Reminder

• I am not there!

• HW11 due **Monday 23 April** electronically by 6am

• Join Piazza!
  – NB: It is easiest to join Piazza by first authenticating through NetBadge, accessing our UVaCollab site, and then connecting to Piazza from the link on the left-hand sidebar

• Office hours: all held in our computer lab, room 022-C of this bldg
  – Me: After lecture 3:30-4:30 every Tuesday ← CANCELED TODAY 😞
  – TAs:
    • Mondays: 3-5pm and 6-8pm
    • Wednesdays: 5-9pm
Review and Outline

• Last time:
  – Gaussian statistics

• Today:
  – Other probability distributions
  – Gaussian limit
  – Interpreting experimental uncertainties
  – Fitting a model to data
  – Problem solving
  – Debugging
  – Good coding style
More Probability Distributions
The Bernoulli Distribution: Success or Failure?
The Bernoulli Distribution:

- Only two possible outcomes (true or false, success or failure).
- The probability, $p$, of one possible outcome is known.

\[
P(\text{heads}) = p \\
P(\text{tails}) = (1-p)
\]
The Bernoulli Distribution

- Only two possible outcomes (true or false, success or failure).
- The probability, $p$, of one possible outcome is known.

\[
P(\text{heads}) = p \\
P(\text{tails}) = (1-p)
\]

Questions:
The Bernoulli Distribution

Questions:

What is the $p$ for tossing heads in one trial?
The Bernoulli Distribution

- Only two possible outcomes (true or false, success or failure).
- The probability, $p$, of one possible outcome is known.

$$P(\text{heads}) = p$$
$$P(\text{tails}) = (1-p)$$

Questions:

What is the $p$ for tossing heads in one trial?

Say I have thrown ten heads in a row – what is the probability of doing that?
The Bernoulli Distribution

Questions:

What is the $p$ for tossing heads in one trial?

Say I have thrown ten heads in a row – what is the probability of doing that?

After ten heads, what is probability that next one is heads too?
The outcomes need not have the same probability! For example:

Outcome 1: Roll a 6.

\[ P(6) = p = \frac{1}{6} \approx 0.17 \]

Outcome 2: Roll something else.

\[ P(tails) = (1-p) = \frac{5}{6} \approx 0.83 \]
The Binomial Distribution: How Many Successes in Many Attempts?
What if we flip a coin many times, or if we flip many coins? How many instances of a given outcome (say, heads) should we expect?

The **Binomial distribution** gives the probability of observing a certain number of true results, $x$, after doing $n$ tests.

$p$ is the probability of success on each test.

It's used when we are interested in a number of TRUE/FALSE experiments.

Consider $n$ coins tosses:
TRUE = HEADS,
FALSE = TAILS

What is the probability of getting $x$ HEADs-up if you flip the coin $n$ times?
The Binomial Distribution

Understanding the Binomial Distribution:

- \( x = \) number of successes (e.g., how many heads?)
- \( n = \) number of trials (e.g., how many tosses?)
- \( p = \) probability of success in a single trial.

- \( P(x; n, p) = \) Probability of seeing \( x \) successes after \( n \) trials, given probability \( p \) of success.

\[
P(x; n, p) = \frac{n!}{x!(n - x)!} p^x (1 - p)^{n-x}
\]

Number of possible ways to arrange the \( x \) successes and \( (n-x) \) failures.

Probability of getting \( x \) successes and \( (n-x) \) failures.
1. Consider a single fair 6-sided die
2. Define “success” as rolling a 5
3. Roll the dice 10 times – what is probability of rolling *three* 5’s?
   • n=10, p=1/6, x=3
   • prob(5) = 1/6 hence prob(three 5’s) = (1/6)^3
   • prob(else) = 5/6 hence prob(seven non-5’s) = (5/6)^7
   • But there are many ways to get three 5’s in 10 rolls:
     • 5-5-5-3-2-6-1-2-4-1
     • 5-5-5-3-3-3-2-2-4-6
     • ...
   • The combination term “n choose x” calculates how many such combinations exist. Here “n choose x” = 10 choose 3 = 10! / (3! (10-3)! = 120
The Binomial Distribution

Binomial Distribution for One Fair Coin Flip:

Here's the Binomial distribution with
\( n = 1 \) (one coin flip)
\( p = 0.5, \) 50% chance of heads on a flip
\( 1 - p = 0.5, \) 50% chance of tails on a flip

One toss, 50% heads/tails

\[
P(x; n, p) = \frac{n!}{x!(n-x)!} p^x (1 - p)^{n-x}
\]

\( x = 0 \): getting zero heads

\( x = 1 \): getting one heads
The Binomial Distribution

Binomial Distribution for One Biased Coin Flip:

Here's the Binomial distribution with
\( n = 1 \) (one coin flip)
\( p = 0.3 \), 30% chance of heads on a flip
\( 1 - p = 0.7 \), 70% chance of tails on a flip

One toss, 30% heads/ 70% tails

\( P(x; n, p) = \frac{n!}{x!(n-x)!} p^x (1 - p)^{n-x} \)

\( x = \text{Number of heads} \)

\( x = 0 \)

\( x = 1 \)
The Binomial Distribution

Binomial Distribution for 20 Fair Coin Flips:

20 tosses of a fair coin

\[ P(x; n, p) = \frac{n!}{x!(n-x)!} p^x (1 - p)^{n-x} \]

Notice that the mean value for \( x \) is 10. Could we have predicted this?
The Binomial Distribution

Mean and Variance for the Binomial Distribution:

Given the binomial distribution:

\[ P(x; n, p) = \frac{n!}{x!(n - x)!} p^x (1 - p)^{n-x} \]

We can calculate the mean value (μ) of x:

\[ \mu = np \]

and the variance (σ^2) of x:

\[ \sigma^2 = np(1 - p) \]
The Binomial Distribution

The Effect of Varying \( p \):  

20 tosses, vary \( p \) from 5% to 50%

- \( p = 5\% \)  
  - Very lopsided coin.

