Université de Ferhat Abbas Sétif 1

Faculté de technologie

Département de l'électrotechnique

TP : TEC81

Modélisation & Identification des Systèmes Electriques

Rapport de TP

Réaliser par :

-khoudri Aymen

-Guehlouz Hamza

-Zakaria Boulkraa

2018 /2019

Convertir la fonction de transfert en modèle d'espace d'état

Nous utilisons l'instruction :

Ss :Créer un modèle d'état, convertir en modèle d'état

Syntaxe

sys = ss (A, B, C, D)

La description

Utilisez ss pour créer des modèles d'espace-état (objets de modèle ss) avec des matrices à valeurs réelles ou complexes ou pour convertir des modèles de système dynamiques en formulaire de modèle d'espace-état. Vous pouvez également utiliser ss pour créer des modèles généralisés d'état-space (gens).

Création de modèles d'espace d'état

sys = ss (A, B, C, D) crée un objet de modèle d'espace d'état représentant le modèle d'espace d'état en temps continu

$$\dot{x}=Ax+Bu$$
$$y=Cx+Du$$

Pour un modèle avec des états Nx, des sorties Ny et des entrées Nu:
A : est une matrice à valeurs réelles ou complexes à Nx-Nx.
B :est une matrice à valeurs complexes ou réelles Nx-by-Nu.
C :est une matrice à valeurs réelles ou complexes de type Ny-by-Nx.
D : est une matrice à valeurs réelles ou complexes de type Ny-by-Nu

Exemple :

```
clc
clear
num=[2 8 6];den=[1 8 16 6];
sys_tf=tf(num,den)
sys_ss=ss(sys_tf)
```

```
sys_tf =

   2 s^2 + 8 s + 6

 ----------------------

 s^3 + 8 s^2 + 16 s + 6

Continuous-time transfer function.

sys_ss =

a =

      x1   x2   x3

  x1   -8   -4 -1.5

  x2    4    0    0
```

```
 x3   0   1   0

b =

   u1

 x1  2

 x2  0

 x3  0

c =

    x1   x2   x3

 y1   1    1  0.75

d =

   u1

 y1  0
```

Continuous-time state-space model.

ss2tf :

Convertir la représentation d'espace d'état en fonction de transfert

syntaxe

```
[b,a] = ss2tf(A,B,C,D)
[b,a] = ss2tf(A,B,C,D,ni)
```

La description

[b, a] = ss2tf (A, B, C, D) convertit une représentation d'espace d'état d'un système en une fonction de transfert équivalente. ss2tf renvoie la fonction de transfert de Laplace-transform pour les systèmes à temps continu et la fonction de transfert de Z-transform pour les systèmes à temps discret.
[b, a] = ss2tf (A, B, C, D, ni) renvoie la fonction de transfert créée lorsque l'entrée multiple d'un système à plusieurs entrées est excitée par une impulsion unitaire.

Exemple :

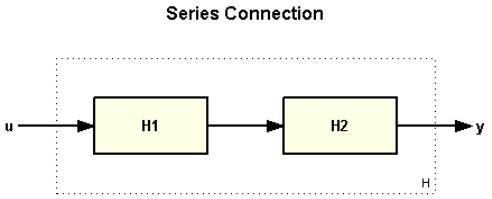
```
A=[0 -2;1 -3]
B=[2;0]
C=[1 0]
D=[0]
[b,a] = ss2tf(A,B,C,D)
num=[b];den=[a];
sys_tf=tf(num,den)

A =     0     -2
```

```
        1     -3
B =
      2
      0

C =      1      0
D =      0
sys_tf =
      2 s + 6
  -------------
  s^2 + 3 s + 2
Continuous-time transfer function.
```

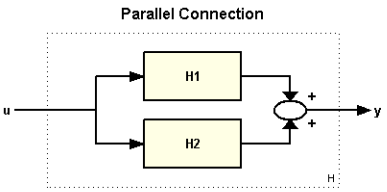
Connexion en série



Utilisez l'opérateur * ou la fonction série pour connecter des modèles LTI en série, par exemple:

```
H = H2 * H1
H = series(H1,H2);
```

Connexion parallèle



Utilisez l'opérateur + ou la fonction parallèle pour connecter des modèles LTI en parallèle, par exemple:

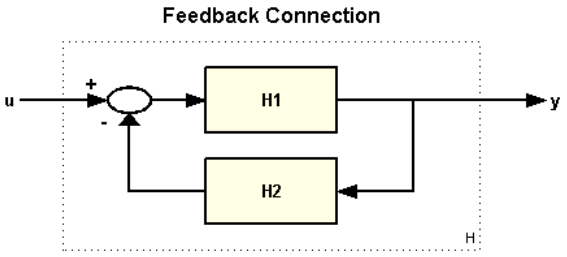
H = H1 + H2

Modèle zéro / pôle / gain à temps continu.ou équivalent

```
H = parallel(H1,H2);
```

