Contemporary Engineering Sciences, Vol. 3, 2010, no. 4, 191 - 200

Chandrasekhar Type Algorithms for the Riccati Equation of Lainiotis Filter

Nicholas Assimakis

Department of Electronics
Technological Educational Institute of Lamia, Greece
assimakis@teilam.gr

Abstract

Chandrasekhar type algorithms for solving the discrete time Riccati equation and Lyapunov equation emanating from Lainiotis filter are presented. The Chandrasekhar type algorithms are compared to the classical per step algorithm consisting of direct implementation of the recursion of the Riccati equation or the Lyapunov equation. It is shown that Chandrasekhar type algorithms may be faster than the classical ones.

Keywords: Riccati equation, Lainiotis filter, Chandrasekhar algorithm

1 Introduction

The discrete time Riccati equation arises in linear estimation and is associated with time invariant systems described by the following state space equations:

$$x(k+1) = Fx(k) + w(k) \quad \textbf{(1)}$$

$$z(k) = Hx(k) + v(k) \quad \textbf{(2)}$$

for $k \geq 0$,

where $x(k)$ is the $n \times 1$ dimensional state vector at time k, $z(k)$ is the $m \times 1$ dimensional measurement vector, F is the system transition matrix, H is the output matrix, $\{w(k)\}$ and $\{v(k)\}$ are independent Gaussian zero-mean white and uncorrelated random processes, Q and R are the plant and measurement noise covariance matrices respectively, and $x(0)$ is a Gaussian random process with mean x_0 and covariance P_0.

The filtering/estimation problem is to produce an estimate at time L of the state vector using measurements till time L, i.e. the aim is to use the measurements set

$\{z(1),\ldots,z(L)\}$ in order to calculate an estimate value $x(L/L)$ of the state vector $x(L)$. The discrete time Lainiotis filter [4] is a well known algorithm that solve the filtering problem, by computing the estimation $x(k/k)$ as well as the estimation error covariance matrix $P(k/k)$ for every k.

The Lainiotis filter equations provide a recursion for the symmetric $n \times n$ dimentional estimation error covariance matrix $P(k/k)$, which is assumed to be non-negative definite $P(k/k) \geq 0$, the **Riccati equation** emanating from Lainiotis filter:

$$P(k+1/k+1) = P_n + F_n[I + P(k/k)O_n]^{-1} P(k/k)F_n^T \quad \textbf{(3)}$$

with initial condition $P(0/0) = P_0$,
where

$$P_n = Q - QH^T AHQ \quad \textbf{(4)}$$

$$F_n = F - QH^T AHF \quad \textbf{(5)}$$

$$O_n = F^T H^T AHF \quad \textbf{(6)}$$

and

$$A = [HQH^T + R]^{-1} \quad \textbf{(7)}$$

For time invariant systems, it is well known [1] that if the signal process model is asymptotically stable (i.e. all eigenvalues of F lie inside the unit circle), then there exists a steady state value P of the estimation error covariance matrix.
It is known that Lyapunov equation is derived from Riccati equation when $R \to \infty$. In this case, $A = 0$ and $P_n = Q$, $F_n = F$, $O_n = 0$ and the Riccati equation (3) becomes the **Lyapunov equation**:

$$P(k+1/k+1) = P_n + F_n P(k/k)F_n^T \quad \textbf{(8)}$$

The discrete time Riccati equation emanating from the Lainiotis filter equations has attracted enormous attention. In view of the importance of the Riccati equation, there exists considerable literature on its recursive solutions [3], [5], concerning per step or doubling algorithms.

In this paper Chandrasekhar type algorithms for solving the discrete time Riccati equation emanating from Lainiotis filter are presented and compared to the classical per step algorithm, i.e. to the direct implementation of the recursion of the Riccati equation. The paper is organized as follows: In section 2 the classical recursive per step algorithm is presented. In section 3 recursive Chandrasekhar type algorithms are presented. In section 4 the computational requirements of all algorithms are established and comparisons are carried out. It is pointed out that the Chandrasekhar type algorithms may be faster than the classical per step algorithm. In addition, a rule is established in order to decide if the Chandrasekhar

type algorithms are faster than the classical one.

2 Per Step Algorithm

The classical per step algorithm consists of the direct implementation of the recursion of the Riccati equation emanating from the Lainiotis filter equations. Also, the corresponding per step algorithm for the solution of the Lyapunov equation is presented.

