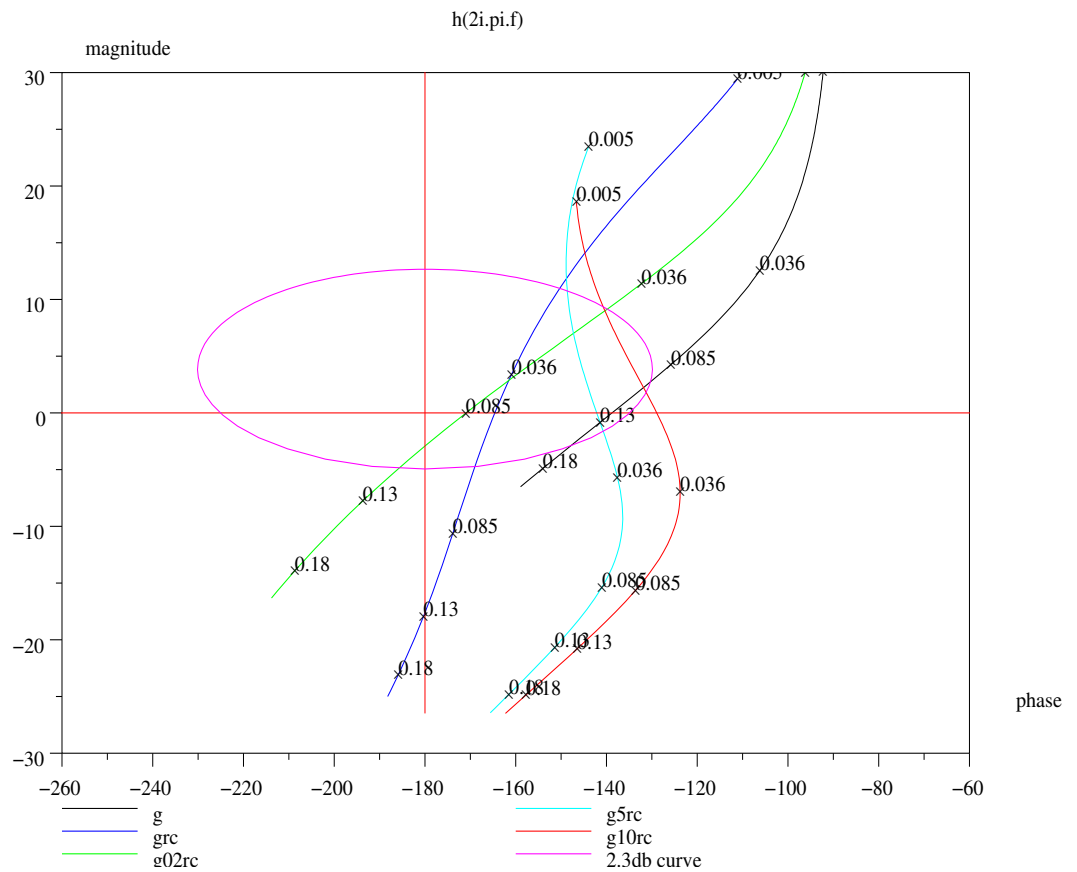


Cours d'automatique illustré avec le logiciel SCILAB

LUCIEN POVY
lucien.povy@free.fr

2017 Version 2



Résumé : Regard sur quelques logiciels de simulation en automatique analogique, exercices d'algèbre et d'automatique avec le logiciel Scilab.

Mots clés : Automatique. Systèmes linéaires. Systèmes à retard. Logiciels de simulation. Scilab.

Remarques : Ce texte a été écrit avec l'éditeur de documents LyX, logiciel libre et gratuit. Ce document et les programmes associés sont protégés par les licences GPL et LGPL, voir la conclusion en fin de rapport. Voici les logiciels qui ont été utilisés pour la simulation et la rédaction de ce document (sous Linux) :

Scilab depuis la version 2.4 jusqu'aux versions 5.5.x, puis la version 6.

Latex2 ϵ avec Lyx (dernière version : 2.2.3).

Xfig versions 3.2.4, 3.2.5 (logiciel de dessin vectoriel). Inkscape version 0.48.

Gv depuis la version 3.5.8 jusqu'à la version 3.6.1 (visualisateur postscript et pdf), puis Okular ou Acrobat Reader.

Table des matières

1	Les logiciels de simulation et l'automatique	11
1.1	Matlab	11
1.1.1	Avantages	11
1.1.2	Inconvénients	11
1.2	Octave	11
1.2.1	Avantages	12
1.2.2	Inconvénients	12
1.3	Scilab	12
1.3.1	Avantages	12
1.3.2	Inconvénients	12
1.4	Mise en place de Scilab sur un PC-LINUX	13
1.4.1	Installation à partir d'un binaire	13
1.4.2	Compilation du logiciel	13
1.4.3	Quelles bogues connus, y remédier, quelques améliorations	13
1.4.4	Notes sur les logiciels LYX et XFIG et INKSCAPE	14
2	Exercices d'algèbre avec Scilab	15
2.1	Démarrage de Scilab	15
2.2	Les différentes manières d'exécuter un programme Scilab	15
2.3	Scalars, Vecteurs, Matrices	18
2.3.1	Vecteurs	19
2.3.2	Matrices	21
2.4	Définir un polynôme par ses racines, ses coefficients. Valeur numérique d'un polynôme	25
2.4.1	Polynôme, racines d'un polynôme	25
2.4.2	Autres paramètres caractérisant un polynôme	28
2.4.3	Vecteur de polynômes, matrices de polynômes	29
2.4.4	Valeur numérique d'un polynôme, changement d'argument	30
2.4.5	Utilisation de l'instruction freq	31
2.4.6	Définition d'un polynôme par ses coefficients	32
2.4.7	Instructions relatives aux polynômes	34
2.4.8	Quelques autres fonctions	37
2.5	La programmation avec Scilab	40
2.5.1	Les fonctions, les macros	40
2.5.2	La programmation	41
2.6	Définir une fraction rationnelle : quelques propriétés	43
2.6.1	Les fractions rationnelles	44
2.6.2	Matrices, vecteurs de fractions rationnelles vus comme des listes	45

Table des matières

2.6.3	Quelques fonctions utiles pour l'étude des fractions rationnelles .	47
3	Définir un système linéaire : enfin un peu d'automatique	52
3.1	Fonction de transfert, matrice de transfert	52
3.1.1	Rappel de cours, définition de la fonction de transfert	52
3.1.2	Diverses représentations	53
3.1.3	Mise sous forme éléments simples, réponse temporelle	54
3.1.4	Représentations d'Evans	56
3.1.5	Forme factorisée de Bode ou forme standard	58
3.1.6	La définition avec Scilab d'une fonction de transfert	59
3.1.7	Définition d'un système linéaire par ses variables d'état	61
3.1.8	Définition d'un système linéaire modèle Z,P,K	61
3.1.9	Passage d'une représentation à une autre représentation	61
3.1.10	Définition d'un système bouclé avec Scilab	61
3.2	Les graphiques dans Scilab, réponses temporelles d'un système	63
3.2.1	Les graphiques en deux dimensions avec Scilab	63
3.2.2	Visualisation des pôles et zéros d'un système	70
3.2.3	Simulation temporelle : réponses impulsionnelle, indicielle, à tout type de signal	71
3.2.4	Introduction des conditions initiales, utilisation de la bibliothèque ode	76
4	Représentation fréquentielle	83
4.1	Rappels sur la réponse fréquentielle : calcul de la réponse d'un système continu	83
4.2	Les divers lieux	86
4.2.1	Représentation de Nyquist	86
4.2.2	Représentation de Bode	87
4.2.3	Représentation de Black	87
4.2.4	L'abaque de Black	88
4.2.5	Quelques caractéristiques d'un système à partir des lieux	90
4.2.6	Rappel de cours, diagrammes asymptotiques de Bode : systèmes à déphasage minimaux et non minimaux	92
4.3	Etude de systèmes simples	93
4.3.1	Le premier ordre	94
4.3.2	Le second ordre	95
5	Etude de la stabilité d'un système, marges de stabilité	101
5.1	Etude de la stabilité	101
5.1.1	Rappels sur la stabilité des systèmes	101
5.1.2	Calcul des pôles d'un système	101
5.1.3	Critère de Routh-Hurwitz	101
5.1.4	La stabilité d'un système bouclé par le critère de Nyquist-Cauchy	107
5.2	Les marges de stabilité d'un système bouclé (robustesse)	113
5.2.1	Marge de stabilité absolue	113

Table des matières

5.2.2	Marge de phase, marge de gain	115
5.2.3	Marge de gain-phase (marge de module ou marge de stabilité)	115
5.2.4	Marge de retard	119
5.2.5	Valeurs usuelles des différentes marges	120
5.2.6	Cercles à gain constant, cercles à phase constante dans le plan de Nyquist : M et N cercles, abaque de Hall	120
5.2.7	Nouveauté : utilisation des pseudo-lieux, pôles dominants sur une droite à amortissement constant	120
6	Principe de la commande	125
6.1	Introduction	125
6.2	Précision	127
6.2.1	Précision statique	127
6.2.2	Précision dynamique	127
6.3	Cahier des charges	129
6.4	Méthodes de synthèse : pourquoi utiliser la méthode de Black	129
6.4.1	Principe de mise en oeuvre	131
7	La correction des systèmes par la méthode de Black	134
7.1	Choix de la structure de réseau correcteur	134
7.1.1	Correcteur dans la chaîne d'action	134
7.1.2	correcteur dans la chaîne de retour	134
7.1.3	Assemblage de correcteurs	135
7.2	Action proportionnelle	136
7.3	Action proportionnelle et dérivée	136
7.4	Réseau correcteur à avance de phase	140
7.4.1	Etude théorique du réseau avance de phase (ou retard)	140
7.4.2	Exemple de synthèse	141
7.5	Réseau correcteur à action proportionnelle et intégrale	144
7.6	Réseau correcteur à retard de phase	147
7.7	Réglage d'un P.I.D. par la méthode du pivot	149
7.8	Réglage d'un réseau retard-avance de phase	157
7.9	Correcteur à action proportionnelle et/ou intégrale et boucle interne dérivée	161
7.9.1	Correcteur à action intégrale dans la chaîne d'action	162
7.9.2	Correcteur proportionnel et intégral dans la chaîne d'action	162
8	Méthodes algébriques et empiriques de synthèse	163
8.1	Les critères mesurant l'erreur dynamique	163
8.2	Méthode de Ziegler et Nichols	164
8.3	Méthode de Chien-Hrones-Reswick	165
8.4	Méthode de Graham et Lathrop	166
8.5	Méthode de Hall et Sartorius	168
8.6	Méthode des polynômes de Naslin	168
8.6.1	Principe	168

Table des matières

8.6.2	Calcul des coefficients	170
8.6.3	Les réponses indicielles	170
8.6.4	Influence d'un numérateur du premier ou second degré	171
9	Représentation d'état des systèmes	173
9.1	Définition de l'état d'un système	173
9.1.1	Modèles de systèmes continus	173
9.1.2	Cas d'un système monovariable (SISO)	174
9.1.3	Cas d'un système multivariable (MIMO)	176
9.2	Intégration des équations d'état	176
9.2.1	Intégration, cas général	176
9.2.2	Intégration sur un intervalle de temps donné à commande constante sur cet intervalle : bloqueur d'ordre zéro, BOZ	177
9.3	Stabilité d'un système continu	178
9.4	Représentations	178
9.4.1	Représentation d'état à partir de la transmittance : cas SISO, forme commandable	178
9.4.2	Passage d'une forme quelconque à la forme commandable	179
9.4.3	Représentation d'état à partir de la transmittance : cas SISO, forme observable	183
9.4.4	Passage d'une forme quelconque à la forme observable	184
9.4.5	Dualité : observabilité, commandabilité	185
9.5	Forme modale (diagonale)	186
9.5.1	Vecteur d'état de la forme modale	186
9.5.2	passage des équations d'état à la matrice de transfert	188
10	Utilisation du logiciel pour simuler les systèmes à retard pur	191
10.1	Principales fonctions utiles	191
10.1.1	Définition d'un système à retard pur	191
10.1.2	Quelques fonctions complémentaires	191
10.1.3	Exemple de programme Scilab pour simuler une réponse tempo- relle à commande retardée : cas SISO	194
10.1.4	Exemple de programme Scilab pour simuler une réponse tempo- relle à sortie retardée	196
10.1.5	Réponse fréquentielle	198
10.1.6	Les systèmes à retard pur : les approximations	202
10.2	La stabilité, les marges de stabilité des systèmes à retard	206
10.2.1	La stabilité	206
10.2.2	Etude de la stabilité des systèmes bouclés à retard [4]	206
10.2.3	Résultats théoriques : méthode de Hermite-Biehler	206
10.2.4	Méthode de Walton et Marshall	207
10.2.5	Les marges de stabilité	211
10.2.6	Lieu des racines adapté aux systèmes à retard pur	212
10.2.7	Les systèmes à retard et les pseudo-lieux	216

Table des matières

10.3	Les systèmes échantillonnés, passage du continu au discret	217
10.3.1	Comment discrétiser un système continu : les méthodes	218
10.3.2	Les systèmes continus sans retard, passage au discret avec blo- queur d'ordre zéro	219
10.3.3	Expression de la transformée en z, par la méthode des résidus	221
10.3.4	Les systèmes échantillonnés, passage du discret au continu et vice versa avec approximations	221
10.3.5	Les systèmes continus avec retard, passage au discret avec blo- queur d'ordre zéro	222
10.3.6	Expression de la transformée en z modifiée par la méthode des résidus	223
10.3.7	Algorithme de discrétisation d'un système à retard pur : pro- gramme dscr_sisord.sci	224
10.3.8	Approximation du retard : algorithme de Thiran	228
10.3.9	Réponses fréquentielles d'un système continu et de son équi- valent échantillonné avec Scilab : programme dpf_cd.sci	229
10.4	Conclusion provisoire	233
11	Introduction à l'étude des systèmes non entiers	235
11.1	Rappel de mathématiques	235
11.1.1	Quelques propriétés de la dérivation non entière	236
11.1.2	Transformée de Laplace	236
11.1.3	Equations différentielles généralisées	237
11.1.4	Solutions analytiques de ces équations exotiques	237
11.2	Comment faire une simulation temporelle d'un système non entier ?	238
11.2.1	Exemples de modélisation de système non entier : un système distribué	238
11.2.2	Exemples de systèmes non entiers, modèle à dérivée non entière	240
11.2.3	Utilisation de la transformée de Laplace	241
11.2.4	Revue de détail (!) pour le calcul d'une réponse temporelle	241
11.2.5	Programmes de simulation fréquentielle des systèmes non entiers	243
12	Conclusion provisoire	249
13	Annexes	250
13.1	Annexe 1 : La bibliothèque « Autoelem Toolbox »	250
13.2	Annexe 2 : Comment faire pour utiliser votre bibliothèque	250
13.3	Annexe 9 : Licences	251
13.3.1	Licence GNU GPL	251
13.3.2	Licence GNU FDL	251

Avant propos

Ce livre a pour but de donner les principales méthodes utilisées dans le domaine de l'automatique analogique (étude des systèmes à modèles linéaires continus, échantillonnés) et d'approfondir par des exercices de simulation cette discipline.

En écrivant ce polycopié j'ai voulu conforter avec un logiciel de simulation, ici Scilab (libre et gratuit), cet enseignement d'automatique. En effet il est impensable de demander aux étudiants d'acheter le logiciel professionnel Matlab qui est très cher, même si c'est le logiciel le plus connu et le plus utilisé.

Je voudrais dire ici que Scilab, bien qu'il soit libre et gratuit, n'est pas un logiciel bricolé, il est robuste car réalisé au départ par des professionnels de l'informatique et de l'automatique : on le doit à un Institut de recherche français l'INRIA en collaboration avec l'ENPC¹. Le logiciel et le service qui l'accompagne est maintenant maintenu et développé par la société SCILAB ENTREPRISES.

Je pense que ce document peut être utile pour les débutants et même les experts en automatique analogique ; vous pouvez bien entendu le diffuser librement et gratuitement, cela va de soit (ceci s'adresse aux enseignants et étudiants), sous toute forme de support possible. De même, si vous souhaitez le compléter, le corriger, apporter vos propres remarques sur ce document et les méthodes de simulation qui lui sont attachées, vous voudrez bien m'en informer à l'adresse électronique donnée, je répondrais à tout courrier électronique relatif aux questions traitées dans ce document.

Il est **indispensable** d'installer la bibliothèque de fonctions **Autoelem** que je propose, afin de refaire les exercices qui sont dans le document. De même si vous avez une certaine compétence en langues (Anglais, Allemand, Espagnol ...) vous pouvez traduire, sans en faire commerce, ce polycopié. Merci d'avance à tous les utilisateurs de me donner le retour (nous faisons de l'automatique, alors le « feedback » est indispensable).

Remarques très importantes :

1. Depuis la version 6 de Scilab, vous trouverez dans la boîte à outils proposée, les principales fonctions qui sont utiles pour l'étude des systèmes à retard pur. Ces fonctions proviennent de la boîte à outils « iodelay toolbox », en effet, cette boîte à outils ne fonctionne plus (tracé des lieux fréquentsiels) avec Scilab-6.0.0.
2. Vous verrez apparaître dans le document des fonctions permettant de dessiner les lieux fréquentsiels des systèmes avec ou sans retard. Vous devez les utiliser

1. INRIA : Institut National de Recherche en Informatique et Automatique, domaine de Voluceau-Rocquencourt ; BP 105, 78153 Le Cheyney.

ENPC : Ecole Nationale des Ponts et Chaussées, Cité Descartes ; 77455 Marne-la Vallée Cedex 02.

SCILAB ENTREPRISES : 143 bis rue Yves Le Coz ; 78000 Versailles.

Table des matières

afin de bien réaliser les programmes de ce cours : sinon vous pouvez avoir des erreurs sur le tracé des lieux de systèmes classiques, et vous ne pourrez pas tracer les lieux des systèmes avec retard, et ceci avec Scilab-6. Ces lieux se nomment `bbode.sci`, `bblack.sci`, `nnyquist.sci`, `ggainplot.sci` et `pphaseplot.sci`.

3. De même le programme `freson.sci` étant corrompu (ne donne pas la fréquence de résonance pour certains systèmes), je l'ai corrigé, mais en attendant la version officielle corrigée, vous devez utiliser la fonction `ffreson.sci` qui est dans ma boîte à outils : la macro `m_margin.sci` utilise ce programme. Dès la correction faite, vous pouvez éliminer `ffreson.sci` et corriger `m_margin.sci`.
4. Vous pouvez sans aucun problème avec un copier coller, à partir du document au format **.pdf**, faire exécuter dans une fenêtre Scilab les petits exercices que je donne. Si vous êtes sous Linux faites : copier = sélection avec le bouton gauche de la souris, coller = click sur le bouton du milieu de la souris.

1 Les logiciels de simulation et l'automatique

Il existe à ma connaissance trois logiciels de simulation en automatique : Matlab, Octave, Scilab. Je ne parle pas ici des logiciels de mathématique symbolique, comme Mathematica, Maple, Maxima ou Mupad. Ce dernier était gratuit pour une plateforme de type Linux pour l'éducation, mais est maintenant intégré à Matlab.

Ces trois logiciels s'installent sur des plateformes diverses à savoir WIN?, MAC (pas tous), divers UNIX, dont LINUX.

1.1 Matlab

On trouvera des renseignements actualisés concernant Matlab sur le site internet : <http://www.mathworks.com>.

1.1.1 Avantages

Le plus connu, utilisé en écoles d'ingénieurs, dans les iut, les universités, dans de nombreux bureaux d'études industriels, c'est un logiciel complet qui possède de nombreuses boîtes à outils. Il est de plus interfaçable avec le logiciel de mathématique symbolique Maple et possède maintenant son propre logiciel de mathématique symbolique Mupad.

1.1.2 Inconvénients

Il est très coûteux, car protégé par un copyright commercial, chaque boîte à outils est payante, toute mise à jour l'est aussi et ce logiciel doit posséder une licence par poste de travail ou par groupe de postes : licence classroom.

De plus vous n'avez pas accès au logiciel source bien que de nombreux programmes soient directement issus de bibliothèques mathématiques libres, comme par exemple la bibliothèque NETLIB, en particulier des programmes écrits en langages fortran ou en C, programmes que l'on obtient librement et gratuitement en se procurant le cdrom de cette bibliothèque : <http://www.netlib.org>.

1.2 Octave

Logiciel de simulation mathématique, d'algèbre linéaire, de simulation de fonctions, de simulation d'équations différentielles ... il est accessible sur différentes plateformes.

C'est un logiciel ouvert écrit en C++, couvert par une licence libre, licence GPL¹, et gratuit. Le site internet d'Octave se nomme : <https://www.gnu.org/software/octave/>.

1.2.1 Avantages

Logiciel non commercial, avec sa licence GPL qui donne automatiquement accès au code source, vous pouvez l'essayer en allant sur le site de ce logiciel et en l'installant sur votre machine, il existe une version pour le système d'exploitation Window : je ne la connais pas.

1.2.2 Inconvénients

Ce logiciel n'a plus été développé pendant un certain temps mais une nouvelle équipe de développeurs a pris le relais, à essayer : la syntaxe de ce logiciel est très proche de la syntaxe de matlab.

1.3 Scilab

C'est un logiciel qui a été écrit à l'origine par des français travaillant à l'INRIA Institut National de Recherche en Informatique et Automatique en collaboration avec l'ENPC, Ecole Nationale de Ponts et Chaussées ; ce logiciel a une licence proche de la licence GPL est libre et gratuit², de plus vous pouvez obtenir de l'aide par le réseau sur le site de Scilab, <http://www.scilab.org>. De même sur ce site vous pourrez télécharger le logiciel, soit les sources, soit différents binaires pour diverses plateformes informatiques. Vous trouverez aussi sur le site de Scilab, de nombreuses contributions vous permettant de traiter de nombreux problèmes. La dernière version stable du logiciel est maintenant la version 5.5.x : il existe aussi une version de développement mise à la disposition des curieux et téméraires.

1.3.1 Avantages

Ce logiciel est très complet et est très proche de Matlab quant à sa syntaxe (voir les exercices proposés : comparaison des instructions et de la programmation), de même de nombreuses boîtes à outils sont disponibles dans le logiciel ou dans les contributions de divers auteurs <https://atoms.scilab.org/>. Son développement se poursuit toujours, de plus et ici je m'adresse aux étudiants et peut être à d'autres personnes, vous ne serez pas tenté de récupérer sur internet le logiciel professionnel précédent et vous mettre ainsi dans l'illégalité ...

1.3.2 Inconvénients

C'est un logiciel libre et gratuit ...

1. Vous trouverez le texte de cette licence à la fin du document : Annexe 4.

2. La nouvelle version (V 6) est maintenant sous une licence GPL 2.

1.4 Mise en place de Scilab sur un PC-LINUX

Même si sur le site de Scilab vous pouvez trouver le logiciel Scilab sous différentes versions compilées, il me semble plus judicieux pour les utilisateurs de Linux, de compiler le logiciel et ainsi vous vous rendrez compte de la structure du logiciel et des problèmes que peut poser la compilation d'un gros logiciel.

1.4.1 Installation à partir d'un binaire

Vous devez rapatrier le binaire de Scilab depuis le site de SCILAB <http://www.scilab.org>, l'archive (la version binaire) se nomme, `scilab-N°.X.bin.linux-i686.tar.gz`. Depuis un répertoire temporaire, `/home/monrepertoire/temp` par exemple, décompressez dans un terminal X (xterm) le fichier Scilab par la commande, `tar xvzf scilab-N°.X.bin.linux-i686.tar.gz`, la commande Linux `tar` va vous créer une archive `scilab-N°.X`.

Vous pouvez maintenant installer `scilab-N°.X` soit en passant sous le compte administrateur (`root`) pour que le logiciel soit utilisable par tous, et dans ce cas, vous mettez `scilab-N°.X` dans le répertoire `/usr/share` ou dans le répertoire `/usr/local`. Puis vous devez créer un lien dans le répertoire `/usr/bin` qui est toujours dans votre variable d'environnement vers le script `/usr/share/scilab-N°.X/bin/scilab` (votre PATH), de même vous pouvez créer un raccourci sur votre bureau, ce raccourci pointera vers le script `/usr/share/scilab-N°.X/bin/scilab`. Vous pouvez maintenant effacer le fichier compressé que vous avez rapatrié : un conseil, lisez le fichier `README_Unix` situé dans le répertoire principal du logiciel.

Enfin si vous avez de la chance, vous pouvez trouver sur le site de votre distribution un scilab préparé pour vous, et alors l'installation sera automatique, cas valable en particulier pour Opensuse que j'utilise, ou pour la distribution Debian et ses dérivées.

1.4.2 Compilation du logiciel

Depuis la version 5.X vous trouverez sur le site de Scilab http://wiki.scilab.org/Compiling_Scilab_5.x_under_GNU-Linux_Unix un document permettant de configurer et compiler Scilab, avec diverses options de configuration, divers compilateurs et bibliothèques.

1.4.3 Quelles bogues connus, y remédier, quelques améliorations

Je ne parlerais ici que de quelques petits bogues ou imperfections que j'ai repérés dans certains programmes situés dans des sous répertoires du répertoire `SCI/modules`, ces bogues, ces imperfections, concernent des fonctions utilisées en automatique classique : vous trouverez dans le corps du document des remarques à ce sujet.

1.4.4 Notes sur les logiciels LYX et XFIG et INKSCAPE

Le document que vous avez sous vos yeux a été écrit avec le logiciel **LYX** <http://www.lyx.org> . Je n'aurais jamais pu écrire un tel document avec un traitement de texte classique, il fallait utiliser un logiciel comme **SCIENTIFIC WORD** <http://www.ritme.com/fr/index.html> mais vu le prix pour une licence monoposte !

Si je donne le document au format **.pdf**, je peux vous fournir le fichier source au format **.lyx**, ce qui vous permettra de compléter ce document à votre convenance, et si vous avez les moyens, de le traduire en une autre langue et de le diffuser. Le logiciel **LYX** s'installe facilement sur le système d'exploitation Linux et même sous Windows, allez sur le site de **LYX** . Enfin un site intéressant pour les francophones doit être mentionné, www.more.re.eu. Pour les utilisateurs du système Windows allez voir aussi le site <http://wiki.lyx.org/Windows/Windows> .

Quant au logiciel de dessin vectoriel **XFIG**, il permet de retoucher et compléter les figures créées par Scilab, en effet toute figure Scilab peut être sauvegardée au format **.fig**. Ceci n'est plus vrai, vous devez utiliser un logiciel de tracé vectoriel **INKSCAPE** par exemple, pour cela sauvez vos figures au format **.svg**. Avec ce document je peux donner aussi les fichiers **.svg** et les fichiers **.eps**, fichiers issues souvent de Scilab, qui pour certains, ont été complétés pour une meilleure compréhension du document.

2 Exercices d'algèbre avec Scilab

Le but de ce chapitre n'est pas de refaire un cours de mathématiques, ni de développer les résultats acquis dans un cours d'algèbre des polynômes et des matrices, mais d'illustrer les principaux résultats d'algèbre qui seront nécessaires à l'analyse d'un système en utilisant un logiciel de simulation.

Nous admettrons comme connue la définition des systèmes à modèle linéaire continu ou échantillonné ; de même les principaux résultats concernant l'outil mathématique permettant d'étudier ces systèmes, à savoir la transformée de Laplace monolatère, ou la transformée en z seront aussi admis. Dans ces conditions le modèle mathématique pourra être : soit une fonction de transfert (cas monovarié), une matrice de transfert (cas multivarié), ou des équations d'état. Dans tous les cas, mathématiquement, on est appelé à manipuler des polynômes (polynômes, vecteurs de polynômes, matrices de polynômes), des fractions rationnelles (sous forme de vecteurs et matrices éventuellement), des vecteurs et matrices de scalaires (si la modélisation sous forme de variables d'état a été préférée).

Dans ce chapitre nous donnerons des exemples de manipulations de vecteurs, matrices (de scalaires, de polynômes, de fractions rationnelles) et utiliserons les principales fonctions contenues dans le logiciel permettant de définir les systèmes linéaires, fonctions qui mettent en évidence les principales propriétés de ces systèmes.

2.1 Démarrage de Scilab

Comme je l'ai signalé précédemment, en frappant dans un terminal X, **scilab**, une fenêtre Scilab apparaît. Dans cette fenêtre vous pouvez maintenant faire exécuter ligne après ligne un programme de simulation : ce que je viens de proposer s'applique bien entendu à un système de type Unix. Pour ceux qui ne peuvent pas se passer des systèmes d'exploitations de la société Microsoft vous trouverez un exécutable **scilab.exe** (déjà compilé par Scilab.org) sur le site de cet organisme, de même vous voudrez bien lire le fichier texte **Lisez-moi-Windows.txt** associé au logiciel.

2.2 Les différentes manières d'exécuter un programme Scilab

La façon la plus évidente de démarrer une session Scilab consiste, dans une fenêtre Scilab, après le prompt de Scilab symbolisé par « **-->** », à frapper les lignes de commande les unes après les autres, en frappant la touche **entrée** après chaque ligne de

programme. Ainsi la ligne correspondante est exécutée et affichée. Si l'on refuse l'affichage des résultats il suffit de mettre un « ; » en fin de ligne de commande, ceci est utile quand on fait un calcul donnant de très nombreux résultats et que l'on ne souhaite pas remplir son écran d'une multitude de valeurs numériques.

On peut aussi, par un copier-coller, introduire des lignes de programme dans une fenêtre Scilab. Cette méthode est très utile quand on veut faire exécuter les programmes exemples donnés dans les pages de manuel : instruction `help` de la fenêtre principale de Scilab ou le bouton « ? » dans la barre de titre.

Une remarque sur les séparateurs d'expressions : il existe deux séparateurs d'expressions qui sont le point-virgule « ; » et la virgule « , » le premier comme on vient de le dire, annule l'affichage du dernier résultat de la commande qui précède (même s'il y a deux commandes) et est un séparateur d'expression. Quant à la virgule, elle sépare deux expressions mais les résultats sont tous les deux affichés.

C'est en utilisant le presse papier X que les programmes Scilab apparaissent dans ce texte : vous pouvez par ce procédé échanger dans les deux sens des lignes de programme entre un éditeur, ou un formateur de documents, ici \LaTeX ¹, et une fenêtre Scilab.

La seconde manière est d'utiliser un éditeur de texte, `emacs`, `kate` ou `kwrite` pour sa coloration syntaxique (sur linux) ou `wordpad` (sur windows)² : vous allez créer ainsi un fichier texte, que vous nommerez `mon_progr.sce`, que vous ferez exécuter par la commande : `exec("path/mon_progr" [,mode])`³.

Voici un exemple de programme permettant d'illustrer cela :

```
tau=input("donner une valeur à tau :")
k=input("donner une valeur a K :")
s=%s;den=1+tau*s;num=k;fr=num/den;
sys=syslin("c",fr)
temps=linspace(0,10,51);
h=csim("step",temps,sys);
scf();
//instruction non obligatoire
//reponse indicielle unitaire
xsetech([0.,0.,0.5,0.5]); plot2d(temps',h')
//black
xsetech([0.5,0.,0.5,0.5]);
black(sys,.01,100,"premier_ordre")
//nyquist
xsetech([0.5,0.5,0.5,0.5]);
nyquist(sys,.01,100,"premier_ordre")
```

1. La première version de ce document a été écrite avec le logiciel `Thot`, mais comme ce logiciel n'est plus développé, je me suis mis au logiciel de formatage de document \LaTeX qui permet d'écrire des gros documents à la \LaTeX (et fournir un fichier `.tex` entre autre) à consommer sans modération : on n'a plus à retenir ou à apprendre le langage de balisage de \LaTeX .

2. Depuis la version 2.7 de Scilab vous avez un petit éditeur avec Scilab.

3. Le drapeau `mode`, prend une valeur numérique entière valant : 0, -1, 1, 2, 3, 4, 7 voir l'instruction `exec` ou `mode` dans l'aide ; vous pouvez aussi faire dans une fenêtre Scilab `apropos mode`.

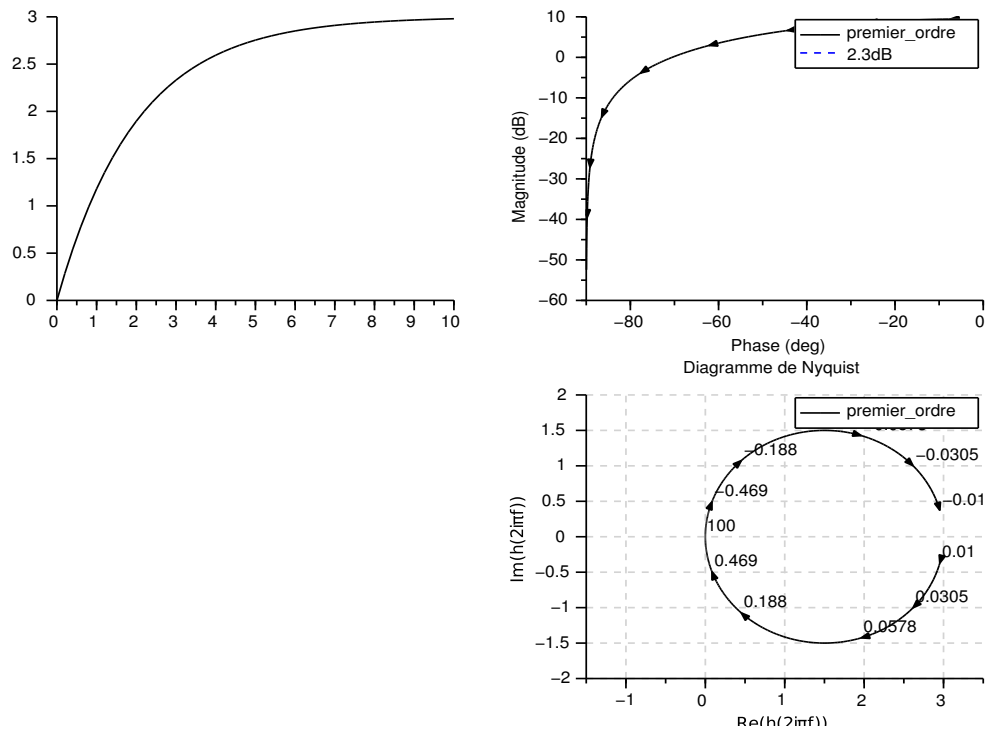


FIGURE 2.1 – Premier programme

Dans ce programme, par l'instruction `variable=input(" ")` Scilab attend que l'utilisateur donne une valeur à la variable. Ce fichier texte attaché à un courrier électronique a été exécuté ligne après ligne dans une fenêtre Scilab par des commandes copier-coller.

Vous remarquerez l'instruction `xsetech()` qui permet ici, dans une même fenêtre graphique⁴, d'afficher trois dessins : le premier situé en position (0, 0), coin haut gauche du graphe qui va être tracé sera dans un cadre de dimensions (0.5, 0.5) : c'est la réponse indicielle (FIG 2.1).

Le second graphique, le lieu de black du système, est situé, pour son coin haut gauche, en position (0.5, 0) et a pour dimensions (0.5, 0.5). Quant au dernier graphique, je souhaite le mettre dans le dernier quadrant de mon cadre à savoir en position, abscisse 0.5, ordonnée 0.5. Ce programme sera commenté dans la suite de l'exposé.

2.3 Scalaires, Vecteurs, Matrices

Dans Scilab tout est matrice : un scalaire réel ou complexe est une matrice (1×1), l'instruction fondamentale est **l'affectation**.

Pour cela on tape dans la fenêtre Scilab :

```
-->a=5
      a =
      5.
```

Par cette instruction on crée une variable « **a** » constante et on lui affecte la valeur « 5 ».

Faisons maintenant :

```
-->5
      ans =
      5.
```

dans ce dernier cas la valeur de expression est affectée à une variable par défaut « **ans** ».

Une expression Scilab peut être simple et ne mettre en jeu que des scalaires, mais elle peut aussi être composée avec des matrices et des vecteurs. Les expressions de type scalaire suivent les règles habituelles, pour des opérandes numériques (réels, complexes) on dispose des 5 opérateurs « $+$, $-$, $*$, $/$, $**$ ou $^$ » (fonction puissance) et les fonctions classiques suivantes (entre autre) :

4. J'ai sauvé ce graphique sous un format `.svg`, puis en `.eps`

2 Exercices d'algèbre avec Scilab

abs(x)	Valeur absolue, module	floor(x)	Partie entière $E(x) = \lfloor x \rfloor = n \Leftrightarrow n \leq x < n + 1$
ceil(x)	Partie entière sup : $\lceil x \rceil = n \Leftrightarrow n - 1 < x \leq n$	int(x) ou fix(x)	Partie entière anglaise : $int(x) = \lfloor x \rfloor$ si $x > 0$
round(x)	Arrondi à l'entier le plus proche	conj(x)	Conjuguée de x
imag(x)	Partie imaginaire de x	real(x)	Partie réelle de x
imult(x)	Multiplication par complexe pur (%i)	sqrt(x)	Racine carrée élément par élément
exp(x)	Exponentielle élément par élément	expm(x)	Exponentielle de matrice
sqrtn(x)	Racine carrée matricielle	log(x)	Logarithme népérien élément par élément
log10(x)	Logarithme décimal élément par élément	log2(x)	Logarithme base 2 élément par élément
logm(x)	Logarithme matriciel	Trigo ...	Fonctions trigonométriques directes inverses ...

2.3.1 Vecteurs

Un vecteur peut être une ligne ou une colonne de nombres réels ou complexes, de booléens, de polynômes, de fractions rationnelles, de systèmes linéaires, de chaînes de caractères.

Des exemples :

```
-->v=[2,-3+%i,7]
```

```
v =
```

```
! 2. - 3. + i 7. !
```

Création d'un vecteur ligne : des crochets, les éléments de la ligne séparés par des virgules ou des blancs.

```
-->v'
```

```
ans =
```

```
! 2. !
```

```
! - 3. - i !
```

```
! 7. !
```

Calcul de vecteur transposé : le « ' ».

```
-->w=[-3;-3+%i;2]
```

```
w =
```

```
! - 3. !
```

```
! - 3. + i !
```

```
! 2. !
```

Un vecteur colonne : des crochets, les éléments de la colonne séparés par des points virgules.

```
-->v'+w
```

```
ans =
```

```
! - 1. !
```

```
! - 6. !
```

```
! 9. !
```

Somme de deux vecteurs colonnes

```
-->v*w
```

```
ans =
```

```
16. - 6.i
```

Produit d'une ligne par une colonne, c'est un scalaire.

```
-->w',*v
```

2 Exercices d'algèbre avec Scilab

```
ans =  
! - 6. 10. 14. !
```

Produit élément par élément d'une ligne par une ligne : signe « `.*` ». Les éléments constitutifs des vecteurs lignes sont séparés par une virgule ou un blanc (espace), tandis que pour un vecteur colonne, on utilise le point virgule. Ces signes différencient un vecteur ligne d'un vecteur colonne. La matrice vide⁵ est représentée par `[]`, elle n'a aucune ligne, aucune colonne. On notera que la transposition d'une matrice se fait en utilisant le signe « `'` ». Ce signe est aussi utilisé pour calculer le complexe conjugué d'un nombre. Les opérations de multiplication et de division sont obtenues par les signes « `*` » et « `/` » ; elles concernent les scalaires, les vecteurs, les matrices. Quant aux signes « `.*` » et « `./` », ils représentent la multiplication et division, élément par élément, utiles pour les vecteurs et matrices. Quand vous définissez un vecteur ligne faites attention à la position des blancs. L'exemple ci-dessous l'illustre.

```
-->v=[1 +3]  
//on pouvait faire  
v=[1,+3]  
v =  
! 1. 3. !  
//mais  
-->w=[1 + 3]  
w =  
4  
-->w=[1+3]  
w =  
4.
```

Faites ici attention à la position des espaces ! On peut aussi construire un vecteur d'une manière incrémentale, le caractère « `:` ».

```
-->v=5:-.5:3  
v =  
! 5. 4.5 4. 3.5 3. !
```

Le vecteur résultat commence par la première valeur donnée, ici 5 et les éléments de la ligne sont incrémentés de `-.5`, jusqu'à la valeur 3. Si l'incrément est égal à un on peut se contenter de l'instruction :

```
-->i=1:5  
i =  
! 1. 2. 3. 4. 5. !
```

Deux instructions spéciales **ones** et **zeros** définissent des vecteurs constitués de uns ou de zéros.

```
-->v=[1 5 6]  
v =  
! 1. 5. 6. !  
-->ones(v)  
ans =
```

5. Depuis la version 6 de Scilab le comportement de la matrice vide `[]` est différent alors attention !

```
! 1. 1. 1. !
-->ones(v')
ans =
! 1. !
! 1. !
! 1. !
-->ones(1:4)
ans =
! 1. 1. 1. 1. !
-->3*ones(1:4)
ans =
! 3. 3. 3. 3. !
-->zeros(v)
ans =
! 0. 0. 0. !
-->zeros(1:5)
ans =
! 0. 0. 0. 0. 0. !
```

On remarquera que les deux instructions précédentes remplacent un vecteur quelconque, ici `v`, par un vecteur de même dimension constitué de uns ou de zéros.

2.3.2 Matrices

Les éléments d'une ligne de la matrice sont séparés par des virgules ou des espaces, les éléments d'une colonne par points virgules. La multiplication de matrices par des scalaires, des vecteurs ou par d'autres matrices se fait d'une manière habituelle. L'addition et la soustraction se fait élément par élément, la multiplication et la division se font d'une manière habituelle et utilisent respectivement les opérateurs « `*` » et « `/` ». Ne pas confondre avec la multiplication élément par élément que l'on verra par la suite.

```
-->A=[2 1 4;5 -8 2] //Matrice ayant 2 lignes et 3 colonnes A(n,m).
A =
! 2. 1. 4. !
! 5. - 8. 2. !
-->b=ones(2,3)
b =
! 1. 1. 1. !
! 1. 1. 1. !
```

On définit la matrice dont les éléments sont des uns. Les instructions `ones` et `zeros` (matrice de zéros) peuvent être utilisées avec des matrices rectangles.

```
-->A.*b
ans =
! 2. 1. 4. !
! 5. - 8. 2. !
```

Ici on fait le produit élément par élément, de la matrice `A` par `b`.

2 Exercices d'algèbre avec Scilab

```
-->A*b'
```

```
ans =  
!   7.   7. !  
! - 1. - 1. !
```

On notera que l'instruction **ones** qui possède ici deux arguments séparés par une virgule, crée une matrice de dimensions égales aux arguments. Ceci est aussi valable pour l'instruction **zeros**.

On peut aussi construire une matrice de zéros ou de uns de même dimension qu'une matrice précédemment définie par les instructions :

```
-->b=ones(A)  
-->c=zeros(A)
```

Si l'on doit créer une matrice (ou une instruction) de grande dimension, on peut écrire (cette matrice, cette instruction) sur plusieurs lignes en mettant à la fin de chaque ligne deux points.

```
-->U=[1 2 3 0 0;..  
-->1 2 5 0 0;..  
-->1 2 5 0 0]
```

```
U =  
! 1. 2. 3. 0. 0. !  
! 1. 2. 5. 0. 0. !  
! 1. 2. 5. 0. 0. !
```

Une autre instruction intéressante est l'instruction **eye()**, voici deux exemples.

Par exemple la matrice unité de rang 4.

```
-->c=eye(4,4)
```

```
c =  
! 1. 0. 0. 0. !  
! 0. 1. 0. 0. !  
! 0. 0. 1. 0. !  
! 0. 0. 0. 1. !
```

La matrice unité définie par l'instruction **eye(n,m)**.

```
-->c=eye(3,2)
```

```
c =  
! 1. 0. !  
! 0. 1. !  
! 0. 0. !
```

La matrice unité de même dimensions que la matrice **A**.

```
-->I=eye(A);
```

Définir une matrice diagonale dont les éléments sont les composantes d'un vecteur, ici le vecteur **b**

```
-->b=[2 1 4 5 -8 2]
```

```
b =
```

2 Exercices d'algèbre avec Scilab

```
! 2. 1. 4. 5. - 8. 2. !
-->B=diag(b)
B =
! 2. 0. 0. 0. 0. 0. !
! 0. 1. 0. 0. 0. 0. !
! 0. 0. 4. 0. 0. 0. !
! 0. 0. 0. 5. 0. 0. !
! 0. 0. 0. 0. - 8. 0. !
! 0. 0. 0. 0. 0. 2. !
```

On peut extraire le vecteur, diagonale principale d'une matrice, en appliquant la même instruction.

```
-->c=diag(B)
c =
! 2. !
! 1. !
! 4. !
! 5. !
! - 8. !
! 2. !
```

De même il existe deux instructions **triu** et **tril** (pour triangulaire supérieure et triangulaire inférieure), (upper, lower), pour extraire respectivement la partie triangulaire supérieure et la partie triangulaire inférieure d'une matrice. Attention, ne pas confondre ces deux instructions avec l'instruction **trianfml** qui effectue une triangularisation, en faisant une série de combinaisons linéaires sur les lignes. Une matrice utile est la matrice **rand(n,m)** qui génère une matrice de nombres pseudo-aléatoires suivant une loi uniforme sur l'intervalle $[0,1[$.

```
-->ALE=rand(2,3)
ALE =
! 0.5442573 0.2312237 0.8833888 !
! 0.2320748 0.2164633 0.6525135 !
```

Une autre particularité du logiciel réside dans le fait que l'on peut utiliser les fonctions d'algèbre classique avec des matrices : on applique ces fonctions à chacun des éléments de la matrice. Exemples :

```
-->ALE=rand(2,3);
-->SALE=sqrt(ALE)

SALE =
! 0.5546252 0.4632502 0.6013619 !
! 0.9658994 0.5591440 0.5405799 !

-->EALE=exp(ALE)
! 1.3601692 1.239367 1.4356764 !
! 2.5420266 1.367032 1.3394066 !
```

On a défini deux matrices dont les éléments sont les racines carrées et les exponentielles de chacun des éléments de la matrice de départ.

2 Exercices d'algèbre avec Scilab

Mais il existe en plus dans certains cas, des fonctions de matrice comme l'exponentielle d'une matrice carrée. Le nom de ces fonctions se termine par un m.

```
-->ALE=ALE([1 2],[1 2])
```

```
ALE =
```

```
! 0.3076091 0.2146008 !
```

```
! 0.9329616 0.3126420 !
```

J'ai extrait de la matrice une sous matrice carrée.

```
-->TM=tanm(ALE)
```

```
TM =
```

```
! 0.4007827 0.2599590 !
```

```
! 1.130153 0.4068794 !
```

```
-->EX=expm(ALE)
```

```
EX =
```

```
! 1.4988522 0.3024921 !
```

```
! 1.3150629 1.5059464 !
```

```
-->SQM=sqrtm(EX)
```

```
SQM =
```

```
! 1.1955973 0.1263459 !
```

```
! 0.5492800 1.1985604 !
```

Les matrices, vecteurs, de chaînes de caractères

Comme pour les vecteurs et matrices de scalaires on peut définir des vecteurs et matrices de chaînes de caractères. On utilise pour cela des apostrophes simples « ' » ou doubles « " » (personnellement je préfère les apostrophes doubles pour différencier une chaîne de caractères de l'opérateur de transposition). De même qu'il existe des fonctions traitant des matrices de scalaires, dans le logiciel, il existe des fonctions spéciales manipulant des matrices de chaînes de caractères.

```
-->A=["x","y","z","w+v"]
```

```
A =
```

```
! x   y   !
```

```
! z w+v !
```

```
-->AT=trianfml(A)//Instruction disparue avec Scilab-6
```

```
AT =
```

```
! z           w+v   !
```

```
!           !
```

```
! 0   z*y-x*(w+v) !
```

```
-->x=4;y=-2;z=5;w=1;v=8;
```

```
-->EVAL=evstr(AT)
```

```
EVAL =
```

```
! 5.    9.  !
```

```
! 0. - 46. !
```

On a défini une matrice de chaînes de caractères, puis on a réalisé une triangulation de cette matrice, et enfin on évalue la valeur de cette matrice. Vous verrez

qu'il existe de nombreuses fonctions traitant les matrices de chaînes de caractères, ceci permet de créer et manipuler des fonctions.

2.4 Définir un polynôme par ses racines, ses coefficients. Valeur numérique d'un polynôme

Avant de définir un système linéaire nous allons faire quelques exercices sur les polynômes et les fractions rationnelles : nous faisons des mathématiques.

2.4.1 Polynôme, racines d'un polynôme

Tout d'abord nous allons définir un polynôme et voir quelques instructions se rapportant aux polynômes.

Définir un polynôme par sa racine :

```
-->s=poly(0,"s")
```

```
s =
```

```
s
```

Pour introduire dans Scilab une variable, ici `s`, nous devons la définir comme un polynôme du premier degré ayant pour racine $s = 0$. Cette instruction est nécessaire pour introduire la variable `s`. Attention ne confondez pas `"s"` avec le polynôme s : `"s"` est une chaîne de caractères, s un polynôme.

Une autre façon plus rapide d'introduire la variable utilisée dans la transformée de Laplace ou dans la transformée en z est de faire :

```
-->s=%s
```

```
s =
```

```
s
```

```
-->z=%z
```

```
z =
```

```
z
```

Par cette méthode on utilise des **variables réservées**, que l'on ne peut détruire, variables précédées du signe « `%` ». En plus des deux variables réservées précédentes, le logiciel offre des constantes spéciales réservées `%i`, `%pi`, `%e`, `%eps`. La constante `%i` est le nombre complexe pur $\sqrt{-1}$, `%pi` est le nombre $\pi = 3,1415927$, `%e` est la constante trigonométrique $e = 2,7182818$, `%eps` représente la précision de la machine : ici $2,220D-16$. Quant aux symboles `%inf` et `%nan`, ils représentent respectivement l'infini et NotANumber et le symbole `[]` est l'élément vide (rien, voir remarque précédente paragraphe 2.3.1).

Enfin Scilab possède des booléens notés `%t` (ou `%T`), `%f` (ou `%F`) pour caractériser les deux symboles logiques 1 et 0 (true, false en anglais). Ces variables sont dites réservées car elles sont protégées, ne peuvent être détruites ni sauvées (par la commande `save()`). Vous pouvez créer vos propres variables réservées par la commande `predef()` : si vous avez dans votre propre répertoire (sous Unix) un fichier script `.scilab`, vous pouvez mettre ces variables spéciales dans ce fichier.

2 Exercices d'algèbre avec Scilab

Contrairement à Matlab, il ne faut pas utiliser le caractère « % » pour faire un commentaire dans votre programme : les lignes de commentaires commencent par les deux caractères « // ».

```
-->s=%s
-->num=poly([-1,-2,-3],"s","r")
//ou num=poly([-1;-2;-3],"s")
num =
```

$$6 + 11s + 6s^2 + s^3$$

Les racines de ce polynôme sont : $\begin{bmatrix} -1 & -2 & -3 \end{bmatrix}$. C'est un vecteur ligne, (ou colonne) on peut si l'on veut, rajouter "r" comme indicatif pour rappeler que l'on définit un polynôme par ses racines : ce drapeau est facultatif. Nous avons défini ici un vecteur ligne par la syntaxe `vecteurligne=[-1,-3 -4]`, on peut **remplacer la virgule par un espace** pour séparer les éléments d'un vecteur ligne.

```
-->num=poly([-1,-2 -3],"s")
num =
```

$$6 + 11s + 6s^2 + s^3$$

On obtient le même résultat. Profitons de ce programme pour faire le calcul des racines de ce polynôme. Attention : Quand on définit un polynôme par ses racines avec l'instruction précédente, on réalise l'opération $(s - s_1)(s - s_2) \dots$ où $s_1, s_2 \dots$ sont les racines du polynôme⁶. Vous remarquerez que Scilab ordonne un polynôme dans le sens des **puissances croissantes**.

```
-->racines=roots(num)
racines =
! - 1. !
! - 2. !
! - 3. !
```

Faire attention ici : Scilab retourne les racines sous forme d'un vecteur colonne. Un vecteur colonne est défini par la syntaxe `vecteurcolonne=[-1;-2;-3]` : la présence du « ; » est obligatoire.

```
-->racines_en_ligne=racines'
racines_en_ligne =
! - 1. - 2. - 3. !
```

La mise sous forme transposée d'un vecteur se fait par le signe « ' » ; ceci est aussi valable pour une matrice⁷.

```
-->num=poly(racines_en_ligne,"s")
num =
```

$$6 + 11s + 6s^2 + s^3$$

6. Dans ce cas les instructions `poly` et `root` sont des instructions inverses. Comme `poly` et `coeff` peuvent l'être quand on définit un polynôme par ses coefficients.

7. Si on transpose une matrice A à coefficients complexes, l'opérateur « ' » réalise la transposition de la matrice conjuguée de A.

2 Exercices d'algèbre avec Scilab

Scilab est suffisamment futé pour reconstruire le polynôme demandé. On peut aussi tout simplement définir un polynôme par son expression.

```
-->s=poly(0,"s")
s =
s
-->num=6+11*s+6*s*s+s^3
num =
      2      3
    6 + 11s + 6s + s
-->num1=(s+1)*(s+2)*(s+3)
num1 =
      2      3
    6 + 11s + 6s + s
```

Cette façon de faire, très naturelle, conduit à des résultats identiques.

```
-->A=[1,2,7;3 4,8;5 6,9]
A =
!  1.  2.  7.  !
!  3.  4.  8.  !
!  5.  6.  9.  !
```

On définit une matrice A, ligne par ligne, le point virgule sépare les trois blocs (vecteurs lignes), chaque élément de ligne est séparé de son suivant par un espace ou une virgule (on peut mélanger les deux); préférer la virgule à l'espace afin de ne pas faire d'erreurs de syntaxe : en fait une matrice est un vecteur colonne de vecteurs lignes (de même dimension).

Définissons maintenant le polynôme caractéristique de la matrice A :

```
-->nA=poly(A,"s")
nA =
      2      3
 -4.602E-16 - 40s - 14s + s
-->ncA=clean(nA)
ncA =
      2      3
 -40s - 14s + s
```

Un coefficient est très petit, en dix moins seize, on lui affecte la valeur zéro en utilisant l'instruction `clean()`.

```
-->roots(ncA)
ans =
! - 1.150E-17 !
! - 2.4339811 !
! 16.4339810 !
```

En faisant `roots()`⁸ du polynôme caractéristique d'une matrice, on calcule les va-

8. Le programme `roots` calcule toutes les racines d'un polynôme de degré n. Mais attention si vous avez un polynôme avec une racine multiple ordre de multiplicité >3 vous aurez du mal à retrouver la racine multiple exactement : ceci est du à la précision de calcul de l'algorithme utilisé, regardez le manuel.

leurs propres de celle-ci et l'on obtient un vecteur colonne.

```
-->racinesncA=roots(ncA)
racinesncA =
!  0          !
! - 2.4339811 !
! 16.433981   !
-->racinesnA=clean(roots(nA))
racinesnA =
!  0          !
! - 2.4339811 !
! 16.433981   !
```

C'était un petit jeu ! Par l'instruction `spec()` on obtient, sans passer par le polynôme caractéristique de la matrice, les valeurs propres de celle ci.

```
-->valpropA=clean(spec(A))
valpropA =
! 16.433981   !
! - 2.4339811 !
!  0          !
```

J'ai profité, comme je savais qu'une des valeurs propres était très petite, pour lui affecter la valeur zéro.

2.4.2 Autres paramètres caractérisant un polynôme

Reprenons l'exemple précédent pour caractériser notre polynôme et introduire l'instruction `type()` : le type d'un objet.

```
-->s=poly(0,"s");num=poly([-1,-2,-3],"s","r");
-->type(num)
ans =
2.
-->typ=typeof(num)
typ =
polynomial
-->type(typ)
ans =
10.
```

Explicitons cette notion de type : la première instruction renvoie un entier, ici 2, qui caractérise un polynôme, un vecteur, une matrice de polynômes. La seconde instruction renvoie une chaîne de caractères suffisamment explicite. Quant à la dernière instruction elle renvoie l'entier 10 qui est du type chaîne de caractères. N'ayant pas défini la variable de retour, Scilab propose comme variable de retour **ans** (answer en anglais).

```
-->deg=degree(num)
deg =
3.
-->type(deg)
ans =
```

```

1.
-->varia=varn(num)
varia =
    s
-->type(varia)
ans =
    10.

```

Par les instructions `degree()`, `type()`, `varn()`, on détermine respectivement le degré du polynôme, le type du nombre degré, qui est ici un scalaire (un scalaire, un vecteur, une matrice de réels ou complexes est de *type* 1) et la variable du polynôme qui est le caractère (chaîne) `s`, (*type* 10) : on explicitera (en particulier pour les listes) cette notion de type dans la suite de l'exposé.

2.4.3 Vecteur de polynômes, matrices de polynômes

Comme pour les scalaires, Scilab sait reconnaître les vecteurs et matrices (pas forcément carrées) de polynômes. De même Scilab sait faire les opérations élémentaires (additions, soustraction, multiplication, élévation à la puissance, si cette opération est justifiée ...) sur ces vecteurs et matrices de polynômes ; voici un exemple :

```

-->s=%s;
-->num=[1+s,3+s^2,s;6+s+s*s,4,7-s]
num =
!
! 1 + s      3 + s^2    s      !
!
!
! 6 + s + s^2  4      7 - s    !
-->num^2
!--error 20
first argument must be square matrix

```

Ici j'ai voulu élever une matrice au carré, elle était rectangulaire, donc l'opération n'était pas valide, Scilab me l'a signalé, il renvoie une erreur numérotée avec une explication.

```

-->num.^2
ans =
!
! 1 + 2s + s^2      9 + 6s + s^2    s^2      !
!
!
! 36 + 12s + 13s^2 + 2s^3 + s^4    16      49 - 14s + s^2    !

```

Dans cet exemple au lieu de faire opération puissance, j'ai avec l'opération « `.^` », réalisé l'opération puissance (entière) élément de la matrice par élément de la matrice.

2.4.4 Valeur numérique d'un polynôme, changement d'argument

On va utiliser maintenant l'instruction `horner()`, elle permet deux choses : soit de calculer la valeur numérique d'un polynôme pour une valeur de la variable, soit de transformer ce polynôme en changeant l'argument `s`, en un polynôme, en un rapport de polynômes. Cette instruction s'applique aussi bien à des polynômes qu'à des fractions rationnelles : on verra plus loin la définition des fractions rationnelles.

```
-->s=%s;nA=-40-14*s+s*s*s;
-->valpi=horner(nA,%i)
valpi =
-40. - 15.i
```

Dans cet exemple on calcule la valeur du polynôme pour $s = j$, (c'est le complexe pur), noté `%i`, constante réservée (voir début de la section). Nous donnons un autre exemple permettant de transformer un polynôme, une fraction rationnelle, par changement de variable (polynôme, fraction rationnelle), en un polynôme ou une fraction.

```
-->valch=horner(nA,[1/s,(1-s)/(1+s),s^3])
valch =
column 1 to 2

!          2      3          2      3 !
! 1 - 14s - 40s          - 53 - 137s + 103s + 27s !
! -----          ----- !
!          3          2      3 !
!          s          1 + 3s + 3s + s !
column 3
!          3      9 !
!- 40 - 14s + s !
! ----- !
!          1          !
```

Très intéressante cette dernière commande avec Scilab : c'est déjà du calcul symbolique. On reviendra sur les fractions rationnelles (rapport de deux polynômes). Dans l'exemple choisi le polynôme `nA` vaut : $nA = -40s - 14s^2 + s^3$. Quant au deuxième argument d'entrée de la fonction `horner()` c'est un vecteur ligne de polynômes et fractions rationnelles, vecteur de valeur

$$\left[\frac{1}{s} \quad \frac{1-s}{1+s} \quad s^3 \right]$$

On peut aussi avec l'instruction `horner()` faire un changement de variable, voici un exemple :

```
-->s=%s;
-->w=poly(0,"w");//on introduit une autre variable
w =
w
-->fr=1+s+s*s;
-->frw=horner(fr,(w+1)/(w-1))
```

```

frw =
      2
    1 + 3w
-----
      2
    1 - 2w + w

```

Une autre utilisation de l'instruction `horner()` est la détermination des valeurs des dérivées successives d'un polynôme en un point (Méthode de Ruffini-Horner) un exemple :

```

-->s=%s; P=poly([1,2,3,4],"s","c") //Anticipation voir paragraphe 2.4.6
P =
      2      3
    1 + 2s + 3s + 4s
-->Q=horner(P,s+.5)//On se place au point .5.
Q =
      2      3
    3.25 + 8s + 9s + 4s
-->coeQ=coeff(Q)
coeQ =
      3.25 8. 9. 4.
-->dP1=derivat(P);dP2=derivat(dP1);dP3=derivat(dP2);//Les dérivées
-->C=horner([P,dP1,dP2,dP3],.5)//Calcul au point .5
C =
      3.25 8. 18. 24.
//Comparaison
-->coeQ.*[1,1,2,6] //[factorial(0),factorial(1),factorial(2),factorial(3)]
ans =
      3.25 8. 18. 24.//comparez avec C.

```

2.4.5 Utilisation de l'instruction `freq`

Il existe une autre instruction permettant de calculer les valeurs numériques d'un polynôme ou d'une fraction rationnelle (matrice de ... voir le manuel) : c'est l'instruction `freq()` qui permet aussi de donner la valeur d'une expression pour une valeur de la variable ou pour un vecteur.

```

-->s=%s;nA=-40-14*s+s*s*s;
-->result=freq(nA,1,[1,.1*i,3])
result =
! - 53. - 40 - 1.401i - 55. !

```

Vous ferez attention en utilisant l'instruction `freq()` : elle demande d'avoir deux polynômes (matrices de polynômes ou un ensemble de matrices A, B, C, D), numérateur puis dénominateur pour ses deux premiers arguments d'entrée : c'est pour cela que j'ai mis le scalaire 1 (Scilab l'accepte), comme deuxième argument. En toute rigueur (pour respecter le type des arguments d'entrée) je devais écrire :

```
result=freq(nA,poly(1,"s","c"),[1,.1*i,3]).
```

Une remarque : On peut très bien définir avec l'instruction `poly` un polynôme complexe mais l'instruction `freq` ne peut donner la valeur de ce polynôme pour une valeur de la variable :

```
-->x=poly(0,"x");
-->P=poly([0,1+%i],"x")

P =
Partie réelle
      2
- x + x
Partie imaginaire
- x
-->result=freq(P,1,2)
      !--error 52 Type erroné de l'argument : Une matrice
réelle attendue.
Mais l'instruction horner le permet.
-->result=horner(P,2)
result =
      2. - 2.i
```

2.4.6 Définition d'un polynôme par ses coefficients

Définir un polynôme par ses coefficients :

```
-->den=poly([1,3 2.5,1],"s","c")
den =
      2      3
      1 + 3s + 2.5s + s
-->coe=[1 3 2.5 1];ceprim=coe'
ceprim =
! 1.  !
! 3.  !
! 2.5 !
! 1.  !
-->dem=poly(ceprim,"s","c")
dem =
      2      3
      1 + 3s + 2.5s + s
```

Le troisième argument de l'instruction est obligatoire : c'est le caractère "c". A l'avant dernière commande j'ai séparé deux instructions par un point virgule « ; » le vecteur ligne ne s'affiche pas, seul le vecteur colonne `ceprim` obtenu avec l'opérateur « ' », est affiché (comme précédemment le vecteur coefficient peut être une ligne ou une colonne). Encore quelques instructions qui mettent en oeuvre des polynômes.

```
-->s=poly(0,"s");//ou s=%s
s =
s
-->num=poly([8 5 4 7 3 9 6 2 1 4],"s","c")
```


2 Exercices d'algèbre avec Scilab

```

num =
      2      3      4      5      6      7      8      9
8 + 5s + 4s + 7s + 3s + 9s + 6s + 2s + s + 4s
-->coe=coeff(num)
coe =
! 8. 5. 4. 7. 3. 9. 6. 2. 1. 4. !
-->long=length(coe)
long =
    10.
-->r1=coe(long:-2:1)
r1 =
! 4. 2. 9. 7. 5. !
-->r2=coe(long-1:-2:1)
r2 =
! 1. 6. 3. 4. !
-->r=[r1;r2]
r =
! 4. 2. 9. 7. 5. !
! 1. 6. 3. 4. 8. !

```

Dans cette session j'ai cherché à faire une table, qui est ici proche des deux premières lignes de la table de Routh⁹, d'un système qui aurait `num` comme polynôme caractéristique ; si l'on voulait faire vraiment la table de Routh il faudrait tester les dimensions de `r1` et de `r2` et compléter le vecteur `r2` à droite, avec un zéro si nécessaire.

Quelques commentaires sur ce petit programme. On définit un polynôme de la variable `s`, par ses coefficients (drapeau "c" dans l'instruction `num=poly()`), puis on recherche les coefficients de ce polynôme par l'instruction `coe=coeff(num)` et enfin on cherche la dimension `length()` de ce vecteur coefficient.

Enfin on va construire une matrice comprenant deux vecteurs lignes, `r1`, `r2`. Le premier, par l'instruction `coe(long:-2:1)`, va être constitué d'un vecteur ligne en prenant les coefficients de deux en deux depuis le dernier (c'est le coefficient de degré le plus élevé du polynôme de départ).

On pouvait tout aussi bien, sans calculer la longueur du vecteur coefficient, sortir le même résultat en exécutant les instructions : `r1=coe($:-2:1)` et `r2=coe($-1:-2:1)`. Le signe « \$ » représente le dernier élément du vecteur ligne `coe`. Quant au second vecteur, c'est aussi un vecteur ligne dont les coefficients sont ceux du polynôme de départ, pris de deux en deux, depuis l'avant dernier coefficient. Avec ces deux vecteurs lignes on construit une matrice `r`.

Une autre façon, illustrant les opérations puissance et le produit de deux vecteurs, l'un constitué de polynômes, l'autre de scalaires, est de faire le programme suivant :

```

-->S=s^(10:-1:0)
S =
! 10 9 8 7 6 5 4 3 2      !
! s s s s s s s s s s 1 !

```

9. Vous verrez cette table lors de l'étude de la stabilité des systèmes : paragraphe 5.1.3

```
-->a=[0;2;5;8;9;6;7;8;9;1;2]
-->r=S*a
r =
      2      3      4      5      6      7      8      9
2 + s + 9s + 8s + 7s + 6s + 9s + 8s + 5s + 2s
-->type(r)
ans =
      2.
```

2.4.7 Instructions relatives aux polynômes

Voici quelques instructions relatives à un polynôme.

```
-->num=6+11*s+6*s^2+s^3
num =
```

```
      2      3
6 + 11s + 6s + s
```

```
-->derivat(num)
```

```
ans =
      2
11 + 12s + 3s
```

Cette dernière instruction calcule la dérivée par rapport à la variable s , du polynôme considéré : l'instruction dérivée s'applique aussi bien à des polynômes qu'à des fractions rationnelles.

```
-->invr(num)
```

```
ans =
      1
-----
      2      3
6 + 11s + 6s + s
```

On pouvait aussi écrire :

```
-->den=1/num
```

```
den =
      1
-----
      2      3
6 + 11s + 6s + s
```

Dans les instructions qui vont suivre on cherche à déterminer les facteurs constituants le polynôme : l'instruction `factors()` renvoie une liste.

```
-->s = %s; num1=poly([1,2,3,4,5], "s", "c")
```

```
num1=
```

```
      2      3      4
1+2s+3s+4s+5s
```

```
-->f1=factors(num1)
```

```
f1=
```

```
f1(1)
```

2 Exercices d'algèbre avec Scilab

```

                2
0.4788911-0.2756645s+s
f1(2)

```

```

                2
0.4176315+1.0756645s+s

```

Scilab a trouvé deux polynômes à racines complexes conjuguées, polynômes constituant le polynôme originel et retourne une liste, mais ne retourne pas le coefficient de plus haut degré avec cette syntaxe. Nous verrons dans la section relative aux fractions rationnelles les autres syntaxes de **factors()**.

```

-->f2=polfact(num1)
f2 =
! 5                !
!                  !
!                  2 !
! 0.4788911 - 0.2756645s + s !
!                  !
!                  2 !
! 0.4176315 + 1.0756645s + s !

```

Dans cette dernière instruction Scilab factorise le polynôme de départ en trois facteurs : l'un de degré zéro et deux facteurs du second degré (parce que ce polynôme a deux couples de racines complexes conjuguées). Cette instruction donne un vecteur colonne, vecteur constitué de polynômes dont le produit des éléments est le polynôme de départ.¹⁰

Vous trouverez une instruction non documentée, factorisant un polynôme, pas une fraction rationnelle, en une liste comprenant le coefficient de degré le plus élevé et en des termes du premier et/ou second degré, de telle sorte que l'on retrouve le polynôme originel en réalisant le produit de tous ces facteurs. Cette instruction se nomme **pfactors()**, cette factorisation ressemble à la factorisation d'Evans¹¹, mais ne s'applique qu'à des polynômes : voici un exemple.

```

-->num=poly([1 2 8 4 7 3 9],"s","c");
-->[res,g]=pfactors(num)
g =
9.
res =
res(1)

```

```

                2
0.1528161 + 0.2462539s + s

```

10. Depuis la version (4.1) de Scilab l'argument de retour est un vecteur ligne et non plus un vecteur colonne :

```

f2 =
5      0.4788911 - 0.2756645s + s      0.4176315 + 1.0756645s + s

```

11. On verra par la suite une autre factorisation dite de Bode, factorisation utile pour l'étude des réponses fréquentielles des systèmes. Cette instruction **bodfact** fait partie de la bibliothèque **Autoelem** que je propose avec ce document.

2 Exercices d'algèbre avec Scilab

```

res(2)
          2
0.7599724 + 1.0995144s + s

res(3)
          2
0.9567326 - 1.012435s + s
-->num1=g*res(1)*res(2)*res(3)
num1 =
          2    3    4    5    6
1 + 2s + 8s + 4s + 7s + 3s + 9s
Nous remarquons que g est un scalaire et que res est une liste (type 15).
-->type(g)
ans =
1.
-->type(res)
ans =
15.
Transformons le tout en un vecteur.
-->A=[g;list2vec(res)]//g scalaire, et list2vec() transforme
//une liste en vecteur.
A =
9
          2
0.9567326 - 1.012435s + s
          2
0.7599724 + 1.0995144s + s
          2
0.1528161 + 0.2462539s + s
Si je veux mettre le tout sous forme de liste (y compris g qui reste un scalaire) je
peux faire :
--> res(0)=g //Remarquer le changement de position des éléments ou
--> res($+1)=g//Même remarque
Si je voulais avoir pour g un type polynôme, comme res(3) par exemple, il faut
faire g=g*s^0 avant le changement en liste.
Jouons maintenant avec des instructions qui concernent deux polynômes.
-->num=6+11*s+6*s^2+s^3;
-->x=ldiv(num1,num,6)
x =
!      9.  !
!    - 51. !
!    214. !
!   - 773. !
!   2598. !
!  - 8367. !

```

```
-->[r,q]=pdiv(num1,num)
```

```
q =
```

```
2 3
```

```
- 773 + 214s - 51s + 9s
```

```
r =
```

```
2
```

```
4639 + 7221s + 2589s
```

L'instruction `ldiv()` donne ici les six premiers coefficients de la division du polynôme `num1` par le polynôme `num`. Attention on divise le polynôme $1 + 2s + 8s^2 + 4s^3 + 7s^4 + 3s^5 + 9s^6$ par le polynôme $6 + 11s + 6s^2 + s^3$, division dans le sens des puissances **décroissantes**. Si ces deux polynômes sont respectivement le numérateur et dénominateur d'une transmittance d'un système linéaire continu, alors sous certaines conditions, cette division donne les paramètres de Markov du système (voir le programme `pmark()` correspondant dans la bibliothèque `Autoelem`).

