```python
# -*- coding: utf-8 -*-

import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize as so


def normPoly(x,y,N,nparam):
    global noise
    noise = np.zeros((len(N),len(N)))
    for i in range(len(N)):
        noise[i,i] = N[i]**2

    A = np.zeros((len(x),nparam))
    for i in range(len(x)):
        for j in range(nparam):
            A[i,j] = x[i]**(nparam-1-j)

    ATA = (np.transpose(A) @ noise) @ A
    ATAinv = np.linalg.inv(ATA)
    ATb = np.dot((np.transpose(A) @ noise),y)
    params = np.dot(ATAinv,ATb)

    return params,ATAinv


def norm2(x,y,N,order):
    order = int(order)
    noise = np.zeros((len(N),len(N)))
    for i in range(len(N)):
        noise[i,i] = N[i]**2

    A = np.zeros((len(x),2*order+1))
    for i in range(len(x)):
        for j in range(-order,order+1):
            A[i,-1*j+order] = x[i]**j
    ATA = (np.transpose(A) @ noise) @ A
    ATAinv = np.linalg.inv(ATA)
    ATb = np.dot((np.transpose(A) @ noise),y)
    params = np.dot(ATAinv,ATb)

    return params,ATAinv




"""
Part 1
Generic polynomial function
y = a1*x + a0 for example
"""


#Defining variables
ai = [5,7]
```

```
nparams = len(ai)
n=100
stdev=1
noise = np.random.normal(0,stdev,n)



#Polynomial function
def f1(x,*params):
    l = []
    N = len(params)
    for i,a in enumerate(params):
        l.append(a*x**(N-i-1))
    return sum(l)

x = np.linspace(1,100,100)
y = [f1(a,*ai) for a in x]


#Fix the noise to have a SNR of 3 at the last point
ratio = abs(y[-1]/noise[-1])
print('Part 1: Linear Case')
print(f'Old SNR: {ratio}')
scaling = 3/ratio

noise = [a/scaling for a in noise]

#Signal + Noise
mixed = [a+b for a,b in zip(y,noise)]


print(f'New SNR: {abs(y[-1]/noise[-1])}')

uncertainty = [stdev/scaling]*n

#Fitting with my method and with scipy
canned,cvar = so.curve_fit(f1,x,mixed,p0=[1]*nparams,sigma=uncertainty)
coeff,var = normPoly(x,mixed,uncertainty,nparams)
fit = [f1(a,*coeff) for a in x]
cfit = [f1(a,*canned) for a in x]


fstr = "My Fit: "
cstr = "Scipy: "
rstr = "Real: "

#Label strings of variable lengths
for i in range(nparams-1,-1,-1):
    fstr += f"\na{i}={coeff[nparams-1-i]:.2f}±{np.sqrt(var[nparams-1-i,nparams-1-i]):.2f}"
    cstr += f"\na{i}={canned[nparams-1-i]:.2f}±{np.sqrt(cvar[nparams-1-i,nparams-1-i]):.2f
    rstr += f"\na{i}={ai[nparams-1-i]:.2f}, "



#Plot of the data
plt.figure()
plt.scatter(x,mixed,label='Signal+Noise')
```

```
plt.plot(x,y,color='orange',label=rstr)
plt.title('Polynomial Data SNR=3')
plt.xlabel('Time')
plt.ylabel('Signal')
plt.legend(loc='upper left')
plt.savefig('polynomial.pdf')


#Plot of the fit
plt.figure()
plt.errorbar(x,mixed,uncertainty,fmt='o',label='Signal+Noise')
plt.plot(x,fit,color='red',label=fstr)
plt.plot(x,cfit,'--g',label=cstr)
plt.title('Polynomial Fit')
plt.xlabel('Time')
plt.ylabel('Signal')
plt.legend(loc='upper left')
plt.savefig('polynomialFit.pdf')

#Chi-squared
chi = []
for i in range(len(x)):
    chi.append( (mixed[i]-fit[i])**2/(2*uncertainty[i]**2) )
chi2 = sum(chi)
print(f"Chi-squared: {chi2}")



"""
Part 2
Nonlinear function
y = a2*x + a1 + a0/x for example
"""

ai = [3,5,7,200,1000]
order = int((len(ai)-1)/2)
n=100
stdev=1
noise = np.random.normal(0,stdev,n)

def f2(x,*params):
    l = []
    N = len(params)
    order = int((N-1)/2)
    for i in range(order,-1*order-1,-1):
        j = -1*i + order
        l.append(params[j]*x**i)
    return sum(l)


x = np.linspace(1,10,100)
y = [f2(a,*ai) for a in x]



#Fix the noise to have a SNR of 3 at the last point
ratio = abs(y[-1]/noise[-1])
```

```
print('Part 2: Nonlinear Case')
print(f'Old SNR: {ratio}')
scaling = 10/ratio

noise = [a/scaling for a in noise]

#Signal + Noise
mixed = [a+b for a,b in zip(y,noise)]


print(f'New SNR: {abs(y[-1]/noise[-1])}')

uncertainty = [stdev/scaling]*n

#Fitting with my method and with scipy
canned,cvar = so.curve_fit(f2,x,mixed,p0=[1]*len(ai),sigma=uncertainty)
coeff,var = norm2(x,mixed,uncertainty,order)
fit = [f2(a,*coeff) for a in x]
cfit = [f2(a,*canned) for a in x]


fstr = "My Fit: "
cstr = "Scipy: "
rstr = "Real: "

#Label strings of variable lengths
for i in range(int(order),-1*int(order)-1,-1):
    j = -1*i+order
    fstr += f"\na{i}={coeff[j]:.2f}±{np.sqrt(var[j,j]):.2f}"
    cstr += f"\na{i}={canned[j]:.2f}±{np.sqrt(cvar[j,j]):.2f}"
    rstr += f"\na{i}={ai[j]:.2f}\n "


#Plot of the data
plt.figure()
plt.scatter(x,mixed,label='Signal+Noise')
plt.plot(x,y,color='orange',label=rstr)
plt.title('Nonlinear Data SNR=10')
plt.xlabel('Time')
plt.ylabel('Signal')
plt.legend(loc='upper right')
plt.savefig('nonlinear.pdf')


#Plot of the fit
plt.figure()
plt.errorbar(x,mixed,uncertainty,fmt='o',label='Signal+Noise')
plt.plot(x,fit,color='red',label=fstr)
plt.plot(x,cfit,'--g',label=cstr)
plt.title('Nonlinear Fit')
plt.xlabel('Time')
plt.ylabel('Signal')
plt.legend(loc='upper right')
plt.savefig('nonlinearFit.pdf')


#Chi-squared
```

```
chi = []
for i in range(len(x)):
    chi.append( (mixed[i]-fit[i])**2/(2*uncertainty[i]**2) )
chi2 = sum(chi)
print(f"Chi-squared: {chi2}")
```