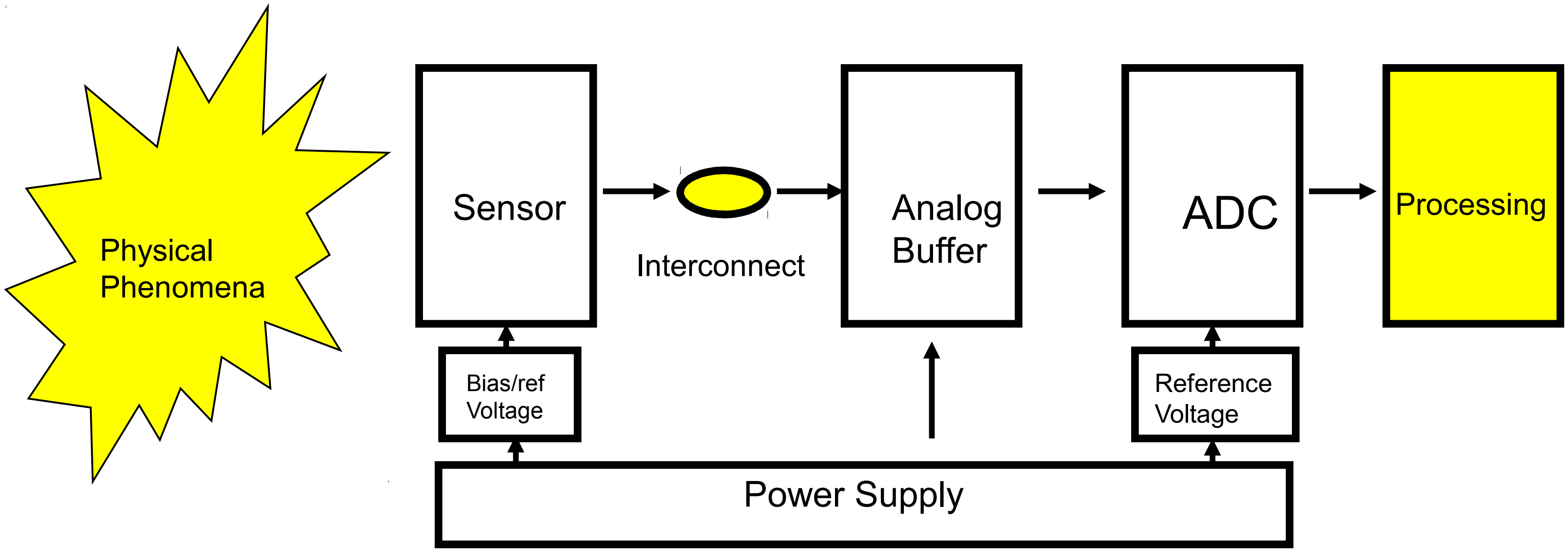
Sensors and time varying noise

- How signal variation gives rise to less than perfect precision
- How to use the PDF and Pe from your measurement data to decide if your measurement precision is good enough.
- How averaging will increase your precision and lower your Pe

Reading:

1) S. Smith, *Statistics, Probability and Noise*
http://www.dspguide.com/CH2.PDF

Looking again at the system



We haven't talked yet about the variability or noise present in the measured phenomena itself, nor the effect of other undesired signals combining in the system, for example from the interconnect, or number round-off noise from the processing.

Processing induces other errors as well.

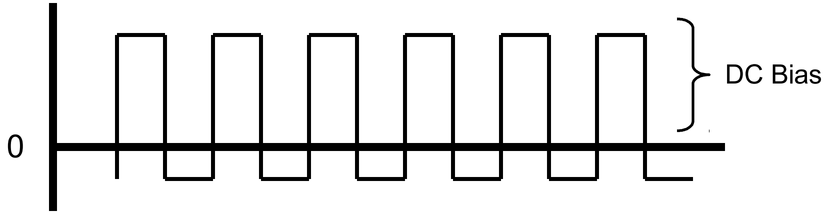
Systematic and Random Noise

- As we saw from our static noise analysis, some noise appears as a non-varying offset or bias. Such noise also can arrive from the signals source as *systematic* noise. Often you can filter it or calibrate it out.

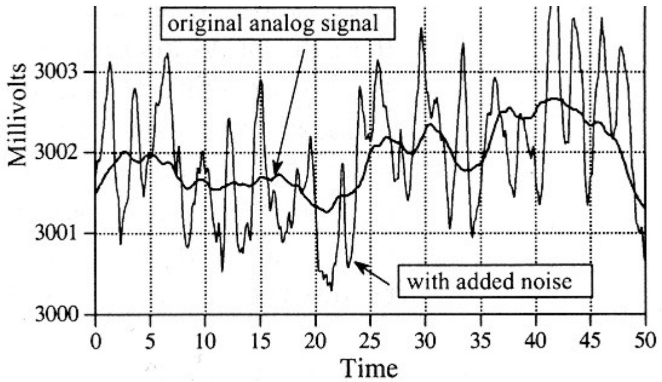
- A signal source that has noise that is randomly varying in time often cannot be easily filtered or calibrated. As we will see, such noise will decrease your measurement precision.

- The task is to determine if the loss of precision is significant, and if so determine if further signal conditioning will improve measurement precision to something acceptable.

