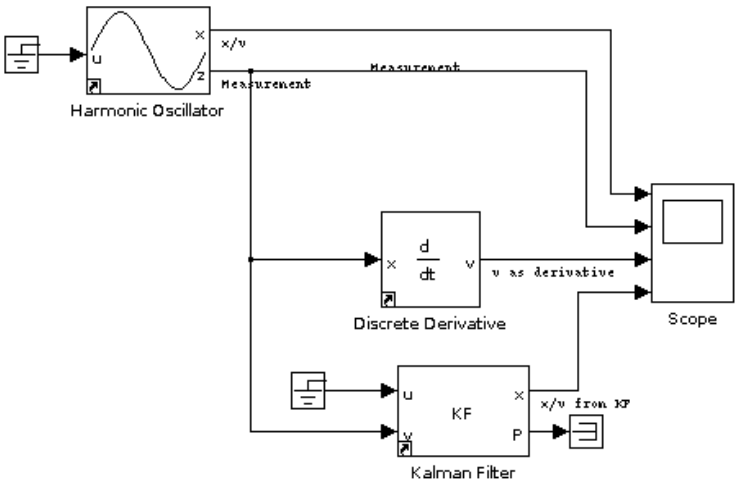
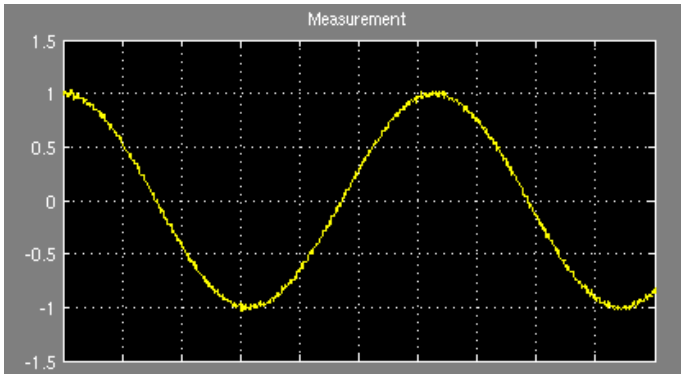
Simulation von Kalmanfiltern mit Simulink

Einleitung

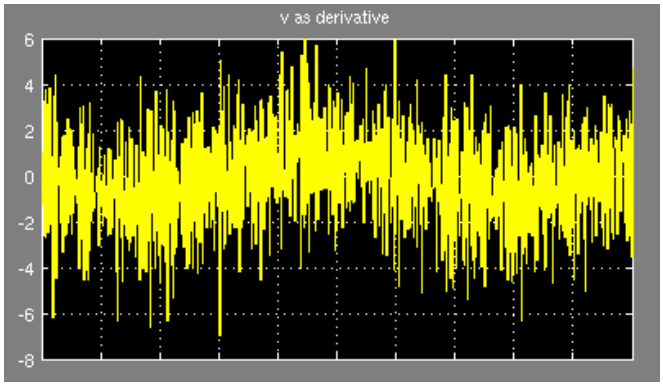
- Worum geht es:
  - Simulink einsetzen können, um Messprobleme mit Kalman-Filter zu simulieren
- Was wird nicht behandelt:
  - Mathematik des Kalman-Filters
    aber Angabe der Grundgleichungen und Motivation
  - fortgeschrittene Anwendungen aus dem Vermessungsbereich
    aber einfache Grundideen der Trägheitsnavigation
- Beispiel:
  - Modell schwingX.mdl



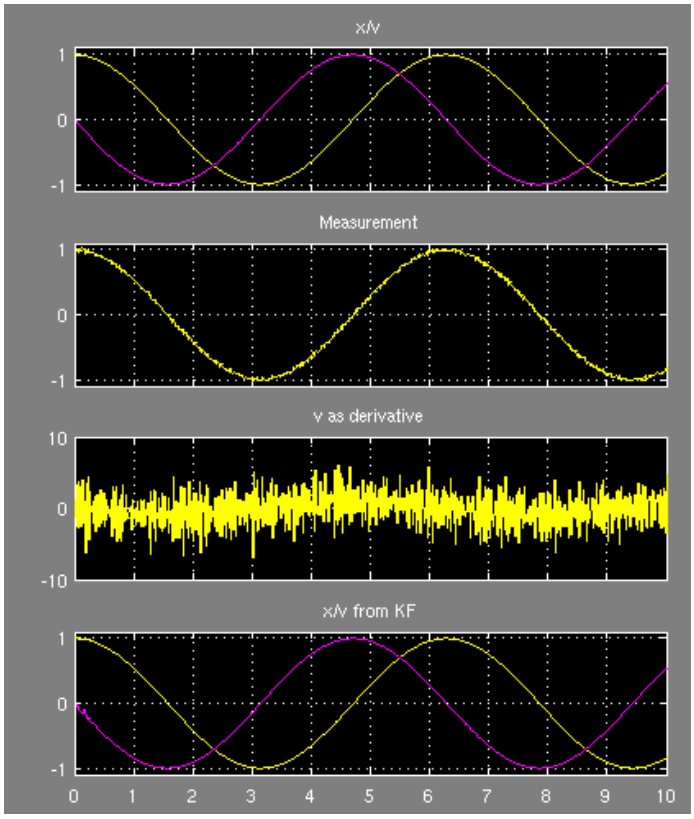
  - benutzt Blockbibliothek kalmanlib.mdl
  - Geschwindigkeit eines harmonischen Schwingers soll bestimmt werden
  - Messung liefert nur Ort, leicht verrauscht



  - naive Rekonstruktion von v als (diskreter) Ableitung liefert Unsinn



- Rekonstruktion mit Kalman-Filter ergibt sehr gute Übereinstimmung



Simulink-Bausteine für eine Kalman-Filter-Bibliothek

Modell für einen harmonischer Oszillator

- Harmonischer Oszillator:
  - beschreibt harmonische Schwingungen x(t)
  - angetrieben durch äußere Kraft F(t)
  - Bewegungsgleichung

$$\ddot{x} + \omega_0^2 x = u(t)$$

  - Parameter

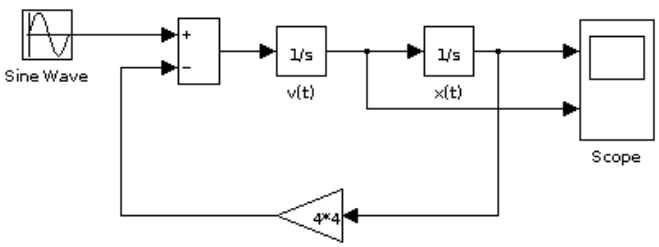
ω_0	Kreisfrequenz 2πf
x_0	Anfangsauslenkung
v_0	Anfangsgeschwindigkeit
u(t)	äußere Beschleunigung F(t)/m

- Grundidee von Simulink:
  - Modell besteht aus Blöcken mit Ein- und Ausgängen
  - Blöcke werden durch Leitungen verbunden (natürlich Ausgang mit Eingang)
  - auf den Leitungen liegen Signale, die in bestimmtem Zeitschritten aktualisiert werden
    - diskrete Systeme: Zeitpunkte festgelegt (meistens mit fester Frequenz)
    - kontinuierliche Systeme (Differentialgleichungen): Zeitpunkte werden vom DGL-Solver bestimmt

- Aufbau des Modells mit Simulink:
  - Vorgehensweise
    - DGL nach höchster Ableitung auflösen (hier $\ddot{x}$)

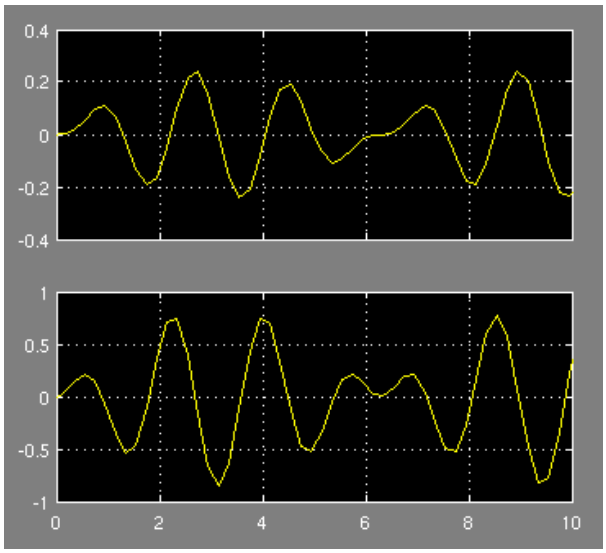
$$\ddot{x} = u(t) - \omega_0^2 x$$

    - Signal x festlegen
    - Integrator davor → am Eingang liegt $\dot{x}$
    - Integrator vor $\dot{x}$→ am Eingang liegt $\ddot{x}$
    - durch geeignete Schleife Differentialgleichung zusammenbauen
  - Modell model_harmonic.mdl

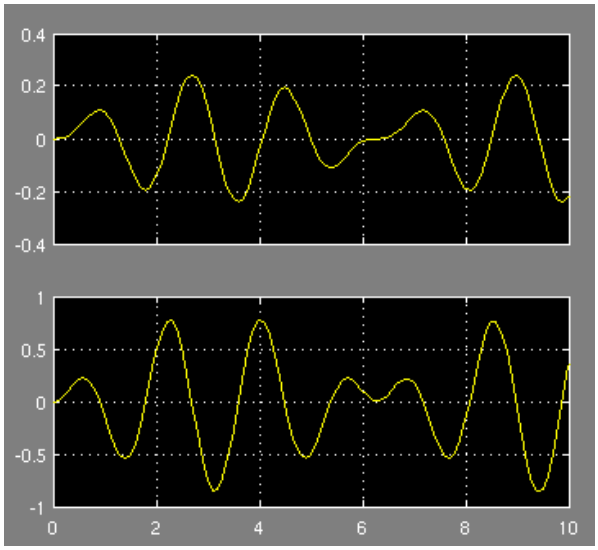


  - Details zum Aufbau

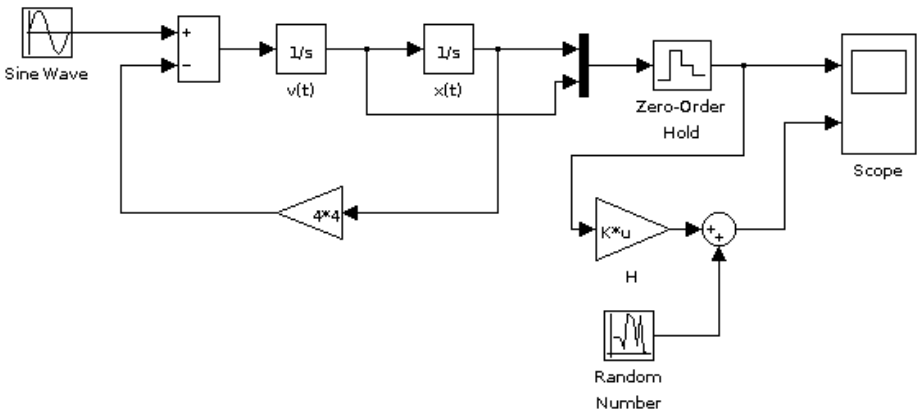
        - Blöcke aus Simulink-Bibliothek holen
        - Gain-Block mit Format/Flip Block spiegeln
        - Oscilloscope öffnen, Parameters öffnen, Number of axes = 2
        - Leitungen verlegen, Abzweigungen mit <STRG>-Klick
        - Oscilloscope etwas vergrößern
        - Sine Wave öffnen, Frequency = 3
        - Gain öffnen, Gain = 4*4
        - Format/Hide Name bei Subtract und Gain
        - Namen der Integratoren auf v(t) bzw. x(t) ändern
    - Simulation starten mit Simulation/Start
        - im Osci auf "Fernglas" klicken (optimale Skalenwahl)
        - → Bild ist eckig



    - Simulation/Configuration Parameters öffnen
        - Solver/Max Step Size = 0.1 →



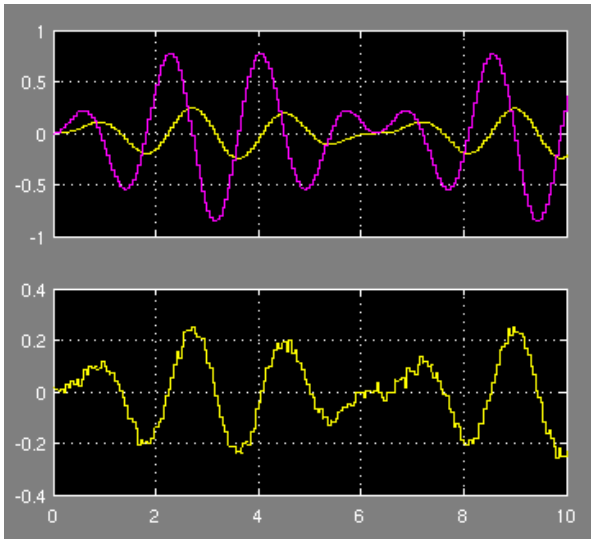
- Oszillator mit Messungen:
    - Modell soll ergänzt werden um Messung incl. Messrauschen
    - Ergebnis model_harmonic2.mdl



- Aufbau
  - Blöcke hinzufügen: Mux, Zero Order Hold, Gain, Sum, Random Number
  - Leitungen zum Osci lösen und neue Signalleitungen legen
  - Block-Parameter

Block	Parameter	Wert
Zero Order Hold	Sample time	0.05
Gain	Gain	[1,0]
Gain	Multiplication	Matrix(K*u)
Random Number	Variance	2e-4
Random Number	Sample time	0.05

- Ergebnis



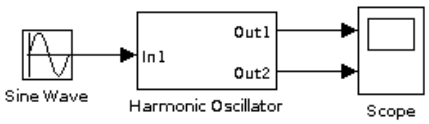
- Funktionsweise
  - Multiplexer fasst x und v zu einer Vektorleitung zusammen mit Signal $(x, v)^T$
  - Zero Order Hold erzeugt daraus Samplewerte im Abstand dt = 0.05 (Oszi oben)
  - Gain macht aus dem Vektorsignal durch Multiplikation mit (1 0) wieder x

$$\begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix} = x$$

  - Random Number erzeugt zu jedem Zeitschritt (Sample Time) normalverteilte

Zufallszahl mit gegebenem Mittelwert (0) und Varianz (0.0002)

