Signals and Systems
Lab 2

Vincent de Wit (s3038858) & Stefan Bussemaker (s2004674)

December 22, 2016

The function files that we made for this lab are shown thoughout the report. The script files which were used to create the plots etc. are shown in the Appendix.

Assignment 1

1. See Listing 1.

Listing 1: *circorr.m*

```
function z = circorr(x, y)
    N = length(x);                      % each consist of N samples
    z = zeros(1, N);                    % allocate memory for result

    for d = 1:N
        for n = 1:N
            k = mod(n-1+d-1, N) + 1;    % apply 'mod' operator, fix index
            z(d) = z(d) + x(n)*y(k);    % add product
        end
    end
end
```

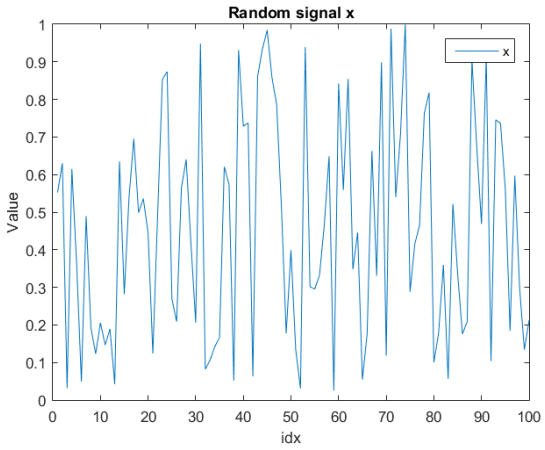
2. The plots of x and the correlation of x with itself are shown in Figure 1a and Figure 1b. In general the plot of `circorr(x,x)` with `x=rand(1,100)` fluctuates between 25 and 28 except at the very beginning of the plot. There it spikes to 35. This makes sense, since at the start of the signal, both are exactly the same.

3. `rotate.m` is shown in Listing 2. This function allows us the rotate a signal. Using this function a new signal y is created, which is `y = rotate(x, 50)`. Figure 2 shows the plot of the circular correlation of x and y and shows a shift by 50 off the previous plot. This means that the spike now is in the middle of the plot.

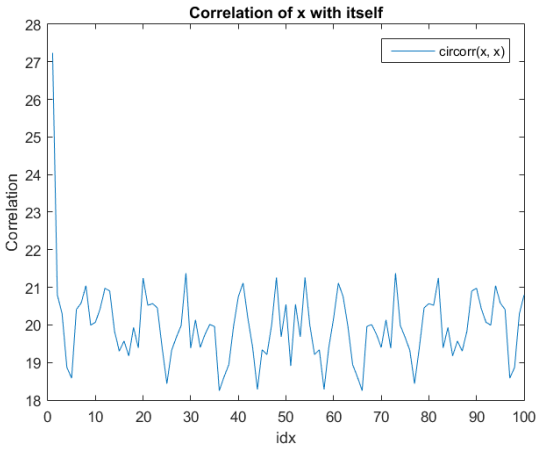
Listing 2: *rotate.m*

```
function rotated = rotate(sig ,n)
  % rotate - Rotate signal sig by n elements
  % needs 0 <= n <= length(sig)
  rotated = [sig(end-n+1:end) sig(1:end-n)];
end
```

4. Listing 3 shows `circpearson.m`. While the results of `circpearson(x, y)` are the same as `pearson(x, y)`, computing takes a lot longer. Using `timeit`, found that for the random signals `x = rand(1, 10000)` and `y = rand(1, 10000)` the time difference was 5.6 seconds.



