Evaluation of Analytical Data

1. Appendix 1; *Principles of Instrumental Analysis, 6th ed., 7th ed.*, Skoog, Douglas A., Holler, F. James, Crouch, Stanley R., (2007, 2018), Thomson Brooks/Cole Cengage Learning, California.
2. Chapters 3 and 4; *Quantitative Chemical Analysis, 8th ed., 9th ed.*, Harris, Daniel C., (2010, 2018), W.H. Freeman and Company, New York, New York.
3. Chapter 1; *Quantitative Analysis, 6th ed.*, Day, R.A., Underwood, A.L., (1991), Prentice Hall, New Jersey.
4. Chapters 5, 6, and 7; *Fundamentals of Analytical Chemistry, 9th ed.*, Skoog, Douglas A., West, Donald M., Holler, F. James, Crouch, Stanley R., (2014), Thomson Brooks/Cole Cengage Learning, California.
5. Chapter 1; *Undergraduate Instrumental Analysis, 6th ed.*, Robinson, James W., Skelly Frame, Eileen M., Frame II, George M., (2005), Marcel Dekker, New York, New York.
6. **Please refer to the "fitting__various_techniques3.xls" file for updates and the actual source code of the spreadsheets.**

lecture by Stephen Lukacs, Ph.D., ©2011 - 2023; updated: March 7, 2023

Introduction

Measurements are gathered in the laboratory or field. The collection of measurements yield data. The data should be quantitatively processed to yield reasonably accurate results and provide conclusions reflective of the true nature of the system under study. This is a lecture to help you understand the nature of data, how to process it, and ensure that conclusions made on the data are based on reality. These techniques should be used in all laboratories.

Properly processing, treating, and evaluating data is to turn raw measurements, raw data, into meaningful information and proper conclusions. Evaluating data takes the values of the experiment and brings the researcher to natural conclusions and hence meaningful information. As with anything, there is a right way and a wrong way to process data.

This lecture is meant to compliment your textbook, which is the *Principles of Instrumental Analysis* by Skoog *et.al.*, here on out as simply "the text", the other supplemental reading materials, and any podcast or media based content. You are responsible for being competent and understanding all of the material presented, whether part of this lecture, the textbook, or any supplemental material.

To that end, I emphatically suggest that you reproduce the following examples in Excel with verification with your scientific calculator. This lecture is not meant to give you recipes, per se, but to provide a guide of how to process data into information. Reproducing and verifying the below examples and methods will ensure that you can actively learn and adapt these analytical processing techniques for yourselves by your own methods. You will be tested on your ability to process all of these forms of data.

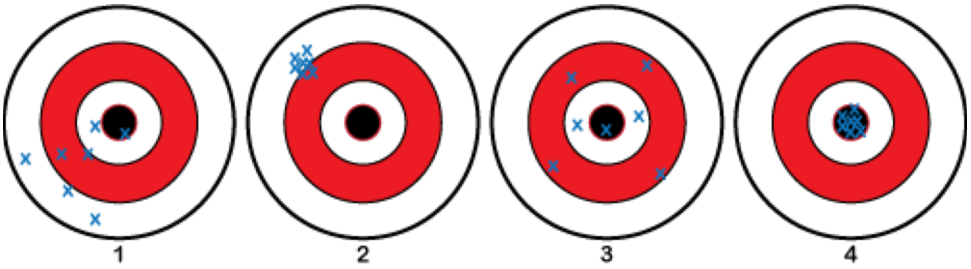
Accuracy and Precision

Accuracy is simply: how close your measurement is, or set of measurements are, i.e., data, to the true or "accepted" value. The problem with that statement is, what is the true value? I mean, if you make a solution of something and you know for sure that all of it dissolved in the solution, then you know the concentration of that something in the solution. You definitely know the true value because you created it. However, if it is a dataset of measurements of something that was not made by you, then how can you know the true value? Like the temperature or brightness of the sun?

This is where the "accepted" value comes in. The accepted value usually occurs when the measurements are done not only through repeated measurements, but also by other research groups reproducing the same experiment, and also if can be done, similar results achieved by a different experimental method or technique. This way the same result is achieve via different perspectives and methods. Once such measurements are made through the course of time, then the accepted value becomes the true value through persistent acceptance throughout the scientific community. Sometimes drama and emotions get involved when the scientists don't agree. Later in this lecture, your accuracy goal is to approach the accepted or true value.

Precision is very different from accuracy. Precision is simply how close your measurements are to each other. Those measurements may collectively be far from the mark of the accepted or true value. Therefore, your measurements could agree with each other, being precise, but they can be wrong because they are inaccurate. Strictly speaking, however, if considering only precision and not accuracy, the goal is to have your measurements as close to each other as possible. Later in this lecture, your precision goal is to minimize the standard deviation with a limit approaching zero.

Here is a nice way to summarize the meanings and differences:



Bullseye 1 has poor precision, the shots are very spread out, and poor accuracy, the average of the group is far from the true value.

Bullseye 2 has great precision, the shots are very close to each other, but poor accuracy, the average of the group is far from the true value.

Bullseye 3 has poor precision, the shots are very spread out, but it happens to have good accuracy, the average of the group is close to the true value.

And, Bullseye 4 has great precision, the shots are very close to each other, and great accuracy, the average of the group is close to the true value. This is obviously the optimum and desired effect where

determinate/systematic errors are reduced or completely eradicated and indeterminate/random errors are reduced. You can never completely eradicate all random error, as discussed below.

Significant Figures

Significant figures are an estimation into a measurement's uncertainty. Handling precise uncertainty is covered in its own section below.

Significant figures are based on the number of certain significant digits and one additional uncertain digit as read directly from the measuring device or instrument. Remember, there is no "exactly" in any real measurement in the world, even when it looks like it off of a digital readout.

Significant figures are more important in this class, and also analytical chemistry, then ever has been in previous chemistry classes. It should have been covered in your general chemistry courses, ad nausea. If you are at all still anxious or uncomfortable with significant figures and their propagation through arithmetic calculations, then you should go back and review your general chemistry materials. For your review, an excerpt from another text is posted under Canvas: "robinson_data_analysis.pdf". Also, you can watch my podcast at: http://iquanta.org/instruct, to familiarize yourself.

You will be tested on this subject to ensure you are up to speed.

Determinate/Systematic Errors

Systematic errors are usually identifiable, based on operator bad habits, or design flaws of the instrument, the manner of collecting the data, or operational incompetence. Hence systematic errors are also called determinate errors, and since they are determinant, such determinant errors are fixable. For example, in the case of operator bad habits, not having been properly trained on the use of volumetric glassware or collecting or reading the data by eye may lead to various types of biases or prejudices. Such types of biases or prejudices lead to data being skewed in a particular direction causing the data to be asymmetrical about an average, bias in the upper-only or lower-only side of the data. Professional scientists learn to diminish, reduce, null, and become aware of such systematic, determinate errors, and flaws through awareness, acknowledgment, education, understanding, standardization, calibration, practice, and mostly discipline. It becomes paramount to have integrity and be true.

Indeterminate/Random Errors and the Gaussian Distribution

We usually run an experiment multiple times because every time we run the experiment, even if it is run exactly the same way with the same amounts and chemicals, will always give some slightly different results. These variations are due to random minute effects, like the butterfly effect. It is common to run an experiment multiple times to ensure that we attain nearly the same results. We then process these multiple runs into an average, or mean. The results will have some range or deviation from each other and the standard way of expressing such a range is called the standard deviation. The standard deviation is a direct indication or representation of the precision of our measurements. The smaller

the standard deviation, the better or greater our precision, and vise versa.

The average is calculated from a general equation, given by

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N} = \mu \tag{1}$$

which, simply put, says if we add up all of our measurements, then divide by the total number or count of measurements, then that is the mean, or μ (mu). The bar over the x, or $\bar{x}$, is the standard way of expressing the average of x, x_i is a single measurement x, and N is the total count of measurements.

The standard deviation s and is given by

$$s = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \bar{x})^2}{N - 1}} = \sigma \tag{2}$$

where you can see you need to calculate the average $\bar{x}$ first before you can calculate the standard deviation, s also σ (sigma).

Lets say you calculated an average of 10.45 mL, to the proper significant figures, for some titration and then calculated the standard deviation of 0.432 mL because you ran the titration 6 separate times, you would express your final results as 10.45±0.43mL. So you would only report the average and the standard deviation to the last uncertain digit.

Sample versus Population

A population is considered the number of measurements approaching an infinite number of measurements. As an infinite number of measurements are approached, the mean and standard deviation approach pure accuracy and represent the true mean and standard deviation. Or, inclusion of an entire population includes all parameters, sectors, circumstances, and extraneous situations reflecting out the true mean, precision, and thus, standard deviation.

A sample, in contrast to population, is considered as a finite set of measurements. Sampling is indicative of the laboratory and reality for infinite measurements are costly and time consuming, if not impossible. Statisticians often qualify that a “random sampling” will more accurately represent the population wherein randomly measuring discrete objects of measurements will null out bias, prejudice, agenda, and/or straight out manipulation or conjuring.

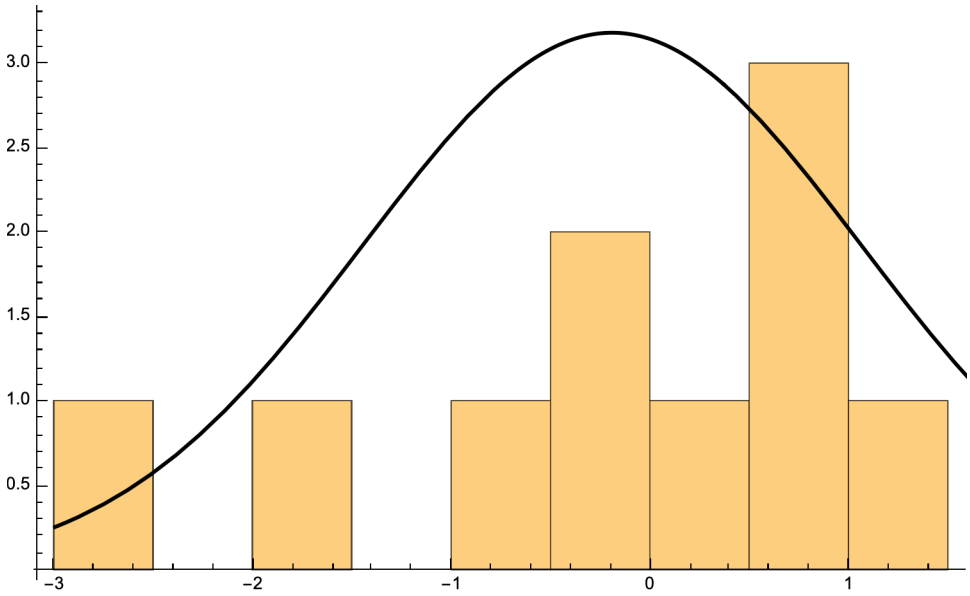
At this point, you’ll notice bold squarish lettering, below. You need not concern yourself with the specific details of that lettering for it is used to direct and create output, calculations, and graphs in *Mathematica* that are important. *Mathematica* is the program I’ve used to generate this document.

To illustrate a sampling versus a population, lets consider we take ten measurements, a sampling, running the experiment exactly the same way every time. A histogram is a “bar chart” which shows the number of measurements within that bin, as shown below in orange. A gaussian is a “bell curve” which is superimposed on top of the histogram as a black curve.

```
data = RandomVariate[NormalDistribution[0, 1], 10]
mA = Mean[data];
sA = StandardDeviation[data];
BinCounts[data, {-3, 3, 0.5}]
h = Histogram[data, 10, ImageSize → 500];
g = 1 Length[data] × PDF[NormalDistribution[mA, sA], x];
Show[h, Plot[ g, {x, -3, 3}, PlotStyle → {Thick, Black}]]
```

```
{0.733515, -0.264379, 0.699922, -1.79362, -2.78442,
 1.10535, -0.0860851, -0.719147, 0.47155, 0.669021}
```

```
{1, 0, 1, 0, 1, 2, 1, 3, 1, 0, 0, 0}
```

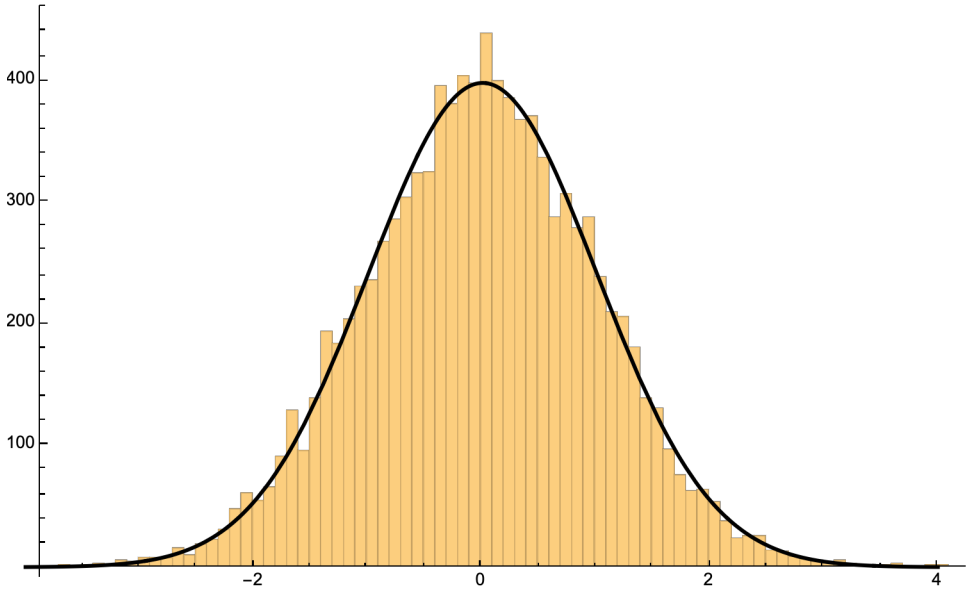


In the histogram, the x-axis shows the bins at every 0.5 increments and the y-axis shows the frequency, or the number of measurements within that measurement range. Therefore, a histogram shows the frequency versus measurement interval for each bin.

You'll also notice in the histogram, that many of our measurements sort of center around zero, with a few outliers tailing off to the left. The gaussian has its maximum right around zero and its width represents the spread, precision, or standard deviation of the measurements.

If we ran the same experiment 10,000 times and increased our bins, which would make our bin increment or range much smaller for each bin, the histogram and gaussian would look like:

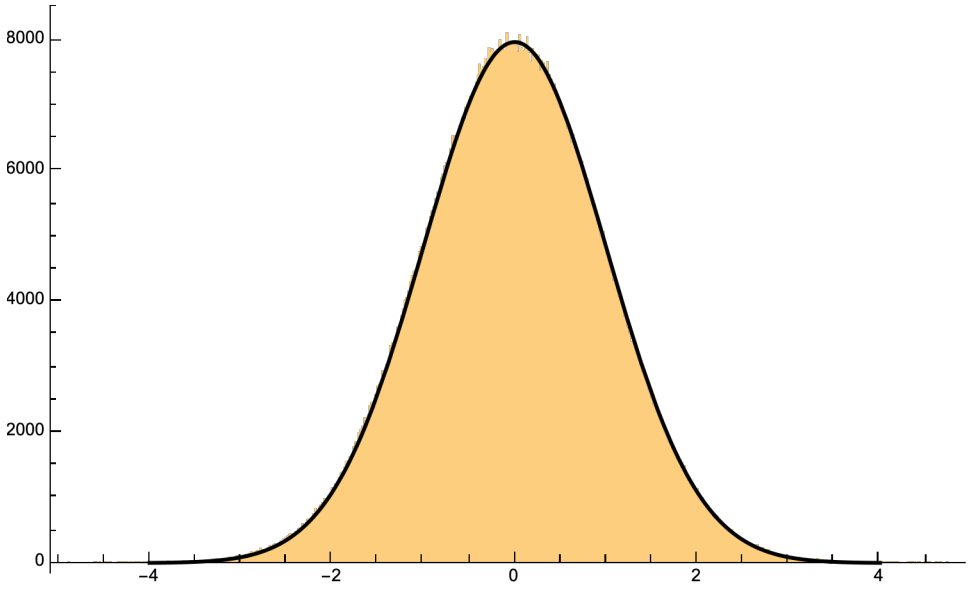
```
data = RandomVariate[NormalDistribution[0, 1], 1*^4];
mA = Mean[data];
sA = StandardDeviation[data];
h = Histogram[data, 100, ImageSize → 500];
g = .1 Length[data] × PDF[NormalDistribution[mA, sA], x];
Show[h, Plot[ g, {x, -4, 4}, PlotRange → All, PlotStyle → {Thick, Black}]]
```



where you'll notice that the shape of the histogram is starting to approach that of the gaussian.

And finally, if we ran the experiment 1,000,000 (a million) times and further subdivide our bins again, we would get:

```
data = RandomVariate[NormalDistribution[0, 1], 1*^6];
mA = Mean[data];
sA = StandardDeviation[data];
h = Histogram[data, 500, ImageSize → 500];
g = .02 Length[data] × PDF[NormalDistribution[mA, sA], x];
Show[h, Plot[ g, {x, -4, 4}, PlotRange → All, PlotStyle → {Thick, Black}]]
```



So, acquiring a million measurements approaches that of the population. Its not just a sampling of the population, but it is the population and the histogram is very close to that of the gaussian. Therefore, the gaussian is the mathematical expression of a histogram that represents the entire population, both in the true mean and standard deviation.

But then again, who can afford to spend the time or money running 10,000 or a million experiments running up to a full population? Sampling a few measurements and running it up against the nature of the gaussian will allow us to save time and money.

A Practical Illustration of the Bell Curve

Lets illustrate the concept of a set centered around a mean. It creates the classic bell curve seen in grades. Here is the data from my last semester's grade from my three classes.

```
nbd = NotebookDirectory[];
dataT = Import[nbd <> "gT.txt", "List"];
dataR = Import[nbd <> "gR.txt", "List"];
dataF = Import[nbd <> "gF.txt", "List"];
data = Join[dataT, dataR, dataF]
```

```
{61.67, 93.56, 77.22, 73.56, 47.78, 37.94, 93.83, 84., 93.83, 87., 85.89,
 91.17, 84.44, 91.67, 90., 70.06, 73.59, 7.35, 90.71, 92.29, 78.71, 89.76,
 95.18, 13.53, 80.88, 2.59, 93.35, 86.29, 85.94, 76., 76.88, 98.29, 95.65,
 70.74, 68.8, 75.6, 90.17, 85.77, 88.91, 65.6, 73.71, 85.31, 54.06, 79.89,
 92.17, 85.94, 80.51, 70.8, 70.23, 60.46, 89.6, 84.17, 58.34, 62.86}
```

the mean and standard deviation of the collection of final grades are:

```
mA = Mean[data];
sA = StandardDeviation[data];
Print["All (", Length[data], "): Mean: ", mA, ", Std.Dev.: ", sA, "."]
```

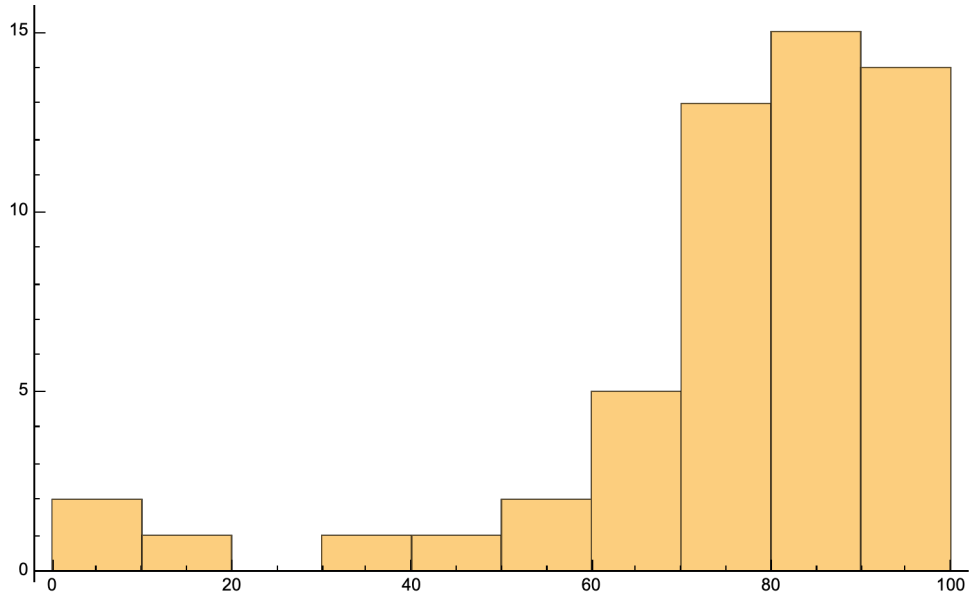
```
All (54): Mean: 75.8194, Std.Dev.: 21.0967.
```

hey, that is a pretty good average, isn't it? anything over 70% fends well for the professor. good.

Lets plot the data in a bar chart, i.e., a histogram, which has the count in increments of a letter grade, or every 10 points.

```
BinCounts[data, {0, 100, 10}]
h = Histogram[data, 10, ImageSize → 500]
```

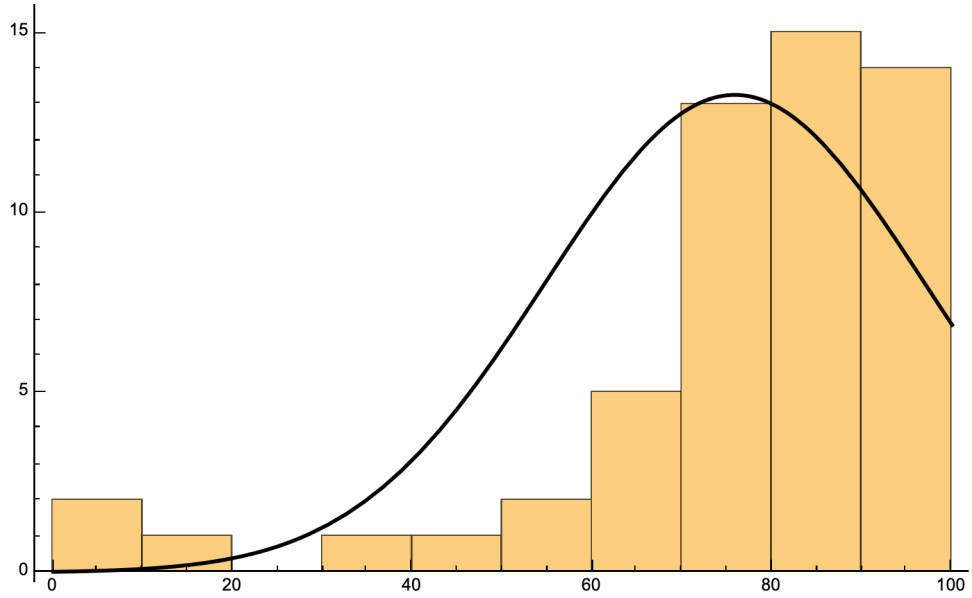
```
{2, 1, 0, 1, 1, 2, 5, 13, 15, 14}
```



Kind-of a rough bell curve, but our grading system is going to be skewed, even biased, by the very nature that a successful grade is average at 70% and the upper limit is 100%. An unskewed curve would have an average at 50%, not at 70%. Nonetheless, lets work with the system as it is.

The mathematical function that represents a standard symmetric bell curve is called the Gaussian function. In *Mathematica*, and fitting it to my grade data, and superimposing it on the histogram looks like:

```
g = 13 Length[data] × PDF[NormalDistribution[mA, sA], x];
Show[h, Plot[ g, {x, 0, 100}, PlotStyle → {Thick, Black}]]
```



ok, so the illustration that the gaussian is a smoothed out histogram should be beginning to make sense.

Example 1-2, Page 972

There are other ways to calculate the standard deviation, as shown on page 972 in your Skoog text, however, you rarely should have to calculate the average and standard deviations manually. All scientific calculators, spreadsheets, and mathematical and statistical software and apps can do these for you, automatically. All you need to learn is how to enter in the raw data into your calculator, for instance, and then push the average and standard deviation buttons and it will give you the answer. You should definitely teach yourself how to do all of these data analysis calculations in Microsoft Excel because Excel is pervasive throughout industry, research, academics, and it is readily available to you. Its practically free. It is installed on these lab computers and upstairs in the learning commons.

