Zadani2024

May 18, 2024

1 Zápočet MMZM 2024

1.0.1 Část 1

V souboru `pikyXY.dat` jsou zadány naměřené polohy píků ze sady spekter gama zdrojů při různých teplotách T a operačním napětí V. Na každém řádku je také uvedena odpovídající energie E dané čáry v kiloelektronvoltech. Poloha píků C a jejich nejistota σC je udávána v jednotkách ADC převodníku (číslo kanálu) označených ADU. Úkolem je najít parametry kalibrační funkce, tedy především kombinovaný zisk (gain) $C = f(E, T, V) = C_0 + E * g(T, V)$ pro 2 uvažované modely:

lineární

$$g(T, V) = p_0 + p_1\ T + p_2\ V$$

kvadratický

$$g(T, V) = p_0 + p_1\ T + p_2\ V + p_3\ T^2 + p_4\ V^2 + p_5 V\ T$$

Nafitujte tyto parametry spolu s určením nejistot a porovnejte tyto modely pomocí F-rozdělení poměru sumy čtverců residuí (stanovte, zda je kvadratický model vhodnější).

Berte do úvahy i zadané hodnoty nejistot měření (σC).

1.0.2 Část 2

Máte zadány 2 spektra v souboru `dvojiceXY.dat`, která v relevantní oblasti mají málo identifikovaných píků (vzhledem ke špatnému spektrálnímu rozlišení přístroje). Úkolem je nalézt poměr zisku těchto 2 měření, tedy naškálování x-ové osy (x_1 a x_2), které vede k nejvyšší možné korelaci ρ hodnot $\ln y_1$ a $\ln y_2$ (vzhledem k velkému dynamickému rozsahu porovnávejte hodnoty v logaritmickém měřítku). Pro přeškálování lze použít buď lineární interpolace hodnot (kód pro numerický python níže), nebo hrubší přístup založený na tvorbě histogramů s posunutými biny (druhá část kódu).

Odhadněte i nejistotu takto určeného poměru z (numerické) druhé derivace závislosti $1 - \rho$ na poměru přeškálováni r.

"Relevantní oblast" znamená vynechání prvního píku, který odpovídá temnému proudu v detektoru a neposouvá se s operačním napětím.

[ ]:
```
#interpolace
import numpy as np
from scipy import interpolate as ip

prof1=ip.interp1d(np.r_[:len(y1)],y1)
```

```
jmin,jmax=0,80
y1_resc=prof1(np.r_[jmin:jmax:1j*len(y2)])

### histogram
def binme(values,bins):
    rep=[]
    for i in range(sum(bins<len(values))-1):
        blen=bins[i+1]-bins[i]
        bsum=sum(vls[int(bins[i]):int(bins[i+1])])
        bsum+=(bins[i+1]-int(bins[i+1]))*values[int(bins[i+1])]
        bsum-=(bins[i]-int(bins[i]))*values[int(bins[i])]
        rep.append(bsum/blen)
    return np.array(rep)
nbins=16
y2_rebin=binme(y2[:181],np.r_[:180:1j*nbins])
y1_rebin=binme(y1[:180],np.r_[:151:1j*nbins])
```