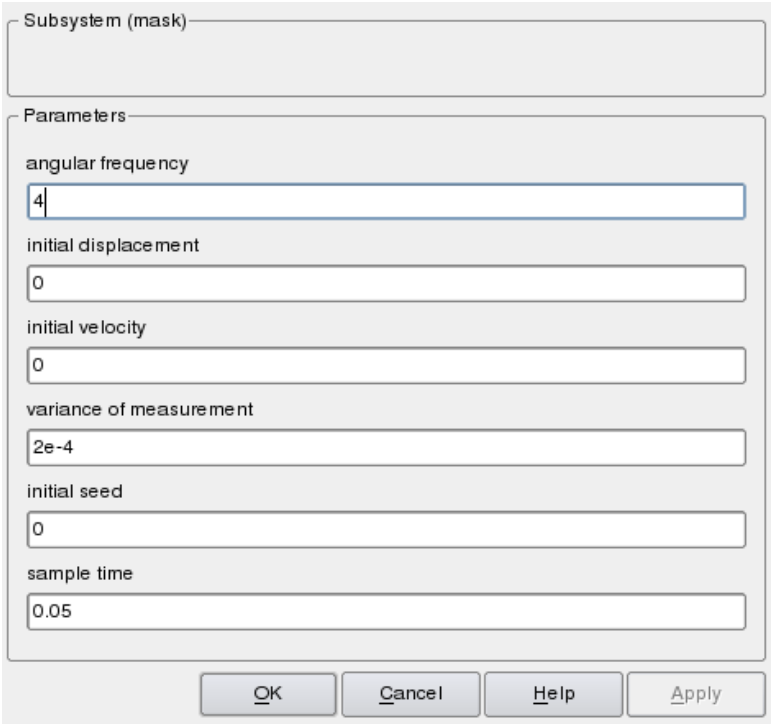
- Erzeugen eines Bibliotheksblocks:
  - Ziel: Erzeugen eines universell einsetzbaren Blocks "Harmonischer Oszillator"
  - 1. Schritt: Erzeugen eines Subsystems
    - Markiere mit der Maus ganzes Modell außer Sine Wave und Osci
    - Edit/Create Subsystem
  - Ergebnis (nach Umbenennung und leichter Verschiebung)



  - 2. Schritt: wichtige Parameter direkt zugreifbar machen
    - Edit/Mask Subsystem/Parameters
    - Eingabe von Beschriftungen und Variablennamen

Beschriftung	**Variablenname**
angular frequency	omega0
initial displacement	x0
initial velocity	v0
variance of measurement	R
initial seed	seed
sample time	dt

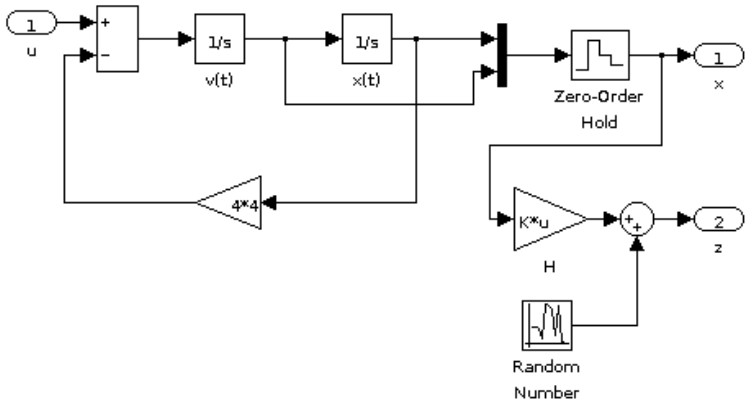
    - Doppelklick → Parametermaske öffnet sich
    - Werte eintragen



  - Hinweis:

- initial seed initialisiert den Zufallszahlengenerator
- anderer Wert → andere Zahlenfolge

- 3. Schritt: Variablen ins Submodell eintragen
  - Subsystem öffnen mit Edit/Look Under Mask
  - Eingang In1 umnennen zu u
  - Ausgänge Out1/Out2 umnennen zu x/v



  - Variablen eintragen:

Block	Parameter	alter Wert	neuer Wert
Gain	Gain	4*4	omega0*omega0
Integrator x(t)	Initial condition	0	x0
Integrator v(t)	Initial condition	0	v0
Random Number	Variance	2e-4	R
Random Number	Initial Seed	0	seed
Random Number	Sample time	0.05	dt
Zero Order Hold	Sample time	0.05	dt

- 4. Schritt: Block dokumentieren
  - Edit/Edit Mask/Documentation
  - Mask Type

```
Harmonic oscillator
```

  - Mask description

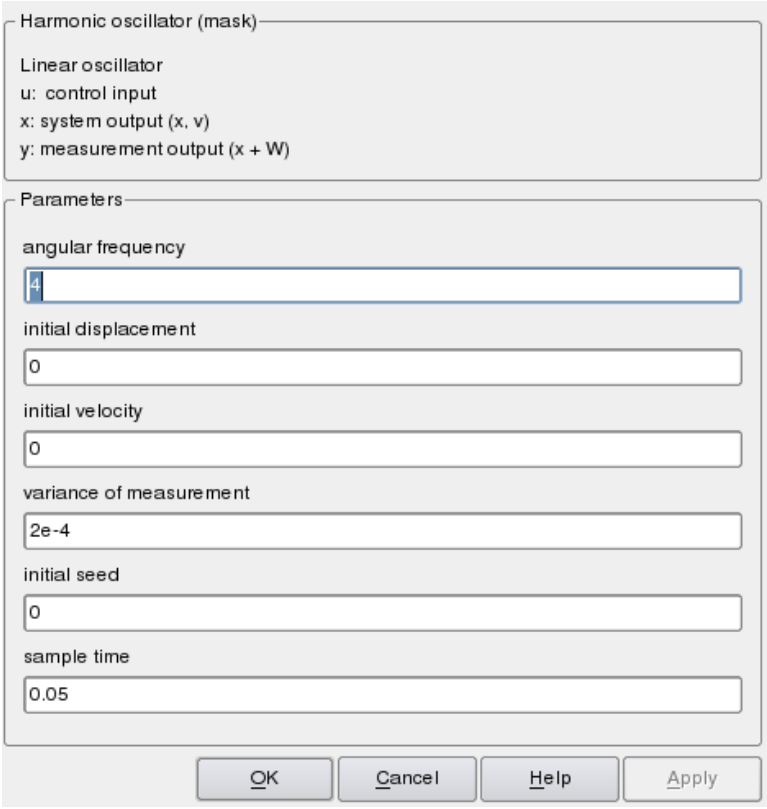
```
Linear oscillator
u: control input
x: system output (x, v)
y: measurement output (x + W)
```

  - Mask help (erscheint bei "Help" in der Matlab-Dokumentation)

```
Harmonic oscillator that is sampled with sample time dt
<ul>
<li>System output x: vector of position and
velocity</li>
<li>Measurement output y: measurement of position x,
including a normal random noise</li>
```

```
<li>Input u: control input, defining an external
driving force</li>
</ul>
```

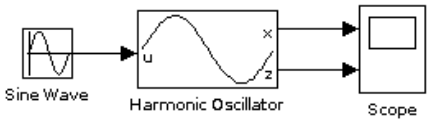
- Parameter-Maske damit



- 5. Schritt: Block verschönern
  - Edit/Edit Mask/Icon/Drawing commands
  - Erzeugen eines Icons mit Matlab-Zeichen-Kommandos, z.B.

```
plot(0:0.1:2*pi,sin(0:0.1:2*pi))
port_label('input', 1, 'u')
port_label('output', 1, 'x')
port_label('output', 2, 'z')
```

  - Modell model_harmonic3.mdl damit



Zeitdiskrete Modelle

- Lineare diskrete Systeme:
  - Idee: System direkt diskret modellieren statt kontinuierliches System zu sampeln
  - lineares Modell mit innerem Zustandsvektor x_k, äußerer Einflussgröße u_k und Rauschen w_k (Systemrauschen):

$$x_{k+1} = Ax_k + Bu_k + w_k$$

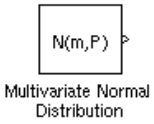
  - w_k ist Gaußsches Rauschen mit Mittelwert 0 und Kovarianzmatrix Q, geschrieben

$$w_k \in \mathcal{N}(0, Q)$$

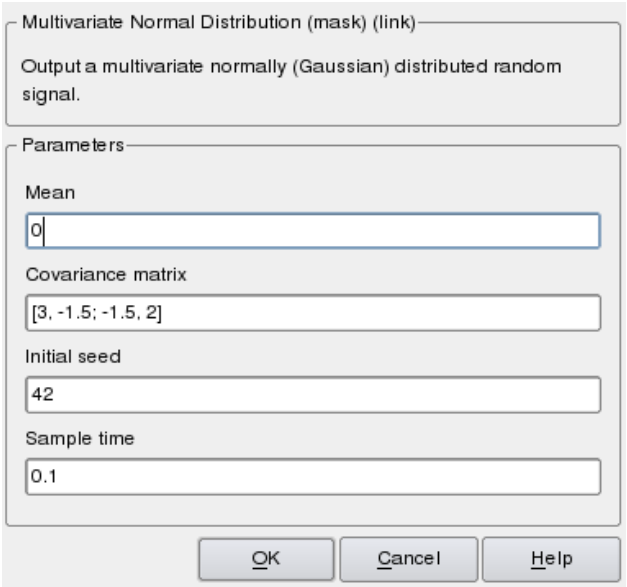
  - häufiger Spezialfall: kein u ⇔ B = 0
  - x nicht direkt beobachtbar, stattdessen Messgrößen z, die linear von x abhängen und ebenfalls Gaußsches Rauschen (Messrauschen) aufweisen:

$$z_k = Hx_k + v_k, \qquad v_k \in \mathcal{N}(0, R)$$

  - Dimensionen der Größen (Zeilenzahl x Spaltenzahl)
    - x(nx1), A(nxn), Q(nxn)
    - z(lx1), H(lxn), R(lxl)
- Modell für Gaußsches Rauschen:
  - eigener Subblock



  - Parameter über Maske vorgebbar



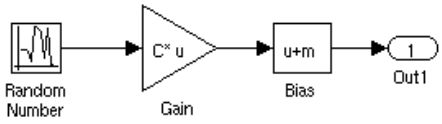
  - zur Berechnung eines Vektors normalverteilter Zufallszahlen mit Kovarianz P bestimme C mit

$$C^T C = P$$

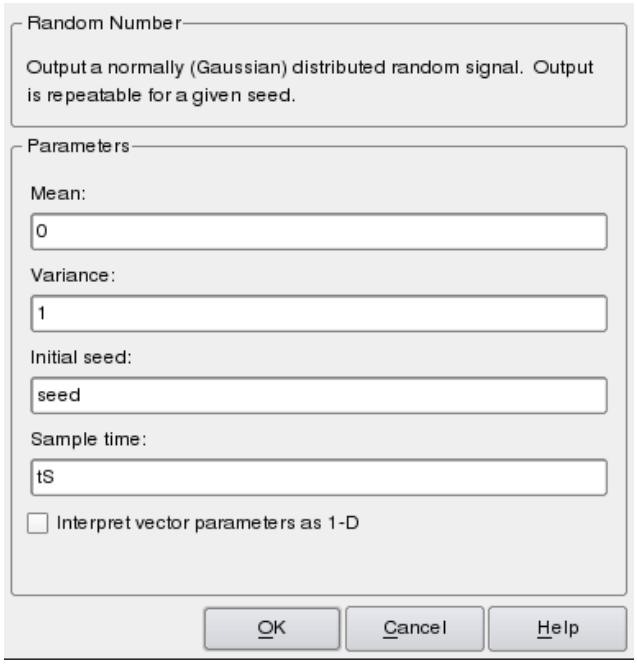
- in Matlab mit

```
[C, res] = sqrtm(P);
```

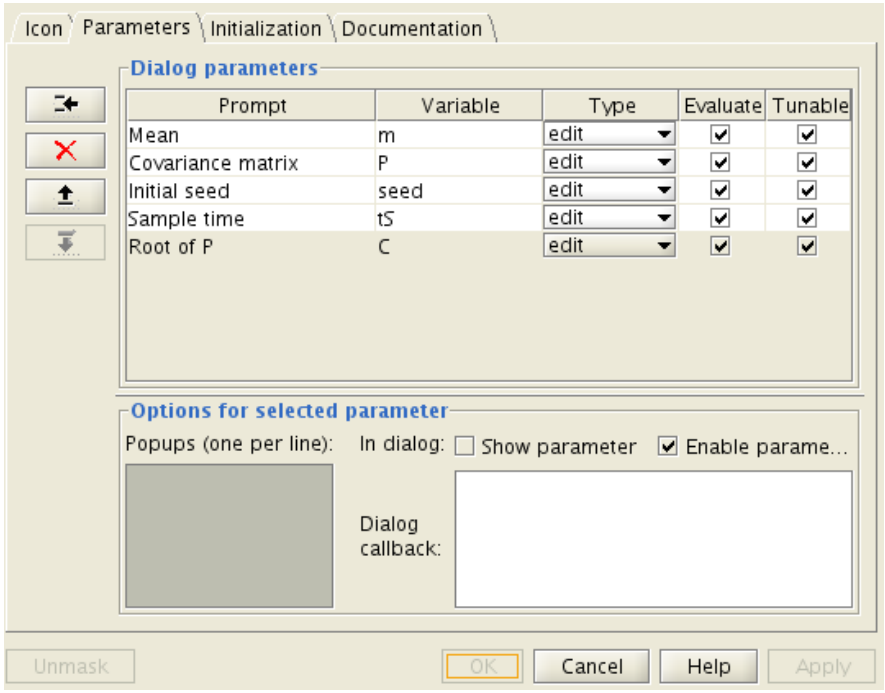
- Simulink-Modell dann



- Parameter von Random Number

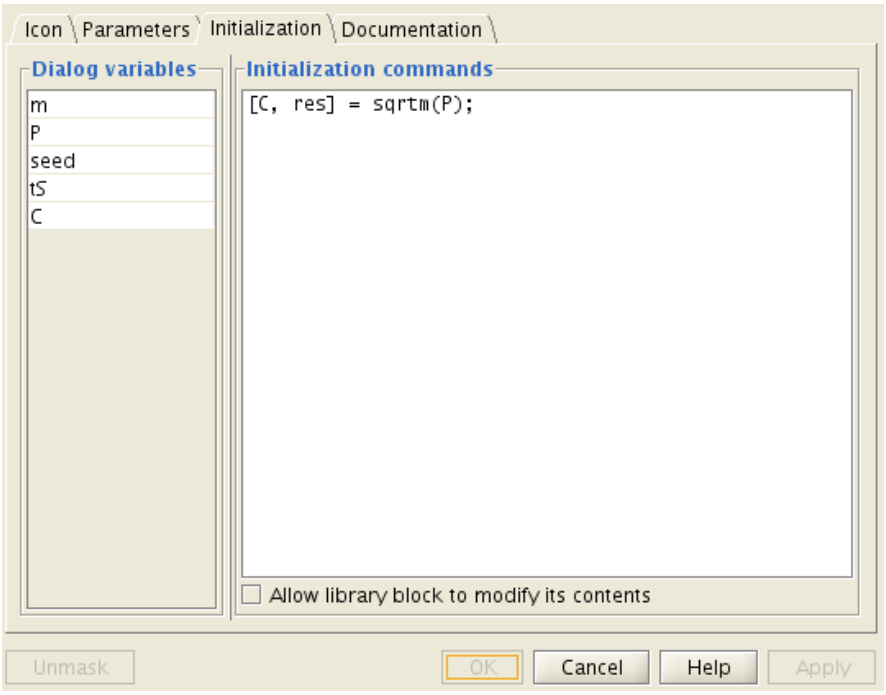


- C wird als zusätzlicher Parameter in Mask Editor/Parameters aufgenommen



  - "Show parameter" deaktiviert → C erscheint nicht in der Maske

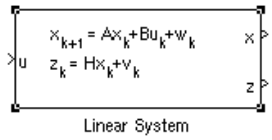
- automatische Berechnung von C aus Q in Mask Editor/Initialization



