Kalman-Filter mal ganz einfach.

Der digitale Filter von Rudolf E. Kálmán dient der Beseitigung von Störungen, die sich in einer Folge von Messwerten befinden.
Die zugehörige Theorie ist sehr komplex, so dass ein Umsetzen des Algorithmus durchaus Schwierigkeiten bereiten kann.

Der Kalman-Filter stellt im wesentlichen einen Tiefpass dar, der durch Parameter beeinflusst wird.

Im Netz findet man eine Reduzierung eines mir nicht näher bekannten Programmes zum Kalman-Filter auf eine Eingangsvariable bzw. einen Messquelle unter Vernachlässigung weiterer Parameter.

Marko Hoerner hat den zugehörigen Code 2009 in der Programmiersprache BASCOM veröffentlicht. Hier der wesentliche Teil:

```
For N = 1 To Length
  Z(n) = Noisysen(n)
Next

Pmin1 = 0

Tempsingle_1 = Pmin1 + R                    'K=Pmin1/(Pmin1+R)
K = Pmin1 / Tempsingle_1

Tempsingle_1 = Z(1) - Noisysen(1)
Tempsingle_1 = K * Tempsingle_1
X_hat(1) = Tempsingle_1 + Noisysen(1)    'X_hat(1)=K*(Z(1) - Noisysen(1))+Noisysen(1)

P = 1 - K
P = P * Pmin1
Pmin = P + Q                                'Pmin=(1-K)*Pmin1+Q
X_hat_min(1) = X_hat(1)

For N = 2 To Length
  K = Pmin + R
  K = Pmin / K
  N2 = N - 1
  X_hat(n) = Z(n) - X_hat_min(n2)
  X_hat(n) = K * X_hat(n)
  X_hat(n) = X_hat_min(n2) + X_hat(n)

  P = 1 - K
  P = P * Pmin
  Pmin = P + Q

  X_hat_min(n) = X_hat(n)

Next N
```

Die ursprünglichen Messwerte befinden sich in Noisysen( ) und die bereinigten oder gefilterten Messwerte in X_hat_min( ).

Die Parameter R und Q bestimmen den Grad der Filterung. Erste günstige Werte sind R=10 und Q=1. Anzumerken ist, dass in der hier angegebenen Form des Filters gleiche Ergebnisse erzielt werden, wenn der Quotient von R / Q jeweils gleich ist.

In meiner Anwendung in einem Echtzeitsystem, der Regelung eines selbst balancierenden Fahrzeuges, ist in einer Reihe von Messwerten eine Filterung oder Korrektur des letzten gemessenen Wertes in einer Reihe von mehreren Messwerten erforderlich. Dabei ist der Verlauf der vorherigen Messwerte zu beachten.

Verschiebt man in Noisysen( ) alle Messwerte um einen Platz entsprechend

```
For i=1 to n-1
    Noisysen(i) = Noisysen(i+1)
Next i
```

und besetzt Noisysenn(n) mit dem jüngsten Messwert, so liefert der Kalman-Filter in seinen gefilterten Werten einen letzten und somit jüngsten Wert, der unter Berücksichtigung aller vorherigen Werte entstanden ist. Dieser letzte gefilterte Wert kann dann für weitere Steuerungsaufgaben weiterverwendet werden.

Im obigen Algorithmus führen die Zeilen vor der Laufschleife zu

Pmin=1 und X_hat_min(1) = Noisysen(1) , *so dass dieser Teil fast vollständig entfallen kann.*

Björn Wittkowski hat die Laufschleife analysiert und dies vereinfacht dargestellt, wie unten zu sehen ist. Insgesamt entstand nachfolgender Code, wobei Y der jüngste Messwert ist.

```
'-------  Verschieben der alten Messwerte und Ergänzen des jüngsten Messwertes ---------

 For N = 1 To Length - 1                    'Verschieben der alten Daten
     N1 = N + 1
     Messwerte(n) = Messwerte(n1)           'Messwerte( ) verschieben
 Next N

Messwerte(length) = Y                       'aktuellen letzten Messwerte ergänzen

Gefilterter_letzer_wert = Messwerte(1)      'als Startwerte vorbesetzen
Pmin = 1                                    'Pmin konvergiert im Laufe der Messreihe

'----------------  vereinfachter Schreibweise von Björn für Kalman-Filter -----------------

For N = 1 To Length
    Messwert = Messwerte(n)                      ' Hole einen Messwert aus der Messreihe
    K = Pmin + R
    K = Pmin / K                            'K=Pmin/(Pmin+R)

    Gefilterter_wert = Messwert - Gefilterter_letzer_wert
    Gefilterter_wert = K * Gefilterter_wert
    Gefilterter_wert = Gefilterter_letzer_wert + Gefilterter_wert

    P = 1 - K
    P = P * Pmin
    Pmin = P + Q                            'Pmin=(1-k)*Pmin+Q
    Gefilterter_letzer_wert = Gefilterter_wert
```

```
  Next N

  Y = Gefilterter_letzer_wert                'Gefilterten Wert weiterverwenden
```

Der Zugriff auf indizierte Variable wurde dabei wesentlich reduziert.

Dieser Filter wurde für Length=24 Messwerte mit Q=1 und R=20 realisiert.
Die gemessene Rechenzeit für einen Durchlauf dieses Algorithmus beträgt 4,25ms bei einer Taktfrequenz von 16MHz auf einem Atmega32.

Die vorherige Version benötigte für die gleiche Arbeit 4,5ms. Man findet also keine wesentliche Zeitersparnis.
Dies erklärt sich damit, dass die Rechenoperationen für Gleitkommazahlen in der Schleife nicht reduziert werden konnten. Durch sie wird der wesentliche Teil der Rechenzeit bestimmt.

In den Zeilen zur Berechnung von Gefilterter_wert *erkennt man auch den Algorithmus eines Standard-Tiefpasses wieder, bei dem sich hier lediglich der Parameter K im Laufe eines Durchlaufes ändert.*

Wolfgang Schmidt
183600@gmail.com