- \( p = 30\% \)

- \( p = 50\% \)  
  - Fair coin.
A Special Case of the Binomial Distribution

The Poisson Distribution: How Many Successes in MANY MANY Attempts if \( p \) is REALLY SMALL?
The Poisson Distribution

An interesting special case of the binomial distribution is the one in which:

- The number of trials, \( n \), approaches infinity,
- The probability of success, \( p \), approaches zero,
- The mean number of successes, \( \mu = np \), remains fixed.

As these limits are approached, the binomial distribution can be approximated by the following (much simpler) expression:

\[
P(x; \mu) = \frac{\mu^x}{x!} e^{-\mu}
\]

This is called the Poisson Distribution, and it is valid when \( p \) is small, \( n \) is large and \( \mu \) is some intermediate value.
The Poisson Distribution

How Good is the Poisson Approximation?

Poisson Approximation

\[ P(x; 20, p) \]

\[ x = \text{Number of heads} \]

\[ \text{binomial distribution, } n=20, \ p=0.05 \]

\[ \text{Poisson distribution, } \mu=1 \]

\[ n = 20 \]
\[ p = 0.05 \]
\[ \mu = np = 1 \]
The Poisson Distribution

• Rule of thumb:
  – The Poisson distribution is fairly good approximation of the Binomial distribution if
    • $n > 20$
    • $p < 0.05$
  – The Poisson distribution is an excellent approximation of the Binomial distribution if
    • $n \geq 100$
    • $np \leq 10$

  – So with just a moderate number of trials and reasonably small probability of success, the Poisson is good.
Example: The Poisson Distribution

Radioactive Decay:

Consider a sample of a radioactive material like Uranium.

We point a detector at the sample and count the number of radioactive decays that happen during a five-minute period.

- \( n \) is large, on the order of Avogadro's Number.
- \( p \) is small (it's unlikely that a particular nucleus will decay while we're looking at it).
- \( \mu = np \), the average number of decays in five minutes, is still an appreciable number, since \( n \) is so large.

The results of several 5-minute observations would be Poisson distributed.
Example: The Poisson Distribution

http://www.engineerguy.com/videos/video-lines.htm
The Poisson Distribution

Mean and Variance of the Poisson Distribution:

Given the Poisson distribution:

\[ P(x; \mu) = \frac{\mu^x}{x!} e^{-\mu} \]

We can derive the very simple expression for the variance of \( x \):

\[ \sigma^2 = \mu \]

So the standard deviation is:

\[ \sigma = \sqrt{\mu} \]
The Poisson Distribution and Histograms

Consider the following:

We fill a histogram with a large number of entries, \( n \).

The probability, \( p \), that any given entry will land in a particular bin is small.

This implies that we can use Poisson statistics to describe the variations in the number of counts in a given histogram bin.

If the count in a given bin is \( m \), then the best estimate of the uncertainty in the bin count is \( \sigma = \sqrt{m} \).
The Poisson Distribution and Histograms

Histogram with Error Bars:
The Gaussian Limit
The Poisson Distribution For Large $\mu$

**Poisson Distribution for $\mu \gg 0$:**

Atoms out of 50 decaying within 0.1, 0.5, 1 half life

- $\mu = 3.35$
- $\mu = 14.6$
- $\mu = 25$

As $\mu$ increases, the distribution approaches a Gaussian ("normal") shape.
The Poisson Distribution For Large \( \mu \)

**How Good is this Approximation?**

Atoms out of 50 decaying w/in 0.1, 0.5, 1 half life

\[ \mu = 3.35 \]

\[ \mu = 14.6 \]

\[ \mu = 25 \]

Gaussian approximation.
As we noticed earlier, the Binomial distribution itself looks like a Gaussian when the mean is sufficiently far away from zero.

\[ \mu = np >> 0 \]
The Gaussian Limit: Binomial Distribution

Calculating the binomial distribution for large $n$ is hard...

DeMoivre figured out that large $n$ behavior is as a Gaussian.

$\mu = np >> 0$
Relations Between Distributions

\[ P(x; n, p) = \frac{n!}{x!(n-x)!} p^x (1 - p)^{n-x} \]

\[ P(x; \mu) = \frac{\mu^x}{x!} e^{-\mu} \]

**Binomial**

\[ n \rightarrow \text{Large, } p \rightarrow \text{Small} \]

**Poisson**

\[ n \rightarrow \text{Large, } np = \mu \gg 0 \]

**Gaussian**

\[ P(x; \mu; \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \]

\[ \mu \gg 0 \]
Relations Between Distributions

**Picking the Right Distribution:**

- Need full Binomial description when there is a good chance you see events at both limits: \( x=0 \) and \( x=n \).
- Can use Poisson limit when you are likely to observe events at \( x=0 \), but not likely to see events at \( x>>\mu \) approaching upper limit.

**Diagram:**

- Binomial \( \rightarrow \) Large, \( p \rightarrow \) Small \( \rightarrow \) Poisson
  - \( n \rightarrow \) Large, \( np = \mu >> 0 \)
  - \( \mu >> 0 \)

- Can use Normal/Gaussian limit when number of trials is large and observed data are not likely to land near either the upper or the lower limits.
Interpreting Experimental Uncertainties
**Example: Neutron Decay:**

Left to themselves, neutrons are unstable and decay into protons, electrons and neutrinos. The mean lifetime of a free neutron is predicted to be about **886** seconds (about 15 minutes).

\[ n^0 \rightarrow p^+ + e^- + \bar{\nu}_e \]

What if we do an experiment to measure the lifetime of a free neutron, and we come up with a result of **926 +/- 20** seconds.

Is our result **consistent** with the theoretical prediction?

Note that the theoretical result differs from our result by **2\sigma**.
Probability of Such a Deviation

When we cite a result like “x +/- σ”, we're saying that we believe that if our experiment were repeated many times, the results would be distributed like the graph below. In particular, we expect that only about 4% of the results will be more than 2σ from the mean.

In our neutron decay example, our measured value differed from theory by 2σ.

If theory and experiment were both correct, we'd expect to see a deviation this large about 4% of the time. So, it isn't out of the question that both theory and our experiment are correct.

