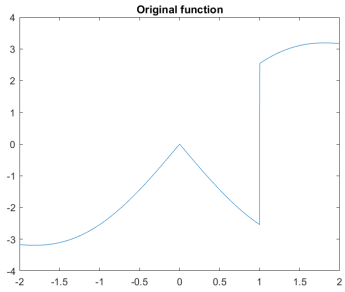
Wavelets assignment: denoising and inpainting with wavelets

Elina Cherman (r0703889) and Pilar Van Der Borght (r0716758)

December 2021

Contents

1 Wavelet-based denoising



(a) Original function

(b) Original Image

Figure 1: Original figures

1.1 Question 2.1

As can be seen in figure 2, the coefficients get larger as the level increases. In this figure the coefficients of the finer scale (higher level) are at the left and the one of the coarse level at the right. In the figure are plotted Daubechies wavelets with different order (order 2, 4 and 6). We see that the coefficients reduce with an increasing order.

A rough upper bound for the coefficient is:

$$|w_{jk}| \leq 2^{-j(d+\frac{1}{2})}$$

Where d is the order of the wavelet.

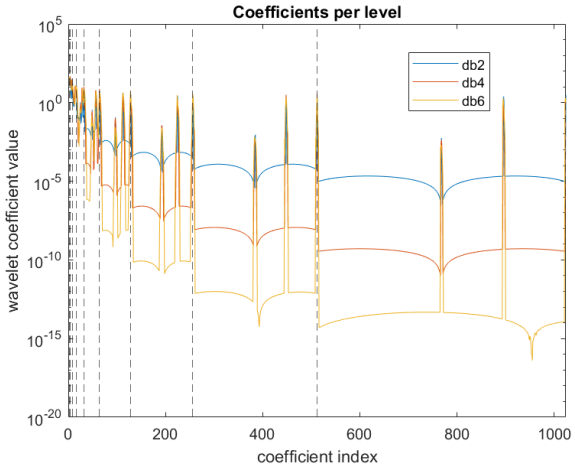


Figure 2: Wavelet coefficients per level

The given function has two discontinuities, one at $x = 0$ and one at $x = 1$. This discontinuities are also visible in the coefficients, figure 3. At the places where a discontinuity appears and at the boundaries the coefficients are larger. This is due to the fact that the function is non-derivable in those points. In figure 3 the effect of different boundary conditions is visible.

In figure 4 Daubechies wavelet coefficients of order 2, 4 and 6 are plotted. A higher order gives a larger support which lads to a larger impact of a single discontinuity.

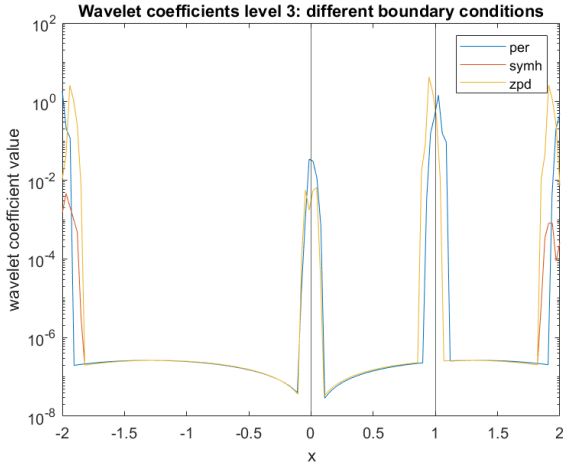


Figure 3: Wavelet coefficients of level 3 with different boundary conditions

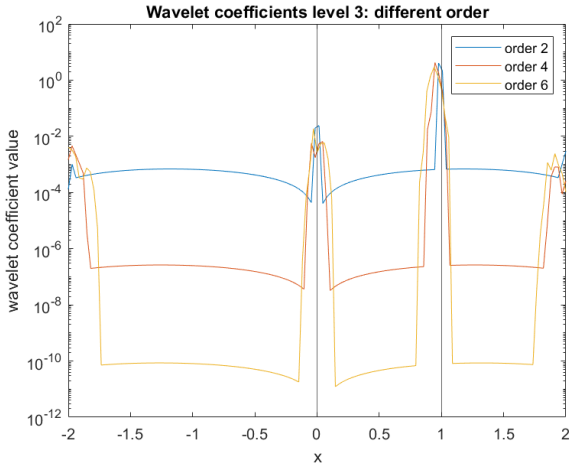


Figure 4: Wavelet coefficients of level 3 with different order

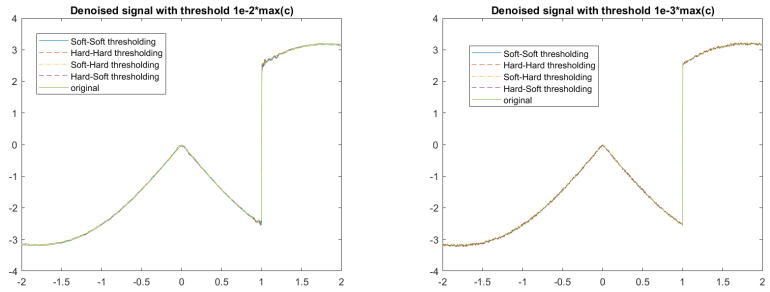
Further experiments will be done with Symmetric extension (half point), default mode of MATLAB.

