Contemporary Engineering Sciences, Vol. 2, 2009, no. 7, 333 - 354

Discrete Time Riccati Equation

Recursive Multiple Steps Solutions

Nicholas Assimakis

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

Abstract

Recursive per multiple steps algorithms for solving the discrete time Riccati and Lyapunov equations emanating from Kalman and Lainiotis filter are developed, for the case where the plant and measurement noise covariance are positive definite matrices. The resulting algorithms preserve the structure of the corresponding recursive per step algorithms and are faster than them, especially for high desired accuracy.

Keywords: Riccati equation, Kalman filter, Lainiotis filter, recursive algorithms

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 for $k \geq 0$:

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

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

where $x(k)$ is the $nx1$ dimensional state vector at time k, $z(k)$ is the $mx1$ 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, $Q(k)$ and $R(k)$ 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 Kalman filter [1], [4] and Lainiotis filter [5] are the most well known algorithms that solve the filtering problem. Note that the Kalman filter computes the estimation $x(L/L)$ through the prediction $x(L+1/L)$.

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 exist a steady state value $\overline{P}_e$ of the estimation error covariance matrix and a steady state value $\overline{P}_p$ of the prediction error covariance matrix.

The steady state prediction error covariance matrix can be calculated by solving the corresponding discrete time Riccati equation emanating from Kalman filter (REKF):

$$P(k+1/k) = c_{KF} + a_{KF}[I + P(k/k-1)b_{KF}]^{-1}P(k/k-1)a_{KF}^T \tag{3}$$

with

$$a_{KF} = F \tag{4}$$

$$b_{KF} = H^T R^{-1} H \tag{5}$$

$$c_{KF} = Q \tag{6}$$

The steady state solution $\overline{P}_p$ of the REKF is calculated by recursively implementing (3) for $k = 0,1,\ldots$, with initial condition $P(0/-1) = P_0$, until the following convergence criterion is satisfied: $\left\|P(k+1/k) - P(k/k-1)\right\| \le \varepsilon$, where $\left\|\cdot\right\|$ 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 $\overline{P}_p = \lim_{k\to\infty} P(k/k-1)$ of REKF is independent of the initial condition [1].

In the sequel we assume zero initial condition $P(0/-1) = 0$, i.e. $P_0 = 0$.

The steady state estimation error covariance matrix can be calculated by solving the corresponding discrete time Riccati equation emanating from Lainiotis filter (RELF):

$$P(k+1/k+1) = c_{LF} + a_{LF}[I + P(k/k)b_{LF}]^{-1}P(k/k)a_{LF}^T \tag{7}$$

with

$$a_{LF} = (I - QH^T AH)F \tag{8}$$

$$b_{LF} = F^T H^T AHF \tag{9}$$

$$c_{LF} = (I - QH^T AH)Q \tag{10}$$

where

$$A = [HQH^T + R]^{-1} \tag{11}$$

The steady state solution $\overline{P}_e$ of RELF is calculated by recursively implementing (7) for $k = 0,1,\ldots$, with initial condition $P(0/0) = P_0 - P_0 H^T [HP_0 H^T + R]^{-1} HP_0$.

The steady state or limiting solution $\overline{P}_e = \lim_{k\to\infty} P(k/k)$ of RELF is independent of the initial condition. We assume zero initial condition $P(0/0) = 0$, i.e. $P_0 = 0$.

The steady state prediction and estimation error covariance matrices are connected through the following relationships:

$$\overline{P}_e = \overline{P}_p - \overline{P}_p H^T [H\overline{P}_p H^T + R]^{-1} H\overline{P}_p \tag{12}$$

and

$$\overline{P}_p = Q + F\overline{P}_e F^T \tag{13}$$

The discrete time Riccati equation has attracted enormous attention. In view of the importance of the Riccati equation, there exists considerable literature on its recursive solutions [1], [3], [5], concerning per step or doubling algorithms.

In this paper we develop recursive per multiple steps algorithms. The case where the plant and measurement noise covariances are positive definite matrices is examined. The paper is organized as follows: In section 2 recursive per step algorithms are presented. In section 3 new recursive per multiple steps algorithms are presented. These algorithms correspond to the per step algorithms and preserve their structure. In section 4 the case of infinite measurement noise covariance matrix is examined; then Riccati equation takes the form of Lyapunov equation. In section 5 the algorithms' computational requirements are established. In section 6 comparisons are carried out. It is pointed out that the proposed per multiple steps algorithms are faster than the corresponding per step algorithms, for high desired accuracy.

2 Per step algorithms

From (3) and (7), it is obvious that the Riccati equations emanating from Kalman and Lainiotis filters are of equivalent form.

The existence of $b_{KF} = H^T R^{-1} H$ which occurs in REKF (3) and the existence of $A = [HQH^T + R]^{-1}$ which occurs in RELF (7), are 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.

Furthermore, we are able to rewrite (3) as:

$$P(k+1/k) = c_{KF} + a_{KF}\left[[P(k/k-1)]^{-1} + b_{KF}\right]^{-1} a_{KF}^T \tag{14}$$

if the inverse of $P(k/k-1)$ exists. It is easy to prove by induction that $P(k/k-1) > 0$, if $P(1/0) = Q$ is positive definite ($Q > 0$).

We are also able to rewrite (7) as:

$$P(k+1/k+1) = c_{LF} + a_{LF}\left[[P(k/k)]^{-1} + b_{LF}\right]^{-1} a_{LF}^T \tag{15}$$

if the inverse of $P(k/k)$ exists. It is easy to prove by induction that $P(k/k) > 0$, if $P(1/1) = c_{LF} = Q - QH^T AHQ$ is positive definite. This is true when Q is

positive definite ($Q > 0$). Thus, the nonsingular plant noise covariances matrix case is assumed in the sequel.

