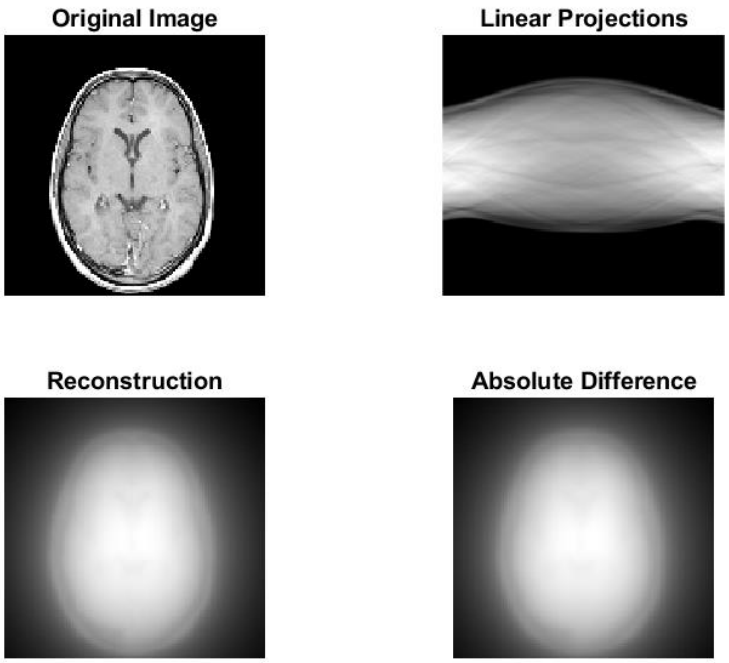
ELEC421 – HW5 - Or Bahari 51277200

1.a.

Running the code with default values results in:



We can see that the SNR is bad, the difference plot has many values (means many details are missing). The reconstruction is not like the original image.

1.b.

You asked for it. These are the 72 plots (36 options, 2 for each – a plot and an Abs difference graph).

Max MSE:

In purple. SPLINE and none.

Value:481.629

Min MSE:

In red. SPLINE and Ram-Lak.

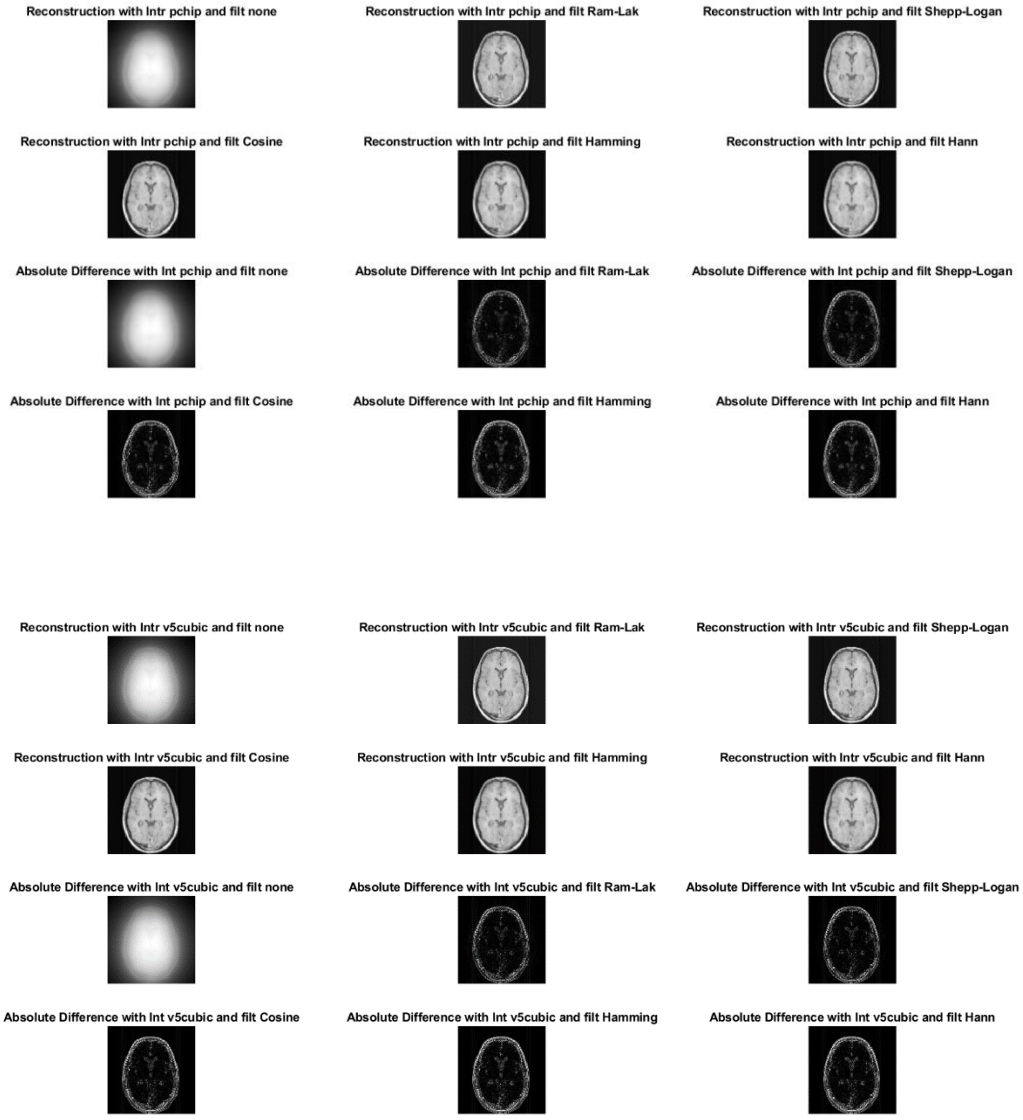
Value: **0.00031**

And not surprisingly the max SNR is the min MSE and vice versa..

