# Homework 2C: Asynchronous Sampling Report

**Timur Uzakov**

## Problem formulation

The homework exercise is based on simulation model of a continuous-time process, the furnace. The transfer function is following:

$$Y(s) = \frac{1}{(1 + s * \tau)^2}(U(s) + D(s))$$

Disturbance is considered as a Wiener process, with intensity Q = 0.001. The property of the Wiener signal is that it is composed of a previous noise value plus drift, which is equal to intensity multiplied by square root of sampling time and a random number.

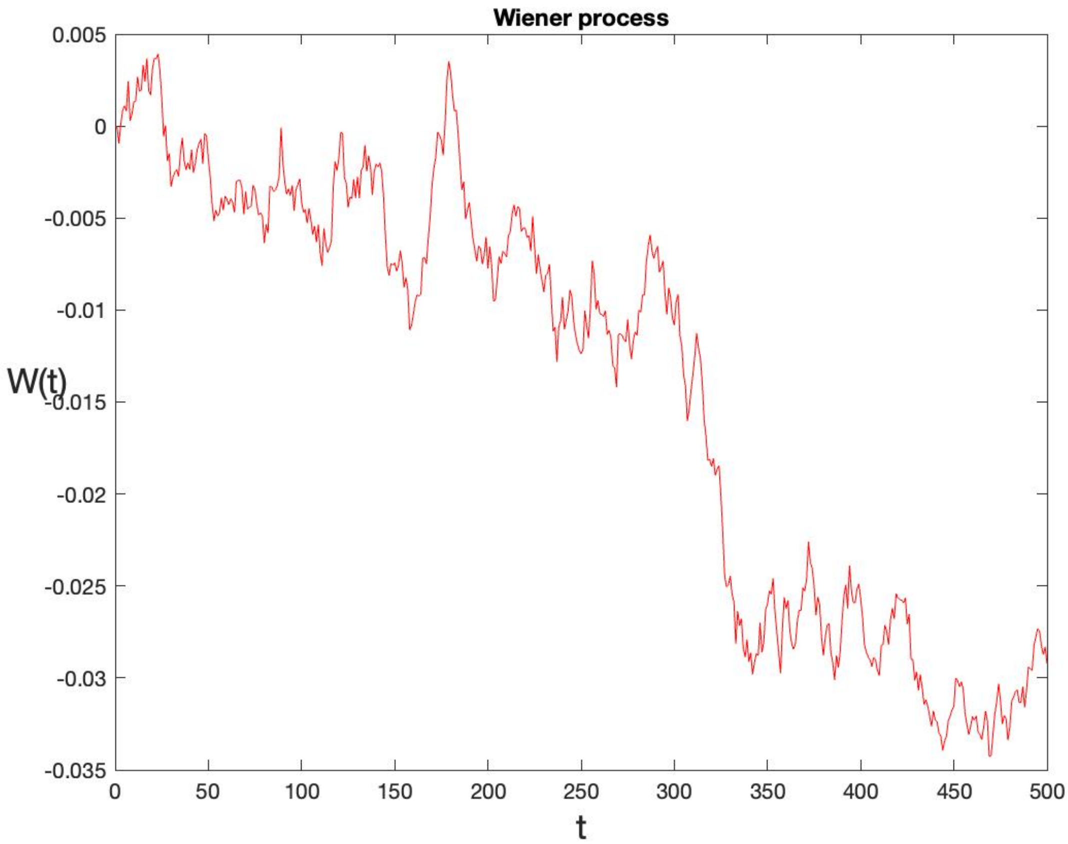


Fig. 1 - Example of Wiener Process

# Steps to solve the problem:

1) Finding the right model
2) Designing Kalman Filter and checking of co-variance values
3) Constructing asynchronous filtering equations
4) Comparing results with S and without S
5) Design KF for de-correlated noise

# Problem 1: Finding continuous-time stochastic state-space mode

Problem Statement:

```
% Y(s) = 1/(1+s*tau)^2 *(U(s)+D(s))
% Find continusous-time stochastic state space model
% Create disturbance as a Wiener process
% Intensity of disturbance Q = 0.001
```

Solution:

Creation of disturbance - dt is selected to be equal to 1, in order to match tau and simulation step size.