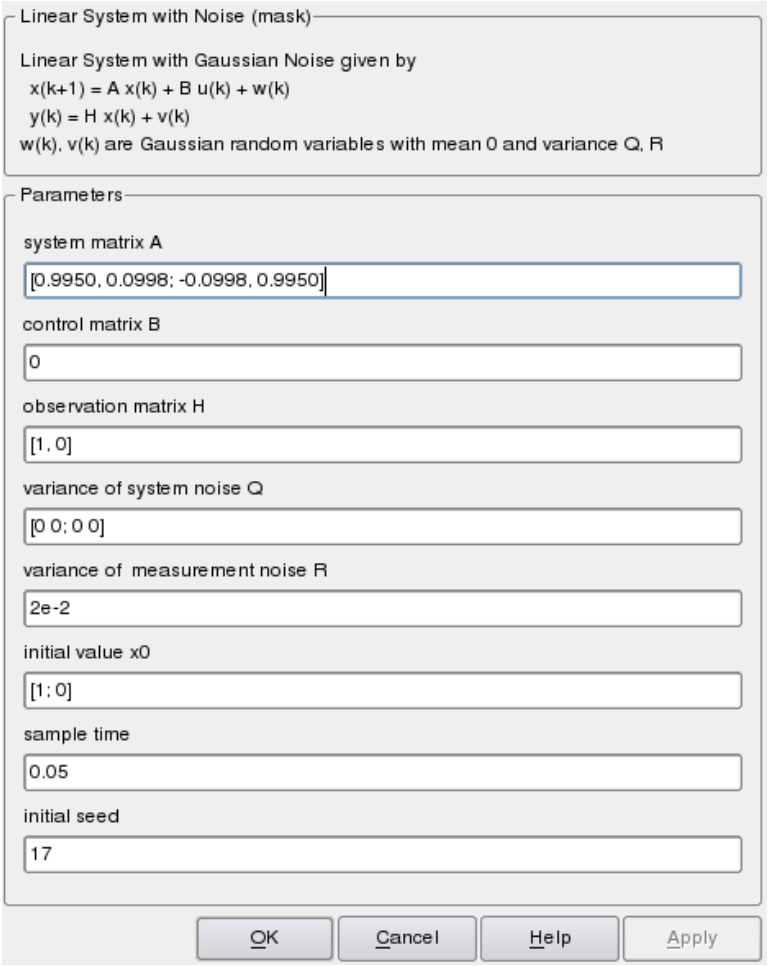
- Beschriftung des Blocks mit Kommando in Mask Editor/Icon

```
disp('N(m,P)')
```

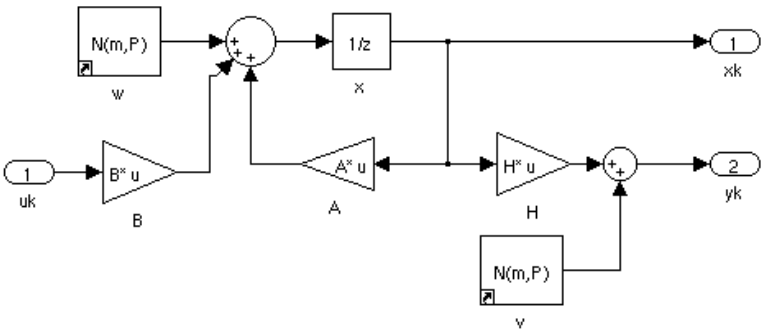
- Modell für lineares diskretes System:
  - Subblock



  - Parameter



- Aufbau



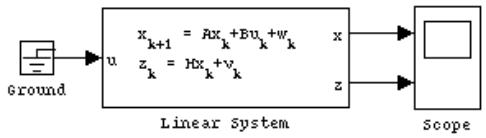
- Block Unit Delay (1/z)
  - gibt Eingangswert beim nächsten Sample-Zeitpunkt aus
  - hat Anfangswerte für Start
  - entspricht Integrator bei kontinuierlichen Systemen
- Problem mit Zufallszahlen
  - gleiche seed → gleiche Werte → unerwünschte Korrelationen im Modell
- Lösung
  - Block w hat seed1, Block v hat seed2
  - Berechnung in Mask Editor/Initialization

```
N = size(Q, 1);
M = size(R, 1);
seed1 = (seed:(seed+N-1))';
seed2 = ((seed+N):(seed+N+M-1))';
```

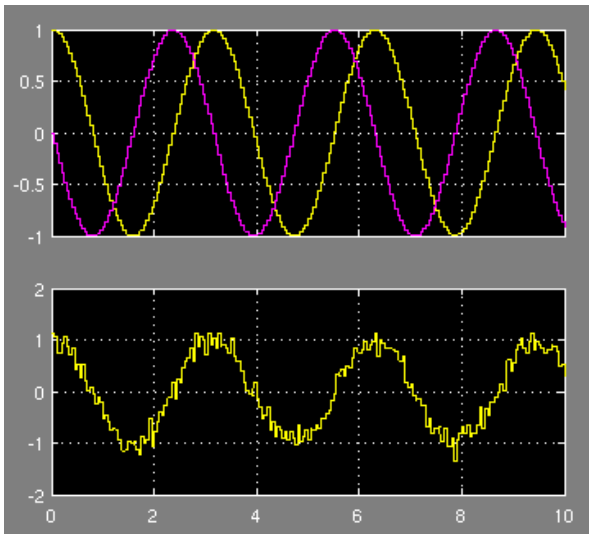
- Achtung:
  - in Modellen mit mehreren Linear Systems seed so wählen, dass keine Überschneidungen auftreten!
- Beschriftung des Blocks in Mask Editor/Icon

```
text(0.15, 0.65, 'x_{k+1} = Ax_k+Bu_k+w_k',
'texmode', 'on', 'horizontalAlignment', 'left')
text(0.15, 0.35, 'z_k = Hx_k+v_k',
'texmode', 'on', 'horizontalAlignment', 'left')
port_label('output', 1, 'x')
port_label('output', 2, 'z')
port_label('input', 1, 'u')
```

- Beispiel-Lauf:
  - Modell model_linear_system.mdl mit Standard-Parametern



  - Ergebnis



  - Ground-Block
    - schickt 0
    - verhindert Warnung über nicht angeschlossenen Eingang
  - System enthält keine kontinuierlichen Zustände, braucht Solver "discrete"
- Harmonischer Oszillator als lineares diskretes System:
  - betrachten nur den Fall u = 0
  - Umschreiben der DGL auf 2d-System 1. Ordnung

$$\mathbf{x} = \begin{pmatrix} x \\ v \end{pmatrix}$$

$$\dot{\mathbf{x}} = \begin{pmatrix} v \\ -\omega_0^2 x \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{pmatrix} \begin{pmatrix} x \\ v \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 1 \\ -\omega_0^2 & 0 \end{pmatrix} \mathbf{x}$$

$$=: \mathbf{F}\mathbf{x}$$

- Lösung der DGL bei gegebenem Anfangsvektor x_0

  $$\mathbf{x}(t) = \mathbf{A}\mathbf{x_0}$$

  mit

  $$\mathbf{A} = e^{\mathbf{F}t} = \begin{pmatrix} \cos\omega_0 t & \frac{1}{\omega_0}\sin\omega_0 t \\ -\omega_0 \sin\omega_0 t & \cos\omega_0 t \end{pmatrix}$$

- Werte nur in Zeitabständen Δt

  $$x_k := x(k\Delta t)$$

  - kann direkt als lineares diskretes System geschrieben werden

    $$x_{k+1} = \mathbf{A}x_k$$

- Beispielwerte

  $$\omega_0 = 1, \Delta t = 0.1 \qquad \Rightarrow \qquad \mathbf{A} = \begin{pmatrix} 0.9950 & 0.0998 \\ -0.0998 & 0.9950 \end{pmatrix}$$

Kalman-Filter

- Prinzip des Kalman-Filters:
  - macht Vorhersage für nächsten Zustandswert x_{k+1} und Genauigkeit P_{k+1} (Kovarianzmatrix P des Filters)
  - geht von linearem diskreten Modell aus

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_{k-1} + w_{k-1}, \qquad w_k \in \mathcal{N}(0, Q) \\ z_k &= Hx_k + v_k, \qquad v_k \in \mathcal{N}(0, R) \end{aligned}$$

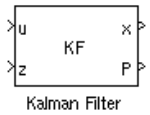
  - bekommt außerdem Messwerte z_k
  - 1. Schritt: Vorhersage aufgrund des Modells (Prädiktor)

$$\begin{aligned} x_k^- &= Ax_{k-1} + Bu_{k-1} \\ P_k^- &= AP_{k-1}A^T + Q \end{aligned}$$

  - 2. Schritt: Korrektur aufgrund des Messwerts (Korrektor)

$$\begin{aligned} K_k &= P_k^- H^T (HP_k^- H^T + R)^{-1} \\ x_k &= x_k^- + K_k(z_k - Hx_k^-) \\ P_k &= (1 - K_k H)P_k^- \end{aligned}$$

  - Schätzwerte für die Anfangswerte x_0 und P_0 müssen vorgegeben werden
  - Kalmanfilter mathematisch definiert durch
    - erwartungstreu (reproduziert Erwartungswerte)
    - Erwartungswert der quadratischen Abweichung zwischen Schätz- und wahrem Wert minimal
  - Formeln sind plausibel:
    - R groß → K klein → $x_k = x_k^-$ (Modell gibt Ausschlag)
    - R = 0 → $K = H^{-1}$ → $x_k = H^{-1} z_k$ (Messwert gibt Ausschlag)
- Simulink-Modell des Kalman-Filters:
  - Subblock



  - Parameter

Parameters

system matrix A

[cos(0.01), sin(0.01); -sin(0.01), cos(0.01)]

control matrix B

0

observation matrix H

[1,0]

variance of system noise Q

[0 0; 0 0]

variance of measurement noise R

[2e-4]

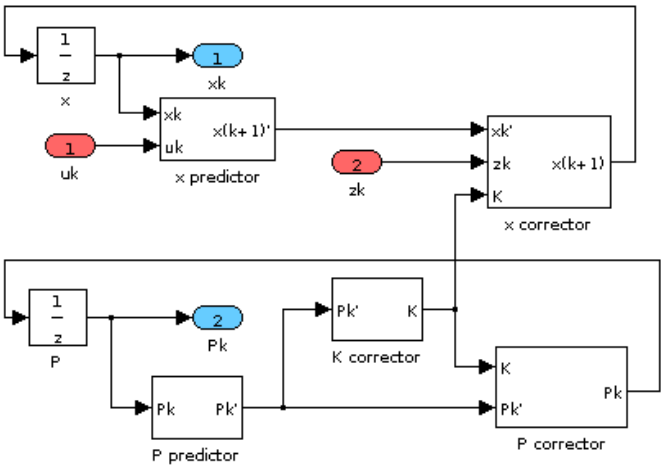
estimated initial value x0

[1; 0]

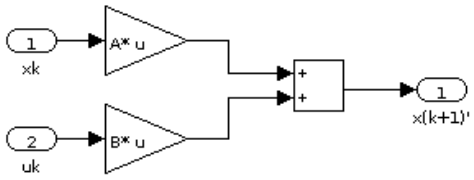
estimated initial filter variance P0

[10, 0; 0, 10]*1e-3

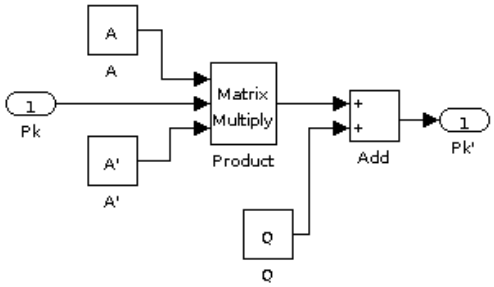
- Aufbau



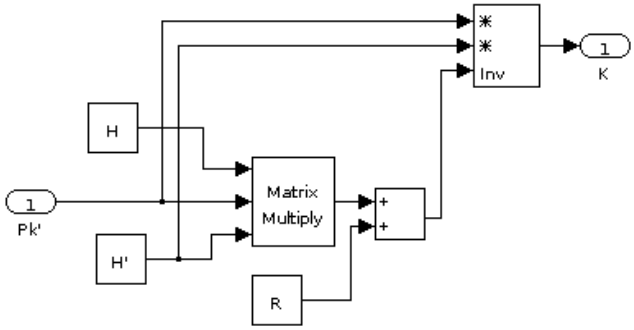
- 5 Unterblöcke implementieren die Formeln
  - x predictor



