Kalman Filter

A man with two watches is never sure of the time.

Disclaimer

- I am a novice to Kalman Filtering
  - Skye Sweeney
  - Skye@FLL-Freak.com
- I have not slept at a Holiday Inn Express
- I may have details wrong
  - But I am pretty sure the Big Idea is right!

History

- Initial work done by Rudolf Kalman in 1960
- Kalman met Stanley Schmidt at NASA
- Together they created an algorithm that allowed us to navigate a man to the moon
  - Tie-in with the 50th anniversary of Apollo 11!

Use

- The filter is primarily used to make good estimates from noisy data
- Apollo’s gyros and accelerometers were pretty crude by today’s standards
- Ground radar was also pretty rough
- And yet we greased a landing 238,000 miles away with computers about as powerful as an Arduino Uno!

Other Uses

- Tracking objects with radar
- Robot Navigation
- Economics
- Feature tracking in computer vision
- Internal GPS position estimation
  - uBlox dynamic models

Needed Skills to Implement

- **Basic math (+, -, *, /, …)**
- Weighted averaging
- Basic concepts of statistics
  - Standard deviation
- An understanding of your sensors
  - Know Your Sensor™
- An understanding of the system

Needed Skills to Implement

- Basic math (+, -, *, /, ...)
- **Weighted averaging**
- Basic concepts of statistics
  - Standard deviation
- An understanding of your sensors
  - Know Your Sensor™
- An understanding of the system

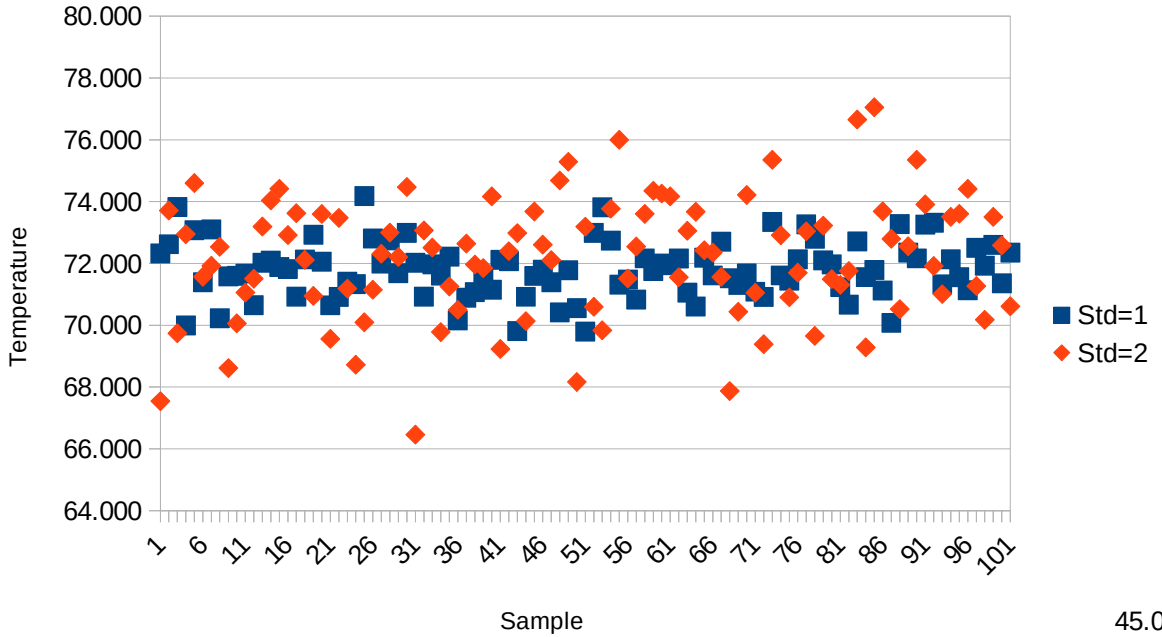
Weighted Average

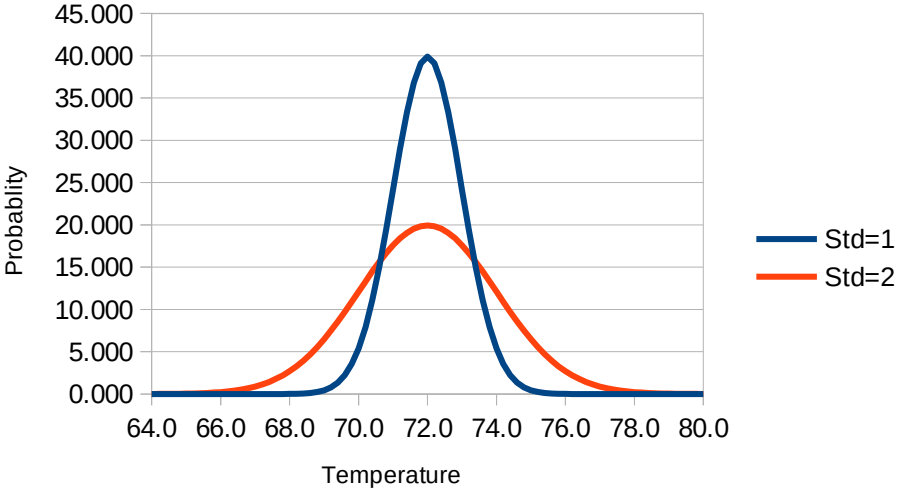
- Avg = ½ * V1 + ½ * V2 (Equal weight avg)
- Avg = ¼ * V1 + ¾ * V2 (Unequal weight)
- General form
  - Avg = K * V1 + (1.0 – K) * V2
    - K must be between 0 and 1.0
    - If K is 1 then V2 has no effect
    - If K is 0 then V1 has no effect

