Linear Phase Estimation

Daniel Paz

Wheel Angle Estimation

We aim to estimate the phase of the wheel at each given time.

We sample uniformly in the period of 1 second.

We have 5% chance of uniformly distributed spikes in the phase, regardless of the gaussian noise added to the phase samples.

$$t_i \sim U[0,1] \qquad \forall i \in [1 \ldots N]$$

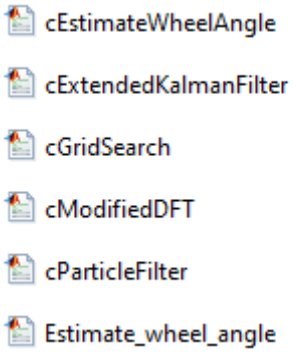
$$y_i = \begin{cases} U[0,1] & w.p. \quad p_{uniform} \\ \mathrm{mod}(\omega t_i + \theta_0 + n_i, 1) & w.p. \quad 1 - p_{uniform} \end{cases}$$

$$p_{uniform} = 0.05$$

$$n \sim N(0, \sigma^2), \ \sigma = 0.03$$

The parameters we aim to estimate are ω, θ_0

To run the code, you need to have all the following files in your path:



The main function that runs the code is:

Estimate_wheel_angle()

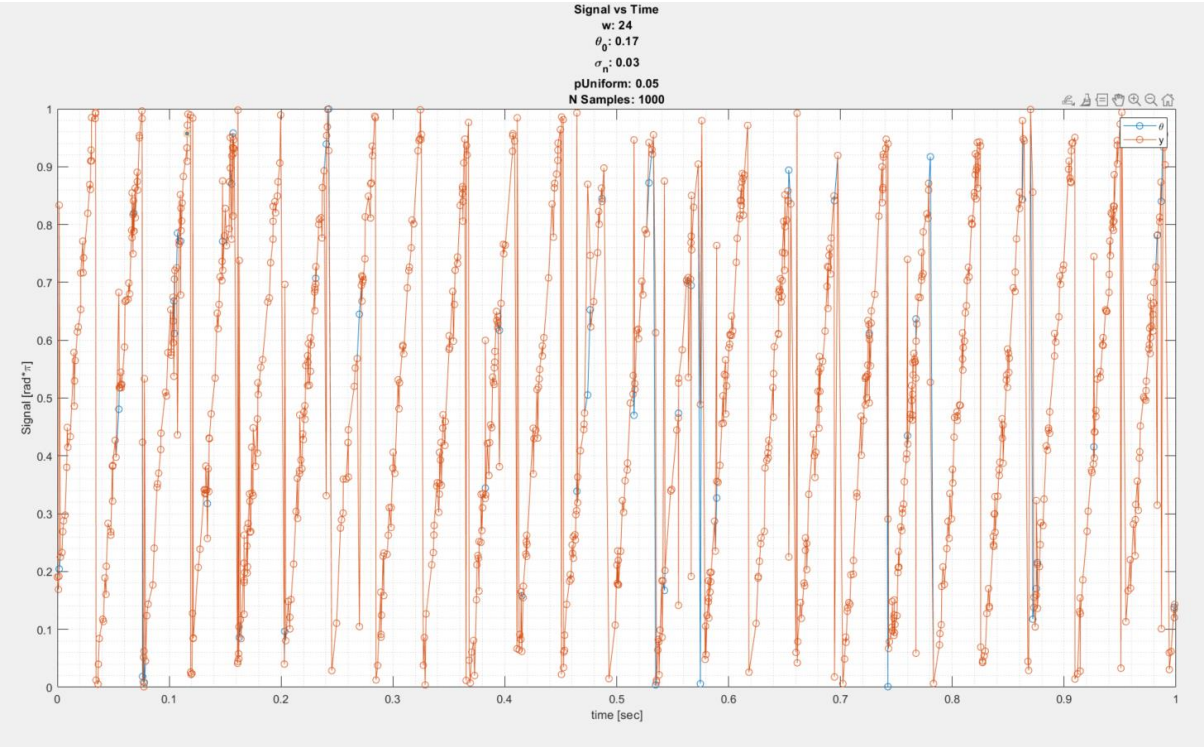
And these are the control parameters:

```
%% Inputs
% methodList = "PF";
methodList = ["MDFT","EKF","PF","GS"];
noiseMode  = "On";
N          = 1000;
w          = 24;
theta0     = 0.17;
bShowPlots = true;
Nstats     = 1000;
seed       = 1111;
rng(seed);
bSaveResults = true;
```

noiseMode gets 3 options:

- “on”: the signal is simulated with Gaussian noise and the uniformly random phase with probability 5% (aka. Spikes)
- “Off”: no Gaussian noise and no spikes
- “Only”: only Gaussian noise and no spikes

First, we built a function that shows the signal in the given period: $t \in [0,1]$



For each one of the algorithms, we needed to provide an initial guess for ω, θ_0.

However, θ_0 is bound to $U[0,1]$ so we would concentrate on the initial guess ω.

We choose to sample the signal with:

$$N = 1000[samples]$$

Initial guess for the angular velocity

The time vector is not deterministic as usual, so we need to save in the memory the time samples in: $\underline{t}$

The angular velocity is the derivative in time of the phase so we will derivate y:

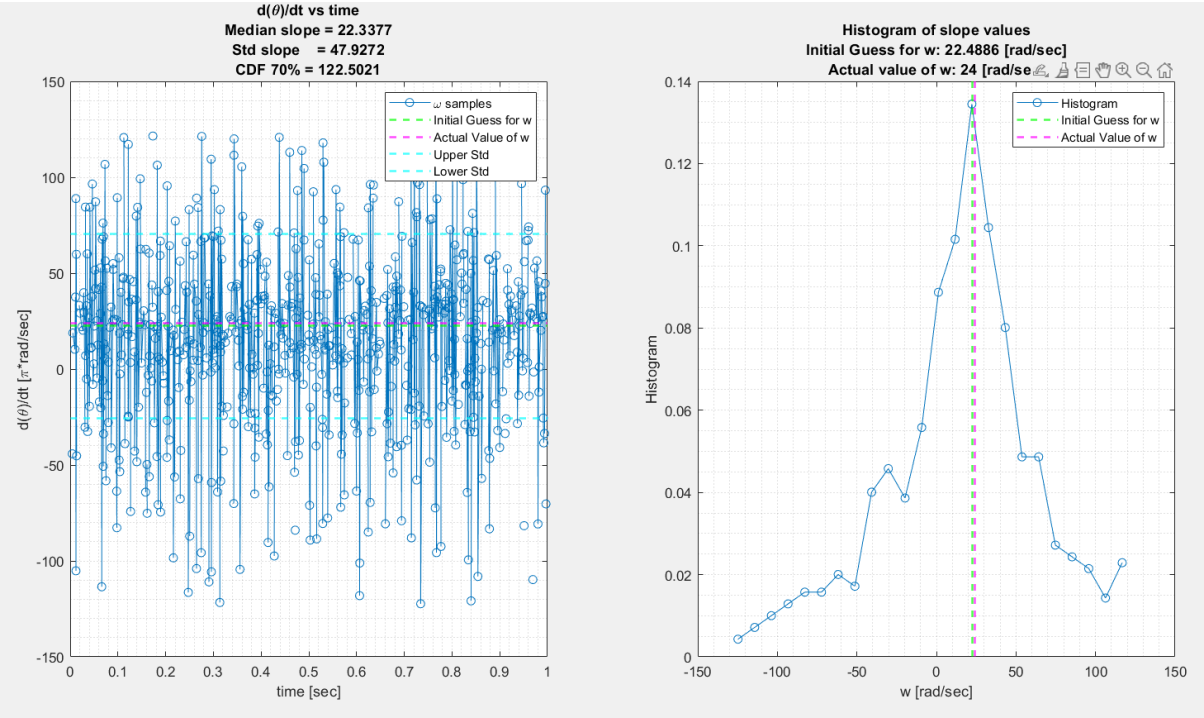
$$\hat{\omega}_i^{Init} = \frac{dy_i}{dt_i} = \frac{y_i - y_{i-1}}{t_i - t_{i-1}}$$

Due to the 5% outliers we need to clean $\hat{\omega}_i^{Init}$ for more reliable initial guess.

We do that by taking the 70% CDF percentage of $\left|\hat{\omega}_i^{Init}\right|$ and ignore all the indices the are higher than this value.

Then, we generate the histogram of the remaining $\hat{\omega}_i^{Init}$ and take the $\hat{\omega}_{i,\max}^{Init}$ that corresponds to the maximal bin of the histogram (the most likely value).

To get a better accuracy for the initial guess we use weighted least squares parabolic interpolation (see MDFT section).



For θ_0 initial value we choose the first received value:

$$\theta_0^{Init} = y(1)$$

This choice can be more prone to noise and phase spikes, so we will trust the initial value of ω more than that of θ_0

We have suggested 4 algorithms:

- Modified DFT (MDFT)
- Extended Kalman Filter (EKF)
- Particle Filter (PF)
- Grid Search (GS)

We define the objective as the RMSE:

$$RMSE_{\omega} = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}\left|\hat{\omega}_i - \omega\right|^2}$$

$$RMSE_{\theta_0} = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1}\left|\hat{\theta}_{i,0} - \theta_0\right|^2}$$

Modified DFT

ω Estimation

First, we transform the phase samples y into phasors z:

$$y[i]=\begin{cases} v_i & w.p. \quad p_u \\ \mathrm{mod}(\omega t_i+\theta_0+n_i,1) & w.p. \ (1-p_u) \end{cases}$$

$$z[i]=e^{j2\pi y[i]}=\begin{cases} e^{j2\pi v_i} & w.p. \quad p_u \\ e^{j2\pi \,\mathrm{mod}(\omega t_i+\theta_0+n_i,1)} & w.p. \ (1-p_u) \end{cases}=\begin{cases} e^{j2\pi v_i} & w.p. \quad p_u \\ e^{j2\pi(\omega t_i+\theta_0+n_i)} & w.p. \ (1-p_u) \end{cases}=$$

$$\begin{cases} z_v[i] & \\ e^{j2\pi v_i} & w.p. \quad p_u \\ \overbrace{e^{j2\pi\omega t_i}\cdot e^{j2\pi(\theta_0+n_i)}}^{z_s[i]} & w.p. \ (1-p_u) \end{cases}, \quad p_u=0.05$$

We can see that we can divide the estimation of ω,θ_0 to ω first due to the fact that the θ_0 term is reduced to 1 when we take the power of the modified DFT of z.

