白色X線二重露光法による四点曲げ応力の測定

新潟大学・鈴木賢治

平成 30 年 12 月 28 日

1　データファイルの作成

1. cdte2.f90 を利用して CdTe 検出器のデータファイル群 img j.txt から各閾値電圧 (-i mV) フォルダーに測定画像 (Image000 i.txt) を作成
2. CdTe_E.f90 を利用して，実験前後の Pb と W の $K\alpha$ 蛍光X線によるエネルギー較正をして CdTe_E.dat [1]を作成する．
3. CdTe_E.dat のデータを基に，E_image.f90 を利用してX線エネルギー 60 keV から 80 keV の回折画像 i keV.txt のセットを作成し，keV_PbW-Ka フォルダーにまとめる．
4. i keV.txt のファイル群から keV-stack-diff.ijm を利用してスタック像を作成．さらに，メジアン処理，平均によりノイズ処理を行い，差分によりX線エネルギー像 i keV_d.tif を作成し，スタック像 Stack61-79keV-d.tif を作成する．

2　回折を得るための解析方法

スタック像 Stack61-79keV-d.tif を利用して，61 ∼ 79 keV のエネルギーによる回折の変化を観察すると，結晶粒ごとのロッキングカーブをみているような回折斑点の挙動がみられる．このことは，回折する波長と結晶粒の方位が関係しており，それを反映して結晶方位に最適なX線波長の条件の時に，回折斑点強度が最大となる．さらに，回折斑点をよく見ると顕著に移動せず，どちらかというと回折強度が変化している挙動である．

回折角を決定するためには，次のような手続きをすることが考えられる．

1. 各波長エネルギーごとの P1 の回折斑点像から**回折斑点を抽出**し，リストを作成する．
   リストする項目は，各 keV ごとの回折位置 (x, y)，回折強度の要素群．
2. **回折斑点の抽出法をどうするのか．2 階差分か．**
3. リストされた各回折斑点の強度を追跡して，回折強度が最大になるX線エネルギーを決定する．
4. どのようにして，各波長エネルギーにおける**回折斑点の同一性**を判定するか．位相・近さか．

このような問題に取り組むに当たり，効率よくと言う考えは捨てて，ビックデータになることを許容しながら進める必要がある．

[1]実際には PbW_CdTe.dat 名のファイルを処理するようにした．関係するプログラムをカスタマイズした．

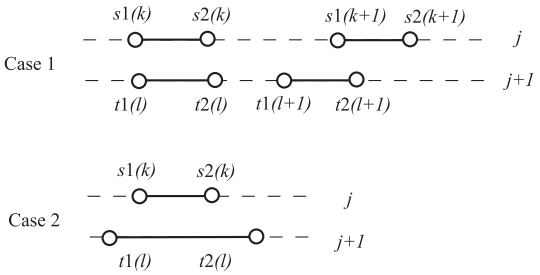


図 1: 掃き出し法

3 斑点の抽出方法

二重露光法では，何をおいても斑点の抽出が重要な意味を持っている．単色X線による二重露光法では二値化した後，輪郭を抽出する方法をとったが，白色X線の二重露光法では多数の回折斑点を抽出して，斑点の変化挙動や最大値と斑点位置を求めることが必要である．それに適した**斑点の抽出**する方法，プログラムを新たに作成する[2]．

1. P1 の *i*keV-d.txt 画像を用意する．
2. *i*keV-d.txt 画像を二値化する．
3. 二値化像を用いて水平 (j) に画素を取り出して輝度の変化する位置 s1 と s2 を取り出し斑点の線分群 (ns 本) を抽出する (図 1).
4. 同様に次の水平 (j+1) 画素を取り出して輝度の変化する位置 t1 と t2 を取り出し斑点の線分群 (nt 本) を抽出する．
5. 各 t1-t2 線分に対して各線分 s1-s2 の対応を比較する．
6. 線分 s1-s2 に属する線分 t1-t2 は，該当する斑点に追加する．
7. 線分 s1-s2 に属しない線分 t1-t2 は，新たな斑点として作成する．

以上を繰り返すことで斑点群を作成できる．この手続きは，図 1 の Case 1 に相当する．さらに，Case 2 についても処理する必要があり，Case1 および 2 に該当しない場合が新しい斑点番号 cnt+1 の生成になる．具体的には，

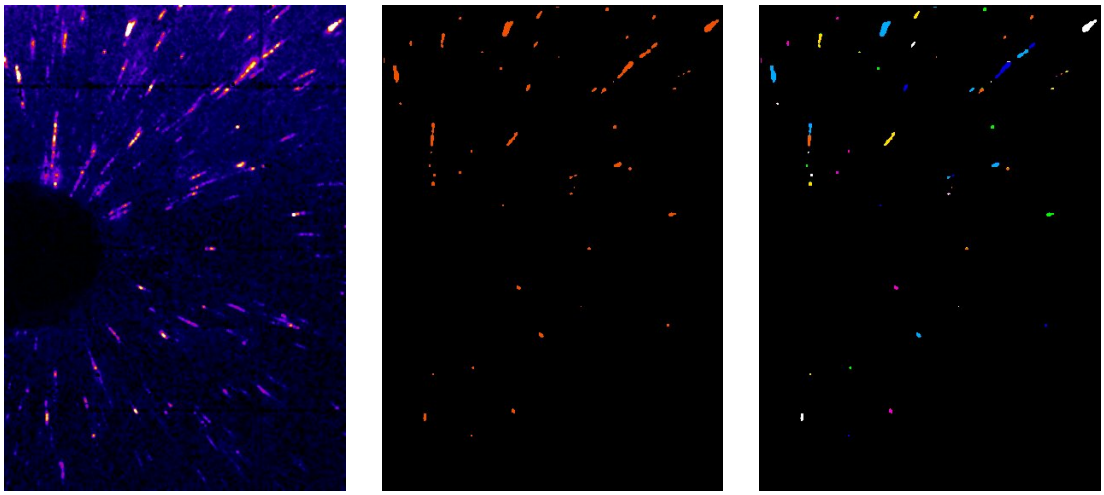
case 1 $(s1(k)$ .le. $t1(l))$ .and. $(t1(l)$ .le. $s2(k))$ または $(s1(k) <= t2(l))$.and.$(t2(l)$ .le. $s2(k))$

case 2 $(t1(l)$ .lt. $s1(k))$.and.$(s1(k)$ .lt. $t2(l))$

の条件に該当する場合は，既設の斑点 (*s*1-*s*2 に接触しているものと判断する．本稿では，これを「掃き出し法」と呼ぶ[3].

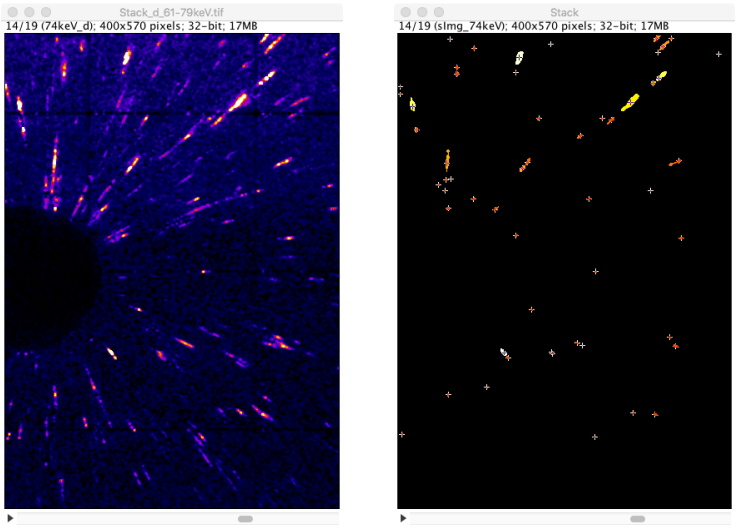
実際に 4 点曲げ実験の $z = 0.2$ mm の 68 keV 波長の像 (68keV_d.txt) を maxima.f90 で処理した例を図 2 に示す．なお，抽出した斑点の区別が付くように，図 2 (c) では斑点を色分けをしている．

[2]そのためのプログラムを spot.f90 として開発する．
[3]断っておくが，この「掃き出し法は」線形代数の掃き出し法とは異なる．



(a) 68 keV 像　　(b) 二値化像　　(c) 斑点抽出像

図 2: 掃きだし法による P1 の回折斑点像から回折斑点の抽出



(a) 斑点像　　(b) 各斑点の最大輝度と位置

図 3: spot.f90 で処理された斑点像の例 ($z = 0.2$ mm 波長エネルギー 74 keV)

さらに，**spot.f90** を作成した．spot.f90 は，X線エネルギー $i_1 \sim i_2$ keV の像 ($i$keV_d.txt) の二値化像 **Ib** i **.txt** を出力し，その画像の斑点群の各最大値と位置を求めて，抽出された斑点の最大輝度 (斑点の色分け)・位置を表した画像 **sImg**_i **keV.txt** を作成する．さらに，spot.f90 は，斑点を抽出し，斑点番号，最大値，位置，斑点画素数および x 座標と y 座標のリストをファイル **sList**_i **keV.dat** としてテキスト出力する[4]．

一例として 74 keV 波長の回折像と斑点を抽出した結果を図 3 に表示する．各斑点の最大輝度と位置を十字で示し，斑点の強度を濃さで表現している．これらの一連の画像のスタック像を作成するために **sImg-stack.ijm** を作成した．

[4] このプログラム spot.f90 は，斑点の抽出を行うので，二重露光法で斑点位置を決定する際にも重要な役割を果たす．斑点の処理のアルゴリズム，出力ファイルが次に利用される．
なお，keV 処理ごとに初期化をしないと残った値が次の処理に影響するので，注意しなければならない．

4　斑点の最大回折強度の探索

前述のように各X線エネルギー像のスタック画像を見ていると，回折斑点は輝き出して消滅する消えて行く挙動を示す．これは，結晶粒ごとに適した回折面方位とそれに適した波長の時に，最大回折を示すことによる．白色X線と CdTe 検出器を利用するメリットは，この点にある．そのため，各斑点ごとに最大回折強度を示すX線エネルギーを探す必要がある．

4.1　グローバル斑点の抽出

1. 各X線エネルギーの斑点群のデータを **sList_** i **keV.dat** から読み出す．
2. 各斑点位置の近さ Δr から同一斑点のマッチングを行う．

$$\Delta r = \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2}$$

これらのデータをどのように指数付けしてグローバル斑点群を作成するのかが課題になる．これらの斑点群のデータベース maxima.dat と逆引き配列・ファイル rev.dat を活用して，グローバル斑点群を作るプログラム maxima.f90 を考える．

1. **グローバル斑点番号 g** を導入する．
   グローバル斑点の**データ数 n(g)**, 斑点 g の列が $i = 1 \sim$ n(g) になり，その中に極大値が含まれる．
2. グローバル斑点 g に対応する**エネルギー値**は **keV(g,i)** で与えられ，そのローカル斑点番号は ***l***(g,i) で与える．
3. **逆引き配列 rev(keV,*l*)** を導入する[5]．逆引き配列は，エネルギー値とローカル斑点番号からグローバル斑点番号 rev(keV,l) =g を与える．もし $g = 0$ ならば，新たなグローバル斑点として，$g + 1$ グローバル斑点を生成する．
4. スタートの sList_*i*keV.dat のファイル番号に従い，rev から判断してグローバル番号 g を付けると，keV++して全 l について近さを計算する．rev で判断して既存のグローバル変数に属しないならば，現在のグローバル変数に属する斑点として，keV(g,i) を行う．
5. Δr を基準に近接斑点 l を決定し， **g, keV(g,i),** l**(g,i)=** l**, top** を **maxima.dat** に保存する．
6. i++を行い，作業を繰り返す．
7. 近接斑点が見つからなければ，call S_plot() は，spot_l=-99 のコードを返して，現在のグローバル斑点の探索を終了する．次の斑点に向かう．
8. 次の斑点は，keV ごとに sList_*i*keV.dat から逐次カウンター ll++しながら，ローカル斑点番号 l を読み出して行く．
9. 読み出したローカル斑点番号 l を配列 rev から確認して，すでに登録 ($\neq 0$) されていれば，何もせずに，次の斑点 ll に移動する．

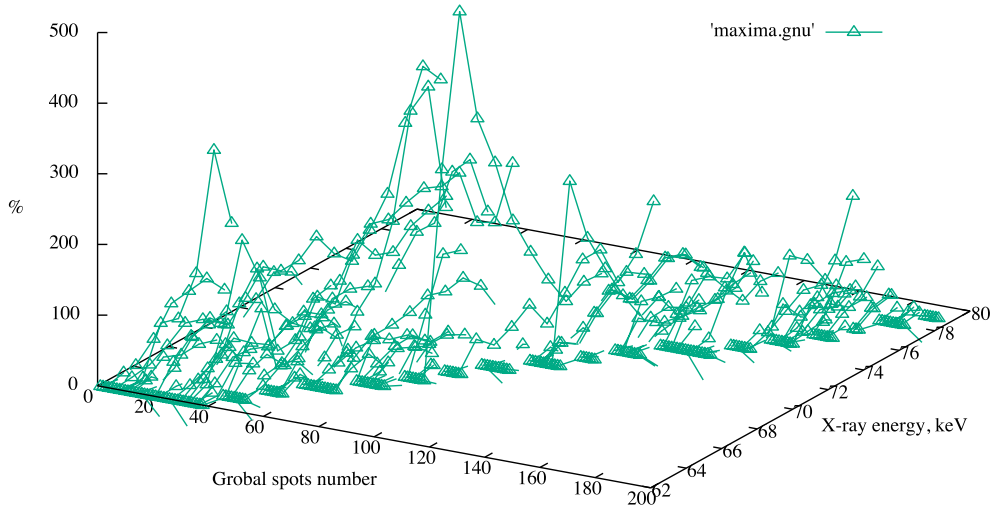


図 4: maxima.f90 により探索したクローバル斑点.

以上の手続きをプログラム maxima.f90 として完成させた．具体的なプログラム内容は，付録に示す．spot.f90 で得られた回折斑点は，各X線エネルギーごとの検出画像における斑点 (ローカル斑点) の抽出であった．maxima.f90 は，これらのデータ (sList_ i keV.dat) を利用して，各斑点のX線エネルギーの変化挙動を追跡する．そして maxima.f90 は，それぞれのグローバル斑点の回折強度が最大になるX線エネルギーを求める．

具体例として，maxima.f90 によって出力されたデータファイル **maxima.gnu** を gnuplot で表示した例を図 4 に示す．なお，gnuplot を立ち上げて，以下のコマンドを入力することで，図 4 のグラフを描くことができる．

```
# gnuplot command list
set ticslevel 0
set zrange [0:500]
set xlabel " Grobal spots number "
set ylabel " X-ray energy, keV "
set zlabel " % "
set points 1
splot 'maxima.gnu' with linespoints pt 6 lc 2
```

図 4 の maxima.gnu は表示は，各斑点の輝度で正規化して表示しているので，z 軸は%表示となっている．各グローバル斑点のピークのX線エネルギー (keV) との関係をみると，図のように最大を示した後に減衰する様子がわかる．各斑点のピークから，X線エネルギー (keV) とそのローカル斑点番号を知ることができるので，二重露光法に利用する斑点が同定できる．

4.2 まとめ

前述のシステムのしくみをまとめるために，図 5 を作成した．spot.f90 は，

1. 各エネルギー回折像 i keV_ d.txt を読み込む．

[5]逆引き配列・ファイルを利用することで，グローバル斑点の作成が効率的になっている．

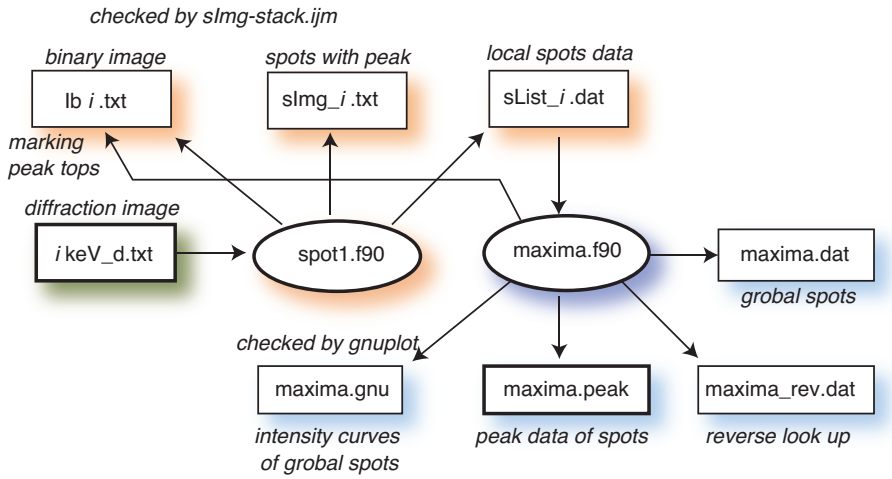


図 5: 二重露光法に用いる回折斑点の探索システム (その 1).