(a) *Circular correlation of x and itself*



(b) *Circular correlation of x and y*

Figure 1

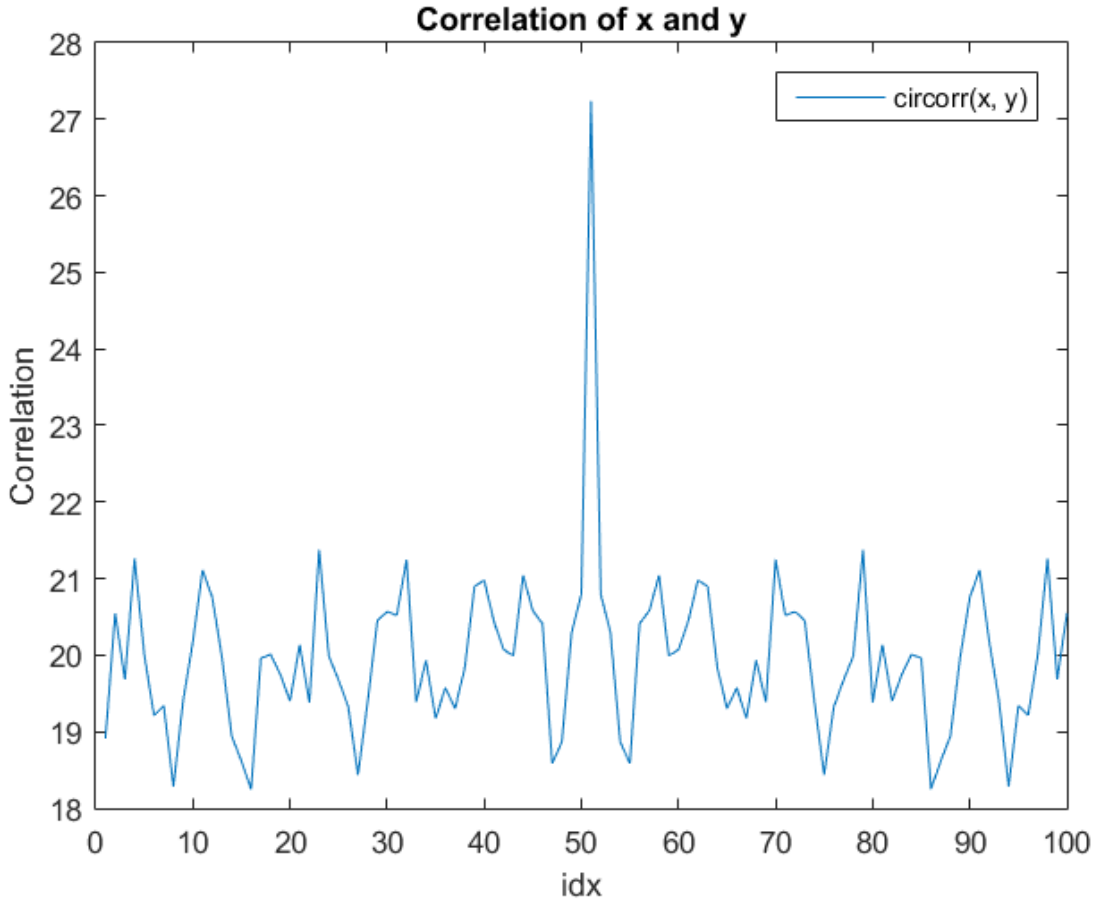


Figure 2: *Circular correlation of x and y*

Listing 3: *circpearson.m*

```
function p = circpearson(x, y)
  lenx = length(x);  % number of samples
  if (lenx ~= length(y))
    error('pearson(x,y): x and y are not of the same shape');
  end

  meanx = sum(x)/lenx;
  meany = sum(y)/lenx;
  p = zeros(lenx, 1);
    for d = 1:lenx
        enum = 0; X2 = 0; Y2 = 0;
        for i = 1:lenx
      X = x(i) - meanx;
      Y = y(mod( ( ( i-1 ) + ( d-1 ) ) ,lenx) + 1 )-meany;
      enum  = enum  + X * Y;
      X2 = X2 + X * X;
      Y2 = Y2 + Y * Y;
    end
    denom = sqrt(X2) * sqrt(Y2);

        if ( denom == 0)
            p(d) = 0;
        else
            p(d) = enum / ( denom );
        end
    end
end
```

