```
import sys,os,math
import iotbx.mtz
from numpy import *
import datetime

sys.path.append("/Users/kuntaro/00.Develop/Prog/02.Python/Libs/")
from ReflWidthStill import *
from ReadMtz import *
from DetectorArea import *
from Qshell import *
from GaussFitXY import *

class ProfileMaker:

        def __init__(self,still_mtz):
                self.still_mtz=still_mtz

        def init(self):
                ## Open MOSFLM MTZ file
                self.smtz=ReadMtz(self.still_mtz)
                self.smtz.getSymmOption()

                ## Extract intensity related cctbx.array
                self.stiI = self.smtz.getIntensityArray()

                ## M/ISYMM
                self.s_isyms=self.smtz.getColumn("M_ISYM").data()%256

                ## resolution
                self.s_d=self.stiI.d_spacings().data()

                ## Batch number
                self.s_ba=self.smtz.getColumn("BATCH").data()

                # Detector area
                self.s_xa=self.smtz.getColumn("XDET").data()
                self.s_ya=self.smtz.getColumn("YDET").data()

                # If neede, make it possible 140129 KH
                # Detector area setting
                #self.da=DetectorArea(3072,8,4)
                #self.da.init()

                print "%10d reflections were read from %s"%(len(self.s_ba),self.still_mtz)
                self.nrefl=len(self.s_ba)

        def isSameRefl(self,i1,i2):
                # HKL information of the first index
                hkl1=self.HKL[i1]
                isym1=self.ISYM[i1]
                # HKL information of the second index
                hkl2=self.HKL[i2]
                isym2=self.ISYM[i2]

                if hkl1==hkl2 and isym1==isym2:
                        return True
                else:
                        return False

        def prepInfo(self,matfile,startphi=35.0,stepphi=0.1,
                wl=1.24,divv=0.02,divh=0.02,mosaic=0.3,dispersion=0.0002):

                # Required class for RLP coodrinate calculation
                rws=ReflWidthStill(matfile,divv,divh,mosaic,dispersion,wl)

                # PHISTART and PHISTEP
                phi0=startphi

                # List of parameters
                self.HKL=[]
                self.Q=[]
                self.RLP=[]
                self.PHI=[]
                self.I=[]
```

MTZファイルから反射情報を抜き取るクラス関数

同じ反射かどうかを確認する（単純に反射インデックスを受け取って共通数値配列にアクセスする）

```
                self.SIGI=[]
                self.ISYM=[]
                self.BATCH=[]

                idx=0
                for (hkl1,sI,ssigI),isym,batch,d in zip(self.stiI,
                                                        self.s_isyms,self.s_ba,self.s_d):

                        # Initial batch number
                        if idx==0:
                                batch0=batch

                        # Convertion HKL -> original HKL in MOSFLM
                        ohkl=self.smtz.getOriginalIndex(hkl1,isym)
                        self.HKL.append(ohkl)
                        self.I.append(sI)
                        self.SIGI.append(ssigI)
                        self.ISYM.append(isym)
                        self.BATCH.append(batch)

                        # Rotation angle
                        phi1=phi0+(batch-batch0)*stepphi
                        self.PHI.append(phi1)

                        # Parameters
                        oh,ok,ol=ohkl
                        rws.setHKL(ohkl,phi1)
                        rws.calcDELEPS()

                        # RLP coordinate
                        rlp=rws.getRLP()
                        self.RLP.append(rlp)

                        # Q calculation
                        q=rws.calcQ()
                        self.Q.append(q)

                        idx+=1

                print "Processed %5d reflections"%idx

        def bunch(self):
                # output file
                ofile=open("data.dat","w")

                # Working list
                lwork=[]

                # Initial condition
                save_i=0

                # Count reflections
                n_alone=0

                # Processing
                for i in range(1,self.nrefl):
                        # check if saved reflection and this one is 'same' reflection
                        # (not including 'equivalent'

                        # DEBUGGING
                        #print i,self.HKL[i],self.ISYM[i]

                        if self.isSameRefl(i,save_i):
                                lwork.append(i)
                        else:
                                if len(lwork)==1:
                                        print "HKL is one",save_i,self.HKL[save_i]
                                        n_alone+=1

                                # Reflection which fills conditions to estimate
                                # intensity profile
                                else:
                                        self.makeProfile(lwork)
                                        #print lwork
```

クラスに共通な配列へ数値を格納
基本的に複雑な計算は
ReflWidthStillへ投げている。

同じ反射かどうかを確認してプロファイル作成用の配列を作っている。lworkはループを回すための配列で同じ反射が来ている間はそのインデックスを格納していく。
現状では試験的にガウス関数へのフィッティングを入れている。

同じ反射ならばlworkに一時的に保存

反射が１個しか無かった時→処理は不可能

反射が二つ以上あった時は処理に進む。

```
                                save_i=i
                                lwork=[]
                                lwork.append(i)

                print "%10d reflections are rejected because observation was once"%n_alone

        def makeProfile(self,iwork):
                xlist=[]
                ylist=[]

                index=iwork[0]

                # if a number of reflections is 2
                # Gauss fitting is not conducted
                if len(iwork) <= 2:
                        print "Gaussian fitting cannot be done!"
                        return

                for i in iwork:
                        xlist.append(self.Q[i])
                        ylist.append(self.I[i])

                # Gaussian fitting
                g=GaussFitXY(xlist,ylist)
                prefix="test_%05d"%index
                pngfile=prefix+".png"
                logfile=prefix+".log"
                g.fit(pngfile,logfile,prefix)

if __name__ == "__main__":

        matfile="still_10.mat"
        divv=0.02
        divh=0.02
        mosaic=0.3
        dispersion=0.0002

        # ARG1 = MOSFLM MTZ file
        h=ProfileMaker(sys.argv[1])
        h.init()
        h.prepInfo(matfile,startphi=35.0,stepphi=0.1)
        h.bunch()
```

プロファイルを作る関数。
難しい計算はGaussFitXYにまかせている。

反射が二つしかない場合にはフィッティングができないので、ムリ！といって返すことにしている。fittingがかからなかった反射は数える方が良いかもね。