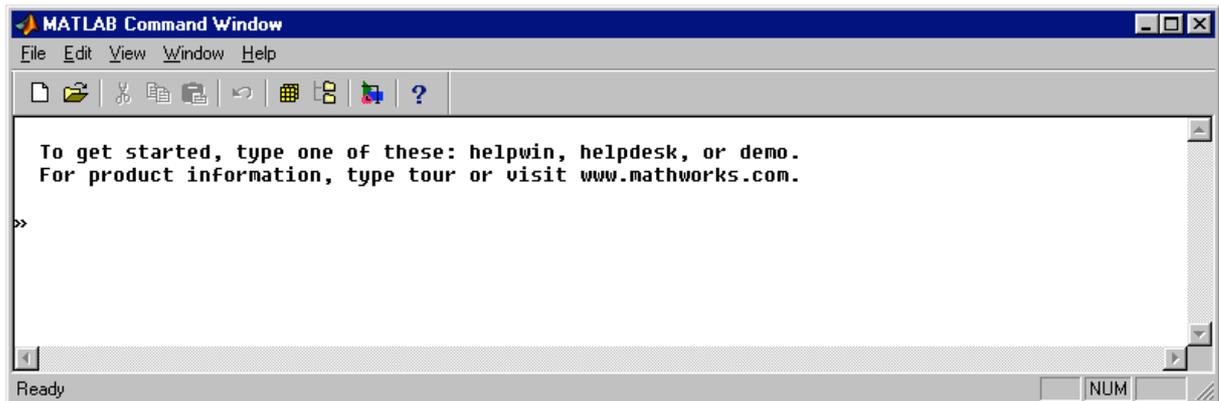


INTRODUCTION A MATLAB

ENVIRONNEMENT MATLAB

Matlab est un environnement de calcul numérique matriciel. Après le lancement de Matlab, une fenêtre de commande apparaît qui permet à l'utilisateur de taper une commande quelconque obéissant à la syntaxe de Matlab. :



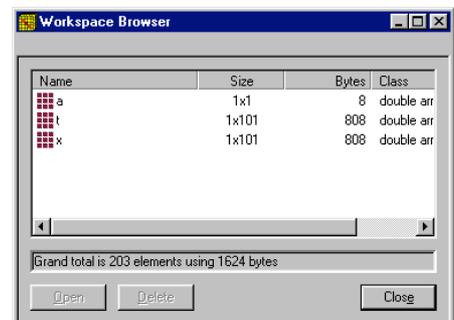
">>" symbole/prompt apparaissant à gauche et indiquant que l'interpréteur est prêt à recevoir une commande.

Variables

Les variables définies par l'utilisateur sont rangées dans l'espace mémoire de Matlab, ces variables sont dites globales.

Le "Workspace browser" permet d'observer les variables existantes. Les commandes who ou whos permettent d'obtenir les mêmes informations.

Pour lancer le Workspace browser, icône :



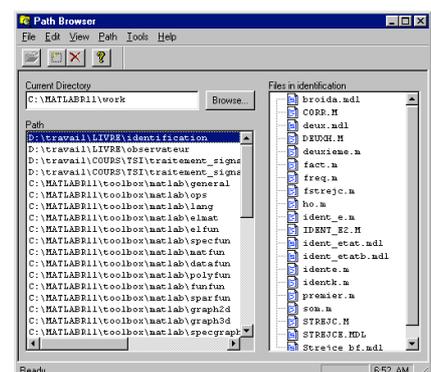
Répertoires de travail

Matlab permet d'ouvrir, de créer, de modifier etc... des fichiers. Matlab sauvegarde tous les fichiers créés dans le répertoire par défaut qu'il est possible de modifier à l'aide de la commande "cd" ou en lançant le "path browser" à l'aide de l'icône suivant :



La fenêtre suivante apparaît, il est alors possible de changer le répertoire courant.

Quand une commande est taper, matlab recherche celle-ci dans l'ensemble des répertoires dont la liste apparaît dans la fenêtre "path", on peut ajouter ou supprimer un répertoire de son choix.

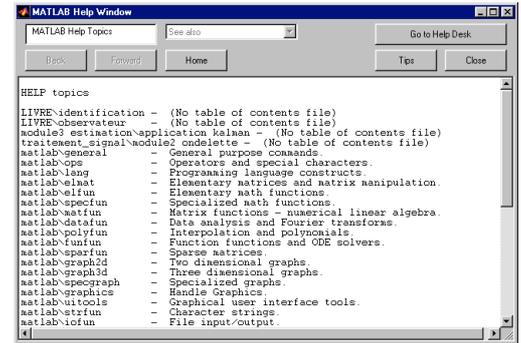


Aide / Help

L'icône  permet d'accéder à l'aide en ligne, la fenêtre d'aide est ouverte

Les commandes suivantes

- >> help
- >> helpwin (la fenêtre ci-contre)
- >> helpdesk (manuel complet avec Acrobat Reader)



Toute commande Matlab possède une entête fournissant des informations sur la commande et sa syntaxe. la commande :

>>lookfor *mot-clé* permet d'avoir la liste des commandes ont l'entête contient *mot-clé*

Autres outils

Editeur : accès par File→New→M_file ou icône 

Debugger : intégrer à l'éditeur

Array_editor : dans le Workspace Browser, double clic sur une variable.

Simulink : environnement graphique de simulation de systèmes dynamique : 

Interpréteur

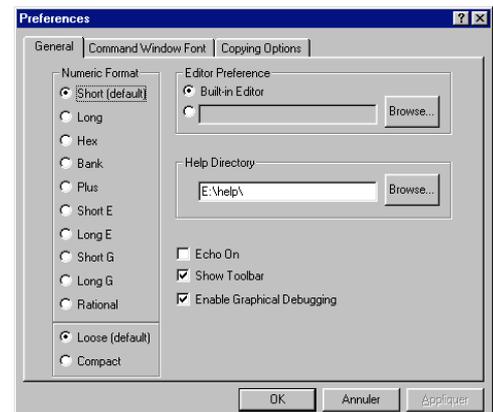
>> *commande*
résultat affichage du résultat
 >> interpréteur disponible

>> *commande* ; **le point virgule provoque l'absence de l'affichage du résultat**
 >> interpréteur disponible

Format d'affichage

Les résultats numériques sont affichés avec le format défini par défaut et redéfinissable par l'utilisateur.

→File→ Preferences



Commentaires

Le symbole % introduit un commentaire, celui-ci n'est pas évalué par l'interpréteur.

DONNEES

Vecteurs

Vecteur ligne

```
>> v=[1 2 3 4 5]; % vecteur 1*5
>> v=[1:5];      % résultat identique à la ligne précédente (incrément de 1 par défaut)
>> v=[1:1:5];   % idem incrément spécifié
```

Vecteur transposé

```
>> v=[1 2 3 4 5]'; % vecteur 5*1
>> v=[1:5]';
>> v=[1:1:5]';
>> w=v';
```

```
>> v1=[borne inf:increment:borne sup];
>> v1(i); %ième élément ATTENTION le premier est à i=1
```

```
>> length(v) %dimension du vecteur
```

Matrices

```
>>A=[1 2 3; 4 5 6;7 8 9; 10 11 12]; %matrice rectangle 4*3
>>B=[1
    2
    3] %matrice 3*1
>> size(A); %dimension de la matrice
>>A(i,j); % élément ligne i colonne j
>>A(:,n); % nième colonne
>>A(p,:); % pième ligne
>>A(i,j,:); % sous matrice des lignes i à j
>>A(i,j,k:l); % sous matrice des lignes i à j colonnes k à l
```

Suppression des données

```
>> clear % supprime toutes les variables
>> clear A % supprime la variable A
```

Sauvegarde des données

```
save nom_fichier A B C % sauve les variables A B et C dans le fichier nom_fichier
load nom_fichier % récupération des données
```

SCRIPT

M_files : est un fichier avec l'extension **.m** qui contient du code Matlab.