Here is an example using Example a1-2 on page 972 under *Mathematica:*

```
data = {0.752, 0.756, 0.752, 0.751, 0.760}
Mean[data];
StandardDeviation[data];
Out[-1] / Out[-2] 100;
Print["Mean = x̄ = ", Out[-3],
 " & Standard Deviation = σ = ", Out[-2], "; ", Out[-1], "%"]
```

```
{0.752, 0.756, 0.752, 0.751, 0.76}
```

```
Mean = x̄ = 0.7542 & Standard Deviation = σ = 0.00376829; 0.499641%
```

so my final result would be reported as 0.754±0.004 ppm Pb. And it is that simple, even in Excel. So teach yourself how to do that in Excel.

Gaussian or Bell Curves of Random Errors

Random or indeterminate errors are given by the symmetric Gaussian curve, commonly referred to as a bell curve. It is given by the equation:

$$\frac{dN}{N} = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} dx \tag{3}$$

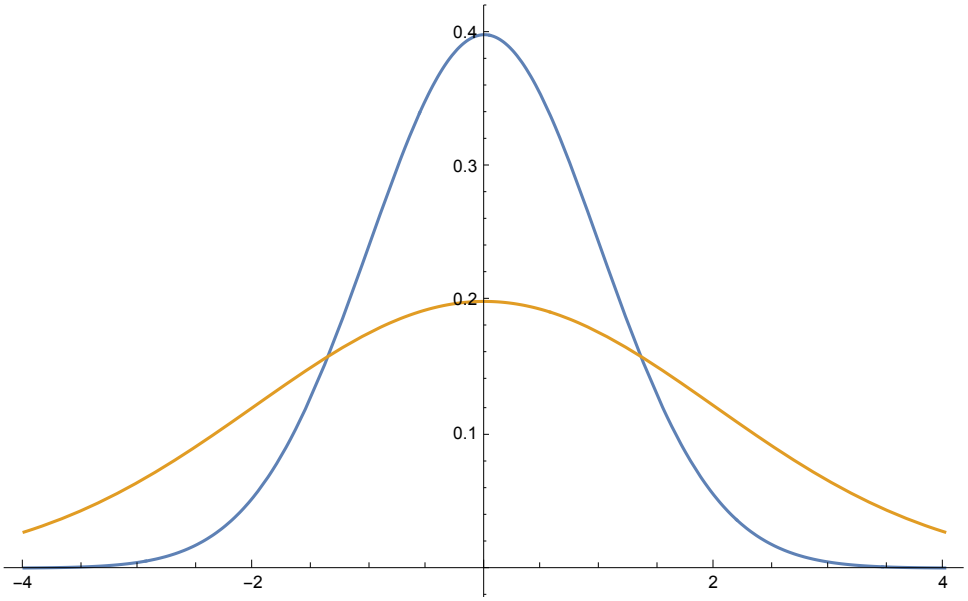
where N is the total number of samples, σ the standard deviation, and μ the population average.

Here is the kick, although the equation and the Gaussian expression (right-hand side) looks complex, it really isn't. The Gaussian expression on the right-hand-side of the above equation 3 has only one unknown variable, which is the variable x. 1, 2, π, σ, and μ are all known. 1, 2, and π are just numbers, where π is 3.14159. When you input raw data into Excel and have it do an average, i.e., (=average(A1:A6)), and a standard deviation, i.e., (=stdev(A1:A6)), then you also know the μ and σ, respectively. So essentially I am saying that by having your calculator or Excel calculate the average (mean), μ, and standard deviation, σ, then you are finding the fitting parameters of your histogramed data for the Gaussian for your data. Plotting, or superimposing, the fitted Gaussian against your histogramed data will correlate to each other. The histogram shows you the discrete dataset and the Gaussian represents the data if you had an infinite number of those representative data points in the dataset. Fitting parameters are covered in a lot more detail in the "Curve Fitting and Method of Least Squares" section down below.

Entering the expression and plotting gives the Gaussian or bell curve...

```
dNNx = 1 / (σ Sqrt[2 Pi]) Exp[-(x - μ)^2 / (2 σ^2)]
Plot[{dNNx /. {σ → 1, μ → 0}, dNNx /. {σ → 2, μ → 0}}, {x, -4, 4}, ImageSize → 500]
```

$$\frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi}\,\sigma}$$



or by normalizing the above Gaussian setting to the relative standard deviation or

$$z = \frac{x - \mu}{\sigma} \tag{4}$$

and taking its derivative so that we can substitute or

$$dz = \frac{dx}{\sigma} \tag{5}$$

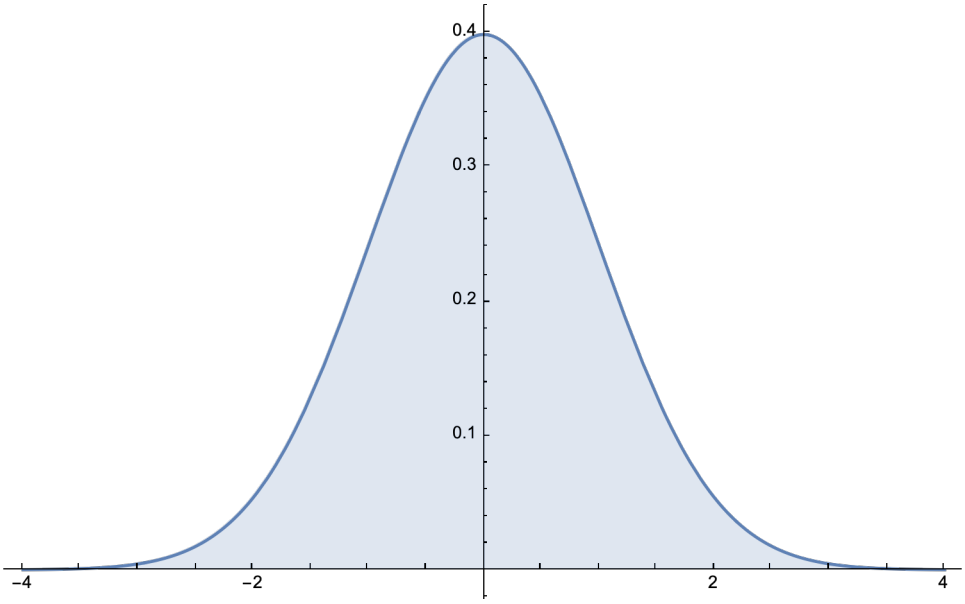
giving

$$\frac{dN}{N} = \frac{1}{\sqrt{1\,\pi}}\, e^{-z^2/2}\, dz \tag{6}$$

which is the normal, normalized, or unity Gaussian curve or the normal error curve.

```
dNNz = 1 / (Sqrt[2 Pi]) Exp[-z^2 / 2]
Plot[dNNz, {z, -4, 4}, Filling → Axis, ImageSize → 500]
```

$$\frac{e^{-\frac{z^2}{2}}}{\sqrt{2\,\pi}}$$



and integrating, finding the area under the curve, over the entire normal Gaussian should give the value of unity or one because that is the definition of “normal” or

```
Integrate[dNNz, {z, -∞, +∞}]
```

1

where the area should be 1 or 100%, for y-axis, or height of the bars, of a histogram should be the frequency or percentage out of 100%. Adding the height of each bar of a normalized histogram should add to 100%.

And so when the deviation is 1 out from the mean (z=1), the area under the curve covers 68.3% of the total area under the curve or

```
NIntegrate[dNNz, {z, -1, +1}]
```

0.682689

```
NIntegrate[dNNz, {z, -2, +2}]
```

0.9545

or 95.4% for a normal deviation of 2, and 3 would be 99.7% or

```
NIntegrate[dNNz, {z, -3, +3}]
```

0.9973

```
NIntegrate[dNNz, {z, -4, +4}]
```

0.999937

or reiterated in a table format

```
Table[{σ, ci → NIntegrate[dNNz, {z, -σ, +σ}]}, {σ, {.5, 1, 2, 3, 3.5, 4, ∞}}]
```

$$\begin{pmatrix} 0.5 & \text{ci} \to 0.382925 \\ 1 & \text{ci} \to 0.682689 \\ 2 & \text{ci} \to 0.9545 \\ 3 & \text{ci} \to 0.9973 \\ 3.5 & \text{ci} \to 0.999535 \\ 4 & \text{ci} \to 0.999937 \\ \infty & \text{ci} \to 1. \end{pmatrix}$$

These percentages are known as confidence levels or intervals. So when I say I am 99% confident, then I am saying that I am 99% confident that my measurements have a 99% chance of reflecting the true real value. And that, I am also only 1% sure that my results do not reflect the real value.

Since we have calculated the area under the curve between various deviations, as above, what about doing it in the reverse? The reverse would be calculating the deviations from various confidence intervals. This is done by

```
Integrate[dNNz, {z, -σ, σ}] == ci
Table[{ci, Solve[%, σ]}, {ci, {.5, .8, .9, .95, .99, .999, .99993, 1}}]
```

$$\text{erf}\left(\frac{\sigma}{\sqrt{2}}\right) = \text{ci}$$

$$\begin{pmatrix} 0.5 & \{\{\sigma \to 0.67449\}\} \\ 0.8 & \{\{\sigma \to 1.28155\}\} \\ 0.9 & \{\{\sigma \to 1.64485\}\} \\ 0.95 & \{\{\sigma \to 1.95996\}\} \\ 0.99 & \{\{\sigma \to 2.57583\}\} \\ 0.999 & \{\{\sigma \to 3.29053\}\} \\ 0.99993 & \{\{\sigma \to 3.97629\}\} \\ 1 & \{\{\sigma \to \infty\}\} \end{pmatrix}$$

where a is the minus/plus deviation σ as reproduced on page 977 of the Skoog text. The above output shows that being 90% confident is also 1.6 standard deviations out from the central mean. Similarly, 99% confident is 2.6 standard deviations.

```
1 / √2 == √2 / 2
```

True

More on Gaussian.. .

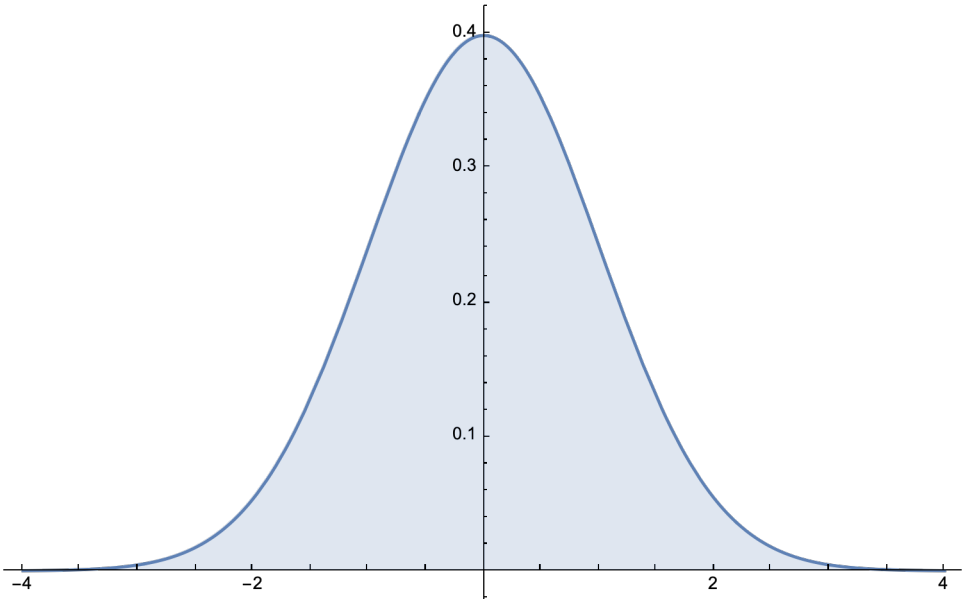
Mathematica has a bunch of built-in functions just for statistics analysis and probability distributions. The following are simply to demonstrate that power of *Mathematica*, you won't be directly responsible for the content of this section, but wander through just to pick up a few extra understandings.

The NormalDistribution function is just the Gaussian function and PDF is the probability distribution function. The Out function grabs the output of *Mathematica*, so Out[-2] pulls the second to last output.

```
PDF[NormalDistribution[], z]
Table[Out[-1], {z, 0, 4}] // N
Plot[Out[-2], {z, -4, 4}, Filling → Axis, ImageSize → 500]
```

$$\frac{e^{-\frac{z^2}{2}}}{\sqrt{2\,\pi}}$$

```
{0.398942, 0.241971, 0.053991, 0.00443185, 0.00013383}
```



The CDF function is the cumulative distribution function that basically returns the integrated area under that distribution. So in the case of a normalized Gaussian, the CDF would return unity or 1, as shown

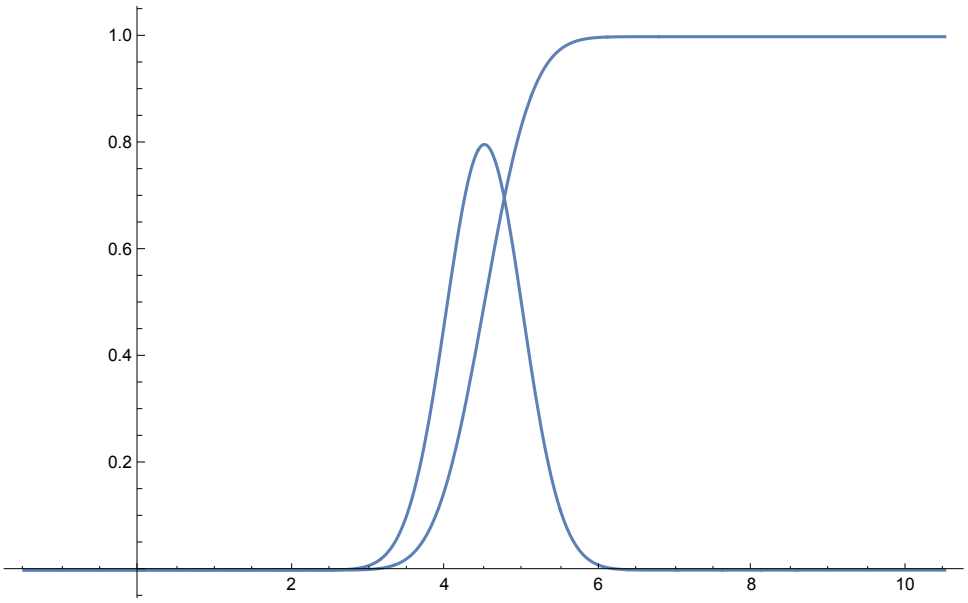
```
pdf = PDF[NormalDistribution[μ, σ], x]
cdf = CDF[NormalDistribution[μ, σ], x]
cdf /. {μ → 4.5, σ → .5, x → 9}
cdf /. {μ → 4.5, σ → .5, x → 4.5}
Plot[{pdf, cdf} /. {μ → 4.5, σ → .5}, {x, -1.5, 10.5}, ImageSize → 500]
```

$$\frac{e^{-\frac{(x-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi}\,\sigma}$$

$$\frac{1}{2}\,\text{Erfc}\left[\frac{-x+\mu}{\sqrt{2}\,\sigma}\right]$$

```
1.
```

```
0.5
```



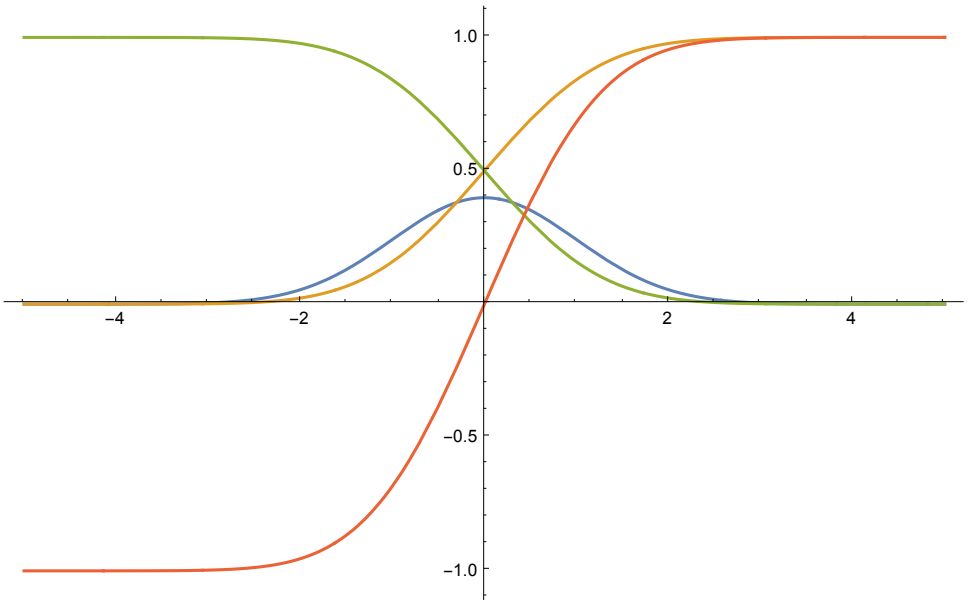
I am also showing above that the mean is going to be centered around some value. In this case I superficially forced it to be at 4.5.

And finally, below, I prove the confidence levels at different intervals of deviations for a normalized Gaussian curve.

```
PDF[NormalDistribution[], z]
CDF[NormalDistribution[], z];
CDF[NormalDistribution[], -z];
N[Table[{z, Out[-2], Out[-1], Out[-2] - Out[-1]}, {z, 0.5, 4.5, 0.5}]]
Plot[{Out[-4], Out[-3], Out[-2], Out[-3] - Out[-2]}, {z, -5, 5}, ImageSize → 500]
```

$$\frac{e^{-\frac{z^2}{2}}}{\sqrt{2\,\pi}}$$

```
{{0.5, 0.691462, 0.308538, 0.382925},
 {1., 0.841345, 0.158655, 0.682689}, {1.5, 0.933193, 0.0668072, 0.866386},
 {2., 0.97725, 0.0227501, 0.9545}, {2.5, 0.99379, 0.00620967, 0.987581},
 {3., 0.99865, 0.0013499, 0.9973}, {3.5, 0.999767, 0.000232629, 0.999535},
 {4., 0.999968, 0.0000316712, 0.999937},
 {4.5, 0.999997, 3.39767 × 10^-6, 0.999993}}
```



Propagation of Measurement Uncertainties

Every measurement is uncertain and can never be “exact”. So never say the measurement is “exact”.

Significant figures, as mentioned above, are an **estimation** of the uncertainties of measurements. Sig.figs., by definition, are an estimation of the uncertainty because the measurement contains as many certain digits as the measurement allows **and** one extra uncertain digit in the last place. That last place uncertain digit is assumed to vary by roughly 10% in that last place. In that assumption is the reason sig.figs. are an estimation.

However, a good scientist, researcher, or engineer will understand their measuring devices well enough to not estimate, but determine the range of uncertainties in each and every measurement. Uncertainties are expressed along with each measurement with an additional ± followed by the range or uncertainty, and finally, the proper units of both. So a typical measurement would look like 5.43±0.25 m, or (8.432±0.865)x10^8 s, or (8.56754±1.83415)x10^3 g. Notice the uncertainty would be rounded to match the precision or decimal place of the measurement itself.

When calculations or computations are done with measurements with uncertainty, then the uncertainty needs to propagate or move through the computations properly so that the expressed uncertainty properly reflects the new magnitude of the measurement. This section is about properly propagating the uncertainty down through a final answer.

Absolute uncertainty is expressed as:

$$\begin{aligned}&\text{value} \pm \text{uncertainty units}\\&\text{value } (\pm \text{uncertainty}) \text{ units}\end{aligned} \tag{7}$$

the parentheses are usually optional. Examples would be:

$$\begin{aligned}&657.3 \pm 0.4\ \text{mL}\\&8.422 \times (\pm 0.042)\ \text{x}10^3\ \text{g}\\&8.12 \times (\pm 1.50)\ \text{x}10^5\ \text{km}\end{aligned} \tag{8}$$

Relative uncertainty is similar as the above absolute except its expressed with percentages as:

$$\begin{aligned}&\text{value} \pm \text{uncertainty} \,\%\, \text{units}\\&\text{value } (\pm \text{uncertainty} \,\%)\, \text{units}\end{aligned} \tag{9}$$

Absolute uncertainty is converted to relative uncertainty by:

$$\pm\text{uncertainty} \,\%\, = \frac{\pm\text{uncertainty}}{\text{value}} \times 100 \tag{10}$$

and, relative uncertainty is converted to absolute uncertainty by:

$$\pm\text{uncertainty} = \frac{\pm\text{uncertainty} \,\%}{100} \times \text{value} \tag{11}$$

The above absolute examples would have their relative uncertainties as:

$$\begin{aligned}&657.3 \pm 0.4\ \text{mL} = 657.3 \pm 0.06 \times \%\ \text{mL}\\&8.422 \times (\pm 0.042)\ \text{x}10^3\ \text{g} = 8.422 \times (\pm 0.50 \times \%)\ \text{x}10^3\ \text{g}\\&8.12 \times (\pm 1.50)\ \text{x}10^5\ \text{km} = 8.12 \times (\pm 18.4 \times \%)\ \text{x}10^5\ \text{km}\end{aligned} \tag{12}$$

The general equations for the propagation of errors are:

$$\text{Addition / Subtraction}: e_4 = \sqrt{e_1^2 + e_2^2 + e_3^2}$$
$$\text{Multiplication / Division}: \%\, e_4 = \sqrt{(\%\, e_1)^2 + (\%\, e_2)^2 + (\%\, e_3)^2} \quad (13)$$

where e is the absolute uncertainty used in addition and subtraction and %e is the relative uncertainty used in multiplication and division.

