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 :

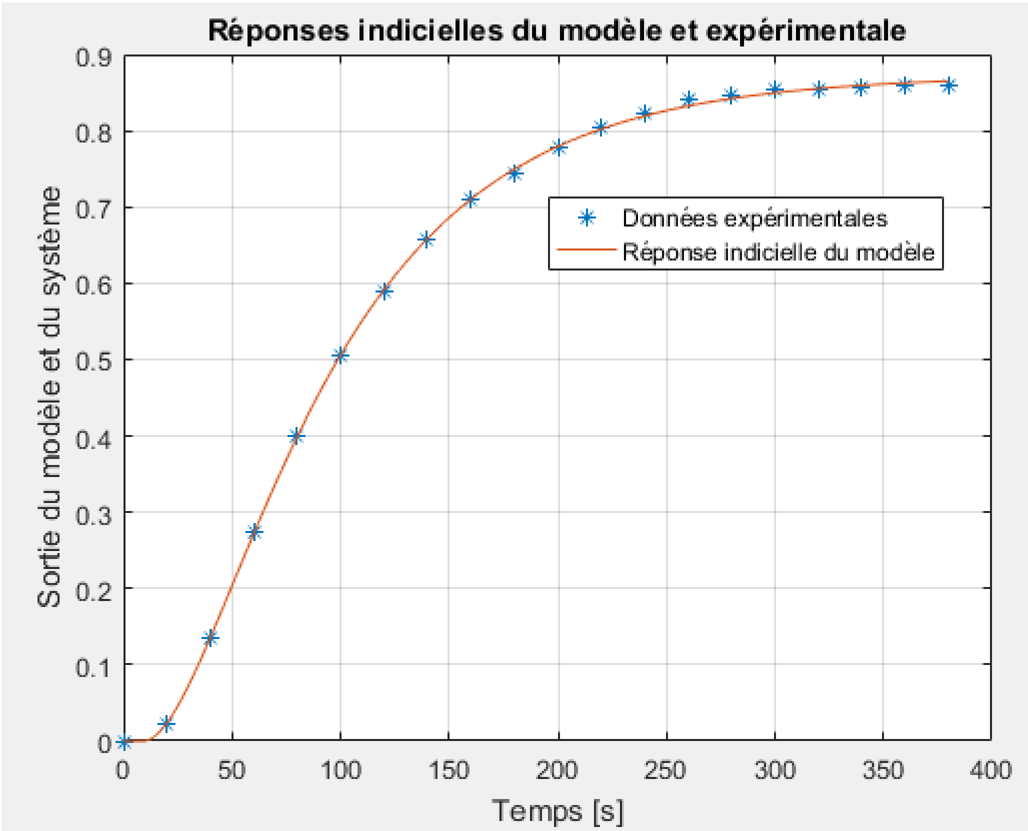
- K, The system'a static gain
- Tm, the dead-time
- T1 and T2, the 2 time constants

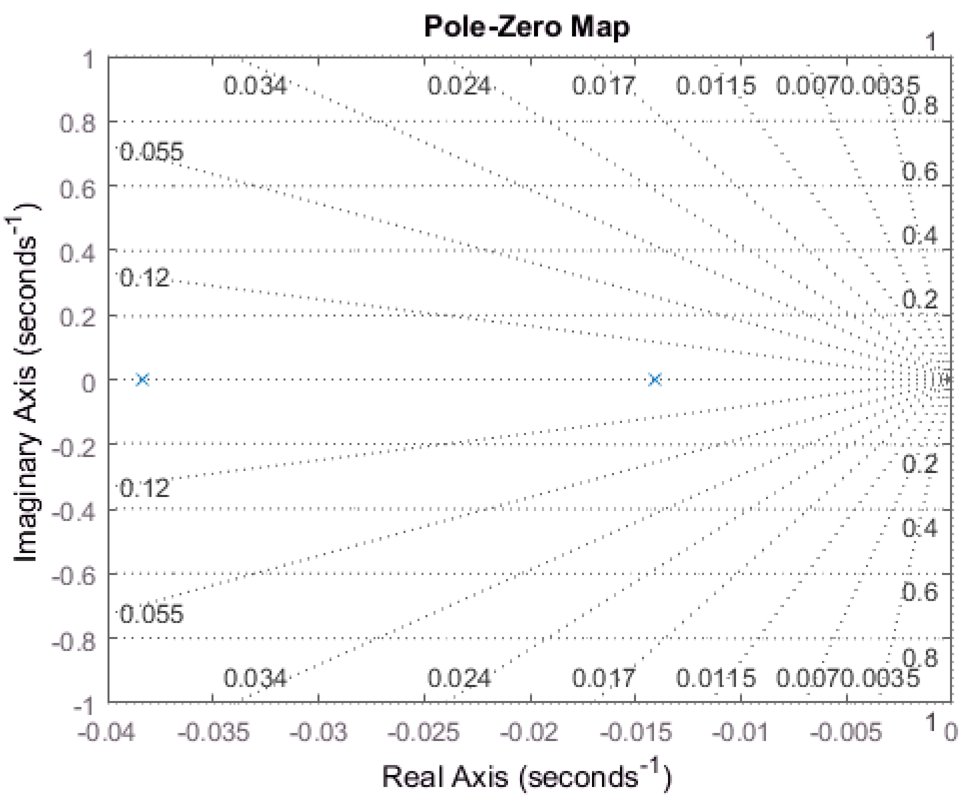
You're asked to :

