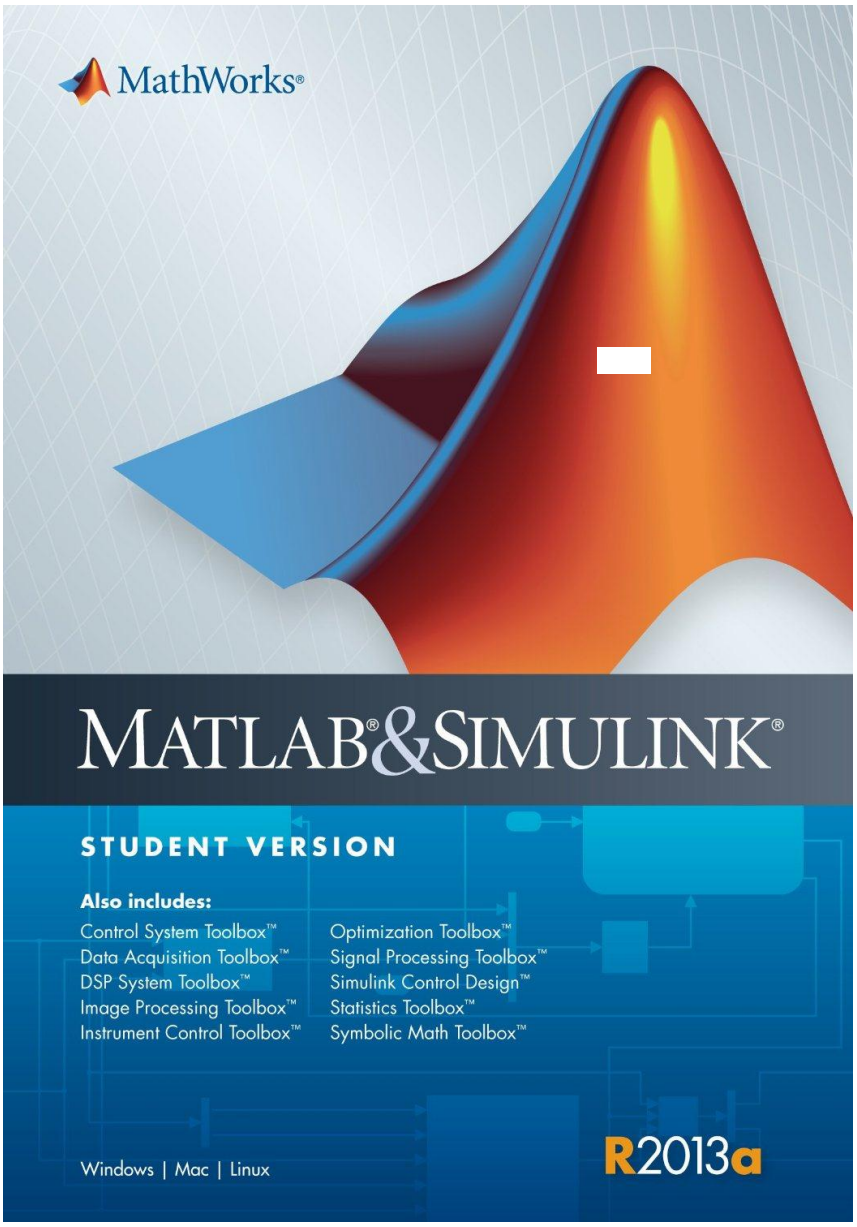
TP1 - Régulation et Asservissement

Initiation à MATLAB – SIMULINK
&
Initiation aux systèmes asservis



Objectifs

- Présenter les différentes fonctionnalités de MATLAB-SIMULINK ;
- Donner une démarche de conception d'un modèle de simulation ;
- Simuler le fonctionnement d'un moteur à courant continu ;
- Rappeler les principales notions d'automatique : précision statique, stabilité, synthèse de correcteurs simples ;
- Intégrer des résultats de simulation dans un rapport.

Il s'agit ici d'une initiation de base à l'environnement de simulation, de nombreuses autres fonctionnalités pourront être découvertes par la suite.

Plan du document :

1. Notions de base sur MATLAB

1.1. Présentation de l'environnement

MATLAB est un environnement de calcul intégré qui allie le calcul numérique, les graphiques avancés, la visualisation et un langage évolué de programmation. MATLAB est utilisé aujourd'hui dans un grand nombre de domaines d'applications comme le Traitement du Signal, le Traitement de l'Image, la conception de systèmes de contrôle, l'ingénierie financière et la recherche médicale.

La famille des produits MATLAB inclut des outils pour :

- L'analyse de données et la visualisation ;
- Le calcul numérique et symbolique ;
- Les graphiques scientifiques ;
- La modélisation et la simulation ;
- La programmation, le développement d'applications et la conception d'interfaces graphiques ;
- La conversion automatique de programmes MATLAB en code C et C++ indépendant.

Les Boîtes à Outils (Toolboxes) sont des bibliothèques spécialisées de fonctions MATLAB, spécialement conçues pour apporter des solutions dans des domaines applicatifs. Elles sont écrites en langage MATLAB, leurs sources sont disponibles et l'on peut s'en inspirer pour développer ses propres algorithmes.

Chacune de ces Toolboxes, ayant été mise au point par des experts de la spécialité, représente l'état de l'art du domaine correspondant couvrent des domaines variés : le traitement du signal, l'automatique, les mathématiques appliquées, l'analyse de données, la finance....

1.2. Lancement de MATLAB-SIMULINK

- Se connecter sur un compte local
- Avec l'explorateur, créer un répertoire : exemple *d:\CMS\groupexx\no***m**
- Cliquer sur l'icône MATLAB 6.5
- Sous MATLAB se placer dans le répertoire créé :
  Commande : *cd_d:\ CMS\groupexx\nom* ou sélectionner directement le répertoire créé dans l'arborescence de MATLAB.

