1 Problem-1 (a) - Estimate true autocorrelation $r_{yy}[m]$ of output random process $y[n]$

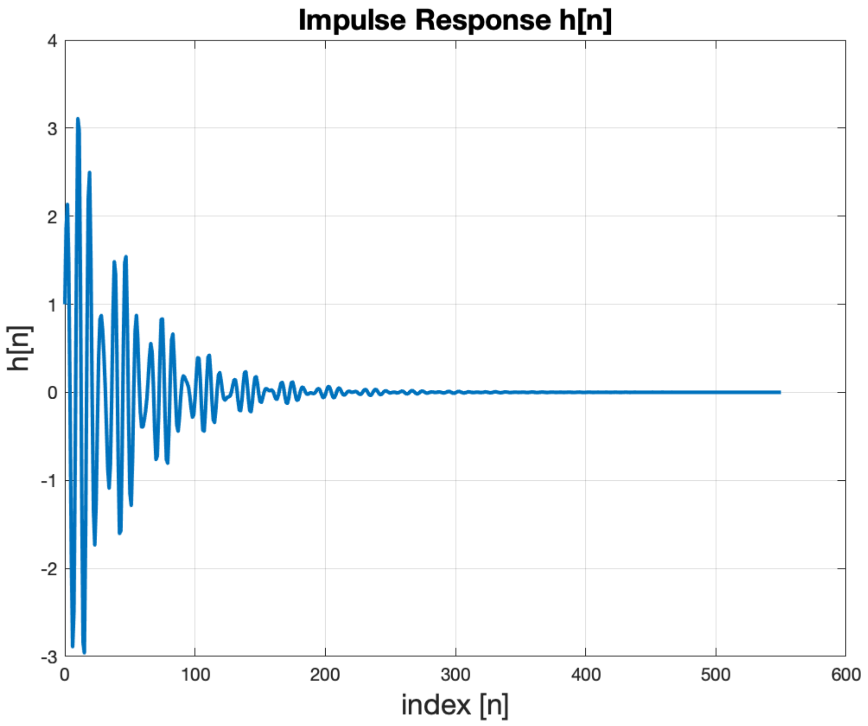

Figure 1: Impulse Response h[n] corresponding to H(z) computed using MATLAB impz function

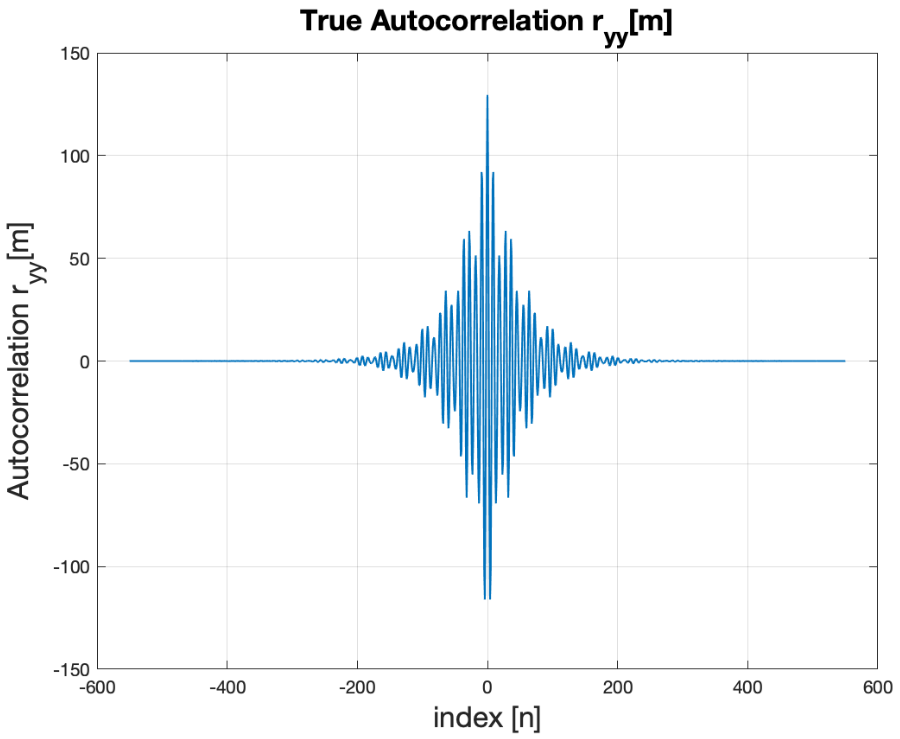

Figure 2: True autocorrelation sequence $r_{yy}[m]$ of the output random process $y[n]$

2 Problem - 1(b): Estimate autocorrelation sequence $\hat{r}_{yy}[m]$ of output random process $y[n]$

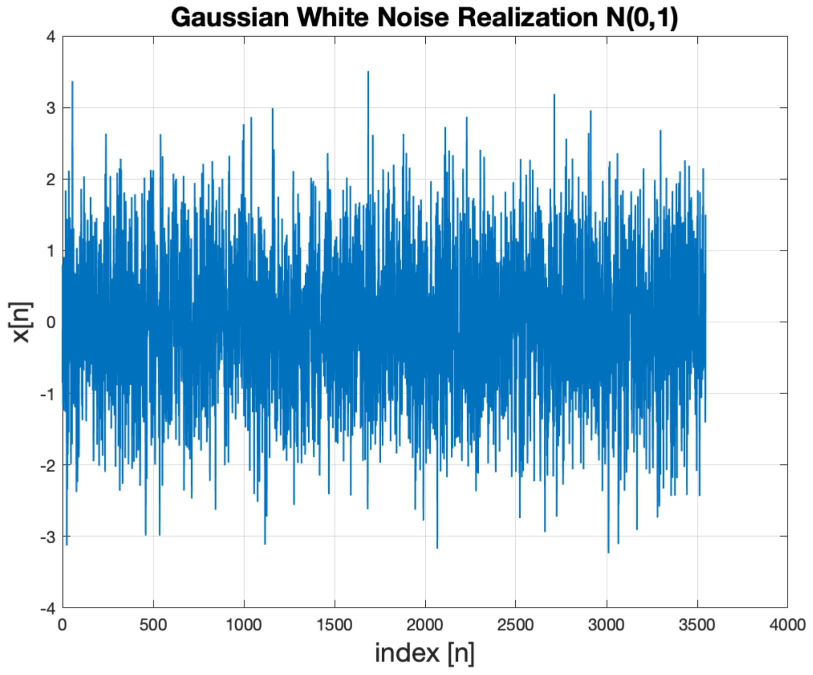

Figure 3: Realization of Gaussian white noise process with 0 mean and variance 1

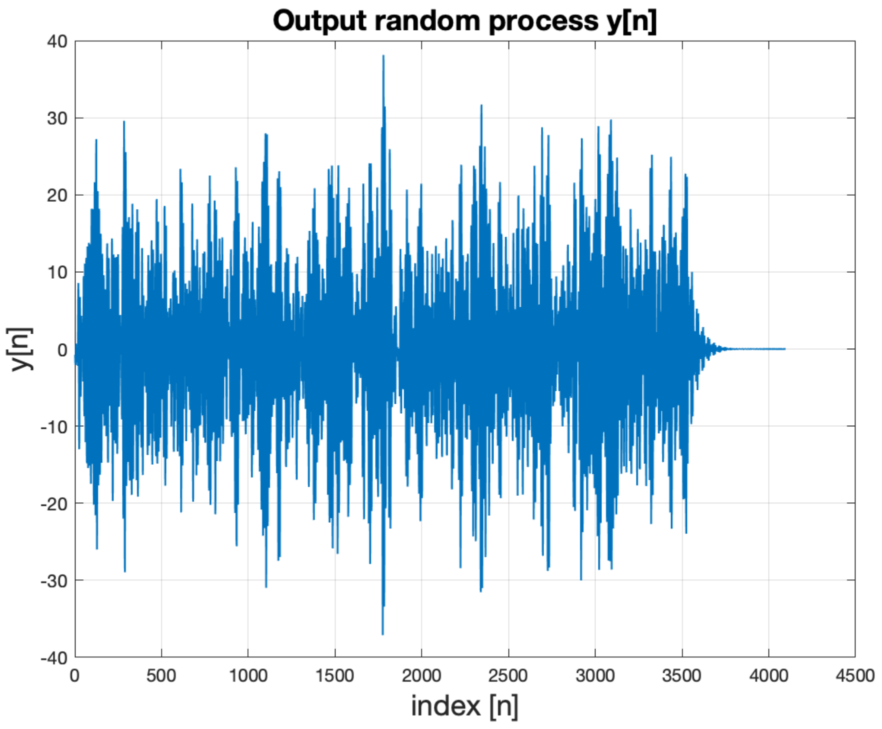

Figure 4: Realization of output random process y[n] generated by passing the above gaussian white noise input through the LTI system H(z). From the generated 4096 output samples, we sample 64, 128, 256, 512, 1024 samples symmetrically from the center so as to avoid any irregularities in the beginning and end.

