Error Analysis EAX

Kevin Chen

January 25, 2015

1 Problem 1

We want to measure the specific activity (number of decays per second) of a radioactive source so that we can use it to calibrate the equipment of the gamma-ray experiment. We use an electronic counter and a timer to measure the number of decays in a given time interval. In round numbers we measure 1000 decays in 5 minutes of observation. For how long should we measure in order to obtain a specific activity with a precision of 1%? Explain.

Assuming that there are no instrumental time errors, the counts obey Poisson Statistics, and that we can expect another 1000 decays in the following 5 minutes, then to obtain a precision of 1% we must have 10000 counts after 50 minutes. We obtain that result by the following Poisson characteristics:

$$\sigma = \sqrt{\mu}$$

$$Precision = \frac{\sigma}{\mu} = \frac{\sqrt{\mu}}{\mu} = \frac{1}{\sqrt{\mu}}$$

μ then equals

$$\frac{1}{Precision^2} = 100^2 = 10000$$

To measure 10000 events, we need 50 minutes of time since we gather data at 200 events per minute.

2 Problem 2

You are given two measurements of distance and their associated uncertainties: $A \pm \sigma_A$

and $B \pm \sigma_B$. Calculate the propagated uncertainty in the (a) total distance $A + B$, (b) difference $A - B$, (c) the perimeter $2A + 2B$ and (d) the area $A \times B$.

1. $\sigma_{x,A+B} = \sqrt{\sigma_A^2 + \sigma_B^2}$
2. $\sigma_{x,A-B} = \sqrt{\sigma_A^2 + \sigma_B^2}$
3. $\sigma_{x,2A+2B} = 2\sqrt{\sigma_A^2 + \sigma_B^2}$
4. $\sigma_{x,A\times B} = \sqrt{B\sigma_A^2 + A\sigma_B^2}$

3 Problem 3

In this problem we will be generating and analyzing lists of normally distributed random numbers. The distribution we are sampling has true mean 0 and standard deviation 1.

1. *If we sample this distribution N=5 times, what do we expect the mean to be? How about the standard deviation? Whats the error on the mean?*
2. *Using Matlab, generate a list of N=5 normally distributed random numbers (the command randn(N,M) will generate M lists of length N). Calculate the mean, standard deviation and the error on the mean. Is this what you expected?*
3. *Now find the mean, standard deviation and error on the mean for M=1000 experiments of N=5 measurements each. Plot a histogram of the distribution. About how many experiments are within 1 sigma of the true mean of 0. About how many are within 2 sigma? Is this what you expected?*
4. *Now repeat questions 1-3 for N=10,50,100,1000.*

For part 1 of this problem, I calculated the following means, standard deviations, and errors:

Part	**N = 5**	**N = 10**	**N = 50**	**N = 100**	**N = 1000**
1	Mean=0	0	0	0	0
	SD=1	1	1	1	1
	Err=$\frac{\sigma}{\sqrt{N}}=\frac{\sigma}{\sqrt{5}}$ =0.447	0.316	0.1414	0.1	0.0316

For parts 2-4, I did not use MATLAB. Instead, I used Python and its numerical analysis package, NumPy. The following is the code I used to generate random numbers and the analysis for the mean, standard deviation, and error from the mean.

```
""" Python Code for Error Analysis Homework """

import numpy as np
import math
import matplotlib.pyplot as plt

def one_normal(n): # n = 5,10,50,100,1000
    measurements = np.random.standard_normal(n)
    mean = measurements.mean(axis = 0)
    std = measurements.std(axis = 0)
    mean_error = std / math.sqrt(n)
    print("Mean: " + str(mean))
    print("Standard Deviation: " + str(std))
    print("Error in the Mean: " + str(mean_error))

def normal(m,n): # m = 1000, n = 5,10,50,100,1000

    " Calculate the stats for the measurements per experiment. "
    experiments = np.random.randn(m,n)
    means = experiments.mean(axis = 1)
    standard_deviations = experiments.std(axis = 1)
    variances = [x*x for x in standard_deviations]
    errors = [x / math.sqrt(n) for x in standard_deviations]

    " Now calculate the stats of all the experiments. "
    total_mean = sum(means)/m
    total_standard_deviation = math.sqrt(sum(variances))/m
    total_errors = total_standard_deviation/math.sqrt(m)
    print("Mean :" + str(total_mean))
    print("Standard Deviation: " + str(total_standard_deviation))
    print("Error :" + str(total_errors))
    plt.hist(means, 30)
    plt.show()
```