Attention : ne pas mettre de blanc dans les noms des répertoires. Eviter les noms trop longs, c'est lourd à utiliser par la suite !!

Vérifier le répertoire en tapant la commande : 'cd'
MATLAB renvoie alors le nom du répertoire courant.

1.3. Variables, espace de travail

MATLAB permet de réaliser des calculs à partir de variables qui peuvent être scalaires ou vectorielles. Le nom des variables doit respecter les règles suivantes :

- 31 caractères maximum
- le 1° caractère est une lettre obligatoirement
- MATLAB différencie les majuscules et les minuscules

Il est inutile de typer ou de dimensionner les variables.

Les variables sont sauvegardées dans l'espace de travail ('Workspace'), elles sont conservées tant qu'elles ne sont pas effacées explicitement par les commandes :

- '*clear*' : efface toutes les variables
- '*clear A B*' efface les variables *A, B.*

Les commandes '*whos*' et '*who*' permettent de lister les variables crées.

1.4. Manipulation de données vectorielles

Construction de tableaux :

La gestion des données vectorielles est très performante sous MATLAB. Quelques petits exercices peuvent aider à comprendre les mécanismes.

Pour rentrer le vecteur colonne $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$, il faut écrire : A = [1;2]

Règle :

';' : changement de ligne ' ' touche « espace » : changement d'élément dans une colonne

On peut concaténer les éléments les uns avec les autres

B = [A [-1;3]] a pour résultat : $B = \begin{bmatrix} 1 & -1 \\ 2 & 3 \end{bmatrix}$

Exercice :

Ajouter une ligne de 0 en 3° position :

C = [B ; [0 0]] a pour résultat : $C = \begin{bmatrix} 1 & -1 \\ 2 & 3 \\ 0 & 0 \end{bmatrix}$

et concaténer un vecteur colonne:

U = [1 ;5 ;4]

D = [C U] a pour résultat : $D = \begin{bmatrix} 1 & -1 & 1 \\ 2 & 3 & 5 \\ 0 & 0 & 4 \end{bmatrix}$

Ajouter une colonne de 0 en 4° position puis concaténer le vecteur $V = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$

Résultat : $E = \begin{bmatrix} 1 & -1 & 1 & 0 \\ 2 & 3 & 5 & 0 \\ 0 & 0 & 4 & 0 \\ 1 & 2 & 3 & 4 \end{bmatrix}$

On peut avoir directement accès aux éléments de la n° colonne en tapant :
E(:,c)
On peut avoir directement accès aux éléments de la m° ligne en tapant :
E(l,:)
Il est possible d'accéder à un élément particulier d'un tableau en tapant :
E(l,c)

Cas particulier des vecteurs régulièrement échantillonnés :

Syntaxe : a = début : pas : fin
Exemple : t = 0 : 0.1 : 1
't ' est un vecteur de 11 éléments régulièrement espacés de 0.1
t(1) = 0, t(11) = 1

1.5. Fonctions graphiques

La commande graphique de base est la suivante :

plot(x1,y1,'s1',x2,y2,'s2', …)

- x : tableau des abcisses
- y : tableau des ordonnées
- s : chaîne de caractères définissant :
  - la couleur,
  - le style de tracé
  - le type de marqueur

La syntaxe est la même avec les commandes semilogx, semilogy, loglog

Symbole	Couleur
y	Yellow
m	Magenta
c	Cyan
r	Red
g	Green
b	Blue
w	White
k	Black

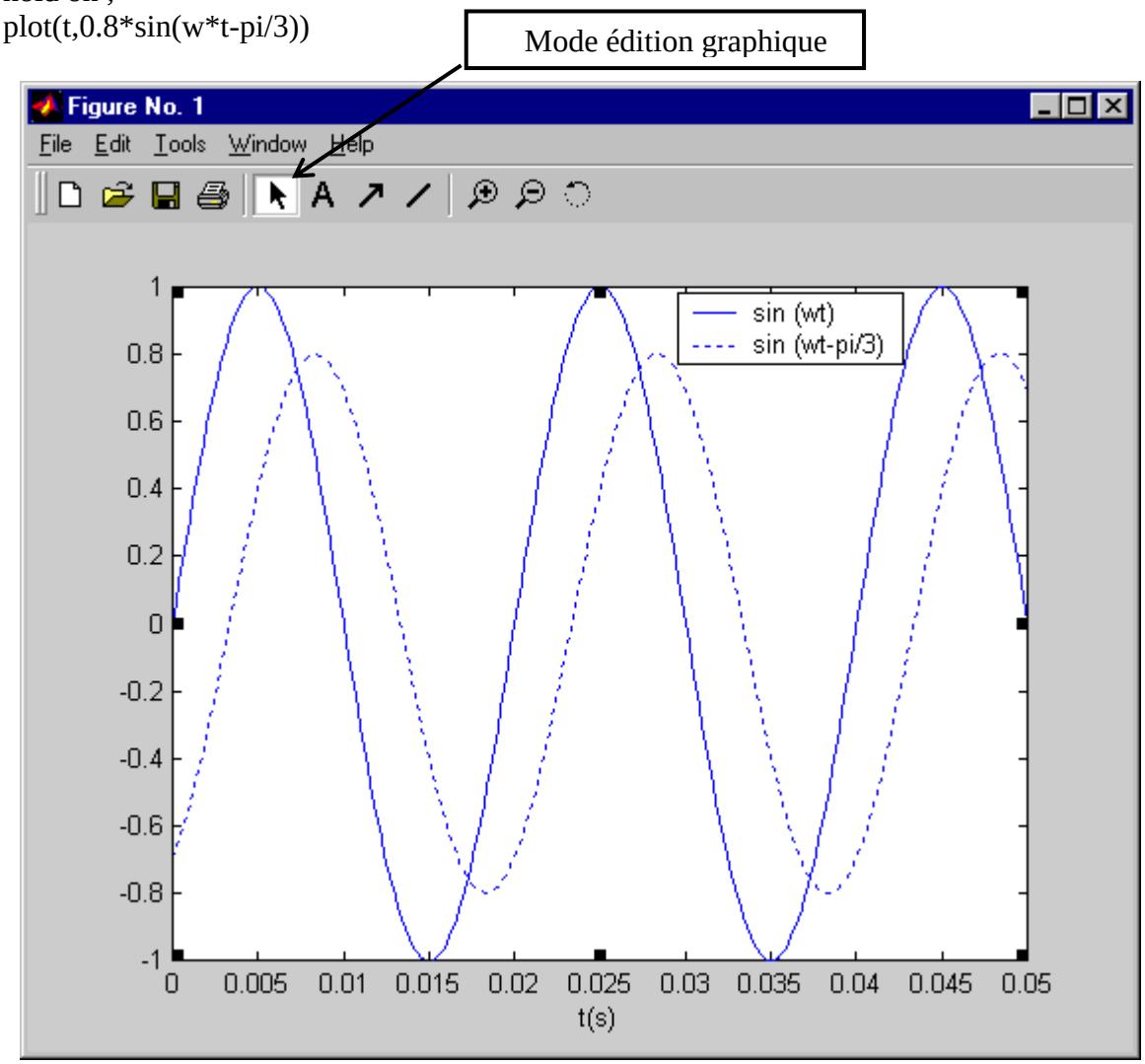
Symbole	Marqueur
+	Plus
o	Cercle
*	Astérisque