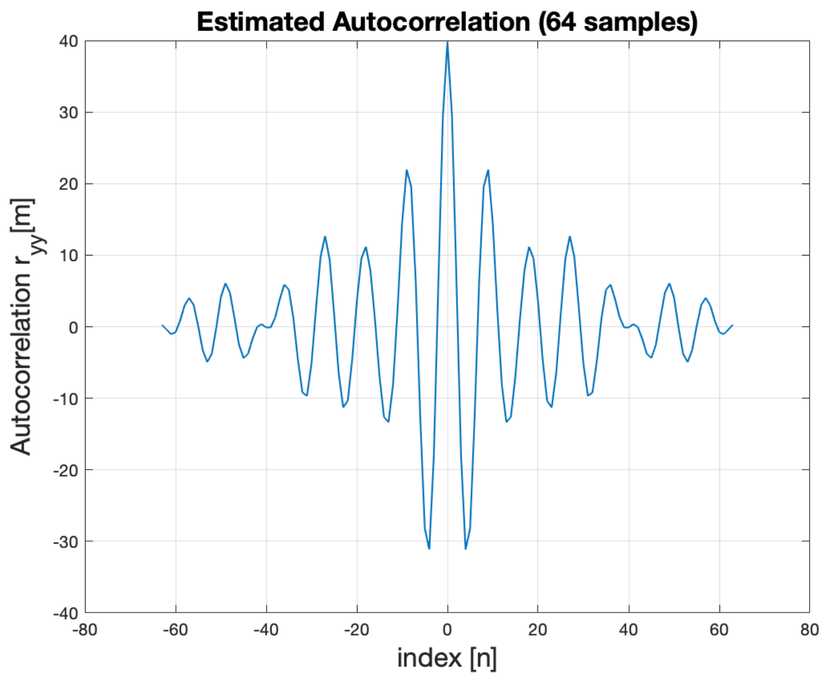

Figure 5: Estimated autocorrelation sequence $r_{yy}[m]$ using 64 samples of random process $y[n]$

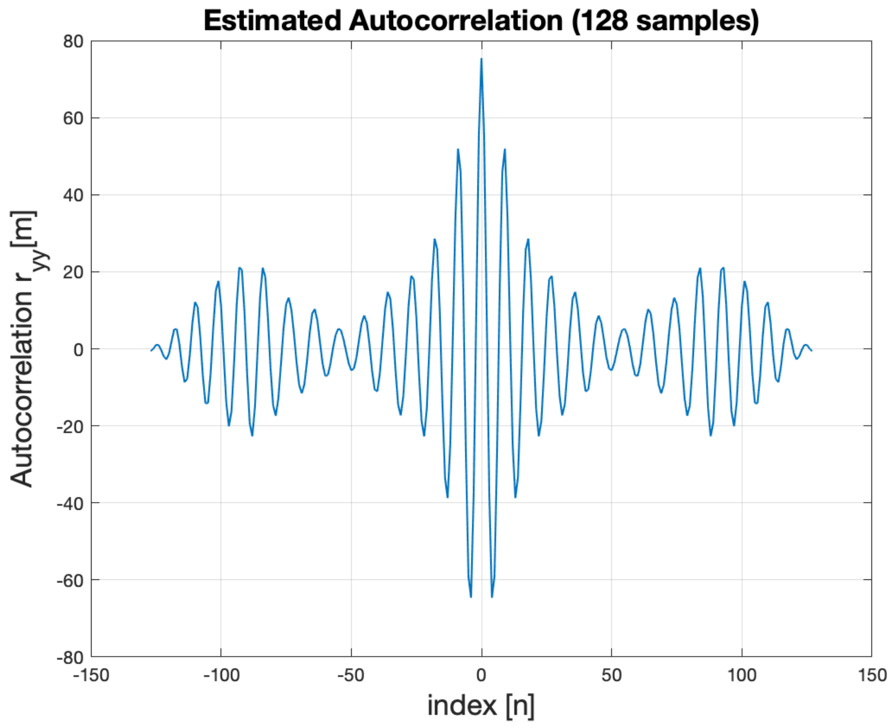

Figure 6: Estimated autocorrelation sequence $r_{yy}[m]$ using 128 samples of random process $y[n]$

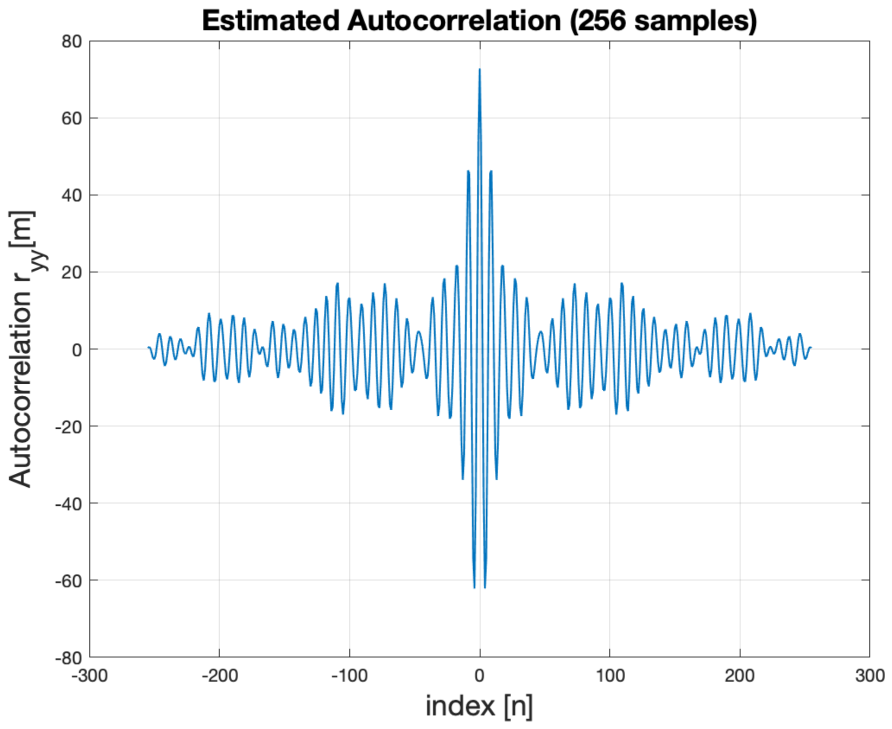

Figure 7: Estimated autocorrelation sequence $r_{yy}[m]$ using 256 samples of random process $y[n]$

