An iterative algorithm is presented for obtaining a positive definite symmetric square root of a positive definite symmetric matrix. This algorithm has application in the implementation of Riccati type equations. The approach presented here has the advantage that apriori upper and lower bounds to the converged answer can be obtained sequentially at any point in the iterative process. These apriori bounds can also be obtained for the Riccati equation using a discrete implementation procedure. Theorems on convergence are proved and examples are worked.
An Algorithm For Computing Matrix Square Roots With Application To Riccati Equation Implementation

D. W. Repperger

Aerospace Medical Research Laboratory
Wright-Patterson Air Force Base, Ohio 45433

Abstract

An iterative algorithm is presented for obtaining a positive definite symmetric square root of a positive definite symmetric matrix. This algorithm has application in the implementation of Riccati type equations. The approach presented here has the advantage that apriori upper and lower bounds to the converged answer can be obtained sequentially at any point in the iterative process. These apriori bounds can also be obtained for the Riccati equation using a discrete implementation procedure. Theorems on convergence are proved and examples are worked.

I. Introduction

The study of optimal control and estimation theory has been influenced in recent years by factorization methods in the analytic expression of the filtering equations [1,2,3]. Computationally the solution of filtering and smoothing equations appears numerically better behaved [4] when the equations are of the square root type in lieu of the standard forms. In the study of Riccati type equations [5], an attempt has been made to apply the factorization methods directly to the steady state Riccati equation but an algorithm to produce these results is required.

In the study of Riccati type equations, many methods exist to determine solutions such as iterative procedures [6,7], the partitioned algorithm approach [8,9,10], the Matrix Sign Function approach [11,12], and other methods applicable in numerical integration, e.g. [13]. The Matrix Sign Function approach is one method which has the advantage of obtaining additional non-positive definite solutions through the use of a Symplectic matrix composed of the matrices in the Riccati equation. This aspect has been discussed by Bucy [13] and Potter [14]. This paper will develop an algorithm for determining the square root of a positive definite symmetric matrix. The convergence of the algorithm will be proved with upper and lower apriori bounds on the converged value determined at any point along the sequential process. This algorithm can then be applied to the implementation of Riccati type equations. The Riccati type equations that are applicable for this approach include, but are not limited to, the discrete version of the partitioned Riccati solutions [8,10], and the various covariance expressions which occur in square root filtering methods already discussed [3,15]. It is noted that the square root algorithm presented here will be compared (through examples) to the Matrix Sign approach [11] and not to the triangular factorization methods. In a hard to find reference [16] another (but different) algorithm was developed which has been discussed by Bellman [17]. The algorithm in [16], however, has limitations on the norm of the matrix considered and appears to be related to a version of the spectral factorization problem. In Astrom's book [18], similar algorithms result from studies of related spectral factorization problems.

In order to better understand why the factorization methods yield algorithms which have numerical advantages over other methods, this paper is divided in four parts. First the motivation for using this approach is demonstrated by working a scalar example using an algorithm from Number Theory called Euclid's algorithm to characterize an irrational number in terms of continued fractions. Such methods are used in Number Theory [19,20], and for scalar irrationals (such as a square root), they give rise to definitions such as "most efficient expansion" and "best possible approximation". By defining a structure of the continued fraction expansion subject to certain constraints, the definition of "most efficient expansion" can be given in an explicit manner. Using a sequential procedure defined as Euclid's algorithm [19,20], the continued fraction expansion is obtained for the irrational number \( \pi \). Euclid's method (or a "most efficient expansion") is then constructed for a scalar square root. Apriori upper and lower bounds are obtained for both expansions of these two scalar irrational numbers.

Part II of this paper introduces the "square root algorithm" which differs from Euclid's method. The square root algorithm is applied first to the scalar square root and the resulting continued fraction expansion is compared to Euclid's method. The apriori upper and lower bounds are also calculated. The third part of this paper extends the scalar square root algorithm to the matrix case. Theorems on convergence are proved and examples are worked. The matrix square
root algorithm considered here is compared (through an example) to the Matrix Sign Approach and the convergence of the algorithm is studied numerically.

Finally, the fourth part of this paper considers the application of this approach to the implementation of Riccati type equations. A manner of obtaining apriori upper and lower bounds is demonstrated for Riccati type equations. Some theorems are given and an algorithm is demonstrated which extends the results obtained here for the matrix case into an algorithm for computing the Riccati equation. The apriori upper and lower bounds for the Riccati equation are obtained sequentially as in the matrix case. First, part I of this paper will introduce Euclid's algorithm and provide a motivation for using factorization algorithms.

II. Part—I Euclid's Algorithm—A Motivation For Using Square Root Methods

It is well known that numerical problems occur in computing Riccati solutions or other matrix equations if the matrices are ill conditioned [21]. One measure of ill conditioning occurs if the eigenvalues of the free system are separated by more than one order of magnitude. One would expect, therefore, that factorization methods should prove to have numerical advantages because the square root matrix may have eigenvalues with less dispersion than the original free system matrix. In an effort to study this effect numerically, Euclid's algorithm is introduced and a scalar example is worked from Number Theory on an irrational number. The purpose of this example is to illustrate to the reader the relationship between the factorization methods to continued fraction approaches and also to define the numerical advantages (in terms of a definition called "most efficient") of the factorization methods considered here. First some definitions are necessary.

