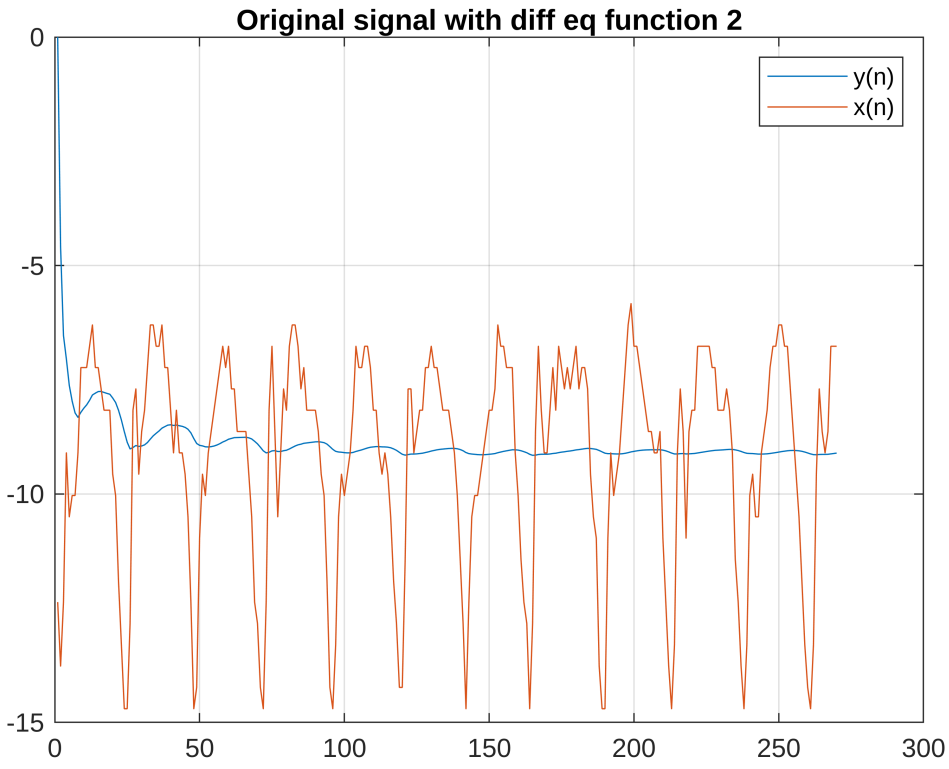
HW 2 - Matlab Exercise 2 - parts 5-8

Module 2 consists of 8 sections. loading data as we did before is where we begin

```
data = load('../HMP_Dataset/Climb_stairs/Accelerometer-2011-03-24-10-24-39-climb_stairs-f1.txt');
% loading in the data from the csv matrix
converted_data = convert(data);
x = converted_data(:,1);
y = converted_data(:,2);
z = converted_data(:,3);
```

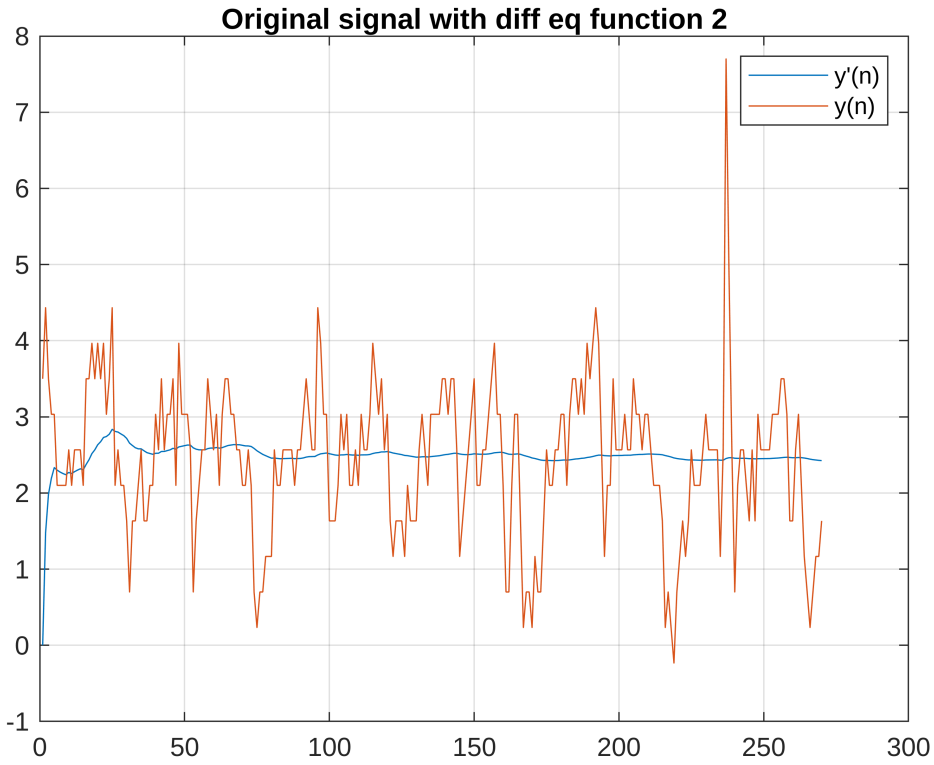
5 Recursive system 2

```
figure()
diff2_x = [diffeq_2(x),x];
plot(diff2_x)
title('Original signal with diff eq function 2')
legend("y(n)","x(n)")
grid on
```

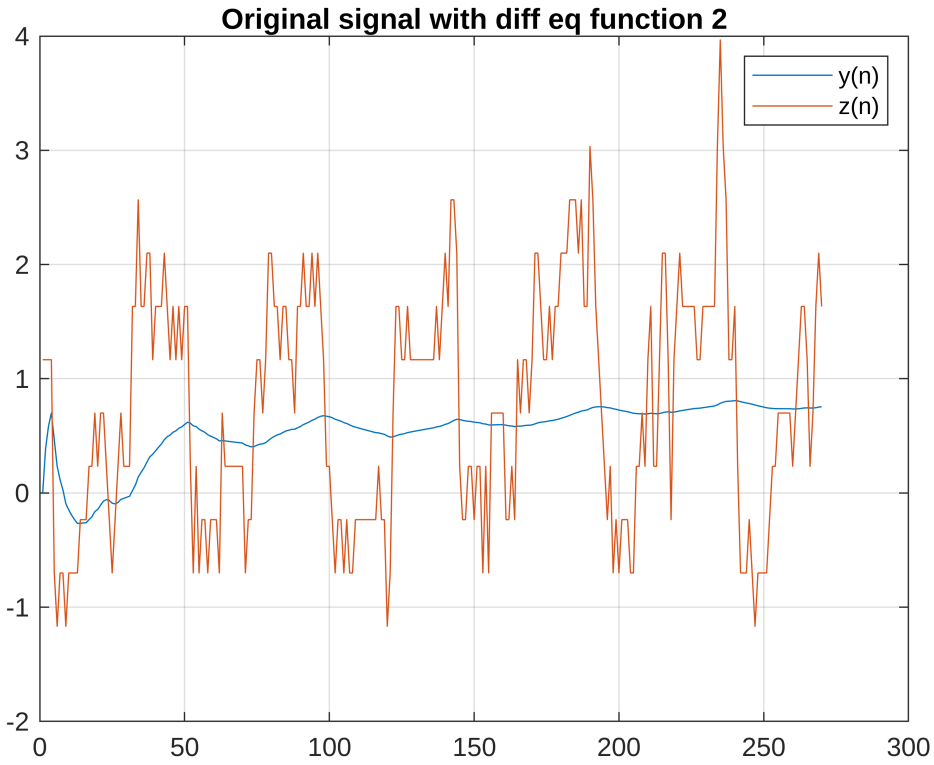



```
figure()
diff2_y = [diffeq_2(y),y];
plot(diff2_y)
title('Original signal with diff eq function 2')
```

```
legend("y'(n)","y(n)")
grid on
```



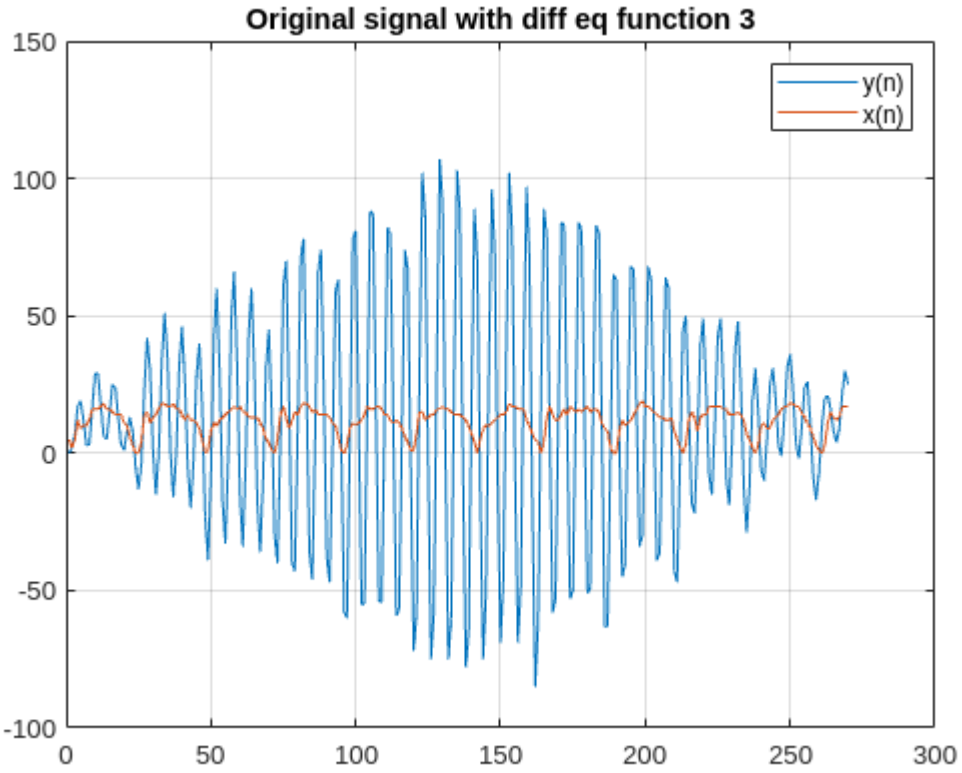

```
figure()
diff2_z = [diffeq_2(z),z];
plot(diff2_z)
title('Original signal with diff eq function 2')
legend("y(n)","z(n)")
grid on
```



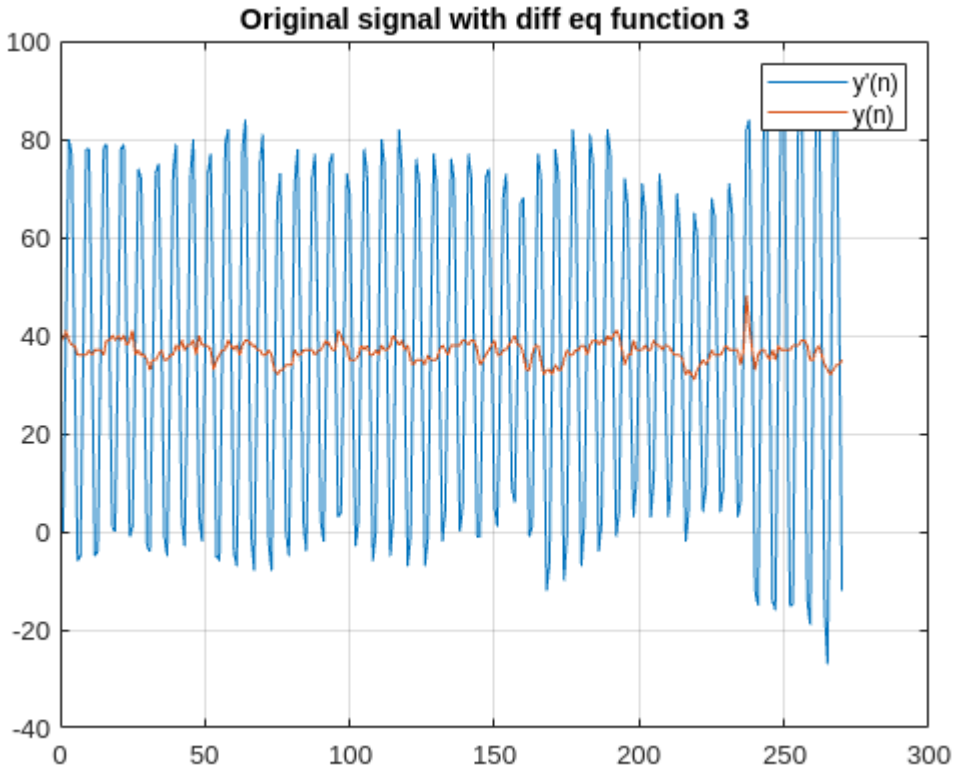
This system is computing a cumulative average. I cannot remember the term mentioned in class, but we can see that the last value calculated is one thatcounts toward the sum the least overall. same effect as having an average that you add one more value to, it is by definition a single value more than the average had increased by prior to this addition. the y term being multiplied by n shows that it must be first multiplied by the total number of elements that had been included thusfar.