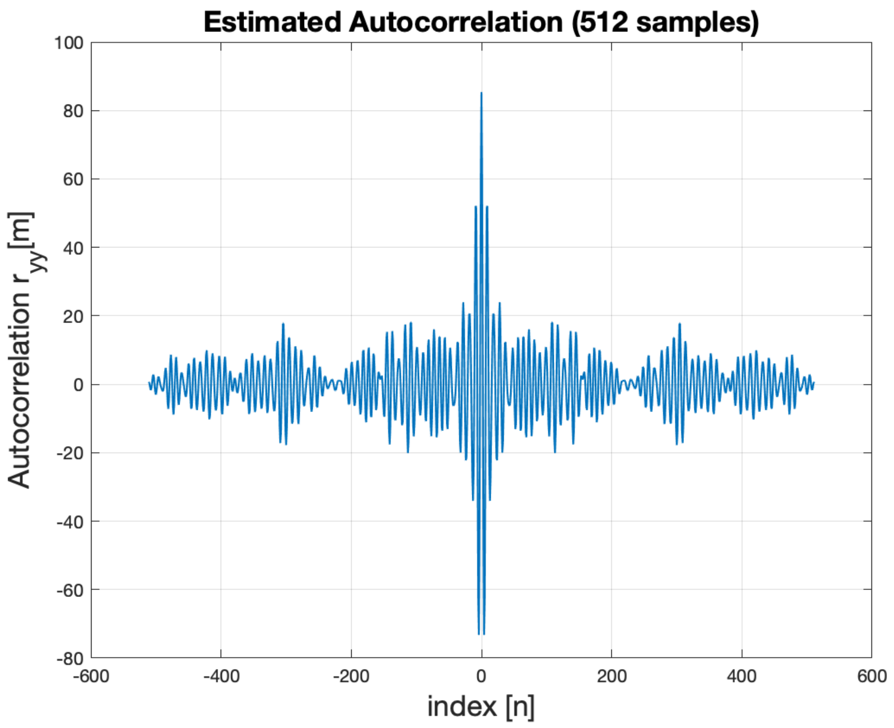

Figure 8: Estimated autocorrelation sequence $r_{yy}[m]$ using 512 samples of random process $y[n]$

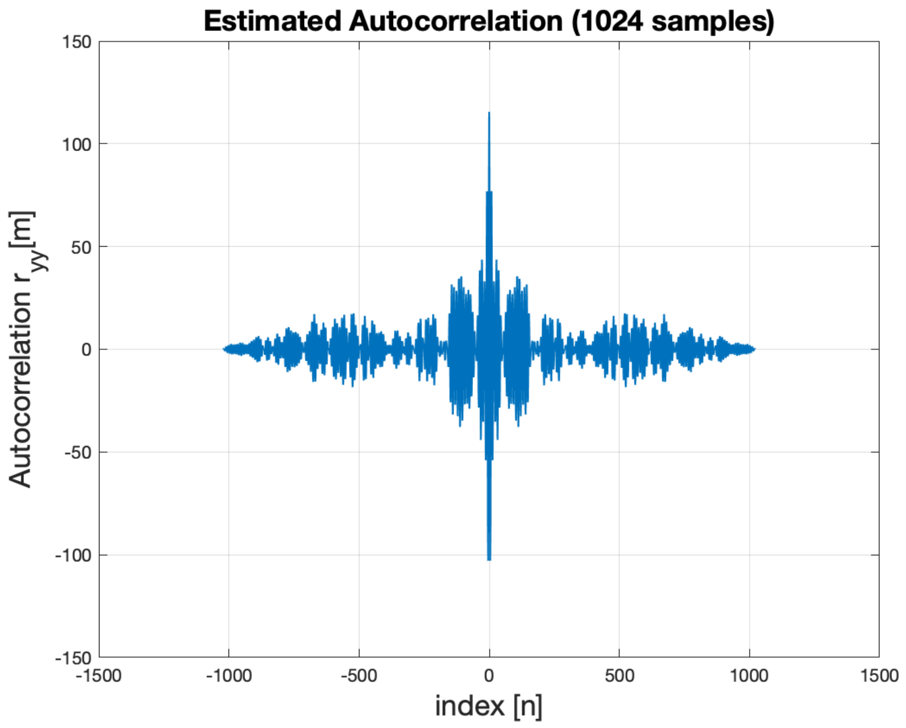

Figure 9: Estimated autocorrelation sequence $r_{yy}[m]$ using 1024 samples of random process $y[n]$

3 Problem - 1(c): Estimate true power spectral density $R_{yy}(e^{j\omega})$ of the output random process $y[n]$

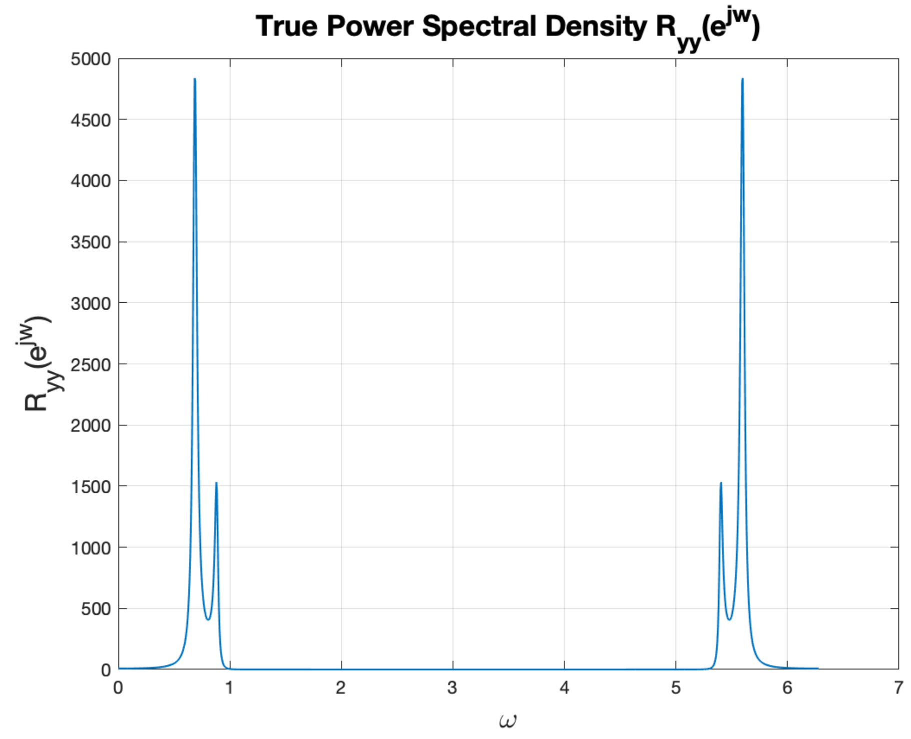

Figure 10: True power spectral density $R_{yy}(e^{j\omega})$ computed by taking the DFT of true autocorrelation sequence in Fig 2

4 Problem - 1(d): Estimate periodogram $\hat{R}_{yy}(e^{j\omega})$ of output random process $y[n]$

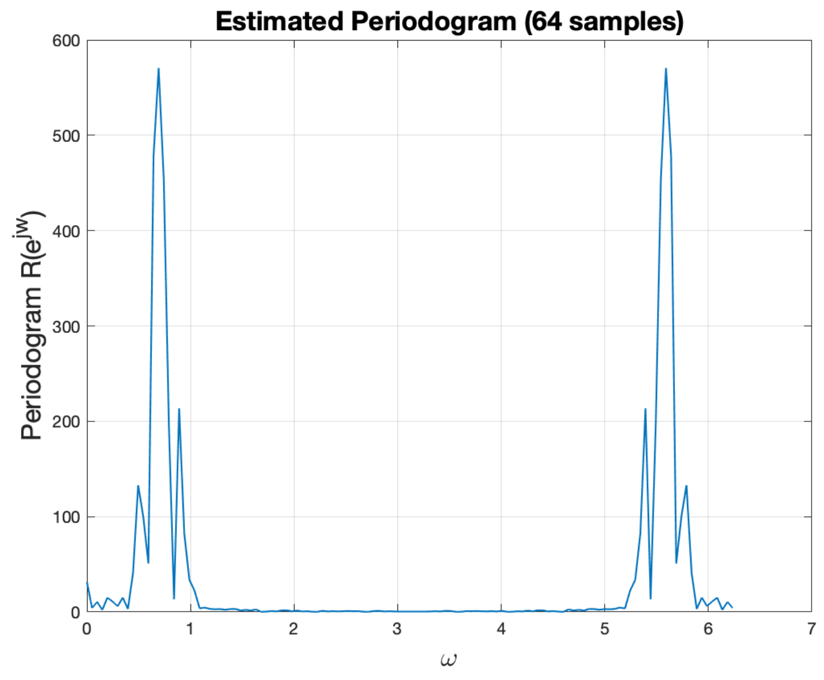

Figure 11: Estimated periodogram of the output random process $y[n]$ for 64 samples

