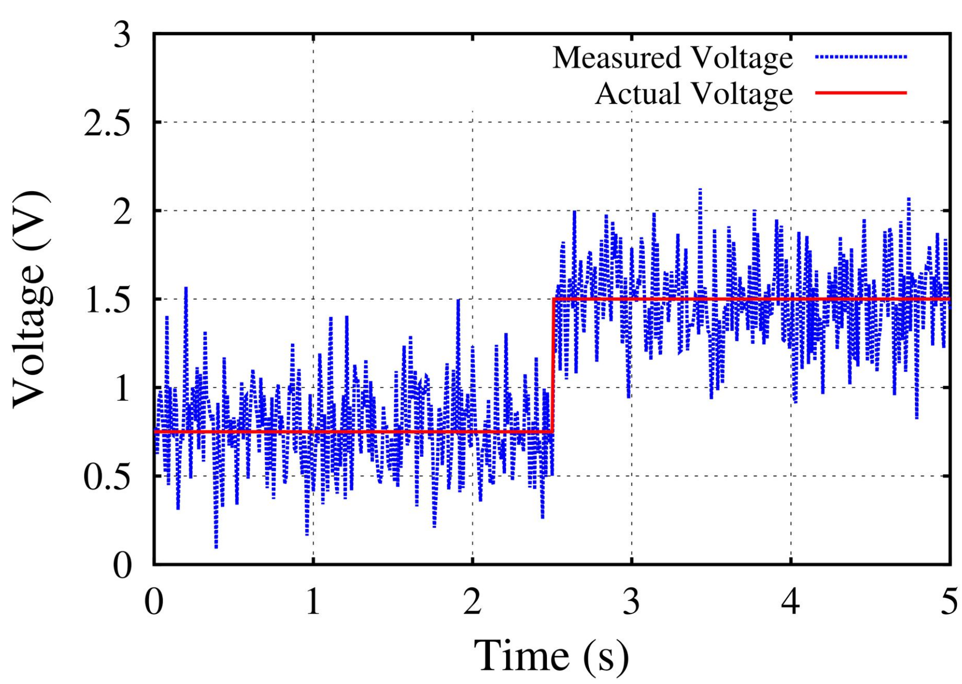
Sensor Processing

Q: What are some possible problems with sensors?
- noise
- update rate
- accuracy and precision

Let's look at how we can deal with sensor noise.

Example: Piecewise constant voltage signal with measurement noise



noise is just random noise (randn in MATLAB)

Q: How can we improve the data?

- average
- ~~if we know when the change occurs, ave. each segment~~ — usually not a good idea
- running average
- moving average

· low pass filter (allow low freq content through, block high freq.)

Running Average:

take the average of the data up to current time ← Easy to do after data is collected

Q: How can we do in "real-time"?

Let's look at the average of a growing series: $m_n = n^{th}$ mean, $y_n = n^{th}$ measurement

$$m_1 = \frac{1}{1} y_1 = y_1$$

$$m_2 = \frac{1}{2}(y_1 + y_2) \longrightarrow \frac{1}{2}(1m_1 + m_2)$$

$$m_3 = \frac{1}{3}(y_1 + y_2 + y_3) \longrightarrow \frac{1}{3}(2m_2 + y_3)$$

$$m_4 = \frac{1}{4}(y_1 + y_2 + y_3 + y_4) \longrightarrow \frac{1}{4}(3m_3 + y_4)$$

$$\vdots \qquad \vdots$$

$$m_{n+1} = \frac{1}{(n+1)}\left[n m_n + y_{n+1}\right]$$

Running Average (cont)

Q: How can we implement this (smartly)?

