# 1 Problem (a) - Plot all the realization PSDs on top of each other

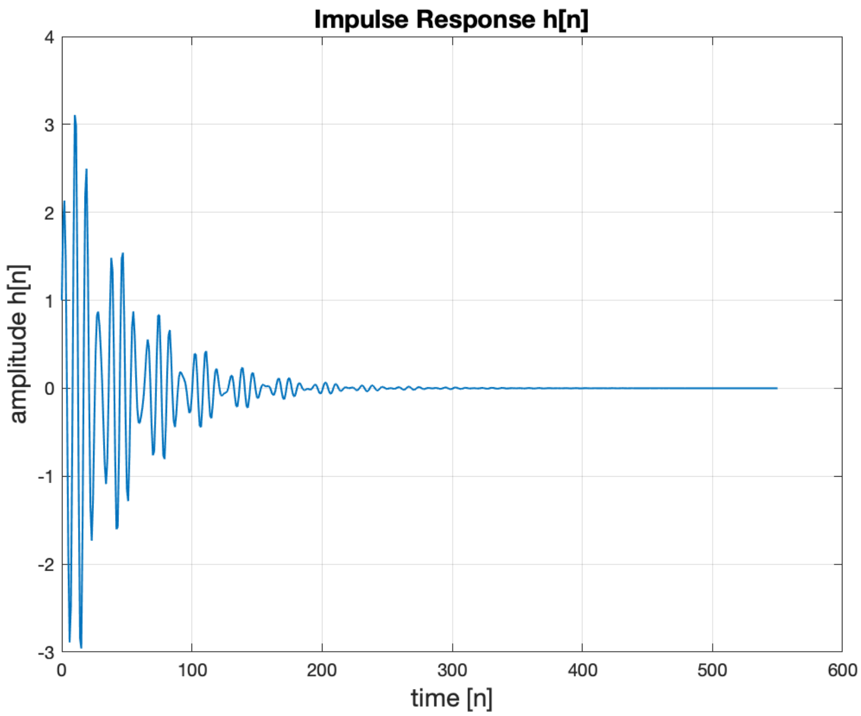



Figure 1: Impulse Response h[n] corresponding to given H(z) computed using MATLAB impz function

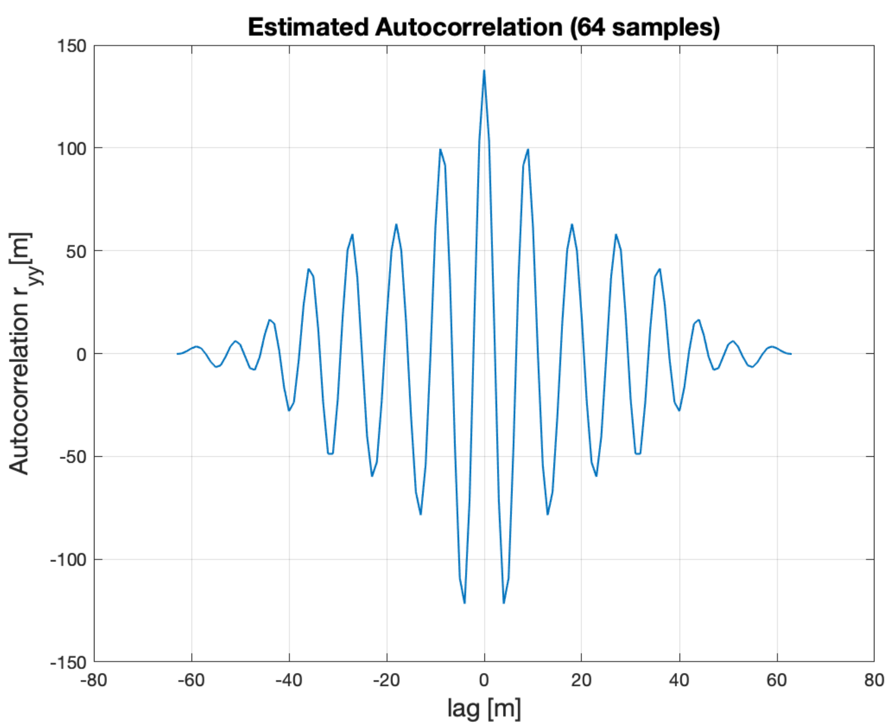



Figure 2: Estimated autocorrelation sequence $r_{yy}[m]$ of the output random process $y[n]$

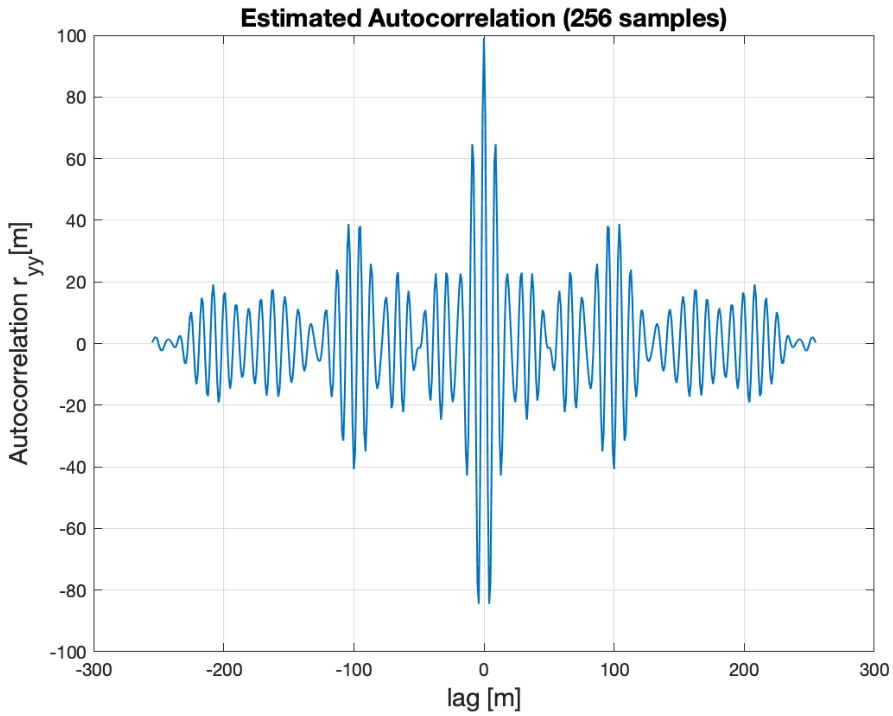



Figure 3: Estimated autocorrelation sequence $r_{yy}[m]$ of the output random process $y[n]$

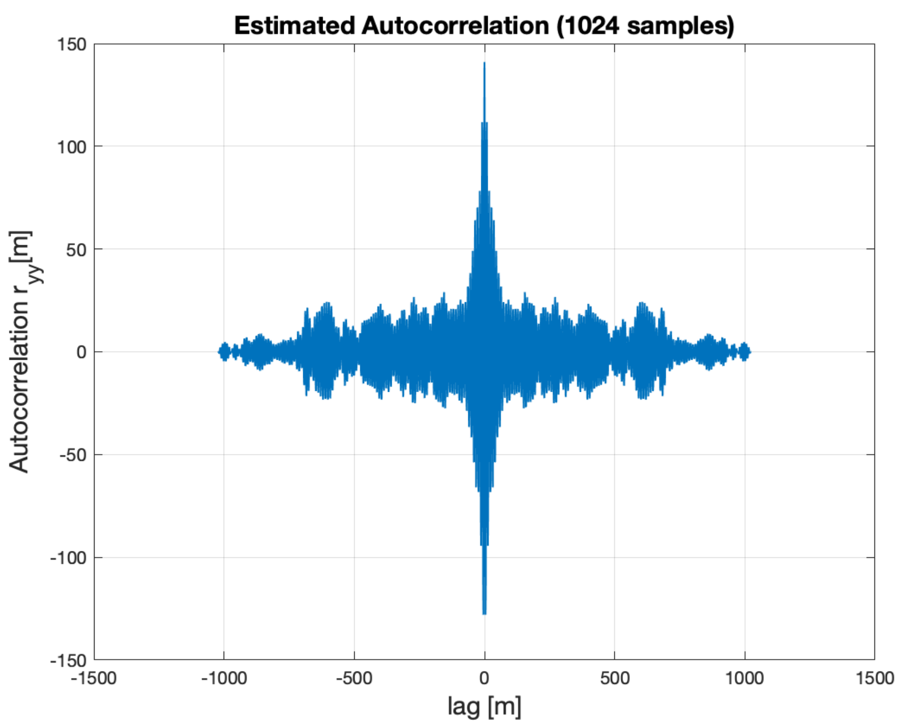



Figure 4: Estimated autocorrelation sequence $r_{yy}[m]$ of the output random process $y[n]$