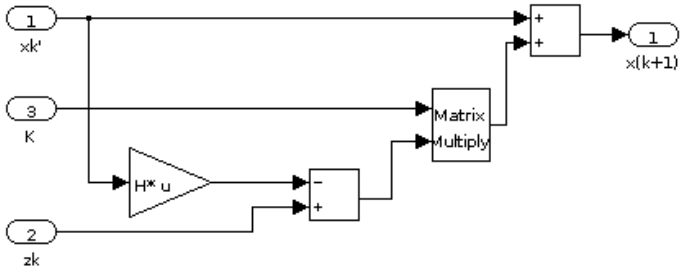
  - P predictor



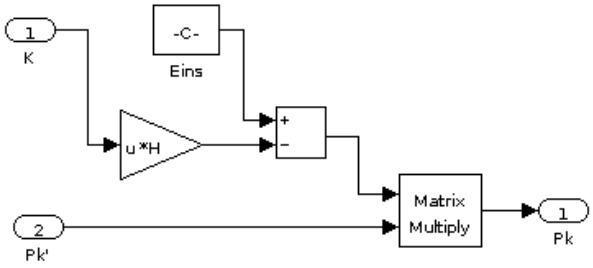
  - K corrector



- x corrector



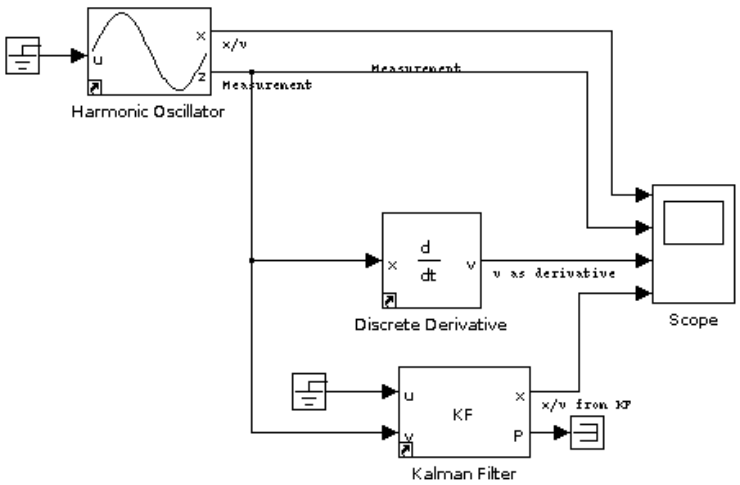
- P corrector



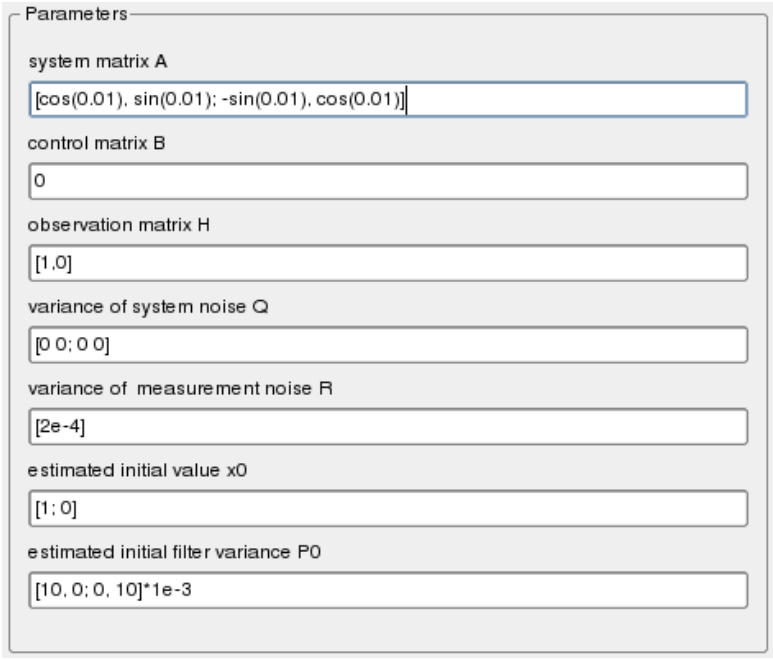
  - Achtung:
    - 1 in Formel für P_k ist die Einheitsmatrix
    - im Simulink in einer Konstanten mit Constant Value

      `eye(size(P0))`

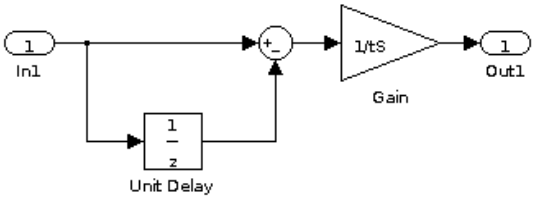
- Startbeispiel, 2. Durchgang:
  - Aufbau klar



- Modell ist harmonischer Oszillator in diskreter Version
- Kalmanfilter hat die entsprechenden Systemparameter



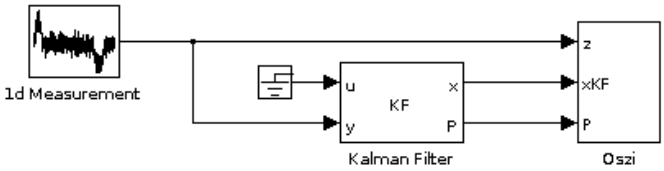
der Vollständigkeit halber die diskrete Ableitung



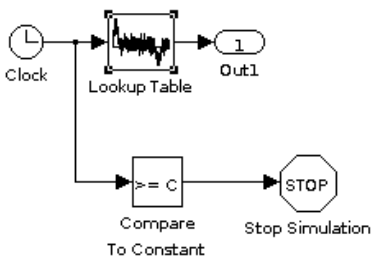
Anwendung: Grundideen der Trägheitsnavigation

Auswertung von Messwerten

- Grundprinzip:
  - Bestimmung der Position eines Körper aus Messung der Beschleunigung
  - Grundidee einfach: Beschleunigung zweimal integrieren
  - Problem:
    - Messfehler (Rauschen) addieren sich auf
    - → zu v kommt eine Random-Walk-Komponente hinzu
    - → Abweichung steigt mit der Zeit an
  - Lösung
    - gelegentliche Orts-Messungen einbauen
- Simulink-Modell:
  - Modell ins1.mdl

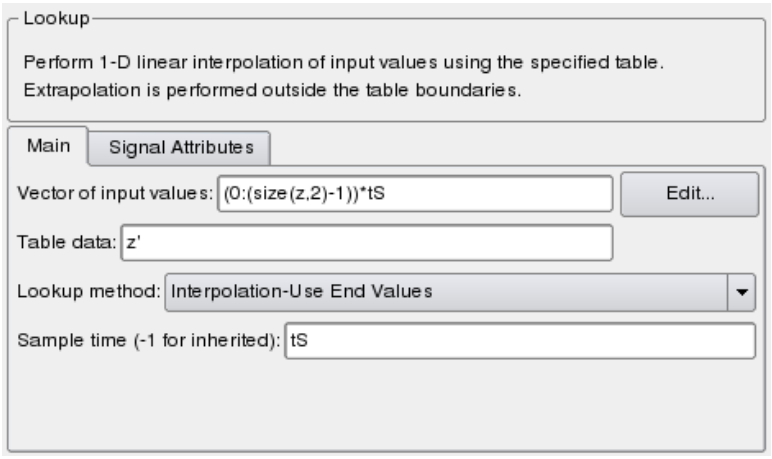


  - 1d Measurement liefert experimentell bestimmte Messwerte für a
  - Kalman-Filter bereitet die Daten auf und bestimmt v und x
  - Oszi zeigt an
    - Messwerte
    - Filterdaten
    - Entwicklung der Filter-Kovarianz
  - Achtung:
    - Stop Time steht auf Inf ("unendlich")
- Mess-Modell:
  - Parameter
    - Vektor mit Messwerten
    - Sample-Zeit
  - Aufbau



  - Block Lookup Table

        - liefert Messwerte als Zeitreihe
        - interpoliert ggf. (hier nicht, da nur Sample-Werte)
        - Parameter



    - Compare To Constant
        - gibt true (bzw. 1) aus, wenn Input ≥ Konstante
        - Wert der Konstanten über Maske, hier

          `size(z,2)*tS`

    - Stop Simulation
        - beendet Simulation, wenn Eingang ≠ 0

- Mathematischer Einschub: Diskretisierung incl. Systemrauschen
    - Ausgangspunkt sei ein lineares kontinuierliches System

      $$\dot{\mathbf{x}} = \mathbf{F}\mathbf{x} + \mathbf{G}\mathbf{w}$$

        - w(t) = Gaußsches weißes Rauschen mit Mittelwert 0 und Kovarianz 1
        - F, G konstante Matrizen
    - Sampeln der x-Werte in festen Intervallen Δt liefert ein lineares diskretes System
        - mit Systemmatrix

          $$\mathbf{A} = e^{F\Delta t}$$

        - und Kovarianzmatrix

          $$\mathbf{Q} = \int_0^{\Delta t} dt' \, e^{\mathbf{F}(\Delta t - t')} \mathbf{G}\mathbf{G}^T \left[e^{\mathbf{F}(\Delta t - t')}\right]^T$$

- Kinematisches Modell:
    - unbekannte Kräfte werden durch Zufallspfad (Random Walk) σ(t) modelliert

      $$a = \ddot{x} = q\sigma(t)$$

    - Ableitung von σ(t) ist weißes Rauschen, daher

      $$\dot{a} = qw(t)$$

    - als System

$$\mathbf{x} = \begin{pmatrix} x \\ v \\ a \end{pmatrix}$$

$$\dot{\mathbf{x}} = \begin{pmatrix} v \\ a \\ qw(t) \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ v \\ a \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ q \end{pmatrix} w(t)$$

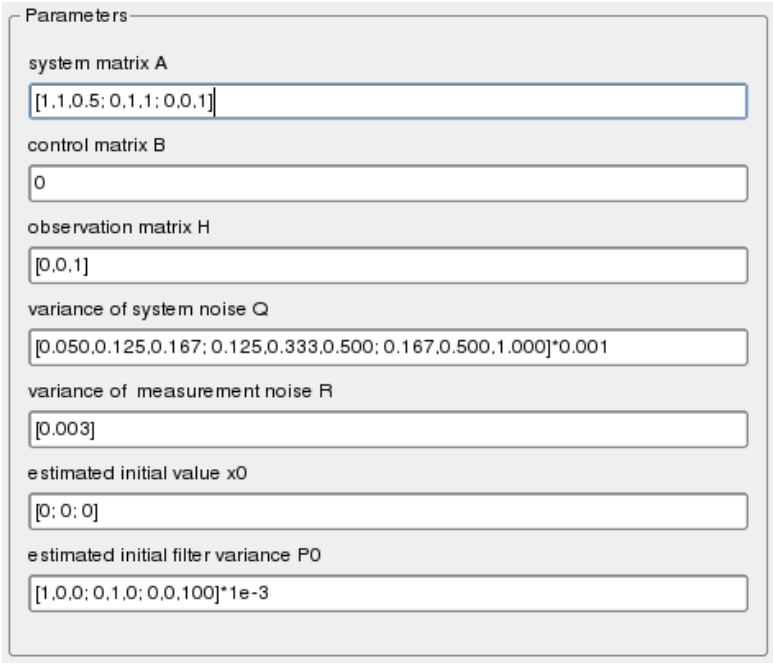
$$=: \mathbf{Fx} + \mathbf{Gw}$$

- Diskretisierung liefert die Systemmatrizen

$$A = \begin{pmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix}$$

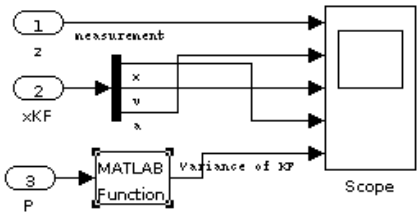
$$Q = \begin{pmatrix} \frac{1}{20}\Delta t^5 & \frac{1}{8}\Delta t^4 & \frac{1}{6}\Delta t^3 \\ \frac{1}{8}\Delta t^4 & \frac{1}{3}\Delta t^3 & \frac{1}{2}\Delta t^2 \\ \frac{1}{6}\Delta t^3 & \frac{1}{2}\Delta t^2 & \Delta t \end{pmatrix}$$

- Kalman-Filter verwendet kinematisches Modell für Δt = 1

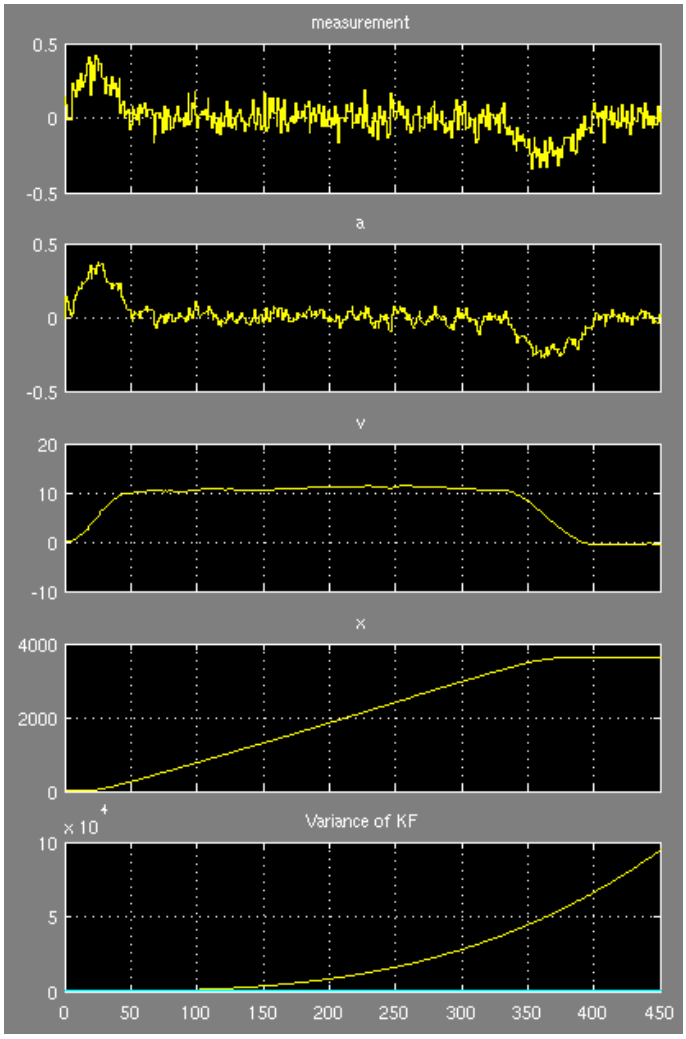


  - Varianz der Stör-Beschleunigung 0.001
  - ungenauere Messung von a mit Varianz 0.003

- Ergebnisse der Simulation:
  - Oszi-Aufbau



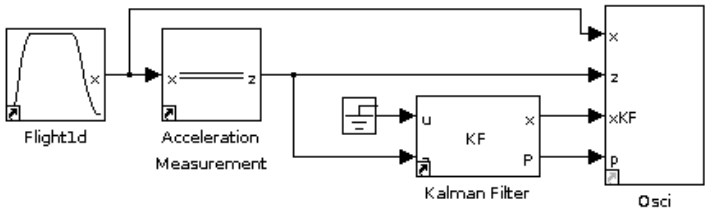
- zeigt Messwerte, a/v/x vom Filter, Filter-Varianz P
- von Matrix P werden mit diag(P) nur die Diagnonalelemente angezeigt
- Oszi-Bild



