REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED BOUDIAF DE M'SILA

FACULTE DES SCIENCES ET SCIENCES DE L'INGENIEUR

DEPARTEMENT D'ELECTROTECHNIQUE

MEMOIRE DE FIN D'ETUDES EN VUE DE L'OBTENTION DU DIPLOME

D'INGENIEUR D'ETAT EN GENIE ELECTROTECHNIQUE

OPTION

ELECTROMECANIQUE

THEME

DEVELOPPEMENT D'UN LOGICIEL D'ANALYSE ET DE SYNTHESE DES SYSTEMES ASSERVIS LINEAIRES CONTINUS MONOVARIABLES

Proposé et dirigé par :

Mr. Yahia LAAMARI

Présenté par :

Fawzi Mourad BISKER

Et

Badreddine LADJAL

Année universitaire 2004/2005

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOHAMED BOUDIAF DE M'SILA

FACULTE DES SCIENCES ET SCIENCES DE L'INGENIEUR

DEPARTEMENT D'ELECTROTECHNIQUE

MEMOIRE DE FIN D'ETUDES EN VUE DE L'OBTENTION DU DIPLOME

D'INGENIEUR D'ETAT EN GENIE ELECTROTECHNIQUE

OPTION

ELECTROMECANIQUE

THEME

DEVELOPPEMENT D'UN LOGICIEL D'ANALYSE ET DE SYNTHESE DES SYSTEMES ASSERVIS LINEAIRES CONTINUS MONOVARIABLES

Proposé et dirigé par :

Mr. Yahia LAAMARI

Présenté par :

Fawzi Mourad BISKER

Et

Badreddine LADJAL

Année universitaire 2004/2005

REMERCIEMENTS

En premier lieu, nous tenons à remercier vivement notre encadreur **Mr.Yahia.LAAMARI** pour les orientations et les conseils judicieux qu'il n'a cesse de nous prodiguer. Ces derniers nous ont été d'un grand soutien tout au long de l'élaboration de ce travail.

Egalement un grand merci à nos enseignants qui tout au long de notre scolarité n'ont ménagé ni leur temps ni leurs efforts contribuant de la manière la plus efficace à notre formation.

Que le chef du département d'Electrotechnique monsieur **Bachir BENDJAÏMA** trouve ici l'expression de notre gratitude et de nos remerciements les plus chaleureux pour tout le soutien qu'il nous a apporté.

Nous n'oublierons pas tous ceux qui nous ont aidés de près ou de loin durant notre étude.

Sommaire

Chapitre III — *ANALYSE TEMPORELLE*

Chapitre VI — *ANALYSE FRÉQUENTIELLE*

Chapitre V — *PERFORMANCES D'UN SYSTÈME ASSERVI*

Chapitre VI — *SYNTHÈSE DE SYSTÈMES ASSERVIS*

Chapitre **VII**

EXEMPLE D'APPLICATION

NOTATIONS

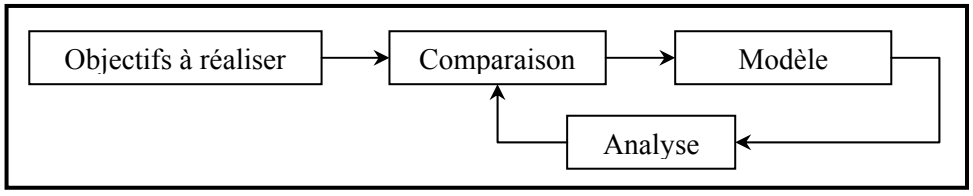
Les symboles	Leurs notations
FT	Fonction de transfert
G(P), FTBO	Fonction de transfert en boucle ouverte.
F(P), FTBF	Fonction de transfert en boucle fermée.
BO	Boucle ouverte.
BF	Boucle fermée.
SLCI	Système Linéaire Continu Invariant.
DPG	Demi-Plan Gauche de *Laplace*
CI	Conditions Initiales
DLL	Dynamic Library Link
N(p)	Le numérateur de la fonction de transfert en boucle ouverte.
D(p)	Le dénominateur de la fonction de transfert en boucle ouverte.
P	La variable de *Laplace*
σ	La partie réelle du variable complexe p
j	La constante complexe $j^2 = -1$
ω	La pulsation
φ	La phase
Im	La partie imaginaire
Re	La partie réelle
P_i	Les pôles.
Z_i	Les zéros.
A, B, C, D	Les matrices relatives à la représentation d'état.
K	Le gain de système.
e(t), x(y), u(t)	Les entrées du système.
δ (t)	L'impulsion de Dirac
s(t), y(t)	Les sorties du système.
Y_{impl}	La réponse impulsionnelle dans le domaine de temps
$\varepsilon(t)$	L'erreur
ϕ (t)	La matrice résolvante
Δ (p)	L'équation caractéristique du système
L	La transformée de *Laplace*

INTRODUCTION GENERALE

De part sa profonde nature, l'homme est toujours animé par le souci de domination. Il use de tous les moyens pour contrôler, maîtriser son environnement. La science est un moyen pacifique et efficace à même de répondre à ses aspirations. Pour mettre en relief l'utilité d'une analyse, nous allons nous inspiré d'un exemple tiré d'un domaine particulier, l'électrotechnique. Avant d'assembler des composants pour en faire un circuit, l'électrotechnicien est censé connaître le comportement du système avant même qu'il soit mis sous alimentation. C'est là où tout se précise, comment ce même manipulateur est arrivé à faire assembler n et non n+1 composants suivant une disposition bien définie ?

Bien sur, la méthode qui vient directement à l'esprit est que notre électrotechnicien se creusera un peu plus la cervelle et usera un peu moins de ses bras pour conclure qu'il fallait avant tout modéliser le système désiré puis l'analyser afin de conclure sur la nature des composants à apporter ainsi que leur disposition.

L'étude d'un système représente une phase où toutes les informations renfermées sont passées au peigne fin dans le but de comparer ses propriétés avec les spécifications recherchées. Si la comparaison est décevante, il suffira de changer le modèle jusqu'à obtenir satisfaction et donner finalement une forme concrète au modèle ainsi obtenu. La réalisation d'un système par le principe de modélisation peut être schématisée comme suit :



La deuxième méthode permettant de réaliser un système consiste à considérer un système déjà existant, mais qui présente toutefois certains écarts par rapport au modèle recherché. Il suffira d'envisager une commande en boucle fermée du système de départ, capable d'approcher le système visé.

Analyser un système n'est pas chose aisée, puisque cela demande une multitude de calculs rigoureux qui doivent mener à des résultats fiables pouvant servir de référence afin de prendre les décisions nécessaires sur les modifications à apporter.

De plus, une analyse est d'autant plus complète que les méthodes utilisées sont nombreuses. Rapidité de calcul, fiabilité des résultats et diversité des méthodes ne pourront être réunies que par l'apport de l'outil informatique, qui a connu un net développement depuis deux décennies et

qui a infiltré tous les domaines scientifiques. Il était donc indispensable de concevoir, dans le domaine de l'automatique, un logiciel qui rassemble les trois critères énoncés plus haut et qui présente une souplesse d'utilisation.

Dans ce mémoire, nous allons énoncer toutes les méthodes d'analyse, qui ont été mises en programme. En insistant notamment sur les outils mathématiques utilisés et en donnant un aperçu sur la structure du logiciel et un détail sur les procédures mises en œuvre.

Ce mémoire comportera plusieurs chapitres, disposés comme suit :

- Dans le Chapitre I, on livrera quelques définitions importantes et notions générales sur l'automatique. Suivi de quelques généralités entourant la réalisation du logiciel (langage de programmation, conception, fonctionnement, structure…).
- Le chapitre II comportera les trois méthodes classiques permettant de définir le système (Fonction de transfert, Pôles et Zéros, Espace d'état), ainsi qu'un détail sur les procédures nécessaires pour l'acquisition des paramètres définissant notre système.
- Le chapitre III est consacré aux différents types de réponses temporelles telles que les réponses en boucle ouverte et en boucle fermée, la réponse impulsionnelle.
- Dans le chapitre IV seront développer les méthodes d'analyse fréquentielle (Bode, Nichols, Nyquist, Evans, caractéristiques fréquentielles en 3 dimensions).
- Dans le chapitre V on exposera les spécifications du système ou bien leurs performances dans les domaines temporel et fréquentiel.
- Chapitre VI est consacré à la correction des systèmes asservis au moyen des régulateurs analogiques classiques (P, PD, PID,…).
- Le chapitre VII présente un exemple illustratif inspiré d'un projet de fin d'études d'ingénieurs Electrotechniciens, où on se proposera d'asservir un groupe moteur-générateur.

En fin, une conclusion générale qui met en évidence l'accent sur les conditions dans lesquelles le logiciel a été réalisé et l'intérêt que porte ce dernier en pratique suivi de quelques suggestions.

Annexes I, on présente les menus développés (menu Nyquist, menu d'impression, menu Editeur d'options de graphe, …).

Annexes II, comprend toutes les fonctions utilisées dans notre logiciel.

Annexes III, porte sur le code source des fonctions importantes développées et utilisées dans le logiciel (Laguerre, leverrier…).

CHAPITRE I

NOTIONS GENERALES ET L'OUTIL DE PROGRAMMATION

1.1. INTRODUCTION

L'automatique est l'art d'analyser, de modéliser puis de commander les systèmes. C'est aussi celui de traiter l'information et de prendre des décisions. Ses domaines d'application sont aussi nombreux que variés : mécanique, électromécanique, électronique, chimique, biotechnologie, industrie spatiale, industries de transformation, économie, ... Composante des systèmes techniques, son étude est essentielle pour appréhender les sciences industrielles.

Les asservissements linéaires continus (ou systèmes asservis linéaires continus, en abrégé SALC) constituent la branche de l'automatique qui traite les phénomènes physiques sous forme analogique (évolution continue des variables d'un système isolé).

Ce chapitre contient deux parties essentielles, dans la première on définit quelques notions générales sur les systèmes asservis linéaires continus monovariables invariants. Dans la deuxième on présente l'outil informatique utilisé pour le développement de notre logiciel « ProTahleel ».

1.2. SYSTEMES LINEAIRES CONTINUS ET INVARIANTS

1.2.1. SYSTEME INVARIANT

L'étude porte sur les évolutions temporelles des sorties des systèmes soumis à des entrées variables dans le temps. La modélisation mathématique se traduit par une relation entre l'évolution temporelle de la sortie s(t) en fonction de celle de l'entrée e(t) ou de celle d'une perturbation p(t). Nous supposerons que cette relation ne varie pas dans le temps : le système sera dit invariant.

1.2.2. SYSTEME LINEAIRE

Un système est Linéaire s'il possède la propriété suivante :
Si $s_1(t)$ est la sortie obtenue en appliquant $e_1(t)$ et $s_2(t)$ est la sortie obtenue en appliquant l'entrée $e_2(t)$, alors pour tout α réel et pour tout β réel, en appliquant l'entrée $(\alpha e_1(t) + \beta e_2(t))$ le système génère la sortie $(\alpha s_1(t) + \beta s_2(t))$; Cette propriété des systèmes linéaires est aussi appelée « Principe de superposition ».

Dans la plupart des cas on essaie de se ramener à l'étude d'un système linéaire. En effet, le principe de superposition simplifie beaucoup les problèmes: en particulier, on peut distinguer l'étude des conditions initiales d'une part et l'étude du comportement dynamique d'autre part.

Note à bien qu'un système peut être linéaire et/ou invariant : les deux propriétés sont indépendantes.

1.2.3. SYSTEMES CONTINUS

Un système physique est dit continu si les grandeurs qui le caractérisent sont de nature continue : l'information que représentent ces grandeurs est disponible à chaque instant et peut prendre toutes les valeurs possibles entre deux limites. Leur évolution dans le temps est un signal continu au sens mathématique du terme. Il est évident que d'un point de vue microscopique, aucun système physique n'est continu. Aussi dirons- nous plus simplement qu'un système est continu s'il peut être représenté avec une précision suffisante par un ensemble de relations continues, et étudié au moyen de méthodes appelées méthodes d'études des systèmes continus.

1.3. LIMITE DE VALIDITE DE L' HYPOTHESE DE LINEARITE

Un système linéaire est un système pour lequel les relations entre les grandeurs d'entrée et les grandeurs de sortie peuvent se mettre sous la forme d'un ensemble d'équations différentielles linéaires à coefficients constants (si le système n'est pas rigoureusement linéaire, on arrive tout de même souvent à le linéariser autour d'un point de fonctionnement).

1.4. CATEGORIES DES SYSTEMES DE COMMANDE

On distinguera deux types de systèmes :

1.4.1. SYSTEME EN BOUCLE OUVERTE

Les systèmes non bouclés pour lesquels aucun contrôle de l'exécution de la commande n'est réalisé. Si des phénomènes parasites perturbent le comportement du système aucune réaction compensatoire ne peut être automatiquement réalisée.

On parle alors de système en boucle ouverte (BO).

1.4.2. SYSTEME EN BOUCLE FERMEE

Les systèmes bouclés pour lesquels un contrôle de l'exécution est fait par rétroaction de la sortie du système sur son entrée. Ce sont les SA. On parle de système en boucle fermée (BF).

Un système est dit asservi lorsqu'une grandeur de sortie suit aussi précisément que possible les variations de la grandeur d'entrée (ordre ou consigne) quelque soient les effets perturbateurs extérieurs.

On peut classer les systèmes asservis de deux manières suivant le type d'entrée de référence :

Lorsque la consigne est indépendante du temps le système est dit Régulateur, par exemples *la régulation de température d'un local, la régulation de la vitesse de rotation d'un lecteur de disquettes, la régulation du débit d'une vanne.*

Lorsque la consigne dépend du temps, on parle d'asservissement ou de système Suiveur par exemples *les asservissements de position des bras de robots, les asservissements de position d'axe d'un machine outil à commande numérique (MOCN), les asservissements de position d'une antenne radar.*

1.5. CLASSIFICATION DES ENTREES - SORTIES

Un système est un ensemble directionnel. Les signaux d'entrée agissent en tant que causes, les effets s'observent sur les sorties.

1.5.1. LES ENTREES

Les entrées de commande permettent d'agir le système et de le piloter vers un but spécifié. Les entrées de perturbation sont subies sans qu'on puisse agir sur elles, leur mode d'action est parfois mal identifié.

1.5.2. LES SORTIES

Elles permettent d'observer le système ; elles dépendent du point de vue où on se place. On peut ainsi observer la vitesse ou bien le couple de moteur.

L'état du système est une notion moderne. Il permet d'observer depuis plusieurs points de vue, à l'aide de variables dites variables d'état. Le programme limite l'étude à des systèmes monovariables (une seule entrée et une seule sortie).

1.6. LE LOGICIEL

1.6.1. INTRODUCTION

Il était indispensable de présenter les grandes lignes du logiciel avant d'entrer dans les détails, dans le but évident de faire apparaître une vision globale sur le fonctionnement du logiciel, ce qui facilitera énormément sa compréhension.

1.6.2. POURQUOI LE DELPHI 7

Borland Delphi est un environnement de programmation visuelle orienté objet permettant de développer des applications 32 bits en vue de leur déploiement sous Windows et sous Linux.

En utilisant Delphi, vous pouvez créer de puissantes applications avec un minimum de programmation.

Delphi propose un ensemble d'outils de conception pour le développement rapide d'applications (RAD), dont des experts programmateur et des modèles d'applications ou de fiches, et gère la programmation orientée objet avec une bibliothèque étendue de classes qui comprend les éléments suivants :

• La *bibliothèque de classes VCL* comprend des objets qui encapsulent l'API Windows ainsi que d'autres techniques de programmation utiles (Windows).

• La *bibliothèque de composants Borland multiplate-forme* (CLX), qui contient des objets encapsulant la bibliothèque Qt (Windows ou Linux).

Parmi les avantages importants de Delphi c'est qu'il peut construire des fichiers ressources de type DLL.

1.6.3.3. LES FICHIERS DLL

DLL, "Dynamic Library Link" ou en français "lien dynamique vers une librairie", le fichier DLL est cette librairie. Le but étant au départ de permettre aux développeurs de bénéficier de fonction déjà existante et aussi de décharger la mémoire. Les DLL contiennent en effet des ressources qu'elles peuvent partager avec plusieurs programmes. C'est ressources permettent d'avoir accès à des fonctions, des composants...

1.6.3.4. QU'EST CE QU'UNE DLL

Qu'est ce qu'une DLL me direz-vous ? Une dll est un fichier contenant des ressources et la capacité de mettre à disposition ces ressources pour les programmeurs et donc pour les applications. Une DLL peut contenir des fonctions (exemple des fonctions mathématiques), une

classe, des composants ou d'autre chose comme des icônes.
Pour ce qui est de son utilité, on peut en distinguer trois :

- La principale est qu'elle permet de partager ses ressources avec tout le monde. Vous pouvez distribuer ou récupérer une dll et ensuite l'utiliser ses ressources. Exemple si vous créez ou récupérer une DLL contenant des fonctions mathématiques, vous n'avez plus besoin de les réaliser dans votre application, il suffit de les importer. Leur utilisation se fait alors comme celle que l'on trouve dans Delphi comme *sqrt*.
- La seconde est qu'elle permet de découper son application en morceau afin d'améliorer la maintenance et le débogage. Si vous modifier une partie du code dans une dll, il vous suffit de redistribuer votre dll pour mettre l'application à jour. Cela permet aussi de séparer par fonctionnalité votre code. On peut dire que l'intérêt de ce point est un peu flou. Il est quasiment inexistant pour les petites applications, il suffit de faire plusieurs unités et on peut aussi utiliser l'héritage, nous ne parlons pas ici de l'héritage des classes quoique mais de celui des unités (Pour Rappel : on peut inclure des unités dans son projet en spécifiant qu'ils ne sont qu'une copie de l'original ou un lien, ce qui implique qu'elles se mettent à jour si l'original est modifié). Reste le problème de la compilation qui concerne tout le projet au contraire de la DLL où seul la dll concernée est à recompiler.
- Le troisième est que l'on peut charger dynamiquement une DLL. Quand une DLL est chargée dynamiquement dans une application, elle ne réside en mémoire que lorsqu'elle est utilisée (appelé) et ensuite la mémoire est libérée (la dll est déchargée).

1.6.3. REALISATION DU LOGICIEL

La réalisation du logiciel, qui a nécessité de longs mois de travail, a été inspirée d'un assemblage de plusieurs taches, ou chacune d'elles réalise une fonction bien définie (saisie, analyse, configuration), structurées de manière arborescence. Cette technique de programmation facilite la clarté et surtout la gestion du programme dont le principal objectif est d'évité des disfonctionnements du programme. Le temps, qu'a nécessité la programmation, ce divise comme suit :

- Environ 45% pour la mise en programme des méthodes d'analyse et des options graphiques.
- Environ 55% pour la gestion de ces méthodes et du programme.

Donc, une importance considérable est accordée à la gestion, sachant pertinemment qu'un logiciel n'a de nom que s'il est bien géré.

1.6.4. FONCTIONNEMENT DU LOGICIEL

Le logiciel est constitué de plusieurs menus, comme l'indique le schéma simplifié suivant :

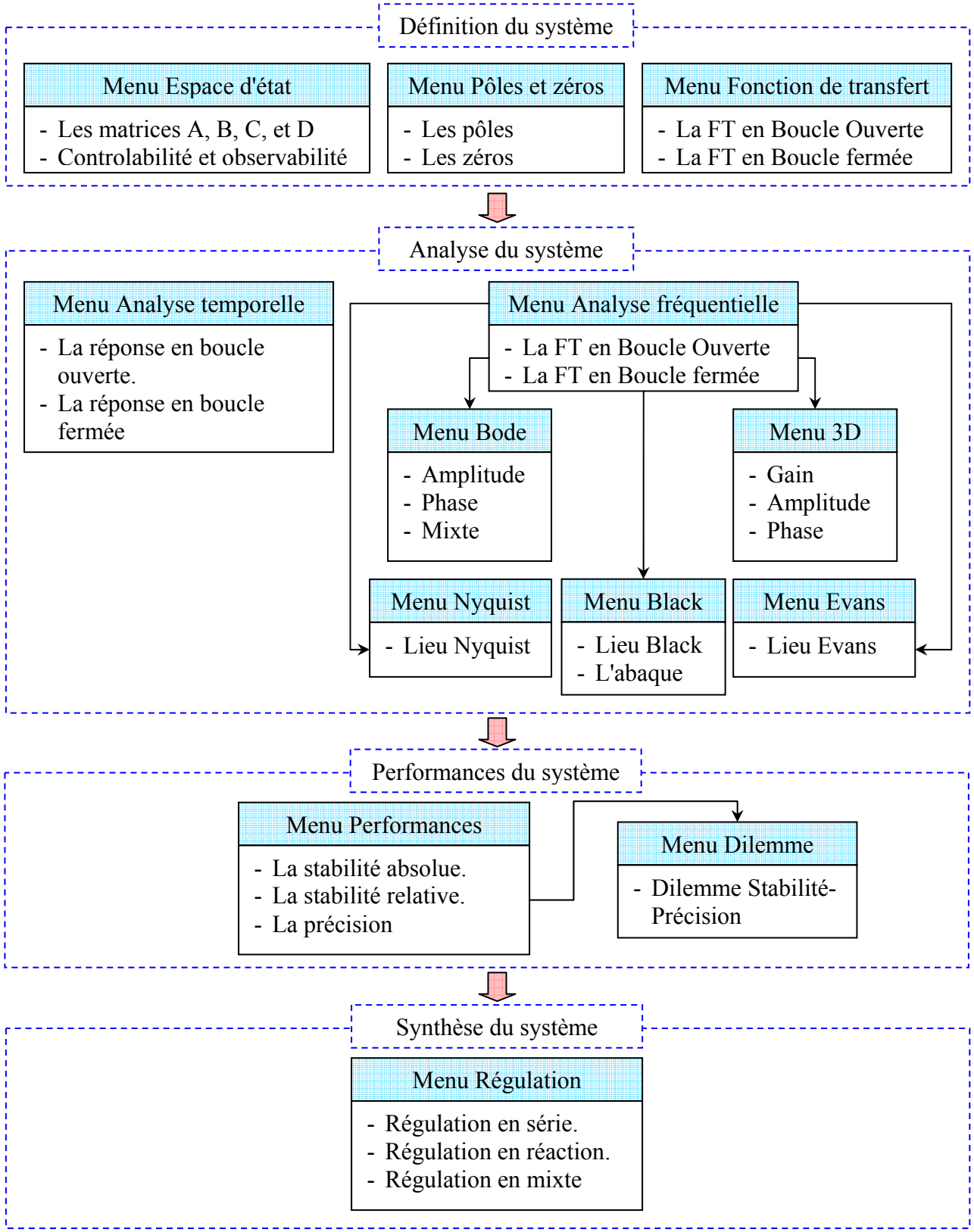


Figure 1.1: schéma simplifié qui représente le fonctionnement de logiciel

1.7. **CONCLUSION**

L'idée-force, qui nous a servi d'inspiration est l'exploitation sachant qu'une analyse est d'autant plus complète que les informations recueillies sont nombreuses et fructueuses. Nous avons essayé de rassembler pratiquement toutes les méthodes classiques de l'analyse des systèmes, que ce soit dans les domaines temps ou fréquence. A ces méthodes d'analyse s'ajoutent les instruments graphiques permettant d'exploiter au maximum une simple courbe issue d'une analyse afin de soutirer un maximum d'informations.

Bien entendu, un logiciel n'est jamais complet et il reste toujours des options à ajouter. Nous espérons toutefois que le travail réalisé satisfasse les besoins des futurs utilisateurs qu'ils soient ingénieurs-chercheurs dans leurs travaux pratiques ou dans les projets de fin d'étude.

CHAPITRE II

REPRESENTATIONS DU SYSTEME

2.1. INTRODUCTION

Dans ce chapitre, nous exposons d'abord les principales représentations classiques d'un système asservi linéaire continu monovariable et le mode de passage entre elles. Cela est réalisé pratiquement dans notre logiciel, cette réalisation représente une phase essentielle et nécessaire avant d'entamés l'analyse de système.

2.2. REPRESENTATION PAR LA FONCTION DE TRANSFERT

La relation entre la sortie et l'entrée du système (SLCI) se représente par une équation différentielle linéaire à coefficients constants. Plus généralement, pour un SLCI, la relation entre l'entrée et la sortie prend la forme :

$$a_n \frac{d^n s(t)}{dt^n} + a_{n-1} \frac{d^{n-1} s(t)}{dt^{n-1}} + ... + a_0 s(t) = b_m \frac{d^m e(t)}{dt^m} + b_{m-1} \frac{d^{m-1} e(t)}{dt^{m-1}} + ... + b_0 e(t) \qquad (2.1)$$

Des considérations sur la nature causale des systèmes physiques imposent que l'on ait systématiquement : $m \leq n$.

La solution générale d'une telle équation différentielle est la solution générale s_0 de l'équation sans second membre plus une solution particulière s_1 de l'équation complète.

