hw06_ChaehyeonKim_2020250028

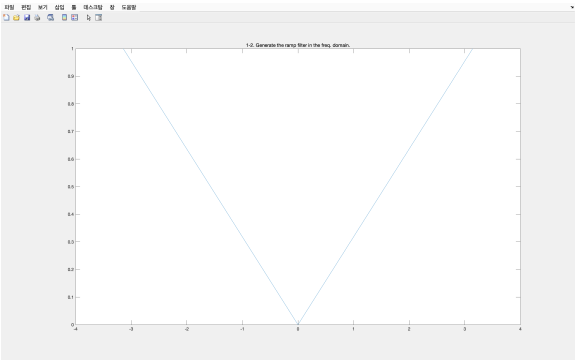
Prob1

1. Read the input file, sinogram.bmp

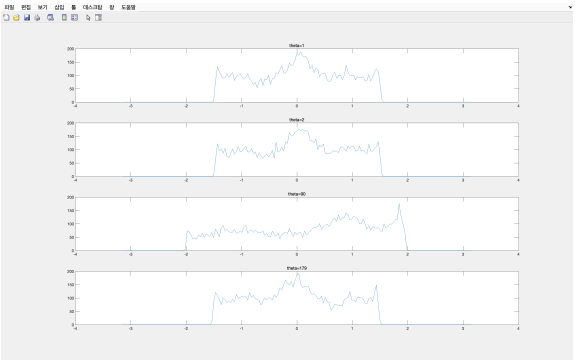
```
% 1. Read the input file
sinogram = imread("sinogram.bmp");
```

2. Frequency Filtering : Generate the ramp filter ($|\omega|$) in the frequency domain. For $0 \leq \theta \leq \pi$, apply it to the 1D projection $g(l,\theta)|_{\theta=0,\ldots,\pi}$ in the given 2D sinogram $g(l,\theta)$.

```
% 2. Frequency Filtering

% Generate the ramp filter in the frequency domain.
x = -pi:2*pi/184:pi;
ramp = abs(x)/pi;
figure; plot(x,ramp); title('1-2. Generate the ramp filter in the freq. domain.');

% Frequency Filtering.
GG = zeros(size(sinogram));
for theta = 1:size(sinogram,1)
    sfft = fftshift(fft(sinogram(theta,:)));
    GGF = real(ifft(ifftshift(ramp.*sfft)));
    GG(theta, :) = GGF;
end

figure;
subplot(4,1,1); plot(x,sinogram(1,:)); title("theta=1");
subplot(4,1,2); plot(x,sinogram(2,:)); title("theta=2");
subplot(4,1,3); plot(x,sinogram(90,:)); title("theta=90");
subplot(4,1,4); plot(x,sinogram(179,:)); title("theta=179");

figure;
subplot(1,2,1); imshow(sinogram,[]); title('Original Sinogram');
subplot(1,2,2); imshow(GG,[]); title('Frequency Filtering');
```