Per Step Algorithm 1

Thus, in the case where the plant and measurement noise covariances are positive definite matrices, it is obvious that the Riccati equation (14) emanating from Kalman filter and the Riccati equation (15) emanating from Lainiotis filter are both of the following equivalent form:

$$P_{k+1} = c + a[P_k^{-1} + b]^{-1} a^T \tag{16}$$

Concerning the Riccati equation emanating from Kalman filter, the prediction error covariance matrix is $P_k = P(k/k-1)$, the Kalman filter parameters are

$$a = F$$
$$b = H^T R^{-1} H$$
$$c = Q$$

and the steady state prediction error covariance matrix is $\overline{P} = \overline{P_p} = \lim_{k\to\infty} P_k$.

Concerning the Riccati equation emanating from Lainiotis filter, the prediction error covariance matrix is $P_k = P(k/k)$, the Kalman filter parameters are

$$a = (I - QH^T[HQH^T + R]^{-1} H)F$$
$$b = F^T H^T [HQH^T + R]^{-1} HF$$
$$c = (I - QH^T[HQH^T + R]^{-1} H)Q$$

and the steady state prediction error covariance matrix is $\overline{P} = \overline{P_e} = \lim_{k\to\infty} P_k$.

From the direct implementation of (16) we derive Per Step Algorithm 1:

PSA1

$$P_{k+1} = c + a\left[P_k^{-1} + b\right]^{-1} a^T$$
$$P_1 = c$$
$$\overline{P} = \lim_{k\to\infty} P_k$$

Per Step Algorithm 2

Setting

$$\pi_k = P_k^{-1} \tag{17}$$

in (16) and using the matrix inversion lemma:

$$[A + BCD]^{-1} = A^{-1} - A^{-1}B[C^{-1} + DA^{-1}B]DA^{-1} \tag{18}$$

we derive [3] Per Step Algorithm 2:

PSA2

$$\pi_{k+1} = \gamma - \alpha\left[\pi_k + \beta\right]^{-1} \alpha^T$$
$$\alpha = c^{-1} a$$
$$\beta = b + a^T c^{-1} a$$
$$\gamma = c^{-1}$$

$$\pi_1 = c^{-1}$$
$$\overline{\Pi} = \lim_{k\to\infty} \pi_k$$
$$\overline{P} = \overline{\Pi}^{-1}$$

Per Step Algorithm 3
Setting
$$\lambda_k = \pi_k + \beta \tag{19}$$
in Per Step Algorithm 2 we derive [3] Per Step Algorithm 3:
PSA3
$$\lambda_{k+1} = \ell - \alpha\lambda_k^{-1}\alpha^T$$
$$\alpha = c^{-1}a$$
$$\beta = b + a^T c^{-1} a$$
$$\gamma = c^{-1}$$
$$\ell = \beta + \gamma$$
$$\lambda_1 = \ell$$
$$\overline{\Lambda} = \lim_{k\to\infty} \lambda_k$$
$$\overline{P} = [\overline{\Lambda} - \beta]^{-1}$$

Comments.
1. All per step algorithms presented in this section require positive definite plant and measurement noise covariances matrices.
2. The Riccati equations emanating from Kalman and Lainiotis filters are of equivalent form and the derived per step algorithms are also of equivalent form.
3. All per step algorithms do converge to the steady state solutions and are stable; the convergence and stability properties of the per step algorithms are examined in [3].
4. The implementation of all the per step algorithms require the same number of recursions, depending on the desired accuracy.

3 Per multiple steps algorithms

In this section we propose recursive per multiple steps algorithms. The basic idea is to implement the per step algorithms of the previous section for multiple steps recursions. The resulting per multiple steps algorithms have the same structure with the corresponding per step algorithms.
First the per two steps algorithms are derived.

Per Two Steps Algorithm 1
The direct implementation of Per Step Algorithm 1 for two steps gives:

$$P_{k+2} = c + a\left[P_{k+1}^{-1} + b \right]^{-1} a^T$$

$$= c + a\left[\left[c + a\left[P_k^{-1} + b \right]^{-1} a^T \right]^{-1} + b \right]^{-1} a^T$$

$$= c + a\left[c^{-1} - c^{-1} a \left[P_k^{-1} + b + a^T c^{-1} a \right]^{-1} a^T c^{-1} + b \right]^{-1} a^T$$

$$= c + a(I + bc)^{-1} ca^T + a(I + bc)^{-1} a\left[P_k^{-1} + b + a^T b(I + bc)^{-1} a \right]^{-1} a^T (I + bc)^{-1} a^T$$

Thus, from the direct implementation of Per Step Algorithm 1 for two steps, we derive Per Two Step Algorithm 1:

P2SA1

$$P_{k+2} = c_2 + a_2\left[P_k^{-1} + b_2 \right]^{-1} a_2^T$$

$$a_2 = a(I + bc)^{-1} a$$

$$b_2 = b + a^T b(I + bc)^{-1} a$$

$$c_2 = c + a(I + bc)^{-1} ca^T$$

$$P_1 = c$$

$$\overline{P} = \lim_{k \to \infty} P_k$$

Per Two Steps Algorithm 2

The direct implementation of Per Step Algorithm 2 for two steps gives:

$$\pi_{k+2} = \gamma - \alpha\left[\pi_{k+1} + \beta \right]^{-1} \alpha^T$$

$$= \gamma - \alpha\left[\gamma - \alpha\left[\pi_k + \beta \right]^{-1} \alpha^T + \beta \right]^{-1} \alpha^T$$

$$= \gamma - \alpha\left[(\beta + \gamma)^{-1} - (\beta + \gamma)^{-1} \alpha\left[-(\pi_k + \beta) + \alpha^T (\beta + \gamma)^{-1} \alpha \right]^{-1} \alpha^T (\beta + \gamma)^{-1} \right] \alpha^T$$

$$= \gamma - \alpha(\beta + \gamma)^{-1} \alpha^T - \alpha(\beta + \gamma)^{-1} \alpha\left[\pi_k + \beta - \alpha^T (\beta + \gamma)^{-1} \alpha \right]^{-1} \alpha^T (\beta + \gamma)^{-1} \alpha^T$$