La théorie permet de déterminer la sortie *s(t)* en fonction de l'entrée *e(t)*: il s'agit de la réponse *s(t)* du système à une excitation *e(t)* Aussi, il faut connaître l'entrée à appliquer pour obtenir une sortie particulière. Plutôt que de chercher à la résoudre directement, on peut utiliser la transformée de *Laplace*, ce qui va permettre de trouver une relation fractionnelle entre l'entrée et la sortie.

Pour résoudre l'équation différentielle précédente, Supposons les conditions initiales nulles et appliquons la transformation de *Laplace* :

$$(a_n p^n + a_{n-1} p^{n-1} + ... + a_1 p + a_0) S(p) = (b_m p^m + b_{m-1} p^{m-1} + ... + b_1 p + b_0) E(p) \qquad (2.2)$$

On pose :

$$G(p) = \frac{S(p)}{E(p)} = \frac{(b_m p^m + b_{m-1} p^{m-1} + ... + b_1 p + b_0)}{(a_n p^n + a_{n-1} p^{n-1} + ... + a_1 p + a_0)} \qquad (2.3)$$

Et l'on obtient :

$$S(p) = G(p) E(p)$$

G(*p*) est appelée Fonction de Transfert (FT) du système ; c'est le rapport de la transformée de *Laplace* de la sortie sur la transformée de *Laplace* de l'entrée sous l'hypothèse des conditions de *Heaviside*.

La fonction de transfert peut s'écrire dans le corps des nombres complexes sous la forme :

$$G(p) = K\frac{(p-Z_1)(p-Z_2)...(p-Z_m)}{(p-p_1)(p-p_2)...(p-p_n)} = \frac{N(p)}{D(p)} \qquad (2.4)$$

Les Z_i sont les zéros de la FT, et les p_i sont les pôles de la FT. En isolant les zéros et les pôles nuls de la FT, on obtient une autre écriture :

$$G(p) = \frac{K\prod_{i=1}^{m}\left(1-\frac{p}{Z_i}\right)}{p^{\alpha}\prod_{j=1}^{n-\alpha}\left(1-\frac{p}{p_j}\right)} \qquad \text{Où} \qquad K = \frac{k\prod_{i=1}^{m}(-Z_i)}{\prod_{j=1}^{n-\alpha}(-p_j)} \qquad (2.5)$$

K : est appelé le gain de la fonction de transfert (gain statique pour $\alpha = 0$),

α : est la classe de la fonction de transfert,

n : est l'ordre de la fonction de transfert (degré de *D(s)*).

❖ ***Remarque*** *:* Dans le cas où les conditions initiales (CI) ne sont pas toutes nulles, on peut définir un polynôme $N_0(p)$ en p qui dépendant des CI, tel que

$$S(p) = \frac{N(p)}{D(p)}E(p) + \frac{N_0(p)}{D(p)} \qquad (2.6)$$

En conclusion, la représentation d'un système physique par la fonction de transfert G(p) est l'une des méthodes les plus utilisées, (l'inconvénient majeur de cette représentation est que les conditions initiales sont supposées nulles et qu'elles ont une influence directe sur le régime transitoire du système). La réponse totale d'un système s'énonce comme suit :

« La réponse globale d'un système à une sollicitation donnée est égale à la somme, d'une part, de la réponse du système aux conditions initiales (avec entrée nulle), et d'autre part la réponse que donnerait le système à la sollicitation *u(t)* seule (conditions initiales nulles) ».

$$s(t) = s_1(t) + s_2(t) \qquad (2.7)$$

$s_1(t)$: Réponse à conditions initiales nulles

$s_2(t)$: Réponse à une entrée nulle.

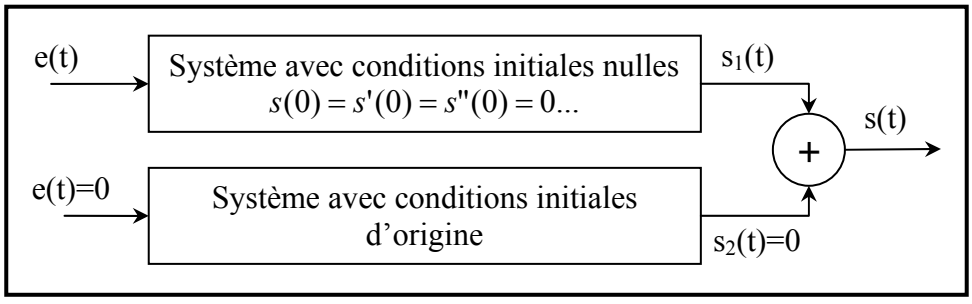


Figure 2.1 : schéma simplifié expliquant la réponse globale d'un système

2.3. REPRESENTATION PAR LES POLES ET LES ZEROS

Etant donné que la fonction de transfert d'un système linéaire se présente sous forme d'une fraction rationnelle de la variable p, elle peut être décomposée en éléments simples. La décomposition donne :

$$G(p) = \frac{K\prod_{i=1}^{p}(p-a_i)}{p^{\alpha}\prod_{j=1}^{q}(p-b_j)} \frac{\prod_{k=1}^{r}\left[(p-a_k)^2+\beta_k^2\right]}{\prod_{li=1}^{s}\left[(p-\delta_{li})^2+\lambda_{li}^2\right]} \tag{2.8}$$

Dans cette expression les a_t et les b_j représentent les racines réelles simples du numérateur et du dénominateur de l'équation G(p) ; les quantités $a_k \pm i\beta_k$ et $\delta_{li} \pm i\lambda_{li}$ représentent les racines complexes du numérateur et du dénominateur de G(p). Par ailleurs, la fonction de transfert (2.8) présente des Zéros de transmission réels et complexes en $p = a_i$ et $p = a_k \pm i\beta_k$ et une des caractéristiques des systèmes à minimum de phase est qu'ils possèdent tous leurs pôles et tous leurs zéros situés à l'intérieur du demi-plan gauche de *Laplace* (DPG), l'origine en p=0 du plan étant inclue dans le (DPG) de *Laplace*.

De plus, il sera démontré ultérieurement que les systèmes linéaires stables ont la particularité d'avoir tous leurs pôles réels et les parties réelles de leurs pôles complexes qui sont situées dans le DPG de *Laplace*. Cette propriété sera démontrée et analysée lors de l'étude de la stabilité des systèmes.

Physiquement parlant, il est très difficile d'identifier les pôles et zéros du système. Par conséquent, on se limite à les calculer analytiquement à partir de la fonction de transfert. C'est une représentation qui se veut moins mathématique et plus physique mais n'est en fait qu'une reformulation de G(p).

En effet, c'est ce mode de représentation puisqu'on simule une représentation par des points singuliers du système qui sont physiquement « intouchables ». Mais elle constitue tout de même un pas supplémentaire dans la compréhension et la maîtrise de la notion de système.

2.4. REPRESENTATION PAR L'ESPACE D'ETAT

2.4.1. INTRODUCTION AUX VARIABLES D'ETAT

Les systèmes linéaires seront décrits à l'aide d'un nouveau concept, celui d'état. La théorie de l'espace d'état a été inspirée par celle des équations différentielles. Enonçons quelques points liés à la méthode :

- Le traitement a lieu dans le domaine temporel et fait appel à l'algèbre matricielle,
- La représentation des systèmes monovariables s'étend aux systèmes multivariables qu'on peut alors appréhender globalement,
- Les notions nouvelles de la contrôlabilité et l'observabilité apparaissent et jouent un rôle très important,
- L'outil est très puissant pour résoudre les problèmes touchant à l'optimisation,
- Certains systèmes présentant des non linéarités ou des paramètres variant dans le temps pourront être traités par la méthode des variables d'état,

2.4.2.NOTION D'ETAT

L'état à l'instant t_0 d'un système représente l'ensemble des n informations que l'on doit posséder sur le système, à cet instant t_0**,** pour pouvoir déterminer son évolution ultérieure (à $t>t_0$), à partir de la seule donnée des entrées ultérieures; ces informations, soit : $x_1(t_0)$, $x_2(t_0)$, ……, $x_n(t_0)$**,** sont nommées variables d'état, et sont rassemblées dans un vecteur $x(t_0)$ nommé vecteur d'état.

On peut dire que les variables d'état représentent l'évolution des conditions initiales, ou encore qu'elles résument tout le passé du système : les variables d'état sont la mémoire du passé.

2.4.3.NOTION DE REPRESENTATION D'ETAT (SYSTEME CONTINU)

On peut montrer que, dans la plupart des cas, l'évolution en fonction du temps du système peut être décrite par les deux équations suivantes, qui constituent la représentation d'état :

$$\begin{aligned} \dot{X} &= f(X,U,t) \text{ : Équation d'état,} \\ Y &= g(X,U,t) \text{ : Équation de sortie.} \end{aligned} \tag{2.9}$$

Dans le cas où le système considéré est linéaire, la représentation d'état se met sous forme :

$$\begin{aligned} \dot{X} &= A(t)X + B(t)U \\ Y &= C(t)X + D(t)U \end{aligned} \tag{2.10}$$

Si le système est supposé invariant, il vient :

$$\begin{aligned} \dot{X} &= AX(t) + BU(t) \\ Y &= CX(t) + DU(t) \end{aligned} \tag{2.11}$$

Avec X (t) : vecteur d'état, Y (t) : vecteur de sortie, U (t) : vecteur de commande,

A : matrice d'état, B : matrice d'entrée, C : matrice de sortie, D : matrice de couplage ; Tous les vecteurs contenants des grandeurs à temps continu.

Dans tout ce qui suit, nous nous placerons dans ce dernier cas (Système invariant).

❖ ***Remarque*** *:*

- La présentation d'état a la même structure qu'il s'agisse d'un système monovariable ou multivariable,
- Les sorties sont liées linéairement aux variables d'état et aux entrées,
- Dans le cas fréquent où D=0 le système est dit « propre »,
- Dans le schéma ci-dessous U(t), X(t), Y(t) représentent respectivement l'évolution d'une entrée, d'un état et d'une sortie quelconques.

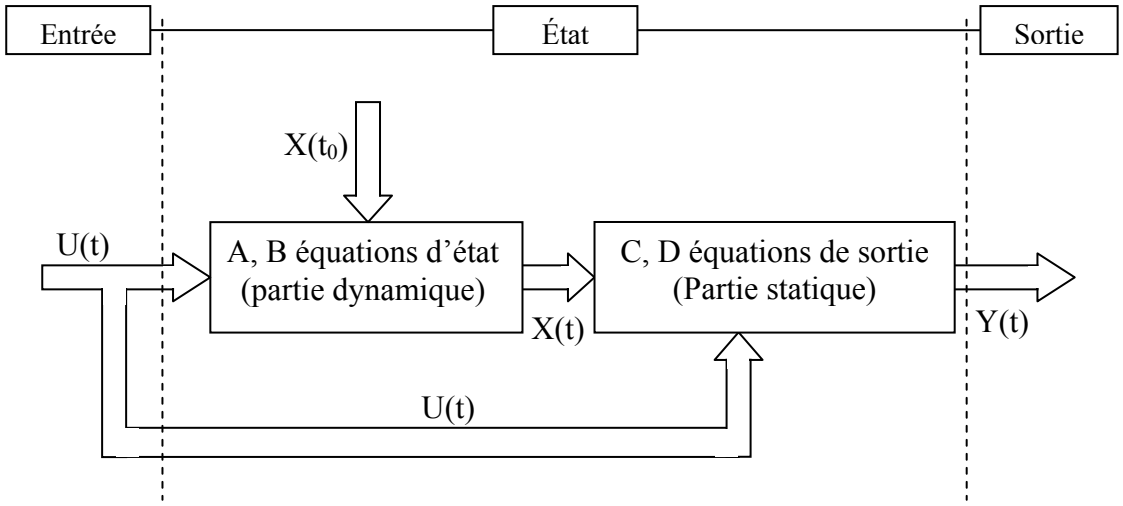


Figure 2.2 : schéma simplifié illustrant la notion de représentation d'état

2.4.4.NOTION DE CONTROLABILITE ET D'OBSERVABILITE

Jusqu'ici, nous nous sommes intéressés à un système dont les propriétés étaient mises en évidence principalement par la matrice d'état A.

Sous cette forme, on verra que les problèmes de commande sont beaucoup plus liés aux autres matrices (B, C et D) de la représentation d'état.

2.4.4.1. CONTROLABILITE DES SYSTEMES LINEAIRES

La commande disponible sur le système devra rendre possible le passage du vecteur d'état d'un point initial à un point final de l'espace d'état en un temps fini, et ceci quels que soient les points choisis ; le système sera alors dit contrôlable.

Un système linéaire, décrit par les équations d'état :

$$\begin{aligned} \dot{X} &= A(t).X(t) + B(t).U(t) \\ Y(t) &= C(t).X(t) + D(t).U(t) \end{aligned} \tag{2.12}$$

Est dit contrôlable (ou gouvernable) à l'instant t_0 si, quel que soit $X(t_0)$, et quel que soit X_f, en un temps fini. On donne les Conditions nécessaires et suffisantes suivantes :

a) Cas où le système est invariant (matrices A et B indépendantes du temps), si n désigne l'ordre du système, c'est-à-dire le rang de la matrice A, le système est contrôlable si et seulement si la matrice :

$$[\, B \,|\, AB \,|\, A^2 B \,|\ldots|\, A^{n-1} B \,|\,] \text{ est de rang n.} \tag{2.13}$$

b) Cas où la représentation d'état choisie est telle que la matrice d'état A est diagonale le système est contrôlable si et seulement si B ne possède pas de lignes nulles.

2.4.4.2. OBSERVABILITE DES SYSTEMES LINEAIRES

La sortie du système devra permettre une observation du vecteur d'état, c'est-à-dire que la connaissance, durant un temps infini, des valeurs prises par la sortie devra suffire pour déterminer l'état du système ; celui-ci sera alors dit observable.

Un système linéaire, décrit par les équations d'état :

$$\begin{aligned} \dot{X} &= A(t).X(t) + B(t).U(t) \\ Y(t) &= C(t).X(t) + D(t).U(t) \end{aligned} \tag{2.14}$$

Est dit observable à l'instant t_0 si, quel que soit $X(t_0)$, il existe un instant t_1, fini et inférieur à t_0, tel que la connaissance de y(t) pour tout $t \in [t_1, t_0]$ soit suffisante pour déterminer $X(t_0)$. On donne les Conditions nécessaires et suffisantes suivantes :

a) Cas où le système est invariant (matrices C et A indépendantes du temps), si n désigne l'ordre du système, c'est-à-dire le rang de la matrice A, le système est observable si et seulement si la matrice :

$$[C^T \,|\, A^T C^T \,|\ldots|\, (A^T)^{n-1} C^T\,] \text{ Est de rang n.} \tag{2.15}$$

b) Cas où la représentation d'état choisie est telle que la matrice d'état A est diagonale, le système est contrôlable si et seulement si C ne possède pas de colonnes nulles.

2.5. PASSAGE ENTRE LES REPRESENTATIONS

2.5.1. REPRESENTATION D'ETAT A PARTIR DE LA FONCTION DE TRANSFERT

Parmi les nombreuses formes de représentations des matrices d'état d'un système à partir de sa fonction de transfert dans le cas monovariable (une seule entrée) nous proposons trois méthodes :

2.5.1.1. METHODE DIRECTE

La fonction de transfert du système est considérée sous la forme suivante :

$$FT = \frac{b_0 + b_1 p + ...b_{n-1} p^{n-1}}{a_0 + a_1 p + ...a_n p^n}$$

n : nombre de pôles de la fonction de transfert : ordre de système (nombre minimal de variable d'état).

Avec prendre en considération la condition de la causalité.

On divise par p^n le numérateur et le dénominateur de la fonction de transfert ; puis on introduit $V(p)$ de façon à écrire :

$$\begin{aligned} Y(p) &= \left(b_0 p^{-n} + b_1 p^{-n+1} + ... + b_{n-1} p^{-1}\right) V(p) \\ U(p) &= \left(a_0 p^{-n} + a_1 p^{-n+1} + ... + a_{n-1} p^{-1} + 1\right) V(p) \end{aligned}$$

On définit alors les n variables d'état de la façon suivante :

$$\begin{cases} X_1(p) = p^{-n} V(p) \\ X_2(p) = p^{-n+1} V(p) = p X_1(p) \\ X_3(p) = p^{-n+2} V(p) = p X_2(p) \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ X_n(p) = p V(p) = p X_{n-1}(p) \end{cases}$$ En supposant les conditions initiales nulles

$$\begin{cases} \dot{X}_1(t) = X_2(t) \\ \dot{X}_2(t) = X_3(t) \\ \vdots \qquad\qquad \vdots \\ \dot{X}_{n-1}(t) = X_n(t) \\ \dot{X}_n(t) = V(t) \end{cases}$$

$$A=\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{bmatrix} \quad B=\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \quad C=\begin{bmatrix} b_0 & b_1 & \cdots & b_{n-2} & b_{n-1} \end{bmatrix} \quad D=[0]$$

La représentation obtenue est dite *Compagnon pour la commande*. Les variables d'état correspondantes sont couramment appelées *variable de phase*.

2.5.1.2. METHODE DE HONER

On reprend l'expression de la fonction de transfert du paragraphe précédent on divise à nouveau par p^n le numérateur et le dénominateur on obtient :

$$Y(p)\left(a_0 p^{-n} + a_1 p^{-n+1} + ... + a_{n-1} p^{-1} + 1\right) = U(p)\left(b_0 p^{-n} + b_1 p^{-n+1} + ... + b_{n-1} p^{-1}\right)$$

D'où l'on tire : $Y(p) = p^{-1}\left[(b_{n-1}U(p) - a_{n-1}Y(p)\right] + p^{-2}\left[(b_{n-2}U(p) - a_{n-2}Y(p)\right] + ...$

Posons :

$$\begin{cases} \dot{X}_1(t) = b_0 U(t) - a_0 Y(t) \\ \vdots \\ \dot{X}_i(t) = b_{i-1}U(t) - a_{i-1}Y(t) + X_{i-1}(t); \quad \text{i} > 1 \end{cases}$$

Alors : $Y(t) = X_n(t)$

D'où la représentation d'état :

$$A=\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -a_0 & -a_1 & -a_2 & \cdots & -a_{n-1} \end{bmatrix} \quad B=\begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{bmatrix} \quad C=\begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \quad D=[0]$$

La représentation obtenue est dite *Compagnon pour l'observation***.**

2.5.1.3. METHODE DES MODES PROPRES

On décompose la fonction de transfert en éléments simples, de façon à faire apparaître les différents modes propres du système. Dans le cas où les pôles sont réels et simples, cela donne :

$$FT = \frac{Y(p)}{U(p)} = \frac{\alpha_1}{p+\lambda_1} + \frac{\alpha_2}{p+\lambda_2} + ... + \frac{\alpha_n}{p+\lambda_n} \tag{2.16}$$

Et l'on définit les variables d'état par :

$$X_i(p) = \frac{U(p)}{p + \lambda_i} \tag{2.17}$$

Alors : $$p.X_i(p) = -\lambda_i.X_i(p) + U(p)$$

Il en résulte : $$\dot{x}_i(t) = -\lambda_i.x_i(t) + u(t) \tag{2.18}$$

En outre : $$y(t) = \alpha_1 x_1(t) + \alpha_2 x_2(t) + ... + \alpha_n x_n(t) \tag{2.19}$$

D'où la représentation d'état :

$$A = \begin{bmatrix} -\lambda_1 & 0 & \cdots & 0 & 0 \\ 0 & -\lambda_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & -\lambda_{n-1} & 0 \\ 0 & 0 & \ldots & 0 & -\lambda_n \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix} \quad C = \begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_{n-1} & \alpha_n \end{bmatrix} \quad D = [0]$$

2.5.2. LA FONCTION DE TRANSFERT A PARTIR DE LA REPRESENTATION D'ETAT

2.5.2.1. METHODE DE LEVERRIER

2.5.2.1.1. SITUATION DU PROBLEME

Connaissant les matrices A, B, C et D, on peut analytiquement déduire la fonction de transfert du système par la formule :

$$G(p) = C.(p.I - A)^{-1}.B + D \tag{2.20}$$

Or d'un point de vue programmation, cette formulation ne peut être explicitée directement puisque la variable p demeure une variable pure donc sans valeur définitive à lui affecter.

Comment développer une méthode qui outrepasse cette contrainte « variable » et qui aboutit au résultat recherché? Leverrier propose de construire ce qu'il appelle la matrice résolvante :

$$\phi(p) = (p.I - A)^{-1} = \frac{W(p)}{\Delta(p)} \tag{2.21}$$

$\Delta(p)$: Équation caractéristique du système

$$\begin{cases} \Delta(p) = p^n + \sum_{i=1}^{N} a_i.p^{n-i} \\ W(p) = \sum_{i=1}^{N} V_i.p^{n-i} \end{cases}$$

Où les V_i sont des matrices carrées d'ordre N, alors la méthode s'énonce comme suit :

$$\begin{cases} V_1 = I \\ V_2 = V_1.A + a_1 I \\ \vdots \quad \vdots \\ V_n = V_{n-1}.A + a_{n-1} I \end{cases} \qquad \begin{cases} \mathrm{a}_1 = -tr(A) \\ \mathrm{a}_2 = -\frac{1}{2} tr(V_2.A) \\ \vdots \quad \vdots \\ \mathrm{a}_n = -\frac{1}{n} tr(V_n.A) \end{cases} \tag{2.22}$$

Où tr(M) représente la trace de la matrice M qui n'est autre que la somme des éléments de la première diagonale. Donc si,

$$G(p) = C.(p.I - A)^{-1}.B = \frac{Num(p)}{Beno(p)}$$

On a $N[i] = C.V_i.B$

Avec C : matrice d'observabilité

B : Matrice de commande

V_i : Matrice numéro i de $\phi(p)$

Et finalement :

$$G(p) = \frac{Num(p)}{Beno(p)} = \frac{N(p) + D.\Delta(p)}{\Delta(p)}$$

Où D est la matrice de transfert direct qui intervient lorsque notre système est à la limite de la causalité.

2.5.2.1.2. UTILITE DE LA MATRICE RESOLVANTE

Elle peut servir comme on vient de le voir, pour calculer la fonction de transfert du système à partir de la représentation sur l'espace d'états mais également pour déterminer la matrice de transition $\phi(p)$ très utile pour le calcul des réponses temporelles.

2.5.3.LES POLES ET ZEROS A PARTIR DE LA FONCTION DE TRANSFERT

2.5.3.1. RESOLUTION D'UN POLYNOME SUR LE PLAN COMPLEXE

Il est tout à fait évident qu'une résolution analytique pure est à exclure. Seules les méthodes numériques sont à même de détecter les racines de l'équation avec une marge d'erreur qui dépend de l'efficacité de la méthode. Nous donnerons ci-dessous un aperçu sur quelques méthodes de résolution.

2.5.3.1.1. METHODE DU BALAYAGE

Elle consiste à transformer notre polynôme P(p) en deux équations non linéaires à résoudre :

$$\begin{cases} \sum_{i=0}^{N} a_i M^i . \cos(i.\theta) = 0 & (1) \\ \sum_{i=0}^{N} a_i M^i . \sin(i.\theta) = 0 & (2) \end{cases} \tag{2.23}$$

En balayant la phase et le module avec un pas fin, on vérifie les équations (1) et (2) de (2.23). Et si :

$$(1) \in [-e \quad +e] \quad et \quad (2) \in [-e \quad +e]$$

Alors, on peut affirmer que le point en équation est une racine du polynôme avec une erreur plus ou moins faible.

- **Inconvénient**

Nous savons que cette méthode n'a que des inconvénients puisque :

- Plus on diminue l'intervalle, plus les chances de trouver une racine diminuent.
- De plus, le temps d'exécution de la méthode est très lent.
- Le domaine de balayage est limité.
- Plus le pas ΔM augmente, plus la surface contenant l'éventuel zéro du polynôme s'élargit et plus les chances de détection des racines diminuent.

2.5.3.1.2. METHODE DE BERNOULLI

C'est une méthode itérative qui consiste à transformer notre équation polynomiale en une équation récurrente. En fixant des conditions initiales appropriées on remarquera qu'au bout d'un certain nombre d'itérations, il y a convergence vers le zéro.

- **Inconvénient**

- La détection des racines complexes est impossible.
- La multiplicité engendre parfois des divergences.

2.5.3.1.3. METHODE DE BAIRSTOW

C'est une méthode qui consiste à décomposer notre polynôme en un produit de polynôme de degré 2 ou plus. La résolution des polynômes du second ordre ne posant aucun problème, les racines sont déduites automatiquement. Cette méthode nous amène à résoudre un système d'équations non linéaires nécessitant des méthodes de résolution itératives assez complexes qui

auront un impact direct sur le temps de calcul sachant que les résultats escomptés ne seront pas forcements justes.