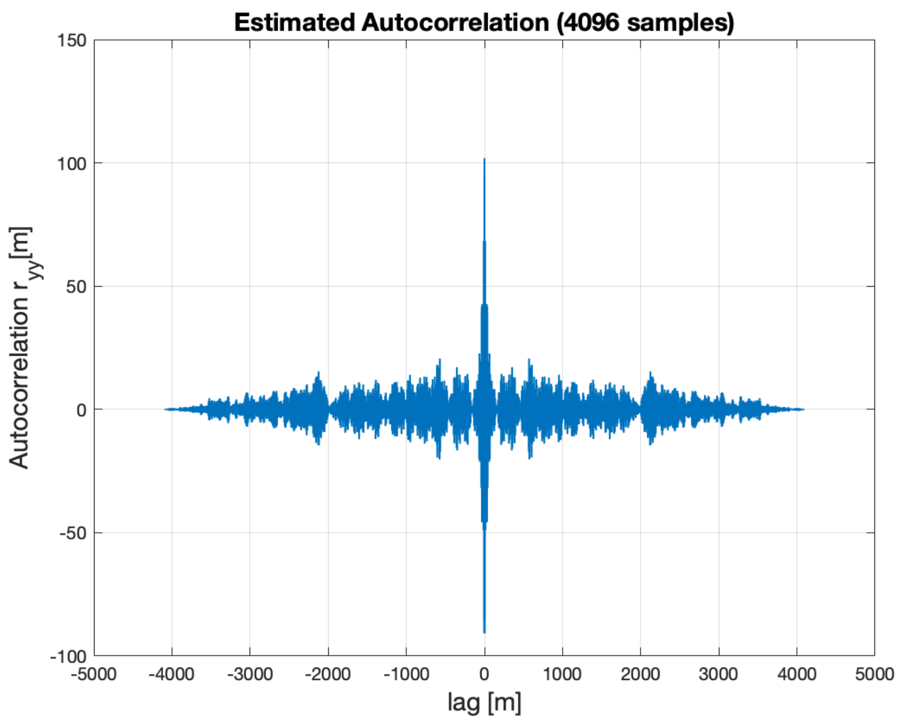



Figure 5: Estimated autocorrelation sequence $r_{yy}[m]$ of the output random process $y[n]$

## 1.1 Case - Realization length = 64, Filter order = 2

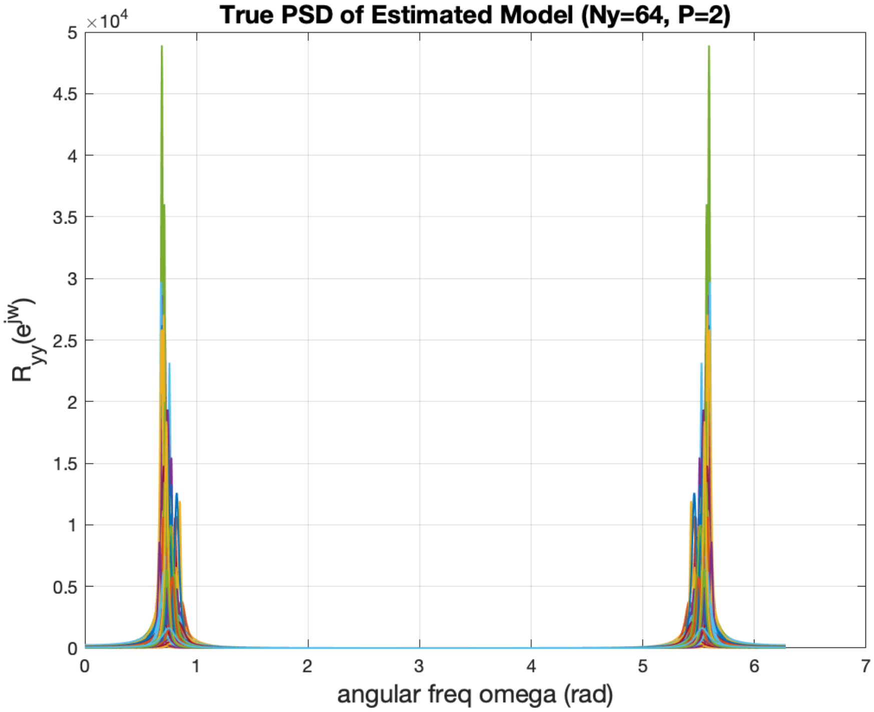



Figure 6: True PSD of the estimated models for 1000 realizations (realization length 64 and filter order 2)

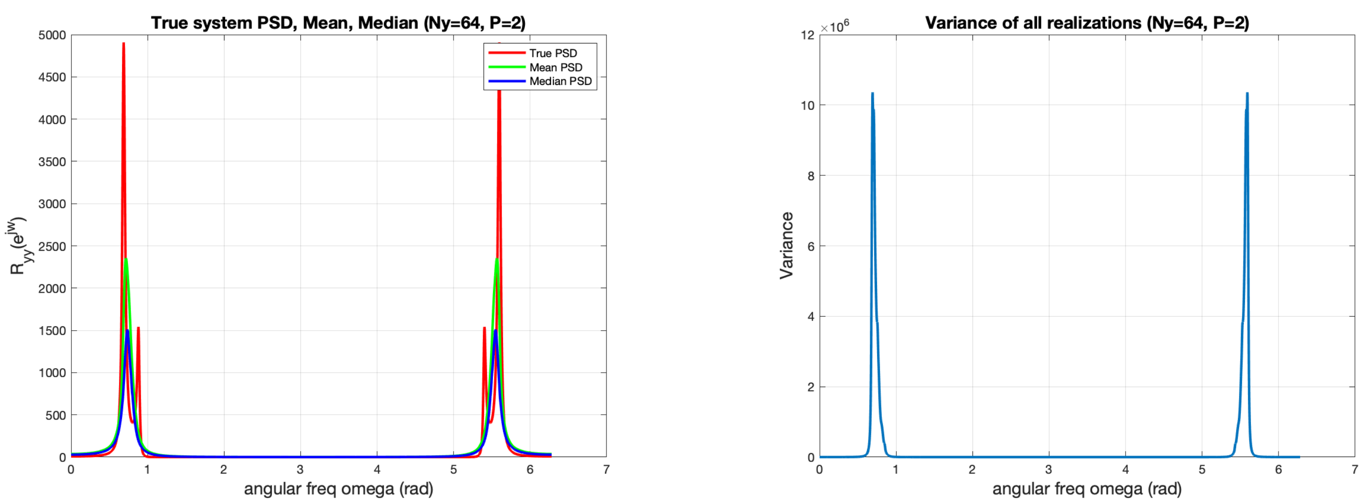



Figure 7: (Left) True system PSD, pointwise mean of all realization PSDs, and pointwise median of all realization PSDs for 1000 realizations. (Right) Variance of the realization PSDs for 1000 realizations (realization length 64 and filter order 2)

## 1.2 Case - Realization length = 64, Filter order = 8

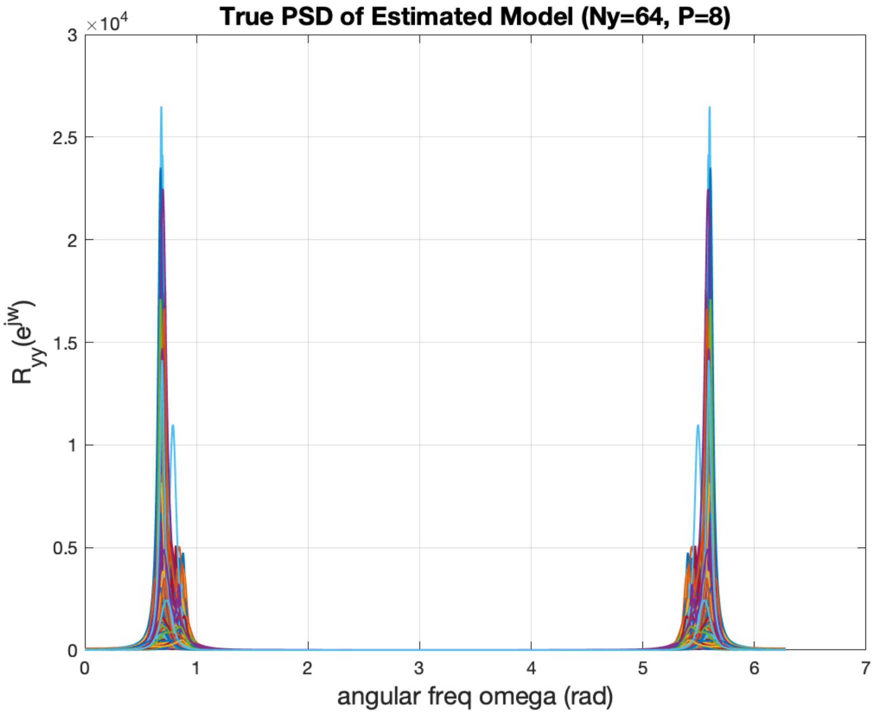



Figure 8: True PSD of the estimated models for 1000 realizations (realization length 64 and filter order 8)

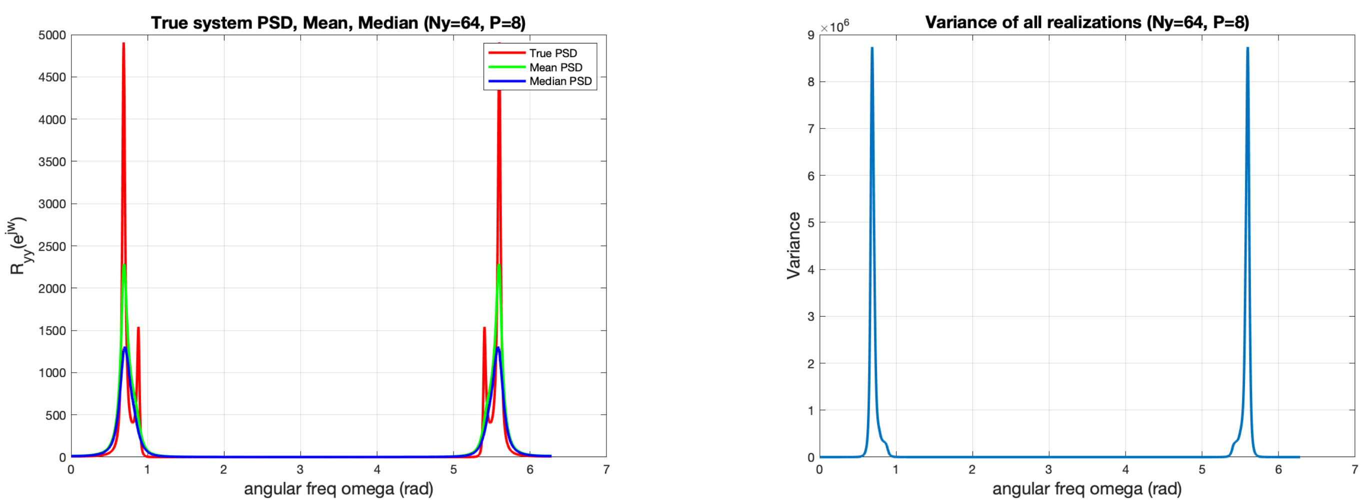



Figure 9: (Left) True system PSD, pointwise mean of all realization PSDs, and pointwise median of all realization PSDs for 1000 realizations. (Right) Variance of the realization PSDs for 1000 realizations (realization length 64 and filter order 8)