The follow figures display the analysis for Problem 3.2 and 3.3, along with the histograms for 3.3.

```
>>> one_normal(5)
Mean: -0.108059242868
Standard Deviation: 0.85550615585
Error in the Mean: 0.38259398393
>>> one_normal(10)
Mean: -0.0455920646707
Standard Deviation: 0.904765414754
Error in the Mean: 0.286111945877
>>> one_normal(50)
Mean: -0.151943082194
Standard Deviation: 0.995160938107
Error in the Mean: 0.140737009541
>>> one_normal(100)
Mean: 0.0012958142959
Standard Deviation: 1.02083949666
Error in the Mean: 0.102083949666
>>> one_normal(1000)
Mean: 0.0447831373766
Standard Deviation: 1.01409636429
Error in the Mean: 0.0320685427804
```

Figure 1: Analysis for Problem 3.2

```
>>> normal(1000,5)
Mean :0.0472362512975
Standard Deviation: 0.0286429479408
Error :0.000905769543945
>>> normal(1000,10)
Mean :-0.0101825510574
Standard Deviation: 0.0301593455345
Error :0.000953722246291
>>> normal(1000,50)
Mean :0.00306431529641
Standard Deviation: 0.0313898043293
Error :0.000992632769875
>>> normal(1000,100)
Mean :0.00579029975406
Standard Deviation: 0.0314411997774
Error :0.00099425803665
>>> normal(1000,1000)
Mean :0.000433769450481
Standard Deviation: 0.0316311564698
Error :0.0010002649947
```

Figure 2: Analysis for Problem 3.3

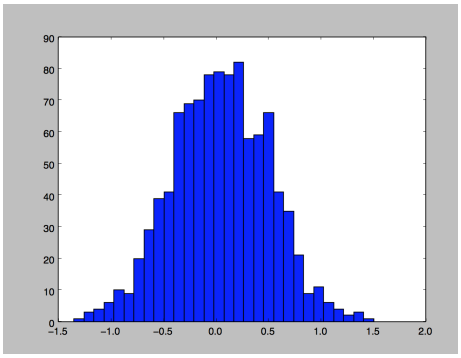


Figure 3: Distribution of the means of 1000 experiments, each with 5 measurements

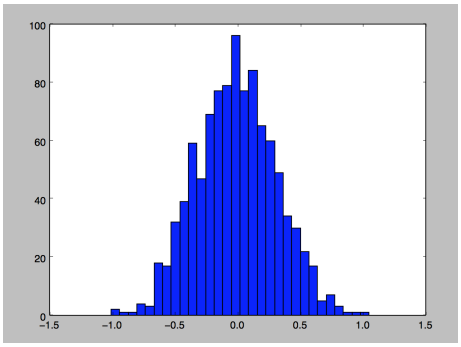


Figure 4: Distribution of the means of 1000 experiments, each with 10 measurements

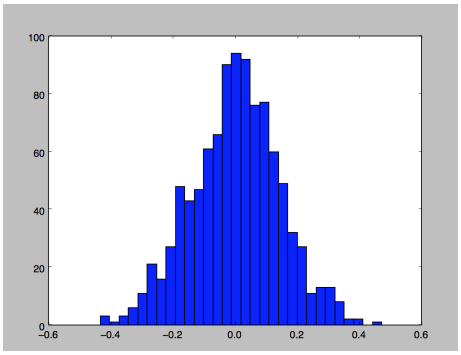


Figure 5: Distribution of the means of 1000 experiments, each with 50 measurements

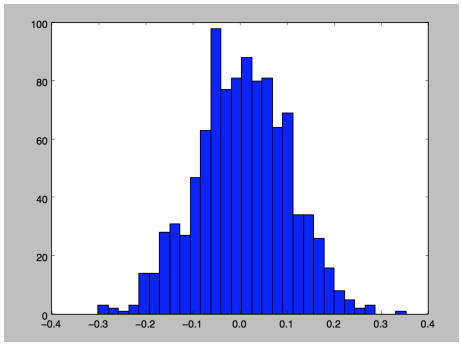


Figure 6: Distribution of the means of 1000 experiments, each with 100 measurements

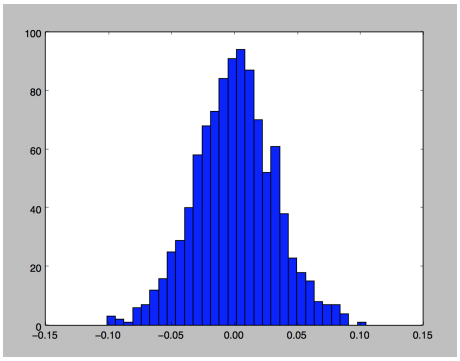


Figure 7: Distribution of the means of 1000 experiments, each with 1000 measurements

4 Problem 4

In this problem we will repeat the above process, but now using lists of exponentially distributed random numbers. The probability of selecting a random number between $xandx + dxis \propto e^{-x}dx$.