2.5.3.1.4. METHODE DE LAGUERRE

a) DESCRIPTION

C'est une méthode itérative de convergence qui permet de détecter un zéro qu'il soit complexe ou non et quel que soit sa multiplicité (du moins théoriquement). Car en pratique, on a constaté que pour une multiplicité d'ordre 4, il commençait à se produite un écartement par rapport à la valeur exacte. Même si ce dernier reste toutefois faible. Dès qu'une racine est détectée, on procèdera à une division polynomiale d'ordre 1 si la racine est réelle, ou d'ordre 2 si elle est complexe. Puis on réapplique la méthode sur le nouveau polynôme issu de la division polynomiale euclidienne jusqu'à tarissement de toutes les racines du polynôme initial.

b) AVANTAGES

- RAPIDITE

On a constaté qu'il fallait en moyenne 5 itérations pour la détection d'une racine. Ce nombre dépend toutefois de la marge d'erreurs tolérées.

Lorsque la racine est éloignée de la condition initiale, le nombre d'itérations augmente et des risques de divergence apparaissent.

Dans notre cas, on fixe un nombre maximal de 300 itérations permettant de balayer un plan complexe assez large.

- PRECISION

La précision est inespérée, elle va jusqu'à 6 chiffres après la virgule. Si la racine est multiple, la précision diminue. Dans tous les cas, elle demeure satisfaisante.

c) CONDITION INITIALE

Il faut la poser comme un complexe ex : $X_0 = -5 + 5i$ c'est-à-dire dans une zone où la susceptibilité de détecter les pôles est grande de façon à réduire le nombre d'itérations.

d) CALCULS

Soit le polynôme : $$P_n(p) = \sum_{i=0}^{n} a_i . P^i$$

Il peut s'écrit sous la forme : $$P_n(p) = \prod_{i=1}^{n} (p - p_i)$$

Où les p_i sont les racines du polynôme.

Puisque le polynôme est à coefficients réels, les zéros complexes possèdent leur conjugués qui sont également des zéros du polynôme, ce qui n'aurait pas été le cas si les coefficients a_i étaient complexes.

- Considérons la dérivée première :

$$L(p) = \frac{d}{dp}\ln\left|P_n(p)\right| = \frac{P_n'}{P_n} \tag{2.24}$$

On peut écrire aussi :

$$\ln\left|P_n(p)\right| = \ln\left|p-p_1\right| + \ln\left|p-p_2\right| + ... + \ln\left|p-p_n\right|$$

$$\frac{d}{dp}\ln\left|P_n(p)\right| = \frac{1}{p-p_1} + \frac{1}{p-p_2} + ... + \frac{1}{p-p_n}$$

- De la même façon, on peut calculer la dérivée seconde. D'autre part, soit :

$$H(p) = -\frac{d^2}{dp^2}\left(\ln\left|P_n(p)\right|\right) = \left(\frac{P_n'(p)}{P_n(p)}\right)^2 - \left(\frac{P_n''(p)}{P_n(p)}\right) \tag{2.25}$$

Laguerre fait la superposition suivante :

« Il suppose que la racine p_i est distante de 'a' (a pouvant être complexe) par rapport à notre position courante p et que toutes les autre racines seront distante de 'b' par rapport à notre position courante p ».

$$\begin{cases} a = p - p_i & \forall\ \mathrm{i} = 1,...,\mathrm{n} \\ b = p - p_j & \forall\ \mathrm{j} = 1,...,\mathrm{n}\ \text{et}\ \mathrm{j} \neq \mathrm{i} \end{cases}$$

De cette supposition du mathématicien Laguerre, on peut expliciter les relations (2.24) et (2.25) jusqu'à obtenir une formule générale sur 'a' et 'b' qui servira de référence.

$$L = \left(\frac{1}{p-p_1}\right) + \left(\frac{1}{p-p_2} + \frac{1}{p-p_3} + ... + \frac{1}{p-p_n}\right)$$

$$L = \frac{1}{a} + \frac{n-1}{b} \tag{2.26}$$

$$H = \frac{1}{a^2} + \frac{n-1}{b^2} \tag{2.27}$$

$$(2.26) \Rightarrow \frac{n-1}{b} = L - \frac{1}{a} = \frac{L.a-1}{a} \Rightarrow b = \frac{(n-1).a}{L.a-1} \tag{2.28}$$

En combinant (2.27) et (2.28) :

$$n\left(\frac{1}{a}\right)^2 - 2L\left(\frac{1}{a}\right) + \left(L^2 - (n-1)H\right) = 0 \tag{2.29}$$

Équation du second ordre à résoudre avec pour variable $\frac{1}{a}$.

$$\Delta' = L^2 - n.L^2 + n.(n-1).H = (n-1)(n.H - L^2)$$

D'où $$a^{-1} = \frac{L \pm \sqrt{\Delta'}}{n} \quad \Rightarrow \quad a = \frac{n}{L \pm \sqrt{\Delta'}}$$

D'où l'équation de Laguerre $$a = \frac{n}{L \pm \sqrt{(n-1).(n.H - L^2)}} \tag{2.30}$$

Avec $$\begin{cases} L = \dfrac{P_n'(p)}{P_n(p)} \\ H = L^2 - \dfrac{P_n''(p)}{P_n(p)} \end{cases}$$

Puisque dans notre cas :

$$P_n(p) = \sum_{i=0}^{n} a_i.p^i$$

On aura $$P_n'(p) = \sum_{i=1}^{n} a_i.i.P^{i-1}$$ De même $$P_n''(p) = \sum_{i=2}^{n} a_i.i.(i-1)P^{i-2}$$

Formulation numérique :

- Le calcul de P_n , $P_n^{'}$ et $P_n^{''}$ est facile puisqu'il ne dépend que des coefficients.
- Les calculs de L(p), H(p) et de a nécessitent des opérations sur le plan complexe. Par conséquent, il est impératif de prévoir une bibliothèque mathématique englobant les opérations de base sur le plan complexe.

e) PROBLEME DE LA CONDITION INITIALE

Pour démarrer la procédure de Laguerre, il est indispensable de choisir une condition initiale appropriée afin que la procédure de détection converge, ce qui n'est pas toujours le cas. Il est difficile de prévoir la condition initiale optimale à injecter en l'absence d'informations sur les racines de l'équation. Pour diminuer les risques de divergence, la procédure de Laguerre sera épaulée d'une procédure de calcul analytique des racines de l'équation, limitée au premier et second ordre. De plus, dans le cas du tracé du Lieu des Racines, on peut exploiter les racines précédentes afin de déterminer les racines en cours sachant qu'elles sont proches les unes des autres. Cette proximité est due à l'utilisation d'un pas adaptatif qui annule le risque d'une variation trop rapide du lieu et assure un tracé régulier. On introduira la notion de vecteur conditions initiales, qui associera à chaque branche sa condition initiale à ajouter. Le tracé, dans ce cas, s'effectue sans problème majeur, pourvu que les premières racines détectées ne soient pas issues d'une divergence.

f) CONDITION DE CONVERGENCE

Dans notre cas, la condition de convergence se fera sur P(p), on supposera que si :

$|P(p)| \in [-10^{-3},\ 10^{-3}]$, alors la racine trouvée est satisfaisante. Il faudra répondre à l'une des conditions suivantes :

- Ou bien, $|P(p)| \in [-10^{-3},\ 10^{-3}]$
- Ou bien, le nombre d'itérations maximal, que nous avons fixé à 300, a été consommé.

g) DIVISION POLYNOMIALE

Lorsqu'une racine est détectée, on procède à une division polynomiale :

- D'ordre 1, si la racine est réelle (partie imaginaire comprise dans une marge de 10^{-3} centrée sur l'axe des réels).
- D'ordre 2, si la racine est complexe.

Une fois la division effectuée, on réapplique la procédure de détection de la racine suivante jusqu'à épuisement de toutes les racines de notre polynôme initial.

- ***Division polynomiale d'ordre 1***

On a : $P_n(p) = \sum_{i=0}^{n} a_i.P^i$ soit p_1 la racine réelle détectée $P_n(p) = (p - p_1)Q_{n-1}(p)$ le reste étant nul puisque p_1 est la racine de $P_n(p)$.

$$\sum_{i=0}^{n} a_i.P^i = (p - p_1)\sum_{i=0}^{n-1} q_i.P^i \Leftrightarrow \sum_{i=0}^{n} a_i.P^i = \sum_{i=0}^{n-1} q_i.P^{i+1} - p_1\sum_{i=0}^{n-1} q_i.P^i$$

$$\Leftrightarrow \sum_{i=0}^{n} a_i.P^i = q_{n-1}p^n + \sum_{i=1}^{n-1}(q_{i-1} - p_1q_i)p^i - p_1q_0 \qquad (2.31)$$

On a un système de n+1 équations à n inconnues : $q_0, q_1, ...$et q_{n-1}

$$\text{Or } \begin{cases} a_n = q_{n-1} \\ \vdots \\ a_{i+1} = q_i - p_1q_{i+1} \\ \vdots \\ a_0 = -p_1q_0 \end{cases} \Rightarrow \begin{cases} q_{n-1} = a_n \\ \vdots \\ q_i = a_i + p_1q_{i+1} \qquad i = 0,...,n-1 \\ \vdots \end{cases}$$

On a deux façons de résoudre le problème :

1) A rebour

Condition initiale : $q_{n-1} = a_n$

Equation d'itérations : $q_i = a_{i+1} + p_1q_{i+1}$ i de (n - 2) à 0

La dernière équation est implicitement vérifiée puisque le reste doit être nul alors l'égalité doit être vérifiée.

2) A incrémentation

Soit p_1 notre racine réelle,

Condition initiale : $q_0 = \dfrac{-a_0}{p_1}$ si $p_1 = 0 \Rightarrow q_0$ ne peut exister donc : problème !

Equation d'itérations $q_i = \dfrac{q_{i-1} - a_i}{p_1}$ i de 1 à (n - 1)

Bien évidement, on optera pour la méthode à rebour puisque elle évite les cas particuliers où $p_i = 0$ et donc les blocages.

- ***Division polynomiale d'ordre 2***

Soit p_1 notre racine complexe donc p_1' son conjoint est également une racine de $P_n(p)$.

On a :

$$\begin{aligned}(p-p_1)(p-p_1') &= p^2 - (p_1 + p_1')p + p_1.p_1' \\ &= p^2 - 2.\mathrm{Re}(p_1).p + (\mathrm{Re}(p_1))^2 + (\mathrm{Im}(p_1))^2 \\ &= p^2 + A_1 p + A_2\end{aligned}$$

$$\Leftrightarrow P_n(p) = (p^2 + A_1 p + A_2).Q_{n-2}(p)$$

$$\Leftrightarrow P_n(p) = p^2.Q_{n-2}(p) + A_1 p.Q_{n-2}(p) + A_2.Q_{n-2}(p)$$

$$\Leftrightarrow \sum_{i=0}^{n} a_i.P^i = q_{n-2}p^n + (q_{n-3} + A_1 q_{n-2})p^{n-1} + \sum_{i=2}^{n-2}(q_{i-2} + A_1 q_{i-1} + A_2 q_i)p^i + (A_1 q_0 + A_2 q_1)p + A_2 q_0$$

De même, on aboutit à un système de n+1 équations à n-1 inconnues ($q_0, q_1, ...et\ q_{n-2}$). Nous procéderons par des itérations à rebour avec les équations suivantes :

$$\begin{cases} q_{n-2} = a_n & (1) \\ q_{n-3} = a_{n-1} - A_1 q_{n-2} & (2) \\ \vdots & \\ q_i = a_{i+2} - A_1.q_{i+1} - A_2.q_{i+2} \quad \text{i = n - 4 à 0} & (3) \end{cases} \qquad (2.32)$$

(1) et (2) de (2.32) forment les équations de conditions initiales. (3) représente l'équation des itérations.

h) CONCLUSION

La méthode de Laguerre est rapide et précise, les cas de divergence existant mais demeurent toutefois assez rares. Par conséquent, cette méthode convient largement pour le tracé du Lieu de Racines.

2.6. MENU « FONCTION DE TRANSFERT »

Le menu Fonction de transfert est l'une des trois possibilités permettant de définir notre système afin de le rendre prêt à subir les différentes techniques d'analyse.

L'utilisateur est appelé à formuler sa fonction de transfert sous forme textuelle à partir de laquelle seront extraits les coefficients du numérateur et du dénominateur, nécessaires pour une analyse correcte du système.

Il existe dans ce menu 06 procédures qui servent à traiter la fonction de transfert :

2.6.1. LA PROCEDURE ErreurEnEcriture ( )

2.6.1.1. LES FONCTIONS UTILISEES

- Fonction MessageDeErreur ( )
- Fonction PositionDeErreur ( )
- Fonction ErreurEnFT ( )

2.6.1.2. DESCRIPTION DE LA PROCEDURE

1ere **Phase :** *détection des erreurs textuelles au cours d'écriture*

Cette phase à pour objectif de détecter les erreurs textuelles au cours de l'écriture du numérateur ou du dénominateur.

1) Détection des caractères illégaux :

Les caractères connus par l'interpréteur sont {0,1,2,3,4,5,6,7,8,9, . , P, ^,+,-,*, ' ',} tout autre caractère validé sera interprété comme une erreur. Le message d'erreur qui correspond par exemple au dénominateur s'affiche suivant la langue choisie :

En Français	" Dénominateur	Erreur : Caractère illégal 'A' "
En anglais	" Denominator	Error : illegal caracter 'A' "

2) Détection d'une suite de caractère :

Si les caractères se répètent textuellement deux fois de façon successive sauf les numéros, cela sera interprété comme une erreur qui s'affiche comme suit :

En Français	" Numérateur	Erreur : seccession illégale '^^' "
En anglais	" Numerator	Error : illegal seccession '^^' "

3) Détection d'une erreur sur la position du symbole puissance '^' :

Sachant que le caractère puissance '^' ne peut se situer qu'après la variable, tout autre positionnement sera notifie comme une erreur. Le message suivant apparaîtra :

En Français	" Dénominateur	Erreur : Position illégal '5^' "
En anglais	" Denominator	Error : illegal Position '5^' "

4) Détection d'une erreur sur la position du caractère '**.**':

Un positionnement de '**.**' ne peut se faire qu'entre deux chiffres car il est censé séparer la partie entière de la partie décimale d'un nombre réel. Dans le cas contraire, on détecte une erreur précisée par le message suivant :

En Français	" Dénominateur	Erreur : Position illégal 'P.' "
En anglais	" Denominator	Error : illegal Position 'P.' "

5) Détection d'une omission du symbole puissance '^':

Textuellement, après le caractère variable ne peut demeurer que l'un des caractères suivants : {'+','-','^'}. Toute autre caractère signifiera que le manipulateur a omis le symbole '^'. Un message d'erreur apparaît comme suit:

En Français	" Dénominateur	Erreur : Position illégal 'P3' "
En anglais	" Denominator	Error : illegal Position 'P3' "

Après avoir bien terminé l'écriture textuelle de la fonction de transfert, on peut appuyer sur le bouton « Appliquer » qui va engendrer la confirmation de la fonction de transfert et une exécution de la 2ème phase et (05) procédures successivement :

2ème Phase: *Détection des erreurs en appuyant sur le bouton 'Appliquer':*

Le but de cette phase est de détecter trois types d'erreurs :

1) Détection d'une erreur de validation:

C'est-à-dire que le numérateur ou le dénominateur ne contiennent aucun caractère. Un message d'erreur apparaît comme suit:

En Français	" Numérateur	Erreur : Le numérateur n'est pas rempli "
En anglais	" Numerator	Error : Numerator not Assigned"

Et dans la FormErreur

En Français	" Corriger l'erreur affichée dans la barre d'erreur"
En anglais	" Correct the error posted in the error bar"

2) Détection d'une erreur non corrigée :

Si nous cliquons sur le bouton « Appliquer » et la barre d'erreurs affiche une erreur ; un message d'erreur s'affiche dans une petite fenêtre « FormErreur » comme suit :

En Français	" Corriger l'erreur affichée dans la barre d'erreur"
En anglais	" Correct the error posted in the error bar"

3) Détection d'une erreur de causalité :

Dans le cas où la condition de Causalité n'est pas vérifié (C'est-à-dire l'ordre du numérateur est supérieur au dénominateur) un message d'erreur s'affichera comme suit :

En Français	" Causalité	Erreur : La causalité n'est pas vérifiée"
En anglais	" Caution	Error : Causality not verified "

Et

En Français	" Corriger l'erreur affichée dans la barre d'erreur"
En anglais	" Correct the error posted in the error bar"

N.B) L'ordre de numérateur et dénominateur est calculé par la fonction *NbrRacines ( )*.

2.6.1.3. STRUCTURE FONCTIONNELLE

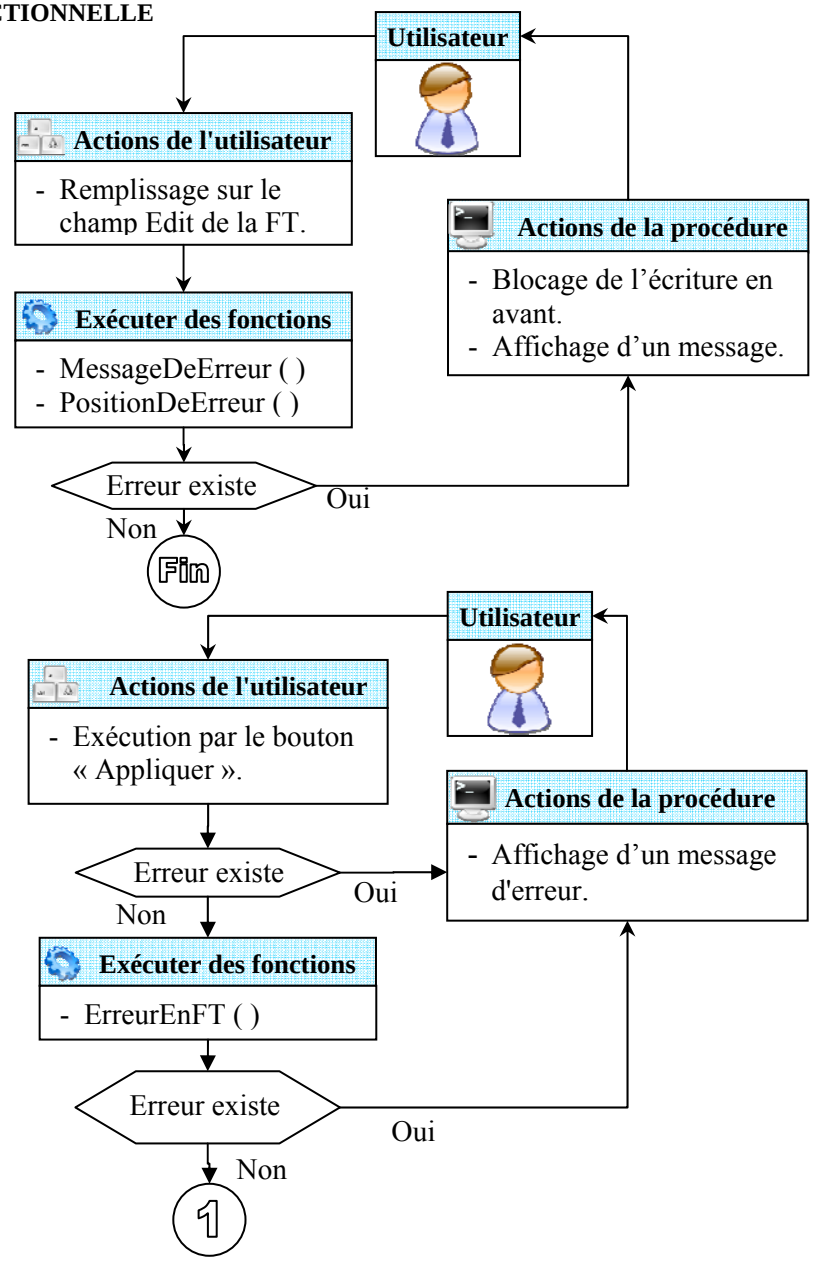


Figure 2.3 : schémas fonctionnels qui interprètent la procédure *ErreurEnEcriture ( )*

2.6.2. LA PROCEDURE ConfirmationFT ( )

2.6.2.1. DESCRIPTION DE LA PROCEDURE

Cette fonction sert à enregistrer la fonction de transfert et d'afficher le message « FONCTION DE TRANSFERT » dans le champ Confirmation (cellule de Confirmation).

L'objectif de cet affichage est de montrer à l'utilisateur que le système est défini par la fonction de transfert.

2.6.2.2. STRUCTURE FONCTIONNELLE

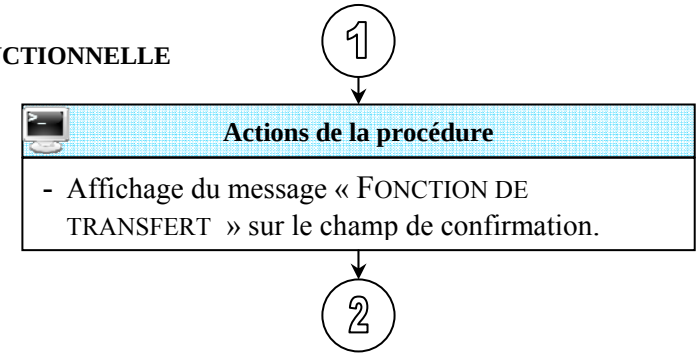


Figure 2.4 : schéma fonctionnel interprétant la procédure *ConfirmationFT ( )*

2.6.3. LA PROCEDURE SegmentationFT ( )

2.6.3.1. LES FONCTIONS UTILISEES

- Fonction Segmentation ( )

2.6.3.2. DESCRIPTION DE LA PROCEDURE

La méthode d'analyse utilisée est basée sur le concept de segmentation de l'expression mathématique. On illustre le concept de segmentation par l'exemple suivant :

LA FONCTION DE TRANSFERT	$G(p)=\frac{p+1}{p\text{^}5+3*p-4}$						
Dénominateur	P^5+3*P-4						
Vecteur des termes	P^5		+3*P		-4		Etape 1
Vecteur premier des coefficients	1		+3		-4		Etape 2
Vecteur des facteurs	5		1		0		
Vecteur des coefficients	1	0	0	0	3	-4	Etape 3

Figure 2.5 : tableau qui interprète la procédure *SegmentationFT ( )*,

★ ***1ère étape :*** « Décomposition de l'expression en terme » :

Par le balayage de l'expression, du premier caractère jusqu'au dernier, on extrait les termes avec les caractères de séparation (+ ou -).

★ ***2ème étape :*** « Extraction du coefficient et du facteur du terme » :

Il existe plusieurs possibilités de positionnement du coefficient et des facteurs dans le terme, qu'on interprète comme suit :

Si (la variable n'existe pas)
- le coefficient est égal au terme.
- le facteur est égal à zéro (0)

Sinon
- le coefficient se place toujours avant la variable ou le caractère «*»

Si (le caractère '^' n'existe pas)
- le facteur est égal à '1'

Sinon
- le facteur se place toujours après le caractère '^'.

Fin

Fin

★ ***3ème étape :*** « Construire le vecteur des coefficients »

Dans cette étape on détermine l'ordre du polynôme qu'est la valeur maximum du vecteur de facteur.

On construit un vecteur de dimension égale à l'ordre du polynôme plus '1', puis en l'initialisant par la valeur '0' dont les cellules représentent les facteurs décroissants. On le remplit par les valeurs des coefficients en respectant la position des facteurs corresponds.

