Eos491assignment02

December 4, 2015

EOS 491/526 Assignment #2
Daniel Scanks V00788200
Question #1

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        import matplotlib.mlab as mlab
        %matplotlib inline
        import numpy.linalg as ag

        #sensitivity matrix
        A = np.matrix([[1.,1.,1.,1.,1.],[1.,1.,1.,1.,0.5],[1.,1.,1.,0.5,0],[1.,1.,0.5,0,0],[1.,0.5,0,0,
        A_T = A.transpose()

        #true model
        m_t = np.matrix([1,2,3,4,5])
        m_t = m_t.transpose()


        #true data
        d_t = np.matrix([15.,12.5,8.,4.5,2.,0.5])
        d_t = d_t.transpose()

        #covariance and inverse covariance matrix
        Cd = np.matrix([[0.3**2,0,0,0,0,0],[0,0.2**2,0,0,0,0],[0,0,0.1**2,0,0,0],[0,0,0,0.1**2,0,0],[0,
        print('Cd')
        print Cd
        Cdinv = ag.inv(Cd)


        #part a)
        print('part a: blue plots')

        a=0
        m = np.zeros((10000,5))
        for a in range(10000):
            #noise (using given standard deviations: stddev)
            noise = np.zeros(6)
            stddev = np.array([0.3,0.2,0.1,0.1,0.05,0.05])
            for i in range(0, len(stddev)):
                sigma = np.random.normal(0,stddev[i])
                noise[i] = sigma
            noise = np.matrix(noise)
            noise = noise.transpose()
```

```
    d = d_t + noise #observed data
    dot = np.dot(A_T,Cdinv) #calculating ((A.Cdinv.A_t)^-1.A_t.Cdinv.d)
    dot2 = np.dot(dot,A)
    invdot2 = ag.inv(dot2)
    product = np.dot(invdot2,A_T)
    product2 = np.dot(product,Cdinv)
    M = np.dot(product2,d) #predicted model
    M = np.matrix(M)
    m[a,:] = M.transpose()
    #put models into rows for easy plotting (matrix of 10000 predicted models)

#C^m calculation (in for loop)
Cm = invdot2
print('Cm variances about true parameters')
print Cm


#Histograms(5 in total for each parameter of predicted model)

fig = plt.figure(figsize =(3,4))
ax=fig.add_subplot(1,1,1)
#plotting only first column of model matrix (1st value of each of 10000 predicted models)
ax.hist(m[:,0], 50, normed=True, histtype='stepfilled')
#theoretical curve using Cm variance
x = np.linspace(0.5,1.5,100)
sigma = np.sqrt(0.00971429)
ax.plot(x,mlab.normpdf(x,1,sigma), color = 'r', linewidth = 2, label = 'theoretical distributio
ax.set_title('Histogram for First Model Paramter for 10000 Inversions')
ax.legend(loc = 'upper left', prop={'size':8})

fig = plt.figure(figsize=(3,4))
ax=fig.add_subplot(1,1,1)
#plotting only second column of model matrix (2nd value of each of 10000 predicted models)
ax.hist(m[:,1], 50, normed=True, histtype='stepfilled')
#theoretical curve using Cm variance
x = np.linspace(1,3,100)
sigma = np.sqrt(0.04742857)
ax.plot(x,mlab.normpdf(x,2,sigma), color = 'r', linewidth = 2, label = 'theoretical distributio
ax.set_title('Histogram for Second Model Paramter for 10000 Inversions')
ax.legend(loc = 'upper left', prop={'size':8})

fig = plt.figure(figsize =(3,4))
ax=fig.add_subplot(1,1,1)
#plotting only third column of model matrix (3rd value of each of 10000 predicted models)
ax.hist(m[:,2], 50, normed=True, histtype='stepfilled')
#theoretical curve using Cm variance
x = np.linspace(2,4,100)
sigma = np.sqrt(0.10171429)
ax.plot(x,mlab.normpdf(x,3,sigma), color = 'r', linewidth = 2, label = 'theoretical distributio
ax.legend(loc = 'upper right', prop={'size':8})
ax.set_title('Histogram for Third Model Paramter for 10000 Inversions')

