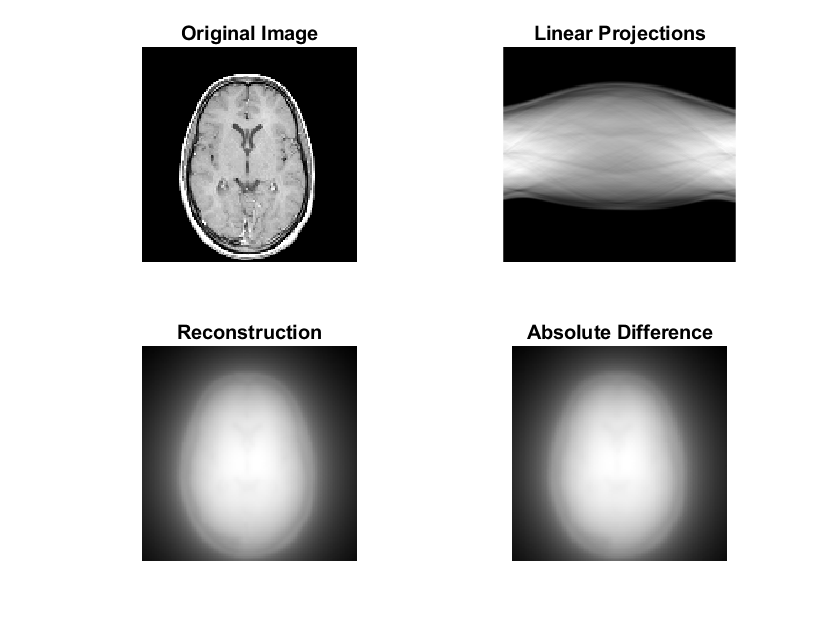
ELEC421 – HW5 - Or Bahari 51277200

1.a.

Running the code with default values results in:

We can see that the SNR is bad, the difference image has many important properties. The reconstruction is not really similar to the original image..

1.b.

You asked for it. So 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:481.629646370018

Min MSE:

In red. SPLINE and Ram-Lak.