1. *What do you expect to be the mean of the distribution? What do you expect to be the standard deviation? (Note: The standard deviation is defined exactly as it is for a normal distribution, but the 1 sigma = 68% rule no longer applies to a exponential distribution.) What do you expect to be the error on the mean for N=100? Given M=1000 lists of N=100 random numbers, what do you expect the distribution of means to look like? What is the error on the mean?*
2. *Make a list of N=100 exponentially distributed random numbers, (Hint: this can be easily done starting with a uniform distribution of random numbers.) Calculate the mean and standard deviation.*
3. *Make M=1000 lists of N=100 exponentially distributed random numbers. Make a histogram of the means. Does the distribution of means look as you thought? What is the standard deviation of the means. Does this agree with what you thought?*
4. *Repeat the previous steps for N=1000 and 10000. Does the error on the mean scale as you thought?*

This is a demonstration of the Central Limit Theorem.

For part 1 of this problem, I calculated the following means, standard deviations, and errors:

Part	**N = 100**	**N = 1000**	**N = 10000**
1	Mean=1	1	1
	SD=1	1	1
	Err=$\frac{\sigma}{\sqrt{N}}=\frac{\sigma}{\sqrt{100}}$ =0.1	0.0316	0.01

I expect the distribution for M=1000 and N=100,1000,10000 to approach a normal distribution. The error on that mean would be $\frac{standarddeviationofexponentiallydistributed}{\sqrt{(M)}}$.

For parts 2-4 of this problem, I used the following NumPy code to generate my random numbers.

```
""" Python Code for Error Analysis Homework """

import numpy as np
import math
import matplotlib.pyplot as plt
```

```
def one_exponential(n): # n = 100, 1000, 10000
    measurements = np.random.exponential(1,n)
    mean = measurements.mean(axis = 0)
    std = measurements.std(axis = 0)
    mean_error = std / math.sqrt(n)
    print("Mean: " + str(mean))
    print("Standard Deviation: " + str(std))
    print("Error in the Mean: " + str(mean_error))

def exponential(m,n): # m = 1000, n = 100, 1000, 10000

    " Calculate the stats for the measurements per experiment. "
    experiments = np.random.exponential(1,(m,n))
    means = experiments.mean(axis = 1)
    standard_deviations = experiments.std(axis = 1)
    variances = [x*x for x in standard_deviations]
    errors = [x / math.sqrt(n) for x in standard_deviations]

    " Now calculate the stats of all the experiments. "
    total_mean = sum(means)/m
    total_standard_deviation = math.sqrt(sum(variances))/m
    total_errors = total_standard_deviation/math.sqrt(m)
    print("Mean: " + str(total_mean))
    print("Standard Deviation: " + str(total_standard_deviation))
    print("Error: " + str(total_errors))
    plt.hist(means, 30)
    plt.show()
```

For part 2, the calculated means and standard deviations are found in the terminal screenshots below, using the code from above.

For part 3, the histograms, means, standard deviations, and errors on the mean are found in the terminal screenshots below. The distribution of means looks as I thought, just like a normal distribution. The standard deviation of the means does not follow the standard deviation of exponentially distributed numbers (which is just 1); instead, it follows that of a normal distribution. The standard deviation on the means is 0.0316, and seems to be approaching $1/\sqrt{M} = 1/\sqrt{1000}$. The follow figures display the analysis for Problem 4.2 and 4.3, along with the histograms for 4.3. As you can see, the Standard

Deviations in 4.3 agrees with 4.1.