Systematic and random noise examples

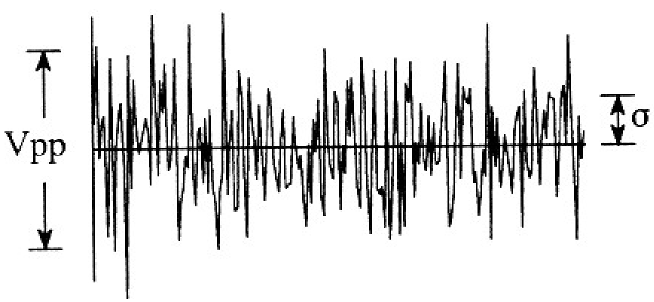


Systematic DC bias.
Calibrate it out.



Higher frequency random noise.
Filter most of it out.

d. Random noise, Vpp ≈ 6-8 σ

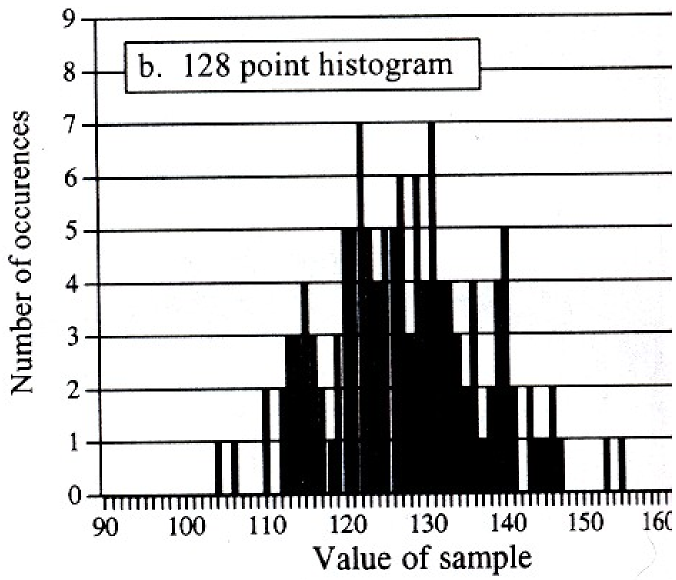


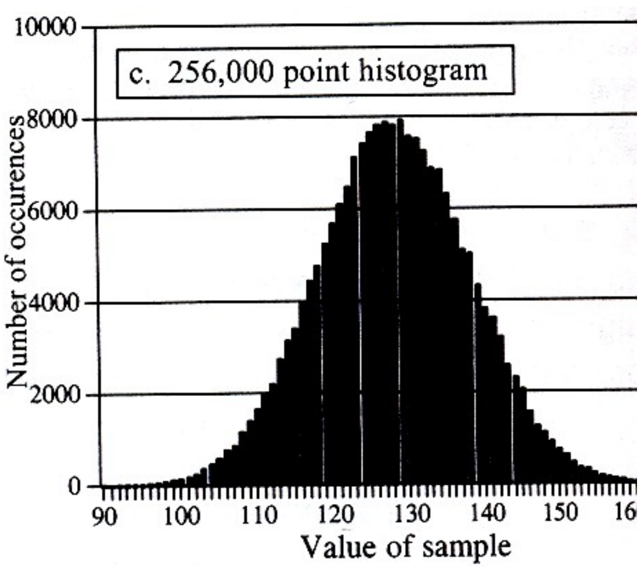
Signal with intrinsic random noise.
Assume we can't easily filter or calibrate the noise out.

Smith, Steven W., Digital Signal Processing, 2003, Newnes, ISBN: 0-750674-44-X

Gaussian noise model

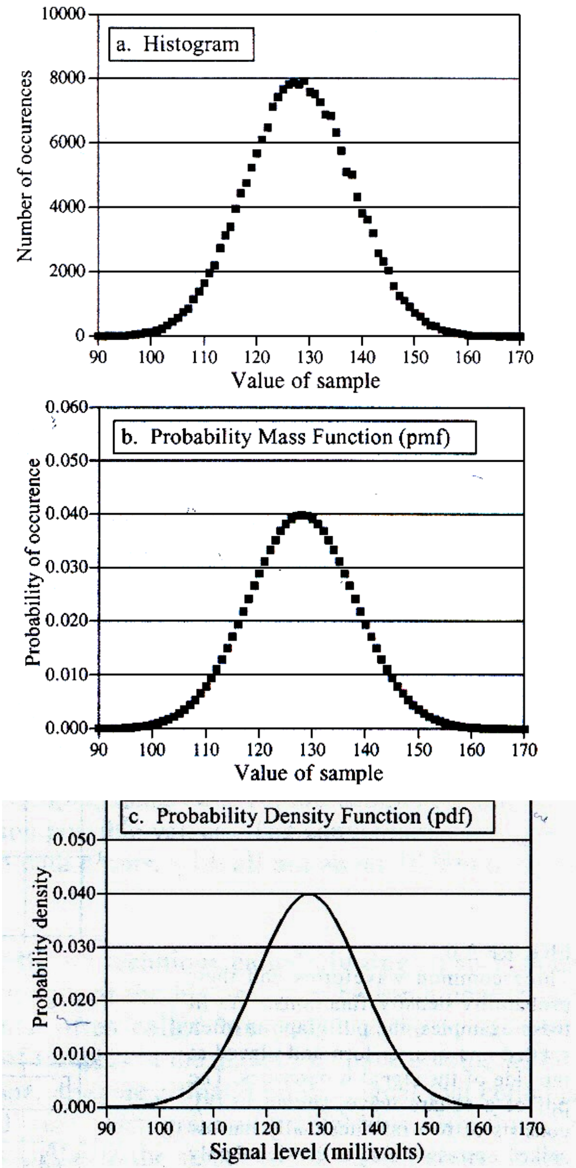
- In some cases we can assume the noise is completely random in nature, and follows a Gaussian or normal distribution.
- It's easy to develop this by looking at the random noise distribution from a series of sensor measurements.
- Repeatedly measure a steady signal with varying noise, and plot a histogram of the raw data. (Accelerometers are good for this.)