Per Step Algorithm – Riccati equation (PSARE)
The steady state solution P is calculated by recursively implementing the Riccati equation (3) for $k = 0,1,...$, with initial condition $P(0/0) = P_0$, until the following convergence criterion is satisfied: $\|P(k+1/k+1) - P(k/k)\| \le \varepsilon$, where $\|\cdot\|$ denotes the matrix norm and ε is a small positive real number pre-specified to give the steady state solution to the accuracy desired. The steady state or limiting solution $P = \lim_{k\to\infty} P(k/k)$ of the Riccati equation is independent of the initial condition [1]. In the sequel we assume zero initial condition $P(0/0) = 0$, i.e. $P_0 = 0$. Then we are able to use $P(1/1) = P_n$ as initial condition.
Note that the existance of $[I + P(k/k)O_n]^{-1}$ is guaranteed due to the presence of the identity matrix I. Also, the existence of $A = [HQH^T + R]^{-1}$ is guaranteed if R is positive definite ($R > 0$), which means that no measurement is exact. This is reasonable in physical problems. Thus, the nonsingular measurement noise covariances matrix case is assumed in the sequel.

Per Step Algorithm – Lyapunov equation (PSALE)
Lyapunov equation is derived from Riccati equation when $R \to \infty$. Then, the per step algorithm for the Lyapunov equation consists of the recursive implementation of the Lyapunov equation (8) with initial condition $P(1/1)$=P$_n$.

Table 1 summarizes the classical per step algorithms for solving the Riccati and the Lyapynov equations emanating from Lainiotis filter.

Table 1. Per Step Algorithms

Riccati equation	**PSARE**	$P(k+1/k+1) = P_n + F_n[I + P(k/k)O_n]^{-1}P(k/k)F_n^T$
Lyapunov equation	**PSALE**	$P(k+1/k+1) = P_n + F_nP(k/k)F_n^T$

3 Chandrasekhar Type Algorithms

The Chandrasekhar type algorithms use the idea of defining the difference:

$$\delta P(k) = P(k+1/k+1) - P(k/k) \quad \textbf{(9)}$$

and its factorization:

$$\delta P(k) = Y(k)S(k)Y^T(k) \quad \textbf{(10)}$$

with $Y(k)$ of dimension $n \times r$ and $S(k)$ of dimension $r \times r$, where

$$0 \le r = rank(\delta P_0) = rank(P_n) = rank(Q) \le n \quad \textbf{(11)}$$

Then using the quantity

$$O(k) = P(k/k) + O_n^{-1} \quad \textbf{(12)}$$

the following recursion is obvious:

$$O(k+1) = O(k) + Y(k)S(k)Y^T(k) \quad \textbf{(13)}$$

Note that the non singularity of O_n is guaranteed if R is positive definite and if F is nonsingular.

The Chandrasekhar type algorithms consist of the recursion:

$$P(k+1/k+1) = P(k/k) + Y(k)S(k)Y^T(k) \quad \textbf{(14)}$$

using recursions for the quantities $Y(k)$ and $S(k)$.

Two versions of the Chandrasekhar type algorithms for the solution of the Riccati equation are presented. Also, the corresponding Chandrasekhar type algorithm for the solution of the Lyapunov equation is presented.

Chandrasekhar Type Algorithm – Riccati equation – version 1 (CTARE1)

Setting

$$Y(k+1) = [F_n O_n^{-1}] O^{-1}(k) Y(k) \quad \textbf{(15)}$$

after some algebra the following recursion is derived [5]:

$$S(k+1) = S(k) - S(k)Y^T(k)O^{-1}(k+1)Y(k)S(k) \quad \textbf{(16)}$$

Note that the non singularity of $O(k)$ is guaranteed if O_n is nonsingular, which means that R is positive definite ($R > 0$).

Assuming zero initial condition $P(0/0) = 0$, we use the following initial conditions:

$$O(0) = O_n^{-1}$$

$$Y(0)S(0)Y^T(0) = P_n$$

Remarks.

1. If Q has full rank ($r = n$), then we are able touse the initial conditions $Y(0) = I$ and $S(0) = P_n$.

2. If $Q = 0$ ($r = 0$), then $A = R^{-1}$ and $P_n = 0$, $F_n = F$, $O_n = FH^T R^{-1} HF$. So the estimation error covariance is $P(k/k) = P_0 = 0$ and the limiting value P of the

estimation error covariance is $P = 0$.

