基于梯度下降优化算法对于 CT 成像的仿真和求解

# 摘要

对于这个 CT 系统，我们通过 MATLAB 仿真和模拟等方法，得到了很好的结果，并且做了相应误差分析，且过程中尽量减少误差使得结果更加精确。

针对问题 1：首先将图片中椭圆和圆的投影进行一个粗略的划分，根据圆的投影使用带冲量优化的梯度下降法拟合圆的曲线，得到圆的半径为 14.453 个探测器间距，进而计算出探测器间距为 0.2768 毫米。根据这一距离和椭圆投影得到精确的椭圆中心和椭圆偏角，进而得到精确的圆投影图像，用三角函数拟合椭圆和圆中心的轨迹，计算出椭圆和圆中心到旋转点的距离。通过三角计算得到椭圆中心的位置（-9.2663，6.2728）和射线的 180 个方向。

针对问题 2：使用 Matlab 提供的拟 radon 变换算法计算出未知介质的信息，再用三次插值算法（cubic）得到所要求点的吸收率数值。

针对问题 3：同样，用了 radon 变换算法精确地计算了图三所给出的 10 个位置处的吸收率。

针对问题 4：我们采用优化的方法对于数据进行了优化和估计，得到了较好的结果。

关键词：梯度下降优化（结合冲量方法），图像重建仿真，曲线拟合，逆 radon 变换

# 一、问题重述

CT 成像是在利用所测试数量场上利用数量场的各个不同位置的密度不同来进行断层成像（所用的射线为 X 射线）。每一个小问都承接着上一问，环环相扣，很好地指引我们一步一步分析，并且仿真建模，还原了一个系统的 CT 成像系统。

# 二、问题分析

CT 成像原理

众所周知，现在很多 CT 成像有非常多的方法，比如 CT 分析法，CT 数值分析法，CT 数和灰度联合分析法，数字体积相关法等等

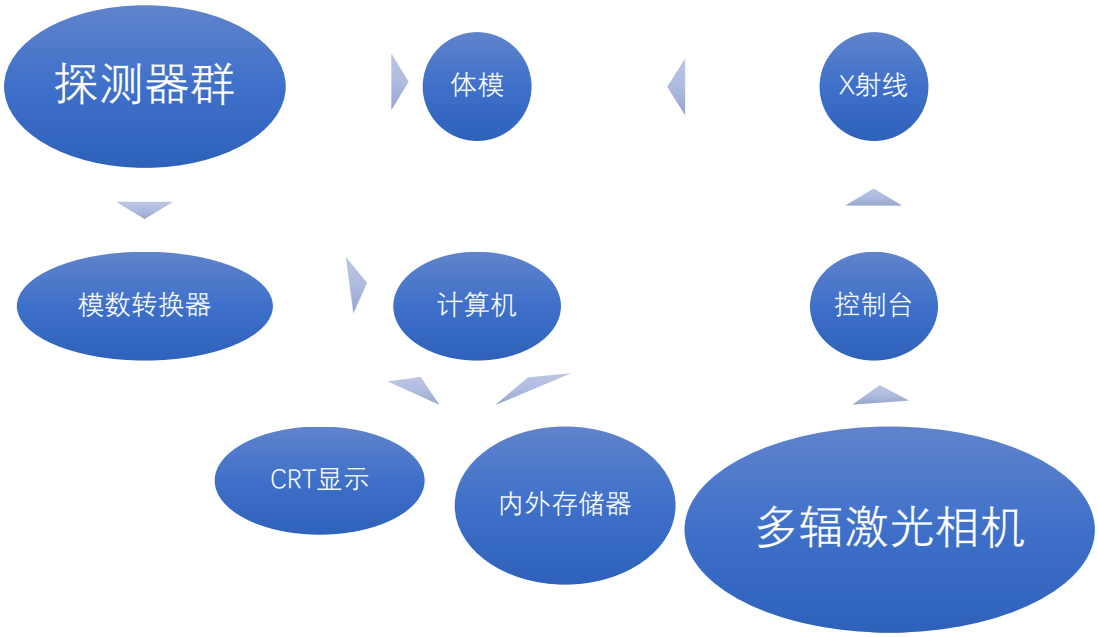
我们认为，CT 成像的实质是由于物质的内在结构不同，X 射线对不同部位有着不同的穿透性（这里可以引入穿透系数来反应这一现象）。对于任意一个三维物体，物体各个部位对于 X 射线的吸收强度不同，导致穿过的 X 射线在投影面上出现强弱的不同，反映到电子设备上来说，具体图像的灰度就相应会有所不同，由此可以获得相应数据，这样便可以相应建模构建出 CT 成像。

理论依据

经过查阅资料，我们知道，CT 的本质是一种利用 X 射线穿透三维数量场后的衰减特性作为判断的依据。

X 射线在介质中的衰减规律为 $\frac{dI}{I}=-\mu dx$，式中：$I$ 表示入射光强，$dI$ 表示光强的变化，$dx$ 表示 X 光的穿透距离，$\mu$ 表示物质对 X 光的吸收率，这是一个与物质结构和 X 光频率有关的函数，当 X 光频率不太高且不在某些原子的特征吸收频率时，可以认为只与物质结构相关。积分上式得到 $I=I_0e^{\int-\mu dx}$，其中 $I_0$ 表示初始入射光强，积分号是对 X 射线经过的路径积分。

一般的 CT 设备的基本结构




图形 1；CT 系统基本结构图

## 三、模型假设

1. X 光在待测物中呈直线传播并遵从理想的衰减规律 $\frac{dI}{I}=-\mu dx$，不考虑散射反射等效应
2. 探测器探测值正比于 X 光所经过路径的吸收率之积分 $\int_L \mu dx$

附注：模型假设的说明

第一条假设是出于物理上的原因，医用 X 光属于软 X 射线，能量较高，波动性较弱，衍射效果可以忽略，而软 X 光一个光子的能量远小于电子的静能量，因此康普顿效应导致的光子散射也应该忽略。

第二条假设是核心假设，因为题干没有给出任何关于探测器性能的信息，所以探测器的性能信息只能从表二中加以推测。由于 X 光直接射入探测器时探测器示数为 0，而被吸收后射入探测器时探测器有正的实数，因而探测器的示数与 X 光被吸收的强度成正相关，进一步 X 光经过小圆圆心附近（光程 20 个格点）探测器示数约为 14.17，而表中最大数值约为 141.7，是 X 光经过椭圆长轴（205 个格点）附近探测器探测到的示数。因此假设探测器探测值正比于 X 光所经过路径的吸收率之积分 $\int_L \mu dx$

## 四、符号说明

| 符号 | 意义 |
|---|---|
| $x_i$ | 第$i$个探测器的位置 |
| $y_i$ | 第$i$个探测器的读数 |
| $\hat{y}_i$ | 给定位形下计算得到的第$i$个探测器的理论读数 |
| $x_\theta$ | 椭圆中心的投影位置 |
| $\theta$ | $y$轴正向与X射线光矢量的夹角 |
| $r$ | 椭圆中心与旋转中心的距离 |
| $\alpha$ | 椭圆中心与旋转中心连线与$y$轴正向的夹角 |
| $R$ | 小圆的半径 |
| $a$ | 椭圆的长半轴长或小圆圆心的投影位置 |
| $b$ | 椭圆的短半轴长 |
| $L$ | 偏差平方的求和$\sum\left(y_i-\hat{y}_i\right)^2$ |
| $j$ | $y$轴基矢 |
| $k$ | X光光矢量 |

## 五、问题一的模型的建立和求解

模型：

（1）探测器间距

首先利用小圆的X射线强度投影来计算探测器间距。观察表中前几列可知，此时小圆和椭圆的投影是分开的。考虑第一列，不失一般性，假设接收小圆投影的29个探测器的间距为 1，第一个接收投影的探测器位置记为$x_1=0$，记小圆圆心位置的横坐标为$a$，小圆半径在此比例尺下为$R$，探测器示数为$n$时对应的此比例