The modified DFT of z is taking the twiddle vectors of the regular DFT and instead of using time constant spaced phasor vectors, we use the randomly generated time sampled vector: t_i

Note:

We do not have a regular DFT with a Dirichle kernel, hence we do not have a sampling frequency or a periodicity in the ω domain (or ambiguity).

That means that we can detect any angular velocity without any ambiguity.

We define, ourselves, the ω resolution, $\Delta\omega$, by choosing the twiddles the following way:

$$\boxed{Z[k]}=\sum_{i=0}^{N-1}z[i]=\sum_{i=0}^{p_u\cdot N-1}e^{j2\pi v_i}\cdot e^{-j2\pi\cdot t_i\Delta\omega\cdot k}+\sum_{i=0}^{(1-p_u)\cdot N-1}e^{j2\pi\omega t_i}\cdot e^{j2\pi(\theta_0+n_i)}\cdot e^{-j2\pi\cdot t_i\Delta\omega\cdot k}=$$

$$\overbrace{\sum_{i=0}^{(1-p_u)\cdot N-1}\overbrace{e^{-j2\pi(k\cdot\Delta\omega t_i-v_i)}}^{Z_v^{(i)}[k]}}^{Z_v[k]}+ \qquad \%\,\text{Uniform distributed phase noise}$$

$$e^{j2\pi(\theta_0)}\cdot\overbrace{\sum_{i=0}^{(1-p_u)\cdot N-1}e^{-j2\pi\left((k\cdot\Delta\omega-\omega)t_i-n_i\right)}}^{Z_s[k]} \qquad \%\,\text{Signal+Gaussian noise}$$

$$EZ_v^{(i)}[k]=\int_0^1 dv_i\overbrace{f_v(v_i)}^{1_{v\in[0,1]}}e^{-j2\pi(k\cdot\Delta\omega t_i-v_i)}=e^{-j2\pi k\cdot\Delta\omega t_i}\int_0^1 dv_i e^{j2\pi v_i}=$$

$$e^{-j2\pi k\cdot\Delta\omega t_i}\left[\frac{e^{j2\pi v_i}}{j2\pi}\right]_0^1=e^{-j2\pi k\cdot\Delta\omega t_i}\left[\frac{\overbrace{e^{j2\pi}}^{1}-1}{j2\pi}\right]_0^1=0$$

We can see that the uniform phase noise's expectation is 0, therefore when summed coherently its limit is 0.

Moreover, the number of elements in $Z_v[k]$ is $p_u = 0.05$ times the number of elements in $Z_s[k]$, therefore we can neglect the affect of it on the output of the MDFT.

Let's now focus on the additive Gaussian noise to the phase element of $Z_s^{(i)}[k]$: $n_i \sim N(0, \sigma_n^2) = N(0, 0.03^2)$. The std of this noise is much smaller than 1: $\sigma_n \ll 1$, therefore is will not affect much the phase of the max peak of the MDFT which is coherently summed over: $(1-p_u)N$ samples.

$$MDFT = \underline{\underline{F}}_{K\times N} = \begin{bmatrix} e^{-j2\pi\Delta\omega t_0\cdot 0} & e^{-j2\pi\Delta\omega t_1\cdot 0} & \cdots & e^{-j2\pi\Delta\omega t_{N-1}\cdot 0} \\ e^{-j2\pi\Delta\omega t_0\cdot 1} & e^{-j2\pi\Delta\omega t_1\cdot 1} & \cdots & e^{-j2\pi\Delta\omega t_{N-1}\cdot 1} \\ \vdots & \ddots & \ddots & \vdots \\ e^{-j2\pi\Delta\omega t_0\cdot (K-1)} & e^{-j2\pi\Delta\omega t_1\cdot (K-1)} & \cdots & e^{-j2\pi\Delta\omega t_{N-1}\cdot (K-1)} \end{bmatrix}_{K\times N}$$

So, the power of each sample after applying MDFT on the phasor signal:

$$|Z[k]|^2 = |Z_v[k] + e^{j2\pi(\theta_0)}\cdot Z_s[k]|^2 \approx |e^{j2\pi(\theta_0)}\cdot Z_s[k]|^2 = \overbrace{|e^{j2\pi(\theta_0)}|^2}^{1}\cdot |Z_s[k]|^2 = |Z_s[k]|^2$$

$$\Rightarrow \boxed{|Z[k]|^2 \approx |Z_s[k]|^2}$$

Now to reduce computation complexity we focus on the frequencies around the initial guess of ω:

$$\underline{\tilde{Z}} = \underline{Z}_{\substack{k\in[k_{start},k_{end}] \\ k\notin k^{Prev}-B}}$$

To optimize computation complexity, we only scan the ω samples we have not scanned yet, but we take overlap of $B = 2$ samples so the peak in the worst case would not be on the edge.

Where δ_ω is the search area around the initial guess ω^{Init}

We initialize:

$$\delta_\omega = \delta_\omega^{init} = 20\frac{[rad]}{[\sec]}$$

We define:

$$k^{start} = \left\lfloor \frac{\omega^{init} - \delta_\omega}{\Delta\omega} \right\rfloor - B$$

$$k^{end} = \left\lfloor \frac{\omega^{init} + \delta_\omega}{\Delta\omega} \right\rfloor + B$$

Take the power of

$$P_i = \left|z_i\right|^2 \qquad \forall i$$

Define the PAPR of the power vector:

$$\bar{P}_{avg} = \frac{J_{avg}^{prev} \cdot \bar{P}_{avg}^{prev} + \sum_{\substack{i=0 \\ i \notin \bar{A}}}^{J-1} P_i}{J_{avg}^{prev} + J - \left|\bar{A}\right|}$$

$$\bar{A} = \left[i_{\max} - B : i_{\max} + B\right]$$

$$PAPR = \frac{\max_i \left\{P_i\right\}}{\bar{P}_{avg}}$$

The average in the current iteration is the average over all indices except the vicinity of the max index (assuming it is a valid index that contains the signal). This vicinity is defined as:

$$\bar{A} = \left[i_{\max} - B : i_{\max} + B\right]$$

We choose $B = 2$, because the native resolution is derived from the continuous Fourier transform:

$$Z_s(\theta) = \int_0^T z_s(t) e^{-j\omega t} = \frac{1}{T}\int_0^T e^{j2\pi\omega t} e^{-j2\pi\theta t} dt = \frac{1}{T}\int_0^T e^{-j2\pi(\theta-\omega)t} dt = sinc\left(\frac{\pi}{T}(\theta-\omega)\right)$$

The time duration of the signal is given by:

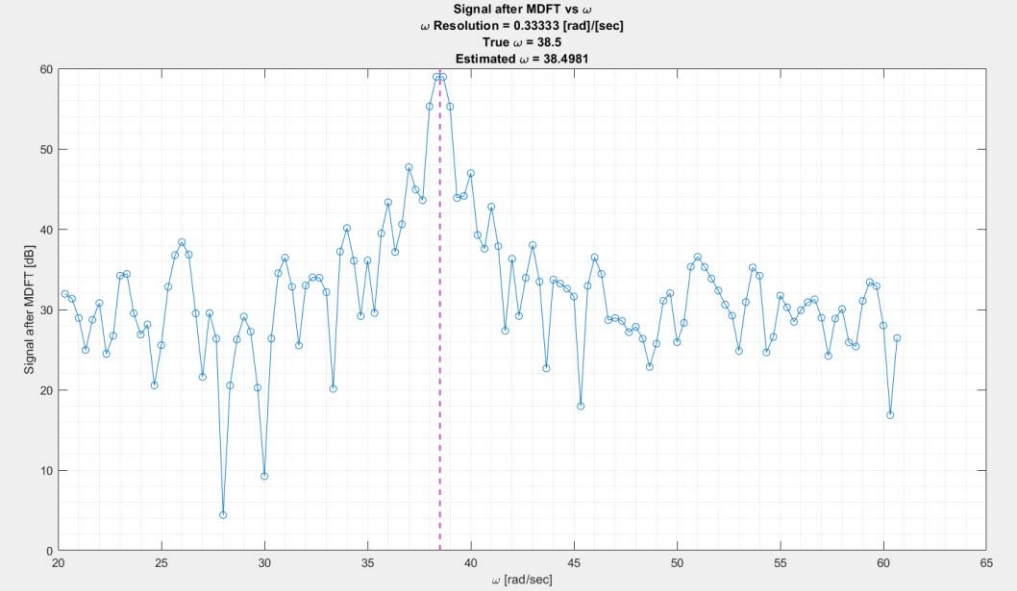
$$T = 1\left[\sec\right] \Rightarrow \omega_{res}^{nativ} = \frac{1}{T} = 1\frac{\left[rad\right]}{\left[\sec\right]}$$

We have chosen the sampled ω resolution:

$$\Delta\omega_{res} = \frac{1}{3}\frac{\left[rad\right]}{\left[\sec\right]}$$

Therefore, we expect to have a main lobe width of 2 samples on each side of a signal driven peak: $B = 2$

We can see it in the following example of $\left|Z\left[k\right]\right|^2$:



If the PAPR in dB is lower than a pre-defined threshold we can say that we didn't find the correct ω, and we need to broaden the search of ω:

$$
\begin{aligned}
&\textit{if} \qquad PAPR_{dB} < PAPR_{dB}^{Th} \qquad \left(=20\left[dB\right]\right) \\
&\qquad J_{avg} = J_{avg}^{prev} + J \\
&\qquad \bar{P}_{avg} = \frac{\bar{P}_{avg}\cdot\left(J_{avg} - \left|\bar{A}\right|\right) + \sum_{i\in\bar{A}}^{J-1} P_i}{J_{avg}} \quad \text{\% The max index belongs to noise} \\
&\qquad \delta_{\omega} \leftarrow \delta_{\omega} + \delta_{\omega}^{init}
\end{aligned}
$$

If the PAPR of this iteration was lower than the threshold, we are updating the average to contain also the indices in $\bar{A}$.

Least Squares Interpolation

We use least squares interpolation for accuracy improvement of the DFT grid.