Chandrasekhar Type Algorithm – Riccati equation – version 2 (CTARE2)
Setting

$$Y(k+1) = [F_n O_n^{-1}]O^{-1}(k+1)Y(k) \quad \textbf{(17)}$$

after some algebra, working as in [5], the following recursion is derived:

$$S(k+1) = S(k) + S(k)Y^T(k)O^{-1}(k)Y(k)S(k) \quad \textbf{(18)}$$

with the same initial conditions used in CTARE1.

Chandrasekhar Type Algorithm – Lyapunov equation (CTALE)
Lyapunov equation is derived from Riccati equation when $R \to \infty$. Then, the Chandrasekhar type algorithm for the Lyapunov equation becomes:

$$Y(k+1) = F_n Y(k) \quad \textbf{(19)}$$

$$P(k+1/k+1) = P(k/k) + Y(k)Y^T(k) \quad \textbf{(20)}$$

with initial conditions

$P(0/0)=0$

$Y(0)Y^T(0) = P_n$

Table 2 summarizes the Chandrasekhar type algorithms for solving the Riccati and the Lyapynov equations emanating from Lainiotis filter..

Table 2. Chandrasekhar Type Algorithms

Riccati equation	**CTARE1**	$O(k+1) = O(k) + Y(k)S(k)Y^T(k)$ $Y(k+1) = [F_n O_n^{-1}]O^{-1}(k)Y(k)$ $S(k+1) = S(k) - S(k)Y^T(k)O^{-1}(k+1)Y(k)S(k)$ $P(k+1/k+1) = P(k/k) + Y(k)S(k)Y^T(k)$
	CTARE2	$O(k+1) = O(k) + Y(k)S(k)Y^T(k)$ $Y(k+1) = [F_n O_n^{-1}]O^{-1}(k+1)Y(k)$ $S(k+1) = S(k) + S(k)Y^T(k)O^{-1}(k)Y(k)S(k)$ $P(k+1/k+1) = P(k/k) + Y(k)S(k)Y^T(k)$
Lyapunov equation	**CTALE**	$Y(k+1) = F_n Y(k)$ $P(k+1/k+1) = P(k/k) + Y(k)Y^T(k)$

4 Computational Comparison of Algorithms

Both the per step and the Chandrasekhar type algorithms are recursive ones. Thus, the total computational time required for the implementation of each algorithm is:

$$t(a\lg) = CB(a\lg) \cdot S(a\lg) \cdot top \quad \textbf{(21)}$$

where $CB(a\lg)$ is the per recursion calculation burden required for the on-line calculations of each algorithm, $S(a\lg)$ is the number of recursions (steps) that

each algorithm executes and *top* is the time required to perform a scalar operation.

The per step and the Chandrasekhar type algorithms presented above are equivalent with respect to their behavior: they calculate theoretically the same steady state estimation error variance. Then, it is reasonable to assume that both algorithms compute the limiting solution of the Riccati equation (or the corresponding Lyapunov equation) executing the same number of recursions, depending on the desired accuracy. Thus, in order to compare the algorithms with respect to their computational time, we have to compare their per recursion calculation burden required for the on-line calculations; the calculation burden of the off-line calculations (initialization process) is not taken into account.

The computational analysis is based on the analysis in [2]: scalar operations are involved in matrix manipulation operations, which are needed for the implementation of the filtering algorithms. Table 3 summarizes the calculation burden of needed matrix operations.

Table 3. Calculation Burden of Matrix Operations

Matrix Operation	Calculation Burden
$A(n\times m)+B(n\times m)=C(n\times m)$	nm
$A(n\times n)+B(n\times n)=S(n\times n)$ $S:symmetric$	$\frac{1}{2}(n^2+n)$
$I(n\times n)+A(n\times n)=B(n\times n)$ $I:identity$	n
$A(n\times m)\cdot B(m\times k)=C(n\times k)$	$2nmk-nk$
$A(n\times m)\cdot B(m\times n)=S(n\times n)$ $S:symmetric$	$n^2m+nm-\frac{1}{2}(n^2+n)$
$[A(n\times n)]^{-1}=B(n\times n)$	$\frac{1}{6}(16n^3-3n^2-n)$

The recursive computational requirements of all per step and Chandrasekhar type algorithms for solving the Riccati equation and the Lyapunov equation are summarized in Table 4. The details are given in the Appendix.