尺下的小圆弦长为$\lambda n$，第$i$条弦长的理论值为$\hat{y}_i = 2\lambda\sqrt{R^2-(x_i-a)^2}$，理论值与测量值的偏差的平方为$\left(y_i-\hat{y}_i\right)^2$，圆的实际位置应使得$\sum_{i=1}^{29}\left(y_i-\hat{y}_i\right)^2$取最小值。

对$L=\sum_{i=1}^{29}\left(y_i-\hat{y}_i\right)^2$中的各个变量求偏导数可得：

$$\frac{\partial L}{\partial \lambda}=\sum_{i=1}^{28}(\hat{y}_i-y_i)\times 2\times\sqrt{R^2-(x_i-a)^2}$$

$$\frac{\partial L}{\partial R}=\sum_{i=1}^{28}2\times(y_i-\hat{y}_i)\times(-2\lambda)\times\frac{R}{\sqrt{R^2-(x_i-a)^2}}$$

$$\frac{\partial L}{\partial a}=\sum_{i=1}^{28}2\times(y_i-\hat{y}_i)\times(-1)\times(2\lambda)^2\times\frac{-(a-x_i)}{\hat{y}_i}$$

（2）旋转中心和旋转角度

考虑一个以坐标轴为对称轴的椭圆$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$，使其逆时针旋转$\theta$角

利用坐标变换$\begin{pmatrix}x'\\y'\end{pmatrix}=\begin{pmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}$

即$\begin{pmatrix}x\\y\end{pmatrix}=\begin{pmatrix}\cos(\theta) & \sin(\theta)\\ -\sin(\theta) & \cos(\theta)\end{pmatrix}\begin{pmatrix}x'\\y'\end{pmatrix}$，将之带入原椭圆方程

得到新的椭圆方程：$\frac{(\cos(\theta)x'+\sin(\theta)y')^2}{a^2}+\frac{(-\sin(\theta)x'+\cos(\theta)y')^2}{b^2}=1$

考虑$y'=k$直线所截椭圆得到的弦，将$y'=k$带入椭圆方程，得到

$$(\frac{(\cos\theta)^2}{a^2}+\frac{(\sin\theta)^2}{b^2})x'^2+\sin(2\theta)k(\frac{1}{a^2}-\frac{1}{b^2})x'+(\frac{(\sin\theta)^2}{a^2}+\frac{(\cos\theta)^2}{b^2})k^2-1=0$$

解得弦长为：

$$y=\lambda\frac{\sqrt{(\sin(2\theta)(x-c)(\frac{1}{a^2}-\frac{1}{b^2})-4(\frac{(\cos\theta)^2}{a^2}+\frac{(\sin\theta)^2}{b^2})(\frac{(\sin\theta)^2}{a^2}+\frac{(\cos\theta)^2}{b^2})(x-c)^2-1)}}{(\frac{(\cos\theta)^2}{a^2}+\frac{(\sin\theta)^2}{b^2})}=\lambda\frac{\sqrt{f(\theta,c)^2-4g(\theta)h(\theta,c)}}{g(\theta)}$$

回到问题中来。首先我们把小圆投影与椭圆投影分开（这里算法的实现需要补充）对椭圆而言以 X 射线入射方向为 $y$ 轴的正方向，假设椭圆长轴与 $y$ 轴夹角为 $\theta$（从 $y$ 轴向长轴转动的逆时针方向为正方向），椭圆圆心位置的投影为 $x_\theta$，探测器间隔取为实际间隔，椭圆长短轴长也取为实际长度，通过与（1）中相同的方法，可以得到夹角 $\theta$ 和椭圆圆心位置的投影 $x_\theta$。假设椭圆中心与旋转中心距离为 $r$，椭圆中心与旋转中心的连线与椭圆长轴所夹的角度为 $\alpha$（$\alpha$ 是椭圆长轴与连线所夹的锐角以椭圆半长轴向连线逆时针旋转为正方向）通过对椭圆中心轨迹的拟合 $x_\theta = r\cos(\theta+\alpha)+x_0$，可以算出 $r$ 和 $\alpha$，这确定了旋转点的两个可能位置，再对圆心做相同的操作，可以确定旋转点相对于圆心的两个可能位置，进而可以确定旋转中心的实际位置。

（3）圆投影处理

由于数据是离散取样，仅仅通过投影的长度估计圆的直径是非常不精确的，为了得到精确的结果，我们选择了拟合投影曲线，由于椭圆投影和圆投影存在不相交的区域，为我们的拟合提供了可能。投影数值的大小正比于射线穿透的圆的长度，公式表述为：

$$l = 2\sqrt{R^2 - y^2}$$

考虑到比例系数，投影曲线应该形如 $I = 2\lambda\sqrt{R^2 - y^2}$

实际投影曲线 $I_{real}$，我们通过最小化平方残差 $L(R,\lambda) = \sum (I_{real} - I)^2$

最小化的办法是使用冲量优化的梯度下降算法：

$$v_{n+1} = \eta v_n - \nabla L$$

$$(R_{n+1,}\lambda_{n+1}) = (R_n,\lambda_n) + v_{n+1}$$

$\eta$ 起到了阻尼的作用，而冲量在每次得到的下降方向相同的时候可以使下降速度逐渐累加，加速下降过程；在每次得到的下降方向不同的时候可以避免在函数性质不好的地方来回震荡。

最终我们根据 68 组不同的数据分别得到了 $R$ 和 $\lambda$，方差分别为 $6.5\times10^{-11}$ 和 $2.0\times10^{-13}$，达到了精度要求。

我们最终取平均值，得到：

$R=14.4533,\lambda=0.4905$
这样可以得到每个探测器之间的距离为：0.2768

（4）椭圆投影处理

4.1 数学公式

当椭圆长轴相对射线相对夹角为$\theta$时，根据旋转公式：

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$$

$$\begin{pmatrix}x'\\y'\end{pmatrix}=\begin{pmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}$$

$$\begin{pmatrix}x\\y\end{pmatrix}=\begin{pmatrix}\cos(\theta) & \sin(\theta)\\ -\sin(\theta) & \cos(\theta)\end{pmatrix}\begin{pmatrix}x'\\y'\end{pmatrix}$$

$$\frac{(\cos(\theta)x'+\sin(\theta)y')^2}{a^2}+\frac{(-\sin(\theta)x'+\cos(\theta)y')^2}{b^2}=1$$

令$y=k$，我们可以得到：

$$(\frac{(\cos\theta)^2}{a^2}+\frac{(\sin\theta)^2}{b^2})x'^2+\sin(2\theta)k(\frac{1}{a^2}-\frac{1}{b^2})x'+(\frac{(\sin\theta)^2}{a^2}+\frac{(\cos\theta)^2}{b^2})k^2-1=0$$

$$y=\lambda\frac{\sqrt{(\sin(2\theta)(x-c)(\frac{1}{a^2}-\frac{1}{b^2})-4(\frac{(\cos\theta)^2}{a^2}+\frac{(\sin\theta)^2}{b^2})(\frac{(\sin\theta)^2}{a^2}+\frac{(\cos\theta)^2}{b^2})(x-c)^2-1)}}{(\frac{(\cos\theta)^2}{a^2}+\frac{(\sin\theta)^2}{b^2})}=\lambda\frac{\sqrt{f(\theta,c)^2-4g(\theta)h(\theta,c)}}{g(\theta)}$$

为了射线和椭圆存在交点，我们可以看到根号中的式子必须是正数，可以得到

$$(x-c)^2<\frac{4(\frac{(\cos\theta)^2}{a^2}+\frac{(\sin\theta)^2}{b^2})}{4(\frac{(\cos\theta)^2}{a^2}+\frac{(\sin\theta)^2}{b^2})(\frac{(\sin\theta)^2}{a^2}+\frac{(\cos\theta)^2}{b^2})-(\sin(2\theta)(\frac{1}{a^2}-\frac{1}{b^2}))}$$

