*Homework3_1 *

Clear workspace

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

Programmers

Mohammad Mahdi Elyasi - 9823007

Moein Nasiri - 9823093

Homework1

Here we want to use diffrential equation and Z-Transform

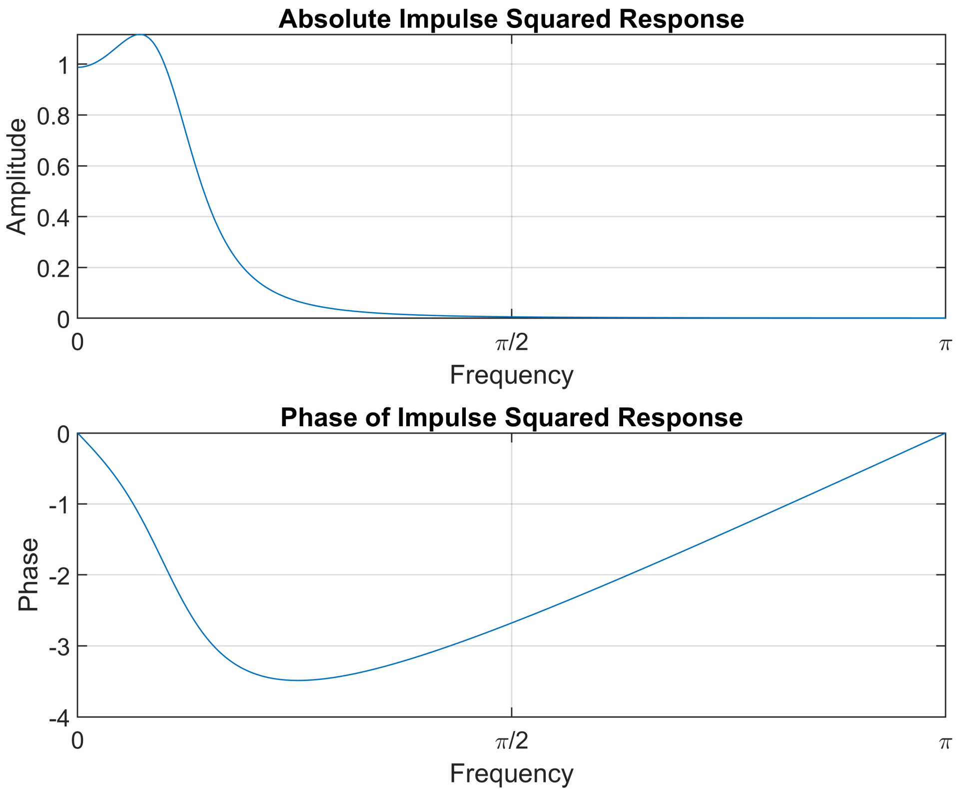
Here we declare some basic variables

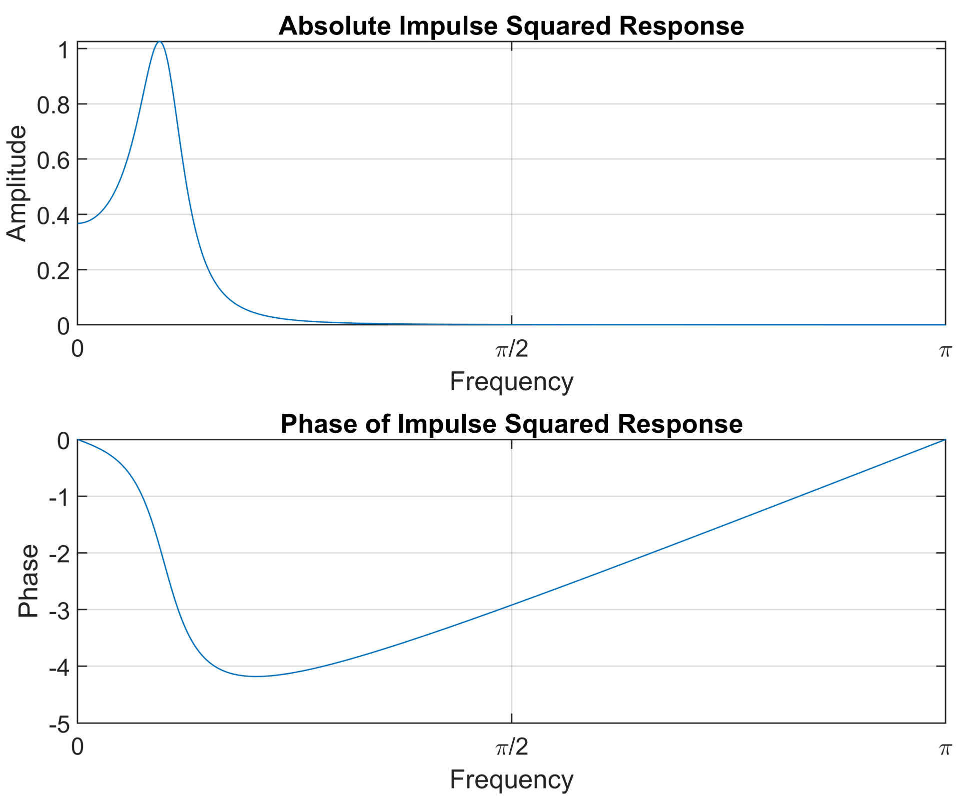
```
fs = 10000;
f0 = 500;
w0 = 2 * pi * f0 / fs;
R = [0.8 0.9 0.99];
```

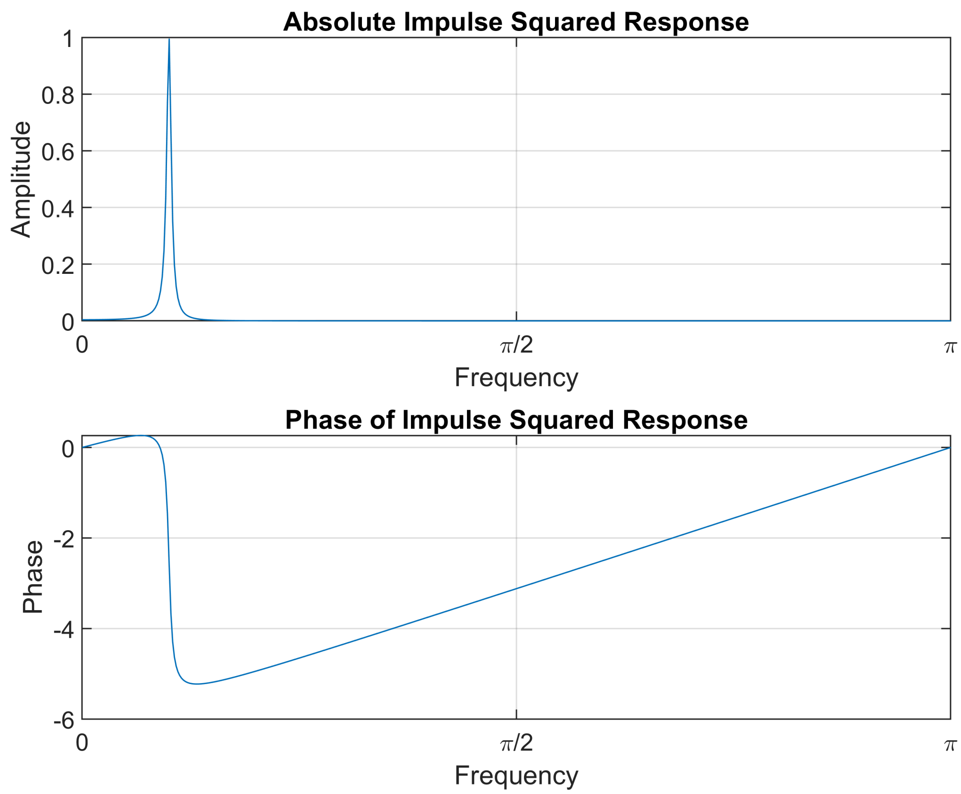
Here we want to plot H for different R and see how it responds.

```
for i = 1:3

    G = (1 - R(i)) * (1 - 2 * R(i) * cos(2 * w0) + R(i) ^ 2) ^ (0.5);
    w = linspace(0, pi, 500);
    H = freqz(G, [1 -2 * R(i) * cos(w0) R(i) ^ 2], w);
    figure('Name', 'Impulse Squared Response');
    subplot(2, 1, 1)
    plot(w, abs(H) .^ 2);
    title('Absolute Impulse Squared Response');
    xlabel('Frequency');
    ylabel('Amplitude');
    xlim([0 pi]);
    xticks([0, pi / 2, pi]);
    xticklabels({'0', '\pi/2', '\pi'});
    grid on;
    subplot(2, 1, 2)
    plot(w, angle(H) * 2);
    title('Phase of Impulse Squared Response');
    xlabel('Frequency');
    ylabel('Phase');
    xlim([0 pi]);
    xticks([0, pi / 2, pi]);
    xticklabels({'0', '\pi/2', '\pi'});
    grid on;
end
```


