# An Alternative to the Extended Kalman Filter for Approximation of Sinusoidal Signals

How can the autocorrelation function be used to approximate sinusoidal functions with additive noise as an alternative to the Extended Kalman Filter?

# Table of Contents

## Introduction

A digital filter is a group of equations that performs mathematical operations on a discrete signal. In its simplest form, a signal is a stream of measurements being taken, usually by a sensor. Digital filters typically aim to recover an underlying "true" signal from a stream of measurements that has been altered due to noise (i.e. the uncertainty in the sensor) (Kim & Bang). For example, a digital filter may receive measurements of the temperature in a room from a thermostat. However, the measurements do not exactly equal the "true" temperature of the room due to uncertainty in the thermostat. In this case, a digital filter would perform mathematical operations on the measurements that remove noise (i.e. the thermostat's uncertainty) to uncover the "true" temperature of the room (i.e. the "true" signal). Digital filters can be much more complicated, however, such as those that help to accurately track the position of satellites or robots (Kim & Bang).

The Extended Kalman Filter (EKF) is an existing digital filter that provides an estimate of a state (i.e. a value we want to know, like the temperature of a room) containing noise. Using initial estimates of a state and the sensor's error (i.e. uncertainty), the filter is able to recover an accurate estimate of the "true" signal/state (Kim & Bang). However, the EKF has limitations, including its reliance on the ability of the filter's user to accurately model the signal and its deterioration in accuracy for highly nonlinear functions like sinusoids.

To solve the EKF's problems, we propose a new method to estimate the parameters of a noisy sinusoidal signal using the autocorrelation function. Hence, the paper's research question arises: How can the autocorrelation function be used to approximate sinusoidal signals with additive noise as an alternative to the Extended Kalman Filter?

This paper begins with derivations of the Kalman Filter and EKF, before deriving a filter based on the autocorrelation function, and, finally, comparing the effectiveness of the filters.

## Derivation of the Kalman Filter

The Kalman Filter, when derived and/or applied in the literature, almost always represents a signal's state and variance in matrix form (Mosammam; Hamed; Ribeiro). In order to simplify the derivation and application of the Kalman Filter, we will only deal with scalars.

In order to optimally estimate the true value of a state, the Kalman Filter combines the two pieces of information we have about the system:

a) How we *think* the system will behave. We can create a model that describes the system and use the model to predict what each iteration of the state should be;

b) What we *observe* from the system. We can use the measurements from sensors measuring the state (e.g. acceleration from an accelerometer, temperature from a thermometer, etc.) to estimate the true value of the state. Unfortunately, measurements almost always contain unwanted noise, so we cannot solely rely on the measurements. This is why we combine uncertain measurements with predictions from a predetermined model.

The five equations that compose the Kalman Filter are recursive. They update our best estimate of the state using the previous value of the state and the new measurement (Ribeiro). This is superior to remembering and performing operations on all past state values because it minimizes storage and computation when implementing the filter in real-life situations. While many derivations of the Kalman Filter exist, we use the one synthesized by Dr. Erik Cheever

from Swarthmore College. Some parts of the following derivation are also modified for the purposes of this essay.

The first of five equations we will build intuitively. This is the equation that predicts the value of the state in the next time step (i.e. iteration). For now, we will only allow linear models. This means the filter can only process states that transform linearly from one time step to the next. The linear transformation (also known as the "state prediction" equation) is written as

$$\hat{x}_n = ax_{n-1} + b \tag{1}$$

where $\hat{x}_n$ is the predicted value of the state for time step $n$, which equals some linear transformation of the previous best estimate of the state $x_{n-1}$. $a$ and $b$ are constants that are set before the filter starts being used. The state prediction equation is combined with the measurement observed by the sensor at time step $n$.

$$z_n = y_n + w_n \tag{2}$$

The measurement is the true value of the state $y_n$ (which is impossible to measure directly) with added zero-mean measurement error $w_n$, representing the uncertainty in the sensor. There are now two values for the state at time step $n$, the predicted value $\hat{x}_n$ and measured value $z_n$. They can be combined linearly to produce the best estimate of the state at time step $n$:

$$x_n = \hat{x}_n + k_n(z_n - \hat{x}_n) \tag{3}$$

The best estimate of the state at time step $n$ is essentially a weighted average of the predicted and measured values of the state. $k_n$ is a real number between $1$ and $0$ that determines "how much" of each value goes into determining the new best estimate. If $k_n$ is $1$, then the best estimate equals the measurement $z_n$, and if it is $0$, then the best estimate equals the predicted

value $\hat{x}_n$. The filter must now determine the optimal value of $k_n$, one that computes the most "accurate" estimate of the state. We define the error in the state estimate as the difference between the real state and best estimate:

$$\varepsilon_n = y_n - x_n \tag{4}$$

The most accurate estimate of the state is one which minimizes the mean square error of $\varepsilon_n$:

$$E[\varepsilon_n^2] \rightarrow min$$

Now expand the left-hand side by substituting in Equation (4), then Equation (3):

$$\begin{aligned} E[\varepsilon_n^2] &= E[(y_n - x_n)^2] \\ &= E[(y_n - \hat{x}_n - k_n(z_n - \hat{x}_n))^2] \end{aligned}$$

We must now optimize $k_n$ such that it produces the minimum value for the mean square error. To do this, we will consider the interior of the expected value function a quadratic in terms of $k_n$:

$$\begin{aligned} y &= (\underbrace{y_n - \hat{x}_n}_{\alpha} - k_n \underbrace{(z_n - \hat{x}_n)}_{\beta})^2 \\ &= (\alpha - k_n\beta)^2 \\ &= \beta^2 k_n^2 - 2\alpha\beta x + \alpha^2 \end{aligned}$$

As shown above, the coefficient of the $k_n^2$ term will be positive for any $\beta$. Hence, the parabola will always open upwards. We can therefore identify the global minimum of the function by finding the point at which the derivative with respect to $k_n$ equals zero.

$$0 = \frac{d}{dk_n} E[(y_n - \hat{x}_n - k_n(z_n - \hat{x}_n))^2]$$

We can take the derivative of the expected value function by rewriting the expected value function in its continuous form as an integral, with arbitrary limits *g* and *h*:

$$\begin{aligned}
0 &= \frac{d}{dk_n}\int_g^h (y_n - \hat{x}_n - k_n(z_n - \hat{x}_n))^2 p(x)dx \\
&= \int_g^h \frac{\partial}{\partial k_n}(y_n - \hat{x}_n - k_n(z_n - \hat{x}_n))^2 p(x)dx \\
&= \int_g^h 2(\hat{x}_n - y_n + k_n(z_n - \hat{x}_n))(z_n - \hat{x}_n)p(x)dx \\
&= 2E[\hat{x}_n - y_n + k_n(z_n - \hat{x}_n))(z_n - \hat{x}_n)]
\end{aligned}$$

Now that the equation has been optimized to minimize the mean square error $E[\varepsilon_n^2]$, we will rearrange the equation to solve for $k_n$:

$$\begin{aligned}
0 &= 2E[(\hat{x}_n - y_n + k_n(z_n - \hat{x}_n))(z_n - \hat{x}_n)] \\
&= 2E[y_n z_n - \hat{x}_n z_n - y_n \hat{x}_n + \hat{x}_n^2] - 2k_n E[{z_n}^2 - 2\hat{x}_n z_n + \hat{x}_n^2] \\
k_n &= \frac{E[y_n z_n - \hat{x}_n z_n - y_n \hat{x}_n + \hat{x}_n^2]}{E[{z_n}^2 - 2\hat{x}_n z_n + \hat{x}_n^2]}
\end{aligned}$$

Substituting in Equation (2) for all $z_n$ gives

$$\begin{aligned}
k_n &= \frac{E[y_n(y_n + w_n) - \hat{x}_n(y_n + w_n) - y_n\hat{x}_n + \hat{x}_n^2]}{E[(y_n + w_n)^2 - 2\hat{x}_n(y_n + w_n) + \hat{x}_n^2]} \\
&= \frac{E[{y_n}^2 - 2\hat{x}_n y_n + \hat{x}_n^2] + \overbrace{E[w_n(y_n - \hat{x}_n)]}^{0}}{E[{y_n}^2 - 2\hat{x}_n y_n + \hat{x}_n^2 + w_n^2] + \underbrace{2E[w_n(y_n - \hat{x}_n)]}_{0}}
\end{aligned}$$