2. メジアン処理，平滑処理をして二値化像 Ib i .txt を出力.
3. ローカル斑点とその画素群の画像 sImg _ i .txt を出力.
4. ローカル斑点とその画素群のデータファイル sList _ i .txt を出力.

の処理を行う.

さらに，maxima.f90 は，

1. ローカル斑点とその画素群のデータファイル sList _ i .txt を読み込む.
2. エネルギー範囲と近傍のデータを入力し，グローバル斑点のデータ maxima.dat を出力する.
3. エネルギーとローカル斑点からグローバル斑点を得る逆引きファイル maxima _ rev.dat を出力する.
4. maxima.dat からグローバル斑点の挙動を gnuplot で見せるためのデータファイル maxima.gnu を出力する.
5. 各グローバル斑点のＸ線エネルギーとローカル斑点情報を記録したファイル maxima.peak を出力する.
6. maxima.peak の決かを確認するために，二重露光法に用いるピークトップを Ib i .text に十文字をマークする．この結果については，sImg-stack.ijm のマクロを用いて ImagJ でスタック像を作成し，確認することができる.

5　回折角の決定

5.1　P1 の回折斑点位置の決定方法

まず，P1 の回折斑点の位置を決定するために，プログラム **P1 _ spot.f90** を作成する．その具体的な手続きは，

1. maxima.peak のデータに基づいて，各グローバル斑点のデータを読み出す.

2. 当該ピークトップの画像 $i$keV_d.txt を読み出す．
3. ピーク強度の半価値 h_{05} で二値化して，当該斑点の重みで回折位置 (xt, yt) を決定する．
   この処理には，subroutine Trace() を改良して活用する[6]．

   (a) 回折斑点の最大位置から左に移動して二値画像の境界にぶつかった所から，右回りで境界をなぞる．
   (b) 境界の方向ベクトルを判断しながら斑点の輪郭位置を bn(k,i) に出力する．k は境界番号，$i = 1, 2$ は x, y の座標
   (c) スタートライに戻れば完了．配列 Ib(i,j) の二値画像に輪郭のデータが含まれている．

図 6: P1 _ spot.f90 で作成された斑点処理像 (Ib _ trace.txt)

4. 重み付きによる回折斑点位置を計算する．

   (a) 各斑点の縦と横のサイズ i_max, i_min, j_max, j_min を求める．
   (b) i_max, i_min, j_max, j_min の範囲で i, j を++しながら座標 (i,j) が斑点に含まれるかを調べる[7]．
   (c) 座標 (i,j) から左右上下に臨んだときに，その延長線上に境界データが 4 方向ともに存在するときのみ内点と判定する．
   (d) 内点と判定された座標 (i,j) を使い，重み付き重心 (Gx,Gy) を計算する．

以上の処理を P1 について実行して，一括リスト **P1_ spot.dat** を作成する．**P1_ spot.dat** は，
斑点数グローバル斑点番号
波長 (keV)，ローカル斑点番号，回折位置 (x, y)，回折強度

[6]Trace は DEM にて開発されたコードである．
[7]当初，Green の定理も検討したが，明確に判定できる方法を撮ることにした

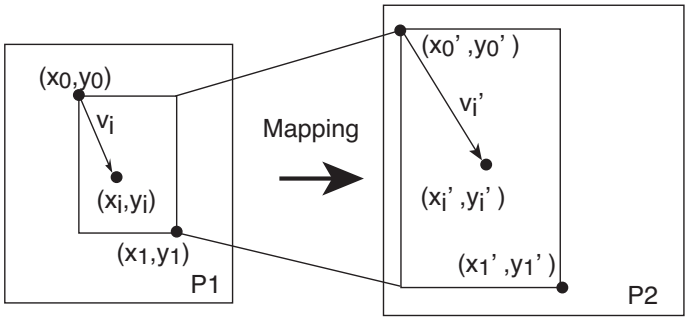


図 7: P1 から P2 への写像.

…

が記録されている.

P1 _ spot.f90 で作成された斑点処理像 (Ib _ trace.txt) の例を図 6 に示す．最大強度の半価値で二値化した斑点の輪郭と重み付き重心が示されている.

6 P1 から P2 への写像

6.1 P1 の画素位置の変換式

二重露光法においては，P1 および P2 の写像関係が重要となる．P1 と P2 の各点のおおよその対応関係を精度よく得ておく必要がある．#SF6 曲げ試験片においては，70 keV 相当のX線エネルギーを利用することから，70 keV 回折像を利用して，写像関係を求める.

P1 から P2 への回折斑点の写像は，完全に対応した画像ではないので，P1 の一部の領域が P2 の一部の領域と一致している関係にある．ゆえに，図 7 のようにな状態になっている．具体的に複数の斑点のグループを取り出して見ると，白色X線を CdTe 検出器で測定したときの P1 と P2 の像では，ノイズが多いこと，斑点の輝度が完全に対応していない斑点も多い．つまり，P1 では強い輝度なのに，P2 で輝度が弱い斑点もあった．その逆もあった．そのようなことから，代表的な斑点数点から予測することは避け，精密に画像をマッチングして，写像関係を総合的に得る方法を検討する．マッチング方法としては SSD (sum of squared difference) を利用する.

具体的には，70 keV の P1 および P2 の画像を利用した．P1 の画像については，$(x_0, y_0) = (200, 150)$ および $(x_1, y_1) = (400, 455)$ を利用してテンプレート画像 $T_p(k, l)$ とした．画像の写像関係は，拡大率を M_g とすると

$$x_i' = M_g\ (x_i - x_0) + x'0 \tag{1}$$

$$y_i' = M_g\ (y_i - y_0) + y'0 \tag{2}$$

で与えられる．P2 とマッチングするためのテンプレート画像 $T_p(k, l)$ の要素の値は P1 の画像の要

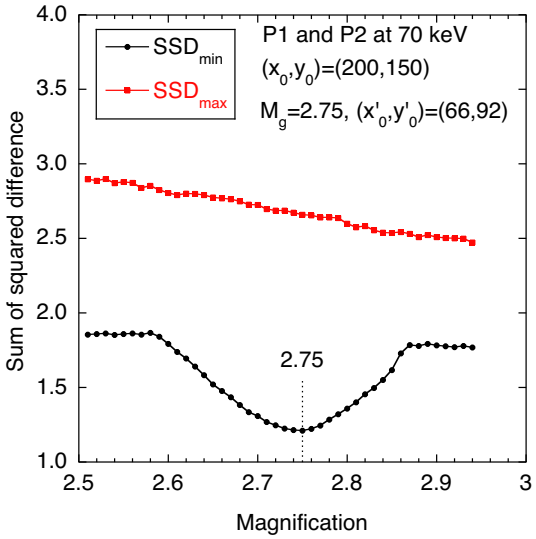


図 8: SSD による P1 から P2 への写像の最適パラメータの決定.

素 $P_1(i,j)$ から次式により与えた.

$$i = \frac{k}{M_g} + x_0 \tag{3}$$

$$j = \frac{l}{M_g} + y_0 \tag{4}$$

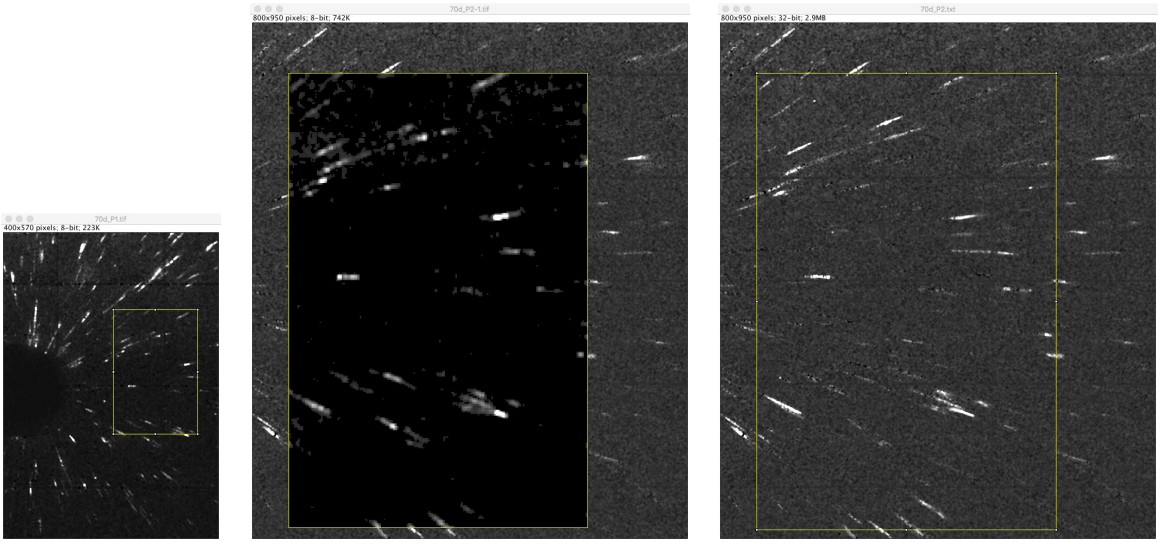
拡大率 M_g を決定した後，P2 のテンプレート $T_p(k,l)$ の原点 (x'_0, y'_0) を逐次変化させながら SSD を計算し，誤差の少ない M_g と (x'_0, y'_0) の組み合わせを求めることとした．なお，斑点の一致が明瞭な誤差計算のをするために，バックグランドを処理するために P1 および P2 において，それぞれ 40 および 80 を閾値として SSD を計算した．SSD の値は，テンプレートの画素の輝度と P2 の画素の輝度の差を自乗した総和である．テンプレートが拡大されるために SSD が大きくなるのを補正するために，計算した SSD をテンプレート画像の輝度の自乗和で除して正規化している．

P1 の領域 $(x_0, y_0) = (200, 150)$-$(x_1, y_1) = (400, 455)$ を逐次拡大しながら，P2 上で (x'_0, y'_0) を変化させて最小の $\mathrm{SSD_max}$ を計算した結果を図 8 に示す．この結果から拡大率 $M_g = 2.75$, $(x_0, y_0) = (66, 92)$ を最適地として得た．

得られた最適パラメータのときの画像を比較したものが図 9 である．P1 は 2×3 で構成され，P2 は，4×5 で構成されているので，その比率で表示している．図 (a) と (c) を比較すると P1 の画像の斑点と P2 の斑点の原画像が完全に一致していないことがわかる．P1 のテンプレートを図 (a) に四角の枠で示した．それに対応する枠を図 (c) にも表示している．その図 (c) の枠に拡大されたテンプレートを埋め込んだのが，図 (b) である．これらの結果から，最適な写像関係が得られていると判断した．

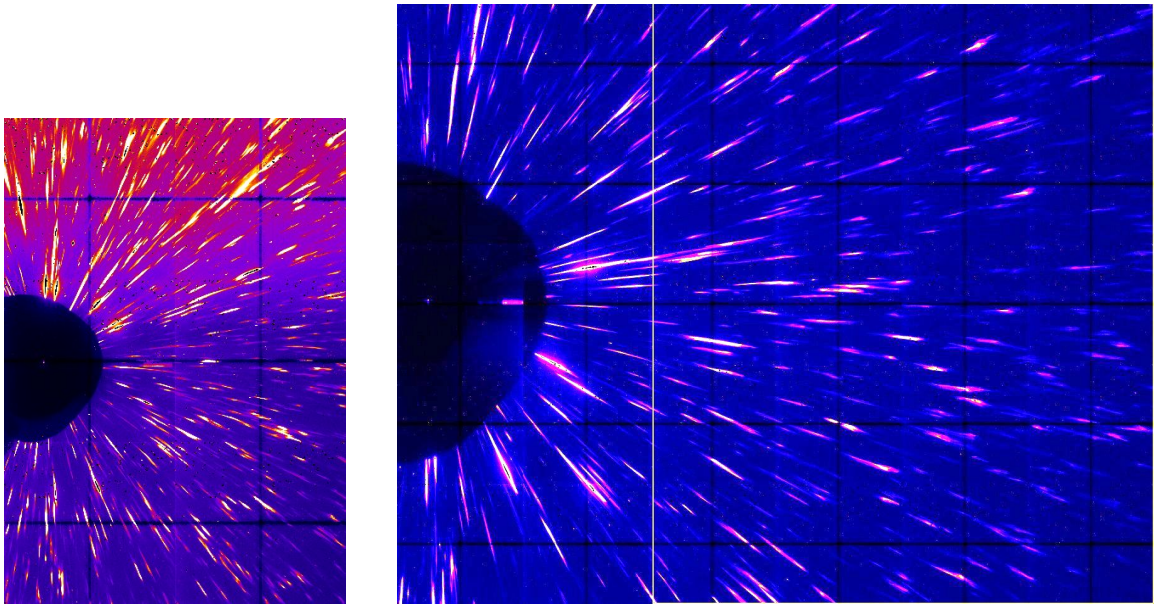
以上のことから，**P1 の画素位置の変換式**は，

$$\left.\begin{aligned} k &= 2.75\,(i - 200) + 66 \\ l &= 2.75\,(j - 150) + 92 \end{aligned}\right\} \tag{5}$$



(a) P1 像　　(b) マッチング像　　(c) P2 像

図 9: 最適パラメータでマッチングした結果 (70keV).



(a) P1 ビーム中心 (2×3)　　(b) P2 ビーム中心 (6×5)

図 10: P1 および P2 のビーム中心 (閾値電圧 -100 mV).

