~~Given~~

Statistical Signal Processing
Assignment-1

Pratyush Jaiswal
18EE35014

Given the random process

$$y(n) = A\cos(\omega_0 n + \phi)$$

where $A = 0.5$, $f_0 = 0.05$

The phase is coupled by AWGN

$$v(n) \sim N(0, \sigma_v^2)$$

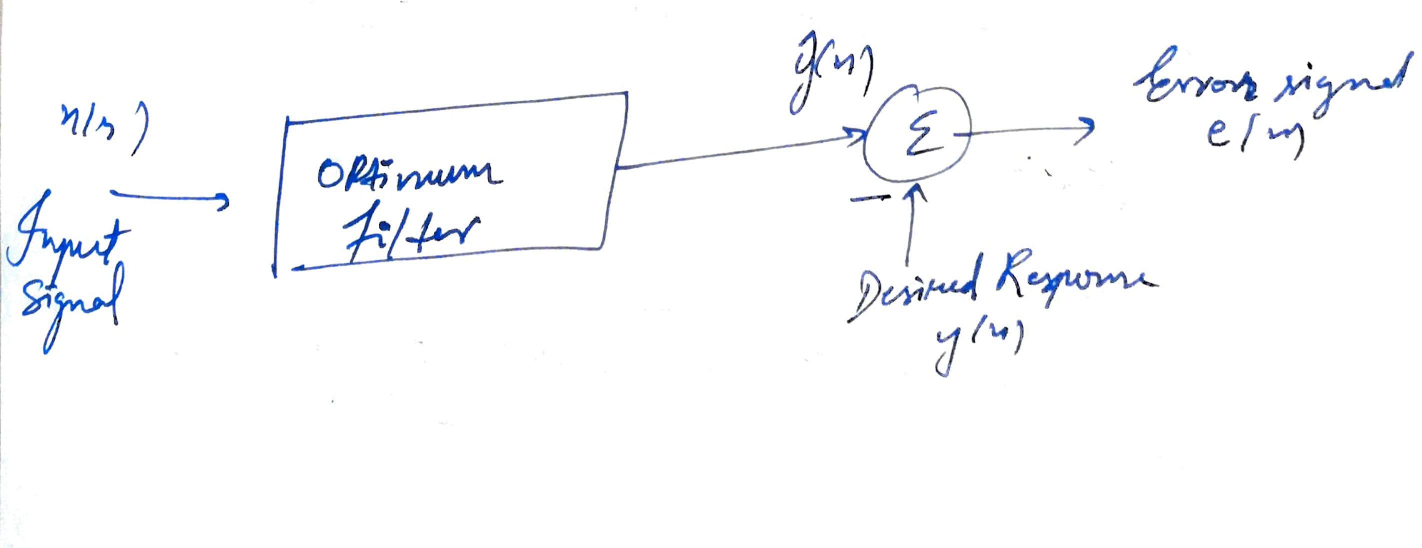
$$\sigma_v^2 = 0.5$$

$$x(n) = y(n) + v(n)$$

$v(n) \rightarrow$ AWGN $\sim N(0, \sigma_v^2)$

$y(n) \rightarrow$ Random Process

$x(n) \rightarrow$ Coupled Random Process



$y(n)$ & $v(n)$ are uncorrelated, autocorrelation of the input signal,

$$r_x(l) = r_y(l) + r_v(l)$$

$$r_{yx}(l) = E\left[y(n)\left[y(n-l) + v(n-l) \right] \right]$$

The auto-correlation matrix

$$R = E\left[x(n)\, x^H(n) \right]$$

$$x(n) = \left[x(n),\ x(n-1)\ \ldots\ x(n-M+1) \right]^T$$

$$R = \begin{bmatrix} r(0) & r(1) & \cdots & r(M-1) \\ r(1) & r(0) & \cdots & r(M-2) \\ \vdots & \vdots & & \vdots \\ r(M-1) & \cdots & & r(0) \end{bmatrix}$$