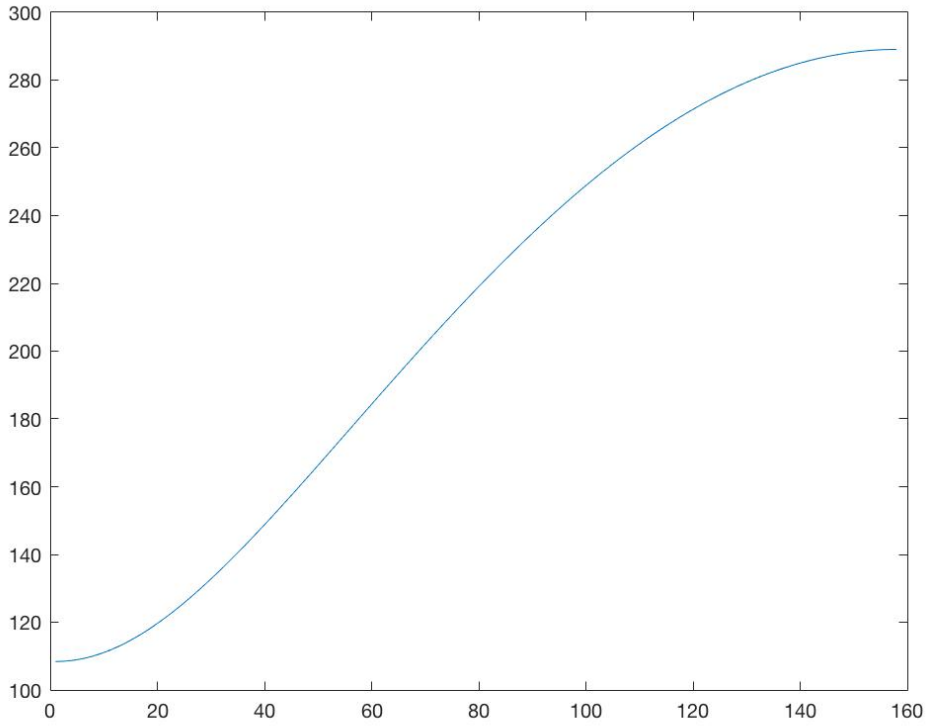
这样投影的宽度应该是：

$$wide(\theta)=4\sqrt{\frac{(\frac{(\cos\theta)^2}{a^2}+\frac{(\sin\theta)^2}{b^2})}{4(\frac{(\cos\theta)^2}{a^2}+\frac{(\sin\theta)^2}{b^2})((\frac{(\sin\theta)^2}{a^2}+\frac{(\cos\theta)^2}{b^2}))}}$$

这样通过得到的实际投影宽度 w 可以给出椭圆的偏角 $\theta = wide^{-1}(w)$，但是由于这时无法得到精确的实际宽度并且 $wide(\theta)$ 的函数形式非常复杂，这里我们通过如下方法给出一个偏角的大概估计：

4.2 画出 wide 的图像

以 0.01 弧度为间隔从 0 到 $\pi/2$ 中取点 $\{a_n\}$，计算得到对应的 $wide(a_n)$，$\{a_n\}$ 随 n 变化的图像如图：



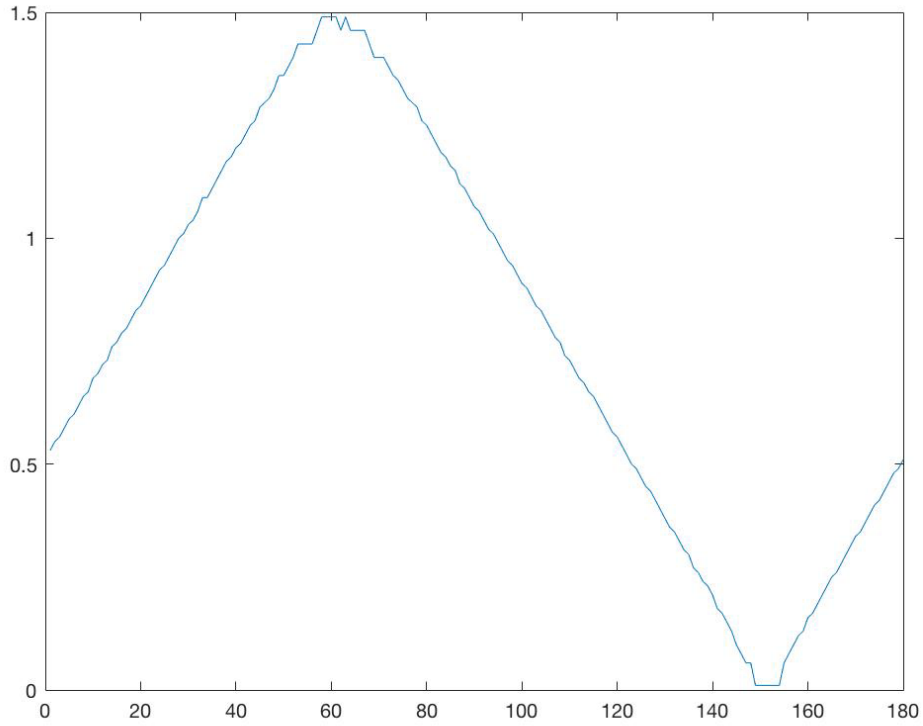

图形 2

4.3 计算粗略偏角

对于每一个实际投影宽度 w，其偏角 $wide^{-1}(w)$ 由如下估计：

$$wide^{-1}(w) \approx \min\{a_n \mid wide(a_n) > w\}$$

这样可以得到 180 次拍摄中椭圆长轴的偏角 guessthetaA 图像为：



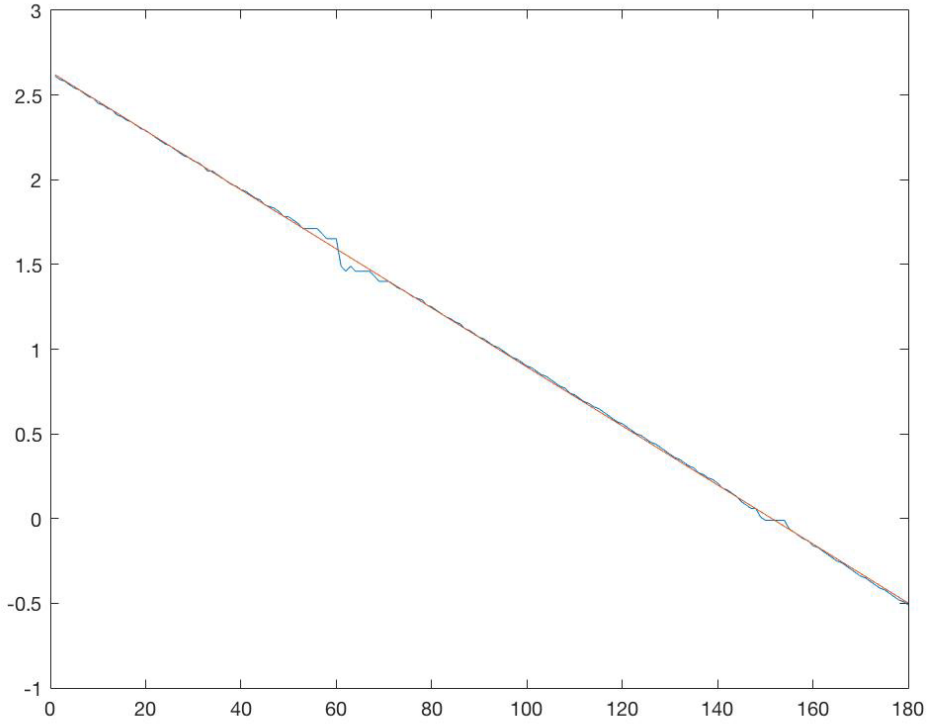

图形 3

4.4 调整

参考实际图像，进行如下调整：

```
guessthetaA(1:60) = pi - guessthetaA(1:60);
guessthetaA(150:180) = - guessthetaA(150:180);
```

