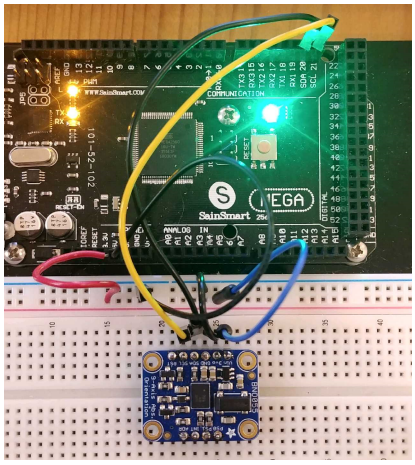
Names: Trevor Liu, Imtisal Ahmad, Ian Yurychuk, Sharuhen Paul
ID : 1589202, 1495401, 1552809
CMPUT 275 Winter2020
Final Project: Kalman Filter using ATMEGA2560 and BNO055 accelerometer

I. Included Files:

* server.cpp
* Kalman.cpp
* kalman.h
* main.cpp
* Makefile
* README
* serial_port.h
* serial_port.cpp
* pdata
* rdata

II. Wiring Instructions:

Follow these instructions hosted on Adafruit's webpage: https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/arduino-code

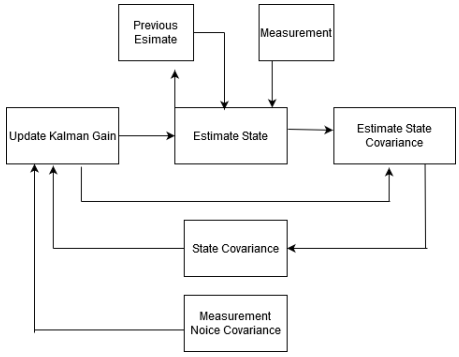


III. Backgroud:

For our final project we interfaced the "Smart Sensor" BNO055 accelerometer, gyroscope, and magnetometer with an Arduino MEGA. We will implement a Kalman filter to reduce noise from our measurements, and to track state variables position and velocity. The results of our filtered measurements will be displayed to a graph on the touchscreen. We will be using a with a one dimensional filter. The sensor will be moved and will detect acceleration readings which will then be communicated in a client server manner, similar to assignment two part two. Then those values will then be saved in matrices (using the eigen library) and used for our Kalman filter algorithmn. Which will then process those values to predict the next estimated acceleration.
The Kalman filter is a commonly used state estimation algorithm. For the objective of filtering noise from sensor measurements, a more traditional filter could be used (such as a low pass filter). The Kalman filter excels in more complex tracking applications, such as estimating the position and velocity of a vehicle.

This goal of this project is to use the Kalman filter to track state variables of position and velocity using the output of an accelerometer. The Kinematic equations will be used to describe these state variables. Since our only measurement will be of acceleration, obtaining the position position variable will be equivalent to double integration in terms of accuracy. Additional measurements would improve this accuracy (such as velocity measurements from a DVL). The ultimate purpose of the project is not to serve any practical application, but to provide an introduction to the Kalman filter algorithm.

The Kalman filter follows an iterative process in which the values of the state variables are estimated based on an original estimate, measurement, and state model. At each iteration, more trust will be placed in either the previous estimate or the current measurement, this is referred to as the Kalman gain. The Kalman gain is also updated upon each iteration based on an approximation of the error of the estimation.



The subscript k represents the current iteration, and p

represents prediction.

x_k - estimated value x_p - predicted value z_k - measured value

The estimated value is calculated by multiplying the difference between the prediction and measurement by *K*, the Kalman gain.

$$x_k = x_p + K(x_p - z_k)$$

The Kalman gain can be found using the error of the prediction P_k and the error of the measurement *R*.

$$K = \frac{P_k}{P_k + R}$$

P_k is updated every iteration using the previous iteration P_{k-1} and the Kalman gain *K*.

$$P_k = \frac{R\ P_{k-1}}{R + P_{k-1}}$$

$$\implies P_k = K(1 - P_{k-1})$$

Since the state we are tracking more than one value, we will require these equations in matrix form, starting with the state vector $\vec{x_k}$ which is the position x_k and velocity $\dot{x}_k$

$$\vec{x_k} = \begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix}$$

The process will cycle between a prediction and update (measurement) steps. These will both be executed once every iteration.

