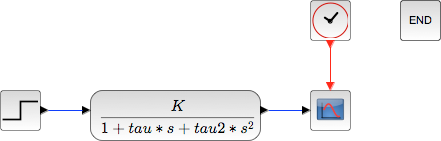
System parameters identification Scilab/Xcos

Cas d’usage classique :

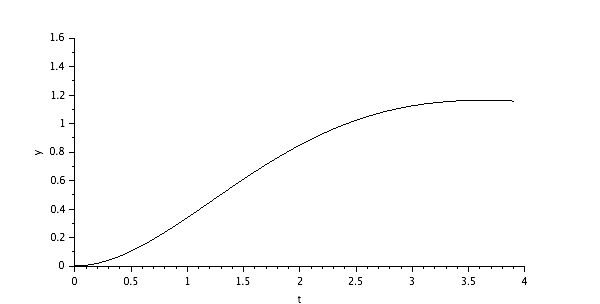
1. Acquisition de données de mon système en boucle ouverte à réguler
2. Modélisation
3. Identification des paramètres du système
4. Mise en place d’une stratégie de commande

*Exemple* :

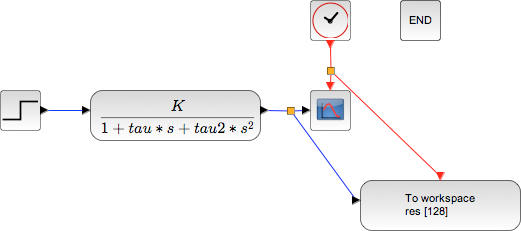
Système du second ordre



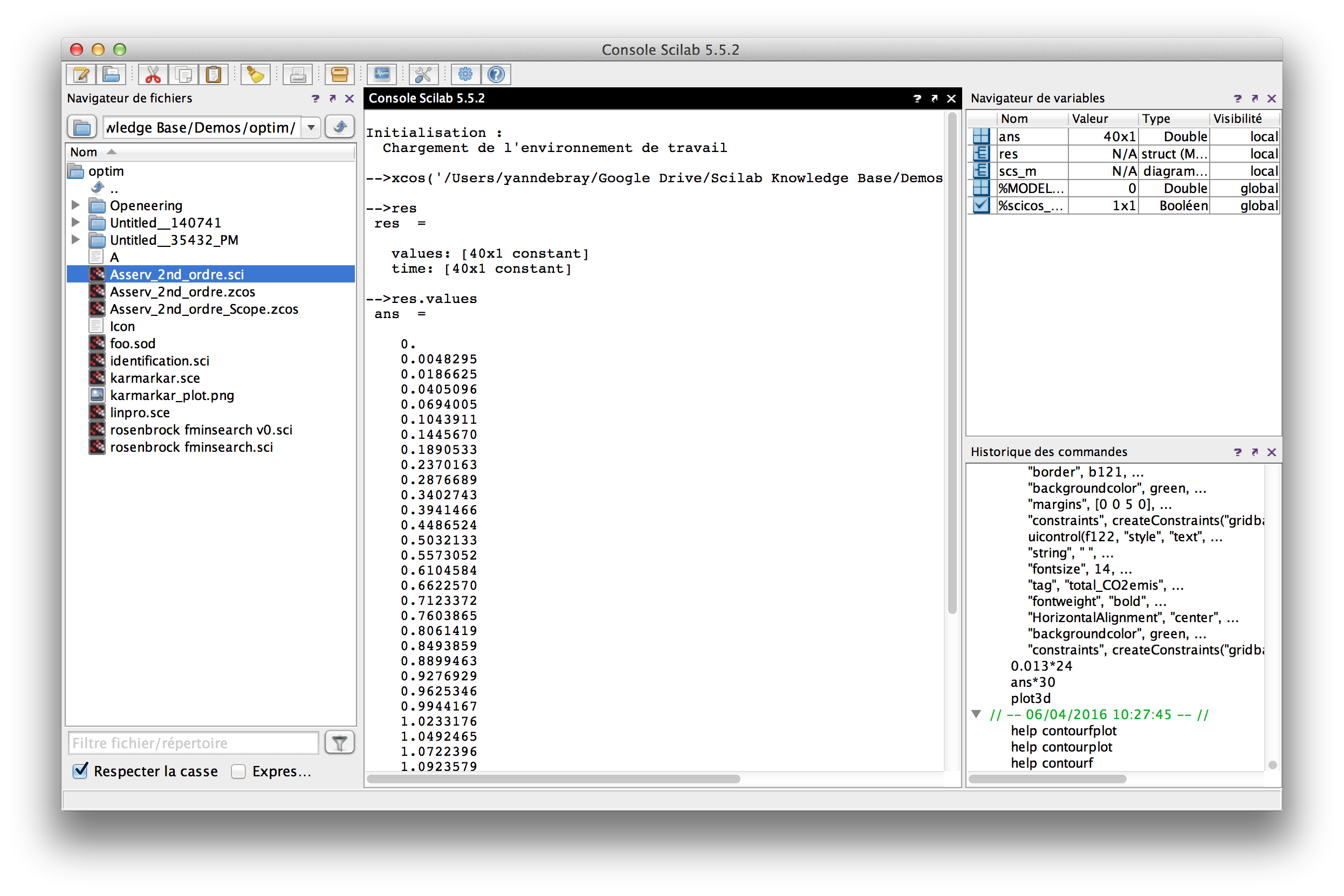
Lancement d’une simulation avec un échelon unitaire   
(avec pour paramètres K, Tau et Tau2 tous égaux à 1)



