

## 核医学機器工学概論 2

### 断層画像CT(Computed Tomography)を得る方法

#### 1. フィルタ重量逆投影法

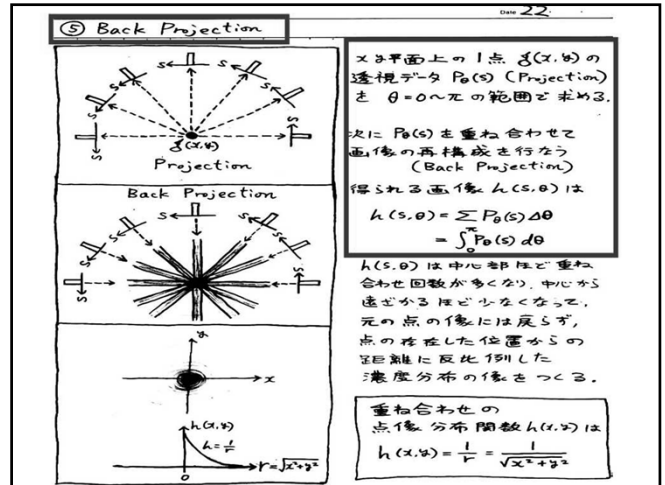
FBP (Filtered Back Projection)

#### 2. 逐次近似再構成法 Iterative Reconstruction

MLEM (Maximum Likelihood Expectation Maximization)

OSEM (Ordered Subsets Expectation Maximization)

1



2

プログラム PSF.exe の実行。フォルダ PSF 内の PSF.exe をダブルクリック。

```
Point Spread Function
PSF BackProjection
max count = 10.000000
min count = 0.000000
Select Reconstruction method
1: Simple BackProjection
2: Filtered BackProjection
Load Real space Filter
Real space filter =
C:\Users\VKatoh\Desktop\医用画像機器工学実
OK? (yes; enter, no: n)
Disp Filtered Pth * Filter
max count = 76308.327148
min count = -8226.814328
Disp FBP Process?
(yes; enter, no: n)
HU Center = 1
```

このテキストウィンドウ内をクリックしてから 1を入力して Simple back projectionを実行。

次にプログラム PSF.exe を終了、再度実行。2を入力して Filtered backprojectionを実行。

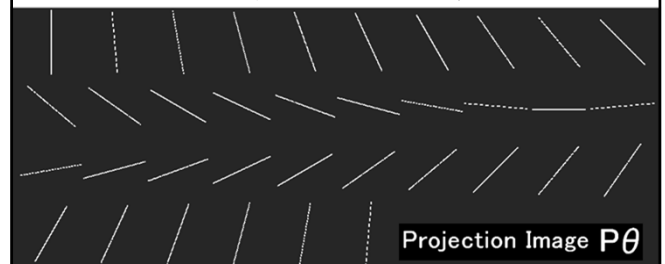
選択する再構成フィルタは、(real space filter は) フォルダ PSF 内にある RealRAMP256.txt を選択。

3

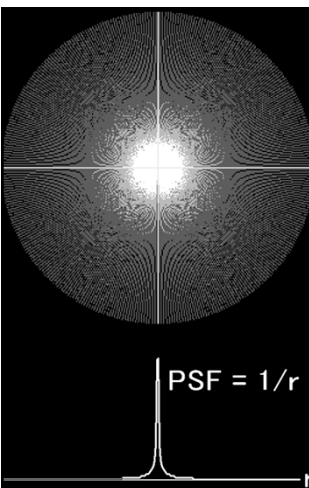
画像中心の1画素だけに画素値を与え、他を0に設定した2次元画像を作成。

その像を180度方向から横から透視したと想定した像  $P_\theta$  を作成(1度ごと 合計180枚の線状画像)。

(スライドでは5度ごとの画像を表示)



4



180枚の線状画像  $P_\theta$  を重ね合わせると、広がりをもつ分布が得られる。

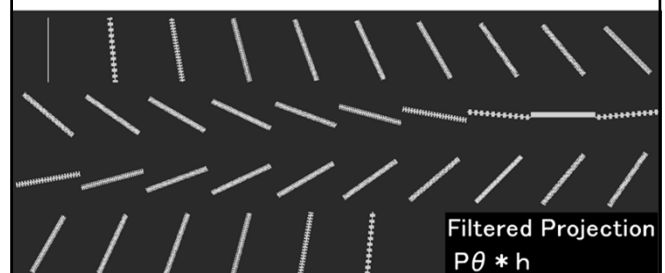
点広がり関数 PSF (Point Spread Function)

$$PSF = 1/r$$

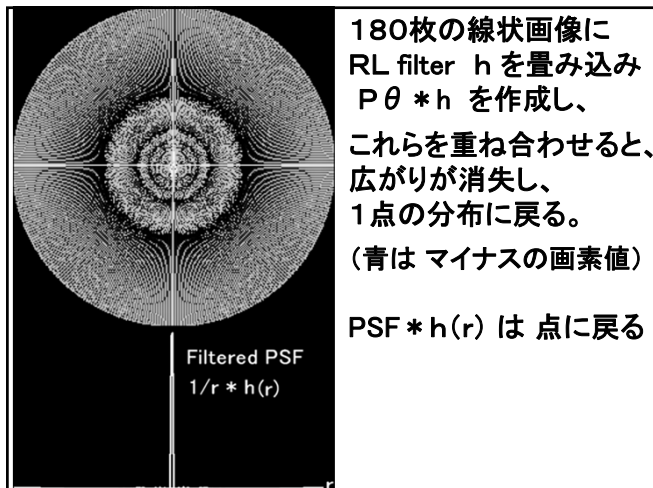
5

180枚の線状画像  $P_\theta$  に、実空間 Ramp (RL) filter  $h$  を畳み込む ( $P_\theta * h$ )。

(青く表示された画素はマイナスの値)