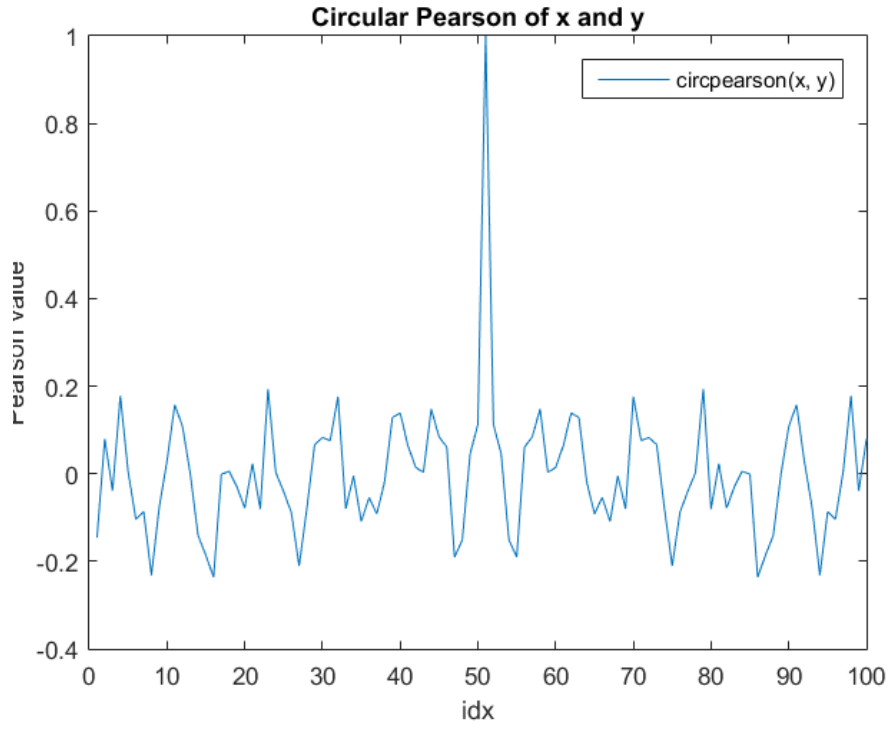


Figure 3: *Circular Pearson correlation of x and y*

5. The Pearson correlator normalizes the correlation. This means that the plot of the returned values are between -1 and +1. Furthermore when the mask exactly matches the signal it is guaranteed to return 1. This makes the Pearson correlator easier to interpret compared to the previous version of the correlator.

