Project V: Signal Modelling

Simon Andreas Bjørn
March 14, 2023

» Functions for linear prediction

We start by implementing the 2 possible methods for linear prediction.

Implemented by folloing lecture notes

```
function corrpred(x, p, r)
    N = length(x)
    X = zeros(N+p, p)
    for i=1:p
        X[i:i+N-1,i] = x
    end

    b = [zeros(p);x]
    a = [1; vec(reverse(-X \ b))]
    err = mean((x - filt(1.0, a, [1; zeros(N-1)]
      )).^2)
    return [a, err]
end
```

Implemented by following the book

```
function covpred(x, p, r)
    N = length(x)
    X = zeros(N+p, p)
    for i=1:p
        X[i:i+N-1,i] = x
    end
    X = X[p:N-1,:]
    b = x[p+1:N]
    a = [ 1; -X \ b ]
    err = mean((x - filt(1.0, a, [1; zeros(N-1)]
      )).^2)
    return [a, err]
end
```

Testing both function on the filter `b=1.0, a=[1.0, 0.2, 0.3]`, I get

```
x = filt(1.0, [1,0.2,0.3], [1; zeros(100)])
@show corrpred(x, 2, 0); # --> corrpred(x, 2, 0) = Any[[1.0, 0.2, 0.3], 4.523e-35]
@show covpred(x, 2, 0);  # --> covpred(x, 2, 0)  = Any[[1.0, 0.2, 0.3], 5.611e-35]
```

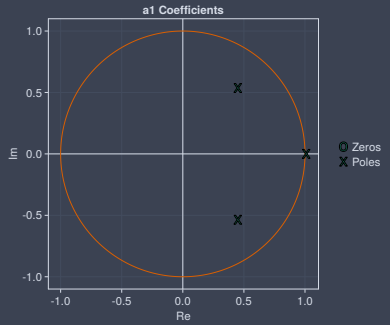
» Testing code on ARdata

We now want to test our code on the provided `ARdata.mat` file.

```
@show a1 # --> a1 = [1.0, -1.9099, 1.3989, -0.4949]
@show corrpred(x1, 3, 0); # --> corrpred(x1, 3, 0) = Any[[1.0, -1.1126, 0.1291, 0.0061], 21.1632]
@show covpred(x1, 3, 0); # --> covpred(x1, 3, 0) = Any[[1.0, -1.9099, 1.3989, -0.4948], 8.8982]
```

Then we want to plot a pole-zero plot of the true coefficients. No in-built pole-zero plot, so define a simple plot function.

```
function zplane!(ax::Axis, sys::LTISystem)
    z = tzeros(sys)
    p = poles(sys)
    return [
        scatter!(ax, real(z), imag(z), marker='o'),
        scatter!(ax, real(p), imag(p), marker='x')
    ]
end
```

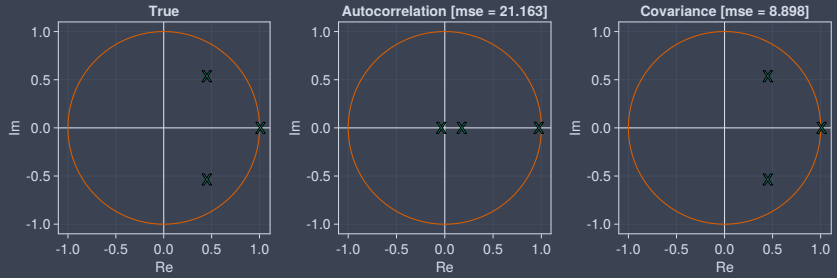


» Pole-zero plot of coefficients from methods

Now we want to plot the coefficients we get from predicting the coefficients. Estimating the coefficients, I create a transferfunction per method and plot the resulting LTISystem using `zplane()` function.

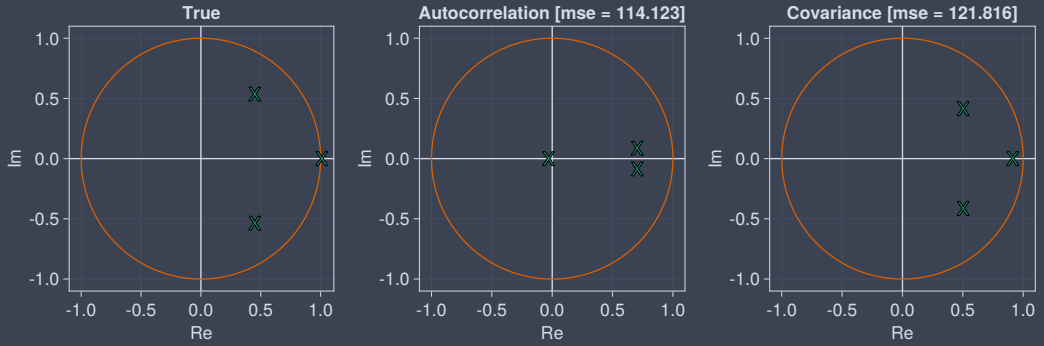
```
a1_corr, err1_corr = corrpred(x1, 3, 1)
a1_cov, err1_cov   = covpred(x1, 3, 1)
zplane!(ax1, tf([1.0], a1)); zplane!(ax2, tf([1.0], a1_corr)); zplane!(ax3, tf([1.0], a1_cov))
```

and get the following plots



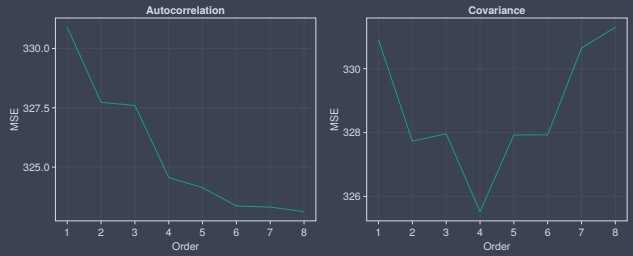
» Pole-zero plot of x2

Now we do the same but predicting on the x2 sequence.