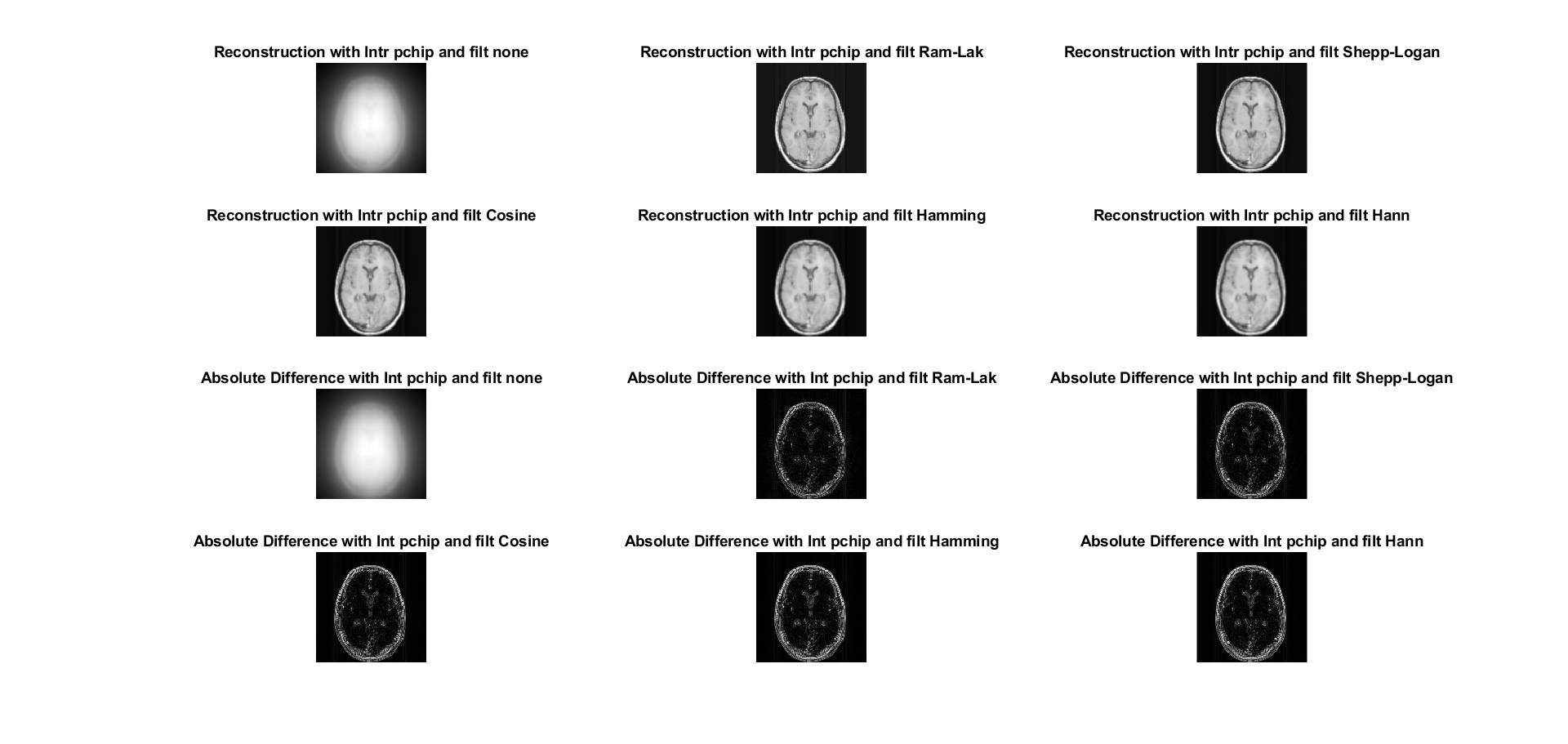
Value: 0.000311116751578031

