Selection of the Fastest Solution of the Complex Riccati Equation

ATHANASIOS POLYZOS[1], CHRISTOS TSINOS[1], MARIA ADAM[2], NICHOLAS ASSIMAKIS[1]
[1]Department of Digital Industry Technologies,
National and Kapodistrian University of Athens,
34400 Psachna Evias,
GREECE

[2]Department of Computer Science and Biomedical Informatics,
University of Thessaly,
2-4 Papasiopoulou Str., 35131, Lamia,
GREECE

Abstract: - Complex Kalman filters are used to process complex signals that appear in various fields of science and engineering. The augmented complex Riccati equation is related to time-invariant augmented or widely linear models. The off-line solution of the Riccati equation is a prerequisite for determining the steady-state augmented complex Kalman filter parameters before any measurements are observed. Iterative per-step and doubling algorithms for the solution of the complex Riccati equation are derived and their computational requirements are derived, leading to the ability to select the fastest iterative algorithm.

Key-Words: - Augmented or Widely Linear Model, Time-Invariant, Complex Kalman Filter, Steady-State, Riccati Equation, Doubling Algorithm.

Received: May 29, 2024. Revised: November 2, 2024. Accepted: December 3, 2024. Published: December 31, 2024.

1 Introduction

The celebrated Kalman filter has been applied in many fields of science and engineering, such as prediction and estimation applications to the aerospace industry, communication systems, filtering noise from images, power system, GPS position estimation, autonomous orbit determination, pose regularization, robotics, control effectiveness estimation on airplanes, applications with time-correlated measurement errors, aircraft state estimation, as referred in [1], [2], [3], [4], [5], [6], [7], [8], [9]. Complex Kalman filter has been used with success in applications with complex signals, such tracking, oceanography, array processing, communications, biomedicine, tracking for Global Navigation Satellite System (GNSS) meta-signals, power system frequency, unbalanced grids, sensor applications as referred in [10], [11], [12], [13], [14], [15].

It is known that a complex signal is characterized by the following statistical properties (a) the covariance, that captures the information corresponding to the total power of the signal and (b) the pseudo-covariance or complementary covariance, that characterizes the correlations between the real and imaginary parts of the signal, [11]. The traditional linear state space model is applicable only to proper (circular) signals as it takes into account the covariance matrix and ignores the pseudo-covariance matrix; then the conventional complex Kalman filter is derived. The augmented or widely linear model is applicable to improper (non-circular) complex signals as it takes into account both the covariance and pseudo-covariance matrices; then the augmented complex Kalman filter is derived.

Traditionally, Kalman filter computes the state estimation and prediction as well as the corresponding estimation and prediction error covariances using the observations. It has been noted [16] that the computations of prediction error covariances are independent of the observations and as such can be calculated a-priori, i.e. before any observation is available. Then the prediction error covariance can be iteratively derived by the Riccati recursion, [16].

In this paper we consider the time-invariant augmented or widely linear model, where all the model parameters are time constant. In this case, the augmented complex Riccati equation appears. In view of the importance of the real Riccati equation, there exists considerable literature on its iterative solutions, [1]. Following the ideas of solving the real Riccati equation [1] and the ideas of solving the

complex Lyapunov equation [17], we present iterative solutions of the complex Riccati equation.

The key contributions of the paper are:
(a) the derivation of iterative solutions (per step and doubling algorithms) for the solution of the complex Riccati equation
(b) the determination of the steady-state augmented complex Kalman filter parameters
(c) the derivation of the computational requirements of the iterative algorithms
(d) the development of a method to select the faster algorithm.

2 Augmented Complex Riccati Equation

The ***time-invariant*** **augmented or widely linear model** arises in linear estimation concerning complex signals and is described by the following state space equations, [16]:

$$x^a(k) = F^a x^a(k-1) + w^a(k) \quad (1)$$

$$z^a(k) = H^a x^a(k) + v^a(k) \quad (2)$$

where

- the $2n \times 1$ augmented state vector $x^a(k) = \begin{bmatrix} x(k) \\ \bar{x}(k) \end{bmatrix}$ consists of the n dimensional state vector $x(k)$ and its conjugate $\bar{x}(k)$
- the $2m \times 1$ augmented measurement vector $z^a(k) = \begin{bmatrix} z(k) \\ \bar{z}(k) \end{bmatrix}$ consists of the m dimensional measurement vector $z(k)$ and its conjugate $\bar{z}(k)$
- $F^a = \begin{bmatrix} F & A \\ \bar{A} & \bar{F} \end{bmatrix}$ is the augmented transition matrix
- $H^a = \begin{bmatrix} H & B \\ \bar{B} & \bar{H} \end{bmatrix}$ is the augmented output matrix
- the augmented state noise $w^a(k) = \begin{bmatrix} w(k) \\ \bar{w}(k) \end{bmatrix}$ consists of the n dimensional state noise vector $w(k)$ and its conjugate $\bar{w}(k)$; $w(k)$ is Gaussian with zero mean, covariance Q and pseudo-covariance U; $w^a(k)$ is non-circular Gaussian with zero mean and covariance $Q^a = \begin{bmatrix} Q & U \\ \bar{U} & \bar{Q} \end{bmatrix}$
- the augmented measurement noise $v^a(k) = \begin{bmatrix} v(k) \\ \bar{v}(k) \end{bmatrix}$ consists of the m dimensional state noise vector $v(k)$ and its conjugate $\bar{v}(k)$; $v(k)$ is Gaussian with zero mean, covariance R and pseudo-covariance V; $v^a(k)$ is non-circular Gaussian with zero mean and covariance $R^a = \begin{bmatrix} R & V \\ \bar{V} & \bar{R} \end{bmatrix}$
- the augmented initial state $x^a(0) = \begin{bmatrix} x(0) \\ \bar{x}(0) \end{bmatrix}$ consists of the initial state $x(0)$ and its conjugate $\bar{x}(0)$; $x(0)$ is Gaussian with mean x_0, covariance P_0 and pseudo-covariance Π_0; $x^a(0)$ is non-circular Gaussian with mean ${x_0}^a = \begin{bmatrix} x_0 \\ \bar{x}_0 \end{bmatrix}$ and covariance ${P_0}^a = \begin{bmatrix} P_0 & \Pi_0 \\ \bar{\Pi}_0 & \bar{P}_0 \end{bmatrix}$.

Note that $Q^a, R^a, {P_0}^a$ are Hermitian matrices. Note also that a covariance matrix M is a Hermitian matrix ($M^* = M$) and a pseudo-covariance matrix N is a symmetric matrix ($N^T = N$); T is the transpose and * is the conjugate transpose.

The augmented complex Kalman filter (ACKF) computes the augmented state prediction and estimation and the corresponding covariances, using the augmented Kalman filter gain, which is derived by minimizing the cost function based on the MSE criterion, [11].

The state prediction $x(k|k-1)$ has covariance matrix $P(k|k-1)$ and pseudo-covariance matrix $\Pi(k|k-1)$. The augmented state prediction $x^a(k|k-1)$ has covariance matrix $P^a(k|k-1) = \begin{bmatrix} P(k|k-1) & \Pi(k|k-1) \\ \bar{\Pi}(k|k-1) & \bar{P}(k|k-1) \end{bmatrix}$.

The state estimation $x(k|k)$ has covariance matrix $P(k|k)$ and pseudo-covariance matrix $\Pi(k|k)$. The augmented state estimation $x^a(k|k)$ has covariance matrix $P^a(k|k) = \begin{bmatrix} P(k|k) & \Pi(k|k) \\ \bar{\Pi}(k|k) & \bar{P}(k|k) \end{bmatrix}$.

Note that $P^a(k|k-1)$and $P^a(k|k)$ are Hermitian matrices.

The augmented Kalman filter gain is $K^a(k) = \begin{bmatrix} K(k) & G(k) \\ \bar{G}(k) & \bar{K}(k) \end{bmatrix}$.

For time-invariant systems, where the augmented model parameters F^a, H^a, Q^a, R^a are constant matrices, the Time Invariant Augmented Complex Kalman Filter is derived:

Time Invariant Augmented Complex Kalman Filter (TIACKF)
initial conditions
$x^a(0|-1)={x_0}^a$
$P^a(0|-1) = {P_0}^a$
iterations $k = 0,1, \dots$
$K^a(k) = P^a(k|k-1)H^{a*}[H^a P^a(k|k-1)H^{a*} + R^a]^{-1}$
$x^a(k|k) = [I^a - K^a(k)H^a]x^a(k|k-1) + K^a(k)z^a(k)$
$P^a(k|k) = [I^a - K^a(k)H^a]P^a(k|k-1)$
$x^a(k+1|k) = F^a x^a(k|k)$
$P^a(k+1|k) = Q^a + F^a P^a(k|k)F^{a*}$

Comment 1. The initial conditions are the statistics (mean, covariance) of the initial augmented state given no measurements.
Comment 2. The existence of the inverse of the matrices in the augmented Kalman filter gain equation is ensured assuming that R^a is positive definite (and hence invertible); this has the significance that no measurement is exact.

It is evident that computations of estimation and prediction error covariances and of the Kalman filter gain are independent of the observations. Then the prediction error covariance can be iteratively derived by the **Augmented Complex Riccati equation,** [16]:

$$\begin{aligned} P^a(k+1|k) = {} & Q^a + F^a P^a(k|k-1)F^{a*} \\ & -F^a P^a(k|k-1)H^{a*} \\ & [H^a P^a(k|k-1)H^{a*} + R^a]^{-1} \\ & H^a P^a(k|k-1)F^{a*} \end{aligned} \quad (3)$$

Note that assuming that R^a is positive definite and using the Matrix Inversion Lemma the Augmented Complex Riccati equation is rewritten as:

$$\begin{aligned} & P^a(k+1|k) = Q^a \\ & +F^a\left[P^{a-1}(k|k-1) + H^{a*}R^{a-1}H^a\right]^{-1}F^{a*} \end{aligned} \quad (4)$$

Note that $b^a = H^{a*}R^{a-1}H^a$ is a Hermitian matrix.

If the signal process model is asymptotically stable, then there exists a steady-state value P^a of the prediction error covariance matrix. This value remains constant after the steady-state time is reached. The steady-state prediction error covariance matrix satisfies the **Augmented Complex Algebraic Riccati equation**

$$\begin{aligned} & P^a = Q^a + F^a P^a F^{a*} \\ & -F^a P^a H^{a*}[H^a P^a H^{a*} + R^a]^{-1} H P^a F^{a*} \end{aligned} \quad (5)$$

which, assuming that R^a is positive definite, can be rewritten as:

$$P^a = Q^a + F^a\left[P^{a-1} + H^{a*}R^{a-1}H^a\right]^{-1}F^{a*} \quad (6)$$

Existence, localization and approximation of complex Riccati equation have been studied in [18]. Solutions of the complex non-symmetric algebraic Riccati equation are discussed in [19].

In the steady-state case, the Steady State Augmented Complex Kalman Filter is derived:

Steady State Augmented Complex Kalman Filter (SSACKF)
initial condition
$x^a(0|0) = x^a{}_0$
iterations $k = 1,2, \dots$
$x^a(k|k) = C^a x^a(k-1|k-1) + D^a z^a(k)$

Comment.
The steady-state coefficients
$C^a = [I^a - K^a H^a]F^a$
$D^a = K^a$
are calculated off-line by first solving the corresponding augmented complex Riccati equation and then and off-line computing the steady-state Kalman filter gain:
$K^a = P^a H^{a*}[H^a P^a H^{a*} + R^a]^{-1}$.

It is clear that the steady-state augmented complex Kalman filter parameters are determined via the solution of the augmented complex Riccati equation.

3 Iterative Per Step Algorithms

3.1 Direct Per Step Algorithm

The conventional solution of the complex Riccati equation is derived by iteratively implementing the Augmented Complex Riccati equation.