The two indicated expressions above are equal to zero because the expected value of two uncorrelated variables, in this case $w$ and $(y_n - \hat{x}_n)$, is zero.

$$
\begin{aligned}
k_n &= \frac{E[{y_n}^2 - 2\hat{x}_n y_n + \hat{x}_n^2]}{E[{y_n}^2 - 2\hat{x}_n y_n + \hat{x}_n^2] + E[w_n^2]} \\
&= \frac{E[(y_n - \hat{x}_n)^2]}{E[(y_n - \hat{x}_n)^2] + E[w_n^2]} \\
&= \frac{E[\hat{\varepsilon}_n^2]}{E[\hat{\varepsilon}_n^2] + R} \\
&= \frac{\hat{P}_n}{\hat{P}_n + R}
\end{aligned} \tag{5}
$$

The formula to calculate the optimal value of $k_n$ is presented in (5). $\hat{P}_n$ is new (and easier to write) notation for the mean square error of the *predicted* value of the state. $R$ is a simpler notation used to denote the mean square of the measurement error $w_n$. It is essentially the uncertainty of the sensor measurement. To simplify the notation, we will also refer to mean square error of the best estimate as

$$P_n = E[\varepsilon_n^2]$$

To derive a formula to calculate $\hat{P}_n$, begin with the definition from Equation (5)

$$
\begin{aligned}
\hat{P}_n &= E[\varepsilon_n^2] \\
&= E[(y_n - \hat{x}_n)^2]
\end{aligned} \tag{6}
$$

Substituting the equation which models the system (Equation (1)) for $y_n$ and adding a zero-mean error $v_n$ to account for error that accumulates between time step $n-1$ and $n$ gives

$$
\begin{aligned}
\hat{P}_n &= E[((ay_{n-1} + b + v_n) - (ax_{n-1} + b))^2] \\
&= E[(a(y_{n-1} - x_{n-1}) + v_n)^2] \\
&= E[(a\varepsilon_{n-1} + v_n)^2] \\
&= E[a^2\varepsilon_{n-1}^2] + \underbrace{2E[av_n\varepsilon_{n-1}]}_{0} + E[v_n^2]
\end{aligned}
$$

Note that the middle term equals zero because $v_n$ and $\varepsilon_{n-1}$ have zero correlation. $v_n$ is error that occurs between $n-1$ and $n$, whereas $\varepsilon_{n-1}$ is the accumulation of error until time step $n-1$. Simplifying further gives

$$\begin{aligned}\hat{P}_n &= E[a^2\varepsilon_{n-1}^2] + E[v_n^2] \\ &= a^2 E[\varepsilon_{n-1}^2] + Q \\ &= a^2 P_{n-1} + Q\end{aligned}$$

This is the "error prediction" equation. We use $Q$ to denote the mean square of the error that accumulates between time steps $n-1$ and $n$, known as process noise. Finally, we can derive the "error update" equation, starting from Equation (6):

$$\begin{aligned}P_n &= E[\varepsilon_n^2] \\ &= E[(y_n - x_n)^2] \\ &= E[(y_n - (\hat{x}_n + k_n z_n - k_n \hat{x}_n)^2] \\ &= E[(y_n - (\hat{x}_n + k_n(y_n + w_n) - k_n \hat{x}_n)^2] \\ &= E[(y_n - \hat{x}_n)^2(1-k_n)^2 - \underbrace{2k_n w_n (y_n - \hat{x}_n)(1-k_n)}_{0} + k_n^2 w_n^2] \\ &= (1-k_n)^2 E[(y_n - \hat{x}_n)^2] + k_n^2 E[w_n^2] \\ &= (1-k_n)^2 \hat{P}_n + k_n^2 R\end{aligned}$$

Substituting in Equation (5) for $R$ gives

$$\begin{aligned}P_n &= (1-k_n)^2 \hat{P}_n + k_n^2 \frac{\hat{P}_n(1-k_n)}{k_n} \\ &= \hat{P}_n(1-k_n)\end{aligned}$$

We have now derived all five equations of the Kalman Filter. See Table 1 for a summary of the equations and Fig. 1 for a block diagram of the Kalman Filter.

Table 1. Summary of Equations for Kalman Filter

| Stage of Filter | Name | Equation |
|---|---|---|
| Predict | State Prediction | $\hat{x}_n = ax_{n-1} + b$ |
| | Error Prediction | $\hat{P}_n = a^2 P_{n-1} + Q$ |
| Update | Kalman Gain | $k_n = \frac{\hat{P}_n}{\hat{P}_n + R}$ |
| | State Update | $x_n = \hat{x}_n + k_n(z_n - \hat{x}_n)$ |
| | Error Update | $P_n = \hat{P}_n(1 - k_n)$ |

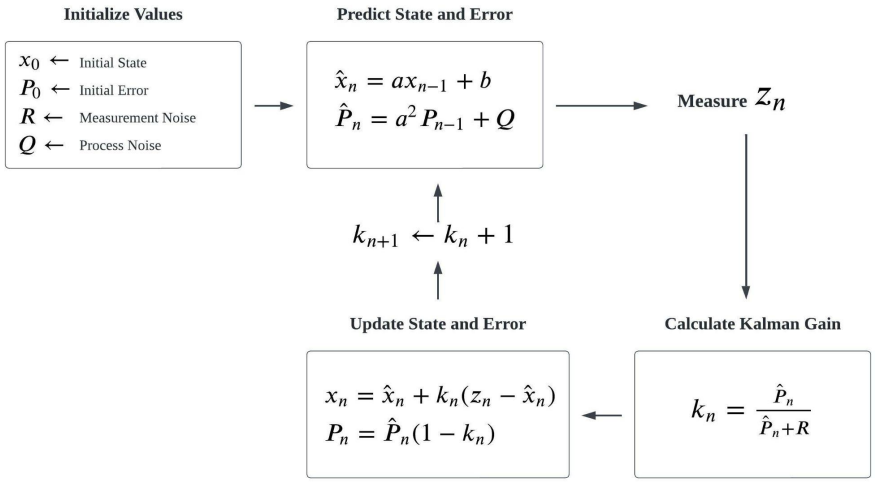


Fig 1. Kalman Filter Block Diagram

## Extended Kalman Filter (EKF)

At the beginning of this paper, we made the simplification to only consider linear transformations of the state from one time step to another. This was because only linear transformations/combinations of one or multiple normal random variable(s) result in another normal random variable (Taboga; Eisenberg & Sullivan). Since the state is effectively being treated as a normal random variable (with $x_n$ as the mean and $P_n$ as the variance), only linear transformations in the state prediction equation and a linear combination in the state update equation are possible in order to ensure the next iteration of the state in the Kalman Filter is also a normal random variable.

The EKF modifies the state update equation so that it approximates the difference between successive iterations of a nonlinear signal as a linear transformation (Speyer & Chung). Instead of linearly transforming the last time step's best estimate with a multiplicative constant *a* and additive constant *b*, the EKF takes the best estimate of the state from the previous time step and adds the change in the signal that *should* occur.

$$\hat{x}_n = x_{n-1} + \Delta f(n)$$

For example, for a sine wave signal, the state prediction would be:

$$\hat{x}_n = x_{n-1} + \sin(n) - \sin(n-1)$$

Table 2 summarizes the equations for the EKF.

Table 2. Summary of Equations for Extended Kalman Filter

| Stage of Filter | Name | Equation |
|---|---|---|
| Predict | State Prediction | $\hat{x}_n = x_{n-1} + \Delta f(n)$ |
| | Error Prediction | $\hat{P}_n = [f'(n-1)]^2 P_{n-1} + Q$ |
| Update | Kalman Gain | $k_n = \frac{\hat{P}_n}{\hat{P}_n + R}$ |
| | State Update | $x_n = \hat{x}_n + k_n(z_n - \hat{x}_n)$ |
| | Error Update | $P_n = \hat{P}_n(1 - k_n)$ |

## Extended Kalman Filter Example

Consider a function $f(n) = \sin(n)$, where $n$ is the current time step. Measurements of the function are noisy due to uncertainty in the sensor. The exact uncertainty is unknown and we can only approximate it (i.e. tuning $R$). We will apply the EKF to the noisy measurements of the function.