で表され，P1 の (i, j) は P2 の (k, l) に変換される．ただし，この写像はあくまで同一中心から投影されていると仮定している．実際は，粗大粒ごとの焦点から回折斑点が投影されるので，P1 と P2 が完全に一致しないことに注意する必要がある．この変換は，P1 上の斑点が P2 のどの位置付近に投影されるかの目安でしかない．

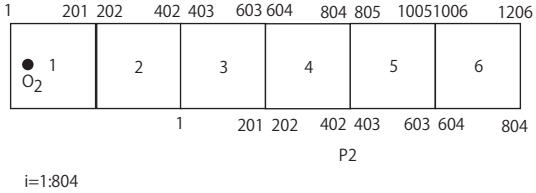


図 11: P2 の画像とビームセンター画像

6.2 実験室座標系と P1 と P2 の関係

閾値 −100 mV のX線のビームセンター像を図 10 に示す．図 (a) の P1 のビームセンターは P1 で測定した画像である．図 (b) に示す P2 の位置の画像は，本来は 4×5 であるが，ビームセンターも含めて測定するために，敢えて 6×5 の画像で測定した．ビームストッパーの丸い影の中に，微小な白い点が見え，それがビームセンターである．

P1 と P2 のビームセンター関係するデータをまとめると．

- P1 の中心画像は $O_1 = (x_1^0, y_1^0) = (46, 288)$ である[8]．
- P2 の中心画像は $O_2 = (x_2^0, y_2^0) = (49, 471)$ である[9]．
- P1-P2 の移動距離 $L = 500$ mm

のデータが得られた．

以上のことから，測定した P1 の任意の点 $P1(i) = (x_1^i, y_1^i)$ は，実験室座標系 (x, y, z) とすると[10]

$$x = (x_1^i - 46) \times 0.2\,, \quad y = (288 - y_1^i) \times 0.2\,, \quad z = 0.0 \quad [\mathrm{mm}] \tag{6}$$

で求められる．この場合，P2 のビームセンターを原点とし，X線光軸が z 軸になり下流が正になる．y 軸は下方向が正になる．x 軸は水平線あり，ホール側を正に定義する．

さて，P2 については，やや複雑となる．図 11 のように，ビームセンター画像と x 方向の画素のずれが生じているので，402 画素分を加える必要がある．その結果，測定した P2 の任意の点 $P2(i) = (x_2^i, y_2^i)$ は，絶対座標系 (x, y, z) とすると

$$x = (x_2^i + 402 - 49) \times 0.2\,, \quad y = (471 - y_2^i) \times 0.2\,, \quad z = 500.0 \quad [\mathrm{mm}] \tag{7}$$

で与えられる．

7 P2 の斑点処理

7.1 P2 の斑点位置の計算

P1, P2 の斑点処理システムのファイルの構成を図 12 に示す．

[8] −100 mV 像のスポット像から確認した．

[9] beam center 像の −100 mV 像のスポット像から確認した．

[10] 配列は 1 から始まるようにしていることに注意すること．プログラムの斑点位置は，すべて画素の座標であことから，座標系の計算では $x_i = i - 1$ で計算しなければならない．

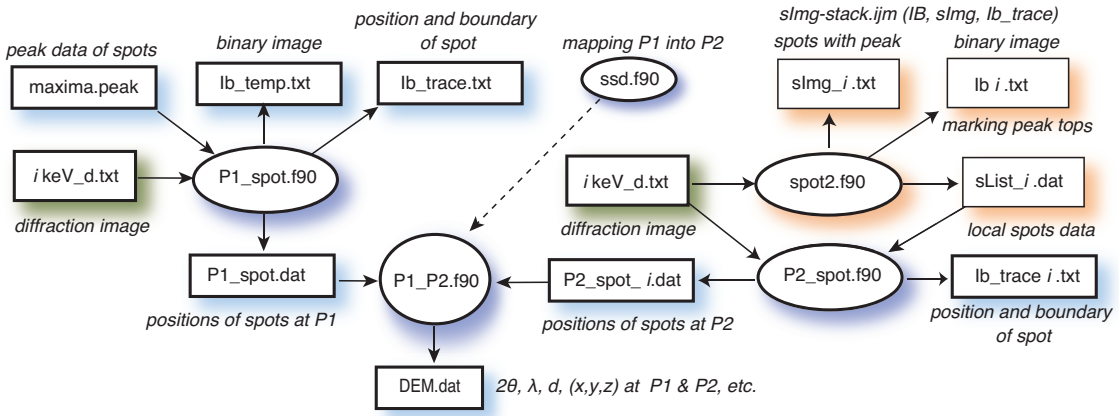


図 12: 二重露光法のシステム (その 2).

1. P2 の斑点抽出プログラム **spot2.f90** を作成．spot2.f90 により，斑点のリスト sList _ *i* keV.dat および画像 sImg _ *i* keV.txt が作成される．
2. **P2_spot.f90** を作成し，各斑点最大値の 7 割の高さで閾値を決定して境界を見つけて，重み付き重心にて，P2 の斑点の位置を計算する．その回折強度は斑点領域の重み付き平均である．
3. **P2_spot.f90** は，*i* keV ごとに，斑点番号，回折斑点の位置および強度を **spotP2_ *i*.dat** に出力する．処理結果の画像を確認するために **Ib _ trace *i*.txt** が作成される．

7.2 白色Ｘ線二重露光法の計算

1. P1 の最大回折斑点データ **P1 _ list.dat** から各斑点の keV と斑点位置 $P_1' = (x_i, y_i)$ を読込み，ssd.f90 で得られた写像関係から P2 の keV と斑点位置 $P_2' = (x_i', y_i')$ を決定する．
2. 該当する keV の spotP2 _ *i*.dat フアイルを開き，$P_2' = (x_i', y_i')$ に近い位置 (x_i', y_i') を 3 点の斑点候補をリストアップする．
3. リストアップされた結果から，回折位置 $P_1 = (x_1, y_1, z_1)$, $P_2 = (x_2, y_2, z_2)$, 波長 λ, 回折角 2θ, 格子面間隔 d を計算する．
4. 最適な回折位置の決定に対して，距離が支配的か，回折強度支配的かを比較・検討する．

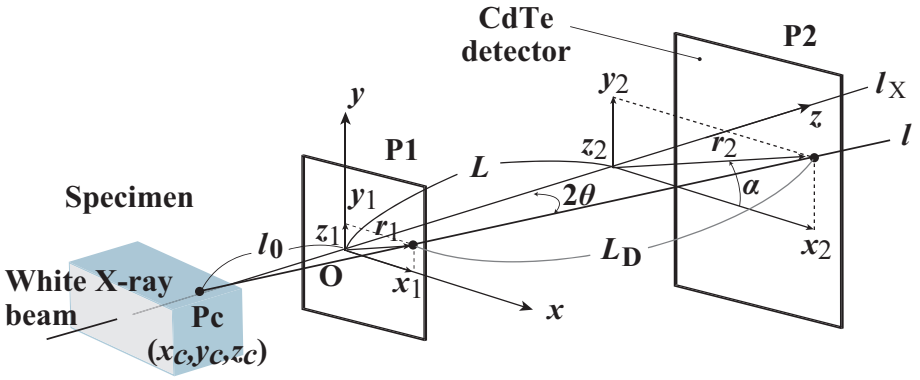
7.3 二重露光法

本研究における二重露光法の実験室座標系を図 13 (a) に示す．回折斑点の半径 r_1, r_2 は，

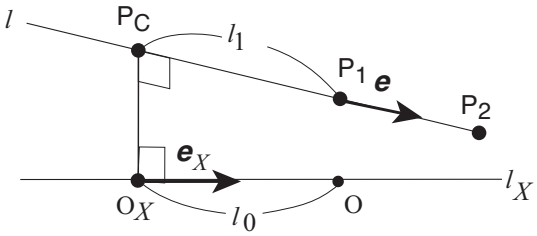
$$r_1 = \sqrt{x_1^2 + y_1^2}, \quad r_2 = \sqrt{x_2^2 + y_2^2} \tag{8}$$

で与えられ，回折角 2θ は，

$$2\theta = \arctan\left(\frac{r_2 - r_1}{z_2 - z_1}\right) = \arctan\left(\frac{r_2 - r_1}{L}\right) \tag{9}$$



(a) P1 および P2 で斑点を検出



(b) 斑点の回折ビーム ℓ とX線ビーム ℓ_X との交差

図 13: 二重露光法の実験室座標系.

X線ビームの直線を ℓ_X，回折ビームの直線を ℓ として，それぞれの単位ベクトル $\boldsymbol{e}$ および $\boldsymbol{e}_X$ は，

$$\boldsymbol{e}_X = (0,0,1)\,, \quad \boldsymbol{e} = \frac{(x_2-x_1, y_2-y_1, z_2-z_1)}{|L_D|} \tag{10}$$

で表される．図 13 (b) に示すように直線 ℓ_X と ℓ が交差する所では，線分 $\overline{P_C O_X}$ は，直線 ℓ_X と ℓ に直交する．ゆえに，ベクトル $\overrightarrow{P_C O_X}$ と ℓ_X，ℓ の内積が 0 になる．ベクトルの関係を表示すると，

$$\overrightarrow{OO_X} = -\ell_0\,\boldsymbol{e}_X \tag{11}$$

$$\overrightarrow{OP_C} = \overrightarrow{OP_1} - \ell_1\,\boldsymbol{e} \tag{12}$$

の関係から，ベクトル $\overrightarrow{P_C O_X}$ は，

$$\overrightarrow{P_C O_X} = \overrightarrow{OO_X} - \overrightarrow{OP_C} = -\ell_0\,\boldsymbol{e}_X - \overrightarrow{OP_1} + \ell_1\,\boldsymbol{e} \tag{13}$$

となる．直線 ℓ と $\overrightarrow{P_C O_X}$ の内積が 0 になるので，

$$\boldsymbol{e} \cdot \overrightarrow{P_C O_X} = 0 \tag{14}$$

具体的な計算は以下となる．

$$\begin{aligned}\boldsymbol{e}\cdot\overrightarrow{P_C O_X} =& \boldsymbol{e}\cdot(-\ell_0\,\boldsymbol{e}_X-\overrightarrow{OP_1}+\ell_1\,\boldsymbol{e})\\ =&-\ell_0\,\boldsymbol{e}\cdot\boldsymbol{e}_X-\boldsymbol{e}\cdot\overrightarrow{OP_1}+\ell_1\\ =&-\frac{z_2-z_1}{L_D}\,\ell_0-\frac{(x_2-x_1)\,x_1+(y_2-y_1)\,y_1+(z_2-z_1)\,z_1}{L_D}+\ell_1\\ =&0\end{aligned}$$

ゆえに，

$$\ell_1=\frac{(x_2-x_1)\,x_1+(y_2-y_1)\,y_1+(z_2-z_1)\,z_1}{L_D}+\frac{z_2-z_1}{L_D}\,\ell_0 \tag{15}$$

が導かれる．同様に，直線 ℓ_X と $\overrightarrow{P_C O_X}$ の内積が 0 になるので，

$$\boldsymbol{e}_X\cdot\overrightarrow{P_C O_X}=\boldsymbol{e}_X\cdot(-\ell_0\,\boldsymbol{e}_X-\overrightarrow{OP_1}+\ell_1\,\boldsymbol{e})=0 \tag{16}$$

具体的な計算は，

$$\begin{aligned}\boldsymbol{e}_X\cdot\overrightarrow{P_C O_X}&=\boldsymbol{e}_X\cdot(-\ell_0\,\boldsymbol{e}_X-\overrightarrow{OP_1}+\ell_1\,\boldsymbol{e})\\ &=-\ell_0-\boldsymbol{e}_X\cdot\overrightarrow{OP_1}+\ell_1\,\boldsymbol{e}_X\cdot\boldsymbol{e}\\ &=-\ell_0-z_1+\frac{(z_2-z_1)}{L_D}\,\ell_1\\ &=0\end{aligned}$$

となる．ゆえに，

$$\ell_0=\frac{(z_2-z_1)}{L_D}\,\ell_1-z_1 \tag{17}$$

を得る．ここで，以下の幾何学関係をまとめると，

$$L_D=\frac{L}{\cos 2\,\theta}\,,\quad z_2-z_1=L\,,\quad z_1=0 \tag{18}$$

となるので，式 (15), (17) を書き直すと

$$\ell_1=\frac{\cos 2\theta}{L}\left[(x_2-x_1)\,x_1+(y_2-y_1)\,y_1\right]+\ell_0\,\cos 2\theta \tag{19}$$

$$\ell_0=\ell_1\,\cos 2\theta \tag{20}$$

なるので，式 (19), (20) の連立方程式を解いて，

$$\ell_1=\frac{\cos 2\theta}{L\,\sin^2 2\theta}\left[(x_2-x_1)\,x_1+(y_2-y_1)\,y_1\right] \tag{21}$$

$$\ell_0=\frac{\cot^2 2\theta}{L}\left[(x_2-x_1)\,x_1+(y_2-y_1)\,y_1\right] \tag{22}$$

が得られる．

以上の計算から，２直線 ℓ および ℓ_X の交差する位置 $P_C = (X_C, y_C, z_C)$ は，

$$\overrightarrow{OP_C} = \overrightarrow{OP_1} - \ell_1 \boldsymbol{e} = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} - \ell_1 \frac{\cos 2\theta}{L} \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ z_2 - z_1 \end{pmatrix} \tag{23}$$

となるので，

$$\begin{pmatrix} x_C \\ y_C \\ z_C \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \\ 0 \end{pmatrix} - \frac{\cot^2 2\theta}{L^2} \Big[(x_2 - x_1)\, x_1 + (y_2 - y_1)\, y_1 \Big] \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \\ L \end{pmatrix}$$

以上の結果から二重露光法の回折ビームとＸ線ビームの交差位置 P_C の座標位置

$$x_C = x_1 - \frac{\cot^2 2\theta}{L^2} \Big[(x_2 - x_1)\, x_1 + (y_2 - y_1)\, y_1 \Big] (x_2 - x_1) \tag{24}$$

$$y_C = y_1 - \frac{\cot^2 2\theta}{L^2} \Big[(x_2 - x_1)\, x_1 + (y_2 - y_1)\, y_1 \Big] (y_2 - y_1) \tag{25}$$

$$z_C = -\frac{\cot^2 2\theta}{L} \Big[(x_2 - x_1)\, x_1 + (y_2 - y_1)\, y_1 \Big] \tag{26}$$

を得る．回折角度 2θ については，式 (9) から得る．なお，単色Ｘ線による二重露光法の経験では，得られた 2θ の評価は，P_C の座標位置から判定することが最も合理的であった．また，回折線の方位角 φ は，

$$\alpha = \arctan\left(\frac{y_1}{x_1}\right) \tag{27}$$

から計算する．

8　応力ひずみ関係

各斑点ごとに波長，回折角などの要因で回折の角度関係が変化するので，測定した格子面間隔 $d_{\varphi\psi}$ は格子定数 $a_{\varphi\psi}$ に変換し，さらに，曲げ応力負荷方向，すなわち図 14 の ε_1 方向で評価する必要がある．

透過Ｘ線が ε_3 方向に入射すると考えると図 14 に示すようになる．これは，図 13 の z 軸が ε_3 方向に一致する．このときの方位角の関係は，以下のようになる．

$$L_D = \frac{L}{\cos 2\theta}\,, \quad r = \frac{L_D}{2} \sin 2\theta = \frac{L}{2} \tan 2\theta$$

$$\Delta x = \frac{L}{2} \cos\alpha \, \tan 2\theta\,, \quad \Delta y = \frac{L}{2} \sin\alpha \, \tan 2\theta$$

$$\tan\varphi = \frac{\Delta x}{L} = \frac{1}{2} \cos\alpha \, \tan 2\theta$$

この結果，方位角 φ については，

$$\varphi = \arctan\left(\frac{\cos\alpha \, \tan 2\theta}{2}\right) \tag{28}$$

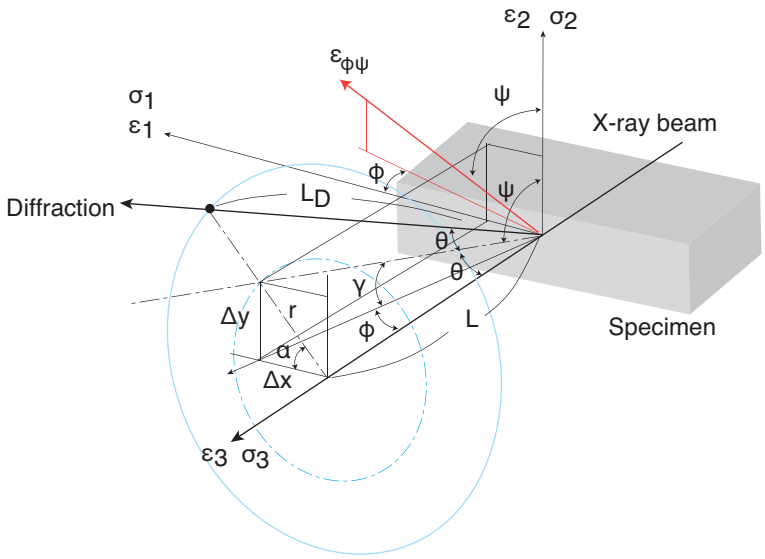


図 14: 応力-ひずみ関係

となる．方位角 γ については，

$$\cos\gamma = \frac{L/\cos\varphi}{L/\cos\theta} = \frac{\cos\theta}{\cos\varphi} \tag{29}$$