Assuming that R^a is positive definite (and hence invertible) and $P^a(0|-1) = 0$, which implies $P^a(1|0) = Q^a$, the direct use of the Augmented Complex Riccati equation leads to the direct per step algorithm:

Direct Per Step Algorithm
initialization
$b^a = H^{a*}R^{a-1}H^a$
initial condition
$P^a(1|0) = Q^a$
iterations $k = 1,2, \dots$
$P^a(k+1|k) = Q^a + F^a\left[P^{a-1}(k|k-1) + b^a\right]^{-1}F^{a*}$
until
$\|P^a(k+1|k) - P^a(k|k-1)\| < \varepsilon$
output
$P^a = \lim_{k\to\infty} P^a(k+1|k)$

3.2 Transformed Per Step Algorithm

Assuming that Q^a and R^a are positive definite (and hence invertible) and using the Matrix Inversion Lemma in the Augmented Complex Riccati equation we get:

$$\begin{aligned} & P^a(k+1|k) = Q^a \\ & +F^a\left[P^{a-1}(k|k-1) + H^{a*}R^{a-1}H^a\right]^{-1}F^{a*} \\ & \Rightarrow P^{a-1}(k+1|k) = Q^{a-1} \\ & -Q^{a-1}F^a \\ & \left[P^{a-1}(k|k-1) + H^{a*}R^{a-1}H^a + F^{a*}Q^{a-1}F^a\right]^{-1} \\ & F^{a*}Q^{a-1} \\ & \Rightarrow P^{a-1}(k+1|k) + H^{a*}R^{a-1}H^a + F^{a*}Q^{a-1}F^a \\ & = Q^{a-1} + H^{a*}R^{a-1}H^a + F^{a*}Q^{a-1}F^a \end{aligned}$$

$-Q^{a^{-1}}F^a$
$\left[P^{a^{-1}}(k|k-1)+H^{a^*}R^{a^{-1}}H^a+F^{a^*}Q^{a^{-1}}F^a\right]^{-1}$
$\left(Q^{a^{-1}}F^a\right)^*$

Then, setting
$L^a(k|k-1)=P^{a^{-1}}(k|k-1)$
$+H^{a^*}R^{a^{-1}}H^a+F^{a^*}Q^{a^{-1}}F^a$
$\Gamma^a=Q^{a^{-1}}+H^{a^*}R^{a^{-1}}H^a+F^{a^*}Q^{a^{-1}}F^a$
$\Phi^a=Q^{a^{-1}}F^a$
we are able to rewrite the augmented complex Riccati equation as:
$L^a(k+1|k)=\Gamma^a-\Phi^aL^{a^{-1}}(k|k-1)\Phi^{a^*}$

Then assuming $P^a(0|-1)=0$, which implies $P^a(1|0)=Q^a$, we derive the transformed per step algorithm:

Transformed Per Step Algorithm
initialization
$\Gamma^a=Q^{a^{-1}}+H^{a^*}R^{a^{-1}}H^a+F^{a^*}Q^{a^{-1}}F^a$
$\Phi^a=Q^{a^{-1}}F^a$
initial condition
$L^a(1|0)=\Gamma^a=Q^{a^{-1}}+H^{a^*}R^{a^{-1}}H^a+F^{a^*}Q^{a^{-1}}F^a$
iterations $k=1,2,\ldots$
$L^a(k+1|k)=\Gamma^a-\Phi^aL^{a^{-1}}(k|k-1)\Phi^{a^*}$
until
$\|L^a(k+1|k)-L^a(k|k-1)\|<\varepsilon$
output
$L^a=\lim_{k\to\infty}L^a(k+1|k)$
$P^a=\left\{L^a-\left(H^{a^*}R^{a^{-1}}H^a+F^{a^*}Q^{a^{-1}}F^a\right)\right\}^{-1}$

3.3 Dual Direct Per Step Algorithm

We are going to derive the iterative dual direct per step algorithm by using the duality concept [16]. In fact, for the $n\times 1$ complex vector $x=x^R+jx^I$, with x^R and x^I its real part and imaginary part, consider its dual $2n\times 1$ vector $x^d=\begin{bmatrix}x^R\\x^I\end{bmatrix}$.Then $x^a=J_nx^d$, where $J_n=\begin{bmatrix}I_n & jI_n\\ I_n & -jI_n\end{bmatrix}$ is of dimension $2n\times 2n$ and I_n is the $n\times n$ identity matrix. Note that the following property holds: $J_n^*=2J_n^{-1}=\begin{bmatrix}I_n & I_n\\ -jI_n & jI_n\end{bmatrix}$.

Furthermore, the dual covariance matrix is related to the $2n\times 2n$ augmented covariance matrix by:
$P^a=J_nP^dJ_n^*$

Also, the dual matrix is related to the $2m\times 2n$ augmented matrix by:
$M^a=J_mM^dJ_n^{-1}$
Then we have:
$F^a=J_nF^dJ_n^{-1}$
$H^a=J_mH^dJ_n^{-1}$
$Q^a=J_nQ^dJ_n^*$
$R^a=J_mR^dJ_m^*$

Assume that R^a and R^d are positive definite (and hence invertible). Then
$b^a=H^{a^*}R^{a^{-1}}H^a$
$b^d=H^{d^T}R^{d^{-1}}H^d$

Then
$b^a=H^{a^*}R^{a^{-1}}H^a$
$=\left(J_mH^dJ_n^{-1}\right)^*\left(J_mR^dJ_m^*\right)^{-1}\left(J_mH^dJ_n^{-1}\right)$
$=J_n^{-1^*}H^{d^*}J_m^*J_m^{*\,-1}R^{d^{-1}}J_m^{-1}J_mH^dJ_n^{-1}$
$=J_n^{-1^*}\left(H^{d^*}R^{d^{-1}}H^d\right)J_n^{-1}=J_n^{-1^*}\left(H^{d^T}R^{d^{-1}}H^d\right)J_n^{-1}$
$=J_n^{*\,-1}b^dJ_n^{-1}$

It is clear that
$b^a=J_n^{*\,-1}b^dJ_n^{-1}=\frac{1}{2}J_nb^dJ_n^{-1}$
and
$b^d=2J_n^{-1}b^aJ_n=J_n^*b^aJ_n$

Also, $P^a(k|k-1)=J_nP^d(k|k-1)J_n^*$

Note that F^d,H^d,Q^d,R^d,b^d and $P^d(k|k-1)$ are real matrices.

Note also that Q^d,R^d,b^d and $P^d(k|k-1)$ are real symmetric matrices.

Then, from the per step algorithm we get:
$P^a(k+1|k)=Q^a+F^a\left[P^{a^{-1}}(k|k-1)+b^a\right]^{-1}F^{a^*}$
$\Rightarrow J_nP^d(k+1|k)J_n^*=J_nQ^dJ_n^*$
$+J_nF^dJ_n^{-1}$
$\left[\left(J_nP^d(k|k-1)J_n^*\right)^{-1}+J_n^{*\,-1}b^dJ_n^{-1}\right]^{-1}$
$\left(J_nF^dJ_n^{-1}\right)^*=J_nQ^dJ_n^*+J_nF^dJ_n^{-1}$
$\left[J_n^{*\,-1}\left(P^d(k|k-1)\right)^{-1}J_n^{-1}+J_n^{*\,-1}b^dJ_n^{-1}\right]^{-1}$
$J_n^{*\,-1}F^{d^*}J_n^*$
$=J_nQ^dJ_n^*$
$+J_nF^dJ_n^{-1}J_n\left[\left(P^d(k|k-1)\right)^{-1}+b^d\right]^{-1}J_n^*J_n^{*\,-1}F^{d^*}J_n^*$
$=J_nQ^dJ_n^*+J_nF^d\left[\left(P^d(k|k-1)\right)^{-1}+b^d\right]^{-1}F^{d^*}J_n^*$
$\Rightarrow P^d(k+1|k)=Q^d$
$+F^d\left[\left(P^d(k|k-1)\right)^{-1}+b^d\right]^{-1}F^{d^T}$
since F^d is a real matrix.

Assuming that $P^a(0|-1)=0$, which implies $P^a(1|0)=Q^a$, we have derived the dual direct per step algorithm is derived:

Dual Direct Per Step Algorithm
initialization
$F^d = J_n^{-1} F^a J_n$
$H^d = J_m^{-1} H^a J_n$
$Q^d = J_n^{-1} Q^a J_n^{-*}$
$R^d = J_m^{-1} R^a J_m^{-*}$
$b^d = H^{d^T} R^{d^{-1}} H^d$
initial condition
$P^d(1|0) = J_n^{-1} Q J_n^{-*}$
iterations $k = 1,2,\ldots$
$P^d(k+1|k) = Q^d + F^d \left[P^{d^{-1}}(k|k-1) + b^d \right]^{-1} F^{d^T}$
until
$\left\| P^d(k+1|k) - P^d(k|k-1) \right\| < \varepsilon$
output
$P^d = \lim_{k\to\infty} P^d(k+1|k)$
$P^a = J_n P^d J_n^*$

3.4 Dual Transformed Per Step Algorithm

Using the duality concept, we get
$b^d = 2J_n^{-1} b^a J_n = J_n^* b^a J_n$
$\Phi^d = J_n^* \Phi^a J_n$
$\Gamma^d = J_n^* \Gamma^a J_n$

since
$b^a = H^{a^*} R^{a^{-1}} H^a$
$= \left(J_m H^d J_n^{-1}\right)^* \left(J_m R^d J_m^*\right)^{-1} \left(J_m H^d J_n^{-1}\right)$
$= J_n^{-1^*} H^{d^*} J_m^* J_m^{-*^{-1}} R^{d^{-1}} J_m^{-1} J_m H^d J_n^{-1}$
$= J_n^{-1^*} \left(H^{d^*} R^{d^{-1}} H^d\right) J_n^{-1} = J_n^{-1^*} \left(H^{d^T} R^{d^{-1}} H^d\right) J_n^{-1}$
$= J_n^{*^{-1}} b^d J_n^{-1}$
$\Phi^a = Q^{a^{-1}} F^a = \left(J_n Q^d J_n^*\right)^{-1} \left(J_n F^d J_n^{-1}\right)$
$= J_n^{-*} Q^{d^{-1}} J_n^{-1} J_n F^d J_n^{-1}$
$= J_n^{-*} Q^{d^{-1}} F^d J_n^{-1} = J_n^{-*} \Phi^d J_n^{-1}$
$\Gamma^a = Q^{a^{-1}} + H^{a^*} R^{a^{-1}} H^a + F^{a^*} Q^{a^{-1}} F^a$
$= Q^{a^{-1}} + b^a + F^{a^*} Q^{a^{-1}} F^a$
$= \left(J_n Q^d J_n^*\right)^{-1} + J_n^{-*} b^d J_n^{-1}$
$+ \left(J_n F^d J_n^{-1}\right)^* \left(J_n Q^d J_n^*\right)^{-1} \left(J_n F^d J_n^{-1}\right)$
$= J_n^{-*} Q^{d^{-1}} J_n^{-1} + J_n^{-*} b^d J_n^{-1}$
$+ J_n^{-*} F^{d^*} J_n^* J_n^{-*} Q^{d^{-1}} J_n^{-1} J_n F^d J_n^{-1}$
$= J_n^{-*} Q^{d^{-1}} J_n^{-1} + J_n^{-*} b^d J_n^{-1} + J_n^{-*} F^{d^T} Q^{d^{-1}} F^d J_n^{-1}$
$= J_n^{-*} \left(Q^{d^{-1}} + b^d + F^{d^T} Q^{d^{-1}} F^d\right) J_n^{-1}$
$= J_n^{-*} \Gamma^d J_n^{-1}$

Also
$c^d = 2J_n^{-1} c^a J_n = J_n^* c^a J_n$
since
$c^a = F^{a^*} Q^{a^{-1}} F^a$
$= \left(J_n F^d J_n^{-1}\right)^* \left(J_n Q^d J_n^*\right)^{-1} \left(J_n F^d J_n^{-1}\right)$
$= J_n^{-*} F^{d^*} J_n^* J_n^{-*} Q^{d^{-1}} J_n^{-1} J_n F^d J_n^{-1}$
$= J_n^{-*} F^{d^T} Q^{d^{-1}} F^d J_n^{-1}$
$= J_n^{-*} c^d J_n^{-1}$

Then
$L^d(1|0) = J_n^* L^a(1|0) J_n$
since
$L^a(1|0) = Q^{a^{-1}} + H^{a^*} R^{a^{-1}} H^a + F^{a^*} Q^{a^{-1}} F^a = \Gamma^a$

Then we define
$L^d(k|k-1) = J_n^* L^a(k|k-1) J_n$
and from
$L^a(k+1|k) = \Gamma^a - \Phi^a L^{a^{-1}}(k|k-1) \Phi^{a^*}$
we get
$L^a(k+1|k) = \Gamma^a - \Phi^a L^{a^{-1}}(k|k-1) \Phi^{a^*}$
$= \left(J_n^{-*} \Gamma^d J_n^{-1}\right)$
$- \left(J_n^{-*} \Phi^d J_n^{-1}\right) \left(J_n^{-*} L^d(k|k-1) J_n^{-1}\right)^{-1} \left(J_n^{-*} \Phi^d J_n^{-1}\right)^*$
$= J_n^{-*} \Gamma^d J_n^{-1}$
$- J_n^{-*} \Phi^d J_n^{-1} J_n \left(L^d(k|k-1)\right)^{-1} J_n^* J_n^{-*} \left(\Phi^d\right)^* J_n^{-1}$
$= J_n^{-*} \Gamma^d J_n^{-1} - J_n^{-*} \Phi^d L^{d^{-1}}(k|k-1) \left(\Phi^d\right)^T J_n^{-1}$
$= J_n^{-*} \left(\Gamma^d - \Phi^d L^{d^{-1}}(k|k-1) \Phi^{d^T}\right) J_n^{-1}$
$= J_n^{-*} L^d(k+1|k) J_n^{-1}$