Thus, from the direct implementation of Per Step Algorithm 2 for two steps, we derive Per Two Step Algorithm 2:

P2SA2

$$\pi_{k+2} = \gamma_2 - \alpha_2\left[\pi_k + \beta_2 \right]^{-1} \alpha_2^T$$

$$\alpha = c^{-1} a$$

$$\beta = b + a^T c^{-1} a$$

$$\gamma = c^{-1}$$

$$\ell = \beta + \gamma$$

$$\alpha_2 = \alpha \ell^{-1} \alpha$$

$$\beta_2 = \beta - \alpha^T \ell^{-1} \alpha$$

$$\gamma_2 = \gamma - \alpha \ell^{-1} \alpha^T$$

$$\pi_1 = c^{-1}$$

$$\overline{\Pi} = \lim_{k\to\infty} \pi_k$$
$$\overline{P} = \overline{\Pi}^{-1}$$

Per Two Steps Algorithm 3

The direct implementation of Per Step Algorithm 3 for two steps gives:

$$\lambda_{k+2} = \ell - \alpha\lambda_{k+1}^{-1}\alpha^T$$
$$= \ell - \alpha\left[\ell - \alpha\lambda_k^{-1}\alpha^T\right]^{-1}\alpha^T$$
$$= \ell - \alpha\left[\ell^{-1} - \ell^{-1}\alpha\left[-\lambda_k + \alpha^T\ell^{-1}\alpha\right]^{-1}\alpha^T\ell^{-1}\right]\alpha^T$$
$$= \ell - \alpha\ell^{-1}\alpha^T - \alpha\ell^{-1}\alpha\left[\lambda_k - \alpha^T\ell^{-1}\alpha\right]^{-1}\alpha^T\ell^{-1}\alpha^T$$

Setting

$$\xi_k = \lambda_k - \alpha^T\ell^{-1}\alpha \tag{20}$$

in the last equation, we derive Per Two Step Algorithm 3:

P2SA3

$$\xi_{k+2} = \ell_2 - \alpha_2\xi_k^{-1}\alpha_2^T$$
$$\alpha = c^{-1}a$$
$$\beta = b + a^Tc^{-1}a$$
$$\gamma = c^{-1}$$
$$\ell = \beta + \gamma$$
$$\alpha_2 = \alpha\ell^{-1}\alpha$$
$$\ell_2 = \ell - \alpha\ell^{-1}\alpha^T - \alpha^T\ell^{-1}\alpha$$
$$\xi_1 = \ell - \alpha^T\ell^{-1}\alpha$$
$$\overline{\Xi} = \lim_{k\to\infty} \xi_k$$
$$\overline{P} = [\overline{\Xi} + \alpha^T\ell^{-1}\alpha - \beta]^{-1}$$

Comments.

1. All per two steps algorithms presented in this section require positive definite plant and measurement noise covariances matrices.
2. The Riccati equations emanating from Kalman and Lainiotis filters are of equivalent form and the derived per two steps algorithms are also of equivalent form.
3. The developed recursive per two steps algorithms preserve the structure of the corresponding recursive per step algorithms: in fact we can see the per two step algorithms as the corresponding per step algorithms with different initial conditions. So, the convergence and stability properties of the proposed per two steps algorithms are the same with those of the per step algorithms.
4. The implementation of all the per two steps algorithms require the same number of recursions. It is obvious that the number of recursions that the per two

steps algorithms execute equals to the one half of the number of recursions that the per step algorithms execute.

Per ν Steps Algorithms

It is obvious that we are able to derive per multiple steps algorithms. All these per ν steps algorithms PνSA1, PνSA2 and PνSA3 have a recursive part and a non-recursive part.

Concerning the recursive part, the per ν steps algorithms preserve the recursive structure of the corresponding recursive per step algorithms. It is clear that the number of recursions that the per ν steps algorithms execute equals to the number of recursions that the per step algorithms execute divided by ν.

Concerning the non-recursive part the per ν steps algorithms require constant calculations needed to be performed once for the Kalman/Lainiotis filter parameter definition, the initialization process and the final result extraction. The per ν steps algorithms preserve the structure of the initialization process of the corresponding per two steps algorithms and require the repetition of the initialization process of the corresponding per 2 steps algorithms for $\nu - 1$ times.

4 Infinite measurement noise covariance matrix case

In the case where the measurement noise covariance matrix is infinite ($R = \infty$), the Riccati equation takes the form of Lyapunov equation.

In fact, when $R = \infty$, from REKF (3) we derive the Lyapunov equation emanating from Kalman filter (LEKF):

$$P(k+1/k) = c_{KF} + a_{KF} P(k/k-1) a_{KF}^T \tag{21}$$

and from RELF (7) we derive the Lyapunov equation emanating from Lainiotis filter (LELF):

$$P(k+1/k+1) = c_{LF} + a_{LF} P(k/k) a_{LF}^T \tag{22}$$

where

$$a_{KF} = a_{LF} = F \tag{23}$$

$$c_{KF} = c_{LF} = Q \tag{24}$$

Note that from the Kalman filter equations it is clear [1] that the prediction error covariace matrix is equal to the estimation error covariance matrix:

$$P(k+1/k) = P(k+1/k+1) \tag{25}$$

It is also clear that the steady state prediction error covarianve matrix coincides with the steady state estimation error covarianve matrix:

$$\overline{P_p} = \overline{P_e} \tag{26}$$

Thus, in the case where the measurement noise covariance matrix is infinite, it becomes obvious that the Lyapunov equation (21) emanating from Kalman filter and the Lyapunov equation (22) emanating from Lainiotis filter are both of the following equivalent form:

$$P_{k+1} = c + aP_k a^T \tag{27}$$

Concerning the Lyapunov equation emanating from Kalman filter, the prediction error covariance matrix is $P_k = P(k/k-1)$, the Kalman filter parameters are

$a = F$
$c = Q$

and the steady state prediction error covariance matrix is $\overline{P} = \overline{P_p} = \lim_{k\to\infty} P_k$.