6. We perform the circular Pearson correlator on $5cos(10pi)$ and $5sin(10pi - 0.5pi)$. Figure 4 shows the result. We observe a new sinusoid, but with a phase change. Varying the amplitudes yields exactly the same result. This is a desired effect and is done thanks to the normalization.

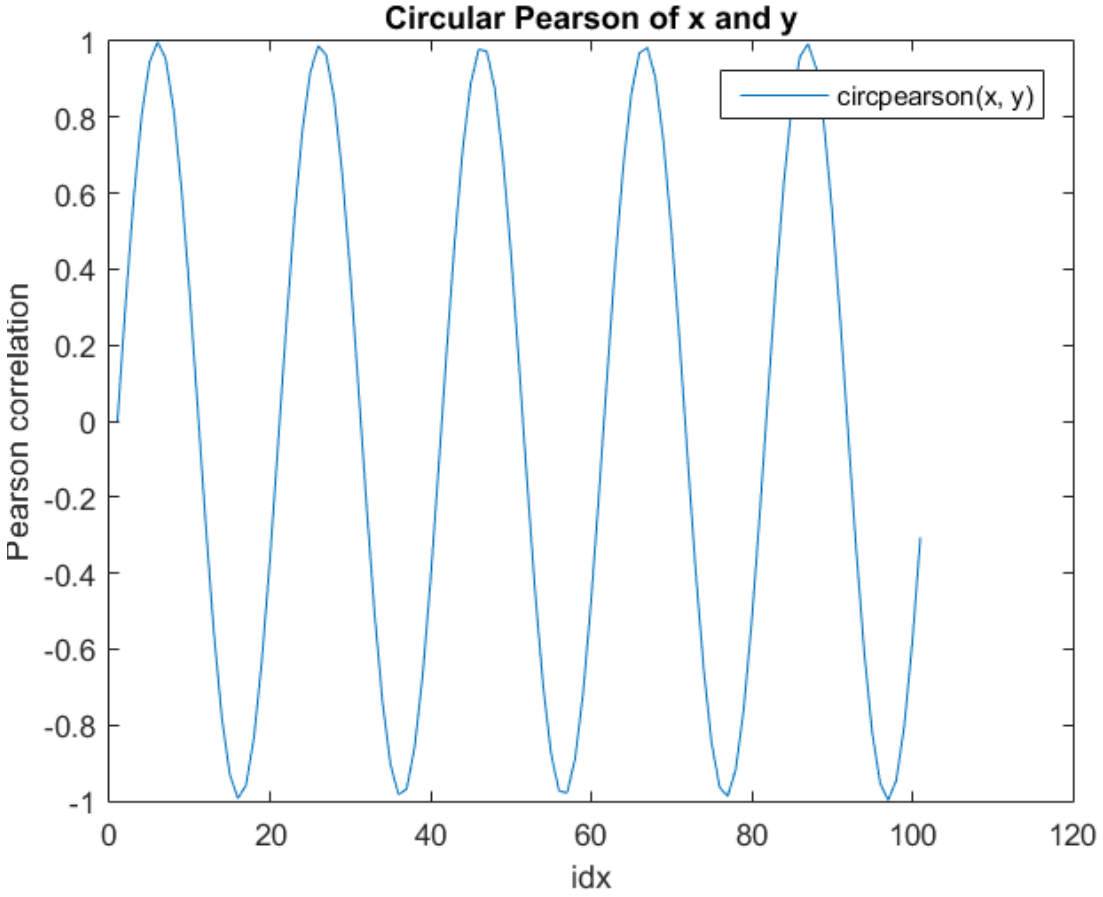
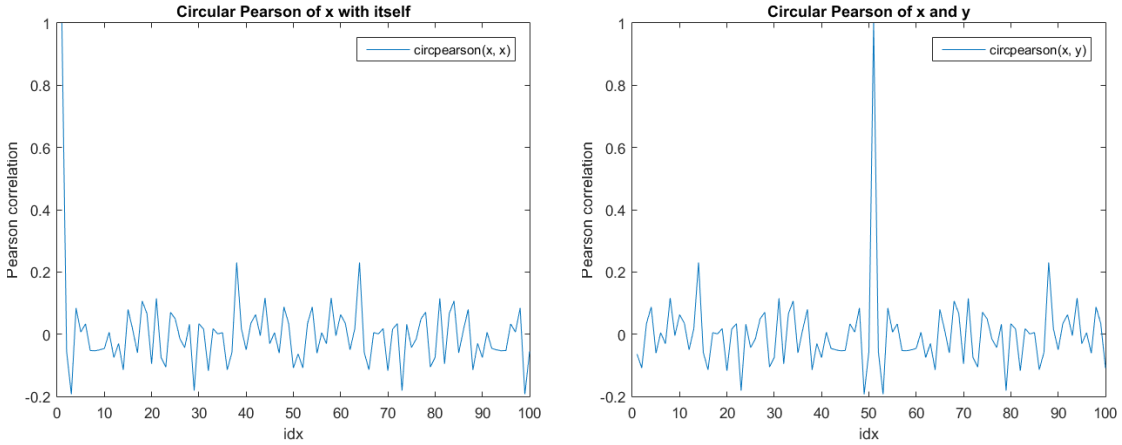