This section is best done with practice, practice, and practice. I want you to read and absorb the section, and then practice, practice, and practice until your eyes are bleeding out of your skull, nice right? I will practice a bit here doing the first example from Skoog:

Example a1-7, Page 981

```
16 (tR / W)^2
sN = Sqrt[D[%, tR]^2 stR^2 + D[%, W]^2 sW^2]
%% /. {tR → 13.36, W → 2.18}
%% /. {tR → 13.36, W → 2.18, stR → 0.043, sW → 0.061}
```

$$\frac{16\, t_R^2}{W^2}$$

$$\sqrt{\frac{1024\, s_{t_R}^2\, t_R^2}{W^4} + \frac{1024\, s_W^2\, t_R^4}{W^6}}$$

```
600.925

33.8515
```

opps, that calculated standard deviation of 34 is not the same answer as calculated in the text on page 981. did i make a mistake?

looking very carefully.. . i think that they inadvertently swapped the s_w and s_{t_R} values when calculating the s_n. that is a simple edit in *Mathematica* or

```
sN /. {tR → 13.36, W → 2.18, stR → 0.061, sW → 0.043}

24.333
```

which is the answer in the text and it does verify that the text swapped the s_w and s_{t_R} values, but that is the substitution. so, yes, there is a mistake or perhaps “typo” in the text. and that happens everywhere in science and life. so be a clear thinker and watch for it.

from Skoog:

Problem a1-11, Page 990

I present these solutions to help you practice these problems to get the same results. The original problem states to round your answer to only the significant digits, but I believe that is rather odd and irregular. Therefore, I present these answers to the regular reporting form of the standard sig.fig. way, which is all certain digits followed by one uncertain digit...

a) y = 1.02(±0.02) x 10^{-8} - 3.54(±0.2) x 10^{-9}

```
1.02 × 10^-8 - 3.54 × 10^-9
Sqrt[(0.02 × 10^-8)^2 + (0.2 × 10^-9)^2]
Print["CV = ", NumberForm[% / %% × 100, 3], "%"]
```

6.66×10^{-9}

2.82843×10^{-10}

CV = 4.25%

so the rule for sig.figs. for addition/subtraction is carry down the least precision or decimal place. watch the exponents, so 1.02×10^{-8} is like 10.2×10^{-9}, so we have to carry down the first decimal. Thus the final answer would be $\mathbf{6.7 \times (\pm 0.3) \times 10^{-9}}$. And finally, the coefficient of variation is given by the above CV of 4%.

b) y = 90.31(±0.08) - 89.32(±0.06) + 0.200(±0.004)

```
90.31 - 89.32 + .2
Sqrt[0.08^2 + 0.06^2 + 0.004^2]
Print["CV = ", NumberForm[% / %% × 100, 3], "%"]
```

1.19

0.10008

CV = 8.41%

and again watch our sig.figs. and carrying down the least precision to the 2nd decimal, the final result should look like **1.2(±0.1)**.

c) y = 0.0020(±0.0005) x 20.20(±0.02) x 300(±1)

```
0.002 × 20.20 × 300
Sqrt[(0.0005 / 0.002)^2 + (0.02 / 20.20)^2 + (1 / 300)^2] %
Print["CV = ", NumberForm[% / %% × 100, 3], "%"]
```

12.12

3.03029

CV = 25.%

that original 300 is very vexing and in poor form because we can not tell, with certainty, if it is to 1, 2, or 3 sig.figs. i will "assume" 2 just for ease. the rule for carrying multiple/divide sig.figs. is to carry through the least count of sig.figs, which is 2 sig.figs. in this case. The final result should be reported as **12(±3)**.

d) y = 163(±0.03) x 10^{-14} / 1.03(±0.04) x 10^{-16}

```
163*^-14 / 1.03*^-16
```

$$(163 \times 10^{-14}) / (1.03 \times 10^{-16})$$

```
Sqrt[(0.03*^-14 / 163*^-14) ^ 2 + (0.04*^-16 / 1.03*^-16) ^ 2] * Out[-1]
```

$$\text{Sqrt}\left[\left((0.03 \times 10^{-14}) / (163 \times 10^{-14})\right)^2 + \left((0.04 \times 10^{-16}) / (1.03 \times 10^{-16})\right)^2\right] * \text{Out}[-2]$$

```
Print["CV = ", NumberForm[Out[-1] / Out[-3] * 100, 3], "%"]
```

15 825.2

15 825.2

614.579

614.579

CV = 3.88%

the final answer should be to 3 sig.figs, or 15800, but that again is ambiguously communicated and in bad form because we are not clearing expressing the answer to 3 sig.figs., but actually 3, 4, or 5. to communicate it clearly, this answer has to be in scientific notation, or $\mathbf{1.58 \times (\pm 0.06) \times 10^4}$.

e) y = 100(±1) / 2(±1)

```
100 / 2
N[Sqrt[(1 / 100)^2 + (1 / 2)^2] %]
Print["CV = ", NumberForm[% / %% * 100, 3], "%"]
```

50

25.005

CV = 50.%

this time the answer is to 1 sig.fig. and watching that we are communicating that clearly, the final answer would be $\mathbf{5. \times (\pm 3) \times 10^1}$.

f) $y = (2.45(\pm 0.02) \times 10^{-2} - 5.06(\pm 0.06) \times 10^{-3}) / (23.2(\pm 0.7) + 9.11(\pm 0.08))$

```
ans = ((2.45 × 10^-2) - (5.06 × 10^-3)) / (23.2 + 9.11)
num = ((2.45 × 10^-2) - (5.06 × 10^-3))
denom = (23.2 + 9.11)
snum = Sqrt[(0.02 × 10^-2)^2 + (0.06 × 10^-3)^2]
sdenom = Sqrt[0.7^2 + 0.08^2]
Sqrt[(snum / num)^2 + (sdenom / denom)^2] ans
Print["CV = ", NumberForm[% / ans × 100, 3], "%"]
```

```
0.000601671

0.01944

32.31

0.000208806

0.704557

0.0000146254

CV = 2.43%
```

and being very very careful to carry through both the addition/subtraction and the multiple/divide rules, when appropriate, for sig.figs., we should report the final answer to 3 sig.figs. and it should be $\mathbf{6.02 \times (\pm 0.15) \times 10^{-4}}$.

from Harris :

worked example, Section 3-4 Propagation of Uncertainty from Random Errors, Mixed Operations:

$$\frac{[1.76 \pm (0.03) - 0.59 \pm (0.02)]}{1.89 \pm (0.02)}$$

```
ans = (1.76 - .59) / 1.89
num = 1.76 - .59
snum = Sqrt[0.03^2 + 0.02^2]
uncertainty = Sqrt[(snum / num)^2 + (0.02 / 1.89)^2] * ans
Print["CV = ", NumberForm[% / ans × 100, 3], "%"]
```

```
0.619048

1.17

0.0360555

0.0201704

CV = 3.26%
```

which is the same answer in Harris or to two sig.figs. 0.62±0.02 or 0.62±3.3%.

and some further examples:

3-15a) 6.2±0.2 - 4.1±0.1

```
ans = 6.2 - 4.1
uncertainty = Sqrt[.2^2 + .1^2]
Print["CV = ", NumberForm[% / ans × 100, 3], "%"]
```

```
2.1
```

```
0.223607
```

```
CV = 10.6%
```

or 2.1±0.2 or 2.1±11%

3-15b) 9.43±0.05 x 0.016±0.001

```
ans = 9.43 * 0.016
uncertainty = Sqrt[(0.05 / 9.43)^2 + (0.001 / 0.016)^2] * ans
Print["CV = ", NumberForm[% / ans × 100, 3], "%"]
```

```
0.15088
```

```
0.00946387
```

```
CV = 6.27%
```

or 0.151±0.009 or 0.151±6.27%

3-15c) [6.2±0.2 - 4.1±0.1] / 9.43±0.05

```
ans = (6.2 - 4.1) / 9.43
num = 6.2 - 4.1
snum = Sqrt[.2^2 + .1^2]
uncertainty = Sqrt[(snum / num)^2 + (0.05 / 9.43)^2] * ans
Print["CV = ", NumberForm[% / ans × 100, 3], "%"]
```

```
0.222694
```

```
2.1
```

```
0.223607
```

```
0.0237417
```

```
CV = 10.7%
```

or 0.22±0.02 or 0.22±11%

3-15d) 9.43±0.05 x [6.2 × (±0.2) x10^{-3} + 4.1 × (±0.1) x10^{-3}]

```
ans = 9.43 * (6.2*^-3 + 4.1*^-3)
add = 6.2 + 4.1
sadd = Sqrt[.2^2 + .1^2]
uncertainty = Sqrt[(sadd / add)^2 + (.05 / 9.43)^2] * ans
Print["CV = ", NumberForm[% / ans × 100, 3], "%"]
```

```
0.097129
```

```
10.3
```

```
0.223607
```

```
0.00217059
```

```
CV = 2.23%
```

or 0.097±0.002 or 0.097±2%

In-Class / Lecture Examples:

548.6(±0.4)+74(±1)-16.453(±0.009)-304.2(±4.3)

```
ans = 548.6 + 74 - 16.453 - 304.2
uncertainty = Sqrt[.4^2 + 1^2 + 0.009^2 + 4.3^2]
Print["CV = ", NumberForm[% / ans × 100, 3], "%"]
```

```
301.947
```

```
4.43284
```

```
CV = 1.47%
```

or 301±4 or 301±1%.

[89.4(±0.4)*72.64(±0.12)]/66.6(±0.6)

```
ans = 89.4 * 72.64 / 66.6
uncertainty = Sqrt[(.4 / 89.4)^2 + (.12 / 72.64)^2 + (.6 / 66.6)^2] * ans
Print["CV = ", NumberForm[% / ans × 100, 3], "%"]
```

```
97.5077
```

```
0.993959
```

```
CV = 1.02%
```

or 98±1 or 98±1%.

Significant Figures Revisited

The REAL rule for significant figures is:

The first digit (or decimal place) of the absolute uncertainty is the last significant digit (or decimal place) in the final answer (or value).

Example:

0.002364±0.000003/0.02500±0.00005

```
ans = .002364 / .025
uncertainty = Sqrt[(.000003 / .002364)^2 + (.00005 / .025)^2] * ans
```

```
0.09456
```

```
0.000223979
```

which properly rounded the final answer would be 0.0946±0.0002. OR, since the 4th decimal place is the first place of the uncertainty, then the final answer would also be rounded to the 4th decimal place, or 0.0946 because the uncertainty is more properly computed from certain uncertainty. Under "normal" or "standard" sig.figs. rule the answer should be to 4 sig.figs., but because the uncertainty is known to certainty, or to within certain ranges, then the uncertainty can be accurately computed and its placement followed through to the final answer with certainty.

Remember sig.figs. are an estimation on the uncertainty but do not express the fully certain uncertainty. So, when the uncertainty is fully known, then the sig.figs. in the final answer will be formally and properly adjusted to reflect the real rule.

Here is the opposite example:

0.821±0.002/0.803±0.002

```
ans = .821 / .803
uncertainty = Sqrt[(.002 / .821)^2 + (.002 / .803)^2] * ans
Sqrt[(.002 / .821 * 100)^2 + (.002 / .803 * 100)^2] * ans / 100
```

```
1.02242
```

```
0.00356202
```

```
0.00356202
```

or 1.022±0.004. Where we can express the final answer to 4 sig.figs. because the absolute uncertainty is to the 3rd decimal place, even though the "standard" or "normal" sig.figs. rules should have the reporting to only 3 sig.figs.

Hypothesis Testing

Hypothesis testing comes down to the acceptance or rejection of a measurement or set of measurements because it is suspected to be outside the range of the majority of other measurements made in the experiment. Such a point or dataset is called an outlier and it is suspect until that data point is put through a quantitative test. Such a set of measurements may be suspected to have bias and thus determinate errors.

As scientists, we can not just dismiss or reject potential outliers without quantitative justification and reason. We can not throw out any data because it feels wrong or we don't like, especially if we feel the experiment is properly designed and running. The quantitative method is a special branch of statistics called hypothesis testing. Essentially, the hypothesis is the suspected measurement that is put into question, then it is put through a quantitative statistical computation, and then the data point is either accepted or rejected based on that quantitative test. If it is accepted, the point must be included in the dataset, if rejected, it is removed and usually another run is performed.

There are many types of hypothesis testing: Dixon's Q-test, Student's T-test, and Fisher's F-test are the most common. Some tests are best suited for single measurement confirmation, and others for set of measurement or bias confirmation. The Grubbs test is best to test a single outlying point within a set of measurements or dataset. The Student's T-test is best when comparing an entire dataset or set of measurements against a true or accepted value. Both are given below as examples, respectively.

Verifying Acceptance or Rejection of a Single Measurement in a Dataset

Most modern analytical chemistry use the Grubbs test for an outlier and it happens to be fairly simple to use also.

As an aside, there are a few tests for single and multiple outliers. The Skoog text explains the Dixon's Q-test and the Harris text replaced its section on the Q-test with the Grubbs test referencing the International Standards Organization and the American Society for Testing and Materials (ASTM) prefer the Grubbs over the Dixon's Q-test. Additionally, the National Institute of Standards and Technology (NIST) recognize both the Q-test and the Grubbs test as valid, but prefer Grubbs over Dixon, as stated under: https://itl.nist.gov/div898/handbook/eda/section3/eda35h.htm. This lecture will continue with the Grubbs test but simply be aware that there are a few tests available for outliers testing.

Essentially, to test a minimum or maximum outlier, use

$$G_{calculated} = \frac{\text{abs}\left[\left(\text{min} \mid \text{max questionable value}\right) - \bar{x}\right]}{s} \tag{14}$$

where the questionable value is subtracted from the mean, $\bar{x}$, and divided by the standard deviation, s, both of which are calculated with inclusion of the questionable value. Also, notice that the subtraction is surrounded by the absolute value, so the value for $G_{calculated}$ will always be positive. If $G_{calculated} > G_{critical}$ then the questionable value may be rejected and excluded from the dataset, where $G_{critical}$ is a value that is looked up in the below table of critical values for G.

In[108]:=
```
gcritical[n_, CL_] := Module[{ddof, Tcritical}, (
    (*CL is for the one-side for ASTM recommends one-
      tailed test for Grubbs outliers.  α=1-CL/100.,
    single-sided significance value is α/n.  double-sided would have been α/2n.*)
    ddof = n - 2;
    Tcritical =
     InverseSurvivalFunction[StudentTDistribution[ddof], (1 - CL / 100.) / n];
    Gcritical = (n - 1) / Sqrt[n] * Sqrt[Tcritical^2 / (n - 2 + Tcritical^2)]
  )]
gci[n_, g_] := Module[{a, gg}, (
    (*CL is for the one-side for ASTM recommends one-
      tailed test for Grubbs outliers.  α=1-CL/100.,
    single-sided significance value is α/n.  double-sided would have been α/2n.*)
    a = (Sqrt[n] * g / (n - 1))^2;
    gg = Sqrt[(a * n - a * 2) / (1 - a)];
    Gci = 100 * (1 - n * SurvivalFunction[StudentTDistribution[n - 2], gg])
  )]
(*little tests*)
gci[3, 1.15470]
gci[6, 1.72888]
gtest[data_?VectorQ, mode : Min | Max, CL : 50 | 80 | 90 | 95 | 98 | 99 | 99.5 | 99.9] :=
 Module[{mean, sd, value, Gcritical, Gcalculated, bb},
  If[Length[data] < 1,
   Print["dataset must have more than 1 points"],
   (mean = Mean[data];
    sd = N[StandardDeviation[data]];
    value = If[mode === Min, Min[data], Max[data]];
    Gcritical = gcritical[Length[data], CL];
    Gcalculated = N[Abs[value - mean] / sd];
    bb = If[Gcalculated > Gcritical, False, True];
    txt = "{ N: " <> ToString[Length[data]] <> ", x̄: " <>
      ToString[mean] <> ", s: " <> ToString[sd] <> ", CL: " <> ToString[CL] <>
      "%, G_calculated: " <> ToString[Gcalculated] <> ", G_critical: " <>
      ToString[Gcritical] <> ", DataPoint: " <> ToString[value] <> " }";
    If[bb,
     Print[txt, Style[" must be Accepted and Retained for Inclusion.", Bold]],
     Print[txt , Style[" may be Rejected and Excluded.", Bold]]
    ]
   )]]
Table[NumberForm[gcritical[n, CL], {10, 5}],
  {n, 3, 25}, {CL, {50, 80, 90, 95, 98, 99, 99.5, 99.9}}];
Join[{{"n", "50%CL", "80%CL", "90%CL", "95%CL", "98%CL", "99%CL", "99.5%CL",
    "99.9%CL"}}, Table[Join[{n}, Table[NumberForm[gcritical[n, CL], {10, 5}],
     {CL, {50, 80, 90, 95, 98, 99, 99.5, 99.9}}]], {n, 3, 25}]];
Print[Style["Grubb's Critical Values (G_crit)", Bold, FontSize → 20]]
Grid[%%, Alignment → Right, Frame → All,
 Background → {{LightGray, None}, {LightRed, None}}, ItemStyle → {1 → Bold}]
```

Out[110]= 99.9078

Out[111]= 90.

Grubb's Critical Values (G_{crit})

Out[116]=

n	50%CL	80%CL	90%CL	95%CL	98%CL	99%CL	99.5%CL	99.9%CL
3	1.00000	1.12947	1.14837	1.15312	1.15445	1.15464	1.15468	1.15470
4	1.12500	1.35000	1.42500	1.46250	1.48500	1.49250	1.49625	1.49925
5	1.22903	1.48971	1.60163	1.67139	1.72528	1.74886	1.76368	1.78025
6	1.31660	1.59429	1.72888	1.82212	1.90360	1.94425	1.97282	2.01074
7	1.39131	1.67850	1.82798	1.93813	2.04161	2.09730	2.13911	2.20059
8	1.45604	1.74908	1.90895	2.03165	2.15249	2.22083	2.27437	2.35863
9	1.51291	1.80978	1.97726	2.10956	2.24427	2.32315	2.38681	2.49195
10	1.56350	1.86296	2.03623	2.17607	2.32203	2.40972	2.48208	2.60593
11	1.60895	1.91022	2.08801	2.23391	2.38915	2.48428	2.56412	2.70460
12	1.65016	1.95270	2.13410	2.28495	2.44796	2.54942	2.63573	2.79095
13	1.68780	1.99123	2.17556	2.33054	2.50011	2.60702	2.69897	2.86730
14	1.72241	2.02645	2.21320	2.37165	2.54685	2.65848	2.75537	2.93538
15	1.75440	2.05885	2.24762	2.40904	2.58909	2.70486	2.80611	2.99656
16	1.78413	2.08884	2.27931	2.44327	2.62756	2.74696	2.85208	3.05192
17	1.81187	2.11672	2.30863	2.47481	2.66282	2.78545	2.89401	3.10233
18	1.83786	2.14276	2.33591	2.50402	2.69532	2.82082	2.93248	3.14846
19	1.86230	2.16717	2.36139	2.53119	2.72542	2.85350	2.96795	3.19090
20	1.88534	2.19014	2.38527	2.55658	2.75343	2.88382	3.00080	3.23012
21	1.90714	2.21181	2.40775	2.58039	2.77959	2.91208	3.03136	3.26649
22	1.92780	2.23231	2.42895	2.60278	2.80411	2.93850	3.05988	3.30036
23	1.94745	2.25176	2.44901	2.62392	2.82716	2.96330	3.08659	3.33200
24	1.96616	2.27026	2.46805	2.64391	2.84890	2.98663	3.11169	3.36164
25	1.98401	2.28788	2.48614	2.66287	2.86946	3.00864	3.13533	3.38950

The above table of Grubbs critical values (or cv) are for your reference and problem solving. Remember the count of total data points is n, as indicated along the left column for each row, with the confidence levels labeled along the top for each column.

Lets say we run some titrations, and the volume of base added for 5 different runs is, 6.18, 6.69, 6.49, 6.28, 4.85 mL, respectively. So the 4.85 mL seems very low and I think I might have misweighed my acid. So instead of just throwing away the value, I want to Grubbs it. First, put the points in order from lowest to highest value; 4.85, 6.18, 6.28, 6.49, 6.69. Then pump it through $G_{calculated}$ and if its greater than $G_{critical}$, then it may be rejected, as follows:

In[117]:=
```
data = {6.69, 6.18, 6.28, 4.85, 6.49}
gtest[data, Min, 95]
gtest[data, Min, 99]
```

Out[117]= {6.69, 6.18, 6.28, 4.85, 6.49}

{ N: 5, $\bar{x}$: 6.098, s: 0.724824, CL: 95%, $G_{calculated}$: 1.7218, $G_{critical}$: 1.67139, DataPoint: 4.85 } **may be Rejected and Excluded.**

{ N: 5, $\bar{x}$: 6.098, s: 0.724824, CL: 99%, $G_{calculated}$: 1.7218, $G_{critical}$: 1.74886, DataPoint: 4.85 } **must be Accepted and Retained for Inclusion.**

So I can reject the outlier and be 95% confident of its rejection, or I can be 99% confident in which case I must accept the outlier.

If I reject the outlier at 95% confidence, my mean and standard deviation would be

In[120]:=
```
data0 = data[[2 ;;]]
Print["95% Confidence: Mean = ", Mean[data0],
 " & Standard Deviation = ", StandardDeviation[data0]]
```

Out[120]= {6.18, 6.28, 4.85, 6.49}

95% Confidence: Mean = 5.95 & Standard Deviation = 0.744625

If I accept the outlier, then I would be 99% confident of the mean and standard deviation

In[122]:=
```
data
Print["99% Confidence: Mean = ", Mean[data],
 " & Standard Deviation = ", StandardDeviation[data]]
```

Out[122]= {6.69, 6.18, 6.28, 4.85, 6.49}

99% Confidence: Mean = 6.098 & Standard Deviation = 0.724824

And it is important to notice that with the acceptance of the outlier, the standard deviation is much larger then when we rejected it. Also the mean is shifted down to compensate for the acceptance of the low value.

Important Mental Connection: Later in this lecture, we get into the "Curve Fitting and Method of Least Squares" section. We take for granted the importance of the mean (average) and the standard deviation. When we calculate the mean and standard deviation we are deriving the parameters to fit the Gaussian to a histogram for our particular data points. So after you read the curve fitting section, you will realize the importance of fitting a line or curve to the data points, like fitting a line with form y=mx+b, to your data points. Calculating the mean, μ, and standard deviation, σ, are the fit parameters for a Gaussian curve for a 1D dataset, as presented above under the "Indeterminate/Random Errors and the Gaussian Distribution" section.

from Harris, Section 4-6:

In[124]:=
```
data = {10.2, 10.8, 11.6, 9.9, 9.4, 7.8, 10.0, 9.2, 11.3, 9.5, 10.6, 11.6};
gtest[data, Min, 50]
gtest[data, Min, 80]
gtest[data, Min, 95]
gtest[data, Min, 99]
```

{ N: 12, $\bar{x}$: 10.1583, s: 1.11393, CL: 50%, $G_{calculated}$: 2.11713, $G_{critical}$: 1.65016, DataPoint: 7.8 } **may be Rejected and Excluded.**

{ N: 12, $\bar{x}$: 10.1583, s: 1.11393, CL: 80%, $G_{calculated}$: 2.11713, $G_{critical}$: 1.9527, DataPoint: 7.8 } **may be Rejected and Excluded.**

{ N: 12, $\bar{x}$: 10.1583, s: 1.11393, CL: 95%, $G_{calculated}$: 2.11713, $G_{critical}$: 2.28495, DataPoint: 7.8 } **must be Accepted and Retained for Inclusion.**

{ N: 12, $\bar{x}$: 10.1583, s: 1.11393, CL: 99%, $G_{calculated}$: 2.11713, $G_{critical}$: 2.54942, DataPoint: 7.8 } **must be Accepted and Retained for Inclusion.**

therefore, the 7.8 data point must be accepted for inclusion in the dataset with 95% and 99% confidence.

from Harris, Problem 4-24:

In[129]:=
```
data = {192, 216, 202, 195., 204}
gtest[data, Max, 95]
gtest[data, Max, 99]
```

Out[129]= {192, 216, 202, 195., 204}

{ N: 5, $\bar{x}$: 201.8, s: 9.33809, CL: 95%, $G_{calculated}$: 1.52065, $G_{critical}$: 1.67139, DataPoint: 216 } **must be Accepted and Retained for Inclusion.**

{ N: 5, $\bar{x}$: 201.8, s: 9.33809, CL: 99%, $G_{calculated}$: 1.52065, $G_{critical}$: 1.74886, DataPoint: 216 } **must be Accepted and Retained for Inclusion.**

and again we must accept the outlier. In my experience, I've messed up some labs and subsequent measurements, and its extremely rare when you can reject an outlier.

from In-Class Lecture.. .

In[132]:=
```
data = {1.318, 1.336, 1.615, 1.935,
  2.287, 2.404, 2.857, 2.876, 3.346, 3.55, 3.845, 8.231}
gtest[data, Max, 50]
gtest[data, Max, 90]
gtest[data, Max, 95]
gtest[data, Max, 99]
```

Out[132]= {1.318, 1.336, 1.615, 1.935, 2.287,
 2.404, 2.857, 2.876, 3.346, 3.55, 3.845, 8.231}

{ N: 12, $\bar{x}$: 2.96667, s: 1.85952, CL: 50%, $G_{calculated}$: 2.83101, $G_{critical}$: 1.65016, DataPoint: 8.231 } **may be Rejected and Excluded.**

{ N: 12, $\bar{x}$: 2.96667, s: 1.85952, CL: 90%, $G_{calculated}$: 2.83101, $G_{critical}$: 2.1341, DataPoint: 8.231 } **may be Rejected and Excluded.**

{ N: 12, $\bar{x}$: 2.96667, s: 1.85952, CL: 95%, $G_{calculated}$: 2.83101, $G_{critical}$: 2.28495, DataPoint: 8.231 } **may be Rejected and Excluded.**

{ N: 12, $\bar{x}$: 2.96667, s: 1.85952, CL: 99%, $G_{calculated}$: 2.83101, $G_{critical}$: 2.54942, DataPoint: 8.231 } **may be Rejected and Excluded.**

since we can reject the 8.231 data point at any confidence level, the new calculations would be:

In[137]:=
```
data = data[[ ;; Length[data] - 1]]
Mean[data]
StandardDeviation[data]
```

Out[137]= {1.318, 1.336, 1.615, 1.935, 2.287, 2.404, 2.857, 2.876, 3.346, 3.55, 3.845}

Out[138]= 2.48809

Out[139]= 0.883378

Bias Testing of a Set of Measurements or Dataset

It is also possible to test an entire dataset against a true or accepted value to determine if the entire dataset, and its corresponding experiment, is valid. This is the Student's T-test and it is covered in the appendix of your Skoog textbook.