We initialize the filter at $n = 0$ with the following values for initial state prediction, initial error prediction, measurement noise and process noise, respectively:

$$\hat{x}_0 = 0$$

$$\hat{P}_0 = 1$$

$$R = 1$$

$$Q = 0.1$$

Then, we compute the Kalman Gain, record the noisy input, and update the state and error.

$$k_0 = \frac{1}{1+1} = 0.5$$

$$z_0 = -0.9192$$

$$x_0 = 0 + 0.5(-0.9192 - 0) = -0.4596$$

$$P_0 = 1(1 - 0.5) = 0.5$$

For $n = 0.1$, we predict the state and error using the equations in Table 2, compute the Kalman Gain, record a measurement and finally update the state and error. We repeat this process for many more iterations.

Table 3. First Four Iterations of the Extended Kalman Filter

| $n$ | State Prediction ($\hat{x}_n$) | Error Prediction ($\hat{P}_n$) | Kalman Gain ($k_n$) | Measurement ($z_n$) | State Update ($x_n$) | Error Update ($P_n$) |
|---|---|---|---|---|---|---|
| 0 | 0.0000 | 1.0000 | 0.5000 | -0.9192 | -0.4596 | 0.5000 |
| 0.1 | -0.3597 | 0.4516 | 0.3111 | 0.1423 | -0.2036 | 0.3111 |
| 0.2 | -0.1047 | 0.3484 | 0.2584 | 0.9741 | 0.1740 | 0.2584 |
| 0.3 | 0.2709 | 0.3006 | 0.2311 | -0.7405 | 0.0371 | 0.2311 |
| … | … | … | … | … | … | … |

The values in the "state update" column are the best estimates of the function $f(n) = \sin(n)$ for each $n$. The values may seem wildly incorrect, although this is because in the first few iterations the filter must adjust the initial state estimate. Fig. 2 graphs the first 301 iterations of the filter (from $n = 0$ to $n = 30$) with the initial values from above.

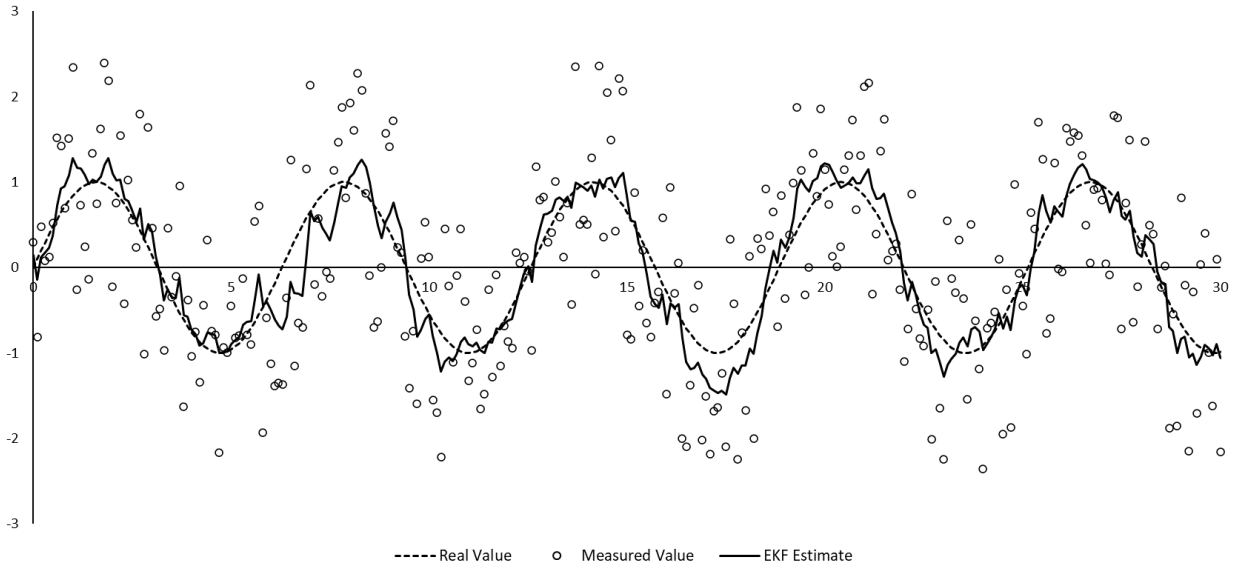


Fig 2. Comparison of Extended Kalman Filter Estimate with Real and Measured Values.

In Fig. 2, it's apparent that the estimated values of the function are not smooth and at many of the peaks and troughs the filter overestimates the state. Increasing Q only makes the estimates more noisy, as shown in Fig 3.

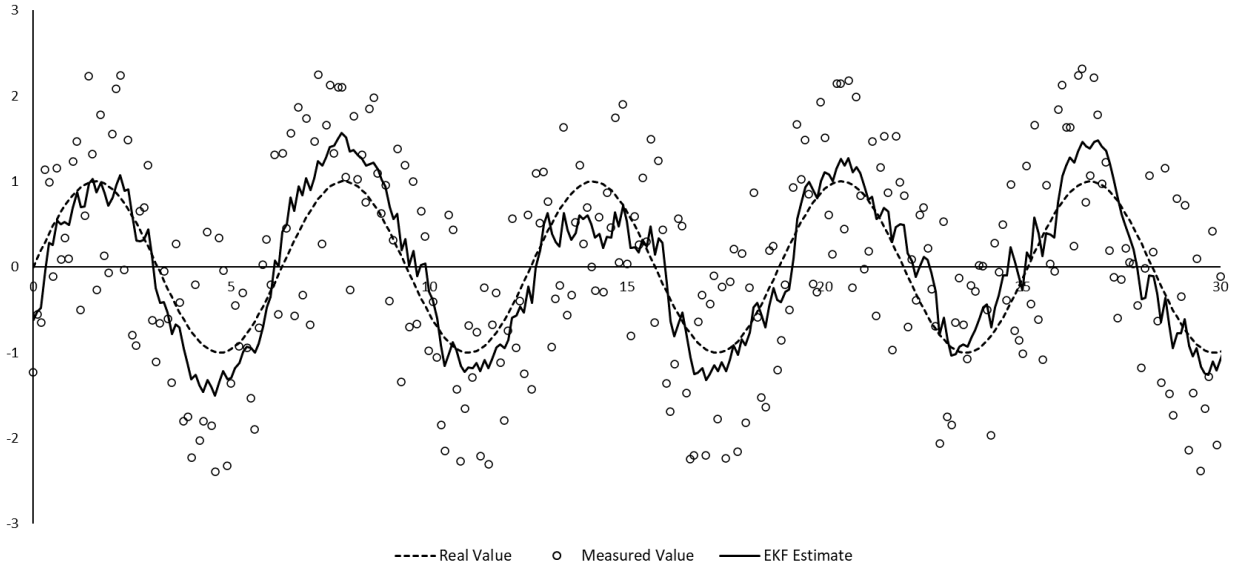


Fig 3. EKF Implementation for Q = 0.3.

Reducing Q smooths the estimates, but continues to over and under estimate at the peaks and troughs, as shown in Fig. 4.