Cross-Correlation Vector

$$P = E\left[x(n)\, y^*(n) \right]$$

$$P = \left[p(0),\ p(-1),\ p(-2)\ \ldots\ p(1-M) \right]^T$$

$$R w_o = P$$

$$w_o = R^{-1} P$$

$$\boxed{w_o = R^{-1} r_{yx}(l)} \longrightarrow \text{Optimum weight vector}$$

$$\hat{y}(n) = \sum_{k=0}^{M-1} w_{ok}\, x(n-k)$$

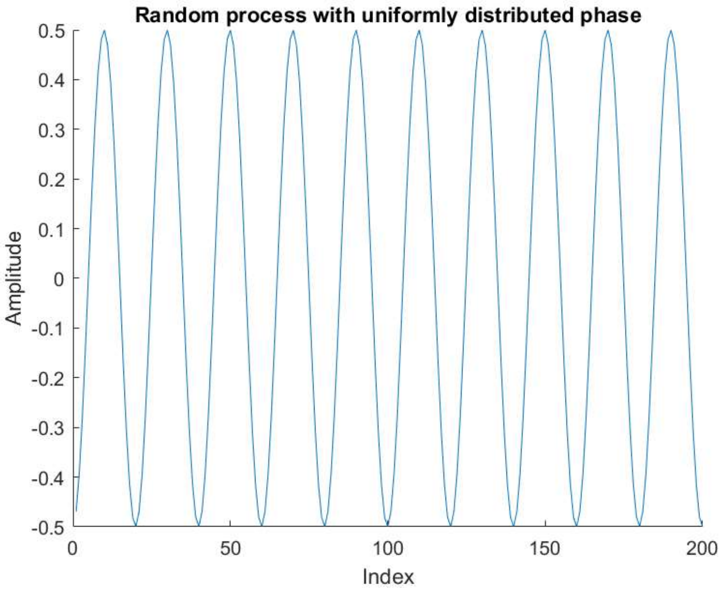
↳ Output of filter

w_{ok} → are the optimum weights.

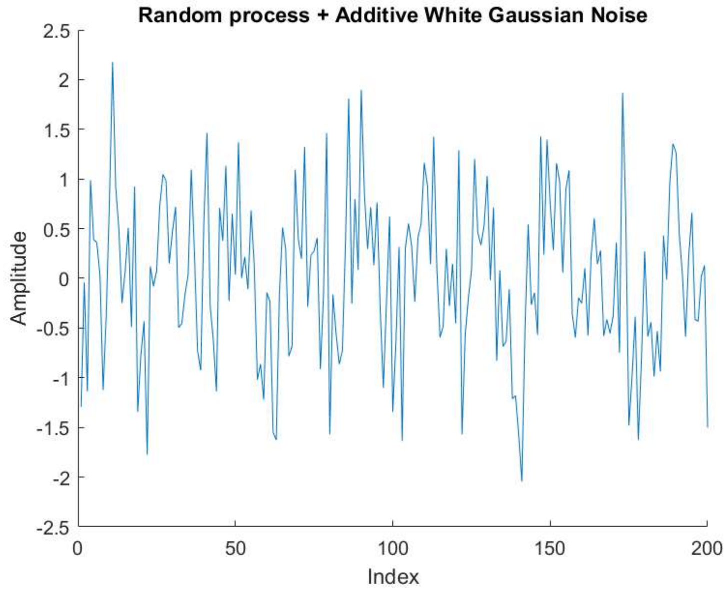
```
% Submitted By:
% Pratyush Jaiswal
% 18EE35014
```

```
close all;
clear all;
clc;
```

```
%% Data generation

A = 0.5;          % Amplitude of random process
f_0 = 0.05;        % Frequency
w = 2*pi*f_0;      % Getting Angular Frequency

indices_arr = 1:200;      % Indices for recording relaization
N = numel(indices_arr);   % Storing number of elements in indices

y = random_prcoess(A, w, indices_arr); % Getting the desired signal
                            % for above parameters

% Plotting the Amplitude of the above random process
figure(1)
hold on;
plot(y)
title('Random process with uniformly distributed phase')
ylabel('Amplitude')
xlabel('Index')
hold off;
```




```
%% Noise Addition

var = 0.5;                   % Setting Noise variance
v = sqrt(var)*randn(1, N);  % Adding Gaussian noise

% Mixing Signal and Noise
x = y + v;

% Plotting parameters of the random process
figure(2)
hold on;
plot(x)
title('Random process + Additive White Gaussian Noise')
ylabel('Amplitude')
xlabel('Index')
hold off;
```



