4. Tracer les réponses fréquentielles d'un système

Pour obtenir le diagramme de Bode d'un système, on peut utiliser la commande *bode* >>bode(H) ; grid on



Figure 407 : diagramme de Bode d'un système

Pour obtenir le diagramme de Black-Nichols d'un système, on peut utiliser la commande *nichols* >>nichols(H) ; grid on



Figure 408 : diagramme de Black-Nichols d'un système

Pour obtenir le diagramme de Nyquist d'un système, on peut utiliser la commande *nyquist* >>nyquist(H) ; grid on



Figure 409 : diagramme de Nyquist d'un système

On peut également tracer deux diagrammes de Bode sur le même graphique. >>bode(H,G) ; grid on



Figure 410 : tracé de deux diagrammes de Bode sur le même graphique

5. Evaluer les marges de gain et de phase

Il est possible d'obtenir automatiquement les marges de gain et de phase d'une fonction de transfert en bloucle ouverte avec la commande *margin* qui affiche un diagramme de Bode en donnant les informations numériques et graphiques sur les marges de gain et de phase. >>margin(H) ; grid on ;



Figure 411 : diagramme de Bode avec indication des marges de gain et de phase

Il est également possible de stocker dans des variables les valeurs des marges de gain et de phase et les différentes pulsations auxquelles elles sont évaluées.

>>[gm,pm,wcg,wcp]=margin(H)
gm =
Inf
pm =
132.6226
wcg =
NaN
wcp =
1
2.5255

6. Tableau récapitulatif des commandes utiles sur les fonctions de transfert

Commandes utiles	
Fonctions	Commandes
Creation d'une variable symbolique	syms x
Resolution d une equation algebrique	Solve(equation,x)
Pactoriser une expression	$\frac{1}{1} \frac{1}{1} \frac{1}$
Développer une expression	$\frac{\exp((x+2)^{2})}{\exp((x+2)^{2})}$
Intégrer une fonction	int (ft)
Résolution d'une équation différentielle	dsolve(equation)
Calculer une intégrale définie	int(ft-ni/3 ni)
Calculer la transformée de Laplace d'une fonction	lanlace(sin(w+t))
Calculer la transformée inverse de Laplace d'une	
fonction	ilaplace(3/(2+s)-5/(3*s)^2)
Décomposer en éléments simples une fraction	
rationnelle	[r p k]=residue (a,b)
Création d'une fonction de transfert à partir de la	$H = +f([4 \ 1] [2 \ 2 \ 6])$
définition des coefficients des polynômes	11-0([4 1],[2 3 0])
Création d'une fonction de transfert à partir de la	G=znk([-1-2][-01-4-10]100)
connaissance du gain, des pôles et des zéros	
Indiquer que s est la variable de Laplace	s=tf('s')
Création directe d'une fonction de transfert après	G=100*((s+1)*(s+2))/((s+0.1)*(s+4)*(s+10))
Utilisation de la commande « s=tr(s) »	
correcteur PID	pid(Kp,Ki,Kd)
Fonctions de transfert en série	
	series(H,G)
Fonctions de transfert en parallèle	
	parallel(H,G)
Fonction de transfert en boucle fermée	
	feedback(H,G)
Inverser une fonction de transfert	inv(H)
Obtenir les pôles d'une fonction de transfert	pole(H)
Obtenir les zéros d'une fonction de transfert	zero(H)
Réponse indicielle	step(H)
Keponse impliisionnelle	IMDUISELHI

Diagramme de bode	bode(H)
Diagramme de Black-Nichols	nichols(H)
Diagramme de Nyquist	nyquist(H)
Tracé multiple de diagrammes	bode(H,G) ou nichols(H,G)
Diagramme de bode avec marge de gain et de phase	margin(H)
Récupérer dans des variables les valeurs des marges de gain et de phase et les pulsations correspondantes	[gm,pm,wcg,wcp]=margin(H)

Pour obtenir des informations complémentaires sur une commande, vous pouvez utiliser l'aide de **MATLAB** en utilisant la commande doc.

>>doc bode

permet d'ouvrir la fenêtre d'aide de la commande « bode »

Chapitre 6 : Prise en main de Simulink

I. Introduction

Ce chapitre a pour objectif de présenter au travers d'un exemple une modélisation effectuée avec **Simulink** et de montrer les communications possibles entre l'environnement **MATLAB** et l'environnement **Simulink**.

II. Régulation en température d'un four

L'étude porte sur l'asservissement en température d'un four industriel.



Une résistance chauffante permet de chauffer l'enceinte du four. Un capteur informe la carte de commande de la température à l'intérieur du four. Cette mesure est comparée à la consigne de température pour former un écart. Un correcteur proportionnel et intégral impose une tension de commande à la résistance.

A. Ouverture du modèle

Ouvrir le fichier *four_0.slx* pour visualiser le schéma bloc de la Figure 413 et explorer les différents blocs du modèle. Ce modèle **Simulink** représente l'asservissement en température d'un four. Les éléments modélisés par les blocs sont indiqués sur la Figure 413.



Figure 412 : ouverture du modèle Simulink du four

Nous constatons que certains blocs apparaissent encadrés en rouge. Cela signifie qu'ils sont paramétrés avec des variables qui ne sont pas encore définies dans le Workspace. Le modèle ne peut donc pas être exécuté pour l'instant sans générer un message d'erreur. Lorsque vous construisez un modèle Simulink, vous pouvez paramétrer les blocs directement avec des valeurs numériques ou au travers de variables que vous définirez dans le Workspace, soit directement dans la fenêtre de commande soit via l'éxécution d'un script. Il est très utile de regrouper la valeur de toutes les grandeurs physiques du système dans un script unique. L'exécution du script permettra la création de toutes les variables dans le **Workspace** de **MATLAB**. Il sera alors possible de lancer la simulation dans Simulink.



Figure 413 : modélisation de l'asservissement en température d'un four

B. Ouverture du script contenant la définition des variables

Ouvrir le script parametres_four.m, en cliquant sur Open script

I	5	1
0	per	1

%gain du capteur a=0.5; %tension maxi aux bornes de la résistance Vsat=100; %gain de la resistance chauffante G=0.8; %paramètres de la fonction de transfert du four K=15; T=2500; %paramètres du correcteur Kp=10; Ki=0.01;

Figure 414 : script contenant les paramètres de la simulation

Exécuter le script, en cliquant sur Run



dans la barre de commande.

Nous constatons que les variables ont été créés et sont visible dans le Workspace de **MATLAB** (Figure 415)

Workspace	Workspace					
Name 📥	Value	Class				
🛨 a	0.5000	double				
🕂 G	0.8000	double				
🕂 К	15	double				
🕂 Ki	0.0100	double				
🕂 Кр	10	double				
🕂 Т	2500	double				
🕂 Vsat	100	double				

Figure 415 : visualisation des variables créées dans le Workspace

La simulation peut maintenant être lancée.

C. Lancement de la simulation

Lancer la simulation du modèle Simulink et visualiser la variation de la température à l'intérieur du four (Figure 416).



Figure 416 : réponse en température du four

Une consigne de 100° est imposée à l'entrée du modèle. Une perturbation est visible à l'instant t=600 s et se caractérise par une chute de la température. Le correcteur proportionnel et intégral permet de rétablir la température du four au niveau de la consigne.

D. Tracer un diagramme de Bode avec Simulink

Il existe de nombreuses méthodes pour tracer un diagramme de Bode à partir d'un schéma bloc réalisé avec Simulink. La plus simple consiste à placer des **points de linéarisation** sur le schéma bloc et d'utiliser le bloc **Bode Plot** de la bibliothèque **Simulink Control Design**. Il existe différents types de points de linéarisation, nous allons voir comment les choisir pour obtenir un diagramme de Bode en boucle ouverte et en boucle fermée.

Avant de commencer il est préférable de donner un nom aux signaux du schéma bloc qui interviendront dans les fonctions de transfert à tracer.

Nommer les signaux conformément à la Figure 417 :

- Entrée
- Sortie
- Ecart
- Retour

Pour nommer un signal **cliquer avec le bouton droit** sur le signal puis choisir **Properties**. La boite de dialogue **Signal Properties** s'ouvre. Le nom du signal est à spécifier dans **Signal Name**.



Figure 417 : nommer un signal Simulink

1. Tracer un diagramme de Bode en boucle ouverte

Pour tracer un diagramme de Bode en boucle ouverte, il faut placer deux points de linéarisation sur les signaux d'entrée et de sortie de la boucle ouverte. Pour cela il faudra choisir des points de linéarisation de type :

- **Open loop input** (pour l'entrée de la boucle ouverte)
- Open loop output (pour la sortie de la boucle ouverte)



Figure 418 : placement des points de linéarisation pour tracer un diagramme de Bode en boucle ouverte

Pour placer un point de linéarisation de type **open loop input** sur le signal « **écart** », il faut **cliquer avec le bouton droit** sur le signal « **écart** », puis choisir **Linear Analysis point/open loop Input**.



Figure 419 : placement d'un point de linéarisation de type « Open loop input »

Pour placer un point de linéarisation de type **open loop output** sur le signal « **retour** », il faut **cliquer avec le bouton droit** sur le signal « **retour** », puis choisir **Linear Analysis point/open loop Output**.



Figure 420 : représentation des points de linéarisation

Fonction du composant Représentation		Bibliothèque
Tracer d'un diagramme de Bode	Bode Plot	Simulink/Simulink Control Design/Linear Analysis Plot

Insérer dans votre modèle un bloc Bode Plot, renommer ce bloc « boucle ouverte ».





Double cliquer sur le bloc pour l'ouvrir.

🚹 Block Paramete	ers: Boucle ouverte					\times	
Bode Plot	Bode Plot						
Compute and dis bounds on the lir	Compute and display a linear system on a Bode plot. You can also specify bounds on the linear system and assert that the bounds are satisfied.						
Linearizations	Bounds Logg	ing /	Assertion				
Linearization in	outs/outputs:						
Block : Po	ort : Bus Element		Conf	iguration			
four_0_Bode_0)/Sum : 1	Ор	en-loop I	nput	-		
four_0_Bode_0)/capteur : 1	Ор	en-loop (Dutput	• +		
					×		
					4	5	
					20	Þ	
Linearize on:	Simulation snap	shots			•	•	
Snapshot times	: [0]					:	
Trigger type:	Rising edge				,	~	
 Algorithm Op 	otions						
Labels							
Show Plot	Show plot on block	open		Response O	ptimizatio	on	
0	ОК	Ca	ncel	Help	App	ly	

Figure 422 : fenêtre de paramétrage du bloc Bode Plot

Par défaut les deux points de linéarisation créés apparaissent. Cliquer sur **Show Plot** pour faire apparaître la fenêtre de tracé.



Figure 423 : fenêtre de tracé du diagramme de Bode

Cliquer sur Run

pour linéariser le modèle. Le diagramme de Bode se trace automatiquement.



Figure 424 : diagramme de Bode de la boucle ouverte

2. Tracer un diagramme de Bode en boucle fermée

Pour tracer un diagramme de Bode en boucle fermée, il faut placer deux points de linéarisation sur les signaux d'entrée et de sortie de la boucle fermée. Pour cela il faudra choisir des points de linéarisation de type :

- Input Perturbation (pour l'entrée de la boucle fermée)
- Output Measurement (pour la sortie de la boucle fermée)

Pour placer un point de linéarisation de type **Input Perturbation** sur le signal « **entrée** », il faut **cliquer avec le bouton droit** sur le signal « **entrée** », puis choisir **Linear Analysis point/Input Perturbation**.

Pour placer un point de linéarisation de type **Output Measurement** sur le signal « **sortie** », il faut **cliquer avec le bouton droit** sur le signal « **sortie** », puis choisir **Linear Analysis point/Output Measurement**.

Les différents points de linéarisation apparaissent sur la Figure 425.



Figure 425: représentation des points de linéarisation

Insérer dans votre modèle un second bloc Bode Plot, renommer ce bloc « boucle fermée ».



Figure 426 : insertion d'un second bloc Bode plot

Double cliquer sur le bloc pour l'ouvrir.

🛅 Block Parameters: Boucle fermée 🛛 🗙					
Bode Plot					
Compute and disp bounds on the line	Compute and display a linear system on a Bode plot. You can also specify bounds on the linear system and assert that the bounds are satisfied.				
Linearizations Bounds Logging Assertion					
Linearization inp	uts/outputs	:			
Block : Po	rt : Bus Eler	nent	Confi	guration	
four_0_Bode_0	/Sum : 1		Open-loop Ir	nput	•
four_0_Bode_0	/Températu	re de c	Input Pertur	oation	+
four_0_Bode_0	/capteur : 1		Open-loop O	utput	• ×
four_0_Bode_0	/four:1		Output Meas	urement	•
					4
					2
Linearize on:	Simulation	n snapshot	s		•
Snapshot times:	[0]				:
Trigger type:	Rising edg	je			~
Algorithm Opt	tions				
Labels					
Show Plot S	how plot on	block ope	en 🛛	Response Op	timization
0	C	Ж	Cancel	Help	Apply

Figure 427: fenêtre de paramétrage du bloc Bode Plot

Par défaut les quatre points de linéarisation créés apparaissent.

Il suffit de garder les deux points de linéarisation correspondant à la boucle fermée et de supprimer les deux points de linéarisation correspondant à la boucle ouverte.

Pour cela, sélectionner les points de linéarisation à supprimer et utiliser le bouton Delete Selected

Linearization I/Os pour obtenir la configuration de Figure 428.

📓 Block Parameters: Boucle fermée 🛛 🗙							
Bode Plot							
Compute and disp bounds on the lin	Compute and display a linear system on a Bode plot. You can also specify bounds on the linear system and assert that the bounds are satisfied.						
Linearizations Bounds Logging Assertion							
Linearization inp	outs/output	s:					
Block : Po	rt : Bus Ele	ment	Config	guration			
four_0_Bode_0	/Températ	ure de c	Input Perturb	oation	•		
four_0_Bode_0	/four:1		Output Meas	urement	+		
					×		
					÷		
					-		
Linearize on:	Simulatio	n snapshot	s		-		
Snapshot times:	[0]				:		
Trigger type:	Rising ed	lge			~		
Algorithm Op	Algorithm Options						
Labels							
Show Plot	show plot o	n block ope	en I	Response Op	timization		
0		ОК	Cancel	Help	Apply		

Figure 428 : paramétrage des points de linéarisation pour la boucle fermée

Cliquer sur **Show Plot** puis **cliquer** sur **Run** pour faire apparaître le diagramme de Bode



Figure 429 : diagramme de Bode de la boucle fermée

Si l'on souhaite ouvrir la fenêtre graphique du diagramme de Bode par un double clic sur le bloc **Bode Plot**, il suffit de cocher la case **Show plot on block open**.

Show Plot	Show p	lot on block (Response Op	timization	
0		OK	Cancel	Help	Apply

Nous disposons alors du modèle Simulink que l'on peut modifier facilement et visualiser très rapidement l'allure des diagrammes de Bode. A chaque modification du modèle, il faudra relancer la linéarisation du modèle pour obtenir le nouveau diagramme de Bode.

E. Tracer d'un diagramme de Black-Nichols

La méthode est rigoureusement la même que pour tracer un diagramme de Bode, en utilisant le bloc **Nichols Plot**.

Le fichier contenant le modèle complet est disponible sous le nom *four_0_Bode_Nichols.slx*

F. Ajout et paramétrage d'une saturation

Nous allons ajouter un saturateur pour tenir compte de la limite de 100 V représentant la tension maximale de commande de la résistance.

Les non-linéarités (seuil, hysteresis...) de la bibliothèque Simulink se trouvent dans Simulink/Discontinuities.



Figure 430 : asservissement en température d'un four avec saturation

Paramétrage

Saturation



Simulink/Discontinuities

Ce composant permet de limiter la sortie du bloc entre deux valeurs mini et maxi de saturation Ici la valeur maximale est donnée par la variable Vsat, la valeur mini étant laissée à 0.

🎦 Block	Parameters: Saturation X	
Saturati	on	
Limit inp	out signal to the upper and lower saturation values.	
Main	Signal Attributes	
Upper lin	nit:	
Vsat	1	
Block Parameters: Saturation × Saturation Limit input signal to the upper and lower saturation values. Main Signal Attributes Upper limit: vsat Itemation Itemation<		
0	1	
🗹 Treat	as gain when linearizing	
🗹 Enable	e zero-crossing detection	
0	OK Cancel Help Apply	

Lancer la simulation et visualiser la réponse obtenue avec la saturation.



Figure 431 : réponse en température de four avec saturation

Nous constatons que la saturation ralentit le système, le temps de réponse à 5% est plus élevé. Pour visualiser la tension en sortie de saturateur il est possible de placer un scope classique. Cependant afin de visualiser des signaux sans avoir à surcharger la modélisation MATLAB propose d'autre mode de visualisation en plaçant un scope directement sur un signal.

Pour cela cliquer droit sur le fil du signal puis sélectionner Create&connect Viewer/Simulink/Scope



Un petit scope apparaît sur le signal en sortie de saturateur.



Figure 432 : mise en place d'un scope sur un signal

Relancer la simulation et double cliquer sur le scope du saturateur pour visualiser la tension en sortie du bloc.



Figure 433 : visualisation de la tension en sortie de saturateur

Nous constatons que la tension sature à 100 V durant environ 200 s ce qui explique le ralentissement du système.

G. Exportation des variables de la simulation vers le Workspace

Il est très souvent utile d'exporter des variables qui contiennent le résultat de simulation vers le **Workspace**. Cela permet d'utiliser toutes les fonctions de **MATLAB** pour la présentation des résultats.

Nous souhaitons tracer un réseau de courbes permettant de voir l'influence du gain **Kcor** du correcteur proportionnel sur la température du four. Il faudra pour cela créer une variable **Tf** correspondant à la température du four et exporter cette variable vers le Workspace à l'aide d'un bloc **To Workspace** que l'on trouve dans la bibliothèque **Simulink/Sinks**.



Ce composant permet de créer une variable pour un signal **Simulink** et de l'exporter vers le Workspace de **MATLAB**. Il suffit de spécifier le nom de la variable (ici **Tf**) et son format. Pour exporter la variable sous la forme d'un simple vecteur il faut choisir le format **Array**.

🚹 Block Parameters: To	o Workspace	\times					
To Workspace							
Write input to specified timeseries, array, or structure in a workspace. For menu-based simulation, data is written in the MATLAB base workspace. Data is not available until the simulation is stopped or paused.							
To log a bus signal, u	se "Timeseries" save format.						
Parameters							
Variable name:							
Tf							
Limit data points to la	ast:						
inf		:					
Decimation:							
1		:					
Save format: Array		•					
Save 2-D signals as:	3-D array (concatenate along third dimension)	•					
✓ Log fixed-point da	ta as a fi object						
Sample time (-1 for i	nherited):						
-1		:					
	OK Cancel Help Appl	у					

Pour pouvoir tracer ensuite la variable **Tf** en fonction du temps, il faudra un second vecteur représentant les abscisses et qui sera constitué des différentes valeurs du temps correspondants aux pas de calcul du solveur. Cette variable est générée automatiquement par Simulink et exporter vers le Workspace. Cette variable est nommée par défaut **tout**.

Pour modifier le nom de cette variable **tout**, **cliquer** sur la configuration du solveur de Simulink choisir l'onglet **Data Import/Export** dans la partie gauche de la fenêtre. Remplacer **tout** par **t**.

Configuration Parameters: four_	0_sat/Configuration (Active)		- 🗆 X
Q Search			
Solver Data Import/Export Math and Data Types • Diagnostics Hardware Implementation Model Referencing Simulation Target • Code Generation	Load from workspace		Connect Input
	Time: States: Output: Final states: Signal logging: Data stores: Log Dataset data to file: Single simulation output: Simulation Data Inspector Record logged workspace Additional parameters	t xout xout yout xFinal logsout dsmout out mat out mat out data in Simulation Data Inspector	Format: Array
			OK Cancel Help Apply

Figure 434 : fenêtre de paramétrage de l'exportation des données vers le Workspace

Enregistrer le modèle sous le nom *four_export_name.slx*

Si nécessaire, le modèle complet est disponible dans le fichier *four_export.slx*.

Lancer la simulation.

Observer le Workspace de **MATLAB** et vérifier la création des variables t et Tf.

Dans la fenêtre de commande de MATLAB taper la commande suivante : >>plot(t,Tf) ;grid on ;

et



La courbe représentant la température en fonction du temps apparaît.

1. Ecriture d'un script pour tracer une série de courbes

La commande *sim* permet de lancer une simulation d'un modèle Simulink à partie d'une ligne de commande. Nous allons l'utiliser pour écrire un script permettant de visualiser l'influence du gain du correcteur proportionnel.

Taper le script suivant, enregistrer le et exécuter le.

```
close all;
figure;
hold all;
grid on;
% pour Kp variant de 1 à 10 avec un pas de 1 (valeur par défaut du pas)
% exécute la simulation du fichier simulink four_import
% trace la température en fonction du temps
for Kp=1:10
    sim('four_export');
    plot(t,Tf,'LineWidth',2);
end
```

Figure 435 : script pour tracer une série de courbes

Le script est disponible dans le fichier *script_courbes_Kcor.slx*.

Vous devez obtenir le résultat suivant.



Figure 436 : influence de la correction proportionnelle sur la température du four

H. Linéarisation d'un modèle Simulink à partir d'un script

Il est souvent utile de linéariser un modèle Simulink à partir d'un script pour pouvoir exploiter les résultats.

Ouvrir le fichier **four_linearisation.slx**. Ce modèle contient le modèle du four avec les deux points de linéarisation permettant de définir la fonction de transfert en boucle ouverte (Figure 437).



Figure 437 : modèle du four pour la linéarisation

Il est possible à partir du script de la Figure 438 de tracer le diagramme de Bode en boucle ouverte du schéma bloc Simulink (**linearisation_schema_bloc.slx**).

```
% affectation dans une variable du nom du fichier contenant le schéma bloc
% à linéariser
model = 'four_linearisation';
% récupération des points de linéarisation
io = getlinio(model);
% création de la fonction de transfert à partir des points de linéarisation
FTBO = tf(linearize(model,io));
% tracé du diagramme de BOde en boucle ouverte
bode(FTBO);
grid on;
```



Figure 438 : script permettant de linéariser un modèle Simulink

Figure 439 : diagramme de Bode obtenu en linéarisant le modèle Simulink

Le script de la Figure 440 permet de tracer une série de diagramme de Bode en faisant varier le gain du correcteur proportionnel (**linearisation_schema_bloc_courbes_Kcor.slx**).

```
% affectation dans une variable du nom du fichier contenant le schéma bloc
% à linéariser
model = 'four_linearisation';
for Kp=1:10
% récupération des points de linéarisation
io = getlinio(model);
% création de la fonction de transfert à partir des points de linéarisation
FTBO = tf(linearize(model,io));
% tracé du diagramme de BOde en boucle ouverte
bode(FTBO);hold on;
end
```

Figure 440 : script permettant de tracer une série de diagrammes de Bode en faisant varier le gain proportionnel du correcteur.



Le résultat de l'exécution du script est visible sur la Figure 441.

Figure 441 : diagrammes de Bode obtenus en faisant varier le gain du correcteur proportionnel

Chapitre 7 : Prise en main de Stateflow

I. Introduction à Stateflow

Stateflow est le module de MATLAB permettant de modéliser les comportements séquentiels des systèmes sous la forme de machines à états. Une machine à état est constituée « d'états » qui représentent les différents modes du système et de « transitions » qui traduisent les conditions de passage d'un état à un autre. Lors de l'activation d'un état, des actions peuvent être effectuées par le système, elles sont alors indiquées dans l'état correspondant.

A. Modélisation d'une machine à état avec Stateflow

Nous allons créer un diagramme d'état élémentaire pour modéliser la logique de la boucle de cap du pilote hydraulique de bateau.

Le système reçoit deux variables d'entrée :

- **mes_cap** : cap réel du bateau mesuré à l'aide d'un compas
- **cons_cap** : consigne de cap du bateau

Le système renvoie une variable de sortie. :

• **cons_barre** : consigne angulaire de la barre du bateau

Afin de ne pas solliciter la batterie du pilote en permanence, la correction de cap ne se fera qu'à partir du moment ou l'erreur de cap atteint un seuil de 1°.

Le graphe d'état comporte 3 états distincts :

Etat « Pause »

Le système est dans cet état si l'erreur de cap est inférieure à 0.1°. Le pilote hydraulique maintient la consigne de barre à 0° tant que l'erreur de cap ne dépasse pas 0.1°. En cas de dépassement de ce seuil, le système entre dans l'état « Temporisation ».

Etat « Temporisation »

Le système entre dans cet état si l'erreur de cap a dépassé le seuil de 0.1°. Si l'erreur de cap se maintient pendant 20s au-dessus de 0.1°, le système passe dans l'état « correction de cap ». Si durant ces 20s, l'erreur de cap repasse sous le seuil de 0.1°, le système retourne dans l'état « Pause ».

Etat «Correction de cap »

Le pilote hydraulique corrige le cap du bateau, la consigne de barre est obtenue à partir de l'erreur de cap. Le système sort de l'état « Correction de cap » et entre dans l'état « Pause » quand l'erreur de cap redevient inférieure à 0.05°.

B. Construction du diagramme d'état

1. Ouverture du modèle

Ouvrir le modèle *pilote_hydraulique_stateflow_0.slx*.

Ce modèle contient la modélisation du pilote hydraulique de bateau à l'exception du diagramme d'état qui gère la boucle de cap.

Double cliquer sur le sous-système « chaine d'information » et observer le diagramme à compléter sur la Figure 442.



Figure 442 : boucle de cap sans le diagramme d'état

On observe sur le modèle que la boucle de cap ne comporte pas de logique de commande, nous allons modéliser cette logique de commande à l'aide d'un diagramme d'états.

2. Insertion d'un « chart »

Le composant qui permet de construire un diagramme d'état s'appelle un « chart ». C'est lui qui va contenir les états et les transitions.

Insérer un nouveau « chart » à partir de la bibliothèque Stateflow (Figure 443).



Figure 443 : bibliothèque Statflow

Fonction du composant	Représentation	Bibliothèque
Chart	Chart	Statflow

Positionner et **redimensionner** le chart pour obtenir la configuration de la Figure 444.



Figure 444 : positionnement d'un chart dans un modèle Simulink

On remarque que pour l'instant le « chart » ne possède aucune entrée/sortie. Elles seront définies ultérieurement.

C. Création d'un diagramme d'état élémentaire

Double cliquer sur le « chart » pour ouvrir l'environnement **Stateflow**.

La barre de commande **Stateflow** apparait à gauche de la fenêtre avec en particulier les commandes qui permettent de créer un état et une « transition par défaut ».





A partie de l'onglet MODELING, cliquer sur **Symbols Pane** afin de faire apparaitre la fenêtre du **Symbols Pane**.



Figure 446 : ouverture de la fenêtre Symbols

La fenêtre **Symbols** apparaît alors sur la partie droite de l'écran. Cette fenêtre permettra de visualiser la nature des variables utilisées dans la conception du graphe d'état avec Stateflow.



Figure 447 : environnement de travail de Stateflow

1. Création des états

Il est maintenant possible de créer les états pour programmer la logique du système.

A l'aide de la commande de création d'un nouvel état, placer et nommer les trois états conformément à la Figure 448. Pour cela **cliquer** avec le bouton gauche de la souris sur la commande de création d'un nouvel état et faire glisser l'état dans la fenêtre de travail. Lors de la création du premier état, une transition par défaut est automatiquement ajoutée.



Figure 448 : création des états du système

Dans cet exemple, nous allons commencer par travailler avec des états **exclusifs**, c'est-à-dire qu'à chaque pas de temps, un seul état pourra être actif. Nous verrons par la suite qu'il est également possible de créer des états **parallèles** qui pourront être activés simultanément.

2. Création d'une transition par défaut

Placer une transition par défaut sur l'état « Pause ». La transition par défaut permet d'activer un état en début de simulation. La présence d'une **transition par défaut** dans le diagramme est **indispensable**.



Figure 449 : création d'une transition par défaut

3. Création des transitions

Pour créer une transition, déplacez le curseur de souris sur le bord de l'état de départ de la transition. Lorsqu'il prend la forme d'une croix, cliquer avec le bouton gauche et glisser avec le curseur jusqu'à l'état destination de la transition.

Créer les transitions pour obtenir la configuration de la Figure 450

Les transitions sont munies de poignées invisibles qui permettent de leur donner la forme souhaitée. Pour les déformer, cliquer avec le bouton gauche de la souris sur la transition et utiliser le glisser déposer.





4. Création des actions dans les états

Il faut maintenant affecter à chacun des états les actions à imposer au système. Pour cela il faut utiliser un mot clé d'action qui déterminera le moment où l'action sera exécutée.

Les mots clés usuels sont :

- entry (ou en) : l'action sera exécutée uniquement lors de l'activation de l'état
- exit (ou ex) : l'action sera exécutée uniquement lors de la désactivation de l'état
- during (ou du) : l'action sera exécutée à chaque pas de temps tant que l'état est actif.

Si aucun mot clé n'est spécifié, par défaut l'action sera de type **entry,during** et sera exécutée à la fois lors de l'activation de l'état et à chaque pas de temps tant que l'état est actif. Compléter les actions des états conformément à la Figure 451.



Figure 451 : affectation des actions dans les états

Pour l'état « **Pause** » et « **Temporisation** », l'utilisation du mot clé **entry** impose l'action cons_cap=0, uniquement à l'activation de l'état. La variable **cons_barre** reste à 0 et ne varie pas quand ces états sont actifs. Par contre lorsque le système est dans l'état « **Correction_de_cap** », la variable **cons_barre** doit être modifiée continuellement lors de l'évolution du cap du bateau d'où la nécessité d'utiliser le mot clé **during**.



5. Création des étiquettes de transitions

Il faut maintenant définir les conditions qui permettront de passer d'un état à un autre. Les étiquettes de transitions représentent des conditions logiques qui doivent être vérifiées pour permettre le passage d'un état à un autre.

Les informations présentes dans l'étiquette de transition peuvent être de différentes natures (Figure 452).

Une condition	[]	
Un évènement	÷.	
Une action de condition		
Un commentaire	% commentaires	

Figure 452 : nature des informations d'une étiquette de transition

Pour créer une étiquette de transition, double cliquer sur la transition pour faire apparaître le menu vous permettant de sélectionner la nature de l'information que vous souhaitez indiquer dans la transition. Cliquer alors sur puisque nous allons saisir des **conditions** sur chacune des transitions.

Pour déplacer une étiquette de transition, utiliser un simple glisser déposer avec le bouton gauche de la souris.

Créer les étiquettes des transitions conformément à la Figure 453.



Figure 453 : écriture des étiquettes de transitions dans un diagramme d'états

Les étiquettes de transitions offrent d'autres possibilités qui sont décrites page 368.

6. Définitions des variables d'entrée et de sortie du diagramme d'état

Si nous retournons dans le sous-système représentant la chaîne d'information, nous remarquons que le chart ne dispose pas d'entrée et de sortie pour être connecté avec le système.



Figure 454 : chart non connecté avec le modèle

Nous pouvons également constater que la fenêtre Symbols montre que la configuration des entrées et sorties de notre système de commande n'est pas encore totalement définie.

Symbols				ą	×
🚳 🐁 🔍 📰 🎯 🗸 🍞 filter					
TYPE	NAME	VALUE	PORT		
B	cons_cap				
B	mes_cap				
8	cons_barr				

Figure 455 : fenêtre Symbols avec paramétrage incomplet des entrées/sorties

Il existe plusieurs moyens de créer les variables d'entrée et de sortie. Le plus simple est de lancer la simulation, **Stateflow** proposera automatiquement une affectation pour les entrées et les sorties du chart en fonction de la logique qui vient d'être programmée.

Lancer la simulation du modèle.

Un message d'erreur apparaît indiquant que la simulation est impossible et la fenêtre du **Symbol Wizard** s'ouvre en proposant une affectation des variables par défaut.

Vérifier que les affectations proposées par défaut correspondent bien à la logique de commande du système. Si nécessaire, il est possible de modifier les affectations à l'aide des menus déroulants.

Symbol Wizard X				×	
Unresolved symbols found in <u>Chart</u> . All selected data/ events/messages will be created in the chart					
	Name	Class	Sc	ope	
	cons_cap	Data 🝷	Input	•	
	mes_cap	Data 🝷	Input	-	
	cons_barre	Data 🝷	Output		
View created data/events/messages in Model Explorer					
OK Cancel Help					

Figure 456 : affectation des variables du diagramme à l'aide du Symbol Wizard

Nous constatons que la fenêtre Symbols indique maintenant que la configuration des entrées et des sorties du modèle Stateflow est finalisée. Cette fenêtre permet de créer rapidement une variable supplémentaire ou un évènement (Figure 457).



Figure 457 : la fenêtre Symbols

De manière générale, il est également possible de visualiser en détail, les variables du modèle, à l'aide de la fenêtre du **Model Explore.** Il est possible d'ouvrir le **Model Explorer** (Figure 458) pour modifier les

caractéristiques d'une variable en cliquant sur adans la barre de commande principale (onglet MODELING). L'arborescence située sur la partie gauche de la fenêtre permet de naviguer dans le système.

🖮 Model Explorer		– 🗆 X
File Edit View Tools Add Help		
👧 🗀 ½ 🖻 🛱 💥 📖 🌻 🗎 .	🖌 fx 📑 🖶 🕾 🖬 🛄	
Model Hierarchy 🖉 🕫	Contents of:0_0/Simulink et StateFlow/Chart (only) Filter Contents	Chart: Chart
V Pa Simulink Root		General Fixed-point properties Documentation
Base Workspace	Column View: Stateflow Show Details 3 of 11 object(s) y	Name: Chart
pilote_hydraulique_Stateflow_0_0	Name Scope Port Resolve Signal DataType Size InitialVa	Machine: (machine) pilote_hydraulique_Stateflow_0_0
Configurations	cons_cap Input 1 Inherit: Same as Simulink -1	Action Language: MATLAB
Model Workspace	mes_cap Input 2 Inherit: Same as Simulink -1	
> 🔎 External Data	Cons_barre Output 1 Inherit: Same as Simulink -1	State Machine Type: Classic
> 🔁 Fluids		Update method: Inherited Sample Time: -1
> 🔁 Multibody		User-specified state/transition even tion order
> 👌 Simscape Electrical Library		
> 🔁 Simulink		
Simulink et StateFlow		Execute (enter) Chart At Initialization
> 📑 Chart		Initialize Outputs Every Time Chart Wakes Up
		Enable Super Step Semantics
		Support variable-size arrays
		Saturate on integer overflow
		Generate preprocessor conditionals
		Create output for monitoring: Child activity
	S Contents Search Results	Revert Help Apply
× >	Currents Startin Results	

Figure 458 : visualisation des variables dans le Model Explorer

Il est possible de modifier les caractéristiques des différentes variables. Vous pouvez également revenir à tout moment dans le **Model Explorer** pour modifier les caractéristiques d'une variable en cliquant sur

dans la barre de commande principale.

Revenir dans le sous-système représentant la chaîne d'information pour visualiser les connexions qui apparaissent sur le Chart. Une prévisualisation du digramme d'état est visible par défaut dans le Mask du sous-système.



Figure 459 : chart avant connexion avec le système

Les connexions étant créées automatiquement, elles peuvent se positionner différemment sur le chart. Vérifiez que la consigne de cap sera bien reliée à cons_cap et que la mesure de cap sera bien reliée à mes_cap.

Si les connexions sont inversées, elles peuvent être modifiées dans la fenêtre **Symbols** Figure 460.
Symbols			4	×
8	2 🖉 🖉			
TYPE	NAME	VALUE	PORT	
J	cons_cap		2	
L	mes_cap		1	
8	cons_barre		1	
			2	

Figure 460 : inversions de la position des ports

Cette opération peut également être réalisée à l'aide du **Model Explorer** ou l'on peut également indiquer les bons numéros de port comme indiqué sur la Figure 461.

🗰 Model Explorer	-	
File Edit View Tools Add Help		
😼 🗀 🔏 🐴 🖓 📖 🌻 🗎	🖌 fx 🛯 🗣 🔜 🖀 🔲	
Model Hierarchy 🖉 🗠	Contents of:0_0/Simulink et StateFlow/Chart (only) Filter Contents Data mes_cap	
🗸 🎦 Simulink Root	Column View: Stateform Show Details 2 of 11 object/c) General Description	
Base Workspace	Name: mes.cap	
V pilote_hydraulique_Stateflow_0_0	Name Scope Port Resolve Signal DataType Size InitialVal	
Configurations	Big cons_cap Input 2 Inherit: Same as Simulink -1	
Model Workspace	Image: Cape Input 1 Inherit: Same as Simulink -1 Size: -1 variable size	
> 🔎 External Data	🛗 cons_barre Output 1 🗌 Inherit: Same as Simulink -1 Complexity: Off	
> 🔁 Fluids	Type: Inherit: Same as Simulink V	>>
> 🔁 Multibody	Lock data tune analyst Fiveri-Point tools	
> 🔄 Simscape Electrical Library		CT. Excellen
> 🔁 Simulink	Unit (e.g., m, m/s ^{-,} 2, N ⁺ m):	SI, English,
Simulink et StateFlow	innent	
> te Chart	Limit range	
	Minimum: Maximum:	
	Add to Watch Window	
	K Revert Help	Apply
< >>	Contents Search Results	

Figure 461 : modification des numéros de port d'un chart

Connecter le chart avec le schéma bloc Simulink conformément à la Figure 462.



Figure 462 : connexion du chart avec le schéma bloc Simulink

7. Simulation du diagramme d'états

Lancer la simulation et observer le suivi du cap en visualisant le scope donnant la consigne de cap et le cap effectif suivi par le bateau.



Figure 463 : résultat de la simulation

D. Architecture des machines à états

1. La hiérarchie des états

Le diagramme d'état précédemment établi comporte 3 états de même niveau. Aucune hiérarchie n'est présente entre ces états. Afin de pouvoir modéliser des comportements logiques plus complexe **Stateflow** utilise le concept de **super-état**.

Un super état est un état qui peut contenir d'autres états de niveaux hiérarchiques inférieurs appelés **sous-états**.

- Super-état : état parent contenant d'autres états
- **Sous-états** : état enfant contenu dans un état parent. Les sous-états ne peuvent être actifs uniquement si l'état parent est actif.

L'utilisation de super-états et de sous-états permet d'améliorer la lisibilité des modèles. Il n'y a pas de limitation dans le nombre de niveaux hiérarchiques que l'on peut construire dans un diagramme.



Figure 464 : représentation de super-état et de sous-état

2. Les priorités de test des transitions

Ces architectures plus complexes nécessitent des précisions sur les règles d'évolution et sur la priorité de test des différentes transitions.

On définit :

- les **transitions internes** : une transition qui ne traverse pas les bordures de l'état ou qui est englobée par l'état
- les transitions externes ou super transition : une transition qui traverse les frontières de l'état

Règles de test des transitions :

• les transitions externes sont testées avant les transitions internes

3. Etats parallèles

A chaque niveau de la hiérarchie, les sous-états peuvent être **parallèles** ou **exclusifs**.

- Si le sous-état est exclusif : un seul et unique sous-état ne peut être actif au sein d'un même niveau hiérarchique (à condition que le super-état parent soit actif)
- Si l'état est parallèle : tous les sous-états d'un même niveau hiérarchique sont actifs simultanément (à condition que le super-état parent soit actif). Il n'est pas possible de définir une transition vers ou depuis un état parallèle, puisque son activation sera conditionnée exclusivement par l'activation de son état parent.

Dans **Stateflow**, les états parallèles s'affichent avec des bords en pointillés.



Figure 465 : représentation de sous-états parallèles

E. Ajout de niveaux hiérarchique et d'états parallèles dans un diagramme d'état

Nous allons reprendre le diagramme d'état de la commande de cap du pilote hydraulique de bateau en ajoutant de nouvelles fonctionnalités.

Lorsque le pilote est en mode « correction_de_cap », un voyant rouge doit s'allumer pour signaler aux utilisateurs que le pilote automatique consomme de l'énergie.

L'allumage de ce voyant sera géré dans l'état « Gestion_voyant ». Une variable « voyant_rouge » sera à 1 lorsque le voyant sera allumé et à 0 lorsque le voyant sera éteint.

Créer les super-états « **Suivi_de_cap** », « **Gestion_voyant** » et « **Commande_de_cap** » en respectant la hiérarchie de la Figure 466. La procédure de création d'un super-état est la même que la procédure de création d'un état. Il faut être attentif à positionner les contours de l'état parent autour des états enfants. **StateFlow** prend en compte automatiquement la hiérarchie entre les états à partir de leur positionnement dans la fenêtre graphique. Si un conflit empêche **Stateflow** d'établir une hiérarchie sans ambiguïté entre les états, les états qui posent un problème apparaissent en rouge.

Commande_de_cap	
Suivi_de_cap Pause entry: cons_barre = 0	[abs(cons_cap - mes_cap) > 0.1] [abs(cons_cap - mes_cap) < 0.1] [abs(cons_cap - mes_cap) < 0.1] [abs(cons_cap - mes_cap) < 0.05] [abs(cons_cap - mes_cap) < 0.05]

Figure 466 : création de super-états

L'état « commande_de _cap » possède 2 états enfants :

- « Gestion_voyant »
- « Suivi_cap »

Ces états enfants doivent être des états **parallèles** puisque l'allumage du voyant doit se faire pendant que l'état « **Suivi_de_cap** » est actif. Le bon fonctionnement du système de commande impose que les deux états enfants soient actifs simultanément lorque l'état « **commande_de_cap** » est activé.

L'état « Suivi_de_cap » possède 3 états enfants :

- « Pause »
- « Temporisation «
- « Correction_de_cap »

Ces états enfants doivent être des états **exclusifs** puisque ces états doivent être activés successivement lorsque l'état parent « **Suivi_de_cap** » est activé. Un seul et unique de ces états enfants sera actif.

Pour indiquer à **Stateflow** que les états enfants de l'état parent « **Commande_de_cap** » seront des états parallèles, **cliquer** avec le bouton droit de la souris à l'intérieur de l'état « **Commande_de_cap** », mais à l'extérieur des états enfants (Figure 467). Dans le menu contextuel choisir **Decomposition/parallel**.



Figure 467 : créer des sous-états parallèles

Les états parallèles apparaissent maintenant en pointillés.

			1
Pause entry: cons_barre = 0	[abs(cons_cap - mes_cap) > 0.1] [abs(cons_cap - mes_cap) < 0.1] [abs(cons_cap - mes_cap) < 0.05]	0 2 (after(20,sec)) 2 Correction_de_cap during: cons_barre = (cons_cap - me	es_cap)

Figure 468 : représentation des états parallèles dans Stateflow

Il faut maintenant indiquer que la variable « voyant_rouge » sera à 1 lorsque l'état « **Correction de cap »** sera actif.

Pour cela **Stateflow** dispose de variables internes qui caractérisent l'activation des états.

La commande *in(nom_de_l_etat)* renvoie la valeur 1 quand l'état est actif et la valeur 0 quand l'état est inactif.

Le nom d'un état est constitué de toute l'arborescence hiérarchique des états parents.

Compléter l'état « Gestion_voyant » comme indiqué sur la Figure 469.

	Le mot clé during indique que cette action est réalisée à chaque pas de temps durant l'activation de l'état	
Gestion_voyant during voyant_rouge = ir	n(Suivi_de_cap.Correction_de_cap)	
	voyant_rouge prendra la valeur 1 quand l'état Correction_de_cap sera actif	

Figure 469 : utilisation de variables internes pour évaluer l'activation d'un état

En utilisant le Symbols Pan indiquer que la variable voyant_rouge est de type Output Data



Figure 470 : paramétrage de la variable voyant_rouge

Avant de lancer la simulation, il faut ajouter une « transition par défaut » sur l'état « **Commande_de_cap** » afin d'activer le diagramme au démarrage.



Figure 471 : chart complet de commande de cap avec super-états et états parallèles

Connecter la variable de sortie voyant_rouge à un scope pour la visualiser.



Figure 472 : connexion de la variable de sortie du chart à un scope

Si nécessaire le modèle complet est disponible dans le fichier *pilote_hydraulique_Stateflow_1.slx*.

Lancer la simulation et observer l'évolution de l'état de la variable voyant rouge dans le scope.



Figure 473 : visualisation de l'état d'activation du voyant

Il est possible d'ajouter un voyant de la bibliothèque Dashboard afin de visualiser l'activation de la correction de cap.

Le fichier pilote_hydraulique_Stateflow_1_dashboard.slx intègre cette fonctionnalité.



Figure 474 : ajout d'un voyant de la bibliothèque Dashboard dans le modèle

Lors de l'exécution du modèle, le voyant est vert si le pilote n'est pas en phase de correction et rouge lorsqu'il est en phase de correction.

F. Récapitulatif et complément des commandes utiles de Stateflow

Commandes utiles	
Fonctions	Commandes/syntaxe
Création d'un état :	
Nom % Commentaires mot clé: actions	<i>Les mots-clés :</i> en ou entry : l'action s'effectue uniquement à l'activation de l'état du ou during : l'action s'effectue en continu tant que l'état est activé ex ou exit : l'action s'effectue uniquement à la désactivation de l'état

Création d'une transition avec jonction:



Pour créer une transition avec chemin multiples, il est possible d'utiliser une jonction disponible dans le menu « Chart ».

Pour passer à l'Etat2, il faut que **[Condition1]** et **[Condition2]** soient vraies.

Pour passer à l'Etat3, il faut que [Condition1] et [Condition3] soient vraies

Le chemin numéroté **1** est évalué avant le chemin numéroté **2**.





L'Action de condition est une action qui s'exécute uniquement si le test de la condition est vrai, et même si le chemin de la transition n'aboutit pas à l'activation de l'Etat2 ou de l'état 3. Si [condition] est vrai alors {action de condition} s'exécute même si [Condition2] et [Condition3] sont faux.

Test sur les variables dans les transitions :	a et b	a&&b
	a ou b	a b
	complément de a	!a
	a égal b	a==b

Suivi de l'activation des états :

in(Etat) : cette variable interne est vrai (valeur 1) tant que l'état est actif.

Temporisation de 15 s dans une transition	[after (15,s)]	

Figure 475 : rappel des principales règles de syntaxe de Stateflow

Chapitre 8 : Prise en main de Multibody

I. Introduction à Multibody

Multibody est un outil de MATLAB permettant d'inclure dans un modèle multi-physique, la maquette CAO 3D du système en l'important depuis Solidworks (ou tout autre logiciel de CAO). Multibody possède également des fonctionnalités d'édition de pièces qui permettent de créer des pièces volumiques directement dans l'environnement MATLAB.

Multibody permet de visualiser les mouvements des solides durant la simulation, de tenir compte du comportement dynamique des solides en mouvements, de prendre en compte toutes les non-linéarités géométriques de manière implicite...

Les résultats issus du modèle multi-physique sont plus proches des résultats observés sur le système réel par la prise en compte de ces nouveaux paramètres sans avoir à écrire les équations caractérisant le comportement dynamique du système.

A. Analyse d'un modèle Multibody

Dans **Multibody** les systèmes sont représentés sous la forme d'un graphe de liaisons. Les solides sont donc représentés par des blocs reliés entre eux par des liaisons.

Ouvrir le fichier « *pilote_Multibody_0.slx* » (Figure 476)

Ce fichier a été importé directement depuis le logiciel Solidworks. Les procédures d'importation seront abordées ultérieurement.



Figure 476 : modèle Multibody de la partie mécanique du pilote hydraulique

Pour une meilleure lisibilité du modèle, il est conseillé de créer des masques sur les blocs « solide » afin de visualiser directement les formes.

Ouvrir le fichier « pilote_Multibody_0_mask.slx »



Figure 477 : modèle Multibody du pilote avec masques sur les solides

Lancer la simulation du modèle.

La fenêtre **Mechanics Explorers** apparaît et la maquette 3D de la partie mécanique du pilote hydraulique est visualisable (.



Figure 478 : fenêtre Mechanics Explorers de la partie mécanique du pilote hydraulique



La barre d'outils de Multibody permet différentes options d'affichage (Figure 479).

Figure 479 : options de la barre de visualisation de Multibody

A noter qu'aucun mouvement n'est observable pour le moement dans la mesure où aucune liaison n'est pilotée.

B. Paramétrage de la gravité



Retourner dans la fenêtre du modèle et ouvrir le bloc Mechanism Configuration

(Figure 480).

<u>ी</u> Mechanism C	Configuration :	_		×	
Description					
Description Sets mechanical and simulation parameters that apply to an entire machine, the target machine to which the block is connected. In the Properties section below, you can specify uniform gravity for the entire mechanism and also set the linearization delta. The linearization delta specifies the perturbation value that is used to compute numerical partial derivatives for linearization. Port C is frame node that you connect to the target machine by a connection line at any frame node of the machine. Properties					
🖃 Uniform Gra	Constant		-	~	
Gravity	[0 0 -9.81]		m/s^2	~	
Linearization	0.001				
OK Cancel Help Apply					

Figure 480 : paramétrage de la pesanteur dans Multibody

Par défaut la pesanteur a été placée sur l'axe z (valeur -9.81 m/s²), ce qui correspond à la réalité de fonctionnement du système. Le repère utilisé pour orienter la pesanteur est le repère fixe « World ».

Pour voir l'influence de la pesanteur sur la simulation, nous allons placer la pesanteur sur l'axe x (Figure 481).

<u>ी</u> Mechanism (Configuration	:	-		×
Description					
Sets mechanical and simulation parameters that apply to an entire machine, the target machine to which the block is connected. In the Properties section below, you can specify uniform gravity for the entire mechanism and also set the linearization delta. The linearization delta specifies the perturbation value that is used to compute numerical partial derivatives for linearization. Port C is frame node that you connect to the target machine by a connection line at any frame node of the machine.					
Properties					
🗆 Uniform Gra	Constant				~
Gravity	[-9.81 0 0]		m/s	^2	~
Linearization	0.001				
		OK Ca	ncel	Help	Apply

Figure 481 : modification de l'action de la pesanteur

Lancer à nouveau la simulation et observer le mouvement du système. On observe un mouvement de rentrée et sortie de la tige du vérin qui correspond à l'action imposée par la pesanteur. Aucun frottement n'étant modélisé, le mouvement est périodique.

Remettre la pesanteur sur l'axe z pour retrouver les conditions réelles de fonctionnement.

II. Intégration d'un modèle Multibody dans un modèle multi-physique

Nous allons apprendre dans cette partie à intégrer un modèle Multibody dans un modèle multi-physique. Pour cela nous utiliserons le modèle du pilote hydraulique de bateau.

Ouvrir le fichier *pilote_hydraulique_Multibody_start.slx*



Figure 482 : modèle du pilote hydraulique avec modèle Multibody à intégrer

Le sous-système **Multibody** contient le modèle de la partie mécanique du pilote, mais qui n'est pas connecté avec le reste du modèle.



Figure 483 : visualisation du système Multibody sans connexion avec le reste du modèle

Sur la Figure 482, il est possible de visualiser les connexions que le sous-système **Multibody** doit assurer avec le reste du modèle.

- A l'entrée, le modèle reçoit une connexion physique qui représente la tige du vérin. Il s'agit d'un port de type **Physical Conserving Port** (Simscape)
- A la sortie le sous-système **Multibody** doit renvoyer l'angle de la barre afin d'agir sur le bateau et de donner l'angle de barre comme information d'entrée à la chaîne d'information. Cette information prend la forme d'un signal Simulink.

A. Connexions du modèle

Afin de connecter le modèle, il faut créer à l'intérieur du sous-système **Multibody** deux ports :

- Un port Simscape pour se connecter à la tige du vérin
- Un port Simulink pour renvoyer l'ange de rotation de la barre

Désignation	Représentation	Bibliothèque
Port de connexions Simscape	(1) Connection Port	Simscape/Utilities
Port de connexions Simulink	X1 Out1	Simulink/Ports & Subsystems

Placer les deux ports et les renommer conformément à la Figure 484.



Figure 484 : placement des ports de connexion dans le modèle

Des ports apparaissent sur le sous-système Multibody. **Connecter** ces ports avec le modèle.