6 Recursive system 3

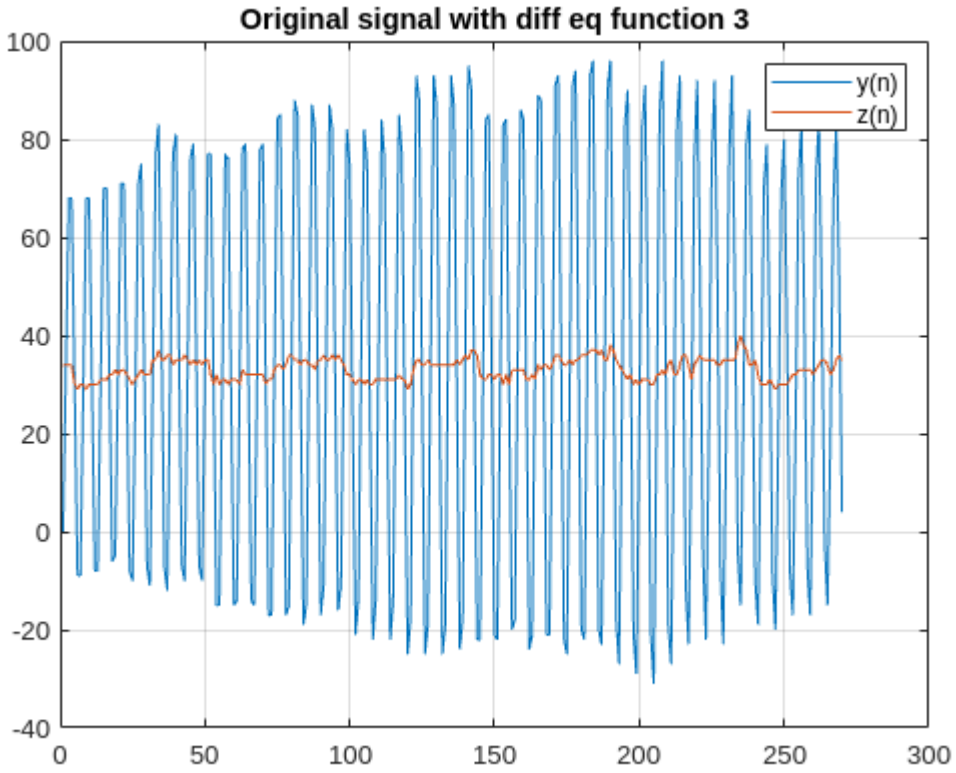
```
x = data(:,1);
y = data(:,2);
z = data(:,3);
figure()
diff3_x = [diffeq_3(x),x];
plot(diff3_x)
title('Original signal with diff eq function 3')
legend("y(n)","x(n)")
grid on
```




```
figure()
diff3_y = [diffeq_3(y),y];
plot(diff3_y)
title('Original signal with diff eq function 3')
legend("y'(n)","y(n)")
grid on
```



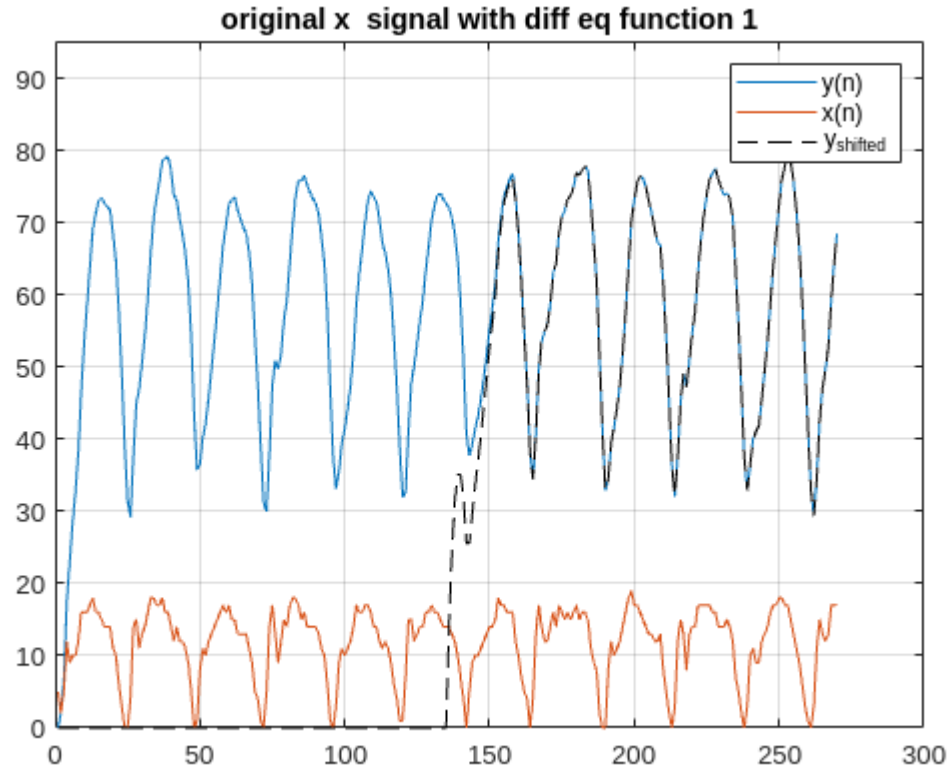

```
figure()
diff3_z = [diffeq_3(z),z];
plot(diff3_z)
title('Original signal with diff eq function 3')
legend("y(n)","z(n)")
grid on
```



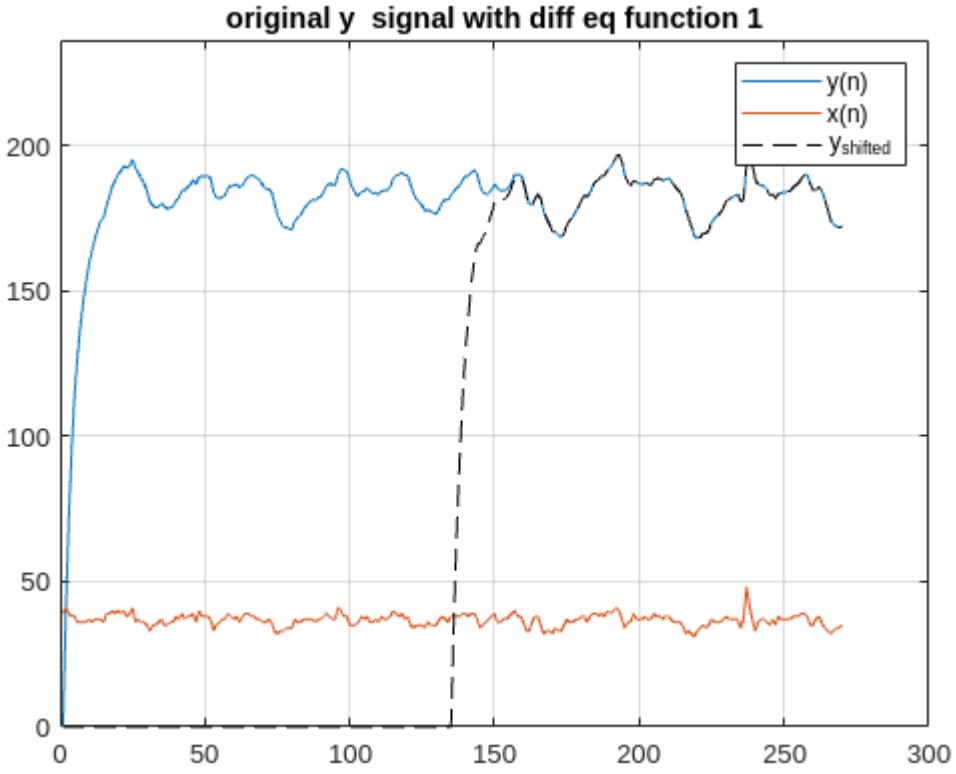
I know these should be LTI systems, but I am not sure exactly if my tests prove that.

recursive shift for system 1 on x data and converted x data to see differences and test LTI Note** none of the below was asked for, but I thought it would be cool to try a test for LTI in this way.

```
x = data(:,1);
y = data(:,2);
z = data(:,3);
figure
lti_test1(x,1,'original x ')
```

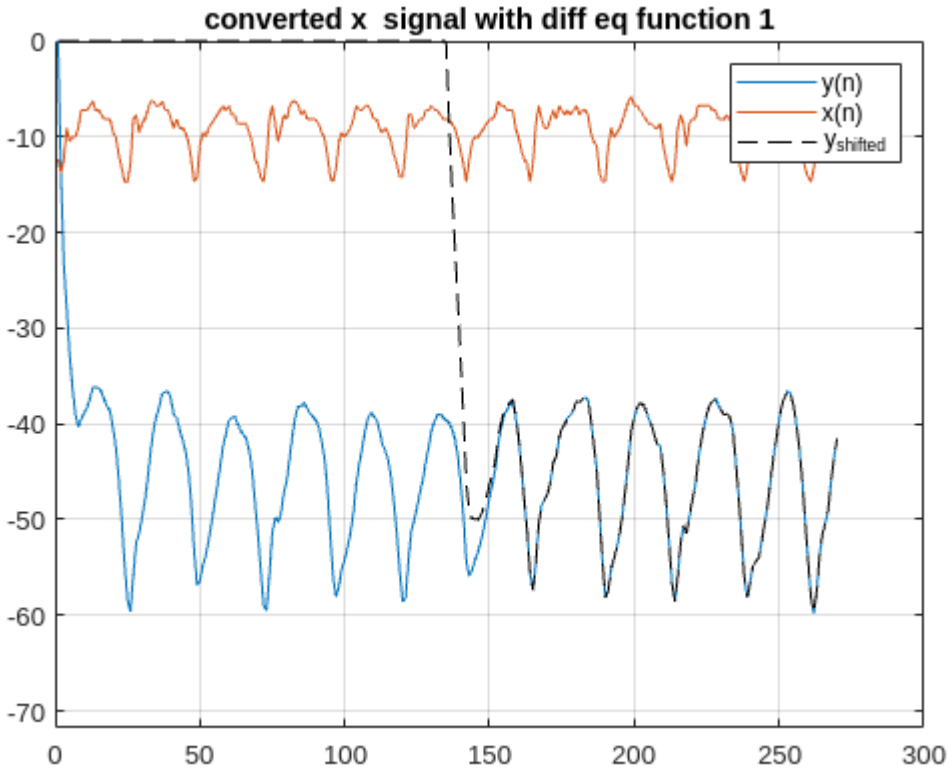



```
figure
lti_test1(y,1,'original y ')
```

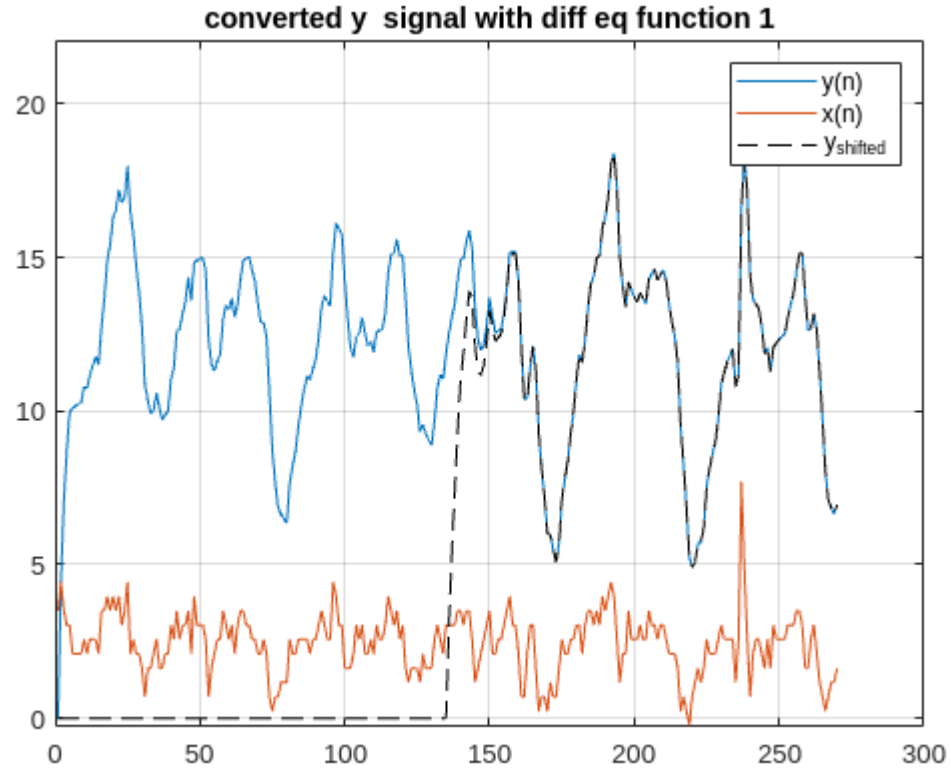


```
figure
lti_test1(z,1,'original z ')
```

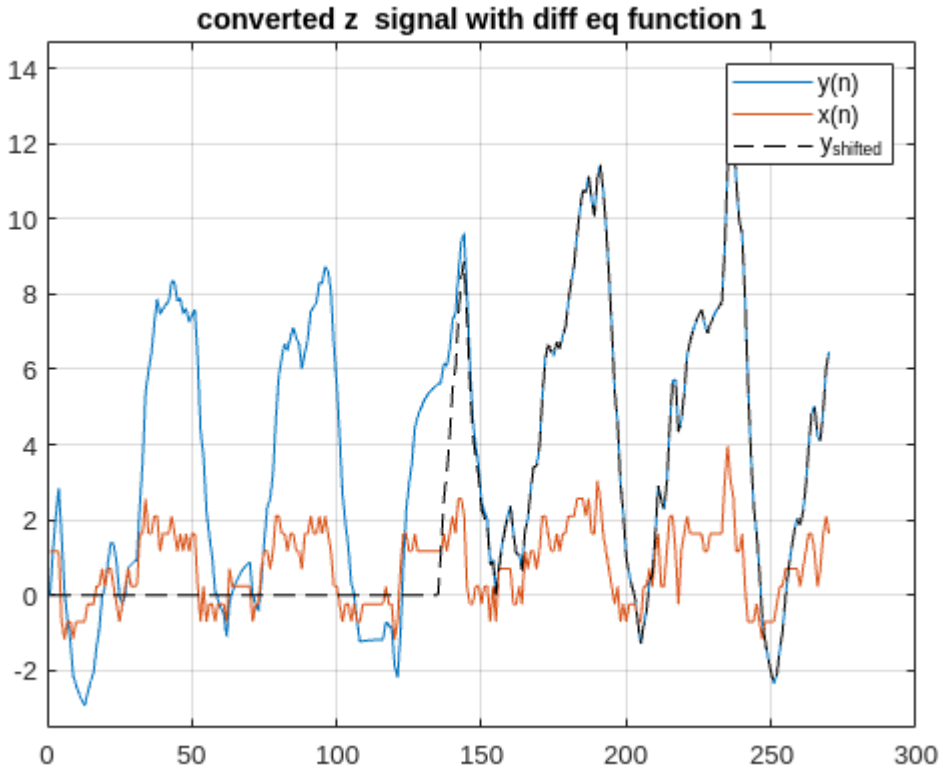

```
x = converted_data(:,1);
y = converted_data(:,2);
z = converted_data(:,3);

figure
lti_test1(x,1,'converted x ')
```




```
figure
lti_test1(y,1,'converted y ')
```