The agreement in this case isn't great, but it doesn't provide any compelling reason to throw out the theory.
Probability of Such a Deviation

Probability of a Given Deviation:

\[ \frac{1 - erf\left(\frac{N}{\sqrt{2}}\right)}{2} \]
Are These Numbers Trustworthy?

When Can We Trust Numerical Probabilities?

Only when:

- The quantity of interest has been correctly measured. (no important systematic biases).

- Size of error has been correctly calculated. Incorrect errors are disastrous for determining significance of experiment. Recall a $2\sigma$ deviation happens 4% of the time.

But if we underestimate errors by a factor of two, the same result implies a $4\sigma$ deviation. The probability for such a result is only $6E-5$. It's very unlikely we'd ever observe this if the theory is correct. This is a reminder that experimental results are meaningless without uncertainties.

- The form of the experiment's uncertainties are adequately modeled by a Gaussian. The Central Limit Theorem makes this common, but it's not always true.
Deviations That Are Too Small

What if our neutron result had been 888 +/- 20 seconds, compared to the prediction of 886 seconds? These only differ by 0.1\sigma. We'd expect this to happen only about 8% of the time.

When results agree too closely, we need to think about their validity.

8%

- Are we unusually lucky?
- Did we overestimate \sigma?
- Did we look at the prediction and cheat on the experiment?
Assume that we measure two values, A and B, and are interested in their sum, $C = A + B$.

If we know the errors in A and B, what's the error in C?

Let's assume A and B fluctuate randomly and independently (we say that they're uncorrelated).
If we have a function of several variables, \( f(x,y,z...) \), we can use propagation of errors to find the error in \( f \), given the errors in \( x, y, z \)...

\[
\sigma^2_{f(x,y,z...)} = \sigma^2_x \left( \frac{\partial f}{\partial x} \right)^2 + \sigma^2_y \left( \frac{\partial f}{\partial y} \right)^2 + \sigma^2_z \left( \frac{\partial f}{\partial z} \right)^2 \ldots
\]

Applying this to \( C(A,B) = A + B \), we get:

\[
\sigma^2_C = \sigma^2_A + \sigma^2_B
\]

We say that the uncertainties **add in quadrature**, meaning that we add the squares instead of just adding the numbers directly.

For example, if \( \sigma_A = 3 \) and \( \sigma_B = 4 \), we'd have \( \sigma_C = 5 \), since \( 5^2 = 3^2 + 4^2 \).
Errors in Averages:

We can apply this same technique to other combinations of variables. Consider the average, $\bar{A}$, of a bunch of $N$ measurements, $A_1$ through $A_N$.

$$\bar{A} = \frac{1}{N} \sum_{i=1}^{N} A_i = \frac{1}{N} (A_1 + A_2 + A_3 \ldots)$$

By propagation of errors, the error in $\bar{A}$ should be given by the sum:

$$\sigma^2_A = \sigma^2_1 \frac{1}{N^2} + \sigma^2_2 \frac{1}{N^2} + \sigma^2_3 \frac{1}{N^2} \ldots$$

Leading to the result that the error in the average is less than the individual errors (all assumed equal here) by a factor of $\sqrt{N}$:

$$\sigma_{\bar{A}} = \frac{\sigma}{\sqrt{N}}$$

This tells us (as we knew intuitively) that we'll get a more precise value by averaging several measurements.
Propagation of Uncertainty

Errors and Non-linear Functions:

What if \( f(x, y, z, \ldots) \) is very nonlinear in one or more of the variables, and the individual errors, though normally distributed, are not small?

In this case the error on \( f \) may no longer be symmetric.

What if \( x = 88 \pm 1 \) degrees, \( f(x) = \tan(x) \)?

What is the error on \( f(x) \)?
Asymmetrical Errors:

From propagation of errors, we'd say that the error in \( f(x) = \tan(x) \) was:

\[
\sigma_f^2 = (1 + \tan^2(x))^2 \sigma_x^2
\]

For \( x = 88 \pm 1 \) degrees, that would give us \( \sigma_f = 14.3 \), and we might say that our value for \( f \) was \( \tan(88) \pm 14.3 \), or \( f = 28.6 \pm 14.3 \).

But:

\[
\begin{align*}
\tan(88) &\sim 29 \\
\tan(89 = 88+1) &\sim 57 \\
\tan(87 = 88-1) &\sim 19
\end{align*}
\]

So \( f(x) = 29^{+29}_{-10} \) is more appropriate in this case!
Monte Carlo Estimation of Uncertainty

It is always possible to combine errors via a Monte Carlo approach. This can be very useful for complex error propagations.

\[ x = 88 \pm 1 \]

Generate a random distribution for each element, \( x \), in our function.

Then plot the distribution \( f(x) \), where \( x \) is drawn from the random sampling.
The Monte Carlo technique allows us to determine the actual uncertainty distribution on our dependent quantity. This technique is not limited to Gaussian uncertainties, but can be applied to any distribution.

\[\tan(x)\], where \(x\) is distributed according to \(88 \pm 1\)

Error bar represents minimal 68% confidence region.
Comparing Data to a Prediction
Comparing Data to Some Prediction

• This is science at its best!
  0. Prediction
  1. Observation
  2. Comparison
  3. Conclusion
  4. Refine Prediction
  5. Repeat as necessary

• The comparison step is a crucial step in how we arrive at a refined picture of how the world works.
  – That’s our mission as scientists, no?

• Great news: there are numerical methods one can use to do this quantitatively – perfect for executing in computer programs!
How good is this theory?

A Gravity Experiment:

Let's say we're doing an experiment that measures the velocity, $v$, of a falling object at some time, $t$.

We collect some data (values of $v$ at various times) and then try to come up with a general theory (a "model") that is consistent with our observed data.

This is the process of "inductive reasoning", whereby we look at specific data and try to develop generalizations from them.

This is a big part of how science progresses!
How good is this theory?
How good is this theory?

Experimental Results:

Suggestions for a simple model?
How good is this theory?

Question: How well does this model fit the data?
How good is this theory?

Question: How well does this model fit the data?
Which theory is better?

How to arbitrate between these two?
Which theory is better?