Let \( \xi_0 \) denote a scalar irrational number. The problem of interest is to determine a continued fraction expansion of \( \xi_0 \) in terms of rational numbers. First the definition of a continued fraction is specified [19,20]:

**Definition:** The most efficient [19,20] expansion of an irrational scalar number \( \xi_0 \) by a continued fraction representation is the one in which \( \xi_0 \) can be accurately expressed to as many decimals as possible by the fewest number of integer terms in the expansion.

Example (1):

Let \( \xi_0 \) be the irrational number \( \pi \), i.e., \( \xi_0 = 3.14159+ \)

Hence \( a_0 = [\xi_0] = 3 \). To calculate \( a_1 \), compute

\[
\xi_1 = \frac{1}{\xi_0 - a_0}
\]

or

\[
\xi_1 = \frac{1}{\xi_0 - 3} = 1.06951504075417358902 \text{ etc.}
\]

Now compute \( a_1 = [\xi_1] = 7 \). In this manner we calculate (using double precision (29 digits) on a CDC computer) the following numerical results:

\[
\begin{align*}
\pi &= 3.141592653589793238462643383279 \\
\xi_0 &= 3.141592653589793238462643383279 \\
\xi_1 &= 7.0625133059104576979300051531 \\
\xi_2 &= 15.996594406856941310599874 \\
\xi_3 &= 1.00341723101339778563949394 \text{ etc.}
\end{align*}
\]

where each \( a_i \) is subject to the constraint of being positive and an integer. Then Euclid's algorithm [20] is defined as follows:

**Denote:**

\[
a_i = [\xi_i]
\]

where \( a_1 \) is defined as the nearest integer smaller than the irrational number \( \xi_1 \). Euclid's algorithm for computing the \( a_i \) from the \( \xi_i \) proceeds as follows:

\[
a_0 = [\xi_0], \text{ now proceeding as an algorithm, let,}
\]

\[
\xi_1 = \frac{1}{\xi_0 - a_0}
\]

and let \( a_1 = [\xi_1] \). Inductively it follows that:

\[
\xi_{i+1} = \frac{1}{\xi_i - a_i}
\]

and \( a_i = [\xi_i] \) and this completes the algorithm for all \( a_i \). It is now appropriate to give a definition of "most efficient expansion" as it relates in the context of the continued fractions in equations (1,2).

**Definition:** "Most Efficient Expansion"

The most efficient [19,20] expansion of an irrational scalar number \( \xi_0 \) by a continued fraction representation is the one in which \( \xi_0 \) can be accurately expressed to as many decimals as possible by the fewest number of integer terms in the expansion. It is noted that \( \xi_0 \) is subject to the constraint of equation (2) with each \( a_i \), positive integers. It has been shown [19,20] that Euclid's method satisfies this property for scalar irrationals. An example will now be worked with the irrational number \( \pi \) to illustrate Euclid's algorithm.

Example (1): Let \( \xi_0 \) be the irrational number \( \pi \), i.e., \( \xi_0 = 3.14159+ \)

Hence \( a_0 = [\xi_0] = 3 \). To calculate \( a_1 \), compute

\[
\xi_1 = \frac{1}{\xi_0 - a_0}
\]

or

\[
\xi_1 = \frac{1}{\xi_0 - 3} = 1.06951504075417358902 \text{ etc.}
\]

Now compute \( a_1 = [\xi_1] = 7 \). In this manner we calculate (using double precision (29 digits) on a CDC computer) the following numerical results:

\[
\begin{align*}
\pi &= 3.141592653589793238462643383279 \\
\xi_0 &= 3.141592653589793238462643383279 \\
\xi_1 &= 7.0625133059104576979300051531 \\
\xi_2 &= 15.996594406856941310599874 \\
\xi_3 &= 1.00341723101339778563949394 \text{ etc.}
\end{align*}
\]
In order to study convergence and to develop apriori bounds, it is of interest to study the partial sums of equation (4). Let \( r_0 = a_0, r_1 = a_0 + 1/a_1 \), and in general:

\[
N_r = \left< a_0, a_1, \ldots, a_N \right>
\]

(5)

It has been shown explicitly [19, 20] that for the scalar case, the following upper and lower bounds exist for \( \pi \):

\[
r_0 < r_2 < r_4 < \ldots < r_N = \pi < \cdots < r_5 < r_3 < r_1
\]

(6)

which gives rise to apriori bounds of the form:

\[
3 < \frac{333}{106} < \ldots < \frac{355}{113} < \frac{22}{7}
\]

(7)

The scalar version of this proof was presented for the square root of a scalar number in [22], the matrix case will be presented in the sequel. Therefore by calculating \( r_0 \) and \( r_1 \), an upper and lower bound on \( \pi \) can be determined "apriori." By then calculating \( r_2 \) and \( r_3 \), an even more accurate bound on \( \pi \) can then be determined. This procedure can be continued indefinitely with convergence to values as close as desired. To illustrate to the reader how these apriori bounds can be obtained for the irrational number \( \pi \), the terms \( r_1, r_2, \ldots, r_N \) are calculated for the fraction in equation (4).

\[
\begin{align*}
r_0 &= 3 \\
r_1 &= 22/7 \\
r_2 &= 333/106 \\
r_3 &= 355/113 \\
r_4 &= 103993/33102 \\
r_5 &= 3716521/1213705 \\
r_6 &= 2892477/964144 \\
r_7 &= 1213705/3926816
\end{align*}
\]

It is easily seen from these numerical values that the \( r_N \) satisfy the relationships specified by equation (6). Also from the example it is observed that the first four terms of the expression \( r_N \) give accuracy beyond 7 decimal places. One should now be motivated to apply Euclid's algorithm to a quadratic equation to observe the results.