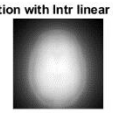
Min SNR: -98.98

Max SNR: 43.549

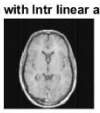
My observation – many pictures have pretty much the same clarity and look like the original image. We can see that all the “none” interpolations result in a bad reconstructed result.



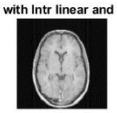
Reconstruction with Intr linear and filt none



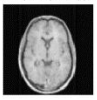
Reconstruction with Intr linear and filt Ram-Lak



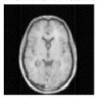
Reconstruction with Intr linear and filt Shepp-Logan



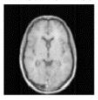
Reconstruction with Intr linear and filt Cosine



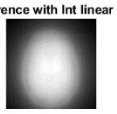
Reconstruction with Intr linear and filt Hamming



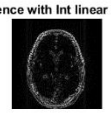
Reconstruction with Intr linear and filt Hann



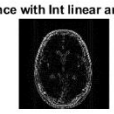
Absolute Difference with Int linear and filt none



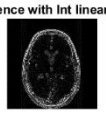
Absolute Difference with Int linear and filt Ram-Lak



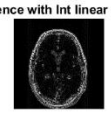
Absolute Difference with Int linear and filt Shepp-Logan



Absolute Difference with Int linear and filt Cosine

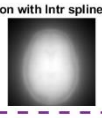


Absolute Difference with Int linear and filt Hamming

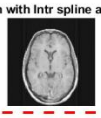


Absolute Difference with Int linear and filt Hann

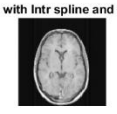
Reconstruction with Intr spline and filt none



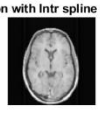
Reconstruction with Intr spline and filt Ram-Lak



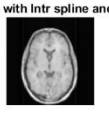
Reconstruction with Intr spline and filt Shepp-Logan



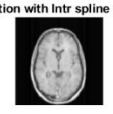
Reconstruction with Intr spline and filt Cosine



Reconstruction with Intr spline and filt Hamming

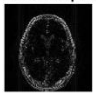


Reconstruction with Intr spline and filt Hann

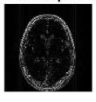


Absolute Difference with Int spline and filt none

Absolute Difference with Int spline and filt Ram-Lak



Absolute Difference with Int spline and filt Shepp-Logan



Absolute Difference with Int spline and filt Cosine

Absolute Difference with Int spline and filt Hamming

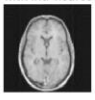
Absolute Difference with Int spline and filt Hann

Reconstruction with Intr nearest and filt none

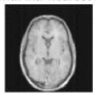
Reconstruction with Intr nearest and filt Ram-Lak

Reconstruction with Intr nearest and filt Shepp-Logan

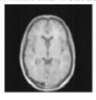
Reconstruction with Intr nearest and filt Cosine



Reconstruction with Intr nearest and filt Hamming

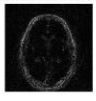


Reconstruction with Intr nearest and filt Hann

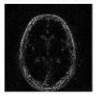


Absolute Difference with Int nearest and filt none

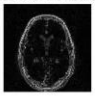
Absolute Difference with Int nearest and filt Ram-Lak



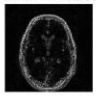
Absolute Difference with Int nearest and filt Shepp-Logan



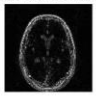
Absolute Difference with Int nearest and filt Cosine

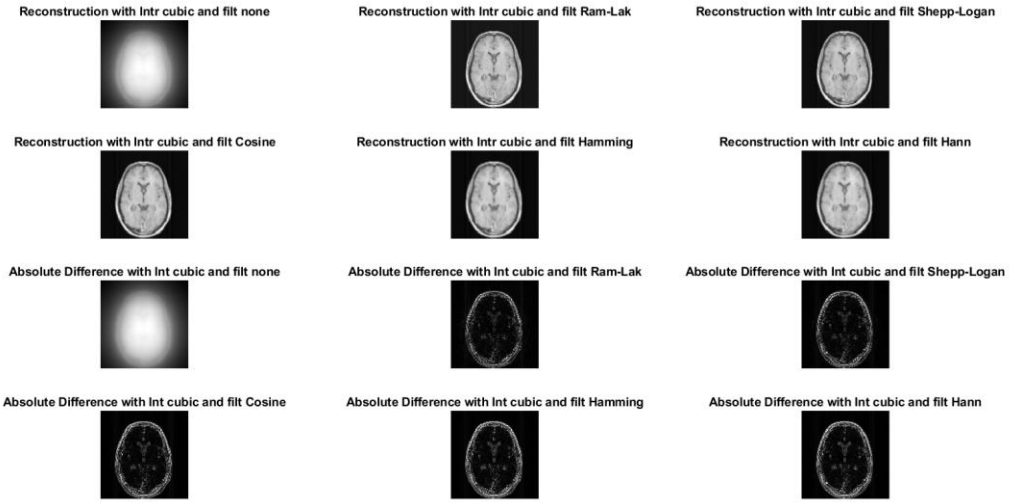


Absolute Difference with Int nearest and filt Hamming



Absolute Difference with Int nearest and filt Hann





MSE results:

481.614166346777 0.000383655341745798 0.000459481614499700
0.000658018950126071 0.000798554148115775 0.000849705593375986
481.585330950280 0.000429838906005806 0.000513225259860759
0.000703129313007628 0.000842245951880782 0.000887861617907434
481.629646370018 0.000311116751578031 0.000395225056647761
0.000601189374536815 0.000748360151734346 0.000800353847663710
481.623022547415 0.000356807822927033 0.000438463277566712
0.000632842070086016 0.000773940805711137 0.000822441491378505
481.623022547415 0.000356807822927033 0.000438463277566712
0.000632842070086016 0.000773940805711137 0.000822441491378505
481.629544808033 0.000340818729267310 0.000424387437431598
0.000624891425313060 0.000768268945673880 0.000818149769173848

SNR results:

-98.9757281350813 41.4533645117625 39.6498212416496 36.0584732269570
34.1227827554133 33.5019112443652 -98.9751293931471 40.3167055189226
38.5436620135436 35.3954023376076 33.5900897723039 33.0626515803131
-98.9760495495154 43.5491280445504 41.1562568687736 36.9617106922517
34.7719670481700 34.1002711373250 -98.9759120191909 42.1788372748059
40.1180499045655 36.4486015650962 34.4358566093429 33.8280370818283
-98.9759120191909 42.1788372748059 40.1180499045655 36.4486015650962
34.4358566093429 33.8280370818283 -98.9760474407996 42.6373030297465
40.4443424766211 36.5750313804369 34.5094119166491 33.8803564144698

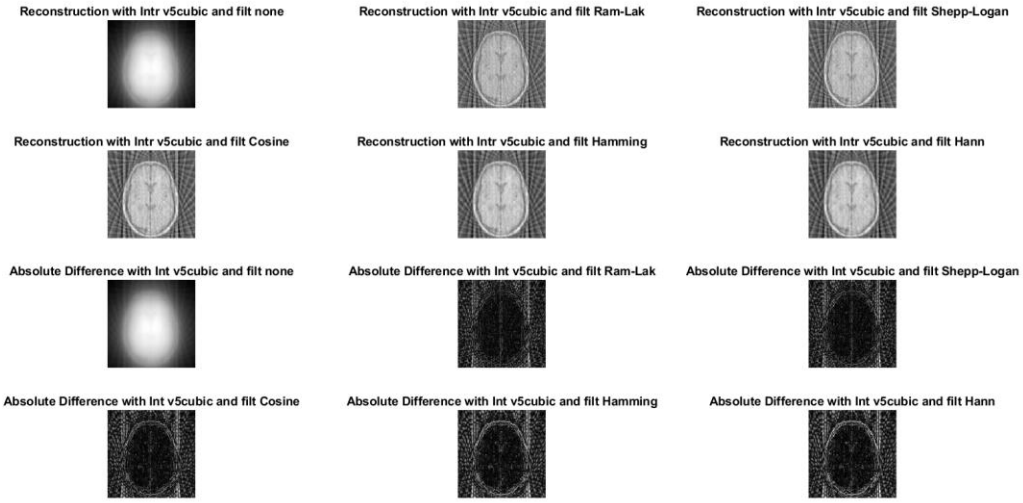
Code:

```
num_angles=200;
```

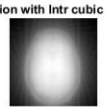
```
angles = linspace(0,180,num_angles);
R = radon(obj,angles);
% Plot the radon transform
%subplot(222);imshow(R,[]); title('Linear Projections');
% RECONSTRUCT FROM PROJECTIONS

options=["nearest","linear","spline","pchip","cubic","v5cubic"];
filters=["none","Ram-Lak","Shepp-Logan","Cosine","Hamming","Hann"];
outer=1;
MSE=[];SNR=[];
for i=options
    figure(outer);
    index=1;
    for filter=filters
    I = iradon(R,angles,i,filter,128); %back projection reconstruction
    subplot(4,3,index);imshow(I,[]); title(sprintf('Reconstruction with Intr %s and filt
%s',i,filter));
    SE = (I-obj).^2;
    MSE = [MSE,mean(SE(:))]; %Mean Squared Error
    SNR = [SNR,20*log(norm(obj,'fro')/norm(obj-I,'fro'))]; %Signal-to-Noise Ratio
    subplot(4,3,index+6);imshow(abs(I-obj),[]); title(sprintf('Absolute Difference with Int %s
and filt %s',i,filter));
    index=index+1;
    end
    outer=outer+1;
end
```

1.c.

