**KALMAN FILTER**

**Algorithm: Kalman Filter**

**Kalman Filter** works on primarily using the concept of Full Bayesian inference. g-h Filter primarily works on the concept of point estimation however, Kalman filter computes the Full Posterior of the new position of the object.

Algorithm uses adaptive updates for both the amount of uncertainty and the position of the object. The MAP estimates at all points could give us the best path of the object with uncertainty denoted by variance of posterior.

Graphical user interface, text, application, email

Description automatically generated

The generalised algorithm for Kalman filter is as follows -

Graphical user interface, text, application, email

Description automatically generated

**Variation 1 – Standard Kalman Algorithm**

Testing involves 6 methods, and each method was run with 2 different initial conditions.

Base – Testing the error in measured data (Filter should provide an error less than this)

Example 1 – Initial Position 125 and Initial Gain 1 (Close to actual)

Example 2 – Initial Position 175 and Initial Gain 5 (Far from actual)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Collection Algorithm** | **Example** | **Root Mean Squared Error** | **Mean Absolute Error** | **KL-Divergence** | **Inverse KL-Divergence** |
| Method 1 | Base | 38.854 | 17.697 | 0.0562 | 0.0499 |
| Example 1 | 23.234 | 16.464 | 0.018 | 0.017 |
| Example 2 | 27.498 | 21.612 | 0.017 | 0.016 |
| Method 2 | Base | 19.953 | 8.3478 | 0.01358 | 0.01172 |
| Example 1 | 13.315 | 9.77 | 0.0047 | 0.0042 |
| Example 2 | 18.354 | 15.928 | 0.0047 | 0.0044 |
| Method 3 | Base | 38.727 | 17.4288 | 0.0565 | 0.0503 |
| Example 1 | 22.996 | 16.169 | 0.018 | 0.017 |
| Example 2 | 27.207 | 21.254 | 0.017 | 0.016 |
| Method 4 | Base | 0.4053 | 0.4013 | 7.606x10-7 | 7.609x10-7 |
| Example 1 | 1.221 | 1.188 | 8.851x10-6 | 8.8405x10-6 |
| Example 2 | 7.587 | 7.459 | 0.00022 | 0.00021 |
| Method 5 | Base | 0.05774 | 0.05 | 2.079x10-7 | 2.078x10-7 |
| Example 1 | 1.616 | 1.592 | 1.207x10-5 | 1.205x10-5 |
| Example 2 | 7.983 | 7.863 | 0.00024 | 0.00023 |
| Method 6 | Base | 0.0592 | 0.0511 | 2.155x10-7 | 2.155x10-7 |
| Example 1 | 1.622 | 1.598 | 1.225x10-5 | 1.223x10-5 |
| Example 2 | 7.988 | 7.87 | 0.00024 | 0.00023 |

The data shows that for Variation 1 the measurements under Methods 1, 2 and 3 are filtered by the algorithm effectively when appropriate initial conditions are applied. However, for last 3 methods the algorithm does not help in filtering the data. However, the result for this variation on the last 3 methods is almost similar to the best variation of g-h Filter.

**Variation 2 – Point Velocity updates with Outlier Removal**

Testing involves 6 methods, and each method was run with 2 different initial conditions.

Example 1 – Initial Position 125 and Initial Gain 1 (Close to actual)

Example 2 – Initial Position 175 and Initial Gain 5 (Far from actual)

