Московский авиационный институт
(национальный исследовательский университет)
Институт № 8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Курсовая работа

По курсу «Численные методы»

тема «Фильтр Калмана»

Выполнил:
Студент группы М8О-409Б-20
Алымов Максим Александрович
Проверил:
Доцент кафедры 806
Пивоваров Дмитрий Евгеньевич

Москва 2023

Введение

Датчики шумные. Данные, которые проиходят от них всегда обладают разбросом. Это объясняет потребность в существовании фильтров как таковых. Но бывают стуации, когда данные нужно получать быстро и точно; пример: навигационная ситема автопилота. Более того, алгоритм не должен быть вычислительно сложным, чтобы работать в реальном времени. Посмотрим как с этой задачей справляется алгоритм фильтра Калмана.

Предпосылки и идеи

Представим ситуацию: самолет летит с постоянной скоростью в режиме автопилота. Но ввиду различных факторов(порывы ветра, осадки, движения экипажа и пассажиров в салоне), датчики будут выдавть значения, которые будут разбросаны вблизи какого-то значения скорости. Но, например, ввиду поддуваний ветра датчики начали давать разброс возле другого значения, тогда, чтобы минимизировать ошибку мы изменим наше предположение о текущей скорости. Это и есть основная идея фильтра Калмана - объединять данные датчиков и представления о движении системы, для получения значения с наименьшим разбросом.

Одномерный фильтр Калмана

Рассмоторим следующую задачу, допустим у нас есть объект(авто, самолет, спутник и т.п.) который двигается равномерно по прямой, и у нас есть измерения положения этого объекта, которые мы получаем каждую секунду. Как по этим данным посторить оценку положения?

Формализум сказанное выше;

Модель движения:

$$x_{k+1} = x_k + v\Delta t + w, \text{ где}$$

x_k - положение в момент времени k

v - положение в момент времени
Δt - период дискретизации
w - шум так же примем обозначение $\Delta x = v\Delta t$

Модель измерения:

$$z_k = x_k + \eta, \text{ где}$$

z_k - измерение положения в момент времени k
η - шум

Помимо того будем считать, что случайные величины, с которыми мы работаем распределены нормально

Итак у нас есть модель и измерения, для того чтобы объединить эту информацию фильтр Калмана использует формулы условных вероятнотсей.

Фильтр Калмана использует следующие соотношения:

- формула условной вероятности

$$P(x|z) = \frac{P(x)P(z|x)}{P(z)}$$

- формула полной вероятности

$$P(x_i) = \sum_i P(x_{i-1})P(x_i|x_{i-1})$$

- формула Байеса

$$P(x|z) = \frac{P(x)P(z|x)}{P(z)}$$

Поскольку мы работаем снормльно распределенными величинами; то для того, чтобы знать распределения, нам необходимы лишь два знчения: μ - мат. ожидание и σ^2 - дисперсия

Алгоритм фильтра Калмана разделяется на два этапа:
- предсказание(prediction) - априорная оценка
- обновление(update) - апостреriорная оценка

