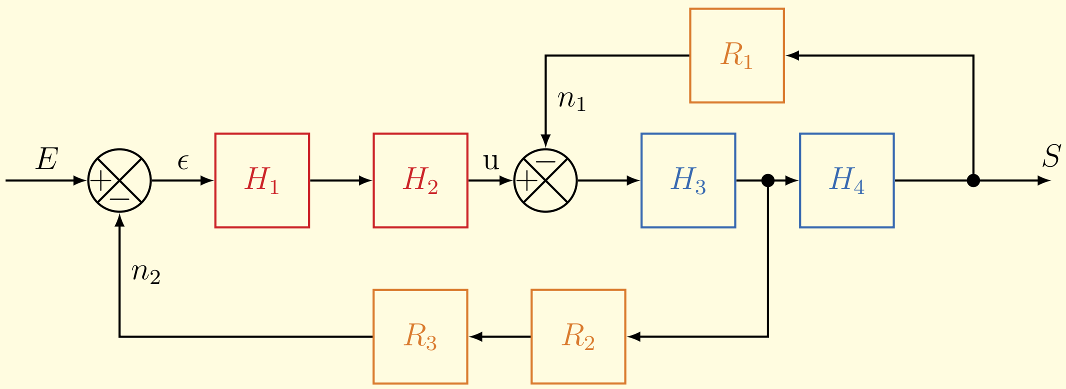
Systèmes mécaniques et automatiques

Notes de cours IngéSpé

Automatique Linéaire



Année 2020–2021

Systèmes mécaniques et automatiques

Notes de cours IngéSpé
Automatique Linéaire

Filipe Manuel Vasconcelos

écrit sous LaTeX, TikZ
version de Novembre2020.

Ce document est destiné aux étudiants du cycle prépa de l'ESME SUDRIA. En constante évolution, il ne pourra que s'améliorer avec votre concours. N'hésitez pas à me communiquer vos remarques et/ou corrections par mail : `filipe.vasconcelos@esme.fr`

Table des matières

Avant-propos

Programme

Ce cours est une introduction à l'**automatique** pour des étudiants de deuxième année de classe préparatoire scientifique.
L'objectif principal de l'automatique est de permettre le contrôle des **systèmes dynamiques** de toutes natures que ce soient : mécanique, chimique, électronique, optique, thermique, acoustique.... Tout en respectant certaines contraintes de performances (rapidité, précision, stabilité...).
Nous limiterons notre étude aux **systèmes linéaires continus et invariants.**
La **modélisation** de ces systèmes passe par la mise en équation du comportement physique des systèmes sous forme d'équations différentielles. Cette étape ne fait pas à proprement parler partie d'un cours d'automatique, en effet chacunes des disciplines construisent cette modélisation en se basant sur les principes et les hypothèses les plus adaptés à un problème donné. La modélisation permet une étude systématique des équations différentielles en proposant des modèles généraux et ce quelque soit la nature du procédé.
L'**analyse** nous permettra de caractériser et d'identifier ces modèles à partir des réponses aux sollicitations et de leurs performances.
Le **contrôle** est un concept très générale permettant de regrouper toutes les méthodes et techniques permettant de commander un système dynamique. Dans ce cours nous présenterons que les principes d'asservissement et de régulation. Nous verrons comment il est possible d'élaborer une commande adaptée (corrigée) pour un procédé quelconque, notamment lorsque ceux-ci présenterons des défauts de performance.

Organisation du document

Les chapitres suivent un découpage classique autour des trois pilliers discutés précedemment que sont la **modélisation**, l'**analyse** et le **contrôle**. (c.f Figure A). Le lecteur pourra s'appuyer sur un grand nombre d'annexes qui ont pour objectifs de rappeler et de détailler des notions prérequises ou encore approfondir quelques aspects hors programme pour une deuxième lecture.