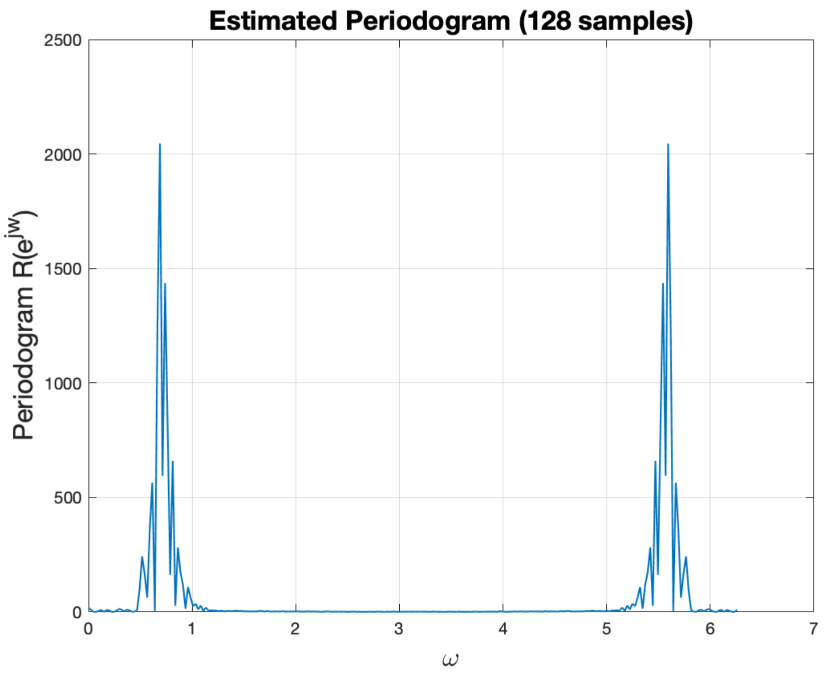

Figure 12: Estimated periodogram of the output random process $y[n]$ for 128 samples

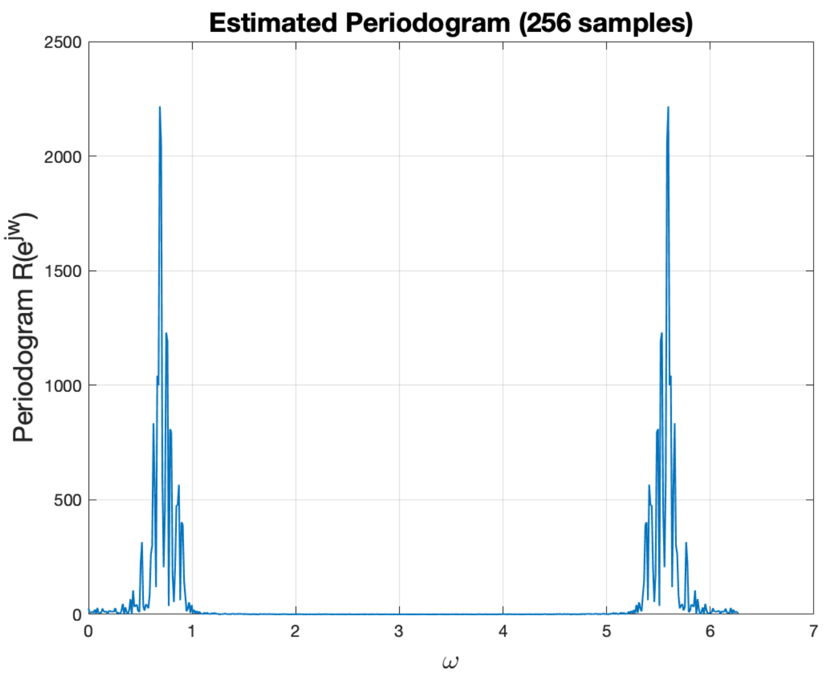

Figure 13: Estimated periodogram of the output random process $y[n]$ for 256 samples

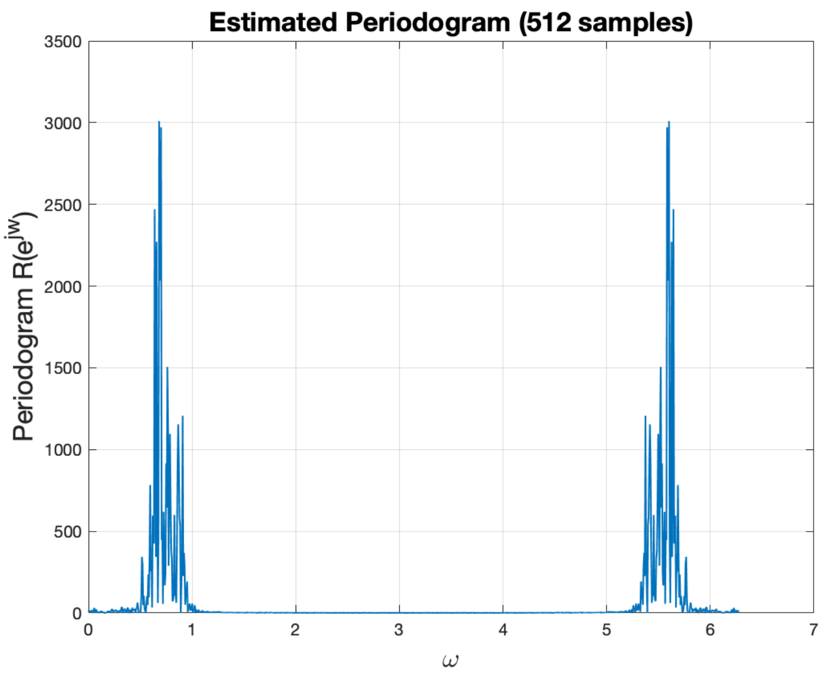

Figure 14: Estimated periodogram of the output random process $y[n]$ for 512 samples

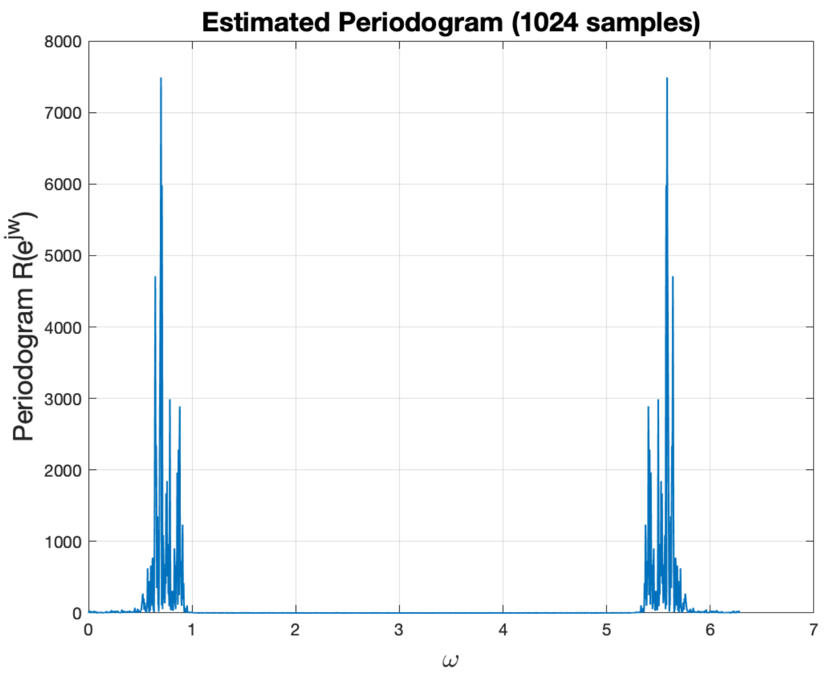

Figure 15: Estimated periodogram of the output random process $y[n]$ for 1024 samples