III Euclid's Algorithm Applied To A Scalar Square Root

Taking as an example the square root of 3, the most efficient method will be studied numerically using example (2):

**Example (2):**

Expanding the square root of 3 to 30 decimal places yields:

\[
\xi_0 = 3.1415926535897932384626433832795028841971693993751
\]

(8)

From the example it is observed that the definition "most efficient" does not have the same meaning in the matrix case. In order to have an algorithm that does extend to the matrix case, the square root algorithm will next be presented. The square root algorithm gives identical results to the "most efficient" algorithm for the scalar case. The square root algorithm, however, can be extended to obtain matrix square roots, to implement the Riccati equation expansions of the partitioned algorithm [8], this same periodicity occurs. The periodicity obtained here numerically occurs due to:

\[
\xi_j = \xi_{j+2} \quad \text{for all } j
\]

with

\[
a_i = 1 \quad \text{for odd } \]

\[
a_i = 2 \quad \text{for even}
\]

and this is a result due to the following identity:

\[
1 + \sqrt{3} = \frac{1}{\sqrt{3} - 1}
\]

(9)

The results obtained here so far indicate that using the "most efficient" approach, the square root methods have a periodic property. If this method were to be applied to the matrix case, the results do not extend readily due to the fact that the definition "most efficient" does not have the same meaning in the matrix case. In order to have an algorithm that does extend to the matrix case, the square root algorithm will next be presented. The square root algorithm gives identical results to the "most efficient" algorithm for the scalar case. The square root algorithm, however, can be extended to obtain matrix square roots, to implement the Riccati equation, and, in addition, to determine upper and lower apriori bounds.

IV - Part II - The Square Root Algorithm (Scalar Case)

Consider example (2) to determine the square root of 3 via the following sequence of steps:

Let \( W = \sqrt{3} \)

Rewrite this as:

\[
W = 1 + \frac{1}{\frac{1}{2} (\sqrt{3} + 1) - 1}
\]

(9)

or

\[
W = 1 + \frac{1}{\frac{1}{2} (\sqrt{3} + 1) - 1}
\]

Distribute: (not shown)

\[
\text{Dist.}
\]

\[
A \quad \text{or SPECIAL}
\]

\[
936
\]
where the inequality representation for two matrices \( A \geq B \) implies that \( A-B \) is positive definite and \( I \) is the \( n \times n \) identity matrix. If equation (15) is not satisfied for a specified \( S_0 \) and \( S_1 \), then we require:

\[ S_0 - \epsilon I > 0, \quad \text{and} \quad S_1 - \epsilon^2 I > 0 \]  

(16)

for some \( \epsilon > 0 \). The ensuing derivation follows exactly with (16) replaced by (15), but for notation simplicity, the inequality of equation (15) will be used. To derive the square root algorithm, proceed as in equations (9-12) for the scalar case.

Denote:

\[ S_0 = S_1^{1/2} \]

This can be written as:

\[ S_0 = I + S_1^{1/2} - I \]  

(18)

Theorem 1 will now be helpful in the derivation:

**Theorem 1:**

\[ S_1^{1/2} - I = (S_1 - I)(S_1^{1/2} + I)^{-1} = (S_1^{1/2} + I)(S_1 - I)^{-1} \]

(19a)

\[ S_1^{1/2} + I = (S_1 - I)(S_1^{1/2} - I)^{-1} = (S_1^{1/2} - I)(S_1 - I)^{-1} \]

(19b)

Proof: Consider the identities:

\[ 1/2 = (S_1 - I)(S_1^{1/2} + I) = S_1 - I \]

(20a)

\[ (S_1^{1/2} + I)(S_1^{1/2} - I) = S_1 - I \]

(20b)

Equation (19a) follows by pre and post multiplying equations (20a-b) by \((S_1^{1/2} + I)^{-1}\) which is positive definite. Equation (19b) follows by pre and post multiplying equations (20a-b) by \((S_1^{1/2} - I)^{-1}\) which is known to be positive definite by the inequality (15). Q.E.D.

The derivation of the algorithm will now continue by rewriting equation (18) as follows:

\[ S_0 = I + (S_1^{1/2} - I)(S_1^{1/2} + I)(S_1^{1/2} + I)^{-1} \]  

(21)

From equation (20a) this implies (21) can be written:

\[ S_0 = I + (S_1^{1/2} - I)(S_1^{1/2} + I)^{-1} \]

(22)

Since for two matrices \( A, B \), \((A - B)^{-1} = -1\), it is desired to represent equation (22) as:

\[ S_0 = I + I (S_1^{1/2} + I)(S_1^{1/2} - I)^{-1} \]

(23)

This can be written as:

\[ S_0 = I + I ( - S_1^{1/2})^{-1} + S_1^{1/2} - I \]

(24)