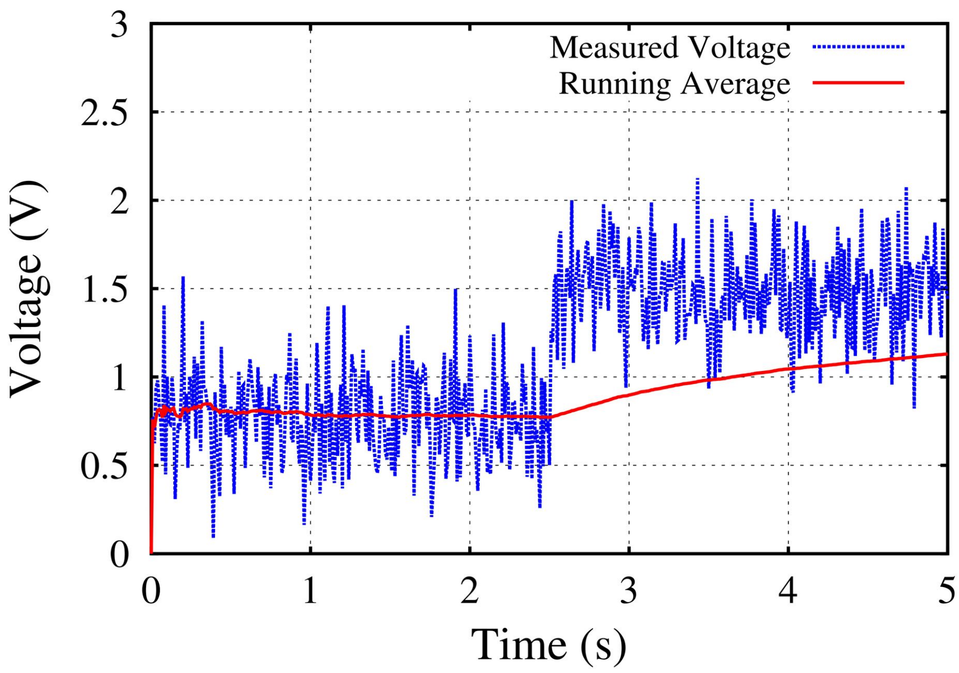
$$m_{n+1} = \frac{1}{(n+1)}\left[\underbrace{n m_n}_{} + y_{n+1}\right]$$

notice that this term is just $\sum^{n} y_i$ so let $S_n = \sum_{i=1}^{n} y_i$

$$m_{n+1} = \frac{1}{(n+1)}\left[S_n + y_{n+1}\right]$$ ← we only need to store:

- n (integer)
- S_n (integer)

} not the entire array



Q: What are the problems with this?

- too heavily weights past values (cannot react to changes quickly)

Q: How can we fix this problem?

$\underbrace{\text{- only average the last N values}}_{\text{moving average}}$

Moving Average:

Take the average of the last N measurements:

$$\text{estimate}_i = \frac{1}{N}\left(y_{i-N} + y_{i-(N-1)} + \ldots + y_i\right)$$ ← Easy to do afterwards (batch processing)

Q: How can we do in "real-time"?

"Strict" moving average

- keep an array of N values
- at each new measurement, remove the oldest and recalculate the average of the array

array[N] - filled with zeros (or other initial guess) initially

counter = 1

loop {

array[counter] = measurement

$\text{estimate} = \frac{\text{sum(array)}}{N}$

counter ++ ← shorthand for counter = counter + 1

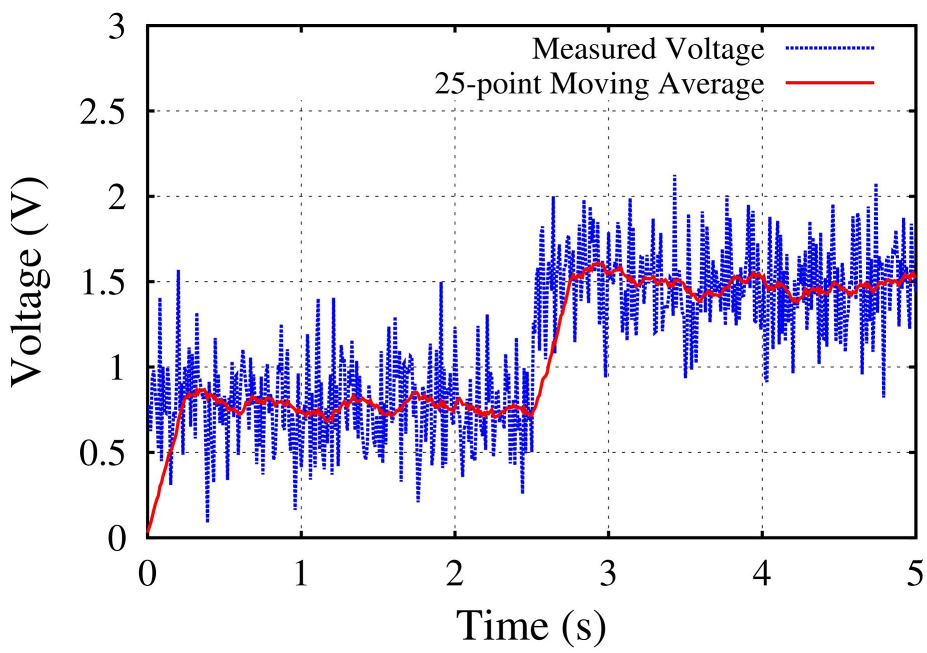
Exponential Moving Average (actually easier to compute)