The Student's T-test is performed by computing the value of T and comparing it critical value T off of the below table. To compute the value of T, use the following equation:

$$T_{computed} = \frac{|\bar{x} - acceptedValue|}{\sigma / \sqrt{n}} \tag{15}$$

where $\bar{x}$ and σ are the computed mean and standard deviation for your dataset, as normal, n is the count or number of data points in your dataset, and the "acceptedValue" is that which your experiment is being compared to. Notice the subtraction in the numerator has the absolute value symbols around it, so the $T_{computed}$ is always positive.

If $T_{computed}$ is less than the critical value from the below table, then the results from the experiment are acceptable and thus not significantly different than the accepted. On the other hand, If $T_{computed}$ is greater than the critical value, then the results from the experiment are rejected and therefore significantly different than the accepted value.

This example, a1-10, was recast from page 985 of your text and I wrote a little *Mathematica* program again to automate the process for myself:

In[77]:=
```
tvalue[n_, CL_] := Module[{ddof},
  ddof = n - 1;
  If[(ddof > 0) && (CL > 0) && (CL < 100),
   InverseSurvivalFunction[StudentTDistribution[ddof], (1 - CL / 100.) / 2],
   Print["n or CL is out of range"]]]
tci[n_, t_] := Module[{ddof}, (
   ddof = n - 1;
   100 * (1 - 2 * SurvivalFunction[StudentTDistribution[ddof], t])
  )]
(*little tests*)
tci[3, 4.3027]
tci[6, 1.4759]
ttest[data_?VectorQ, trueMean_, CL : 80 | 90 | 95 | 99 | 99.9] :=
 Module[{mean, sd, t, cv, bb},
  If[Length[data] < 1,
   Print["dataset must have more than 1 points"],
   (mean = Mean[data];
    sd = StandardDeviation[data];
    cv = Tvalue[Length[data], CL];
    t = (mean - trueMean) / (sd / Sqrt[Length[data]]);
    bb = If[t < 0, (cv = -cv;
       If[t < cv, False, True]), If[t > cv, False, True]];
    txt = "{ N: " <> ToString[Length[data]] <> ", x̄: " <>
      ToString[mean] <> ", s: " <> ToString[sd] <> ", CL: " <> ToString[CL] <>
      "%, t: " <> ToString[t] <> ", cv: " <> ToString[cv] <> " }";
    If[bb,
     Print[txt, Style[" Accept Dataset; Bias NOT Detected.", Bold]],
     Print[txt , Style[" Reject Dataset; Bias Detected.", Bold]]
    ]
   )
  ]
 ]
Table[NumberForm[tvalue[n, CL], {10, 5}],
  {n, 2, 25}, {CL, {50, 80, 90, 95, 98, 99, 99.5, 99.9}}];
Join[{{"n", "50%CL", "80%CL", "90%CL", "95%CL", "98%CL", "99%CL", "99.5%CL",
    "99.9%CL"}}, Table[Join[{n}, Table[NumberForm[tvalue[n, CL], {10, 5}],
     {CL, {50, 80, 90, 95, 98, 99, 99.5, 99.9}}]], {n, 2, 25}]];
Print[Style["Student's t Values", Bold, FontSize → 24]]
Grid[%%, Alignment → Right, Frame → All,
 Background → {{LightGray, None}, {LightRed, None}}, ItemStyle → {1 → Bold}]
```

Out[79]= 95.0001

Out[80]= 80.0004

Student's t Values

Out[85]=

n	50%CL	80%CL	90%CL	95%CL	98%CL	99%CL	99.5%CL	99.9%CL
2	1.00000	3.07768	6.31375	12.70620	31.82052	63.65674	127.321 ⋱ 34	636.619 ⋱ 25
3	0.81650	1.88562	2.91999	4.30265	6.96456	9.92484	14.08905	31.59905
4	0.76489	1.63774	2.35336	3.18245	4.54070	5.84091	7.45332	12.92398
5	0.74070	1.53321	2.13185	2.77645	3.74695	4.60409	5.59757	8.61030
6	0.72669	1.47588	2.01505	2.57058	3.36493	4.03214	4.77334	6.86883
7	0.71756	1.43976	1.94318	2.44691	3.14267	3.70743	4.31683	5.95882
8	0.71114	1.41492	1.89458	2.36462	2.99795	3.49948	4.02934	5.40788
9	0.70639	1.39682	1.85955	2.30600	2.89646	3.35539	3.83252	5.04131
10	0.70272	1.38303	1.83311	2.26216	2.82144	3.24984	3.68966	4.78091
11	0.69981	1.37218	1.81246	2.22814	2.76377	3.16927	3.58141	4.58689
12	0.69745	1.36343	1.79588	2.20099	2.71808	3.10581	3.49661	4.43698
13	0.69548	1.35622	1.78229	2.17881	2.68100	3.05454	3.42844	4.31779
14	0.69383	1.35017	1.77093	2.16037	2.65031	3.01228	3.37247	4.22083
15	0.69242	1.34503	1.76131	2.14479	2.62449	2.97684	3.32570	4.14045
16	0.69120	1.34061	1.75305	2.13145	2.60248	2.94671	3.28604	4.07277
17	0.69013	1.33676	1.74588	2.11991	2.58349	2.92078	3.25199	4.01500
18	0.68920	1.33338	1.73961	2.10982	2.56693	2.89823	3.22245	3.96513
19	0.68836	1.33039	1.73406	2.10092	2.55238	2.87844	3.19657	3.92165
20	0.68762	1.32773	1.72913	2.09302	2.53948	2.86093	3.17372	3.88341
21	0.68695	1.32534	1.72472	2.08596	2.52798	2.84534	3.15340	3.84952
22	0.68635	1.32319	1.72074	2.07961	2.51765	2.83136	3.13521	3.81928
23	0.68581	1.32124	1.71714	2.07387	2.50832	2.81876	3.11882	3.79213
24	0.68531	1.31946	1.71387	2.06866	2.49987	2.80734	3.10400	3.76763
25	0.68485	1.31784	1.71088	2.06390	2.49216	2.79694	3.09051	3.74540

The above table of Student's t-values (critical values or cv) are for your reference and problem solving. The number of data points, n, is indicated along the left column for each row, the degrees of freedom (dof) would be n-1, with the confidence levels labeled along the top for each column.

The following problem has four measurements and the dataset is being tested against the accepted value of 0.123.

In[86]:=
```
data = {0.112, 0.118, 0.115, 0.119}
ttest[data, 0.123, 95]
ttest[data, 0.123, 99]
```

Out[86]=
```
{0.112, 0.118, 0.115, 0.119}

{ N: 4, x̄: 0.116, s: 0.00316228, CL: 95%, t: -4.42719, cv: -3.18245 }
  Reject Dataset; Bias Detected.

{ N: 4, x̄: 0.116, s: 0.00316228, CL: 99%, t: -4.42719, cv: -5.84091 }
  Accept Dataset; Bias NOT Detected.
```

In other words, we can reject the dataset and be 95% confident we are doing the right thing. However, we must accept the dataset, and its corresponding experiment, if we want to be 99% confident.

You may question the value of t as compared to the solution in the text, where the solution in the text is -4.375, and that of above is -4.42719. But it is just a matter of how the standard deviation was rounded as shown:

```
(.116 - .123) / (.0032 / Sqrt[4])
(.116 - .123) / (.00316228 / Sqrt[4])
```

```
-4.375
```

```
-4.42719
```

from Harris, Section 4-4, Case 1:

In[89]:=
```
data = {3.29, 3.22, 3.30, 3.23}
ttest[data, 3.19, 90]
ttest[data, 3.19, 95]
ttest[data, 3.19, 99]
```

Out[89]=
```
{3.29, 3.22, 3.3, 3.23}

{ N: 4, x̄: 3.26, s: 0.0408248, CL: 90%, t: 3.42929, cv: 2.35336 }
  Reject Dataset; Bias Detected.

{ N: 4, x̄: 3.26, s: 0.0408248, CL: 95%, t: 3.42929, cv: 3.18245 }
  Reject Dataset; Bias Detected.

{ N: 4, x̄: 3.26, s: 0.0408248, CL: 99%, t: 3.42929, cv: 5.84091 }
  Accept Dataset; Bias NOT Detected.
```

therefore the dataset must be rejected unless you want to be 99% confident.

from In-Class Lecture.. .

In[93]:=
```
data = {1.318, 1.336, 1.615, 1.935,
  2.287, 2.404, 2.857, 2.876, 3.346, 3.55, 3.845, 8.231}
ttest[data, 4.1, 90]
ttest[data, 4.1, 95]
ttest[data, 4.1, 99.9]
data = {1.318, 1.336, 1.615, 1.935, 2.287, 2.404, 2.857, 2.876, 3.346, 3.55, 3.845}
ttest[data, 4.1, 90]
ttest[data, 4.1, 95]
ttest[data, 4.1, 99.9]
```

Out[93]=
```
{1.318, 1.336, 1.615, 1.935, 2.287,
 2.404, 2.857, 2.876, 3.346, 3.55, 3.845, 8.231}
```

```
{ N: 12, x̄: 2.96667, s: 1.85952, CL: 90%, t: -2.11128, cv: -1.79588 }
  Reject Dataset; Bias Detected.
{ N: 12, x̄: 2.96667, s: 1.85952, CL: 95%, t: -2.11128, cv: -2.20099 }
  Accept Dataset; Bias NOT Detected.
{ N: 12, x̄: 2.96667, s: 1.85952, CL: 99.9%, t: -2.11128, cv: -4.43698 }
  Accept Dataset; Bias NOT Detected.
```

Out[97]=
```
{1.318, 1.336, 1.615, 1.935, 2.287, 2.404, 2.857, 2.876, 3.346, 3.55, 3.845}
```

```
{ N: 11, x̄: 2.48809, s: 0.883378, CL: 90%, t: -6.05188, cv: -1.81246 }
  Reject Dataset; Bias Detected.
{ N: 11, x̄: 2.48809, s: 0.883378, CL: 95%, t: -6.05188, cv: -2.22814 }
  Reject Dataset; Bias Detected.
{ N: 11, x̄: 2.48809, s: 0.883378, CL: 99.9%, t: -6.05188, cv: -4.58689 }
  Reject Dataset; Bias Detected.
```

take home message: leaving in the outlier of 8.231 makes us accept the dataset with 95% confidence but reject it at 90%.

Calibration Curves, Curve Fitting, and Method of Least Squares

All data is collected as a series of discrete points, hence the phrase “data points”. The implication here is that data is never collected as a completely smooth and continuous set of data. Whenever we are collecting our data from an instrument, it is vital to see if there is a relationship or trend between the data points. The best and really only effective way to mentally see or visualize such a relationship is by plotting the data on a graph. The first step to analyzing data is to visualize the data by graphing, in real-time as you run an experiment. It is vital to make such mental associations to realize the relationships or trends in the data you collect. Such mental visualizations then lead to mathematical expression fitting. Much of the software you will use for the instruments will plot the data for you and the relationships will reveal themselves well. Some instruments will not, therefore, I recommend as a standard practice that you manually plot the data in Excel, or other such similar software, as you run your experiments. This is super critical to ensure your instrument is following some form or structure and not just pumping out a bunch of random scattered points.

Once the relationship and trend is visually realized by graphing the data, or through a proper physical/-chemical molecular interaction relation, the next natural step is to mathematically connect those points through some mathematical function or expression. This method will provide a smooth and continuous expression for your data points. A line is the most simple relationship for 2D data, hence the term linear regression. However, all mathematical expressions, i.e., polynomials of any order, natural logs or logs base 10, exponentials, and even the Gaussian, can be used to fit any data through the more general technique called least squares fitting. The primary reason for performing such mathematical expression fitting to your discrete data points is to provide a smooth and continuous mathematical expression to your data. Such a continuous expression will, first, effectively fill in the gaps between your data points, and second, lead to physically relevant contributions to your data. Meaning, your data points will be based on a mathematical relationship that has physical associations or relevance to the atomic or molecular interactions with the instrument.

Curve fitting is an extremely important and powerful technique. The collection of measurements, leading to a set of data, from a real system usually comes in the form of discrete points. The points are inherently disconnected, disjointed. Curve fitting is the mathematical and statistical technique that binds those separate points of data together to form a relationship in the form of a mathematical equation. Once a mathematical relationship is accurately fit to the data points, then a model can be applied to understand the magnitudes of the data points and their respective interconnectiveness. Therefore, curve fitting allows one to associate the physical measurements with a mathematical and physically relevant model, leading to great understanding.

Curve fitting involves a lot of 2D or 3D fitting of functions using a lot of statistics. Luckily software applications exist to do the hard work for us. The statistical technique is called least-squares because the culminated distances of all data points to a proposed line or function must be minimized. That makes sense. The smallest distances between all points and the line or function should make the proposed line or function the “best fit”. There the “least” is that minimization of distances, and “squares” is how the technique calculates the distances between each point and the proposed line or

function.

The line represents the most simple mathematical function. It is given by the equation y = mx + b, where x is the independent variable and the horizontal axis on a 2D plot, y is the dependent variable and the vertical axis on a 2D plot, m is the slope of the line, and b is the y-intercept. Typically, chemists deal with analytes, which is the target compound providing the measured response, and their respective concentrations, along the x-axis or the independent variable, versus their measured instrument responses, along the y-axis or the dependent variable. The dependent variable depends on the independent variable, or the measured responses depends on the concentration of the analyte, or measured response versus concentration, or y versus x. Therefore, the equation of the line takes on the form of [measured response]= mC +b, where C is the concentration of the analyte. The least-square technique takes each of the x and y data points, pumps it through a series of statistical techniques and returns the m and b parameters. The line can then be plotted on a graph.

Upon collecting your set of (x,y) measurements, it is always best to first plot the data. The eye and the mind can imagine the relationship between the points quite well. It is always the best step to help decide what type of function will best fit the points.

Any instrument will fluctuate its measured responses depending on a long list of internal and external conditions, situations, and circumstances. Such conditions include atmospheric and laboratory temperature, pressure, and humidity, electrical power deviations, magnetic field fluctuations, radio and microwave noise, room or infrared light sources, ambient electrical sources, such as human presence, or cell phones, etc, and instrument leveling or movement. Variations in sample collection, preparation, and treatment will also affect an instrument's responses. Calibration curves are created on a daily or routine basis for a particular instrument to null out most of these effects of an instrument's responses. A calibration curve is created by generating a set of instrument measurements against a set of known concentrations or standard samples or solutions for a particular analyte. The measured response is then plotted and fitted against the known concentrations causing any fluctuating conditions to be nullified. The generated calibration curve provides a standard equation for that daily or span of time, standardizing the instrument, nullifying fluctuations, while simultaneously providing a measured response to concentration equation for the analyte in question to solve unknown concentrations accurately and precisely.

I am going to do a few examples and it would serve you extremely well to reproduce these examples in Excel. I suggest learning how to do these techniques in Excel because Excel is available all over the college for you, in every classroom and learning center we have in the college. Excel is also prominent in the work place and graduate schools. You can purchase Excel within Microsoft Office at a nominal student expense.

Demonstration of Least Squares Fitting

The main idea of least squares fitting is to propose a function, i.e., a line, a polynomial, log, etc., and calculate the sum of the squares of the differences between the actual data points and the proposed function. Let that sink in. Or, mathematically:

$$SS_{residual} = \sum_{i=1}^{N} (y_i - f(x_i))^2 \tag{16}$$

where $f(x_i)$ is the yield of the proposed function, y_i is of the actual data point, and the reason for squaring the difference is to make the subtraction always a positive value. Summing all those positive values would give a single positive value. A SSresidual of zero, therefore, would be a perfect fit of the data to the proposed function. The lower the summed squared differences, the better the fit. So the idea is to lower the residual as much as possible.

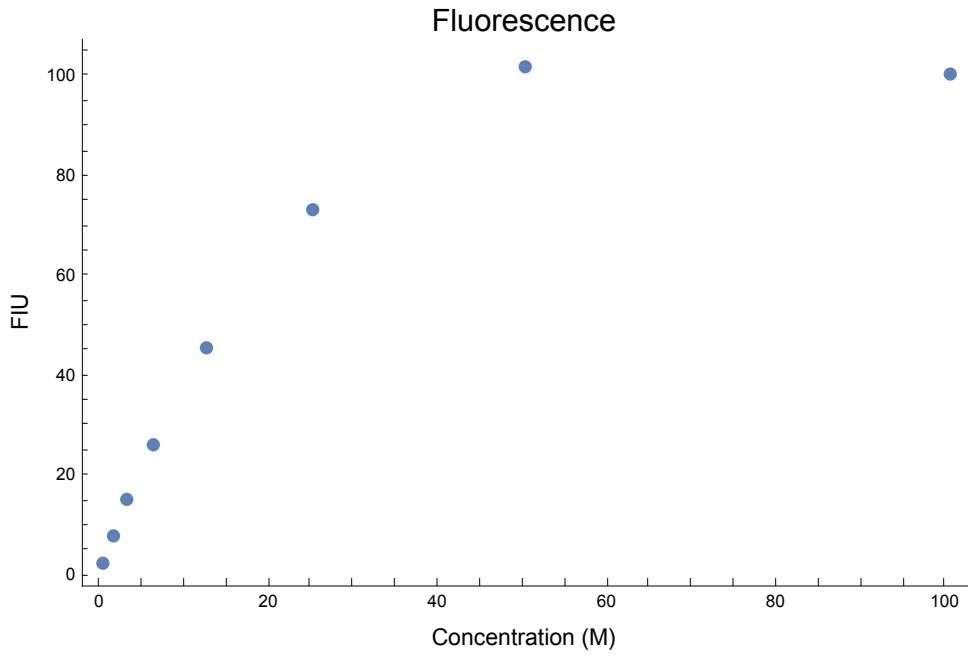
Most common expressions are easy and can be analytically solved, those of linear expressions. Nonlinear expressions are a bit more complex and usually require many proposals and tests to find and hunt for the lowest residual and best fit.

For instance fluorescence versus concentration of a fluorophore follows the nonlinear expression $FIU = a\left(1 - 10^{-b\,C}\right)$, where FIU is the fluorescence intensity units, C the concentration, and a and b are fitting parameters. Again, a and b can be traced back to physically relevant constants of how and why the molecule absorbs light of specific frequencies, holds on to that energy for a certain extended amount of time, and then emits or fluoresces the light at a lower energy. But for this demonstration, lets just consider a and b as fitting parameters.

We'll start with real fluorescent data and immediately plot it as shown:

```
data = {{100.5, 100.5}, {50.25, 101.96}, {25.125, 73.31}, {12.5625, 45.64},
   {6.28125, 26.22}, {3.14063, 15.29}, {1.57031, 7.96}, {0.31406, 2.49}}
p1 = ListPlot[data, PlotStyle → {PointSize[0.015]},
   ImageSize → 500, PlotLabel → Style["Fluorescence", 16],
   FrameLabel → {Style["Concentration (M)", 12], Style["FIU", 12]},
   Frame → {{True, False}, {True, False}}]
```

```
{{100.5, 100.5}, {50.25, 101.96}, {25.125, 73.31}, {12.5625, 45.64},
 {6.28125, 26.22}, {3.14063, 15.29}, {1.57031, 7.96}, {0.31406, 2.49}}
```



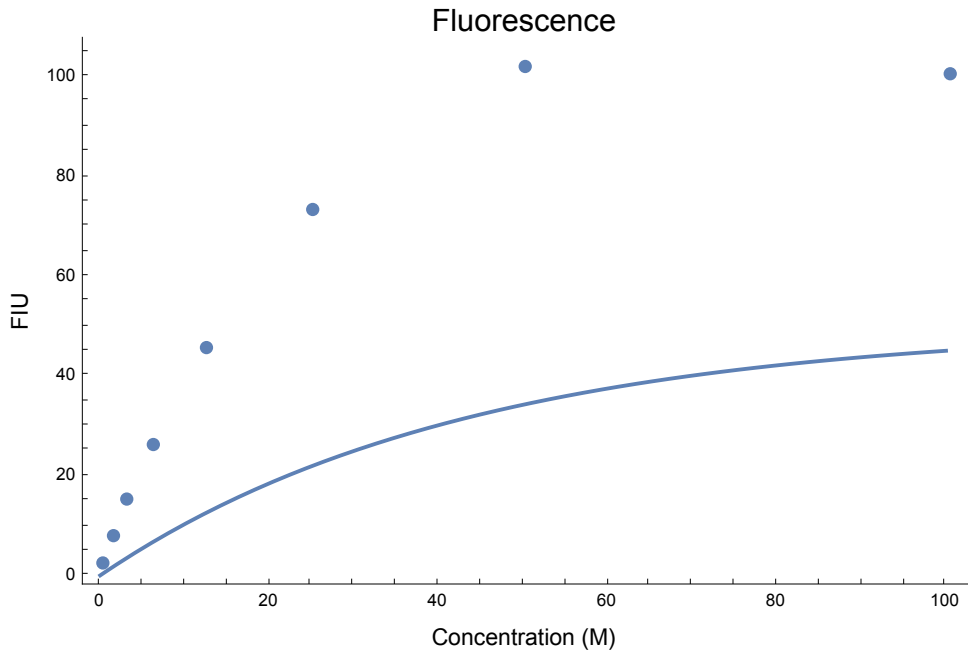
Fluorescence reaches an asymptote is because of quenching. Proposing a fluorescence expression (q1) with a=50 and b=0.01 (f1), and I create a function (SS) that calculates the sum of the squares of the differences, the SSresidual, uses that SS function to generate the sum of the squares, and then plots the data points against the proposed function:

```
q1 = a (1 - 10^(-b C))
f1 = q1 /. {a → 50, b → 0.01}
SS[data_, exp_] := Module[{summed = 0., p, pC, pF, ss},
    (For[i = 1, i ≤ Length[data], i++, (
       p = data[[i]]; pC = p[[1]]; pF = p[[2]];
       ss = (pF - (exp /. (C → pC)))^2;
       summed += ss;
      )];
     summed)];
SS[data, f1]
p2 = Plot[f1, {C, 0, 100}, PlotStyle → Thick];
Show[p1, p2]
```

$$\left(1 - 10^{-b\,C}\right)\,a$$

$$50 \times \left(1 - 10^{-0.01\,C}\right)$$

```
11947.2
```



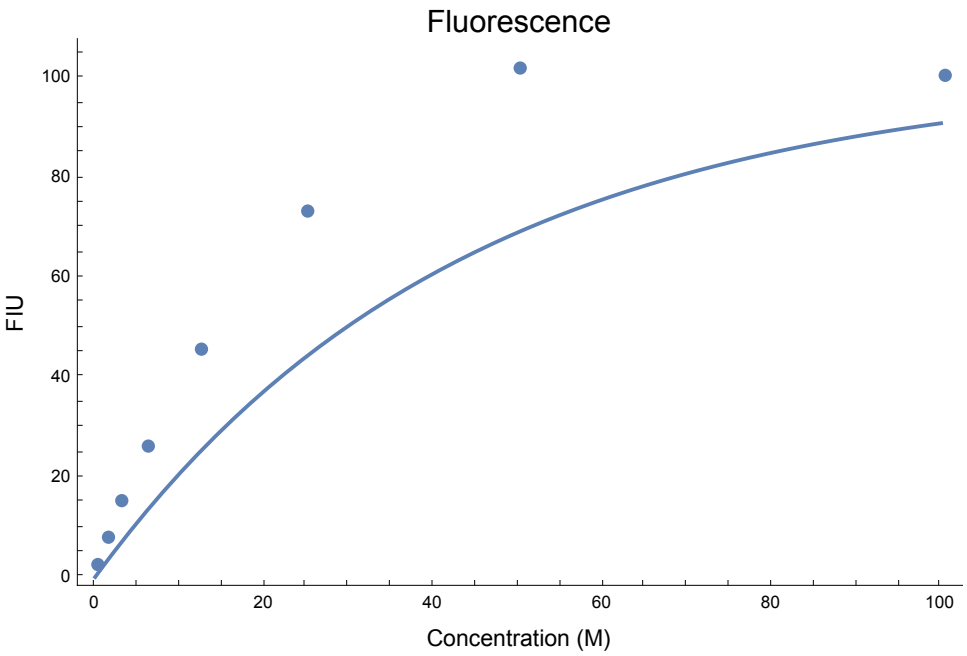
where line 1 is just the function, line 2 with a and b substituted, line 3 the SSresidual = 11,947.2, and finally the plot.

