Transfer function parameters identification (based on experimental data). ***

A dynamic system has been excited using a step on its input.

The system's output y(t) has been logged; the logging started when the step was applied, so when t=0. The recorded datas are available in the file "data_system_ok.mat".

One would like to approximate this dynamic system with a model build on a second order transfer function having 2 real poles and a dead-time :

$$H_m(s) = \frac{K\, e^{-s*T_m}}{(\mathrm{sT}_1 + 1)(\mathrm{sT}_2 + 1)}$$

with :

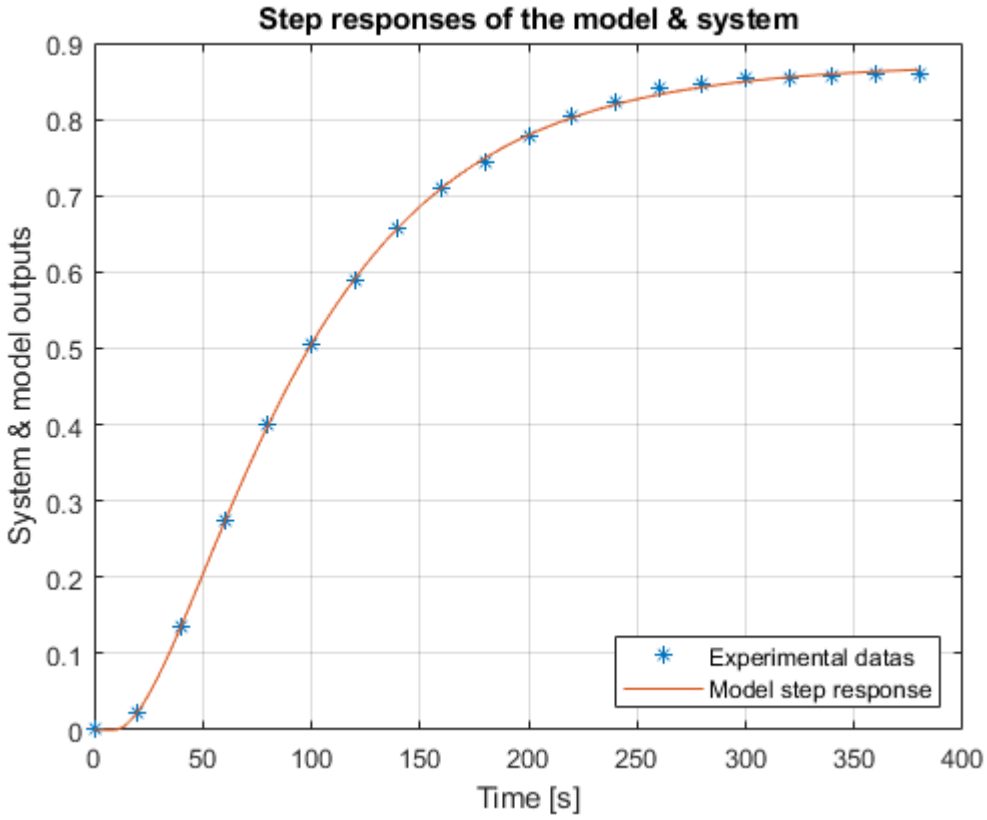
- K, The system'a static gain
- Tm, the dead-time
- T1 and T2, the 2 time constants

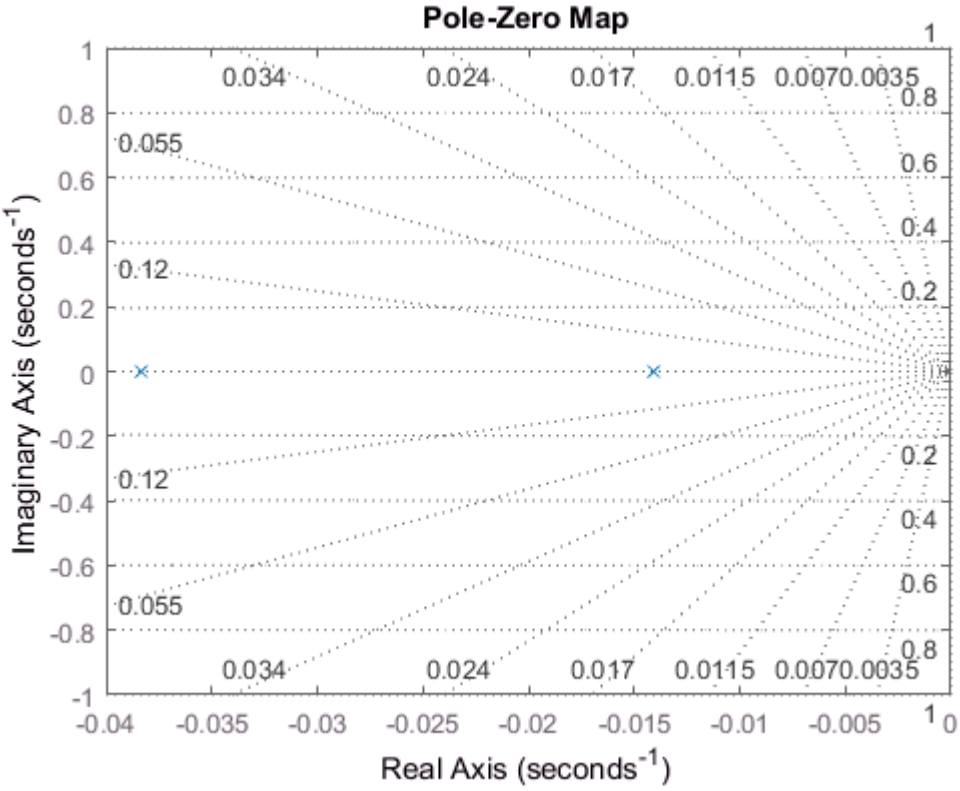
You're asked to :

- Find the transfer function's parameters, ensuring its step answer would fit as good as possible to the experimental data. (Remember the tutorial about optimization)
- Plot on one unique chart the experimental datas and the model's answer, using for this one 500 points equally splitted on the time duration of the experimental data.
- Compute and plot the model's poles in the complex domain.