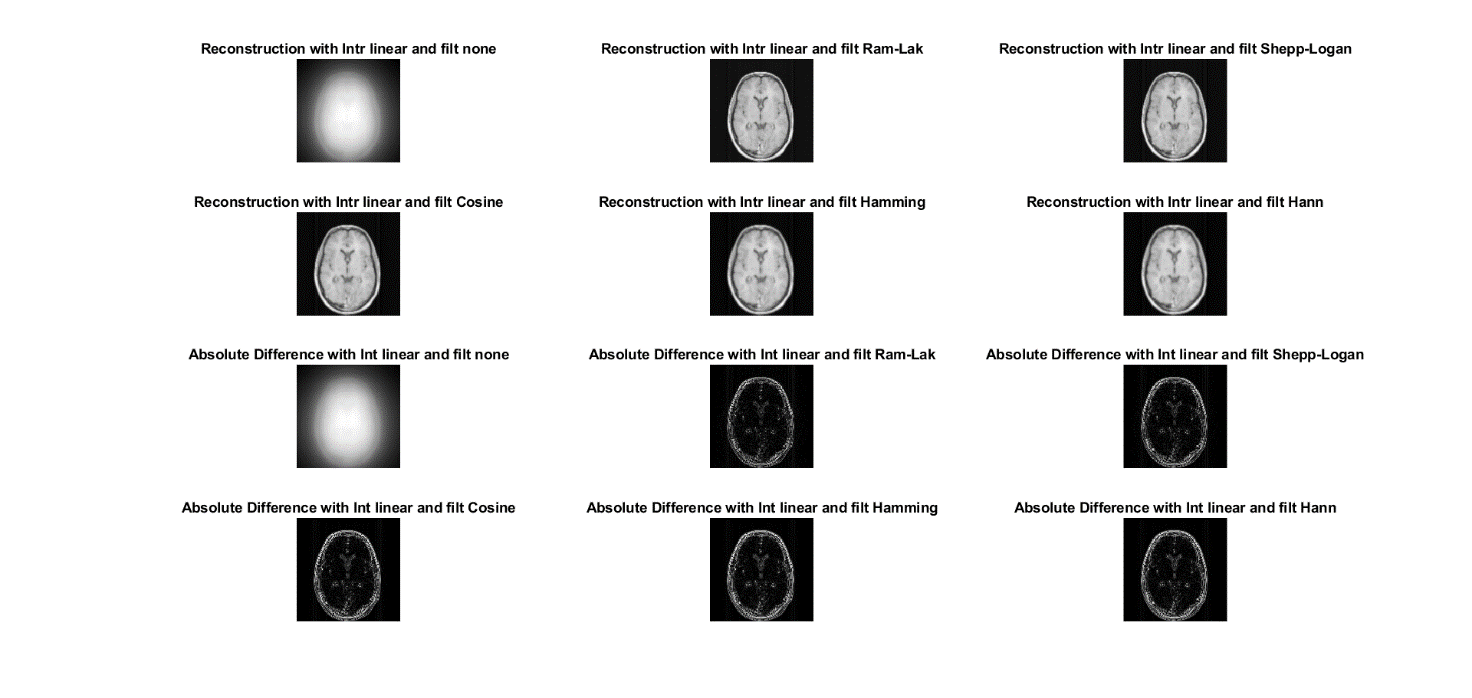
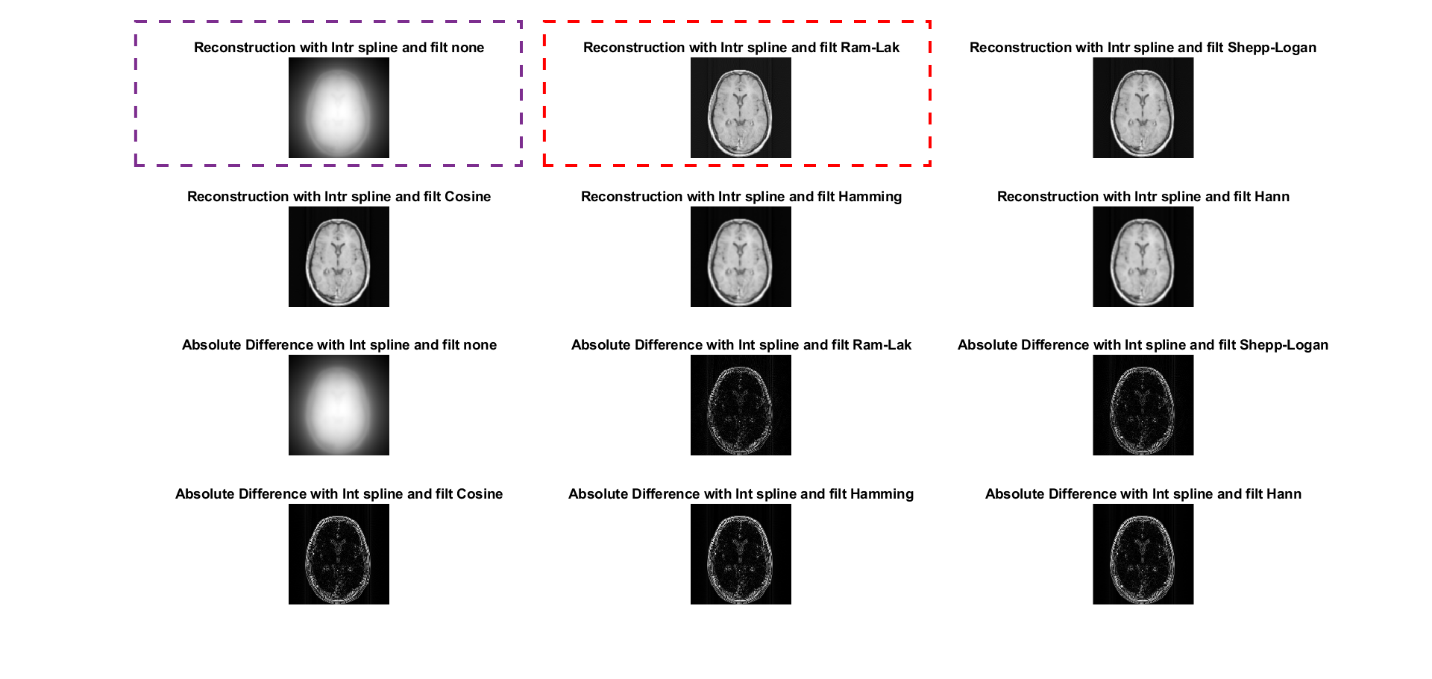