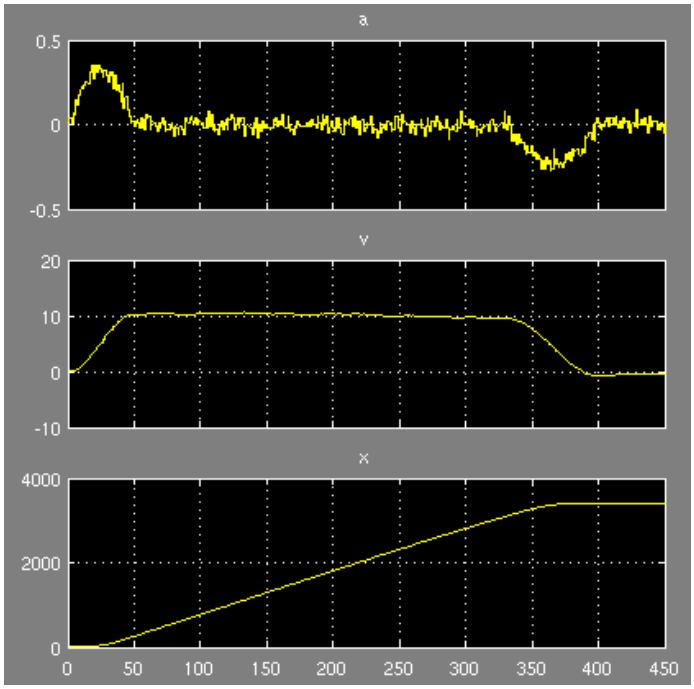
- Interpretation
  - am Anfang Beschleinigung, am Ende Abbremsen, sonst konstante Geschwindigkeit
  - a wird vom Filter leicht geglättet
  - Varianz von x steigt stark an (wie zu erwarten war)
- kein Vergleich mit realen Werten möglich, da nicht vorhanden

Simulation des Flugs

- Erweiterung um Flug-Modell:
  - Modell eines Geradeausflugs mit Start- und Landungsphase
    - erzeugt Messwerte wie in ins1
    - erlaubt Vergleich von Modell- und Filterwerten
  - Simulink-Modell ins2.mdl



  - gleiche Konfiguration des Kalman-Filters
- Flug-Modell:
  - liefert x/v/a mit
    - Startphase (Beschleunigung in vorgegebener Zeit auf Fluggeschwindigkeit)
    - Flugphase (konstantes v)
    - Landephase (Abbremsen in vorgegebener Zeit auf v = 0)
  - Beschleunigung enthält Systemrauschen

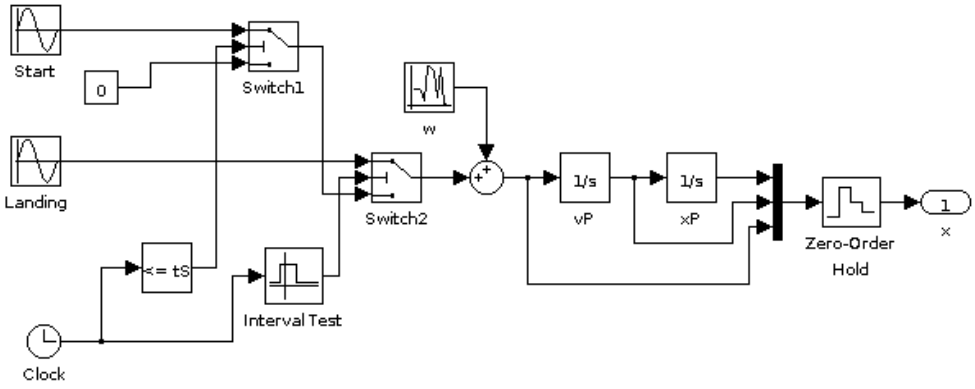


    - aufgrund der a-Schwankungen landet v nicht genau bei 0
  - Parameter

Bezeichnung	Variable
duration of start	tS
duration of cruise	tF
duration of landing	tL

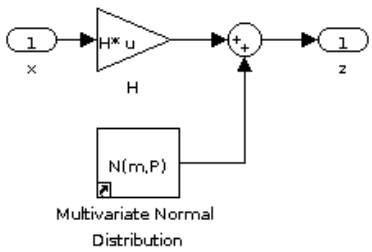
cruise velocity	v0
variance of acceleration	q
sample time	dt
initial seed	seed

- Aufbau



  - Start- und Landephase werden als positive bzw. negative Sinus-Halbwelle modelliert
  - Switch1 schaltet die Startphase durch für t ≤ tS, sonst ist a = 0
  - während der Landephase (tS+tF ≤ t ≤ tS+tF+tL) schaltet Switch2 auf die Landing-Halbwelle
- besondere Details
  - bei Compare und Interval Test sollte Output data type mode auf boolean
  - Parameter von Start
    - Amplitude v0*pi/(2*tS)
    - Frequency pi/tS
  - Parameter von Landing
    - Amplitude -v0*pi/(2*tL)
    - Frequency pi/tL
    - Phase -pi*(tS+tF)/tL

- Mess-Modell:
  - allgemeines lineares Messmodell als eigener Block



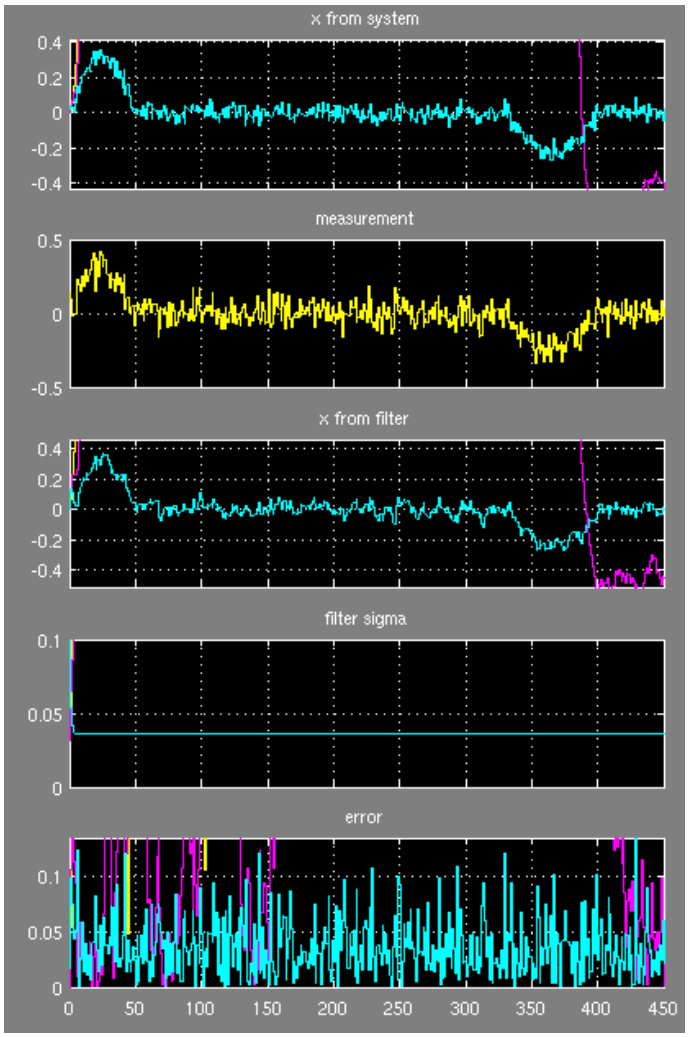
  - Parameter

Bezeichnung	**Variable**
observation matrix	H
variance of measurement	R

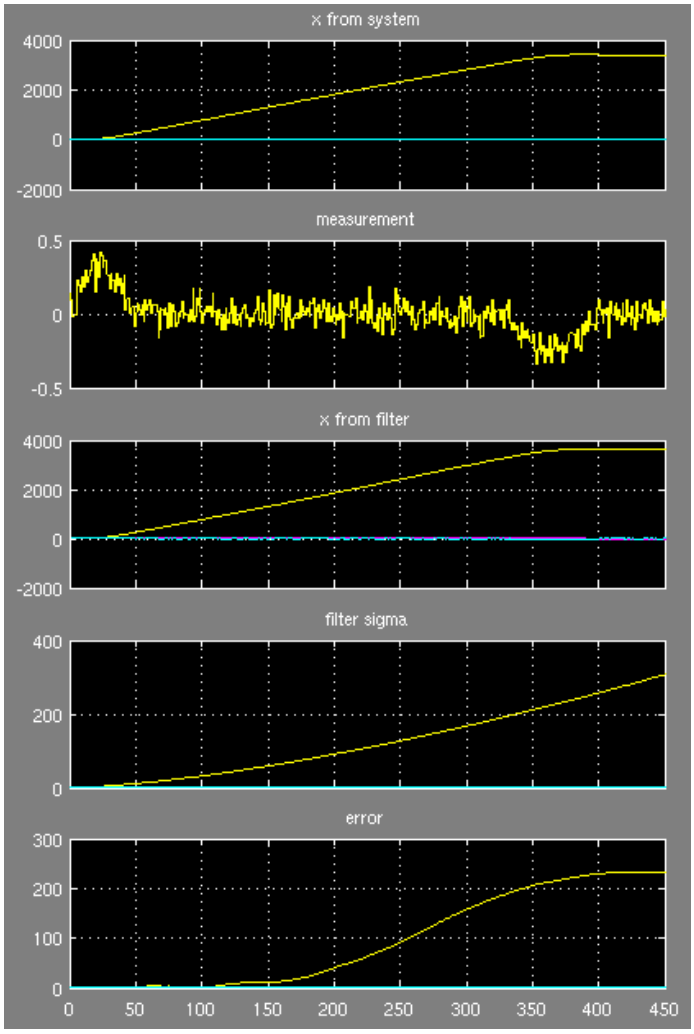
sample time	dt
initial seed	seed

- hier H = [0 0 1] → nur a wird gemessen

- Ergebnisse der Simulation:
  - Beschleunigung



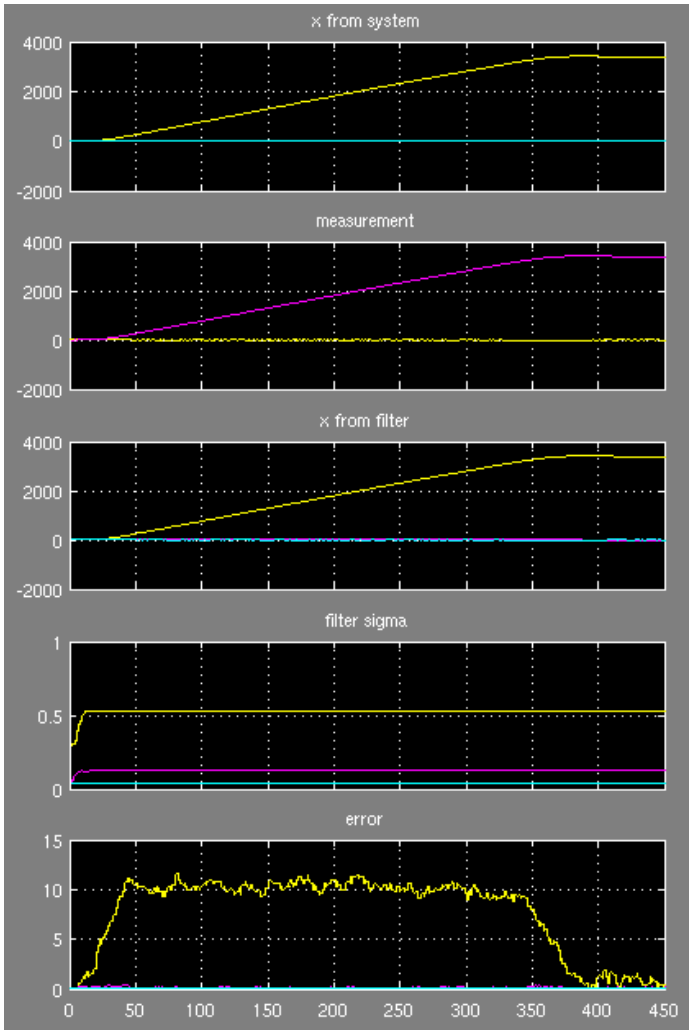
    - a vom System und a vom Filter nahezu identisch
    - Fehler in der Größenordnung der Filtervarianz
  - Ort



- Abweichung von x wächst kontinuierlich an
- Filtervarianz steigt entsprechend
- klar: Messmethode liefert prinzipiell keine besseren Werte

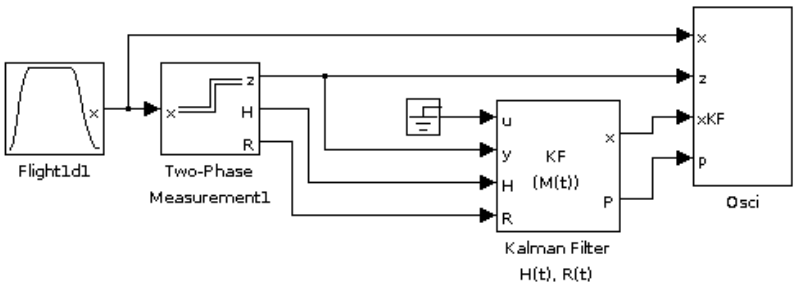
Integration von Ortsmessungen

- Ständige Ortsmessung:
  - Modell ins3.mdl
    - wie ins2.mdl, aber andere Parameter (in Messung und Filter!)
    - H = [0 0 1; 1 0 0]
    - R = [0.003, 0; 0, 1]
    - zusätzlich wird auch x gemessen, allerdings mit niedriger Genauigkeit
  - Ergebnis