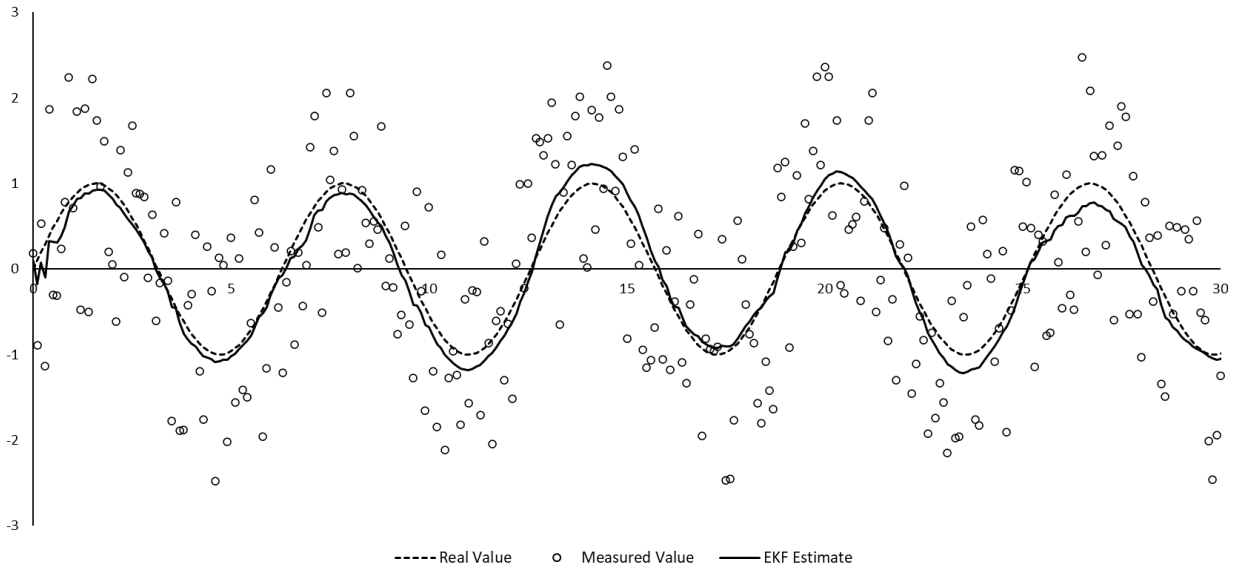


Fig 4. EKF Implementation for Q = 0.01.

Further reductions in Q continue to smooth the function, at the cost of accuracy, as shown in Fig. 5. Even when Q is 0, large accuracy issues are observed.

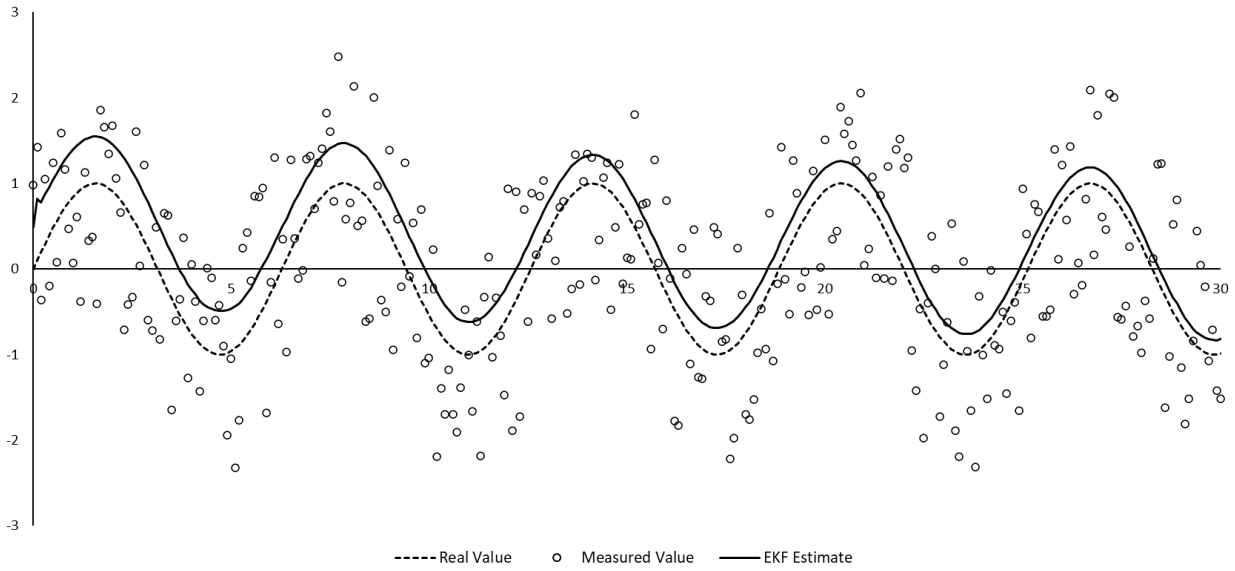


Fig 5. EKF Implementation for Q = 0.001.

The changes above are simply a reflection of how changing Q increases or decreases the filter's sensitivity to the measurements. Higher values of Q increase the Kalman Gain (as per the Error Prediction and Kalman Gain equations), thereby weighing measurements more than the predicted state in the state update equation. This makes sense because greater process noise (denoted as Q) decreases the previous state estimate's reliability and, therefore, the filter should rely more on new measurements. However, the by-product is that the noise in the measurements translates to noise in the state estimates. Lower values of Q do the opposite - decreasing the process noise and increasing the filter's reliance on the predicted state rather than measurements.

## Autocorrelation Function (ACF)

The EKF has significant issues when estimating a sinusoidal signal. As an alternative to the EKF, we will use the autocorrelation function (ACF) to estimate a signal (specifically sine waves) without the issues that occur when using the EKF, which include (a) incurring significant errors in accuracy due to overshoot and undershoot, (b) dismissing changes in the underlying signal as noise, (c) incorrect state estimation due to wrong initial estimates and/or incorrect definition of the model.

We begin by defining the cross-correlation of two time series (a necessity to later define autocorrelation). The cross-correlation of two time series is a number which describes how closely the two series behave. For discrete time, it is defined as the sum of the point-by-point multiplication of the two data sets (Derrick & Thomas).

$$r_{xy} = \sum_{i=0}^{N} x_i y_i \tag{7}$$

This is essentially the dot product of vectors $x$ and $y$. Similarly, the cross-correlation of two deterministic signals $x(t)$ and $y(t)$ in continuous time over a finite interval $[0, T]$ is given as

$$r_{xy} = \int_0^T x(t)y(t)dt$$

Like the discrete version in Equation (7), this equation is the sum of the multiplication of $x(t)$ and $y(t)$ at every point. Whereas the cross-correlation describes how similarly two signals behave, the ACF describes how similarly *one* signal behaves with a time-shifted version of itself (Haddad). In other words, the ACF outputs a number that is the cross-correlation between a signal $y(t)$ and itself shifted in time $y(t + \tau)$. The ACF of a continuous periodic deterministic signal $y(t)$ is given as follows (Haddad).

$$R_{yy}(\tau) = \frac{1}{T}\int_0^T y(t)y(t + \tau)dt \qquad t, \tau \geq 0 \tag{8}$$

We can now use this equation to derive the theoretical values of the ACF at any lag $\tau \geq 0$ for any real-valued sine wave of the form

$$y(t) = A\sin(wt) \tag{9}$$

where $A$ is the amplitude and $w$ is the angular frequency, such that

$$w = \frac{2\pi}{T} \tag{10}$$

where $T$ is the period of the sine wave defined in Equation (9). We can substitute the model for the sine wave into Equation (8) and apply a product-to-sum trigonometric identity.

$$
\begin{aligned}
R_{yy}(\tau) &= \frac{1}{T}\int_0^T A\sin(wt)A\sin(wt+w\tau)dt \\
&= \frac{A^2}{T}\int_0^T \left(\frac{\cos(w\tau)}{2} - \frac{\cos(2wt+w\tau)}{2}\right)dt \\
&= \frac{A^2\cos(w\tau)}{2T}t\Big|_0^T - \frac{A^2}{2T}\int_0^T \cos(2wt+w\tau)dt
\end{aligned}
$$

Then we evaluate the first term and solve the second term by substituting $u = 2wt + w\tau$.

$$
\begin{aligned}
R_{yy}(\tau) &= \frac{A^2}{2}\cos(w\tau) - \frac{A^2}{4Tw}\int_0^T \cos(u)du \\
&= \frac{A^2}{2}\cos(w\tau) - \frac{A^2}{4Tw}\left(\sin(2wt-w\tau)\Big|_0^T\right) \\
&= \frac{A^2}{2}\cos(w\tau) + \frac{A^2}{T}\left(\frac{\sin(w\tau)-\sin(2wT+w\tau)}{4w}\right)
\end{aligned}
\tag{11}
$$