2.6.3.3. STRUCTURE FONCTIONNELLE

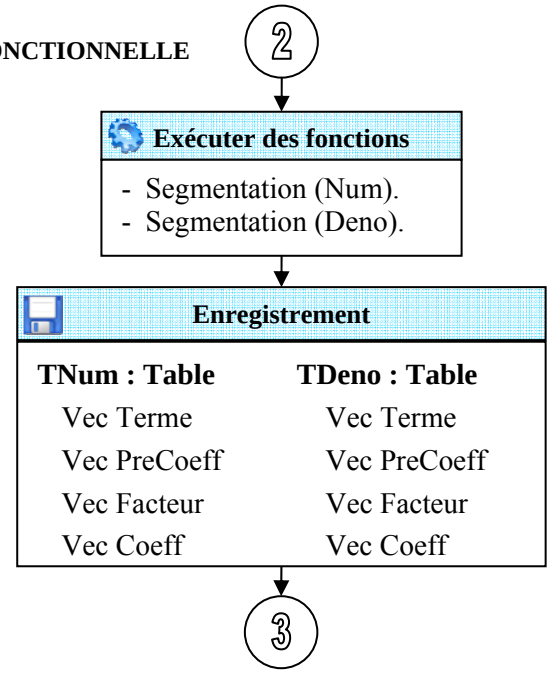


Figure 2.5 : schéma fonctionnel interprétant la procédure *SegmentationFT ( )*

2.6.4.LA PROCEDURE PassageBOauBF ( )

2.6.4.1. LES FONCTIONS UTILISEES

- Fonction SommePoly ( )
- Fonction MultScalPoly ( )

2.6.4.2. DESCRIPTION DE LA PROCEDURE

L'objectif de cette procédure est de déterminer la fonction de transfert en boucle fermée dont la formule est:

$$F(p)=\frac{K.N(p)}{D(p)+K.N(p)}$$ Avec K : le gain du système

En utilisant les deux fonctions :

- MultScalPoly ( ) : pour calculer le produit d'un polynôme par un scalaire (*K.N(p)*).
- SommePoly ( ): pour calculer la somme de deux polynômes (*D(p)+K.N(p)*)

2.6.4.3. STRUCTURE FONCTIONNELLE

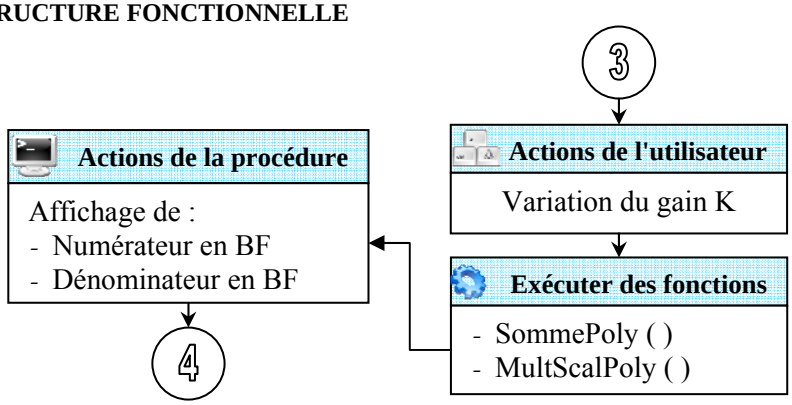


Figure 2.6 : schéma fonctionnel interprétant la procédure *PassageBOauBF ( )*

2.6.5.LA PROCEDURE PassageFTauPZ ( )

2.6.5.1. LES FONCTIONS UTILISEES

- Fonction NbrRacines ( ).
- Fonction RacinesLaguerre ( )

2.6.5.2. DESCRIPTION DE LA PROCEDURE

Cette procédure sert à résoudre les polynômes par application de la méthode numérique « Laguerre » et les résultats sont affichés dans le Menu « Pôles et Zéros »:

- Numérateur N(p)=0 ; pour obtenir les zéros,
- Dénominateur D(p)=0 ; pour obtenir les pôles.

2.6.5.3. STRUCTURE FONCTIONNELLE

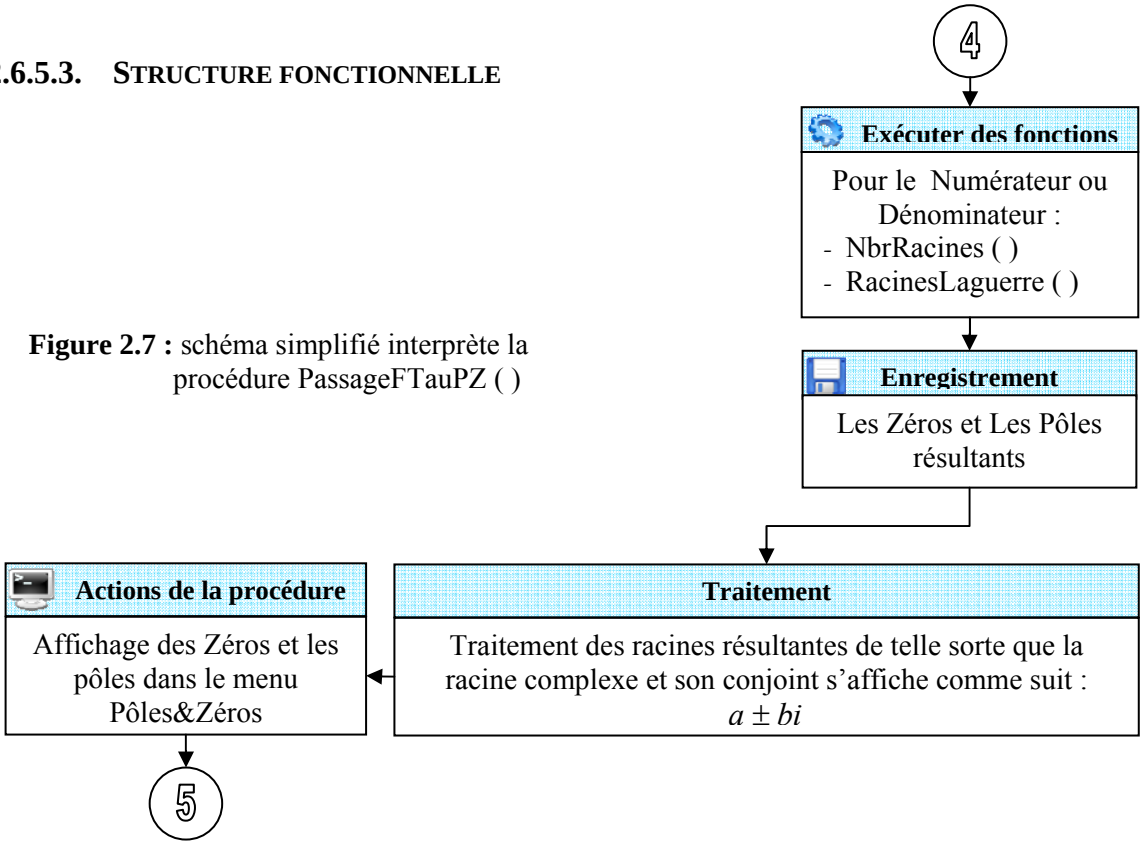


Figure 2.7 : schéma simplifié interprète la procédure PassageFTauPZ ( )

2.6.6. LA PROCEDURE PassageFTauEE ( )

2.6.6.1. LES FONCTIONS UTILISEES

- Fonction DimduMatriceA ( )
- Fonction MatriceA_du_FT ( )
- Fonction MatriceB_du_FT ( )
- Fonction MatriceC_du_FT ( )
- Fonction MatriceD_du_FT ( )

2.6.6.2. DESCRIPTION DE LA PROCEDURE

Cette procédure utilise la méthode directe où elle transforme les vecteurs de coefficients du numérateur et du dénominateur pour obtenir les matrices d'état A, B, C et D qui seront affichées dans le Menu « Espace d'état ».

2.6.6.3. STRUCTURE FONCTIONNELLE

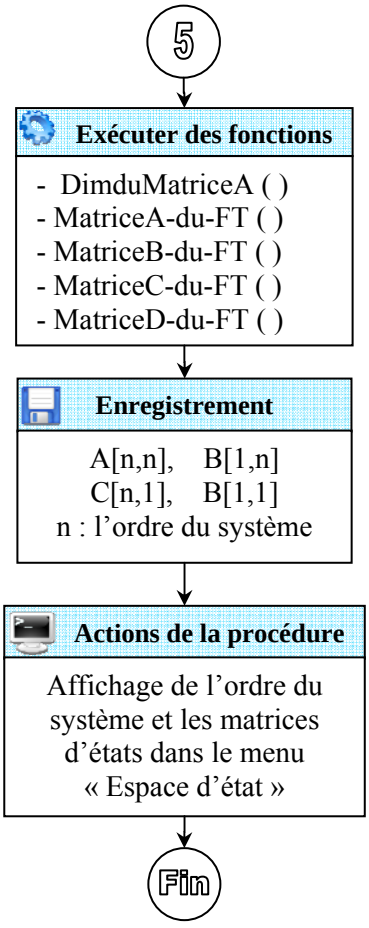


Figure 2.8 : schéma fonctionnel expliquant la procédure *PassageFTauEE ( )*

2.7. MENU « POLES & ZEROS

2.7.1. LA PROCEDURE ErreurEnEcriture ( )

2.7.1.1. LES FONCTIONS UTILISEES

- Fonction ErreurEnPZ ( )

2.7.1.2. DESCRIPTION DE LA PROCEDURE

L'exécution de cette procédure se fait en deux phases :

★ ***1ére Phase :*** *Détection d'une erreur de validation*

Si les cellules des pôles et/ou les Zéros ne contiennent aucun caractère ", cela est alors considéré comme une erreur et un message d'erreur sera affiché :

En Français	" Les Zéros	Erreur : La cellule N° 2 n'est pas remplie "
En anglais	" Zeros	Error : The cellul N°2 not Assigned"

★ ***2éme Phase :*** *Détection d'une erreur de Causalité*

Si la causalité n'est pas vérifiée, cela est considéré comme une erreur, et alors un message d'erreur est affiché dans la barre d'erreur :

En Français	" Causalité	Erreur : La causalité n'est pas vérifier (3Zéros/2Poles) "
En anglais	" Causality	Error : La causality not verified (3Zeros/2Poles)

Remarque :

Pour saisir les racines complexes sous forme $(a \pm bi)$ il suffit de les écrire sous forme $(a+bi)$ ou $(a-bi)$; ces dernières se transforment automatiquement lorsqu'on change la cellule.

2.7.1.3. STRUCTURE FONCTIONNELLE

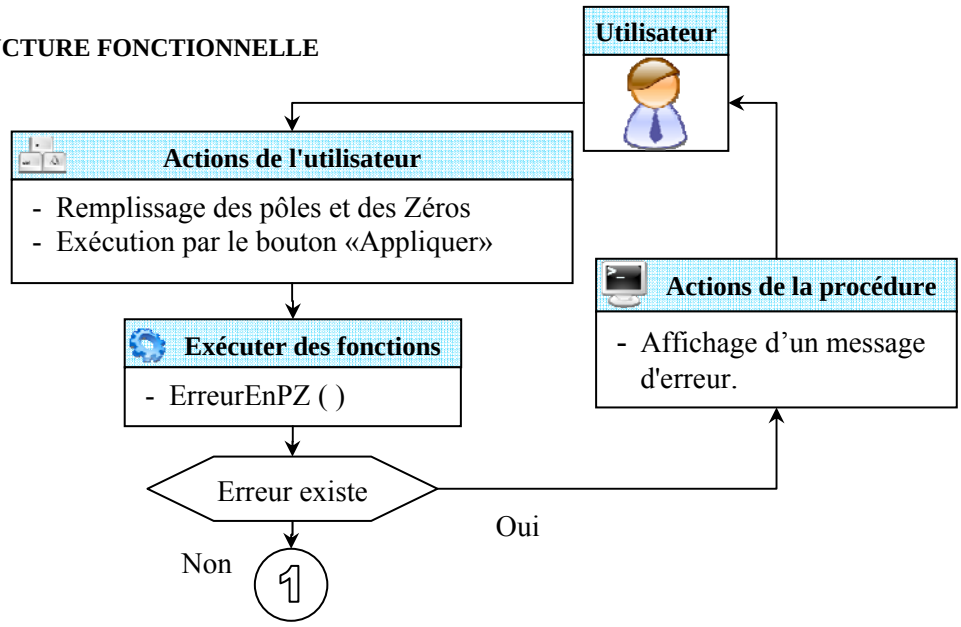


Figure 2.9 : schéma fonctionnel interprétant la procédure *ErreurEnEcriture ( )*

2.7.2.LA PROCEDURE ConfirmationPZ ( )

2.7.2.1. DESCRIPTION DE LA PROCEDURE

Le principe de cette procédure est de confirmer que le système est défini par les Pôles et les Zéros, et se traduit par l'affichage de l'expression « LES POLES ET ZEROS » dans la cellule de confirmation.

2.7.2.2. STRUCTURE FONCTIONNELLE

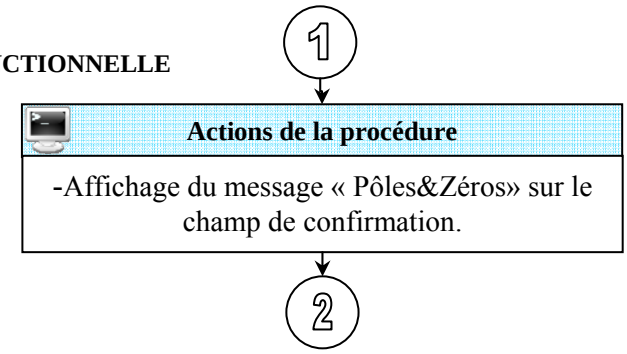


Figure 2.10 : schéma fonctionnel interprétant la procédure *ConfirmationPZ ( )*

2.7.3.LA PROCEDURE PassagePZauFT ( )

2.7.3.1. LES FONCTIONS UTILISEES

- Fonction Polynome ( )

2.7.3.2. DESCRIPTION DE LA PROCEDURE

Cette procédure sert à déterminer les polynômes numérateur et dénominateur de la FT.

En premier lieu, on utilise une procédure pour détecter et traiter le nombre et la nature des racines (Zéros et pôles séparément) qui ont été introduites par l'utilisateur.

Les racines réelles et complexes sont portées successivement dans les (Vecteur Racine Réelle) et (Vecteur Racine Complexe).

- Pour les m racines complexes $Z_j = a_j \pm b_j i$, on calcul les m polynômes $Poly_j$ par la méthode suivante :

$$Poly_j = \left(p - \left(a_j + b_j i\right)\right)\left(p - \left(a_j - b_j i\right)\right) = p^2 + \alpha_j p + \beta_j \quad \text{Où} \begin{cases} \alpha_j = -2a_j \\ \beta_j = (a_j^2 + b_j^2) \end{cases} ; j = 1...m.$$

$$\prod_{j=1}^{m} (p^2 + \alpha_j p + \beta_j) \tag{2.33}$$

La multiplication entre les polynômes résultants est faite par la fonction *Multipoly ( )* :

- Pour les n racines réelles, on fait le produit **:**

$$\prod_{i=1}^{n} (p - Z_i) \tag{2.34}$$

Le résultat obtenu est un polynôme de degré n.

En fin, le produit entre les polynômes (2.33) et (2.34) est déterminée par la fonction *MultiPoly ( )* ; la dernière équation calculée, présente le numérateur ou bien le dénominateur qui sera affiché dans le menu de la fonction de transfert.

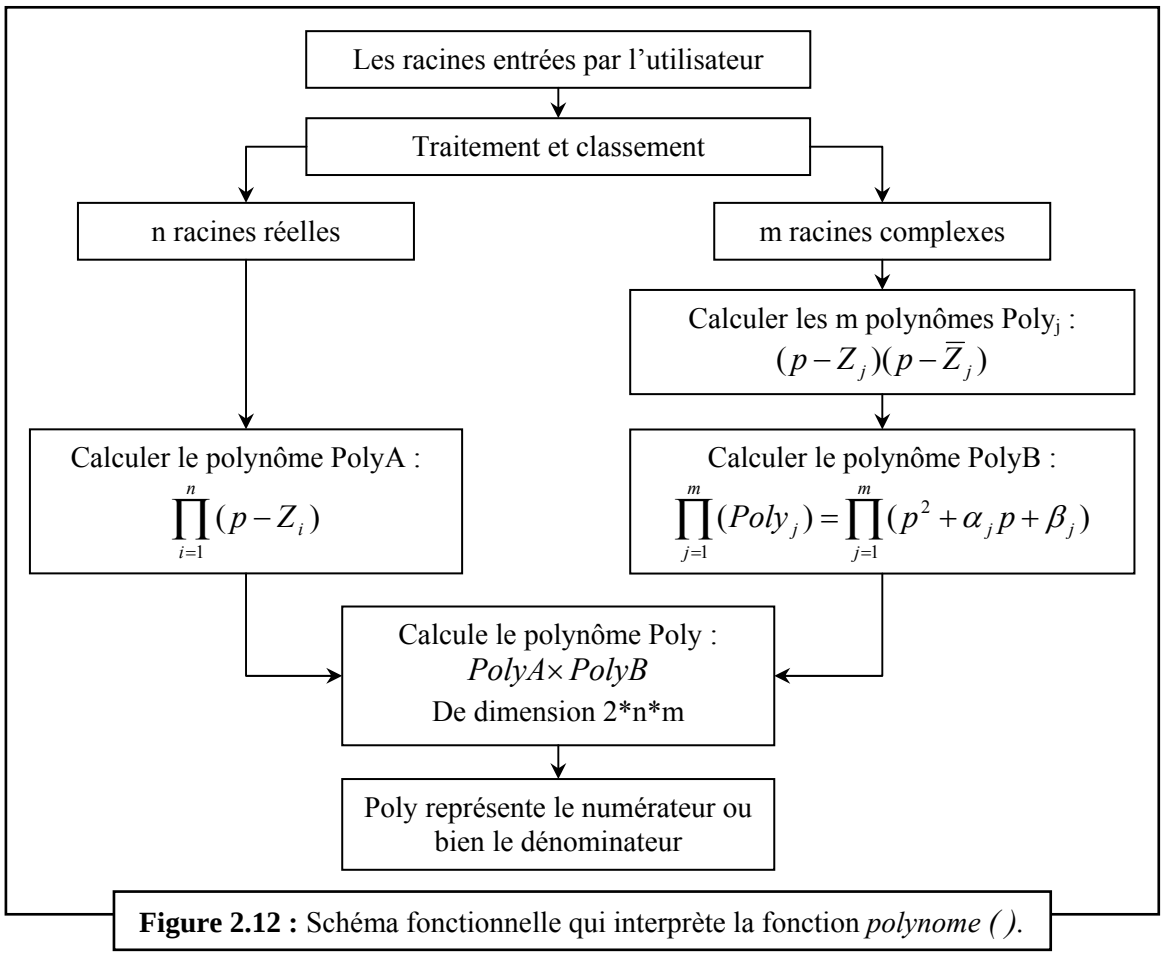


Figure 2.12 : Schéma fonctionnelle qui interprète la fonction *polynome ( ).*

2.7.3.3. STRUCTURE FONCTIONNELLE

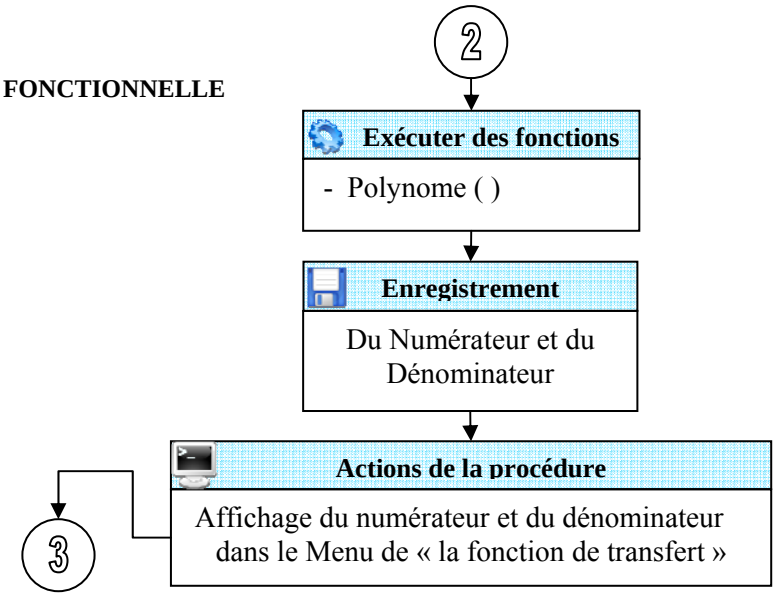


Figure 2.11 : schéma fonctionnel interprétant la procédure *PassagePZauFT ( )*

2.7.4.LA PROCEDURE PassageBOauBF ( )

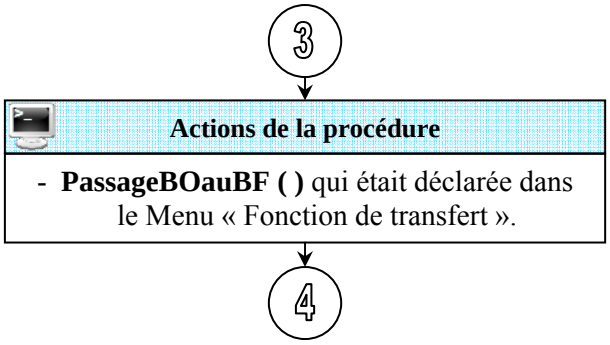


Figure 2.13 : schéma fonctionnel interprétant la procédure *PassageBOauBF ( )*

2.7.5.LA PROCEDURE passagePZauEE ( )

2.7.5.1. LES FONCTION UTILISEES

- Fonction PassagePZauFT ( )
- Fonction PassagePZauFT ( )

2.7.5.2. DESCRIPTION DE LA PROCEDURE

Pour déterminer le passage entre la représentation par les pôles et zéros et l'espace d'état on a utilisé la procédure *PassagePZauEE ( )* qui permet de transférer les données de manière indirecte de la représentation par les pôles et zéros vers l'espace d'état est cela en deux étapes :

- ⋆ **1ère étape** : de la représentation par les pôles et Zéros vers la représentation de la fonction de transfert en utilisation la fonction *PassagePZauFT ( )*
- ⋆ **2éme étape :** de la représentation par la fonction de transfert vers la représentation de l'espace d'état en utilisation la fonction *PassageFTauEE ( )*

2.7.5.3. STRUCTURE FONCTIONNELLE

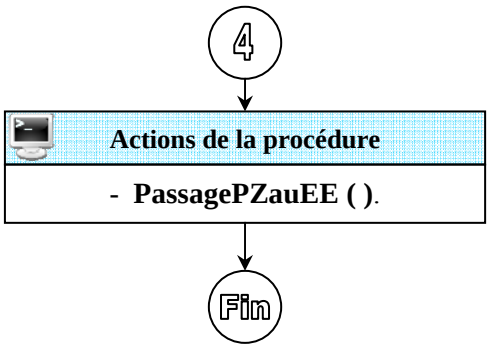


Figure 2.14 : schéma fonctionnel interprétant la procédure *PassagePZauEE ( )*

2.8. MENU « L'ESPACE D'ETAT » : Ce menu contient trois sous menus :

2.8.1. SOUS MENU « MATRICE D'ETAT »

2.8.1.1. LA PROCEDURE ErreurEnEE ( )

2.8.1.1.1. LES FONCTIONS UTILISEES

- Fonction ErreurEnEE ( )

2.8.1.1.2. DESCRIPTION DE LA PROCEDURE

Si les cellules des matrices A, B, C et D ne contiennent aucun caractère ", la procédure détecte l'anomalie et un message d'erreur est alors affiché sur la barre d'erreur. Exemple :

En Français	" La Matrice A	Erreur : La cellule [1, 3] n'est pas remplie "
En anglais	" The Matrix A	Error : The cellul [1, 3] not Assigned"

2.8.1.1.3. STRUCTURE FONCTIONNELLE

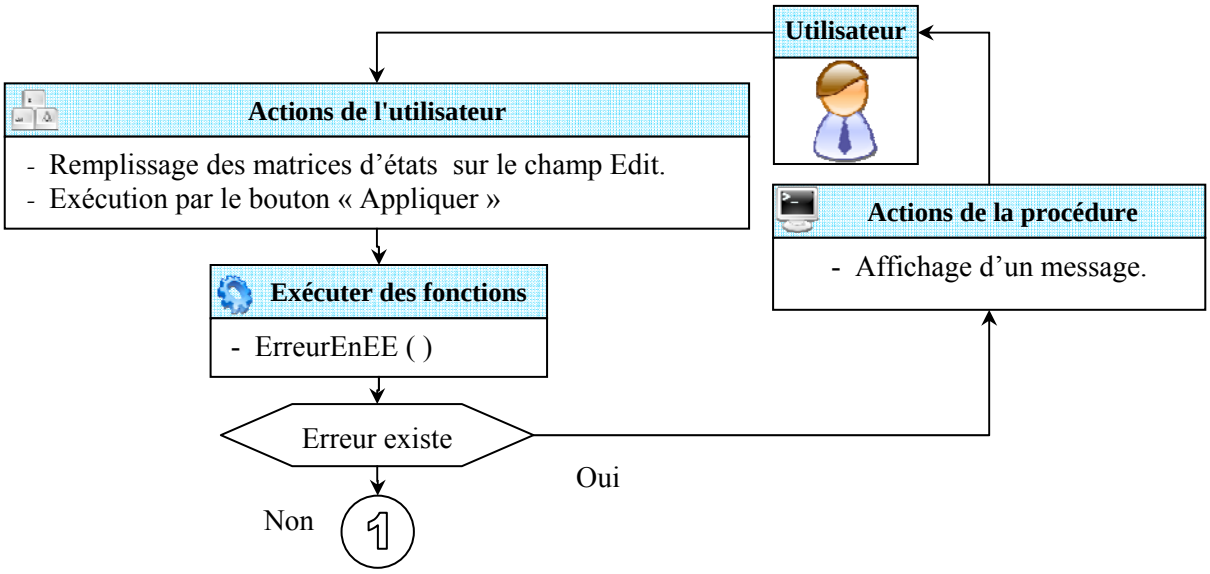


Figure 2.15 : schéma fonctionnel interprétant la procédure *ErreurEnEE ( )*

2.8.1.2. LA PROCEDURE ConfirmationEE ( )

2.8.1.2.1. DESCRIPTION DE LA PROCEDURE

Lorsque l'utilisateur choisi la représentation d'état, le message de « **ESPACE D'ETAT** » s'affichage sur le champ de confirmation.