Absolute Impulse Squared Response
Amplitude
1
0.8
0.6
0.4
0.2
0
0
π/2
π
Frequency
Phase of Impulse Squared Response
Phase
0
-1
-2
-3
-4
0
π/2
π
Frequency


Absolute Impulse Squared Response
Amplitude
1
0.8
0.6
0.4
0.2
0
0
π/2
π
Frequency
Phase of Impulse Squared Response
Phase
0
-1
-2
-3
-4
-5
0
π/2
π
Frequency



Homework2

In this task we want to show how impulse input responds to our filter by 3 different methods

Firstly, we declare essential variables

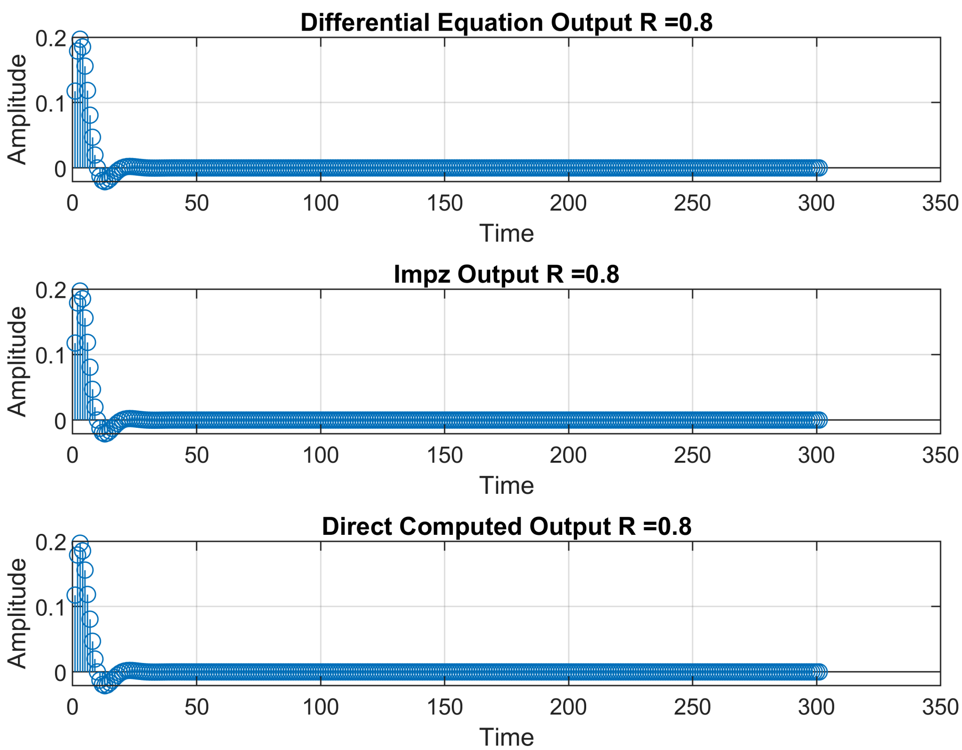
```
for j = 1:3
    G = (1 - R(j)) * (1 - 2 * R(j) * cos(2 * w0) + R(j) ^ 2) ^ (0.5);
    x = [1 zeros(1, 300)];
    w1 = 0;
    w2 = 0;
    y = zeros(1, 301);
```