### 1.3 Case - Realization length = 64, Filter order = 14

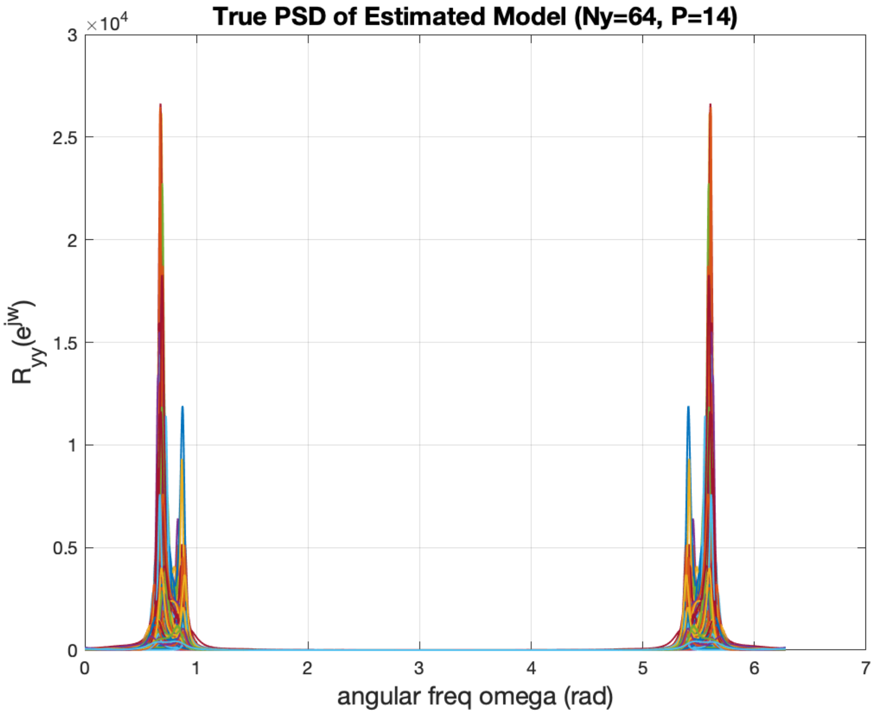



Figure 10: True PSD of the estimated models for 1000 realizations (realization length 64 and filter order 14)

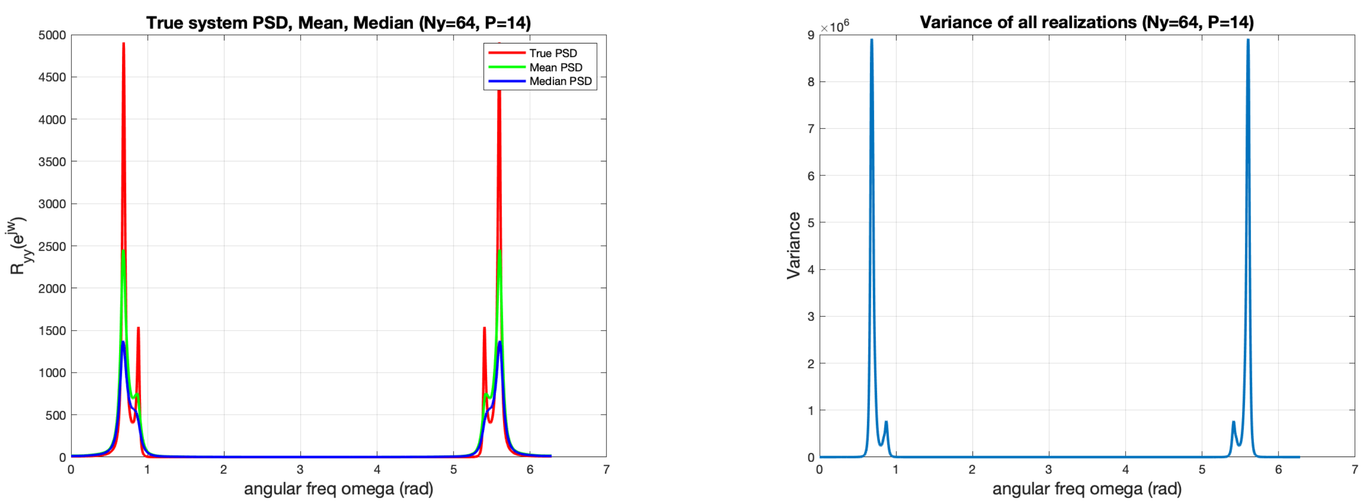



Figure 11: (Left) True system PSD, pointwise mean of all realization PSDs, and pointwise median of all realization PSDs for 1000 realizations. (Right) Variance of the realization PSDs for 1000 realizations (realization length 64 and filter order 14)

### 1.4 Case - Realization length = 256, Filter order = 2

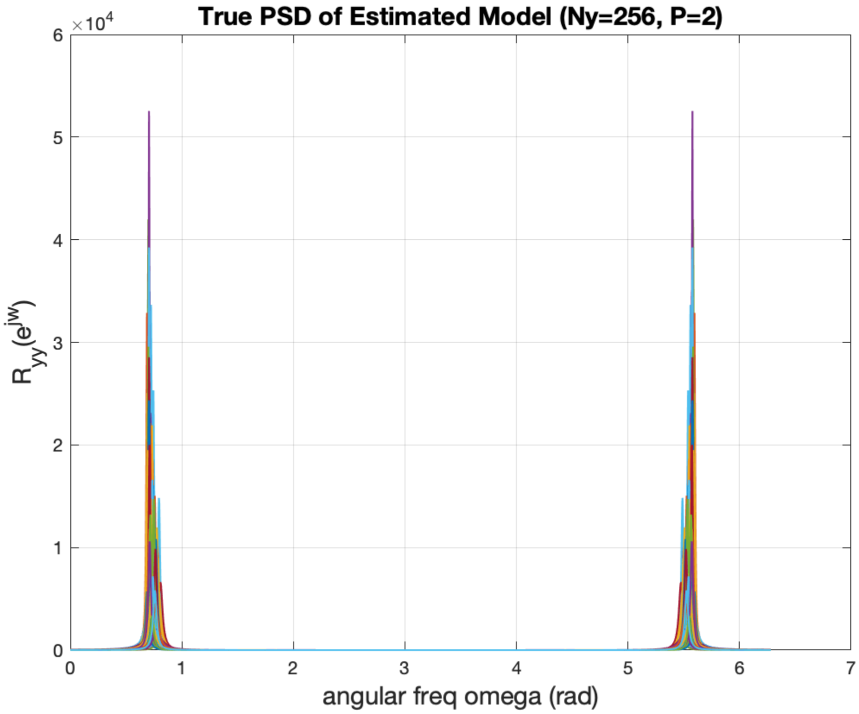
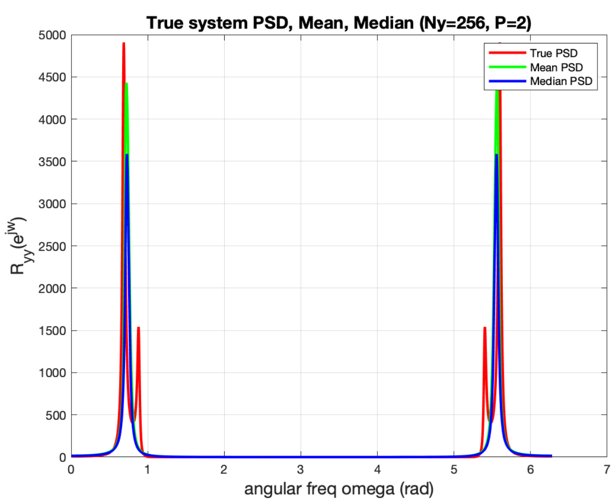



Figure 12: True PSD of the estimated models for 1000 realizations (realization length 256 and filter order 2)



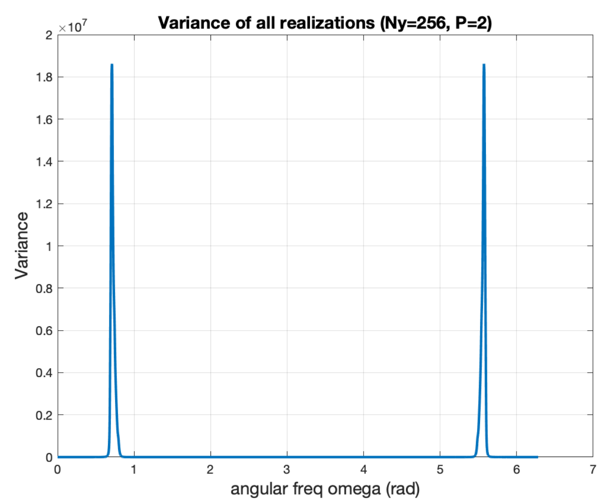




Figure 13: (Left) True system PSD, pointwise mean of all realization PSDs, and pointwise median of all realization PSDs for 1000 realizations. (Right) Variance of the realization PSDs for 1000 realizations (realization length 256 and filter order 2)