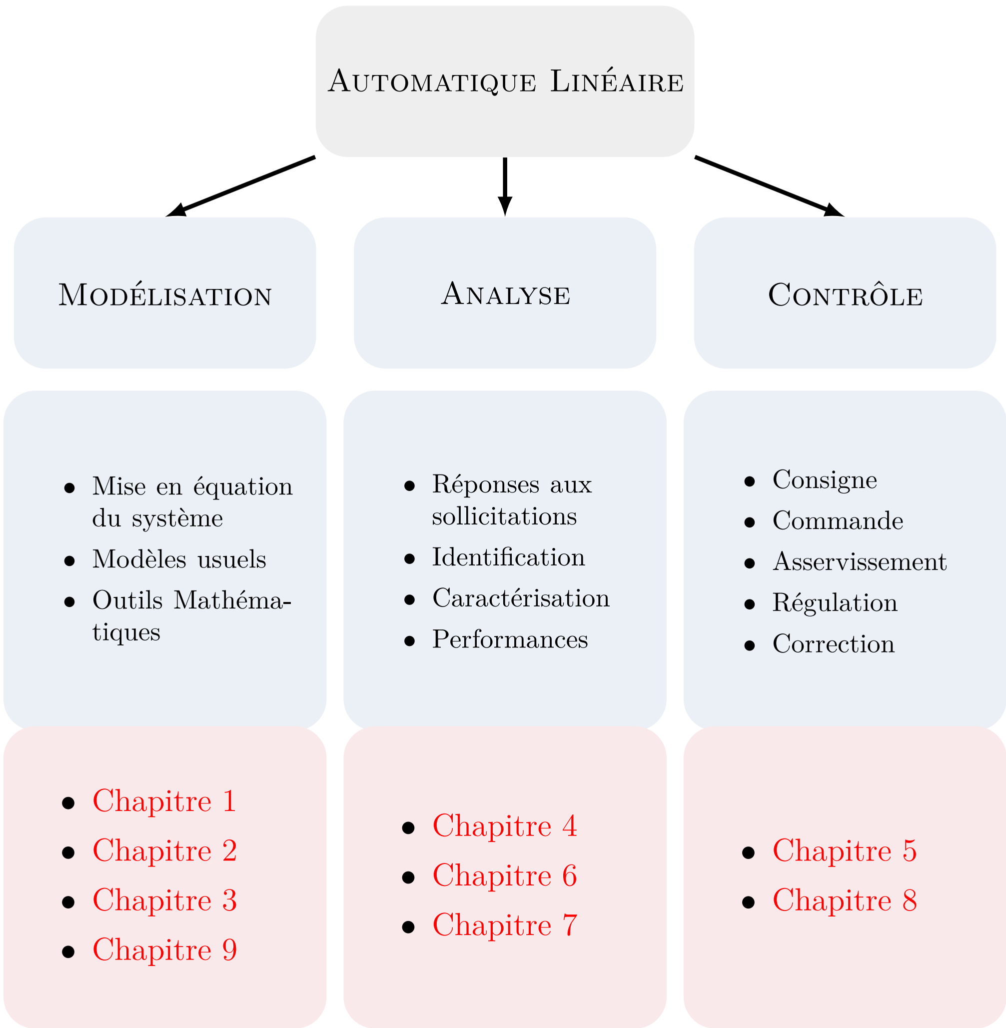


Figure A. – Organisation du document.

Annexes

A. Alphabet Grec

Nom	Minuscule	Majuscule	Correspondance latine	Usages courants
alpha	α	A	a	angles
bêta	β	B	b	angles
gamma	γ	Γ	g	angles
delta	δ	Δ	d	variations
epsilon	ϵ, ε	E	e	petite quantité
zéta	ζ	Z	z	-
êta	η	H	é (long)	rendement
thêta	θ, ϑ	Θ	th	angles
iota	ι	I	i	-
kappa	κ, $\varkappa$	K	k	-
lambda	λ	Λ	l	longueur, densité linéique
mu	μ	M	m	masse réduite
nu	ν	N	n	fréquence
ksi	ξ	Ξ	ks	coefficient sans dimension
omicron	o	O	o	-
pi	π, ϖ	Π	p	Π :plan
rhô	ρ, ϱ	P	r	densité volumique
sigma	σ, ς	Σ	s	σ : densité surfacique, Σ : Système
tau	τ	T	t	temps, durée relative
upsilon	υ	Y	u	-
phi	ϕ, φ	Φ	f,ph	angles
khi	χ	X	kh	coefficients
psi	ψ	Ψ	ps	fonction d'onde
oméga	ω	Ω	ô	vitesse angulaire, angle solide

Tableau A.1. – Lettres de l'alphabet Grec et leurs usages courants en physique (non exhaustifs)

J. Initiation à Scilab

1. Présentation générale (source Wikipédia)

Scilab est un logiciel libre de calcul numérique fournissant un environnement de calcul pour des applications scientifiques. Il est utilisé pour le traitement du signal, l'analyse statistique, le traitement d'images, les simulations de dynamique des fluides, l'optimisation numérique, et la modélisation et simulation de systèmes dynamiques.
La syntaxe et les possibilités offertes par Scilab sont similaires à celles de Matlab. Scilab peut exécuter des instructions en ligne de commande, ainsi que des fichiers de commande (scripts) contenant des instructions (format texte). On peut exécuter des programmes Fortran ou C à partir de Scilab. Scilab est complété par un environnement graphique Xcos comparable à l'environnement graphique Simulink fourni avec Matlab.