Figure 485 : connexions des ports du sous-système avec le modèle

B. Interfaçage entre Simscape et Multibody

Dans cet exemple la connexion « tige vérin » provenant de **Simscape** doit servir à actionner la liaison entre le corps du vérin et la tige. Dans **Multibody**, les liaisons sont actionnées par des efforts de type « Force » ou de type « Torque » (moment). Il faut donc actionner la liaison pivot-glissant entre le corps et la tige du vérin avec une force qu'il faudra prélever sur la tige du vérin à l'aide d'un capteur de force. Cette force va agir sur la partie mécanique du système. Il résultera un mouvement de tout le mécanisme qui prendra en compte la dynamique de la partie mécanique ce qui imposera une vitesse de sortie à la tige du vérin. Afin de ne pas casser la boucle physique et de permettre à la tige du vérin modélisée dans **Simscape** de se déplacer à la même vitesse que la tige du vérin du modèle **Multibody**, il est impératif de réinjecter cette vitesse dans le modèle **Simscape**. Cela garantira durant tout le fonctionnement une équivalence de comportement entre **Simscape** et **Multibody**.



1. Interfaçage entre Simscape et Multibody pour la translation

Figure 486 : interface de translation entre Simscape et Multibody

2. Interfaçage entre Simscape et Multibody pour la rotation



Figure 487 : interfaçage de rotation entre Simscape et Multibody

Ouvrir le fichier « Simscape_Multibody_Interfaces.slx ».

Ce fichier contient les interfaces entre **Simscape** et **Multibody**. Il est également possible de mettre ces interfaces sous la forme de sous-systèmes afin de les réutiliser facilement.



Figure 488 : les interfaces entre Simscape et Multibody

Copier le bloc Simscape Multibody Interface en translation dans le modèle.



Figure 489 : ajout du bloc Simscape Multibody Interface

3. Ajout de ports sur une liaison

Il faut maintenant ajouter des ports de connexions sur le bloc liaison

- Un port afin de lui permettre d'être actionnée
- Un port afin de renvoyer la valeur de la vitesse

Double cliquer sur le bloc de la liaison pivot glissant pour faire apparaître la boite de dialogue du paramétrage d'une liaison et paramétrer la liaison conformément à la Figure 490 afin de créer un port pour actionner la liaison et un port pour relever la vitesse. Paramétrer également un coefficient de frottement visqueux (Damping Coefficient)de 200 N.s/m.

Cylindrical Joint : pivot glissant		-	- 🗆	×	
Description					
Represents a cylindrical joint between tw represented by one revolute primitive an rotation and prohibits relative translation the follower origin translates along the b	vo frames. This joint has one translational and d one prismatic primitive coincident along the n in the base xy-plane. The follower frame first ase z-axis.	one rotational degre e same axis. This joir rotates about the ba	e of freedor t allows on se z-axis an	m ly nd then	
In the expandable nodes under Propertie the primitives of this joint. After you app Ports B and F are frame ports that repres motion of the follower frame relative to	es, specify the state, actuation method, sensing by these settings, the block displays the corresp ent the base and follower frames, respectively. the base frame.	g capabilities, and in ponding physical sig The joint direction i	ternal mech nal ports. s defined by	nanics of y	
Properties					
Z Revolute Primitive (Rz)					
State Targets		Perm	at da sné	cifier u	ne raideur nour la liaison
Internal Mechanics		- rerm	et de spe	ciller ui	
Equilibrium Position	0	rad		~	
Spring Stiffness	0	m*N/rad		~	
Damping Coefficient	0	m*s*N/rad		~	
Limits		_			
Actuation		Permet de s	pécifier u	un frotte	ement visqueux pour la liaison
Sensing					
Z Prismatic Primitive (Pz)					
🗉 State Targets		Permet de fa	ire anna	raîtro ur	n port pour actionner la liaison
Internal Mechanics		r critice de la	пс арра	raiti e ui	
Equilibrium Position	A	m		~	
Spring Stiffness	0	N/m		~	
Damping Coefficient	200	s*N/m		~	
Limits					
 Actuation 					
Force	Provided by Input			~	
Motion	Automatically Computed			~	
Sensing					
Position		Permet de	e faire an	paraître	e des ports pour relever la
Velocity			stiion Is	vitocco	oul'accélération
Acceleration		pt pt	5111011, 10	a vitesse	
Actuator Force					
Lower-Limit Force					
Upper-Limit Force					
Mode Configuration					
Composite Force/Torque Sensing					
		OK Ca	ncel Help	Apply	

Figure 490 : paramétrage des ports d'une liaison

Des ports apparaissent sur le bloc de la liaison. **Connecter** l'interface conformément à la Figure 491.



Figure 491 : connexion entre Simscape et Multibody

L'interface entre Simscape et Multibody étant réalisée, nous allons maintenant ajouter sur la liaison « pivot2 » entre le « Bâti » et le « Bras de mèche ».

- Un port pour imposer l'effort résistant qui agit sur la barre du bateau.
- Un port pour prélever l'angle de rotation de la barre du bateau.

Pour cela **double cliquer** sur la liaison « pivot2 » et ajouter un port pour imposer une action mécanique sur la liaison comme représenté sur la Figure 492.

<i></i>				~~~
Revolute Joint : pivot2		-	Ш	×
Description				
Represents a revolute joint acting betwe primitive. The joint constrains the origin coincident, while the follower x-axis and In the expandable nodes under Properti	en two frames. This joint has one rotational degree of freedom represent s of the two frames to be coincident and the z-axes of the base and follo J y-axis can rotate around the z-axis. es, specify the state, actuation method, sensing capabilities, and internal	ed by on wer fram mechani	e revolut es to be ics of the	e
primitives of this joint. After you apply t	hese settings, the block displays the corresponding physical signal ports.			
Ports B and F are frame ports that repres follower frame relative to the base frame	ent the base and follower frames, respectively. The joint direction is define.	ned by m	otion of	the
= 7 Develute Drimitive (De)				
Z Revolute Primitive (Rz)				
State largets				
Internal Mechanics				
Limits				
	Descrided hyderest			
Matian	Provided by Input			
	Automatically Computed			~
🗆 Sensing				
Valasiti				
Acceleration				
Actuator Torque				
Lower-Limit Torque				
D Mode Configuration				
Composite Force/Tergue Service				
- Composite Porce/Torque Sensing				
	ок	Cancel	Help	Apply

Figure 492 : ajout d'un port pour imposer un couple à la liaison

Les deux nouveaux ports apparaissent maintenant sur la liaison. Ajouter un bloc **PS-S** pour convertir le signal d'angle de rotation de la barre en signal **Simulink**. Choisir les degrés comme unité d'angle.

Block Parameters: PS-Sir	nulink Converter				
PS-Simulink Converter					
Converts the input Physi signal.	Converts the input Physical Signal to a unitless Simulink output signal.				
The unit expression in 'C be commensurate with t the conversion from the output signal. 'Apply affine conversion'	The unit expression in 'Output signal unit' parameter must match or be commensurate with the unit of the Physical Signal and determines the conversion from the Physical Signal to the unitless Simulink output signal.				
offset (such as temperat	ture units).				
Parameters					
Output signal unit:	deg 🔹				
Apply affine conversion					
ОК	Cancel Help Apply				

Figure 493 : conversion du signal physique en signal Simulink

Vous devez obtenir la configuration de la Figure 494.



Figure 494 : ajout de port sur une liaison dans Multibody

En début de simulation l'angle de rotation de la barre possède une valeur arbitraire fixée lors de l'importation du modèle depuis Solidworks. Il est possible de connaître la valeur initiale (offset) de la barre.

Pour cela double-cliquer sur la liaison « pivot2 » et développer State Target/Specify Position Target.

🔮 Revolute Joint : pivot2		_		×		
Description						
Represents a revolute joint acting between two frames. This joint has one rotational degree of freedom represented by one revolute primitive. The joint constrains the origins of the two frames to be coincident and the z-axes of the base and follower frames to be coincident, while the follower x-axis and y-axis can rotate around the z-axis.						
In the expandable nodes under Properties, spe primitives of this joint. After you apply these s	cify the state, actuation method, sensing capabilities, and ettings, the block displays the corresponding physical sign	internal mechanics o al ports.	of the			
Ports B and F are frame ports that represent th follower frame relative to the base frame.	e base and follower frames, respectively. The joint directio	n is defined by motio	on of the			
Properties						
Z Revolute Primitive (Rz)						
State Targets						
Specify Position Target						
Priority	Low (approximate)			~		
Value	91.342079820133691	deg		~		
Specify Velocity Target						
Internal Mechanics						
⊕ Limits						
Actuation						
Sensing						
Mode Configuration						
Composite Force/Torque Sensing						
		OK Cancel	Help	Apply		

Figure 495 : visualisation de l'offset de l'angle de barre

En début de simulation l'angle de la barre doit être nul, il faut donc retrancher cette offset à la valeur de l'angle renvoyé par la liaison. Pour cela **compléter** le modèle conformément à la Figure 496 en ajoutant

un bloc **constant** que vous compléterez en faisant un copier/coller de la valeur de l'offset de l'angle de barre et un soustracteur pour retrancher l'offset au signal.



Figure 496 : compensation de l'offset de l'angle de la barre

4. Modélisation d'un effort extérieur variable

Le moment qui s'oppose à la rotation de la barre durant le mouvement varie en fonction de l'angle de rotation de la barre. Cette variation étant non linéaire, nous utiliserons un bloc **1D Lookup Table** pour le modéliser. Le **1D Lookup Table** représente un gain qui pourra varier en fonction de la valeur de l'entrée. Ce bloc est très pratique pour modéliser des lois entrée sortie non linéaires de types géométrique ou dynamique.

Désignation	Représentation	Bibliothèque
1D Lookup Table	1-D T(u) 1-D Lookup Table	Simulink/ Lookup Tables

Ouvrir le fichier *pilote_Multibody_fin.slx* et observer l'ajout de l'effort non linéaire sur la liaison. L'angle de la barre est prélevé et le modèle impose à la liaison « pivot2 » un effort qui varie en fonction de l'angle de la barre.



Figure 497 : ajout d'un effort sur la liaison, utilisation d'une Lookup Table

Double cliquer sur le bloc **1D Look-up Table** pour explorer le paramétrage de ce type de bloc.

指 Block Parameters: 1-D Look	up Table1	×			
Lookup Table (n-D)	Lookup Table (n-D)				
Perform n-dimensional inter function in N variables. Brea to the top (or left) input por	olated table lookup including index searches. The table is a sampled representation of a point sets relate the input values to positions in the table. The first dimension correspondent	nds			
Table and Breakpoints A	gorithm Data Types				
Number of table dimensions:	1 ~				
Data specification:	Table and breakpoints				
Table data:	[0 2.5 5 10 30 60 100]				
Breakpoints specification:	Explicit values 👻				
Breakpoints 1:	[0 5 10 15 20 25 30]				
Edit table and breakpoints					
		~			
<		>			
	OK Cancel Help App	у			

Figure 498 : paramétrage d'une Lookup Table

Pour paramétrer la table, il suffit d'entrée des points appartenant à la courbe qui caractérise la loi entrée sortie du bloc. Le logiciel effectue automatiquement une interpolation pour associer à chaque valeur de l'entrée, la valeur de la sortie correspondante.

Les coordonnées des points sont saisies dans les champs **Breakpoints 1** (abscisses) et **Table data** (ordonnées).

Il est possible de visualiser la loi ainsi saisie en cliquant sur **Edit table breakpoints**.

Lookup Table Editor: pilote_M	lultibody_fin/Multi	body/1-D Loo	kup Table1		_	×
≊ C °≦ ×≣ h A M ()						
Models:	Viewing "n-D Lo	okup Table" bl	ock data [T(:)]:			
) 🚹 pilote_Multibody 🗸 🖆	Breakpoints	Column	(1)			
Table blocks:	Row					
- 🔄 Multibody	(1)	0	0			
🖉 🖉 1-D Lookup Table1	(2)	5	2.5			
	(3)	10	5			
	(4)	15	10			
	(5)	20	30			
	(6)	25	60			
	(7)		100			
	Data Type: Row:	double 🗸 (Column: double	e 🗸 Table: double 🗸	•	
	Dimension Select		- 1			
	Ulmension	size	7			
	Showin	g	All ~			
	Transpose dis	play				

Figure 499 : visualisation d'une Lookup Table

Il est possible de visualiser la courbe représentant la loi entrée sortie en cliquant sur **Linear Plot** ^[].



Figure 500 : visualisation de la courbe représentative de la loi entrée sortie d'une Lookup Table

Valider le paramétrage de la lookup Table.

C. Résultat de la simulation

Lancer la simulation et observer la réponse en cap du bateau qui prend en compte le modèle **Multibody** ajouté.



Figure 501 : résultat de la simulation du cap suivi par le bateau

III. Importation d'un modèle SolidWorks dans Multibody

A. Les principes

Pour exploiter dans l'environnement **MATLAB** – Simulink une maquette CAO 3D issu de Solidworks, il faut convertir cette maquette en un fichier portant l'extension **xml**. Pour cela il faut d'abord installer dans SolidWorks un addon « **Multibody Link** » permettant de réaliser la conversion d'un fichier assemblage de Solidworks en fichier xml. Le fichier xml est ensuite converti par **MATLAB** en fichier exploitable par Multibody.

Lors de la conversion chaque fichier « pièce » de Solidworks sera converti en fichier STL. Ces fichiers contiennent les formes des pièces et les caractéristiques cinétiques. La présence de ces fichiers dans le « path » de **MATLAB** est indispensable afin de visualiser les animations dans la fenêtre **Mechanics Explorer.**

B. Installation de « Multibody Link »

Pour effecteur le téléchargement de **Multibody Link**, vous devez posséder un compte Mathworks . Si vous ne possédez pas de compte Mathworks, vous devez donc le créer.

Solidworks et MATLAB doivent être installés sur votre ordinateur.

Connectez-vous avec votre login et votre mot de passe. **Rendez-vous** à l'adresse suivante :

https://www.mathworks.com/campaigns/offers/download smlink.html

Vous êtes alors sur la page permettant de télécharger les fichiers d'installation nécessaires (Figure 502)

Download and installation instructions

Expand all

Simscape Multibody Link 7.2 – Release 2020b (Simscape Multibody 7.2)

Simscape Multibody 7.2				
Win64 (PC) Platform	smlink.r2020b.win64.zip install_addon.m			
UNIX (64-bit Linux)	smlink.r2020b.glnxa64.zip install_addon.m			
Mac OS X (64-bit Intel)	smlink.r2020b.maci64.zip install_addon.m			

- > Simscape Multibody Link 7.1 Release 2020a (Simscape Multibody 7.1)
- > Simscape Multibody Link 7.0 Release 2019b (Simscape Multibody 7.0)
- > Simscape Multibody Link 6.1 Release 2019a (Simscape Multibody 6.1)
- > Simscape Multibody Link 6.0 Release 2018b (Simscape Multibody 6.0)

Figure 502 : choix de la version de Multibody Link

Il faut choisir la version de **Multibody Link** correspondant à votre version de **MATLAB** et télécharger les deux fichiers correspondant à votre système d'exploitation et les placer impérativement dans un dossier du « path » de **MATLAB**.

- Fichier script MATLAB : install_addon.m
- Fichier archive qui contient les fichiers à installer : smlink.r2020b.win64.zip

Dans la fenêtre de commande de MATLAB taper la commande qui lance l'installation de Multibody Link :

>> install_addon('smlink.r2020b.win64.zip')

Installation of smlink complete.

.

To view documentation, type "doc smlink"

Attention, le nom du fichier archive contenu entre les parenthèses de la commande **install_addon** doit être exactement celui que vous avez téléchargé et dépend donc de votre version de MATLAB et de votre système d'exploitation.

Il faut ensuite enregistrer MATLAB comme serveur Automation en tapant la commande suivante : >> regmatlabserver

Une fois l'installation terminée, taper la commande suivante afin de créer le lien entre MATLAB et Solidworks.

>> smlink_linksw

La procédure d'installation est terminée.

C. Conversion d'un fichier assemblage de Solidworks en fichier xml

Lancer Solidworks.

La fenêtre d'accueil du logiciel s'ouvre



Figure 503 : fenêtre d'accueil de Solidworks

La fenêtre d'accueil du logiciel s'ouvre, dérouler le menu **options** puis sélectionner **compléments** pour ouvrir la fenêtre de définition des compléments du logiciel.

SolidWORKS Fichier Affichage	Outils ? 🖈 🏠 🖿 🔭 🖛 🗸	a · ŋ · b · 🛛 🗉 🕸 ·	Rechercher dans l'aide de SOLIDV D · 2 · - + C ×
	Applications SOLIDWORKS	→	
	Produits Xpress	→	
	SOLIDWORKS CAM	→	
	Design Checker	→	
	Comparer	→	
	Macro	→	
	Compléments		
	Enregistrer/Restaurer les paramètres		
	Personnaliser		
	Options		

Figure 504 : ouverture de la fenêtre des compléments de Solidworks

Cocher les deux cases à gauche et à droite de **Simscape Multibody Link** afin d'activer **Simscape Multibody Link** à chaque démarrage de Solidworks (Figure 505).

Cliquez sur **OK**.

Comple	éments			×
Comple	éments actifs	Démarrag e	Durée du dernier chargeme nt	^
	Autotrace			
	SOLIDWORKS CAM 2019	\checkmark	Зs	
	SOLIDWORKS Composer	\checkmark	< 1s	
	SOLIDWORKS Electrical			
	SOLIDWORKS Flow Simulation 2019	\checkmark	1s	
	SOLIDWORKS Forum 2019	\checkmark	< 1s	
	SOLIDWORKS PCB 2019			
	SOLIDWORKS Plastics			
	SOLIDWORKS Visualize			
🗆 Aut	res compléments			
	3DCloudByMe Plug-in			
	Meca3d v18.0	\checkmark	< 1s	
	Simscape Multibody Link	\checkmark	< 1s	
	SOLIDWORKS 3DEXPERIENCE SmartLink			
	SOLIDWORKS Social 2019			
	SOLIDWORKS XPS Driver 2019			
				~
,	OK Annule	r		

Figure 505 : sélection des compléments de Solidworks

Quitter, puis relancer Solidworks.

Avec Solidworks, **Ouvrir** le fichier *Assemblage pilote hydraulique.sldasm* dans le dossier « Modèle CAO 3D du pilote/Partie Mécanique »



Figure 506 : ouverture du modèle Solidworks de la partie mécanique du pilote

Pour convertir l'assemblage au format **xml**, cliquer dans la barre de menu sur **Outils/Simscape Multibody Link/Export/Simscape Multibody**

Prendre soin de choisir comme dossier d'exportation un dossier qui appartient au « path » de MATLAB. Ici nous prendrons le dossier vide « conversion_pilote-xml » placé au même niveau que le dossier « Modèle CAO 3D du pilote ». Choisir comme nom de fichier *Assemblage_pilote_hydraulique*.

Après quelques secondes de travail, Solidworks va ouvrir et fermer tous les fichiers pièces qui constituent l'assemblage pour les convertir au format STL. Une fenêtre qui s'ouvre automatiquement indique la fin de la conversion.

Ouvrir le dossier « conversion_pilote_xml » avec l'explorateur Windows pour constater qu'il contient toutes les pièces du mécanisme au format **STL** ainsi qu'un fichier nommé **Assemblage_pilote_hydraulique.xml**.

Retourner dans MATLAB et taper dans la fenêtre de commande :

>>smimport('Assemblage_pilote_hydraulique.xml')

Cette commande va permettre de créer le fichier **Multibody** correspondant à l'assemblage et ouvre automatiquement un nouveau fichier contenant le modèle.



Figure 507 : modèle Multibody obtenu

Nous retrouvons le fichier sur lequel nous avons travaillé au début de ce chapitre.

Lancer la simulation, la fenêtre Mechanics Explorer s'ouvre et vous pouvez visualiser la maquette du pilote dans MATLAB. Il se peut que la pesanteur ne soit pas par défaut sur le bon axe.



Figure 508 : fenêtre Mechanics Explorer

Chapitre 9 : L'identification d'un modèle

I. La modélisation black-box, l'identification

A. Présentation de la méthode

Cette approche, consiste à associer un modèle mathématique à un comportement mesuré expérimentalement. Le système est considéré ici comme une boite noire avec une grandeur d'entrée et une grandeur de sortie. Le principe consiste à imposer une grandeur d'entrée connue au système et à relever la grandeur de sortie. Il suffit ensuite de trouver un modèle mathématique pour la fonction de transfert du système qui donnera la même relation entre l'entrée et la sortie. Cette méthode donne uniquement un modèle du comportement global du système sans se préoccuper de l'influence séparée des différents paramètres sur les performances.



Figure 509 : la modélisation black box

Cette méthode permet d'obtenir un modèle validé expérimentalement sans avoir à écrire les équations qui régissent le comportement du système. Le modèle doit être manipulé avec précaution dans le cas ou des phénomènes non-linéaires entre en compte dans le comportement réel du système. En pratique on réalise toujours plusieurs essais correspondant à des sollicitations différentes (grandeur d'entrée constante ou sinusoïdale avec une variation de la fréquence).

Mise à part quelques cas simples, l'identification de la fonction de transfert demande une ressource de calcul importante. L'outil logiciel va nous permettre d'automatiser cette tâche en proposant des modèles mathématiques pour des processus d'identification qui peuvent être complexes.

MATLAB propose une application appelée « System Identification » qui permet de mener à bien ce processus. Comme toutes les applications, elle est accessible à partie de l'onglet **APPS**.

B. Mise en œuvre de la méthode en utilisant la toolbox Identification

1. Analyse des données utilisées pour l'identification

Ouvrir et exécuter le script **donnees_identification.m.**

Ce script permet d'afficher les grandeurs d'entrée et de sortie d'un essai en boucle ouverte sur un axe linéaire. Une tension est imposée aux bornes du moteur de l'axe linéaire (grandeur d'entrée) et la vitesse linéaire de l'axe est relevée (grandeur de sortie). Les conditions dans lesquelles l'essai a été réalisé sont illustrées Figure 510.



Figure 510 : les conditions de l'essai

La Figure 511 permet de visualiser les signaux d'entrée et de sortie.



Figure 511 : visualisation des données utilisées pour l'identification

L'exécution du script a également crée dans le **Workspace** de MATLAB les variables suivantes :

Workspace			
Name 🔺	Value	Class	
Reponse_indicielle	251x17 double	double	
🖶 t	251x1 double	double	
💾 u	251x1 double	double	
H v	251x1 double	double	
🛨 x	251x1 double	double	

Figure 512 : visualisation dans le Workspace des données utilisées pour l'identification

- t : vecteur contenant la variable temps
- u : vecteur contenant la variable tension d'alimentation
- v : vecteur contenant la variable vitesse linéaire de l'axe
- x : vecteur contenant la variable position linéaire de l'axe

2. Ouverture et présentation de la toolbox « System Identification »

Pour lancer l'application « **System Identification** », sélectionner l'onglet APPS dans la fenêtre de commande de MATLAB et activer le menu déroulant des applications proposées par le logiciel. Vous pouvez alors voir apparaître toutes les applications disponibles. Sélectionner l'application **System Identification** (Figure 513).



Figure 513 : menu déroulant des applications de MATLAB

Une fenêtre s'ouvre et présente les fonctionnalités de l'application (Figure 514).



Figure 514 : description de la fenêtre de la toolbox identification

3. Importation des données

Il est possible d'importer tout type de données : temporelle, fréquentielle, Data object... Dans notre exemple les données issues de l'expérimentation sont des données temporelles.

Sélectionner Import data/Time domain data.



Figure 515 : importation de données temporelles

Cette commande permet d'ouvrir la fenêtre permettant de définir les données nécessaires pour l'identification. Compléter cette fenêtre avec les données de la Figure 516.


Figure 516 : sélection des données temporelles pour l'identification

Cliquer ensuite sur **Import** pour importer les données dans la fenêtre d'identification. Les données apparaissent alors dans la fenêtre de la « System Identification » toolbox comme le montre Figure 517. Cocher la case **Time plot** pour vérifier et visualiser la bonne importation des données (Figure 518).



Figure 517 : visualisation des données importées dans la fenêtre de la « System Identification » toolbox



Figure 518 : visualisation des données importées dans la « System Identification »toolbox

Il faut maintenant choisir la forme du modèle que l'on souhaite obtenir (fonction de transfert, représentation d'état, modèle non-linéaires...). Nous choisirons ici un modèle de type **fonction de transfert.**



Figure 519 : choix de la forme du modèle dans la « System Identification » toolbox

La fenêtre qui s'ouvre permet de choisir la forme souhaitée pour la fonction de transfert. Ici nous prendrons une fonction de transfert du second ordre. Choisir **2 pôles et pas de zéro** puis cliquer sur **Estimate**.

📣 Transfer Functio	ns						_	×
Model name: tf1 🤌	1							
Number of poles:	2							
Number of zeros:	0							
 Continuous-ti 	ime	⊖ Disc	rete-	time (Ts	= 0.01)	Feed	dthrough	
▶ I/O Delay								
Estimation Optio	ons							
	_				_			
	Es	timate		Close		Help		

Figure 520 : choix de la forme de la fonction de transfert dans la « System Identification » toolbox

La fenêtre Plant Identification Progress donne les détails sur le processus numérique d'identification

Estimatio	on Progress						
Initia Initia done.	lizing model p lizing using '	arameters iv' method					Â
Initia	lization compl	ete.					
Algorit	thm: Nonlinear	least squar	es with autor	natically ch	nosen line sear	ch method	
		Norm	of First-	-order]	Improvement (%)		 -
Iterat:	ion Cost	step	optimal:	ity Expec	cted Achieved	Bisections	
0	1.56786e+06	-	0.114	6.37e-05	-	-	 -
1	784699	1.1e+05	2.83e+03	6.37e-05	50	0	
2	34690.7	1.44e+04	1.04e+04	0.000127	95.6	0	
3	1154.43	1.26e+04	7.16e+03	0.00278	96.7	0	
4	1076.55	1.04e+05	1.37e+04	0.0181	6.75	1	
5	1059.99	9.03e+04	5.03e+03	0.0146	1.54	4	
6	1054.7	1.07e+05	1.93e+04	0.0137	0.499	5	
7	1053.1	2.16e+05	5.89e+04	0.0134	0.152	5	
8	999.157	116	3.18e+03	0.0133	5.12	0	
9	999.032	5.52	246	0.00872	0.0124	0	
10	999.032	0.344	13.3	0.00867	4.19e-05	0	 -
Estima: done.	ting parameter	covariance.					
							~
Result		n. Near (log	al) minimum,	(norm(g) <	tol)		
done.		n. Noon (log	al) minimum,	(norm(g) <	tol)		

Figure 521 : la fenêtre Plant Identification Progress de la « System Identification » toolbox

Le résultat de l'identification est consultable en cliquant avec le bouton droit de la souris sur la case **tf1** représentant le résultat de l'identification (Figure 522 et Figure 523). Cocher également la case **Model output** pour visualiser la superposition des données expérimentales et du modèle issu de l'identification (Figure 524).



Figure 522 : visualisation de résultats de l'identification

Data/model Info: tf1 Model name: tf1 Color: [0,0,1]	Fonction de transfert déterminée par identification
From input "ul" to output yl": 5.746e05 	function.
Diary and No	tes
<pre>% Import Ident % Transfer function estimation Options = tfestOptions; Options.Display = 'on'; Options.WeightingFilter = []; tfl = tfest(Ident, 2, 0, Options)</pre>	~
Show in LTI Viewer	
Present Export	Close Help

Figure 523 : fonction de transfert obtenue par identification



Figure 524 : superposition des données issues de l'expérimentation et du modèle issu de l'identification

Il est également possible à partir de la fenêtre Data/Model info (Figure 523) d'exporter la fonction de transfert déterminée par identification vers le Workspace de MATLAB en cliquant sur **Export**. La fonction de transfert tf1 apparaît alors dans le Workspace. Il est possible par exemple de tracer un diagramme de Bode de cette fonction de transfert en tapant la commande **bode (tf1)** dans la fenêtre de commande de MATLAB.



Figure 525 : diagramme de Bode de la fonction de transfert tf1

C. Utilisation de la méthode en utilisant les lignes de commandes

Pour identifier la réponse du système en utilisant les lignes de commande, il faut dans un premier temps créer un objet de type *iddata* contenant tous les éléments nécessaires à l'identification :

- Le vecteur contenant la réponse du système
- Le vecteur contenant l'entrée du système
- La période d'échantillonnage des données

Taper la commande suivante :

```
>> mydata=iddata(v,u,0.01)
mydata =
Time domain data set with 251 samples.
Sample time: 0.01 seconds
Outputs Unit (if specified)
y1
Inputs Unit (if specified)
u1
```

Il suffit maintenant de demander une identification par une fonction de transfert en spécifiant le nombre de pôle et le nombre de zéros de la fonction de transfert. On utilise pour cela la commande *tftest* en spécifiant 2 pôles et 0 zéro comme cela a été fait en utilisant la toolbox identification.

```
>>H = tfest(mydata,2,0)
```

H =

From input "u1" to output "y1": 5.746e05 -----s^2 + 3218 s + 2.119e04

Continuous-time identified transfer function.

Parameterization: Number of poles: 2 Number of zeros: 0 Number of free coefficients: 3 Use "tfdata", "getpvec", "getcov" for parameters and their uncertainties.

Status: Estimated using TFEST on time domain data "mydata". Fit to estimation data: 93.84% (simulation focus) FPE: 1040, MSE: 999

MATLAB renvoie alors la fonction de transfert du second ordre, identique à celle obtenue en utilisant l'application **System Identification**.

Commandes utiles pour l'identification	
Fonctions	Commandes
Création d'un objet <i>iddata</i> pour l'identifiaction	mydata=iddata(y,u,Ts) y : vecteur contenant les données de la sortie u : vecteur contenant les données de l'entrée Ts : période d'échantillonnage des données
Identification par une fonction de transfert de la réponse d'un système	H=tftest(mydata,np,nz) mydata : objet créé à l'aide de la fonction iddata np : nombre de pôles souhaités pour la fonction de transfert nz : nombre de zéros souhaités pour la fonction de transfert

Il existe également de nombreuses commandes permettant de préparer les données pour l'identification (filtrage, réglages de l'offset, remplacement des données manquantes...). Pour plus d'informations sur ces commandes, vous pouvez consulter l'aide de la **System Identification Toolbox**.

Chapitre 10 : Le contrôle commande avec MATLAB - Simulink

I. Introduction

MATLAB – Simulink possède de très puissants outils de contrôle commande des systèmes asservis. Le logiciel peut réaliser automatiquement le réglage d'un correcteur PID dans un environnement ou l'utilisateur agit sur les paramètres de réglages sans qu'il lui soit nécessaire d'avoir recours à des bases théoriques. Le logiciel propose également des modes experts ou toutes les méthodes classiques du contrôle commande sont exploitées et peuvent être mises en œuvre. Le principal avantage est de pouvoir disposer de tous les tracés fréquentiels et temporels nécessaires au processus de synthèse d'un correcteur. Ces tracés évoluent de manière dynamique en fonction de la structure du correcteur imposée par l'utilisateur.

II. Réglage automatique d'un PID

Pour illustrer la démarche de contrôle commande avec MATLAB et Simulink, nous allons étudier l'asservissement en vitesse d'un moteur à courant continu.

A. Modélisation

Les équations électrique, mécanique et de couplage d'un moteur à courant continu sont données

 $\begin{cases} u(t) = e(t) + Ri(t) + L\frac{di(t)}{dt} & u(t): \text{ tension de commande du moteur (V)} \\ e(t) = K_e \omega_m(t) & i(t): \text{ force contre électromotrice du moteur (V)} \\ J\frac{d\omega_m(t)}{dt} = C_m(t) - C_r(t) - f \omega_m(t) & C_m(t): \text{ couple exercé par le moteur (N.m)} \\ C_m(t) = K_m i(t) & \omega_m(t) & i(t): \text{ vitesse de rotation du moteur (rad/s)} \\ R : résistance de l'induit (\Omega) \\ L : inductance de l'induit (H) \\ J : inertie équivalente ramenée sur l'arbre moteur (kg.m²) \\ f : paramètre de frottemement visqueux (N.m.s) \\ K_t : constante de couple (N.m/A) \end{cases}$

K_e : coefficient de force contre électromotrice (V.s/rad)

Figure 526 : modélisation du moteur à courant continu

Après avoir appliqué la transformée de Laplace à ces équations on obtient le schéma bloc du moteur à courant continu. On asservit ce moteur en vitesse selon le schéma bloc de la Figure 527.



Figure 527 : asservissement en vitesse d'un moteur à courant continu

B. Ouverture du modèle

Ouvrir le fichier *moteur_cc_ass_vit.slx*. Certains blocs apparaissent entourés en rouge à l'ouverture du modèle (Figure 528). Cela signifie que certaines variables ne sont pas connues et que l'exécution du modèle entrainera un message d'erreur. Il faut donc donner des valeurs numériques à l'ensemble des variables du modèle.



Figure 528 : modélisation de l'asservissement en vitesse du moteur

Ouvrir le script *parametres_moteur_cc.m* et éxécuter-le.

Lancer la simulation et visualiser la réponse dans le scope.



Figure 529 : réponse temporelle du système non corrigé à un échelon unitaire

Pour effecteur le réglage automatique d'un PID, il faut **insérer** le bloc **PID controller** conformément à la Figure 530.



Figure 530 : ajout d'un bloc PID Controller dans l'asservissement

Double cliquer sur le bloc **PID Controller** pour ouvrir la fenêtre de paramétrage de la Figure 531.

Block Parameters: PID Controller PID 1dof (mask) (link) This block implements continuous- and discreated and signal tracking. You can tune the PID ga	e nms and includes advanced features such as anti-windup, external reset, 'Tune' button (requires Simulink Control Design).
Controller: PID	▼ Form: Parallel ▼
Time domain:	Discrete-time settings
Continuous-time	Sample time (-1 for inherited): -1 de transfert du
○ Discrete-time	correcteur
▼ Compensator formula	
P + Main Initialization Output Saturation Data Types State	$-I\frac{1}{s}+D\frac{N}{1+N\frac{1}{s}}$ Visualisation de la fonction de transfert du correcteur
Controller parameters	
Source: internal	•
Proportional (P): 1 Integral (I): 1 Derivative (D): 0	Valeurs des
Use filtered derivative	
Filter coefficient (N): 100	:
Automated tuning	
Select tuning method: Transfer Function Based (PID Tuner App)	▼ Tune
Enable zero-crossing detection Choix de la méthode de réglage	Réglage automatique du correcteur
	OK Cancel Help Apply

Figure 531 : fenêtre de paramétrage du PID Controller

1. Analyse de la réponse temporelle

Cliquer sur **Tune** pour ouvrir la fenêtre de réglage des performances du système.

MATLAB propose un réglage réalisant un compromis entre tous les critères de performance du système comme le montre la Figure 532. Par défaut la réponse indicielle de la fonction de transfert en boucle fermée (reference tracking) s'affiche pour le système corrigé et pour le système non corrigé.



Figure 532 : fenêtre de réglage des performances du système

Il est cependant possible d'agir manuellement sur les curseurs pour modifier les performances du système et le rendre conforme à cahier un cahier des charges précis. Afin de disposer de toutes les informations nécessaires pour faire les réglages l'outil propose différentes fonctionnalités.

Cliquer sur **Show Parameters** en haut à droite de la fenêtre pour visualiser les critères de performances et les valeurs des gains du correcteur avant et après la correction comme le montre la Figure 533.

Controller Parameters		
	Tuned	Block
Р	0	1
l	2.2488	1
D	0	0
N	100	100
Performance and Rob	ustness	Block
Performance and Robi Rise time	Tuned	Block
Performance and Rob Rise time Settling time	Ustness Tuned 0.719 seconds 1.85 seconds	Block 2.95 seconds 6.05 seconds
Performance and Rob Rise time Settling time Overshoot	Tuned 0.719 seconds 1.85 seconds 0.466 %	Block 2.95 seconds 6.05 seconds 0 %
Performance and Rob Rise time Settling time Overshoot Peak	Ustness Tuned 0.719 seconds 1.85 seconds 0.466 % 1	Block 2.95 seconds 6.05 seconds 0 % 0.999
Performance and Rob Rise time Settling time Overshoot Peak Gain margin	Ustness Tuned 0.719 seconds 1.85 seconds 0.466 % 1 7.32 dB @ 11.8 rad/s	Block 2.95 seconds 6.05 seconds 0 % 0.999 Inf dB @ Inf rad/s
Performance and Rob Rise time Settling time Overshoot Peak Gain margin Phase margin	Tuned 0.719 seconds 1.85 seconds 0.466 % 1 7.32 dB @ 11.8 rad/s 83.8 deg @ 2.54 rad/s	Block 2.95 seconds 6.05 seconds 0 % 0.999 Inf dB @ Inf rad/s 33.7 deg @ 16.1 rad/s

Figure 533 : les critères de performances du système et les valeurs de gain du correcteur

Pour obtenir le **diagramme de Bode la fonction de transfert en boucle ouverte**, sélectionner le menu **Add Plots/Bode/Open Loop** (Figure 534).



Figure 534 : ajout du diagramme de Bode de la fonction de transfert en boucle ouverte



Figure 535 : visualisation du diagramme de Bode de la fonction de transfert en boucle ouverte pour le réglage du PID

Les diagrammes de Bode du système corrigé et non- corrigé apparaissent sur une seconde fenêtre (Figure 535).

Pour régler les options des critères de performances, **cliquer droit** dans la fenêtre graphique de la réponse temporelle puis sélectionner **Properties**, puis onglet **Options**

📣 Property Editor: Step Plot: Reference tracking 🛛 —	×
Labels Limits Units Style Options	
Response Characteristics Show settling time within 5 % Show rise time from 10 to 90 %	
Confidence Region for Identified Models Number of standard deviations for display: 1.000	
Close	Help

Figure 536 : réglage des options des critères de performances

Indiquer que le temps de réponse est évalué sur le critère du temps de réponse à 5% en indiquant **5**% dans le champ **Show settling time within**. Le critère **rise time** concerne le temps de montée qui, ici, est évalué entre 10% et 90% de la valeur finale atteinte. Cliquer sur **Close**.

Il est également possible de visualiser graphiquement tous les critères de performances.

Cliquer droit dans la fenêtre graphique de la réponse temporelle puis sélectionner **Characteristics/Peak Response** (dépassement), puis **Characteristics/Settling Time** (temps de réponse à 5%), puis **Characteristics/Rise Time** (temps de montée) puis **Characteristics/Steady State** (précision).

Cliquer droit dans la fenêtre graphique du diagramme de Bode, puis sélectionner **Characteristics/All Stability Margins** (marges de gain et marge de phase).

Il est maintenant possible de visualiser les critères de performances dans la fenêtre graphique (Figure 537).



Figure 537 : visualisation des critères de performances

A ce stade, il est possible d'agir sur les curseurs de réglage et de voir évoluer de manière interactive les courbes et les paramètres.

Une fois le réglage effectué, **Cocher** la case **Update Block** pour mettre à jour automatiquement le bloc **PID Controller** de **Simulink**. Seule la réponse corrigée apparait alors dans la fenêtre graphique.



Figure 538 : réponse corrigée

2. Importation dans Simulink

Une fois que la réponse du système et bien réglée, cliquer sur OK pour retourner dans le bloc de paramétrage du PID. Nous pouvons constater que les gains du correcteur ont été actualisés avec les valeurs correspondant au réglage qui vient d'être effectué.

Block Parameters: PID Controller	×				
PID 1dof (mask) (link)					
This block implements continuous- and discrete-time PID control algorithms and includes advanced features such as anti-windup, external reset, and signal tracking. You can tune the PID gains automatically using the 'Tune' button (requires Simulink Control Design).					
Controller: PID 🗸	Form: Parallel 🔹				
Time domain:	Discrete-time settings				
Continuous-time	Sample time (-1 for inhorited):				
O Discrete-time	Sample time (-1 for innented).				
▼ Compensator formula					
$P+I\frac{1}{2}$	$+ D \frac{N}{1+N\frac{1}{s}}$				
Main Initialization Output Saturation Data Types State At	tributes				
Controller parameters					
Source: internal	•				
Proportional (P): 0					
Integral (I): 2.24882422091649	:				
Derivative (D): 0					
☑ Use filtered derivative					
Filter coefficient (N): 100	1				
Automated tuning					
Select tuning method: Transfer Function Based (PID Tuner Ann)	▼ Tune				
Enable zero-crossing detection					
	OK Cancel Help Apply				

Figure 539 : actualisation automatique des paramètres du PID

Cliquer sur **OK**.

Lancer la simulation et visualiser la réponse corrigée dans le scope.



Ivan LIEBGOTT - Modélisation et simulation des systèmes multi-physiques avec MATLAB/Simulink 2020b

III. Réglage manuel d'un PID avec l'outil « Control System Designer »

Il est également possible de régler un PID tout en visualisant l'influence des réglages sur les réponses temporelles et fréquentielles de la fonction de transfert en boucle ouverte et de la fonction de transfert en boucle fermée. Pour cela, nous utiliserons l'application « Control System Designer ».

A. Ouverture du modèle

Ouvrir le fichier *moteur_cc_ass_vit_control_sys_des_PID.slx*.

Ce fichier contient le modèle du moteur à courant continu asservi en vitesse avec un PID qui n'a pas encore été réglé (Gain proportionnel à 1 et les autres gains sont nuls).



Figure 541 : modèle du moteur à courant continu asservi en vitesse avec perturbation

La consigne imposée est de 100 rad/s et une perturbation de couple est appliquée au système à l'instant t=3 s.

Les critères de performances sont les suivants :

- Écart statique nul
- Temps de montée (de 0 à 90%) de 0.8 s
- Temps de réponse à 5% de 1 s
- Dépassement maxi de 10%
- Marge de gain : 20 dB
- Marge de phase 45°
- Rejet des perturbations en régime permanent

Lancer la simulation et observer la réponse du système dans le scope.



Figure 542 : réponse temporelle du système non corrigé

Nous pouvons constater que la réponse du système n'est pas satisfaisante :

Le système n'est pas précis, mal amorti et ne rejette pas correctement la perturbation.

B. Réglage du PID

1. Lancement de Control System Designer

A partir de l'onglet APPS de la fenêtre de commande de MATLAB, sélectionner l'application « Control System Designer » (Figure 543).



Figure 543 : lancement de l'application « Control System Designer »

📣 Cont	trol Systen	n Designer - m	oteur_cc_ass_vit	_control_sys_des_PID	book						•••	_	×
CON	NTROL SYST	ΈM	VIEW								te të	9 G	? 🖲
Open		[문말] Edit	Multimodel	Tuning	New	Store	Retrieve	Compare) Update	Preferences			
Session	Session	Architecture	Configuration	Methods 🕶	Plot 🕶		•	-	Blocks	DEFERENCES			_
Data Bro	LE wser	ARCHI		TUNING METHODS	ANALYSIS		DESIGN	5	SIMULINK	PREFERENCES			-
▼ Contr	ollers and	Fixed Blocks		Edit Architecture -	Simulink (Configur	ation			x			
				Use this dialog to	edit the bl	ocks and	l signals ir	the current	t architectu	ire.			
				Blocks Signals	Linearizat	tion Opt	ions						
				Block Name			Val	ue					
								Add	d Blocks				
▼ Desig	ns									-			
-													
▼ Kespo	nses												
							OK	Cancel	Help	_			
Previe	W												
													1
				_									

Figure 544 : Control System Designer

Afin de sélectionner le bloc à régler cliquer sur **Add Blocks** et choisir PID Controller comme bloc à régler, puis cliquer sur **OK**. (Figure 545).

Page Select Blocks to Tune		×
moteur_cc_ass_vit_control_sys	elect blocks in th	e table below to tune:
	Tune?	Block Name
		PID Controller1
		Gain2
		Gain1
		Transfer Fcn1
		Transfer Fcn
< >		Highlight Selected Block
		OK Cancel

Figure 545 : choix du bloc à régler

Dans la fenêtre **Edit Architecture** (Figure 546) qui apparaît, cliquer sur OK pour valider votre choix.

Edit Architecture - Simulink Configuration X						
Use this dialog to edit the blocks and signals in the current architecture.						
Blocks	Signals	Linearization Op	tions			
Block N	lame		Value			
des_F	PID_book	/PID Controller1	<1x1 zpk>	đ	k 🎭 🗙	
				Ad	d Blocks	
			ОК	Cancel	Help	

Figure 546 : fenêtre Edit Architecture

2. Diagrammes de Bode et de Black de la FTBO

Nous allons maintenant choisir la méthode de réglage. Dans la fenêtre du Control System Designer, sélectionner **Tuning Method/Bode Editor**.

La fenêtre **Select Response to Edit** apparaît. La fonction de transfert en boucle ouverte (FTBO) est automatiquement détectée. Les points de linéarisation sont indiqués (Figure 547).

Select Response to Edit - Bode	×
Select Response to Edit: LoopTransfer_moteur_cc_a	ss_vit_control_sys_des_PID_book_PID_Controller1 💌
Open-Loop Transfer Function Name: LoopTransfer_moteur_cc_ass_vit_control_sy Locations: moteur_cc_ass_vit_control_sys_des_PID_book/PI	/s_des_PID_book_PID_Controller1
	Plot Cancel Help

Figure 547 : choix des points de linéarisation pour la FTBO

Nous verrons plus tard comment choisir nous-même les points de linéarisation.

Cliquer sur **Plot** pour tracer le diagramme de Bode de la FTBO. Pour ajouter un quadrillage, cliquer avec le bouton droit dans la fenêtre du diagramme de Bode et choisir **Grid**.



Figure 548 : diagramme de Bode de la FTBO

Il est également possible d'ajouter d'autres types de diagramme pour la FTBO. Cliquer sur **New Plot/New Nichols** puis cliquer sur **Plot** pour faire apparaître un diagramme de Black.



Figure 549 : diagramme de Black de la FTBO

Afin de visualiser les deux graphiques dans la même fenêtre, cliquer sur l'onglet **View** puis **Custom** pour choisir la configuration des fenêtres graphiques (Figure 550).

📣 Cor	ntrol System Desig	ner - m	noteur_cc_as	s_vit_con	rol_sys_des_PID_book - LoopTransfer_moteur_cc_ass_vit_control_sys_des_PID_book_PID_Control – 🛛	×
CONTROL SYSTEM			VIEW		- El 2 単語 つ C - El 2 単語 つ C - El 2	\odot
	Left/Right	-	■ Tabs	Position N	ie	
Single	Custom V	Float	Alpha	betize		-
Data Bro				+1	LoopTransfer_moteur_cc_ass_vit_control_sys_des_PID_book_PID_Controller1: nichols	
moteur_			es_PID_bo	actour	Nichols Chart	ak/DI
	Cancel			noteur_	20 ass vit_control_sys_des_PID_book/PID_Controller1_10: moteur_cc_ass_vit_control_sys_des_PID_book	эклен

Figure 550 : choix de la configuration des fenêtres graphiques

Choisir un affichage à 4 fenêtres pour obtenir la configuration de la Figure 551.

📣 Control System Designer - mc	oteur_cc_as	s_vit_contro	ol_sys_des_PID_book - LoopTransfer_moteur_cc_ass_vit_co	ontrol_sys_des_PID_book_PID_Control — 🗆 🗙		
CONTROL SYSTEM	VIEW			🖥 / ñ i to c 🗗 ? 💿		
Left/Right Single Top/Bottom Custom V TILES	Tabs F Shrink	Position ▼ Tabs to Fit betize				
Data Browser		Bode	Editor for LoopTransfer_moteur_cc_ass_vit_control_sys_ moteur_cc_ass_vit_control_sys_des_PIL	LoopTransfer_moteur_cc_ass_vit_control_sys_des_PID_book_F		
Controllers and Fixed blocks moteur_cc_ass_vit_control_sys_des Designs	;_PID_bo	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	G.M.: inf Freq: Inf Stable loop P.M.: 37.3 deg Freq: 16.1 rad/s	Nichols Chart des_PID_book/PID Controller1 To: moteur_cc_ass_vit_c 0 dB 0 20 1 dB 1 dB		
▼ Responses						
LoopTransfer_moteur_cc_ass_vit_co	ontrol_s		-			

Figure 551 : modification de la configuration des fenêtres graphiques

3. Visualisation des réponses temporelles

Afin d'effectuer le réglage, il faut pouvoir visualiser les réponses temporelles vis-à-vis de la consigne et vis-à-vis de la perturbation.

Pour cela nous allons choisir les points de linéarisation pour obtenir ces deux réponses.

Cliquer sur New Plot/New Step.

Dans la fenêtre New Step to Plot, sélectionner New Input-Output Transfer Response (Figure 552).

New Step to plot	×
Select Response to Plot: LoopTransfer_moteur_cc_ass_vit_control_sys_des_PID_book_PID_Controller LoopTransfer_moteur_cc_ass_vit_control_sys_des_PID_book_PID_Controller Open-Loop Transfer Fu Name: LoopTransfer_m New Sensitivity Transfer Response	1 ▼ 1
Locations: moteur_cc_ass_vit_control_sys_des_PID_book/PID Controller1	Ţ
Plot Cancel He	lp

Figure 552 : choix des points de linéarisation

Dans la fenêtre qui apparaît, nommer la fonction de transfert dans le champ **Response Name** 'FTBF_consigne'.

Dans le champ Specify input Signals cliquer sur Add signal to list et Choisir Select Signal from Model.

Dans le modèle Simulink de l'asservissement, cliquer sur le point d'entrée correspondant à la consigne de l'asservissement. Puis cliquer sur **Add signal** dans la fenêtre **New Step to Plot**. Cliquer alors sur le point d'entrée correspondant à la fonction de transfert en boucle fermée vis-à-vis de la consigne.



Figure 553 : choix du point d'entrée de la fonction de transfert

Cliquer alors sur Add signal(s) dans la fenêtre New Step to Plot pour valider votre choix.



Figure 554 : fenêtre New Step to Plot

Recommencer cette opération avec le champ **Specify output Signal** pour choisir le point de sortie de la fonction de transfert et valider votre choix.



Figure 555 : choix du point de sortie de la fonction de transfert

Vous devez obtenir la configuration de la Figure 556.

lew Step to plot		×
Select Response to Plot: New Input-Output Transfer Response		•
Response Name: FTBF consigne		
Specify input signals:		
moteur_cc_ass_vit_control_sys_des_PID_book/consigne/1	🔶 🖖 🏞 🗙	
+ Add signal to list		
Specify output signals:		
moteur_cc_ass_vit_control_sys_des_PID_book/Transfer Fcn1/1[wm(s)]	🔶 🕆 🏞 🗙	
+ Add signal to list		
Compute input/output response with the following loops open:		
+ Add loop opening location to list		
Plot	Cancel Help	
		.:

Figure 556 : choix des points d'entrée et de sortie de la FTBF vis-à-vis de la consigne

Cliquer alors sur **Plot** pour tracer la réponse temporelle souhaitée.



Recommencer cette opération pour obtenir la réponse de la fonction de transfert en boucle fermée vis-àvis de la perturbation.



Figure 557 : point d'entrée et de sortie pour la réponse temporelle vis-à-vis de la perturbation

ew Step to plot	×
Select Response to Plot: New Input-Output Transfer Response	•
Response Name: FTBF_perturbation	
Specify input signals:	
moteur_cc_ass_vit_control_sys_des_PID_book/perturbation/1[Cr(s)]	🔶 🔮 😓 🗙
+ Add signal to list	
Specify output signals:	
moteur_cc_ass_vit_control_sys_des_PID_book/Transfer Fcn1/1[wm(s)]	🔶 🔶 🐎 🗙
+ Add signal to list	
Compute input/output response with the following loops open:	
+ Add loop opening location to list	
Plot	Cancel Help

Figure 558 : choix des points d'entrée et de sortie de la FTBF vis-à-vis de la perturbation





Figure 559 : visualisation des fenêtres utiles pour effectuer le réglage

4. Réglage du PID

Dans la fenêtre graphique du diagramme de Bode de la FTBO, cliquer avec le bouton droit et choisir **Edit Compensator** pour ouvrir la fenêtre de réglage du PID.

Sélectionner ensuite l'onglet **Parameter** pour pouvoir agir sur les paramètres de réglage du PID.

Il est maintenant possible de procéder aux réglages du PID et de voir évoluer de manière dynamique l'ensemble des tracés.

Faire glisser les différents curseurs et observer l'influence du réglage sur les diagrammes.