Thus
$L^d(k+1|k) = J_n^* L^a(k+1|k) J_n$

Finally, we define
$P^d = J_n^{-1} P^a J_n^{-*}$
and from
$P^a = \left\{L^a - \left(H^{a^*} R^{a^{-1}} H^a + F^{a^*} Q^{a^{-1}} F^a\right)\right\}^{-1}$
we get
$P^a = \left\{L^a - \left(H^{a^*} R^{a^{-1}} H^a + F^{a^*} Q^{a^{-1}} F^a\right)\right\}^{-1}$
$\Rightarrow P^{a^{-1}} = L^a - \left(H^{a^*} R^{a^{-1}} H^a + F^{a^*} Q^{a^{-1}} F^a\right)$
$\Rightarrow P^{a^{-1}} = L^a - \left(b^a + F^{a^*} Q^{a^{-1}} F^a\right)$
$= J_n^{-*} L^d J_n^{-1}$
$- \left(J_n^{-*} b^d J_n^{-1} + \left(J_n F^d J_n^{-1}\right)^* \left(J_n Q^d J_n^*\right)^{-1} \left(J_n F^d J_n^{-1}\right)\right)$
$= J_n^{-*} L^d J_n^{-1} - \left(J_n^{-*} b^d J_n^{-1}\right.$
$\left. + J_n^{-*} F^d J_n^* J_n^{-*} Q^{d^{-1}} J_n^{-1} J_n F^d J_n^{-1}\right)$
$= J_n^{-*} L^d J_n^{-1} - \left(J_n^{-*} b^d J_n^{-1} + J_n^{-*} F^d Q^{d^{-1}} F^d J_n^{-1}\right)$
$= J_n^{-*} \left(L^d - \left(b^d + F^d Q^{d^{-1}} F^d\right)\right) J_n^{-1}$
$\Rightarrow P^a = J_n \left\{L^d - \left(b^d + F^d Q^{d^{-1}} F^d\right)\right\}^{-1} J_n^*$
$= J_n P^d J_n^*$

Then we derive the dual transformed per step algorithm:

Dual Transformed Per Step Algorithm

initialization

$$F^d = J_n^{-1} F^a J_n$$
$$H^d = J_m^{-1} H^a J_n$$
$$Q^d = J_n^{-1} Q^a J_n^{-*}$$
$$R^d = J_m^{-1} R^a J_m^{-*}$$
$$\Phi^a = Q^{a^{-1}} F^a$$
$$\Phi^d = J_n^* \Phi^a J_n$$
$$\Gamma^a = Q^{a^{-1}} + H^{a^*} R^{a^{-1}} H^a + F^{a^*} Q^{a^{-1}} F^a$$
$$\Gamma^d = J_n^* \Gamma^a J_n$$
$$b^a = H^{a^*} R^{a^{-1}} H^a$$
$$b^d = J_n^* b^a J_n$$
$$c^a = F^{a^*} Q^{a^{-1}} F^a$$
$$c^d = J_n^* c^a J_n$$

initial condition

$$L^a(1|0) = \Gamma^a = Q^{a^{-1}} + H^{a^*} R^{a^{-1}} H^a + F^{a^*} Q^{a^{-1}} F^a$$
$$L^d(1|0) = J_n^* L^a(1|0) J_n$$

iterations $k = 1,2, \dots$

$$L^d(k+1|k) = \Gamma^d - \Phi^d L^{d^{-1}}(k|k-1) \Phi^{d^T}$$

until

$$\left\| L^d(k+1|k) - L^d(k|k-1) \right\| < \varepsilon$$

output

$$L^d = \lim_{k \to \infty} L(k+1|k)$$
$$P^d = \left\{ L^d - \left(b^d + F^d Q^{d^{-1}} F^d \right) \right\}^{-1}$$
$$P^a = J_n P^d J_n^*$$

Note that $F^d, H^d, Q^d, R^d, \Phi^d, \Gamma^d$ and $L^d(k|k-1)$ are real matrices.

Note also that Q^d, R^d and $L^d(k|k-1)$ are real symmetric matrices.

4 Iterative Doubling Algorithms

4.1 Direct Doubling Algorithm

Doubling algorithms have been part of the folklore associated with Riccati equations in linear systems problems for some time, [1].

After some algebra, assuming that R^a is positive definite (and hence invertible), the augmented complex Riccati equation can be rewritten as:

$$\begin{bmatrix} X(k+1|k) \\ Y(k+1|k) \end{bmatrix} = S^a \begin{bmatrix} X(k|k-1) \\ Y(k|k-1) \end{bmatrix}$$

where

$$P^a(k+1|k) = Y(k+1|k) X^{-1}(k+1|k)$$
$$a^a = F^{a^*}$$
$$b^a = H^{a^*} R^{a^{-1}} H^a$$
$$c^a = Q^a$$

and

$$S^a = \begin{bmatrix} a^{a^{-1}} & a^{a^{-1}} b^a \\ c^a a^{a^{-1}} & a^{a^*} + c^a a^{a^{-1}} b^a \end{bmatrix}$$

is a symplectic matrix.

The doubling concept is to compute the prediction error covariance matrix at time double the previous time. Then, we are able to derive the direct doubling algorithm for solving the complex Riccati equation.

Direct Doubling Algorithm

initial conditions

$$a^a(1) = F^{a^*}, b^a(1) = H^{a^*} R^{a^{-1}} H^a, c^a(1) = Q^a$$

iterations $k = 1,2, \dots$

$$a^a(k+1) = a^a(k) [I^a + b^a(k) c^a(k)]^{-1} a^a(k)$$
$$b^a(k+1) = b^a(k)$$
$$+ a^a(k) [I^a + b^a(k) c^a(k)]^{-1} b^a(k) a^{a^*}(k)$$
$$c^a(k+1) = c^a(k)$$
$$+ a^{a^*}(k) c^a(k) [I^a + b^a(k) c^a(k)]^{-1} a^a(k)$$

until

$$\| c^a(k+1) - c^a(k) \| < \varepsilon$$

output

$$P^a = \lim_{k \to \infty} c^a(k)$$

Note that $b^a(k), c^a(k)$ are Hermitian matrices. Note also that $b^a(k) c^a(k)$ is not a Hermitian matrix, since $b^a(k) c^a(k) \neq c^a(k) b^a(k)$.

4.2 Transformed Doubling Algorithm

Assuming that Q^a and R^a are positive definite (and hence invertible) and using the Matrix Inversion Lemma in the Augmented Complex Riccati equation we get:

$$P^a(k+1|k) = Q^a$$
$$+ F^a \left[P^{a^{-1}}(k|k-1) + H^{a^*} R^{a^{-1}} H^a \right]^{-1} F^{a^*}$$
$$\Rightarrow P^{a^{-1}}(k+1|k) = Q^{a^{-1}}$$
$$- Q^{a^{-1}} F^a$$
$$\left[P^{a^{-1}}(k|k-1) + H^{a^*} R^{a^{-1}} H^a + F^{a^*} Q^{a^{-1}} F^a \right]^{-1}$$
$$F^{a^*} Q^{a^{-1}}$$

Then, setting

$$\Pi^a(k|k-1) = P^{a^{-1}}(k|k-1)$$
$$\beta^a = H^{a^*} R^{a^{-1}} H^a + F^{a^*} Q^{a^{-1}} F^a$$
$$\alpha^a = Q^{a^{-1}} F^a$$

We are able to rewrite the augmented complex Riccati equation as:

$$\Pi^a(k+1|k) = Q^{a^{-1}} - \alpha^a [\Pi^a(k|k-1) + \beta^a]^{-1} \alpha^{a^*}$$

After some algebra, the augmented complex Riccati equation can be rewritten as:

$$\begin{bmatrix} X(k+1|k) \\ Y(k+1|k) \end{bmatrix} = \Sigma^a \begin{bmatrix} X(k|k-1) \\ Y(k|k-1) \end{bmatrix}$$

where

$$\Pi^a(k+1|k) = X(k+1|k) Y^{-1}(k+1|k)$$

$\alpha^a = Q^{a^{-1}}F^a$
$\beta^a = F^{a^*}Q^{a^{-1}}F^a + H^{a^*}R^{a^{-1}}H^a$
$\gamma^a = Q^{a^{-1}}$
and
$\Sigma^a = S^a = \begin{bmatrix} \gamma^a\alpha^{a^{-*}} & \gamma^a\alpha^{a^{-*}}\beta^a - \alpha^a \\ \alpha^{a^{-*}} & \alpha^{a^{-*}}\beta^a \end{bmatrix}$
is a symplectic matrix.

Then assuming $P^a(0|-1) = 0$, which implies $P^a(1|0) = Q^a$, we derive the transformed doubling algorithm:

Transformed Doubling Algorithm
initial conditions
$\alpha^a(1) = Q^{a^{-1}}F^a$
$\beta^a(1) = F^{a^*}Q^{a^{-1}}F^a + H^{a^*}R^{a^{-1}}H^a$
$\gamma^a(1) = Q^{a^{-1}}$
iterations $k = 1,2,\ldots$
$\alpha^a(k+1) = \alpha^a(k)[\beta^a(k) + \gamma^a(k)]^{-1}\alpha^a(k)$
$\beta^a(k+1) = \beta^a(k) - \alpha^{a^*}(k)[\beta^a(k) + \gamma^a(k)]^{-1}\alpha^a(k)$
$\gamma^a(k+1) = \gamma^a(k) - \alpha^a(k)[\beta^a(k) + \gamma^a(k)]^{-1}\alpha^{a^*}(k)$
until
$\|\gamma^a(k+1) - \gamma^a(k)\| < \varepsilon$
output
$\Pi^a = \lim_{k\to\infty}\gamma^a(k)$
$P^a = \Pi^{a^{-1}}$

Note that $\beta^a(k), \gamma^a(k)$ are Hermitian matrices.

4.3 Dual Direct Doubling Algorithm

We are going to derive the iterative dual direct doubling algorithm by using the duality concept, [16]. Using the duality concept and assuming that R^a and R^d are positive definite (and hence invertible), we get:
$b^a = H^{a^*}R^{a^{-1}}H^a$
$= \left(J_m H^d J_n^{-1}\right)^*\left(J_m R^d J_m^*\right)^{-1}\left(J_m H^d J_n^{-1}\right)$
$= J_n^{-1^*}H^{d^*}J_m^*{}^{-1}R^{d^{-1}}J_m^{-1}J_m H^d J_n^{-1}$
$= J_n^{-1^*}\left(H^{d^*}R^{d^{-1}}H^d\right)J_n^{-1} = J_n^{-1^*}\left(H^{d^T}R^{d^{-1}}H^d\right)J_n^{-1}$
$= J_n^{*^{-1}}b^d J_n^{-1}$

It is clear that
$b^a = H^{a^*}R^{a^{-1}}H^a = J_n^{*^{-1}}b^d J_n^{-1} = \frac{1}{2}J_n b^d J_n^{-1}$
and
$b^d = H^{d^T}R^{d^{-1}}H^d = 2J_n^{-1}b^a J_n = J_n^* b^a J_n$

Also,
$I^a = J_n I^d J_n^{-1}$
$a^a(k) = J_n a^d(k) J_n^{-1}$
$b^a(k) = \frac{1}{2}J_n b^d(k) J_n^{-1}$
$c^a(k) = J_n c^d(k) J_n^* = 2J_n c^d(k) J_n^{-1}$

Note that $a^d(k), b^d(k), c^d(k)$ are real matrices.

Then using the doubling algorithm we get:
$a^a(k+1) = a^a(k)[I^a + b^a(k)c^a(k)]^{-1}a^a(k)$
$\Rightarrow J_n a^d(k+1)J_n^{-1} = J_n a^d(k)J_n^{-1}$
$\left[J_n I^d J_n^{-1} + \frac{1}{2}J_n b^d(k)J_n^{-1}2J_n c^d(k)J_n^{-1}\right]^{-1}J_n a^d(k)J_n^{-1}$
$\Rightarrow J_n a^d(k+1)J_n^{-1} = J_n a^d(k)J_n^{-1}$
$\left[J_n I^d J_n^{-1} + J_n b^d(k)c^d(k)J_n^{-1}\right]^{-1}J_n a^d(k)J_n^{-1}$
$\Rightarrow J_n a^d(k+1)J_n^{-1} = J_n a^d(k)J_n^{-1}J_n$
$\left[I^d + b^d(k)c^d(k)\right]^{-1}J_n^{-1}J_n a^d(k)J_n^{-1}$
$= J_n a^d(k)\left[I^d + b^d(k)c^d(k)\right]^{-1}a^d(k)J_n^{-1}$
$\Rightarrow a^d(k+1) = a^d(k)\left[I^d + b^d(k)c^d(k)\right]^{-1}a^d(k)$