fig = plt.figure(figsize =(3,4))
ax=fig.add_subplot(1,1,1)
```

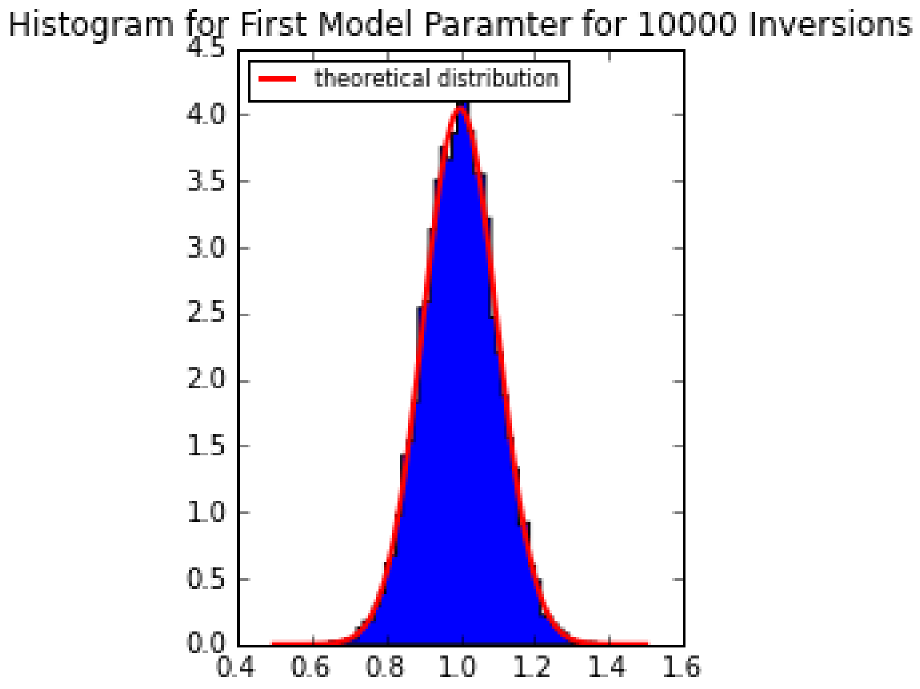
```
#plotting only fourth column of model matrix (4th value of each of 10000 predicted models)
ax.hist(m[:,3], 50, normed=True, histtype='stepfilled')
#theoretical curve using Cm variance
x = np.linspace(2.5,5,100)
sigma = np.sqrt(0.20685714)
ax.plot(x,mlab.normpdf(x,4,sigma), color = 'r', linewidth = 2, label = 'theoretical distributio
ax.legend(loc = 'upper left', prop={'size':8})
ax.set_title('Histogram for Fourth Model Paramter for 10000 Inversions')

fig = plt.figure(figsize =(3,4))
ax=fig.add_subplot(1,1,1)
#plotting only last column of model matrix (5th value of each of 10000 predicted models)
ax.hist(m[:,4], 50, normed=True, histtype='stepfilled')
#theoretical curve using Cm variance
x = np.linspace(3,7,100)
sigma = np.sqrt(0.18971429)
ax.plot(x,mlab.normpdf(x,5,sigma), color = 'r', linewidth = 2, label = 'theoretical distributio
ax.legend(loc = 'upper left', prop={'size':8})
ax.set_title('Histogram for Fifth Model Paramter for 10000 Inversions')




#b)
print('part b: green plots')

Cd = np.matrix([[0.1**2,0,0,0,0,0],[0,0.1**2,0,0,0,0],[0,0,0.1**2,0,0,0],[0,0,0,0.1**2,0,0],[0,
a=0
m = np.zeros((10000,5))
for a in range(10000):
    #noise (using given standard deviations: stddev)
    noise = np.zeros(6)
    stddev = np.array([0.3,0.2,0.1,0.1,0.05,0.05])
    for i in range(0, len(stddev)):
        sigma = np.random.normal(0,stddev[i])
        noise[i] = sigma
    noise1 = np.matrix(noise)
    noise = noise1.transpose()
    d = d_t + noise #observed data
    dot = np.dot(A_T,A)
    invdot = ag.inv(dot)
    product = np.dot(invdot,A_T)
    M = np.dot(product,d) #predicted model
    M = np.matrix(M)
    m[a,:] = M.transpose()
    #put models into rows for easy plotting (matrix of 10000 predicted models)

#Histograms(5 in total for each parameter of predicted model)

fig = plt.figure(figsize =(3,4))
ax=fig.add_subplot(1,1,1)
#plotting only first column of model matrix (1st value of each of 10000 predicted models)
ax.hist(m[:,0], 50, normed=1, histtype='stepfilled',color ='g')
```

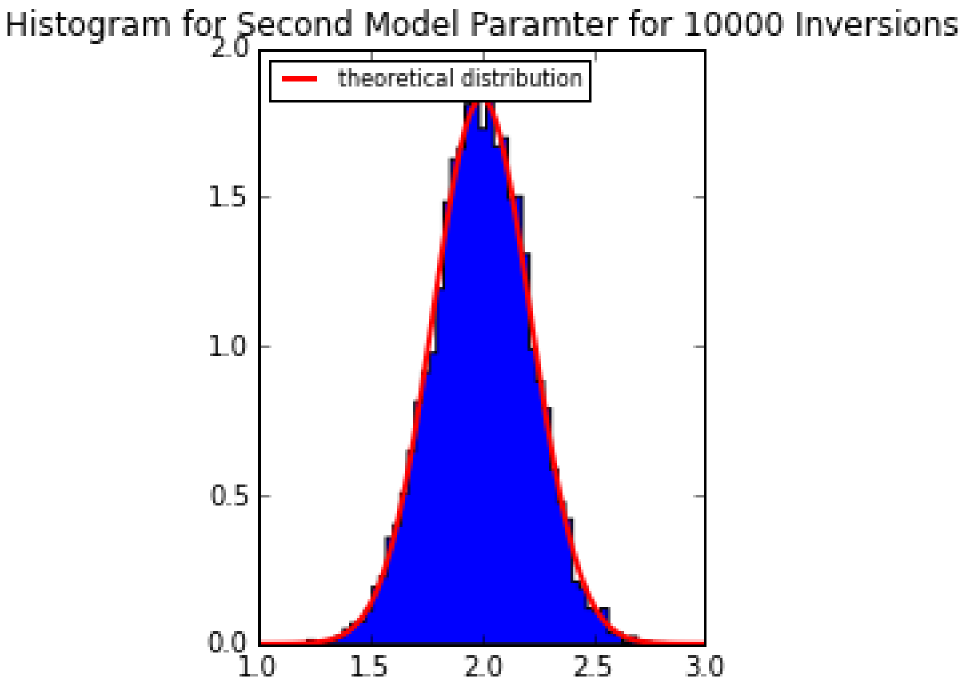
```
        #theoretical curve using Cm Variance
        x = np.linspace(0.5,1.5,100)
        sigma = np.sqrt(0.00971429)
        ax.plot(x,mlab.normpdf(x,1,sigma), color = 'r', linewidth = 2, label = 'theoretical distributio
        ax.legend(loc = 'upper left',prop={'size':8})
        ax.set_title('Histogram for First Model Paramter for 10000 Inversions (unweighted)')