6



7

回転中心からの距離  $r$  に反比例した濃度に補正するフィルタ  
 $1/r$  を正確な断層像  $g$  に畳み込んだ像が  $l$  である。  
式で表現すると  $l = g * (1/r)$  となる。  
 $l$ 、 $g$ 、 $1/r$  のフーリエ変換を  $L$ 、 $G$ 、 $F(1/r)$  と  
表現すると、畳み込みの定理より  
 $L = G \cdot F(1/r)$  となる。  
2次元フーリエ変換の公式の極座標表現を用いると、  
( $r$  は周波数空間上の原点からの距離 )  
 $F(1/r) =$   
 $\iint (1/r) \exp(-j(2\pi r f r)) r dr d\theta = 1/f r$   
これより  $L = G / f r$  なので  $G = L \cdot f r$

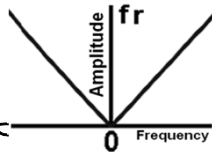
8

$G = L \cdot f r$  の意味は、2次元周波数空間上で、  
単純重ね合わせ画像をフーリエ変換した2次元データ  $L$  に、  
フィルタ関数  $f r$  ( $f r$  は周波数空間上の原点からの距離)  
をかけると、正しい再構成画像をフーリエ変換したデータ  $G$   
になる。

$G = L \cdot f r$  に、畳み込みの定理を  
用いると、以下のような実空間での計算に  
変換できる。

この式を 逆フーリエ変換すると、

$g = l * h \quad (h \text{ は フィルタ } f r \text{ の 逆フーリエ変換})$



9

この式に、 $l = \int P\theta d\theta$  を代入すると、  
 $g = \int P\theta d\theta * h$   
 $g = \int (P\theta * h) d\theta$  ( $h$  は  $\theta$  と独立した値なので交換可)

$g = \int \overline{P\theta} d\theta \quad (\overline{P\theta} = P\theta * h)$

**FBPの式**

$P\theta$  に 実空間フィルタ  $h$  ( $= fr$  の逆フーリエ変換) を畳み込めば、  
重ね合わせると正確な断層像  $g$  になる2次元透視画像  $\overline{P\theta}$  を  
算出できる。これを Filtered Back Projection (FBP) という。

周波数空間での実際の計算においては、フィルタ  $H$  ( $= fr$ ) は  
常に正の値であり(絶対値)、  
さらにサンプリング定理より、ナイキスト周波数以上の成分を  
削除する必要がある。

10

周波数空間での再構成フィルタ  $H$  は、  
 $H = |f r|$  ( $f r$  がナイキスト周波数未満の場合)  
 $H = 0$  ( $f r$  がナイキスト周波数以上の場合)  
となる。これを Ramp フィルタという。

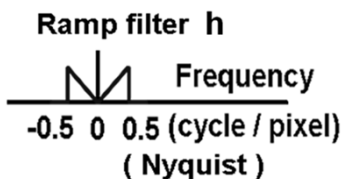
Ramp フィルタを逆フーリエ変換して  
実空間 Ramp フィルタ  $h$  にしてから、  
実空間で  $P\theta$  に  $h$  を畳み込む。

$\overline{P\theta} = P\theta * h$   
(\* は畳み込み演算)

Ramp filter  $h$

Frequency

-0.5 0 0.5 (cycle / pixel)  
(Nyquist)



11

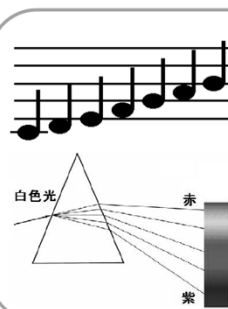
$1/r$  のフーリエ変換が  $1/fr$  になる理由

2次元フーリエ変換の公式の極座標表現を用いると  
( $r$  と  $\theta$  は実空間の原点からの距離と偏角、 $fr$  は周波数空間の原点からの距離)  
関数  $g(r)$  のフーリエ変換は  $\iint g(r) \exp(-j \cdot 2\pi r f r) r dr d\theta$  (ヤコビアンに注意)  
 $g(r) = 1/r$  とすると  $\iint (1/r) \exp(-j \cdot 2\pi r f r) r dr d\theta = \iint \exp(-j \cdot 2\pi r f r) dr d\theta = 1/fr$

周波数空間を理解することは難しい印象を持つが、周波数空間や周波数分布を表現しているものは一般にも結構ある。

たとえば楽譜は、音楽の周波数分布、曲のフーリエ変換とも解釈できる。五線譜の下側の音符は低音、低周波数成分を表し、上側の音符は高音、高周波成分を示している。


虹は白色光に含まれる色の周波数分布を示している。雨粒中で屈折しやすい高周波の紫色成分から、屈折しにくい低周波の赤色成分まで、周波数の順序で色が並んでいる。



12

基礎 Rampフィルタ

周波数空間のRampフィルタ  $H$



$-0.5 \ 0 \ 0.5 \text{ (cycle / pixel)}$   
(Nyquist)