から

$$\gamma = \arccos\left(\frac{\cos\theta}{\cos\varphi}\right) \tag{30}$$

が得られる．

以上のことから，回折斑点により測定する格子面法線の方位角 φ および ψ は以下のようになる．

$$\varphi = \arctan\left(\frac{\cos\alpha\ \tan 2\theta}{2}\right) \tag{31}$$

$$\psi = \frac{\pi}{2} - \gamma = \frac{\pi}{2} - \arccos\left(\frac{\cos\theta}{\cos\varphi}\right) \tag{32}$$

測定する格子面間隔の方向角 φ および ψ が得られたので，次に応力とひずみについて検討する．図 14 に示すように各方向の主ひずみを $\varepsilon_1, \varepsilon_2, \varepsilon_3$，主応力を $\sigma_1, \sigma_2, \sigma_3$ とする．まず平面応力 (σ_3) を仮定して，測定方向のひずみ $\varepsilon_{\varphi\psi}$ と主ひずみとの関係は，

$$\varepsilon_{\varphi\psi} = \left(\varepsilon_1 \cos^2\varphi + \varepsilon_2 \sin^2\varphi\right) \sin^2\psi + \varepsilon_3 \cos^2\psi \tag{33}$$

で与えられる．4 点曲げのスパンの間は単純曲げに相当するので，$\sigma_2 = \sigma_3 = 0$ より，応力-ひずみ関係は

$$\varepsilon_1 = \frac{1}{E}\sigma_1\ ,\quad \varepsilon_2 = \varepsilon_3 = -\frac{\nu}{E}\sigma_1 \tag{34}$$

となる．ただし，E はヤング率，ν はポアソン比である[11]．ゆえに，

$$\varepsilon_2 = \varepsilon_3 = -\nu\,\varepsilon_1 \tag{35}$$

[11] X線ひずみを考えるときは，回折面 (hkl) の依存性を考慮する必要がある．Kröner model を利用すること．

となり，式 (33) は，

$$\begin{aligned}\varepsilon_{\varphi\psi} &= \left(\varepsilon_1 \cos^2\varphi - \nu\,\varepsilon_1 \sin^2\varphi\right) \sin^2\psi - \nu\,\varepsilon_1 \cos^2\psi \\ &= \left[\left(\cos^2\varphi - \nu \sin^2\varphi\right) \sin^2\psi - \nu \cos^2\psi\right] \varepsilon_1 \end{aligned} \tag{36}$$

から，次の式が導かれる．

$$\varepsilon_1 = \frac{1}{\left(\cos^2\varphi - \nu \sin^2\varphi\right) \sin^2\psi - \nu \cos^2\psi}\,\varepsilon_{\varphi\psi} \tag{37}$$

測定した格子定数 $a_{\varphi\psi}$ の関係とひずみの関係は，無ひずみの格子定数を a_0 として

$$\varepsilon_{\varphi\psi} = \frac{a_{\varphi\psi}}{a_0} - 1\,,\ \ \varepsilon_1 = \frac{a_1}{a_0} - 1 \tag{38}$$

より，

$$a_1 = \frac{1}{\left(\cos^2\varphi - \nu \sin^2\varphi\right) \sin^2\psi - \nu \cos^2\psi}\ \left(a_{\varphi\psi} - a_0\right) + a_0 \tag{39}$$

が得られる[12]

[12] $\varphi = 0, \psi = \pi/2$ のとき，$a_1 = a_{\varphi\psi}$ となる．その他，1, 2 方向を与えると $-1/\nu$ が得られるので，正しい．

付　　録

spot1.f90

spot.f90 は，回折斑点を抽出するプログラムである．

- **Ib** *i* **.txt**: 差分により作成したＸ線波長エネルギー *i* keV 像をメジアン，平滑化処理して二値化した像
- **sImg**_*i* **keV.txt**: さらに，斑点を抽出し，最大輝度の位置と輝度 (斑点の色分け) で示した画像
- **sList**_*i* **keV.dat**: 斑点を抽出し，斑点番号，最大値，位置，斑点画素数および x 座標と y 座標のリストをテキスト出力している．

```
!*********************************************************
!     search spots and maxima in i keV_d.txt
!     P1=400 x 570
!   spot1.f90         by Kenji Suzuki, Niigata University
!************************************************** December 20, 2018

program spot1

  character*30 ::a,filename
  real :: b(1:400,1:570),c(1:400,1:570)
  integer :: i,j,k,l,m,n,d
  integer :: Ib(1:400,1:570)   !binery image
  integer :: keV1,keV2,keV
  real :: thr,a1, a2,a3,a4,df2,s

print *,'start and stop keV= keV1,keV2'
read(5,*) keV1,keV2

do keV=keV1, keV2,1

!  画像ファイルの読み込み
!  filename='68keV_d.txt'
 write (filename,*) keV     ! cfile に 61 を用意
 filename=trim(adjustl(filename))//'keV_d.txt' !左詰にして，keV_d.txt を加える.
 print *, filename
  open(11,file=filename,status='old')
    do j=1,570
      read(11,*) (c(i,j),i=1,400,1)
    end do
  close(11)

!------------------ 移動平均 m にて平均距離を決定 9 点移動平均 --------------------
  m=1
  do j=3,568
   do i=3,398
     s=0.0
     do l=j-m,j+m
        do k=i-m,i+m
          s=s+c(k,l)
        end do
      end do
      b(i,j)=s/float((2*m+1)**2)
   end do
```

```
  end do
! P1 の周囲に斑点が含まれていると，斑点が閉じないので枠を黒=b で処理する
 do i=1,400
  b(i,1)=0.0; b(i,570)=0.0
 end do
 do i=1,570
  b(1,i)=0.0; b(400,i)=0.0
 end do

!--------------------------- 二値化 -----------------------------
 thr=200.0
 do j=1,570
   do i=1,400
     if (b(i,j) .gt. thr) then
       Ib(i,j)=128
      else
        Ib(i,j)=0
     end if
   end do
 end do

!------------------- 二値画像 Ib をファイル Ib.txt で出力 ----------------
 write (filename,*) keV     ! cfile に 61 を用意
 filename=trim('Ib'//adjustl(filename))//'.txt' !左詰にして，keV_d.txt を加える.
 open (unit=12, file=filename, status='unknown')
 do j=1,570
  write(12,*) (Ib(i,j),i=1,400,1)
 end do
 close(12)
print *,'Go into',keV
 call spots(Ib,b,keV) !二値画像から斑点を探し出す.
print *,'Return from',keV

end do
end program spot1

!************************* SUBROUTINE ****************************
! 斑点を認識するサブルーチン spot は，機織の横糸のようにして斑点の画素データを作成している.
subroutine spots(ib,c,keV)
  character*30 :: filename
  integer :: keV

  real :: c(1:400,1:570) ! CdTe-image
  integer :: ib(1:400,1:570),w,b   !binery image
  integer :: sp(1:400,1:570)=0 ! spot image
  integer :: x(1:200,1:300)=0, y(1:200,1:300)=0, sn(1:200)=0,cnt !spot no, x,y
  integer :: s1(1:10)=0,s2(1:10)=0,t1(1:10)=0,t2(1:10)=0,ns, nt, i,j,k,l
  integer :: m,n, ni,tk,mx,FRG,ntt,tt1(1:10)=0,tt2(1:10)=0,m_old
  integer :: tx(1:200),ty(1:200),i0,j0 !斑点の最大値と位置 (tx,ty)
  real :: tmax,top(1:200)

! 【重要】初期化　繰り返し call されるので，変数とパラメータを初期化しないとエラーが起きる.
  w=128
  b=0
  cnt=0

 do j=1,570
  do i=1,400
   sp(i,j)=0
```

```
  end do
 end do
 do j=1,200
  tx(j)=0; ty(j)=0; sn(j)=0
  do i=1,300
   x(j,i)=0; y(j,i)=0
  end do
 end do

!      掃き出し sweep mode

 do j=1,568

   do k=1, 10
   s1(k)=0; s2(k)=0; t1(k)=0; t2(k)=0   !線分の初期化
  end do

!                     j 行目のライン設定 s1, s2
  k=0    ! line 1
    do i=1,399
      if ((ib(i,j).eq.b).and.(ib(i+1,j).eq.w)) then
       k=k+1
       s1(k)=i+1
      end if
      if ((ib(i,j).eq.w).and.(ib(i+1,j).eq.b)) then
       s2(k)=i
      end if
    end do
    ns=k !Number of line segments

!                     j+1 行目のライン設定 s1, s2
    k=0    ! line 2
    do i=1,399
      if ((ib(i,j+1).eq.b).and.(ib(i+1,j+1).eq.w)) then
       k=k+1
       t1(k)=i+1
      end if
      if ((ib(i,j+1).eq.w).and.(ib(i+1,j+1).eq.b)) then
       t2(k)=i
      end if
    end do
    nt=k !Number of line segments
!*                     j+2 行目のライン設定 tt1, tt2   上の斑点が下で結合している場合
    k=0    ! line 2
    do i=1,400-1
      if ((ib(i,j+2).eq.b).and.(ib(i+1,j+2).eq.w)) then
       k=k+1
       tt1(k)=i+1
      end if
      if ((ib(i,j+2).eq.w).and.(ib(i+1,j+2).eq.b)) then
       tt2(k)=i
      end if
    end do
    ntt=k !Number of line segments

!ここまでは perfect

!                     セグメント s1,s2 と t1,t2 の接触を調べて斑点の追加または生成を行う
   do l=1,nt
```

```
    tk=0
    do k=1,ns
    !t1 または t2 が斑点 k(s1,s2) を探す．該当線分 (tk) がない場合 (tk=0) は，新規作成
     if (((s1(k) <= t1(l)).and.(t1(l) <= s2(k))).or.((s1(k) <= t2(l)).and.(t2(l) <= s2(k)))) then
      tk=k
     end if
     if ((t1(l).lt.s1(k)).and.(s1(k).lt.t2(l)).and.(s2(k).lt.t2(l))) then
      tk=k
     end if
    end do !繰り返し k の end
!*  j+2 で 線分 t1-t2 を共有している場合 cnt++をさせない．FRG=3
    do k=1,ntt
     if (((tt1(k).le.t1(l)).and.(t1(l).le.tt2(k))).and.((tt1(k).le.t2(l-1)).and.(t2(l-1).le.tt2(k))))
      tk=cnt; FRG=3
     end if
    end do

    if (tk.eq.0) then ! tk=0: t1, t2 が斑点 (s1,s2) に含まれない，新しい斑点の生成
!         新しい斑点の生成，cnt:斑点番号，sn(cnt):斑点の画素数，(x,y):画素位置
          cnt=cnt+1
           do i=t1(l),t2(l)
            sn(cnt)=sn(cnt)+1
            x(cnt,sn(cnt))=i
            y(cnt,sn(cnt))=j+1
           end do

    else !*---該当線分 (tk) の斑点番号を探して追加する．セグメント k のスポット番号 m を探して
t1-t2 を受け渡す
!------------------線分 S2(k) を 2 つの線分 t が共有する場合，x(mx,sn(mx)) はすでに書き換えられ
ている．
!                   s1(k) <= t2(l-1) =<s2(k) ならば，t2(l) と t2(l-1) を共有すること．
     if (FRG.eq.3) then ! 線分 t1-t2 を次で共有している場合 FRG=3
  m=cnt; FRG=-1
 else
   mx=cnt
       m=0
       FRG=-1
       do while(FRG .eq. -1)
         if (s2(tk).eq.x(mx,sn(mx))) then
           m=mx
           FRG=1
         else
           mx=mx-1
            if (mx.eq.0) then
             FRG=0
            end if
         end if
        end do
        if ((s1(tk).ge.t2(l-1)).and.(t2(l-1).le.s2(tk)).and.(l.lt.1)) then !s1-s2 を t2(l-1)
と共有
          m=m_old
        end if
      end if ! FRG=3
!---
       do i=t1(l),t2(l)
        sn(m)=sn(m)+1    !スポット番号 m のカウンター sn(m) を++しながら新しいスポット座標を
t1(l) から t2(l) までを保存
        x(m,sn(m))=i
        y(m,sn(m))=j+1
```

```
       end do
   m_old=m
    end if

  end do !繰り返し l の end

 end do !繰り返し j の end

!斑点の最大値 top(k), と位置 tx(k); ty(k) を探し, データを出力する.
 do k=1,cnt
  n=sn(k)
  top(k)=0.0
  do l=1,n
   i=x(k,l); j=y(k,l)
   tmax=c(i,j)
   if (tmax.gt.top(k)) then
    top(k)=tmax
    tx(k)=i; ty(k)=j
   end if
  end do
 end do

! 斑点データを spots.dat として出力
!open(unit=12,file='spots.dat',status='unknown')
 write (filename,*) keV     ! filename に 61 を用意
 filename=trim('sList_'//adjustl(filename))//'keV.dat' !sList_ に数字, keV.dat を加える.
 open(unit=12,file=filename,status='unknown')
print*,filename
   write(12,*) cnt !斑点数
  do k=1,cnt
   write(12,*) k,top(k),tx(k),ty(k),sn(k)!斑点番号, 最大値, 位置, 斑点画素数
   write(12,*) (x(k,l),l=1,sn(k),1) !斑点画素 x 座標
   write(12,*) (y(k,l),l=1,sn(k),1) !斑点画素 y 座標
  end do
 close(12)

!    最大輝度 tmax を探す
 tmax=top(1)
 do k=2,cnt
  if(top(k).gt.tmax) then
   tmax=top(k)
  end if
 end do

!  作成された斑点群 1 から sn(cnt) の画素位置 (i,j)=(x,y) を確認する
 do k=1,cnt
  n=sn(k)
  do l=1,n
   i=x(k,l); j=y(k,l)
!   sp(i,j)= sp(i,j)+100+150*top(k)/tmax     !最大強度 top で色分けして斑点区別しやすくした.
    sp(i,j)= sp(i,j)+120+50*(mod(k,5)-2) ! 色分け重視 16 カラーで見るとよい
  end do
 end do

 do k=1,cnt
  !最大値の位置に十字
  i0=tx(k); j0=ty(k)
  do m=-3,3
  if ((i0+m.ge.1).and.(i0+m.le.400).and.(j0+m.ge.1).and.(j0+m.le.570)) then !はみ出し処理
```

```
    sp(i0+m,j0)=255-sp(i0+m,j0)
    sp(i0,j0+m)=255-sp(i0,j0+m)
   end if
  end do
 end do

 write (filename,*) keV     ! filename に 61 を用意
 filename=trim('sImg_'//adjustl(filename))//'keV.txt' ! sImg_ に数字，keV.txt を加える.
 open(unit=12,file=filename,status='unknown')
print*,filename
 do j=1,570,1
  write(12,*) (sp(i,j), i=1,400,1)
 end do
 close(12)
end subroutine spots
```