On ne conserve que six coefficients dans cette division. Le résultat est donc :

$$9s^3 - 51s^2 + 214s - 773s^0 + \frac{2598}{s^1} - \frac{8367}{s^2}$$

Quant à l'autre type de division, elle nous donne le quotient et le reste de la division de `num1` par `num` : c'est la division dans le sens des puissances **croissantes** de la variable.

2.4.8 Quelques autres fonctions

Equation de Bezout : cette équation concerne deux polynômes `p1` et `p2` et retourne un polynôme `thegcd`, le plus grand commun diviseur de deux polynômes $p1(s)$ et $p2(s)$. De même cette fonction renvoie une matrice (2×2) `U` unimodale. Dans ce programme on calcule par la même occasion le plus petit commun multiple `lelcm` de $p1(s)$ et $p2(s)$: la fonction `lcm(p1,p2)`.

```
-->s=%s;p1=(s+1)*(s+2)^3*(s*s+s+1)
```

```
p1 =
```

```
2 3 4 5 6
```

```
8 + 28s + 46s + 45s + 26s + 8s + s
```

```
-->p2=(s+2)^2*(s*s+s+1)
```

```
p2 =
```

```
2 3 4
```

```
4 + 8s + 5s + s
```

```
-->[thegcd,U]=bezout(p1,p2)
```

```
U =
```

```
! 5.536E-18 1 !
```

```
! !
```

```
! ! 2 !
```

```
! 1 - 2.149E-17s - 2 - 3s - s !
```

```
thegcd =
```

2 Exercices d'algèbre avec Scilab

```

      2      3      4
      4 + 8s + 9s + 5s + s
-->U=clean(U)
U =
      0              1
              1      2
      1      - 2 - 3s - s
-->deter=det(U)
deter =
      - 1
-->Racgcd=roots(thegcd)
Racgcd =
! - 0.5 + 0.8660254i !
! - 0.5 - 0.8660254i !
! - 2.              !
! - 2.              !
-->Racp1=roots(p1)
Racp1 =
! - 0.5 + 0.8660254i !
! - 0.5 - 0.8660254i !
! - 1.              !
! - 2.              !
! - 2. + 1.686E-07i !
! - 2. - 1.686E-07i !
-->Racp2=roots(p2)
Racp2 =
! - 0.5 + 0.8660254i !
! - 0.5 - 0.8660254i !
! - 2.              !
! - 2.              !
-->clean([p1,p2]*U)
ans =
!      2      3      4      !
! 4 + 8s + 9s + 5s + s      0 !
-->thelcm=p1*U(1,2)
thelcm =
      2      3      4      5      6
      8 + 28s + 46s + 45s + 26s + 8s + s
-->lelcm=lcm([p1,p2])
//ici on calcule le plus petit commun multiple de p1 et p2
lelcm =
      2      3      4      5      6
      8 + 28s + 46s + 45s + 26s + 8s + s

```

2 Exercices d'algèbre avec Scilab

Remarque : Je propose dans la boîte à outils **Autoelem**, un programme **mroots.sci** donnant les racines multiples d'un polynôme en utilisant les instructions **bezout()** et **roots()**.

Enfin une dernière instruction mettant en oeuvre l'algorithme de Leverrier.

```
-->H=[s,s*s+2;1-s,1+s]
H =
!           2 !
! s       2 + s !
!           !
! 1 - s   1 + s !

-->invr(H)

ans =
!           2 !
! 1 + s      -2 - s !
! -----
!           3      3 !
! - 2 + 3s + s  - 2 + 3s + s !
!           !
! - 1 + s      s !
! -----
!           3      3 !
! - 2 + 3s + s  - 2 + 3s + s !

-->[NU,DE]=coffg(H)
DE =
      3
- 2 + 3s + s
NU =
!           2 !
! 1 + s      - 2 - s !
!           !
! - 1 + s      s !

-->RES=NU/DE;
```

Si H est une matrice de polynômes ou de fractions rationnelles, la fonction **invr(H)** calcule la matrice inverse de H. Quant à l'instruction **coffg(H)** elle donne le dénominateur commun DE et la matrice NU (numérateur) de la matrice inverse de H.

```
-->detr(H)
ans =
      3
- 2 + 3s + s
```

De même l'instruction **detr(H)** donne le polynôme déterminant de la matrice de polynômes ou de rationnels H.

2.5 La programmation avec Scilab

¹²Avant de continuer l'étude des fractions rationnelles, je préfère introduire quelques notions de programmation. Comme tout logiciel scientifique, Scilab permet par des instructions spéciales de réaliser en ligne ou à l'aide d'un éditeur de texte (celui de Scilab par exemple), des programmes.

2.5.1 Les fonctions, les macros

Les fonctions sont des ensembles d'instructions Scilab qui sont exécutées dans un nouvel environnement, isolant ainsi les variables introduites dans ses fonctions des variables des environnements originaux. Les fonctions peuvent être créées et exécutées de différentes manières. Par exemple les fonctions peuvent passer des arguments, on peut réaliser dans des fonctions des instructions conditionnelles et des boucles, on peut les appeler récursivement. Les fonctions peuvent être des arguments d'autres fonctions, elles peuvent être éléments de listes. La façon la plus simple de créer des fonctions est d'ouvrir un éditeur de texte, **emacs** ou **kate** par exemple, ou alors les fonctions peuvent être créées directement dans Scilab en utilisant la primitive **deff**, un exemple :

```
-->deff("[x]=foo(y)","if y>0 then,x=1;else,x=-1;end")
-->foo(5)
ans =
    1.
-->foo(-3)
ans =
   - 1.
```

Si l'on crée une fonction à l'aide d'un éditeur de texte, on peut charger cette fonction dans l'environnement Scilab par l'instruction :

exec("chemindelafonction/nomdelafonction",-1) ^{13 14}. Ceci peut aussi être fait en cliquant sur le bouton **Fichier exécuter** de la fenêtre principale de Scilab. Cette instruction charge la ou les fonctions depuis le fichier **nomdelafonction** et compile la ou les fonctions. La première ligne du fichier contenant le programme doit impérativement commencer par l'instruction :

```
function [y1,...,yn]=jojo(x1,...,xk)
```

Ici les arguments **yi** sont les variables de sorties tandis que les variables d'entrées sont les **xi**. N'oublier pas de faire un retour chariot ¹⁵, à la fin de votre fichier, pour que la dernière ligne de programme soit prise en compte.

On peut depuis la version 2.6, dans la fenêtre principale de Scilab définir un programme permettant de réaliser la fonction souhaitée, voici un exemple :

12. Cette section est une traduction pratiquement mot à mot d'un ancien document Scilab : « Introduction to Scilab » .

13. On peut aussi avec l'instruction **getd("chemindurépertoire/répertoire")** dans une fenêtre Scilab, charger et compiler l'ensemble des macros, fichiers texte avec le suffixe **.sci**, contenues dans le répertoire.

14. Appeler votre programme avec un suffixe **.sce** et une fonction avec le suffixe **.sci**.

15. Je ne sais pas si cela reste valable avec les dernières versions de Scilab : n'oublier donc pas de mettre l'instruction **endfunction** à la fin de votre macro .


```
-->function [u]=creneau(t,T,T1)
-->u=bool2s(T<=t)-bool2s(t>(T+T1));
-->endfunction
-->//La fonction est comprise entre function et
-->//endfunction
```

Cette fonction réalise une fonction créneau et utilise une instruction `bool2s()` qui sera commentée dans un des paragraphes qui suit.

2.5.2 La programmation

L'une des plus intéressantes fonctionnalités de Scilab réside dans la possibilité de créer et d'utiliser des fonctions. Ceci permet de développer des programmes spécialisés qui peuvent ensuite être mis dans Scilab, d'une manière simple et modulaire, à travers l'utilisation de bibliothèques spécialisées (voir Annexe 2). Dans ce chapitre nous allons traiter les sujets suivants.

- Les outils de programmation.
- La définition et l'utilisation de fonctions.
- La définition d'opérateurs pour de nouveau type de données.

Scilab possède un nombre important d'outils de programmation, comprenant les boucles, les instructions conditionnelles, la sélection et la création de nouveaux environnements. Les tâches les plus importantes de programmation peuvent être accomplies dans le cadre des fonctions, voici les principaux outils de programmation.

Les opérateurs de comparaison

Il existe six méthodes pour faire la comparaison entre les valeurs d'objets dans Scilab. Un tableau décrit ces méthodes.

Ces opérateurs de comparaison sont utilisés dans les instructions conditionnelles.

caractère	signification
<code>== ou =</code>	égal à
<code><=</code>	inférieur ou égal à
<code>>=</code>	supérieur ou égal à
<code><</code>	inférieur à
<code>></code>	supérieur à
<code><> ou ~=</code>	pas égal à

Associé à ces opérateurs de comparaison on trouve trois opérations sur les booléens :

- Opération logique et : `&` et `and()` voir le manuel pour la différence de syntaxe.
- Opération logique ou : `|` et `or()` voir le manuel pour la différence de syntaxe.
- Opération logique non : `~`.

Les boucles

Deux types de boucles existent dans Scilab : la boucle `for` et la boucle `while`. La boucle `for` voit sa progression indexée à un vecteur d'indices, elle se termine obliga-

2 Exercices d'algèbre avec Scilab

toirement par la commande `end`. Voici quelques exemples :

```
-->x=1;for k=1:3,x=x+k,end
x =
    2.
x =
    4.
x =
    7.
```

La boucle `for` peut s'itérer en utilisant les éléments d'un vecteur ou les colonnes d'une matrice.

```
-->x=1;for k=[6,-2,1],x=x/k,end
x =
    0.1666667
x =
    - 0.0833333
x =
    - 0.0833333
```

On peut aussi faire itérer la boucle `for` à l'aide des éléments d'une liste.

```
-->l=list(1,[1 2;3 4],"jojo");
-->for k=1,disp(k),end
    1.
! 1. 2. !
! 3. 4. !
    jojo
```

L'instruction `disp()` affiche (display) les éléments de la liste.

Quant à la boucle `while`, elle exécute d'une manière répétitive une séquence d'instructions, jusqu'au moment où une condition est satisfaite.

```
-->x=1; while x<9, x=2*x,end
x =
    2.
x =
    4.
x =
    8.
x =
   16.
```

Les boucles `for` et `while` peuvent être arrêtées par l'instruction `break`.

```
-->for k=1:3; for j=1:4;if k+j>4 then break; else disp(k);
end;end;end;
    1.
    1.
    1.
    2.
    2.
    3.
```

Les instructions conditionnelles

Deux types d'instructions conditionnelles existent dans Scilab : l'instruction **if-then-else** et **select-case**. L'instruction **if-then-else** évalue une expression et si elle est vraie, le programme exécute les instructions comprises entre l'ordre **then** et **else** (ou l'ordre **end**). Si l'expression est fausse, le programme exécute les instructions comprises entre **else** et l'ordre **end**. L'ordre **else** n'est pas obligatoire. Quant à l'ordre **elseif** il a le sens habituel et est un mot clef reconnu par l'interpréteur. Un exemple :

```
-->x=1
x =
    1.
-->if x>0 then y=-x, else, y=x, end
y =
   -1.
-->x=-3
y =
   -3.
```

L'instruction conditionnelle **select-case** compare une expression à plusieurs expressions possibles et exécute les instructions qui suivent le premier cas qui rend vrai l'expression initiale.

```
-->x=1
x =
    1.
-->select x,case 1,y=2*x,case -1,y=sqrt(x),end
y =
    2.
-->x=-1
x =
   -1
y =
    i
```

Il est possible d'introduire l'ordre **else** quand aucun cas n'est vrai.

Pour votre gouverne personnelle allez voir dans le répertoire **modules** de Scilab l'ensemble des fonctions qui vous intéressent et vous en apprendrez plus que dans n'importe quel document sur Scilab.

2.6 Définir une fraction rationnelle : quelques propriétés

Dans ce paragraphe nous allons décrire quelques instructions relatives aux fractions rationnelles avant d'introduire la notion de système linéaire.

2.6.1 Les fractions rationnelles

Une fraction rationnelle, (chaque élément d'un vecteur de fractions, d'une matrice de fractions rationnelles), est le quotient de deux polynômes de la même variable symbolique, ici, s . C'est une liste¹⁶ typée¹⁷ (*type 16*).

```
-->s=%s;
-->num=6+11*s+6*s*s+s*s*s
num =
           2   3
6 + 11s + 6s + s

-->den=poly([-1,-2,3],"s")
den =
           3
- 6 - 7s + s

-->n1=roots(num)
n1 =
! - 1. !
! - 2. !
! - 3. !

-->d1=roots(den)
d1 =
! - 1. !
! - 2. !
!   3. !

-->fr=num/den
fr =
      3 + s
-----
- 3 + s

-->simp_mode(%F)
-->fr1=num/den
fr1 =
           2   3
6 + 11s + 6s + s
-----
           3
- 6 - 7s + s
```

16. Dans Scilab la notion de liste est très importante, avec ce type de structure, on peut mélanger des objets de type différent : scalaire, matrices, polynômes, chaîne de caractères ... On peut de même définir des opérations avec ces structures (addition, division, concaténation ...). Voir les instructions `list`, `tlist`, `mlist`, `overloading` dans l'aide.

17. Dans Scilab les objets ont un type. C'est un entier prenant les valeurs suivantes : 1, 2, 4, 5, 8, 10, 11, 13, 15 (pour une liste), 16 (pour une liste typée), 17 (pour les mlists), 128 . Voir l'aide avec les instructions `type(x)` et `typeof(objet)`.

2 Exercices d'algèbre avec Scilab

Dans cet exercice on définit facilement une fraction rationnelle, mais attention si les deux polynômes ont des racines communes comme dans l'exemple choisi, par défaut Scilab simplifie la fraction rationnelle sauf si vous lui dites par l'instruction `simp_mode(%F)` de ne pas le faire. Pour revenir à la situation antérieure il faut, dans votre session Scilab, introduire l'instruction `simp_mode(%T)`. %F et %T (%f,%t) sont des variables logiques : (false, true en anglais) ceux sont des variables réservées.

```
-->nu=numer(fr1)
```

```
nu =
```

```
      2   3  
6 + 11s + 6s + s
```

```
-->de=denom(fr1)
```

```
de =
```

```
      3  
- 6 - 7s + s
```

Ces dernières instructions se passent de commentaires.

On pouvait aussi extraire le numérateur et le dénominateur de la fraction rationnelle en faisant :

```
-->nu1=fr1("num")
```

```
nu1 =
```

```
      2   3  
6 + 11s + 6s + s
```

```
-->de1=fr1("den")
```

```
de1 =
```

```
      3  
- 6 - 7s + s
```

On peut aussi réaliser les instructions suivantes pour extraire de `fr` ou `fr1` les numérateurs et dénominateurs, par exemple :

```
-->nu=fr1.num
```

```
nu =
```

```
      2   3  
6 + 11s + 6s + s
```

```
-->de1=fr1.den
```

```
de1 =
```

```
      3  
- 6 - 7s + s
```

Dans ce cas on fait appel à l'extraction du deuxième et troisième élément de la liste typée `fr1` (explication au paragraphe ci-dessus).

2.6.2 Matrices, vecteurs de fractions rationnelles vus comme des listes

Nous allons voir comment Scilab traite les fractions rationnelles et comment sont stockées les données relatives à une fraction rationnelle.

2 Exercices d'algèbre avec Scilab

Nous reprenons l'exemple de la fraction précédente qui vaut :

$$fr1 = \frac{6 + 11s + 6s^2 + s^3}{-6 - 7s + s^3}$$

Scilab stocke cette fraction sous forme d'une liste typée (*type 16*) : voici un exemple de session permettant d'extraire les éléments de cette liste typée **tlist** .

```
-->typ=type(fr1)
typ =
    16.
-->typeof(fr1)
ans =
    rational
-->elem1=fr1(1)
elem1 =
! r num den dt !
-->elem12=fr1(1)(2)
elem12 =
    num
//j'ai extrait le deuxième élément du vecteur elem1
-->type(elem12)
ans =
    10.
-->type(fr1(1))
ans =
    10.
-->elem2=fr1(2)
elem2 =
           2   3
    6 + 11s + 6s + s
-->elem3=fr1(3)
elem3 =
           3
    - 6 - 7s + s
-->elem4=fr1(4)
elem4 =
    []
```

Comme c'est une liste typée on peut aussi **extraire** les numérateur et dénominateur par les instructions :

```
-->elem2=fr1.num
-->elem3=fr1.den
```

De même on peut **affecter** à un élément de la liste une valeur par exemple :

```
-->fr1.num=(1+s)*(1+s*s*s)
```

Nous voyons que cette liste comprend quatre éléments : le premier est un vecteur chaîne de caractères de dimension quatre où est stockée la lettre **r** pour rationnel puis la chaîne **num** puis la chaîne **den** et enfin le caractère **[]** (pas de définition pour le temps). Ensuite dans le deuxième et troisième éléments on stocke respectivement les vrais polynômes puis rien. Une autre syntaxe de cette fraction pouvait être :

```
fr1=tlist(["r","num","den","dt"],num,den,[])
```

Il est évident qu'avec ce type de structure on peut maintenant définir des opérations : extraction, insertion, addition etc. . . Bien entendu les concepteurs du logiciel ont prévu ces opérations élémentaires pour des rationnels.

Dans l'aide à la rubrique **rational** (faire dans la fenêtre Scilab : **apropos rational**), vous trouverez un exemple où le numérateur et le dénominateur sont des matrices de polynômes.

2.6.3 Quelques fonctions utiles pour l'étude des fractions rationnelles

Voici une session simple donnant les éléments simples d'une fraction rationnelle.

Cette fonction est maintenant boguée (Scilab-6), quand le polynôme numérateur est de degré supérieur ou égal à celui du dénominateur, alors dans votre session, vous devez faire :

```
-->s=%s;num=poly([-1,-2,-3],"s");den=poly([-1.5,-2.5,-3.5],"s");
-->fr = num/den // le dénominateur est de même degré que le numérateur.
fr =
```

$$\frac{6 + 11s + 6s^2 + s^3}{13.125 + 17.75s + 7.5s^2 + s^3}$$

```
-----
1.
elementsimp(1)//Tous les autres termes sont décalés
- 0.9375
-----
3.5 + s
elementsimp(3)
- 0.1875
-----
1.5 + s
elementsimp(4)
```

2 Exercices d'algèbre avec Scilab

- 0.375

2.5 + s

Nous voyons, par l'instruction `pfss()` que Scilab propose une liste donnant les éléments simples d'une fraction rationnelle (attention à cette instruction quand le polynôme dénominateur a des racines multiples)¹⁸. On trouvera dans l'aide en ligne ou dans le manuel de référence les différentes instructions traitant des fractions rationnelles.

Comme je l'ai dit à la section 2.4.7 la fonction `factors()` sait traiter les fractions rationnelles (et les systèmes linéaires). Voici un exemple de programme :

```
-->s=%s;n=poly([0.2,2,5],"s");
-->d=poly([0.1,0.3,7],"s");
-->R=n/d
R =
          2    3
- 2 + 11.4s - 7.2s + s
-----
          2    3
- 0.21 + 2.83s - 7.4s + s
-->[tn,td,g]=factors(R)
g =
1.
td =
    td(1)
- 0.1 + s
    td(2)
- 0.3 + s
    td(3)
- 7 + s
tn =
    tn(1)
- 0.2 + s
    tn(2)
- 2 + s
    tn(3)
- 5 + s
-->[tn1,td1,g1]=factors(R,"c")
g1 =
1.
td1 =
    td1(1)
```

18. Effectivement quand la fraction a des pôles multiples avec un ordre de multiplicité important (>3) vous pouvez dans certains cas avoir du mal à trouver tous les éléments simples : problème sur la précision du calcul des racines du dénominateur de la fraction.

2 Exercices d'algèbre avec Scilab

```

0.1 + s
    td1(2)
0.3 + s
    td1(3)
7 + s
    tn1 =
    tn1(1)
0.2 + s
    tn1(2)
2 + s
    tn1(3)
5 + s

```

On voit apparaître par cette commande que **factors()** sort une liste comprenant respectivement le nombre **g** rapport des coefficients de plus haut degré du numérateur et dénominateur, puis des termes du premier et/ou second degré (cas de racines complexes conjuguées) du numérateur et enfin des termes analogues pour le dénominateur. Avec la présence du drapeau "c" dans l'instruction, **factors()** retourne des termes du premier et/ou second degré dont les racines sont dans le demi plan gauche du plan complexe si à l'origine elles ne l'étaient pas (voir le manuel en ligne). Cette fonction est utile en automatique, quand on étudie le lieu d'Evans d'un système : on a affaire à la factorisation d'Evans.

Dans la boîte à outils **Autoelem** que je donne avec ce document je propose une nouvelle fonction nommée **bodfact()**. Cette fonction est une fonction mathématique, qui factorise un polynôme, une fraction rationnelle, un système SISO (Single Input Single Output), sous forme dite de Bode. Elle retourne un vecteur constitué de :

- K : le gain statique du système, en position, en vitesse, en accélération, suivant le nombre d'intégrations (de dérivations) que possède le système (le polynôme, la fraction, le système, le nombre de pôles ou zéros à l'origine). C'est le réel $s^{-L}sl(s)$ pour $s = 0$ où $sl(s)$ est la transmittance du système, la fraction, ou le polynôme.
- L : le nombre de dérivations (intégrations si L est négatif) de l'expression, voir remarque précédente.
- D : le retard pur, si on a affaire à un système retardé.
- TN : un vecteur colonne comprenant des polynômes du premier et/ou second degré de la forme $1 + \tau_1 s$ et/ou $1 + \tau_2 s + \tau_3 s^2$ (si le drapeau de **bodfact** est égal à "y2") ; de plus on a pour ce polynôme de degré deux, la relation : $\tau_2^2 - 4\tau_3 < 0$ (racines complexes conjuguées). Ces polynômes caractérisent le numérateur.
- TD : un vecteur analogue à TN caractérisant le dénominateur.

Ce n'est pas à proprement parlé une fonction utile pour étudier les fractions rationnelles qui font l'objet de cette étude, mais cette fonction est utile pour la nouvelle fonction **dbphifr**¹⁹ utilisée dans l'étude des réponses fréquentielles des systèmes : cette fonction est proche de l'instruction **pfactors**.

19. Cette fonction remplace les deux fonctions **phasemag** et **dbphi**, **phasemag** est boguée pour l'automaticien (décalage de phase de 360° dans certains cas). (Ceci vient du fait que l'on calcule le gain et la phase par l'instruction **repfreq** qui est un complexe et donc la phase est calculée modulo 360°).

2 Exercices d'algèbre avec Scilab

```

-->x=poly(0,"x");num=poly([-1 -3,-1+%i,-1-%i],"x")
num =
      2      3      4
6 + 14x + 13x + 6x + x
-->den=poly([0,0,-3,-7,2+%i,2-%i],"x")
den =
      2      3      4      5      6
105x - 34x - 14x + 6x + x
-->fr=num/den;
-->[k,l,d,tn,td]=bodfact(fr,"y2")
td =
! 1 + 0.1428571x      !
!                    !
!                    2 !
! 1 - 0.8x + 0.2x    !
tn =
! 1 + x              !
!                    !
!                    2 !
! 1 + x + 0.5x      !
d =
0
l =
-2.
k =
0.0571429
-->fr1=k*x^l*prod(tn,1)/(prod(td,1));
-->clean(fr-fr1)
ans =
0
-
1

```

Vous remarquerez qu'il y a eu une simplification d'un pôle par un zéro (ceci est voulu par le programme `bodfact()`).

Je vais profiter de cet exemple pour expliciter les instructions `prod()`, `cumprod()`, `sum()`, `cumsum()`. Nous voyons par l'exemple précédent que `prod(tn,1)` fait le produit des lignes du vecteur colonne `tn`. On pouvait écrire `prod(tn,"r")` : "r" pour row. Quant au produit des colonnes d'un vecteur ligne on écrirait : `prod(vecteur,2)` ou `prod(vecteur,"c")` ; il en est de même pour l'instruction `sum()` qui fait la somme au lieu du produit. Ces deux instructions s'appliquent aussi bien à des vecteurs qu'à des matrices.

Les instructions `cumprod()` et `cumsum()` réalisent en plus des opérations cumulatives : un exemple extrait de l'aide.

```

-->A=[1,2;3,4]
A =

```

2 Exercices d'algèbre avec Scilab

```
! 1.  2.  !
! 3.  4.  !
-->cumprod(A)
ans =
! 1.   6.  !
! 3.  24.  !
-->cumprod(A,"r")
ans =
! 1.  2.  !
! 3.  8.  !
-->cumprod(A,"c")
ans =
! 1.   2.  !
! 3.  12.  !
```

3 Définir un système linéaire : enfin un peu d'automatique

3.1 Fonction de transfert, matrice de transfert

Dans ce petit rappel de cours nous expliciterons la notion de fonction ou matrice de transfert isomorphe et donnerons les principales caractéristiques et formes que peut prendre cette matrice (fonction) de transfert.

3.1.1 Rappel de cours, définition de la fonction de transfert

La modélisation de connaissance, souvent nécessaire afin de comprendre le comportement dynamique d'un procédé, conduit dans la majorité des cas à un modèle liant les grandeurs de sorties (conséquences) aux grandeurs d'entrées (causes), modèle souvent représenté par un système différentiel linéaire ou non (système à constantes localisées).

Pour trouver ce modèle, lors de la description du procédé, on fait intervenir les variables temporelles d'entrée, les variables de sortie et des variables internes (qui peuvent être des variables d'état) au procédé. Ces variables sont liées par des éléments passifs dissipant ou stockant de l'énergie, des éléments fournissant de l'énergie comme les amplificateurs de puissance, des éléments assurant la mise en forme et le traitement des signaux support de ces variables.

La modélisation de connaissance utilise souvent comme méthodologie le principe d'analogie. En effet dans les systèmes, on peut classer les variables en deux catégories :

- Les variables d'effort : tension électrique, force ou couple en mécanique, pression en hydraulique, température dans des procédés thermiques.
- Les variables de flux : courant électrique, vitesse linéaire ou angulaire en mécanique, débit en hydraulique, entropie en thermique.

Pour caractériser le stockage et la dissipation d'énergie, les lois fondamentales de la physique introduisent des paramètres (des constantes localisées) caractérisant ces stockages ou transformations d'énergie. Un exemple en électricité est la transformation de l'énergie électrique en chaleur à travers une résistance R . De même pour caractériser le stockage d'énergie potentielle on introduit l'élément capacitif caractérisé par le paramètre C , quant au stockage d'énergie cinétique c'est le paramètre L (self inductance).

Si l'on revient à la modélisation on peut trouver deux lois fondamentales en physique :

- La loi des mailles reliant les variables d'effort aux variables de flux (on introduit là les paramètres cités au dessus).

3 Définir un système linéaire : enfin un peu d'automatique

— La loi des noeuds (conservation de la masse ou des débits en mécanique des fluide), qui énonce qu'en un noeud il n'y a pas génération spontanée de matière. Comme ces lois, en particulier la loi des mailles, font intervenir des équations intégrales¹, l'ensemble des relations liant les variables d'entrées les variables internes et les variables de sorties seront aussi des équations intégrales. Au sens mathématique, si ces équations sont linéaires ou linéarisables autour d'un point de fonctionnement statique, on pourra par élimination des variables internes trouver, en prenant la transformée de Laplace de ces équations, une relation linéaire liant le vecteur transformée de Laplace des grandeurs d'entrée au vecteur transformée de Laplace des grandeurs de sortie : on a ainsi pour noyau de la relation linéaire une matrice de transfert de dimension $(p \times m)$ le nombre p est le nombre de variables de sortie et m le nombre de variables d'entrée.

$$Y(s) = G(s)U(s)$$

avec $Y(s) = [y_1(s) \ y_2(s) \ \dots \ y_p(s)]^t$, $U(s) = [u_1(s) \ u_2(s) \ \dots \ u_m(s)]^t$ et $G(s) = \{g_{ij}(s)\}$

Le terme $g_{ij}(s)$ représente la fonction de transfert élémentaire liant la sortie i à l'entrée j : comme on est en linéaire, le principe de superposition s'applique et donc pour avoir le terme $g_{ij}(s)$ on met les autres entrées à zéro et l'on ne considère que la sortie numéro i .

3.1.2 Diverses représentations

Si l'on a un système monovarié (une entrée, une sortie) ou si l'on considère la sortie i liée à l'entrée j d'un système multivarié, le terme $g_{ij}(s)$ que l'on notera maintenant $g(s)$ est la transmittance isomorphe du système considéré. Cette transmittance, pour un système linéaire classique, sera un rapport de deux polynômes de la variable complexe s , généralement le degré du numérateur de cette transmittance est inférieur ou égal au degré du dénominateur (système propre), et dans ces conditions on peut écrire :

$$g(s) = \frac{b_ms^m + b_{m-1}s^{m-1} + \dots + b_1s + b_0}{a_ns^n + a_{n-1}s^{n-1} + \dots + a_1s + a_0} = \frac{N(s)}{D(s)}$$

La première idée qui vient à l'esprit pour étudier cette transmittance est de rechercher les zéros du polynôme numérateur $N(s)$ (zéros de la transmittance) et les zéros du polynôme dénominateur $D(s)$ (pôles de la transmittance). Bien entendu un autre paramètre, en plus des pôles et zéros, est nécessaire pour reconstruire cette transmittance : c'est soit $g(0)$, le gain statique, si le système ne possède pas de pôles (ou zéros) nuls ou $g(+\infty)$ (s'il existe) ou encore $s^l g(s)$ pour $s = 0$ (si le modèle possède l intégrations, respectivement l dérivations : l pôles à l'origine, respectivement l zéros à l'origine).

1. C'est ici que des équations aux dérivées non entières ou des équations aux dérivées partielles peuvent apparaître.

3.1.3 Mise sous forme éléments simples, réponse temporelle

Quand on connaît les pôles on peut facilement décomposer cette transmittance en éléments simples et $g(s)$ vaut donc :

$$g(s) = \frac{b_n}{a_n} + \sum_{i=1}^r \sum_{k=1}^{n_i} \frac{c_{ik}}{(s - p_i)^k}$$

où $b_n = 0$ sauf si $m = n$, p_i ($i \in [1, n]$) sont les pôles de la transmittance, n_i est l'ordre de multiplicité du pôle p_i . Quant à c_{ik} résidu élémentaire relatif au pôle p_i il vaut :

$$c_{ik} = \left\{ \frac{1}{(n_i - k)!} \frac{d^{(n_i - k)}}{ds^{(n_i - k)}} [(s - p_i)^{n_i} g(s)] \right\}_{s=p_i}$$

Vous reverrez avec intérêt le cours de mathématique sur la mise sous forme éléments simples d'une fraction rationnelle. Scilab réalisant cette opération par l'instruction (voir section 2.6.3) : `elementsimp=pfss(g)`.

Bien entendu à partir de cette mise sous forme éléments simples, on peut trouver facilement l'original de la fraction rationnelle et obtenir diverses réponses temporelles suivant le signal (ou plutôt la transformée de Laplace du signal) d'entrée². Un petit exemple simple pour illustrer ceci.

Soit un système de transmittance isomorphe

$$g(s) = \frac{s + 3}{s(s^2 + 2s + 2)(s + 2)(s + 1)^2}$$

Cette transmittance s'écrit, car les pôles valent $\begin{bmatrix} 0 & -1 + j & -1 - j & -2 & -1 & -1 \end{bmatrix}$, un pôle simple à l'origine et en -2 , un pôle double en -1 et deux pôles complexes conjugués, que je regroupe, en $-1 + j$ et $-1 - j$,

$$g(s) = \frac{c_1}{s} + \frac{c_2 s + c_3}{s^2 + 2s + 2} + \frac{c_4}{s + 2} + \frac{c_5}{(s + 1)} + \frac{c_6}{(s + 1)^2}$$

Calculons les résidus :

$$c_1 = sg(s) \text{ pour } s = 0; c_1 = 3/4$$

$$c_4 = (s + 2)g(s) \text{ pour } s = -2; c_4 = -1/4$$

$$c_6 = (s + 1)^2 g(s) \text{ pour } s = -1; c_6 = -2$$

Il reste à trouver les trois derniers coefficients. On multiplie $g(s)$ par s et on fait tendre la variable $s \rightarrow +\infty$ soit :

$$c_1 + c_2 + c_4 + c_5 = 0 \text{ ou :}$$

$$c_2 + c_5 = -1/2$$

On donne maintenant deux valeurs à s . Par exemple si $s = 1$ on a :

$$2c_2 + 2c_3 + 5c_5 = -1/4$$

et par exemple si $s = -3$ (qui est un zéro pour $g(s)$) on a :

$$-24c_2 + 8c_3 - 20c_5 = 35.$$

2. On peut facilement (??) avec Scilab programmer cette méthode des résidus et retrouver l'original d'une expression $g(s)$, mais le résultat n'est pas garanti.

3 Définir un système linéaire : enfin un peu d'automatique

Par la résolution de ce système de trois équations à trois inconnues c_2, c_3, c_5 on a l'ensemble des résidus, soit :

$$c_1 = 3/4, c_2 = 1/2, c_3 = 3/2, c_4 = -1/4, c_5 = -1, c_6 = -2$$

On obtient ainsi pour l'original de $g(s)$ (la réponse impulsionnelle du système)

$$h(t) = \frac{3}{4} + \frac{\sqrt{5}}{2} \exp(-t) \sin(t + \phi) - \frac{1}{4} \exp(-2t) - (1 + 2t) \exp(-t)$$

$$\phi = \arctan\left(\frac{1}{2}\right)$$

Vous vérifierez plus tard que par l'instruction `pfss(g,100)` on obtient la mise sous forme éléments simples. Vous devez donner comme deuxième argument à cette fonction un nombre assez grand sinon vous n'obtenez pas le résultat (à cause du pôle double).

```
-->s=%s;g=syslin("c",(s+3)/(s*(s*s+2*s+2)*(s+2)*(s+1)^2))
```

```
g =
```

$$3 + s$$

```
-----
      2      3      4      5      6
4s + 14s + 20s + 15s + 6s + s
```

```
-->Elem=pfss(g,100)
```

```
Elem =
```

```
Elem(1)
```

$$1.5 + 0.5s$$

```
-----
      2
```

$$2 + 2s + s$$

```
Elem(2)
```

$$- 3 - s$$

```
-----
      2
```

$$1 + 2s + s$$

```
Elem(3)
```

$$- 0.25$$

```
----
```

$$2 + s$$

```
Elem(4)
```

$$0.75$$

```
-----
- 7.054D-16 + s
```

```
-->Elem(4)=clean(Elem(4));
```

On retrouve l'original (réponse impulsionnelle) de la fonction de transfert.

Remarque : Le deuxième élément de la liste peut s'écrire : $Elem(2) = \frac{-3-s}{1+2s+s^2} = -\frac{1}{1+s} - \frac{2}{(1+s)^2}$ et on retrouve bien $c_5 = -1$ et $c_6 = -2$. C'est à cause de la racine double que l'on n'a pas le résultat théorique directement.

3 Définir un système linéaire : enfin un peu d'automatique

J'ai construis une nouvelle fonction nommée `frac_decomp.sci` qui utilise le programme `mroots.sci` et permet de mettre une fraction (un système) sous forme éléments simples.

On reprend le même système en utilisant le programme mentionné.

```
-->[ELM,elm1]=frac_decomp(g)
elm1 = 0.
ELM =
- 0.25
-----
2 + s
  - 1
-----
1 + s
  - 2
-----
          2          2
1 + 2s + s  //C'est (1+s).
0.75
-----
      s
1.5 + 0.5s
-----
          2
2 + 2s + s  //Trinôme avec racines complexes.
-->clean(sum(ELM))//Vérification.
```

3.1.4 Représentations d'Evans

Quand l'analyse et la synthèse des systèmes asservis sont effectuées par la méthode du lieu des pôles ou lieu d'Evans³, on écrit la transmittance isomorphe du système sous la forme :

$$g(s) = \frac{k_e \frac{s^m + (b_{m-1}/b_m)s^{m-1} + \dots + (b_0/b_m)}{s^l \frac{s^{n-l} + (a_{n-1}/a_n)s^{n-l-1} + \dots + (a_l/a_n)}} = \frac{k_e}{s^l} g_e(s)$$

A cause des intégrations (ce qui est souvent le cas dans certains systèmes) on met à part les pôles à l'origine dans la transmittance : ainsi $s^l g(s)$ aura une valeur finie pour le régime permanent ($s \rightarrow 0$, i.e $t \rightarrow +\infty$).

Forme polynomiale d'Evans C'est la forme précédente que l'on nomme forme polynomiale d'Evans, elle vaut :

$$g(s) = \frac{N(s)}{D(s)} = k_e \frac{g_e(s)}{s^l}$$

3. Méthode permettant de faire la synthèse d'un système bouclé quand le gain de la chaîne d'action varie.

3 Définir un système linéaire : enfin un peu d'automatique

avec :

$$g_e(s) = \frac{s^m + b_{m-1}s^{m-1} + \dots + b_1s + b_0}{s^{n-l} + a'_{n-1}s^{n-l-1} + \dots + a'_{l+1}s + a'_l}$$

Par exemple prenons

$$g(s) = \frac{s+1}{4s^5 + 3s^4 + 2s^3 + s^2}$$

on aura pour la forme polynômiale d'Evans :

$$g(s) = \frac{1}{4} \frac{1}{s^2} \frac{s+1}{s^3 + (3/4)s^2 + (1/2)s + (1/4)}$$

Forme factorisée d'Evans Quant à la forme factorisée d'Evans elle consiste à rechercher les pôles non nuls (p_i) et zéros (z_j) de $g(s)$, à mettre de côté les pôles nuls si ils existent, on obtient ainsi :

$$g(s) = \frac{k_e}{s^l} \frac{\prod_{j=1}^m (s - z_j)}{\prod_{i=l+1}^n (s - p_i)}$$

Une remarque importante : le paramètre k_e (c'est le gain à fréquence très grande) n'a aucune signification physique. Cette forme peut être obtenue par Scilab avec l'instruction `factors()`, voici un exercice (cette instruction retourne une liste).

```
-->s=%s;
-->sl=syslin("c", (1+3*s+s*s+4*s^3)/(s+6*s^2+7*s^3+8*s^4));
//j'anticipe sur la définition d'un système
-->[fnum,fden,gaininf]=factors(sl)
gaininf =
    0.5
fden =
    fden(1)
        s
    fden(2)
    0.2038539 + s
    fden(3)
                                2
    0.6131843 + 0.6711461s + s
fnum =
    fnum(1)
    0.3231486 + s
    fnum(2)
                                2
    0.7736379 - 0.0731486s + s
```

De même cette dernière expression peut être représentée géométriquement dans le plan complexe : voir le cours sur le lieu d'Evans d'un système bouclé.

3.1.5 Forme factorisée de Bode ou forme standard

Quand on traite l'analyse et la synthèse des systèmes bouclés par une méthode fréquentielle, on doit à tout prix factoriser la transmittance isomorphe sous forme de Bode. Avant la factorisation on met l'expression de la transmittance sous la forme :

$$g(s) = \frac{k}{s^l} \frac{1 + (b_1/b_0)s + (b_2/b_0)s^2 + \dots + (b_m/b_0)s^m}{1 + (a_{l+1}/a_l)s + (a_{l+2}/a_l)s^2 + \dots + (a_n/a_l)s^{n-l}} = \frac{k}{s^l} g_b(s) = \frac{k}{s^l} \frac{tn(s)}{td(s)}$$

avec $tn(0) = 1$ et $td(0) = 1$.

Puis on factorise (factorisation complète) le rationnel $g_b(s)$ en le mettant sous la forme :

$$g_b(s) = \frac{(1 + \tau_1 s)(1 + \tau_2 s) \dots [1 + (2\xi_m/\omega_{n,m})s + (1/\omega_{n,m}^2)s^2] \dots}{(1 + \lambda_1 s)(1 + \lambda_2 s) \dots [1 + (2\xi_n/\omega_{n,n})s + (1/\omega_{n,n}^2)s^2] \dots}$$

L'expression de $g(s)$ ainsi donnée met en évidence le nombre k gain statique (en position, en vitesse ...) du système, suivant que l prend les valeurs $0, 1, \dots : l$ est le nombre de pôles à l'origine de $g(s)$.

De même on voit apparaître au numérateur et dénominateur des constantes de temps τ_i ou λ_i quand les pôles (ou zéros) sont réels, ainsi que des facteurs du second degré pour chaque paire de pôles (ou zéros) complexes conjugués : ceci correspond à des systèmes élémentaires du premier et second ordre qui seront étudiés dans une section particulière (section 4.3). Voici la factorisation complète de Bode du système précédent.

```
-->[k,l,d,tn,td]=bodfact(s1,"y2")
td =
! 1 + 4.905474s          !
!                       !
!                       2 !
! 1 + 1.094526s + 1.6308312s !
tn =
! 1 + 3.0945515s          !
!                       !
!                       2 !
! 1 - 0.094551s + 1.2925944s !
d =
0.//c'est le retard pur
l =
- 1.//c'est le nombre de "dérivateurs" : 1 pôle à l'origine.
k =
1.
```

Forme standard des systèmes continus et échantillonnés

Systèmes continus

Nous venons de voir cette forme standard pour un système sans retard, mais plus généralement cette forme s'écrit :

3 Définir un système linéaire : enfin un peu d'automatique

$$g(s) = \frac{k}{s^l} e^{-ds} \frac{1 + (b_1/b_0)s + (b_2/b_0)s^2 + \dots + (b_m/b_0)s^m}{1 + (a_{l+1}/a_l)s + (a_{l+2}/a_l)s^2 + \dots + (a_n/a_l)s^{n-l}} = \frac{k}{s^l} e^{-ds} g_b(s)$$

Le terme e^{-ds} représente le terme de retard « delay » en série avec la transmittance du système (d est le retard). Vous remarquez que l'on a : $g_b(s) = 1$ pour $s = 0$.

Systèmes discrets ou échantillonnés

En ce qui concerne les systèmes discrets ou échantillonnés, ils se mettent aussi sous une forme standard, à savoir :

$$g(z) = \frac{k}{(z-1)^l} z^{-d} \frac{tn(z)}{td(z)}$$

avec $\frac{tn(z)}{td(z)} = 1$ pour $z = 1$ et td unitaire.

Pôles, zéros, gain statique, constantes de temps, amortissement, pulsation naturelle

Comme nous venons de le voir, si la transmittance isomorphe $g_b(s)$ est donnée sous la forme $g(s) = \frac{tn(s)}{td(s)}$: alors les pôles sont les racines de l'équation $td(s) = 0$ et les zéros les racines de $tn(s) = 0$. Quant au gain statique (en position, en vitesse ...) c'est la valeur de $s^l g(s)$ pour $s = 0$. C'est le comportement permanent ($t \rightarrow +\infty$) du sous système de transmittance $s^l g(s)$: c'est la valeur de k dans la factorisation de Bode.

Si l'on introduit les constantes de temps que l'on voit apparaître dans la factorisation de Bode, pour des zéros et pôles réels on a : $\tau_j = -\frac{1}{z_j}$ et $\lambda_i = -\frac{1}{p_i}$. Si les pôles (zéros) sont complexes, comme ils sont conjugués deux par deux, on peut calculer le coefficient d'amortissement ξ et la pulsation naturelle ω_n à partir de la partie réelle et de la partie imaginaire des deux pôles par les relations :

$$\xi = -\frac{R_i}{\sqrt{R_i^2 + I_i^2}} \quad \text{et} \quad \omega_n = \sqrt{R_i^2 + I_i^2}$$

avec R_i la partie réelle du pôle et I_i la partie imaginaire de ce même pôle p_i . Ce calcul est aussi valable pour les zéros de la transmittance.

3.1.6 La définition avec Scilab d'une fonction de transfert

La principale commande permettant de définir des systèmes linéaires, est l'instruction `syslin()` qui permet de définir un système continu ou échantillonné, par des polynômes (matrices de polynômes) numérateur et dénominateur, par une fraction rationnelle, ou par un quadruplet $[A, B, C, D]$ donnant les équations d'état. Voici une session Scilab permettant d'illustrer cette instruction (c'est une liste typée : *type 16*).

3 Définir un système linéaire : enfin un peu d'automatique

```
-->s=%s;
-->num=poly([0,-1,-2],"s")

num =
      2   3
2s + 3s + s
-->dem=poly([-1.5,-1.5,-2.5,-3.5],"s")

dem =
      2   3   4
6.5625 + 22s + 21.5s + 8s + s

-->fr=num/dem
fr =
      2   3
2s + 3s + s
-----
      2   3   4
6.5625 + 22s + 21.5s + 8s + s
-->sys=syslin("c",fr)
sys =
      2   3
2s + 3s + s
-----
      2   3   4
6.5625 + 22s + 21.5s + 8s + s
```

Cette session met bien en évidence dans l'instruction `syslin()` que l'on définit un système décrit par une transmittance, rapport de deux polynômes de la variable complexe s , que ce système est défini en temps continu : la présence du drapeau "c" dans l'instruction. De la même manière on pourrait avec le drapeau "d" définir un système discret, ou mettre à la place de "d" un nombre réel positif pour définir un système échantillonné de période $T_{sam} = \text{ceréel}$.

Après l'étude de la réponse fréquentielle des systèmes, je définirai un système par ses équations d'état.

Nous allons maintenant voir la structure qui caractérise un système linéaire. La syntaxe de cette instruction quand on introduit un système linéaire (qui peut être multivariable) est :

`sl=syslin("c",N,D)` ou `sl=syslin("c",G)`, où G est une fraction rationnelle ou une matrice de rationnels. En fait, comme pour les rationnels un système linéaire est une liste typée et l'on peut, comme pour toute liste, extraire, affecter, additionner, multiplier, diviser (attention au sens que l'on donne à la division) soit les éléments de cette liste, soit deux ou plusieurs listes entre elles.

`sl=tlist(["r","num","den","dt"],N,D,dom)` ou
`sl=tlist(["r","num","den","dt"],G(2),G(3),dom).`

On retrouve la même représentation que dans la définition des fractions rationnelles

3 Définir un système linéaire : enfin un peu d'automatique

(`rlist`), ce qui change par rapport à celles-ci, c'est la présence de la chaîne de caractères "`dt`" et la présence de la variable `dom` (valeur du temps).

Si on ne souhaite pas simplifier une fraction rationnelle, quand elle possède des pôles et zéros identiques, on fera : `sys=syslin("c",num,den)` sinon vous pouvez faire : `sys=syslin("c",num/den)` , on utilisera la même méthode pour définir un rationnel.

3.1.7 Définition d'un système linéaire par ses variables d'état

Nous verrons au cours de l'exposé (paragraphe 9), qu'un système linéaire continu, discret, ou échantillonné, peut être défini par un vecteur $x(t)$ ou $x(nT_s)$, vecteur constitué par la mise en colonne de variables nommées variables d'état. On définira ainsi deux équations matricielles liant le vecteur d'état aux vecteurs des entrées et des sorties , ces équations s'écrivent pour un système continu :

$$\begin{aligned}\frac{dx(t)}{dt} &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t)\end{aligned}$$

Scilab sait définir une telle représentation en utilisant la même macro sous la forme : `sl=syslin(dom,A,B,C [,D [,x0]])` avec `dom="c"` ou `dom="d"` ou encore `dom=[]` ou un scalaire. Le quadruplet $[A, B, C, D]$ est un ensemble de vecteurs et matrices, quant à x_0 c'est le vecteur d'état initial. En réalité le système est une liste typée `sl=tlist(["lss","A","B","C","D","x0","dt"],A,B,C,D,x0,dom)`.

3.1.8 Définition d'un système linéaire modèle Z,P,K

Depuis la sortie de la version 6 de Scilab on a à notre disposition une nouvelle définition d'un système linéaire ceci correspond à une `mlist` :

```
sl=mlist(["zpk","Z","P","K","dt"],z,p,k,dt).
```

3.1.9 Passage d'une représentation à une autre représentation

Outre les possibilités d'extraire, d'affecter ... , on peut par des instructions spéciales passer d'une représentation à une autre, par exemple en utilisant les macros `tf2ss.sci` ou `zpk2tf.sci` ... etc.

3.1.10 Définition d'un système bouclé avec Scilab

Nous avons donné la définition générale d'un système avec le logiciel Scilab. Nous devons maintenant introduire la notion de système bouclé (feedback en Anglais). Cette opération se fait simplement en utilisant l'opérateur « $/$ ». L'avantage de cet opérateur réside dans le fait que l'on peut utiliser des matrices donc traiter les systèmes multivariables (voir le navigateur d'aide de Scilab). Mais pour un système monovariable on peut réaliser simplement $W = \frac{G}{1+GH}$ qui correspond à la transmittance d'un système bouclé dont la chaîne d'action est G et la chaîne de retour vaut H (le bouclage est négatif).

3 Définir un système linéaire : enfin un peu d'automatique

```
-->s=%s;g=syslin("c",1/(s^3+2*s*s+.5*s+1));
-->h=syslin("c",1+4*s,1);
-->W=g/. h // Attention à la syntaxe
W =
```

$$\frac{1}{2^2 + 4.5s + 2s^2 + s^3}$$

```
-->W1=g/(1+g*h)
W1=
```

$$\frac{1}{2^2 + 4.5s + 2s^2 + s^3}$$

Faisons une vérification :

```
-->typeof(W1)//Même chose pour W.
ans =
    rational
-->W1.dt
ans =
    c
```

Remarque :

Si le retour est unitaire ou constant, $H(s) = 1(s)$ et non pas $H(s) = 1$. Voici le programme confirmant ce fait.

```
-->W=g/.1 //mais W2=g/ . 1
W = //Résultat faux.
```

$$\frac{10}{1 + 0.5s + 2s^2 + s^3}$$

```
-->uns=syslin("c",1,poly(1,"s","c"));//Transmittance 1(s).
```

//ou

```
-->uns=1/s^0;uns.dt="c";
```

```
-->W=g/.uns
```

W = //Résultat juste.

$$\frac{1}{2 + 0.5s + 2s^2 + s^3}$$

```
-->W1=g/(1+g) //Comparons.
```

W1 =

$$\frac{1}{2 + 0.5s + 2s^2 + s^3}$$

Nous voyons que le résultat est un système linéaire en temps continu. L'opérateur proposé réalise l'opération mathématique $W(s) = G(s)(I + G(s)H(s))^{-1}$.

3.2 Les graphiques dans Scilab, réponses temporelles d'un système

Les principaux graphiques utilisés par Scilab permettent de placer dans le plan complexe les pôles et zéros d'un système, de visualiser les réponses temporelles et fréquentielles, ainsi que d'étudier l'évolution des pôles d'un système bouclé quand le gain de la chaîne d'action de ce système varie (lieu d'Evans).

3.2.1 Les graphiques en deux dimensions avec Scilab

La suite de la session va nous permettre de voir les graphiques utilisés par Scilab, en particulier d'étudier les instruction `plot()`, `plot2d()`, `xsetech()`, `subplot()` ⁴

Exemple issu de la Demo de Scilab, `plot`, `plot2d`.

Voici ce que vous allez obtenir en frappant l'instruction `xtitle()` dans une fenêtre Scilab (FIG 3.1, FIG 3.2) .

```
Startup execution : loading initial environment
-->xtitle()
Demo of xtitle
x=(1:10)';
plot2d(x,x);xtitle(["Titre";"Principal"],"x","y")
Demo of plot2d
x=0:0.1:2*pi;
plot2d([x;x;x]',[sin(x);sin(2*x);sin(3*x)]',...
[-1,-2,3],"151","L1@L2@L3",[0,-2,2*pi,2]);
```

Dans cette démonstration, par l'instruction `plot2d()` on affiche une fenêtre graphique contenant la première bissectrice dans un cadre ayant les valeurs maximales de l'abscisse et de l'ordonnée, avec des graduations entières et des sous graduations. Puis par l'instruction `xtitle(...)` on met dans cette fenêtre le **Titre** en dessous **Principal** puis **x** sur l'axe des x et **y** sur l'axe des y : dans cette instruction on a défini un vecteur colonne de deux chaînes de caractères.

Quant à la démonstration sur la commande `plot2d` qui suit, vous remarquerez la syntaxe :

```
plot2d(x,y[,style,strf,leg,rect,nax])
x,y : deux matrices de même taille [nl,nc] nc donne le nombre de courbes et nl le
nombre de points sur chaque courbe.
nc : le nombre de courbes.
nl : le nombre de points sur chaque courbe par exemple :
```

4. Avec des titres, des légendes, en couleur et tout et tout.

3 Définir un système linéaire : enfin un peu d'automatique

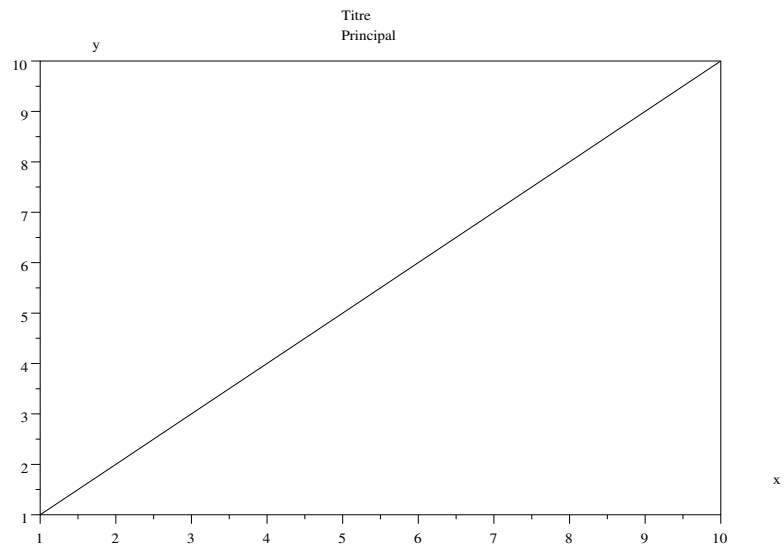


FIGURE 3.1 – Demo xtitle

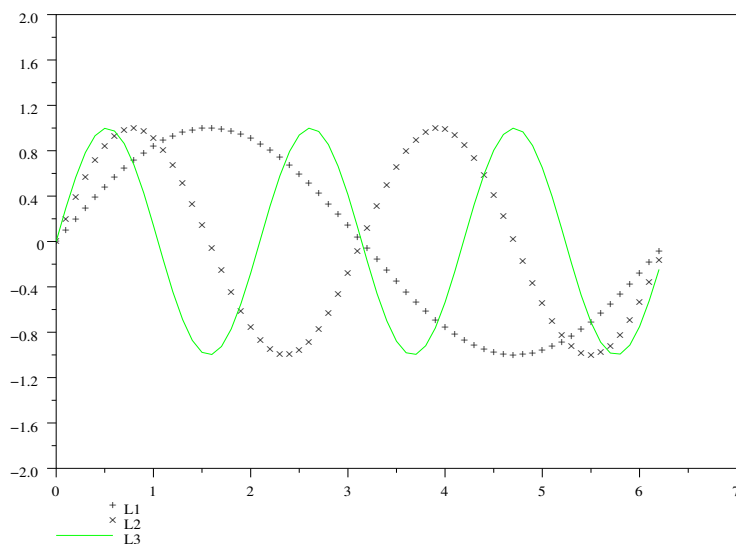


FIGURE 3.2 – Demo plot2d

3 Définir un système linéaire : enfin un peu d'automatique

`x=[1:10;1:10]'`, `y=[sin(1:10);cos(1:10)]'`
`style` : c'est un vecteur réel de taille $(1,nc)$. La façon de tracer la courbe numéro j est défini par le nombre j .
- Si `style[i]` est négatif la courbe est tracée en utilisant un caractère alphanumérique spécial portant le numéro d'identification de `style[i]`.
- Si `style[i]` est strictement positif une ligne pleine ou pointillée de numéro d'identification (ou de couleur) `abs(style[i])` est utilisé.
- Quand on désire tracer une courbe seulement, l'information `style`, peut avoir la dimension $(1,2)$ ⁵ : vecteur de composantes `[style,pos]` ou `style` est utilisé pour spécifier le type de tracé et `pos` est un entier prenant une valeur de 1 à 6, valeur spécifiant la position à utiliser pour la légende (ceci est utile quand un utilisateur souhaite tracer de nombreuses courbes, dans la même fenêtre, en appelant plusieurs fois la fonction `plot2d` et en mettant une légende sur chaque courbe).
`strf` : c'est une chaîne de caractères de longueur 3 `xyz`
`x` : des légendes sont affichées quand `x` prend la valeur 1.
`y` : ce caractère contrôle le cadre.
`y=0` : les bornes courantes sont utilisées (données par le précédent appel).
`y=1` : l'argument `rect` est utilisé pour spécifier les bornes du tracé,
`rect=[xmin,ymin,xmax,ymax]`.
`y=2` : les bornes du tracé sont calculées en utilisant les valeurs minimales et maximales de `x` et `y`.
`y=3` : même chose que pour `y=1`, mais produit une échelle proportionnelle.
`y=4` : même chose que pour `y=2`, mais produit une échelle proportionnelle.
`y=5` : même chose que pour `y=1`, mais les bornes et la façon de graduer les axes sont différentes afin d'obtenir une meilleure graduation : ce mode est utilisé quand le bouton de `zoom` est activé.
`y=6` : même chose que pour `y=2`, mais les bornes et la façon de graduer les axes sont différentes afin d'obtenir une meilleure graduation : ce mode est utilisé quand le bouton de `zoom` est activé.
`z` : contrôle l'affichage des informations relatives au cadre entourant le dessin.
`z=1` : des axes sont tracés : le nombre de graduations peut être spécifié par l'argument `nax` : c'est un vecteur à quatre entrées, `[nx,Nx,ny,Ny]` `nx(ny)` représente le nombre de sous-graduations sur l'axe des `x(y)`, `Nx(Ny)` est le nombre de graduations sur l'axe des `x(y)`.
`z=2` : le dessin est seulement contenu dans un cadre rectangulaire.
`autre valeur` : on ne dessine aucun cadre autour de la courbe.
Depuis la version 2.6 de Scilab une nouvelle syntaxe pour l'instruction `plot2d()` est apparue :
`-->plot2d(x', [sin(x);sin(2*x);sin(3*x)]', ...`
`-->[1,2,3],leg="L1@L2@L3",nax=[2,10,2,10],rect=[0,-2,2*pi,2])`
Par cette syntaxe on définit des mots clés qui permettent une plus grande lisibilité de l'instruction.

5. Il semble que dans la version 4.1 de Scilab on ne puisse plus définir la position.

Le nouveau mode graphique de scilab

À partir de la version 5.0 de Scilab l'ancien mode graphique est devenu partiellement obsolète. On peut maintenant, soit en ligne de commande, soit directement dans la fenêtre **Figure n°x**, changer les différents paramètres d'une figure. Le lecteur aura intérêt à consulter le wikilivre sur Scilab http://fr.wikibooks.org/wiki/Decouvrir_Scilab/Graphiques_et_sons

L'entité **Figure n°x** est une variable Scilab de type nouveau nommée **handle** (une poignée, en fait un pointeur) caractérisée par une arborescence : chaque entité va posséder des propriétés ainsi que des parents et/ou des descendants, certaines propriétés des descendants étant directement héritées des propriétés du parent. Voici l'ordre hiérarchique.

- Le handle **Figure n°x**, contenant des informations sur la fenêtre graphique (taille, nom de la figure (id x), positionnement, mode d'affichage ...).
- Puis vient comme « enfant » (**children**) le handle **Axes**, qui contient des informations sur les axes (échelles, graduations qui apparaissent dans la fenêtre, couleur ...).
- Puis viennent ensuite les objets graphiques de base qui composent la figure : **Polyline**, **Rectangle**, **Arc**, **Segs** ... qui peuvent aussi être agrégés grâce au handle **compound** pour former le dessin de la figure.

Voici un exemple :

Dans la fenêtre Scilab taper

```
-->plot()
```

Vous verrez apparaître une fenêtre graphique **Figure n°0**. Dans cette fenêtre cliquer sur **Edition** et choisir dans le menu déroulant **Propriétés de la figure** : vous avez maintenant sous les yeux une fenêtre **Figure Editor**. Sur le côté gauche apparaît la structure hiérarchique du handle **figure**. Cette figure a deux « enfants » **Axes(1)**, **Axes(2)**, et chacun a pour « enfant » le handle **Compound**, qui lui même a pour « enfants » différentes entités **Polyline()**. On peut bien sûr changer différentes caractéristiques du ou des graphiques.

Une autre façon de procéder est de taper dans la console Scilab :

```
-->f=gcf //get current figure.
```

On obtient :

```
f = Handle of type "Figure" with properties:
=====
children: ["Axes","Axes"]
figure_position = [321,130]
figure_size = [714,660]
axes_size = [710,538]
auto_resize = "on"
viewport = [0,0]
figure_name = "Figure n°%d"
figure_id = 0
info_message = ""
color_map = matrix 37x3
```

3 Définir un système linéaire : enfin un peu d'automatique

```
pixel_drawing_mode = "copy"
anti_aliasing = "off"
immediate_drawing = "on"
background = -2
visible = "on"
rotation_style = "unary"
event_handler = ""
event_handler_enable = "off"
user_data = []
resizefcn = ""
closerequestfcn = ""
resize = "on"
toolbar = "figure"
toolbar_visible = "on"
menubar = "figure"
menubar_visible = "on"
infobar_visible = "on"
dockable = "on"
layout = "none"
layout_options = "OptNoLayout"
default_axes = "on"
icon = ""
tag = ""
```

xsetech() et subplot()

La première instruction `xsetech()` apparaît dans ce document lors de l'exécution de notre premier programme à la section 2.2. Elle a pour but de créer des sous graphiques à l'intérieur d'une fenêtre.

L'instruction `subplot()` est une version édulcorée de `xsetech()` : elle découpe la fenêtre graphique courante en une matrice ($n \times m$) de lignes et colonnes et dans chaque case correspondante, on tracera une courbe ou un graphique 3d. Faites `help subplot` dans Scilab et copier l'exemple.

Les couleurs

Nous avons vu dans la syntaxe de l'instruction `plot2d()` que l'on pouvait gérer, par des paramètres optionnels, le style du tracé, les légendes, la taille du cadre, les graduations, etc. Cette syntaxe devient plus explicite en passant dans la liste d'appel, un mot clé, sous la forme : `plot2d(x,y,mot_clé=...)`. Ainsi, pour spécifier le style du tracé on peut utiliser la syntaxe : `plot2d(x,y,style=3)` et alors le tracé de la courbe se fera avec la couleur verte claire. Pour voir le numéro des couleurs par défaut dans Scilab, exécutez dans Scilab `getcolor()`. Si le mot clé `style` est négatif ou nul, alors le tracé est en noir sur fond blanc (normalement) avec une marque particulière.

Un exemple de manipulation des « handle »

Je souhaitais changer le contexte des lieux de Bode d'un système continu (voir ce document au paragraphe 4.2.2). Pour ce faire je réalise le programme suivant :

```
-->s = poly(0, "s");
-->h = syslin("c", (s^2+2*0.9*10*s+100)/(s^2+2*0.3*10.1*s+102.01));
-->clf();//Ouverture d'une fenêtre graphique.
-->bode(h, 0.01, 100);
```

On a défini un système linéaire continu, on ouvre une fenêtre graphique, on trace les lieux de Bode (gain, phase de ce système). Maintenant on change le contexte graphique.

```
-->a=gca();//get current axes.
Handle of type "Axes" with properties:
=====
parent: Figure
children: "Compound"
visible = "on"
axes_visible = ["on","on","on"]
axes_reverse = ["off","off","off"]
grid = [33,33]
grid_position = "background"
grid_thickness = [1,1]
grid_style = [3,3]
x_location = "bottom"
y_location = "left"
title: "Label"
x_label: "Label"
y_label: "Label"
z_label: "Label"
auto_ticks = ["on","on","on"]
x_ticks.locations = [0.01;0.1;1;10;100]
y_ticks.locations = [0;5;10]
z_ticks.locations = []
x_ticks.labels = ["0.01";"0.1";"1";"10";"100"]
y_ticks.labels = ["0";"5";"10"]
z_ticks.labels = []
ticks_format = ["","",""]
ticks_st = [1,1,1;0,0,0]
box = "off"
filled = "on"
sub_ticks = [1,4]
font_style = 6
font_size = 1
font_color = -1
fractional_font = "off"
isoview = "off"
```

3 Définir un système linéaire : enfin un peu d'automatique

```
cube_scaling = "off"
view = "2d"
rotation_angles = [0,270]
log_flags = "lnn"
tight_limits = "off"
data_bounds = [0.01,-0.1723667;100,9.4565594]
zoom_box = []
margins = [0.125,0.125,0.125,0.2202837]
auto_margins = "on"
axes_bounds = [0,0,1,0.456]
auto_clear = "off"
auto_scale = "on"
hidden_axis_color = 4
hiddencolor = 4
line_mode = "on"
line_style = 1
thickness = 1
mark_mode = "off"
mark_style = 0
mark_size_unit = "tabulated"
mark_size = 0
mark_foreground = -1
mark_background = -2
foreground = -1
background = -2
arc_drawing_method = "lines"
clip_state = "clipgrf"
clip_box = []
user_data = []
tag =
```

Cette liste donne les propriétés des deux axes (trois en réalité mais l'axe des z n'intervient pas). Je change maintenant une partie du contexte graphique par les instructions suivantes :

```
-->a.title.text = "Courbes de Bode";
-->a.title.font_style = 4;
-->a.title.font_size = 2;
Le titre a changé.
-->f = gcf();//get current figure.
Avec cette instruction je charge le handle « figure ».
-->f.background = 33;//La couleur du fond de la figure.
-->f.children.grid = [13,13];//Une grille couleur verte.
//maintenant des textes pour les sous figures.
-->f.children(2).y_label.text = "Gain en décibels";
-->f.children(2).x_label.text = "Fréquences en hertz";
-->f.children(1).y_label.text = "Phase en degrés";
```

3 Définir un système linéaire : enfin un peu d'automatique

```
-->f.children(1).x_label.text = "Fréquences en hertz";
```

Conclusion, applications à l'automatique

Scilab a prévu de nombreux graphiques spécialisés dont voici la liste.

- `plzr(système)`. Position dans le plan complexe des pôles et zéros. Le système peut être multivariable, sous forme matrice de transfert, ou sous forme état.
- `evans(boucle_ouverte,gain)`. Lieu des pôles ou d'Evans d'un système bouclé quand le gain de la chaîne d'action varie.
- `sgrid(zeta,omegan[,couleur])`. Exemple : `sgrid(0.6,2,7)`, ou `sgrid("new")`. Ce graphique peut être avantageusement associé avec la fonction `evans()`. Il donne dans le plan complexe, les courbes d'iso-amortissement et les courbes d'iso-pulsation naturelle.
- `zgrid()`. Pour les systèmes échantillonnés : lignes à amortissement constant et lignes à pulsation naturelle constante à l'intérieur du cercle de rayon unité du plan des z .
- `chart(...)`. L'abaque de Back (obsolète remplacée par `nicholschart`).
- `black(...)`. La courbe de Black.
- `bode(...)`. Les courbes de Bode de gain et phase.
- `gainplot(...)`. La courbe de Bode de gain seule.
- `phaseplot(...)`. La courbe de Bode de phase seule.
- `nyquist(...)`. La courbe de Nyquist.
- `m_circle()` ou `m_circle(gain)`. Isogains dans le plan de Nyquist (abaque de Hall).

Dans ce document nous verrons ces principaux graphiques. Je propose malgré tout, des lieux fréquentiels légèrement transformés, permettant directement de tracer les lieux pour des systèmes sans ou avec retard pur, et utilisant la macro `dbphifr.sci` exclusivement, ce qui permet d'éliminer le petit bug qui apparaît quelque fois lors du tracé des lieux (voir explication détaillée dans le répertoire **docs** de la boîte à outils sous le nom **programs.pdf** section 4).

- `bblack(...)`. La courbe de Black.
- `bbode(...)`. Les courbes de Bode.
- `ggainplot(...)`. La courbe de Bode de gain seule.
- `pphaseplot(...)`. La courbe de Bode de phase seule.
- `nnyquist(...)`. La courbe de Nyquist.

3.2.2 Visualisation des pôles et zéros d'un système

La suite de la session Scilab va nous permettre de visualiser les pôles et zéros du système précédemment introduit (FIG 3.3).

```
-->s=%s;  
-->sys=syslin("c",((1+2*s)*(1+3*s))/(s*(s*s+s+1)));  
-->plzr(sys)
```

Vous remarquerez que cette instruction trace en plus des pôles et zéros un cercle de rayon unité, cercle utile pour l'étude de la stabilité des systèmes échantillonnés (ce

3 Définir un système linéaire : enfin un peu d'automatique

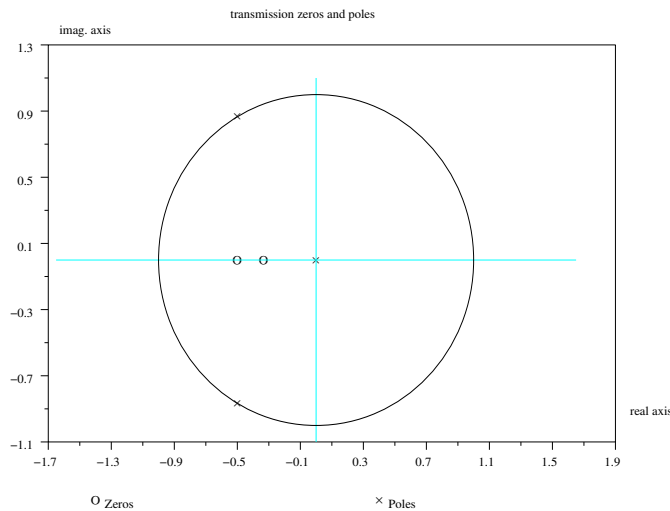


FIGURE 3.3 – Pôles et zéros

n'est plus vrai); si on veut le cercle faire :

```
-->xarc(-1,1,2,2,0,64*360)
-->arc=get("hdl");
-->arc.line_style=6;arc.foreground=6;
```

3.2.3 Simulation temporelle : réponses impulsionnelle, indicielle, à tout type de signal

La représentation temporelle d'un système continu se fait en utilisant la commande `csim()`. Cette instruction utilise un programme fortran nommé `ode`, programme complexe permettant de simuler des équations différentielles linéaires ou non, pour les curieux vous pourrez, dans le manuel en ligne, découvrir les subtilités de ce programme. Pour ce faire on doit d'abord définir une échelle de temps et une graduation de ce vecteur temps en créant un vecteur `t` ou `instants`. Quand on créera ce vecteur ne pas oublier de mettre un « ; » à la fin de l'instruction, sinon on se retrouve avec une quantité énorme de valeurs sur son écran.