Script : contient des commandes Matlab

function : procédure

	Script	function
arguments	Non	Oui
valeur retournée	Non	oui
action sur	var globales	variables locales et globales
utilité	actions répétées	Extension matlab

function

définition	function f = fact(n)
accès par lookfor	%fact factorielle
accès par help	% fact(n) retourne n!
corps	f=prod(1:n);

Plusieurs variables d'entrée / sortie

function [a,b]= nomFonction(c,d,e)

GRAPHISME

Tout tracé avec Matlab, s'effectue dans une fenêtre graphique que l'on crée par la commande `figure` ou quand on exécute une commande de dessin (`plot ...`). On peut créer autant de fenêtres graphiques que l'on veut celles-ci étant numérotées de 1 à N au fur et à mesure de leur création. La fenêtre graphique par défaut et la dernière qui a été créée par `figure` ou la dernière activée par une commande de dessin ou sélectionnée avec la souris.

- **figure** % crée une fenêtre graphique qui devient la figure par défaut,
- **figure(n)** % crée une fenêtre graphique numéro n qui devient la fenêtre active

Fonctions

- *plot*
`t = 0:0.1:5;`
`x = 2*sin(2*pi*t);`
`plot(t,x);` % dessin de x en fonction de t, interpolation linéaire entre les points.
`plot(t,x,'-')` % idem avec style - - -
`plot(t,x,'b-')` % idem style --- couleur bleue
`plot(t,x,'o')` % idem pas d'interpolation, chaque point marqué par o

Un plot provoque l'effacement du dessin précédent (par défaut) on peut superposer des dessins en mettant le commutateur `hold` à on

- **hold on** % désactivation par `hold off`
- **title**('Titre de la figure');
- **xlabel**('commentaire sur l'axe x');
- **ylabel**('idem axe y');
- **axis**([xmin,xmax,ymin,ymax]); % définit l'échelle des axes
- **legend**('tracé 1','tracé 2',...); % chaque tracé est associé à une légende
- **grid** % affiche une grille
- **text**(x,y,'texte') % place texte à la position x y dans la fenêtre
- **gtext**('texte') % place texte à la position définie avec la souris

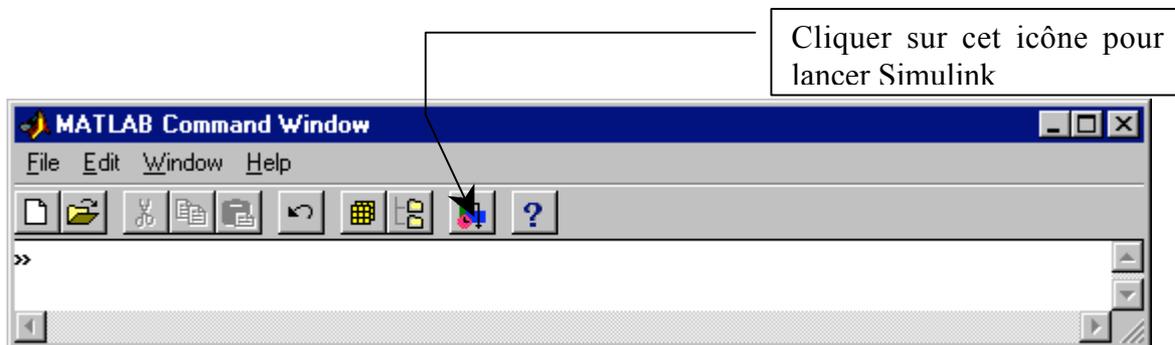
Une fenêtre graphique peut être subdivisée en plusieurs tracés,

subplot(n,p,q)% subdivision en n*q dessin et sélectionne à qième

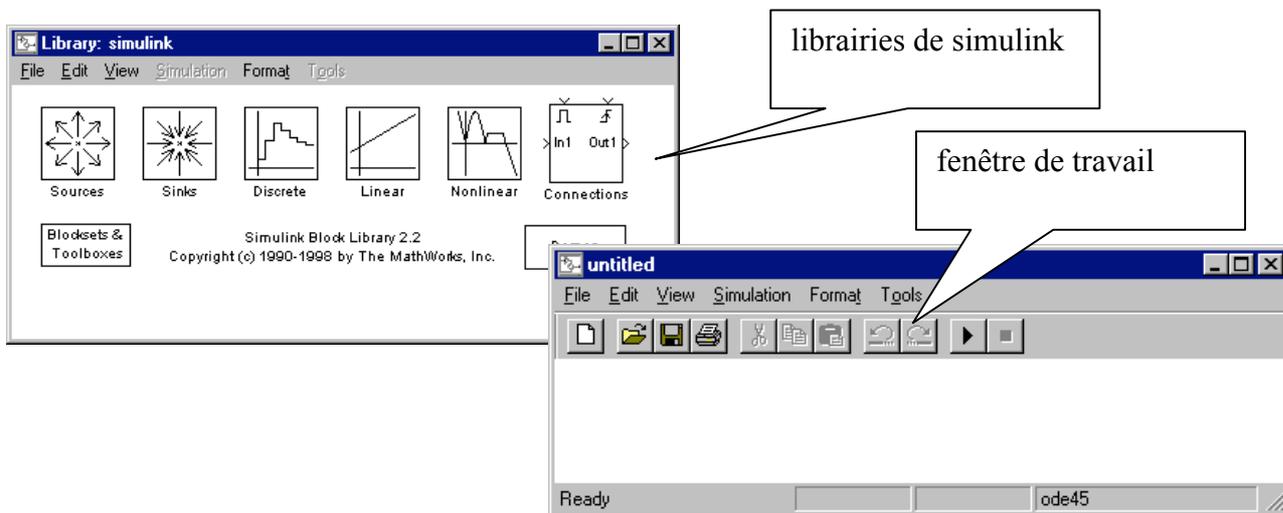
SIMULINK

LANCEMENT DE SIMULINK:

Dans la fenêtre de commande de Matlab:

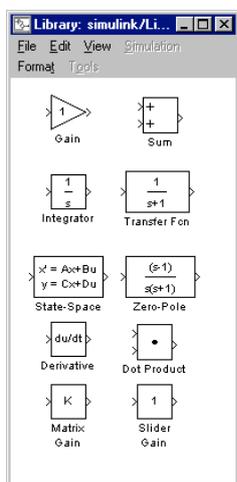


La fenêtre suivante contenant les bibliothèques de simulink, apparaît ainsi qu'une fenêtre de travail.