$$\text{estimate}_i = \frac{1}{N}\Big(\underbrace{y_{i-N} + y_{i-(N-1)} + \ldots + y_i}_{\text{assume all are equal} \;-\; = \text{estimate}}\Big)$$

remove 1 value from old estimate (to represent oldest data point)

$$\text{make new estimate} = \left(1-\frac{1}{N}\right)\overset{\text{old}}{\text{estimate}} + \frac{1}{N}(\text{measurement}) \leftarrow$$ very similar to "strict" N-point average

Q: What's different?
weighting of old values

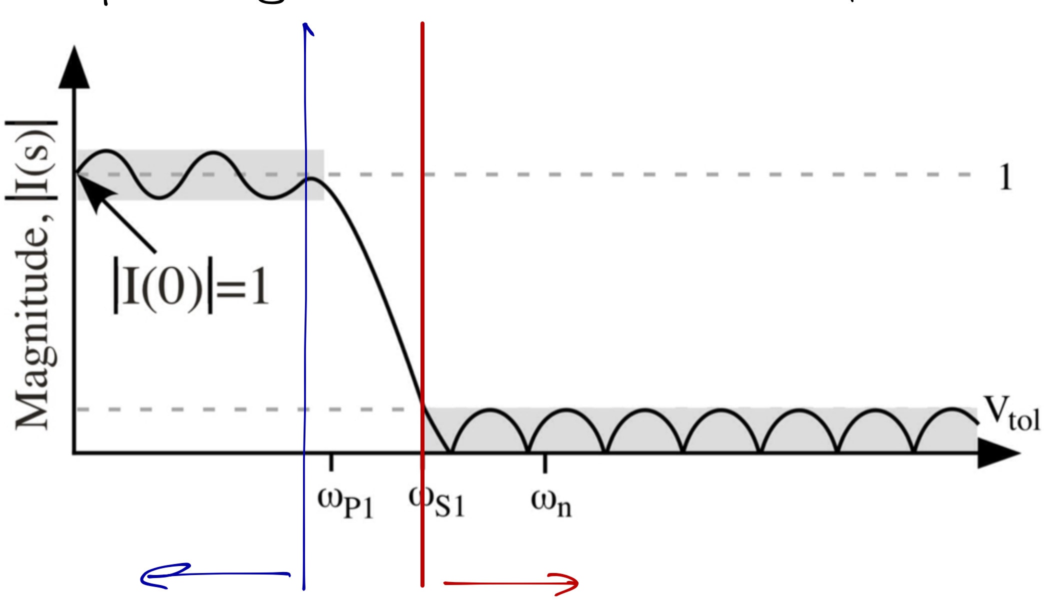


Q: Problems?

- still slow to react to changes
  (this may be good in some circumstances)
- doesn't use any knowledge of signal
  (we generally have some idea of what the signal will be)