However, this equation can be further simplified by simply ignoring the second term because it is so small. This is evident when we substitute in equation (10) for $T$.

$$
\begin{aligned}
\frac{A^2}{T}\left(\frac{\sin(w\tau)-\sin(w(2T+\tau))}{4w}\right) &= \frac{A^2w}{2\pi}\left(\frac{\sin(w\tau)-\sin(w(2(\frac{2\pi}{w})+\tau))}{4w}\right) \\
&= \frac{A^2}{8\pi}\left(\sin(w\tau)-\sin(\tau)\right)
\end{aligned}
$$

The resulting expression is clearly very small, even for large values of $A$. After eliminating the second term from Equation (11), the theoretical ACF of $y(t)$ for lag $\tau$ is found to be

$$
R_{yy}(\tau) = \tfrac{A^2}{2}\cos(w\tau) \tag{12}
$$

Equation (12) is the ACF for all sine waves of the form in Equation (9). One important observation to make that will be significant later on is that the autocorrelation of a sine wave is only a function of the parameters of the sine wave and lag (which we assume the user knows).

## Autocorrelation Function Example

The theoretical formula for the autocorrelation of a sinusoid with fixed amplitude and angular frequency has been derived. Fig. 8 shows the theoretical and measured autocorrelation of a sine wave with $A, w = 1$. The inputs for the measured values of the ACF are the noisy measurements of the sine wave.

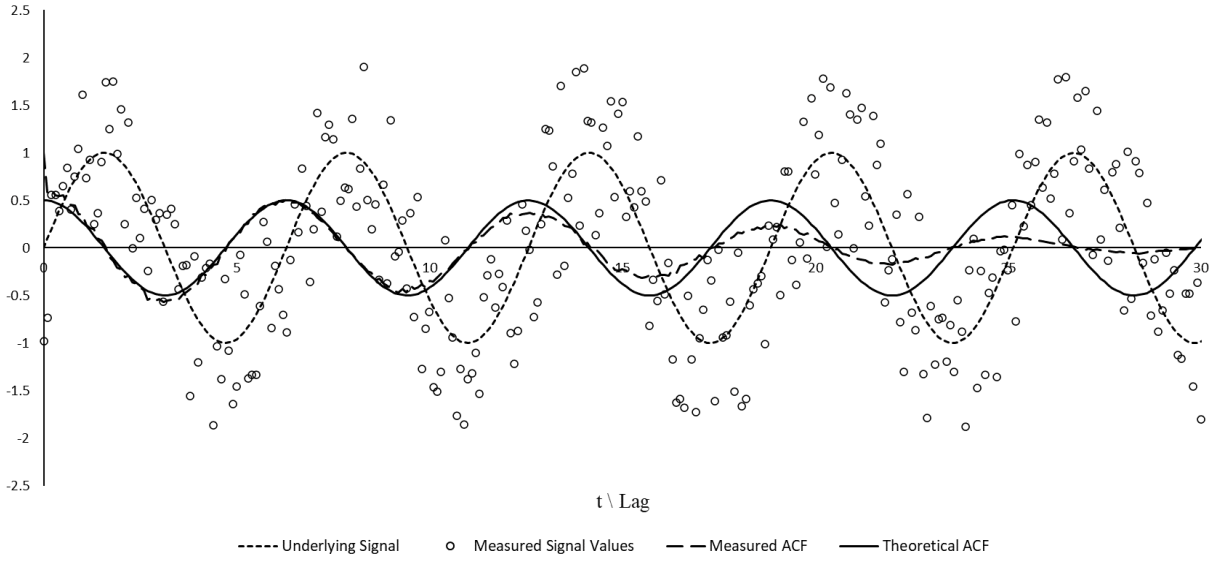


Fig 8. Theoretical and Measured ACF of a Sinusoidal Signal.

The measured autocorrelation in Fig. 8 very closely matches the theoretical ACF near the beginning. The measured ACF dampens over time because of the accumulated noise from the measurements of the sine wave.

Fig. 8 indicates a method of reconstructing a sinusoidal signal underlying a series of noisy measurements: we can estimate the parameters $A$ and $w$ of the measured ACF values during the interval when the measured ACF is highly consistent with the theoretical ACF. We can then reconstruct the underlying signal by simply "plugging in" the estimates of $A$ and $w$ for the amplitude and angular frequency of the underlying sine wave. Such a method is

advantageous compared to many numerical methods and filters because it is (a) more computationally efficient and (b) requires no initial estimates of the parameters of the sine wave, of the initial state or a model of the amplitude and angular frequency.

## Sinusoidal Parameter Estimation

As explained in the previous section, it is possible to estimate the parameters $A$ and $w$ of a sine wave with additive noise by equation measured values of the ACF with the theoretical ACF expression, Equation (12), and then solving for $A$ and $w$. Unfortunately, there are two unknowns in this one equation. It would be possible to use numerical methods to solve for $A$ and $w$. However, that would be very computationally costly, which, as explained before, is particularly important since our new method must run efficiently in real time.

Many other methods exist in the literature for angular frequency estimation of a sinusoid using its autocorrelation (Lui & So; Cao et. al; Xu et. al.). However, they almost all require computing large summations or matrix multiplications repeatedly. Instead, we will attempt to find a much simpler and computationally efficient nearly-closed form solution for angular frequency and amplitude estimation. The goal is to identify a simple formula which equates a measurable number from a signal to at most one sinusoid. The formula must only contain one unknown variable (angular frequency). Such an equation would allow us to meet the ultimate goal of creating an algorithm that estimates the state of a noisy sine wave in real time. We will now derive this equation.

Our derivation begins by taking the difference of the ACF at subsequent time steps and changing them into complex functions. We will take the difference of the ACF at subsequent time steps and then use the complex representation of sinusoids to combine them.

$$
\begin{aligned}
\delta_1 &= R_{yy}(\tau_2) - R_{yy}(\tau_1) \\
&= \frac{A^2}{2} cos(w\tau_2) - \frac{A^2}{2} cos(w\tau_1) \\
&= \frac{A^2}{2} (cos(w\tau_2) - cos(w\tau_1)) \\
&= \frac{A^2}{2} (Re(e^{iw\tau_2}) - Re(e^{iw\tau_1})) \\
&= \frac{A^2}{2} Re(e^{iw\tau_2} - e^{iw\tau_1})
\end{aligned} \tag{13}
$$

The real and imaginary parts of the two sinusoids are then grouped together.

$$
\begin{aligned}
e^{iw\tau_2} - e^{iw\tau_1} &= (\cos(w\tau_2) + i\sin(w\tau_2)) - (\cos(w\tau_1) + i\sin(w\tau_1)) \\
&= (\cos(w\tau_2) - \cos(w\tau_1)) + i(\sin(w\tau_2) - \sin(w\tau_1)) \\
&= re^{i\theta}
\end{aligned}
$$

new expressions for $\theta$ and $r$ must be derived. Using sum-to-product identities, $\theta$ is derived:

$$
\begin{aligned}
\theta &= arctan\left(\frac{\sin(w\tau_2) - \sin(w\tau_1)}{\cos(w\tau_2) - \cos(w\tau_1)}\right) \\
&= arctan\left(\frac{2\cos\left(\frac{w\tau_2 + w\tau_1}{2}\right)\sin\left(\frac{w\tau_2 - w\tau_1}{2}\right)}{-2\sin\left(\frac{w\tau_2 + w\tau_1}{2}\right)\sin\left(\frac{w\tau_2 - w\tau_1}{2}\right)}\right) \\
&= arctan\left(-\frac{\cos\left(\frac{w\tau_2 + w\tau_1}{2}\right)}{\sin\left(\frac{w\tau_2 + w\tau_1}{2}\right)}\right) \\
&= arctan\left(-cot\left(\frac{w\tau_2 + w\tau_1}{2}\right)\right) \\
&= arctan\left(tan\left(\frac{\pi}{2} + \frac{w\tau_2 + w\tau_1}{2}\right)\right) \\
&= \frac{\pi}{2} + \frac{w}{2}(\tau_2 + \tau_1)
\end{aligned}
$$