5 Problem - 1(e): Validation of statistical properties of the periodogram estimate $\hat{R}_{yy}(e^{j\omega})$

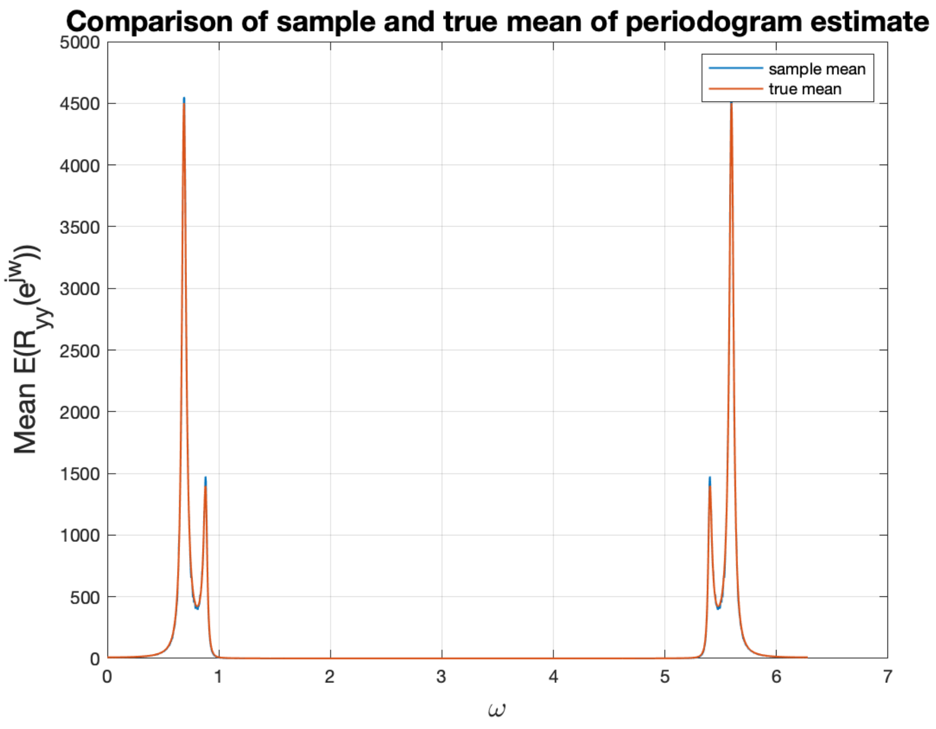

Figure 16: Figure showing a comparison of sample mean (in blue) vs actual mean (in orange) of periodogram estimate. The sample mean is computed across 1000 different realizations by generating the corresponding random process y[n], computing its periodogram estimate and averaging over 1000 iterations. The actual mean is computed using the equation $E\big(\hat{R}_{yy}(e^{j\omega})\big) = \sum_{-(N-1)}^{N-1} \frac{N-|m|}{N} r[m]e^{-j\omega m}$.

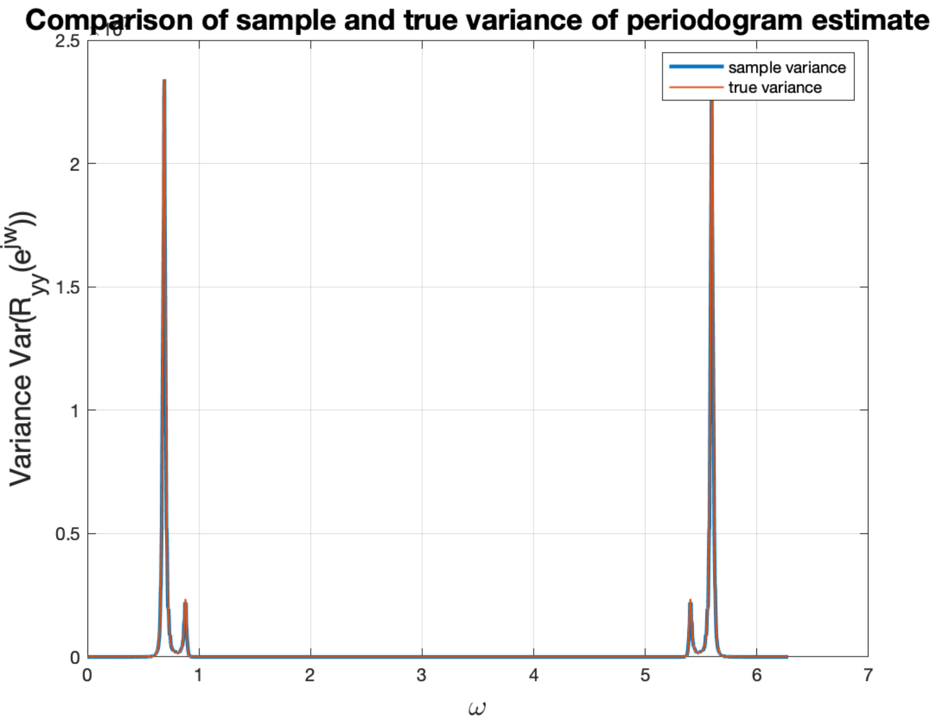

Figure 17: Figure showing a comparison of sample variance (in blue) vs actual variance (in orange) of periodogram estimate. The sample variance is computed across 1000 different realizations by generating the corresponding random process y[n], computing its periodogram estimate, and computing variance as a function of ω by averaging over 1000 iterations. The actual mean is computed using the equation $\mathrm{var}\left(\hat{R}_{yy}(e^{j\omega})\right) = R^2(e^{j\omega})\left[1 + \left(\frac{sinN\omega}{Nsin\omega}\right)^2\right]$.

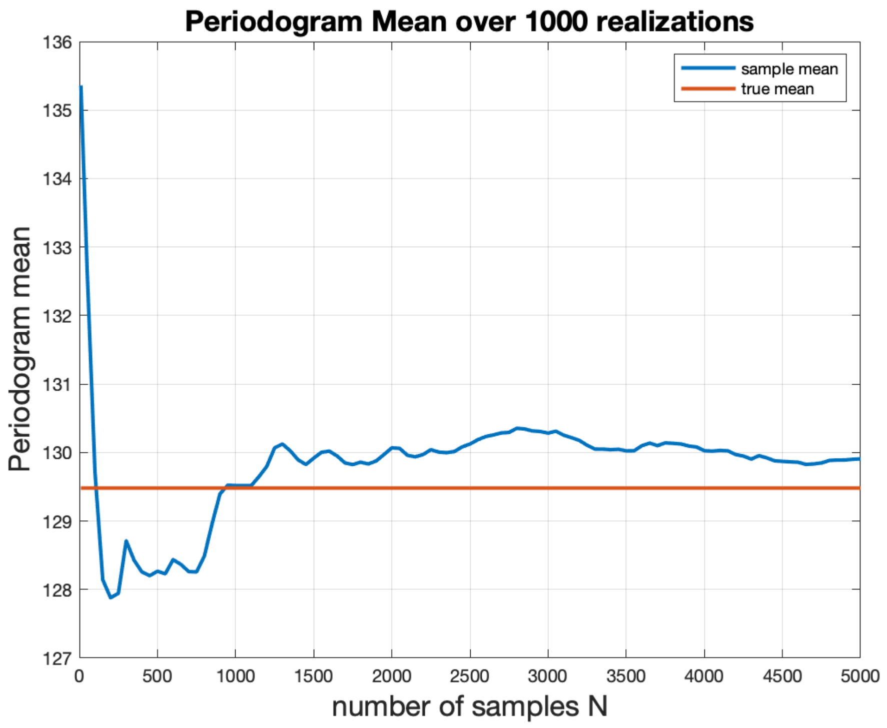

Figure 18: Figure validates the statistical properties of the periodogram estimate. The orange horizontal line refers to the true mean of the periodogram computed using the true power spectral density of output random process y[n]. The blue curve is the sample mean of the periodogram estimate generated for N samples over 1000 different realizations of the output random process. As N increases, we can observe and validate that the sample mean converges towards the true mean