maxima.f90

spot.f90 で作成されたローカル斑点群は，X線エネルギーごとの斑点データである．それらを利用して，X線エネルギー変化に対応して個々の斑点回折強度の変化を追跡し，各斑点の最大強度を見つけ出すのが，maxima.f90 である．

```
!*************************************************************
!      Searching for maxima of diffraction spots
!                 maxima.f90   ver. 1.0
!         by Kenji Suzuki, Niigata University
!******************************************** Nov. 28, 2018 ***
program maxima
  character*30 :: filename
  integer :: g, i,j,k,En,En1, En2,tx,ty,sn,cnt, kcnt(1:200)=0
!            グローバル斑点数を 1000 個に定義
  integer :: n(1:1000)=0,keV(1:1000,1:30)=0,l(1:1000,1:30)=0,rev(1:200,1:500)=0
  integer :: fn, ll, wl, spot_l
  integer :: ktx(1:200),kty(1:200),ksn(1:200)
  real*4 :: dr, tmax,top, peak(1:1000,1:30), ktop(1:200), gr1,gr2
  integer :: pn(1:200),pkeV(1:200),pl(1:200),px(1:200),py(1:200)
  real*4 :: ptop(1:200)
  integer :: Ib(1:400,1:570),m,i0,j0

  g=0     !       g: グローバル斑点番号

  spot_l=0
  print *,'Start  and stop X-ray energy: En1, En2 and radius dr' ! 処理対象の波長エネルギー範囲を指定
  read(5,*) En1, En2, dr !  【重要】斑点最大値のマッチングの近傍距離 dr =2.0

  do En=En1,En2-1  !最終の En2 を利用すると，En2 までしかないのにグローバル斑点カウンターが繰り上がる.
   spot_l=0
   write (filename,*) En     ! filename を用意
   filename=trim('sList_'//adjustl(filename))//'keV.dat' !sList_ に数字，keV.dat を加える.
print *,filename
   open(unit=11,file=filename,status='old')
   read(11,*) cnt; kcnt(En)=cnt  !斑点数
   do ll=1,cnt
     read(11,*) k,ktop(k),ktx(k),kty(k),ksn(k) !グローバル斑点番号，最大値，位置，斑点画素数
     read(11,*)  !読飛ばし (x(k,l),l=1,sn(k),1) !斑点画素 x 座標
     read(11,*)  !読飛ばし (y(k,l),l=1,sn(k),1) !斑点画素 y 座標
   end do
   close(11)

   do ll=1,cnt  ! ll=ローカル斑点番号
```

```
       top=ktop(ll); tx=ktx(ll); ty=kty(ll); sn=ksn(ll)

     If (rev(En,ll).eq.0) then !未処理の斑点 rev=0 であれば，斑点追跡を実行.
      wl=En
      g=g+1  !新たな斑点番号を生成
       !斑点追跡の繰り返し
      do while((spot_l.ne.-99).and.(wl.lt.En2)) ! spot_l=-99 ならば，斑点が消えたことになり，斑点追跡
終了
                                                ! 波長エネルギー範囲を超えた場合も斑点追跡終了
       wl=wl+1 !波長エネルギーを++しながら斑点を追跡
       call S_peak(wl,tx,ty,dr,top,spot_l) ! wl keV の画像内の斑点から近い点を探し出す.
       !対応するスポットがなかったら spot_l=-99 で打ち止め
        if (spot_l.ne.-99) then !該当スポットがあれば，グローバル斑点 g に追記
         n(g)=n(g)+1; keV(g,n(g))=wl; l(g,n(g))=spot_l; peak(g,n(g))=top
         rev(wl,spot_l)=g
        end if
       end do
     end if

     if (spot_l.eq.-99) then !グローバル斑点の追跡終了後に，spot_l を初期化する.
      spot_l=0
  if (n(g).eq.0) then !ローカル番号 n(g)=0 のとき，グローバル番号を取り消す g-1.
   g=g-1
  end if
     end if
! write(6,*) g,(peak(g,i),i=1,n(g),1)
   end do

  end do

print "('Number of global spots=',i0)",g

! output groval spot number
 open(unit=12,file='maxima.dat',status='unknown')
  write(12,*) g  ! Number of global spots
  do i=1,g
   write(12,*) i,n(i)
    do j=1,n(i)
      write(12,*) keV(i,j), l(i,j), peak(i,j)
    end do
  end do
 close(12)
print *,'make maxima.dat'

! 逆引きファイル maxima_rev.dat の出力
 open(unit=12,file='maxima_rev.dat',status='unknown')
  write(12,*) g  ! Number of global spots
  do i=En1,En2-1 !En2 の所は，斑点無しのため，rev を指定できない
   write(12,*) i, kcnt(i)
    do j=1,kcnt(i)
      write(12,*) j,rev(i,j)
    end do
  end do
 close(12)
print *,'make maxima_rev.dat'

!  グローバル斑点の輝度変化を GNUPLOT で表示するための 3D ファイル maxima.gnu
! g, keV, Intensity
!                         gnuplot command list
!> set ticslevel 0
!> set zrange [0:500]
!> set xlabel "Grobal spots number"
!> set ylabel "X-ray energy, keV"
!> set zlabel "%"
```

```
!> set points 1
!> splot 'maxima.gnu' with linespoints pt 6 lc 2
 open(unit=12,file='maxima.gnu',status='unknown')
  do i=1,g
    do j=1,n(i)
      write(12,*) i,keV(i,j), (peak(i,j)-peak(i,1))/peak(i,1)*100.0
    end do
    write(12,*)
  end do
 close(12)
print *,'make maxima.gnu'
!--------------- 各グローバル斑点の頂点リストを出力する --------------

 open(unit=12,file='maxima.peak',status='unknown')
  do i=1,g
  gr1=0.0; gr2=0.0
    if (n(i).ge.3) then
     do j=2,n(i)-1
!print *,i,n(i),j
      gr1= peak(i,j)-peak(i,j-1)
      gr2= peak(i,j+1)-peak(i,j)
      if ((gr1 .gt. 0.0).and.(gr2 .lt. 0.0)) then
       En=keV(i,j); spot_l=l(i,j)
       call Peak_top(En,spot_l,tx,ty,tmax) !peak top の斑点情報を求める
       write(12,*) i,En,spot_l,tx,ty,tmax
      end if
     end do
    end if
  end do
 close(12)
!                  maxima.peak の多重斑点については，混乱をなくすために最大のピークを選択するように修正.
  open(unit=11,file='maxima.peak',status='unknown')
 j=1
  do
    read (11, *, end=900) pn(j),pkeV(j),pl(j),px(j),py(j),ptop(j) ! ファイル終端ならば 900 に飛ぶ
    if (pn(j).eq.pn(j-1)) then
     if (ptop(j).gt.ptop(j-1)) then
      pn(j-1)=pn(j);pkeV(j-1)=pkeV(j);pl(j-1)=pl(j)
      px(j-1)=px(j);py(j-1)=py(j);ptop(j-1)=ptop(j)
      j=j-1
       else
      j=j-1
     end if
    end if
    j=j+1
  end do
900 continue
 close(11)

 open(unit=12,file='maxima.peak',status='unknown')
 write(12,'(I6)') j-1 !ピークトップ斑点数
  do i=1,j-1
!   pn(i),pkeV(i),pl(i),px(i),py(i),ptop(i) 斑点 g, エネルギー, ローカル番号 l, 画素位置 (x,y), 強度
    write(12,120) pn(i),pkeV(i),pl(i),px(i),py(i),ptop(i)
  end do
120 FORMAT(5(I4,2X),F8.1)
 close(12)
print *,'make maxima.peak'




! maxima.peak の斑点を Ib i.txt にマーキングする. 計算結果の確認にも利用
 sn=j-1
```

```
 do k=1,sn
!   pn(i),pkeV(i),pl(i),px(i),py(i),ptop(i) 斑点 g，エネルギー，ローカル番号 l，画素位置 (x,y)，強度
!                                        二値画像 Ib を開く
   write (filename,*) pkeV(k)     ! cfile に 61 を用意
   filename=trim('Ib'//adjustl(filename))//'.txt' !左詰にして，keV_d.txt を加える.
   open (unit=13, file=filename, status='unknown')
   do j=1,570
     read(13,*) (Ib(i,j),i=1,400,1)
   end do
   close(13)

  !ピークトップ斑点の位置に十字
   i0=px(k); j0=py(k)
   do m=-3,3
    if ((i0+m.ge.1).and.(i0+m.le.400).and.(j0+m.ge.1).and.(j0+m.le.570)) then !はみ出し処理
      Ib(i0+m,j0)=255
      Ib(i0,j0+m)=255
    end if
   end do

   open (unit=12, file=filename, status='unknown')
   do j=1,570
    write(12,*) (Ib(i,j),i=1,400,1)
   end do
   close(12)

 end do

end program maxima
!**********************************************************************
! S_peak search for local spot No. in keV
subroutine S_peak(keV,x,y,dr,t_max,spot_l)
  character*30 :: filename
  integer :: keV,x,y  !波長，最大輝度座標 (x,y) 最大輝度
  integer :: spot_l,k,tx,ty,sn,l,ns,xo,yo
  real*4 :: dr,top,r,r_min,t_max

   spot_l=0
   r_min=50.0

   write (filename,*) keV     ! filename に 61 を用意
   filename=trim('sList_'//adjustl(filename))//'keV.dat' !sList_ に数字，keV.dat を加える.
   open(unit=13,file=filename,status='unknown')

   read(13,*) ns  !ローカル斑点の総数
    do l=1,ns   ! l=ローカル斑点番号
      read(13,*) k,top,tx,ty,sn !斑点番号，最大値，位置，斑点画素数
      read(13,*)  !読飛ばし (x(k,l),l=1,sn(k),1) !斑点画素 x 座標
      read(13,*)  !読飛ばし (y(k,l),l=1,sn(k),1) !斑点画素 y 座標

      r=sqrt(float((tx-x)**2+(ty-y)**2))
      if(r.le.r_min) then    ! より近い斑点があれば，r_min を入れ替え
   r_min=r; spot_l=l; t_max=top; xo=tx; yo=ty
  end if
    end do
    close(13)

    ! 最近傍斑点を dr にて判定
    if (r_min.gt.dr) then  ! 近い斑点がなければ，spot_l=-99 を返して追跡終了
      spot_l=-99; t_max=0.0; x=0; y=0
    end if
x=xo; y=yo  !斑点位置をエネルギー keV の画像にあわせて，x,y を更新
end subroutine S_peak
```

```
!*********************************************************************
! Peak_top search for peak top of spot at keV
subroutine Peak_top(keV,l,x,y,tmax)
  character*30 :: filename
  integer :: i,j,k, keV,l,x,y,ns,sn
  real :: tmax

   write (filename,*) keV     ! filename に keV を用意
   filename=trim('sList_'//adjustl(filename))//'keV.dat' !sList_ に数字，keV.dat を加える.
   open(unit=11,file=filename,status='unknown')
   read(11,*) ns
   do i=1,l
    read(11,*) k,tmax,x,y,sn !斑点番号，最大値，位置，斑点画素数
    read(11,*)  !読飛ばし (x(k,l),l=1,sn(k),1) !斑点画素 x 座標
    read(11,*)  !読飛ばし (y(k,l),l=1,sn(k),1) !斑点画素 y 座標
   end do
   close(11)
end subroutine Peak_top
```

sImg-stack.ijm

各波長エネルギー *i* 像を二値化して，斑点抽出した最大輝度と位置の画像 sImg_ *i* keV.txt ファイルのスタック像を作成する ImageJ のマクロプログラムです．また，これで Ib *i* .txt を表示して二値化画像および回折斑点の二重露光法で利用するピークトップ位置のスタック像を作成・確認もできます.

```
// sImg ファイルのスタック像作成 sImg-stack.ijm
macro "Make sImg-stack" {
  dir = getDirectory("Choose a Directory"); //directry の選択
    XS = getNumber("スタートの keV=",XS); //最初のファイル
    XE = getNumber("ストップの keV=",XE); //読込みファイル数
    XE=XE+1; //読み込み終了条件
  for (i=XS; i<XE;i++){
    path1 = "open="+ dir+"sImg_"+i+"keV.txt"; //読込みファイル名
    print(path1);
    run("Text Image... ", path1); //テキストイメージの読込み
    run("Fire"); //Image Table として fire スタイルを定義
    setMinAndMax(0, 255);    //run("Brightness/Contrast..."); の min,max 指定
  }
 run("Images to Stack", "name=Stack title=[] use");
}

// Ib i.txt ファイルのスタック像作成
macro "Make Ib-stack" {
  dir = getDirectory("Choose a Directory"); //directry の選択
    XS = getNumber("スタートの keV=",XS); //最初のファイル
    XE = getNumber("ストップの keV=",XE); //読込みファイル数
    XE=XE+1; //読み込み終了条件
  for (i=XS; i<XE;i++){
    path1 = "open="+ dir+"Ib"+i+".txt"; //読込みファイル名
    print(path1);
    run("Text Image... ", path1); //テキストイメージの読込み
    run("Fire"); //Image Table として fire スタイルを定義
    setMinAndMax(0, 255);    //run("Brightness/Contrast..."); の min,max 指定
  }
 run("Images to Stack", "name=Stack title=[] use");
```