        fig = plt.figure(figsize=(3,4))
        ax=fig.add_subplot(1,1,1)
        #plotting only second column of model matrix (2nd value of each of 10000 predicted models)
        ax.hist(m[:,1], 50, normed=1, histtype='stepfilled',color ='g')
        #theoretical curve using Cm variance
        x = np.linspace(1,3,100)
        sigma = np.sqrt(0.04742857)
        ax.plot(x,mlab.normpdf(x,2,sigma), color = 'r', linewidth = 2, label = 'theoretical distributio
        ax.set_title('Histogram for Second Model Paramter for 10000 Inversions (unweighted)')
        ax.legend(loc = 'upper left',prop={'size':8})


        fig = plt.figure(figsize=(3,4))
        ax=fig.add_subplot(1,1,1)
        #plotting only third column of model matrix (3rd value of each of 10000 predicted models)
        ax.hist(m[:,2], 50, normed=1, histtype='stepfilled',color ='g')
        #theoretical curve using Cm variance
        x = np.linspace(2,4.5,100)
        sigma = np.sqrt(0.10171429)
        ax.plot(x,mlab.normpdf(x,3,sigma), color = 'r', linewidth = 2, label = 'theoretical distributio
        ax.set_title('Histogram for Third Model Paramter for 10000 Inversions (unweighted)')
        ax.legend(loc = 'upper right',prop={'size':8})

        fig = plt.figure(figsize=(3,4))
        ax=fig.add_subplot(1,1,1)
        #plotting only fourth column of model matrix (4th value of each of 10000 predicted models)
        ax.hist(m[:,3], 50, normed=1, histtype='stepfilled',color ='g')
        #theoretical curve using Cm variance
        x = np.linspace(2.5,6.5,100)
        sigma = np.sqrt(0.20685714)
        ax.plot(x,mlab.normpdf(x,4,sigma), color = 'r', linewidth = 2, label = 'theoretical distributio
        ax.legend(loc = 'upper left',prop={'size':8})
        ax.set_title('Histogram for Fourth Model Paramter for 10000 Inversions (unweighted)')