Smith, Steven W., Digital Signal Processing, 2003, Newnes, ISBN: 0-750674-44-X

Histogram to a Probability Density Function



The histogram plots the number of occurrences as a function of the measurement reading.

If you now divide the number of occurrences for each measurement reading by the total number of measurements, you get the fraction out of the whole that each measurement occurs. In other words, you get the probability that a particular measurement will occur. Ie, a measurement of 128 will occur about 4% of the time, or 1 in 25 measurements. This curve is called the *Probability Mass Function* (PMF). Note that it is dependent on your measurement resolution.

If you now smooth it out, and treat the probabilities as a continuous range instead of a few discrete values, you end up with the *Probability Density Function* (PDF). It gives you the probability of a signal existing in a *range* of values, ie between 120.4 and 120.5 is:
(120.5 – 120.4) * 0.03 = 0.003 or 0.3% (area under the range)

Smith, Steven W., Digital Signal Processing, 2003, Newnes, ISBN: 0-750674-44-X

Mean and standard deviation

The PDF resulting from the histogram in the example approximates a Gaussian or normal distribution. The central peak of the curve is the mean of the data, and the width of the curve around the mean represents the standard deviation.

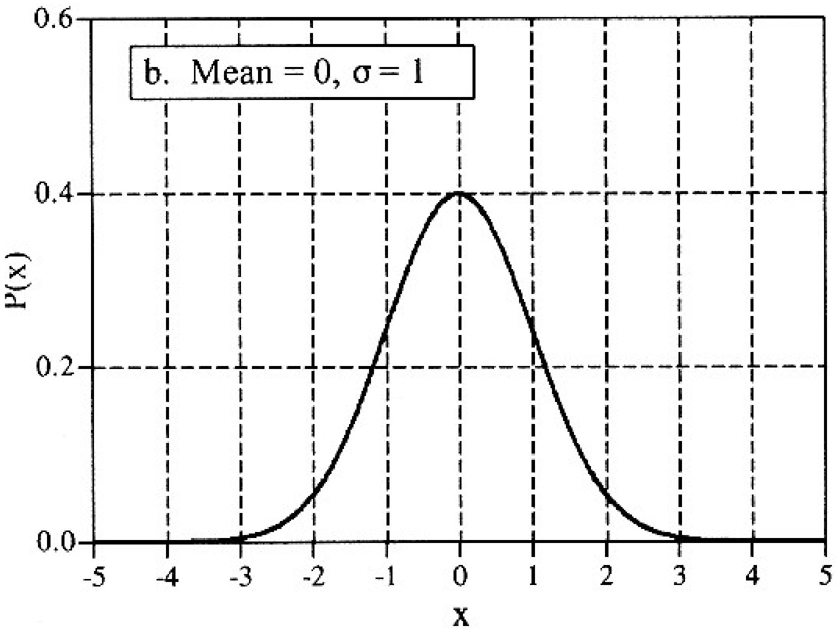
You can directly calculate the mean and standard deviation from the histogram of your data:

$$\mu = \frac{1}{N} \sum_{i=0}^{M-1} i\, H_i$$

$$\sigma^2 = \frac{1}{N-1} \sum_{i=0}^{M-1} (i-\mu)^2 H_i$$

N = total number of points
i = discrete measurement value
H = number of samples with value i
M = total number of possible i values,
ie an 8 bit ADC would give M=256
(M is based on your resolution)