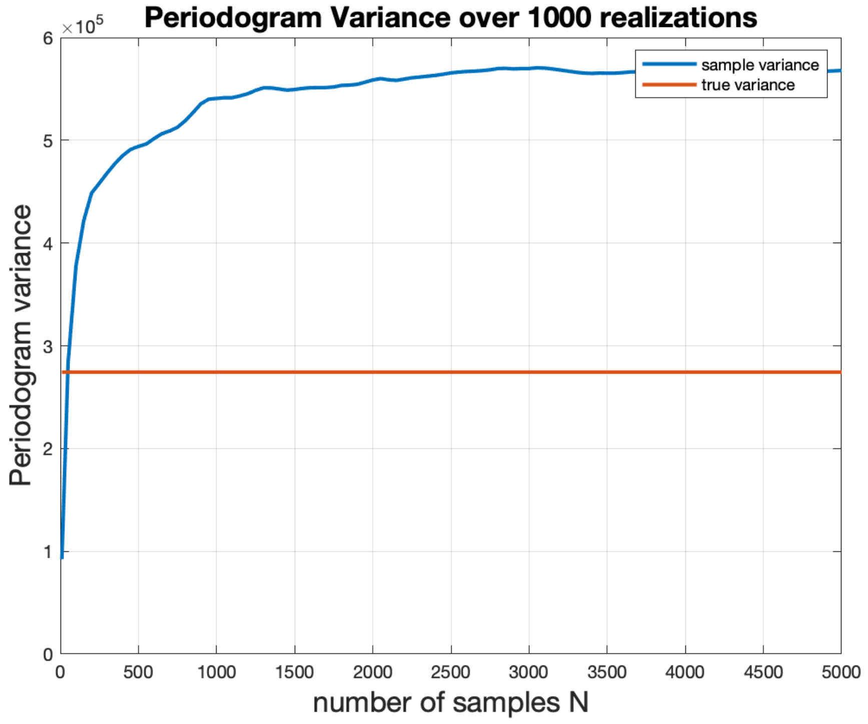

Figure 19: Figure validates the statistical properties of the periodogram estimate. The orange horizontal line refers to the true variance of the periodogram computed using the true power spectral density of output random process y[n]. The blue curve is the sample variance of the periodogram estimate generated for N samples averaged over 1000 different realizations of y[n]. As N increases, we can observe and validate that the sample variance does not converges to the true variance

