Physics 281 - Computational Physics

Wednesday/Friday Section

Fall 2015

Exercise 10 - Monte Carlo Error Simulation of Least Squares Fit

Here is the script

```
# Exercise 10
import numpy as np
import math
import matplotlib.pyplot as pl
from numpy.random import RandomState

def fit_linear_model(X,Y):
    # linear fit to polynomial, given X,Y arrays
    npar = 3

    # create M - ones in first col, X in second col.
    M = np.column_stack( (np.ones(nobs),X,X**2) )

    # solve
    MTM = np.dot(M.transpose(),M)
    MTMINV = np.linalg.inv(MTM)
    MTY = np.dot(M.transpose(),Y)

    P = np.dot(MTMINV,MTY) # SOLUTION

    # compute RMS and print it
    Residuals = Y - np.dot(M,P)
    ChiSq = np.dot(Residuals.transpose(),Residuals)
    RMS = math.sqrt(ChiSq/nobs)

    C = MTMINV * ChiSq/(nobs-npar)  # COVARIANCE MATRIX
    # return parameters, RMS of Fit, Covariance Matrix, and ChiSq
    return P, RMS, C, ChiSq


# anonymous function to provide the true polynomial
f = lambda x: 5.+3.*x+2.*x**2
# random state object to produce random numbers
r = RandomState()

# X positions of data - same X used for all trials
X = np.linspace(0., 20., 21)
nobs = len(X)

ntrials = 10000
# define arrays to hold results of each trial
P0 = np.zeros(ntrials)
P1 = np.zeros(ntrials)
P2 = np.zeros(ntrials)
E0 = np.zeros(ntrials)
E1 = np.zeros(ntrials)
E2 = np.zeros(ntrials)
ChiSq = np.zeros(ntrials)

# now we do a loop to carry out experiment ntrials times
for itrial in range(ntrials):

    # Y positions of data follow function f with random error
    Y = f(X) + r.randn(nobs)

    P, RMS, C, CC  = fit_linear_model(X,Y)

    P0[itrial] = P[0]
    P1[itrial] = P[1]
    P2[itrial] = P[2]
    E0[itrial] = math.sqrt(C[0,0])
    E1[itrial] = math.sqrt(C[1,1])
    E2[itrial] = math.sqrt(C[2,2])
    ChiSq[itrial] = CC

# print out some results
print '%d Monte Carlo Trials'%(ntrials)
print 'Parameter 0: Average Error=%10.5f MC Result=%10.5f Ratio=%5.3f'%(E0.mean(),P0.std(),E0.mean()/P0.std())
print 'Parameter 1: Average Error=%10.5f MC Result=%10.5f Ratio=%5.3f'%(E1.mean(),P1.std(),E1.mean()/P1.std())
print 'Parameter 2: Average Error=%10.5f MC Result=%10.5f Ratio=%5.3f'%(E2.mean(),P2.std(),E2.mean()/P2.std())

# compute Reduced ChiSq for each trial (3 parameters in fit)
RChiSq = ChiSq/(nobs-3)

# plot the derived parameters against each other to show correlations
pl.ion()
pl.close('all')
pl.figure(1)
pl.subplot(2,2,1)
pl.plot(P0,P1,'.')
pl.plot([5,5],[0,10],'k')
pl.plot([0,10],[3,3],'k')
pl.axis([P0.min()-P0.std(),P0.max()+P0.std(),P1.min()-P1.std(),P1.max()+P1.std()])
pl.xlabel('P0')
pl.ylabel('P1')
pl.subplot(2,2,2)
pl.plot(P1,P2,'.')
pl.plot([3,3],[0,10],'k')
```

