Table of Contents

ECE537: Lab 5 Report

*It is recommended to access this report by opening the `html` file on the browser or by clicking **here**.*

Throughout this lab, the **Distributions.jl** package in Julia has been utilized to be able to use the probability constructs in code.

```
using Distributions , StatsBase , Plots , LinearAlgebra , LaTeXStrings ,
PlutoUI , DSP , FFTW
```

1. Autoregressive Filters and Coefficient Estimation

Here we consider a system excited by i.i.d. white Gaussian noise $N[n] \sim \mathcal{N}(0,1)$ and produces a random sequence $X[n]$ as output by the following autoregressive (AR) difference relation,

$$X[n] = 1.5X[n-1] - 0.8X[n-2] + N[n]$$

We will now simulate this process in the code below.

```
Nₙ (generic function with 1 method)
```

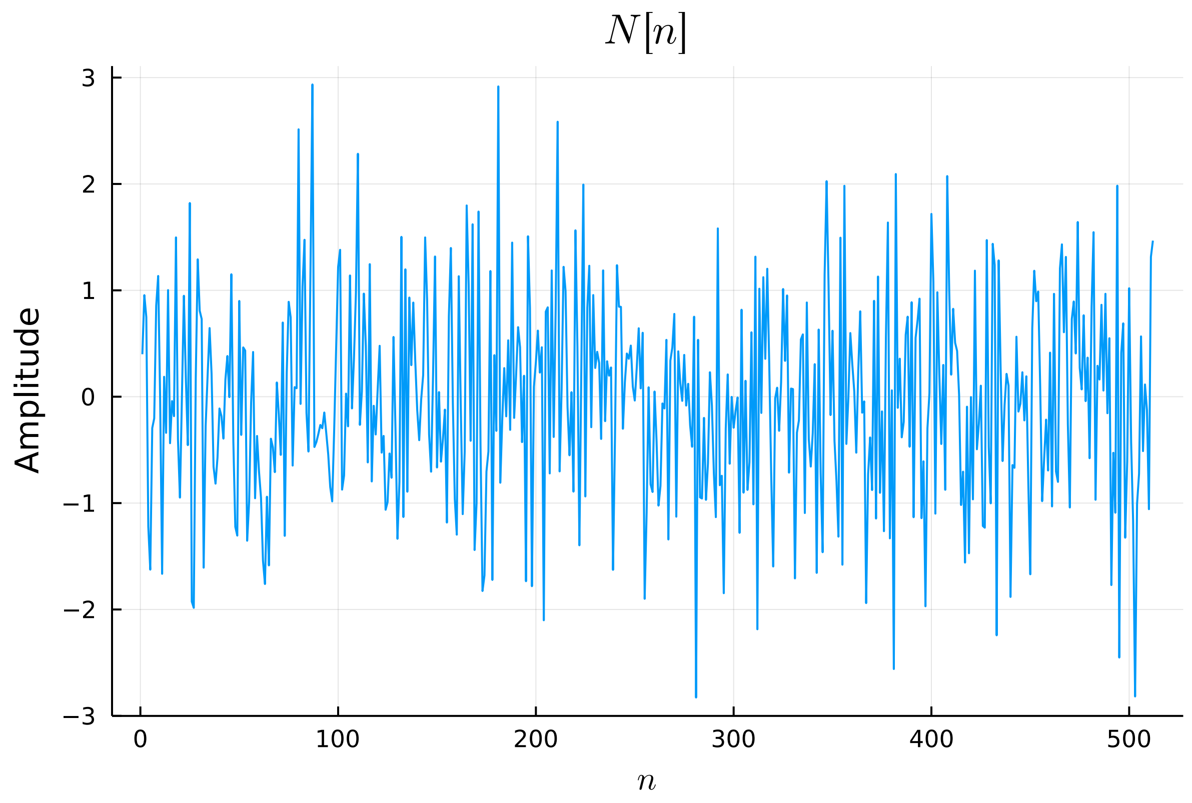
```
Nₙ(n) = [Normal(0,1) for k ∈ 1:n]
```