The Chi-squared Statistic:

One way to objectively compare the quality of various models is by calculating a number called the $\chi^2$ statistic for each model. As we'll see, models that are more likely to produce the observed set of measurements will have a smaller value for $\chi^2$.

\[
\chi^2 = \sum_{i=1}^{k} \frac{(v_i - v_{\text{model}}(t_i))^2}{\sigma_i^2}
\]

Where "k" is the number of independent measurements. k is called the number of "degrees of freedom".

“The facts once classified, once understood, the judgment based upon them ought to be independent of the individual mind which examines them.”

- Karl Pearson, The Grammar of Science, (1900)
Calculation of Chi²

\[ \chi^2 = \sum_{i=1}^{k} \frac{(v_i - \hat{v}_{model}(t_i))^2}{\sigma_i^2} \]

\[ v_{model}(t) = 30\sin(t) + 40 \]

\[ \chi^2 = (1.1)^2 + (2.3)^2 + (3.0)^2 + \ldots \]
Comparison of Models: Chi² Values

With $\chi^2$, we can quantitatively compare our two models, to see which is more likely to have produced the data we observed.

As you can see at left, our linear model wins the $\chi^2$ contest!

Since $\chi^2$ measures the collective deviation of the data points from the model, the model that produces the smallest $\chi^2$ is most consistent with the data.

We'll prove this a little more rigorously in the next section.
Chi-squared Distribution:

If we just picked one time (say, 5 sec) and repeatedly measured the speed at that time, we'd probably find that our measurements fell into a Normal distribution, like the one at right.

\[
\chi^2 \text{ Distribution, 1 Degree of Freedom}
\]

\[
\chi^2 = \frac{(v_1 - v_{\text{model}}(t_1))^2}{\sigma_1^2}
\]

If we generated a \(\chi^2\) value for each of these speed measurements, the \(\chi^2\) values would fall in a distribution like the one at the left.
Chi$^2$ Distribution

Integral of $\chi^2$ probability (1 degree of freedom), from $\chi^2 = N$ to $\infty$

- $\chi^2 > 1$, 32% …and still be right
- $\chi^2 > 4$, 5% …and still be right
Chi$^2$ Distribution

- So the probability of having a measurement with $\chi^2 > N$ can be determined from this $\chi^2$ distribution.

- This distribution is same as the one for the integral of the Gaussian dist.

- Why use this other thing?
  - We can calculate the $\chi^2$ for more than one data point and easily combine into a single figure of merit.

\[
\begin{align*}
\text{one data point} & \quad = \\
& \quad \text{one degree of freedom}
\end{align*}
\]
More Data Points: More Degrees of Freedom

\[ \chi^2 > 1, \quad 96\% \]

Integral of \( \chi^2 \) probability
(5 degrees of freedom),
from \( \chi^2 = N \) to \( \infty \)

Remember: \[ \chi^2 = \sum_{i=1}^{k} \left( \frac{v_i - v_{\text{model}}(t_i))^2}{\sigma_i^2} \]
Degrees of Freedom

• When comparing a theory to some data, as we are doing here, each compared prediction from the model is called a **degree of freedom** of the comparison
  – comparing 1 data point = 1 degree of freedom
  – comparing 5 data points = 5 degrees of freedom
  – comparing N data points = N degrees of freedom

• The $\chi^2$ distribution **changes as one considers a comparison with more degrees of freedom**
Many Degrees of Freedom

For large num of degrees of freedom $k$, the most probable value of $\chi^2$ is equal to $k$. 

Like many things, the $\chi^2$ distribution approaches a Gaussian (Normal) shape for large values of $k$. 

At large values of $k$, the probability of seeing a $\chi^2$ value in excess of $k$ approaches 50%. 

50% $\chi^2 > k$
Reduced $\chi^2$

When comparing $\chi^2$ values with different degrees of freedom, it's often useful to look at what's called the “reduced $\chi^2$” value. This is just $\chi^2$ divided by the number of degrees of freedom, $k$:

$$\chi_{\text{red}}^2 = \frac{\chi^2}{k}$$

If all of our data points deviated from the model's predictions by 1 standard deviation, the value of the reduced chi-squared would be 1:

$$\chi_{\text{red}}^2 = \frac{1}{k} \sum_{i=1}^{k} \frac{(v_i - v_{\text{model}}(t_i))^2}{\sigma_i^2} = \frac{1}{k} \sum_{i=1}^{k} \frac{\sigma_i^2}{\sigma_i^2} = \frac{1}{k} k = 1$$
As a rough rule of thumb, a reduced chi² of ~1.0 indicates good agreement between samples, given their uncertainties.
Probabilities for Reduced $\chi^2$

A graph like this (from the Particle Data Group's “Review of Particle Properties”) can also be useful for understanding chi-squared values. The contours show the likelihood of observing reduced chi-squared values $> \chi^2/k$ for a given number of degrees of freedom (k).
Probability of being consistent?

For example, say we have 40 data points (k) and we calculate a reduced $\chi^2$ value of 1.4 relative to our model. 1% probability of $\chi^2/k > 1.6$. 5% probability of $\chi^2/k > 1.4$. 

$\chi^2/k$ vs Degrees of freedom k
Probabilities for Reduced $\chi^2$

Notes:

- **Too LARGE reduced chi$^2$** implies poor agreement between theory and data.

- **Too SMALL reduced chi$^2$** implies one could be overfitting the data – the agreement should still be impacted by the uncertainty on each point.

*reduced chi$^2 \approx 1.0$* indicates theory and data are in accord within uncertainties, i.e., measurement collection sometimes high (50%) sometimes low (50%).
Usefulness of $\chi^2$

What Questions Can We Answer with $\chi^2$?

- Is my theory correct?
  - More data might show flaws in your theory. One small data set can't prove that your theory is correct.

- Based on my data, what is the probability that my theory is correct?
  - There are infinitely many possible theories. And what does “correct” mean? Maybe more than one theory is “correct”, in that it makes accurate predictions.

- What is the probability that my theory produced the physics we observe?

- Given a particular theory, what is the probability that data this different or more different could have occurred?