P1 _ spot.f90

```
!*********************************************************
!     Determination of positions of diffraction spots
!        P1_spot.f90  ver. 1.0
!            by Kenji SUZUKI, Niigata University
!************************************** Dec 1,2018 ******
program P1_spot

  character*30 :: filename, a
  integer :: n(1:300),keV(1:300),l(1:300)
  integer :: wl,ln,xs,ys
  real*4 ::top(1:300),tops, ti,xi,yi,x(1:300),y(1:300)
  integer :: Ib(1:400,1:570),m,i0,j0
  integer :: loop
  integer :: bn(1:200,1:2),n_data

   real*4 :: P1(1:400,1:570)

 open(unit=11,file='maxima.peak',status='old')
 read(11,*) m !ピークトップ斑点数
  do i=1,m
!   n(i),keV(i),l(i),x(i),y(i),top(i) 斑点 g, エネルギー, ローカル番号 l, 画素位置 (x,y), 強度
   read(11,*) n(i),keV(i),l(i),x(i),y(i),top(i)
  end do
 close(11)

 ! loop *******
 do loop=1,m
   wl=keV(loop);ln=l(loop); xs=x(loop); ys=y(loop) !斑点位置 (xs,ys) と強度 tops
   tops=top(loop)
   !グローバル斑点の呼び出し, 斑点の位置と強度 (xi,yi),ti

  call  spotP1(wl,xs,ys,tops, xi,yi,ti)
   x(loop)=xi; y(loop)=yi; top(loop)=ti

 write(6,601) loop, wl,ln, xi,yi,ti
601 format(3I5,3F9.2)
!print *, ' next spot OK? (y)'
!read(5,*) a
 ! loop *********
 end do

 open(unit=12,file='P1_spot.dat', status='unknown')
 write(12,*) m
 do i=1,m
  write(12,120)  i,keV(i),l(i),x(i),y(i),top(i)
 end do
 close(12)
120 format(3I6,3F10.2)

end program P1_spot
!
!************************* SUBROUTINE *********************************
!
!----------
subroutine spotP1(wl,x0,y0,top,x,y,t)
  character*30 :: filename,a
  real*4 :: b(1:400,1:570),c(1:400,1:570), top,t
  real*4 :: Gx,Gy,x,y
  integer :: i,j,k,l,m,n,d,x0,y0
  integer :: Ib(1:400,1:570)   !binery image
  integer :: wl, bn(1:200,1:2),n_data
!  real :: thr,a1, a2,a3,a4,df2,s
```

```
!  画像ファイルの読み込み
!  filename='68keV_d.txt'
 write (filename,*) wl     ! cfile に 61 を用意
 filename=trim(adjustl(filename))//'keV_d.txt' !左詰にして, keV_d.txt を加える.
! print *, filename
  open(11,file=filename,status='old')
    do j=1,570
      read(11,*) (c(i,j),i=1,400,1)
    end do
  close(11)

!------------------ 移動平均 m にて平均距離を決定 9 点移動平均 -------------------
  m=1
  do j=3,568
   do i=3,398
     s=0.0
     do l=j-m,j+m
        do k=i-m,i+m
          s=s+c(k,l)
        end do
      end do
      b(i,j)=s/float((2*m+1)**2)
   end do
  end do


!--------------------------- 二値化 -----------------------------
 thr=top/2.0   !各ピークの半価値で二値化

 do j=1,570
   do i=1,400
     if (b(i,j) .gt. thr) then
       Ib(i,j)=128
      else
        Ib(i,j)=0
     end if
   end do
 end do

!------------------- 二値画像をファイル Ib_temp.txt  ----------------
!  write (filename,*) keV     ! cfile に 61 を用意
!  filename=trim('Ib_temp'//adjustl(filename))//'.txt' !左詰にして, keV_d.txt を加える.
!  open (unit=12, file='Ib_temp.txt', status='unknown')
!  do j=1,570
!   write(12,*) (Ib(i,j),i=1,400,1)
!  end do
!  close(12)

  n_data = 0
! 二値画像から輪郭をとる peak(x,y), 画像 Ib,bn(*1,*2) 1*:1 から n_data, 2*:1=x,2=y, n_data=輪郭数

  call Trace(x0,y0,Ib,bn,n_data) !


  call centroid(c,bn,n_data,x,y,t) !calculate centroid(Gx,Gy)

  ! ---------------------- 輪郭確認用 Ib_trace.txt ----------------------
  open (unit=12, file='Ib_trace.txt', status='unknown')
  ! marking centroid(Gx,Gy)
   x0=int(x); y0=int(y)
   do i=x0-3,x0+3
    Ib(i,y0)=255
   end do
   do j=y0-3,y0+3
```

```
    Ib( x0,j)=255
   end do

  do j=1,570
   write(12,*) (Ib(i,j),i=1,400,1)
  end do
  close(12)
!*****
! print *,'you can check for file=Ib_trace.txt by ImageJ. (y=next)'
! read(5,*) a


end subroutine spotP1


!*******************************
subroutine Trace(k,l,x,p,n_data) ! tracing spot

integer :: k,l,z,x0,y0,vec,n_spot,n_data,i
integer :: s,t,dir,cnt,Frg,entry,wsum,n
integer ::  x(1:400,1:570), D(0:7,0:7),round(0:7), p(1:200,1:2)
integer :: BND

!            (incident direction, procedure No.)
data D(0,0),D(0,1),D(0,2),D(0,3),D(0,4),D(0,5),D(0,6),D(0,7)/6,7,0,1,2,3,4,5/
data D(1,0),D(1,1),D(1,2),D(1,3),D(1,4),D(1,5),D(1,6),D(1,7)/7,0,1,2,3,4,5,6/
data D(2,0),D(2,1),D(2,2),D(2,3),D(2,4),D(2,5),D(2,6),D(2,7)/0,1,2,3,4,5,6,7/
data D(3,0),D(3,1),D(3,2),D(3,3),D(3,4),D(3,5),D(3,6),D(3,7)/1,2,3,4,5,6,7,0/
data D(4,0),D(4,1),D(4,2),D(4,3),D(4,4),D(4,5),D(4,6),D(4,7)/2,3,4,5,6,7,0,1/
data D(5,0),D(5,1),D(5,2),D(5,3),D(5,4),D(5,5),D(5,6),D(5,7)/3,4,5,6,7,0,1,2/
data D(6,0),D(6,1),D(6,2),D(6,3),D(6,4),D(6,5),D(6,6),D(6,7)/4,5,6,7,0,1,2,3/
data D(7,0),D(7,1),D(7,2),D(7,3),D(7,4),D(7,5),D(7,6),D(7,7)/5,6,7,0,1,2,3,4/

! incident direction No.
! 6 5 4
! 7  *  3
! 0  1 2
!write(6,*) k,l
 z=1   !numbers of spot shape data
 vec=2
 BND=255
  i=x(k,l)


 do while(i.gt.0)
  k=k-1
  i=x(k,l)
 end do
 k=k+1

!     initial position(k,l)
  x0=k; y0=l
  p(z,1)=k; p(z,2)=l
!write(6,*) n_spot,z,p(z,1),p(z,2)
 entry=1
! return at start position
 do while (entry .eq. 1)

!write(6,*) k,l

!------- setting data around x(k,l) ---
! dir:0
s=k-1
t=l+1
```

```
if (x(s,t) == 0) then
 round(0)=0
else
 round(0)=1
end if
 ! dir:1
s=k
t=l+1
if (x(s,t) == 0) then
 round(1)=0
else
 round(1)=1
end if
 ! dir:2
 s=k+1
 t=l+1
if (x(s,t) == 0) then
 round(2)=0
else
 round(2)=1
end if
! dir:3
s=k+1
t=l
if (x(s,t) == 0) then
 round(3)=0
else
 round(3)=1
end if
! dir:4
 s=k+1
 t=l-1
if (x(s,t) == 0) then
 round(4)=0
else
 round(4)=1
end if
! dir:5
 s=k
 t=l-1
if (x(s,t) == 0) then
 round(5)=0
else
 round(5)=1
end if
! dir:6
 s=k-1
 t=l-1
if (x(s,t) == 0) then
 round(6)=0
else
 round(6)=1
end if
! dir:7
 s=k-1
 t=l
if (x(s,t) == 0) then
 round(7)=0
else
 round(7)=1
end if


!write(6,*) (round(i),i=0,7,1)
```

```
!    --- trace step ---
Frg=0
cnt=0  !step of detectig next direction

do while(Frg ==0)

dir=D(vec,cnt)

if (round(dir) == 1) then
 select case(dir)
 case(0)
  k=k-1
  l=l+1
  x(k,l)=BND
  z=z+1
  p(z,1)=k
  p(z,2)=l
  Frg=1
  vec=dir
  cnt=0
 case(1)
  l=l+1
  x(k,l)=BND
  z=z+1
  p(z,1)=k
  p(z,2)=l
  Frg=1
  vec=dir
  cnt=0
 case(2)
  k=k+1
  l=l+1
  x(k,l)=BND
  z=z+1
  p(z,1)=k
  p(z,2)=l
  Frg=1
  vec=dir
  cnt=0
 case(3)
  k=k+1
  x(k,l)=BND
  z=z+1
  p(z,1)=k
  p(z,2)=l
  Frg=1
  vec=dir
  cnt=0
 case(4)
  k=k+1
  l=l-1
  x(k,l)=BND
  z=z+1
  p(z,1)=k
  p(z,2)=l
  Frg=1
  vec=dir
  cnt=0
 case(5)
  l=l-1
  x(k,l)=BND
  z=z+1
  p(z,1)=k
```

```
  p(z,2)=l
  Frg=1
  vec=dir
  cnt=0
 case(6)
  k=k-1
  l=l-1
  x(k,l)=BND
  z=z+1
  p(z,1)=k
  p(z,2)=l
  Frg=1
  vec=dir
  cnt=0
 case(7)
  k=k-1
  x(k,l)=BND
  z=z+1
  p(z,1)=k
  p(z,2)=l
  Frg=1
  vec=dir
  cnt=0
 end select

!write(6,*) n_spot,z, p(z,1), p(z,2)
end if

! entry=0: finish the trace around the spot, addition of (Frg .eq. 1)
    if ((k .eq. x0) .and.(l .eq. y0) .and. (Frg .eq. 1)) then
 entry= 0
    end if
 cnt=cnt+1
 ! only one spot
 wsum=0
 do n=0,7
   wsum=wsum+round(n)
 end do
 if (wsum == 0) then
   p(z,1)=k
   p(z,2)=l
   x(k,l)=BND
   z=z+1
   Frg=1
   entry= 0
 end if

end do

end do
n_data=z
end subroutine

!-------------------------
!    calculating centroid of spot
subroutine centroid(b,bn,n,Gx,Gy,t)  ! b:画像, bn:輪郭, n:データ数, G:重心
  real*4 :: b(1:400,1:570)
  real*4 :: Gx,Gy, sumb,Mx,My,t
  integer :: bn(1:200,1:2),i,j,k,n,x1,x2,y1,y2
  integer :: h1,h2,h3,h4,h, ip,jp
  integer ::x(1:200),y(1:200), i_max,i_min,j_max,j_min

    ! パラメー i,j の斑点重心計算のための捜査範囲 i_max,i_min,j_max,j_min
  i_max=bn(1,1); i_min=bn(1,1); j_max=bn(1,2); j_min=bn(1,2)
```

```
  do i=2,n
    if (bn(i,1).gt.i_max) then
       i_max=bn(i,1)
    end if
    if (bn(i,1).lt.i_min) then
       i_min=bn(i,1)
    end if
    if (bn(i,2).gt.j_max) then
       j_max=bn(i,2)
    end if
    if (bn(i,2).lt.j_min) then
       j_min=bn(i,2)
    end if
  end do

! 初期化
  sumb=0.0; Mx=0.0; My=0.0
!  loop j
  do j=j_min,j_max
    do i=i_min,i_max
      ! 画素位置 (i,j) を選択

      h1=0; h2=0; h3=0; h4=0 ! h 判定フラグ

      do ip=i,i_min,-1
       do k=1,n
       if ((ip.eq.bn(k,1)).and.(j.eq.bn(k,2))) then !左一致
        h1=1
       end if
       end do
      end do

      do ip=i,i_max,1
       do k=1,n
       if ((ip.eq.bn(k,1)).and.(j.eq.bn(k,2))) then !右一致
        h2=1
       end if
       end do
      end do

      do jp=j,j_min,-1
       do k=1,n
       if ((i.eq.bn(k,1)).and.(jp.eq.bn(k,2))) then !上一致
        h3=1
       end if
       end do
      end do

      do jp=j,j_max,1
       do k=1,n
       if ((i.eq.bn(k,1)).and.(jp.eq.bn(k,2))) then !下一致
        h4=1
       end if
       end do
      end do

      h=h1*h2*h3*h4
      if (h.eq.1) then !斑点に含まれる (i,j) なら重み付き計算
        sumb=sumb+b(i,j); Mx=Mx+b(i,j)*float(i); My=My+b(i,j)*float(j)
      end if
    end do
!  loop j
  end do
```

```
  Gx=Mx/sumb; Gy=My/sumb; i=int(Gx); j=int(Gy)
  t=b(i,j)
end subroutine centroid
```

ssd.f90

P1 から P2 への最適写像パラメータを求めるプログラム．P1 の (x0,y0)-(x1y1) のテンプレートと拡大率 Mg と P2 の原点 (i,j) を移動しながら差分自乗和を最小にする方法 (SSD: sum of squared difference) を利用している．

```
! --------------- SSD (sum of squared difference) ---------------
!   ssd.f90                    by K. Suzuki, Niigata University
!-----------------------------------  Dec 11 2018 -------------

program SSD
 character*30 ::filename,a
 integer :: i,j,k,l,m,x0,y0,x1,y1,x1P2,y1P2
 integer :: P1x,P1y,P2x,P2y,iE,jE, op
 data P1x,P1y,P2x,P2y/400,570,800,950/
 integer ::cnt,i_min(1:300),j_min(1:300),i_max(1:300),j_max(1:300)
 real*4 :: P1(1:400,1:570)=0.0,P2(1:801,1:951)=0.0
 real*4 :: tp(1:600,1:910)=0.0,er ! Mg: magnification, tp:template of P1
 real*4 :: Mg(1:300),er_min(1:300),er_max(1:300),t1, th1=40.0, th2=80.0
 real*4 :: fit

 x0=200; y0=150; x1=400; y1=455 ! template in P1

!                       P1, P2 の読み込み 70d_P1.txt,70d_P2.txt
 filename='70d_P1.txt' ! print *,'File name of P1?'
! read(5,*) filename
 open(unit=11,file=filename, status='old')
  do j=1,P1y
   read(11,*) (P1(i,j),i=1,P1x,1)
  end do
 close(11)

! print *,'File name of P2?'
 filename='70d_P2.txt'
! read(5,*) filename
 open(unit=11,file=filename, status='old')
  do j=1,P2y
   read(11,*) (P2(i,j),i=1,P2x,1)
  end do
 close(11)

!------- 閾値 th1 および th2 でバックグラウンド処理 start-----------------

  do j=1,P1y
   do i=1,P1x
    if (P1(i,j) .lt. th1) then
      P1(i,j)=0.0
    end if
   end do
  end do

  do j=1,P2y
   do i=1,P2x
    if (P2(i,j) .lt. th2) then
      P2(i,j)=0.0
    end if
   end do
  end do
!------- バックグラウンド処理 end -----------------
```

```
 iE=5;jE=5; y1P2=300
 cnt=0
 do while((iE.gt.2).and.(jE.gt.2).and.(y1P2.lt.897))
  cnt=cnt+1
  Mg(cnt)= 2.5+0.01*float(cnt)

 !  テンプレート tp の作成
  x1P2=int(float(x1-x0)*Mg(cnt)); y1P2=int(float(y1-y0)*Mg(cnt)) ! テンプレートサイズ x1P2,y1P2
  iE=P2x-x1P2-1; jE=P2y-y1P2-1     !パラメータ i,j の範囲の計算
  call cpu_time(t1)
print 601,t1/60.0,iE,jE,x1P2,y1P2
601 format(F8.2,' min',I5,I5,' template sizes',I5,I5)

                  c=0.0         ! 濃度 c の初期化
  do l=1,y1P2
   do k=1,x1P2
    i = int(float(k)/Mg(cnt))+x0 ; j = int(float(l)/Mg(cnt))+y0
    tp(k,l)=P1(i,j)
                  c=c+tp(k,l)**2  ! 濃度 c 計算
   end do
  end do

 open(unit=14,file='tp.txt',status='unknown')
 do l=1,y1P2
  write(14,*) (tp(k,l),k=1,x1P2)
 end do
 close(14)

  er=0.0   ! er 初期化
    do l=1,y1P2
     do k=1,x1P2
      er=er+(P2(k,l)-tp(k,l))**2
     end do
    end do
  er_min(cnt)=er; i_min(cnt)=1; j_min(cnt)=1 ! 初期値
  er_max(cnt)=er; i_max(cnt)=1; j_max(cnt)=1 ! 初期値

!print *,Mg,k,l
!    P2 画像の原点位置 (i,j) を変化させてながら誤差 er の計算
  do j=1,jE
   do i=1,iE
    er=0.0   ! er 初期化
    do l=1,y1P2
     do k=1,x1P2
      er=er+(P2(i+k-1,j+l-1)-tp(k,l))**2
     end do
    end do
    if(er .lt. er_min(cnt)) then
     er_min(cnt)=er; i_min(cnt)=i-1; j_min(cnt)=j-1
    end if
    if(er .gt. er_max(cnt)) then
     er_max(cnt)=er; i_max(cnt)=i-1; j_max(cnt)=j-1
    end if
   end do
  end do

  er_min(cnt)=er_min(cnt)/c !最小誤差
  er_max(cnt)=er_max(cnt)/c !最大誤差

 print 120,Mg(cnt), i_min(cnt), j_min(cnt),er_min(cnt)
 print 120,Mg(cnt), i_max(cnt), j_max(cnt),er_max(cnt)

 end do
```

```
!------------------------------

 open(unit=12,file='ssd_result.dat', status='unknown')  !ssd_result.dat として ssd の結果を出力
  write(12,*) cnt
  do i=1,cnt-1
   write(12,121) Mg(i),i_min(i), j_min(i),er_min(i),i_max(i),j_max(i),er_max(i)
  end do
 close(12)

 fit=er_min(1); op=1
 do i=2,cnt
  if (er_min(i) .lt. fit) then
   fit=er_min(i); op=i
  end if
 end do
 !  最適な拡大率の画像 P1_fitting.txt を出力
  x1P2=int(float(x1-x0)*Mg(op)); y1P2=int(float(y1-y0)*Mg(op)) ! テンプレートサイズ x1P2,y1P2
  iE=P2x-x1P2-1; jE=P2y-y1P2-1     !パラメータ i,j の範囲の計算
  do l=1,y1P2
   do k=1,x1P2
    i = int(float(k)/Mg(op))+x0 ; j = int(float(l)/Mg(op))+y0
    tp(k,l)=P1(i,j)
   end do
  end do
 write(6,121) Mg(op),i_min(op), j_min(op),er_min(op),i_max(op),j_max(op),er_max(op)
 open(unit=14,file='P1_fitting.txt',status='unknown')
 do l=1,y1P2
  write(14,140) (tp(k,l),k=1,x1P2)
 end do
 close(14)

 120 format(F6.2,2I7,2x,E9.4)
 121 format(F6.2,2I7,2x,E9.4,2I7,2x,E9.4)
 140 format(800(F6.1,2x))
end program SSD
```

spot2.f90

spot2.f90 は，P2 の *i* keV _ d.txt 画像から二値化して斑点を取り出し，その最大値をマークする．フォルダー/P2/keV _ PbW/の中で実行すると，ローカル斑点番号，斑点の強度，位置を保存したファイル sList_ *i* keV.dat および斑点処理結果を示す画像 sImg_ *i* keV.txt[13]を作成する．

```
!************************ maxima.f90 *********************************
!     Searching maxima of i keV_d.txt in P2 images
!     P1=400 x 570
!   spot2.f90          by Kenji Suzuki, Niigata University
!****************************************************** Desember 13, 2018

program spot2

  character*30 ::a,filename
  real :: b(1:800,1:950),c(1:800,1:950)
  integer :: i,j,k,l,m,n,d,hsize,vsize
  integer :: Ib(1:800,1:950)   !binery image
  integer :: keV1,keV2,keV
  real :: thr,a1, a2,a3,a4,df2,s
```