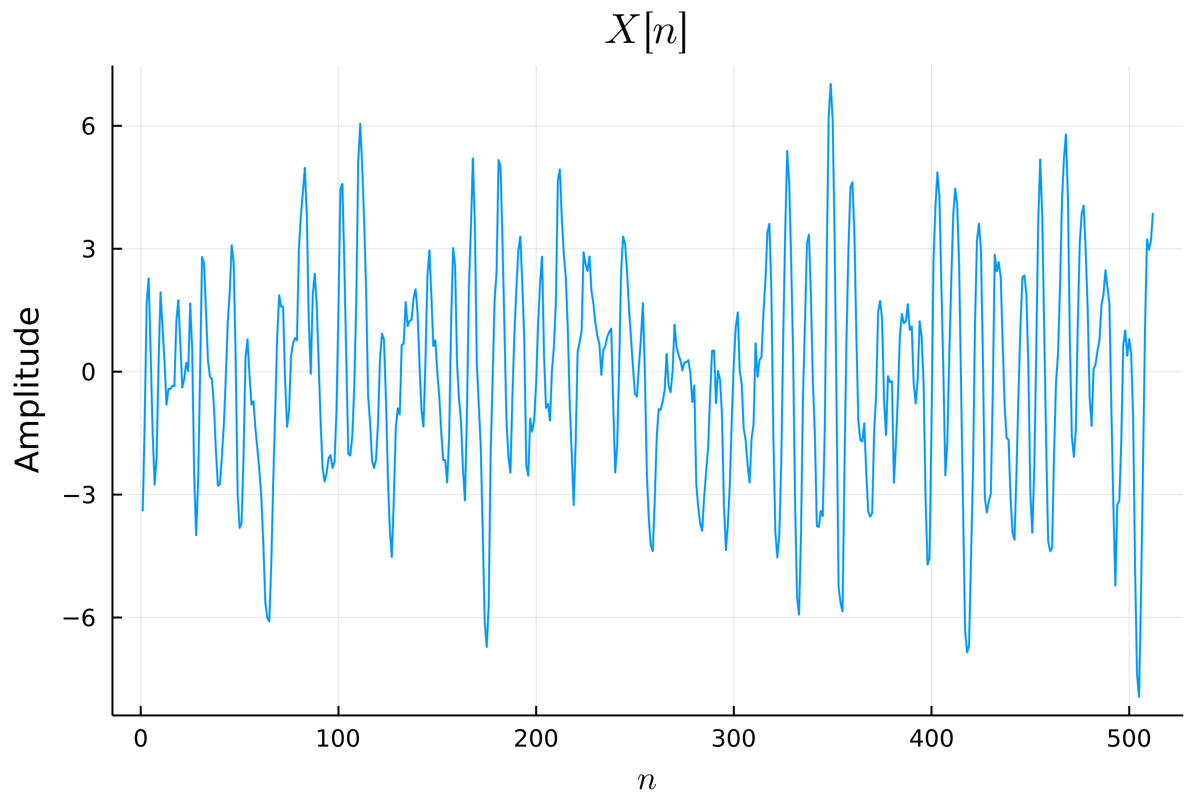
```
Xₙ (generic function with 1 method)
```

```julia
function Xₙ(Nn)
    n = length(Nn); Xn = zeros(n);
    Xn[1] = Nn[1]; Xn[2] = 1.5Nn[1]+Nn[2];

    for k ∈ 3:n  Xn[k] = 1.5Xn[k-1] - 0.8Xn[k-2] + Nn[k]  end

    return Xn
end
```

```julia
Nn = rand.(Nₙ(600));
```

```julia
Xn = Xₙ(Nn)[89:end]; # using only the last 512 samples
```





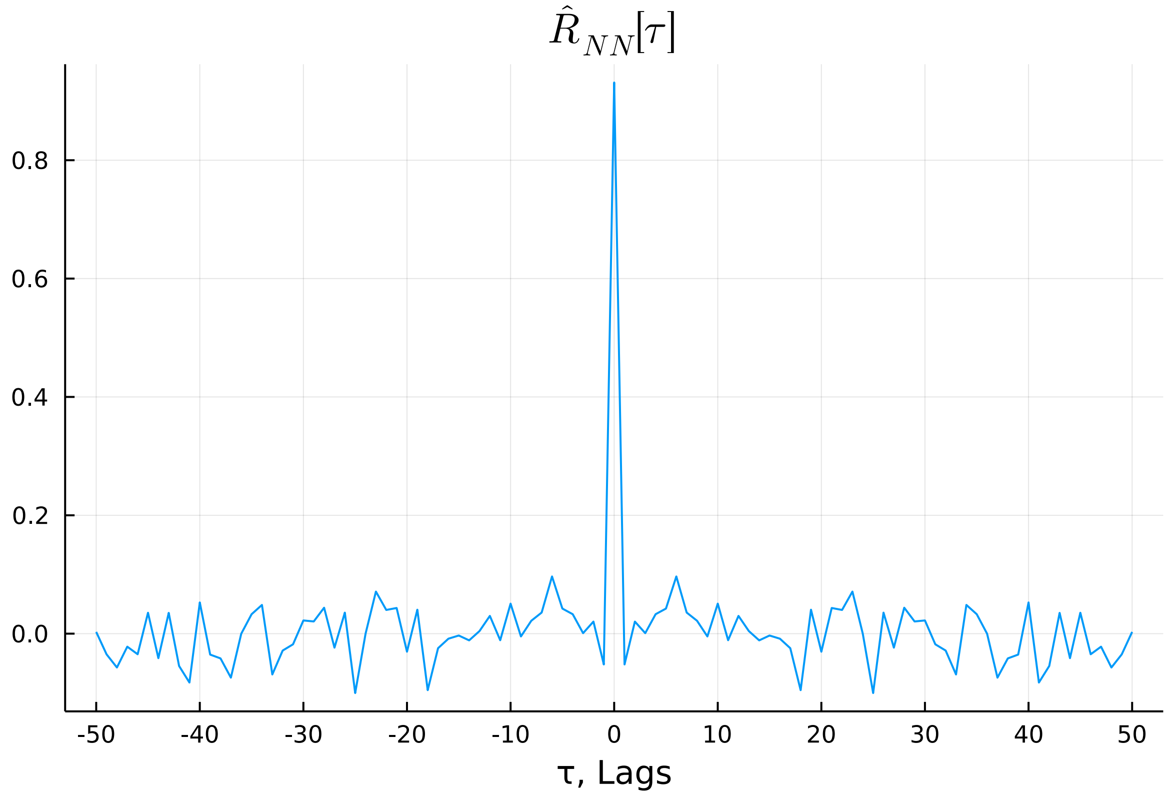
Note above that $N[n]$ looks like expected, uncorrelated white Gaussian noise, mostly between 0 to 1. A realization above of the process $X[n]$ is highly amplified with respect to the input signal, largely because of accumulation and scaling factors. It also varies less rapidly than the input $N[n]$.