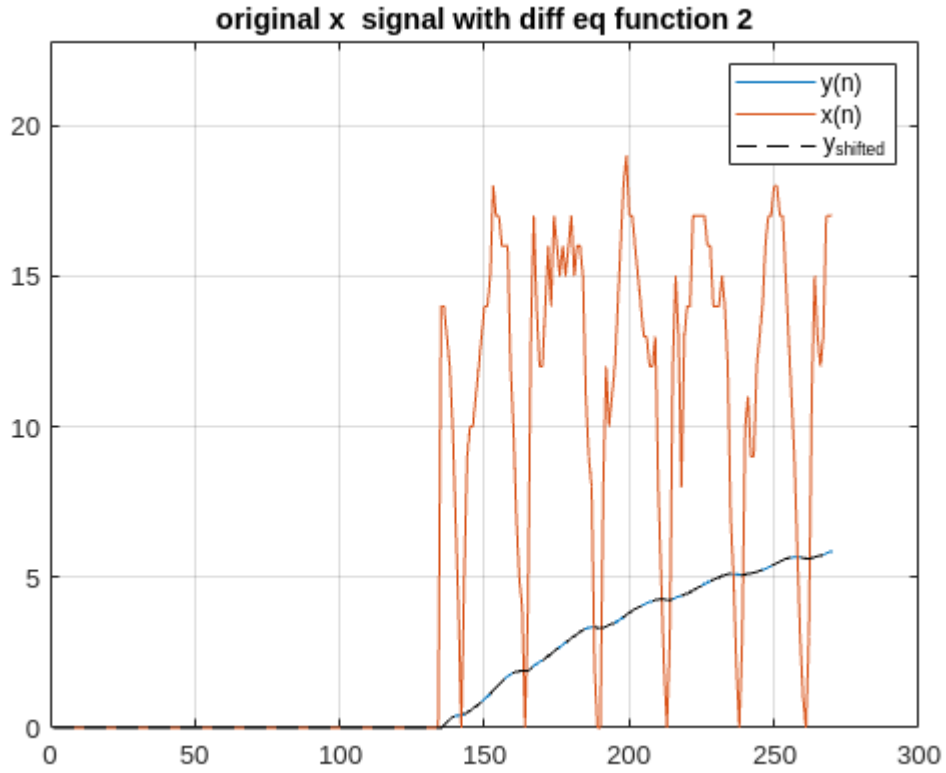


```
figure
lti_test1(z,1,'converted z ')
```

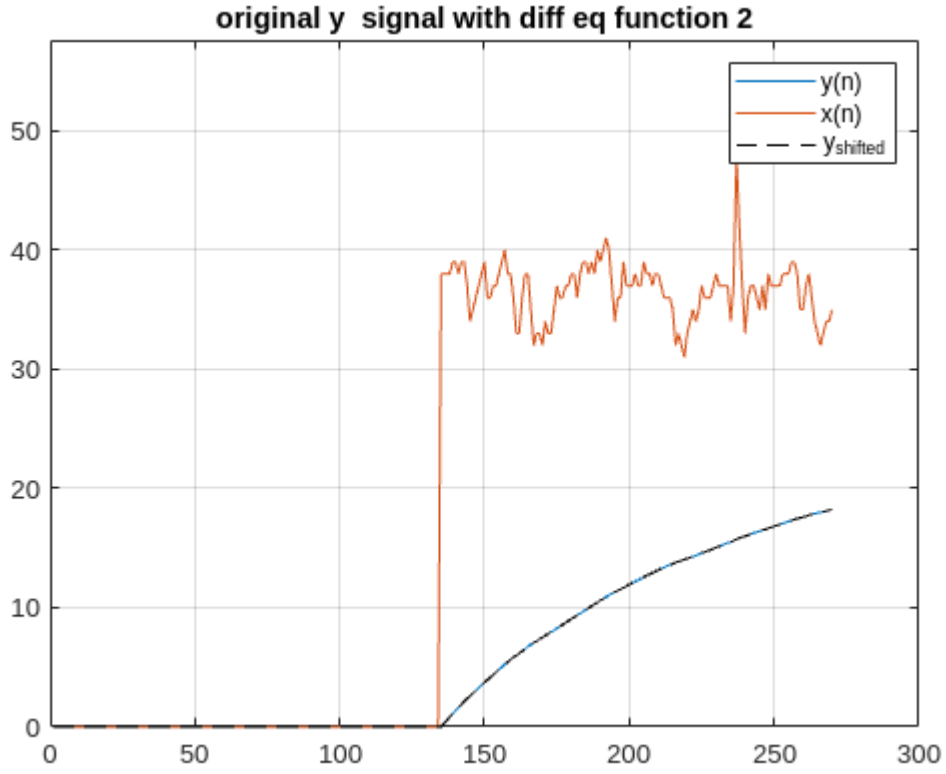


recursive shift for system 2 on x data and converted x data to see differences and test LTI

```
x = data(:,1);
y = data(:,2);
z = data(:,3);
figure
lti_test2(x,1,'original x ')
```

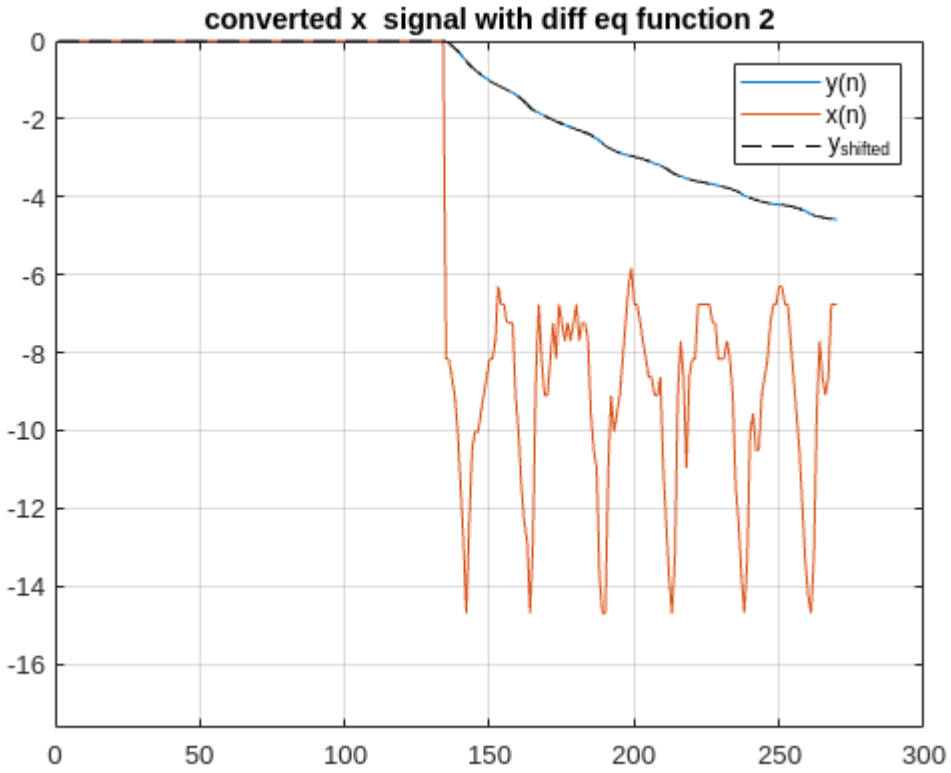



```
figure
lti_test2(y,1,'original y ')
```

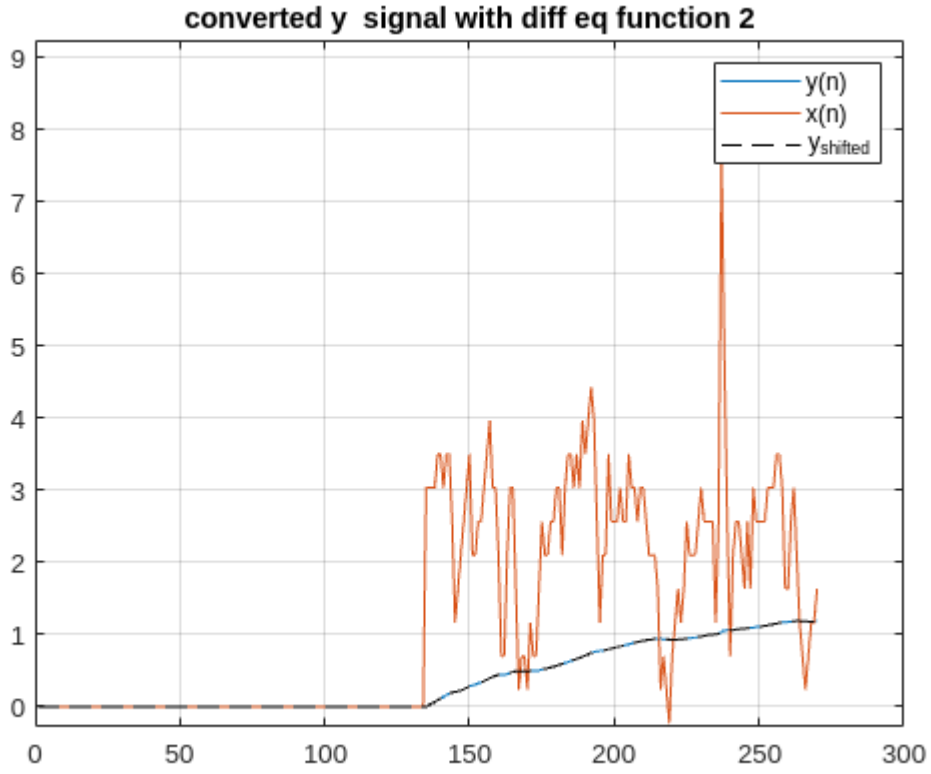


```
figure
lti_test2(z,1,'original z ')
```

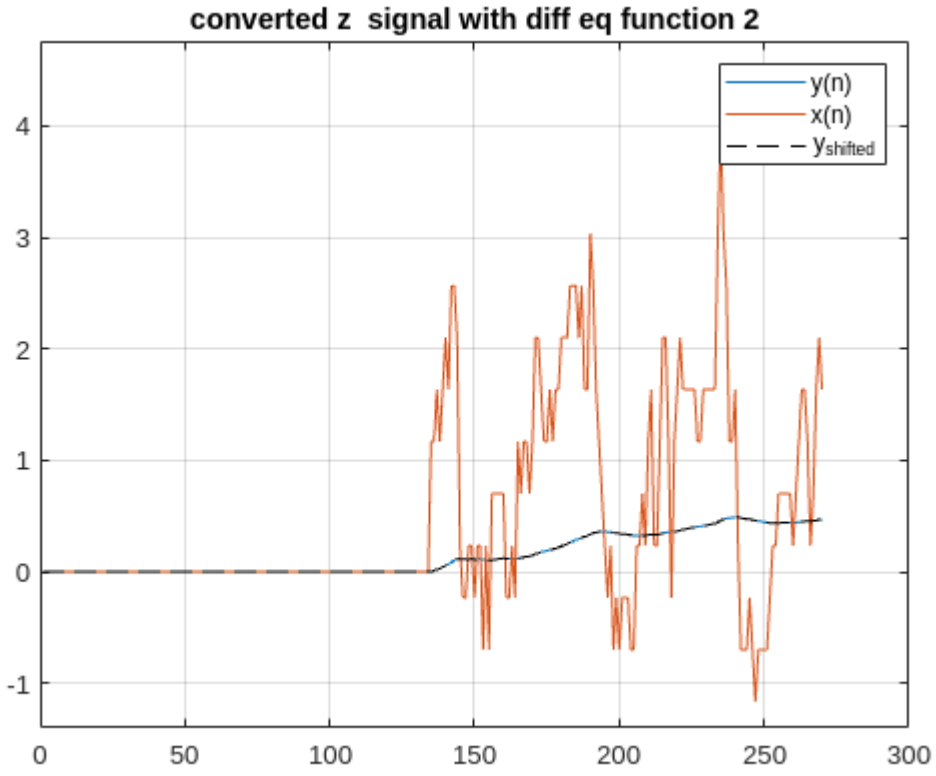

```
x = converted_data(:,1);
y = converted_data(:,2);
z = converted_data(:,3);

figure
lti_test2(x,1,'converted x ')
```




```
figure
lti_test2(y,1,'converted y ')
```

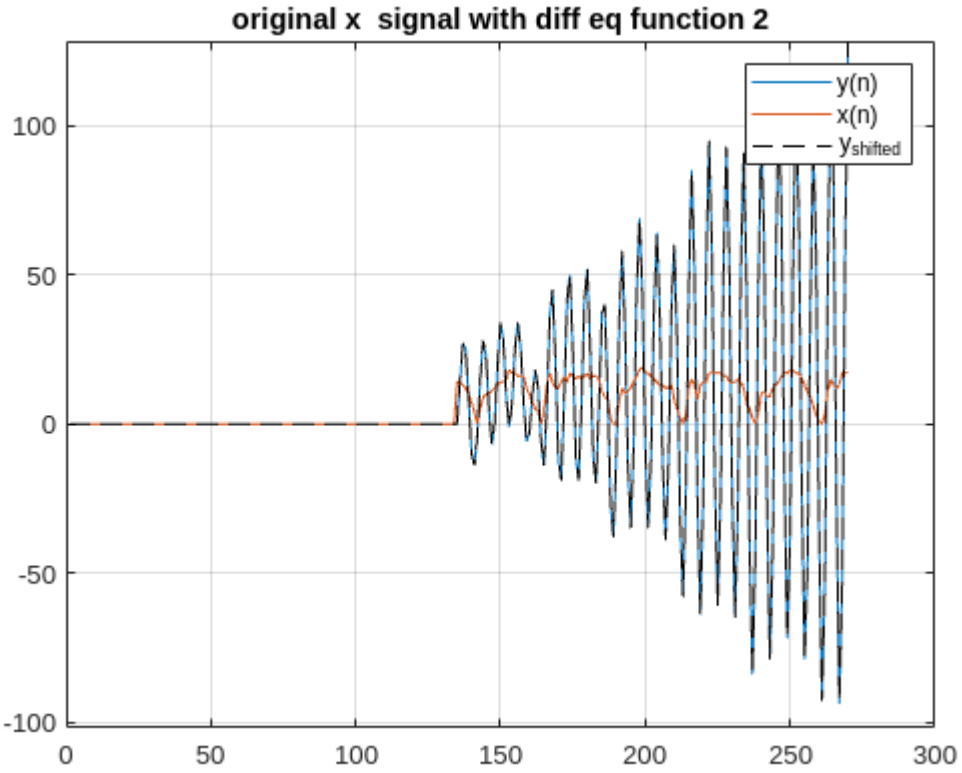


```
figure
lti_test2(z,1,'converted z ')
```