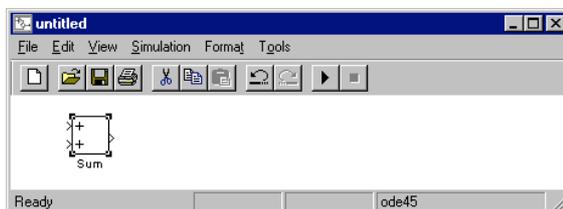


Construction d'un modèle dans la fenêtre de travail

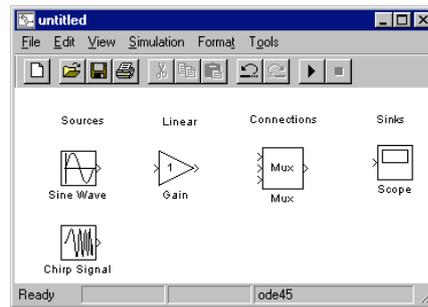
Méthode de placement d'un composant



- on sélectionne une bibliothèque de simulink : double clic pour l'ouvrir (exemple bibliothèque : Linear)
- on sélectionne un composant (exemple Sum):
 - on maintient l'appui sur le bouton gauche de la souris
 - on fait glisser l'élément dans la fenêtre de travail
 - on relâche le bouton.



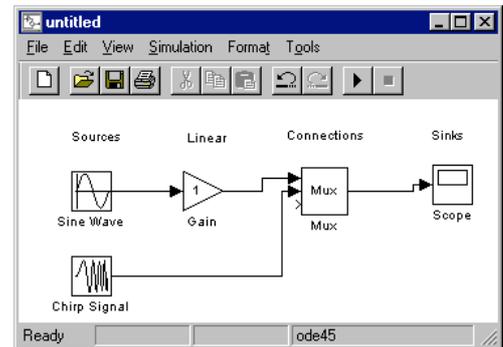
Exercice : construire l'environnement décrit dans la figure suivante, on indique au-dessus de chaque élément, la librairie d'origine.



Réalisation des connexions

Méthode :

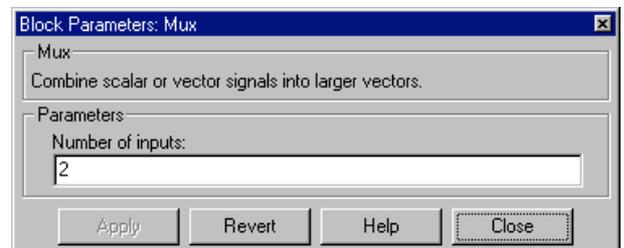
- On sélectionne avec la souris, le symbole > situé sur un composant
- On maintient l'appui sur le bouton et on tire le lien vers un symbole >
- On peut relâcher le bouton pour changer de direction.
- On vérifie que la connexion est correcte par le fait que la flèche est accentuée



Paramétrage des composants

Méthode

- On effectue un double clic sur le composant exemple Mux, la fenêtre de paramétrage s'ouvre,
- On tape les valeurs désirées : ici la valeur 2 pour indiquer 2 entrées,
- On ferme cette fenêtre par Close, les nouvelles valeurs sont prises en compte.



Exercice

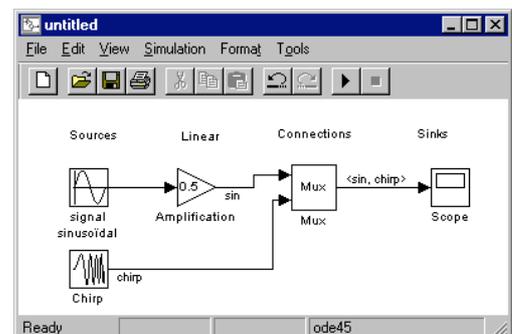
- paramétrer le générateur sinusoïdal avec une amplitude de 2 et une fréquence de 0.5 Hz
- le gain sera réglé à 0.5.
- régler la fréquence de départ du chirp à 0.01 hz

Désignation des composants

Chaque composant possède un nom par défaut exemple Gain, on peut modifier ce nom.

Méthode

- clic sur le nom
- on tape un nouveau nom

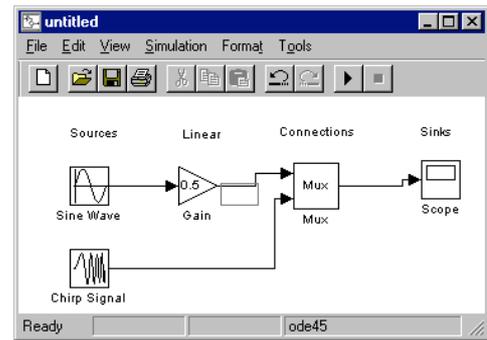
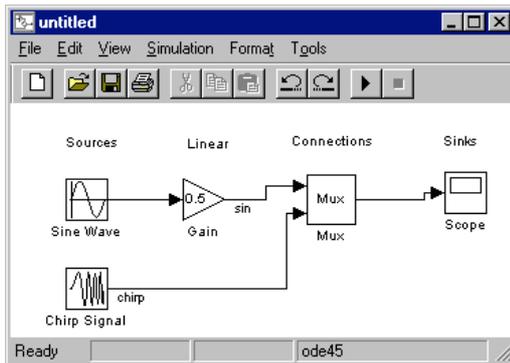


Marquage des liens ou connexions

Méthode

- Double clic sur le lien que l'on veut marquer,
- une fenêtre apparaît qui vous indique le bon déroulement de l'opération.
- on tape un nom exemple sin,

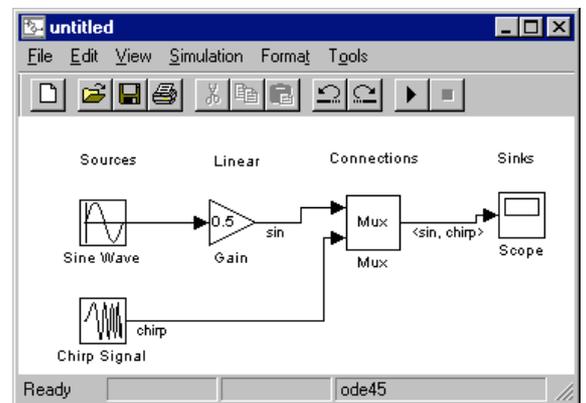
Réitérer l'opération pour le chirp



Propagation des marquages

Méthode

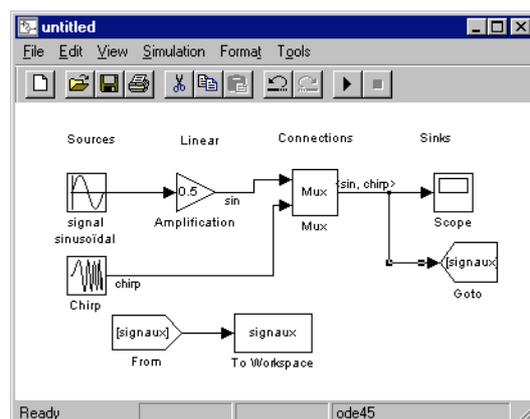
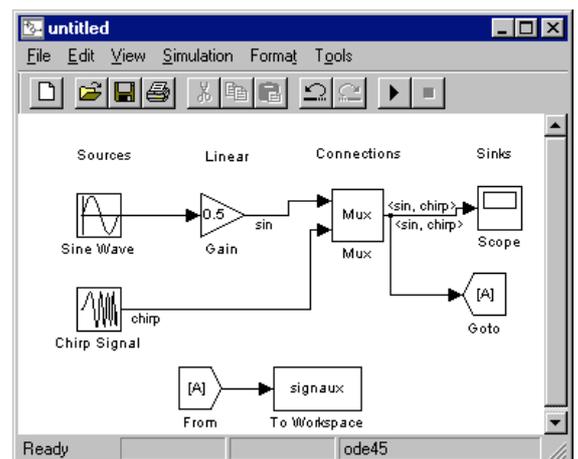
- à la sortie d'un composant : double clic sur le lien,
- taper le symbole < puis cliquer hors de cette fenêtre,
- taper Ctrl D (contrôle D)