Obviously it is a terrible plot, but it shows us that “a” controls the asymptote and “b” the amount of curvature. Lets do the same thing, except with a=101 and b=0.01:

```
f1 = q1 /. {a → 101, b → 0.01}
SS[data, f1]
p2 = Plot[f1, {C, 0, 100}, PlotStyle → Thick];
Show[p1, p2]
```

$$101 \times \left(1 - 10^{-0.01\,C}\right)$$

```
2658.31
```

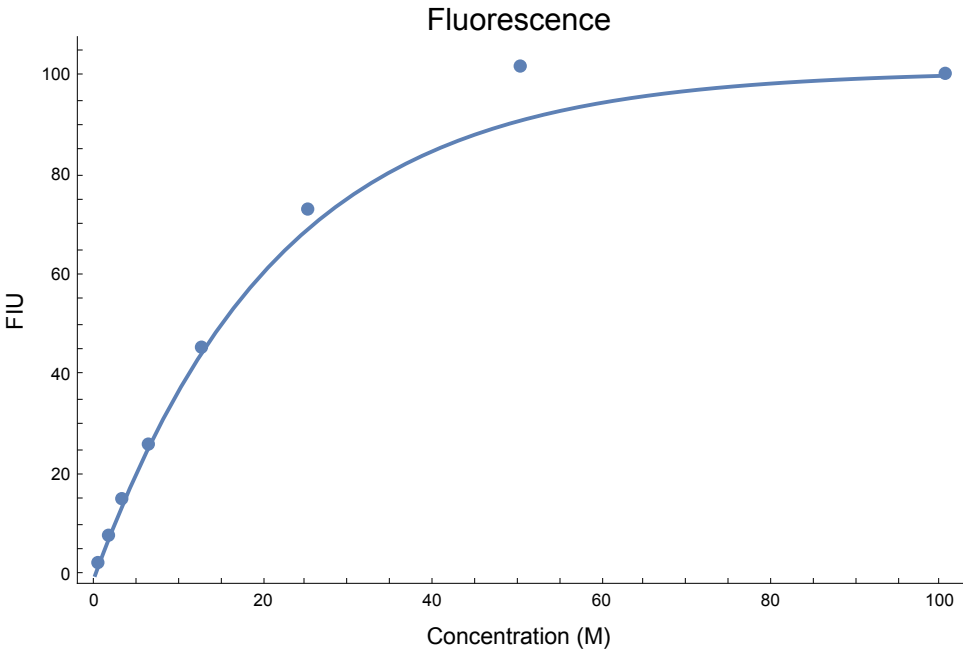


wow, that really reduced the SSresidual from 11,947.2 down to 2,658.31. Lets give it more curvature, say b=0.02

```
f1 = q1 /. {a → 101, b → 0.02}
SS[data, f1]
p2 = Plot[f1, {C, 0, 100}, PlotStyle → Thick];
Show[p1, p2]
```

$$101 \times \left(1 - 10^{-0.02\,C}\right)$$

```
143.662
```

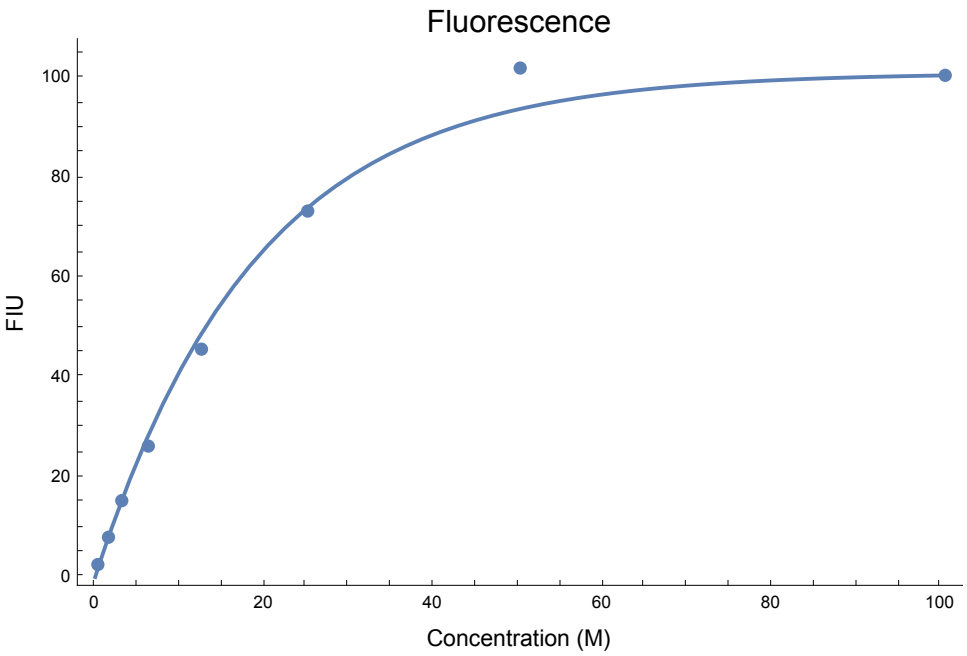


now the SSresidual is down from 2,658.31 to 143.662. And playing with only b a bit, I settled on b=0.0228:

```
f1 = q1 /. {a → 101, b → 0.0228}
SS[data, f1]
p2 = Plot[f1, {C, 0, 100}, PlotStyle → Thick];
Show[p1, p2]
```

$$101 \times \left(1 - 10^{-0.0228\,C}\right)$$

```
82.5527
```



which brought my rough approximations or proposed functions into a visually good fit. Jsust playing with a and b to minimize SSresidual, gave not only a fairly small SSresidual of 82.5527, but also a fit that is visually very good.

Mathematica can fit nonlinear functions very well. So, using its computation power, I can get even better:

```
nlf1 = NonlinearModelFit[data, q1, {a, b}, C];
nlf1["Properties"];
nlf1["FitResiduals"]
Total[%^2]
nlf1["RSquared"]
nlf1["BestFitParameters"]
f2 = Normal[nlf1]
SS[data, f2]
p2 = Plot[f1, {C, 0, 100}, PlotStyle → Thick];
Show[p1, p2]
```

```
{-3.8705, 6.05769, -0.622902, -2.20936, -1.3062, 0.485026, 0.276953, 0.906434}
```

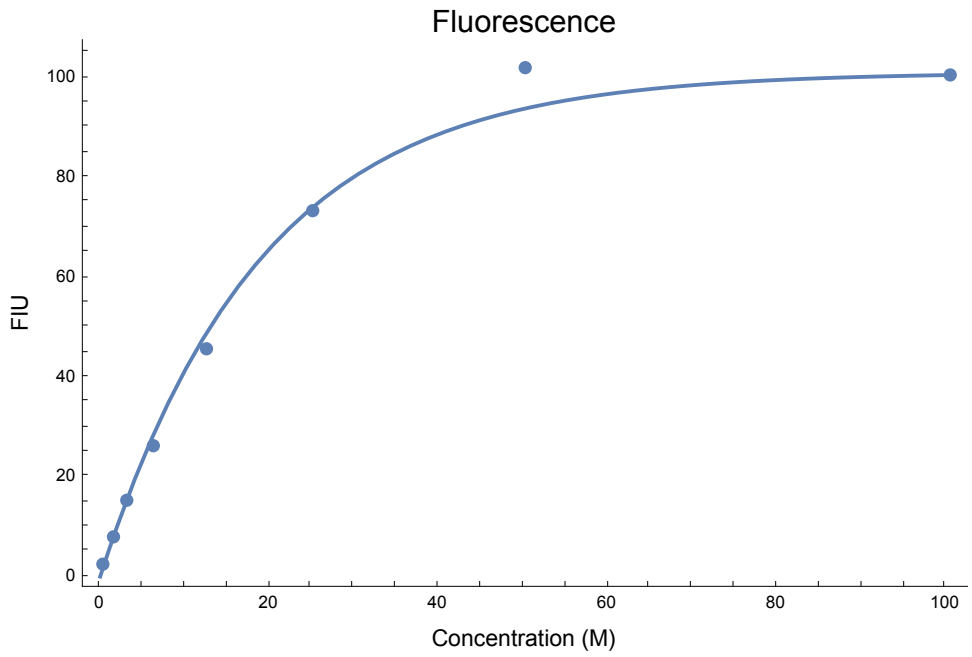
```
59.7854
```

```
0.997934
```

```
{a → 105.191, b → 0.0209759}
```

$$105.191 \times \left(1 - 10^{-0.0209759\, C}\right)$$

```
59.7854
```



where you'll see the very lowest SSresidual using *Mathematica* gives a value of 59.7854 with final fitting parameters of a=105.191 and b=0.0209759. So, it can be assumed that is the absolute minimum SSresidual for this data and getting a value of 82.5527 by just eyeballing the fit and SSresidual is quite good.

Accompanying this text is an Excel workbook, named "fitting__various_techniques2.xlsx", which contains three tab sheets at the bottom once you open it in Excel. The first tab is similar to the above fluorescence data and fit, where you'll adjust the "a" and "b" and attempt to minimize the "sum" cell at the bottom and visually fit the plot. Tab sheets 2 and 3 are covered below for the conductivity fit.

It is important to *not* use this Excel workbook as a template, but when you have your own data, to start your own blank worksheets and learn from these examples to recreate your own worksheets from

scratch. That is how you'll learn how to process your data and learn Excel at the same time. Its how you'll learn to adapt to other forms of data, proposed functions, circumstances, and situations. This is how you'll learn how to teach yourself and verify that the results you achieve are valid. You'll be able to pick up pointers and guidance from tutorials and "how to"s from the internet, but you'll ultimately learn to problem solve and fill in the gaps by your own methods and creativity. I can not stress this enough, be sure to verify your results and not just take the results as *word* or *gospel* all because the, "computer said so". With all computers, garbage in will yield garbage out.

In my own personal writing of this document, I verified the results of *Mathematica* with that of the textbook, other papers, Python, and Excel to verify the results of *Mathematica*. And *Mathematica* is one of the most powerful programs on the planet.

Here is the nonlinear fit from *Mathematica* for the data in the Excel worksheet tab 1:

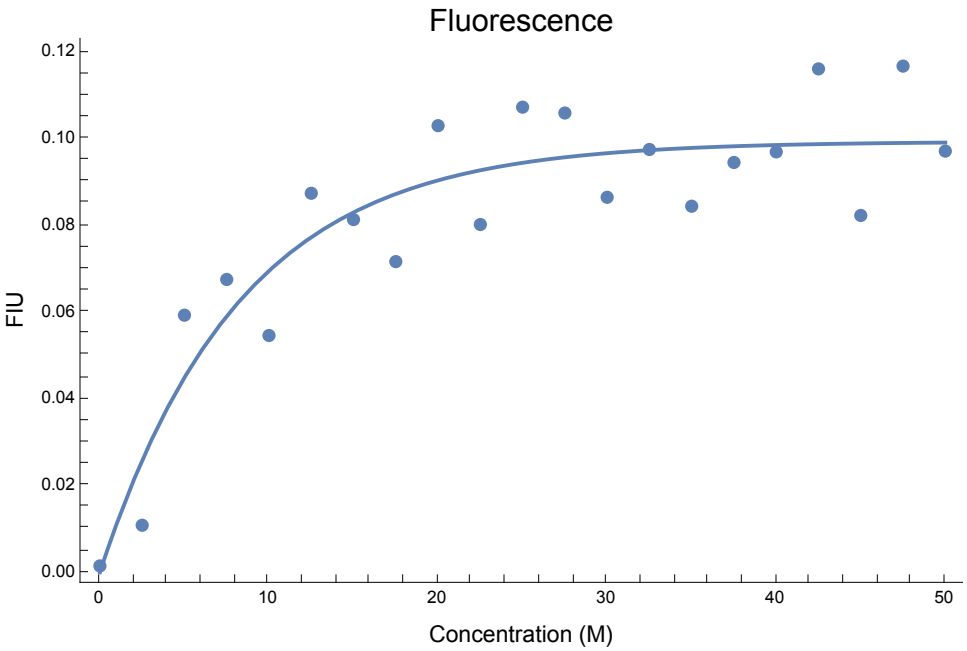
```
data = {{0, 0.001460182}, {2.5, 0.010837851}, {5, 0.05927319},
    {7.5, 0.06751057}, {10, 0.054586926}, {12.5, 0.08738231}, {15, 0.081319692},
    {17.5, 0.071623572}, {20, 0.102968185}, {22.5, 0.080178715},
    {25, 0.107236078}, {27.5, 0.105886099}, {30, 0.086454802}, {32.5, 0.097478985},
    {35, 0.084398055}, {37.5, 0.094482613}, {40, 0.096926302}, {42.5, 0.116064327},
    {45, 0.082266461}, {47.5, 0.116702879}, {50, 0.097106221}};
p1 = ListPlot[data, PlotStyle → {PointSize[0.015]},
    ImageSize → 500, PlotLabel → Style["Fluorescence", 16],
    FrameLabel → {Style["Concentration (M)", 12], Style["FIU", 12]},
    Frame → {{True, False}, {True, False}}];
q1 = a (1 - 10^(-b C));
nlf1 = NonlinearModelFit[data, q1, {a, b}, C];
nlf1["Properties"];
nlf1["FitResiduals"];
Total[%^2];
nlf1["RSquared"]
nlf1["BestFitParameters"]
f1 = Normal[nlf1]
SS[data, f2]
p2 = Plot[f1, {C, 0, 50}, PlotStyle → Thick];
Show[p1, p2]
```

```
0.9817
```

```
{a → 0.0993105, b → 0.0522651}
```

$$0.0993105 \times \left(1 - 10^{-0.0522651\, C}\right)$$

```
0.00285479
```



where the ultimate best fit is with a=0.003105 and b=0.0522651.

Please refer to the “fitting__various_techniques3.xls” file for updates and the actual source code of the spreadsheets.

Curve Fitting Examples

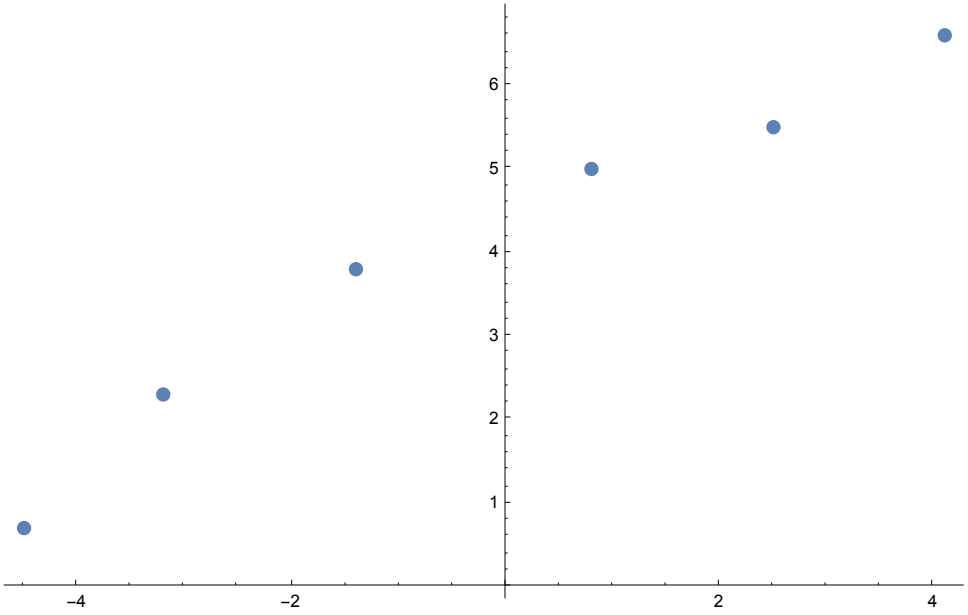
In the following examples, different forms of data are presented and different real world methods of fitting are applied to that data. The take home message, the big picture, is to first plot the data and ensure that the data is following a form, and not just scattered about. You may need to plot the data on semi-log or log-log scales to ensure the data is spread out properly, which usually occurs with the data is stretched over many decades (x10). Then research which form of proposed function is best for that type of data and its associated experiment or instrument. Then apply your best methods to get the best fit you can to the data.

Example 1

Here is some (x,y) data you may acquire in the lab:

```
data = {{-4.5, 0.7}, {-3.2, 2.3}, {-1.4, 3.8}, {0.8, 5.0}, {2.5, 5.5}, {4.1, 6.6}}
ListPlot[data, PlotStyle → {PointSize[0.015]}, ImageSize → 500]
```

```
{{-4.5, 0.7}, {-3.2, 2.3}, {-1.4, 3.8}, {0.8, 5.}, {2.5, 5.5}, {4.1, 6.6}}
```



It is best to always plot your data first so that the eye and brain can determine if the data has a form. meaning, if the data is without form, it will appear completely scattered when plotted. If it has a form, then that should be clear when plotted and a function should be fit to the data.

The plotted data looks pretty linear to me, but there is some chance of a slight curve, except for that last upper right point. In *Mathematica*, the Fit function does a linear or polynomial least-squares fit wherein the second parameter sets what type of polynomial to fit to. This first fit is to a simple line:

```
f1[x_] = Fit[data, {1, x}, x]
lmf = LinearModelFit[data, {1, x}, x][{"BestFit", "RSquared"}];
Print["y = f[x] = ", fxn1 = Normal[lmf[[1]]],
 "; correlation coefficient R^2 = ", lmf[[2]], "."]
```

```
4.1653 + 0.642226 x
```

```
y = f[x] = 4.1653 + 0.642226 x; correlation coefficient R^2 = 0.963586.
```

The second fit is to a quadratic equation:

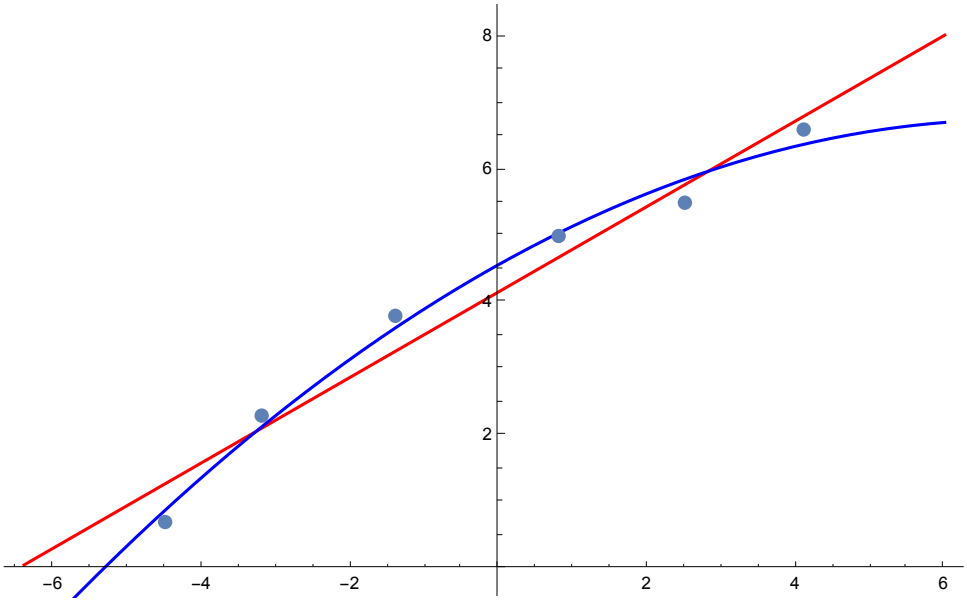
```
f2[x_] = Fit[data, {1, x, x^2}, x]
lmf = LinearModelFit[data, {1, x, x^2}, x][{"BestFit", "RSquared"}];
Print["y = f[x] = ", fxn2 = Normal[lmf[[1]]],
 "; correlation coefficient R^2 = ", lmf[[2]], "."]
```

```
4.57521 + 0.621742 x - 0.0444224 x^2
```

```
y = f[x] = 4.57521 + 0.621742 x -
 0.0444224 x^2; correlation coefficient R^2 = 0.988721.
```

The extracted R^2 value for each curve fit is called the correlation coefficient and the closer it is to unity, the closer the fit to the data points. Therefore, the quadratic fit, with R^2=0.99, is better fit then the linear fit, with R^2=0.96, as is also clear when we plot both fits against the raw data points:

```
fit1 = Plot[fxn1, {x, -6.4, 6}, PlotStyle → Red];
fit2 = Plot[fxn2, {x, -6.4, 6}, PlotStyle → Blue];
points = ListPlot[data, PlotStyle → {PointSize[0.015]}];
Show[fit1, fit2, points, ImageSize → 500]
```



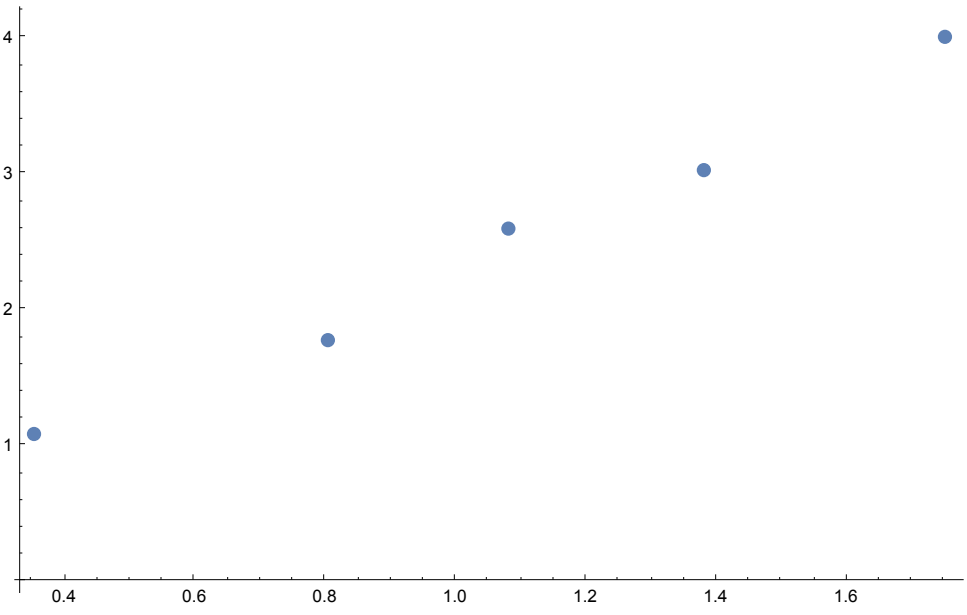
And again, that upper right data point is sort of out-of-curve. I would probably run another test that is a bit farther out to see where that point lies on the plot. That would also better determine if a line or a quadratic will best fit these points.