1.2 Task 2.2

Soft thresholding is not usually applied to the scaling coefficients, only to the wavelet coefficients. Why?
The scaling coefficients are usually larger than the chosen threshold. Therefore when applying hard thresholding it will not affect the scaling coefficients. However when applying soft threshold, the scaling coefficients will become smaller, but there will be no visual difference noticeable. Since we are primarily interested in the visual reconstruction, this is no problem.

Denoising scheme: parameters
$\epsilon = 0.1$
Wavelet: Daubechie order 4.
Number of levels: 4 (This level is chosen because a low level has a low compression rate, a high compression rate is more preferable. However high levels will contain a lot of negative boundary effects. Furthermore for the first few levels there is a lot to gain, but for higher levels there is less to gain and the boundary effects become worse. We want to have a trade-off between the two.)
The effect of different thresholds δ on hard and soft thresholds will be discussed in section 1.3.

1.3 Question 2.3



(a) threshold $10^{-2} \max w_{jk}$ (b) threshold $10^{-3} \max w_{jk}$

Figure 5: Denoised signal

In figure 5 the original noise-free signal and the denoised signals are shown. Figure on the left shows the signal with a threshold value 10^{-2} times the maximum coefficient and the one on the right with threshold 10^{-3} times the maximum coefficient. The denoised signal with a larger threshold reduces more noise. The larger the threshold, the more coefficients are set to zero and will therefore give a smoother result.

At the places of discontinuities we still see a lot of oscillations. At the discontinuities the coefficients are larger as seen before, after thresholding these coefficients will stay (not set to zero). That is why at this location we observe noise.

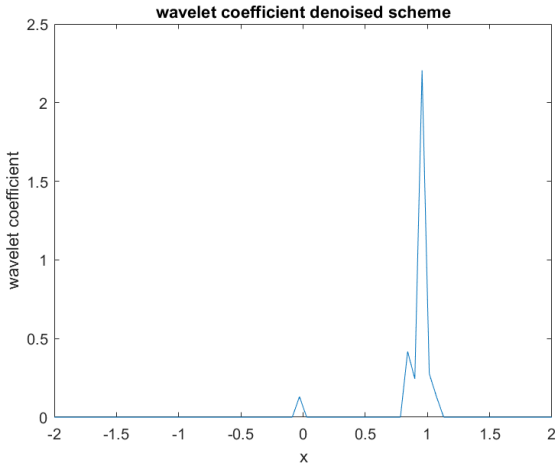


Figure 6: Wavelet coefficients denoised

There is no clear visual difference between soft and hard thresholding. But looking at the mean error between the denoised signal and the original noise free signal, we observe that soft thresholding gives a slightly worse result. In hard thresholding, some coefficients are set to zero. But in soft thresholding, all the coefficients will be changed and that is why the error will be larger.

Figure 7: Mean error

wavelet/scaling	Soft	Hard
Soft	0.0319	0.009
Hard	0.0307	0.0069

1.4 Task 2.4

Denoising scheme: parameters
$\epsilon = 0.3$
Number of levels: 4
threshold = 0.02 * maximum value of the coefficients

1.5 Question 2.5

The image with the noise is shown in figure 8. The denoised images with different techniques and parameters are shown in the figures 9, 10, 11 and 12. Both Daubechies and Coiflet wavelets have the property of orthogonality and are therefore also non-symmetric. The results of these wavelet transforms are similar. The noise in the final image is less visible, but on the background the tree and the small tower are also filtered. Furthermore there is ringing at the edge of the man. These ringing effects are less visible in the results obtained with the Biorthogonal wavelets. The Biorthogonal wavelet family has the property of linear phase, which is needed for signal and image reconstruction. The last wavelet family used is the Haar wavelet. The Haar wavelet is the only symmetric and orthogonal wavelet. In the resulting images with the Haar wavelet transform the ringing effect is gone, since the functions are more blocky it is useful for edge detection and therefore there is no ringing effect. On the other hand when zooming into the face of the man in figure 12, the blocks are visible in the face of the man. The difference between soft and hard thresholding is similar for all wavelet transforms. The images reconstructed with the hard threshold are less blurred than the ones with the soft threshold. Soft thresholding affects all coefficients by subtracting the threshold from them all. This explains why the images reconstructed with soft thresholding are more blurred. The table in figure 13 shows the signal to noise ratio's for each reconstruction. The images reconstructed with the hard thresholding have also persistently better ratio's.

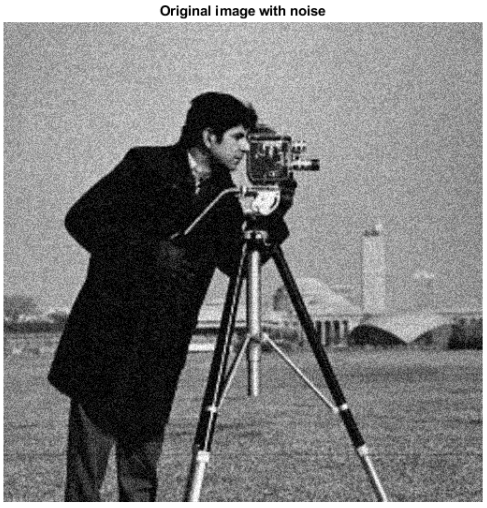
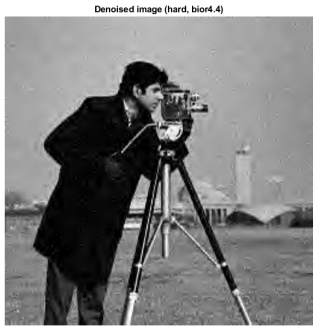