2.8.1.2.2. STRUCTURE FONCTIONNELLE

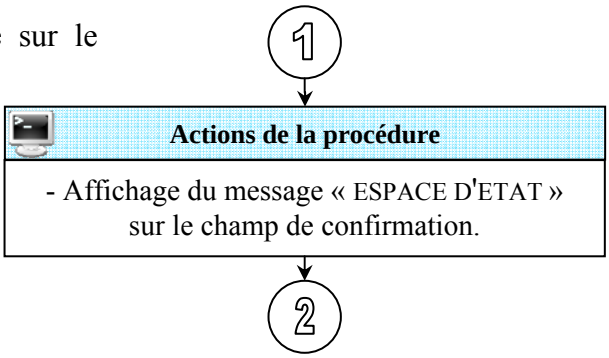


Figure 2.16 : schéma fonctionnel interprétant la procédure *ConfirmationEE ( )*

2.8.1.3. LA PROCEDURE PassageEEauFT ( )

2.8.1.3.1. LES FONCTIONS UTILISEES

- Fonction NumFT ( )
- Fonction DenoFT ( )

2.8.1.3.2. DESCRIPTION DE LA PROCEDURE

Connaissant les matrices A, B, C et D, on peut déduire la fonction de transfert du système par la méthode de « Leverrier ». Pour illustrer le mécanisme de cette dernière on présente quelques fonctions utilisées :

- La fonction *Trace ( )* ; qui calcul la trace d'une matrice,
- Les fonctions *NumDuFT ( )* et *DenoDuFT ( )*, qui servent à déterminer les polynômes Numérateur et Dénominateur de la fonction de transfert.

2.8.1.3.3. STRUCTURE FONCTIONNELLE

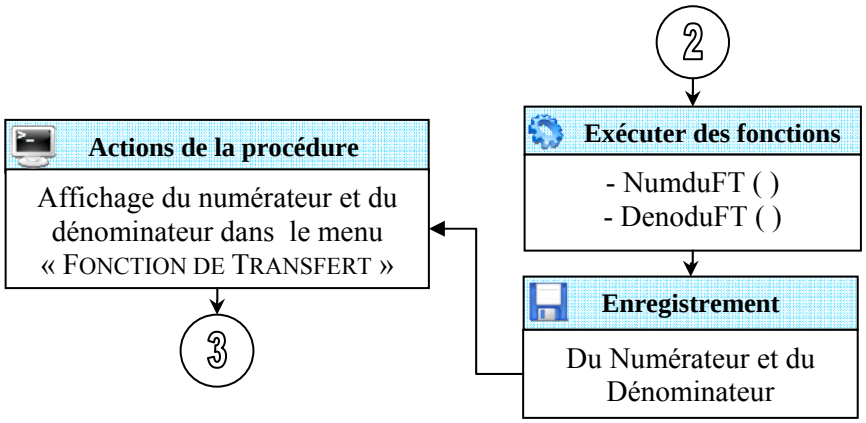


Figure 2.17 : schéma fonctionnel interprétant la procédure *PassageEEauFT ( )*

2.8.1.4. LA PROCEDURE PassageBOauBF ( )

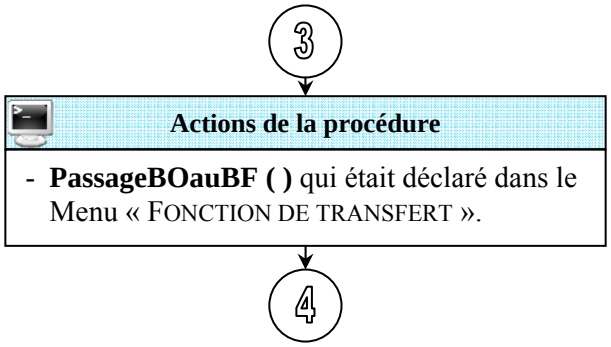


Figure 2.18 : schéma fonctionnel interprétant la procédure *PassageBOauBF ( )*

2.8.1.5. LA PROCEDURE PassageEEauPZ ( )

2.8.1.5.1. LES FONCTIONS UTILISEES

- Fonction PassageEEauFT ( )
- Fonction PassageFTauPZ ( )

2.8.1.5.2. STRUCTURE FONCTIONNELLE

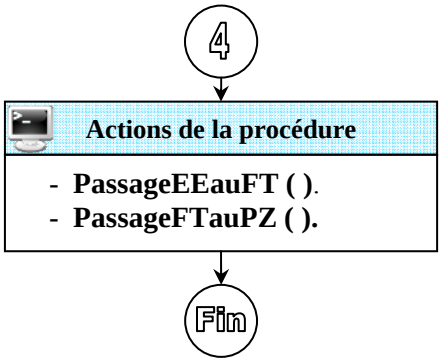


Figure 2.19 : schéma fonctionnel interprétant la procédure *PassageEEauPZ ( )*

2.8.2. SOUS MENU « OBSERVABILITE »

2.8.2.1. LA PROCEDURE Observabilite ( )

2.8.2.1.1. LES FONCTIONS UTILISEES

- Fonction CalMatObser ( )
- Fonction TestdeObser ( )
- Fonction Det ( )

2.8.2.1.2. DESCRIPTION DE LA PROCEDURE

L'intérêt de cette procédure est de juger l'observabilité du système, en déduisant la matrice d'observabilité et son déterminant à partir de la matrice A et C par applications successives des fonctions *CalMatObser ( )* et *Det ( )*.

La fonction *TestdeObser ( )* donne le résultat du test sous forme d'un message qui s'affiche comme suit :

En Français	" Le système est complètement Observable"
En anglais	" The system is completely Observable"

2.8.2.1.3. STRUCTURE FONCTIONNELLE

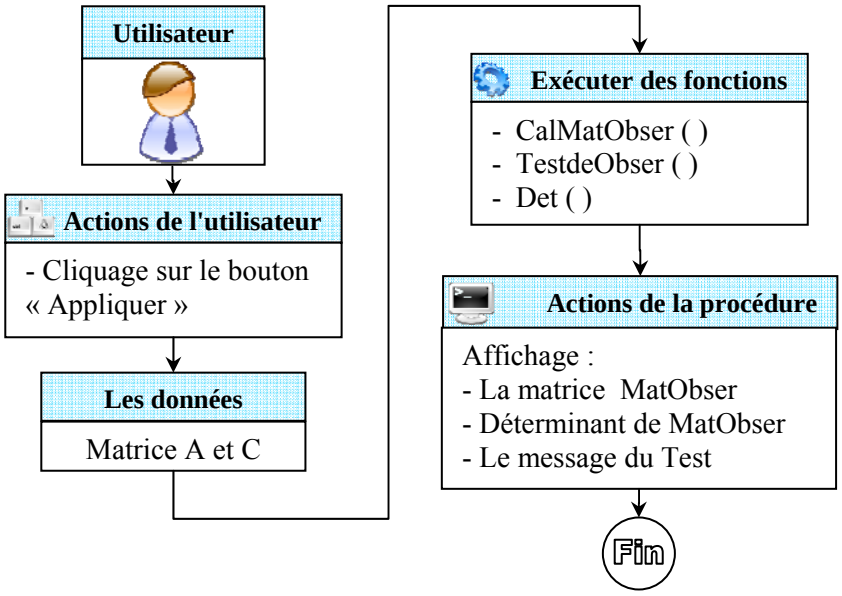


Figure 2.20 : schéma fonctionnel interprétant la procédure *Observabilite ( )*

2.8.3.SOUS MENU « CONTROLABILITE »

2.8.3.1. LA PROCEDURE Controlabilite ( )

2.8.3.1.1. LES FONCTIONS UTILISEES

- Fonction CalMatContr ( )
- Fonction Testdecontr ( )
- Fonction Det ( )

2.8.3.1.2. DESCRIPTION DE LA PROCEDURE

A partir des matrices A et B, on déduit la matrice de contrôlabilité et son déterminant par applications successives des fonctions *CalMatContr ( )*, et *Det ( )*.

La fonction *TestdeContr ( )* donne le résultat du test sous forme d'un message qui s'affiche comme suit :

En Français	" Le système est complètement Controlable "
En anglais	" The system is completely Controlable "

2.8.3.1.3. STRUCTURE FONCTIONNELLE

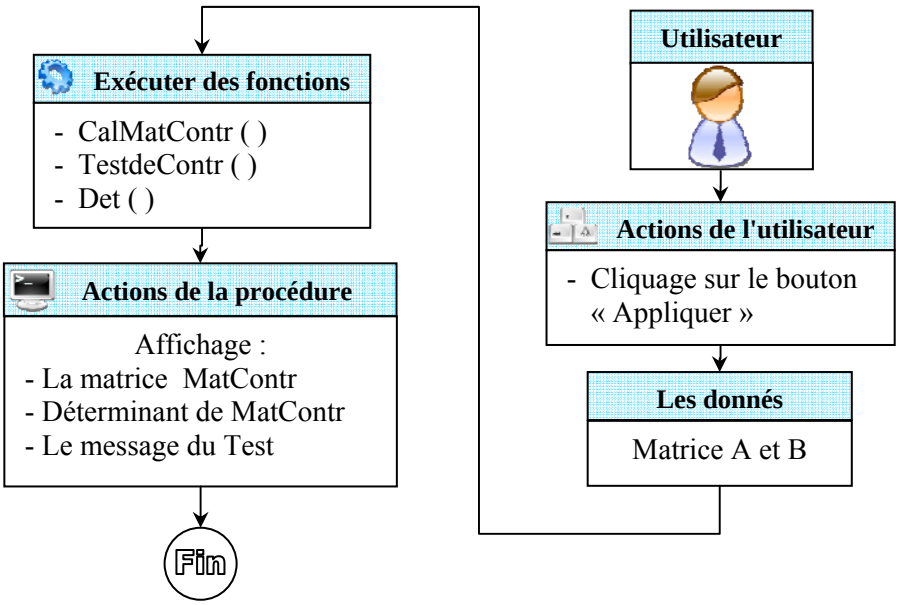


Figure 2.21 : schéma fonctionnel interprétant la procédure *Controlabilite ( )*

2.9. CONCLUSION

Ce chapitre a fait l'objet de présenter les trois représentations classiques d'un système asservis linaire à l'aide de notre logiciel. L'utilisateur peut représenter leur système de degré « n » soit par la fonction de transfert, les pôles et zéros ou bien l'espace d'état ; sachant que la liaison ou bien le passage entre ces représentations précédentes est évidente.

CHAPITRE III

ANALYSE TEMPORELLE

3.1. INTRODUCTION

L'analyse d'un système dans le domaine temporel est une des plus anciennes méthodes utilisées, mais qui été pendant un certain temps délaissée à cause de sa portée d'analyse qui reste limitée et de la difficulté à résoudre les équations différentielles d'ordre moyen (supérieur à quatre). Ces deux raisons ont poussé les automaticiens à se tourner vers d'autres méthodes telles que l'analyse dans le domaine fréquentiel, qui sont essentiellement des méthodes graphiques. Le domaine temporel se distingue également par la difficulté à extraire les informations depuis une réponse donnée et, si informations il y a, elles ne permettront pas de contourner le système et de le maîtriser. Comme exemple de caractéristiques typiques issues d'une analyse temporelle, on notera : (stabilité – précision – rapidité).

Or, ces caractéristiques à elles seules ne pourront nous renseigner sur la nature d'une réponse à une entrée donnée et donc d'envisager une prédiction du comportement du système. Toutefois, l'un des objectifs majeurs d'une analyse est bel et bien l'esprit de prédiction, de pronostic sur le comportement, témoignant ainsi de sa parfaite maîtrise. Cependant, depuis l'avènement des calculateurs numériques, l'analyse dans le domaine des temps a connu un regain d'intérêt.

En effet, le développement des techniques numériques a permis de résoudre des problèmes jusque là insolubles et d'aboutir à la construction des réponses temporelles avec une plus grande facilité.

Ceci nous a donné la possibilité d'intégrer cette méthode d'analyse dans le logiciel *"ProTahleel"* et donc d'accroître ses capacités.

Elle s'appuie simplement sur la résolution de l'équation différentielle représentant le système en utilisant la méthode de *Rung-Kutta d'ordre 4*.

Les caractéristiques du système peuvent être obtenues à partir des réponses à certaines entrées appelées communément, Entrées tests. On cite (Entrée échelon, Entrée impulsion, Entrées rampes, paraboliques et sinusoïdales).

L'entrée impulsion, étant riche en fréquence, engendre une réponse dite impulsionnelle qui renferme un très grand nombre de renseignements, notamment la stabilité du système. La réponse indicielle mue par une entrée échelon dévoile les performances dynamiques du système (temps de réponse, premier dépassement…).

3.2. SIGNAUX FONDAMENTAUX D'ENTREES (SIGNAUX DE TESTS)

Les trois signaux les plus utilisés en analyse temporelle sont l'impulsion, l'échelon et la rampe ; nous allons les passer en revue.

3.2.1. SIGNAL EN IMPULSION (DITE IMPULSION DE DIRAC)

Ce signal n'est pas une fonction au sens mathématique mais une distribution. En automatique continue, on se contentera de l'assimiler à une fonction (au grand dame des mathématiciens pour lesquels, il s'agit d'une absurdité) ; en voici alors la « définition » admise :
On appelle impulsion de Dirac, la limite d'une famille de fonctions f_{t0} telles que :

$$\begin{cases} x(t) = \dfrac{1}{t_0} & \text{pour } 0 \le t \le t_0 \\ x(t) = 0 & \text{pour } t > t_0 \text{ et } t < 0 \end{cases}$$

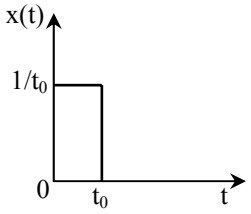


Figure 3.1 : Signal en impulsion

On note : impulsion de Dirac unitaire : $\delta(t) = \lim_{t_0 \to 0} x(t)$

On vérifie la propriété suivante : $\int_{-\infty}^{+\infty} \delta(t)dt = 1$

Un calcul montre que la transformée de Laplace de l'impulsion est :

$$L(\delta(t)) = 1 \qquad (3.1)$$

On a alors une autre définition d'une FT d'un système : La FT d'un système est la transformée de Laplace de sa réponse impulsionnelle.
En effet, $Y(p) = G(p) \times 1$ implique $y(t) = g(t)$
Où $g(t)$ est la transformée inverse de Laplace de $G(p)$.

3.2.2. SIGNAL EN ECHELON

La fonction échelon est définie de la manière suivante :

$$\begin{cases} x(t) = a & \text{pour } t \ge 0 \\ x(t) = 0 & \text{pour } t < 0 \end{cases}$$

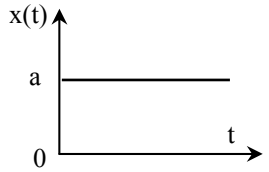


Figure 3.2 : Signal en échelon

Si a=1, L'échelon est dit unitaire.
L'intégrale de l'impulsion de Dirac est la fonction échelon unitaire.
La transformée de Laplace de la fonction échelon est :

$$L(x(t)) = X(p) = \int_0^{\infty} a.\exp(-pt)dt = \frac{a}{p} \qquad (3.2)$$

3.2.3. SIGNAL EN RAMPE

La rampe est définie de la manière suivante :

$$\begin{cases} x(t) = bt & \text{pour } t \ge 0 \\ x(t) = 0 & \text{pour } t < 0 \end{cases}$$

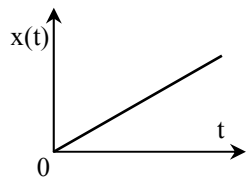


Figure 3.3 : Signal en rampe

Cette fonction est l'intégrale de la fonction échelon
La transformée de Laplace de la rampe est :

$$L(x(t)) = \frac{b}{p^2} \qquad (3.3)$$

3.2.4.SIGNAL EN PARABOLE

La rampe est définie de la manière suivante :

$$\begin{cases} x(t) = ct^2 & \text{pour } t \geq 0 \\ x(t) = 0 & \text{pour } t < 0 \end{cases}$$

Cette fonction est l'intégrale de la fonction rampe

La transformée de Laplace de la parabole est :

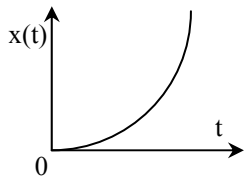


Figure 3.4 : Signal en parabole

$$L(x(t)) = \frac{c}{p^3} \tag{3.4}$$

3.3. RESOLUTION NUMERIQUE DE L'EQUATION D'ETAT PAR LA METHODE DE RUNG-KUTTA

La méthode de Runge-Kutta d'ordre 4 consiste à évaluer quatre paramètres dynamiques à chaque instant t. La différence entre les points X(t+1) et X(t) sera une somme pondérée des 04 paramètres dynamiques ainsi calculés. Les paramètres dynamiques sont estimés de la manière suivante :

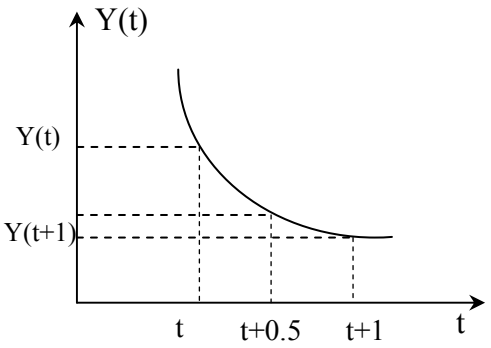


Figure 3.4 : graphe qui interprète la méthode de Rung-Kutta 4

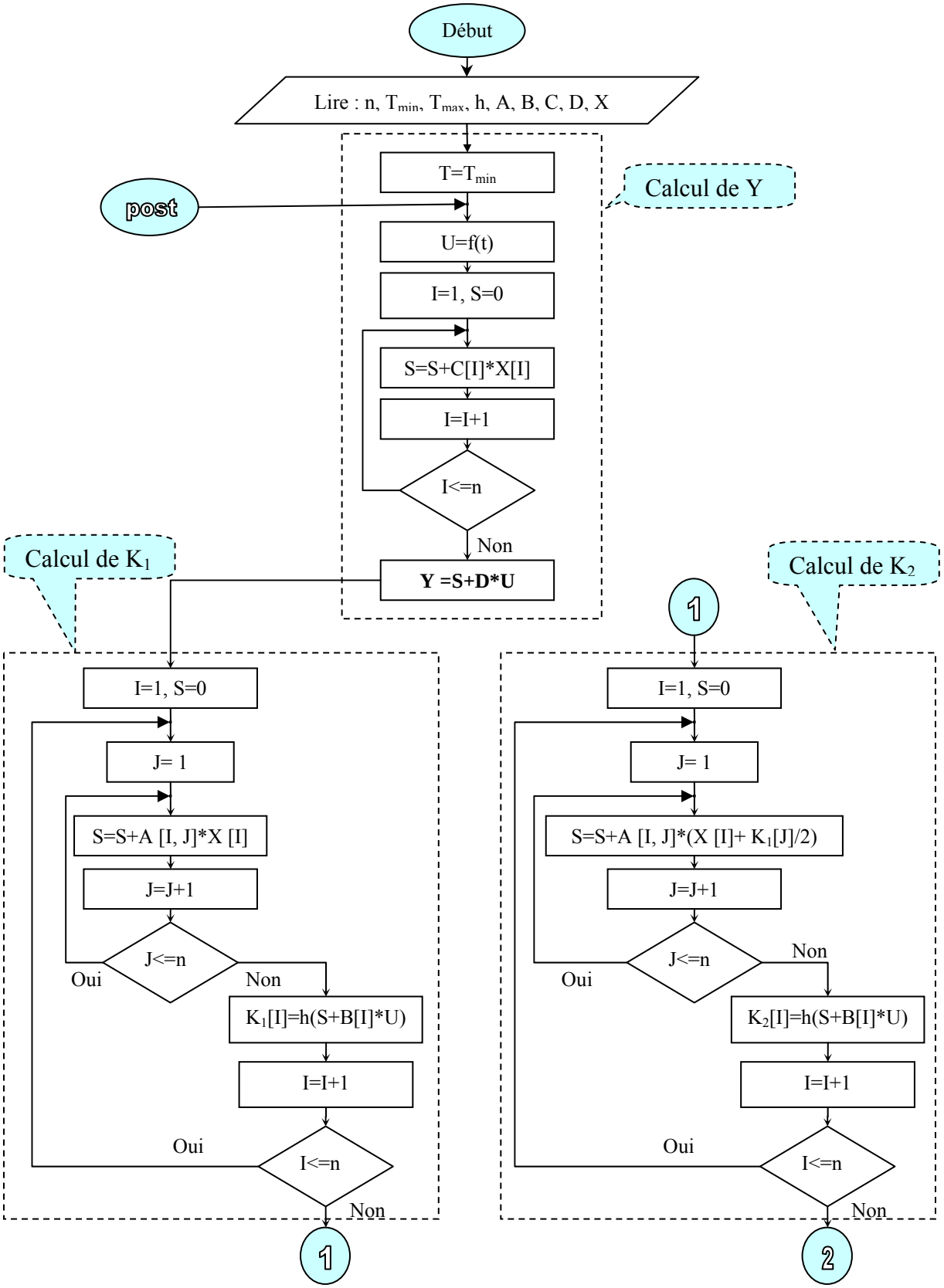
$$k1 = h.f'[X(t),\ Y(t)]$$
$$k2 = h.f'[X(t+1/2),\ Y(t)+k1/2]$$
$$k3 = h.f'[X(t+1/2),\ Y(t)+k2/2]$$
$$k4 = h.f'[X(t+1),\ Y(t)+k3]$$

et donc, $$y(t+1) = Y(t) + [k1 + 2k2 + 2k3 + k4] \times \frac{1}{6} \tag{3.5}$$

On peut remarquer que le calcul de X(t+1) se fait en fonction de X(t) ce qui permet de cerner la forme de la courbe et donc de déduire les risques d'erreurs au maximum contrairement aux autres méthodes. En outre, la méthode de *Rung-Kutta4* se prête facilement à la programmation et produit des résultats satisfaisants pourvu que le pas de discrétisation « h » soit convenablement choisi.

3.3.1.L'ORGANIGRAMME DE CALCUL DE LA REPONSE EN BOUCLE OUVERTE

Puisque notre système sera représenté par les équations matricielles. On a ainsi n équation du 1er ordre à résoudre donc 4*n paramètres dynamiques à résoudre puisque pour chaque équation i, on assigne 04 paramètres $ki = [k_{1i}, k_{2i}, k_{3i}, k_{4i}]$. L'organigramme suivant présente les étapes de calcul:


Début
Lire : n, Tmin, Tmax, h, A, B, C, D, X
T=Tmin
post
Calcul de Y
U=f(t)
I=1, S=0
S=S+C[I]*X[I]
I=I+1
I<=n
Non
Y =S+D*U
Calcul de K1
I=1, S=0
J= 1
S=S+A [I, J]*X [I]
J=J+1
J<=n
Oui
Non
K1[I]=h(S+B[I]*U)
I=I+1
Oui
I<=n
Non
1
Calcul de K2
1
I=1, S=0
J= 1
S=S+A [I, J]*(X [I]+ K1[J]/2)
J=J+1
J<=n
Non
Oui
K2[I]=h(S+B[I]*U)
I=I+1
Oui
I<=n
Non
2

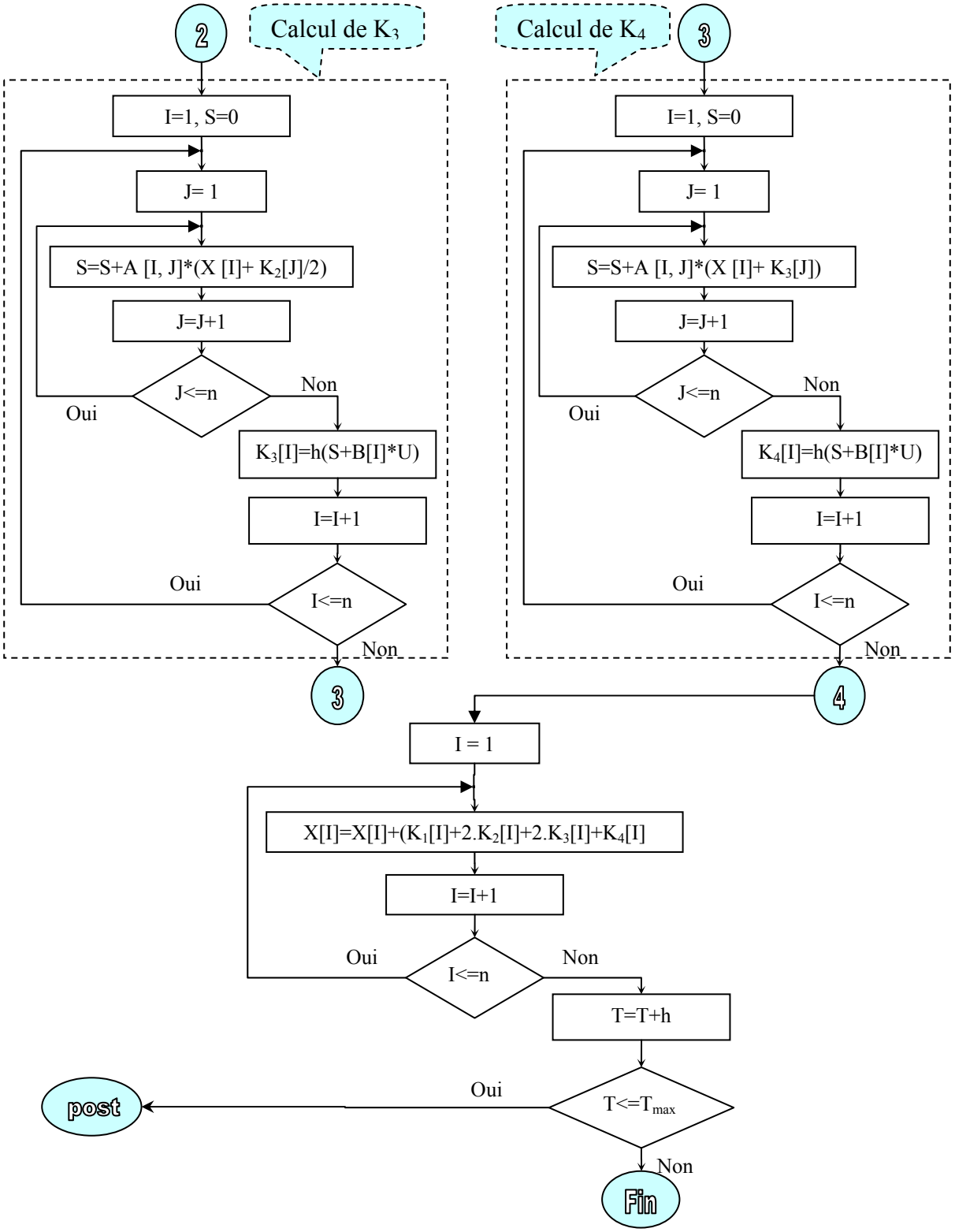


Figure 3.5 : Organigramme de calcul de la réponse en boucle ouverte par Rung-Kutta 4

3.3.2. L'ORGANIGRAMME DE CALCUL DE LA REPONSE IMPULSIONNELLE

On déduira la réponse impulsionnelle à partir de la réponse indicielle en exploitant l'une des propriétés des systèmes linéaires. Si on excite le système par la dérivée du signal d'entrée, on obtient la dérivée de la réponse du même signal.