        fig = plt.figure(figsize=(3,4))
        ax=fig.add_subplot(1,1,1)
        #plotting only last column of model matrix (5th value of each of 10000 predicted models)
        ax.hist(m[:,4], 50, normed=1, histtype='stepfilled',color ='g')
        #theoretical curve using Cm variance
        x = np.linspace(3,7,100)
        sigma = np.sqrt(0.18971429)
        ax.plot(x,mlab.normpdf(x,5,sigma), color = 'r', linewidth = 2, label = 'theoretical distributio
        ax.legend(loc = 'upper left',prop={'size':8})
        ax.set_title('Histogram for Fifth Model Paramter for 10000 Inversions (unweighted)')
```

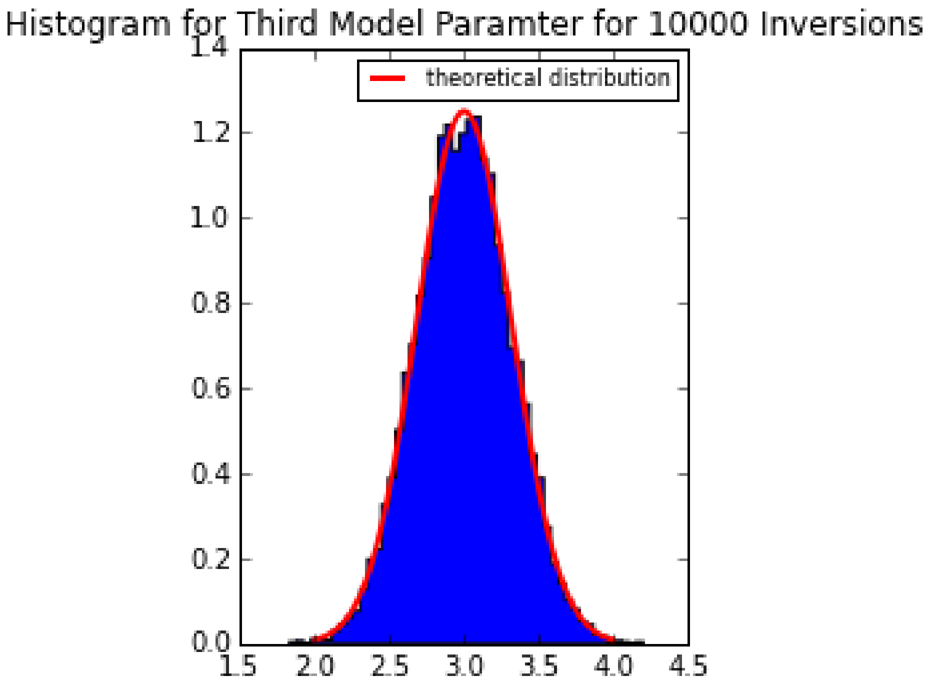
```
Cd
[[ 0.09    0.      0.      0.      0.      0.    ]
```

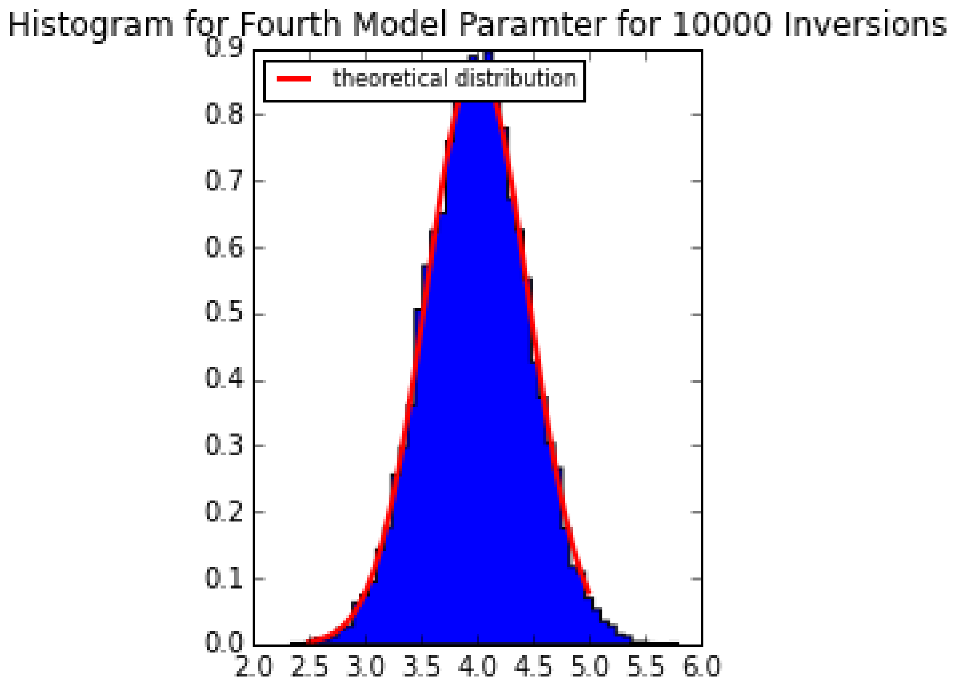
```
 [ 0.       0.04    0.      0.      0.      0.    ]
 [ 0.       0.      0.01    0.      0.      0.    ]
 [ 0.       0.      0.      0.01    0.      0.    ]
 [ 0.       0.      0.      0.      0.0025  0.    ]
 [ 0.       0.      0.      0.      0.      0.0025]]
part a: blue plots
Cm variances about true parameters