Figure 8: Original image with noise

(a) With hard thresholding

(b) With soft thresholding

Figure 9: Denoised figures with Daubechies order 4



(a) With hard thresholding

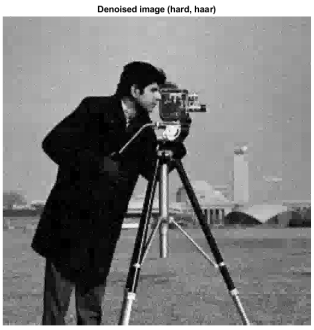
(b) With soft thresholding

Figure 10: Denoised figures with Biorthogonal wavelets order 4 for both primal and dual

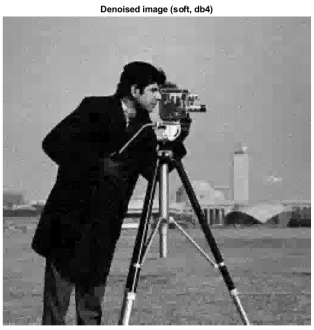
(a) With hard thresholding

(b) With soft thresholding

Figure 11: Denoised figures with Coiflets order 4



(a) With hard thresholding



(b) With soft thresholding

Figure 12: Denoised figures with Haar wavelets

Figure 13: Signal to noise ratio

	Soft	Hard
db4	14.6885	15.3912
coif4	14.7358	15.4699
bior4.4	14.7291	15.4566
haar	14.6160	15.4369

1.6 Task 2.6

Denoising scheme: parameters
$\epsilon = 0.3$
Number of levels: 4
threshold = 0.02 * maximum value of the coefficients

1.7 Question 2.7

In the next part of the assignment, we only use Daubechies wavelets and Biorthogonal wavelets. Since the Daubechies wavelets are good for signal denoising and the Biorthogonal wavelets for image processing. The figures 14 and 15 are obtained by using these wavelet families in the redundant wavelet based denoising scheme. The redundant wavelet transform is called like this because it skips the step of downsampling. Due to this there will be more coefficients than for the non-redundant wavelet transform and multiple reconstructions are possible. As in the previous section the images reconstructed with the soft thresholding are more blurry than the ones reconstructed with the hard thresholding. This is also confirmed by the signal to noise ratio's in the table in figure 16. However the ringing effect is less visible for the reconstructions with the redundant

wavelets than for images reconstructed with the non-redundant wavelet transforms. Consequently the images denoised with the redundant wavelet transforms are visually better than the ones denoised with non-redundant wavelet transforms.

(a) With hard thresholding

(b) With soft thresholding

Figure 14: Denoised figures with redundant Daubechies order 4

(a) With hard thresholding

(b) With soft thresholding

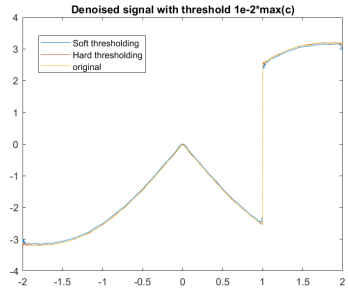
Figure 15: Denoised figures with redundant Biorthogonal wavelets order 4 for both primal and dual

Figure 16: Signal to noise ratio

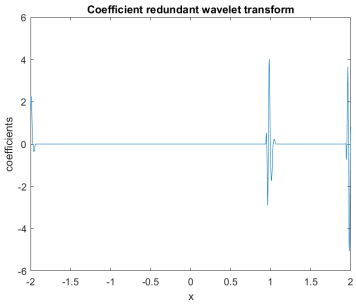
	Soft	Hard
db4	14.5041	15.4398
bior4.4	14.6383	15.5367

1.8 Question 2.8

If we compare the result of a redundant wavelet transform with the non redundant wavelet transform, we notice that the noise at the boundaries is less reduced. For the same (relative) threshold more coefficients are left and are responsible for the large noise at the boundaries and discontinuities.



(a) Denoised signal threshold $10^{-2} \max w_{jk}$



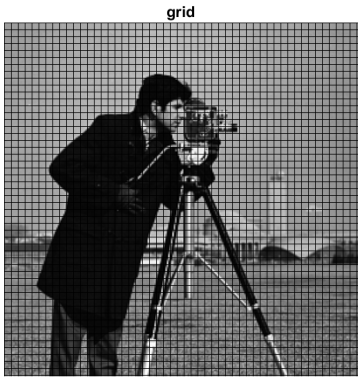
(b) Coefficients level 3

Figure 17: Redundant wavelet transform on smooth function

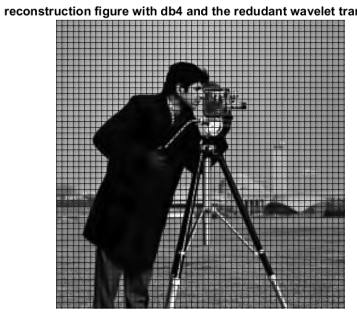
1.9 Task 2.9

In image 18 the grid is not recovered. Instead of recovering the full image, the image becomes fuzzy. We observe a new less clear grid pattern in between the original grid.