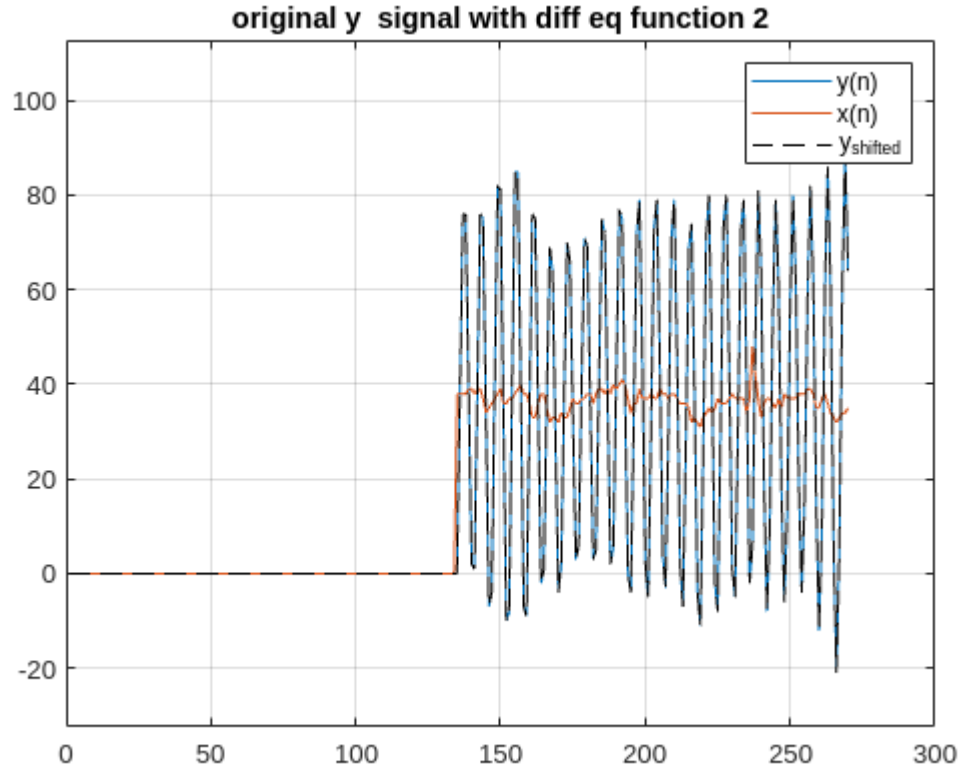


recursive shift for system 3 on x data and converted x data to see differences and test LTI

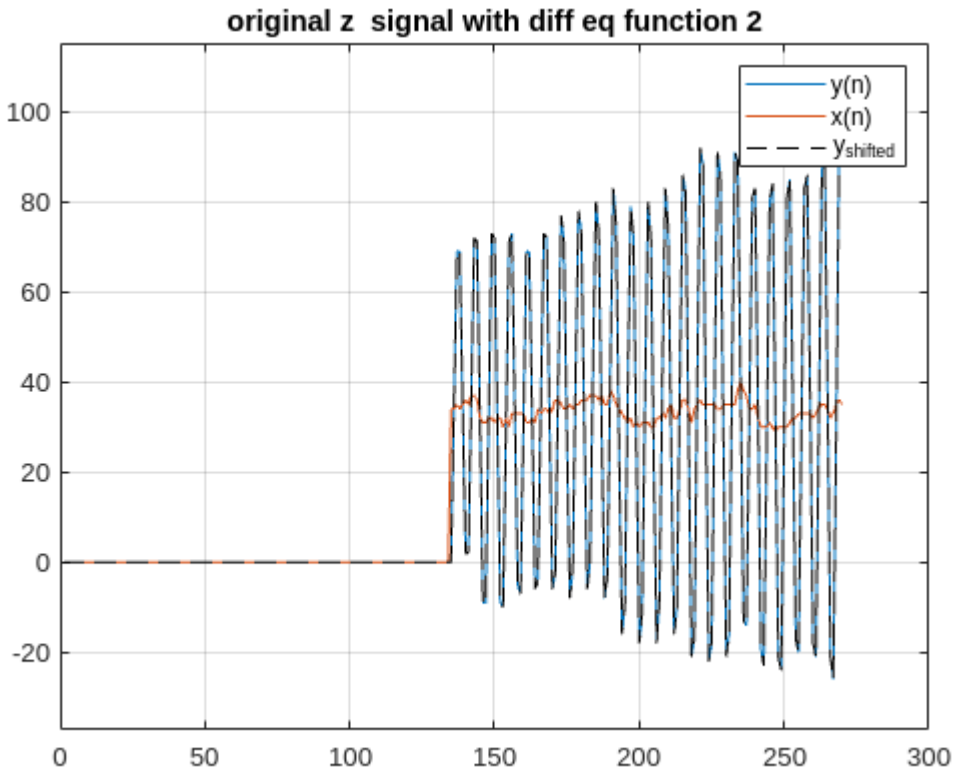
```
x = data(:,1);
y = data(:,2);
z = data(:,3);
figure
lti_test3(x,1,'original x ')
```



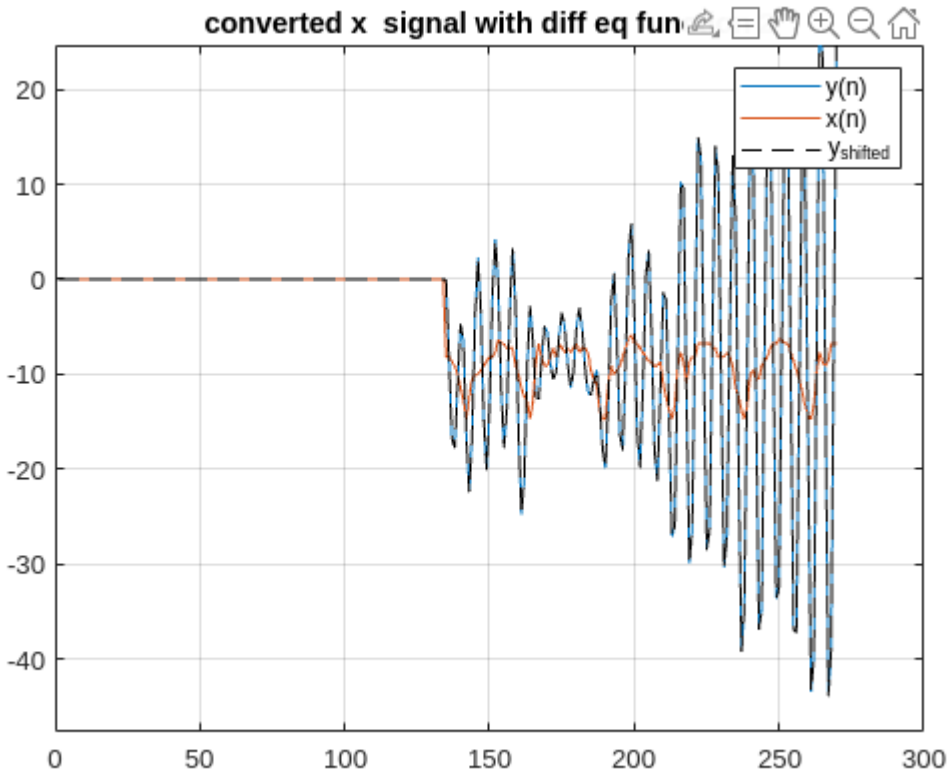

```
figure
lti_test3(y,1,'original y ')
```



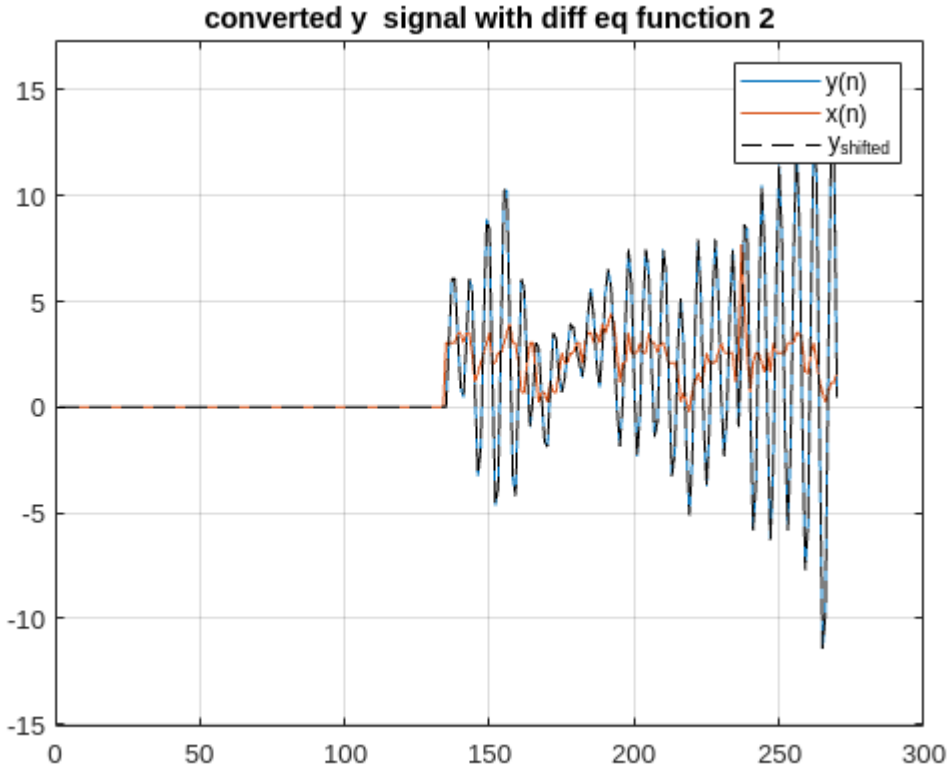
```
figure
lti_test3(z,1,'original z ')
```




```
x = converted_data(:,1);
y = converted_data(:,2);
z = converted_data(:,3);

figure
lti_test3(x,1,'converted x ')
```



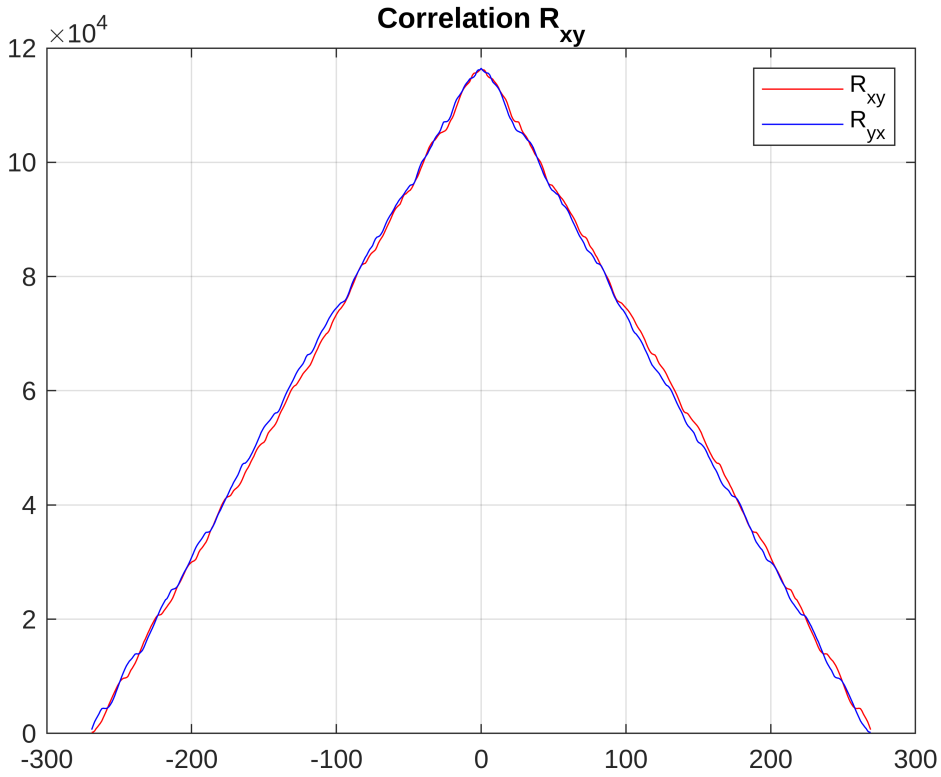

```
figure
lti_test3(y,1,'converted y ')
```