$b^a(k+1) = b^a(k) + a^a(k)$
$[I^a + b^a(k)c^a(k)]^{-1}b^a(k)a^{a^*}(k)$
$\Rightarrow \frac{1}{2}J_n b^d(k+1)J_n^{-1} = \frac{1}{2}J_n b^d(k)J_n^{-1}$
$+J_n a^d(k)J_n^{-1}$
$\left[J_n I^d J_n^{-1} + \frac{1}{2}J_n b^d(k)J_n^{-1}2J_n c^d(k)J_n^{-1}\right]^{-1}$
$\frac{1}{2}J_n b^d(k)J_n^{-1}\left(J_n a^d(k)J_n^{-1}\right)^*$
$= \frac{1}{2}J_n b^d(k)J_n^{-1}$
$+J_n a^d(k)J_n^{-1}$
$\left[J_n I^d J_n^{-1} + J_n b^d(k)J_n^{-1}J_n c^d(k)J_n^{-1}\right]^{-1}$
$\frac{1}{2}J_n b^d(k)J_n^{-1}\left(J_n a^d(k)J_n^{-1}\right)^*$
$= \frac{1}{2}J_n b^d(k)J_n^{-1} + \frac{1}{2}J_n a^d(k)J_n^{-1}J_n$
$\left[I^d + b^d(k)c^d(k)\right]^{-1}J_n^{-1}J_n b^d(k)J_n^{-1}\left(J_n a^d(k)J_n^{-1}\right)^*$
$= \frac{1}{2}J_n b^d(k)J_n^{-1} + \frac{1}{2}J_n a^d(k)$
$\left[I^d + b^d(k)c^d(k)\right]^{-1}b^d(k)J_n^{-1}\left(J_n a^d(k)J_n^{-1}\right)^*$
$= \frac{1}{2}J_n b^d(k)J_n^{-1} + \frac{1}{2}J_n a^d(k)$
$\left[I^d + b^d(k)c^d(k)\right]^{-1}b^d(k)J_n^{-1}J_n^{-1^*}\left(a^d(k)\right)^*J_n^*$
$= \frac{1}{2}J_n b^d(k)J_n^{-1} + \frac{1}{2}J_n a^d(k)$
$\left[I^d + b^d(k)c^d(k)\right]^{-1}b^d(k)J_n^{-1}J_n^{*^{-1}}\left(a^d(k)\right)^*J_n^*$
$= \frac{1}{2}J_n b^d(k)J_n^{-1} + \frac{1}{2}J_n a^d(k)$
$\left[I^d + b^d(k)c^d(k)\right]^{-1}b^d(k)J_n^{-1}\frac{1}{2}J_n\left(a^d(k)\right)^*J_n^*$
$= \frac{1}{2}J_n b^d(k)J_n^{-1} + \frac{1}{2}J_n a^d(k)$
$\left[I^d + b^d(k)c^d(k)\right]^{-1}b^d(k)J_n^{-1}\frac{1}{2}J_n\left(a^d(k)\right)^*2J_n^{-1}$
$= \frac{1}{2}J_n b^d(k)J_n^{-1} + \frac{1}{2}J_n a^d(k)$
$\left[I^d + b^d(k)c^d(k)\right]^{-1}b^d(k)\left(a^d(k)\right)^*J_n^{-1}$
$\Rightarrow b^d(k+1) = b^d(k)$
$+a^d(k)\left[I^d + b^d(k)c^d(k)\right]^{-1}b^d(k)\left(a^d(k)\right)^*$
$\Rightarrow b^d(k+1) = b^d(k)$
$+a^d(k)\left[I^d + b^d(k)c^d(k)\right]^{-1}b^d(k)\left(a^d(k)\right)^T$
since $a^d(k)$ is a real matrix

$$c^a(k+1) = c^a(k) + a^{a^*}(k)c(k)$$
$$[I^a + b^a(k)c^a(k)]^{-1}a^a(k)$$
$$\Rightarrow 2J_n c^d(k+1)J_n^{-1} = 2J_n c^d(k)J_n^{-1}$$
$$+\left(J_n a^d(k)J_n^{-1}\right)^* 2J_n c^d(k)J_n^{-1}$$
$$\left[J_n I^d J_n^{-1} + \frac{1}{2}J_n b^d(k)J_n^{-1}2J_n c^d(k)J_n^{-1}\right]^{-1} J_n a^d(k)J_n^{-1}$$
$$= 2J_n c^d(k)J_n^{-1} + \left(J_n a^d(k)J_n^{-1}\right)^* 2J_n c^d(k)J_n^{-1}$$
$$\left[J_n I^d J_n^{-1} + J_n b^d(k)c^d(k)J_n^{-1}\right]^{-1} J_n a^d(k)J_n^{-1}$$
$$= 2J_n c^d(k)J_n^{-1} + 2\left(J_n a^d(k)J_n^{-1}\right)^* J_n c^d(k)J_n^{-1}J_n$$
$$\left[I^d + b^d(k)c^d(k)\right]^{-1}J_n^{-1}J_n a^d(k)J_n^{-1}$$
$$= 2J_n c^d(k)J_n^{-1} + 2\left(J_n a^d(k)J_n^{-1}\right)^* J_n c^d(k)$$
$$\left[I^d + b^d(k)c^d(k)\right]^{-1}a^d(k)J_n^{-1}$$
$$= 2J_n c^d(k)J_n^{-1} + 2J_n^{*-1}\left(a^d(k)\right)^* J_n^* J_n c^d(k)$$
$$\left[I^d + b^d(k)c^d(k)\right]^{-1}a^d(k)J_n^{-1}$$
$$= 2J_n c^d(k)J_n^{-1} + 2\frac{1}{2}J_n\left(a^d(k)\right)^* 2J_n^{-1}J_n c^d(k)$$
$$\left[I^d + b^d(k)c^d(k)\right]^{-1}a^d(k)J_n^{-1}$$
$$= 2J_n c^d(k)J_n^{-1} + 2J_n\left(a^d(k)\right)^* c^d(k)$$
$$\left[I^d + b^d(k)c^d(k)\right]^{-1}a^d(k)J_n^{-1}$$
$$\Rightarrow c^d(k+1) = c^d(k)$$
$$+\left(a^d(k)\right)^* c^d(k)\left[I^d + b^d(k)c^d(k)\right]^{-1}a^d(k)$$
$$\Rightarrow c^d(k+1) = c^d(k)$$
$$+\left(a^d(k)\right)^{\mathrm{T}} c^d(k)\left[I^d + b^d(k)c^d(k)\right]^{-1}a^d(k)$$

since $a^d(k)$ is a real matrix

Thus the dual direct doubling algorithm is derived:

Dual Direct Doubling Algorithm

initial conditions

$$a^a(1) = F^{a^*}$$
$$b^a(1) = H^{a^*}R^{a^{-1}}H^a$$
$$c^a(1) = Q^a$$
$$a^d(1) = J_n^{-1}a^a(1)J_n$$
$$b^d(1) = 2J_n^{-1}b^a(1)J_n$$
$$c^d(1) = \frac{1}{2}J_n^{-1}c^a(1)J_n$$

iterations $k = 1,2,\ldots$

$$a^d(k+1) = a^d(k)\left[I^d + b^d(k)c^d(k)\right]^{-1}a^d(k)$$
$$b^d(k+1) = b^d(k)$$
$$+a^d(k)\left[I^d + b^d(k)c^d(k)\right]^{-1}b^d(k)\left(a^d(k)\right)^{\mathrm{T}}$$
$$c^d(k+1) = c^d(k)$$
$$+\left(a^d(k)\right)^{\mathrm{T}} c^d(k)\left[I^d + b^d(k)c^d(k)\right]^{-1}a^d(k)$$

until

$$\left\|c^d(k+1) - c^d(k)\right\| < \varepsilon$$

output

$$P^d = \lim_{k\to\infty} c^d(k)$$
$$P^a = J_n P^d J_n^*$$

Note that $b^d(k), c^d(k)$ are real symmetric matrices.

4.4 Dual Transformed Doubling Algorithm

$$\alpha^d(1) = Q^{d^{-1}}F^d$$
$$\alpha^a(1) = Q^{a^{-1}}F^a = \left(J_n Q^d J_n^*\right)^{-1}J_n F^d J_n^{-1}$$
$$= J_n^{*-1}Q^{d^{-1}}J_n^{-1}J_n F^d J_n^{-1}$$
$$= J_n^{*-1}Q^{d^{-1}}F^d J_n^{-1} = J_n^{*-1}\alpha^d(1)J_n^{-1}$$
$$= \frac{1}{2}J_n\alpha^d(1)J_n^{-1}$$
$$\alpha^a(k) = J_n^{*-1}\alpha^d(k)J_n^{-1} = \frac{1}{2}J_n\alpha^d(k)J_n^{-1}$$

$$\beta^d(1) = F^{d^{\mathrm{T}}}Q^{d^{-1}}F^d + H^{d^{\mathrm{T}}}R^{d^{-1}}H^d$$
$$\beta^a(1) = F^{a^*}Q^{a^{-1}}F^a + H^{a^*}R^{a^{-1}}H^a$$
$$= \left(J_n F^d J_n^{-1}\right)^*\left(J_n Q^d J_n^*\right)^{-1}J_n F^d J_n^{-1}$$
$$+\left(J_m H^d J_n^{-1}\right)^*\left(J_m R^d J_m^*\right)^{-1}J_m H^d J_n^{-1}$$
$$= J_n^{*-1}F^{d^{\mathrm{T}}}J_n^* J_n^{*-1}Q^{d^{-1}}J_n^{-1}J_n F^d J_n^{-1}$$
$$+J_n^{*-1}H^{d^{\mathrm{T}}}J_m^* J_m^{*-1}R^{d^{-1}}J_m^{-1}J_m H^d J_n^{-1}$$
$$= J_n^{*-1}\left(F^{d^{\mathrm{T}}}Q^{d^{-1}}F^d + H^{d^{\mathrm{T}}}R^{d^{-1}}H^d\right)J_n^{-1}$$
$$= J_n^{*-1}\beta^d(1)J_n^{-1} = \frac{1}{2}J_n\beta^d(1)J_n^{-1}$$
$$\beta^a(k) = J_n^{*-1}\beta^d(k)J_n^{-1} = \frac{1}{2}J_n\beta^d(k)J_n^{-1}$$

$$\gamma^d(1) = Q^{d^{-1}}$$
$$\gamma^a(1) = Q^{a^{-1}} = \left(J_n Q^d J_n^*\right)^{-1} = J_n^{*-1}Q^{d^{-1}}J_n^{-1}$$
$$= J_n^{*-1}\gamma^d(1)J_n^{-1} = \frac{1}{2}J_n\gamma^d(1)J_n^{-1}$$
$$\gamma^a(k) = J_n^{*-1}\gamma^d(k)J_n^{-1} = \frac{1}{2}J_n\gamma^d(k)J_n^{-1}$$

Then using the transformed doubling algorithm we get:

$$\alpha^a(k+1) = \alpha^a(k)[\beta^a(k) + \gamma^a(k)]^{-1}\alpha^a(k)$$
$$\Rightarrow J_n^{*-1}\alpha^d(k+1)J_n^{-1} = J_n^{*-1}\alpha^d(k)J_n^{-1}$$
$$\left[J_n^{*-1}\beta^d(k)J_n^{-1} + J_n^{*-1}\gamma^d(k)J_n^{-1}\right]^{-1}J_n^{*-1}\alpha^d(k)J_n^{-1}$$
$$= J_n^{*-1}\alpha^d(k)J_n^{-1}J_n\left[\beta^d(k) + \gamma^d(k)\right]^{-1}J_n^* J_n^{*-1}\alpha^d(k)J_n^{-1}$$
$$= J_n^{*-1}\alpha^d(k)\left[\beta^d(k) + \gamma^d(k)\right]^{-1}\alpha^d(k)J_n^{-1}$$
$$\Rightarrow \alpha^d(k+1) = \alpha^d(k)\left[\beta^d(k) + \gamma^d(k)\right]^{-1}\alpha^d(k)$$

$$\beta^a(k+1) = \beta^a(k)$$
$$-\alpha^{a^*}(k)[\beta^a(k) + \gamma^a(k)]^{-1}\alpha^a(k)$$
$$\Rightarrow J_n^{*-1}\beta^d(k+1)J_n^{-1} = J_n^{*-1}\beta^d(k)J_n^{-1}$$
$$-\left(J_n^{*-1}\alpha^d(k)J_n^{-1}\right)^*$$
$$\left[J_n^{*-1}\beta^d(k)J_n^{-1} + J_n^{*-1}\gamma^d(k)J_n^{-1}\right]^{-1}J_n^{*-1}\alpha^d(k)J_n^{-1}$$
$$= J_n^{*-1}\beta^d(k)J_n^{-1} - J_n^{*-1}\alpha^{d^{\mathrm{T}}}(k)J_n^{-1}$$
$$\left[J_n^{*-1}\beta^d(k)J_n^{-1} + J_n^{*-1}\gamma^d(k)J_n^{-1}\right]^{-1}J_n^{*-1}\alpha^d(k)J_n^{-1}$$
$$= J_n^{*-1}\beta^d(k)J_n^{-1} - J_n^{*-1}\alpha^{d^{\mathrm{T}}}(k)J_n^{-1}J_n$$
$$\left[\beta^d(k) + \gamma^d(k)\right]^{-1}J_n^* J_n^{*-1}\alpha^d(k)J_n^{-1}$$
$$= J_n^{*-1}\beta^d(k)J_n^{-1} - J_n^{*-1}\alpha^{d^{\mathrm{T}}}(k)$$

$$\left[\beta^{d}(k)+\gamma^{d}(k)\right]^{-1}\alpha^{d}(k)J_n^{-1}$$
$$=J_n^{*-1}$$
$$\left(\beta^{d}(k)-\alpha^{d^T}(k)\left[\beta^{d}(k)+\gamma^{d}(k)\right]^{-1}\alpha^{d}(k)\right)J_n^{-1}$$
$$\Rightarrow \beta^{d}(k+1)=\beta^{d}(k)$$
$$-\alpha^{d^T}(k)\left[\beta^{d}(k)+\gamma^{d}(k)\right]^{-1}\alpha^{d}(k)$$