Needed Skills

- Basic math (+, -, *, /, ...)
- Weighted averaging
- **Basic concepts of statistics**
  - Standard deviation
- An understanding of your sensors
  - My mantra
- An understanding of the system

Statistics


80.000
78.000
76.000
74.000
72.000
70.000
68.000
66.000
64.000
Temperature
1
6
11
16
21
26
31
36
41
46
51
56
61
66
71
76
81
86
91
96
101
Sample
Std=1
Std=2


45.000
40.000
35.000
30.000
25.000
20.000
15.000
10.000
5.000
0.000
Probablity
64.0
66.0
68.0
70.0
72.0
74.0
76.0
78.0
80.0
Temperature
Std=1
Std=2

Needed Skills

- Basic math (+, -, *, /, ...)
- Weighted averaging
- Basic concepts of statistics
  - Standard deviation
- **An understanding of your sensors**
  - **My mantra**
- An understanding of the system

Needed Skills

- Basic math (+, -, *, /, ...)
- Weighted averaging
- Basic concepts of statistics
  - Standard deviation
- An understanding of your sensors
  - My mantra
- **An understanding of the system**

Equations

- $x_{t|t-1} = F_t x_{t-1|t-1} + B_t u_t$
- $P_{t|t-1} = F_t P_{t-1|t-1} F^T_t + Q_t$
- $x_{t|t} = x_{t|t-1} + K_t(y_t - H_t x_{t|t-1})$
- $K_t = P_{t|t-1} H^T_t (H_t P_{t|t-1} H^T_t + R_t)^{-1}$
- $P_{t|t} = (I - K_t H_t) P_{t|t-1}$

Equations

- $x_{t|t-1} = F_t x_{t-1|t-1} + B_t u_t$
- $P_{t|t-1} = F_t P_{t-1|t-1} F^T_t + Q_t$
- $x_{t|t} = x_{t|t-1} + K_t(y_t - H_t x_{t|t-1})$
- $K_t = P_{t|t-1} H^T_t (H_t P_{t|t-1} H^T_t + R_t)^{-1}$
- $P_{t|t} = (I - K_t H_t) P_{t|t-1}$
- **So obvious they speak for themselves.**

Equations

- $x_{t|t-1} = F_t x_{t-1|t-1} + B_t u_t$
- $P_{t|t-1} = F_t P_{t-1|t-1} F^T_t + Q_t$
- $x_{t|t} = x_{t|t-1} + K_t(y_t - H_t x_{t|t-1})$
- $K_t = P_{t|t-1} H^T_t (H_t P_{t|t-1} H^T_t + R_t)^{-1}$
- $P_{t|t} = (I - K_t H_t) P_{t|t-1}$
- **So obvious they speak for themselves.**
- **NOT!**

Frustration!

- Books do not do a simple introduction
- They tend to be at the Masters level
- They expect mastery in Control Theory and Linear Algebra

- The filter is actually pretty simple

An Ultra Simple Example

- We want to estimate the temperature inside an oven. (Happens to be 72 degrees)
- The oven is off, and has been for days
- Our BangGood thermometer is very noisy
  - Probably should by a new one
- The readings are random (Gaussian)
- It gives readings within 2 degrees 68% of the time
  - Standard deviation is 2.0 degree

Iterative

- Kalman filter is an iterative algorithm
- Every so often you do the same calculations
- Normally they would be done each time a new measurement is made
- The inputs to the filter are:
  - The new measurement
  - Your last estimate
  - Some numbers related to the error in the system
- The primary output is the new estimate

The Second Equation First

- $Est_{new} = K * Mea + (1-K) * Est_{last}$
- If K is 1.0, we trust the measurement
- If K is 0.0, we trust our last estimate
- For values in between, we take part of one and part of the other.
- Alternate form
  - $Est_{new} = Est_{last} + K * (Mea - Est_{last})$

What Value is K?

- K will not be a constant!
- We will compute K with some knowledge of how good we think our measurement and our previous estimate are

First Equation (Kalman Gain)

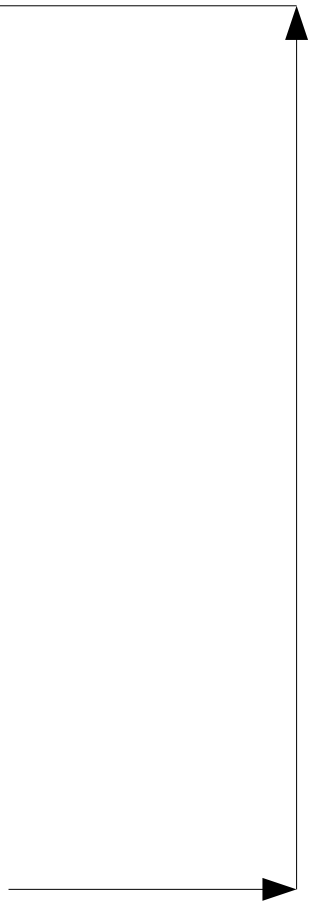
- $K = \varepsilon_{estimate}/(\varepsilon_{estimate} + \varepsilon_{measuremt})$
- $\varepsilon_{measuremt}$ is the amount of error in our measurement.
  - For best results this should be the standard deviation of the error.
- For our oven, $\varepsilon_{measuremt} = 2.0$
- But what about $\varepsilon_{estimate}$?
  - How do we know what that is?