周波数空間でのフィルタ  $fr$  とは、どのような形をしているのか。1次元周波数空間の縦軸の周波数成分が  $fr$  の値を持つフィルタであるが、横軸が周波数軸すなわち  $fr$  である。つまり周波数空間で、正比例のような形状になる。

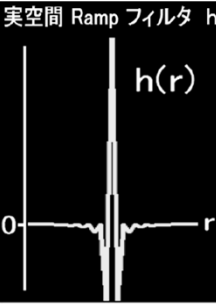
周波数にはマイナスの値は無いが、高速フーリエ変換 (FFT) のアルゴリズムの都合で見かけ上、マイナスの周波数データが存在する。縦軸の周波数成分にはマイナスは無いので、周波数がマイナス側の成分は絶対値となり、原点を中心に左右対称な絶対値  $|fr|$  の関数として表現される。

さらにサンプリング定理によって、ナイキスト周波数 (0.5 cycle / pixel) 以上の高周波成分はノイズなので除去する。

よって、このフィルタは上図のようになり、これをRampフィルタという。


13

実空間 Ramp フィルタ  $h$

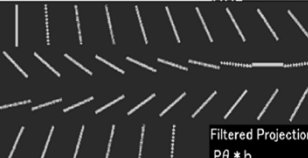


周波数空間の Ramp フィルタを  $H$  と表現する。 $H$  を 1 次元逆フーリエ変換すると、実空間での Ramp フィルタ  $h$  が左図のように算出される。

画像  $g$

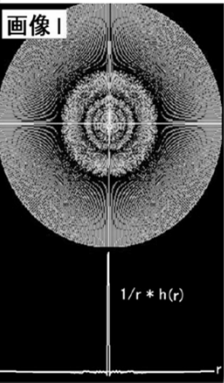


Filtered Projection  $P\theta * h$



14

画像  $I$

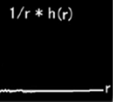


画像中心の 1 画素だけ値が 1 で、他は全て 0 の 256x256 画素の画像  $g$  を 1 度毎に 180 度方向から透過投影した投影像  $P\theta$  を作成する。次に全ての  $P\theta$  に Ramp フィルタ  $h$  を畳込む (式で表すと  $P\theta * h$ ) (上図で青色の部位はマイナスの画素値) (全ての  $P\theta * h$  の断面が  $h$  と同じ形状をしている)。さらに、全ての  $P\theta * h$  を重ね合わせて画像  $I$  を算出する。式で表すと  $I = \int (P\theta * h) d\theta$

画像  $I$  は中心の 1 画素だけ値が 1 で、周囲は殆ど 0 に近い値を示し、ほぼ画像  $g$  に戻っている。

$P\theta$  の単純重ね合わせ像では、画像中心の 1 画素が最大値を示し、その近傍に中心からの距離  $r$  に反比例する  $1/r$  の関数に画素値が広がっていたが、Ramp フィルタ  $h$  を畳込むと、その広がりが補正されて元の画像が再現される。この Ramp フィルタ  $h$  の機能によって断層画像を算出することができる。

$1/r * h(r)$

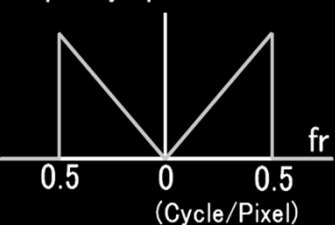


15


周波数空間 RL フィルタ ( $1/fr$ ) を 1 次元逆フーリエ変換して、実空間 RL フィルタ  $h(r)$  を作成。

フォルダ RAMP256 内の RAMP256.exe を実行。

Frequency Space RL filter

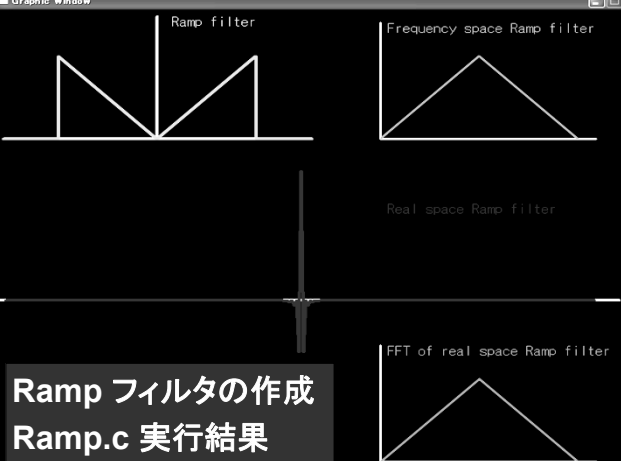


Real Space RL filter



16

Graphic Window



Ramp フィルタの作成  
Ramp.c 実行結果

17

プログラム CTFBP.exe の実行。フォルダ CT 内の CTFBP.exe をダブルクリック。

CT FBP

```
CT BackProjection
Load CT Sinogram data
max count = 707071.474638
min count = 0.000000

Select Reconstruction method
1: Simple BackProjection
2: Filtered BackProjection
1

Disp Filtered Pth * Filter

max count = 97780557.639242
min count = 683.125000

Disp FBP Process?
(yes:enter, no:n)
```