For part 4, the N=1000 and N=10000 are accounted for in the pictures below. The error in the mean for 4.2 scales as I thought, which is by $1/\sqrt{N}$.

```
>>> one_exponential(100)
Mean: 0.988260825611
Standard Deviation: 1.01919948977
Error in the Mean: 0.101919948977
>>> one_exponential(1000)
Mean: 1.04970564749
Standard Deviation: 1.0326110982
Error in the Mean: 0.0326540300749
>>> one_exponential(10000)
Mean: 0.976405557304
Standard Deviation: 0.959371164763
Error in the Mean: 0.00959371164763
```

Figure 8: Analysis for Problem 4.2

```
>>> exponential(1000,100)
Mean: 0.997500034148
Standard Deviation: 0.0312704585116
Error: 0.000988858723743
>>> exponential(1000,1000)
Mean: 0.999823895814
Standard Deviation: 0.0316149870841
Error: 0.000999753673824
>>> exponential(1000,10000)
Mean: 1.00018041901
Standard Deviation: 0.0316156204651
Error: 0.000999773703091
```

Figure 9: Analysis for Problem 4.3. Histograms are below.

N=100.png

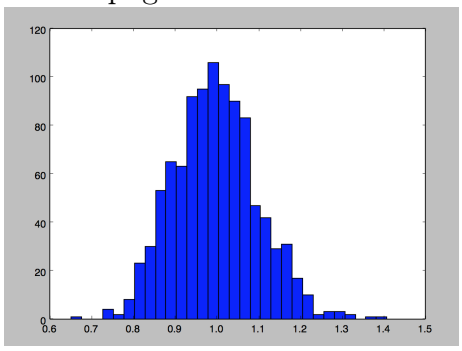


Figure 10: Distribution of the means of 1000 experiments, each with 100 measurements

N=1000.png

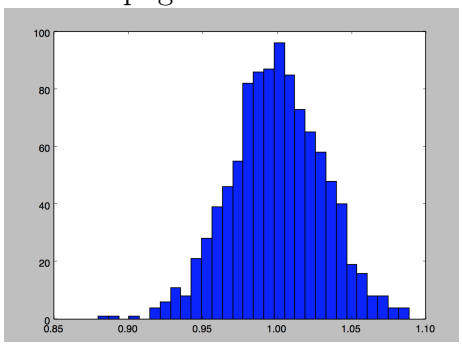


Figure 11: Distribution of the means of 1000 experiments, each with 1000 measurements

N=10000.png

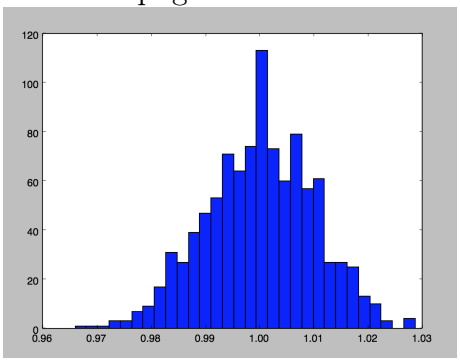
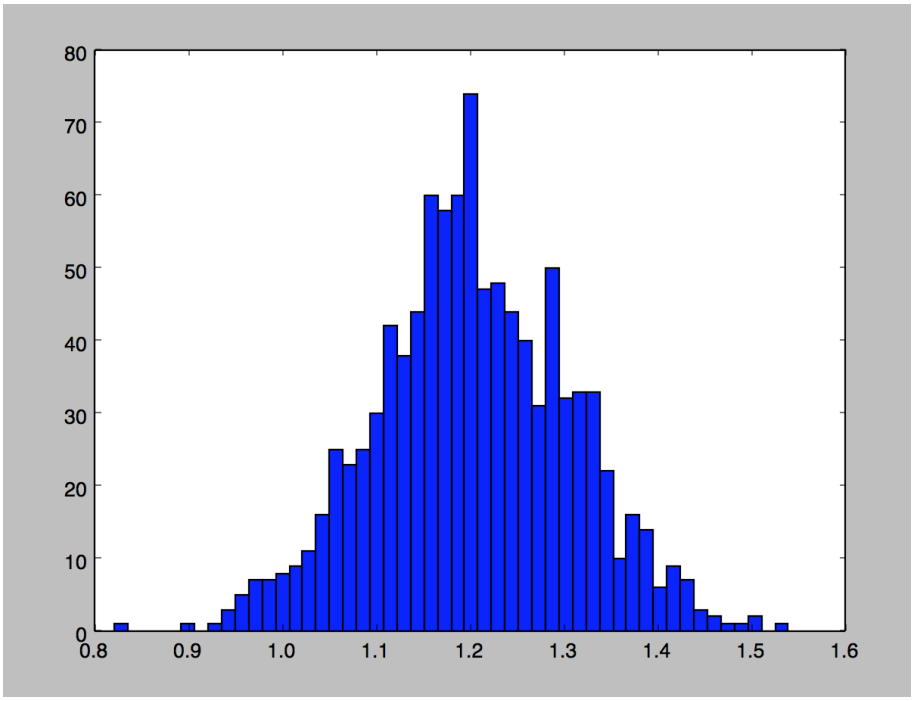


Figure 12: Distribution of the means of 1000 experiments, each with 1000 measurements

5 Problem 5

Figure 13: Peak.dat data.

