Kalman Filter Implementation Methods

Ali Saeizadeh
Instructor: Dr.Olfat

July, 2021

Abstract

In designing Kalman filters, two critical types of computational questions arise. First, what is the nature of the errors that can be encountered, and what is their effect on the filter's performance? Secondly, how may one minimize the computational burden of design? Of course, the two questions are not entirely independent since, for example, procedures involving a small number of computations have poor error performance. In this report, we sketch some of the ideas that are useful for dealing with these questions. To deal with these problems in numeric inherent to the Kalman filter algorithm, alternate recursion relationships have been developed to propagate and update a state estimate and error covariance square root or inverse covariance square root instead of the covariance or its inverse themselves. Although equivalent algebraically to the conventional Kalman filter recursion, these square root filters exhibit improved numerical precision and stability, particularly in ill-conditioned problems (i.e., the cases described that yield erroneous results due to finite word length). The square root approach can yield twice the effective precision of the conventional filter in ill-conditioned problems. In other words, the same precision can be achieved with approximately half the word length. Moreover, this method is entirely successful in maintaining the positive semi definiteness of the error covariance. [2], [3]

1 Introduction

The Kalman Filter that has been discussed in class suffers from numerical difficulties. Measurement updating of the covariance matrix requires long word length to maintain acceptable numerical accuracy. Low precision arithmetic (short word length in computers) yields an error source in the Kalman filter procedure, usually not included in designers' error budget. Computer roundoff is the main reason that caused this problem. Alternative implementation methods were developed to cope with roundoff errors. The numerical solution of the Kalman filter equation tends to be more robust against roundoff errors if the so-called *square roots* of the covariance matrix are used as the dependent variables. Also, Information filtering is an alternative state vector implementation that improves numerical stability properties. It is beneficial for problems with considerable initial estimation uncertainty [1].

2 Computer Roundoff

In computers, fixed or floating-point data have been used with a fixed number of bits. Computer roundoff makes the precision of our computation limited and provokes an error source. The difference between the actual value of the result and the value approximated by the processor is called *roundoff error*. [1]

Computer roundoff for floating-point arithmetic is often characterized by a single parameter $\varepsilon_{\text{roundoff}}$, called the unit roundoff error, and defined in different sources as the largest number such that either

$$1 + \varepsilon_{\text{roundoff}} \equiv 1 \text{ in machine precision} \tag{1}$$

In MATLAB, this parameter can be used by *eps* variable:

```
d = eps
d = 2.2204e-16
```

$-log2(eps)$ shows number of bits in the mantissa of the standard data word.

```
d = -log2(eps)
d = 52
```

3 Effect of Computer Roundoff on Kalman Filter Performance

3.1 An Example of Roundoff Error Effect on Calculations

As discussed in previous sections, roundoff error makes Kalman filter unusable due to the low precision arithmetic computation, but how it's generated this error source. Consider measurement matrix H:

$$H = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1+\delta \end{bmatrix}$$

and covarince matrices P and R:

$$P_0 = I_3 \quad \text{and} \quad R = \delta^2 I_2$$

where $\delta^2 < \varepsilon_{\text{roundoff}}$ but $\delta > \varepsilon_{\text{roundoff}}$. In this case, the product HP_0H^{T} with roundoff will equal

$$\begin{bmatrix} 3 & 3+\delta \\ 3+\delta & 3+2\delta \end{bmatrix}$$

which is singular [1], [4]. The result is unchanged when R is added to HP_0H^{T}. In this case, then, the filter observational update fails because the matrix $HP_0H^{\mathrm{T}} + R$ is not invertible. This condition will be simulated in different precision and error will be measured.

Also, it should be noted that in all problems we may don't face this kind of problem. A problem like this, which has a high sensitivity of roundoff error called *ill-conditioned* problems.

3.2 Some Terminology and Definitions

First of all, some general terms should be defined and cleared to have the same terminology and language.

Robustness: refers to the relative insensitivity of the solution to errors of some sort.

Numerical stability: refers to robustness against roundoff errors.

Using long word length can reduce relative roundoff errors (i.e., more bits in the mantissa of the data format). Still, the accuracy of the result is also influenced by the accuracy of the initial parameters used and the procedural details of the implementation method. Mathematically equivalent implementation methods can have very different numerical stabilities at the same precision.