- Are the data inconsistent with the model?

These are the kinds of statements that can be supported by chi-squared values.
Summary so far...

- Compare some data to a model, account for uncertainties
- Calculate reduced $\chi^2$
- If good agreement
  - should see different points sometimes high/low
  - if k large, reduced $\chi^2 \sim 1.0$
Tuning a Model to Best Match Some Data
A Parametrized Model:

\[ v_{\text{model}}(t) = a + bt \]
Tuning a Model

A Parametrized Model:

\[ v_{model}(t) = a + bt \]

Can we figure out which model – which values of \( a \) and \( b \) – the data most favors?
Probability of Some Observation

Given a particular choice of model parameters, what’s the probability that we’ll observe a given velocity value?

If we assume the observed values are drawn from a Gaussian (Normal) distribution, we can calculate the probability like this:

\[ P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \]

\[ P_i = \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{1}{2} \left( \frac{v_i - v_{model(t_i)}}{\sigma_i} \right)^2} \]
The probability of the collection of data – 3 observations – is just the product of the three individual probabilities.
The probability of the collection of data – $k$ observations – is just the product of the $k$ individual probabilities.
To find the best set of parameters, \(a\) and \(b\), for our model

\[v(t) = a + b \, t\]

we want to maximize the probability, \(P\). This will tell us which values of \(a\) and \(b\) are most likely to produce our observed data.

\[P = \prod_{i=1}^{k} \frac{1}{\sqrt{2\pi}\sigma_i^2} \cdot e^{-\frac{1}{2} \sum \frac{(v_i - v_{\text{model}}(t_i))^2}{\sigma_i^2}}\]

Since the first term doesn't depend on \(a\) and \(b\), the problem reduces to maximizing the second, exponential, term. This is just equivalent to minimizing the sum in the exponent. But wait! This sum is just:

\[\chi^2 \equiv \sum_{i=1}^{k} \frac{(v_i - v_{\text{model}}(t_i))^2}{\sigma_i^2}\]
Minimizing the $\chi^2$

So, to find the best values for $a$ and $b$ in our model, we must find the values of $a$ and $b$ that minimize $\chi^2$. We can find the minimum by looking for the values of $a$ and $b$ that satisfy the following conditions:

$$\frac{\partial \chi^2}{\partial a} = 0$$

and

$$\frac{\partial \chi^2}{\partial b} = 0$$
Minimizing the $\chi^2$

Let's start out by assuming that the standard deviation is the same for all of our data points. In other words, set all the $\sigma_i$ values to the same number, $\sigma$.

Our theory predicts that $v_{\text{model}}(t) = a + bt$, so:

$$\frac{\partial \chi^2}{\partial a} = \frac{\partial}{\partial a} \left[ \frac{1}{\sigma^2} \sum (v_i - a - bt_i)^2 \right]$$

$$= \frac{-2}{\sigma^2} \sum (v_i - a - bt_i) = 0$$

$$\frac{\partial \chi^2}{\partial b} = \frac{\partial}{\partial b} \left[ \frac{1}{\sigma^2} \sum (v_i - a - bt_i)^2 \right]$$

$$= \frac{-2}{\sigma^2} \sum [t_i(v_i - a - bt_i)] = 0$$
Solution for Uniform Sigma:

We can solve the preceding equations for $a$ and $b$:

$$a = \frac{1}{\Delta} \left( \sum t_i^2 \sum v_i - \sum t_i \sum t_i v_i \right)$$

$$b = \frac{1}{\Delta} \left( k \sum t_i v_i - \sum t_i \sum v_i \right)$$

Where, for convenience, we've defined the quantity $\Delta$ as:

$$\Delta = k \sum t_i^2 - (\sum t_i)^2$$

(Notice that $\sigma$ doesn't appear anywhere in these equations, because we've assumed that all the $\sigma_i$ values are the same.)

We can plug numbers into these equations and get the values of $a$ and $b$ that maximize the probability of getting our observed data.
Minimizing the $\chi^2$

Solution for Non-uniform Sigma:

If we don't assume that all the $\sigma_i$ values are the same, we can still work through the algebra and come up with a (slightly more complicated) solution for the best values of $a$ and $b$:

$$a = \frac{1}{\Delta} \left( \sum \frac{t_i^2}{\sigma_i^2} \sum \frac{v_i}{\sigma_i^2} - \sum \frac{t_i}{\sigma_i^2} \sum \frac{t_i v_i}{\sigma_i^2} \right)$$

$$b = \frac{1}{\Delta} \left( \sum \frac{1}{\sigma_i^2} \sum \frac{t_i v_i}{\sigma_i^2} - \sum \frac{t_i}{\sigma_i^2} \sum \frac{v_i}{\sigma_i^2} \right)$$

Where, this time, we've defined the quantity $\Delta$ as:

$$\Delta = \sum \frac{1}{\sigma_i^2} \sum \frac{t_i^2}{\sigma_i^2} - \left( \sum \frac{t_i}{\sigma_i^2} \right)^2$$

It would be tedious to work through these calculations by hand, but it's easy to write a computer program to do them for us.
More Powerful Application: An Arbitrary Theory

We've been talking about fitting a linear function, \( v_{\text{model}}(t) = a + bt \), to a set of data. What if we want to fit a more complex function?

In some cases, we could follow a similar procedure and come up with an analytical solution giving the best-fit values for the parameters in our model.

For complicated functions, we can just try different parameter values (\( a \) and \( b \), in our example), calculating \( \chi^2 \) for each one until we find the minimum. We can do this by brute force, stepping through a grid of values, or we can use root-finding algorithms like Newton's method to help us find the minimum quickly.

We can do this for any model, with any number of parameters.
Fitting with Gnuplot

After 5 iterations the fit converged.
final sum of squares of residuals : 11.2835
rel. change during last iteration : -9.9567e-06

degrees of freedom (FIT_NDF) : 22
rms of residuals (FIT_STDFIT) = sqrt(WSSR/ndf) : 0.716162
variance of residuals (reduced chisquare) = WSSR/ndf : 0.512888

Final set of parameters

$P(x; a, b, c) = \frac{1}{\sqrt{2\pi c^2}} e^{-\frac{(ax-b)^2}{2c^2}}$

Fortunately, programs like gnuplot are there to help us with simple fitting jobs, so we don't have to do it by hand!
We'll experiment with this in lab this week.
Assessing the Quality of a Fit

How Good is Our Best Fit?