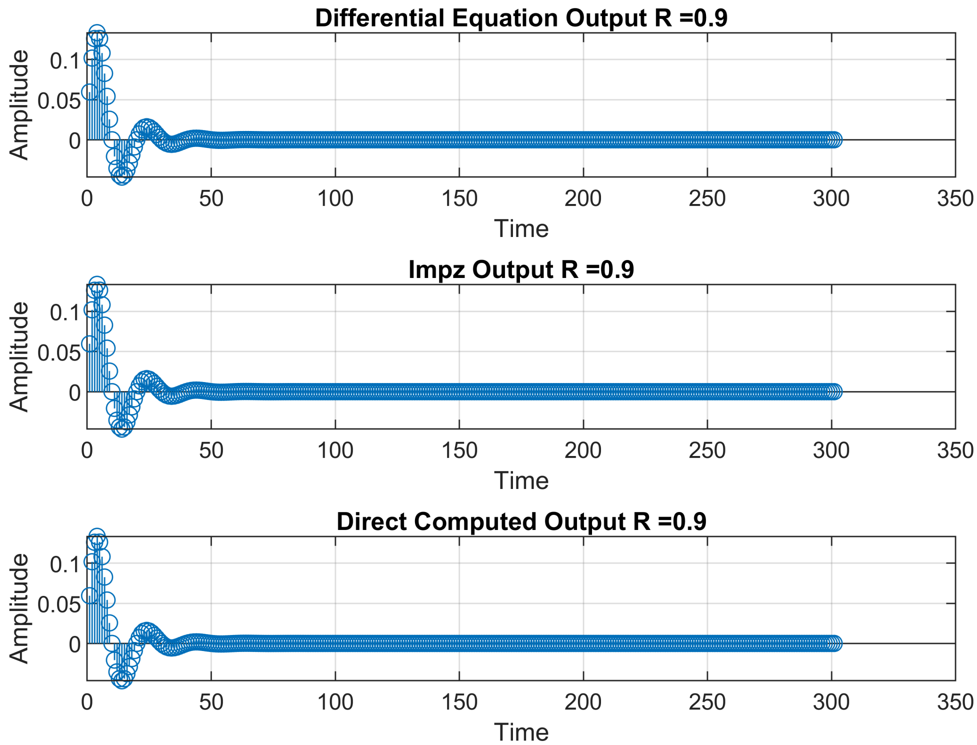
Here we show output with three different methods for different R and plot them