ramp filter



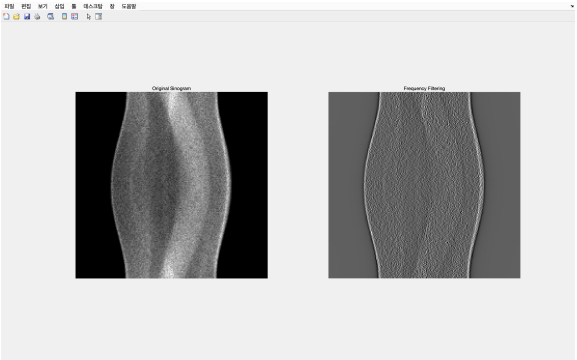
sinogram for each theta

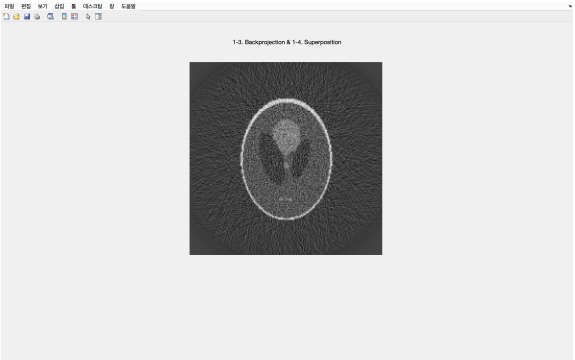
- frequency domain에서 그림과 같은 ramp filter를 생성하였음. 이미지의 크기인 185에 맞게 x vector를 생성하고, 그에 abs() 함수를 이용하는 방식으로 생성함.
- frequency filtering을 위해 fft와 fftshift, ifft 함수를 이용하였으며, 각 sinogram에서 행 별로 적용하였음.
- $\theta = 1, 2, 90, 179$에 대해 sinogram을 plot한 결과 그림과 같이 나타났음

3. Backprojection & Superposition

```
% 3. Backprojection & 4. Superposition
re = zeros(185,185);
for i = 1:180
    mm = repmat(GG(i,:),185,1);
    mmrot = imrotate(mm,i-1,"crop");
    re(1:size(mmrot,1),1:size(mmrot,2)) = re(1:size(mmrot,1),1:size(mmrot,2)) + mmrot;
end
figure; imshow(re,[]); sgtitle('1-3. Backprojection & 1-4. Superposition');
```

- repmat을 통해 결과를 반복해서 matrix를 만들어주고, 각각에 대해 imrotate로 회전시킴.
- crop을 통해 원본 sinogram의 최대 길이인 185로 맞춤.
- “re”라는 행렬에 각각을 합쳐줌으로써 superposition을 수행함.





Prob2

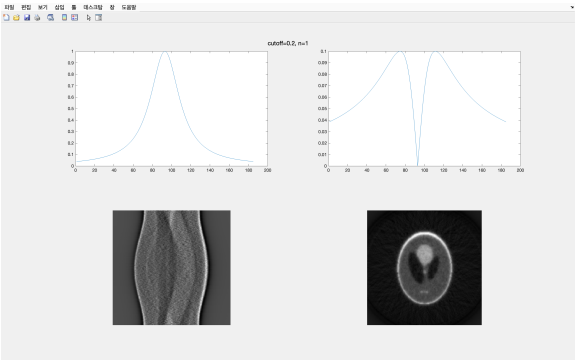
- 여러 값을 실험하기 위해 함수를 작성하여 사용함

```
function filter = butterworthLPF(f, cutoff, n)
    cof = pi*cutoff;
    filter = 1./(1+((f/cof).^(2*n)));
end
```

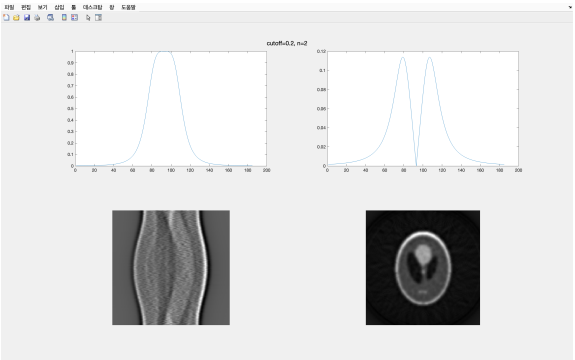
```
function filtered = freqfilt(sinogram,filter)
    filtered = zeros(size(sinogram));
    for i=1:180
        fft_result = fftshift(fft(sinogram(i,:)));
        filtered(i,:) = real(ifft(ifftshift(filter.*fft_result)));
    end
end
```