## 1.5 Case - Realization length = 256, Filter order = 8

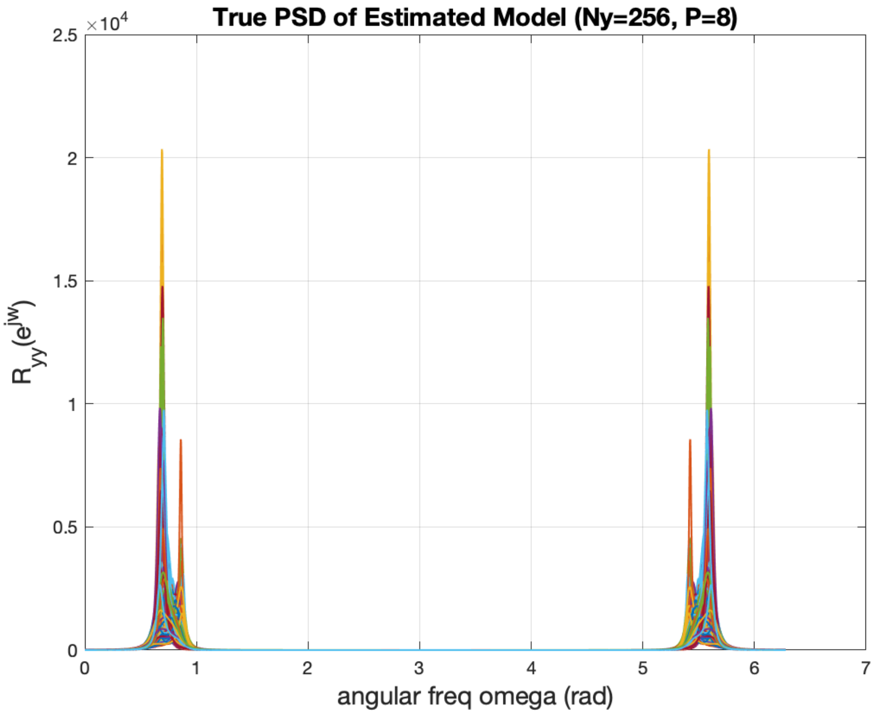



Figure 14: True PSD of the estimated models for 1000 realizations (realization length 256 and filter order 8)

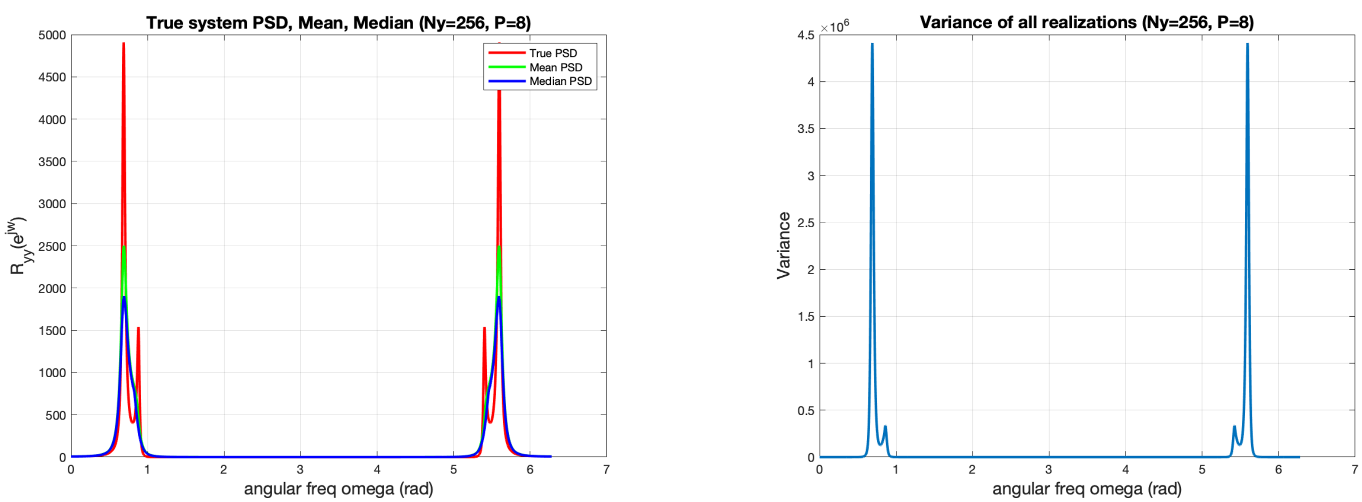



Figure 15: (Left) True system PSD, pointwise mean of all realization PSDs, and pointwise median of all realization PSDs for 1000 realizations. (Right) Variance of the realization PSDs for 1000 realizations (realization length 256 and filter order 8)

## 1.6 Case - Realization length = 256, Filter order = 14

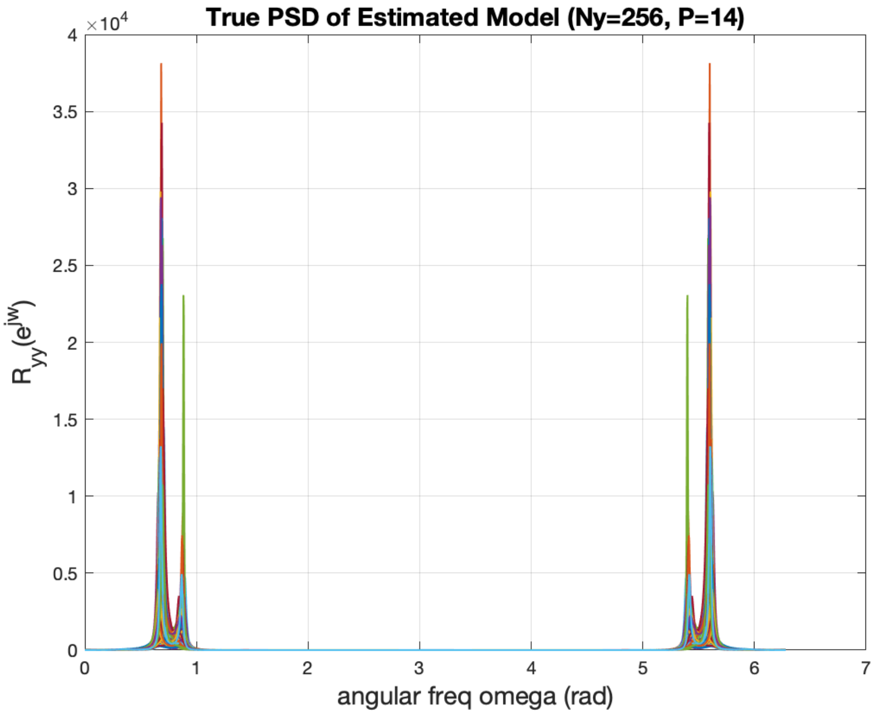
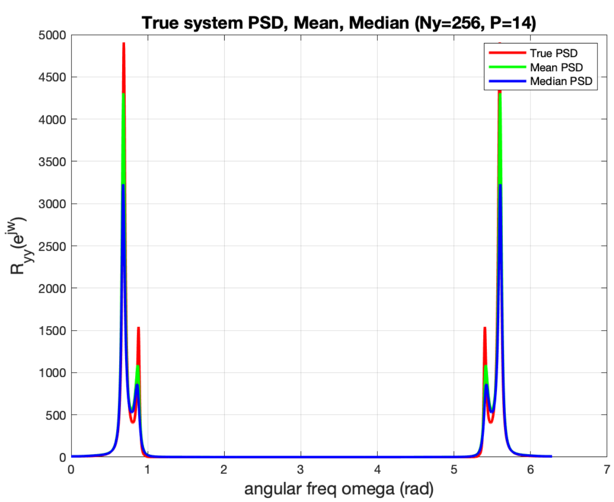



Figure 16: True PSD of the estimated models for 1000 realizations (realization length 256 and filter order 14)



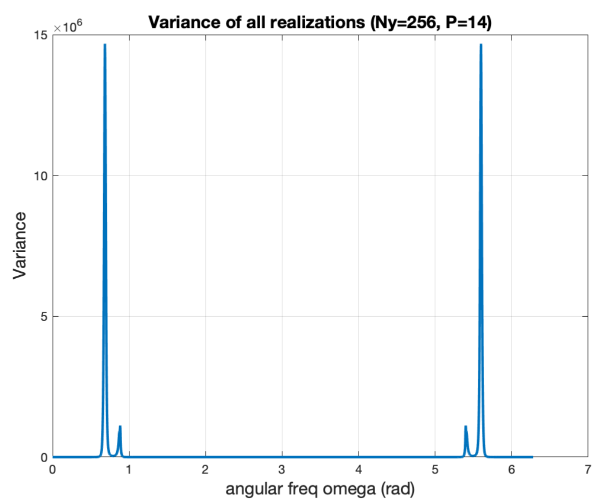




Figure 17: (Left) True system PSD, pointwise mean of all realization PSDs, and pointwise median of all realization PSDs for 1000 realizations. (Right) Variance of the realization PSDs for 1000 realizations (realization length 256 and filter order 14)