Table 4. Per Recursion Calculation Burden of Algorithms

Riccati equation	**PSARE**	$\frac{1}{6}(52n^3-6n^2+2n)$
	CTARE	$\frac{1}{6}(32n^3-3n^2+n)+3nr^2-2nr+7n^2r$
Lyapunov equation	**PSALE**	$3n^3$
	CTALE	$3n^2r$

From Table 4, we derive the following conclusions:

1. The per recursion calculation burdens of the classical per step algorithms depend only on the state dimension n.

The per recursion calculation burdens of the Chandrasekhar type algorithms

depend on the state dimension n and on dimension r.
2. The two versions of Chandrasekhar type algorithms are equivalent with respect to their computational burdens.
3. Concerning the Riccati equation solution algorithms:
- if $r = n$, then the classical per step algorithm is faster than the Chandrasekhar type algorithms
- if $r < n$, then the Chandrasekhar type algorithms may be faster than the classical per step algorithm; in fact Chandrasekhar type algorithms are faster than the classical per step algorithm if the following relation holds:

$$CB(PSARE) - CB(CTARE) = \tfrac{1}{6}(20n^3 - 3n^2 + n) - (3nr^2 - 2nr + 7n^2 r) > 0 \qquad \textbf{(22)}$$

Figure 1 depicts the relation between the dimensions n and r that may hold in order to decide which algorithm is faster.

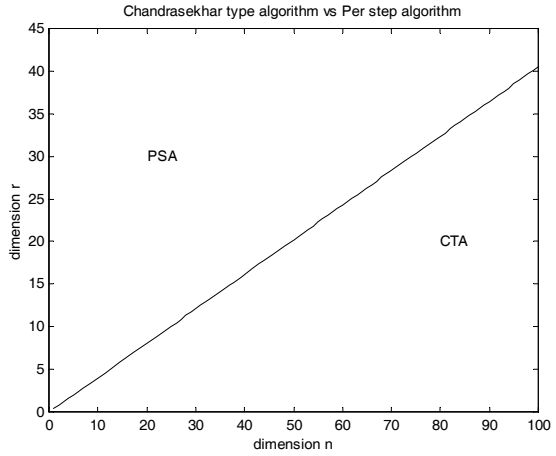

Figure 1. Chandrasekhar type algorithm may be faster than per step algorithm

Then, we are able to establish the following *Rule of Thumb*: Chandrasekhar type algorithms are faster than the classical per step algorithm if the following relation holds:

$$r < 0.4n \qquad \textbf{(23)}$$

4. Concerning the Lyapunov equation solution algorithms:
- if $r = n$, then the classical per step algorithm is as fast as the Chandrasekhar type algorithm
- if $r < n$, then the Chandrasekhar type algorithm is faster than the classical per step algorithm
Thus, Chandrasekhar type algorithms possess the advantage that there is a reduction in computational burden in comparison to the classical per step algorithm, especially when r is enough less than n.

References

[1] B. D. O. Anderson, J. B. Moore, Optimal Filtering, Prentice Hall inc., 1979.
[2] N. Assimakis and M. Adam, Discrete time Kalman and Lainiotis filters comparison, Int. Journal of Mathematical Analysis (IJMA), 1 (2007), 635-659.
[3] N. D. Assimakis, D. G. Lainiotis, S. K. Katsikas, F. L. Sanida, A survey of recursive algorithms for the solution of the discrete time Riccati equation, Nonlinear Analysis, Theory, Methods & Applications, 30 (1997), 2409-2420.
[4] Lainiotis D. G., "Partitioned linear estimation algorithms: Discrete case", IEEE Trans. on AC, vol. AC-20, pp. 255-257, 1975.
[5] Lainiotis D. G., Assimakis N. D., Katsikas S. K., "A new computationally effective algorithm for solving the discrete Riccati equation", Journal of Mathematical Analysis and Applications, vol. 186, no. 3, pp. 868-895, 1994.