Introduction d’un bloc “to workspace” pour faire un retour des résultats de simulation dans l’environnement Scilab pour post-processing :



Ainsi ces données sont accessibles à travers une variable res, contenant 2 structures de valeurs (values) et de temps (time) :



Etape suivante : lancement d’un script d’optimisation, exécutant les simulations Xcos en batch, afin d’identifier les paramètres du système du second ordre (3 paramètres : K, Tau et Tau2).

Définition d’une fonction de coût y, déterminant la distance entre la courbe de réponse simulée du système et l’échelon d’entrée du système.

function **y**=cost\_for\_fminsearch(**x**)  
 context = ["K="+string(**x**(1)) "tau="+string(**x**(2)) "tau2="+string(**x**(3))];  
  
 *// On impose un cout infini si on sort de nos bornes de recherche.*  
 if **x**(1) < 0 | **x**(2) < 0 | **x**(3) < 0 then  
 **y**=%inf  
 return  
 end  
  
 try  
 *// Ouverture du diagramme => cree une variable scs\_m*  
 importXcosDiagram(pwd()+"/Asserv\_2nd\_ordre.zcos");  
 *// Changement du contexte*  
 scs\_m.props.context = context;  
 *// Simulation*  
 xcos\_simulate(scs\_m, 4);  
 catch  
 disp("Error during xcos Simulation ...");  
 error("cost function failed.")  
 **y** = %inf;  
 return  
 end  
  
 res\_ref = ones(40,1);  
 **y** = norm(res.values - res\_ref)^2;  
 e = get("costPolyline");  
   
 e.data(:, 2) = res.values;  
endfunction

Les résultats sont représentées dynamiquement à partir de ce script d’optimisation qui itère, jusqu’à trouver la solution optimale dans l’intervalle défini

f = gcf();  
plot([0, 3.9], [1, 1], "r");  
plot([0:0.1:3.9], zeros(40, 1));  
e = gce();  
e.children(1).tag = "costPolyline";  
a = gca();  
a.data\_bounds = [0, 0 ; 4, 1.8]

La fonction d’optimisation appelée afin d’identifier les 3 paramètres de ce système est [fminsearch](https://help.scilab.org/docs/5.5.2/fr_FR/fminsearch.html) :

opt = optimset( "Display" , "iter" );  
[x fval] = fminsearch ( cost\_for\_fminsearch , [0 0 0] , opt );

La section d’appel de cette fonction est :

* fonction de coût définissant la distance à la solution, qu’il convient de minimiser  
  ici : cost\_for\_fminsearch
* le point de départ de la recherche  
  ici : [0 0 0]
* des options avancées avec la fonction [optimset](https://help.scilab.org/docs/5.5.2/fr_FR/optimset.html)  
  ici : options.Display = "iter" → l’algorithme renvoit un message d’une ligne à chaque itération

Vous pouvez développer une interface utilisateur (à gauche) afin de modifier manuellement les paramètres du système ou lancer une optimisation automatique. En déployant cette application sur le Cloud, on obtient l’interface de droite.