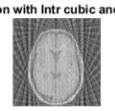
The results are:



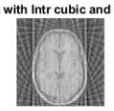
Reconstruction with Intr cubic and filt none



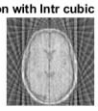
Reconstruction with Intr cubic and filt Ram-Lak



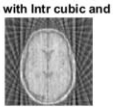
Reconstruction with Intr cubic and filt Shepp-Logan



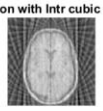
Reconstruction with Intr cubic and filt Cosine



Reconstruction with Intr cubic and filt Hamming



Reconstruction with Intr cubic and filt Hann



Absolute Difference with Int cubic and filt none

Absolute Difference with Int cubic and filt Ram-Lak

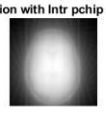
Absolute Difference with Int cubic and filt Shepp-Logan

Absolute Difference with Int cubic and filt Cosine

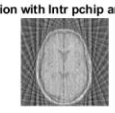
Absolute Difference with Int cubic and filt Hamming

Absolute Difference with Int cubic and filt Hann

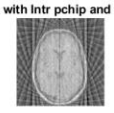
Reconstruction with Intr pchip and filt none



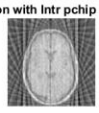
Reconstruction with Intr pchip and filt Ram-Lak



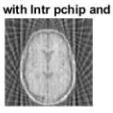
Reconstruction with Intr pchip and filt Shepp-Logan



Reconstruction with Intr pchip and filt Cosine

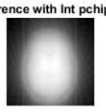


Reconstruction with Intr pchip and filt Hamming

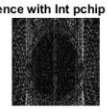


Reconstruction with Intr pchip and filt Hann

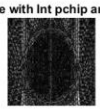
Absolute Difference with Int pchip and filt none



Absolute Difference with Int pchip and filt Ram-Lak



Absolute Difference with Int pchip and filt Shepp-Logan

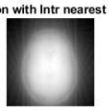


Absolute Difference with Int pchip and filt Cosine

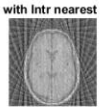
Absolute Difference with Int pchip and filt Hamming

Absolute Difference with Int pchip and filt Hann

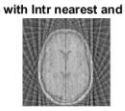
Reconstruction with Intr nearest and filt none



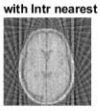
Reconstruction with Intr nearest and filt Ram-Lak



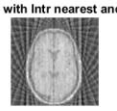
Reconstruction with Intr nearest and filt Shepp-Logan



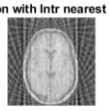
Reconstruction with Intr nearest and filt Cosine



Reconstruction with Intr nearest and filt Hamming



Reconstruction with Intr nearest and filt Hann



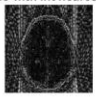
Absolute Difference with Int nearest and filt none

Absolute Difference with Int nearest and filt Ram-Lak

Absolute Difference with Int nearest and filt Shepp-Logan

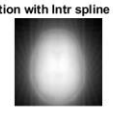
Absolute Difference with Int nearest and filt Cosine

Absolute Difference with Int nearest and filt Hamming

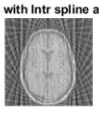


Absolute Difference with Int nearest and filt Hann

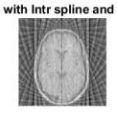
Reconstruction with Intr spline and filt none



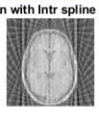
Reconstruction with Intr spline and filt Ram-Lak



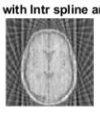
Reconstruction with Intr spline and filt Shepp-Logan



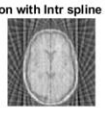
Reconstruction with Intr spline and filt Cosine



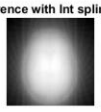
Reconstruction with Intr spline and filt Hamming



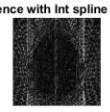
Reconstruction with Intr spline and filt Hann



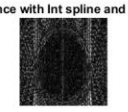
Absolute Difference with Int spline and filt none



Absolute Difference with Int spline and filt Ram-Lak



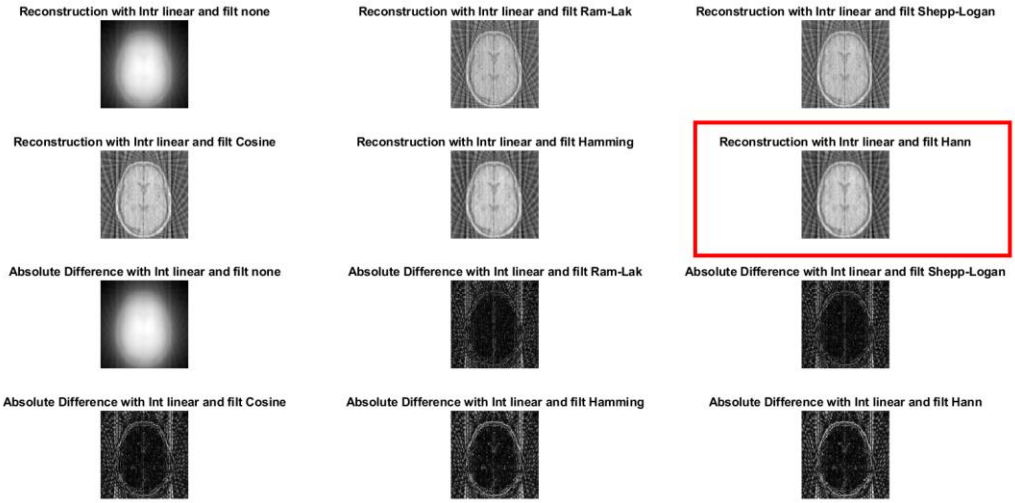
Absolute Difference with Int spline and filt Shepp-Logan