Now the procedure used in equations (17-18) will be reapplied to equation (24) resulting in:

\[ S_0 = I + I [ (S_1^{1/2} - I)^{-1} + I^{1/2} - I(S_1^{1/2} - I)^{-1} - I(S_1^{1/2} - I)^{-1} ]^{-1} \]

(25)

It is worthwhile to rewrite this as:

\[ S_0 = I + I [ 2(S_1^{1/2} - I) + I^{1/2} - I(S_1^{1/2} - I)^{-1} - I(S_1^{1/2} - I)^{-1} ]^{-1} \]

Using the results of equation (19b) yields:

\[ S_0 = I + I [ 2(S_1^{1/2} - I) + I^{1/2} + I^{1/2} - 1(S_1^{1/2} - I)^{-1} ]^{-1} \]

(26)
Now expanding the inner term yields:

\[ S_0 = I + [2(S_1 - I)^{-1} + I] (S_1^{1/2} + I - I)^{-1} \]  

(27)

This can be rewritten as:

\[ S_0 = I + I[2(S_1 - I)^{-1} + I(2I^{1/2} - I)^{-1}]^{-1} \]  

(28)

The observant reader may now compare the term \( S_1^{1/2} \) in equation (18) with the same term in the inner-most brackets of equation (28). The next step in the derivation is to repeat the procedure from equations (18-28) resulting in the next expression as follows:

\[ S_0 = I + I[2(S_1 - I)^{-1} + I(2S_1^{1/2} - I)(S_1^{1/2} + I)^{-1}]^{-1} \]  

(29)

eq \text{etc. Since the derivation is periodic from this point on, the results can be summarized as follows:}

Let

\[ S_A = 2I \]  

(30a)

\[ S_B = (S_1 - I)^{-1} \]  

(30b)

Then:

\[ S_0 = I + S_B + I \]  

(31)

This could also be written as follows:

\[ S_0 = I + S_B + I \]  

(32)

The expressions (31-32) obtained are desired for several reasons. First, the matrix case reduces to the "most efficient" approach as \( S_A \) and \( S_B \) become scalars. Secondly, the matrices \( S_A \) and \( S_B \) are positive definite and it is desirable to have this property to obtain apriori bounds. Thirdly, it appears that this method has application in the implementation of the Riccati equation as observed in references [3,8,15]. The theorems on convergence will now be shown with their proofs in Appendix (A). Some examples will then be worked to illustrate this approach.

VI-Convergence Proofs of The Square Root Algorithm

In order to establish a theorem on the convergence of the nested matrix sequence (31), a sequence of partial sums of the expression (31) will be derived. Denote \( R_i \), \( i=0,1,\ldots \) as the matrix partial sums of (31) which reduce to the scalar partial sums \( r_i \) defined in equation (5).

Let:

\[ R_0 = I \]  

(33a)

\[ R_1 = I + I[2(S_1 - I)^{-1}]^{-1} \]  

(33b)

\[ R_2 = I + I[2(S_1 - I)^{-1} + I(2I^{-1})^{-1}]^{-1} \]  

(33c)

\[ R_3 = I + I[2(S_1 - I)^{-1} + I(2I^{-1})^{-1}]^{-1} \]  

(33d)

or as an algorithm:

\[ R_0 = I \]  

(34a)

\[ R_1 = I + I[2(S_1 - I)^{-1}]^{-1} \]  

(34b)

\[ \ldots \]  

\[ R_N = I + I[2(S_1 - I)^{-1} + I(2I^{-1})^{-1}]^{-1} \]  

(34c)

Theorem 2 contains the properties of convergence of this algorithm.

**Theorem 2:**

\[ \lim_{N \to \infty} R_N = S_1^{1/2} \]  

(35a)

with the following apriori bounds:

\[ R_0 < R_2 < R_4 < \cdots < R_N < S_1 < \cdots < R_5 < R_3 < R_1 \]  

(35b)

The proof of theorem (2) is given in appendix (A). The scalar version of this proof was given in [22] and similar proofs of the scalar example are given in the references on Number Theory [19,20]. Some examples will now be worked to illustrate the convergence properties of the algorithm and also how the apriori bounds may be used.

VII Some Numerical Examples Using The Matrix Square Root Algorithm:

As the simplest example to observe the procedure presented here, consider the square root of the following matrix:

\[ S_1 = \begin{bmatrix} 4.0 & 0.0 \\ 0.0 & 25.0 \end{bmatrix} \]  

(36)

Using the algorithm results in \( R_{10} \) and \( R_{11} \) as follows:

\[ R_{10} = \begin{bmatrix} 1.999797 & 0.0 \\ 0.0 & 4.746472 \end{bmatrix} \]  

and \( R_{11} = \begin{bmatrix} 2.000068 & 0.0 \\ 0.0 & 5.176476 \end{bmatrix} \)  

Please recall the inequality:

\[ R_N < S_1^{1/2} < R_M \]  

is true for all \( N \) even integers and all \( M \) odd integers. Also, the apriori bounds \( R_0 \) and \( R_1 \) indicate the accuracy of this procedure.