On a : $Y_1(t) = C.X_1(t) + D.U_1(t)$

Et $\delta(t) = \frac{d}{dt}U_1(t)$

En utilisant la propriété de linéarité on trouve:

$$Y_{imp}(t) = \frac{d}{dt}Y_1(t) = C.\frac{d}{dt}X_1(t) + D.\delta(t) \tag{3.6}$$

Où :

- $Y_1(t)$: Réponse indicielle
- $Y_{imp}(t)$: Réponse impulsionnelle
- $U_1(t)$: Entrée échelon
- $\delta(t)$: Impulsion de Dirac

On déduit ainsi aisément la réponse impulsionnelle,

Si $D = 0 \Rightarrow Y_{imp}(t) = C.\frac{d}{dt}X_1(t) \quad pour\ t \geq 0$

Dans le cas contraire :

$Y_{imp}(0) = \infty$ et $Y_{imp}(t) = C.\frac{d}{dt}X_1(t) \quad pour\ t > 0$

Le vecteur d'état $\frac{d}{dt}X_1(t)$ est calculé bien évidemment en considérant $U(t) = U_1(t)$ (Entrée échelon)

$$\frac{d}{dt}X_1(t) = A.X_1(t) + B.U_1(t) = A.X_1(t) + B \tag{3.7}$$

Ainsi, le calcul de la réponse impulsionnelle en boucle ouverte ou en boucle fermée se fera à partir de :

- L'équation d'état à l'entrée échelon pour la détermination de $\frac{d}{dt}X_1(t)$.
- L'équation de sortie à l'entrée impulsionnelle avec pour états $Y_{imp}(t) = \frac{d}{dt}X_1(t)$

Avec toutes ces explications, on peut dire que l'organigramme qui calcul la réponse impulsionnelle est le même que le précédant organigramme (Figure 3.5), la seule différence est porté sur le calcul de la sortie Y. L'organigramme ci-après va mettre en évidence cette différente:

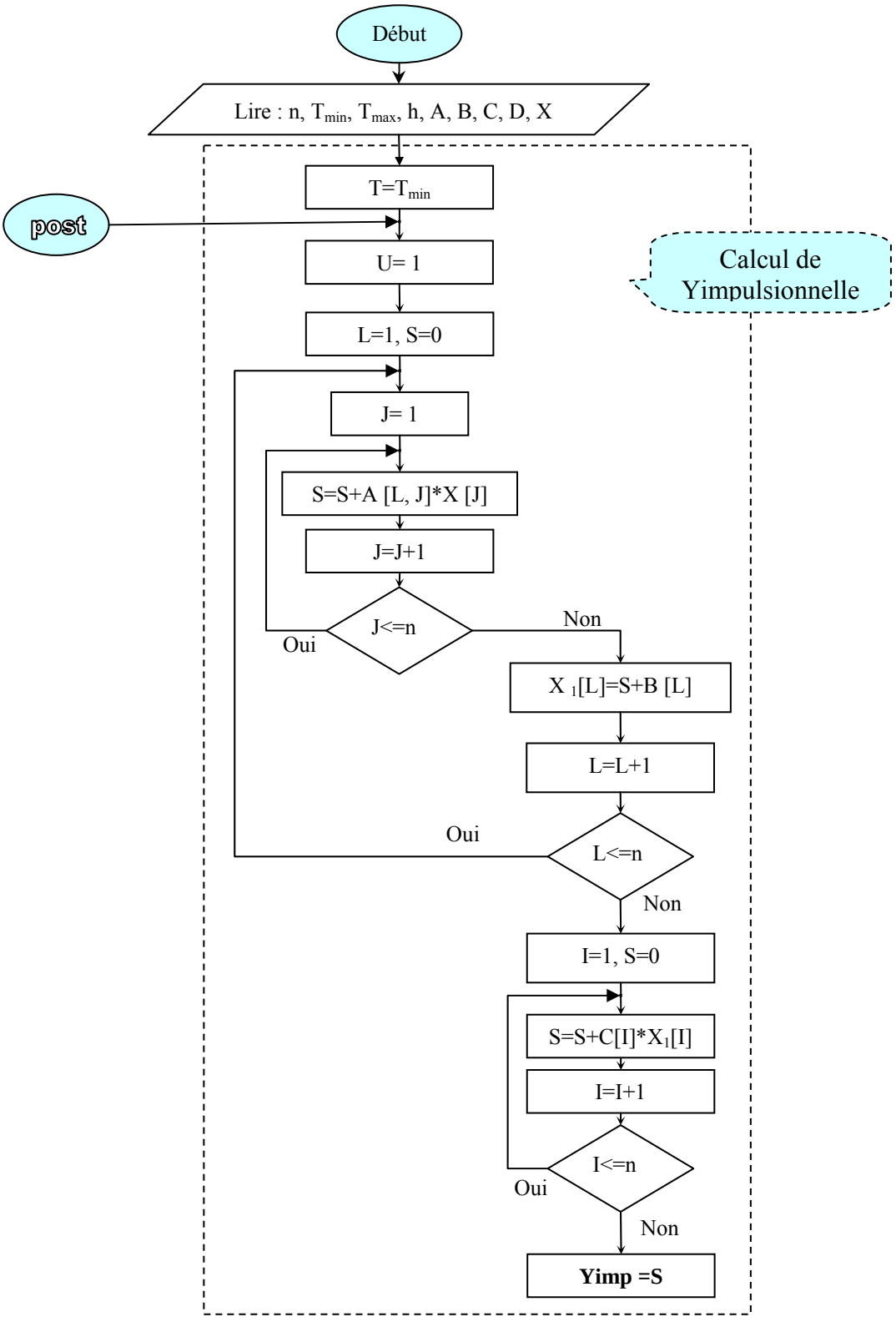


Figure 3.6 : Organigramme partiel de calcul de la réponse impulsionnelle par Rung-Kutta 4

3.4. MENU « ANALYSE TEMPORELLE »

La structure d'exécution du bouton de traçage dans ce menu est donnée par le schéma généralisé suivant :

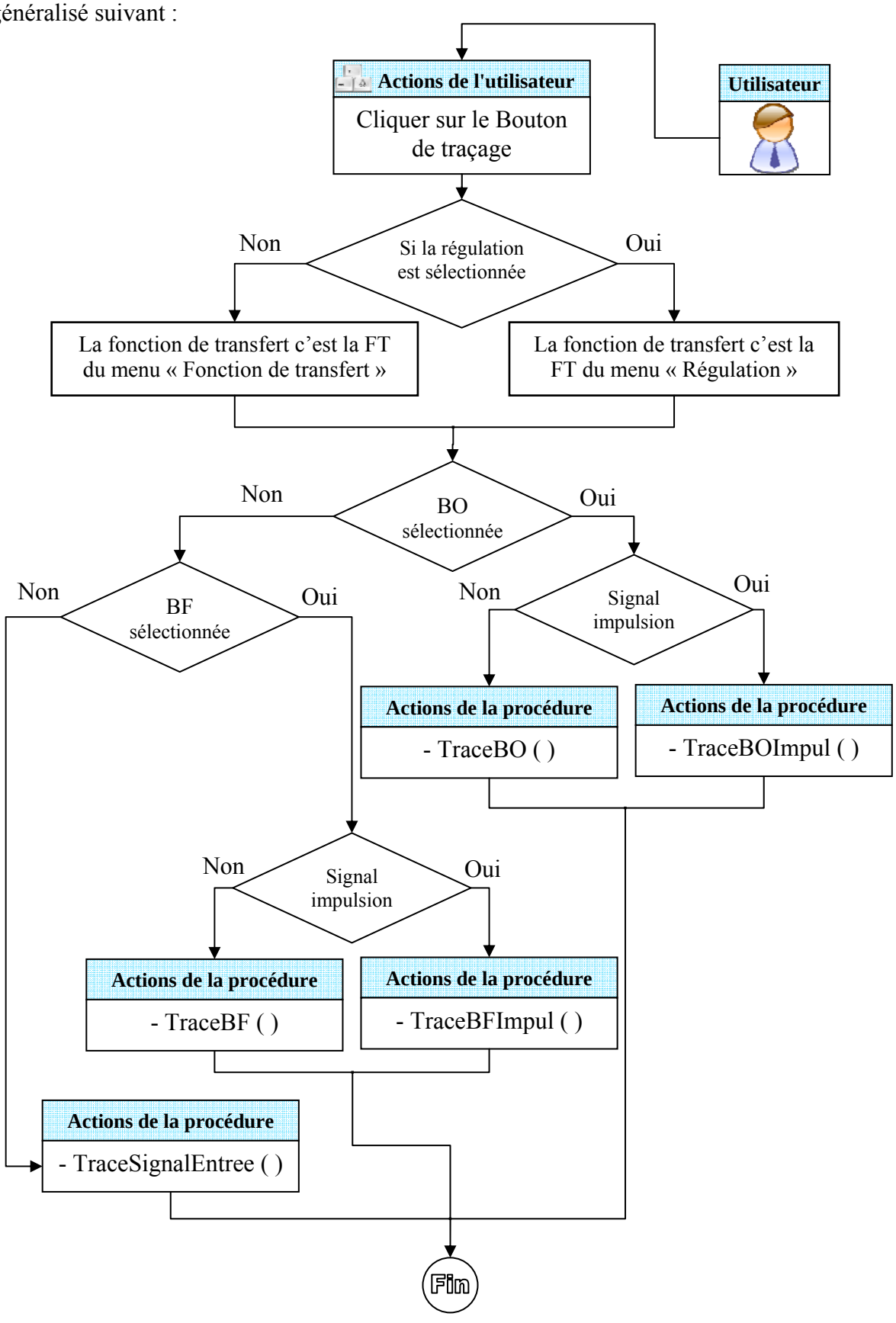


Figure 3.7 : schéma généralisé interprétant la structure de fonctionnement de menu analyse temporelle

3.4.1.LA PROCEDURE TraceBO ( )

En boucle ouverte, le système peut être schématisé de la manière suivante :

Figure 3.8 : Schémas bloc utilisés pour calculer la réponse en boucle ouverte
a- FTBO pour le système sans régulation
b- FTBO pour le système avec régulation

On a suivi pour tracer la réponse en boucle ouverte trois étapes :

- ***Etape 1 :*** le calcul des matrices d'état A, B, C et D à partir de la fonction de transfert à analyser (FT du système avec ou sans régulation) pour cela on a utilisées les fonctions *DimduMatriceA ( ), MatriceA-du-FT ( ), MatriceB-du-FT ( ), MatriceC-du-FT ( )* et la fonction *MatriceD-du-FT ( ).*
- ***Etape 2 :*** la détermination de la réponse en boucle ouverte par application de la fonction *RungKutta4 ( ),* qui sert à résoudre l'équation d'état.
- ***Etape 3 :*** le traçage de la réponse en boucle ouverte.

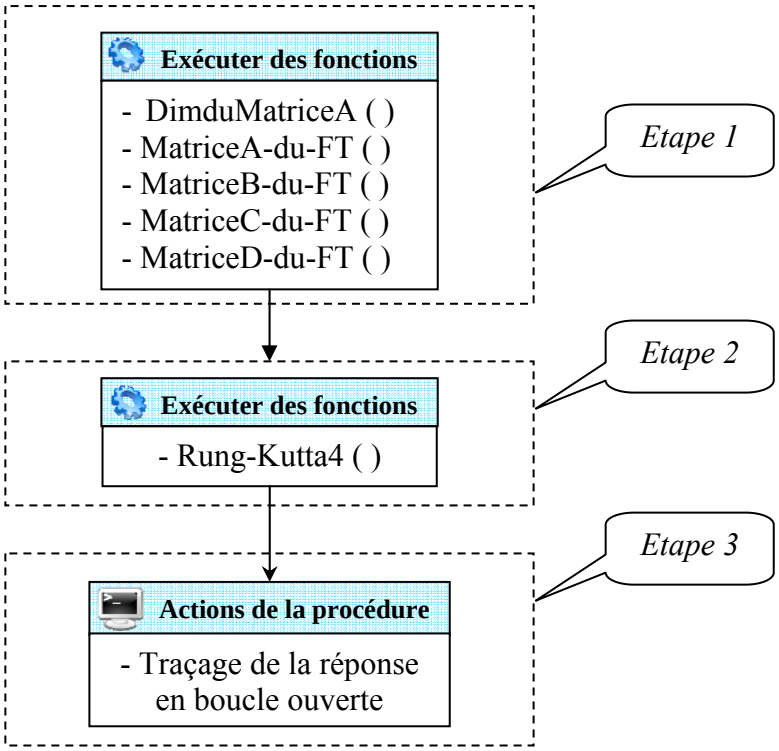


Figure 3.9 : schéma fonctionnel qui interprète la procédure *TraceBO ( )*

3.4.2.LA PROCEDURE TraceBOimpul ( )

Pour obtenir la réponse impulsionnelle en boucle ouverte on a suivait les mêmes étapes que dans la procédure *TraceBO ( )*, sauf que dans la deuxième étape on a appliquée la fonction *RungKutta4impul ( )*, (voir partie 3.3.2).

3.4.3.LA PROCEDURE TraceBF ( )

En boucle fermée, le système peut être schématisé de la manière suivante :

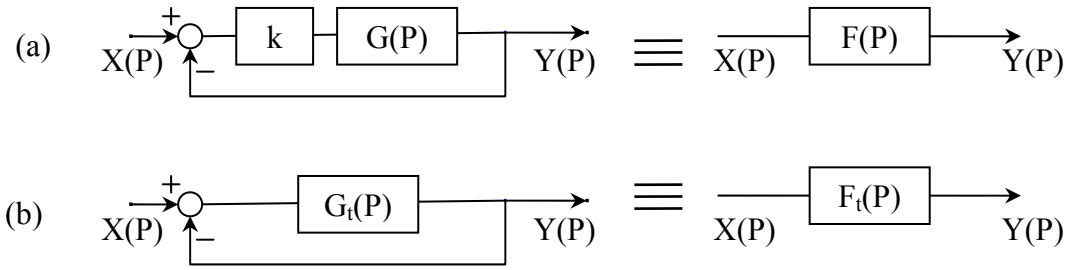


Figure 3.10 : Schémas bloc utilisés pour calculer la réponse en boucle fermée
a- FTBF pour le système sans régulation (non corrigé dans le menu régulation)
b- FTBF pour le système avec régulation

On approchera le problème en considérant qu'un système en boucle fermée est le même système en boucle ouverte attaqué à l'entrée par une erreur amplifiée $u(t) = k.e(t) = k.[x(t) - y(t)]$. On revient ainsi au même calcul que celui fait précédemment pour le traçage de la réponse en boucle ouverte, où on simplifié de plus, cette fonction de transfert en boucle fermée sous forme d'une fonction en boucle ouverte. (Voir figure 3.10)

On peut interpréter la structure d'exécution de la procédure *TraceBF ( )* suivant les étapes :

- ***Etape 1 :*** si la régulation est sélectionné précédemment dans le menu « Régulation », on calcul la fonction de transfert en boucle fermée à retour unitaire de Gt(p) (fonction de système corrigé); Si non, on calcul la fonction de transfert en boucle fermée à retour unitaire de la fonction G(p) (fonction de système non corrigé) avec la valeur de gain k. Notant que le calcul est fait par application des fonctions : *SommePoly ( )* et *MultScalPoly ( )*.
- ***Etape 2 :*** on déduire les matrices d'état à partir la fonction de transfert en boucle fermée résultante par application des fonctions citées précédemment dans l'*étape 1* de la procédure *TraceBO ( )*.
- ***Etape 3 :*** Résoudre l'équation d'état obtenue par application de la fonction *RungKutta4 ( )*.
- ***Etape 4 :*** Traçage de la réponse en boucle fermée.

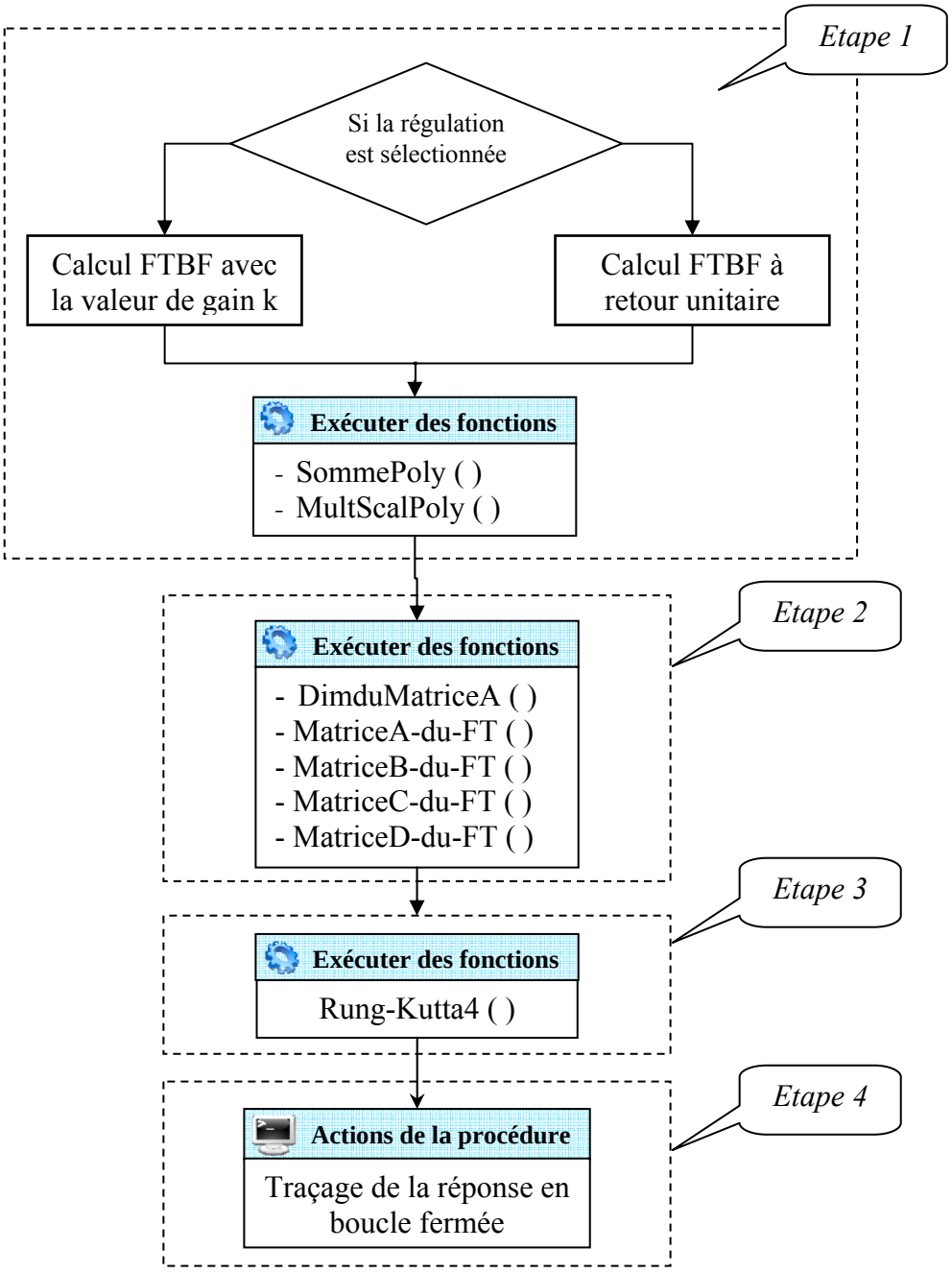


Figure 3.11 : schéma fonctionnel qui interprète la procédure *TraceBF ( )*

3.4.4.LA PROCEDURE TraceBFimpul ( )

Pour tracer la réponse impulsionnelle en boucle fermée, on peut suivre les mêmes étapes que dans la procédure *TraceBF ( )*, la seul différence est que la résolution d'équations d'état est fait par application de la fonction *RungKutta4Impul ( )*.

3.4.5.LA PROCEDURE TraceSignalEntre ( )

Cette procédure sert à tracer le signal d'entrée choisie par l'utilisateur dans le sous menu « Options ». Les signaux d'entrées qui sont réalisés sont les quatre types canoniques (échelon,

rampe, parabolique et impulsion) et en plus le signal harmonique, le signal quelconque et mixte nous les interprétons plus tard dans le sous menu « options ».

3.4.6. SOUS MENU « OPTIONS »

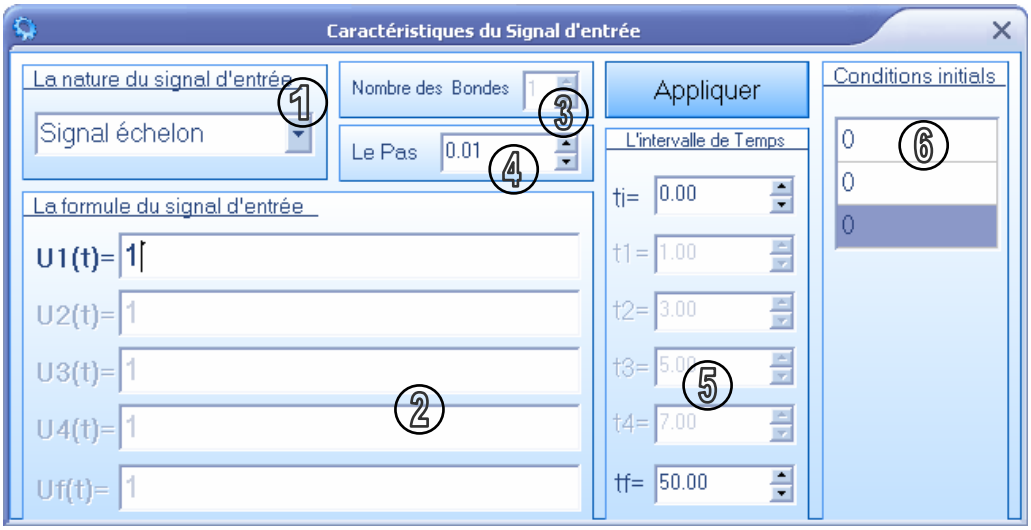
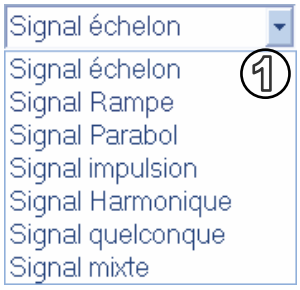


Figure 3.12 : Interface de la fenêtre options du menu Analyse temporelle



Un champ pour sélectionner le signal d'entrée du système, il comporte le signal indiciel, rampe, parabole, impulsionnelle, harmonique,quelconque et mixte où ce dernier peut composer de plusieurs signaux U_1,U_2,U_3…, chaque signal est défini dans leur intervalle (ex: U_1a l'intervalle [ti,t1]).