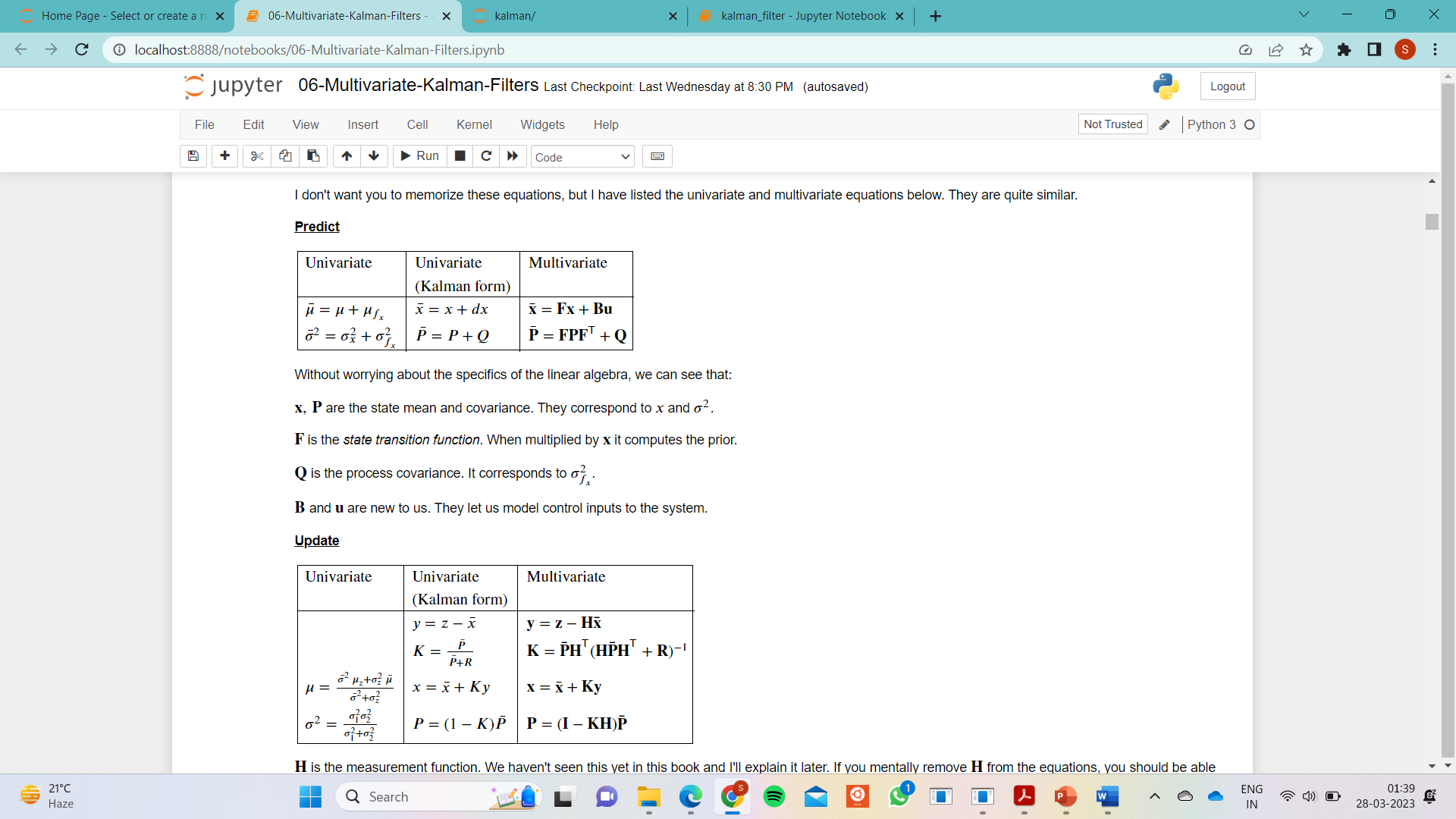
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Collection Algorithm** | **Example** | **Root Mean Squared Error** | **Mean Absolute Error** | **KL-Divergence** | **Inverse KL-Divergence** |
| Method 1 | Base | 38.854 | 17.697 | 0.0562 | 0.0499 |
| Example 1 | 9.777 | 8.477 | 0.0026 | 0.0025 |
| Example 2 | 9.893 | 8.505 | 0.0026 | 0.0026 |
| Method 2 | Base | 19.953 | 8.3478 | 0.01358 | 0.01172 |
| Example 1 | 9.733 | 7.577 | 0.0032 | 0.0031 |
| Example 2 | 9.879 | 7.668 | 0.0033 | 0.0032 |
| Method 3 | Base | 38.727 | 17.4288 | 0.0565 | 0.0503 |
| Example 1 | 9.477 | 8.167 | 0.0025 | 0.0025 |
| Example 2 | 9.599 | 8.199 | 0.00264 | 0.00262 |
| Method 4 | Base | 0.4053 | 0.4013 | 7.606x10-7 | 7.609x10-7 |
| Example 1 | 5.055 | 2.937 | 0.0016 | 0.0016 |
| Example 2 | 24.986 | 11.653 | 0.0163 | 0.0153 |
| Method 5 | Base | 0.05774 | 0.05 | 2.079x10-7 | 2.078x10-7 |
| Example 1 | 5.25 | 2.819 | 0.0017 | 0.0016 |
| Example 2 | 25.12 | 11.591 | 0.016 | 0.015 |
| Method 6 | Base | 0.0592 | 0.0511 | 2.155x10-7 | 2.155x10-7 |
| Example 1 | 5.2102 | 2.803 | 0.0016 | 0.0016 |
| Example 2 | 25.108 | 11.575 | 0.016 | 0.015 |

The data shows that for Variation 2 the measurements under Methods 1, 2 and 3 are filtered by the algorithm very effectively as compared to the previous filter when appropriate initial conditions are applied. However, for last 3 methods the algorithm does not help in filtering the data.

**MULTIVARIATE KALMAN FILTER**

**Algorithm: Multivariate Kalman Filter**

**Multivariate Kalman Filter** generalises the previous approach to its full potential of a multi-dimensional full Bayesian inference.



The generalised algorithm for Kalman filter is as follows -

Graphical user interface, text, application, email

Description automatically generated

**Variation 3 – Multidimensional Kalman filter with Outlier Removal**

**Results got are statistically same as the previous variation however the mechanism in which outliers are detected is different here.**

Testing involves 6 methods, and each method was run with 2 different initial conditions.