[[ 0.00971429 -0.01914286  0.01771429 -0.01542857  0.00971429]
 [-0.01914286  0.04742857 -0.05314286  0.04628571 -0.02914286]
 [ 0.01771429 -0.05314286  0.10171429 -0.12342857  0.07771429]
 [-0.01542857  0.04628571 -0.12342857  0.20685714 -0.15542857]
 [ 0.00971429 -0.02914286  0.07771429 -0.15542857  0.18971429]]
part b: green plots
```

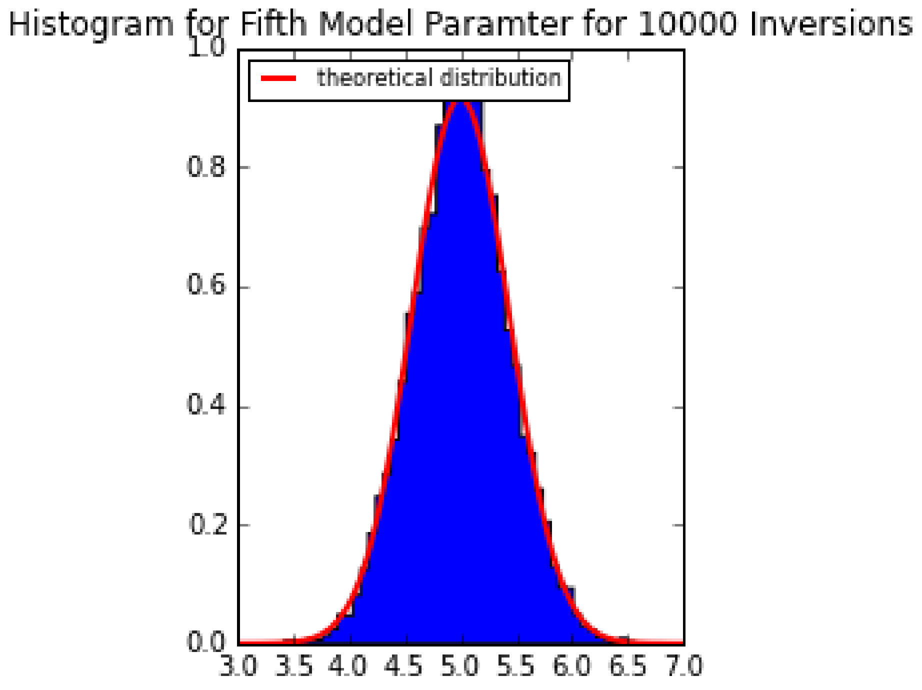
```
Out[1]: <matplotlib.text.Text at 0xbbef780>
```



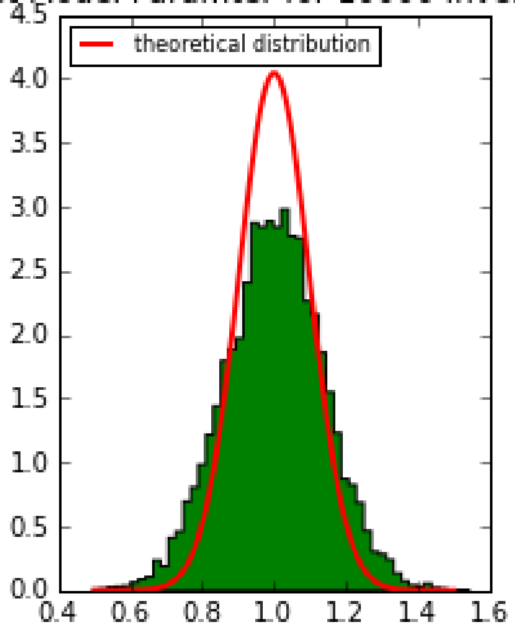

Histogram for Second Model Paramter for 10000 Inversions
theoretical distribution
2.0
1.5
1.0
0.5
0.0
1.0
1.5
2.0
2.5
3.0


Histogram for Third Model Paramter for 10000 Inversions
theoretical distribution
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5

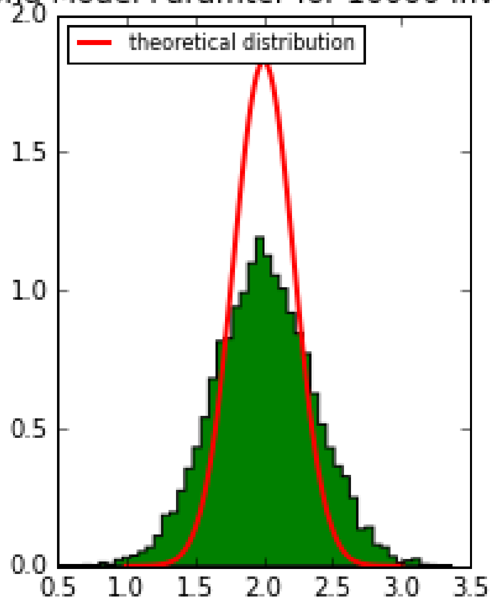

Histogram for Fourth Model Paramter for 10000 Inversions
theoretical distribution
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0


Histogram for Fifth Model Paramter for 10000 Inversions
theoretical distribution
0.0
0.2
0.4
0.6
0.8
1.0
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0