La simulation d'un système : réponses impulsionnelle, indicielle

```
Startup execution: loading initial environment
-->s=%s;
-->n=poly([-1,-2],"s");
-->d=poly([-0.5,-1+%i,-1-%i],"s");
-->fr=n/d;
```

3 Définir un système linéaire : enfin un peu d'automatique

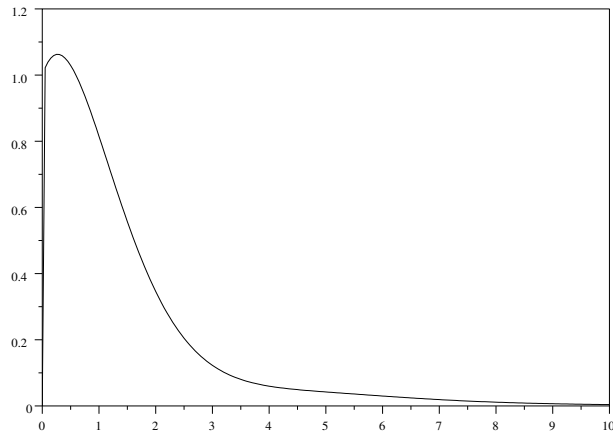


FIGURE 3.4 – Réponse impulsionnelle

```
-->sl=syslin("c",fr)
sl =

          2
    2 + 3s + s
-----
          2   3
    1 + 3s + 2.5s + s
-->t=0:.05:10;//très important le ;
-->h=csim("imp",t,sl);//très important le ;
-->clf();//pas forcément utile
-->plot(t,h)//on pourrait rajouter un titre.
```

Par ces commandes on définit un système linéaire continu de transmittance `sl`, puis une base de temps avec comme origine 0 et comme fin 10 secondes avec un pas d'échantillonnage de 0,05 seconde (c'est le vecteur ligne `t`, vous remarquerez la syntaxe : `début:pas:fin`). Enfin on simule la réponse impulsionnelle (FIG 3.4), présence du drapeau "imp" dans l'instruction `csim`. Les deux dernières instructions permettent d'une part d'effacer la fenêtre graphique, (si la fenêtre par défaut avait déjà été utilisée), puis de tracer sans aucune légende la courbe $h(t) = fonction(t)$. On pouvait tout aussi bien, dans la dernière instruction écrire : `plot(t,h)` car on ne trace qu'un simple graphique. On peut par l'instruction `xtitle` rajouter un titre à la figure : par exemple dans la même session nous allons tracer la réponse indicielle à l'aide de la fonction `plot()` et `xtitle(" ")` (FIG 3.5).

```
-->y1=csim("step",t,sl);
```


3 Définir un système linéaire : enfin un peu d'automatique

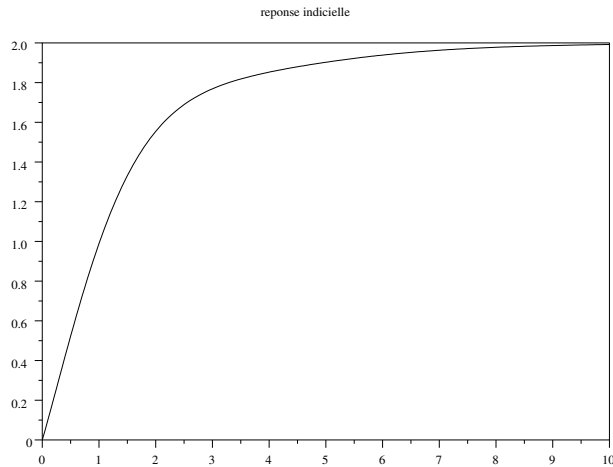


FIGURE 3.5 – Réponse indicielle

```
-->clf();//inutile dans certains cas
-->plot2d(t,y1)//ou plot(t,y1)
-->xtitle("réponse indicielle")
```

Je voudrais faire ici une remarque sur la création d'une base de temps nécessaire à la simulation de système. Nous avons créé un vecteur temps, en partant d'une borne inférieure, en se donnant un pas et ceci jusqu'une borne supérieure. Une instruction spéciale `linspace(début,fin,nombre de graduations)` peut réaliser aussi une échelle de temps : vous choisissez une borne inférieure, puis une borne supérieure et enfin un nombre entier de graduations (les bornes sont incluses dans ces graduations). Vous obtiendrez ainsi une échelle linéaire, d'une manière plus précise qu'en utilisant la première façon de procéder : il n'y a pas d'erreur cumulative. Quant à l'instruction `logspace()`, elle permet de découper une variable, suivant une échelle logarithmique. La troisième façon de procéder, consiste à créer un vecteur d'entiers et à diviser chaque élément de ce vecteur par l'inverse du nombre qui représente la quantification (le pas) demandée :

```
-->t=[0:10000]./.1000;
```

Vous utilisez ici la division élément par élément « `./` », vous pouviez aussi utiliser la division normale car le diviseur est un scalaire.

Création par Scilab d'un signal d'entrée : simulation d'un système soumis à un signal quelconque

Le but de cette section est préparer un programme permettant de générer par Scilab un signal d'entrée pour un système donné. Je voudrais ici signaler le document mis à la

3 Définir un système linéaire : enfin un peu d'automatique

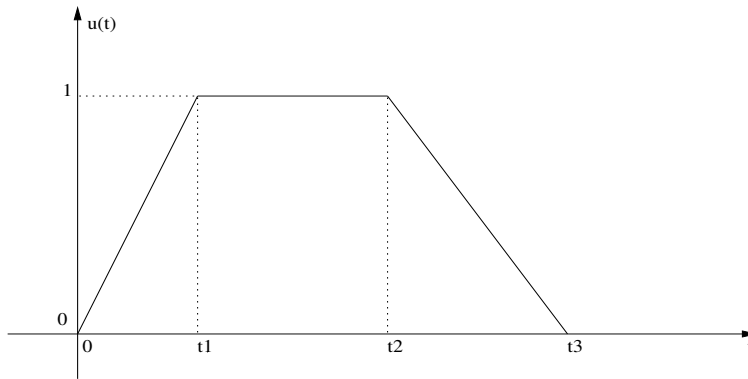


FIGURE 3.6 – Signal d'entrée

disposition des francophones par le Professeur Bruno Pinçon (Bruno.Pinçon@iecn.u-nancy.fr), document relatif à l'enseignement de l'analyse numérique, ce document est librement distribué. Je me suis très fortement inspiré d'un des programmes de ce document (section 3.5.5) pour réaliser le signal d'entrée.

On cherche à réaliser un signal d'entrée $u(t)$ constitué d'une rampe montante, d'un palier, puis d'une rampe descendante pour que le signal revienne à zéro (FIG 3.6).

Voici un exemple de programme optimisé ne nécessitant aucune boucle `for`.

```
function [u]=signentre(t,t1,t2,t3)
tinft1=(t<=t1)
//c'est un booléen donnant %t quand t<=t1, %f autrement.
tinft2=(t<=t2)
//idem
inter_0_t1=bool2s(tinft1)
//remplacement d'un booléen par les scalaires 0 ou 1
inter_t1_t2=bool2s(~tinft1&tinft2)
//idem, voir le signe ~ (pas égal à)
inter_t2_t3=bool2s(~tinft2&(t<=t3))
//idem
u=inter_0_t1.*(t/t1)+inter_t1_t2.*(1.)+inter_t2_t3.*(t3-t)/(t3-t2)
endfunction
//création du vecteur sortie
```

L'utilisation de l'instruction `bool2s()`⁶ permet de convertir une matrice de booléens en matrices de réels. De même on a utilisé l'opération « `.*` » qui représente la multiplication élément par élément pour réaliser la fonction demandée. Vous vérifierez que ce programme est nettement plus rapide que celui fait à partir de boucles `for` : utilisez pour cela la fonction `timer()`. Cet exemple de programme peut facilement être adapté à tout type de signal d'entrée défini par morceaux.

6. La fonction `bool2s` remplace le booléen `%f` par le scalaire 0 et remplace le booléen `%t` par le scalaire 1. Ce programme est un bel exemple de vectorisation des calculs.

3 Définir un système linéaire : enfin un peu d'automatique

Un exemple plus simple de programme Scilab manipulant l'instruction `csim()`. J'ai créé une fonction créneau, $u(t)$, valant 1 de l'instant 0 à l'instant $t1$ puis revenant à zéro que je range dans le fichier texte `creneau.sce` et que je mets dans le répertoire de fonctions provisoires nommé `/home/.../pointsce`.

```
function [u]=creneau(t,t1), u=bool2s(t < t1), endfunction
```

Voici un programme utilisant ce créneau comme signal d'entrée.

```
-->s=%s;  
-->sl=syslin("c",1/(1+s));  
-->t=linspace(0,10,101);  
-->t1=.5;  
-->exec("/home/.../pointsce/creneau.sce",-1);  
-->y=csim(creneau,t,sl);  
-->plot(t,y)
```

La grandeur $t1$ est le paramètre donnant la durée du créneau. Ne voulant pas le définir dans la fonction, afin de conserver toute la généralité au problème, je donne ici à $t1$ dans le programme principal, la valeur 0.5. On charge la fonction `creneau`, puis on exécute la simulation par l'instruction `csim()`.

Attention : il faut dans `csim()` donner le nom de la fonction d'entrée, ici `creneau` et pas le résultat, que j'ai appelé u . En effet u est de *type* 1 (scalaire), tandis que `creneau` est de *type* 13 (function) et `csim()` accepte pour entrée un argument de *type* function ou list. Voici le même programme en utilisant une liste.

```
-->s=%s;sl=syslin("c",1/(1+s));  
-->t=linspace(0,10,101);  
-->exec("/home/.../creneau.sce");  
-->ul=list(creneau,.5)  
ul =  
ul(1)  
[u]=function(t,t1)  
ul(2)  
0.5  
-->y=csim(ul,t,sl);  
Warning redefining function: ul  
-->plot(t,y)
```

La liste `ul` est constituée de deux éléments : la fonction et le paramètre $t1$.

On pouvait aussi faire l'étude de la réponse de ce système du premier ordre en définissant la fonction d'entrée en ligne de commande, voici un programme exemple.

```
-->s=%s;sl=syslin("c",1/(1+s));  
-->t=linspace(0,10,101);  
-->deff("u = creneau(t,t1)","if t < t1 then,u = 1;else,u = 0;end")  
-->t1=.5;  
-->y=csim(creneau,t,sl);  
-->plot(t,y)
```

Vous remarquerez, que dans ce cas l'argument de sortie u est une chaîne de caractères, de même que `creneau`. On ne peut donc, tracer le signal d'entrée $u(t)$. Un conseil : lisez attentivement le manuel sur la fonction `csim`. Une extension de cette

3 Définir un système linéaire : enfin un peu d'automatique

fonction de simulation permettant d'utiliser un vecteur u comme entrée, (à une valeur de t , on associe une composante du vecteur u , t et u ont les mêmes dimensions) : à $t(i)$, on associe $u(i)$, a été proposée dans la version 2.6 de Scilab.

Voici un programme permettant de tracer sur le même graphique l'entrée et la sortie d'un système avec la fonction créneau précédente.

```
-->s=%s;g=syslin("c",1/(s^3+2*s*s+.5*s+1))
-->t=[0:.05:20];
-->t1=1;
-->deff("u = creneau(t,t1)","u = bool2s(t < t1)")
-->x=creneau(t,t1);
-->y=csim(x,t,g);
-->plot2d(t',[x',y'],[3,6])
```

3.2.4 Introduction des conditions initiales, utilisation de la bibliothèque ode

Les condition initiales

On peut aussi introduire facilement des conditions initiales avec l'instruction `csim()`, remplacez `y=csim()` par les instructions suivantes (suite du même exercice) en encapsulant le créneau et sa durée dans une liste `ul`.

```
-->y0=-.3
-->y=csim(ul,t,s1,y0);//s1 ordre 1 donc une condition initiale.
//ou
-->y=csim(ul,t,s1,-.4);
//Mais
-->y=csim(creneau,t,g,[-2;0;0]);
//attention g ordre 3 donc vecteur initial de dimension 3.
```

Je ne reproduis pas ici les réponses : vous réaliserez l'exercice avec Scilab.

La bibliothèque ode : résolution de systèmes différentiels

Avant de décrire les outils utiles à la résolution temporelle de systèmes dynamiques je voudrais signaler une série d'articles parus dans une revue grand public Linux Magazine, articles écrits par les concepteurs de Scilab, traitant entre autre de la résolution de systèmes différentiels linéaires et non linéaires⁷. Comme je l'ai signalé au début de cette section, l'instruction `csim()` est capable de simuler la réponse d'un système à tout type de signal, mais peut en plus donner l'évolution temporelle des diverses composantes d'état d'un système linéaire (modèle décrit par un système d'équations différentielles linéaires du premier ordre à coefficients constants) : cette instruction utilise un des programmes fortran contenus dans le package « **ODEPACK** » : `csim()` fait appel à la fonction `ode()`, méthode de résolution nommée méthode d'Adams. En explorant le programme `csim.sci` situé dans le répertoire `SCI/modules/cacsd/macros` vous verrez la structure du programme.

7. Vous trouverez ces articles sur le site de Scilab, ou dans la revue Linux Magazine de mai 2000.

3 Définir un système linéaire : enfin un peu d'automatique

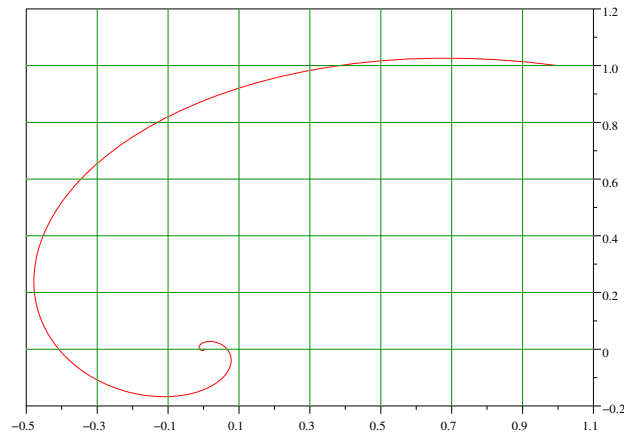


FIGURE 3.7 – Plan de phase

Voici un exemple de programme Scilab permettant de simuler et de sortir dans l'espace d'état (plan de phase) la trajectoire d'un système du second ordre à commande nulle ($u(t) = 0, \forall t \geq t_0$), partant d'un vecteur d'état initial de valeur $x_0 = [1, 1]^t$ (FIG 3.7). Attention le vecteur d'état n'est pas constitué dans le cas général, de $y(t)$ et de $\frac{dy(t)}{dt}$ mais d'une combinaison linéaire de ces deux variables et donc le vecteur d'état initial est une combinaison linéaire des composantes du vecteur conditions initiales.

```
-->s=%s;  
-->g=syslin("c", (1+s)/(1+s*s*s));  
-->m=500;T=30;  
-->t=linspace(0,T,m);  
-->deff("u=in(t)", "u=0")  
-->x0=[1;1];  
-->[rep,x]=csim(in,t,g,x0);  
//on sort la réponse et l'évolution du vecteur d'état.  
-->plot2d(x(1,:)',x(2,:)',style=5,axesflag=3)  
-->xgrid(13,1,1)//voir manuel
```

Dans le chapitre relatif aux variables d'état je reviendrais sur le plan de phase pour les systèmes linéaires d'ordre deux.

Nous allons afin de s'initier au programme `ode()`, refaire l'exercice sur le plan de phase, mais avant je vais faire un petit rappel sur les systèmes d'équations différentielles.

Système d'équations différentielles du premier ordre Soit un système d'équations différentielles du premier ordre (ou une équation différentielle d'ordre n mise sous forme

3 Définir un système linéaire : enfin un peu d'automatique

d'un système d'équations différentielles d'ordre 1), avec un vecteur conditions initiales x_0 .

$$\frac{dx}{dt} = f(t, x(t), u(t)) \text{ avec } x(t_0) = x_0 \text{ et } u(t) \text{ la commande}$$

Ici $x(t)$ est un vecteur de \mathbb{R}^n , f une fonction de $\mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^n$, et $x_0 \in \mathbb{R}^n$. On supposera l'existence et l'unicité de la solution sur un horizon d'observation $[t_0 \ t_1] = T$. Dans le cas de l'étude du système autonome ($u(t) = 0, \forall t \geq t_0$), on écrira le second membre du système différentiel f comme une fonction Scilab dans un fichier ou à la ligne de commande dans la fenêtre Scilab, avec la syntaxe suivante :

```
function f=mafonction(t,x)
//ici on code la fonction
endfunction
```

Reprenons la transmittance précédente qui vaut :

$$g(s) = \frac{1+s}{1+s+s^2}$$

qui correspond à l'équation différentielle suivante :

$$\frac{d^2 y(t)}{dt^2} + \frac{dy(t)}{dt} + y(t) = u(t) + \frac{du(t)}{dt}$$

Le solveur `ode()` traitant les systèmes différentiels d'ordre 1, on met cette équation sous la forme de deux équations différentielles d'ordre 1. Ceci peut être fait en utilisant l'instruction Scilab : `sbg=tf2ss(g)` et on obtient ainsi le système différentiel (solution non unique, voir cours sur les équations d'état d'un système) :

$$\begin{aligned} \frac{dx_1(t)}{dt} &= -0,5x_1(t) - x_2(t) - 2u(t) \\ \frac{dx_2(t)}{dt} &= 0,75x_1(t) - 0,5x_2(t) + u(t) \end{aligned}$$

avec :

$$y(t) = -0,5x_1(t)$$

Une autre mise en équation, donnant pour la première composante du vecteur d'état la valeur de la sortie $y(t)$ peut être :

$$\begin{aligned} \frac{dx_1^*(t)}{dt} &= x_2^*(t) + u(t) \\ \frac{dx_2^*(t)}{dt} &= -x_1^*(t) - x_2^*(t) \end{aligned}$$

avec :

$$y(t) = x_1^*(t)$$

équations plus simples à simuler et plus lisibles (forme compagnon observable). C'est cette forme que l'on simulera avec le solveur `ode()`. Afin de comparer avec l'exercice précédent, seul le régime autonome ($u(t) = 0, \forall t \geq 0$) sera envisagé : voici la programmation Scilab (FIG 3.8).

3 Définir un système linéaire : enfin un peu d'automatique

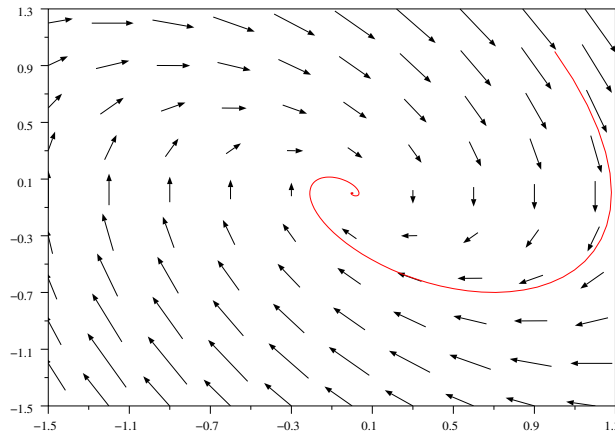


FIGURE 3.8 – Plan de phase avec « ode »

```
-->function [f]=mafonct(t,x)
-->f(1)=x(2)
-->f(2)=-x(1)-x(2)
-->endfunction
-->//on trace le champ de vecteurs
-->z=linspace(-1.5,1.2,10);
-->fchamp(mafonct,0,z,z)
-->//on résoud le système
-->x0=[1;1]; //condition initiale
-->t=linspace(0,50,501);
-->[x]=ode(x0,0,t,mafonct);
-->plot2d(x(1,:),x(2,:),5,'000')
```

Nous allons, en nous inspirant (c'est plus que de l'inspiration!)⁸ du document du Professeur Bruno Pinçon cité à la section 3.2.3, faire une petite animation permettant de visualiser les trajectoires de phase à partir de diverses conditions initiales. Voici un programme que je nomme `odeanim.sce` que je stocke dans mon répertoire :

```
function [f]=mafonct(t,x);
f(1)=x(2);
f(2)=-x(1)-x(2);
endfunction
//c'était la programmation des équations
xxmin=-2;xxmax=2;yymin=-3;ymax=3;n=10;
xx=linspace(xxmin,xxmax,n);
```

8. Voir aussi la revue Linux Magazine de mai 2000, que j'ai déjà cité.

3 Définir un système linéaire : enfin un peu d'automatique

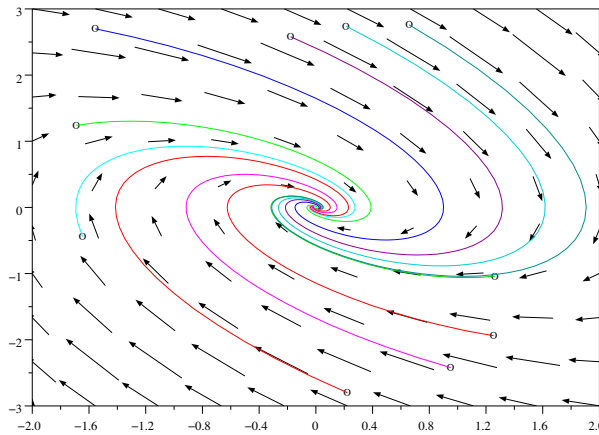


FIGURE 3.9 – Plan de phase animé

```
yy=linspace(yymin,ymax,n);
fchamp(mafonct,0,xx,yy)
//le cadre et les lignes de champ
t=linspace(0,10,101);
coul=[2,3,4,5,6,7,15,16,18,22,21,27]; num=-1;
while %t
ret=xclick();
if ret(1)==0 then//bouton gauche de la souris
plot2d(ret(2),ret(3),-9,"000")
//abscisse et ordonnée du point choisi
x0=[ret(2);ret(3)];
x=ode(x0,0,t,mafonct);
num=modulo(num+1,length(coul));
plot2d(x(1,:)',x(2,:)',coul(num+1),"000")
elseif ret(1)==2 then
//si bouton droit alors on arrête l'animation.
break
end
end
end
xclick() renvoie un vecteur ligne de trois informations : le numéro de bouton de
souris choisi, abscisse et ordonnée du point du graphique courant choisi. Pour exécuter
ce programme, faire dans une fenêtre Scilab (FIG 3.9) :
-->exec("/home/.../.../odeanim.sce",-1);
```


3 Définir un système linéaire : enfin un peu d'automatique

Si maintenant vous souhaitez simuler des équations non linéaires avec le programme `ode()`, vous n'aurez qu'à changer dans le programme précédent `mafonct` par `tafonct`, le cadre pour le tracé des lignes de champ, les couleurs et éventuellement le choix des boutons de la souris : exemple l'équation de Van der Pol :

$$\frac{d^2 y(t)}{dt^2} - a(1 - y^2(t)) \frac{dy(t)}{dt} + y(t) = 0$$

Vous reformulerez cette équation en un système différentiel et vous obtiendrez la fonction :

```
function [f]=vander(t,x,a);
f(1)=x(2);
f(2)=-x(1)+a*(1-x(1)*x(1))*x(2);
//ici on met a en paramètre
endfunction
```

Comme on a mis un paramètre `a` dans la fonction, on va créer une liste et appeler `ode()` de la manière suivante :

```
x=ode(x0,t0,t,list(vander,a))
//dans le programme principal a aura une valeur
```

Dernière remarque : les trajectoires de phase dans les deux programmes ne se ressemblent pas, la mise en équations n'est pas la même car les variables d'état sont différentes. Si on voulait comparer les deux méthodes (avec `csim()` et `ode()`), il faut définir le système directement dans le premier programme par ces équations d'état.

De même on peut avec l'instruction `param3d()` donner, dans un système d'axes 3d, la trajectoire.

```
f=figure(1); //la figure(0) est toujours dessinée
x=ode([1;2],0,t,mafonct);
param3d(x(1,:),x(2,:),t,leg="x1@x2@t",flag=[1,4],ebox=[0,3,0,3,0,10])
```

Système d'équations différentielles d'ordre supérieur à un

On considère le système mécanique décrit par la figure (FIG 3.10) :

Soit une poulie de rayon r et de moment d'inertie $I = \frac{Mr^2}{2}$ par rapport à l'axe de rotation Oz (perpendiculaire à la figure), sur laquelle s'enroule un câble inextensible dont une des extrémités est liée au sol par l'intermédiaire d'un ressort de raideur k et dont l'autre extrémité est reliée à une masse m par l'intermédiaire d'un ressort identique au premier. On admettra que les frottements sont négligeables. Si θ et x représentent respectivement la rotation de la poulie et le déplacement de la masse m par rapport à la position d'équilibre du à la pesanteur, on a les deux équations différentielles suivantes, que l'on obtient par exemple à partir des équations de Lagrange : où $F(t)$ représente la force instantanée que l'on applique à la masse en mouvement m .

Afin d'étudier ces deux équations différentielles du second ordre couplées on peut poser :

$x_1 = x, x_2 = \frac{dx}{dt}, x_3 = \theta, x_4 = \frac{d\theta}{dt}$ et l'on obtient le système différentiel :

3 Définir un système linéaire : enfin un peu d'automatique

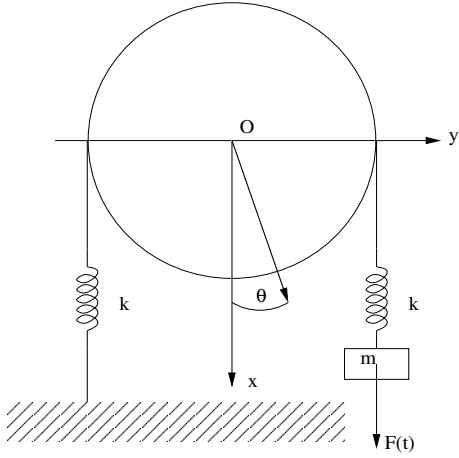


FIGURE 3.10 – Système différentiel

$$\frac{d}{dt} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{k}{m} & 0 & -\frac{kr}{m} & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{kr}{I} & 0 & -\frac{2kr^2}{I} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ -\frac{F(t)}{m} \\ 0 \\ 0 \end{bmatrix}$$

système que l'on pourra facilement intégrer à l'aide de l'outil `ode()`.

4 Représentation fréquentielle

4.1 Rappels sur la réponse fréquentielle : calcul de la réponse d'un système continu

Cette partie de l'exposé a pour but d'étudier la réponse permanente d'un système continu soumis à une entrée sinusoïdale de fréquence variable.

Attention : Scilab emploie comme argument la fréquence, alors que les automaticiens préfèrent généralement utiliser comme argument la pulsation.

Je rappelle que la réponse fréquentielle d'un système (vecteur de systèmes, car Scilab sait traiter les réponses fréquentielles d'un vecteur système), revient à calculer le nombre complexe $g(j\omega)$ transmittance isochrone, si $g(s)$ est le modèle transmittance isomorphe du système. Dans Scilab comme l'argument n'est pas la pulsation mais la fréquence, le nombre qui sera calculé est : $g(j2\pi f)$; f représente le vecteur (ou la matrice) fréquence. Ce nombre peut être représenté par la fraction :

$$g(j2\pi f) = \frac{N(j2\pi f)}{D(j2\pi f)}$$

où $N(s)$ et $D(s)$ sont respectivement le polynôme numérateur (degré m) et le polynôme dénominateur (degré n) de la transmittance isomorphe. Cette relation peut aussi s'écrire

$$g(j2\pi f) = \frac{N(j2\pi f)}{D(j2\pi f)} \frac{D(-j2\pi f)}{D(-j2\pi f)}$$

dans ces conditions on peut voir que cette quantité se transforme en sa conjuguée quand on change f en $-f$. Plus généralement comme $g(s)$ est un rapport de deux polynômes de la variable s on a : $g(s^*) = g^*(s)$. Ceci interviendra dans l'étude du lieu de Nyquist complet.

Nous allons dans les exercices qui vont suivre présenter les principales instructions permettant l'étude de la réponse fréquentielle d'un système ; dans le paragraphe suivant je ferai un petit rappel sur les systèmes à déphasage minimaux et non minimaux, ainsi qu'un rappel sur la relation de Bayard-Bode. La première instruction intéressante est l'instruction `freq()`¹.

```
-->x=poly(0,"x")
x =
x
-->fra=(x+1)/(x^3+x^2+x+2)
```

1. Cette instruction est une instruction mathématique et a déjà été vue dans un paragraphe précédent (section 2.4.4).

4 Représentation fréquentielle

```

fra =
      1 + x
-----
      2    3
2 + x + x + x
-->rep=freq(fra("num"),fra("den"),[1,2,3,5,10])
rep =
! 0.4 0.1875 0.0975610 0.0382166 0.0098921 !

```

Cette instruction `freq()` a pour but de calculer pour différentes valeurs de la variable (ici `x`), les valeurs de la fraction rationnelle `fra`. Faites attention cette commande est valable pour une fraction rationnelle ou un quadruplet (A,B,C,D) , modèle d'état d'un système et dans ce cas l'instruction calcule la valeur du scalaire $C \cdot \text{inv}(x(k) \cdot \text{eye} - A) \cdot B + D$ ou $x(k)$ est la k ème valeur du vecteur argument.

Un autre exemple, dans la même session Scilab, pour illustrer cette instruction :

```

-->rep=freq(fra("num"),fra("den"),[%i,2*%i,3,5,10*%i])
rep =
column 1 to 4
! 1. + i - 0.35 + 0.05i 0.0975610 0.0382166 !
column 5
! - 0.0101020 + 0.0000101i !

```

J'ai ici panaché, des arguments réels et complexes, on a donc un vecteur ligne des résultats qui sont réels et complexes.

```

-->rep=freq(fra("num"),fra("den"),[1:5])
rep =
! 0.4 0.1875 0.0975610 0.0581395 0.0382166 !

```

Vous remarquerez que cette instruction ressemble à l'instruction `horner()` mais utilise un vecteur comme donnée d'entrée (`horner()` aussi finalement). Cette instruction s'applique à une fraction rationnelle qui ne représente pas forcément un système linéaire et est semble t'il plus performante que l'instruction `horner()`.

Passons maintenant aux instructions spécifiques à l'étude des systèmes. Une instruction très importante pour l'étude de la réponse fréquentielle d'un système est l'instruction `repfreq()`.

```

-->A=diag([-1,-2]);B=[1;1];C=[1,1];
//On définit un modèle d'état pour le système.
-->Sys=syslin("c",A,B,C)
-->f=0:0.2:1;
-->[frq1,rep]=repfreq(Sys,f)
rep =
column 1 to 3
! 1.5 0.7462050 - 0.7124703i 0.3305398 - 0.5871213i !
column 4 to 5
! 0.1755529 - 0.4548199i 0.1064100 - 0.3631223i !
column 6
! 0.0707044 - 0.2997358i !
frq1 =

```

4 Représentation fréquentielle

```
! 0. 0.2 0.4 0.6 0.8 1. !
```

Nous voyons dans cet exemple que l'instruction `repfreq()` renvoie deux vecteurs lignes, un vecteur fréquence ici `frq1` et un vecteur complexe `rep` donnant les valeurs réelles et imaginaires du nombre complexe $Sys(j2\pi f)$. Le vecteur `frq1` est égal au vecteur `f` sauf pour des valeurs de fréquence donnant des discontinuités dans la (les) réponse(s). De même dans cette instruction on peut simplement donner les deux extrémités du vecteur fréquence, Scilab avec l'instruction `calfreq()` appelée par `repfreq()` ce chargeant de calculer d'une manière optimale le vecteur fréquence (on peut aussi afficher les discontinuités). Voir le manuel de `repfreq()` et `calfreq()`.

Profitons de cet exercice pour déterminer la fonction de transfert d'un système à partir des équations d'état (attention à la simplification de pôles par des zéros).

```
-->A=diag([-1,-2,-3]);B=[1;1;1];C=[1,0,0];
-->sys=syslin("c",A,B,C);
-->gp=ss2tf(sys)
//passage d'une représentation d'état à la fonction de transfert
gp =
    1
-----
1 + s
-->simp_mode(%F)
-->gp=ss2tf(sys)
gp =
           2
      6 + 5s + s
-----
           2    3
      6 + 11s + 6s + s
-->sys1=tf2ss(gp)
```

Cet exemple illustre le passage d'une représentation d'état à la fonction de transfert du système (instruction `ss2tf()`), par défaut la variable complexe choisie est pour un système continu s). Le système possède deux zéros : $s = -3$, $s = -2$ et trois pôles : $s = -1$, $s = -2$, $s = -3$ et Scilab par défaut simplifie l'expression sauf si on lui impose la non simplification : présence de l'instruction `simp_mode(%F)`. L'opération inverse, passage de la fonction de transfert aux équations d'état, se fait par l'instruction : `tf2ss(gp)`. (Vous remarquerez que par cette instruction on retrouve pour A un scalaire).

Revenons à la réponse fréquentielle d'un système.

De même on peut obtenir par l'instruction `dbphi()` à partir du vecteur complexe `rep` (voir avant dernier exercice), le module en décibels et l'argument en degrés de ce vecteur.

```
-->[db,phi]=dbphi(rep)
phi =
column 1 to 5
! 0. - 80.956939 - 85.440135 - 86.963211 - 87.721475 !
column 6
```

4 Représentation fréquentielle

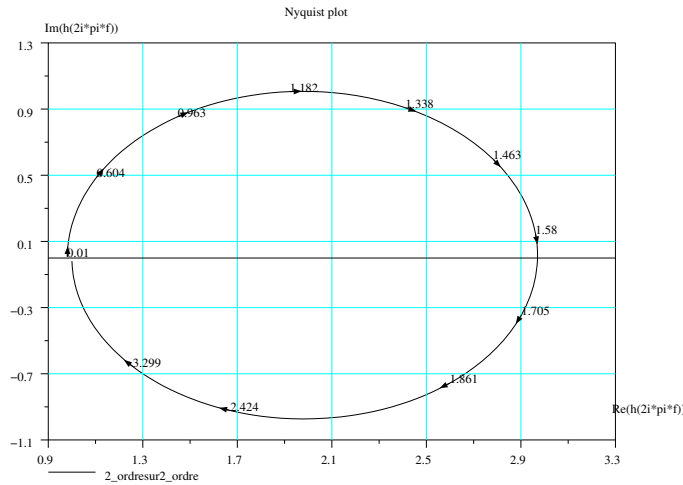


FIGURE 4.1 – Lieu de Nyquist

```
! - 88.176834 !
db =
column 1 to 5
! 0. - 16.072235 - 22.011613 - 25.518228 - 28.011667 !
column 6
! - 29.947396 !
```

Note au 20.03.2000 : les instructions précédentes (`dbphi()`, `phasemag()`) peuvent être avantageusement remplacées par une seule instruction nommée `dbphifr()`.

4.2 Les divers lieux

4.2.1 Représentation de Nyquist

La représentation de Nyquist (FIG 4.1) est une courbe donnant en abscisse la partie réelle de $g(j\omega)$ et en ordonnée la partie imaginaire de $g(j\omega)$. Attention Scilab n'utilise pas la variable ω mais la fréquence à savoir f pour graduer le lieu de Nyquist.

```
-->clf();//ou scf();
-->s=%s;
-->sys=syslin("c", (s^2+2*.9*10*s+10^2)/(s^2+2*.3*10.1*s+10.1^2));
-->nyquist(sys,.01,100,"2_ordresur2_ordre",%f)
//je ne veux pas de fréquences négatives
-->nyquist(sys,.01,100,"2_ordresur2_ordre")
```

Pour plus de renseignements sur le lieu de Nyquist faites appel au manuel : cette instruction ne pose aucun problème particulier.

4.2.2 Représentation de Bode

La représentation de Bode est constituée de deux courbes : la première appelée courbe de gain (celui-ci exprimée en db), est la représentation de $g_{db} = 20 \log |(g(j\omega))|$ comme fonction du $\log(\omega)$. La seconde courbe, est le graphe de la phase $\varphi(\omega)$, argument de $g(j\omega)$, comme fonction du $\log(\omega)$. Je rappelle que la phase $\varphi(\omega)$ (exprimée en degrés) représente le déphasage entre le signal de sortie sinusoïdal et le signal d'entrée lui même sinusoïdal.

Comme pour le lieu de Nyquist, Scilab n'utilise pas normalement l'argument pulsation, mais la fréquence pour le tracé du lieu de Bode (FIG 4.2) sauf si vous l'imposez par le drapeau "rad". Voici une session :

```
-->s=%s;
-->fr=(s^2+2*.9*10*s+10^2)/(s^2+2*.3*10.1*s+10.1^2);
-->sys=syslin("c",fr)
```

sys =

2
100 + 18s + s

2
102.01 + 6.06s + s

```
-->clf();//peut être inutile
```

```
-->bode(sys,.01,100,"2_ordresur_2_ordre")
```

Si l'on ne souhaite pas tracer les deux courbes, gain et phase, on peut utiliser l'instruction `gainplot()` qui ne donne que la courbe de gain du système (même chose pour la phase : on utilise l'instruction `phaseplot()`).

Vous remarquerez que le lieu de Bode ainsi dessiné correspond bien à un système (circuit) avance-retard de phase et que normalement, avec ce système, quand la fréquence augmente le déphasage commence par croître en étant positif.

Maintenant réalisez l'exercice suivant :

```
-->sl=syslin("c",1/(s*s+s*s*s))
```

```
-->bode(sl,.01,1)//ou black
```

```
//Charger Autoelem Toolbox et utiliser l'instruction bbode et comparer.
```

```
-->figure(1);bbode(sl,.01,1)//ou bblack
```

Il y a bien un problème sur la phase, car en utilisant le premier programme `bode` la réponse est calculée avec `repfreq` : dans la phase on ne tient pas bien compte de la phase introduite par le double intégrateur. Voir explication complète dans le répertoire `Autoelem/docs/programs.pdf`.

4.2.3 Représentation de Black

La représentation de Black permet de synthétiser sur un même graphe les courbes de gain et phase. En abscisse on définit le déphasage φ en degrés et en ordonnées le gain g en décibels du système (FIG 4.3).

```
-->sys=syslin("c",1/(s^2+s+1));
```

4 Représentation fréquentielle

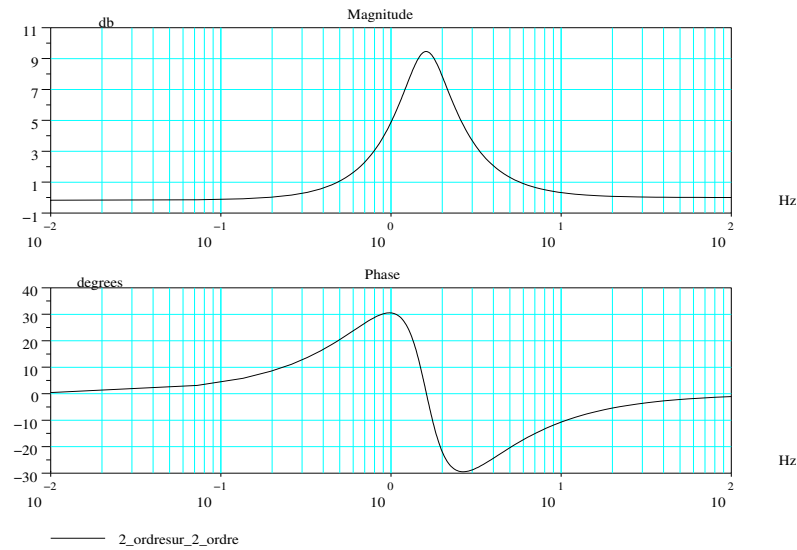


FIGURE 4.2 – Lieux de Bode

```
-->clf();//peut être inutile
-->black(sys,.01,100,"2_ordre")
```

Vous remarquerez, que la commande `black()` affiche en plus de la courbe demandée, le lieu iso-gain 2,3 db, valeur utile pour la synthèse d'un système.

4.2.4 L'abaque de Black

L'abaque de Black est constitué d'un ensemble de deux réseaux de courbes orthogonales, lieux des points en boucle fermée, où le gain est constant (valeur affichée sur une courbe de gain) et où la phase est constante.

```
-->s=%s;
-->fr=(2+3*s+s*s)/(1+3*s+2.5*s*s+s^3);
-->sl=syslin("c",fr)
```

```
sl =
      2
    2 + 3s + s
-----
      2   3
    1 + 3s + 2.5s + s
-->clf();
```


4 Représentation fréquentielle

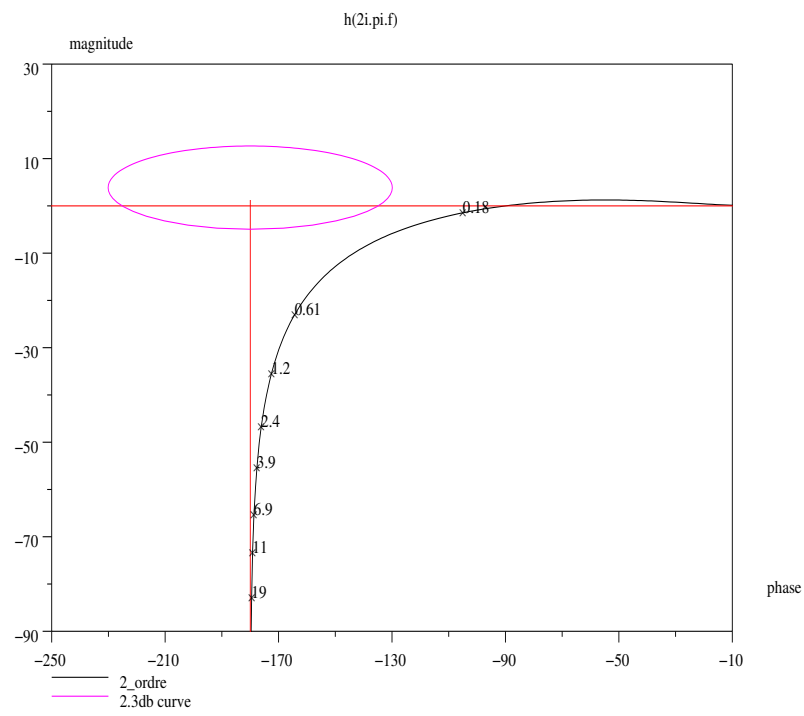


FIGURE 4.3 – Lieu de Black

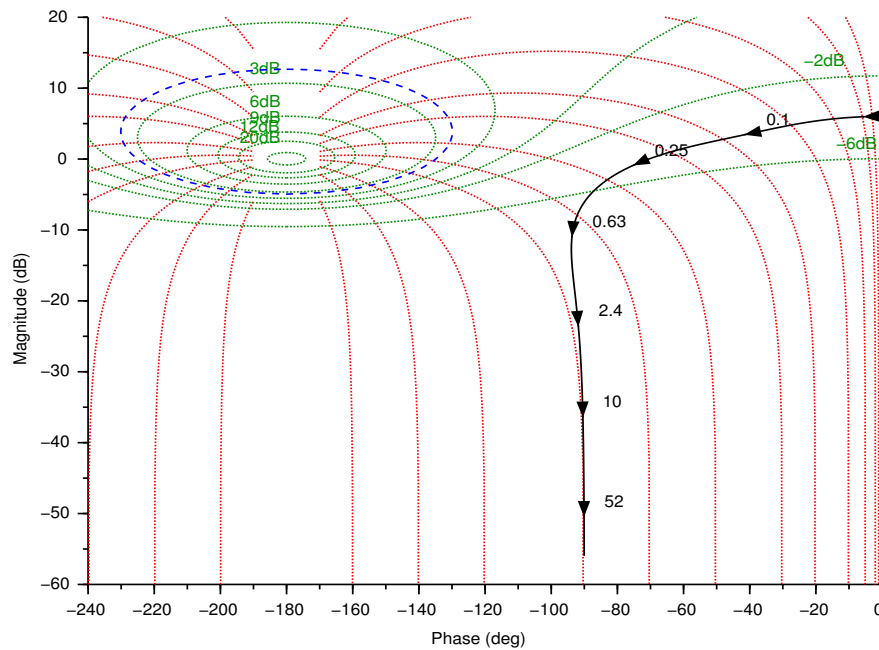


FIGURE 4.4 – Abaque de Black

```
-->black(sl,.01,100)
-->nicholschart([-6,-2,-.5,1,3,6,9,12,20],[-5,-50],[14,5])
```

La représentation graphique est donné à la figure qui suit (FIG 4.4).

La première instruction graphique, `clf()` va ouvrir une fenêtre graphique vide (ce qui n'est pas obligatoire, car l'instruction qui suit à savoir `black()` ouvre automatiquement une fenêtre), puis va demander à Scilab de tracer dans cette fenêtre le lieu de Black du système précédemment défini et enfin à placer sur ce lieu de Black l'abaque de Black. Attention sur la figure de ce document je n'ai reproduit qu'une partie du lieu, pour ce faire j'ai utilisé le bouton `zoom` de la fenêtre graphique.

Enfin même si j'anticipe un peu, voici quelques instructions qui peuvent être utiles pour déterminer quelques caractéristiques de systèmes bouclés : on étudiera cela plus profondément au chapitre concernant la synthèse des systèmes.

4.2.5 Quelques caractéristiques d'un système à partir des lieux

```
-->sl=syslin("c",fr/s)
```

4 Représentation fréquentielle

```

sl =
      2
    2 + 3s + s
-----
      2      3      4
s + 3s + 2.5s + s
-->[mg,f1]=g_margin(sl)
f1 =
    0.4090484
mg =
    14.271982
-->[mp,f2]=p_margin(sl)
f2 =
    0.1849030
mp =
    28.352224

```

On voit bien dans l'avant dernier exemple choisi, que la marge de gain `g_margin()` est infinie, ce qui est logique étant donné que la phase reste comprise entre 0 et $-\pi$; j'ai donc rajouté un intégrateur au système linéaire précédent en multipliant l'ancien système linéaire par le rationnel $\frac{1}{s}$ et obtenu, sans le redéfinir, un nouveau système linéaire continu. On voit donc apparaître la marge de gain `mg` et la fréquence correspondante `f2`.

Quant à l'instruction `p_margin()`, elle donne le vecteur phase du système à une pulsation (fréquence) pour laquelle le gain vaut 1 (fréquence de coupure à *0db*) : vous ferez attention que cette instruction peut vous renvoyer un vecteur à plusieurs composantes (s'il y a plusieurs fréquences à donner un gain de 1).

Enfin une dernière instruction `ffreson()` permet de déterminer la (ou les) fréquences pour lesquelles le gain est maximal (résonance). De même attention si vous avez une indétermination pour la fréquence nulle (au moins un pôle et un zéro non simplifié à l'origine) `ffreson()` vous renvoie une erreur. Je ne reviens pas sur cette instruction qui dans Scilab se nomme `freson.sci` : voir l'avant propos.

```

-->fres=ffreson(sl)
!--error 9999
infinite gain at zero frequency
at line 8 of function ffreson called by :
fres=ffreson(sl)

```

Le résultat obtenu est normal car le système possède un intégrateur donc a un gain infini à la fréquence zéro. Réalisons maintenant un système bouclé à retour unitaire avec `sl(s)` comme chaîne d'action :

```

-->sb=sl/(1+sl)//système bouclé
sb =

```

4 Représentation fréquentielle

```

          2
      2 + 3s + s
      -----
          2      3      4
      2 + 4s + 4s + 2.5s + s
-->fres=ffreson(sb)
fres =
    0.1958009
Le système bouclé est facilement réalisé n'est ce pas. Utilisons la nouvelle fonction
dbphifr() 2 afin de déterminer le facteur de résonance en boucle fermé (valeur D).
-->[FR,D,P]=dbphifr(sb,ffreson(sb))
P =
    - 88.931411
D =
    6.3927001
FR =
    0.1958009

```

4.2.6 Rappel de cours, diagrammes asymptotiques de Bode : systèmes à déphasage minimaux et non minimaux

Après de nombreuses années d'enseignement, je me suis aperçu que beaucoup d'étudiants n'avaient pas assimilé la relation de Bayard-Bode, relation liant la courbe de gain à la courbe de phase pour des systèmes à déphasage minimaux. Nous allons donc présenter les principaux résultats et définir la notion de système à déphasage minimal.

Systèmes à déphasage minimaux

On dira qu'un système est à déphasage minimal s'il ne possède ni pôle ni zéro dans le demi plan droit du plan complexe : système stable ne possédant pas de zéro dans le demi plan droit. Dans ces conditions il existe une relation liant la courbe de phase à la courbe de gain. Cette relation dite de Bayard-Bode est donnée par l'équation suivante :

$$\varphi(\omega) = \frac{2\omega}{\pi} \int_{-\infty}^{+\infty} \frac{\ln A(\alpha) - \ln A(\omega)}{\alpha^2 - \omega^2} d\alpha$$

Dans cette relation, φ représente le déphasage en fonction de la pulsation et A est le gain en fonction de cette même pulsation ($A = |g(j\omega)|$ ou $g(s)$ est la transmittance isomorphe du système considéré).

Si l'on construit un diagramme en ayant factorisé (factorisation de Bode) le système avec un terme constant (gain statique, en position, en vitesse ... suivant le cas), avec un terme dérivateur ou intégrateur pur muni du bon exposant, avec des termes du premier

2. Cette instruction retourne trois vecteurs, D le vecteur gain en db, P le vecteur phase en degrés et retourne le vecteur fréquence (qui est une entrée pour la fonction).

4 Représentation fréquentielle

et/ou second degré pour le numérateur et le dénominateur de la forme : $(1 + \tau_1 s)$ et / ou $(1 + \tau_2 s + \tau_3 s^2)$ (s étant la variable symbolique, avec $(\tau_2^2 - 4\tau_3) < 0$ et $\tau_3 > 0$), en ayant pour chacun de ces termes tracé des diagrammes réduits aux asymptotes (comportement aux basses et hautes fréquences des systèmes élémentaires), en ayant sommé algébriquement ces diagrammes, il existe alors pour les systèmes à déphasages minimaux, une relation simple liant la courbe de phase asymptotique à la courbe de gain asymptotique. Cette relation se résume en une phrase : à une pente de x fois 20 db par décade pour la courbe de gain correspond un déphasage de x fois $+90^\circ$.

Il est bien évident que le logiciel de simulation devra respecter cette relation, pas de saut intempestif, pas de décalages de 360° dans un sens ou un autre, étant donné que par convention en automatique on considère qu'un gain seul k , sans constante de temps, positif, est représenté en Bode par la droite $20\log(k)$ et par la droite 0° . Pour un gain négatif, la courbe de Bode est $20\log(|k|)$ et -180° , et non $+180^\circ$ (voir aussi le diagramme de Black et position du point -1 pris par convention à 0 db et -180°).

Systèmes à déphasage non minimaux, comment s'en sortir ?

D'abord il faut respecter la factorisation de Bode comme dans le cas précédent. On a donc des valeurs de τ_1 et/ou de τ_2 qui peuvent être négatives dans les expressions $(1 + \tau_1 s)$ et/ou $(1 + \tau_2 s + \tau_3 s^2)$. En multipliant le numérateur **et** le dénominateur de la transmittance par les termes $(1 - \tau_1 s)$ ou par des termes de la forme $(1 - \tau_2 s + \tau_3 s^2)$, ce qui ne change rien pour la réponse fréquentielle, on mettra ainsi en évidence dans la transmittance un déphaseur pur. Un exemple mathématique éclaire le raisonnement soit :

$$g(s) = \frac{1 - s + s^2}{s(1 + s)}$$

on peut écrire cette expression sous la forme :

$$g(s) = \frac{1 - s + s^2}{s(1 + s)} \frac{1 + s + s^2}{1 + s + s^2}$$

soit encore :

$$g(s) = \frac{1 - s + s^2}{1 + s + s^2} \frac{1 + s + s^2}{s(1 + s)}$$

nous voyons apparaître pour le premier terme de la transmittance un déphaseur pur ; quant au second terme, il est à déphasage minimal. Le raisonnement que j'ai tenu pour deux zéros situés dans le demi plan droit s'applique aussi à deux pôles, la seule différence concerne la stabilité du système, de même pour des termes du premier degré.

Remarque : je ne parlerais pas dans ce chapitre des systèmes à retard pur qui ne sont pas des systèmes à déphasage minimaux et dont le modèle mathématique n'est pas une fraction rationnelle : on verra cela plus tard.

4.3 Etude de systèmes simples

Comme lors de la synthèse d'un système bouclé on compare souvent ce système bouclé à un système du premier ou du second ordre, il est donc intéressant maintenant

4 Représentation fréquentielle

de donner les réponses temporelles et fréquentielles de ces systèmes élémentaires.

4.3.1 Le premier ordre

Le pôle réel négatif (stable)

Un seul pôle caractérise un système du premier ordre il est noté : p_1 . La transmittance isochrone a pour équation :

$$g(s) = \frac{k_b}{1 + \tau s} = \frac{k_e}{s - p_1} \text{ avec } \tau = -\frac{1}{p_1} \text{ et } k_b = -\frac{k_e}{p_1} \geq 0$$

Je donne ici les deux formulations celle de Bode et celle d'Evans : τ est la constante de temps (supposée positive, système stable), p_1 le pôle, k_b le gain statique.

Les réponses impulsionnelle, indicielle

La condition initiale étant nulle, l'instant initial aussi, les réponses impulsionnelle et indicielle valent respectivement :

$$h(t) = \frac{k_b}{\tau} \exp\left(-\frac{t}{\tau}\right) \text{ et } y_1(t) = k_b(1 - \exp\left(-\frac{t}{\tau}\right))$$

Le gain statique k_b est la réponse indicielle permanente et pour $t = \tau$ la réponse $y_1(t)$ vaut 63% de la réponse finale. Enfin pour $t = 3\tau$ cette même réponse vaut 95% de la réponse permanente (temps de réponse à 5%). Vous pourrez avec Scilab tracer, en prenant $\tau = 1$, ce qui revient à prendre pour abscisse non pas t mais $\frac{t}{\tau}$, les réponses correspondantes (FIG 4.5).

La réponse fréquentielle

Quant à la réponse fréquentielle (le lieu de Nyquist est un demi cercle), on a pour le gain en db et la phase en degrés :

$$\begin{aligned} g_{db} &= 20 \log(k_b) - 10 \log(1 + \tau^2 \omega^2) \\ \varphi &= -\arctan(\tau \omega) \end{aligned}$$

Pour la pulsation de cassure $\omega = \frac{1}{\tau}$ l'atténuation par rapport au gain statique est de 3db et le déphasage correspondant est de -45° . Si on cherche le diagramme asymptotique de Bode, il est constitué pour la courbe de gain des deux droites $20 \log(k_b)$ et $20 \log(k_b) - 20 \log \tau - 20 \log \omega$ qui se coupent au point $[\omega = \frac{1}{\tau}, 20 \log(k_b)]$, pour la courbe asymptotique de phase on a les deux horizontales 0° et -90° .

4 Représentation fréquentielle

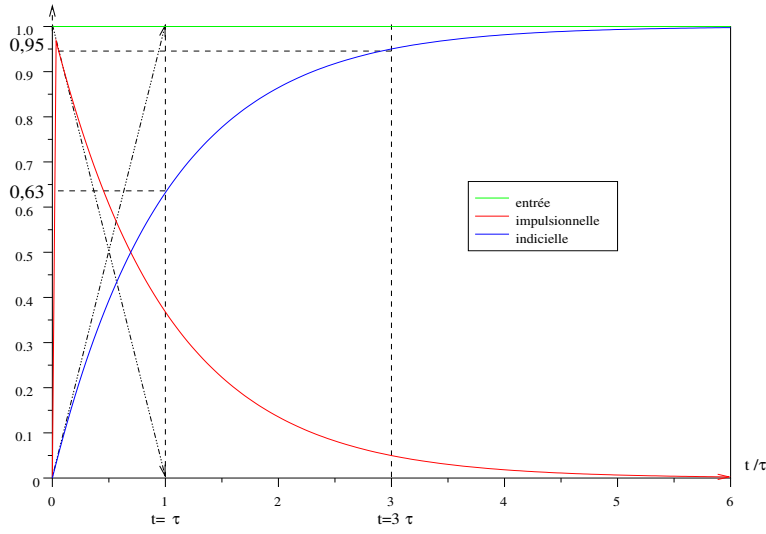


FIGURE 4.5 – Réponses d'un premier ordre

4.3.2 Le second ordre

Les pôles stables

On a l'habitude de mettre un système du second ordre sous la forme canonique suivante :

$$g(s) = \frac{k_b}{1 + 2\xi \frac{s}{\omega_n} + \frac{s^2}{\omega_n^2}}$$

Avec comme paramètres du système : k_b le gain statique, ω_n la pulsation naturelle et ξ le coefficient d'amortissement. Suivant la valeur de ξ on peut avoir des pôles réels ($\xi \geq 1$) qui valent : $p_{1,2} = -\xi\omega_n \pm \omega_n\sqrt{\xi^2 - 1}$ ou des pôles complexes conjugués ($\xi \leq 1$) qui dans ce cas valent : $p_{1,2} = -\xi\omega_n \pm j\omega_n\sqrt{1 - \xi^2}$ (FIG 4.6). Dans le cas où ces deux pôles sont réels ($\xi \geq 1$), la transmittance est le produit de deux transmittances du premier ordre et on ce ramène au cas précédent.

Les réponses impulsionnelle, indicielle

Les condition initiales étant nulles, l'instant initial aussi, les réponses impulsionnelle et indicielle valent respectivement (FIG 4.7, FIG 4.8) :

$$h(t) = k_b \frac{\omega_n^2}{\omega_p} \exp(-\xi\omega_n t) \sin(\omega_p t) \text{ avec } \omega_p = \omega_n \sqrt{1 - \xi^2}$$

4 Représentation fréquentielle

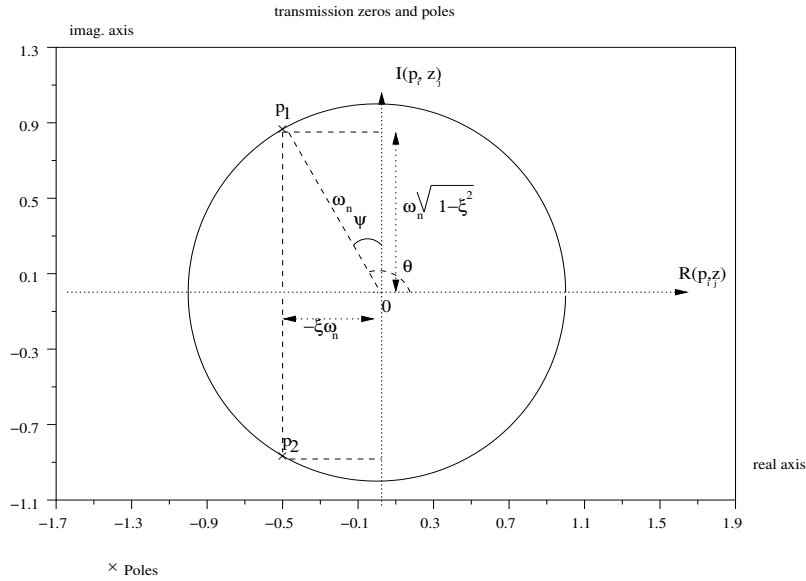


FIGURE 4.6 – Pôles second ordre

$$y_1(t) = k_b \left[1 - \frac{\omega_n}{\omega_p} \exp(-\xi \omega_n t) \sin(\omega_p t + \theta) \right] \text{ avec } \theta = \arccos(\xi)$$

On appelle la pulsation ω_p la pulsation propre qui vaut la valeur absolue de la partie imaginaire des pôles, $\omega_p = |I(p_{1,2})|$. Quant au facteur de l'exponentielle, c'est la partie réelle des pôles $-\xi \omega_n = R(p_{1,2})$: ces formules sont valables pour $\xi \leq 1$.

Si $\xi \geq 1$, on peut remplacer dans ces formules le $\sin()$ par $\sinh()$ et $\omega_n \sqrt{1 - \xi^2}$ par $\omega_n \sqrt{\xi^2 - 1}$.

Voici le programme donnant les pôles, et les réponses impulsionnelle et indicielle d'un ensemble de systèmes du second ordre pour $\xi = [0, 2 \quad 0, 3 \quad 0, 4 \quad 0, 6 \quad 0, 8]$. De même on résume dans un tableau les principales propriétés concernant la réponse indicielle d'un second ordre oscillatoire.

```
//pôles et zéros
s=%s;g=syslin("c",1/(s*s+s+1));plzr(g)
//création des second ordre
den=(s*s+1)*ones(5,1)+2*s* [.2;.3;.4;.6;.8];
fr=poly(1,"s","c")*ones(5,1)./den;
g=syslin("c",fr);
t=linspace(0,12,121);h=csim("imp",t,g);
figure(0);
plot2d(t',h',style=[2;3;4;5;6],axesflag=5)
legends(["z=.2";"z=.3";"z=.4";"z=.6";"z=.8"],[2,3,4,5,6])
y1=csim("step",t,g);figure(1);
```


4 Représentation fréquentielle

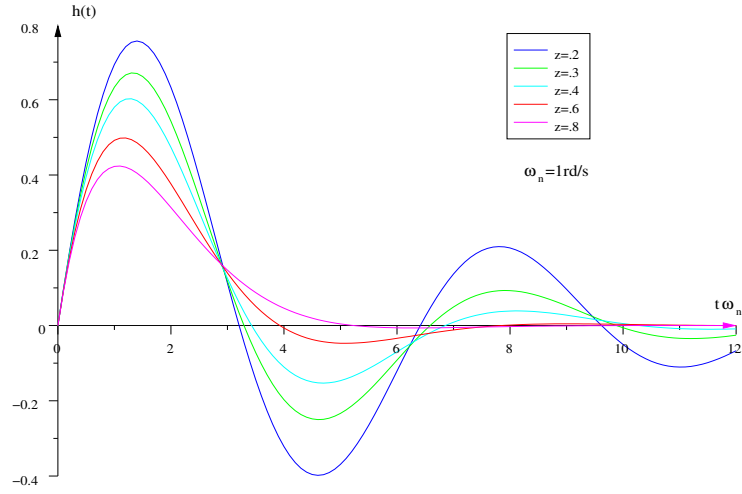


FIGURE 4.7 – Réponse impulsionnelle

```
plot2d(t',y1',style=[2;3;4;5;6],axesflag=5)
legends(["z=.2";"z=.3";"z=.4";"z=.6";"z=.8"],[2,3,4,5,6])
Nous donnons la réponse indicielle d'un seul système pour  $\xi = 0, 2$ , (FIG 4.8).
figure(3);
plot2d(t',y1(1,:)',style=2,axesflag=5)
legends("zeta=.2",2)
```

Enfin voici un tableau donnant les principaux résultats issus de la réponse indicielle :

Caractéristiques	Valeurs
Temps de montée t_m	$t_m = \frac{1}{\omega_n \sqrt{1-\xi^2}} (\pi - \arccos \xi)$
Temps de pic t_{pic}	$t_{pic} = \frac{\pi}{\omega_n \sqrt{1-\xi^2}}$
Pseudo-période, période propre T_p	$T_p = \frac{2\pi}{\omega_n \sqrt{1-\xi^2}}$
Pseudo-pulsation, pulsation propre ω_p	$\omega_p = \omega_n \sqrt{1-\xi^2}$
Dépassement $D\%$	$D\% = 100 \exp(-\pi \frac{\xi}{\sqrt{1-\xi^2}})$
Rapport de 2 maxima successifs	$\frac{D_1}{D_2} = \exp(2\pi \frac{\xi}{\sqrt{1-\xi^2}})$
Temps de réponse à $n\%$, souvent $n=\pm 5$	$t_{r\,n\%} \approx \frac{1}{\omega_n \xi} \ln(\frac{100}{n})$
Temps de réponse à 5%	$t_{r\,5\%} \approx \frac{3}{\omega_n \xi}$
Nombre d'oscillations complètes	$n \approx Q = \frac{1}{2\xi}$

4 Représentation fréquentielle

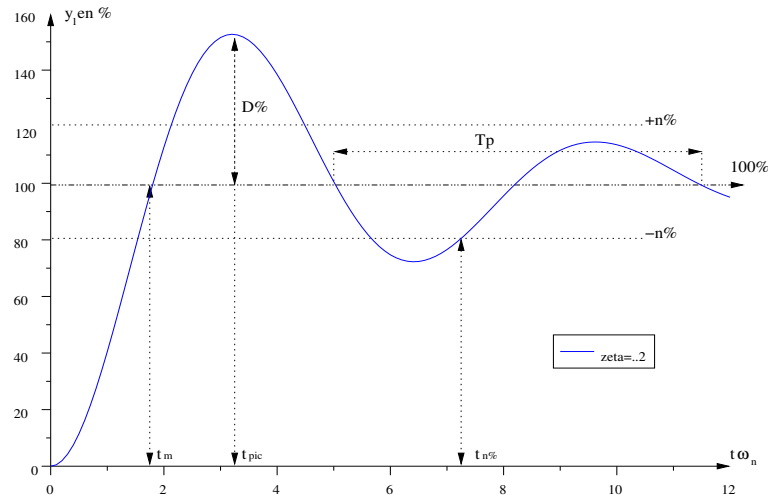


FIGURE 4.8 – Réponse indicielle

La réponse fréquentielle

Nous reprenons l'exercice précédent et traçons les lieux de Bode (complet), de Bode (gain) (FIG 4.9) et de Bode (FIG 4.10) par les instructions :

```
-->com=[["z=.2";"z=.3";"z=.4";"z=.6";"z=.8"]];
-->bode(g,.01,1,com)
-->figure(1);
-->gainplot(g,.01,1,com)
-->figure(2);
-->black(g,.01,1,com)
```

Comme nous l'avons fait pour la réponse indicielle nous allons donner sous forme de tableau, les principaux résultats relatifs aux réponses fréquentielles.

Caractéristiques	Valeurs
Pulsation de résonance	$\omega_r = \omega_n \sqrt{1 - 2\xi^2}$
Pulsation de coupure à 0db	$\omega_{co} = \omega_r \sqrt{2}$
Facteur de résonance	$Q = \frac{1}{2\xi\sqrt{1-\xi^2}}$
Pulsation de coupure à -6db	$\omega_{-6} =$
Relation temps-fréquence	$\omega_{co} t_r 5\% \approx \pi$

```
-->bode(g,.01,1,com)
-->figure(1);
```

4 Représentation fréquentielle

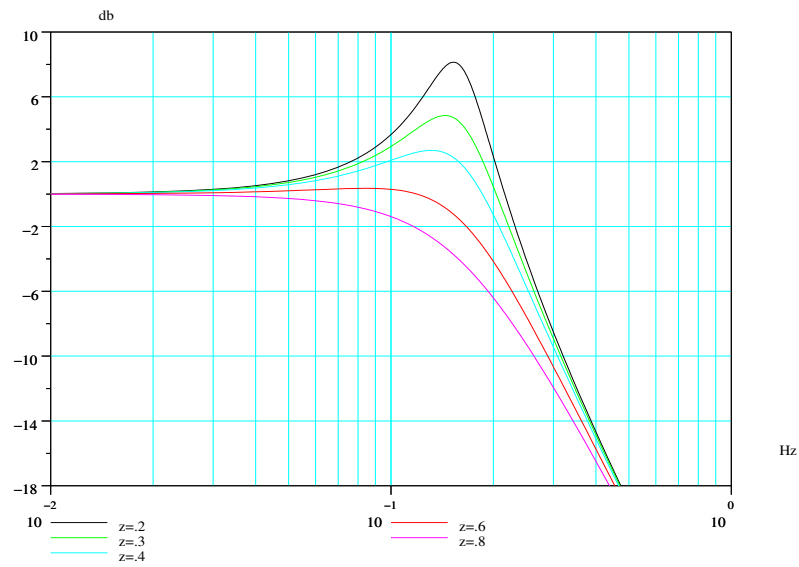


FIGURE 4.9 – Lieux de Bode (gain), second ordre

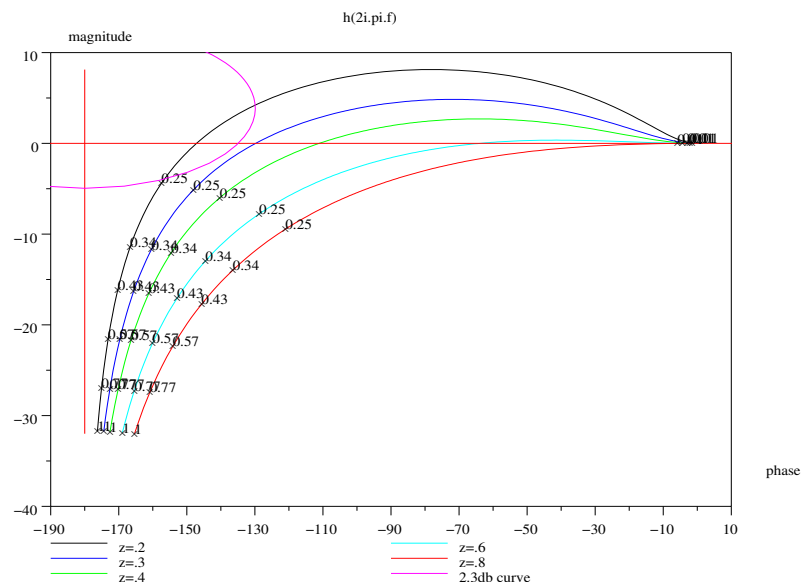


FIGURE 4.10 – Lieu de Black, second ordre

4 Représentation fréquentielle

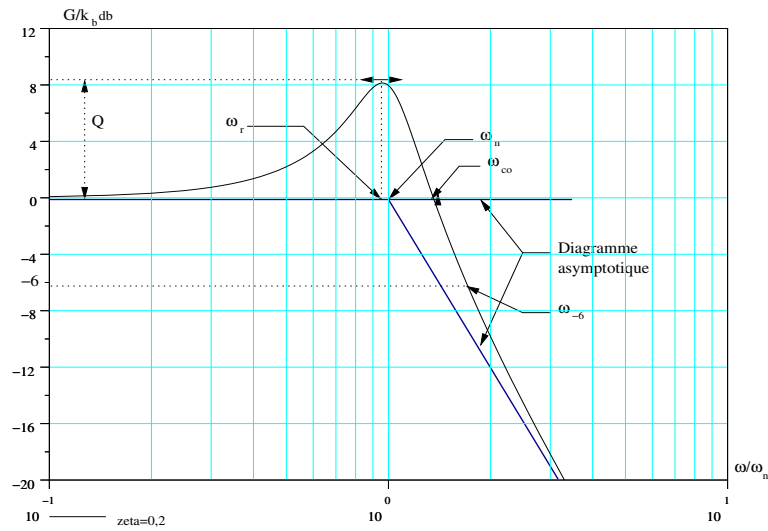


FIGURE 4.11 – Gain d'un second ordre

```
-->gainplot(g,.01,1,com)
-->figure(2);
-->black(g,.01,1,com)
```

En posant $u = \frac{\omega}{\omega_n}$ on peut exprimer pour un système du second ordre dont le gain statique est égal à 1, dont la pulsation naturelle vaut ω_n et le coefficient d'amortissement ξ , le gain complexe

$$g(ju) = \frac{1}{1 - u^2 + 2j\xi u}$$

ce qui donne pour le gain et la phase

$$|g(ju)| = \frac{1}{\sqrt{(1 - u^2)^2 + 4\xi^2 u^2}} \text{ et } \varphi = -\arctan(2\xi \frac{u}{1 - u^2})$$

ou encore pour le gain en décibels

$$g_{db} = -10 \log[(1 - u^2)^2 + 4\xi^2 u^2]$$

Sur la figure précédente (FIG 4.11) j'ai mis en abscisse le logarithme décimal de $u = \frac{\omega}{\omega_n}$ et non la fréquence.

5 Etude de la stabilité d'un système, marges de stabilité

5.1 Etude de la stabilité

5.1.1 Rappels sur la stabilité des systèmes

Pour un système à modèle linéaire, la stabilité est la propriété selon laquelle ce système écarté de sa position d'équilibre par une sollicitation extérieure (entrée ou perturbation), revient à cette position d'équilibre quand la sollicitation a cessé. L'étude de la stabilité se fait donc sur la base du régime transitoire ou du régime libre.

Pour tout système bouclé ou non, la **stabilité est une condition impérative**, c'est l'existence même du système qui est en jeu. Comme je le fais en cours à partir du théorème de convolution, on introduit la E.B.S.B stabilité : B.I.B.O stability pour les anglosaxons, entrée bornée, sortie bornée. La conséquence sur un système à modèle linéaire est la suivante : **Un système sera dit stable au sens strict s'il ne possède pas de pôle dans le demi plan droit du plan complexe**. Ceci revient à dire que pour un modèle d'état, la matrice d'état, notée généralement A aura **toutes ses valeurs propres dans ce même demi plan**. Ceci revient à dire aussi que sa réponse impulsionnelle sera décroissante en fonction du temps. Quant aux systèmes qui possèdent des pôles simples en nombre fini sur l'axe imaginaire pur, ou un pôle unique à l'origine, on dira que ces systèmes sont stables au sens large (déplacement d'un ou plusieurs pôles du demi plan gauche au demi plan droit par variation d'un paramètre du modèle).