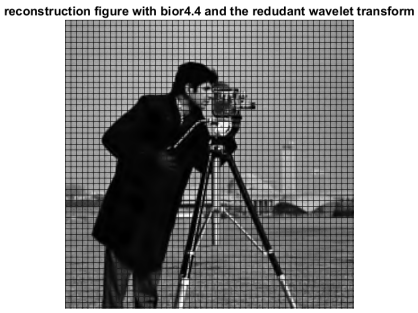
A denoising scheme removes information, usually the noise on the signal. In the case of the grid, we want to recover the image below the grid. However the grid consists of discontinuities. As seen before the discontinuities are less filtered and are responsible for the ringing effect. We conclude that the image below the grid will not be recovered. The grid will have ringing effect following that the rest of the image becomes fuzzy.



(a) Grid over image



(b) Reconstruction with db4



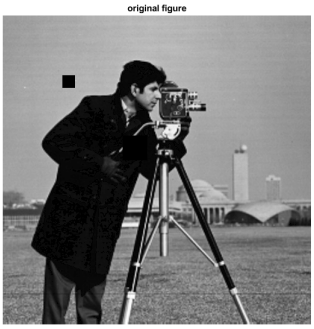
(c) Reconstruction with bior4.4

Figure 18

2 Wavelet-based inpainting

2.1 Task 3.1

As initial guess for B, we took the starting image. Furthermore as stopcriterium we used the maximum difference between the new matrix B and previous B. Whenever the matrix doesn't change much anymore, the algorithm is stopped. The images for which the experiments are conducted on is shown in figure 19. Furthermore, we chose for two different images with different gap locations, since the location of the gap determines the accuracy of the approximation. For example the larger square in 19a is very plain except for one line in the original figure. In the approximation the line can be missed easily. Whereas the small square in the background is easily approximated because it is plain everywhere.



(a) Image 1

(b) Image 2

Figure 19

2.2 Question 3.2

The results of the inpainting algorithm are shown in figures 20 and 21. For image 2 the stop criterium is stricter, more iterations. For the reason that in image 2 the results got better when using a stricter criterium. For image 1 the results remained largely the same even with a stricter criterium. The hard threshold performs better than the soft threshold. This is in line with what we concluded in previous sections. This is confirmed by the tables in 22 and 23 which show the signal-to-noise ratio.

(a) Image 1: db4

(b) Image 1: bior4

(c) Image 2: db4

(d) Image 2: bior4

Figure 20: Inpainting with different redundant wavelet transforms with hard thresholding

(a) Image 1: db4

(b) Image 1: bior4

(c) Image 2: db4

(d) Image 2: bior4

Figure 21: Inpainting with different redundant wavelet transforms with soft thresholding

Figure 22: Image 1: Signal to noise ratio (redundant scheme)

	Soft	Hard
db4	32.9728	33.9853
bior4.4	33.4271	33.7611

Figure 23: Image 2: Signal to noise ratio (redundant scheme)

	Soft	Hard
db4	25.6755	26.7622
bior4.4	25.3008	26.2859

2.3 Question 3.3

In figure 24 the results with the non-redundant wavelet transform are shown. The results of inpainting with the redundant scheme reached faster the required criterium, whereas the results with the non-redundant scheme needed the maximum number of iterations. Even with the maximum number of iterations the result is worse. For example in 21d the lines of the tripod are partially reconstructed. However, this is not the case in figure 24d.

(a) Image 1: db4

(b) Image 1: bior4

(c) Image 2: db4

(d) Image 2: bior4

Figure 24: Inpainting with different non-redundant wavelet transforms with hard thresholding

Figure 25: Image 1: Signal to noise ratio (hard threshold)

	redundant	non-redundant
db4	33.9853	30.2691
bior4.4	33.7611	30.2217

Figure 26: Image 2: Signal to noise ratio (hard threshold)

	redundant	non-redundant
db4	26.7622	24.4478
bior4.4	26.2859	22.9081

2.4 Question 3.4

From the images above, we don't see significant differences between Daubechies 4 and Biorthogonal 4.4. Looking to the signal-to-noise ratio Daubechies perform slightly better. But visually different conclusions can be made.

3 Final remarks

- The visual results of the code were usually in line with our expectations based on the theory from the course notes. Therefore, we are convinced that our implementations are correct.
- Some extra information about specific functions is provided in the comments of the code, see A.