Renvoi d'un signal et récupération

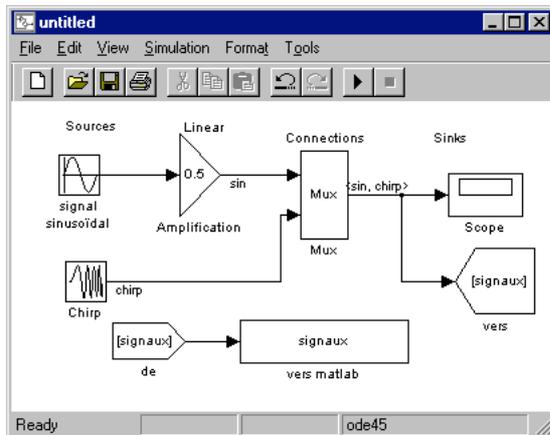
Afin de ne pas surcharger le dessin, on peut utiliser 2 composants situés dans la librairie connections qui permettent d'effectuer une transition sans fil. Ces 2 composants s'appellent GOTO et FROM

Le "tag" possède un nom que l'on peut modifier



Chaque « tag » doit être modifié un par un, attention aux correspondances.

Personnalisation de la fenêtre de travail

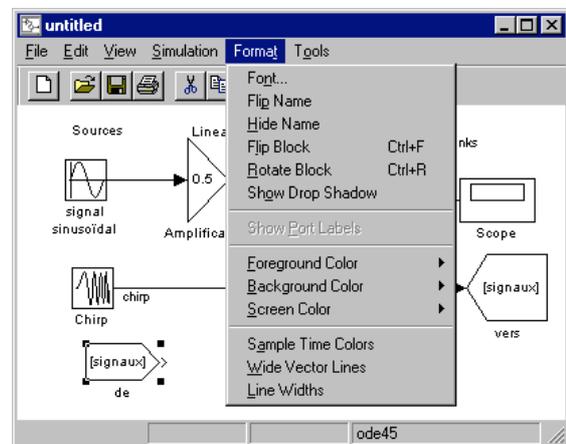


Il est possible de redimensionner chaque composant

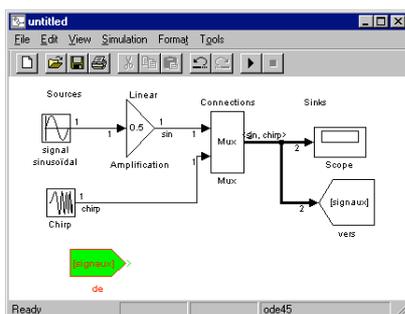
- on le sélectionne,
- on saisit une poignée,
- on étire ou on diminue.

Dans le menu Format de la fenêtre on dispose d'autres commandes (il faut d'abord sélectionner un composant).

- Font permet de choisir le type de caractères
- Flip name : de placer le nom au dessus /en dessous
- Hide name de cacher le nom
- Flip block de retourner le bloc
- Rotate block de le tourner de 90°
- Foreground color de sélectionner une couleur pour le texte
- Background color : de sélectionner une couleur pour le bloc



On peut de même personnaliser les liens ou connexions :



- Wide Vector Lines permet de dimensionner l'épaisseur des liens en fonction du nombre de signaux,
- Line Width permet d'obtenir l'indication du nombre de signaux sur les liens
- Ctrl D permet de mettre à jour tout ceci en cas de modification.

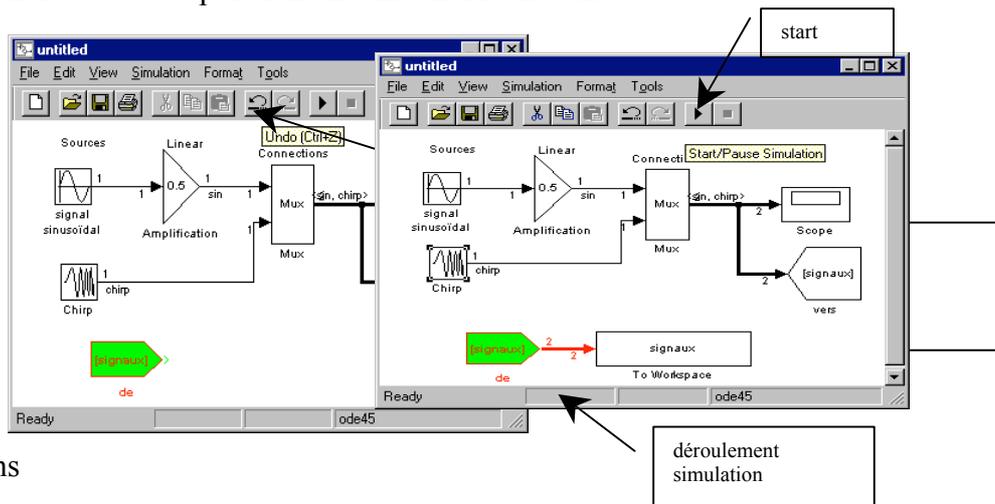
Modifications

Modification des composants

On peut :

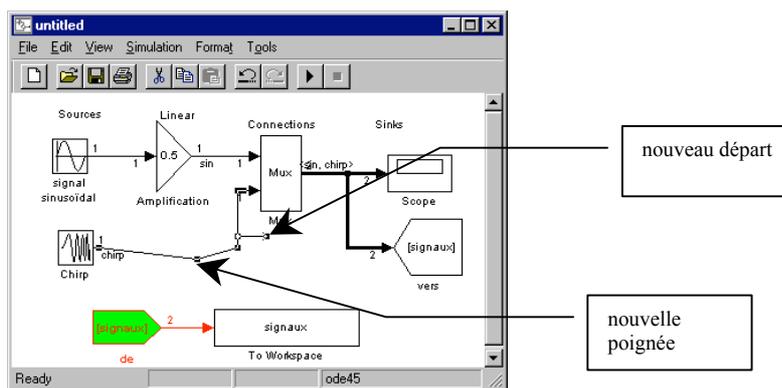
- ajouter un composant à tout moment,
- supprimer un composant en le sélectionnant et touche Suppr,
- Modifier la position d'un composant en le sélectionnant on laisse la touche gauche de la souris appuyée et on le déplace.
- dupliquer un composant : on le sélectionne, on appuie sur la touche Ctrl on faisant glisser le composant.

On peut revenir en arrière de toute opération en utilisant l'icône Undo



Modification des liens

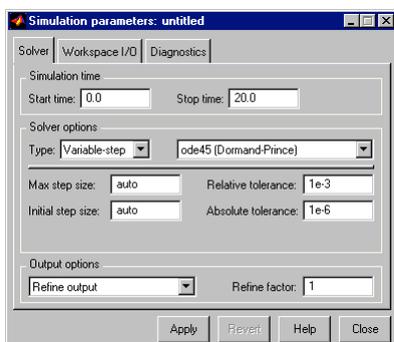
- en utilisant les poignées situées sur le lien (une fois sélectionner celles-ci apparaissent),
- en appuyant sur le bouton droit de la souris on ajoute un nouveau départ,
- shift et bouton gauche permet d'ajouter de nouvelles poignées de changement de direction.



