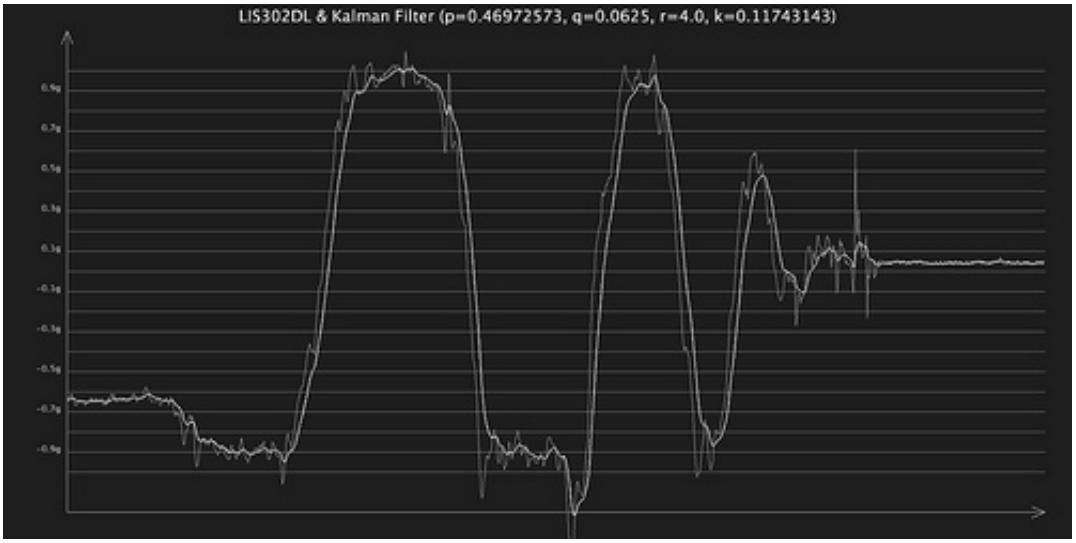
Filtering Sensor Data with a Kalman Filter

December 18, 2009

in Tinkering

Some days ago I wrote about noise with a LIS302DL accelerometer. There is obviously something wrong with my hardware. But before I get a new version out I need to implement some software filtering.



After some unsuccessful results with low pass filters I choose the Kalman Filter. The Kalman Filter involves an awful lot of complicated mathematics. But fortunately I just needed a simple one dimensional Kalman Filter without any driving information, which makes the filter much easier.

Implementing the Kalman Filter

If you check the Kalman Filter formulas, you will encounter a lot of complicated matrix and vector math. But after some research on the application domain for an LIS302DL accelerometer I found out that a simple one dimensional filter will do the job. The accelerometer puts out three dimensional acceleration data. But if you just want to get clean acceleration data from it all formulas simply deal with one dimension. So it was easier to use three different one dimensional Kalman filters. It all ended in a small set of formulas:

$x = x$
$p = p + q;$

$k = p / (p + r);$
$x = x + k * (measurement - x);$
$p = (1 - k) * p;$

The first two formulas represent the prediction of the Kalman Filter. And since there is no information about the driving forces it is very simple. The second three formulas calculate the measurement update. The variables are x for the filtered value, q for the process noise, r for the sensor noise, p for the estimated error and k for the Kalman Gain. The state of the filter is defined by the values of these variables.