Connexions de rétroaction `feedback`

La configuration de retour standard est indiquée ci-dessous:



Pour créer un modèle du transfert en boucle fermée de u à y, tapez

```
H = feedback(H1,H2)
```

la racine :

Syntaxe :

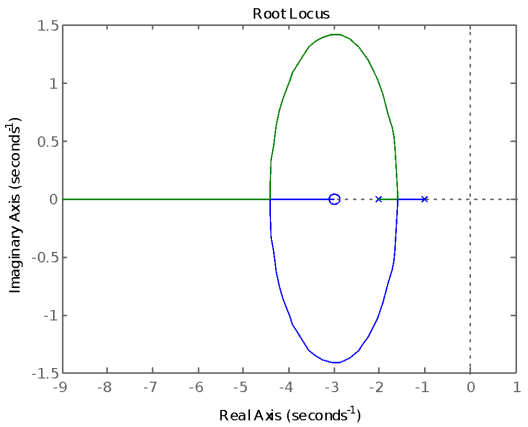
```
rlocus(sys)
rlocus(sys1,sys2,...)
```

La description :

rlocus calcule le lieu racine d'un modèle en boucle ouverte SISO. Le locus racine donne les trajectoires des pôles en boucle fermée en fonction du gain de rétroaction k (en supposant une rétroaction négative). Les locus de racines sont utilisés pour étudier les effets de différents gains de rétroaction sur les emplacements de pôles en boucle fermée. À leur tour, ces emplacements fournissent des informations indirectes sur les réponses temporelles et fréquentielles.

De l'exemple précédent :

```
rlocus(sys)
```



Valeurs propres et vecteurs propres

Syntaxe :

```
e = eig(A)
[V,D] = eig(A)
[V,D,W] = eig(A)
e = eig(A,B)
[V,D] = eig(A,B)
[V,D,W] = eig(A,B)
```

La description :

e = eig (A) renvoie un vecteur de colonne contenant les valeurs propres de la matrice carrée A.
[V, D] = eig (A) renvoie la matrice diagonale D de valeurs propres et la matrice V dont les colonnes correspondent aux vecteurs propres droits correspondants, de sorte que A * V = V * D
[V, D, W] = eig (A) renvoie également la matrice complète W dont les colonnes correspondent aux vecteurs propres gauches correspondants, de sorte que W '* A = D * W'.
Le problème de la valeur propre consiste à déterminer la solution à l'équation Av = λv, où A est une matrice n-par-n, v est un vecteur de colonne de longueur n et λ est un scalaire. Les valeurs de λ qui satisfont l'équation sont les valeurs propres. Les valeurs correspondantes de v qui satisfont l'équation sont les vecteurs propres de droite. Les vecteurs propres de gauche, w, vérifient l'équation w'A = λw '.
e = eig (A, B) renvoie un vecteur colonne contenant les valeurs propres généralisées des matrices carrées A et B.
[V, D] = eig (A, B) renvoie la matrice diagonale D de valeurs propres généralisées et la matrice complète V dont les colonnes correspondent aux vecteurs propres droits correspondants, de sorte que
A * V = B * V * D.
[V, D, W] = eig (A, B) renvoie également la matrice complète W dont les colonnes correspondent aux vecteurs propres gauches correspondants, de sorte que W '* A = D * W' * B.
Le problème des valeurs propres généralisé consiste à déterminer la solution à l'équation Av = λBv, où A et B sont des matrices n-par-n, v est un vecteur de colonne de longueur n et λ est un scalaire. Les valeurs de λ qui satisfont l'équation sont les valeurs propres généralisées. Les valeurs correspondantes de v

sont les vecteurs propres droits généralisés. Les vecteurs propres de gauche, w, vérifient l'équation w'A = λw'B.

De l'exemple précédent :

```
poles=eig(sys)
poles =    -1
           -2
```

Lsim :