SIMULATION

Paramétrage de la simulation

La simulation utilise un certain nombre de paramètres : menu simulation → parameters

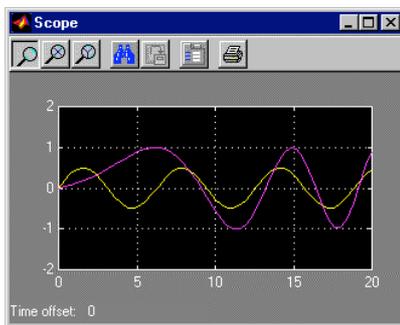


- instant de départ (0 par défaut)
 - instant d'arrêt (mettre 20s)
- On étudiera ultérieurement les autres paramètres.
Faire close ce qui valide les modifications.

Lancement de la simulation

- menu simulation → start
 - ou ctrl T
 - ou icône >
- Une sonnerie indique la fin de la simulation

Exercice lancer la simulation après avoir ouvert le scope



- On peut effectuer des zooms avant/arrière avec les 3 premiers icônes
- ajuster les axes avec le 4°
- sauver les données avec le 5°
- régler le scope avec le 6°
- imprimer avec le dernier

LIEN ENTRE SIMULINK ET MATLAB

Pour diverses opérations, il est intéressant de disposer des signaux dans l'environnement de matlab ou de récupérer des signaux définis dans matlab.

Envoi de signaux vers l'environnement de matlab

Les blocs ToWorkspace de la librairie Sinks permettent de diriger les signaux vers l'environnement de matlab dans l'exemple traité jusqu'à présent ceci est réalisé avec le bloc nommé "signaux" sur lequel arrive le tag "de"

Exercice :

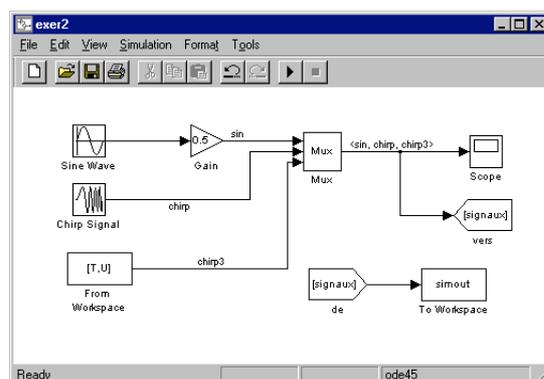
Dans la fenêtre de commande matlab, taper `plot(tout,signaux)`
 signaux est une variable contenant les signaux générés
 tout est une variable que l'on verra plus tard

Récupération de signaux issus de matlab

Le bloc FromWorkspace de la librairie Source permet de définir des signaux dans l'environnement Matlab et de les utiliser dans l'environnement de Simulink.

Exemple

On définit la variable $T=0:0.01:20$; ainsi que le signal désiré $U=\sin(2*\pi*t^3)$;
 Attention les vecteurs T et U doivent être des vecteurs colonnes.



SIMULATION D'UN SYSTEME NON LINEAIRE

Construction d'un modèle non linéaire

Exemple :

$$\dot{x}_1 = x_2 + u \cos(x_1)$$

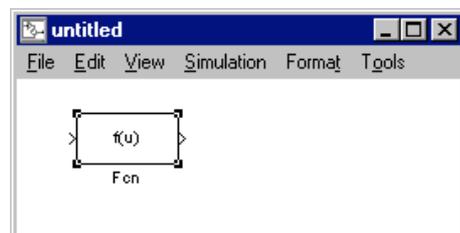
$$\dot{x}_2 = u$$

$$y = x_1$$

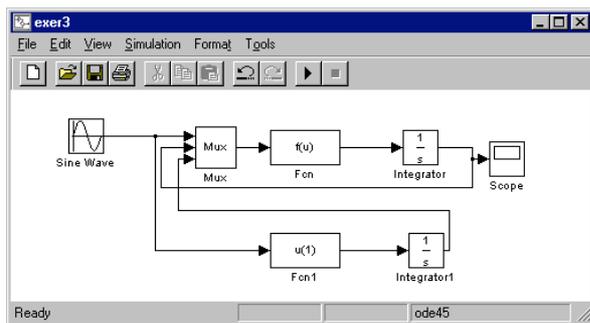
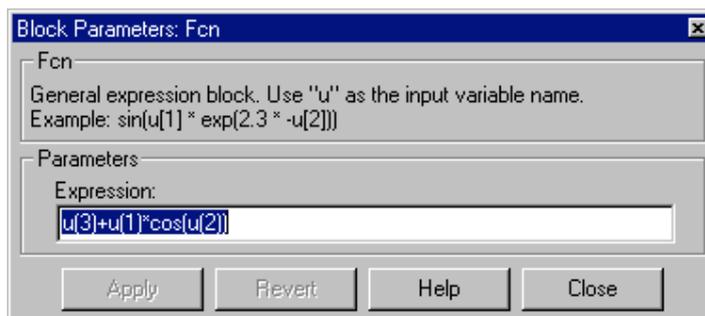
On commence par créer le système à l'aide du bloc Fcn de la librairie NonLinear

L'entrée de ce bloc est nommée u et peut-être un vecteur
Les composantes sont désignées par u(1) u(2) etc...

- la fonction correspondant à \dot{x}_1 s'écrit : $u(3)+u(1)*\cos(u(2))$
- si les entrées sont respectivement u,x1 et x2
- ceci est programmé dans la fenêtre de définition de Fcn



La construction du système ci-dessus, utilise des intégrateurs (librairie linear) et des multiplexeurs (librairie Connexions) comme sur la figure suivante.



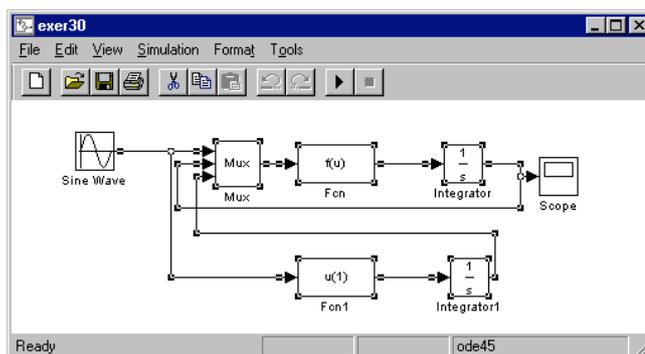
On peut alors effectuer une simulation du système, on voit que celui-ci diverge, le contrôle de ce système sera vu ultérieurement dans le chapitre commande non linéaire.