Next, the modulus $r$ is derived:

$$\begin{aligned} r^2 &= (\cos\omega\tau_2 - \cos\omega\tau_1)^2 + (\sin\omega\tau_2 - \sin\omega\tau_1)^2 \\ &= \cos^2 w\tau_2 - 2\cos w\tau_2 \cos w\tau_1 + \cos^2 w\tau_1 \\ &\quad + \sin^2 w\tau_2 - 2\sin w\tau_2 \sin w\tau_1 + \sin^2 w\tau_1 \\ &= (\cos^2 w\tau_2 + \sin^2 w\tau_2) + (\cos^2 w\tau_1 + \sin^2 w\tau_1) \\ &\quad - 2(\cos w\tau_2 \cos w\tau_1 + \sin w\tau_2 \sin w\tau_1) \end{aligned}$$

The first two groups of terms can be simplified using the Pythagorean identity and the final group of terms can be simplified using a compound angle identity. Therefore,

$$\begin{aligned} r &= \sqrt{2 - 2\cos(w\tau_2 - w\tau_1)} \\ &= \sqrt{2}\sqrt{1 - \cos(w\tau_2 - w\tau_1)} \end{aligned}$$

We obtain the final equation for $r$ by using the half-angle identity.

$$\begin{aligned} r &= \sqrt{2}\sqrt{2\sin^2\left(\frac{w(\tau_2 - \tau_1)}{2}\right)} \\ &= 2\sin\left(\frac{w(\tau_2 - \tau_1)}{2}\right) \end{aligned}$$

We can now substitute in the new values for $\theta$ and $r$ into Equation (13) to obtain the final equation for the difference of two subsequent values of the ACF of a sine wave.

$$\begin{aligned} \delta_1 &= R_{yy}(\tau_2) - R_{yy}(\tau_1) \\ &= \frac{A^2}{2} Re(e^{iw\tau_2} - e^{iw\tau_1}) \\ &= \frac{A^2}{2} Re\left(re^{i\theta}\right) \\ &= \frac{A^2}{2} Re\left(2\sin\left(\frac{w(\tau_2 - \tau_1)}{2}\right) e^{i(\frac{\pi}{2} + \frac{w}{2}(\tau_2 + \tau_1))}\right) \\ &= A^2 \sin\left(\frac{w(\tau_2 - \tau_1)}{2}\right) \cos\left(\frac{\pi}{2} + \frac{w}{2}(\tau_2 + \tau_1)\right) \end{aligned} \tag{14}$$

Assuming that measurements of the sinusoid are taken at equal intervals, then $\tau_2 - \tau_1$ equals $\tau_n - \tau_{n-1}$ for all $n$. We denote this value as

$$\Delta\tau = \tau_n - \tau_{n-1}$$

Since the measurement interval is constant, the difference of $\theta$ between time steps $n$ and $n-1$ is also constant and equals

$$\begin{aligned}\Delta\theta &= \left(\frac{\pi}{2} + \frac{w}{2}(\tau_n + \tau_{n-1})\right) - \left(\frac{\pi}{2} + \frac{w}{2}(\tau_{n-1} + \tau_{n-2})\right) \\ &= \frac{w}{2}(\tau_n - \tau_{n-2}) \\ &= \frac{w}{2}(2\Delta\tau) \\ &= w\Delta\tau\end{aligned}$$

For the final step of the derivation, we define the following measurable value:

$$\Lambda = \frac{\delta_3 - \delta_1}{\delta_2 - \delta_1}$$

Using four consecutive values of the measured ACF of a sine wave we can obtain $\Lambda$. Substituting in Equation (14), we get:

$$\begin{aligned}\Lambda &= \frac{A^2 \sin\left(\frac{w\Delta\tau}{2}\right)(\cos(\theta + 2w\Delta\tau) - \cos(\theta))}{A^2 \sin\left(\frac{w\Delta\tau}{2}\right)(\cos(\theta + w\Delta\tau) - \cos(\theta))} \\ &= \frac{\cos(\theta + 2w\Delta\tau) - \cos(\theta)}{\cos(\theta + w\Delta\tau) - \cos(\theta)} \\ &= \frac{\cos(\theta)\cos(2w\Delta\tau) - \sin(\theta)\sin(2w\Delta\tau) - \cos(\theta)}{\cos(\theta)\cos(w\Delta\tau) - \sin(\theta)\sin(w\Delta\tau) - \cos(\theta)}\end{aligned} \tag{15}$$

We will assume that the interval between measurements ($\Delta\tau$) is very small. For example, it is reasonable to expect measurements would be made every 0.1 or 0.05 seconds.

Therefore, we can apply the small angle approximation for the cosines in the first terms of the numerator and denominator of Equation (15). This gives

$$\begin{aligned}\Lambda &\approx \frac{\cos(\theta) - \sin(\theta)\sin(2w\Delta\tau) - \cos(\theta)}{\cos(\theta) - \sin(\theta)\sin(w\Delta\tau) - \cos(\theta)} \\ &\approx \frac{\sin(2w\Delta\tau)}{\sin(w\Delta\tau)}\end{aligned}$$

By applying a double-angle identity, this equation can be reduced to one cosine:

$$\begin{aligned}\Lambda &\approx \frac{2\sin(w\Delta\tau)\cos(w\Delta\tau)}{\sin(w\Delta\tau)} \\ &\approx 2\cos(w\Delta\tau)\end{aligned} \tag{16}$$

An equation has now been derived that satisfies the initial objectives set. Equation (16) only has one unknown variable - the angular frequency $w$ - making it easy to solve for. No large/expensive calculations are necessary to obtain $\Lambda$ as it is just the difference and then quotient of four subsequent measured values of the ACF of the original sine wave. Once we estimate the angular frequency, we can also substitute it into Equation (12) to solve for the amplitude. There is, however, an issue with this formula: it is numerically "fragile." Due to the noise carried over from the original sine wave, solving for omega $\omega$ using Equation (16) can result in wildly different answers (or possibly no answer if $\Lambda > 2$). Furthermore, the accuracy of the estimate of $\omega$ reduces when $\omega$ is large, due to the small angle identity used. Fig. 9 shows the values of $\Lambda$ for an example noisy sine wave with $A, \omega = 1$.

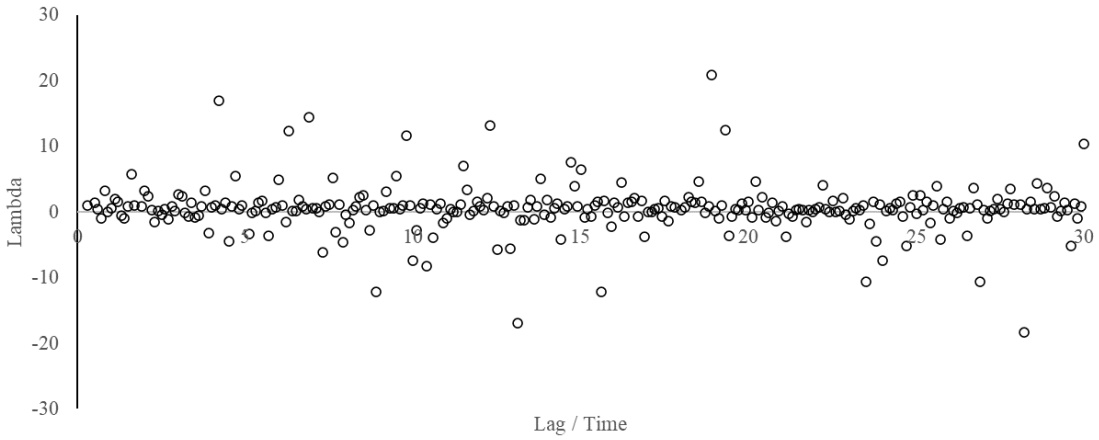


Fig 9. $\Lambda$ measured from the ACF of a Sine Wave

In addition it is evident from Fig. 9 that the values of $\Lambda$ vary greatly. In the next section, we will create the ACF estimation algorithm using Equation (16).

