}R^{-1}\tilde{B}^T\tilde{Y} = 0$. It follows from Lemma 5 that $(\tilde{A}, \tilde{B})$ is uncontrollable. □

We are now ready to prove that Algorithm 2 reduces the residual $\tilde{R}(\tilde{X}_j)$ [and hence $R(X_j)$] asymptotically to zero if the computed step sizes are bounded away from zero.

Theorem 7: If $(\tilde{A}, \tilde{B})$ is a controllable pair, and the sequence of step sizes t_j computed by Algorithm 2 is uniformly bounded from below by $t_L > 0$, then the residual norms $\|\tilde{R}(\tilde{X}_j)\|_F$ decrease monotonically to zero and cluster points of the sequence $\tilde{X}_j$ are solutions of the algebraic Riccati equation (1).

Proof: Lemma 6 shows that the sequence of approximate roots $\tilde{X}_j$ is bounded. Consequently, the steps $t_j\tilde{N}_j$ are also bounded. Here $\tilde{N}_j = E^T N_j E$, and t_j is the step size computed by minimizing $f_j(t) = \|\tilde{R}(\tilde{X}_j + t\tilde{N}_j)\|_F^2$. The $t_j \in [0, 2]$ also form a bounded sequence, and since we assumed $0 < t_L \le t_j$ for all j, the $\tilde{N}_j$'s are bounded, too. Select a subsequence $\tilde{X}_{j_k}$ of the $\tilde{X}_j$'s such that $\hat{X} = \lim_{k\to\infty} \tilde{X}_{j_k}, \hat{t} = \lim_{k\to\infty} t_{j_k}$, and $\hat{N} = \lim_{k\to\infty} \tilde{N}_{j_k}$ exist. Note that the residual norms $\|\tilde{R}(\tilde{X}_j)\|_F$ are monotonically decreasing, so they approach a limit and hence

$$\|\tilde{R}(\hat{X} + \hat{t}\hat{N})\|_F = \|\tilde{R}(\hat{X})\|_F. \tag{12}$$

Therefore, the coefficients $\alpha_{j_k}, \beta_{j_k}$, and γ_{j_k} in (4) approach limits and the minimum value of the polynomial $\hat{f}(t) = \|\tilde{R}(\hat{X} + t\hat{N})\|_F^2$ is the limit of the minimum values of the f_{j_k}'s, i.e., we have $\lim_{k\to\infty} f_{j_k}(t_{j_k}) = \hat{f}(\hat{t}) \le \hat{f}(0)$. However, using (12), we obtain $\hat{f}(0) = \|\tilde{R}(\hat{X})\|_F = \|\tilde{R}(\hat{X} + \hat{t}\hat{N})\|_F = \hat{f}(\hat{t})$. It follows that $\hat{f}'(0) = 0$. But as observed in Remark 1, $\hat{f}'(0) = -2\|\tilde{R}(\hat{X})\|_F^2$. Thus, $\tilde{R}(\hat{X}) = 0$. □

In summary, we have the following convergence result for Newton's method with Exact Line Search.

Theorem 8: Suppose $(\tilde{A}, \tilde{B})$ defines a controllable matrix pair. If Algorithm 2 is applied to the algebraic Riccati equation (6) with a stabilizing starting guess $\tilde{X}_0$ and the step sizes t_j are bounded away from zero, then $\tilde{X}^* = \lim_{j\to\infty} \tilde{X}_j$ exists and is the stabilizing solution of (6).

Remark 9: The above convergence result relies on the fact that $t_j \ge t_L$ for all j and a given constant $t_L > 0$. We can modify Algorithm 2 such that the step size is set to one if t_j drops below a prescribed (small) constant. By (7) it is clear that the so-defined new iterate $X_{j+1} = X_j + N_j$ satisfies $X^* \le X_{j+1}$. We can now apply the Newton iteration (Algorithm 1) with the "starting guess" X_{j+1} and use the standard convergence theory for Newton's method [18], [23], [26] to show that iterates produced by this hybrid algorithm converge to the stabilizing solution of (1).

In our numerical experiments, very small step sizes occurred 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 the Exact Line Search can be expected to improve the accuracy of the approximate solution of (1) any further.