Création d'un sous système

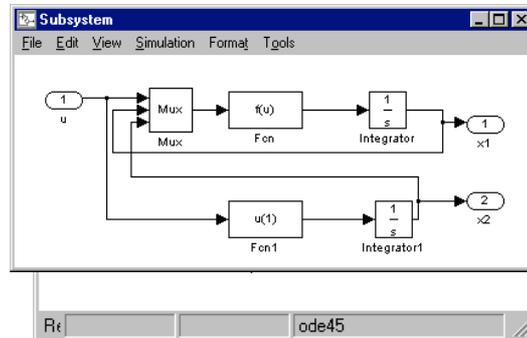
On va encapsuler le système précédent dans un sous système, cette méthode permet de rendre plus lisible un schéma et de paramétrer le sous-système.

Méthode

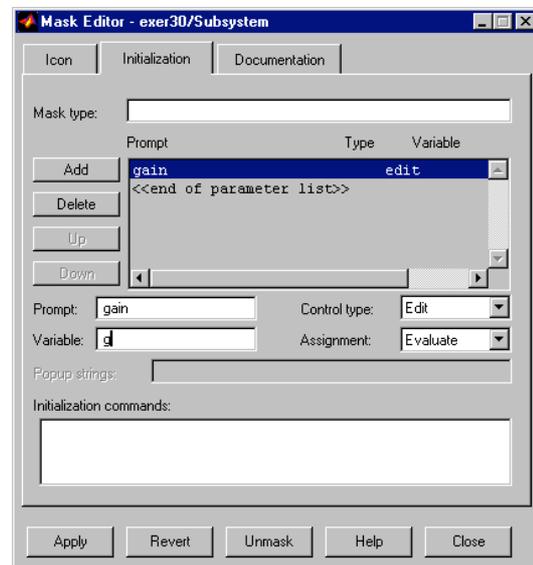
- on sélectionne tous les éléments qui formeront le sous-système
- On tape Ctrl G (ou Edit → Create SubSystem)



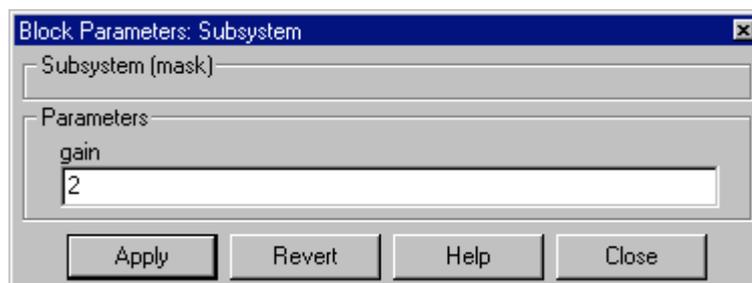
- On obtient : un bloc nommé Subsystem que l'on peut renommer, les entrées et sorties peuvent être elle aussi renommées à l'intérieur du bloc (double clic pour ouvrir le bloc).
- Des ports d'entrée et de sortie on été rajoutés automatiquement, on peut en ajouter par exemple le port correspondant à x2



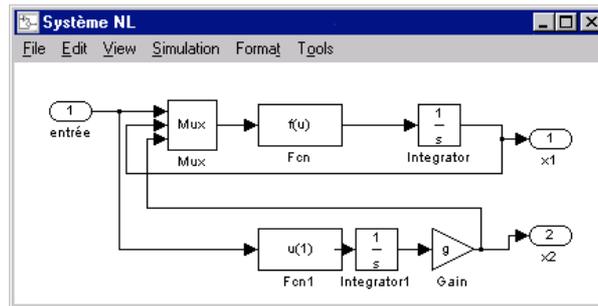
- On va introduire un gain suivant x2 d'une valeur désignée par g → edit → Mask Subsystem → initialization



- En cliquant maintenant sur le sous-système, on obtient la fenêtre suivante, dans laquelle on fixe une valeur de g



- On prend en compte cette valeur pour la dérivée de x2 par exemple par → edit → Look under Mask



MODELISATION D'UN SYSTEME LINEAIRE MONOVARIALE ET SIMULATION

But : Construire un modèle continu en représentation externe et le simuler.

Définition du système : $H(p) = \frac{1}{p^2 + 0.2p + 1}$

Format :

- Une fonction de transfert monovariante peut être représentée
 - Soit par le rapport de 2 polynômes en p (ou s). Dans Matlab, un polynôme se représente par un vecteur de paramètres suivant les puissances décroissantes de p par exemple : le dénominateur de l'exemple ci-dessus se représente par le vecteur : [1 0.2 1]
 - Soit par une représentation par zéros et pôles
- Matlab utilise un modèle LTI pour la définition de systèmes

création d'un modèle LTI {Linear Time Invariant}

- Définir le numérateur exemple : `>num = 1 ;`
- Définir le dénominateur exemple `>den = [1 0.2 1] ;`
- Construire le système LTI exemple `> sys1= tf(num,den) ;`
- Faire afficher sys1 en tapant le nom puis sur la touche Return

Ou

- Vecteur définissant les zéros exemple : `> zz=[] %aucun`
- Vecteur définissant les pôles exemple : `> pp=[-0.1000 + 0.9950i -0.1000-0.9950i]`
- Gain exemple `> k=1`
- Construire le système : `> sys2=zpk(zz,pp,k)`

Les propriétés d'un modèle LTI sont obtenues en tapant :

`> get(sys1)`

On obtient :

```

num = {[0 0 1]}
den = {[1 0.2 1]}
Variable = 's' ..... choix possible {continu : {s p} échantillonné : {z z-1 q}}
Ts = 0 ..... 0 / système continu sinon la période d'échantillonnage
Td = 0 ..... 0 ou la valeur d'un retard sur l'entrée
InputName = {} ..... on peut donner un nom à l'entrée
OutputName = {} ..... Idem pour la sortie
Notes = {}
UserData = []

```

modification des champs à l'aide de la fonction set

```
>set(sys,'Variable','p')
```

Transfer function:

```

1
-----
p^2 + 0.2 p + 1

```

analyse du système

- Diagramme de bode > bode(sys1)
- Diagramme de Nyquist > nyquist(sys1)
- Diagramme de Nichols > nichols(sys1)
- Tracé des pôles et des zéros > pzmap(sys1)
- Lieu des pôles > rlocus(sys1)

Note : après chacune de ces commandes, la commande grid permet d'obtenir un maillage adapté au type de diagramme.

- Calcul des zéros, des pôles et du gain > [Z,P,K]=zpkdata(sys1,'v')
 - Pôles seuls > pole(sys1)
 - ou eig(sys1)
- Calcul des marges de phase et de gain > margin(sys1)

Simulation

- Réponse impulsionnelle > impulse(sys1)
- Réponse à un échelon > step(sys1)
- Réponse à une entrée quelconque :
 - Il faut définir la variable temporelle t=0 :0.01 :10 ;
 - Il faut définir l'entrée u=sin(t)
 - On simule > lsim(sys1,u,t)

Interface de travail avec les modèle LTI

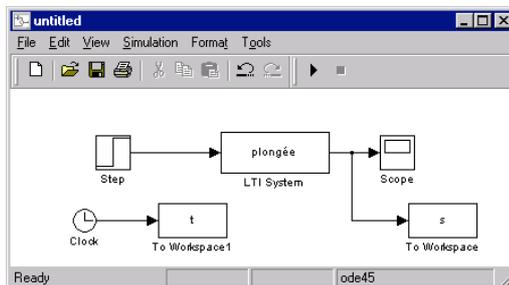
Matlab offre un environnement de travail que l'on obtient en tapant >ltiview

Utilisation de simulink

Le modèle LTI peut-être utilisé dans l'environnement de simulink /exer5/aexer5

- Ouvrir successivement : Blocksets&toolboxes → LTI

- Faire glisser dans la fenêtre de travail le bloc lti
- Taper le nom sys1 après un double clic sur le bloc.
- Simuler le système.