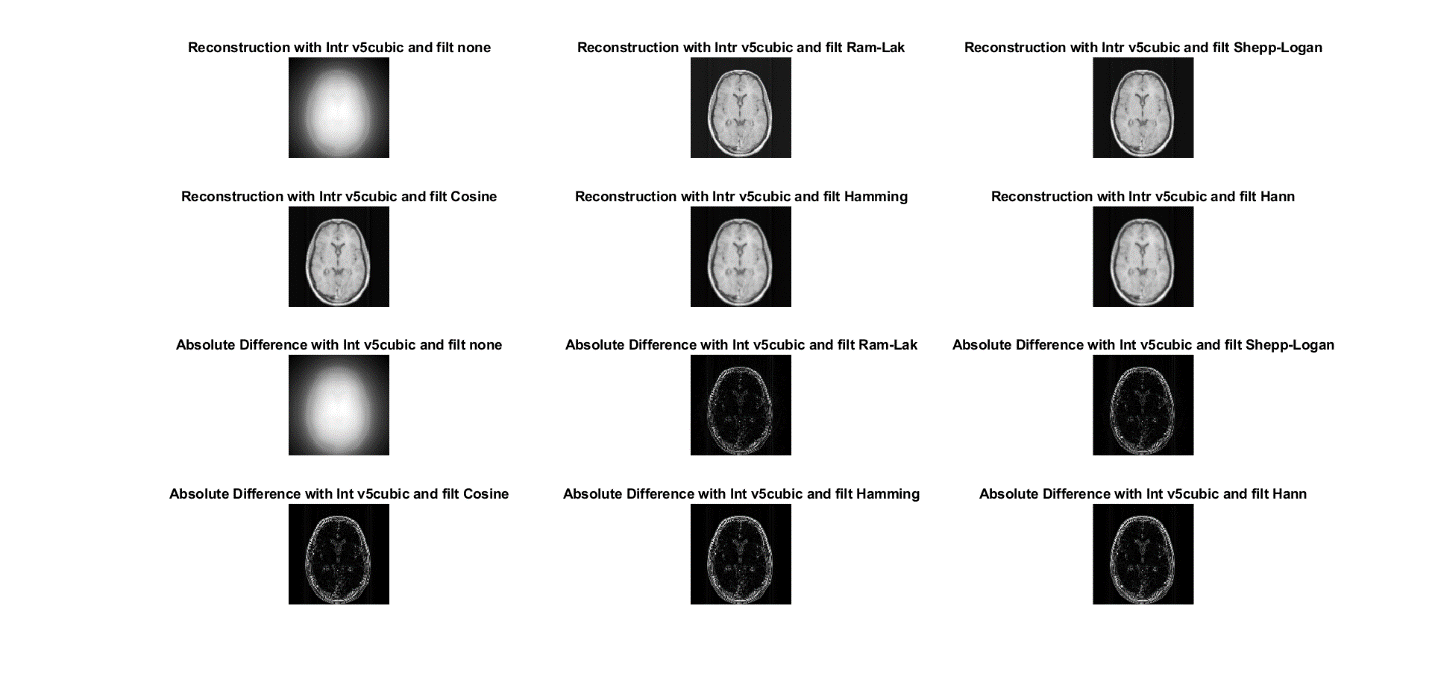
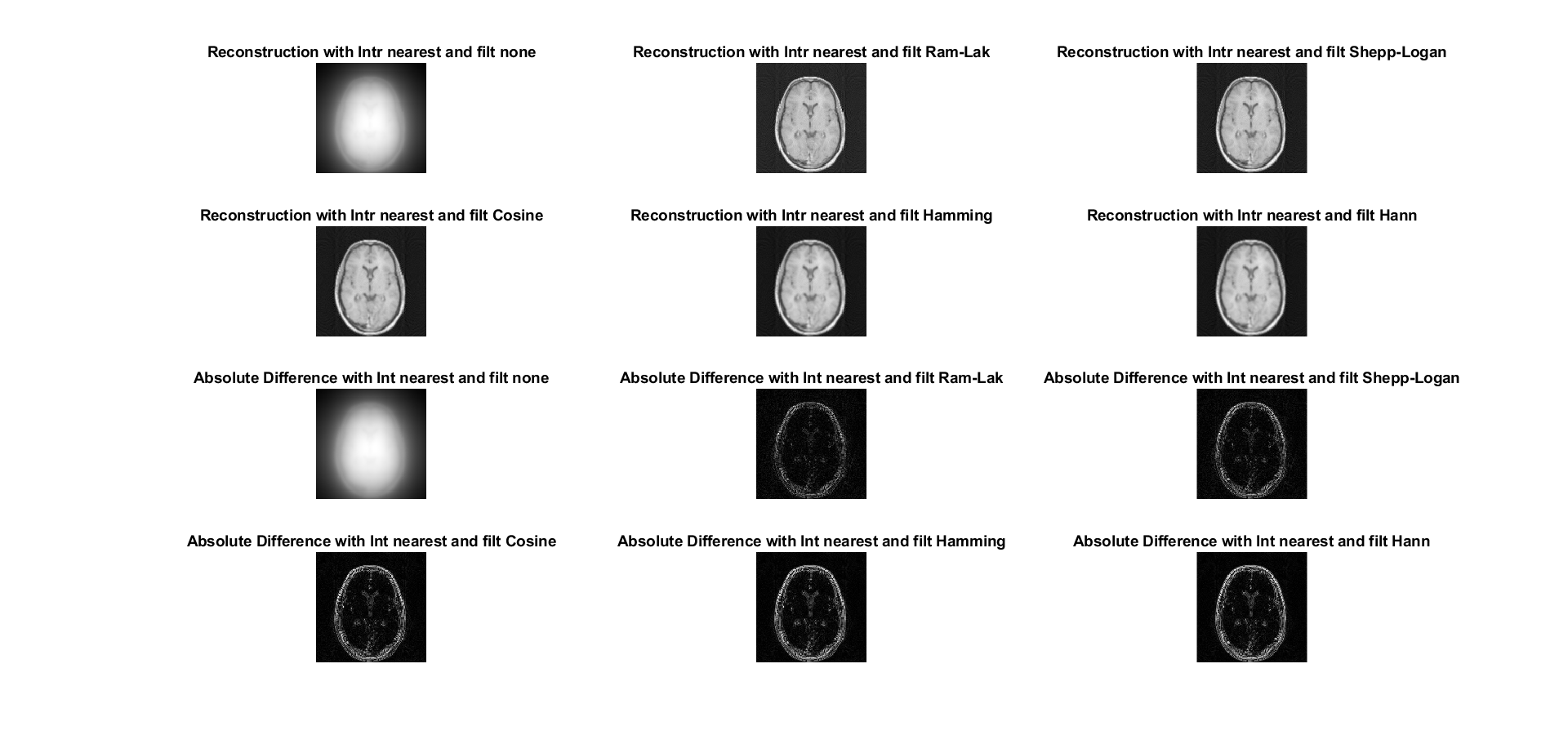
And not surprisingly the max SNR is the min MSE and vice versa..