## Autocorrelation Function Filter (ACFF)

This new algorithm, which we will call the Autocorrelation Function Filter (ACFF), aims to estimate the real (underlying) value of a sine wave in real-time with greater accuracy and flexibility than the Extended Kalman Filter. The algorithm uses equation (16) derived in the previous section.

The first step in the algorithm is to accept the measurements of the sine wave as input. This measurement includes noise. The measurements are stored in an array. Second, the autocorrelation of all the measurements up to the current step is calculated:

$$R_{yy}(t) = \frac{1}{N}\sum_{\tau=0}^{N} y(t)y(t+\tau)$$

Where $y(t)$ is the first measurement in the array, $y(t+\tau)$ is the $\tau$th measurement in the array and $N$ is the number of measurements in the array. This calculation is repeated for all measurements $y(t)$ in the array from $t=1$ to $t=N$. We now have an array of $R_{yy}(t)$ for $t=1$ to $t=N$. Third, $\Lambda$ is computed for all $t$ using

$$\begin{aligned}\Lambda(t) &= \frac{\delta_3 - \delta_1}{\delta_2 - \delta_1} \\ &= \frac{(R_{yy}(t+3) - R_{yy}(t+2)) - (R_{yy}(t+1) - R_{yy}(t))}{(R_{yy}(t+2) - R_{yy}(t+1)) - (R_{yy}(t+1) - R_{yy}(t))}\end{aligned}$$

Fourth, the mean, $\Lambda_{av}$, of all $\Lambda(t)$ is taken. Fifth, we rearrange Equation (16) to estimate $w$:

$$w = \frac{1}{\Delta\tau} \arccos \frac{\Lambda_{av}}{2}$$

Finally, an estimate of the amplitude $A$ is calculated from the theoretical autocorrelation of a sine wave derived earlier:

$$R_{yy}(t) = \frac{A^2}{2} \cos(w\tau)$$

An estimate of the amplitude is found by substituting in the estimate for $w$ and each measured $R_{yy}(t)$. We now have the estimate for both the angular frequency and amplitude of a sine wave from noisy measurements.

## Comparison of the Algorithms

We now have two algorithms that can be used to estimate a "true" sinusoidal signal from measurements of the signal that include unwanted noise: the Extended Kalman Filter (EKF) and the Autocorrelation Function Filter (ACFF). To compare the accuracy of the filters, they were implemented and simulated in Python. The code is in Appendix A.

A sine wave with an amplitude of 1 and angular frequency of 1 was simulated with varying amounts of noise in their measurements. The Root-Mean-Square-Error between the "true" signal and the signal estimated using the ACFF and EKF was computed for different quantities of noise in the simulated signal measurements. The standard deviation of the noise was varied from 0 to 1 in steps of 0.05.

Since the EKF requires a model of the sine wave (estimates of the amplitude and angular frequency) to update from one time step to another, the EKF was tested twice, each time with a different model. The first time the EKF assumed both the amplitude and angular frequency were 1 (which is the true underlying signal), while the second time they were assumed to be 100. The results of the simulation are shown in Fig. 10.

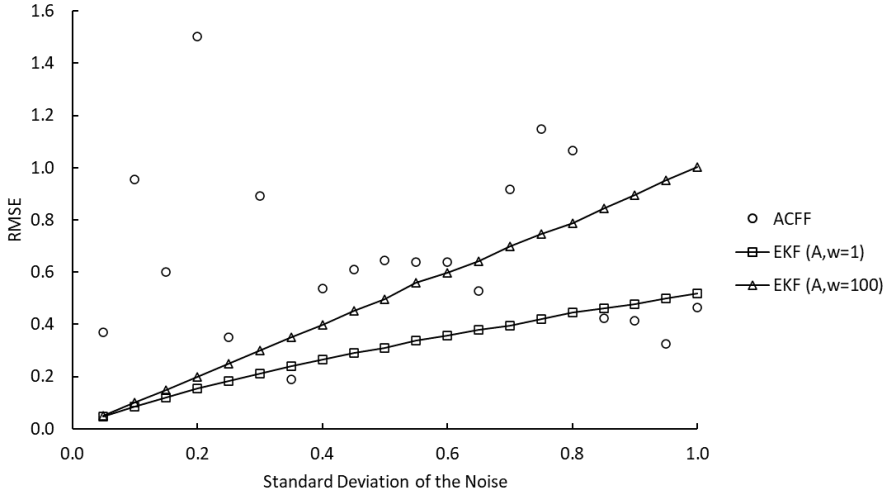


Fig 10. RMSE of varying filters' estimates of a noisy signal under varying amounts of noise.

The RMSE increases nearly linearly for both simulations of the EKF. The error for both start near 0, but the EKF with a completely incorrect model $(A, w = 100)$ increases much faster for greater noise, eventually reaching an RMSE of approximately 1 for a standard deviation of the noise of 1. Meanwhile, the RMSE for the ACFF does not follow any pattern. It varies greatly for all quantities of noise, but stays below an upper limit of ~1.5. The miscellany in RMSE for the ACFF is likely due to the filter's numerical fragility discussed earlier, as well as the small angle approximation used to create the filter.

For low standard deviations of the noise, it is clearly more beneficial to use the EKF. One can have high confidence that the filter will approximate the underlying "true" signal with little error. However, if the noise's standard deviation is greater than about 0.8 and the model for the EKF is either wildly off or completely unknown, it may be better to use the ACFF. While one would not be certain of the accuracy of the ACFF, with an extremely inaccurate model and high noise, it could be more accurate than the EKF.

The ACFF also has other qualitative benefits compared to the EKF:

1) The ACFF does not require any initial estimates, therefore reducing reliance on the expertise of the user;
2) The ACFF updates the angular frequency and amplitude of the sine wave every iteration, whereas the model in the EKF cannot be changed. Therefore, the ACFF could adapt to changes in the underlying sine wave that the EKF cannot;
3) The EKF combines two pieces of information, the system's model and measurements, whereas the sinusoidal estimation algorithm only utilizes measurements of the wave. This

takes away any influence that incorrect thinking about the signal has on estimation of the "true" signal.

## Conclusion

Existing mathematical filters were first derived: the Kalman Filter (KF) and Extended Kalman Filter (EKF). The limitations of the KF and EKF were shown, thereby providing motivation for the creation of a new filter - one that is responsive to changes in the underlying "true" signal and does not require any initial estimates or models from the user. A specific type of signal, the sine wave, was chosen to focus this investigation. Using the autocorrelation function, an equation was derived to estimate the underlying "true" angular frequency of a sine wave. This equation was then successfully used to create an algorithm for a mathematical filter, which we call the Autocorrelation Function Filter (ACFF), to compute the underlying "true" signal of a sine wave with additive zero-mean noise. The accuracies of the ACFF and EKF were tested for a sine wave with varying quantities of noise in its measurements. While the EKF is generally more reliable and accurate, for signals where there is very high noise and the model chosen is completely incorrect, it may be better to use the ACFF.