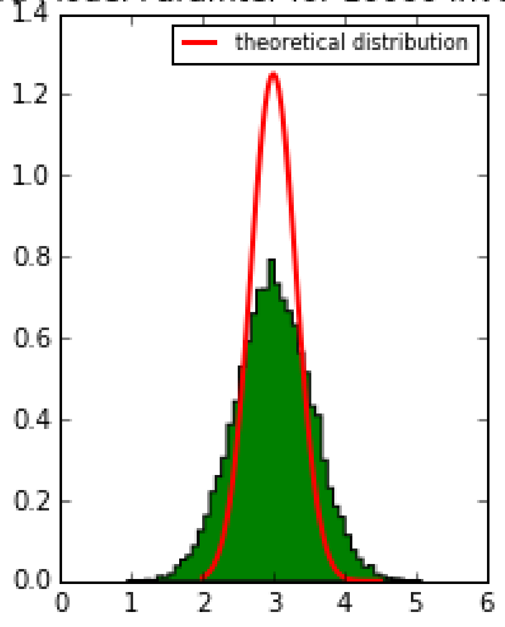
Histogram for First Model Paramter for 10000 Inversions (unweighted)



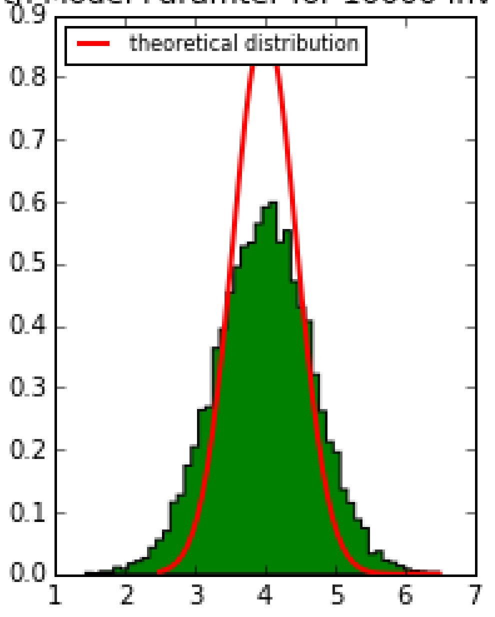
Histogram for Second Model Paramter for 10000 Inversions (unweighted)

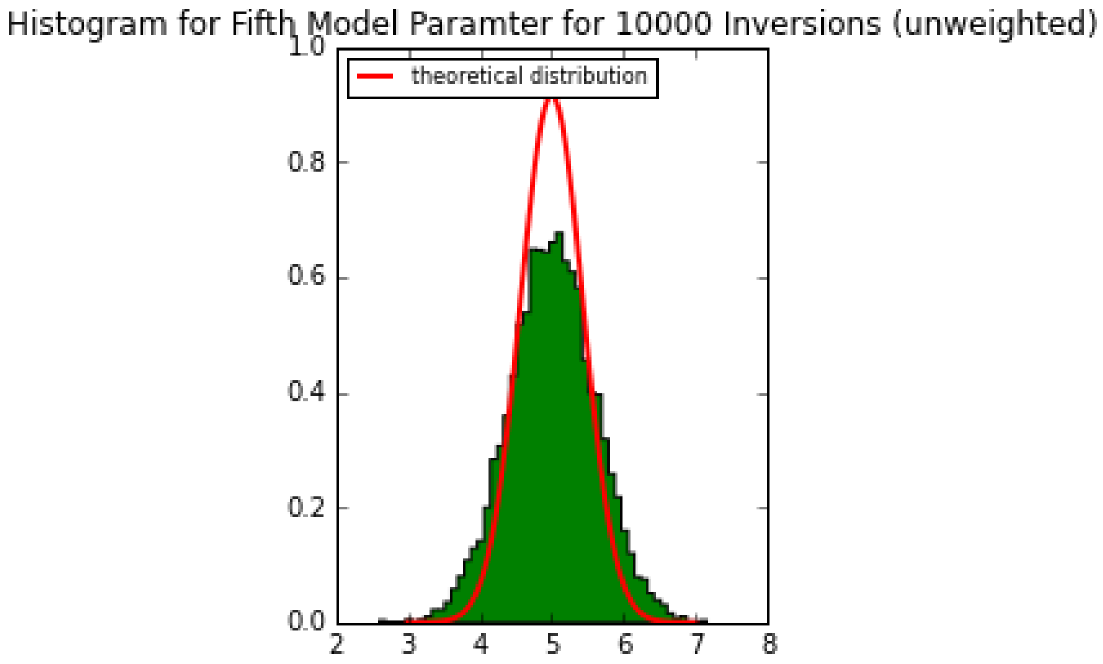


Histogram for Third Model Paramter for 10000 Inversions (unweighted)



Histogram for Fourth Model Paramter for 10000 Inversions (unweighted)





For part a, the histograms fit the theoretical gaussian distributions very well. The histograms in part b were wider distributions than the theoretical distributions. This is because the inversion used the same standard deviation (0.1 in my claculation) which was in the middle of the true spectrum of given standard deviations. Can say the method for part a provides a much better estimation of the model parameters for an overdetermined problem.

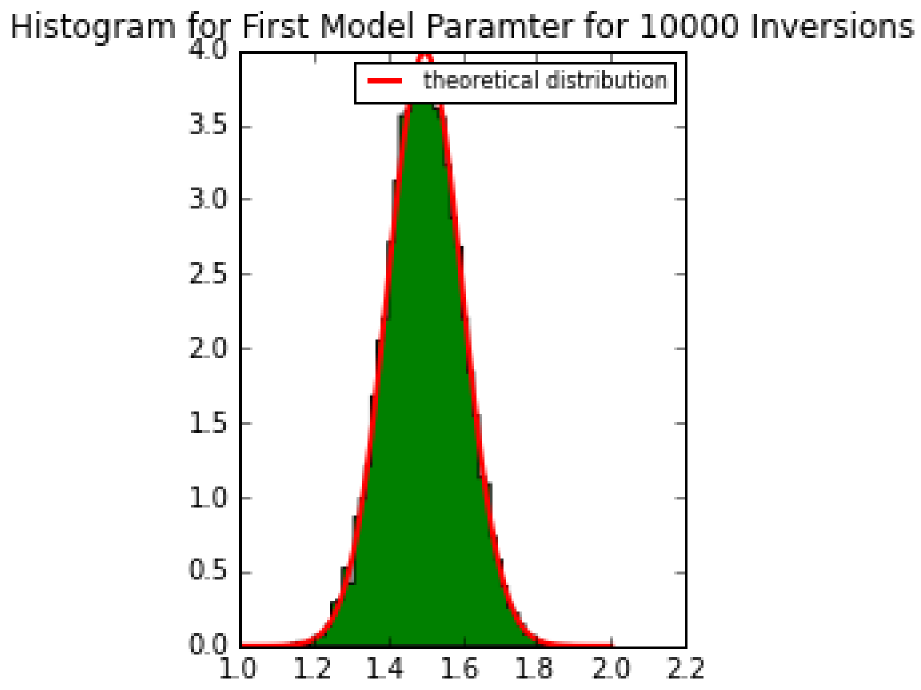
Question 2

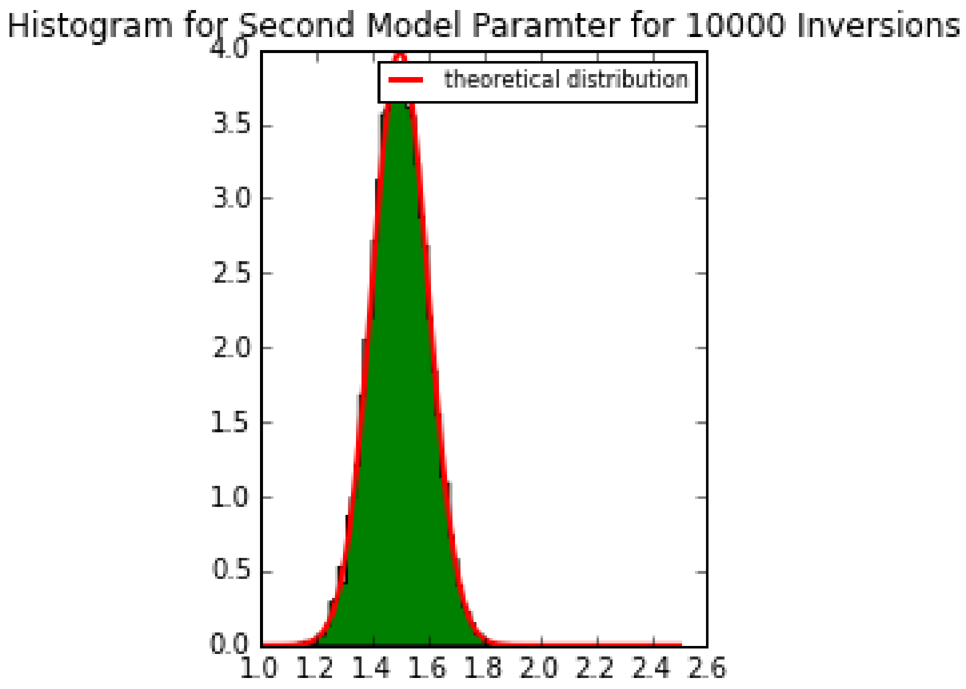
```
In [4]: print('Question 2')
        #sensitivity matrix
        A = np.matrix([[1.,1.,1.],[1.,1.,0]])
        A_T = A.transpose()