$$\gamma^{a}(k+1)=\gamma^{a}(k)$$
$$-\alpha^{a}(k)[\beta^{a}(k)+\gamma^{a}(k)]^{-1}\alpha^{a*}(k)$$
$$\Rightarrow J_n^{*-1}\gamma^{d}(k+1)J_n^{-1}=J_n^{*-1}\gamma^{d}(k)J_n^{-1}$$
$$-J_n^{*-1}\alpha^{d}(k)J_n^{-1}$$
$$\left[J_n^{*-1}\beta^{d}(k)J_n^{-1}+J_n^{*-1}\gamma^{d}(k)J_n^{-1}\right]^{-1}\left(J_n^{*-1}\alpha^{d}(k)J_n^{-1}\right)^{*}$$
$$=J_n^{*-1}\gamma^{d}(k)J_n^{-1}-J_n^{*-1}\alpha^{d}(k)J_n^{-1}$$
$$\left[J_n^{*-1}\beta^{d}(k)J_n^{-1}+J_n^{*-1}\gamma^{d}(k)J_n^{-1}\right]^{-1}J_n^{*-1}\alpha^{d^T}(k)J_n^{-1}$$
$$=J_n^{*-1}\gamma^{d}(k)J_n^{-1}-J_n^{*-1}\alpha^{d}(k)J_n^{-1}J_n$$
$$\left[\beta^{d}(k)+\gamma^{d}(k)\right]^{-1}J_n^{*}J_n^{*-1}\alpha^{d^T}(k)J_n^{-1}$$
$$=J_n^{*-1}\gamma^{d}(k)J_n^{-1}-J_n^{*-1}\alpha^{d}(k)$$
$$\left[\beta^{d}(k)+\gamma^{d}(k)\right]^{-1}\alpha^{d^T}(k)J_n^{-1}$$
$$=J_n^{*-1}$$
$$\left(\gamma^{d}(k)-\alpha^{d}(k)\left[\beta^{d}(k)+\gamma^{d}(k)\right]^{-1}\alpha^{d^T}(k)\right)J_n^{-1}$$
$$\Rightarrow \gamma^{d}(k+1)=\gamma^{d}(k)$$
$$-\alpha^{d}(k)\left[\beta^{d}(k)+\gamma^{d}(k)\right]^{-1}\alpha^{d^T}(k)$$

Note that $\alpha^{d}(k),\beta^{d}(k),\gamma^{d}(k)$ are real matrices.

Then we derive the iterative dual transformed doubling algorithm.

Dual Transformed Doubling Algorithm

initial conditions

$$\alpha^{a}(1)=Q^{a^{-1}}F^{a}$$
$$\beta^{a}(1)=F^{a*}Q^{a^{-1}}F^{a}+H^{a*}R^{a^{-1}}H^{a}$$
$$\gamma^{a}(1)=Q^{a^{-1}}$$
$$\alpha^{d}(1)=J_n^{*}\alpha^{d}(1)J_n$$
$$\beta^{d}(1)=J_n^{*}\beta^{a}(1)J_n$$
$$\gamma^{d}(1)=J_n^{*}\gamma^{a}(1)J_n$$

iterations $k=1,2,\ldots$

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

until

$$\left\|\gamma^{d}(k+1)-\gamma^{d}(k)\right\|<\varepsilon$$

output

$$\Pi^{d}=\lim_{k\to\infty}\gamma^{d}(k)$$
$$P^{a}=J_n\Pi^{d^{-1}}J_n^{*}$$

Note that $\beta^{d}(k),\gamma^{d}(k)$ are real symmetric matrices.

5 Computational Requirements

It is established that the iterative algorithms (per step and doubling) are algebraically equivalent. Then it is clear that they calculate theoretically the same Riccati equation solution.

Example. Assume the augmented model complex parameters:

$$F^{a}=\begin{bmatrix}0.2+0.3j & 0.4+0.5j & 0.02-0.06j & 0.04-0.1j\\ 0.01+0.01j & 0.08+0.09j & 0.02-0.02j & 0.14+0.18j\\ 0.02+0.06j & 0.04+0.1j & 0.2-0.3j & 0.4-0.5j\\ 0.02+0.02j & 0.14-0.18j & 0.01-0.01j & 0.08+0.09j\end{bmatrix}$$
$$H^{a}=\begin{bmatrix}2+3j & 0.4+0.5j & 3-4j & 0.6+0.1j\\ 3+4j & 0.6-0.1j & 2-0.3j & 0.4-0.05j\end{bmatrix}$$
$$Q^{a}=\begin{bmatrix}5 & 2+5j & 2+3j & 5+9j\\ 2-5j & 8 & 5+9j & 1+4j\\ 2-3j & 5-9j & 5 & 2-5j\\ 5-9j & 1-j & 2+5j & 8\end{bmatrix}$$
$$R^{a}=\begin{bmatrix}0.5 & 0.2+0.3j\\ 0.2-0.3j & 0.5\end{bmatrix}$$

Using the same convergence criterion $\varepsilon=10^{-8}$, all the augmented algorithms compute the same augmented solution:

$$P^{a}=\begin{bmatrix}6.4828 & 2.5467+3.4481j & 3.0697+1.9908j & 6.4426+9.6689j\\ 2.5467-3.4481j & 10.0854 & 6.4426+9.6689j & 1.1305+5.8535j\\ 3.0697-1.9908j & 6.4426-9.6689j & 6.4828 & 2.5467-3.4481j\\ 6.4426-9.6689j & 1.1305-5.8535j & 2.5467+3.4481j & 10.0854\end{bmatrix}$$

while all the dual algorithms compute the same dual solution:

$$P^{d}=\begin{bmatrix}4.7763 & 4.4946 & 0.9954 & 3.1104\\ 4.4946 & 5.6079 & 6.5585 & 2.9268\\ 0.9954 & 6.5585 & 1.7066 & -1.9479\\ 3.1104 & 2.9268 & -1.9479 & 4.4775\end{bmatrix}$$

with $P^{a}=J_nP^{d}J_n^{*}=2J_nP^{d}J_n^{-1}$

It is obvious that the per step algorithms may provide the information when the steady-state time is reached. It is also clear that the per step algorithms are iterative algorithms and thus it is reasonable to assume that they compute the Riccati equation solution executing the same number of iterations, which is denoted as s, where obviously $s>1$. It is also clear that the doubling algorithms are iterative algorithms and thus it is reasonable to assume that they compute the Riccati equation solution executing the same number of iterations, which is $\lceil\log_2 s\rceil$, due to the doubling concept. Thus, in order to compare the iterative algorithms with respect to their computational time, we compare their total calculation burden, which is the per iteration calculation burden multiplied by the number of iterations; the calculation burden of the initialization process and the output process are not taken into account.

Scalar operations are involved in matrix manipulation operations, which are needed for the implementation of the iterative algorithms. In fact,

real/complex matrix additions, multiplications and inversions are required for the implementation of the algorithms. The calculation burdens of real/complex matrix operations are given in the Appendix, using the calculation burdens of complex matrix operations in [20].

The iteration calculation burdens of the per step and doubling iterative algorithms are analytically calculated in Table 1, Table 2, Table 3, Table 4, Table 5, Table 6, Table 7 and Table 8.

Table 1. Direct per step algorithm per iteration calculation burden

Matrices Operation	Iteration Calculation Burden
$P^{a^{-1}}(k\|k-1)$	$\frac{74n^3-39n^2-5n}{3}$
$P^{a^{-1}}(k\|k-1)+b^a$	$2n^2+n$
$\left[P^{a^{-1}}(k\|k-1)+b^a\right]^{-1}$	$\frac{74n^3-39n^2-5n}{3}$
$F^a\left[P^{a^{-1}}(k\|k-1)+b^a\right]^{-1}$	$32n^3-12n^2$
$F^a\left[P^{a^{-1}}(k\|k-1)+b^a\right]^{-1}F^{a^*}$	$32n^3-6n^2+n$
$P^a(k+1\|k)=Q^a$ $+F^a\left[P^{a^{-1}}(k\|k-1)+b^a\right]^{-1}F^{a^*}$	$2n^2+n$

Table 2. Transformed per step algorithm per iteration calculation burden

Matrices Operation	Iteration Calculation Burden
$L^{a^{-1}}(k\|k-1)$	$\frac{74n^3-39n^2-5n}{3}$
$\Phi^a L^{a^{-1}}(k\|k-1)$	$32n^3-12n^2$
$\Phi^a L^{a^{-1}}(k\|k-1)\Phi^{a^*}$	$32n^3-6n^2+n$
$L^a(k+1\|k)=\Gamma^a$ $-\Phi^a L^{a^{-1}}(k\|k-1)\Phi^{a^*}$	$2n^2+n$

Table 3. Dual direct step algorithm per iteration calculation burden

Matrices Operation	Iteration Calculation Burden
$P^{d^{-1}}(k\|k-1)$	$\frac{7(2n)^3-(2n)}{6}$
$P^{a^{-1}}(k\|k-1)+b^d$	$\frac{1}{2}(2n)^2+\frac{1}{2}(2n)$
$\left[P^{d^{-1}}(k\|k-1)+b^d\right]^{-1}$	$\frac{7(2n)^3-(2n)}{6}$
$F^d\left[P^{d^{-1}}(k\|k-1)+b^d\right]^{-1}$	$2(2n)^3-(2n)^2$
$F^d\left[P^{d^{-1}}(k\|k-1)+b^d\right]^{-1}F^{d^T}$	$(2n)^3+\frac{1}{2}(2n)^2-\frac{1}{2}(2n)$
$P^d(k+1\|k)=Q^d$ $+F^d\left[P^{d^{-1}}(k\|k-1)+b^d\right]^{-1}F^{d^T}$	$\frac{1}{2}(2n)^2+\frac{1}{2}(2n)$

Table 4. Dual transformed step algorithm per iteration calculation burden

Matrices Operation	Iteration Calculation Burden
$L^{d^{-1}}(k\|k-1)$	$\frac{7(2n)^3-(2n)}{6}$
$\Phi^d L^{d^{-1}}(k\|k-1)$	$2(2n)^3-(2n)^2$
$\Phi^d L^{d^{-1}}(k\|k-1)\Phi^{d^T}$	$(2n)^3+\frac{1}{2}(2n)^2-\frac{1}{2}(2n)$
$L^d(k+1\|k)=\Gamma^d$ $-\Phi^d L^{d^{-1}}(k\|k-1)\Phi^{d^T}$	$\frac{1}{2}(2n)^2+\frac{1}{2}(2n)$

Table 5. Direct doubling algorithm per iteration calculation burden

Matrices Operation	Iteration Calculation Burden
$b^a(k)c^a(k)$	$32n^3-4n^2$
$I^a+b^a(k)c^a(k)$	n
$[I^a+b^a(k)c^a(k)]^{-1}$	$\frac{132n^3-54n^2+12n}{3}$
$a^a(k)[I^a+b^a(k)c^a(k)]^{-1}$	$32n^3-4n^2$
$a^a(k+1)=a^a(k)$ $[I^a+b^a(k)c^a(k)]^{-1}a^a(k)$	$32n^3-4n^2$
$b^a(k)a^{a^*}(k)$	$32n^3-4n^2$
$a^a(k)[I^a+b^a(k)c^a(k)]^{-1}$ $b^a(k)a^{a^*}(k)$	$32n^3-6n^2+n$
$b^a(k+1)=b^a(k)$ $+a^a(k)[I^a+b^a(k)c^a(k)]^{-1}$ $b^a(k)a^{a^*}(k)$	$2n^2+n$
$a^{a^*}(k)c^a(k)$	$32n^3-4n^2$
$a^{a^*}(k)c^a(k)$ $[I^a+b^a(k)c^a(k)]^{-1}$	$32n^3-4n^2$
$a^{a^*}(k)c^a(k)$ $[I^a+b^a(k)c^a(k)]^{-1}a^a(k)$	$32n^3-6n^2+n$
$c^a(k+1)=c^a(k)$ $+a^{a^*}(k)c^a(k)[I^a+b^a(k)c^a(k)]^{-1}a^a(k)$	$2n^2+n$

Table 6. Transformed doubling algorithm per iteration calculation burden

Matrices Operation	Iteration Calculation Burden
$\beta^a(k)+\gamma^a(k)$	$2n^2+n$
$[\beta^a(k)+\gamma^a(k)]^{-1}$	$\frac{74n^3-39n^2-5n}{3}$
$\alpha^a(k)[\beta^a(k)+\gamma^a(k)]^{-1}$	$32n^3-4n^2$
$\alpha^a(k+1)$ $=\alpha^a(k)[\beta^a(k)+\gamma^a(k)]^{-1}\alpha^a(k)$	$32n^3-4n^2$
$\alpha^{a^*}(k)[\beta^a(k)+\gamma^a(k)]^{-1}$	$32n^3-4n^2$
$\alpha^{a^*}(k)[\beta^a(k)+\gamma^a(k)]^{-1}\alpha^a(k)$	$32n^3-6n^2+n$
$\beta^a(k+1)=\beta^a(k)$ $-\alpha^{a^*}(k)[\beta^a(k)+\gamma^a(k)]^{-1}\alpha^a(k)$	$2n^2+n$
$\alpha^a(k)[\beta^a(k)+\gamma^a(k)]^{-1}$	$32n^3-4n^2$
$\alpha^a(k)[\beta^a(k)+\gamma^a(k)]^{-1}\alpha^{a^*}(k)$	$32n^3-6n^2+n$
$\gamma^a(k+1)=\gamma^a(k)$ $-\alpha^a(k)[\beta^a(k)+\gamma^a(k)]^{-1}\alpha^{a^*}(k)$	$2n^2+n$