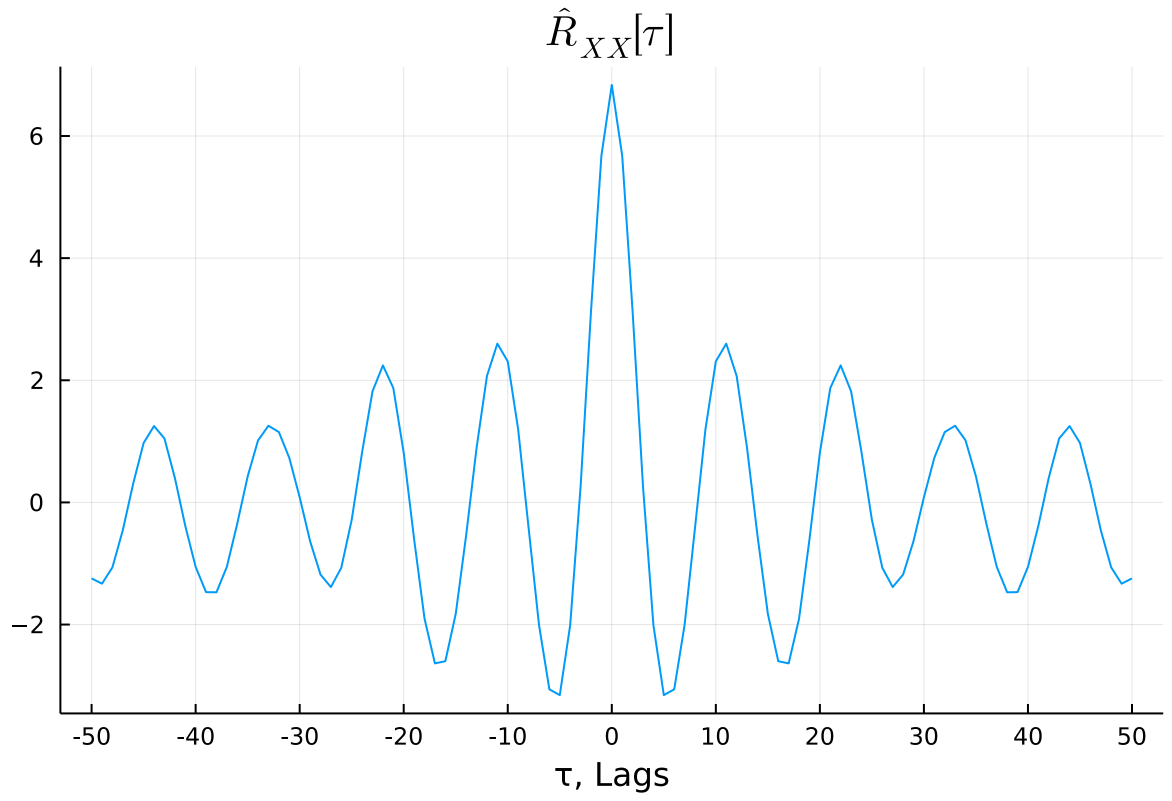
We now compute the sample autocorrelation functions of both processes below. Note that we are presuming these processes to be Ergodic in nature and therefore taking sample autocorrelation as close estimates $\hat{R}[\tau]$ of the true autocorrelation function $R[\tau]$. The validity of this assumption will be tested at each step.

```
autocorr (generic function with 1 method)
```

```
function autocorr(X; nlags=nothing) # assumes ergodicity of X
    n = length(X); L = (nlags === nothing) ? n-1 :
        ((0 < nlags < n) ? nlags : throw(AssertionError("0 ≮ nlags ≮ n-1")));
    Rₓ = zeros(L+1);                # initial length = number of lags

    for m ∈ 1:L+1
        for k ∈ 1:n-m+1
            Rₓ[m] += X[k]*X[k+m-1]; # add product terms
        end
        Rₓ[m] /= n-m+1;             # normalize by number of terms
    end

    Rₓ = [reverse(Rₓ[2:end]); Rₓ];  # all lags [-L to L]
end
```

```
Rₙₙ = autocorr(Nn; nlags=50); Rₓₓ = autocorr(Xn; nlags=50);
```


$\hat{R}_{NN}[\tau]$
0.8
0.6
0.4
0.2
0.0
-50
-40
-30
-20
-10
0
10
20
30
40
50
τ, Lags


$\hat{R}_{XX}[\tau]$
6
4
2
0
−2
-50
-40
-30
-20
-10
0
10
20
30
40
50
τ, Lags

From above we can see that the autocorrelation function $\hat{R}_{NN}[\tau]$ shows that $N[n]$ is uncorrelated as expected since at 0 lag it is close to 1 and it sharply drops to 0 for other values of lag, τ. On the other hand, from the plot for $\hat{R}_{XX}[\tau]$ we can see that its adjacent samples are highly correlated and no longer i.i.d. (as the input) due to the AR difference relation.