Algorithm 2 inherits its quadratic convergence from Newton's method [24]. Suppose that $\tilde{X}_j$ is within the region of quadratic convergence of Newton's method. In this case [23]

$$\tilde{N}_j = \tilde{X}^* - \tilde{X}_j + O(\|\tilde{X}^* - \tilde{X}_j\|_F^2) \tag{13}$$

and

$$\|\tilde{R}(\tilde{X}_j + \tilde{N}_j)\|_F = O(\|\tilde{X}^* - \tilde{X}_j\|_F^2). \tag{14}$$

Let $\tilde{\Omega}(Z) = (\tilde{A} - \tilde{G}\tilde{X}^*)^T Z + Z(\tilde{A} - \tilde{G}\tilde{X}^*), Z \in \mathbb{R}^{n\times n}$. Then the residual produced by Algorithm 2 satisfies

$$\begin{aligned}\tilde{R}(\tilde{X}_j + t_j\tilde{N}_j) &= \tilde{R}(\tilde{X}^* + (\tilde{X}_j + \tilde{N}_j - \tilde{X}^*) + (t_j - 1)\tilde{N}_j)\\ &= \tilde{\Omega}(\tilde{X}_j + \tilde{N}_j - \tilde{X}^*) + (t_j - 1)\tilde{\Omega}(\tilde{N}_j)\\ &\quad - ((\tilde{X}_j + \tilde{N}_j - \tilde{X}^*) + (t_j - 1)\tilde{N}_j)\\ &\quad \cdot \tilde{G}((\tilde{X}_j + \tilde{N}_j - \tilde{X}^*) + (t_j - 1)\tilde{N}_j).\end{aligned}$$

Taking norms, using (13), and recognizing that $|t_j - 1| \le 1$ gives

$$\begin{aligned}\|\tilde{R}(\tilde{X}_j + t_j\tilde{N}_j)\|_F \le{}& 2|t_j - 1|\|\tilde{X}^* - \tilde{X}_j\|_F\|\tilde{A} - \tilde{G}\tilde{X}^*\|_F\\ &+ O(\|\tilde{X}_j - \tilde{X}^*\|_F^2).\end{aligned} \tag{15}$$

Recall that $t_j \in [0, 2]$ is chosen to minimize $\|\tilde{R}(\tilde{X}_j + t\tilde{N}_j)\|_F$, so (14) implies

$$\|\tilde{R}(\tilde{X}_j + t_j\tilde{N}_j)\|_F \le \|\tilde{R}(\tilde{X}_j + \tilde{N}_j)\|_F = O(\|\tilde{X}^* - \tilde{X}_j\|_F^2). \tag{16}$$

It follows from (15) and (16) that $|t_j - 1| = O(\|\tilde{X}^* - \tilde{X}_j\|_F)$ which implies that $t_j \approx 1$ in the neighborhood of the stabilizing solution. Hence

$$\begin{aligned}\|\tilde{X}^* - \tilde{X}_{j+1}\|_F &\le \|\tilde{X}^* - (\tilde{X}_j + \tilde{N}_j)\|_F + |1 - t_j|\|\tilde{N}_j\|_F\\ &= O(\|\tilde{X}^* - \tilde{X}_j\|_F^2)\end{aligned}$$

which proves the quadratic convergence of Algorithm 2.

The following theorem summarizes the convergence theory.

Theorem 10: If $(E^{-1}A, E^{-1}B)$ is controllable and $X_0 = X_0^T$ is stabilizing in the sense that $\lambda(E, A - BK_0) \subset \mathbb{C}^-$, then the sequence of approximate solutions X_j produced by the modified Algorithm described in Remark 9 converges quadratically to the stabilizing solution X^*, at each step, $\lambda(E, A - BK_j) \subset \mathbb{C}^-$, and the residual norms $\|R(X_j)\|_F$ converge monotonically and quadratically to zero.

The theorem is more general than the one stated in [23] since it does not require X_0 to be positive semidefinite. In contrast to Newton's method, the iterates X_j are not necessarily positive semidefinite and they do not necessarily converge monotonically (in terms of definiteness). On the other hand, the theorem needs the strong hypothesis of controllability. Numerical experiments suggest that this can be weakened to stabilizability, but as of this writing we do not have a proof.