Absolute Difference with Int spline and filt Cosine

Absolute Difference with Int spline and filt Hamming

Absolute Difference with Int spline and filt Hann



Code used to decide the max and min MSE (I did a clever move and made the MSE and SNR arrays such that each index matches a different set of parameters).

```
>> a=find(MSE==min(MSE))

a =

   12

>> b=find(MSE==max(MSE))

b =

   13
```

Max MSE (index 13):

purple image – spline and none.

value 483.890

and the min SNR (same image): -99.022

Min MSE (index 12):

red image – linear and Hann

Value **0.00280**

And the **max SNR** (the same image): **21.545**

**To sum up, there is a difference in Min MSE and max SNR values between part c and part b. A reasonable explanation could be differences between the algorithms of the methods (one can be more angular and the other less, so with less angles we don't get as accurate as we should have).

With more angles the best reconstruction plot is a curve('spline') and with less angles the best is linear, which goes along with the fact that the spline is piecewise polynomial.

SNR results:

-99.0227272319282	15.5942412265074	17.0285151354942	19.2369667428185	20.6705024655802	20.9136939081375	-
99.0219807095574	17.2936658491232	18.4300974577866	20.1641439986438	21.3525329550010	21.5450370713936	-
99.0228898821205	16.2005517292580	17.5238011849147	19.5817629131172	20.9760120005913	21.2064413878610	-
99.0227428867012	16.6717555761509	17.8985882339680	19.7963613807806	21.1043188774523	21.3186682596655	-
99.0227428867012	16.6717555761509	17.8985882339680	19.7963613807806	21.1043188774523	21.3186682596655	-
99.0228740598757	16.4728253028871	17.7372399695663	19.6934252038668	21.0338607385082	21.2546233190283	

MSE results:

483.883036998595	0.00509317623946247	0.00441264490181498	0.00353823883787988
0.00306569877171443	0.00299204285556634	483.846915395698	0.00429718084091336
0.00383556222770282	0.00322493041845948	0.00286357968044723	0.00280898179477946
483.890907429503	0.00479354682192471	0.00419941678828990	0.00341832095912555
0.00297345499339773	0.00290572124158616	483.883794507098	0.00457291207082121
0.00404494094464714	0.00334574582596358	0.00293554723240047	0.00287329354436072
483.883794507098	0.00457291207082121	0.00404494094464714	0.00334574582596358
0.00293554723240047	0.00287329354436072	483.890141806071	0.00466479199051431
0.00411073472299763	0.00338036351936968	0.00295630358872199	0.00289175458953382

Code:

```
num_angles=20;
```

2.a.

The signal strength, I, is determined by:

$I = I_0 e^{\int \mu(x,y)ds}$

Where μ is the attenuation coefficient.

2.b.

As we learned in class, the relation to Radon Transform is:

$Rad(\theta, \rho) = \ln(\frac{I_0}{I})$

2.c

Point a$\left(\frac{\pi}{4}, 7\right)$:

According to what we studied in class we get:

$\rho = xcos(\theta) + ysin(\theta)$

$7 = \frac{\sqrt{2}}{2}x + \frac{\sqrt{2}}{2}y$

Negative slope, when $x \cong 10$ $we\ get\ y = 0$. The graph which corresponds to negative slope is (B), and with these values the correct line is: I=1, I0=5.

$Rad\left(\frac{\pi}{4}, 7\right) = \ln\left(\frac{I_0}{I}\right) = \ln\left(\frac{5}{1}\right) = 1.609$

Point b$\left(\frac{\pi}{2}, 6\right)$:

$\rho = xcos(\theta) + ysin(\theta)$

$6 = 0x + 1y$

y=6, graph (C). y=6 has values: I=6, i0=8.

$$Rad\left(\frac{\pi}{2}, 6\right) = \ln\left(\frac{I_0}{I}\right) = \ln\left(\frac{8}{6}\right) = 0.287$$

Point c$(0,3)$:

$$\rho = xcos(\theta) + ysin(\theta)$$

$$3 = 1x + 0y$$

X=3, graph (C). x=3 has values: I=5, i0=7.

$$Rad(0,3) = \ln\left(\frac{I_0}{I}\right) = \ln\left(\frac{7}{5}\right) = 0.336$$

Point d$(\frac{3\pi}{4}, 3\sqrt{2})$:

$$\rho = xcos(\theta) + ysin(\theta)$$

$$3\sqrt{2} = -\frac{\sqrt{2}}{2}x + \frac{\sqrt{2}}{2}y$$

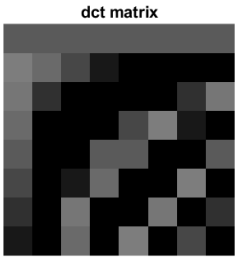
$$6 + x = y$$

When x=0 we get y=6. Slope is positive, so we pick from graph (A). the linear line that matches these values has: I=3, i0=5.

$$Rad(\frac{3\pi}{4}, 3\sqrt{2}) = \ln\left(\frac{I_0}{I}\right) = \ln\left(\frac{5}{3}\right) = 0.510$$

3.a.