Figure 4: *Circular pearson of cosine x and sine y*

7. The plots of the circular Pearson as shown in Figure 5a and Figure 5b have the same shape as the plots from the circular correlation. The only difference is the scale. This time, the values are between -1 and +1. Figure 6 shows the circular Pearson correlation of a noisy x and y. We made x noisy by adding or subtracting a random value from the amplitude at each index of x. We find that the delay can be reasonably detected to up to 25% noise.



(a) *Circular Pearson correlation of x with itself* **(b)** *Circular Pearson correlation of x and y*

Figure 5

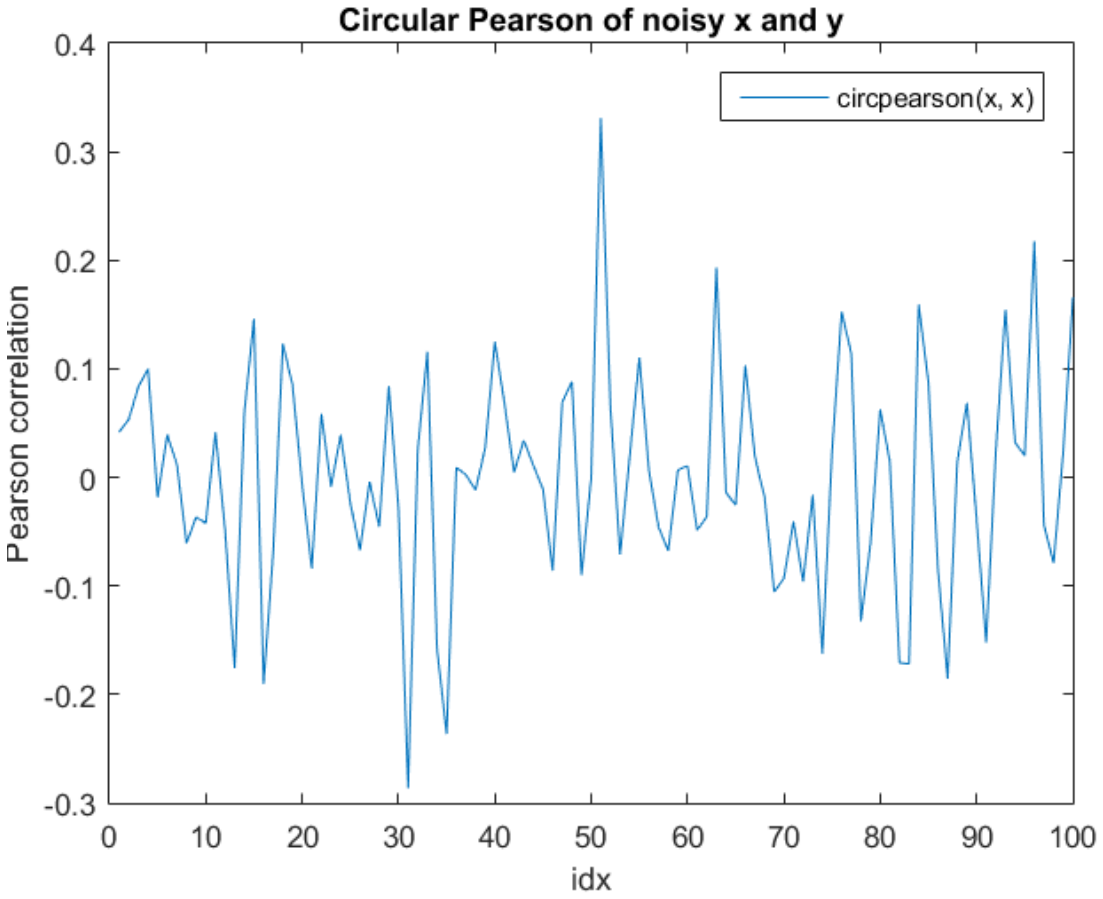


Figure 6: *Circular Pearson correlation of a noisy x and y*

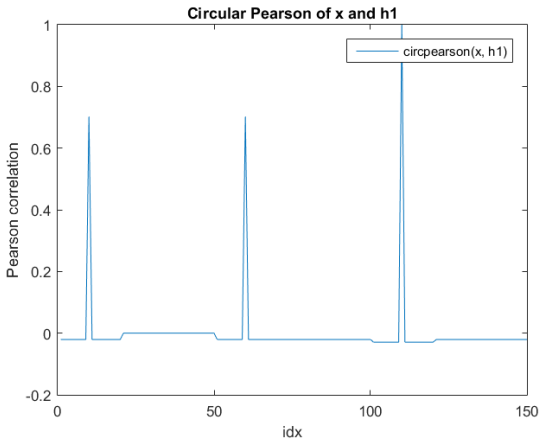
Assignment 2

1. Listing 4 shows our implementation of the circular Pearson correlator for signals of unequal length. Some samples of x are prepended. The computation of the numerator makes use of the build-in `conv` function.

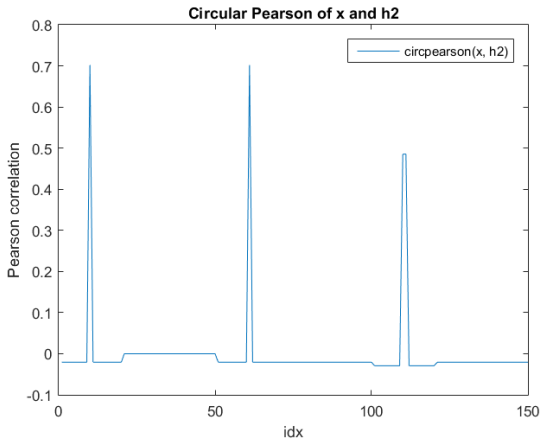
Listing 4: *pearson2.m*

```
function p = pearson2(x, y)
  N = length(x);  % number of sign samples
  M = length(y);  % number of mask samples
  if (N < M)
    error('pearson2(x,y): mask is longer than signal');
  end

  p = zeros(1, N);

  x2 = [x(end-M+1:end); x];       % prepend x

  Y2 = sqrt(sum((y - mean(y)).*(y - mean(y))));
  for d = 1:N
    x_masked = x2(d:d+M-1);
    x_minus_mean_x = x_masked - mean(x_masked);
    X2 = sqrt(sum(x_minus_mean_x.*x_minus_mean_x));
    denom = X2 * Y2;

    if (denom == 0)
      p(d) = 0;
    else
      enum  = sum(conv(x_minus_mean_x, fliplr(y) - mean(y),'valid'));
      p(d) = enum / denom;
    end
  end
end
```

2. Figure 7a and Figure 7b show the correlations for x with h1 and h2 respectively. We can see a perfect match in the first case, with a peak of 1. In the second case where the mask is just one sample off the peak that was 1 when using the previous sample now is about halved. We conclude that the Pearson correlation is not very robust for finding patterns.