**Example 4**

\[ S_1 = \begin{bmatrix} 4.0 & 6.0 \\ 6.0 & 25.0 \end{bmatrix} \]  

(37)

Using the Matrix Square Root Algorithm results in \( R_{30} \) and \( R_{31} \) as follows:
It is interesting to compute the products $R_{30}^TR_{30}$ and $R_{31}^TR_{31}$:

$$
R_{30}^T = \begin{bmatrix} 3.999921371 & 5.999700491 \\ 5.999700491 & 24.9988798 \end{bmatrix},
R_{31}^T = \begin{bmatrix} 6.0000555 & 6.0002036 \\ 6.0002036 & 25.00075808 \end{bmatrix}
$$

which can be compared to $S_1$ of equation (36). The next example was taken from reference [11] to serve as a comparison to the approach presented here and also to look at higher order systems.

**Example 5**

$$
S_1 = \begin{bmatrix} 1.2 \times 10^5 & 230. & 10. \\ 230. & 1000. & 1. \\ 10. & 1. & 0.5 \end{bmatrix}
$$

Using the Matrix Square Root Algorithm presented here, the results obtained were:

$$
R_{632} = \begin{bmatrix} 328.73994 & 0.608420 & 0.0287555 \\ 0.608420 & 31.61690 & 0.0303966 \\ 0.028755 & 0.0303966 & 0.7058786 \end{bmatrix}

R_{633}^T = \begin{bmatrix} 364.40961 & 0.608420 & 0.0287555 \\ 0.608420 & 31.61690 & 0.0303966 \\ 0.028755 & 0.0303966 & 0.7058786 \end{bmatrix}
$$

Using the Matrix Sign Approach [11], Denman found the computed square root as:

$$
S_0 = \begin{bmatrix} 346.40961 & 0.608420 & 0.0287555 \\ 0.608420 & 31.61690 & 0.0303966 \\ 0.028755 & 0.0303966 & 0.7058786 \end{bmatrix}
$$

Using the Matrix Square Root Algorithm presented here, the results obtained were:

$$
R_{632} = \begin{bmatrix} 328.73994 & 0.608420 & 0.027283 \\ 0.608420 & 31.616842 & 0.030394 \\ 0.027283 & 0.030394 & 0.705868 \end{bmatrix}

R_{633}^T = \begin{bmatrix} 364.18997 & 0.644195 & 0.030298 \\ 0.644195 & 31.616978 & 0.030400 \\ 0.030298 & 0.030400 & 0.705868 \end{bmatrix}
$$

Obviously for higher order systems, some modes converge much faster than other modes. In other words the ill conditioning of $S_1$ will effect this algorithm (by slowing down its convergence) in a manner dependent on the degree of ill conditioning of $S_1$. Also the effects of numerical bias occur in this algorithm as can be seen because the element in the Riccati equation. The algorithm yields:

$$
S_1 = \begin{bmatrix} 4.0 & 10.0 \\ 10.0 & 25.0 \end{bmatrix}
$$

The interesting result occurs by considering the product $R_{30}^TR_{30}$ and $R_{31}^TR_{31}$:

$$
R_{30}^T = \begin{bmatrix} 4.00072821 & 9.998849043 \\ 9.998849043 & 24.99831445 \end{bmatrix},
R_{31}^T = \begin{bmatrix} 4.001202784 & 10.000123404 \\ 10.000123404 & 25.001421516 \end{bmatrix}
$$

which can be compared to $S_1$ of equation (37); this procedure appears to work if $S_1$ is only positive semi-definite. Theorem (3) illustrates that a best estimate of a matrix square root can be obtained by using the mean value of the best two a priori bounds.

**Theorem 3:**

(3a) The "best estimate" of a matrix square root can be obtained as follows:

$$
\hat{S}_0 = \max \left( \frac{1}{2} \right) \left| R_N + R_{N+1} \right|
$$

(3b) The error in the estimate specified by equation (38a) can be no worse than the following bound:

$$
\left| S_0 - \hat{S}_0 \right| \leq \left| R_{N+1} - R_N \right|
$$

Hence even if $S_1$ is not known, $R_{N1}$ and $R_N$ provide an estimate of the accuracy of this procedure. One may now take appropriate matrix norms and examine examples 3, 4, and 5. Appendix B demonstrates a proof of this theorem with an appropriate definition of "best estimate". This result can be seen numerically in examples (3-5). The estimate specified by equation (38a) is as accurate a method as possible to guess a matrix square root. The methods obtained here will now be applied to the discrete version of the Riccati equation.

**VIII Part IV—Application of This Approach To Discrete Implementation of The Riccati Equation**

In the implementation of the continuous Riccati equation at discrete times $0 \leq t_k < t_1 < t_2$ with

$$
\Delta = t_{k+1} - t_k,
$$
the following recursion relationships can be obtained:

$$
P_{k+1} = P_0 + A \left( W + P_k^{-1} \right)^{-1} A^T
$$

(39)
where \( P_0, A, \) and \( W \) are quantities obtained over the time interval \( \Delta \) of a Riccati equation with zero initial conditions. Equation (39) appears in one form or another in the partitioned algorithm approach of Lainiotis [8], the square root filtering algorithms of Morf and Kailath [3], and in Potter and Kombie [15]. To apply this approach similar to the manner in which the matrix algorithm (34a-c) was obtained, note the following sequences:

\[
\begin{align*}
R_0 & = P_0 \\
R_1 & = P_0 + A[W]^{-1}A \\
R_2 & = P_0 + A[W + (P_0)I - I]^{-1}A \\
R_3 & = P_0 + A[W + (P_0 + A[W]^{-1}A^{-1} - 1)^{-1}A].
\end{align*}
\]

The Riccati Equation Algorithm becomes:

\[
\begin{align*}
\bar{R}_0 & = P_0 \\
\bar{R}_1 & = P_0 + A[W]^{-1}A \\
\bar{R}_2 & = P_0 + A[W + (\bar{R}_{N-2})]^{-1}A \\
\bar{R}_N & = P_0 + A[W + (\bar{R}_0)I - I]^{-1}A.
\end{align*}
\]

Theorem (4) contains the application of the approach here for Riccati equations.