Concerning the Lyapunov equation emanating from Lainiotis filter, the prediction error covariance matrix is $P_k = P(k/k)$, the Kalman filter parameters are

$a = F$
$c = Q$

and the steady state prediction error covariance matrix is $\overline{P} = \overline{P_e} = \lim_{k\to\infty} P_k$.

Per Step Algorithm

The Per Step Algorithm for the Lyapunov equation consists of the direct implementation of (27):

PSALE

$P_{k+1} = c + aP_k a^T$
$P_1 = c$
$\overline{P} = \lim_{k\to\infty} P_k$

Per Two Steps Algorithm

The direct implementation of (27) for two steps gives:

$$P_{k+2} = c + aP_{k+1}a^T = c + a\left[c + aP_k a^T\right]a^T = c + aca^T + aaP_k a^T a^T$$

The Per Step Algorithm for the Lyapunov equation consists of the direct implementation of (27) for two steps:

P2SALE

$P_{k+2} = c_2 + a_2 P_k a_2^T$
$a_2 = aa$
$c_2 = c + aca^T$
$P_1 = c$
$\overline{P} = \lim_{k\to\infty} P_k$

Per Three Steps Algorithm

The direct implementation of (27) for three steps gives:

$$P_{k+3} = c + aP_{k+2}a^T = c + a\left[c + aP_{k+1}a^T\right]a^T$$

$$= c + a\left[c + a\left[c + aP_k a^T\right]a^T\right]a^T = c + aca^T + aaca^T a^T + aaaP_k a^T a^T a^T$$

The Per Step Algorithm for the Lyapunov equation consists of the direct

implementation of (27) for three steps:

P3SALE

$$P_{k+3} = c_3 + a_3 P_k a_3^T$$

$$a_3 = aaa = aa_2$$

$$c_3 = c + aca^T + aaca^T a^T = c + ac_2 a^T$$

$$P_1 = c$$

$$\overline{P} = \lim_{k\to\infty} P_k$$

Per ν Steps Algorithm

It is obvious that we are able to derive per multiple steps algorithms. The per ν steps algorithm PνSALE has a recursive part and a non-recursive part.

Concerning the recursive part, the per ν steps algorithm PνSALE preserves the recursive structure of the corresponding recursive per step algorithm. It is clear that the number of recursions that the per ν steps algorithm executes equals to the number of recursions that the per step algorithm executes divided by ν.

Concerning the non-recursive part the per ν steps algorithm requires constant calculations needed to be performed once for the initialization process). The per ν steps algorithm preserve the structure of the initialization process of the corresponding per two steps algorithm and requires the repetition of the initialization process of the corresponding per 2 steps algorithm for $\nu - 1$ times.

Comments.

1. The per step algorithm as well as the per multiple steps algorithms presented in this section require infinite measurement noise covariance matrix and solve the Lyapunov equation.
2. The Lyapunov equations emanating from Kalman and Lainiotis filters are of equivalent form and the derived algorithms are also of equivalent form.
3. The per step algorithm does converge to the steady state solution and is stable; the convergence and stability properties of the per step algorithm is examined in [4].
4. The developed recursive per multiple steps algorithms preserve the structure of the corresponding recursive per step algorithm: in fact we can see the per multiple steps algorithms as the corresponding per step algorithm with different initial conditions. So, the convergence and stability properties of the proposed per multiple steps algorithms are the same with those of the per step algorithm.
5. The initialization process of the per ν steps algorithm requires repetition of the initialization process of the per 2 steps algorithm.
6. The number of recursions that the per two steps algorithm executes equals to the one half of the number of recursions that the per step algorithm executes. The number of recursions that the per three steps algorithm executes equals to the one third of the number of recursions the per step algorithm executes. The number of recursions that the per ν steps algorithms execute equals to the number of recursions that the per step algorithms execute divided by ν.

5 Computational requirements

All algorithms presented in sections 3 and 4 have a recursive part and a non-recursive part. Due to the fact that all algorithms are recursive, the total calculation time of each is equal to:

$$t(alg) = [CB_C(alg) + CB_R(alg) \cdot N_\nu(alg)] \cdot t_{op} \quad (28)$$

where t_{op} is the time required to perform a scalar addition operation, $CB_C(alg)$ is the constant calculation burden of the calculations needed to be performed once for the Kalman/Lainiotis filter parameter definition, the initialization process and the final result extraction), $CB_R(alg)$ is the per recursion calculation burden, $N_\nu(alg)$ is the number of recursions that each per step or per multiple (ν) steps algorithm executes in order to reach the steady state solution of the Riccati or the Lyapunov equation. Note that, if the number of recursions that each per step algorithm executes equals to:

$$N_1(alg) = N \quad (29)$$

then the number of recursions that each per ν steps algorithm executes equals to:

$$N_\nu(alg) = N / \nu \quad (30)$$

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 Riccati equation solution recursive algorithms. Table 1 summarizes the calculation burden of needed matrix operations.

Table 1. 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 constant computational requirements of all per step and per multiple steps algorithms for solving the Riccati equation are summarized in Table 2. The details are given in the Appendix.