The kind of problems that are considered to be ill-conditioned problems include the following:

1. Large uncertainties in the values of the matrix parameters Φ, Q, H, or R. Such modeling errors are not accounted for in the derivation of the Kalman filter.

2. Large ranges of the actual values of these matrix parameters, the measurements, or the state variables - all of which can result from poor choices of scaling or dimensional units.

3. Ill-conditioning of the intermediate result $R^* = HPH^{\mathrm{T}} + R$ for inversion in the Kalman gain formula.

4. Ill-conditioned theoretical solutions of the matrix Kalman Equations equation-without considering numerical solution errors. With numerical errors, the solution may become indefinite, which can destabilize the filter estimation error.

5. Large matrix dimensions. The number of arithmetic operations grows as the square or cube of matrix dimensions, and each operation can introduce roundoff errors.

6. Poor machine precision, which makes the relative roundoff errors larger. [1]

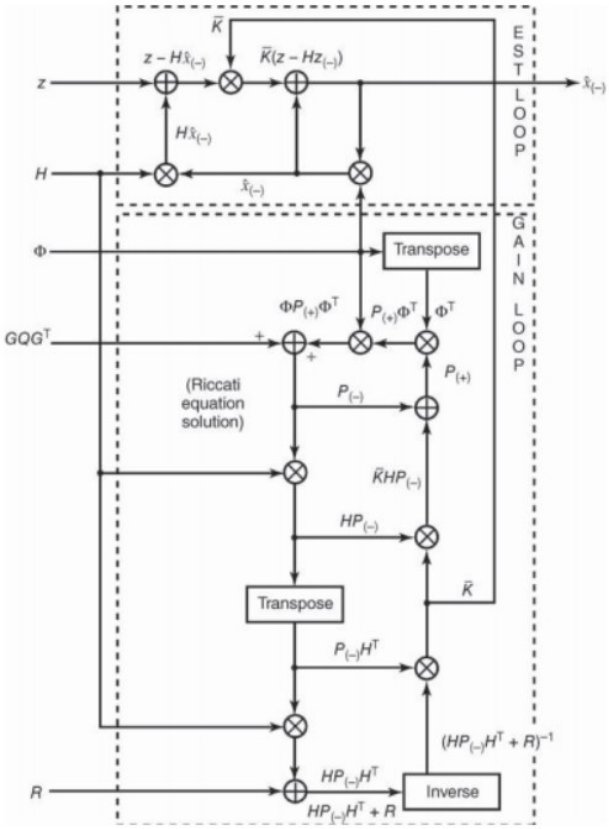

Figure 1: Kalman filter data flow.

3.3 Roundoff Error Propagation in Kalman Filters

Roundoff errors can cause the computed value of covariance matrix, P to have a negative characteristic value. The Kalman Filter equation is stable, and the problem will eventually rectify itself. However, the effect on the actual estimation error can be a more severe problem. Because P is a factor in the Kalman gain $\bar{K}$, a negative characteristic value of P can cause the gain in the prediction error feedback loop to have the wrong sign. However, in this transient condition, the estimation loop is immediately destabilized. In Figure 1 Kalman data flow has been illustrated, the estimate $\hat{x}$ converges toward the actual value x until the gain changes sign. Then the error diverges momentarily. The gain computations may eventually recover with the correct sign, but the accumulated error due to divergence is not accounted for in the gain computations. The gain is not as significant as it should be, and convergence is slower than it should be [1].

4 Implementation Methods

In this section, three typical implementations of Kalman Filter have been introduced: Standard Kalman filter, Information filter, and a specific form of square root filter called Carlson-Schmit, which is widespread and easy-to-implement variety of square root filter.

4.1 Standard Kalman Filter

The discrete-time model for a linear stochastic system has the form

$$x_k = \Phi_{k-1} x_{k-1} + G_{k-1} w_{k-1},$$
$$z_k = H_k x_k + v_k,$$

where the zero-mean uncorrelated random processes $\{w_k\}$ and $\{v_k\}$ have covariances Q_k and R_k, respectively, at time t_k. The corresponding Kalman filter equations have the form