Les principales fonctions graphiques sont les suivantes

Figure	Création d'une nouvelle figure
Subplot	Création de plusieurs axes
Grid	Affichage d'une grille
Axis	Définition des limites des axes
hold	Superposition de tracés
Close	Fermeture de figure

Exemple :

```
t = 0 :200e-6 :50e-3 ;        % création d'un tableau de point
w = 314 ;
plot(t,sin(w*t)) ;            % affichage de la fonction sin(wt)
hold on ;
plot(t,0.8*sin(w*t-pi/3))
```



Il est possible de modifier les attributs d'un graphique en passant en mode édition graphique en cliquant sur l'icône ↖ .

- sélection de l'objet à modifier (courbe, axe)
- bouton droit puis 'properties'

1.6. Manipulation de fonctions de transfert

La procédure de définition des fonctions de transfert est la suivante :

Le numérateur (num) et le dénominateur (den) sont déclarés comme deux vecteurs par ordre décroissant des puissances de « s » (opérateur de Laplace).

On définit ensuite une fonction de transfert à partir de ces deux vecteurs en utilisant la fonction 'tf'.

Exemple :

```
num = [1 2]
den= [1 0.1 1] ;
f = tf(num,den)
```

'f' correspond à la fonction de transfert : $\frac{s+2}{s^2+0.1\,s+1}$

Il est alors possible :

- d'afficher les différents lieux : bode(f), rlocus(f)
- d'examiner la réponse indicielle : step(f) …

Attention, avant d'afficher un de ces graphiques, il faut fermer les autres figures déjà ouvertes (commande 'close all').

A la fin de cet exercice, détruire les variables créées (commande : 'clear').

Remarque générale importante : Dans la plupart des cas, il existe plusieurs syntaxes pour les fonctions Matlab de base. Il est donc toujours intéressant de taper 'help nom de la fonction' de manière à connaître ces différentes possibilités ainsi que les autres fonctions qui s'y rattachent.

1.7. Résumé des principales commandes

path	Spécifie la liste des répertoires connus dans MATLAB
help fonction	Aide en ligne de la fonction 'fonction'
type fichier	Liste le contenu d'un fichier ASCII
whos, who	Liste les variables de l'espace de travail
clear	Efface toutes les variables de l'espace de travail
cd, dir, mkdir	Commande du système d'exploitation disponible dans MATLAB
!, dos	Accès au système d'exploitation
exit, quit	Fin de la session MATLAB

Il est possible de rappeler les précédentes commandes

- Utiliser les touches ↑ ou ↓ du clavier
- taper un ou plusieurs caractères puis les touches ↑ ou ↓, seules les commandes commençant par ces caractères seront alors rappelées.

  Exemple : Si la fonction plot a déjà été utilisée, on peut la rappeler rapidement en tapant 'pl' puis ↑.

2. Conception d'un modèle de simulation SIMULINK

2.1. Présentation de SIMULINK

SIMULINK est un module particulier venant compléter le noyau MATLAB, et fournir une interface graphique pour la modélisation de systèmes dynamiques sous forme de schémas-blocs. Grâce aux nombreux blocs de base fournis, il est possible de créer des modèles sans écrire une seule ligne de code.

L'architecture ouverte permet d'étendre l'environnement de simulation par :

- la création de blocs personnalisés et de bibliothèques à partir du code MATLAB, Fortran ou C ou bien de façon graphique ;
- l'intégration de code Fortran ou C existant pour récupérer des modèles validés ;
- la génération de code C à partir des modèles de simulation.

De même que MATLAB et ses Toolboxes, Simulink peut être complété de bibliothèques de blocs spécialisés - les Blocksets, qui viennent s'ajouter à la bibliothèque de base.

2.2. Mise en équation du processus étudié

Nous nous proposons de simuler le comportement d'un moteur à courant continu. Nous définissons les paramètres du processus les noms des grandeurs importantes et les équations qui décrivent le fonctionnement.