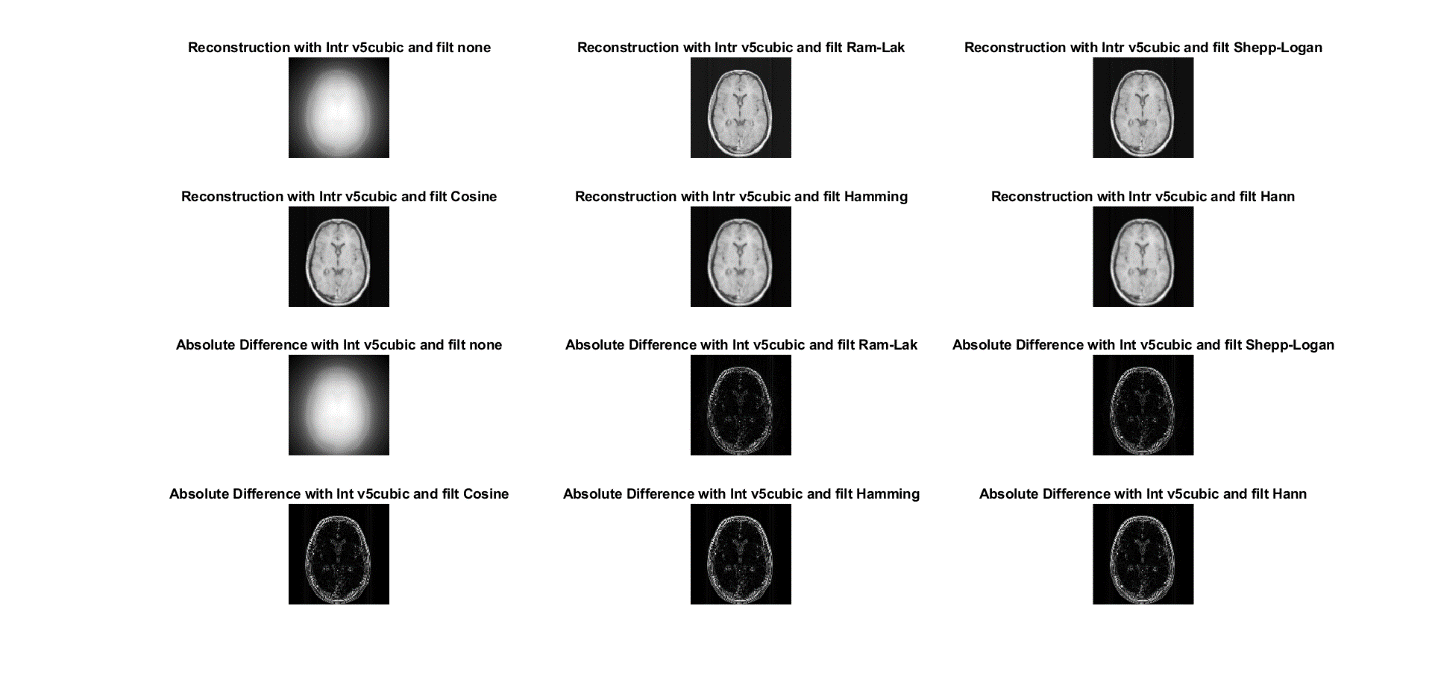
Min SNR: -98.9760495495154

Max SNR: 43.5491280445504

My observation – many pictures have pretty much the same clarity and look like the original image.



­­­­­



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;

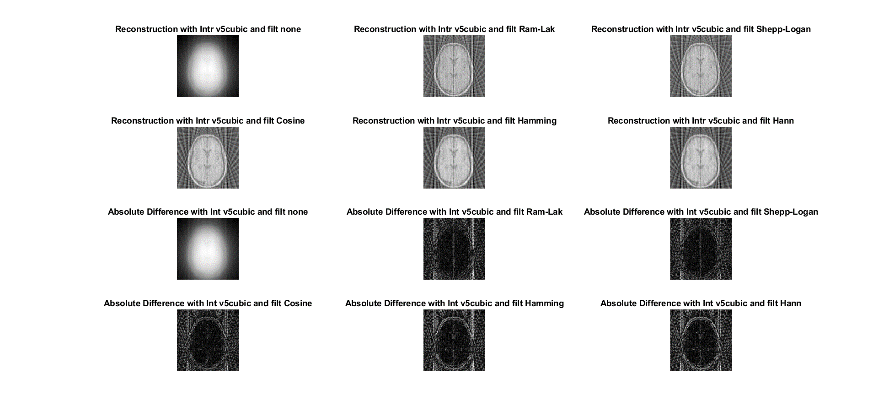
end

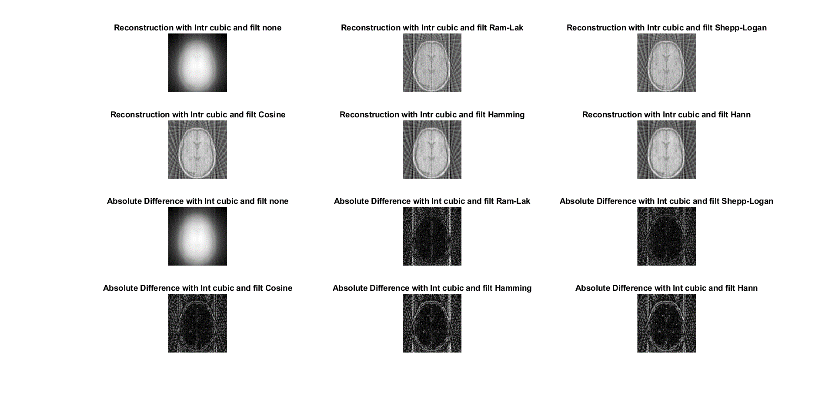
outer=outer+1;