5.1.2 Calcul des pôles d'un système

Si le modèle a tous ses paramètres constants et connus, la stabilité d'un système revient à chercher les valeurs des racines d'une équation de degré n . Ceci revient aussi (avec un choix de modèle d'état), à rechercher les valeurs propres de la matrice d'état A . Numériquement c'est exactement le même problème et cela peut être fait avec Scilab par les instructions `roots()` ou `spec()` avec les problèmes de précision que cela comporte dans le cas de pôles multiples.

5.1.3 Critère de Routh-Hurwitz

Historiquement ce problème de stabilité a été résolu par essentiellement deux mathématiciens (après 1850) Hurwitz (1895) et Routh (1877). Ces deux mathématiciens ont donné des critères très semblables, critères qui mettent en oeuvre les coefficients

d'un équation de degré n . A ce sujet, ces travaux viennent de l'étude de la régulation de la vitesse des machines à vapeur par l'utilisation du régulateur de Watt (1788), on lira l'article http://fr.wikipedia.org/wiki/James_Watt

Définition du critère de Routh-Hurwitz

C'est un critère algébrique permettant de calculer le nombre de racines à parties réelles positives (modèle instable) du polynôme caractéristique du modèle entrée-sortie du système. Ce polynôme noté $D(s)$ est de degré n .

$$D(s) = a_n s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0$$

Deux conditions doivent être tout d'abord vérifiées :

1. Tous les coefficients de $D(s)$ doivent exister.
2. Tous les coefficients de $D(s)$ doivent être de même signe.
3. Si ces deux conditions sont vérifiées, alors on construit la table de Routh :

ligne 1	$a_{1,1} = a_n$	$a_{1,2} = a_{n-2}$	$a_{1,3} = a_{n-4}$	$a_{1,4} = a_{n-6}$
ligne 2	$a_{2,1} = a_{n-1}$	$a_{2,2} = a_{n-3}$	$a_{2,3} = a_{n-5}$	$a_{2,4} = a_{n-7}$
ligne 3	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$	$a_{3,4}$
ligne 4	$a_{4,1}$	$a_{4,2}$	$a_{4,3}$	$a_{4,4}$
ligne 5

Les valeurs de la première colonne du tableau sont appelés les pivots, c'est le coefficient $a_{i,1}$ et le coefficient $a_{i,j}$ vaut :

$$a_{i,j} = \frac{a_{i-1,1}a_{i-2,j+1} - a_{i-2,1}a_{i-1,j+1}}{a_{i-1,1}}$$

La table de Routh associée au polynôme $D(s)$ d'ordre n comporte $n+1$ lignes, et la condition de stabilité s'énonce : **il y a autant de pôles instables pour le système que de changements de signe dans la première colonne de la table de Routh.**

Dans le cas particulier où un pivot est nul ou le tableau a une ligne de zéros, alors,

1. Si dans le calcul des éléments du tableau de Routh l'un des pivots est nul, les autres coefficients de la ligne étant tous non nuls, alors il existe au moins une racine de $D(s)$ dans le demi plan droit.
2. Si tous les coefficients de la ligne correspondante sont nuls, il existe un ou des couples de racines à partie réelle nulle, (le système est théoriquement un oscillateur oscillant à une pulsation dont la valeur est la valeur absolue de la partie imaginaire de ces deux racines).

Malgré cela on peut continuer à construire le tableau en remplaçant ce zéro par un nombre ε très petit quelconque (sauf dans le cas où tous les termes sont nuls), et l'on fera tendre ce petit nombre vers zéro.

Remarque : On peut facilement dans le particulier où une ligne est nulle déterminer les racines imaginaires pures du polynôme caractéristique : en prenant la ligne qui la

5 Etude de la stabilité d'un système, marges de stabilité

précède, on forme un polynôme en α^2 dont les coefficients ordonnés dans le sens des puissances décroissantes sont les coefficients de cette ligne et on résoud cette équation.

L'étude algébrique de la stabilité des systèmes peut être faite par Scilab en utilisant les deux instructions `routh_t()` et `kpure()`, voici deux exemples :

```
-->s=%s;den=poly([-1,6,-4,3,1+%i,1-%i],"s")
den =
          2      3      4      5      6
144 - 36s - 82s + 92s - 13s - 6s + s
-->routh_t(den)
ans =
!  1.      -13      -82.      144.  !
! -6.       92.      -36.       0.  !
! 2.3333333 -88.      144.       0.  !
! -134.28571 334.28571  0.       0.  !
! -82.191489 144.      0.       0.  !
! 99.016309.  0.       0.       0.  !
! 144.       0.       0.       0.  !
```

Dans ce cas, vous travaillez avec un polynôme sensé être le polynôme caractéristique d'un système linéaire. Dans cet exercice on a quatre changements de signe dans la première colonne du tableau de Routh, on a quatre racines à parties réelles positives : $[6, 3, 1+j, 1-j]$.

```
-->sl=syslin("c",.5/den);table=routh_t(denom(sl));
```

Dans ce contexte la table de Routh ne se justifie pas, en effet, comme on connaît le dénominateur de la fonction de transfert du système, on peut donc par l'instruction `roots()` trouver directement les racines de ce polynôme.

Nous allons faire deux exercices qui permettent de calculer les racines imaginaires pures dans le cas où le système est à la limite de la stabilité.

```
-->den=s^4+4*s^3+7*s^2+16*s+12
den =
          2      3      4
12 + 16s + 7s + 4s + s
-->tab=routh_t(den)
tab =
!  1.       7.      12.  !
!  4.      16.       0.  !
!  3.      12.       0.  !
! 8.882E-16 0.       0.  !
! 12.       0.       0.  !
-->tab=clean(tab)
tab =
!  1.       7.      12.  !
!  4.      16.       0.  !
!  3.      12.       0.  !
!  0.       0.       0.  !
! 12.       0.       0.  !
```

5 Etude de la stabilité d'un système, marges de stabilité

La ligne quatre possède un pivot très petit, les autres coefficients de cette ligne sont nuls. Scilab a remplacé ce pivot par un nombre très petit et continue le calcul : par l'instruction `clean()` on obtient la vraie table. On peut maintenant construire le polynôme donnant les racines imaginaires (il suffit de former le polynôme en α^2 dont les coefficients dans le sens des puissances décroissantes sont les termes de la ligne au dessus du pivot nul) ; ce polynôme vaut :

$$P(\alpha^2) = 3\alpha^2 + 12$$

soit en résolvant $P(\alpha^2) = 0$ on a : $\alpha = \pm 2j$. Le système oscille à la pulsation de $\omega = 2rd/s$. De plus dans le tableau de Routh il n'y a pas de changement de signe : les autres pôles sont stables. Vérifions ceci.

```
-->poles=clean(roots(den))
```

```
poles =
```

```
! - 1. !
```

```
! 2.i !
```

```
! -2.i !
```

```
! -3. !
```

Le deuxième exercice est plus un cas d'école car il s'applique à un système qui est de toute façon instable, tous les coefficients ne sont pas de même signe.

```
-->s=%s;
```

```
-->den=s^6-5*s^5+11*s^4-25*s^3+34*s^2-20*s+24
```

```
den =
```

```
          2      3      4      5      6
24 - 20s + 34s - 25s + 11s - 5s + s
```

```
-->table=routh_t(den)
```

```
table =
```

```
! 1.      11.      34.  24. !
```

```
! -5.     -25.     -20.  0. !
```

```
! 6.      30.      24.  0. !
```

```
! 1.110E-15 8.882E-16 0.  0. !
```

```
! 25.2     24.      0.  0. !
```

```
! -1.692E-16 0.      0.  0. !
```

```
! 24.      0.      0.  0. !
```

```
-->tab=clean(table)
```

```
tab =
```

```
! 1.      11.      34.  24. !
```

```
! -5.     -25.     -20.  0. !
```

```
! 6.      30.      24.  0. !
```

```
! 0.      0.      0.  0. !
```

```
! 25.2     24.      0.  0. !
```

```
! 0.      0.      0.  0. !
```

```
! 24.      0.      0.  0. !
```

Vérifions ces résultats en calculant les racines de ce polynôme.

```
-->poles=clean(roots(den))
```



```
poles =
!      i !
!     -i !
!    2.i !
!   -2.i !
!    2.  !
!    3.  !
```

Il y a deux changements de signe dans la première colonne du tableau de Routh, donc deux pôles instables. De plus ce polynôme possède des racines imaginaires pures vérifiant l'équation, ligne trois du tableau :

$$P(\alpha^2) = 6\alpha^4 + 30\alpha^2 + 24 = 0$$

soit $\alpha_1 = 2j$, $\alpha_2 = -2j$, $\alpha_3 = j$, $\alpha_4 = -j$.

Nouveau programme routh_t.sci

Depuis les versions quatre ou cinq de Scilab un nouveau programme `routh_t.sci` est proposé.

Il permet entre autre de sortir le nombre de changements de signe de la première colonne du tableau de Routh.

De même la présentation de la table change quand un zéro est détecté dans la première colonne du tableau. Enfin quand une ligne complète du tableau est nulle (voir exemple précédent), en dérivant la ligne précédente on peut continuer le tableau :

```
-->den=s^4+4*s^3+7*s^2+16*s+12;[tab,change]=routh_t(den)
change =
    0. //nombre de changements de signe.
tab =
    1.   7.  12.
    4.  16.   0.
    3.  12.   0.
    6.   0.   0. //ici le tableau change
//normalement cette ligne est constituée de zéros.
   12.   0.   0.
```

Critère de Routh-Hurwitz et les systèmes bouclés

Dans le cas d'un système bouclé, c'est autre chose, car dans le cas d'un bouclage avec un gain k , l'instruction `routh_t(s1,k)` donne la table formelle, voici un exemple :

```
-->s=%s;num=poly([5,1],"s","c");den=poly([0,-2,-3,-3],"s");
s1=syslin("c",num/den)
s1 =
      5 + s
-----
      2    3    4
18s + 21s + 8s + s
```

5 Etude de la stabilité d'un système, marges de stabilité

```
-->k=poly(0,"k");//on definit la variable k.
-->table=routh_t(sl,k,%f)//non normalisée : sans diviser par le pivot.
table =
! 1                21        5k !
! 8                18 + k    0  !
! 150 - k          40k       0  !
!                2          !
! 2700 - 188k - k   0        0  !
!                2      3    !
! 108000k - 7520k - 40k 0      0  !

Au cas où le troisième argument de la fonction n'est pas présent ou vaut « %t » on
a la table suivante :
-->table=routh_t(sl,k,%t)//ou -->table=routh_t(sl,k)
table =
1                21        5k
-                ---        ---
1                1         1
8                18 + k    0
-                -
1                1         1
150 - k          40k       0
-----        ---        -
8                8         1
                2
21600 - 1504k - 8k 0        0
-----        -        -
1200 - 8k          1         1
5k                0         0
---              -        -
1                1         1
-->[KL,PL]=kpure(sl)
PL =
! 1.9813434i !// Il y a deux pôles complexes conjugués.
KL =
13.405773
```

Dans cet exemple nous définissons un système bouclé à retour unitaire, dont la chaîne d'action est constituée de la mise en cascade d'un amplificateur de gain k positif et du système linéaire de transmittance sl , Scilab construit la table de Routh formelle (table dépendant du gain k).

De même l'instruction `kpure()` donne s'ils existent, les valeurs limites de k ici `KL`, et les valeurs des pôles (imaginaires purs conjugués : seul le pôle à partie réelle positive est renvoyé, c'est la pulsation de l'oscillation) `PL` du système bouclé pour les valeurs limites de k .

Vous verrez à titre de complément, par l'instruction `evans()`, le tracé du lieu des pôles du système bouclé. Dans l'instruction `evans()` on donne comme argument d'entrée la transmittance de la boucle ouverte (sans le gain), Scilab se chargeant de

construire le système bouclé, avec un gain k dans la chaîne d'action, on reviendra en temps utile sur cette instruction.

5.1.4 La stabilité d'un système bouclé par le critère de Nyquist-Cauchy

Le but de ce critère est de donner la **stabilité d'un système bouclé** à partir de la **connaissance de la boucle ouverte** seulement : c'est un **critère géométrique**.

Rappel du critère de Nyquist-Cauchy

Avant de rappeler le lemme de Cauchy qui permet de démontrer le critère de Nyquist-Cauchy, je voudrais définir dans quelles conditions ce critère peut s'appliquer.

Tracé de fonctions complexes d'une variable complexe

Un exemple évident de ce tracé est le lieu de Nyquist d'un système. En effet le modèle d'un procédé est souvent un rapport de deux polynômes de la variable complexe s : pour tracer le lieu de Nyquist on remplace cette variable par le complexe pur $j\omega$ et on trace une courbe en coordonnées paramétriques $\Re(g(j\omega)) = \text{fonct1}(\omega)$, $\Im(g(j\omega)) = \text{fonct2}(\omega)$ et ceci pour $\omega \in [0 \rightarrow +\infty]$. Il est évident que l'on peut très bien à l'aide du logiciel Scilab faire le tracé d'une courbe en paramétrique : voici un exemple de « pseudo-lieu » de Nyquist (FIG. 5.1). On se donne un système de transmittance :

$$g(s) = \frac{1 + 10s}{1 + s}$$

et on désire tracer les transformés des points du plan complexe situés sur la deuxième bissectrice, ces points ont pour affixes $w = re^{\frac{3j\pi}{4}}$ avec $r \in [0 \rightarrow 5]$. Le lieu de Nyquist étant les transformés des points ayant pour affixes $w = re^{\frac{j\pi}{2}}$ avec $r \in [0 \rightarrow +\infty]$.

```
-->s=%s;g=syslin("c",(1+10*s)/(1+s));//La fonction
-->r=linspace(0,5,501);//La valeur du rayon vecteur
-->w=r*exp(i*3*pi/4);//La droite
-->z=horner(g,w);
//ou mieux z=freq(g("num"),g("den"),w);
-->x=real(z);y=imag(z);
-->plot2d(x(:)',y(:)',style=5,axesflag=3)
-->xgrid(4)
Ou en réalisant le programme mettant en jeu les « pseudo-lieux ».
-->s=%s;
-->g=syslin("c",(1+10*s)/(1+s));
-->nnyquist([g;g],.001,10,["135°";"90°"],["pseudonyquist";"nyquist"])
//syntaxe : nnyquist(sys,fmin,fmax,angle,comments)
```

Ainsi en choisissant un vecteur **angle** de même dimension que le vecteur système situé ici en position 4 dans les arguments d'entrée, on peut directement tracer les lieux et pseudo-lieux. On verra l'utilisation lors de la synthèse des systèmes.

5 Etude de la stabilité d'un système, marges de stabilité

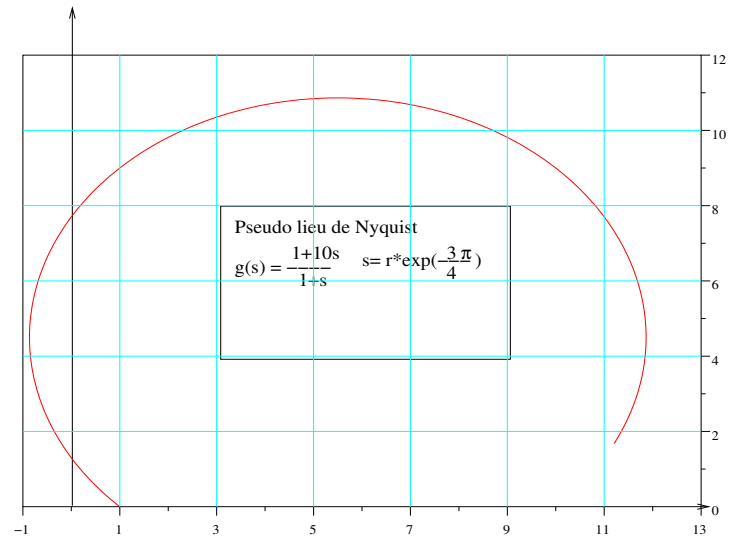


FIGURE 5.1 – Pseudo lieu de Nyquist

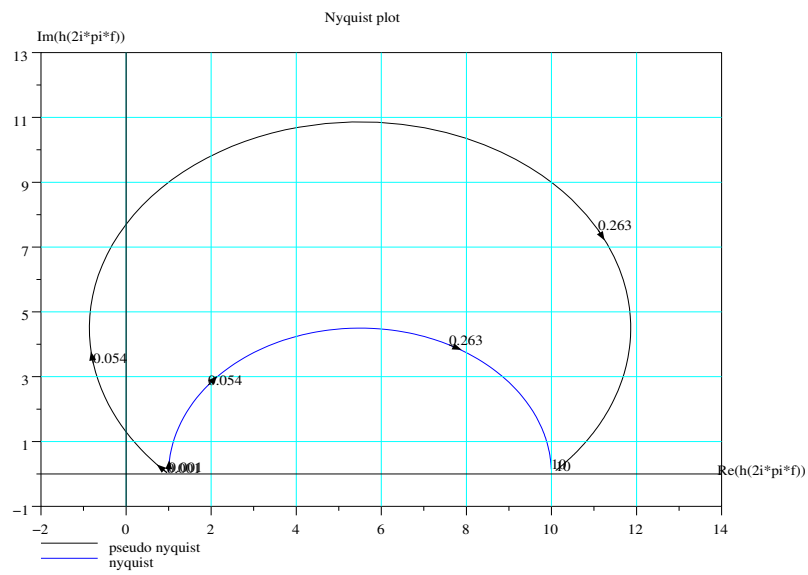


FIGURE 5.2 – Nyquist, pseudo Nyquist

Fonctions analytiques, contours, transformations conformes.

En mathématique, on dit qu'une fonction complexe $F(s)$ est analytique dans un domaine, si en tout point P d'affixe s_0 de ce domaine du plan complexe (s), la dérivée $\frac{dF}{ds}|_{s=s_0} = \lim_{s \rightarrow s_0} \frac{F(s) - F(s_0)}{s - s_0}$ existe et est unique.

Les fonctions de transfert des systèmes physiques considérés ici sont analytiques dans le plan complexe, sauf aux pôles de ces fonctions de transfert. De même on admettra comme connu dans le plan complexe la notion de contours fermés sans point double.

De même on doit, enfin d'utiliser correctement le lemme de Cauchy qui va suivre, introduire la notion de transformation conforme. On dit qu'une transformation $F(s)$ de la variable complexe est une transformation conforme si :

- A un point d'affixe s_1 du domaine considéré correspond un et un seul point transformé d'affixe $F(s_1)$.
- La transformation est analytique sauf en un nombre fini de singularités.
- Le contour choisi doit éviter ces singularités.
- A un contour fermé du plan de la variable s correspond un contour fermé du plan transformé.
- Une transformation conforme conserve les angles des tangentes aux intersections de deux courbes.

Lemme de Cauchy.

On peut énoncer le lemme de Cauchy, vu en mathématiques, de la manière suivante :

Hypothèses : On se donne une transformation conforme $F(s)$ de la variable complexe s (donc analytique), on se donne aussi un contour C fermé sans point double englobant un domaine du plan complexe s , domaine D , contenant P pôles et Z zéros de la transformation $F(s)$.

Lemme : Quand un point M parcourt le contour C dans le sens des aiguilles d'une montre, alors le point transformé M' par la transformation $F(s)$ fait un nombre de tours N autour de l'origine (du plan transformé), dans le sens inverse des aiguilles d'une montre, égal à :

$$N = P - Z$$

C'est ce lemme qui peut être utilisé pour étudier la stabilité d'un système bouclé.

Application à la stabilité d'un système bouclé, critère de Nyquist-Cauchy.

Le problème que l'on se pose est l'étude de la stabilité d'un système bouclé dont la transmittance de la chaîne d'action est $G(s)$, celle de la chaîne de retour $H(s)$. La transmittance de la boucle vaut donc :

$$W(s) = \frac{G(s)}{1 + G(s)H(s)}$$

La stabilité de la boucle dépend donc des pôles de $W(s)$ qui sont les zéros de $1 + G(s)H(s)$. Mais les zéros dangereux instables, sont d'après la théorie de la stabilité, ceux situés dans le demi plan droit du plan complexe.

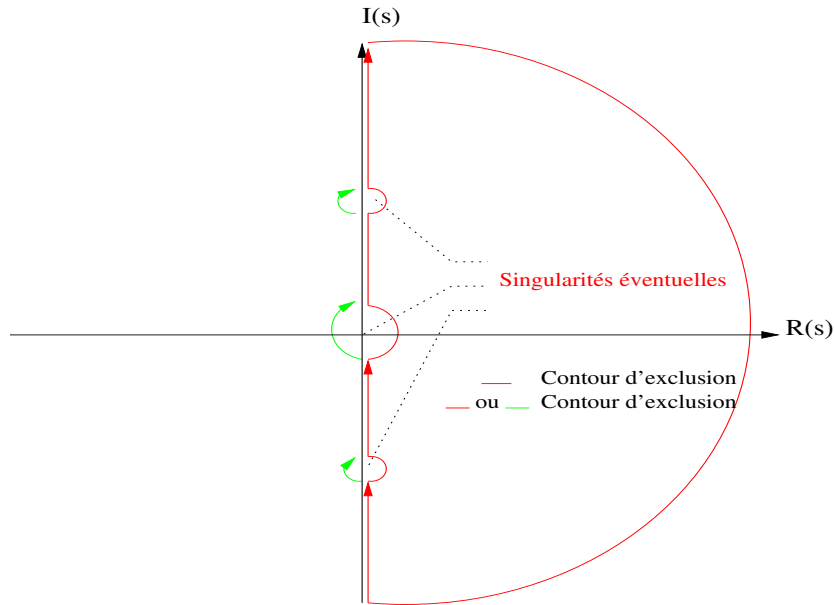


FIGURE 5.3 – Contour d'exclusion

On va donc utiliser le lemme de Cauchy dans le sens $Z = P - N$ en se donnant :

- Un domaine du plan complexe où les **pôles de $W(s)$ ne doivent pas être situés**. Ce domaine contiendra donc tous les points situés dans le demi plan droit du plan complexe et sera entouré par un contour C_{ex} fermé sans point double.
- On tracera normalement les transformés des points de C_{ex} par la transformation $1 + G(s)H(s)$, qui est la transformation $G(s)H(s)$ translatée de l'entier 1. Comme on ne souhaite pas faire cette translation, on comptera le nombre N de tours autour de -1 .
- On comptera le nombre P de pôles de $1 + G(s)H(s)$ qui sont les pôles de $G(s)H(s)$ contenus dans le domaine entouré par C_{ex} et on appliquera la relation $Z = P - N$ pour avoir le nombre de pôles instables de $W(s)$.

Contour d'exclusion C_{ex} et transformé de ce contour pour l'étude de la stabilité d'un système bouclé.

Pour englober l'ensemble du demi plan droit du plan complexe on l'entoure par la droite des nombres imaginaires purs (axe vertical), en prenant soin de contourner les singularités de $G(s)H(s)$ (par la droite ou la gauche) si elles sont sur cet axe : présence d'un intégrateur ou de paire(s) de pôles imaginaires purs conjugués sur l'axe vertical et on ferme le contour par un demi cercle de rayon infini (FIG 5.3).

Quant au transformé par la transformation conforme $G(s)H(s)$ du contour d'exclu-

sion ainsi choisi, il va être constitué de :

1. Du lieu de Nyquist de la boucle ouverte $G(s)H(s)$, points transformés de l'axe imaginaire positif .
2. Du symétrique par rapport à l'axe réel du lieu de Nyquist de la boucle ouverte $G(s)H(s)$, en effet ce lieu est le transformé de l'axe imaginaire négatif et quand on change $j\omega$ en $-j\omega$ dans la transmittance isochrone, on obtient le nombre complexe conjugué et ceci quelque soit la fréquence (parce que $G(s)H(s)$ est un rapport de deux polynômes).
3. Du transformé du grand demi cercle à l'infini qui est un point à distance finie (si le degré du numérateur de $G(s)H(s)$ est égal au degré du dénominateur), ou l'origine (si le degré du numérateur de $G(s)H(s)$ est inférieur au degré du dénominateur).
4. Enfin on cherche les transformés des demi cercles entourant les singularités de $G(s)H(s)$ situés sur l'axe imaginaire pur.
5. Puis on compte le nombre de tours que fait le point transformé autour de -1 et appliquons la relation $Z = P - N$. Un exemple explicite ceci.

Exemple :

Nous allons prendre un exemple de boucle ouverte possédant une singularité à l'origine. Soit

$$G_k(s) = \frac{k}{s(1+s)(1+\frac{s}{3})}$$

k est le gain d'un amplificateur et est supposé positif. Les trois pôles de la boucle ouverte sont $p_1 = 0$, $p_2 = -1$, $p_3 = -3$. Comme il y a une singularité sur l'axe vertical on contourne soit à droite (demi cercle rouge), soit à gauche (demi cercle vert) ce pôle à l'origine. Dans le premier cas $P = 0$ (pas de pôles de la boucle ouverte dans le domaine d'exclusion), dans le second cas $P = 1$, puis on réalise les opérations (FIG 5.4) :

1. tracé du lieu de Nyquist de la boucle ouverte.
2. tracé du symétrique par rapport à l'axe réel de ce lieu.
3. tracé de la fermeture, deux cas sont à envisager :
 - le contournement de l'origine se fait par la droite, alors les points transformés de ce **demi cercle** de rayon infiniment petit, sont sur un **demi cercle** de rayon infiniment grand, pour démontrer cela posons $s = re^{j\varphi}$ et recherchons l'équivalent par $G_k(s)$ de ce demi cercle : $G_k(re^{j\varphi}) \approx \frac{k}{r}e^{-j\varphi}$ qui est un demi cercle de rayon $\frac{k}{r}$ infiniment grand (φ varie de π), on a deux points de ce demi cercle, les points à l'infini sur le lieu de Nyquist et son symétrique, recherchons un troisième point : en prenant le transformé du point A ($\varphi = 0$), on obtient le point A_t , **la fermeture se fait par la droite**.
 - si au contraire le contournement de l'origine se fait par la gauche, le transformé du petit **demi cercle** est encore un **demi cercle** et en recherchant le transformé de B ($\varphi = \pi$) on obtient le point B_t et **la fermeture se fait par la gauche**.

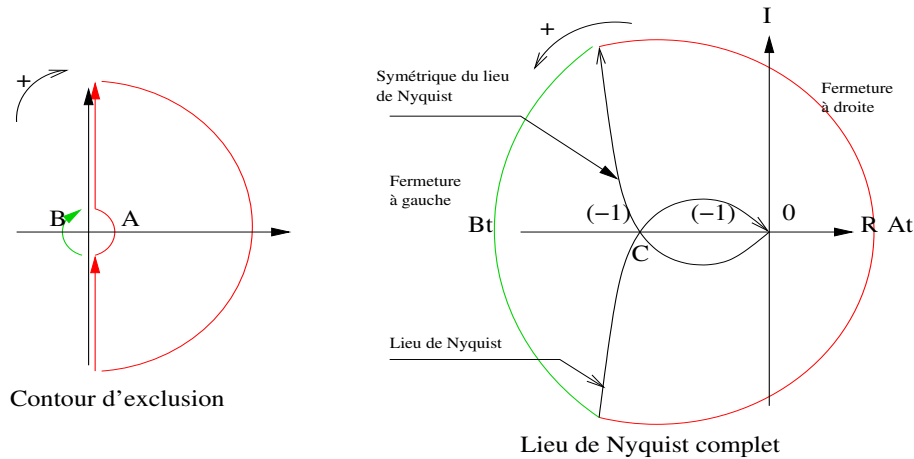


FIGURE 5.4 – Stabilité, exemple

Il nous reste à compter le nombre de tours (N) que fait le point transformé autour du point de coordonnées $[-1, 0]$ et à appliquer la relation $Z = P - N$.

Terminons l'exemple proposé, on voit que le lieu de Nyquist dépend du paramètre k et qu'en changeant ce paramètre on effectue une homothétie du lieu.

- En prenant le premier contour d'exclusion, si la longueur OC est inférieure à 1 alors le point transformé ne tourne pas autour de -1 donc $N = 0$, comme $P = 0$ on a donc $Z = 0 - 0 = 0$; **la boucle fermée est stable**.
- Si l'on prend le deuxième contour d'exclusion on a ici $P = 1$ et l'on voit que $N = 1$, (en respectant les deux sens de rotation) donc $Z = 1 - 1 = 0$ on retrouve le même résultat.

On montrerait facilement que si la longueur OC est supérieure à 1 alors avec le premier contour on a : $Z = 0 - (-2) = 2$: **la boucle fermée est instable**.

Avec le deuxième contour on a $Z = 1 - (-1) = 2$ en effet ici $P = 1$.

Le critère du revers

Le critère du revers, utilisé dans 90% des cas est une version simplifiée du critère de Nyquist-Cauchy et ne s'applique **qu'aux systèmes bouclés dont la boucle ouverte est stable et à déphasage minimum** (dans certains ouvrages on dit que le système bouclé est régulier), en prenant comme contour d'exclusion un contour n'englobant pas les pôles éventuels de l'axe imaginaire du plan complexe.

Son énoncé est le suivant :

Pour un **système stable en boucle ouverte**, il sera **stable en boucle fermée**, si en parcourant le lieu de Nyquist dans le sens des fréquences croissantes **on laisse le point $[-1, 0]$ à gauche**. Quand on raisonne avec le lieu de Black de la boucle ouverte, on remplace le mot gauche par droite.

5.2 Les marges de stabilité d'un système bouclé (robustesse)

Nous avons déjà introduit à la section 4.2.5 la notion de **marge de phase** et de **marge de gain**. Ces deux notions sont issues de la stabilité des systèmes bouclés en utilisant le critère de stabilité de Nyquist-Cauchy, je reviendrai sur ces notions dans l'étude de la synthèse des systèmes bouclés par la méthode fréquentielle. En fait chiffrer la marge de stabilité d'un système bouclé revient à chiffrer **une distance qui sépare un lieu fréquentiel de la boucle ouverte (Nyquist ou Black) au point $[-1, 0]$** .

5.2.1 Marge de stabilité absolue

On peut facilement avec Scilab introduire la notion de **marge de stabilité absolue** en utilisant l'instruction `horner()`.

Si $G(s)$ est la transmittance isomorphe de la chaîne d'action d'un système bouclé dont la chaîne de retour a pour transmittance $H(s)$, alors la transmittance du système bouclé vaut :

$$W(s) = \frac{G(s)}{1 + G(s)H(s)} = \frac{N(s)}{D(s)}$$

Si nous appliquons le critère de Routh-Hurwitz non pas au polynôme $D(s)$ mais au polynôme $D_\alpha(s) = D(s - \alpha)$, avec $\alpha > 0$, alors la condition : les racines de $D_\alpha(s) = 0$ dans le demi plan gauche, impliquera que les racines de $D(s) = 0$ sont situées à gauche de la droite d'abscisse $-\alpha$. Si l'on peut trouver une valeur de $\alpha > 0$ satisfaisant cette condition, alors cette valeur maximale pour α sera appelée **marge absolue de stabilité** et est notée m_a .

L'application du critère de Routh-Hurwitz au polynôme $D_\alpha(s)$ implique que toute racine de cette équation est à gauche de la verticale $-\alpha$ dans le plan complexe, et alors toute conséquence de perturbation appliquée à l'instant t_0 aura une décroissance aussi rapide qu'une exponentielle de la forme $e^{-\alpha(t-t_0)}$. Voici un exemple de programme Scilab.

On donne un système bouclé à retour unitaire de transmittance de chaîne d'action

$$G(s) = \frac{1}{s(1+s)(1+\frac{s}{3})}$$

on met en cascade avec $G(s)$ un amplificateur de gain k positif on obtient donc un système bouclé stable si $k \leq 4$. En prenant $k = 2$, on obtient pour marge de stabilité absolue $m_a = 0,1855394$, qui correspond aux pôles les plus à droite du plan complexe, voici le programme :

```
-->s=%s;g=syslin("c",1/(s*(1+s)*(1+s/3)));
-->[kl,p1]=kpure(g)
p1 =
! - 1.7320508i !
kl =
```

```

4.
-->//ici on prend k=kl/2
-->k=kl/2
-->//la boucle fermée
-->w=k*g/(1+k*g);
-->rw=roots(w("den"))
rw =
! - 0.1855394 + 1.2723832i !
! - 0.1855394 - 1.2723832i !
! - 3.6289211 !
-->rmax=abs(max(real(rw(2,1))))
rmax =
0.1855394 //marge de stabilité absolue
Ce système bouclé de transmittance

```

$$W(s) = \frac{2G(s)}{1 + 2G(s)} = \frac{6}{6 + 3s + 4s^2 + s^3}$$

a deux pôles dominants complexes conjugués $p_{1,2} = -0,1855394 \pm 1,2723832j$, la valeur absolue du maximum de la partie réelle des pôles est bien m_a .

On peut maintenant envisager le **problème inverse**, à savoir par exemple trouver le gain k pour que le ou les pôles les plus à droite soient situés sur une verticale donnée. Reprenons le même exemple et recherchons k pour que les pôles dominants soient situés sur la verticale passant par $-0,2$ ($m_a = 0,2$).

```

-->ga=horner(g,s-.2)//changement de variable
-->ga=syslin("c",ga)//indispensable pour redéfinir un système continu
//ou ga.dt="c";
-->[k1,p1]=kpure(ga)
p1 =
! - 1.2328828i !
k1 =
1.872//c'est la valeur du gain recherché
//Vérification
-->w1=k1*g/(1+k1*g)
w1 =
5.616
-----
2 3
5.616 + 3s + 4s + s
-->rac=roots(w1("den"))//vérification
rac =
! - 0.2 + 1.2328828i !
! - 0.2 - 1.2328828i !
! - 3.6 !

```

Dans cet exemple nous avons trouvé le gain de la chaîne d'action $k = 1,872$, donnant une marge de stabilité absolue $m_a = 0,2$.

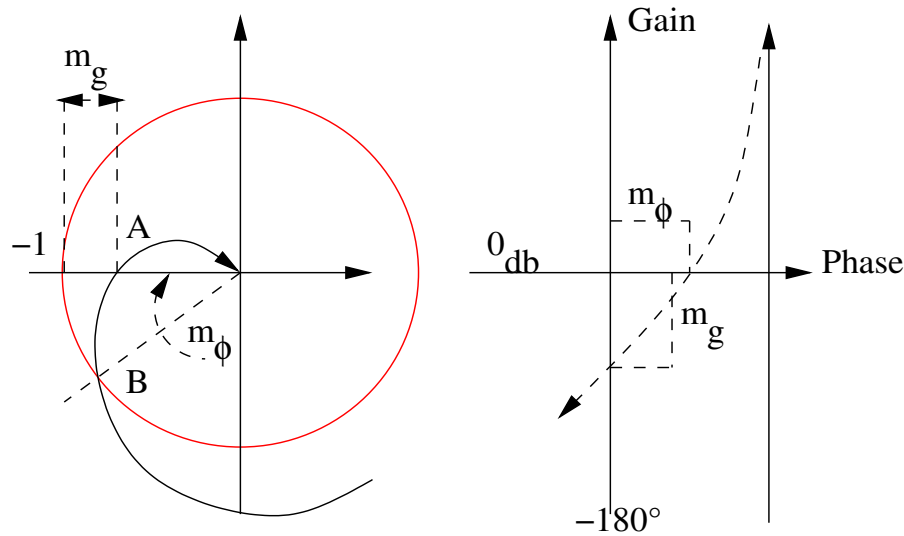


FIGURE 5.5 – Marge de phase, de gain

5.2.2 Marge de phase, marge de gain

Comme nous l'avons dit en introduction, le chiffrage de la distance qui sépare un lieu fréquentiel du point critique $[-1, 0]$, peut être fait dans le plan de Nyquist (ou Black), en calculant sur l'axe horizontal la distance qui sépare le point A au point $[-1, 0]$. Cette distance, (homothétie pour le lieu de Nyquist, translation vers le haut pour le lieu de Black), exprimée en **décibels** est la **marge de gain** (m_g).

Quant à la **marge de phase** m_ϕ c'est la phase supplémentaire (retard de phase) qu'il faut rajouter (en négatif) à un lieu fréquentiel de la boucle ouverte pour faire passer ce lieu par $[-1, 0]$: rotation dans le sens des aiguilles d'une montre à faire au point B pour le faire passer par le point critique (FIG 5.5). Je ne fais pas d'exercice avec le logiciel Scilab pour illustrer ces deux notions, on verra cela lors des exercices de synthèse en utilisant la méthode de Black.

Les fonctions Scilab donnant les marges de gain et de phase sont les fonctions `m_margin` et `p_margin` respectivement.

5.2.3 Marge de gain-phase (marge de module ou marge de stabilité)

L'inconvénient des marges de phase et de gain réside dans le fait que pour certains systèmes réels et en particulier quand on a négligé dans le modèle un petit retard pur, on obtient pour la boucle ouverte un lieu de Black moins horizontal qu'il ne l'est en réalité (plus la fréquence augmente et plus le déphasage devient important), dans ces conditions, imposer une bonne marge de phase ne veut pas dire que la marge de gain (qui est calculée pour une fréquence supérieure à la fréquence de coupure à 0 db)

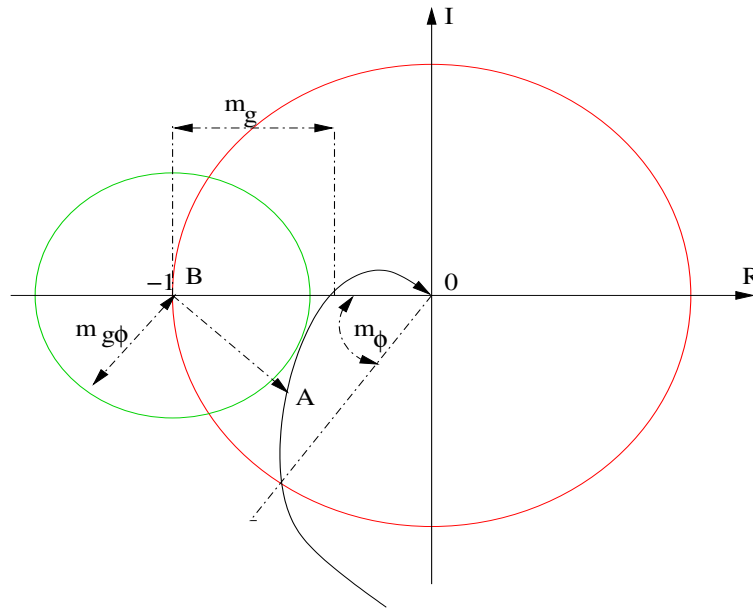


FIGURE 5.6 – Marge de gain-phase ou marge de module.

sera bonne même si on le croît. Il est donc intéressant de regrouper les deux marges précédentes sous la forme d'un seul nombre qui mesure la distance minimale entre le lieu de Nyquist de la boucle ouverte et le point $[-1, 0]$. Ce nombre est la marge de gain-phase (marge de module) notée $m_{g\phi}$ (FIG 5.6) ; c'est le rayon du plus grand cercle de centre -1 tangent au lieu de Nyquist de la boucle ouverte **supposé situé à sa droite (système bouclé stable)**.

Calcul avec Scilab de la marge de gain-phase ou marge de module

Pour un système bouclé stable donné, ceci est très simple en utilisant l'instruction `repfreq()` : on calcule en fonction de la fréquence la distance du point courant du lieu de Nyquist avec le point $[-1, 0]$ et l'on cherche la valeur minimale de cette distance, on obtient ainsi la fréquence correspondante et cette distance minimale. Cette fréquence et cette distance minimale n'ont pas de rapport avec la fréquence de résonance en boucle fermée et avec le coefficient de surtension. Si A est le point courant sur le lieu de nyquist et B le point $[-1, 0]$, on a :

```
-->s=s;g=syslin("c",.5/(s^3+s^2+s));
-->nnyquist(g,.05,10)//pour avoir une idée de la fréquence recherchée
//utiliser la fonction gestionnaire de datatips dans la figure
-->[f,rep]=repfreq(g,.135,.140);//real(rep) et imag(rep) représente le
point A.
-->BAx=real(rep)+ones(rep);
```

```

-->BAy=imag(rep);
-->LBA=sqrt(BAx.^2+BAy.^2); //longueur du vecteur BA.
-->mgp=min(LBA)
mgp =
0.4234996 //valeur de la marge de module.
-->fmgp=f(find(LBA==mgp))
fmgp =
0.1397121
-->mgpdb=20*(log(mgp)/log(10))
mgpdb =
- 7.4629404 //distance en db.
Voici un programme faisant la même chose mais qui utilise l'instruction orthProj.
-->[f,rep]=repfreq(g,.135,.140);
-->X=real(rep)';
-->Y=imag(rep)';
-->mgp=orthProj([X,Y],[-1,0])
mgp =
0.4234218
-->[mgp,ptp,ind,c]=orthProj([X,Y],[-1,0])
c =
0.5316343
ind =
7.
ptp =
- 0.6105550 - 0.1661884i
mgp =
0.4234218
-->fmgp=f(ind)
fmgp =
0.1381249

```

Remarque : Par l'instruction `orthProj(data,pt)` on peut trouver la distance entre un point et un ensemble de valeurs numériques représentant une courbe (ceci est utile si l'on veut calculer la fréquence de résonance en boucle fermée).

Enfin un dernier programme noté `m_margin` (marge de module), qui à partir de la boucle ouverte donne la fréquence `fmgp` puis le gain `mgp`. Ce programme est plus précis, car utilise directement la boucle ouverte $G(s)$ ou $G(s)H(s)$, et recherche la/les valeurs de la pulsation sur le lieu de Nyquist où la/les **normales** au lieu passent par le point $[-1,0]$: ceci revient à étudier la sensibilité et son inverse.

```

-->s=%s;g=syslin("c",.5/(s^3+s^2+s));
-->[mgp,fmgp]=m_margin(g)
fmgp =
0.1389461
mgp =
0.4233591
-->nnyquist(g,.1,1)//Le lieu de Nyquist, je ne donne pas la figure.

```

5 Etude de la stabilité d'un système, marges de stabilité

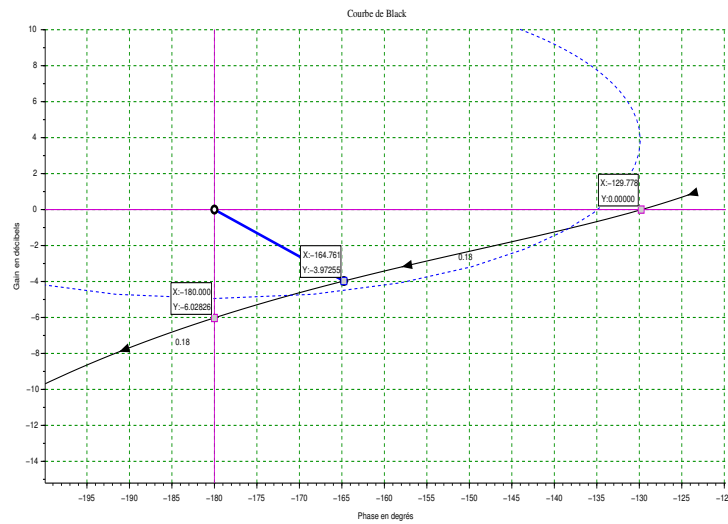


FIGURE 5.7 – Les marges dans le plan de Black

```
-->xarc(-1-mgp,mgp,2*mgp,2*mgp,0,64*360)//Cercle centré au point [-1,0],
rayon mgp.
-->arc=get("hdl");//Récupération du "handle".
-->arc.line_style=6;arc.foreground=6;//Style, couleur.
```

Représentons la marge de module dans le plan de Black, (suite de l'exercice), FIG 5.7.

```
-->figure(1);
-->bblack(g,0.08,1)//
-->[fmgp,mgp,phimgp]=dbphifr(g,fmgp)
phimgp =
- 164.7613
mgp =
- 3.9725522
fmgp =
0.1389461
-->xpoly([-180,phimgp],[0,mgp])
-->p=get("hdl");
-->p.foreground=2;
-->p.thickness=3;
-->p.mark_style=9;
```

Exemple de système ayant des bonnes marges de gain et phase mais une mauvaise marge de module

On considère le système à retour unitaire de transmittance en boucle ouverte :

$$G(s) = \frac{0.39(s^2 + 0.1s + 0.55)}{s(s + 1)(s^2 + 0.06s + 0.5)}$$

avec Scilab nous allons calculer les différentes marges.

```
-->s=%s;G=syslin("c",.39*(s*s+.1*s+.55),s*(s+1)*(s*s+.06*s+.5))
-->[mphi,fmphi]=p_margin(G)
fmphi =
0.0662823
mphi =
69.32547 //Excellent !
-->[mg,fmg]=g_margin(G)
fmg =
[]
mg =
Inf //Idem
-->[mmodule,fmmodule]=m_margin(G)
fmmodule =
0.1137814
mmodule =
0.2573004
-->nnyquist(G,.01,10,"G")
-->xarc(-1-mmodule,mmodule,2*mmodule,2*mmodule,0,64*360)
-->arc=get("hdl");
-->arc.line_style=6;arc.foreground=6;
```

Dans cet exemple on voit bien que la marge de phase est importante, la marge de gain aussi!!, mais que la marge de stabilité (marge de module) n'est pas très grande, il vaut donc mieux faire intervenir la marge de module pour bien chiffrer le degré de stabilité.

5.2.4 Marge de retard

Comme nous venons de le dire précédemment, le retard pur (quelquefois négligé), produit un déphasage qui augmente avec la fréquence et est source d'instabilité. En ramenant la marge de phase à la pulsation à laquelle elle a lieu, on introduit la marge de retard notée

$$m_r = \frac{m_\varphi}{\omega_{cr}} = \frac{\pi + \Phi_{cr}}{\omega_{cr}}$$

ω_{cr} et Φ_{cr} sont respectivement la pulsation de coupure à 0 db et le déphasage de la boucle ouverte correspondant. Ainsi plus cette pulsation de coupure à 0 db (pulsation de croisement) sera importante moins le système sera tolérant vis à vis des retards purs. Il faut donc que la bande passante ne soit pas trop importante (à vous de trouver le bon compromis). Je ne fais pas le calcul avec Scilab : utiliser l'instruction `p_margin`.

5.2.5 Valeurs usuelles des différentes marges

En résumé les différentes marges proposées sont les marges de robustesse, et chiffrent la distance minimale par rapport au point critique. Ces marges caractérisent la robustesse de la boucle fermée vis-à-vis des variations des paramètres de la boucle ouverte. Voici la liste donnant ces différentes marges.

- **Marge de gain** : $m_g = \min_i \left(\frac{1}{|G_{bo}(j\omega_{180,i})|} \right)$ pour $\Phi(\omega_{180,i}) = -180^\circ$. Marges usuelles 10 à 12db.
- **Marge de phase** : $m_\varphi = \min_i (180^\circ - |\Phi(\omega_{cr,i})|)$ pour $|G_{bo}(\omega_{cr,i})| = 1$. Avec $\omega_{cr,i}$ les pulsations de croisements. Marges usuelles 40 à 60 degrés.
- **Marge de retard** : $m_r = \frac{m_\varphi}{\omega_{cr}}$.
- **Marge de module** : $m_{g\varphi} = |1 + G_{bo}(j\omega)|_{\min}$. Valeur supérieure à 0.5 (-6db), minimale 0.4(-8db).

5.2.6 Cercles à gain constant, cercles à phase constante dans le plan de Nyquist : M et N cercles, abaque de Hall

Nous avons vu à la section 4.2.4 l'abaque de Black qui donne le gain (en db), la phase (en degrés), d'un système bouclé à retour unitaire, connaissant le gain et la phase de la chaîne d'action. Et bien, l'abaque de Hall, est la présentation dans le plan de Nyquist de la transformation homographique $W(s) = \frac{G(s)}{1+G(s)}$, avec $G(s)$ la transmittance de la chaîne d'action et $W(s)$ la transmittance du système complet : c'est la transcription dans le plan de Nyquist de l'abaque de Black. Scilab donne par l'instruction `m_circle()` ou `m_circle(vecteurgain)` les courbes isogains mais ne donne pas, contrairement à l'instruction `nicholschart()`, les courbes isophases. On verra, en faisant `apropos m-circle` dans Scilab, l'exercice proposé. Pour les curieux, vous verrez dans le livre [3, pages 278-279], la théorie sur l'abaque de Hall et la transformation homographique $W(s) = \frac{G(s)}{1+G(s)}$.

5.2.7 Nouveauté : utilisation des pseudo-lieux, pôles dominants sur une droite à amortissement constant

Le problème que l'on peut se poser est le suivant ¹ : étant donné un **système bouclé stable** on cherche, par analogie avec un système du second ordre (dont l'amortissement est $\xi < 1$, pôles complexes conjugués), à avoir deux pôles dominants sur des droites (droites OA , OB) passant par l'origine et faisant un angle ψ donné par rapport à l'axe vertical du plan complexe : **on fait du placement des pôles dominants**.

Utilisation du lemme de Cauchy.

Ne voulant pas raisonner en temporel, on va rechercher les transformées de ces droites (en Nyquist ou Black) par la transformée $G(s)H(s)$: on aura donc affaire à

1. A ma connaissance cette méthode n'a jamais été publiée et mise en oeuvre ; on trouverait peut être une méthode semblable dans les très vieux ouvrages d'automatique : si lecteur vous le savez, informez moi.

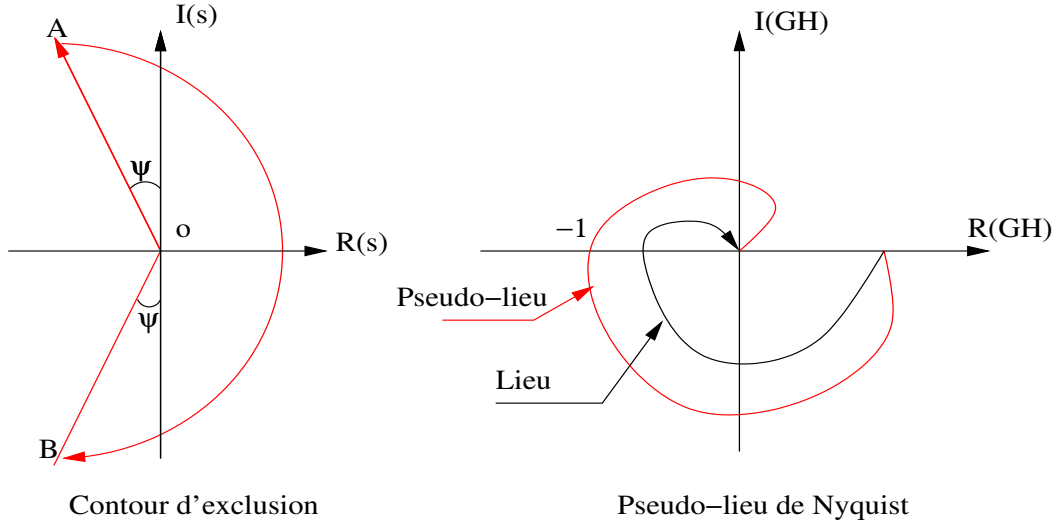


FIGURE 5.8 – Nouveau contour et pseudo-lieu

des pseudo-lieux (voir section 5.1.4).

Il est évident qu'en réglant les paramètres de la boucle ouverte (gain, paramètres de réseaux correcteurs éventuels), de telle sorte que ce pseudo-lieu passe par le point $[-1, 0]$, on placera les deux pôles dominants de la boucle fermée sur ces deux droites (utilisation du lemme de Cauchy) ; ce qui change dans cette méthode, c'est le contour d'exclusion² (FIG 5.8).

Avant de commencer, nous allons définir (pour $\Im(s) > 0$) l'équation de la droite OA qui fait un angle ψ par rapport à l'axe vertical. Si je pose $\theta = \psi + \frac{\pi}{2}$, on pourra utiliser les pseudo-lieux. Dans ces conditions, l'équation de la droite OA vaut : $\omega e^{j\theta}$ avec $\omega \in [0, +\infty]$ et θ l'angle précédemment défini. Bien entendu, si l'on cherche à avoir les pôles de la boucle fermée en dehors de ce domaine d'exclusion, on devra tracer le symétrique du pseudo-lieu de Nyquist par rapport à l'axe réel. De même, si le pseudo-lieu de Nyquist possède des points à l'infini, il faut assurer la fermeture pour le lieu complet. Enfin, si la boucle ouverte possède P pôles dans le domaine d'exclusion, on devra utiliser le lemme de Cauchy complet, à savoir : $Z = P - N$. Dans le cas où la boucle ouverte ne possède pas de pôles dans le domaine d'exclusion choisi, on aura l'équivalent du critère du revers et ainsi on pourra mettre facilement en œuvre la méthode.

2. Je ne démontre pas, mais cela est facile, que si on change θ en $-\theta$, le nouveau pseudo-lieu est symétrique par rapport à l'axe réel du pseudo-lieu.

Un exemple de mise en œuvre de cette méthode : réglage du gain k de la chaîne d'action d'un système bouclé.

Avant tout une **remarque essentielle** : sur la figure 5.7 nous avons choisi un contour défini par deux demi droites et un arc de cercle de rayon infini, pour que la méthode s'applique simplement, (critère du revers) il faut s'assurer qu'il n'y a pas de pôles (p_i) de la boucle ouverte dans le domaine défini par ce contour, dans ce cas des Z_j qui sont les pôles de la boucle fermée (Lemme de Cauchy) devront être sur les deux droites OA OB, donc le pseudo lieu devra passer par le point $[-1, 0]$ dans le plan de Nyquist ; à vous de trouver les réglages du correcteur pour avoir cette condition.

Nous allons reprendre l'exercice vu à la section 5.2.1 et faire varier le gain de la chaîne d'action : correcteur proportionnel. De même je rappelle que pour un système du second ordre d'amortissement ξ , on a $\sin \psi = \xi$ car $\sin \psi = \frac{\xi \omega_n}{\omega_n} = \cos \theta$ (voir la figure FIG 4.6 du paragraphe 4.3.2). On va faire l'étude pour un angle $\theta = 120^\circ$; on trouvera un autre exemple avec un système à retard pur au paragraphe 10.2.5. Nous savons que les pôles de la boucle ouverte ($p_1 = 0, p_2 = -1, p_3 = -3$) ne sont pas dans le domaine défini par le contour choisi : donc la méthode proposée s'applique simplement.

```
-->s=s; g1=syslin("c",1/(s*(1+s)*(1+s/3)));
-->bblack([g1;g1],.01,1,["90";"120"],["g1","pseudo120"])
```

Ici on a tracé le lieu de Black classique de $g_1(s)$, ($\theta = 90^\circ$) et le pseudo lieu pour $\theta = 120^\circ$ (ceci pour un gain de la chaîne d'action $k = 1$). On travaille maintenant sur le pseudo-lieu que l'on fait passer par le point -1 . (Je ne donne pas les courbes de Black : exécutez les deux instructions précédentes).

```
-->[bout,posx,posy]=xclick();//Utiliser locate(1) si vous voulez
-->k=10^(-posy/20)//c'est le bon gain (k=0.61 environ)
-->gk=k*g1;
```

Autre commentaire, par `xclick()` (après un bon zoom) je recherche sur l'axe -180° le point du pseudo-lieu de Black, (manière de connaître la marge de gain, car ici on ne peut pas utiliser `g_margin()`), puis je calcule le gain qu'il faut donner au système pour faire passer le pseudo-lieu par $[-1, 0]$: ici on doit **descendre** le lieu de Black, donc comme `posy` est positif, l'exposant intervenant dans `k` doit être négatif. La suite du programme est une vérification.

```
-->W=gk/(1+gk)//la boucle fermée
W =
    1.8263613
-----
          2    3
1.8263613 + 3s + 4s + s
//Vérification
-->rbf=roots(W("den"))//les pôles de W
rbf =
    - 3.249797
    - 0.3751015 + 0.6490696i
    - 0.3751015 - 0.6490696i
```

```

-->amort=abs(real(rbf(2)))/abs(rbf(2))
amort =
    0.5003612 //Le zéta précision du xclick()
-->//Angle choisi vaut (%pi/2 + %pi/6) soit 120° que l'on compare :
-->angle=atand(imag(rbf(2)),real(rbf(2)))
angle =
    120.02389 //au lieu de 120° (voir instruction précédente bblack)
-->[gmarg,fgm]=g_margin(gk)
fgm =
    0.2756644
gmarg =
    16.40835
-->[pmarg,fmp]=p_margin(gk)
fpm =
    0.0838767
pmarg =
    52.246669
-->[mgp,fmgp]=m_margin(gk)
fmgp =
    0.1328766
mgp =
    0.6362511 //c'est excellent !!
-->bblack([gk;g1;g1],.01,1,["90";"90";"120"],["gk","g1","pseudog1"])

```

On compare l'amortissement (du aux deux pôles les plus à droite, dominants) et le $\sin \psi$, il y a une différence très faible du à l'instruction `xclick`. On trouve une marge de gain de 16, 4db, une marge de phase de 52, 246° et une marge de module de 0.6362. Quant au troisième pôle (réel) il est beaucoup plus à gauche dans le plan complexe.

On trace le lieu de Black du système corrigé avec cette valeur de k : (gk), le vrai lieu de Black pour $k = 1$: ($g1$), et le pseudo-lieu pour $k = 1$: ($pseudog1$) (FIG 5.9). La valeur de k trouvée assure un amortissement tel que $\xi = \sin \frac{\pi}{6}$. Je reviendrais sur cette méthode en la comparant avec la méthode de synthèse dite méthode d'Evans.

Remarque : Nous avons tracé ici le pseudo-lieu de Black du système de transmittance $g_k(s) = \frac{k}{s(1+s)(1+\frac{s}{3})}$, et ceci pour une valeur de $s = \omega e^{(j\frac{4}{3}\frac{\pi}{2})}$. Ceci veut dire que ce pseudo-lieu, est le lieu de Nyquist, Bode et Black, du système de transmittance $g_{\alpha,k} = \frac{k}{s^\alpha(1+s^\alpha)(1+\frac{s^\alpha}{3})}$ avec $\alpha = \frac{4}{3}$: l'échelle des graduations en fréquences n'est pas la même, sur le lieu de Nyquist et de Black mais le lieu a la même forme, (pas en Bode car on remplace s par s^α). De même si $g_k(s) = \frac{k}{(1+s)}$ on a affaire à un système du premier ordre et tracer le pseudo-lieu de ce système revient à étudier fréquemment le modèle dit Coole et Coole ou explicite $g_{\alpha,k}(s) = \frac{k}{(1+s^\alpha)}$, (voir l'étude de ce modèle au chapitre 11).

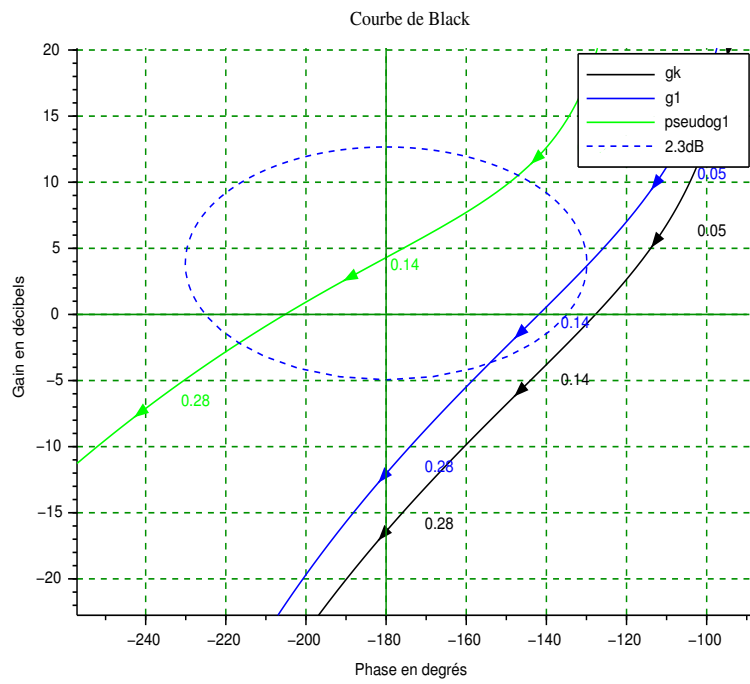


FIGURE 5.9 – Synthèse par les pseudo-lieux de Black

6 Principe de la commande

6.1 Introduction

Le but que se fixe l'automaticien en élaborant un système bouclé, consiste au choix d'un réseau correcteur qui mis en cascade avec la procédé, permet de réaliser un système bouclé ayant des performances nettement supérieures au système originel. Avant toute étude du système bouclé, il faut donc connaître les qualités et défauts du procédé (la boucle ouverte).

On peut, et cette liste n'est pas exhaustive, découvrir que le procédé est (ou n'est pas) :

- lent, son inertie est trop importante ;
- mal amorti, il oscille trop longtemps sous l'effet d'une perturbation, ou d'un changement de point de consigne ;
- il a tendance à dériver, sa sortie évoluant, alors que l'entrée reste constante : ceci est le signe de la présence d'une ou plusieurs intégrations ;
- est instable, le correcteur devant donc le rendre stable, avant de le corriger.

Du point de vue de l'automaticien, une structure de réseau correcteur et les valeurs des paramètres de celui ci doivent permettre :

- d'améliorer la stabilité si nécessaire, ainsi qu'améliorer le comportement statique et dynamique du procédé ;
- d'obliger le système à suivre au plus près la consigne désirée (même si celle ci évolue au cours du temps) : problème de poursuite. Il faut aussi que le réseau correcteur puisse annuler, autant que faire ce peut, les perturbations qui ne manquent pas d'influencer la dynamique du procédé : rejet des perturbations ;
- Il sera donc nécessaire d'étudier à tout instant, erreur de l'asservissement : différence entre ce que l'on souhaite avoir et ce que l'on a réellement.

On peut faire un schéma représentant un procédé bouclé avec un réseau correcteur (FIG 6.1).

On voit apparaître sur ce schéma, l'erreur de l'asservissement, différence entre la consigne et la sortie. Dans la suite on définira la précision statique (ce qui revient à faire l'étude de l'erreur en régime permanent) et la précision dynamique, caractérisant cette erreur au cours du temps.

6 Principe de la commande

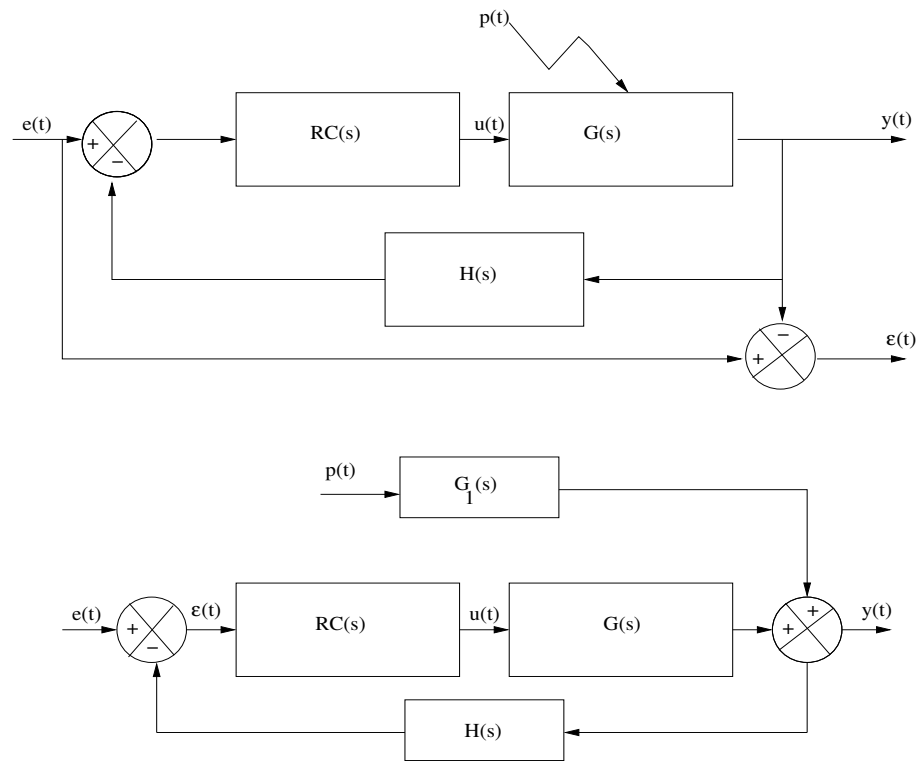


FIGURE 6.1 – Schéma d'un système bouclé

6.2 Précision

6.2.1 Précision statique

Avec le schéma précédent, on déduit d'après le principe de superposition, que l'erreur de l'asservissement vaut :

$$\varepsilon(s) = e(s) \frac{1}{1 + RC(s)G(s)} - p(s) \frac{G_1(s)}{1 + RC(s)G(s)}$$

En vertu du théorème sur les limites de la transformée de Laplace, l'étude de l'erreur permanente revient à faire l'étude de $s\varepsilon(s)$ quand $s \rightarrow 0$, et ceci pour un signal d'entrée donné (donc pour une expression de $e(s)$).

— Erreur statique due à la consigne ;

Seul l'original du premier terme intervient, on a donc le tableau suivant : (K est le gain statique en position de la chaîne d'action, K_1 le gain en vitesse, et K_2 le gain en accélération (gains apparaissant dans la factorisation de Bode)).

nombre de pôles en 0	0	1	2	> 2
erreur en position	$\frac{1}{1+K}$	0	0	0
erreur en vitesse	∞	$\frac{1}{K_1}$	0	0
erreur en accélération	∞	∞	$\frac{1}{K_2}$	0

— Erreur statique due à la perturbation ;

Dans cette situation, en vertu du principe de superposition, on peut mettre la consigne à zéro et l'on montre facilement, par l'étude du deuxième terme intervenant dans $\varepsilon(s)$, que l'erreur statique due à cette perturbation, diminue en augmentant le nombre de pôles à l'origine (en amont du point d'application de la perturbation), ainsi qu'en augmentant le gain statique de la chaîne d'action.

6.2.2 Précision dynamique

Le comportement dynamique d'un système peut être entièrement caractérisé par sa réponse impulsionnelle (ou aussi par sa réponse indicielle).

L'étude temporelle étant généralement plus complexe on préfère se ramener à une étude fréquentielle (généralement dans le plan de Black) et comparer le comportement du système par rapport à des systèmes connus : du premier et/ou second ordre.

Comparaison avec un premier ordre.

Que la boucle ouverte $RC(s)G(s)$ ait pour modèle une transmittance $\frac{k}{s}$ ou $\frac{k}{1+\tau s}$ la boucle fermée sera toujours du premier ordre (seule l'erreur permanente à un échelon différera : elle sera nulle dans le premier cas et vaudra $\frac{1}{1+k}$ dans le second). Mais si l'on considère l'erreur par rapport à cette réponse permanente, on sait que cette erreur est inférieure à 5 % pour un temps $t_5 = 3\tau_1$ (τ_1 constante de temps de la boucle fermée) : ce temps est aussi appelé **temps de réponse** (à 5 %).

6 Principe de la commande

Si l'on se ramène maintenant dans le domaine fréquentiel, on remarque que la rapidité de réponse est directement liée à la bande passante du système bouclé, bande passante que l'on caractérise par la pulsation de coupure (à 3 db ou 6 db, la cassure du lieu de Bode se faisant en un point de pulsation $\omega = \frac{1}{\tau_1}$). Vous remarquerez ceci en changeant dans le modèle s en $\tau_1 s$, et dans ce cas vous changez t en $\frac{t}{\tau_1}$ et ω la pulsation en $\tau_1 \omega$. En se fixant le temps de réponse à 5 % on se donne $\tau_1 = \frac{t_5}{3}$, donc la pulsation de coupure à 3 db est donnée par $\frac{1}{\tau_1} = \frac{3}{t_5}$: toutes ces propriétés apparaissent au paragraphe 4.3.1.

Comparaison avec un second ordre.

Afin d'obtenir une erreur permanente nulle pour une réponse à un échelon, on choisit pour boucle ouverte du système de référence (à retour unitaire) :

$$RC(s)G(s) = \frac{k}{s(1 + \tau s)}$$

La boucle fermée vaut donc :

$$W(s) = \frac{k}{k + s + \tau s^2}$$

et est un système du second ordre de pulsation naturelle $\omega_n = \sqrt{\frac{k}{\tau}}$ et de coefficient d'amortissement $\xi = \frac{1}{2\sqrt{k\tau}}$ (voir paragraphe 4.3.2).

Si le coefficient $\xi > 1$ les pôles de $W(s)$ sont réels et la réponse indicielle est semblable à celle d'un système du premier ordre (sauf pour $t = 0$).

Quand $\xi < 1$ on peut constater les faits suivants :

- La valeur du premier dépassement D_1 % de la réponse indicielle constitue un bon indicateur de l'amortissement. Ce premier dépassement, ainsi que le temps de pic (valeur du temps pour ce premier pic), valent respectivement :

$$D_1 \% = 100 \exp\left(-\pi \frac{\xi}{\sqrt{1 - \xi^2}}\right) \text{ et } t_{pic} = \frac{\pi}{\omega_n \sqrt{1 - \xi^2}}$$

- A ξ fixé, les oscillations sont d'autant plus rapides que la pulsation naturelle est grande et à ω_n fixé, les amplitudes des oscillations s'amortissent d'autant plus rapidement que ξ est grand. Mais attention si ξ est trop grand, la réponse peut être très lente.

Par une étude complète du système du second ordre on montre, qu'à ω_n fixé, le temps de réponse à 5 % passe par un minimum pour ξ voisin de 0,7, le premier dépassement valant alors 4 %.

Quant à la réponse fréquentielle, on la caractérise par le facteur de résonance Q , qui pour un système du second ordre vaut :

$$Q = \frac{1}{2\xi\sqrt{1 - \xi^2}}$$

Pour un facteur de résonance $Q = 1,3$ soit $Q_{db} = 2.3\text{ db}$, on a une valeur de ξ de 0,42, valeur donnant un dépassement $D_1\% = 30\%$. Cette valeur pour Q est souvent prise comme référence.

De même, comme la pulsation de résonance vaut $\omega_r = \omega_n \sqrt{1 - 2\xi^2}$, cette pulsation est un indicateur de la rapidité de la réponse indicielle : plus ω_n et donc ω_r sont élevés et plus le temps de réponse est court. Vous verrez dans les tableaux concernant les systèmes du second ordre (paragraphe 4.3.2) les différentes relations. Mais retenez que le temps de réponse à 5 % vaut $t_5 \approx \frac{3}{\xi\omega_n}$ (si on impose un temps de réponse à 5 % on se fixe le produit $\xi\omega_n$, et maintenant en s'imposant un amortissement, on détermine tous les paramètres du second ordre). Une dernière remarque, ce produit $\xi\omega_n$ est au signe près la partie réelle des deux pôles complexes conjugués du système du second ordre.

Attention : pour avoir une bonne précision statique il faut soit rajouter un ou plusieurs intégrateurs dans la chaîne d'action et/ou augmenter le gain statique de cette même chaîne, ceci a pour conséquence de déstabiliser ou de rendre plus instable le système bouclé : dilemme stabilité-précision.

6.3 Cahier des charges

A partir des remarques que l'on vient de faire, on peut définir un cahier des charges type. Le système bouclé devra être stable et précis en régime statique : mais attention au fameux dilemme **stabilité-précision**.

Il faudra le régler afin qu'il soit rapide, mais attention à ne pas saturer les actionneurs ; par simulation on vérifiera l'amplitude de la commande : étude du signal $u(t)$. Ceci pourra être fait par analogie avec un système du second ordre.

Toutes ses considérations permettent de faire un dessin permettant de placer les pôles ainsi que les pôles dominants (ceux qui sont le plus près de l'axe imaginaire), dans une région du plan complexe : le dessin ci-dessus présente cette région (FIG 6.2). En résumé lors de la synthèse d'un système asservi on cherchera, pour réaliser un bon asservissement, à avoir :

- un système stable avec des marge de phase et marge de gain suffisantes et une marge de module de l'ordre de 0,5.
- avoir un gain en boucle ouverte assez grand.
- avoir une fréquence de résonance élevée.
- lié à la remarque précédente on devra donc essayer, autant que faire ce peut, d'augmenter la bande passante, mais attention cette augmentation a pour effet de ne pas atténuer les bruits.