$$\begin{aligned}
\hat{x}_{k(-)} &= \Phi_{k-1}\hat{x}_{(k-1)(+)}, \\
P_{k(-)} &= \Phi_{k-1}P_{(k-1)(+)}\Phi_{k-1}^{\mathrm{T}} + G_{k-1}Q_{k-1}G_{k-1}^{\mathrm{T}}, \\
\hat{x}_{k(+)} &= \hat{x}_{k(-)} + \bar{K}_k\left(z_k - H_k\hat{x}_{k(-)}\right), \\
\bar{K}_k &= P_{k(-)}H_k^{\mathrm{T}}\left(H_kP_{k(-)}H_k^{\mathrm{T}} + R_k\right)^{-1}, \\
P_{k(+)} &= P_{k(-)} - \bar{K}_kH_kP_{k(-)},
\end{aligned}$$

where the (-) indicates the a prior values of the variables (before the information in the measurement is used) and the (+) indicates the a posterior values of the variables (after the information in the measurement is used). The variable $\bar{K}$ is the Kalman gain. Implementation of Kalman filter in each iteration will be like:

```
K  = P*H'/(H*P*H'+R);
xK = x0 + K*(z - H*x0);
PK = P - K*H*P;
```

4.2 Information Filter

4.2.1 The Matrix Inversion Lemma and Information Filter

By applying different matrix equities to the usual Kalman filter equations, new filter equations will be obtained in terms of the inverse of the covariance matrix, $\sum_{k/k-1}^{-1}$ and $\sum_{k/k}^{-1}$. These inverses are called *information matrices*.

4.2.2 The Matrix Inversion Lemma

In terms of an $n \times n$ matrix Σ, a $p \times p$ matrix R, and an $n \times p$ matrix H, the following equalities hold on the assumption that the various inverses exists:

$$\left(I + \Sigma HR^{-1}H'\right)^{-1}\Sigma = \left(\Sigma^{-1} + HR^{-1}H'\right)^{-1} = \Sigma - \Sigma H\left(H'\Sigma H + R\right)^{-1}H'\Sigma \tag{2}$$

Multiplication on the right by HR^{-1} and application of the identity

$$H'\Sigma HR^{-1} = \left[\left(H'\Sigma H + R\right)R^{-1} - I\right] \tag{3}$$

yields a variation of (2) as:

$$\left(I + \Sigma HR^{-1}H'\right)^{-1}\Sigma HR^{-1} = \left(\Sigma^{-1} + HR^{-1}H'\right)^{-1}HR^{-1} = \Sigma H\left(H'\Sigma H + R\right)^{-1} \tag{4}$$

That the first equality of (2) holds is immediate. That

$$\left(I + \Sigma HR^{-1}H'\right)\left[I - \Sigma H\left(H'\Sigma H + R\right)^{-1}H'\right] = I \tag{5}$$

holds can be verified in one line by direct verification. These two results together yield the remaining equality in (2). [2]

4.2.3 The Information Filter

Applying the above matrix inversion lemma to Kalman filter equations yields another filter called information filter [2]. However, This is just for the case that input noise and output noise are independent. i.e $S = 0$. However, there is some way to cope with dependent noises.

$$\Sigma_{k'|k} = \Sigma_{k|k-1} - \Sigma_{k|k-1}H_k\left(H_k'\Sigma_{k|k-1}H_k + R_k\right)^{-1}H_k'\Sigma_{k|k-1} \tag{6}$$

Consider equation (6). Applying the matrix inversion lemma yields an equation for information matrix $\sum_{k|k}^{-1}$ as

$$\Sigma_{k/k}^{-1} = \Sigma_{k/k-1}^{-1} + H_k R_k^{-1} H_k' \tag{7}$$

A further application to the identity:

$$\Sigma_{k+1/k} = F_k \Sigma_{k/k} F_k' + G_k Q_k G_k' \tag{8}$$

with

$$A_k = \left[F_k^{-1}\right]' \Sigma_{k/k}^{-1} F_k^{-1} \tag{9}$$

identified with Σ of (2) and $G_k Q_k G_k'$ identified with $HR^{-1}H'$ of (2), yields an expression for the information matrix $\sum_{k+1/k}^{-1}$ as

$$\begin{aligned} \Sigma_{k+1/k}^{-1} &= \left[A_k^{-1} + G_k Q_k G_k'\right]^{-1} \\ &= \left[I - A_k G_k \left[G_k' A_k G_k + Q_k^{-1}\right]^{-1} G_k'\right] A_k \end{aligned} \tag{10}$$