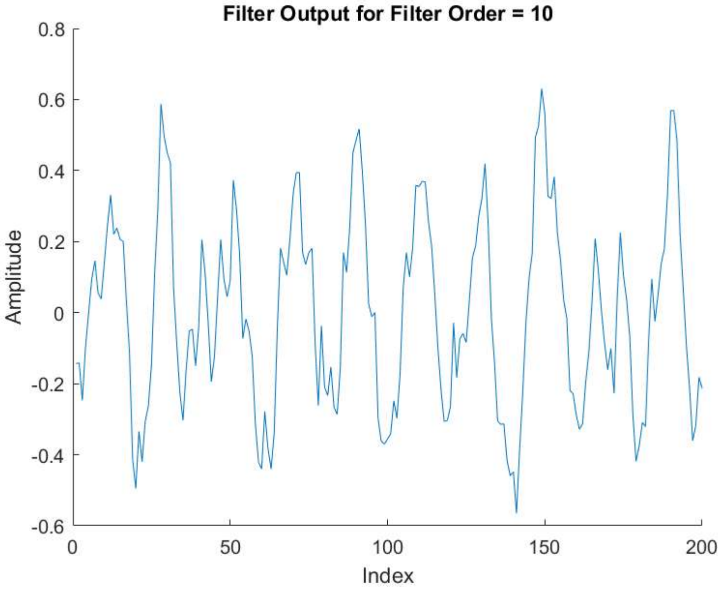

```
err = rms(x - y)^2;          % Mean Squared Error
disp("Initial mean squared error = "+num2str(err));
```

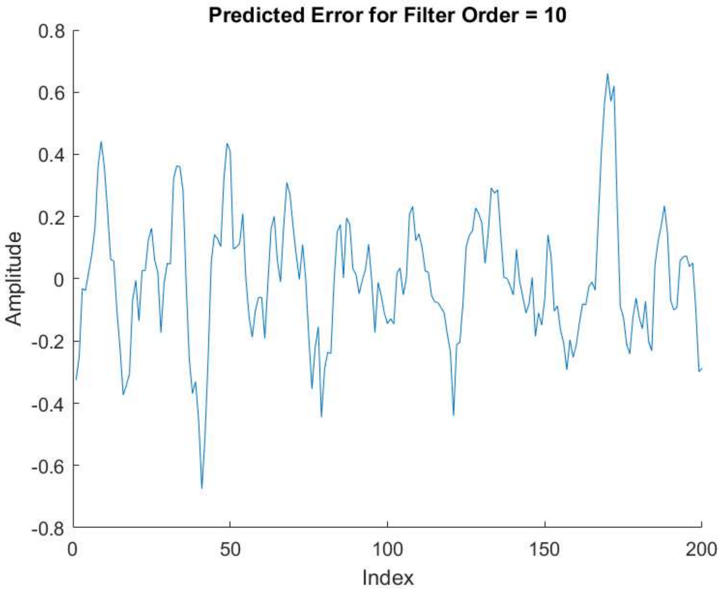
```
Initial mean squared error = 0.50127
```

```
%% Weiner-Hopf Filter Implementation

filter_orders = [10 15 20];   % Array containing the filter orders

mean_squared_error_arr = [];  % Array for storing the mean squared errors

for order = filter_orders
    wiener_weights = wiener_filter(A, var, w, order, order);    % Obtaining optimum weights (from function defined below)

    % Convolution with filter coefficients
    y_hat = zeros(1, N);      % Generating an array for storing the convultionized signal
    for n=1:N
        if n < order
            % If the order is greater than number of samples, we will have
            % to append zeros in front of the signal to make the sample
            % size equal to n for convoluation
            x_w = [zeros(1, order-n), x(1:n)].';    % Window having n elements of x (signal) (making
        else
            x_w = x(n-order+1:n).';                 % Window having n elements of x (signal)
        end
        % Calculating the convolutionized result for current sample and
        % storing it
        y_hat(n) = conj(wiener_weights)'*flipud(x_w);   % y_hat(n) = sum over k(wH(k) * x(n-k))
    end


    err_sig = y - y_hat;                                          % Computing error signal
    error = rms(y(order:N)-y_hat(order:N))^2;                     % Calculating Mean Squared Error
    mean_squared_error_arr = [mean_squared_error_arr; error];     % Sotring the mean squqared error
    disp("Mean squared error with filter order "+num2str(order)+" = "+num2str(error));

    % Plotting y(n)
    figure
    hold on;
    plot(y_hat);
    title("Filter Output for Filter Order = "+num2str(order));
    ylabel('Amplitude');
    xlabel('Index');
    hold off;

    % Plotting the preicted error for this filter order
    figure
    hold on;
    plot(err_sig);
    title("Predicted Error for Filter Order = "+num2str(order));
    ylabel('Amplitude');
    xlabel('Index');
    hold off;
```