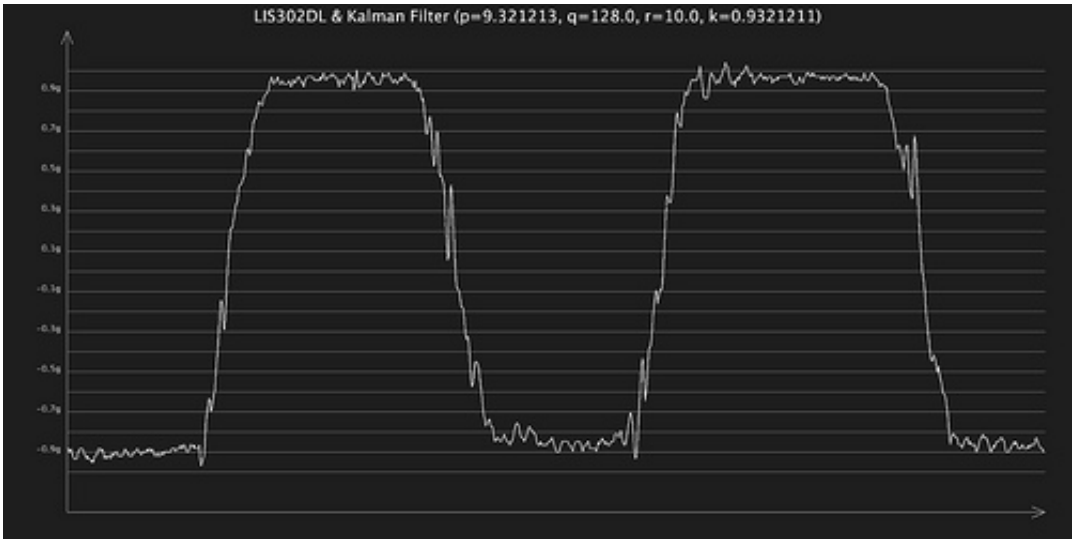
The filter is applied with each measurement and initialized with the process noise q, the sensor noise r, the initial estimated error p and the initial value x. The initial values for p is not very important since it is adjusted during the process. It must be just high enough to narrow down. The initial value for the readout is also not very important, since it is updated during the process.

But tweaking the values for the process noise and sensor noise is essential to get clear readouts.

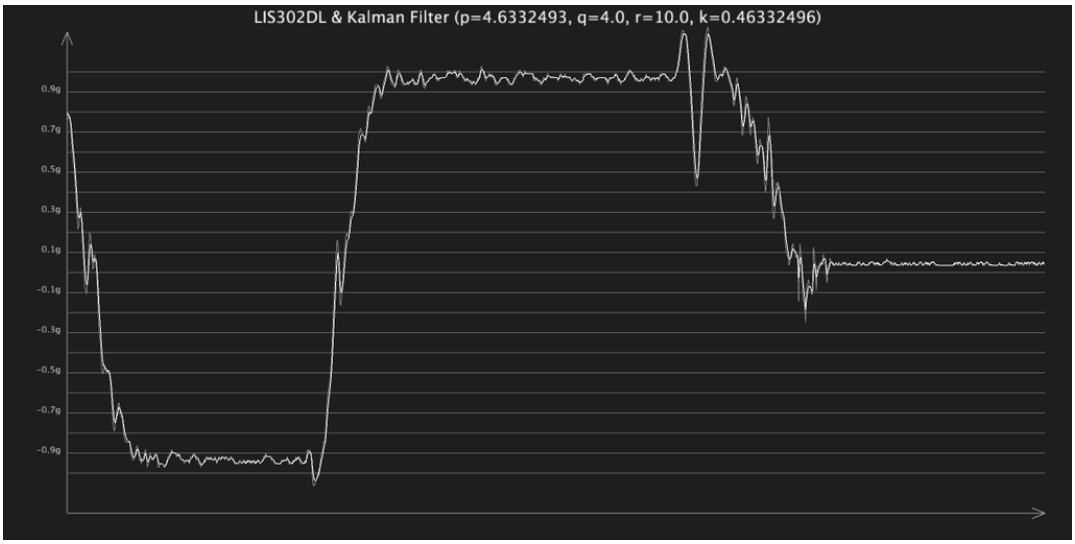
Tweaking the Kalman Filter

The first results were less than ideal. So I decided to use processing and my good old LIS302DL breakout board to get some insight of the optimal values of the different parameters:

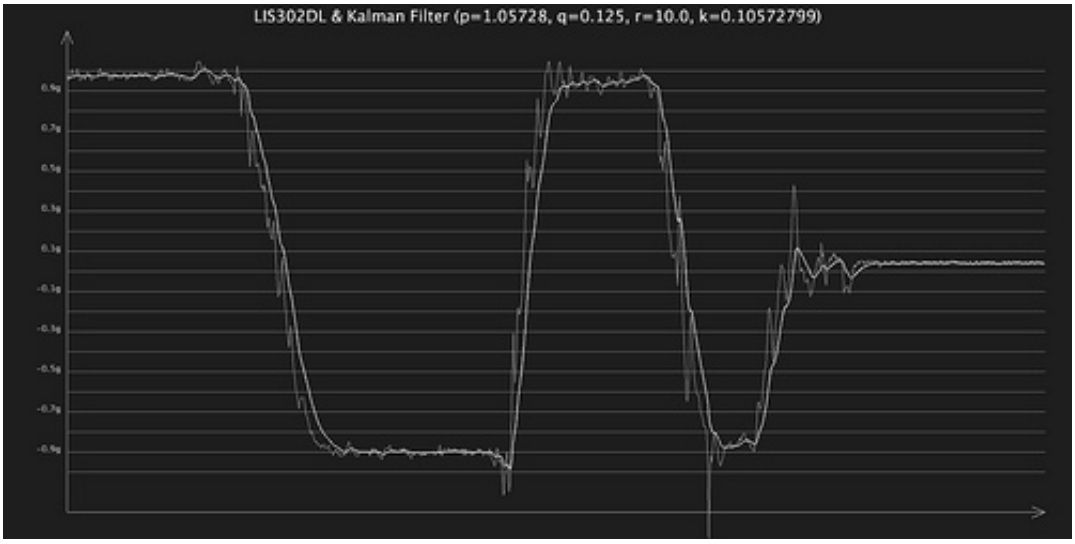
First I looked into the importance of the process noise q, starting by a very high value of 128 (which is the maximum output of the accelerometer):



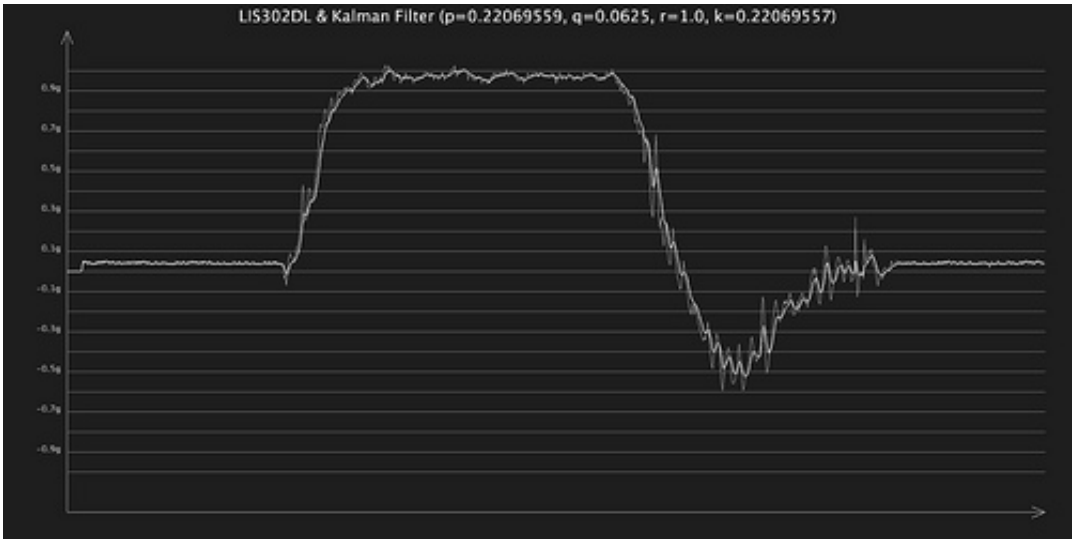
You see there is nearly no difference between the filtered data (white) and the original data (grey) – since you just see the white line.
If you turn down the process noise, e.g. to a value of 4 things start to change:



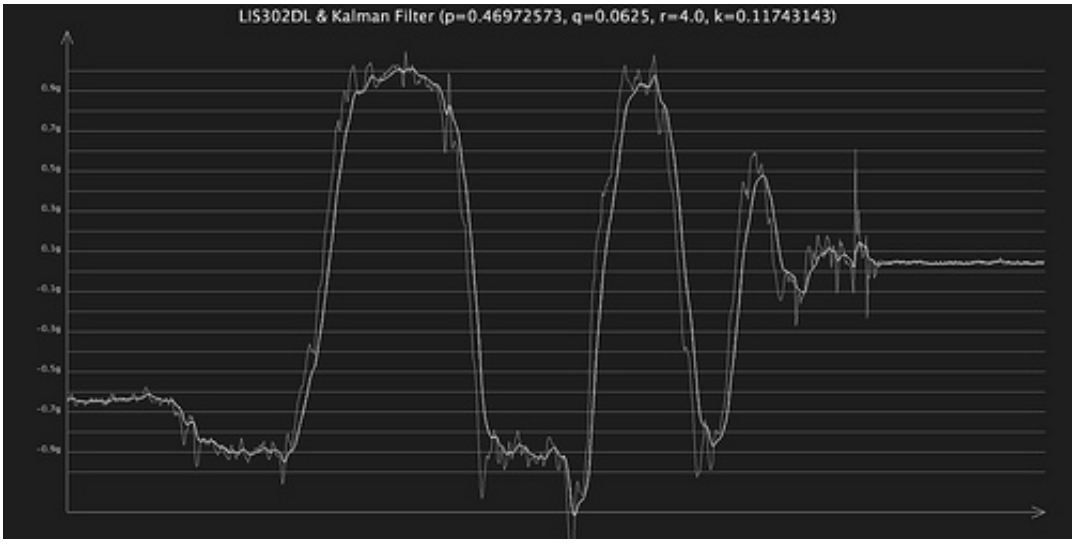
If you look closely you see that the sensor data often overshoots the filtered value. The filtered value is pretty close to the real value, but is missing a lot of noise. For my application I needed a more static output, smoothing out nearly all of the noise, giving a clean an steady signal. So lets turn down the process noise value a bit more, e.g. something like 0.125:



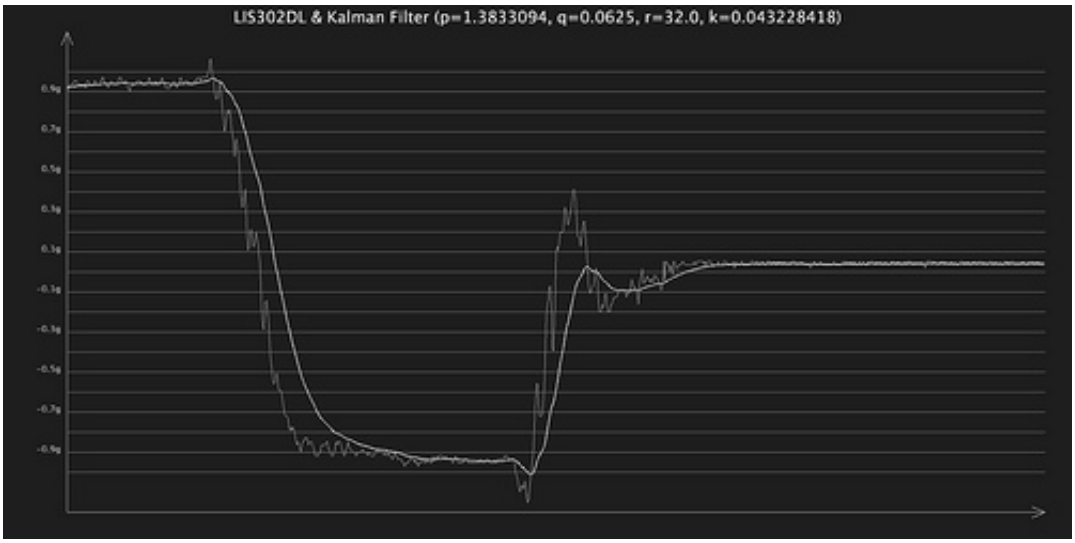
Now we got a quite a clean signal from quite a noisy source. It lags a bit behind the real data, but that is not critical for my application. Compared to the amount of noise reduction it is still quite fast.
After this is set let's play a bit around with the sensor noise *r*:
We start back at *r*=1:



As expected if we turn down the assumed noise of the sensor the Kalman filter 'relies' more on the sensor data and gives more noisy results. If we increase the noise factor of the sensor to 4 we get a more stable result again:



If we go up extreme and assume a noise level of 32 we get the expected steady result:



As expected the filtered value lags significantly behind the real measurement. But it is a much cleaner signal. But it is questionable if this is still useful.

Implementing Kalman in C

Implementing this filter in C code (e.g. for an Arduino) is quite simple. First of all we define a state for a kalman filter:

```
typedef struct {
  double q; //process noise covariance
  double r; //measurement noise covariance
  double x; //value
  double p; //estimation error covariance
  double k; //kalman gain
} kalman_state;
```

Next we need a function to initialize the kalman filter:

```
kalman_state
```

```
kalman_init(double q, double r, double p, double intial_value)
{
  kalman_state result;
  result.q = q;
  result.r = r;
  result.p = p;
  result.x = intial_value;

  return result;
}
```

And a function to update the kalman state, by calculating a prediction and verifying that against the real measurement:

```
void
kalman_update(kalman_state* state, double measurement)
{
  //prediction update
  //omit x = x
  state->p = state->p + state->q;

  //measurement update
  state->k = state->p / (state->p + state->r);
  state->x = state->x + state->k * (measurement - state->x);
  state->p = (1 - state->k) * state->p;
}
```

What did we learn from this?

First of all the Kalman filter can be much easier to implement if the underlying model is simple. We had not several dimensions, nor different sensor data to combine. The results are pretty good, but it is a challenge to find the right values for the process and sensor noise. Instead of just coming up with some guesses for the values, they can of course be estimated, but than we have to live with the result of the estimation.

In the end it is a good, robust, simple filter (in one dimension).

I think that in lack of several sensors we throw out the most advantage of the filter to combine several sensor data to a combined result. I think going a bit more in this direction can lead to real interesting results.

Or is there a much easier filter to use for my problem, leading to better results?

Related posts:

- BMP085 Barometric Pressure Sensor Breakout Boards arrived!
- Arduino & Barometric Pressure Sensor BMP085
- Decoupling LIS302DL – There I fixed it!

Tagged as: Accelerometer, Arduino, Filter, Kalman Filter, LIS302DL, Noise, sensor noise

{ 41 comments… read them below or add one }

Tod E. Kurt December 18, 2009 at 23:32

Great article. What kind of lag (in samples I guess) are you seeing?

Reply

Marcus December 19, 2009 at 09:05

I employed a very damped filter in one of my designs and you can see that the readout takes a bit longer to accept reality. You can really see how the value is updated. But this is with a very agressive filtering. and it is far less extreme as it was with a simple low pass filter.

Reply

John Honniball December 18, 2009 at 23:18

This is a really useful and informative article! I've never used a Kalman filter, although I've heard of it many times. In a sonar filtering program that I worked on some years ago, we used an Infinite Impulse Response filter, which is a bit like a moving average. It's very simple, both mathematically and as an implementation. I wrote one recently to make a VU-meter style display using an Arduino — I used the IIR filter to make a smoothly decaying peak reading dot on a line of LEDs.

Reply

SiliconFarmer December 19, 2009 at 00:30

In addition to IIR, there is the Finite Impulse Response filter, FIR, to consider. You should be able to design an FIR to clean up that signal, using formulas instead of trial and error for the number of taps and coefficient.