8x8 dct matrix:



This is a transform similar in a way to the Fourier transform we know. But the upper left corner is dominant (instead of the center in Fourier), meaning all the low frequencies of the signal are concentrated around the upper left corner. It will help us discard some high frequency components (and compressing the image).

Code:

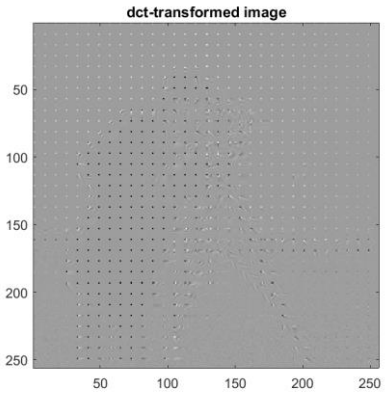
```
%3.a
figure(31);
dct=dctmtx(8);
imshow(dct);
title("dct matrix");
```

3.b.

We want to center the image around zero to reduce dynamic range.

3.c.

The transformed image is:



We can still see a bit of the original cameraman.

Code:

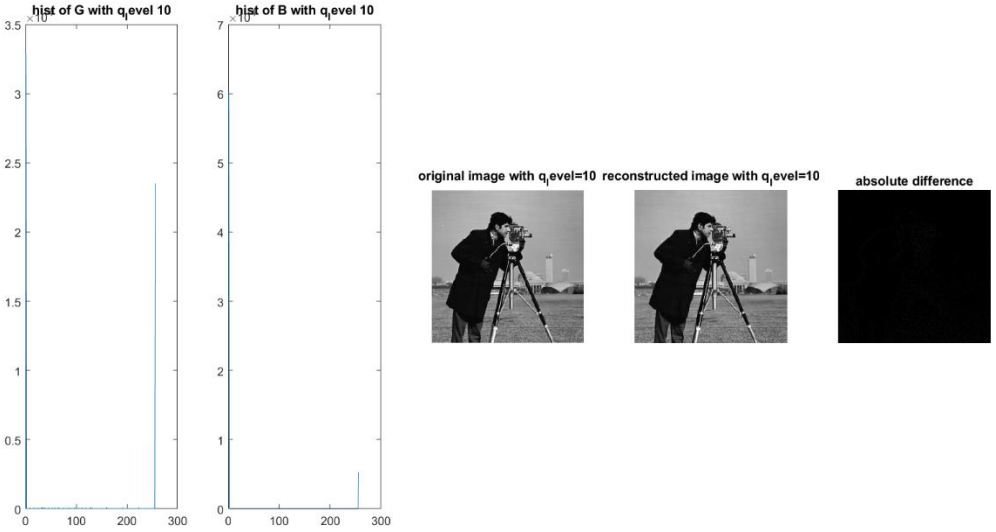
```
%3.c
img=imread('cameraman.tif');
img_mod=double(img)-128;
fun= @(block_struct) dct*block_struct.data*dct';
comp=blockproc(img_mod,[8 8],fun);
imagesc(comp);
colormap('gray');
axis image;
title('dct-transformed image');
```

3.d.

The quantization matrix wants to demolish /zero-round the high frequencies and keep the low frequencies (which are cantered in the upper left corner). That's why low values (values that are closer to 1) are positioned at the upper left corner (to keep those frequencies), and the 'discarding' level/strength rises as we move diagonally towards the right lower corner.

4.a.+b.

With q_level=10 we get:



As we can see, the histogram of the quantized signal (B) has more zeroes than those in G (as expected).

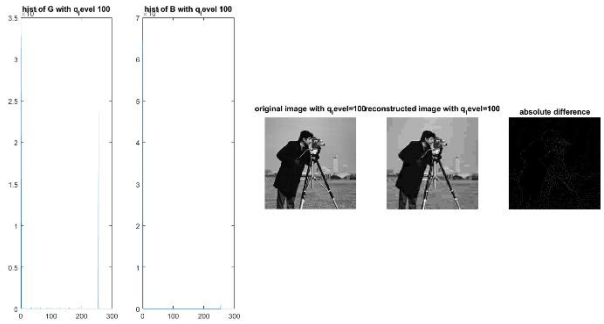
The reconstructed image is very similar to the original, we can see that the difference plot is almost all black (meaning there is almost no difference between the two images). Explanation: we zeroed out many values but kept most of the significant values in the signal hence the two images look pretty much the same.

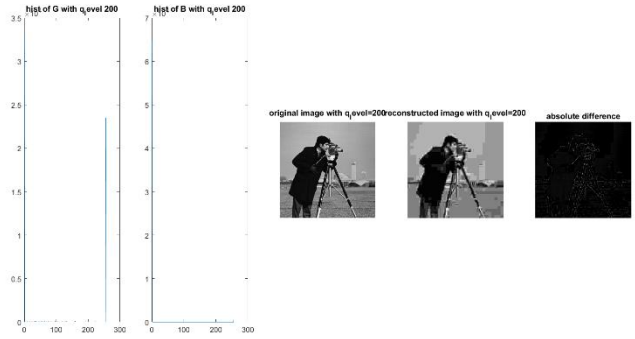
Code:

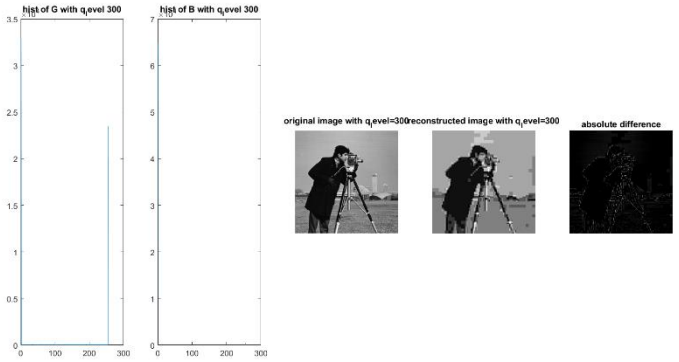
```
q_levels=[10];
Q=[1 1 1 2 2 2 4 4; 1 1 2 2 2 4 4 4;1 2 2 2 4 4 4 8; 2 2 2 4 4 4 8 8;2 2 4 4 4 8 8 8;2 4 4 4 8 8
8 16;4 4 4 8 8 8 16 16;4 4 8 8 8 16 16 16];

fig=1;
fun2= @(block_struct) round(block_struct.data./(q_level*Q));
B=blockproc(comp,[8 8],fun2);
h_g=imhist(comp);
h_b=imhist(B);
figure(fig);
subplot(1,5,1);
plot(h_g);
title(sprintf('hist of G with q_level %d',q_level));subplot(1,5,2);
plot(h_b);
title(sprintf('hist of B with q_level %d',q_level));
fun3= @(block_struct) block_struct.data.*(q_level*Q);
G1=blockproc(B,[8 8],fun3);
inverse_dct= @(block_struct) dct'*block_struct.data*dct;
G=blockproc(G1,[8 8],inverse_dct);
img_rec=uint8(G+128);
subplot(1,5,3);
imshow(img);
title(sprintf('original image with q_level=%d',q_level));
subplot(1,5,4);
imshow(img_rec);
title(sprintf('reconstructed image with q_level=%d',q_level));
delta=abs(img-img_rec);
subplot(1,5,5);
imshow(delta);
title('absolute difference');
```

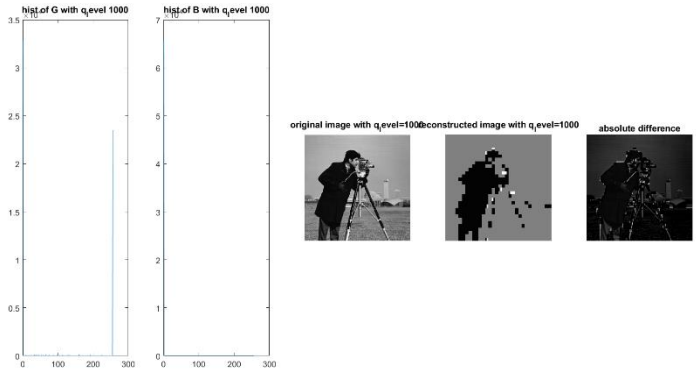
4.c.

For q_levels =100, 200, 300, 1000 we get these results:









As we increase `q_level` the histogram of B(quantized signal) has more zeros -> More significant values/frequencies of the original image are discarded and the reconstructed result become worse. (we compress more but the image is ruined… so it doesn't help),

Code:

```
q_levels=[10 100 200 300 1000];
Q=[1 1 1 2 2 2 4 4; 1 1 2 2 2 4 4 4;1 2 2 2 4 4 4 8; 2 2 2 4 4 4 8 8;2 2 4 4 4 8 8 8;2 4 4 4 8 8
8 16;4 4 4 8 8 8 16 16;4 4 8 8 8 16 16 16];

fig=1;
for q_level=q_levels
    fun2= @(block_struct) round(block_struct.data./(q_level*Q));
    B=blockproc(comp,[8 8],fun2);
    h_g=imhist(comp);
    h_b=imhist(B);
    figure(fig);
    subplot(1,5,1);
    plot(h_g);
    title(sprintf('hist of G with q_level %d',q_level));
    subplot(1,5,2);
    plot(h_b);
    title(sprintf('hist of B with q_level %d',q_level));
    fun3= @(block_struct) block_struct.data.*(q_level*Q);
    G1=blockproc(B,[8 8],fun3);
    inverse_dct= @(block_struct) dct'*block_struct.data*dct;
    G=blockproc(G1,[8 8],inverse_dct);
    img_rec=uint8(G+128);
    subplot(1,5,3);
    imshow(img);
    title(sprintf('original image with q_level=%d',q_level));
    subplot(1,5,4);
    imshow(img_rec);
    title(sprintf('reconstructed image with q_level=%d',q_level));
    delta=abs(img-img_rec);
    subplot(1,5,5);
    imshow(delta);
    title('absolute difference');
    fig=fig+1;
end
```

5.a

Information Entropy – stands for the minimal number of bits needed to present our data, the best-case for compressing our data.

5.b.

With q_levels=10, we get:

ent =

7.1056

The cameraman is a greyscale image with 8bits per pixel.

The maximum compression in % that can be achieved is:

$$comp_{rate} = 100 - \frac{7.1056}{8} * 100 = 11.18\%$$

Which means that we can reduce 11.18% of image size with best-case compression.