6.4 Méthodes de synthèse : pourquoi utiliser la méthode de Black

Nous avons dans les sections précédentes, analysé le comportement temporel et fréquentiel d'un système et introduit les différentes représentations, ainsi que l'abaque

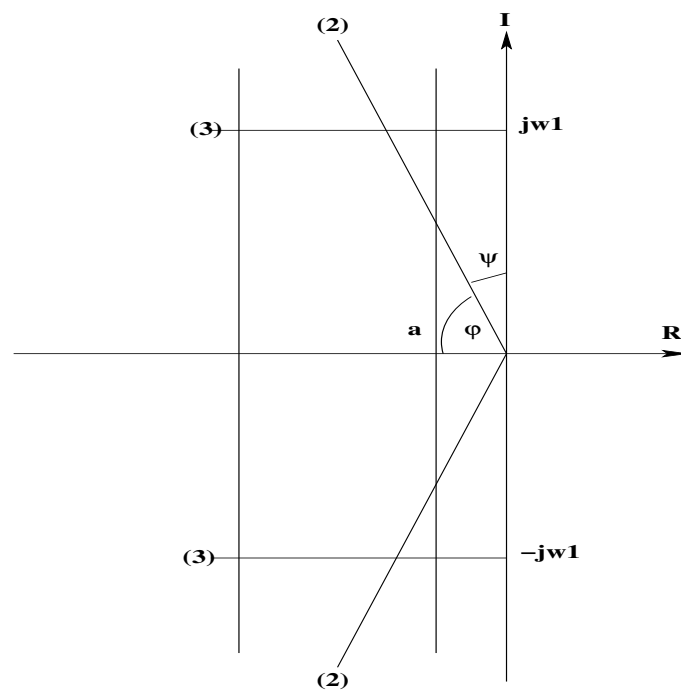


FIGURE 6.2 – Placement des pôles

6 Principe de la commande

de Black. Ce qu'il faut retenir de ces sections peut se résumer par les points suivants :

1. On veut que le système bouclé soit précis, il faut donc se donner cette précision et de là on déduit, s'il faut ou non rajouter un ou plusieurs intégrateurs dans la chaîne d'action et/ou augmenter le gain statique (en position, en vitesse ...) de cette même chaîne d'action.
2. On veut un système bouclé stable (obligation) avec des marges de stabilité.
3. On veut une dynamique déterminée, donc il faut placer les pôles dominants du système bouclé dans une certaine région du plan complexe.
4. On ne veut pas saturer les actionneurs et de plus ne pas consommer trop (?) d'énergie.

Nous avons vu que ceci pouvait être fait, d'une manière peut être un peu approximative, en prenant pour référence un système du premier et/ou second ordre dont on connaît parfaitement le comportement temporel et fréquentiel et que pour ces systèmes il y avait une liaison directe entre le comportement temporel et fréquentiel : du comportement fréquentiel on en déduisait le comportement temporel. On va donc plutôt travailler en fréquentiel car avec cet outil, on déduira les propriétés de la boucle fermée à partir des lieux de transfert de la boucle ouverte : la méthode de Black permet de faire cela d'une manière très élégante. On réalisera donc les opérations suivantes :

1. Ramener votre système bouclé, à un système à retour unitaire.
2. Tracer le lieu de Black de la chaîne d'action de ce système à retour unitaire.
3. A partir du tracé de la courbe de Black de la boucle ouverte, on déduira les propriétés essentielles de la boucle fermée.

Un exemple illustre ces faits (FIG 6.3).

```
-->s=%s;sl=syslin("c",5*(1+s)/(.1*s^4+s^3+15*s^2+3*s+1))
sl =
      5 + 5s
-----
      2    3    4
1 + 3s + 15s + s + 0.1s
-->bblack(sl,.0001,100)
```

Si nous appelons $sb(s)$, dans l'exemple choisi, la transmittance de la boucle fermée, $sl(s)$ étant le modèle de la boucle ouverte, la précision en régime statique vaut donc $sl(0)/(1+sl(0))$.

6.4.1 Principe de mise en oeuvre

On choisira une structure de régulateur permettant de vérifier ce cahier des charges. Il conviendra donc :

- d'éloigner le lieu de Black de la boucle ouverte du point -1 : augmenter la marge de phase et de gain ; avoir une marge de phase supérieure à 45° et une marge de gain supérieure à 10 ou 12 *db*, avoir un facteur de résonance en boucle fermée donné.

6 Principe de la commande

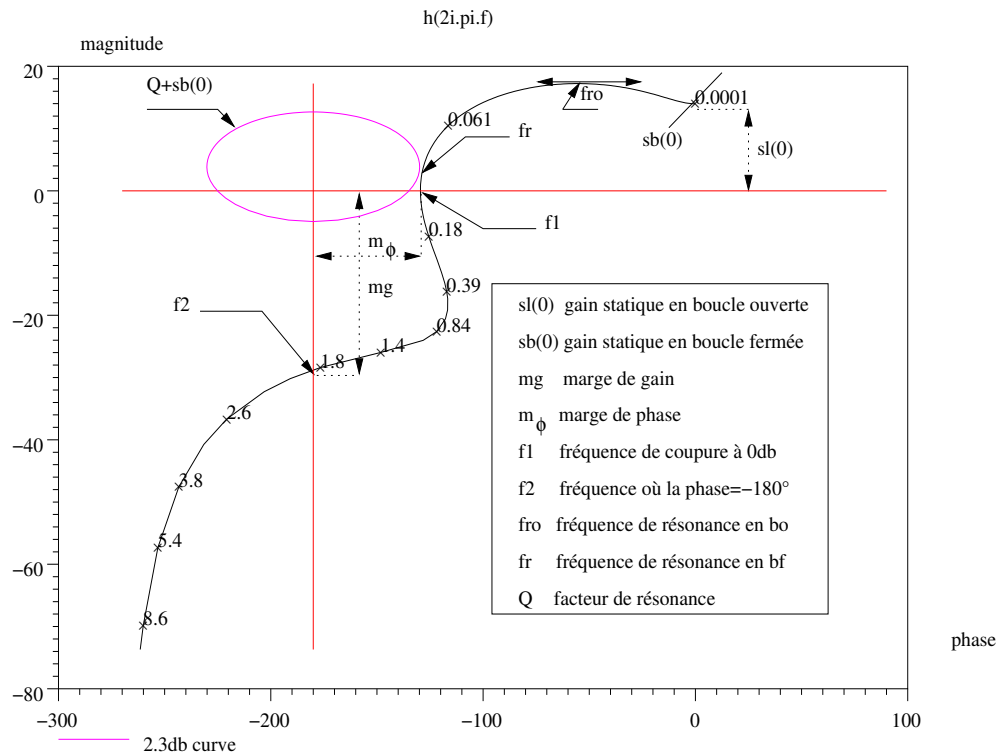


FIGURE 6.3 – Lieu de black

6 Principe de la commande

- d'augmenter le gain de la boucle ouverte, ou mettre le point à fréquence zéro à l'infini (introduction d'un ou plusieurs intégrateurs dans la chaîne d'action).
- d'augmenter la bande passante donc provoquer un tassement en fréquence vers les gains élevés : on diminue le temps de réponse de la boucle. Rappelez vous que pour un système du second ordre, la pulsation de coupure à 0 db , ω_{co} est liée au temps de réponse à 5 % par la relation : $\omega_{co}t_{r\,5\%} \approx \pi$.
- Ceci peut être résumé par le choix d'un réseau correcteur apportant si nécessaire, une avance de phase aux fréquences moyennes et un retard de phase aux basses fréquences.

7 La correction des systèmes par la méthode de Black

Les exercices qui vont suivre ont pour but de réaliser la synthèse d'un réseau correcteur de structure donnée, (avance de phase, retard de phase, retard-avance de phase, P.I.D. . .) afin que le système bouclé à retour unitaire constitué de la mise en série d'un système et du réseau correcteur choisi ait certaines propriétés. Mais avant cela, nous allons voir à quel endroit de la boucle il est peut être intéressant de placer le ou les réseaux correcteurs.

7.1 Choix de la structure de réseau correcteur

Ce choix dépend beaucoup des informations qui sont disponibles, en qualité, sur le procédé. Le cas le plus courant est de disposer d'un capteur récupérant une image (plus ou moins bonne) de la sortie. C'est donc à partir de cette seule information, éventuellement traitée et mise sous un format normalisé, que l'on réalisera le bouclage.

Dans quelques cas particuliers on dispose d'informations supplémentaires pouvant caractériser le procédé : un asservissement de position par exemple, où l'on capte la position et la vitesse à l'aide d'un dispositif adéquat, ou d'un asservissement par moteur électrique où l'on peut se permettre de mesurer le courant d'induit de moteur en plus de sa vitesse : quelques variables d'état du procédé sont accessibles.

7.1.1 Correcteur dans la chaîne d'action

C'est le cas le plus fréquent, on ne dispose que d'un capteur donnant une image de la sortie. Cette image combinée avec la consigne va nous permettre d'élaborer le signal erreur de la boucle (signal $\varepsilon(t)$). C'est ce signal (éventuellement amplifié et traité) qui sera l'entrée du réseau correcteur. Ce réseau élaborera une commande $u(t)$ que l'on amplifiera en tension et surtout en puissance et appliquera au procédé (amplificateur(s) et actionneurs).

7.1.2 correcteur dans la chaîne de retour

Un exemple classique de correcteur introduit dans la chaîne de retour est la correction tachymétrique d'un asservissement de position : un exemple.

Soit un moteur à courant continu servant au positionnement (sortie $\theta(t)$) d'un objet, ce moteur est alimenté en courant, par un amplificateur de gain k , on dispose de plus en sortie d'une image de la position $\theta(t)$ par potentiomètre par exemple, de même

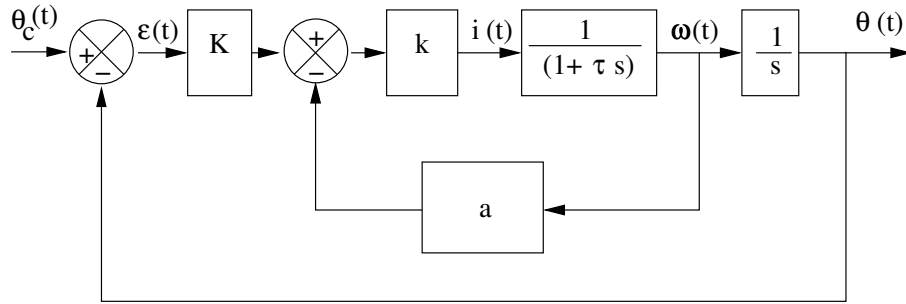


FIGURE 7.1 – Retour tachymétrique

en couplant une génératrice tachymétrique on a une image de la vitesse de rotation $\omega(t)$ de l'arbre moteur. On peut donc réaliser avec ces deux signaux le système bouclé suivant (FIG 7.1); on rajoute un gain K dans la chaîne d'action afin d'avoir deux paramètres de réglage : K et a (taux de retour tachymétrique) : On admettra que la charge mécanique entraînée par le moteur est constituée exclusivement d'inertie J et de frottement visqueux f , alors $\tau = \frac{J}{f}$. D'après cette figure la transmittance du système bouclé vaut :

$$W(s) = \frac{1}{1 + \frac{1+ak}{Kk}s + \frac{\tau}{Kk}s^2}$$

Ce système bouclé est du second ordre avec une pulsation naturelle réglable par K et un amortissement, lui aussi réglable par K et a ; on peut faire du placement de pôles : voir placement de pôles par retour d'état¹.

7.1.3 Assemblage de correcteurs

Pour que l'on puisse honorer le cahier des charges, on devra choisir une structure de réseau correcteur capable de réaliser les actions suivantes :

- Action proportionnelle : translation verticale du lieu de Black.
- Action dérivée : translation plus ou moins horizontale du lieu de Black et ceci vers la droite.
- Action intégrale : remonter vers le haut les points du lieu de Black qui correspondent aux basses fréquences.

Pour réaliser ces contraintes on utilisera des réseaux correcteurs, à action proportionnelle, à avance de phase (action dérivée), à retard de phase (action intégrale) ou des combinaisons de ses réseaux.

1. On peut facilement ici, se fixer le temps de réponse à 5% et par exemple la valeur du premier dépassement de la réponse indicielle, et en déduire les deux paramètres Kk et ak .

7.2 Action proportionnelle

Le modèle du réseau est $RC(s) = K$, son action a pour effet de remonter ($K > 1$) ou descendre ($K < 1$) la courbe de Black de la boucle ouverte : reprenons un exemple de système à la limite de stabilité,

$$sl = \frac{1}{s(1+s)(1+\frac{s}{3})}$$

et avec Scilab traçons : $sl(s)$, $0,5sl(s)$, $2sl(s)$ (FIG 7.2).

```
-->s=%s;den=s*(1+s)*(1+s/3);
-->sl=syslin("c",1/den);
-->[K,P]=kpure(sl)
P =
    1.7320508i
K =
    4.
//je me place à la limite de stabilité
-->sl=K*sl
sl =
           4
-----
           2           3
s + 1.3333333s + 0.3333333s
-->sl05=.5*sl; sl2=2*sl;
-->bblack([sl05;sl;sl2],.1,10)
-->legends(["k=0,5";"k=1";"k=2"],[1,2,3])
```

Nous voyons par cet exemple simple, l'effet d'un gain k sur le lieu de Black. Le même exemple a été traité au paragraphe 5.2.6 en plaçant les pôles dominants sur une droite du plan complexe faisant un angle de 120° par rapport à l'axe horizontal, $4k$ avait une valeur d'environ 0.61 : pour cette valeur vous pouvez maintenant tracer le lieu de Black de la boucle ouverte et déduire les différentes marges.

7.3 Action proportionnelle et dérivée

Le réseau correcteur théorique (une explication sera donnée plus loin), a pour modèle²

$$RC(s) = K(1 + \tau_d s)$$

et en prenant $K = 1$, ce qui ne change pas le raisonnement, on voit apparaître une avance de phase appréciable, (vous tracerez le lieu de Bode d'un réseau pour $\tau_d = 1$, par exemple) mais attention au choix de τ_d . Il faut prendre pour τ_d une valeur supérieure ou proche de l'inverse de la valeur de la pulsation de résonance (en boucle fermée) du système avant correction : $\tau_d > 1/\omega_r$: l'effet d'avance de phase doit se faire

2. Lors de l'étude du réseau P.I.D. on donnera l'explication en question.

7 La correction des systèmes par la méthode de Black

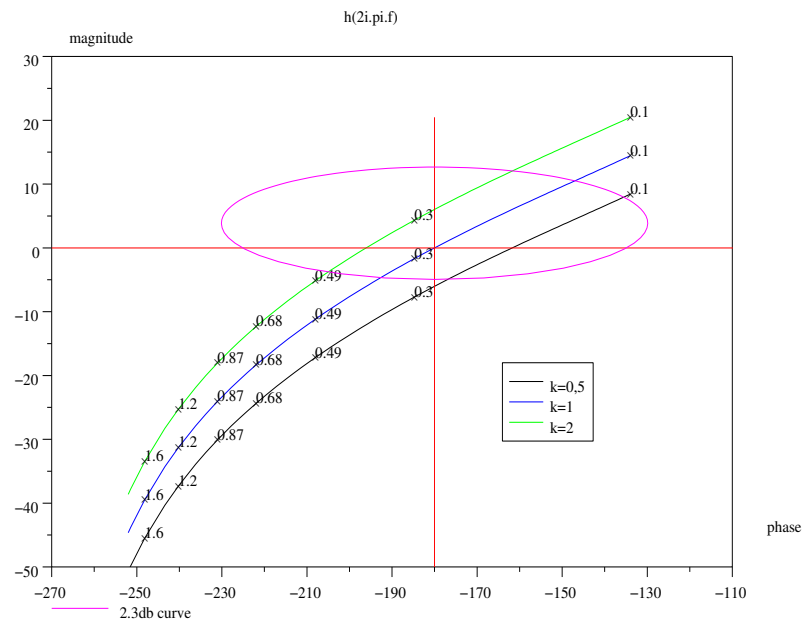


FIGURE 7.2 – Correction proportionnelle

7 La correction des systèmes par la méthode de Black

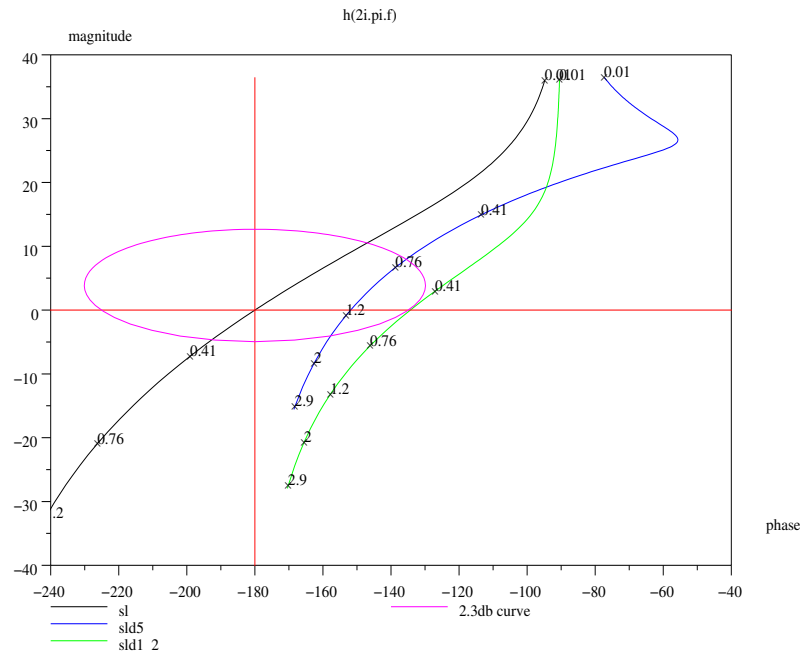


FIGURE 7.3 – Correction avec un réseau P.D.

suffisamment tôt, mais pas trop tôt. La valeur théorique de τ_d est facilement obtenue par le programme Scilab qui continue l'exercice précédent (FIG 7.3).

```
-->taud=1/imag(P)
```

```
taud=
```

```
0.5773503 //valeur théorique que je ne prends pas :
```

```
//je prends approximativement deux et neuf fois cette
```

```
//valeur
```

```
-->sld5=(1+5*s)*sl; sld12=(1+1.2*s)*sl;
```

```
-->bblack([sl;sld5;sld12],.01,3,["tau=0";"tau=5";"tau=1.2"])
```

Reprenons ce même exemple en diminuant fortement τ_d : nous voyons l'effet sur le lieu de Black : j'ai conservé le lieu sans correction, le lieu avec $\tau_d = 1,2$ et avec $\tau_d = 0,2$. Vous pouvez tracer, $sl_{taud}(s) = (1 + \tau_d s)sl(s)$, afin de le comparer par rapport aux exemples précédents et suivants.

```
-->sld02=(1+.2*s)*sl;
```

```
-->bblack([sl;sld02;sld12],.01,3,["tau=0";"tau=0.2";"tau=1.2"])
```

Je vous conseille fortement de réaliser le programme suivant : (suite du même exercice).

```
-->sltaud=sl*(1+taud*s)
```

```
-->bblack([sl;sltaud],.01,3,["sl";"sltaud"])
```

```

-->sltaud2=s1*(1+2*taud*s)
-->bblack([s1;sltaud;sltaud2],.01,3,["s1";"sltaud";"sltaud2"])
//on recherche avec la fonction dbphifr, les facteurs
//de résonance en boucle fermée, les déphasages et
//fréquences correspondantes.
-->[f,d,phi]=dbphifr(sltaud/(1+sltaud),ffreson(sltaud/(1+sltaud)))
-->[f2,d2,phi2]=dbphifr(sltaud2/(1+sltaud2),ffreson(sltaud2/(1+sltaud2)))
//Attention bug ici dans « freson », utiliser « ffreson ».
//puis faire les réponses indicielles des systèmes
//bouclés
-->t=linspace(0,10,101);
-->sbtaud=sltaud/(1+sltaud)
-->y1=csim("step",t,sbtaud);
-->[ymax,indmax]=max(y1)
-->sbtaud2=sltaud2/(1+sltaud2)
-->y2=csim("step",t,sbtaud2);
-->[y2max,ind2max]=max(y2)
-->plot2d(t',[y1;y2]')

```

Cet exercice permet, avec deux choix du paramètre du réseau correcteur, d'étudier les fréquences de résonance et les facteurs de résonance en boucle fermée. Puis, par une simulation temporelle, réponse à un échelon unitaire, de déterminer l'amplitude du premier pic de la réponse ainsi que l'instant (l'indice) auquel ce produit ce premier pic. Vous voyez bien par cette petite simulation, en choisissant une valeur double de la valeur théorique du paramètre du réseau proportionnel et dérivé, que l'on obtient un dépassement de 21,6 %, et un coefficient de surtension Q de presque 2,19 db.

On pourra avec ses deux exemples par les instructions `p_margin()`, `g_margin()` et `m_margin()` vérifier l'augmentation des marges de stabilité. De même, pour un réseau bien centré, on voit le tassement des fréquences vers le haut : augmentation de la bande passante.

Voici les instructions et les résultats.

```

-->sltaud2 //Je désire visualiser la boucle ouverte
sltaud2 =
      4 + 4.6188022s
-----
              2              3
s + 1.3333333s + 0.3333333s
-->[pmarg,fmarg]=p_margin(sltaud2)
fmarg =
0.5012020
pmarg =
45.851219
-->[gmarg,fgmarg]=g_margin(sltaud2)
fgmarg =
[]
gmarg =

```

```

Inf
-->[mmarg,fmmarg]=m_margin(sltaud2)
fmmarg =
    0.6782981
mmarg =
    0.6264024 //Très bonne marge de module
    
```

7.4 Réseau correcteur à avance de phase

Le réseau correcteur à action dérivée étant physiquement irréalisable (présence du dérivateur pur), on utilise de préférence, le réseau à avance de phase. Le modèle de ce réseau est

$$RC(s) = \frac{1 + a\tau s}{1 + \tau s} \text{ avec } a > 1$$

Comme nous l'avons vu en cours, on constate que l'avance de phase maximale est indépendante de τ , et vaut : $\phi_{max} = \arcsin(\frac{a-1}{a+1})$. Cette avance de phase maximale a lieu pour une pulsation : $\omega_a = \frac{1}{\tau\sqrt{a}}$. On pourra à l'aide de Scilab, tracer les différents lieux fréquentiels de ce réseau pour différentes valeurs de a .

7.4.1 Etude théorique du réseau avance de phase (ou retard)

On montre géométriquement que les lieux de Nyquist (ou pseudo-Nyquist) sont des demi cercles passant par les points fixes (1,0) et (a,0).

En effet le réseau se met sous la forme :

$$RC(s) = 1 + (a-1) \frac{1}{1 + \frac{1}{\tau s}} \text{ avec } a > 1$$

pour un réseau à avance de phase ou pour un réseau à retard de phase (paragraphe 7.6) l'on a :

$$RC(s) = \frac{1 + \tau s}{1 + b\tau s} \text{ avec } b > 1$$

soit :

$$RC(s) = 1 - \frac{b-1}{b} \frac{1}{1 + \frac{1}{b\tau s}} \text{ avec } b > 1$$

On montre facilement, qu'après avoir tracé les droites $1 + \frac{1}{\tau s}$ ou $1 + \frac{1}{b\tau s}$ dans le plan complexe, fait l'inversion (l'inverse d'une droite qui ne passe pas par le pôle d'inversion est un cercle), fait les homothéties (facteur $(a-1)$ ou $-\frac{b-1}{b}$) puis la translation de 1, on obtient bien des demi cercles (transformées de demi droites). En ce qui concerne les lieux ou pseudo-lieux du circuit à avance de phase, ils passent par deux points fixes (1,0) et (a,0). Le raisonnement reste valable pour le circuit à retard de phase.

Pour tracer les différents lieux de ce réseau à avance de phase, je prends $\tau = 1$ ce qui ne change rien pour l'allure des lieux : voici un programme permettant de voir, pour différentes valeurs de a , le lieu de bode de ce réseau.

```
-->a=[2;4;6;8;10;12];s=%s;RC=syslin("c",(ones(a)+a*s)./(1+s));
-->com=["a=2";"a=4";"a=6";"a=8";"a=10";"a=12"];
-->bbode(RC,.002,1.4)
-->legends(com,[1,2,3,4,5,6]);
```

7.4.2 Exemple de synthèse

Voici un exemple de synthèse de ce type de réseau correcteur (genre TP d'illustration).

On veut réaliser la synthèse d'un système à retour unitaire, dont la chaîne d'action est constituée d'un amplificateur de gain k , et d'une transmittance

$$G(s) = \frac{1}{s(1+s)(1+s/3)}$$

comme le système est de classe 1, on cherche k pour avoir une erreur en vitesse de 40 %; par Scilab on cherche à déterminer la marge de phase, de gain, la pulsation de résonance de même que le facteur de résonance en boucle fermée (pour la valeur du gain trouvée). On fera une simulation des réponses impulsionnelle, indicielle, puis fréquentielle : on aura préalablement vérifié que le système bouclé est stable et trouvé la valeur du gain limite k_l et la pulsation d'oscillation ω_l pour cette valeur limite du gain.

```
-->s=%s;g=syslin("c",1/(.4*s*(1+s)*(1+s/3)));
// 1/.4 est la bonne valeur voir paragraphe 6.2.1 (1/K=.4)
-->bblack(g,.1,10,"g")// lieu de Black
-->[mp,fp]=p_margin(g)//marge de phase, fréquence correspondante.
fp =
    0.2154724
mp =
    12.161872
-->[mg,fg]=g_margin(g)//marge de gain, fréquence correspondante.
fg =
    0.2756644
mg =
    4.0823997
-->[mfp,fgp]=m_margin(g) //marge de module, fréquence correspondante.
fgp =
    0.2269988
mfp =
    0.1838880 //beaucoup trop faible !!!
-->gbf=g/(1+g);//La boucle fermée.
-->facb=bodfact(g/(1+g))//gain statique de la boucle fermée.
facb =
    1.
-->fres=ffreson(gbf)//fréquence de résonance en boucle fermée.
```

7 La correction des systèmes par la méthode de Black

```
fres =
    0.2233085
-->[factres,phifres]=dbphifr(gbf,fres)
//facteur de résonance en boucle fermée, le gain statique en bf vaut 1.
phifres =
    - 104.51467 //phase correspondante.
factres =
    14.009724 //facteur de résonance en décibels.
-->[kl,pl]=kpure(g)
pl =
    1.7320508i//pôle sur axe imaginaire (omegi = 1.732 rd/s).
kl =
    1.6 //gain limite qui est > au gain choisi (ici 1.25).
```

Je ne fais pas les tracés des réponses impulsionnelle, indicielle, fréquentielle (voir paragraphe 7.3).

On place en cascade avec la chaîne d'action un réseau correcteur à avance de phase de transmittance

$$RC(s) = \frac{1 + a\tau s}{1 + \tau s} \text{ avec } a > 1$$

et l'on cherche à rajouter, pour la valeur de k précédemment calculée, une phase supplémentaire de l'ordre de 55° , donnez la valeur de a . Initialisez le réseau, c'est à dire, trouvez une première valeur à τ . On remarquera que pour une pulsation valant $\omega_a = \frac{1}{\tau\sqrt{a}}$ le réseau a un gain de $10\log(a)^3$. On tracera les lieux de Bode du réseau correcteur, puis de la boucle ouverte corrigée.

Tracez sur le même graphique, pour la valeur de k choisie, les lieux de Black de la boucle ouverte du système non corrigé et du système corrigé.

Le réseau étant maintenant identifié, faire une étude temporelle de ce système bouclé : réponse indicielle ; de même faire l'étude temporelle du signal de commande $u(t)$, sortie du réseau correcteur, pour l'entrée précédente.

Comme ce réseau a pour propriétés de tasser les fréquences vers le haut, pour choisir a on fera : $a = \frac{1+\sin(\phi_{max})}{1-\sin(\phi_{max})}$ et on choisira $\tau = \frac{1}{\omega_c\sqrt{a}}$ avec ω_c une pulsation au moins le double ou le triple de la pulsation de résonance. Voici le programme.

```
-->g
g =
    1
-----
    2          3
0.4s + 0.5333333s + 0.1333333s
-->alp=sin(%pi/180)*55)//on demande une augmentation de phase de 55°.
alp =
    0.8191520
```

3. En se plaçant à cette pulsation sur le lieu de Black de la boucle ouverte non corrigée, le point correspondant se déplace de ϕ_{max} vers la droite et $10\log(a)$ vers le haut, quand on rajoute le correcteur.

7 La correction des systèmes par la méthode de Black

```

-->a=(1+alp)/(1-alp)//
a = 10.059014
-->10*log(a)/log(10) //valeur de a en db.
ans =
10.025554
-->tau=1/((sqrt(a))*2*pi*fres*2.5);//deux fois et demi fres.
-->RC=(1+a*tau*s)/(1+tau*s);grc=g*RC
-->bblack([g;grc],.1,10,["g";"grc"])
//faites un zoom autour du point (0 db,-180°) et affinons.
-->tau1=1/((sqrt(a))*2*pi*fres*2.6);//c'est la bonne valeur.
-->RC1=(1+a*tau1*s)/(1+tau1*s);
-->grc1=g*RC1;
-->bblack([g;grc;grc1],.1,1,["g";"grc";"grc1"])
-->[mg,fg]=g_margin(grc1)
fg =
    0.9086634
mg =
    15.941468
-->[mp,fp]=p_margin(grc1)
fp =
    0.2993909
mp =
    45.227004//on a bien la bonne marge de phase.
-->[mgprc1,fgprc1]=m_margin(grc1) //marge de module;Attention m_margin
utilise ffreson !
fgprc1 =
    0.4253548
mgprc1 =
    0.5887506 //Elle a augmenté de manière importante.
-->fres=ffreson(grc1/(1+grc1)) //fréquence résonance en BF.
fres =
    0.3011005
-->[factbf,phibf]=dbphifr(grc1/(1+grc1),fres)
phibf =
    - 68.016347
factbf =
    2.2815257 //facteur de résonance.

```

Quant aux différentes réponses vous voudrez bien regarder les chapitres précédents (voir paragraphe 7.3).

7.5 Réseau correcteur à action proportionnelle et intégrale

Comme nous l'avons vu précédemment, l'introduction d'un intégrateur dans la chaîne d'action permet d'améliorer la précision en régime statique, (section 6.2.1) mais ce résultat est au détriment de la stabilité. En effet un intégrateur pur introduit un déphasage de -90° quelque soit la fréquence. Pour cette raison on préfère introduire un réseau de type proportionnel et intégral (P.I.).

Ce correcteur a pour transmittance :

$$RC(s) = 1 + \frac{1}{\tau_i s}$$

par une étude des courbes de Bode de ce réseau on peut remarquer qu'aux hautes fréquences ($\omega > \frac{1}{\tau_i}$) ce correcteur n'introduit plus de déphasage et ne change pas le gain du système quand on met en cascade ce réseau et la boucle ouverte du système à étudier. Afin d'éviter l'effet déstabilisant de l'intégrateur, on prendra $\frac{1}{\tau_i} \ll \omega_r$. Cette pulsation ω_r étant la pulsation de résonance de la boucle fermée avant introduction du réseau.

Par un réglage satisfaisant, ni la pulsation de résonance ω_r ni le facteur de résonance Q ne sont modifiés, seule la précision statique est améliorée.

Voici un exemple de détermination d'un réseau P.I., la boucle ouverte a pour transmittance

$$G(s) = \frac{1}{s(1+s)(1+\frac{s}{4})}$$

```
-->s=%s;num=1;den=s*(1+s)*(1+s/4);sl=syslin("c",num/den)
```

```
sl =
```

```
1
```

```
-----
```

```
2
```

```
3
```

```
s + 1.25s + 0.25s
```

Vous pouvez, à ce stade tracer la courbe de Black de `sl` et déterminer la fréquence de résonance, le gain et la phase correspondante, de la boucle fermée par l'instruction : `dbphifr()`.

```
-->sb=sl/(1+sl);
```

```
-->[frb,db,phi]=dbphifr(sb,ffreson(sb))
```

```
phi =
```

```
- 76.236457
```

```
db =
```

```
3.0914789
```

```
frb =
```

```
0.1299495
```

On détermine ici la transmittance de la boucle fermée et par instruction `dbphifr()`, nous obtenons la fréquence de résonance `frb` et le facteur de résonance : on peut ainsi déterminer une première valeur pour τ_i soit : `tor`.

7 La correction des systèmes par la méthode de Black

```
-->tor=1/(2*pi*frb)
tor =
1.2247449
```

On peut maintenant, pour plusieurs valeurs de τ_i construire différents réseaux correcteurs : par exemple,

$$RC(s) = 1 + \frac{1}{t_{or}s}; \quad RC(s) = 1 + \frac{1}{0,1t_{or}s}; \quad RC(s) = 1 + \frac{1}{5t_{or}s}$$

pour voir l'influence de τ_i sur les caractéristiques de la boucle fermée.

```
-->rctor=syslin("c",1+1/(tor*s))
rctor =
1 + 1.2247449s
-----
1.2247449s
-->sltorsl*rctor
sltorsl =
1 + 1.2247449s
-----
2          3          4
1.2247449s + 1.5309311s + 0.3061862s
-->rc01tor=syslin("c",1+1/(.1*tor*s))
rc01torsl =
1 + 0.1224745s
-----
0.1224745s
-->sl01torsl*rc01tor
sl01torsl =
1 + 0.1224745s
-----
2          3          4
0.1224745s + 0.1530931s + 0.0306186s
```

D'une manière analogue, on peut construire par exemple `sl5tor`, `sl10tor` et avec la fonction `bblack()` afficher sur un même graphe les cinq lieux de Black.

```
-->sl5torsl*(1+1/(5*tor*s));
-->sl10torsl*(1+1/(10*tor*s));
-->com=["sl","sltorsl","sl01torsl","sl5tor","sl10tor"];
-->bblack([sl;sltorsl;sl01torsl;sl5tor;sl10tor],.001,.3,com)
```

Ce graphe (FIG 7.4) est suffisamment clair et met bien en évidence que pour $\tau_i = \frac{1}{\omega_r}$, on a rendu instable un système qui était de toute façon stable à l'origine (courbes `sltorsl`, `sl01torsl`). La valeur de τ_i est beaucoup trop faible, et une valeur normale de τ_i doit être de l'ordre de 5 à 10 fois $\frac{1}{\omega_r}$. On remarquera de même que l'adjonction d'un réseau correcteur bien placé, permet, tout en conservant des marges de stabilité quasiment identiques, par l'introduction dans la boucle ouverte d'un intégrateur, d'avoir,

7 La correction des systèmes par la méthode de Black

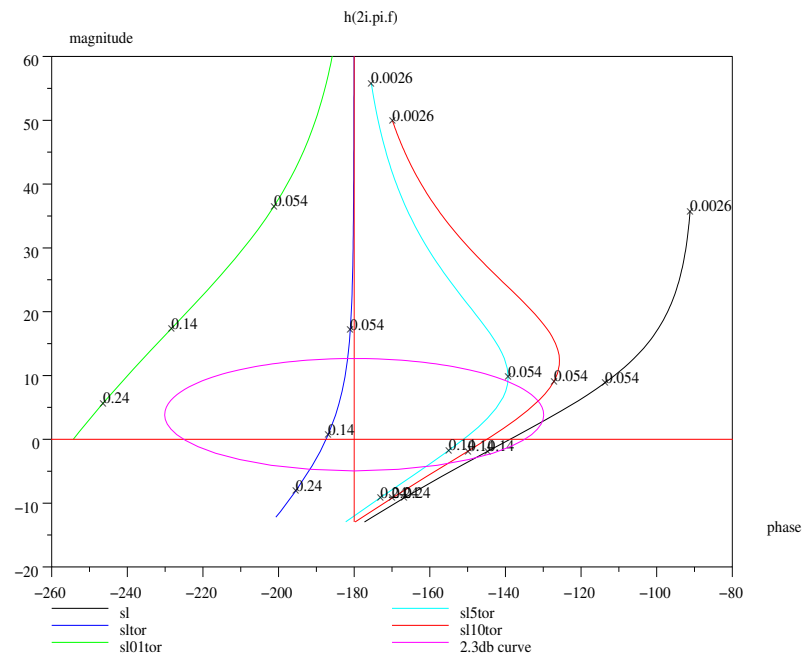


FIGURE 7.4 – Correction avec un réseau P.I.

dans ce cas, non seulement une erreur permanente nulle pour une entrée en échelon, mais d'avoir aussi une erreur permanente nulle pour une entrée en rampe : le système est de classe deux.

Ce réseau améliore, et comment ! la précision d'un système bouclé.

7.6 Réseau correcteur à retard de phase

Ce réseau est une approche du réseau de type P.I., et s'il n'introduit pas d'intégration dans la chaîne d'action il peut, s'il est bien placé, en conservant une bonne marge de stabilité, permettre d'augmenter le gain de la chaîne d'action et ainsi, d'augmenter sérieusement la précision en régime statique. Son modèle mathématique est :

$$RC(s) = \frac{1 + \tau s}{1 + b\tau s} \text{ avec } b > 1$$

On pourra avec Scilab très simplement étudier les courbes de Nyquist, Bode et Black de ce réseau pour différentes valeurs de b (en prenant $\tau = 1$ le produit $\tau\omega$ étant un nombre sans dimension). Comme précédemment on placera le nombre $\frac{1}{\tau} \ll \omega_r$. Le seul effet déplaisant de ce réseau est la diminution de la bande passante : voici un exemple avec Scilab de placement d'un tel réseau (FIG. 7.5).

```
-->s=%s;sl=syslin("c",1/(s*(1+s)*(1+s/4)));
-->sb=sl/(1+sl);
-->[frb,db,phi]=dbphifr(sb,ffreson(sb))
phi =
- 76.236457
db =
3.0914789
frb =
0.1299495
-->tor=1/(2*pi*frb)
tor =
1.2247449
-->rctor=syslin("c",(1+tor*s)/(1+10*tor*s))
rctor =
1 + 1.2247449s
-----
1 + 12.247449s
-->sltor=sl*rctor;
-->rc02tor=syslin("c",(1+0.2*tor*s)/(1+10*0.2*tor*s));
-->sl02tor=sl*rc02tor;
-->rc5tor=syslin("c",(1+5*tor*s)/(1+10*5*tor*s));
-->sl5tor=sl*rc5tor;
-->rc10tor=syslin("c",(1+10*tor*s)/(1+10*10*tor*s));
-->sl10tor=sl*rc10tor;
-->com=["sl","sltor","sl02tor","sl5tor","sl10tor"];
```

7 La correction des systèmes par la méthode de Black

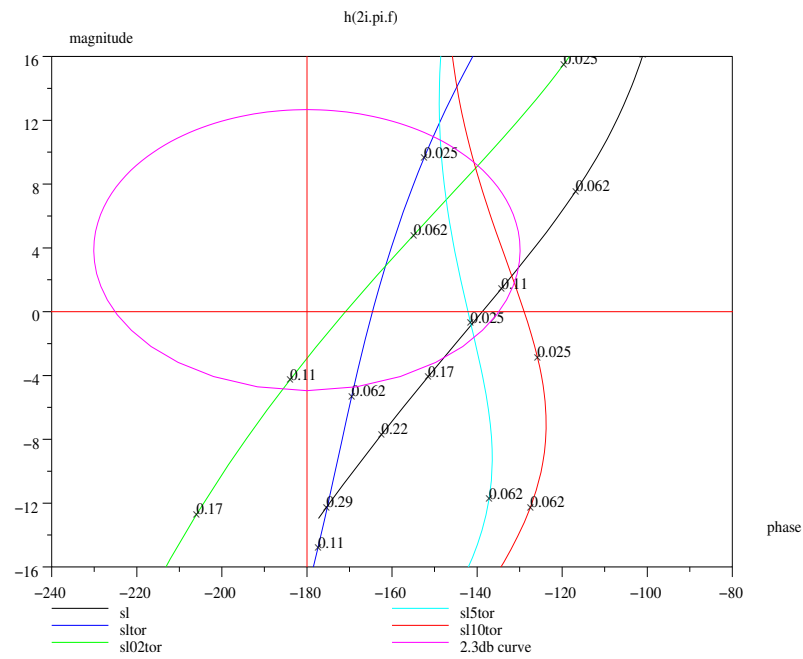


FIGURE 7.5 – Correction avec un retard de phase

7 La correction des systèmes par la méthode de Black

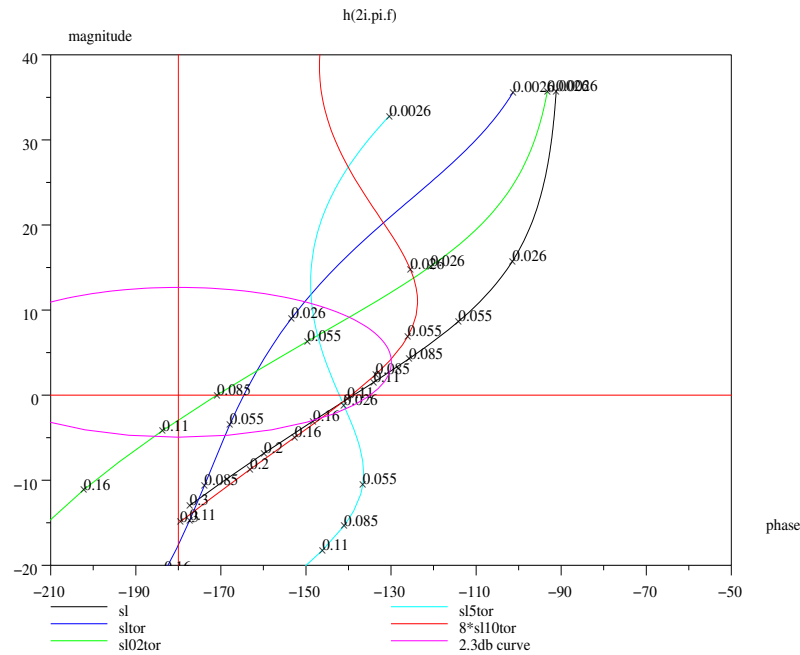


FIGURE 7.6 – Correction avec retard de phase et gain

```
-->bblack([sl;sltor;sl02tor;sl5tor;sl10tor],.001,.3,com)
```

Dans cet exemple nous voyons que l'adjonction d'un réseau à retard de phase avec $\tau = \frac{1}{\omega_r}$ rend le système nettement moins stable : courbe `sltor`, afin de montrer l'influence de la valeur de τ , j'ai fait tracer à Scilab le lieu de Black un système `sl02tor` dont la valeur de τ est cinq fois plus faible : on rend encore moins stable la boucle fermée. Par contre en donnant à τ une valeur dix fois plus importante, et **augmentant le gain de la chaîne d'action d'un facteur huit**, on conserve le même degré de stabilité tout en augmentant la précision en régime statique, pour une entrée en rampe, de ce facteur huit.

Le réseau améliore, s'il est bien placé, la précision du système bouclé.

Voici les deux instructions qui permettent de tracer le bon lieu de Black (FIG 7.6).

```
-->com=["sl","sltor","sl02tor","sl5tor","8*sl10tor"];
```

```
-->clf();
```

```
-->bblack([sl;sltor;sl02tor;sl5tor;8*sl10tor],.001,.3,com)
```

7.7 Réglage d'un P.I.D. par la méthode du pivot

Parmi tous les types de réseaux correcteurs utilisés industriellement, le réseau de type P.I.D. est sans doute encore, et pour longtemps, le plus utilisé. Son modèle ma-

7 La correction des systèmes par la méthode de Black

thématique théorique est :

$$RC(s) = K(1 + \frac{1}{\tau_i s} + \tau_d s)$$

En réalité, le terme dérivé, dans sa réalisation pratique étant impossible, a pour modèle le facteur suivant :

$$\frac{\tau_d s}{1 + a\tau_d s}$$

avec a prenant une valeur proche de un dixième (les deux points de cassure de ce terme séparés d'une décade au moins).

Quand on fait l'étude théorique de ce réseau, par exemple en traçant son lieu de Bode, on remarque qu'il existe une pulsation particulière, $\omega_p = \frac{1}{\sqrt{\tau_i \tau_d}}$ où le réseau n'avance ni ne retarde la phase, en ce point, le gain est de 0 db.

C'est ce point, qui une fois convenablement choisi, permettra de basculer le lieu de Black non corrigé, vers la gauche pour des pulsations inférieures à ω_p , et vers la droite pour des pulsations supérieures à ce pivot : en effet par une étude des lieux de Bode du P.I.D., on montre facilement que la phase de ce réseau est négative (comprise entre -90° et 0°) pour des fréquences inférieures à $\frac{\omega_p}{2\pi}$ et est positive (comprise entre 0° et 90°) pour des fréquences supérieures à cette valeur. Quant au gain, il décroît jusqu'à 0 db (pour $\omega = \omega_p$) puis recroît quand la fréquence augmente. Vous ferez, avec Scilab l'étude du lieu de Bode du P.I.D. théorique pour vérifier ces remarques.

Le choix du point de pivotement est donc très important, en effet l'introduction du réseau P.I.D a pour but de :

- Améliorer la précision en régime statique (introduction de l'intégrateur) sans pour cela réduire la stabilité.
- Améliorer la stabilité : marge de phase, de gain, amortissement Q .

Quant au choix du point de pivotement, si on souhaite un amortissement élevé, ordre de grandeur 0,7, on devra prendre un point au dessus de l'axe 0 db, légèrement à gauche de la verticale -90° (20° à gauche). Afin de rendre la méthode plus simple dans une première approche, on fixera $\tau_i = 4\tau_d$: avec cette valeur, **le P.I.D. a un zéro double**. Si on exprime la transmittance du réseau en fonction de ω_p avec $\tau_i = 4\tau_d$ on obtient :

$$RC(p) = K(1 + \frac{\omega_p}{2s} + \frac{s}{2\omega_p})$$

Voici un exemple de session Scilab mettant en évidence ces considérations.

```
-->s=%s;s1=syslin("c",1/(s*(1+s)*(1+s/4)));
-->[k1,p1]=kpure(s1)
p1 =
    2.i
k1 =
    5.
-->bblack(s1,.01,.4)
```

Je ne mets pas dans le texte le lieu de Black : il a déjà été tracé à l'exercice précédent, (courbe s1). On remarque qu'une fréquence de l'ordre de 0,05 hertz correspond aux

7 La correction des systèmes par la méthode de Black

remarques faites précédemment. En première approche on a donc $\tau_i = \frac{1}{0,05s}$. Le réseau a donc pour modèle, ($\tau_i = 4\tau_d$) :

$$RC(s) = 1 + \frac{1}{6,7s} + 1,59s$$

```
-->toi=1/(%pi*.05)
//toi=2/(2*%pi*fp)
-->tod=toi/4
//tod=1/(2*(2*%pi*fp))
-->rc=syslin("c", (1+toi*s+toi*tod*s*s)/(toi*s))
rc =
                2
1 + 6.3661977s + 10.132118s
-----
        6.3661977s
-->slrc=sl*rc
slrc =
                2
1 + 6.3661977s + 10.132118s
-----
                2          3          4
6.3661977s + 7.9577472s + 1.5915494s
-->bblack([sl;slrc], .01, .4, ["sl", "slrc"])
Nous voyons, par cet exemple très simple, l'intérêt de cette méthode (FIG 7.7). J'ai
pris ici une fréquence correspondant à une phase en boucle ouverte de l'ordre de -110°.
On pouvait par le programme qui suit affiner ce point de pivotement.
-->s=%s;sl=syslin("c", 1/(s*(1+s)*(1+s/4)));
-->[f,d,p]=dbphifr(sl, .04, .1, .005);
-->omegp=2*%pi*(f(find(p<-109&p>-111)))
omegp =
column 1 to 5
! 0.2724205 0.2755750 0.2787660 0.2819940 0.2852593 !
column 6 to 9
! 0.2885625 0.2919039 0.2952840 0.2987032 !
-->omegp=mean(omegp)
//On prend la valeur moyenne
omegp =
0.2840349
-->rc=syslin("c", (1+1/((2/omegp)*s)+(1/(2*omegp))*s))
rc =
                2
1 + 7.0413878s + 12.395285s
-----
        7.0413878s
```

7 La correction des systèmes par la méthode de Black

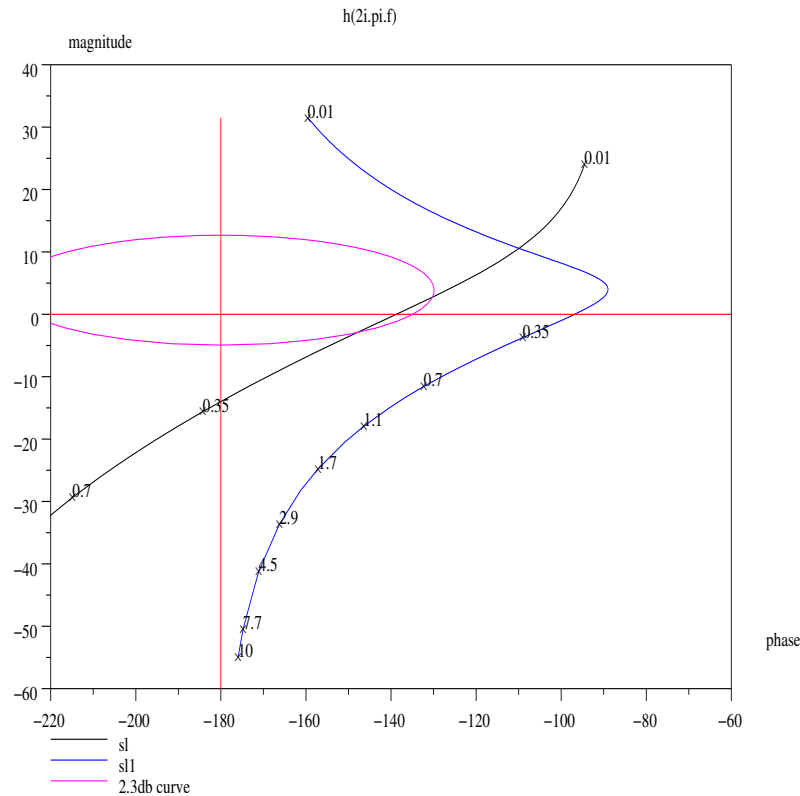


FIGURE 7.7 – Correction par la méthode du pivot

```
-->s11=rc*s1
s11 =
          2
1 + 7.0413878s + 12.395285s
-----
          2          3          4
7.0413878s + 8.8017347s + 1.7603469s
-->bblack([s1;s11],.01,10,["s1","s11"])
```

Cette méthode est beaucoup plus précise pour le choix du point de pivotement, on obtient ce point par les deux instructions des lignes trois et quatre du programme.

Cette fois, en augmentant le gain de la chaîne d'action et en nous plaçant à la limite de stabilité pour le système sans réseau correcteur ($k_l = 5$ voir le début de l'exercice avec $\omega_l = 2 \text{ rad/s}$), par la même méthode, en choisissant pour fréquence de pivot $f_p = 0,08 \text{ hertz}$ on obtient les résultats suivants :

```
-->s1=5*s1
```


7 La correction des systèmes par la méthode de Black

```

sl =
      5
-----
      2      3
s + 1.25s + 0.25s
-->toi=1/(%pi*.08)
toi =
3.9788736
-->tod=toi/4
tod =
0.9947184
-->rc=syslin("c",(1+toi*s+toi*tod*s*s)/(toi*s))
rc =
      2
1 + 3.9788736s + 3.9578587s
-----
      3.9788736s
-->slrc=sl*rc
slrc =
      2
5 + 19.894368s + 19.789294s
-----
      2      3      4
3.9788736s + 4.973592s + 0.9947184s
-->bblack([sl;slrc],.05,1,["sl","slrc"])
-->[fr,db,phi]=dbphifr(slrc/(1+slrc),ffreson(slrc/(1+slrc)))
phi =
- 59.703174
db =
1.9266743
fr =
0.5401616

```

Nous voyons facilement qu'avec ce réseau (FIG 7.7), on a stabilisé le système (le facteur de résonance est maintenant très proche de 2,3 *db*, en réalité 1,93 *db*) et bien sur amélioré la précision : la boucle ouverte possède deux intégrateurs. On peut même, maintenant se permettre d'augmenter légèrement le gain de la chaîne d'action afin d'avoir un facteur de résonance de 2,3 *db*.

Je voudrais faire ici une remarque sur cette **méthode du pivot** et la **méthode de synthèse dite méthode de Ziegler et Nichols**. Par la méthode de Ziegler et Nichols, quand on travaille en boucle ouverte, on détermine expérimentalement un modèle de la boucle ouverte [4, pages 245-247], modèle caractérisé par deux paramètres : a et T_r et c'est à partir de ces deux paramètres que l'on détermine τ_i avec $\tau_i = 4\tau_d$. On remarque facilement que si $\omega_p = \frac{1}{\sqrt{\tau_i\tau_d}}$ est la pulsation du pivot on a : $\omega_p = \frac{1}{T_r}$,

7 La correction des systèmes par la méthode de Black

en effet on a $\tau_d = \frac{1}{2\omega_p}$ et $\tau_i = \frac{2}{\omega_p}$ par la méthode du pivot et l'on obtient ces mêmes valeurs par le critère de Ziegler et Nichols, en ayant $\omega_p = \frac{1}{T_r}$.

Peut-on faire un choix optimal de la pulsation du pivot ? On penserait normalement que le réglage du point de pivotement peut conduire à des résultats différents, non seulement sur la transmittance du réseau, mais surtout sur les caractéristiques dynamiques du système bouclé. Réalisons l'expérience suivante : en prenant la boucle ouverte précédente, avec un système à la limite de la stabilité, en choisissant différents points de pivotement pour des déphasages de -120° , -125° , -130° , -135° , on obtient les lieux de Black :

```
-->omegp=2*%pi*mean((f(find(p<-119&p>-121))))
omegp =
0.4398403
-->omegp1=2*%pi*mean((f(find(p<-124&p>-126))))
omegp1 =
0.5226819
-->omegp2=2*%pi*mean((f(find(p<-129&p>-131))))
omegp2 =
0.6070681
-->omegp3=2*%pi*mean((f(find(p<-134&p>-136))))
omegp3 =
0.7009965
-->rc=1+omegp/(2*s)+s/(2*omegp)
rc =
                2
0.4398403 + 2s + 2.2735526s
-----
                2s
-->rc1=1+omegp1/(2*s)+s/(2*omegp1)
rc1 =
                2
0.5226819 + 2s + 1.9132097s
-----
                2s
-->rc2=1+omegp2/(2*s)+s/(2*omegp2)
rc2 =
-->omegp=2*%pi*mean((f(find(p<-119&p>-121))))
omegp =
0.4398403
-->omegp1=2*%pi*mean((f(find(p<-124&p>-126))))
omegp1 =
0.5226819
-->omegp2=2*%pi*mean((f(find(p<-129&p>-131))))
omegp2 =
0.6070681
```

7 La correction des systèmes par la méthode de Black

```

-->omegp3=2*%pi*mean((f(find(p<-134&p>-136))))
omegp3 =
0.7009965
-->rc=1+omegp/(2*s)+s/(2*omegp)
rc =

$$\frac{0.4398403 + 2s + 2.2735526s^2}{2s}$$

-->rc1=1+omegp1/(2*s)+s/(2*omegp1)
rc1 =

$$\frac{0.5226819 + 2s + 1.9132097s^2}{2s}$$

-->rc2=1+omegp2/(2*s)+s/(2*omegp2)
rc2 =

$$\frac{0.6070681 + 2s + 1.6472616s^2}{2s}$$

-->rc3=1+omegp3/(2*s)+s/(2*omegp3)
rc3 =

$$\frac{0.7009965 + 2s + 1.4265407s^2}{2s}$$

-->SL=s1*[1;rc;rc1;rc2;rc3];
-->com=["bo";"piv120";"piv125";"piv130";"piv135"];
-->bblack(SL,.1,1,com)

$$\frac{0.6070681 + 2s + 1.6472616s^2}{2s}$$

-->rc3=1+omegp3/(2*s)+s/(2*omegp3)
rc3 =

$$\frac{0.7009965 + 2s + 1.4265407s^2}{2s}$$

-->SL=s1*[1;rc;rc1;rc2;rc3];
-->com=["bo";"piv120";"piv125";"piv130";"piv135"];
-->bblack(SL,.1,1,com)

```

7 La correction des systèmes par la méthode de Black

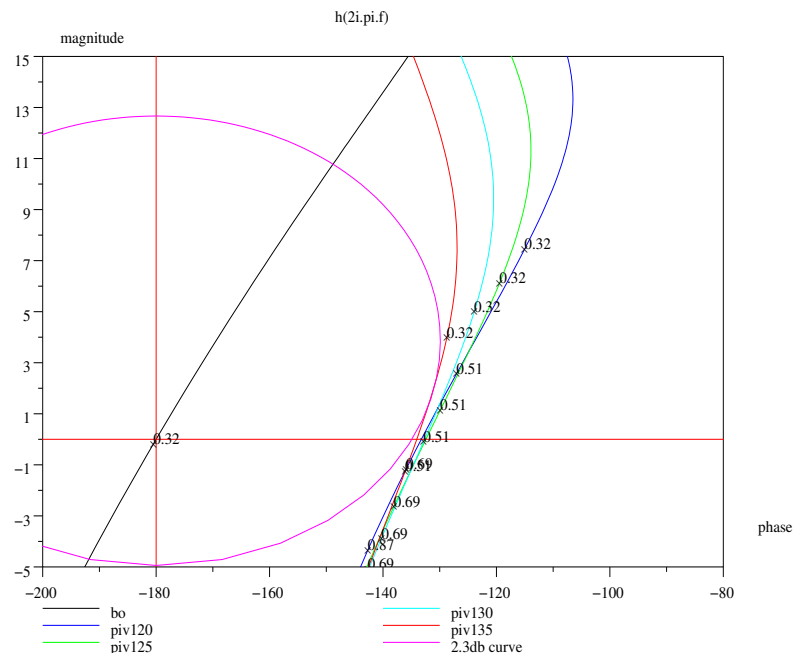


FIGURE 7.8 – Influence du choix du pivot

Nous voyons sur cet exemple que le point de pivotement n'a pas une influence prépondérante sur le résultat (FIG 7.8). Comment le prouver ?, attention au fait qu'avec le P.I.D. on peut simplifier un pôle du système avec un zéro du correcteur.

7.8 Réglage d'un réseau retard-avance de phase

Après avoir étudié le réseau P.I.D., il est tout naturel introduire le réseau retard-avance de phase qui tout en ayant des effets comparables, peut être plus facilement réalisé (en électrique on le réalise facilement avec des quadripôles).

Je rappelle que le but de ce type de réseau est de faire un retard de phase aux basses fréquences, et une avance de phase aux moyennes fréquences. De même on cherchera à augmenter le gain de la chaîne d'action afin d'améliorer la précision.

Comme c'est la mise en série d'un réseau à retard de phase et d'un réseau à avance de phase, son modèle mathématique est donc :

$$RC(p) = \frac{(1 + \tau_2 s)(1 + \tau_3 s)}{(1 + \tau_1 s)(1 + \tau_4 s)}$$

Si on veut que le gain en hautes fréquences reste égal à 1, on doit avoir la relation : $\tau_1 \tau_4 = \tau_2 \tau_3$. De même on cherchera à avoir une avance de phase la plus grande possible, en restant dans des limites raisonnables (influence de la dérivation sur les signaux bruités), il est donc normal de prendre le rapport des constantes de temps égal à environ 10 (avance de phase maximale de l'ordre de 55° : voir la section 7.4).

On fera de même avec le réseau à retard de phase, dans ces conditions on aura une courbe de Bode de ce réseau, pour le gain symétrique par rapport à $\frac{1}{\sqrt{\tau_2 \tau_3}}$ avec $\tau_1 \tau_4 = \tau_2 \tau_3$, et une courbe de phase symétrique par rapport à ce point (en ce point le réseau n'introduit pas de déphasage, mais contrairement au P.I.D, il introduit une atténuation) :

```
-->s=%s;rc=syslin("c",((1+s)*(1+3*s))/((1+.1*s)*(1+30*s)))
rc =
```

$$\frac{1 + 4s + 3s^2}{1 + 30.1s + 3s^2}$$

Tout en gardant le même point de symétrie, traçons un autre réseau où l'on remplace, τ_2 en $2\tau_2$, τ_3 en $\frac{\tau_3}{2}$ etc..., on a donc (FIG 7.9) :

```
-->rc1=syslin("c",((1+2*s)*(1+1.5*s))/((1+.2*s)*(1+15*s)))
rc1 =
```

$$\frac{1 + 3.5s + 3s^2}{1 + 15.2s + 3s^2}$$

7 La correction des systèmes par la méthode de Black

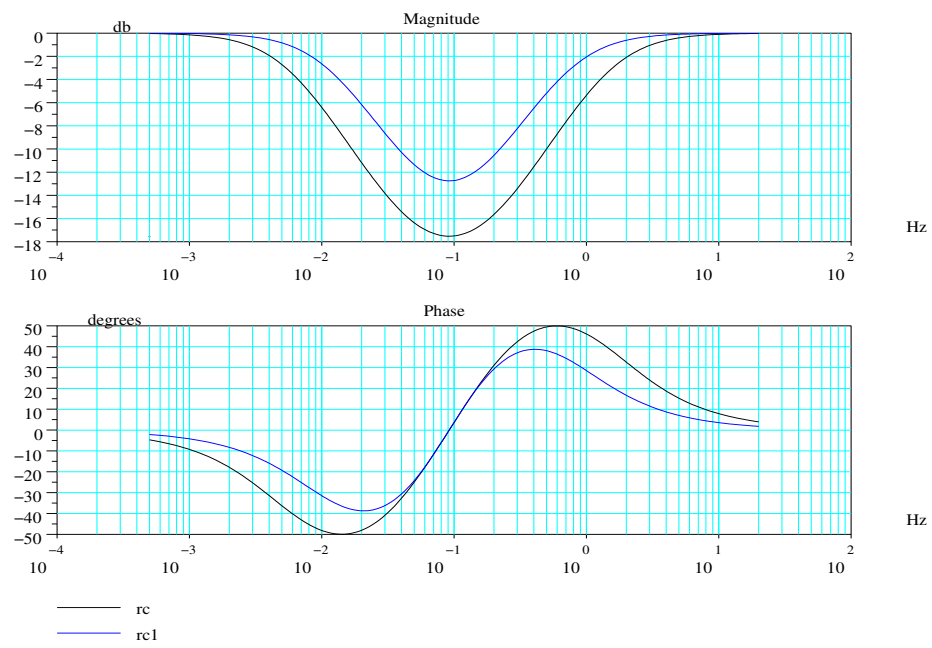


FIGURE 7.9 – Correcteur retard-avance

```
-->bode([rc;rc1],.0005,20,["rc","rc1"])
```

A partir des remarques précédentes, on choisit un point, sur la courbe de Black de la boucle ouverte où le déphasage ne changera pas par l'introduction du réseau, puis, on cherche une pulsation où l'on doit appliquer l'avance de phase maximale, cette valeur vaut pratiquement $\frac{1}{\sqrt{\tau_3\tau_4}}$, car à cette pulsation le réseau à retard de phase n'agit que très peu. Bien sur par symétrie, le minimum de déphasage introduit par le réseau retard de phase se produit à une pulsation valant $\frac{1}{\sqrt{\tau_1\tau_2}}$.

Appelons f_1 , f_2 , f_3 les fréquences où les déphasages introduits par le réseau sont respectivement : minimum, nul et maximum on a donc :

$$\sqrt{\tau_2\tau_3} = \frac{1}{2\pi f_2}, \quad \sqrt{\tau_3\tau_4} = \frac{1}{2\pi f_3}, \quad \text{et } \tau_1 = 10\tau_2, \quad \tau_3 = 10\tau_4$$

En prenant les logarithmes de ses fonctions on obtient un système de quatre équations à quatre inconnues.

Dans l'exemple choisi, si $f_2 = 0,1$ hertz, point correspondant à un déphasage en boucle ouverte de -120° environ (pseudo pivot), si l'on prend f_3 aux environs de 1 hertz (endroit où l'on désire l'avance de phase maximale), en utilisant la méthode proposée, on trouve, l'ensemble des valeurs de τ . Vérifions ceci avec Scilab (FIG 7.10).

```
-->s=%s;sl=syslin("c",5/(s*(1+s)*(1+s/4)));
-->bblack(sl,.08,1.5)
-->A=[1,-1,0,0;0,1,1,0;0,0,1,-1;0,0,1,1];
-->C=[log(10);-2*log(2*pi*.1);log(10);-2*log(2*pi*1)];
Les logarithmes des constantes de temps X vérifient l'équation : AX = C.
-->X=inv(A)*C;
-->tau=exp(X)
tau =
! 50.329212 !
! 5.0329212 !
! 0.5032921 !
! 0.0503292 !
-->num=(1+tau(2,1)*s)*(1+tau(3,1)*s);
-->den=(1+tau(1,1)*s)*(1+tau(4,1)*s);
-->rc=syslin("c",num/den)
rc =
2
1 + 5.5362133s + 2.5330296s
-----
2
1 + 50.379541s + 2.5330296s
-->slrc=sl*rc;
-->bblack([sl;slrc],.01,1.5,["s1","slrc"])
-->mg=g_margin(slrc)
mg =
35.795429
```

7 La correction des systèmes par la méthode de Black

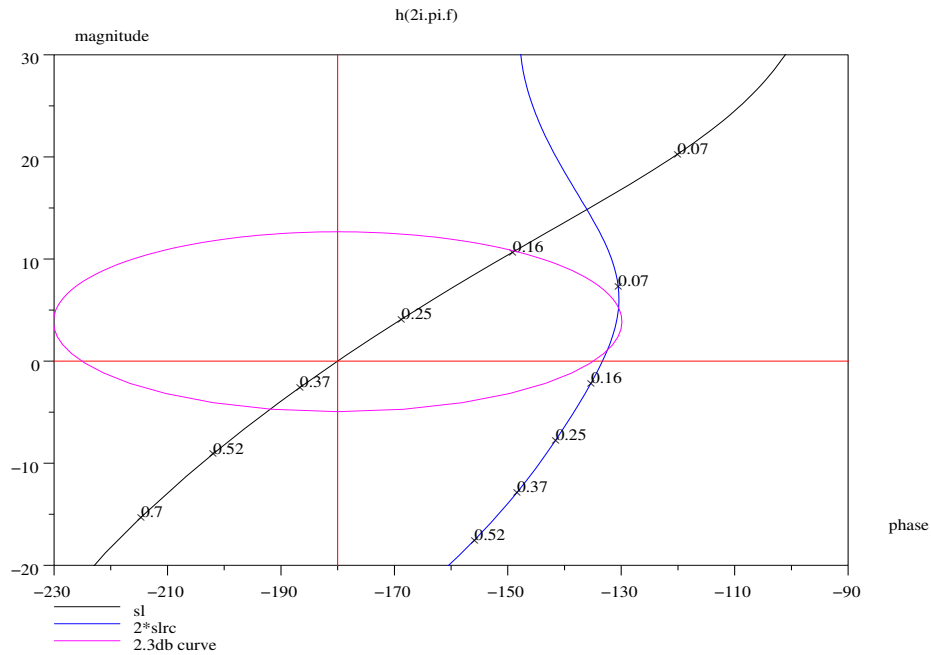


FIGURE 7.10 – Correction retard-avance de phase

```
-->mphi=p_margin(slrc)
mphi =
49.601046
-->bblack([sl;2*slrc],.01,1.5,["sl","2*slrc"])//Gain multiplié par deux.
Avec ce réseau, le système n'a pas une marge très très importante, (on a malgré tout
multiplié le gain par deux!), on va donc retoucher les deux fréquences,  $f_2$  et  $f_3$  : on
prend  $f_2 = 0,09$  hertz : on remonte légèrement le point de pseudo pivotement, et on
remonte plus fortement le point où l'avance de phase est maximale en prenant  $f_3 = 0,5$ 
hertz : on obtient ainsi le programme Scilab :
-->C=[log(10);-2*log(2*pi*.09);log(10);-2*log(2*pi*.5)];
-->X=inv(A)*C;
-->tau=exp(X)
tau =
! 31.067415 !
! 3.1067415 !
! 1.0065842 !
! 0.1006584 !
-->num=(1+tau(2,1)*s)*(1+tau(3,1)*s);
-->den=(1+tau(1,1)*s)*(1+tau(4,1)*s);
```

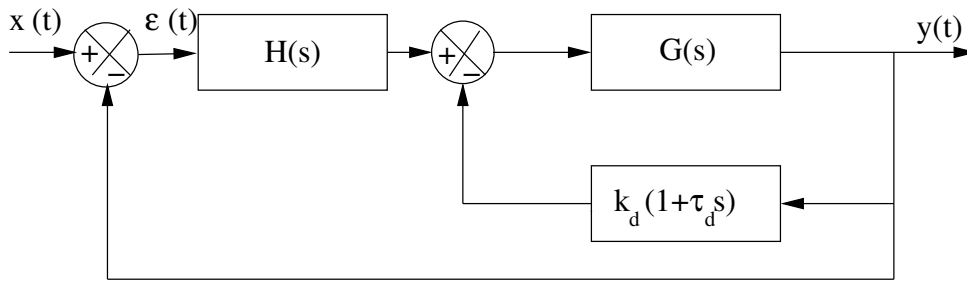



FIGURE 7.11 – Avec boucle interne dérivée.

```
-->rc=syslin("c",num/den)
rc =
          2
1 + 4.1133257s + 3.127197s
-----
          2
1 + 31.168073s + 3.127197s
-->slrc=sl*rc;
-->bblack([sl;4*slrc],.01,1.5,["sl","4*slrc"])
```

Par cette retouche on a pu ainsi augmenter le gain de la chaîne d'action, il est **quatre fois plus important**, tout en conservant une bonne marge de phase, de gain et un bon facteur de résonance ; vous vérifierez ces constatations avec le logiciel.

Par cette méthode on peut avec un tel logiciel de simulation, faire le choix le plus adéquat du réseau correcteur.

Une dernière remarque : N'oubliez pas de simuler les réponses temporelles du système bouclé et les réponses de l'actionneur : sortie du réseau correcteur (faire un exercice analogue à l'exercice de la section 7.3).

7.9 Correcteur à action proportionnelle et/ou intégrale et boucle interne dérivée

Un exemple vu au paragraphe 7.1.2 est le correcteur dans la chaîne de retour ou correcteur tachymétrique, en fait on utilise pour corriger le système un retour d'état partiel, en utilisant un capteur tachymétrique donnant la dérivée exacte de la sortie : dans l'exemple choisi on avait affaire à un asservissement de position (FIG 7.11).

D'après le schéma donné par la figure, la chaîne d'action de ce système a pour valeur : $F(s) = H(s) \frac{N(s)}{D(s) + k_d(1 + \tau_d s)N(s)}$ si $N(s)$ et $D(s)$ sont respectivement le numérateur et dénominateur de $G(s) = \frac{N(s)}{D(s)}$.

7.9.1 Correcteur à action intégrale dans la chaîne d'action

Dans ce cas $H(s)$ a pour expression $H(s) = K(\frac{1}{\tau_i s})$ et la chaîne d'action de ce système a pour valeur : $F(s) = K(\frac{1}{\tau_i s}) \frac{N(s)}{D(s) + k_d(1 + \tau_d s)N(s)}$. Ainsi nous avons à notre disposition trois paramètres de réglage : $\frac{K}{\tau_i}$, k_d , et τ_d . Ce type de schéma a pour objet de diminuer l'influence des variations brutales de la consigne et d'éviter des saturations sur les actionneurs, de même l'introduction du terme $\frac{K}{\tau_i s}$ (rajout d'un pôle à l'origine), améliore la précision statique (augmentation de la classe du système : voir chapitre 6.2.1), mais ceci se fait souvent au détriment d'un dépassement important de la réponse indicielle.