2. Syntaxe : console

Les instructions sont tapées après le prompt `-->`. Le résultat est donné sauf si l'instruction se termine par un point-virgule auquel cas le résultat est caché. Les variables sont sensibles à la casse. `ans` est la variable qui contient le dernier résultat. Un commentaire commence par `\\`. La plupart des opérateurs sont communs avec d'autres langages (affectation `=`, addition `+`, soustraction `-`, multiplication `*`, puissance `** ou ^`).

Exemples :

```
-->// ceci n'est pas un commentaire

-->a=2
 a  =

    2.
-->A=3;  \\ le résultat est caché
```

```
-->a*A   \\ produit
 ans  =

    6.
-->ans**2 \\ utilisation de la valeur précédente
 ans  =

    36.
```

Il existe quelques variables prédéfinies sous Scilab, elles sont appelées à l'aide du symbole « `%` ». La liste de ces variables est accessible par la fonction `whos -name %`

```
%e    // constante d'Euler
%eps  // précision machine epsilon
%F %f // booléen false 0
%T %t // booléen True 1
%pi   // constante pi
%i    // nombre imaginaire
%inf  // infini
%nan  // not-a-number
%s %z // définition polynôme
```

Quelques opérateurs logiques :

```
-->A=0;

-->B=1;

-->A&B    // ET logique
 ans  =

  F

-->A|B    // OU logique
 ans  =

  T

-->~A     // NON logique
 ans  =
```

```
  T

-->A==B   // Egalité logique
 ans  =

  F

-->A~=B   // Différence logique
 ans  =

  T
```

3. Polynômes et fractions rationnelles

En automatique, la définition de la fonction de transfert fait intervenir des polynômes et des fractions rationnelles.
La déclaration d'un polynôme se fait à l'aide de deux instructions. La première utilise la fonction `poly(0, "p")` qui définit « `p` » comme l'indéterminée d'un polynôme. La seconde instuction est l'énoncé du polynôme utilisant cette indéterminée.

Exemple :
On définit deux polynômes $D(p) = 1 + 2p + 3p^2$ et $N(p) = 1 + p$ de la façon suivante :

```
-->p = poly(0, 'p');

-->D=(1+2*p+3*p**2)
 D  =

               2
    1 + 2p + 3p

-->N=1+p
 N  =

    1 + p
```

La fonction `roots(D)` donne les racines du polynôme $D(p)$:

```
-->roots(D)
 ans  =

  - 0.3333333 + 0.4714045i     // nombre complexe
  - 0.3333333 - 0.4714045i     //
-->roots(N)
 ans  =

   - 1.
```

Scilab gère de la même manière les fractions rationnelles :

```
-->H=N/D
 H  =

       1 + p
    -----------
               2
    1 + 2p + 3p
```

Il est possible d'extraire les numérateurs et dénominateurs de $H(p) = N(p)/D(p)$ par les variables `H.num` et `H.den` respectivement :

```
-->roots(H.den)               // pôles
 ans  =

  - 0.3333333 + 0.4714045i
  - 0.3333333 - 0.4714045i

-->roots(H.num)               // zéros
 ans  =

  - 1.
```

Il est également possible de définir un polynôme à partir de ces racines ou de ces coefficients en appelant seulement la fonction `poly`.

À partir de ses racines :

```
-->poly([1 2],'p')    // polynôme dont les racines sont 1 et 2
 ans  =

             2
    2 - 3p + p
```

À partir des ses coefficient :

```
-->poly([1 2],'p','c')    // option 'c' nécessaire
 ans  =

    1 + 2p
```

Remarque :
Lorsque que l'on écrit `p=poly(0,'p')`, on définit la variable `p` comme le polynôme dont la racine est nulle.

```
-->p=poly(0,'p')
 ans  =

    p
```

4. Vecteurs et matrices

Pour définir un vecteur ligne :

```
--> v=[1,2,3]
 v  =

    1.    2.    3.

-->v=1:3
 v  =

    1.    2.    3.
```

```
-->v=1:0.5:4                // a:incr:b
 v  =                       // liste allant de a à b par incrément
                            // de incr

    1.    1.5    2.    2.5    3.    3.5    4.
```