The average power for the signals is given by the respective autocorrelation functions evaluated at $\tau = 0$, where we are assuming (and also know) them to be WSS. The values are shown below.

```
0.931297807105049
```

```
Rₙₙ[51]
```

```
6.835668116760194
```

```
Rₓₓ[51]
```

We now estimate the power spectral density (PSD) $S_{XX}(f)$ by averaging over 10 periodograms of independent realizations of the process $X[n]$.

```
Nns = [rand.(Nₙ(600)) for i ∈ 1:10];
```

```
257×1 Matrix{Float64}:
  9.204530583500212
 11.46437455992002
  7.659182825394973
 12.776992102003781
 20.83034722983878
 17.496774530044476
 24.547677081179764
  ⋮
  0.2703730545005268
  0.14214744637863536
  0.21360254057983377
  0.3138489067984082
  0.13286225445249042
  0.26679278364495346
```

```
begin
    Xns   = [Xₙ(Nns[i])[89:end] for i ∈ 1:10];
    Ssₓₓ  = [periodogram(Xns[i]) for i ∈ 1:10];
    Pavgₓ = mean(hcat((Ssₓₓ.|> power)...); dims=2);
end
```

```
freqₓ = Ssₓₓ[1].freq;
```

Here, we take a step back and consider a general AR filter given by,

$$X[n] = -\sum_{p=1}^{P} a_p X[n-p] + N[n]$$

From the above input-output difference equation formulation of the AR filter, we have the filter's Z-transform given by, $H(z) = \frac{1}{A(z)}$, where,

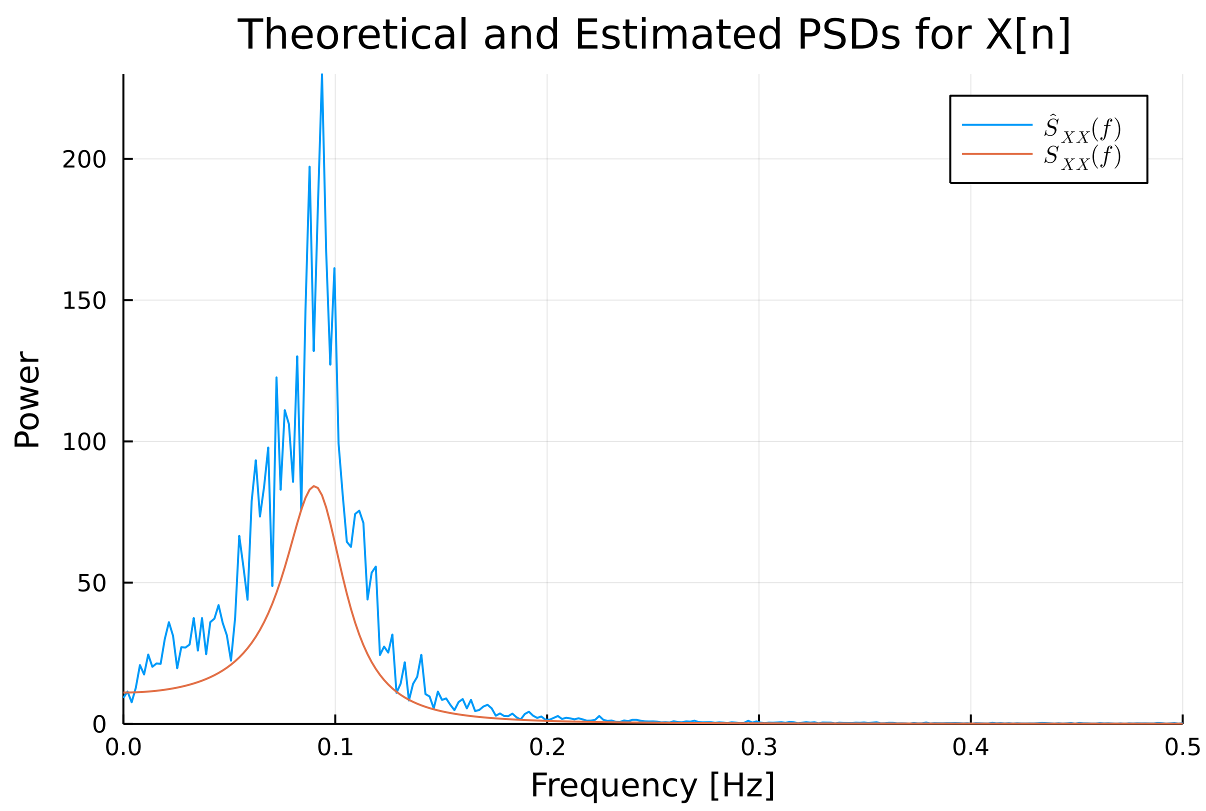
$$A(z) = 1 + \sum_{p=1}^{P} a_p z^{-p}$$