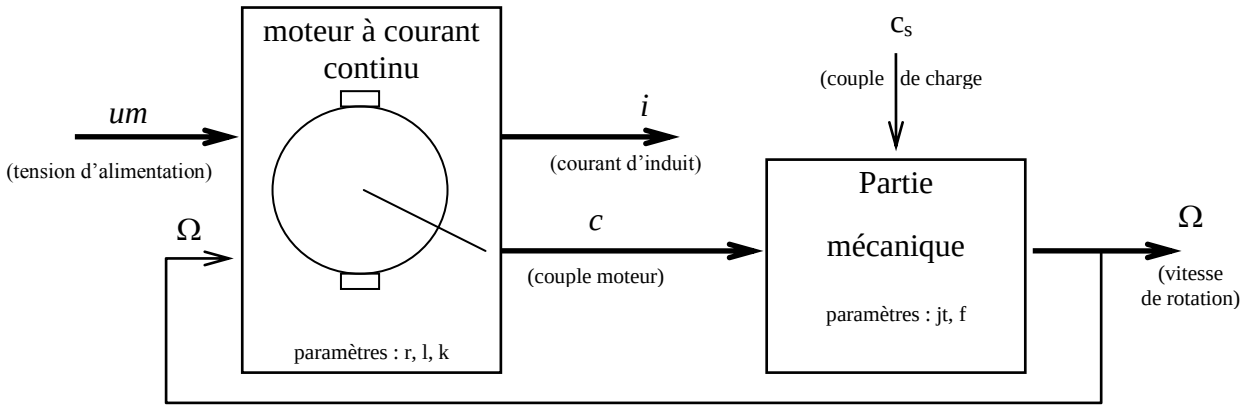


Figure 1 : Moteur à courant continu entraînant une charge mécanique

Paramètres électriques :	r :	résistance d'induit
	l :	inductance d'induit
	k :	coefficient de fem
Paramètres mécaniques :	jt :	Inertie
	f :	Frottement visqueux

Grandeurs physiques du processus : i : courant d'induit
e : fem
c : couple électromagnétique
Ω : vitesse

Grandeur d'entrée : tension d'induit : u_m
Couple de charge : c_s

- Partie électrique : $l\frac{d\,i}{dt} + r\,i = u_m - e$
- Partie mécanique : $j_t\frac{d\,\Omega}{dt} + f\,\Omega = c - c_s$
- Conversion électromécanique : $c = k\,i$; $e = k\,\Omega$

2.3. Constitution du modèle SIMULINK

Les points suivants précisent la méthode de conception d'un modèle de simulation

- **Lancement de Simulink**

Cliquer sur l'icône SIMULINK  située sur la barre principale de MATLAB
Cliquer sur les différents modules de la bibliothèque afin d'en découvrir les différents éléments.

- **Ouverture d'un nouveau modèle**
A partir de la fenêtre SIMULINK :
'**File**' puis '**New**' et '**Model**'

- **Création d'un modèle de simulation**

1. Pour ajouter un bloc, cliquer sur le bloc désiré dans la bibliothèque et le faire glisser vers la fenêtre souhaitée
2. Changer le nom du bloc en cliquant sur la zone 'nom'
3. Pour paramétrer un bloc, double-cliquer sur le bloc.

Remarque : Le paramétrage du bloc fonction de transfert se fait en rentrant le numérateur et le dénominateur par ordre décroissant des puissances de « s ». Il s'agit bien sûr de la même syntaxe que sous MATLAB.

4. Pour relier deux blocs, cliquer sur la sortie du premier bloc et faire glisser la souris jusqu'à l'entrée du second bloc, puis relâcher la souris. On peut faire la même chose en partant de l'entrée du second bloc.
5. Pour copier un bloc, cliquer avec le bouton droit de la souris sur le bloc à copier.
6. Ajouter des commentaires si nécessaire dans la figure en double-cliquant sur la zone souhaitée.

La figure suivante présente le modèle de simulation souhaité.