        #true model
        mt = np.matrix([1.,2.,3.])
        m_t = mt.transpose()

        #true data
        d_t = np.matrix([6.,3.])
        d_t = d_t.transpose()

        #a) d_t = d
        A_AT = np.dot(A,A_T)
        inv = ag.inv(A_AT)
        dot = np.dot(A_T,inv)
        m = np.dot(dot,d_t)
        mtr = m.transpose()
        print('smallest model solution:')
        print m

        #b) calculation method hand written on attached page
        m1t = np.matrix([2.,1.,3.])
```

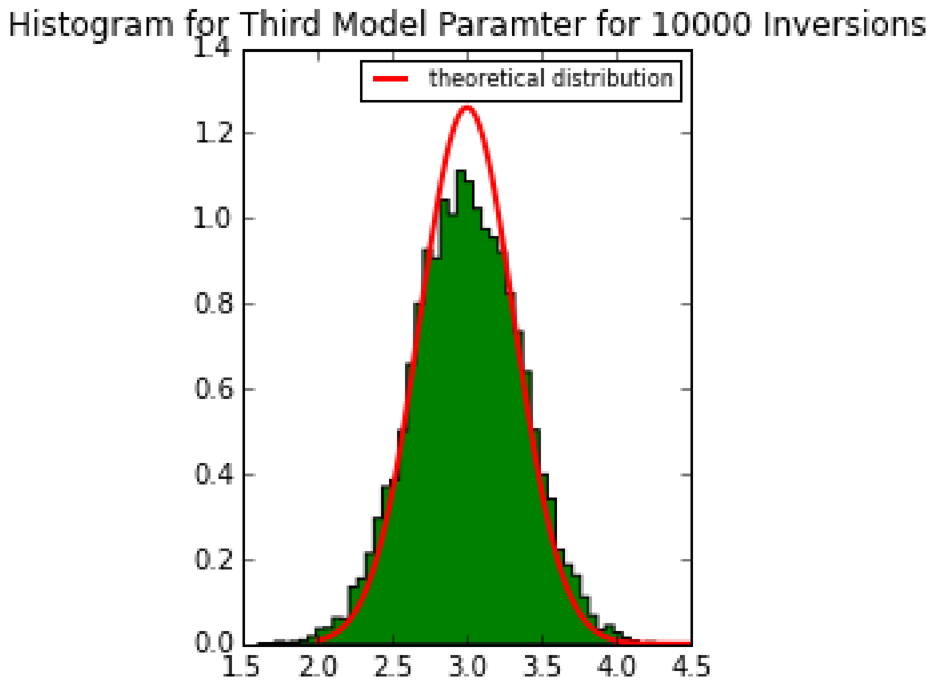
```
m1 = m1t.transpose()