Table 7. Dual direct doubling algorithm per iteration calculation burden

Matrices Operation	**Iteration Calculation Burden**
$b^d(k)c^d(k)$	$2(2n)^3-(2n)^2$
$I^d+b^d(k)c^d(k)$	$(2n)$
$\left[I^d+b^d(k)c^d(k)\right]^{-1}$	$\frac{14(2n)^3-15(2n)^2+7(2n)}{6}$
$a^d(k)\left[I^d+b^d(k)c^d(k)\right]^{-1}$	$2(2n)^3-(2n)^2$
$a^d(k+1)=a^d(k)\left[I^d+b^d(k)c^d(k)\right]^{-1}a^d(k)$	$2(2n)^3-(2n)^2$
$b^d(k)a^{d^T}(k)$	$2(2n)^3-(2n)^2$
$a^d(k)\left[I^d+b^d(k)c^d(k)\right]^{-1}b^d(k)a^{d^T}(k)$	$(2n)^3+\frac{1}{2}(2n)^2-\frac{1}{2}(2n)$
$b^d(k+1)=b^d(k)+a^d(k)\left[I^d+b^d(k)c^d(k)\right]^{-1}b^d(k)a^{d^T}(k)$	$\frac{1}{2}(2n)^2+\frac{1}{2}(2n)$
$a^{d^T}(k)c^d(k)$	$2(2n)^3-(2n)^2$
$a^{d^T}(k)c^d(k)\left[I^d+b^d(k)c^d(k)\right]^{-1}$	$2(2n)^3-(2n)^2$
$a^{d^T}(k)c^d(k)\left[I^d+b^d(k)c^d(k)\right]^{-1}a^d(k)$	$(2n)^3+\frac{1}{2}(2n)^2-\frac{1}{2}(2n)$
$c^d(k+1)=c^d(k)+a^{d^T}(k)c^d(k)\left[I^d+b^d(k)c^d(k)\right]^{-1}a^d(k)$	$\frac{1}{2}(2n)^2+\frac{1}{2}(2n)$

Table 8. dual transformed doubling algorithm per iteration calculation burden

Matrices Operation	**Iteration Calculation Burden**
$\beta^d(k)+\gamma^d(k)$	$\frac{1}{2}(2n)^2+\frac{1}{2}(2n)$
$\left[\beta^d(k)+\gamma^d(k)\right]^{-1}$	$\frac{7(2n)^3-(2n)}{6}$
$\alpha^d(k)\left[\beta^d(k)+\gamma^d(k)\right]^{-1}$	$2(2n)^3-(2n)^2$
$\alpha^d(k+1)=\alpha^d(k)\left[\beta^d(k)+\gamma^d(k)\right]^{-1}\alpha^d(k)$	$2(2n)^3-(2n)^2$
$\alpha^{d^T}(k)\left[\beta^d(k)+\gamma^d(k)\right]^{-1}$	$2(2n)^3-(2n)^2$
$\alpha^{d^T}(k)\left[\beta^d(k)+\gamma^d(k)\right]^{-1}\alpha^d(k)$	$(2n)^3+\frac{1}{2}(2n)^2-\frac{1}{2}(2n)$
$\beta^d(k+1)=\beta^d(k)-\alpha^{d^T}(k)\left[\beta^d(k)+\gamma^d(k)\right]^{-1}\alpha^d(k)$	$\frac{1}{2}(2n)^2+\frac{1}{2}(2n)$
$\alpha^d(k)\left[\beta^d(k)+\gamma^d(k)\right]^{-1}$	$2(2n)^3-(2n)^2$
$\alpha^d(k)\left[\beta^d(k)+\gamma^d(k)\right]^{-1}\alpha^{d^T}(k)$	$(2n)^3+\frac{1}{2}(2n)^2-\frac{1}{2}(2n)$
$\gamma^d(k+1)=\gamma^d(k)-\alpha^d(k)\left[\beta^d(k)+\gamma^d(k)\right]^{-1}\alpha^{d^T}(k)$	$\frac{1}{2}(2n)^2+\frac{1}{2}(2n)$

The order of the iteration calculation burdens of the per step and doubling iterative algorithms are summarized in Table 9.

Table 9. Order of calculation burdens of the per step and doubling algorithms

Per Step Algorithm	**Iterations**	**Iteration Calculation Burden**
direct	s	$\frac{340}{3}n^3=113.333n^3$
transformed		$\frac{266}{3}n^3=88.666n^3$
dual direct		$\frac{128}{3}n^3=42.666n^3$
dual transformed		$\frac{100}{3}n^3=33.333n^3$
Doubling Algorithm	**Iterations**	**Iteration Calculation Burden**
direct	$\lceil\log_2 s\rceil$	$300n^3$
transformed		$\frac{650}{3}n^3=216.666n^3$
dual direct		$\frac{392}{3}n^3=130.666n^3$
dual transformed		$\frac{268}{3}n^3=89.333n^3$

From Table 9 we get:

a) per step algorithms

- The transformed algorithm is faster than the direct algorithm.
- The dual transformed algorithm is faster than the dual direct algorithm.
- The dual direct algorithm is faster than the transformed algorithm.
- The dual transformed algorithm is the fastest per step algorithm.

b) doubling algorithms

- The transformed algorithm is faster than the direct algorithm.
- The dual transformed algorithm is faster than the dual direct algorithm.
- The dual direct algorithm is faster than the transformed algorithm.
- The dual transformed algorithm is the fastest doubling algorithm.

c) per step vs doubling algorithms

- For the ratio of the total calculation burden of the direct per step algorithm divided by the total calculation burden of the direct doubling algorithm we derive:

$$\frac{\frac{340}{3}n^3\cdot s}{300n^3\cdot\lceil\log_2 s\rceil}=\frac{\frac{340}{3}\cdot s}{300\cdot\lceil\log_2 s\rceil}=0.377\frac{s}{\lceil\log_2 s\rceil}>1, when\ s\geq 8$$

- For the ratio of the total calculation burden of the transformed per step algorithm divided by the total calculation burden of the transformed doubling algorithm we derive:

$$\frac{\frac{266}{3}n^3\cdot s}{\frac{650}{3}n^3\cdot\lceil\log_2 s\rceil}=\frac{266\cdot s}{650\cdot\lceil\log_2 s\rceil}=0.409\frac{s}{\lceil\log_2 s\rceil}>1, when\ s\geq 8$$

- For the ratio of the total calculation burden of the dual direct per step algorithm divided by the total

calculation burden of the dual direct doubling algorithm we derive:

$$\frac{\frac{128}{3}n^3 \cdot s}{\frac{392}{3}n^3 \cdot \lceil \log_2 s \rceil} = \frac{128 \cdot s}{392 \cdot \lceil \log_2 s \rceil} = 0.327 \frac{s}{\lceil \log_2 s \rceil} > 1, when\ s \geq 13$$

- For the ratio of the total calculation burden of the dual transformed per step algorithm divided by the total calculation burden of the dual transformed doubling algorithm we derive:

$$\frac{\frac{100}{3}n^3 \cdot s}{\frac{268}{3}n^3 \cdot \lceil \log_2 s \rceil} = \frac{100 \cdot s}{268 \cdot \lceil \log_2 s \rceil} = 0.373 \frac{s}{\lceil \log_2 s \rceil} > 1, when\ s \geq 11$$

Thus,

- If the convergence criterion is small/large enough, the more/less accurate solution is derived and the faster algorithm is the doubling/per step algorithm.
- The dual transformed algorithm is the fastest algorithm.

6 Conclusions

Complex Kalman filters are used to process complex signals that are ubiquitous in many fields of science and engineering. The augmented complex Riccati equation appears for time-invariant augmented or widely linear models. The solution of the Riccati equation is prerequisite for determining the steady-state augmented complex Kalman filter parameters before any measurements are observed. Iterative per step and doubling algorithms for the solution of the complex Riccati equation are derived.

Table 10 summarizes the complex Riccati equation solution algorithms with respect to the model dimension and the arithmetic they use.

Table 10. Complex Riccati equation solution algorithms

Per Step Algorithm	**Dimension**	**Arithmetic**
direct	n	complex
transformed	n	complex
dual direct	2n	real
dual transformed	2n	real
Doubling Algorithm	**Dimension**	**Calculation Burden**
direct	n	complex
transformed	n	complex
dual direct	2n	real
dual transformed	2n	real

The computational requirements of the iterative algorithms were derived, enabling the selection of the faster iterative algorithm. The main characteristics of the derived algorithms are:

- The direct use of the Riccati recursion leads to the direct per step algorithm for augmented matrices.
- The use of the Matrix Inversion Lemma leads to the transformed per step algorithm for augmented matrices.
- The dual direct and transformed per step algorithms use real arithmetic exploiting the dual concept.
- The doubling algorithms use the doubling concept (computation of the prediction error covariance matrix at time double the previous time).
- The per step algorithms may provide the information when the steady-state time is reached.
- The transformed algorithms are faster than the direct algorithms.
- The dual algorithms are faster than the non-dual (complex) algorithms.
- The dual transformed per step algorithm is the fastest per step algorithm.
- The dual transformed doubling algorithm is the fastest doubling algorithm.
- The speed of the doubling algorithms is undoubtable, but they do not provide the information when the steady-state time is reached.
- If the convergence criterion is small/large enough, the more/less accurate solution is derived and the faster algorithm is the doubling/per step algorithm.
- The doubling algorithms are faster than the per step algorithm for a sufficiently large number of iterations (small enough convergence criterion).
- The dual transformed algorithm is the fastest algorithm. The advantage of the use of the dual concept is the computational complexity reduction, due to the fact that the dual approach involves real matrix operations instead of complex matrix operations.

A subject of future research is to investigate the applicability of the presented algorithms to the solution of the complex Riccati equation derived from continuous-time systems and to the solution of the complex Sylvester equation.

References:

[1] B. D. O. Anderson, J. B. Moore, Optimal *Filtering*, Dover Publications, New York, 2005, https://researchportalplus.anu.edu.au/en/publications/optimal-filtering, (Accessed Date: November 30, 2024).

[2] B. Ristic, S. Arulampalam, N. Gordon, *Beyond The Kalman Filter*, Boston London, Artech House, 2004, [Online]. https://us.artechhouse.com/Beyond-the-Kalman-Filter-Particle-Filters-for-Tracking-Applications-P1376.aspx (Accessed Date: November 30, 2024).

[3] H. Durrant-Whyte, T. Bailey: Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine* 13(2), 99–110 (Jun 2006), https://doi.org/10.1109/MRA.2006.1638022, (Accessed Date: November 30, 2024).

[4] H. Coskun, F. Achilles, R. DiPietro, N. Navab, F. Tombari, F.: Long short-term memory Kalman filters: Recurrent neural estimators for pose regularization. In: *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5525–5533. IEEE, Venice (Oct 2017), https://arxiv.org/abs/1708.01885, (Accessed Date: November 30, 2024).

[5] R. Verma, L. Shrinivasan and K. Shreedarshan, GPS/INS integration during GPS outages using machine learning augmented with Kalman filter, *WSEAS Transactions on Systems and Control* (16), pp. 294-301, 2021, DOI: https://doi.org/10.37394/23203.2021.16.25., (Accessed Date: November 30, 2024).

[6] Y. Li, Q. Gui, S. Han, Y. Gu, Tikhonov Regularized Kalman Filter and its Applications in Autonomous Orbit Determination of BDS, *WSEAS Transactions on Mathematics*, (16), pp. 187-196, 2017.

[7] A. Guven and C. Hajiyev, Two-Stage Kalman Filter Based Estimation of Boeing 747 Actuator/Control Surface Stuck Faults, *WSEAS Transactions on Signal Processing*, vol. 19, 2023, pp. 32-40, https://doi.org/10.37394/232014.2023.19.4.

[8] C. Hajiyev and U. Hacizade, A Covariance Matching-Based Adaptive Measurement Differencing Kalman Filter for INS's Error Compensation, *WSEAS Transactions on Systems and Control*, vol. 18, 2023, pp. 478-486, https://doi.org/10.37394/23203.2023.18.51.

[9] M. Sever, T.Y. Erkeç, C. Hajiyev, Comparison of Adaptive Kalman Filters in Aircraft State Estimation, *WSEAS Transactions on Signal Processing*, vol. 19, pp. 128-138, 2023, DOI:10.37394/232014.2023.19.14

[10] A. Mohammadi, K.N. Plataniotis, Structure induced complex Kalman filter for decentralized sequential Bayesian estimation, *IEEE Signal Process. Lett.* 22 (9), 1419–1423, 2015, DOI: http://dx.doi.org/10.1109/LSP.2015.2407196.