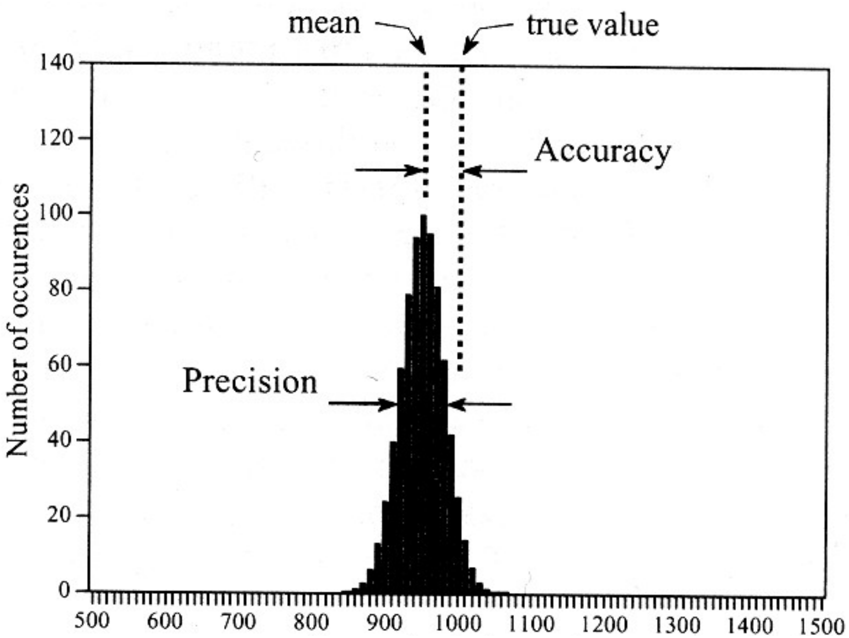
Normalized Gaussian

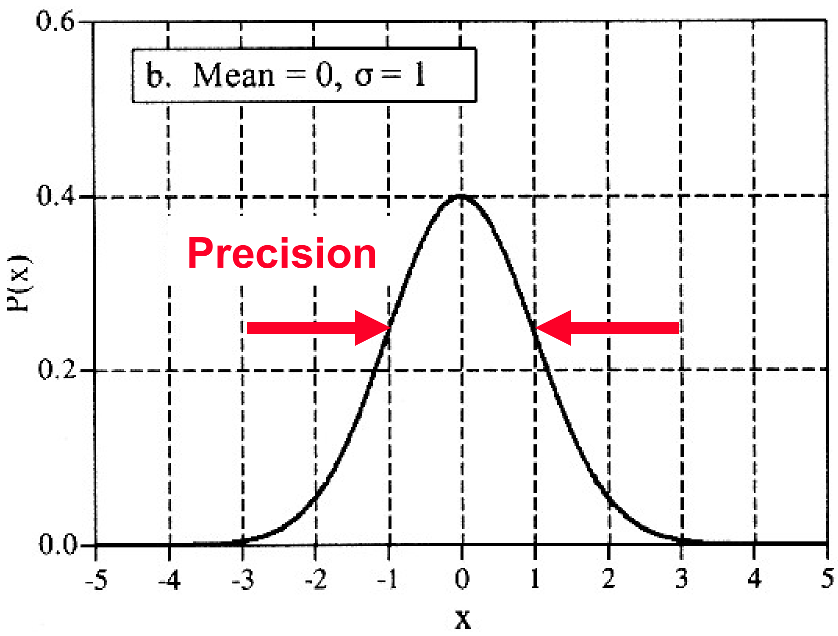


Smith, Steven W., Digital Signal Processing, 2003, Newnes, ISBN: 0-750674-44-X

Precision of the measured data

You can see now the relationship between the standard deviation derived from your measured data, and precision. The larger the standard deviation, the worse the precision.



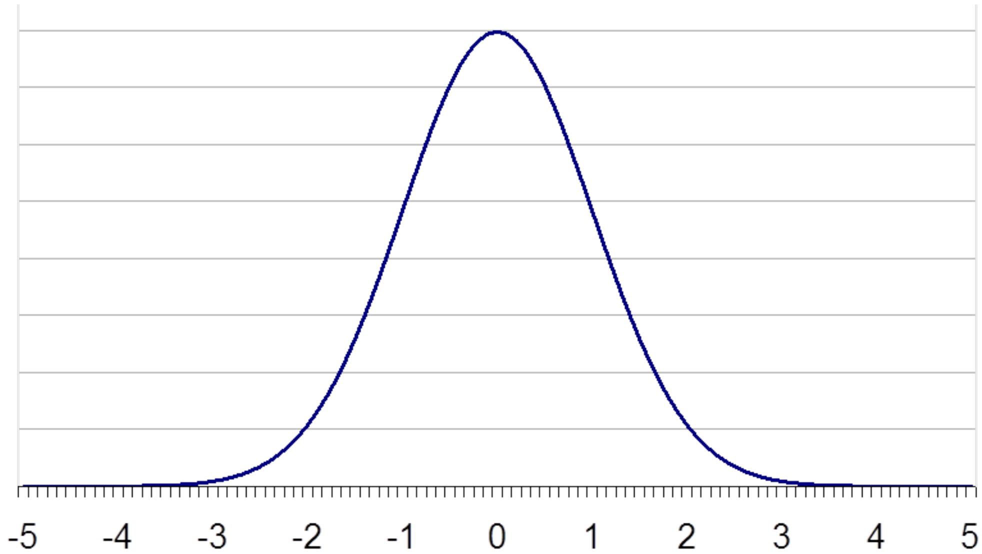


The reason this is important is that it will have an impact on how well you can resolve, or tell the difference, between two adjacent values that you want to read.