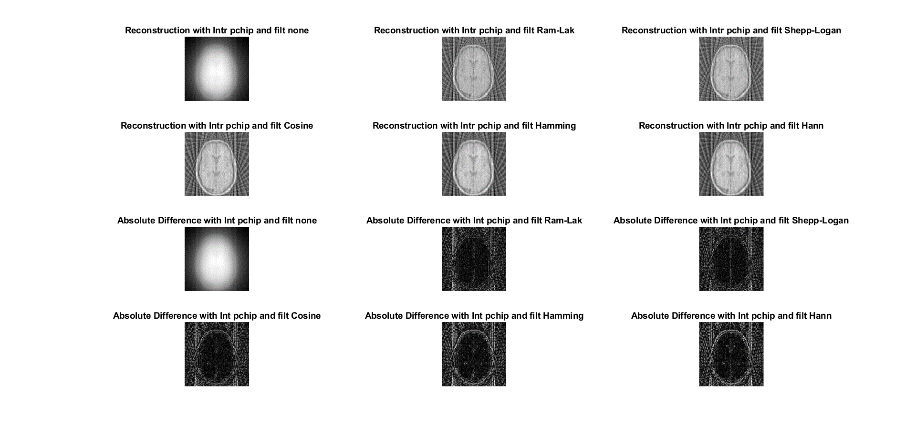
end

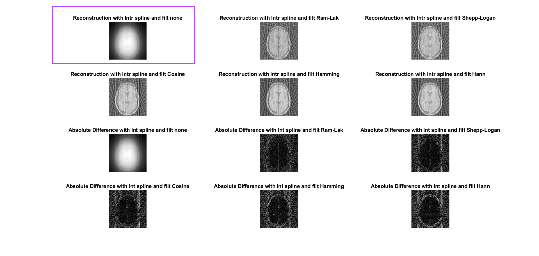
1.c.

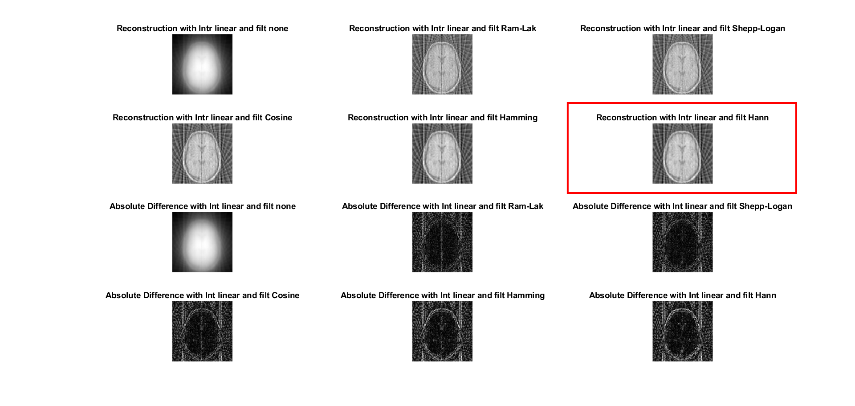
The results are:

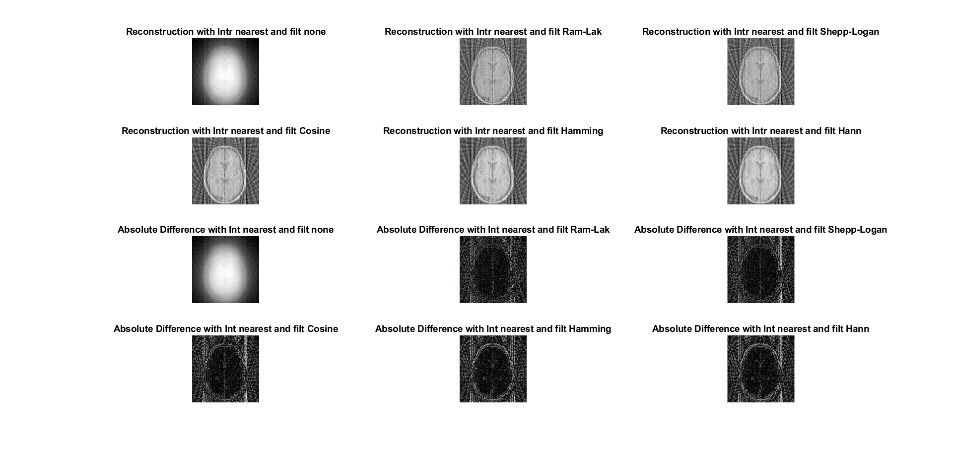












Code used to decided the max and min MSE (I was smart and made the MSE and SNR arrays such that each index matches a different set of parameters).

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

a =

12

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

b =

13

Max MSE:

purple image.

value 483.890907429503

Min MSE:

red image.

Value 0.00280898179477946

3.a.

5.a

Information Entropy – stands for the minimal amount 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:

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

Code:

ent=entropy(img\_rec);