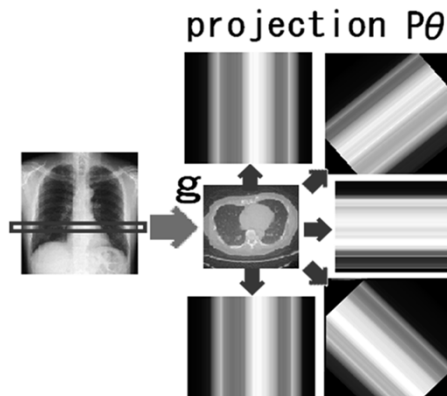
このテキストウィンドウ内をクリックする。  
選択するプロジェクションデータは、フォルダ CT 内の CTprojection を選択。

1を入力して  
Simple back projectionを実行。

Disp FBP Process? と出たら Enter キーを押す。  
Simple Back projection が実行される様子を観察。

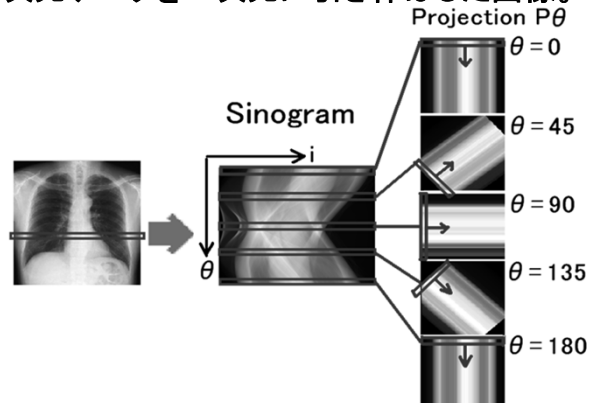
18

求めたい正確な断層像  $g$  を算出するために、  
多方向の角度  $\theta$  から 2次元透視像  $P_\theta$  を測定。



19

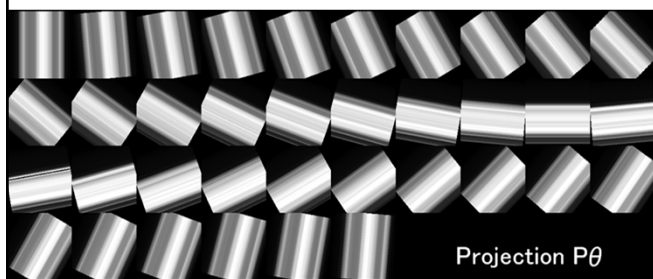
2次元透視像  $P_\theta$  は、サイノグラムの各行  $\theta$  の  
1次元データを 2次元に引き伸ばした画像。



20

例として胸部の1断面のサイノグラムから  
180度方向から横から透視したと想定した像  $P_\theta$  を  
作成(1度ごと、合計180枚の2次元透視画像)。

(スライドでは5度ごとの画像を表示)



21

180度方向から透視した像  $P_\theta$  を重ね合わせると  
ぼけて画像中心の値が持ち上がった像  $I$  を得る。  
正確な断層像  $g$  と  $I$  の関係式は、

$$I = g * h$$

正確な断層像  $g$  に 点広がり関数  $h$  が 畳み込まれ  
ぼやけた断層像  $I$  を得る と考える。



22

単純重ね合わせ再構成法 Simple Back Projection

収集された各々の角度に傾いた2次元透視画像 ( $P_\theta$ ) を  
全部単純に重ねると再構成画像ができる。  
(回転中心近傍の値が盛り上がった不正確な画像。)

スライス  $j$  におけるサイノグラムを求める。  
サイノグラムの各スライスの 1次元配列は、  
各々の角度から収集されたデータ。

サイノグラムの各スライスの 1次元配列から、  
収集された各々の角度に傾いた 2次元透視画像  $P_\theta$  を作成する。  
 $P_\theta$  を単純に重ね合わせた画像を  $I$  とすると