Simuler la réponse temporelle d'un système dynamique à des entrées arbitraires

syntaxe

```
lsim(sys,u,t)
```

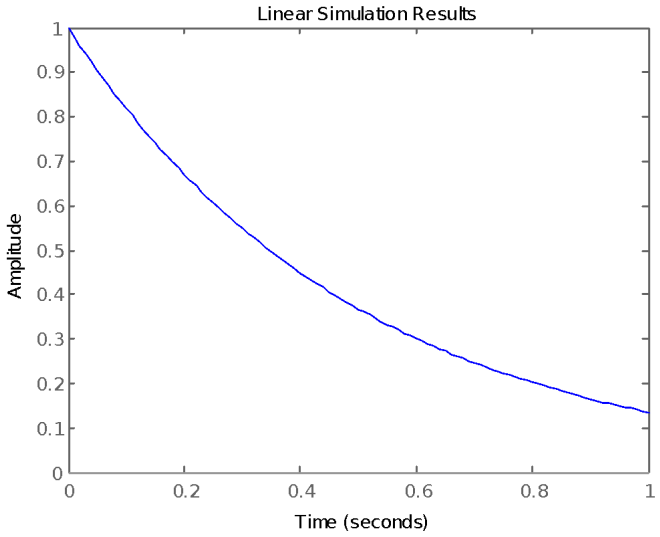
La description

lsim simule la réponse (temporelle) de systèmes linéaires continus ou discrets à des entrées arbitraires. Lorsqu'il est appelé sans arguments de gauche, lsim trace la réponse à l'écran.
lsim (sys, u, t) produit un graphique de la réponse temporelle du modèle de système dynamique sys à l'historique d'entrée, t, u. Le vecteur t spécifie les échantillons de temps pour la simulation (en unités de temps système, spécifiées dans la propriété TimeUnit de sys) et consiste en des échantillons de temps espacés régulièrement:
t = 0: dt: Tfinal
L'entrée u est un tableau comportant autant de lignes que d'échantillons temporels (longueur (t)) et autant de colonnes que d'entrées système. Par exemple, si sys est un système SISO, alors u est un vecteur t-par-1. Si sys a trois entrées, alors u est un tableau t-by-3. Chaque ligne u (i, :) spécifie la ou les valeurs d'entrée à l'instant temporel t (i). Le signal u apparaît également sur la parcelle.

Le modèle sys peut être continu ou discret, SISO ou MIMO. En temps discret, u doit être échantillonné au même rythme que le système. Dans ce cas, l'entrée t est redondante et peut être omise ou définie sur une matrice vide. En temps continu, l'échantillonnage temporel dt = t (2) -t (1) est utilisé pour discrétiser le modèle continu. Si dt est trop volumineux (sous-échantillonnage), lsim envoie un avertissement suggérant que vous utilisiez une heure d'échantillonnage plus appropriée, mais utilisera l'heure d'échantillonnage spécifiée. Voir Algorithmes pour une discussion sur les exemples de temps.

De l'exemple précédent

```
sys=ss(A,B,C,D)
x0=[1 1];
t=[0:0.01:1];
u=0*t;
lsim(sys,u,t,x0);
```



La contrôlabilité :

syntaxe

```
Co = ctrb(A,B)
Co = ctrb(sys)
```

La description

Co = ctrb (A, B) renvoie la matrice de contrôlabilité:

$Co=[BABA^2B\ldots A^{n-1}B]$

où A est une matrice n-par-n, B est une matrice n-par-m et Co a n lignes et nm colonnes.
Co = ctrb (sys) calcule la matrice de contrôlabilité de l'objet LTI de l'espace d'états, sys. Cette syntaxe est équivalente à:
Co = ctrb (sys.A, sys.B);
Le système est contrôlable si Co a le rang complet n.

De l'exemple précédent

```
Co = ctrb(sys)
 n=rank(Co)
Co =
      2      0
      0      2
n =   2
```

le système est contrôlable.

Observabilité :

syntaxe :

```
obsv(A,C)
Ob = obsv(sys)
```

La description :

obsv calcule la matrice d'observabilité pour les systèmes à espace d'états. Pour une matrice n-par-n A et une matrice p-pour-n C, obsv (A, C) renvoie la matrice d'observabilité

$$Ob = \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^{n-1} \end{bmatrix}$$

avec n colonnes et np lignes.
Ob = obsv (sys) calcule la matrice d'observabilité du modèle d'état-espace sys. Cette syntaxe équivaut à exécuter
Ob = obsv (sys.A, sys.C)
Le modèle est observable si Ob a le rang complet n.

De l'exemple précédent

```
Ob = obsv (sys)
n=rank(Ob)
```

Ob =1 0
  0 -2
n =2

placement de Pole :

syntaxe :

```
K = place(A,B,p)
```

La description :

Étant donné le système à une ou plusieurs entrées

$\dot{x}=Ax+Bu$

et un vecteur p des emplacements de pôles à boucle fermée auto-conjugués désirés, la position calcule une matrice de gain K telle que le retour d'état u = –Kx place les pôles à boucles fermées aux emplacements p. En d'autres termes, les valeurs propres de A - BK correspondent aux entrées de p (jusqu'au classement).
K = place (A, B, p) place les pôles de boucle fermée souhaités p en calculant une matrice de gain à rétroaction d'état K. Toutes les entrées de l'installation sont supposées être des entrées de commande. La longueur de p doit correspondre à la taille de la ligne de A. place fonctionne pour les systèmes à entrées multiples et est basé sur l'algorithme de [1]. Cet algorithme utilise les degrés de liberté supplémentaires pour trouver une solution minimisant la sensibilité des pôles en boucle fermée aux perturbations en A ou en B.

De l'exemple précédent :

```
p=eig(sys)
K = place(A,B,p)

p =  -1
     -2
K =
   -0.4441    0.4441
```