Pour définir un vecteur colonne :

```
--> v=[1;2;3]
 v  =

    1.
    2.
    3.
```

Nous combinons les deux syntaxes précédentes pour définir une matrice :

```
--> m=[1 2; 3 4 ; 5 6]
 m  =

    1.    2.
    3.    4.
    5.    6.
```

Le vecteur ou la liste nul est simplement déclaré par :

```
-->liste=[];
```

Les parenthèses permettent d'accéder aux élément d'une matrice et le symbole « : » permet d'accéder à toute ou une partie d'une ligne ou d'une colonne

```
-->v(1)
 ans  =

    1.

-->m(2,1)
 ans  =

    3.
--> // accéder à une ligne entière
```

```
-->m(1,:)
 ans  =

     1.    2.
```

Pour concaténer deux vecteurs ou matrices :

```
-->u=[1,2,3];

-->v=[4,5,6];

-->w=[v,u]
 w  =

    4.    5.    6.    1.    2.    3.

-->m1=[1 2; 3 4 ; 5 6]

-->m2=[1 2; 3 4 ; 5 6]

-->m3=[m1,m2]
 m3  =

    1.    2.    1.    2.
    3.    4.    3.    4.
    5.    6.    5.    6.
```

Les opérateurs mathématiques de bases peuvent être utilisés sur les vecteur et matrices mais doivent respecter une certaine cohérence des dimensions des objets de chaque coté de l'opérateur. Notamment, les opérations `*` et `/` sont des opérations matricielles. Pour faire des opérations élément par élément, on fera précéder le signe opératoire d'un point : `.* ./`.

Exemple :

```
// définition de l'indéterminée du polynôme
-->p = poly(0, 'p');

//définition d'un vecteur de numérateurs
-->num=[1 10 20];
```

```
//définition d'un vecteur de dénominateurs
-->den=[p*(p+1) p*(p+10) p*(p+20) ];

//définition d'un vecteur de fonctions de transfert
-->H=num ./ den
 H  =

      1         10          20
    -----     ------      ------
         2          2           2
    p + p     10p + p     20p + p
```

Scilab a été conçu pour le calcul matriciel et numérique. Il existe de nombreuses opérations spécifiques aux matrices et à la résolution numériques que nous ne traiterons pas dans ce document.
Pour accéder à l'aide en ligne, cliquez sur `? >Aide Scilab` dans la barre de menus, ou tapez dans la console :

```
-->help <fonction>
```

5. Tracer de figures

On utilisera les fonctions `scf` et `clf` pour respectivement créer une nouvelle fenêtre graphique et éffacer le contenu d'une fenêtre graphique.

Exemple : plot de deux fenêtres (sinus/cosinus)

```
scf(0);clf(0); // création d'une première fenêtre
t=0.0:0.05:100;
plot(t,sin(t),'r')
legend('$s_1(t)$','$e_1(t)$')
xlabel("$t$","fontsize",4);
ylabel("$s_2(t)$","fontsize",4);
title('fonction sinus',"fontsize",4);

scf(1);clf(1); / création d'une seconde fenêtre
t=0.0:0.05:100;
```

```
plot(t,cos(t),'r')
legend('$s_2(t)$','$e_1(t)$')
xlabel("$t$","fontsize",4);
ylabel("$s_2(t)$","fontsize",4);
title('fonction cosinus',"fontsize",4);
```

Les textes des légendes, titre des axes acceptent la syntaxe LaTeX.
Il existe de nombreuses commandes pour modifier l'apparence d'une figure, de ces axes et pour pouvoir la sauvegarder dans différents formats (vectorielle ou matricielle). Nous renvoyons au lecteur à la documentation de Scilab pour cet aspect. Il est également possible d'éditer une image en accédant au menu de la fenêtre graphique après l'avoir générée.

6. Programmation (source Wikibooks)

Scilab est également un langage de programmation, il accepte un certain nombre d'instructions autres que mathématiques, permettant la formulation et l'exécution d'algorithmes : `for, while, if, do, do, case`... ou définition de fonction.
L'écriture de programmes se fait idéalement avec l'éditeur de texte SciNotes ; celui-ci met en exergue les instructions en couleurs, les parenthésages (correspondance entre les paires de parenthèses et de crochets), et surligne les lignes continuées avec un fond jaune. On peut aussi utiliser un autre éditeur de texte en sauvegardant le fichier avec l'extension `.sce` ou `.sci`. Lorsque l'environnement le permet, on peut faire du copier-coller depuis l'éditeur de texte externe vers SciNotes ou bien l'éditeur de ligne de commande.