Example 2: from a1-11, Page 987

Now to solve an example from the text, page 987:

```
data = {{.352, 1.09}, {.803, 1.78}, {1.08, 2.60}, {1.38, 3.03}, {1.75, 4.01}}
ListPlot[data, PlotStyle → {PointSize[0.015]}, ImageSize → 500]
```

```
{{0.352, 1.09}, {0.803, 1.78}, {1.08, 2.6}, {1.38, 3.03}, {1.75, 4.01}}
```



```
f1[x_] = Fit[data, {1, x}, x]
lmf = LinearModelFit[data, {1, x}, x][{"BestFit", "RSquared"}];
Print["y = f[x] = ", fxn1 = Normal[lmf[[1]]],
 "; correlation coefficient R^2 = ", lmf[[2]], "."]
```

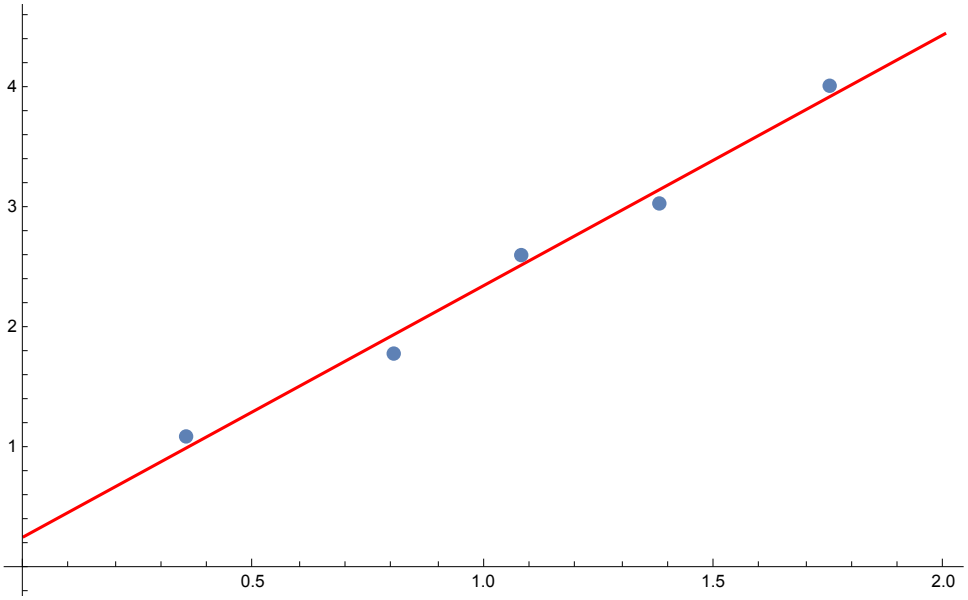
$$0.256741 + 2.09251\, x$$

$$y = f[x] = 0.256741 + 2.09251\, x;\ \text{correlation coefficient } R^2 = 0.987712.$$

```
LinearModelFit[data, {1, x, x^2}, x][{"BestFit", "RSquared"}]
```

$$\{0.521997 + 1.45327\, x + 0.304724\, x^2,\ 0.991909\}$$

```
fit1 = Plot[fxn1, {x, 0, 2}, PlotStyle → Red];
points = ListPlot[data, PlotStyle → {PointSize[0.015]}];
Show[fit1, points, ImageSize → 500]
```



Where the linear plot matches the results for the example on page 988 in the text. Even though the quadratic is a better fit, the correlation for the linear fit is certainly an excellent fit, and since linear is a more simple equation, I would go with that one in this case.

Example 3

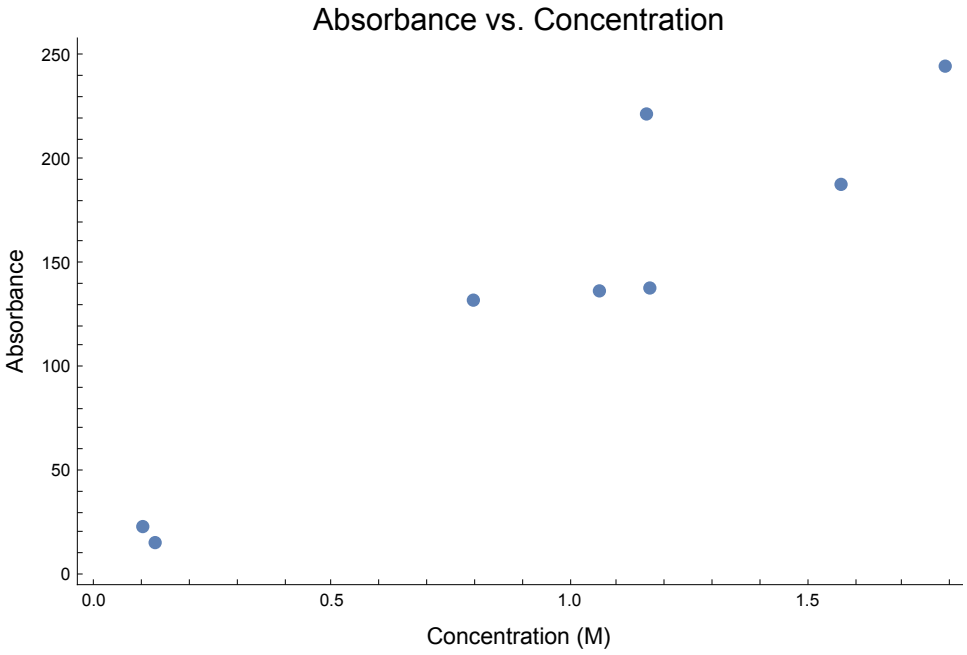
Question: Using a least-square line (aka linear regression) fitting, find the slope, y-intercept, and correlation coefficient (R^2) for the calibration of Absorbance (A) vs. Concentration (C in M) of Vitamin C at 246 nm UV light as outlined for Beer's Law: {C, A} = {{1.568, 188.33}, {1.166, 138.38}, {1.787, 245.32}, {0.100, 23.38}, {0.126, 15.65}, {0.795, 132.52}, {1.060, 136.99}, {1.159, 222.27}}. Formally plot the points with the superimposed fitted line with the proper title, axis labels and units, etc. What is the unknown molar concentration of Vitamin C if the absorbance is measured to be 90.850? Did you interpolate or extrapolate to arrive at your final concentration?

Solution:

Beer's Law is a linear function and its form and function I leave to you which is discussed in detail in your Skoog and Sawyer textbooks. But even if I didn't know that, I can take the information directly from the question and blinding plot it to see its form.

```
data = {{1.568, 188.33}, {1.166, 138.38}, {1.787, 245.32}, {0.100, 23.38},
  {0.126, 15.65}, {0.795, 132.52}, {1.060, 136.99}, {1.159, 222.27}}
ppts = ListPlot[data, PlotStyle → {PointSize[0.015]},
  PlotLabel → Style["Absorbance vs. Concentration", 16],
  FrameLabel → {Style["Concentration (M)", 12], Style["Absorbance", 12]},
  Frame → {{True, False}, {True, False}}, ImageSize → 500]
```

```
{{1.568, 188.33}, {1.166, 138.38}, {1.787, 245.32}, {0.1, 23.38},
 {0.126, 15.65}, {0.795, 132.52}, {1.06, 136.99}, {1.159, 222.27}}
```



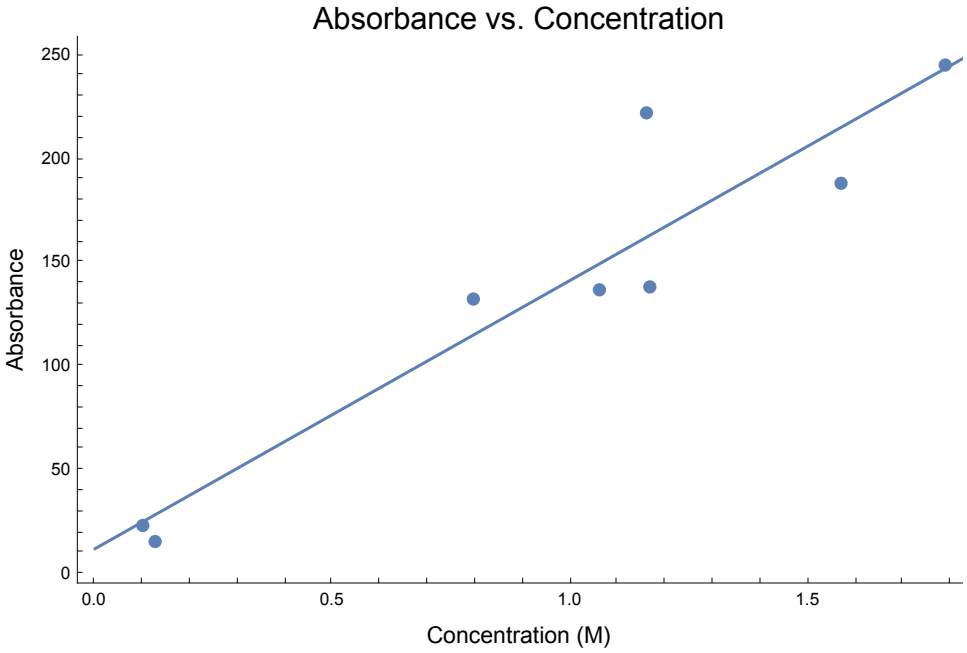
where the title is always y vs. x or dependent vs. independent variable and my axis are properly labeled with appropriate units.

And by graphing, it is easy, even obvious, to visualize the trend line between the data. Meaning the data is not completely scattered. An old method of helping your eye to see the trend is to angle this printed graph flat with your eye down towards the bottom left of the plot, or its origin. Looking that the plot flat like that will nearly superimpose the points on top of each other and the linear relationship will reveal itself, if it doesn't already for you.

Now I can tell *Mathematica* to fit the points through a least-squares regression and superimpose that line on the data points to see how well the fit is.

```
lmf = LinearModelFit[data, {1, C}, C][{"BestFit", "RSquared"}];
Print["Abs = f[C] = ", fxn = Normal[lmf[[1]]],
 "; correlation coefficient R^2 = ", lmf[[2]], "."]
Show[{ppts, Plot[fxn, {C, 0, 2}]}, ImageSize → 500]
```

```
Abs = f[C] = 12.2241 + 129.5 C; correlation coefficient R^2 = 0.887049.
```



Now you can clearly see that the fitted line best represents all of the discrete data points as a smooth continuous mathematical relation given by equation: Abs = 129.5 C+12.2241, where Abs is the dependent variable of absorbance with dimensionless units and C is the independent variable of concentration with units of molarity.

The other nice feature of having this function is that the unknown measured an absorbance of 90.850 lies between any of the other points, and we would have a more difficult time figuring out the concentration without the function. So I can then solve the function for concentration and plug in my absorbance and I will magically have my concentration, or

```
Solve[Abs == fxn, C][[1, 1]]
% /. Abs → 90.850
```

```
C → -0.00772203 × (12.2241 - Abs)
```

```
C → 0.607151
```

where I would report my final unknown concentration of 0.6072 M with four significant figures.

Example 4

Question: The conductivity of ions in solution is linear at molarities less than 0.01 M. However, at greater concentrations the ions interfere with each other and reduce the mobility and thus the conductivity at increasing concentration given by the relation: $\mu S = aC^{3/2} + bC + c$, where C is the concentration in molarity and the first term, $aC^{3/2}$, accounts for the interference of the ions at higher concentrations. The calibration of conductivity in μS versus the concentration in molarity is given by: {M, μS} = {{3.04256*10^-02, 2.71917*10^+03}, {5.91034*10^-03, 6.57791*10^+02}, {1.22314*10^-03,

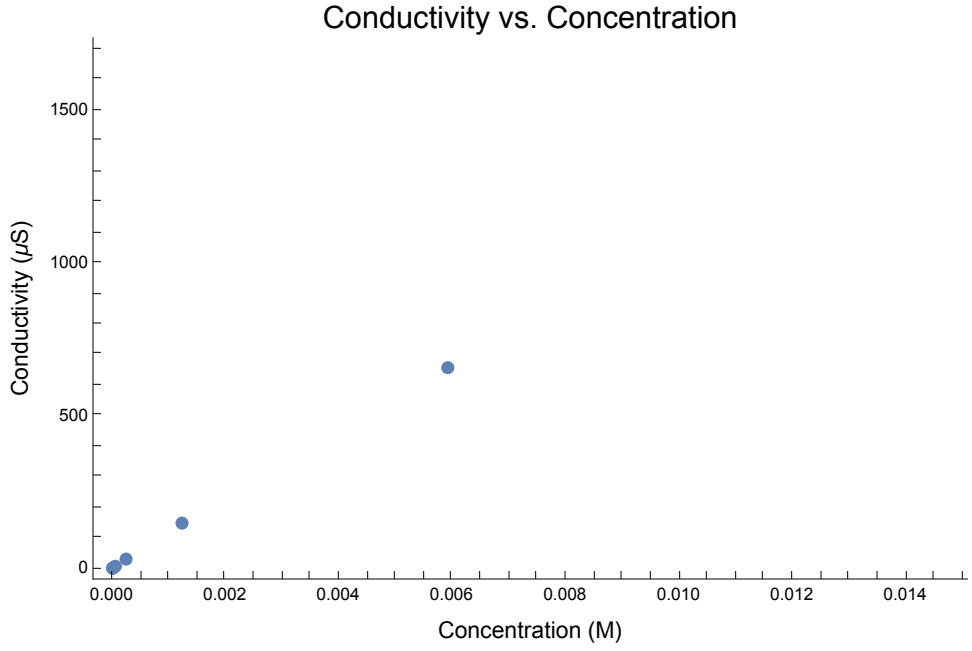
1.48900*10^+02}, {2.36204*10^-04, 3.11846*10^+01}, {4.76091*10^-05, 7.76093*10^+00}, {9.52183*10^-06, 2.94873*10^+00}, {1.89372*10^-06, 1.97758*10^+00}, {3.89330*10^-07, 1.78540*10^+00}, {7.71209*10^-08, 1.74546*10^+00}}. Fit this data in Excel to the above polynomial to find the a, b, and c parameters. Formally plot the points with the superimposed fitted polynomial with the proper title, axis labels and units, etc. What is the unknown molar concentration of electrolytes if the conductivity is measured to be 1.11007*10^+03 μS?

Solution:

The most important thing to see there is that we are fitting the data to an odd polynomial. Excel can fit this data and you can learn how to fit a polynomial like this on many written-out or podcasted tutorials on the internet. Most of these internet tutorials will show you how to fit a quadratic polynomial, i.e., $ax^2 + bx + c$, and you will have to adapt that method to our special polynomial of $ax^{3/2} + bx + c$, to fit our needs here.

```
data = {{3.04256 * 10^-02, 2.71917 * 10^+03},
  {5.91034 * 10^-03, 6.57791 * 10^+02}, {1.22314 * 10^-03, 1.48900 * 10^+02},
  {2.36204 * 10^-04, 3.11846 * 10^+01}, {4.76091 * 10^-05, 7.76093 * 10^+00},
  {9.52183 * 10^-06, 2.94873 * 10^+00}, {1.89372 * 10^-06, 1.97758 * 10^+00},
  {3.89330 * 10^-07, 1.78540 * 10^+00}, {7.71209 * 10^-08, 1.74546 * 10^+00}}
ppts = ListPlot[data, PlotStyle → {PointSize[0.015]},
  PlotLabel → Style["Conductivity vs. Concentration", 16],
  FrameLabel → {Style["Concentration (M)", 12], Style["Conductivity (μS)", 12]},
  Frame → {{True, False}, {True, False}}, ImageSize → 500]
```

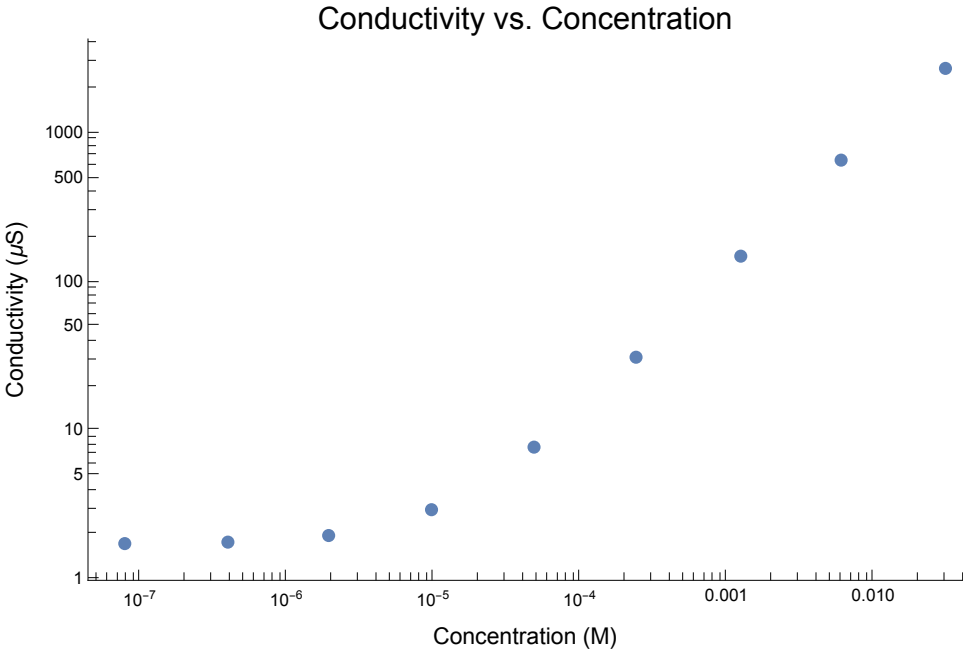
```
{{0.0304256, 2719.17}, {0.00591034, 657.791}, {0.00122314, 148.9},
 {0.000236204, 31.1846}, {0.0000476091, 7.76093}, {9.52183 × 10^-6, 2.94873},
 {1.89372 × 10^-6, 1.97758}, {3.8933 × 10^-7, 1.7854}, {7.71209 × 10^-8, 1.74546}}
```



oh, see how all of the points bunched up there in the bottom left because our data is reaching across many decades (x10). Not good for presentation methods. You will want to stretch out those points using semi-log or log-log plots. In Excel you will choose the log option for that axis to make it look like

this

```
ppts = ListLogLogPlot[data, PlotStyle → {PointSize[0.015]},
  PlotLabel → Style["Conductivity vs. Concentration", 16],
  FrameLabel → {Style["Concentration (M)", 12], Style["Conductivity (μS)", 12]},
  Frame → {{True, False}, {True, False}}, ImageSize → 500]
```

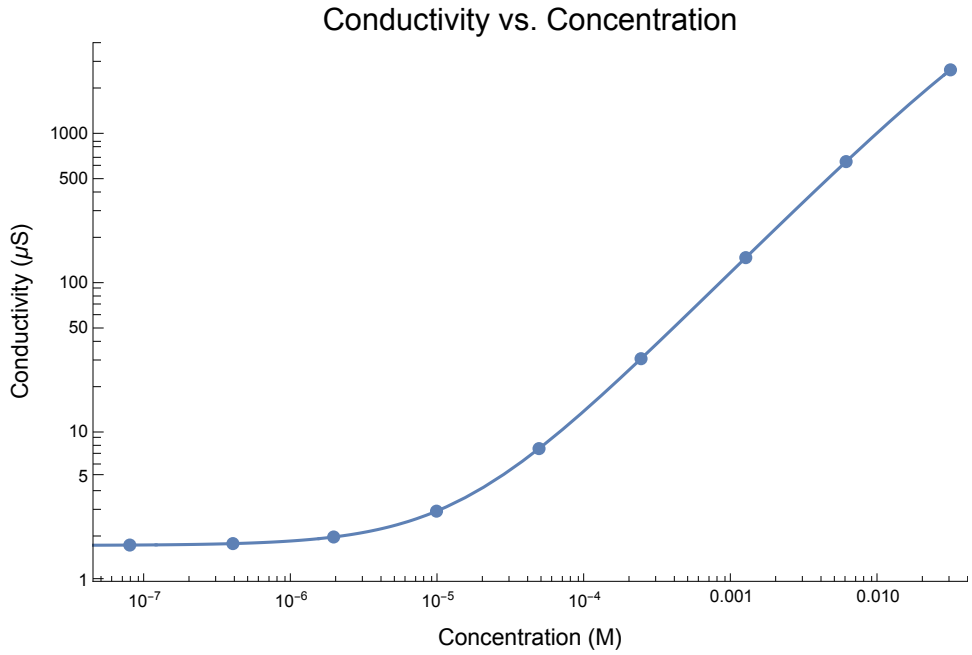


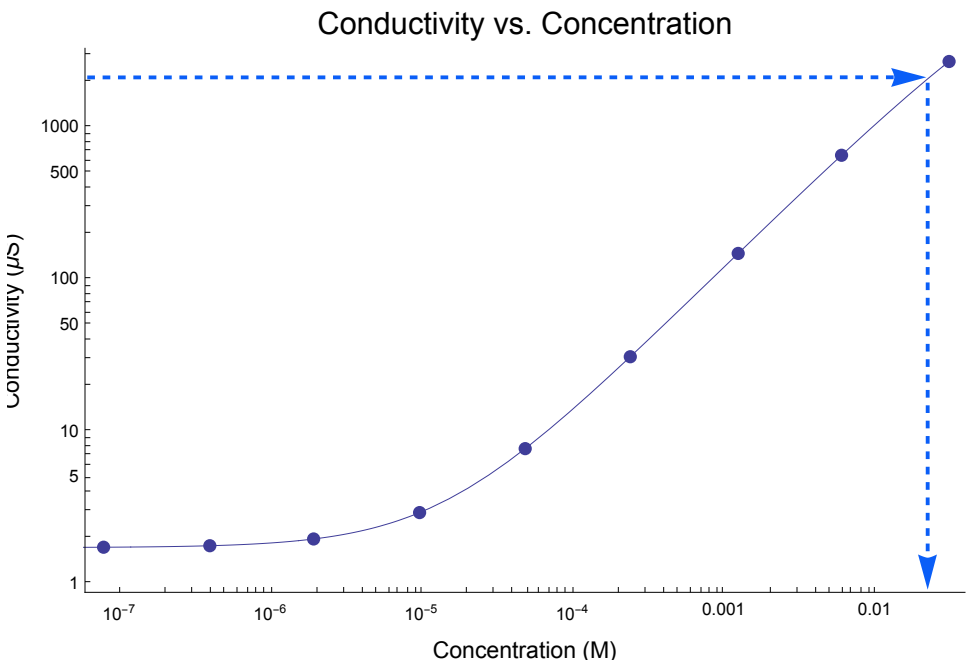
see how nicely the points are represented in this log-log plot. This can be done in Excel also.

Now to fit the data.

```
lmf = LinearModelFit[data, {1, C, C^(3/2)}, C][{"BestFit", "RSquared"}];
Print["μS = f[C] = ", fxn = Normal[lmf[[1]]],
 "; correlation coefficient R^2 = ", lmf[[2]], "."]
Show[{ppts, LogLogPlot[fxn, {C, 10^-8, 3 × 10^-2}]}, ImageSize → 500]
```

```
μS = f[C] = 1.7355 + 128 093. C - 222 317. C^(3/2); correlation coefficient R^2 = 1..
```





where my fit parameters are a = -222317, b = 128093, and c = 1.7355.

Now there is no exact function to solve for C for a $C^{3/2}$order polynomial, like there is for a second, C^2, order polynomial. So we will have to find the unknown either graphically, like that shown above where I backtracked the 1110 μS to the trend line down to the concentration, shown in red.