But perhaps that's what you used to create the low-pass filter that didn't do the job for you.

I haven't done an FIR on an AVR processor. It might not have the horsepower, depending on the number of taps you would need.

Reply

Michael December 19, 2009 at 00:35

A very pragmatic post on a broad, deep topic.

It is easy to find numerous articles, research papers and textbooks dealing with Kalman filters and its variants, but it's a rarity to find a practical, application oriented article. I guess most people believe they understand the underlying statistics and know how to readily program such a filter if they can name textbooks or make smart comments about "taming" R. E. Kalman's work.

Well done!

Michael.

Reply

Marcus December 19, 2009 at 09:09

Thanks,

to be honest. I understand not very much of the filter. I just wanted to apply it. Multidimensional problems would be much harder and woul take much longer time.
I just tried to filter my data and it worked.

Reply

Pimi December 8, 2012 at 16:14

I feel like Michael too. There are a lot of stuff about Kalman Filter. But no clear sample code. I'm trying now to implement a multi-dimensional one.

Well done
Pimi

Reply

Bala January 30, 2010 at 01:54

Hi Marcus,
I am also trying to filter sensor data but no luck with the low pass filter. Can you please post your code for the Kalman filter?? Thank you…..

Reply

Marcus January 31, 2010 at 09:41

Ok, for further references I have included the necessary pieces of code for you.

Reply

Rohit March 9, 2010 at 04:51

Hey man,

great article and I am actually trying to implement kalman filtering of accelerometer data with about 1.5 g's of sensitivity on an arduino. H/e, I was just wondering what you did to get the accelerometer data onto a graph?

Thanks

Reply

Marcus March 9, 2010 at 08:06

The graph is done with a processing script.
Basically with the Arduino I print out the numbers to the serial port and read them with my processing script and plot them as a graph in processing.

You can download the scipt here.

Reply

Rene July 16, 2010 at 03:20

Hi Marcus!

Great article! you write that you used processing for the graph plots? The link you gave seems to be code for a BMP85? where and how do you plot the serial data? Sorry, i am new to processing.

thanks so much,
-r

Sebastian Stark April 26, 2010 at 18:44

Your Kalman filter is in fact a variation of an even more simple filter:
if you skip all the q,r,p stuff and just leave k and set it to a value 0<ax = state->x + 0.06 * (measurement – state->x);
would do it. You could also calculate it this way:
(b = 1-a; a = 0.06; b = 0.94)
state->x = b*state->x + a*measurement;

Reply

Marcus April 26, 2010 at 19:11

OK, never seen it that way. Is there an official name for that filter variant? Thanks for the hint. Nevertheless reducing the Kalman to the max gave me quite a good insight into the kalman filter. Perhaps this oversimplification helps somebody, too.

Reply

Sebastian Stark April 26, 2010 at 20:06

http://en.wikipedia.org/wiki/Low-pass_filter#Algorithmic_implementation

I think its a simple Low-Pass Filter now :)

Reply

Marcus April 26, 2010 at 21:35

Ouch !
I have overseen it.
Whatever – another lesson learned.
Next time I will do a real kalman filter! With several dimensions! And proper calculations ;)

Markus Mayer January 28, 2014 at 01:15

On a side note though, the whole purpose of the Kalman filter is to find that (i.e. the optimal) k.

Reply

jobby July 12, 2010 at 10:04

hai,
generally noise is a high freuency signal. even a low pass filter can perform the function of removing noise, then what makes the kalman filter differ from low pass filter. could u pls make it clear?

Reply

Marcus July 12, 2010 at 11:11

Yes, you are right. In this special case the Kalman Filter makes not so much sense. I will post a follow up soon, with a real kalman application:
Filtering data from multiple sensors. This is where the real power of the Kalman Filter lies.
But even though this was a useless implementation I at learned at least how to apply a Kalman filter, perhaps someone else learns also something by this fruitless approach.