Syntaxe d'une fonction :

La fonction doit commencer par le mot réservé `function` et finir par `endfunction` sous la forme :

```
function [out1,out2,...,outN]=nomfonction(in1,in2,...,inP)

  // out1,out2,...,outN sont les variables de sortie
  // in1,in2,...,inP variables d entree

<instructions>
endfunction
```

Une façon usuelle de définir des fonctions est de mettre celles-ci dans un fichier à extension `.sci`. Il faut alors la charger avec la fonction `exec()`.

Appel d'une fonction :

Pour exécuter une fonction il suffit de l'appeler en passant les arguments nécessaires.

```
function [u,v]=mafonction(a,b)
    u=exp(a)
    v=u*sin(b)
endfunction

-->mafonction(rand(),rand())
```

L'appel précédent ne renvoie que la valeur de `u`. Pour obtenir les deux valeurs escomptées il faut faire un appel sous la forme :

```
-->[u,v]=test_function(rand(),rand())
 v  =

    0.5695456
 u  =

    1.0707657
```

7. Slci avec Scilab

Scilab permet de réaliser des études avancées des systèmes linéaires continus et invariants.

7.1. Définition d'un système linéaire

Fonction syslin (extrait de la doc officiel : help syslin)

- Syntaxe :

```
sl=syslin(dom,N,D)
sl=syslin(dom,H)
```

- Paramètres :
  - `dom` : chaîne de caractères (`'c'`,`'d'`), ou `[]` ou un scalaire.
  - `N,D` : matrices polynomiales

  - `H` : matrice rationnelle
  - `sl` : tlist (liste de type "`syslin`") représentant le système dynamique
- Description :
  - `syslin` définit un système dynamique linéaire en tant que liste typée, et vérifie la consistance des données.
  - `dom` spécifie le domaine temporel : `dom='c'` pour un système à temps continu, `dom='d'` pour un système à temps discret, `n` pour un système échantillonné à la période n (en secondes), `dom=[]` si le domaine temporel n'est pas défini

Exemple d'utilisation

On souhaite

7.2. Simulation temporelle d'un système linéaire

Fonction csim (extrait de la doc officiel (en anglais) : help csim)

- Syntax :

```
[y [,x]]=csim(u,t,sl,[x0 [,tol]])
```

- Parameters :
  - `u` function, list or string (control)
  - `t` real vector specifying times with, `t(1)` is the initial time `(x0=x(t(1)))`.
  - `sl` syslin list (SIMO linear system) in continuous time.
  - `y` a matrix such that `y=[y(t(i)], i=1,..,n`
  - `x` a matrix such that `x=[x(t(i)], i=1,..,n`
  - `tol` a 2 vector `[atol rtol]` defining absolute and relative tolerances for `ode` solver
- Description :
  - `csim` simulation of the controlled linear system `sl`. `sl` is assumed to be a continuous-time system represented by a `syslin` list.
  - `u` is the control and `x0` the initial state.

- `y` is the output and `x` the state.

The control can be :

- a function : `[inputs]=u(t)`
- a list : `list(ut,parameter1,....,parametern)` such that : `inputs=ut(t,parameter1,....,parametern)` (`ut` is a function)
- the string `"impuls"` for impulse response calculation (here `sl` must have a single input and `x0=0`). For systems with direct feed-through, the infinite pulse at `t=0` is ignored.
- the string `"step"` for step response calculation (here `sl` must have a single input and `x0=0`)
- a vector giving the values of `u` corresponding to each `t` value.

7.3. Système du premier ordre

Soit, un système du premier ordre définit par la fonction de transfert :

$$H(p) = \frac{K}{1+\tau p}$$

La définition sous Scilab de ce système se fait simplement par les quelques instructions suivantes :

```
// ======================================
//  Définir un système du premier ordre
// ======================================

p=poly(0,'p');
K=1,tau=1;                          // paramètres du système

H=K/(1+tau*p);                      // fonction de tranfert

PremierOrdre=syslin('c',H)          // définition du SLCI
```

Nous allons maintenant étudier les réponses temporelles à différentes excitations du système du premier ordre.