A Appendix

A.1 Question 2.1

```
clear; close all;

%% Q2.1

% clean function
n = 10;
N=2^n;
x = linspace(-2,2,N);
fx = piecSmothFunc(x);
foriginal = fx;

% DWT mode (= Discrete wavelet transform extension mode)
% provides options for dealing with the problem of
% border distortion in signal or image analysis
dwtmode('symh') %default mode 'symh'

% wavelet and scaling coefficients
% wavedec performs a wavelet decomposition
% appcoef returns the scaling coefficients
% retrieve other coefficients (the wavelet coefficients)
% by c2(numel(s2)+1:end)
[c2,l2] = wavedec(fx,n,'db2');
s2 = appcoef(c2,l2,'db2');
w2 = c2(numel(s2)+1:end);

[c4,l4] = wavedec(fx,n,'db4');
s4 = appcoef(c4,l2,'db4');
w4 = c4(numel(s4)+1:end);

[c6,l6] = wavedec(fx,n,'db6');
s6 = appcoef(c6,l2,'db6');
w6 = c6(numel(s6)+1:end);


figure()
semilogy(abs(c2))
hold on
semilogy(abs(c4))
semilogy(abs(c6))
for i = 1:numel(l2)
    hold on
    xline(l2(i),"--");
```

```
43 end
44 xlabel('coefficient index')
45 ylabel('wavelet coefficient value')
46 xlim([0 l2(end)])
47 title('Coefficients per level')
48 legend('db2','db4','db6')
49
50 %% Support on different order
51 figure()
52 [c,l] = wavedec(fx,n,'db2');
53 w = abs(detcoef(c,l,3));
54 x = linspace(-2,2,numel(w));
55 semilogy(x,w)
56 hold on
57 [c,l] = wavedec(fx,n,'db4');
58 w = abs(detcoef(c,l,3));
59 x = linspace(-2,2,numel(w));
60 semilogy(x,w)
61 [c,l] = wavedec(fx,n,'db6');
62 w = abs(detcoef(c,l,3));
63 x = linspace(-2,2,numel(w));
64 semilogy(x,w)
65 xline(0);
66 xline(1);
67 xlabel('x')
68 ylabel('wavelet coefficient value')
69 title('Wavelet coefficients level 3: different order')
70 legend('order 2','order 4','order 6')
71
72 %% Different boundary conditions
73
74 figure()
75 dwtmode('per')
76 [c2,l2] = wavedec(fx,n,'db4');
77 w = abs(detcoef(c2,l2,3));
78 x = linspace(-2,2,numel(w));
79 semilogy(x,w)
80 hold on
81 dwtmode('symh')
82 [c2,l2] = wavedec(fx,n,'db4');
83 w = abs(detcoef(c2,l2,3));
84 x = linspace(-2,2,numel(w));
85 semilogy(x,w)
86 dwtmode('zpd')
87 [c2,l2] = wavedec(fx,n,'db4');
88 w = abs(detcoef(c2,l2,3));
```

```
x = linspace(-2,2,numel(w));
semilogy(x,w)
xline(0);
xline(1);
xlabel('x')
ylabel('wavelet coefficient value')
title('Wavelet coefficients level 3: different boundary
    conditions')
legend('per','symh','zpd')


%% Functions

% constructs function described in 2.1
function y = piecSmothFunc(x)
    y=(2+cos(x)).*abs(x).*sig(x-1);
end

%sign function 1 x>=0 -1 x<0
function y = sig(x)
    for i = 1:numel(x)
    if x(i)>=0
        y(i)=1;
    else
        y(i)=-1;
    end
    end
end
```

A.2 Task 2.2 and Question 2.3

```
clear; close all;

%% Task 2.2

n = 10;
N=2^n;
x = linspace(-2,2,N);
fx = piecSmothFunc(x);
foriginal = fx;

% contaminate function with noise
epsilon = 0.1;
for i=1:size(fx,2)
    fx(i)=fx(i)+epsilon*(-0.5+rand());
```

```
end
foriginalNoise = fx;
figure()
plot(fx)
title('Original function with noise')

% take wavelet transform of noisy function
[c,l] = wavedec(fx,4,'db4');
s = appcoef(c,l,'db4');
w = c(numel(s)+1:end);

threshHold = 1e-2*max(c);

wd_Soft = softThresholding(w,threshHold);
sd_Soft = softThresholding(s,threshHold);

wd_Hard = hardThresholding(w,threshHold);
sd_Hard = hardThresholding(s,threshHold);


c_rec_SS = [sd_Soft,wd_Soft];
c_rec_SH = [sd_Soft,wd_Hard];
c_rec_HS = [sd_Hard,wd_Soft];
c_rec_HH = [sd_Hard,wd_Hard];

fdenoised_SS = waverec(c_rec_SS,l,'db4');
fdenoised_HH = waverec(c_rec_HH,l,'db4');
fdenoised_SH = waverec(c_rec_SH,l,'db4');
fdenoised_HS = waverec(c_rec_HS,l,'db4');


figure()
plot(x,fdenoised_SS)
hold on
plot(x,fdenoised_HH, '--')
plot(x,fdenoised_SH,'-.')
plot(x,fdenoised_HS,'--')
plot(x,foriginal)
legend('Soft-Soft thresholding','Hard-Hard thresholding',
    'Soft-Hard thresholding','Hard-Soft thresholding','
    original')
title("Denoised signal with threshold 1e-2*max(c)")


sum(abs(foriginal-fdenoised_SS))/numel(foriginal)
sum(abs(foriginal-fdenoised_HH))/numel(foriginal)
```

```
sum(abs(foriginal-fdenoised_SH))/numel(foriginal)
sum(abs(foriginal-fdenoised_HS))/numel(foriginal)

%%
figure()
w = abs(detcoef(c_rec_HH,1,4));
x = linspace(-2,2,numel(w));
plot(x,w)
xlim([-2  2])
xlabel('x')
ylabel('wavelet coefficient')
title('wavelet coefficient denoised scheme')

%% Functions

function y = piecSmothFunc(x)
    y=(2+cos(x)).*abs(x).*sig(x-1);
end

%sign function 1 x>=0 0 x<0
function y = sig(x)
    for i = 1:numel(x)
    if x(i)>=0
        y(i)=1;
    else
        y(i)=-1;
    end
    end
end

function y = softThresholding(x,delta)
y = zeros(size(x));
    for i = 1:numel(x)
    if abs(x(i))>=delta
        y(i)=sig(x(i)).*(abs(x(i))-delta);
    else
        y(i)=0;
    end
    end
end

function y = hardThresholding(x,delta)
y = zeros(size(x));
for i = 1:numel(x)
    if abs(x(i))>=delta
        y(i)=x(i);
```

```
105         else
106             y(i)=0;
107         end
108     end
109     end
```