IV. Numerical Examples

Newton's Method (Algorithm 1) and the Exact Line Search (Algorithm 2) were implemented as MATLAB [22] functions. We did not use the hybrid algorithm proposed in Remark 9. All computations were done under MATLAB Version 4.2a [22] on Hewlett Packard Apollo series 700 computers under IEEE double precision and machine precision $\varepsilon \approx 2.2204 \cdot 10^{-16}$. We compared the algorithms on the examples in the benchmark collection of CARE's [6], several randomly generated examples, and some contrived examples [5].

We observed the following.

1) In examples where Newton's method is much more expensive than the Schur vector method (Examples 2 and 3), the Exact Line Search was competitive and sometimes faster than the Schur vector method [20].
2) When used as defect correction or iterative refinement method, it sometimes even improves on Newton's method (see Example 4).

In most cases, when used as defect correction method, the computed step sizes are $t_j \approx 1$, so the Exact Line Search behaves like Newton's method. This is expected from the discussion of quadratic convergence in Section III. For more detailed numerical studies and other examples see [5].

Exact Line Search, and the Schur vector method as proposed in [4], [5], and [20], have been implemented on vector and parallel

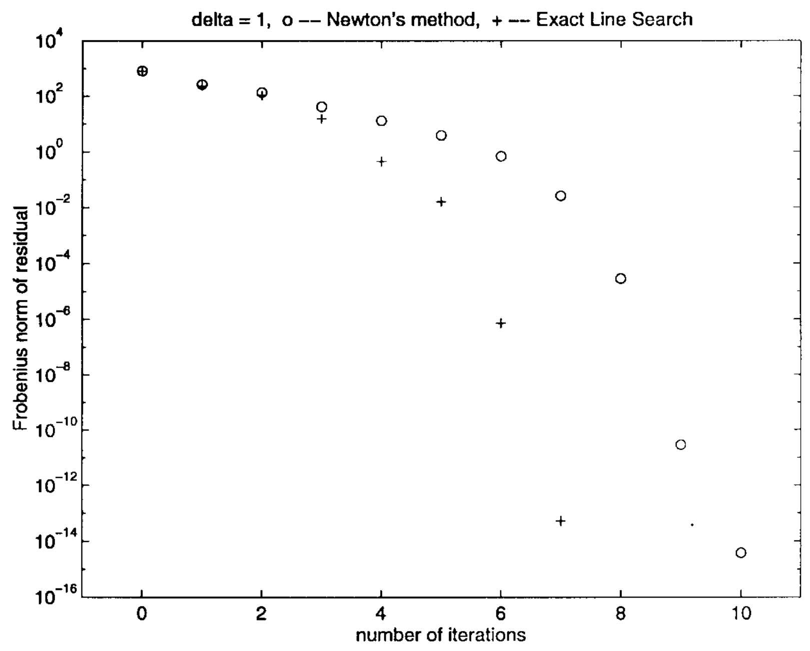

Fig. 1. Example 2, $\delta = 1$.