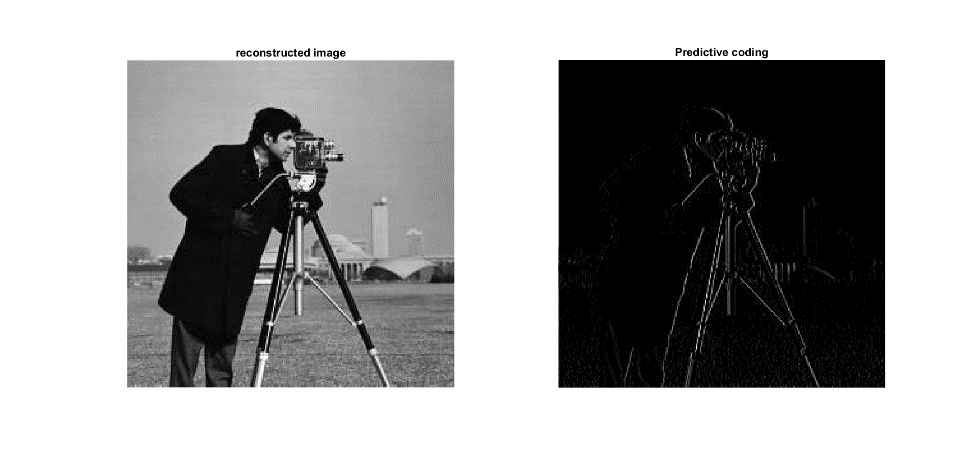
5.c.

Let’s check how many info bits we get using the Huffman compression code.

Therefore the Huffman code achieved <2>% of the maximum compression possible.

???

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);

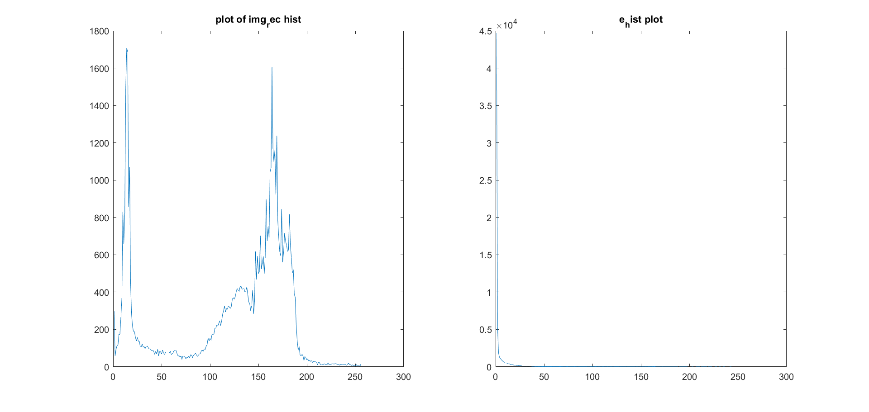
title('reconstructed image');

subplot(1,2,2);

imshow(e);

title('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:

e\_ent =

0.7281

rec\_ent =

0

Comments-????

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\_hist);

rec\_ent=entropy(rec\_hist);

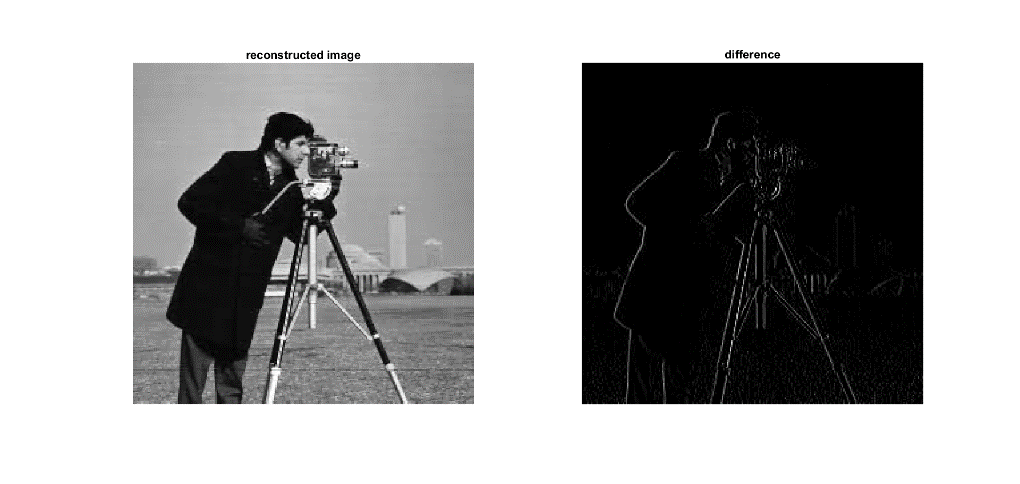
6.c.

Math:

We can’t un-round a value, so this is the best we can do. But actually the f(x,y) is of time uint8, and alpha=1, which means in our case the Round function doesn’t do anything. On the other hand, when we created e(x,y) we subtracted two values and if the value was negative it kept as zero (each picture is of 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');