[11] G. Wang, S. S. Ge, R. Xue, J. Zhao, C. Li, Complex-valued Kalman filters based on Gaussian entropy, *Signal Processing* 160, 178–189, 2019, DOI: http://dx.doi.org/10.1016/j.sigpro.2019.02.024

[12] E. T. Andrew, K. H. Ahmed, D. Holliday, A new model predictive current controller for grid-connected converters in unbalanced grids, *IEEE Transactions on Power Electronics*, 37(8), pp. 9175-9186, 2022, DOI: 10.1109/TPEL.2022.3158016.

[13] D. Borio, M. Susi, M., Bicomplex Kalman Filter Tracking for GNSS Meta-Signals, In Proceedings of the *36th International Technical Meeting of the Satellite Division of The Institute of Navigation* (ION GNSS+ 2023), Denver, Colorado, pp. 3353-3373, 2023, DOI: http://dx.doi.org/10.33012/2023.19233.

[14] M. El-Nagar, K. Ahmed, E. Hamdan, A. S. Abdel-Khalik, M. S. Hamad, S. Ahmed, Modified extended complex Kalman filter for DC offset and distortion rejection in grid-tie transformerless converters, *Applied Sciences*, 13(15), 2023, https://doi.org/10.3390/app13159023.

[15] J. T. Lassen, P. Mercorelli, Tuning Kalman Filter in Linear Systems, WSEAS Transactions on Systems and Control, vol. 14, pp. 209-212, 2019, [Online]. https://www.wseas.org/multimedia/journals/control/2019/a525103-858.php (Accessed Date: February 7, 2025).

[16] D. H. Dini, D. P. Mandic, Class of widely linear complex Kalman filters. *IEEE Transactions on Neural Networks and Learning Systems* 23(5), 775–786 (May 2012). https://doi.org/10.1109/TNNLS.2012.2189893

[17] A. Polyzos, C. Tsinos, M. Adam, N. Assimakis, Faster complex Lyapunov equation solution selection. *1st International Conference on Frontiers of Artificial*

Intelligence, Ethics, and Multidisciplinary Applications (FAIEMA 2023), Athens, 2023.

[18] M. Hernández-Verón, N. Romero, Existence, localization and approximation of solution of symmetric algebraic Riccati equations. *Computers & Mathematics with Applications* 76(1), 187–203 (Jul 2018). https://doi.org/10.1016/j.camwa.2018.04.014

[19] L. Dong, J. Li, A new class of complex nonsymmetric algebraic Riccati equations with its ω -comparison matrix being an irreducible singular m-matrix. *International Journal of Computer Mathematics* 98(1), 75–105 (Jan 2021). https://doi.org/10.1080/00207160.2020.1729358

[20] A. Polyzos, C. Tsinos, M. Adam, N. Assimakis: Complex Information Filter and Complex Kalman Filter Comparison: Selection of the Faster Filter, *WSEAS Transactions on Systems and Control* (19), pp. 324-333, 2024.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare.

APPENDIX

Calculation Burdens of Matrix Operations

Real Matrices Operations

The calculation burdens (CB) of real matrices operations (additions, multiplications, inversions) depend on the matrices' dimensions and involve real scalar additions, multiplications and divisions, the calculation burdens of which are assumed to be equal.

Table 11 summarizes real scalar operations. In the following, $r, r1, r2$ are real numbers.

Table 11. Real scalar operations

code	real scalar operation	real scalar adds	real scalar mults	real scalar divs	CB
R1	$r1 + r2 = r$	1	0	0	1
R2	$r1 \cdot r2 = r$	0	1	0	1
R3	$r1/r2 = r$	0	0	1	1

Table 12 summarizes real matrices additions and Table 13 summarizes real matrices multiplications. In the following R, R1, R2 are general real matrices. Σ is a real symmetric matrix. I is the identity matrix.

Table 12. Real Matrices Addition

Real Matrices Operation	R1	R2	Calculation Burden
$I + R1 = R$ $(n \times n) + (n \times n)$	n		n
$R1 + R2 = \Sigma$ $(n \times n) + (n \times n)$	$\frac{n^2 + n}{2}$		$\frac{1}{2}n^2 + \frac{1}{2}n$

Table 13. Real Matrices Multiplication

Real Matrices Operation	R1	R2	Calculation Burden
$R1 \cdot R2 = R$ $(n \times n) \cdot (n \times n)$	$n^2(n-1)$	n^3	$2n^3 - n^2$
$R1 \cdot R2 = \Sigma$ $(n \times n) \cdot (n \times n)$	$\frac{n^2 + n}{2}(n-1)$	$\frac{n^2 + n}{2}n$	$n^3 + \frac{1}{2}n^2 - \frac{1}{2}n$

Table 14 summarizes real matrices inversions. Table 15 presents the real general matrix inversion and Table 16 presents the real symmetric matrix inversion.

Table 14. Real Matrices Inversion

Real Matrices Operation	Calculation Burden
R^{-1} $(n \times n)$	$\frac{14n^3 - 15n^2 + 7n}{6}$
Σ^{-1} $(n \times n)$	$\frac{7n^3 - n}{6}$

Inverse of Real General matrix

Recursive algorithm

$$M = \begin{bmatrix} A & b \\ c & d \end{bmatrix}$$

Dimensions:

M nxn

A $(n-1)x(n-1)$

b $(n-1)x1$

c $1x(n-1)$

d 1x1

$$M^{-1} = \begin{bmatrix} A^{-1} + s\left(\frac{1}{s}A^{-1}b\right)\left(\frac{1}{s}cA^{-1}\right) & -\frac{1}{s}A^{-1}b \\ -\frac{1}{s}cA^{-1} & \frac{1}{s} \end{bmatrix}, s = d - cA^{-1}b$$

Table 15. Real General Matrix Inversion

Matrix operation	R1	R2	R3	Calculation Burden
A^{-1} (NxN)				T(N)
$A^{-1}b$ $(NxN) \cdot (Nx1)$	$(N-1)N$	N^2		$2N^2 - N$
cA^{-1} $(1xN) \cdot (NxN)$	$(N-1)N$	N^2		$2N^2 - N$
$c(A^{-1}b)$ $(1xN) \cdot (Nx1)$	$N-1$	N		$2N-1$
$s = d - cA^{-1}b$ $(1x1) + (1x1)$	1			1
$\frac{1}{s}$ $(1x1)/(1x1)$			1	1
$\frac{1}{s}(A^{-1}b)$ $-\frac{1}{s}(A^{-1}b)$ $(1x1) \cdot (Nx1)$		N		N
$\frac{1}{s}(cA^{-1})$ $-\frac{1}{s}(cA^{-1})$ $(1x1) \cdot (1xN)$		N		N
$\left(\frac{1}{s}A^{-1}b\right)$ $\left(\frac{1}{s}cA^{-1}\right)$ $(Nx1) \cdot (1xN)$		N^2		N^2
$s\left(\frac{1}{s}A^{-1}b\right)$ $\left(\frac{1}{s}cA^{-1}\right)$ $(1x1) \cdot (NxN)$		N^2		N^2
A^{-1} $+s$ $\left(\frac{1}{s}A^{-1}b\right)$ $\left(\frac{1}{s}cA^{-1}\right)$ $(NxN) + (NxN)$	N^2			N^2
	$3N^2 - N$	$4N^2 + 3N$	1	$7N^2 + 2N + 1$

$N = n - 1$

$T(n) = T(n-1) + f(n)$

$f(n) = 7(n-1)^2 + 2(n-1) + 1 = 7n^2 - 12n + 6, f(1) = 1$

$T(n) = T(n-1) + 7n^2 - 12n + 6, T(1) = 1$

$T(1) = 1$

$T(2) = T(1) + f(2)$

$T(3) = T(2) + f(3) = T(1) + f(2) + f(3)$

...

$T(n) = T(1) + f(2) + f(3) + \cdots + f(n)$

$$T(n) = \{T(1) - f(1)\} + \{f(1) + f(2) + f(3) + \cdots + f(n)\}$$
$$= 7\sum_{i=1}^{n} i^2 - 12\sum_{i=1}^{n} i + 6\sum_{i=1}^{n} 1$$
$$= 7\frac{n(n+1)(2n+1)}{6} - 12\frac{n(n+1)}{2}$$
$$+ 6n = \frac{14n^3 - 15n^2 + 7n}{6}$$

Inverse of Real Symmetric matrix

Recursive algorithm

$$M = \begin{bmatrix} A & b \\ b^T & d \end{bmatrix}, M = M^T, A = A^T$$

Dimensions:

M nxn

A $(n-1)x(n-1)$

b $(n-1)x1$

b^T $1x(n-1)$

d 1x1

$$M^{-1} = \begin{bmatrix} A^{-1} + s\left(\frac{1}{s}A^{-1}b\right)\left(\frac{1}{s}b^TA^{-1}\right) & -\frac{1}{s}A^{-1}b \\ -\frac{1}{s}b^TA^{-1} & \frac{1}{s} \end{bmatrix}, s$$
$$= d - b^TA^{-1}b$$

Table 16. Real Symmetric Matrix Inversion

Matrix operation	R1	R2	R3	Calculation Burden
A^{-1} (NxN)				T(N)
$A^{-1}b$ (NxN) · (Nx1)	$(N-1)N$	N^2		$2N^2 - N$
$b^TA^{-1} = (A^{-1}b)^T$ (1xN) · (NxN)				
$b^T(A^{-1}b)$ (1xN) · (Nx1)	N − 1	N		2N − 1
$s = d - b^TA^{-1}b$ (1x1) + (1x1)	1			1
$\frac{1}{s}$ (1x1)/(1x1)			1	1
$\frac{1}{s}(A^{-1}b)$ $-\frac{1}{s}(A^{-1}b)$ (1x1) · (Nx1)		N		N
$\frac{1}{s}(b^TA^{-1})$ $-\frac{1}{s}(b^TA^{-1})$ (1x1) · (1xN)				
$\left(\frac{1}{s}A^{-1}b\right)$ $\left(\frac{1}{s}b^TA^{-1}\right)$ (Nx1) · (1xN) symmetric		$\frac{N^2+N}{2}$		$\frac{N^2+N}{2}$
$s\left(\frac{1}{s}A^{-1}b\right)$ $\left(\frac{1}{s}b^TA^{-1}\right)$ (1x1) · (NxN) symmetric		$\frac{N^2+N}{2}$		$\frac{N^2+N}{2}$
A^{-1} +s $\left(\frac{1}{s}A^{-1}b\right)$ $\left(\frac{1}{s}b^TA^{-1}\right)$ (NxN) + (NxN) symmetric	$\frac{N^2+N}{2}$			$\frac{N^2+N}{2}$
	$\frac{3N^2+N}{2}$	$2N^2 + 3N$	1	$\frac{7N^2+7N+2}{2}$

$$N = n - 1$$
$$T(n) = T(n-1) + f(n)$$
$$f(n) = \frac{7(n-1)^2 + 7(n-1) + 2}{2} = \frac{7n^2 - 7n + 2}{2}, f(1) = 1$$
$$T(n) = T(n-1) + \frac{7n^2 - 7n + 2}{2}, T(1) = 1$$

$$T(1) = 1$$
$$T(2) = T(1) + f(2)$$
$$T(3) = T(2) + f(3) = T(1) + f(2) + f(3)$$
$$\ldots$$
$$T(n) = T(1) + f(2) + f(3) + \cdots + f(n)$$
$$T(n) = \{T(1) - f(1)\} + \{f(1) + f(2) + f(3) + \cdots + f(n)\}$$
$$= \frac{1}{2}\left\{7\sum_{i=1}^{n} i^2 - 7\sum_{i=1}^{n} i + 2\sum_{i=1}^{n} 1\right\}$$
$$= \frac{1}{2}\left\{7\frac{n(n+1)(2n+1)}{6} - 7\frac{n(n+1)}{2} + 2n\right\} = \frac{7n^3 - n}{6}$$

Complex Matrices Operations

The calculation burdens of complex matrix operations (additions, multiplications and inverses) depend on the matrices' dimensions and involve complex scalar operations (additions, multiplications, divisions). The complex scalar operations involve real scalar operations (additions, multiplications, divisions) the calculation burdens of which are assumed to be equal.

Table 17 summarizes complex scalar operations. In the following $c, c1, c2$ are complex numbers.

Table 17. Complex scalar operations

code	complex scalar operation	real scalar adds	real scalar mults	real scalar divs	CB
C1	$c1 + c2 = c$	2	0	0	2
C2	$c1 \cdot c2 = c$	2	4	0	6
C3	$c1 + c2 = r$	1	0	0	1
C4	$r + c1 = c$	1	0	0	1
C5	$r \cdot c1 = c$	0	2	0	2
C6	$c1 \cdot c2 = r$	1	2	0	3
C7	$c \cdot \bar{c} = r$	1	2	0	3
C8	$c1/r = c$	0	0	2	2
C9	$c1/c2 = c$	3	6	2	11
C10	$1/c1 = c$	2	1	2	5

Table 18 summarizes complex matrices additions and Table 19 summarizes augmented complex matrices additions. Table 20 summarizes complex matrices multiplications and Table 21 summarizes augmented complex matrices multiplications.In the following, C, C1, C2 are $n \times n$ general complex matrices; H, H1, H2 are $n \times n$ complex Hermitian matrices; S, S1, S2 are $n \times n$ complex symmetric matrices; I is the $n \times n$ identity matrix.