- 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 :

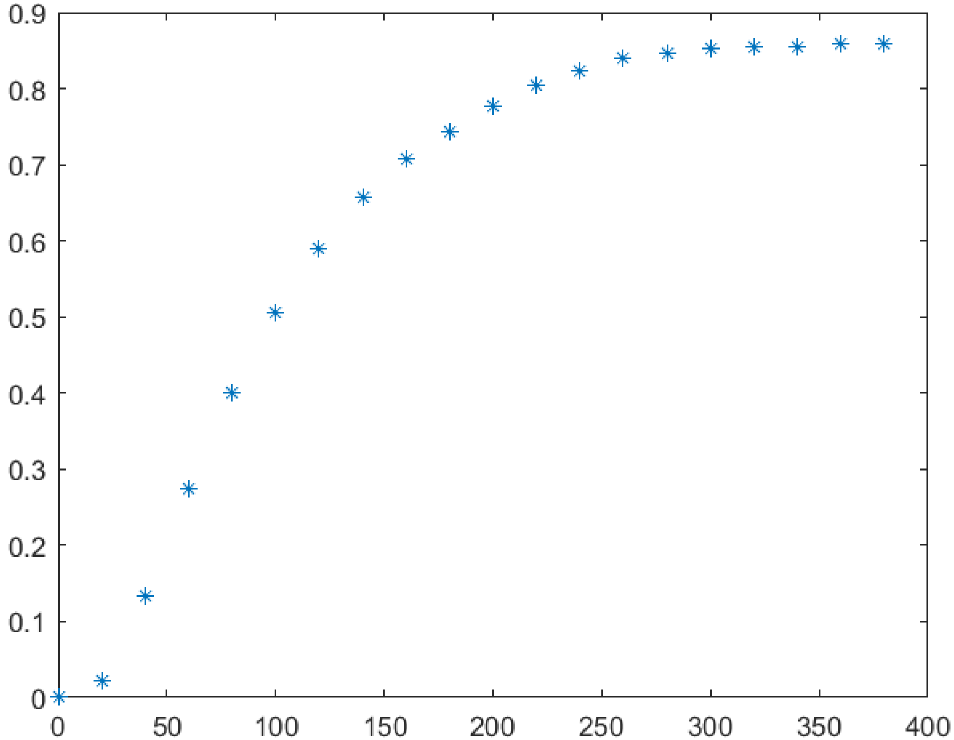
- ***Optimized parameters of the model : K= 0.87, Tm= 9.02 s, T1= 26.11 s, T2= 70.87 s***
- ***Model's poles : -0.0383 & -0.0141***






```
clear all
close all
load data_system_ok.mat
```

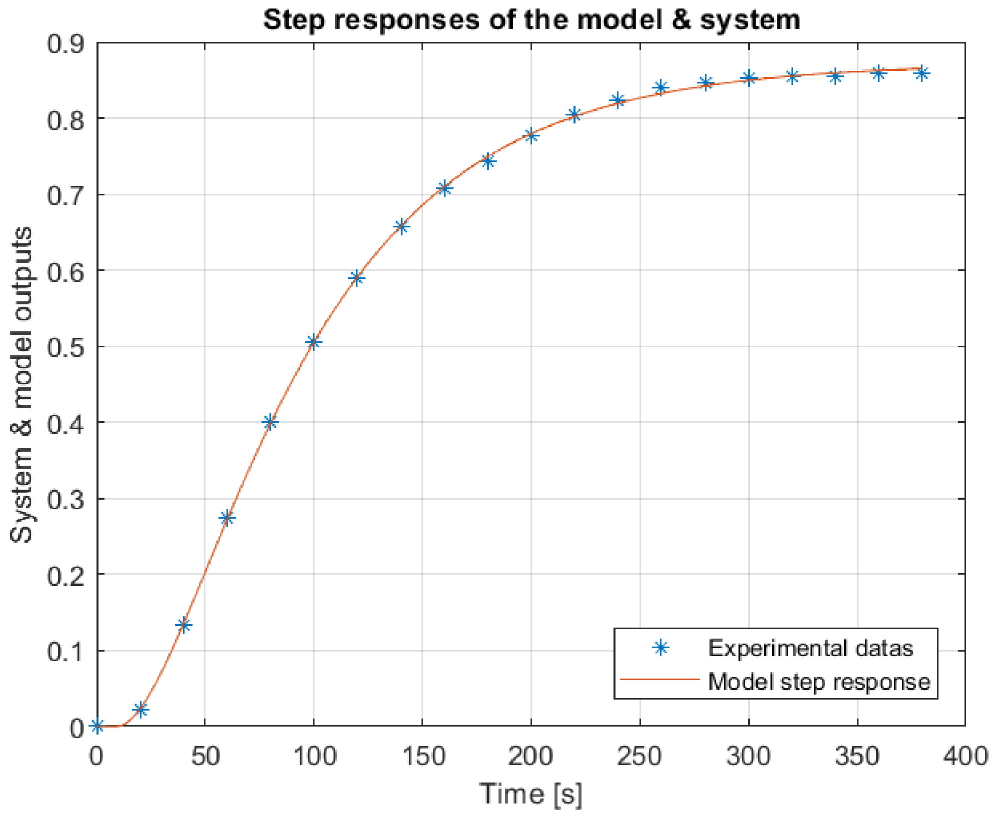
```
plot(t,y,'*')
```




```
s=tf('s');    % Laplace variable
fun=@(P,t) step(P(1)*exp(-s*abs(P(2)))/((s*P(3)+1)*(s*P(4)+1)),t);    %
Function handle computing the step answer of the TF for each time sample
contained in the vector t.
cost=@(P,t,fun,y) norm(fun(P,t)-y);    % Cost function
P_opt=fminsearch(@(P) cost(P,t,fun,y),[1 25 50 50])
```

```
P_opt =
    0.8724    9.0227   70.8665   26.1116
```