$$I = \int P_\theta d\theta \quad (\text{Simple back projection})$$

$I$  は、回転中心部ほど重ね合わせ回数が増え、  
中心から距離が遠いほどカウントの低い像になる。

23

プログラム CTFBP.exe の再実行。

フォルダ CT 内のCTFBP.exe をダブルクリック。



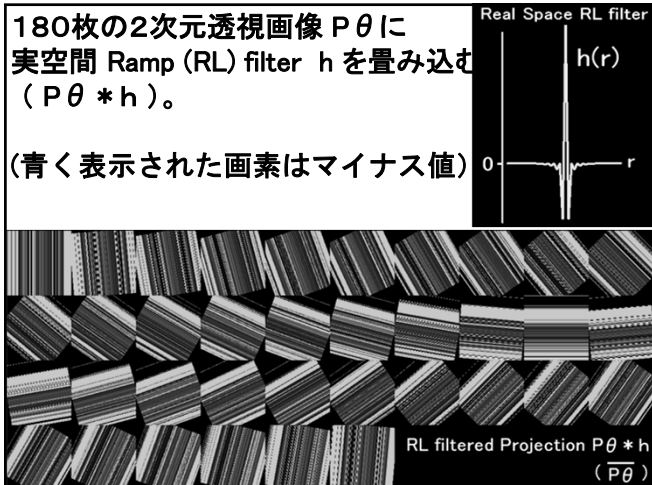
このテキストウィンドウ内を  
クリック。

選択するプロジェクション  
データは、フォルダ CT 内の  
CTprojection を選択。

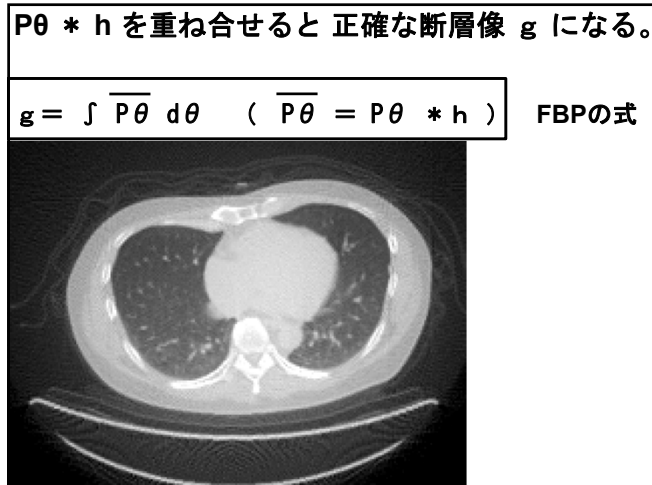
2を入力して  
Filtered back projectionを実行。  
選択するReal space filter は  
フォルダ CT 内の  
RealRAMP256.txtを選択。

Disp FBP Process? と出たら  
Enterキーを押す。

24



25

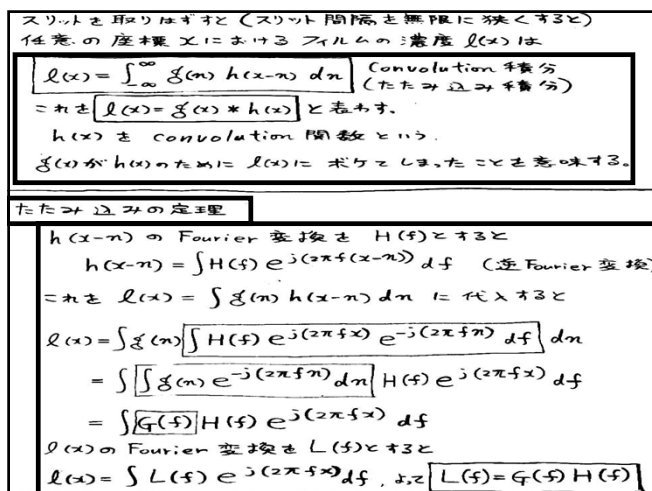


26

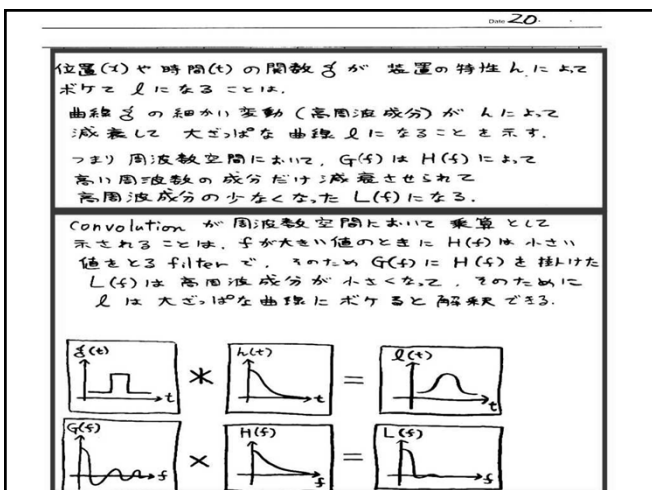
### 畳み込みの定理

データ  $g$  をフーリエ変換して、  
その周波数空間成分  $G$  に  
周波数空間 Rampフィルタ  $H$  をかけて  
逆フーリエ変換すると、  
実空間で、実空間 Rampフィルタ  $h$  を  
 $g$  に畳み込みしたデータと同じになる。  
( $G \times H$  と  $g * h$  は等価演算)

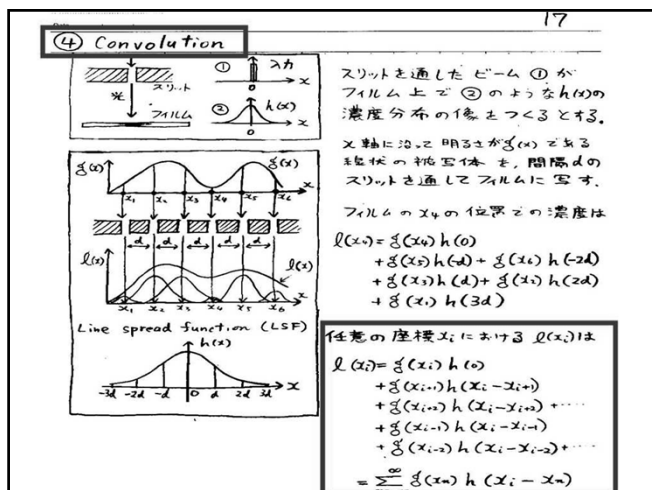
27



28

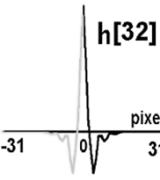


29



30

データ数64の1次元データ  $g[0] \sim g[63]$  と、右図に示すようなデータ数32の実空間フィルタ  $h[0] \sim h[31]$  がある ( $h[0]$  が原点)。  
データ  $g$  にフィルタ  $h$  を畳み込んで配列  $l[0] \sim l[63]$  に書き込むプログラムを記述して下さい。



```
//----- Convolution h[] into g[] -----
int      i, j;
double   g[64], h[32], l[64], gh;
for(i = 0; i <= 63; i++) {          // i < 64 と書いても同じ
    gh = 0.0;
    for(j = 0; j <= 31; j++) { if( i+j <= 63 ) gh += g[i+j] * h[j]; }
    for(j = 1; j <= 31; j++) { if( i-j >= 0 ) gh += g[i-j] * h[j]; }
    l[i] = gh;
}
```