Wiener process:

```
n_seconds = 500;
N = 5000;
dt = n_seconds/N;

dW = zeros(1,N);
W = zeros(1,N);
dW(1) = Qc*sqrt(dt)*randn();
W(1) = dW(1);
for j = 2:N
    dW(j)= Qc*sqrt(dt)*randn();
    W(j) = W(j-1) + dW(j);
end

.
x = Ax + Bu + Gw      {State equation}
```

```
y = Cx + Du +  v      {Measurements}

tau = 50; % 50 seconds
[A,B,C,D] = tf2ss(1,[tau^2 2*tau 1]);
G = [1;0];
B = [B G];
D = [D 0];
sys = ss(A,B,C,D);
step_input = ones(1,N+1);
disturbance = [0,W];
```

# Continuous-time State-Space Model

$$\dot{x} = A * x + B * u + G * d$$

$$y = C * x + D * u + 0 * d$$

$x = \begin{bmatrix} x_{flow} \\ \dot{x}_{flow} \end{bmatrix}$, where $x_{flow}$ is flow of gas in the furnace, $u$ is input gas to the furnace, $d$(disturbance) is variability of gas caloric value.

Values of matrices A,B,C,D are obtained from the transfer function.

# Model augmented

$$\dot{x} = A * x + \begin{bmatrix} B \\ G \end{bmatrix} \begin{bmatrix} u & d \end{bmatrix}$$

$$y = C * x + \begin{bmatrix} D \\ 0 \end{bmatrix} \begin{bmatrix} u & d \end{bmatrix}$$

```
response = lsim(sys,[step_input;disturbance],[0:dt:n_seconds]);
```

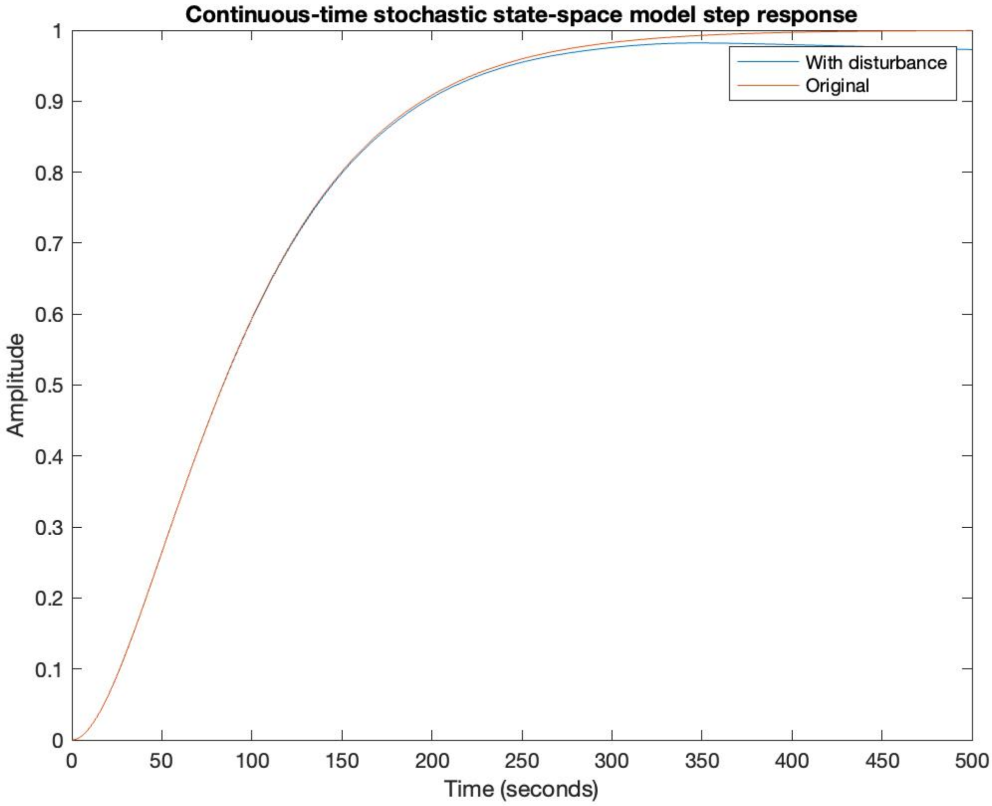


Fig.2 System response: original and with disturbance

# Problem 2: Synchronous discrete-time Kalman Filter design and evaluation of co-variances

Problem Statement:

```
% Find discrete-time Kalman-filter model

% Ts = 20s
% Evaluate predicted and filtered values
```

Solution:

The sampling time for discrete Kalman Filter is given and set to 20s. Measurement noise R was selected as $0.01^2$ times identity matrix and Q was selected with value 0.001.

