Lecture 5
First Order Transfer Function Models

Process Control
Prof. Kannan M. Moudgalya

IIT Bombay
Tuesday, 30 July 2013

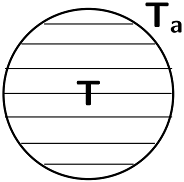
Outline

1. **Thermal sensor as a first order system**
2. **Scilab demonstration of curve fitting**
3. **First order systems in series**

Scilab demonstration

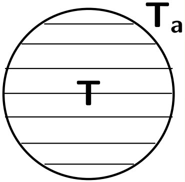
1. Thermal sensor as a I order system

Thermocouple to Measure Temperature



- **Shaded: sensor**
- **Mass m and heat capacity C_p**
- **Area available for heat transfer**
- **Initially, sensor and ambient are at T**
- **All of a sudden, the ambient temperature goes to T_a**
- **Model this: get T as a fn. of T_a**

Model of temp. measurement system



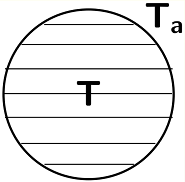
$$mC_p \frac{dT(t)}{dt} = hA(T_a - T(t))$$

- Initially, sensor & ambient at $\bar{T}$
- Ambient temperature goes to T_a
- Subtract the steady state and express in deviational variables
- $\Delta T(s) = G(s) \Delta T_a(s)$
- $G(s) = \dfrac{1}{\tau s + 1}$. Gain? Why?
- where, $\tau = \dfrac{mC_p}{hA}$
- What is the unit of τ?

A sample problem statement

- **A thermometer at 30°C is suddenly immersed into boiling water**
- **Thermometer be modelled as a first order system with time constant of 0.5min**
- **Calculate the temperature profile as indicated by the thermometer.**

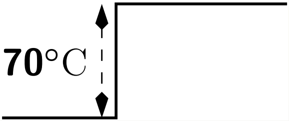
Solution to thermometer problem



- $Y(s) = G(s)U(s)$
- $G(s) = \dfrac{K}{0.5s+1}$, $K = 1$
- $U(s)$ = ambient temperature
- $Y(s)$ = temperature indicated by the thermometer, deviational
- $y(t) \leftrightarrow Y(s)$, $u(t) \leftrightarrow U(s)$
- Initial temperature = 30°C

1. $y(t) = T(t)$, $u(t) = T_a(t)$.
2. $y(t) = T(t) - 30$, $u(t) = T_a(t)$.
3. $y(t) = T(t)$, $u(t) = T_a(t) - 30$.
4. $y(t) = T(t) - 30$, $u(t) = T_a(t) - 30$. right

Can also use appropriate variables

- $\Delta T(s) = \frac{1}{\tau s + 1} \Delta T_a(s)$
- $\Delta T(t) \leftrightarrow \Delta T(s)$, $\Delta T_a(t) \leftrightarrow \Delta T_a(s)$
- $\tau = 0.5\text{min}$
- $\Delta T_a(t) =$ 

- $\Delta T_a(s) = \frac{70}{s}$
- $\Delta T(t) = 70\left(1 - e^{-t/0.5}\right)$
- $T(t) = 30 + \Delta T(t)$

2. Scilab Demonstration of Curve Fitting

Recall: Step response of a first order system

Let the plant have a 1st order transfer function:

$$\mathbf{y(s) = \frac{K}{\tau s + 1} u(s)}$$

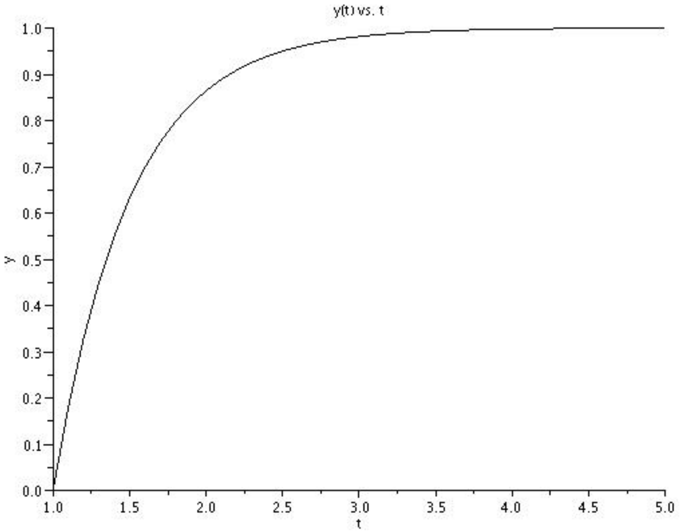
Its step response is given by,

$$\mathbf{y(t) = K\left(1 - e^{-t/\tau}\right)}$$

Scilab code to plot step response of a first order system: 02-step.sce

```
s = %s
tau = 0.5;
G = 1/(tau*s+1)
t = 1:0.1:5;
y = csim('step',t,G);
plot2d(t,y)
```