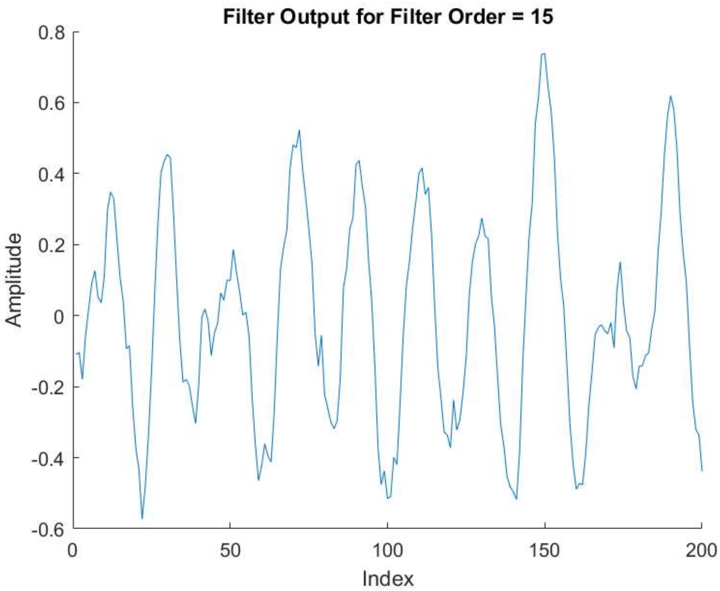
```
end
```

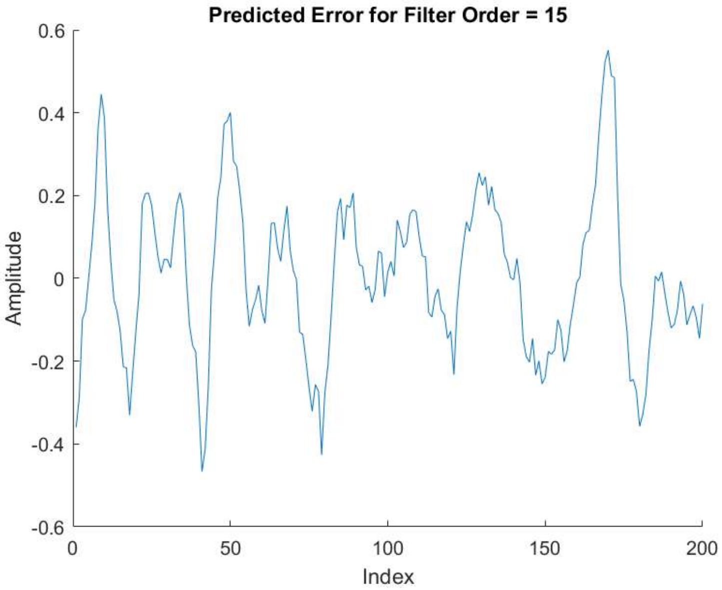
```
Mean squared error with filter order 10 = 0.042915
```