$\underline{\underline{X}}$ - Coefficients matrix of the powers of index offsets from the max index

$$\underline{\underline{X}} = \begin{pmatrix} x_{-1}^0 & x_{-1}^1 & x_{-1}^2 \\ x_0^0 & x_0^1 & x_0^2 \\ x_1^0 & x_1^1 & x_1^2 \end{pmatrix}, \qquad x_i = i$$

$$\underline{\underline{X}}_R = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{pmatrix}, \qquad \underline{\underline{X}}_L = \begin{pmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \end{pmatrix}, \qquad \underline{\underline{X}}_C = \begin{pmatrix} 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}$$

$$I_{\max} = \arg\max\left\{\underline{P}\right\}$$
$$\underline{P}_R = \left[P_{I_{\max}}, P_{I_{\max}+1}, P_{I_{\max}+2}\right]$$
$$\underline{P}_L = \left[P_{I_{\max}-2}, P_{I_{\max}-1}, P_{I_{\max}}\right]$$
$$\underline{P}_C = \left[P_{I_{\max}-1}, P_{I_{\max}}, P_{I_{\max}+1}\right]$$

Weighted Least Squares Solution:

$$\hat{\underline{\alpha}} = \arg\min_{\underline{\alpha}} \left\{ \overbrace{\left\| \left(\underline{P} - \underline{\underline{X}}\,\underline{\alpha}\right) \right\|^2}^{L(\underline{\alpha})} \right\}$$

$$\frac{\partial L(\underline{\alpha})}{\partial \underline{\alpha}} = 2\left(-\underline{\underline{X}}\right)^H \cdot \left(\underline{P} - \underline{\underline{X}}\,\underline{\alpha}\right) \overset{\underline{\underline{W}} \in \mathrm{Re}}{=} -2\underline{\underline{X}}^H \cdot \underline{P} + 2\underline{\underline{X}}^H \cdot \underline{\underline{X}} \cdot \underline{\alpha} = 0$$

$$\boxed{\hat{\underline{\alpha}} = \left(\underline{\underline{X}}^H \cdot \underline{\underline{X}}\right)^{-1} \cdot \underline{\underline{X}}^H \cdot \underline{P}}, \qquad \hat{\underline{\alpha}}_{3x1} = \left[\alpha_1, \alpha_2, \alpha_3\right]^T$$

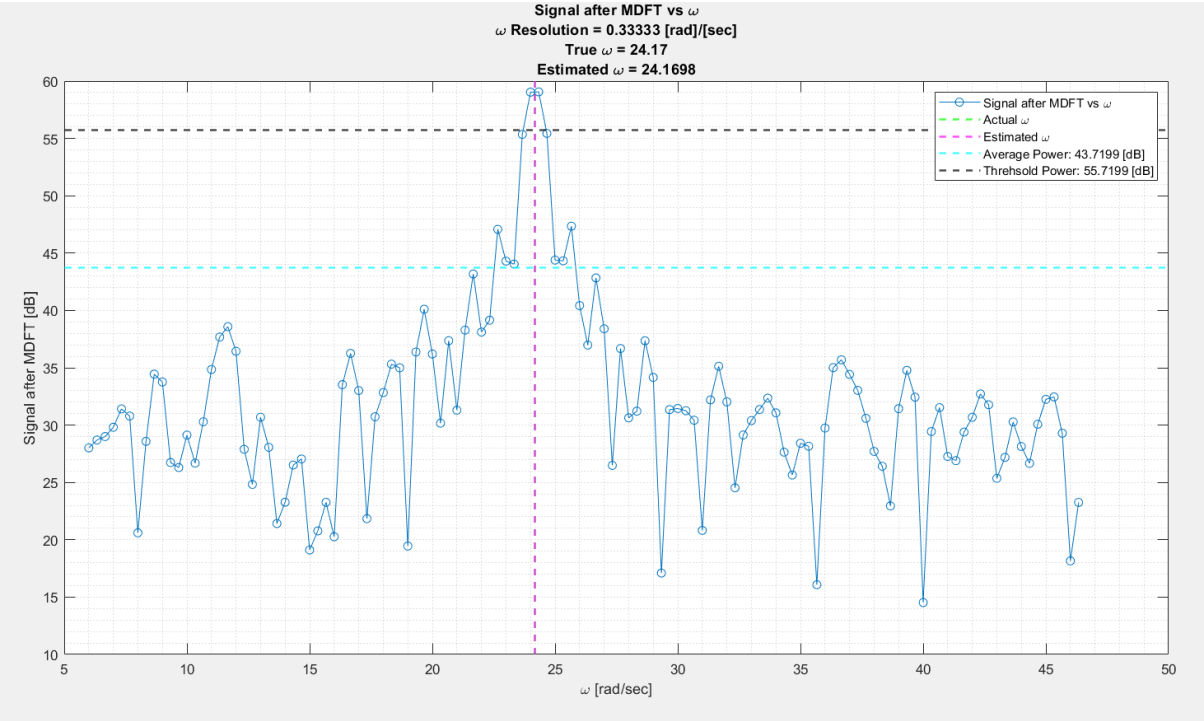
$$\hat{\underline{\alpha}} = \overbrace{\left(\underline{\underline{X}}^H \cdot \underline{\underline{X}}\right)^{-1} \cdot \underline{\underline{X}}^H}^{\underline{\underline{H}}} \cdot \underline{P} = \underline{\underline{H}} \cdot \underline{P}$$

Derivate parabolic polynomial for ω estimation:

$$P = \hat{\alpha}_2 \cdot x^2 + \hat{\alpha}_1 \cdot x + \hat{\alpha}_0 \Rightarrow \frac{\partial P}{\partial x} = 2\hat{\alpha}_2 \cdot \Delta x_{\max} + \hat{\alpha}_1 = 0$$

$$\boxed{\Delta x_{\max} = -\frac{\hat{\alpha}_1}{2\hat{\alpha}_2}} \Rightarrow \boxed{\omega_{\max} = \Delta\omega \cdot \left(I_{\max} + \Delta x_{\max} + k^{start}\right)}$$

The output of the ω estimation:



θ_0 Estimation

After estimating ω we can multiply z by the conjugate of the linear phase phasor:

$$\tilde{z}[i] = z[i]\cdot e^{-j2\pi\hat{\omega}\cdot t_i} = \begin{cases} \overset{z_v[i]}{e^{j2\pi v_i}\cdot e^{-j2\pi\hat{\omega}\cdot t_i}} & w.p. \quad 0.05 \\ \overbrace{e^{j2\pi\omega t_i}\cdot e^{j2\pi(\theta_0+n_i)}}^{z_s[i]}\cdot e^{-j2\pi\hat{\omega}\cdot t_i} & w.p. \quad 0.95 \end{cases} =$$

$$\begin{cases} e^{j2\pi v_i}\cdot e^{-j2\pi\hat{\omega}\cdot t_i} & w.p. \quad 0.05 \\ e^{j2\pi(\theta_0+n_i)}\cdot e^{-j2\pi(\hat{\omega}-\omega)\cdot t_i} & w.p. \quad 0.95 \end{cases} \approx \begin{cases} e^{-j2\pi(\hat{\omega}\cdot t_i - v_i)} & w.p. \quad 0.05 \\ e^{j2\pi(\theta_0+n_i)} & w.p. \quad 0.95 \end{cases}$$

Where n_i is the gaussian noise in the sample i and v_i is the interference noise uniformly distributed with probability 5%.

We learn that we can use the phase of $\tilde{z}_i$ to get a good estimation of θ_0:

$$\tilde{\theta}[i] = \frac{\sphericalangle\tilde{z}[i]}{2\pi} = \begin{cases} \hat{\omega}t_i - v_i & w.p. \quad p_u \\ \theta_0 + n_i & w.p. \quad (1-p_u) \end{cases}$$

$$E\tilde{\theta}[i] = p_u E(\hat{\omega}t_i - v_i) + (1-p_u)E(\theta_0 + n_i) =$$

$$p_u\left(\hat{\omega}t_i - \overset{0.5}{Ev_i}\right) + (1-p_u)\left(\theta_0 + \overset{0}{En_i}\right) \overset{p_u \ll 1-p_u}{\approx} (1-p_u)(\theta_0)$$

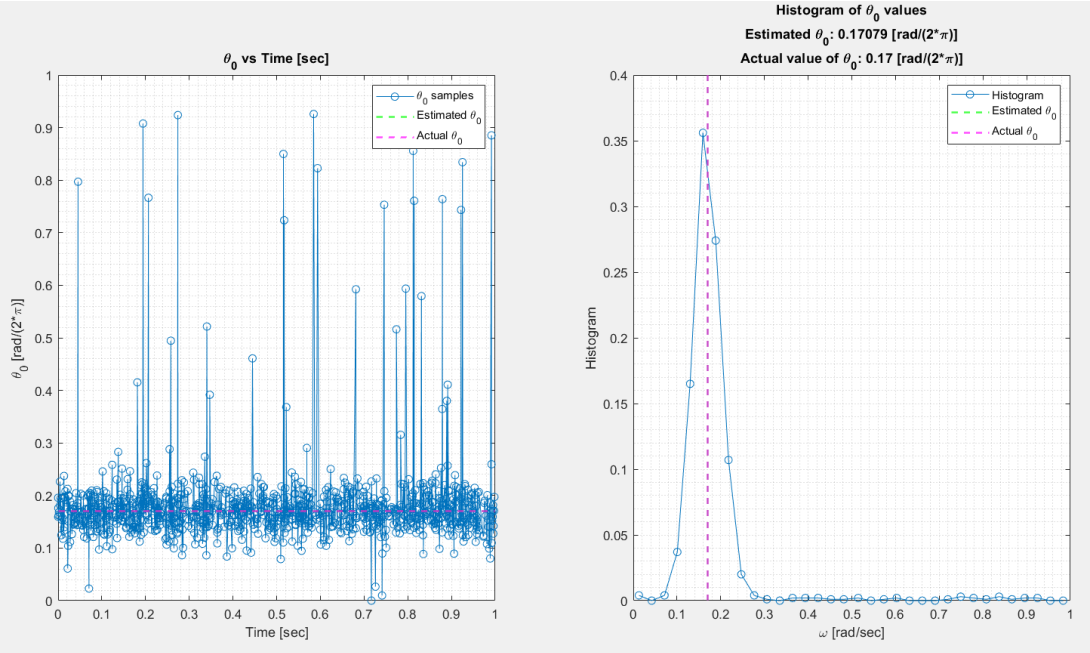
$$\hat{\theta}_0^{(i)} = \frac{\tilde{\theta}[i]}{(1-p_u)} \Rightarrow \hat{\theta}_0 = \frac{1}{N}\sum_{i=0}^{N-1}\hat{\theta}_0^{(i)} = \frac{1}{N}\sum_{i=0}^{N-1}\frac{\tilde{\theta}[i]}{(1-p_u)} =$$