[13] sImg_ *i* keV.txt の画像ファイル群は，sImg-stack.ijm のマクロで読込むことでスタックを作成できる．

```
hsize=800; vsize=950        ! P2 画像サイズ

print *,'start and stop keV= keV1,keV2'
read(5,*) keV1,keV2

do keV=keV1, keV2,1

!  画像ファイルの読み込み
!  filename='68keV_d.txt'
 write (filename,*) keV     ! cfile に 61 を用意
 filename=trim(adjustl(filename))//'keV_d.txt' !左詰にして，keV_d.txt を加える.

  open(11,file=filename,status='old')
    do j=1,vsize
      read(11,*) (c(i,j),i=1,hsize,1)
    end do
  close(11)

!------------------ 移動平均 m にて平均距離を決定 9 点移動平均 -------------------
  m=1
  do j=3,vsize-2
   do i=3,hsize-2
     s=0.0
     do l=j-m,j+m
        do k=i-m,i+m
          s=s+c(k,l)
        end do
      end do
      b(i,j)=s/float((2*m+1)**2)
   end do
  end do


! P2 の周囲に斑点が含まれていると，斑点が閉じないので枠を黒=b で処理する
 do i=1,hsize
  b(i,1)=0.0; b(i,vsize)=0.0
 end do
 do i=1,vsize
  b(1,i)=0.0; b(hsize,i)=0.0
 end do

!--------------------------- 二値化 -----------------------------
 thr=20.0       !二値化の閾値
 do j=1,vsize
   do i=1,hsize
     if (b(i,j) .gt. thr) then
       Ib(i,j)=128
      else
        Ib(i,j)=0
     end if
   end do
 end do

!------------------- 二値画像 Ib をファイル Ib.txt で出力 ----------------
 write (filename,*) keV     ! cfile に 61 を用意
 filename=trim('Ib'//adjustl(filename))//'.txt' !左詰にして，keV_d.txt を加える.
print *, filename
 open (unit=12, file=filename, status='unknown')
 do j=1,vsize
```

```
  write(12,*) (Ib(i,j),i=1,hsize,1)
 end do
 close(12)
print *,'Go into',keV
 call spots(Ib,b,keV,hsize,vsize)                    !二値画像から斑点を探し出す.
print *,'Return from',keV

end do
end program spot2

!************************ SUBROUTINE ****************************
! 斑点を認識するサブルーチン spot は，機織の横糸のようにして斑点の画素データを作成している.
subroutine spots(ib,c,keV,hsize,vsize)
  character*30 :: filename
  integer :: keV, hsize,vsize
  real :: c(1:800,1:950) ! CdTe-image
  integer :: ib(1:800,1:950),w,b   !binery image
  integer :: sp(1:800,1:950)=0 ! spot image
  integer :: x(1:400,1:900)=0, y(1:400,1:900)=0, sn(1:900)=0, cnt !spot no, x,y
  integer :: s1(1:20)=0,s2(1:20)=0,t1(1:20)=0,t2(1:20)=0,ns, nt, i,j,k,l
  integer :: m,n, ni,tk,FRG,mx,m_old,tt1(1:20),tt2(1:20),ntt
  integer :: tx(1:400),ty(1:400),i0,j0 !斑点の最大値と位置 (tx,ty)
  real :: tmax,top(1:500)

! 【重要】初期化　繰り返し call されるので，変数とパラメータを初期化しないとエラーが起きる.
  w=128
  b=0
  cnt=0

 do j=1,vsize
  do i=1,hsize
   sp(i,j)=0
  end do
 end do
 do j=1,400
  tx(j)=0; ty(j)=0; sn(j)=0
  do i=1,900
   x(j,i)=0; y(j,i)=0
  end do
 end do

!      掃き出し sweep mode

 do j=1,vsize-2

  do k=1, 20
   s1(k)=0; s2(k)=0; t1(k)=0; t2(k)=0   !線分の初期化
  end do
!                     j 行目のライン設定 s1, s2
  k=0    ! line 1
    do i=1,hsize-1
      if ((ib(i,j).eq.b).and.(ib(i+1,j).eq.w)) then
       k=k+1
       s1(k)=i+1
      end if
      if ((ib(i,j).eq.w).and.(ib(i+1,j).eq.b)) then
       s2(k)=i
      end if
    end do
```

```
    ns=k !Number of line segments

!                       j+1 行目のライン設定 s1, s2
    k=0    ! line 2
    do i=1,hsize-1
      if ((ib(i,j+1).eq.b).and.(ib(i+1,j+1).eq.w)) then
       k=k+1
       t1(k)=i+1
      end if
      if ((ib(i,j+1).eq.w).and.(ib(i+1,j+1).eq.b)) then
       t2(k)=i
      end if
    end do
    nt=k !Number of line segments
!                       j+2 行目のライン設定 tt1, tt2  上の斑点が下で結合している場合
    k=0    ! line 2
    do i=1,hsize-1
      if ((ib(i,j+2).eq.b).and.(ib(i+1,j+2).eq.w)) then
       k=k+1
       tt1(k)=i+1
      end if
      if ((ib(i,j+2).eq.w).and.(ib(i+1,j+2).eq.b)) then
       tt2(k)=i
      end if
    end do
    ntt=k !Number of line segments
!ここまでは perfect

!                       セグメント s1,s2 と t1,t2 の接触を調べて斑点の追加または生成を行う
   do l=1,nt
    tk=0
    do k=1,ns
    !t1 または t2 が斑点 k(s1,s2) を探す．該当線分 (tk) がない場合 (tk=0) は，新規作成
     if (((s1(k) <= t1(l)).and.(t1(l) <= s2(k))).or.((s1(k) <= t2(l)).and.(t2(l) <= s2(k)))) then
      tk=k
     end if
     if ((t1(l).lt.s1(k)).and.(s1(k).lt.t2(l)).and.(s2(k).lt.t2(l))) then !s1-s2がt1-t2
に含まれる
      tk=k
     end if
    end do !繰り返し k の end
!  j+2 で 線分 t1-t2 を共有している場合 cnt++をさせない．FRG=3
    do k=1,ntt
     if (((tt1(k).le.t1(l)).and.(t1(l).le.tt2(k))).and.((tt1(k).le.t2(l-1)).and.(t2(l-1).le.tt2(k))
      tk=cnt; FRG=3
     end if
    end do

    if (tk.eq.0) then ! tk=0: t1, t2 が斑点 (s1,s2) に含まれない，新しい斑点の生成
!         新しい斑点の生成，cnt:斑点番号，sn(cnt):斑点の画素数，(x,y):画素位置
         cnt=cnt+1
          do i=t1(l),t2(l)
           sn(cnt)=sn(cnt)+1
           x(cnt,sn(cnt))=i
           y(cnt,sn(cnt))=j+1
          end do
    else !該当線分 (tk) の斑点番号を探して追加する．セグメント k のスポット番号 m を探して t1-t2
を受け渡す
!------------------線分 S2(k) を 2 つの線分 t が共有する場合，x(mx,sn(mx)) はすでに書き換えられ
```

ている.
```
!                  s1(k) <= t2(l-1) =<s2(k) ならば，t2(l) と t2(l-1) を共有すること.
     if (FRG.eq.3) then ! 線分 t1-t2 を次で共有している場合 FRG=3
  m=cnt; FRG=-1
 else
   mx=cnt
       m=0
       FRG=-1
       do while(FRG .eq. -1)
         if (s2(tk).eq.x(mx,sn(mx))) then
           m=mx
           FRG=1
         else
           mx=mx-1
            if (mx.eq.0) then
             FRG=0
            end if
         end if
        end do
        if ((s1(tk).ge.t2(l-1)).and.(t2(l-1).le.s2(tk)).and.(l.lt.1)) then !s1-s2 を t2(l-1)
と共有
          m=m_old
        end if
      end if ! FRG=3
!--
        do i=t1(l),t2(l)
         sn(m)=sn(m)+1  !スポット番号 m のカウンター sn(m) を++しながら新しいスポット座標を t1(l)
から t2(l) までを保存
         x(m,sn(m))=i
         y(m,sn(m))=j+1
        end do
        m_old=m
    end if

  end do !繰り返し l の end

 end do !繰り返し j の end

!斑点の最大値 top(k), と位置 tx(k); ty(k) を探し，データを出力する.
 do k=1,cnt
  n=sn(k)
  top(k)=0.0
  do l=1,n
   i=x(k,l); j=y(k,l)
   tmax=c(i,j)
   if (tmax.gt.top(k)) then
    top(k)=tmax
    tx(k)=i; ty(k)=j
   end if
  end do
 end do

! 斑点データを spots.dat として出力
!open(unit=12,file='spots.dat',status='unknown')
 write (filename,*) keV     ! filename に 61 を用意
 filename=trim('sList_'//adjustl(filename))//'keV.dat' !sList_ に数字，keV.dat を加える.
 open(unit=12,file=filename,status='unknown')
print*,filename
   write(12,*) cnt !斑点数
```

```
  do k=1,cnt
   write(12,*) k,top(k),tx(k),ty(k),sn(k)!斑点番号，最大値，位置，斑点画素数
   write(12,*) (x(k,l),l=1,sn(k),1) !斑点画素 x 座標
   write(12,*) (y(k,l),l=1,sn(k),1) !斑点画素 y 座標
  end do
 close(12)

!     最大輝度 tmax を探す
 tmax=top(1)
 do k=2,cnt
  if(top(k).gt.tmax) then
   tmax=top(k)
  end if
 end do

!  作成された斑点群 1 から sn(cnt) の画素位置 (i,j)=(x,y) を確認する
 do k=1,cnt
  n=sn(k)
  do l=1,n
   i=x(k,l); j=y(k,l)  !最終の y(k,l)=0
!   sp(i,j)= sp(i,j)+100+150*top(k)/tmax     !最大強度 top で色分けして斑点区別しやすくした.
    sp(i,j)= sp(i,j)+120+50*(mod(k,5)-2) ! 色分け重視 16 カラーで見るとよい
  end do
 end do

 do k=1,cnt
  !最大値の位置に十字
  i0=tx(k); j0=ty(k)
  do m=-3,3
  if ((i0+m.ge.1).and.(i0+m.le.hsize).and.(j0+m.ge.1).and.(j0+m.le.vsize)) then !はみ出し
処理
    sp(i0+m,j0)=255!-sp(i0+m,j0)
    sp(i0,j0+m)=255!-sp(i0,j0+m)
   end if
  end do
 end do

 write (filename,*) keV      ! filename に 61 を用意
 filename=trim('sImg_'//adjustl(filename))//'keV.txt' ! sImg_ に数字，keV.txt を加える.
 open(unit=12,file=filename,status='unknown')
print*,filename
 do j=1,vsize,1
  write(12,*) (sp(i,j), i=1,hsize,1)
 end do
 close(12)
end subroutine spots
```