The Third Equation

- $\varepsilon_{estimate} = (1.0 - K) * \varepsilon_{previous\ estimate}$
- Start with a wild guess of $\varepsilon_{estimate.}$
  - For our example, we could assume:
    - The temperature is 50.0
    - The standard deviation is 60.0
    - That would allow a range of -80 to 200 degrees.
  - So we start with $\varepsilon_{estimate} = 60.0$

Summary

- Guess $\varepsilon_{estimate}$ Est_{last}
- Get a new measurement
  - Mea, $\varepsilon_{measuremt}$ = from sensor
- Compute K
  - $K = \varepsilon_{estimate}/(\varepsilon_{estimate} + \varepsilon_{measuremt})$
- Compute new estimate
  - $Est_{new} = K * Mea + (1\text{-}K) * Est_{last}$
- Compute new error in estimate
  - $\varepsilon_{estimate} = (1.0 - K) * \varepsilon_{previous\ estimate}$



Filter in operation (pass 1)

- Compute K
  - $K = \varepsilon_{estimate}/(\varepsilon_{estimate} + \varepsilon_{measuremt})$
  - $K = 60.0/(60.0 + 2.0) = 0.968$
    - (Ignore the estimate, use the measurement!)
- Compute new estimate
  - $Est_{new} = K * Mea + (1\text{-}K) * Est_{last}$
  - $Est_{new} = 0.968 * \mathbf{70.743} + (1\text{-}0.968) * 50.0$
  - $Est_{new} = 70.074$

Filter in operation (pass 1)

- Compute new error in estimate
  - $\varepsilon_{estimate} = (1.0 - K) * \varepsilon_{previous\ estimate}$
  - $\varepsilon_{estimate} = (1.0 - 0.968) * 60.0$
  - $\varepsilon_{estimate} = 1.935$
    - Wow! This is better than the measurement!

Filter in operation (pass 2)

- Compute K
  - $K = \varepsilon_{estimate}/(\varepsilon_{estimate} + \varepsilon_{measuremt})$
  - $K = 1.935/(1.935 + 2.0) = 0.492$
    - (Close to 50/50!)
- Compute new estimate
  - $Est_{new} = K * Mea + (1\text{-}K) * Est_{last}$
  - $Est_{new} = 0.492 * \mathbf{73.694} + (1\text{-}0.492) * 50.0$
  - $Est_{new} = 71.855$

Filter in operation (pass 2)

- Compute new Error in estimate
  - $\varepsilon_{estimate} = (1.0 - K) * \varepsilon_{previous\ estimate}$
  - $\varepsilon_{estimate} = (1.0 - 0.492) * 1.935$
  - $\varepsilon_{estimate} = 0.984$

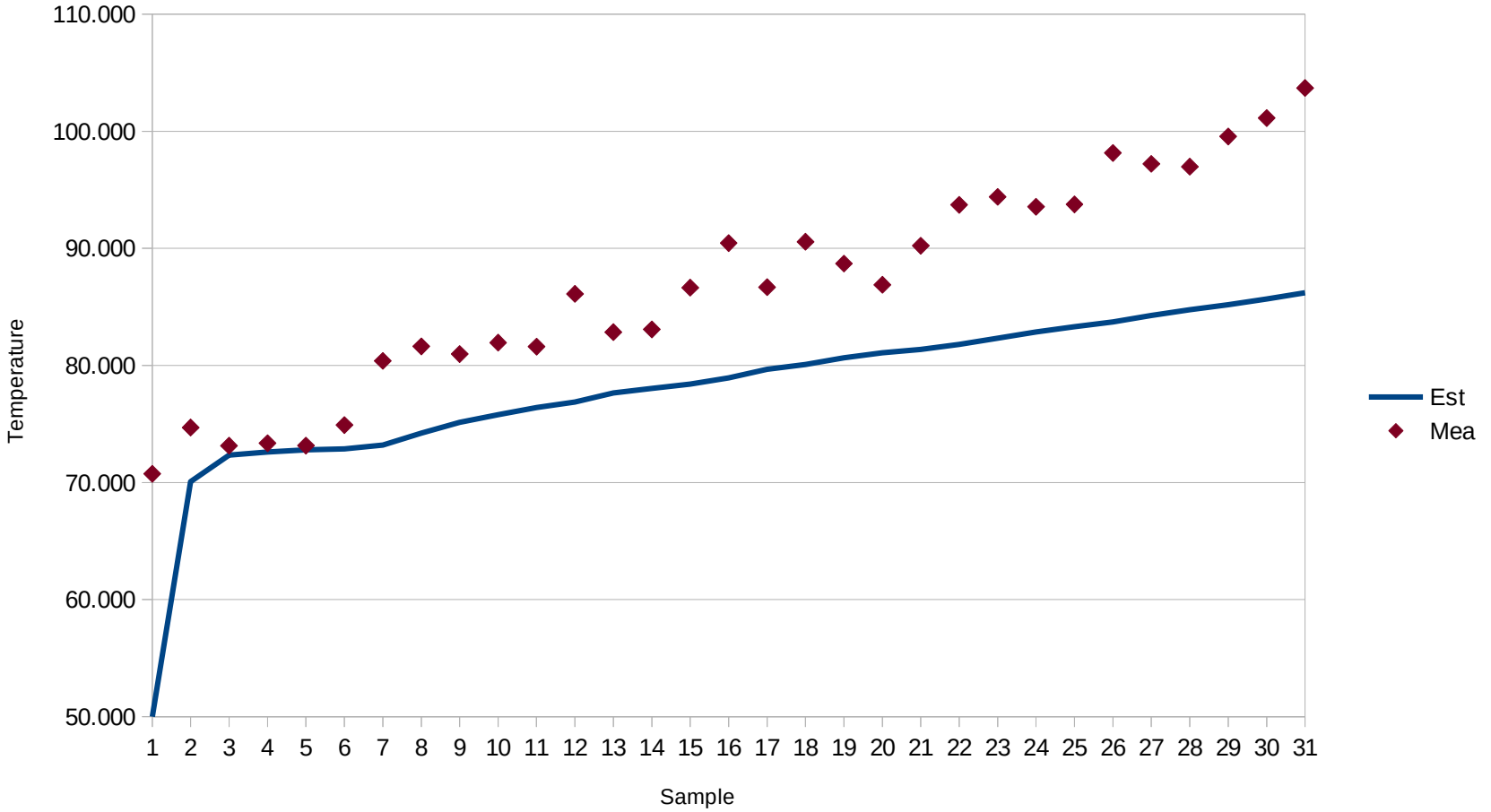
A Nice Plot


Oven Example
80.000
75.000
70.000
65.000
60.000
55.000
50.000
Temperature
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Sample

Lets preheat the oven!