Reply

kalmannoob November 24, 2010 at 22:07

sir, im trying to implement my own kalman filter but ive had a trouble understanding these parts:

1. "First I looked into the importance of the process noise q, starting by a very high value of 128 (which is the maximum output of the accelerometer)" ——> how did you know that 128 is the maximum?
2. "If we go up extreme and assume a noise level of 32 " ——> same as my first question, how did you know that this is the maximum

Reply

Marcus November 25, 2010 at 09:09

128 was the maximum output of the accelrometer at 1g – so assuming the same level of noise as the output value is a extreme measuer.
The same is true for 32 – since it is a quarter of the maximum output level.
A realistic value would be lower than 32 – since we can assume that the sensor is not that noisy.
If I would know it a bit better I would talk about signal to noise ration in dB and so on.

Reply

kalmannoob November 29, 2010 at 22:10

ive implemented my own 1-d kalman. during testing though, ive noticed that even when stationary i get a huge error. for the first few 20 or so seconds i do okay but past that i get HUGE values such as this 18604.000000. any idea as to why?

Reply

Marcus December 1, 2010 at 11:14

Sorry, no.
It seems that you got a problem with either the implementation itself or the data format. Or something completely unrelated.

Reply

Samantha April 10, 2011 at 10:36

Hi, just wanted to thank you for your detailed post; I'm a university student studying how to use the Kalman Filter and your post has helped a lot thanks :)

Reply

Marcus April 10, 2011 at 16:03

If my diletantism helps you I am happy ;)

Reply

bolke May 31, 2011 at 10:21

Thanks for the article. I couldn't care less what it's called, kalman or low pass, it works. Filtering like whale in the sea. :)

Reply

Deepankar Singh November 16, 2011 at 22:28

hey i was wondering if u have given an article on 2 dimentional kalman filter ???

Regards
Deepankar Singh
NIT Rourkela

Reply

Marcus November 22, 2011 at 07:54

Hi I still think I will do a three dimensional Kalman Filter. But that may take some time.

Reply

Andy May 25, 2012 at 09:38

Hi,

This example is not realy kalman aghorithm, but is usefull low pass filter.The basic idea of the Kalman approach is measure corection in the way to calculate error between the real measurements and estimated data from the plant(process/math model /dynamic). The kalman gain K is a error corection calculated on the basis of the error covariance matrix P. Non-recursive kalman aghorithm is similar to the state observers in control theory.

Reply

Marcus May 26, 2012 at 09:12

Yes, you are right. Especially due to the fact that we do not consider any control inputs and have only one dimension and one input it is really just a low pass filter. But it helped me to understand the basics of Kalman filtering, hopefully it helps others too. Even though it is over simplified.

Reply

Janne May 29, 2012 at 07:32

The formulas include a simple first order low-pass filter:
x = x + k * (measurement – x);
I think it is a sufficient filter for many purposes. It has only one parameter to tune (k). This parameter defines the filter bandwidth. Is the Kalman filter much better than this simple low-pass filter?

Reply

Marcus May 29, 2012 at 08:59

Yes, you are right. I was very proud to implement a Kalman filter – only tolearn that I reduced it to the max, so that it is just a low pass filter. Anyway I think that my efforts perhaps help to understand the Kalman filter …

Reply

chefo June 5, 2012 at 14:53

Great article. This little code is perfect solution for multiple applications. In compare to moving avg is kalman configurable and just better.

Reply

Marco June 20, 2012 at 20:59

Hey Marcus, were you ever able to incorporate sensor fusion with the Kalman Filter? i.e. combining gyro and accelerometer data?

Reply

Marcus June 20, 2012 at 21:04

Unfortunately not yet. The principles are the same the mathematics much more complicated.

Reply

Paul July 10, 2012 at 11:36

Part of the post is hard to understand to me: where is the input signal, that is being filtered? All the considerations about process noise and sensor noise are refered to which INPUT SIGNAL? Even the first image in the post has mouse-over title “Process Noise 4.0″.

But thanks, it saved my days anyway.