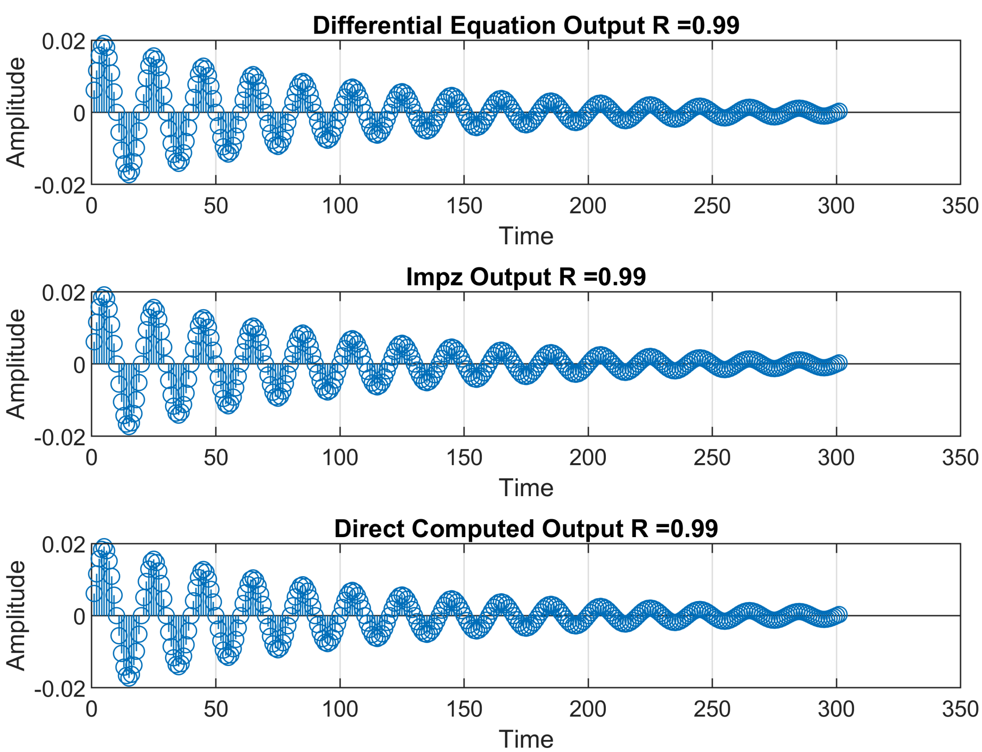
```
    for i = 1:301
        y(i) = 2 * R(j) * cos(w0) * w1 -R(j) ^ 2 * w2 + G * x(i);
        w2 = w1;
        w1 = y(i);
    end

    figure('Name', 'Output');
    subplot(3, 1, 1)
    stem(y);
    title(strcat('Differential Equation Output R =', string(R(j))));
    xlabel('Time');
    ylabel('Amplitude');
    grid on;
```

```
    subplot(3, 1, 2)
    [h, n] = impz(G, [1 -2 * R(j) * cos(w0) R(j) ^ 2], 301);
    stem(h)
    title(strcat('Impz Output R =', string(R(j))));
    xlabel('Time');
    ylabel('Amplitude');
    grid on;
    subplot(3, 1, 3)
    n = 0:300;
    h = G / sin(w0) * R(j) .^ n .* sin(w0 * n + w0);
    stem(h)
    title(strcat('Direct Computed Output R =', string(R(j))));
    xlabel('Time');
    ylabel('Amplitude');
    grid on;
end
```




Differential Equation Output R =0.9
Impz Output R =0.9
Direct Computed Output R =0.9
Amplitude
Time
0.1
0.05
0
0
50
100
150
200
250
300
350


Differential Equation Output R =0.99
Impz Output R =0.99
Direct Computed Output R =0.99
Amplitude
Time
0.02
0
-0.02
0
50
100
150
200
250
300
350

Homework3

As we see R=0.99 filter is similar to our input, but it has delay, because our filter(differential) needs to save 4 datas before computing the answer