Smith, Steven W., Digital Signal Processing, 2003, Newnes, ISBN: 0-750674-44-X

Properties of normal distributions

$$\text{Mean} = 0,\ \ \sigma = 1$$

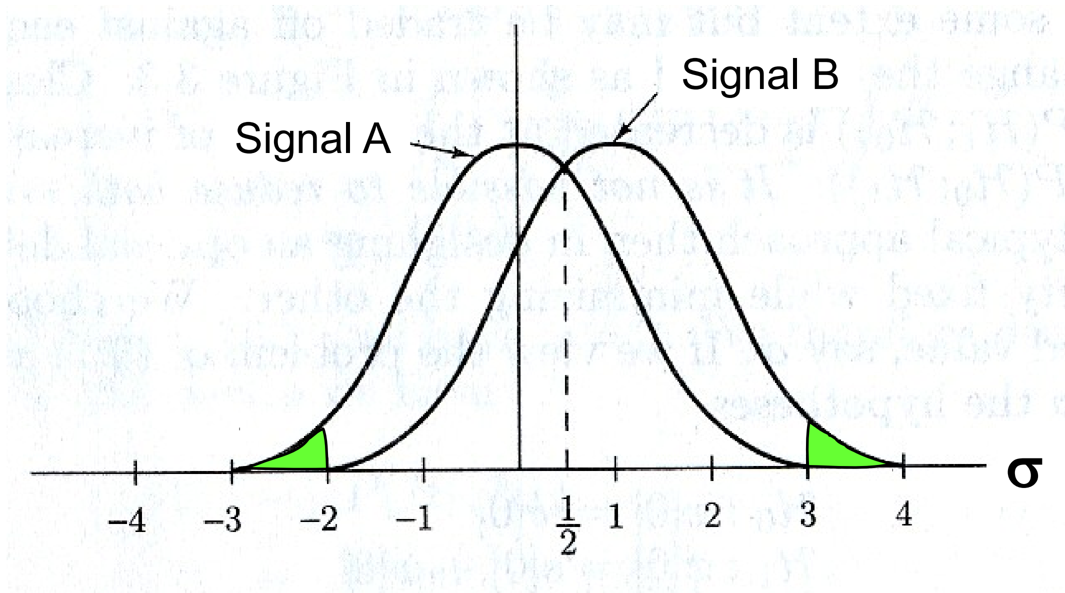


- The probability that a measurement value will exist anywhere under the normal curve is 1.
- The area corresponding to each SD is also consistent in a normal curve. This area is equal to the probability that a measurement will fall within one SD away from the mean or “true value”.

A few more useful Gaussian properties

- The probability of a measurement value being within +-1 SD of the true value is 68%.
- The probability of a measurement value being within +- 2 SD of the true value is 95%.
- The probability of a measurement values being within +- 3 SD of the true value is 99.9%
- This is true of ALL Gaussian distributions!

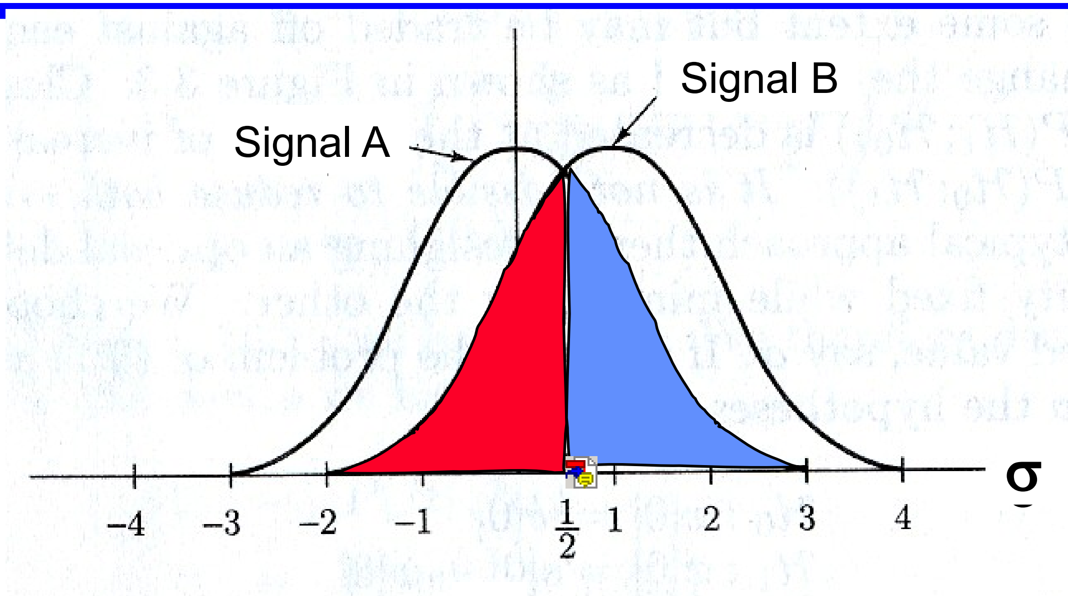
Overlapping readings and error probability



Here are two input signals, A and B, and their distributions. Here is what is interesting when we try to use our measurement system:

1. Our measurement precision won't be perfect due to noise.
2. Values read from the green shaded areas can probably be resolved.
3. Values read elsewhere may not. They could belong to either signal. Which one? For now, let's say there is no other way to decide.
4. What is the probability of error, or Pe?