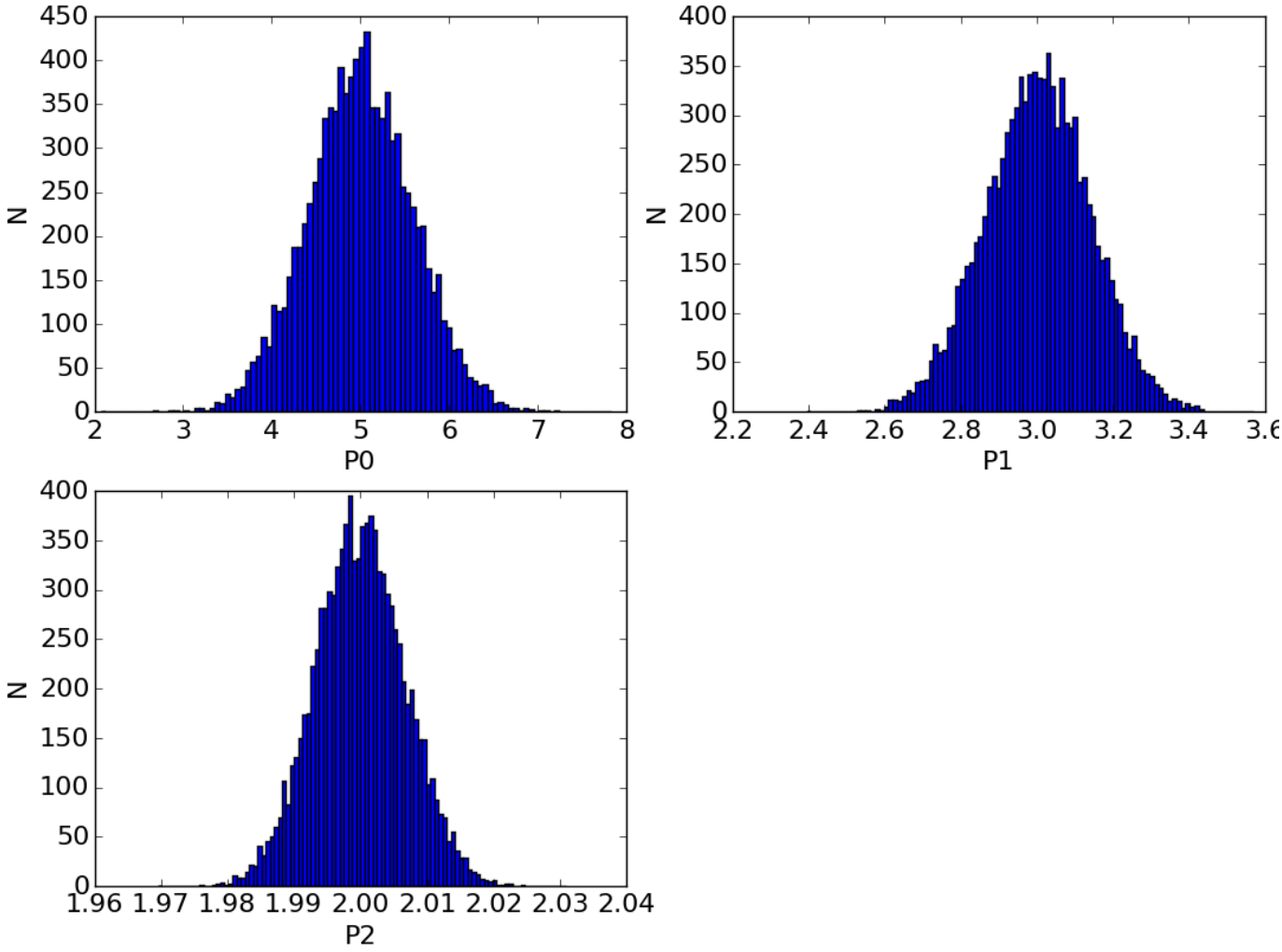
```
pl.plot([0,10],[2,2],'k')
pl.axis([P1.min()-P1.std(),P1.max()+P1.std(),P2.min()-P2.std(),P2.max()+P2.std()])
pl.xlabel('P1')
pl.ylabel('P2')
pl.subplot(2,2,3)
pl.plot(P0,P2,'.')
pl.plot([5,5],[0,10],'k')
pl.plot([0,10],[2,2],'k')
pl.axis([P0.min()-P0.std(),P0.max()+P0.std(),P2.min()-P2.std(),P2.max()+P2.std()])
pl.xlabel('P0')
pl.ylabel('P2')

# show histograms of the results for each parameter.
pl.figure(2)
P0bins = np.linspace(P0.min()-P0.std(),P0.max()+P0.std(),num=100,endpoint=True)
pl.subplot(2,2,1)
P0hist = pl.hist(P0,P0bins)
pl.xlabel('P0')
pl.ylabel('N')

P1bins = np.linspace(P1.min()-P1.std(),P1.max()+P1.std(),num=100,endpoint=True)
pl.subplot(2,2,2)
P1hist = pl.hist(P1,P1bins)
pl.xlabel('P1')
pl.ylabel('N')

P2bins = np.linspace(P2.min()-P2.std(),P2.max()+P2.std(),num=100,endpoint=True)
pl.subplot(2,2,3)
P2hist = pl.hist(P2,P2bins)
pl.xlabel('P2')
pl.ylabel('N')

# display distribution of reduced chi sq.
pl.figure(3)
RChiSqBins = np.linspace(0.,10.,101)
RChiSqHist,RChiSqEdges = np.histogram(RChiSq,RChiSqBins)
RChiSqCenters = np.zeros(100)
for i in range(100):
    RChiSqCenters[i] = (RChiSqEdges[i]+RChiSqEdges[i+1])/2.
pl.plot(RChiSqCenters,RChiSqHist/float(ntrials)*100.,'o-')
pl.xlabel('Reduced Chi Square')
pl.ylabel('Percentage')

# compute and display the cumulative distribution of reduced chi sq.
pl.figure(4)
RChiSqCum = np.zeros(101)
RChiSqCum[0] = 100.
for i in range(1,101):
    RChiSqCum[i] = RChiSqCum[i-1]-RChiSqHist[i-1]/float(ntrials)*100.
pl.plot(RChiSqEdges,RChiSqCum,'o-')
pl.xlabel('Reduced Chi Square')
pl.ylabel('Cumulative Percentage')
```