- Every second the oven gets one degree hotter

Bad Kalman!


110.000
100.000
90.000
80.000
70.000
60.000
50.000
Temperature
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Sample
Est
Mea

What Happened?

- The three equations works great for systems at rest
- For dynamic system these are not enough
- Each time you take a new measurement to average in, the estimate is out of date
- You are adding in only a fraction of the new temperature to the estimate you made when the oven was colder
- You are always playing catch-up

The Fourth Equation

- This equation is not in any book
- It depends on **your** system
- You must make a guess how your system should behave given the current state
- This is the forecasting/prediction step

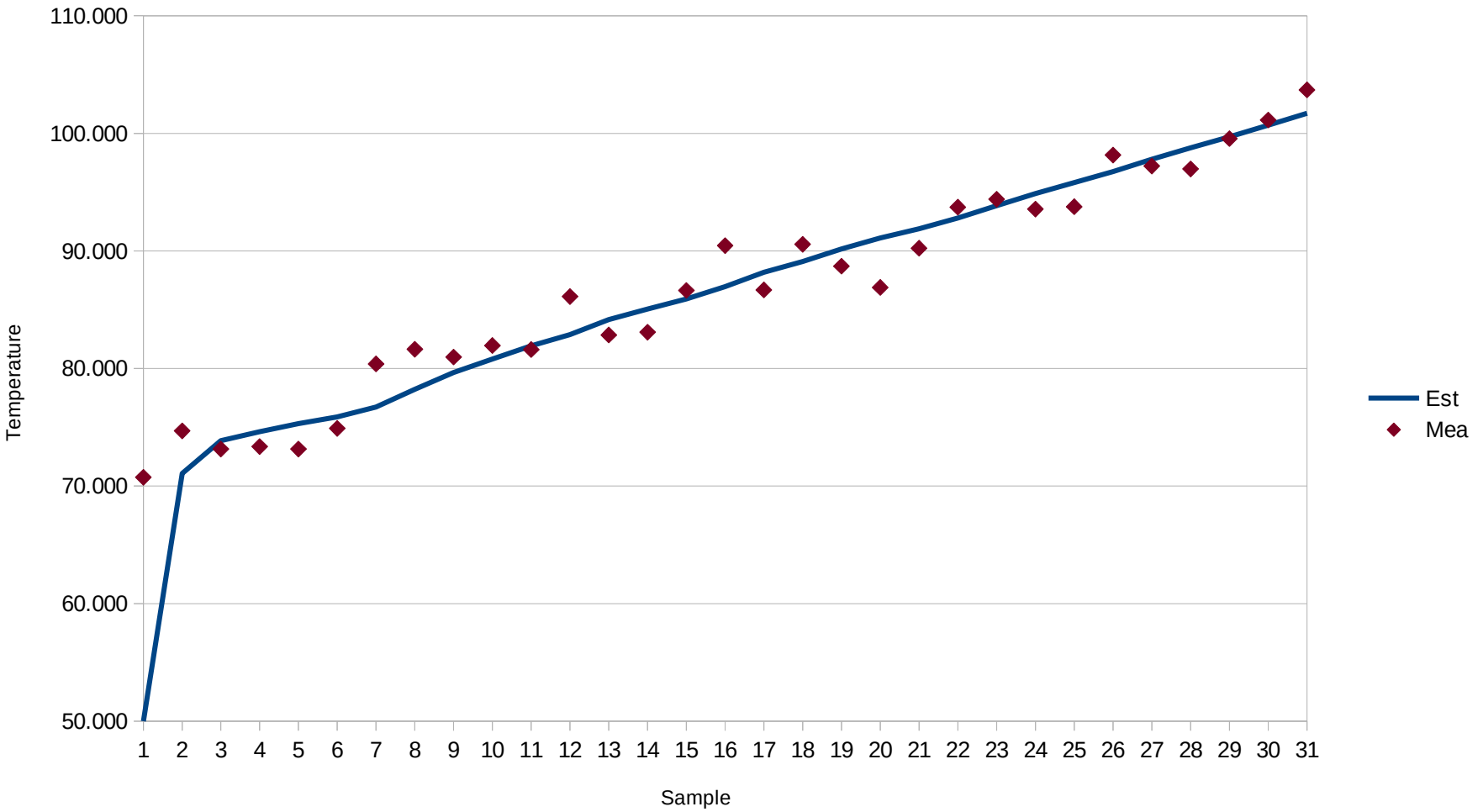
Forecasting Examples

- Oven preheat
  - $Est_{new} = Est_{Computed} + TempRate * DeltaTime$
- Position of train on tracks
  - $Est_{new} = Est_{Computed} + Speed * DeltaTime$
- Position of ball falling
  - $Est_{new} = Est_{Computed} + ½ * gravity * DeltaTime^2$
- Kinematic model of a robot
  - …