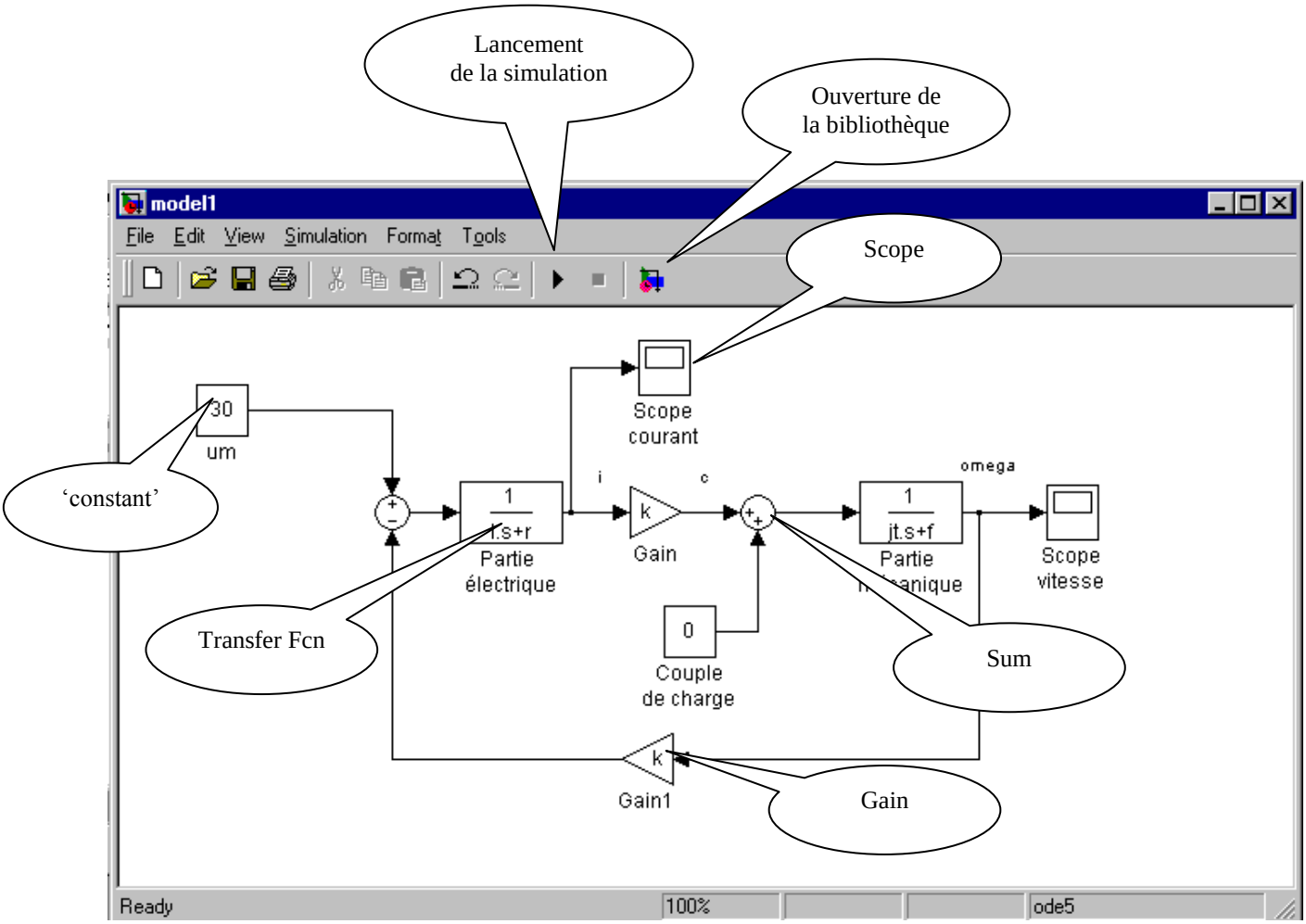


Figure 2 : Modèle de simulation associé au moteur à courant continu

Une fois que le modèle est constitué, sauvegarder le avec le nom de votre choix dans le répertoire courant.

Raccourcis clavier utiles : CTRL R : rotation de blocs
CTRL F : retournement de blocs

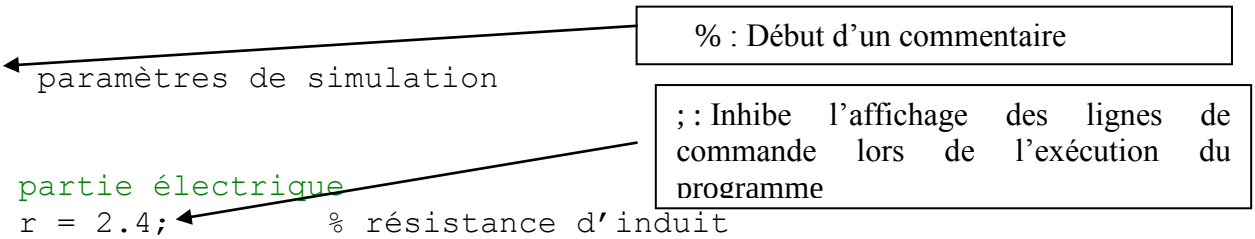
- **Fichier d'initialisation**

Comme on peut le constater sur la figure précédente, il n'y a aucune valeur numérique. Toute les variables utilisées sont en fait des variables MATLAB qui doivent être définies dans l'espace de travail.

Pour ce faire, il est nécessaire dès la conception d'un fichier de simulation, de lister tous les paramètres de la simulation et de les définir dans l'environnement MATLAB grâce à un fichier d'initialisation.

Commande MATLAB : **'File'** puis '**New**' puis '**M-File'**
Une nouvelle fenêtre s'ouvre.

Saisir les paramètres suivants :




```
%  paramètres de simulation


% partie électrique
  r = 2.4;          % résistance d'induit
  l = 40e-3;        % inductance d'induit
  k = 0.139;        % coefficient de couple et de fem

% partie mécanique
  jt = 0.84e-3;     % moment d'inertie équivalent ramené à
                    % l'arbre moteur
  f = 0.001;        % coefficient de frottements visqueux

% paramètres de simulation

  Tmax = 500e-3;
  Dt = 1e-4;
```

Quand le texte est saisi, sauvegarder le fichier ('**File**' puis '**Save**') en lui donnant le nom '**init**'. L'extension « .m » est automatiquement ajoutée.

Revenir sous MATLAB, pour que les instructions du fichier soient prises en compte, il est faut exécuter le programme en tapant : **init**.

- **Paramétrage de la simulation**

Avant de lancer la simulation, il faut la paramétrer (Menu '**Simulation**' puis '**Parameters**')

Simulation Time :
Start Time : 0
Stop Time : Tmax

Solver Options
Type : Fixed-Step ode5
Fixed Step size : Dt

- **Lancement de la simulation**

Menu '**Simulation**' puis '**Start**'

Un signal sonore indique la fin de la simulation.

2.4. Traitement des résultats de simulation : Scopes et WorkSpace

Les ‘Scopes’ permettent de visualiser directement les résultats de simulation et de sauvegarder ces variables dans l’espace de travail MATLAB. Les ‘workspace’ stockent uniquement les variables sans affichage du résultat. Le stockage des informations dans l’espace de travail est utilisé en particulier pour comparer les résultats de différentes simulations.

