Praktikum “Zustandsbeobachter”

Kalman Filter

Aufgabe 1 (Funktionsweise und Parameter des Kalman Filters)**.** *Nutzen Sie das auf Ilias bereitgestellte Skript “P2KalmanFilter.sce” um die Zustandsschätzung eines linearen Systems durchzuführen. Dokumentieren Sie Ihre Beobachtungen (wo nötig mit Grafikausgabe) in einem formlosen Dokument.*

Variieren Sie (jeweils unabängig)

- *die Rauschamplitude,*
- *das Eingangssignal,*
- *die Systemparameter,*
- *die Initialisierung von*
  - *Startzustand und*
  - R_w*-Matrix.*

Führen Sie weiterhin zwischen Systemsimulation und Kalman Filter eine kleine Störung der A*-Matrix ein.*

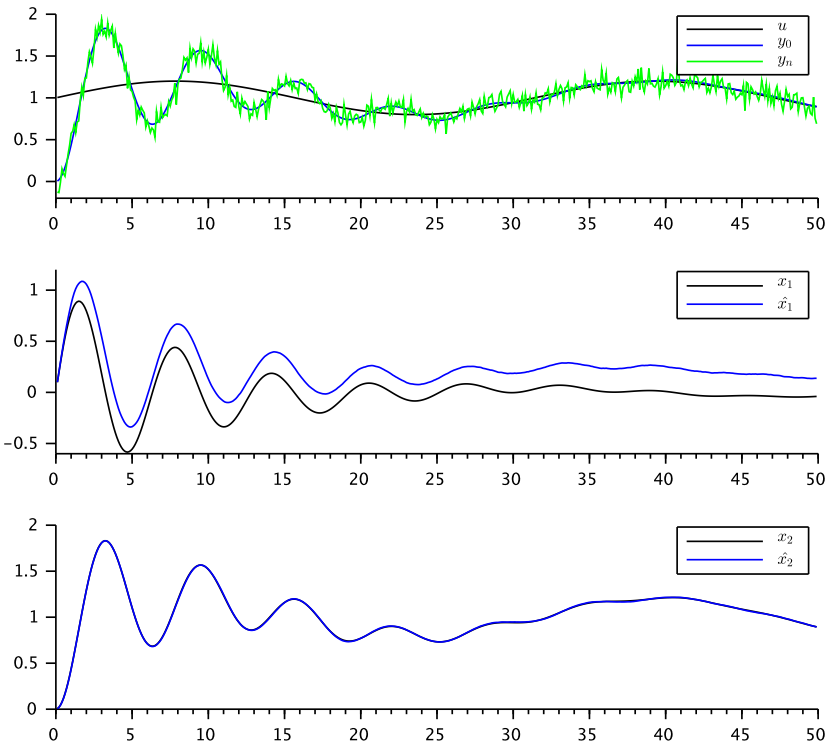
```
// P2KalmanFilter: KF state estimation using the Kalman filter
clear;xdel(winsid()),clc;

// System parameters
a1 = -1.5;a2 = 0.7;b0 = 0.9;
// System matrices
A = [0, -a2;
     1, -a1];
B = [0, b0]';
C = [0, 1];

// Simulation parameters
n = 2;// Number of states
N = 50;// Set the number of samples
sigma = 2;// Noise variance
x0 = [0; 0]; // Initial state

// System definition
Sys = syslin('d', A, B, C, 0, x0);
t = (1:N)';

// Input
// Input to the system is a step at start
start = 10;
// Define the noise
```

```
noise = rand(N,1,"normal");noise = sigma*(noise-mean(noise,"m"));
u = [0*ones(start,1);1*ones(N-start,1)]+0*rand(N,1);

y = zeros(3,1);
//Simulate system
y = dsimul(Sys, u');
xsim=ltitr(A,B,u',x0)

y0 = y;
y = y'+noise;

// Kalman filter
// Initialisation
x = zeros(n,1);
P = eye(n,n)*1e4;
// Noise information
rv = 1;
Rw = 0.01*eye(n,n);

xSave = [];

for t = 3:N
  //Prediction
  x = A*x+B*u(t-1);  // x^(t+1|t)= P x(t|t) + Q u(t)
  // Rw process noise covariance matrix
  P = (A*P)*A'+Rw;  // phi(t+1|t) = P phi(t|t) P'' + Rw
  //Correction
  // rv ouput noise variance
  K = (P*C')/(rv+(C*P)*C');  // K(t+1) = [phi(t+1|t) H'' ] / [rv + H phi(t+1|t) H'']
  x = x+K*(y(t)-C*x);  // x^(t+1|t+1) = x^(t+1|t) + K(t+1) [y(t+1) - H x^(t+1|t)]
  P = (eye(n,n)-K*C)*P;  // phi(t+1|t+1) = [I - K(t) H] phi(t+1|t)
  xSave = [xSave,x];
end;

subplot(3, 1, 1)
plot2d(1:N,y0,2)
plot2d(1:N,y,3)
plot2d(1:N,u,4)
legend('$y_0$', '$y$', '$u$',4)
subplot(3, 1, 2)
plot2d(1:N, xsim(1,:), 2)
plot2d(3:N, xSave(1,:),3)
legend('$x_1$', '$\hat{x}_1$')
subplot(3, 1, 3)
plot2d(1:N, xsim(2,:), 2)
plot2d(3:N, xSave(2,:),3)
legend('$x_2$', '$\hat{x}_2$', 4)
xlabel("Iterations")
```

Lösung 1 (Skript zur Lösung der Aufgaben 1)**.**
Explorative Aufgabe, keine Lösung vorgegeben.