The calculation burdens for the constant calculations needed to be performed once include:

- the Kalman/Lainiotis Filter Parameter Definition (FPD) calculation burden cb_{FPD} (i.e. the calculation burden required to compute the Kalman/Lainiotis

filter parameters a, b, c) which depends on the state dimension n as well as on the measurement dimension m

- the calculation burden required for the initialization process and the final result extraction, which depends on the state dimension n.

Table 2. Constant Calculation Burden
of Riccati equation solution algorithms

Algorithm	Filter Parameter Definition	Initialization Process	Final Result Extraction
PSA1	cb_{FPD}		
PSA2	cb_{FPD}	$\frac{1}{6}(34n^3 - 3n^2 - n)$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
PSA3	cb_{FPD}	$\frac{1}{6}(34n^3 + 2n)$	$\frac{1}{6}(16n^3 + 2n)$
P2SA1	cb_{FPD}	$\frac{1}{6}(100n^3 - 15n^2 - n)$	
P2SA2	cb_{FPD}	$\frac{1}{6}(98n^3 - 9n^2 + n)$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
P2SA3	cb_{FPD}	$\frac{1}{6}(98n^3 - 9n^2 + n)$	$\frac{1}{6}(16n^3 + 3n^2 + 5n)$
PνSA1	cb_{FPD}	$(\nu - 1)\cdot(100n^3 - 15n^2 - n)$	
PνSA2	cb_{FPD}	$(\nu - 1)\cdot\frac{1}{6}(98n^3 - 9n^2 + n)$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
PνSA3	cb_{FPD}	$(\nu - 1)\cdot\frac{1}{6}(98n^3 - 9n^2 + n)$	$\frac{1}{6}(16n^3 + 3n^2 + 5n)$

where

$$cb_{FPD} = \begin{cases} cb_{KF} = n^2 m + 2nm^2 - \frac{1}{2}(n^2 + n) + \frac{1}{6}(16m^3 - 3m^2 - m) & \textit{Kalman filter} \\ cb_{LF} = 3n^2 m + 3nm^2 + \frac{1}{2}(16n^3 - 5n^2 - n) + \frac{1}{6}(16m^3 - 3m^2 - m) & \textit{Lainiotis filter} \end{cases}$$

The recursive computational requirements of all per step and per multiple steps algorithms for solving the Riccati equation are summarized in Table 3. The details are given in the Appendix.

The per recursion calculation burdens depend only on the state dimension n.

The initialization process of the per ν steps algorithms requires repetition of the initialization process of the corresponding per two steps algorithms for $\nu - 1$ times.

Table 3. Recursive Calculation Burden
of Riccati equation solution algorithms

Algorithm	Per Recursion Calculation Burden	Number of Recursions
PSA1	$\frac{1}{6}(50n^3 - 3n^2 + n)$	N
PSA2	$\frac{1}{6}(34n^3 + 2n^2)$	N
PSA3	$\frac{1}{6}(34n^3 - 3n^2 - n)$	N
P2SA1	$\frac{1}{6}(50n^3 - 3n^2 + n)$	$N/2$
P2SA2	$\frac{1}{6}(34n^3 + 2n^2)$	$N/2$
P2SA3	$\frac{1}{6}(34n^3 - 3n^2 - n)$	$N/2$
PνSA1	$\frac{1}{6}(50n^3 - 3n^2 + n)$	N/ν
PνSA2	$\frac{1}{6}(34n^3 + 2n^2)$	N/ν
PνSA3	$\frac{1}{6}(34n^3 - 3n^2 - n)$	N/ν

The computational requirements of all per step and per multiple steps algorithms for solving the Lyapunov equation are summarized in Table 4. The details are given in the Appendix.
The initialization process of the per ν steps algorithms requires repetition of the initialization process of the corresponding per two steps algorithms for $\nu - 1$ times.

Table 4. Calculation Burden of Lyapunov equation solution algorithms

Algorithm	Constant Calculation Burden (Initialization Process)	Per Recursion Calculation Burden	Number of Recursions
PSALE		$3n^3$	N
P2SALE	$5n^3 - n^2$	$3n^3$	$N/2$
P3SALE	$2\cdot(5n^3 - n^2)$	$3n^3$	$N/3$
PνSALE	$(\nu - 1)\cdot(5n^3 - n^2)$	$3n^3$	N/ν

6 Algorithms comparison

From Table 2, we derive the following conclusions concerning the Riccati equation solution algorithms:
1. The implementation of all the per step algorithms require the same number of recursions.
2. The implementation of all the per two steps algorithms require the same number of recursions. The number of recursions that the per two steps algorithms execute equals to the one half of the number of recursions that the per step

algorithms execute. The number of recursions that each per ν steps algorithm executes equals to the number of recursions that the per step algorithms execute divided by ν.

3. Concerning the recursive calculation burdens, PSA3 is the fastest per step algorithm, P2SA3 is the fastest per two steps algorithm and PνSA3 is the fastest per ν step algorithm:

$CB_R(PSA1) > CB_R(PSA2) > CB_R(PSA3)$

$CB_R(P2SA1) > CB_R(P2SA2) > CB_R(P2SA3)$

$CB_R(P\nu SA1) > CB_R(P\nu SA2) > CB_R(P\nu SA3)$

4. Concerning the calculation times, PSA3 is the fastest per step algorithm and P2SA3 is the fastest per two steps algorithm, for small N, i.e. for high desired accuracy:

$t(PSA1) > t(PSA2) > t(PSA3)$ for $N \geq 4$

$t(P2SA1) > t(P2SA2) > t(P2SA3)$ for $N \geq 4$

5. The per two steps algorithms are faster than the corresponding per step algorithms for small N, i.e. for high desired accuracy:

$t(PSA1) > t(P2SA1)$ for $N \geq 5$

$t(PSA2) > t(P2SA2)$ for $N \geq 4$

$t(PSA3) > t(P2SA3)$ for $N \geq 4$

6. The calculation burden of the initialization process increases as ν increases. Thus, the technique of multiple steps algorithms for large ν does not lead to algorithms faster than the per step algorithms.