得到了：




图形 4

可以看到近似为一条直线。

4.5 计算精确偏角和中心位置

使用梯度下降下最小化平方残差和：

$$L(\theta,c)=\sum(y(\theta,c)-y_{real})^2$$

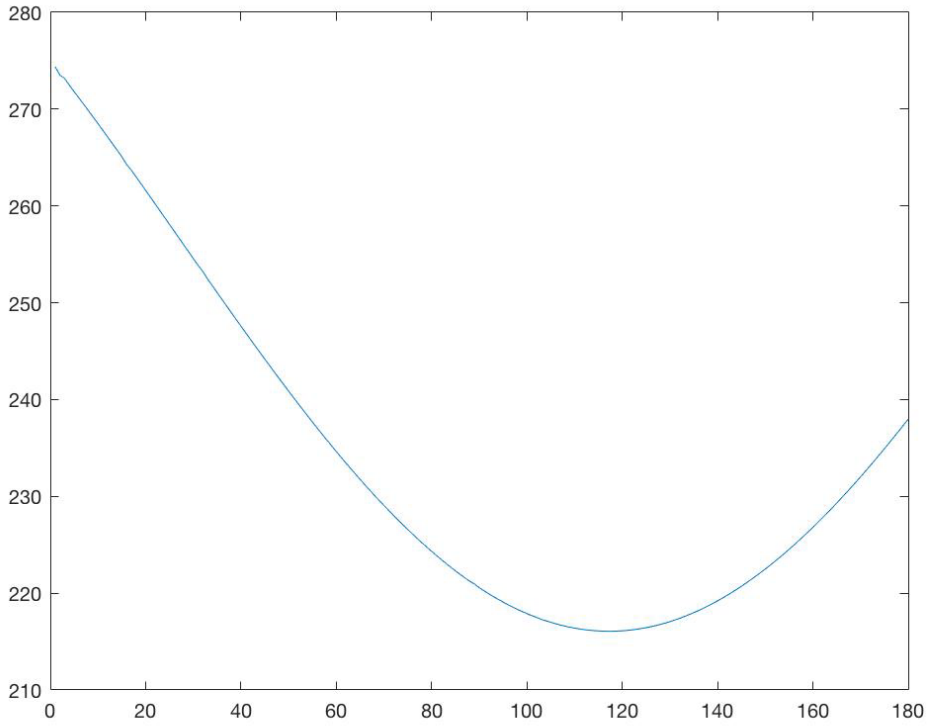
$$\frac{\partial L(\theta,c)}{\partial \theta}=\sum 2(y(\theta,c)-y_{real})\frac{\partial y(\theta,c)}{\partial \theta}$$

$$\frac{\partial L(\theta,c)}{\partial c}=\sum 2(y(\theta,c)-y_{real})\frac{\partial y(\theta,c)}{\partial c}$$

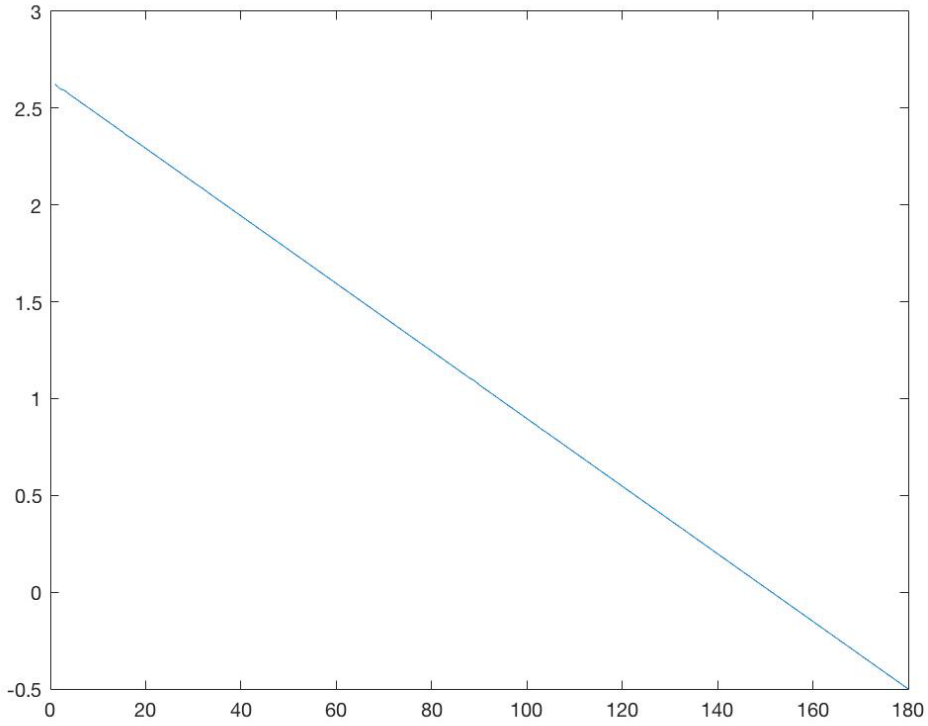
$$\frac{\partial y}{\partial \theta}=\lambda\frac{\dfrac{f(\theta)\dfrac{\partial f(\theta,c)}{\partial \theta}-2(g(\theta)\dfrac{\partial h(\theta,c)}{\partial \theta}+g'(\theta)h(\theta,c))}{\sqrt{f(\theta)^2-4g(\theta)h(\theta)}}g(\theta)-g'(\theta)\sqrt{f(\theta,c)^2-4g(\theta)h(\theta,c)}}{g^2(\theta)}$$

$$\frac{\partial y}{\partial c}=\frac{\lambda(2f(\theta,c)\dfrac{\partial f(\theta,c)}{\partial c}-4g(\theta)\dfrac{\partial h(\theta,c)}{\partial c})}{2g(\theta)\sqrt{f(\theta)^2-4g(\theta)h(\theta,c)}}$$