7.3.1. Réponse impulsionnnelle

```
// ------------------------
// réponse impulsionnelle
// ------------------------
e2='imp'                        // 'imp' : dirac
s2=csim(e2,t,PremierOrdre);

scf(1);clf(1);
plot(t,s2,'r')

legend('$s_2(t)$','$e_2(t)$')
xlabel("$t$","fontsize",4);
ylabel("$s(t)$","fontsize",4);
title('réponse impulsionnelle',"fontsize",4);
```

7.3.2. Réponse indicielle

```
t=0.0:0.05:20;                  // définition du vecteur
                                // de temps
// ---------------------
//  réponse indicielle
// ---------------------
e1='step'                       // 'step' : échelon
s1=csim(e1,t,PremierOrdre);

// clf : effacer le contenu de
// la fenêtre graphique
// scf : creer une nouvelle
// fenêtre graphique

scf(0);clf(0);
plot(t,s1,'r')

legend('$s_1(t)$','$e_1(t)$')
xlabel("$t$","fontsize",4);
ylabel("$s(t)$","fontsize",4);
title('réponse indicielle',"fontsize",4);
```

7.3.3. Réponse à une excitation sinusoïdale

```
// ------------------------------------------
// réponse à une excitation sinuso\"idale
// ------------------------------------------
e3=sin(t)
s3=csim(e3,t,PremierOrdre);

scf(2);clf(2);
plot(t,s3,'r',t,e3,'b')

legend('$s_3(t)$','$e_3(t)$')
xlabel("$t$","fontsize",4);
ylabel("$s(t)$","fontsize",4);
title('réponse harmonique',"fontsize",4);
```

Ci-dessous nous présentons une façon d'étudier la réponse temporelle pour différentes valeurs d'un des paramètres du système du premier ordre.

```
scf(3);clf(3);
for tau=1:1.0:10.
    H2=K/(1+tau*p)
    PremierOrdre=syslin('c',H2)
    e1='step'
    s1=csim(e1,t,PremierOrdre);
    plot(t,s1,'r')
end
```

7.3.4. Réponse fréquentielle

Scilab permet de tracer facilement les différents diagrammes de la réponse fréquentielle d'un système. Nous donnons ici les fonctions les plus importantes :

```
fMin =0.01,fMax=100;
p=poly(0,'p')
K=1.,tau=1.;
H=K/(1+tau*p);
PremierOrdre=syslin('c',[K],[1+tau*p])

// diagrammme de Bode
```

```
scf(0);clf(0);
bode(PremierOrdre,fMin,fMax); bode_asymp(PremierOrdre,fMin,fMax);

// diagrammme de Nyquist
scf(1);clf(1);
nyquist(PremierOrdre) ;

// diagramme de Black
scf(2);clf(2);
black(PremierOrdre,0.01,10);
nicholschart(colors=color('gray')*[2 2]) //abaque de Black

// Lieu de Evans
scf(3);clf(3);
evans(PremierOrdre) ;
```

7.4. Carte des pôles et zéros

Fonction plzr (extrait de la doc officiel (en anglais) : help plzr)

- Syntax :

```
plzr(sl)
```

- Arguments :
  - `sl` syslin list (SIMO linear system) in continuous time.
- Description :
  - `plzr(sl)` produces a pole-zero plot of the linear system sl.

7.5. Asservissement

Fonction feedback (extrait de la doc officiel (en anglais) : help feedback)

- Syntax :

```
Sl=Sl1/.Sl2
```

- Parameters :
  - `Sl1,Sl2` linear systems (syslin list) in state-space or transfer form, or ordinary gain matrices.
  - `Sl` linear system (syslin list) in state-space or transfer form.
- Description :
  - The feedback operation is denoted by /. (slashdot).
  - This command returns `Sl=Sl1*(I+Sl2*Sl1)^-1`, i.e the (negative) feedback of `Sl1` and `Sl2`. `Sl` is the transfer `v -> y` for `y = Sl1 u`, `u = v - Sl2 y`.
  - The result is the same as `Sl=LFT([0,I;I,-Sl2],Sl1)`.
  - Caution : do not use with decimal point (e.g. `1/.1` is ambiguous !)

8. Scilab-Xcos

Nous présentons ici le module Xcos intégré à Scilab. Xcos inclut un éditeur graphique pour facilement représenté les schémas fonctionnels en connectant des blocs entre eux. Cette présentation/tutoriel étant loin d'être complète, nous renvoyons le lecteur à la documentation officielle de Xcos pour obtenir davantage de détails [21, 2, 3]. petit tuto suivant.)