(a) *Circular Pearson correlation of x and h1*



(b) *Circular Pearson correlation of x and h2*

Figure 7

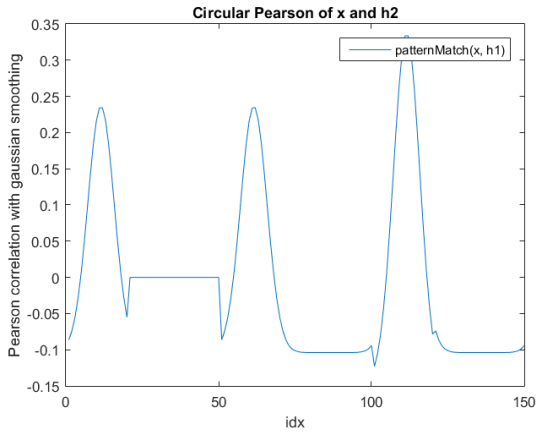
3. In order to make the correlator more robust, we use Gaussian smoothing. This way, the peaks are more smeared out, resulting in higher relative peaks when a mask is similar to the signal, but not the same. The results are shown in Figure 8a and Figure 8b. The implementation is shown in Listing 5.

Listing 5: *patternMask.m*

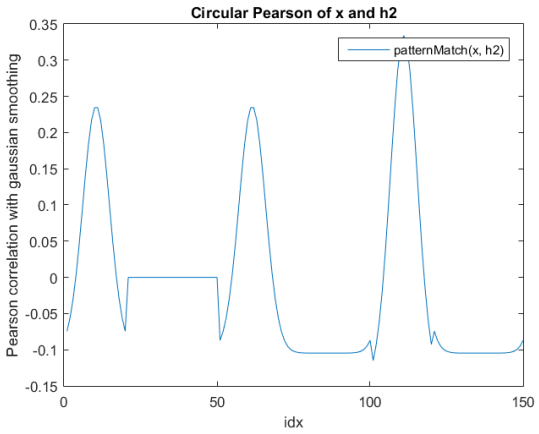
```
function result = patternMatch(x, h)
  % matches pattern
  % using a gaussian mask
  N = length(h);
  n = -(N-1)/2:(N-1)/2;
  alpha = 8;

  stdev = (N-1)/(2*alpha);
  y = exp(-1/2*(n/stdev).^2);

  h2 = conv(y, h, 'same');
  result = pearson2(x, h2);
end
```

4. Figure 9a shows the circular Pearson correlator for Handel. Both occurrences of "Hallelujah" are clearly detected in the signal.

5. Figure 9 shows the circular Pearson correlator for fur Elise. This time the occurrences aren't that clear.

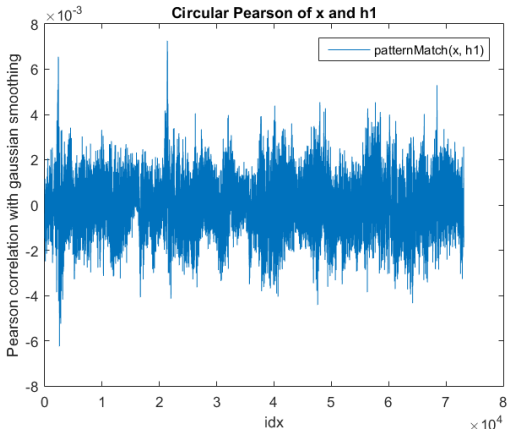


(a) *patternMask of x and h1 with a Gaussian smoothing*

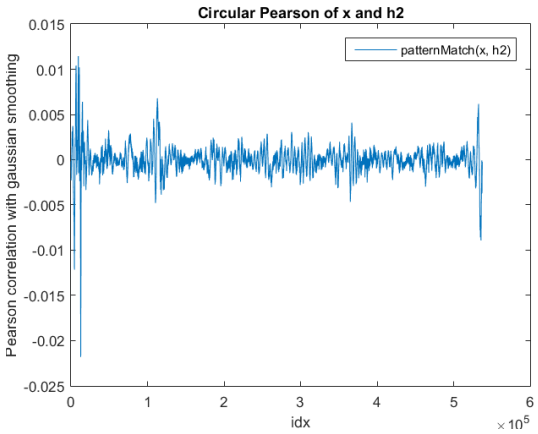


(b) *patternMask of x and h2 with a Gaussian smoothing*

Figure 8



(a) *patternMask of x (handel.wav) and h1 (hallelujah.wav) with a Gaussian smoothing*



(b) *patternMask of x (furelise8kHZ.wav) and h2 (cropelise8kHz.wav) with a Gaussian smoothing*

Figure 9

Assignment 3

1. For this exercise we made use of the Matlab function `normxcorr2` form the Image processing toolbox. This function uses the Pearson correlator for 2D signals. In order to detect occurences of the letter, we use thresholding. We handpicked a value which gives us the right amount of occurences. Afterwards, an increasing amount of Salt 'n Pepper noise is added to test the robustness. Our approach appears to not be too robust, since the performance rapidly decreases. No 'm' is recognized only after adding 6% noise. Figure 10 shows the results.