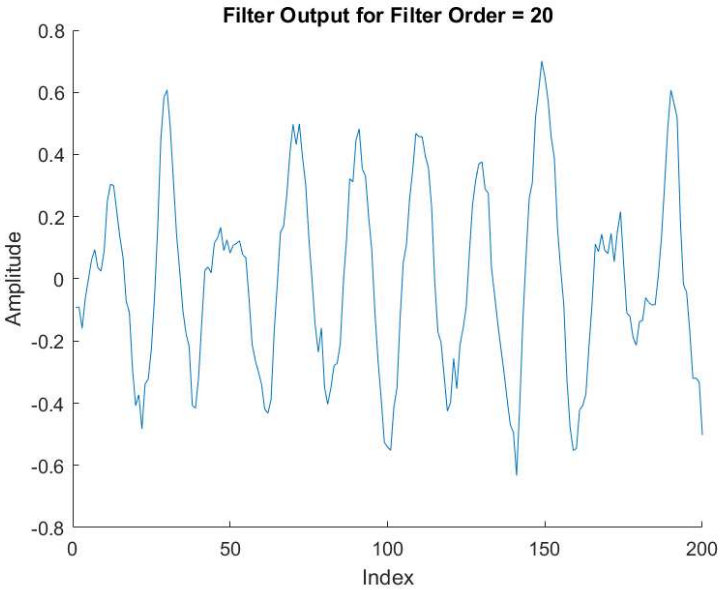


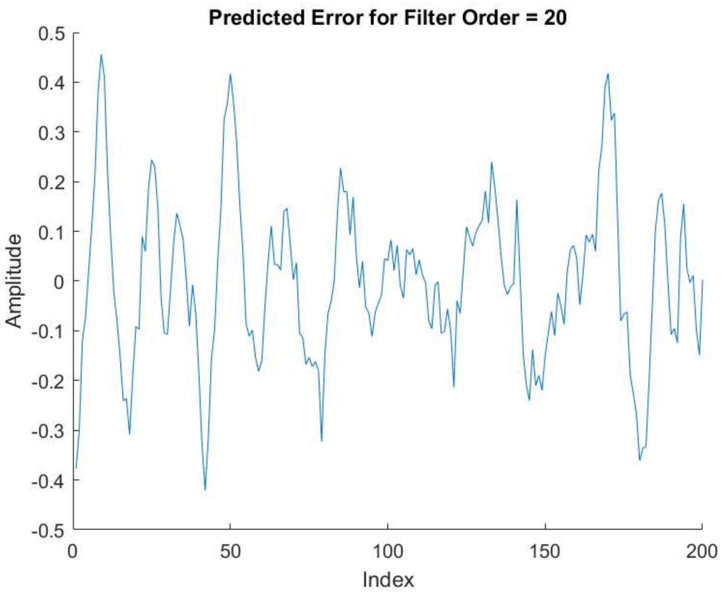

```
Mean squared error with filter order 15 = 0.034537
```






```
Mean squared error with filter order 20 = 0.02353
```






```
% Plotting the means squared error
figure
hold on
plot(filter_orders, mean_squared_error_arr, 'bo-', 'LineWidth', 2)
title('Mean Squared Error vs Filter Order')
xlabel('Filter Order')
ylabel('Mean Squared Error')
```

```
hold off
```


```
disp("It can be seen that as the filter order is increasing, the mean sqaured error is decreasing.");
```

```
It can be seen that as the filter order is increasing, the mean sqaured error is decreasing.
```

Functions

```
function y = random_prcoess(A, w0, n_arr)
    % Generate random process
    N = numel(n_arr);
    phi = 2*pi*rand();      % Uniformly distributed phase
    y = A*cos(n_arr.*w0+ones(1,N).*phi);
end

function w = wiener_filter(A, var, w, M, N)
    % Implementing Wiener-Hopf equation

    % This function Generate optimum Wiener filter weights by solving Wiener-Hopf matrix equation

    % Obtain the theoretical autocorrelation
    auto = [0.5*(abs(A)^2) + var, zeros(1, N-1)];
    for i=2:N
        auto(1, i) = 0.5*cos((i-1)*w)*(abs(A)^2);
    end

    % Obtain the theoretical crosscorrelation
    cross = [0.5*(abs(A)^2), zeros(1, N-1)];
    for i=2:N
        cross(1, i) = 0.5*cos((1-i)*w)*(abs(A)^2);
    end

    p = cross(N-(M-1):N);                % Obtain top M values of crosscorrelation
    toeplitz_row = auto(N-(M-1):N);      % Obtain top M values of autocorrelation

    R = toeplitz(toeplitz_row);          % Generate toeplitz matrix

    w = R\p.';                           % Obtain optimum Weiner-Hopf filter
end
```