Step response of I order system in Scilab



Curve Fitting

- **Given model parameters τ and K, we can obtain step response**
- **Can we back-calculate?**
- **Given the step response, can we calculate τ and K?**

Identification of first order TF through step response

- **Experimental data have noise**
- **Does it affect identification?**
- **Demonstrate with**

$$G(s) = \frac{2.5}{0.75s + 1}$$

with noise superimposed

Data generation to simulate a noisy system: step-noise.sce I

```
s = %s
tau = 0.75;
G = 2.5/(tau*s+1)
t = 0:0.01:5;
y = csim('step',t,G);
plot2d(t,y)
xlabel("t")
ylabel("y")
xtitle("y(t) vs. t")
```

Data generation to simulate a noisy system: step-noise.sce II

```
noise = rand(1,length(y),'normal');
ynoise = y + 0.25*noise;
// plot2d(t,ynoise)
y2noise = y + 0.025 * y .* noise;
plot2d(t,y2noise)
```

We will use noise corrupted y as the plant data

Can we extract the model parameters by curve fitting using plant data?

Curve fitting by optimisation

- **Plant data:**
  y = noise + output of the plant with $G = 2.5/(0.75s + 1)$
- **Fit this with the output y_prediction from a plant with transfer function with $G = K/(\tau s + 1)$**
- **Get K and τ by minimising $||y - y\_prediction||^2$**
- **What should we get for K and τ?**

Scilab code to identify a first order transfer function: order_1.sce I

```
exec('step-noise.sce');
exec('costf_1.sci');
y = y2noise;
global('y','t');

x0 = [3 2];
[f,xopt,gopt] = optim(costf_1,'b',[0.1 0.1],[5 5],x0,'ar',500,500)
kp = xopt(1);
```

Scilab code to identify a first order transfer function: order_1.sce II

```
tau = xopt(2);
y_prediction = kp * ( 1 - exp(-t/tau) );
format('v',6);
plot2d(t,y)
plot2d(t,y_prediction)
```

Scilab code for objective function: costf_1.sci I

```
function [f,g,ind] = costf_1(x,ind)
kp = x(1); tau = x(2);
y_prediction = kp * ( 1 - exp(-t/tau) );
f = (norm(y-y_prediction,2))^2;
g = numdiff(func_1,x);
endfunction
```

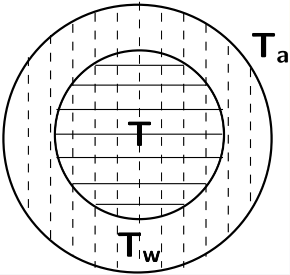
Scilab code for objective function: costf_1.sci II

```
function f = func_1(x)
kp = x(1); tau = x(2);
y_prediction = kp * ( 1 - exp(-t/tau) );
f = (norm(y-y_prediction,2))^2;
endfunction
```

3. First order systems in series

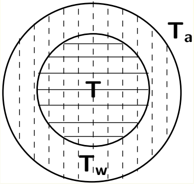
More realistic model of thermometer

- **Modelled as a first order system**
- **A more realistic model:**



- **The sheath is also modelled**
- **Its temperature is denoted as T_w, T-wall**
- **Derive the second order relation between $\Delta T_a(s)$ and $\Delta T(t)$**

More realistic model of thermometer



- Model of the sensor:
- $mC_p\frac{dT}{dt} = h_{sw}A_{sw}(T_w - T)$
- Model of the wall:
- $m_wC_{pw}\frac{dT_w}{dt} = h_{wa}A_{wa}(T_a - T_w) - h_{sw}A_{sw}(T_w - T)$
- Derive the relation between T_a and T
- Same procedure as before: deviational
- Order of the transfer function: second

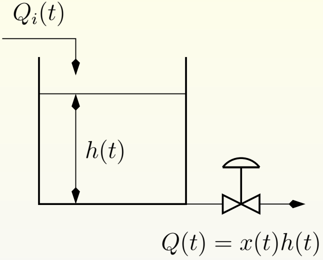
Model derivation method

- **Laplace transform models are in deviational form**
- **For each equation, subtract the corresponding steady state**
- **Assume small deviation, if required**
- **Take Laplace transform of each equation and simplify**
- **Substitute the expression for T_w into the other equation**
- **You will get T as a function of T_a only**
- **The transfer function will be second order: denominator will be a second degree polynomial**