- **Fenêtre d’origine**

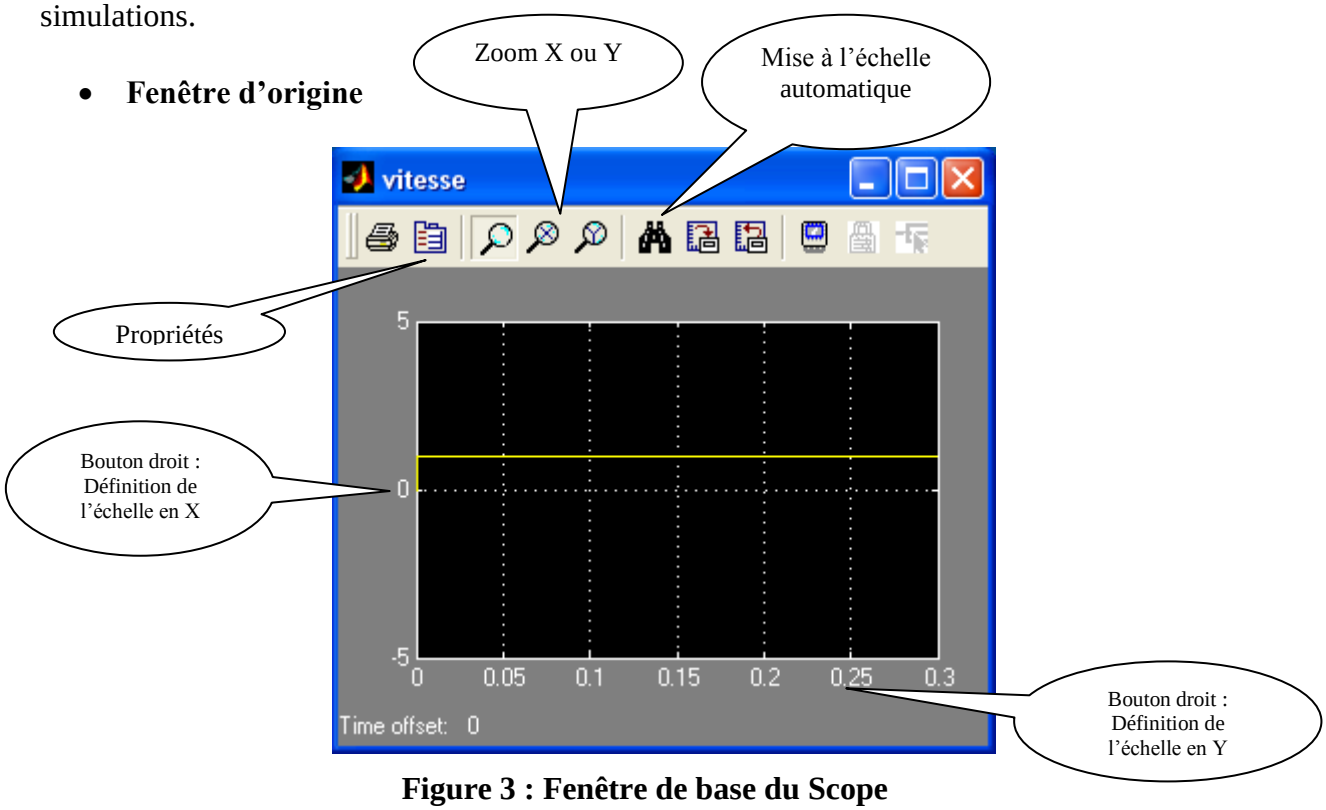


Figure 3 : Fenêtre de base du Scope

- **Stockage des information sous MATLAB :bouton ‘Properties’ puis ‘Data History’**

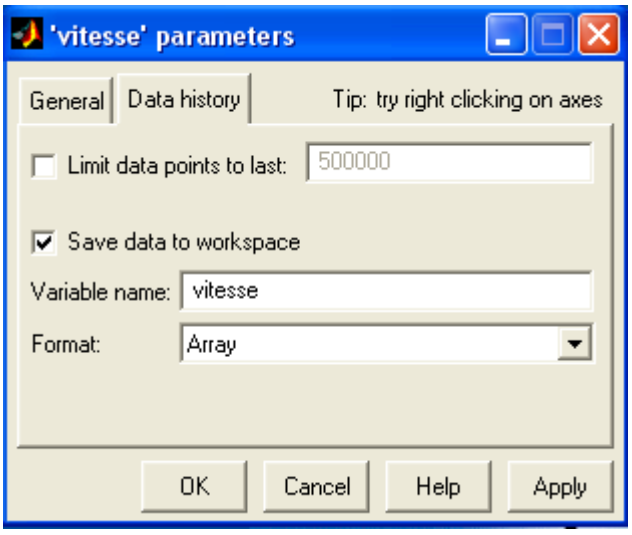


Figure 4 : Paramétrage du stockage d’informations

Save Data to workspace : vitesse ← Nom du vecteur 'MATLAB' pour stocker les informations

Format : array ← Format de stockage des données

- **Format de stockage**

Pour simplifier, nous choisissons un format de stockage de type Matrix. Dans la première colonne est stockée le temps courant de la simulation dans les colonnes suivantes, les variables visualisées.

- **Limitations du nombre de points stockés**

Bien souvent, il est inutile de stocker tous les points de simulation. En effet le traitement ultérieure dans des fichiers texte est alourdi et la simulation ralentie. Il est possible de limiter le nombre de points stockés. Pour cela, il faut cliquer sur le bouton 'Properties' du Scope puis dans la fenêtre 'General' apparaît un champ 'Sampling'.

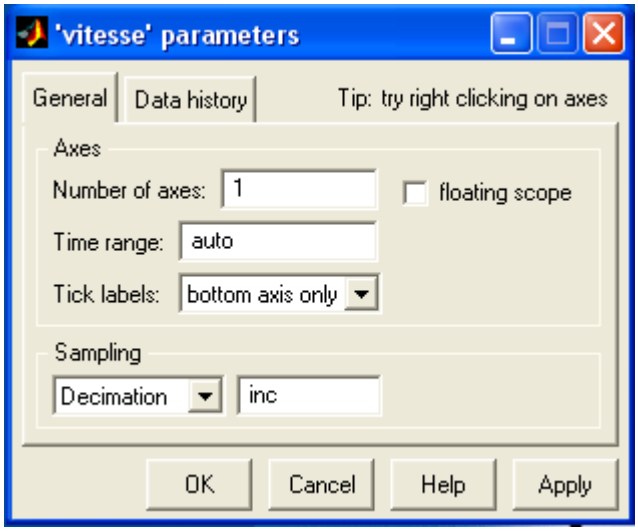


Figure 5 : Limitation du nombre de points

Deux solutions sont envisageables :

- Decimation : Stockage d'un point sur N
- Sampling : échantillonnage toutes les 'x' secondes

Dans le cas présent, nous choisissons l'option 'Décimation', la valeur de l'incrément est fixée à 'inc'. Cette variable est fixée dans le fichier d'initialisation à 5.

Attention : Il est fortement recommandé de mettre le même paramétrage (à définir dans le fichier d'initialisation) dans tous les scopes et workspaces.