Before proceeding to Kalman filtering, the system was discretised.

```
Ts = 20*dt;
dsys = c2d(sys,Ts,'tustin');
```

Kalman filter is designed using *kalman* command, which returns system model *kest,* Kalman gain *L* and prediction *P* and filtering *Z* co-variances.

```
S = [0.001; 0.001];
Qc  = [0.001 0; 0 0.001];
Rc  = 0.0001;
Px = 10*eye(2);
Qn = A*Px*A'+ Qc;
Sn = A*Px*C'+ S;
Rn = C*Px*C'+ Rc;
[kest,L,P,M,Z] = kalman(dsys,Qn,Rn,Sn);
```

The values are following:

```
P

   1.0e+03 *

    0.6183    1.2043
    1.2043    5.5307
Z
   1.0e+03 *

    0.3107    0.4309
    0.4309    3.5862
```

# Problem 3: Set of corresponding models and asynchronous sampling KF design

Problem Statement:

```
% Find Kalman Filter models
% Use asynchronous sampling with controller computation time
% Tc = 10/1/0.1/s
% Compare properties of noise models
% Evaluate P (predicted value)
```

Solution:

For solving this task, the equations for deterministic part and stochastic part of asynchronous sampling were obtained. (Slide 108 in the course's lecture slides)

| |
|---|
| $\epsilon = \frac{T_s - T_c}{T_s}$ |
| $A_{async} = e^{A_c T_s}$ |
| $B_{async} = \int_0^{T_s} e^{A_c \nu} d\nu B_c$ |
| $C_{async} = C_c e^{A_c \epsilon Ts}$ |
| $D_{async} = C_c \int_0^{\epsilon T_s} e^{A_c \nu} d\nu B_c$ |
| $Q = \int_0^{T_s} e^{A_c \nu} Q_c e^{A_c^T \nu} d\nu$ |
| $S = \int_0^{\epsilon T_s} e^{A_c \nu} Q_c e^{A_c^T \nu} C_c^T d\nu$ |
| $R = \int_0^{\epsilon T_s} C_c e^{A_c \nu} Q_c e^{A_c^T \nu} C_c^T d\nu + R_c$ |

Equations from slides, asynchronous schema

For each of the controller sampling times the sets of models were constructed and predicted co-variance values have been presented.

```
Ts = 20*dt;
```

```
Tcs = [10 1 0.1].*dt;
for i = 1:3
    Tc = Tcs(i);
    etta = (Ts-Tc)/Ts;
    funB = @(x) expm(A.*x);
    funD = @(x) expm(A.*x);
    funQ = @(x) Qc*expm(A.*x)*expm(A'.*x);
    funS = @(x) Qc*expm(A.*x)*expm(A'*x)*C';
    funR = @(x) Qc*C*expm(A.*x)*expm(A'.*x)*C';

    Aasync = expm(A.*Ts);
    Casync = C*expm(A.*etta*Ts);
    Basync = integral(funB,0,Ts,'ArrayValued', true);
    Dasync = integral(funD,0,etta*Ts,'ArrayValued', true);
    Q = integral(funQ,0,Ts,'ArrayValued', true);
    S = integral(funS,0,etta*Ts,'ArrayValued', true);
    R = integral(funR,0,etta*Ts,'ArrayValued', true);
    Basync = Basync*B;
    Dasync = C*Dasync*B;
    R = Rc + R;

    disp("Tc")
    disp(Tcs(i)/dt)
    [kest,L,P,M,Z] =
kalman(ss(Aasync,Basync,Casync,Dasync,Tcs(i)),Q,R,S);
    P
    Z

end
```

The resulting prediction co-variance values are following:

**Tc = 10**

```
P

     1.0e+04 *

    0.0250    0.1971
    0.1971    1.6343

Z
   1.0e+03 *

    0.0171    0.0891
    0.0891    1.1309
```

**Tc = 1**

```
P

   41.8238   85.9226
   85.9226  740.0424



Z
   33.0200   60.3392
   60.3392  665.6981
```

**Tc = 0.1**

```
P

   27.0894   48.5662
   48.5662  619.9598

Z
   23.5563   37.8957
   37.8957  587.7330
```

# Problem 4: Illustrate the impact of neglecting the mutual covariance S between the process and measurement noise in the case of asynchronous sampling.