7.9.2 Correcteur proportionnel et intégral dans la chaîne d'action

Afin de palier à l'inconvénient que je viens de citer, on préfère choisir pour $H(s)$ un correcteur proportionnel et intégral de la forme $H(s) = K(1 + \frac{1}{\tau_i s})$. Dans ce cas la chaîne d'action du système vaut : $F(s) = K(1 + \frac{1}{\tau_i s}) \frac{N(s)}{D(s) + k_d(1 + \tau_d s)N(s)}$.

Avec ce type de schéma on peut plus ou moins modeler la transmittance de la chaîne d'action et donc déterminer les propriétés du système asservi (en particulier en utilisant des méthodes algébriques de synthèse type méthode de P. Naslin, des moments ... méthodes que l'on étudiera plus loin).

8 Méthodes algébriques et empiriques de synthèse

8.1 Les critères mesurant l'erreur dynamique

Nous avons vu que par les méthodes fréquentielles on cherchait souvent, en comparant le système à un système du second ordre, à définir un gabarit fréquentiel pour la boucle ouverte. Mais on peut aussi raisonner en temporel, faire l'étude en fonction du temps d'une réponse type : la réponse à l'échelon par exemple pour le système bouclé. Dans ce cas l'étude de l'erreur $e(t)$ de la boucle est essentiel, on a donc introduit différents critères permettant de chiffrer globalement l'erreur. Je redonne la réponse à un échelon d'un second ordre afin de visualiser la réponse, l'erreur dynamique et la valeur absolue de l'erreur.

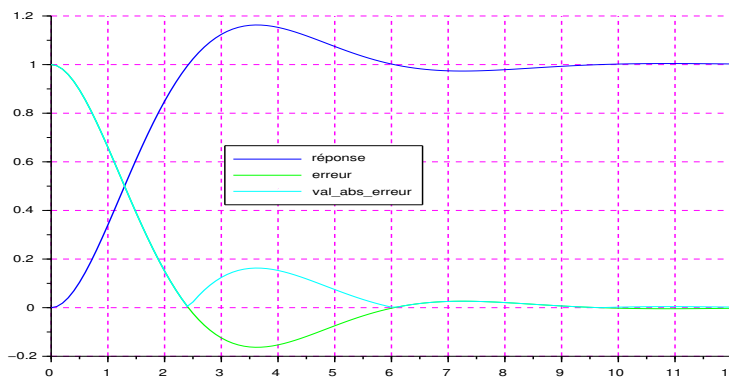


FIGURE 8.1 – Réponse à l'échelon, erreur, valeur absolue de l'erreur.

1. **Critère I.E : « Integral of Error ».** Cette intégrale est facilement calculable en fonction de l'erreur, c'est la surface de la courbe erreur (couleur verte FIG 8.1) par rapport à l'axe horizontal, bien sur les bornes dépendent de l'erreur statique, si celle ci est nulle ($e(t) = 0$ qd $t \rightarrow +\infty$) on intègre sur un temps très grand, sinon on détermine l'erreur permanente à cet échelon $e(+\infty)$ et l'on intègre non pas $e(t)$ mais $e(t) - e(+\infty)$. Le critère vaut : $I = \int_0^{+\infty} e(t) dt$. ou $I = \int_0^{+\infty} (e(t) - e(+\infty)) dt$. Cette remarque est valable pour tous les critères.

2. **Critère I.A.E : « Integral of Absolute Error »**. Sur cette même figure on a le tracé de la valeur absolue de l'erreur (courbe bleu clair) et de même le calcul de l'intégrale ce fait comme pour l'intégrale de l'erreur, pour un système du second ordre, cette intégrale est minimale pour un amortissement $\xi = 0.7$. Une remarque, la première partie de la courbe erreur a la surface la plus importante. Le critère vaut : $I = \int_0^{+\infty} |e(t)| dt$: cette remarque est valable pour tous les critères.
3. **Critère I.S.E : « Integral of Square Error »**. Ici le critère vaut $I = \int_0^{+\infty} e^2(t) dt$ et ici la première partie de la courbe erreur est prépondérante et pour un second ordre, ce critère est minimal pour $\xi = 0.43$, voir la méthode de Hall et Sartorius.
4. **Critère I.T.A.E : « Integral Time multiplied by Absolute Error »**. On a $I = \int_0^{+\infty} t |e(t)| dt$: on pénalise une réponse très oscillatoire, pour un second ordre la valeur minimale du critère est obtenue pour $\xi = 0.7$, on verra avec intérêt la méthode de Graham et Lathrop.
5. **Critère I.T.S.E : « Integral Time multiplied by Square Error »**. C'est un compromis entre les critères I.S.E et I.T.A.E, on pénalise plus la fin du régime transitoire, pour un second ordre la valeur minimale est obtenue pour un $\xi = 0.58$. On a $I = \int_0^{+\infty} te^2(t) dt$.
6. En résumé avec ces différents critères on peut : minimiser l'énergie de réglage, on prendra le critère I.A.E ou I.T.A.E (pour un second ordre on a $\xi = 0.7$), si on privilégie le temps de montée on prendra le critère I.T.S.E.

8.2 Méthode de Ziegler et Nichols

Cette méthode empirique a pour objet le réglage d'un réseau correcteur P.I.D de la forme $k_p(1 + \frac{1}{\tau_i s} + \tau_d s)$, quand le procédé à régler est à **réponse indicielle apériodique**, sans que ce dernier soit parfaitement identifié. Les auteurs ont déterminé, à partir de simulations analogiques, de divers modèles de systèmes physiques, les paramètres du réseau correcteur P.I.D minimisant l'indice de performance (critère I.A.E) $I = \int_0^{\infty} |e(t)| dt$ avec $e(t)$ l'erreur de l'asservissement. Deux méthodes sont proposées, la première demande de déterminer, en boucle ouverte, deux paramètres de la réponse indicielle apériodique : (FIG 8.2).

A partir de deux paramètres, l'abscisse à l'origine de la tangente d'inflexion T_r et la pente de cette tangente d'inflexion a , on détermine le réglage du P.I.D (voir tableau ci-après), remarquons que $a = \frac{1}{T_u}$. Quant à la seconde façon de procéder elle demande de boucler le procédé avec un amplificateur de gain réglable k et d'augmenter ce gain jusqu'à ce qu'apparaissent des oscillations entretenues : phénomène de pompage, (période des oscillations T_l , gain limite k_l).

Correcteur	Essai indiciel (a, T_r)	Essai de pompage (k_l, T_l)
k_p	$k_p = \frac{1}{aT_r} = \frac{T_u}{T_r}$	$k_p = 0.5k_l$
$k_p(1 + \frac{1}{\tau_i s})$	$k_p = \frac{0.9}{aT_r} = \frac{0.9T_u}{T_r}, \tau_i = 3.3T_r$	$k_p = 0.45k_l, \tau_i = 0.83T_l$
$k_p(1 + \frac{1}{\tau_i s} + \tau_d s)$	$k_p = \frac{1.2}{aT_r} = \frac{1.2T_u}{T_r}, \tau_i = 2T_r, \tau_d = 0.5T_r$	$k_p = 0.6k_l, \tau_i = 0.5T_l, \tau_d = 0.125T_l$

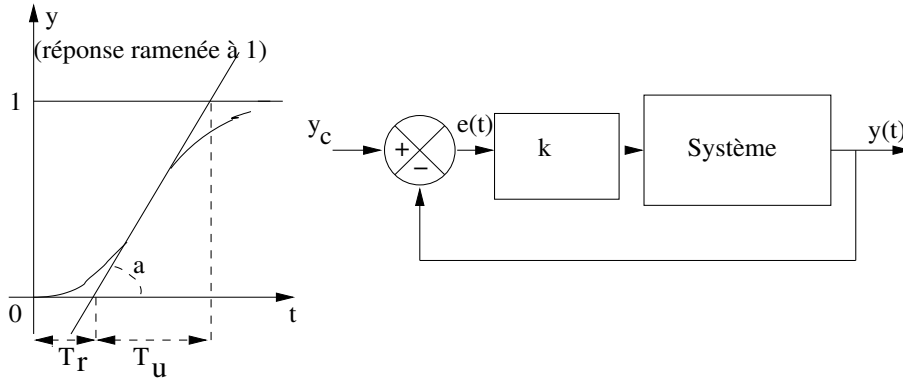


FIGURE 8.2 – Méthode de Ziegler et Nichols.

Le seul problème qui peut se poser réside en la détermination du point d'inflexion et de sa tangente : on réalise ceci à partir de données expérimentales qui peuvent être plus ou moins bruitées.

La réponse réelle à un échelon suggère un modèle de procédé de type soit : $g(s) = \frac{e^{-ds}}{(1+\tau s)^n}$, tandis que le modèle choisi par Ziegler et Nichols a la forme : $\frac{e^{-T_r s}}{s T_u}$ au gain statique près.

Remarques :

- Pour que l'indice de performance est un sens on doit avoir une erreur permanente nulle pour une entrée échelon.
- De même on remarque que cette méthode s'applique surtout pour une régulation de maintien, avec perturbations, pour un système à réponse apériodique.
- Les réglages proposés donnent en général un bon comportement en régulation, mais avec un transitoire de médiocre qualité (dépassement de l'ordre de 30% à 50% pour la réponse indicielle).

8.3 Méthode de Chien-Hrones-Reswick

Cette méthode s'applique sur le même modèle que celui proposé dans la méthode de Ziegler et Nichols (voir figure précédente). On obtient ainsi les paramètres des régulateurs P, P.I, P.I.D en utilisant le tableau ci après.

Régul	Coef	Apériodique	Apériodique	Dépas $D = 20\%$	Dépas $D = 20\%$
		Maintien	Correspondance	Maintien	Correspondance
P	k_p	$0.3T_u/T_r$	$0.3T_u/T_r$	$0.7T_u/T_r$	$0.7T_u/T_r$
P.I	k_p	$0.6T_u/T_r$	$0.35T_u/T_r$	$0.7T_u/T_r$	$0.6T_u/T_r$
P.I	τ_i	$4T_r$	$1.2T_u$	$2.3T_r$	$1T_u$
P.I.D	k_p	$0.95T_u/T_r$	$0.6T/T_r$	$1.2T_u/T_r$	$0.95T_u/T_r$
P.I.D	τ_i	$2.4T_r$	$1T_u$	$2T_r$	$1.3T_u$
P.I.D	τ_d	$0.42T_r$	$0.5T_r$	$0.42T_r$	$0.47T_r$

8.4 Méthode de Graham et Lathrop

Cette méthode est basée sur la minimisation du critère I.T.A.E soit $I = \int_0^{+\infty} t |e(t)| dt$ et ceci pour une entrée à l'échelon pour la boucle fermée. Ce critère pénalise une réponse très oscillatoire et l'erreur dynamique peut être importante en début de réponse. On peut donner deux tableaux donnant les transmittances « optimales » suivant que les erreurs statiques en position ou en position et en vitesse sont annulées.

Transmittances à erreurs en position nulle.	
Ordre	Boucle fermée $W(s)$, minimum de $I = \int_0^{+\infty} t e(t) dt$
1	$\frac{\omega_n}{s + \omega_n}$
2	$\frac{\omega_n^2}{s^2 + 1.4\omega_n s + \omega_n^2}$
3	$\frac{\omega_n^3}{s^3 + 1.75\omega_n s^2 + 2.15\omega_n^2 s + \omega_n^3}$
4	$\frac{\omega_n^4}{s^4 + 2.1\omega_n s^3 + 3.4\omega_n^2 s^2 + 2.7\omega_n^3 s + \omega_n^4}$
5	$\frac{\omega_n^5}{s^5 + 2.8\omega_n s^4 + 5\omega_n^2 s^3 + 5.5\omega_n^3 s^2 + 3.4\omega_n^4 s + \omega_n^5}$
6	$\frac{\omega_n^6}{s^6 + 3.25\omega_n s^5 + 6.6\omega_n^2 s^4 + 8.6\omega_n^3 s^3 + 7.45\omega_n^4 s^2 + 3.95\omega_n^5 s + \omega_n^6}$

Transmittances à erreurs en position et vitesse nulles.	
Ordre	Boucle fermée $W(s)$, minimum de $I = \int_0^{+\infty} t e(t) dt$
2	$\frac{3.2\omega_n s + \omega_n^2}{s^2 + 3.2\omega_n s + \omega_n^2}$
3	$\frac{3.25\omega_n^2 s + \omega_n^3}{s^3 + 1.75\omega_n s^2 + 3.25\omega_n^2 s + \omega_n^3}$
4	$\frac{5.14\omega_n^3 s + \omega_n^4}{s^4 + 2.41\omega_n s^3 + 4.93\omega_n^2 s^2 + 5.14\omega_n^3 s + \omega_n^4}$
5	$\frac{5.24\omega_n^4 s + \omega_n^5}{s^5 + 2.19\omega_n s^4 + 6.5\omega_n^2 s^3 + 6.3\omega_n^3 s^2 + 5.24\omega_n^4 s + \omega_n^5}$
6	$\frac{6.76\omega_n^5 s + \omega_n^6}{s^6 + 6.12\omega_n s^5 + 13.42\omega_n^2 s^4 + 17.6\omega_n^3 s^3 + 14.14\omega_n^4 s^2 + 6.76\omega_n^5 s + \omega_n^6}$

Voici un programme Scilab donnant les réponses à l'échelon des boucles fermées pour $\omega_n = 1$ des systèmes (ordres (2, 3, 4, 5) pour le premier tableau : courbes $S2, S3, S4, S5$) FIG 8.3. Mêmes ordres pour le deuxième tableau : (courbes $T2, T3, T4, T5$) FIG 8.4.

```
-->s=%s;
```

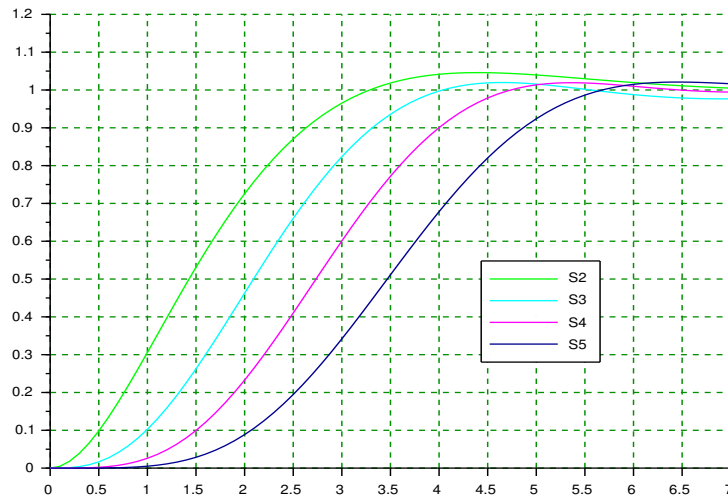


FIGURE 8.3 – Transmittances optimales, numérateurs constants.

```
-->DEN=[s*s+1.4*s+1;s^3+1.75*s^2+2.25*s+1;s^4+2.1*s^3+3.4*s^2+2.7*s+1;..
s^5+2.8*s^4+5*s^3+5.5*s^2+3.4*s+1];
-->NUM=ones(DEN);
-->S=syslin("c",NUM,DEN);
-->t=0:.1:7;
-->REP=csim("step",t,S);
-->plot2d(t',REP',style=[3,4,6,9])
-->legend(["S2","S3","S4","S5"],5)
-->xgrid(13)

-->DEN=[s*s+3.2*s+1;s^3+1.75*s^2+3.25*s+1;s^4+2.41*s^3+4.93*s^2+..
5.14*s+1;s^5+2.19*s^4+6.5*s^3+6.3*s^2+5.24*s+1];
-->NUM=[1+3.2*s;1+3.25*s;1+5.14*s;1+5.24*s];
-->T=syslin("c",NUM,DEN)
-->REP=csim("step",t,T);
-->plot2d(t',REP',style=[3,4,6,9])
-->legend(["T2","T3","T4","T5"],5);
-->xgrid(13);
```

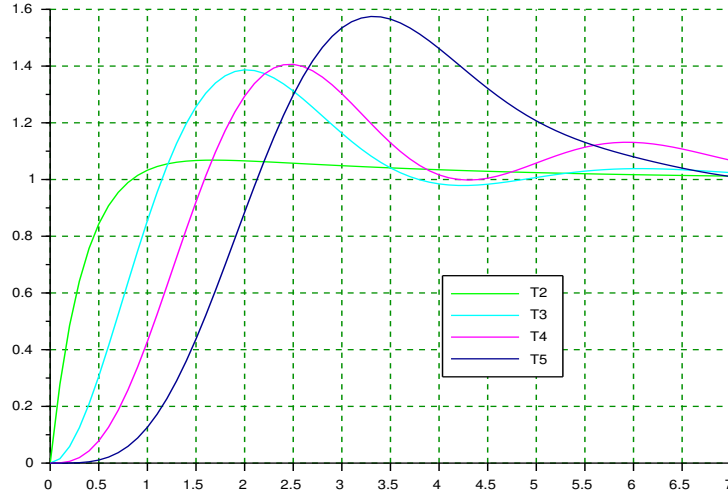


FIGURE 8.4 – Transmittances optimales, numérateurs degré un.

8.5 Méthode de Hall et Sartorius

bla

8.6 Méthode des polynômes de Naslin

8.6.1 Principe

Les méthodes de synthèse, en particulier les méthodes fréquentielles, consistent en un modelage de la transmittance en boucle ouverte à l'aide de réseaux correcteurs, alors que ce sont les performances de la boucle fermée qui importent. P. Naslin [11] propose une méthode directe modelant la transmittance de la boucle fermée à l'aide de polynômes normaux à amortissement réglable. En effet pour un système du second ordre, la boucle fermée de celui-ci a pour expression $W(s) = \frac{b_0}{a_0 + a_1 s + a_2 s^2}$ ou encore

$W(s) = \frac{b_0 \frac{\omega_n^2}{a_0}}{\omega_n^2 + 2\xi\omega_n s + s^2}$ avec $\omega_n = \sqrt{\frac{a_0}{a_2}}$ la pulsation naturelle appelée quelquefois la pulsation propre sans amortissement, et ξ le facteur d'amortissement ou plus simplement l'amortissement qui vérifie $4\xi = \frac{a_1^2}{a_0 a_2}$. Si cet amortissement vaut 1 alors c'est la plus petite valeur de ξ donnant une réponse indicielle apériodique sans dépassement. Si $\xi = 0.7$ on a un amortissement dit optimal avec un dépassement indiciel de 5%.

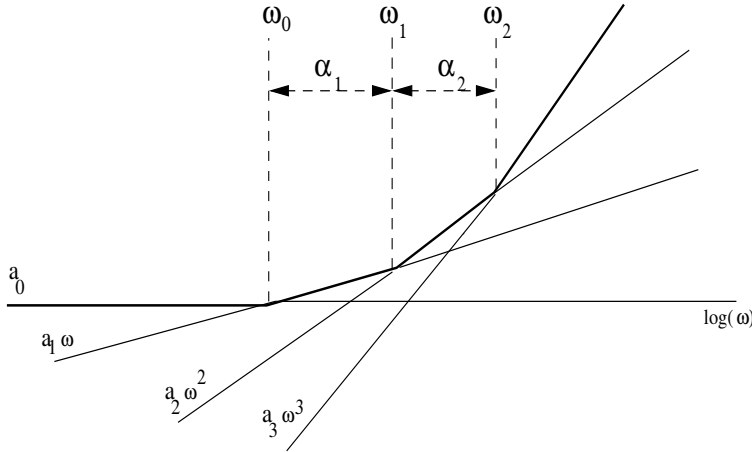


FIGURE 8.5 – Courbe de gain pseudo-asymptotique

Enfin si $\xi = 0.5$ ce dépassement est de 15%. Ces deux dernières valeurs correspondent respectivement à deux pôles complexes conjugués sur des droites passant par l'origine du plan complexe faisant des angles de 135° et 120° par rapport à l'axe réel.

Par généralisation avec un système quelconque de modèle $W(s) = \frac{b_0}{a_0 + a_1 s + a_2 s^2 + \dots + a_N s^N}$ mais de numérateur constant, P. NASLIN définit les **rapports caractéristiques** les nombres suivants :

$$\alpha_1 = \frac{a_1^2}{a_0 a_2}, \alpha_2 = \frac{a_2^2}{a_1 a_3}, \dots, \alpha_n = \frac{a_n^2}{a_{n-1} a_{n+1}}, \dots$$

Ces rapports peuvent être calculés à partir des **pulsations caractéristiques** :

$$\omega_0 = \frac{a_0}{a_1}, \omega_1 = \frac{a_1}{a_2}, \dots, \omega_n = \frac{a_n}{a_{n+1}}, \dots$$

on a

$$\alpha_1 = \frac{\omega_1}{\omega_0}, \alpha_2 = \frac{\omega_2}{\omega_1}, \dots, \alpha_n = \frac{\omega_n}{\omega_{n-1}}, \dots$$

Une interprétation géométrique est proposée par l'auteur en définissant le **diagramme pseudo-asymptotique** du polynôme $a_0 + a_1 s + a_2 s^2 + \dots + a_N s^N$ voir FIG 8.5. Ce diagramme est constitué des droites donnant le gain des termes successifs $a_i s^i$, les pulsations caractéristiques sont les abscisses des points anguleux, et les rapports caractéristiques mesurent les segments successifs.

Enfin le critère de P. NASLIN définit un ensemble de polynômes normaux à amortissement réglable, en choisissant des rapports caractéristiques de même valeur α , (ou proches de cette valeur α), liée à l'amortissement : en fait on impose aux deux pôles complexes conjugués dominants d'être sur deux droites symétriques de l'axe réel et faisant un angle supérieur à 90° par rapport à l'axe réel positif, même principe que dans la définition des pseudo-lieux (paragraphe 5.2.7).

8.6.2 Calcul des coefficients

Un polynôme de cette famille est entièrement déterminé par, son degré N , son facteur d'amortissement α , et une pulsation caractéristique. Si l'on se donne la première pulsation ω_0 on obtient l'ensemble des pulsations sous la forme :

$$\omega_0, \alpha\omega_0, \alpha^2\omega_0, \alpha^3\omega_0, \dots,$$

Pour avoir tous les coefficients du polynôme on doit donner à l'un quelconque la valeur 1. Ainsi si $a_0 = 1$ alors le vecteur des coefficients du polynôme vaut :

$$1, \omega_0^{-1}, \alpha^{-1}\omega_0^{-2}, \dots$$

soit plus généralement pour $a_0 = 1$ on a $\omega_n = \alpha^n\omega_0$ et donc $a_n = \alpha^{\frac{-n(n-1)}{2}}\omega_0^{-n}$.

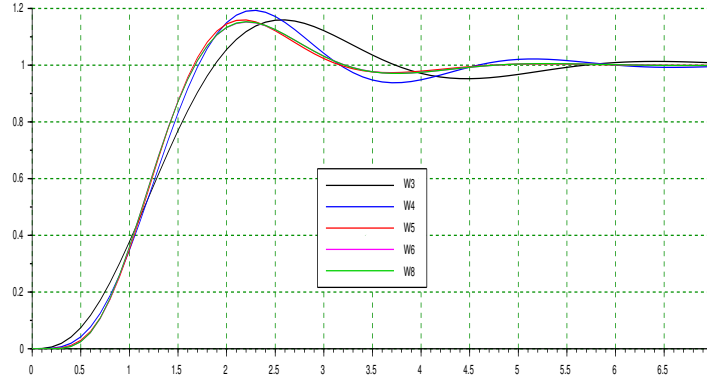
8.6.3 Les réponses indicielles

Par simulation analogique P. NASLIN a montré que le terme α jouait le rôle de facteur d'amortissement, en faisant de nombreuses simulations pour différentes valeurs de α et de N .

- On constate que pour une même valeur de α et pour $N > 2$, les réponses indicielles sont peu dispersées et présentent des dépassements pratiquement égaux.
- Pour une valeur de N et pour des valeurs de α différentes, le dépassement de la réponse indicielle est d'autant plus grand que le nombre α est petit : la valeur proche de $\alpha = 1.5$ conduit à un système instable quelque soit N .
- Le dépassement D en pour-cent est lié à α par une formule empirique : $\log_{10}(D\%) = 4.8 - 2\alpha$, avec en plus $b_0 = a_0$ (gain statique égal à 1). Le temps de montée où on a un maximum de la réponse, vaut $t_m = \frac{2.2}{\omega_0}$, la pente maximale est égale à : $E = 0.8 \frac{b_0}{a_0} \omega_0 \frac{100+D}{100}$.

Voici un programme Scilab permettant de mettre en évidence ces résultats. On prendra $\alpha = 1.75$, $a_0 = 1$, $b_0 = 1$, et $\omega_0 = 1 \text{ rd/s}$ cette dernière valeur normalise le temps, en effet pour la réponse indicielle l'axe horizontal peut être gradué en $\omega_0 t$. De même on choisit $N = [3, 4, 5, 6, 8]$ (FIG 8.6).

```
-->alp=1.75;
-->P3=poly([1,1,1/alp,1/alp^3],"s","c")
-->P4=poly([1,1,1/alp,1/alp^3,1/alp^6],"s","c")
-->P5=poly([1,1,1/alp,1/alp^3,1/alp^6,1/alp^10],"s","c")
-->P6=poly([1,1,1/alp,1/alp^3,1/alp^6,1/alp^10,1/alp^15],"s","c")
-->P8=poly([1,1,1/alp,1/alp^3,1/alp^6,1/alp^10,1/alp^15,...
1/alp^21,1/alp^28],"s","c")
-->W3=syslin("c",1,P3)
//Idem de W4 à W8
-->t=[0:.1:7];
-->SW3=csim("s",t,W3); //idem pour W4 à W8.
-->plot2d(t',[SW3',SW4',SW5',SW6',SW8'],style=[1;2;5;6;15])
-->legend(["W3";"W4";"W5";"W6";"W8"],5)
-->xgrid(13)
```


 FIGURE 8.6 – Réponses pour $\alpha = 1.75$ et $N = [3, 4, 5, 6, 8]$

Maintenant pour $N = 4$ nous allons tracer les réponses indicielles pour différentes valeurs de $\alpha = [1.6; 1.8; 2; 2.2; 2.4]$ avec toujours les mêmes conditions, à savoir $a_0 = 1$, $b_0 = 1$, et $\omega_0 = 1 \text{rd/s}$ (FIG 8.7).

```
-->alp1=1.6;alp2=1.8;alp3=2;alp4=2.2;alp5=2.4;
-->P1=poly([1,1,1/alp1,1/alp1^3,1/alp1^6],"s","c");//idem P2 à P5.
-->W1=syslin("c",1,P1)//idem W2 à W5.
-->SW1=csim("s",t,W1);//idem SW2 à SW5.
-->plot2d(t',[SW1',SW2',SW3',SW4',SW5'],style=[1;2;5;6;15])
-->legend(["W1_1.6";"W2_1.8";"W3_2";"W4_2.2";"W5_2.4"],5);xgrid(13);
```

Voici un tableau donnant en fonction de α et en comparaison avec l'amortissement d'un second ordre ξ , la valeur approximative du dépassement.

α	$D\%$	ξ	α	$D\%$	ξ
1.6	40	0.3	2	6	0.7
1.75	20	0.45	2.4	1	0.9

8.6.4 Influence d'un numérateur du premier ou second degré

Dans de nombreuses applications de synthèse, en particulier quand on commande un système avec un réseau correcteur P.I.D, il apparaît au numérateur de la transmittance bouclée, un numérateur du premier et ou second degré. Il peut résulter un accroissement du premier dépassement et une réduction du temps de montée lors de la réponse indicielle. On doit donc considérer des transmittances bouclées de la forme

$$W(s) = \frac{b_0 + b_1 s}{a_0 + a_1 s + a_2 s^2 + \dots + a_N s^N}$$

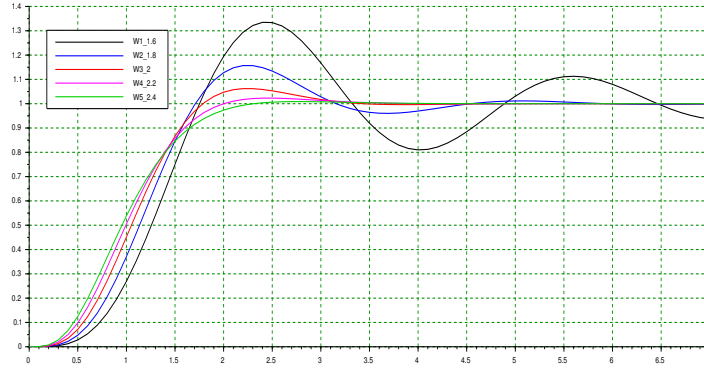


FIGURE 8.7 – Réponses pour $N = 4$ et $\alpha = [1.6; 1.8; 2; 2.2; 2.4]$

on pose $\omega'_0 = \frac{b_0}{b_1}$. Les formules précédemment données restent valables à condition de remplacer α par α_e et ω_0 par ω_{0c} (ω_0 corrigé) valeurs données par les relations : $\alpha_e = 1.5 + \frac{\omega'_0}{4\omega_0}(\alpha - 1.5)$ et $\frac{1}{\omega_{0c}} = \frac{1}{\omega_0} - \frac{1}{2\omega'_0}$. Pour une transmittance possédant un numérateur du second degré

$$W(s) = \frac{b_0 + b_1s + b_2s^2}{a_0 + a_1s + a_2s^2 + \dots + a_Ns^N}$$

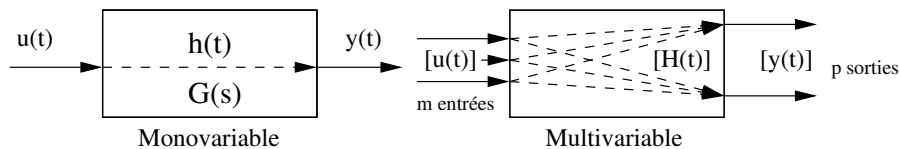
on pose $\omega'^2_0 = \frac{b_0}{b_2}$ et $4\xi'^2 = \frac{b_1^2}{b_0b_2}$. Les valeurs de α sont remplacées par α_e et ω_0 par ω_{0c} suivant les relations : $\alpha_e = 1.5 + \frac{1}{16\xi'^3} \frac{\omega'^2_0}{\omega_0^2}(\alpha - 1.5)$ et $\frac{1}{\omega_{0c}} = \frac{1}{\omega_0} - \frac{1}{\omega'_0}$. Ces formules pour un numérateur du premier ou second degré ne sont valables que si elles donnent une valeur de α_e inférieure à α . Sinon, c'est que l'influence du numérateur est négligeable.

9 Représentation d'état des systèmes

9.1 Définition de l'état d'un système

9.1.1 Modèles de systèmes continus

Soit un système (Figure ci après), l'entrée est la fonction $u(t)$, la sortie $y(t)$, et la réponse impulsionnelle la fonction $h(t)$ pour un système monovarié. Pour un système multivarié l'entrée est un vecteur $u(t)$ de dimensions $(m \times 1)$, la sortie un vecteur $y(t)$ de dimensions $(p \times 1)$, quant à la matrice des réponses impulsionnelles elle a pour dimensions $(p \times m)$.



On peut donc donner un tableau résumant les différentes formes du modèle de ce système monovarié : tableau ci dessous.

Equation différentielle	Fonction de transfert	Equations d'état
$\frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + \dots + a_0 y = b_m \frac{d^m u}{dt^m} + \dots + b_0 u$ $m \leq n$	$\frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_0}{s^n + a_{n-1} s^{n-1} + \dots + a_0}$ $\frac{Y(s)}{U(s)} = G(s) = \frac{N(s)}{D(s)}$ $m \leq n$	$\frac{dx(t)}{dt} = Ax(t) + Bu(t)$ $y(t) = Cx(t) + Du(t)$
Equation caractéristique (E C) $r^n + a_{n-1} r^{n-1} + \dots + a_0 = 0$	Equation caractéristique (E C) $s^n + a_{n-1} s^{n-1} + \dots + a_0 = 0$	Polynôme caractéristique $P(\lambda) = \det(\lambda I - A)$
ordre du système : n	ordre du système : n	ordre du système : $n = \dim(A)$
Modes = racines de E C $r_i \quad i = 1, \dots, n$	Modes = racines de E C $s_i \quad i = 1, \dots, n$	Modes = valeurs propres de A $\lambda_i \quad i = 1, \dots, n$

Quant au système multivarié présenté sur la figure, les différentes formes du modèle sont données par le tableau ci après :

Système différentiel	Matrice de transfert	Equations d'état
$D(r)y(t) = N(r)u(t)$ $r = \frac{d}{dt}$	$G(s) = D^{-1}(s)N(s)$	$\frac{dx(t)}{dt} = Ax(t) + Bu(t)$ $y(t) = Cx(t) + Du(t)$
Equation caractéristique (E C) $\det(D(r)) = 0$	Equation caractéristique (E C) <i>Forme de Smith – Mc Millan</i>	Polynôme caractéristique $P(\lambda) = \det(\lambda I - A)$
Ordre du système $n = \text{degré}(\det(D(r)))$	Ordre du système <i>Forme de Smith – Mc Millan</i>	Ordre du système $n = \dim(A)$
Modes = racines de E C $r_i \quad i = 1, \dots, n$	Modes = pôles de $G(s)$ <i>Forme de Smith – Mc Millan</i>	Modes = valeurs propres de A $\lambda_i \quad i = 1, \dots, n$

9.1.2 Cas d'un système monovariable (SISO)

La représentation d'état d'un système linéaire ou linéarisable est issue du plan de phase (paragraphe 3.2.4). Je rappelle que le vecteur d'état x d'un procédé, est un ensemble de variables, en nombre minimum, dont la connaissance à l'instant t_0 , associée à la connaissance de l'évolution de l'entrée $u(t)$ pour $t \geq t_0$ permet, ayant le modèle du procédé, de prévoir l'évolution future du système et ceci en l'absence de perturbations (sauf si l'on connaît un modèle des perturbations). Ainsi dans l'exemple du paragraphe 3.2.4 j'ai choisi une équation différentielle d'ordre deux, cette équation est intégrable si l'on connaît d'une part, l'évolution de l'entrée entre l'instant initial t_0 et l'instant t de calcul et d'autre part, si l'on a les deux conditions initiales (sortie, dérivée de la sortie). Plus généralement à partir d'une équation différentielle d'ordre n , n conditions initiales sont indispensables pour intégrer cette équation. Le procédé aura un vecteur d'état de dimension n .

Soit donc un système d'ordre n dont le modèle est le suivant : (j'ai normalisé la transmittance en prenant $a_n = 1$).

$$\frac{d^n y}{dt^n} + a_{n-1} \frac{d^{n-1} y}{dt^{n-1}} + a_{n-2} \frac{d^{n-2} y}{dt^{n-2}} + \dots + a_1 \frac{dy}{dt} + a_0 y = b_m \frac{d^m u}{dt^m} + \dots + b_0 u$$

auquel on associe n conditions initiales sous forme du vecteur :

$$y_0^t = \left[y(t_0) \frac{dy}{dt}(t_0) \dots \frac{d^{n-1}}{dt^{n-1}}(t_0) \right]^t$$

Ce modèle correspond à la fonction de transfert $G(s) = \frac{N_{um}(s)}{D_{en}(s)}$ (fonction définie pour des conditions initiales nulles).

$$\frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0} = \frac{N_{um}(s)}{D_{en}(s)}$$

Compte tenu des définitions précédentes on peut trouver une troisième représentation pour le modèle du procédé, à savoir :

$$\begin{aligned} \frac{dx(t)}{dt} &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned}$$

On a $D = 0$ si $m < n$ et $D \neq 0$ si $m = n$.

En effet on peut, si $m = n$, mettre la transmittance sous la forme :

$$\frac{Y(s)}{U(s)} = \frac{b_{n-1}^* s^{n-1} + b_{n-2}^* s^{n-2} + \dots + b_1^* s + b_0^*}{s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0} + d_0$$

avec $D = d_0 = b_n$.

Dans le cas où l'on a affaire à un système monovariable, on a :

9 Représentation d'état des systèmes

$x(t) \in \mathbb{R}^n$ est le vecteur d'état du système, $u(t) \in \mathbb{R}$ est la commande, $y(t) \in \mathbb{R}$ est la sortie. Enfin, $A \in \mathbb{R}^{n \times n}$ est dite matrice d'évolution, $B \in \mathbb{R}^{n \times 1}$ est le vecteur de commande, $C \in \mathbb{R}^{1 \times n}$ le vecteur d'observation, et enfin $D \in \mathbb{R}$ caractérise la transmission directe.

Comme on l'a vu (paragraphe 3.2.4), la représentation n'est pas unique : en effectuant un changement de variables défini par la matrice P (P régulière) tel que $x(t) = Px^*(t)$, le système d'équations devient :

$$\begin{aligned}\frac{dx^*(t)}{dt} &= A^* x^*(t) + B^* u(t) \\ y(t) &= C^* x^*(t) + Du(t)\end{aligned}$$

avec : $A^* = P^{-1}AP$, $B^* = P^{-1}B$, $C^* = CP$.

Ceci montre que si la représentation d'état n'est pas unique, le polynôme caractéristique de la matrice A est, d'une part indépendant du choix du vecteur d'état, et d'autre part identique au dénominateur de la fonction de transfert du système, plus généralement on a :

$$\frac{N_{um}(s)}{D_{en}(s)} = C(sI - A)^{-1}B + D$$

On peut donc dire que les pôles de la fonction de transfert (non simplifiée s'il y a lieu) sont les valeurs propres de A .

Remarque très importante : Simplification pôles-zéros.

Dans le tableau que j'ai donné précédemment (paragraphe 10.1.1) il y a une imprécision évidente : quand je donne à la première colonne du tableau, l'ordre (n) et les modes (les racines de l'équation caractéristique $r^n + a_{n-1}r^{n-1} + \dots + a_0 = 0$), et que je compare avec la deuxième colonne du tableau j'admets implicitement que dans le modèle transmittance, il n'y a pas eu une simplification d'un pôle par un zéro dans la transmittance (fraction rationnelle irréductible) et donc que le dénominateur de cette transmittance ($D_{en}(s)$) est identique à l'équation caractéristique précédemment définie. Prenons un exemple pour expliciter cette notion.

Soit un système dont le modèle est l'équation différentielle suivante :

$$\frac{d^2 y(t)}{dt^2} - y(t) = \frac{du(t)}{dt} - u(t)$$

à laquelle on doit associer les conditions initiales :

$$y(0^+), \frac{dy}{dt}(0^+), u(0^+)$$

Maintenant, définissons la transmittance, en n'oubliant pas, que dans ce cas, les conditions initiales sont supposées nulles. On a donc la relation :

$$(s^2 - 1)y(s) = (s - 1)u(s)$$

soit la transmittance

$$G(s) = \frac{y(s)}{u(s)} = \frac{s - 1}{s^2 - 1} = \frac{1}{s + 1}$$

9 Représentation d'état des systèmes

Par cette simplification on a diminué d'une unité l'ordre du système et plus grave on a fait disparaître un mode instable ($s_1 = 1$). En réalité, en prenant la transformée de Laplace de l'équation différentielle on devait écrire l'équation :

$$s^2 y(s) - s \frac{dy}{dt}(0^+) - y(0^+) = s u(s) - u(s) - u(0^+)$$

et donc le modèle complet a la forme :

$$y(s) = \frac{1}{s+1} u(s) + \frac{s \frac{dy}{dt}(0^+) + y(0^+) - u(0^+)}{s^2 - 1}$$

Le premier terme représente le transfert (transmittance) entre $u(t)$ et $y(t)$, quant au second terme c'est le transfert des conditions initiales à $y(t)$. Ce deuxième terme diverge quand $t \rightarrow +\infty$: mode instable. Dans une modélisation sous forme d'état, (troisième colonne du tableau précédent) on devra donc partir de l'équation différentielle et non de la transmittance (sauf si l'on sait que la fraction rationnelle correspondante est irréductible).

9.1.3 Cas d'un système multivariable (MIMO)

Dans le cas où l'on a affaire à un système multivariable (tableau 2) le problème se complique et en général on recherche une modélisation sous forme d'état, sinon on doit passer par une forme minimale : forme de Smith-Mc Millan.

Malgré ce que je viens de dire, la représentation d'état est la même que la représentation d'état d'un système SISO, seules les dimensions des vecteurs $u(t)$ et $y(t)$, changent ainsi que les matrices B, C, D : on a donc avec mes hypothèses les relations suivantes :

$$\begin{aligned} \frac{dx(t)}{dt} &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned}$$

$x(t) \in \mathbb{R}^n$ est le vecteur d'état du système, $u(t) \in \mathbb{R}^m$ est la commande, $y(t) \in \mathbb{R}^p$ est la sortie. Enfin, $A \in \mathbb{R}^{n \times n}$ est dite matrice d'évolution, $B \in \mathbb{R}^{n \times m}$ est la matrice de commande, $C \in \mathbb{R}^{p \times n}$ la matrice d'observation, et enfin $D \in \mathbb{R}^{p \times m}$ caractérise la transmission directe. Comme pour un système SISO, la représentation d'état n'est pas unique.

9.2 Intégration des équations d'état

9.2.1 Intégration, cas général

Soit un instant initial noté t_i et l'instant $t > t_i$, on peut, par intégration des équations d'état, trouver l'état $x(t)$ en fonction de l'état initial $x(t_i)$ et de la commande $u(\tau)$ entre ces deux instants : soit

$$\begin{aligned}x(t) &= e^{A(t-t_i)}x(t_i) + \int_{t_i}^t e^{A(t-\tau)}Bu(\tau)d\tau \\y(t) &= Cx(t) + Du(t)\end{aligned}$$

Le premier terme représente la contribution de l'état initial (réponse libre), quant au second c'est la contribution de l'entrée à l'instant t : ces équations sont valables pour un système multivariable.

9.2.2 Intégration sur un intervalle de temps donné à commande constante sur cet intervalle : bloqueur d'ordre zéro, BOZ

Soit deux instants t_1 et t_2 avec $t_1 < t_2$ on notera $T_{2,1}$ l'intervalle de temps entre t_1 et t_2 , durant cet intervalle la commande est constante et égale à $u_{1,2}$ (bloqueur d'ordre zéro) : on cherche la valeur du vecteur d'état à l'instant t_2 noté $x(t_2)$ en fonction du vecteur d'état à l'instant t_1 noté $x(t_1)$ et de la commande constante $u_{1,2}$. On montre à partir des équations précédentes que :

$$\begin{aligned}x(t_2) &= e^{A(t_2-t_1)}x(t_1) + \int_{t_1}^{t_2} e^{A(t_2-\theta)}Bu(\theta)d\theta \\y(t_2) &= Cx(t_2)\end{aligned}$$

en supposant $D = 0$.

Avec les notations proposées on a :

$$x(t_2) = e^{AT_{2,1}}x(t_1) + \left(\left(\int_{t_1}^{t_2} e^{A(t_2-\theta)}d\theta \right) B \right) u_{1,2}$$

en faisant $\theta = t_2 - \mu$ alors

$$x(t_2) = e^{AT_{2,1}}x(t_1) + \left(\left(\int_0^{T_{2,1}} e^{A\mu}d\mu \right) B \right) u_{1,2}$$

soit encore (si A est inversible) :

$$\begin{aligned}x(t_2) &= e^{AT_{2,1}}x(t_1) + [A^{-1}(e^{AT_{2,1}} - I)]Bu_{1,2} \\y(t_2) &= Cx(t_2)\end{aligned}$$

ces deux équations peuvent s'écrire maintenant en tenant compte de D :

$$\begin{aligned}x(t_2) &= \Phi(T_{2,1})x(t_1) + \Psi(T_{2,1})u_{1,2} \\y(t_2) &= Cx(t_2) + Du_{1,2}\end{aligned}$$

De même on peut montrer facilement que les matrices Φ et Ψ vérifient :

$$\frac{d}{dt} \begin{bmatrix} \Phi(t) & \Psi(t) \\ 0 & I \end{bmatrix} = \begin{bmatrix} \Phi(t) & \Psi(t) \\ 0 & I \end{bmatrix} \begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix}$$

soit en intégrant cette équation différentielle :

$$\begin{bmatrix} \Phi(T) & \Psi(T) \\ 0 & I \end{bmatrix} = \exp\left(\begin{bmatrix} A & B \\ 0 & 0 \end{bmatrix} T\right)$$

C'est à partir de cette formule que la macro Scilab `dscr.sci` discrétise un système continu.

9.3 Stabilité d'un système continu

A partir de la réponse libre on déduit la condition : un système linéaire invariant continu est asymptotiquement stable si et seulement si toutes les valeurs propres de la matrice d'évolution $A \in \mathbb{R}^{n \times n}$ sont à parties réelles négatives. Si l'on connaît la matrice A on détermine facilement ses valeurs propres et donc la stabilité par la commande Scilab `poles=spec(A)`.

9.4 Représentations

9.4.1 Représentation d'état à partir de la transmittance : cas SISO, forme commandable

Le principe revient à faire un graphe où apparaissent des intégrateurs : opérateurs $1/s$ (c'est un schéma de calcul analogique pour les personnes qui ont connu ces calculateurs !) (FIG 9.1).

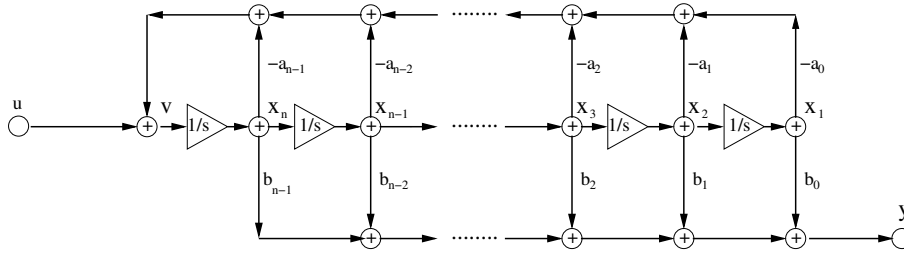


FIGURE 9.1 – Schéma du système : forme commandable

Ce schéma correspond en choisissant une variable intermédiaire v aux équations :

$$\begin{aligned}
 \frac{dx_1}{dt} &= x_2 \\
 \frac{dx_2}{dt} &= x_3 \\
 &\vdots \\
 \frac{dx_{n-1}}{dt} &= x_n \\
 \frac{dx_n}{dt} &= v \\
 v &= u - (a_0x_1 + a_1x_2 + \dots + a_{n-1}x_n) \\
 y &= b_0x_1 + b_1x_2 + \dots + b_{n-1}x_n
 \end{aligned}$$

$$\begin{aligned}
 \frac{dx}{dt} &= \begin{bmatrix} 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n-2} & -a_{n-1} \end{bmatrix} x + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 1 \end{bmatrix} u \\
 y &= [b_0 \ b_1 \ \dots \ b_{n-2} \ b_{n-1}] x
 \end{aligned}$$

Dans cette représentation chaque composante du vecteur d'état est la dérivée de la précédente composante : on a une représentation dans l'espace de phase (on a pris dans ce cas $b_n = 0$, sinon le schéma est à compléter par une liaison directe entre $u(t)$ et $y(t)$).

9.4.2 Passage d'une forme quelconque à la forme commandable

Soit un système décrit par ses équations d'état :

$$\begin{aligned}
 \frac{dx(t)}{dt} &= Ax(t) + Bu(t) \\
 y(t) &= Cx(t) + Du(t)
 \end{aligned}$$

Les matrices A, B, C, D étant quelconques mais de bonnes dimensions et le vecteur d'état étant noté $X(t)$ de dimensions $n \times 1$, on peut moyennant la condition de commandabilité $\text{rang}(C_{(A,B)}) = n$ avec $C_{(A,B)} = [B, AB, \dots, A^{n-1}B]$ déterminer un changement de base défini par une matrice P régulière $x = Px^*$, x^* étant le vecteur d'état dans cette nouvelle base.

On aura ainsi

$$\begin{aligned}
 \frac{dx^*(t)}{dt} &= A^*x^*(t) + B^*u(t) \\
 y(t) &= C^*x^*(t) + Du(t)
 \end{aligned}$$

9 Représentation d'état des systèmes

Si on note $A^* = P^{-1}AP$, $B^* = P^{-1}B$, $C^* = CP$, comme on recherche un modèle sous forme commandable avec

$$A^* = \begin{bmatrix} 0 & 1 & 0 & . & . & 0 & 0 \\ 0 & 0 & 1 & . & . & 0 & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & 1 & 0 \\ 0 & 0 & 0 & . & . & 0 & 1 \\ -a_0 & -a_1 & -a_2 & . & . & -a_{n-2} & -a_{n-1} \end{bmatrix}$$

$$B^* = \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

on aura donc les deux relations : $PA^* = AP$ et $PB^* = B$ soit, si P_i représente la i -ième colonne de P .

$$[P_1 P_2 \cdots P_n] \begin{bmatrix} 0 & 1 & 0 & . & . & 0 & 0 \\ 0 & 0 & 1 & . & . & 0 & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & 1 & 0 \\ 0 & 0 & 0 & . & . & 0 & 1 \\ -a_0 & -a_1 & -a_2 & . & . & -a_{n-2} & -a_{n-1} \end{bmatrix} = A[P_1 P_2 \cdots P_n]$$

$$[P_1 P_2 \cdots P_n] \begin{bmatrix} 0 \\ 0 \\ . \\ . \\ 0 \\ 0 \\ 1 \end{bmatrix} = B$$

Ces deux relations donnent le système d'équations, I_n étant la matrice unité de dimensions $n \times n$:

$$\begin{aligned} P_n &= I_n B \\ P_{n-1} &= (A + a_{n-1} I_n) B \\ P_{n-2} &= (A^2 + a_{n-1} A + a_{n-2} I_n) B \\ &\vdots \\ P_1 &= (A^{n-1} + a_{n-1} A^{n-2} + \cdots + a_1 I_n) B \\ 0 &= (A^n + a_{n-1} A^{n-1} + \cdots + a_1 A + a_0 I_n) B \end{aligned}$$

9 Représentation d'état des systèmes

Cette dernière relation est toujours vérifiée car A vérifie son propre polynôme caractéristique (théorème de Cayley- Hamilton).

Un exemple de mise sous forme commandable.

Soit un système monovarié de modèle d'état :

$$\frac{dx(t)}{dt} = \begin{bmatrix} -2 & +1 \\ +2 & -3 \end{bmatrix} x(t) + \begin{bmatrix} +1 \\ -1 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 2 \end{bmatrix} x(t)$$

montrons qu'il est commandable, et recherchons sa représentation commandable. La matrice de commandabilité est : $C_{(A,B)} = [B, AB] = \begin{bmatrix} +1 & -3 \\ -1 & +5 \end{bmatrix}$ et son déterminant qui vaut 2 est bien différent de zéro, donc il existe une forme commandable pour le système. Soit P la matrice de passage on a : $P_2 = B = \begin{bmatrix} +1 \\ -1 \end{bmatrix}$; $P_1 = (A + a_1 I)B = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$; donc $P = \begin{bmatrix} 2 & +1 \\ 0 & -1 \end{bmatrix}$ et $P^{-1} = \begin{bmatrix} 0.5 & 0.5 \\ 0 & -1 \end{bmatrix}$.

De même le polynôme caractéristique vaut $P(\lambda) = \lambda^2 + 5\lambda + 4$ et donc la nouvelle matrice d'état est $A^* = \begin{bmatrix} 0 & 1 \\ -4 & -5 \end{bmatrix}$ et $B^* = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. Quant au vecteur C^* il vaut $C^* = CP = \begin{bmatrix} 2 & -1 \end{bmatrix}$. La représentation commandable est donc :

$$\frac{dx^*(t)}{dt} = \begin{bmatrix} 0 & 1 \\ -4 & -5 \end{bmatrix} x^*(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 2 & -1 \end{bmatrix} x^*(t)$$

Maintenant vérifions ceci avec Scilab.

```
-->A=[-2,1;2,-3]; B=[1;-1]; C=[1,2];
-->g=syslin("c",A,B,C);//Le système.
-->[Ac,Bc,U,ind]=canon(A,B);//forme canonique de commandabilité.
ind =
2.
U =
0.7071068    1.4142136
- 0.7071068    0.
Bc =
1.4142136
0.
Ac =
- 5. - 4.
1. 0.
-->Ccom=cont_mat(A,B)//Matrice de commandabilité.
```

9 Représentation d'état des systèmes

```

Ccom =
    1. - 3.
- 1.  5.
-->Cc=C*U
Cc =
- 0.7071068 1.4142136
-->rank(Ccom)//Le rang de la matrice de commandabilité.
ans =
2.
La forme n'est pas tout à fait la même mais on a bien là une forme commandable.
On peut aussi passer par la matrice de transfert et trouver une forme commandable :
celle décrite dans l'exemple.
-->gt=ss2tf(g)
gt =
    2 - s
-----
          2
    4 + 5s + s
-->cont_frm(gt.num,gt.den)
ans =
ans(1) (state-space system:)
!lss A B C D X0 dt !
ans(2) = A matrix =
    0.  1.
- 4. - 5.
ans(3) = B matrix =
0.
1.
ans(4) = C matrix =
2. - 1.
ans(5) = D matrix =
0.
ans(6) = X0 (initial state) =
0. 0.
ans(7) = Time domain =
[]
-->ans(7)="c"//Pour retrouver un système continu : bug dans cont_frm,
//ce n'est pas un bug car les arguments d'entrée sont des polynômes.
Une remarque :
Si l'on a défini le système par sa matrice de transfert la matrice de commandabilité
obtenue par la commande cont_mat ne peut être obtenue directement par la commande
cont_mat(gt), on doit passer par la commande cont_mat (tf2ss(gt)).
-->cont_mat(gt)
!--error 10000 cont_mat : Type erroné de l'argument d'entrée n°1 : Un système
linéaire dynamique attendu. at line 26 of function cont_mat called by :

```

```
cont_mat(gt)
gt est bien un système linéaire dynamique : la page du manuel n'est pas très claire!!!
mais :
-->cont_mat(g)
ans =
    1. - 3.
- 1.    5.
-->cont_mat(tf2ss(gt))
ans =
1.2247449 - 8.5732141
1.2247449 - 3.6742346
-->cont_mat(A,B)
ans =
    1. - 3.
- 1.    5.
Alors faites très attention aux résultats. De même effectuez la commande :
-->[n,U,ind,V,Ac,Bc]=contr(A,B)
```

9.4.3 Représentation d'état à partir de la transmittance : cas SISO, forme observable

On peut, comme on l'a vu à la section précédente, par un schéma de calcul analogique, définir le graphe du système en utilisant des opérateurs $1/s$, des intégrateurs.

Reprenons la transmittance

$$\frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0} = \frac{N_{um}(s)}{D_{en}(s)}$$

que l'on peut mettre sous la forme :

$$Y(s) = \left(\frac{b_0}{s^n} + \frac{b_1}{s^{n-1}} + \dots + \frac{b_{n-1}}{s} \right) U - \left(\frac{a_0}{s^n} + \frac{a_1}{s^{n-1}} + \dots + \frac{a_{n-1}}{s} \right) Y$$

cette équation correspond au schéma de calcul (FIG 9.2).

En choisissant une variable d'état x_i comme la sortie du i -ième intégrateur, on aboutit aux équations :

$$\begin{aligned} \frac{dx_1}{dt} &= -a_0 x_n + b_0 u \\ \frac{dx_2}{dt} &= x_1 - a_1 x_n + b_1 u \\ &\vdots \\ \frac{dx_{n-1}}{dt} &= x_{n-2} - a_{n-2} x_n + b_{n-2} u \\ \frac{dx_n}{dt} &= x_{n-1} - a_{n-1} x_n + b_{n-1} u \\ y &= x_n \end{aligned}$$

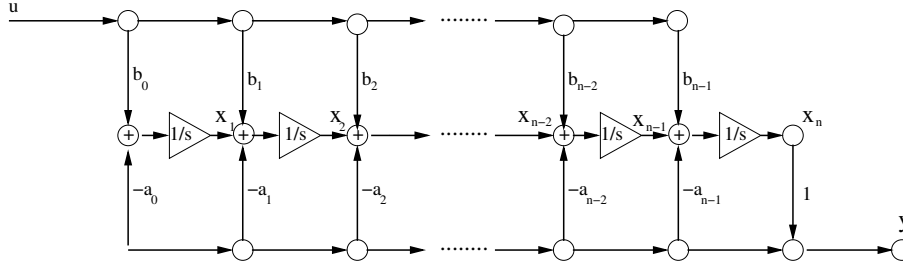


FIGURE 9.2 – Schéma du système : forme observable.

ou encore :

$$\frac{dx}{dt} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & -a_0 \\ 1 & 0 & 0 & \dots & 0 & -a_1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix} x + \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{bmatrix} u$$

$$y = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 \end{bmatrix} x$$

9.4.4 Passage d'une forme quelconque à la forme observable

Comme nous l'avons fait pour la forme commandable on peut trouver moyennant la condition d'observabilité une matrice de passage P permettant d'écrire le système sous la forme précédente. La condition d'observabilité est : $\text{rang}(O_{(A,C)}) = n$ avec

$$O_{(A,C)} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}.$$

La démonstration pour trouver la matrice de passage est analogue à la démonstration effectuée pour la forme commandable (on travaillera avec la transposée de la matrice de passage P).

Si l'on reprend l'exercice précédent on trouve :

$O_{(A,C)} = \begin{bmatrix} 1 & 2 \\ 2 & -5 \end{bmatrix}$ dont le déterminant est non nul (système observable). Le polynôme caractéristique est : $P(\lambda) = \lambda^2 + 5\lambda + 4$, comme précédemment, et les deux vecteurs constitutifs de la matrice de passage sont : $P_2 = C^T = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$ et $P_1 = (A^T + a_1 I)C^T$ soit

$P_1 = \begin{bmatrix} 7 \\ 5 \end{bmatrix}$ donc $(P^{-1})^T = \begin{bmatrix} 7 & 1 \\ 5 & 2 \end{bmatrix}$ et $P^{-1} = \begin{bmatrix} 7 & 5 \\ 1 & 2 \end{bmatrix}$ d'où la valeur de la matrice de passage $P = \begin{bmatrix} \frac{2}{9} & -\frac{5}{9} \\ -\frac{1}{9} & \frac{7}{9} \end{bmatrix}$. Finalement la forme observable du système s'écrit :

$$\frac{dx^*(t)}{dt} = \begin{bmatrix} 0 & -4 \\ 1 & -5 \end{bmatrix} x^*(t) + \begin{bmatrix} 2 \\ -1 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 0 & 1 \end{bmatrix} x^*(t)$$

Exemples Scilab

Calcul de la matrice d'observabilité, (même exercice que précédemment) :

```
-->OB=obsv_mat(A,C)
```

```
OB =
```

```
1.    2.
```

```
2.   -5.
```

```
-->OB=obsv_mat(g)//même résultat à condition que g soit sous forme d'état.
```

```
-->sl0=obsvss(g)//partie observable de g (ici) g lui même.
```

9.4.5 Dualité : observabilité, commandabilité

Soit le système S (continu ou discret), de vecteur d'état $x(t)$, de vecteur d'entrée $u(t)$, de vecteur de sortie $y(t)$, décrit par les équations :

$$\begin{aligned} \frac{dx(t)}{dt} &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned}$$

On dénomme le système dual de S un système S_{dual} (continu ou discret), de vecteur d'état $w(t)$, de vecteur d'entrée $v(t)$, de vecteur de sortie $z(t)$, décrit par les équations :

$$\begin{aligned} \frac{dw(t)}{dt} &= A^T w(t) + C^T v(t) \\ z(t) &= B^T w(t) + Dv(t) \end{aligned}$$

Dans la pratique l'étude de la **commandabilité** de S revient à faire l'étude de **l'observabilité** de S_{dual} , et réciproquement. On peut montrer aussi que quand on fait un changement de variables d'état pour S ($x^* = Px$), pour l'espace dual S_{dual} on réalise le changement $w^* = P_{dual}w$, alors $P_{dual} = (P^{-1})^T$.

En résumé :

Si (A_c, B_c, C_c, D_c) est la réalisation canonique de commandabilité, et (A_0, B_0, C_0, D_0) est la réalisation canonique d'observabilité on a :

$$(A_c, B_c, C_c, D_c) = (A_0^T, C_0^T, B_0^T, D_0)$$

9.5 Forme modale (diagonale)

La forme modale ou forme diagonale fait apparaître les valeurs propres de la matrice d'état sur la diagonale principale : cela revient à mettre la transmittance sous forme éléments simples.

9.5.1 Vecteur d'état de la forme modale

Nous avons ici plusieurs cas à traiter suivant que les pôles de la transmittance, sont réels simples, réels multiples, ou complexes conjugués simples ou multiples.

Reprenons la transmittance :

$$\frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_1 s + b_0}{s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0}$$

que l'on peut présenter sous la forme :

$$\frac{Y(s)}{U(s)} = \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_1 s + b_0}{(s - p_n)(s - p_{n-1}) \cdots (s - p_2)(s - p_1)}$$

où $(p_1, p_2, \dots, p_{n-1}, p_n)$ sont les pôles du système (réels ou complexes conjugués, simples et ou multiples). En calculant les pôles et en faisant la décomposition en éléments simples on aura :

Cas des pôles distincts (simples) réels

$$\frac{Y(s)}{U(s)} = \sum_{i=1}^{i=n} \frac{el_i}{s - p_i} \Rightarrow Y(s) = \sum_{i=1}^{i=n} \frac{el_i}{s - p_i} U(s)$$

soit en temporel le choix des variables d'état :

$$X_i(s) = \frac{U(s)}{(s - p_i)} \Rightarrow sX_i(s) = p_i X_i(s) + U(s)$$

$$\frac{dx_i(t)}{dt} = p_i x_i(t) + u(t) \quad i = 1, \dots, n$$

de même on a :

$$y(t) = \sum_{i=1}^{i=n} el_i x_i(t)$$

9 Représentation d'état des systèmes

cela donne les équations temporelles :

$$\begin{aligned} \frac{dx(t)}{dt} &= \begin{bmatrix} p_1 & 0 & 0 & \cdot & \cdot & 0 & 0 \\ 0 & p_2 & 0 & \cdot & \cdot & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdot & p_{n-1} & 0 \\ 0 & 0 & 0 & \cdot & \cdot & 0 & p_n \end{bmatrix} x(t) + \begin{bmatrix} 1 \\ 1 \\ \cdot \\ \cdot \\ \cdot \\ 1 \\ 1 \end{bmatrix} u(t) \\ y(t) &= [el_1 \cdots el_{n-1} el_n] x(t) \end{aligned}$$

Remarques :

- Cette représentation fait apparaître les modes.
- La matrice de passage P ($A_{diag} = P^{-1}AP$) est constituée des vecteurs propres associés aux valeurs propres correspondantes.

Les pôles sont complexes conjugués

Comme précédemment on prend les deux racines complexes conjuguées avec les deux états correspondants (qui sont complexes!!) et on choisi deux nouveaux états z_1 et z_2 tels que $z_1 = x_1 + x_2$ et $z_2 = j(x_1 - x_2)$ qui eux sont réels, cela conduit aux deux équations :

$$\begin{aligned} \frac{d}{dt}(z_1) &= \sigma z_1 + \omega z_2 + 2u(t) \\ \frac{d}{dt}(z_2) &= -\omega z_1 + \sigma z_2 \end{aligned}$$

Si R_{elm} et $\pm I_{elm}$ représentent la partie réelle et les parties imaginaires des deux résidus on a :

$$y_{1,2} = R_{elm} z_1 + I_{elm} z_2$$

pour la sortie partielle. Ainsi on a pour représentation :

$$\begin{bmatrix} \frac{dz_1}{dt} \\ \frac{dz_2}{dt} \end{bmatrix} = \begin{bmatrix} \sigma & \omega \\ -\omega & \sigma \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} 2 \\ 0 \end{bmatrix} u(t)$$

Les pôles sont multiples

Soit p_m un pôle multiple de multiplicité l . On va choisir les équations d'état de la manière suivante :

$$\begin{aligned}\frac{dx_1}{dt} &= p_m x_1 + x_2 \\ \frac{dx_2}{dt} &= p_m x_2 + x_3 \\ &\vdots \\ \frac{dx_{l-1}}{dt} &= p_m x_{l-1} + x_l \\ \frac{dx_l}{dt} &= p_m x_l + u\end{aligned}$$

ainsi par ce choix on voit apparaître un bloc de Jordan, dans la matrice d'état, ce bloc pour le mode correspondant s'écrit :

$$\begin{bmatrix} \frac{dx_1}{dt} \\ \frac{dx_2}{dt} \\ \vdots \\ \frac{dx_{l-1}}{dt} \\ \frac{dx_l}{dt} \end{bmatrix} = \begin{bmatrix} p_m & 1 & \cdots & 0 & 0 \\ 0 & p_m & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & p_m & 1 \\ 0 & 0 & \cdots & 0 & p_m \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{l-1} \\ x_l \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} u(t)$$

avec pour la sortie partielle correspondante :

$$y_m = \begin{bmatrix} el_{m,1} & el_{m,2} & \cdots & el_{m,l-1} & el_{m,l} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{l-1} \\ x_l \end{bmatrix}$$

9.5.2 passage des équations d'état à la matrice de transfert

Soit le système S (continu ou discret), de vecteur d'état $x(t)$, de vecteur d'entrée $u(t)$, de vecteur de sortie $y(t)$, décrit par les équations :

$$\begin{aligned}\frac{dx(t)}{dt} &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t)\end{aligned}$$

si $X(s)$ la transformée de Laplace de $x(t)$, de même pour $Y(s)$ transformée de Laplace de $y(t)$, on obtient :

$$\begin{aligned}sX(s) - x(0^+) &= AX(s) + BU(s) \\ Y(s) &= CX(s) + DU(s)\end{aligned}$$

Ainsi si le vecteur $x(0^+)$ des conditions initiales est nul on a la relation :

$$\frac{Y(s)}{U(s)} = G(s) = C(sI - A)^{-1}B + D$$

qui représente la matrice de transfert du système (mono ou multivariables).

Exemples Scilab

Le logiciel Scilab possède deux instructions permettant de passer des équations d'état à la fonction de transfert et vice-versa.

Je donne ici l'exemple proposé dans le manuel.

```
-->s=poly(0,"s");
-->h=[1,1/s;1/(s^2+1),s/(s^2-2)]
-->sl=tf2ss(h)//passage la matrice de transfert aux équations d'état.
-->h=clean(ss2tf(sl))//passage état vers transfert.
-->[Ds,NUM,chi]=ss2tf(sl)//idem.
-->clean(NUM)
-->clean(chi)
-->clean(sl(2))//La matrice A
```

La méthode proposée consiste essentiellement au calcul de $(sI - A)^{-1}$ qui se fait en diagonalisant par blocs la matrice A , puis en appliquant l'algorithme de Leverrier sur chacun des blocs. Il existe aussi deux fonctions nommées `coff()` et `nlev()` qui permettent de calculer $(sI - A)^{-1}$, la première fonction calcule cette matrice inverse par la méthode des cofacteurs, la seconde utilise la méthode de Leverrier.

Dernière méthode plus élégante

Cette méthode utilise une propriété particulière des déterminants pour calculer, non pas $(sI - A)^{-1}$ mais l'expression totale recherchée $C(sI - A)^{-1}B$.

Pour un **système monovarié**, on peut à partir du lemme du déterminant d'une somme particulière de matrices, déterminer la transmittance du système. En effet, si H est une matrice carrée $(n \times n)$, U un vecteur colonne $(n \times 1)$, V^T un vecteur ligne $(1 \times n)$, on a :

$$\det(H + UV^T) = \det(H)(1 + V^T H^{-1}U)$$

soit $H = zI - A$ ou $sI - A$, $U = B$ et $V^T = C$

$$C(zI - A)^{-1}B = \frac{\det(zI - A + BC)}{\det(zI - A)} - 1$$

et avec le logiciel Scilab ceci sera réalisé facilement par l'instruction :

`poly(A-BC,"z")/poly(A,"z")-1`. S'il y a un terme de transfert direct entre l'entrée et la sortie (terme D) il faudra le rajouter à la transmittance.

On voit apparaître successivement, pour le numérateur le polynôme caractéristique de la matrice $A - BC$, puis pour le dénominateur celui de A , (voir définition du polynôme caractéristique d'une matrice carrée, paragraphe 2.4.1), un exemple :

9 Représentation d'état des systèmes

```

-->A=[-2,1;2,-3]; B=[1;-1]; C=[1,2]; g=syslin("c",A,B,C);s=%s;
-->g1=poly(A-B*C,"s");//Poly caractéristique de A-B*C.
g1 =
      2
6 + 4s + s
-->g2=poly(A,"s");//Poly caractéristique de A.
g2 =
      2
4 + 5s + s
-->res=g1/g2-1//La transmittance.
res =
      2 - s
-----
      2
4 + 5s + s
-->res.dt="c";
-->ss2tf(g)//Méthode classique de passage état vers transfert.
ans =
      2 - s
-----
      2
4 + 5s + s

```

10 Utilisation du logiciel pour simuler les systèmes à retard pur

Depuis la première version de ce document, une nouvelle boîte à outils « **iodelay toolbox** » est proposée par le Consortium Scilab afin d'étudier fréquemment les systèmes à retard pur (en entrée-sortie).

Mais comme cette boîte à outils ne fonctionne plus avec Scilab-6, j'ai récupéré les principales fonctions que j'ai intégré dans « **Autoelem Toolbox** », merci à Serge Steer pour ce travail.

10.1 Principales fonctions utiles

La majorité des logiciels de simulation posent problème dès que l'on souhaite faire la simulation de systèmes dont le modèle mathématique n'est pas une fraction rationnelle ou un quadruplet $\begin{bmatrix} A & B & C & D \end{bmatrix}$. L'utilisation de la notion de **list** (*type* 15, 16, 17) permet de résoudre d'une manière élégante ce problème. Dans la boîte à outils proposée, un système à retard pur (en entrée-sortie) est défini comme une **mlist** (*type* 17), en associant une matrice de transfert $H(s)$ et une matrice de scalaires le retard d de mêmes dimensions.

L'instruction de création de cette **mlist** est :

```
-->Hd=iodelay(H,d)
```

En réalité cette **mlist** est de la forme $Hd = mlist(["rd", "H", "iodelay"], H, d)$. On définit un nouveau type d'objet : **rd** un rationnel retardé, composé de deux matrices de mêmes dimensions H et d . Différentes opérations sur ces objets sont proposées (les fonctions qui commencent par le caractère % dans le répertoire macros du logiciel).

10.1.1 Définition d'un système à retard pur

En automatique les systèmes retardés (retard entrée-sortie) ont pour modèle mathématique la fonction $g(s) = \frac{N(s)}{D(s)} e^{-ds}$: voir le paragraphe 3.1.5 de ce document.

10.1.2 Quelques fonctions complémentaires

En plus des fonctions de la boîte à outils **iodelay toolbox** je propose, à partir des fonctions contenues dans le logiciel, un ensemble de macros, permettant de compléter l'étude des systèmes à retard pur : un peu comme pour faire l'étude des systèmes sans retard.

Mais avant de faire la description des fonctions proposées, je voudrais faire une remarque que je considère comme très importante .

Changement de variable sur le modèle de départ.