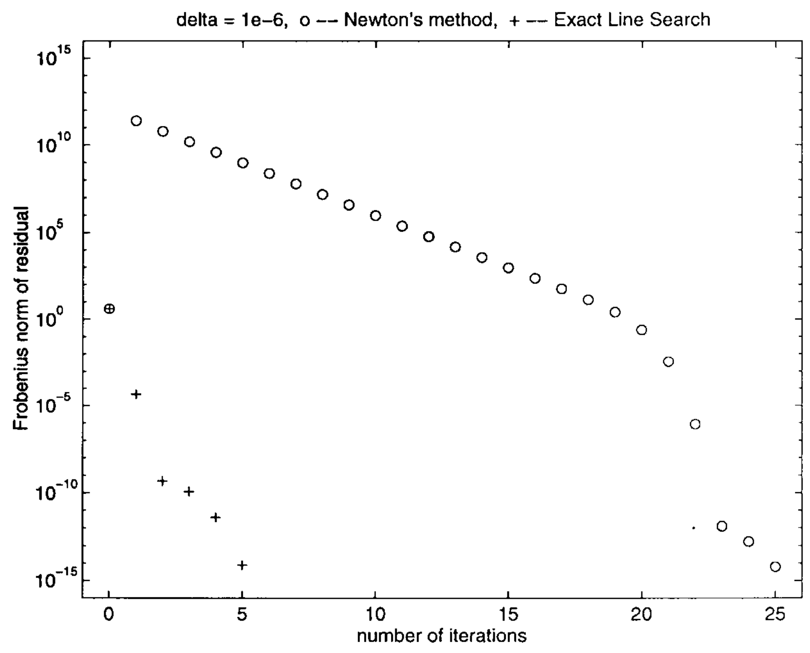

Fig. 2. Example 2, $\delta = 10^{-6}$.

computers using block-oriented algorithms [25]. The results reported there suggest that on computers with advanced architectures, the Exact Line Search and even Newton's method compare favorably to the Schur vector method.

Example 2 [6, ex. 14], [2, ex. 2]: Here, A depends upon a parameter δ. If $\delta \to 0$, the system approaches one which is unstabilizable and a conjugate complex pair of closed-loop eigenvalues approaches the imaginary axis. The system matrices are given by

$$A = \begin{bmatrix} -\delta & 1 & 0 & 0 \\ -1 & -\delta & 0 & 0 \\ 0 & 0 & \delta & 1 \\ 0 & 0 & -1 & \delta \end{bmatrix}, \qquad B = C^T = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$E = I_4, \qquad Q = R = [1], \qquad S = 0.$$

Stabilizing starting guesses X_0 were generated by the method described in [1], [14], and [27]. Figs. 1 and 2 show the behavior of the algorithms for $\delta = 1$ and $\delta = 10^{-6}$. The initial slow convergence behavior of Newton's method grows worse as $\delta \to 0$, but the Exact Line Search neatly avoids the problem. As opposed to Newton's method, the Exact Line Search needs no more than six to eight iterations and is therefore competitive with or even cheaper than the Schur vector method.

Example 3 [6, ex. 15], [20, ex. 4]: This example is frequently used to test CARE solution methods. It is a position and velocity control model of a string of N high-speed vehicles. We have $n = 2N - 1, m = N$, and $p = N - 1$. The nonzero entries in the transition matrix A are $a_{2N-1,2N-1} = 1$ and for $i = 1, \cdots, N - 1, a_{2i-1,2i-1} = a_{2i,2i+1} = -1$, and $a_{2i,2i-1} = 1$.

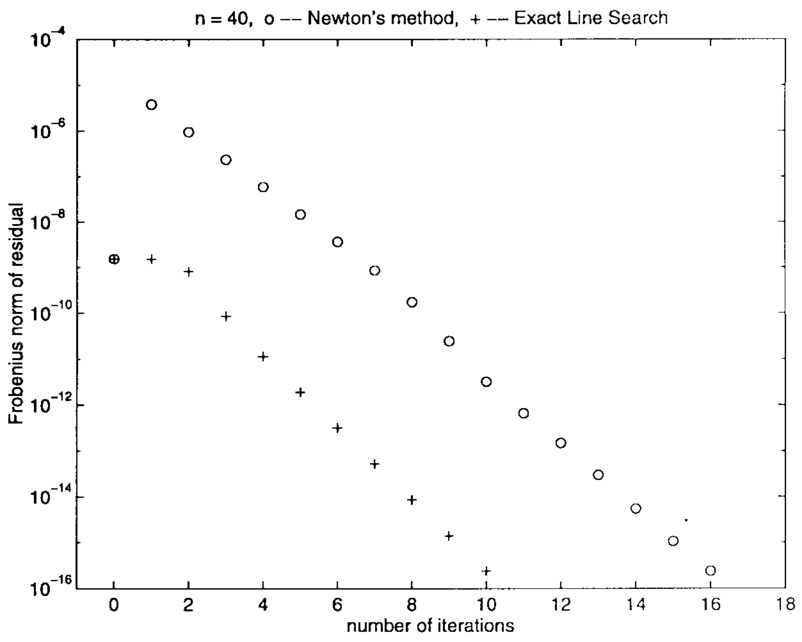

Fig. 3. Example 4, $n = 40$.