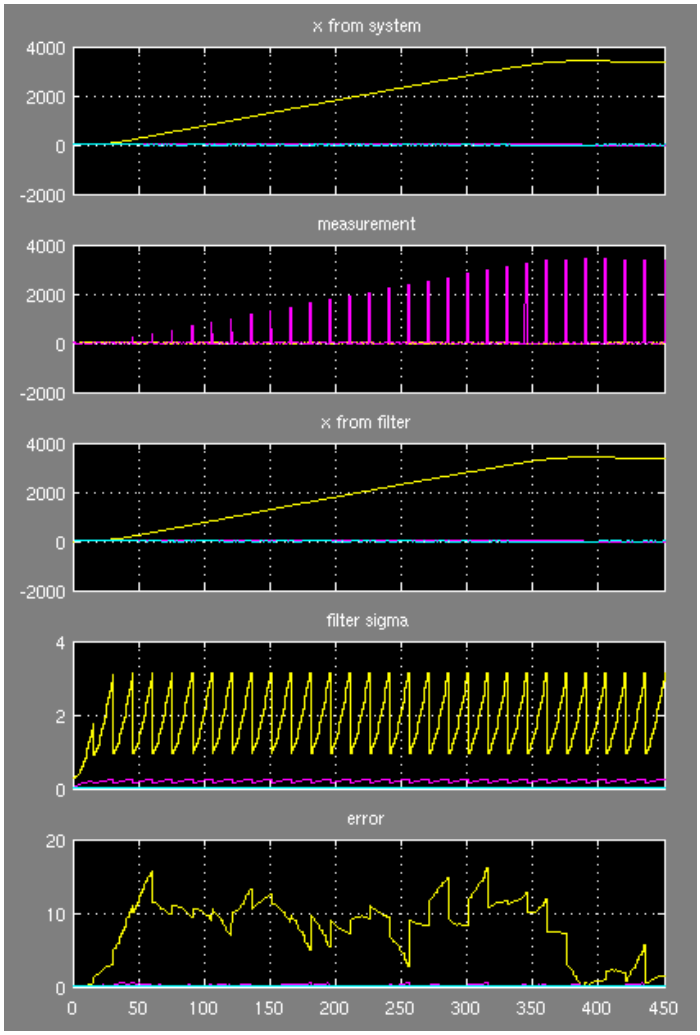


    - Fehler bleibt beschränkt, aber relativ groß
  - höhere Genauigkeit der x-Messung
    - → keine wesentliche Verbesserung
    - vermutete Ursache: internes Zufallspfad-Modell kann der schnellen Änderung von a nicht folgen

- Seltene Ortsmessungen:
  - Modell ins4.mdl



- Mess-Block liefert verschiedene Arten von Messungen
  - jedes n-te Mal (n einstellbar) a und x mit entsprechendem R
  - sonst nur a
- Kalman-Filter braucht sich änderndes (zeitabhängiges) H und R
- Ergebnis bei x-Messung bei jedem 15. Schritt



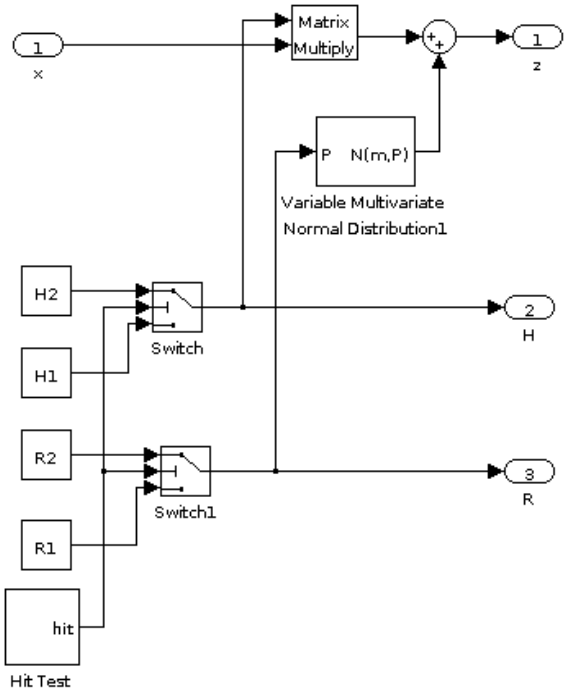
  - Fehler bleibt in der Größenordnung des vorigen Modells

- Zwei-Phasen-Messblock:
  - hat zwei verschiedene Messmethoden (jeweils mit H und R)
    - verwendet normalerweise H1, aber jedes n-te Mal H2
  - Parameter

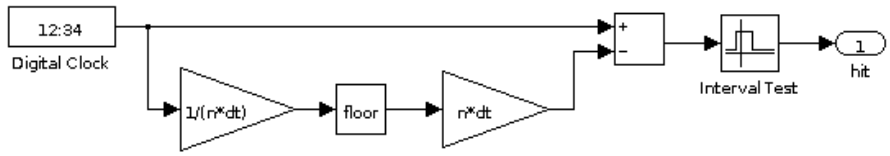
Bezeichnung	Variable
observation matrix H1	H1

observation matrix H2	H2
variance of measurement R1	R1
variance of measurement R2	R2
decimation factor n of measurement 2	n
sample time	dt
initial seed	seed

- Aufbau

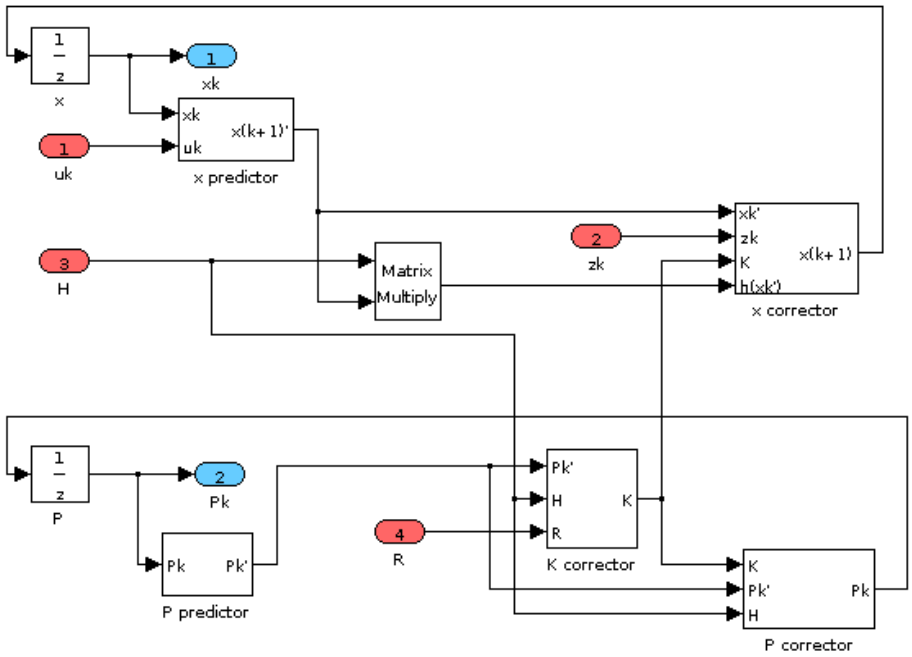


- Hit Test schaltet jedes n-te Mal durch



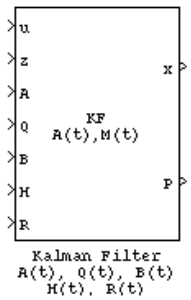
  - Interval Test statt Test auf 0 wegen möglichen Rundungsfehlern

- Kalman-Filter mit zeitabhängiger Messung:
  - H und R nicht fest über Parameter, sondern variabel über Eingänge
  - Aufbau

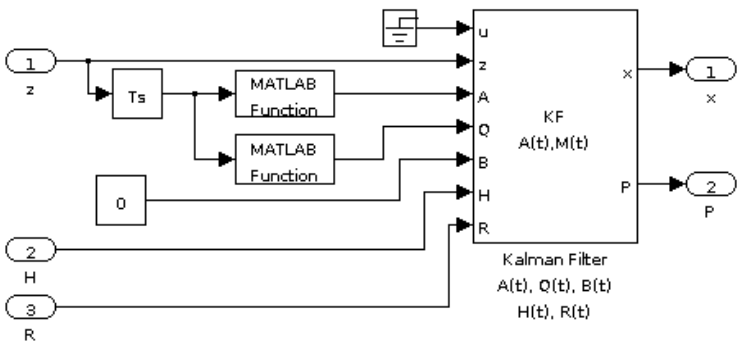


- setzt Formeln in direkter Weise um

- Erweiterung auf beliebige Sample-Zeiten:
  - Änderung von dt → A und Q ändern sich
  - daher bei Kalman-Filter auch A und Q als Eingänge



    - nur noch x0 und P0 als Parameter
    - Implementierung wie üblich
  - daraus spezielles Kalmanfilter für kinematisches Modell



    - Block Ts bestimmt Sample Time der Messung
    - Matlab-Funktion für A

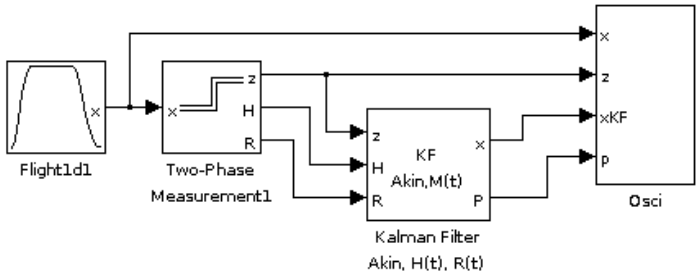
```
[1, u, 0.5*u^2; 0, 1, u; 0, 0, 1]
```

- Matlab-Funktion für Q

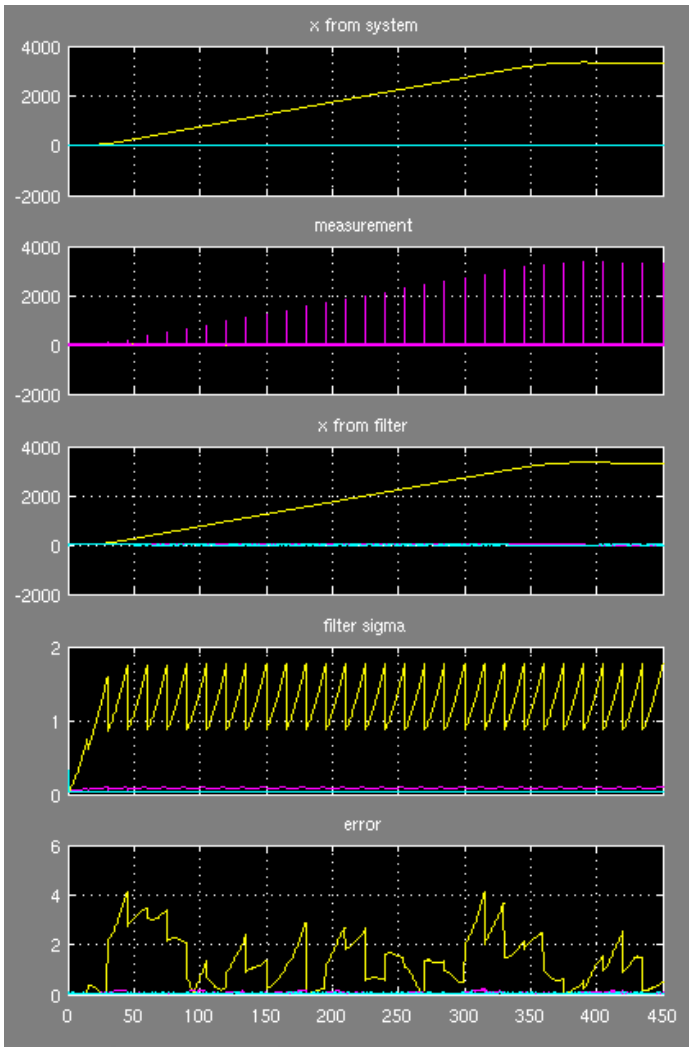
```
[0.050*u^5, 0.125*u^4, 0.167*u^3;...
0.125*u^4, 0.333*u^3, 0.500*u^2; ...
0.167*u^3, 0.500*u^2, 1.000*u]*q
```

- mit Parameter q für Kovarianz des kontinuierlichen Systems

- Gesamtmodell ins5.mdl



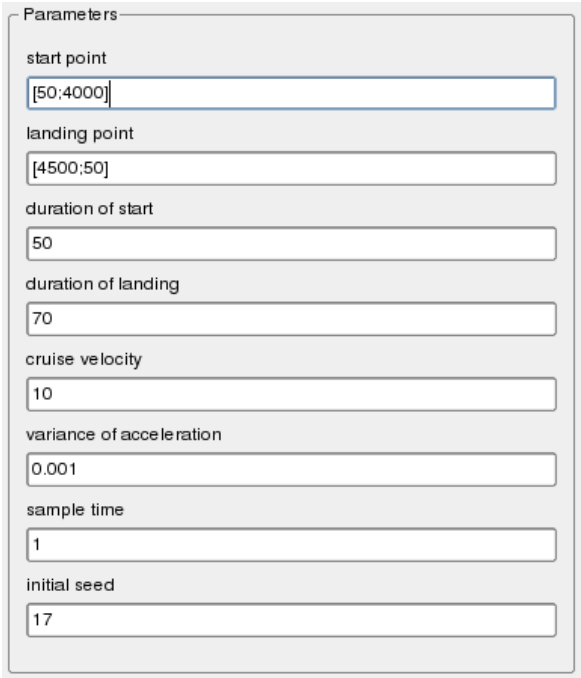
- Lauf mit Sample Time = 0.1 (bei Flight1d und Measurement!) und Dezimierung um n = 150 der Ortsmessung



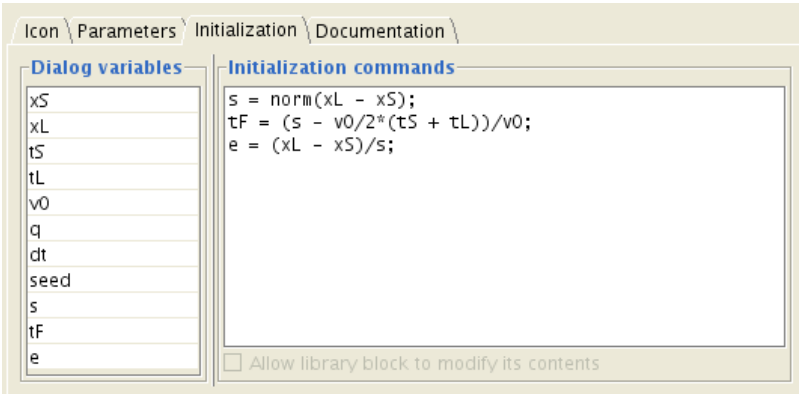
- Ergebnis
  - Zahl und Genauigkeit der Ortsmessungen wie vorher
  - aber Fehler hat stark abgenommen
  - also Vermutung bestätigt: genauere Beschleunigungsmessungen nötig für einfaches Zufallspfad-Modell