Reply

Ankit Srivastava November 1, 2012 at 12:21

Hi Marcus…It was really a nice article…I would like to take a suggestion from you…we are planning to use accelerometers for body activity monitoring…where ECG will be coupled with accelerometer to identify what was the patient’s activity when his heart rate increases….we are planning to use 2-3 accelerometers and accordng to us 2g or 3g should be enough for the case….can u suggest whether i should go for Kalman filtering or any other filtering technique for the project and why??
Thanks in advance…

Reply

jayendra July 1, 2013 at 14:33

hello,
i want to use kalman filter for our robotics project using ARM7 board. we have a module of three sensors accelerometer, gyroscope, magnetometer(compass). so how can we use kalman filter for sensor fusion? just mail me if you have any useful material or sample code.
thank you

Reply

Jayne August 23, 2013 at 16:53

Hello!
Could you please take some time writing and uploading a new, simpe sketch for Arduino IDE so that I can figure out what exactly is going on in the loops?

Say my sensor reading for the x-axis is stored in the variable “x”. If I choose a new variable “a” which will hold the filtered variable and printed using Serial.println(a), what should the code be? I may later draw the graph in excel using the filtered “a” values. I would really appreciate you helping me out with this class project where i need the filtering :/

Reply

Jayne August 23, 2013 at 16:54

simple*

Reply

Marcus July 16, 2010 at 08:11

Yes, you are right, I did not post the link to the processing sketch. But you can find it on github.

Reply

Leave a Comment

Name *

E-mail *

Website

Submit

Notify me of follow-up comments by email.

Notify me of new posts by email.

{ 4 trackbacks }

- Messwerte filtern « thewknd
- Low Pass Filter – Kalman « FRC Spark Team
- Yet another Kalman filter implementation for Arduino at Didikot's blog
- Sensordaten eines Accelerometers mit Kalman-Filter bereinigen | OpenDGPS

- To search, type and hit ente

- **Popular Projects**
  - µTVBG - World Smallest TV-B-Gone clone
  - Developing a LED & Motion installation
  - Space Invaders Button
  - Gravitron
  - SkatePOV
- **Popular Posts**
  - Filtering Sensor Data with a Kalman Filter
  - Developing Software for the Atmel AVR with AVR-Eclipse, AVR-GCC & AVRDude
  - Installing KiCAD on Mac OS X
  - Driving Circuits from a CR2032 Lithium Coin Cell
  - Measure your Pulse with Arduino
- **Recent Posts**
  - The TOS-100 Arduino Stepper Motor Shield for Trinamic
  - Electric XMAS: Making Poinsettias sing!
  - Measure your Pulse with Arduino
  - Finally some footage for the LED & Motion installation
  - Installing KiCAD on Mac OS X
- **On Github**

  - aJson
    aJson is an Arduino library to enable JSON processing with Arduino. It easily enables you to decode, create, manipulate and encode JSON directly from and to data structures.
    April 25, 2014 - 12:36

- LiPoCharger

  April 24, 2014 - 13:42

- TrickleCharger

  April 22, 2014 - 14:51

- CmdMessenger
  Fork of Neil Dudman's CmdMessenger. Serial messaging system for Arduino platform
  April 15, 2014 - 12:04

- TMC26XStepper-Generator
  a dummy implementation which does not communicate but generates register values
  March 31, 2014 - 14:45

- Twittballon2
  An Arduino based physical Twitter visualization
  February 20, 2014 - 08:13

- Kultpfunzel
  The Customisable Smart Torch
  February 5, 2014 - 20:00

- WiFly
  copy of sparkfun Wifly Library
  February 4, 2014 - 03:39

Recent Comments

- Marcus on Arduino JSON Library
- 64 Pixels Roundup @ interactive-matter.org | My 2µF on 64 Pixels Roundup
- donsan on Arduino JSON Library
- Serial JSON-RPC Server for Arduino - Cloud Rocket on aJson – Handle JSON with Arduino
- Mostrando los datos de netatmo con un Arduino | josema™ on aJson – Handle JSON with Arduino

License