Or, I can find the root of the expression as long as it is equal to zero. If I start with the above fitted

equation and solve it for zero, I wind up with

$$\mu S = -222317.\, C^{3/2} + 128093.\, C + 1.7355$$

and

$$0 = -222317.\, C^{3/2} + 128093.\, C + (1.7355 - \mu S) \tag{17}$$

and

$$0 = -222317.\, C^{3/2} + 128093.\, C + (1.7355 - 1110.07)$$

by subtracting by unknown μS, it shifts the fitted function so that the function crosses the x-axis exactly at the concentration where the unknown is. Then by finding the root of that function, it will give us the unknown concentration. So in *Mathematica* I do it here

```
fxn - 1110.07
FindRoot[%, {C, 0}]
```

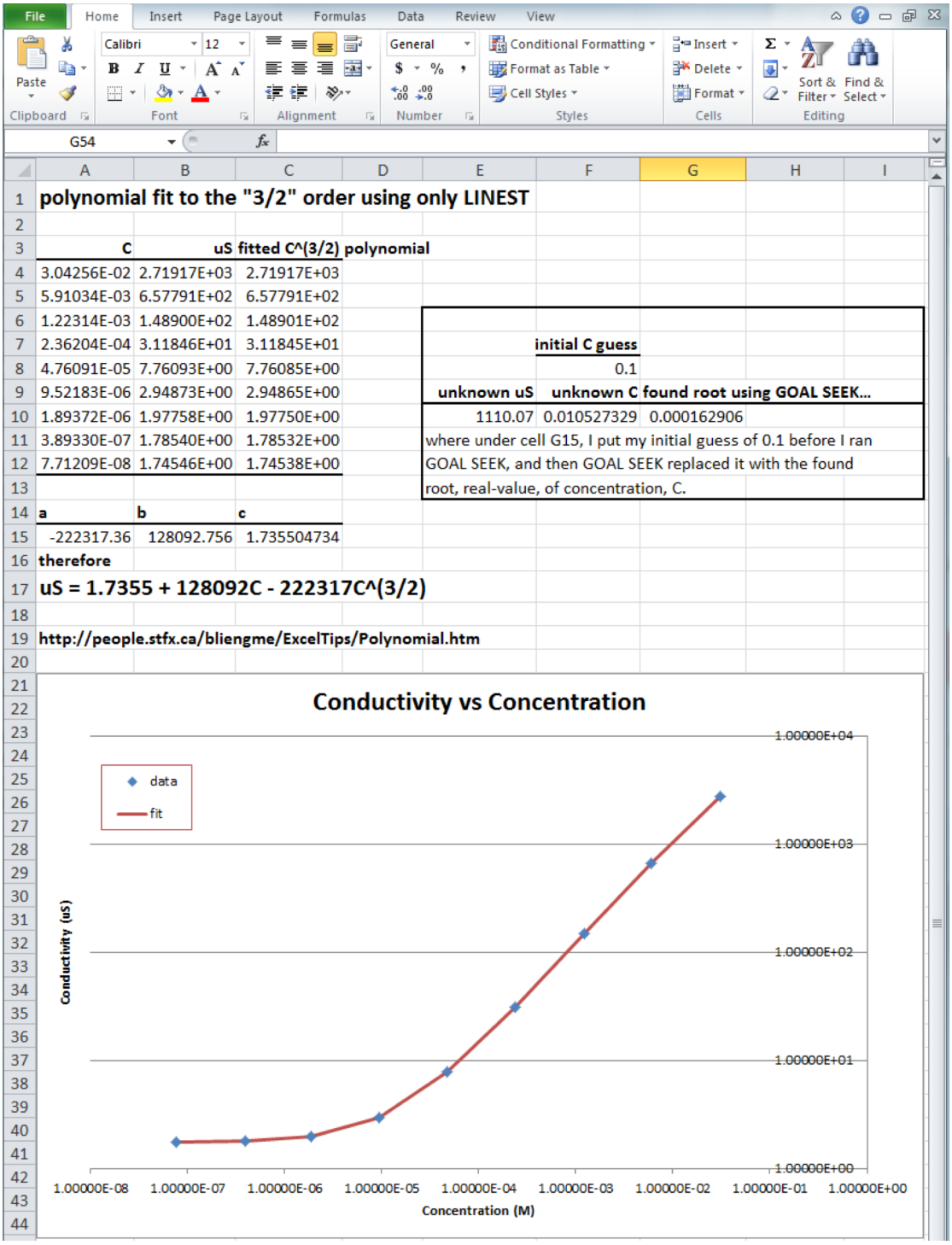
```
-1108.33 + 128093. C - 222317. C^(3/2)
```

```
{C → 0.0105272}
```

which will be properly reported as unknown C = 0.0105272 M to six significant figures.

Excel can do root finding as above, but even your scientific or graphing calculator can do root finding also. It is best to learn how to do it in both. In fact, I learned more about data processing for the laboratory from reading the user's manual for my graphing calculator, then I learned anywhere else.

For the nay-sayers out there that say, "It can't be done in Excel." I say, "Hogwash, Hornswagle, Poppy-cock." I can prove it can be done and it can be done at least two difference ways. Here are the screen shots of my Excel spreadsheets of me doing it two separate ways and if you are reading carefully, clues to the methods are shown in black and white. Please refer to the "fitting__various_techniques3.xls" file for updates and the actual source code of the spreadsheets.



	A	B	C	D	E	F	G	H	I
1	**polynomial fit to the "3/2" order using only LINEST**								
2									
3	**C**	**uS**	**fitted C^(3/2) polynomial**						
4	3.04256E-02	2.71917E+03	2.71917E+03						
5	5.91034E-03	6.57791E+02	6.57791E+02						
6	1.22314E-03	1.48900E+02	1.48901E+02						
7	2.36204E-04	3.11846E+01	3.11845E+01			**initial C guess**			
8	4.76091E-05	7.76093E+00	7.76085E+00			0.1			
9	9.52183E-06	2.94873E+00	2.94865E+00			**unknown uS**	**unknown C**	**found root using GOAL SEEK...**	
10	1.89372E-06	1.97758E+00	1.97750E+00			1110.07	0.010527329	0.000162906	
11	3.89330E-07	1.78540E+00	1.78532E+00			where under cell G15, I put my initial guess of 0.1 before I ran			
12	7.71209E-08	1.74546E+00	1.74538E+00			GOAL SEEK, and then GOAL SEEK replaced it with the found			
13						root, real-value, of concentration, C.			
14	**a**	**b**	**c**						
15	-222317.36	128092.756	1.735504734						
16	**therefore**								
17	**uS = 1.7355 + 128092C - 222317C^(3/2)**								
18									
19	**http://people.stfx.ca/bliengme/ExcelTips/Polynomial.htm**								
20									

Example 5

Question: A very strange ISE probe and configuration returned this calibration voltage versus concentration via serial dilution data: {M, mV} = {{5.09190*10^+00, 1.07792*10^+00}, {5.08547*10^-01, 9.44268*10^-01}, {5.02331*10^-02, 8.60833*10^-01}, {5.00592*10^-03, 7.97702*10^-01}, {5.03086*10^-04, 6.58726*10^-01}, {4.92793*10^-05, 6.22126*10^-01}, {5.06499*10^-06, 6.02176*10^-01}, {4.96558*10^-07, 4.82502*10^-01}, {4.98804*10^-08, 4.67956*10^-01}, {5.07944*10^-09, 4.62284*10^-01}, {5.02837*10^-10, 3.87707*10^-01}, {5.06658*10^-11, 3.38326*10^-01}, {4.93057*10^-12, 2.87455*10^-01}, {4.95690*10^-13, 2.80967*10^-01}}. Fit the raw data and determine the best equation to linearize the data. Then, find the slope, y-intercept, and correlation coefficient (R^2) for the linearized equation. Formally plot the lin-

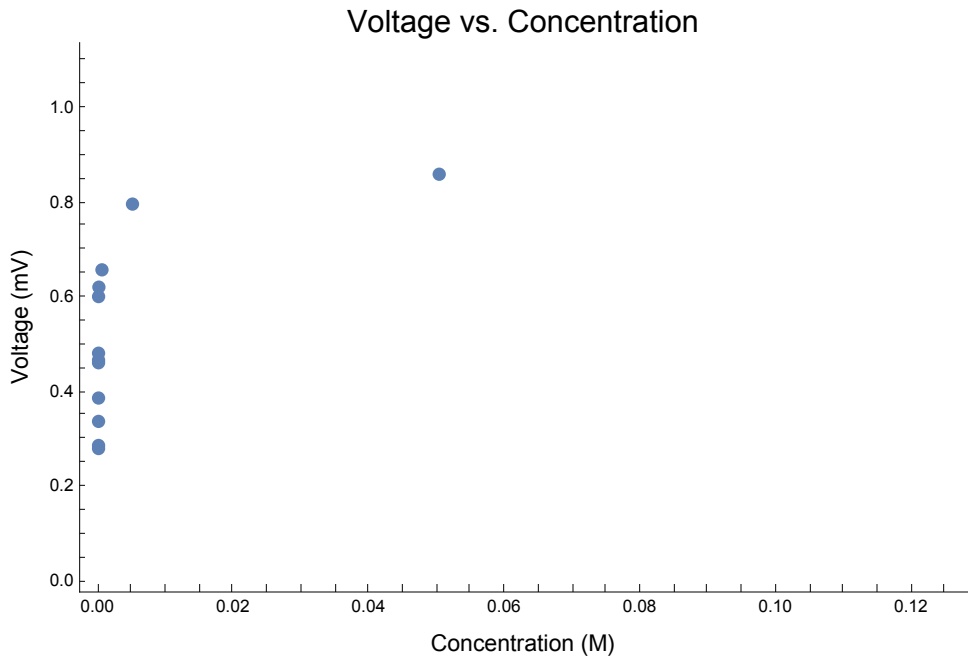
earized data with the superimposed fitted line with the proper title, axis labels and units, etc. What is the unknown molar concentration of the ion if the potential is measured to be 8.1782*10^-01 mV?

Solution:

Again, raw data makes no sense until you plot it.

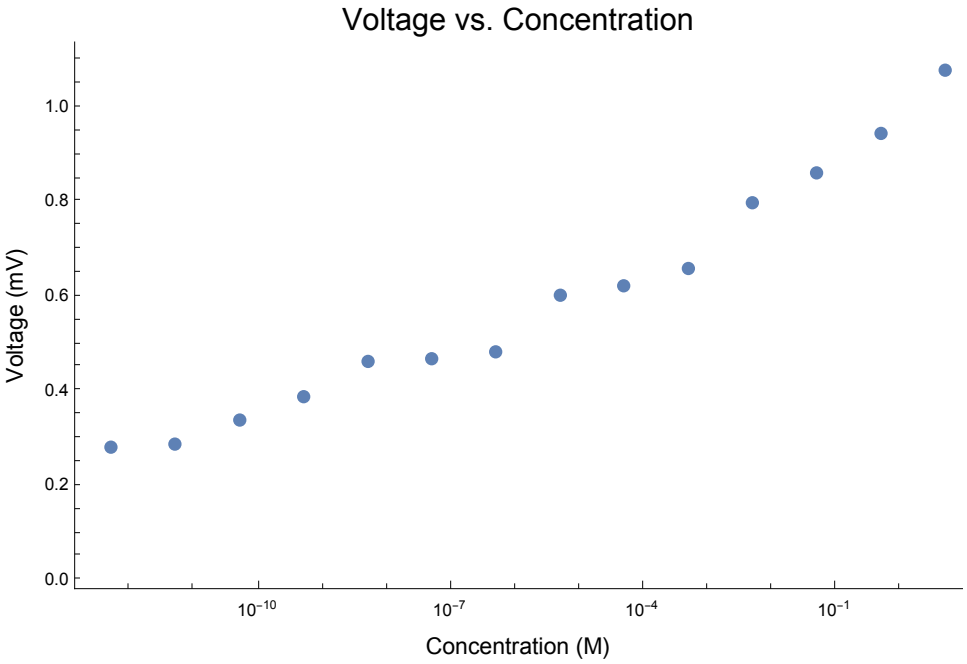
```
data = {{5.09190 * 10^+00, 1.07792 * 10^+00}, {5.08547 * 10^-01, 9.44268 * 10^-01},
  {5.02331 * 10^-02, 8.60833 * 10^-01}, {5.00592 * 10^-03, 7.97702 * 10^-01},
  {5.03086 * 10^-04, 6.58726 * 10^-01}, {4.92793 * 10^-05, 6.22126 * 10^-01},
  {5.06499 * 10^-06, 6.02176 * 10^-01}, {4.96558 * 10^-07, 4.82502 * 10^-01},
  {4.98804 * 10^-08, 4.67956 * 10^-01}, {5.07944 * 10^-09, 4.62284 * 10^-01},
  {5.02837 * 10^-10, 3.87707 * 10^-01}, {5.06658 * 10^-11, 3.38326 * 10^-01},
  {4.93057 * 10^-12, 2.87455 * 10^-01}, {4.95690 * 10^-13, 2.80967 * 10^-01}}
ListPlot[data, PlotStyle → {PointSize[0.015]},
 PlotLabel → Style["Voltage vs. Concentration", 16],
 FrameLabel → {Style["Concentration (M)", 12], Style["Voltage (mV)", 12]},
 Frame → {{True, False}, {True, False}}, ImageSize → 500]
```

```
{{5.0919, 1.07792}, {0.508547, 0.944268}, {0.0502331, 0.860833},
 {0.00500592, 0.797702}, {0.000503086, 0.658726}, {0.0000492793, 0.622126},
 {5.06499 × 10^-6, 0.602176}, {4.96558 × 10^-7, 0.482502},
 {4.98804 × 10^-8, 0.467956}, {5.07944 × 10^-9, 0.462284},
 {5.02837 × 10^-10, 0.387707}, {5.06658 × 10^-11, 0.338326},
 {4.93057 × 10^-12, 0.287455}, {4.9569 × 10^-13, 0.280967}}
```



Oh, no. It looks like the data is related, but it is all smashed up against the y-axis. Lets expand the plot using a semi-log plot.

```
ListLogLinearPlot[data, PlotStyle → {PointSize[0.015]},
 PlotLabel → Style["Voltage vs. Concentration", 16],
 FrameLabel → {Style["Concentration (M)", 12], Style["Voltage (mV)", 12]},
 Frame → {{True, False}, {True, False}}, ImageSize → 500]
```



which looks much better, at least the points are stretched out nicely. A semi-log plot is easy in Excel, just highlight the x-axis of your original plot, right click the x-axis, and choose the properties or axis properties/options menu item. Then look for the log scale option and check that off. Excel will then give you a log scale on the x-axis, a similar action on the y-axis can make that a log scale also.

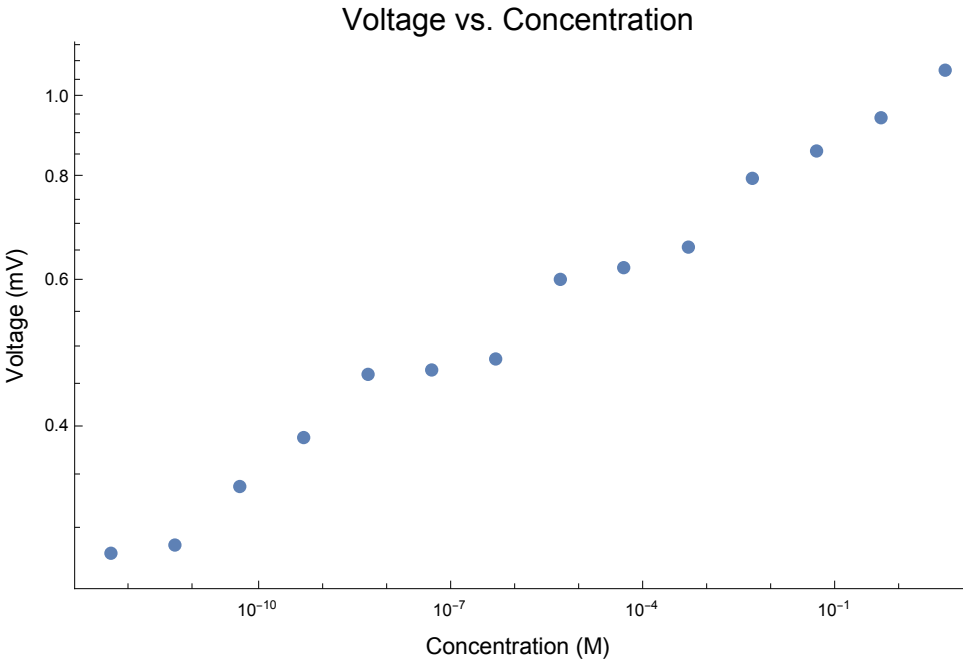
Now at this point, I do see quite a bit of curvature in these points, but if it did linearize out nicely at this point, I could do a linear regression if I take the logs of the molarities (x-axis) and fit a line to that data. The fitted line would then have the form

$$mV = m \cdot \log(M) + b \tag{18}$$

where the fitted linear parameters of slope, m, and y-intercept, b, would be found as normal.

However, since there is some curvature in the semi-log plot, maybe the points would form a nice line if I plotted it on a log-log plot, or

```
ListLogLogPlot[data, PlotStyle → {PointSize[0.015]},
 PlotLabel → Style["Voltage vs. Concentration", 16],
 FrameLabel → {Style["Concentration (M)", 12], Style["Voltage (mV)", 12]},
 Frame → {{True, False}, {True, False}}, ImageSize → 500]
```



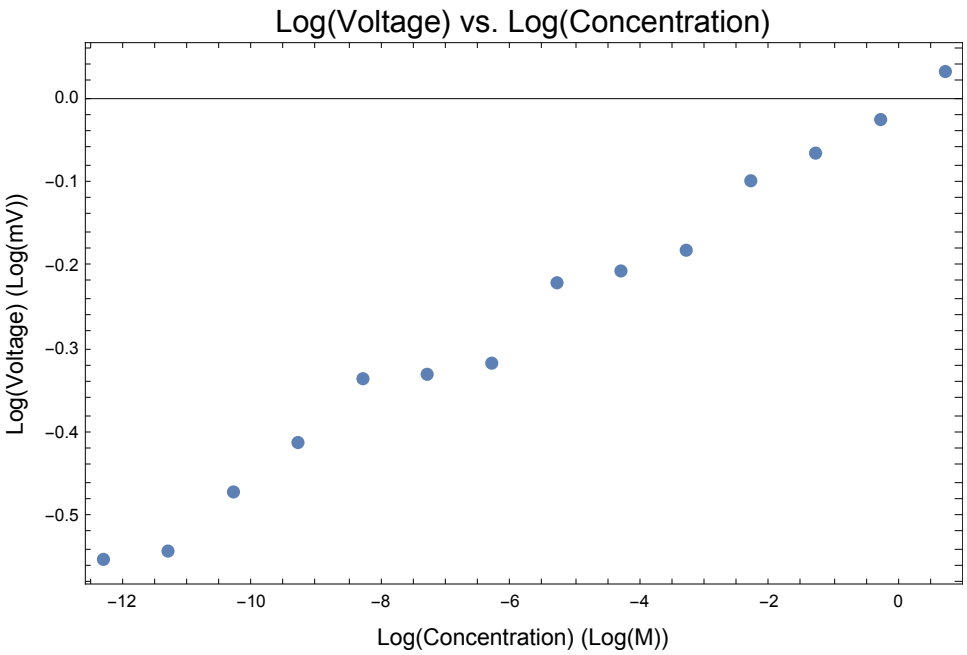
wow, that straightened out nicely. I mean there is some scatter about a line but there is definitely a line. Such a log-log plot is essentially scaling both axis to log scales. Mathematically, this is the same as taking the log of the y data, in this case the mV, and the log of the x data, in our case the molarity, forcing us to fit a line to the equation

$$\log(\mathrm{mV}) = \mathrm{m} \cdot \log(\mathrm{M}) + \mathrm{b} \tag{19}$$

where we still have the basic form of a line with slope, m, and y-intercept, b, but now with the log-log twist. So if I take the logs of my mVs and the logs of my molarities, my data changes to

```
data0 = Table[{Log[10, data[[i, 1]]], Log[10, data[[i, 2]]]}, {i, Length[data]}]
ppts = ListPlot[data0, PlotStyle → {PointSize[0.015]},
  PlotLabel → Style["Log(Voltage) vs. Log(Concentration)", 16],
  FrameLabel → {Style["Log(Concentration) (Log(M))", 12],
    Style["Log(Voltage) (Log(mV))", 12]}, Frame → True, ImageSize → 500]
```

```
{{0.70688, 0.0325865}, {-0.293669, -0.0249047},
 {-1.29901, -0.0650811}, {-2.30052, -0.0981593}, {-3.29836, -0.181295},
 {-4.30734, -0.206122}, {-5.29542, -0.220277}, {-6.30403, -0.316501},
 {-7.30207, -0.329795}, {-8.29418, -0.335091}, {-9.29857, -0.411496},
 {-10.2953, -0.470665}, {-11.3071, -0.54143}, {-12.3048, -0.551345}}
```

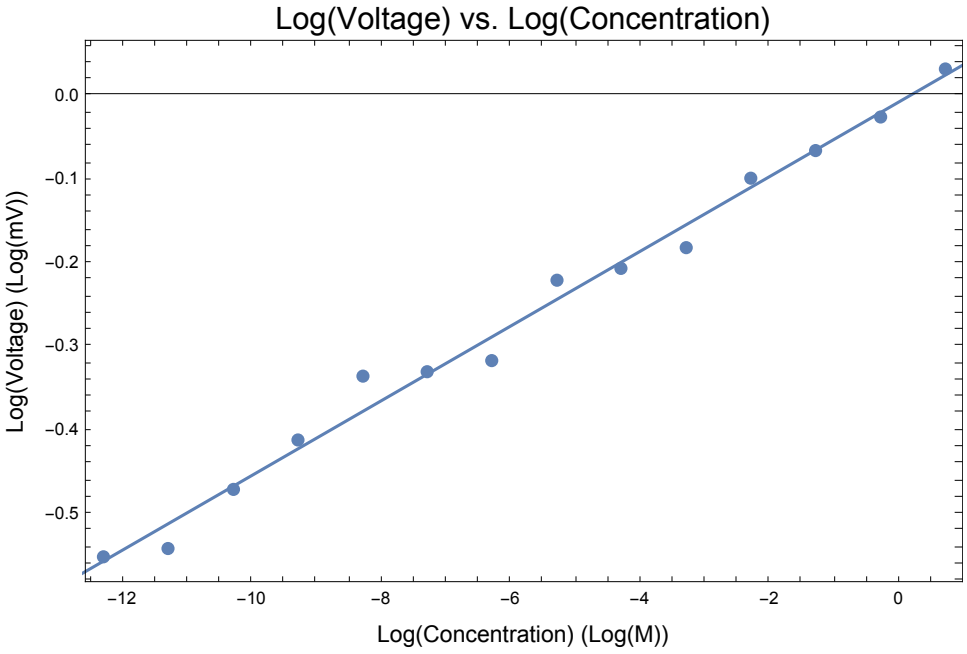


where I plotted the log-log data just to prove that I get the same form of the plot back out.

And now fit that data

```
lmf = LinearModelFit[data0, {1, x}, x][{"BestFit", "RSquared"}];
Print["y = f[x] = ", fxn = Normal[lmf[[1]]],
 "; correlation coefficient R^2 = ", lmf[[2]], "."]
Show[{ppts, Plot[fxn, {x, Log[5 × 10^-13], Log[5]}]}]
```

```
y = f[x] = -0.00637464 + 0.0447121 x; correlation coefficient R^2 = 0.98883.
```



where the fitted line gave us the fit of y = 0.0447121·x-0.00637464, but remember that we had to linearize the data through a log-log plot, so y=log(mV) and x=log(M), which gives us the final equation of

$$\log(\mathrm{mV}) = 0.0447121 \cdot \log(\mathrm{M}) - 0.00637464 \tag{20}$$

and I can use this final equation and use it to solve for my unknown, or

$$\mathrm{M} = 10^{(\log(\mathrm{mV}) + 0.00637464)/0.0447121} \tag{21}$$

or in *Mathematica* speak

```
Solve[Log[10, mV] == (fxn /. x → Log[10, C]), C][[1, 1]]
% /. mV → 0.81782
```

```
C → 1.38858 mV^22.36531300376363
```

```
C → 0.0154573
```

which would be properly reported as 0.015457 M for the unknown concentration at five significant figures.

Summary

These least square fitting techniques are extremely powerful and very necessary in instrumental analysis. It is a very common practice to calibrate an instrument with known standards and serial dilutions to finally generate such a fitted relation, and then use that relation to solve for an unknown sample.

Interpolation vs. Extrapolation

Once data XY data is fit to a curve, it is common to solve for that equation, then, afterward, having a new measurement for Y, and then using the solved equation to get the X value. Many instruments are calibrated in this manner.

For example, as in the previous example from the text, page 987, we fit the Peak Area vs. Mole Percent Isooctane and go the above straight line in red, giving the equation expressed to the proper significant figures:

```
y == 2.09 x + 0.26
Solve[%, x]
solvedExpression = Expand[%[[1, 1, 2]]]
```

```
y == 0.26 + 2.09 x
```

```
{{x → -0.478469 × (0.26 - y)}}
```

```
-0.124402 + 0.478469 y
```

So, then taking a new measurement on a new unknown sample, my instrument kicks out a peak area of 3.57 of isooctane. I would take this peak area, Y, and plug it into the solved expression to get the mole percent of isooctane, x, or

```
solvedExpression /. y → 3.57
```

```
1.58373
```

To find out that I have a mole percent of 1.58 of isooctane in my solution.