Appendix. Calculation burdens of algorithms

A. Per Step Algorithms

Per Step Algorithm – Riccati equation (PSARE)

Matrix Operation	Matrix Dimensions	Calculation Burden
$P(k/k)O_n$	$(n\times n)\cdot(n\times n)$	$2n^3-n^2$
$[I+P(k/k)O_n]$	$(n\times n)+(n\times n)$ I identity	n
$[I+P(k/k)O_n]^{-1}$	$(n\times n)^{-1}$	$\frac{1}{6}(16n^3-3n^2-n)$
$[I+P(k/k)O_n]^{-1}P(k/k)$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3+\frac{1}{2}(n^2-n)$
$F_n[I+P(k/k)O_n]^{-1}P(k/k)$	$(n\times n)\cdot(n\times n)$	$2n^3-n^2$
$F_n[I+P(k/k)O_n]^{-1}P(k/k)F_n^T$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3+\frac{1}{2}(n^2-n)$
$P(k+1/k+1)=P_n$ $+F_n[I+P(k/k)O_n]^{-1}P(k/k)F_n^T$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2+n)$
PSARE	Total	$\frac{1}{6}(52n^3-6n^2+2n)$

Per Step Algorithm – Lyapunov equation (PSALE)

Matrix Operation	Matrix Dimensions	Calculation Burden
$F_nP(k/k)$	$(n\times n)\cdot(n\times n)$	$2n^3-n^2$
$F_nP(k/k)F_n^T$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3+\frac{1}{2}(n^2-n)$
$P(k+1/k+1)=P_n$ $+F_nP(k/k)F_n^T$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2+n)$
PSALE	Total	$3n^3$

B. Chandrasekhar Type Algorithms

Chandrasekhar Type Algorithm – Riccati equation – version 1/2 (CTARE1/2)

Matrix Operation	Matrix Operation	Matrix Dimensions	Calculation Burden
$Y(k)S(k)$	$Y(k)S(k)$	$(n\times r)\cdot(r\times r)$	$2nr^2-nr$
$Y(k)S(k)Y^T(k)$	$Y(k)S(k)Y^T(k)$	$(n\times r)\cdot(r\times n)$ symmetric	$n^2r+nr-\frac{1}{2}(n^2+n)$
$O(k+1)=O(k)$ $+Y(k)S(k)Y^T(k)$	$O(k+1)=O(k)$ $+Y(k)S(k)Y^T(k)$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2+n)$
$O^{-1}(k)$	$O^{-1}(k+1)$	$(n\times n)^{-1}$	$\frac{1}{6}(16n^3-3n^2-n)$
$O^{-1}(k)Y(k)$	$O^{-1}(k+1)Y(k)$	$(n\times n)\cdot(n\times r)$	$2n^2r-nr$
$Y(k+1)=[F_nO_n^{-1}]$ $O^{-1}(k)Y(k)$	$Y(k+1)=[F_nO_n^{-1}]$ $O^{-1}(k+1)Y(k)$	$(n\times n)\cdot(n\times r)$	$2n^2r-nr$
$O^{-1}(k+1)$	$O^{-1}(k)$	$(n\times n)^{-1}$	$\frac{1}{6}(16n^3-3n^2-n)$
$O^{-1}(k+1)Y(k)S(k)$	$O^{-1}(k)Y(k)S(k)$	$(n\times n)\cdot(n\times r)$	$2n^2r-nr$
$S(k)Y^T(k)$ $O^{-1}(k+1)Y(k)S(k)$	$S(k)Y^T(k)$ $O^{-1}(k)Y(k)S(k)$	$(r\times n)\cdot(n\times r)$ symmetric	$nr^2+nr-\frac{1}{2}(r^2+r)$
$S(k+1)=S(k)$ $-S(k)Y^T(k)$ $O^{-1}(k+1)Y(k)S(k)$	$S(k+1)=S(k)$ $+S(k)Y^T(k)$ $O^{-1}(k)Y(k)S(k)$	$(r\times r)+(r\times r)$ symmetric	$\frac{1}{2}(r^2+r)$
$P(k+1/k+1)=P(k/k)$ $+Y(k)S(k)Y^T(k)$	$P(k+1/k+1)=P(k/k)$ $+Y(k)S(k)Y^T(k)$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2+n)$
CTARE1	CTARE2	Total	$\frac{1}{6}(32n^3-3n^2+n)$ $+3nr^2-2nr+7n^2r$

Chandrasekhar Type Algorithm – Lyapunov equation (CTALE)

Matrix Operation	Matrix Dimensions	Calculation Burden
$Y(k+1)=F_n Y(k)$	$(n\times n)\cdot(n\times r)$	$2n^2r-nr$
$Y(k)Y^T(k)$	$(n\times r)\cdot(r\times n)$ symmetric	$n^2r+nr-\frac{1}{2}(n^2+n)$
$P(k+1/k+1)=$ $P(k/k)+Y(k)Y^T(k)$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2+n)$
CTALE	Total	$3n^2r$

Received: April, 2010