

```

# fichier créé avec python 3.9.1

import numpy as np

#####

def cMin(p, pMin: float) -> float:

    # contraint le minimum d'un paramètre p en évitant d'influencer les autres
    cM = pMin + abs(p - pMin)
    return cM

#####

def cMax(p, pMax: float) -> float:

    # contraint le maximum d'un paramètre p en évitant d'influencer les autres
    cM = pMax - abs(pMax - p)
    return cM

#####
#####
# prédéfini pour la "compilation", mais doit être redéfini par l'utilisateur
# selon le modèle étudié (ici elliptique en coordonnées polaires)

def fonc(NPar: int, p, x: float) -> float: # p devrait être un array(30) float

    # définit la fonction théorique ajustée sur les données expérimentales

    # les paramètres utilisés dans l'expression ajustée sont ici notés p[i]
    # des noms plus explicites sont définis dans le programme appelant minimi,
    # mais non utilisés ici
    # (ceci n'a rien d'obligatoire, fonc n'est utilisé que par FCN qui peut être
    # écrit autrement)

    # le nombre maximum de paramètres est fixé à 30, ceci est imposé dans minimi
    # le nombre de paramètres effectivement actifs est NPar, les suivants sont
    # ignorés
    # (NPar n'est transmis ici que pour vérification éventuelle)

    # l'abscisse est notée x

    f: float = np.nan
    f = p[0] / (1 + p[1] * np.cos(x - p[2])) # l'indexation python commence à 0
    return f

#####
#####

# il faut définir (ou importer) ici les points expérimentaux
# (abscisses x et ordonnées y) et leurs incertitudes

```

```

Npts = 10 # nombre de points
xPts = np.array([0.1,0.5,1.,2.,3.4,4.1,5.,5.4,5.8,6.1])
yPts = np.array([14.2,7.3,5.,2.5,2.,3.,5.5,8.,16.,21.])
dxPts = np.array([0.1,0.2,0.2,0.3,0.1,0.3,0.2,0.2,0.1,0.1])
dyPts = np.array([0.4,1.1,0.8,0.2,0.5,0.6,0.7,0.2,0.9,0.3])

autres = [Npts, xPts, yPts, dxPts, dyPts]
# devrait être Npts: int, puis array(Npts) float
# Dans le cas général, minimi ne sait pas a priori de quoi peut dépendre FCN
# à part les paramètres ;
# ces quantités, si elles existent, sont de type variable donc
# transmises en argument comme structure à décoder lors de l'utilisation

#####
#####
# prédéfini pour la "compilation" (ici un chi2)...
# ...mais peut être redéfini par l'utilisateur

def FCN(NPar: int, xp, autres) -> float: # xp devrait être un array(30) float
# autres est une structure contenant ce dont FCN peut dépendre en plus
# minimi est compilée sans en connaître le contenu, que seule FCN utilise

# définit la fonction minimisée (généralement un chi2)

# le nom FCN est réservé par minimi
# le nom myFCN est réservé par minimi

Npts, xPts, yPts, dxPts, dyPts = autres # décodage de la structure transmise
f: float = 0.
# reste à définir la fonction fonc(NPar, params, abscisse) qui décrit le
# modèle utilisé

for ii in range(Npts): # l'indexation commence à 0
# on ajoute ici quadratiquement l'incertitude de l'ordonnée des points
# expérimentaux et la propagation sur l'ordonnée théorique de
# l'incertitude de l'abscisse des points expérimentaux (en supposant que
# les abscisses et les ordonnées sont des mesures indépendantes)
der=(fonc(NPar,xp,xPts[ii]+dxPts[ii]) - fonc(NPar,xp,xPts[ii]-dxPts[ii]))/2
ddd = dyPts[ii]**2 + der**2
f = f + ((yPts[ii]-fonc(NPar,xp,xPts[ii]))**2)/ddd
return f

```