Also, $A^a, A1^a, A2^a$ are $2n \times 2n$ augmented complex matrices of the form $\begin{bmatrix} C1 & C2 \\ \overline{C2} & \overline{C1} \end{bmatrix}$; $A_s^a, A1_s^a, A2_s^a$ are $2n \times 2n$ special augmented complex matrix of the form $A_s^a = \begin{bmatrix} H & S \\ \overline{S} & \overline{H} \end{bmatrix}$; I^a is the $2n \times 2n$ identity matrix.

Table 18. Complex Matrices Addition

code	Complex Matrices Operation	R1	C1	Calculation Burden
A1	$I + C1 = C$	n	0	n
A2	$C1 + C2 = C$		n^2	$2n^2$
A3	$C1 + C2 = H$		$\frac{n^2+n}{2}$	n^2
A4	$H1 + H2 = H$	n	$\frac{n^2-n}{2}$	n^2
A5	$S1 + S2 = S$		$\frac{n^2+n}{2}$	$n^2 + n$
A6	$C1 + C2 = S$		$\frac{n^2+n}{2}$	$n^2 + n$

Table 19. Augmented Complex Matrices Addition

code	Complex Matrices Operation	operation	times	Calculation Burden
A7	$I^a + A1^a = A^a$	A1	1	n
A8	$A1_s^a + A2_s^a = A_s^a$	A4	1	$2n^2 + n$

Table 20. Complex Matrices Multiplication

M1	$C1 \cdot C2 = C$ $C \cdot S = C$ $S \cdot C = C$ $S \cdot S = C$	C1	$n^2(n-1)$	$8n^3 - 2n^2$
		C2	n^3	
M2	$C1 \cdot C2 = H$	R1	$n(n-1)$	$4n^3 - n^2$
		C1	$\left(\frac{n^2-n}{2}\right)(n-1)$	
		C2	$\left(\frac{n^2-n}{2}\right)n$	
		C7	n^2	
M3	$C1 \cdot H = C$ $H \cdot C = C$ $S \cdot H = C$ $H \cdot S = C$	C1	$n^2(n-1)$	$8n^3 - 6n^2$
		C2	$n^2(n-1)$	
		C5	n^2	
M4	$H \cdot C = S$ $C \cdot H = S$	C1	$\left(\frac{n^2+n}{2}\right)(n-1)$	$4n^3 + n^2 - 3n$
		C2	$\left(\frac{n^2+n}{2}\right)(n-1)$	
		C5	$\left(\frac{n^2+n}{2}\right)$	
M5	$H \cdot H = C$	C1	$n(n-1)^2 + n(n-2)$	$8n^3 - 4n^2 + 2n$
		C2	$n(n-1)(n-2) + n(n-1)$	
		C4	n	
		C5	$2n(n-1)$	
		R2	n	

Table 21. Augmented Complex Matrices Multiplication

code	Complex Augmented Matrices Multiplication	operation	times	total CB
M6	$A1^a \cdot A2^a = A^a$	M1	4	$32n^3 - 4n^2$
		A2	2	
M7	$A1^a \cdot A2^a = A_s^a$	M1	4	$32n^3 - 6n^2 + n$
		A3	1	
		A6	1	
M8	$A1^a \cdot A_s^a = A^a$	M1	2	$32n^3 - 12n^2$
		M3	2	
		A2	2	
M9	$A1^a \cdot A1_s^a = A_s^a$	M5	1	$8n^3 - 4n^2 + 2n$
		M1	1	
		M3	2	
		A2	2	

Table 22 summarizes complex matrices inversions. Table 23 presents the complex general matrix inversion and Table 24 presents the complex Hermitian matrix inversion.

Table 22. Complex General Matrices Inversion

Complex Matrices Operation	Calculation Burden
C^{-1} $(n \times n)$	$\frac{60n^3 - 42n^2 + 12n}{6}$
H^{-1} $(n \times n)$	$\frac{26n^3 - 24n^2 + 4n}{6}$

Complex General Matrix Inversion

Recursive algorithm

$$M = \begin{bmatrix} A & b \\ c & d \end{bmatrix}$$

Dimensions:

M nxn

A $(n-1)$x$(n-1)$

b $(n-1)$x1

c 1x$(n-1)$

d 1x1

$$M^{-1} = \begin{bmatrix} A^{-1} + s\left(\frac{1}{s}A^{-1}b\right)\left(\frac{1}{s}cA^{-1}\right) & -\frac{1}{s}A^{-1}b \\ -\frac{1}{s}cA^{-1} & \frac{1}{s} \end{bmatrix}, s = d - cA^{-1}b$$

Table 23. Complex General Matrix Inversion

Matrix operation	oper	times	total CB
A^{-1} (NxN)			T(N)
$A^{-1}b$ (NxN) · (Nx1)	C1	$(N-1)N$	$8N^2-2N$
	C2	N^2	
cA^{-1} (1xN) · (NxN)	C1	$(N-1)N$	$8N^2-2N$
	C2	N^2	
$c(A^{-1}b)$ Real (1xN) · (Nx1)	C1	N − 1	8N − 2
	C2	N	
$s = d - cA^{-1}b$ real (1x1) + (1x1)	C1	1	2
$\frac{1}{s}$ (1x1)/(1x1)	C10	1	5
$\frac{1}{s}(A^{-1}b)$ $-\frac{1}{s}(A^{-1}b)$ (1x1) · (Nx1)	C2	N	6N
$\frac{1}{s}(cA^{-1})$ $-\frac{1}{s}(cA^{-1})$ (1x1) · (1xN)	C2	N	6N
$\left(\frac{1}{s}A^{-1}b\right)\left(\frac{1}{s}cA^{-1}\right)$ (Nx1) · (1xN)	C2	N^2	$6N^2$
$s\left(\frac{1}{s}A^{-1}b\right)\left(\frac{1}{s}cA^{-1}\right)$ (1x1) · (NxN)	C2	N^2	$6N^2$
$A^{-1}+s\left(\frac{1}{s}A^{-1}b\right)\left(\frac{1}{s}cA^{-1}\right)$ (NxN) + (NxN)	C1	N^2	$2N^2$
			$30N^2+16N+5$

$N = n - 1$
$T(n) = T(n-1) + f(n)$
$f(n) = 30(n-1)^2 + 16(n-1) + 5 = 30n^2 - 44n + 19$
$f(1) = 5$
$T(n) = T(n-1) + 30n^2 - 44n + 19, T(1) = 5$

$T(1) = 1$
$T(2) = T(1) + f(2)$
$T(3) = T(2) + f(3) = T(1) + f(2) + f(3)$
...
$T(n) = T(1) + f(2) + f(3) + \cdots + f(n)$
$T(n) = \{T(1) - f(1)\} + \{f(1) + f(2) + f(3) + \cdots + f(n)\}$

$$= 30\sum_{i=1}^{n} i^2 - 44\sum_{i=1}^{n} i + 19\sum_{i=1}^{n} 1$$

$$= 30\frac{n(n+1)(2n+1)}{6} - 44\frac{n(n+1)}{2} + 19n$$

$$= 10n^3 - 7n^2 + 2n$$

Complex Hermitian Matrix Inversion
Recursive algorithm
$M = \begin{bmatrix} A & b \\ b^H & d \end{bmatrix}, M = M^H, A = A^H$

Dimensions:
M nxn, A $(n-1)$x$(n-1)$, b $(n-1)$x1, b^H 1x$(n-1)$, d 1x1

$$M^{-1} = \begin{bmatrix} A^{-1} + s\left(\frac{1}{s}A^{-1}b\right)\left(\frac{1}{s}b^HA^{-1}\right) & -\frac{1}{s}A^{-1}b \\ -\frac{1}{s}b^HA^{-1} & \frac{1}{s} \end{bmatrix}$$

$s = d - b^HA^{-1}b$

Table 24. Complex Hermitian Matrix Inversion

Matrix operation	**operation**	**times**	**total CB**
A^{-1} (NxN)			T(N)
$A^{-1}b$ (NxN) · (Nx1)	C1	$(N-1)N$	$8N^2-2N$
	C2	N^2	
$b^HA^{-1} = (A^{-1}b)^H$ (1xN) · (NxN)			
$b^H(A^{-1}b)$ Real (1xN) · (Nx1)	C3	N − 1	4N − 1
	C6	N	
$s = d - b^HA^{-1}b$ real (1x1) + (1x1)	R1	1	1
$\frac{1}{s}$ Real (1x1)/(1x1)	R3	1	1
$\frac{1}{s}(A^{-1}b)$ $-\frac{1}{s}(A^{-1}b)$ (1x1) · (Nx1)	C5	N	2N
$\frac{1}{s}(b^HA^{-1}) = \left[\frac{1}{s}(A^{-1}b)\right]^H$ $-\frac{1}{s}(b^HA^{-1}) = -\left[\frac{1}{s}(A^{-1}b)\right]^H$ (1x1) · (1xN)			
$\left(\frac{1}{s}A^{-1}b\right)\left(\frac{1}{s}b^HA^{-1}\right)$ Hermitian (Nx1) · (1xN)	C2	$\frac{N^2-N}{2}$	$3N^2$
	C6	N	
$s\left(\frac{1}{s}A^{-1}b\right)\left(\frac{1}{s}b^HA^{-1}\right)$ Hermitian (1x1) · (NxN)	C5	$\frac{N^2+N}{2}$	N^2+N
$A^{-1}+s\left(\frac{1}{s}A^{-1}b\right)\left(\frac{1}{s}b^TA^{-1}\right)$ Hermitian (NxN) + (NxN)	R1	N	N^2
	C1	$\frac{N^2-N}{2}$	N^2
			$13N^2+5N+1$

$N = n - 1$
Then
$T(n) = T(n-1) + f(n)$
with
$f(n) = 13(n-1)^2 + 5(n-1) + 1 = 13n^2 - 21n + 9$
$f(1) = 1$
$T(n) = T(n-1) + 13n^2 - 21n + 9, T(1) = 1$

$T(1) = 1$
$T(2) = T(1) + f(2)$
$T(3) = T(2) + f(3) = T(1) + f(2) + f(3)$
...

$T(n) = T(1) + f(2) + f(3) + \cdots + f(n)$
$T(n) = \{T(1) - f(1)\} + \{f(1) + f(2) + f(3) + \cdots + f(n)\}$

$$= 13\sum_{i=1}^{n} i^2 - 21\sum_{i=1}^{n} i + 9\sum_{i=1}^{n} 1$$

$$= 13\frac{n(n+1)(2n+1)}{6} - 21\frac{n(n+1)}{2} + 9n = \frac{26n^3 - 24n^2 + 4n}{6}$$

Finally, Table 25 presents special augmented complex matrix inversion and Table 26 presents augmented complex matrix inversion.

Table 25. Special Augmented Complex Matrix Inversion

Complex Matrices Operation	**Calculation Burden**
$S^{a-1} = \begin{bmatrix} H & S \\ \bar{S} & \bar{H} \end{bmatrix}^{-1}$ $(2n \times 2n)$ $S^{a-1} = \begin{bmatrix} h & s \\ \bar{s} & \bar{h} \end{bmatrix}$ $h = [H - S\bar{H}^{-1}\bar{S}]^{-1}$ $s = -[H - S\bar{H}^{-1}\bar{S}]^{-1}S\bar{H}^{-1}$	$\frac{74n^3 - 39n^2 - 5n}{3}$
$\bar{H}^{-1}$	$\frac{26n^3 - 24n^2 + 4n}{6}$
$S\bar{H}^{-1}$	$8n^3 - 6n^2$
$S\bar{H}^{-1}\bar{S}$	$4n^3 - n^2$
$H - S\bar{H}^{-1}\bar{S}$	n^2
$[H - S\bar{H}^{-1}\bar{S}]^{-1}$	$\frac{26n^3 - 24n^2 + 4n}{6}$
$-[H - S\bar{H}^{-1}\bar{S}]^{-1}S\bar{H}^{-1}$	$4n^3 + n^2 - 3n$

Table 26. Augmented Complex Matrix Inversion

Complex Matrices Operation	**Calculation Burden**
$A^{a-1} = \begin{bmatrix} C1 & C2 \\ \overline{C2} & \overline{C1} \end{bmatrix}^{-1}$ $(2n \times 2n)$ $A^{a-1} = \begin{bmatrix} C3 & C4 \\ \overline{C4} & \overline{C3} \end{bmatrix}^{-1}$ $\overline{C3} = [\overline{C1} - \overline{C2}C1^{-1}C2]^{-1}$ $\overline{C4} = -\overline{C3}\,\overline{C2}\,C1^{-1}$	$\frac{132n^3 - 54n^2 + 12n}{3}$
$C1^{-1}$	$\frac{60n^3 - 42n^2 + 12n}{6}$
$\overline{C2}C1^{-1}$	$8n^3 - 2n^2$
$\overline{C2}C1^{-1}C2$	$8n^3 - 2n^2$
$\overline{C1} - \overline{C2}C1^{-1}C2$	$2n^2$
$\overline{C3} = [\overline{C1} - \overline{C2}C1^{-1}C2]^{-1}$	$\frac{60n^3 - 42n^2 + 12n}{6}$
$\overline{C4} = -\overline{C3}\,\overline{C2}\,C1^{-1}$	$8n^3 - 2n^2$