6 Problem - 2(a): Compute autocorrelation and periodogram of the audio signal

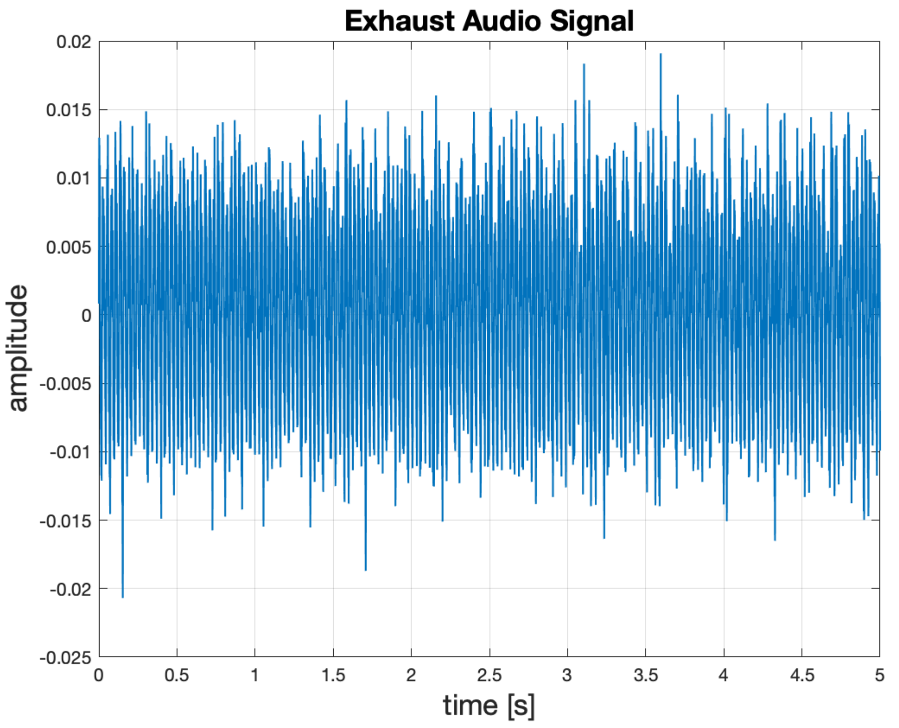

Figure 20: Time domain exhaust audio signal

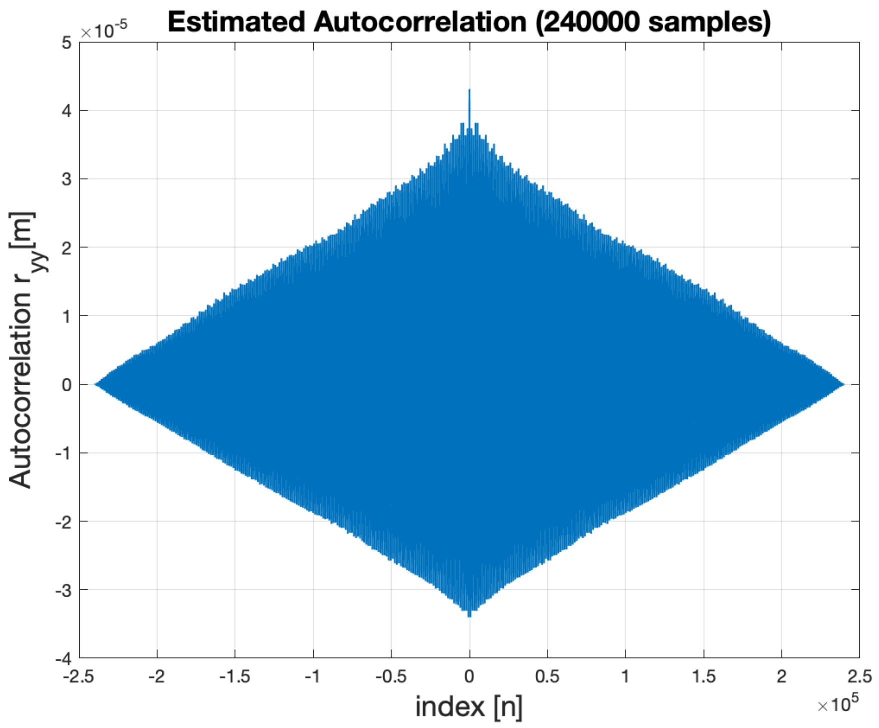

Figure 21: Estimated autocorrelation of the exhaust audio signal

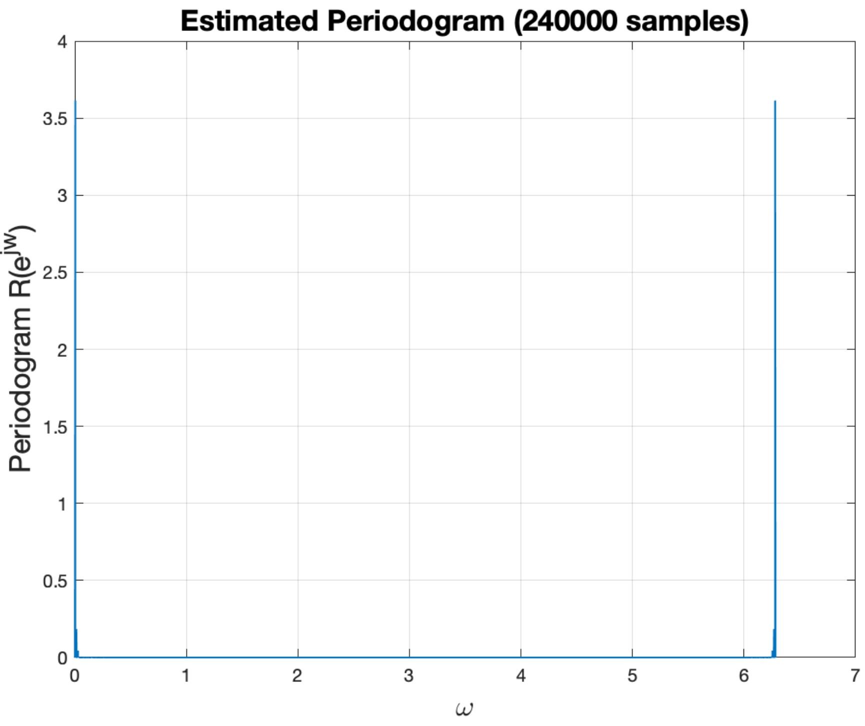

Figure 22: Estimated periodogram of the exhaust audio signal

7 Problem - 2(b): Estimate RPM of the engine

Based on the autocorrelation and periodogram plot, the autocorrelation sequence consists of 2 major tone frequencies - one close to $\omega = 0$ and other close to $\omega = 2\pi$. Given that there are 4 cylinder engines (c1, c2, c3, c4), each firing on every other revolution. Thus, if c1 fires on revolution 1, it would not fire on revolution 2, and then fire on revolution 3, and so on. From the plot we can infer that two cylinders fire on revolution 1,3,5,7,... and the remaining two cylinders fire on revolution 2,4,6,8,....

Period of harmonic sinusoid in samples = 1300

Time period $\approx \frac{1300}{48000} = 0.027083$ sec

Frequency = 1/time period = 36.9235 Hz or RPS

RPM = 2215 RPM

Since there are two sets of cylinders firing on alternate revolutions, RPM of the engine = 2215/2 = 1107.5 RPM

8 MATLAB Code

8.1 prob1.m

```
clc; clear;

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

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

fig = figure;
plot(tn, hn, LineWidth=2);
xlabel("index [n]", FontSize=16);
ylabel("h[n]", FontSize=16);
title("Impulse Response h[n]", FontSize=16);
grid on;
saveas(fig, "../plots/prob1a/impulse_response.png");
close;

% compute true autocorrelation of y[n]
savepath = "../plots/prob1a/true_autocorr.png";
ryy_true = compute_true_autocorr(hn, true, savepath);

% generate 1024 samples of white gaussian noise random process x[n] ~
% N(0,1)
Ny = 4097;
[xn, yn] = generate_random_process(hn, Ny, true);

% estimate autocorrelation sequence of y[n] for various sample lengths
num_samp_lst = [64, 128, 256, 512, 1024];
for num_samp = num_samp_lst
    % 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);

    savepath = "../plots/prob1b/est_autocorr_"+num_samp+".png";
    ryy_est = estimate_autocorr(y_samp, true, savepath);
end

% compute true power spectral density of y[n], Ryy(e^jw)
savepath = "../plots/prob1c/true_psd.png";
Ryy_true = compute_true_periodogram(ryy_true, true, savepath);


% compute periodogram estimates
num_samp_lst = [64, 128, 256, 512, 1024];
```

```
for num_samp = num_samp_lst
%     y_samp = yn(1:num_samp, 1);
    y_samp = yn((Ny+1)/2 - num_samp/2 : (Ny+1)/2 + num_samp/2 - 1, 1);
    assert(size(y_samp, 1) == num_samp);

    savepath = "../plots/prob1d/est_periodogram_"+num_samp+".png";
    Ryy_est = estimate_periodogram(y_samp, true, savepath);

%     ryy_est = estimate_autocorr(y_samp, false, "");
%     Ryy_est_2 = abs(fft(ryy_est));
%     close;
%     disp("This should be same:"+max(abs(Ryy_est_2 - Ryy_est)));
end


% generate several realization of the periodogram estimate
num_realizations = 1000;
Ny = 4097;
num_samp = 1024;

% estimate sample mean of the periodogram estimate
savepath = "../plots/prob1e/periodogram_mean_comparison.png";
[Ryy_est_true_mean, Ryy_est_samp_mean] = validate_periodogram_mean(hn, ryy_true, num_realizati

% estimate sample variance of the periodogram estimate
savepath = "../plots/prob1e/periodogram_var_comparison.png";
[Ryy_est_true_var, Ryy_est_samp_var] = validate_periodogram_var(hn, Ryy_est_true_mean, Ryy_tru


% validate the statistical properties of the periodogram estimate
% generate several realization of the periodogram estimate
num_realizations = 1000;
Ny = 4096*2+1;

num_samp_lst = transpose([10, 50:50:5000]);
len_lst = size(num_samp_lst, 1);

periodogram_mean_arr = zeros(len_lst, num_realizations);
periodogram_var_arr = zeros(len_lst, num_realizations);

disp("Computing mean and variance across 1000 realizations...");
for i = 1:num_realizations
    [xn, yn] = generate_random_process(hn, Ny, false);
    for j = 1:len_lst
        num_samp = num_samp_lst(j,1);
        y_samp = yn((Ny+1)/2 - num_samp/2 : (Ny+1)/2 + num_samp/2 - 1, 1);
        assert(size(y_samp,1) == num_samp);

        Ryy_est = estimate_periodogram(y_samp, false, "");
        periodogram_mean_arr(j, i) = mean(Ryy_est);
        periodogram_var_arr(j, i) = var(Ryy_est);
    end
end

fig = figure;
plot(num_samp_lst, mean(periodogram_mean_arr, 2), LineWidth=2); hold on;
plot(num_samp_lst, mean(real(Ryy_true)) + num_samp_lst * 0, LineWidth=2); hold off;
xlabel("number of samples N", FontSize=16);
ylabel("Periodogram mean", FontSize=16);
title("Periodogram Mean over 1000 realizations", FontSize=16);
grid on;
legend("sample mean", "true mean");
saveas(fig, "../plots/prob1e/periodogram_sample_mean.png");
close;

fig = figure;
plot(num_samp_lst, mean(periodogram_var_arr, 2), LineWidth=2); hold on;
plot(num_samp_lst, var(real(Ryy_true)) + num_samp_lst * 0, LineWidth=2); hold off;
xlabel("number of samples N", FontSize=16);
ylabel("Periodogram variance", FontSize=16);
title("Periodogram Variance over 1000 realizations", FontSize=16);
grid on;
legend("sample variance", "true variance");
saveas(fig, "../plots/prob1e/periodogram_sample_var.png");
close;
```

8.2 compute_true_autocorr.m

```
function [rxx] = compute_true_autocorr(hn, tosave, savepath)
    Nh = size(hn, 1);

    rxx = zeros(2 * Nh - 1, 1);

    for m = -(Nh - 1) : (Nh - 1)
        for p = 0 : Nh - 1
            k = m + p;
            if k >= 0 && k <= Nh - 1
                rxx(Nh + m, 1) = rxx(Nh + m, 1) + hn(k + 1, 1) * conj(hn(p + 1, 1));
            end
        end
    end

    fig = figure;
    plot(-(Nh-1):(Nh-1), rxx, LineWidth=1);
    xlabel("index [n]", FontSize=16);
    ylabel("Autocorrelation r_{yy}[m]", FontSize=16);
    title("True Autocorrelation r_{yy}[m]", FontSize=16);
    grid on;
    if tosave == true
        saveas(fig, savepath);
        close;
    end
end
```

8.3 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("index [n]", FontSize=16);
        ylabel("x[n]", FontSize=16);
        title("Gaussian White Noise Realization N(0,1)", FontSize=16);
        grid on;
        saveas(fig, "../plots/prob1b/gaussian_white_input.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("index [n]", FontSize=16);
        ylabel("y[n]", FontSize=16);
        title("Output random process y[n]", FontSize=16);
        grid on;
        saveas(fig, "../plots/prob1b/out_random_process.png");
        close;
    end
end
```

8.4 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("index [n]", FontSize=16);
        ylabel("Autocorrelation r_{yy}[m]", FontSize=16);
        title("Estimated Autocorrelation ("+N+" samples)", FontSize=16);
        grid on;
        saveas(fig, savepath);
        close;
    end
end
```