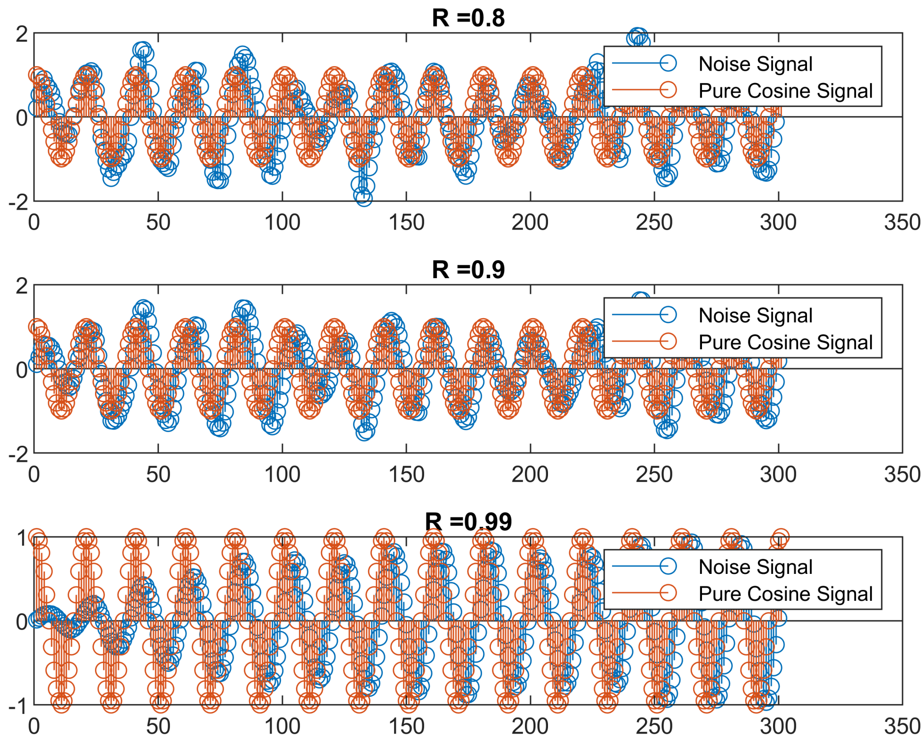
```
figure('Name', 'Output');
v = 1 * randn(1, 301);
n = 0:300;
x = v + cos(w0 * n);

for j = 1:3
    G = (1 - R(j)) * (1 - 2 * R(j) * cos(2 * w0) + R(j) ^ 2) ^ (0.5);
    w1 = 0;
    w2 = 0;
    y = zeros(1, 301);
```

Here we show output by differential equation and plot it

```
    for i = 1:301
        y(i) = 2 * R(j) * cos(w0) * w1 -R(j) ^ 2 * w2 + G * x(i);
        w2 = w1;
        w1 = y(i);
    end

    subplot(3, 1, j)
    stem(y);
    hold on;
    stem(cos(w0 * n))
    title(strcat('R =', string(R(j))));
    legend('Noise Signal', 'Pure Cosine Signal');
end
```



Homework4

```
figure('Name', 'Output');
v = 1 * randn(1, 301);
n = 0:300;
x = v;

for j = 1:3
    G = (1 - R(j)) * (1 - 2 * R(j) * cos(2 * w0) + R(j) ^ 2) ^ (0.5);
    w1 = 0;
    w2 = 0;
    y = zeros(1, 301);
```