$$\frac{1}{(1-p_u)}\frac{1}{N}\sum_{i=0}^{N-1}\tilde{\theta}[i] =$$

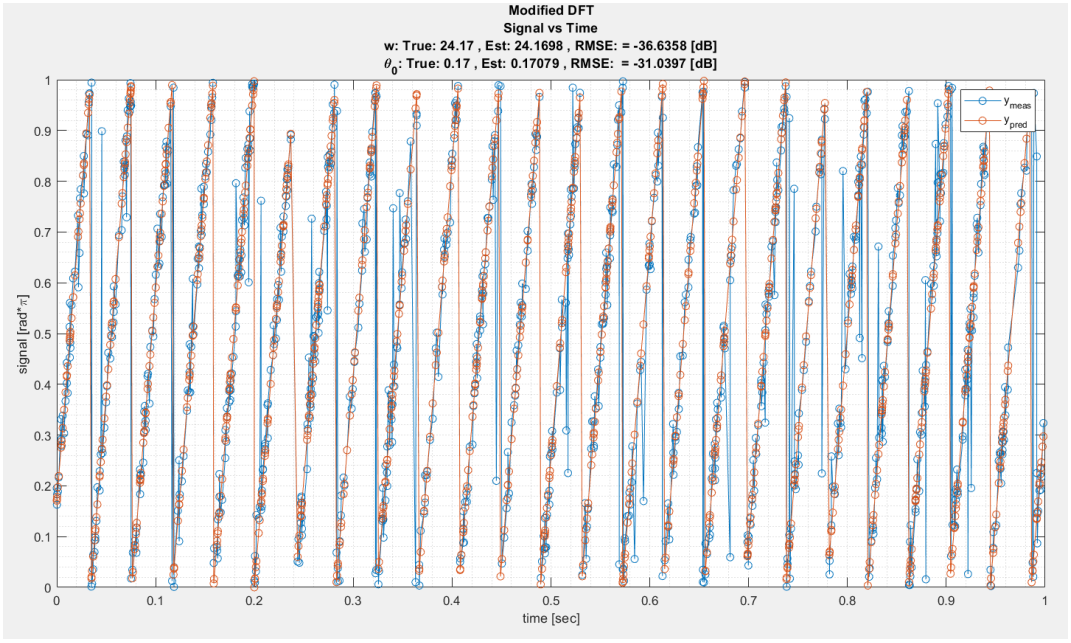
$$\boxed{\hat{\theta}_0 \approx \frac{1}{(1-p_u)}\frac{1}{N}\sum_{i=0}^{N-1}\tilde{\theta}[i]}$$

Since we deal with these spikes in phase v_i, we will use the histogram of $\tilde{\theta}_0$ and take the most likely value to rule out any outliers originating from v_i.

Once we got the histogram peak value we use, again, the weighted least squares parabolic interpolation for better accuracy.



The result is:



We can see the RMSE of the estimation is very low! (around -27[dB])

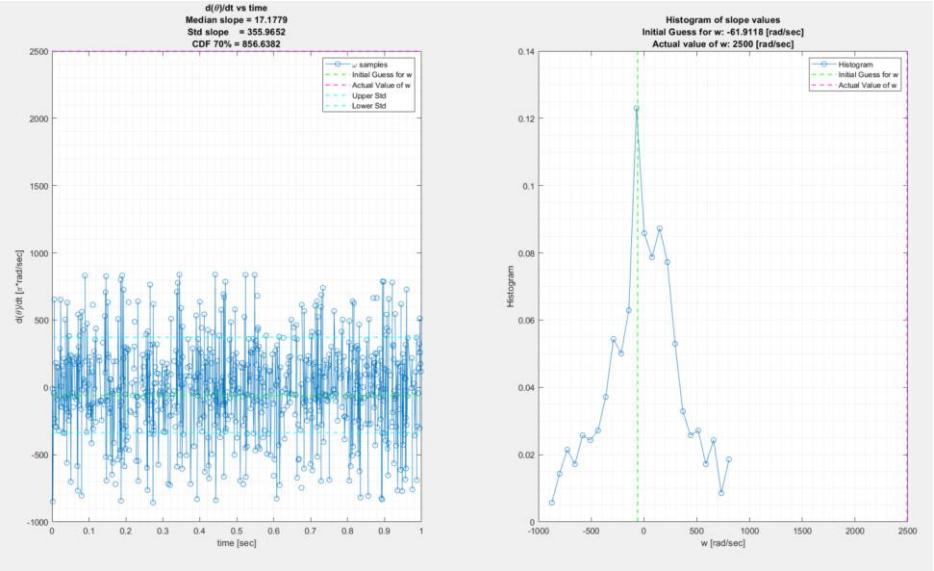
The time complexity of this algorithm is:

$$O\left(N \cdot N_{iter} \cdot \frac{\overbrace{2\left(\delta_\omega + B\right)}^{\substack{Initial\\ Search\\ Area}}}{\Delta\omega}\right) = O\left(N \cdot \overbrace{N_{iter}}^{M} \cdot \frac{\overbrace{2\left(\delta_\omega + B\right)}^{J}}{\Delta\omega}\right) = O\left(NJM\right)$$

Where $M = N_{iter}$ is the number of iterations.

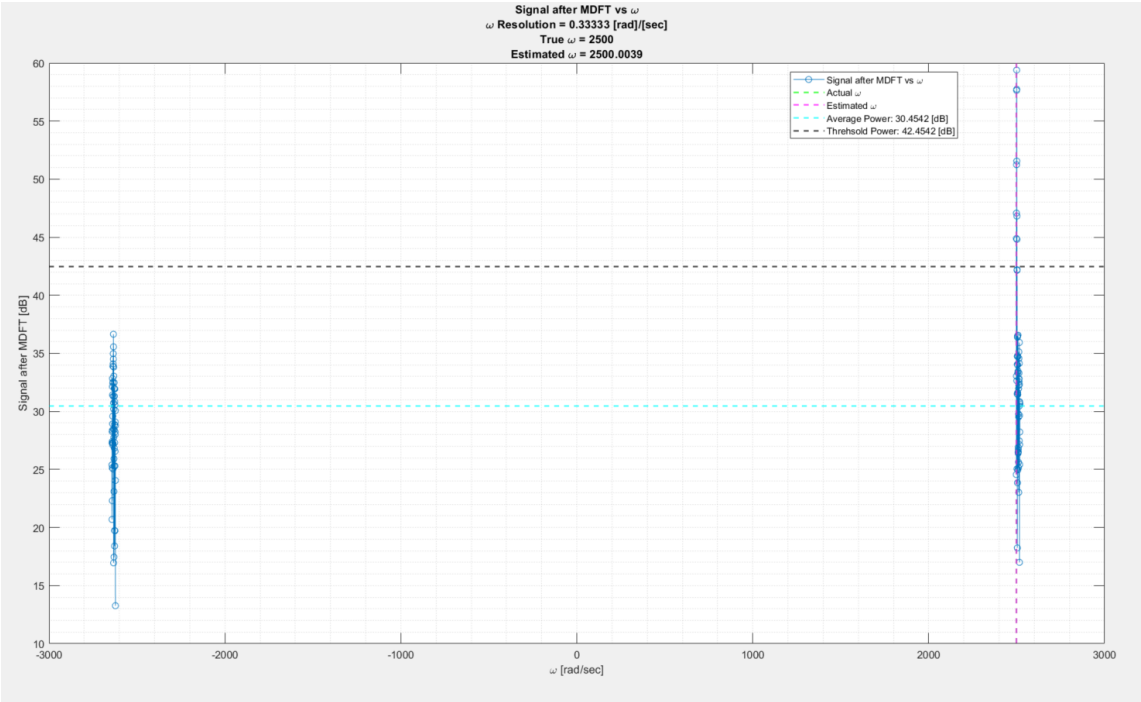
This is the complexity bound of the ω estimation stage, since the complexity of the θ_0 estimation stage is only $O(N)$ due to multiplying the vector $\underline{\tilde{z}}$ by the conjugate linear phase and using only a histogram.

This algorithm is very robust due to its separation of ω, θ_0 estimations, and it's iteratively increasing search area for ω. Thank to the latter we can detect well higher radial velocities than the number of samples (something that is not possible for EKF and PF)

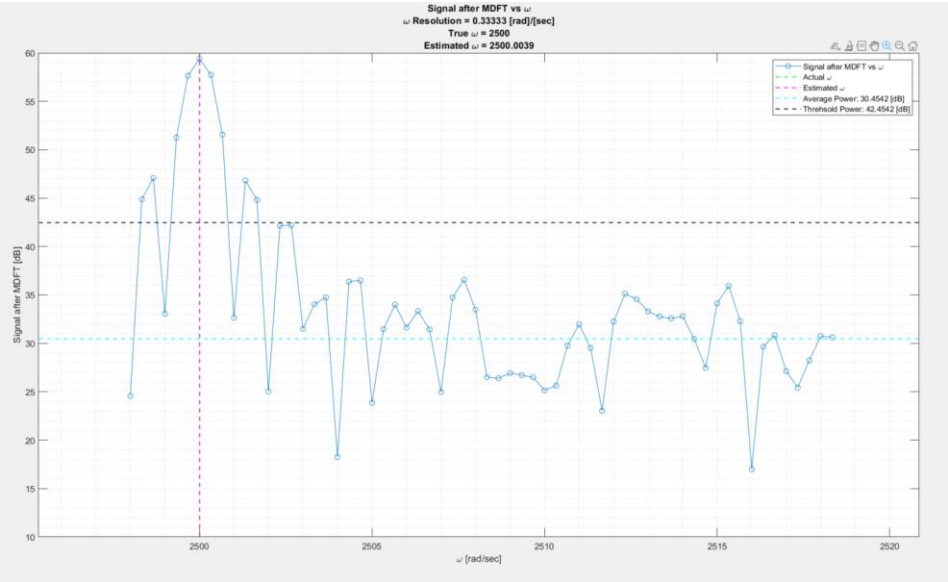


Here we chose $\omega = 2500\frac{[rad]}{[\sec]} > N = 1000[samples]$

And still, we could estimate ω, θ_0 correctly:



Zoom in:




```
Iteration: 120 , Omega PAPR: 6.1833 , Threshold: 20 , Search Area: 2400
Iteration: 121 , Omega PAPR: 5.8854 , Threshold: 20 , Search Area: 2420
Iteration: 122 , Omega PAPR: 3.622 , Threshold: 20 , Search Area: 2440
Iteration: 123 , Omega PAPR: 7.4121 , Threshold: 20 , Search Area: 2460
Iteration: 124 , Omega PAPR: 5.9751 , Threshold: 20 , Search Area: 2480
Iteration: 125 , Omega PAPR: 6.4714 , Threshold: 20 , Search Area: 2500
Iteration: 126 , Omega PAPR: 6.1848 , Threshold: 20 , Search Area: 2520
Iteration: 127 , Omega PAPR: 6.5549 , Threshold: 20 , Search Area: 2540
Iteration: 128 , Omega PAPR: 28.9175 , Threshold: 20 , Search Area: 2560
```

The maximal number of iterations is set to:

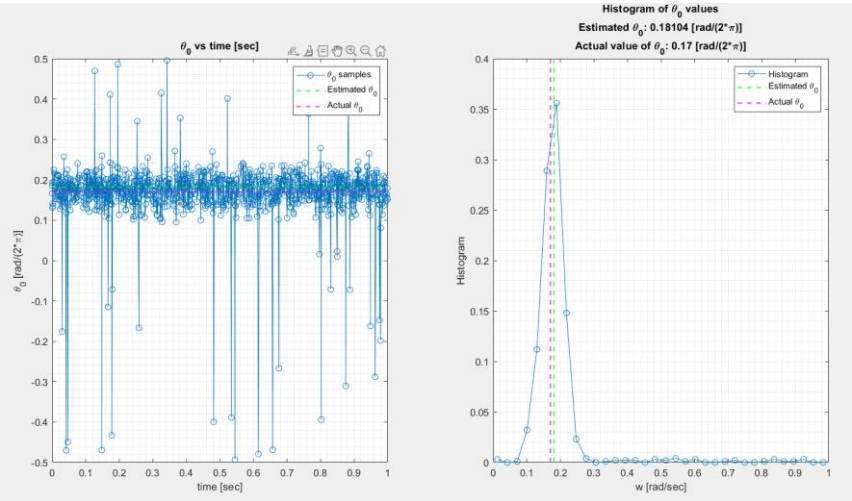
$$N_{iter} = 200$$

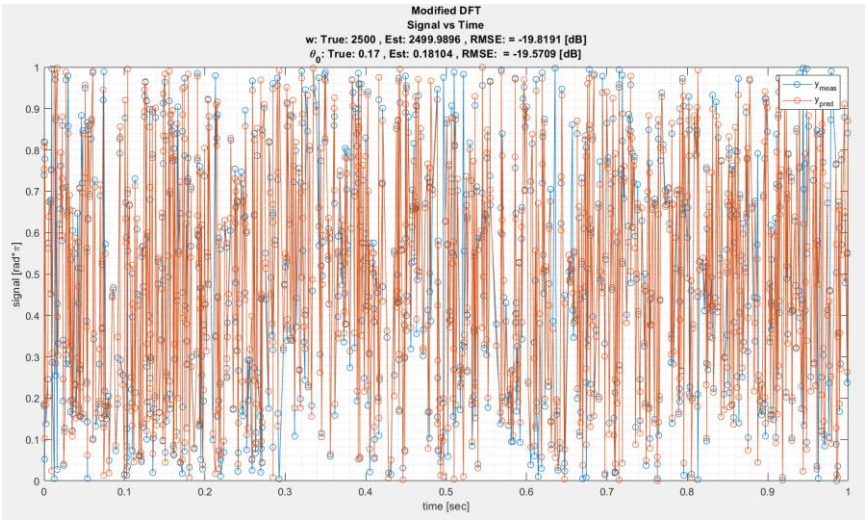
Which means that we can detect ω up to:

$$\omega_{\max} = \omega^{init} + \delta_{\omega} \cdot N_{iter} = \omega^{init} + 20 \cdot 200 = \omega^{init} + 4000 \frac{[rad]}{[\sec]}$$

$$\omega_{\min} = \omega^{init} - \delta_{\omega} \cdot N_{iter} = \omega^{init} - 20 \cdot 200 = \omega^{init} - 4000 \frac{[rad]}{[\sec]}$$

The search area expanded to 2340.


θ_0 vs time [sec]
θ_0 samples
Estimated θ_0
Actual θ_0
θ_0 [rad/(2*π)]
time [sec]
Histogram of θ_0 values
Estimated θ_0: 0.18104 [rad/(2*π)]
Actual value of θ_0: 0.17 [rad/(2*π)]
Histogram
Estimated θ_0
Actual θ_0
Histogram
w [rad/sec]


Modified DFT
Signal vs Time
w: True: 2500 , Est: 2499.9896 , RMSE: = -19.8191 [dB]
θ_0: True: 0.17 , Est: 0.18104 , RMSE: = -19.5709 [dB]
y_{meas}
y_{pred}
signal [rad^-1]
time [sec]

Extended Kalman Filter

We chose the extended version of the Kalman filter due to the non-linearity of the problem caused by the modulo operator. When a gaussian random process is passed through nonlinear system it is not gaussian anymore in the output, hence we cannot use the regular Kalman Filter which is designed for only gaussian noise.

Extended-Kalman-Filter$\left(x_{t-1}, P_{t-1}, y_t\right)$

Init : $\underline{x}_0, \underline{\underline{P}}_0$

$\underline{\bar{x}}_t = \underline{x}_{t-1}$ % State Prediction

$\underline{\underline{\bar{P}}}_t = \underline{\underline{P}}_{t-1} + \underline{\underline{Q}}_t$ % Predicted State Covariance

$\underline{K}_t = \underline{\underline{\bar{P}}}_t \underline{H}_t^T \left(\underline{H}_t \underline{\underline{\bar{P}}}_t \underline{H}_t^T + R_t\right)^{-1}$ % Kalman Gain

$\underline{x}_t = \underline{\bar{x}}_t + \underline{K}_t \left(y_t - h\left(\underline{\bar{x}}_t\right)\right)$ % State Correction

$\underline{\underline{P}}_t = \left(\underline{\underline{I}} - \underline{K}_t \underline{H}_t\right) \underline{\underline{\bar{P}}}_t$ % State Covariance Correction

return: $\underline{x}_t, \underline{\underline{P}}_t$

We initialize the state vector as described above:

$$\underline{x}_0 = \begin{pmatrix} \omega^{init} \\ \theta_0^{Init} \end{pmatrix}$$

We initialize predicted state covariance with the following values:

$$\underline{\underline{P}}_0 = \begin{pmatrix} 10^5 & 0 \\ 0 & 0.05 \end{pmatrix}$$

The reason for that high covariance for ω is that we want to converge to its right value as fast as we can so we can converge also to theta. This originates from the fact that an error of ω is more significant for the total RMSE between the actual signal y and the predicted signal y^{pred} due to the accumulative nature of the slope of a linear line.

The process noise covariance matrix $\underline{\underline{Q}}$ is defined as:

$$\underline{\underline{Q}} = \begin{pmatrix} 1 & 0 \\ 0 & 10^{-3} \end{pmatrix}$$

After the initial state we want to converge slowly to the actual state value hence these values are much lower than that of $\underline{\underline{P}}_0$, however we still give the process of ω higher noise variance than θ_0 due to the reason explained above.

The transformation function from the state vector to the samples is:

$$h(\underline{x}) = \mathrm{mod}(x_0 \cdot t + x_1, 1)$$

In EKF we calculate the Jacobian of $h(\underline{x})$ to get $\underline{H}$:

$$\underline{H} = \left[\frac{\partial h}{\partial x_0}, \frac{\partial h}{\partial x_1} \right]$$

Because we have modulo operator in h we derivate h in the discrete domain:

$$h_\omega = \frac{\partial h(x_0, x_1)}{\partial x_0} = \frac{h(x_0 + \delta, x_1) - h(x_0, x_1)}{\delta}$$

$$h_{\theta_0} = \frac{\partial h(x_0, x_1)}{\partial x_1} = \frac{h(x_0, x_1 + \delta) - h(x_0, x_1)}{\delta}$$

$$\underline{H} = \left[h_\omega, h_{\theta_0} \right]$$

The measurements noise covariance (variance) is:

$$R = 0.2 + \sigma^2$$

We added 0.2 to regard the random spikes we have in the phase with probability 5%

The higher the Kalman gain is the more we trust in the received samples y_t and vice versa.

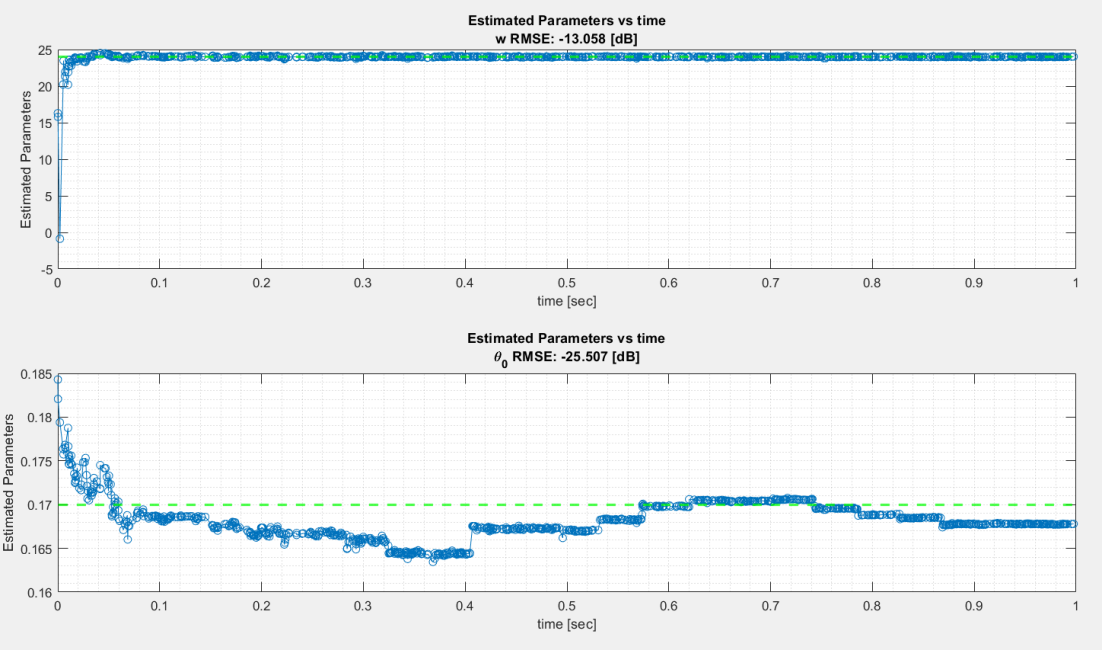
We added a protection for high measurement errors:

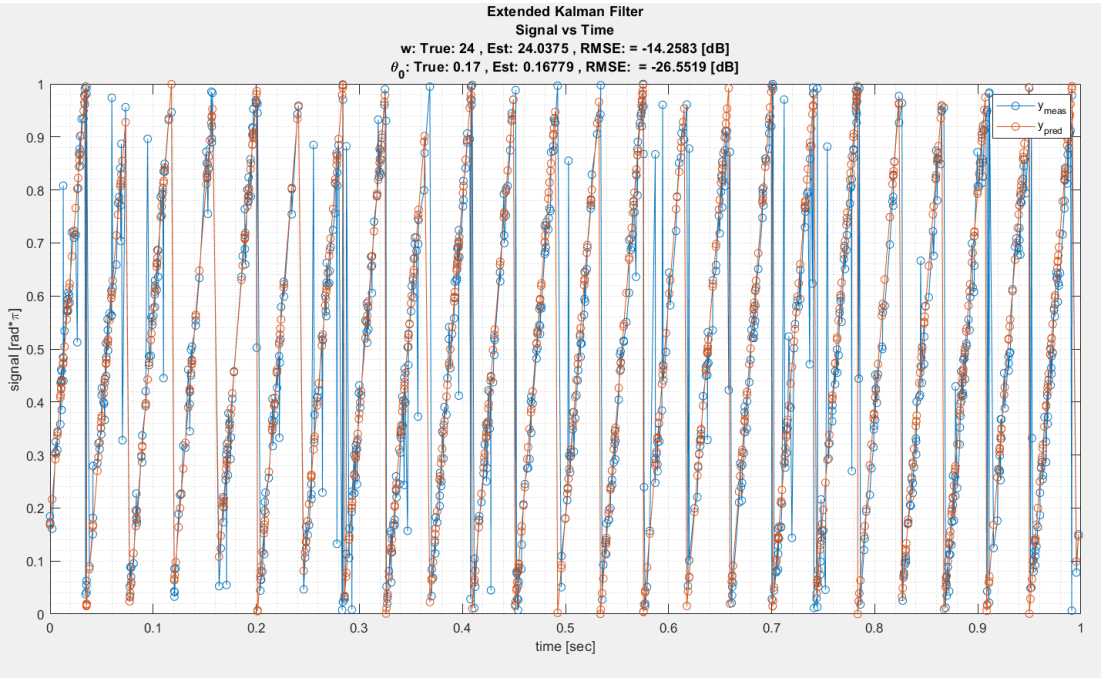
$$if \quad \left| y_t - y_t^{pred} \right| > 0.2$$

$$K_t = 0$$

We set the Kalman gain to 0, which means we do not trust the measurements at that time (probably the phase spikes or the modulo wrap around)

Performance of a single run of the EKF:





We can see that we were able to converge to the actual values of ω, θ_0 with rather good RMSE.

The time complexity of this algorithm is:

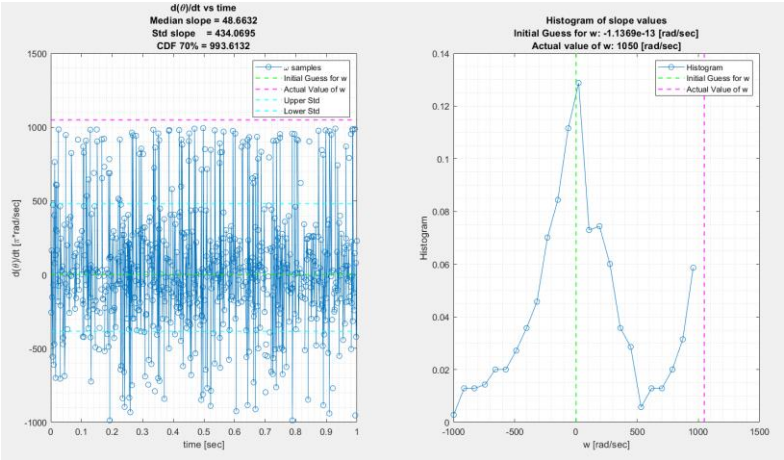
$$O(N)$$

Since in each iteration the calculation is $O(1)$ (or in fact the dimension of the parameter domain which is 2)

This algorithm is the only one of the 4 algorithms that can produce an estimate each time sample, due to its tracking nature.

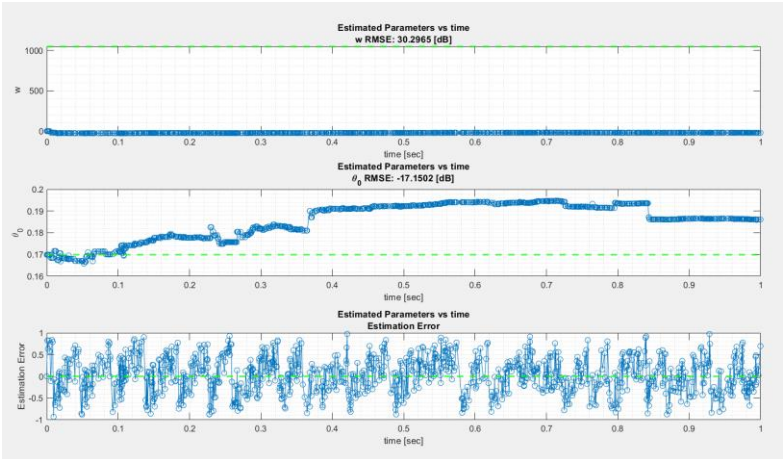
This algorithm works well when the initial guess of ω is rather close to the true value of ω, due to its tracking nature. Hence, we will get bad results for: $\omega > N$

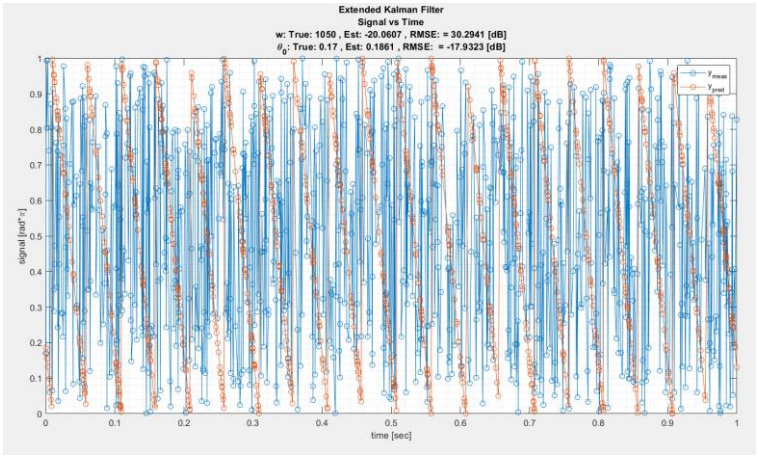
Let's look on a case when $\omega = 1700 > N = 1000$



The derivative of the input signal does not give us any information on ω, because of the modulo operation. Instead, we detect the ω as close to 0 by mistake.

That can lead to a false behavior of the Kalman filter:





Particle Filter

The concept of the particles is searching for the desired parameters in a dynamic hypothesis domain, where each one of the particles is in fact a hypothesis for ω, θ_0

$$\begin{aligned}
&\text{Particle_Filter}\left(\underline{\underline{X}}_{i-1}, y_i, t_i\right): \\
&\quad Init: \underline{\underline{X}}_0, \underline{W}_0 \quad \%\,\text{Init particles and their weights} \\
&\quad \underline{\underline{\tilde{X}}}_{i-1} = \text{Add_Gaussian_Noise_to_Particles}\left(\underline{\underline{X}}_{i-1}\right) \\
&\quad \underline{y}^{pred} = \text{Predict_signal}\left(\underline{\underline{\tilde{X}}}_{i-1}, \underline{t}\right) \\
&\quad \underline{W}_i = \text{Generate_weights}\left(\underline{y}, \underline{y}^{pred}, \underline{W}_{i-1}\right) \\
&\quad \underline{\underline{X}}_i = \text{Low_Variance_Resampling}\left(\underline{\underline{\tilde{X}}}_{i-1}, \underline{W}_i\right)
\end{aligned}$$

We first initialize the particles (initial hypothesizes) the following way:

$$\begin{aligned}
&\underline{\tilde{\omega}}^{(0)} \sim N\left(\omega^{init} \cdot \underline{1}, \sigma^2_{\omega,init} \cdot \underline{\underline{I}}\right) \\
&\underline{\tilde{\theta}}_0^{(0)} \sim \text{mod}\left(N\left(\theta_0^{init} \cdot \underline{1}, \sigma^2_{\theta_0,init} \cdot \underline{\underline{I}}\right), 1\right) \\
&\sigma_{\omega,init} = S_\omega \cdot \omega^{init}, \qquad S_\omega = 0.2 \qquad \%\,\text{Scale factor} \\
&\sigma_{\theta_0,init} = 0.2
\end{aligned}$$

We initialize the weights of each particle to be equal as we do not know yet anything about the correlation of each particle to the measurements.

$$w_i = \frac{1}{J}$$

Where J is the number of particles. In our case $J = 200$

In each step we add the previously generated particles a gaussian noise as a step of optimization (like mutation in evolution).

$$\begin{aligned}
&\underline{\tilde{\tilde{\omega}}}^{(i)} = \underline{\tilde{\omega}}^{(i-1)} + N\left(0, \sigma^2_\omega \cdot \underline{\underline{I}}\right) \\
&\underline{\tilde{\tilde{\theta}}}_0^{(0)} \sim \text{mod}\left(N\left(0, \sigma^2_{\theta_0} \cdot \underline{\underline{I}}\right), 1\right) \\
&\sigma_\omega = 0.2 \\
&\sigma_{\theta_0} = 0.01
\end{aligned}$$

We chose the noise sigma of θ_0 to be smaller than that of ω because of the reason explained in the EKF section.