Thus, the output spectral density is given by,

$$S_{XX}(f) = \sigma_N^2 \frac{1}{|A(f)|^2}$$

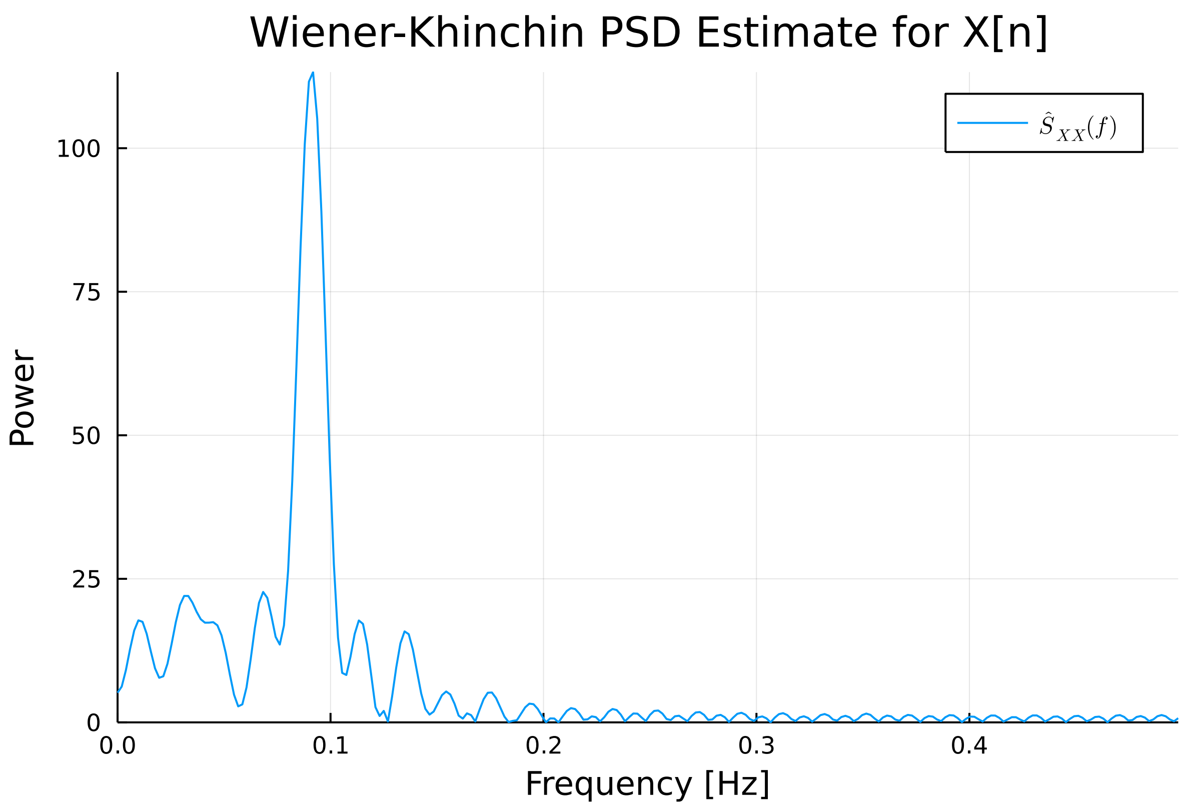
In our case, we have $H(z) = \frac{1}{1-1.5z^{-1}+0.8z^{-2}}$ and we get its magnitude response using the `freqresp` fucntion in the **DSP.jl** package. This is the theoretical PSD for the AR process and we plot it against the estimate calculated above.

```
H, ω = freqresp(PolynomialRatio([1.0], [1.0, -1.5, 0.8]));
```



There is a difference in the amplitudes between the estimate and true values, but they follow the same trend and once normalized they should match very closely. We also know by the Wiener-Khinchin theorem, that we can get the PSD of the WSS random process from the Fourier transform of its autocorrelation function. We calulate and plot it below.

```
Ŝₓₓ = fft([Rₓₓ; zeros(512-length(Rₓₓ))]) |> fftshift; # 512-pt fft
```



We can observe that the above estimate more closely resembles the true spectral density in both shape (less abrupt jumps) and magnitude than the periodogram. It is also worthy to note that all estimates still have a peak just before 0.1 Hz and thereafter decay to zero quickly.

Now, to estimate the AR filter coefficients, we can use the Yule-Walker equations, for p-lags.

$$\underbrace{\begin{bmatrix} R_{XX}[1] \\ R_{XX}[2] \\ \vdots \\ R_{XX}[p] \end{bmatrix}}_{\mathbf{r}_X} = \underbrace{\begin{bmatrix} R_{XX}[0] & R_{XX}[-1] & \dots & R_{XX}[1-p] \\ R_{XX}[1] & R_{XX}[0] & \dots & R_{XX}[2-p] \\ \vdots & \vdots & \vdots & \vdots \\ R_{XX}[p-1] & R_{XX}[p-2] & \dots & R_{XX}[0] \end{bmatrix}}_{R_X} \underbrace{\begin{bmatrix} h[1] \\ h[2] \\ \vdots \\ h[p] \end{bmatrix}}_{\mathbf{h}}$$

Thus, we can get an estimate of the filter coefficients $\mathbf{h}$ by,

$$\mathbf{h} = R_X^{-1}\mathbf{r}_X$$

Also, note again that $R_{XX}[k] = R_{XX}[-k]$ since autocorrelation for a WSS process is an even function of one variable.

```
rₓ (generic function with 1 method)
```
```
rₓ(p) = Rₓₓ[52:52+p-1]
```

```
Rₓ (generic function with 1 method)
```
```
Rₓ(p) = hcat([Rₓₓ[51-k:51+p-1-k] for k ∈ 0:p-1]...)
```

The 3-tap, 4-tap, and 5-tap filters for the AR process are estimated below.

```
h₃ =  [1.42418, -0.694826, -0.0365923]
```
```
h₃ = Rₓ(3)\rₓ(3)
```

```
h₄ =  [1.42424, -0.693664, -0.0389734, 0.00167189]
```
```
h₄ = Rₓ(4)\rₓ(4)
```

```
h₅ =  [1.42425, -0.693704, -0.0396852, 0.00313352, -0.00102625]
```
```
h₅ = Rₓ(5)\rₓ(5)
```

Note above that the first two coefficients of the filter closely match the AR coefficients of $[1.5 \quad 0.8]$ after which all components are very close to zero. This makes sense as the true filter only has two components and therefore rest of the components in the estimated filters are redundant.