A.3 Task 2.4

```
1   % Implements the denoising scheme based on non redundant
        wavelets
2   % A is the coefficients matrix
3   % soft is a boolean, true is soft thresholding
4   % false is hard thresholding
5   % level is the level used in wavedec2
6   % wavelet is the family wavelet used
7   % thresholdfactor is during thresholding
8   % multiplied with the maximum value of the coefficients
9   function [A] = denoisingScheme(A, soft, level, wavelet,
        thresholdFactor)
10      [C,S] = wavedec2(A,level,wavelet);
11
12      if soft == true
13          [threshold,Cnew] = softThreshold(C,
                thresholdFactor);
14      else
15          [threshold,Cnew] = hardThreshold(C,
                thresholdFactor);
16      end
17
18      A = waverec2(Cnew,S,wavelet);
19      end
20
21      %% functions
22
23  function [threshold,Cnew] = softThreshold(C,
        thresholdFactor)
24      T = max(abs(C));
25      threshold = thresholdFactor*T;
26      Cnew = C;
27      for i=1:numel(C)
28          if abs(C(i)) < threshold
29              Cnew(i) = 0;
30          else
31              Cnew(i) = sign2(C(i))*(abs(C(i)) - threshold)
                    ;
```

```
        end
    end
end

function [threshold,Cnew] = hardThreshold(C,
    thresholdFactor)
    T = max(abs(C));
    threshold = thresholdFactor*T;
    Cnew = C;
    for i=1:numel(C)
        if abs(C(i)) < threshold
            Cnew(i) = 0;
        end
    end
end
```

```
% Implements the denoising scheme based on non redundant
    wavelets
% The difference with the previous version
% thresholdfactor is not given by the use
function [A] = nonRedudantDenoising(A, soft, level,
    wavelet)
    [C,S] = wavedec2(A,level,wavelet);

    if soft == true
        [threshold,Cnew] = softThreshold(C);
    else
        [threshold,Cnew] = hardThreshold(C);
    end

    A = waverec2(Cnew,S,wavelet);
    end

    %% functions

function [threshold,Cnew] = softThreshold(C)
    T = max(abs(C));
    threshold = 0.02*T;
    Cnew = C;
    for i=1:numel(C)
        if abs(C(i)) < threshold
            Cnew(i) = 0;
        else
            Cnew(i) = sign2(C(i))*(abs(C(i)) - threshold)
                ;
        end
```

```
    end
end

function [threshold,Cnew] = hardThreshold(C)
    T = max(abs(C));
    threshold = 0.02*T;
    Cnew = C;
    for i=1:numel(C)
        if abs(C(i)) < threshold
            Cnew(i) = 0;
        end
    end
end
```

A.4 Question 2.5

```
% Shows the original image, the image with noise and the
    denoised image.
% To obtain figures with other properties, change
% boolean, wavelet name and thresholdfactor
% in the function denoisingScheme();
clear; close all

% showing original image
[A,cmap] = imread('cameraman.png');

A = convertAtoActualColors(A, cmap);

figure
imshow(A)
title('Original figure')

% adding noise to the image
epsilon = 0.3;
for i=1:size(A,1)
    for j=1:size(A,2)
        A(i,j)=A(i,j)+epsilon*(-0.5+rand());
    end
end
figure
imshow(A)
title('Original image with noise')

B = denoisingScheme(A, false, 4, 'haar', 0.02);
SNR = signalToNoiseRatio(A, B);
```

```
29
30 figure
31 imshow(B)
32 title('Denoised image (hard, haar)')
```