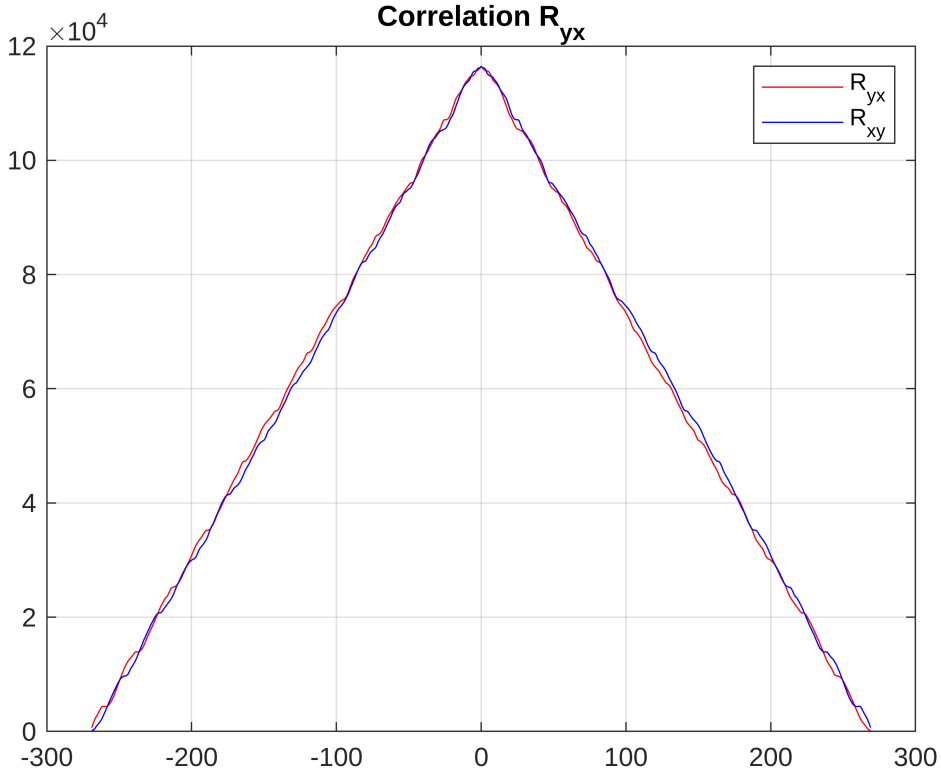
```
figure
lti_test3(z,1,'converted z ')
```

7 crosscorrelations of original data

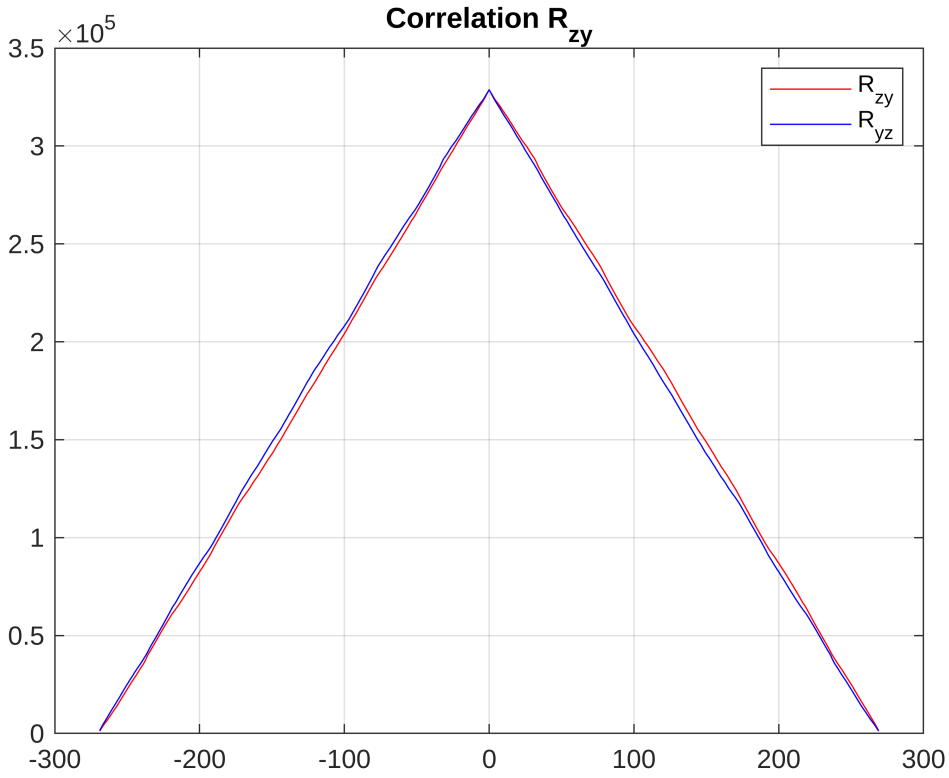
```
x = data(:,1);
y = data(:,2);
z = data(:,3);
cross_corr(x,y,'xy')
```



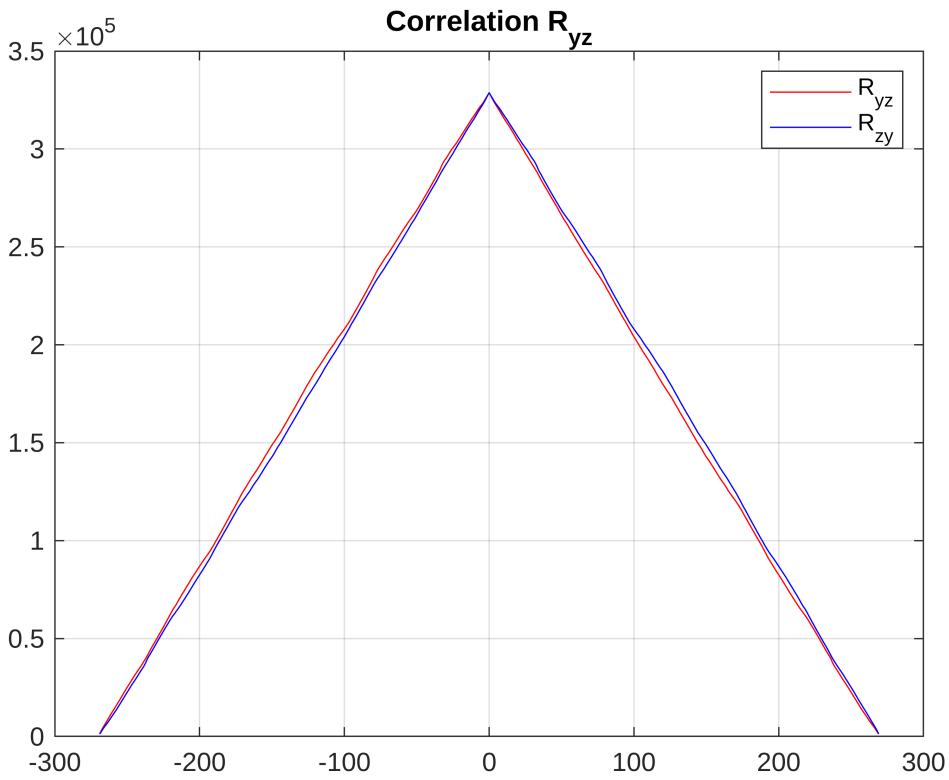

```
cross_corr(y,x,'yx')
```



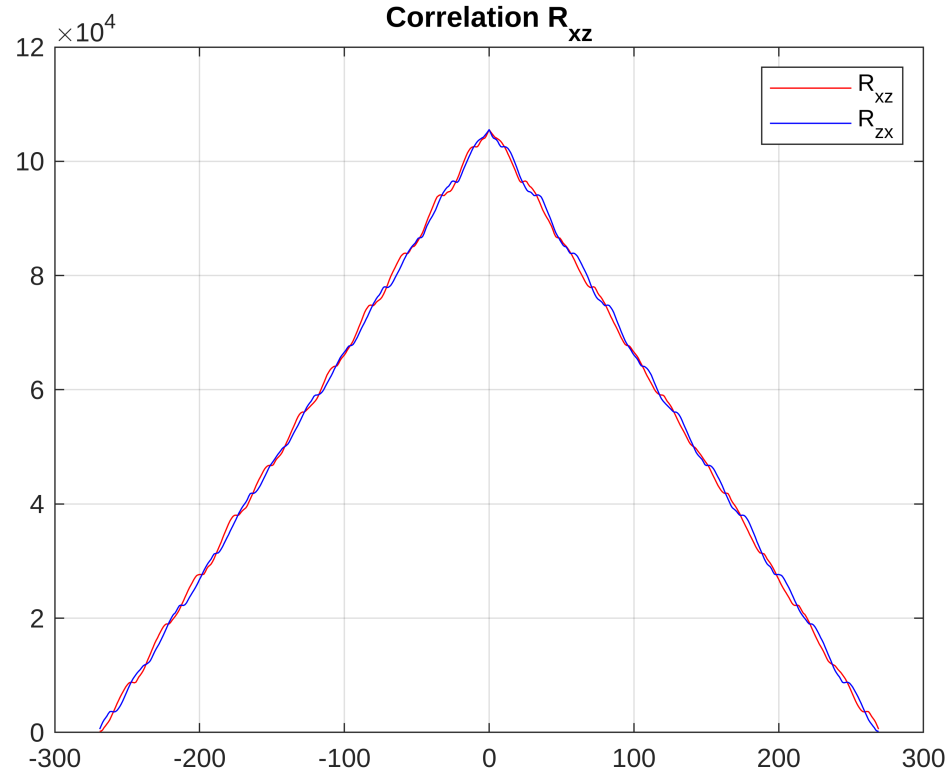
```
cross_corr(z,y,'zy')
```



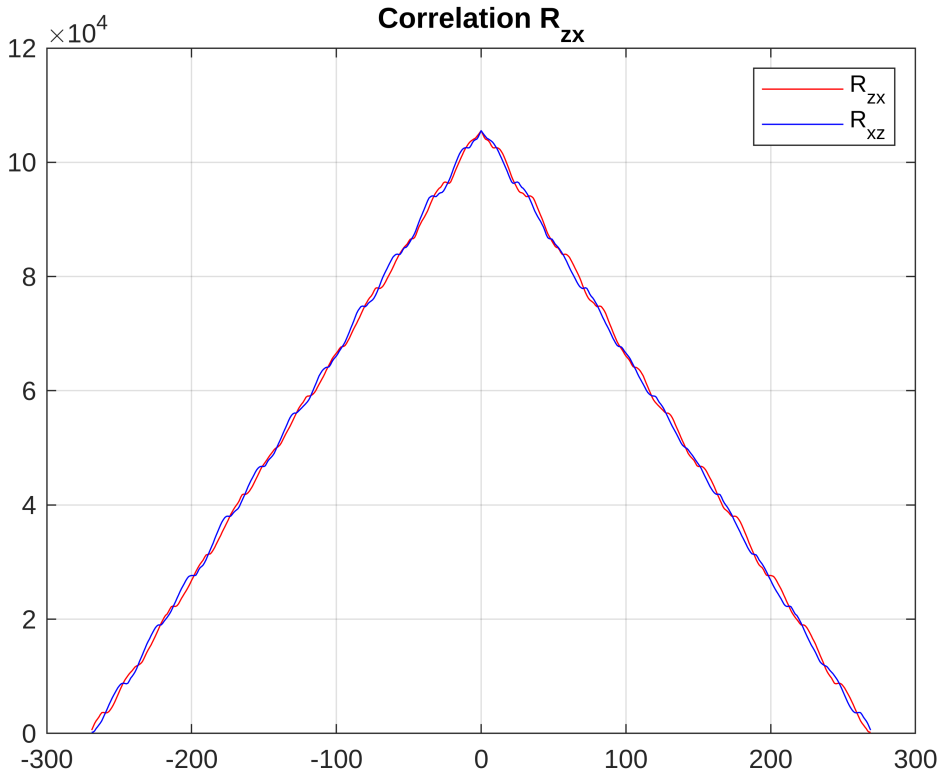

```
cross_corr(y,z,'yz')
```




```
cross_corr(x,z,'xz')
```