📣 Compensator Ed	litor		Fonction de transfert du correcteur					
Compensator			/	•				
PID_test_PID_Con	PID_test_PID_Controller1 v = 1							
Pole/Zero Paramete	Valeurs mini des gains imposés par les curseurs (modifiable)							
Parameter	Value	Min Value	Slider	Max Value				
Р	1	0.1		1				
I	0	-1	· · · · · · · · · · · · · · · · · · ·	1				
D	0	-1	·	1				
N	100	10		100				
Curse	Curseurs de réglage des différents gains							
Valeurs maxi des gains imposés par les curseurs (modifiable)								
Select a parameter i	Select a parameter in the table and tune it manually							
				Help				

Figure 560 : les fonctions de la fenêtre de réglage du PID

5. Définition et visualisation des critères de performance

Le réglage du correcteur ne peut se faire que si des critères de performance sont définis. Il est possible de définir et de visualiser ces critères sur les courbes des réponses indicielles.

Cliquer droit dans la fenêtre graphique donnant la réponse temporelle de la FTBF vis-à-vis de la consigne et choisir **Properties**, onglet **Options** puis définir que le critère de rapidité est le temps de réponse à 5%.

📣 Property Editor: Step Response	_		×
Labels Limits Units Style Options			
Response Characteristics Show settling time within 5 % Show rise time from 10 to 90 %			
Confidence Region for Identified Models Number of standard deviations for display: 1			
	Close	e H	lelp

Figure 561 : définition des critères de rapidité

Cliquer droit dans la fenêtre graphique donnant la réponse temporelle de la FTBF et choisir **Characteristics** puis **Settling Time** pour afficher le temps de réponse à 5% (Figure 562).



Figure 562 : visualisation du temps de réponse à 5%

Clique droit dans la fenêtre graphique donnant la réponse temporelle de la FTBF et choisir **Design Requirement New** et entrer les critères de performance pour le réglage du PID conformément à la Figure 563.

📣 New Desigr	n Requirement		_		×				
Design requiren	nent type: Step response				~				
Design requirement parameters									
Initial value:	0]	Final value:			1			
Step time :	0	seconds							
		1							
Rise time :	0.8000	seconds	% Rise:			80			
Settling time :	1	seconds	% Settling:			5.0000			
% Overshoot:	10.0000]	% Undershoot:			1			
			ОК	Clos	e	Help			

Figure 563 : définition des critères de performances

On impose ici :

- un temps de montée (de 0 à 90%) de 0.8 s
- un temps de réponse à 5% de 1 s
- un dépassement maxi de 10%

La fenêtre graphique permet de visualiser la zone dans laquelle doit se situer la réponse indicielle pour satisfaire le cahier des charges (Figure 564).



Figure 564 : visualisation des critères de performances dans la fenêtre graphique

La réponse indicielle doit se trouver en dehors de la zone jaune qui est définie par les différents critères pour que le cahier des charges soit respecté.

6. Réglage du PID à l'aide des curseurs

La réponse n'est pas satisfaisante. A l'aide des curseurs, effectuer manuellement le réglage du PID. On peut voir évoluer dynamiquement les tracés La fenêtre de la Figure 565 montre un réglage satisfaisant du PID.

📣 Compensator Editor — 🗆 🗙									
Compensator book_PID_Controller1 \checkmark = 7 x $\frac{(1 + 0.37s + (0.22s)^2)}{s(1 + 0.01s)}$									
Pole-Zero Parameter	Pole-Zero Parameter								
Parameter	Value	Min Value	Slider		Max Va	lue			
P	2.5				10				
I	7	1			10				
D	0.319	-1			1				
Ν	100	10			100				
Select a parameter in the table and tune it manually									
	Help								

Figure 565 : réglage satisfaisant le cahier des charges du PID

La fenêtre de la Figure 566 montre la réponse indicielle corrigée de la FTBF.



Figure 566 : résultat du réglage manuel du PID sur la réponse temporelle

Le réglage étant satisfaisant vis-à-vis de la consigne, nous pouvons maintenant vérifier le comportement vis-à-vis de la perturbation.



Figure 567 : réponse indicielle du système vis-à-vis d'un échelon de perturbation

Nous constatons que la perturbation est rapidement rejetée et que son influence peut être considéré comme négligeable au bout de 0.8 s . Le réglage effectué est donc satisfaisant.

Les diagrammes de Bode et de Black de la FTBO permettent de visualiser l'influence de ces réglages et d'apprécier les critères de marge de gain et de marge de phase.



Figure 568 : influence des réglages effectués sur les diagrammes fréquentiels de la FTBO

7. Exportation du réglage dans le modèle Simulink

Cliquer maintenant sur **Update Block** dans la fenêtre de commande du **Control System Designer** exporter les réglages effectués dans le PID du modèle Simulink.

Retourner dans le modèle Simulink puis **Lancer** la simulation et visualiser la réponse du moteur avec le PID réglé.



Figure 569 : réponse temporelle après réglage du PID

Nous constatons sur la Figure 569 que la réponse temporelle est plus rapide, bien amortie et que la perturbation est bien rejetée par le système. Les critères de performances sont respectés.

IV. Conception et réglage d'un correcteur de forme quelconque

Dans de nombreuses applications de contrôle commande, le correcteur peut prendre d'autres formes que celle d'un simple correcteur PID. **MATLAB** offre la possibilité de construire un correcteur de forme quelconque à partir de l'outil « **Control System Designer** ». La démarche sera sensiblement la même que celle présentée dans la partie III. La fonction de transfert du correcteur sera construite par étape en lui ajoutant de nouveaux éléments dans le but d'améliorer les performances du système. L'outil « **Control System Designer** » permet d'intégrer dans la fonction de transfert du correcteur :

- des intégrateurs
- des pôles complexes
- des pôles réels
- des zéros complexes
- des zéros réels
- des correcteurs à avance de phase (Lead)
- des correcteurs à retard de phase (Lag)
- des filtres rejecteurs (Notch)

A. Ouverture du modèle

Ouvrir le fichier « moteur_cc_ass_vit_control_sys_des_tf »

Ce fichier contient le modèle de l'asservissement en vitesse d'un moteur à courant continu (Figure 570). Le correcteur a été remplacé par une fonction de transfert que l'on va construire à l'aide de l'outil « **Control System Designer** ». Pour l'instant cette fonction de transfert est un gain pur de valeur 1.

La consigne imposée est de 100 rad/s et une perturbation de couple est appliquée au système à l'instant t=3 s.



Figure 570 : modèle de l'asservissement du moteur à continu avec correction par fonction de transfert à régler
Exécuter le script *parametres_moteur_cc.m.*

Lancer la simulation et visualiser la réponse dans le scope.



Figure 571 : reponse du systeme non corrige

Nous pouvons constater que la réponse du système n'est pas satisfaisante. Le système n'est pas précis, mal amorti et ne rejette pas correctement la perturbation.

B. Conception du correcteur

A partir de l'onglet APPS de la fenêtre de commande de MATLAB, sélectionner l'application « Control System Designer » (Figure 572).



Figure 572 : lancement de l'application « Control System Designer »

Cette commande entraîne l'ouverture de la fenêtre **Control System Designer** qui propose de choisir les blocs que l'on veut régler, de visualiser les points d'entrée et de sortie de la boucle fermée.

📣 Con	trol Systen	n Designer - m	oteur_cc_ass_vit	_control_sys_des_tf							 _		\times
со	NTROL SYST	TEM	VIEW								ħ ₿ <) ¢ [• ? •
						5			2	٢			
Open Session	Save Session	Edit Architecture	Multimodel Configuration	Tuning Methods ▼	New Plot ▼	Store	Retrieve •	Compare	Update Blocks	Preferences			
FI	LE	ARCHI	TECTURE	TUNING METHODS	ANALYSIS		DESIGN	S	SIMULINK	PREFERENCES			A state of the
Data Bro	wser												
Contr	ollers and	Fixed Blocks		Edit Architecture -	Simulink C	Configura	ition			×			
2				Use this dialog to	edit the blo	ocks and	signals in	the current	t architectu	re.			
				Blocks Signals	Linearizat	tion Opti	ons						
				Block Name			Val	ue					
								Add	d Blocks				
🔻 Desig	ns												
Response	onses												
						[ОК	Cancel	Help				
									·				
▼ Previe	w												
-													

Figure 573 : Control System Designer

Afin de sélectionner le bloc à régler cliquer sur **Add Blocks** et choisir **Correcteur** comme bloc à régler, puis cliquer sur **OK**. (Figure 574).

🎦 Select Blocks to Tune			×
🚡 moteur_cc_ass_vit_control_sys Se	elect blocks in th	e table below to tune:	
	Tune?	Block Name	
		Gain2	
		Gain1	
		Transfer Fcn1	
		Transfer Fcn	
		Correcteur	
			Highlight Selected Block
			OK Cancel

Figure 574 : choix du bloc à régler

Dans la fenêtre Edit Architecture (Figure 575) qui apparaît, cliquer sur OK pour valider votre choix.

Edit Archi	tecture -	Simulink Configu	ration		×
Use this o	dialog to	edit the blocks an	d signals in tl	he current a	rchitecture.
Blocks	Signals	Linearization Op	tions		
Block N	lame trol_sys_d	es_tf/Correcteur	Value <1x1 zpk>		🐉 🗙
				Add	Blocks
			ОК	Cancel	Help

Figure 575 : fenêtre Edit Architecture

1. Diagrammes fréquentiels de la FTBO

Nous allons maintenant choisir la méthode de réglage. Dans la fenêtre du Control System Designer, sélectionner **Tuning Method/Bode Editor**.

La fenêtre **Select Response to Edit** apparaît. La fonction de transfert en boucle ouverte (FTBO) est automatiquement détectée. Les points de linéarisation sont indiqués (Figure 576).

Select Response to Edit - Bode	×
Select Response to Edit: LoopTransfer_moteur_cc_ass_vit_control_sys_des_tf_Correcteu	r 🕶
Open-Loop Transfer Function Name: LoopTransfer_moteur_cc_ass_vit_control_sys_des_tf_Correcteur Locations: moteur_cc_ass_vit_control_sys_des_tf/Correcteur	Î
Plot Cancel Hel	р

Figure 576 : choix des points de linéarisation pour la FTBO

Cliquer sur **Plot** pour tracer le diagramme de Bode de la FTBO. Pour ajouter un quadrillage, cliquer avec le bouton droit dans la fenêtre du diagramme de Bode et choisir **Grid**.



Figure 577 : diagramme de Bode de la FTBO

Il est également possible d'ajouter d'autres types de diagramme pour la FTBO. Cliquer sur **New Plot/New Nichols** puis cliquer sur **Plot** pour faire apparaître un diagramme de Black.



Figure 578 : diagramme de Black de la FTBO

Afin de visualiser les deux graphiques dans la même fenêtre, cliquer sur l'onglet **View** puis **Custom** pour choisir la configuration des fenêtres graphiques (Figure 579).



Figure 579 : choix de la configuration des fenêtres graphiques



Choisir un affichage à 4 fenêtres pour obtenir la configuration de la Figure 580.

Figure 580 : modification de la configuration des fenêtres graphiques

2. Visualisation des réponses temporelle et fréquentielle de la FTBF

Afin d'effectuer le réglage, il faut pouvoir visualiser les réponses temporelles et fréquentielle de la FTBF vis-à-vis de la consigne. Nous allons donc afficher :

- La réponse indicielle de la FTBF vis-à-vis de la consigne
- Le diagramme de Bode de la FTBF vis-à-vis de la consigne afin de visualiser les éventuels problèmes de résonnance en boucle fermée qui pourraient apparaître lors des réglages.

Pour cela nous allons choisir les points de linéarisation pour obtenir ces deux réponses. Cliquer sur **New Plot/New Step.**

Dans la fenêtre New Step to Plot, sélectionner New Input-Output Transfer Response (Figure 581).

New Step to plot ×
Select Response to Plot: LoopTransfer_moteur_cc_ass_vit_control_sys_des_tf_Correcteur LoopTransfer_moteur_cc_ass_vit_control_sys_des_tf_Correcteur Open-Loop Transfer Fu New Input-Output Transfer Response Name: LoopTransfer_m New Sensitivity Transfer Response Locations: moteur_cc_ass_vit_control_sys_des_tf/Correcteur
Plot Cancel Help

Figure 581 : choix des points de linéarisation

Dans la fenêtre qui apparaît, nommer la fonction de transfert dans le champ **Response Name 'FTBF_consigne'** (Figure 583).

Dans le champ **Specify input Signals** cliquer sur **Add signal to list** et Choisir **Select Signal from Model**.

Dans le modèle Simulink de l'asservissement, cliquer sur le point d'entrée correspondant à la consigne de l'asservissement. Puis cliquer sur **Add signal** dans la fenêtre **New Step to Plot**. Cliquer alors sur le point d'entrée correspondant à la fonction de transfert en boucle fermée vis-à-vis de la consigne.



Figure 582 : choix du point d'entrée de la fonction de transfert

Cliquer alors sur **Add signal(s)** dans la fenêtre **New Step to Plot** pour valider votre choix.

New Step to plot			×
Select Response to	Plot: New Input-Output Tran	sfer Response	-
Response Name:	FTBF consigne		
Specify input sig	nals:	Nom de la	
+ Add signal to list		transfert	
Select signals			×
Click signals to	add them to list from Simulink m s_vit_control_sys_des_tf_test/c	odel, moteur_cc_ass_vit_co onsigne:1 (consigne) Signal ajouté	ontrol_sys_des_tf_test
	Validation du choix	Ad d Sig	nal(s) Cancel 🝞
		Plot	ancel Help

Figure 583 : fenêtre New Step to Plot

Recommencer cette opération avec le champ **Specify output Signal** pour choisir le point de sortie de la fonction de transfert et valider votre choix.



Figure 584 : choix du point de sortie de la fonction de transfert

Vous devez obtenir la configuration de la Figure 585.

New Step to plot		×
Select Response to Plot: New Input-Output Transfer Response		•
Response Name: FTBT_consigne		
Specify input signals:		
moteur_cc_ass_vit_control_sys_des_tf/consigne/1[consigne]		🐎 🗙
+ Add signal to list		
Specify output signals:		
moteur_cc_ass_vit_control_sys_des_tf/Transfer Fcn1/1[sortie]	- 🔶 🕹	🐉 🗙
+ Add signal to list		
Compute input/output response with the following loops open:		
+ Add loop opening location to list		
Plot	Cancel	Help

Figure 585 : définition des points de linéarisation de FTBF_consigne

Cliquer sur Plot pour obtenir le tracé de la réponse indicielle de la FTBF.



Figure 586 : ajout de la réponse indicielle de la FTBF

Nous allons maintenant tracer le diagramme de Bode de la FTBF vis-à-vis de la consigne. Pour cela cliquer sur **New Plot/New Bode**.

Choisir la fonction de transfert FTBF_consigne (déjà définie) dans le menu déroulant de la fenêtre **New Bode to Plot** puis cliquer sur **Plot** (Figure 587).

New Bode to plot	×	
Select Response to Plot:	FTBT_consigne	
Input-Output Transfer I Name: FTBT_consigne Inputs: moteur_cc_ass_vit_co Outputs: moteur_cc_ass_vit_co	LoopTransfer_moteur_cc_ass_vit_control_sys_des_tf_Correcteur New Input-Output Transfer Response New Open-Loop Response New Sensitivity Transfer Response ntrol_sys_des_tf/Transfer Fcn1/1[sortie]	
	Plot Cancel Help	
		ļ

Figure 587 : tracé du diagramme de Bode de la FTBF

Vous devez obtenir la configuration de la Figure 588 qui permet de visualiser tous les diagrammes nécessaires à la conception du correcteur.



Figure 588 : configuration des diagrammes pour effectuer le réglage

3. Synthèse du correcteur

L'outil Control System Designer permet d'ajouter dans le correcteur différents éléments :

- des intégrateurs
- des pôles complexes
- des pôles réels
- des zéros complexes
- des zéros réels
- des correcteurs à avance de phase (Lead)
- des correcteurs à retard de phase (Lag)
- des filtres rejecteurs (Notch)

Pour cela plusieurs méthodes permettent d'intégrer ces fonctions de transfert au correcteur.

Cliquer dans la fenêtre graphique du diagramme de Bode de la FTBO pour faire apparaître l'onglet **Bode Editor** (Figure 589).



Figure 589 : la barre de commande du Bode Editor

Il est également possible d'ajouter des éléments au correcteur en cliquant avec le bouton droit dans la fenêtre graphique du diagramme de Bode de la FTBO, puis Add Pole/zero pour faire apparaître un menu déroulant qui permet de choisir les éléments à ajouter au correcteur.



Figure 590 : ajouts d'éléments au correcteur

4. Visualisation de l'influence du gain de la FTBO

Il est possible dans un premier temps de faire varier le gain du correcteur et de visualiser son influence sur l'ensemble des diagrammes.

Dans la fenêtre graphique du diagramme de Bode en gain de la fonction de transfert en boucle ouverte, **déplacer** le pointeur de la souris sur la courbe de gain. Lorsqu'il prend la forme d'une main, utiliser le **glisser déposer** pour faire translater la courbe verticalement.

Cette translation aura comme influence de modifier la valeur du gain du correcteur que nous concevons.

Déplacer la courbe de gain afin d'obtenir un gain basse fréquence de l'ordre de **20dB** pour la fonction de transfert en boucle ouverte.



Figure 591 : variation du gain de la fonction de transfert en boucle ouverte

Nous constatons que les autres diagrammes se mettent à jour de manière dynamique en prenant en compte l'augmentation du gain de la fonction de transfert en boucle ouverte.

Sur la réponse indicielle, la précision et la rapidité augmentent, mais l'amortissement diminue laissant apparaître de plus fortes oscillations. Sur le diagramme de Bode de la FTBF, on observe une augmentation de la bande passante à -3dB (amélioration de la rapidité) et une résonnance plus importante.

Il est également possible de visualiser la nouvelle fonction du correcteur dans la fenêtre. Pour cela cliquer avec le bouton droit dans la fenêtre du diagramme de Bode de la FTBO et choisir **Edit Compensator** dans le menu contextuel.

📣 Compensator Editor		-		×
Compensator				
rol_sys_des_tf_Correcteur v = 8.7181				
Pole-Zero Parameter				
Dynamics	Edit Selected Dynamics Select a single row	to edit val	ues	
			ł	Help

Figure 592 : Edition de la fonction de transfert su correcteur

Il apparaît que le correcteur se limite ici à un correcteur proportionnel de gain **8,7181**.

Il est également possible de modifier le gain du correcteur en utilisant d'autres procédés :

- En faisant translater verticalement le tracé de Nichols
- En indiquant directement la valeur du gain dans la fenêtre **Compensator Editor**.

Dans tous les cas la mise à jour des autres diagrammes est réalisée automatiquement.

5. Ajout d'un intégrateur

Afin de rendre le système précis, il est possible d'ajouter un intégrateur dans le correcteur. Pour cela **cliquer** avec le bouton droit de la souris, dans la fenêtre **graphique du diagramme de Bode de la FTBO** puis sélectionner **Add Pole zero/Integrator** (Figure 593). Un intégrateur est ajouté à notre correcteur et toutes les courbes sont mises à jour.



Figure 593 : ajout d'un intégrateur dans le correcteur

L'ajout d'un intégrateur à rendu l'asservissement instable. Les marges de gain et de phase sont négatives et sont visibles sur le diagramme de Bode et le tracé de Nichols. La réponse indicielle met en évidence cette instabilité (Figure 594).



Figure 594 : visualisation de l'instabilité de la réponse indicielle

Il est possible de visualiser la nouvelle fonction de transfert du correcteur (Figure 595).

📣 Compensato	or Editor				-		×	
Compensator								
rol_sys_des_tf	f_Correcteur ~	= 8.7181	x - s					
Pole-Zero Parameter								
Dynamics			Edit Selected Dynam	nics				
Туре	Location	Location Damping						
Integrator	0	-1	0					
Right-click to add or delete poles/zeros				Select a singl	e row to edit valu	: values		
						ł	Help	

Figure 595 : visualisation de la fonction de transfert du correcteurt

6. Ajout d'un correcteur à avance de phase (Lead)

Afin d'augmenter les marges de gain et de phase, il est possible d'ajouter un correcteur à avance de phase (Lead).

Pour cela **cliquer** avec le bouton droit de la souris, puis sélectionner **Add Pole zero/Lead.**

A l'aide du pointeur de la souris cliquer ensuite au niveau de la pulsation de coupure sur le diagramme de Bode en gain de la fonction de transfert en boucle ouverte. Le pôle du correcteur à avance de phase sera alors placé au niveau de la pulsation de coupure. Le zéro sera placé automatiquement à une pulsation inférieure.

Il est possible de visualiser les pôles et les zéros sur toutes les courbes (Figure 596).



Figure 596 : visualisation des pôles et des zéros sur les courbes

Pour régler le correcteur à avance de phase, il suffit de déplacer à l'aide du pointeur de la souris le pôle et le zéro ainsi ajoutés pour obtenir une marge de gain et de phase positive afin de rendre le système stable.

Positionner le **pôle** autour de la pulsation **1000 rad/s** et le **zéro** autour de la pulsation **4 rad/s**.

Les marges de gain et de phase sont maintenant positives et le système est stable (Figure 597).



Figure 597 : stabilisation du système par ajout d'un correcteur à avance de phase

La réponse indicielle converge et la précision est bonne.

Le diagramme de Bode en gain de la fonction de transfert en boucle fermée, fait apparaître une forte résonance autour de la pulsation 22 rad/s qui témoigne de la mauvaise qualité de l'amortissement de la réponse indicielle (Figure 597).

Il est possible de visualiser la nouvelle fonction de transfert du correcteur (Figure 598).

📣 Compensato	or Editor				-	- 🗆	×	
Compensator			(1					
I_sys_des_tf_C	l_sys_des_tf_Correcteur \checkmark = 8.7181 x $\frac{(1 + 0.25s)}{s(1 + 0.001s)}$							
Pole-Zero Para	meter							
Dynamics				Edit Selected Dyna	nics			
Туре	Location	Damping	Frequency					
Integrator	0	-1	0					
Right-click to a	add or delete pol	es/zeros		Real Zero Real Pole Max Delta Phase (deg) at Frequency	-4 -1000 82.762 63.246			
							Help	

Figure 598 : visualisation de la fonction de transfert du correcteur

7. Ajout d'un filtre rejecteur (Notch)

Afin de compenser le phénomène de résonance constaté sur le diagramme de Bode de la fonction de transfert en boucle fermée (pic à la pulsation 20 rad/s). Il est possible d'ajouter un filtre rejecteur (**Notch**). Ce filtre permettra d'atténuer la résonance et de diminuer les oscillations.

Pour cela cliquer avec le bouton droit de la souris, puis sélectionner Add Pole zero/Notch.

Cliquer ensuite sur le diagramme de Bode en gain de la fonction de transfert en boucle ouverte légèrement en retrait du pic de résonnance constaté sur le diagramme de Bode de la fonction de transfert en boucle fermée (environ 10 rad/s).

Le filtre rejecteur est maintenant ajouté au correcteur et les diagrammes sont mis à jour (Figure 599).



Figure 599 : ajout d'un filtre rejecteur au correcteur

Nous pouvons constater l'influence du filtre rejecteur sur la réponse indicielle et sur le diagramme de Bode de la boucle fermée. Il apparaît cependant que le filtre n'est pas bien réglé dans la mesure où la résonnance du diagramme de Bode de la boucle fermée n'est pas totalement compensée et des oscillations importantes persistent sur la réponse indicielle.

8. Réglage d'un filtre rejecteur

Pour régler le filtre rejecteur, il est possible de modifier graphiquement 3 paramètres :

- La fréquence à laquelle va agir le filtre (**Natural Frequency**)
- L'atténuation en dB que l'on souhaite obtenir (Notch Depth)
- La bande de fréquence dans laquelle le filtre doit agir (Notch Width)

Ces paramètres sont visualisables dans la fenêtre **Compensator Editor** et peuvent être modifier directement (Figure 600).

📣 Compensator Editor		_		×				
Compensator s_tf_test_Correcteur \checkmark = 8.7181 x $(1 + 0.24s)(1 + 0.52s)(1 + 0.52$	0098s + (0.098s)^2) + 0.098s + (0.098s)^2)							
Pole/Zero Parameter								
Dynamics Type Location Damping Frequency Integrator 0 -1 0 Lead -4.18, -1.02e+03 1 4.18, 1.02e+03 Notch -0.512 +/- 10 0.05, 0.5 10.2	Edit Selected Dynamic Natural Frequency Damping (Zero) Damping (Pole) Notch Depth (dB) Notch Width (Log)	10.237 0.05 0.5 -20 0.28547						
				Help				

Figure 600 : visualisation des paramètres de réglage du filtre rejecteur

Cependant, ces paramètres peuvent être réglés de manière dynamique à partir du diagramme de Bode en gain de la fonction de transfert en boucle ouverte par simple glisser déposer des poignées représentant les paramètres du filtre indiqués sur la Figure 601.



Figure 601 : réglage du filtre rejecteur

En visualisant simultanément le diagramme de Bode de la fonction de transfert en boucle fermée et le diagramme de Bode de la fonction de transfert en boucle ouverte, régler les trois paramètres du filtre de manière à voir disparaître la résonnance sur le diagramme de Bode de la fonction de transfert en boucle fermée.

Une fois les réglages effectuées les performances de la boucle fermée sont sensiblement améliorées (Figure 602).



Figure 602 : performances de la boucle fermée après réglage du filtre

La réponse indicielle est précise, rapide et bien amorti. Le correcteur a permis de trouver un bon compromis entre toutes les performances recherchées.

Il est possible de visualiser la fonction de transfert du correcteur dans la fenêtre dans la fenêtre du **Compensator Editor** (Figure 603).

A Compensator Editor – 🗆 🗙					×		
Compensator (1 + 0.25s) (1 + 0.045s + (0.086s)^2)							
s_des_tf_Correc	$s_des_tf_Correcteur \lor = 8.7181 x \frac{(c + 0.001s)(1 + 0.086s)(1 + 0.086s)}{s (1 + 0.001s) (1 + 0.086s) (1 + 0.086s)}$						
Pole-Zero Paran	Pole-Zero Parameter						
Dynamics				Edit Selected Dyna	mics		
Туре	Location	Damping	Frequency				
Integrator	0	-1	0	1			
Lead	-4, -1e+03	1	4, 1e+03				
Notch	-3.03 +/- 11.2i	0.26, 1	11.6	Natural Frequency	/ 11.64		
				D . 7	0.00		
				Damping (Zero) 0.20		
				Damping (Pole) 1		
				Notch Depth (dB) -11.701		
				Notch Width (Log) 0.73445		
Right-click to ac	dd or delete poles	/zeros					
							Help

Figure 603 : visualisation de la fonction de transfert du correcteur

Il est également possible de voir l'influence du filtre rejecteur sur l'ensemble des courbes représentant le comportement de la boucle ouverte (Figure 604).



Figure 604 : influence du filtre sur le comportement de la boucle ouverte

9. Exportation de la fonction de transfert du correcteur vers le modèle Simulink

Cliquer maintenant sur **Update Block** dans la barre de commande du **Control System Designer** afin d'exporter la fonction de transfert du correcteur dans modèle Simulink.

Retourner dans le modèle Simulink, **lancer** la simulation et visualiser la réponse du système dans le scope.



Figure 605 : réponse indicielle corrigée

La réponse indicielle corrigée (Figure 605) est précise, la rapidité et l'amortissement sont satisfaisants et la perturbation bien rejetée.

La comparaison des réponses corrigée et non corrigée permet de visualiser l'apport du correcteur (Figure 606).



Chapitre 11 : Ingénierie numérique avec MATLAB

I. Introduction

Dans un processus de conception ou de validation des performances d'un système, l'ingénieur est amené à mettre en place des modèles qui lui permettent de décrire le comportement des éléments qui composent le système. La phase de modélisation va permettre d'établir des équations qui caractérisent le comportement d'un système. Sauf cas particuliers, la résolution de ces équations n'est en général pas possible en utilisant un processus de résolution analytique. La démarche de résolution est donc le plus souvent menée en utilisant un processus de calcul numérique. L'ingénieur doit être en mesure de coder avec un langage performant et adapté le processus de résolution qui correspond au problème traité. Ce chapitre présente les méthodes et les algorithmes qui sont à la base des compétences en ingénierie numérique :

- Utilisation et gestion des fonctions
- Dérivation numérique
- Résolution des équations algébriques
- Intégration numérique
- Filtrage des signaux
- Résolution des équations différentielles
- Résolution des systèmes d'équations...

La maitrise et l'utilisation courante de ces algorithmes permettra au futur ingénieur de renforcer ses compétences et d'être en mesure de résoudre une grande partie des problèmes auxquels il sera confronté.

II. Les fonctions

A. Organisation et structure

La bonne utilisation des fonctions est un élément déterminant de la conception des algorithmes en langage MATLAB. Il faut donc commencer ce chapitre par définir les modalités de création et d'utilisation des fonctions. Il existe en langage MATLAB de très nombreuse fonctions prédéfinies qui peuvent être utilisées directement en se référant à l'aide du logiciel afin de prendre connaissance des entrées (arguments) et des sorties de la fonction.

Il est par contre indispensable de définir ses propres fonctions pour les besoins d'un algorithme. L'utilisation des fonctions permet de structurer le code et le rendre plus lisible. Ainsi un script principal pourra faire appel durant son exécution à des fonctions définies préalablement. En langage MATLAB, il existe plusieurs manières de définir et d'utiliser les fonctions.

La première méthode consiste à définir les fonctions dans des fichiers différents du script principal. Sur la Figure 607, le script **Main_script_1** fait appel aux fonctions **function_1** et **function_2** durant son exécution. Le script **Main_script_2** fait appel aux fonctions **function_3**, **function_4** et **function_2** durant son exécution. L'intérêt de coder les fonctions dans des scripts séparés permet donc à des scripts différents de faire appel à une même fonction ou d'appeler une fonction directement à partir de la fenêtre de commande. C'est très pratique et c'est en général la méthode à privilégier. Pour pouvoir être appelée, une fonction doit se trouver dans le « **path** » de MATLAB, il est donc inutile d'indiquer le chemin d'accès à la fonction. L'inconvénient est que deux fonctions différentes ne pourront pas porter le même nom.



Figure 607 : organisation des scripts et des fonctions, méthode 1

La seconde méthode consiste à définir les fonctions directement dans les scripts dans lesquels elles seront utilisées. Ces fonctions sont obligatoirement placées à la fin du script.

Sur la Figure 608, le script **Main_script_1** fait appel aux fonctions **function_1** et **function_2** durant son exécution. Le script **Main_script_2** fait appel aux fonctions **function_3**, **function_4** durant son exécution. **Mainscript_2** ne pourra pas faire appel aux fonctions **function_1** ou **function_2**. Ce type de structure est adaptée à des fonctions qui n'auront d'intérêt que pour le script dans lequel elles sont définies.



Figure 608 : organisation des scripts et des fonctions, méthode 2

B. Portée des variables

D'une manière générale, chaque fonction possède son propre **workspace** et utilise ses propres variables. Que ce soit dans le cas de la Figure 607 ou dans le cas de la Figure 608 les variables du **workspace** de MATLAB ne seront pas accessibles dans les différentes fonctions. De même les variables utilisées dans les fonctions n'apparaitront pas dans le **workspace** de MATLAB et ne pourront pas être utilisées dans le script. Si des variables doivent être partagées entre le script et les fonctions, des commandes spécifiques permettent de le préciser, mais cela n'aura pas lieu par défaut.

C. Création et définition d'une fonction

Vous pouvez trouver les fichiers de toutes les fonctions présentées dans ce paragraphe dans le dossier **Algorithmique_avec_MATLAB/ Fonctions.**

Prenons l'exemple d'une fonction élémentaire **my_0_function** (Figure 609) permettant de réaliser un calcul simple en prenant un nombre **x** comme argument et renvoyant la valeur calculée de **y** en fonction de **x**.



Figure 609 : my_0_function

La première méthode consiste à créer la fonction dans un fichier dédié à la fonction. Ce fichier doit respecter les règles suivantes (Figure 610) :

- commencer par le mot clé function
- préciser le nom de la fonction : **my_0_function**
- définir l'ensemble des paramètres d'entrée (arguments) : x
- définir l'ensemble des paramètres de sortie : y
- Le nom du fichier doit correspondre au nom de la fonction : **my_0_function.m**
- Se terminer par le mot réservé end



Figure 610 : codage de la fonction my_0_function

Créer le dossier « Mes_fonctions » que vous placerez dans le **path** de MATLAB. Choisir ce dossier comme dossier courant dans MATLAB.

Ouvrir un nouveau script et le sauvegarder sous le nom « **my_0_function.m** ». **Taper** les lignes de code suivante :

```
function y = my_0 function(x)
y = 3*x + 2
end
```

Sauvegarder votre saisie.



Figure 611 : codage de la fonction my_0_function

La fonction **my_0_function** est prête à être utilisée et apparaît comme une fonction dans le répertoire courant.

Currer	Current Folder	
	Name 🔺	
Function		
1	🖄 my_0_function.m	

Figure 612 : visualisation de la fonction my_0_function dans le répertoire courant

Dans la fenêtre de commande, faire un appel à la fonction en spécifiant une valeur pour l'argument d'entrée.



ans	=
1'	7

Il est également possible de stocker le contenu du résultat dans une variable :

>> a = my_0_function(5) a = 17

Cette fonction peut maintenant être utilisée depuis la fenêtre de commande ou être appelée par un script durant son exécution.

Une fonction peut posséder plusieurs entrées et plusieurs sorties. C'est le cas de la fonction **my_1_function** de la Figure 613.



Figure 613 : my_1_function

La fonction sera codée comme indiqué sur la Figure 614.



Figure 614 : codage de my_1_function

Ouvrir un nouveau script et le sauvegarder dans le dossier **Mes_fonctions** sous le nom « **my_1_function.m** ».

Taper les lignes de code suivante :

function [s1,s2,s3] = my_1_function(e1,e2,e3)
s1 = e1 + e2;
s2 = 2*e1 + e2 + 3*e3;
s3 = 3*e3;
end

Sauvegarder votre saisie.

	my	_0_function.m × my_1_function.m × +
1		$-$ function [s1,s2,s3] = my_1_function(e1,e2,e3)
2	—	s1 = e1 + e2;
3	-	s2 = 2*e1 + e2 + 3*e3;
4	—	s3 = 3*e3;
5	—	end

Figure 615 : codage de la fonction my_1_function

La fonction **my_1_function** est prête à être utilisée et apparaît comme une fonction dans le répertoire courant.

Current Folder		
	Name 📥	
Function		
🖄 my_0_function.m		
🖄 my_1_function.m		
	-	

Figure 616 : visualisation de la fonction my_1_function dans le répertoire courant

Dans la fenêtre de commande, faire un appel à la fonction **my_1_function** en spécifiant une valeur pour chaque argument d'entrée.

```
>> my_1_function(1,2,3)
ans =
3
```

Nous constatons que par défaut la fonction ne renvoie pas toutes ses sorties mais uniquement la première sortie **s1**. Il est possible de lui demander la sortie que l'on souhaite en tapant la commande suivante :

```
>> [~,~,a] = my_1_function(1,2,3)
a =
9
```

Enfin, il est également possible de stocker le contenu des trois sorties dans des variables :

Cette fonction peut maintenant être utilisée depuis la fenêtre de commande ou être appelée par un script durant son exécution.

D. Utilisation d'une fonction par un script

Nous allons maintenant examiner comment un script fait appel à une fonction et utilise le résultat de cette fonction. L'objectif est de créer un script permettant d'obtenir la représentation graphique d'une fonction avec la variation d'un paramètre. La fonction sera créée dans un fichier séparé **my_2_function.m**.

La fonction à tracer est définie par $y(x) = \sin(x) e^{-\frac{x}{a}}$



Figure 617 : my_2_function

La fonction sera codée comme indiqué sur la Figure 618.



Figure 618 : codage de my_2_function

Ouvrir un nouveau script et le sauvegarder dans le dossier **Mes_fonctions** sous le nom « **my_2_function.m** ».

Taper les lignes de code suivante :

```
function y = my_2_function(x,a)
y = sin(x).*exp(-x/a);
end
```

Sauvegarder votre saisie.



Figure 619 : codage de la fonction my_2_function

La fonction **my_2_function** est prête à être utilisée et apparaît comme une fonction dans le répertoire courant.



Figure 620 : visualisation de la fonction my_2_function dans le répertoire courant

Créer un nouveau script et saisir les lignes de commandes suivantes ou **ouvrir** directement le fichier « script_0.m ».

```
%% script 0
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Ce script permet de tracer une série de courbes en faisant varier un paramètre
% L'équation de la courbe est stockée dans la fonction my 2 function.m
% création d'une nouvelle fenêtre graphique
figure;
hold on;
% création du vecteur des temps
t = [0:0.001:60];
% boucle permettant le tracé avec la variation du paramètre
for k = 5:5:40
plot(t,my 2 function(t,k),'LineWidth',2); % appel de la fonction
end
% affichage de la grille
grid on;
grid minor;
% titre et étiquettes des axes
title('sin(t) *exp(-t/a)', 'Color', 'red');
xlabel('Temps','Color','blue');
ylabel('Valeur du sinus','Color','blue');
```

Figure 621 : script_0.m

Ici lors de l'appel à la fonction **plot**, la fonction qui sera tracée sera la fonction contenue dans **my_2_function**. Le premier argument de **my_2_function** sera le temps **t** et le second argument sera le paramètre **k** qui va varier pour chacune des courbes.

Sauvegarder et exécuter le script.

Le graphique obtenu est visualisable sur la Figure 622.



Figure 622 : résultat obtenu suite à l'exécution de script_0.m

Les différentes courbes apparaissent, mais il est impossible de relier une courbe particulière à la valeur du paramètre **a** correspondant, ce qui rend cette série de courbes inexploitables. Nous allons apporter quelques améliorations à ce script en lui permettant de générer automatiquement une légende afin de relier les courbes aux différentes valeurs du paramètre **a**. Nous utiliserons également l'interpréteur Tex afin d'écrire l'équation de manière plus lisible.

Pour créer la légende nous utiliserons un tableau de type **cell** permettant de stocker tout type de données et en particulier des chaines de caractères. A chaque itération, une ligne de la légende est générée puis stockée dans la cellule sous la forme d'une chaine de caractères. La légende est ensuite ajoutée au graphique à l'aide de la commande **legend** qui permet d'afficher directement le contenu de la cellule.

Pour utiliser l'interpréteur Tex, nous ferons appel à la commande **texlabel**.

Modifier le script conformément à la Figure 623 ou ouvrir le fichier « script_0_legende.m ».

```
%% script_0_legend
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
```

Ivan LIEBGOTT - Modélisation et simulation des systèmes multi-physiques avec MATLAB/Simulink 2020b

```
% Ce script permet de tracer une série de courbes en faisant varier un paramètre
% La légende est générée automatiquement
% L'équation de la courbe est stockée dans la fonction my 2 function.m
% création d'une nouvelle fenêtre graphique
figure;
hold on;
% création du vecteur des temps
t = [0:0.001:60];
% boucle permettant le tracé avec la variation du paramètre
for k = 5:5:40
plot(t,my_2_function(t,k),'LineWidth',2); % appel de la fonction
end
% affichage de la grille
grid on;
grid minor;
% titre et étiquettes des axes
title('sin(t) *exp(-t/a)', 'Color', 'red');
xlabel('Temps', 'Color', 'blue');
ylabel('Valeur du sinus','Color','blue');
% Ce script permet de tracer une série de courbes en faisant varier un paramètre
% L'équation de la courbe est stockée dans la fonction my_2_function.m
% création d'une nouvelle fenêtre graphique
figure;
hold on;
% création du vecteur des temps
t = [0:0.001:60];
%création d'une cellule pour stocker les lignes de la légende
Titre Legende = cell(1,1);
ind Legende = 1;
% boucle permettant le tracé avec la variation du paramètre
for k = 5:5:40
plot(t,my_2_function(t,k), 'LineWidth',2);
Titre_Legende{1, ind_Legende} =
                                'a=' + string(k); % création de la ligne de la légende
ind Legende = ind Legende + 1;
end
% affichage de la légende
legend(Titre Legende);
% affichage de la grille
grid on;
grid minor;
% titre et étiquettes des axes
titre = texlabel('sin(t).e^((-t/a)) en fonction de t');
title(titre, 'Color', 'red');
xlabel('Temps t en (s)','Color','blue');
ylabel('Valeur du sinus amorti', 'Color', 'blue');
```

Figure 623 : ajout d'une légende automatiquement au graphique

Sauvegarder, puis exécuter le script pour obtenir la représentation graphique de la Figure 624.


Figure 624 : représentation graphique avec légende automatique

Il est maintenant possible de relier chaque courbe à la valeur du paramètre **a** correspondant. L'équation du titre apparaît de manière plus explicite.

Il est également possible d'intégrer la fonction à tracer directement dans le script. Un exemple est donné dans le fichier « **script_0_legend_all_in_one.m** » (Figure 625). L'exécution de ce script donne un résultat rigoureusement identique.

```
%% script_0_legend_all_in_one.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Ce script permet de tracer une série de courbes en faisant varier un paramètre
% L'équation de la courbe est stockée dans la fonction définie en fin de
% fichier
% création d'une nouvelle fenêtre graphique
figure;
hold on;
% création du vecteur des temps
t = [0:0.001:60];
```

```
%création d'une cellule pour stocker les lignes de la légende
Titre Legende = cell(1,1);
ind Legende = 1;
% boucle permettant le tracé avec la variation du paramètre
for k = 5:5:40
plot(t,f(t,k),'LineWidth',2);
                                'a=' + string(k); % création de la ligne de la légende
Titre_Legende{1, ind_Legende} =
ind_Legende = ind_Legende + 1;
end
% affichage de la légende
legend(Titre Legende);
% affichage de la grille
grid on;
grid minor;
% titre et étiquettes des axes
titre = texlabel('sin(t).e^((-t/a)) en fonction de t');
title(titre, 'Color', 'red');
xlabel('Temps t en (s)', 'Color', 'blue');
ylabel('Valeur du sinus amorti', 'Color', 'blue');
% définition de la fonction
function y=f(x,a)
y = sin(x) \cdot exp(-x/a);
end
```

Figure 625 : script_0_legend_all_in_one

E. Création d'une fonction prenant en argument une autre fonction

Il est très fréquent d'avoir besoin de créer une fonction qui prendra en argument une autre fonction. A titre d'exemple, nous allons créer une fonction qui correspond à un besoin fréquent des utilisateurs de MATLAB. Afin de pouvoir formater rapidement et efficacement les graphiques que l'on doit tracer il peut être intéressant de créer une fonction qui permettra d'obtenir la représentation graphique d'une courbe en imposant un formatage du graphique prédéfini et qui sera donc le même pour toutes les courbes que l'on souhaite tracer. Pour cela nous allons créer une fonction **my_plot_function** qui prendra en argument une autre fonction **my_3_function** qui contient l'équation de la courbe.

La fonction my_plot_function (Figure 626) prendra plusieurs arguments en entrée :

- **a** : limite inférieure pour les abscisses
- **b** : limite supérieure pour les abscisses
- **nb_points** : nombres de points à calculer pour le tracé
- **f** : fonction à tracer
- titre: titre du graphique
- Xtitle: titre de l'axe des abscisses
- **Ytitle**: titre de l'axe des ordonnées



Figure 626 : my_plot_function

Nous pouvons remarquer que cette fonction ne renvoie aucune variable. Son action sera limitée à la génération d'une fenêtre graphique contenant la courbe à tracer.

Ouvrir le fichier contenant la fonction « **my_plot_function.m** » (Figure 627)

```
%% my_plot_function.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Trace une courbe avec une mise en forme prédéfinie
2
% my plot function(a,b,nb points,f,titre,Xtitle,Ytitle)
% a : limite inférieure pour les abscisses
% b : limite supérieure pour les abscisses
% nb points : nombres de points à calculer pour le tracé
% f : fonction à tracer
% titre: titre du graphique
% Xtitle: titre de l'axe des abscisses
% Ytitle: titre de l'axe des ordonnées
function my_plot_function(a,b,nb_points,f,titre,Xtitle,Ytitle)
% Création d'une fenêtre graphique
figure;
% création du vecteur des temps
t = [a:(b-a)/nb points:b];
% tracé de la courbe et utilisation de la fonction prise en argument
plot(t,f(t),'LineWidth',2);
%génération de la grille
grid on;
grid minor;
% titre et étiquette des axes
title(titre, 'Color', 'red');
```

```
xlabel(Xtitle,'Color','blue');
ylabel(Ytitle,'Color','blue');
% spécification des limites sur l'axe des abscisses
xlim([a,b]);
% spécification des limites sur l'axe des ordonnée
% axis donne les limites ymin et ymax
% les limites sur l'axe des ordonnées sont fixées à 20% au dessus des
% valeurs mini et maxi de la courbe
l = axis;
ymin = l(3); ymax = l(4);
ylim([1.2*ymin,1.2*ymax]);
end
```

Figure 627 : my_plot_function

Saisir la commande suivante dans la fenêtre de commande.

>> help my_plot_function

my_plot_function.m Ivan LIEBGOTT @ Novembre 2019 Modélisation et Simulation des Systèmes Multi-Physiques avec MATLAB - Simulink Trace une courbe avec une mise en forme prédéfinie my_plot_function(a,b,nb_points,f,titre,Xtitle,Ytitle)

a : limite inférieure pour les abscisses b : limite supérieure pour les abscisses nb_points : nombres de points à calculer pour le tracé f : fonction à tracer titre: titre du graphique Xtitle: titre de l'axe des abscisses Ytitle: titre de l'axe des ordonnées

La fenêtre de commande renvoie le commentaire placé en début de fonction de précédé du signe %%. Ce commentaire constitue une aide pour l'utilisateur de la fonction. Il est très important de bien documenter sa fonction en vue de son utilisation ultérieure.

Il faut maintenant créer la fonction **my_3_function** qui sera utilisée en argument : $y(x) = \sin(x) \cdot e^{-i0}$



Figure 628 : my_3_function

Ouvrir un nouveau script et créer la fonction **my_3_function**. **Sauvegarder** cette fonction sous le nom « **my_3_function.m** ».

+1 my_1_function.m × my_2_function.m × my_3_function.m ×
1 function
$$y = my_3_function(x)$$

2 y = sin(x).*exp(-x/10);
3 - end

Figure 629 : my_3_function

Quand une fonction que l'on a codée utilise en argument une autre fonction, il est impératif de placer le signe « @ » devant la fonction qui est utilisée en argument. Pour utiliser la fonction **my_plot_function** avec **my_3_function** en argument, il faut taper la commande suivante dans la fenêtre de commande ou dans un script. La fonction sera tracée sur l'intervalle**[0;60]** et contiendra **1000 points**.

>> my_plot_function(0,60,1000,@my_3_function,'Sinus amorti','Temps en s','sans unité')

On obtient la courbe de la Figure 630.



Figure 630 : appel de la fonction my_plot_function

Si l'on souhaite tracer une autre courbe en gardant la même mise en forme, il suffit de modifier l'équation contenu dans **my_3_function** et de faire un nouvel appel à la fonction **my_plot_function**.

Si l'on souhaite tracer : $y(x) = \sin(x) \cdot \cos^2(x)$, il faut modifier **my_3_function** (Figure 631).

Figure 631 : modification de my_3_function

On fait alors un nouvel appel à **my_plot_function** pour un tracé de la courbe sur l'intervalle **[0;20]** avec **1000 points**.

On obtient alors le graphique de la Figure 632.



Figure 632 : appel de la fonction my_plot_function

Les mises en forme de la Figure 630 et de la Figure 632 sont strictement identiques et permettent d'uniformiser la présentation tout en accélérant le processus de construction des graphiques.

Il est également possible de créer une fonction ne prenant aucun argument en entrée et ne générant aucune variable en sortie. C'est le cas de **my_4_function** (Figure 633) dont le code est donné Figure 634.

my_4_function	
Hello, how are you?	

Figure 633 : my_4_function

-	-2	my_2_function.m × my_3_function.m × my_4_function.m	×
1		- function my_4_function	
2	—	<pre>disp('hello, how are you?')</pre>	
3	—	end	

Figure 634 : codage de my_4_function

En appelant la fonction dans la fenêtre de commande, on obtient juste l'affichage d'un message.

F. Appel d'une fonction à plusieurs arguments

Supposons que l'on veuille trouver les racines du polynôme $y(x) = x^2 - 3$ en utilisant la fonction **my_5_function** (Figure 635) qui donne la définition générale d'un polynôme du second degré. Cette fonction qui prend 4 arguments est codée sur la Figure 636.



Figure 635 : my_5_function



Figure 636 : codage de my_5_function

Ouvrir un nouveau script et créer la fonction **my_5_function**. **Sauvegarder** cette fonction sous le nom « **my_5_function.m** ».

Nous utiliserons la fonction **fzero (fonction,x0)**. La fonction **fzero** (Figure 637) donne la valeur du zéro de la **fonction** qui est donnée en premier argument au voisinage d'une valeur x_0 . La valeur de x_0 est donnée en second argument.



Figure 637 : la fonction fzero

La fonction **fzero** attend en argument une **fonction** à un seul argument (à une seule variable). Or **my_5_function** possède 4 arguments et si rien n'est précisé le calcul ne pourra pas être mené car **fzero** ne pourra pas faire la différence entre ces 4 arguments et ne pourra donc pas trouver la variable.

Il faut donc transformer **my_5_function** en fonction à un seul argument **(x)** en spécifiant tous les autres arguments à une valeur donnée **(a,b,c)**. On utilisera encore le symbole @.

Taper dans la fenêtre de commande les instructions suivantes afin de trouver les racines de $y(x) = x^2 - 3$ au voisinage de x₀=2 et au voisinage de x₀=-2

>> racine1 = fzero(@(x)my_5_function(x,1,0,-3),2)

racine1 =

1.7321

>> racine2 = fzero(@(x)my_5_function(x,1,0,-3),-2)

racine2 =

-1.7321

Le symbole @(x) précise ici que l'unique argument pour la fonction **my_5_function** est la variable x, les autres variables étant fixées à une valeur donnée.

III. Dérivation numérique

A. Aspects théoriques

La Figure 638 montre la courbe représentative de la fonction f(x). La valeur de la dérivée en x_0 de la fonction f(x) notée $f'(x_0)$ est donnée par la pente de la tangente à la courbe représentative de la fonction au point x_0 (repéré 1 sur la figure).



Figure 638 : calcul de la dérivée numérique au point x0

Afin d'avoir une estimation numérique du coefficient directeur de cette tangente, il est nécessaire de choisir 2 points proches de x_0 et de calculer le coefficient directeur de la droite qui passe par ces deux points. Plusieurs choix sont alors possibles pour initier un processus de dérivation numérique.

1. Différence finie progressive

On choisit les points M₀ et M₊ (Figure 639) pour déterminer le coefficient directeur.



Figure 639 : différence finie progressive

Dans ce cas le coefficient directeur de la droite s'exprime par la formule de différence finie progressive :

$$f'(x_0) \simeq \frac{f(x_0 + h) - f(x_0)}{h} \quad \text{(h tend vers 0)}$$

En faisant tendre h vers 0 on se rapproche du coefficient directeur de la tangente à la courbe en x₀.

2. Différence finie rétrograde

On choisit les points M. et M₀ (Figure 640) pour déterminer le coefficient directeur.



Figure 640 : formule de différence finie rétrograde

Dans ce cas le coefficient directeur de la droite s'exprime par la formule de différence finie rétrograde :

$$f'(x_0) \simeq \frac{f(x_0) - f(x_0 - h)}{h} \quad \text{(h tend vers 0)}$$

En faisant tendre h vers 0 on se rapproche du coefficient directeur de la tangente à la courbe en x₀.

3. Différence finie centrée

On choisit les points M₁ et M₂ (Figure 641) pour déterminer le coefficient directeur.



Figure 641 : formule de différence finie centrée

Dans ce cas le coefficient directeur de la droite s'exprime par la formule de différence finie centrée :

$$f'(x_0) \simeq \frac{f(x_0 + \frac{h}{2}) - f\left(x_0 - \frac{h}{2}\right)}{h} \qquad (h \text{ tend vers } 0)$$

En faisant tendre h vers 0 on se rapproche du coefficient directeur de la tangente à la courbe en x₀.