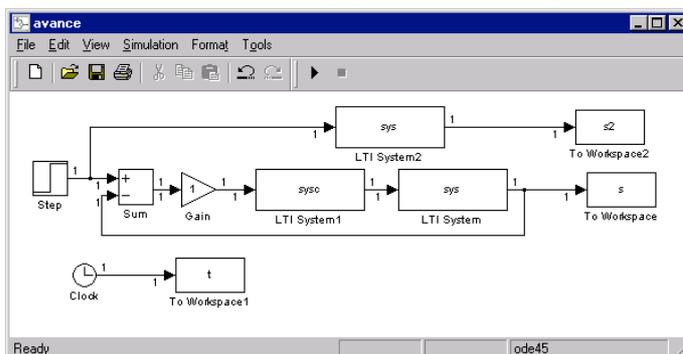
Tout ce qui précède permet de travailler pratiquement indépendamment de simulink, dans la suite, nous allons nous servir uniquement de simulink qui permet d'interconnecter facilement des systèmes.

Interconnexion de systèmes

But : définir un correcteur à avance de phase et corriger le système du stage précédent {aexer6/exer6}

Rappel : calcul d'un correcteur à avance de phase (pendant la séance)

- calculer le correcteur
- construire l'environnement suivant qui permet de comparer sans et avec correcteur /exer6/

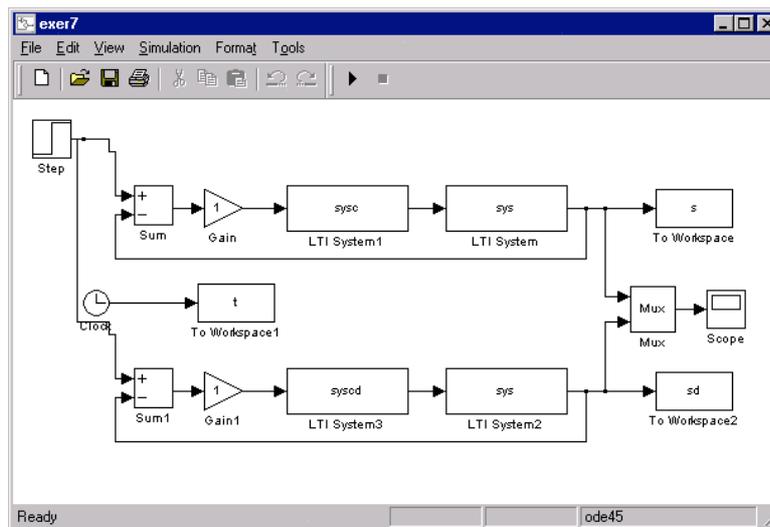


- Calculer le gain statique du système

Discrétisation du correcteur

Rappel : cf. cours /exer7/aexer7

- Extraire le numérateur et le dénominateur de sysc le correcteur continu
- Utiliser la fonction c2d avec la méthode de Tustin pour transformer le correcteur continu en correcteur discret avec une période d'échantillonnage de 0.1s,
- Simuler ce correcteur discret et comparer le au résultats obtenus avec le correcteur continu.



Compléments

Le correcteur à avance de phase calculé précédemment, possède un gain statique que l'on peut déterminer de la manière suivante : `/aexer8/exer8`

Calcul du système en boucle ouverte > `sysbo = sysc*sys`

Calcul du système en boucle fermée > `sysbf = feedback(sysbo,1)`

Attention le dénominateur n'est pas normalisé

Gain statique : `Gs = sysbf(0)`

Si on désire ramener le gain statique à 1 il faut ajouter un amplificateur de $1/G_s$. Le correcteur devient `sysc*1/Gs`

```
[nu,de]=tfdata(sysbf,'v');
gs=length(de);
nu=nu./gs;
de=de./gs;
sysbf=2*tf(nu,de);
[mg,mp,wg,wp]=margin(sysbf);
%erreur
mpd-mp
```

On vérifie dans ce cas que la marge de phase diminue, pour modifier la marge de phase, on recalcule le correcteur en prenant en compte cette écart.

MODELISATION D'UN SYSTEME LINEAIRE MULTIVARIABLE ET SIMULATION

Rappel sur la représentation d'état (cf. stage)

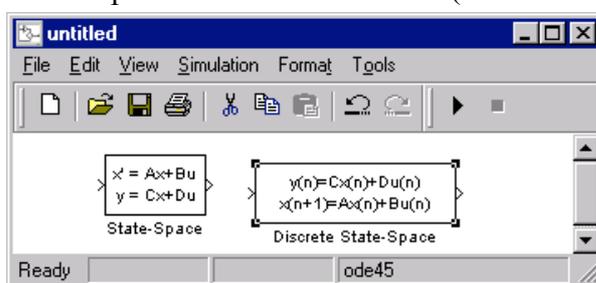
Un système dynamique linéaire multivariable se met sous la forme suivante

$$\begin{aligned} \dot{X}(t) &= AX(t) + Bu(t) \\ s(t) &= Cu(t) + Du(t) \end{aligned} \quad \text{ou} \quad \begin{aligned} X_{k+1} &= FX_k + Gu_k \\ s_k &= CX_k + Du_k \end{aligned}$$

- Où
- X(t) (Xk) est le vecteur d'état de dimension n
 - A (F) la matrice d'évolution ou d'état de dimension n n
 - U(t) (uk) la commande de dimension r
 - B (G) la matrice d'entrée de dimension n r
 - S(t) (sk) la sortie de dimension m
 - C la matrice d'observation de dimension m n
 - D la matrice de couplage entrée/sortie de dimension m r

Création d'un modèle LTI en représentation d'état : ss
 Les propriétés s'obtiennent et se modifient par get et set

Sous simulink, utiliser le bloc StateSpace de la librairie Linear (ou Discret)



Exemple : pendule inversé

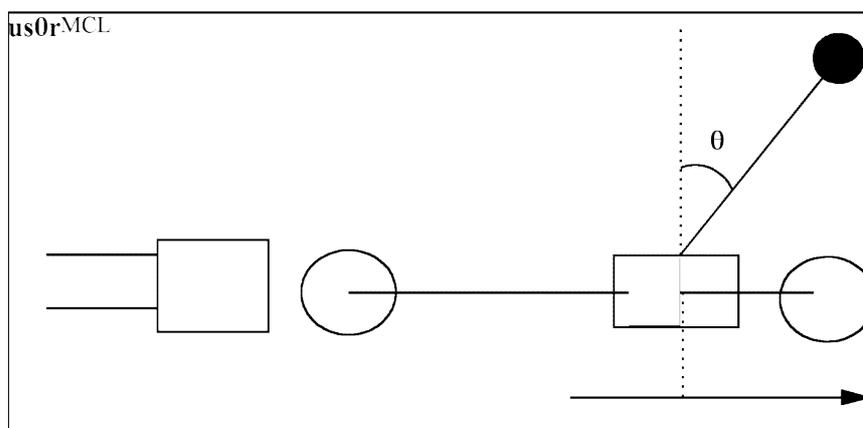


Figure 1: Chariot et pendule.