We use IIR filtering after the addition of the Gaussian noise so the change will be smoother, and we will not miss any minimum in the cost function.

$$\underline{\tilde{\omega}}^{(i)} = \alpha_{\omega}^{IIR} \cdot \underline{\tilde{\omega}}^{(i-1)} + \left(1-\alpha_{\omega}^{IIR}\right) \cdot \underline{\tilde{\tilde{\omega}}}^{(i-1)}$$
$$\underline{\tilde{\theta}}_0^{(i)} = \alpha_{\theta_0}^{IIR} \cdot \underline{\tilde{\theta}}_0^{(i-1)} + \left(1-\alpha_{\theta_0}^{IIR}\right) \cdot \underline{\tilde{\tilde{\theta}}}_0^{(i-1)}$$

When we choose large α^{IIR} we take more of the previous particles and less of the new noised ones.

$$\alpha_{\omega}^{IIR} = 0.8$$
$$\alpha_{\theta_0}^{IIR} = 0.9$$

We chose θ_0 to be higher than that of ω because we want a more subtle change in the particle of θ_0.

Then we predict the signal y^{pred} for each of the particles:

$$\underline{y}^{pred} = \mathrm{mod}\left(\underline{\tilde{\omega}} \cdot \underline{t} + \tilde{\theta}_0, 1\right)$$

Generating the weights is done by calculating the Mahalanobis distance between the measured y to the predicted y^{pred}:

$$D_{mahalanobis} = \frac{1}{N} \sum_{\substack{i=0 \\ i \notin Outliers}}^{N-1} \frac{\left|y_i - y_i^{pred}\right|^2}{\sigma^2}$$

Where $\sigma = \max\left(\sigma_{noise}, \sigma_{\min}\right)$ $\sigma_{\min} = 10^{-3}$

For these cases we run the code without noise.

We determine the outliers by calculating the CDF of the $d_i = \frac{\left|y_i - y_i^{pred}\right|^2}{\sigma^2}$ for all i and removing the d_i that are higher than the CDF 70%

Finally, the weights are calculated by:

$$w_i = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2} D_i^{mahalanobis}}$$

Normalize the weights:

$$w_i = \frac{w_i}{\sum_{j=0}^{N-1} w_j}$$

We check if the weights are valid (not spread even as the initial weights) due to too large $D_i^{mahalanobis}$ numbers (happen a lot when we get a new spike), in that case we do not update the weights and particles, waiting for a better sample in further iterations for better convergence.

Now we resample the particles according to the weights vector.

Particles that correspond to higher weight will be get more instances of new particles, and vice versa.

$\text{Low\_Variance\_Resampling}\left(X_t, W_t\right):$

$step = \frac{1}{J}$

$r = step \cdot rand\left(1\right)$

$c = W_t\left[0\right]$

$i = 1$

$\bar{X} = \underline{0}$

$for \;\; j = 1:J$

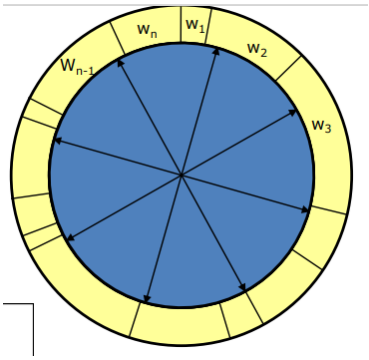
$U = r + \left(j-1\right) \cdot step$

$while \; U > c \;\&\; i < J-1$

$i += 1$

$c += W_t\left[i\right]$

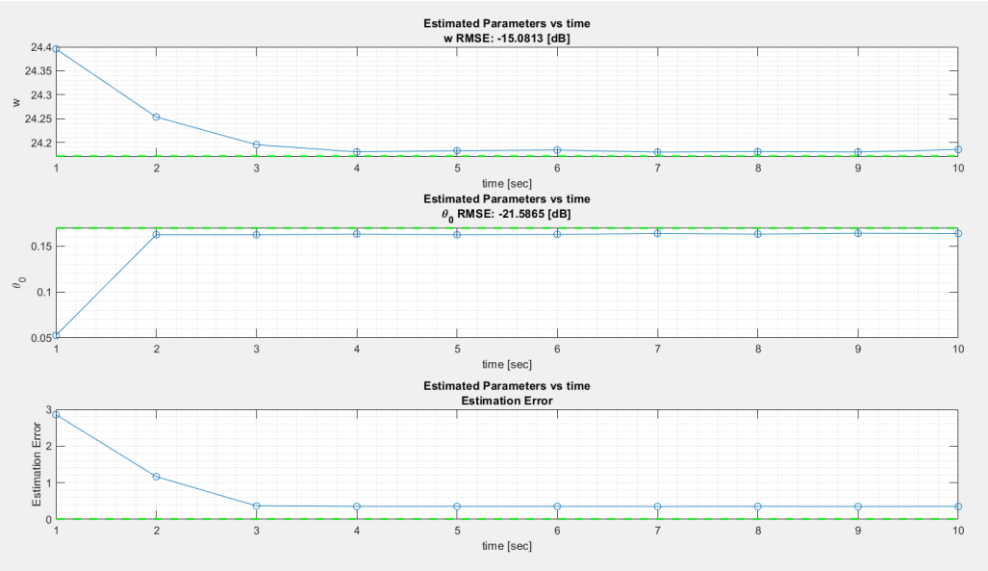
$\bar{X}_t\left[j\right] = X_t\left[i\right]$



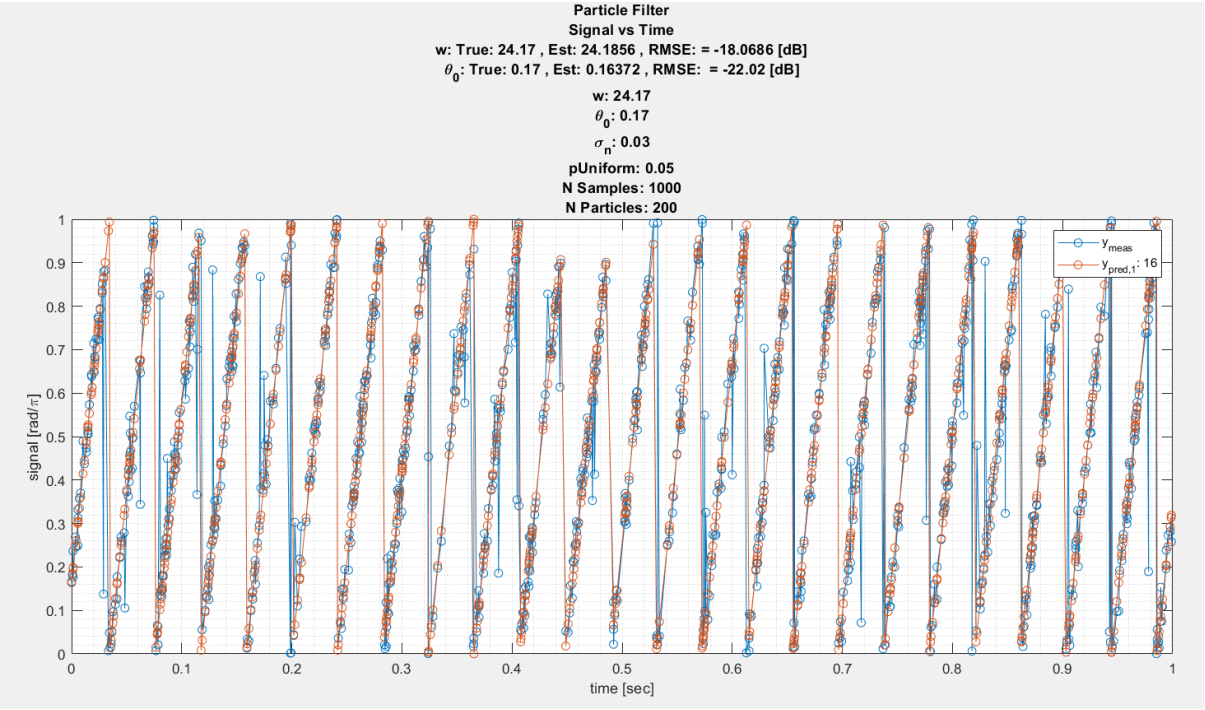
For the ω estimation we take the strongest weight:

$$\hat{\omega} = \max_i \left\{W_t\left[i\right]\right\}$$

The parameters estimation convergence over iterations:



The performance of the algorithm:



We can see that the RMSE of the particle filter is not the best but still rather good.

The computation complexity is:

$$O(NJM)$$

Where:

- N is the number of samples
- J is the number of particles
- M is the number of maximum iterations

The particle filter algorithm works well when the initial guess is rather close to the true value (as in EKF).

This algorithm has many higher-level parameters which makes it very difficult to calibrate and less robust.

Grid Search

The concept of the grid search is to scan ω, θ_0 on a 2D grid and converge to the global minima of the minimum distance objective by zooming around the minimum of each iteration.

The minimum distance objective is calculated by taking the distance between the measured signal and the predicted signal per each ω, θ_0 :

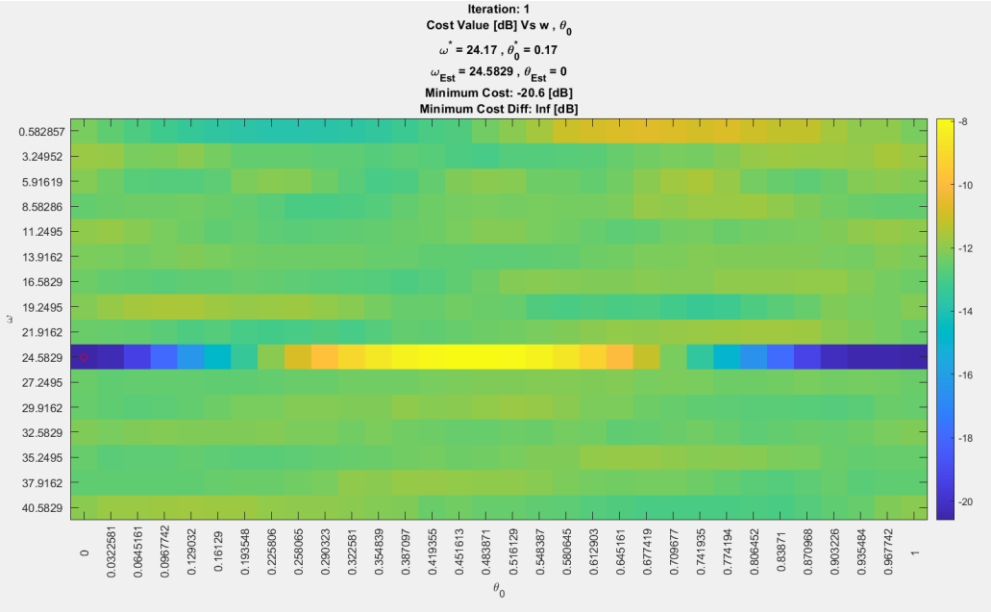
$$\underline{y}^{pred} = \mathrm{mod}\left(\tilde{\underline{\omega}} \cdot \underline{t} + \tilde{\theta}_0, 1\right)$$

The distance objective:

$$d_i(\omega, \theta_0) = \left| y_i^{meas} - y_i^{pred}(\omega, \theta_0) \right|^2$$

We calculate the average of the $d_i(\omega, \theta_0)$ for $\forall i \notin Outliers$ (as mentioned above we discard all the distances that are higher than 70% CDF, to avoid averaging also on the phase spikes with probability of 5%).