B. Codage en langage MATLAB

Vous pouvez trouver les fichiers de toutes les fonctions présentées dans ce paragraphe dans le dossier **Algorithmique_avec_MATLAB/ Dérivation_numérique.**

L'expression mathématique de la fonction à dériver sera définie dans la fonction **f1.m (**Figure 642). Cette expression peut être modifiée pour tester toutes les fonctions souhaitées. Il faudra juste vérifier que lors de l'appelle de la fonction l'intervalle d'étude [a ; b] appartienne bien au domaine de définition de la fonction.

```
%% f1.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% f1.m contient l'expression mathématique de la fonction à dériver
%%
function y = f1(x)
y = sin(x);
end
```

Figure 642 : codage de la fonction f1 qui contient l'expression de la fonction à dériver

1. Différence finie progressive

Pour calculer la dérivée numérique en utilisant la méthode de la différence fine progressive, nous allons créer la fonction **fderiv_forward** (a,b,nb_points,f) (Figure 643).

Cette fonction prendra en arguments : **a** : borne inférieure de l'intervalle d'étude

b : borne supérieur de l'intervalle d'étude

nb points : nombre de points à calculer

f : fonction à dériver

La fonction aura comme sorties :

t : vecteur contenant les instants correspondant aux points ou la dérivée est calculée

 ${\bf df}$: vecteur contenant les valeurs de la dérivée numérique au différents instants



Figure 643 : fonction fderiv_forward (a,b,nb_point,f)

Afin de pouvoir tracer **df** en fonction de **t**, les deux vecteurs doivent contenir le même nombre d'éléments. Par construction de notre algorithme, le vecteur **t** possède une composante de plus que le vecteur **df**. Il faudra donc supprimer la dernière composante du vecteur **t**. L'objectif sera de comparer les erreurs effectuées lors de l'utilisation des différentes méthodes de dérivation numérique. Afin de pouvoir effectuer cette comparaison, nous devons prendre quelques précautions quant à la génération des sorties des différentes fonctions.

Pour la méthode de différence finie progressive : il n'est pas possible de calculer **df** pour le dernier point de l'intervalle [a ; b] correspondant à $\mathbf{t} = \mathbf{b}$. Il faudrait pour cela la valeur de la fonction en $\mathbf{t} = \mathbf{b} + \mathbf{h}$ (en dehors de l'intervalle d'étude)

Pour la méthode de différence finie rétrograde : il n'est pas possible de calculer **df** pour le premier point de l'intervalle [a ; b] correspondant à $\mathbf{t} = \mathbf{a}$. Il faudrait pour cela la valeur de la fonction en $\mathbf{t} = \mathbf{a} - \mathbf{h}$ (en dehors de l'intervalle d'étude)

Pour la méthode de différence finie centrée : il n'est pas possible de calculer **df** pour le premier point de de l'intervalle [a ; b] correspondant à $\mathbf{t} = \mathbf{a}$ et pour le dernier point de l'intervalle [a ; b] correspondant à $\mathbf{t} = \mathbf{a}$ et pour le dernier point de l'intervalle [a ; b] correspondant à $\mathbf{t} = \mathbf{a}$ et pour le dernier point de l'intervalle [a ; b] correspondant à $\mathbf{t} = \mathbf{b}$. Il faudrait pour cela la valeur de la fonction en $\mathbf{t} = \mathbf{a} - \mathbf{h}/2$ et en $\mathbf{t} = \mathbf{b} + \mathbf{h}/2$ (en dehors de l'intervalle d'étude)

Afin de pouvoir comparer les trois méthodes de dérivation, nous choisirons de travailler avec le même vecteur t pour les trois types de dérivation soit $t \in [a+h; b-h]$. Par construction de nos algorithmes, le dernier point calculé correspondra à $\mathbf{t} = \mathbf{b} - \mathbf{h}$. Le premier point correspondant à $\mathbf{t} = \mathbf{a}$ devra être systématiquement supprimé afin de rendre la comparaison possible.

La Figure 644 montre le codage de la fonction **fderiv_forward** (a,b,nb_points,f).

```
%% [t,df] = fderiv forward(a,b,nb points,f)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul la dérivée numérique de la fonction f sur l'intervalle [a;b] en
% prenant nb points pour le calcul
% La méthode est celle de la différence finie progressive
88
function [t,df] = fderiv_forward(a,b,nb_points,f)
% calcul du pas h
h = (b-a)/nb points;
% défintion du vecteur t
t = [a:h:b];
% calcul de la dérivée en chaque point
for k = 1:1:nb points
   df(k) = (f(t(k+1)) - f(t(k)))/h;
end
% suppression du dernier élément du vecteur t afin d'égaliser le nombre de composantes des
deux vecteurs
t(end) = [];
% suppression du premier point pour éviter les effets de
% bords
df(1) = [];
t(1) = [];
end
```

Figure 644 : codage de la fonction fderiv_forward (a,b,nb_point,f)

2. Différence finie rétrograde

Pour calculer la dérivée numérique en utilisant la méthode de la différence fine rétrograde, nous allons créer la fonction **fderiv_backward** (a,b,nb_points,f) (Figure 645).

Cette fonction prendra en arguments :

a : borne inférieure de l'intervalle d'étude

b : borne supérieur de l'intervalle d'étude

 \mathbf{nb}_{-} points : nombre de points à calculer

f : fonction à dériver

La fonction aura comme sorties :

t : vecteur contenant les instants correspondant aux points ou la dérivée est calculée df : vecteur contenant les valeurs de la dérivée numérique au différents instants



Figure 645 : fonction fderiv_backward (a,b,nb_point,f)

Afin de pouvoir tracer **df** en fonction de **t**, les deux vecteurs doivent contenir le même nombre d'éléments. Par construction de notre algorithme, le vecteur **t** possède une composante de plus que le vecteur **df**. Il faudra donc supprimer la dernière composante du vecteur **t**.

Comme expliqué précédemment nous exploiterons **t** et **df** sur l'intervalle $t \in [a+h; b-h]$. Le premier point correspondant à $\mathbf{t} = \mathbf{a}$ sera supprimé afin de rendre la comparaison entre les différentes méthodes de dérivation significative.

La Figure 646 montre le codage de la fonction fderiv_backward (a,b,nb_points,f).

```
%% [t,df] = fderiv backward(a,b,nb points,f)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul la dérivée numérique de la fonction f sur l'intervalle [a;b] en
% prenant nb points pour le calcul
% La méthode est celle de la différence finie rétrograde
88
function [t,df] = fderiv backward(a,b,nb points,f)
% calcul du pas h
h = (b-a)/nb points;
% défintion du vecteur t
t = [a:h:b];
% calcul de la dérivée en chaque point
for k = 2:1:nb points;
    df(k) = (f(t(k)) - f(t(k-1)))/h;
end
% suppression du dernier élément du vecteur t afin d'égaliser le nombre de composantes des
deux vecteurs
t(end) = [];
% suppression du premier point pour éviter les effets de
% bords
df(1) = [];
t(1) = [];
```

end

Figure 646 : codage de la fonction fderiv_backward (a,b,nb_point,f)

3. Différence finie centrée

Pour calculer la dérivée numérique en utilisant la méthode de la différence finie centrée, nous allons créer la fonction **fderiv_center** (a,b,nb_points,f) (Figure 647).

Cette fonction prendra en arguments :

- **a** : borne inférieure de l'intervalle d'étude
- **b** : borne supérieur de l'intervalle d'étude
- **nb**_points : nombre de points à calculer
- **f** : fonction à dériver

La fonction aura comme sorties :

- t : vecteur contenant les instants correspondant aux points ou la dérivée est calculée
- df : vecteur contenant les valeurs de la dérivée numérique au différents instants



Figure 647 : fonction fderiv_center (a,b,nb_point,f)

Afin de pouvoir tracer **df** en fonction de **t**, les deux vecteurs doivent contenir le même nombre d'éléments. Par construction de notre algorithme, le vecteur **t** possède une composante de plus que le vecteur **df**. Il faudra donc supprimer la dernière composante du vecteur **t**.

Comme expliqué précédemment nous exploiterons **t** et **df** sur l'intervalle $t \in [a+h; b-h]$. Le premier point correspondant à $\mathbf{t} = \mathbf{a}$ sera supprimé afin de rendre la comparaison entre les différentes méthodes de dérivation significative.

La Figure 648 montre le codage de la fonction **fderiv_center** (a,b,nb_points,f).



```
% calcul de la dérivée en chaque point
for k = 2:1:nb_points
    df(k) = (f(t(k)+h/2)-f(t(k)-h/2))/h;
end
% suppression du dernier élément du vecteur t afin d'égaliser le nombre de composantes des
deux vecteurs
t(end) = [];
% suppression du premier point pour éviter les effets de
% bords
df(1) = [];
t(1) = [];
end
```



4. Calcul exact de la fonction dérivée en utilisant le calcul symbolique

Afin de pouvoir évaluer l'erreur induite par les différentes méthodes de dérivation numérique, nous allons créer une fonction **fderiv_ana_symbolique** (a,b,nb_points,f) qui calcule la dérivée analytique exacte d'une fonction en calcul symbolique (Figure 649). Cette dérivée constituera la référence pour calculer l'erreur.

Cette fonction prendra en arguments : **a** : borne inférieure de l'intervalle d'étude **b** : borne supérieur de l'intervalle d'étude **nb**_points : nombre de points à calculer **f** : fonction à dériver

La fonction aura comme sorties :

t : vecteur contenant les instants correspondant aux points ou la dérivée est calculée

df : vecteur contenant les valeurs de la dérivée numérique au différents instants



Figure 649 : fonction fderiv_ana_symbolique (a,b,nb_point,f)

Comme expliqué précédemment nous exploiterons \mathbf{t} et \mathbf{df} sur l'intervalle $t \in [a+h; b-h]$. Il faudra donc supprimer le premier et le dernier point pour pouvoir effectuer la comparaison avec les autres méthodes de dérivation.

La Figure 650 montre le codage de la fonction **fderiv_ana_symbolique** (a,b,nb_points,f).

```
%% [t,df] = fderiv_anasymbolique(a,b,nb_points,f)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul de la dérivée analytique exacte d'une fonction en utilisant le calcul
% symbolique
function [t,df]=fderiv_ana_symbolique(a,b,nb_points,f)
h = (b-a)/nb_points;
```

```
t = [a:h:b];
% x est déclarée comme variable symbolique
svms x;
% création de la fonction symbolique f symbolique à partir de f(x)
f symbolique(x) = f(x);
% dérivation de la fonction f symbolique pour créer la fonction dérivée
% symbolique
df symbolique(x) = diff(f symbolique,x);
% calcul des valeurs du vecteur df qui contient les valeurs de la dérivée
% calculée analytiquement
df = df symbolique(t);
% conversion du vecteur df en double précision
df = double(df);
df(1) = [];
t(1) = [];
df(end) = [];
t(end) = [];
end
```

Figure 650 : codage de la fonction fderiv_ana_symbolique (a,b,nb_point,f)

C. Exploitation de l'algorithme

1. Comparaison des différentes méthodes de dérivation numérique

Afin d'évaluer les écarts entre les différentes méthodes de dérivation numériques et la dérivée analytique calculée en calcul symbolique, nous allons créer un premier script qui va tracer ces 4 courbes dans la même fenêtre graphique, puis quantifier et tracer l'amplitude comparée de ces écarts.

Ouvrir le script num_deriv_plot_comparaison.m.

Le script est structuré en trois sections (Figure 651) :

- Définition de l'intervalle d'étude de la fonction [a ;b] et du nombre de points **nb_points** à calculer pour tracer les courbes
- Appel des différentes fonctions et tracé des différentes courbes sur la même fenêtre graphique
- Evaluation des écarts entre la dérivée analytique et les trois méthodes de dérivation numérique et tracé des écarts sur la même fenêtre graphique

```
%% num_deriv_plot_comparison
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
%% Calcul de la dérivée numérique d'une fonction en utilisant différentes
% méthodes de calcul:
% - différence finie centrée
% - différence finie rétrograde
% - différence finie progressive
% Une comparaison est effectuée avec le calcul de la dérivée théorique
% exacte effectuée en calcul symbolique
% Toutes les courbes sont placées sur le même graphiques afin de pouvoir
% évaluer visuellement les erreurs dues au processus de dérivation
% numérique
% Il est possible de faire varier l'intervalle [a;b] d'étude de la fonction
% et le nombre de points nb points que l'on souhaitent pour effectuer le
% calcul
% La fonction à dériver est stockée dans la fonction f1.m
%% Script
```

```
% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 10;
% définition du nombre de point à utiliser pour le calcul
nb points = 40;
%% Calcul et tracé de la dérivée numérique avec les différentes méthode
% création et positionnement de la fenêtre graphique
figure('Units','centimeters','Position',[2 8 18 12]);
grid on; grid minor; hold on;
 % calcul et tracé de la dérivée exacte en calcul symbolique
[t_ana_symbolique,df_num_ana_symbolique] = fderiv_ana_symbolique(a,b,nb_points,@f1);
plot(t_ana_symbolique,df_num_ana_symbolique,'red','LineWidth',2);
% calcul et tracé de la dérivée avec différence finie centrée
[t_center,df_num_center] = fderiv_center(a,b,nb_points,@f1);
plot(t center, df num center, 'green', 'LineWidth', 2);
% calcul et tracé de la dérivée avec différence finie progressive
[t forward, df num forward] = fderiv forward(a, b, nb points, @f1);
plot(t forward, df num forward, 'blue', 'LineWidth', 2);
% calcul et tracé de la dérivée avec différence finie retrograde
[t backward,df num backward] = fderiv_backward(a,b,nb_points,@f1);
plot(t backward,df num backward,'cyan','LineWidth',1);
% titre et légende
title({'Comparaison des méthodes de dérivation numérique';...
     de la fonction f1'});
legend('dérivée analytique', 'différence finie centrée',...
    'différence finie progressive', 'différence fine rétrograde')
xlim([a b]);
%% Calcul des différents écarts en prenant comme référence la dérivée
% exacte déterminée en calcul symbolique
error_center = df_num_ana_symbolique - df_num_center;
error_forward = df_num_ana_symbolique - df_num_forward;
error_backward = df_num_ana_symbolique - df_num_backward;
% création du vecteur des abscisses
t = [a:(b-a)/nb_points:b];
t(1) = [];
t(end) = [];
% création et positionnement de la fenêtre graphique
figure('Units','centimeters','Position',[22 8 18 12]);
hold on; grid on; grid minor;
% tracé des différents écarts
plot(t,error_center,'green','LineWidth',2)
plot(t,error_forward,'blue','LineWidth',2)
plot(t,error backward,'cyan','LineWidth',2)
% titre et légende
title({'Erreurs comparées des différentes méthodes de dérivation numérique';...
    'par rapport à la dérivée analytique'});
legend ('différence finie centrée', 'différence finie progressive',...
    'différence fine rétrograde')
```

xlim([a b]);

Figure 651 : comparaison des différentes méthodes de dérivation numérique

Dans un premier temps le script sera exécuté sur l'intervalle [0;10] avec 40 points calculés. La fonction définie dans **f1.m** est la fonction f(x) = sin(x). Cette fonction peut-être modifiée par l'utilisateur.

L'exécution du script permet de visualiser les résultats du calcul numérique de la dérivée avec les différentes méthodes de calcul (Figure 652, Figure 653 et Figure 654).



Figure 652 : comparaison des différentes méthodes de dérivation numérique



Figure 653 : zoom sur la courbe



Figure 654 : visualisation des erreurs induites par les différentes méthodes de dérivation numérique

L'analyse des différents courbes nous permet de dégager les tendances suivantes :

- La différence finie progressive est en avance sur la dérivée théorique exacte
- La différence finie rétrograde est en retard sur la dérivée théorique exacte
- Les erreurs induites par les méthodes de différences finies progressive et rétrograde semblent être du même ordre de grandeur
- L'erreur induite par la méthode de différence finie centrée semble significativement plus faible

2. Quantification et visualisation de l'erreur induite par les différentes méthodes

Il serait utile d'avoir une idée de l'évolution de cette erreur en fonction du nombre de point calculés et donc en fonction du pas $h = \frac{b-a}{nb}$.

Pour cela ouvrir le script num_deriv_plot_comparaison_subplot.m.

Ce script permet de tracer 4 courbes avec un nombre de points calculés différents pour chaque courbe et de voir ainsi l'évolution de l'erreur en fonction du nombre de points et du pas h choisi pour le calcul. Les 4 courbes correspondent à 4 valeurs du nombre de points calculés choisis équitablement réparties entre les variables **nb_points_min** et **nb_points_max** définies en début de script.

Le script est structuré en deux sections (Figure 655):

- Définition de l'intervalle d'étude de la fonction [a ;b] et du nombre de points **nb_points_min** et **nb_points_max**
- Appel des différentes fonctions et tracé des différentes courbes sur 4 fenêtres graphiques différentes

```
%% num_deriv_plot_comparison_subplot
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul de la dérivée numérique d'une fonction en utilisant différentes
% methodes de calcul:
 - différence finie centrée
% - différence finie rétrograde
% - différence finie progressive
% Une comparaison est effectuée avec le calcul de la dérivée théorique
% exacte effectuée en calcul symbolique
% Il est possible de faire varier l'intervalle [a;b] d'étude de la fonction
% Représentation de 4 courbes correspondant à 4 valeurs du nombres de
% points calculés choisies équitablement réparties entre nb points min et
% nb points max
% La fonction à dériver est stockée dans la fonction fl.m
%% Script
% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 10;
% spécification des nombres de points mini et maxi
nb points min = 10;
nb points max = 50;
pas = round((nb_points_max - nb_points_min) / 3);
fig = figure('Units','centimeters','Position',[2 5 35 25]);
set(fig, 'Color', 'white');
sqtitle({'Comparaison des différentes méthodes de dérivation numérique';...
    ' de la fonction f1 en fonction du nombre de points calculés'});
% initialisation du compteur de boucle
```

```
k = 1;
for nb points = nb points min : pas : nb points max;
subplot(2,2,k); hold on; grid on; grid minor;
% calcul et tracé de la dérivée exacte en calcul symbolique
[t_ana_symbolique,df_num_ana_symbolique] = fderiv_ana_symbolique(a,b,nb_points,@f1);
plot(t_ana_symbolique,df_num_ana_symbolique,'red','LineWidth',2);
% calcul et tracé de la dérivée avec différence finie centrée
[t center,df num_center] = fderiv_center(a,b,nb_points,@f1);
plot(t center, df num center, 'green', 'LineWidth', 2);
% calcul et tracé de la dérivée avec différence finie progressive
[t forward,df_num_forward] = fderiv_forward(a,b,nb_points,@f1);
plot(t forward,df num forward,'blue','LineWidth',2);
% calcul et tracé de la dérivée avec différence finie retrograde
[t_backward,df_num_backward] = fderiv_backward(a,b,nb_points,@f1);
plot(t_backward,df_num_backward,'cyan','LineWidth',1);
% legend('dérivée analytique','différence finie centrée',...
     'différence finie progressive', 'différence fine rétrograde');
title('Nombre de points calculés ' + string(nb points))
k = k + 1;
end
% spécification de la légende générale du graphique
leg = legend('dérivée analytique','différence finie centrée',...
    'différence finie progressive', 'différence fine rétrograde');
% Positionnement de la légende au centre du graphique
newPosition = [0.45 0.47 0.1 0.07];
newUnits = 'normalized';
set(leg, 'Position', newPosition, 'Units', newUnits);
```

Figure 655 : script permettant de visualiser l'influence du pas

L'exécution du script permet de visualiser l'influence du pas sur l'allure des courbes (Figure 656).



Figure 656 : influence du pas sur l'allure des courbes

L'analyse de ces courbes nous confirme que l'erreur va diminuer avec le pas mais ne nous permet pas de connaître la loi d'évolution de cette erreur en fonction du pas h. Lors de l'utilisation d'un processus de dérivation numérique, il est important de déterminer cette loi d'évolution.

3. Détermination de la loi d'évolution de l'erreur en fonction du pas

Pour connaître la loi d'évolution de cette erreur, il suffit de se placer en un point x_0 arbitraire de l'intervalle et de calculer l'erreur pour différentes valeurs du pas h. L'erreur en fonction du pas h se définie de la manière suivante :

• Pour la différence finie progressive :

$$f_{err_forward}(h) = \left| f'(x_0) - \frac{f(x_0 + h) - f(x_0)}{h} \right|$$

• Pour la différence finie rétrograde :

$$f_{err_backward}\left(h\right) = \left|f'\left(x_0\right) - \frac{f\left(x_0\right) - f\left(x_0 - h\right)}{h}\right|$$

• Pour la différence finie centrée :

$$f_{err_center}(h) = \left| f'(x_0) - \frac{f(x_0 + h/2) - f(x_0 - h/2)}{h} \right|$$

L'utilisation de la formule de Taylor pourrait nous permettre de trouver la forme mathématique de ces fonctions erreurs. L'objectif ici est plutôt d'exploiter notre algorithme et les capacités de calcul de MATLAB afin de déterminer cette forme.

Pour cela nous allons construire un script qui se placera arbitrairement au centre de notre intervalle d'étude et qui va tracer automatiquement l'erreur en fonction du pas h. Le script contient une boucle qui calculera les différentes fonctions erreurs pour un nombre de points allant de 100 à 1000 avec un

incrément de 100. Le pas h est défini par $h = \frac{b-a}{nb_{-} points}$.

La boucle permettra donc de calculer 10 valeurs de l'erreur au point x_0 pour chacune des trois méthodes de dérivation numérique. Le script permet ensuite d'interpoler les points calculés avec une fonction polynomiale adaptée pour trouver l'ordre de variation des fonctions erreurs.

Ouvrir le script num_deriv_plot_error.m (Figure 657).

```
%% num_deriv_plot_error
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul de la forme mathémathique de l'erreur effectuée lors des
% opérations de dérivation numériques pour différentes méthodes de calcul:
% - différence finie centrée
% - différence finie rétrograde
% - différence finie progressive
% Une comparaison est effectuée avec le calcul de la dérivée théorique
% exacte effectuée en calcul symbolique
% On se place arbitrairement au centre de l'intervalle d'étude de la
% fonction en un point x0 afin d'évaluer l'erreur en ce point pour
% différentes valeurs de h
% et du nombre de points de calcul
% on utilise ensuite une interpolation polynômiale pour déterminer la forme
% mathématique de l'erreur en fonction de h
% La fonction à dériver est stockée dans la fonction fl.m
%% Script
% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 10;
% initialisation du compteur qui servira d'indice
k = 1;
%% On définit ici une boucle qui permettra de calculer l'erreur pour un
% échantillon de points allant de 100 points jusqu'à 1000 points par pas
% de 100 pour toutes les méthodes de dérivation
for nb points = 100:100:1000
% calcul de la dérivée exacte en utilisant le calcul symbolique
[t ana symbolique,df num ana symbolique] = fderiv ana symbolique(a,b,nb points,@f1);
% calcul de la dérivée avec différence finie centrée
[t center,df num center] = fderiv center(a,b,nb points,@f1);
% calcul et tracé de la dérivée avec différence finie rétrograde
```

```
[t backward,df_num_backward] = fderiv_backward(a,b,nb_points,@f1);
% calcul et tracé de la dérivée avec différence finie progressive
[t forward,df num forward] = fderiv forward(a,b,nb points,@f1);
% calcul des erreurs au point x0, milieu de l'intervalle d'étude en prenant
% comme référence la dérivée exacte déterminée en calcul symbolique
x0 = round(nb_points/2);
error_f_center_h(k) = df_num_ana_symbolique(x0) - df_num_center(x0);
error_f_forward_h(k) = df_num_ana_symbolique(x0) - df_num_forward(x0);
error_f_backward_h(k) = df_num_ana_symbolique(x0) - df_num_backward(x0);
% détermination de la valeur du pas h(k) pour l'itération en cours
h(k) = (b-a)/nb points;
% incrémentation de k
k = k + 1;
end
%% Tracé et interpolation de l'erreur pour la différence finie centrée
% Création d'une nouvelle figure pour les graphiques
figure('Units','centimeters','Position',[2 8 18 12]);
hold all; grid on; grid minor;
plot(h,error_f_center_h,'sm','MarkerSize',10,'LineWidth',2);
% Calcul des coefficients du polynôme d'interpolation
coeff poly inter center = polyfit (h,error f center h,2);
% Tracé du polynôme d'interpolation
t = linspace(0, h(1), 100);
poly inter center = polyval(coeff poly inter center,t);
plot(t,poly_inter_center, 'LineWidth', 2);
% Ecriture de la légende
legend('différence finie centrée',...
     string(coeff poly inter center(3)) + ' h^2 + ' ...
    + string(coeff_poly_inter_center(2)) + ' h + ' ...
     + string(coeff poly inter center(1)),...
    'Location', 'northwest', 'FontSize',12);
% Ecriture du titre et des étiquettes des axes
title({'Variation de l''erreur en fonction du pas h' ;...
    'pour la différence finie centrée'});
xlabel('pas h');
ylabel('erreur');
%% Tracé et interpolation de l'erreur pour la différence finie progressive
figure('Units','centimeters','Position',[22 8 18 12]);
hold all; grid on; grid minor;
plot(h,error_f_forward_h,'sb','MarkerSize',10,'LineWidth',2)
% Détermination du polynome d'interpolation pour la différence finie
% progressive
% Calcul des coefficients du polynôme d'interpolation
coeff_poly_inter_forward = polyfit (h,error_f_forward_h,1);
% Tracé du polynôme d'interpolation
t = linspace(0, h(1), 100);
poly inter forward = polyval(coeff poly inter forward,t);
plot(t,poly_inter_forward, 'LineWidth',2);
%% Tracé et interpolation de l'erreur pour la différence finie rétrograde
plot(h,error f backward h, 'og', 'MarkerSize', 10, 'LineWidth', 2)
```

```
% Détermination du polynôme d'interpolation pour la différence finie
% rétrograde
% Calcul des coefficients du polynôme d'interpolation
coeff poly inter backward = polyfit (h,error f backward h,1);
% Tracé du polynôme d'interpolation
t = linspace(0,h(1),100);
poly_inter_backward = polyval(coeff_poly_inter_backward,t);
plot(t,poly_inter_backward, 'LineWidth',2);
% écriture de la légende
legend('différence finie progressive',...
       string(coeff_poly_inter_forward(1)) + ' h + ' ...
     + string(coeff_poly_inter_forward(2)) ,...
       'différence finie rétrograde',...
     string(coeff poly inter backward(1)) + ' h + ' ...
     + string(coeff_poly_inter_backward(2)),...
     'Location', 'northwest', 'FontSize', 12);
% titre et étiquette des axes
title({'Variation de l''erreur en fonction du pas h' ;...
    'pour les différences finies progressive et retrograde'});
xlabel('pas h');
ylabel('erreur');
```



Exécutez le script et analyser les courbes obtenues sur la Figure 658 et sur la Figure 659.



Figure 658 : variation de l'erreur de dérivation en fonction de h pour la différence finie centrée



Figure 659 : variation de l'erreur de dérivation en fonction de h pour la différence finie progressive et rétrograde

En observant les points qui représentent l'erreur en fonction du pas h, nous choisissons une régression linéaire pour la différence finie progressive et rétrograde et une régression polynomiale d'ordre 2 pour la différence finie centrée. Il apparait que ces choix permettent d'interpoler la série de données avec une excellente précision. Les équations des polynômes d'interpolation apparaissent dans les légendes des différentes fenêtres graphiques.

En conclusion :

- l'erreur pour la différence finie progressive et pour la différence finie rétrograde est d'ordre 1 en h
- l'erreur pour la différence finie centrée est d'ordre 2 en h

L'exploitation d'un algorithme peut donc nous amener à connaître la tendance de variation d'un paramètre en exploitant la puissance de calcul et en utilisant les outils adaptés. Notre seule intuition est parfois insuffisante et nous devons nous appuyer sur le potentiel de notre algorithme. Ces tendances peuvent évidemment être démontrées mathématiquement en utilisant la formule de Taylor qui confirmerait ces observations.

Les concepts que nous avons implantés dans notre algorithme sont très simples. Une exploitation pertinente de ces concepts nous permet de dégager des tendances que seuls des concepts mathématiques avancés peuvent prévoir.

4. Robustesse du processus

Afin de tester la robustesse de notre algorithme, nous pouvons tester plusieurs fonctions en modifiant la fonction **f1.m** et en reproduisant le processus.

Modifier la fonction **f1.m** pour reproduire l'analyse sur une fonction polynomiale $y(x) = 5x^3 - 3x^2 + 5x + 1$ (Figure 660).

```
function y = f1(x)
y = 5*x^3-3*x^2+5*x+1;
end
```



Relancer les différents scripts pour visualiser les résultats dans les différentes figures.



Figure 661 : comparaison des différentes méthodes de dérivation numérique



Figure 662 : visualisation des erreurs induites par les différentes méthodes de dérivation numérique



Figure 663 : influence du pas sur l'allure des courbes



Figure 664 : variation de l'erreur de dérivation en fonction de h pour la différence finie centrée



Figure 665 : variation de l'erreur de dérivation en fonction de h pour la différence finie progressive et rétrograde

D. Application de la dérivation numérique

1. Dérivation du signal issu d'un codeur incrémental

Le signal relevé sur la Figure 666 (zoom sur la Figure 667) représente la position mesurée à l'aide d'un codeur incrémental placé sur l'arbre d'un moteur. Le signal une fois traité présente de nombreuses discontinuités qui seront très préjudiciables lors d'un processus de dérivation numérique. Si nous voulons obtenir la vitesse de rotation de l'arbre du moteur en fonction du temps, il faudra dériver numériquement le signal mais il ne sera pas possible de le faire sur le signal brut.



Figure 666 : mesure de la position à partir du signal d'un codeur incrémental



Figure 667 : zoom sur la courbe

2. Mise en évidence des problèmes rencontrés

Lorsqu'un signal présente des discontinuités importantes ou lorsqu'il est fortement bruité, le processus de dérivation numérique simple ne pourra pas donner un résultat satisfaisant. En effet le Figure 668 montre la variation très forte des coefficients directeurs calculés par la méthode des différences finies sur les intervalles (1, 2), (3) et (4). Sur les intervalles (1) et (2) la dérivée numérique est nulle alors qu'elle est très élevée sur l'intervalle (3).



Figure 668 : influence du pas d'échantillonnage sur le calcul de la dérivée numérique

L'application directe d'un algorithme de dérivation numérique sur le signal brut nous donne le résultat visible sur la Figure 669. Le signal obtenu montre les variations intempestives du coefficient directeur calculé sur une distance correspondant au pas h. Cette méthode n'est donc pas satisfaisante.



Figure 669 : dérivation numérique du signal brut

3. Dérivation numérique à pas variable

Afin d'obtenir une allure correcte pour notre signal obtenu par dérivation numérique, nous allons allonger le pas de dérivation afin de limiter au maximum les variations intempestives de la dérivé.



Figure 670 : influence de l'augmentation du pas de dérivation

Pour dériver un signal avec pas variable nous allons utiliser la fonction **fderiv_num_pas_variable(t,Y,step_size)** (Figure 671).

Cette fonction prendra en arguments :

- t : vecteur contenant les abscisses du signal à dériver
- Y : vecteur contenant les valeurs du signal à dériver
- **step_size_**points : nombre de points à considérer pour construire le pas de dérivation

La fonction aura comme sorties :

- **t_step** : vecteur contenant les instants correspondant aux points ou la dérivée est calculée
- **dY_step** : vecteur contenant les valeurs de la dérivée numérique au différents instants



Figure 671 : fonction fderiv_num_pas_variable(t,Y,step_size)

Le codage de la fonction fderiv_num_pas_variable(t,Y,step_size)

```
%% fderiv pas variable(t,Y,step size)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction permet de réaliser une dérivation numérique en choisissant
% la longueur du pas de dérivation
% step size représente la longueur du pas de dérivation
function [t step,dY step] = fderiv num pas variable(t,Y,step size)
% Evaluation du nombre de points des vecteurs
nb points = length(t);
% calcul du pas h
h = (t(end) - t(1)) / (nb points - 1);
% calcul du pas à utiliser pour l'opération de dérivation numérique
pas = step size * h;
% calcul de la dérivée en chaque point
i = 1;
% construction des vecteurs t step et dY step
for k = 1:step size:(nb points-1-step size)
    dY step(i) = (Y(k+step size)-Y(k))/pas;
    t step(i) = t(k+step size);
    i = i + 1;
end
end
```

Figure 672 : codage de la fonction fderiv_num_pas_variable(t,Y,step_size)

Afin de tester cette fonction et de voir l'influence du pas de dérivation sur la forme du signal, ouvrir le script **deriv_num_pas_variable_comparaison.m**. Ce script permet de tracer dans une même fenêtre graphique 4 courbes représentant la dérivée numérique calculée avec 4 valeurs du pas de dérivation. Il est possible de modifier les variables **nb_points_min** et **nb_points_max**.

```
%% deriv num pas variable comparaison
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Représentation de 4 courbes correspondant à 4 dérivations numériques du
% signal en considérant différents pas de dérivation choisis équitablement
% répartis entre nb points min et nb points max
%% Effacement des variables du workspace
clear all;
% chargement dans le workspacedes données correspondant au signal du codeur et au
temps
load num deriv data.mat;
% tracé du signal du codeur
fig = figure('Units','centimeters','Position',[2 5 35 25]);
set(fig, 'Color', 'white');
sqtitle({'Position donnée par le codeur en fonction du temps';...
    ''});
plot(t,position codeur);
grid on; grid minor;
% spécification des nombres de points mini et maxi
nb points min = 50;
```

```
nb points max = 450;
pas = round((nb points max - nb points min) / 3);
88
fig = figure('Units','centimeters','Position',[2 5 35 25]);
set(fig, 'Color', 'white');
sgtitle({'Analyse de l''influence du pas de dérivation';...
    ' sur la forme du signal à dériver'});
% initialisation du compteur de boucle
k = 1;
for step_size = nb_points_min : pas : nb_points_max;
subplot(2,2,k); hold on; grid on; grid minor;
% calcul et tracé de la dérivée exacte en calcul symbolique
[t_step,dY_step] = fderiv_num_pas_variable(t,position_codeur,step_size);
plot(t step, dY step, 'blue', 'LineWidth', 2);
title('La longueur du pas couvre ' + string(step size) + ' points')
k = k + 1;
end
% spécification de la légende générale du graphique
leg = legend('signal dérivé');
% Positionnement de la légende au centre du graphique
newPosition = [0.45 0.47 0.1 0.07];
newUnits = 'normalized';
set(leg, 'Position', newPosition, 'Units', newUnits);
```

Figure 673 : script permettant de faire varier le pas de dérivation

Exécuter le script et observer le résultat sur la Figure 674.



Analyse de l'influence du pas de dérivation sur la forme du signal à dériver

Figure 674 : influence du pas de dérivation numérique

Nous pouvons constater que l'augmentation du pas de dérivation permet d'obtenir un signal dérivé exploitable. A partir de 300 points le résultat est satisfaisant.

Cette application montre que la dérivation numérique est une opération délicate dans le cas de signaux provenant de capteurs, qui peuvent être fortement bruité et présenter des discontinuités.

Il est également possible de filtrer le signal avant de le dériver pour éliminer le bruit et les discontinuités.
IV. Résolution de l'équation f(x)=0

Vous pouvez trouver les fichiers de toutes les fonctions et scripts présentés dans ce paragraphe dans le dossier **Algorithmique_avec_MATLAB/ Résolution f(x)=0**

A. Dichotomie

<u>Principe de la méthode</u> : On part de deux valeurs a et b encadrant une solution unique de l'équation f(x)=0. A chaque itération, l'intervalle est divisé en deux parties égales en conservant l'intervalle qui contient la racine.

- Déterminer le point milieu de l'intervalle : $m = \frac{a+b}{2}$
- Evaluer f(a).f(m)
- Si $f(a) \cdot f(m) > 0$, l'intervalle [a ; m] ne contient pas la racine. La racine se trouve dans l'intervalle [m ; b] et a prend la valeur de m pour la prochaine itération.
- Si f(a).f(m) < 0, l'intervalle [a ; m] contient la racine et b prend la valeur de m pour la prochaine itération.

Le processus s'arrête quand l'intervalle (b-a) devient inférieur à epsilon qui représente la précision que l'on souhaite obtenir dans l'évaluation de la solution. Afin de tester la condition de fin de notre algorithme, nous utiliserons une boucle « tant que ».

La Figure 675 illustre les 2 premières itérations de la méthode de recherche des solutions de l'équation f(x)=0 par la méthode de dichotomie. La solution de l'équation f(x)=0 est également appelée « racine ».



Figure 675 : illustration de la recherche des solutions de l'équation f(x)=0 par la méthode de dichotomie

Afin de mettre en place le processus de recherche de solution par dichotomie, nous allons créer la fonction **fdichotomie(f,a,b,epsilon)** (Figure 676) qui calcule la solution de l'équation f(x)=0 dans l'intervalle **[a ; b]** avec une précision **epsilon**.

Cette fonction prendra en arguments :

f : fonction dont on recherche la solution de l'équation f(x)=0

a : borne inférieure de l'intervalle d'étude

b : borne supérieur de l'intervalle d'étude

epsilon : précision souhaitée pour la processus de recherche de la solution

La fonction aura comme sortie :

sol: solution de l'équation f(x)=0 dans l'intervalle [a ; b] avec une précision epsilon



Figure 676 : fonction fdichotomie(f,a,b,epsilon)

La Figure 677 montre le codage de la fonction **fdichotomie(f,a,b,epsilon)**.

```
%% fdichotomie.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction renvoie la solution de l'équation f(x)=0 recherchée dans
% l'intervalle [a;b] avec une précision epsilon en utilisant la méthode de
% dichotomie
function sol = fdichotomie(f,a,b,epsilon)
while (b-a) > epsilon
   m = (a+b)/2;
    if f(a) * f(m) > 0
        a = m;
    else
        b = m;
    end
    end
sol = (a+b)/2;
end
```

Figure 677 : codage de la fonction fdichotomie(f,a,b,epsilon)

Nous souhaitons résoudre l'équation f(x)=0 avec $f(x)=3x^2+3x-15$. Cette fonction est stockée dans **f2.m** dont le codage est décrit sur la Figure 678.

```
%% f2.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% f2.m contient l'expression mathématique de la fonction dont on recherche
% la solution de type f(x)=0
function y=f2(x)
y = 3.*x.^2+3.*x-15;
```

Figure 678: fonction f2.m

Afin de tester la fonction **fdichotomie**, il faut créer un script qui va appeler la fonction **fdichotomie** pour résoudre l'équation stockée dans la fonction **f2.m**.

Ouvrir le script **dichotomie.m**.

Ce script va dans un premier temps tracer la courbe représentative de la fonction et demander à l'utilisateur de lui donner un intervalle **[a ; b]** et une précision de recherche **epsilon** afin de démarrer la méthode de résolution. Le script donnera également le temps d'exécution de la fonction.

```
%% dichotomie.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Recherche par dichotomie de la solution de l'équation f(x)=0 dans
% l'intervalle [a;b] avec une précision espilon
% La courbe y = f(x) est tracée
% L'utilisateur peut alors spécifier l'intervalle [a;b] de recherche de la
% solution et la précision epsilon souhiatée
% la fonction f(x) est stockée dans la fonction f2.m
%% Tracé de la fonction dont on cherche les racines par dichotomie
figure;
ezplot(@f2);
grid on; grid minor;
% affichage des axes sur le graphique
ax = gca;
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';
%% Spécification de l'intervalle [a;b] de recherche de la solution f(x)=0 et de
la précision souhaitée espilon
disp('La solution de l''équation f(x)=0 est recherchée par dichotomie dans
l''intervalle [a;b]');
a = input('Saisir la valeur de a: ');
b = input('Saisir la valeur de b: ');
disp('La recherche de la solution se fera avec une précision epsilon: ');
eps = input('Spécifier la valeur de epsilon: ');
%% Appel de la fonction fdichotomie et affichage du résultat
% démarrage du compteur pour évaluer le temps d'exécution de la fonction
tic:
% appel de la fonction fdichotomie
sol = fdichotomie(@f2,a,b,eps);
% création de la variable t dicho contenant le temps d'exécution de la
% fonction
t dicho = toc;
% affichage de la solution et du temps nécessaire à l'exécution de la
% fonction
```

end

```
reponse = ['La solution de l''équation f(x)=0 dans l''intervalle
[',num2str(a),';',num2str(b),'] est: ', num2str(sol)];
disp(reponse);
temps = ['Le temps nécessaire au calcul est de ',num2str(t_dicho),' secondes'];
disp(temps);
```

Figure 679 : recherche de solution par dichotomie

Exécutez le script. La courbe représentative de la fonction apparait (Figure 680).



Figure 680 : courbe représentative de la fonction f(x)

Observez la courbe représentative de la fonction et indiquez l'intervalle **[a; b]** dans lequel vous souhaitez rechercher les racines et la précision **epsilon** souhaitée (Figure 681).

La solution de l'équation f(x)=0 est recherchée par dichotomie dans l'intervalle [a;b] Saisir la valeur de a: -4 Saisir la valeur de b: -2 La recherche de la solution se fera avec une précision epsilon: Spécifier la valeur de epsilon: 0.001 La solution de l'équation f(x)=0 dans l'intervalle [-4;-2] est: -2.7915 Le temps nécessaire au calcul est de 0.0040617 secondes

Figure 681 : exécution du script dichotomie.m pour la recherche d'une racine dans l'intervalle[-4 ; 2]

Vous pouvez relancer le script pour chercher une solution dans un autre intervalle (Figure 682).

La solution de l'équation f(x)=0 est recherchée par dichotomie dans l'intervalle [a;b] Saisir la valeur de a: 0 Saisir la valeur de b: 2 La recherche de la solution se fera avec une précision epsilon: Spécifier la valeur de epsilon: 0.001 La solution de l'équation f(x)=0 dans l'intervalle [0;2] est: 1.7915 Le temps nécessaire au calcul est de 0.0012746 secondes

Figure 682 : exécution du script dichotomie.m pour la recherche d'une racine dans l'intervalle[0 ; 2]

B. Méthode de Newton

<u>Principe de la méthode</u> : Cette méthode consiste à approximer la courbe représentative de f(x) par sa tangente en un point x_i . Le processus démarre avec le choix d'un point x_0 choisi proche de la solution recherchée. On cherche ensuite l'intersection de cette tangente avec l'axe des abscisses pour trouver un nouveau point x_1 . On cherche alors la tangente à la courbe représentative de f(x) au point x_1 et on recherche de la même manière le point x_2 . Les x_i définissent alors une suite qui converge vers la solution de l'équation f(x)=0. La solution est alors trouvée par itérations successives avec une précision epsilon spécifiée.

La Figure 683 illustre les deux premières itérations de la méthode de Newton.



L'équation de la tangente au point x_0 à la courbe représentative de f(x) est donnée par :

$$T_0(x) = f(x_0) + f'(x_0)(x - x_0)$$

Première itération : On cherche l'intersection de la tangente T₀ avec l'axe des abscisses en cherchant la solution x₁ tel que T₀(x₁) = 0 :

$$T_0(x_1) = f(x_0) + f'(x_0)(x_1 - x_0) = 0$$

$$\Rightarrow x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

-
- ième itérations : On cherche l'intersection de la tangente T_i avec l'axe des abscisses en cherchant x_{i+1} tel que $T_i(x_{i+1}) = 0$:

$$T_{i}(x_{i+1}) = f(x_{i}) + f'(x_{i})(x_{i+1} - x_{i}) = 0$$

$$\Rightarrow \boxed{x_{i+1} = x_{i} - \frac{f(x_{i})}{f'(x_{i})}} \quad \text{(relation de récurrence à programmer)}$$

A chaque itération, il faudra évaluer les valeurs en x_i de la fonction f et de sa dérivée. Il faudra donc prévoir dans la fonction **fNewton.m** que nous allons définir un processus de dérivation numérique.

Le processus de recherche de la solution s'arrêtera lorsque le critère de précision sera atteint, nous utiliserons une boucle « tant que ».

Afin de mettre en place le processus de recherche de solution par la méthode de Newton, nous allons créer la fonction **fNewton(f,x0,epsilon)** (Figure 684) qui calcule la solution de l'équation f(x)=0 autour de x_0 avec une précision **epsilon**.

Cette fonction prendra en arguments : **f** : fonction dont on recherche la solution de l'équation f(x)=0 **x**₀ : valeur autour de laquelle on recherche la solution **epsilon** : précision souhaitée pour la recherche de la solution

La fonction aura comme sortie :

sol: solution de l'équation f(x)=0 dans l'intervalle [a ; b] avec une précision epsilon



Figure 684 : fonction fNewton(f,x0,epsilon)

La Figure 685 montre le codage de la fonction **fNewton(f,x₀,epsilon)**.

```
%% fNewton.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction renvoie la solution de l'équation f(x)=0 recherchée
% autour de la valeur x0 avec une précision epsilon en utilisant la
% méthode de Newton
% La fonction derive=df(f,x) permet d'obtenir la dérivée numérique de
% la fonction f avec un pas h
function [sol,N] = Newton(f,x0,epsilon)
sol = x0 - f(x0)/df(f,x0);
% tant que la précision n'est pas atteinte, on continue la méthode
while abs(f(sol)) > epsilon
    sol = sol - f(sol)/df(f, sol);
end
end
```

```
% fonction permettant de calculer la dérivée numérique de la fonction
% (différence finie progressive)
function derive = df(f,x)
h = 1e-5;
derive = (f(x+h/2)-f(x-h/2))/h;
end
```

Figure 685 : codage de la fonction fNewton(f,x₀,epsilon)

Nous souhaitons résoudre l'équation f(x)=0 avec $f(x)=3x^2+3x-15$. Cette fonction est stockée dans **f2.m** dont le codage est décrit sur la Figure 686.

```
%% f2.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% f2.m contient l'expression mathématique de la fonction dont on recherche
% la solution de type f(x)=0
function y=f2(x)
y = 3.*x.^2+3.*x-15;
end
```

Figure 686: fonction f2.m

Afin de tester la fonction **fNewton**, il faut créer un script qui va appeler la fonction **fNewton** pour résoudre l'équation stockée dans la fonction **f2.m**.

Ouvrir le script Newton.m.

Ce script va dans un premier temps tracer la courbe représentative de la fonction et demander à l'utilisateur de lui donner une valeur de x_0 et une précision de recherche **epsilon** afin de démarrer la méthode de résolution. Le script donnera également le temps d'exécution de la fonction.

```
%% Newton.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Recherche par la méthode de Newton de la solution de l'équation f(x)=0
% autour de la valeur x0 avec une précision espilon
La courbe y = f(x) est tracée
% L'utilisateur peut alors spécifier la valeur de x0 permettant de démarrer
% le processus de recherche de solution ainsi que la précision epsilon
% souhaitée
%% Tracé de la fonction dont on cherche les racines par la méthode de Newton
figure;
ezplot(@f2);
grid on; grid minor;
% affichage des axes sur le graphique
ax = gca;
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';
%% Spécification de la valeur de x0 permettant de démarrer le processus de
% recherche de la solution f(x)=0 et de la précision souhaitée espilon
disp(La solution de l''équation f(x)=0 est recherchée par la méthode de Newton
autour de la valeur x0');
```

```
x0 = input('Saisir la valeur de x0: ');
disp('La recherche de la solution se fera avec une précision epsilon: ');
eps = input('Spécifier la valeur de epsilon: ');
%% Appel de la fonction fNewton et affichage du résultat
tic;
sol = fNewton(@f2,x0,eps);
t_dicho = toc;
reponse = ['La solution de l''équation f(x)=0 autour de x=',num2str(x0),' est:
', num2str(sol)];
disp(reponse);
temps = ['Le temps nécessaire au calcul est de ',num2str(t_dicho),' secondes'];
disp(temps);
```

Figure 687 : recherche de solution par la méthode de Newton

Exécutez le script. La courbe représentative de la fonction apparait (Figure 688).



Figure 688 : courbe représentative de la fonction f(x)

Observez la courbe représentative de la fonction et indiquez la valeur de x_0 et la précision **epsilon** souhaitée (Figure 689).

La solution de l'équation f(x)=0 est recherchée par la méthode de Newton autour de la valeur x0 Saisir la valeur de x0: 3

La recherche de la solution se fera avec une précision epsilon:

Spécifier la valeur de epsilon: 0.001

La solution de l'équation f(x)=0 autour de x=3 est: 1.7913 Le temps nécessaire au calcul est de 0.00063 secondes

Figure 689 : exécution du script Newton.m pour rechercher la solution autour de x=3

Vous pouvez relancer le script pour chercher une solution autour d'une autre valeur de x₀ (Figure 690).

La solution de l'équation f(x)=0 est recherchée par la méthode de Newton autour de la valeur x0 Saisir la valeur de x0: -2 La recherche de la solution se fera avec une précision epsilon: Spécifier la valeur de epsilon: 0.001 La solution de l'équation f(x)=0 autour de x=-2 est: -2.7913 Le temps nécessaire au calcul est de 0.0014403 secondes

Figure 690 : exécution du script Newton.m pour rechercher la solution autour de x=-2

V. Intégration numérique

A. Principe

Intégrer numériquement une fonction sur un intervalle [a ; b] consiste à évaluer numériquement l'aire sous la courbe représentative de la fonction.



Figure 691 : calcul de l'intégrale d'une fonction sur un intervalle [a ; b]

Pour réaliser une intégration numérique et calculer cette aire A, il faut dans un premier temps discrétiser l'intervalle [a ; b] en N intervalles de longueur identique en choisissant un pas d'intégration **h**. Plus ce pas sera petit et plus le calcul numérique sera précis.



Figure 692 : discrétisation du domaine d'intégration

La fonction à intégrer sera alors remplacée par son évaluation numérique y_i en chaque point x_i de l'intervalle [a; b].

 $x_i = a + i.h$

 $y_i = f(x_i)$

Les points x_i sont définis tels que :

Les valeurs de y_i sont calculées à partir des x_i:

Il faut ensuite choisir une méthode d'interpolation de la fonction f(x) entre deux valeurs de y_i. Nous aborderons deux cas d'interpolations :

• Interpolation de degré 0 : La fonction gardera une valeur constante à l'intérieur d'un intervalle discrétisé. On parlera de méthode des rectangles (Figure 693). Le calcul numérique de l'intégrale se ramène à additionner l'aire de chaque rectangle élémentaire.

$$I_{rec} = \sum_{i=0}^{i=N-1} h.y_i \simeq \int_a^b f(x) dx$$

• Interpolation de degré 1 : La fonction sera remplacée par une interpolation linéaire à l'intérieur d'un intervalle discrétisé. On parlera de méthode des trapèzes (Figure 694). Le calcul numérique de l'intégrale se ramène à additionner l'aire de chaque trapèze élémentaire.

$$I_{trap} = \sum_{i=0}^{i=N-1} h \cdot \frac{y_i + y_{i+1}}{2} \simeq \int_{a}^{b} f(x) dx$$



Figure 693 : méthode des rectangles



Il apparaît déjà à ce stade que plus le polynôme d'interpolation est de degré élevé et plus la méthode d'intégration numérique sera précise. La méthode des trapèzes sera plus précise que la méthode des rectangles.

B. Codage en langage MATLAB

Vous pouvez trouver les fichiers de toutes les fonctions et scripts présentées dans ce paragraphe dans le dossier **Algorithmique_avec_MATLAB/ Intégration_numérique.**

L'expression mathématique de la fonction à intégrer sera définie dans la fonction **f3.m (**Figure 695). Cette expression peut être modifiée pour tester toutes les fonctions souhaitées. Il faudra juste vérifier que lors de l'appelle de la fonction l'intervalle d'étude [a ; b] appartienne bien au domaine de définition de la fonction. On choisira $f_3(m) = \cos(x)$.

```
%% f3.m
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% f3.m contient l'expression mathématique de la fonction à intégrer
%%
function y = f3(x)
y = cos(x);
end
```

Figure 695 : codage de la fonction f3.m qui contient l'expression de la fonction à intégrer

1. Méthode des rectangles

Pour calculer l'intégrale d'une fonction par la méthode des rectangles, nous allons créer la fonction **fintegration_rec** (a,b,nb_points,f) (Figure 696).