or

$$\Sigma_{k+1/k}^{-1} = \left[I - B_k G_k'\right] A_k \tag{11}$$

where

$$B_k = A_k G_k \left[G_k' A_k G_k + Q_k^{-1}\right]^{-1} \tag{12}$$

From filter gain expression which is:

$$K_k = F_k \Sigma_{k/k-1} H_k \left(H_k' \Sigma_{k/k-1} H_k + R_k\right)^{-1} \tag{13}$$

applying the matrix inversion lemma yields to:

$$K_k = F_k \left(\Sigma_{k/k-1}^{-1} + H_k R_k^{-1} H_k'\right)^{-1} H_k R_k^{-1} = F_k \Sigma_{k/k}^{-1} H_k R_k^{-1} \tag{14}$$

It's hard to write expressions with S or M; instead, we find recursions by following quantities

$$\begin{aligned} \hat{a}_{k/k-1} &= \Sigma_{k/k-1}^{-1} \hat{x}_{k/k-1} \\ \hat{a}_{k/k} &= \Sigma_{k/k}^{-1} \hat{x}_{k/k} \end{aligned} \tag{15}$$

from which the state estimates may be recovered by the solution of algebraic equations without the need for actually taking inverses of $\Sigma_{k/k}^{-1}$ and $\Sigma_{k/k-1}^{-1}$ to obtain $\Sigma_{k/k}$ and $\Sigma_{k/k-1}$ explicitly. Application of information matrix equations yields

$$\begin{aligned} \hat{a}_{k+1/k} &= \left[I - B_k G_k'\right] A_k F_k \hat{x}_{k/k} \\ &= \left[I - B_k G_k'\right] \left(F_k^{-1}\right)' \Sigma_{k/k}^{-1} \hat{x}_{k/k} \end{aligned} \tag{16}$$

or

$$\hat{a}_{k+1/k} = \left[I - B_k G_k'\right] \left[F_k^{-1}\right]' \hat{a}_{k/k} \tag{17}$$

The measurement-update equations:

$$\hat{x}_{k/k} = \hat{x}_{k/k-1} + \Sigma_{k/k-1} H_k \left(H_k' \Sigma_{k/k-1} H_k + R_k\right)^{-1} \left(z_k - H_k' \hat{x}_{k/k-1}\right) \tag{18}$$

yields to:

$$\hat{a}_{k/k} = \hat{a}_{k/k-1} + H_k R_k^{-1} z_k \tag{19}$$

[2]
In Discrete time we define information matrix this way:

$$Y \stackrel{\text{def}}{=} P^{-1} \tag{20}$$

$$\begin{aligned} \hat{x}_{k(+)} &= \hat{x}_{k(-)} + H_k^{\mathrm{T}} R_k^{-1} z_k \\ Y_{k(+)} &= Y_{k(-)} + H_k^{\mathrm{T}} R_k^{-1} H_k \end{aligned} \tag{21}$$

[1]
and in MATLAB, implementation will be like:

```
xI = x0 + (H'/R)*z;
YI = inv(P) + (H'/R)*H;
PI = inv(YI);
```

4.3 Carlson–Schmidt Square-root Filtering

Before introducing square root filter we need to distinguish different type of square root of a matrix.

4.3.1 Factorization Methods for Square-root Filtering

We need to define Cholesky factor to implements square-root filter. Cholesky factors are triangular (and therefore square) with positive diagonal entries. They may be either

1. lower triangular (i.e., with zeros above the diagonal), or
2. upper triangular (i.e., with zeros below the diagonal).

Also, there are other Cholesky factors like Generalized Cholesky factors and Modified Cholesky factorization. But we don't need them to implement Carlson–Schmidt Square-root Filter. A Cholesky decomposition algorithm is a procedure for calculating the elements of a triangular Cholesky factor of a symmetric, nonnegative-definite matrix. It solves the Cholesky decomposition equation $P = CC^{\mathrm{T}}$ for a triangular matrix C, given the matrix P. [1] For Upper Triangular Cholesky Decomposition Algorithms we have:

```
for j=m:-1:1,
    for i=j:-1:1,
    sigma = P(i,j);
      for k=j+1:m,
      sigma = sigma - C(i,k)*C(j,k);
      end;
     C(j,i) = 0;
      if (i==j)
        C(i,j) = sqrt(max([0,sigma]));
      elseif (C(j,j) == 0)
        C(i,j) = 0;
      else
        C(i,j) = sigma/C(j,j);
      end;
    end;
```

Computational complexity: $\frac{1}{6}m(m-1)(m+4)$ flops $+m\sqrt{\cdot}$

4.3.2 Carlson–Schmidt Square-Root Filtering

There are different forms of square root filtering that all of them reform Kalman filter equations such that instead of working with P, we use different forms of the square root of P (it's not unique). We present just one of them, Carlson–Schmidt square-root filtering, which uses Cholesky factors of P (Upper Triangular decomposition), which is the more favored implementation form and easy to implement.