So, we've found a set of parameters that minimizes $\chi^2$. Does that mean that our model, when we use these parameters, is a good one? Not necessarily.

A bad model may not fit our data very well even with the best possible choice of parameters. Consider the model below. No matter what values we choose for $a$ and $b$, it still won't fit the data very well.

After we've fit our model to the data, we need to look at the minimum value of $\chi^2$ to see how good our best fit really is.

$$v_{\text{model}}(t) = a \cdot \sin(t) + b$$
Assessing the Quality of a Fit

Consequence: If the number of fit parameters is greater than or equal to the number of data points the $\chi^2$ is undefined.
Assessing the Quality of a Fit

Checking Goodness of Fit:

Number of data points = 25
Number of fitting parameters = 3
\( \chi^2 / k = 0.512888 \)
\( k = 22 \)

\[
P(x; a, b, c) = \frac{1}{\sqrt{2\pi c^2}} e^{\frac{(ax-b)^2}{2c^2}}
\]
So there is a 90% probability that, if the data were consistent with the model (here a Gaussian-like thing with 3 params), the data would have a higher chi2 value.

Too good to be true? Why are the points so close to the model? Did the fit procedure cheat in some way? Are the uncertainties over-estimated?
Fitting is Done EVERYWHERE
The Art of Curve Fitting:

There are many things that may make it difficult or impossible to get a model to fit your data well. Some of them are:

- Using an **incorrect model** to represent data.

- Making a **poor choice of starting parameters**.
  Perhaps they are too far from correct values? Also, some programs have trouble with starting parameters at 0.0.

- Sometimes parameters land on **unphysical values** during the $\chi^2$ minimization process: 1/0, log(-1), 10^{300}, sqrt(negative #), ...

- Sometimes the fitting program has **difficulty settling** into stable values for the parameters (convergence):

  - Maybe you're fitting **too many parameters** at once, while far from the minimum $\chi^2$.
  - Maybe you've chosen a **poor set of model parameters**: high correlations, large differences in scale among parameters (leading to rounding errors), ...
Deviations from the Model

One way of gauging the quality of your fit is by looking at the “fit residuals”. These are the deviations of your data points from the values predicted by your model.

Fit residuals can be measured in terms of number of standard deviations:

\[ f_r = \frac{(data - fit)}{\sigma} \]

- A model, fit to some data.
- \( f_r \) for each data point.
- Distribution of \( f_r \) values
The Pull Distribution

The distribution of the fit residuals is called the “pull distribution”. It helps us gauge the validity of our model.

Properties of the pull distribution:

- **Mean is 0** if the model's shape matches the data well.

- **Width (σ) is 1** if the data points are normally distributed around the model's predictions, consistent with their uncertainties (σ_i).

In this example: no bias, good errors within statistical precision of study.

This implies we're using an appropriate model for this data.

\[
fr_i = \frac{(data_i - fit_i)}{\sigma_i}
\]

Histogram of fit residuals

Mean = 0.045 +/- 0.069
Sigma = 0.94 +/- 0.07
Bias – Is the Prediction In Accord with the Data?

Looking for Bias:

Consider the following two data distributions:
They’ve both resulted in the same fit, with the same $\chi^2$:

Data Set 1

Data Set 2

If large groups of points cluster above or below the best fit, this may indicate a problem with your choice of model.

The $\chi^2$ statistic just adds up the squares of the deviations. It won't notice clusters of points like this.

What is the probability of n adjacent points fluctuating above or below the nominal value at random?
Clusters of Data Above/Below

Probability of Clusters Above/Below the Mean:

\[ p = 50\% \quad p = 50\% \quad p = 50\% \]

Model

If the model is well-matched to the data, the probability of getting three heads in a row (or, equivalently, of three consecutive data points above the predicted values) is:

\[ P(3) = 0.5 \times 0.5 \times 0.5 = 0.125 \]

The probability of \( n \) points in a row above or below the line is:

\[ P(n) = 2^{-n} \]
Clusters of Data Above/Below

In data set 2, a bias of low results on one end and high results on the other end may indicate that we should use a line with a slope.

A good indication for this is if we add a slope and see a significant reduction in $\chi^2$. 
More Testing of Compatibility
For every probability distribution, there's an associated "Cumulative Distribution Function". This is the integral from $-\infty$ to $x$ of the probability distribution.
Properties of the CDF:

At any point $x$, the CDF tells us the probability of observing a random number less than or equal to $x$. As $x$ approaches infinity, the value of the CDF approaches 1. (In other words, it's certain that we'll get a random number less than infinity!)

Gaussian “Cumulative Distribution Function” (CDF).
Example: PDF and CDF

PDF and CDF for a Uniform Distribution:

PDF

\[ \frac{1}{b-a} \]

We can compute the CDF for other probability distributions, too.

At the left is the PDF for a uniform distribution, for \( a < x < b \).

And here is the associated CDF. Notice that there's zero probability of \( x < a \), and a probability of 1 for \( x < b \).
Empirical Distribution Function

If we have a set of data points, we can construct a cumulative distribution function. In this case, it's called an “empirical” CDF (or ECDF), meaning that it's obtained from data, not some mathematical model.

The empirical CDF is just:

\[ ECDF(x) = \frac{\text{number of data points} \leq x}{\text{total number of data points}} \]

An example of an ECDF:
(A grim example.)

Now we're ready to talk about another technique for quantifying how well a fitting function matches our data.
Empirical Distribution Function

• The ECDF is made from “unbinned” data
  – not from a binned histogram
  – use raw measured values

• Do this by:
  1. say you have N values, \( x_i \)
  2. sort the N values in order of increasing value
  3. plot each of the N values with \( x_i \) on the x-axis and \( i/N \) on the y-axis

• Now, compare model’s CDF and the data’s ECDF…
The Kolmogorov-Smirnov (KS) Test:

The KS test consists of plotting the fitting model's CDF and the empirical CDF, then finding the maximum vertical deviation between the two curves.