- 메인 코드에서 cutoff와 n 값을 변경해가며 실험 진행하였음

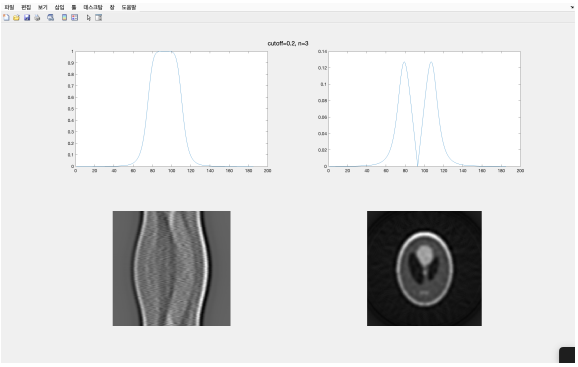
```
%% Prob 2. Ramp filter with BPF
% butterworthLPF, freqfilt 함수 작성하였음

% 변경할 값
cutoff = 0.2;
n = 10;

butt = butterworthLPF(x,cutoff,n);
bff = ramp.*butt;
fff = freqfilt(sinogram,bff);

re = zeros(185,185);
for i = 1:180
    mm = repmat(fff(i,:),185,1);
    mmrot = imrotate(mm,i-1,"crop");
    re(1:size(mmrot,1),1:size(mmrot,2)) = re(1:size(mmrot,1),1:size(mmrot,2)) + mmrot;
end
figure;
subplot(2,2,1); plot(butt);
subplot(2,2,2); plot(bff);
subplot(2,2,3); imshow(fff,[]);
subplot(2,2,4); imshow(re,[]);

figtitle = "cutoff=" + string(cutoff) + ", n=" + string(n);
sgtitle(figtitle);
```