Cette fonction prendra en arguments : **a** : borne inférieure de l'intervalle d'étude **b** : borne supérieur de l'intervalle d'étude **nb_points** : nombre de points à calculer **f** : fonction à intégrer

La fonction aura comme sorties :

t : vecteur contenant les instants correspondant aux points ou la valeur de l'intégrale est calculée df : vecteur contenant les valeurs de la fonction intégrée numériquement



Figure 696 : fonction fintégration_rec (a,b,nb_point,f)

La Figure 697 montre le codage de la fonction **fintegration_rec** (a,b,nb_points,f).

```
%% [t,df] = fintegration rec(a,b,nb points,f)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul l'intégrale d'une fonction par la méthode des rectangles
88
function [t, int f] = fintegration rec(a, b, nb points, f)
% calcul du pas h
h = (b-a)/nb points;
% définition des conditions initiale de l'intégration (par défaut CI=0)
int f(1) = 0;
t(1) = a;
% La boucle va ajouter l'aire des rectangles pour chaque intervalle et
% créer le vecteur contenant les temps et le vecteur contenant les valeurs
% de l'intégrale de la fonction prise en argument
for k = [2:1:nb points]
   % construction du vecteur des abscisses
   t(k) = t(k-1) + h;
    % construction du vecteur contenant les valeurs de l'intégrale
    % de la fonction
    int f(k) = int f(k-1) + f3(t(k)) * h;
end
```

Figure 697 : codage de la fonction fintegration_rec (a,b,nb_point,f)

2. Méthode des trapèzes

Pour calculer l'intégrale d'une fonction par la méthode des trapèzes, nous allons créer la fonction **fintegration_trap** (a,b,nb_points,f) (Figure 698).

Cette fonction prendra en arguments :

- **a** : borne inférieure de l'intervalle d'étude
- **b** : borne supérieur de l'intervalle d'étude

nb_points : nombre de points à calculer

f : fonction à intégrer

La fonction aura comme sorties :

t : vecteur contenant les instants correspondant aux points ou la valeur de l'intégrale est calculée **df** : vecteur contenant les valeurs de la fonction intégrée numériquement



Figure 698 : fonction fintégration_trap (a,b,nb_point,f)

La Figure 699 montre le codage de la fonction **fintegration_trap** (a,b,nb_points,f).

```
%% [t,df] = fintegration trap(a,b,nb points,f)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul l'intégrale d'une fonction par la méthode des trapèzes
22
function [t, int f] = fintegration trap(a, b, nb points, f)
% calcul du pas h
h = (b-a)/nb points;
% définition des conditions initiale de l'intégration (par défaut CI=0)
int f(1) = 0;
t(1) = a;
% La boucle va ajouter l'aire des trapèzes pour chaque intervalle et
% créer le vecteur contenant les temps et le vecteur contenant les valeurs
% de l'intégrale de la fonction prise en argument
for k = [2:1:nb points]
    % construction du vecteur des abscisses
   t(k) = t(k-1) + h;
    % construction du vecteur contenant les valeur de l'intégrale
    % de la fonction
    int_f(k) = int_f(k-1) + (f3(t(k))+f3(t(k-1))) * h/2;
end
```

Figure 699 : codage de la fonction fintegration_trap (a,b,nb_point,f)

3. Calcul exact de l'intégrale d'une fonction en utilisant le calcul symbolique

Afin de pouvoir évaluer l'erreur induite par les différentes méthodes d'intégration numérique, nous allons créer une fonction **fintegration_ana_symbolique** (a,b,nb_points,f) qui calcule l'intégrale analytique exacte d'une fonction en calcul symbolique (Figure 700). Cette intégration constituera la référence pour calculer l'erreur.

Cette fonction prendra en arguments : **a** : borne inférieure de l'intervalle d'étude **b** : borne supérieur de l'intervalle d'étude **nb**_points : nombre de points à calculer **f** : fonction à dériver

La fonction aura comme sorties :

t : vecteur contenant les instants correspondant aux points ou l'intégrale est calculée df : vecteur contenant les valeurs de l'intégrale de la fonction calculée analytiquement



Figure 700 : fonction fintegration_ana_symbolique (a,b,nb_point,f)

La Figure 701 montre le codage de la fonction fintegration_ana_symbolique (a,b,nb_points,f).

```
%% [t,df] = fintegration anasymbolique(a,b,nb points,f)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul analytique exacte de l'intégrale d'une fonction en utilisant
% le calcul symbolique
function [t,int f]=fintegration ana symbolique(a,b,nb points,f)
h = (b-a)/nb points;
t = [a:h:b];
% x est déclarée comme variable symbolique
syms x;
% création de la fonction symbolique f symbolique à partir de f(x)
f symbolique(x) = f(x);
% intégration de la fonction f symbolique pour créer la fonction dérivée
% symbolique
int f symbolique(x) = int(f symbolique,x);
% calcul des valeurs du vecteur int f qui contient les valeurs de
% l'intégrale de la fonction calculée analytiquement
int f = int f symbolique(t);
% conversion de la fonction symbolique en double précision
int f = double(int f);
int f(end) = [];
t(end) = [];
end
```

Figure 701 : codage de la fonction fintegration_ana_symbolique (a,b,nb_point,f)

C. Exploitation de l'algorithme

1. Comparaison des différentes méthodes d'intégration numérique

Afin d'évaluer les écarts entre les différentes méthodes d'intégration numérique et l'intégrale de la fonction calculée analytiquement, nous allons créer un script qui va tracer les intégrales calculées avec les 3 méthodes dans la même fenêtre graphique, puis quantifier et tracer l'amplitude comparée de ces écarts.

Ouvrir le script num_integration_plot_comparaison.m.

Le script est structuré en trois sections (Figure 702) :

- Définition de l'intervalle d'étude de la fonction **[a ;b]** et du nombre de points **nb_points** à calculer pour tracer les courbes
- Appel des différentes fonctions et tracé des différentes courbes sur la même fenêtre graphique
- Evaluation des écarts entre l'intégrale analytique et les deux méthodes d'intégration numérique et tracé des écarts sur la même fenêtre graphique

```
%% num integration comparaison
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
%Calcul numérique de l'intégrale d'une fonctionn en utilisant différentes
% methodes de calcul:
% - méthode des rectangles
% - méthode des trapèzes
% Une comparaison est effectuée avec le calcul de l'intégrale théorique
% exacte effectuée en calcul symbolique
% Il est possible de faire varier l'intervalle [a;b] d'étude de la fonction
% et le nombre de points nb points pris en compte pour le calcul
% La fonction à intégrer est stockée dans la fonction f3.m
%% Script
% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 10;
% définition du nombre de point à utiliser pour le calcul
nb points = 20;
close all;
%% Calcul et tracé de l'intégrale de la fonction en utilisant différente méthodes
% création et positionnement de la fenêtre graphique
figure('Units','centimeters','Position',[2 8 18 12]);
grid on; grid minor; hold on;
% calcul analytique exacte et tracé de l'intégrale de la fonction f en
% utilisant le calcul symbolique
[t ana symbolique, int f ana symbolique] =
fintegration ana symbolique(a,b,nb_points,@f3);
plot(t ana symbolique,int f ana symbolique,'red','LineWidth',2);
% calcul et tracé de l'intégrale de la fonction en utilisant la méthode des
% rectangles
[t rec, int f rec] = fintegration rec(a, b, nb points, @f3);
```

```
plot(t rec,int f rec,'cyan','LineWidth',2);
% calcul et tracé de l'intégrale de la fonction en utilisant la méthode des
% trapèzes
[t trap, int f trap] = fintegration trap(a, b, nb points, @f3);
plot(t trap, int f trap, 'blue', 'LineWidth', 2);
% titre et légende
title({'Comparaison des méthodes d''intégration numérique';...
    ' de la fonction f3'});
legend ('Intégrale calculée analytiquement', 'Intégrale calculée avec la méthode
des rectangles',...
    'Intégrale calculée avec la méthode des trapèzes');
%% %% Calcul des différents écarts en prenant comme référence le calcul
% de l'intégrale exacte déterminée en calcul symbolique
error rec = int f ana symbolique - int f rec;
error trap = int f ana symbolique - int f trap;
% création du vecteur des abscisses
t = [a:(b-a)/nb points:b];
t(end) = [];
% création et positionnement de la fenêtre graphique
figure('Units','centimeters','Position',[22 8 18 12]);
hold on; grid on; grid minor;
% tracé des différents écarts
plot(t,error rec,'green','LineWidth',2)
plot(t,error trap, 'blue', 'LineWidth',2)
% titre et légende
title({'Erreurs comparées des différentes méthodes d''intégration numérique';...
    'par rapport à l''intégrale calculée analytiquement'});
legend('Méthode des rectangles', 'Méthode des trapèzes')
xlim([a b]);
```

Figure 702 : comparaison des différentes méthodes de dérivation numérique

Dans un premier temps le script sera exécuté sur l'intervalle [0 ; 10] avec 20 points calculés. La fonction définie dans **f3.m** est la fonction f(x) = cos(x). Cette fonction peut être modifiée par l'utilisateur.

L'exécution du script permet de visualiser les résultats du calcul numérique de l'intégrale de la fonction avec les différentes méthodes de calcul (Figure 703, Figure 704 et Figure 705).



Figure 703 : comparaison des différentes méthodes d'intégration numérique







Figure 705 : visualisation des erreurs induites par les différentes méthodes d'intégration numérique

L'analyse des différents courbes nous permet de voir de manière évidente que la méthode des trapèzes est plus précise que la méthode des rectangles.

2. Quantification et visualisation de l'erreur induite par les différentes méthodes

Il serait utile d'avoir une idée de l'évolution de cette erreur en fonction du nombre de point calculés et donc en fonction du pas $h = \frac{b-a}{nb_{-}points}$.

Pour cela ouvrir le script num_integration_plot_comparaison_subplot.m.

Ce script permet de tracer 4 courbes avec un nombre de points calculés différents pour chaque courbe et de voir ainsi l'évolution de l'erreur en fonction du nombre de points et du pas h choisi pour le calcul. Les 4 courbes correspondent à 4 valeurs du nombre de points calculés choisies équitablement réparties entre les variables **nb_points_min** et **nb_points_max** définies en début de script.

Le script est structuré en deux sections (Figure 706).

- Définition de l'intervalle d'étude de la fonction **[a ; b]** et du nombre de points **nb_points_min** et **nb_points_max**
- Appel des différentes fonctions et tracé des différentes courbes sur 4 fenêtres graphiques différentes

```
%% num integration comparaison subplot
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
8
% Calcul numérique de l'intégrale d'une fonctionn en utilisant différentes
% methodes de calcul:
% - méthode des rectangles
% - méthode des trapèzes
% Une comparaison est effectuée avec le calcul de l'intégrale théorique
% exacte effectuée en calcul symbolique
% Il est possible de faire varier l'intervalle [a;b] d'étude de la fonction
% Représentation de 4 courbes correspondant à 4 valeurs du nombres de
% points calculés choisies équitablement réparties entre nb points min et
% nb points max
% La fonction à intégrer est stockée dans la fonction f3.m
%% Script
% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 10;
% spécification des nombres de points mini et maxi
nb points min = 10;
nb points max = 50;
pas = round((nb points max - nb points min) / 3);
22
fig = figure('Units','centimeters','Position',[1 1 35 25]);
set(fig, 'Color', 'white');
sgtitle({'Comparaison des différentes méthodes d''intégration numérique';...
    ' de la fonction f3 en fonction du nombre de points calculés'});
% initialisation du compteur de boucle
k = 1;
for nb points = nb points min : pas : nb points max;
subplot(2,2,k); hold on; grid on; grid minor;
% calcul analytique exacte et tracé de l'intégrale de la fonction f en
% utilisant le calcul symbolique
[t ana symbolique, int f ana symbolique] =
fintegration ana symbolique(a,b,nb points,@f3);
plot(t ana symbolique,int f ana symbolique,'red','LineWidth',2);
% calcul et tracé de l'intégrale de la fonction en utilisant la méthode des
% rectangles
[t rec, int f rec] = fintegration rec(a, b, nb points, @f3);
plot(t_rec, int_f_rec, 'cyan', 'LineWidth', 2);
% calcul et tracé de l'intégrale de la fonction en utilisant la méthode des
% trapèzes
[t trap,int f trap] = fintegration trap(a,b,nb points,@f3);
plot(t trap, int f trap, 'blue', 'LineWidth', 2);
% legend('dérivée analytique','différence finie centrée',...
8
      'différence finie progressive', 'différence fine rétrograde');
title('Nombre de points calculés ' + string(nb points))
k = k + 1;
end
```

```
% spécification de la légende générale du graphique
leg = legend('Intégration analytique','Méthode des rectangles',...
'Méthode des trapèzes');
% Positionnement de la légende au centre du graphique
newPosition = [0.45 0.47 0.1 0.07];
newUnits = 'normalized';
set(leg,'Position', newPosition,'Units', newUnits);
```

Figure 706 : script permettant de visualiser l'influence du pas

L'exécution du script permet de visualiser l'influence du pas sur l'allure des courbes (Figure 656).



Figure 707 : influence du pas sur l'allure des courbes

L'analyse de ces courbes nous confirme que l'erreur va diminuer avec le pas mais ne nous permet pas de connaître la loi d'évolution de l'erreur en fonction du pas h. Lors de l'utilisation d'un processus d'intégration numérique, il est important de déterminer cette loi d'évolution.

3. Détermination de la loi d'évolution de l'erreur en fonction du pas

Pour connaître la loi d'évolution de cette erreur, il suffit de se placer en un point x_0 arbitraire de l'intervalle et de calculer l'erreur pour différentes valeurs du pas h.

L'erreur en fonction du pas h se définie de la manière suivante :

• Pour la méthode des rectangles

$$f_{err_{rec}}(h) = \int_{a}^{b} f(x) dx - \sum_{i=0}^{i=N-1} h.y_{i}$$

• Pour la méthode des trapèzes :

$$f_{err_trap}(h) = \int_{a}^{b} f(x) dx - \sum_{i=0}^{i=N-1} h \cdot \frac{y_i + y_{i+1}}{2}$$

Nous allons construire un script qui se placera arbitrairement au centre de notre intervalle d'étude et qui va tracer automatiquement l'erreur en fonction du pas h. Le script contient une boucle qui calculera les différentes fonctions erreurs pour un nombre de points allant de 100 à 1000 avec un incrément de 100.

Le pas h est défini par $h = \frac{b-a}{nb_points}$

La boucle permettra donc de calculer 10 valeurs de l'erreur au point x_0 pour les deux méthodes d'intégration numérique. Le script permet ensuite d'interpoler les points calculés avec une fonction polynomiale adaptée pour trouver l'ordre de variation des fonctions erreurs.

Ouvrir le script num_integration_error.m (Figure 708).

```
%% num integration error
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul de la forme mathémathique de l'erreur effectuée lors des
% opérations d'intégration numérique pour différentes méthodes de calcul:
% - méthode des rectangles
% - méthode des trapèzes
% Une comparaison est effectuée avec le calcul de l'intégrale théorique
% exacte effectuée en calcul symbolique
% On se place arbitrairement au centre de l'intervalle d'étude de la
% fonction en un point x0 afin d'évaluer l'erreur en ce point pour
% différentes valeurs de h et du nombre de points de calcul
% on utilise ensuite une interpolation polynômiale pour déterminer la forme
% mathématique de l'erreur en fonction de h
% La fonction à intégrer est stockée dans la fonction f3.m
%% Script
% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 10;
% initialisation du compteur qui servira d'indice
k = 1;
%% On définit ici une boucle qui permettra de calculer l'erreur pour un
% échantillon de points allant de 100 points jusqu'à 1000 points par pas
% de 100 pour toutes les méthodes de dérivation
for nb points = 100:100:1000
% calcul analytique exacte et tracé de l'intégrale de la fonction f en
% utilisant le calcul symbolique
[t ana symbolique, int f ana symbolique] =
fintegration ana symbolique(a,b,nb points,@f3);
% calcul et tracé de l'intégrale de la fonction en utilisant la méthode des
% rectangles
```

```
[t rec, int f rec] = fintegration rec(a, b, nb points, @f3);
% calcul et tracé de l'intégrale de la fonction en utilisant la méthode des
% trapèzes
[t trap, int f trap] = fintegration trap(a, b, nb points, @f3);
% calcul des erreurs au point x0, milieu de l'intervalle d'étude en prenant
% comme référence la dérivée exacte déterminée en calcul symbolique
x0 = round(nb points/2);
error f rec h(k) = int f ana symbolique(x0) - int f rec(x0);
error f trap h(k) = int f ana symbolique(x0) - int f trap(x0);
% détermination de la valeur du pas h(k) pour l'itération en cours
h(k) = (b-a)/nb points;
% incrémentation de k
k = k + 1;
end
%% Tracé et interpolation de l'erreur pour la méthode des rectangles
% Création d'une nouvelle figure pour les graphiques
figure('Units','centimeters','Position',[2 8 18 12]);
hold all; grid on; grid minor;
plot(h,error f rec h,'sm','MarkerSize',10,'LineWidth',2);
% Calcul des coefficients du polynôme d'interpolation
coeff poly inter rec = polyfit (h,error_f_rec_h,1);
% Tracé du polynôme d'interpolation
t = linspace(0, h(1), 100);
poly inter rec = polyval(coeff poly inter rec,t);
plot(t,poly inter rec, 'LineWidth',2);
% Ecriture de la légende
legend('Méthode des rectangles',...
     string(coeff poly inter rec(2)) + ' h + ' ...
     + string(coeff poly inter rec(1)), 'FontSize', 12);
% Ecriture du titre et des étiquettes des axes
title({'Variation de l''erreur en fonction du pas h' ;...
    'pour la méthode des rectangles'});
xlabel('pas h');
ylabel('erreur');
%% Tracé et interpolation de l'erreur pour la méthode des trapèzes
figure('Units','centimeters','Position',[22 8 18 12]);
hold all; grid on; grid minor;
plot(h,error f trap h,'sb','MarkerSize',10,'LineWidth',2)
% Détermination du polynome d'interpolation pour la différence finie
% progressive
% Calcul des coefficients du polynôme d'interpolation
coeff poly inter_trap = polyfit (h,error_f_trap_h,2);
% Tracé du polynôme d'interpolation
t = linspace(0, h(1), 100);
poly inter trap = polyval(coeff poly inter trap,t);
plot(t,poly inter trap, 'LineWidth',2);
% écriture de la légende
```

```
legend('Méthode des trapèzes',...
    string(coeff_poly_inter_trap(3)) + ' h^2 + ' ...
    + string(coeff_poly_inter_trap(2)) + ' h + ' ...
    + string(coeff_poly_inter_trap(1)),...
    'Location','southwest','FontSize',12);
% titre et étiquette des axes
title({'Variation de l''erreur en fonction du pas h' ;...
    'pour la méthode des trapèzes'});
xlabel('pas h');
ylabel('erreur');
```



Exécutez le script et analyser les courbes obtenues sur la Figure 709 et sur la Figure 710.



Figure 709 : variation de l'erreur d'intégration en fonction de h pour la méthode des rectangles



Figure 710 : variation de l'erreur de d'intégration en fonction de h pour la méthode des trapèzes

En observant les points qui représentent l'erreur en fonction du pas h, nous choisissons une régression linéaire pour interpoler l'erreur pour la méthode des rectangles et une interpolation polynomiale d'ordre 2 pour la méthode des trapèzes. Il apparait que ces choix permettent d'interpoler la série de données avec une excellente précision. Les équations des polynômes d'interpolation apparaissent dans les légendes des différentes fenêtres graphiques.

En conclusion :

- l'erreur pour la méthode des rectangles est d'ordre 1 en h
- l'erreur pour la méthode des trapèzes est d'ordre 2 en h

L'exploitation d'un algorithme peut donc nous amener à connaître la tendance de variation d'un paramètre en exploitant la puissance de calcul et en utilisant les outils adaptés. Notre seule intuition est parfois insuffisante et nous devons nous appuyer sur le potentiel de notre algorithme pour conforter nos prévisions.

4. Robustesse du processus

Afin de tester la robustesse de notre algorithme, nous pouvons tester plusieurs fonctions en modifiant la fonction **f3.m** et en reproduisant le processus.

Modifier la fonction **f3.m** pour reproduire l'analyse sur une fonction polynomiale $y(x) = 5x^3 - 3x^2 + 5x + 1$ (Figure 711).

```
function y = f3(x)
y = 5*x^3-3*x^2+5*x+1;
end
```

Figure 711 : modification de la fonction f1.m



Relancer les différents scripts pour visualiser les résultats dans les différentes figures.

Figure 712 : comparaison des différentes méthodes d'intégration numérique



Figure 713 : visualisation des erreurs induites par les différentes méthodes d'intégration numérique



Figure 714 : influence du pas sur l'allure des courbes



Figure 715 : variation de l'erreur d'intégration en fonction de h pour la méthode des rectangles



Figure 716 : variation de l'erreur d'intégration en fonction de h pour la méthode des trapèzes

D. Applications

1. Analyse d'une loi en trapèze de vitesse

Nous relevons à l'aide d'un accéléromètre monté sur le chariot d'un axe linéaire le signal a(t) de la Figure 717. Nous souhaitons à partir de ce signal calculer numériquement la vitesse v(t) et la position x(t) du chariot à chaque instant.



Figure 717 : relevé de l'accélération en fonction du temps

Pour avoir accès à la vitesse et à la position du chariot, nous allons mettre en place un processus d'intégration numérique. En intégrant l'accélération par rapport au temps, nous obtiendrons la vitesse v(t). En intégrant la vitesse par rapport au temps, nous obtiendrons la position x(t).

Nous considérons qu'à l'instant t=0, le chariot est à l'arrêt (v(0)=0) et que sa position initiale est prise comme origine (x(0)=0).

Pour effectuer le processus d'intégration numérique, nous utiliserons la méthode des trapèzes afin de minimiser l'erreur de calcul.

Ouvrir le script Loi_en_trapeze_de_vitesse.m (Figure 718).

Ce script permet de charger les données correspondant à la mesure de l'accélération. Ensuite un processus d'intégration numérique calcule la vitesse et la position. Les trois courbes sont tracées dans la même fenêtre graphique.

```
%% Loi_en_trapeze_de_vitesse.m
% Ivan LIEBGOTT @ décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Calcul l'intégrale d'une fonction par la méthode des trapèzes
% Un relevé expérimental donne l'accélération en mm/s2 en fonction du
% temps. En utilisant l'intégration par la méthode des trapèzes, on calcule
% la vitesse et la position.
% les trois courbes sont tracées dans la même fenêtre graphique
% chargement des mesures de l'accélération
clear all;
load loi en trapeze de vitesse data.mat
%% Calcul de la vitesse par intégration de l'acceleration
% définition des conditions initiale de la vitesse
vitesse(1) = 0;
% La boucle va ajouter l'aire des trapèzes pour chaque intervalle
% d'amplitude h = t(k-1)-t(k)
for k = [2:1:length(t)]
    % calcul du pas d'intégration
   h = t(k) - t(k-1);
    % construction du vecteur contenant les valeur de la vitesse en
    % utilisant la méthode des trapèzes
    vitesse(k) = vitesse(k-1) + (acceleration(k)+acceleration(k-1)) * h/2;
end
%% Calcul de la position par intégration de la vitesse
% définition des conditions initiale de la vitesse
position(1) = 0;
% La boucle va ajouter l'aire des trapèzes pour chaque intervalle
% d'amplitude h = t(k-1)-t(k)
for k = [2:1:length(t)]
   % calcul du pas d'intégration
   h = t(k) - t(k-1);
    % construction du vecteur contenant les valeur de la vitesse en
    % utilisant la méthode des trapèzes
    position(k) = position(k-1) + (vitesse(k)+vitesse(k-1)) * h/2;
end
% Tracé des trois courbes dans la même fenêtre graphique
```

```
figure('Units','centimeters','Position',[2 8 25 20]);
subplot(3,1,1);
plot(t,acceleration);
grid on; grid minor;
xlabel('temps en s');
ylabel('accélération en mm/s^2')
title('Mesure de l''accélération du chariot en fonction du temps en mm/s^2')
subplot(3,1,2);
plot(t,vitesse);
grid on; grid minor;
xlabel('temps en s');
ylabel('vitesse en mm/s')
title({'Vitesse du chariot en fonction du temps en mm/s',...
    'obtenue par intégration numérique'})
subplot(3,1,3);
plot(t,position);
grid on; grid minor;
xlabel('temps en s');
ylabel('Position en mm')
title({'Position du chariot en fonction du temps en mm',...
    'obtenue par intégration numérique'});
xlabel('temps en s');
ylabel('Position en mm')
title({'Position du chariot en fonction du temps en mm',...
    'obtenue par intégration numérique'});
```

Figure 718 : intégration numérique de l'accélération

Exécuter le script et observer le résultat obtenu sur la Figure 719.



Figure 719 : loi en trapèze de vitesse

L'analyse de la courbe de vitesse nous donne la vitesse maximale atteinte par le chariot qui est de 26 mm/s. La courbe de position nous permet de déterminer la position finale qui est 400 mm.

VI. Filtrage d'un signal

A. Intérêt du filtrage d'un signal

Lors du processus d'acquisition d'une grandeur physique, les signaux provenant des capteurs sont très souvent perturbés par des signaux qui viennent s'ajouter au signal que l'on souhaite mesurer (Figure 720). Ce bruit prend souvent la forme d'un signal haute fréquence. Ces perturbations ou signaux parasites sont de différentes natures :

- Signal périodique 50 Hz, provenant du couplage électromagnétique entre le circuit utilisé et les conducteurs du réseau d'alimentation
- Perturbations électromagnétiques dues à la présence d'onde Hertzienne (téléphone portable, réseaux wifi...)
- Perturbations liées à la nature du circuit (bruit de fond)
- Discontinuités due à la nature des capteurs (codeur incrémental)



Figure 720 : exemple de signal bruité

Le filtrage consiste à extraire la composante que l'on souhaite mesurer en éliminant tous les bruits parasites ou à lisser un signal en vue d'un processus de traitement numérique (dérivation numérique en particulier).

La Figure 721 illustre le principe du filtrage numérique.



Figure 721 : filtre numérique

 y_k : valeur numérique de l'échantillon en sortie de filtre

- xk : valeur numérique de l'échantillon en entrée de filtre
- T_e : période d'échantillonnage

Nous verrons dans ce chapitre deux filtres élémentaires qui permettent de mettre en évidence l'intérêt du filtrage d'un signal.

- Le filtre à moyenne glissante
- Le filtre du premier ordre

Nous allons voir comment coder en langage MATLAB ces filtres élémentaires et les utiliser dans quelques cas pratiques.

Nous travaillerons sur le signal issu d'un codeur incrémental représenté sur la Figure 722 et sur la Figure 723. Les discontinuités dans le signal devront être atténuées par le processus de filtrage.



Figure 722 : mesure de la position à partir du signal d'un codeur incrémental



Figure 723 : zoom sur la courbe

B. Le filtre à moyenne glissante

Ce filtre très simple à coder est largement utilisé pour améliorer la qualité d'un signal bruité. Il permet de lisser le signal. Le principe est de calculer l'échantillon y_k comme étant la moyenne des n échantillons d'entrée précédents.

$$y_k = \sum_{i=0}^{i=n-1} \frac{1}{n} x_{k-i}$$

Par exemple si n=3, l'échantillon y_k sera calculé en faisant la moyenne des trois derniers échantillons :

$$y_{k} = \frac{1}{3} (x_{k} + x_{k-1} + x_{k-2})$$

La valeur de n sera choisie en fonction de la fréquence du bruit que l'on souhaite supprimer et de la fréquence d'échantillonage. Parfois plusieurs essais doivent être réalisés pour obtenir un lissage satisfaisant sans trop retarder le signal filtré par rapport au signal d'origine. Dans tous les cas il faudra trouver le meilleur compromis sachant que la composante continue subira de toute façon une légère déformation.

La fonction **filtre_mgl(Y,n)** (Figure 724) permet le filtrage par moyenne glissante d'un signal Y en prenant n échantillons pour faire la moyenne.

Cette fonction prendra en arguments :

Y : vecteur contenant les valeurs de la fonction à filtrer

n : nombre d'échantillons à prendre en compte pour la moyenne

La fonction aura comme sorties :

Y_mgl : vecteur contenant les valeurs du signal filtré par moyenne glissante



Figure 724 : fonction filtre_mgl(Y,n)

La Figure 725 montre le codage de la fonction **filtre_mgl(Y,n)**.

```
%% filtre mgl(Y,n)
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction permet de filtrer un signal en utilisant un filtre à
% moyenne glissante. Chaque point calculé représente la moyenne des n
% dernières valeurs du signal.
% Y : représente le vecteur contenant le valeurs du signal à filtrer
% n : nombre de points pris en compte pour faire la moyenne
function [Y mgl] = filtre mgl(Y,n)
% pour chaque point du signal Y à partir du point Y(n+1)
for p = n+1:1:length(Y)
    % calcul de la moyenne des n termes précédents
   moyenne glissante = 0;
        for k = 1:1:n
        moyenne glissante = moyenne glissante + Y(p-k)/n;
        end
    % affectation de la valeur de l'échantillon pour la valeur de p
    Y mgl(p) = moyenne glissante;
% transposition pour obtenir une matrice colonne
Y mgl = Y mgl';
end
```

Figure 725 : codage de la fonction fliltre_mgl(Y,n)

Afin de tester cette fonction sur un signal en prenant plusieurs valeurs pour **n**, **ouvrir** le script **filtrage_mgl_n_variable.m** (Figure 726).

```
%% filtrage mgl n variable
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Ce script permet de tester la fonction filtre mgl(Y,n) qui permet de
% filtrer un signal en utilisant le principe de la moyenne glissante
% Le test est réalisé sur le signal issu d'un codeur
% Le script trace le signal non filtré et les signaux obtenus après
% filtrage par moyenne glissante en utilisant différentes valeurs du nombre
% n d'échantillons retenus pour faire la moyenne
22
% Effacement des variables du workspace
clear all;
% chargement des données correspondant au signal du codeur et au temps dans
% le workspace
load filtrage mgl.mat;
% spécification des valeurs mini et maxi de n
n min = 10;
n max = 110;
pas = round((n max - n min) / 5);
% création d'une nouvelle fenêtre graphique
fig = figure('Units','centimeters','Position',[2 5 33 23]);
set(fig, 'Color', 'white');
hold all;
% tracé du signal du codeur non filtré
plot(t,position codeur,'r','LineWidth',3);
```

```
%création d'une cellule pour stocker les lignes de la légende
Titre Legende = cell(1,1);
Titre Legende{1,1} = 'Signal brut';
ind Legende = 2;
for n = n_min : pas : n_max;
% calcul du signal filtré à l'aide de la fonction filtre mgl
pos mgl = filtre mgl(position codeur,n);
Titre Legende{1, ind Legende} =
                               'n=' + string(n); % création de la ligne de la
légende
% Tracé du signal filtré correspondant à la valeur de n dans la même fenêtre
graphique
plot(t,pos mgl);
ind Legende = ind Legende + 1;
end
grid on; grid minor;
% affichage de la légende
legend(Titre Legende, 'Location', 'northwest');
title('Filtrage d''un signal en utilisant un filtre à moyenne glissante avec n
compris entre'...
    + string(n min) + ' et ' + string(n_max));
```

Figure 726 : script permettant de voir l'influence de n pour un filtre à moyenne glissante

Exécuter le script et observer le résultat sur la Figure 727 et sur la Figure 728.





Figure 727 : filtrage par moyenne glissante d'un signal

Figure 728 : zoom sur la courbe

Nous pouvons constater que le filtre à moyenne glissante a pour effet de lisser le signal mais également de le retarder. Plus la valeur de n est élevée, plus le signal est lissé et plus il est retardé. Il est donc intéressant de tester plusieurs valeurs de n afin de trouver le meilleur compromis.

C. Le filtre passe-bas du premier ordre

Le filtre passe-bas du premier ordre agit sur le signal en coupant les composantes hautes fréquences tout en laissant passer la composante continue.

La constante de temps du filtre va permettre de choisir la bande de fréquences à partir de laquelle le filtre va atténuer l'amplitude du bruit. Le réglage de la constante τ de temps du filtre dépendra donc de la fréquence des perturbations à éliminer (Figure 729 et Figure 730).



Figure 729 : filtre passe-bas du premier ordre dans le domaine de Laplace


Figure 730 : filtre passe-bas du premier ordre dans le domaine temporel

Dans le domaine temporelle la fonction filtrée y(t) est solution de l'équation différentielle suivante :

$$\tau y'(t) + y(t) = x(t)$$

Si le signal est échantillonné avec une période d'échantillonnage $T_e = h$ et en utilisant la formule de différence finie progressive, il est possible de discrétisée cette équation différentielle :

$$\tau\left(\frac{y_{k+1}-y_k}{h}\right) + y_k = x_k$$

On obtient alors l'équation de récurrence qui permet de calculer l'échantillon y_{k+1} en fonction de y_k et de x_k :

$$y_{k+1} = \frac{h}{\tau} x_k + \left(\frac{\tau - h}{\tau}\right) y_k$$

Cette relation de récurrence sera utilisée par la fonction qui permettra de coder le filtre passe-bas du premier ordre.

La valeur de τ sera choisie en fonction de la fréquence du bruit que l'on souhaite supprimer. Parfois plusieurs essais doivent être réalisés pour obtenir un lissage satisfaisant sans trop retarder le signal filtré par rapport au signal d'origine. Dans tous les cas il faudra trouver le meilleur compromis sachant que la composante continue subira de toute façon une légère déformation.

La fonction **filtre_ordre_1(t,Y,tau)** (Figure 731) permet le filtrage par un filtre passe-bas du premier ordre du signal Y.

Cette fonction prendra en arguments :

Y : vecteur contenant les valeurs de la fonction à filtrer

t : vecteur contenant les valeurs du temps correspondant au vecteur Y **tau** : constante de temps du filtre

La fonction aura comme sorties :

Y_pb_ordre_1 : vecteur contenant les valeurs du signal filtré par moyenne glissante



Figure 731 : fonction filtre_pb_ordre_1(t,Y,tau)

La Figure 725 montre le codage de la fonction **filtre_pb_ordre_1(t,Y,tau)**.

```
%% filtre pb ordre 1(t,Y,tau)
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction permet de filtrer un signal en utilisant un filtre
% passe-bas du premier ordre
% t : vecteur contenant les valeurs du temps correspondant au signal à
% filtrer
% Y : représente le vecteur contenant le valeurs du signal à filtrer
% tau : constante de temps du filtre
function [Y pb ordre 1] = filtre_pb_ordre_1(t,Y,tau)
% définition du pas h
h = (t(end) - t(1)) / (length(t)-1);
Y pb ordre 1(1) = Y(1);
for \overline{k} = 1:\overline{1}:\text{length}(Y)-1
    % calcul de Y au rang k+1
    Y pb ordre 1(k+1) = Y(k)*(h/tau) + Y pb ordre 1(k)*(tau-h)/tau;
end
```

Figure 732 : codage de la fonction filtre_pb_ordre_1(t,Y,tau)

Afin de tester cette fonction sur un signal en prenant plusieurs valeurs pour **tau**, **ouvrir** le script **filtrage_pb_ordre_1_tau_variable.m**.

```
%% filtrage pb ordre 1 tau variable
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Ce script permet de tester la fonction filtre pb ordre 1.m qui permet de
% filtrer un signal en utilisant un filtre passe-bas du premier ordre
% Le test est réalisé sur le signal issu d'un codeur
% Le script trace le signal non filtré et les signaux obtenus après
% filtrage en utilisant différentes valeurs pour la constante de temps tau
% du filtre d'échantillons retenus pour faire la moyenne
88
% Effacement des variables du workspace
clear all;
% chargement des données correspondant au signal du codeur et au temps dans
% le workspace
load filtrage pb ordre 1.mat;
% spécification des valeurs mini et maxi de n
tau min = 0.3;
tau max = 2;
pas = ((tau max - tau min) / 5);
% création d'une nouvelle fenêtre graphique
fig = figure('Units','centimeters','Position',[2 5 33 23]);
set(fig, 'Color', 'white');
hold all;
% tracé du signal du codeur non filtré
plot(t,position codeur, 'r', 'LineWidth',2);
%création d'une cellule pour stocker les lignes de la légende
Titre Legende = cell(1,1);
Titre Legende{1,1} = 'Signal brut';
ind Legende = 2;
```

```
for tau = tau min : pas : tau max;
% calcul du signal filtré à l'aide de la fonction filtre_mgl
pos_ordre_1 = filtre_pb_ordre_1(t,position_codeur,tau);
Titre Legende{1, ind Legende} = 'tau=' + string(tau); % création de la ligne de
la légende
% Tracé du signal filtré correspondant à la valeur de n dans la même fenêtre
graphique
plot(t,pos ordre 1);
ind Legende = ind Legende + 1;
end
grid on; grid minor;
% affichage de la légende
legend(Titre Legende, 'Location', 'northwest');
title('Filtrage d''un signal en utilisant un filtre du premier ordre avec tau
comprise entre '...
    + string(tau min) + ' et ' + string(tau max));
```

Figure 733 : script permettant de voir l'influence de tau pour un filtre passe-bas du premier ordre

Exécuter le script et observer le résultat sur la Figure 733et sur la Figure 734.



Figure 734 : filtrage d'un signal par un filtre passe-bas du premier ordre avec différentes valeurs de t

Nous pouvons constater que le filtre passe-bas du premier ordre a pour effet de lisser le signal mais également de le retarder. Plus la valeur de τ est élevée et plus le signal est retardé. Il est donc intéressant de tester plusieurs valeurs de τ afin de trouver une valeur de τ adaptée.

D. Utilisation de Simulink pour le filtrage d'un signal

Simulink propose de nombreuses fonctionnalités permettant de filtrer un signal. Il est possible d'utiliser les blocs classiques comme les fonctions de transfert ou d'utiliser des blocs de la DSP System Toolbox qui propose tous les blocs dédiés au traitement du signal.

Afin de comparer les différentes fonctionnalités, nous allons lui appliquer le processus de traitement indiqué sur la Figure 735. Le signal sera dans un premier temps filtré afin de lui appliquer une méthode de dérivation numérique.



Figure 735 : processus de traitement du signal

Ouvrir le modèle **Simulink_numerical_derivative.slx** et lancer la simulation. Ce modèle permet de visualiser un signal provenant d'un codeur incrémental ainsi que sa dérivée numérique obtenue en utilisant le bloc **Numerical Derivative** de la bibliothèque **Continuous** de Simulink.



Figure 736 : dérivation numérique en utilisant Simulink

Ouvrir le scope permettant de visualiser le signal brut et observer la forme du signal sur la Figure 737. Nous pouvons constater que ce signal présente des discontinuités qui vont nuire au processus de dérivation numérique.

Ouvrir ensuite le scope correspondant au signal dérivé afin d'observer l'influence de ces discontinuités sur le calcul de la dérivée numérique (Figure 738).

Nous pouvons constater que le signal dérivé n'est pas exploitable, un processus de filtrage est nécessaire afin de lisser le signal avant le processus de dérivation numérique. Plusieurs méthodes seront présentées dans ce paragraphe.



Figure 738 : dérivation d'un signal numérique discontinu sans filtrage

1. Utilisation d'une fonction de transfert de la bibliothèque Continuous de Simulink

Afin de reproduire l'effet d'un filtre passe-bas du premier ordre, il est possible d'utiliser un bloc **Transfer Fcn** de la bibliothèque **Continuous**.

Ouvrir le modèle Simulink **filtrage_Simulink_Transfert_Function.slx** et **lancer** la simulation (Figure 739). Ce modèle contient une fonction de transfert du premier ordre qui permet de modéliser le comportement d'un filtre passe-bas du premier ordre. La Figure 740 montre le paramétrage de la fonction de transfert $H(s) = \frac{1}{1+0.3 s}$, ou s représente la variable de Laplace.



Figure 739 : filtrage réalisé avec une fonction de transfert

Block Parameters: Transfer Fcn	×
Transfer Fcn	
The numerator coefficient can be a vector or matrix expression. The denominator coefficient must be a vector. The output width equals the number of rows in the numerator coefficient. You should specify the coefficients in descending order of powers of s.	
Parameters	
Numerator coefficients:	
[1]	÷
Denominator coefficients:	
[0.3 1]	:
Absolute tolerance:	
auto	:
State Name: (e.g., 'position')	
п	
OK Cancel Help App	bly

Figure 740 : paramétrage du bloc Transfer Fcn

Ouvrir le scope permettant de visualiser le signal brut, le signal filtré et le signal dérivé après filtrage. La Figure 741 permet de voir l'apport du filtrage. Le signal dérivé est maintenant exploitable. La constante de temps a été réglée à 0.3 s. Il est possible de faire d'autre simulation en faisant varier la constante de temps pour visualiser l'effet sur le signal filtré et sur le signal dérivé.



Figure 741 : dérivation numérique d'un signal filtré en utilisant Simulink

2. Utilisation du bloc Analog Filter Design de la DSP Toolbox de Simulink

Afin de reproduire l'effet d'un filtre passe-bas du premier ordre, il est également possible d'utiliser le bloc **Analog Filter Design** de la **DSP Toolbox** de Simulink (Figure 742).

Ouvrir le modèle Simulink **filtrage_Simulink_DSP_toolbox.slx** et **lancer** la simulation. Ce modèle contient une fonction de transfert du premier ordre qui permet de modéliser le comportement d'un filtre passe-bas du premier ordre.

La Figure 743 montre le paramétrage du bloc Analog Filter Design.



Figure 742 : filtrage réalisé avec le bloc Analog Filter Design

Block Parameters: Analog Filter Design3			
Analog Filter Design (mask) (link)			
Design one of several standard analog filters, implemented in state- space form.			
Parameters			
Design method: Butterworth	•		
Filter type: Lowpass			
Filter order:			
1	:		
Passband edge frequency (rad/s):			
5	:		
OK Cancel Help Apply			

Figure 743 : paramétrage du bloc Analog Filter Design

Ouvrir le scope permettant de visualiser le signal brut, le signal filtré et le signal dérivé après filtrage. La Figure 744 permet de voir l'apport du filtrage. Le signal dérivé est maintenant exploitable. En réglant le paramètre « **Passband edge fequency** » à 5 rad/s, nous obtenons un résultat satisfaisant Il est possible de faire d'autre simulation en faisant varier ce paramètre pour visualiser l'effet sur le signal filtré et sur le signal dérivé.



Figure 744 : dérivation numérique d'un signal filtré en utilisant Simulink

3. Utilisation du bloc Moving Average de la DSP System Toolbox

Afin de reproduire l'effet d'un filtre à moyenne glissante, il est également possible d'utiliser le bloc **Moving Average** de la **DSP system Toolbox** de Simulink.

Ouvrir le modèle Simulink **filtrage_Simulink_Moving_average.slx** et **lancer** la simulation (Figure 745). Ce modèle contient un bloc **Moving Average** de la **DSP System Toolbox**.

La Figure 746 montre le paramétrage du bloc Moving Average.

-	Moving average Filter + derivative	al filtré <u>Au</u> signal dérivé après filtrage par moyenne glissante
[t,position_codeur]	derivative without filtering	signal dérivé sans filtrage

Figure 745 : filtrage réalisé avec le bloc Moving Average

指 Block Paramete	ers: Moving Average	×	
Moving average			
Compute moving	Compute moving average		
Source code			
Parameters			
Method:	Sliding window	•	
Specify windo	w length		
Window length:	300	:	
Simulate using:	Code generation	•	
0	OK Cancel Help	Apply	

Figure 746 : paramétrage du bloc Moving Average

Ouvrir le scope permettant de visualiser le signal brut, le signal filtré et le signal dérivé après filtrage. La Figure 747 permet de voir l'apport du filtrage par moyenne glissante. Le signal dérivé est maintenant exploitable. En réglant le paramètre « **Window length** » à 300, on obtient un résultat satisfaisant (la moyenne est calculée avec les 300 derniers échantillons du signal). Il est possible de faire d'autre simulation en faisant varier ce paramètre pour visualiser l'effet sur le signal filtré et sur le signal dérivé.



Figure 747 : dérivation numérique d'un signal filtré en utilisant Simulink

VII. Résolution numérique des équations différentielles

De nombreux problèmes scientifiques se modélisent sous la forme d'une équation différentielle. Un processus de résolution doit alors être défini afin de déterminer les solutions. Dans quelques cas particuliers, l'équation différentielle admet une solution qui peut être déterminée par un processus de calcul analytique. En général, les équations différentielles obtenues ne sont pas linéaires et la mise en place d'un processus de résolution numérique est nécessaire afin de déterminer une solution approchée au problème.

A. Résolution d'une équation différentielle d'ordre 1 – Méthode d'Euler

1. Aspects théoriques

La méthode d'Euler va nous permettre de trouver la solution d'une équation différentielle d'ordre 1 donnée avec sa condition initiale sous la forme suivante :

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0 \end{cases}$$

Cette mise en forme d'une équation différentielle est appelée **problème de Cauchy**. Sous de bonnes hypothèses, le théorème de Cauchy-Lipschitz affirme qu'un tel problème admet une solution unique.

Exemple : Considérons l'équation différentielle suivante :

$$y'(t) + y(t) = \sin(t) \operatorname{avec} y(0) = 1$$

La mise sous la forme d'un problème de Cauchy de cette équation sera donc la suivante :

$$\begin{cases} y'(t) = \sin(t) - y(t) \\ y(0) = 1 \end{cases} \quad \text{avec} \quad f(t, y(t)) = \sin(t) - y(t) \end{cases}$$

<u>Principe de la méthode</u>: La fonction $f(t_i, y(t_i)) = y'(t_i)$ va être utilisée au point t_i pour évaluer le coefficient directeur de la tangente à la courbe représentative de f(t) pour l'intervalle $[t_i; t_{i+1}]$. La fonction $f(t_i, y(t_i))$ permettra de connaître la tendance de variation de la fonction f(t) au niveau du point t_i . La solution numérique de l'équation différentielle sera constituée des tangentes aux points t_i pour chaque intervalle $[t_i; t_{i+1}]$.

La Figure 748 illustre les 3 premières itérations de la méthode d'Euler. Les différentes valeurs des y_i permettront de reconstituer la solution numérique.



Figure 748 : illustration de la méthode d'Euler de résolution numérique d'une équation différentielle

- L'intervalle d'étude [a ; b] est discrétiser en N intervalles de longueur identique h.
- Sur l'intervalle $[x_0; x_1]$, l'équation de la tangente au point t_0 sera donné par :

$$T_0(t) = y'(t_0)(t - t_0) + y(t_0) = f(t_0, y_0)(t - t_0) + y_0 \qquad \text{en exploitant } y'(t) = f(t, y(t))$$

La valeur de y₁ est donnée par $T_0(t_1)$:

$$y_{1} = T_{0}(t_{1}) = f(t_{0}, y_{0})(t_{1} - t_{0}) + y_{0} \quad \text{avec} \quad (t_{1} - t_{0}) = h$$
$$y_{1} = h.f(t_{0}, y_{0}) + y_{0}$$

• Sur l'intervalle $[x_i; x_{i+1}]$, l'équation de la tangente au point t_i sera donné par :

$$T_i(t) = y'(t_i)(t_{i+1} - t_i) + y(t_i) = f(t_i, y_i)(t - t_i) + y_i \qquad \text{en exploitant } y'(t) = f(t, y(t))$$

La valeur de y_{i+1} est donnée par $T_i(t_{i+1})$:

$$y_{i+1} = T_i(t_{i+1}) = f(t_i, y_i)(t_{i+1} - t_i) + y_i \text{ avec } (t_1 - t_0) = h$$
$$y_{i+1} = h.f(t_i, y_i) + y_i$$

La solution numérique approchée sera alors reconstituée en joignant par des segments les points (t_k, y_k) tels que :

$$\begin{cases} t_{k+1} = t_k + h \\ y_{k+1} = h.f(t_k, y_k) + y_k \end{cases}$$

2. Codage en langage MATLAB

Vous pouvez trouver les fichiers de toutes les fonctions et scripts présentés dans ce paragraphe dans le dossier **Algorithmique_avec_MATLAB/ Equations_différentielles**

Nous prendrons comme exemple l'équation différentielle suivante :

$$\begin{cases} 5y'(t) + y(t) = 10\\ y(t_0) = 0 \end{cases}$$

La mise sous la forme d'un problème de Cauchy nous permet de déterminer la fonction : f(t, y(t))

$$\begin{cases} y'(t) = f(t, y(t)) = \frac{10 - y(t)}{5} \\ y(t_0) = 0 \end{cases} \text{ avec } f(t, y(t)) = \frac{10 - y(t)}{5} \end{cases}$$

L'expression mathématique de cette fonction sera définie dans la fonction $F_1.m$ (Figure 749). Cette expression peut être modifiée si l'on souhaite résoudre une autre équation différentielle du premier ordre.

```
%% F_1.m
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% F_1.m contient l'expression de la fonction F(t,y) utilisée pour résoudre
% une équation différentielle en utilisant la méthode d'Euler
% cette fonction correcpond à l'équation différentielle suivante:
% 5.dy/dt + y = 10
% dy/dt = F(t,y) = (10 - y) / 5
function f = F_1(t,y);
f = (10 - y) / 5;
end
```

Figure 749 : codage de la fonction F1_m qui contient l'expression de la fonction f(t,y)

Pour calculer les solutions de cette équation différentielle, nous allons créer la fonction **fEuler** (f,a,b,CI,nb_points) (Figure 750).

Cette fonction prendra en arguments : **f** : fonction contenant l'expression de f(t,y) **a** : borne inférieure de l'intervalle d'étude **b** : borne supérieur de l'intervalle d'étude **CI** : condition initiale de l'équation différentielle **nb_points** : nombre de points à calculer La fonction aura comme sorties :

t : vecteur contenant les instants correspondant aux points ou les valeurs de la solution sont calculées y : vecteur contenant les valeurs de la fonction solution de l'équation différentielle



Figure 750 : fonction fEuler (f, a, b, CI, nb_points)

La Figure 751 montre le codage de la fonction **fEuler** (f,a,b,CI,nb_points).

```
%%fEuler
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Résolution d'une équation différentielle du premier ordre mise sous la forme
d'un problème
% de Cauchy
% Equation scalaire: dy/dt = F(t,y)
function [t,y] = fEuler(f,a,b,CI,nb points)
\% calcul du pas h en fonction de a, b et nb points
h = (b-a)/nb points;
% définition du vecteur t
t = [a:h:b];
y(1) = CI;
for k = 1:1:nb_points
    y(k+1) = y(k) + h * f(t(k), y(k));
end;
```

Figure 751 : codage de la fonction fEuler(f, a, b, CI, nb_points)

Afin de procéder à la résolution de l'équation différentielle, ouvrir le script **Euler.m**.

Le script est structuré en deux sections (Figure 752) :

- Définition de l'intervalle d'étude de la fonction **[a ;b]** et du nombre de points **nb_points** à calculer pour déterminer la solution numérique de l'équation différentielle.
- Appel de la fonction fEuler, puis tracé de la solution numérique dans une fenêtre graphique

```
%% Euler
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Résolution d'une équation différentielle du premier ordre en utilisant
% la méthode d'Euler
%
% la fonction F(y,t) est stockée dans la fonction F_1.m
%% définition de l'intervalle d'étude [a;b] de la fonction considérée
```

```
a = 0;
b = 30;
%spécification du nombre de points
nb points = 10;
%% Résolution d'une équation différentielle du premier ordre
% spécification de la condition initiale
y0 = 0;
% Résolution avec fEuler
[t,y] = fEuler(@F_1,a,b,y0,nb_points);
figure; hold all
plot(t,y,'LineWidth',2);
grid on; grid minor;
title({'Solution numérique d''une équation différentielle' ; 'du premier ordre
(méthode d''Euler)'})
legend(string(nb points) + ' points sont calculés ')
axis([0 30 0 12]);
```

Figure 752 : résolution d'une équation différentielle du premier ordre

Exécuter le script avec un nombre de points très faible (nb_points = 10) afin de visualiser le processus de résolution et d'identifier les différents segments qui constituent la solution numérique.



Figure 753 : solution numérique calculée avec 10 points

Exécuter à nouveau le script avec un nombre de points plus important (nb_points = 100)



Figure 754 : solution numérique calculée avec 100 points

B. Résolution d'une équation différentielle d'ordre n – Méthode d'Euler

1. Aspects théoriques

La méthode d'Euler peut être généralisée et utilisée pour résoudre des équations différentielles d'ordre n en utilisant la vectorisation. Le principe consiste à transformer une équation différentielle d'ordre n en un système de n équations différentielles d'ordre 1.