The algorithm is due to Carlson [5]. It generates an upper triangular Cholesky factor W for the Potter factorization and has generally lower computational complexity than the Potter algorithm. It is a specialized and simplified form of an algorithm used by Agee and Turner [6] for Kalman filtering. It is a rank 1 modification algorithm, like the Potter algorithm, but it produces a triangular Cholesky factor. which can be solved in terms of the elements of the upper triangular Cholesky factor W :

$$W_{ij} = \begin{cases} 0, & i > j, \\ \sqrt{\frac{R+\sum_{k=1}^{j-1} v_k^2}{R+\sum_{k=1}^{j} v_k^2}}, & i = j, \\ \frac{-v_i v_j}{\left(R+\sum_{k=1}^{j-1} \mathbf{v}_k^2\right)\left(R+\sum_{k=1}^{j} \mathbf{v}_k^2\right)}, & i < j. \end{cases}$$

Given the above formula for the elements of W, one can derive the formula for the elements of $C_{(+)} = C_{(-)}W$, the upper triangular Cholesky factor of the a posteriori covariance matrix of estimation uncertainty $P_{(+)}$, given $C_{(-)}$, the upper triangular Cholesky factor of the a priori covariance matrix $P_{(-)}$.

Because both C and W are upper triangular, the elements $C_{ik} = 0$ for $k < i$ and the elements $W_{kj} = 0$ for $k > j$. Consequently, for $1 \leq i \leq j \leq n$, the element in the i th row and j th column of the matrix product $C_{(-)}W = C_{(+)}$ will be

$$\begin{aligned} C_{ij(+)} =& \sum_{k=i}^{j} C_{ik(-)}W_{kj} + \text{ terms with zero factors} \\ =& C_{ij(-)}W_{jj} + \sum_{k=i}^{j-1} C_{ik(-)}W_{kj} \\ =& C_{ij(-)}\sqrt{\frac{R + \sum_{k=1}^{j-1}\mathbf{v}_k^2}{R + \sum_{k=1}^{j}\mathbf{v}_k^2}} \\ & - \sum_{k=i}^{j-1} \frac{C_{ik(-)}\mathbf{v}_k\mathbf{v}_j}{\left(R + \sum_{k=1}^{j-1}\mathbf{v}_k^2\right)\left(R + \sum_{k=1}^{j}\mathbf{v}_k^2\right)} = \left(R + \sum_{k=1}^{j}\mathbf{v}_k^2\right)^{-1/2} \\ & \times \left[C_{ij(-)}\sqrt{R + \sum_{k=1}^{j-1}\mathbf{v}_k^2} - \frac{\mathbf{v}_j \sum_{k=i}^{j-1} C_{ik(-)}\mathbf{v}_k}{\left(R + \sum_{k=1}^{j-1}\mathbf{v}_k^2\right)^{1/2}}\right]. \end{aligned} \tag{22}$$

This is a general formula for the upper triangular a posteriori Cholesky factor of the covariance matrix of estimation uncertainty, in terms of the upper triangular a priori Cholesky factor $C_{(-)}$ and the vector $\mathbf{v} = C^{\mathrm{T}}H^{\mathrm{T}}$, where H is the measurement sensitivity matrix (a row vector). An algorithm implementing the formula is given in following code. This algorithm performs the complete observational update, including the update of the state estimate, in place. (Note that this algorithm forms the product $\mathbf{v} = C_{(-)}^{\mathrm{T}}H^{\mathrm{T}}$ internally, computing and using the components $\sigma = \mathbf{v}_j$ as needed, without storing the vector $\mathbf{v}$. It does store and use the vector $\mathbf{w} = C_{(-)}\mathbf{v}$, the unscaled Kalman gain, however.)