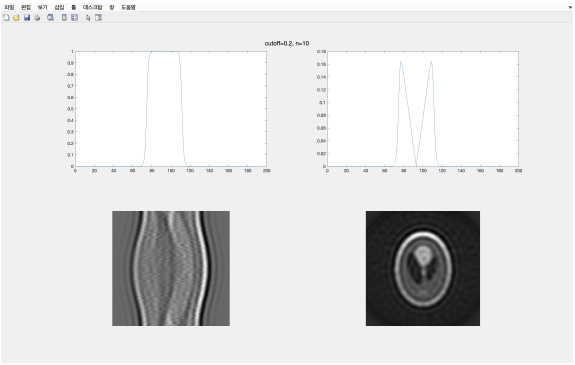
cutoff=0.2, n=1



cutoff=0.2, n=2

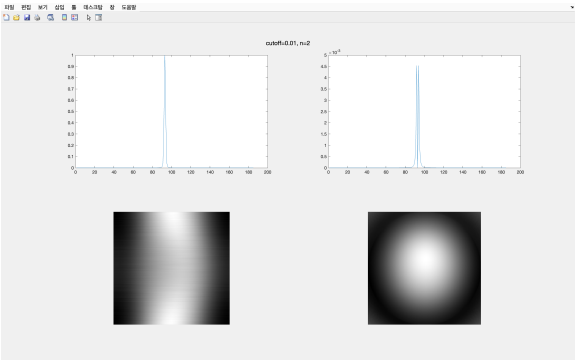


cutoff=0.2, n=3

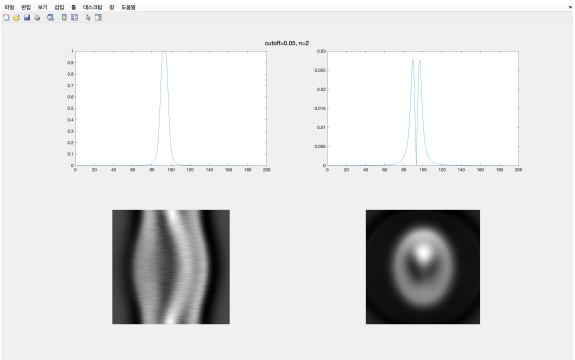


cutoff=0.2, n=10

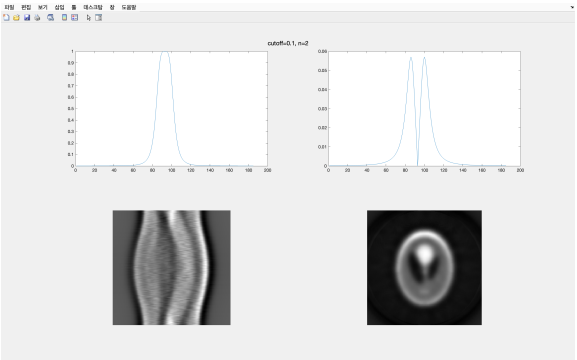
n을 1, 2, 3, 10으로 변화시켜본 경우 (cutoff=0.2로 고정), n=2일때가 reconstruction 이미지와 sinogram 모두 가장 괜찮게 나타남을 확인할 수 있다. 따라서 이후 n=2로 고정 후, cutoff frequency를 달리하였다.



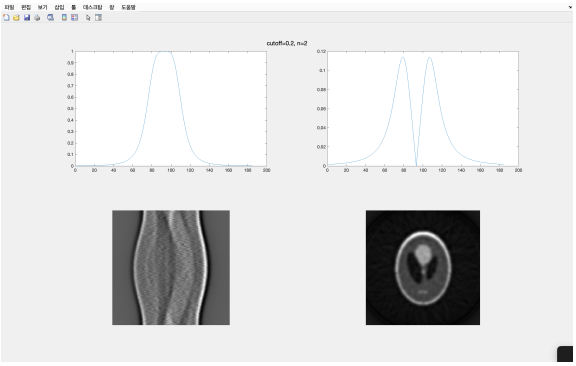
cutoff=0.01



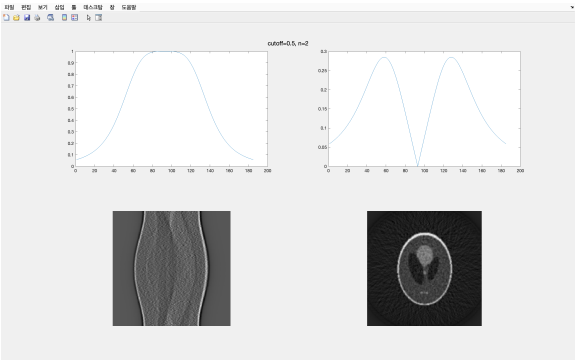
cutoff=0.05



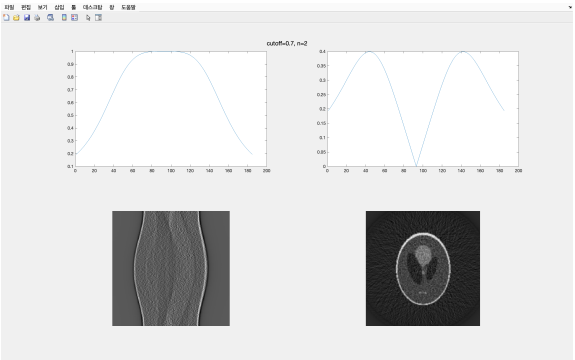
cutoff=0.1



cutoff=0.2



cutoff=0.5



cutoff=0.7

cutoff frequency를 0.01~0.7까지 다르게 설정하여 실험을 진행해 보았다. 전반적으로 보았을 때, cutoff frequency가 0.2는 되어야 영상이 선명하게 복원되기 시작하는 것을 확인할 수 있다. cutoff=0.7인 경우 선명하긴 하지만 배경에 노이즈가 강조되는 것을 확인할 수 있어, cutoff frequency가 0.5일때의 결과가 가장 좋아 보인다.

ramp filter와 비교했을 때, ramp filter에서는 reconstruction 된 이미지의 배경이 매우 노이즈가 심한 것을 확인할 수 있다. band pass filter를 사용한 경우, low frequency는 통과시키지 않고, high frequency에 대해서는 증폭시켜 준 뒤, 매우 high한 frequency, 즉 noise인 정보에 대해서는 또 다시 통과시키지 않는 것으로 이해할 수 있기 때문에, Band pass filter를 사용한 결과가 ramp filter보다는 좋은 결과를 얻을 수 있다고 볼 수 있다.