Exemple : Considérons l'équation différentielle suivante :

$$y''(t) + a y'(t) + b y(t) = g(t)$$

Avec pour conditions initiales : $y(t_0) = C_0$ et $y'(t_0) = C_1$

Posons: $\begin{cases} y_1(t) = y(t) \\ y_2(t) = y'_1(t) \end{cases}$

L'équation différentielle peut alors s'écrire en fonction de $y_1(t)$ et $y_2(t)$:

$$y'_{2}(t) + a y_{2}(t) + b y_{1}(t) = g(t)$$

Il est alors possible de se ramener au système suivant contenant deux équations différentielles d'ordre 1 :

$$\begin{cases} y_1'(t) = y_2(t) \\ y_2'(t) = g(t) - a y_2(t) - b y_1(t) = \varphi(t, y_1(t), y_2(t)) \end{cases}$$

En considérant les vecteurs $Y(t) = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$, $Y'(t) = \begin{pmatrix} y'_1 \\ y'_2 \end{pmatrix}$ et $F(t, Y(t)) = \begin{pmatrix} y_2(t) \\ \varphi(t, y_1(t), y_2(t)) \end{pmatrix}$

Le système peut alors s'écrire sous forme vectorielle :

$$\overline{Y'(t) = F(t, Y(t))}$$
avec
$$F(t, Y(t)) = \begin{pmatrix} y_2 \\ g(t) - a \ y_2(t) - b \ y_1(t) \end{pmatrix}$$

Le problème se ramène à un problème de Cauchy avec une équation différentielle vectorialisée de dimension 2 et sa condition initiale Y_0 de dimension 2 également.

$$\begin{cases} Y'(t) = F(t, Y(t)) \\ Y_O = \begin{bmatrix} y_1'(t_0) = C_0 \\ y_2'(t_0) = C_1 \end{bmatrix} \end{cases}$$

Ce raisonnement peut se généraliser à l'ordre n : En considérant une équation différentielle d'ordre n mise sous la forme suivante :

$$y^{(n)}(t) = \varphi(t, y(t), y'(t), ..., y^{(n-1)}(t))$$

Posons: $\begin{cases} y_{1}(t) = y(t) \\ y_{2}(t) = y'(t) \\ y_{3}(t) = y^{(2)}(t) \\ \vdots \\ \vdots \\ y_{n}(t) = y^{(n-1)}(t) \end{cases}$

L'équation différentielle peut s'écrire comme un système de n équations différentielles d'ordre 1 :

$$\begin{cases} y_1'(t) = y_2(t) \\ y_2'(t) = y_3(t) \\ \vdots \\ \vdots \\ y_{n-1}'(t) = y_n(t) \\ y_n'(t) = \varphi(t, y(t), y'(t), \dots, y^{(n-1)}(t)) \end{cases}$$

En considérant les vecteurs :

$$Y(t) = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ y_n \end{pmatrix}, \ Y'(t) = \begin{pmatrix} y'_1 \\ y'_2 \\ \vdots \\ \vdots \\ \vdots \\ y'_n \end{pmatrix} \text{ et } F(t, Y(t)) = \begin{pmatrix} y_2 \\ y_3 \\ \vdots \\ \vdots \\ y_n \end{pmatrix}$$

Le système peut s'écrire sous la forme vectorielle suivante :

$$Y'(t) = F(t, Y(t))$$

Le problème se ramène à un problème de Cauchy avec une équation différentielle vectorialisée de dimension n et sa condition initiale Y_0 de dimension n également.

$$\begin{cases} Y'(t) = F(t, Y(t)) \\ Y_O = \begin{bmatrix} C_0 \\ \vdots \\ \vdots \\ C_n \end{bmatrix}$$

La méthode d'Euler permettant de résoudre une équation différentielle d'ordre 1 peut alors s'appliquer. Les grandeurs manipulées par les algorithmes ne seront plus de dimension 1 mais deviendront des vecteurs de dimension n. Ce formalisme permettra de coder très simplement un algorithme de résolution d'équations différentielles d'ordre n. C'est également ce formalisme qui est utilisé par tous les solveurs de MATLAB qui permettent de résoudre les équations différentielles (ode45, ode23...). Il est donc indispensable pour un utilisateur de prendre en main ce formalisme.

2. Codage en langage MATLAB

Pour calculer les solutions d'une équation différentielle d'ordre n, nous allons créer la fonction **my_ode** (f,a,b,CI,nb_points) (Figure 755).



Figure 755 : fonction my_ode (f, a, b, CI, nb_points)

Cette fonction prendra en arguments :

f : fonction contenant l'expression de f(t,y). Cette fonction sera un vecteur colonne de dimension n.

a : borne inférieure de l'intervalle d'étude

 ${\bf b}$: borne supérieur de l'intervalle d'étude

CI : condition initiale de l'équation différentielle. La condition initiale sera un vecteur colonne de dimension n.

nb_points : nombre de points à calculer

La fonction aura comme sorties :

t: vecteur contenant les instants correspondant aux points ou les valeurs de la solution sont calculées **sol**: matrice contenant les valeurs de toutes les fonction y_i définie dans la méthode. La structure de cette matrice est donnée sur la Figure 756. La première colonne de cette matrice regroupe les valeurs calculées de la fonction solution recherchée y(t).



Figure 756 : structure de la matrice sol

La Figure 757 montre le codage de la fonction **my_ode** (f,a,b,CI,nb_points).

```
%% my ode
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Résolution d'une équation différentielle mise sous la forme d'un problème
% de Cauchy
 Equation vectorielle: dy/dt = F(t,y)
function [t,sol] = my ode colonne(f,a,b,CI,nb points)
h = (b-a)/nb points;
t = [a:h:b];
% La première ligne de la matrice solution est constituée du vecteur
% condition initiales CI
sol(1,:) = CI;
% Chaque ligne de la matrice est construite en utilisant le schéma d'Euler
% Il est nécessaire de transposer l'expression "h * f(t(k),sol(k,:))" afin
% de la rendre de même dimension que les autres termes de l'équation
for k = 1:1:nb_points
    sol(k+1,:) = sol(k,:) + h * f(t(k),sol(k,:))';
end
end
```

Figure 757 : codage de la fonction my_ode (f,a,b,CI,nb_points)

Afin de tester notre solveur **my_ode**, nous allons utiliser un script qui va rechercher la solution d'une équation différentielle du second ordre de trois manières différentes :

- En utilisant **my_ode**
- En utilisant ode23 (solveur intégré à MATLAB)
- En recherchant la solution analytique en calcul symbolique

L'équation sera la suivante :

$$\begin{cases} 5 y''(t) + y'(t) + y(t) = 2 \\ y(0) = 0 \text{ et } y'(0) = 0 \end{cases}$$

Que l'on peut mettre sous la forme :

ſ

$$\begin{cases} y_1'(t) = y_2(t) \\ y_2'(t) = \frac{2 - y_2(t) - y_1(t)}{5} \Rightarrow F(t, Y) = \begin{bmatrix} y_2(t) \\ \frac{2 - y_2(t) - y_1(t)}{5} \end{bmatrix} \\ Y_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{cases}$$

$$\begin{cases} y_1'(t) = y_2(t) \\ y_2'(t) = \frac{2 - y_2(t) - y_1(t)}{5} \\ Y_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{cases}$$

$$Y'(t) = F(t, Y(t)) \text{ avec } F(t, Y) = \begin{bmatrix} y_2(t) \\ \frac{2 - y_2(t) - y_1(t)}{5} \end{bmatrix} \text{ et } Y_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

L'expression mathématique de cette équation différentielle sera définie dans la fonction **F_2.m** (Figure 758). Cette expression peut être modifiée si l'on souhaite résoudre une autre équation différentielle du second ordre. Dans cette fonction la sortie **F_2** calculée sera une matrice (2,1).

```
%% F_2.m
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% F_2.m contient l'expression de la fonction F(t,y) utilisée pour résoudre
% une équation différentielle du second ordre en utilisant la méthode d'Euler
% cette fonction correspond à l'équation différentielle suivante:
% 5.d2y/dt2 + dy/dt + y = 2
% f(t,y) est une matrice 2-by-1 tel que :
% f(1) = y(2)
% f(2) = 2-y(2)-y(1))/5
function f = F_2(t,y);
f = [y(2) ; (2-y(2)-y(1))/5];
end
```

Figure 758 : codage de la fonction F_2.m qui contient l'expression de la fonction f(t,y)

Ouvrir le script num_ode_comparaison.m (Figure 759).

Ce script va permettre de tracer dans une même fenêtre graphique la solution calculée avec **my_ode**, **ode23** et la **solution analytique**.

Il est possible de définir en début de script l'intervalle de calcul [a ; b] et le nombre de points **nb_points** utilisés pour le calcul.

```
%% num_ode_comparaison
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% comparaison de my_ode et ode_23 pour la résolution d'une équation
% différentielle du deuxième ordre
% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 40;
% spécification du nombre de points et du pas h
nb_points = 100;
h= (b-a) / nb_points;
```

```
%% Résolution d'une équation différentielle du second ordre en utilisant
% des solveurs numériques
% spécification des conditions initiales CI = [y0 ; dydt0] (matrice 2-by-1)
CI = [0;0];
% Résolution avec my ode
[t my ode 2, sol my ode 2] = my ode(@F 2, a, b, CI, nb points);
% Résolution avec ode23
% pour utiliser ode23 on rentre directement en argument le vecteur temporel
% nécéssaire au calcul (tspan). Dans my ode, il fallait rentrer [a;b] et le
nombre
% de points ce qui revient exactement au même
tspan = [a:h:b];
[t ode23 2, sol ode23 2] = ode23(@F 2,tspan,CI);
%% Résolution analytique en calcul symbolique
%définition des variables et de la fonction symbolique
syms t;
syms y(t);
% définition des dérivées symboliques successives de f(t)
Dly=diff(y,1);
D2y=diff(y,2);
% Construction de l'équation à résoudre et résolution à l'aide de la
% fonction dsolve et spécification des conditions initiales
equ=5*D2y+D1y+y==2;
sol symbolique(t)=dsolve(equ,y(0)==CI(1),D1y(0)==CI(2));
% calcul des valeurs du vecteur sol ana qui contient les valeurs de la
% solution calculée analytiquement
t ana = [a:h:b];
sol ana = sol symbolique(t ana);
% conversion de la fonction symbolique en double précision
sol ana = double(sol ana);
%représentation graphique de la solution
figure;
hold all;
% Tracé de la première colonne de la matrice sol qui correcspond à la
% fonction y, solution de l'équation différentielle
plot(t_my_ode_2, sol_my_ode_2(:,1), 'b', 'LineWidth',2);
plot(t ode23 2, sol ode23 2(:,1), 'k--', 'LineWidth',2);
plot(t ana, sol ana, 'r');
grid on; grid minor;
title({'Comparaison des solveurs pour la résolution',...
    'd''une équation différentielle du second ordre', string (nb points) + ' points
sont calculés '});
legend ('Solution calculée avec my ode', 'Solution calculée avec ode23',...
    'Solution calculée analytiquement');
```

Figure 759 : comparaison des différentes méthodes de résolution d'une équation différentielle

Exécuter le script avec un nombre de points calculés très faible (nb_points = 100) et observer le résultat



Figure 760 : comparaison des différentes méthodes de résolution d'une équation différentielle (100 points calculés)

Il apparaît que notre solveur **my_ode** nous donne une approximation de la solution avec une erreur très importante. Le solveur **ode23** est très proche de la solution analytique exacte et s'avère bien plus performant.

La simplicité du codage que nous avons adopté pour coder **my_ode** (6 lignes de code) le rend très sensible au nombre de points retenus pour le calcul. En toute logique et conformément à la méthode utilisée pour coder notre solveur la diminution du pas et l'augmentation du nombre de points calculés devrait améliorer considérablement sa précision.

Relancer le script en choisissant un nombre de point plus important (nb_points = 1000).



Figure 761 : comparaison des différentes méthodes de résolution d'une équation différentielle (1000 points calculés)

Nous pouvons constater que le solveur **my_ode** nous fournit une solution acceptable avec 1000 points calculés.



Figure 762 : zoom sur la courbe

A partir de 10000 points notre solveur donne le même ordre de grandeur de précision que **ode23**. Le temps de calcul est par contre considérablement augmenté.



Figure 763 : comparaison des différentes méthodes de résolution d'une équation différentielle(10000 points calculés)



Figure 764 : zoom sur la courbe

3. Robustesse du processus. Résolution d'une équation différentielle d'ordre 3

Afin de tester notre solveur sur une équation différentielle d'ordre supérieur à 2, nous allons considérer l'équation différentielle du troisième ordre suivante :

$$\begin{cases} y'''(t) + 3 y''(t) + 3 y'(t) + y(t) = 10 \\ y(0) = 1, y'(0) = 0 \text{ et } y''(0) = 3 \end{cases}$$

Que l'on peut mettre sous la forme :

$$\begin{cases} y_1'(t) = y_2(t) \\ y_2'(t) = y_3(t) \\ y_3'(t) = 10 - 3 \ y_3(t) - 3 \ y_2(t) - y_1(t) \\ y_2(t) \\ y_2(t) \\ 10 - 3 \ y_3(t) - 3 \ y_2(t) - y_1(t) \end{bmatrix} \\ \begin{cases} y_1'(t) = y_2(t) \\ y_2'(t) = y_3(t) \\ y_2'(t) = 10 - 3 \ y_3(t) - 3 \ y_2(t) - y_1(t) \\ y_0 = \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix} \end{cases}$$
$$Y'(t) = F(t, Y(t)) \text{ avec } F(t, Y) = \begin{bmatrix} y_2(t) \\ y_2(t) \\ y_2(t) - y_1(t) \\ y_2(t) - y_1(t) \end{bmatrix} \text{ et } Y_0 = \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix}$$

L'expression mathématique de cette équation différentielle sera définie dans la fonction **F_3.m** (Figure 765). Cette expression peut être modifiée si l'on souhaite résoudre une autre équation différentielle du second ordre. Dans cette fonction la sortie **F_3** calculée sera une matrice (3,1).

```
%% F_3.m
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% F_2.m contient l'expression de la fonction F(t,y) utilisée pour résoudre
% une équation différentielle du second ordre en utilisant la méthode d'Euler
% cette fonction correspond à l'équation différentielle suivante:
% d3y/dt3 + 3.d2y/dt2 + 3dy/dt + y = 10
% f(t,y) est une matrice 3-by-1 tel que :
% f(1) = y(2)
% f(2) = y(3)
% f(3) = 10 - 3*y(3) - 3*y(2) - y(1))/5
function f = F_3(t,y);
f = [y(2) ; y(3) ; (10 - 3*y(3) - 3*y(2) - y(1))/5];
end
```

Figure 765 : codage de la fonction F_3.m

Ouvrir le script solve_ode_3.m (Figure 766).

Ce script va permettre de tracer dans une même fenêtre graphique la solution de cette équation différentielle du troisième ordre calculée avec **my_ode** et avec **ode23**. Il n'y a aucune modification à faire sur la fonction **my_ode** qui peut traiter des matrices de toute dimension.

Il est possible de définir en début de script l'intervalle de calcul [a ; b] et le nombre de points **nb_points** utilisés pour le calcul.

```
%% solve ode 3
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% comparaison de my_ode et ode_23 pour la résolution d'une équation
% différentielle du troisième ordre
% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 40;
% spécification du nombre de points et du pas h
nb points = 4000;
h=(b-a) / nb points;
%% Résolution d'une équation différentielle du second ordre en utilisant
% des solveurs numériques
% spécification des conditions initiales CI = [y0 ; dydt0] (matrice 2-by-1)
CI = [1;0;3];
% Résolution avec my ode
[t_my_ode_3, sol_my_ode_3] = my_ode(@F_3, a, b, CI, nb_points);
% Résolution avec ode23
% pour utiliser ode23 on rentre directement en argument le vecteur temporel
% nécéssaire au calcul (tspan). Dans my ode, il fallait rentrer [a;b] et le
% nombre de points ce qui revient exactement au même
tspan = [a:h:b];
[t_ode23_3, sol_ode23_3] = ode23(@F_3, tspan, CI);
%représentation graphique de la solution
```

```
figure;
hold all;
% Tracé de la première colonne de la matrice sol qui correspond à la
% fonction y, solution de l'équation différentielle
plot(t_my_ode_3,sol_my_ode_3(:,1),'b','LineWidth',2);
plot(t_ode23_3,sol_ode23_3(:,1),'k--','LineWidth',2);
grid on; grid minor;
title({'Comparaison des solveurs pour la résolution',...
'd'une équation différentielle du troisième ordre',string(nb_points) + '
points sont calculés '});
legend('Solution calculée avec my ode','Solution calculée avec ode23');
```

Figure 766 : résolution d'une équation différentielle d'ordre 3 avec my_ode et ode23

Exécuter le script et observer le résultat sur la Figure 767.



Figure 767 : solution d'une équation différentielle du troisième ordre

Le calcul étant réalisé avec 4000 points, le résultat obtenu avec **my_ode** est satisfaisant.

4. Robustesse du processus. Application à la résolution de l'équation de Van der Pol

La documentation de la fonction **ode23** de MATLAB propose la résolution de l'équation différentielle de Van der Pol.

 $y_1''(t) - (1 - y^2(t)) + y(t) = 0$

Ouvrir le script **solve_ode_Van_der_Pol.m** permet de tester la résolution de cette équation à l'aide de **my_ode** et de **ode23**. La fonction F(t,y) correspondant à l'équation de Van der Pol est définie en fin de script.

```
%% solve ode Van der Pol
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% comparaison de my ode et ode 23 pour la résolution de l'équation
% différentielle de Van der Pol (@ Mathworks documentation)
% La fonction qui définit la fonction F(t,y) est définie en fin de script
% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 40;
% spécification du nombre de points et du pas h
nb points = 10000;
h= (b-a) / nb points;
%% Résolution de l'équation différentielle de Van der Pol
% spécification des conditions initiales CI = [y0 ; dydt0] (matrice 2-by-1)
CI = [2;0];
% Résolution avec my ode
[t_my_ode_vdp,sol_my_ode_vdp] = my_ode(@vdp,a,b,CI,nb_points);
% Résolution avec ode23
% pour utiliser ode23 on rentre directement en argument le vecteur temporel
% nécéssaire au calcul (tspan). Dans my ode, il fallait rentrer [a;b] et le
nombre
% de points ce qui revient exactement au même
tspan = [a:h:b];
[t ode23 vdp, sol ode23 vdp] = ode23(@vdp, [a b], CI);
%représentation graphique de la solution
figure;
hold all;
% Tracé de la première colonne de la matrice sol qui correspond à la
% fonction y, solution de l'équation différentielle
plot(t my_ode_vdp,sol_my_ode_vdp(:,1),'c','LineWidth',2);
plot(t ode23 vdp,sol ode23 vdp(:,1),'k--','LineWidth',2);
grid on; grid minor;
title({'Comparaison des solveurs pour la résolution',...
    'd''une équation différentielle du troisième ordre', string (nb points) + '
points sont calculés '});
legend('Solution calculée avec my ode', 'Solution calculée avec ode23');
function f = vdp(t, y)
f = [y(2); (1-y(1)^2)*y(2)-y(1)];
end
```

Figure 768 : résolution de l'équation différentielle de Van der Pol



Exécuter le script et observer le résultat sur la Figure 769.

Figure 769 : solution de l'équation différentielle de Van der Pol, 10000 points calculés

La prise en compte d'un grand nombre de points (10000) garantit pour cet exemple la précision de résolution pour notre solveur **my_ode**.

Vous pouvez donc utiliser le solveur **my_ode** pour résoudre toutes les équations différentielles d'ordre quelconques, linéaires et non linéaires à conditions de choisir un nombre important de points de calcul.

C. Applications

1. Modélisation d'un oscillateur mécanique

La modélisation d'un problème se ramène très souvent à la résolution d'une équation différentielle. Prenons l'exemple de l'oscillateur mécanique de la Figure 770. Le mouvement est déclenché par un écartement de la masse par rapport à sa position d'équilibre de $\Delta x=0.1$ m. La masse évolue ensuite librement.



Figure 770 : oscillateur mécanique avec prise en compte de la pesanteur

En isolant la masse et en appliquant le principe fondamental de la dynamique, on obtient l'équation différentielle qui caractérise l'évolution de la position x(t). en fonction du temps.

$$M v'(t) + b v(t) + K x = M g$$

En exploitant le fait que $v = \frac{dx}{dt}$, l'équation différentielle se ramène une équation différentielle du second ordre en x(t).

$$M x''(t) + b x'(t) + K x = M g$$

Les conditions initiales sont données par : $\begin{cases} x(0) = 0.1 \\ x'(0) = 0 \end{cases}$

2. Les différentes méthodes de résolution d'une équation différentielle avec MATLAB

Nous allons les différentes méthodes que nous propose MATLAB pour résoudre ce type d'équation et trouver l'évolution de x(t) en fonction du temps.

- Résolution en utilisant notre solveur **my_ode** (code MATLAB)
- Résolution en utilisant ode23 (code MATLAB)
- Résolution en utilisant le calcul symbolique (code MATLAB)
- Résolution à l'aide de Simulink en construisant un modèle Simulink de l'équation différentielle
- Résolution avec Simscape en construisant le modèle multi-physique de système masse ressort

Toutes ces méthodes vont donner des résultats satisfaisants.

3. Résolution avec my_ode, ode 23 et en calcul symbolique

L'équation différentielle du second ordre peut se mettre sous la forme d'un système d'équations différentielle du premier ordre.

$$\begin{cases} y_1' = y_2 \\ y_2' = \frac{1}{M} (M \ g - b \ y_2 - K \ y_1) \end{cases}$$

Pour utiliser les solveurs pour résoudre cette équation différentielle il faut commencer par créer la fonction qui va contenir l'équation différentielle. Le codage de la fonction **F_mass_spring_damper.m** est donné sur la Figure 771.

```
%% F mass spring damper.m
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% F mass sprog damper.m contient l'expression de la fonction F(t,y) utilisée pour
résoudre
% l'équation différentielle du second ordre correspondant à un oscillateur
% mécanique:
% M.d2y/dt2 + b.dy/dt + K.y = M.q
% F(t,y) est une matrice 2-by-1 tel que :
% f(1) = y(2)
% f(2) = M.g-b.y(2)-K.y(1))/M
function f = F mass spring damper(t,y);
% définition des grandeurs physique qui interviennent dans l'équation
m = 1;
         % mass : kg
k = 100;
          % spring rate : N/m
b = 2;
          % damping coefficient : N/(m/s)
q = 9.81; % gravity acceleration : N/kg
f = [y(2); (m*q-b*y(2)-k*y(1))/m];
end
```

Figure 771 : codage de la fonction F_mass_spring_damper.m

Le script **ode_solve_mass_spring_damper.m** permet de résoudre l'équation différentielle en utilisant le solveur **my_ode**, le solveur **ode23** et le calcul symbolique pour déterminer la solution analytique exacte.

```
%% ode_solve_mass_spring_damper.m
% Ivan LIEBGOTT @ Décembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Permet la résolution de l'équation différentielle de l'oscillateur
% mécanique en utilisant my_ode et ode23
% définition de l'intervalle d'étude [a;b] de la fonction considérée
a = 0;
b = 5;
% spécification du nombre de points
nb_points = 5000;
```

```
h= (b-a) / nb points;
%% Résolution d'une équation différentielle du second ordre en utilisant
% des solveurs numériques
% spécification des conditions initiales CI = [y0 ; dydt0] (matrice 2-by-1)
CI = [0.1;0];
% Résolution avec my ode
[t my ode, sol my ode] = my_ode(@F_mass_spring_damper, a, b, CI, nb_points);
% Résolution avec ode23
% pour utiliser ode23 on rentre directement en argument le vecteur temporel
% nécéssaire au calcul (tspan). Dans my ode, il fallait rentrer [a;b] et le
nombre
% de points ce qui revient exactement au même
tspan = [a:h:b];
[t ode23 2, sol ode23 2] = ode23(@F mass spring damper,tspan,CI);
%% Résolution analytique en calcul symbolique
%définition des variables et de la fonction symbolique
syms t;
syms y(t);
% définition des grandeurs physique qui interviennent dans l'équation
        % mass : kg
m = 1;
k = 100;
         % spring rate : N/m
          % damping coefficient : N/(m/s)
b = 2;
g = 9.81; % gravity acceleration : N/kg
% définition des dérivées symboliques successives de f(t)
Dly=diff(y,1);
D2y=diff(y,2);
% Construction de l'équation à résoudre et résolution à l'aide de la
% fonction dsolve et spécification des conditions initiales
equ = m*D2y + b*D1y + k*y ==m*g;
sol symbolique(t)=dsolve(equ,y(0)==CI(1),D1y(0)==CI(2));
% calcul des valeurs du vecteur sol ana qui contient les valeurs de la
% solution calculée analytiquement
t ana = [a:h:b];
sol ana = sol symbolique(t ana);
% conversion de la fonction symbolique en double précision
sol ana = double(sol ana);
%représentation graphique de la solution
figure;
hold all;
% Tracé de la première colonne de la matrice sol qui correcspond à la
% fonction y, solution de l'équation différentielle
plot(t my ode,sol my ode(:,1),'b','LineWidth',2);
plot(t ode23 2,sol ode23 2(:,1),'k--','LineWidth',2);
plot(t ana, sol ana, 'r');
grid on; grid minor;
title ({ 'Résolution de l''équation différentielle de l''oscillateur mécanique',...
    'en utilisant 3 méthodes différentes', string(nb points) + ' points sont
calculés '});
```

```
legend('Solution calculée avec my ode','Solution calculée avec ode23',...
'Solution calculée en symbolique');
```

Figure 772 : Calcul et tracé de la solution de l'équation différentielle en utilisant 3 méthodes de résolution

Ouvrir le script et exécuter le.



Figure 773 : résolution de l'équation différentielle avec 3 méthodes

Nous pouvons constater que les trois méthodes sont valables et donnent des résultats satisfaisants.

4. Résolution de l'équation différentielle avec Simulink

Pour résoudre une équation différentielle avec Simulink, il suffit de construire un modèle à partir de la forme suivante de l'équation :

$$x''(t) = \frac{1}{M} \left(M g - b x'(t) - K x \right)$$

Ouvrir le fichier simulink_resolution.slx.

La Figure 774 représente le modèle permettant de résoudre l'équation différentielle de l'oscillateur mécanique en utilisant Simulink.



Figure 774 : modèle permettant la résolution de l'équation différentielle avec Simulnk

Afin de prendre en compte la condition initiale x(0)=0.1, il faut compléter le bloc intégrateur correspondant à la position x comme indiqué sur la Figure 775.

🚹 Block Parameters: Integ	rator1	>
Integrator		
Continuous-time integrat	ion of the input signa	al.
Parameters		
External reset: none		•
Initial condition source:	internal	•
Initial condition:		
0.1		:
Limit output		
Wrap state		
Show saturation port		
Show state port		
Absolute tolerance:		
auto		:
☐ Ignore limit and reset when linearizing		
Enable zero-crossing	detection	
State Name: (e.g., 'posit	ion')	
"		
-		
0	OK Cance	el Help Apply

Figure 775 : spécification de la condition initiales x(0)=0.1 dans l'intégrateur de la position

Lancer la simulation et observer la réponse dans le scope de la Figure 776.


Figure 776 : visualisation de la solution x(t) déterminée avec Simulink

Nous pouvons constater que la réponse est identique à celles déterminées à l'aide des solveurs **my_ode** et **ode23** ainsi qu'en calcul symbolique.

5. Résolution de l'équation différentielle avec Simscape

Ouvrir le fichier simscape_resolution.slx.

Afin de résoudre l'équation différentielle avec Simscape, il faut construire le modèle multi-physique de la Figure 777.



Figure 777 : modèle multi_physique du système masse-ressort-amortisseur

Afin de spécifier la condition initiale, il faut indiquer que le ressort n'est pas à sa position d'équilibre en début de simulation. Pour cela, **double-cliquer** sur le ressort et sélectionner l'onglet **Variables**.

Indiquer ensuite que la déformation initiale est de 0.1 m comme indiqué sur la Figure 778.

눰 Block Para	ameters: Translat	ional Spring		×				
Translationa	al Spring							
The block represents an ideal mechanical linear spring.								
Connections R and C are mechanical translational conserving ports. The block positive direction is from port R to port C. This means that the force is positive if it acts in the direction from R to C. Source code								
Settings								
Paramete	rs Variables							
Override	Variable	Priority	Beginning Value	Unit				
	Velocity	None 💌	0	m/s ~				
	Force	None 🔻	0	N ~				
	Deformation	High 🔻	0.1					
			OK Cancel	Help Apply				

Figure 778 : spécification de la condition initiale dans Simscape

Lancer la simulation et observer la réponse dans le scope de la Figure 779.



Figure 779 : visualisation de la solution x(t) calculée avec Simscape

Nous pouvons constater que la réponse est identique à celles déterminées par les autres méthodes.

VIII. Résolution de systèmes linéaires – Méthode de Gauss

A. Présentation de la méthode

Ce chapitre présente l'application de la méthode de Gauss dans la résolution d'un système linéaire de n équations à n inconnues admettant une solution unique. La méthode permet de substituer au système initial, un système équivalent qui sera plus facile à résoudre.

Soit le système linéaire de n équations à n inconnues suivant :

$$\begin{cases} a_{11} x_1 + a_{12} x_2 + \dots + a_{1n} x_n = b_1 \\ a_{21} x_1 + a_{22} x_2 + \dots + a_{2n} x_n = b_1 \\ \vdots \\ a_{n1} x_1 + a_{n2} x_2 + \dots + a_{nn} x_n = b_1 \end{cases}$$

En posant : $A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$, $X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ et $B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$

Ce système peut se mettre sous forme matricielle : AX = B

$\begin{bmatrix} a_{11} \end{bmatrix}$	a_{12}		a_{1n}	$\begin{bmatrix} x_1 \end{bmatrix}$	$\begin{bmatrix} b_1 \end{bmatrix}$
<i>a</i> ₂₁	<i>a</i> ₂₂	•••	a_{2n}	x_2	b_2
:	:	÷	:	:	:
a_{n1}	a_{n2}	•••	a_{nn}	$\lfloor x_n \rfloor$	b_n

Connaissant les matrices A et B, l'algorithme de résolution à mettre en place doit nous permettre de trouver la solution X.

Dans un premier le système sera transformé en système triangulaire équivalent :

$$\begin{cases} a'_{11} x_1 + a'_{12} x_2 + \dots + a'_{1n} x_n = b_1 \\ 0 + a'_{22} x_2 + \dots + a'_{2n} x_n = b_1 \\ \vdots \\ 0 + 0 + \dots + a'_{nn} x_n = b_1 \end{cases} \begin{bmatrix} a'_{11} & a'_{12} & \dots & a'_{1n} \\ 0 & a'_{22} & \cdots & a'_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & a'_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b'_1 \\ b'_2 \\ \vdots \\ b'_n \end{bmatrix}$$

Afin de ne travailler que sur une seule matrice pour réaliser les différentes opérations, nous travaillerons avec la matrice augmentée M en concaténant A et B.

$$M = \begin{bmatrix} a'_{11} & a'_{12} & \dots & a'_{1n} & b'_{1} \\ 0 & a'_{22} & \cdots & a'_{2n} & b'_{2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & a'_{nn} & b'_{n} \end{bmatrix}$$

La résolution se fera alors de bas en haut en partant de la dernière équation (une équation pour une inconnue) :



Il suffira ensuite de substituer les inconnues trouvées dans les lignes inférieures pour trouver l'inconnue de la ligne supérieure. Le système est alors résolu en remontant ainsi jusqu'à la première équation.

B. Exemple

Soit le système suivant de 3 équations à 3 inconnues :

$\int x + 4y + z = 4$	(L_1)	[1	4	1	$\begin{bmatrix} x \end{bmatrix}$	[4	
$\begin{cases} 2x + 2y + z = 1 \end{cases}$	(L_2)	2	2	1	<i>y</i>	=	1	ļ
$\left(x+4y+2z=2\right)$	(L_3)	1	4	2			_2_	

1. Recherche du pivot

Nous commençons par chercher le plus grand coefficient devant la première inconnue x. Ce premier coefficient s'appelle le premier **pivot**. Dans notre exemple, le premier pivot pour l'inconnue x se trouve en L_2 .

$\int x + 4y + z = 4$	(L_1)	[1	4	1]	$\begin{bmatrix} x \end{bmatrix}$		4
$\begin{cases} 2x + 2y + z = 1 \end{cases}$	(L_2)	2	2	1	<i>y</i>	=	1
$\left(x+4y+2z=2\right)$	(L_3)	1	4	2			2

Le choix du pivot aura une incidence sur la stabilité de l'algorithme. Les choix de pivots trop petits peuvent engendrer des erreurs d'arrondis. Nous admettrons que le choix du plus grand pivot améliore la stabilité de l'algorithme.

2. Echange de lignes

Nous faisons remonter l'équation correspondant au pivot en première position.

$\int 2x + 2y + z = 1$	(L_1)	2	2	1	$\begin{bmatrix} x \end{bmatrix}$		1
$\begin{cases} x+4y+z=4 \end{cases}$	(L_2)	1	4	1	y	=	4
x + 4y + 2z = 2	(L_3)	1	4	2			2

3. Transvection - Triangularisation

Il faut maintenant faire disparaitre l'inconnue correspondant au pivot choisi dans les deux autres équations. Pour cela, nous allons réaliser des opérations de transvection :

$$\begin{bmatrix} L_2 \Rightarrow L_2 - \frac{1}{2}L_1 \\ L_3 \Rightarrow L_3 - \frac{1}{2}L_1 \end{bmatrix}$$

$$\begin{bmatrix} 2x + 2y + z = 1 \\ 3y + \frac{z}{2} = \frac{7}{2} \\ x - \frac{2x}{2} + 4y - \frac{2y}{2} + z - \frac{z}{2} = 4 - \frac{1}{2} \quad \Leftrightarrow \quad \begin{cases} 2x + 2y + z = 1 \\ 3y + \frac{z}{2} = \frac{7}{2} \\ 3y + \frac{3}{2}z = \frac{3}{2} \end{cases}$$

$$\begin{bmatrix} 2 & 2 & 1 \\ 0 & 3 & \frac{1}{2} \\ 0 & 3 & \frac{3}{2} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 7/2 \\ 3/2 \end{bmatrix}$$

Nous recommençons la recherche du plus grand pivot avec la seconde inconnue y. Le plus grand pivot pour l'inconnue y est en L_2 . Il n'y pas d'échange de lignes à faire.

$$2x + 2y + z = 1$$
$$3y + \frac{z}{2} = \frac{7}{2}$$
$$3y + \frac{3}{2}z = \frac{3}{2}$$

Il faut refaire une opération de transvection pour faire disparaitre la variable y de l'équation 3.

$$L_3 \Longrightarrow L_3 - L_2$$

$$\begin{cases} 2x+2y+z=1\\ 3y+\frac{z}{2}=\frac{7}{2}\\ 3y-3y+\frac{3}{2}z-\frac{z}{2}=\frac{3}{2}-\frac{7}{2} \end{cases} \Leftrightarrow \begin{cases} 2x+2y+z=1\\ 3y+\frac{z}{2}=\frac{7}{2}\\ z=-2 \end{cases} \qquad \begin{bmatrix} 2 & 2 & 1\\ 0 & 3 & 1/2\\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x\\ y\\ z \end{bmatrix} = \begin{bmatrix} 1\\ 7/2\\ -2 \end{bmatrix}$$

4. Résolution

ſ

Le processus de résolution peut alors débuter.

On résout pour commencer l'équation du bas (L_3) :

z = -2

On injecte la valeur de z pour déterminer y dans la seconde équation (L_2) :

$$3y + \frac{z}{2} = \frac{7}{2} \Longrightarrow y = \frac{1}{3}\left(\frac{7}{2} - \frac{z}{2}\right) \Longrightarrow \boxed{y = \frac{3}{2}}$$

On injecte les valeurs de y et z pour déterminer x dans la première équation (L_1) :

$$2x + 2y + z = 1 \Longrightarrow x = \frac{1}{2}(1 - 2y - z) \Longrightarrow \boxed{x = 0}$$

La solution recherchée est
$$\boxed{X = \begin{bmatrix} 0\\ 1.5\\ \end{bmatrix}}$$

C. Codage en langage MATLAB

| -2 ||

Vous pouvez trouver les fichiers de toutes les fonctions présentées dans ce paragraphe dans le dossier **Algorithmique_avec_MATLAB/ Pivot de Gauss.**

Ce type d'algorithme de résolution nécessite de nombreuses opérations et peut s'avérer complexe à valider s'il n'est pas structuré rigoureusement avec des fonctions. Une bonne méthode consiste à isoler chaque fonction élémentaire de l'algorithme, de les coder dans des fichiers séparés afin de pouvoir les valider les unes après les autres. Une fois la validation de toutes les fonctions élémentaires effectuée, il sera très simple de construire une dernière fonction permettant de coordonner toutes les fonctions de notre algorithme.

1. Recherche du pivot

Cette recherche consiste à trouver la ligne correspondant au plus grand pivot (en valeur absolue). Il suffira de chercher la ligne avec le plus grand coefficient devant la variable considérée. Pour une colonne k donnée, la fonction devra parcourir tous les coefficients de la matrice qui correspondent à cette colonne en partant de l'élément A(k,k) (Figure 780).



Figure 780 : recherche du numéro de ligne du plus grand pivot

La fonction num_ligne_pivot.m (A,num_col) permet de réaliser ce processus de recherche (Figure 781).

Cette fonction prendra en arguments :

A: matrice correspondant au système à résoudre

num_col : numéro de la colonne correspondant à la recherche du pivot

La fonction aura comme sorties :

n : numéro de la colonne correspondant au plus grand pivot



Figure 781 : fonction num_ligne_pivot.m

La Figure 782 montre le codage de la fonction **num_ligne_pivot(A,num_col)**.

```
%% n = num_ligne_pivot(A,num_col)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction est utilisée par le script Pivot_de_Gauss.m
% Recherche la ligne correspondant au plus grand pivot de la colonne
% num_col pour la matrice A
function n = num_ligne_pivot(A,num_col);
% affectation du nombre de lignes et du nombre de colonne de la matrice
[nb_lignes, nb_colonnes] = size(A);
% affectation du pivot provisoire à la ligne correspondant à num_col. La
% ligne du pivot provisoire correspond à la colonne de recherche
n = num_col;
```

```
% on parcourt la colonne en effectuant une comparaison élément par élément
for k = num_col:1:nb_lignes
    if abs(A(k,num_col)) > abs(A(n,num_col))
        % on récupère la valeur du pivot si l'élément comparé est plus
        % grand que le pivot mémorisé
        n = k;
    end
end
end
```

Figure 782 : codage de la fonction num_ligne_pivot(A,num_col)

Afin de valider la fonction, nous allons faire des tests sur une matrice générée aléatoirement. **Tapez** dans la fenêtre de commande la commande suivante :

```
>> A = randi(50,5)
A =
39 15 26 34 13
22 23 48 15 12
5 27 32 34 34
14 23 48 35 43
8 44 13 4 18
```

La commande randi(50,5) permet de créer une matrice 5x5 en choisissant des nombres entiers aléatoires entre 1 et 50.

Testons maintenant notre fonction en recherchant le plus grand pivot de chaque colonne. Il faut que la fonction **num_ligne_pivot.m** soit dans le path de MATLAB pour que nous puissions l'utiliser dans la fenêtre de commande ou dasn un script.

Il faut bien noter que **pour chaque colonne k, la recherche commence à la ligne k** et notre algorithme ne recherche pas le plus grand élément de la colonne.

```
>> num_ligne_pivot(A,1)
```

ans =

1 (le plus grand pivot de la colonne 1 est bien en ligne 1, la recherche commence en ligne 1)

```
>> num_ligne_pivot(A,2)
```

ans =

5 (le plus grand pivot de la colonne 2 est bien en ligne 5, la recherche commence en ligne 2)

```
>> num_ligne_pivot(A,3)
```

```
ans =
```

4 (le plus grand pivot de la colonne 3 est bien en ligne 3, la recherche commence en ligne 3)

```
>> num_ligne_pivot(A,4)
```

```
ans =
```

4 (le plus grand pivot de la colonne 4 est bien en ligne 4, la recherche commence en ligne 4)

```
>> num_ligne_pivot(A,5)
ans =
```

5 (le plus grand pivot de la colonne 5 est bien en ligne 5, la recherche commence en ligne 5)

Il est également possible de tester notre fonction avec une matrice contenant des nombres positifs et négatifs.

```
>> A = randi([-50,50],5)
A =
    -23 8 15 21 -4
    27 19 18 -27 16
    -31 5 14 -38 27
    -21 -8 45 11 -15
    -41 15 -29 -5 16
>> num_ligne_pivot(A,1)
ans =
    5
>> num_ligne_pivot(A,2)
ans =
    2
```

Notre fonction num_ligne_pivot(A,num_col) est maintenant validée.

2. Echange de lignes

La fonction **echange_lignes(A,ligne_i,ligne_j)** (Figure 783) permet d'échanger les lignes i et j de la matrice A.

Cette fonction prendra en arguments : A: matrice ligne_i : ligne à échanger ligne_j : ligne à échanger

La fonction aura comme sorties :

A : matrice A modifiée avec les lignes i et j échangées.



Figure 783 : fonction echange_lignes(A, ligne_i, ligne_j)

La Figure 784 montre le codage de la fonction echange_ligne(A, ligne_i, ligne_k).

```
%% B = echange_lignes(A,ligne_i,ligne_k)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction est utilisée par le script Pivot de Gauss.m
```

```
% Permet d'échanger la ligne_i et la ligne_k dans la matrice A
function A = echange_lignes(A,ligne_i,ligne_k)
% permutation des éléments des lignes i et j
A([ligne_i ligne_k],:) = A([ligne_k ligne_i],:);
end
```

Figure 784 : codage de la fonction echange_ligne(A, ligne_i, ligne_k)

Testons maintenant notre fonction en créant une matrice aléatoirement et en testant la permutation des lignes i et j.

```
>> A = randi(100,5,5)
A =
 76 74 2 83 40
 25 40 34 43 81
 45 69 43 89 76
 69 71 28 40 38
 36 45 20 77 22
>> echange_lignes(A,1,5)
ans =
 36 45 20 77 22
 25 40 34 43 81
 45 69 43 89 76
                               (les lignes 1 et 5 ont bien permutés par rapport à la matrice A)
 69 71 28 40 38
 76 74 2 83 40
>> echange_lignes(A,2,3)
ans =
 76 74 2 83 40
 45 69 43 89 76
 25 40 34 43 81
                               (les lignes 2 et 3 ont bien permutés par rapport à la matrice A)
 69 71 28 40 38
 36 45
        20 77 22
```

Notre fonction echange_lignes(A, ligne_i, ligne_j) est maintenant validée.

3. Transvection

La fonction **transvection(A, Ligne_pivot)** (Figure 785) permet de réaliser les opérations de transvection qui éliminent l'inconnue correspondant au pivot dans toutes les autres équations.

Cette fonction prendra en arguments : A: matrice augmentée correspondant au système à résoudre Ligne_pivot : numéro de la ligne du pivot ou se trouve l'inconnue à éliminer dans les autres équations

La fonction aura comme sorties :

A : matrice A modifiée par les opérations de transvection



Figure 785 : fonction transvection(A, num_ligne_pivot)

La Figure 786 montre le codage de la fonction transvection(A,num_ligne_pivot).

```
%% A = transvection (A, num ligne pivot)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction est utilisée par le script Pivot de Gauss.m
% Permet de soustraire à la ligne Li la Ligne pivot multipliée par un
% coefficient c.
function A = transvection(A,num_ligne_pivot)
% affectation du nombre de lignes et du nombre de colonne de la matrice
[nb lignes, nb colonnes] = size(A);
% la boucle parcourt les lignes en partant de num_ligne_pivot + 1
for k = num ligne pivot + 1:1:nb lignes
    % calcul du coefficient c permettant d'éliminer l'inconnue
    c = A(k,num ligne pivot) / A(num ligne pivot,num ligne pivot);
    % opération de transvection
    A(k,:) = A(k,:) - c * A(num ligne pivot,:);
    end
end
```

Figure 786 : codage de la fonction transvection(A, num_ligne_pivot)

Testons maintenant notre fonction en créant une matrice aléatoirement et en appliquant plusieurs fois la fonction transvection à la matrice M afin de la triangulariser.

```
>> A= transvection(A,1)
A =
 5.0000 14.0000 16.0000 1.0000
        -24.2000 -32.8000 0.2000
    0
        12.4000 12.6000 17.6000
    0
    0
        -16.4000 -29.6000 7.4000
>> A = transvection(A,2)
A =
 5.0000 14.0000 16.0000 1.0000
         -24.2000 -32.8000 0.2000
    0
         0.0000 -4.2066 17.7025
    0
    0
         0
                  -7.3719 7.2645
>> A = transvection(A,3)
```

```
A =

5.0000 14.0000 16.0000 1.0000

0 -24.2000 -32.8000 0.2000

0 -0.0000 -4.2066 17.7025

0 0.0000 0 -23.7583
```

Notre fonction transvection(A,num_ligne_pivot) est maintenant validée.

4. Triangularisation

La fonction triangularisation(A) (Figure 787) va triangulariser la matrice augmentée.

Cette fonction prendra en arguments : A: matrice augmentée correspondant au système à résoudre

La fonction aura comme sorties :

A : matrice A triangularisée



Figure 787 : fonction triangularisation(A)

La Figure 788 montre le codage de la fonction triangularisation(A).m.

```
%% A = triangularisation(A)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction est utilisée par le script Pivot de Gauss.m
% Cette fonction permet de triangulariser la matrice augmentée du système
function A = triangularisation(A)
% affectation du nombre de lignes et du nombre de colonne de la matrice
[nb lignes, nb colonnes] = size(A);
% la boucle parcourt les lignes, fait remonter la ligne de plus grand
% pivot en utilisant la fonction echange lignes et réalise les opérations de
% transvection en utilisant la fonction transvection afin de triangulariser la
% matrice
for k = 1:1: (nb lignes-1)
ligne pivot = num ligne pivot(A,k);
   % remonté du plus grand pivot
   if ligne pivot ~= k
        A = echange lignes(A, ligne pivot, k);
   end
    % opérations de transvections pour toutes les lignes sous la ligne du
    % pivot
    for p = k:1:nb lignes-1
        A = transvection(A, p);
    end
```

end end

Figure 788 : codage de la fonction triangularisation(A)

Afin de valider la fonction triangularisation nous allons construire une matrice augmenter de 5 lignes et 6 colonnes.

>	>> A = rand(5,5,6)							
A	=							
	1	1	1	2	3	4		
	2	3	2	2	4	5		
	5	1	3	1	4	3		
	1	5	1	2	1	1		
	2	1	5	1	1	5		
>	>> triangularisation(A)							

ans =

5.0000	1.0000	3.0000	1.0000	4.0000	3.0000
0	2.6000	0.8000	1.6000	2.4000	3.8000
0	0	0.1538	1.3077	1.4615	2.2308
0	0	0	8.0000	6.0000	9.0000
0	0	0	0	-12.620	-15.1875

Notre fonction **triangularisation(A)** est maintenant validée. La matrice est prête pour la phase de résolution.

5. Résolution

La fonction **resolution(M)** (Figure 789) va résoudre le système en déterminant la valeur des inconnues.

Cette fonction prendra en arguments : M: matrice augmentée correspondant au système à résoudre

La fonction aura comme sorties :

X : vecteur contenant les solutions du système d'équation



Figure 789 : fonction résolution.m

La matrice augmentée M comporte n lignes et p=n+1 colonnes et le vecteur X , solution du système

$$AX = B$$
 est de la forme $X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$.

A ce stade, la matrice M a été triangularisée et sa forme avant la phase de résolution est la suivante :

$$M = \begin{bmatrix} a'_{11} & a'_{12} & \dots & a'_{1n} & b'_{1} \\ 0 & a'_{22} & \dots & a'_{2n} & b'_{2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & a'_{nn} & b'_{n} \end{bmatrix} = \begin{bmatrix} M(1,1) & M(1,2) & \dots & M(1,n) & M(1,n+1) \\ 0 & M(2,2) & \dots & \dots & \vdots \\ \vdots & \vdots & \vdots & M(n-1,n+1) \\ 0 & 0 & 0 & M(n,n) & M(n,n+1) \end{bmatrix}$$

Nous commençons par résoudre l'équation du bas correspondant à la ligne n:

$$x_n = \frac{M(k, n+1)}{M(n, n)}$$

Ensuite, nous passons à la ligne (n-1), en utilisant la valeur de x_n trouvée précédemment :

$$x_{n-1} = \frac{1}{M(n-1,n-1)} \Big[M(n-1,n+1) - M(k,n) \cdot X(n) \Big]$$

Cette relation peut être généralisée au rang k et les autres solutions x_k sont données par la relation suivante qui sera implémentée dans l'algorithme :

$$x_{k} = \frac{1}{M(k,k)} \left[M(k,n+1) - \sum_{i=k+1}^{n} M(k,i) \cdot X(i) \right]$$

La Figure 790 montre le codage de la fonction **resolution(M)**.

```
%% M = résolution(M)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Cette fonction est utilisée par le script Pivot de Gauss.m
% Résolution d'un système triangulaire
function X = resolution(M)
[nb lignes, nb colonnes] = size(M);
X = zeros(nb lignes,1);
% on parcourt les lignes en partant du bas et en remontant
for k = nb lignes:-1:1
   somme = 0;
    % calcul de la somme en injectant les valeurs des inconnues déjà
   % déterminées
   for i = k+1:1:nb lignes
        somme = somme + M(k,i) * X(i);
```

```
end
% calcul de la solution X(k)
X(k) = (1/M(k,k))*(M(k,nb_lignes + 1) - somme);
end
end
```

Figure 790 : codage de la fonction resolution(M)

Afin de tester notre fonction **resolution(M)**, nous allons saisir la matrice triangulaire augmentée qui correspond au système donné en exemple en début de paragraphe. Ensuite nous allons la triangulariser puis la résoudre à l'aide des fonctions que nous avons définies.

>> M = [1414;2211;1422] M = 1 4 1 4 2 2 1 1 1 4 2 2 >> X = resolution(triangularisation(M)) X =

-2.0000 Nous pouvons vérifier que l'algorithme nous donne bien la solution de notre système.

6. Pivot de Gauss

0 1.5000

La fonction **Pivot_de_Gauss(A,B)** (Figure 791) prendra en argument les matrice A et B telles que la système à résoudre soit de la forme AX=B.

Cette fonction prendra en arguments :

A: matrice A telle que le système à résoudre soit de la forme AX=B

B: matrice B telle que le système à résoudre soit de la forme AX=B

La fonction aura comme sorties :

X : vecteur contenant les solutions du système d'équation AX=B



Figure 791 : fonction Pivot_de_Gauss(A,B)

La Figure 792 montre la hiérarchisation des fonctions qui sont utilisées dans l'algorithme du pivot de Gauss.



Figure 792 : hiérarchisation des fonctions utilisées dans l'algorithme du pivot de Gauss

La Figure 793 montre le codage de la fonction **Pivot_de_Gauss(A,B)**.

```
%% X = Pivot_de_Gauss(A,B)
% Ivan LIEBGOTT @ Novembre 2019
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
% Algorithme de résolution d'un système d'équation linéaire en utilisant la
% méthode du pivot de Gauss
% le système est de la forme [A][X] = [B]
% On forme la matrice augmentée
function X = Pivot de Gauss(A,B)
% concaténation de A et de B pour former la matrice augmentée M
M = [A B];
% triangularisation de M
M = triangularisation (M);
% résolution de M
X = resolution(M)
end
```

Figure 793 : codage de la fonction Pivot_de_Gauss(A,B)

Afin de tester notre fonction Pivot_de_Gauss.m(A,B), nous allons créer les deux matrice A et B correspondant à notre système et utiliser notre fonction.

>> A = [1 4 1; 2 2 1; 1 4 2]

A = 1 4 1 2 2 1 1 4 2 >> B = [4;1;2] B = 4 1 2 >> Pivot_de_Gauss(A,B) X = 0 1.5000 -2.0000

Nous pouvons vérifier que la fonction nous donne bien la solution du système d'équations.