**Theorem 4:**

If \( P_0 > 0, A > 0, \) and \( W > 0 \), then:

(a) All \( \bar{R}_N > 0, i = 0, 1, \ldots \)

(b) \( \bar{R}_N < R < \bar{R}_{N+1} \), \( N \) even, \( N \) odd with \( N < K \), and \( M < K \).

(c) The following apriori bounds exist as \( N \to \infty \) (or as \( \Delta t \to 0 \)) for the continuous case:

\[
\frac{R_0}{R_n} < \frac{R_1}{R_{n+1}} < \cdots < \frac{1}{1 + R_{n+1}} < \cdots < \frac{1}{P_0} < A[W + (R_0)^{-1}A]^{-1}A < \cdots \]

(i.e. as many apriori bounds can be found as is desired for the numerical scheme.

(d) The best estimate of \( P_N \) (in the continuous case) can be expressed as follows:

\[
\hat{P}_N = \max N \left( \frac{1}{2} \right) \left( \bar{R}_N + \bar{R}_{N+1} \right)
\]

(e) The error in the estimate specified by equation (41) can be no worse than the following bound:

\[
\| P_N - \hat{P}_N \| \leq \| \bar{R}_{N+1} - \bar{R}_N \|
\]

The proof of theorem (4) is outlined in Appendix (C).

The results of this last section seem most interesting in the study of the accuracy of methods to implement the Riccati equation. It is the contention here that since these methods reduce to the "most efficient" method in the scalar case, they should be considered first in the calculation of Riccati type equations.

**IX Summary and Conclusions**

An iterative algorithm is presented for obtaining matrix square roots. The techniques used to derive the algorithm can be used in discrete implementation of the Riccati equation. The advantage of this approach is the development of apriori bounds on both the matrix computations and the Riccati solutions. Examples are worked to illustrate this method.

**Appendix A**

In this appendix all inequalities refer to matrix inequalities, i.e. \( A \geq B \) implies \( x^T[A-B]x \) is positive for all vectors \( x \) of the appropriate dimensions. To prove theorem 2, lemmas 1-7 must be shown to be true. Lemma (1):

Let \( A_1, B, A_2, C_1, C_2 \) be positive definite matrices with:

\[
A_1 = [B^{-1} + C_1^{-1}]^{-1} \quad (A.1)
\]

\[
A_2 = [B^{-1} + C_2^{-1}]^{-1} \quad (A.2)
\]

if \( C_1 \geq C_2 \), then \( A_1 \geq A_2 \).

Proof:

\[
A_1^{-1} = B^{-1} + C_1^{-1}
\]

and \( A_2^{-1} = B^{-1} + C_2^{-1} \)

since \( C_1 \geq C_2 \) and both matrices are positive definite then we have [17]: \( C_2^{-1} \geq C_1^{-1} \)

but

\[
C_2^{-1} - A_2^{-1} = -B^{-1} - B^{-1}
\]

\[
C_1^{-1} - A_1^{-1} = -B^{-1} \quad (A.3)
\]

or \( A_2^{-1} \geq A_1^{-1} \), but since both matrices are positive definite implies \( A_1 \geq A_2 \). Q.E.D.

**Lemma 2:**

Let \( A, B, \) and \( C \) be positive definite matrices with

\[
C = (A^{-1} + B^{-1})^{-1} \quad (A.4)
\]

\[
C \leq B \quad (A.5)
\]

940
Proof of Lemma 2:

Note
\[ C^{-1} = A^{-1} + B^{-1} \]

Hence
\[ C^{-1} \succ A^{-1} \]

And
\[ C^{-1} \succ B^{-1} \]

Therefore \( C \prec B \) and \( C \ll A \) follows.

Q.E.D.

Lemma 3

The matrix subsequences \( R_N \) for \( N=0,2,4,6,\ldots \) (all even integers) form a monotonically increasing matrix sequence, i.e.

\[ R_0 < R_2 < R_4 < R_6 < \cdots < R_N < \cdots \]

Proof:
The proof is by mathematical induction. By calculation:

\[ R_0 = I \]

\[ R_2 = I - I + [2(S_1 - I)^{-1} + (1 + R_3)^{-1}]^{-1} = I \]

Hence \( R_2 > R_0 \). For notation simplicity let \( B = \frac{S_1 - I}{2} \geq 0 \), then:

\[ R_2 = I + [B^{-1} + (1 + R_3)^{-1}]^{-1} \quad \text{(A.6)} \]

\[ R_4 = I + [B^{-1} + (1 + R_2)^{-1}]^{-1} \quad \text{(A.7)} \]

But from lemma (1) if \( 1 + R_2 \geq 1 + R_0 \), and using (A.6) and (A.7) yields:

\[ R_4 > R_2 \]

Now assume the results hold for \( R_N \), it is necessary to show that they hold for \( R_{N+2} \). Note we assume

\[ R_N \succ R_{N-2} \quad \text{(N even)} \]

but

\[ R_N = I + [B^{-1} + (1 + R_{N-2})^{-1}]^{-1} \]

\[ R_{N+2} = I + [B^{-1} + (1 + R_{N})^{-1}]^{-1} \]

But from lemma (1) if \( 1 + R_N \geq 1 + R_{N-2} \) implies

\[ R_{N+2} > R_N \]. Since the results hold for \( N=0,2 \), and \( N=2 \), they hold for all \( N \).