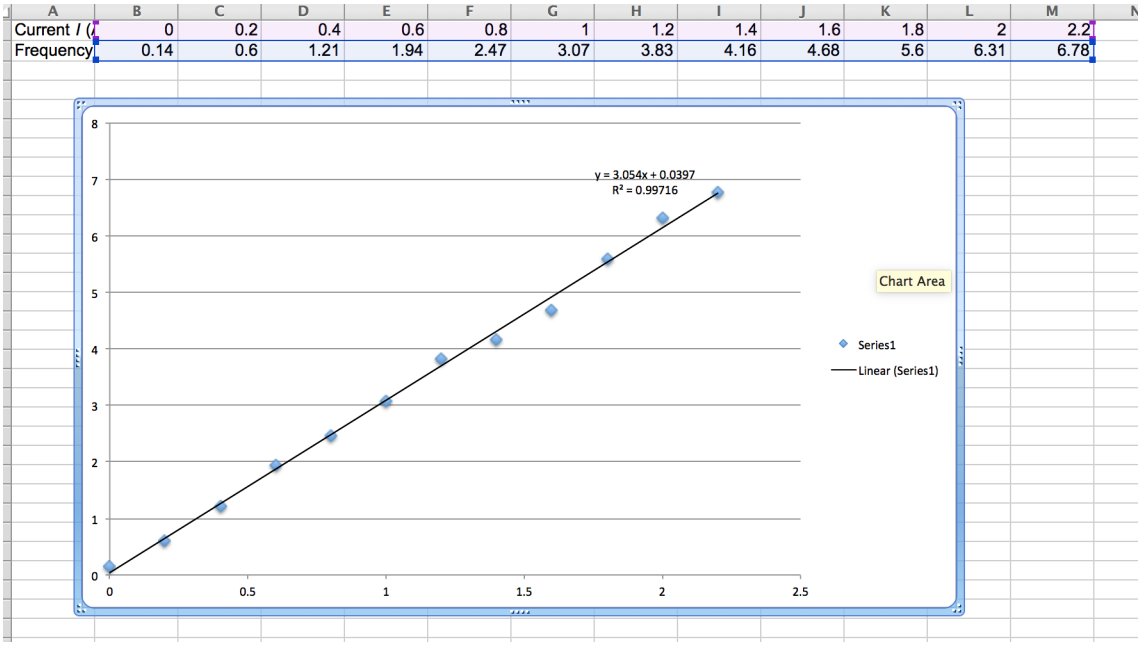


- Above is the distribution of energies. I picked the 50 bins to get a nice compromise between granularity and high population in the most populated bin is relatively large.
- The mean = 1.20, SD = 0.104, the error on the mean = 0.00328.
- I was not able to fit a Gaussian using NumPy. I will switch to MATLAB in the future since fitting curves in NumPy is excruciatingly difficult.
- N/A

6 Problem 6

- The slope is 3.05 and the intercept is 0.0397. See figure below.
- By using the formula for Chi-squared, we find the with an uncertainty of 0.01 MHz, we get a Chi-squared of 1520. The probability that this is an adequate fit is 0. With an uncertainty of 1 MHz, we get a Chi-squared of 0.152. In the latter case the probably that the straight line is an adequate fit is 0.999. Therefore, we conclude that a good uncertainty is near 1MHz.
- The uncertainty in the frequency is 0.12 MHz. The uncertainty in the intercept is about 0.067. The uncertainty in the slope is about 0.052.
- With a weighted least-squares fit, the y-intercept is 0.0015 and the slope is 3.1.

Figure 14: Plot of the data with linear fit.



7 Problem 7

- The lower errors get larger and larger at the tail end of the exponential function due to the scaling of the y-axis. Quantitatively, the statistical error does not change. It is just the displaying of the error that changes.
- $E_0 = \exp 2.1 = 8.17$ and $\sigma_x = 0.5(E_0) = 4.08$ Note that these are extremely large experimental bounds.