31

$g$  に  $h$  を畳みこんだ結果を  $l$  とする。座標  $i$  における  $l$  の値  $l[i]$  は、  

$$l[i] = g[i] * h[0] + g[i+1] * h[1] + g[i+2] * h[2] + g[i+3] * h[3] + \dots$$


$$+ g[i-1] * h[1] + g[i-2] * h[2] + g[i-3] * h[3] + \dots$$
  
 ここでフィルタ  $h$  は偶関数(左右対称)なので、 $h[-j] = h[j]$  を代入し、  

$$l[i] = g[i] * h[0] + g[i+1] * h[1] + g[i+2] * h[2] + g[i+3] * h[3] + \dots$$

$$+ g[i-1] * h[1] + g[i-2] * h[2] + g[i-3] * h[3] + \dots$$
  
 これを C 言語で表す。  $h$  の要素数  $j$  は  $h[0]$  から  $h[31]$  まで。  
 $g$  の要素数は  $g[0]$  から  $g[63]$  までなので、 $g$  の添字を表す  $i+j$  は 63 以下、 $i-j$  は 0 以上の場合だけ加算する条件式を入れる。  
 $gh = 0.0;$   
 $for(j = 0; j <= 31; j++) { if( i+j <= 63 ) gh += g[i+j] * h[j]; }$   
 $for(j = 1; j <= 31; j++) { if( i-j >= 0 ) gh += g[i-j] * h[j]; }$   
 $l[i] = gh;$   
 これを  $i$  が 0 から 63 まで繰り返すと、 $l[0]$  から  $l[63]$  が計算される。

32

**プログラム PETFBP.exe の実行。フォルダ PETFBP 内の PETFBP.exe をダブルクリック。**



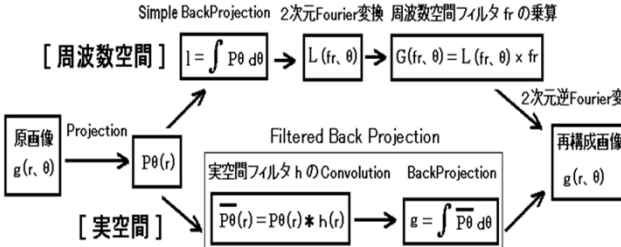
このテキストウィンドウ内をクリックする。  
 選択するプロジェクションデータは、フォルダ PETFBP 内の PETsinogram を選択。  
 Select slice おすすめスライス は、36、37、38あたり。  
 2を入力して  
 Filtered back projectionを実行。  
 選択フィルタは2種類。  
 RealRAMP256.txt  
 RealSheppLogan256.txt

33

投影像  $P\theta$  を単純重ね合わせた画像  $l$  を2次元フーリエ変換し、周波数空間でフィルタ  $fr$  を乗算したデータを2次元逆フーリエ変換すると、正確な断面画像  $g$  が得られる。

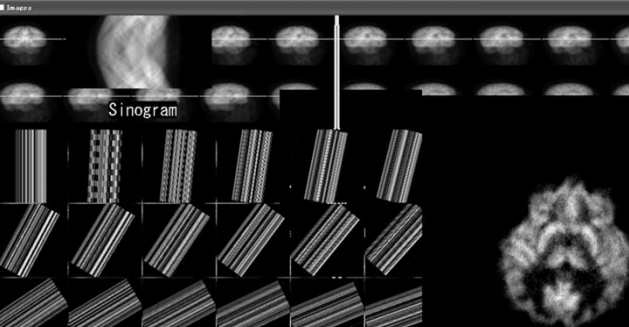
畳込みの定理を使うと、これらの周波数空間での処理が実空間で簡略化される。

投影像  $P\theta$  にフィルタ  $h$  ( $fr$  の1次元逆フーリエ変換) を畳込み、それを重ね合わせると正確な断面画像  $g$  が得られる。