P2 _ spot.f90

P2 _ spot.f90 は，spot2.f90 で作成された sList _ *i*keV.dat を読み込んで，keV の斑点リストに従って斑点強度の 7 割の二値化で，斑点の重み付き重心を計算してリストアップする．各 keV の P2 画像の斑点リストは，**P2_spot** _ *i* **.dat** に出力される．また，各 keV の P2 画像の斑点の輪郭と重み付き重心は，**Ib _trace** _ *i* **.txt** に出力される．Ib _trace _ *i* .txt は，ImageJ マクロ sImg-stack.ijm で読込み，スタックを作成できる．

```
!**********************************************************************
!    P2_spot.f90    by Kenji Suzuki, Niigata University
!      Calculation positions of spots for P2 images
! ****************************************December 18, 2018 ****
```

```
program P2_spot
 integer :: kev,k1,k2, hsize, vsize

  hsize=800; vsize=950

 print *,'start keV and stop keV: k1, k2'
 read(5,*) k1,k2

 do kev=k1,k2

 call spotP2(kev,hsize,vsize)

 end do  !kev

end program P2_spot

!*****************************************************************************************
! SUBRUTINE  SUBRUTINE  SUBRUTINE SUBRUTINE  SUBRUTINE  SUBRUTINE  SUBRUTINE  SUBRUTINE

! ------------------------------------- spotP2 ----------------------------
subroutine spotP2(keV,hsize,vsize)
 character*30 :: filename
 integer :: keV,i,j,k,l,m,ns,is,x0,y0
 integer :: ix(1:400),iy(1:400),sn,hsize,vsize
 integer :: Ib(1:800,1:950)=0,bn(1:800,1:2)=0,n_data, It(1:800,1:950)=0
 real*4 :: top(1:400),c(1:800,1:950),s,thr,x,y,t

 ! 斑点データを sList_i keV.dat を読み込む
  write (filename,*) keV     ! filename に keV を代入
  filename=trim('sList_'//adjustl(filename))//'keV.dat' !sList_ に数字, keV.dat を加える.

  open(unit=11,file=filename,status='unknown')
!print*, keV,filename
   read(11,*) m !斑点数
  do k=1,m
   read(11,*) i,top(i),ix(i),iy(i),sn  !斑点番号, 最大値, 位置, 斑点画素数
   read(11,*) !斑点画素 x 座標
   read(11,*) !斑点画素 y 座標
  end do
  close(11)

  ns=m  !斑点数 ns

  write (filename,*) keV     !keV の値を代入
  filename=trim(adjustl(filename))//'keV_d.txt' !左詰にして, keV_d.txt を加える.
  open(11,file=filename,status='old')
    do j=1,vsize
      read(11,*) (c(i,j),i=1,hsize,1)
    end do
  close(11)

!------------------ 移動平均 m にて平均距離を決定 9 点移動平均 -------------------
  m=1
  do j=3,vsize-2
   do i=3,hsize-2
     s=0.0
     do l=j-m,j+m
        do k=i-m,i+m
```

```
          s=s+c(k,l)
        end do
      end do
      c(i,j)=s/float((2*m+1)**2)
   end do
  end do

 ! P2 の周囲に斑点が含まれていると，斑点が閉じないので枠を黒=0.0 で処理する
  do i=1,hsize
   c(i,1)=0.0; c(i,vsize)=0.0
  end do
  do i=1,vsize
   c(1,i)=0.0; c(hsize,i)=0.0
 end do

!keV 斑点のデータファイル spotP2 i.dat
  write (filename,*) keV     !keV の値を代入
  filename=trim('spotP2_'//adjustl(filename))//'.dat' !spotP2 + 数字 + keV.dat を加える.
  open(10,file=filename,status='unknown')
  write(10,*) ns
!  ループ is 開始
 do is=1,ns
  thr=top(is)*0.7   !ピークトップの 6 割 または 20 の高さで斑点の領域を二値化
!  if (thr .le. 20.0) then
!   thr=20.0
!  end if
  do j=1,vsize
   do i=1,hsize
    if (c(i,j).gt.thr) then
     Ib(i,j)=128
    else
     Ib(i,j)=0
    end if
   end do
  end do
 n_data=0

  call Trace(ix(is),iy(is),Ib,bn,n_data) ! tracing spot

  do l=1,n_data  ! 斑点の輪郭を It に書き込む
   i=bn(l,1); j=bn(l,2); It(i,j)=100
  end do

  call centroid(c,bn,n_data,x,y,t) !重み付き重心 (Gx,Gy) と高さを計算する

  write(10,101) is,x,y,t

    ! marking centroid(Gx,Gy)
   x0=int(x); y0=int(y)
   do i=x0-3,x0+3
    if ((i.ge.1).and.(i.le.hsize)) then
      It(i,y0)=255
end if
   end do
   do j=y0-3,y0+3
    if ((j.ge.1).and.(j.le.vsize)) then
      It(x0,j)=255
end if
   end do
```

```
 end do ! ns
 close(10)
print 102, keV,filename,ns

!  ループ is 終了

! ----------------------- 輪郭確認用 Ib_trace i.txt ----------------------
  write (filename,*) keV     !keV の値を代入
  filename=trim('Ib_trace'//adjustl(filename))//'.txt' !spotP2 + 数字 + keV.dat を加える.

  open (unit=12, file=filename, status='unknown')
    do j=1,vsize
     write(12,*) (It(i,j),i=1,hsize)
    end do
  close(12)
print*,filename

  do j=1,vsize  !初期化 It(i,j)
   do i=1,hsize
    It(i,j)=0
   end do
  end do

 101 format(I4,2x,F8.3,2x,F8.3,2x,F10.2)
 102 format(I4,'keV 'A20,2x,I6)
end subroutine spotP2

!*******************************
subroutine Trace(k,l,x,p,n_data) ! tracing spot

integer :: k,l,z,x0,y0,vec,n_spot,n_data,i
integer :: s,t,dir,cnt,Frg,entry,wsum,n
integer ::  x(1:800,1:950), D(0:7,0:7),round(0:7), p(1:800,1:2)
integer :: BND

!            (incident direction, procedure No.)
data D(0,0),D(0,1),D(0,2),D(0,3),D(0,4),D(0,5),D(0,6),D(0,7)/6,7,0,1,2,3,4,5/
data D(1,0),D(1,1),D(1,2),D(1,3),D(1,4),D(1,5),D(1,6),D(1,7)/7,0,1,2,3,4,5,6/
data D(2,0),D(2,1),D(2,2),D(2,3),D(2,4),D(2,5),D(2,6),D(2,7)/0,1,2,3,4,5,6,7/
data D(3,0),D(3,1),D(3,2),D(3,3),D(3,4),D(3,5),D(3,6),D(3,7)/1,2,3,4,5,6,7,0/
data D(4,0),D(4,1),D(4,2),D(4,3),D(4,4),D(4,5),D(4,6),D(4,7)/2,3,4,5,6,7,0,1/
data D(5,0),D(5,1),D(5,2),D(5,3),D(5,4),D(5,5),D(5,6),D(5,7)/3,4,5,6,7,0,1,2/
data D(6,0),D(6,1),D(6,2),D(6,3),D(6,4),D(6,5),D(6,6),D(6,7)/4,5,6,7,0,1,2,3/
data D(7,0),D(7,1),D(7,2),D(7,3),D(7,4),D(7,5),D(7,6),D(7,7)/5,6,7,0,1,2,3,4/

! incident direction No.
! 6 5 4
! 7  *  3
! 0  1 2
!write(6,*) k,l
 z=1   !numbers of spot shape data
 vec=2
 BND=255
  i=x(k,l)

 do while(i.gt.0)
  k=k-1
  i=x(k,l)
 end do
```

```
 k=k+1

!     initial position(k,l)
  x0=k; y0=l
  p(z,1)=k; p(z,2)=l
!write(6,*) n_spot,z,p(z,1),p(z,2)
 entry=1
! return at start position
 do while (entry .eq. 1)

!write(6,*) k,l

!------- setting data around x(k,l) ---
! dir:0
s=k-1
t=l+1
if (x(s,t) == 0) then
 round(0)=0
else
 round(0)=1
end if
 ! dir:1
s=k
t=l+1
if (x(s,t) == 0) then
 round(1)=0
else
 round(1)=1
end if
 ! dir:2
 s=k+1
 t=l+1
if (x(s,t) == 0) then
 round(2)=0
else
 round(2)=1
end if
! dir:3
s=k+1
t=l
if (x(s,t) == 0) then
 round(3)=0
else
 round(3)=1
end if
! dir:4
 s=k+1
 t=l-1
if (x(s,t) == 0) then
 round(4)=0
else
 round(4)=1
end if
! dir:5
 s=k
 t=l-1
if (x(s,t) == 0) then
 round(5)=0
else
```

```
 round(5)=1
end if
! dir:6
 s=k-1
 t=l-1
if (x(s,t) == 0) then
 round(6)=0
else
 round(6)=1
end if
! dir:7
 s=k-1
 t=l
if (x(s,t) == 0) then
 round(7)=0
else
 round(7)=1
end if


!write(6,*) (round(i),i=0,7,1)

!    --- trace step ---
Frg=0
cnt=0  !step of detectig next direction

do while(Frg ==0)

dir=D(vec,cnt)

if (round(dir) == 1) then
 select case(dir)
 case(0)
  k=k-1; l=l+1; x(k,l)=BND; z=z+1
  p(z,1)=k; p(z,2)=l
  Frg=1; vec=dir; cnt=0
 case(1)
  l=l+1; x(k,l)=BND; z=z+1
  p(z,1)=k; p(z,2)=l
  Frg=1; vec=dir; cnt=0
 case(2)
  k=k+1; l=l+1; x(k,l)=BND; z=z+1
  p(z,1)=k; p(z,2)=l
  Frg=1; vec=dir; cnt=0
 case(3)
  k=k+1; x(k,l)=BND; z=z+1
  p(z,1)=k; p(z,2)=l
  Frg=1; vec=dir; cnt=0
 case(4)
  k=k+1; l=l-1; x(k,l)=BND; z=z+1
  p(z,1)=k; p(z,2)=l
  Frg=1; vec=dir; cnt=0
 case(5)
  l=l-1; x(k,l)=BND; z=z+1
  p(z,1)=k; p(z,2)=l
  Frg=1; vec=dir; cnt=0
 case(6)
  k=k-1; l=l-1; x(k,l)=BND; z=z+1
  p(z,1)=k; p(z,2)=l
```

```
  Frg=1; vec=dir; cnt=0
 case(7)
  k=k-1; x(k,l)=BND; z=z+1
  p(z,1)=k; p(z,2)=l
  Frg=1; vec=dir; cnt=0
 end select

!write(6,*) n_spot,z, p(z,1), p(z,2)
end if

! entry=0: finish the trace around the spot, addition of (Frg .eq. 1)
    if ((k .eq. x0) .and.(l .eq. y0) .and. (Frg .eq. 1)) then
 entry= 0
    end if
 cnt=cnt+1
 ! only one spot
 wsum=0
 do n=0,7
   wsum=wsum+round(n)
 end do
 if (wsum == 0) then
   p(z,1)=k; p(z,2)=l
   x(k,l)=BND; z=z+1
   Frg=1
   entry= 0
 end if

end do

end do
n_data=z
end subroutine

!-------------------------
!    calculating centroid of spot
subroutine centroid(b,bn,n,Gx,Gy,t)  ! b:画像, bn:輪郭, n:データ数, G:重心
  real*4 :: b(1:800,1:950)
  real*4 :: Gx,Gy, sumb,Mx,My,t,area
  integer :: bn(1:800,1:2),i,j,k,n,x1,x2,y1,y2
  integer :: h1,h2,h3,h4,h, ip,jp
  integer ::x(1:200),y(1:200), i_max,i_min,j_max,j_min

    ! パラメー i,j の斑点重心計算のための捜査範囲 i_max,i_min,j_max,j_min
  i_max=bn(1,1); i_min=bn(1,1); j_max=bn(1,2); j_min=bn(1,2)
  do i=2,n
    if (bn(i,1).gt.i_max) then
       i_max=bn(i,1)
    end if
    if (bn(i,1).lt.i_min) then
       i_min=bn(i,1)
    end if
    if (bn(i,2).gt.j_max) then
       j_max=bn(i,2)
    end if
    if (bn(i,2).lt.j_min) then
       j_min=bn(i,2)
    end if
  end do
```

```
! 初期化
  sumb=0.0; Mx=0.0; My=0.0; area=0.0
!  loop j
  do j=j_min,j_max
    do i=i_min,i_max
      ! 画素位置 (i,j) を選択

      h1=0; h2=0; h3=0; h4=0 ! h 判定フラグ

      do ip=i,i_min,-1
       do k=1,n
       if ((ip.eq.bn(k,1)).and.(j.eq.bn(k,2))) then !左一致
        h1=1
       end if
       end do
      end do

      do ip=i,i_max,1
       do k=1,n
       if ((ip.eq.bn(k,1)).and.(j.eq.bn(k,2))) then !右一致
        h2=1
       end if
       end do
      end do

      do jp=j,j_min,-1
       do k=1,n
       if ((i.eq.bn(k,1)).and.(jp.eq.bn(k,2))) then !上一致
        h3=1
       end if
       end do
      end do

      do jp=j,j_max,1
       do k=1,n
       if ((i.eq.bn(k,1)).and.(jp.eq.bn(k,2))) then !下一致
        h4=1
       end if
       end do
      end do

      h=h1*h2*h3*h4
      if (h.eq.1) then !斑点に含まれる (i,j) なら重み付き計算
        sumb=sumb+b(i,j); Mx=Mx+b(i,j)*float(i); My=My+b(i,j)*float(j); area=area+1
      end if
    end do
!  loop j
  end do

  Gx=Mx/sumb; Gy=My/sumb; i=int(Gx); j=int(Gy)
  t=sumb/area
end subroutine centroid
```

P1_P2.f90

P1_P2.f90 は，P1, P2 の解析結果から DEM の計算を行うプログラムである．**P1 _ spot.dat** および **P2 _ spot _ *i*.dat** を読込ながら，可能性のある斑点の対に対して計算結果を **DEM.dat** に出力す

る．このデータから，DEM の計算結果を総合的に判断して抽出する．具体的には，回折位置が入射ビーの光路であること，回折強度が大きいこと，格子面間隔が合理的であることなどから判断する．

```
!****************************************************************
!        Double Exposure Method for White X-rays, DEM_WX
!            P1_P2.f90        by Kenji Suzuki
!---------------------------------------------- December, 20 2018

program P1_P2
 implicit none
  character*30 filename, a
  integer keV,ln,tn
  double precision x1,y1,t1,x2,y2,Px(1:100),Py(1:100),Pr(1:100),Pt(1:100)
  double precision P1x,P1y,P2x,P2y,ell, lambda,d, Er
  double precision :: P1z=0.0,P2z=500.0,Lz2=500.0,Lz1=0.0
  double precision  Lx1,Ly1,Lx2,Ly2,alpha1,alpha2,th2,Pcx,Pcy,Pcz
  integer i,j,k,l,m,n,loop,ns
  integer ::counts=0
 double precision :: pi=3.1415926, hc=12.39842

 tn=10 ! 対象点３つ
 ell=Lz2-Lz1
counts=0
print *,'Do you prepare files of P1_spot.dat and spotP2_i.dat in the folder DEM-WX? (ret)'
read(5,*) a
   open (unit=12, file='dem.dat', status='unknown')
! P1_spot.dat を開いて，読み込む
   open (unit=13, file='P1_spot.dat', status='old')
    read(13,*) ns !number of spots
   do i=1,ns
     read(13,*) k,keV,ln,x1,y1,t1

     call mapping(x1,y1,x2,y2)   ! P2 の該当画像位置を計算

     if (((x2.gt.1.0).and.(x2.lt.800.0)).and.((y2.gt.1).and.(y2.lt.950.0))) then !  写像範囲の確認
       call P2spot(keV,x2,y2,Px,Py,Pr,Pt,tn)  !対応する斑点位置 (Px,Py) を探す
        counts=1+counts
print *,i,keV,t1
       write(12,604) counts,keV,t1
       do j=1,tn
! print 602,Px(j),Py(j),Pr(j),Pt(j)
        !Laboratory coordinates: Lx1,Ly1,Lz1,Lx2,Ly2,Lz2
        Lx1=(x1 - 46.0)*0.2; Ly1=(288. - y1)*0.2 !; Lz1=0.0
        Lx2=(Px(j) + 402. - 49.0)*0.2; Ly2=(471. - Py(j))*0.2!; Lz2=500.0

        call DEM(Lx1,Ly1,Lx2,Ly2,ell,alpha1,alpha2,th2,Pcx,Pcy,Pcz)
         lambda= hc/float(keV); d=lambda/2.0/sin(th2/2.0) ; th2=th2*180.0/pi
         alpha1=alpha1*180.0/pi; alpha2=alpha2*180.0/pi
         Er=sqrt(Pcx**2+Pcy**2)
print 603,lambda, alpha1,alpha2,th2,d,Pcx,Pcy,Pcz, Er,Pt(j) ! deg, deg, A, mm, mm, mm
        write(12,605) lambda, alpha1,alpha2,th2,d,Pcx,Pcy,Pcz,Er,Pt(j)
       end do
     end if

   end do !do i=1,ns
   close(13)
```

```
  close(12)

601 format(I4,4(F8.3,2x))
602 format(4(F8.3,2x))
603 format('λ'F8.5,' φ',F5.1,1X,F5.1,'  2θ',F8.4,'  d',F9.6,2X,3(F8.3,2X),' err',F8.3,F10.1)
604 format(I4,I4,X,F8.1)
605 format(F8.5,2(2X,F5.1),2X,F8.4,2X,F9.6,2X,3(F8.3,2X),F8.3,F10.1)
! P1(x1,y1,z1), P2(x2,y2,z2) から，実験室座標系に変換して，回折角を求める.

end program P1_P2

!***************** SUBROUTINE SUBROUTINE SUBROUTINE SUBROUTINE *********
!---------- mapping -------------------------
subroutine mapping(x1,y1,x2,y2) !P1(x1,y1) を P2(x2,y2) へ変換
  implicit none
  double precision x1,y1,x2,y2
  double precision:: x0=200.0, y0=150.0, cx=66.0,cy=92.0, Mg=2.75

   x2=Mg*(x1-x0)+cx
   y2=Mg*(y1-y0)+cy
end subroutine mapping

!---------- P2spot -------------------------
subroutine P2spot(keV,x2,y2,Px,Py,Pr,Pt,tn) ! P1 のスポットに該当する斑点群を探す
 implicit none
 character*30  filename
 double precision x1,y1,t1,x2,y2,Px(1:100),Py(1:100), x(1:300),y(1:300),t(1:300)
 double precision r(1:300), rs, Pt(1:100),Pr(1:100)
 integer keV,tn,i,j,k, n(1:300),ns

 do i=1,10 ! 初期化
  Px(i)=0.0; Py(i)=0.0; Pt(i)=0.0
 end do

 write (filename,*) keV     !keV の値を代入
 filename=trim('spotP2_'//adjustl(filename))//'.dat' !spotP2 + 数字 + keV.dat を加える.
 open(11,file=filename,status='old')
  read (11,*) ns
   do i=1,ns
    read(11,*) j,x(i),y(i),t(i) ! P2 の斑点位置と高さを読込む
    r(i)=sqrt((x(i)-x2)**2+(y(i)-y2)**2) !距離の計算
   end do
 close(11)
!                並び替え
 do i=ns-1,1,-1
  do k=1,i
   if (r(k).gt.r(k+1)) then
     rs=r(k+1); r(k+1)=r(k); r(k)=rs
     rs=x(k+1); x(k+1)=x(k); x(k)=rs
     rs=y(k+1); y(k+1)=y(k); y(k)=rs
     rs=t(k+1); t(k+1)=t(k); t(k)=rs
   end if
  end do
 end do

 do i=1,tn
  Px(i)=x(i); Py(i)=y(i); Pt(i)=t(i); Pr(i)=r(i)
 end do
```

```
end subroutine P2spot

!---------- DEM -------------------------
subroutine DEM(x1,y1,x2,y2,L,alpha1,alpha2,th2,xc,yc,zc)
 implicit none
 double precision x1,y1,x2,y2,L,alpha1,alpha2,th2,xc,yc,zc,r1,r2,a

 r1=sqrt(x1**2+y1**2); r2=sqrt(x2**2+y2**2)
 th2=((r2-r1)/L) ! radian
 a=(x2-x1)*x1+(y2-y1)*y1
 xc=x1-a*(x2-x1)/L/L/tan(th2)/tan(th2)
 yc=y1-a*(y2-y1)/L/L/tan(th2)/tan(th2)
 zc=-a/L/tan(th2)/tan(th2)
 alpha1=atan(y1/x1); alpha2=atan(y2/x2)! radian

end subroutine DEM
```