Oven Preheat Example

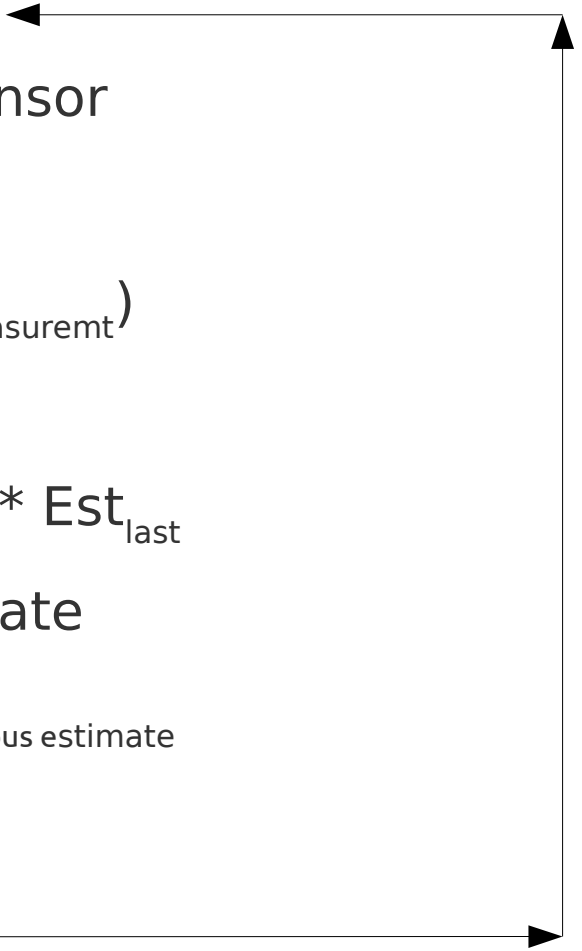
- Based on our knowledge of the oven the temperature increases about 1 degree every time we take a new measurement.
- So after we compute a new estimate, we can forecast what the temperature is likely to be when we take the new measurement.
- $Est_{forecast} = Est_{computed} + 1.0$

Oven Preheat Example



Summary

- Guess $\varepsilon_{estimate}$ Est_{last}
- Get a new measurement
  - Mea, $\varepsilon_{measuremt}$ = from sensor
- Compute K
  - $K = \varepsilon_{estimate}/(\varepsilon_{estimate} + \varepsilon_{measuremt})$
- Compute new estimate
  - $Est_{new} = K * Mea + (1-K) * Est_{last}$
- Compute new error in estimate
  - $\varepsilon_{estimate} = (1.0 - K) * \varepsilon_{previous\ estimate}$
- Forecast the estimate

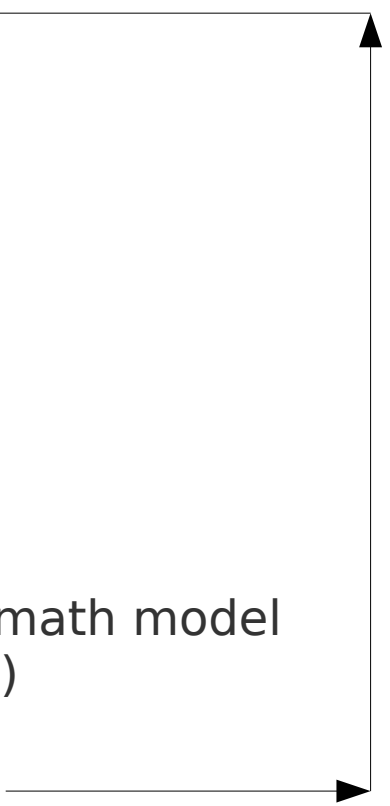


The Fifth Equation

- $\varepsilon_{estimate} = (1.0 - K) * \varepsilon_{previous\ estimate}$
- Since K must be between 0 and 1, $\varepsilon_{estimate}$ can only get smaller
- For a dynamic case, this is not the case
  - We introduce uncertainty in the estimate in our forecast model
  - Therefore we need to bump up the error
    - $\varepsilon_{estimate} = f(\varepsilon_{previous\ estimate}, \text{math model})$
  - How much we do this is based on our model
  - Good luck!

Final Equations

- Guess $\varepsilon_{estimate}$ Est_{last}
- Get a new measurement
  - Mea, $\varepsilon_{measuremt}$ = from sensor
- Compute K
  - $K = \varepsilon_{estimate}/(\varepsilon_{estimate} + \varepsilon_{measuremt})$
- Compute new estimate
  - $Est_{new} = K * Mea + (1-K) * Est_{last}$
- Compute new error in estimate
  - $\varepsilon_{estimate} = (1.0 - K) * \varepsilon_{previous\ estimate}$
- Forecast the estimate
  - $Est_{new} = f(Est_{new}$, math model)
- Compute new error in estimate due to math model
  - $\varepsilon_{estimate} = f(\varepsilon_{previous\ estimate}$, math model)



Original Equations

- $x_{t|t-1} = F_t x_{t-1|t-1} + B_t u_t$ Forecast
- $P_{t|t-1} = F_t P_{t-1|t-1} F^T_t + Q_t$ Error in forecast
- $x_{t|t} = x_{t|t-1} + K_t(y_t - H_t x_{t|t-1})$ New Estimate
- $K_t = P_{t|t-1} H^T_t (H_t P_{t|t-1} H^T_t + R_t)^{-1}$ Kalman gain
- $P_{t|t} = (I - K_t H_t) P_{t|t-1}$ Error in Estimate
- **So why the complexity?**

Complexity

- Because so far we have only talked about simple 1D problems
  - Temperature of an oven
  - Train on tracks
  - Ball falling
- What about tracking an Apollo Command Module on its way to the moon?
  - This is a 6 degree of freedom problem
    - Pitch, roll, yaw, X, Y, and Z positions

36K of SRAM and 2K DRAM

- Apollo had limited resources to implement the filter
- They had to assume the forecasting model was linear
- This allowed them to implement the filter with efficient matrices
- The normal Kalman filter equations are all in terms of matrices
- That is what makes then ‘ugly’ to look at