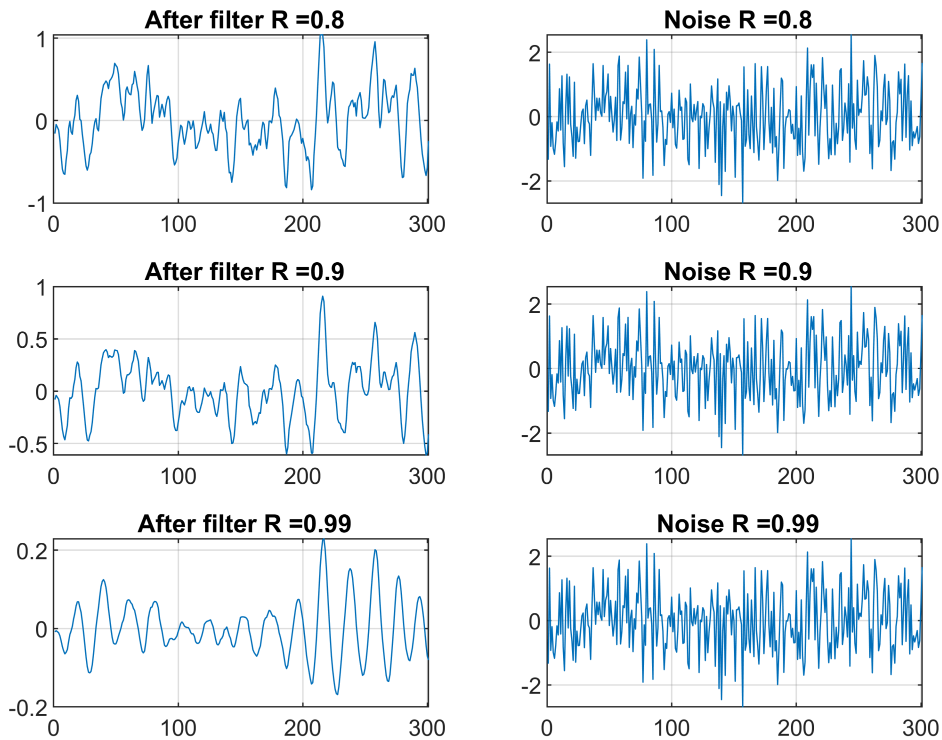
- PSD Noise is DC and also PSD for delta func. is DC.
- This is one of the reasons the noise after filter is cosine.
- Another reason is, this filter is kind of a band-pass filter and lets some frequency pass

Here we show output by differential equation and plot it

```
    for i = 1:301
        y(i) = 2 * R(j) * cos(w0) * w1 -R(j) ^ 2 * w2 + G * x(i);
        w2 = w1;
        w1 = y(i);
    end

    subplot(3, 2, 2 * j - 1)
```

```
    plot(y);
    title(strcat(' After filter R =', string(R(j))));
    hold on;
    grid on;
    subplot(3, 2, 2 * j)
    plot(v)
    title(strcat(' Noise R =', string(R(j))));
    grid on;
end
```



Homework5

In this homework we want to see how the formula is valid

Our purpose is how variance of noise before and after filtering is related to square of filter

```
v = 1 * randn(1, 1001);
n = 0:1000;
x = v;

for j = 1:3
    G = (1 - R(j)) * (1 - 2 * R(j) * cos(2 * w0) + R(j) ^ 2) ^ (0.5);
    w1 = 0;
    w2 = 0;
    y = zeros(1, 1001);

    for i = 1:1001
        y(i) = 2 * R(j) * cos(w0) * w1 -R(j) ^ 2 * w2 + G * x(i);
```

```
        w2 = w1;
        w1 = y(i);
    end

    h = G / sin(w0) * R(j) .^ n .* sin(w0 * n + w0);
    disp(strcat('NRR for R =', string(R(j))))
    disp('Sum of H^2 signal:')
    a = sum(h .^ 2)
    disp(strcat('NRR for R =', string(R(j))))
    disp('Using the variance of signal:')
    va_y = var(y);
    va_x = var(x);
    a = va_y / va_x
    disp(strcat('NRR for R =', string(R(j))))
    disp('Using formula:')
    answer = G ^ 2 / (2 * sin(w0) ^ 2) * ((1 / (1 - R(j) ^ 2) - (cos(2 * w0) - R(j) ^ 2) / (1 -
end
```

```
NRR for R =0.8
Sum of H^2 signal:
a = 0.1683
NRR for R =0.8
Using the variance of signal:
a = 0.1851
NRR for R =0.8
Using formula:
answer = 0.1683
NRR for R =0.9
Sum of H^2 signal:
a = 0.0975
NRR for R =0.9
Using the variance of signal:
a = 0.1107
NRR for R =0.9
Using formula:
answer = 0.0975
NRR for R =0.99
Sum of H^2 signal:
a = 0.0100
NRR for R =0.99
Using the variance of signal:
a = 0.0136
NRR for R =0.99
Using formula:
answer = 0.0100
```