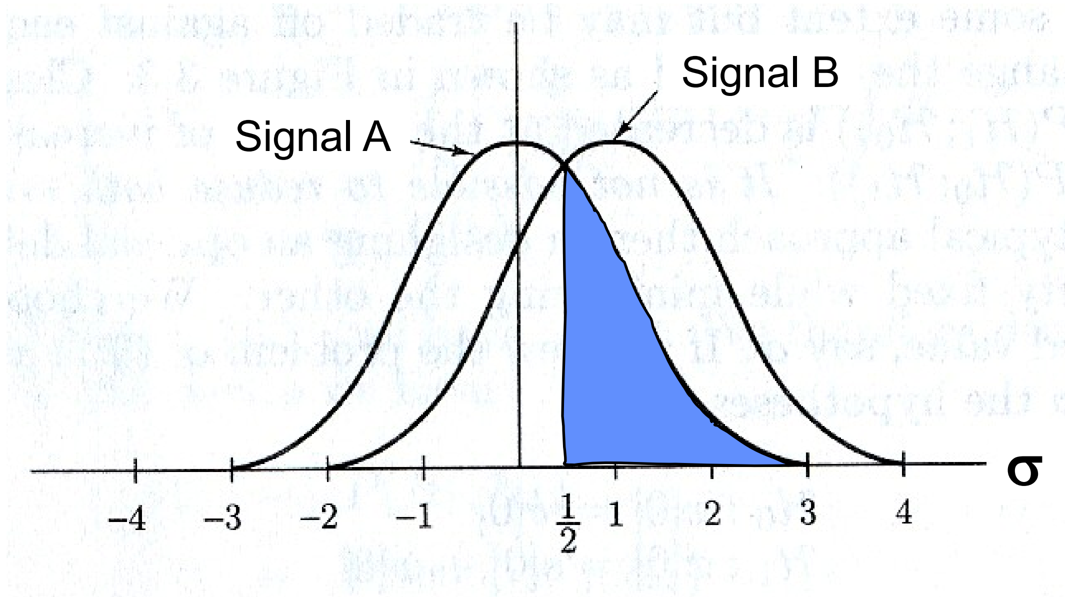
Probability of Error



Equal overlapping areas occur at **σ** = 0.5, so a reasonable thing would be to decide:

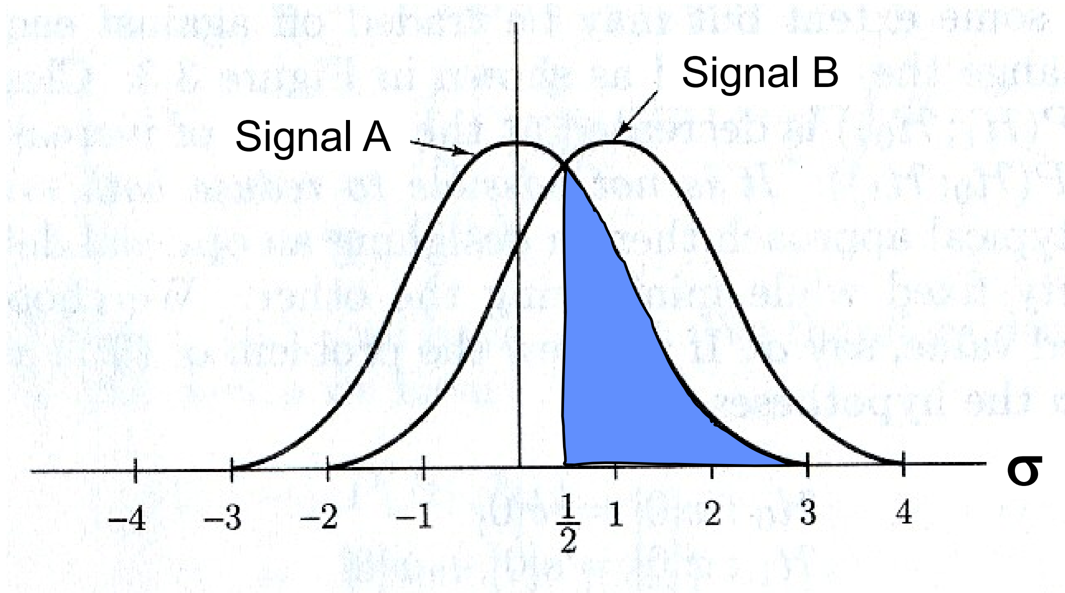
1. If a measurement value lies in the region above this we guess it is due to signal B. The blue area is the overlap where a signal from A could be instead. The probability of a signal A point being in this area is the probability of error.
2. If it lies in the region below this we guess it belongs to signal A. The red area is the overlap where a signal from B could be instead. The probability of signal B being in this area is the probability of error.

Example: Computing Pe



We've decided to make the decision that signal value B will be accepted for any measurement reading that falls on our curve to the right side of $\sigma = 0.5$
What is the probability that we are wrong, and that the data was really generated by signal A, or what is our Pe?

Example: Computing Pe



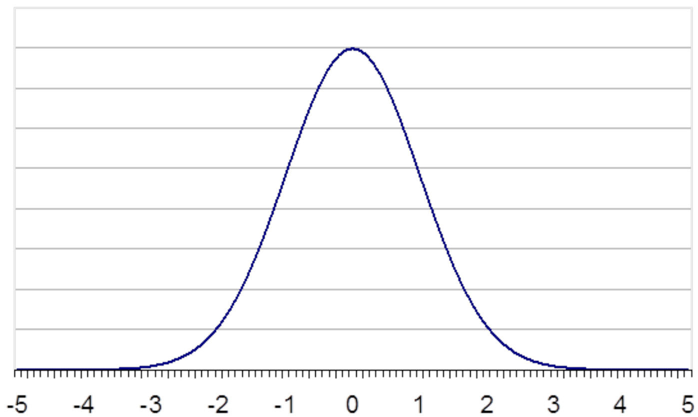
Remember that a PDF tells you the probability of a data reading being in a certain range or *interval*. In our case, the interval we are interested in is $0.5 < \sigma < \infty$

Integrate over the blue area: $$\frac{1}{\sqrt{2\pi}}\int_{0.5}^{\infty} e^{(-u^2/2)}\,du$$

or, take the number off a precomputed table.