Code:

```
ent=entropy(img_rec);
```

5.c.

Measuring the number of bits in img_rec (before Huffman) and after applying Huffman coding result in:

Img_rec_Len =

524288

HuffmanLen =

467694

Meaning we compressed:

$$comp_{rate} = 100 - \frac{467694}{524288} * 100 = 10.79\%$$

So we compressed 10.79% of the signal using Huffman, or in other words that the number of information bits in Huffman is:

$$bits = \frac{467694}{524288} * 8 = 7.1364$$

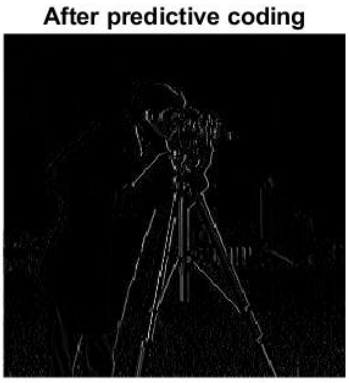
To sum up – Huffman **does not** achieve the minimal information bits (for this image).

Code:

```
%5
ent=entropy(img_rec);
binarySig = de2bi(img_rec);
Img_rec_Len = numel(binarySig)
symbols = unique(img_rec(:));
counts = imhist(img_rec(:), 256);
p = double(counts) ./ sum(counts);
[dict,avglen] = huffmandict(symbols,p);
comp = huffmanenco(img_rec(:),dict);
bits_huffman=(comp);
binaryComp = de2bi(comp);
HuffmanLen = numel(binaryComp)
```

6.a.

With q_level=10, after applying the code we get:



Well the resulting image looks like as we applied a vertical differential (1st derivative) on it.

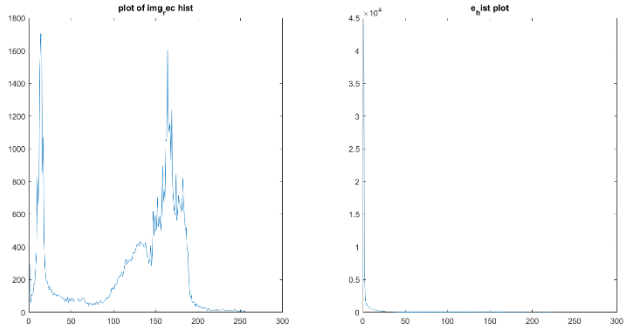
Code:

```
%6.a
alpha=1;
[R,C]=size(img_rec);
f_prime=uint8(zeros([R,C]));
for r=1:R
    for c=2:C
        f_prime(r,c)=round(alpha*img_rec(r,c-1));
    end
```

```
end
e=img_rec-f_prime;
figure(6);
subplot(1,2,1);
imshow(img_rec);
title('Before predictive coding');
subplot(1,2,2);
imshow(e);
title('After predictive coding');
```

6.b.

The two histograms:



The original histogram is more varied and the values are spread in the range [0 255] while in the encoded image the values are all close to 0.

Entropies:

>> entropy(img_rec)

ans =

7.1056

>> entropy(e)

ans =

2.4416

As expected the encoded image has a better entropy.

Code:

```
%6.b.
rec_hist=imhist(img_rec);
e_hist=imhist(e);
figure(62);
subplot(1,2,1);
plot(rec_hist);
title("plot of img_rec hist");
subplot (1,2,2);
plot(e_hist);
title("e_hist plot");
e_ent=entropy(e);
rec_ent=entropy(img_rec);
```

6.c.

Math:

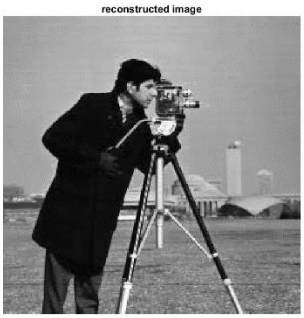
$e(x,y) = f(x,y) - \hat{f}(x,y) = f(x,y) - Round(\alpha f(x,y-1))$

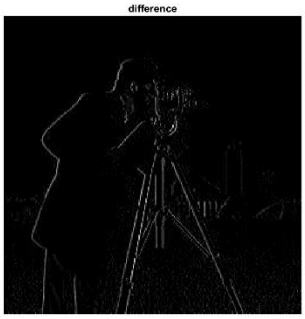
$f(x,y) = e(x,y) + \alpha f(x,y-1)$

We can't un-round a value, so this is the best we can do. But actually the f(x,y) is of type uint8, and alpha=1, which means in our case the Round function has no effect. On the other hand, when we created e(x,y) we subtracted two values and if the value was negative it was kept as zero (type uint8), so we cannot retrieve the lost info unfortunately.

To sum up -we expect some difference between the reconstructed image and the pre-reconstructed image.

Results:





Code:

```
%6.c
[R,C]=size(e);
rec_pred=uint8(zeros([R,C]));
for r=1:R
    rec_pred(r,1)=e(r,1);
    for c=2:C
        rec_pred(r,c)=e(r,c)+f_prime(r,c);
    end
end
figure(63);
subplot(1,2,1);
imshow(rec_pred);
title('reconstructed image');
diff=abs(rec_pred-img_rec);
subplot(1,2,2);
imshow(diff);
title('difference');
```