From Table 3, we derive the following conclusions concerning the Lyapunov equation solution algorithms:

1. The number of recursions that each per ν steps algorithm executes equals to the number of recursions that the per step algorithms execute divided by ν.

2. All algorithms have the same recursive calculation burden.

3. The per two steps algorithm is faster than the corresponding per step algorithm for small N, i.e. for high desired accuracy. In fact:

$t(PSALE) > t(P2SALE)$ for $N \geq 4$, since

$t(PSALE) > t(P2SALE) \Rightarrow 3n^3N > 3n^3\frac{N}{2} + (5n^3 - n^2) \Rightarrow N > 2 \cdot \frac{5n-1}{3n} \Rightarrow N \geq 4$.

4. The per three steps algorithm is faster than the corresponding per step algorithm for small N, i.e. for high desired accuracy. In fact:

$t(PSALE) > t(P3SALE)$ for $N \geq 6$, since

$t(PSALE) > t(P3SALE) \Rightarrow 3n^3N > 3n^3\frac{N}{3} + 2 \cdot (5n^3 - n^2) \Rightarrow N > 3 \cdot \frac{5n-1}{3n} \Rightarrow N \geq 6$.

5. The calculation burden of the initialization process increases as ν increases. So, the per ν steps algorithm is faster than the corresponding per step algorithm for large N, i.e. for low desired accuracy. In fact:

$t(PSALE) > t(P\nu SALE)$ for $N > \nu \cdot \frac{5n-1}{3n} \Rightarrow N > \frac{5}{3} \cdot \nu$

Thus, the technique of multiple steps algorithms for large ν does not lead to algorithms faster than the per step algorithms.

Appendix. Calculation burdens of algorithms

A. Riccati equation solution algorithms

FPD		
Matrix Operation	Matrix Dimensions	Calculation Burden
$a_{KF} = F$		
$c_{KF} = Q$		
R^{-1}	$(m \times m)^{-1}$	$\frac{1}{6}(16m^3 - 3m^2 - m)$
$R^{-1}H$	$(m \times m) \cdot (m \times n)$	$2nm^2 - nm$
$b_{KF} = H^T R^{-1} H$	$(n \times m) \cdot (m \times n)$ symmetric	$n^2 m + nm - \frac{1}{2}(n^2 + n)$
Kalman FPD		$n^2 m + 2nm^2 - \frac{1}{2}(n^2 + n) + \frac{1}{6}(16m^3 - 3m^2 - m)$
HQ	$(m \times n) \cdot (n \times n)$	$2n^2 m - nm$
HQH^T	$(m \times n) \cdot (n \times m)$ symmetric	$nm^2 + nm - \frac{1}{2}(m^2 + m)$
$HQH^T + R$	$(m \times m) + (m \times m)$ symmetric	$\frac{1}{2}(m^2 + m)$
$A = (HQH^T + R)^{-1}$	$(m \times m)^{-1}$	$\frac{1}{6}(16m^3 - 3m^2 - m)$
AH	$(m \times m) \cdot (m \times n)$	$2nm^2 - nm$
$H^T AH$	$(n \times m) \cdot (m \times n)$ symmetric	$n^2 m + nm - \frac{1}{2}(n^2 + n)$
$QH^T AH$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$I - QH^T AH$	$(n \times n) + (n \times n)$	n
$c_{LF} = (I - QH^T AH)Q$	$(n \times n) \cdot (n \times n)$ symmetric	$n^3 + (\frac{1}{2}n^2 - \frac{1}{2}n)$
$a_{LF} = (I - QH^T AH)F$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$H^T AHF$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$b_{LF} = F^T H^T AHF$	$(n \times n) \cdot (n \times n)$ symmetric	$n^3 + (\frac{1}{2}n^2 - \frac{1}{2}n)$
Lainiotis FPD		$3n^2 m + 3nm^2 + \frac{1}{2}(16n^3 - 5n^2 - n) + \frac{1}{6}(16m^3 - 3m^2 - m)$

PSA1		
Matrix Operation	Matrix Dimensions	Calculation Burden
Filter Parameter Definition		cb_{PFD}
P_k^{-1}	$(n\times n)^{-1}$	$\frac{1}{6}(16n^3-3n^2-n)$
$P_k^{-1}+b$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2+n)$
$\left[P_k^{-1}+b\right]^{-1}$	$(n\times n)^{-1}$	$\frac{1}{6}(16n^3-3n^2-n)$
$a\left[P_k^{-1}+b\right]^{-1}$	$(n\times n)\cdot(n\times n)$	$2n^3-n^2$
$a\left[P_k^{-1}+b\right]^{-1}a^T$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3+\frac{1}{2}(n^2-n)$
$P_{k+1}=c+a\left[P_k^{-1}+b\right]^{-1}a^T$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2+n)$
Recursive Part		$\frac{1}{6}(50n^3-3n^2+n)$