Si l'on veut être sûr de ne pas dépasser une valeur limite, il est possible de valider dans le Scope ('**Data history**' puis '**Properties**') l'option : '**Limit Rows to last**'. On saisit alors le nombre N de points maxi à stocker. Dans ces conditions, seuls les N derniers points seront affichés sur le scope et sauvegardés. La procédure est identique avec les blocs 'Workspace'.

Dans le cas où l'on souhaite stocker sur une durée plus importante, il faut stocker moins de points.

- **Exportation vers Word**

Il est possible d'exporter une figure MATLAB vers Word. Pour cela :

- Se placer sur la figure choisie
- Menu '**Edit'** puis '**Copy Figure'**.

La figure est alors insérée dans le presse papier.

2.5. Structuration du fichier : regroupement et création de masques

L'assemblage de blocs de la bibliothèque devient rapidement inextricable si l'on n'organise pas rapidement le fichier en regroupant les blocs par fonctions. SIMULINK permet cette possibilité.

Cette procédure de regroupement est l'occasion de bien réfléchir aux entrées et aux sorties du système étudié.

Il est alors important de bien analyser le processus sous forme causale.

La procédure est la suivante :

1. Sélectionner les blocs à regrouper avec la souris.
2. Grouper avec le menu ('**Edit**' '**Create Subsystem**').
3. Un nouveau bloc remplace tous les blocs qui étaient sélectionnés.
4. En double cliquant sur ce nouveau bloc, une fenêtre s'ouvre avec tous les blocs qui ont été groupés. Les entrées et les sorties sont représentées respectivement par des bloc '*In*' et '*Out*'.

Remarque : La sélection des éléments doit être opérée en une seule fois. Il est donc nécessaire de bien disposer les blocs à sélectionner au préalable.

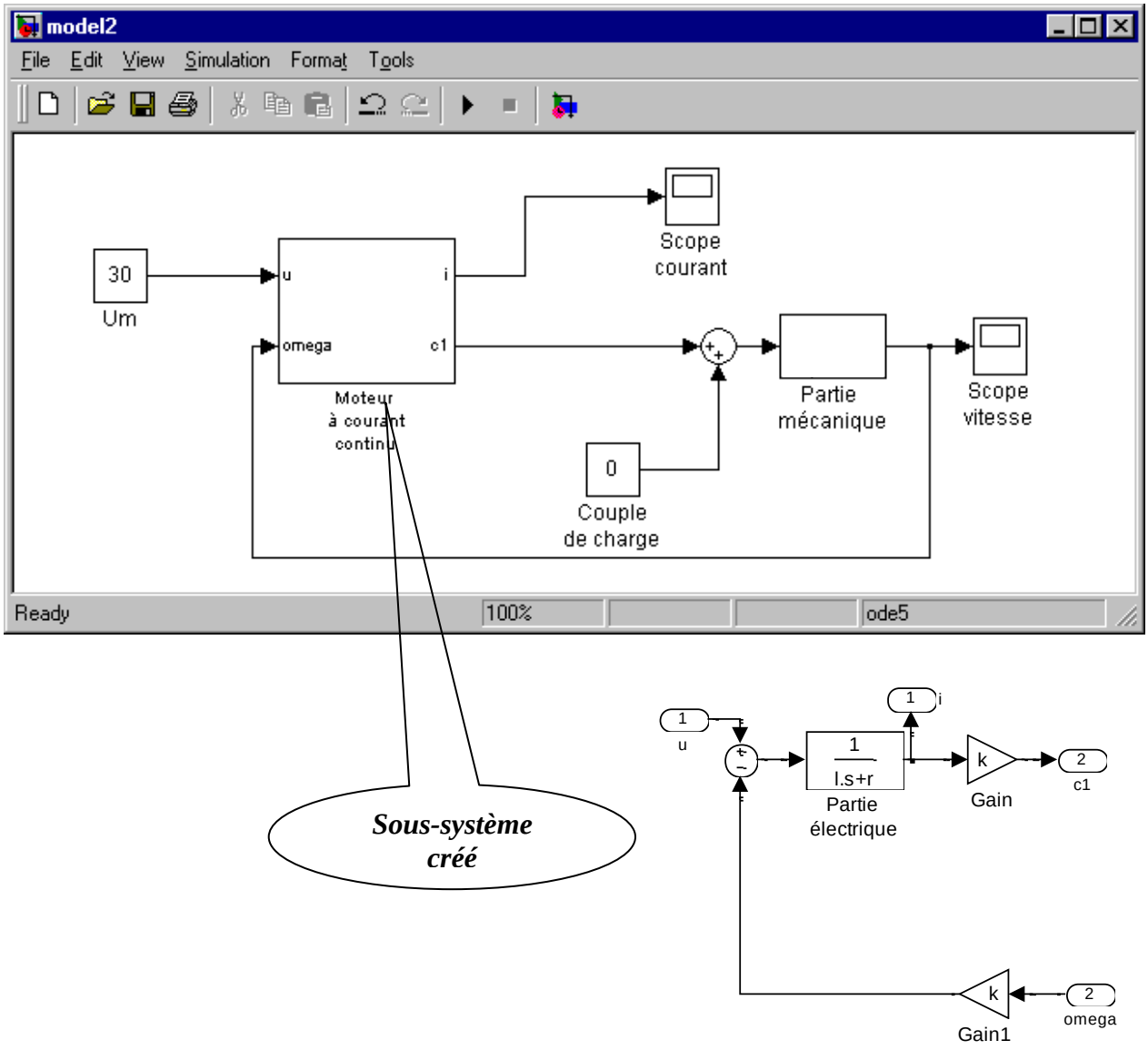


Figure 6 : Regroupement de blocs

3. Etude de la boucle de courant du moteur à courant continu

On souhaite asservir le courant du moteur à courant continu dont le comportement à été simulé. Avant de continuer l'étude :

- Enregistrer le dernier fichier construit.
- Enregistrer le à nouveau sous le nom '**boucle_courant1**' ('**File**' puis '**Save as**').