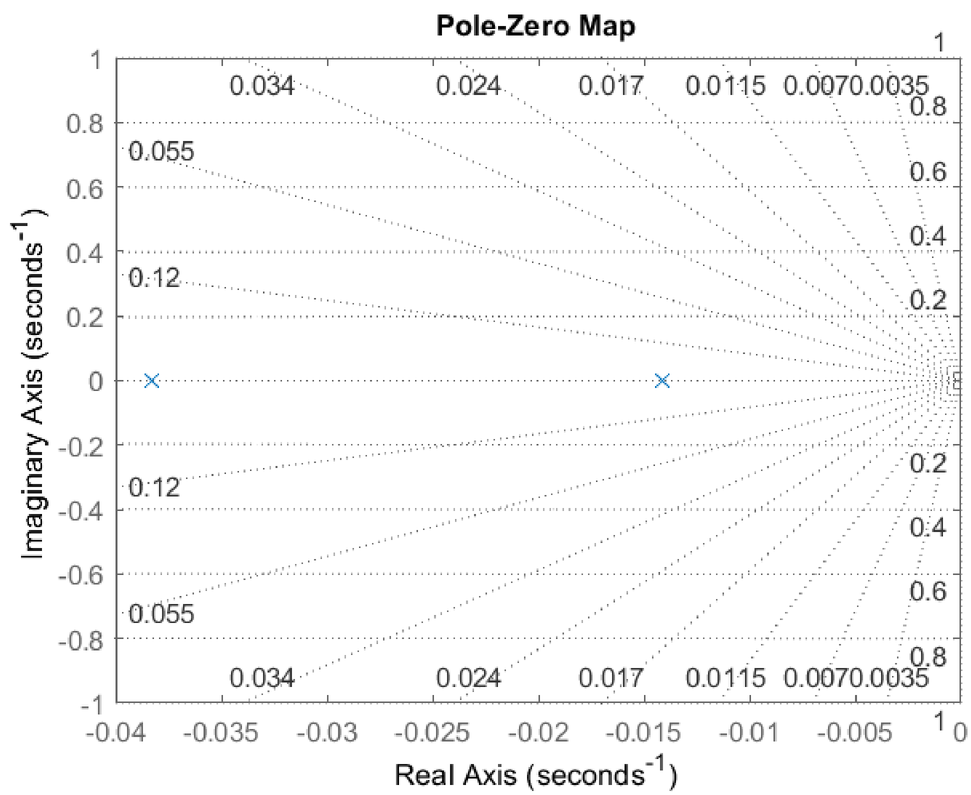
```
tm=linspace(0,t(end),500);    % Time vector creation
ym=fun(P_opt,tm);             % Computing of the optimized model output at
each time sample of the new time vector
plot(t,y,'*')                 % Plotting
hold on
plot(tm,ym)
legend('Experimental datas','Model step response','location','southeast')
xlabel('Time [s]')
ylabel('System & model outputs')
title('Step responses of the model & system')
grid
hold off
```




```
% Model's pole computation
H=P_opt(1)*exp(-s*P_opt(2))/((s*P_opt(3)+1)*(s*P_opt(4)+1));
pole(H)
```

```
ans =
   -0.0383
   -0.0141
```

```
pzmap(H)
grid
```



Pole-Zero Map
Imaginary Axis (seconds^{-1})
Real Axis (seconds^{-1})
0.034
0.024
0.017
0.0115
0.0070.0035
0.055
0.12
1
0.8
0.6
0.4
0.2
-0.04
-0.035
-0.03
-0.025
-0.02
-0.015
-0.01
-0.005
0
-0.2
-0.4
-0.6
-0.8
-1