2 Ces sont des champs faites pour saisir la formule du signal en fonction du temps.

3 Champ pour entrer le nombre des bondes (nombre des intervalles).

4 Champ pour valider la valeur de pas h de Rung-Kutta4.

5 Les champs pour saisir les valeurs des temps correspondent aux bondes.

6 Un tableau pour remplir les valeurs du vecteur de conditions initiales

3.5. CONCLUSION

Cette partie a bénéficié de l'expérience que nous avons acquis et, qui a servi à élaborer une analyse temporelle souple et complète. La méthode de *Rung-Kutta4* s'est montrée efficace puisque elle permet de résoudre facilement les équations différentielles avec des risques minimes de divergence.

CHAPITRE IV

ANALYSE FREQUENCIELLE

4.1. INTRODUCTION

L'analyse fréquentielle d'un système est primordiale pour la prédiction du comportement que pourrait exprimer le système soumis à des excitations données. Nous savons à présent, que tout signal, périodique ou non, peut être décomposé en une série dénombrable (spectre discret) ou indénombrable (spectre continu) de fréquences. La juxtaposition du signal d'entrée et de la fonction de transfert sur le plan fréquentiel définira totalement la réponse du système considéré. A une fréquence donnée, le système jouera le rôle d'amplificateur, d'atténuateur, ou de suiveur au niveau de l'amplitude et de déphaseur au niveau de la phase. Ces contributions apportées par le système sur le signal d'entrée sont mises en évidence par le diagramme de *Bode* qui constitue un outil d'analyse puissant et efficace.

Un moyen d'étude d'un système linéaire continu invariant (SLCI) consiste en l'analyse du comportement dynamique de celui–ci. Elle est fondée sur l'entrée d'un signal harmonique et sur l'étude de la réponse du système en régime permanent.

4.2. CARACTERISATION HARMONIQUE D'UN SLCI

L'entrée du système est un signal sinusoïdal de la forme $E(t) = E_0 \sin \omega t$ .On montre qu'en régime permanent, la réponse d'un SLCI est elle aussi de la forme sinusoïdale avec la même pulsation ω que le signal d'entrée *E*, mais déphasée d'un angle φ par rapport à lui. La réponse a donc la forme :

$$S(t) = S_0 \sin(\omega t + \varphi) \tag{4.1}$$

On utilise alors les nombres complexes afin d'étudier l'amplitude S_0 et la phase φ de la réponse du SLCI. Introduisons deux variables complexes $\overline{E}$ et $\overline{S}$ définies par :

$$\overline{E} = E_0 . e^{j\omega t} \tag{4.2}$$

$$\overline{S} = S_0 . e^{j(\omega t + \varphi)} \tag{4.3}$$

L'entrée E et la sortie S sont respectivement les parties imaginaires de $\overline{E}$ et $\overline{S}$.

L'équation différentielle traduisant le comportement du SLCI est:

$$a_n \frac{d^n S(t)}{dt^n} + a_{n-1} \frac{d^{n-1} S(t)}{dt^{n-1}} + ... + a_0 S(t) = b_m \frac{d^m e(t)}{dt^m} + b_{m-1} \frac{d^{m-1} e(t)}{dt^{m-1}} + ... + b_0 e(t) \tag{4.4}$$

et peut alors s'écrire dans le domaine complexe :

$$a_n (j\omega)^n \overline{S} + a_{n-1} (j\omega)^{n-1} \overline{S} + ... + a_0 \overline{S} = b_m (j\omega)^m \overline{E} + b_{m-1} (j\omega)^{m-1} \overline{E} + ... + b_0 \overline{E} \tag{4.5}$$

La fonction de transfert du système s'écrit alors en fonction de la pulsation ω comme suit :

$$G(j\omega) = \frac{S(p)}{E(p)} = \frac{b_m (j\omega)^m + b_{m-1} (j\omega)^{m-1} + ... + b_1 (j\omega) + b_0}{a_n (j\omega)^n + a_{n-1} (j\omega)^{n-1} + ... + a_1 (j\omega) + a_0} \tag{4.6}$$

Et l'amplitude S_0 et le déphasage φ entre l'entrée et la sortie se calcule par :

$$S_0 = E_0 |G(j\omega)| \tag{4.7}$$

$$\varphi = Arg\, G(j\omega) \tag{4.8}$$

4.3. OUTILS MATHIMATIQUES :

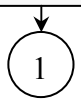
Dans cette partie, on expliquera les fonctions qui seront intégrées dans la même structure d'exécution des méthodes fréquentielles que l'on détaillera plus tard :

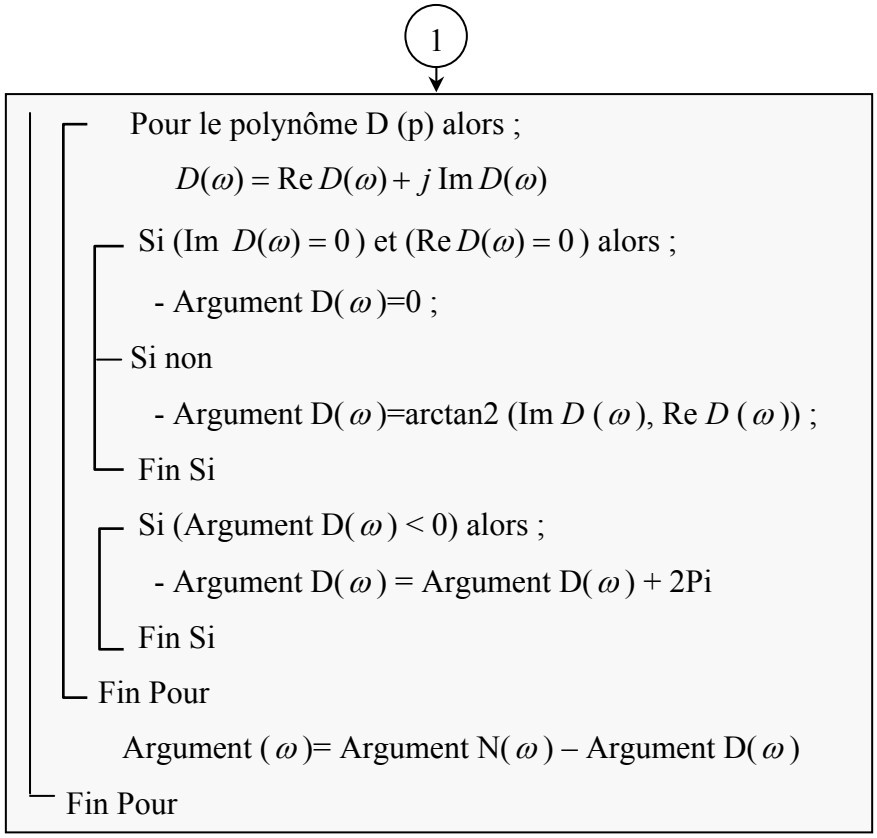
- **Fonction Modullog ( )**

```
Si G (p)=N (p)/D (p) alors ;
    Pour A=N (p) et B=D (p) alors ;
        - Calcul module (N) ;
        - Calcul module (D) ;
        - Calcul 20log (A/B);
    Fin Pour
Fin Si
```

- **Fonction Argument ( )**

```
Pour le Numérateur N (p) alors ;
    N(ω) = Re N(ω) + j Im N(ω)
    Si (Im N(ω) = 0 ) et (Re N(ω) = 0 ) alors ;
        - Argument N( ω )=0 ;
    Si non
        - Argument N( ω )=arctan2 (ImN ( ω ), Re N ( ω )) ;
    Fin Si
    Si (Argument N( ω ) < 0) alors ;
        - Argument N( ω ) = Argument N( ω ) + 2Pi
    Fin Si
Fin Pour
```





4.4. METHODES D'ANALYSE FREQUENTIELLE

Les valeurs prises par une fonction de transfert complexe *G(jω)* dépendent des valeurs de la pulsation ω. Traditionnellement, on représente graphiquement suivant les diagrammes, l'évolution de *G(jω)* en fonction de ω.

Dans notre logiciel, lorsque on veut valider l'analyse fréquentielle, toute une procédure se met en route en considérant le dernier système saisi. Le choix est laissé aux soins de l'utilisateur qui verra afficher sur l'écran les tracés de différents diagrammes fréquentiels à partir de la fonction de transfert à analyser que ce soit du système corrigé (dans le menu régulation) ou non corrigé.

La figure ci-dessous schématise comment obtenir la fonction de transfert à analyser.

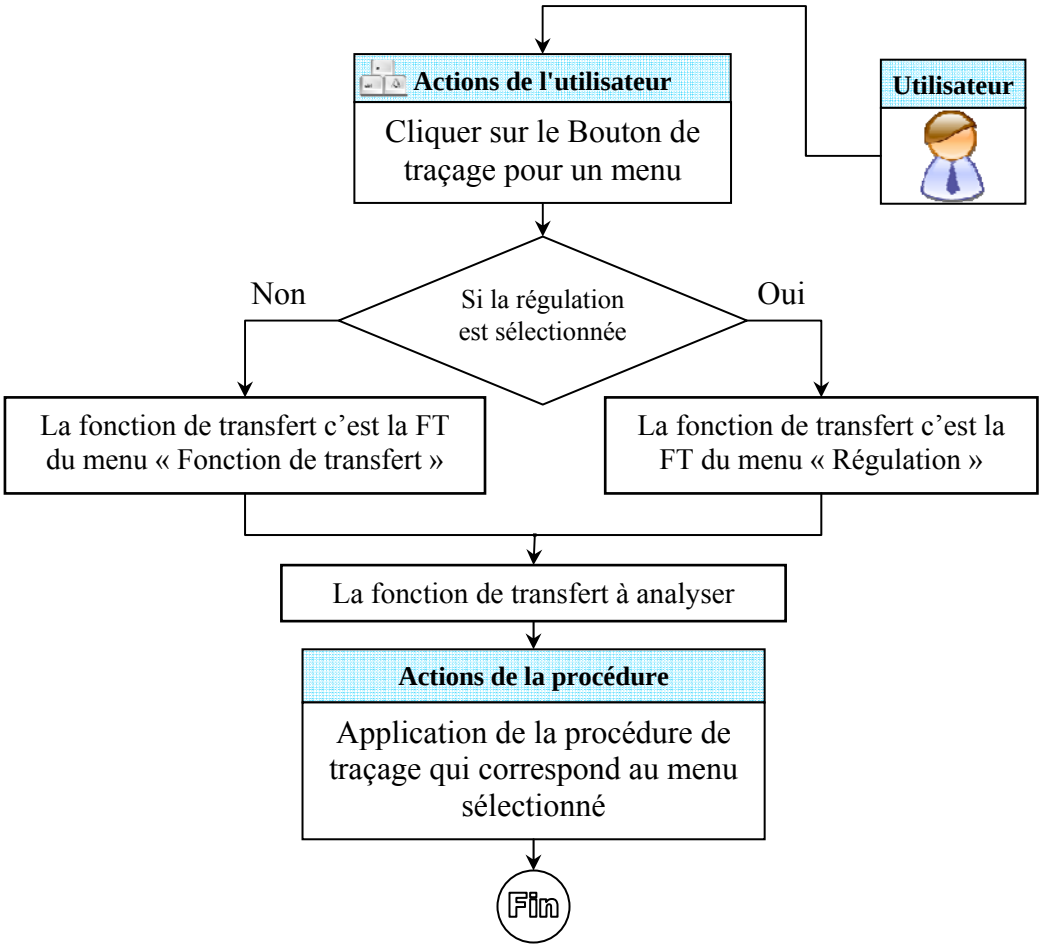


Figure 4.1: schéma fonctionnel expliquant comment on calcul la FT à analyser

4.4.1.MENU DE BODE

Il est réalisé à l'aide de deux courbes distinctes : la courbe représentant le module de la fonction de transfert (exprimé en décibel, dB) en fonction de la pulsation, et la courbe représentant la phase de la fonction de transfert (exprimée généralement en degré) en fonction de la pulsation. Rappelons que le module exprimé en décibels correspond à $20\log|G(j\omega)|$ (logarithme décimal). Les axes des abscisses sont gradués sur une échelle logarithmique (on parle de graphiques semi–logarithmiques).

Le module et l'argument d'une fonction de transfert sont déterminés sous les formes suivantes :

$$20\log|G(j\omega)| = 20\log|N(j\omega)| - 20\log|D(j\omega)| \tag{4.9}$$

$$ArgG(j\omega) = ArgN(j\omega) - ArgD(j\omega) \tag{4.10}$$

Où $N(j\omega)$ et $D(j\omega)$ représentent respectivement le numérateur et le dénominateur de $G(j\omega)$.

En ce qui concerne la programmation du menu « Bode », on a partagé leur fonctionnement en trois menus, le premier illustre l'amplitude, le second la phase et le troisième est mixte. Ces menus sont représentés comme suit :

4.4.1.4. MENU « AMPLITUDE »

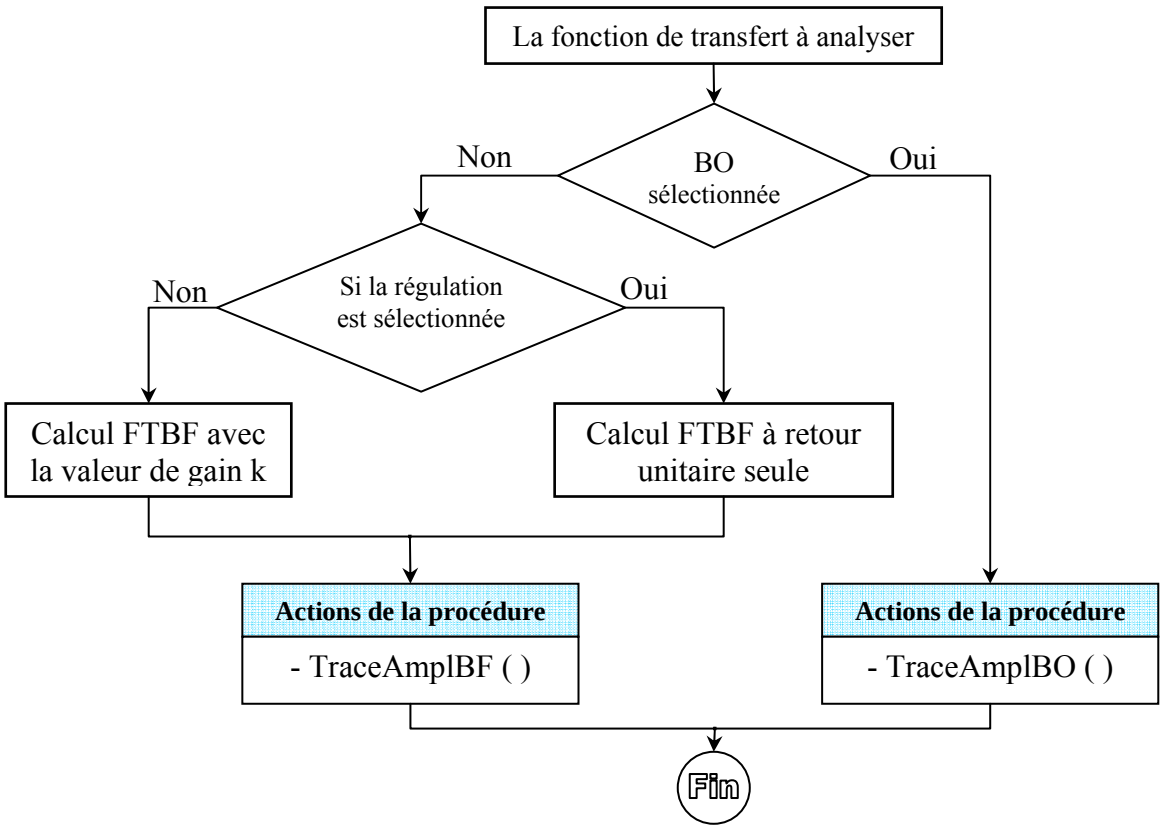


Figure 4.2: schéma fonctionnel expliquant le traçage de l'amplitude

4.4.1.4.1. LA PROCEDURE **TraceAmplBO ( )**

La structure d'exécution de traçage du diagramme d'amplitude se présente comme suit :

- *Etape1* **:** chargement des paramètres : $\omega(initial)$, $\omega(final)$, fonction de transfert en BO

- *Etape2* **:** calcul du module en décibel pour l'intervalle $[\omega(initial), \omega(final)]$ avec un pas défini par application de la fonction *Modullog ( )*

- *Etape3***:** traçage de la courbe de l'amplitude en boucle ouverte

Début
Var : ω : réel;
Lire : $\omega(initial)$, $\omega(final)$, $G(j\omega)$;
$\omega = \omega(initial)$;
Répéter
$\omega = \omega + \omega/100$;
- Calcul **Modullog *G(j ω)*** ;
- Tracer le point (ω , **Modullog *G(j ω)***) ;
Jusqu'à $\omega \geq \omega(final)$
Fin

4.4.1.4.2. LE PROCEDURE TraceAmplBF ( )

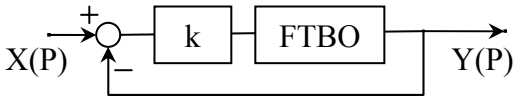


Figure 4.3 : Schéma bloc utilisé en boucle fermée

Comme l'indique la figure ci-dessus, sachant que notre boucle fermée se résume à un retour unitaire, la fonction de transfert en boucle fermée s'écrit :

$$F(p) = \frac{k.FTBO}{1 + k.FTBO} \tag{4.11}$$

Notons que cette derniere est calculée par application des fonctions *MultScalPoly ( )* et *SommePoly ( ).*

Si la fonction de transfert est corrigée on pose la valeur du gain égale à l'unité (k=1).

A partir de la formule (4.1), il suffira de faire varier la pulsation ω avec un pas bien déterminé pour obtenir les courbes en boucles fermées. Le calcul des valeurs de l'amplitude en boucle fermée se fait par la fonction *Modullog ( ).* L'algorithme ci-dessous explique cette procédure :

Début

Var : $K,\ \omega$: réels

Lire : $K,\ ,\ \omega(initial)\ ,\omega(final)\ ,G(j\omega)$;

Si (Le système est corrigé dans le menu régulation)

- (Calcul de la fonction de transfert en boucle fermée $F(j\omega)$ correspond au K=1)

Sinon

- (Calcul de la fonction de transfert en boucle fermée $F(j\omega)$ correspond au K)

Fin Si

$\omega = \omega(initial)$

Répéter

$\omega = \omega + \omega / 100;$

- Calcul **Modullog** $F(j\omega)$) ;
- Tracer le point (ω , **Modullog** $F(j\omega)$) ;

Jusqu'à $\omega \geq \omega(final)$

Fin

4.4.1.5. MENU « PHASE »

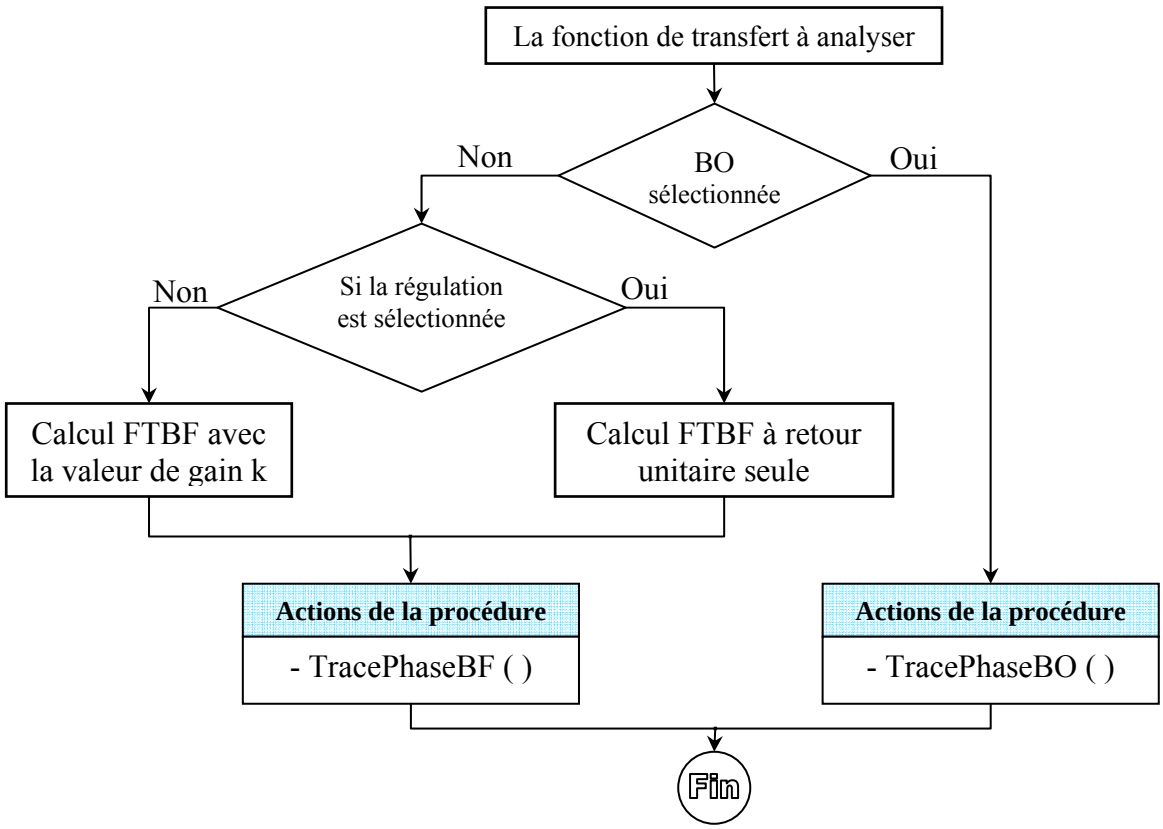


Figure 4.4: schéma fonctionnel expliquant le traçage de la phase

4.4.1.5.1. LA PROCEDURE TracePhaseBO ( )

Le fonctionnement de cette procédure est le même que dans la procédure *TraceAmplBO ( )*. La seule différence présente ici est le calcul de l'argument au lieu du module en décibel en appliquant la fonction *Argument ( )*. Pour observer cette différence on peut se reporter à l'algorithme suivant:

```
Début
 Var : ω : réel;
 Lire : ω(initial), ω(final), G(jω);
    ω = ω(initial);
  Répéter
     ω = ω + ω/100;
     - Calcul Argument G(jω) ;
     - Tracer le point (ω, Argument G(jω)) ;
  Jusqu'à ω ≥ ω(final)
 Fin
```

4.4.1.5.2. LA PROCEDURE TRACEPHASEBF ( )

L'algorithme ci-dessous interprète le fonctionnement de cette procédure :

```
Début
 Var : K, ω : réels
 Lire : K, ω(initial), ω(final), G(jω) ;
      Si (Le système est corrigé dans le menu régulation)
       - (Calcul de la fonction de transfert en boucle fermée F(jω) correspond au K=1)
      Sinon
       - (Calcul de la fonction de transfert en boucle fermée F(jω) correspond au K)
      Fin Si
        ω = ω(initial)
      Répéter
       ω = ω + ω/100;
       - Calcul Argument F(jω)) ;
       - Tracer le point (ω, Argument F(jω)) ;
      Jusqu'à ω ≥ ω(final)
 Fin
```

4.4.1.6. MENU « MIXTE »

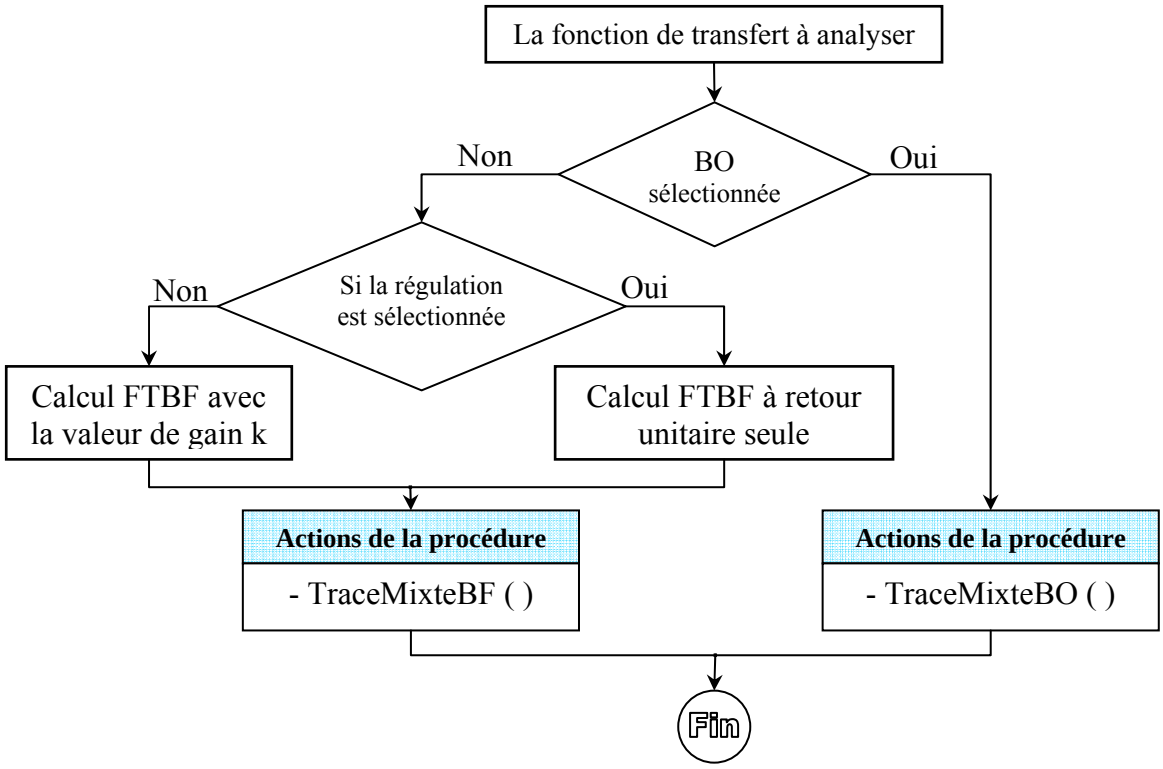


Figure 4.5: schéma fonctionnel expliquant le traçage du diagramme mixte

4.4.1.6.1. LA PROCEDURE TraceMixteBO ( )

Dans ce menu, on suppose que l'on désire représenter le traçage de l'amplitude et la phase dans le même plan. L'algorithme ci-dessous montre la structure d'exécution de cette procédure de traçage :

Début
Var : ω réel;
Lire : $\omega(initial), \omega(final), G(j\omega)$;
$\omega = \omega(initial)$;
Répéter
$\omega = \omega + \omega/100$;
- Calcul **Argument** $G(j\omega)$;
- Tracer le point (ω, **Argument** $G(j\omega)$) ;
- Calcul **Modullog** $G(j\omega)$;
- Tracer le point (ω, **Modullog** $G(j\omega)$) ;
Jusqu'à $\omega \geq \omega(final)$
Fin

4.4.1.6.2. LA PROCEDURE TraceMixteBF ( )

L'algorithme ci-dessous explique le fonctionnement de cette procédure :

Début
Var : K, ω : réels
Lire : K, $\omega(initial)$, $\omega(final)$, $G(j\omega)$;
  Si (Le système est corrigé dans le menu régulation)
  - (Calcul de la fonction de transfert en boucle fermée $F(j\omega)$ correspond au K=1)
  Sinon
  - (Calcul de la fonction de transfert en boucle fermée $F(j\omega)$ correspond au K)
  Fin Si
  $\omega = \omega(initial)$
  Répéter
  $\omega = \omega + \omega / 100;$
  - Calcul **Argument** $F(j\omega)$) ;
  - Tracer le point (ω ,**Argument** $F(j\omega)$) ;
  - Calcul **Modullog** $F(j\omega)$;
  - Tracer le point (ω , **Modullog** $F(j\omega)$) ;
  Jusqu'à $\omega \geq \omega(final)$
Fin

4.4.2.LA REPRESENTATION DE NYQUIST

Le menu *Nyquist* est destiné à visualiser la transformé du contour Γ du plan P incluant le demi plan droit de ce même plan (voir la figure 4.6). L'utilité de l'étude de *Nyquist* réside dans le fait de déduire la stabilité de notre système en boucle fermée en connaissant :

- Le nombre de pôles de notre système en BO qui sont situés sur le demi plan d'instabilité.
- La forme de la transformée du contour Γ sur le plan complexe et son comportement autour du point critique (-1, 0).