Here is output:

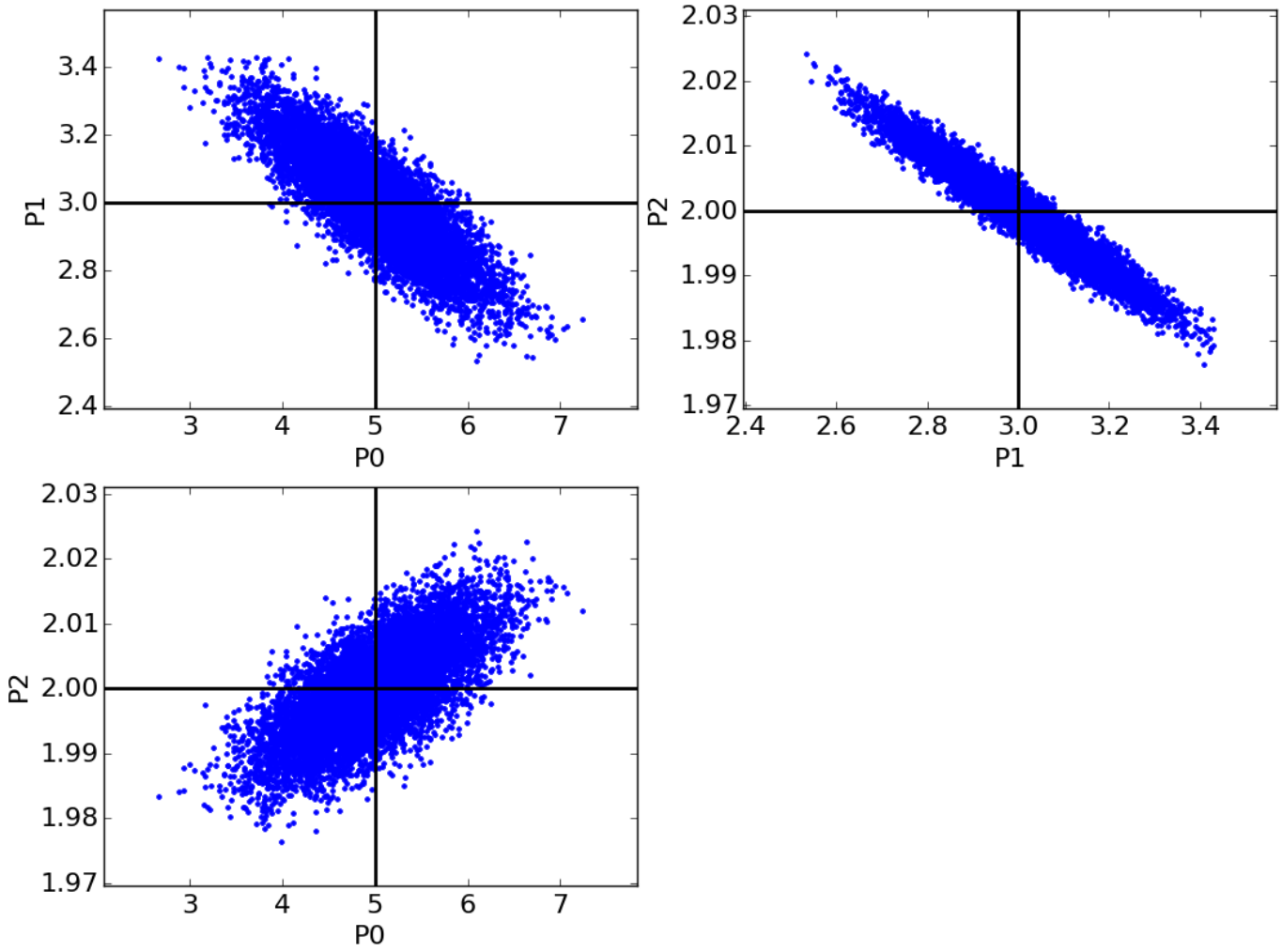
```
10000 Monte Carlo Trials
Parameter 0: Average Error=   0.58812 MC Result=   0.58976 Ratio=0.997
Parameter 1: Average Error=   0.13628 MC Result=   0.13696 Ratio=0.995
Parameter 2: Average Error=   0.00658 MC Result=   0.00663 Ratio=0.992
```