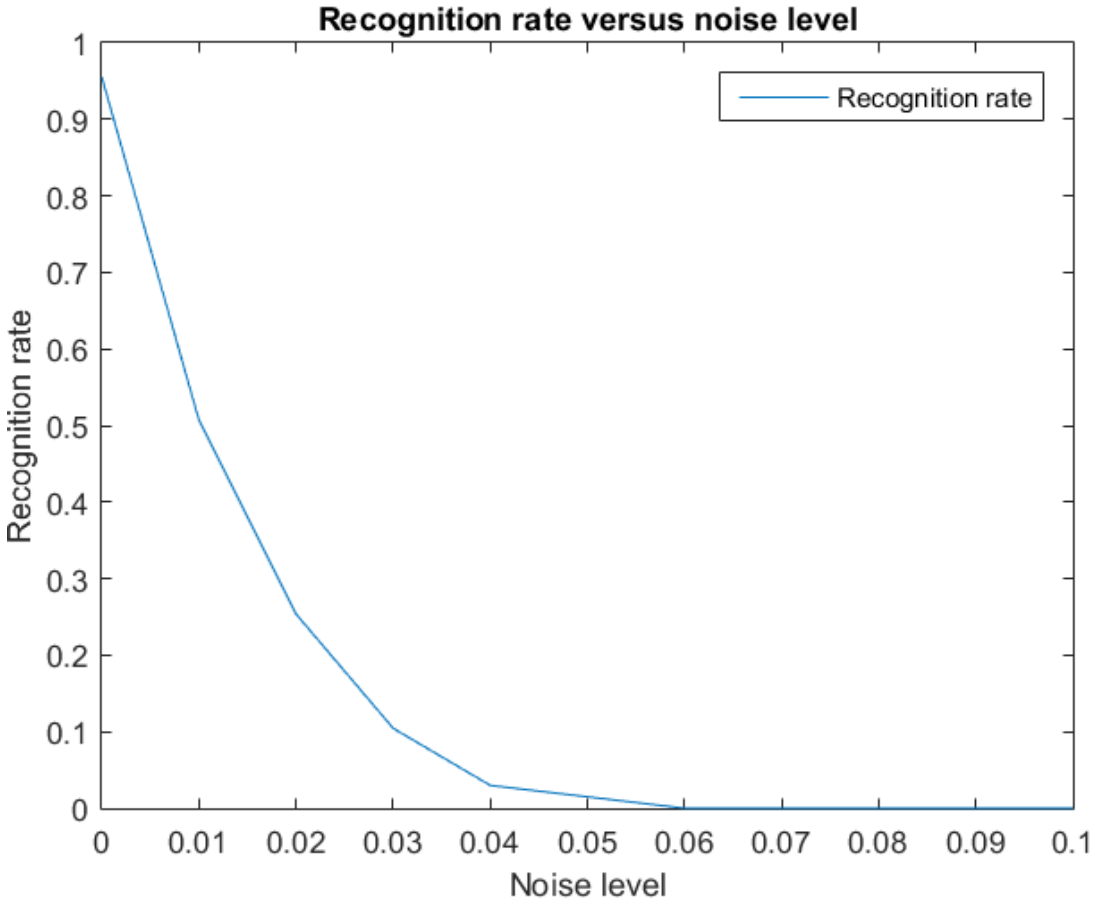


Figure 10: *Recognition rate versus noise level*

A. Assignment 1

Listing 6: *Script file for exercise 1*

```
% Vincent de Wit (s3038858)
% Stefan Bussemaker (s2004674)
%
% Exercise 1 - Time delay and correlators

%% 1b)
x = rand(1,100);
figure; plot(x);
xlabel('idx'); ylabel('Value');
title('Random signal x'); legend('x');
figure; plot(circorr(x, x));
xlabel('idx'); ylabel('Correlation');
title('Correlation of x with itself'); legend('circorr(x, x)');

%% 1c)
y = rotate(x, 50);
plot(circorr(x, y));
xlabel('idx'); ylabel('Correlation');
title('Correlation of x and y'); legend('circorr(x, y)');

%% 1d)
plot(circpearson(x, y));
xlabel('idx'); ylabel('Pearson correlation');
title('Circular Pearson of x and y'); legend('circpearson(x, y)');

x = rand(1, 10000);
y = rand(1, 10000);
f = @() pearson(x, y);
g = @() circpearson(x, y);
diffRunTime = timeit(g) - timeit(f);

%% 1f)
x = sinusoid2samples(5, 2*pi*5, 0, 100);       % 5 cos (10pi)
y = sinusoid2samples(5, 2*pi*5, -0.5*pi, 100); % 5 sin (10pi - 0.5pi)
plot(circpearson(x, y));
xlabel('idx'); ylabel('Pearson correlation');
title('Circular Pearson of x and y'); legend('circpearson(x, y)');

%% 1g)
x = rand(1,100);
plot(circpearson(x, x));
xlabel('idx'); ylabel('Pearson correlation');
title('Circular Pearson of x with itself'); legend('circpearson(x, x)');
figure;

y = rotate(x, 50);
plot(circpearson(x, y));
xlabel('idx'); ylabel('Pearson correlation');
title('Circular Pearson of x and y'); legend('circpearson(x, y)');

figure;
noise = 2.5;
```

```
x = x + noise * 2*(rand(1, length(x)) - 0.5);
plot(x); figure;
plot(circpearson(x, y));
xlabel('idx'); ylabel('Pearson correlation');
title('Circular Pearson of noisy x and y'); legend('circpearson(x, x)');
```

B. Assignment 2

Listing 7: *Script file for exercise 2*

```
% Vincent de Wit (s3038858)
% Stefan Bussemaker (s2004674)
%
% Exercise 2 - A more general Pearson correlator

%% 2b)
x = zeros(150,1); x(50) = 1; x(100) = 1;
h1= zeros(70,1); h1(10) = 1; h1(60) = 1;