### 1.7 Case - Realization length = 1024, Filter order = 2

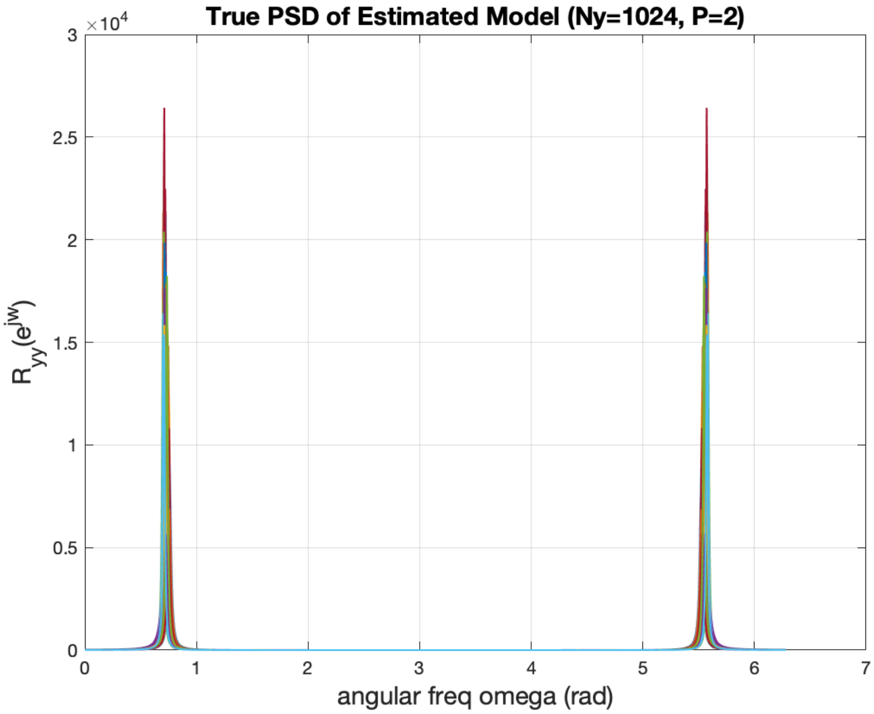
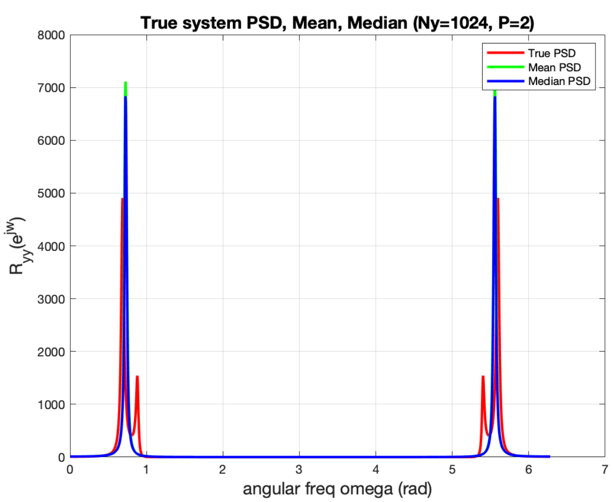



Figure 18: True PSD of the estimated models for 1000 realizations (realization length 1024 and filter order 2)



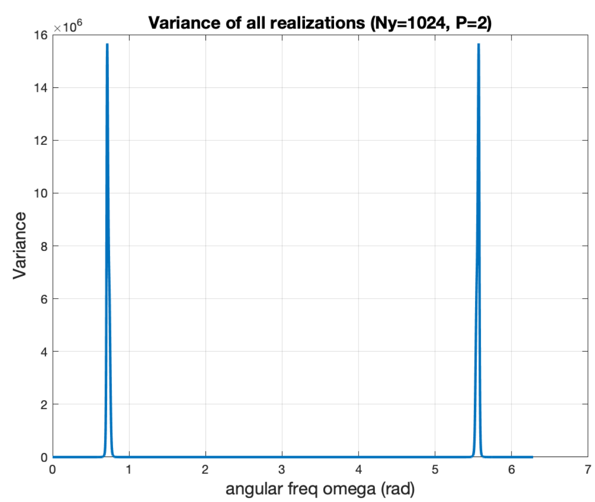




Figure 19: (Left) True system PSD, pointwise mean of all realization PSDs, and pointwise median of all realization PSDs for 1000 realizations. (Right) Variance of the realization PSDs for 1000 realizations (realization length 1024 and filter order 2)

### 1.8 Case - Realization length = 1024, Filter order = 8

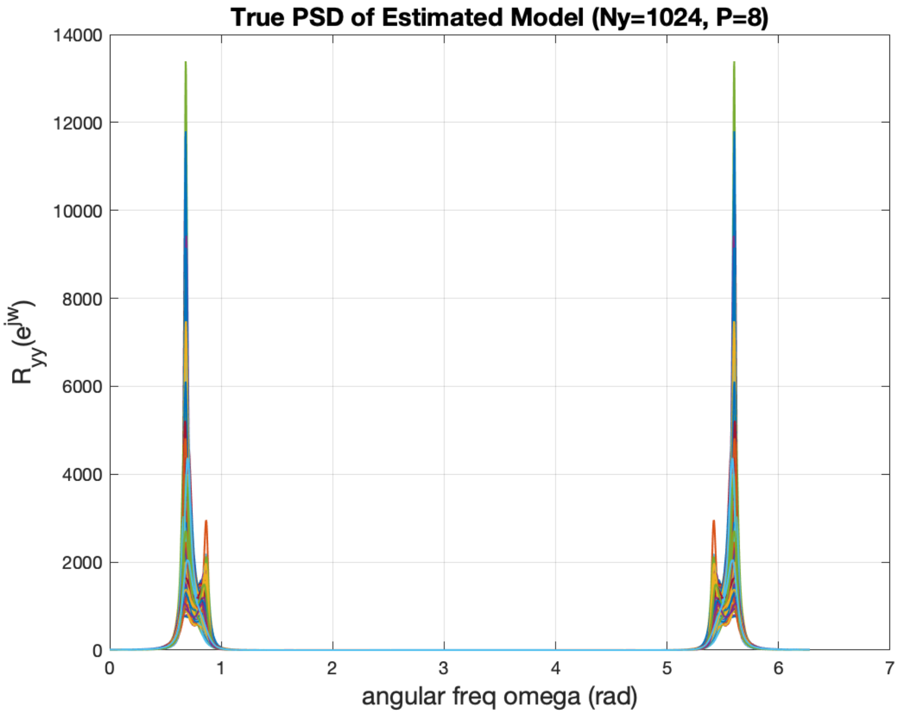



Figure 20: True PSD of the estimated models for 1000 realizations (realization length 1024 and filter order 8)

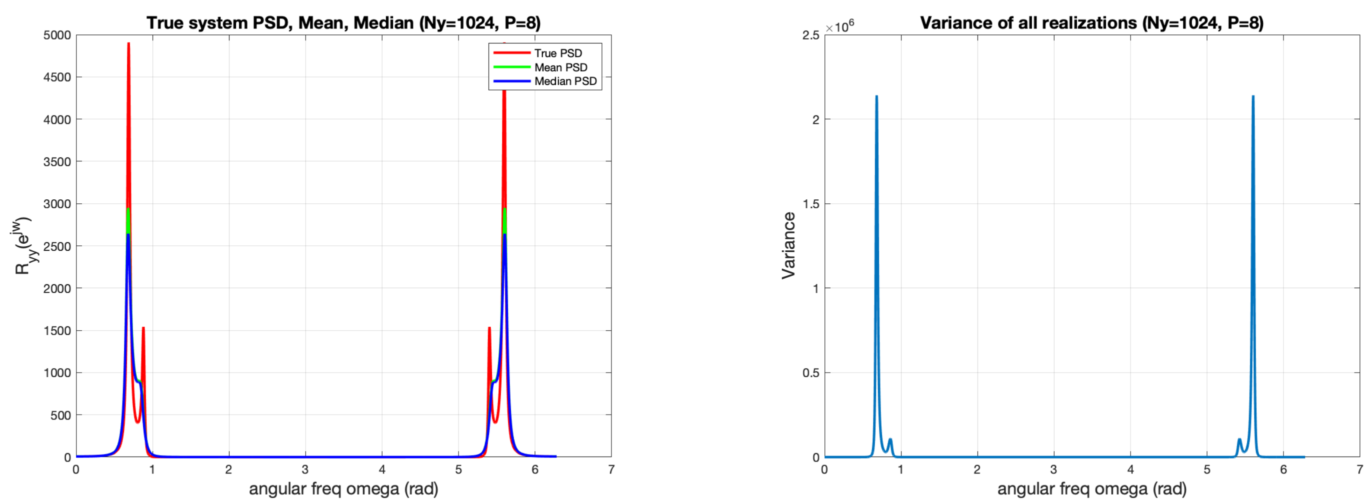



Figure 21: (Left) True system PSD, pointwise mean of all realization PSDs, and pointwise median of all realization PSDs for 1000 realizations. (Right) Variance of the realization PSDs for 1000 realizations (realization length 1024 and filter order 8)