» Iterating order and estimating x3

We want to loop over the filter order and estimate the filter coefficients for x3. Plotting the MSE of each method per iteration, I have



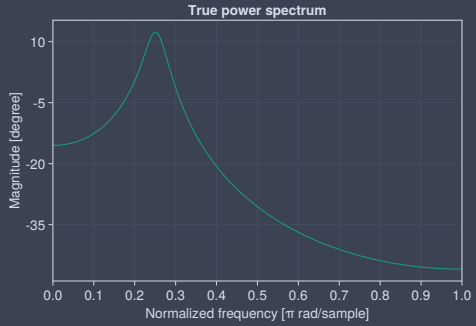
- Covariance is a bit strange after order $P = 4$. Should flatten out.
- MSE is within the same order of magnitude between the methods
- Using the solution provided in the assignment, I conclude with P=4 as a good estimate of the filter order.
- Flattening out means no more dimensions can improve the estimate.

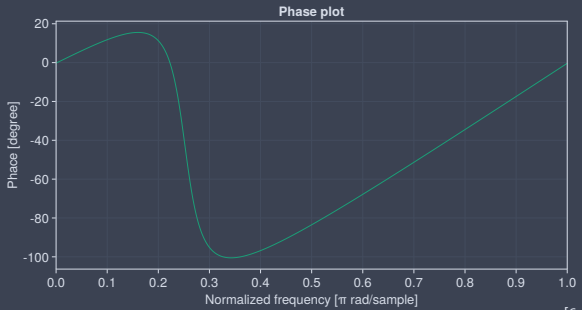
» Frequency response of new filter

Now we have a new filter with `b = 5; a = [1, -1.3, 0.845]`. Now we want to plot the true power spectrum.

```
x = [1; zeros(N-1)]; h = filt(b, a, x)
H_true = fftshift(fft(h))[end÷2:end];
P_h_true = abs.(H_true.^2); P_h_true /= length(P_h_true)
```

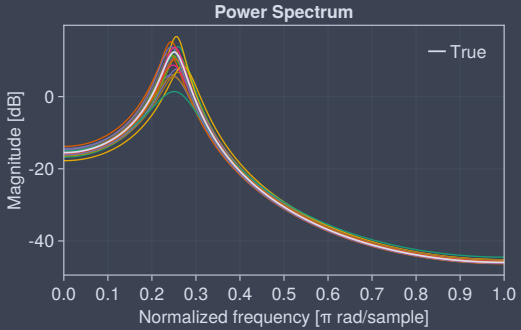
Plotting the true power spectrum and the phase angle, I get the following plots.

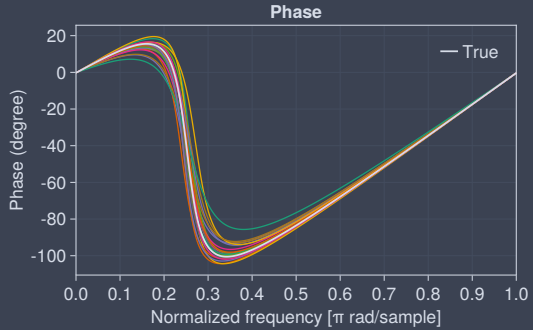




» Noise driven filten experiment

Now we want to estimate the coefficients of the filter using random noise. Instead of using the impuls response, we use gaussian white noise. Estimating this and plotting similarly to the previous plots, I get

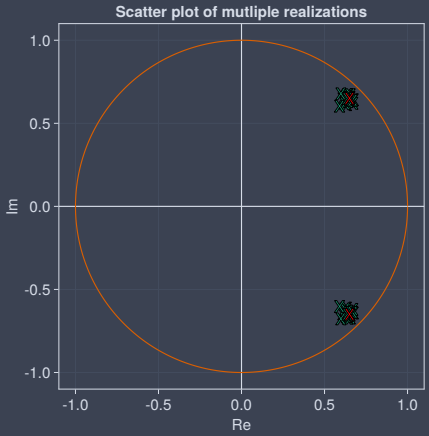




» Pole-zero plot from noise

Now, using the coefficients found in the previous task, we plot a pole-zero plot.

* Small scattering a round true poles

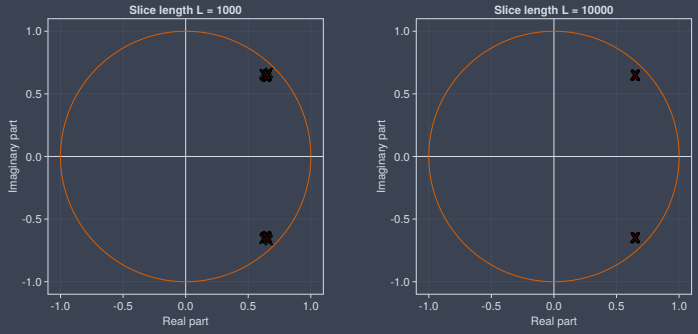


» Testing with longer sequences

Now we increase the number of samples we use to estimate. Code is then

```
x = randn(N)
h = filt(b, a, x)
h_samples = h[20:20+1000]
h_samples2 = h[20:20+10_000]
```

Plotting this I get



» Tapering the sequence

Very lastly, we want to try and taper the x-sequences with a Hamming window, and compare them to using unit weights (boxcar). Code is then

```
h1 = filt(b, a, x)
h2 = filt(b, a, x.*hamming(N))
```

Plotting them to compare with both $L = 128$ and $L = 256$, I get the following plots