Problem Statement:

```
% Show impact of neglecting S in case of asynchronous sampling
```

Solution:

For solving this task, again an iteration loop has been constructed:

```
for i = 1:3
    Tc = Tcs(i);
    etta = (Ts-Tc)/Ts;

    funB = @(x) expm(A.*x);
    funD = @(x) expm(A.*x);
    funQ = @(x) Qc.*expm(A.*x)*expm(A'.*x);
    funS = @(x) Qc.*expm(A.*x)*expm(A'.*x)*C';
```

```
    funR = @(x) Qc.*C*expm(A.*x)*expm(A'.*x)*C';

    Aasync = expm(A*Ts);
    Casync = C*expm(A*etta*Ts);
    Basync = integral(funB,0,Ts,'ArrayValued', true);
    Dasync = integral(funD,0,etta*Ts,'ArrayValued', true);
    Q = integral(funQ,0,Ts,'ArrayValued', true);
    S = integral(funS,0,etta*Ts,'ArrayValued', true);
    R = integral(funR,0,etta*Ts,'ArrayValued', true);
    Basync = Basync*B;
    Dasync = C*Dasync*B;
    R = Rc + R;

    disp("Tc")
    disp(Tcs(i))
    [kest,L,P,M,Z] = kalman(ss(Aasync,Basync,Casync,Dasync,Ts),Q,R,S);
    disp("With S")
    P
    Z
    disp("Value of determinant with S")
    disp(det(P))
    disp("With S=0")
    [kest,L,P,M,Z] = kalman(ss(Aasync,Basync,Casync,Dasync,Ts),Q,R);
    P
    Z
    disp("Value of determinant without S")
    det(P)
end
```

At first part, it assigns S from formula and, at the second part, it sets S to 0 (which is default)

The comparison table is provided below:

**Tc = 10**

```
With S
P

      1.0e+04 *

    0.0250    0.1971
    0.1971    1.6343


Z
      1.0e+03 *

    0.0171    0.0891
    0.0891    1.1309
```

```
Without S

P
   1.0e+04 *

    0.0250    0.1977
    0.1977    1.6424


Z
      1.0e+03 *

    0.0170    0.0885
    0.0885    1.1265
```

**Tc = 1**

```
With S
P

   41.8238   85.9226
   85.9226  740.0424

Z
   33.0200   60.3392
   60.3392  665.6981

Without S
P

   43.5880   90.9456
   90.9456  754.6480

Z
   33.9951   63.0886
   63.0886  673.7543
```

**Tc = 0.1**

```
With S
P

   27.0894   48.5662
   48.5662  619.9598


Z
   23.5563   37.8957
   37.8957  587.7330
```

```
Without S
P

   28.2612   51.2440
   51.2440  628.0586

Z
   24.3899   39.6566
   39.6566  593.3755
Conclusion: inclusion of S decreases the values of covariances
```

# Problem 5: Evaluate the filtered values of state covariance matrix P(t|t) using filter design for system model with de-correlated noise.

Problem Statement:

```
% Evaluate filtered P(t|t)
% Use filter design for system with not correlated noise (S=0)
```

Solution:

To solve this task, the appropriate section in lecture slides has been applied. Noise properties of KF model have been used as before, however, A,B and Q matrices were updated accordingly to formulas in the slides (115).

| |
|---|
| $A' = A - SR^{-1}C$ |
| $B' = B - SR^{-1}D$ |
| $Q' = Q - SR^{-1}S^T$ |
| $R = R$ |

Equations from slides, de-correlated noise

```
Tc = Tcs(3);
etta = (Ts-Tc)/Ts;

funB = @(x) expm(A.*x);
funD = @(x) expm(A.*x);
funQ = @(x) Qc.*expm(A.*x)*expm(A'.*x);
funS = @(x) Qc.*expm(A.*x)*expm(A'.*x)*C';
funR = @(x) Qc.*C*expm(A.*x)*expm(A'.*x)*C';
```

```
Aasync = expm(A*Ts);
Casync = C*expm(A*etta*Ts);
Basync = integral(funB,0,Ts,'ArrayValued', true);
Dasync = integral(funD,0,etta*Ts,'ArrayValued', true);
Q = integral(funQ,0,Ts,'ArrayValued', true);
S = integral(funS,0,etta*Ts,'ArrayValued', true);
R = integral(funR,0,etta*Ts,'ArrayValued', true);
Basync = Basync*B;
Dasync = C*Dasync*B;
R = Rc + R;

A_decor = Aasync - S*inv(R)*Casync;
B_decor = Basync - S*inv(R)*Dasync;
Q_decor = Q - S*inv(R)*S';

disp("State covariance matrix P(t|t) for system model with decorrelated
noise")

[kest,L,P,M,Z] = kalman(ss(A_decor,B_decor,C,D,Ts),Q_decor,R);
Tc
P
Z
```

The following are resulting values:

**Tc = 0.1**

```
P

   1.0e+04 *

     0.0306    0.3621
     0.3621    4.6320


Z
   26.9668   48.3192
   48.3192  618.1228
```