m2t = np.matrix([5,-2.,3.])
m2 = m2t.transpose()

m3t = np.matrix([4.,-1.,3.])
m3 = m3t.transpose()

print('3 other exact models')
print m1
print m2
print m3

print('comparing mTm for exact models, true model, and smallest model:')
m1Tm1 = np.dot(m1t,m1)
print('exact 1')
print m1Tm1
m2Tm2 = np.dot(m2t,m2)
print('exact 2')
print m2Tm2
m3Tm3 = np.dot(m3t,m3)
print('exact 3')
print m3Tm3
mTm = np.dot(mt,m_t) #
print('true model ')
print mTm
msTms = np.dot(mtr,m)
print('smallest model ')
print msTms
print('Can see that the solution length is highly variable, as two of the model parmters can be
print('The smallest solution has the smallest model length of all possible solutions.')










#c)

a=0
m = np.zeros((10000,3))
for a in range(10000):
    #noise (using given standard deviations: stddev)
    noise = np.zeros(2)
    stddev = np.array([0.3,0.2])
    for i in range(0, len(stddev)):
        sigma = np.random.normal(0,stddev[i])
        noise[i] = sigma
    noise = np.matrix(noise)
    noise = noise.transpose()
    d= d_t + noise
    A_AT = np.dot(A,A_T)
    inv = ag.inv(A_AT)
    dot1 = np.dot(A_T,inv)
    M = np.dot(dot1,d) #predicted model
    M = np.matrix(M)
    m[a,:] = M.transpose()
    #put models into rows for easy plotting (matrix of 10000 predicted models)

#Cm calculation
Cd = np.matrix([[0.3**2,0,0],[0,0.2**2,0]])
one = np.dot(dot1,Cd)
two = np.dot(one,dot1)
Cm = np.dot(two,A)
print('Cm matrix:')
print Cm

#Histograms(3 in total for each parameter of predicted model)

fig = plt.figure(figsize=(3,4))
ax=fig.add_subplot(1,1,1)
#plotting only first column of model matrix (1st value of each of 10000 predicted models)
ax.hist(m[:,0], 50, normed= True, histtype='stepfilled',color ='g')
#theoretical curve using Cm Variance
x = np.linspace(1,2,100)
sigma = np.sqrt(0.01)
ax.plot(x,mlab.normpdf(x,1.5,sigma), color = 'r', linewidth = 2, label = 'theoretical distribut
ax.legend(loc = 'upper right',prop={'size':8})
ax.set_title('Histogram for First Model Paramter for 10000 Inversions')

fig = plt.figure(figsize=(3,4))
ax=fig.add_subplot(1,1,1)
#plotting only second column of model matrix (2nd value of each of 10000 predicted models)
ax.hist(m[:,1], 50, normed=True, histtype='stepfilled',color ='g')
#theoretical curve using Cm variance
x = np.linspace(1,2.5,100)
sigma = np.sqrt(0.01)
ax.plot(x,mlab.normpdf(x,1.5,sigma), color = 'r', linewidth = 2, label = 'theoretical distribut
ax.set_title('Histogram for Second Model Paramter for 10000 Inversions')
ax.legend(loc = 'upper right',prop={'size':8})

fig = plt.figure(figsize=(3,4))
ax=fig.add_subplot(1,1,1)
#plotting only third column of model matrix (3rd value of each of 10000 predicted models)
ax.hist(m[:,2], 50, normed=True, histtype='stepfilled', color = 'g')
#theoretical curve using Cm variance
x = np.linspace(2,4.5,100)
sigma = np.sqrt(0.1)
ax.plot(x,mlab.normpdf(x,3,sigma), color = 'r', linewidth = 2, label = 'theoretical distributio
ax.legend(loc = 'upper right',prop={'size':8})
ax.set_title('Histogram for Third Model Paramter for 10000 Inversions')
```

```
Question 2
smallest model solution:
[[ 1.5]
 [ 1.5]
 [ 3. ]]
3 other exact models
[[ 2.]
 [ 1.]
 [ 3.]]
[[ 5.]
 [-2.]
 [ 3.]]
[[ 4.]
 [-1.]
 [ 3.]]
comparing mTm for exact models, true model, and smallest model:
exact 1
[[ 14.]]
exact 2
[[ 38.]]
exact 3
[[ 26.]]
true model
[[ 14.]]
smallest model
[[ 13.5]]
Can see that the solution length is highly variable, as two of the model parmters can be manipulated.
The smallest solution has the smallest model length of all possible solutions.
Cm matrix:
[[ 0.01   0.01   0.    ]
 [ 0.01   0.01   0.    ]
 [ 0.025  0.025  0.    ]]
```

```
Out[4]: <matplotlib.text.Text at 0xb560cc0>
```


Histogram for First Model Paramter for 10000 Inversions
theoretical distribution
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1.0
1.2
1.4
1.6
1.8
2.0
2.2


Histogram for Second Model Paramter for 10000 Inversions
theoretical distribution
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
1.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
2.6



Question 3

```
In [ ]:
```