Extended Kalman

- With resources on something like a Raspberry Pie, you can ignore the Apollo area assumptions
- Forecasting model can be non linear
- You could implement the filter with discreet equations, although a matrix approach might still be better

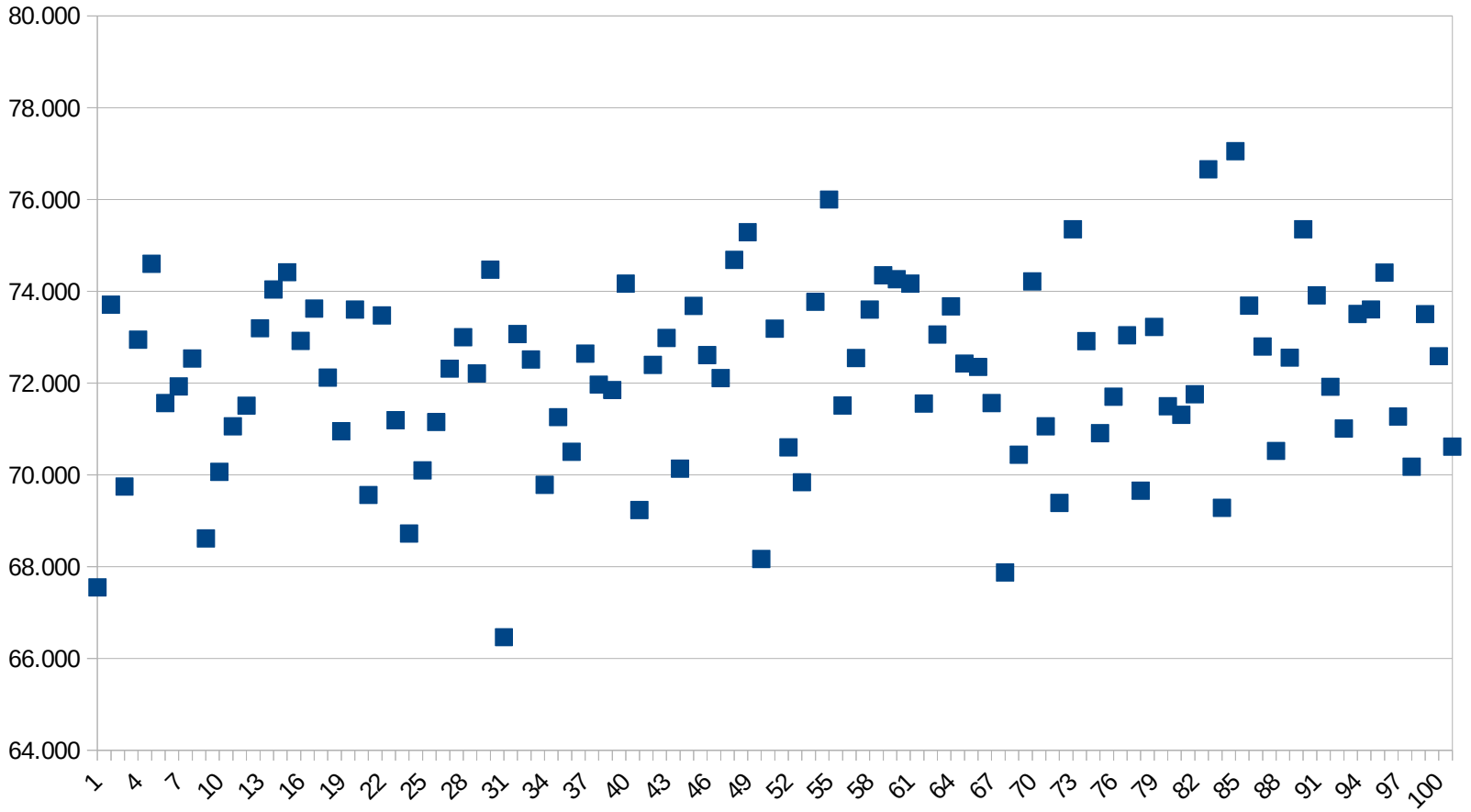
My Use

- 2D navigation of a robot using GPS
  - GPS outputs its best estimate of lat and lon as well as the expected errors in each.
  - GPS is not anywhere as good as you think.
    - +- 6 meters (18 feet) is not uncommon.
    - That is about the width of a road.