TABLE I
EXAMPLE 3

	Newton's method			Exact Line Search		
n	it.	$\|\|R(\hat{X})\|\|_F$	$\|\|R(\hat{X})\|\|_F/\|\|\hat{X}\|\|_F$	it.	$\|\|R(\hat{X})\|\|_F$	$\|\|R(\hat{X})\|\|_F/\|\|\hat{X}\|\|_F$
9	5	$1.2 \cdot 10^{-13}$	$6.1 \cdot 10^{-15}$	5	$5.6 \cdot 10^{-15}$	$2.9 \cdot 10^{-16}$
49	7	$6.9 \cdot 10^{-14}$	$1.1 \cdot 10^{-15}$	6	$2.2 \cdot 10^{-14}$	$3.6 \cdot 10^{-16}$
99	8	$8.2 \cdot 10^{-14}$	$8.1 \cdot 10^{-16}$	6	$3.8 \cdot 10^{-14}$	$3.8 \cdot 10^{-16}$
199	9	$1.1 \cdot 10^{-13}$	$6.5 \cdot 10^{-16}$	6	$8.0 \cdot 10^{-14}$	$4.6 \cdot 10^{-16}$

The other system matrices are $E = R = C = I_n, S = 0, B = \operatorname{diag}(1, 0, 1, 0, \cdots, 1, 0, 1)$, and $Q = \operatorname{diag}(0, 10, 0, 10, \cdots, 0, 10, 0)$. Stabilizing starting guesses X_0 were generated by the method described in [1], [14], and [27]. Table I shows the number of iterations and the Frobenius norm of the last absolute and relative residual for some values of n. ($\hat{X}$ denotes the computed approximation to X^*.) Exact Line Search is somewhat faster than the Schur vector method while Newton's method slows down as n increases. In agreement with our observations, timings on vector and parallel computers in [25] indicate that the Exact Line Search requires about two-thirds of the time of the Schur vector method.

Example 4: One of the situations in which defect correction or iterative refinement [16], [17] has the most to offer is when the Riccati equation is ill-conditioned. Rounding errors make it unlikely that any Riccati solver will produce much accuracy, but with its excellent structure-preserving rounding error properties, Newton's method is likely to squeeze out as much accuracy as possible [2], [16], [17]. This example is contrived to be highly ill-conditioned. Let $e \in \mathbb{R}^n$ denote the vector of ones, and $n = m = p$, then the CARE (1) is given by

$$E = R = I, \qquad A = S = 0, \qquad B = 10^3 I$$

$$C = I - \frac{2}{n} ee^T, \qquad Q = \operatorname{diag}\left(\frac{1}{9^1}, \frac{1}{9^2}, \frac{1}{9^2}, \frac{1}{9^3}, \frac{1}{9^3}, \cdots\right).$$

The exact stabilizing solution is given by $X^* = 10^{-3} C^T QC$.

We obtained the starting guess as $X_0 = (X + X^T)/2$ where X is the "solution" of (1) computed by the Schur vector method as discussed in [2]. Observe in Figs. 3 and 4 that Newton's method increases the initial residual norm by several orders of magnitude. The graph of relative errors closely matches the graph of residuals.

Using the CARE condition number K^+ proposed in [7] and [15] we obtain $K^+ \approx 1.8 \cdot 10^9$ for $n = 40$ and $K^+ \approx 4.2 \cdot 10^{11}$ for $n = 50$. Rounding errors made while forming $C^T QC$ are sufficient to change the smaller eigenvalues and corresponding invariant subspaces of the solution X^* and the closed-loop system $A - BR^{-1}B^T X^*$ by over 100%. The closed-loop poles are so close to the imaginary axis that the symmetrized Schur vector solution for $n = 50$ did not appear to be stabilizing as it should have been; one of the smaller eigenvalues of $A - BR^{-1}B^T X_0$ computed by MATLAB was of the wrong sign. The Exact Line Search preserves inertia, so for $n = 50$ it did not converge to a stabilizing solution either, while for Newton's method two more eigenvalues cross the imaginary axis.

Notice in Fig. 3 that for $n = 40$, refining the Schur vector solution reduced the residual down to machine precision. In both cases, the Exact Line Search required about two-thirds of the computational cost of Newton's method to reach the limiting accuracy. This shows that also for defect correction, the Exact Line Search does in some cases compare favorably to Newton's method. In both examples, the first Newton step is a disaster.