Prediction:
$\vec{x_{kp}} = \mathbf{A}\vec{x_{k-1}} + \mathbf{B}z_k$
$\mathbf{P_{kp}} = \mathbf{AP_{k-1}A^T} + \mathbf{Q}$

Update:
$\mathbf{K_k} = \mathbf{P_{kp}H^T}(\mathbf{HP_{kp}H^T} + \mathbf{R})^{-1}$
$\vec{x_k} = \vec{x_{kp}} + \mathbf{K_k}(z_k - \mathbf{H}\vec{x_{kp}})$
$\mathbf{P_k} = (\mathbf{I} - \mathbf{K_kH})\mathbf{P_{kp}}$

The state variables can be modelled with the following kinematic equations:

$$x_{k+1} = x_k + \dot{x}_k\Delta t + \frac{1}{2}a_k(\Delta t)^2$$

$$\dot{x}_{k+1} = \dot{x}_k + a_k\Delta t$$

and in matrix form:

$$\vec{x}_k = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} \frac{(\Delta)^2}{2} \\ \Delta t \end{bmatrix} a_k$$

$\mathbf{P_k}$ is the state covariance matrix.

$$\mathbf{P_k} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_v^2 \end{bmatrix}$$

The zero entries are the covariance between position and velocity. For simplicity we will leave these entries as zero. σ_x^2 represents variance for position and σ_v^2.

$\mathbf{R}$ is the measurement noise covariance matrix.

$$\mathbf{R} = \begin{bmatrix} \sigma_{zx}^2 & 0 \\ 0 & \sigma_{zv}^2 \end{bmatrix}$$

The zero entries are covariance and will again be left as zero. σ_{zx}^2 represents variance for position and σ_{zv}^2.

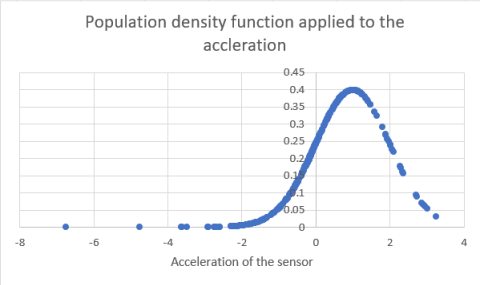
σ_x^2, σ_v^2, σ_{zx}^2 σ_{zv}^2 are determined experimentally. Datasets are collected from measurements. For the $\mathbf{P}$ entries, the measurements are taken during the process to give a range of normal values during operation. $\mathbf{R}$ entries are generated from a dataset of the same measurement, giving the variance of one measurement. Since the only measurement we can take is of acceleration and the variances we require are for position and velocity, these values require more calculations. For our example we will use the variance from acceleration for each entry.

$\mathbf{Q}$ is the process noise covariance matrix. We will just set this to zero.

$\mathbf{H}$ will be the identity matrix. It's purpose is to ensure that the each matrix is the right size for the necessary matrix multiplications.

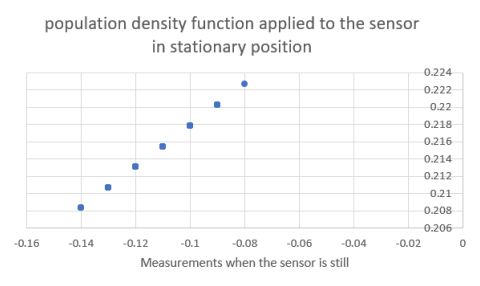
The datasets taken for variance calculations are included in the file "pdata" and "rdata". The calculations and plots were completed using excel.

For the state covariance matrix, the measurements taken produced a variance of $1.117419876\frac{m}{s^2}$ with the following PDF:



The PDF for this dataset is a clear Gaussian distribution.

For the measurement noise covariance matrix, the measurements taken produced a variance of 9.4661^5 $\frac{m}{s^2}$ with the following pdf:



The PDF for this dataset looks odd because the sensor used is quite accurate. Although a large dataset was used, only seven different values were recorded.

IV. Makefile Targets:

- make (server): build the server. - make clean: Remove all object and files and executables.

V. Running Instruction:

1. Type in the command line ”make” while in the directory of the file 2. Type ”./server” 3. Type ”make clean” to remove the object and executable