PSA2		
Matrix Operation	Matrix Dimensions	Calculation Burden
Filter Parameter Definition		cb_{PFD}
$\gamma=\pi_1=c^{-1}$	$(n\times n)^{-1}$	$\frac{1}{6}(16n^3-3n^2-n)$
$\alpha=c^{-1}a$	$(n\times n)\cdot(n\times n)$	$2n^3-n^2$
$a^Tc^{-1}a$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3+\frac{1}{2}(n^2-n)$
$\beta=b+a^Tc^{-1}a$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2+n)$
Initialization Process		$\frac{1}{6}(34n^3-3n^2-n)$
$\pi_k+\beta$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2+n)$
$\left[\pi_k+\beta\right]^{-1}$	$(n\times n)^{-1}$	$\frac{1}{6}(16n^3-3n^2-n)$
$\alpha\left[\pi_k+\beta\right]^{-1}$	$(n\times n)\cdot(n\times n)$	$2n^3-n^2$
$\alpha\left[\pi_k+\beta\right]^{-1}\alpha^T$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3+\frac{1}{2}(n^2-n)$
$\pi_{k+1}=\gamma-\alpha\left[\pi_k+\beta\right]^{-1}\alpha^T$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2+n)$
Recursive Part		$\frac{1}{6}(34n^3+2n)$
$\overline{P}=\overline{\Pi}^{-1}$	$(n\times n)^{-1}$	$\frac{1}{6}(16n^3-3n^2-n)$
Final Result Extraction		$\frac{1}{6}(16n^3-3n^2-n)$

PSA3		
Matrix Operation	Matrix Dimensions	Calculation Burden
Filter Parameter Definition		cb_{PFD}
$\gamma = c^{-1}$	$(n\times n)^{-1}$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
$\alpha = c^{-1}a$	$(n\times n)\cdot(n\times n)$	$2n^3 - n^2$
$a^T c^{-1} a$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3 + \frac{1}{2}(n^2 - n)$
$\beta = b + a^T c^{-1} a$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2 + n)$
$\ell = \lambda_1 = \beta + \gamma$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2 + n)$
Initialization Process		$\frac{1}{6}(34n^3 + 2n)$
λ_k^{-1}	$(n\times n)^{-1}$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
$\alpha\lambda_k^{-1}$	$(n\times n)\cdot(n\times n)$	$2n^3 - n^2$
$\alpha\lambda_k^{-1}\alpha^T$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3 + \frac{1}{2}(n^2 - n)$
$\lambda_{k+1} = \ell - \alpha\lambda_k^{-1}\alpha^T$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2 + n)$
Recursive Part		$\frac{1}{6}(34n^3 - 3n^2 - n)$
$\overline{\Lambda} - \beta$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2 + n)$
$\overline{P} = [\overline{\Lambda} - \beta]^{-1}$	$(n\times n)^{-1}$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
Final Result Extraction		$\frac{1}{6}(16n^3 + 2n)$

P2SA1		
Matrix Operation	Matrix Dimensions	Calculation Burden
Filter Parameter Definition		cb_{PFD}
bc	$(n\times n)\cdot(n\times n)$	$2n^3-n^2$
$I+bc$	$(n\times n)+(n\times n)$	n
$(I+bc)^{-1}$	$(n\times n)^{-1}$	$\frac{1}{6}(16n^3-3n^2-n)$
$a(I+bc)^{-1}$	$(n\times n)\cdot(n\times n)$	$2n^3-n^2$
$a_2=a(I+bc)^{-1}a$	$(n\times n)\cdot(n\times n)$	$2n^3-n^2$
$b(I+bc)^{-1}$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3+\frac{1}{2}(n^2-n)$
$b(I+bc)^{-1}a$	$(n\times n)\cdot(n\times n)$	$2n^3-n^2$
$a^Tb(I+bc)^{-1}a$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3+\frac{1}{2}(n^2-n)$
$b_2=b+a^Tb(I+bc)^{-1}a$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2+n)$
$(I+bc)^{-1}c$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3+\frac{1}{2}(n^2-n)$
$a(I+bc)^{-1}c$	$(n\times n)\cdot(n\times n)$	$2n^3-n^2$
$a(I+bc)^{-1}ca^T$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3+\frac{1}{2}(n^2-n)$
$c_2=c+a(I+bc)^{-1}ca^T$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2+n)$
Initialization Process		$\frac{1}{6}(100n^3-15n^2-n)$
P_k^{-1}	$(n\times n)^{-1}$	$\frac{1}{6}(16n^3-3n^2-n)$
$P_k^{-1}+b_2$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2+n)$
$\left[P_k^{-1}+b_2\right]^{-1}$	$(n\times n)^{-1}$	$\frac{1}{6}(16n^3-3n^2-n)$
$a_2\left[P_k^{-1}+b_2\right]^{-1}$	$(n\times n)\cdot(n\times n)$	$2n^3-n^2$
$a_2\left[P_k^{-1}+b_2\right]^{-1}a_2^T$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3+\frac{1}{2}(n^2-n)$
$P_{k+2}=c_2+a_2\left[P_k^{-1}+b_2\right]^{-1}a_2^T$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2+n)$
Recursive Part		$\frac{1}{6}(50n^3-3n^2+n)$

P2SA2		
Matrix Operation	Matrix Dimensions	Calculation Burden
Filter Parameter Definition		cb_{PFD}
$\gamma = \pi_1 = c^{-1}$	$(n\times n)^{-1}$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
$\alpha = c^{-1}a$	$(n\times n)\cdot(n\times n)$	$2n^3 - n^2$
$a^T c^{-1} a$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3 + \frac{1}{2}(n^2 - n)$
$\beta = b + a^T c^{-1} a$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2 + n)$
$\ell = \beta + \gamma$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2 + n)$
ℓ^{-1}	$(n\times n)^{-1}$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
$\alpha\ell^{-1}$	$(n\times n)\cdot(n\times n)$	$2n^3 - n^2$
$\alpha_2 = \alpha\ell^{-1}\alpha$	$(n\times n)\cdot(n\times n)$	$2n^3 - n^2$
$\ell^{-1}\alpha$	$(n\times n)\cdot(n\times n)$	$2n^3 - n^2$
$\alpha^T\ell^{-1}\alpha$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3 + \frac{1}{2}(n^2 - n)$
$\beta_2 = \beta - \alpha^T\ell^{-1}\alpha$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2 + n)$
$\alpha\ell^{-1}\alpha^T$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3 + \frac{1}{2}(n^2 - n)$
$\gamma_2 = \gamma - \alpha\ell^{-1}\alpha^T$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2 + n)$
Initialization process		$\frac{1}{6}(98n^3 - 9n^2 + n)$
$\pi_k + \beta_2$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2 + n)$
$\left[\pi_k + \beta_2\right]^{-1}$	$(n\times n)^{-1}$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
$\alpha_2\left[\pi_k + \beta_2\right]^{-1}$	$(n\times n)\cdot(n\times n)$	$2n^3 - n^2$
$\alpha_2\left[\pi_k + \beta_2\right]^{-1}\alpha_2^T$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3 + \frac{1}{2}(n^2 - n)$
$\pi_{k+2} = \gamma_2 - \alpha_2\left[\pi_k + \beta_2\right]^{-1}\alpha_2^T$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2 + n)$
Recursive Part		$\frac{1}{6}(34n^3 + 2n)$
$\overline{P} = \overline{\Pi}^{-1}$	$(n\times n)^{-1}$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
Final Result Extraction		$\frac{1}{6}(16n^3 - 3n^2 - n)$