# Appendix A: Python Simulations

## Extended Kalman Filter Simulation

```
from math import sin, acos, sqrt, cos, floor, pi
from random import gauss
import numpy as np
from pandas import DataFrame

def rmse(y_true, y_pred):
    rmse = sqrt(np.square(np.subtract(y_true,y_pred)).mean())
    return rmse

def test(iters=10000, noise_variance_=[0], amplitude_=[1], omega_=[1], periods=1,
process_noise=0.5, model_amplitude=1, model_omega=1):
    end = np.pi*2*periods
    step = end/iters

    state_prediction = 0
    error_prediction = 1

    excel_df = DataFrame(index=noise_variance_)

    for noise_variance in noise_variance_:
        measurement_noise = noise_variance
        for amplitude in amplitude_:
            for omega in omega_:
                time = [0]
                signal = []
                signal_estimate = []
                true_signal = [0]

                end = floor(periods*((2*pi)/omega)/step)

                #First iteration
                kalman_gain = error_prediction/(error_prediction +
measurement_noise)
                measurement = np.random.normal(0, noise_variance, 1)
                signal.append(measurement)
                state_update = state_prediction + kalman_gain*(measurement -
state_prediction)
                signal_estimate.append(state_update)
                error_update = error_prediction*(1 - kalman_gain)

                #All iterations after first iteration
                for t in [x*step for x in range(1, end)]:
                    true_value = amplitude*sin(omega*t)
                    measurement = true_value + gauss(0, noise_variance)

                    true_signal.append(true_value)
                    time.append(t)
                    signal.append(measurement)

                    state_prediction = state_update +
model_amplitude*sin(model_omega*t) - model_amplitude*sin(model_omega*(t-step))
                    error_prediction =
((model_amplitude*model_omega*cos(model_omega*(t-step)))**2)*error_update +
process_noise
                    kalman_gain = error_prediction/(error_prediction +
measurement_noise)
                    state_update = state_prediction + kalman_gain*(measurement -
```

```
state_prediction)
                    error_update = error_prediction*(1 - kalman_gain)

                    signal_estimate.append(state_update)

                error = rmse(true_signal, signal_estimate)
                print("Step: %s | Noise Var: %s | Amplitude: %s | Omega: %s | RMSE:
%s" % (step, noise_variance, amplitude, omega, error))
                excel_df.at[noise_variance, omega] = error

    excel_df.to_excel(results.xlsx', sheet_name="sheet1")
```

## Autocorrelation Function Filter Simulation

```
from math import sin, acos, sqrt, cos, floor, pi
import matplotlib.pyplot as plt
from random import gauss
import pandas as pd
import numpy as np
from numpy import roll
from scipy.signal import correlate

def test(iters=10000, noise_variance_=[0], amplitude_=[1], omega_=[1], periods=1):

    excel_df = pd.DataFrame(index=noise_variance_)

    for noise_variance in noise_variance_:
        for amplitude in amplitude_:
            for omega in omega_:

                #Simulating the signal according to the chosen parameters
                end = np.pi*2*periods
                step = end/iters
                t = np.linspace(0, end, iters)
                realSignal = amplitude*np.sin(omega*t)
                noisySignal = amplitude*np.sin(omega*t) + np.random.normal(0,
noise_variance, iters)

                #Measuring the cyclic autocorrelation
                measured_autocorr = []
                for k in range(0, len(t)):
                    measured_autocorr.append(correlate(noisySignal,
roll(noisySignal, k), 'valid'))
                autocorr = np.array(measured_autocorr)/len(t)

                #Measuring big lambda
                lam = []
                for i in range(4, len(t)):
                    lam.append(((autocorr[i] - autocorr[i-1] - autocorr[i-2] +
autocorr[i-3])/(autocorr[i-1] - 2*autocorr[i-2] + autocorr[i-3]))/2)
                min_pres = min(lam)
                max_pres = max(lam)

                #Estimating omega
                w_options = [(x - min_pres)/(max_pres - min_pres) for x in lam]
                estimate_w = np.mean([acos(j) for j in w_options])

                #Estimating the amplitude
                A_options = 0
```

```
                for j in range(0, len(autocorr)):
                    A_options += sqrt(abs((2*autocorr[j]) /
(cos(estimate_w*step*j))))
                estimate_a = A_options/len(autocorr)

                #Calculating the RMSE
                calcSignal = estimate_a*np.sin(estimate_w*t)
                rmse = sqrt(np.square(np.subtract(realSignal,calcSignal)).mean())

                print("Step: %s | Noise Var: %s | Amplitude: %s | Omega: %s | RMSE:
%s" % (step, noise_variance, amplitude, omega, rmse))

                excel_df.at[noise_variance, omega] = rmse

    excel_df.to_excel('results.xlsx', sheet_name="sheet1")
```

Works Cited

Bozic, S. *Digital and Kalman filtering : an introduction to discrete-time filtering and optimum linear estimation*. Halsted Press, 1994.

Cao, Yan, et. al. "A closed-form expanded autocorrelation method for frequency estimation of a sinusoid." Signal Processing, vol. 92, no. 4, pp. 885-892, 2012. 10.1016/j.sigpro.2011.09.025. Date of Access 4 Aug. 2022.

Cheever, Erik. "The Scalar Kalman Filter." Swarthmore College. https://cheever.domains.swarthmore.edu/Ref/Kalman/ScalarKalman.html#References. Date of Access 6 Aug. 2022.

Derrick, Timothy, Thomas, Joshua. "Time Series Analysis: The Cross-Correlation Function." Iowa State University, 2004. 20.500.12876/52528. Date of Access 2 Aug. 2022.

Eisenberg, Bennett, Sullivan, Rosemary. "Why Is the Sum of Independent Normal Random Variables Normal?" The College Mathematics Journal, *Taylor & Francis*, vol. 8, no. 2, pp. 137, 2017. https://www.tandfonline.com/doi/citedby/10.1080/0025570X.2008.11953577. Date of Access 10 Jul. 2022.

Elliott, Douglas. "Handbook of Digital Signal Processing." Academic Press, *Elsevier*, 28 Dec. 1987. https://www.elsevier.com/books/handbook-of-digital-signal-processing/elliott/978-0-08-050780-4. Date of Access 7 Aug. 2022.

Grewal, Mohinder, Andrews, Angus. "Applications of Kalman Filtering in Aerospace 1960 to the Present [Historical Perspectives]." IEEE Control Systems Magazine, vol. 30, no. 3, pp. 69-78, June 2010. 10.1109/MCS.2010.936465. Date of Access 8 Aug. 2022.

Haddad, Abraham. *Probability Systems and Random Signals*. Pearson, 2005.

Kim, Youngjoo, Bang, Hyochoong. “Introduction to Kalman Filter and Its Applications.” Introduction and Implementations of the Kalman Filter, edited by Felix Govaers, IntechOpen, 2018. 10.5772/intechopen.80600. Date of Access 28 Jul. 2022.

Lui, Kenneth, So, H. “Two-stage autocorrelation approach for accurate single sinusoidal frequency estimation.” Signal Processing, vol. 88, no. 7, pp. 1852-1857, 2008. 10.1016/j.sigpro.2008.01.023. Date of Access 15 Aug. 2022.

Manolakis, D. *Statistical and Adaptive Signal Processing*. McGraw Hill, 2000.

Masnadi-Shirazi, Hamed, et. al. “A Step by Step Mathematical Derivation and Tutorial on Kalman Filters.” Cornell University, *arXiv*, 8 Oct. 2019. arXiv:1910.03558. Date of Access 4 Jun. 2022.

Mosammam, Ali. “Kalman Filter: A Simple Derivation.” Horizon Research Publishing Corporation, vol. 3, no. 2, pp. 41-45, 2015. 10.13189/ms.2015.030203. Date of Access 15 Jul. 2022.

Ribeiro, Maria Isabel. “Kalman and Extended Kalman Filters: Concept, Derivation and Properties.” Instituto Superior Tecnico, Feb. 2004. http://users.isr.ist.utl.pt/~mir/pub/kalman.pdf. Date of Access 25 Jul. 2022.

Siouris, George. “An Engineering Approach to Optimal Control and Estimation Theory.” John Wiley & Sons, Feb. 1996.

Speyer, Jason, Chung, Walter. “Stochastic Processes, Estimation and Control.” Society for Industrial and Applied Mathematics, 2008. 10.1137/1.9780898718591. Date of Access 17 Aug. 2022.

Taboga, Marco. “Linear combinations of normal random variables.” Lectures on probability

theory and mathematical statistics, *Kindle Direct Publishing*, 2021. http://www.statlect.com/probability-distributions/normal-distribution-linear-combinations. Date of Access 1 Aug. 2022.

Xu, Dingjie, et. al.. "A two-stage autocorrelation method for frequency estimation of sinusoidal signal." International Journal of Electronics, vol. 101, no. 2, pp. 194-203, 2014. 10.1080/00207217.2013.775635. Date of Access 26 Jul. 2022.