## 1.9 Case - Realization length = 1024, Filter order = 14

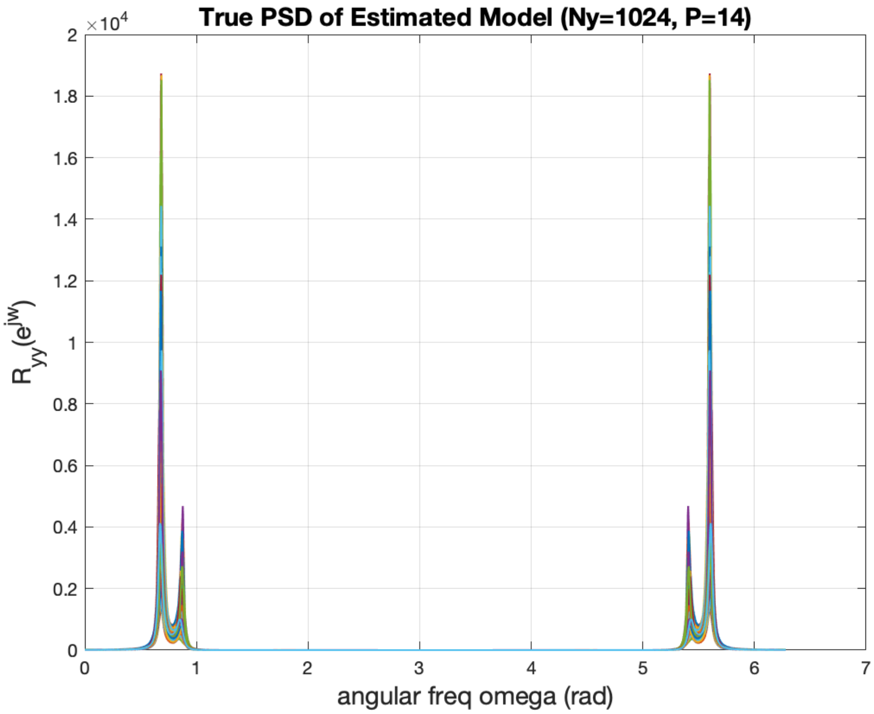
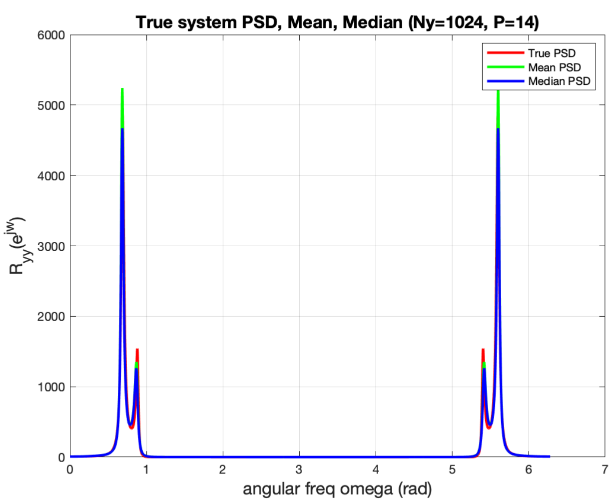



Figure 22: True PSD of the estimated models for 1000 realizations (realization length 1024 and filter order 14)



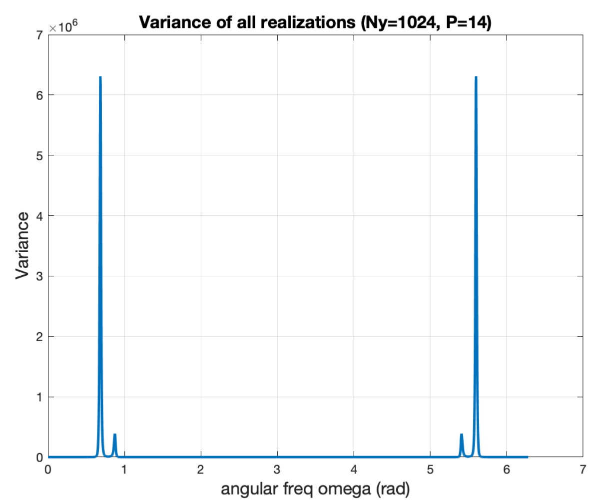




Figure 23: (Left) True system PSD, pointwise mean of all realization PSDs, and pointwise median of all realization PSDs for 1000 realizations. (Right) Variance of the realization PSDs for 1000 realizations (realization length 1024 and filter order 14)

## 1.10 Case - Realization length = 4096, Filter order = 2

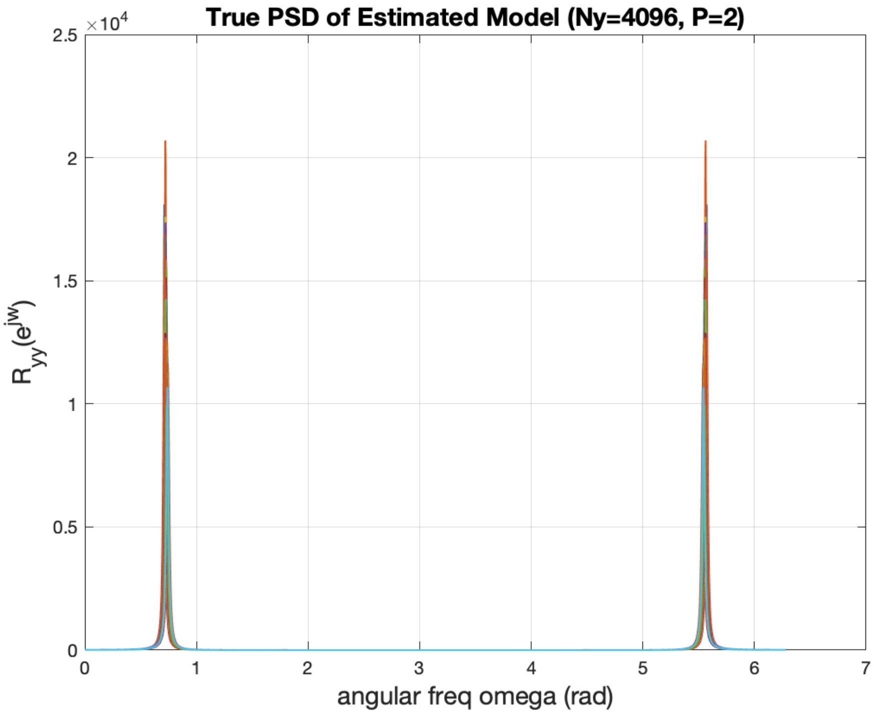



Figure 24: True PSD of the estimated models for 1000 realizations (realization length 4096 and filter order 2)

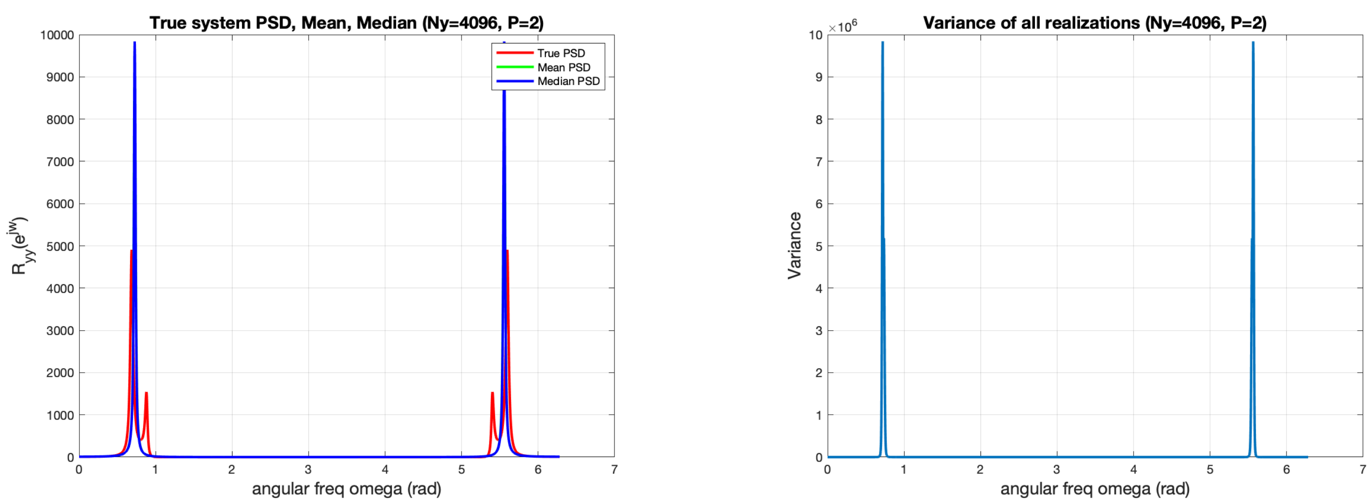



Figure 25: (Left) True system PSD, pointwise mean of all realization PSDs, and pointwise median of all realization PSDs for 1000 realizations. (Right) Variance of the realization PSDs for 1000 realizations (realization length 4096 and filter order 2)

## 1.11 Case - Realization length = 4096, Filter order = 8

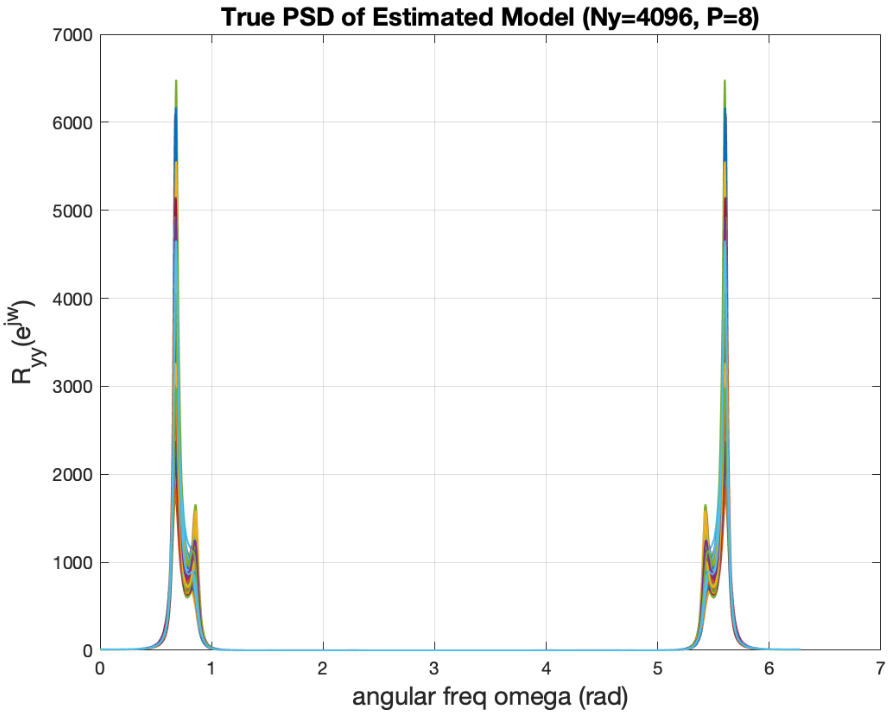



Figure 26: True PSD of the estimated models for 1000 realizations (realization length 4096 and filter order 8)