Here are the graphs. Note improvement as number of trials increases.

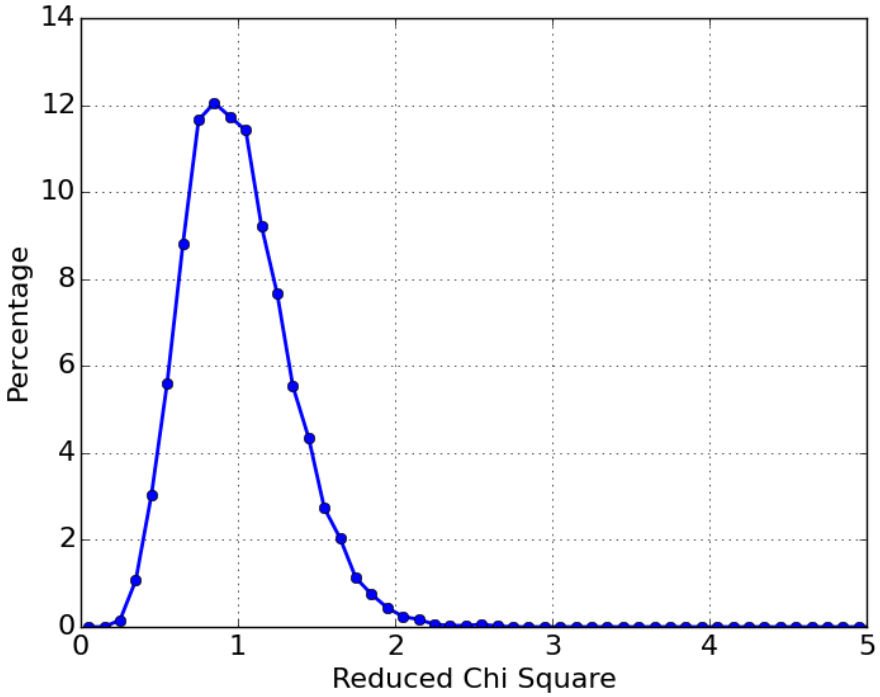
Histograms of the distribution of results for each parameter.



Plots of values of one parameter versus another. Note correlations between values obtained.



Here is a graph of the result of the distribution of reduced chi square. Note that the most likely value of the reduced chi square is about 1, with very little chance that the value will be larger than 2.



Here is a graph of the cumulative probability distribution of reduced chi square for our simulation. This shows the probability of obtaining a reduced chi square higher than a particular value in our experiment.