This maximum deviation, \( D_N \), is called the Kolmogorov-Smirnov statistic.

As with \( \chi^2 \), we can directly relate \( D_N \) to a probability that the model produced the observed data. For example, with 35 data points, the probability of producing the observed data is:

- 20% if \( D_N = 0.180 \)
- 10% if \( D_N = 0.210 \)
- 5% if \( D_N = 0.230 \)
- 1% if \( D_N = 0.270 \)
### Testing Compatibility

<table>
<thead>
<tr>
<th>SAMPLE SIZE (N)</th>
<th>LEVEL OF SIGNIFICANCE FOR D = MAXIMUM [ F₀(X) - Sₙ(X) ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>.900</td>
</tr>
<tr>
<td>2</td>
<td>.684</td>
</tr>
<tr>
<td>3</td>
<td>.565</td>
</tr>
<tr>
<td>4</td>
<td>.494</td>
</tr>
<tr>
<td>5</td>
<td>.448</td>
</tr>
<tr>
<td>6</td>
<td>.410</td>
</tr>
<tr>
<td>7</td>
<td>.381</td>
</tr>
<tr>
<td>8</td>
<td>.358</td>
</tr>
<tr>
<td>9</td>
<td>.339</td>
</tr>
<tr>
<td>10</td>
<td>.322</td>
</tr>
<tr>
<td>11</td>
<td>.307</td>
</tr>
<tr>
<td>12</td>
<td>.295</td>
</tr>
<tr>
<td>13</td>
<td>.284</td>
</tr>
<tr>
<td>14</td>
<td>.274</td>
</tr>
<tr>
<td>15</td>
<td>.266</td>
</tr>
<tr>
<td>16</td>
<td>.258</td>
</tr>
<tr>
<td>17</td>
<td>.250</td>
</tr>
<tr>
<td>18</td>
<td>.244</td>
</tr>
<tr>
<td>19</td>
<td>.237</td>
</tr>
<tr>
<td>20</td>
<td>.231</td>
</tr>
<tr>
<td>25</td>
<td>.210</td>
</tr>
<tr>
<td>30</td>
<td>.190</td>
</tr>
<tr>
<td>OVER 35</td>
<td>1.07</td>
</tr>
</tbody>
</table>
Problem Solving
Problem Solving: Using Logic

- Computer programs are a powerful tool for solving problems!
- But like any problem-solving venture, it is best to be methodical and logical when setting out to find a solution

Simple steps:

1. Define and understand the problem
2. Outline a solution
3. Design an algorithm
4. Convert the algorithm into a program, compile and execute
5. Check the results

(These somewhat differ from what was presented in Reddy but the elements are largely the same)

- These simple steps can be used in any setting, using any language.
- In fact they can be adjusted for use in any setting (even non-programming)
Problem Solving and Logic: Example

• Say you have some data, like these values: 0, 4, 3, 2, 5

• I want you to use a computer program to take this data and report the values in order, high to low
1. Define the problem:
   – Take this data and order it numerically, high to low
Problem Solving and Logic: Example

1. Define the problem:
   – Take this data and order it numerically, high to low

2. Outline a solution: use human language
   – I need to access this data and put it in a format that a program can understand
   – I need to report back the data in numbered order
3. Write an algorithm: use “pseudocode”
   – Input these numbers into a file

0, 4, 3, 2, 5

numbers.dat
3. **Write an algorithm: use “pseudocode”**

- Input these numbers into a file
- Read them in into a list and store them
- an array is best here, for convenience
  - call this the “input array”
Problem Solving and Logic: Example

3. Write an algorithm: use "pseudocode"
   - Input these numbers into a file
   - Read them in into a list and store them
   - An array is best here, for convenience
     • call this the "input array"
   - Create another array, same size, called "output array"

```
0,4,3,2,5
0,4,3,2,5
```

```
numbers.dat
```

```
[ 0, 4, 3, 2, 5 ]
```

```
input array
```

```
[ , , , , ]
```

```
output array
```
3. Write an algorithm: use “pseudocode”
   - Input these numbers into a file
   - Read them in into a list and store them
     - an array is best here, for convenience
       • call this the “input array”
   - Create another array, same size, called “output array”
   - Read in first value from input array, put in first position of output array
3. Write an algorithm: use “pseudocode”
   - Input these numbers into a file
   - Read them in into a list and store them
   - An array is best here, for convenience
     • call this the “input array”
   - Create another array, same size, called “output array”
   - Read in first value from input array, put in first position of output array
   - Read in next value from input array; find value in output array that is smaller than this new value, push all subsequent values down and insert this one
3. Write an algorithm: use “pseudocode”
   - Input these numbers into a file
   - Read them in into a list and store them
   - an array is best here, for convenience
     - call this the “input array”
   - Create another array, same size, called “output array”
   - Read in first value form input array, put in first position of output array
   - Read in next value from input array; find value in output array that is smaller than this new value, push all subsequent values down and insert this one
   - Repeat until all values exhausted

```
numbers.dat

0,4,3,2,5
```

```
input array

[0,4,3,2,5]
```

```
output array

[ , , , , ]
```

```
[0, , , , ]
```

```
[4,0, , , ]
```

```
[4,3,0, , ]
```

```
[5,4,3,2,0]
```
3. Write an algorithm: use "pseudocode"
   - Input these numbers into a file
   - Read them in into a list and store them
     - an array is best here, for convenience
       • call this the "input array"
   - Create another array, same size, called "output array"
   - Read in first value from input array, put in first position of output array
   - Read in next value from input array; find value in output array that is smaller than this new value, push all subsequent values down and insert this one
   - Repeat until all values exhausted
   - Print to screen: ordered: 5,4,3,2,0
4. Write the code...omitted for now.