Kalman-Filter für nichtlineare Systeme

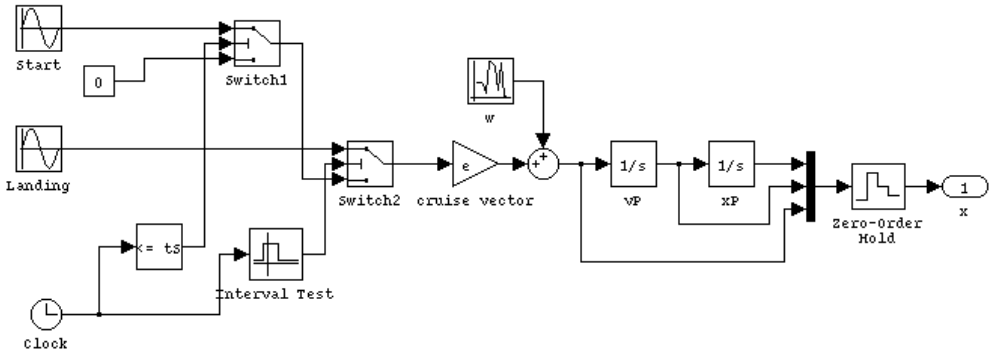
- Erweiterung der Modelle:
  - 2d-Flug
  - 2d-Abstandsmessungen zu vorgegebenen Bezugspunkten
  - Problem: Messwertbestimmung ist nichtlinear
  - Erweiterung des Kalman-Filters nötig (Extended Kalman Filter)
- Flugmodell:
  - analog zu Flight1D, aber mit 2d-Start- und Landekoordinaten
  - Zustandsvektor 6-dimensional (x, y, vx, vy, ax, ay)
  - Parameter



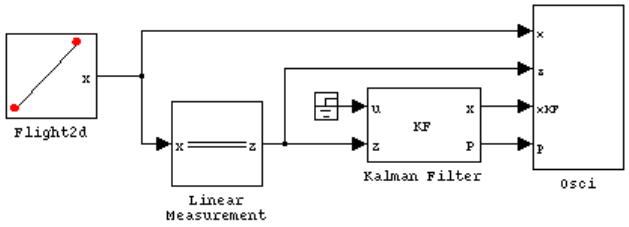
  - einige Hilfsrechnungen der Einfachheit halber in der Maske



  - Aufbau



- Einfaches Testsystem:
  - Modell flug2d1.mdl



  - Messung der beiden Beschleinigungskomponenten

```
H = [0 0 0 0 1 0; 0 0 0 0 0 1]
```

  - System-Matrizen
    - kinematisches System wie oben, mit dt = 1
    - aber alle Größen jetzt selbst zweidimensional

$$A = \begin{pmatrix} 1 & 0 & 1 & 0 & \frac{1}{2} & 0 \\ 0 & 1 & 0 & 1 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & \frac{1}{2} \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

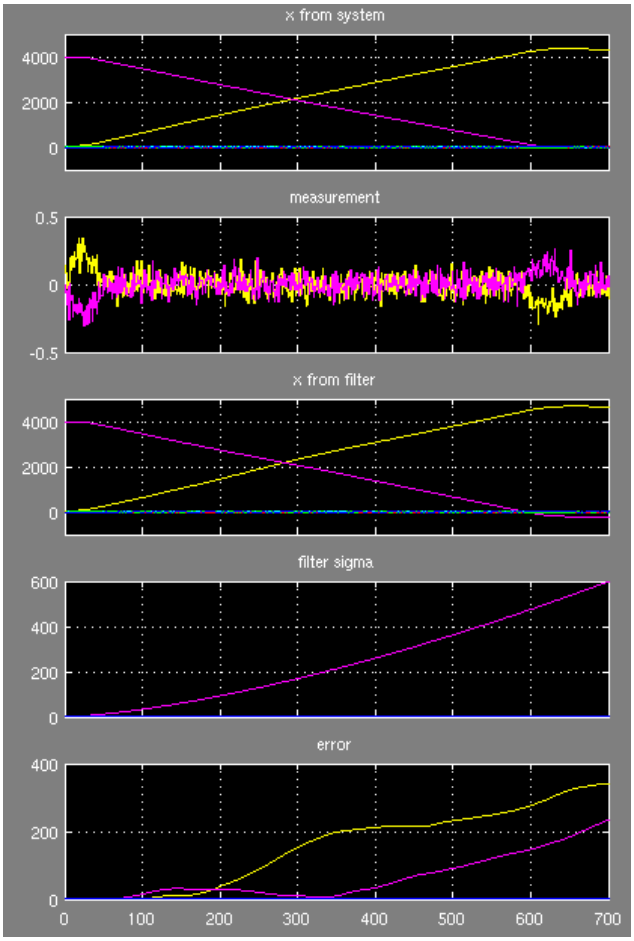
    - in Matlab

```
kron([1, 1, 0.5; 0, 1, 1; 0, 0, 1], [1, 0; 0, 1])
```

    - analog

$$Q = \begin{pmatrix} \frac{1}{20} & \frac{1}{8} & \frac{1}{6} \\ \frac{1}{8} & \frac{1}{3} & \frac{1}{2} \\ \frac{1}{6} & \frac{1}{2} & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

  - Ergebnis



    - Abweichung ähnlich wie im 1d-Fall

- Extended Kalman Filter:
    - Kalman-Filter kann für nichtlineares System erweitert werden
    - hier nur Spezialfall nichtlinearer Messung
    - Systemgleichungen

$$
\begin{aligned}
x_k &= Ax_{k-1} + Bu_{k-1} + w_{k-1}, \qquad w_k \in \mathcal{N}(0, Q) \\
z_k &= h(x_k) + v_k, \qquad v_k \in \mathcal{N}(0, R)
\end{aligned}
$$

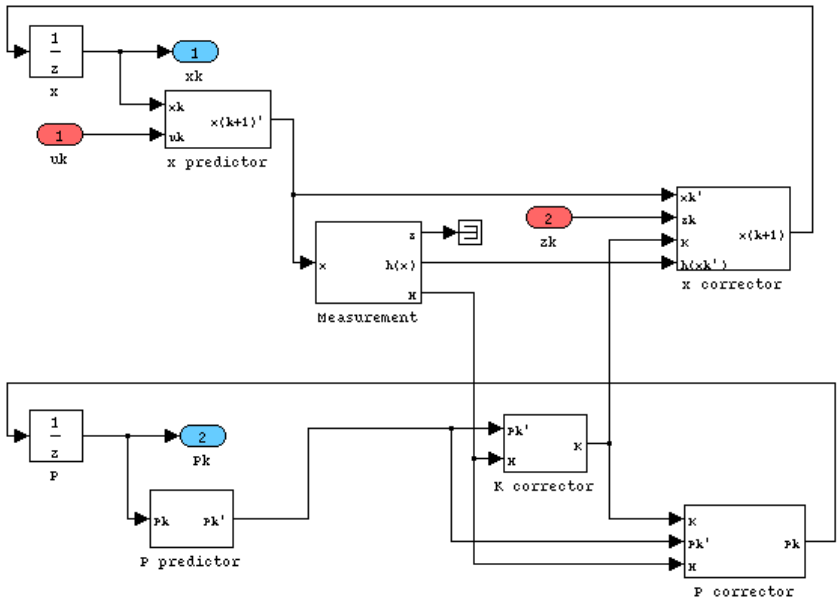
    - Filter-Gleichungen
        - Prädiktor

$$
\begin{aligned}
\hat{x}_k^- &= A\hat{x}_{k-1} + Bu_{k-1} \\
P_k^- &= AP_{k-1}A^T + Q
\end{aligned}
$$

        - Korrektor

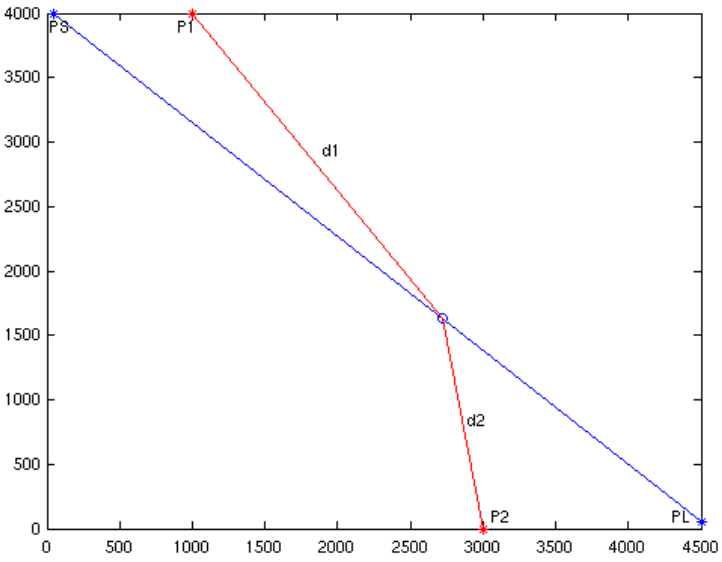
$$
\begin{aligned}
H_k &= \frac{\partial h}{\partial x}(\hat{x}_k) \\
K_k &= P_k^- H_k^T (H_k P_k^- H_k^T + R)^{-1} \\
\hat{x}_k &= \hat{x}_k^- + K_k(z_k - h(\hat{x}_k^-)) \\
P_k &= (1 - K_k H_k) P_k^-
\end{aligned}
$$

    - als Simulink-Modell



- Block Measurement muss für spezielle Messung angepasst werden

- Messmodell:
  - zusätzlich zu Beschleunigungen werden Abstände zu zwei vorgegebenen Punkten P1, P2 bestimmt



  - leicht berechnet als

$$
\begin{aligned}
h_1 &= \sqrt{(x-x_1)^2+(y-y_1)^2} \\
h_2 &= \sqrt{(x-x_2)^2+(y-y_2)^2} \\
h_3 &= a_x \\
h_4 &= a_y
\end{aligned}
$$

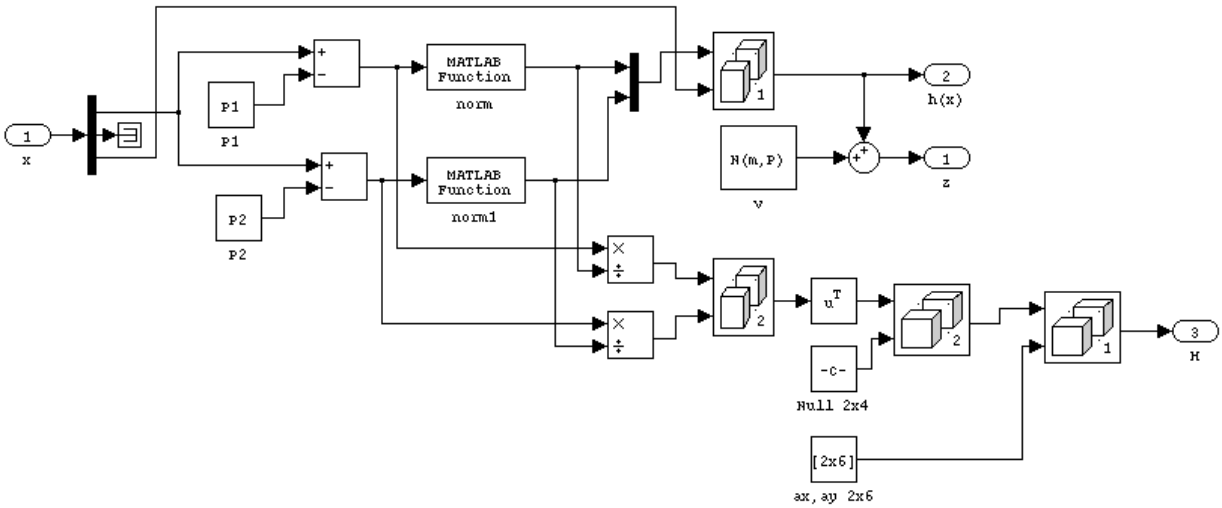
  - Jacobi-Matrix der Messung
    - mit

$$\frac{\partial h_1}{\partial x} = \frac{x - x_1}{\sqrt{(x-x_1)^2 + (y-y_1)^2}}$$
$$\frac{\partial h_1}{\partial y} = \frac{y - y_1}{\sqrt{(x-x_1)^2 + (y-y_1)^2}}$$
$$\frac{\partial h_1}{\partial v_x} = 0 = \frac{\partial h_1}{\partial v_y} = \frac{\partial h_1}{\partial a_x} = \frac{\partial h_1}{\partial a_y}$$

- erhält man

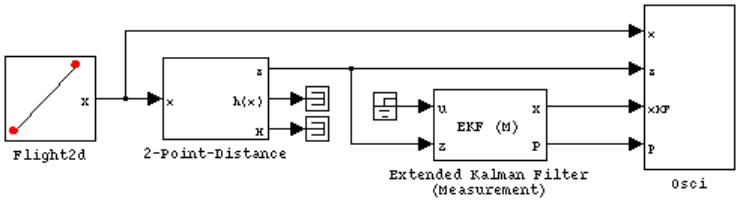
$$H = \begin{pmatrix} \frac{x - x_1}{\sqrt{(x-x_1)^2 + (y-y_1)^2}} & \frac{y - y_1}{\sqrt{(x-x_1)^2 + (y-y_1)^2}} & 0 & 0 & 0 & 0 \\ \frac{x - x_2}{\sqrt{(x-x_2)^2 + (y-y_2)^2}} & \frac{y - y_2}{\sqrt{(x-x_2)^2 + (y-y_2)^2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

  - liefert
    - Messwerte mit Rauschen (z)
    - Messwerte ohne Rauschen (h(x))
    - Jacobi-Matrix H
  - Koordinaten der Punkte P1, P2 als Parameter
  - Aufbau



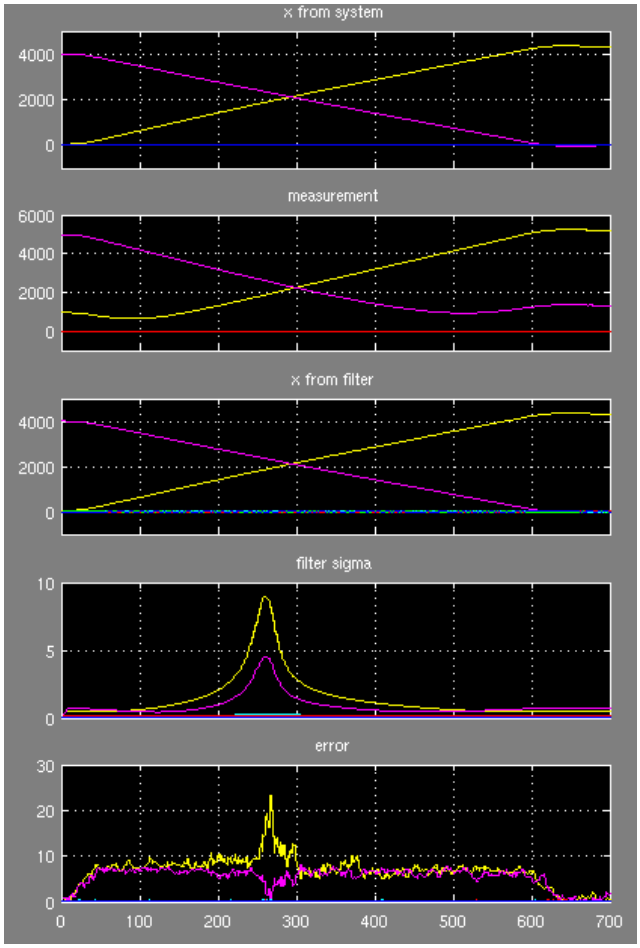
    - Zusammenschieben der Teile mit Block Matrix Concatenate