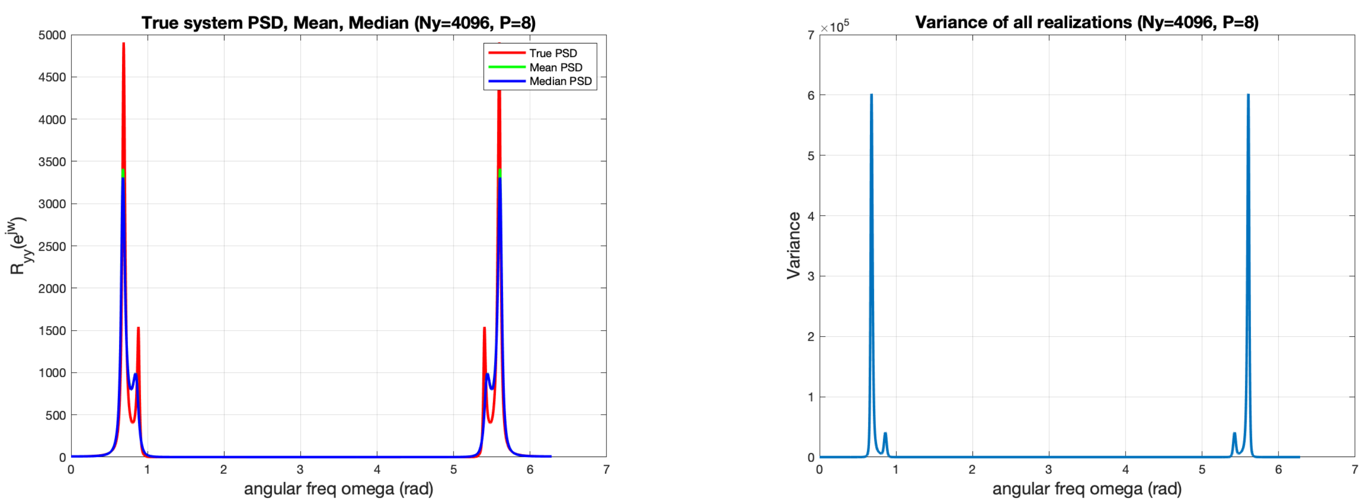



Figure 27: (Left) True system PSD, pointwise mean of all realization PSDs, and pointwise median of all realization PSDs for 1000 realizations. (Right) Variance of the realization PSDs for 1000 realizations (realization length 4096 and filter order 8)

## 1.12 Case - Realization length = 4096, Filter order = 14

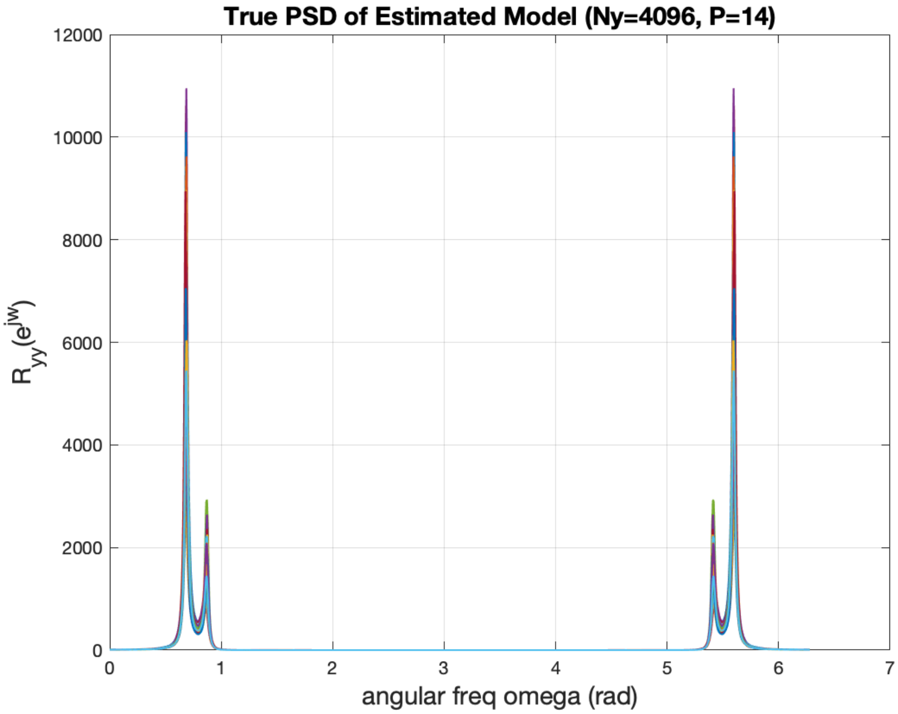



Figure 28: True PSD of the estimated models for 1000 realizations (realization length 4096 and filter order 14)

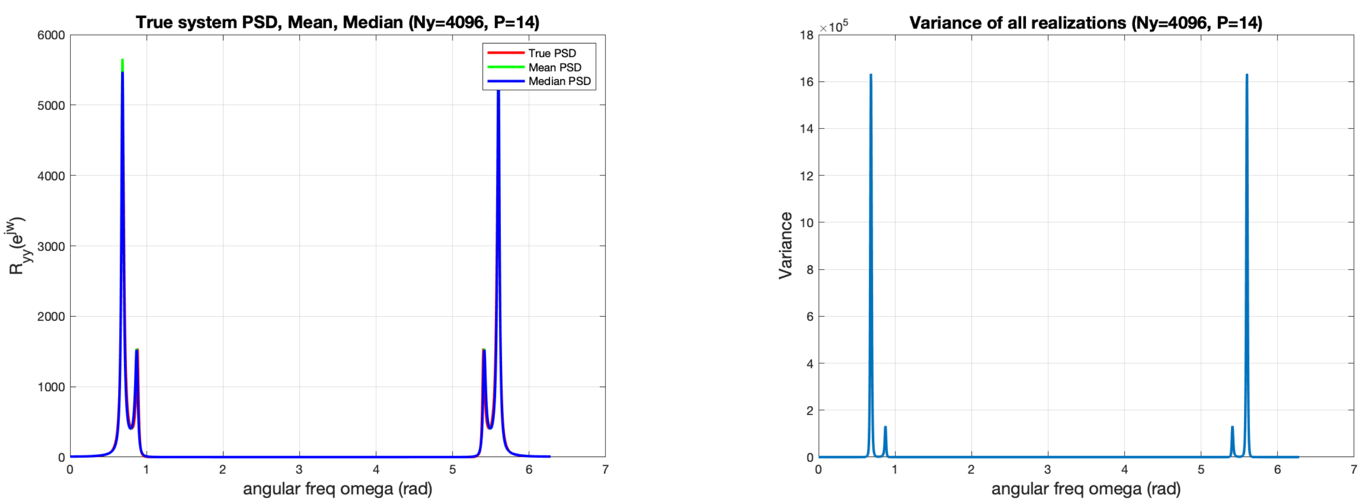



Figure 29: (Left) True system PSD, pointwise mean of all realization PSDs, and pointwise median of all realization PSDs for 1000 realizations. (Right) Variance of the realization PSDs for 1000 realizations (realization length 4096 and filter order 14)

# 2 Comments on the observations

- For lower values of N (number of realization samples) and P (filter order), the pointwise mean and median of all realization PSDs do not match the true system PSD. Since we have very few samples of the random process and only a few lag values in the autocorrelation matrix, the AR model is represented by a small number of parameters and is not able to approximate the true ARMA model. Therefore, the mean and median PSDs of all realizations are not able to match the true system PSD for small N and P.

- As we increase N (64 to 4096), the pointwise mean and median of all realization PSDs are better able to approximate the true system PSD.

- For a fixed N, increasing P (from 2 to 14) increases the accuracy of the estimated AR model system. This can be observed from the mean, median and true system PSD plot for any fixed N. Thus, keeping N fixed and increasing P results in realizations whose mean and median PSDs are closer to the true PSD, and therefore, the individual realization PSDs are better able to approximate true system PSD.

- For a fixed P, increasing N (from 64 to 4096) also results in realizations that better approximate the actual system PSD.

- Overall, more the number of realization samples N, better the autocorrelation estimate and the autocorrelation matrix, and hence better AR model approximation of the true ARMA system.

- More the filter order P, the more would be the representing power of the AR model approximation. Thus, out of all cases, the best approximation of the given ARMA system is observed for N = 4096 and P = 14.

- Mean of all realizations has higher peak (in most cases) as compared to median of all realizations

- The variance plots are very consistent with two major peaks around $\omega = 0.75$ and $\omega = 5.6$ rad along with a small side-peak in some cases. The minor side peaks are observed only for P = 8 and P = 14. This might be because, for small values of P, the PSDs generated by the estimated AR model does not have much variation near major peaks. Therefore, the side peaks are not observed.