A.5 Task 2.6

```
1  % Implements the denoising scheme based on redundant
       wavelets
2  % A is the coefficients matrix
3  % soft is a boolean, true is soft thresholding
4  % false is hard thresholding
5  % level is the level used in wavedec2
6  % wavelet is the family wavelet used
7  % thresholdfactor is during thresholding
8  % multiplied with the maximum value of the coefficients
9  function [A] = redudantDenoisingScheme(A, soft, level,
       wavelet, thresholdFactor)
10     [C,H,V,D] = swt2(A,level,wavelet);
11
12     if soft
13         [threshold, Cnew, Hnew, Vnew, Dnew] =
               softThresholding(C,H,V,D, thresholdFactor);
14     else
15         [threshold, Cnew, Hnew, Vnew, Dnew] =
               hardThreshold(C,H,V,D, thresholdFactor);
16     end
17     figure()
18     spy(Dnew)
19     figure()
20     spy(Hnew)
21     figure()
22     spy(Vnew)
23     A = iswt2(Cnew,Hnew,Vnew,Dnew,wavelet);
24
25     end
26
27     %% functions
28
29 function [threshold, Cnew, Hnew, Vnew, Dnew] =
       softThresholding(C,H,V,D, thresholdFactor)
30     bigTensor = cat(3,C,H,V,D);
31     threshold = thresholdFactor* max(max(max(abs(
           bigTensor))));
32     Cnew = C;
```

```
33      Hnew = H;
34      Vnew = V;
35      Dnew = D;
36      for i=1:size(C,1)
37          for j=1:size(C,2)
38              for k=1:size(C,3)
39                  if abs(C(i,j,k)) < threshold
40                      Cnew(i,j,k) = 0;
41                  else
42                      Cnew(i,j,k) = sign2(C(i,j,k))*(abs(C(
                            i,j,k)) - threshold);
43                  end
44                  if abs(H(i,j,k)) < threshold
45                      Hnew(i,j,k) = 0;
46                  else
47                      Hnew(i,j,k) = sign2(H(i,j,k))*(abs(H(
                            i,j,k)) - threshold);
48                  end
49                  if abs(V(i,j,k)) < threshold
50                      Vnew(i,j,k) = 0;
51                  else
52                      Vnew(i,j,k) = sign2(V(i,j,k))*(abs(V(
                            i,j,k)) - threshold);
53                  end
54                  if abs(D(i,j,k)) < threshold
55                      Dnew(i,j,k) = 0;
56                  else
57                      Dnew(i,j,k) = sign2(D(i,j,k))*(abs(D(
                            i,j,k)) - threshold);
58                  end
59              end
60          end
61      end
62  end
63
64  function [threshold, Cnew, Hnew, Vnew, Dnew] =
        hardThreshold(C,H,V,D, thresholdFactor)
65      bigTensor = cat(3,C,H,V,D);
66      threshold = thresholdFactor* max(max(max(abs(
            bigTensor))));
67      Cnew = C;
68      Hnew = H;
69      Vnew = V;
70      Dnew = D;
71      for i=1:size(C,1)
72          for j=1:size(C,2)
```

```
            for k=1:size(C,3)
                if abs(C(i,j,k)) < threshold
                    Cnew(i,j,k) = 0;
                end
                if abs(H(i,j,k)) < threshold
                    Hnew(i,j,k) = 0;
                end
                if abs(V(i,j,k)) < threshold
                    Vnew(i,j,k) = 0;
                end
                if abs(D(i,j,k)) < threshold
                    Dnew(i,j,k) = 0;
                end
            end
        end
    end
end
```

```
% Implements the denoising scheme based on redundant
    wavelets
% The difference with the previous version
% thresholdfactor is not given by the use
function [A] = redudantDenoising(A, soft, level, wavelet)
    [C,H,V,D] = swt2(A,level,wavelet);

    if soft
        [threshold, Cnew, Hnew, Vnew, Dnew] =
            softThresholding(C,H,V,D);
    else
        [threshold, Cnew, Hnew, Vnew, Dnew] =
            hardThreshold(C,H,V,D);
    end
    A = iswt2(Cnew,Hnew,Vnew,Dnew,wavelet);

    end

    %% functions

function [threshold, Cnew, Hnew, Vnew, Dnew] =
    softThresholding(C,H,V,D)
    bigTensor = cat(3,C,H,V,D);
    threshold = 0.02* max(max(max(abs(bigTensor))));
    Cnew = C;
    Hnew = H;
    Vnew = V;
    Dnew = D;
```

```
25      for i=1:size(C,1)
26          for j=1:size(C,2)
27              for k=1:size(C,3)
28                  if abs(C(i,j,k)) < threshold
29                      Cnew(i,j,k) = 0;
30                  else
31                      Cnew(i,j,k) = sign2(C(i,j,k))*(abs(C(
                            i,j,k)) - threshold);
32                  end
33                  if abs(H(i,j,k)) < threshold
34                      Hnew(i,j,k) = 0;
35                  else
36                      Hnew(i,j,k) = sign2(H(i,j,k))*(abs(H(
                            i,j,k)) - threshold);
37                  end
38                  if abs(V(i,j,k)) < threshold
39                      Vnew(i,j,k) = 0;
40                  else
41                      Vnew(i,j,k) = sign2(V(i,j,k))*(abs(V(
                            i,j,k)) - threshold);
42                  end
43                  if abs(D(i,j,k)) < threshold
44                      Dnew(i,j,k) = 0;
45                  else
46                      Dnew(i,j,k) = sign2(D(i,j,k))*(abs(D(
                            i,j,k)) - threshold);
47                  end
48              end
49          end
50      end
51  end
52
53  function [threshold, Cnew, Hnew, Vnew, Dnew] =
        hardThreshold(C,H,V,D)
54      bigTensor = cat(3,C,H,V,D);
55      threshold = 0.02* max(max(max(abs(bigTensor))));
56      Cnew = C;
57      Hnew = H;
58      Vnew = V;
59      Dnew = D;
60      for i=1:size(C,1)
61          for j=1:size(C,2)
62              for k=1:size(C,3)
63                  if abs(C(i,j,k)) < threshold
64                      Cnew(i,j,k) = 0;
65                  end
```

```
                    if abs(H(i,j,k)) < threshold
                        Hnew(i,j,k) = 0;
                    end
                    if abs(V(i,j,k)) < threshold
                        Vnew(i,j,k) = 0;
                    end
                    if abs(D(i,j,k)) < threshold
                        Dnew(i,j,k) = 0;
                    end
                end
            end
        end
    end
end
```

```
%sign function 1 x>=0 -1 x<0
function s = sign2(a)
    if a >= 0
        s=1;
    else
        s=-1;
    end
end
```