8.5 compute_true_periodogram.m

```
function [Ryy_true] = compute_true_periodogram(ryy, tosave, savepath)
    N_ryy = size(ryy, 1);
    Ryy_true = zeros(N_ryy, 1);

    m_arr = -(N_ryy - 1) / 2 : (N_ryy - 1) / 2;

    for n = 0 : N_ryy - 1
        exp_term = transpose(exp(-1j * 2 * pi * n * m_arr / N_ryy));
        Ryy_true(n+1, 1) = sum(ryy(m_arr + (N_ryy - 1)/2 + 1, 1) .* exp_term);
    end

%     Ryy_true_2 = zeros(N_ryy, 1);
%     for n = 0 : N_ryy - 1
%         for m = -(N_ryy - 1) / 2 : (N_ryy - 1) / 2
%             Ryy_true_2(n+1, 1) = Ryy_true_2(n+1, 1) + ryy(m + (N_ryy - 1)/2 + 1, 1) * exp(-1
%           end
%       end
%       Ryy_diff = max(abs(real(Ryy_true) - real(Ryy_true_2)));
%       disp("Diff between custom FFT and system FFT:"+Ryy_diff);

%       Ryy_fft = fft(circshift(ryy, 550));
%       Ryy_diff = sum(abs(real(Ryy_true) - abs(Ryy_fft)));
%       disp("Diff between custom FFT and system FFT:"+Ryy_diff);

    if tosave == true
        fig = figure;
        plot((2 * pi / N_ryy) * (0 : N_ryy - 1), real(Ryy_true), LineWidth=1);
    %     plot((2 * pi / N_ryy) * (0 : N_ryy - 1), abs(Ryy_fft), LineWidth=1); hold off;
        xlabel("\omega", FontSize=16);
        ylabel("R_{yy}(e^{jw})", FontSize=16);
        title("True Power Spectral Density R_{yy}(e^{jw})", FontSize=16);
        grid on;
        saveas(fig, savepath);
        close;
    end
end
```

8.6 estimate_periodogram.m

```
function [rxx_fft] = estimate_periodogram(xn, tosave, savepath)
    % estimate periodogram using N samples
    Nx = size(xn, 1);
    xn_fft = fft(xn, 2 * Nx - 1);
    rxx_fft = (1/Nx) * abs(xn_fft).^2;
    N_rxx = size(rxx_fft, 1);

    if tosave == true
        fig = figure;
        plot((2 * pi / N_rxx) * (0 : N_rxx - 1), rxx_fft, LineWidth=1);
        xlabel("\omega", FontSize=16);
        ylabel("Periodogram R(e^{jw})", FontSize=16);
        title("Estimated Periodogram ("+Nx+" samples)", FontSize=16);
        grid on;
        saveas(fig, savepath);
        close;
    end
end
```

8.7 validate_periodogram_mean.m

```
function [Ryy_est_true_mean, Ryy_est_samp_mean] = validate_periodogram_mean(hn, ryy_true, num_
    % estimate sample periodogram mean over many realizations
    Ryy_est_samp_mean = zeros(2 * num_samp - 1, 1);

    for i = 1:num_realizations
        [xn, yn] = generate_random_process(hn, Ny, false);

        % y_samp = yn(1:num_samp, 1);
        y_samp = yn((Ny+1)/2 - num_samp/2 : (Ny+1)/2 + num_samp/2 - 1, 1);
        assert(size(y_samp,1) == num_samp);

        Ryy_est = estimate_periodogram(y_samp, false, "");
        Ryy_est_samp_mean = Ryy_est_samp_mean + Ryy_est;
    end

    Ryy_est_samp_mean = Ryy_est_samp_mean / num_realizations;

    % estimate actual periodogram mean using the formula in slides
    N_ryy_true = size(ryy_true, 1);
    m = -(N_ryy_true - 1)/2 : (N_ryy_true - 1)/2;

    N = 2 * num_samp - 1;
    Ryy_est_true_mean = zeros(N, 1);
    for n = 0 : N-1
        window = transpose(1 - abs(m) / ((N_ryy_true + 1)/2));
        exp_term = transpose(exp(-1j * 2 * pi * n * m / N));
        Ryy_est_true_mean(n+1, 1) = sum(window .* ryy_true .* exp_term);
    end

    if tosave == true
        fig = figure;
        plot((2*pi/N)*(0:N-1), Ryy_est_samp_mean, LineWidth=1); hold on;
        plot((2*pi/N)*(0:N-1), real(Ryy_est_true_mean), LineWidth=1); hold off;
        xlabel("\omega", FontSize=16);
        ylabel("Mean E(R_{yy}(e^{jw}))", FontSize=16);
        title("Comparison of sample and true mean of periodogram estimate", FontSize=16);
        grid on;
        legend("sample mean", "true mean");
        saveas(fig, savepath);
        close;
    end
end
```

8.8 validate_periodogram_var.m

```
function [Ryy_est_true_var, Ryy_est_samp_var] = validate_periodogram_var(hn, Ryy_est_true_mean
    % estimate sample periodogram variance over many realizations
    N_Ryy_est = 2 * num_samp - 1;
    Ryy_est_samp_var = zeros(N_Ryy_est, 1);
    for i = 1:num_realizations
        [xn, yn] = generate_random_process(hn, Ny, false);

        % y_samp = yn(1:num_samp, 1);
        y_samp = yn((Ny+1)/2 - num_samp/2 : (Ny+1)/2 + num_samp/2 - 1, 1);
        assert(size(y_samp, 1) == num_samp);

        Ryy_est = estimate_periodogram(y_samp, false, "");
        Ryy_est_samp_var = Ryy_est_samp_var + (Ryy_est - Ryy_est_true_mean).^2;
    end

    Ryy_est_samp_var = Ryy_est_samp_var / (num_realizations-1);

    % estimate true variance of the periodogram estimate
    N_Ryy_true = size(Ryy_true, 1);
    omega_arr = transpose((2*pi / N_Ryy_true) * (0 : N_Ryy_true - 1));

    Ryy_est_true_var = (Ryy_true.^2) .* (1 + (sin(N_Ryy_true * omega_arr) ./ (N_Ryy_true * sin
    disp(size(Ryy_est_true_var));
    disp(size(Ryy_est_samp_var));

    if tosave == true
        fig = figure;
        plot((2*pi / N_Ryy_est) * (0 : N_Ryy_est - 1), real(Ryy_est_samp_var), LineWidth=2); h
        plot((2*pi / N_Ryy_true) * (0 : N_Ryy_true - 1), real(Ryy_est_true_var), LineWidth=1);
        xlabel("\omega", FontSize=16);
        ylabel("Variance Var(R_{yy}(e^{jw}))", FontSize=16);
        title("Comparison of sample and true variance of periodogram estimate", FontSize=16);
        grid on;
        legend("sample variance", "true variance");
        saveas(fig, savepath);
        close;
    end
end
```

8.9 prob2.m

```
clc; clear;

audio_filepath = "../data/exhaust.wav";
[audio_data, samp_freq] = audioread(audio_filepath);
num_samp = size(audio_data,1);
% sound(audio_data, samp_freq);

fig = figure;
plot((0:num_samp-1)/samp_freq, audio_data, LineWidth=1);
xlabel("time [s]", FontSize=16);
```

```
ylabel("amplitude", FontSize=16);
title("Exhaust Audio Signal", FontSize=16);
grid on;
saveas(fig, "../plots/prob2a/exhaust_audio_signal.png");
close;


% estimate autocorrelation of the audio signal
disp("Computing autocorrelation of audio signal...");
savepath = "../plots/prob2a/exhaust_autocorr.png";
ryy_audio = estimate_autocorr(audio_data, true, savepath);
disp("Estimated autocorrelation of audio signal");

% estimate periodogram of the audio signal
disp("Computing periodogram of audio signal...");
savepath = "../plots/prob2a/exhaust_periodogram.png";
Ryy_audio = estimate_periodogram(audio_data, true, savepath);
disp("Estimated periodogram of audio signal");
```