Using the Cumulative Distribution Function

$$\text{Mean} = 0,\ \sigma = 1$$

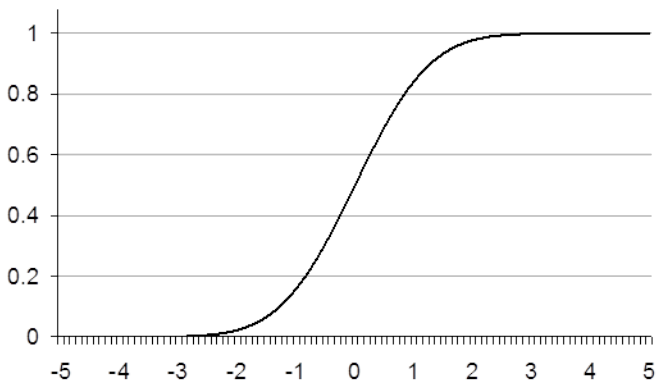


Normalizing

$$F(x) = \frac{1}{\sqrt{2\pi\sigma}} \int_{-\infty}^{\infty} \exp\left[\frac{-(u-\overline{X})^2}{2\sigma^2}\right] du$$

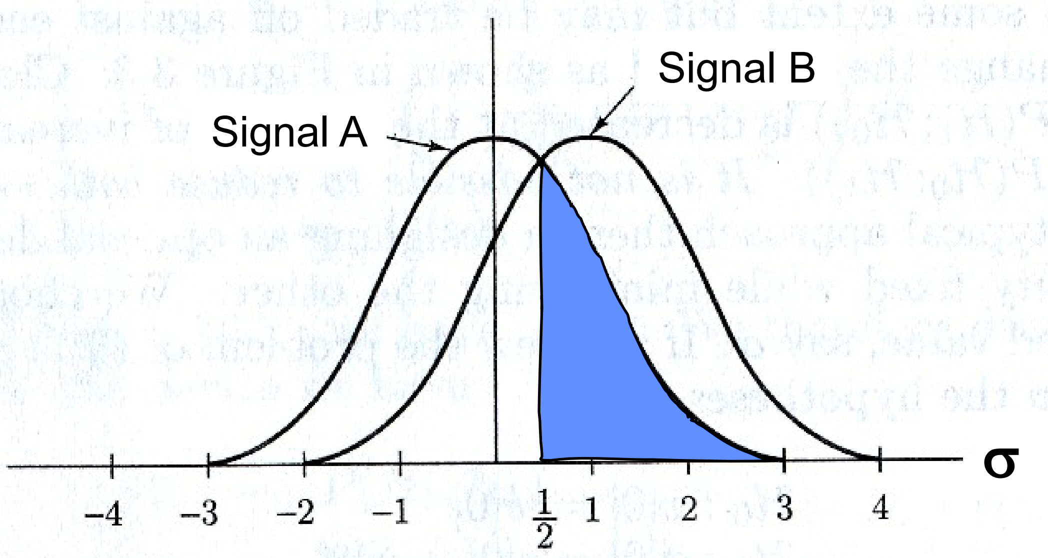
$$\overline{\mathrm{X}} = 0 \text{ and } \sigma = 1$$

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{X} e^{-u^2/2} du$$



This is the CDF of the normalized Gaussian PDF. From this you can take off numbers that give you the probability of being in some range of the PDF.

Example: Computing Pe



$$\begin{aligned}Pe &= \Phi(\infty) - \Phi(0.5)\\ &= 1.0 - 0.6915\\ &= .3085\end{aligned}$$

30.85% of the time we will be in error.
Is there any way to do better?

x	$\Phi(x)$	x	$\Phi(x)$
-3.4	0.0003	0	0.5000
-3.3	0.0005	0.1	0.5398
-3.2	0.0007	0.2	0.5793
-3.1	0.0010	0.3	0.6179
-3	0.0013	0.4	0.6554
-2.9	0.0019	0.5	0.6915
-2.8	0.0026	0.6	0.7257
-2.7	0.0035	0.7	0.7580
-2.6	0.0047	0.8	0.7881
-2.5	0.0062	0.9	0.8159
-2.4	0.0082	1	0.8413
-2.3	0.0107	1.1	0.8643
-2.2	0.0139	1.2	0.8849
-2.1	0.0179	1.3	0.9032
-2	0.0228	1.4	0.9192
-1.9	0.0287	1.5	0.9332
-1.8	0.0359	1.6	0.9452
-1.7	0.0446	1.7	0.9554
-1.6	0.0548	1.8	0.9641
-1.5	0.0668	1.9	0.9713
-1.4	0.0808	2	0.9772
-1.3	0.0968	2.1	0.9821
-1.2	0.1151	2.2	0.9861
-1.1	0.1357	2.3	0.9893
-1	0.1587	2.4	0.9918
-0.9	0.1841	2.5	0.9938
-0.8	0.2119	2.6	0.9953
-0.7	0.2420	2.7	0.9965
-0.6	0.2743	2.8	0.9974
-0.5	0.3085	2.9	0.9981
-0.4	0.3446	3	0.9987
-0.3	0.3821	3.1	0.9990
-0.2	0.4207	3.2	0.9993
-0.1	0.4602	3.3	0.9995
0	0.5000	3.4	0.9997