2. Optimal Linear Filters for Autoregressive Processes

Here we consider another system excited by same i.i.d. white Guassian noise process $N[n]$ as before, and it produces a random sequence $S[n]$ at the output which is given by the following AR difference relation,

$$S[n] = 0.2S[n-1] - 0.8S[n-2] + N[n]$$

We then simulate the observation of this signal and introduce some measurement i.i.d. noise $W[n] \sim \mathcal{N}(0,1)$ which is independent to $N[n]$. The observed sequence is thus given by the relation,

$$Y[n] = S[n] + W[n]$$

Both processes are simulated in code below.

```
Sₙ (generic function with 1 method)
```

```
function Sₙ(Nn)
    n = length(Nn); Sn = zeros(n);
    Sn[1] = Nn[1]; Sn[2] = 0.2Nn[1]+Nn[2];

    for k ∈ 3:n  Sn[k] = 0.2Sn[k-1] - 0.8Sn[k-2] + Nn[k]  end

    return Sn
end
```

```
Sn = Sₙ(Nn)[89:end]; # using only the last 512 samples
```

```
Wn = rand.(Nₙ(length(Sn)));
```

```
Yn = Sn + Wn;
```

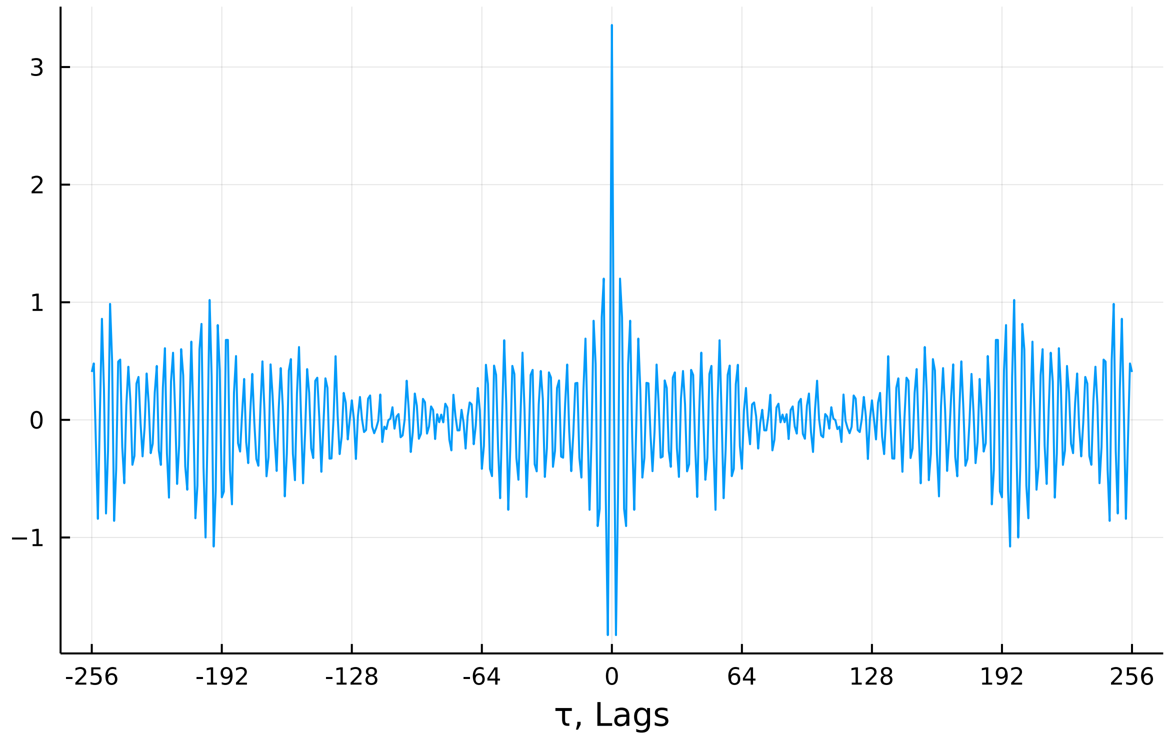
We now calculate the sample cross-correlation function of $Y[n]$ to estimate its autocorrelation $\hat{R}_{YY}[\tau]$ without replicating values for negative lags. This has been done to properly test for the evenness of the true autocorrelation $R_{YY}[\tau]$.

```
crosscorr (generic function with 1 method)
```

```
function crosscorr(X, Y; nlags=nothing) # assumes joint ergodicity of X and Y
    nₓ = length(X); nᵧ = length(Y); n = min(nₓ, nᵧ);
    L = (nlags === nothing) ? n-1 :
        ((0 < nlags < n) ? nlags : throw(AssertionError("0 ≮ nlags ≱ min(nₓ,
nᵧ)-1")));
    Rₓᵧ = zeros(L+1);                    # crosscorr for +ve lags; length = number of
lags
    X̃ = reverse(X); Ỹ = reverse(Y);   # flipped signals for negative lags
    R̃ₓᵧ = zeros(L+1);                    # crosscorr for -ve lags; length = number of
lags

    for m ∈ 1:L+1
        for k ∈ 1:n-m+1
            Rₓᵧ[m] += X[k]*Y[k+m-1];   # add product terms
            R̃ₓᵧ[m] += X̃[k]*Ỹ[k+m-1];
        end
        Rₓᵧ[m] /= n-m+1;                # normalize by number of terms
        R̃ₓᵧ[m] /= n-m+1;
    end

    Rₓᵧ = [reverse(R̃ₓᵧ[2:end]); Rₓᵧ]; # all lags [-L to L]
end
```

```
Rᵧᵧ = crosscorr(Yn, Yn; nlags=256);
```