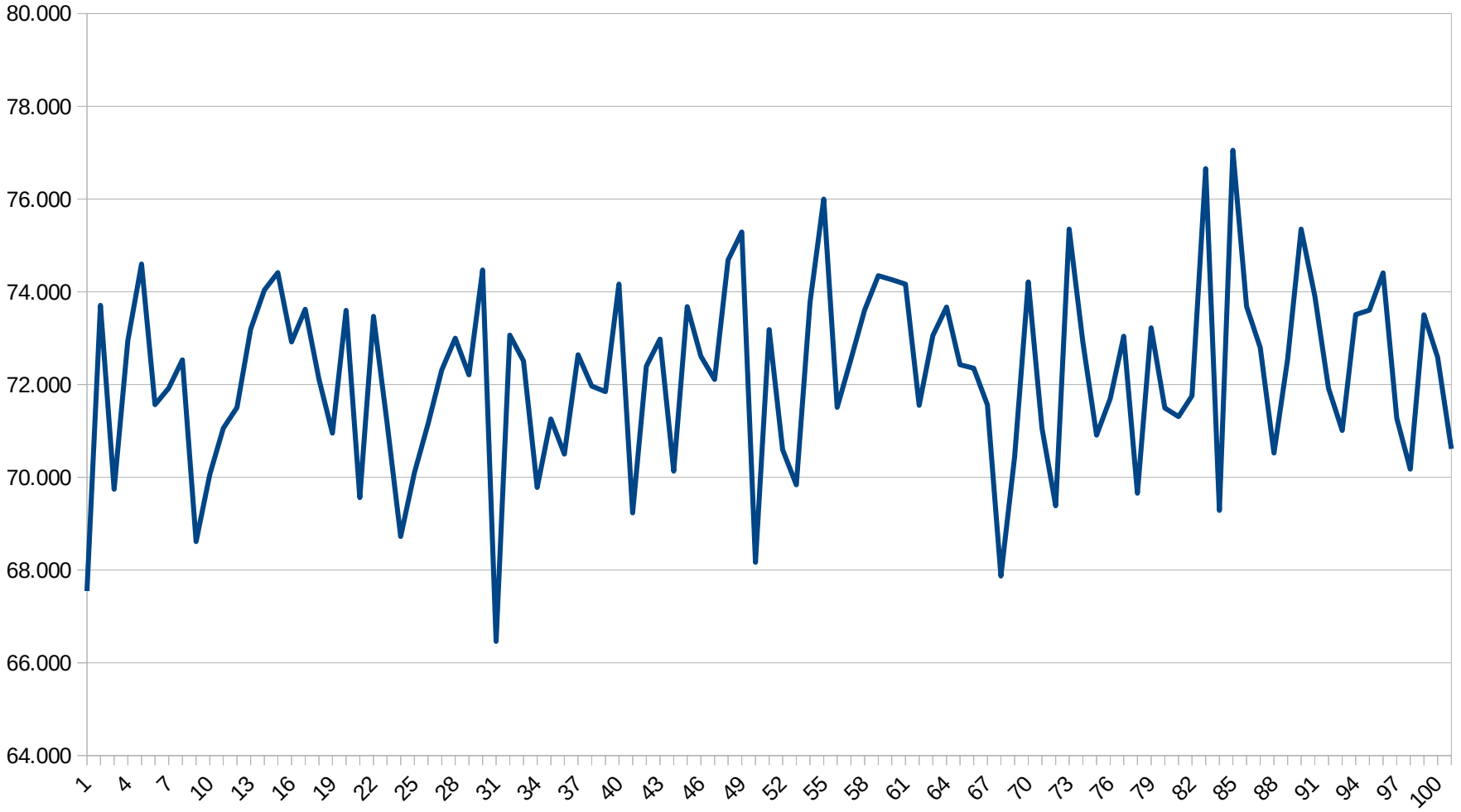
Problem

- GPS output is not short term Gaussian.
- Short term it is a Drunkard’s Walk
- Long term (3 minutes or more) it is Gaussian.