使用带冲量的梯度下降算法最小化 $L(\theta,c)$，由于我们有前述的粗略估计，所以能得到令人满意的结果。


280
270
260
250
240
230
220
210
0
20
40
60
80
100
120
140
160
180

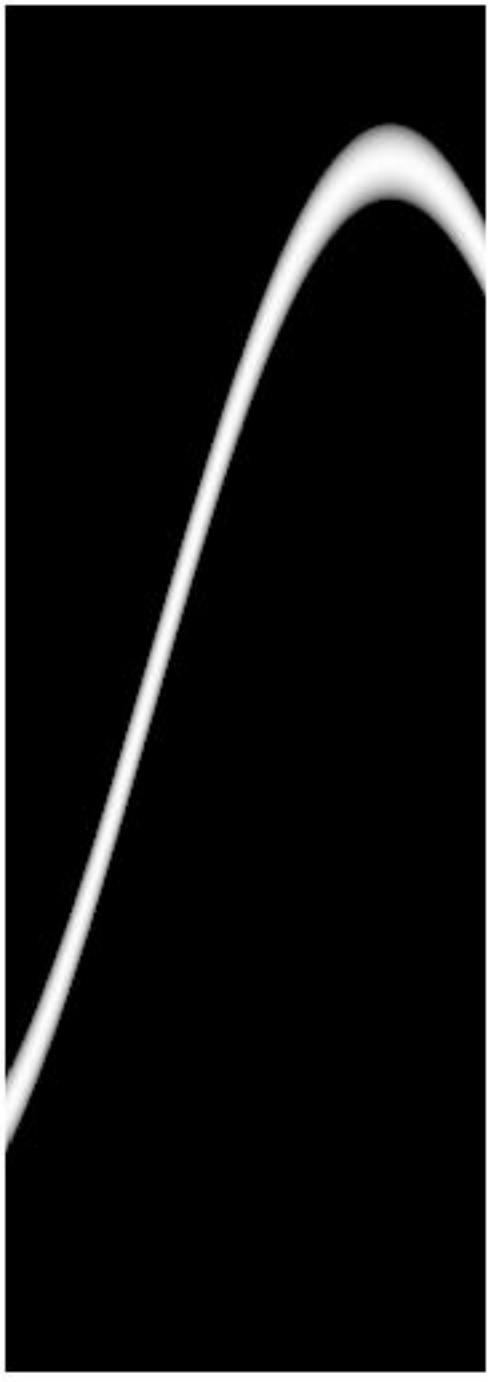
图形 5


3
2.5
2
1.5
1
0.5
0
-0.5
0
20
40
60
80
100
120
140
160
180

图形 6

（4）得到精确的圆投影中心曲线

当我们得到精确的椭圆中心及偏移角度后，就可以重构出精确的椭圆投影，原投影图片减去椭圆投影就是原投影图像：

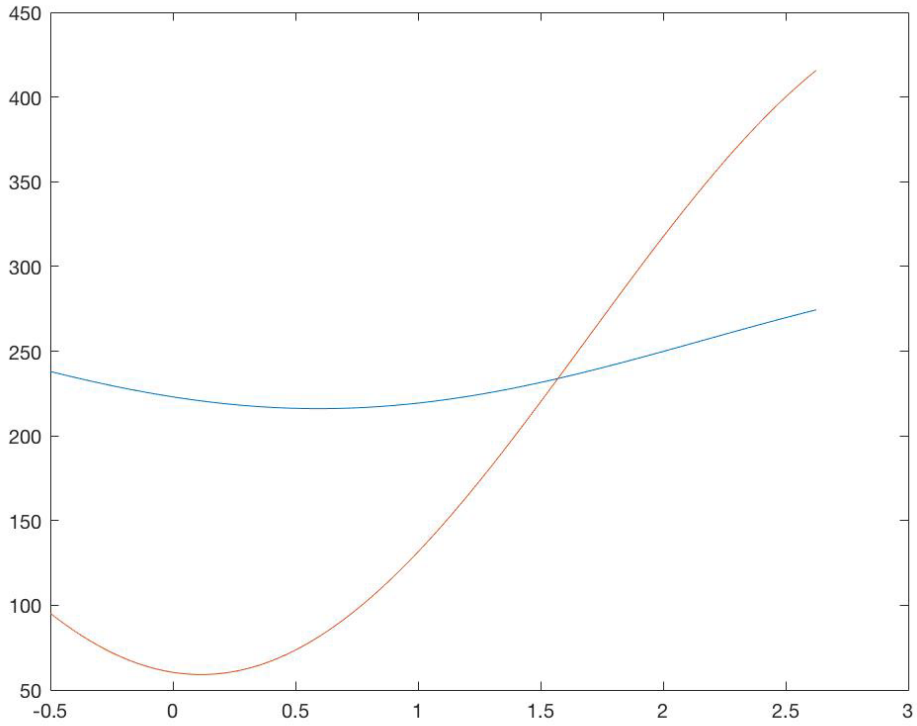


图形 7

使用第一步中的梯度下降算法得到圆投影中心的精确曲线。

（5） 三角曲线拟合

将圆中心和椭圆中心随椭圆长轴与射线夹角 thetaA 关系曲线放在一起的到如图：




图形 8

这显然是一个三角曲线，我们对其进行拟合，需要拟合的公式为：

$y = A\sin(thetaA + phi) + Bias$

同样使用加冲量的梯度下降法得到

椭圆的参数为：

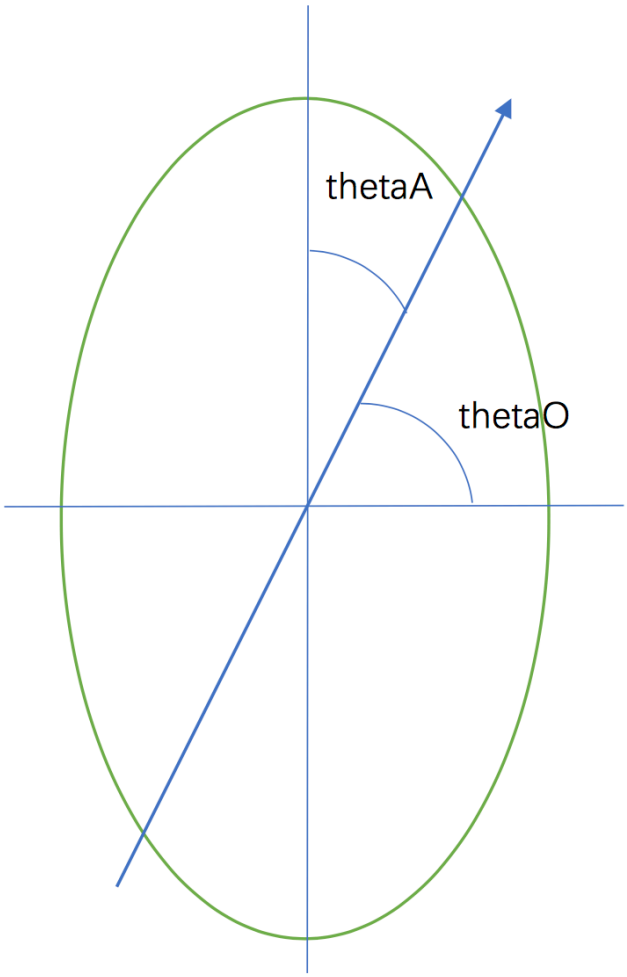
A（40.4325），phi（-2.1659），Bias（256.5）；

圆的参数为：

A（197.3870），phi（-1.6859），Bias（256.5）

Bias 相同说明了旋转中心投影恰好在屏幕的中心。

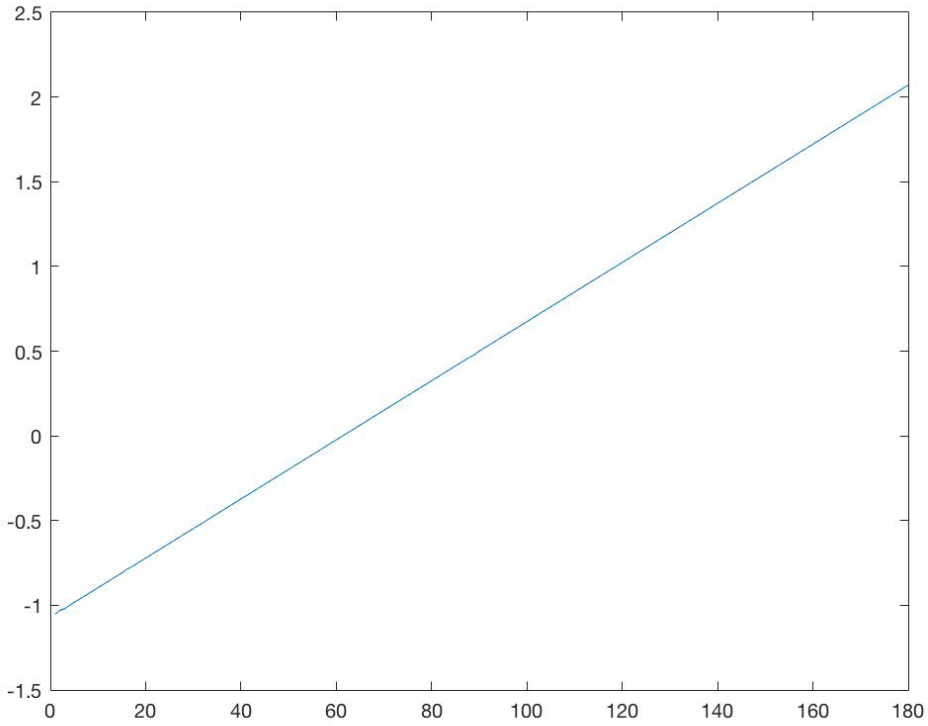
振幅恰好是中心到旋转点的距离，至此我们得到了椭圆的中心和圆的中心距旋转中心的距离分别为：11.1899 和 54.6277。有圆和椭圆中心的距离为 45，根据简单的三角计算可得旋转中心的位置是（-9.2663，6.2728），旋转中心相对两几何体中心的偏角为 0.4800。