C'est ce dernier fichier qui va servir de base pour la construction des modèles de simulation suivants.

3.1. Structure de la commande

La figure 7 présente la structure de base de la boucle de courant. Nous n'avons représenté du moteur que la partie qui nous intéresse pour l'asservissement du courant. Le système est du

premier ordre, la force électromotrice (fem) représente une perturbation pour le réglage du courant.

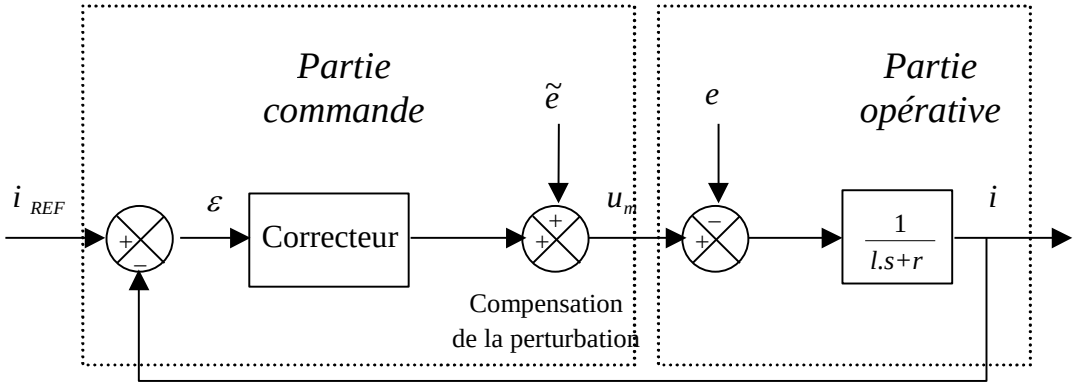


Figure 7 : Structure de la commande

Dans la partie commande on remarque la présence :

- d'une fonction de correction ;
- d'une fonction de compensation de la perturbation.

Sachant que la fem n'est pas mesurable, il est nécessaire de l'estimer à partir de la vitesse : $\tilde{e} = \tilde{k}\ \Omega$

Afin de distinguer la fem réelle de la fem estimée, nous notons cette dernière $\tilde{e}$. En toute rigueur, il est impossible de connaître la valeur exacte de k, c'est pour cela que nous l'avons noté $\tilde{k}$.

3.2. Correcteur proportionnel

Le correcteur choisi est du type proportionnel de gain k_{i1} fixé à 10.

Définir dans le fichier d'initialisation les variables **ki1** : gain du correcteur et **k_est** (pour $\tilde{k}$). Sauvegarder puis exécuter le fichier d'initialisation pour que les modifications soient prises en compte. Le courant de référence est fixé à 1A.

Modifier le fichier **boucle_courant1** pour intégrer la boucle de courant.

Attention : la disposition des blocs a une grande importance pour la faciliter la compréhension du modèle. La figure suivante donne une allure du modèle à construire.

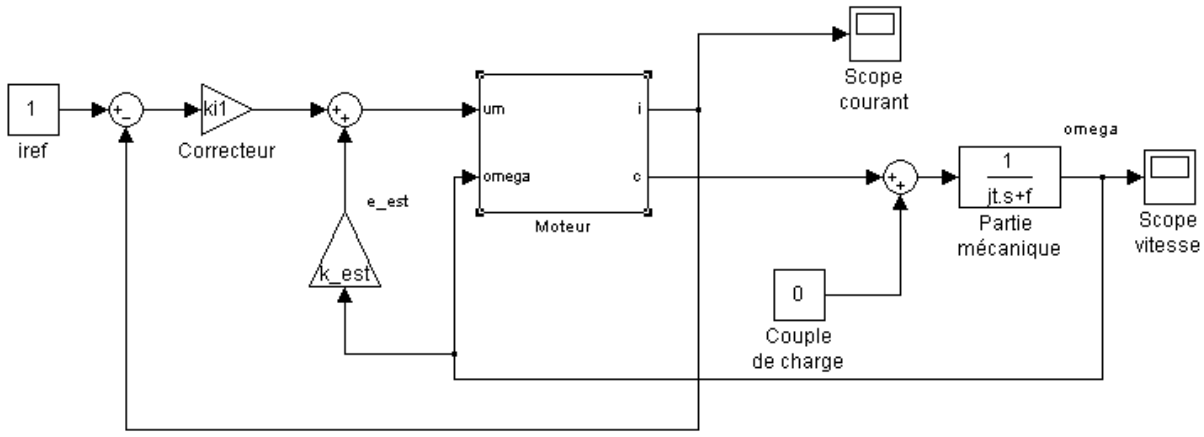


Figure 8 : Modèle de simulation de la boucle de courant

Questions :

- Déterminer et justifier théoriquement l'erreur statique entre le courant *i* et sa référence (détermination de la fonction de transfert du système bouclé puis application du théorème de la valeur finale).
- Sauvegarder les résultats de simulation pour le courant puis tracer sur une figure MATLAB, le courant en fonction du temps.
- Déterminer le temps de réponse. Justifier théoriquement cette valeur.
- Par application du théorème de la valeur initiale, calculer la pente de la vitesse à l'origine (Hypothèse : on assimilera le courant à sa valeur finale (i_0)).
- Tracer sur une figure MATLAB, la vitesse en fonction du temps, puis la pente de la vitesse.
- Comparer et justifier la légère différence entre le résultat théorique et le résultat de simulation.

4. Conclusion

Ce document n'a pas vocation à remplacer une documentation complète sur le logiciel. Il est surtout conçu pour donner les informations principales pour démarrer dans cet environnement et donne des éléments de base dans la démarche de conception d'un modèle de simulation.

En tout état de cause, de nombreux éléments complémentaires sont disponibles dans les fichiers d'aide du logiciel.