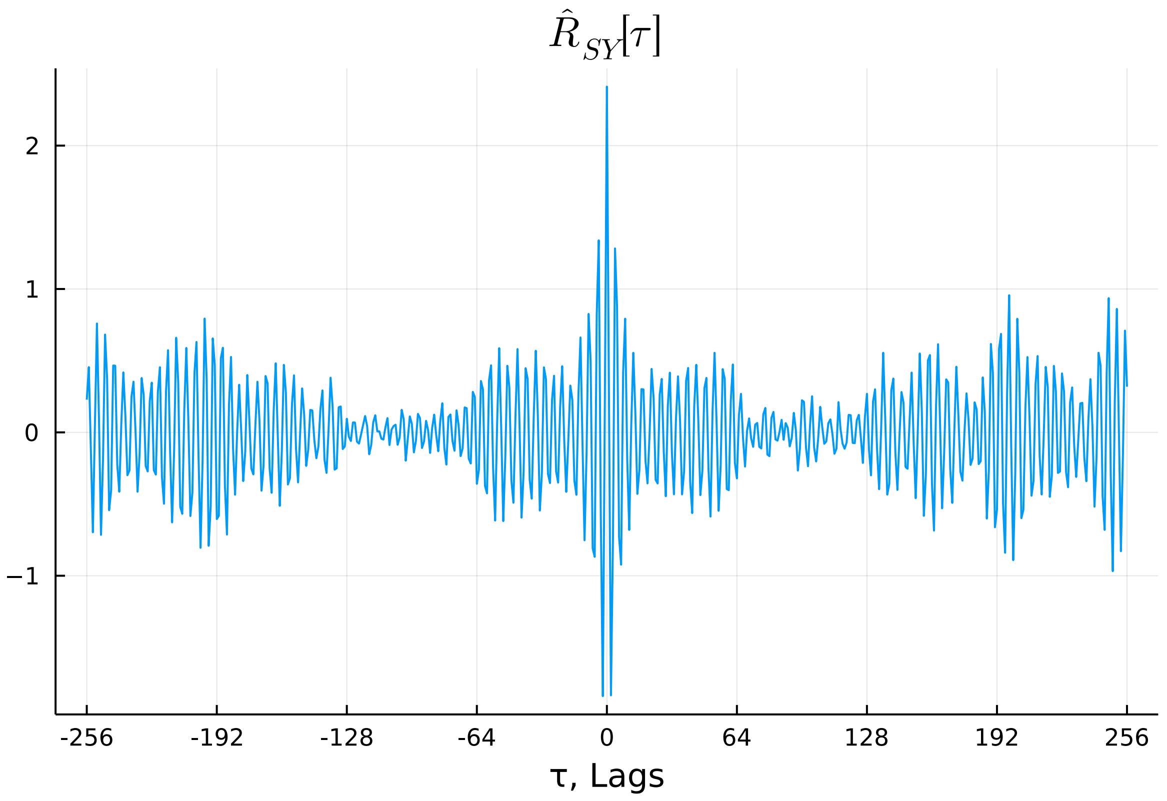
$\hat{R}_{YY}[\tau]$



We can visually observe that $\hat{R}_{YY}[\tau]$ is an even function, even though on some runs it may have a slight non-symmetry on a few positive and negative lags, but overall this trend hints at the WSS nature of the process. Therefore, as $Y[n]$ is a real-valued signal, we can expect the evenness of the autocorrelation process $R_{YY}[\tau]$.

We now estimate the cross-correlation of the processes $Y[n]$ and $S[n]$.

```
Rₛᵧ = crosscorr(Sn, Yn; nlags=256);
```



From the plot above, we can see that $R_{SY}[\tau]$ is not a perfectly even function (notice points around $\tau =\approx 0, \pm 64, \pm 192, \ldots$); however, it is nearly symmetrical as it shows similar trends for positive and negative lags, and furthermore, it also resembles $\hat{R}_{YY}[\tau]$. Note that, in general, the cross-correlation function will not be perfectly symmetric (even) unless the processes are jointly WSS. Here we have $Y[n]$ corrupted version of $S[n]$ due to (independent) noise, and we can (numerically) assume the joint WSS nature of the two signals as the plot also suggests. This will allow us to derive optimal linear filters to estimate the underlying signal $S[n]$ from the measured sequence $Y[n]$.

Following the assumption above, we would now like to design a 7-tap optimal filter as a linear estimator for S_n which is given by,

$$Z[n] = \sum_{k=0}^{6} h[k]Y[n-k] = (Y \star h)[n]$$

where, in general, a p-tap filter impulse response $h[n]$ satisfies the Wiener-Hopf equations, which in the discrete-FIR case are *similar* to the Yule-Walker equations as (causal) FIR prediction coefficients act as estimated AR parameters!

$$\underbrace{\begin{bmatrix} R_{SY}[0] \\ R_{SY}[1] \\ \vdots \\ R_{SY}[p-1] \end{bmatrix}}_{\mathbf{r}_{SY}} = \underbrace{\begin{bmatrix} R_{YY}[0] & R_{YY}[-1] & \ldots & R_{YY}[1-p] \\ R_{YY}[1] & R_{YY}[0] & \ldots & R_{YY}[2-p] \\ \vdots & \vdots & \vdots & \vdots \\ R_{YY}[p-1] & R_{YY}[p-2] & \ldots & R_{YY}[0] \end{bmatrix}}_{R_Y} \underbrace{\begin{bmatrix} h[0] \\ h[1] \\ \vdots \\ h[p-1] \end{bmatrix}}_{\mathbf{h}}$$

Thus, we get $\mathbf{h} = R_Y^{-1}\mathbf{r}_{SY}$.