图形 9

而射线与 x 正方向的夹角和 thetaA 恰好互余，所以射线的角度为$\pi/2-thetaA$

图像如图所示：




图形 10

## 六、问题 2 和 3 的模型的建立和求解

使用 Matlab 提供的拟 radon 变换算法计算出未知介质的信息，再用三次插值算法（cubic）得到所要求点的吸收率数值。（具体数据都存放在了 problem2.xls 和 problem3.xls 中）

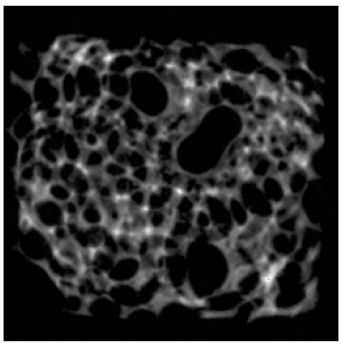
其中值得注意的是：由于数值都不是整点（偏移量是小数），所以用了 cubic 插值算法，插值函数用的是 interp2

具体的各个点 Point = [10.0000, 18.0000 ;
34.5000 ,25.0000;
43.5000 ,33.0000;
45.0000 ,75.5000;
48.5000 ,55.5000 ;
50.0000 ,75.5000 ;
56.0000 ,76.5000 ;
65.5000 ,37.0000 ;

79.5000 ,18.0000 ;
98.5000 ,43.5000 ;];
（其中十个点数据颠倒）
最后主要由 step5.m 中的程序可以计算得出。
最终的结果都存放在了附件中的 problem2.xls 和 problem3.xls 中

最终重构的图像如下图所示

图形 11



图形 12

第二问的结果

0
0
1.0061
0.0002
1.0074
0.0002
0.0035
1.2960
0
0.0060

第三问的结果

0.0506
0.1974
0.3641
1.7199
1.3103
4.2497
1.4668
3.4117
1.2886
3.7889

# 七、误差分析

得到椭圆中心与旋转中心的距离 $dA$，拟合结果残差的标准差 $epsilonA$=3.92*10$^{-4}$$mm$ 和小圆中心与旋转中心距离 $dB$，拟合结果残差的标准差 $epsilonB$=7.84*10$^{-4}$$mm$ 之后，利用余弦定理，解以椭圆中心，小圆中心和旋转中心为顶点的三角形。得到旋转中心的坐标为（另一个由于不符合逆时针转动，舍去）

$$X=\frac{dA^2+2025-dB^2}{90}$$

$$Y=\sqrt{dA^2-X^2}$$

由不确定度传递公式，得到 $X$ 的测量不确定度为：

$$epsilonX=\sqrt{\left(\frac{dA*epsilonA}{45}\right)^2+\left(\frac{dB*epsilonB}{45}\right)^2}$$

$Y$ 的测量不确定度为：

$$epsilonY=\sqrt{\left(\frac{dA*epsilonA}{Y}\right)^2+\left(\frac{X*epsilonX}{Y}\right)^2}$$

带入相应的数值之后，计算得到：

$$epsilonX=9.6*10^{-4}mm$$

$$epsilonY=16.0*10^{-4}mm$$

可以看出，利用椭圆和圆的复合图形求解旋转点位置最终得到的结果误差较大，而之后利用 iradon 算法重构吸收率分布函数时 3 极度依赖旋转点位置的精确度的，因此此模板给出的测量结果会有误差。

仔细分析原模板的问题，我们可以发现利用给定的圆去得到探测器间距的算法精度是很高的。而由于椭圆的弦长计算公式极其复杂，在利用椭圆计算旋转角度时每个初始数据经过多次运算误差会很快增加，这会导致测量精度急剧地降低，同时不同实验下定标结果也会有较大波动，稳定性较低。

可以将原模板中的椭圆换为边长为 60㎜ 的正方形，而旁边的小圆位置不变。正方形对角线与 $y$ 轴正方向所夹的较小的锐角 $\beta$（$\beta$ 介于 0 到 45 度）满足，由于

正方形做 radon 变换之后得到的是分段线性的函数，而$\beta$只与该分段线性函数的斜率有关，因此$\beta$有一个简单的线性表达式，计算量会比椭圆弦长计算转角少许多，因此具有更高的精确度和稳定性。

## 八、模型评价

模型的优点：

- 使用大致估计一开始在数据很少的情况下粗略得到各个参数的估计值，方便梯度下降的收敛速度增加并且不会呈发散趋势，而且容易极大程度上减小我们找到的极小值只是局部的，而不是
- 使用深度学习的神经网络中的梯度下降法很好地非线性拟合了数据，且非常有效地减少了误差
- 为了加快梯度下降的速度，在迭代的过程中，引入了冲量进行加速：
  $v_{n+1} = \eta v_n - \alpha \nabla f(a_n)$
  $a_{n+1} = a_n + v_{n+1}$
- 冲量在每次得到的下降方向相同的时候可以使下降速度逐渐累加，加速下降过程；在每次得到的下降方向不同的时候可以避免在函数性质不好的地方来回震荡。

模型的缺点：

- 梯度下降法进行拟合，得到的是局部的最小值，有时候并不是全局最小值，解决办法是事先先进行实现估计粗略结果，这样可以在预定估计周边寻找极小值点，可以有效避免局部与整体之间关系。
- CT 系统的误差会逐步累加，前期的误差会逐渐放大，不利于后面准确模拟和仿真，同时，我们也尽可能每一步保留小数精确，减小这种误差，不过由于后期计算过于繁琐，误差还是存在。

# 九、参考文献

[1]．姜启源 谢金星 叶俊：数学模型（第三版） 高等教育出版社.2004
[2]．卓金武：MATLAB 在数学模型中的应用 北京航空航天大学出版社.2011
[3]．薛定宇 陈阳泉：高等应用数学问题的 MATLAB 求解.清华大学出版社.2004
[4]．杨福家. 原子物理
[5]．Duane Hanselman、Bruce Littlefield 著，朱仁峰译，Matlab 7，北京：清华大学出版社，2006 年 5 月第一版
[6]．丁卫华，许彦卿，薄毅彬等. X 射线岩石 CT 的历史与现状[J]. 地震地质，2003，25(3)：467～467
[7]．Bay BK，Smith T S，Fyhrie D P，et al. Digital volume correlation： three-dimensional strain mapping using X-ray topography[J]. Experimental Mechanics，1999，39(2)：217～226
[8]．赵荣椿. 数字图像处理导论[M]. 西安：西北工业大学出版社，1995

# 十、附件程序

1、总程序 RUNME.m

```
step1
step2
step3
step4
step5
```

2、子程序

1)bound.m

```
function [ down, up] = bound( mask )

down = zeros(1,size(mask,2));
up = zeros(1,size(mask,2));

for i = 1:size(mask,2)
      down(i) = find(mask(:,i)==1, 1 );
      up(i) = find(mask(:,i)==1, 1, 'last' );
end

end
```

2)expandmask.m

```
function [newmask] = expandmask(mask, k)

[n,m] = size(mask);
temp = (zeros(n+2*k,m+2*k)==1);
temp((k+1):(k+n),(k+1):(k+m)) = mask;
newmask = mask;
for i = k+1:k+n
      for j = k+1:k+m
            if sum(sum(temp((i-k):(i+k),(j-k):(j+k))))~=0
                  newmask(i-k,j-k) = 1;
            end
      end
end

end
```