This was done through interpolation. Interpolation is when you have a solved equation through some fitted function for some real data, and the unknown was within the limits of the fitted data. Or, in the example, the Y of 3.57 was within the range of the originally fitted data, which was between [1.09, 4.01] for Y. That is interpolation, the unknown value is within the range of the original data. And as such, so was the X value within range of the original data range of [0.352, 1.75]. Interpolation is very good and ensures that your unknown value is valid.

Extrapolation can lead to problems. Extrapolation is when the unknown values are outside the range of the originally fitted data. In this example, outside would mean Y<1.09 or Y>4.01. Since chromatography usually has a zero measurement for zero concentration, then for Y<1.09, that kind of extrapolation may not be that bad. So, if my peak area Y was measured to be 0.39 giving an mole percent X=0.06, or

```
solvedExpression /. y → 0.39
```

```
0.062201
```

However, extrapolation can give increasing bad results the farther we go out from the reasonable range. In this example, in the other direction, Y>4.01. Now one can justify a reasonable answer if we are within an order of magnitude within the range, like if Y were to equal 7.51 giving

```
solvedExpression /. y → 7.51
```

```
3.4689
```

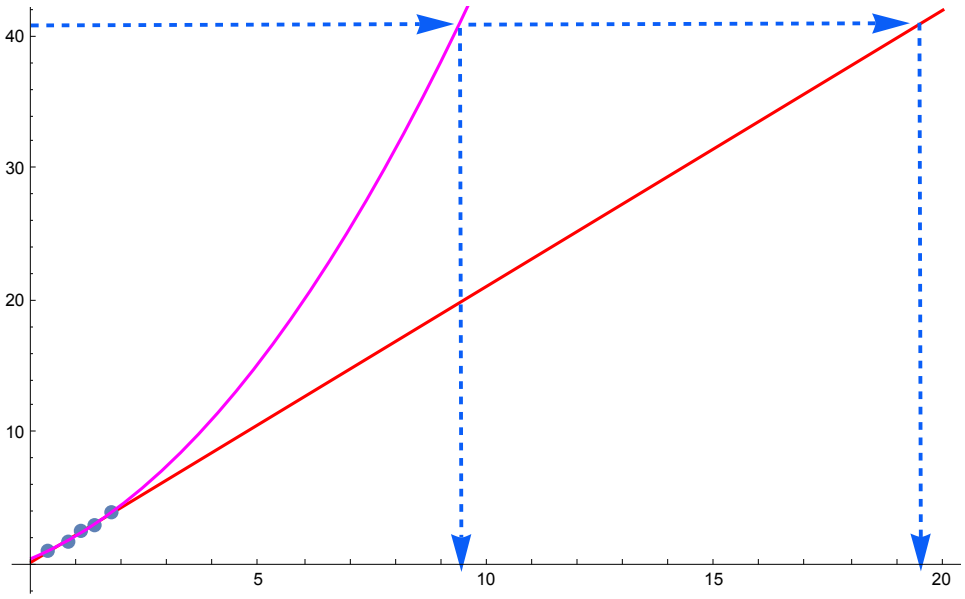
and see X is also out of its original range but probably still a reasonable answer with reasonable accu-

But when we are extrapolating far outside the limits of the original data, things can go awry.

```
2.09 x + 0.2567
0.3047 x^2 + 1.453 x + 0.521
Show[Plot[Out[-2], {x, 0, 20}, PlotStyle → Red],
 points, Plot[Out[-1], {x, 0, 20}, PlotStyle → Magenta],
 PlotRange → {{0, 20}, {0, 40}}, ImageSize → 500]
```

$$0.2567 + 2.09\ x$$

$$0.521 + 1.453\ x + 0.3047\ x^2$$



where the red line is the previously fitted line, the magenta line is the fitted quadratic, which is actually a better fit within the original data range. But what I am trying to show here is that the farther we move out of the range of the originally fitted data, the faster things can diverge and reach into the unknown and farther from the probably true values. For example, the blue arrows, if your measurement were Y=40.5, your mole percent would be

```
solvedExpression /. y → 40.5
```

$$19.2536$$

or if we used the quadratic fit, we would get

```
Solve[y == 0.3047 x^2 + 1.453 x + 0.521, x]
%[[2, 1, 2]]
% /. y → 40.5
```

Solve: Solve was unable to solve the system with inexact coefficients. The answer was obtained by solving a corresponding exact system and numericizing the result.

$$\left\{\left\{x \to 0.00164096 \times \left(-1453. - 0.447214\ \sqrt{7.38107 \times 10^6 + 6.094 \times 10^6\ y}\right)\right\},\right.$$
$$\left.\left\{x \to 0.00164096 \times \left(-1453. + 0.447214\ \sqrt{7.38107 \times 10^6 + 6.094 \times 10^6\ y}\right)\right\}\right\}$$

$$0.00164096 \times \left(-1453. + 0.447214\ \sqrt{7.38107 \times 10^6 + 6.094 \times 10^6\ y}\right)$$

$$9.3158$$

or for a measured peak area of 40.5 we could get a value of 9.32 or 19.3 mole percent isooctane, or anything in between. In reality, one could easily argue that the true mole percent could be anywhere between 2.5 and 1,000, and they would be right.

The final point to be reiterated, is that the fitted data within the original data range fits well, either the line or quadratic. Even the curved quadratic looks “linear” within the fitted data range. However, upon zoom out, moving far beyond that data range to extrapolate a reasonable answer could be dangerous and potentially wildly inaccurate. You don’t want to base your answers, or truths, on fits based on extrapolation. And the farther from the original data, the exponentially increased problems you will have in your final answers.

To avoid extrapolating, just increase your range of original data by taking measurements farther out and to include your unknown range, refit your curves, and then you can insure you are back to interpolation. This is called characterizing your instrument to your samples.

Standard Addition

Referring back to the introduction of the previous section “Calibration Curves, Curve Fitting, and Method of Least Squares”, many times measured responses are not due to environmental or instrumental fluctuations, but more directly to the sample itself. For instance, a Sodium Ion Selective Electrode (Na ISE) will of course measure the response of sodium in a solution, but other interfering ions such as potassium, lithium, or even hydrogen cations may contribute to the measured signal because they all have a +1 charge. Cations with a +2 charge may also interfere. Any contribution, detraction, or interference of the measured response by anything in the sample, other then the analyte itself, is known as the matrix effect. Typically, matrix effects should be closely duplicated in standard solutions when generating calibrations curves, as discussed in the previous section, so as to null out their effects on the particular analyte. For instance, measuring sodium in ocean or ground water can be a challenge since ocean and ground water contains a host of cations, and thus matrix effects, including potassium, calcium, and magnesium.

The method of standard addition adds a set of small discrete amounts of the purified analyte to the sample itself and measured for each addition’s responses. By adding the purified analyte to the sample itself, the matrix effects are already present in the sample, and by addition of the analyte the changes in corresponding measured responses will largely be that of only the analyte since the interfering compounds and matrix effects are perfectly duplicated in the sample itself.

Please refer to the “fitting__various_techniques3.xls” file for updates and the actual source code of the spreadsheets.

from Harris, Example from Section 5-3:

Serum containing Na^+ gave a signal of 4.27 mV in an atomic emission analysis. Then 5.00 mL of 2.08 M NaCl were added to 95.0 mL of serum. This spiked serum gave a signal of 7.98 mV. Find the original concentration of Na^+ in the serum.

Solution: When only a single standard addition is added to the sample, then we may use the equation:

$$\frac{[X]_i}{[S]_f + [X]_f} = \frac{I_X}{I_{X+S}}$$

and (22)

$$[X]_f = [X]_i \frac{V_0}{V} \quad \text{and} \quad [S]_f = [S]_i \frac{V_S}{V}$$

where X refers to the unspiked analyte and S is the spiked analyte. I_X is the measured response of the unspiked unknown sample, I_{X+S} is the measured response of the spiked sample, $[X]_i$ is the initial concentration of the unspiked analyte or simply, the concentration of the original unknown analyte, $[X]_f$ is the final concentration of the unknown analyte after being spiked, $[S]_i$ is the initial concentration of the spiking solution before being added to the sample, $[S]_f$ is the final concentration of the spiking solution after being added to the sample, V_0 is the initial volume of the unspiked unknown solution, V_S is the volume of the spike solution added to the sample, and $V = V_0 + V_S$ which is the total volume of the solution after spiking the sample.

Follow the numbers carefully as they are substituted from the problem into the above equations:

```
V = 5.00 + 95.0
Sf = 2.08 * 5.00 / V
Xf = Xi * 95.0 / V
Xi / (Sf + Xf) == 4.27 / 7.98
Solve[%, Xi]
```

```
100.
```

```
0.104
```

```
0.95 Xi
```

$$\frac{Xi}{0.104 + 0.95\ Xi} == 0.535088$$

```
{{Xi → 0.113185}}
```

or there is 0.113 M Na^+ in the original serum sample.

If the spiked serum gave a signal of 6.50 mV, what was the original concentration of Na^+?

```
Xi / (Sf + Xf) == 4.27 / 6.5
Solve[%, Xi]
```

$$\frac{Xi}{0.104 + 0.95\ Xi} == 0.656923$$

```
{{Xi → 0.181739}}
```

or there is 0.182 M Na^+ in the original serum sample.

from Harris, Excel Example from Section 5-3:

A variation of the latter example is to perform more then one standard addition to generate a line with more than only two points. This is a reproduction from the Excel spreadsheet provided in full under Canvas as provided by the Harris text. Apparently, they're measuring ascorbic acid (vitamin C) in orange juice by measuring the current through the solution using electrochemistry. The standard solution has an initial concentration of 279 mM, Si below, and they're probably using a buret to add multiple increments of the standard solution as given in VS below and cells B7..B15 in the spreadsheet with the corresponding measured responses in ISX below and cells C7..C15 in the spreadsheet. The initial volume of the sample is 50 mL, V0 below.

As you'll find, multiple additions set up a line with the pure sample at x = 0. The x-intercept in the negative direction is actually the initial unknown concentration of the sample. Therefore, once the line is fit to the (x,y) points, setting y = 0 and solving for x will return the concentration of the unknown. Standard addition is a very clever method of determining unknown concentrations at the same time of nullifying matrix effects. Sweet...

```
V0 = 50
Si = 279
VS = {0, .05, .25, .4, .55, .7, .85, 1, 1.15}
ISX = {1.78, 2, 2.81, 3.35, 3.88, 4.37, 4.86, 5.33, 5.82}
X = Si * VS / V0;
Y = (ISX * (V0 + VS)) / V0;
data = N[Transpose[{X, Y}]]
lst = ListPlot[data];
lmf = LinearModelFit[data, {1, C}, C][{"BestFit", "RSquared"}];
Print["mV = f[C] = ", fxn = Normal[lmf[[1]]],
 "; correlation coefficient R^2 = ", lmf[[2]], "."]
Show[{lst, Plot[fxn, {C, -3, 7}]}, ImageSize → 500,
 PlotRange → {-3, 7}, AxesOrigin → {0, 0}]
Abs[Solve[fxn == 0, C][[1, 1, 2]]]
```

```
50
```

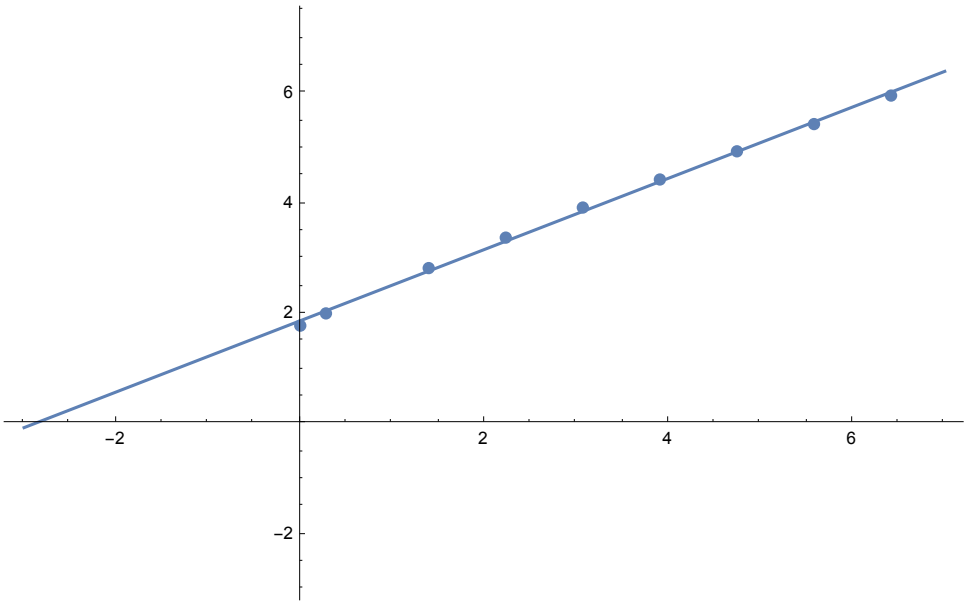
```
279
```

```
{0, 0.05, 0.25, 0.4, 0.55, 0.7, 0.85, 1, 1.15}
```

```
{1.78, 2, 2.81, 3.35, 3.88, 4.37, 4.86, 5.33, 5.82}
```

```
{{0., 1.78}, {0.279, 2.002}, {1.395, 2.82405},
 {2.232, 3.3768}, {3.069, 3.92268}, {3.906, 4.43118},
 {4.743, 4.94262}, {5.58, 5.4366}, {6.417, 5.95386}}
```

```
mV = f[C] = 1.8687 + 0.6463 C; correlation coefficient R^2 = 0.998333.
```

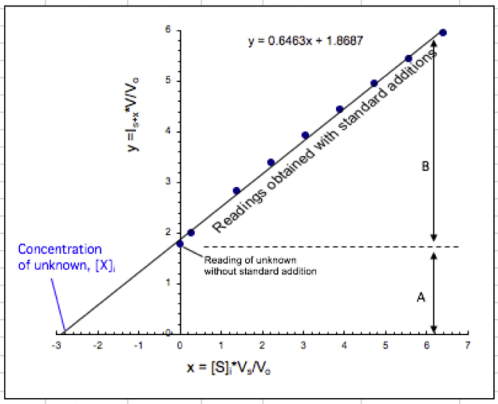



```
2.89139
```

by finding the x- and y-axis values and plotting then fitting the line, the concentration of the original unknown analyte will be the absolute value of the x-intercept when y = 0. The final concentration of the unknown is 2.89 mM ascorbic acid in orange juice. Notice that all of the volumes have to be the same, in this case all are in mL, and since the initial standard concentration is mM, then the final answer must be in millimolar (mM) also.

A screen shot of the Excel spreadsheet is included here for completeness, wherein the cell expressions are clearly marked also:

	A	B	C	D	E
1	Vitamin C standard addition experiment				
2	Add 0.279 M ascorbic acid to 50.0 mL of orange juice				
3					
4		Vs =			
5	Vo (mL) =	mL ascorbic	I(s+x) =	x-axis function	y-axis function
6	50	acid added	signal (μA)	Si*Vs/Vo	I(s+x)*V/Vo
7	[S]i (mM) =	0.000	1.78	0.000	1.780
8	279	0.050	2.00	0.279	2.002
9		0.250	2.81	1.395	2.824
10		0.400	3.35	2.232	3.377
11		0.550	3.88	3.069	3.923
12		0.700	4.37	3.906	4.431
13		0.850	4.86	4.743	4.943
14		1.000	5.33	5.580	5.437
15		1.150	5.82	6.417	5.954
16				D7 = $A$8*B7/$A$6	
17				E7 = C7*($A$6+B7)/$A$6	
18					
19	B21:D24 = LINEST(E7:E15,D7:D15,TRUE,TRUE)				
20					
21		LINEST output:			
22	m	0.646299952	1.8687043	b	
23	s_m	0.009980942	0.0374181	s_b	
24	R^2	0.998333334	0.0644701	s_y	
25					
26	x-intercept =	-2.89139	= -b/m		
27					
28	n =	9	= COUNT(B7:B15)		
29	Mean y =	3.852	= AVERAGE(E7:E15)		
30	$\Sigma(x_i - \text{mean } x)^2$ =	41.722776	= DEVSQ(D7:D15)		
31					
32	Std deviation of				
33	x-intercept =	0.09787			
34	B33 = (C24/ABS(B22))*SQRT((1/B28) + B29^2/(B22^2*B30))				



from Harris, Reworked Example from Section 5-3:

Even there are only two data points, I decided to rework the first standard addition example, amount of Na^+ in serum example, in the same way as the previous example by properly computing the (x,y) values, plotting, and using linest in Excel to reproduce the x-intercept...

```
V0 = 95
Si = 2.08
VS = {0, 5}
ISX = {4.27, 7.98}
data = Transpose[{Si * VS / V0, (ISX * (V0 + VS)) / V0}]
lmf = LinearModelFit[data, {1, C}, C][{"BestFit", "RSquared"}];
Print["mV = f[C] = ", fxn = Normal[lmf[[1]]],
 "; correlation coefficient R^2 = ", lmf[[2]], "."]
Show[{ListPlot[data], Plot[fxn, {C, -.15, .15}]},
 ImageSize → 500, PlotRange → All, AxesOrigin → {0, 0}]
Abs[Solve[fxn == 0, C][[1, 1, 2]]]
```

```
95
```

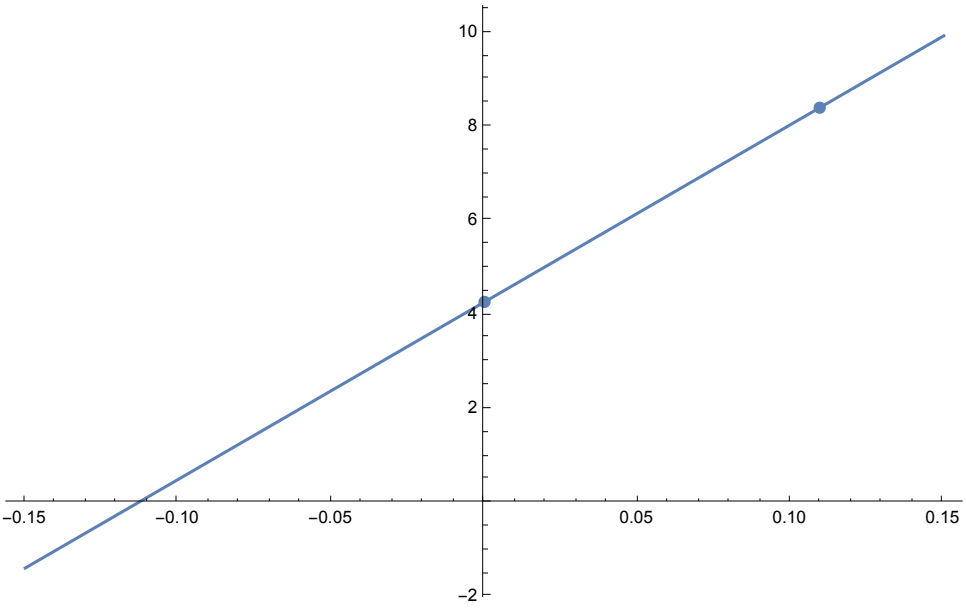
```
2.08
```

```
{0, 5}
```

```
{4.27, 7.98}
```

```
{{0., 4.27}, {0.109474, 8.4}}
```

```
mV = f[C] = 4.27 + 37.726 C; correlation coefficient R^2 = 1..
```




```
0.113185
```

with a final concentration of 0.113 M Na^+ in serum, as verified with the 1st example. As you can see, this method of plotting and fitting the line to the standard additions and finding the x-intercept is a beautiful method of handling unknowns in complex samples with matrix effects.

from the Simulator:

A combination of conductivity with standard addition was used to test the concentration of magnesium sulfate (Epsom Salt) in a hot water spring in Russia that claims healing powers because of the high salinity. The conductivity of the raw unprocessed sample from this spring measured 147.13 µS/cm for an initial volume of 50.00 mL. Although conductivity is rather unspecific to the measurement of particular ions, standard addition of 5.00 mL increments of a 419 mM MgSO4 standard solution pro-

vided the following conductivity responses: [147.13, 180.46, 205.16, 239.48, 248.78, 269.27, 309.27] μS/cm. Using Excel, compute and plot the [I(s+x)*(V0+Vs)/V0)] vs. [Si*Vs/V0] data and fit the data using least-square line (aka linear regression via linest) fitting. Find the slope, y-intercept, and correlation coefficient (R2) and report the final concentration, in mM, of MgSO4 of the magical spring.

```
V0 = 50
Si = 419
VS = {0.0, 5.0, 10.0, 15.0, 20.0, 25.0, 30.0}
ISX = {147.13, 180.46, 205.16, 239.48, 248.78, 269.27, 309.27}
X = Si * VS / V0;
Y = (ISX * (V0 + VS)) / V0;
data = N[Transpose[{X, Y}]]
lst = ListPlot[data];
lmf = LinearModelFit[data, {1, C}, C][{"BestFit", "RSquared"}];
Print["mV = f[C] = ", fxn = Normal[lmf[[1]]],
 "; correlation coefficient R^2 = ", lmf[[2]], "."]
Show[{lst, Plot[fxn, {C, -110, 310}]}, ImageSize → 500,
 PlotRange → {0, 500}, AxesOrigin → {0, 0}]
Abs[Solve[fxn == 0, C][[1, 1, 2]]]
```

```
50
```

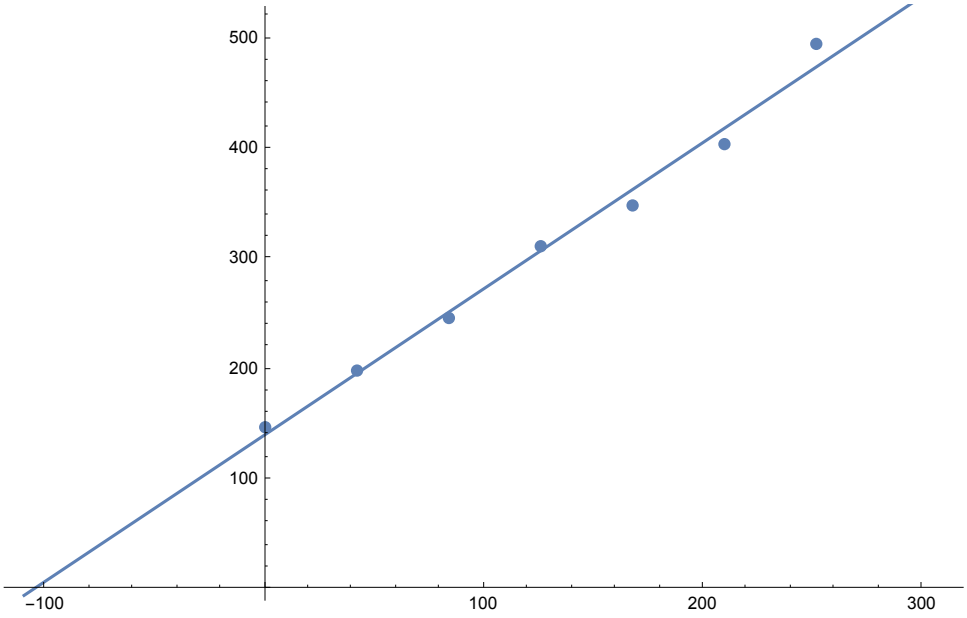
```
419
```

```
{0., 5., 10., 15., 20., 25., 30.}
```

```
{147.13, 180.46, 205.16, 239.48, 248.78, 269.27, 309.27}
```

```
{{0., 147.13}, {41.9, 198.506}, {83.8, 246.192}, {125.7, 311.324},
 {167.6, 348.292}, {209.5, 403.905}, {251.4, 494.832}}
```

```
mV = f[C] = 140.454 + 1.32629 C; correlation coefficient R^2 = 0.989106.
```




```
105.9
```

where the final concentration of MgSO4 in the unknown Russian Spring water turns out to be 105.9 mM

Please refer to the “fitting__various_techniques3.xls” file for updates and the actual source code of the spreadsheets.

Conclusion

Properly analyzing data from real measurements of valid and well-thought experiments can lead to a vast understanding of our universe. All of our physical understanding to this day led from this process of developing and running valid experiments to accurately and repeatably observe the universe, and then conforming mathematical models based on physical parameters to those observations.