The theoretical minimum mean-sqaured error for the above optimal filter is given by,

$$E[e_n^2] = R_{SS}[0] - \sum_{k=0}^{6} h[k]R_{SY}[k]$$

This should ideally be close to the empirical variance of the process $(S_n - Z_n)$, both being zero mean processes.

Note above that $R_{SS}[0]$ is the average power that is also given by,

$$R_{SS}[0] = P_S = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} S[n] \approx \frac{1}{N_S} \sum_{n=-\frac{N_S}{2}}^{\frac{N_S}{2}} S[n]$$

In the code below, we first calculate the optimal linear estimator.

```
rₛᵧ (generic function with 1 method)
```

```
rₛᵧ(p) = Rₛᵧ[257:257+p-1]
```

```
Rᵧ (generic function with 1 method)
```

```
Rᵧ(p) = hcat([Rᵧᵧ[257-k:257+p-1-k] for k ∈ 0:p-1]...)
```

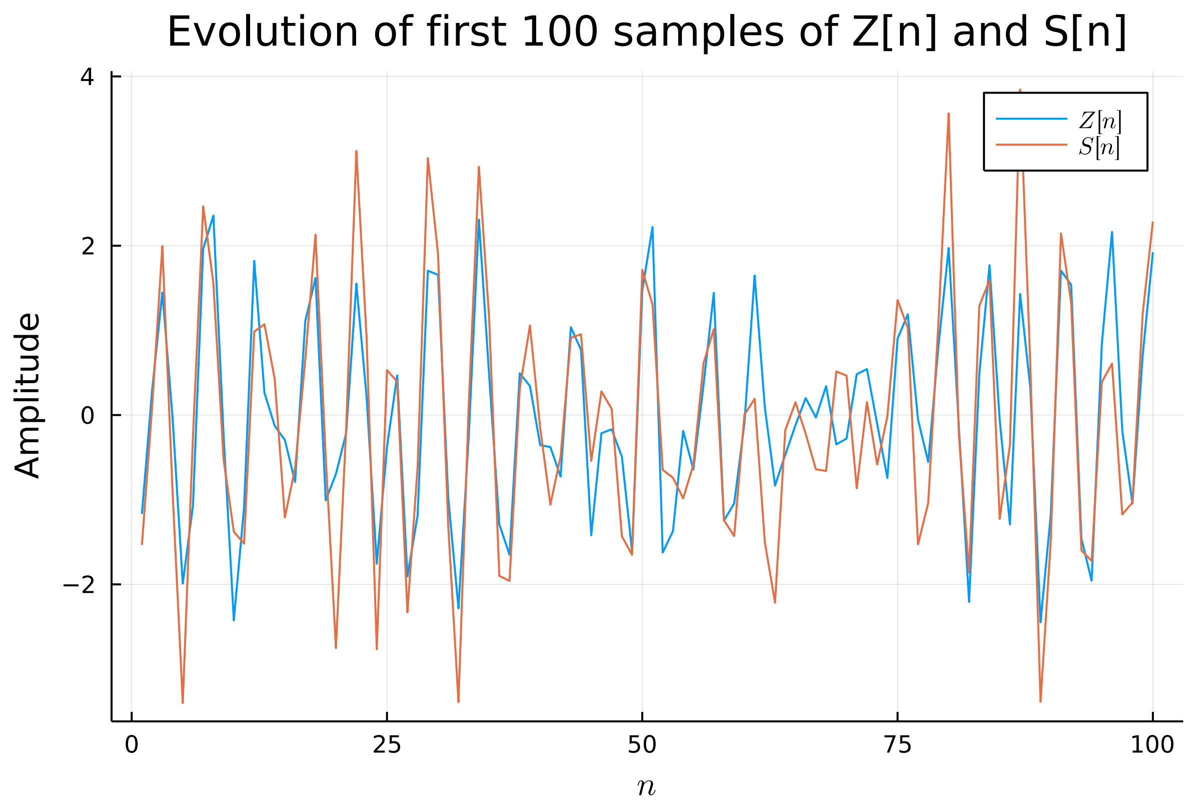
```
h₇ =  [0.56464, 0.0406149, -0.18309, -0.0659483, 0.0903068, 0.0228309, -0.000816895]
```

```
h₇ = Rᵧ(7)\rₛᵧ(7)
```

Now, we test the filter for how close it is to the theoretical mean-squared error (MSE) and also plot the first 100 samples of both processes to demonstrate the working of the optimal filter.

```
Zn =
 [-1.16727, 0.294076, 1.44801, -0.0262205, -1.99125, -1.06954, 1.96401, 2.35554, -0.241
```

```
Zn = conv(h₇, Yn)
```



```
0.5621096383132504
```

```
var(Sn-Zn[1:length(Sn)])
```

```
Pₛ = 2.438373820620594
```

```
Pₛ = (Sn' * Sn)/length(Sn)
```

```
mseₜₕ = 0.5442733015905683
```

```
mseₜₕ = Pₛ - sum(h₇ .* Rₛᵧ[257:263])
```

And indeed, as expected by the Wiener-Hopf equations, the optimal linear estimator is close to the minimum MSE.

3. Code

Note that this lab report can be run on the cloud and viewed as is on the github repository page **here**. All code for the notebook can be accessed **here**.