Lowpass filter

only allow signals with freq. less than ω_c to pass



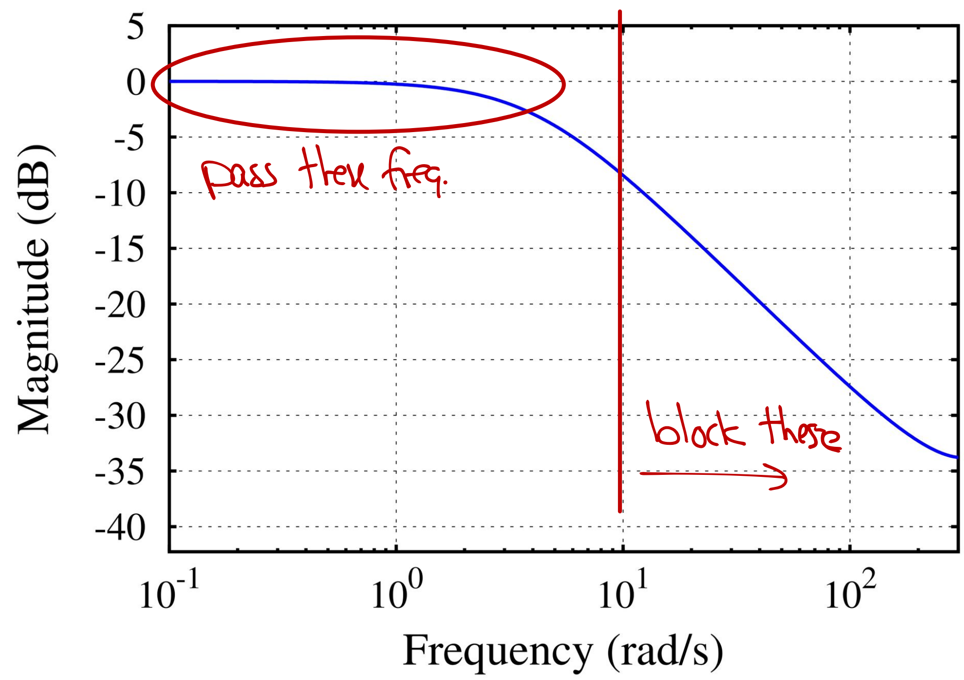
let these freq pass

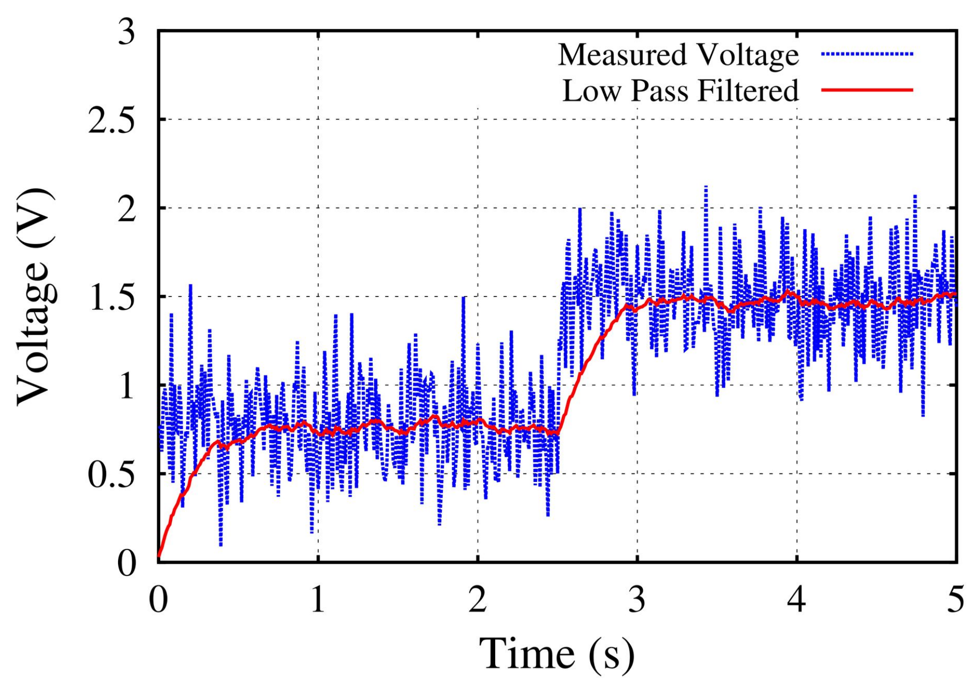
block these

Noise is typically higher freq. than the data.

Lowpass filter (cont)

We won't go into the design of these. Most software packages have them built-in, or make them easy to create. For our example:





Q: Problems?

- (by design) does not react to rapid changes
- uses some info about system, but we can do better.