# 3 MATLAB Code

## 3.1 main.m

```
clc; clear;

input_mu = 0;
input_var = 1;

% compute the impulse function h[n] from the transfer function H(z)
b = [1, -0.9, 0.81];     % numerator coeffs
a = [1, -2.76, 3.809, -2.654, 0.924];    % denominator coeffs

[hn, tn] = impz(b, a);
Nh = size(hn, 1);

% plot the impulse response
fig = figure;
plot(tn, hn, LineWidth=1);
xlabel("time [n]", FontSize=14);
ylabel("amplitude h[n]", FontSize=14);
title("Impulse Response h[n]", FontSize=14);
grid on;
saveas(fig, "../plots/impulse_response_impz.png");
close;

num_eval_pts = 2048;

% compute the true system PSD for the given H(z) = D(z) / A(z)
n = 0:(num_eval_pts - 1);
omega = 2 * pi * n / num_eval_pts;
Dz = 1 - 0.9 * exp(-1j * omega) + 0.81 * exp(-2j * omega);
Az = 1 - 2.76 * exp(-1j * omega) + 3.809 * exp(-2j * omega) - 2.654 * exp(-3j * omega) + 0.924

Hz = Dz ./ Az;
true_sys_psd = input_var * abs(Hz).^2;

num_realizations = 1000;
Ny = 8193;

% estimate autocorrelation sequence of y[n] for various sample lengths
num_samp_lst = [64, 256, 1024, 4096];
filter_order_lst = [2, 8, 14];

for num_samp = num_samp_lst
    for P = filter_order_lst
        disp("Case - N="+num_samp+", P="+P);
        true_psd_est_model = zeros(num_realizations, num_eval_pts);

        fig1 = figure;
```

```
        for itr = 1:num_realizations
            % generate samples of white gaussian noise random process x[n] ~ N(0,1)
            % Using generated x[n], generate a realization of the process y[n] = h[n] *
            % x[n] of length Ny
            if itr == 1 && P == 2
                [xn, yn] = generate_random_process(hn, Ny, true);
            else
                [xn, yn] = generate_random_process(hn, Ny, false);
            end

            % sample y[n] from the center
            y_samp = yn((Ny+1)/2 - num_samp/2 : (Ny+1)/2 + num_samp/2 - 1, 1);
            assert(size(y_samp, 1) == num_samp);

            if itr == 1 && P == 2
                savepath = "../plots/est_autocorr_"+num_samp+".png";
                ryy_est = estimate_autocorr(y_samp, true, savepath);
            else
                ryy_est = estimate_autocorr(y_samp, false, "");
            end

            % create the autocorrelation toeplitz matrix
            autocorr_toeplitz_mat = compute_autocorr_toeplitz_mat(ryy_est, P);

            % 1st method
%             rhs_vec = zeros(P+1, 1);
%             rhs_vec(1, 1) = input_var;
%             ap_par_vec = autocorr_toeplitz_mat \ rhs_vec;
%             ap_par_vec = ap_par_vec / ap_par_vec(1,1);

            % 2nd method
            rhs_vec = -conj(ryy_est(num_samp+1:num_samp+P));
            ap_vec = autocorr_toeplitz_mat(2:P+1, 2:P+1) \ rhs_vec;
            sigma_sq = ryy_est(num_samp) + sum(ryy_est(num_samp+1:num_samp+P) .* ap_vec);
            ap_par_vec = zeros(P+1, 1);
            ap_par_vec(1,1) = 1;
            ap_par_vec(2:P+1, 1) = ap_vec;

            % compute true psd of this estimated model H(z) = 1 / A(z)
            Az_hat = compute_Az(ap_par_vec, P, num_eval_pts);
            Hz_hat = 1 ./ Az_hat;

            true_psd_est_model(itr, :) = sigma_sq * abs(Hz_hat).^2;

            plot((0 : num_eval_pts - 1) * 2 * pi / num_eval_pts, true_psd_est_model(itr, :), L
        end

        hold off;
        xlabel("angular freq omega (rad)", FontSize=14);
        ylabel("R_{yy}(e^{jw})", FontSize=14);
```

```
        title("True PSD of Estimated Model (Ny="+num_samp+", P="+P+")", FontSize=14);
        grid on;
        saveas(fig1, "../plots/true_psd_est_model_Ny"+num_samp+"_P"+P+".png");
        close;

        % part (b) and (c)
        mean_est_psd = mean(true_psd_est_model, 1);
        median_est_psd = median(true_psd_est_model, 1);
        var_est_psd = var(true_psd_est_model, 1);

        % plot the true system PSD, mean and median of all realizations for
        % each case
        fig = figure;
        freq_axis = (0 : (num_eval_pts - 1)) * 2 * pi / num_eval_pts;
        plot(freq_axis, true_sys_psd, LineWidth=2, Color="r"); hold on;
        plot(freq_axis, mean_est_psd, LineWidth=2, Color="g"); hold on;
        plot(freq_axis, median_est_psd, LineWidth=2, Color="b"); hold off;
        xlabel("angular freq omega (rad)", FontSize=14);
        ylabel("R_{yy}(e^{jw})", FontSize=14);
        title("True system PSD, Mean, Median (Ny="+num_samp+", P="+P+")", FontSize=14);
        legend('True PSD', 'Mean PSD', 'Median PSD');
        grid on;
        saveas(fig, "../plots/true_sys_psd_mean_med_Ny"+num_samp+"_P"+P+".png");
        close;

        % plot the variance of all realizations for each case
        fig = figure;
        freq_axis = (0 : (num_eval_pts - 1)) * 2 * pi / num_eval_pts;
        plot(freq_axis, var_est_psd, LineWidth=2);
        xlabel("angular freq omega (rad)", FontSize=14);
        ylabel("Variance", FontSize=14);
        title("Variance of all realizations (Ny="+num_samp+", P="+P+")", FontSize=14);
        grid on;
        saveas(fig, "../plots/var_psd_Ny"+num_samp+"_P"+P+".png");
        close;
    end
end

close all;
```

## 3.2 generate_random_process.m

```
function [xn, yn] = generate_random_process(hn, Ny, tosave)
    Nh = size(hn,1);
    Nx = Ny - Nh + 1;
    xn = randn(Nx, 1);

    if tosave == true
        fig = figure;
```

```
        plot(0:Nx-1, xn, LineWidth=1);
        xlabel("time index [n]", FontSize=14);
        ylabel("amplitude x[n]", FontSize=14);
        title("Gaussian White Noise Realization ~ N(0,1)", FontSize=14);
        grid on;
        saveas(fig, "../plots/gaussian_white_input_"+Ny+".png");
        close;
    end

    % estimate the random process y[n] by convolving x[n] and h[n]
    yn = conv(xn, hn);
    assert(size(yn,1) == Ny);

    if tosave == true
        fig = figure;
        plot(0:Ny-1, yn, LineWidth=1);
        xlabel("time index [n]", FontSize=14);
        ylabel("amplitude y[n]", FontSize=14);
        title("Output random process realization y[n]", FontSize=14);
        grid on;
        saveas(fig, "../plots/out_random_process_"+Ny+".png");
        close;
    end
end
```

### 3.3 estimate_autocorr.m

```
function [rxx1] = estimate_autocorr(xn, tosave, savepath)
    % estimate autocorrelation sequence using N samples
    N = size(xn, 1);
    rxx1 = zeros(2 * N - 1, 1);    % 2(N-1)+1

    for m = 0:N-1
        rxx1(N+m, 1) = sum(xn(1+m:N, 1) .* conj(xn(1:N-m, 1)));
    end
    rxx1(1:N-1, 1) = flip(rxx1(N+1:2*N-1, 1));
    rxx1 = rxx1 / N;

%     rxx2 = zeros(2 * N - 1, 1);    % 2(N-1)+1
%     for m = 0:N-1
%         rxx2(N+m,1) = sum(xn(1+m:N,1) .* conj(xn(1:N-m,1)));
%     end
%     for m = -(N-1):-1
%         rxx2(N+m,1) = sum(xn(1:N-abs(m), 1) .* conj(xn(abs(m)+1:N, 1)));
%     end
%     rxx2 = rxx2 / N;
%     disp(sum(abs(rxx1 - rxx2)));
```

```
    if tosave == true
        fig = figure;
        plot(-(N-1):(N-1), rxx1, LineWidth=1);
    %     plot(-(N-1):(N-1), rxx2, LineWidth=2); hold on;
        xlabel("lag [m]", FontSize=14);
        ylabel("Autocorrelation r_{yy}[m]", FontSize=14);
        title("Estimated Autocorrelation ("+N+" samples)", FontSize=14);
        grid on;
        saveas(fig, savepath);
        close;
    end
end
```

### 3.4 compute_Az.m

```
function [Az] = compute_Az(ap_par_vec, P, N)
    assert(size(ap_par_vec, 1) == P+1);
    Az = zeros(N, 1);
    l_vec = transpose(0:P);

    for n = 0:N-1
        omega = 2 * pi * n / N;
        tmp1 = conj(ap_par_vec);
        tmp2 = exp(-1j * omega .* l_vec);
%         assert(size(tmp1) == size(tmp2));
        Az(n+1, 1) = sum(tmp1 .* tmp2);
    end
end
```

### 3.5 compute_autocorr_toeplitz_mat.m

```
function [autocorr_mat] = compute_autocorr_toeplitz_mat(ryy, P)
    N = (size(ryy, 1) + 1) / 2;

    autocorr_mat = zeros(P+1, P+1);
    for i = 1:P+1
        autocorr_mat(i, :) = ryy(N - i + 1 : N - i + 1 + P);
    end

    first_row = ryy(N:N+P);
    autocorr_mat2 = toeplitz(first_row);

    assert(size(autocorr_mat,1) == size(autocorr_mat2,1));
    assert(size(autocorr_mat,2) == size(autocorr_mat2,2));
    assert( sum(abs(autocorr_mat2 - autocorr_mat), 'all') == 0 );
```

```
end
```