It is possible-and often desirable-to save array space by storing triangular matrices in singly subscripted arrays. An algorithm (in FORTRAN) for such an implementation of this algorithm is given in Carlson's original paper [1], [5].

Carlson's Fast Triangular Observational Update:

z Value of scalar measurement
R Variance of scalar measurement uncertainty
H Scalar measurement sensitivity vector ($1 \times n$ matrix)
C Cholesky factors of $P_{(-)}$ (input) and $P_{(+)}$ (output)
x State estimates $x_{(-)}$ (input) and $x_{(+)}$ (output)
w Unscaled Kalman gain (internal)

```
alpha = R;
delta = z;
for j = 1 : n,
    delta = delta - H (j)*x (j);
    sigma = 0;
    for i = 1 : j,
        sigma = sigma + C (i, j)*H (i);
        end;
        beta = alpha;
        alpha = alpha + sigma^2;
        gamma = sqrt (alpha*beta);
        eta = beta/gamma;
```

```
        zeta = sigma/gamma;
        w (j) = 0;
        for i = 1 : j,
        tau = C (i, j);
        C (i, j) = eta*C (i, j) - zeta*w (i);
        w (i) = w (i) + tau*sigma;
    end;
end;
epsilon = delta/alpha;
for i = 1 : n,
x (i) = x (i) + w (i)*epsilon;
end;
```

Computational complexity: $\left(2n^2 + 7n + 1\right)$ flops $+n_{\surd}$. [1]

5 Simulations

Two different experiments have been simulated. In the first one, the conditioning parameter is the same for all implementations. However, the precision is going to change using the *vpa* function in MATLAB. In the second experiment, On the variably ill-conditioned problem mentioned in earlier this report, as the conditioning parameter $\delta \to 0$. All solution methods were implemented with the same precision (64-bit floating-point) in MATLAB to check numerical stability of each methods. The problem parameter that has been used in both experiments is:

$$H = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1+\delta \end{bmatrix}$$

and covarince matrices P and R:

$$P_0 = I_3 \quad \text{and} \quad R = \delta^2 I_2$$

5.1 Numerical Stability (Different Precision)

The precision changed using the *vpa* function every iteration (from 2 significant digits to 32). δ (conditioning parameter) is fixed to $\delta = 1e-5$. Each implementation runs for one iteration and is compared by a closed-form solution of the problem with actual values (no approximations and degrading precision). The relative RMS error (due to the closed form with actual values) has been calculated for each precision and plotted in figure 2. As shown in figure 2, the square root filter works much better than the standard Kalman filter and Information filter in low precision. Also, The Kalman Filter works better than the Information filter. However, In high precision, both the Information filter and Kalman filter works better than the square root filter.

5.2 Ill-conditioned Robustness (Different Conditioning Parameter)

Figure 3 illustrates how condition parameter (with fixed precision) affects different methods. δ change (10^{-20} to 1) when delta is higher means that the problem is in better condition (well-conditioned). When δ gets lower values, our problems seem to be more ill-conditioned. As Figure shows, Carlson has the best performance in ill-conditioned cases, and the information filter is the worst. But when the problems tend to be more well-conditioned, three implementations have the same result (in some precision).

6 Conclusion

Computer roundoff errors can and do seriously degrade the performance of Kalman filters. The solution of the KF equation is a major cause of numerical difficulties in the conventional Kalman filter implementation, from the standpoint of computational load as well as from the perspective of computational errors.