Q.E.D.

Lemma (4):
The matrix subsequences \( R_N \) for \( N=1,3,5,7,\ldots \) (all odd integers) form a monotonically decreasing matrix sequence, i.e.

\[ R_1 > R_3 > R_5 > R_7 > \cdots \]

Proof:
Again the proof is by mathematical induction. By calculation:

\[ R_1 = (1/2) \left[ S_1 + I \right] \]

\[ R_3 = I + \left[ (S_1 - 1)^{-1} + I [2I + 2(S_1 - 1)^{-1} - 1]^{-1} \right]^{-1} \]

But using lemma (2) implies:

\[ R_3 - I \ll \frac{S_1 - I}{2} \]

or

\[ R_3 \ll \frac{S_1 + I}{2} = R_1 \]

Hence

\[ R_3 < R_1 \]

For notation simplicity let \( B = \frac{S_1 - I}{2} > 0 \), then:

\[ R_3 = I + [B^{-1} + (1 + R_3)^{-1}]^{-1} \]

\[ R_5 = I + [B^{-1} + (1 + R_5)^{-1}]^{-1} \]

Now use lemma (1). If \( 1 + R_1 \geq 1 + R_3 \), then this implies from lemma (1) that \( R_3 \not\geq R_5 \) or \( R_3 > R_5 \). Now assume the results hold for \( R_N \), it is necessary to show that they hold for \( R_{N+2} \). We assume:

\[ R_N < R_{N+2} \quad \text{(N odd)} \]

but from lemma (1) if \( 1 + R_N \ll 1 + R_{N-2} \), this implies

\[ R_{N+2} < R_N \]. Since the results hold for \( N+1,3,5, \) then they hold for all \( N \).

Q.E.D.

Lemma (5)
The following statements are true:

(A.5.1) All \( R_N \) (N even) are bounded above by \( R_1 \).

(A.5.2) All \( R_N \) (N odd) are bounded below by \( R_0 \).

(A.5.3) \( \lim_{N \to \infty} R_N = R_0 \).

(A.5.4) \( \lim_{N \to \infty} R_N = R_{N+2} \).

Proof:
To show (A.5.1) is true, i.e.

\[ R_0 < R_2 < R_4 < R_6 < \cdots < R_1 \] \quad \text{(A.8)}

First it is known that all \( R_N \) (N even) satisfy:

\[ R_{N+2} = I + \left[ \left( \frac{S_1 - I}{2} \right)^{-1} + (1 + R_N)^{-1} \right]^{-1} \]

or

\[ R_{N+2} - I = \left[ \left( \frac{S_1 - I}{2} \right)^{-1} + (1 + R_N)^{-1} \right]^{-1} \]

Now use lemma 2 which implies that:

\[ R_{N+2} - I < \frac{S_1 - I}{2} \]
or

\[ R_{N+2} < \frac{S_1 + 1}{2} = R_1 \]  

\[ \text{(A.9)} \]

Since \( R_0 > 1 \) for \( S_1 > 1 \), then (A.9) implies that \( R_{N+2} < R_1 \) for all \( N \) even.

To show (A.5.2) is true, i.e.

\[ R_1 > R_3 > R_5 > \ldots > R_M \ldots \rightarrow R_0 \quad (M \text{ odd}) \]

\[ \text{(A.10)} \]

Proof:

\[ R_1 = \frac{S_1 + 1}{2} \rightarrow 1 - R_0 \text{ if } S_1 > 1 \]

Hence \( R_1 > R_0 \), but all \( R_M \) (M odd) satisfy:

\[ R_{N+2} = 1 + \left( \frac{S_1}{2} \right)^{-1} \right) + (I + R_m)^{-1} \right] - 1 \]

or

\[ R_{M+2} = R_1 \rightarrow \frac{S_1}{2} \right] + (I + R_N)^{-1} \right] - 1 \]

or \( R_{M+2} \rightarrow R_0 \) for all \( M \) odd. This implies (A.5.2) is true for all \( M \) odd.

The statements (A.5.3) and (A.5.4) follow from (A.5.1), (A.5.2), lemmas (3,4) and the theorems of monotone operators in a Banach space [23].