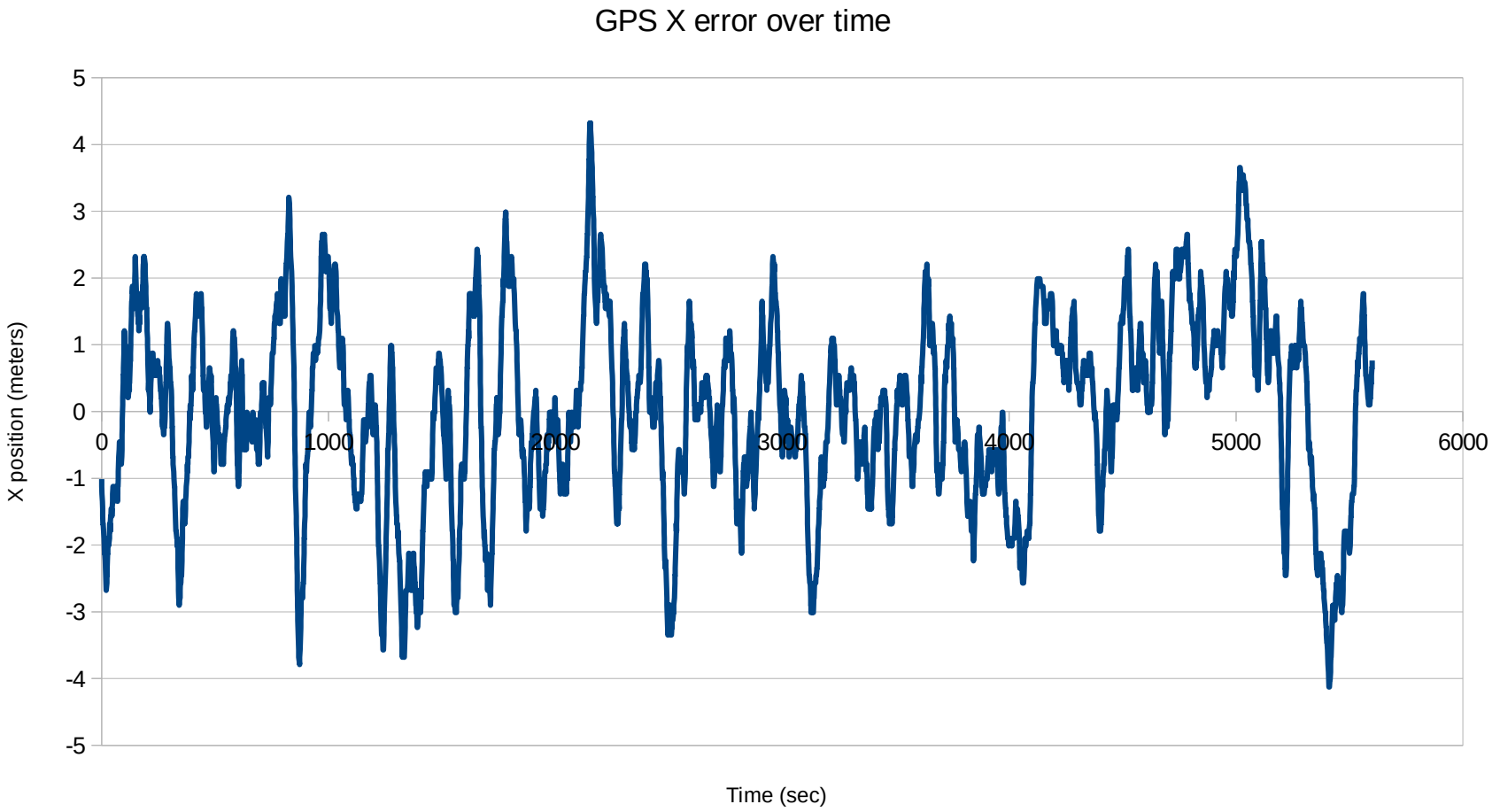
A Gaussian Distribution



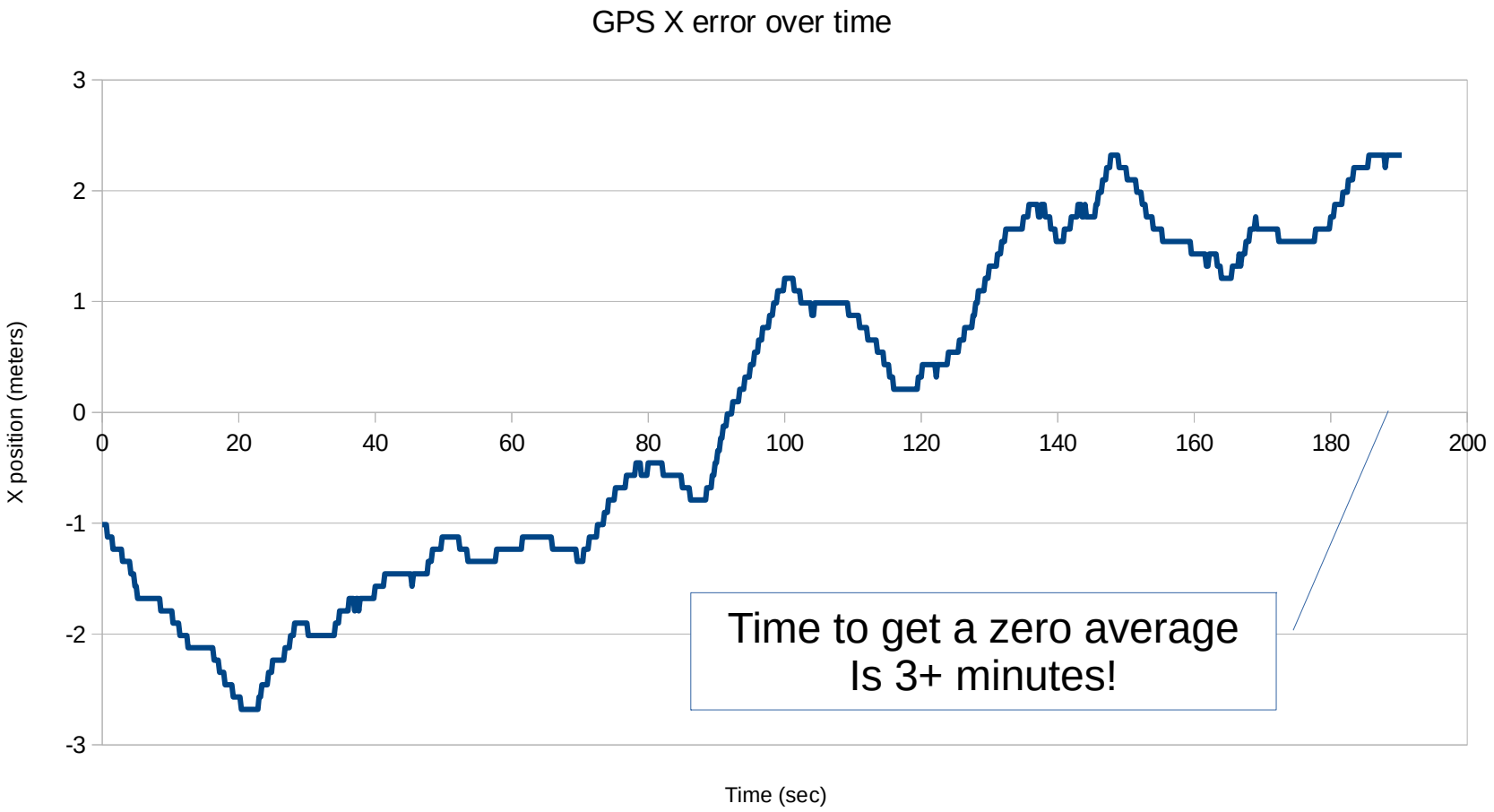
A Gaussian Distribution



GPS Error


GPS X error over time
X position (meters)
5
4
3
2
1
0
-1
-2
-3
-4
-5
0
1000
2000
3000
4000
5000
6000
Time (sec)

GPS X error over 3 minutes



Solution

- Still working on this
- Have some half baked ideas

Still To Go

- Must write a kinematic model of the robot
  - Wheel sizes, geometry, motor behavior, ground friction, weight, battery performance, ...
- Then I can use a Kalman filter to merge the long term stability of the GPS and the short term accuracy of the model
- Currently working on simulations in Python
- Implement Kalman Filter
- Test in real life

Resources (in order)

- YouTube
  - Michel Van Biezen (40 part series)
- Websites
  - www.kalmanfilter.net
- Books
  - Probabilistic Robotics (Sebastian Thrun et all)
  - Optimal State Estimation (Dan Simon)

Questions

- Skye Sweeney
- skye@fll-freak.com