Solution :

- ***Optimized parameters of the model : K= 0.87, Tm= 9.02 s, T1= 26.11 s, T2= 70.87 s***
- ***Model's poles : -0.0383 & -0.0141***






```
clear all
close all
load data_system_OK
plot(t,y,'*')
```

```
hold on
s=tf('s');

Tm = 20;
K = 0.87;
T1 = 80;
T2 = 10;
p0 = [Tm K T1 T2];
p = p0;
H = exp(-s*p(1))*p(2)/(s*p(3)+1)/(s*p(4)+1);

cout = @(p,t,y) norm(step(exp(-s*p(1))*p(2)/(s*p(3)+1)/(s*p(4)+1),t)-y);
cout(p0,t,y)
```

```
ans = 0.0538
```

```
p_opt = fminsearch(@(p) cout(p,t,y), p0)
```

```
p_opt = 1×4
    9.0227    0.8724   70.8665   26.1115
```

```
cout(p_opt,t,y)
```

```
ans = 0.0158
```

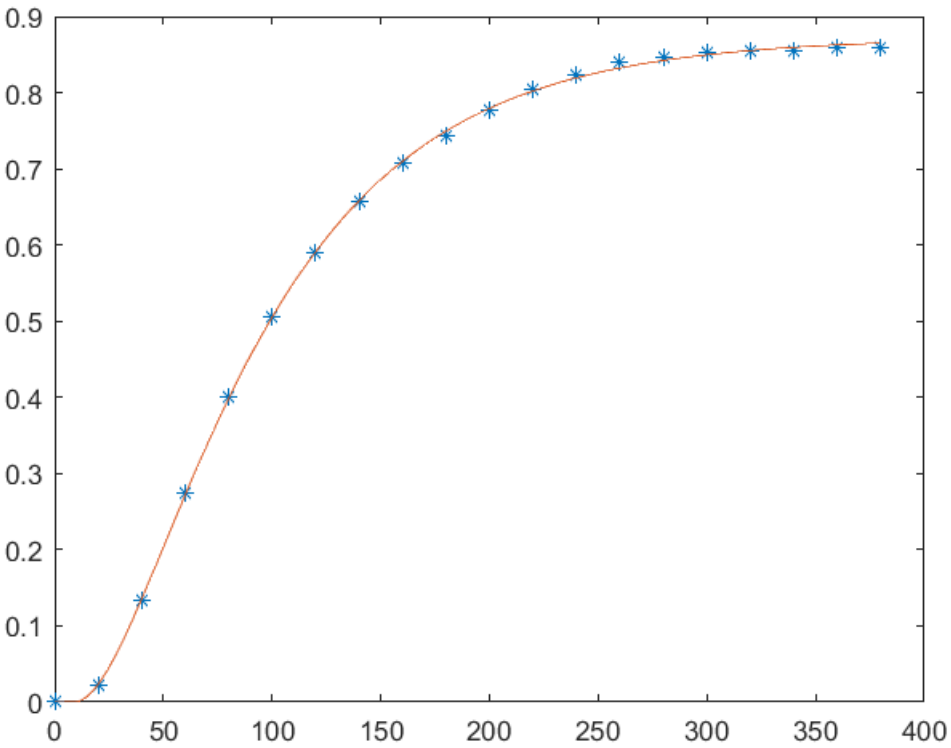
```
H_opt = exp(-s*p_opt(1))*p_opt(2)/(s*p_opt(3)+1)/(s*p_opt(4)+1)
```

```
H_opt =

                        0.8724
  exp(-9.02*s) * ----------------------
                 1850 s^2 + 96.98 s + 1

Continuous-time transfer function.
```

```
t_plot = linspace(t(1),t(end),500);
y_sim=step(H_opt,t_plot);
plot(t_plot,y_sim,'-')
```


0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
50
100
150
200
250
300
350
400

```
fun=@(p,s) step((p(1)*exp(-s*abs(p(2))))/((s*p(3)+1)*(s*p(4)+1)),t)
```

```
fun = function_handle with value:
    @(p,s)step((p(1)*exp(-s*abs(p(2))))/((s*p(3)+1)*(s*p(4)+1)),t)
```

```
cost=@(fun,p,s,y) norm(fun(p,s)-y)
```

```
cost = function_handle with value:
    @(fun,p,s,y)norm(fun(p,s)-y)
```

```
p0=[1 1 1 1];
p_opti=fminsearch(@(p) cost(fun,p,s,y),p0)
```

```
Exiting: Maximum number of function evaluations has been exceeded
         - increase MaxFunEvals option.
         Current function value: 0.126869
p_opti = 1×4
    0.8483  -37.1071    0.0000   62.3838
```