3)gradient.m

```
function [lambda, R, a, L] = gradient(alpha, B)

lambda = 0.5;
R = 14;
a = 14;
X = 1:27;

v = zeros(3,1);
L = 1;
LastL = 100;



while L>1e-10 && (LastL - L > 1e-20   || L > 0.01) && isreal(L)
      Bdot = 2*lambda*sqrt(R*R-(X-a).^2);
      dlambda = sum(2*(Bdot-B).*Bdot/lambda);
      dR = sum(2*(Bdot-B)*4*lambda^2*R./Bdot);
      da = sum(2*(Bdot-B)*4*lambda^2.*(X-a)./Bdot);
      v = 0.9*v - [dlambda;dR;da]*alpha;
      lambda = lambda + v(1);
      R = R + v(2);
      a = a + v(3);
      LastL = L;
      L = sum((Bdot-B).^2);
end
```

4)gradientB.m

```
function [center, L] = gradientB(x, y, center, lambda, R, alpha)

v = 0;
L = 1;
LastL = 100;

n = 1;

while L>1e-11 && (LastL - L > 1e-10   || (L > 0.01 && n<200)) &&
isreal(L)
      n = n+1;
      ydot = 2*lambda*sqrt(R*R-(x-center).^2);
      dcenter = sum(2*(ydot-y)*4*lambda^2.*(x-center)./ydot);
      v = 0.9*v - dcenter * alpha;
      center = center + max([min([v,0.1]),-0.1]);
```

```
      LastL = L;
      if n>2000 && alpha < 1e-7
            alpha = 1e-7;
      end
      L = sum((ydot-y).^2);
end
```

5)gradientfixed.m

```
function [a, L] = gradientfixed(alpha, B, R, lambda)

a = 14;
X = 1:27;

v = 0;
L = 1;
LastL = 100;



while L>1e-11 && (LastL - L > 1e-20   || L > 0.01) && isreal(L)
      Bdot = 2*lambda*sqrt(R*R-(X-a).^2);
%      dlambda = sum(2*(Bdot-B).*Bdot/lambda);
%      dR = sum(2*(Bdot-B)*4*lambda^2*R./Bdot);
      da = sum(2*(Bdot-B)*4*lambda^2.*(X-a)./Bdot);
      v = 0.9*v - da*alpha;
%      lambda = lambda + v(1);
%      R = R + v(2);
      a = a + v;
      LastL = L;
      L = sum((Bdot-B).^2);
end
```

6)gradientoval.m

```
function [ center, theta , L] = gradientoval( x , y , center , theta,
lambda, a, b, alpha)
%UNTITLED5 Summary of this function goes here
%    Detailed explanation goes here

v = zeros(2,1);
L = 1;
LastL = 100;
```

```
n = 1;

while L>1e-10 && (LastL - L > 1e-10 || (L > 0.01 && n<200))
      n = n + 1;
      f = sin(2*theta)*(x-center)*(1/a^2-1/b^2);
      fpt = 2*cos(2*theta)*(x-center)*(1/a^2-1/b^2);
      fpc = -sin(2*theta)*(1/a^2-1/b^2);
      g = cos(theta)^2/a^2 + sin(theta)^2/b^2;
      gpt = sin(2*theta)/b^2-sin(2*theta)/a^2;
      gpc = 0;
      h = (cos(theta)^2/b^2 + sin(theta)^2/a^2) * (x-center).^2-1;
      hpt = (x-center).^2*(1/a^2-1/b^2)*sin(2*theta);
      hpc = -(cos(theta)^2/b^2 + sin(theta)^2/a^2) * 2 * (x-center);
      s = sqrt(f.*f-4*g*h);
      ydot = lambda*s/g;
      ypt = lambda*(((f.*fpt-2*(g*hpt+gpt*h))./s*g)-gpt*s)/g^2;
      ypc = lambda*(((f.*fpc-2*(g*hpc+gpc*h))./s*g)-gpc*s)/g^2;
      dcenter = sum(2*(ydot-y).*(ypc));
      dtheta = sum(2*(ydot-y).*(ypt));
      v = 0.9*v - [dcenter;dtheta]*alpha;
      center = center + v(1);
      theta = theta + v(2);
      LastL = L;
      L = sum((ydot-y).^2);
end

end
```

7) processmask.m

```
function [mask] = processmask( mask )
%UNTITLED2 Summary of this function goes here
%    Detailed explanation goes here

[n,m] = size(mask);

for i = 1:n
      for j = 2:m-1
            if mask(i,j) == 1 && mask(i,j+1) == 0 && mask(i,j-1) == 0
                  mask(i,j) = 0;
            end
```

```
      end
end
for i = 2:n-1
      for j = 1:m
            if mask(i,j) == 1 && mask(i+1,j) == 0 && mask(i-1,j) == 0
                  mask(i,j) = 0;
            end
      end
end
for i = 3:(n-2)
      for j = 3:(m-2)
                if  mask(i,j)  ==  1  &&  sum(sum(mask((i-2):(i+2),(j-
2):(j+2))))<=8
                  mask(i,j) = 0;
            end
      end
end


end
```

8) sinfit.m

```
function [A, phi, B] = sinfit(theta, y, A, phi, B, alpha)


v = zeros(3,1);
L = 1;
LastL = 100;

n = 1;

while L>1e-9 && (LastL - L > 1e-12   || (L > 0.01 && n<20000)) &&
isreal(L)
      n = n+1;
      ydot = A*sin(theta+phi)+B;
      dA = sum(2*(ydot-y).*sin(theta+phi));
      dphi = sum(2*(ydot-y).*(A*cos(theta+phi)));
      dB =sum(2*(ydot-y));
      v = 0.9*v - [dA;dphi;dB] * alpha;
      A = A + v(1);
      phi = phi + v(2);
```

```
      B = B + v(3);
      LastL = L;
      if n>2000 && alpha < 1e-5
            %alpha = 1e-6;
      end
      L = sum((ydot-y).^2)
end

9) step1.m
data = xlsread('B.xlsx');

imageA = data;

location = zeros(1,180);
location(180) = 81;

B = data(81:81+28,180);

for i = 179:-1:1
      C = zeros(1,512-39);
      for j = 1:(512-29)
            if (i < 100 && j<location(i+1)) || abs(j-location(i+1)) >
10
                  C(j) = 1e10;
                  continue;
            end
            temp = data(:,i);
            temp(j:j+28) = temp(j:j+28)-B;
            error = temp(2:end)-temp(1:end-1);
            error = error' * error;
            C(j) = error;
      end
      [~,b] = min(C);
      location(i) = b;
      imageA(b:b+28,i) = imageA(b:b+28,i) - B;
end

imageA(81:81+28,180) = imageA(81:81+28,180) - B;

imageB = data - imageA;

figure
```

```
imshow(imageA/max(max(imageA)));
figure
imshow(imageB/max(max(imageB)));


N = 68;

Rn = zeros(1,N);
lambdan = zeros(1,21);

for testn = 180:-1:(181-N)
[lambda, R, a, L] = gradient(0.00001,data((location(testn)+1):(location(testn)+27),testn)');
Rn(181-testn) = R;
lambdan(181-testn) = lambda;
end

mean(Rn)
var(Rn)
mean(lambdan)
var(lambdan)

R = mean(Rn)
lambda = mean(lambdan)

centercircleB = location + R;

disp('step1 finishs');
```

