```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as ss
import scipy.optimize as scp

exec(open('Kalibrering/kalibrering.py').read())
exec(open('../Scripts/Statistik.py').read())
exec(open('../Scripts/data_renser.py').read())
exec(open('../Scripts/chi_sq.py').read())

fig, ax = plt.subplots(figsize = (16,8))

sol1 = Data('Kalibrering/30grader')
spÃ¦nding = sol1.points
ts = sol1.t*1000

mask = sol1.rinse2(0.15, 0.02)
ts1 = ts[mask]

vink = vinkel(spÃ¦nding, *kali)*(360/(2*np.pi))

error = propagation_function(spÃ¦nding[mask], vinkel, list(kali), pcov)
vinks = vink[mask]


index = []
err = []
j = 0
k = 0

# Her bestemmer jeg toppunkterne

bool = True
while bool:
    vmax = max(vinks[j:])
    index_max = list(vinks[j:]).index(vmax)
    tp = []
    for i in [index_max + j for j in range(20)]:
        if vinks[j:][i] == vmax:
            tp.append(i+j)
    index.append(tp[int(round(len(tp)/2, 0))])
    if len(tp) == 1:
        err.append(0.0033)
    else:
        err.append((ts1[tp[-1]] - ts1[tp[0]])/4)
    j += index_max + 120
    if j > len(vinks):
        bool = False

def sinus(t, *p):
     A = p[0]
     w = p[1]
     k = p[2]
     d = p[3]
     return (A*np.cos(w*t+k)+d)

A = []
peri = []
peri_err = []

for i in range(len(index)-5):

    # Det her er noget rod... sorry...
```

```
    # Det gik lige lidt hurtigt, beklager at det er sÃ¥ uoverksueligt

    # Jeg starter med at finde de relevante vinkler, dem mellem to toppunkter

    vink = vinks[index[i]:index[i+1]]

    # afstand mellem toppunkter

    ts = ts1[index[i]:index[i+1]]

    # LÃ¦ngden pÃ¥ perioden

    per = (ts1[index[i+1]]-ts1[index[i]])

    # Perioden gemmes

    peri.append((ts1[index[i+1]]-ts1[index[i]]))

    # T_0

    peri_0 = ts1[index[-6]] - ts1[index[-7]]

    # Fejlpropageringen laves som beskrevet i Resultater

    peri_err.append(np.sqrt((err[i+1]**2 + err[i]**2)/peri_0**2 +
       per**2*1/(peri_0)**4*(err[30]**2 + err[29]**2)))
    guess = [50-i*2,-5,2,0]

    # Amplituden findes med curve_fit og gemmes

    popt, pcov2 = scp.curve_fit(sinus, ts, vink, guess,
                                sigma = error[index[i]:index[i+1]], absolute_sigma = True)

    A.append(abs(popt[0]))

# Perioderne plottes med error bars. Jeg har undladt de fÃ¸rste og sidste
# punkter, da disse ikke giver mening da der enten er for meget stÃ¸j eller
# Pendulet er meget dÃ¦mpet.

ax.errorbar(A[1:-3], peri[1:-3]/peri[-4], yerr = peri_err[1:-3], fmt = 'o')

print(peri[-3])

A = np.array(A)*2*np.pi/360
peri = np.array(peri)
peri_err = np.array(peri_err)

# Funktion fittes

def fitter(A, n):
     ns = sum([(np.math.factorial(2*n)/(2**n*np.math.factorial(n))**2)**2*np.sin(A/2)**(2*n)
           for n in range(n)])
     return ns

# Chi-2 test. Funktionen er defineret i Scripts/

chisq, pval = chi_sq(x = peri[1:-3]/peri[-4],t = A[1:-3],
                     err = peri_err[1:-3], f = fitter,
                     popt = [5], df = len(A[1:-3]))

print(chisq, pval)

ttss = np.linspace(0, 0.25*np.pi, 100)
```

```
ax.plot(np.linspace(0, 45, 100), fitter(ttss, 10))


ax.set_title('$\chi_m$ = {}'.format(round(chisq, 3))
             + '      $p$ = {}'.format(round(pval, 3)), fontsize = 20)

ax.set_xlabel('Vinkel', fontsize = 16)
ax.set_ylabel('T', fontsize = 16)
ax.legend()

fig.savefig('Ampl_Peri')

plt.show()
```