Will do it as an assignment

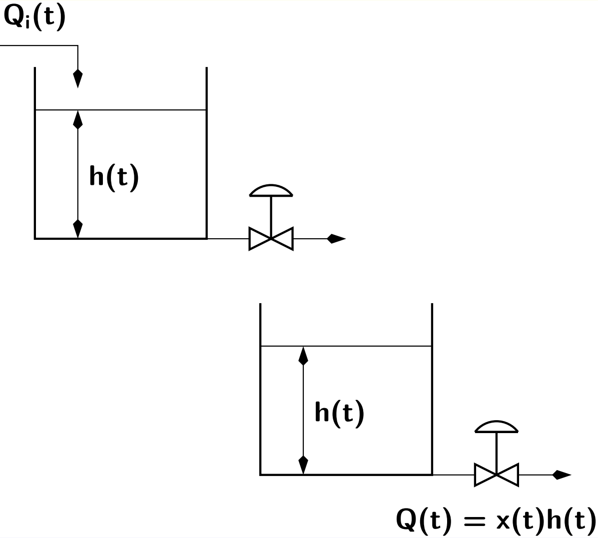
Another second order example

Recall: Model of Flow Control System



- **Suppose that x is constant**
- $\Delta h(s) = \dfrac{K_1}{\tau s + 1}\Delta Q_i(s)$
- $\tau = A/x_s$, $K_1 = 1/x_s$, $K_2 = h_s/x_s$.
- $\Delta Q(s) = x_s \Delta h(s)$

Two tanks in series



- Let out flow rate of I tank $= \Delta Q_1(s)$
- $\Delta Q_1(s) = G_1(s)\Delta Q_i(s)$
- $\Delta Q(s) = G_2(s)\Delta Q_1(s)$
- $= G_2(s)G_1(s)\Delta Q_i(s)$

How do you model this?

Comparing response of first and second order systems

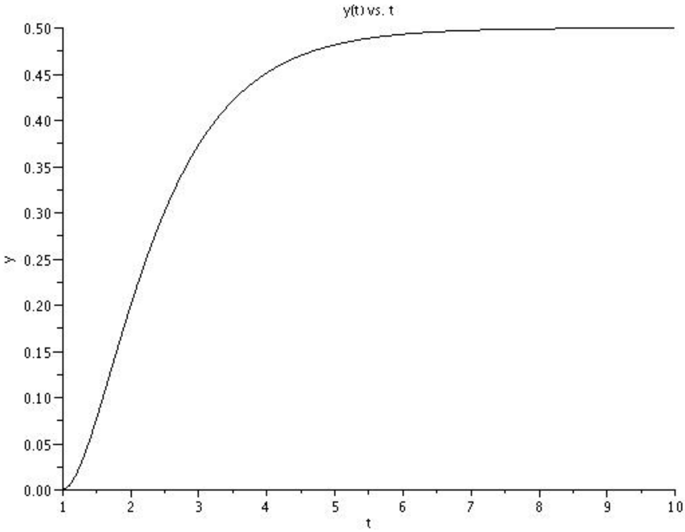
- **Response of**
  - **a first order system**
  - **two first order systems in series**

are similar

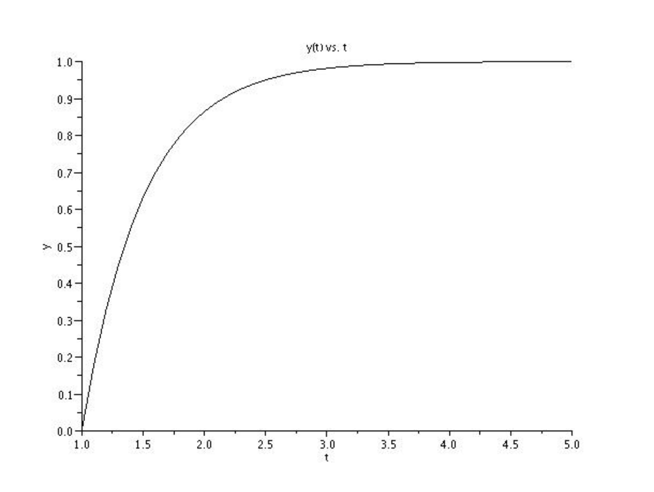
Scilab code to plot step response of a seond order system: step-2nd.sce

```
s = %s
G_1 = 1/(s+1)
G_2 = 1/(s+2)
G = G_1*G_2
t = 1:0.1:10;
y = csim('step',t,G);
plot2d(t,y)
xlabel("t")
ylabel("y")
xtitle("y(t) vs. t")
```

Step response of second order system



Recall: step response of a first order system: $\mathbf{1/(0.5s+1)}$



Can we interchange data and model?

What is learnt today

- First order system
- Second order systems
- Examples
- Scilab demonstration

Thanks