Faisons l'expérience suivante :

```
-->s=%s;g=syslin("c",.5/(s+1));
-->hd=iodelay(g,1.5)
hd =
          0.5
exp(-1.5*s)*-----
          1 + 1s
-->delay = hd.iodelay
delay =
          1.5
-->h1=horner(hd.H,s/delay) //on change la variable s en s/delay.
h1 =
          0.5
-----
1 + 0.6666667s
-->h1.dt = "c";//très important sinon h1 n'est plus un système continu
//cela est du à la fonction horner.
-->hd1=iodelay(h1,1)
hd1 =
          0.5
exp(-1*s)*-----//on fera l'étude de ce système plutôt que hd.
          1 + 0.6666667s
-->nnyquist([hd;hd1],.1,10)//Examiner ces deux lieux.
```

En faisant cette opération j'obtiens deux lieux de Nyquist (de Black) de **formes identiques**, qui se superposent, mais bien sur de graduations différentes : on pourra donc étudier les réponses fréquentielles, non pas sur le système originel, mais sur un système de retard pur égal à 1 (ou tout autre valeur que vous choisirez). Pour retrouver les propriétés du système originel il faudra diviser les fréquences par le retard pur de départ. **Attention** pour les lieux de Bode une translation horizontale de $\log_{10}(\text{delay})$ est à faire pour que les courbes de Bode se superposent.

Cette remarque est aussi valable en temporel en effet d'après une des propriétés de la transformée de Laplace quand on change s en αs , en fréquentiel on change ω en $\alpha \omega$ et en temporel on change le temps t en $\frac{t}{\alpha}$.

C'est à partir de cette remarque que les programmes `g_marginrd`, `m_marginrd` ont été contruits (voir ci après).

Liste des fonctions relatives aux systèmes à retard pur dans Autoelem Toolbox.

Les macros de calcul.

- bodfact** : programme permettant de mettre un système SISO sous forme standard (continu ou échantillonné/discret, **avec ou sans retard**).
- dbphifr** : programme calculant respectivement la fréquence, le gain, la phase des systèmes SIMO continus, **avec ou sans retard**, ou échantillonnés-discrets. On peut aussi balayer le plan complexe avec une droite passant par l'origine et faisant un angle φ avec axe horizontal (voir les exercices de ce document : j'appelle cela les pseudo-lieux, paragraphe 5.2.6).
- fdelay** : **retarde** des signaux (vecteur) d'une valeur donnée (delay) en argument d'entrée.
- m_margin** : marge de module complément de la marge de gain, de phase, pour les systèmes sans retard.
- m_marginrd** : marge de module complément de la marge de gain, de phase, pour les **systèmes avec retard**.
- p_marginrd** : marge de phase des **systèmes à retard**.
- g_marginrd** : marge de gain des **systèmes à retard**.
- mroots** : racines d'un polynôme en particulier les racines multiples (utilisation de deux fonctions scilab **bezout** et **roots**).
- padedelay** : approxime un retard par un rapport de deux polynômes (approximants de Padé, de Laguerre, de Kautz ou de Padé du second ordre au choix). Les approximants de Padé se calculent par récurrence : **spécifique aux systèmes avec retard**.
- thiran** : **approximation d'un retard pur** par un filtre de Thiran (filtre numérique de gain 1), (voir le document et le manuel).
- pade** : approximant de Padé d'une fonction, donnée sous forme de son développement de Mac-Laurin, soit sous forme d'une fonction, enfin soit sous forme d'une chaîne de caractères dont on évaluera la valeur. Dans les deux derniers cas on utilise la **fft** pour trouver le développement de Mac-Laurin de cette fonction (fonction **maclau** intégrée à la fonction **pade.sci**).
- mompon** : moments pondérés d'un rapport de polynômes (utile pour la synthèse des systèmes par la méthode des moments) : développement de Taylor Mac-Laurin d'un vecteur de systèmes (SIMO) autour d'une valeur $s = a$.
- pmark** : paramètres de Markov d'un vecteur de systèmes (SIMO) autour de $s = +\infty$.
- dscr_sisord** : discrétisation (avec BOZ) d'un système SISO à retard.
- dpr_cd** : calcul des matrices fréquences, gains, phases de **deux** vecteurs SIMO de systèmes, l'un continu l'autre échantillonné : on peut ainsi comparer les systèmes continus et leurs homologues échantillonnés (avec ou sans retard pur).
- %...** : fonctions de surcharge celles du logiciel **iodelay** et quelques fonctions complémentaires.

Les macros de dessin.

Ces macros de dessin et de calcul des réponses fréquentielles, utilisent exclusivement le programme `dbphifr`, elles tiennent compte du retard : on peut ainsi tracer des lieux avec et sans retard dans une même fenêtre, et même les pseudo-lieux.

`bbode`, `bblack`, `nnyquist`, `ggainplot`, `pphaseplot`.

Vous trouverez dans la suite du document, un exercice qui va vous permettre de mettre en évidence l'influence du retard pur sur un système du second ordre classique.

Dans cette partie nous nous intéresserons à la simulation de systèmes non bouclés qui possèdent en cascade avec un système classique de transmittance $G(s) = \frac{N(s)}{D(s)}$ ($N(s)$ est un polynôme de degré m et $D(s)$ un polynôme de degré $n \geq m$), un retard pur représenté par l'opérateur $G_r(s) = e^{-ds}$.

Les réponses peuvent être de deux types : soit temporelles, soit fréquentielles.

En ce qui concerne les premières, la création d'une fonction d'entrée retardée dans le temps de d associée à l'instruction `csim()` donne le résultat convenu (voir l'exercice au paragraphe 10.1.3). Si le retard est en sortie, la fonction `fdelay.sci` va permettre de trouver la forme du signal retardé.

10.1.3 Exemple de programme Scilab pour simuler une réponse temporelle à commande retardée : cas SISO

Schéma d'un système à entrées retardées

Soit un vecteur de commande $u(t)$, de dimension m , cette commande est retardée par l'opérateur retard caractérisé par le vecteur $[e^{-d_1s}; e^{-d_2s}; e^{-d_3s}; \dots; e^{-d_ms}]$ en série avec le système multivariable $G(s)$ possédant m entrées et p sorties (FIG 10.1).

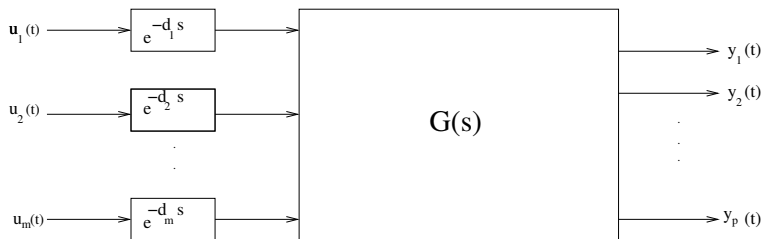


FIGURE 10.1 – Entrées retardées

Je rappelle les méthodes déjà rencontrées dans ce document afin de créer un signal d'entrée.

Création par Scilab d'un signal d'entrée retardé à l'aide d'une fonction

Je crée un signal $u(t)$, nul avant l'instant *delay* puis valant 1 de *delay* à *delay* + T_1 et revenant ensuite à 0. Voici la fonction nommée `creneau.sce` (fonction stockée dans le répertoire `.../pointsce`) :

```
function [ut]=creneau(t,delay,T1)
ut=bool2s(t>delay)-bool2s(t>(delay+T1));
endfunction
et voici le programme principal :
-->s = %s;g = syslin('c',1/(1+s));
-->sl = iodelay(g,.5);
-->t = linspace(0,10,101);
-->delay = sl.iodelay;T1 = 2;
-->exec("/home/.../pointsce/creneau.sce");
-->y=csim(creneau,t,sl.H);//on peut rajouter une condition initiale.
-->plot(t,y)
```

Les grandeurs *delay* et T_1 représentent respectivement le retard pur et la durée du créneau. Ne voulant pas le définir dans la fonction, afin de conserver toute la généralité, je donne ici à *delay* et à T_1 dans le programme principal (variables globales) les valeurs 0.5 et 2. On charge la fonction **creneau**, puis on exécute la simulation par l'instruction **csim**.

Attention : il faut dans **csim** donner le nom de la fonction d'entrée, ici **creneau** et non le résultat, que j'ai appelé *ut*. En effet *ut* est de type 1 (scalaire), tandis que **creneau** est de *type 13* (fonction compilée) et **csim** accepte pour entrée un argument de type : **function** ou **list**.

Mais on peut aussi faire :

```
-->ut=creneau(t,delay,T1);//puis
-->y=csim(ut,t,sl.H);//csim l'accepte.
```

Création par Scilab d'un signal en utilisant la notion de liste (surchage d'opérateur)

Cette deuxième façon de procéder utilise la notion de liste, manière élégante de surcharger un opérateur : ici une fonction¹.

```
-->t=linspace(0,10,101);
-->exec("/home/.../Projets/pointsce/creneau.sce");
-->ul=list(creneau,.5,2)
ul =
    ul(1)
[u]=function(t,delay,T1)
    ul(2)
0.5
    ul(3)
2.
-->y=csim(ul,t,sl.H);
Warning redefining function: ut
-->plot(t,y)
```

1. Voir l'article Scilab, de Serge Steer dans la revue Linux Magazine 14 de février 2000.

La liste `ul` est constituée de trois éléments : la fonction, et les paramètres *delay* et *T1*.

Création par Scilab d'un signal en ligne de commande.

On peut aussi, sans définir un sous programme, créer le signal retardé à la ligne de commande : voici un exemple.

```
-->deff("u=creneau(t,delay,T1)","u=bool2s(t>=delay)-bool2s(t>(delay+T1))")
-->s=%s;g=syslin('c',1/(1+s));sl=iodelay(g,.5);
-->t=linspace(0,10,101);
-->delay=sl.iodelay;T1=2;
-->y=csim(creneau,t,sl.H);
-->plot2d(t',y')
```

Comme précédemment, vous devez utiliser dans la fonction `csim` l'expression `creneau` et non pas le vecteur `u`, car ce vecteur est un scalaire.

Une autre façon de procéder est de créer dans la fenêtre Scilab la fonction par la syntaxe :

```
-->function [u]=creneau(t,delay,T1)
-->u=bool2s(t>delay)-bool2s(t>(delay+T1));
-->endfunction
```

Le programme principal ne change pas.

La même fonction `bool2s` permet de construire un échelon retardé :

```
-->deff("u=echretard(t,delay)","u=bool2s(t>delay)")
```

puis d'utiliser cette fonction pour réaliser la réponse à un échelon retardé.

10.1.4 Exemple de programme Scilab pour simuler une réponse temporelle à sortie retardée

J'ai créé une petite fonction permettant de retarder d'une valeur de temps donnée un ensemble de signaux quelconques (matrice de données de dimensions $k \times l$) : cette fonction se nomme `fdelay.sci` (ce programme est valable avec un vecteur de données, mais avec le même retard).

Voici le programme pour avoir une réponse retardée : retard en sortie (retard sur capteur). On discrétise le temps en prenant une discrétisation sous multiple du retard (FIG 10.2).

```
//réponse à un échelon
//On discrétise le temps en prenant un deltat sous multiple du retard.
-->delay = 4.7; t = [0:delay/20:4*delay];
-->s = %s;
-->hd = iodelay(syslin("c",1/(1+s*s*s)),delay);
-->sys = hd.H;
-->f = csim("step",t,sys);//réponse indicielle du système non retardé.
//réponse retardée.
-->[fretard,t1] = fdelay(f,t,delay);//lire l'aide à fdelay.
-->t1($) == t($)
```

```

ans =
  T // Ici t1 == t.
//tracé des réponses retardée et non retardée.
-->plot2d([t1',t'],[fretard',f'],style=[6,3])
// plot2d(t',[fretard',f'],style=[6,3])
-->legend(["retard 4.7 s";"sans retard"]);
-->xgrid(13);
Maintenant une discrétisation quelconque : on donne un pas de 0.2seconde, non
sous multiple de retard.
-->delay = 4.7; t = [0:0.2:4*delay];
-->f = csim("step",t,sys);
-->[fretard,t1] = fdelay(f,t,delay);
-->t1($) == t($)
ans =
  F // Ici t1 <> t alors attention à plot2d.
plot2d([t1',t'],[fretard',f'],style=[6,3])
//et non plot2d(t',[fretard',f'],style=[6,3])
//le reste est identique.

```

1. Cette fonction présente donc l'inconvénient suivant : si le retard n'est pas un multiple du pas de discrétisation temporel, l'argument retourné `t1`, le temps, n'est plus équiréparti : c'est le deuxième exemple.
2. On peut rééchantillonner le signal en choisissant un période de temps sous multiple du retard, mais cela va demander d'approximer (par interpolation avec des splines par exemple) le signal puis de le rééchantillonner. Vous trouverez dans la boîte à outil **cardiovascular toolbox** sur le site **Atoms** de Scilab une fonction qui réalise ce travail : `function [tn,xn]=Resample(t,x,step,tbounds)`.
3. Enfin la solution la plus élégante est de choisir, si possible, une discrétisation du temps, `t` sous multiple du retard par exemple : `t=[0:delay/20:4*delay]`; et alors dans ce cas, le retard vaut 20 fois le pas de discrétisation du temps : il suffit donc de retarder la réponse de 20 pas : c'est le premier exemple donné.
4. On peut aussi faire le changement de variable s en $\frac{s}{\text{delay}}$ proposé au paragraphe 11.1.2, on se ramène ainsi à un retard d'une seconde, on discrétise le temps comme précédemment, on calcule la réponse et on fait sur cette réponse le changement de variable inverse s en $s.\text{delay}$: l'ancien vecteur temps t pour $\text{delay} = 1$ devient un vecteur temps $t_{\text{nouv}} = \text{delay} \times t$.

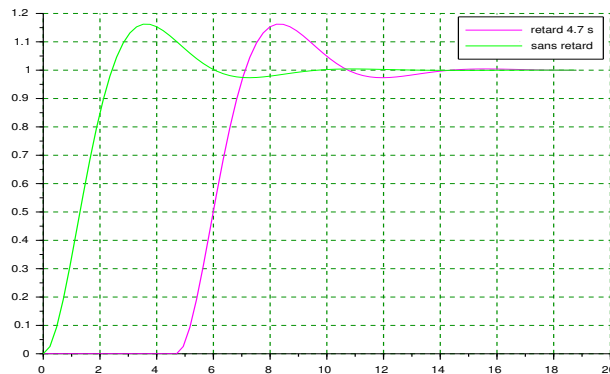


FIGURE 10.2 – Réponses à un échelon.

10.1.5 Réponse fréquentielle

Dès que l'on aborde avec des logiciels de simulation les réponses fréquentielles des systèmes à retard pur, on doit dans le calcul de la réponse fréquentielle rajouter à la phase de $G(s)$ un terme en $-2\pi fd$ (f est la fréquence et d le retard). On doit donc tout d'abord définir un système retardé : ceci peut être fait en utilisant la notion de liste. Ainsi on crée une `mlist` constituée d'une matrice de systèmes continus et d'une matrice de constantes positives (matrice des retards en série avec les systèmes).²

```
-->s=%s;
-->s1=1/(1+s*s*s);
-->s2=(1+s)/(1+s+2*s*s);
-->sl=syslin('c',[s1;s2])
-->slr=iodelay(sl,[.5;.2])
slr =
      1
exp(-0.5*s)*-----
                2
      1+ 1s + 1s
      1 + 1s
exp(-0.2*s)*-----
                2
      1 + 1s + 2s
-->type(slr)
ans =
```

2. Dans la précédente édition de ce document et avant la venue de `iodelay toolbox`, j'avais introduit une liste de `type 15` comprenant un vecteur de systèmes continus et un vecteur de retard, ce qui permettait d'étudier les systèmes à entrées-sorties retardées en fréquentiel.

```

17.
-->slr.H //les systèmes sans retard
ans =
      1
      -----
      2
      1 + s + s
      1 + s
      -----
      2
      1 + s + 2s
->slr.iodelay//la matrice des retards
ans =
      0.5
      0.2

```

Cette façon de procéder est très élégante, mais a demandé de transformer légèrement les principales fonctions intervenant dans l'étude des systèmes, afin de faire l'étude des systèmes à entrées-sorties retardées. Voici les principales fonctions utiles.

Les macros provenant de la boîte à outils iodelay toolbox

Création d'un système à entrée/sortie retardé (voir iodelay1.sce et iodelay2.sce).

```

s=poly(0,'s');
//Fonction de transfert classique.
h=syslin('c',(s^2+2*0.9*10*s+100)/(s^2+2*0.3*10.1*s+102.01))
//On rajoute un retard entrée/sortie de 0.01s.
hd1=iodelay(h,0.01)
//Autre système retardé.
hd2=iodelay(h+1,0.0133)//ici hd2.H vaut h+1.
//Connexion en série.
hds=hd1*hd2// De deux systèmes retardés.
hds2=h*hd2//Un système classique et un retardé.
//Connexion en parallèle : même retard pour les deux systèmes.
hdp= hd1+iodelay(h+1,0.01)
//Système SIMO (voir exercice précédent).
//Assignation (valable pour système sans retard et le retard)
hds(1,1) = h //ou hds.H=h
hds.iodelay = 1

```

Les macros classiques : Programmes de surcharge %....

Les fonctions de calcul : calfrqrd, freqd, repfreqrd.

Un exemple de tracé de lieux de systèmes retardés

```

-->s=%s;sl=syslin("c",1/(1+s*s*s));
-->SL=iodelay([sl;sl;sl;sl;sl],[0;.1;.2;.5;.8;1]);
-->com=["T=0";"T=0.1";"T=0.2";"T=0.5";"T=0.8";"T=1"];
-->nnyquist(SL,0.1,1,com)

```

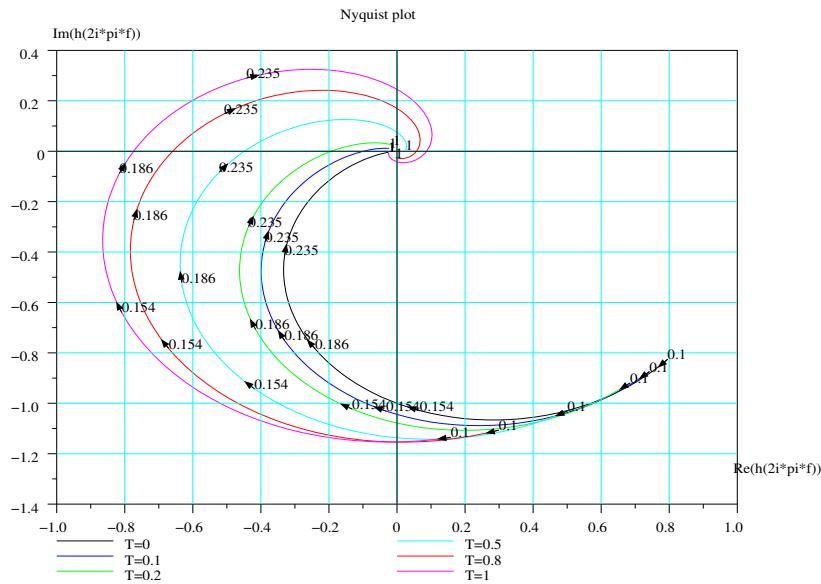


FIGURE 10.3 – Lieux de Nyquist avec retard

```
-->figure(1);
-->bbode(SL,0.1,1,com)
-->figure(2);
-->bblack(SL,0.1,1,com)
```

Remarque :

A la deuxième ligne du programme, on peut écrire :

```
-->SL=iodelay(ones(6,1)*s1,[0;.1;.2;.5;.8;1]);
```

ou

```
-->SL=iodelay(s1*ones(6,1),[0;.1;.2;.5;.8;1]);
```

Je reproduis ces trois lieux sur les figures ci-dessous (FIG. 10.1, 10.2, 10.3).

Une explication sur ces différentes commandes est nécessaire. Par la première instruction je construis une *mlist* (*type 17* la commande *iodelay*) contenant deux vecteurs de même dimensions : le premier vecteur colonne est constitué de cinq fois le système linéaire *s1* (ici je ne cherche qu'à étudier l'influence d'un retard pur sur un système), quant au deuxième vecteur, c'est un vecteur de scalaires en colonne de dimension cinq donnant respectivement la valeur du retard à associer au système correspondant. Une remarque : vous n'aurez plus tout à fait les mêmes dessins, car j'utilisais dans la première version de ce document une liste classique de *type 15* mais les résultats sont les mêmes.

FIGURE 10.4 – Lieux de Bode avec retard

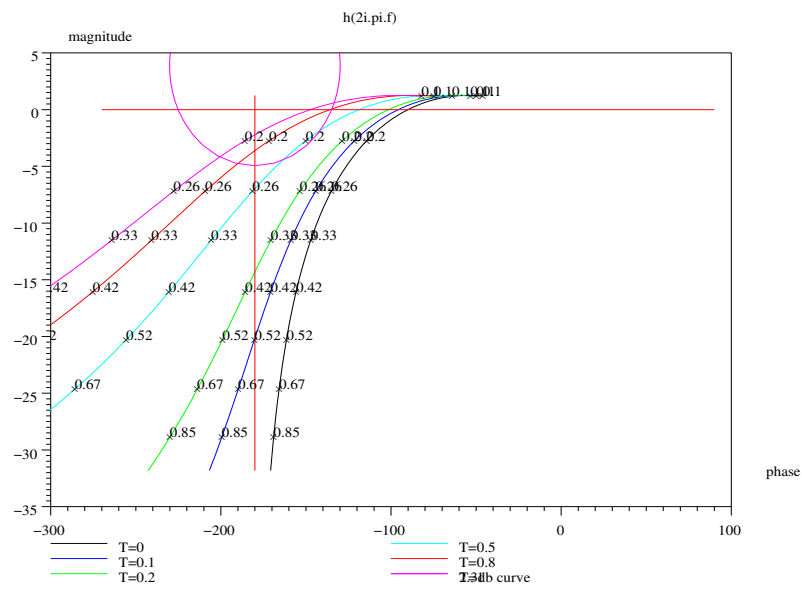
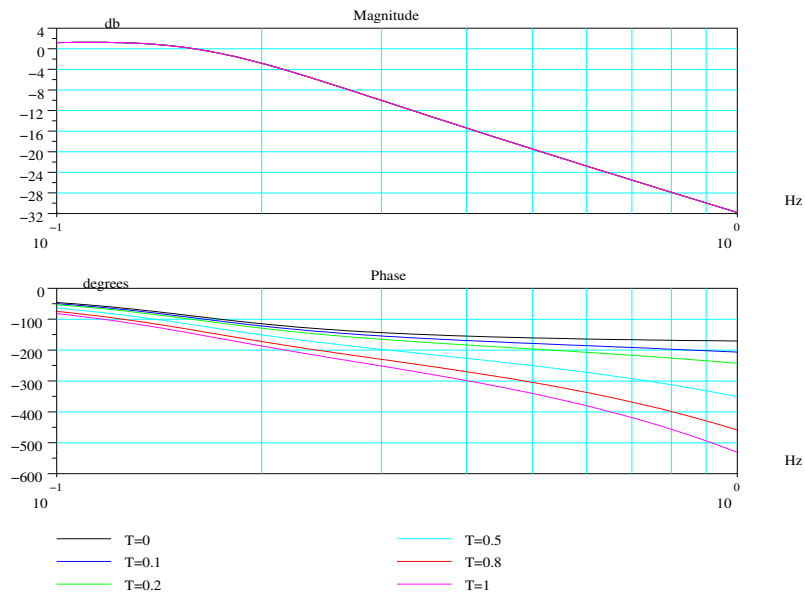


FIGURE 10.5 – Lieux de black avec retard

10.1.6 Les systèmes à retard pur : les approximations

On cherche avec ces approximations à approcher fréquemment le retard pur e^{-ds} en utilisant un rapport de deux polynômes de la variable s (transmittance classique $gd(s)$).

Pour ce faire on devra vérifier les deux propriétés suivantes :

1. Le module de $gd(s)$ pour $s = j\omega$ devra être égal à 1, et ceci quel que soit la fréquence.
2. L'argument de $gd(s)$ pour $s = j\omega$ devra décroître linéairement en fonction de la fréquence.

Il n'existe pas en temps continu des transmittances classiques vérifiant ces deux conditions quel que soit la fréquence : on est donc amené à approximer le retard avec un rapport de deux polynômes dont le module vaut 1 et dont l'argument approchera une fonction linéaire de la fréquence dans une gamme assez grande de fréquences. Pour ce faire on utilisera les approximants de Padé ou d'autres approximants tel que les approximants de Laguerre, de Kautz ou de Padé du second ordre (ces deux derniers étant des rapports de polynômes du second degré élevés à une puissance n). Tous ces approximants doivent avoir des pôles stables et avoir des zéros symétriques des pôles par rapport à l'axe imaginaire pur pour que leurs gains soient unitaires.

Approximants de Padé : programme `padedelay`

Ils sont issus du développement de e^{-s} autour de $s = 0$ (développement de Mac-Laurin). Mais au lieu d'utiliser la méthode classique de recherche d'un approximant de Padé d'une fonction par ce développement autour de $s = 0$, je propose un programme `padedelay.sci` qui utilise une récurrence sur les coefficients du numérateur et du dénominateur de la fraction rationnelle approximant de Padé de degré m pour le numérateur et de degré n pour le dénominateur : voici ces récurrences.

$b_{j+1} = -\frac{m-j+1}{j(n+m-j+1)}b_j$ $b_1 = 1$ $j \in [1, m]$. Ceci concerne les coefficients b_j du numérateur de degré m .

$a_{j+1} = \frac{n-j+1}{j(n+m-j+1)}a_j$ $a_1 = 1$ $j \in [1, n]$. Pour les coefficients a_j du dénominateur de degré n .

(Ces récurrences sont valables pour e^{-s} , pour un retard quelconque d , on recalculera les coefficients en utilisant le programme Scilab `horner.sci`, changement de variable de s en ds).

Approximants de Laguerre

Dans ce cas une formule donne les approximants de Laguerre d'ordre n et le degré du numérateur est le même que celui du dénominateur.

$$Lag_n(s) = \frac{(1-\frac{s}{2n})^n}{(1+\frac{s}{2n})^n} \text{ toujours pour } e^{-s}.$$

Approximants de Kautz

Ici on a affaire à des approximants dit du second ordre, pour e^{-s} on a :

$Kautz_n(s) = \frac{(1 - \frac{s}{2n} + \frac{1}{2}(\frac{s}{2n})^2)^n}{(1 + \frac{s}{2n} + \frac{1}{2}(\frac{s}{2n})^2)^n}$ Les degrés du numérateur et du dénominateur sont égaux à $2n$.

Approximants de Padé du second ordre

$$Pade_{2n}(s) = \frac{(1 - \frac{s}{2n} + \frac{1}{3}(\frac{s}{2n})^2)^n}{(1 + \frac{s}{2n} + \frac{1}{3}(\frac{s}{2n})^2)^n}.$$

Exemple de programme Scilab utilisant ces approximations

Le programme proposé est utilisable avec un système multivariable.

```
-->s=%s;sl=syslin('c',1/(1+s*s*s));
-->SL=iodelay([[sl;sl;sl],[sl;sl;sl]],[[0;.1;.2],[.5;.8;1]])
-->//je n'affiche pas les systèmes.
-->np=[[2;1;2],[1;2;3]];//np matrice des degrés des approximants
//de Padé ayant les mêmes dimensions que SL.
-->SLPAD=padedelay(SL,np)//mp absent : donc même degré pour les
//numérateurs et dénominateurs des approx du retard
SLPAD =
```

$\frac{1}{1 + s + s^2}$	$\frac{1 - 0.25s}{1 + 1.25s + 1.25s^2 + 0.25s^3}$
$\frac{1 - 0.05s}{1 + 1.05s + 1.05s^2 + 0.05s^3}$	$\frac{1 - 0.4s + 0.0533333s^2}{1 + 1.4s + 1.4533333s^2 + 0.4533333s^3 + 0.0533333s^4}$

```
//etc je ne reproduis pas la troisième ligne.
Avec cet exemple on a remplacé les retards par leurs approximants de Padé.
On peut, tout en conservant des approximants de Padé choisir des degrés différents
pour les numérateurs et dénominateurs, mais alors le module complexe de ces approxi-
mants n'est plus égal à 1 (voir plus loin la réponse temporelle du retard pur et de ses
approximants).
Remarques : essayer les instructions suivantes :
-->SLPAD=padedelay(SL,np,"L");// et
-->SLPAD=padedelay(SL,np,"K");// et
-->SLPAD=padedelay(SL,np,"P")
Maintenant vous pouvez comparer les différents lieux (colonne par colonne) : un
exemple simple.
-->SL12=SL(1,2)
-->SLPAD12=SLPAD(1,2)
-->bode([SL12;SLPAD12],.1,10,["SL12";"SLPAD12"])
-->SL12.iodelay
ans =
    0.5
```

On vérifie bien que les deux courbes de gain se superposent et que les deux courbes de phase divergent fortement à partir de 0.5 hertz (ordre de grandeur 14° en utilisant les datatips), le retard est de 0.5 seconde et l'approximant est de degré 1.

Etude temporelle de l'approximant de Padé.

Dans cette section nous allons comparer les réponses temporelles du retard pur et des approximations de ce retard.

```
-->s = %s;
-->uns = syslin("c",1,poly(1,"s","c"))//Syslin 1(s)
uns =
1
-
1
//Autre façon de faire :
-->uns=1/s^0

uns =
1
-
1
-->uns.dt="c"
-->expon = iodelay(uns,1.8)
//Système 1(s) retardé de 1,8 s donc le retard correspondant.
expon =
      1
exp(-1.8*s)*-
      1
->exponpad3 = paddedelay(expon,3)//Approximant 3x3.
exponpad3 =
              2          3
1 - 0.9s + 0.324s - 0.0486s
-----
              2          3
1 + 0.9s + 0.324s + 0.0486s
-->delay = expon.iodelay
delay =
1.8
-->deff("u=echretard(t,delay)","u=bool2s(t>delay)")
//Définition d'un échelon retardé.
-->t1=[0:0.02:4*delay];
//Base de temps, deltat1, sous multiple du retard.
-->yexpon=echretard(t1,delay);//Réponse retard pur de 1.8 s.
-->ypad3=csim("step",t1,exponpad3);
//Réponse de l'approximant de Padé 3x3.
```

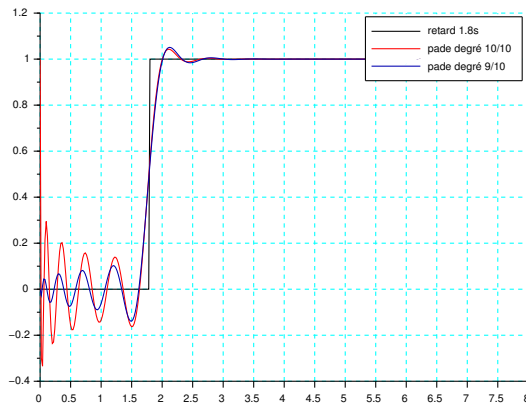


FIGURE 10.6 – Réponse à l'échelon d'un retard pur et ses approximations.

```
-->plot2d(t1', [yexpon', ypad3'], style=[6,3])
// Ici comparaison des réponses.
```

Bien sûr par cette méthode on peut comparer l'échelon retardé, avec différentes approximations du retard, amusez vous, mais ne prenez pas un approximant de Padé de degré trop élevé sinon lors de la simulation, vous aurez un résultat incorrect (ici $n = 10$ est le maximum pour e^{-s}) : on est limité par la fonction `clean()` au terme de plus haut degré de l'approximant à $1.e - 10$, voir programme `padedelay()`.

Remarque :

Réaliser les simulations suivantes (suite de l'exercice) : résultat FIG 10.6.

```
-->exponpad10=padedelay(expon,10)
-->exponpad10_9=padedelay(expon,10,9)
-->ypad10=csim("step",t1,exponpad10);
-->ypad10_9=csim("step",t1,exponpad10_9);
-->plot2d(t1', [yexpon', ypad10', ypad10_9'], style=[1,5,10])
-->xgrid(4);
-->legend(["retard 1.8s", "pade degré 10/10", "pade degré 9/10"]);
```

L'approximant de Padé de degré $m = 9$ au numérateur et de degré $n = 10$ au dénominateur a une plus « belle » dynamique autour de l'origine que l'approximant $m = 10, n = 10$: mais tracez les lieux fréquentiels et comparez.

10.2 La stabilité, les marges de stabilité des systèmes à retard

Comme nous l'avons vu avec les systèmes sans retard, nous pouvons définir la stabilité et les différentes marges de stabilité.

En toute rigueur l'étude de la stabilité des systèmes à entrée-sortie retardés bouclés, demande de faire l'étude de fonctions qui ne sont plus des polynômes en s mais des expressions contenant la variable s et la variable e^{-ds} : on a affaire à des **quasi-polynômes**.

10.2.1 La stabilité

Mais contrairement aux systèmes classiques, l'étude de la stabilité des systèmes bouclés, ne peut pas être faite avec le critère de Routh-Hurwitz. On doit donc soit utiliser une méthode de Ljapunov ou utiliser le lemme de Cauchy (paragraphe 5.1.4) ou encore appliquer les critères donnés ci-après. A partir de ce lemme on peut donner différents théorèmes aboutissant à la stabilité.

10.2.2 Etude de la stabilité des systèmes bouclés à retard [4]

Comme le proposent les auteurs du livre précédemment mis en référence, le système retardé considéré (régime libre) a pour équation :

$$\frac{dX}{dt} = AX(t) + BX(t-d), \text{ avec } X \in R^n, d > 0.$$

On dira que $X(t)$ est le **vecteur d'état instantané**. L'état du système est le vecteur de fonctions $X_t(\theta) = \{X(t+\theta), -d \leq \theta < 0\}$. Quant au vecteur initial $X_0(\theta)$ c'est un vecteur contenant n fonctions définies et continues sur l'intervalle $[-d, 0]$. Pour les curieux on trouvera un modèle analogue, par le calcul opérationnel appliqué aux équations différentielles à argument retardé [9] (quand les coefficients de cette équation sont constants).

On montre qu'en étudiant l'équation caractéristique donnée par l'expression :

$$p(s, d) = \det(sI - A - Be^{-ds}) = 0$$

le système **retardé est stable si et seulement si son équation caractéristique ne possède pas de zéros dans le demi plan droit du plan complexe**, même à l'infini.

10.2.3 Résultats théoriques : méthode de Hermite-Biehler

Considérons un système linéaire retardé (régime libre) dont le modèle est de la forme :

$$\frac{dX}{dt} = AX(t) + \sum_{i=1}^n B_i X(t-d_i)$$

Les matrices A et B_i sont de dimensions adéquates, les d_i représentent les retards. Son équation caractéristique s'écrit donc :

$$p(s) = \det(sI_n - A - \sum_{i=1}^n B_i e^{-d_i s}).$$

Cette relation est un polynôme en s et en $e^{-d_i s}$: c'est un quasi-polynôme ou polynôme exponentiel. Dans le cas où les retard d_i sont des multiples d'un même réel d le système est dit à retards commensurables et alors l'équation caractéristique s'écrit : $p(s) = \det(sI_n - A - \sum_{i=1}^n B_i e^{-ids})$ et le quasi-polynôme est fonction de deux variables seulement s et e^{-ds} .

Un exemple

Soit un système bouclé à retour unitaire de transmittance de chaîne d'action $G(s) = \frac{N(s)}{D(s)} = \frac{e^{-2s}}{s^2 + s + 1}$, la transmittance du système bouclé est donc $W(s) = \frac{1}{s^2 + s + 1 + e^{-2s}} e^{-2s}$ et l'équation caractéristique est un quasi-polynôme qui a pour expression : $p(s) = s^2 + s + 1 + e^{-2s}$. L'étude de la stabilité du système bouclé revient à étudier les racines de $p(s) = 0$, mais les racines de cette équation sont en **nombre infini**. On peut remarquer que $p(s)$ s'écrit sous deux formes :

$$p(s) = (s^2 + s + 1)e^{-0s} + 0e^{-s} + 1e^{-2s}$$

ou

$$p(s) = (1e^{-0s} + 0e^{-s} + 1e^{-2s})s^0 + (1e^{-0s} + 0e^{-s} + 0e^{-2s})s^1 + (1e^{-0s} + 0e^{-s} + 0e^{-2s})s^2$$

soit plus généralement $p(s) = \sum_{i=1}^n p_i(s)e^{-ids}$ ou $p(s) = \sum_{k=1}^{k=m} q_k(s)s^k$.

La stabilité par le théorème de Hermite-Biehler

On peut montrer que les racines de $p(s) = 0$ sont les racines de $e^{ds}p(s)$ et ainsi, dans l'exemple choisi, on doit étudier l'expression $p(s) = (s^2 + s + 1)e^{2s} + 1 = 0$.

Conditions de stabilité

Soit un quasi-polynôme $p(s, e^s)$ ayant un **terme principal**, ce qui veut dire un terme maximum en s et en e^{ns} , pour que toutes les racines de $p(s, e^s) = 0$ soient situées dans le demi plan gauche du plan complexe il **faut et suffit** que les deux conditions suivantes soient vérifiées :

1. Les racines de $p_r(\omega)$ et $p_i(\omega)$ doivent être réelles, simples et alternées ($p(j\omega) = p_r(\omega) + jp_i(\omega)$).
2. Il existe une valeur de ω contenue dans l'intervalle $]-\infty, +\infty[$ où $p'_i(\omega)p_r(\omega) - p_i(\omega)p'_r(\omega) > 0$, avec $p(j\omega) = p_r(\omega) + jp_i(\omega)$ et $p'_i(\omega)$ et $p'_r(\omega)$ respectivement les dérivées de $p_i(\omega)$ et $p_r(\omega)$ par rapport à ω . Cette dernière condition traduit la croissance de la phase de $p(j\omega)$ quand ω varie de $]-\infty, +\infty[$.

10.2.4 Méthode de Walton et Marshall

L'idée de base de ce critère de stabilité est de faire l'étude d'un lieu des racines de l'équation $p(s) = \det(sI_n - A - \sum_{i=1}^n B_i e^{-d_i s}) = 0$ quand les retards varient. Pour un

retard seul cette équation devient : $p(s, d) = D(s) + N(s)e^{-ds}$ quand la boucle ouverte vaut $G(s) = \frac{N(s)}{D(s)}e^{-ds}$. Les polynômes $N(s)$ et $D(s)$ sont à coefficients réels avec $\deg(N) \leq \deg(D)$. Afin d'étudier la stabilité de la boucle fermée, on va rechercher s'il existe des valeurs du retard d pour lesquelles l'équation caractéristique admet des racines imaginaires pures : pour ces valeurs, un changement de comportement asymptotique se produira. La procédure est donc la suivante :

1. Déterminer la position dans le plan complexe des racines de $p(s, 0) = 0$, (utiliser le critère de Routh par exemple).
2. Calculer le polynôme en ω^2 , $q(\omega^2) = D(j\omega)D(-j\omega) - N(j\omega)N(-j\omega)$, obtenu à partir de $p(s, d) = p(\bar{s}, d) = 0$ pour des valeurs de $s = j\omega$.
3. Rechercher s'il existe des racines réelles positives de $q(\omega^2) = 0$, qui sont aussi racines de $p(s, d) = 0$.
4. Etudier le comportement du lieu des racines pour ces valeurs particulières de ω , ω_k . Si $q(\omega_k^2)$ traverse l'axe des abscisse ω^2 de haut en bas, alors $p(s, d_k)$ traverse l'axe imaginaire du plan des racines, de la droite vers la gauche : on a donc stabilité de la boucle à partir de $d = d_k$ (pour des valeurs supérieures à d_k) et ceci jusqu'à la prochaine valeur de ω trouvée.

Une remarque :

Le polynôme $q(\omega^2) = D(j\omega)D(-j\omega) - N(j\omega)N(-j\omega)$, peut être étudié de la même manière pour un système avec ou sans retard en transformant légèrement le programme scilab `kpure.sci`.

Voici l'illustration avec Scilab des exemples proposés par [4].

Premier exemple

La boucle ouverte a pour transmittance $G(s) = -\frac{e^{-ds}}{4(s+2)^2}$ ainsi le quasi-polynôme s'écrit $p(s, d) = (s+2)^2e^{ds} - \frac{1}{4}$. Appliquons les quatres règles :

1. Le système sans retard : ($d = 0$) $p(s, 0) = (s+2)^2 - \frac{1}{4} = s^2 + 4s + \frac{15}{4}$, les deux racines de l'équation $p(s, 0) = 0$ sont $s_1 = -\frac{5}{2}$ et $s_2 = -\frac{3}{2}$, et donc situées dans le demi plan gauche du plan complexe (modes stables).
2. Formons le polynôme en ω^2 , $q(\omega^2) = D(j\omega)D(-j\omega) - N(j\omega)N(-j\omega)$ soit $q(\omega^2) = (\omega^2 + \frac{15}{4})(\omega^2 + \frac{17}{4})$.
3. L'équation $q(\omega^2) = 0$ n'a pas de racines réelles positives.
4. Le système est **stable quelque soit le retard d** .

Les résultats avec Scilab : on prendra pour d les valeurs $[0.1, 1, 10]$.

```
-->s=%s; g3=syslin("c",-0.25,(s+2)^2)
g3 =
- 0.25
-----
                2
4 + 4s + s
-->g1d01=iodelay(10*g3,0.1);
```

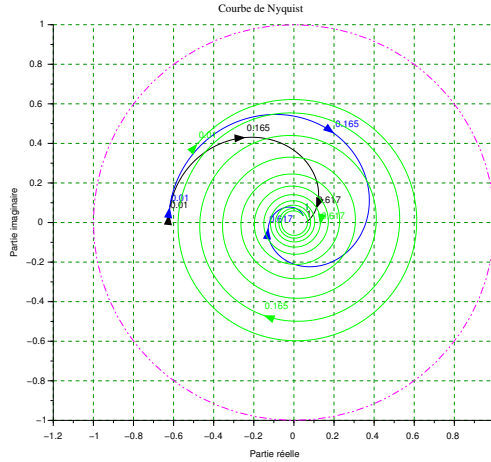



FIGURE 10.7 – Lieux de Nyquist $10 \frac{N(s)}{D(s)} e^{-ds}$, premier exemple.

```
-->g1d1=iodelay(10*g3,1);
-->g1d10=iodelay(10*g3,10);
-->nnyquist([g1d01;g1d1;g1d10],.01,5,..
["retard 0.1 s";"retard 1 s";"retard 10 s"])
```

Pour le dessin j'ai rajouté un gain K_p de 10 dans la chaîne d'action (régulateur proportionnel), la boucle fermée est encore stable.

Maintenant traçons le cercle de rayon unité centré en $[0, 0]$ et de rayon 1.

```
-->xarc(-1,1,2,2,0,64*360)
-->arc=get("hdl");
-->arc.line_style=6;arc.foreground=6;
```

Vous voyez que tous les lieux sont dans le cercle : il n'y a pas d'intersection du cercle de rayon unité (e^{ds} (quand $s = j\omega$), ω variant de 0 à $+\infty$), et le lieu $10 \frac{N(s)}{D(s)}$ (ceci revient à prendre le critère de Cypkin) FIG 10.7.

Second exemple.

Ici $G(s) = \frac{1}{s^3 + s^2 + 2s + 1} e^{-ds}$, et donc l'équation caractéristique vaut $p(s, d) = s^3 + s^2 + 2s + 1 + e^{-ds} = 0$, appliquons les règles.

1. Le système sans retard est stable asymptotiquement car $p(s, 0) = (s + 1)(s^2 + 2)$ donne des pôles $s_1 = -1$, $s_2 = j\sqrt{2}$, $s_3 = -j\sqrt{2}$.
2. L'équation en ω^2 : $q(\omega^2) = D(j\omega)D(-j\omega) - N(j\omega)N(-j\omega)$ soit $q(\omega^2) = \omega^2(\omega^2 - 1)(\omega^2 - 2)$. Les racines positives ou nulle sont : $\omega_1 = 0$, $\omega_2 = 1$, et $\omega_3 = \sqrt{2}$.
3. On vérifie que $\omega = 0$ donc $s = 0$ n'est pas racine de $p(s, d) = s^3 + s^2 + 2s + 1 + e^{-ds} = 0$, on ne la prend pas.

10 Utilisation du logiciel pour simuler les systèmes à retard pur

Maintenant $\omega = 1$ donne $p(j\omega, d) = j + e^{-dj}$ et $p(j, d) = 0$ soit $d = \frac{\pi}{2} + 2\pi l$, et la courbe $q(\omega^2)$ traverse l'axe des ω^2 de haut en bas quand ω^2 varie en croissant, donc $p(s, d)$ traverse l'axe imaginaire de droite vers la gauche. La stabilité est acquise pour des valeurs de $d > \frac{\pi}{2} + 2\pi l$ (l entier > 0).

Enfin pour $\omega = \sqrt{2}$, on a $p(j\omega, d) = -1 + e^{-dj\sqrt{2}} = 0$, ceci donne $d = l\pi\sqrt{2}$, la courbe $q(\omega^2)$ traversant l'axe des ω^2 de bas en haut, on a donc stabilité pour $d < l\pi\sqrt{2}$.

En résumé le système bouclé est stable pour des valeurs de : $\frac{\pi}{2} < d < \pi\sqrt{2}$ ou $\frac{5\pi}{2} < d < 2\pi\sqrt{2}$ ou $\frac{9\pi}{2} < d < 4\pi\sqrt{2} \dots$.

Vérification avec Scilab, on prendra $d1 < \frac{\pi}{2}$, $d2 = \pi\sqrt{2}$, $d3 = \frac{5\pi}{2}$. Voici le programme.

```
-->s=%s;sl=syslin("c",1,s^3+s*s+2*s+1)
-->d1=.8*pi/2;d2=%pi*sqrt(2);d3=5*pi/2;])
-->g1=iodelay(sl,d1);g2=iodelay(sl,d2);g3=iodelay(sl,d3);
-->nnyquist([g1;g2;g3],.01,5,["retard 1,256 s";"retard 4,443 s";...
"retard 7,85 s"]])
//ici je trace les lieux et les compare par rapport au point [-1,0].
```

Nous voyons que le premier lieu est à gauche du point $[-1, 0]$ et donc le système correspondant est instable, le second et troisième passent par ce point : on est à la limite de la stabilité. Vous vérifierez que pour $\frac{\pi}{2} < d < \pi\sqrt{2}$ on a bien des systèmes stables FIG 10.8.

Troisième exemple

Soit une boucle ouverte de transmittance, ici $G(s) = \frac{s}{s^2+s+1}e^{-ds}$, ceci correspond à un quasi-polynôme $p(s, d) = se^{-ds} + s^2 + s + 1$ on applique encore les règles.

1. $p(s, 0) = (s + 1)^2$ donne un système sans retard stable.
2. $q(\omega^2) = (1 - \omega^2)^2$.
3. $\omega = 1$ est une racine double pour $q(\omega^2)$, donc $p(s, d) = se^{-ds} + s^2 + s + 1$ touche l'axe imaginaire, sans le traverser, pour $d = (2l + 1)\pi$, avec l entier positif.
4. Conclusion le système est asymptotiquement stable pour toutes les valeurs de $d \neq (2l + 1)\pi$.

Comme précédemment traçons deux lieux de Nyquist. Voici le programme.

```
-->sl=syslin("c",s,s*s+s+1); d1=%pi; d2=%pi/2;
-->g1=iodelay(sl,d1); g2=iodelay(sl,d2);
-->nnyquist([g1;g2],.1,1,["retard pi";"retard pi/2"]])
```

Dans ce dernier exemple on voit que le premier lieu passe juste par le point critique, quant au second lieu il ne passe pas par ce même point et reste toujours dans le cercle centré à l'origine et de rayon 1 (FIG 10.8).

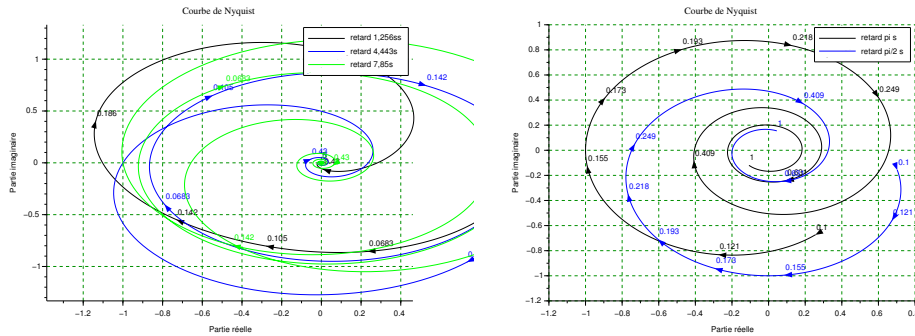


FIGURE 10.8 – Lieux de Nyquist $\frac{N(s)}{D(s)}e^{-ds}$, deuxième, troisième exemples.

10.2.5 Les marges de stabilité

Dans la boîte à outils proposée, trois programmes permettent de calculer les marges de stabilité des systèmes à retard pur. Il s'agit des macros `p_marginrd()`, `g_marginrd()` et `m_marginrd()`. Nous allons proposer un exercice, semblable à celui que l'on a vu pour les systèmes classiques. Voici la suite du troisième exemple proposé : (FIG 10.9)

```
-->g1
g1 =
          1s
exp(-3.1415927*s)*-----
                    2
                1 + 1s + 1s
-->[pm,fpm]=p_marginrd(g1)
fpm =
0.1591549
pm =
0.0000032
//c'est normal le lieu de Nyquist passe par le point critique.
-->[gm,fgm]=g_marginrd(g1)
fgm =
0.1591549
gm =
0.//même remarque.
-->[mm,fmm]=m_marginrd(g1)
fmm =
0.1591549
mm =
0.0000001 //même remarque.
```

Maintenant examinons le deuxième système.

```
-->g2 =
g2 =
          1s
exp(-1.5707963*s)*-----
                    2
                1 + 1s + 1s
-->[pm,fpm]=p_marginrd(g2)
fpm =
0.1591549
pm =
89.999998//le lieu passe par le point [0;-1].
//tracer le cercle centré en [0;0] de rayon 1.
-->[gm,fgm]=g_marginrd(g2)
fgm =
0.2447582
gm =
2.5233032
-->[mm,fmm]=m_marginrd(g2)
fmm =
0.2401550
mm =
0.244207
-->nnyquist(g2,.01,5,["retard pi/2 s"])
//nous traçons le lieu de Nyquist et basculons en mode datatips
```

10.2.6 Lieu des racines adapté aux systèmes à retard pur

Je ne rappelle pas l'ensemble des propriétés du lieu des racines d'un système classique quand le gain de la chaîne d'action varie, pour cela un bon cours d'automatique classique (voir réf : [4]) est souhaitable. Je vais malgré tout donner les principales propriétés du lieu des racines pour les systèmes à retard pur en entrée-sortie.

Le lieu des racines (de l'équation caractéristique) revient à faire l'étude de l'expression :

$$E_c = 1 + ke^{-ds} \frac{N(s)}{D(s)} = 0$$

il s'agit en fait de trouver le lieu des racines de cette équation ou encore trouver les racines d'une équation transcendante quand k varie à priori de $-\infty$ à $+\infty$, cette équation s'écrit :

$$D(s) + ke^{-ds} N(s) = 0$$

Si on pose $s = \sigma + j\omega$ on a deux conditions (condition des modules et condition des angles) :

$\left| \frac{N(\sigma+j\omega)}{D(\sigma+j\omega)} \right| e^{-d\sigma} = \frac{1}{|k|}$ et $\arg\left(\frac{N(\sigma+j\omega)}{D(\sigma+j\omega)}\right) - d\omega = (1 + 2l)\pi$ avec l entier positif nul ou négatif, ou $\arg\left(\frac{N(\sigma+j\omega)}{D(\sigma+j\omega)}\right) = d\omega + (1 + 2l)\pi$, nous constatons que pour un retard d non

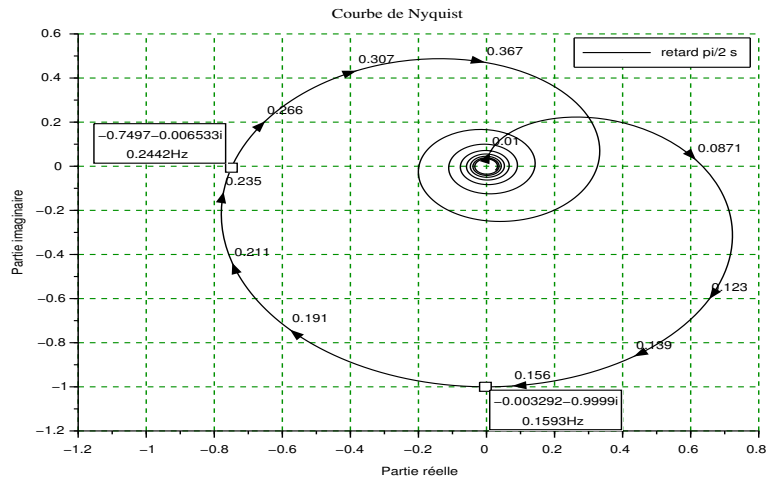


FIGURE 10.9 – Lieu Nyquist du troisième exemple : les marges de stabilité.

nul, la condition des angles contient un terme linéaire vis à vis de la pulsation : pour une valeur de k on un nombre infini de solutions de l'équation caractéristique. Maintenant nous allons comparer les règles de construction du lieu pour un système sans et avec retard.

Lieu des pôles des systèmes sans retard.

- L'équation caractéristique est de degré n donc le lieu a n branches.
- L'axe réel est un axe de symétrie.
- Chaque branche a pour point de départ l'un des n pôles p_i de la boucle ouverte.
- S'il y a m zéros z_j dans la boucle ouverte il y a m points d'arrivée à distance fini.
- Il y a $n - m$ points d'arrivée à l'infini (branches asymptotiques).
- En vérifiant la condition des angles, on déduit qu'il y a $n - m$ asymptotes qui font des angles $\theta_k = \frac{(1+2l)\pi}{n-m}$, l entier ≥ 0 , par rapport à l'axe réel.
- Le point d'intersection des asymptotes est sur l'axe réel et a pour abscisse $\delta = \frac{\sum p_i - \sum z_j}{n-m}$.
- Tout point sur l'axe réel appartient au lieu s'il est situé à gauche d'un nombre impair de pôles et zéros réels.
- Les intersections avec l'axe imaginaire sont obtenus en annulant le reste de la division du polynôme $D(s) + kN(s)$ par un terme $s^2 + \omega_1^2$. (Avec Scilab on peut utiliser le programme **kpure()**).
- Les points de séparation s_s sur l'axe réel vérifient $\sum_1^m \frac{1}{s_s - z_j} = \sum_1^n \frac{1}{s_s - p_i}$.
- L'angle de la tangente en un point de départ (ou d'arrivée) est donné par la relation : $\theta_s = \sum_1^m \arg(s_d - z_j) - \sum_1^n \arg(s_d - p_i) - \frac{3\pi}{2}$.

Lieu des pôles des systèmes avec retard.

- L'équation caractéristique est une équation transcendante donc le lieu a un nombre infini de branches.
- L'axe réel est un axe de symétrie.
- Les n premiers points de départ sont les pôles de la boucle ouverte, les autres sont situés sur les asymptotes horizontales dans le demi plan gauche : $s = \sigma + j\omega$ avec $\sigma < 0$.
- Les m premiers points d'arrivée sont les zéros de la boucle ouverte, les autres sont situés sur les asymptotes horizontales dans le demi plan droit : $\sigma > 0$.
- Les asymptotes en nombre infini sont parallèles à l'axe des réels. Leurs intersections avec l'axe imaginaire sont données par $\omega = \frac{\pi v}{d}$; voir tableau.
- Les asymptotes sont parallèles.
- Tout point sur l'axe réel appartient au lieu s'il est situé à gauche d'un nombre impair de pôles et zéros réels.
- Les intersections avec l'axe imaginaire posent problème : pour les points le plus près de l'axe réel on peut utiliser l'algorithme de **g_marginrd()** (approximant de Padé puis méthode de Newton-Raphson, je donne un exemple).
- Les points de séparation s_s sur l'axe réel vérifient l'équation : $\frac{d((N(s)/D(s))e^{-sd})}{ds} = 0$, racine double réelle de cette équation.
- L'angle de la tangente en un point de départ (ou d'arrivée) est donné par la relation des angles : $\arg(\frac{N(s)}{D(s)}) = (1+2l)\pi + \omega d$, l entier ≥ 0 .

$n - m$	asymptotes de départ	asymptotes d'arrivée
Impair	$v =$ entiers pairs	$v =$ entiers pairs
Pair	$v =$ entiers impairs	$v =$ entiers impairs

Un exemple (FIG 10.10) :

Soit un système bouclé de transmittance de chaîne d'action $G(s) = ke^{-ds} \frac{1}{s(s+1)}$; il a pour équation caractéristique : $s^2 + s + ke^{-ds} = 0$, les règles de 3 à 10 et le tableau précédent donnent :

1. **Points de départ** : ($k = 0$) on a $s = 0$, $s = -1$ et $\sigma = -\infty$, alors $\omega = \pm\pi/d$, $\omega = \pm 3\pi/d$, $\omega = \pm 5\pi/d, \dots$ etc.
2. **Points d'arrivée** : ($k = +\infty$) pas de points à distance finie, mais $\sigma = +\infty$, donne $\omega = \pm\pi/d$, $\omega = \pm 3\pi/d$, $\omega = \pm 5\pi/d, \dots$ etc.
3. **Points sur axe réel** : ces points sont sur le segment $[-1, 0]$.

4. **Points de séparation sur l'axe réel :** en reprenant l'équation $\frac{d(k \frac{1}{s(s+1)} e^{-sd})}{ds} = 0$ on obtient $s_{\text{separ}} = \frac{-(d+2) \pm \sqrt{d^2+4}}{2d}$, si $d = 1$ on trouve $s_{\text{separ}} = -0.381966$ ce qui donne pour k la valeur $k_{\text{separ}} = 0.16112$. On pourrait programmer une fonction analogue au programme `krac2.sci` pour trouver les valeurs de k_{separ} et les valeurs correspondantes de s_{separ} .
5. **Points sur l'axe imaginaire :** en ces points il y a passage de la stabilité à l'instabilité pour le système bouclé ou vice versa. On doit donc résoudre l'équation $s^2 + s + ke^{-ds} = 0$ pour $s = \pm j\omega$ soit $\tan(d\omega_l) = \frac{1}{\omega_l}$, que l'on résout graphiquement ; quant à la valeur de k elle vérifie $k = k_l = \frac{\omega_l}{\sin(d\omega_l)}$. On peut aussi utiliser la fonction Scilab que je propose, `g_marginrd()` pour ici $k = 1$ et à partir de cette marge de gain retrouver l'homothétie à faire sur le lieu de Nyquist pour le faire passer par le point $[-1, 0]$. On trouve $k_l = 1.1349147$ et $\omega_l = 0.8603336$ rd/s. On pourrait programmer une fonction analogue au programme `kpure.sci` pour trouver **la première valeur** de k_l et la valeur correspondante de ω_l .

Mais voici une autre méthode donnant **la première valeur** de k_l et la valeur de ω_l correspondante (programme équivalent à `kpure` pour les systèmes à retard).

```
-->s=%s;g=syslin("c",1,s*s+s);
-->gd=iodelay(g,1);
-->[mg,fmg]=g_marginrd(gd)
fmg =
0.1369263
mg =
1.099264//Marge de gain en db.
-->kl=10^(mg/20)//gain limite.
kl =
1.1349147//Première valeur de k limite.
//Vérifions
-->gdkl=kl*gd
gdkl =
1.1349147
exp(-1*s)*-----
2
1s + 1s
-->nnyquist([gdkl;gd],.1,3)//Le lieu passe par [-1,0] vous vérifierez.
-->puls=2*pi*fmg//Valeur de la racine imaginaire positive.
puls =
0.8603336 //Pulsation limite (racine).
```

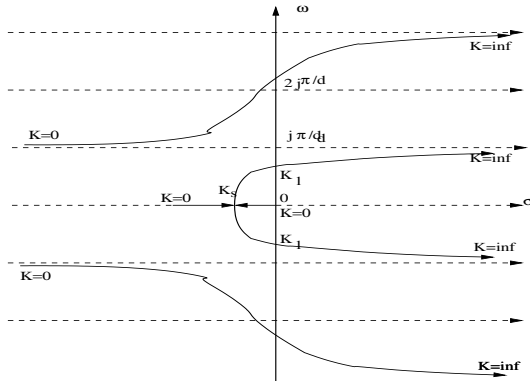


FIGURE 10.10 – Lieu des racines.

10.2.7 Les systèmes à retard et les pseudo-lieux

On peut, comme pour les systèmes sans retard (voir paragraphe 5.2.6), faire l'étude des pseudo-lieux, et en particulier rechercher le gain k_o de la chaîne d'action pour que le pseudo-lieu paramétré avec cette valeur de k passe par le point $[-1, 0]$ dans le plan complexe ou passe par le point $[-180^\circ, 0]$, dans le plan de Black. Voici l'exemple du système précédent (voir lieu des racines), traité dans le plan de Black on a : (FIG 10.11)

```
-->s=%s;g=syslin("c",1,s*s+s);gd=iodelay(g,1);
-->bblack(gd,.01,1,"120","gd_120")//Pseudo lieu que je ne donne pas.
-->[bout,posx,posy]=xclick()//Distance sur l'axe vertical du pseudo-lieu
de gd au point [-180°,0db].
posy =
8.8837595 //Comme posy est positif, l'intersection avec l'axe vertical
est au dessus de l'axe horizontal.
posx =
- 179.92844
bout =
3.
-->k=10^(-posy/20)//Avec la remarque précédente l'exposant doit être négatif
(k<1).
k =
0.3593596
-->gk=k*gd
gk =
0.3593596
exp(-1*s)*-----
2
1s + 1s
//Vérification graphique.
```

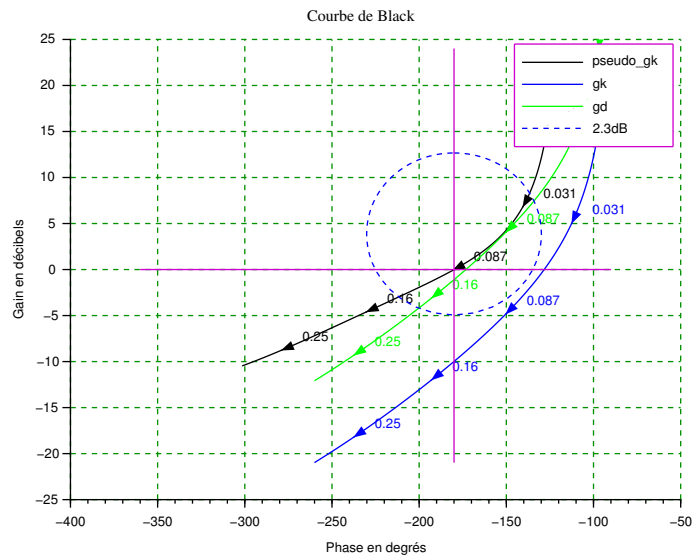



FIGURE 10.11 – Règlage du gain d'un système à retard par les pseudo-lieux

```
-->clf();
-->bblack([gk;gk;gd],.01,0.3,["120";"90";"90"],["pseudo_gk";"gk";"gd"])
Faisons une petite vérification, en calculant les marges de gain et module :
-->[mg,fmg]=g_marginrd(gk)
fmg =
0.1369263
mg =
9.988678//en db.
-->[mm,fmm]=m_marginrd(gk)
fmm =
0.0926199
mm =
0.5737916
```

10.3 Les systèmes échantillonnés, passage du continu au discret

Je fais ici un petit rappel sur les systèmes échantillonnés pour étudier la discrétisation et introduire les différents filtres et bloqueurs. Si vous n'êtes pas habitué à traiter les

problèmes relatifs aux systèmes échantillonnés procurez vous la « bible » française écrite par le professeur Y. Sevely un des pères de l'automatique en France, « Systèmes et asservissements échantillonnés » : on le trouve encore d'occasion.

10.3.1 Comment discrétiser un système continu : les méthodes

Toutes les méthodes proposées ont pour but de transformer un système continu $G(s)$ en un système échantillonné $G_e(z)$ avec T_s la période d'échantillonnage.

La méthode qui vient à l'esprit est d'utiliser les tables de transformées en « z », en ayant au préalable décomposé la transmittance du système continu $G(s)$ en éléments simples. L'inconvénient de cette méthode réside dans le fait que l'échantillonneur (ou échantillonneur-modulateur) doit être parfaitement défini.

Echantillonnage d'un signal continu

Soit $x(t)$ un signal continu causal et soit un échantillonnage à période constante T_s , le résultat $x^*(t)$ correspond à une suite $[x_k]$, cette suite est le résultat de l'opération : $x^*(t) = x(t)\delta_{T_s}(t) = \sum_{k=-\infty}^{+\infty} x_k\delta(t - kT_s)$ où $\delta_{T_s}(t)$ est appelé peigne de Dirac (c'est une distribution). En prenant la transformée de Laplace de cette équation, comme $x(t)$ est nul pour $t < 0$, on obtient la transformée en z du signal ($z = e^{T_s s}$), soit $X(z) = \sum_{k=0}^{+\infty} x_k z^{-k}$. On montre aussi que la transformée de Laplace du signal $x^*(t)$ est donné par l'expression : $X^*(s) = \frac{1}{2\pi j} \int_{c-j\infty}^{c+j\infty} \frac{X(\mu)}{1-e^{T_s \mu} e^{-T_s s}} d\mu$, et par la méthode des résidus on retrouve $X(z)$.

Restitution du signal incident : filtres, bloqueurs, échantillonneur modulateur

Un échantillonneur modulateur (voir cours sur les systèmes échantillonnés) est un dispositif électronique qui en fonction d'une période d'échantillonnage T_s , à un instant $t_n = nT_s$ envoie une commande $v_n = f(u_n, t)$, u_n étant la commande de l'échantillonneur modulateur, et ceci durant le temps T_s . Un exemple est l'échantillonneur modulateur (modulateur de largeur) qui envoie à un instant $t_n = nT_s$ une impulsion rectangulaire ou autre dont la largeur l est proportionnelle à la commande u_n et dont le signe peut être le signe de u_n (dans ce cas on aura affaire à un système échantillonné non linéaire).

Parmi les échantillonneurs modulateurs il y a le bloqueur d'ordre zéro (B_o) qui est le plus utilisé et dont le modèle de transfert est : $B_o(s) = \frac{1-e^{-sT_s}}{s}$, ce bloqueur permet de reproduire exactement un signal constant échantillonné. Un autre bloqueur est le bloqueur d'ordre 1 qui réalise une extrapolation linéaire de l'évolution du signal avant échantillonnage à partir des deux dernières prises d'information : il permet de reproduire exactement un signal linéaire échantillonné. Son modèle de transfert est : $B_1(s) = \frac{(1+T_s s)}{T_s} \left(\frac{1-e^{-sT_s}}{s} \right)^2$. Si τ est une variable temps $0 \leq \tau < T_s$ alors la sortie de ce modulateur vaut : $v(nT_s + \tau) = u(nT_s) + \frac{\tau}{T_s}(u(nT_s) - u((n-1)T_s))$. Enfin un dernier modulateur est l'extrapolateur linéaire à retard pur dont le modèle est $B_e(s) = \frac{1}{T_s} \left(\frac{1-e^{-sT_s}}{s} \right)^2$ et dont la sortie vaut $v(nT_s + \tau) = u((n-1)T_s) + \frac{\tau}{T_s}(u(nT_s) - u((n-1)T_s))$.

10.3.2 Les systèmes continus sans retard, passage au discret avec bloqueur d'ordre zéro

L'intégration des équations d'état, vu au paragraphe 9.2.2, permet le passage simple des équations d'état d'un modèle continu à un modèle discret avec **un bloqueur d'ordre zéro**, et Scilab le réalise avec les fonctions `dscr.sci` et `ss2tf.sci`, voici un exemple (FIG 10.12) :

Rappel des équations précédentes :

$$x((n+1)T_s) = e^{AT_s}x(nT_s) + \left(\int_0^{T_s} e^{A\mu} d\mu \right) Bu(nT_s)$$

ou

$$\begin{aligned} x(n+1) &= \Phi(T_s)x(n) + \Psi(T_s)u(n) \\ y(n) &= Cx(n) + Du(n) \end{aligned}$$

L'exemple :

```
-->s=%s;z=%z;g=syslin("c",10,s*s+3*s+10)//Modèle continu.
g =
      10
-----
              2
10 + 3s + s
-->Tsam=.1;//Période d'échantillonnage.
-->gse=dscr(g,Tsam);//Modèle d'état discrétisé Tsam = 0.1 sec.
-->ge=ss2tf(gse)//Modèle de transfert échantillonné.
ge =
0.0406929 + 0.0449846z
-----
              2
0.7408182 - 1.6551408z + z
//Réponses indicielles.
-->u=0:.05:4;//Le pas vaut Tsam/2.
-->repcstep=csim("step",u,g);
-->ue=0:Tsam:4;//Discrétisation à la période d'échantillonnage.
-->repestep=flts(ones(ue),ge);
-->plot(u,repcstep,"r",ue,repestep,"b")
-->e=gce();
-->e.children(1).polyline_style=2;
-> xtitle("Réponses à un échelon unitaire");
-->legend(["continu";"échantillonné bloqué"]);
-->xgrid(13)
```

Nous allons maintenant faire une vérification avec une table de transformées en z . Pour un système continu du second ordre précédé d'un bloqueur d'ordre zéro, les modèles continu et échantillonné sont :

10 Utilisation du logiciel pour simuler les systèmes à retard pur

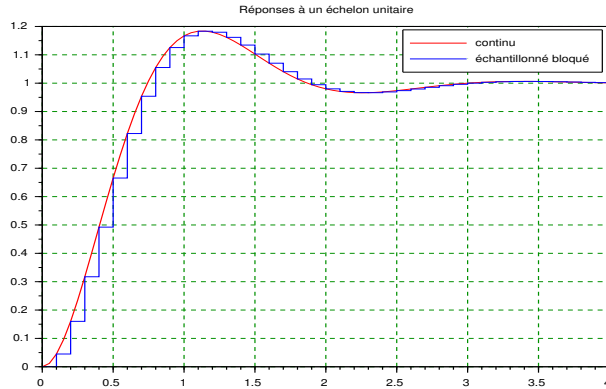


FIGURE 10.12 – Discrétisation bloqueur d'ordre zéro.

Transmittance continue (s)	Transmittance échantillonnée (z)
$\frac{kB_0(s)\omega_n^2}{s^2+2\zeta\omega_n s+\omega_n^2}$	$k \frac{b_1 z + b_0}{D(z)}$
$\omega_p = \omega_n \sqrt{1 - \zeta^2}$	$D(z) = z^2 - 2ze^{-\zeta\omega_n T_s} \cos(\omega_p T_s) + e^{-2\zeta\omega_n T_s}$
$\zeta < 1$	$b_0 = e^{-2\zeta\omega_n T_s} + e^{-\zeta\omega_n T_s} \left(\frac{\zeta \sin(\omega_p T_s)}{\sqrt{1-\zeta^2}} - \cos(\omega_p T_s) \right)$
	$b_1 = 1 - e^{-\zeta\omega_n T_s} \left(\cos(\omega_p T_s) + \frac{\zeta \sin(\omega_p T_s)}{\sqrt{1-\zeta^2}} \right)$

Voici le programme Scilab :

```
-->wn=sqrt(10);zet=1.5/wn;wp=wn*sqrt(1-zet^2);
-->Dz=z*z-2*exp(-zet*wn*Tsam)*cos(wp*Tsam)*z+exp(-2*zet*wn*Tsam)
Dz =
                2
0.7408182 - 1.6551408z + z
-->b0=exp(-2*zet*wn*Tsam)+exp(-zet*wn*Tsam)*(zet*sin(wp*Tsam)..
/sqrt(1-zet^2)-cos(wp*Tsam))
b0 =
0.0449846
-->b1=1-exp(-zet*wn*Tsam)*(zet*sin(wp*Tsam)/sqrt(1-zet^2)+cos(wp*Tsam))
b1 =
0.0449846
-->Nz=b0+b1*z
Nz =
0.0406929 - 0.0449846z
-->gge=syslin(Tsam,Nz,Dz)
```

```

gge =
0.0406929 - 0.0449846z
-----
                                2
0.7408182 - 1.6551408z + z
-->ge
ge =
0.0406929 + 0.0449846z
-----
                                2
0.7408182 - 1.6551408z + z //Même résultat.