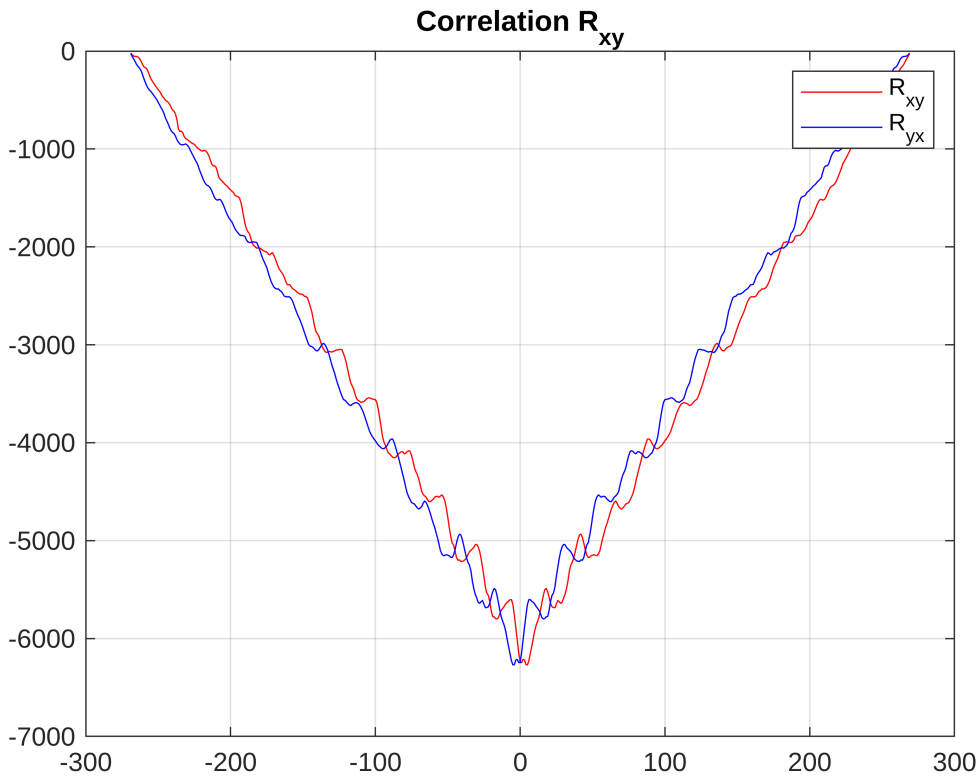


```
cross_corr(z,x,'zx')
```

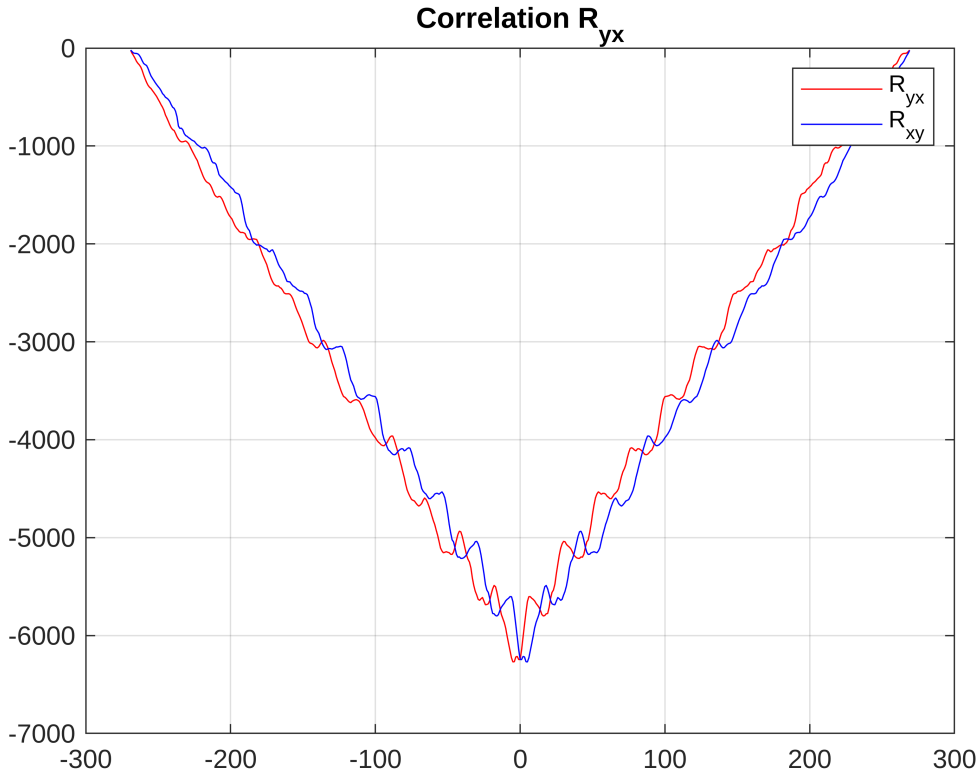


crosscorrelations of converted data

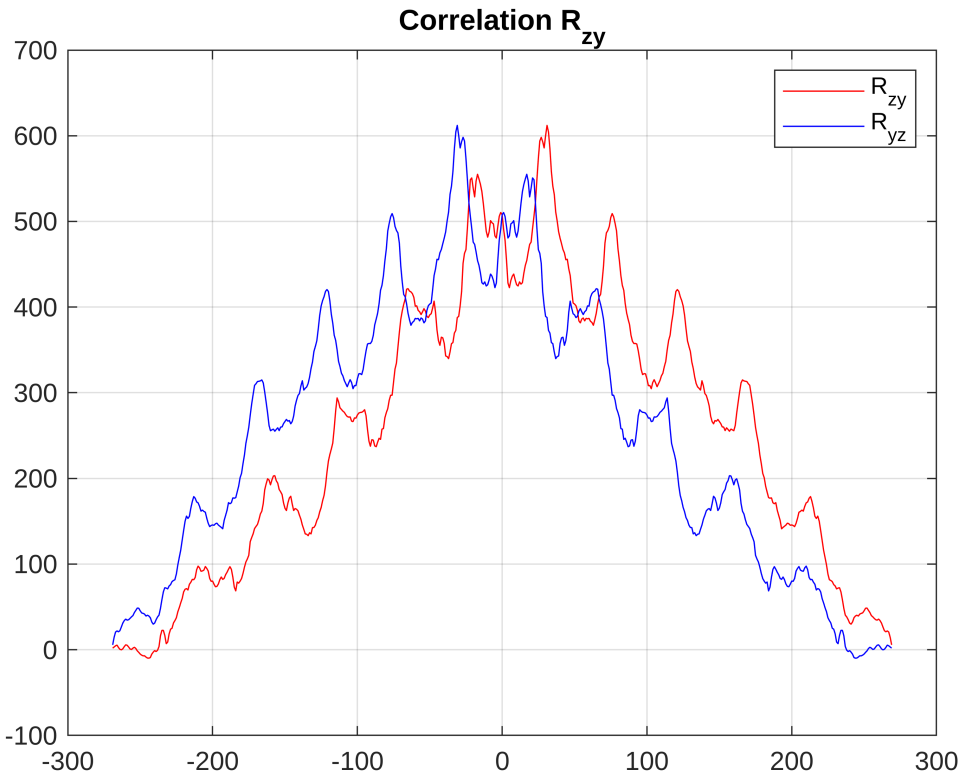
```
x = converted_data(:,1);
y = converted_data(:,2);
z = converted_data(:,3);
cross_corr(x,y,'xy')
```



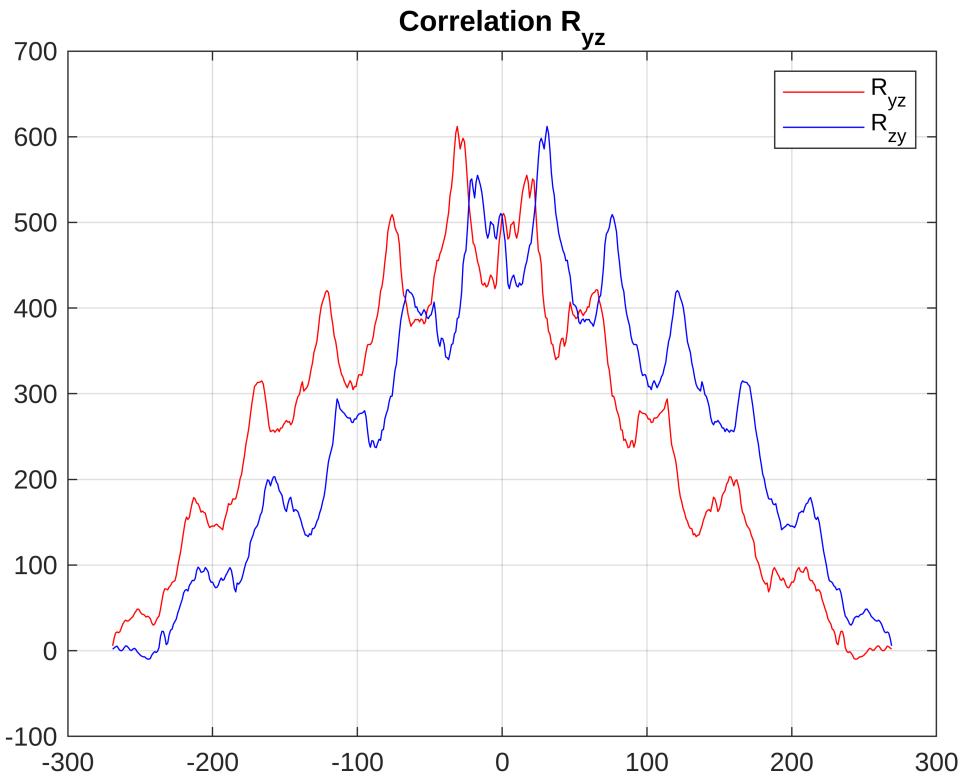

```
cross_corr(y,x,'yx')
```



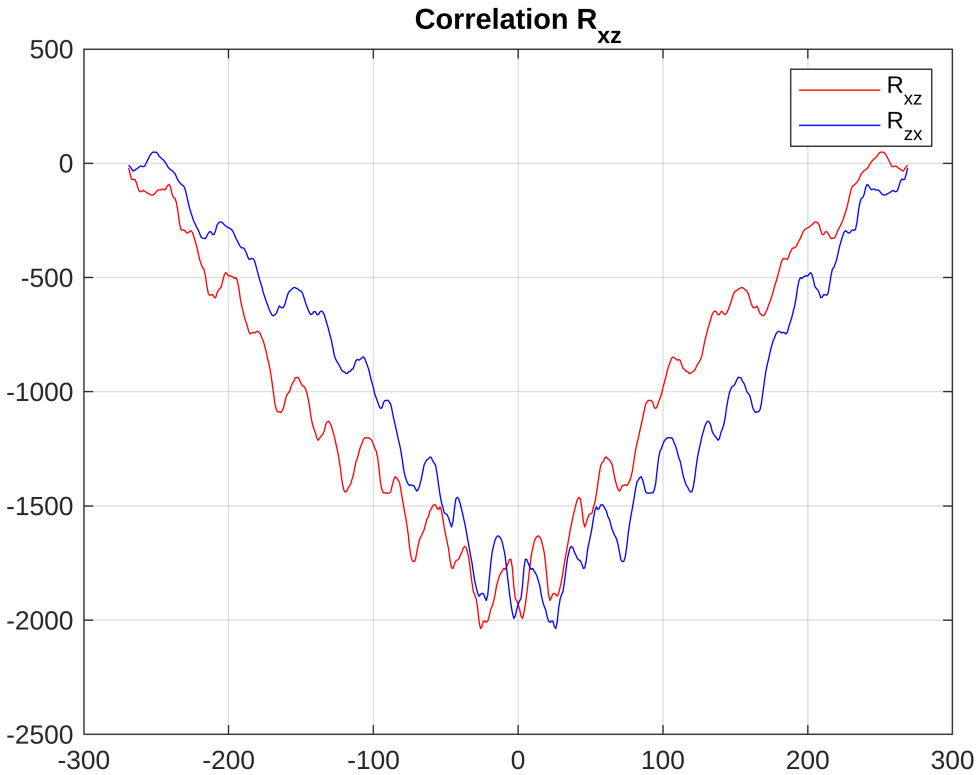
```
cross_corr(z,y,'zy')
```



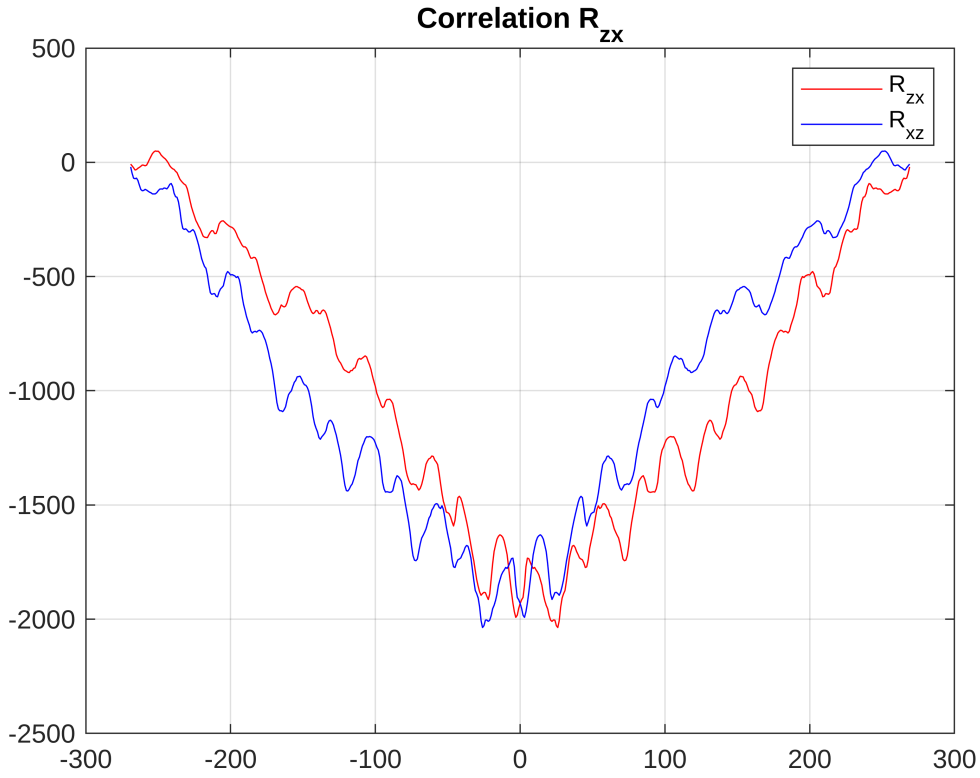

```
cross_corr(y,z,'yz')
```




```
cross_corr(x,z,'xz')
```



```
cross_corr(z,x,'zx')
```



Problem 2.65 from text parts a-d

```
clear
clc
a = [1, 1, 1, 1, 1, -1,-1, 1, 1, -1, 1, -1, 1];
variance = [.01,.1, 1];
delay = 20;
ya = zeros(1,199);
x_a = [zeros(1,delay),a,zeros(1,delay)];
```

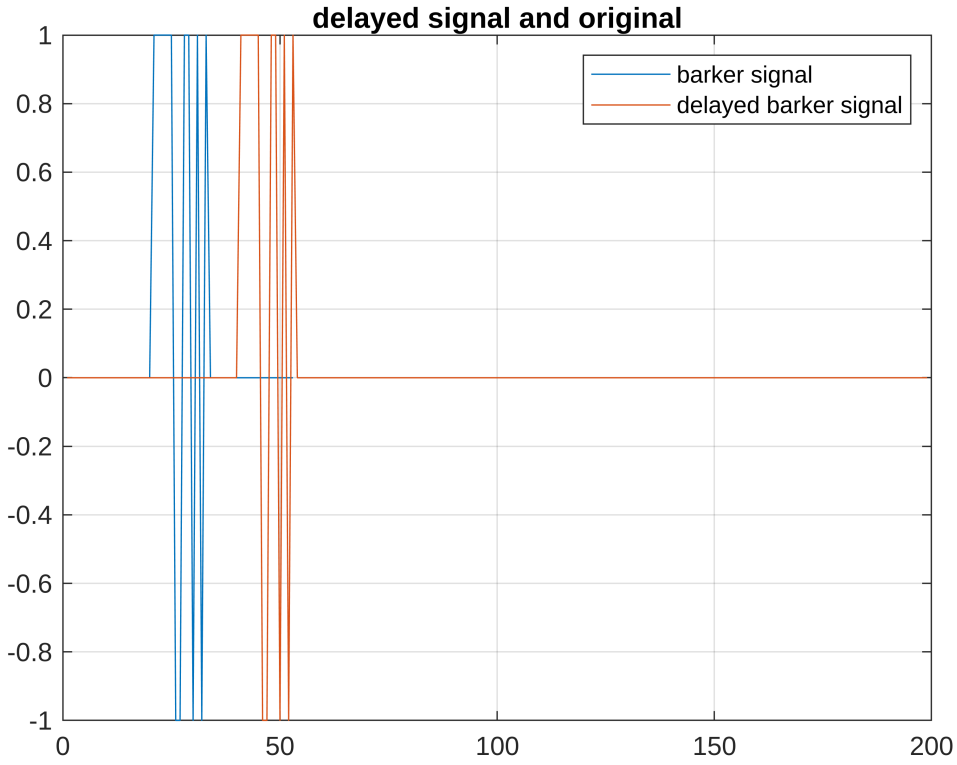
a) the location between peaks in the correlation and the autocorrelation will be the total delay.