Lemma (6):

\[ R_{N+1} - R_N < 0 \text{ if } N \text{ is odd} \]

\[ R_{N+1} - R_N > 0 \text{ if } N \text{ is even} \]

Proof: The following expression can be calculated:

\[ R_{N+1} - R_N = \left[ \left( \frac{S_1}{2} \right) + (I + R_N)^{-1} \right] - 1 \]

\[ - \left[ \left( \frac{S_1}{2} \right) + (I + R_N)^{-1} \right] - 1 \]

rewrite this as:

\[ R_{N+1} - R_N = A_1 - A_2 \]

From lemma (1) we have: \( A_1 - A_2 > 0 \) if \( R_{N-1} > R_{N-2} \) and \( A_1 - A_2 < 0 \) if \( R_{N-1} < R_{N-2} \)

but from lemma (5) we know \( R_0 < R_2 < R_4 \)

Hence:

\[ R_1 - R_0 > 0 \text{ if } N = 0 \]

\[ R_2 - R_1 < 0 \text{ if } N = 1 \]

By mathematical induction assume \( R_{N-1} < R_{N-2} \) (for \( N \) odd), then from lemma (1), \( A_1 - A_2 < 0 \). Since this is true for \( N=3 \), it holds for all \( N \).

Q.E.D.

Assume \( R_{N-1} > R_{N-2} \) (for \( N \) even). By mathematical induction:

\[ A_1 - A_2 > 0 \text{ by lemma (1)} \]

Since this is true for \( N=2 \), it holds for all \( N \).

Q.E.D.

Lemma (7):

For \( N \) even \[ \lim_{N \to \infty} R_N = R_{\text{mod}} \]

And

For \( M \) odd \[ \lim_{N \to \infty} R_M = R_{\text{mod}} \]

Proof: The proof is by contradiction. Assume the contrary and

\[ R_{\text{mod}} - R_{\text{mod}} = \Delta + 0 \]

The following expression is easily calculated:

\[ \Delta = \lim_{N \to \infty} (R_N - R_N) = \lim_{N \to \infty} B^{-1} + (I + R_N)^{-1} \]

\[ = B^{-1} + (I + R_N)^{-1} \]

If \( N \) is odd, then from lemma (6) this implies \( \Delta < 0 \).

If \( N \) is even, then from lemma (6) this implies \( \Delta > 0 \). These last two statements yield a contradiction if \( \Delta > 0 \). Note if \( \Delta = 0 \), then everything is consistent. This completes the proof of theorem (2).

Appendix B

To show the best estimate of a matrix square root is:

\[ \hat{S}_0 = \max \left( \frac{1}{2} \right) \frac{R_0 + R_{N-1}}{N} \]

Define the best estimate \( \hat{S}_0 \) as that estimate which minimizes:

\[ \min \left( \frac{1}{2} \right) \frac{R_0 - S_{1/2}}{N} \]

\[ \text{(B.1)} \]

But from lemma (3) and (4) of appendix A we have

\[ R_0 < R_2 < R_4 < \cdots < R_{2N} < S_{1/2} < R_{2N+1} < \cdots < R_1 \]

\[ \text{(B.3)} \]

Now substituting \( \hat{S}_0 \) from (B.1) into (B.2) yields:

\[ J = \left( \frac{1}{2} \right) (R_0 - S_{1/2}) + \frac{1}{2} (R_{N-1} - S_{1/2}) \]

\[ \text{(B.4)} \]

The following inequality now results:

\[ J \leq \left( \frac{1}{2} \right) (R_0 - S_{1/2}) + \frac{1}{2} (R_{N-1} - S_{1/2}) \]

\[ \text{(B.5)} \]
The inequality (8.4) has the smallest upper bound if $A^N$ and $A^{N+1}$ are chosen as the largest values of the computed a priori bounds in (8.3). The term best estimate is the guess of $F_0$ which minimizes the right hand side of equation (8.4).

To show that the statement (3b) of theorem (3) is true, the worst case guesses of $F_0$ and $F_{N+1}$ are substituted into equation (8.4). This yields the upper bound (3.8).

Q.E.D.

Appendix C

One would expect to obtain bounds on Riccati equations when formulated in the factorization framework. See, e.g. [24, 25] for simplification of Riccati equations when studied within the context of optimal control and estimation theory. The proof of theorem (4) now follows as previously demonstrated in Appendices A and B using the same technique. To outline the proof of theorem (4) the following lemmas can be easily shown to be true by following the similar proofs in Appendices A and B.

Lemma (8): The matrix subsequences $F_i$ for $N=0,2,4,6,\ldots$ (all even integers) form a monotonically increasing matrix sequence, i.e., $F_0 \leq F_2 \leq F_4 \leq F_6 \leq \cdots \leq F_N \leq \cdots$

Lemma (9): The matrix subsequences $F_i$ for $N=1,3,5,\ldots$ (all odd integers) form a monotonically decreasing matrix sequence, i.e., $F_1 > F_3 > F_5 > \cdots$.

Lemma (10): The following statements are true:

(C.10.1) All $F_N$ ($N$ even) are bounded above by $F_0$.

(C.10.2) All $F_N$ ($N$ odd) are bounded below by $F_0$.

(C.10.3) $\lim_{N \to \infty} F_N = F_{\infty}$.

(C.10.4) $\lim_{N \to \infty} F_N = F_{\infty}$.

Lemma (11): $F_{N+1} - F_N < 0$ if $N$ is even

Lemma (12): For $N$ even $\lim_{N \to \infty} F_N = F_{\infty}$

For $M$ odd $\lim_{N \to \infty} F_N = F_{\infty}$

and $F_{\infty} = F_{\infty}$. All statements in theorem (4) follow when lemmas (8-12) are proved.

References