```

Remarque : Si $G(s)$ est précédé d'un bloqueur d'ordre zéro, alors la transmittance en z recherchée (notée $G_{bo}(z)$) vaut :

$$G_{bo}(z) = \frac{z-1}{z} Z\left(\frac{G(s)}{s}\right)$$

ne pas oublier le terme en $\frac{1}{s}$. On pourra utiliser cette relation en mettant $\frac{G(s)}{s}$ en éléments simples, puis on prendra la transformée en z de chacun des éléments, ($\frac{G(s)}{s}$ est la réponse à un échelon unitaire du système continu considéré).

10.3.3 Expression de la transformée en z , par la méthode des résidus

Soit un système continu de fonction de transfert $H(s)$, ou un signal $h(t)$ de transformée de Laplace $H(s)$, on montre que si $H(s)$ a un nombre fini de pôles (p pôles) alors le dénominateur est un polynôme de degré p et $H(z)$ sa transformée en z vaut

$$H(z) = \sum_{p_i} \text{résidus}\left(\frac{H(\nu)}{1 - e^{\nu T_s} z^{-1}}\right)_{\nu=p_i}$$

Quand $H(\nu) = \frac{N(\nu)}{D(\nu)}$ n'a que des pôles simples (p_i un pôle) alors :

$$H(z) = \sum_{p_i} \frac{N(p_i)}{D'(p_i)} \frac{z}{(z - e^{p_i T_s})}$$

Dans le cas de pôles multiples (ordre k) alors le résidu r_i associé au pôle p_i vaut :

$$r_i = \frac{1}{(k-1)!} \left[\frac{d^{k-1}}{d\nu^{k-1}} \left((\nu - p_i)^k H(\nu) \frac{1}{1 - e^{\nu T_s} z^{-1}} \right) \right]_{\nu=p_i}$$

10.3.4 Les systèmes échantillonnés, passage du discret au continu et vice versa avec approximations

Vous trouverez dans les **demos** de **Autoelem Toolbox** un exercice comparant en fréquentiel et en temporel les différentes approximations.

Equivalence d'un signal échantillonné et bloqué avec un signal continu retardé de $\frac{T_s}{2}$

Quand on cherche à faire la synthèse d'un système échantillonné (détermination d'un correcteur numérique) qui est le résultat d'un système continu échantillonné et bloqué, si la période d'échantillonnage est faible vis à vis des constantes de temps du procédé, on peut considérer qu'un bloqueur d'ordre zéro est équivalent à un retard pur de $\frac{T_s}{2}$ qui a pour modèle continu $e^{-\frac{T_s}{2}}$, ainsi faire la synthèse d'un système échantillonné revient à faire la synthèse d'un système continu retardé de $\frac{T_s}{2}$.

Approximation linéaire de la variable z

Quand la période d'échantillonnage est petite, on peut faire l'approximation de $z = e^{T_s s}$ par $z \simeq 1 + sT_s$ ou $s \simeq \frac{(z-1)}{T_s}$ (méthode d'Euler progressive ou Euler's forward method). Cette approximation peut, dans certains cas, à partir d'un système continu stable, conduire à un système échantillonné instable.

Une autre approximation consiste à remplacer la variable s par $s \simeq \frac{(z-1)}{zT_s}$ ou $z \simeq \frac{1}{1-sT_s}$ (méthode d'Euler rétrograde ou Euler's backward method), qui donne un système stable si l'original continu était stable. Il est très facile de programmer ces deux approximations avec l'instruction Scilab `horner.sci`.

Approximation au second ordre : transformation homographique

C'est une approximation au second ordre, la transformation associée est homographique : $z = e^{sT_s} = \frac{e^{s\frac{T_s}{2}}}{e^{-s\frac{T_s}{2}}} \simeq \frac{1+sT_s/2}{1-sT_s/2}$ (un approximant du premier ordre) ou $s \simeq \frac{2}{T_s} \frac{z-1}{z+1}$ (Transformation bilinéaire ou homographique ou de Tustin) : Scilab propose le programme `cls2dls.sci` pour réaliser cette opération.

Transformée en w

Quand on étudie la stabilité des systèmes échantillonnés, on peut utiliser le critère de Routh-Hurwitz en effectuant un changement de variable $z \rightarrow \frac{1+w}{1-w}$ ou $w \rightarrow \frac{z-1}{z+1}$, le polynôme caractéristique $P(\lambda)$ du système échantillonné, par cette transformation, deviendra $P1(w) = (1-w)^n P(\frac{1+w}{1-w})$ et une C.N.S. de stabilité du système échantillonné est que $P1(w)$ vérifie le critère de Routh-Hurwitz. De même lors de la correction d'un système bouclé, l'utilisation de cette transformée permet d'utiliser les méthodes de synthèse du continu et de trouver ainsi le correcteur échantillonné recherché.

10.3.5 Les systèmes continus avec retard, passage au discret avec bloqueur d'ordre zéro

Quand le modèle du système possède un retard pur, alors la transmittance continue est de la forme $Gdelay(s) = G(s)e^{-ds}$ où d est le retard et $G(s)$ la partie transmittance classique de $Gdelay(s)$.

Remarquons tout de suite que si on voulait définir un modèle d'état pour le système retardé on aurait une dimension infini pour ce modèle : par échantillonnage on va approximer le modèle continu par un système échantillonné de dimension fini.

Quand on échantillonne (T_s étant la période d'échantillonnage) deux cas sont à envisager :

Le retard est un multiple entier de la période $d = kT_s$, et dans ce cas, si $G(z)$ est la transmittance associée à $G(s)$, on a $G_{delay}(z) = z^{-k}G(z)$ (théorème du retard de la transformée en z). Si on raisonne en variables d'état le modèle continu est de dimension infini, mais le modèle d'état échantillonné sera de dimension fini.

Dans le cas contraire on aura $d = kT_s + \alpha T_s$, avec k un entier positif et $0 \leq \alpha < 1$. On introduira avec ce mécanisme des zéros supplémentaires dans la transmittance, ces zéros pouvant être de module supérieur à 1 (zéros dit « instables » et ceci si $\alpha > 0.5$). Cette opération revient à calculer la transmittance en z modifiée $G(z, m)$ (voir un cours sur les systèmes échantillonnés) ; on peut facilement montrer qu'avec un bloqueur d'ordre zéro, si $G_1(z, m)$ est la transmittance échantillonnée modifiée de $\frac{G(s)}{s}$, alors la transmittance en z recherchée vaut : avec $m = 1 - \alpha$.

$$G_{bo}(z, 1 - \alpha) = \left(\frac{z-1}{z}\right)G_1(z, 1 - \alpha)$$

Voici d'après les tables, la transformée en z d'un second ordre à pôles complexes conjugués, avec bloqueur d'ordre zéro, retardé de $(k + \alpha)T_s$, on remarquera que le dénominateur de la transmittance ne change pas pour un système avec ou sans retard.

$G_{delay}(s)$	$G_{delay}(z)$
$\frac{B_0(s)\omega_n^2 e^{-s(k+\alpha)T_s}}{s^2 + 2\zeta\omega_n s + \omega_n^2}$	$\frac{b_1 z + b_2 + \frac{b_3}{z}}{D(z)} z^{-k}$
$\zeta < 1$	$D(z) = z^2 - 2ze^{-\zeta\omega_n T_s} \cos(\omega_p T_s) + e^{-2\zeta\omega_n T_s}$
$\omega_p = \omega_n \sqrt{1 - \zeta^2}$	$b_1 = 1 - e^{-\zeta\omega_n (1-\alpha)T_s} (\cos(\omega_p (1-\alpha)T_s) + \frac{\zeta \sin(\omega_p (1-\alpha)T_s)}{\sqrt{1-\zeta^2}})$
$0 \leq \alpha < 1$	$b_3 = e^{-2\zeta\omega_n T_s} - e^{-\zeta\omega_n (2-\alpha)T_s} (\cos(\omega_p \alpha T_s) - \frac{\zeta \sin(\omega_p \alpha T_s)}{\sqrt{1-\zeta^2}})$

avec :

$$b_2 = e^{-\zeta\omega_n (2-\alpha)T_s} (\cos(\alpha\omega_p T_s) - \frac{\zeta \sin(\alpha\omega_p T_s)}{\sqrt{1-\zeta^2}}) + \dots$$

$$\dots e^{-\zeta\omega_n (1-\alpha)T_s} (\cos(\omega_p (1-\alpha)T_s) + \frac{\zeta \sin(\omega_p (1-\alpha)T_s)}{\sqrt{1-\zeta^2}}) - 2e^{-\zeta\omega_n T_s} \cos(\omega_p T_s)$$

10.3.6 Expression de la transformée en z modifiée par la méthode des résidus

Soit un système continu de fonction de transfert $H(s)$, ou un signal $h(t)$ de transformée de Laplace $H(s)$, on montre que si $H(s)$ a un nombre fini de pôles (p pôles) alors le dénominateur est un polynôme de degré p et $H(z, m)$ sa transformée en z modifiée vaut

$$H(z, m) = z^{-1} \sum_{p_i} \text{résidus} \left(\frac{H(\nu) e^{mT_s \nu}}{1 - e^{\nu T_s} z^{-1}} \right)_{\nu=p_i}$$

Ce qui différencie la transformée en z , $H(z)$ de la transformée en z modifiée $H(z, m)$ c'est :

1. La présence d'un facteur en z^{-1} .
2. La présence d'un terme $e^{mT_s\nu}$ au numérateur : le terme contenant m n'apparaît qu'au numérateur.

Quand $H(\nu) = \frac{N(\nu)}{D(\nu)}$ n'a que des pôles simples alors

$$H(z, m) = z^{-1} \sum_{p_i} \frac{N(p_i)}{D'(p_i)} \frac{e^{mT_s p_i}}{(1 - e^{p_i T_s} z^{-1})}$$

Dans le cas de pôles multiples (ordre k) alors le résidu r_i associé au pôle p_i vaut :

$$r_i = \frac{1}{(k-1)!} \left[\frac{d^{k-1}}{d\nu^{k-1}} ((\nu - p_i)^k H(\nu) \frac{e^{mT_s \nu}}{1 - e^{\nu T_s} z^{-1}}) \right]_{\nu=p_i}$$

10.3.7 Algorithme de discrétisation d'un système à retard pur : programme dscr_sisord.sci

Comme nous l'avons dit au paragraphe 10.3.5 si le retard est multiple de la période d'échantillonnage ($d = kT_s$), alors la transmittance en z du système retardé vaut $G_{delay}(z) = z^{-k} G(z)$ si $G(z)$ est la transmittance du système sans retard.

Dans le cas général, on aura $d = kT_s + \alpha T_s$, avec k le plus grand entier positif inférieur à d et $0 \leq \alpha < 1$: on recherchera la transmittance échantillonnée pour un retard de αT_s notée $G_\alpha(z)$ et l'on aura $G_{delay}(z) = z^{-k} G_\alpha(z)$.

Principe de calcul de $G_\alpha(z)$

Retarder le signal d'entrée du retard αT_s revient à appliquer entre les instants nT_s et $(n+1)T_s$ un signal constitué de deux paliers successifs, de nT_s à $(n+\alpha)T_s$ le signal $u_1(t) = u((n-1)T_s)$, puis de $(n+\alpha)T_s$ à $(n+1)T_s$ le signal $u_2(t) = u(nT_s)$.

On va donc intégrer successivement les équations d'état du système continu avec comme signal d'entrée les deux paliers $u_1(t)$ puis $u_2(t)$.

Sur l'intervalle de temps de nT_s à $(n+1)T_s$ le signal $u_1(t)$ vaut $u((n-1)T_s)$ de nT_s à $(n+\alpha)T_s$, puis 0(t) de $(n+\alpha)T_s$ à $(n+1)T_s$. Quant au signal $u_2(t)$ il vaut 0(t) de nT_s à $(n+\alpha)T_s$ puis $u(nT_s)$ de $(n+\alpha)T_s$ à $(n+1)T_s$.

D'après ce raisonnement, on aura un système continu échantillonné et bloqué ayant une sortie $y(t)$ et deux entrées $[u_1(t), u_2(t)]$ que l'on intégrera sur l'intervalle de temps T_s ; remarquons que $Z(u_1(t)) = k \frac{Z(u_2(t))}{z}$.

Calcul

Notons respectivement $x(nT_s) = x_n$, $x((n+1)T_s) = x_{n+1}$, $x((n+\alpha)T_s) = x_{n+\alpha}$ on a :

$$x_{n+1} = e^{AT_s} x_n + \int_{nT_s}^{(n+1)T_s} e^{A(T_s-\mu)} B u(\mu) d\mu$$

Ici l'entrée $u(t)$ est constituée de deux paliers successifs $[u_1(t), u_2(t)]$, le premier terme de la somme étant la contribution à l'instant $(n+1)T_s$ du vecteur d'état initial (instant nT_s), le second terme sera constitué d'une somme qui dépendra linéairement de u_{n-1} et de u_n . Notons $\Phi(T_s) = e^{AT_s}$, $\Phi(\theta T_s) = e^{A(\theta T_s)}$, $\Psi(T_s) = (\int_0^{T_s} e^{A\mu} d\mu)B$, $\Psi(\theta T_s) = (\int_0^{\theta T_s} e^{A\mu} d\mu)B$ et comme $\Phi(T_s) = e^{AT_s} = e^{A(1-\alpha)T_s} e^{A\alpha T_s}$, (voir le paragraphe 9.2.2), avec ces notations et en intégrant sur ces deux intervalles de temps, le premier αT_s , le second $(1-\alpha)T_s$ nous avons :

$$x_{n+1} = \Phi(T_s)x_n + \Phi((1-\alpha)T_s)\Psi(\alpha T_s)u_{n-1} + \Psi((1-\alpha)T_s)u_n$$

Cette équation peut être représentée matriciellement (comme pour les systèmes sans retard) par l'équation :

$$\begin{bmatrix} x_{n+1} \\ u_n \end{bmatrix} = \begin{bmatrix} \Phi(T_s) & \Phi((1-\alpha)T_s)\Psi(\alpha T_s) \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_n \\ u_{n-1} \end{bmatrix} + \begin{bmatrix} \Psi((1-\alpha)T_s) \\ I \end{bmatrix} u_n$$

Maintenant prenons la transformée en z de la première équation, on a donc :

$$(zI - \Phi(T_s))x(z) = \left[\frac{1}{z} \Phi((1-\alpha)T_s)\Psi(\alpha T_s) + \Psi((1-\alpha)T_s) \right] u(z)$$

Ainsi pour trouver la transmittance on réalisera les opérations matricielles :

$$G_\alpha(z) = \frac{G_1(z)}{z} + G_2$$

avec $G_1 = C(zI - \Phi)^{-1}\Phi((1-\alpha)T_s)\Psi(\alpha T_s)$ et $G_2 = C(zI - \Phi)^{-1}\Psi((1-\alpha)T_s)$. Le logiciel Scilab permet de calculer facilement les expressions $\Phi(?T_s)$ et $\Psi(?T_s)$, il suffit donc d'utiliser une partie de la macro `dscr.sci` sur différentes durées. Je propose une macro Scilab réalisant cette discrétisation : programme `dscr_sisord.sci` (ce programme est juste écrit pour un système SISO mais peut être généralisé pour des systèmes multivariables). Cette macro donne directement la transmittance du système échantillonné en utilisant la formule proposée au paragraphe 9.5.2 : on utilise les polynômes caractéristiques.

Exemple :

Voici un exemple tiré de la documentation de Matlab.

```
-->s=%s;z=%z;g=syslin("c",10,s*s+3*s+10);
-->gh=iodelay(g,0.25)
gh =
          10
exp(-0.25*s)*-----
                2
          10 + 3s + 1s
-->Ts=0.1;//Période d'échantillonnage.
```

10 Utilisation du logiciel pour simuler les systèmes à retard pur

```
-->gh_sample=dscr_sisord(gh,Ts)
gh_sample =

$$\frac{0.0097207 + 0.0640836z + 0.0118732z^2}{0.7408182z^3 - 1.6551408z^4 + z^5}$$

Nous allons maintenant voir les équations d'état du système échantillonné obtenu :
-->gh_sample_ss=tf2ss(gh_sample)
gh_sample_ss =
gh_sample_ss(1) (state-space system:)
!lss A B C D X0 dt !
gh_sample_ss(2) = A matrix =
0.3186919 - 0.9478583 2.776D-17 - 6.939D-18 1.317D-16
0.0791092 0.0265984 0.9965110 - 2.776D-17 7.083D-17
- 0.0184431 - 0.0330652 - 0.0743338 1.3178474 5.493D-16
0.0036225 - 0.1370224 - 0.3912195 1.5403814 - 0.0028370
0.9327715 0.3136251 - 0.0824041 0.0196678 - 0.1561971
gh_sample_ss(3) = B matrix =
0.
0.
0.0558372
0.2873320
- 0.0000118
gh_sample_ss(4) = C matrix =
- 0.2251231 0. 0. - 2.168D-19 - 1.388D-17
gh_sample_ss(5) = D matrix =
0.
gh_sample_ss(6) = X0 (initial state) =
0.
0.
0.
0.
0.
0.
gh_sample_ss(7) = Time domain =
0.1
-->clean(gh_sample_ss(2))//je ne donne pas la matrice.
```

Remarquons que l'on a rendu un système continu d'ordre infini (le système continu retardé) en un système échantillonné classique (d'ordre fini, ici l'ordre est 5) : s'il n'y avait pas eu de retard, l'ordre du système serait 2. De plus la matrice d'état (**A matrix**) présente un triangle de zéros du à la présence, dans la transmittance d'un facteur en z^{-3} . On peut maintenant, facilement, comme pour les systèmes sans retards, tracer les réponses temporelles (comme par exemple la réponse indicielle voir paragraphe 10.3.1) (FIG 10.13).

Voici l'exemple précédent :

10 Utilisation du logiciel pour simuler les systèmes à retard pur

```

-->s=%s;z=%z;g=syslin("c",10,s*s+3*s+10);
-->gh=iodelay(g,0.25)
gh =
          10
exp(-0.25*s)*-----
              2
          10 + 3s + 1s
-->Ts=0.1;gh_sample=dscr_sisord(gh,Ts)
gh_sample =
          2
0.0097207 + 0.0640836z + 0.0118732z
-----
          3          4    5
0.7408182z - 1.6551408z + z
-->delay=gh.iodelay
delay =
0.25
-->deff("u=echretard(t,delay)","u=bool2s(t>delay)")
-->t=0:Ts/2:4;//Discrétisation du temps.
-->ur=echretard(t,delay);//Echelon retardé.
-->repstep=csim(ur,t,gh.H)//simulation de la réponse.
-->ue=0:Ts:4;//Discrétisation à la période d'échantillonnage.
-->repestep=flts(ones(ue),gh_sample);//Réponse échantillonnée.
-->plot(t,repstep,"r",ue,repestep,"b")
-->e=gce();e.children(1).polyline_style=2;
-->xtitle("Réponses à un échelon unitaire");
-->legend(["continu retardé";"échantillonné bloqué"]);xgrid(13);

```

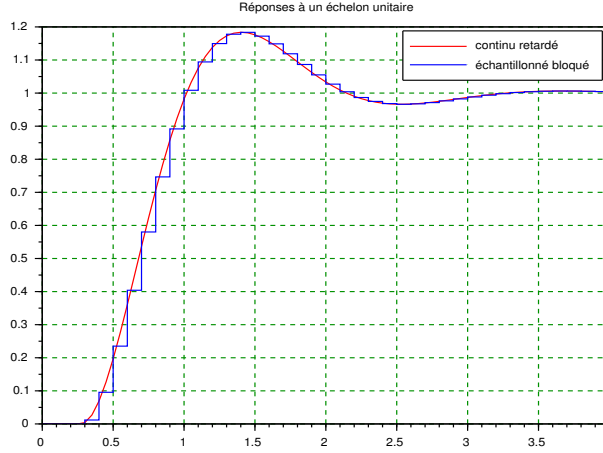


FIGURE 10.13 – Réponses indicielles

10.3.8 Approximation du retard : algorithme de Thiran

Le filtre de Thiran qui permet d'approximer le retard pur, quand on travaille avec un modèle échantillonné, fait partie des filtres numériques appelés filtres passe tout. Le modèle mathématique de ces filtres est :

$$H(z) = \frac{z^{-l} D(\frac{1}{z})}{D(z)}$$

L'exemple au second ordre donne $(D(z) = 1 + d_1 z^{-1} + d_2 z^{-2})$ avec $l = 2$, un filtre

$$H(z) = \frac{d_2 z^2 + d_1 z + 1}{z^2 + d_1 z + d_2}$$

Les coefficients du numérateur et dénominateur sont organisés dans un ordre inverse. On montre que le module de la transmittance isochrone est toujours égal à 1 quelque soit la fréquence, (filtre passe tout). Afin d'approximer un retard pur non multiple de la période d'échantillonnage Thiran a proposé un filtre passe tout de la forme :

$$H(z) = \frac{z^{-l} D(1/z)}{D(z)}$$

avec $D(z) = 1 + \sum_{n=1}^{n=l} d(n) z^{-n}$ et $d(n) = (-1)^n \binom{l}{n} \frac{(delay-l)_n}{(delay+1)_n}$ et $\binom{l}{n} = \frac{l!}{n!(l-n)!}$ et $(x)_n = (x)(x+1) \cdots (x+n-1)$. Ce filtre peut être calculé par récurrence (comme les approximants de Padé), avec les formules : $d(0) = 1$, $d(n+1) = d(n) \frac{(l-n)(l-n-delay)}{(n+1)(n+1+delay)}$, $0 \leq n \leq l-1$. Vous trouverez dans la boîte à outils proposée la macro `thiran.sci`.

Voici un programme illustrant l'utilisation de l'instruction `thiran.sci` : c'est la suite des deux programmes précédents (FIG 10.14).

```
-->gt=thiran(delay,Ts,"y") //Approximation du retard.
//ou gt=thiran(delay,Ts) : voir manuel.
gt =
1 + 0.3333333z
-----
          2   3
0.3333333z + z
-->g_sample=ss2tf(dscr(gh.H,Ts))//Discrétisation sans retard.
g_sample =
0.0406929 + 0.0449846z
-----
          2
0.7408182 - 1.6551408z + z
-->gresult=gt*g_sample//Le système échantillonné équivalent.
-->gh_sample//L'approximation avec un B.O.Z.
gh_sample =
          2
0.0097207 + 0.0640836z + 0.0118732z
-----
          3           4   5
0.7408182z - 1.6551408z + z
-->gresult=gt*g_sample//L'approximation avec thiran.sci.
gresult =
          2
0.0406929 + 0.0585489z + 0.0149949z
-----
          2           3           4   5
0.2469394z + 0.1891046z - 1.3218074z + z
-->bbode([gh_sample;gresult],0.01,10)//L'approx avec B.O.Z et Thiran.
```

10.3.9 Réponses fréquentielles d'un système continu et de son équivalent échantillonné avec Scilab : programme `dpf_cd.sci`

Nous avons, sans trop de problèmes, réussi à tracer les réponses temporelles d'un système continu avec ou sans retard, et les réponses des équivalents échantillonnés, sur une même figure. Mais quand on travaille en fréquentiel Scilab ne permet pas de mélanger des systèmes continus et des systèmes échantillonnés en particulier on ne peut définir un vecteur `[g;ge]` si `g` est un système continu et `ge` sont équivalent discrétisé. Voici un exemple :

```
-->s=%s;g=syslin("c",s*s+5*s+7,s*s+5*s+6);//Le continu.
```

10 Utilisation du logiciel pour simuler les systèmes à retard pur

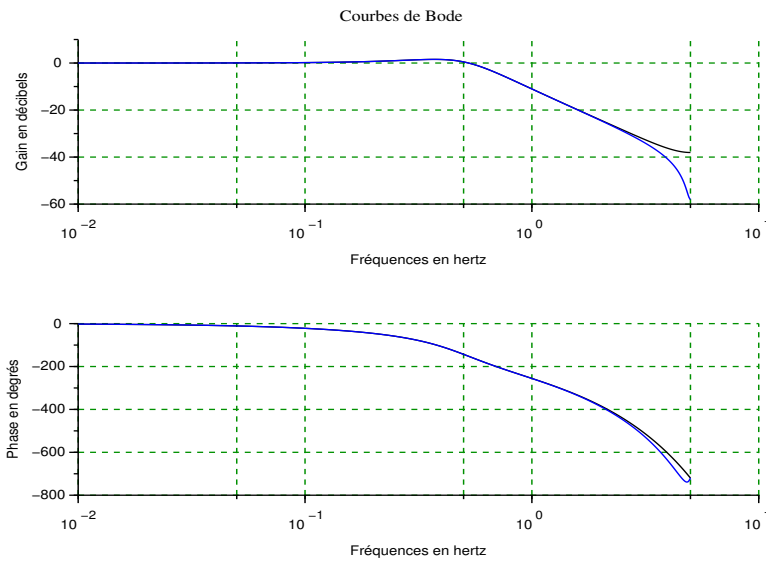


FIGURE 10.14 – Comparaison fréquentielle des approximations B.O.Z et Thiran

```
-->z=%z; Ts=.1;ge=ss2tf(dscr(g,Ts));%Discrétisation de g.
-->G=[g;ge]%/Vecteur
!--error 21
Index invalide.
at line 10 of function %lss_e called by :
at line 6 of function %r_f_r called by :
G=[g;ge]
```

Bien sur on ne peut tracer d'une seule fois les lieux de Bode ... Si on veut le faire en deux fois par superposition, il y a un problème à cause des sous fenêtres de l'instruction `bode`. Nous proposons un petit programme permettant de résoudre ce fait, ce programme se nomme `dpf_cd.sci`.

Principe de l'algorithme :

On se donne deux vecteurs colonne de systèmes; d'une part les systèmes continus (avec ou sans retard) SC et d'autre part les systèmes échantillonnés SD. Puis on se donne la **gamme** de fréquences `[fmin, fmax]` dans laquelle on va étudier ces systèmes (les continus et les échantillonnés) ou le vecteur fréquence sous la forme `[fmin:pas:fmax]` ou encore le vecteur fréquence `[f1,f2,...fn]`. Dans le premier cas avec les instructions `calfrq` ou `calfrqrd` on va discrétiser la gamme de fréquences en prenant les systèmes continus comme référence. On recherchera la fréquence de Nyquist des systèmes échantillonnés et on réalisera une homothétie sur les fréquences nécessaires au tracé des lieux échantillonnés pour que l'on ait une matrice fréquence, constituée de deux blocs, le premier pour les systèmes continus, le second pour les

10 Utilisation du logiciel pour simuler les systèmes à retard pur

systèmes échantillonnés et avec cette matrice, on calculera les deux matrices de gain et phase à l'aide de l'instruction `dbphifr` : enfin avec ces trois matrices on peut maintenant facilement tracer les divers lieux. Voici un exercice illustrant l'utilisation de cette fonction (FIG 10.15).

```
-->s=%s;z=%z;g=syslin("c",10,s*s+3*s+10); //Un système sans retard.
-->gh=iodelay(g,0.25)//Avec un retard
gh =
      10
exp(-0.25*s)*-----
              2
      10 + 3s + 1s
-->Ts=0.1;gh_sample=dscr_sisord(gh,Ts)//discrétisation avec retard.
gh_sample =
      2
0.0097207 + 0.0640836z + 0.0118732z
-----
      3      4      5
0.7408182z - 1.6551408z + z
-->g_sample=ss2tf(dscr(g,Ts))//discrétisation sans retard.
g_sample =
      0.0406929 + 0.0449846z
-----
      2
0.7408182 - 1.6551408z + z
-->nyq_f = 1/(2*Ts); //la limite en fréquence.
-->[F,D,P]=dpf_cd([g;gh],[g_sample;gh_sample],0.1,6);
-->comments=["continu sans retard";"continu avec retard";"échant sans..
-->retard";"échant avec retard"];bbode(F,D,P,comments)
Enfin le même exemple proposé en rajoutant la discrétisation du système sans retard
et l'approximant du retard avec un filtre de Thiran (FIG 10.16).
-->s=%s;z=%z;g=syslin("c",10,s*s+3*s+10); //Un système continu.
-->delay=0.25;
-->gh=iodelay(g,delay); //Avec un retard
-->Ts=0.1;gh_BOZ=dscr_sisord(gh,Ts); //discrétisation avec retard.
-->g_s=ss2tf(dscr(g,Ts)); //discrétisation sans retard.
-->gt=thiran(delay,Ts); //Filtre de Thiran.
-->gh_THI=gt*g_sample; //L'équivalent avec Thiran.
-->[F,D,P]=dpf_cd([g;gh],[g_s;gh_BOZ;gh_THI],0.1,6);
-->comments=["g";"gh";"g_s";"gh_BOZ";"gh_THI"];
-->bblack(F,D,P,comments)
```

On peut maintenant avec ces approximations échantillonnées faire l'étude des systèmes bouclés équivalents.

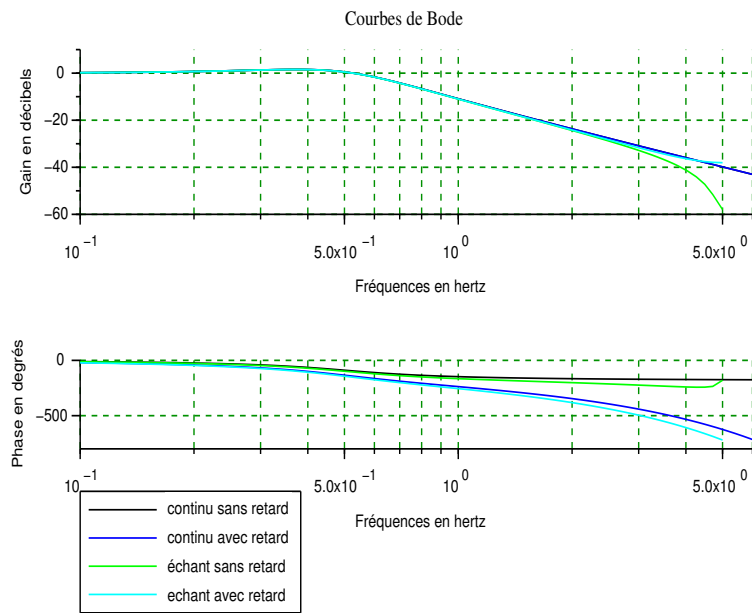


FIGURE 10.15 – Lieux de Bode systèmes continus et équivalents échantillonnés.

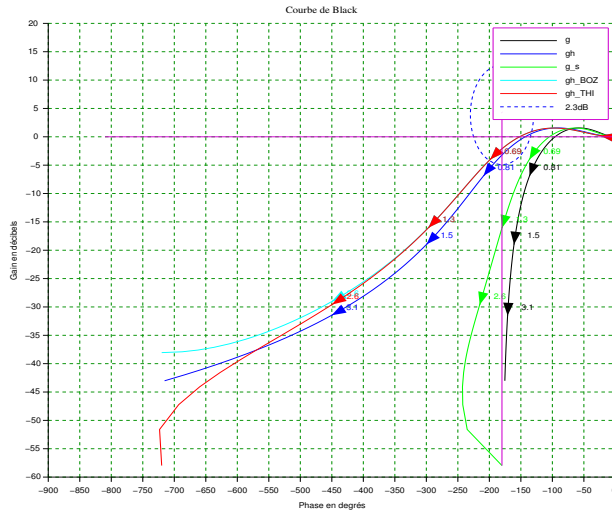


FIGURE 10.16 – Lieux de Black continus et approximants échantillonnés .

10.4 Conclusion provisoire

Que reste t'il à faire ? de nombreuses choses sont encore à programmer, en particulier il faut programmer l'étude des systèmes à retard bouclés, définir le retard interne, puis programmer par une méthode par pas « step method », la réponse du système bouclé en fonction, non pas des conditions initiales mais en fonction du passé (tenir compte de « l'histoire »). De même il faut faire, avec les fonctions proposées, la synthèse des systèmes à retard pur (calcul des correcteurs P, P.D, P.I.D, Prédicteur de Smith) comme pour les systèmes sans retard : de nombreuses thèses ont été soutenues ces dernières années sur ce sujet.

Méthode pas à pas « step method » : Le principe.

Un système d'équations différentielles retardées est un type particulier d'équations différentielles où les dérivées premières des fonctions inconnues dépendent à chaque instant, du passé des fonctions. Ainsi si on se donne une séquence de valeurs $0 \leq d_1 < d_2, \dots, < d_m$ de retard, alors on a un système d'équations de la forme :

$$\begin{aligned} \frac{d}{dt}(X(t)) &= F(t, X(t-d_1), \dots, X(t-d_m)) \\ X(t) &= G(t) \text{ avec} \\ t_0 - d &< t \leq t_0 \end{aligned}$$

Dans le cas particulier où les équations ne dépendent que d'un retard, alors le système vaut :

$$\begin{aligned}\frac{d}{dt}(X(t)) &= F(t, X(t), X(t-d)) \\ X(t) &= G(t) \text{ avec} \\ t_0 - d &< t \leq t_0\end{aligned}$$

Ainsi le problème consiste à trouver la solution sur les intervalles $[t_0, kd]$ avec $k \in \mathbb{N}_+$. Pour résoudre ce problème, on va donc intégrer ce système d'équations par pas successifs en utilisant la méthode traditionnelle d'intégration des équations différentielles (O.D.E.) sur chacun des pas. En effet à l'étape i on connaît $X(t)$ pour $t \leq t_0 + id$ à partir de l'étape précédente.

11 Introduction à l'étude des systèmes non entiers

11.1 Rappel de mathématiques

Depuis quelques années, on trouve dans la littérature scientifique, connus sous le nom de systèmes non entiers ou fractionnaires, des modèles de transfert qui font apparaître, non pas l'opérateur de Laplace s mais un opérateur de type s^α qui symbolise la dérivation non entière : donnons les trois définitions admises de cette dérivée. Plus généralement, on peut définir des systèmes non entiers non commensurables, en se donnant des opérateurs sous forme d'un vecteur $S = [s^{\alpha_1}, s^{\alpha_2}, \dots, s^{\alpha_{n-1}}, s^{\alpha_n}]$, et dans ce cas le modèle de transfert est $G_{nc} = f(S)$. Dans la suite de l'exposé je ne proposerai que des modèles de type non entier commensurables.

Définition de la dérivée non entière au sens de Riemann-Liouville La dérivée d'ordre α ($0 < \alpha < 1$) d'une fonction f continue, est par extension de la notion de dérivée, l'intégrale suivante :

$$D^\alpha(f(t)) = f(0) \frac{t^{-\alpha}}{\Gamma(1-\alpha)} + \int_0^t \frac{\theta^{-\alpha}}{\Gamma(1-\alpha)} D^1(f(t-\theta)) d\theta$$

où $u(t)$ est la fonction échelon unitaire, où $D^1(f(t))$ est la dérivée première de la fonction $f(t)$ et où $\Gamma(1-\alpha)$ est la fonction gamma pour l'argument $1-\alpha$: extension de la fonction factorielle.

Définition de Grünwald-Letnikov Cette deuxième définition de la dérivée non entière d'une fonction est donnée par une série et pourra donc être utilisée pour le calcul numérique de la dérivée.

$$D^\alpha(f(t)) = \lim_{h \rightarrow 0} \frac{1}{h^\alpha} \sum_{k=0}^{\infty} (-1)^k \binom{\alpha}{k} f(t - kh)$$

où $\binom{\alpha}{k} = \frac{\Gamma(\alpha)}{\Gamma(k)\Gamma(\alpha-k)}$.

Si la fonction est causale et si h est petit, on peut calculer numériquement la dérivée par l'expression :

$$D^\alpha(f(t_m)) = \frac{1}{h^\alpha} \sum_{k=0}^m (-1)^k \binom{\alpha}{k} f(t_m - kh)$$

En appelant h le pas de discrétisation du temps, en notant que $f(t_i) = f(ih)$ et sachant que $\mu_{j,\alpha} = (-1)^j \binom{\alpha}{k}$ on a alors la récurrence suivante :

$$\mu_{0,\alpha} = 1$$

et

$$\mu_{j,\alpha} = (1 - \frac{\alpha+1}{j})\mu_{j-1,\alpha}$$

et ceci pour $j = 1 \dots m$.

Ces deux définitions sont identiques si la fonction $f(t)$ est $n-1$ fois continûment dérivable et si l'opérateur $D^n(f)$ est intégrable sur l'intervalle de temps $[0, t]$.

Définition de la dérivée non entière au sens de Caputo La seule différence entre la dérivée au sens de Riemann-Liouville et au sens de Caputo réside en la prise en compte des conditions initiales. On verra plus loin l'introduction des conditions initiales en définissant la transformée de Laplace de ces deux dérivées (paragraphe 11.1.2).

11.1.1 Quelques propriétés de la dérivation non entière

C'est un opérateur linéaire, on peut analytiquement trouver les expressions de la dérivée α ieme de certaines fonctions relativement simplement, des exemples :

$$D^\alpha(\exp(zt)) = z^\alpha \exp(zt)$$

$$D^\alpha(\cos(\omega t + \varphi)) = \omega^\alpha \cos(\omega t + \varphi + \alpha \frac{\pi}{2})$$

$$D^\alpha(\sin(\omega t + \varphi)) = \omega^\alpha \sin(\omega t + \varphi + \alpha \frac{\pi}{2})$$

$$D^\alpha(\exp^{\frac{t}{\tau}} \sin \omega t) = R^\alpha \exp^{\frac{t}{\tau}} \sin(\omega t + \alpha \theta)$$

avec : $R = |\frac{1}{\tau} + j\omega|$ et $\theta = \arg(\frac{1}{\tau} + j\omega)$ ($j^2 = -1$)

11.1.2 Transformée de Laplace

Une autre propriété intéressante pour décrire un système par son transfert, est la transformée de Laplace de la dérivée non entière :

On montre que pour une fonction causale $f(t)$ on a :

$$L(D^\alpha f(t), s) = s^\alpha L(f(t), s)$$

pour l'opérateur de Grünwald-Letnikov, ainsi on déduit facilement des modèles sous forme transfert.

Deux exemples :

$$G_{explicite}(s) = \frac{K}{1 + (\tau s)^\alpha}$$

$$G_{implicite}(s) = \frac{K}{(1 + \tau s)^\alpha}$$

ici α est un réel, éventuellement un complexe.

Remarques :

On donne aussi les transformées de Laplace des deux opérateurs, Riemann-Liouville et Caputo qui font intervenir les conditions initiales de deux manières différentes. Pour le premier on a :

$$L[D^\alpha(f(t))] = s^\alpha F(s) - \sum_{k=0}^{m-1} s^k [D^{\alpha-k-1}(f(t))]_{t=0}$$

et pour le second,

$$L[D^\alpha(f(t))] = s^\alpha F(s) - \sum_{k=0}^{m-1} s^{\alpha-k-1} f^k(0)$$

avec $m-1 \leq \alpha < m$.

Je donnerai par la suite deux modèles issus d'expériences concrètes, mais avant introduisons la notion d'équations différentielles généralisées.

11.1.3 Equations différentielles généralisées

A partir de ces modèles on peut retrouver l'équivalent des équations différentielles ordinaires en les définissant comme des relations linéaires où intervient l'opérateur de dérivation non entière, par exemple, si l'on considère le modèle explicite, on obtient l'équation :

$$\tau^\alpha D^\alpha(y(t)) + y(t) = Ku(t)$$

et si $\alpha = 1$ on retrouve l'équation différentielle ordinaire du premier ordre. Par analogie avec les équations différentielles, on peut définir des systèmes non entiers commensurables sous la forme de rationnels, non en fonction de l'opérateur s , mais de l'opérateur s^α , (voir par exemple, l'approche par les pseudo-lieux : remarque à la fin du paragraphe 5.2.7).

11.1.4 Solutions analytiques de ces équations exotiques

A partir des propriétés de la transformée de Laplace, avec une excellente table de transformées[9], on peut quelquefois trouver l'original $y(t)$ à partir de $u(t)$ et des conditions initiales (sur la fonction et ses dérivées non entières). Ces solutions analytiques (quand on les trouve) font généralement appel aux fonctions spéciales (fonction $\Gamma()$, $Erf()$, de Mittag-Leffler ...).

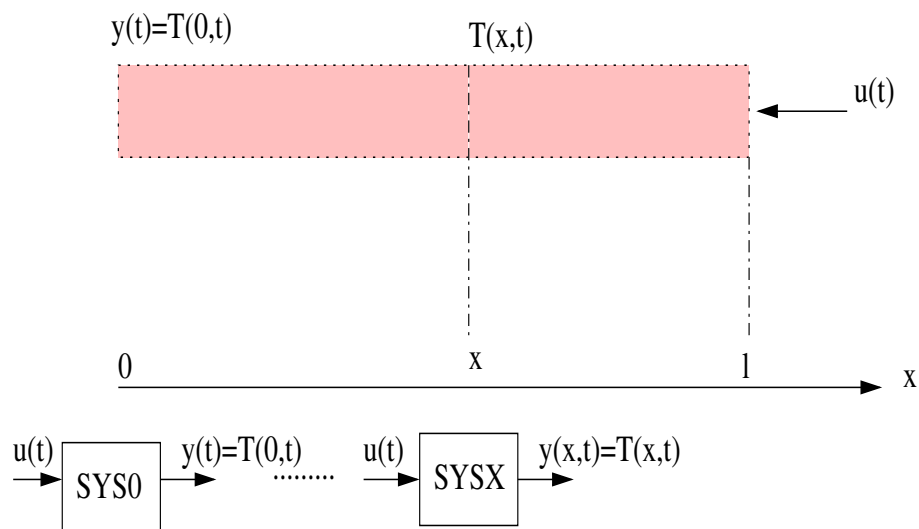


FIGURE 11.1 – Echauffement d'une barre.

11.2 Comment faire une simulation temporelle d'un système non entier ?

On entre ici dans le domaine de la recherche, il n'est pas question d'utiliser l'outil classique que l'on trouve dans les logiciels de simulation, à savoir le groupe de programmes fortran ODEPACK (qu'utilise l'instruction `csim()`), qui ne traite que des équations différentielles ordinaires (mises sous forme de systèmes différentiels ou sous forme d'équations d'état éventuellement). Mais avant donnons quelques exemples de systèmes à modèles « exotiques ».

11.2.1 Exemples de modélisation de système non entier : un système distribué

¹Je rappelle d'abord le concept de système non entier en introduisant la notion de transfert en prenant un exemple classique, à savoir l'équation de la propagation de la chaleur (diffusion). Cet exemple a été traité, dans ma jeunesse (!!), dans le cours de Méthodes Mathématiques de la Physique par la transformée de Laplace.

Considérons une barre uniforme, isolée, constituée d'un matériau homogène dont le coefficient de conduction positif est noté $C = c^2$.

On chauffe cette barre (commande : $u(t)$ flux de chaleur) à une extrémité d'abscisse $x = l$: la température de la barre $T(x, t)$ mesurée par rapport à la température initiale

¹. On lira avec intérêt le vieil article de G.JUMARIE sur la notion de pôles et résidues généralisés.[8].

T_0 (homogène) de la barre, vérifie l'équation de la chaleur à savoir

$$c^2 \frac{\partial T(x, t)}{\partial t} = \frac{\partial^2 T(x, t)}{\partial x^2}$$

avec les deux conditions (initiale et aux limites) suivantes

$$\begin{aligned} \frac{\partial T(0, t)}{\partial t} &= 0 \\ \frac{\partial T(l, t)}{\partial x} &= u(t) \end{aligned}$$

Dans tout l'exercice on prendra comme variable de sortie la température à l'autre extrémité de la barre : $x = 0$, (voir FIG 11.1), on définit pour cette variable spatiale $x = 0$ un système noté SYS0 et pour une abscisse x quelconque on aura un système noté SYSX. En prenant la transformée de Laplace de cette équation (la variable t est transformée en la variable s), on obtient :

$$\frac{\partial^2}{\partial x^2} T(x, s) = c^2 s T(x, s) - c^2 T(x, 0)$$

équation différentielle linéaire par rapport à la variable x ($T(x, s)$ est la transformée de Laplace de $T(x, t)$). En intégrant cette dernière équation compte tenu des conditions aux limites on obtient :

$$T(x, s) = y(x, s) = \frac{\exp(cx\sqrt{s}) + \exp(-cx\sqrt{s})}{c\sqrt{s}(\exp(cl\sqrt{s}) - \exp(-cl\sqrt{s}))} u(s)$$

Pour une abscisse donnée x on a un transfert (fonction de transfert) qui vaut :

$$G(x, s) = \frac{\exp(cx\sqrt{s}) + \exp(-cx\sqrt{s})}{c\sqrt{s}(\exp(cl\sqrt{s}) - \exp(-cl\sqrt{s}))}$$

et comme cas particulier pour $x = 0$ on a :

$$G(0, s) = G_0(s) = \frac{2 \exp(-cl\sqrt{s})}{c\sqrt{s}(1 - \exp(-2cl\sqrt{s}))}$$

cette fonction (de transfert n'est pas un rapport de deux polynômes de la variable complexe s) et fait de plus apparaître seulement la variable complexe \sqrt{s} , ($\alpha = 1/2$) : on peut dire que l'on a un système non entier.

Remarque : on trouvera dans les ouvrages de mathématique la solution temporelle exacte quand $u(t) = u_0$, réponse à un échelon. De même ce système étant causal on a donc, au sens des distributions, une relation entrée-sortie de la forme

$$y(0, t) = \int_0^t h_0(t - \tau) u(\tau) d\tau$$

avec $h_0(t)$ ² la transformée de Laplace inverse (au sens des distributions) de $G_0(s)$. Théoriquement cette dernière relation doit nous permettre de calculer la réponse à tout type de signal, connaissant $h_0(t)$ (théorème de convolution).

11.2.2 Exemples de systèmes non entiers, modèle à dérivée non entière

Modèle utilisé en résistance des matériaux

Quand un système fait intervenir la physique des surfaces et des interfaces, quand on étudie des procédés mettant en oeuvre des phénomènes viscoélastiques, quand on étudie la diffusion fractale, on est amené lors de la modélisation du système, à introduire la notion de dérivée non entière [6, 7] d'une variable (dérivation au sens de Riemann-Liouville) : voici un exemple de modélisation³.

L'étude d'un matériau viscoélastique en régime harmonique a conduit Vinh [10] à exprimer le module de Young complexe $E(j\omega)$ sous la forme :

$$E(j\omega) = \frac{\sigma(j\omega)}{\epsilon(j\omega)} = E_0 \frac{\prod_{i=0}^{i=n} (1 + (jT_i\omega)^{\alpha_i})}{\prod_{k=1}^{k=q} (1 + (jT_k\omega)^{\alpha_k})}$$

Une formule simplifiée de cette expression, où $\sigma(j\omega)$ est la contrainte complexe, et $\epsilon(j\omega)$ la déformation complexe, peut être utilisée pour de nombreux matériaux, et l'on obtient :

$$E(s) = \frac{\sigma(s)}{\epsilon(s)} = E_0 \frac{1 + (T_1 s)^{\alpha_1}}{1 + (T_2 s)^{\alpha_2}}$$

ce modèle dit à quatre paramètres, peut être considéré pour les automaticiens, comme un système non entier et son étude fréquentielle par Scilab ne pose pas de problèmes particuliers⁴. L'exemple proposé dans l'article de M. SOULA et Y. CHEVALIER est le suivant :

$$E(s) = E_0 \frac{1 + \frac{1}{3,23} \left(\frac{s}{650}\right)^{0,65}}{1 + \left(\frac{s}{650}\right)^{0,65}}$$

cas particulier du modèle plus général :

$$E(s) = E_0 \frac{1 + \frac{1}{Z} (T_1 s)^\alpha}{1 + (T_1 s)^\alpha}$$

cela ressemble à un réseau correcteur à retard de phase non entier : dans ce réseau on introduit un nouveau paramètre à savoir α . Je donnerais par la suite, le programme permettant d'étudier la réponse fréquentielle de ce modèle.

2. Cette réponse impulsionnelle est malgré tout une fonction au sens classique du terme : on peut trouver son expression à partir de $G_0(s)$ avec une bonne table de transformées de Laplace (A. ERDELYI, ..., TABLE OF INTEGRAL TRANSFORMS, MCGRAW HILL, NEW YORK, 1954).

3. Cet exemple est extrait de l'article de M. SOULA et Y. CHEVALIER, ESAIM Vol. 5, 1998, pages 193-204.

4. Je ne dirais pas la même chose pour l'étude des réponses temporelles !

Modèle issue de l'électrochimie www-ist.cea.fr/publiccea/

Ce modèle est celui d'un capteur permettant de détecter de manière sélective et reproductible la présence d'un composé biochimique (protéine, ADN, molécule toxique).

Cette équipe de chercheurs a introduit un modèle dit de Randles (impédance complexe de Randles) qui a la forme suivante :

$$Z(j\omega) = R_e + \frac{R_t(j\omega)^{0.5} + \sigma\sqrt{2}}{C_d R_t(j\omega)^{1.5} + \sigma\sqrt{2}C_d(j\omega)^1 + (j\omega)^{0.5}}$$

les paramètres intervenants dans cette équation sont des résistances (R_e, R_t) une capacité (C_d) et un paramètre de diffusion σ .

11.2.3 Utilisation de la transformée de Laplace

Comme je l'ai dit au paragraphe 11.2 il n'est pas question d'utiliser le groupe de programmes ODEPACK pour simuler les réponses temporelles d'un système non entier. L'idée qui vient naturellement à l'esprit quand on a modélisé un système non classique sous forme transfert est de rechercher chez les analystes numériques des méthodes permettant **d'inverser l'intégrale de Laplace**, à savoir calculer d'une manière numérique l'équation :

$$f(t) = \frac{1}{2\pi j} \int_{a-j\infty}^{a+j\infty} f(s)e^{st} ds$$

avec comme condition : $a > c$, c étant tel que $f(s)$ converge pour $R(s) > c$.

A ma connaissance il n'existe que quelques programmes fortran permettant d'inverser numériquement une transformée de Laplace (qui ne soit pas un rapport de deux polynômes, car alors on intègre un système différentiel).

11.2.4 Revue de détail (!) pour le calcul d'une réponse temporelle

Utilisation de la linéarité et de la convolution Mis à part les systèmes linéaires classiques, dans certain cas on peut avec l'aide d'une bonne table de transformée de Laplace inverse, et si l'on peut mettre la transformée de Laplace sous la forme :

$$f(s) = \sum_{k=1}^{k=N} f_k(s)g_k(s)$$

connaissant les originaux de $f_k(s)$ et de $g_k(s)$ (à savoir $f_k(t)$ et $g_k(t)$), alors on aura pour l'original de $f(s)$ l'expression :

$$f(t) = \sum_{k=1}^{k=N} f_k(t) \star g_k(t)$$

ou l'opérateur « \star » représente la convolution entre les deux fonctions $f_k(t)$ et $g_k(t)$.

La mise en oeuvre numérique de la convolution pose un certain nombre de problèmes : la précision sur le résultat peut être très aléatoire.

Inversion de fonctions méromorphes : inversion de la transformée de Laplace Si la fonction $f(s)$ dont on recherche l'original est méromorphe (i.e n'a que des pôles) alors la théorie des résidus dit que l'inversion de l'équation de Laplace se résume à calculer l'intégrale le long d'un contour fermé comprenant la droite $[c - j\infty, c + j\infty]$ et un demi cercle situé à gauche du plan complexe, de rayon infini. Si pour $t > T$ on a $|e^{st} f(s)| < A|s|^{-k}$ avec $k > 0$ l'intégrale le long du demi cercle est nulle et l'inversion de l'intégrale de Laplace revient au calcul des résidus relatifs aux pôles.

Un exemple d'inversion utilisant la formule de quadrature de Gauss Si $F(s)$ est une transformée de Laplace ayant pour expression $F(s) = s^{-\mu} G(s^{-1})$ avec $\mu \geq 1$ et $G(s^{-1})$ un polynôme de degré N , alors on trouve facilement l'original $f(t)$ numériquement en utilisant n points de la formule de quadrature de Gauss, avec $2n - 1 \geq N$. On montre que si $F(s) = s^{-\mu} G(s^{-1})$ alors :

$$f(t) = \frac{1}{2\pi j} \int_{c-j\infty}^{c+j\infty} F(s) e^{st} ds$$

soit si $u = st$:

$$f(t) = \frac{t^{\mu-1}}{2\pi j} \int_{ct-j\infty}^{ct+j\infty} e^u u^{-\mu} G\left(\frac{u}{t}\right) du$$

ou encore

$$f(t) \simeq t^{\mu-1} \sum_{k=1}^N A_k G\left(\frac{u_k}{t}\right)$$

Cette approximation donne la valeur exacte de l'original chaque fois que $G(s)$ est un polynôme en s^{-1} de degré inférieur ou égal à $2N - 1$: le problème d'analyse numérique consiste donc à trouver d'une part les valeurs de u_k et ensuite les nombres A_k .

Résolution numérique d'une équation différentielle non entière On peut comme on l'a dit dans le paragraphe précédent utiliser la deuxième expression pour la dérivée non entière et utiliser la somme :

$$D^\alpha(f(t_m)) = \frac{1}{h^\alpha} \sum_{k=0}^m (-1)^k \binom{\alpha}{k} f(t_{m-k})$$

avec h le pas de discrétisation du temps, en notant que $f(t_i) = f(ih)$ et sachant que $\mu_{j,\alpha} = (-1)^j \binom{\alpha}{k}$ on a alors la récurrence suivante :

$$\mu_{0,\alpha} = 1$$

et

$$\mu_{j,\alpha} = \left(1 - \frac{\alpha + 1}{j}\right) \mu_{j-1,\alpha}$$

et ceci pour $j = 1 \cdots m$. C'est ces équations qu'il faut programmer avec le logiciel sous forme d'une fonction écrite en C ou en fortran, ou une fonction en langage Scilab.

11.2.5 Programmes de simulation fréquentielle des systèmes non entiers

Je donnerai ici quelques exemples de simulation des réponses fréquentielles avec la boîte à outils que je propose.

- Modèles $G_{implicite}$ et modèle $G_{explicite}$. On crée dans des fichiers le modèle gain phase de ces deux fonctions de transfert, ainsi que le modèle du premier ordre qui nous servira de comparaison, en particulier pour discrétiser la gamme de fréquence d'étude. Voici les fonctions $G_{implicite}$ et $G_{explicite}$.

$$G_{explicite}(s) = \frac{K}{1 + (\tau s)^\alpha}$$

$$G_{implicite}(s) = \frac{K}{(1 + \tau s)^\alpha}$$

- Modèle utilisé en résistance des matériaux nommé $G_{resismat}$ qui ici est comparé à un retard de phase classique.

$$E(s) = 1,005 \frac{1 + \frac{1}{3,23} \left(\frac{s}{650}\right)^{0,65}}{1 + \left(\frac{s}{650}\right)^{0,65}}$$

$$G_{resismat}(s) = \frac{1 + as^\alpha}{1 + s^\alpha}$$

Le modèle est donc de la forme $G_{resismat}(s) = \frac{1+as^\alpha}{1+s^\alpha}$ et donc proche d'un modèle « retard de phase ». Afin de travailler plus simplement je prendrais des variables réduites à savoir $\frac{E(s)}{1,005}$, $\frac{s}{650}$, ici $a = \frac{1}{3,23}$ et enfin $\alpha = 0,65$.

Voici les programmes pour ces différents modèles.

Modèle explicite.

```
function[fexp,dbexp,phexp]=Gexplicite(fr,a)
//a vecteur colonne des ordres de dérivation (n,1), "a"=alpha.
//fr vecteur ligne des fréquences (1,m).
//fexp matrice des fréquences retournées (n,m).
//dbexp matrice (n,m) des gains.
//phexp matrice (n,m) des phases.
n = length(a); m = length(fr);
fexp = ones(n,1)*fr;
res = [];
for i = 1:n
    ai = a(i);
    resai = (2*%pi*%i*fr)^ai;
```

```

    res = [res;resai];
end
den = ones(n,m) + res;
dbexp = -(20/log(10))*log(abs(den));
phexp = -phasemag(den);
endfunction

```

Voici le programme donnant les réponses fréquentielles pour différentes valeurs de « α ».

```

//Exemples de transferts explicites
s=%s;G1=syslin("c",1,1+s)//ref pour calfrq.
fr=calfrq(G1,0.01,10);
//exécuter Gexplicite.sci ici, avec bouton Fichier-->Exécuter
a=[0.5;0.8;1;1.2;1.5;1.9];
[fexp,dbexp,phexp]=Gexplicite(fr,a);
bblack(fexp,dbexp,phexp,["a=0.5";"a=0.8";"a=1";"a=1.2";"a=1.5";"a=1.9"])
figure(1);
bbode(fexp,dbexp,phexp);
legend(["a=0.5";"a=0.8";"a=1";"a=1.2";"a=1.5";"a=1.9"],5);

```

Modèle implicite.

```

function[fimp,dbimp,phimp]=Gimplicite(fr,b)
//b vecteur colonne des ordres de dérivation (n,1), "b"=alpha.
//fr vecteur ligne des fréquences (1,m).
//fimp matrice des fréquences retournées (n,m).
//dbimp matrice (n,m) des gains.
//phimp matrice (n,m) des phases.
n = length(b); m = length(fr);
fimp = ones(n,1)*fr;
res = [];
for i = 1:n
    bi = b(i);
    resbi = (ones(1,m)+2*%pi*%i*fr)^bi;
    res = [res;resbi];
end
den = res;
dbexp = -(20/log(10))*log(abs(den));
phexp = -phasemag(den);
endfunction

```

Voici le programme donnant les réponses fréquentielles pour différentes valeurs de « α »

```

//Exemples de transferts implicites
s=%s;G1=syslin("c",1,1+s)//ref pour calfrq.
fr=calfrq(G1,0.01,10);//discrétisation : vecteur fréquence.
//Exécuter Gimplicite.sci ici, avec bouton Fichier-->Exécuter

```

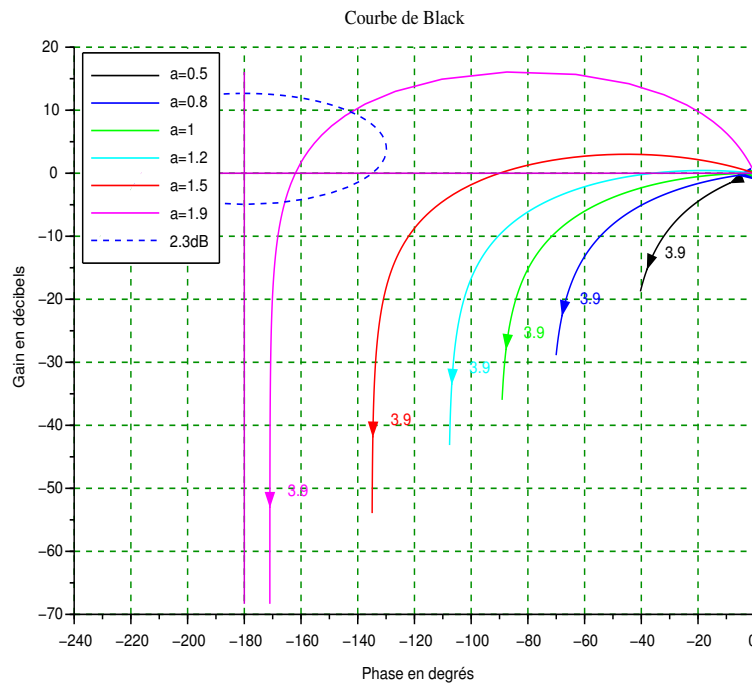


FIGURE 11.2 – Lieu de Black transfert explicite $\frac{1}{1+s^\alpha}$

11 Introduction à l'étude des systèmes non entiers

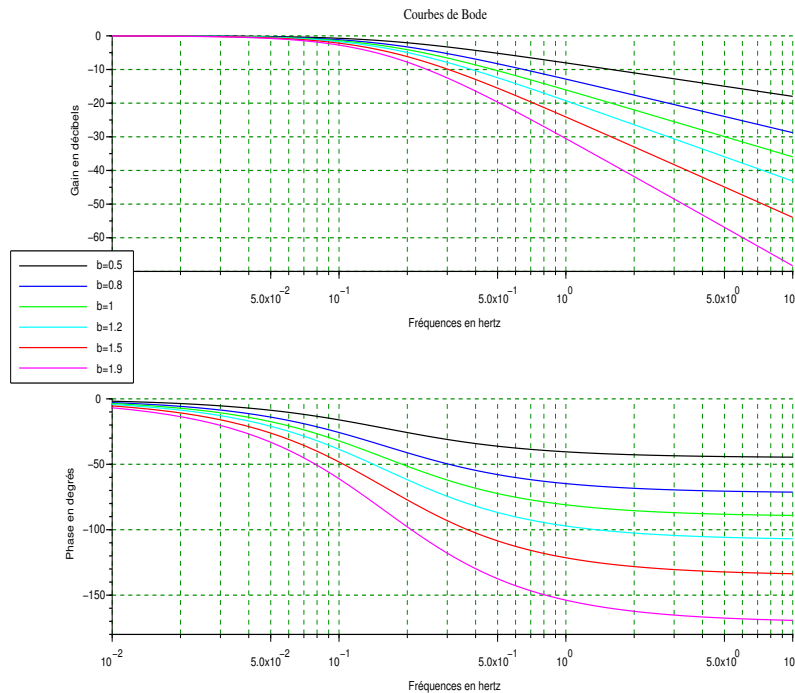


FIGURE 11.3 – Lieu de Bode transfert implicite $\frac{1}{(1+s)^\alpha}$

```
b=[0.5;0.8;1;1.2;1.5;1.9];
[fimp,dbimp,phimp]=Gimplicite(fr,b);
bblack(fimp,dbimp,phimp,...
["b=0.5";"b=0.8";"b=1";"b=1.2";"b=1.5";"b=1.9"]);
figure(1);
bbode(fimp,dbimp,phimp);
legend(["b=0.5";"b=0.8";"b=1";"b=1.2";"b=1.5";"b=1.9"],5)
```

Modèle résistance des matériaux.

Ce modèle ressemble à un réseau correcteur à retard de phase non entier, dont voici la fonction Scilab.

```
function [fmat,dbmat,phmat] = Gresismat(fr,c,d)
//c vecteur colonne des ordres de dérivation (n,1), "c"=alpha.
//d ici je prends un scalaire , mais on peut changer en vecteur.
//fr vecteur ligne des fréquences (1,m).
//fmat matrice des fréquences retournées (n,m).
```

```

//dbmat matrice (n,m) des gains.
//phmat matrice (n,m) des phases.
n = length(c); m = length(fr);
fmat = ones(n,1)*fr;
resden = [];
for i = 1:n
    ci = c(i);
    resdeni = (2*%pi*%i*fr)^ci;
    resden = [resden;resdeni];
end
resnum = d*resden;//d scalaire ici.
num = ones(n,m) + resnum;
den = ones(n,m) + resden;
dbmat=(20/log(10))*(log(abs(num))-log(abs(den)));
phmat=phasemag(num)-phasemag(den);
endfunction

```

Voici le programme donnant les réponses fréquentielles pour différentes valeurs de « α » : vecteur « c ».

```

s = %s;d = 0.6;
SREF = syslin("c",1+d*s,1+s);//retard de phase pour référence.
fr = calfrq(SREF,0.05,1);
//Exécuter Gresismat.sci ici avec bouton Fichier-->Exécuter.
c = [0.8;1;1.5;1.6;1.75;1.9];
[fmat,dbmat,phmat] = Gresismat(fr,c,d);
bode(fmat,dbmat,phmat);
legend(["c=0.8";"c=1";"c=1.5";"c=1.6";"c=1.75";"c=1.9"],5)

```

Vous trouverez dans le répertoire « `pointscosci` » les fonctions présentées, et dans un sous répertoire nommé « `exemples_nonentiers` » les trois programmes de simulation. J'ai aussi traité les trois problèmes avec la seule variable « s » et non « τs », en normalisant les problèmes (équations réduites). Ces trois problèmes vous permettent, par analogie, de traiter des problèmes plus complexes.

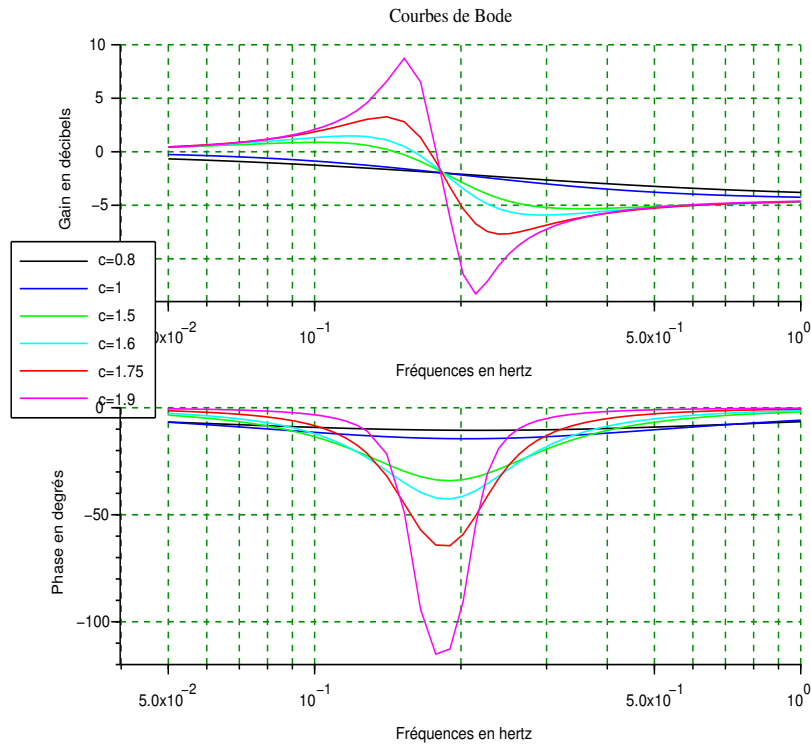


FIGURE 11.4 – Lieu de Bode des modèles $\frac{1+ds^\alpha}{1+s^\alpha}$.

12 Conclusion provisoire

Le but de ce document est d'initier l'étudiant à l'automatique de base, en éclairant à l'aide du logiciel de simulation qu'est Scilab, le premier cours d'automatique. Pour un étudiant, il serait souhaitable de refaire les exercices proposés. De même j'ai fait une petite introduction aux systèmes à retard et systèmes que j'ai appelé exotiques, le sujet reste ouvert et toute contribution et correction est souhaitable : vous connaissez mon adresse, vous pouvez aussi compléter ce document que je peux donner au format .lyx.

13 Annexes

13.1 Annexe 1 : La bibliothèque « Autoelem Toolbox »

Vous trouverez dans cette bibliothèque, un répertoire nommé « docs » et dans ce répertoire, un fichier appelé « programs.pdf » où sont décrit les principales macros contenues dans la boîte à outils : ces fonctions concernent l'algèbre et l'automatique, voici la liste :

- **Algèbre et Automatique** : Des macros de la boîte à outils « iodelay toolbox » que j'ai récupérées, car cette boîte à outils ne fonctionne plus avec Scilab-6.0. ? , macros « iodelay », « repfreqrd », « freqd », « calfrqrd », de nombreuses macros « %... » de surcharge. Encore merci à Serge Steer de l'INRIA.
- **Algèbre** : « mroots », « frac_decomp », « pade ».
- **Automatique** : « bodfact », « dbphifr », « dpf_cd », « dscr_sisord », « fdelay », « g_marginrd », « m_margin », « m_marginrd », « p_marginrd », « padedelay », « thiran ».
- **Automatique, les diagrammes** : bode, bblack, ggainplot, pphaseplot, nnyquist (utilisables avec les systèmes à retard) et des fonctions de surcharge « % ... ».

13.2 Annexe 2 : Comment faire pour utiliser votre bibliothèque

Le but de ce dernier exposé est de pouvoir d'une manière temporaire ou définitive exploiter des macros que vous avez testées avec Scilab.

- **Utilisation temporaire**

Prenons pour exemple les trois fonctions « Gexplicite.sci », « Gimplicite.sci », « Gresismat.sci », situées dans le répertoire « /home/.../Autoelem/pointscesci ».

Pour exécuter ces trois fonctions on doit faire :

`getd("/home/.../Autoelem/pointscesci")`, puis faites dans la console Scilab `who`; vous verrez apparaître les trois fonctions précédentes. Toutes les macros (fichiers texte avec le suffixe .sci) situées dans ce répertoire seront compilées et disponibles à partir de cet instant. Mais l'inconvénient de cette méthode, est qu'il faut à chaque session Scilab, retaper cette commande afin d'avoir accès aux fonctions contenues dans cette bibliothèque.

- **Utilisation définitive**

Cela revient à construire une « toolbox » que vous allez déposer ou non sur

le le site Scilab Atoms. Pour savoir comment construire votre boîte à outils rendez vous sur « [https://wiki.scilab.org/howto/Create a toolbox](https://wiki.scilab.org/howto/Create_a_toolbox) », ou allez dans le répertoire « [Scilab-6.0.0/scilab/contrib/toolbox_skeleton](https://wiki.scilab.org/howto/Create_a_toolbox) ». Si vous ne déposer pas votre toolbox sur le site « Atoms », vous devez compiler votre logiciel en utilisant le programme « `exec builder.sce` », puis pour le charger, faites de même : « `exec loader.sce` ».

13.3 Annexe 9 : Licences

Lucien Povy
lucien.povy@free.fr
Copyright © 2017 L.Povy

13.3.1 Licence GNU GPL

Les programmes qui sont dans la boîte à outils **Autoelem** sont distribués selon les termes de GPL v2.

GNU GENERAL PUBLIC LICENSE Version 2, June 1991.<http://www.gnu.org/licenses/licenses.fr.html#GPL>

13.3.2 Licence GNU FDL

Quant aux documents ils sont soumis à la licence GNU FDL. (Même site que pour la licence GNU GPL). Permission vous est donnée de distribuer, modifier, modifier des copies des pages des documents fournis, tant que ces notes sur les licences et le Copyright apparaissent.

Bibliographie

- [1] SCILAB GROUP, Scilab reference manuel, INRIA.
- [2] KUO, Automatic control systems PRENTICE HALL, New York, 1968.
- [3] J-C. GILLE, M. PELEGRIN, P. DECAULNE, Théorie et technique des asservissements, DUNOD, Paris, 1956.
- [4] P. BORNE, G. DAUPHIN-TANGUY, J.P. RICHARD, F. ROTELLA, I. ZAMBETAKIS, Analyse et régulation des processus industriels, tome 1, Régulation continue, EDITIONS TECHNIP, Paris, 1993.
- [5] J.J. DISTEFANO, A.R. STUBBERUD, I.J. WILLIAMS, Theory and problems of feedback and control systems, SCHAUUM PUBLISHING CO, New York, 1967.
- [6] A. OUSTALOUP, La Commande Crone, EDITIONS HERMES, Paris, 1991.
- [7] A. OUSTALOUP, La dérivation non entière, théorie, synthèse et applications, EDITIONS HERMES, Paris, 1995.
- [8] G. JUMARIE, Further applications of the non-stationary operational calculus to the synthesis of closed-loop distributed systems, INT. J. SYSTEMS SCI, 1977, VOL. 8, NO. 12, 1337-1364.
- [9] V. DITKINE, A. PROUDNIKOV, Transformations intégrales et calcul opérationnel, EDITIONS MIR, Moscou, 1978.
- [10] T. VINH, Sur le passage du régime harmonique au régime transitoire viscoélastique, Extrait du mémorial de l'Artillerie Française, 3e Fasc, pp.725-776, 1967.
- [11] P.NASLIN, Technologie et calcul pratique des systèmes asservis DUNOD, Paris, 1968.