8.1. Lancer Xcos

Après avoir lancé Scilab, éxécuter une des instructions suivantes :

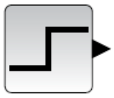
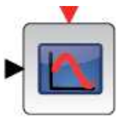
- Taper la commande `xcos` dans la console ;
- Cliquer sur l'icône : 
- Menu → Applications → Xcos

Xcos ouvre par défaut le navigateur de palettes et une fenêtre d'édition. Pour construire le diagramme il suffit de faire glisser les blocs dans la fenêtre d'édition.

8.2. Diagramme simple

Le module inclut un grand nombre de blocs (c.f Navigateur de palettes). Il est possible de construire des super-blocs qui incorpore d'autres blocs pour faciliter la lecture d'un diagramme complexe.

Nous allons créer un diagramme simple. Pour celà, placer les blocs suivants dans la fenêtre d'édition :

Désignation	Représentation	Sous-palette
Échelon		Sources / STEP_FUNCTION
Fonction de transfert continue	$\frac{1}{1+s}$	Systèmes à temps continu / CLR
Horloge		Sources / Clock_c
Visualisation		Sinks / CSCOPE

Connecter les blocs pour obtenir le schéma bloc Xcos de la figure J.1.
Il faut maintenant simuler et visualiser les résultats.

8.3. Simulation

Pour lancer une simulation : cliquer sur l'icône : 
Pour arrêter une simulation : cliquer sur l'icône : 
Plusieurs paramètres peuvent être ajustés :

- La durée de la simulation : Simulation → Configurer → Temps d'intégration final
- La période d'échantillonage : Cliquer sur l'horloge.
- La fonction échelon : Cliquer sur le bloc de la fonction échelon
- Les paramètres de la fonction de transfert : Cliquer sur le bloc CLR (N'oubliez pas d'ajouter un contexte si vous utiliser des variables)

8.4. Blocs « To Workspace » ou « From Workspace »

On utilisera les blocs particuliers dans le cas où l'on souhaite utiliser des données à partir de Scilab (« From Workspace ») ou récupérer ces données après la simulation (« To Workspace »).

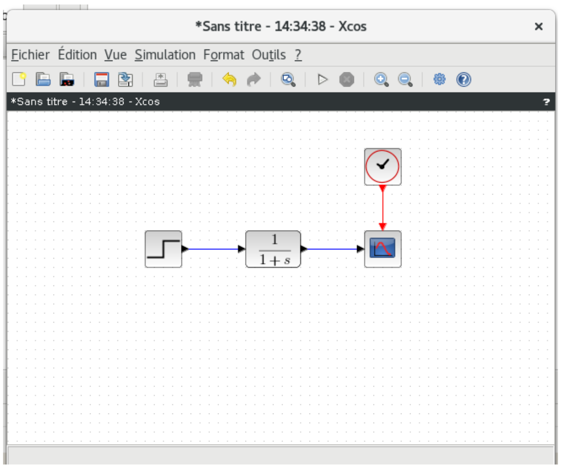


Figure J.1. – Exemple de diagramme simple

Figure J.2. – Blocs d'échange avec Scilab

Références

[1] Régulation automatique (analogique) (REG). http://php.iai.heig-vd.ch/~mee/.

[2] http://www.demosciences.fr/projets/scilab-xcos/-utilisation/premiers-pas.

[3] Xcos pour les vrais debutants. https://scilab.developpez.com/tutoriels/debuter/apprendre-xcos-debutant/.

[4] Denis Arzelier. Représentation et analyse des systèmes lineaires (pc7bis), 2005.

[5] B. Bayle and J. Gangloff. Systèmes et asservissements à temps continu, 2009.

[6] S. L. Campbell, J.-P. Chancelier, and R. Nikoukhah. *Modeling and Simulation in Scilab/Scicos*. Springer, 2006.

[7] H. Garnier. http://w3.cran.univ-lorraine.fr/hugues.garnier/?q=content/teaching.

[8] Y. Granjon. *Automatique : systèmes linéaires, non linéaires, à temps continu, à temps discret, représentation d'état, événements discrets*. Dunod, Paris, 2015.

[9] E. Laroche and H. Halalchi. Asservissement des systèmes lineaires à temps continu. http://eavr.u-strasbg.fr/~laroche/student.

[10] O. Le Gallo. *Automatique des systèmes mécaniques : Cours, travaux pratiques et exercices corrigés*. Sciences de l'ingénieur. Dunod, 2009.