V. CONCLUSIONS

We have studied an Exact Line Search method based on Newton's method for solving (generalized) continuous–time algebraic Riccati equations. It avoids Newton's method's problem with disastrously large first steps, and it accelerates convergence when Newton steps are too small or too long. Numerical experiments verify that it

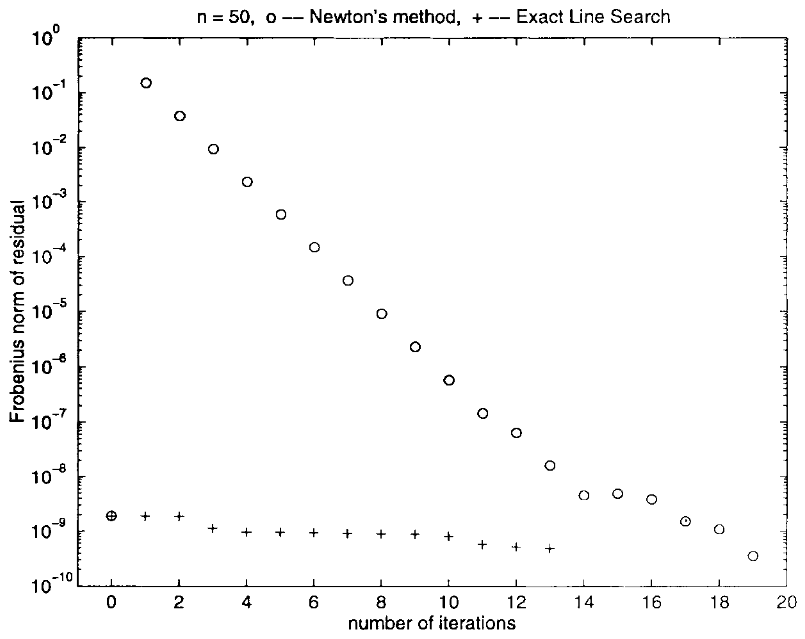

Fig. 4. Example 4, $n = 50$.

sometimes significantly reduces the number of iterations. Theoretical convergence properties are similar to Newton's method. Used as a defect correction method or for iterative refinement, it has the ability to obtain high accuracy. The Exact Line Search adds less than 10% to the cost of a Newton iteration, i.e., the additional work to perform the line search is small relative to the work needed to calculate the Newton step.

A Fortran 77 implementation of the Exact Line Search method will complement Newton's method in a forthcoming release of the Subroutine Library in Control and Systems Theory (SLICOT) [29].

Acknowledgment

The authors wish to thank S. G. Nash for his helpful comments on the proof for quadratic convergence of the Exact Line Search method and Alan J. Laub for providing the MATLAB function care implementing the Schur vector method as discussed in [2].