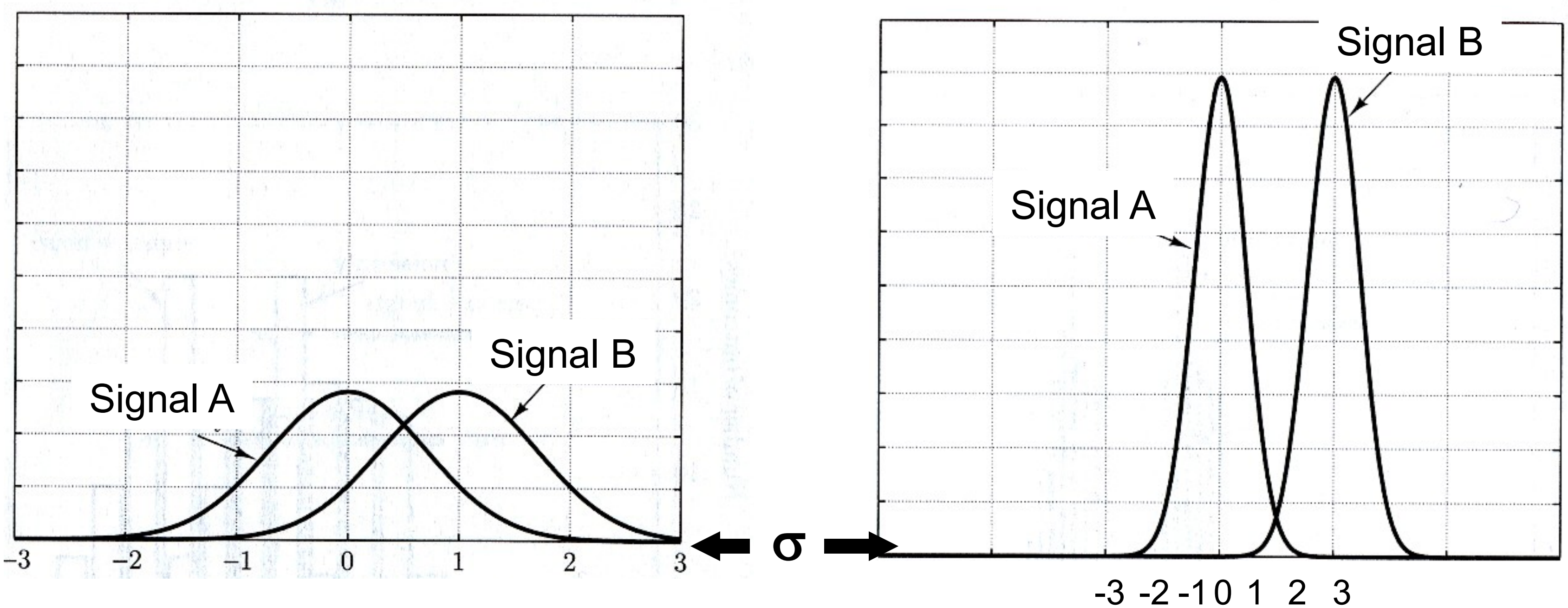
The effect of averaging

- If your noise is truly random, averaging increases your precision because it reduces your variance.
- You can think of your data set that defines your Probability Mass Function, instead of being made up of single data points, now being made up of the averages of some number of data points.
- The population of averages will have less variation around the true mean than the single data points.
- How much less depends upon how many points you average. The greater the number, the less variation you have. In the limit, if you could average an infinite number of data points, your variation would be zero, and your precision would be "perfect".
- This indicates the limitation. How much time do you have to average? It's a space/time tradeoff. The bigger the sample space, the longer the time.

The effect of averaging

Raw Data
X axis are data values

Averaged Data
X axis are data value set averages



In the averaged case, σ^2 has decreased 10x. The two curves now intersect at $\sigma = \sqrt{10} * 0.5 \approx 1.6$

Resulting $Pe = \Phi(\infty) - \Phi(1.6) = 0.054 \text{ or } 5.4\%$ Much Better!

Graphs from: Steven M. Kay, *Fundamentals of Statistical Signal Processing*, 1998, Prentice-Hall, ISBN 0-13-504135-X

Multiple and diverse sensors

- It is also possible to increase precision by using more and diverse sensors.

- Using arrays of similar sensors and averaging across them can increase precision, depending on your application.
- Using multiple heterogeneous sensors and correlating or *fusing* their data can in effect increase your precision. For many complex tasks, such as object identification or various security applications, using multiple sensors may be the only way.
- Using other pieces of data such as time, or history (this is what machine learning tries to do).

Effect of the processor on noise

The processing system can introduce significant computational error:

- The OS can introduce nondeterministic latencies.
- Poor interrupt response.
- Processor clocks used for timers are not accurate.
  - This may be a systematic error on one processor, but across processors and combined with nondeterministic latencies, it becomes random.
- Number formats are not always optimal:
  - Is floating point available? Is it fast enough to be useful?
  - If not, what integer formats are available? Bytes, shorts, ints, longs?
  - Will round off error be significant?
- Does the ALU have resources for:
  - Filtering: Need multiply-adds and sometimes fixed point number formats.
  - Averaging: Need divides and large enough number formats.
- Is there sufficient throughput?
  - Are there enough MIPs to do what you need in the time available?

Summary: To reduce noise

- Determine the source of your random noise
  - Use a better sensor
  - User better components that support the sensor
  - Better design techniques to reduce noise: PC Board layout, analog and digital circuit separation, shielding, bypassing, decoupling, etc

- If your circuit is OK, and you can't use a better sensor:
  - Attempt to filter it out
  - Average it out