5. Check results: do they look sensible?
   - this is the most important – and most often overlooked – part of the process
   - "sanity check" – use a simple case in complicated settings

ordered: 5,4,3,2,0
Debugging
Errors: Two Types

**Compile-time versus Run-time Bugs:**

The bugs that afflict our programs can generally be divided into two categories:

- **Compile-time** bugs are caught by the compiler, which will warn us about them. This kind of bug includes all of the various **syntax errors** we've talked about already: variable type mismatches, missing semicolons, and any **typo** that isn't valid C code.

- **Run-time** bugs occur when our program is all perfectly good C code, but it doesn't do what we want it to do. It may produce strange results, or crash in some way. These bugs are due to mistakes in our program's **design**, not typos.
Compile-time Bugs

Compile-time Bugs:

Here's an excerpt from the error messages observed when compiling a complicated piece of code.

This looks bad, but the first error gives us the solution:

src/MemoryMap.cpp:26: parse error before ...

Looking around line 26, the programmer found that line 25 was missing its semicolon. Often one simple fix will clear up many errors.

(And often that simple fix is a semicolon!)

Rule of thumb: When you get a large number of error messages from the compiler, just look at the first one. Errors cascade, so one bad line will corrupt many following lines.
Most Frequent Compile-time Errors

Top seven compiler errors in my experience:

7. Use of variable outside scope of conditional

6. Passing wrong arguments to functions

5. Mismatched () on conditionals

4. Mis-cast variables

3. Missing include statement for header file

2. Missing semi-colon

1. General typo
Run-time Errors

• Two types:
  – innocuous kind which just foul up your data in a subtle fashion
  – catastrophic kind: cause program to cease running immediately
    • aka a “crash!”
    • output form a crash is stored typically in a core dump, a file of human-readable instructions the computer was following when the crash happened – like the black box of an airplane

```c
int i=25;
printf("%f \n",i);
```
Run-time Errors: Examples

**Segmentation faults:**
Your program has tried to access memory that is not allocated to it. That is, it's trying to manipulate data in memory locations it has no privilege to access. This is the OS limiting your access to resources.

Examples:
1) `int i;
   scanf("%d", i); // should have used &i`

2) `FILE *outfile;
   // outfile = fopen("my_file.txt","w");
   fprintf(outfile, "hello\n");  // bad things happen if you try
   // to access an unopened file!`

**Divide by zero:**
Example:
```
int i = 1;
float f = 3.14/(i-1);
```
Divide by zero generates a "floating exception" error for integer arithmetic, but with floating-point arithmetic your program will continue to run, using the special values "inf" or "NaN" ("Not a Number") as the result of the division. This is almost certainly not what you want.
Finding Errors: Tips

• Follow good coding practices, which we are learning
• Read compiler messages – avoid warnings and address errors according to line number
• Make incremental changes to code and compile frequently to keep track of changes
• For run-time errors, insert many printf() statements around the wonky bits to see how your variables are being manipulated. Remove these later for performance sake.
• If all else fails, back out all intricate code, recompile successfully the simple remnants and add back in small piece by piece, until it breaks again
• Last advice: use google, etc., to find if others have encountered your error!
Some Notes on Style:
Good Coding Style
Some Guidelines for Good Coding Style

• There are many opinions on this issue, many choices of convention one can adopt
• Ideas presented here are widely accepted as “best practices”

• These are summarized on the class wiki at the style guide link

• You should start using these conventions
• Why?
  – make your code look more professional
  – easier for me, TA and grader to assess
  – easier for others to examine if it is ever distributed
  – easier for you to debug
Some Guidelines for Good Coding Style

• Variable names:
  – start variable names with a lower case letter and initialize properly:
    ```c
    int area = 0;
    ```
  – use variable case for multi-word names:
    ```c
    int xLength = 0;
    ```
  – make sure the name is descriptive but short esp. for variables of extended scope. Also do not abbreviate:
    • Ex: use beginTime and endTime
      – instead of begT and endT
      – instead of x and y
      – instead of timeOfBeginning and timeOfEnding

• Function names:
  – as for variables, use mixed case and avoid abbreviations:
    ```c
    void calculateDeltaT(int start, int end, *deltaT)
    ```
Some Guidelines for Good Coding Style

• Constant variable names:
  – define these with descriptive names that are all CAPS
    const double TWO_PI = 6.283185;

• Do not use pre-processor definitions:
  do not use #define
  – these are potentially dangerous if used improperly – so just avoid using them in this class and in life
Some Guidelines for Good Coding Style

• Variables of limited scope:
  – These can be short but take care:
    ```
    for (int i=0; i<100; i++) pages[i]=0;
    – i,j,k are commonly used iterators in for loops
    ```

• use prefix “n” for counters:
  ```
  int nPages = 0;
  int nAttempts = 0;
  ```

• Do not distinguish variables just through their capitalization:
  – Do not do this:
    ```
    int itemCounter = 0;
    int ItemCounter = 0;
    ```
Some Guidelines for Good Coding Style

• Indentation of long lines:
  – Avoid code that has very long individual lines. Have each line have less than 80 characters – you will get a feel for this
  – If you have an exceptionally long statement, you can take it to a new line and indent it properly for easy debugging:

```c
printf("Acceleration of %f Kg body due to force of %f Newtons is %f m/s^2:\n",mass, force, force/mass);

makeGraph(xValues, yValues, xRMS, yRMS, outputArray);
```
Some Guidelines for Good Coding Style

• Guidelines for `for(...)` loops and `if()` statements and related:
  – do not make the conditional expression unnecessarily complicated
  – indent the body of the loop/conditional/switch to easily see what is going on:

```c
const int NITEMS=10; // define data size
int items[NITEMS]; // array is NITEMS in size

for (int i=0; i<NITEMS; i++) { // loop over NITEMS
    items[i] = myFunction(i);
    ...
}
```
Some Guidelines for Good Coding Style

• **Location of `#include` statements:**
  – always put these at the beginning of your piece of code

• **Location of function prototypes:**
  – always put these near the beginning of your code, after the `#include` statements but before definition of `main(…)`

• **Definition of functions:**
  – always put these after the definition of `main(…)`
  – indent the body of functions

• **The `main(…)` program:**
  – indent the body of `main(…)`
We’ll pick up from here next time.

See you Thursday at labs!