Un chariot [C] se déplace sur un rail (position r, vitesse r-dot), le dispositif pendulaire est constitué d'une tige [L] à l'extrémité de laquelle, est montée une masse [M]. La liaison entre le chariot et le pendule est une rotoïde non actionnée.

Les équations différentielles régissant le mouvement du chariot et celui du pendule sont données par les équations suivantes :

$$J\ddot{\theta} + K\dot{\theta} - M_1 l_s g \sin(\theta) + M_1 l_s \ddot{r} \cos(\theta) = 0$$

$$M \ddot{r} + f_R \dot{r} + M_1 l_s [\ddot{\theta} \cos(\theta) - \dot{\theta}^2 \sin(\theta)] = f$$

avec :

- r : la position du chariot (m)
- f : la force horizontale appliquée au chariot (en N)

- θ : la position angulaire du pendule (en rd)
- M_1 : la masse du pendule (Kg)
- M_0 : la masse du chariot
- M : la masse totale du chariot et du pendule
- l_s : distance au chariot, du centre de gravité du pendule (m)
- J : Inertie (kg m^2)
- K, f_R : coefficients de frottement visqueux du pendule et du chariot
- g : gravité.

Mise en équation voir le document en annexe

Représentation d'état du système linéarisé et normalisé:

$$\dot{X} = \begin{bmatrix} 0 & 0 & 1.9503 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0.1289 & 1.9148 & 0.0008 \\ 0 & 21.4964 & 26.3388 & 0.1362 \end{bmatrix} X + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -6.1347 & 0 \\ 84.3862 & 0 \end{bmatrix} u$$

$$Y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} X + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} u$$

Simulation du système stabilisé (cor_lin1)

- 1 Calcul de l'observateur
- 2 Calcul du retour d'état
- 3 Calcul du gain statique
- 4 Construction de l'environnement
- 5 Simulation

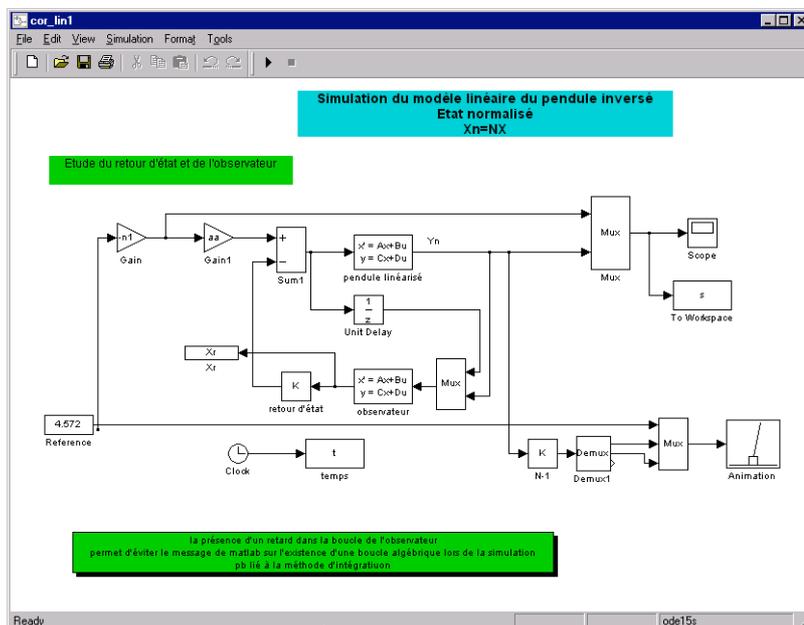
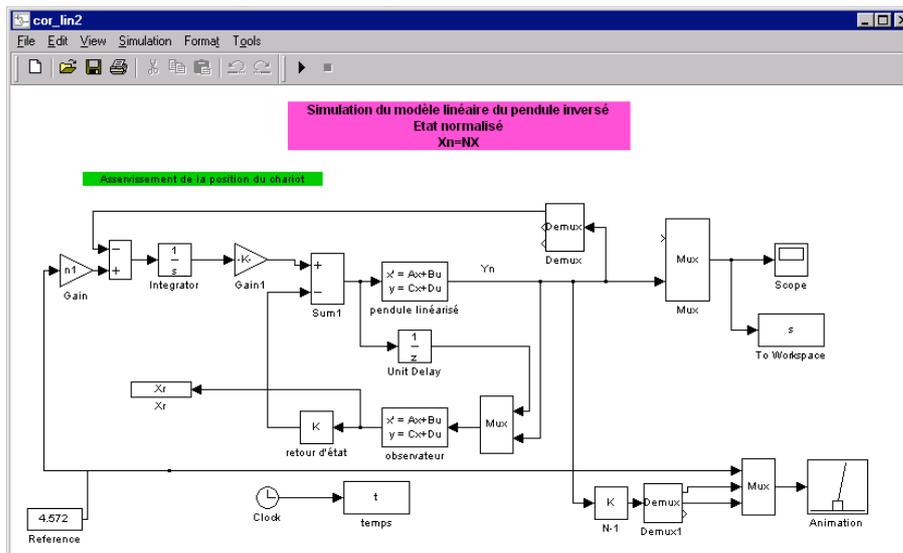


Figure 2 Schéma du système stabilisé

- Simuler le fonctionnement du système
- Le système est-il stabilisé ?
- Vérifier le bon fonctionnement de l'observateur
- Exercer une forte variation de position au chariot, le modèle est-il réaliste ?

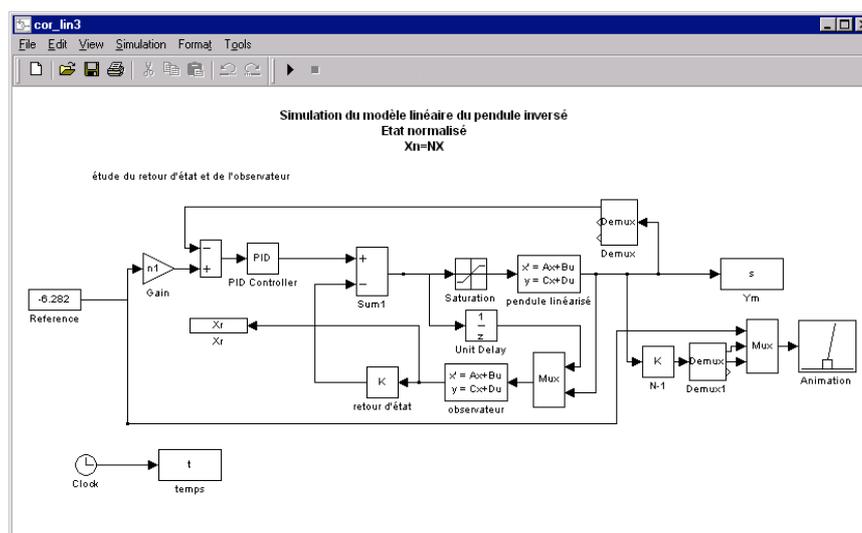
Etape 2 : Asservissement de la position du chariot (cor_lin2)

Ajout d'une correction intégrale sur la position du chariot



Etape 3 : Limitation de la commande (cor_lin3)

Ajout d'une saturation sur l'entrée du pendule



Etape 4 : Simulation avec le modèle non linéaire (cor_nl)

On remplace le modèle linéarisé par le modèle non linéaire du pendule