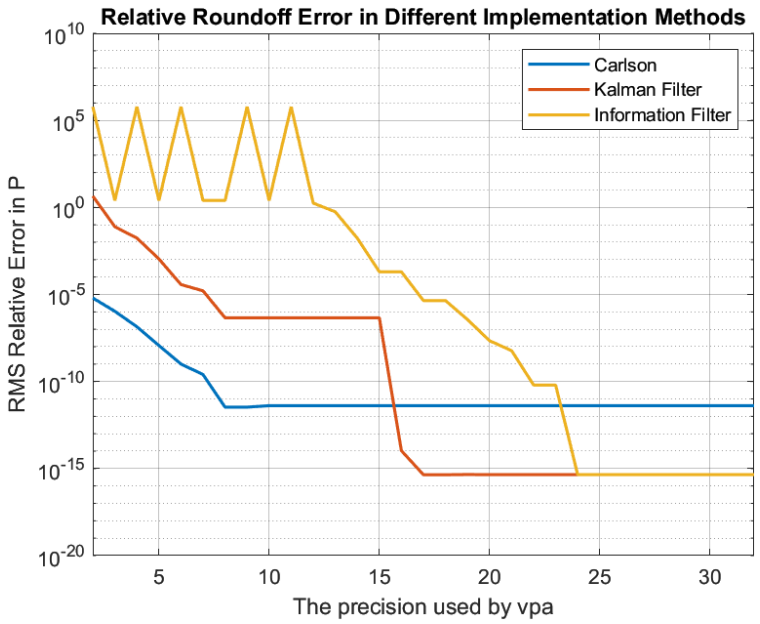

Figure 2: Relative Roundoff Error in Different Implementation Methods as precision

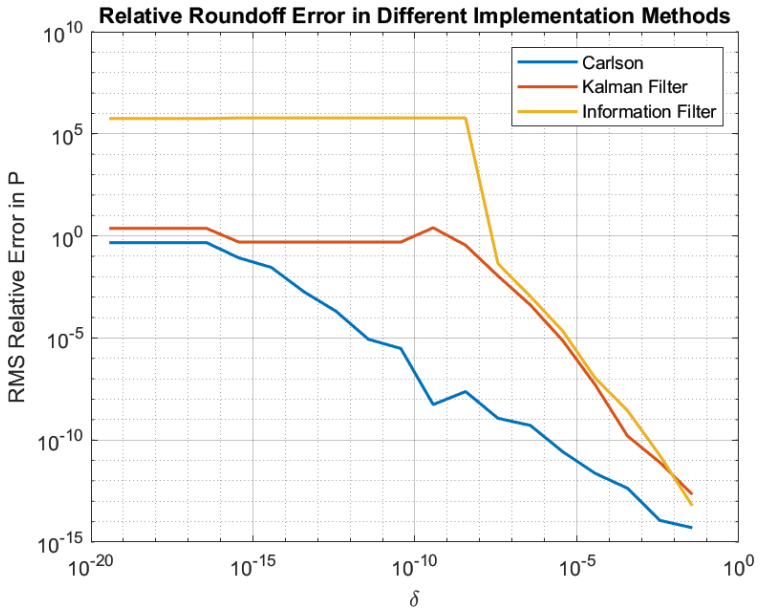

Figure 3: Relative Roundoff Error in Different Implementation Methods as conditioning parameter

Although Kalman filtering has been called "ideally suited to digital computer implementation", the digital computer is not ideally suited to the task. The conventional implementation of the Kalman filter—in terms of covariance matrices—is particularly sensitive to roundoff errors. The numerical solution of the KF equation tends to be more robust against roundoff errors if the so-called "square roots" of the covariance matrix are used as the dependent variables.

'Square-root" covariance filters, which use a decomposition of the covariance matrix of estimation uncertainty as a symmetric product of triangular Cholesky factors:

$$P = CC^{\mathrm{T}}$$

which can be used in ill-conditioned problems and also in low precision and low wavelength.

Information filtering is an alternative state vector implementation that improves numerical stability properties. It is beneficial for problems with considerable initial estimation uncertainty bur when we can have high precision.

References

[1] Grewal, Mohinder S., and Angus P. Andrews. Kalman filtering: Theory and Practice with MATLAB. John Wiley and Sons, 2014.

[2] Anderson, Brian, and John B. Moore. "Optimal filtering." Prentice-Hall Information and System Sciences Series (1979).

[3] Maybeck, P. S. (1979). Stochastic Models, Estimation and Control, Volume 1, volume 141 of. Mathematics in Science and Engineering.

[4] Verhaegen, Michel, and Paul Van Dooren. "Numerical aspects of different Kalman filter implementations." IEEE Transactions on Automatic Control 31.10 (1986): 907-917.

[5] Carlson, Neal A. "Fast triangular formulation of the square root filter." AIAA journal 11.9 (1973): 1259-1265.

[6] Agee, William S., and Robert H. Turner. Triangular decomposition of a positive definite matrix plus a symmetric dyad with applications to Kalman filtering. NATIONAL RANGE OPERATIONS DIRECTORATE WHITE SANDS MISSILE RANGE NM ANALYSIS AND COMPUTATION DIV, 1972.