- Gesamtsystem:
  - Modell flug2d2.mdl



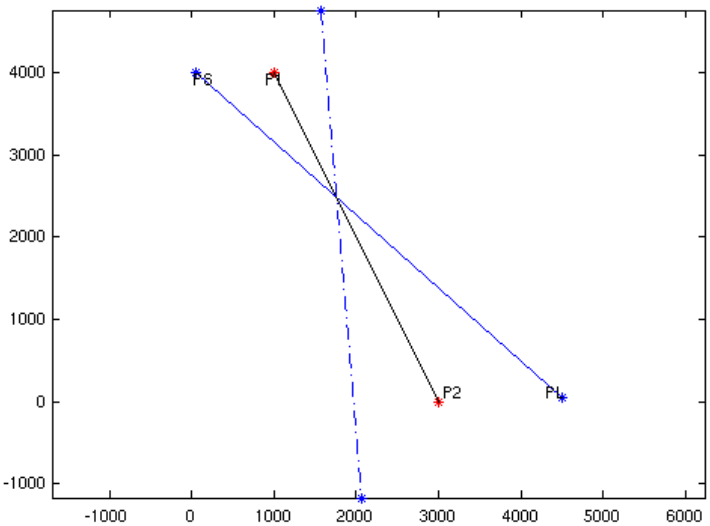
- Messungsblock 2-Point-Distance steckt auch im Extended Kalman Filter !

- Ergebnis



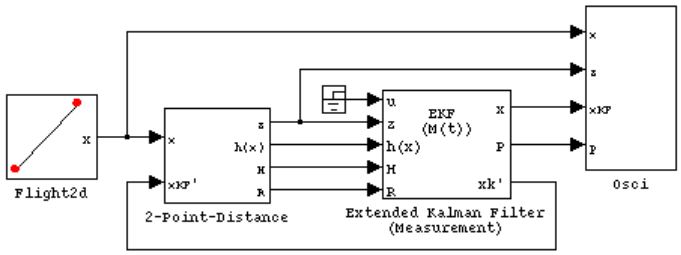
  - i.W. wie bei 1d
  - seltsame Peaks (hohe Ungenauigkeit) bei t = 250

- Ursache
  - Abstandsmessung liefert grundsätzlich zwei mögliche Orte

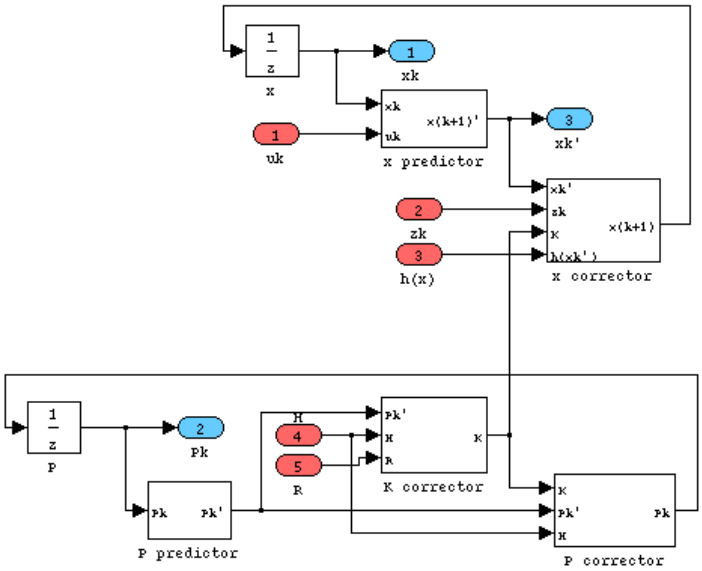


  - Anfangswert x0 legt pS fest
  - nahe des Schnittpunkts liefern die Abstandsmessungen keine Entscheidung
  - ungenaue Beschleunigungsmessung gibt den Ausschlag

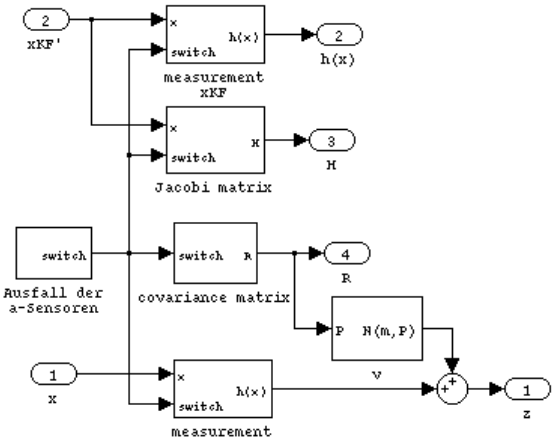
- Erweiterung des Modells:
  - Simulation eines temporären Ausfalls der Beschleunigungssensoren
    - Parameter für Beginn und Ende des Ausfalls
    - Umschaltung zwischen entsprechenden Messungen
  - Verbesserung der Bedienbarkeit
    - bisher zwei identische Messmodelle (eins im Filter)
    - Parameteränderungen immer bei beiden eintragen!
    - stattdessen jetzt nur ein Messmodell
    - liefert Ergebnisse für "reale" Messungen und für interne Filterberechnungen
  - Gesamtsystem



  - Messung wurde aus dem Filter ausgelagert

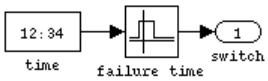


    - gibt seinen x-Predictor-Wert nach außen (zum Messblock)
  - Verbesserung des Messblocks
    - modularer Aufbau

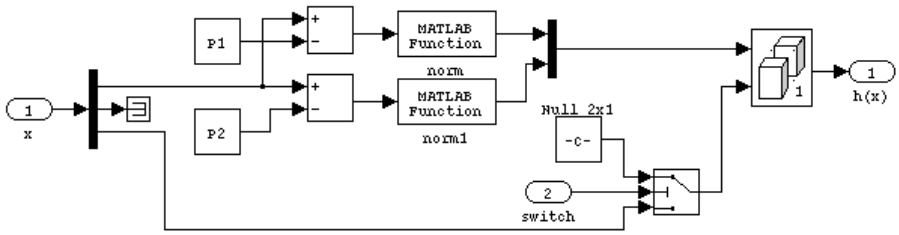


- verbessert deutlich die Übersichtlichkeit und Verständlichkeit

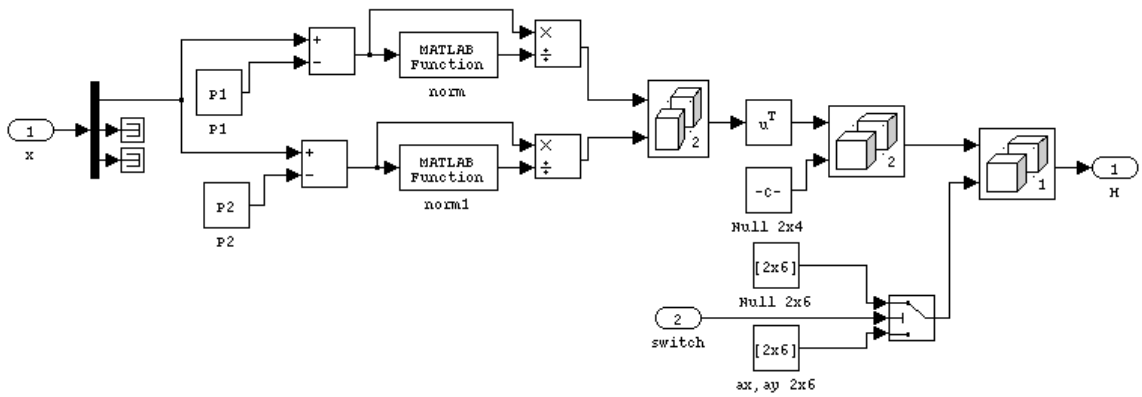
- Unterblöcke
  - Umschaltung bei Ausfall der a-Sensoren



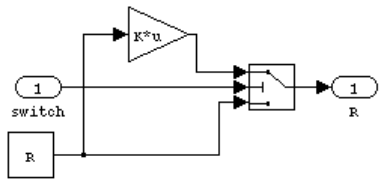
  - Berechnung der Messwerte (wird 2x verwendet)



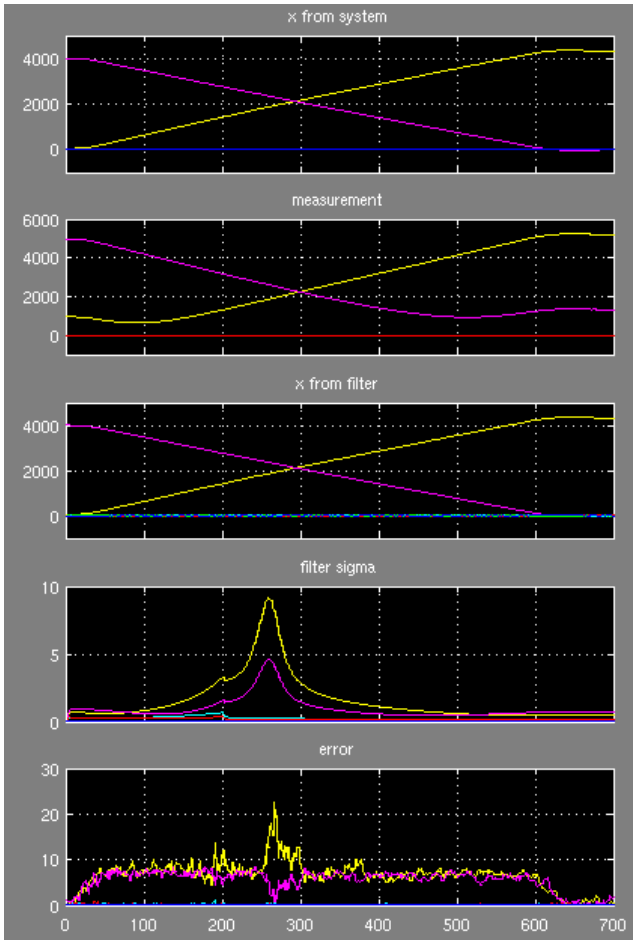
  - Berechnung der Jacobi-Matrix



  - Berechnung der Kovarianz-Matrix

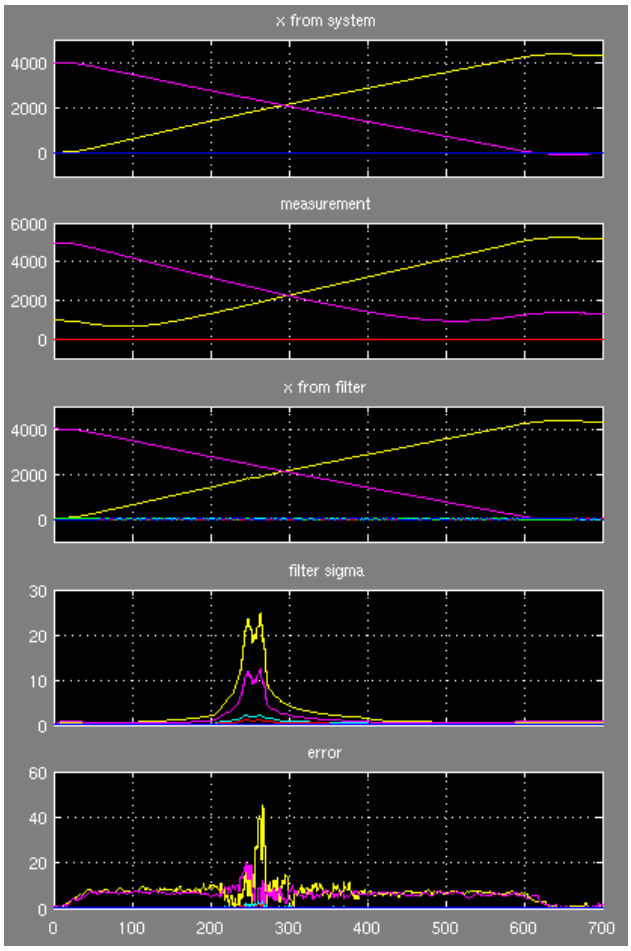


- Ergebnisse:
  - Ausfall von 0 bis 200



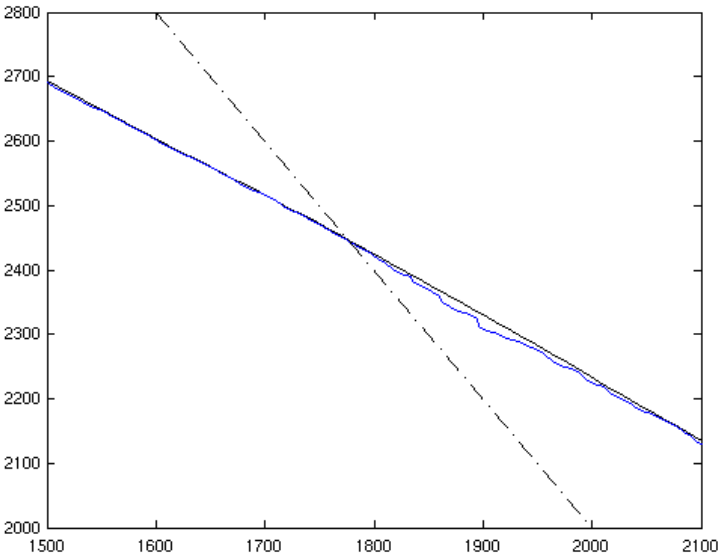
- Werte fasst wie vorher, etwas schlechter mit zunehmender Ausfallzeit

- Ausfall von 200 bis 400



- deutliche Verschlechterung

- Flugbahn und Filter-Vorhersage
  - mit funtionierendem a-Sensor



  - mit Ausfall des a-Sensors von 200 bis 400