First iteration grid:



We can see that the ω estimation is sharper than the θ_0 due to the explanation above that the consequence on the distance between the predicted and measured signal is higher when we have an estimation error of ω rather than θ_0

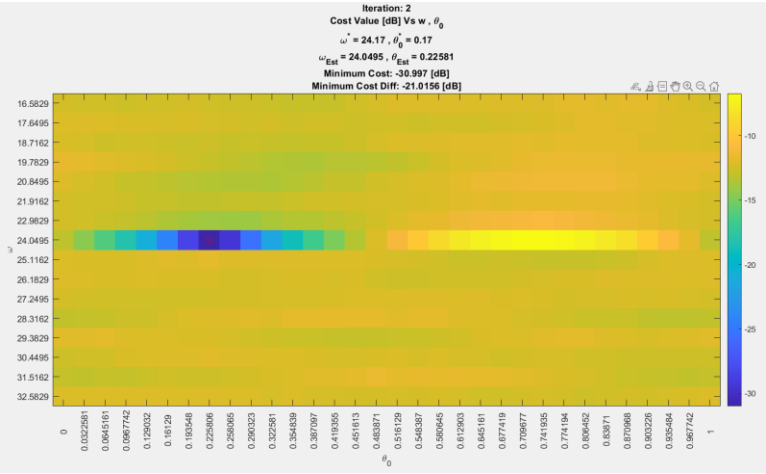
Adjusting the grid search area

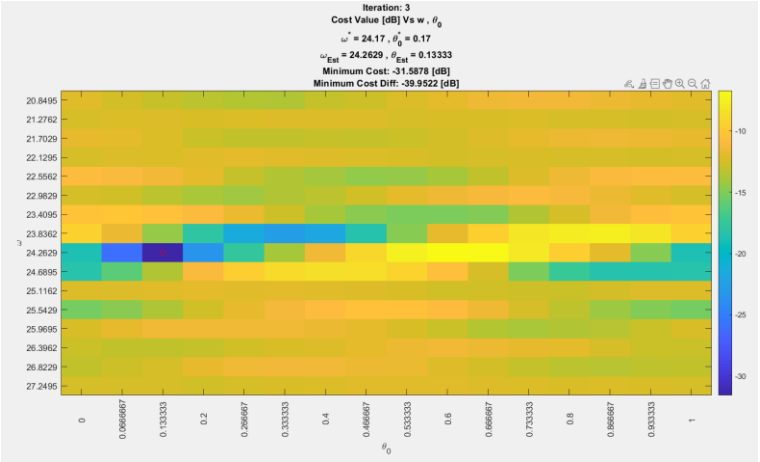
We adjust each axis separately: ω, θ_0

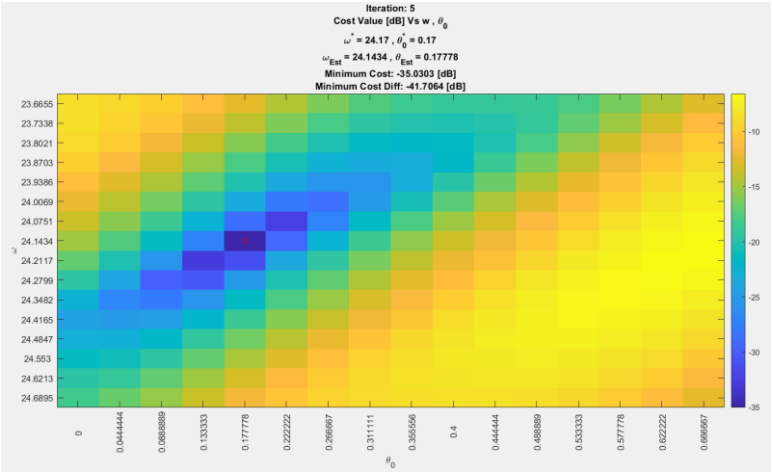
If the minimum in inside the region of the grid, we zoom in ω axis by the factor of 0.4 and θ_0 by the factor of 0.8 (slower rate because we trust the accuracy of ω better than θ_0)

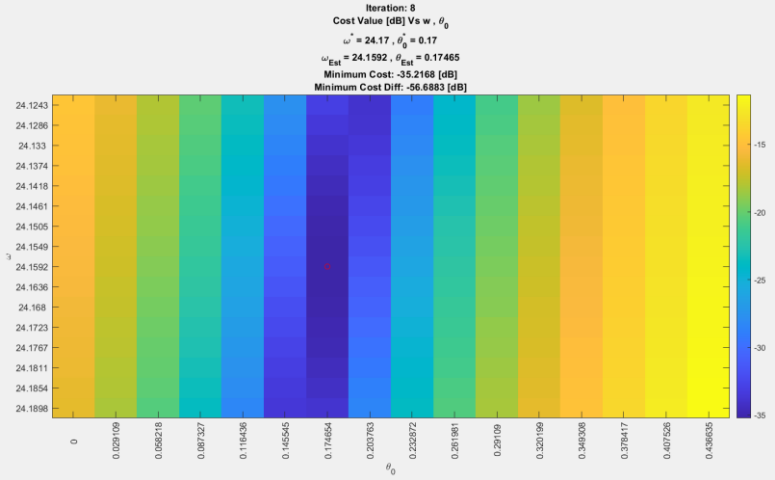
If the minimum is in the edge of the grid, we jump the grid area by the factor 1.5 to the side of the minimum location (we have overlapping grids each iteration)

Further iterations:

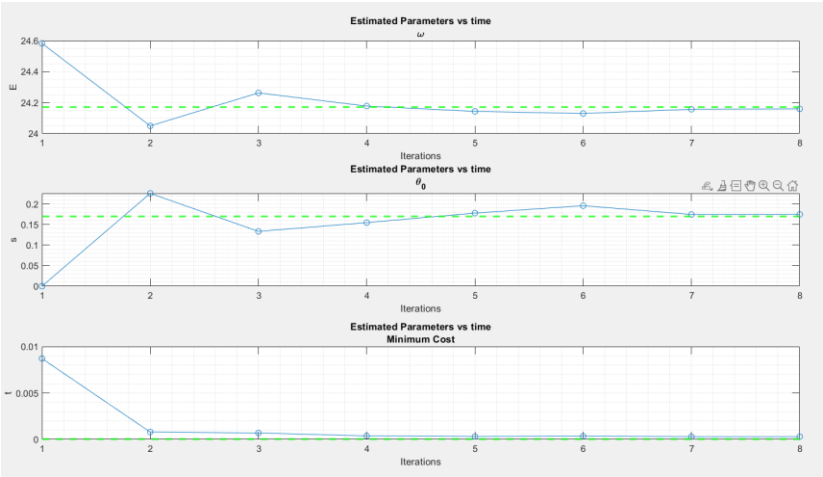






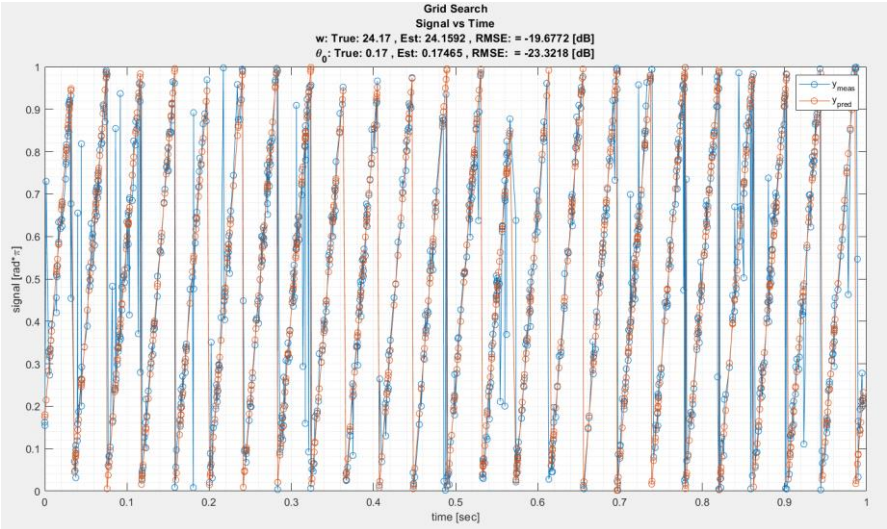


Parameters estimation convergence:



We can see that the parameters converge to the true ones and the minimum cost is going down to 0 with each iteration.

The performance of the algorithm:



We see that the RMSE is rather low (around -19.6[dB] for ω and -23.3[dB] for θ_0)

The computation complexity is:

$$O(NJM)$$

Where:

- N is the number of samples
- $J = J_\omega \cdot J_\theta$ is the number of grid elements of ω, θ_0
- M is the number of maximum iterations

This algorithm is also vulnerable to the initial guess accuracy. (as EKF and PF)

Performance Table

Algorithm	$\hat{\omega}$ Average RMSE [dB]	$\hat{\theta}_0$ Average RMSE [dB]	Complexity
MDFT	-25.1331	-25.8557	$O(NJM)$
EKF	-4.7467	-14.8965	$O(N)$
PF	-6.0910	-15.6316	$O(NJM)$
GS	-12.4154	-19.2206	$O(NJM)$

MDFT is the best in the means of RMSE.

EKF is the best in the means of computation complexity due to its tracking nature, though is uses the least information each iteration step, therefore its performance is the worst.

PF is better than EKF since it uses all the signal at each iteration, though its heavier in computation complexity

GS is better than EKF and PF in the means of performance, and like PF in the means of complexity but still worse than MDFT.