b) above as

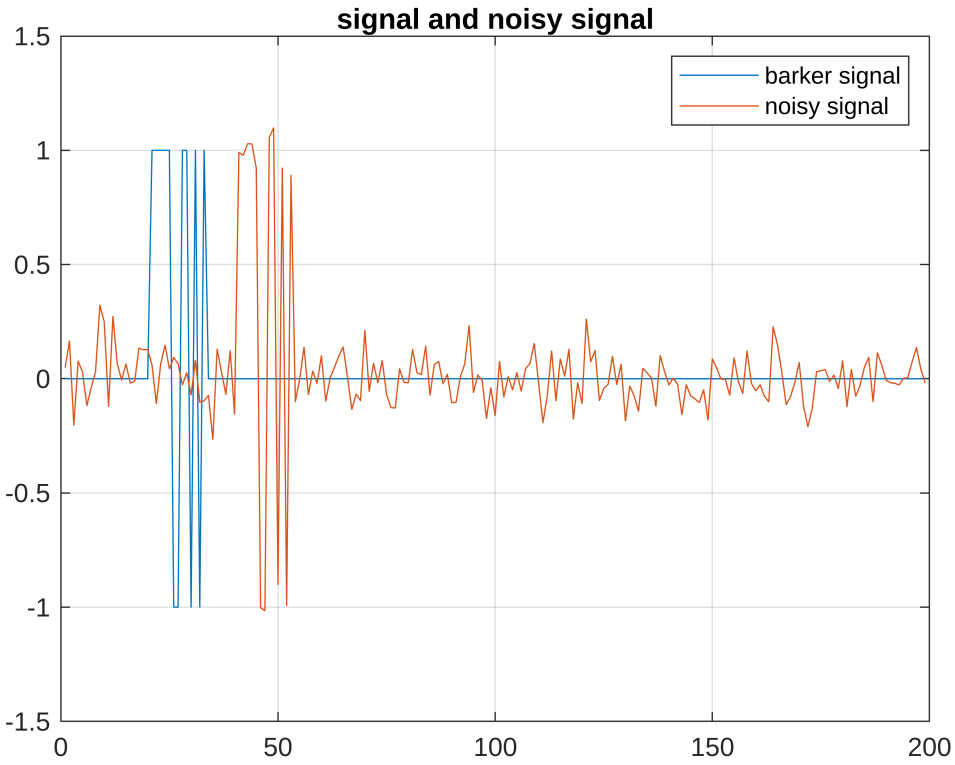
generation of rands and vec for part c and d.

```
figure1 = figure();
n = 1;
plot(x_a)
hold on
alpha = .9;
noise = alpha.*sqrt(variance)'*randn(1,199);
ya(delay+1:delay+length(x_a)) = x_a;
new_x = zeros(1,199);
```

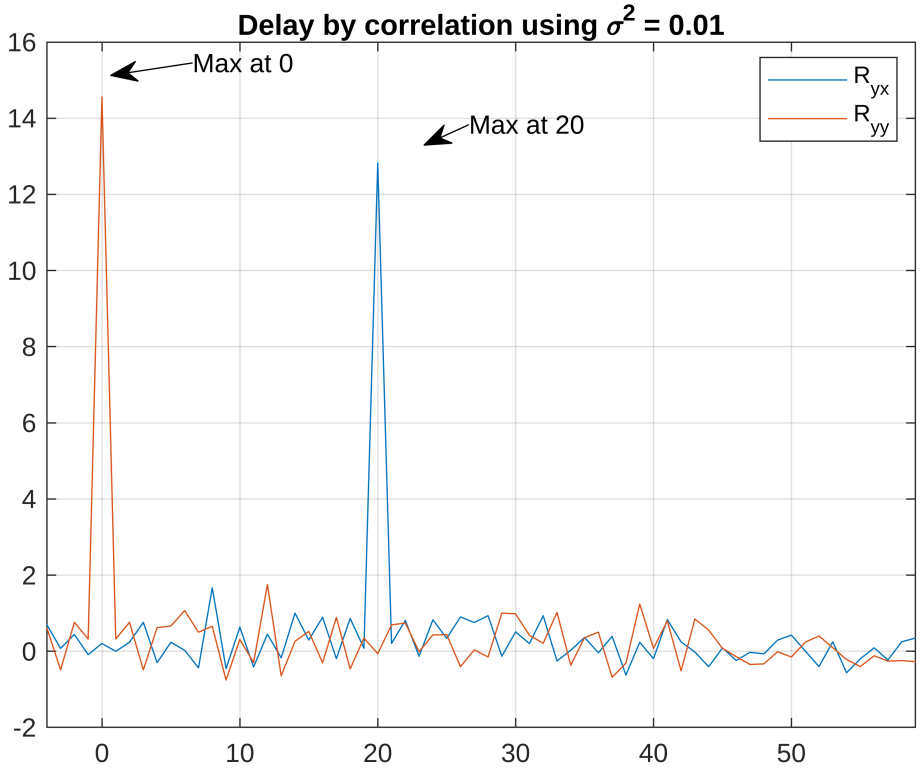
```
new_x(1:length(x_a)) = x_a;
x_a = new_x;
plot(ya)
grid on
title('delayed signal and original')
legend("barker signal","delayed barker signal")
hold off
```




```
ya = ya + noise(n,:);


figure2 = figure();
plot(x_a)
hold on
grid on
plot(ya)
title('signal and noisy signal')
legend("barker signal","noisy signal")
hold off
```

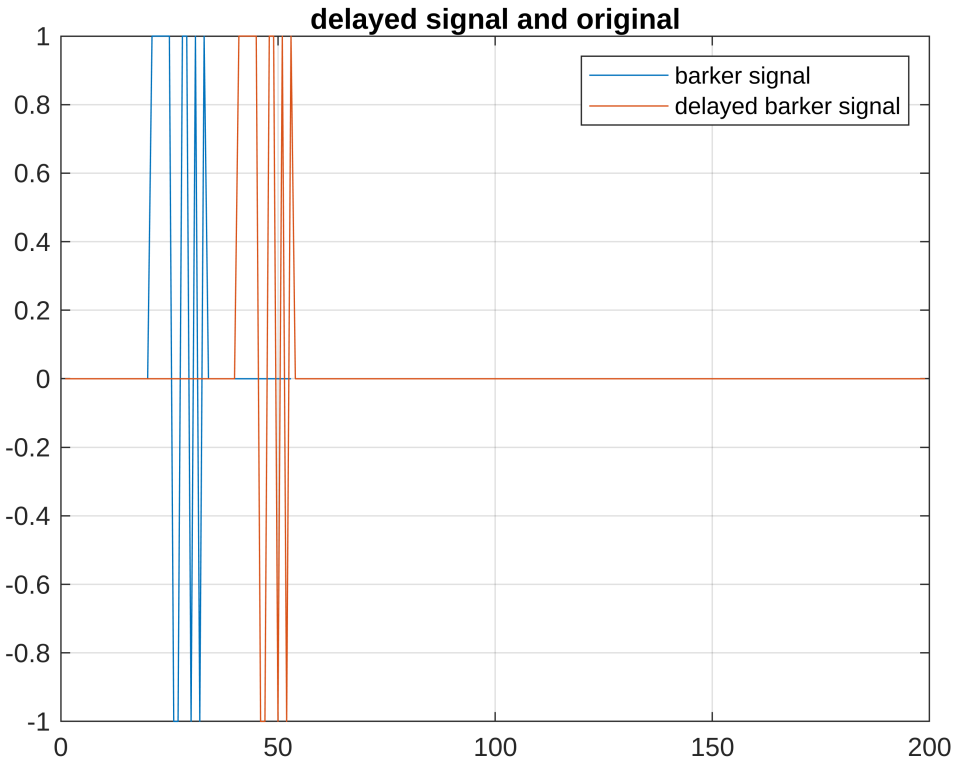



```
figure_delay = figure();
ryx = conv(ya,fliplr(x_a));
new_range = -floor(length(ryx)/2):1:floor(length(ryx)/2);
plot (new_range,ryx)
title('Delay by correlation')
hold on
ryx2 = conv(ya,fliplr(ya));
plot (new_range,ryx2);
[~, ryx_index] = max(ryx);
[~, ryx2_index] = max(ryx2);
title(sprintf('Delay by correlation using \\sigma^2 = %2.2f',variance(n)))
% Create textarrow
annotation(figure_delay,'textarrow',[0.26015625 0.187109375000001],...
    [0.900221729490022 0.885439763488544],'String',{sprintf('Max at
%d',ryx2_index + new_range(1)-1)});

% Create textarrow
annotation(figure_delay,'textarrow',[(delay)+18 delay+15]./(59+16),
[0.82680410742496 0.802527646129542],'String',{sprintf('Max at
%d',ryx_index+new_range(1)-1)});
legend('R_{yx}','R_{yy}',Location='best')
xlim([-4 59])
grid on
hold off
```

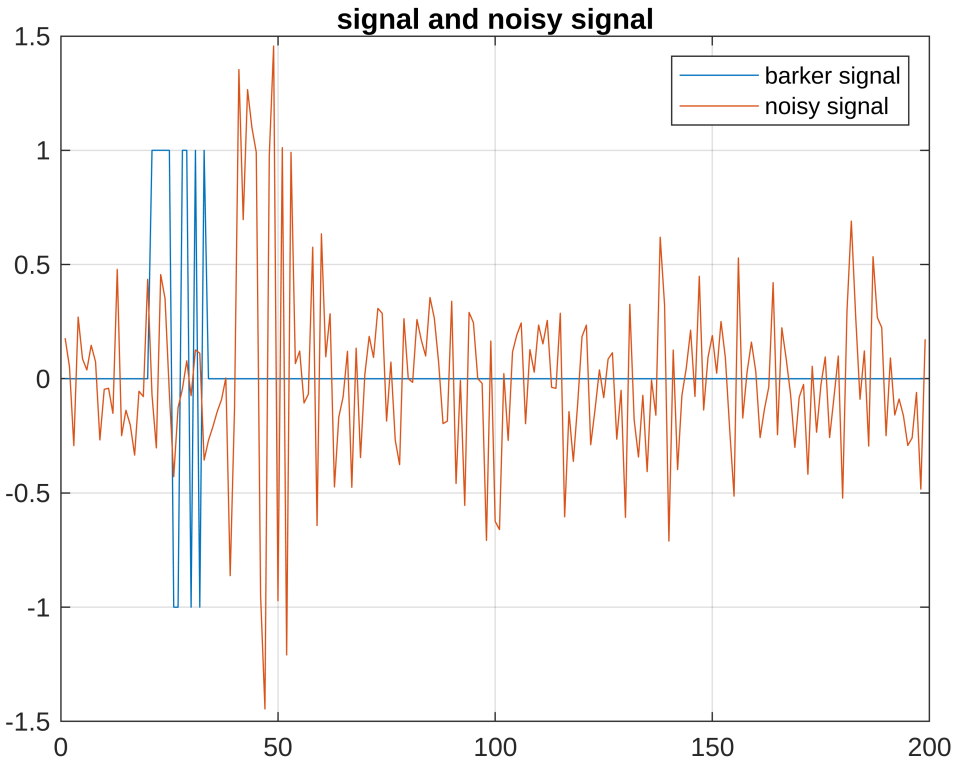


for second variance value

```
a = [1, 1, 1, 1, 1, -1,-1, 1, 1, -1, 1, -1, 1];
variance = [.01,.1, 1];
delay = 20;
ya = zeros(1,199);
x_a = [zeros(1,delay),a,zeros(1,delay)];
n = 2;
figure1 = figure();
plot(x_a)
hold on
alpha = .9;
noise = alpha.*sqrt(variance)'*randn(1,199);
ya(delay+1:delay+length(x_a)) = x_a;
new_x = zeros(1,199);
new_x(1:length(x_a)) = x_a;
x_a = new_x;
plot(ya)
grid on
title('delayed signal and original')
legend("barker signal","delayed barker signal")
hold off
```




```
ya = ya + noise(n,:);
figure
plot(x_a)
hold on
plot(ya)
grid on
title('signal and noisy signal')
legend("barker signal","noisy signal")
hold off
```

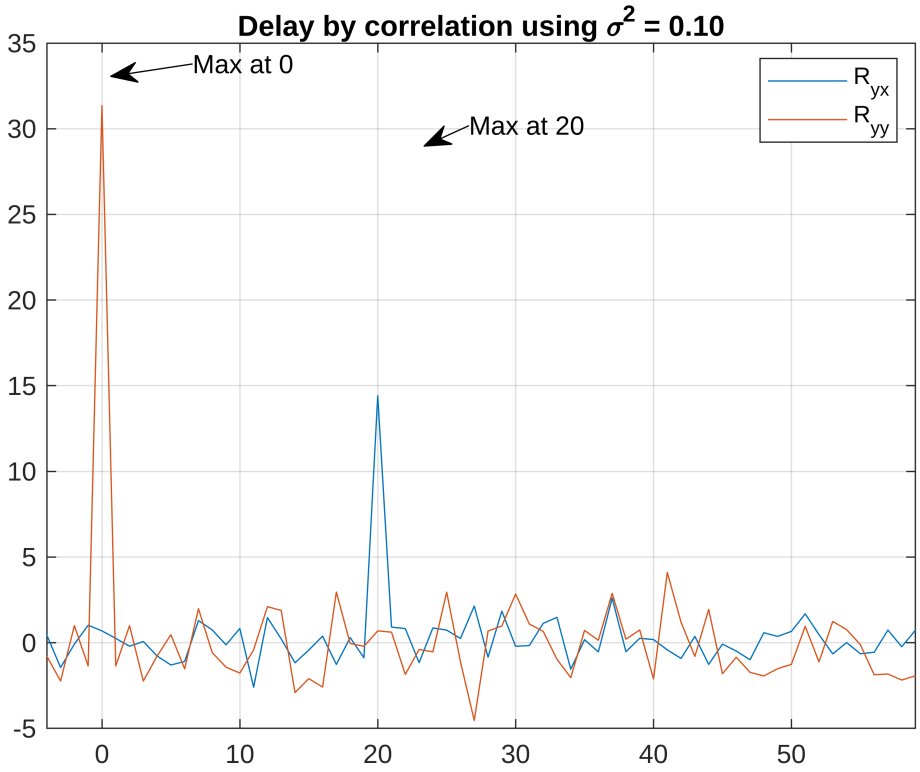



```
figure_delay = figure();
ryx = conv(ya,fliplr(x_a));
new_range = -floor(length(ryx)/2):1:floor(length(ryx)/2);
plot (new_range,ryx)
title(sprintf('Delay by correlation using \\sigma^2 = %2.2f',variance(n)))
hold on
ryx2 = conv(ya,fliplr(ya));
plot (new_range,ryx2);
[max_ryx, ryx_index] = max(ryx);
[max_ryx2, ryx2_index] = max(ryx2);
fprintf('delay = %3.3f',abs(ryx2_index - ryx_index))
```

```
delay = 20.000
```