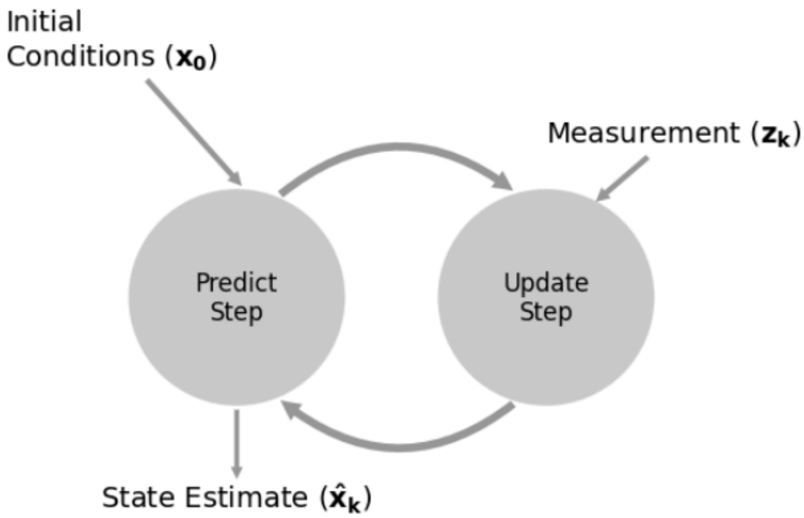

Figure 1: Filter scheme

В нашем случае

Prediction	Update
$\overline{x}_k = \hat{x}_{k-1} + \Delta x$ $\overline{\sigma}_k^2 = \sigma_{k-1}^2 + \sigma_w^2$	$\hat{x}_k = \frac{\sigma_\eta^2 \overline{x}_k + \overline{\sigma}_k^2 z_k}{\overline{\sigma}_k^2 + \sigma_\eta^2}$ $\sigma_k^2 = \overline{\sigma}_k^2 + \sigma_\eta^2$

Пояснение: этап обновления дает нам среднее и дисперсию для новой оценки; если мы попробуем вычислить плотность $P(\hat{x}|z) = \frac{P(\hat{x})P(z|\hat{x})}{P(z)}$, то мы опять получим нормальное распределение со соответствующими параметрами.

Введем новые обозначения:

$Q = \sigma_w^2$; $\overline{P}_k = \overline{\sigma}_k^2$; $R = \sigma_\eta^2$; $y_k = z_k - \overline{x}_k$

$P_k = \sigma_k^2$

Тогда можем переписать уравнения в новых обозначениях

Prediction	Update

$\overline{x}_k = \hat{x}_{k-1} + \Delta x$ $\overline{P}_k = P_{k-1} + Q$	$y_k = z_k - \overline{x}_k$ $K = \frac{\overline{P}_k}{\overline{P}_k + R}$ $\hat{x}_k = \overline{x}_k + K y_k$ $P_k = \overline{P}_k + R$

Посмотрим на результат работы такого фильтра

Результат моделирования:

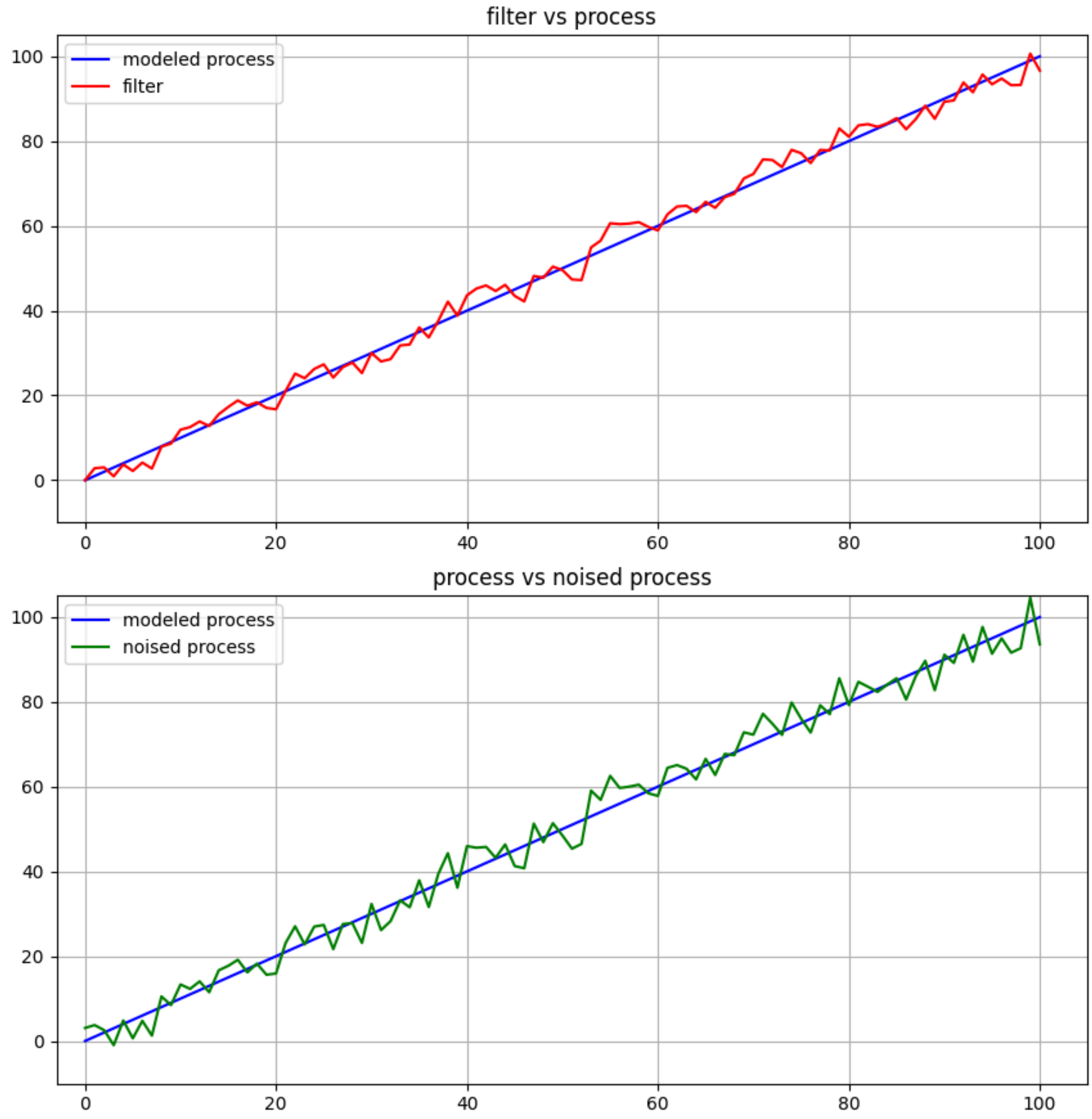

Figure 2: Filter vs model vs noise

Оценивать качество работы фильтра можно с помощью СК-ошибки

filter_error	pure_error
6.445225374768028	13.662836949233665

Многомерный фильтр Калмана

Легко представить пример, когда нам требуется оценка не одной, а нескольких переменных. И даже невсегда они будут независимы(как в случае пространственого положения: координат x и y). Если, например, фильтр разрабатывается для отслеживания движущегося в пространстве объекта, то нам может потребоваться и скорость и ускорение(такой вариант уже дает 9 переменных). Кроме того, поскольку фильтр Калмана использует информацию о динамике модели, то добавление переменных состояния(как например скорости к алгоритму для оценки положения) проиводит к повышению точности работы фильтра.

Будем рассмотривать ту же модель, что и в примере с одномерным фильтром, но сам фильтр уже будет оценивать две переменные - положение и скорость.

Мы помним, что работа филтра осуществляется в два этапа, посмотрим как устроен каждый из них в многомерном случае, но более формально;

Предсказание

$$\overline{\boldsymbol{x}}_k = \boldsymbol{F}\hat{\boldsymbol{x}}_{k-1}$$

$$\overline{\boldsymbol{P}}_k = \boldsymbol{F}\boldsymbol{P}_{\boldsymbol{k-1}}\boldsymbol{F}^T + \boldsymbol{Q}$$

Здесь мы, основываясь на оценке предыдущего состояния, предсказываем значение в следующий момент времени. Соответственно в первой строке вычисляем мат. ожидание, во второй строке - ковариационную матрицу(которая заменяет дисперсию, в случае, если мы работаем со случайными векторами). $\boldsymbol{F}$ - переходная матрица, $\boldsymbol{Q}$ - ков. матрица шума процесса.

Обновление

$$\boldsymbol{y_k} = \boldsymbol{z}_k - \boldsymbol{H}\overline{\boldsymbol{x}}_k$$
$$\boldsymbol{K} = \overline{\boldsymbol{P}}_k\boldsymbol{H}^T\left(\boldsymbol{H}\overline{\boldsymbol{P}}_k\boldsymbol{H}^T + \boldsymbol{R}\right)^{-1}$$
$$\hat{\boldsymbol{x}}_{k+1} = \overline{\boldsymbol{x}}_k + \boldsymbol{K}\boldsymbol{y_k}$$
$$\boldsymbol{P}_{k+1} = (\boldsymbol{I} - \boldsymbol{H}\boldsymbol{K})\overline{\boldsymbol{P}}_k$$