D. Robustesse

Afin d'évaluer la robustesse de notre algorithme, nous allons comparer les résultats trouvés avec un calcul réalisé par MATLAB. En langage MATLAB en tapant A\B dans la fenêtre de commande, MATLAB donne la solution X du système d'équation.

>> A\B ans = 0 1.5000

-2.0000

Nous allons tester notre fonction sur un système de 10 équations à 10 inconnues

```
>> A = randi(100,10,10);

>> B = randi(100,10,1);

>> A\B

ans =

0.4849

-0.6699

0.3289

0.6217

-0.3165
```

0.1280
0.1572
1.5665
-0.8841
-0.6414

>> Pivot_de_Gauss(A,B)

X =

0.4849 -0.6699 0.3289 0.6217 -0.3165 0.1280 0.1572 1.5665 -0.8841 -0.6414

Nous pouvons constater que les solutions données par MATLAB sont identiques aux solutions déterminées à l'aide de notre algorithme.

I. Présentation

MATLAB/Simulink est également utilisable en version online. Il n'est donc pas forcément nécessaire d'installer MATLAB/Simulink sur son ordinateur personnel pour utiliser les fonctionnalités du logiciel. Une simple connexion à MATLAB/Simulink online à l'aide d'un login et d'un mot de passe permettra de développer des modèles ou du code. La capacité de calcul est déportée vers des serveurs et l'ordinateur joue alors le rôle d'un simple afficheur. Cette solution sera particulièrement intéressante pour les utilisateurs qui ne dispose pas d'un ordinateur performant ou pour ceux qui ne souhaite pas utiliser de l'espace sur leur disque dur avec les fichiers d'installation. Cette solution permet également d'utiliser MATLAB/Simulink dans les laboratoires d'enseignement sans procéder à l'installation et en travaillant toujours sur la dernière version du logiciel avec les étudiants. Ces derniers pourront sans difficultés travailler sur leurs fichiers à la maison en se connectant à MATLAB/Simulink online. L'utilisation de MATLAB/Simulink online est étroitement liée à l'exploitation du cloud dédié de 5Go, appelé MATLAB Drive et associé à chaque compte Mathworks. Il sera ainsi très simple d'accéder aux fichiers depuis n'importe quel ordinateur (Figure 794). L'outil MATLAB Drive Connector permettra de lier le dossier de MATLAB Drive au gestionnaire de fichier de Windows afin de faciliter l'exploitation de ces fichiers. Les fichiers sont alors stockés sur le disque dur de l'ordinateur et sont synchronisés avec le cloud.



Figure 794 : MATLAB/Simulink online et MATLAB Drive

II. Installation de l'environnement de travail à distance

Afin d'utiliser MATLAB/Simulink online, procéder à l'installation de **MATLAB Drive**, puis de **MATLAB Drive Connector** à partir de liens suivants :

Lien pour installer **MATLAB Drive** : <u>https://www.mathworks.com/products/matlab-drive.html</u>

Lien pour installer MATLAB Drive Connector :

https://www.mathworks.com/products/matlab-drive.html#matlab-drive-connector

Une fois l'installation réalisée, vous pouvez voir apparaître votre dossier MATLAB Drive dans la version de MATLAB installée sur votre ordinateur (Figure 795)

📣 MATLAB R2020a - primary and seconda	ary school use			– 🗆 X
HOME PLOTS AF	PPS	🖥 / 🖣 🏗 🧿 🖻 🖻 🖉	earch Documentation	n 🛛 🔎 🌻 Sign In .
New New New Open Cor Script Live Script	d Files Import Save Open Variable Data Workspace Clear Workspace VARIABLE	Image: Code Image: Code Favorites Image: Code Image: Code Image: Code	ENVIRONMENT RES	SOURCES
Current Folder	ssier MATLAB and Window		Workspace	• 🗡
□ Name ▲	Drive		Name 🔺	Value
Folder 18/07/202 B MbbileSensorData 18/07/202 B My_models 11/06/202 Published 28/01/201 Shared 29/07/202 Shared 18/07/202 Shared 18/07/202 Shared 18/07/202 Nared 18/07/202 No details available 18/07/202	20 1Folder 20 1Folder 15 1Folder 20 0Scrint Contenu du dossier MATLAB Drive		٤	>

Figure 795 : Dossier MATLAB Drive dans la version installée de MATLAB/Simulink

Vous pouvez également visualiser l'interface de MATLAB Drive Connector dans la barre de tâche de Windows (en bas à droite de votre écran) (Figure 796).





Cliquer sur l'icône de MATLAB Drive Connector afin d'ouvrir l'interface (Figure 797).



Figure 797 : interface de MATLAB Drive Connector

Il est possible à partir de cette interface :

- D'accéder à MATLAB Drive online
- D'accéder au dossier MATLAB Drive à partir du gestionnaire de fichiers de Windows
- D'accéder aux paramètres de MATLAB Drive pour indiquer la localisation du dossier MATLAB Drive sur le disque dur de l'ordinateur.

Nous verrons plus tard dans ce chapitre comment exploiter cette plateforme de travail.

III. Utilisation de MATLAB/Simulink online

Pour utiliser MATLAB online, il suffit de cliquer sur le lien suivant et de renseigner le login et le mot de passe de son compte Mathworks.

Lien vers MATLAB online :

https://www.mathworks.com/products/matlab-online.html

Une fois les informations d'authentification saisies, la fenêtre de travail de MATLAB/Simulink online apparaît (Figure 798).

📣 MATLAB Online R	2020a ×	+										-	□ ×
(←) → C' @	🗊 🔒 htt	tps://matlab.r	mathworks.com				⊽ է	C Re	chercher		<u>↓</u> III\	0	€ ≣
													»
HOME PLOT	rs app	PS								- ? -	Search Docu	nentation 🤇	🔪 Ivan 👻
3 5 to 🕫	Upload 🛛 🛁 Go	o to File 🚽	5 🛃	»	"		O Preferences	&	2 A Community				
New New New 🖑 Script Live Script 👻) Download 🗔 Fir	nd Files Impo Dat	ort Clear a Workspace -	Favorites Cor	Clear nmands 🔻	Simulink Lay	out 🛄 Parallel 👻	Add-Ons •	Help				
FILE			VARIABLE	CODE		SIMULINK	ENVIRONMENT		RESOURCES				<u> </u>
	MATLAB Drive >		0										•
Name		Type	• >>										
Intrice Intrice Intrice	ie l	Folder											
MobileSensorData	i	Folder											
My_models	I	Folder											
Published (my site)	F	Folder											
Shared	I	Folder											
🖄 test_trace_ACC.m	1	m											
WORKSPACE			0										
Name Value	Size Cla	SS											
14													

Figure 798 : fenêtre de MATLAB online

Il est maintenant possible de travailler dans l'environnement online, les fonctionnalités et les menus sont strictement identiques à la version MATLAB installée sur le PC.

A. Création d'un script élémentaire avec MATLAB online



Sauvegarder le script en cliquant sur

Ivan LIEBGOTT - Modélisation et simulation des systèmes multi-physiques avec MATLAB/Simulink 2020b

Le fichier est alors automatiquement sauvegardé dans le dossier courant de MATLAB Drive et sera accessible dans le cloud.



Cliquer sur Run pour exécuter le script et visualiser le résultat



B. Création d'un modèle élémentaire avec Simulink online



Cliquer maintenant sur simulink pour lancer Simulink online. La Simulink Start Page s'ouvre alors et vous pouvez créer un Blank Model (Figure 800).



Figure 800 : création d'un Blank Model dans Simulink online



Cliquer alors sur LIBRARY pour ouvrir la librairie Simulink.



Figure 801 : librairie de Simulink online

Vous pouvez maintenant créer vos modèles en faisant glisser les blocs dans la fenêtre du modèle. Créer un modèle simple et exécuter le.



Figure 802 : modèle créer à l'aide de Simulink online

Ouvrir le scope pour visualiser le résultat (Figure 803).



Figure 803 : résultat de la simulation

🔒 Save

Sauvegarder le fichier en cliquant sur

Le fichier est alors automatiquement sauvegardé dans le dossier courant de MATLAB Drive et sera accessible dans le cloud.

C. Exploitation d'un modèle multi-physique avec MATLAB/Simulink on line

Nous allons maintenant exécuter un modèle multi-physique complexe en utilisant MATLAB/Simulink online. Pour cela copier tout le contenu du dossier

Chapitre_3_Strategie_de_conception_d_un_modele_multi_physique / Maxpid dans le dossier MATLAB drive/My_Models.

Une fois le dossier copié, placer le dossier dans le Path de MATLAB online en cliquant avec le bouton droit de la souris sur le dossier Maxpid et en sélectionnant Add to Path/ Selected Folders and Subfolders (Figure 804).

CURRI	ENT FOLDER			Q
Name 🔺			Туре	
🕨 🛅 Ma	xpid		Folder	
	Open	Enter		
	Rename	F2		
	Create Zip File			
	🗙 Delete	Delete		
	New	•		
	Share	•		
	Indicate Files	s Not on Path		
	Add to Path	•	Selected Folders	
	Preview		Selected Folders and Subfolders	
	👗 Cut	Ctrl+X		
	🖻 Сору	Ctrl+C		
	🗎 Paste	Ctrl+V		

Figure 804 : ajout du dossier Maxpid dans le Path de MATLAB online

Ouvrir le dossier Maxpid et ouvrir le modèle MAXPID.slx.



Figure 805 : modèle du robot Maxpid ouvert dans MATLAB/Simulink online

Lancer la simulation du modèle en cliquant sur Run et observer les résultats obtenus dans le dashboard scope et dans le scope.

 (\mathbf{b})



Figure 806 : résultats de la simulation



Figure 807 : visualisation des résultats dans le scope

Nous pouvons constater que la simulation se déroule tout à fait normalement à l'exception de la fenêtre Mechanics Explorer qui n'est pas encore disponible sur la version online. Il n'est donc pas possible de visualiser l'animation de la maquette 3D. Cette fonctionnalité sera disponible sur les versions ultérieures.

D. Partager des dossiers contenant ses modèles avec d'autres utilisateurs

Il est très simple à partir du dossier MATLAB Drive de partager ses fichiers ou ses dossiers avec d'autres utilisateurs qui possèdent également un compte Mathworks et un Drive MATLAB. Cette fonctionnalité est très pratique pour mettre à dispositions des étudiants le dossier qui contient les fichiers de travail. Il existe plusieurs méthodes pour partager un dossier de MATLAB Drive.

- En utilisant la version de MATLAB installée sur son ordinateur
- En utilisant MATLAB/Simulink online
- En utilisant MATLAB Drive online

Il est possible de créer un lien qui pointera vers le dossier que l'on souhaite mettre à disposition

1. Partage d'un dossier à l'aide de la version de MATLAB installée sur son ordinateur

Sélectionner MATLAB Drive pour qu'il apparaisse dans la fenêtre Current Folder.

Cliquer avec le bouton droit de la souris sur le dossier que vous souhaitez partager et sélectionner **Share/Manage Members** (Figure 808).

Current Folder						\odot
📄 Name 🔺		Date Modified			Туре	
Folder						
MobileSensorD	lata	29/07/2020 19:1	5		Folder	
My_models		11/06/2020 10:4	4	_	Folder	
Maxpid	Open	I	Entrée		Folder	
Shared	Show in Explorer				Folder	
Script					rolder	
test_trace_4	Create Zip File				Script	
🖺 untitled2.m	Rename		F2		Script	
	Delete	:	Supprimer			
	New		1	>		
0				>		
	Share			*	Managaman	
	Share			*	ivianage memb	ers
	Compare Selecter	d Files/Folders		C	Manage link	Manage members
	Compare Against	t	:	>		Manage members
	Cut	(Ctrl+X			
	Сору		Ctrl+C			
	Paste		Ctrl+V			
	Add to Path			>		
~	Indicate Files Not	t on Path				
	Refresh	1	F5			

Figure 808 : partage d'un dossier du Drive de MATLAB

La fenêtre MATLAB Drive Sharing s'ouvre et vous permet d'inviter des membres à accéder à votre dossier. Il suffit d'indiquer l'adresse mail de leur compte Mathworks et de préciser si le membre aura un accès en simple consultation (**Can View**) ou s'il aura la possibilité de modifier les fichiers (**Can Edit**) (Figure 809).

MATLAB Drive Sharing		-		×
Invite Members to Share "Maxpid"				
Invite				
my_MATLAB_friend@gmail.com	Can Edit	•	Send	
Separate multiple emails with a semicolon or comma	Can Edit Can View			
Shared With				
Currently there are no participants.				
			Close	e

Figure 809 : définition des membres et des droits d'accès au dossier

Un fois l'invitation envoyée, l'utilisateur recevra un mail lui indiquant que vous voulez partager un dossier avec lui et pourra ensuite accéder aux fichiers contenus dans le dossier partagé.

En cliquant ensuite sur **Back**, vous revenez dans la fenêtre initiale et vous pouvez alors modifier à tout moment les droits et les membres qui peuvent accéder à votre dossier (Figure 810).

MATLAB Drive Sharing		-	- 🗆 X
Invite Members to Share "Maxpid"			
Invite			
Enter email address		Can Edit	 Send
Separate multiple emails with a semico	olon or comma		
Shared With			
ivan.liebgott@ac-nice.fr	Owner	Accepted	
my_matlab_friend@gmail.com	Can Edit	Pending	×
			Close

Figure 810 : gestion des droits d'accès et des membres

Il est également possible de créer un simple lien de téléchargement qui pointera sur le dossier partagé que vous pouvez alors diffuser (Figure 811).

🖹 Name 🔺		Date Modified	Date Modified		
Folder					
🗄 🔚 Mobil	eSen	sorData	29/07/2020 19:15		Folder
🗉 My_m	odel	S	11/06/2020 10:44		Folder
± Mi		Open	Entrée		Folder
Share		Show in Explorer			Folder
Script		· · ·			
칠 test_t		Create Zip File			Script
ٵ untitl		Rename	F2		Script
		Delete	Supprim	er	
		New		>	
	۵	MATLAB Drive		>	
		Share		⇒ ‡	Manage members
		Compare Selected Fil	les/Folders	e	Manage link
		Compare Against		>	Manage link
		Cut	Ctrl+X		
		Сору	Ctrl+C		
		Paste	Ctrl+V		
		Add to Path		>	
	~	Indicate Files Not on	Path		
		Refresh	F5		

Figure 811 : création d'un lien pour partager un dossier de MATLAB Drive

2. Partage d'un dossier en utilisant MATLAB online

A partir de la fenêtre **CURRENT FOLDER** de **MATLAB Drive**, cliquer avec le bouton droit de la souris sur le dossier que vous voulez partager et sélectionner **Share/Invite Members** ou **Share/Create Link** (Figure 812). Le reste de la procédure est identique au partage en utilisant la version installée de MATLAB.

▼ CURRENT	FOLDER			0		
Name 🔺			Туре			
🕨 🛅 Mobile	SensorData		Folder			
🔺 🛅 My_m	odels		Folder			
🕨 🗖 Maxi	bid		Folder			
🛅 Publis	Open E	nter	Folder			
Image: Share	Rename	F2	Folder			
🖺 test_t	Create Zip File		m			
🖺 untitle	🗙 Delete De	lete	m			
	New	•				
	Share	•	🔅 Invite Members			
	Indicate Files Not on F	Path	🔗 Create Link			
	Remove from Path	•				
	Preview					
	👗 Cut Ct	rl+X				
	Copy Ctr	rl+C				
	Paste Ct	rl+V				

Figure 812 : partage d'un dossier en utilisant MATLAB on line

3. Partage d'un dossier en utilisant MATLAB Drive online

Ouvrir l'interface de **MATLAB Drive Connector** dans la barre de tâche de Windows (en bas à droite de votre écran) (Figure 813).



Figure 813 : ouverture de l'interface MATLAB Drive Connector

A partir de la fenêtre de l'interface de **MATLAB Drive Connector**, cliquer sur l'icône 🕄 pour lancer **MATLAB Drive online** (Figure 814)

🚸 MATLAB Drive Connector 🦳 🗌	×	
MATLAB' Drive 🎾 🕨	• 0	
All files are up to date.		
Accès à MATLAB Drive online > updated 5 minutes ago	0 0 0	•
VisChcM4_Défaut_sldprt.STL updated 5 minutes ago	0 0	
VisBille_Défaut_sldprt.STL updated 5 minutes ago	0 0 0	
> amaxpid.camrec updated 5 minutes ago	0 0 0	
tp_cinemati ator Moteur Support-1.STL updated 5 minutes ago	0 0 0	•
R2020a now available >> Learn what's in the latest release of MATLAB and Simulink.	>	<

Figure 814 : lancement de MATLAB Drive online

MATLAB Drive online permet d'afficher l'ensemble des dossiers de **MATLAB Drive**. Cliquer avec le bouton droit de la souris sur le dossier que vous voulez partager et sélectionner **Share/Invite Members** ou **Share/Create Link**. Le reste de la procédure est identique au partage en utilisant la version installée de MATLAB.

📣 MATLAB Dri	ve						
Files							
Files	韋 Upload 🚽 🖿 Ne	ew Folder 🛛 < Sha	re 🗸 👎	Download 🖉 Rena	me 🛅 Move to 🔓 Copy to 🔓 De	elete	
Shared Content	MATLAB Drive						
Deleted Files	Name			Size	Date Modified	Owned By	
Deleteu Files	MobileSensorData				29/07/2020 19:15	Ме	
	▲ 🛅 My_models				29/07/2020 19:53	Ме	
	Maxpict	Shara	 Invite Members 		11/06/2020 10:44	Ме	
	Published F	Rename F2	F2 Create	Link	28/01/2015 12:36	Ме	
	Shared	Move to Ctrl+X Copy to Ctrl+C	Ctrl+X Ctrl+C Delete , 1 KB		29/07/2020 13:51	Ме	
	🛃 test_trace				18/07/2020 09:51	Ме	
	suntitled2.n	Upload Delete			29/07/2020 14:45	Ме	
	1	New Folder					

Figure 815 : partage d'un dossier à partir de MATLAB Drive online

I. Présentation

MATLAB peut également être utilisé à partir d'un smartphone en téléchargeant l'application **MATLAB Mobile**. Cette application permet :

- de profiter de toutes les fonctionnalités de MATLAB pour le calcul scientifique (création de variables, opérations sur les variables, tracés de courbes, calcul symbolique, résolution d'équation différentielles...)
- de monitorer les mesures issues des capteurs du téléphone (accéléromètre, capteur de champ magnétique, capteur d'orientation, gyroscope, capteur de position terrestre...). Le téléphone peut être utilisé comme capteur pour de nombreuses expériences scientifiques
- d'exécuter des scripts et de visualiser les résultats

Comme pour **MATLAB online**, la capacité de calcul est déportée vers des serveurs et le smartphone joue le rôle d'un simple afficheur, il n'est donc pas nécessaire de disposer d'un smartphone performant pour pouvoir utiliser l'application mobile. Pour pouvoir utiliser l'application mobile, il suffit de disposer d'un compte Mathworks, aucune licence n'est requise et l'application mobile est libre d'utilisation. **MATLAB Mobile** permet d'accéder au contenu de **MATLAB Drive** dans lequel l'utilisateur peut stocker des fichiers créés à partir de **MATLAB online** ou de la version de MATLAB installée sur un ordinateur. L'interaction entre **MATLAB Mobile**, **MATLAB online** et **MATLAB Drive** permet à l'utilisateur de disposer en permanence de l'ensemble de ses fichiers de travail depuis toutes les plates formes (Figure 816).



II. Les fonctionnalités de MATLAB Mobile

1. Taper des lignes de commandes

Au lancement de l'application, la fenêtre de la Figure 817 apparaît. Il est alors possible de saisir des commandes dans la fenêtre de commande et de visualiser les résultats obtenus.



Figure 817 : ouverture de MATLAB Mobile

Taper les lignes de commandes de la Figure 818 dans l'application et observer le résultat obtenu.



Figure 818 : utilisation des lignes de commandes dans MATLAB Mobile

2. Configurer le clavier MATLAB Keyboard

Afin de saisir plus facilement les lignes de commandes, il est possible d'utiliser un clavier spécialement dédié à l'utilisation de MATLAB. Pour cela suivre la procédure indiquée sur la Figure 819.



Figure 819 : configuration MATLAB Keyboard

Le clavier qui apparaît maintenant est parfaitement adapté à la saisie des lignes de commandes, il est conseillé de l'utiliser afin de gagner du temps dans le processus de saisie.

3. Accès à MATLAB Drive

Afin d'accéder à MATLAB Drive depuis MATLAB Mobile, suivez la procédure indiquée sur la Figure 820.



Figure 820 : accès à MATLAB Drive depuis MATLAB Mobile

Il est alors possible de naviguer dans les dossiers et d'accéder à tous les fichiers de MATLAB Drive.

4. Exécuter un script depuis MATLAB Mobile

Le dossier « 1 Dérivation Numérique » du chapitre « Ingénierie Numérique avec MATLAB » a été placé dans MATLAB Drive, il est donc visible depuis l'application mobile. Naviguer dans les dossiers pour ouvrir le script **num_deriv_plot_error.m** puis exécuter le script et visualiser les résultats conformément à la Figure 821.



Figure 821 : exécution d'un script depuis MATLAB Mobile

5. Acquisition des données des capteurs du smartphone avec MATLAB Mobile

MATLAB Mobile permet de relever et d'exploiter très facilement les données issues des capteurs intégrés dans le smartphone :

- Accéléromètre 3 axes
- capteur de champ magnétique 3 axes
- capteur d'orientation (tangage, roulis, lacet)
- gyroscope 3 axes
- capteur de position terrestre (lattitude, longitude, vitesse, altitude...)

Pour relever une série de mesure, suivez la procédure indiquée sur la Figure 822.


Figure 822 : configuration et démarrage d'une acquisition des mesures issues des capteurs du smartphone

En faisant défiler la fenêtre des capteurs, il est possible d'activer les autres capteurs (ce ne sera pas utile pour l'instant). Après avoir appuyé sur « START », les données des capteurs activés sont stockées dans des variables.

Réaliser alors une série de mouvements avec le smartphone puis appuyer sur « STOP » pour arrêter le processus d'acquisition. Indiquer alors le nom du fichier « **test_mesures_acc_3_axes** » qui sera automatiquement sauvegardé dans le dossier **MATLAB Drive /MobileSensorData/ test_mesures_acc_3_axes .mat**. Le fichier porte l'extension **.mat** qui indique qu'il contient des variables que l'on pourra visualiser dans le Workspace de MATLAB en utilisant la commande **load('test_mesures_acc_3_axes .mat')**.



Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive

A partir du menu Files, il est possible de voir le fichier sauvegardé dans MATLAB Drive. Le fichier est alors instantanément accessible à partir de MATLAB online ou de la version de MATLAB installée sur votre PC

6. Exploitation des données des capteurs du smartphone avec MATLAB Mobile

L'exploitation des données doit se faire à partir de MATLAB online ou de la version de MATLAB installée sur votre PC. La condition nécessaire est de pouvoir accéder à MATLAB Drive.

Ouvrir MATLAB ou MATLAB online, sur votre PC et ouvrir le dossier **MATLAB Drive** /**MobileSensorData**. Observez l'apparition du fichier de mesure dans MATLAB Drive (Figure 824).

📣 MATLAB R2020a - primar	y and secondary school use							-		×
HOME PLOTS	APPS			/ h h 500 E) ? 🖲 🧐	Search Document	ation		۶ 冬	Sign In
New New New C Script Live Script V FILE	Den Compare Import Data	Bave Wew Variable Save Øpen Variable ▼ Workspace Ø Clear Workspace ▼ VARIABLE VARIABLE	>> Favorites ▼	 Analyze Code Run and Time Clear Commands CODE 	Simulink SIMULINK		RESO	URCES		►
수 🔶 🔁 🔁 🐼 🗤 🛛	MATLAB Drive 🕨 MobileSenso	rData 🕨								- P
Current Folder			Co	mmand Window			۲	Works	pace	۲
🗋 Name 🔺	Date Modified	Туре	fx.	>>				Name		
test_acc2.mat test_mesures_acc_3 Simulink Model test_sim.slx	11/10/2019 14:36 axes.mat 18/08/2020 10:26 18/07/2020 10:12	MAT-file MAT-file Simulink Model								
test_mesures_acc_3_axes.mat	(MAT-file)		~							
H Name	Valu	re table						<		>

Figure 824 : visualisation du fichier de mesure sur le PC

Dans la fenêtre de commande, taper la commande suivante puis valider :

>> load test_mesures_acc_3_axes.mat

La variable« Acceleration » apparaît alors dans le Workspace et peut être exploité.

📣 MATLAB R2020a	- primary	and secondary :	chool use										-		×
HOME	PLOTS	APPS							h 🔓 🔊	c 6 () 💿 Search	Documental	tion 🖌	Þ 🦊	Sign In
New New Script Live Script	New Op FILE	☐ [ゐ Find Fi pen [] Comp ▼	les are Import Data	Save Workspace	B New Variable → Open Variable ▼ → Clear Workspace ▼ RIABLE	Favorites	Ar Ru Ø Cl CC	nalyze Code un and Time ear Commands VDE	Simulink	Layout	Ø Preferences Set Path ENVIRONMENT	Add-Ons	RESOURCES		Ā
< 🔶 🖬 🎘	📣 🕨 M	1ATLAB Drive 🕨	MobileSenso	orData			_								- P
Current Folder							•	Command Windo	w			\odot	Workspace		۲
🗋 Name 🔺			Date Modified	1	Туре		_	>> load('1	test_mesu:	ces_acc_	3_axes.mat	')	Name 🔺		
	nat es_acc_3_a «	1 axes.mat 1 1 (MAT-file)	1/10/2019 14: 8/08/2020 10: 8/07/2020 10:	36 26 12 Value timetable	MAT-file MAT-file Simulink Model		~	~~~					Accelerati	on	
													٢		>

Figure 825 : visualistion du fichier de mesure dans le Workspace

La variable « **Acceleration** » porte par défaut le nom du capteur qui a permis de créer la variable. Si nous avions activé d'autres capteurs durant l'acquisition, chaque capteur activé aurait générer la création d'une variable associée. Dans la fenêtre de commande, taper « Acceleration » afin de visualiser la structure de la variable (Figure 826).

>> Acceleration

Acceleration =			
844×3 <u>timetable</u>			
Timestamp	x	Y	Z
	-0.17478	6.1509	8.2626
18-Aug-2020 10:25:53.026	-0.3328	6.4861	8.3057
18-Aug-2020 10:25:53.035	-0.4166	6.6297	7.9681
18-Aug-2020 10:25:53.044	-0.64885	7.1278	7.5084
18-Aug-2020 10:25:53.053	-1.4102	7.5563	6.9554
18-Aug-2020 10:25:53.062	-1.3695	7.9131	6.3113
18-Aug-2020 10:25:53.071	-1.4749	8.2555	5.8037
18-Aug-2020 10:25:53.080	-1.0942	7.9634	5.7391
18-Aug-2020 10:25:53.089	-0.38069	7.4606	5.8013
18-Aug-2020 10:25:53.098	0.0838	7.2403	5.9426
18-Aug-2020 10:25:53.107	0.22506	7.0368	6.1341
18-Aug-2020 10:25:53.116	0.29689	6.9458	6.6106
18-Aug-2020 10:25:53.125	0.38069	6.7782	7.0296
18-Aug-2020 10:25:53.134	0.31126	6.7495	7.099
18-Aug-2020 10:25:53.143	0.13887	6.692	6.941
18-Aug-2020 10:25:53.152	0.079011	6.7878	6.8716
18-Aug-2020 10:25:53.161	-0.10535	6.9266	6.8333
18-Aug-2020 10:25:53.170	-0.61772	7.0272	6.6968
18-Aug-2020 10:25:53.179	-0.99123	7.032	6.6896
18-Aug-2020 10:25:53.188	-1.0702	7.0009	6.7854

Figure 826 : structure de la variable Acceleration

La variable **Acceleration** est de type **« timetable »** et contient les accélérations sur les 3 axes associées aux différents instants de l'acquisition.

Pour extraire la colonne des instants taper la commande suivante (Figure 827) :

>>t = Acceleration.Timestamp

```
t =
    844×1 datetime array
    18-Aug-2020 10:25:53.017
    18-Aug-2020 10:25:53.026
    18-Aug-2020 10:25:53.035
    18-Aug-2020 10:25:53.053
    18-Aug-2020 10:25:53.062
    18-Aug-2020 10:25:53.071
```

Figure 827 : extraction de la colonne des instants

La variable « **t** » de type **datatime** apparaît alors dans le Workspace. Il est possible d'extraire la colonne correspondant uniquement aux secondes en tapant la commande suivante (Figure 828) :

>>t_sec = t.Second

t_s	ec	=
	52	0170
		.0170
	53.	0260
	53.	.0350
	53.	.0440
	53.	.0530
	53.	0620
	53.	0710
	53.	0800
	53.	0890
	53.	0980

Figure 828 : extraction de la colonne des instants correspondant aux secondes

La variable **t_sec** de type double apparaît alors dans le Workspace et pourra être utilisée pour tracer une courbe.

Pour extraire la colonne correspondant aux accélérations relevées selon l'axe X, taper la commande suivante (Figure 829):

>>AX = Acceleration.X

AX = -0.1748 -0.3328 -0.4166 -0.6488 -1.4102 -1.3695 -1.4749 -1.0942 -0.3807 0.0838 0.2251 0.2969

Figure 829 : extraction de la colonne correspondant aux accélérations mesurées selon l'axe X

Saisir le script de la Figure 830 et lancer son exécution. Le script est également disponible dans le dossier « Chapitre_12_MATLAB_Mobile ».

```
8% Exploitation des données issues de l'accéléromètre d'un smartphone
% Ivan LIEBGOTT @ aout 2020
% Modélisation et Simulation des Systèmes Multi-Physiques
% avec MATLAB - Simulink
88
load test mesures acc 3 axes.mat
close all;
% création des variables permettant d'effectuer les tracés
t = Acceleration.Timestamp;
AX = Acceleration.X;
AY = Acceleration.Y;
AZ = Acceleration.Z;
% tracés des courbes d'accélération
subplot(3,1,1);
plot(t,AX);grid on;
subplot(3,1,2);
plot(t,AY);grid on;
subplot(3,1,3);
plot(t,AZ);grid on;
```



Le résultat de l'exécution du script peut être visualisé sur la Figure 831.



Figure 831 : accélérations selon les trois axes en fonction du temps

Il est maintenant possible d'utiliser cette série de données pour lui appliquer un ou plusieurs processus de traitement et d'analyse décrits dans le chapitre 10 « Ingénierie Numérique en langage MATLAB » (filtrage, intégration numérique...).

Ouvrir le fichier « circuit_RL.slx » et lancer la simulation.



Figure 832 : scopes mesurant le courant dans le circuit et la tension aux bornes de la bobine

Ouvrir le scope mesurant la tension aux bornes de la bobine et visualiser l'allure et la mise en forme de la courbe.



Figure 833 : mesure de la tension aux bornes de la bobine

Lors de l'ouverture du scope, la barre de commande est disponible sur la partie supérieure de la fenêtre (Figure 834).



Figure 834 : barre de commande du scope

La commande la plus utile est la mise à l'échelle automatique des axes 🙆. A la fin d'une simulation, il faut penser à l'utiliser dès l'ouverture du scope pour visualiser la courbe. Il est possible que lors de l'ouverture du scope la courbe ne soit pas dans la zone affichée par les axes et que rien ne soit visible sur le scope.

Il est possible de modifier la mise ne forme des courbes en cliquant sur l'icône 🔊 accessible depuis le menu déroulant **Configuration Properties**



Figure 835 : mise en forme des courbes dans un scope

Modifier les paramètres de mise en forme comme indiqué sur la Figure 835 et observer le résultats obtenus (Figure 836).



Figure 836 : modification de la mise en forme d'une courbe dans un scope

Pour afficher plusieurs courbes dans le même scope cliquer sur **Configuration Properties** et modifier le champ **Number of input ports** à **2** afin de définir deux entrées pour le scope (Figure 837).

📣 Conf	iguration	Properties: S	cope		×
Main	Time	Display	Logging		
Оре	n at simu	ulation start			
🗌 Disp	olay the f	ull path			
Numbe	r of input	t ports: 2			Layout
Sample	e time:	-1			
Input p	rocessin	g: Ele	ments as cl	nannels (sa	mple based) 🔻
Maximi	ze axes:	Off	F		•
Axes so	aling:	Ma	nual	•	<u>Configure</u>
0			ОК	Cance	el Apply

Figure 837 : modification du nombre d'entrées d'un scope

Le scope a maintenant 2 entrées. **Raccorder** la seconde entrée au signal mesurant le courant dans le circuit comme indiqué sur la Figure 838.



Figure 838 : raccordement d'un scope à deux entrées

Lancer la simulation et ouvrir le scope pour obtenir la courbe de la Figure 839.



Figure 839 : visualisation des deux courbes dans le même scope

Modifier les caractéristiques de la seconde courbe en utilisant l'icône 🔊, comme indiqué sur la Figure 840.

🐼 Style: Scope — 🗆 🗙	
Figure color: 🕭 - Plot type: Auto 🗸	
Axes colors: 🕭 🔹 🏄	
Preserve colors for copy to clipboard	
Active display: 1 V	
Properties for line: PS-Simulink Converter • • •	Choix de la
☑ Visible	seconde courbe pour la
Line: 2 🗾 🗸	mise en forme
Marker: none 🗸	
OK Cancel Apply	

Figure 840 : modification des caractéristiques de la deuxième courbe

Vous devez obtenir la configuration de la Figure 841.



Figure 841 : modification des caractéristiques de la deuxième courbe

Si vous souhaitez afficher ces deux courbes dans des graphes différents, cliquer sur l'icône **layout** \boxplus et choisir la disposition des courbes que l'on souhaite (Figure 842).



Figure 842 : modification du nombre de fenêtre dans le scope

Après mise à l'échelle des courbes à l'aide de la commande 🔯 , on obtient l'affichage représenté sur la Figure 843.



Figure 843 : affichage des courbes dans un scope sur plusieurs fenêtres

Un outil curseur est très pratique à utiliser dans les scopes, pour l'ouvrir cliquer sur l'icône **Cursor Measurement** afin de faire apparaître les curseurs. Il est alors possible d'avoir accès à tous les paramètres de mesure utiles ΔT , ΔY , coefficient directeur de la droite reliant les deux curseurs...(Figure 844).



Figure 844 : utilisation des curseurs dans un scope

Pour plus d'informations sur l'utilisation des scopes vous pouvez utiliser l'aide de MATLAB.

Annexe 2 : utilisation du signal builder

Le « signal builder » est un générateur de signaux que l'on peut paramétrer manuellement. Ce bloc est très utile pour imposer des lois de commandes.

Fonction du composant	Représentation	Bibliothèque
Source de signal paramétrable	Group 1 Signal 1 Signal Builder	Simulink/Sources

Glisser-déposer un bloc « Signal Builder » depuis la bibliothèque Simulink et ouvrir ce bloc.



Figure 845 : fenêtre de paramétrage du Signal builder

La barre de commande du signal buider permet d'accéder à de nombreuses fonctions.



Figure 846 : barre de commande du signal builder

Dans un premier temps il faut indiquer la durée du signal. Pour cela sélectionner **Axes/Change Time Range** accessible depuis la barre de commande.

承 Set the f	to	-		×
Min time: 0				
Max time: 2	s			
	C	к	Ca	ncel

Figure 847 : choix de la durée du signal dans le « signal builder »

Choisir une durée de signal de 25 s (Figure 848).



Figure 848 : modification de l'échelle de temps

Il est possible de déplacer manuellement les segments qui composent le signal. Avec le curseur de la souris, en utilisant le **glisser déposer**, déplacer horizontalement les segments verticaux et verticalement les segments horizontaux (Figure 849).



Figure 849 : déplacement manuel des segments dans le « signal builder »

Pendant le déplacement, il est possible de visualiser dans les cases « left point » et « right point » situées sous le signal la position du segment.

En sélectionnant un segment vertical (clique gauche avec la souris), il est possible de rentrer la valeur exacte de l'instant dans la case **T** : « **Left Point** ». Il est également possible de choisir exactement l'ordonnée du point d'origine et du point final dans les cases **Y** : « **Left Point** » « **Right Point** ». Les mêmes opérations peuvent être réalisées en sélectionnant un segment horizontal.

Il est également possible de déplacer uniquement un point pour obtenir des signaux de type rampe.



Figure 850 : déplacement d'un point dans le « signal builder »

Le principe de construction d'un signal quelconque est de créer des points qui seront reliés par des segments que l'on pourra ensuite déplacer librement en utilisant la méthode précédemment présentée.

Pour créer de nouveaux points, il faut utiliser la combinaison de touche **Maj+clique gauche** en plaçant les nouveaux points directement sur le signal.

Placer approximativement des nouveaux points conformément à la Figure 851.



Figure 851 : placement de nouveaux points sur le signal

Déplacer les points et les segments pour obtenir approximativement le signal de la Figure 852.



Figure 852 : mise en forme du signal par déplacement de segments et de points

Il est également possible de créer un bloc permettant de générer plusieurs signaux.

Cliquer sur les trois signaux accessibles depuis la barre de commande et choisir d'introduire de nouveaux signaux.



Figure 853 : génération de plusieurs signaux

Le bloc comporte maintenant 4 sorties.



Figure 854 : visualisation des sorties multiples du « signal builder »

Table des figures

Figure 1 : cycle en V simplifié	
Figure 2 : les compétences de l'ingénieur	
Figure 3 : triptyque Cahier des charges/Système réel/Modèle	
Figure 4 : architecture matérielle nécessaire à la mise en place de la démarche	23
Figure 5 : la phase d'expression et de spécification du besoin	24
Figure 6 : diagramme des exigences de l'asservissement de position	24
Figure 7 : la phase de Conception, Modélisation, Simulation	25
Figure 8 : relation entrée/sortie pour le moteur à courant continu	26
Figure 9 : équations de comportements du moteur à courant continu	26
Figure 10 : modélisation sous forme de schéma bloc du moteur à courant continu	27
Figure 11 : modélisation « white box » du moteur à courant continu	27
Figure 12 : modélisation par assemblage de composants du moteur à courant continu	
Figure 13 : modélisation multi-physique acausal du moteur à courant continu	28
Figure 14 : résolution des modèles et visualisation des résultats	
Figure 15 : mesure de l'évolution de la vitesse de rotation du moteur en fonction du temps	29
Figure 16 : évaluation des écarts entre les performances simulée et les performances mesurées	30
Figure 17 : essai réalisé en vue de l'estimation des paramètres inconnus	
Figure 18 : illustration du principe de l'estimation de paramètre	
Figure 19 : les réponses du modèle avant et après estimation des paramètres inconnus	
Figure 20 : modélisation de l'asservissement de position du moteur	32
Figure 21 : réponse de l'asservissement de position non corrigé	32
Figure 22 : illustration de l'utilisation d'un outil de contrôle commande pour régler un correcteur	33
Figure 23 : réponse de l'asservissement de position corrigée	33
Figure 24 : le Model-in-the-loop (MIL)	34
Figure 25 : la phase de Codage Implémentation	34
Figure 26 : structure du modèle à l'issu de la phase de Conception Modélisation Simulation	35
Figure 27 : le Software-in-the-loop(SIL)	
Figure 28 : le Processor-in-the-loop	
Figure 29 : la phase d'Intégration Vérification	
Figure 30 : architecture matérielle du hardware-in-the-loop en mode externe	
Figure 31 : le hardware-in-the-loop en mode externe: intégration du matériel dans la boucle	
Figure 32 : architecture matérielle du Hardware-In-the-Loop en mode embarqué	
Figure 33 : le Hardware-In-the-Loop mode embarqué: fonctionnement autonome du matériel	
Figure 34 : visualisation de la vitesse de sortie du moteur à l'issu de la phase Hardware-in-the-loop	
Figure 35 : la phase de Validation Recette	
Figure 36 : exemple de script écrit en langage MATLAB	
Figure 37 : le résultat obtenu après exécution du script	
Figure 38 : exemple de modélisation Simulink	43
Figure 39 : la visualisation dans un scope des résultats obtenus	
Figure 40 : exemple de modélisation Simscape	
Figure 41 : la visualisation dans un scope des résultats obtenus	
Figure 42 : Simscape et ses bibliothèques	
Figure 43 : modélisation du comportement d'une boîte de vitesse automatique avec Stateflow	
Figure 44 : les outils MATLAB pour la modélisation multi-physique	
Figure 45 : la fenêtre de l'environnement MATLAB	
Figure 46 : la barre de commande de l'environnement MATLAB	
Figure 47 : Simulink Start Page	
Figure 48 : Blank Model Simulink	
Figure 49 : la bibliothèque de Simulink	
Figure 50 : la fenêtre de l'environnement Simulink	
Figure 51 : le path de MATLAB	
Figure 52 : ajout de deossier au « Path » pour la session courante	
Figure 53 : le diagramme Chaîne d'énergie / Chaîne d'information	
Figure 54 : correspondance entre le diagramme chaîne d'énergie / Chaîne d'information et les outils MATLAB	
Figure 55 : photo du pilote automatique de bateau	
Figure 56 : correspondance entre le diagramme chaîne d'énergie / Chaîne d'information et les outils MATLAB	
Figure 57 : copie d'écran du modèle multi-physique du pilote automatique de bateau réalisé avec MATLAB	
Figure 58 : visualisation de la consigne de cap	
Figure 59 : fenêtre de visualisation de Multibody (Mechanics Explorer)	
Figure 60 : barre de visualisation de Multibody.	
Figure 61 : fenêtre de visualisation de Multibody (Mechanics Explorer)	
······································	

Figure 62 : barre temporelle de Multibody	64
Figure 63 : visualisation de la consigne de cap et du cap effectif suivi par le bateau	65
Figure 64 : visualisation des grandeurs relatives au moteur	65
Figure 65 : visualisation de l'angle de rotation de la barre	66
Figure 66 : visualisation des pressions dans les chambres avant et arrière du vérin	66
Figure 67 : visualisation des debits dans les chambres avant et arrière du verin	67
Figure 68 : modele MATLAB – Simulink de la chaine d'information du pilote automatique	68
Figure 69 : modelisation de la commande de cap avec des diagrammes d'énergie du pilete automatique.	68
Figure 70 : modèle siniscape de la partie electrique de la chaine d'energie du pilote automatique	69
Figure 71 : Inducie realise avec siniscape ridus de la partie hydraunque de la channe d'energie du phote automatique	70
Figure 72 : Modelisation a l'alte de siniscape Multibody de la su deur e du phote automatique	/ 1 72
Figure 74 : modèle du pilote bydraulique avec pilotage interactif	72
Figure 75 : utilisation du nilotage interactif d'un modèle	73
Figure 76 : présentation du robot Maxnid	75
Figure 77 : modélisation multi-physique du robot MAXPID	
Figure 78 : position du bras du robot MAXPID	
Figure 79 : visualisation de la tension de commande du moteur.	
Figure 80 : visualisation du courant dans l'induit du moteur	
Figure 81 : l'axe linéaire Control'X	79
Figure 82 : modélisation multi-physique du système Control'X	80
Figure 83 : visualisation de la vitesse et de la position de l'axe linéaire	81
Figure 84 : visualisation de la tension de commande du moteur et du courant dans l'induit	81
Figure 85 : évaluation des écarts entre les performances du modèle et celles de l'axe réel	82
Figure 86 : évaluation des écarts réel/modèle pour la vitesse et la position de l'axe	83
Figure 87 : évaluation des écarts réel/modèle pour le courant d'induit et la tension de commande	83
Figure 88 : circuit R-L	85
Figure 89 : les composants nécessaires à la modélisation du circuit R-L avec Simscape	86
Figure 90 : composants nécessaires à la modélisation du circuit RL	87
Figure 91 : commandes utiles	87
Figure 92 : Modèle Simscape du circuit R-L	88
Figure 93 : commandes utiles	88
Figure 94 : les différents types de ports de Simscape	88
Figure 95 : identification des connexions d'un modèle Simscape	89
Figure 96 : évolution de l'intensité du courant dans le circuit RL	93
Figure 97 : mise en forme des scopes	94
Figure 98 : modification de la mise en forme d'un scope	94
Figure 99 : modification de l'aspect des courbes d'un scope	95
Figure 100 : évolution de la tension aux bornes de la bobine pour le circuit RL	95
Figure 101 : evolution du courant avec une autre valeur de l'inductance	96
Figure 102 : definition du nom des signaux à analyser	97
Figure 103 : selection des signaux à suivre avec le Data Inspector	97
Figure 104 : Visualisation des signaux à suivi e avec le Data Inspector	90 00
Figure 105 : Tellette uu Data Inspector	90
Figure 100 : visualisation de plussieurs courbes à l'aide du Data Inspector	99
Figure 107 : visualisation des régultats de simulations différentes à l'aide du Data Inspector	100
Figure 100 : visualisation des resultats de sindlations différences à l'aide du Data Inspector	100
Figure 110 : modification de la couleur des courbes à l'aide du Data Inspector	101
Figure 110 : visualisation de la modification de la couleur des courbes à l'aide du Data Inspector	
Figure 112 : sélection des simulations à comparer à l'aide du Data Inspector	
Figure 113 : visualisation de la comparaison entre deux simulations à l'aide du Data Inspector	
Figure 114 : les composants nécessaires à la modélisation du circuit R-L avec Simulink	103
Figure 115 : modèle du circuit R-L réalisé avec Simulink	104
Figure 116 : Commandes utiles	104
Figure 117 : évolution de l'intensité du courant dans le circuit RL	106
Figure 118 : avantages et inconvénients des approches causales et acausales	107
Figure 119 : les ports PCP des composants Simscape	108
Figure 120 : les références des principaux domaines physiques de Simscape	109
Figure 121 : les variables Across et through de Simscape	110
Figure 122 : source de force orientée positivement	112
Figure 123 : source de force orientée négativement	112
Figure 124 : orientation d'une source de force	113
Figure 125 : visualisation de l'influence de l'orientation de la source de force	113
Figure 126 : exemple de mesure de grandeurs électriques	114
Figure 127 : implantation des capteurs	115

Figure 128 : visualisation de la tension aux bornes de la bobine pour les deux orientations des capteurs	116
Figure 129 : visualisation de l'intensité du courant dans le circuit en fonction des deux orientations des capteurs	116
Figure 130 : fenêtre de paramétrage du solveur	
Figure 131 : photo de l'axe lineaire	120
Figure 132 : les composants necessaires à la modelisation de l'axe inteaire avec Siniscape	121
Figure 133 : visualisation des domaines physiques intervenant dans la modensation de l'axe intear e avec siniscape Figure 134 · modèle Simscape de l'axe linéaire	123
Figure 135 : commandes utiles	
Figure 136 : spécification des variables Simscape à prendre en compte dans le logger	133
Figure 137 : fenêtre du Simscape Result Explorer	134
Figure 138 : visualisation des résultats de la simulation	135
Figure 139 : création d'un sous-système	
Figure 140 : commandes utiles	
Figure 141 : creation d un sous-systeme	120 1
Figure 142. visualisation du contenu du sous-système	130
Figure 144 : naviguer dans les sous-systèmes d'un modèle	
Figure 145 : modèle Simscape de l'axe linéaire avec sous-système « moteur »	
Figure 146 : modèle Simscape de l'axe linéaire sans le capteur de vitesse de rotation	140
Figure 147 : modèle Simscape de l'axe linéaire avec le nom des signaux	141
Figure 148 : modèle Simscape de l'axe linéaire, création du sous-système « Axe »	141
Figure 149 : modèle Simscape de l'axe linéaire avec sous-système « moteur » et « Axe »	
Figure 150 : suppression du port d'un sous-système	142
Figure 151 : modele Simscape termine de l'axe inteaire	142
Figure 152 : Infettre une finage sur un sous-systeme	142
Figure 159 : comparation de la reneure « Add mask teon mage »	
Figure 155 : réponse en position de l'axe linéaire non asservi	
Figure 156 : modèle de l'asservissement en position linéaire de l'axe	144
Figure 157 : composants à ajouter au modèle	145
Figure 158 : paramétrage du bloc sommateur	146
Figure 159 : réponse en position de l'axe linéaire asservi	147
Figure 160 : modèle de l'asservissement en position de l'axe linéaire	
Figure 161 : paramétrage du bloc PID	148
Figure 162 : les composants nécéssaires à la modélisation de la commande d'un vérin hydraulique	149
Figure 164 : visualisation des domaines physiques intervenant dans la modélisation de la commande du vérin hydrauli	aue.151
Figure 165 : modèle Simscape de la commande du vérin hydraulique	
Figure 166 : évolution de la vitesse et de la position de la tige du vérin	
Figure 167 : modèle Simscape de la commande du vérin hydraulique avec routage des signaux	161
Figure 168 : visualisation des résultats de la simulation	163
Figure 169 : modèle Simscape de la commande d'un vérin hydraulique avec source de débit	
Figure 170 : paramétrage d'un bloc limiteur de pression	
Figure 171 : variation de l'ouverture de soupape en fonction de pression	166
Figure 172 : evolution de la vitesse et de la position de la tige du verin	100
Figure 173 : evolution de la vitesse et de la position de la tige du verm	168
Figure 177 : Illustration du fonctionnement du bloc « Controlled PWM Voltage »	
Figure 176 : tension de commande du bloc Controlled PWM Voltage	
Figure 177 : signal PWM en fonction de la tension de commande	170
Figure 179 : paramétrage de l'anglet DWM du bloc Controlled DWM Voltage	
Figure 178 . parametrage de l'ongiet P WM du bloc controlled P WM voltage	171
Figure 178 : paramétrage de l'onglet Input Scaling du bloc Controlled PWM Voltage	171 172
Figure 178 : paramétrage de l'onglet Input Scaling du bloc Controlled PWM Voltage	171 172 172
Figure 178 : paramétrage de l'onglet I nyut Scaling du bloc Controlled PWM Voltage	171 172 172 173
Figure 178 : paramétrage de l'onglet Input Scaling du bloc Controlled PWM Voltage Figure 180 : paramétrage de l'onglet Output Voltage du bloc Controlled PWM Voltage Figure 181 : commande PWM d'un moteur à courant continu Figure 182 : vitesse de rotation du moteur	171 172 172 173 173
Figure 178 : paramétrage de l'onglet Input Scaling du bloc Controlled PWM Voltage Figure 180 : paramétrage de l'onglet Output Voltage du bloc Controlled PWM Voltage Figure 181 : commande PWM d'un moteur à courant continu Figure 182 : vitesse de rotation du moteur après modification de la fréquence PWM Figure 184 : commande PWM d'un moteur à courant continu avec pont en H	171 172 172 173 173 174 175
Figure 178 : paramétrage de l'onglet Input Scaling du bloc Controlled PWM Voltage Figure 180 : paramétrage de l'onglet Input Scaling du bloc Controlled PWM Voltage Figure 181 : commande PWM d'un moteur à courant continu Figure 182 : vitesse de rotation du moteur après modification de la fréquence PWM Figure 183 : vitesse de rotation du moteur après modification de la fréquence PWM Figure 184 : commande PWM d'un moteur à courant continu avec pont en H Figure 185 : vitesse de rotation du moteur commandé par le pont en H	171 172 172 173 173 173 174 175 176
Figure 178 : paramétrage de l'onglet Input Scaling du bloc Controlled PWM Voltage Figure 180 : paramétrage de l'onglet Input Scaling du bloc Controlled PWM Voltage Figure 181 : commande PWM d'un moteur à courant continu Figure 182 : vitesse de rotation du moteur Figure 183 : vitesse de rotation du moteur après modification de la fréquence PWM Figure 184 : commande PWM d'un moteur à courant continu avec pont en H Figure 185 : vitesse de rotation du moteur commandé par le pont en H Figure 186 : tension movenne d'alimentation du moteur	171 172 172 173 173 173 174 175 176 176
Figure 178 : paramétrage de l'onglet I nyut Scaling du bloc Controlled I WM Voltage Figure 179 : paramétrage de l'onglet Input Scaling du bloc Controlled PWM Voltage Figure 180 : paramétrage de l'onglet Output Voltage du bloc Controlled PWM Voltage Figure 181 : commande PWM d'un moteur à courant continu Figure 182 : vitesse de rotation du moteur Figure 183 : vitesse de rotation du moteur après modification de la fréquence PWM Figure 184 : commande PWM d'un moteur à courant continu avec pont en H Figure 185 : vitesse de rotation du moteur commandé par le pont en H Figure 186 : tension moyenne d'alimentation du moteur Figure 187 : paramétrage de l'onglet PWM du bloc Controlled PWM Voltage	171 172 172 173 173 173 174 175 176 176 176 177
Figure 178 : paramétrage de l'onglet l'nym du bloc Controlled l'WM Voltage Figure 179 : paramétrage de l'onglet Input Scaling du bloc Controlled PWM Voltage Figure 180 : paramétrage de l'onglet Output Voltage du bloc Controlled PWM Voltage Figure 181 : commande PWM d'un moteur à courant continu Figure 182 : vitesse de rotation du moteur après modification de la fréquence PWM Figure 183 : vitesse de rotation du moteur après modification de la fréquence PWM Figure 184 : commande PWM d'un moteur à courant continu avec pont en H Figure 185 : vitesse de rotation du moteur commandé par le pont en H Figure 186 : tension moyenne d'alimentation du moteur Figure 187 : paramétrage de l'onglet PWM du bloc Controlled PWM Voltage Figure 188 : paramétrage de l'onglet Input Scaling du bloc Controlled PWM Voltage	171 172 172 173 173 173 174 175 176 176 176 177 178
Figure 178 : paramétrage de l'onglet l'nym du bloc Controlled rwin voltage	171 172 172 173 173 173 174 175 176 176 177 178 178 178
Figure 178 : paramétrage de l'onglet l'nym du bloc Controlled rWM Voltage	171 172 172 173 173 173 174 175 176 176 176 177 178 178 178 178
Figure 178 : paramétrage de l'onglet l'nym du bloc Controlled PWM Voltage	171 172 172 173 173 173 174 175 176 176 176 177 178 178 178 181
Figure 178 : paramétrage de l'onglet l'NM du bloc Controlled PWM Voltage	171 172 172 173 173 173 174 175 176 176 176 176 177 178 178 178 181 182