h2= zeros(70,1); h2(11) = 1; h2(60) = 1;

z1 = pearson2(x, h1);
figure(1);
plot(z1);
xlabel('idx'); ylabel('Pearson correlation');
title('Circular Pearson of x and h1'); legend('circpearson(x, h1)');

z2 = pearson2(x, h2);
figure(2);
plot(z2);
xlabel('idx'); ylabel('Pearson correlation');
title('Circular Pearson of x and h2'); legend('circpearson(x, h2)');

%% 2c)
z3 = patternMatch(x, h1);
figure(3);
plot(z3);
xlabel('idx'); ylabel('Pearson correlation with gaussian smoothing');
title('Circular Pearson of x and h2'); legend('patternMatch(x, h1)');

z4 = patternMatch(x, h2);
figure(4);
plot(z4);
xlabel('idx'); ylabel('Pearson correlation with gaussian smoothing');
title('Circular Pearson of x and h2'); legend('patternMatch(x, h2)');

%% 2d)
[y5, ~] = audioread('data/handel.wav');
[h5, ~] = audioread('data/hallelujah.wav');

z5 = patternMatch(y5, h5);
figure(5);
plot(z5);
xlabel('idx'); ylabel('Pearson correlation with gaussian smoothing');
title('Circular Pearson of x and h1'); legend('patternMatch(x, h1)');
```

```
%% 2e)
[y6, ~] = audioread('data/furelise8kHZ.wav');
[h6, ~] = audioread('data/cropelise8kHz.wav');

z6 = patternMatch(y6, h6);
figure(6);
plot(z6);
xlabel('idx'); ylabel('Pearson correlation with gaussian smoothing');
title('Circular Pearson of x and h2'); legend('patternMatch(x, h2)');
```

C. Assignment 3

Listing 8: *Script file for exercise 3*

```
% Vincent de Wit (s3038858)
% Stefan Bussemaker (s2004674)
%
% Exercise 3 - Correlation in 2D (Images)

%% 3
A = imread('data/page.pgm');
B = imread('data/maskM.pgm');
imshow(A);

%% Noise
threshold = 0.865;
noises = 0:0.01:1;
total = 67;

results = zeros(1, length(noises));
for idx = 1:length(noises)
    noise = noises(idx);
    x2 = imnoise(A,'salt & pepper', noise);
    y2 = normxcorr2(B, x2);

    results(idx) =  sum(sum(abs(y2) > threshold)) / 67;
end

plot(noises, results);
xlabel('Noise level'); ylabel('Recognition rate');
title('Recognition rate versus noise level'); legend('Recognition rate');
```