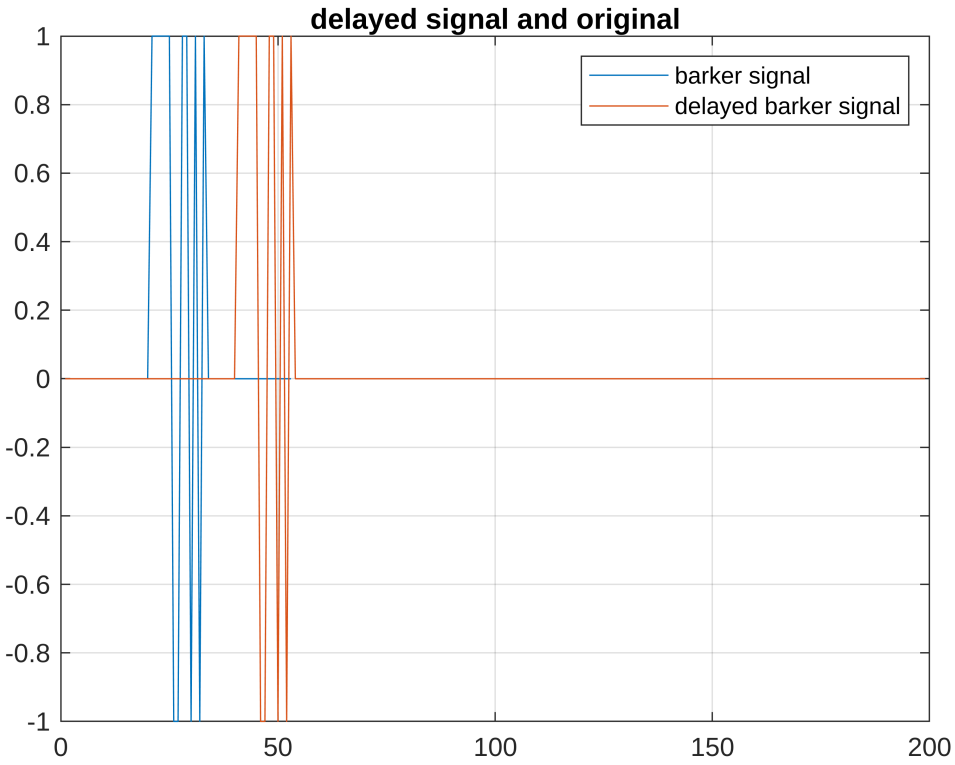
```
% Create textarrow
annotation(figure_delay,'textarrow',[0.26015625 0.187109375000001],...
    [0.900221729490022 0.885439763488544],'String',{sprintf('Max at
%d',ryx2_index + new_range(1)-1)});

% Create textarrow
annotation(figure_delay,'textarrow',[(delay)+18 delay+15]./(59+16),
[0.82680410742496 0.802527646129542],'String',{sprintf('Max at
%d',ryx_index+new_range(1)-1)});
legend('R_{yx}','R_{yy}',Location='best')
xlim([-4 59])
```

```
grid on
hold off
```

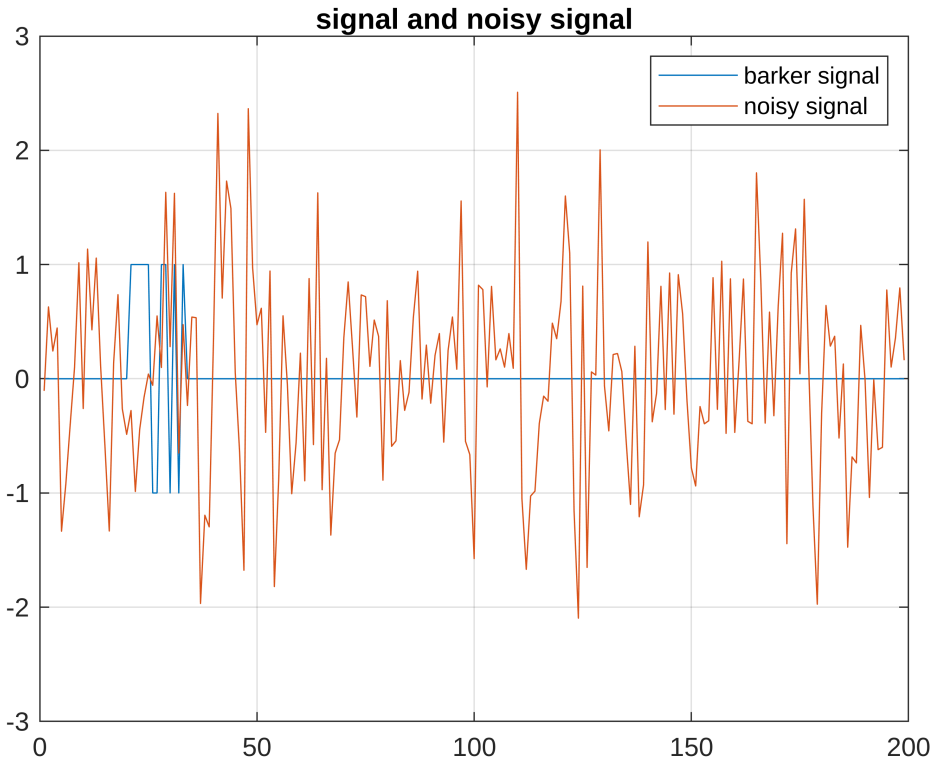


for third variance value

```
a = [1, 1, 1, 1, 1, -1,-1, 1, 1, -1, 1, -1, 1];
variance = [.01,.1, 1];
delay = 20;
ya = zeros(1,199);
x_a = [zeros(1,delay),a,zeros(1,delay)];
n = 3;
figure1 = figure();
plot(x_a)
hold on
grid on
alpha = .9;
noise = alpha.*sqrt(variance)'*randn(1,199);
ya(delay+1:delay+length(x_a)) = x_a;
new_x = zeros(1,199);
new_x(1:length(x_a)) = x_a;
x_a = new_x;
plot(ya)
title('delayed signal and original')
legend("barker signal","delayed barker signal")
hold off
```




```
ya = ya + noise(n,:);
figure
plot(x_a)
hold on
plot(ya)
grid on
title('signal and noisy signal')
legend("barker signal","noisy signal")
hold off
```

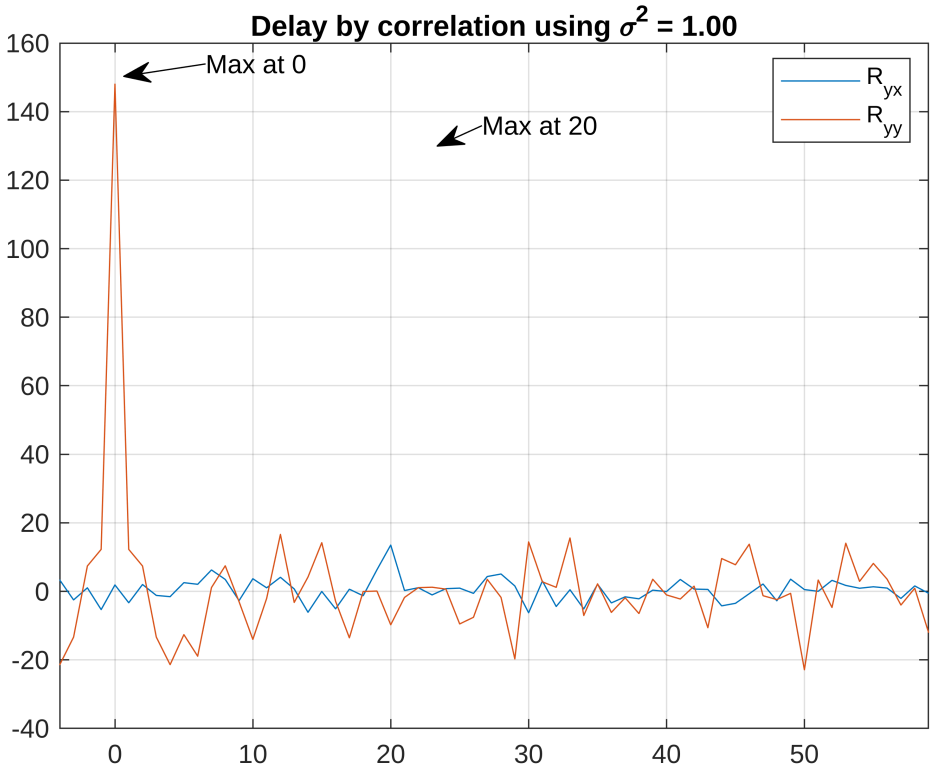



```
figure_delay = figure();
ryx = conv(ya,fliplr(x_a));
new_range = -floor(length(ryx)/2):1:floor(length(ryx)/2);
plot (new_range,ryx)
title(sprintf('Delay by correlation using \\sigma^2 = %2.2f',variance(n)))
hold on
ryx2 = conv(ya,fliplr(ya));
plot (new_range,ryx2);
[max_ryx, ryx_index] = max(ryx);
[max_ryx2, ryx2_index] = max(ryx2);
fprintf('delay = %3.3f',abs(ryx2_index - ryx_index))
```

```
delay = 20.000
```

```
% Create textarrow
annotation(figure_delay,'textarrow',[0.26015625 0.187109375000001],...
    [0.900221729490022 0.885439763488544],'String',{sprintf('Max at
%d',ryx2_index + new_range(1)-1)});

% Create textarrow
annotation(figure_delay,'textarrow',[(delay)+18 delay+15]./(59+16),
[0.82680410742496 0.802527646129542],'String',{sprintf('Max at
%d',ryx_index+new_range(1)-1)});
legend('R_{yx}','R_{yy}',Location='best')
xlim([-4 59])
```

```
grid on
hold off
```




```
function a = convert(data)
g = 9.8; % as in m/s^2
a = -1.5*g + 3*g*(data/63);
end

function y = diffeq_2(x)
    y = zeros(size(x));
    y(1) = 0;
    for n = 2:length(x)
        y(n) = (n/(n+1))*y(n-1) + (1/(n+1))*x(n);
    end
end

function y = diffeq_3(x)
    y = zeros(size(x));
    y(1) = 0;
    y(2) = y(1) + x(2);
    for n = 3:length(x)
        y(n) = y(n-1) - y(n-2) + x(n);
    end
end
```

```
function lti_test1(x,t,sig)
for i = 1:1:floor(length(x)/2)
    diff1= [diffeq1(x,t),x];
    plot(diff1)
    ylim([min(min(diff1(:,1)),min(diff1(:,2))) ...
        max(max(diff1(:,1)),max(diff1(:,2))) ]*1.2)
    title(sprintf('%s signal with diff eq function 1',sig))
    grid on
    hold on
    diff1= [diffeq1(x,i),x];
    plot(diff1(:,1),'--k')
    legend("y(n)","x(n)","y_{shifted}")
    hold off
    pause(.05)
end
end


function lti_test2(x,t,sig)
for i = 1:1:floor(length(x)/2)
    x_new = zeros(size(x));
    x_new(i:end) = x(i:end);
    diff2= [diffeq2(x_new,i),x_new];
    plot(diff2)
    ylim([min(min(diff2(:,1)),min(diff2(:,2))) ...
        max(max(diff2(:,1)),max(diff2(:,2))) ]*1.2)
    title(sprintf('%s signal with diff eq function 2',sig))
    grid on
    hold on
    diff2= [diffeq2(x_new,i),x_new];
    neweq = diff2(:,1);
    plot(neweq,'--k')
    legend("y(n)","x(n)","y_{shifted}")
    hold off
    pause(.05)
end
end


function lti_test3(x,t,sig)

diff3= [diffeq3(x,1),x];
lims = [min(min(diff3(:,1)),min(diff3(:,2))) ...
        max(max(diff3(:,1)),max(diff3(:,2))) ];

for i = 1:1:floor(length(x)/2)
    x_new = zeros(size(x));
    x_new(i:end) = x(i:end);
    diff3= [diffeq3(x_new,i),x_new];
```

```
    plot(diff3)
    ylim(lims*1.2)
    title(sprintf('%s signal with diff eq function 2',sig))
    grid on
    hold on
    diff3= [diffeq3(x_new,i),x_new];
    neweq = diff3(:,1);
    plot(neweq,'--k')
    legend("y(n)","x(n)","y_{shifted}")
    hold off
    pause(.05)
end
end

function cross_corr(x,y,signals)
% figure()
% plot(x,'k*-')
% hold on
% plot(y,'b:')
% legend("x_{i}(n)","x_{j}(n)")
% title("signals x_{i}(n) and x_{j}(n)")
% grid on
% hold off
convXy = conv(x,flip(y));
figure()
new_range = -floor(length(convXy)/2):1:floor(length(convXy)/2);
plot(new_range,convXy,'r')
hold on
plot(new_range,conv(y,flip(x)),'b')
if x == y
hold on
plot(conv(x,y),'b')
title(sprintf("Correlation R_{%s} with autocorr",signals))
legend(sprintf("R_{%s}",signals), "autocorr")
grid on
hold off
else
title(sprintf("Correlation R_{%s}",signals))
legend(sprintf("R_{%s}",signals),sprintf("R_{%s}",fliplr(signals)))
hold off
grid on
end
end
```