Figure 194 : pilotage interactif d'un modèle	
Figure 195 : visualisation en temps réel de la position de l'axe linéaire	
Figure 196 : modèle « axe_lineaire_dashboard_start.slx	
Figure 197 : sélection de la fonction Simulation Pacing	
Figure 198 : paramétrage de la fenetre du Simulation Pacing.	
Figure 199 : Insertion des plocs de la bibliotneque Dashboard	
Figure 200 : Tellette de paralitettage du bloc knob	107 197
Figure 201 : anule du bouton knob après parametrage et connexion avec le modele	
Figure 202 : simulation d'un modèle interactif avec les composants de la bibliothèque dashboard	189
Figure 200 : transfert de chaleur par conduction	
Figure 205 : bloc Conductive Heat Transfer de Simscape	
Figure 206 : Paramétrage du bloc Conductive Heat Transfer	
Figure 207 : transfert de chaleur par convection	
Figure 208 : bloc Convective Heat Transfer de Simscape	
Figure 209 : Paramétrage du bloc Convective Heat Transfer	193
Figure 210 : transfert de chaleur par rayonnement	
Figure 211 : bloc Radiative Heat Transfer de Simscape	
Figure 212 : Paramétrage du bloc Radiative Heat Transfer	
Figure 213 : bloc Thermal Mass de Simscape	
Figure 214 : Paramentage du Dioc Therman Mass	
Figure 215 : Tesistance de l'inque Km	
Figure 217 : Paramétrage du bloc Thermal Resistance	
Figure 218 : les éléments de la bibliothèque Simscape Thermal	
Figure 219 : les capteurs de la bibliothèque Simscape Thermal	
Figure 220 : les sources de la bibliothèque Simscape Thermal	
Figure 221 : chauffage d'une barre par conduction	
Figure 222 : ouverture d'un «Blank Model » Simulink	201
Figure 223 : bibliothèque Simscape Thermal	201
Figure 224 : les composants nécessaires à la modélisation du chauffage d'une barre par conduction	
Figure 225 : positionnement des composants dans la fenêtre de travail	
Figure 226 : les commandes de rotation et d'inversion des composants	
Figure 227 : modele Simscape du chauffage d'une barre par conduction	
Figure 220 : les poils d'alistificieurs de puissance (PCP) du domaine thermique	204
Figure 229 : paramétrage du bloc Conductive Heat Transfer	205
Figure 230 : paramétrage du bloc Conductive ricat Transfer	206
Figure 232 : les ports du capteur de température du domaine thermique	
Figure 233 : implantation du capteur de température	
Figure 234 : paramétrage du bloc Ideal Temperature Source	
Figure 235 : les composants associés au capteur de température	209
Figure 236 : implantation du capteur de température	209
Figure 237 : paramétrage du bloc PS – Simulink Converter	210
Figure 238 : les ports du capteur de flux thermique	
Figure 239 : mesure d'une grandeur physique de type flux thermique	
Figure 240 : parametrage du bloc Ideal Heat Flow Sensor	
Figure 241 : les composants associes au capteur de flux thermique.	
Figure 242 : Implantation des capteurs de nux mer mique dans le modèle	
Figure 244 · spécification de la température initiale de la harre	
Figure 245 : configuration des paramètres du solveur	
Figure 246 : température au milieu de la barre en fonction du temps	
Figure 247 : flux thermiques mesurés sur le modèle	
Figure 248 : circulation des flux thermiques dans les différentes parties du modèle	
Figure 249 : vérification de la conservation du flux thermique dans le modèle	
Figure 250 : visualisation de la conservation du flux thermique dans le scope	
Figure 251 : chauffage d'une barre par conduction non isolée	217
Figure 252 : modélisation de l'échange de chaleur par convection	
Figure 253 : paramétrage du bloc Temperature Source	
Figure 254 : parametrage du bloc Convective Heat Transfer	
Figure 255 : temperature au milieu de la parre en fonction du temps avec pertes par convection	
Figure 250 - nux uter inques mesures sur le inouele avec pertes par convection.	220 221
Figure 257 : ajour a un capical pour mesare le nux mermique de convection	
Figure 259 : les échanges thermiques à travers une paroi	
0 ···· 0··· 1··· r··· r···	

Figure 260 : coefficient de transfert thermique par convection	222
Figure 261 : coefficient de transfert thermique par convection	223
Figure 262 : conductivité thermique des matériaux usuels	223
Figure 263 : modélisation d'une couche de la paroi	223
Figure 264 : modélisation Simscape d'une couche de la paroi	224
Figure 265 : paramétrage du bloc Convective Heat Transfer	225
Figure 266 : paramétrage du bloc Conductive Heat Transfer	226
Figure 267 : paramétrage du bloc Thermal Mass	226
Figure 268 : modélisation Simscape d'une couche de la paroi	227
Figure 269 : Paramétrage du bloc Thermal Resistance pour la convection	228
Figure 270 : Paramétrage du bloc Thermal Resistance pour la conduction	
Figure 271 : modélisation d'une couche de paroi en utilisant	
Figure 272 : modélisation d'une paroi d'un bâtiment	230
Figure 273 : caractéristiques physiques des différentes couches de matériau	230
Figure 274 : coefficients de transfert thermique par convection	
Figure 275 : paramètres nécessaires à la construction du modèle de la paroi multicouche	231
Figure 276 : modélisation d'une paroi multicouche, démarrage de la modélisation	231
Figure 277 : paramétrage du bloc Température intérieure	
Figure 278 : paramétrage du bloc Convection Air intérieur/paroi	
Figure 279 : paramétrage du bloc Capteur de flux thermique	
Figure 280 : paramétrage du bloc PS – Simulink Converter	
Figure 281 : parametrage du bloc Conduction laine de verre	
Figure 282 : parametrage du bloc Masse thermique de la laine de verre	234
Figure 283 : parametrage du bloc Conduction beton plein	
Figure 284 : paramétrage du bloc Masse thermique du béton plein	
Figure 285 : paramétrage du bloc Conduction enduit	
Figure 286 : parametrage du bloc Masse thermique de l'enduit	
Figure 287 : parametrage du bloc Convection Air exterieur/paroi	
Figure 288 : parametrage du bloc Temperature exterieure	
Figure 289 : regiage du temps de simulation	
Figure 290 : visualisation du flux thermique en interieur de paroi en regime permanent	
Figure 291 : modification d'un parametre dans le script	238
Figure 292 : visualisation de la nouvelle valeur du flux thermique	
Figure 293 : modelisation equivalente avec des resistance thermiques	
Figure 294 : parametrage du bloc Resistance thermique de convection Air interieur/paroi	
Figure 295 : parametrage du bloc Résistance thermique de conduction du la lame de verie	240
Figure 296 : parametrage du bloc Résistance thermique de conduction du beon pien	240
Figure 257 : parametrage du bloc Résistance thermique de contraction dir enterior (norse)	240 241
Figure 296 : parametrage du bioc Resistance mermique de convection Air exterieur/paror	
Figure 209 : double vitrage avec couche d'argon	
Figure 300 - double viti age avec couche u al gon	241 242
Figure 301 - modele feterut pour la piece metricute et pour la paroi viu ee	
Figure 302 : paramètres pérsecires à la construction du modèle d'un double vitrage	
Figure 303 : parallettes necessaries a la construction du modele d'un double vitrage	
Figure 304 : modelisation d'une paroi multicouche, denna rage de la modelisation	
Figure 305 : sous-systeme « Double vitrage + algon »	
Figure 300 : sous-systeme « Double Vitage »	244 245
Figure 309 : paramétrage du bloc Convoction Air intériour (parai	245 245
Figure 300 : paramétrage du bloc Convection An metricul / parot	243 246
Figure 307 · paramétrage du bloc Convection paron vince/an exterieure	240 216
Figure 310 · paramétrage du bloc PS – Simulink Converter	240 247
Figure 312 : paramétrage du bloc l'onduction de la vitre intérieure	247
Figure 313 · paramétrage du bloc Conduction de la vitre intérieure	247 21Q
Figure 314 · paramétrage du bloc Conduction de la couche d'argon	270 7 <u>4</u> 9
Figure 315 · paramétrage du bloc Masse thermique de l'argon	270 749
Figure 316 · paramétrage du bloc Conduction de la vitre extérieure	247 749
Figure 317 : paramétrage du bloc Masse thermique de la vitre extérieure	250
Figure 318 : paramétrage du bloc Conduction de la vitre intérieure	
Figure 319 : paramétrage du bloc Masse thermique de la vitre intérieure	
Figure 320 : réglage du temps de simulation	252
Figure 321 : évolution de la température dans la pièce en fonction du temps nour les deux types de vitrage	
Figure 322 : modèles Simscape Thermal	
Figure 323 : modèle House Heating System . Mathworks ©	
Figure 324 : modèle thermique de la maison réalisé avec Simscape	
Figure 325 : évolution des températures intérieures et extérieures et du coût du chauffage	

Figure 326 : température des différentes parois de la maison	
Figure 327 : flux de chaleur au travers des différentes parois de la maison	
Figure 328 : évolution de la température sans système de chauffage	256
Figure 329 : présentation du hacheur série	257
Figure 330 : principe de fonctionnement du hacheur série	258
Figure 331 : schéma de principe du hacheur série	258
Figure 332 : schéma de commande d'un moteur à courant par un hacheur série	259
Figure 333 : circulation du courant dans le circuit en phase active et en phase de roue libre	259
Figure 334 : visualisation de l'influence du rapport cyclique sur l'ondulation de courant	
Figure 335 : visualisation de l'influence de la fréquence de hachage sur l'ondulation de courant	
Figure 336 : visualisation de l'influence de la valeur de l'inductance sur l'ondulation de courant	
Figure 337 : analogie entre la forme du modèle Simscape et le schéma électrique de principe	
Figure 338 : modèle Simscape du hacheur série	
Figure 339 : les composants nécessaires à la modélisation du hacheur série	
Figure 340 : paramétrage du bloc Diode	
Figure 341 : paramétrage du bloc MOSFET	
Figure 342 : paramétrage du bloc Pulse Generator	
Figure 343 : évolution de la vitesse de rotation du moteur commandé par un hacheur série	
Figure 344 : création d'un sous-système pour didactiser le modèle	
Figure 345 : mise en place des capteurs de courant dans le circuit	
Figure 346 : ajout d'un capteur sur le modèle	
Figure 347 : modèle instrumenté et didactisé	
Figure 348 : sous-système « capteur de courant »	270
Figure 349 : évolution du courant dans les trois branches du circuit	271
Figure 350 : le modèle non didactisé	273
Figure 351 : le modèle didactisé	273
Figure 352 : amélioration de la didactisation du modèle	274
Figure 353 : le modèle didactisé et optimisé	275
Figure 354 : réglage de la période de commutation du transistor	
Figure 355 : visualisation de la valeur moyenne et instantanée du courant moteur	
Figure 356 : ajout d'une inductance de lissage en série avec le moteur	
Figure 357 : exploitation et visualisation de la circulation du courant dans le hacheur série	
Figure 358 : réglages des paramètres de la simulation	
Figure 359 : exploitation et visualisation de l'influence du rapport cyclique sur le courant moteur	
Figure 360 : réglage des paramètres de la simulation	
Figure 361 : exploitation et visualisation de l'influence de la fréquence de hachage sur le courant moteur	
Figure 362 : paramètrage du bloc Pulse Generator	
Figure 363 : reglage des parametres de la simulation.	
Figure 364 : visualisation et exploitation de l'influence de l'inductance de la charge sur le courant moteur	
Figure 365 : tableau de données	
Figure 366 : trace de deux vecteurs	290
Figure 367 : trace de deux courbes dans la meme fenetre graphique	
Figure 368 : affichage de la grille sur un graphique.	
Figure 369 : ouverture d'une nouvelle fenetre graphique	
Figure 370 : codes de mise en forme des courbes	
Figure 3/1 : mise en forme d'une courbe : modification de la couleur et du style de ligne	
Figure 3/2 : mise en forme d'une courbe : ajout de marqueurs	
Figure 3/3 : mise en forme d'une courbe : choix de l'épaisseur de trait	
Figure 3/4 : mise en forme a une courbe : legende et annotation des axes	
Figure 3/5 : mise en forme a une courbe : modification des échelles des axes	
Figure 376 : indexation des lignes et des colonnes d'une matrice	
Figure 377 : Indexation des elements à une matrice avec un indice unique	
Figure 378 : extraction d une matrice	
Figure 3/9 : extraction d une ligne entiere ou d une colonne entiere	
rigure 500 : trace de la loncuon I(t) Figure 201 : fonêtre Editor d'édition des gavinte	
Figure 202 : avenue de cavint	
rigure 302 : exemple de script	
rigure 505 : ieneire a ajout automatique de dossier au « path » de MATLAB	
Figure 364 : 1 race de plusieurs sinus sur le meme graphique	
Figure 365 : resultats au script de trace de plusieurs sinus	
rigure 500 : les operateurs logiques et de comparaison de MAILAB	
rigure 50/ : repartition d'une serie de points	
rigure 500 : script a interpolation a une serie de aonnees	
rigure 200 : graphique de l'interpolation d'une serie de données	
Figure 370 : script de resolution d'une équation du second degre	
rigure 391 : resolution a une equation au second degre avec racines complexes	

Figure 392 : développer et factoriser une expression	
Figure 393 · script nermettant de dériver une contrion	312
Figure 394 · ser le remettant d'infører une fonction et de calculer une intégrale définie	312
Figure 591 - son prependent a megler la transformée de Laplace de fonctions	313
Figure 396 · soriet permettant d'obtenir la transformée inverse de Laplace d'une fonction	314
Figure 397 · seriet permettant de décomposer en éléments simples une fraction rationnelle	314
Figure 5.98 · seriet permettant la résolution d'une équation différentielle du second ordre	316
Figure 300 - représentation de la solution 1 de l'équation différentielle d'ordre 2	317
Figure 3.00 - représentation de la solution 2 de l'équation différentielle d'ordre 2	318
Figure 400 - representation de la solution 2 de requation differentiene d'ordre 2	
Figure 402 · fonction de transfert en parallèle	
Figure 402 : fonction de transfert en parale formán	220
Figure 404 - fonction de transfert en bouche en mee	
Figure 404 : fonction de transfert d'un système quelconque	
Figure 405 : reponse indiciene a un systeme.	
Figure 406 : reponse impusionnelle d'un systeme	
Figure 407: diagramme de Bode d'un systeme	
Figure 408 : diagramme de Black-Nichols d'un système	
Figure 409 : diagramme de Nyquist d'un système	
Figure 410 : tracé de deux diagrammes de Bode sur le même graphique	
Figure 411 : diagramme de Bode avec indication des marges de gain et de phase	
Figure 412 : ouverture du modèle Simulink du four	
Figure 413 : modélisation de l'asservissement en température d'un four	
Figure 414 : script contenant les paramètres de la simulation	330
Figure 415 : visualisation des variables créées dans le Workspace	330
Figure 416 : réponse en température du four	331
Figure 417 : nommer un signal Simulink	332
Figure 418 : placement des points de linéarisation pour tracer un diagramme de Bode en boucle ouverte	332
Figure 419 : placement d'un point de linéarisation de type « Open loop input »	333
Figure 420 : représentation des points de linéarisation	333
Figure 421 : insertion d'un bloc Bode Plot	334
Figure 422 : fenêtre de paramétrage du bloc Bode Plot	334
Figure 423 : fenêtre de tracé du diagramme de Bode	335
Figure 424 : diagramme de Bode de la boucle ouverte	335
Figure 425: représentation des points de linéarisation	
Figure 426 : insertion d'un second bloc Bode plot	
Figure 427: fenêtre de paramétrage du bloc Bode Plot	
Figure 428 : paramétrage des points de linéarisation pour la boucle fermée	
Figure 429 : diagramme de Bode de la boucle fermée	
Figure 430 : asservissement en température d'un four avec saturation	
Figure 431 : réponse en température de four avec saturation	
Figure 432 : mise en place d'un scope sur un signal	
Figure 433 : visualisation de la tension en sortie de saturateur	
Source 434 : fenêtre de paramétrage de l'exportation des données vers le Workspace	
Figure 435 : script pour tracer une série de courbes	345
Figure 436 · influence de la correction proportionnelle sur la température du four	346
Figure 437 · modèle du four nour la linéarisation	346
Figure 139 : scrint nermettant de linéariser un modèle Simulink	347
Figure 430 - diagramme da Bodo obtanu en linéaricant le modèle Simuliak	347
Figure 430 : chagt annue de bode obtend en meansant le modele similarité da la pade on faicant variar la gain proportionnal du	corroctour
rigure 440. script permettant de tracer une serie de diagrammes de bode en faisant varier re gam proportionner du	247
Figure 441 - diagrammes de Pode obtenus en faisant variar le gain du correctour proportionnel	
Figure 441, diagrammes de bode obtenus en faisant varier le gam du correcteur proportionner	
Figure 442 : bouce de cap sans le utagramme d'état	
Figure 443 : bibliotneque Stattlow	
Figure 444 : positionnement d'un chart dans un modele Simulink	
rigure 445 : commande de creation d'un etat et d'une « transition par defaut »	
rigui e 440 : ouverture de la leneure symbols	
Figure 447 : environnement de travail de Stateflow	
Figure 448 : creation des états du système	
Figure 449 : création d'une transition par défaut	353
Figure 450 : création des transitions entre les états	353
Figure 451 : affectation des actions dans les états	354
Figure 452 : nature des informations d'une étiquette de transition	355
Figure 453 : écriture des étiquettes de transitions dans un diagramme d'états	355
Figure 454 : chart non connecté avec le modèle	356
Figure 455 : fenêtre Symbols avec paramétrage incomplet des entrées/sorties	356
Figure 456 : affectation des variables du diagramme à l'aide du Symbol Wizard	357

Figure 457	: la fenêtre Symbols	
Figure 458	visualisation des variables dans le Model Explorer	358
Figure 459	chart avant connexion avec le système	
Figure 460	inversions de la position des ports	
Figure 461	modification des numéros de port d'un chart	
Figure 462	connexion du chart avec le schéma bloc Simulink	
Figure 463	: résultat de la simulation	
Figure 464	: représentation de super-état et de sous-état	
Figure 465	: représentation de sous-états parallèles	
Figure 466	: creation de super-états	
Figure 467	: creer des sous-états parallèles	
Figure 468	representation des états parallèles dans Stateflow	
Figure 469	: utilisation de variables internes pour evaluer l'activation d'un état	
Figure 470	: parametrage de la variable voyant_rouge	
Figure 471	: chart complet de commande de cap avec super-etats et états parallèles	
Figure 472	: connexion de la variable de sortie du chart à un scope	
Figure 475	: visualisationi de l'état d'activation du voyant	
Figure 474	: ajout u un voyant de la bibliotheque Dashboard dans le modele	
Figure 475	: rapper des principales regies de syntaxe de statenow	
Figure 470	. modèle Multibody de la partie mécanique du phôte nyur aunque	
Figure 477	: fanôtra Machanics Evalorare da la partia mécanique du pilota hydraulique	
Figure 470	: ontions de la barre de visualisation de Multibody	
Figure 480	· paramétrage de la pecanteur dans Multibody	372
Figure 481	· modification de l'action de la necanteur	272
Figure 482	: modèle du nilote hydraulique avec modèle Multibody à intégrer	374
Figure 483	: visualisation du système Multibody sans connexion avec le reste du modèle	374
Figure 484	: visualisation du système Muthody sails connexion avec le reste du modèle	375
Figure 485	: connexions des ports du sous-système avec le modèle	376
Figure 486	: interface de translation entre Simscape et Multibody	
Figure 487	: interfacage de rotation entre Simscape et Multibody	
Figure 488	: les interfaces entre Simscape et Multihody	377
Figure 489	aiout du bloc Simscape Multibody Interface	
Figure 490	; paramétrage des ports d'une liaison	
Figure 491	connexion entre Simscape et Multibody	
Figure 492	ajout d'un port pour imposer un couple à la liaison	
Figure 493	: conversion du signal physique en signal Simulink	
Figure 494	ajout de port sur une liaison dans Multibody	
Figure 495	visualisation de l'offset de l'angle de barre	
Figure 496	compensation de l'offset de l'angle de la barre	
Figure 497	ajout d'un effort sur la liaison, utilisation d'une Lookup Table	
Figure 498	: paramétrage d'une Lookup Table	
Figure 499	: visualisation d'une Lookup Table	
Figure 500	visualisation de la courbe représentative de la loi entrée sortie d'une Lookup Table	
Figure 501	résultat de la simulation du cap suivi par le bateau	
Figure 502	choix de la version de Multibody Link	
Figure 503	: fenêtre d'accueil de Solidworks	
Figure 504	ouverture de la fenêtre des compléments de Solidworks	
Figure 505	: sélection des compléments de Solidworks	
Figure 506	ouverture du modèle Solidworks de la partie mécanique du pilote	
Figure 507	: modèle Multibody obtenu	
Figure 508	: fenêtre Mechanics Explorer	
Figure 509	: la modélisation black box	
Figure 510	: les conditions de l'essai	
Figure 511	: visualisation des données utilisées pour l'identification	
Figure 512	: visualisation dans le Workspace des données utilisées pour l'identification	
Figure 513	: menu derouiant des applications de MATLAB	
Figure 514	: description de la fenetre de la toolbox identification	
Figure 515	: Iniportation de données temporelles	
Figure 516	: selection des données temporenes pour l'identification	
Figure 51/	. visualisation des données importées dans la feneure de la « System Identification » toolbox	
Figure 518	. visualisation des donnees importees dans la « system identification » toolboy.	
Figure 519	, choix de la forme de la fonction de transfart dans la « System Identification » toolboy.	
Figure 520	. choix de la forme de la fonction de d'ansier d'ansier d'ans la « System Identification » toolbox	
Figure 521	. la reneu e i lant fuentification i i ogress ue la « system fuentification » tooloox	200
i igui e 322	. אוסעמווסמנוסוו עב ו בסעונמנס עב ו ועבוונוונמנוסוו	

Figure 523 : fonction de transfert obtenue par identification	398
Figure 524 : superposition des données issues de l'expérimentation et du modèle issu de l'identification	399
Figure 525 : diagramme de Bode de la fonction de transfert tf1	399
Figure 526 : modélisation du moteur à courant continu	402
Figure 527 : asservissement en vitesse d'un moteur à courant continu	403
Figure 528 : modélisation de l'asservissement en vitesse du moteur	403
Figure 529 : réponse temporelle du système non corrigé à un échelon unitaire	404
Figure 530 : ajout d'un bloc PID Controller dans l'asservissement	404
Figure 531 : fenêtre de paramétrage du PID Controller	405
Figure 532 : fenêtre de réglage des performances du système	406
Figure 533 : les critères de performances du système et les valeurs de gain du correcteur	407
Figure 534 : ajout du diagramme de Bode de la fonction de transfert en boucle ouverte	407
Figure 535 : visualisation du diagramme de Bode de la fonction de transfert en boucle ouverte pour le réglage du PID	408
Figure 536 : réglage des options des critères de performances	408
Figure 537 : visualisation des critères de performances	409
Figure 538 : reponse corrigee	410
Figure 539 : actualisation automatique des parametres du PID	411
Figure 540 : reponse du système après reglage du PID	411
Figure 541 : modele du moteur a courant continu asservi en vitesse avec perturbation	412
Figure 542 : reponse temporelle du systeme non corrige	413
Figure 543 : Tancement de l'application « Control System Designer »	414
Figure 544 : Control System Designer	414
Figure 545 : Choix du Dioc a l'èglet	415 115
Figure 540 : leffette Euri Al cificecture	415
Figure 547 : choix des points de linearisation pour la FIDO	410
Figure 548 : diagramme de Bode de la FTBO	410
rigule 549 : utagramme de black de la r I bU	/ 417
Figure 530 : choix de la comiguration des reneures graphiques.	/110
Figure 551 : inounication de la comparation	110 م. 110
Figure 552 : choix des points de inicalisation	114 110
Figure 555 : choix du point d'entrée de la fonction de transier :	ر ۲۲ ۸10
Figure 555 : choix du noint de sortie de la fonction de transfert	
Figure 556 : choix des points d'antrée et de sortie de la FTRE vis-à-vis de la consigne	420
Figure 550 : choix des points d'entrée et de sortie nour la rénonse temnorelle vis-à-vis de la perturbation	421
Figure 558 : choix des noints d'entrée et de sortie de la FTRF vis-à-vis de la perturbation	422
Figure 550 : visualisation des fenêtres utiles nour effectuer le réglage	422
Figure 560 · les fonctions de la fenêtre de réglage du PID	423
Figure 561 : définition des critères de rapidité	
Figure 562 : visualisation du temps de réponse à 5%	
Figure 563 : définition des critères de performances	
Figure 564 : visualisation des critères de performances dans la fenêtre graphique	
Figure 565 : réglage satisfaisant le cahier des charges du PID	
Figure 566 : résultat du réglage manuel du PID sur la réponse temporelle	426
Figure 567 : réponse indicielle du système vis-à-vis d'un échelon de perturbation	427
Figure 568 : influence des réglages effectués sur les diagrammes fréquentiels de la FTBO	428
Figure 569 : réponse temporelle après réglage du PID	429
Figure 570 : modèle de l'asservissement du moteur à continu avec correction par fonction de transfert à régler	430
Figure 571 : réponse du système non corrigé	431
Figure 572 : lancement de l'application « Control System Designer »	432
Figure 573 : Control System Designer	432
Figure 574 : choix du bloc à régler	433
Figure 575 : fenêtre Edit Architecture	433
Figure 576 : choix des points de linéarisation pour la FTBO	434
Figure 577 : diagramme de Bode de la FTBO	434
Figure 578 : diagramme de Black de la FTBO	435
Figure 579 : choix de la configuration des fenêtres graphiques	435
Figure 580 : modification de la configuration des fenêtres graphiques	436
Figure 581 : choix des points de linéarisation	437
Figure 582 : choix du point d'entrée de la fonction de transfert	437
Figure 583 : fenêtre New Step to Plot	438
Figure 584 : choix du point de sortie de la fonction de transfert	438
Figure 585 : définition des points de linéarisation de FTBF_consigne	439
Figure 586 : ajout de la réponse indicielle de la FTBF	439
Figure 587 : tracé du diagramme de Bode de la FTBF	440
Figure 588 : configuration des diagrammes pour effectuer le réglage	440

Figure 589 : la barre de commande du Bode Editor	441
Figure 590 : ajouts d'éléments au correcteur	
Figure 591 : variation du gain de la fonction de transfert en boucle ouverte	
Figure 592 : Edition de la fonction de transfert su correcteur	444
Figure 593 : ajout d'un intégrateur dans le correcteur	445
Figure 594 : visualisation de l'instabilité de la réponse indicielle	445
Figure 595 : visualisation de la fonction de transfert du correcteurt	
Figure 596 : visualisation des pôles et des zéros sur les courbes	
Figure 597 : stabilisation du système par ajout d'un correcteur à avance de phase	
Figure 598 : visualisation de la fonction de transfert du correcteur	
Figure 599 : ajout d'un filtre rejecteur au correcteur	
Figure 600 : Visualisation des parametres de regiage du nitre rejecteur	
Figure 601 : regiage du filtre rejecteur	
Figure 602 : performances de la boucle fermée après reglage du filtre	
Figure 603 : visualisation de la fonction de la disferi du correcteur	434 455
Figure 604 : Innuence du fille sur le comportement de la boucle ouverte	433
Figure 606 : comparaison des rénonses indicielles corrigée et non corrigée	430 456
Figure 607 : organisation des scripts et des fonctions méthode 1	
Figure 608 : organisation des scripts et des fonctions, méthode ?	
Figure 609 · my 0 function	459
Figure 610 : codage de la fonction my 0 function	460
Figure 611 · codage de la fonction my 0 function	460
Figure 612 · visualisation de la fonction my 0 function dans le rénertoire courant	460
Figure 613 · mv 1 function	461
Figure 618 · my_r_rate of my 1 function	461
Figure 615 : codage de la fonction my 1 function	
Figure 616 · visualisation de la fonction w_1 function dans le répertoire courant	462
Figure 617 : my 2 function	
Figure 618 : codage de my 2 function	
Figure 619 : codage de la fonction my 2 function	
Figure 620 : visualisation de la fonction my 2 function dans le répertoire courant	
Figure 621 : script 0.m.	
Figure 622 : résultat obtenu suite à l'exécution de script_0.m	
Figure 623 : ajout d'une légende automatiquement au graphique	
Figure 624 : représentation graphique avec légende automatique	467
Figure 625 : script_0_legend_all_in_one	
Figure 626 : my_plot_function	469
Figure 627 : my_plot_function	470
Figure 628 : my_3_function	470
Figure 629 : my_3_function	471
Figure 630 : appel de la fonction my_plot_function	471
Figure 631 : modification de my_3_function	471
Figure 632 : appel de la fonction my_plot_function	472
Figure 633 : my_4_function	472
Figure 634 : codage de my_4_function	472
Figure 635 : my_5_function	473
Figure 636 : codage de my_5_function	473
Figure 637 : la fonction fzero	
Figure 638 : calcul de la dérivée numérique au point x ₀	
Figure 639 : différence finie progressive	
Figure 640 : formule de difference finie retrograde	
rigure 041 : iormule de difference finie centree	
rigure 642 : conage de la fonction 11 qui contient l'expression de la fonction à deriver	
rigure 644 - codogo do la fonction fdority forward (a b nb noint f)	
Figure 644 : couage de la loncuon lueriv_lorward (a,b,nb_point,I)	
Figure 045 : 10110101110011V_DdCKWd10 (d,D,110_P011101)	
Figure 647 : fonction fdorig contor (a h nh point f)	481 401
Figure 649 : codage de la fonction fderiv, conter (a b nh neint f)	48144
Figure 649 fonction fderiv and symbolicule (a h nh noint f)	402. 107
Figure $650 \cdot codage de la fonction fderiv and symbolique (a h nh noint f)$	402. 102
Figure 651 · comparaison des différentes méthodes de dérivation numérique	۲03- ۸.۵۲
Figure 652 : comparaison des différentes méthodes de dérivation numérique	
Figure 653 : zoom sur la courbe	486
Figure 654 : visualisation des erreurs induites par les différentes méthodes de dérivation numérique	486

Figure 655 : script permettant de visualiser l'influence du pas	
Figure 656 : influence du pas sur l'allure des courbes	
Figure 657 : script permettant de connaître l'ordre de variation des fonctions erreurs	
Figure 658 : variation de l'erreur de dérivation en fonction de h pour la différence finie centrée	492
Figure 659 : variation de l'erreur de dérivation en fonction de h pour la différence finie progressive et rétrograde	
Figure 660 : modification de la fonction f1.m	
Figure 661 : comparaison des différentes méthodes de dérivation numérique	
Figure 662 : visualisation des erreurs induites par les différentes méthodes de dérivation numérique	
Figure 663 : influence du pas sur l'allure des courbes	
Figure 664 : variation de l'erreur de dérivation en fonction de h pour la différence finie centrée	
Figure 665 : variation de l'erreur de dérivation en fonction de h pour la différence finie progressive et rétrograde	
Figure 666 : mesure de la position à partir du signal d'un codeur incrémental	
Figure $667 : zoom sur la courbe$	
Figure 668 : influence du pas d'échantillonnage sur le calcul de la dérivée numérique	
Figure 669 : derivation numerique du signal brut	
Figure 670 : initiance de l'augmentation du pas de derivation	
Figure 671 : Ionculon Ideriv_num_pas_variable(t,Y,step_size)	
Figure 672 : couldge de la foircuoir ideniv_num_pas_variable(t, r, step_size)	500 E01
Figure 675 : Script permetant de faire varier le pas de dérivation	
Figure 675 : illustration de la recharche des solutions de l'équation $f(x)=0$ nar la méthode de dichotomie	502
Figure 675 · function fdichotomie (f a h ensilon)	504
Figure 677 · codage de la fonction fdichotomie(f a h ensilon)	504
Figure 678: fonction f2 m	505
Figure 679 · recherche de solution nar dichotomie	506
Figure 680 : courbe représentative de la fonction f(x)	506
Figure 681 : exécution du script dichotomie m pour la recherche d'une racine dans l'intervalle[-4 : 2]	
Figure 682 : exécution du script dichotomie.m pour la recherche d'une racine dans l'intervalle[0 : 2]	
Figure 683 : illustration de la méthode de Newton	
Figure 684 : fonction fNewton(f,x ₀ ,epsilon)	508
Figure 685 : codage de la fonction fNewton(f,x ₀ ,epsilon)	509
Figure 686: fonction f2.m.	509
Figure 687 : recherche de solution par la méthode de Newton	510
Figure 688 : courbe représentative de la fonction f(x)	510
Figure 689 : exécution du script Newton.m pour rechercher la solution autour de x=3	510
Figure 690 : exécution du script Newton.m pour rechercher la solution autour de x=-2	511
Figure 691 : calcul de l'intégrale d'une fonction sur un intervalle [a ; b]	511
Figure 692 : discrétisation du domaine d'intégration	512
Figure 693 : méthode des rectangles	513
Figure 694 : méthode des trapèzes	
Figure 695 : codage de la fonction f3.m qui contient l'expression de la fonction à intégrer	
Figure 696 : fonction fintegration_rec (a,b,nb_point,f)	
Figure 69/ : codage de la fonction fintegration_rec (a,b,nb_point,f)	514 515
Figure 698 : fonction fintegration_trap (a,b,no_point,f)	
Figure 699 : codage de la fonction fintegration_trap (a,b,nb_point,f)	515 F1C
Figure 700 : Ionculon Innegration_ana_symbolique (a,b,ino_point,i)	
Figure 701 : couldge de la fonction fintegration_and_symbolique (a,D,IID_point,I)	510 E10
Figure 702 : comparaison des différentes méthodes de denvation numérique	510 510
Figure 703 : comparaison des unierentes methodes à integration numerique	
Figure 705 · visualisation des erreurs induites par les différentes méthodes d'intégration numérique	520
Figure 706 · scrint nermettant de visualiser l'influence du nas	522
Figure 707 · influence du pas sur l'allure des courbes	522
Figure 708 : script permettant de connaître l'ordre de variation des fonctions erreurs	525
Figure 709 : variation de l'erreur d'intégration en fonction de h pour la méthode des rectangles	
Figure 710 : variation de l'erreur de d'intégration en fonction de h pour la méthode des trabèzes	
Figure 711 : modification de la fonction f1.m	
Figure 712 : comparaison des différentes méthodes d'intégration numérique	
Figure 713 : visualisation des erreurs induites par les différentes méthodes d'intégration numérique	
Figure 714 : influence du pas sur l'allure des courbes	
Figure 715 : variation de l'erreur d'intégration en fonction de h pour la méthode des rectangles	
Figure 716 : variation de l'erreur d'intégration en fonction de h pour la méthode des trapèzes	529
Figure 717 : relevé de l'accélération en fonction du temps	529
Figure 718 : intégration numérique de l'accélération	531
Figure 719 : loi en trapèze de vitesse	532
Figure 720 : exemple de signal bruité	533

Figure 721 : filtre numérique	533
Figure 722 : mesure de la position à partir du signal d'un codeur incrémental	534
Figure 723 : zoom sur la courbe	534
Figure 724 : fonction filtre_mgl(Y,n)	535
Figure 725 : codage de la fonction fliltre_mgl(Y,n)	536
Figure 726 : script permettant de voir l'influence de n pour un filtre à moyenne glissante	537
Figure 727 : filtrage par moyenne glissante d'un signal	538
Figure 728 : zoom sur la courbe	538
Figure 729 : filtre passe-bas du premier ordre dans le domaine de Laplace	538
Figure 730 : filtre passe-bas du premier ordre dans le domaine temporel	539
Figure 731 : fonction filtre_pb_ordre_1(t,Y,tau)	539
Figure 732 : codage de la fonction filtre_pb_ordre_1(t,Y,tau)	
Figure 733 : script permettant de voir l'influence de tau pour un filtre passe-bas du premier ordre	541
Figure 734 : filtrage d'un signal par un filtre passe-bas du premier ordre avec différentes valeurs de τ	
Figure 735 : processus de traitement du signal	
Figure 736 : derivation numerique en utilisant Simulink	
Figure 737 : signal brut a deriver	
Figure 738 : derivation d un signal numerique discontinu sans filtrage	
Figure 739 : filtrage realise avec une fonction de transfert	
Figure 740 : parametrage du bloc Transfer Fcn	
Figure 741 : derivation numerique a un signal filtre en utilisant Simulink	
Figure 742 : Intrage realise avec le bloc Analog Filter Design	540 E46
Figure 743 : parametrian numérique d'un signal filtré en utilizant Simulink	
Figure 744 : dei Vation numerique à un signa intre en autisant Simunik	
Figure 745 : Intrage realise avec le Dioc Moving Average	
Figure 740 : parametrian numérique d'un signal filtré en utilizant Simulink	
Figure 749 : illustration de la méthode d'Euler de récolution numérique d'une équation différentielle	551
Figure 740 : industration de la finetioue d'Euler de resolution numerique d'une equation différenteme	552
Figure 750 · fonction fEuler (f a h CL nh noints)	552
Figure 750 : rolection Euler (i, a, b, ci, hb_points).	553
Figure 752 : résolution d'une équation différentielle du premier ordre	554
Figure 752 · resolution numérique calculée avec 10 noints	554
Figure 755 · solution numerique calculée avec 10 points	555
Figure 755 : fonction my ode (f. a. h. Cl. nh points)	558
Figure 756 : structure de la matrice sol	
Figure 757 : codage de la fonction my ode (f.a.b.Cl.nb points)	
Figure 758 : codage de la fonction F 2.m qui contient l'expression de la fonction f(t.v)	
Figure 759 : comparaison des différentes méthodes de résolution d'une équation différentielle	
Figure 760 : comparaison des différentes méthodes de résolution d'une équation différentielle (100 points calculés)	
Figure 761 : comparaison des différentes méthodes de résolution d'une équation différentielle (1000 points calculés)	
Figure 762 : zoom sur la courbe	563
Figure 763 : comparaison des différentes méthodes de résolution d'une équation différentielle(10000 points calculés).	564
Figure 764 : zoom sur la courbe	564
Figure 765 : codage de la fonction F_3.m	566
Figure 766 : résolution d'une équation différentielle d'ordre 3 avec my_ode et ode23	567
Figure 767 : solution d'une équation différentielle du troisième ordre	567
Figure 768 : résolution de l'équation différentielle de Van der Pol	568
Figure 769 : solution de l'équation différentielle de Van der Pol, 10000 points calculés	569
Figure 770 : oscillateur mécanique avec prise en compte de la pesanteur	570
Figure 771 : codage de la fonction F_mass_spring_damper.m	571
Figure 772 : Calcul et tracé de la solution de l'équation différentielle en utilisant 3 méthodes de résolution	573
Figure 7/3 : résolution de l'équation différentielle avec 3 méthodes	
Figure 7/4 : modele permettant la resolution de l'equation differentielle avec Simulnk	
Figure $7/5$: specification de la condition initiales x(U)=0.1 dans l'integrateur de la position	5/4
rigure //o : visualisation de la solution x(t) determinée avec simulink	
Figure 777 : modele mulu_physique du systeme masse-ressort-amortisseur	
Figure 770 visualization de la colution mitidle dans Simscape.	5/0 E74
Figure 777 · visualisationi de la solutioni x(t) calculee avec Simscape	
Figure 700. recherche au numero de lighe au plus grand pivot	
Figure 782 · codage de la fonction num ligne nivot(A num col)	
Figure 783 · fonction echange lignes (A ligne i ligne i)	502
Figure 784 · codage de la fonction echange ligne (A ligne i ligne b)	505 ςq1
Figure 785 · fonction transvection(A num ligne nivot)	525 525
Figure 786 · codage de la fonction transvection(A num ligne nivot)	585
0	

88	586
Figure 788 : codage de la fonction triangularisation(A)	587
Figure 789 : fonction résolution.m	587
Figure 790 : codage de la fonction resolution(M)	589
Figure 791 : fonction Pivot_de_Gauss(A,B)	
Figure 792 : hiérarchisation des fonctions utilisées dans l'algorithme du pivot de Gauss	
Figure 793 : codage de la fonction Pivot_de_Gauss(A,B)	
Figure 794 : MATLAB/SIMULINK ONLINE ET MATLAB Drive	
Figure 795 : Dossier MAILAB Drive dans la version installee de MAILAB/Simulink	
Figure 790 : Visualisation de MATLAB Drive Connector	
Figure 797 : Interface de MATLAB DITVE Connector	
Figure 790 : script à saisir online	
Figure 800 · création d'un Blank Model dans Simulink online	598
Figure 801 : librairie de Simulink online	598
Figure 802 : modèle créer à l'aide de Simulink online	
Figure 803 : résultat de la simulation	
Figure 804 : ajout du dossier Maxpid dans le Path de MATLAB online	600
Figure 805 : modèle du robot Maxpid ouvert dans MATLAB/Simulink online	600
Figure 806 : résultats de la simulation	
Figure 807 : visualisation des résultats dans le scope	601
Figure 808 : partage d'un dossier du Drive de MATLAB	602
Figure 809 : définition des membres et des droits d'accès au dossier	603
Figure 810 : gestion des droits d'accès et des membres	603
Figure 811 : création d'un lien pour partager un dossier de MATLAB Drive	604
Figure 812 : partage d'un dossier en utilisant MATLAB on line	604
Figure 813 : ouverture de l'interface MATLAB Drive Connector	605
Figure 814 : lancement de MATLAB Drive online	605
Figure 815 : partage d'un dossier à partir de MATLAB Drive online	606
Figure 816 : MATLAB Mobile	607
Figure 817 : ouverture de MATLAB Mobile	608
Figure 818 : utilisation des lignes de commandes dans MATLAB Mobile	
Figure 819 : configuration MATLAB Keyboard	
Figure 820 : acces a MATLAB Drive depuis MATLAB Mobile	
Figure 821 : execution a un script depuis MATLAB Mobile	
PROVIDE X77 · CONTROLL'ATION EL DEMATTARE O TINE ACONTECTION DES MESTILES (SELES DES CANTEURS OU SUBTITIONNE	611
Figure 022 : comparation du fichier de menures dans MATI AD Drive	611
Figure 822 : sauvegarde du fichier de mesures dans MATLAB Drive	611 611
Figure 823 : sauvegarde du fichier de mesure dans MATLAB Drive	611 611
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive Figure 824 : visualisation du fichier de mesure sur le PC Figure 825 : visualistion du fichier de mesure dans le Workspace Figure 826 : structure de la variable Acceleration Figure 827 : extraction de la colonne des instants	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive Figure 824 : visualisation du fichier de mesure sur le PC Figure 825 : visualistion du fichier de mesure dans le Workspace Figure 826 : structure de la variable Acceleration Figure 827 : extraction de la colonne des instants Figure 828 : extraction de la colonne des instants correspondant aux secondes	611 611 612 613 613 614 614
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive	
Figure 823 : sauvegarde du fichier de mesure adaus MATLAB Drive	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive	
Figure 822 : compared du fichier de mesures dans MATLAB Drive. Figure 824 : visualisation du fichier de mesure sur le PC. Figure 825 : visualistion du fichier de mesure dans le Workspace Figure 826 : structure de la variable Acceleration Figure 827 : extraction de la colonne des instants correspondant aux secondes Figure 828 : extraction de la colonne des instants correspondant aux secondes Figure 829 : extraction de la colonne correspondant aux accélérations mesurées selon l'axe X Figure 830 : script permettant de faire le tracer les accélérations en fonction du temps. Figure 831 : accélérations selon les trois axes en fonction du temps Figure 832 : scopes mesurant le courant dans le circuit et la tension aux bornes de la bobine Figure 833 : mesure de la tension aux bornes de la bobine Figure 834 : barre de commande du scope Figure 835 : mise en forme des courbes dans un scope. Figure 836 : modification de la mise en forme d'une courbe dans un scope Figure 837 : modification de la mise en forme d'une courbe dans un scope Figure 838 : raccordement d'un scope à deux entrées Figure 839 : visualisation des deux courbes dans le même scope Figure 839 : visualisation des caractéristiques de la deuxième courbe Figure 840 : modification des caractéristiques de la deuxième courbe	
Figure 824 : visualisation du fichier de mesures dans MATLAB Drive	
Figure 822 : souvegarde du fichier de mesure acquisition insure sites des depteurs du sinar priorie	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive. Figure 824 : visualisation du fichier de mesures dans MATLAB Drive. Figure 825 : visualistion du fichier de mesure sur le PC. Figure 826 : structure de la variable Acceleration . Figure 827 : extraction de la colonne des instants . Figure 829 : extraction de la colonne des instants correspondant aux secondes	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive. Figure 824 : visualisation du fichier de mesures dans MATLAB Drive. Figure 825 : visualisation du fichier de mesure sur le PC. Figure 826 : structure de la variable Acceleration . Figure 827 : extraction de la colonne des instants . Figure 828 : extraction de la colonne des instants . Figure 829 : extraction de la colonne des instants correspondant aux secondes	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive Figure 824 : visualisation du fichier de mesure dans le Workspace Figure 825 : visualistion du fichier de mesure dans le Workspace Figure 826 : structure de la variable Acceleration Figure 827 : extraction de la colonne des instants Figure 828 : extraction de la colonne des instants correspondant aux secondes Figure 829 : extraction de la colonne des instants correspondant aux secondes Figure 829 : extraction de la colonne correspondant aux accélérations mesurées selon l'axe X Figure 830 : script permettant de faire le tracer les accélérations en fonction du temps Figure 831 : accélérations selon les trois axes en fonction du temps Figure 832 : scopes mesurant le courant dans le circuit et la tension aux bornes de la bobine Figure 833 : mesure de la tension aux bornes de la bobine Figure 834 : barre de commande du scope Figure 835 : mise en forme des courbes dans un scope Figure 836 : modification de la mise en forme d'une courbe dans un scope Figure 837 : modification du nombre d'entrées d'un scope Figure 838 : raccordement d'un scope à deux entrées Figure 839 : visualisation des deux courbes dans le même scope Figure 840 : modification des caractéristiques de la deuxième courbe Figure 841 : modification des caractéristiques de la deuxième courbe Figure 842 : modification du nombre de fenêtre dans le scope Figure 843 : affichage des courbes dans un scope Figure 844 : utilisation des curseurs dans un scope sur plusieurs fenêtres Figure 845 : fenêtre de paramétrage du Signal builder Figure 845 : fenêtre de paramétrage du Signal builder Figure 845 : fenêtre de paramétrage du Signal builder Figure 847 : choix de la durée du signal dans le « signal builder »	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive. Figure 824 : visualisation du fichier de mesures dans MATLAB Drive. Figure 825 : visualisation du fichier de mesure sur le PC. Figure 825 : visualistion du fichier de mesure dans le Workspace. Figure 826 : structure de la variable Acceleration Figure 827 : extraction de la colonne des instants correspondant aux secondes. Figure 828 : extraction de la colonne des instants correspondant aux secondes. Figure 829 : extraction de la colonne correspondant aux accélérations mesurées selon l'axe X. Figure 830 : script permettant de faire le tracer les accélérations en fonction du temps. Figure 831 : accélérations selon les trois axes en fonction du temps. Figure 832 : scopes mesurant le courant dans le circuit et la tension aux bornes de la bobine. Figure 833 : mesure de la tension aux bornes de la bobine. Figure 834 : barre de commande du scope. Figure 835 : mise en forme des courbes dans un scope. Figure 836 : modification de la mise en forme d'une courbe dans un scope. Figure 837 : modification du nombre d'entrées d'un scope . Figure 839 : visualisation des deux courbes dans le même scope. Figure 839 : visualisation des caractéristiques de la deuxième courbe Figure 840 : modification des caractéristiques de la deuxième courbe. Figure 841 : modification des caractéristiques de la deuxième courbe. Figure 842 : modification du nombre de fenêtre dans le scope Figure 843 : affichage des courbes dans un scope sur plusieurs fenêtres. Figure 844 : utilisation des curset sdans un scope. Figure 844 : utilisation des curset sdans un scope. Figure 845 : fenêtre de paramétrage du Signal builder Figure 845 : fenêtre de paramétrage du Signal builder Figure 845 : choirte de paramétrage du Signal builder Figure 846 : barre de commande du signal aluider Figure 848 : modification de l'échelle de temps Figure 848 : modification de l'échelle de temps Figure 848 : modification de l'échelle de temps Figure 848 : modification de l'échelle de	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive. Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive. Figure 825 : visualisation du fichier de mesure sur le PC Figure 826 : structure de la variable Acceleration Figure 827 : extraction de la colonne des instants. Figure 828 : extraction de la colonne des instants correspondant aux secondes. Figure 829 : extraction de la colonne correspondant aux accélérations mesurées selon l'axe X Figure 830 : script permettant de faire le tracer les accélérations en fonction du temps. Figure 831 : accélérations selon les trois axes en fonction du temps. Figure 832 : scopes mesurant le courant dans le circuit et la tension aux bornes de la bobine Figure 833 : mesure de la tension aux bornes de la bobine. Figure 833 : mesure de la tension aux bornes de la bobine. Figure 835 : mise en forme des courbes dans un scope. Figure 835 : mise en forme des courbes dans un scope. Figure 836 : modification du nombre d'entrées d'un scope Figure 837 : modification du la mise en forme d'une courbe dans un scope. Figure 838 : raccordement d'un scope à deux entrées. Figure 839 : visualisation des deux courbes dans le même scope. Figure 840 : modification des caractéristiques de la deuxième courbe. Figure 841 : modification du nombre d'entrées d'un scope Figure 842 : modification du nombre de le ale scope Figure 842 : modification du nombre de le deuxième courbe. Figure 843 : affichage des courbes dans un scope Figure 844 : utilisation des caractéristiques de la deuxième courbe. Figure 842 : modification du nombre de le nêtre dans le scope Figure 843 : affichage des courbes dans un scope Figure 844 : utilisation des caractéristiques de la deuxième courbe. Figure 845 : fenêtre de paramétrage du Signal builder Figure 846 : barre de commande du signal builder Figure 847 : choix de la durée du signal builder Figure 848 : modification de léchelle de temps. Figure 848 : modification de léchelle de tem	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive	
Figure 823 : sauvegarde du fichier de mesures dans MATLAB Drive	

Figure 853 :	génération de plusieurs signaux	629
Figure 854 :	visualisation des sorties multiples du « signal builder »	629

Modélisation et Simulation des Systèmes Multi-Physiques avec MATLAB – Simulink (R2020a) pour l'étudiant et l'ingénieur Quatrième édition











Ivan LIEBGOTT