10) step2.m

```
maskA = (imageA>2);
maskA = processmask(maskA);
maskB = processmask(imageB>2);
a = 10*R;
b = 15/4*R;
theta = 0:0.01:pi/2;
beta = (cos(theta).^2/a^2+sin(theta).^2/b^2);
alpha = sin(2*theta)*(1/a^2-1/b^2);
```

```
gamma = (cos(theta).^2/b^2+sin(theta).^2/a^2);
guessthick = 4*sqrt(beta./(4*gamma.*beta-alpha.^2));
[downA, upA] = bound(maskA);
thetaA = upA-downA;
guessthetaA = zeros(1,180);
for i = 1:180
      guessthetaA(i) = 0.01 * find(guessthick-thetaA(i)>0, 1 );
end
guessthetaA(1:60) = pi - guessthetaA(1:60);
guessthetaA(150:180) = - guessthetaA(150:180);
figure
plot(guessthetaA);
P = polyfit(1:180, guessthetaA, 1);
guessthetaA = polyval(P, 1:180);
hold on
plot(guessthetaA);
guesscenterA = (downA + upA) / 2;
tempmaskB = expandmask(maskB,5);
tempA = (processmask(imageA>20) & ~tempmaskB);
thetaA = zeros(1,180);
centerA = zeros(1,180);
LA = zeros(1,180);
for i = 1:180
      x = find(tempA(:,i));
        [centerA(i),    thetaA(i),    LA(i)]    =    gradientoval(x,
data(x,i),guesscenterA(i), guessthetaA(i), lambda, a, b, 1e-5);
      if isreal(LA(i)) && abs(LA(i))>1e-6 && abs(LA(i))<1e-3
                [centerA(i),  thetaA(i),  LA(i)]  =  gradientoval(x,
data(x,i),centerA(i), thetaA(i), lambda, a, b, 1e-6);
      end
      if isnan(LA(i)) || abs(LA(i)) > 1e-6 || ~isreal(LA(i))
                [centerA(i),  thetaA(i),  LA(i)]  =  gradientoval(x,
data(x,i),guesscenterA(i), guessthetaA(i), lambda, a, b, 1e-6);
      end
      if isnan(LA(i)) || abs(LA(i)) > 1e-6 || ~isreal(LA(i))
                [centerA(i),  thetaA(i),  LA(i)]  =  gradientoval(x,
data(x,i),guesscenterA(i), guessthetaA(i), lambda, a, b, 1e-7);
      end
      if isnan(LA(i)) || abs(LA(i)) > 1e-6 || ~isreal(LA(i))
                [centerA(i),  thetaA(i),  LA(i)]  =  gradientoval(x,
data(x,i),guesscenterA(i), guessthetaA(i), lambda, a, b, 1e-8);
      end
```

```
end

figure
plot(centerA);
figure
plot(thetaA);

disp('step2 finishs');
```

11) step3.m

```
newimageA = zeros(512,180);

%rebuild
for i = 1:180
      for j = 1:512
            if j < downA(i) - 5 || j > upA(i) + 5
                  continue
            end
            f = sin(2*thetaA(i))*(j-centerA(i))*(1/a^2-1/b^2);
            g = cos(thetaA(i))^2/a^2 + sin(thetaA(i))^2/b^2;
             h = (cos(thetaA(i))^2/b^2 + sin(thetaA(i))^2/a^2) * (j-
centerA(i)).^2-1;
            if f*f-4*g*h > 0
                  s = sqrt(f*f-4*g*h);
                  ydot = lambda*s/g;
                  newimageA(j,i) = ydot;
            end
      end
end

newimageB = data - newimageA;
newimageB(~tempmaskB) = 0;
figure
imshow(newimageA);
figure
imshow(newimageB);

centerB = zeros(1,180);
tempB = processmask(newimageB>6);

LB = zeros(1,180);
for i = 1:180
```

```
      x = find(tempB(:,i));
          [centerB(i),   LB(i)]   =   gradientB(x,   newimageB(x,i),
centercircleB(i), lambda, R, 1e-5);
      if isnan(LB(i)) || abs(LB(i)) > 1e-3 || ~isreal(LB(i))
                [centerB(i),  LB(i)]  =  gradientB(x,  newimageB(x,i),
centercircleB(i), lambda, R, 1e-6);
      end
      if isnan(LB(i)) || abs(LB(i)) > 1e-3 || ~isreal(LB(i))
                [centerB(i),  LB(i)]  =  gradientB(x,  newimageB(x,i),
centercircleB(i), lambda, R, 1e-7);
      end
      if isnan(LB(i)) || abs(LB(i)) > 1e-3 || ~isreal(LB(i))
                [centerB(i),  LB(i)]  =  gradientB(x,  newimageB(x,i),
centercircleB(i), lambda, R, 1e-9);
      end
end

disp('step3 finishs');
```

12)step4.m

```
  [distanceA, phiA, biasA] = sinfit(thetaA, centerA, 40.4325, -2.1659,
256.5000, 1e-5)
  [distanceB, phiB, biasB] = sinfit(thetaA, centerB, 197.3870, -1.6859,
256.5000, 1e-7);
  [distanceB, phiB, biasB] = sinfit(thetaA, centerB, distanceB, phiB,
biasB, 1e-8)

  dA = distanceA/R*4
dB = distanceB/R*4

cosalphaA = (dA^2+45^2-dB^2)/2/dA/45;
alphaA = acos(cosalphaA);

x = cosalphaA*dA
y = sqrt(1-cosalphaA*cosalphaA)*dA

cosalphaB = (dB^2+45^2-dA^2)/2/dB/45;
alphaB = acos(cosalphaB);

cosalpha0 = (dB^2+dA^2-45^2)/2/dB/dA;
alpha0 = acos(cosalpha0)
```

```
theta0 = pi/2-thetaA;


  disp('step4 finishs');
```

13)step5.m

```
d2 = xlsread('p2.xlsx');
d2 = d2(end:-1:1,:);
ttdata2 = zeros(1000,180);
ttdata2(245:756,:) = d2;

IR2 = iradon(ttdata2/lambda, (theta0-pi/2)/pi*180,'Hamming');

d1 = data;
d1 = d1(end:-1:1,:);
ttdata1 = zeros(1000,180);
ttdata1(245:756,:) = d1;

IR1 = iradon(ttdata1/lambda, (theta0-pi/2)/pi*180,'Hamming');

d3 = xlsread('p3.xlsx');
d3 = d3(end:-1:1,:);
ttdata3 = zeros(1000,180);
ttdata3(245:756,:) = d3;

IR3 = iradon(ttdata3/lambda, (theta0-pi/2)/pi*180,'Hamming');

[X,Y] = meshgrid(-352.5:352.5,-352.5:352.5);
[DX,DY]    =    meshgrid(((-127.5:127.5)/256*100/4*R+abs(x)/4*R),((-
127.5:127.5)/256*100/4*R+abs(y)/4*R));


RB1 = interp2(X,Y,IR1,DX,DY,'cubic');
RB2 = interp2(X,Y,IR2,DX,DY,'cubic');
RB3 = interp2(X,Y,IR3,DX,DY,'cubic');

RB1(RB1<0.02) = 0;
RB2(RB2<0.02) = 0;
RB3(RB3<0) = 0;

xlswrite('problem2.xls', RB2);
xlswrite('problem3.xls', RB3);
```

```
figure
imshow(RB1/max(max(RB1)));
figure
imshow(RB2/max(max(RB2)));
figure
imshow(RB3/max(max(RB3)));

Point = [10.0000, 18.0000 ;
      34.5000 ,25.0000;
43.5000 ,33.0000;
45.0000 ,75.5000;
48.5000 ,55.5000 ;
50.0000 ,75.5000 ;
56.0000 ,76.5000 ;
65.5000 ,37.0000 ;
79.5000 ,18.0000 ;
98.5000 ,43.5000 ;];

Point = Point - 50;
Point = Point/4*R;
PX = Point(:,1);
PY = Point(:,2);

VALUE2 = interp2(X,Y,IR2,PX,PY,'cubic');
VALUE3 = interp2(X,Y,IR3,PX,PY,'cubic');

VALUE2(VALUE2<0) = 0;
VALUE3(VALUE3<0) = 0;

disp('step5 finishs');
```