[11] Joe Mabel. Régulateur à boules au Georgetown PowerPlant Museum à Seattle. CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=5694146.

[12] B. Marx. Outils Mathématiques pour l'ingénieur - Traitement du Signal. http://w3.cran.univ-lorraine.fr/perso/benoit.marx/enseignement.html.

[13] B. Marx. Contrôle des systèmes linéaires. http://w3.cran.univ-lorraine.fr/perso/-benoit.marx/enseignement.html.

[14] F. Orieux. *Automatique : Systèmes linéaires et asservissements*. Notes de Cours, Master 2 Outils et systèmes de l'astronomie et de l'Espace, 20017-1018.

[15] E. Ostertag. *Systèmes et asservissements continus : Modélisation, analyse, synthèse des lois de commande*. Ellipses Marketing, 2004.

[16] R. Papanicola. Schéma-blocs avec PGF/TIKZ. https://sciences-indus-cpge.papanicola.info/IMG/pdf/schema-bloc.pdf.

[17] R. Papanicola. *Sciences industrielles PCSI : Mécanique et automatique*. Ellipses Marketing, 2003.

[18] R. Papanicola. *Sciences industrielles PSI : Mécanique et automatique*. Ellipses Marketing, 2010.

[19] Marsyas-Travail personnel. Clepsydre athénienne reconstituée, Musée de l'Agora antique d'Athènes. CC BY-SA 2.5, https://commons.wikimedia.org/w/index.php?curid=476174.

[20] Consortium Scilab. Introduction to Scilab. www.scilab.org/content/download/247/1702/file/introscilab.pdf.

[21] S. Steer and Y. Degré. *Scilab : De la théorie à la pratique - II. Modéliser et simuler avec Xcos*. Éditions D-BookeR, 2014.

[22] C. Sueur, P. Vanheeghe, and P. Borne. *Automatique des systèmes continus*. Editions Technip.

[23] E. Thomas. TP Scilab. http://cpgeptljg.free.fr/scenari/TP_INFO/TP_info_12_ordre/co/module_TP_1_2_ordre_5.html.

Acronymes

DES Décomposition en Éléments Simples

FTBF Fonction de Transfert en Boucle Fermée

FTBO Fonction de Transfert en Boucle Ouverte

FTCD Fonction de Transfert de la Chaîne Directe

FTCR Fonction de Transfert de la Chaîne de Retour

MEI Matière-Énergie-Information

MIMO Multiple Input Multiple Output

SISO Single Input Single Output

SLCI Système Linéaire Continu et Invariant

TL Transformée de Laplace

Glossaire

Asservissement L'asservissment consiste à contrôler un système dynamique pour que sa réponse temporelle suive une consigne variable au cours du temps.

Régulation La régulation est un particulier d'asservissement consistant à garder une consigne constante en présence de perturbation.

Liste des Symboles

t	Variable temporelle
p	Indéterminée de polynôme
$s(t)$	Fonction/Signal dans le domaine temporel
$S(p)$	Fonction/Signal dans le domaine de Laplace de la fonction $s(t)$
$u(t)$	Fonction échelon unité ou de Heaviside
$\delta(t)$	Distribution de Dirac
$r(t)$	Fonction rampe unité
$\mathscr{L}\{f(t)\}$	Transformation de Laplace de la fonction $f(t)$
$\mathscr{L}^{-1}\{F(p)\}$	Transformation de Laplace inverse de la fonction $F(p)$
$H(p)$	Fonction de transfert
$N(p)$	Polynôme du numérateur d'une fraction rationnelle
$D(p)$	Polynôme du dénominateur d'une fraction rationnelle
ω	Pulsation
$H(j\omega)$	Nombre complexe associé à la fonction de transfert $H(p)$
E_0	Paramètre dimensionnelle d'amplitude de l'entrée
K	Gain statique
ω_0	Pulsation propre

$\mathrm{Im}[H(j\omega)]$	Partie imaginaire du nombre complexe $H(j\omega)$
$\mathrm{Re}[H(j\omega)]$	Partie réelle du nombre complexe $H(j\omega)$
ξ	Coefficient d'amortissement
$G(\omega)$	Gain naturel de la réponse harmonique en fonction de la pulsation
$G_{dB}(\omega)$	Gain en dB de la réponse harmonique en fonction de la pulsation
$\phi(\omega)$	Déphasage de la réponse harmonique en fonction de la pulsation
D_k	k-ème dépassement
$t_{5\%}$	Temps de réponse à 5%