На этом этапе вычиления следует следующей логике:

- строка 1: вычисляем остаток - разница между измерением и предсказанием
- строка 2: пока игнорируем
- строка 3: вычисляем оценку состояния в следующий момент времени
- строка 4: вычисляем дисперсию(ковариационную матрицу) оценки

Теперь поясненим, что происходит во 2-ой строке: желаемая оценка должна быть оптимальной, т.е. ее дисперсия должна быть минимальной; с точки зрения математики это означает $\operatorname{tr}(\boldsymbol{P}_{k+1}(\boldsymbol{K})) \to \min_{\boldsymbol{K}}$ после чего, из уравнения $\frac{\mathrm{d}\operatorname{tr}(\boldsymbol{P}_{k+1})}{\mathrm{d}\boldsymbol{K}} = 0$

Подытожим:

Prediction	Update
$\overline{\boldsymbol{x}}_k = \boldsymbol{F}\hat{\boldsymbol{x}}_{k-1}$ $\overline{\boldsymbol{P}}_k = \boldsymbol{F}\boldsymbol{P_{k-1}}\boldsymbol{F}^T + \boldsymbol{Q}$	$\boldsymbol{y_k} = \boldsymbol{z}_k - \boldsymbol{H}\overline{\boldsymbol{x}}_k$ $\boldsymbol{K} = \overline{\boldsymbol{P}}_k\boldsymbol{H}^T\left(\boldsymbol{H}\overline{\boldsymbol{P}}_k\boldsymbol{H}^T + \boldsymbol{R}\right)^{-1}$ $\hat{\boldsymbol{x}}_{k+1} = \overline{\boldsymbol{x}}_k + \boldsymbol{K}\boldsymbol{y_k}$ $\boldsymbol{P}_{k+1} = (\boldsymbol{I} - \boldsymbol{H}\boldsymbol{K})\overline{\boldsymbol{P}}_k$

Посмотрим на результат работы такого фильтра

Результат моделирования:

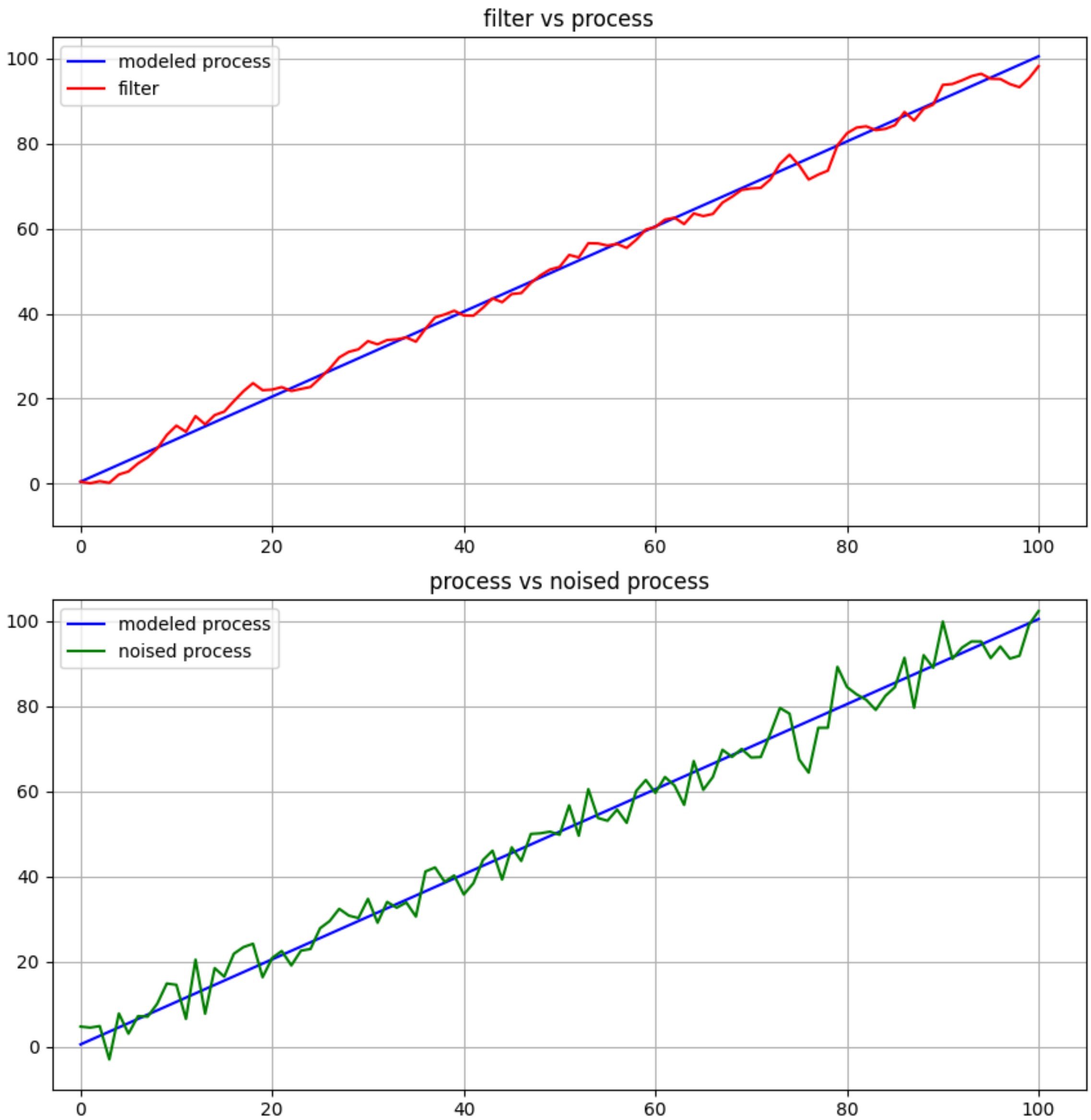

Figure 3: Filter vs model vs noise

filter_error	pure_error
4.373575084042279	15.774612573664305

Вывод

В качестве заключения можно перечислить достоинства и нодостатки алгоритма фильтра Калмана:

Плюсы:

- точность (в случае линейных моделей и гаусовских ошибок, фильтр Калмана - оптимальный)
- возможность разработки фильтра для отслеживания состояния систем со сложной динамикой
- рекурсивность: которая позволяет алгоритму работать в реальном времени
- распространенность: ввиду ппулярности алгоритма, для многих задач Калмановсой фильтрации уже сущестует решение
- слияение данных от нескольких датчиков: здесь это не было показано, но фильтр может объединять результат предсказания с более чем одним датчиком.

Минусы:

- Сложность посторения фильтра: чем сложнее система, там сложнее будет построение фильтра и настройка его параметров.
- Алгоритм может разойтись: это связанно с тем, что мы вычисляем ковариационные матрицы(которые по определению симметричны), которые могут ввиду неточности вычислений таковыми не оказаться, это влечет расхождение алгоритм.
- Спецефическое свойство фильтра Калмана: может перестать доверять измерениям(если тот окажется очень шумным), и начать выдаваь оценку, руководствуясь только моделью. Чтобы этого не произошло, нужно правильно настраивать параметры фильтра.

Что мы получили в итоге: фильтр Калмана позволяет получать оценки переменых состояния системы, с учетом специфики модели. Не все фильтры позволяют отслеживать переменные состояния(которые меняются со временем). Это в совокупности, с тем что фильтр умеет объединять данные с разых датчиков и возможностью работы в реальном времени, делает алгоритм Калмана основой многих технических систем.

ссылка на репозиторий с кодом:

https://github.com/CorporalCleg/yet_another_kf.git

Источники и литература:

1. Jupyter notebooks про фильры Калмана https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python
2. Вики - https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%BB%D1%8C%D1%82%D1%80_%D0%9A%D0%B0%D0%BB%D0%BC%D0%B0%D0%BD%D0%B0
3. Хабр про фильтр -> https://habr.com/ru/companies/singularis/articles/516798/
4. Хабр про YOLO -> https://habr.com/ru/articles/514450/