Example 1 – Initial Position 125 and Initial Gain 1 (Close to actual)

Example 2 – Initial Position 175 and Initial Gain 5 (Far from actual)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Collection Algorithm** | **Example** | **Root Mean Squared Error** | **Mean Absolute Error** | **KL-Divergence** | **Inverse KL-Divergence** |
| Method 1 | Base | 38.854 | 17.697 | 0.0562 | 0.0499 |
| Example 1 | 9.976 | 8.405 | 0.0029 | 0.0029 |
| Example 2 | 10.094 | 8.446 | 0.003 | 0.0029 |
| Method 2 | Base | 19.953 | 8.3478 | 0.01358 | 0.01172 |
| Example 1 | 10.393 | 7.698 | 0.0042 | 0.0041 |
| Example 2 | 10.508 | 7.741 | 0.0043 | 0.0041 |
| Method 3 | Base | 38.727 | 17.4288 | 0.0565 | 0.0503 |
| Example 1 | 9.686 | 8.093 | 0.0029 | 0.0029 |
| Example 2 | 9.808 | 8.134 | 0.00298 | 0.00295 |
| Method 4 | Base | 0.4053 | 0.4013 | 7.606x10-7 | 7.609x10-7 |
| Example 1 | 42.755 | 19.601 | NaN | NaN |
| Example 2 | 207.64 | 77.74 | NaN | NaN |
| Method 5 | Base | 0.05774 | 0.05 | 2.079x10-7 | 2.078x10-7 |
| Example 1 | 42.314 | 18.663 | 0.073 | 0.095 |
| Example 2 | 209.23 | 77.44 | NaN | NaN |
| Method 6 | Base | 0.0592 | 0.0511 | 2.155x10-7 | 2.155x10-7 |
| Example 1 | 43.328 | 19.729 | NaN | NaN |
| Example 2 | 209.437 | 78.505 | NaN | NaN |

The data shows that for Variation 2 the measurements under Methods 1, 2 and 3 are filtered by the algorithm very effectively as compared to the previous filter when appropriate initial conditions are applied. However, **for last 3 methods the algorithm behaves destructively and adds more noise**.

**Comparison among Variations of Kalman Filter**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Method Number** | **Examples** | **Variation 1** | **Variation 2** | **Variation 3** |
| **Method 1** | **Example 1** |  | **Better** | |
| **Example 2** |  | **Better** | |
| **Method 2** | **Example 1** |  | **Better** | |
| **Example 2** |  | **Better** | |
| **Method 3** | **Example 1** |  | **Better** | |
| **Example 2** |  | **Better** | |
| **Method 4** | **Example 1** | **Best** |  | **Worst** |
| **Example 2** | **Best** |  | **Worst** |
| **Method 5** | **Example 1** | **Best** |  | **Worst** |
| **Example 2** | **Best** |  | **Worst** |
| **Method 6** | **Example 1** | **Best** |  | **Worst** |
| **Example 2** | **Best** |  | **Worst** |

From the above observations in general using outlier detection is not beneficial for g-h Filter and we should go for the standard algorithm itself which in general gives better filtering results. Also, for Methods 4,5 and 6 all the variations are not overall effective.

**Comparison between Kalman Filter with g-h Filter Variation 1 (Best)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Number** | **Examples** | **g-h Filter**  **(Variation 1)** | **Kalman Filter**  **(Variation 2/3)** |
| **Method 1** | **Example 1** |  | **Better** |
| **Example 2** |  | **Better** |
| **Method 2** | **Example 1** |  | **Better** |
| **Example 2** |  | **Better** |
| **Method 3** | **Example 1** |  | **Better** |
| **Example 2** |  | **Better** |
| **Method 4** | **Example 1** | **Better** |  |
| **Example 2** | **Better** |  |
| **Method 5** | **Example 1** | **Better** |  |
| **Example 2** | **Better** |  |
| **Method 6** | **Example 1** | **Better** |  |
| **Example 2** | **Better** |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Number** | **Examples** | **g-h Filter**  **(Variation 1)** | **Kalman Filter**  **(Variation 1)** |
| **Method 1** | **Example 1** |  | **Marginally Better** |
| **Example 2** | **Marginally Better** |  |
| **Method 2** | **Example 1** |  | **Marginally Better** |
| **Example 2** | **Marginally Better** |  |
| **Method 3** | **Example 1** |  | **Marginally Better** |
| **Example 2** | **Marginally Better** |  |
| **Method 4** | **Example 1** | **Equal** | |
| **Example 2** | **Equal** | |
| **Method 5** | **Example 1** | **Equal** | |
| **Example 2** | **Equal** | |
| **Method 6** | **Example 1** | **Equal** | |
| **Example 2** | **Equal** | |