Aufgabe 2 (Luenberger-Beobachter in Scilab)**.** *Ergänzen Sie das System-Modell aus dem ersten Praktikum um einen Luenberger-Beobachter zur Zustandsschätzung. Beachten Sie folgende Punkte:*

- *Basis der Schätzung ist das Ausgangssignal des Systems, versehen mit additivem weißen Rauschen.*
- *Das Rauschsignal importieren Sie mittels eines "fromWorkspace"-Blocks.*
- *Zeichnen Sie folgenden Zustände auf:*
  - *Eingangssignal* u
  - *Wahre Zustände* x_1, $y_0 = x_2$
  - *Geschätzte Zustände* $\hat{x}_1$, $\hat{x}_2$
  - *Gemessenes (verrauschtes) Ausgangssignal* y_n
- *Stellen Sie in je einem Plot gegenüber:*
  - u, y_0, y_n
  - x_1, $\hat{x}_1$
  - x_2, $\hat{x}_2$
- *Wählen Sie* $L = (0{,}2\ 0{,}1)^T$



Lösung 2 (Skript zur Lösung der Aufgaben 2)**.**

```
clear
delete(gcf())

// load the blocks library and the simulation engine
loadXcosLibs(); loadScicos();
// load diagram
importXcosDiagram("P2.zcos")

N = 1000; // time steps
tmax = 50; // final simulation time
Sigma = 0.1; // Noise amplitude

time = linspace(0,tmax,N)'; // time axis
// Define input data for fromworkspace block
myInput.time = time;
myInput.values = ones(time)+ .2*sin(0.2*time)+0*rand(time);
myNoise.time = time;
myNoise.values = grand(N,1, "nor", 0, Sigma);
// System parameters
a1 = .2;
a0 = 1;
b0 = 1;

A = [0, -a0;1, -a1];
B = [b0; 0];
C = [0 1];

L = [.2; .1];

// Execute scicos diagram
scicos_simulate(scs_m)
// Plot data
subplot(3,1,1)
plot2d(myOutput.time, myOutput.values(:,[1,3,6]))
legend('$u$', '$y_0$', '$y_n$')
subplot(3,1,2)
plot2d(myOutput.time, myOutput.values(:,[2,4]))
legend('$x_1$', '$\hat{x_1}$')
subplot(3,1,3)
plot2d(myOutput.time, myOutput.values(:,[3,5]))
legend('$x_2$', '$\hat{x_2}$')
```

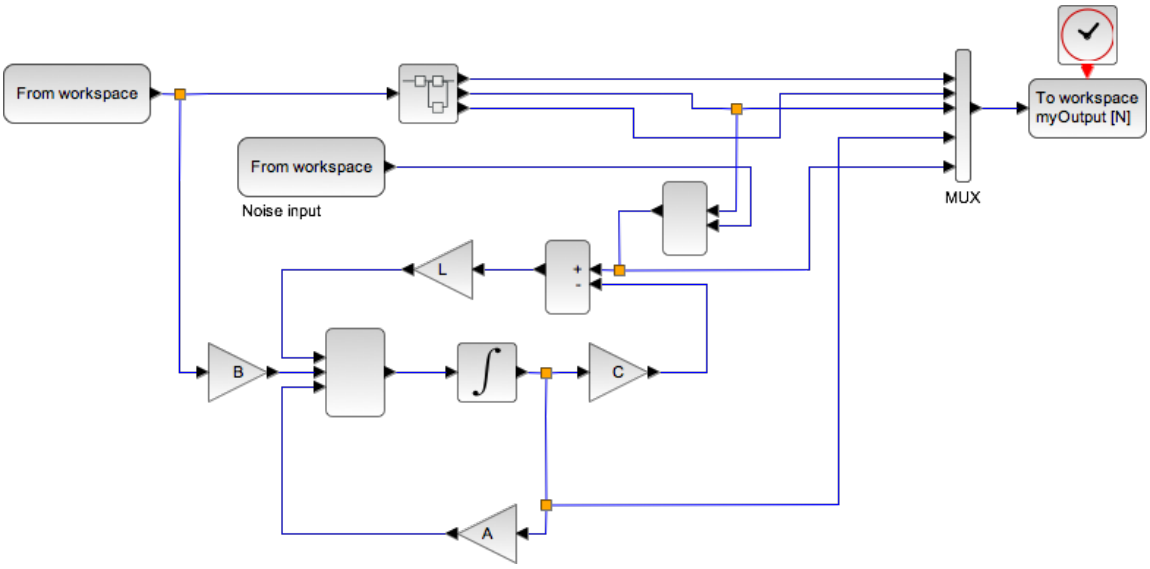


Abbildung 1: XCos Blockdiagramm