A.6 Question 2.7

```
% Shows the original image, the image with noise and the
    denoised image.
% To obtain figures with other properties, change
% boolean, wavelet name and thresholdfactor
% in the function denoisingScheme();
clear; close all

[A,cmap] = imread('cameraman.png');

A = convertAtoActualColors(A, cmap);

figure
imshow(A)
title('Original figure')

% adding noise
epsilon = 0.3;
for i=1:size(A,1)
    for j=1:size(A,2)
        A(i,j)=A(i,j)+epsilon*(-0.5+rand());
```

```
    end
end
figure
imshow(A)
title('Original figure with noise')

B = redudantDenoisingScheme(A, false, 4, 'bior4.4', 0.02);
SNR = signalToNoiseRatio(A, B);

figure
imshow(B)
title('Denoised image with redundant wavelet transform (hard, bior4.4)')
```

A.7 Question 2.8

```
clear; close all;

% clean function
n = 10;
N=2^n;
x = linspace(-2,2,N);
fx = piecSmothFunc(x);
foriginal = fx;

figure()
plot(x,fx)
title('Original function')

%% Task 2.8

% contaminate function with noise
epsilon = 0.1;
for i=1:size(fx,2)
    fx(i)=fx(i)+epsilon*(-0.5+rand());
end
foriginalNoise = fx;
figure()
plot(fx)
title('Original function with noise')

% take wavelet transform of noisy function
swc = swt(fx,4,'db4');

```

```
29 threshHold = 1e-2*max(max(swc));
30
31
32 swc_soft = softThresholding(swc,threshHold);
33
34 swc_hard = hardThresholding(swc,threshHold);
35
36 f_swc_soft = iswt(swc_soft, 'db4');
37 f_swc_hard = iswt(swc_hard, 'db4');
38
39 figure()
40 plot(x,f_swc_soft)
41 hold on
42 plot(x,f_swc_hard)
43 plot(x,foriginal)
44 legend('Soft thresholding','Hard thresholding','original'
       )
45 title("Denoised signal with threshold 1e-2*max(c)")
46
47
48 figure()
49 x = linspace(-2,2,size(swc,2));
50 plot(x,swc_soft(3,:))
51 xlabel('x')
52 ylabel('coefficients')
53 title("Coefficient redundant wavelet transform")
54
55
56 %% Functions
57
58 function y = piecSmothFunc(x)
59     y=(2+cos(x)).*abs(x).*sig(x-1);
60 end
61
62 %sign function 1 x>=0 0 x<0
63 function y = sig(x)
64     for i = 1:numel(x)
65     if x(i)>=0
66         y(i)=1;
67     else
68         y(i)=-1;
69     end
70     end
71 end
72
73 function y = softThresholding(x,delta)
```

```
74      y = size(x);
75      for i = 1:size(x,1)
76          for j = 1:size(x,2)
77              if abs(x(i,j))>=delta
78                  y(i,j)=sig(x(i,j)).*(abs(x(i,j))-delta);
79              else
80                  y(i,j)=0;
81              end
82          end
83      end
84  end
85
86  function y = hardThresholding(x,delta)
87      y = (x);
88      for i = 1:size(x,1)
89          for j = 1:size(x,2)
90              if abs(x(i,j))<delta
91                  y(i,j)=0;
92              end
93          end
94      end
95  end
```

A.8 Question 2.9

```
1  clear; close all;
2
3  [A,cmap] = imread('cameraman.png');
4
5  A = convertAtoActualColors(A, cmap);
6
7  figure
8  imshow(A)
9  title('original figure')
10
11 % toevoegen grid
12 A(1:10:end,:)=0;
13 A(:,1:10:end)=0;
14
15 B = A;
16
17 figure
18 imshow(B)
19 title('grid')
20
```

```
A = redudantDenoising(A, false, 4, 'db4');
B = A;

figure
imshow(B)
title('reconstruction figure with db4 and the redudant
    wavelet transform')
```

A.9 Code for whole section 3

```
% implementation of the inpainting algorithm
clear;
close all

[A,cmap] = imread('cameraman.png');

A = convertAtoActualColors(A, cmap);
A_original = A;

figure
imshow(A)
title('original figure')

% masks maken -> kies lege plekken
mask = zeros(size(A));
mask(200:240, 200:240) = 1;
mask(100:120,100:120) = 1;
mask = mask > 0;
A(mask) = 0;

figure
imshow(A)
title('original figure')

% mask complementMask
complementMask = mask == 0;

B = A;
B_pre=A;
i=1;
% while (max(max(abs(B-B_pre))) > 1e-3 || i==1) && i<200
while (max(max(abs(B-B_pre))) > 1e-2 || i==1) && i<100
    K = redudantDenoising(B, true, 4, 'db4');
    K(complementMask) = 0;
    B_pre = B;
```

```
36        B = A + K;
37        i = i + 1;
38  end
39
40   SNR = signalToNoiseRatio(A_original,B)
41
42  figure
43  imshow(B)
```