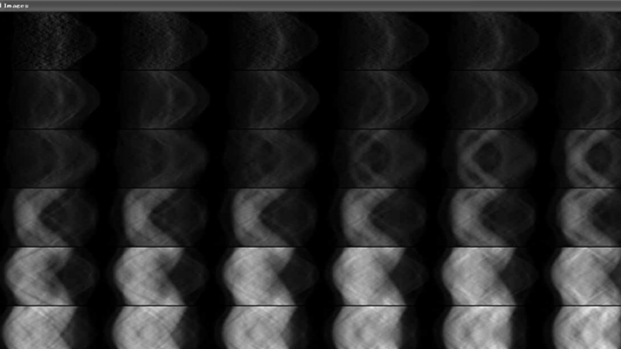
34

**プログラム PETFBP.exe**  
 PET画像を Simple BackProjection、または FBP (Filtered BackProjection) で再構成する。



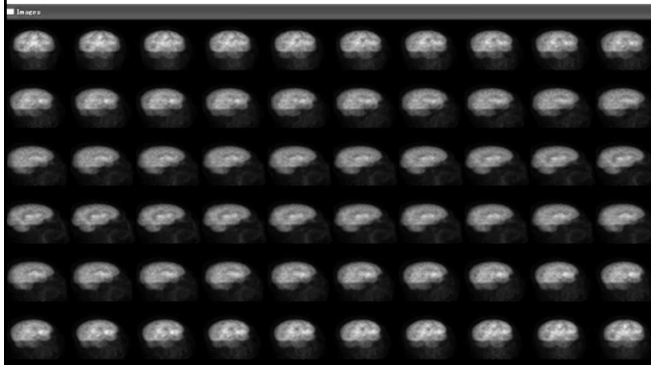
35

**PETsinogram ファイル**  
 PETの収集データは各スライスのサイノグラムが並んでいる3次元データ。(CTと同じ)。



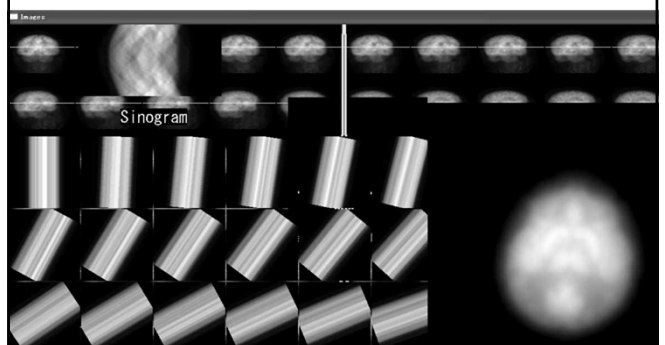
36

各スライスのサイノグラムが並んでいる3次元データを並べ替えれば、SPECTのプロジェクトン像と同じ並びになる。  $\text{Sinogram}[i][\theta][j] = \text{Projection}[i][j][\theta]$



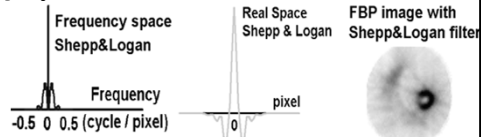
37

PETFBP.c を実行して、Simple BackProjection と FBP の 再構成画像の違いを観察し、Ramp等のフィルタ関数の必要性を理解してください。



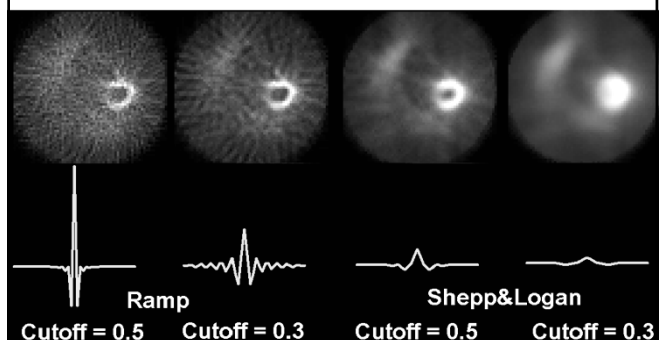
38

Rampフィルタは理論的には正確な再構成フィルタだが、実際の臨床データに適用するとナイキスト周波数以上の高周波成分を不連続に遮断するために生じる高周波ノイズや放射状アーチファクトが目立つ場合が多いため、臨床では高周波成分を抑制する工夫を施した再構成フィルタが用いられている。  
SPECTで用いられる一般的な再構成フィルタとして、Shepp&Loganフィルタがある。  
周波数空間上で、ナイキスト周波数近傍の高周波成分を連続的に減衰させるように設計されており再構成画像に生じる高周波ノイズや放射状アーチファクトを抑制する効果をもつ。

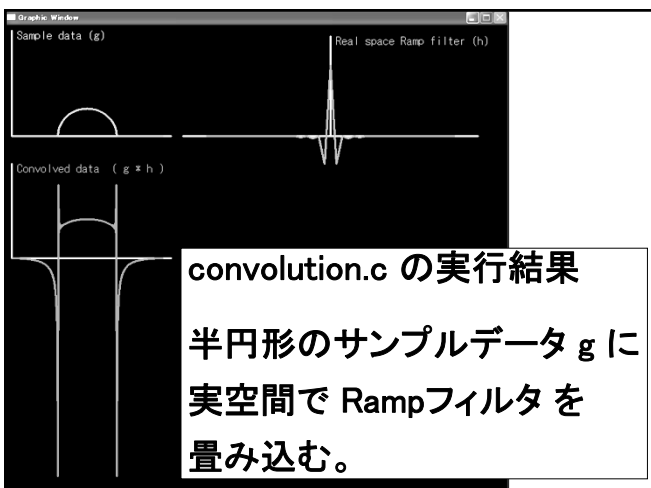


39

実空間フィルタを畳んだ2次元透視画像 ( $\overline{P\theta}$ ) を重ね合わせるとフィルタ逆投影再構成像ができる。フィルタの形状で、再構成画像の高周波成分が変わる。