P2SA3		
Matrix Operation	Matrix Dimensions	Calculation Burden
Filter Parameter Definition		cb_{PFD}
$\gamma = c^{-1}$	$(n \times n)^{-1}$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
$\alpha = c^{-1}a$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$a^T c^{-1} a$	$(n \times n) \cdot (n \times n)$ symmetric	$n^3 + \frac{1}{2}(n^2 - n)$
$\beta = b + a^T c^{-1} a$	$(n \times n) + (n \times n)$ symmetric	$\frac{1}{2}(n^2 + n)$
$\ell = \beta + \gamma$	$(n \times n) + (n \times n)$ symmetric	$\frac{1}{2}(n^2 + n)$
ℓ^{-1}	$(n \times n)^{-1}$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
$\alpha \ell^{-1}$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$\alpha_2 = \alpha \ell^{-1} \alpha$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$\ell^{-1} \alpha$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$\alpha^T \ell^{-1} \alpha$	$(n \times n) \cdot (n \times n)$ symmetric	$n^3 + \frac{1}{2}(n^2 - n)$
$\xi_1 = \ell - \alpha^T \ell^{-1} \alpha$	$(n \times n) + (n \times n)$ symmetric	$\frac{1}{2}(n^2 + n)$
$\alpha \ell^{-1} \alpha^T$	$(n \times n) \cdot (n \times n)$ symmetric	$n^3 + \frac{1}{2}(n^2 - n)$
$\ell_2 = \ell - \alpha \ell^{-1} \alpha^T - \alpha^T \ell^{-1} \alpha$	$(n \times n) + (n \times n)$ symmetric	$\frac{1}{2}(n^2 + n)$
Initialization Process		$\frac{1}{6}(98n^3 - 9n^2 + n)$
ξ_k^{-1}	$(n \times n)^{-1}$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
$\alpha_2 \xi_k^{-1}$	$(n \times n) \cdot (n \times n)$	$2n^3 - n^2$
$\alpha_2 \xi_k^{-1} \alpha_2^T$	$(n \times n) \cdot (n \times n)$ symmetric	$n^3 + \frac{1}{2}(n^2 - n)$
$\xi_{k+2} = \ell_2 - \alpha_2 \xi_k^{-1} \alpha_2^T$	$(n \times n) + (n \times n)$ symmetric	$\frac{1}{2}(n^2 + n)$
Recursive Part		$\frac{1}{6}(34n^3 - 3n^2 - n)$
$\alpha^T \ell^{-1} \alpha - \beta$	$(n \times n) + (n \times n)$ symmetric	$\frac{1}{2}(n^2 + n)$
$\overline{\Xi} + \alpha^T \ell^{-1} \alpha - \beta$	$(n \times n) + (n \times n)$ symmetric	$\frac{1}{2}(n^2 + n)$
$\overline{P} = [\overline{\Xi} + \alpha^T \ell^{-1} \alpha - \beta]^{-1}$	$(n \times n)^{-1}$	$\frac{1}{6}(16n^3 - 3n^2 - n)$
Final Result Extraction		$\frac{1}{6}(16n^3 + 3n^2 + 5n)$

B. Lyapunov equation solution algorithms

PSALE		
Matrix Operation	Matrix Dimensions	Calculation Burden
aP_k	$(n\times n)\cdot(n\times n)$	$2n^3-n^2$
$aP_k a^T$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3+\frac{1}{2}(n^2-n)$
$c+aP_k a^T$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2+n)$
Recursive Part		$3n^3$

P2SALE		
Matrix Operation	Matrix Dimensions	Calculation Burden
$a_2=aa$	$(n\times n)\cdot(n\times n)$	$2n^3-n^2$
ac	$(n\times n)\cdot(n\times n)$	$2n^3-n^2$
aca^T	$(n\times n)\cdot(n\times n)$ symmetric	$n^3+\frac{1}{2}(n^2-n)$
$c_2=c+aca^T$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2+n)$
Initialization Process		$5n^3-n^2$
a_2P_k	$(n\times n)\cdot(n\times n)$	$2n^3-n^2$
$a_2P_k a_2^T$	$(n\times n)\cdot(n\times n)$ symmetric	$n^3+\frac{1}{2}(n^2-n)$
$P_{k+2}=c_2+a_2P_k a_2^T$	$(n\times n)+(n\times n)$ symmetric	$\frac{1}{2}(n^2+n)$
Recursive Part		$3n^3$

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] R. E. Kalman, A new approach to linear filtering and prediction problems, J. Bas. Eng., Trans. ASME, 82 (1960), 34-45.

[5] D. G. Lainiotis, Discrete Riccati Equation Solutions: Partitioned Algorithms, IEEE Transactions on AC, AC-20 (1975), 555-556.

Received: January, 2009