References

[1] E. S. Armstrong, "An extension of Bass' algorithm for stabilizing linear continuous constant systems," *IEEE Trans. Automat. Contr.,* vol. AC-20, pp. 153–154, 1975.
[2] W. Arnold, III and A. Laub, "Generalized eigenproblem algorithms and software for algebraic Riccati equations," *Proc. IEEE,* vol. 72, pp. 1746–1754, 1984.
[3] R. Bartels and G. Stewart, "Solution of the matrix equation $AX + XB = C$: Algorithm 432," *Comm. ACM,* vol. 15, pp. 820–826, 1972.
[4] P. Benner and R. Byers, "Step size control for Newton's method applied to algebraic Riccati equations," in *Proc. Fifth SIAM Conf. Appl. Lin. Alg.,* Snowbird, UT, J. Lewis, Ed. Philadelphia, PA: SIAM, 1994, pp. 177–181.
[5] ——, "Newton's method with exact line search for solving the algebraic Riccati equation," Fakultät für Mathematik, TU Chemnitz-Zwickau, 09107 Chemnitz, FRG, Tech. Rep. SPC 95–24, 1995; available as SPC95_24.ps by anonymous ftp from ftp.tu-chemnitz.de, directory/pub/Local/mathematik/Benner.
[6] P. Benner, A. Laub, and V. Mehrmann, "A collection of benchmark examples for the numerical solution of algebraic Riccati equations I: Continuous-time case," Fakultät für Mathematik, TU Chemnitz-Zwickau, 09107 Chemnitz, FRG, Tech. Rep. SPC 95-22, 1995.
[7] R. Byers, "Numerical condition of the algebraic Riccati equation," *Contemporary Math.,* vol. 47, pp. 35–49, 1985.
[8] J. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Englewood Cliffs, NJ: Prentice-Hall, 1983.
[9] L. Dieci, "Some numerical considerations and Newton's method revisited for solving algebraic Riccati equations," *IEEE Trans. Automat. Contr.,* vol. 36, pp. 608–616, 1991.
[10] L. Dieci and R. D. Russell, "Some iterative methods for solving algebraic Riccati equations," Georgia Inst. Technol., School Math., Atlanta, GA, Tech. Rep. 091389-001, 1989.
[11] J. Gardiner, A. Laub, J. Amato, and C. Moler, "Solution of the Sylvester matrix equation $AXB + CXD = E$," *ACM Trans. Math. Software,* vol. 18, pp. 223–231, 1992.
[12] A. Ghavimi, C. Kenney, and A. Laub, "Local convergence analysis of conjugate gradient methods for solving algebraic Riccati equations," *IEEE Trans. Automat. Contr.,* vol. 37, pp. 1062–1067, 1992.
[13] G. Golub and C. Van Loan, *Matrix Computations,* 2nd ed. Baltimore, MD: Johns Hopkins Univ. Press, 1989.
[14] S. Hammarling, "Newton's method for solving the algebraic Riccati equation," Nat. Phys. Lab., Teddington, Middlesex TW11 0LW, U.K., NPL Rep. DITC 12/82, 1982.
[15] C. Kenney and G. Hewer, "The sensitivity of the algebraic and differential Riccati equations," *SIAM J. Contr. Optim.,* vol. 28, pp. 50–69, 1990.
[16] C. Kenney, A. Laub, and M. Wette, "A stability-enhancing scaling procedure for Schur-Riccati solvers," *Syst. Contr. Lett.,* vol. 12, pp. 241–250, 1989.
[17] ——, "Error bounds for Newton refinement of solutions to algebraic Riccati equations," *Math. Contr., Signals, Syst.,* vol. 3, pp. 211–224, 1990.
[18] D. Kleinman, "On an iterative technique for Riccati equation computations," *IEEE Trans. Automat. Contr.,* vol. AC-13, pp. 114–115, 1968.
[19] P. Lancaster and M. Tismenetsky, *The Theory of Matrices,* 2nd ed. Orlando, FL: Academic, 1985.
[20] A. Laub, "A Schur method for solving algebraic Riccati equations," *IEEE Trans. Automat. Contr.,* vol. AC-24, pp. 913–921, 1979; see also *Proc. 1978 Conf. Decision Contr.,* pp. 60–65.
[21] F. Man, "The Davidon method of solution of the algebraic matrix Riccati equation," *Int. J. Contr.,* vol. 10, pp. 713–719, 1969.
[22] *MATLAB.* Natick, MA: The MathWorks, 1984–1994.
[23] V. Mehrmann, "The autonomous linear quadratic control problem, theory and numerical solution," no. 163 in *Lecture Notes in Control and Information Sciences.* Heidelberg: Springer-Verlag, July 1991.
[24] S. Nash, private communication, 1994.
[25] E. Quintana and V. Hernández, *Parallel Algorithms for Solving Algebraic Riccati Equations.* 1996.
[26] N. Sandell, "On Newton's method for Riccati equation solution," *IEEE Trans. Automat. Contr.,* vol. AC-19, pp. 254–255, 1974.
[27] V. Sima, "An efficient Schur method to solve the stabilization problem," *IEEE Trans. Automat. Contr.,* vol. AC-26, pp. 724–725, 1981.
[28] A. Varga, "On stabilization methods of descriptor systems," *Syst. Contr. Lett.,* vol. 24, pp. 133–138, 1995.
[29] "Working group on software (WGS)," *SLICOT Mark 3,* to be published; http://www.win.tue.nl/wgs/slicot.html.