40



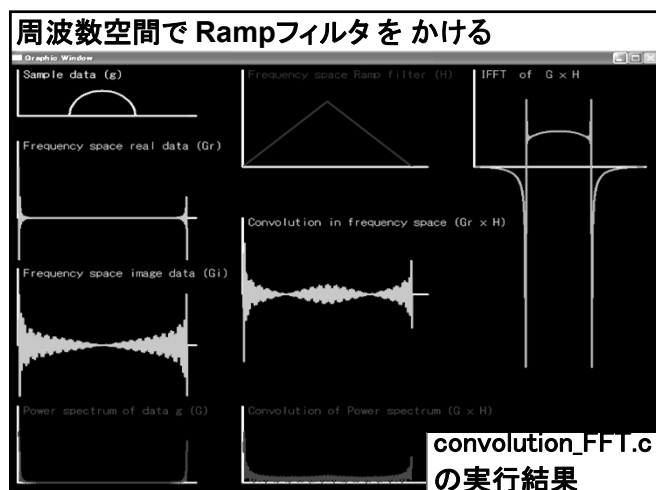
41

畳み込み Convolution  $I = g * h$

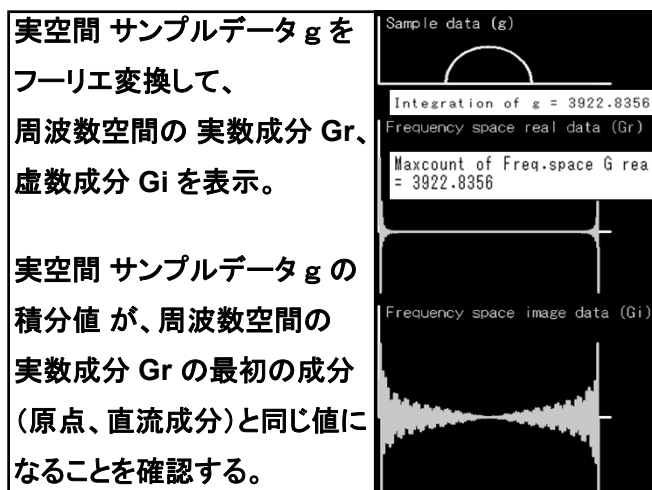
サンプルデータ  $g[]$  に 実空間Rampフィルタ  $h[]$  を畳み込んで、配列  $I[]$  に書き込む。

```
//----- Convolution h[] into g[] -----
printf("\n\n Convolution Real space Ramp filter "); scanf("%c",&syn);
for( i=0; i<=255; i++) {
    gh = 0.0;
    for(j=0; j<=127; j++){ if( i+j < 256) gh += g[i+j] * h[j]; }
    for(j=1; j<=127; j++){ if( i-j >= 0 ) gh += g[i-j] * h[j]; }
    I[i] = gh;
}
```

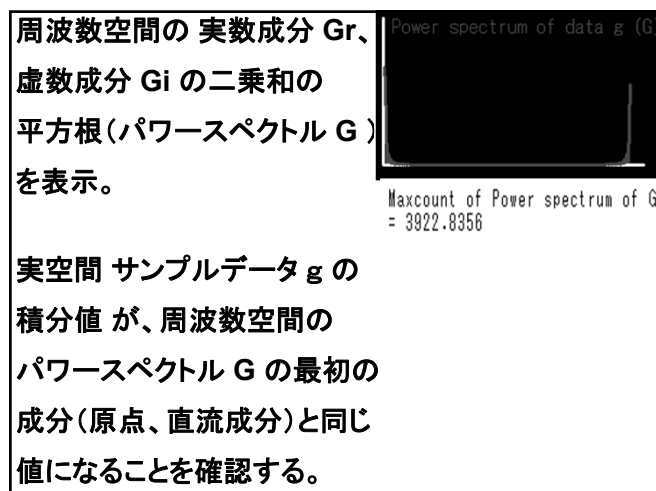
42



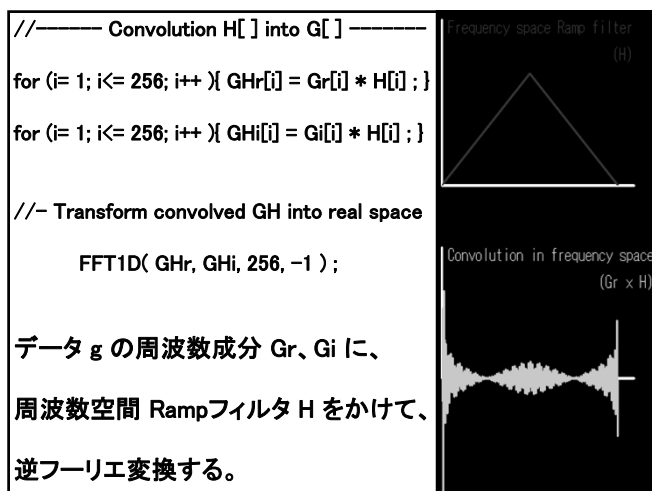
43



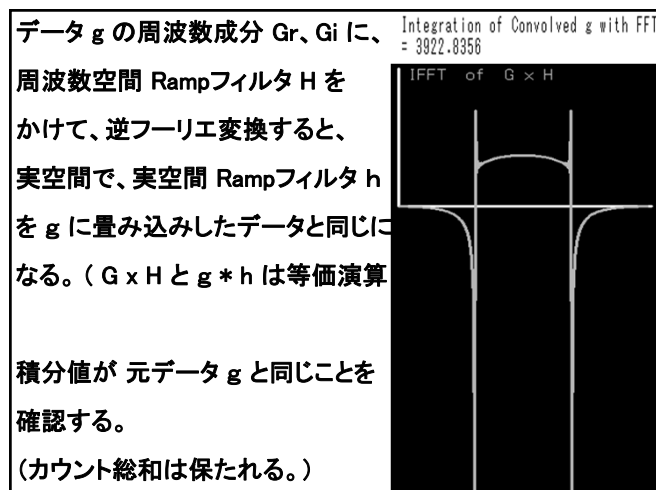
44



45



46



47