La transformé T appliquée à notre contour Γ n'a d'autre loi que la fonction de transfert régissant notre système.

Pour réaliser la transformé du contour Γ il faut tenir compte de la probable existence de pôles sur l'axe des imaginaires. On citera deux problèmes essentiels :

- Le pas du tracé
- La gestion des portions dues à l'éventuelle existence de pôles sur l'axe des imaginaires.

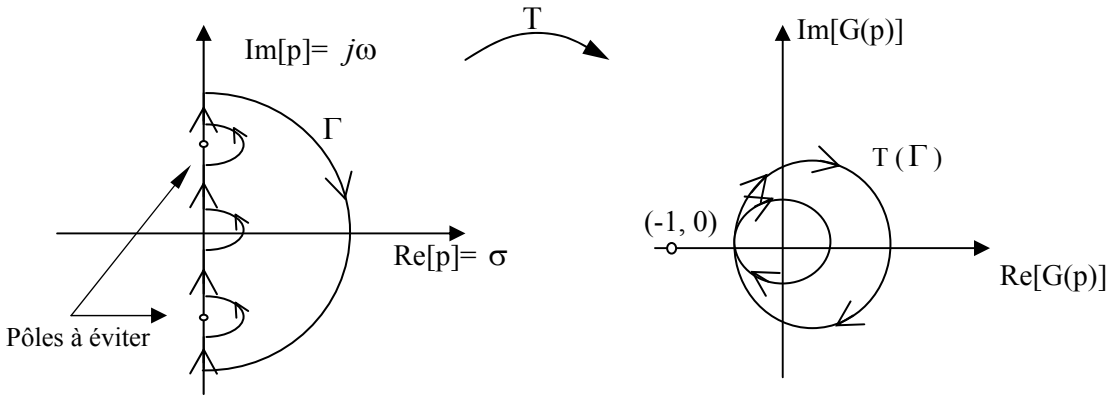


Figure 4.6: graphes représentant la transformée dans le plan complexe

Le contour fermé Γ est un demi cercle de rayon infiniment grand et situé dans le demi plan droit de l'axe imaginaire, enfermant tous les pôles à partie réelle positive d'une fonction $G(p)$.

Si $G(p)$ présente des pôles sur l'axe imaginaire, et afin d'éviter ces points on modifie le contour en lui rajoutant de petits demi cercle autour de chaque pôle. Le contour est parcouru dans le sens des aiguilles d'une montre.

Il est évident que le tracé du lieu sur une échelle finie n'aura aucun sens dans le cas où le système serait pourvu de pôles sur l'axe des imaginaires puisque ces derniers engendreraient des variations à l'infini et que donc seules les asymptotes seront visibles. On a ainsi deux types de tracé :

4.4.2.4. TRACE SUR ECHELLE FINIE

La pulsation ω prendra des valeurs croissantes avec un pas bien choisi. Cependant, la présence de pôles imaginaires va nous pousser à associer une marge de détection fixée à ±20% de la valeur du pôle. La procédure sera comme suit :

Si $j\omega$ entre dans la marge [pôle -20%, pôle +20%], on enclenchera la procédure correspondant au tracé du demi cercle qui entoure le pôle. Cette procédure a le mérite de mettre en évidence la continuité asymptotique du tracé.

4.4.2.5. TRACE SUR ECHELLE INFINIE

Le tracé sur échelle infini ne sera possible que si notre système est pourvu de pôles imaginaires. On pensera toutefois à la continuité du Lieu qui sera matérialisée par des asymptotes. Bien entendu, ce genre de tracé est pédagogique.

4.4.2.6. MENU NYQUIST

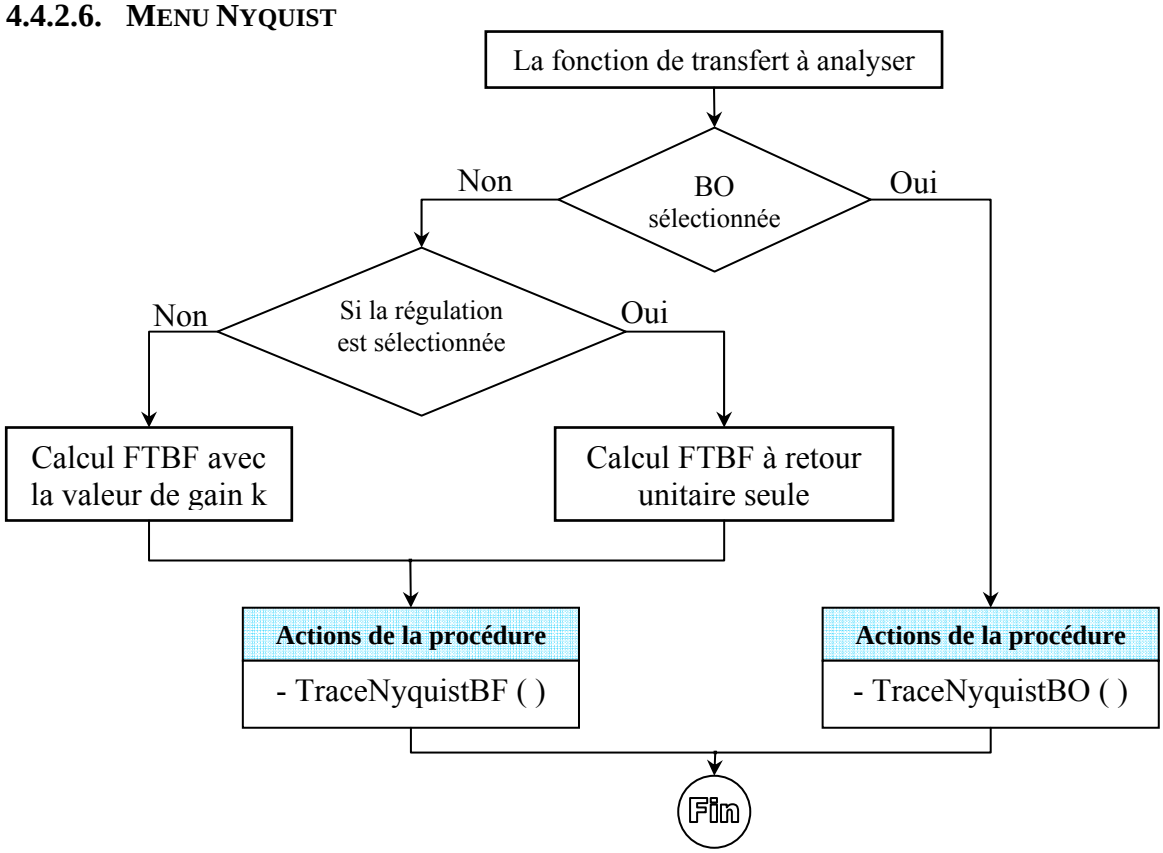


Figure 4.7: schéma fonctionnel expliquant le traçage du *Nyquist*

4.4.2.6.1. LA PROCEDURE Fleche ( )

On lui fera appel pendant les calculs selon une procédure séquentielle, basée sur l'utilisation d'un compteur qui servira de repère séquentiel, et en lui communiquant les cordonnées des deux points à partir desquels la flèche sera affichée.

4.4.2.6.2. LA PROCEDURE TraceNyquistBO ( ) ET TraceNyquistBF ( )

On peut expliquer le fonctionnement de cette procédure par les étapes suivantes:

- *Etape 1:* initialisé ω par une valeur minimale choisie;
- *Etape 2:* entré la fonction de transfert en boucle ouverte *G(p)* ou *F(p)* en boucle fermée.
- *Etape 3:* calcul des ordonnées graphiques du point d'affichage p=(x, y) où:

$$\begin{cases} X = |G(p)|.Cos(ArgG(p)) \\ Y = |G(p)|.Sin(ArgG(p)) \end{cases} \quad \text{ou} \quad \begin{cases} X = |F(p)|.Cos(ArgF(p)) \\ Y = |F(p)|.Sin(ArgF(p)) \end{cases}$$

Par application de la fonction *Argument ( )* et *Module ( )*.

- *Etape 4:* Affichage du point et de son conjugué
- *Etape 5:* utilisation de la procédure *Fleche ( )*
- *Etape 6:* utilisation de la procédure qui détectée la proximité de pôle imaginaire, Si oui engager la procédure associée aux tracés des asymptotes.

4.4.3.LA REPRESENTATION DE BLACK

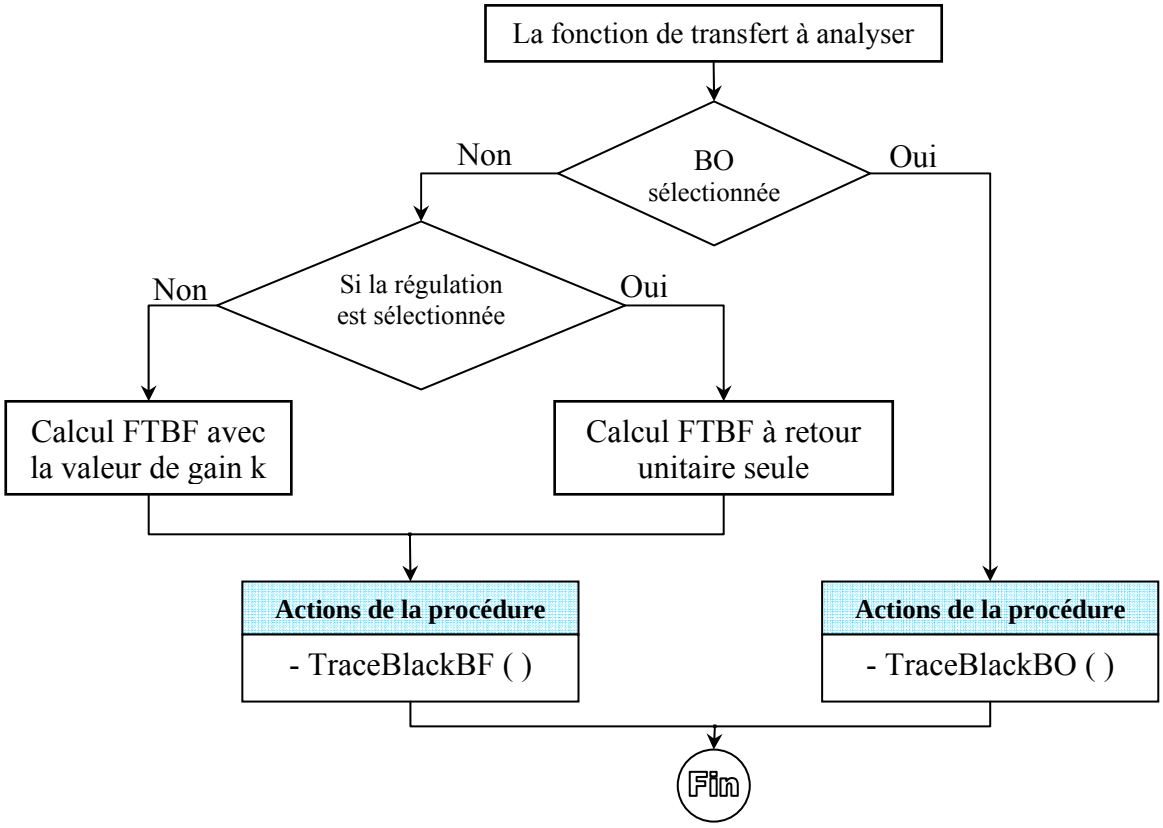


Figure 4.8: schéma fonctionnel expliquant le traçage du lieu de Black

4.4.3.2. MENU « BLACK-NICHOLS »

Ce diagramme (également appelé diagramme de *Nichols*) consiste à représenter le lieu d'une fonction de transfert $G(j\omega)$ dans un plan ayant pour axe des abscisses la phase exprimée en degré, et pour axe des ordonnées le module de $G(j\omega)$ exprimé en décibel, Comme c'est le cas dans le diagramme de *Bode*.

Une telle méthode est d'une utilisation très commode, car un changement du gain K d'un système se traduit par une translation parallèle à l'axe des ordonnées. Ce changement a un effet direct sur la stabilité relative du système.

4.4.3.2.1. LA PROCEDURE TraceBlackBO ( )

- *Etape1* **:** chargement des paramètres : $\omega(initial)$, $\omega(final)$, fonction de transfert en BO
- *Etape2* **:** calcul du module en décibel et l'argument en degré.
- *Etape2***:** traçage du lieu en boucle ouverte :

```
Début
  Var : ω réel;
  Lire : ω(initial) , ω(final) , G(jω) ;
     ω = ω(initial) ;
  Répéter
     ω = ω + ω / 100;
     - Calcul Argument G(jω) ;
     - Calcul Modullog G(jω) ;
     - Tracer le point (Argument G(jω), Modullog G(jω)) ;
  Jusqu'à  ω ≥ ω(final)
  Fin
```

4.4.3.2.2. LA PROCEDURE TraceBlackBF ( )

```
Début
 Var : K, ω : réels;
 Lire : K, ω(initial) , ω(final) , G(jω) ;
      Si (Le système est corrigé dans le menu régulation)
      -  (Calcul de la fonction de transfert en boucle fermée F(jω) correspond au K=1)
      Sinon
      -  (Calcul de la fonction de transfert en boucle fermée F(jω) correspond au K)
      Fin Si
      ω = ω(initial) ;
      Répéter
      ω = ω + ω / 100;
      -  Calcul Argument F(jω)) ;
      - Calcul Modullog F(jω) ;
      - Tracer le point (Argument F(jω), Modullog F(jω)) ;
      Jusqu'à  ω ≥ ω(final)
 Fin
```

4.4.4.L'ABAQUE DE BLACK

L'abaque de Black, permet à partir du tracé du lieu représentant la réponse en fréquences en boucle ouverte G($j\omega$), de tracer directement le lieu représentant la réponse en fréquences en boucle fermée F($j\omega$) à retour unitaire équivalent au système donné :

$$F(j\omega) = \frac{G(j\omega)}{1+G(j\omega)} \tag{4.12}$$

Soient $p = j\omega$ et $G(j\omega) = U + jV$, alors la fonction de transfert en boucle fermée s'écrit :

$$F(j\omega) = \frac{[U + jV]}{[(1+U) + jV]} \tag{4.13}$$

En calculant le module et la phase de $F(j\omega)$, nous aurons les expressions suivantes :

$$M^2 = \frac{[U^2 + V^2]}{[(1+U)^2 + V^2]} \tag{4.14}$$

$$N = tg(\Phi) = \frac{V}{[U + U^2 + V^2]} \tag{4.15}$$

Pour un couple de valeurs (*U, V*), on peut déduire la phase et l'amplitude de notre fonction de transfert en boucle fermée.

Par ailleurs on suppose qu'on désire représenter le problème inverse, quel seraient l'ensemble des points *U* et *V* de notre fonction en boucle ouverte sur le plan de black pour lesquels on peut trouver une amplitude *M* en boucle fermée ?

La même question se pose pour une phase *N* en boucle fermée. On définit ainsi deux familles de lieux géométriques qui représenteront l'abaque de *Black-Nichols* qui servira de référence pour le tracé. Ces familles seront régies par les lois suivantes :

$$\left(U + \frac{M^2}{M^2 - 1}\right)^2 + V^2 = \left(\frac{M}{1 - M^2}\right)^2 \tag{4.16}$$

Qui représente la famille des gains,des cercles de rayon $\left|\frac{M}{1-M^2}\right|$ et de centre $\left(\frac{M}{1-M^2},\ 0\right)$.

$$\left(U + \frac{1}{2}\right)^2 + \left(V - \frac{1}{2N}\right)^2 = \frac{N^2 + 1}{4N^2} \tag{4.17}$$

Qui représente la famille des phases, des cercles de rayon $\frac{\sqrt{N^2+1}}{4N}$ et de centre $\left(-\frac{1}{2},\ \frac{1}{2N}\right)$.

Bien entendu, ces équations sont explicitées sur le plan *G* avec pour axe imaginaire *j.V* et pour axe réel *U*. Puisque le plan de *Nichols* est caractérisé par le couple d'axes (Gain, phase), il sera nécessaire de faire subir à ces équations une petite transformation. En manipulant les

relations entre (*U, V*) et (Gain, phase). Pour le traçage des contours d'amplitudes et phases on a suivi les étapes suivantes :

4.4.4.4. TRAÇAGE DES CONTOURS D'AMPLITUDE

Dans le calcul on utilise la valeur de *M* qui représente le module en BF, et pour le tracé on transforme cette valeur en décibels. (Exemple: un contour de 2db exige M= 1.059).

Soient $U_0 = \frac{M^2}{M^2 - 1}$ et $R = \frac{M}{1 - M^2}$,

$$Gain = 10.Log\left(U^2 + R^2 - (U + U_0)^2\right)$$

(4.18)

$$Phase = +Arctg\left(\frac{\sqrt{R^2 - (U + U_0)^2}}{U}\right) \quad (4.19)$$

$$Phase = -Arctg\left(\frac{\sqrt{R^2 - (U + U_0)^2}}{U}\right) \quad (4.20)$$

où $U \in [(-U_0 - R), (-U_0 + R)]$

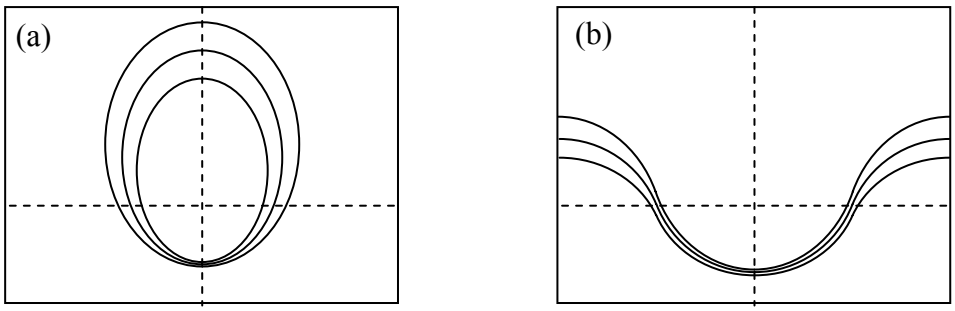


Figure 4.9 : Les contours de l'amplitude
a - Les contours fermés
b - Les contours ouverts

Pour tracer les contours fermés on utilise les valeurs de M positives. Pour les contours ouverts on utilise les valeurs de M négatives.

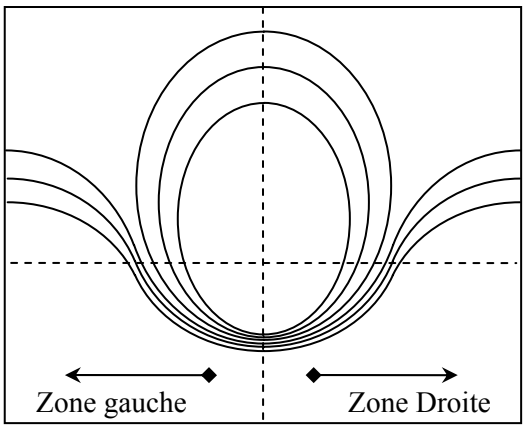


Figure 4.10 : La symétrie du contour d'amplitude

Nous observons une symétrie dans le traçage des contours d'amplitudes, c'est-à-dire que pour tracer le demi contour gauche on utilise les équations (4.18) et (4.19), et pour le traçage du demi contour droit les équations (4.18) et (4.20).

4.4.4.5. TRAÇAGE DES CONTOURS DE PHASE

Dans le calcul on utilise la valeur de *N* qui représente la phase en BF exprimée en radiants, et pour le tracé on transforme cette valeur en degrés. (Exemple: un contour de 30° exige N= 0.524 rad).

Soient $R = \dfrac{\sqrt{N^2+1}}{2N}$

$$Gain = 10.Log\left(U^2 + \left(\frac{1}{2N} + \sqrt{R^2 - (U+\frac{1}{2})^2} \right)^2 \right) \tag{4.21}$$

$$Phase = +Arctg\left(\frac{\sqrt{R^2 - (U+U_0)^2}}{U} \right) \tag{4.22}$$

$$Gain = 10.Log\left(U^2 + \left(\frac{1}{2N} - \sqrt{R^2 - (U+\frac{1}{2})^2} \right)^2 \right) \tag{4.23}$$

$$Phase = -Arctg\left(\frac{\sqrt{R^2 - (U+U_0)^2}}{U} \right) \tag{4.24}$$

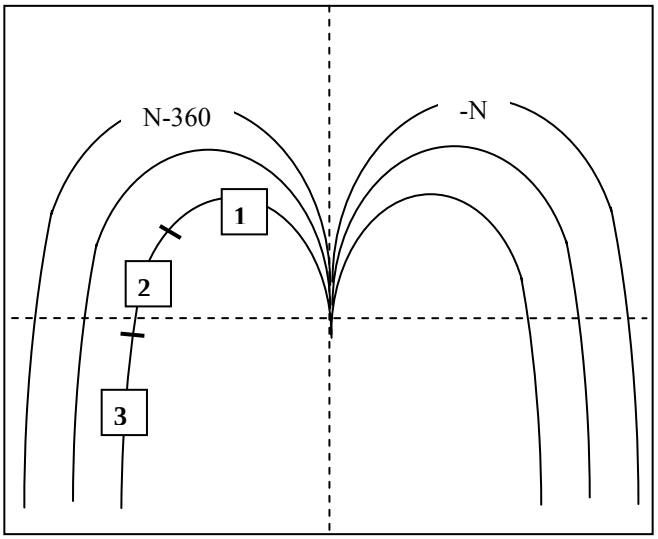


Figure 4.11 : Les contours de phase

On constate qu'il y a une symétrie dans le traçage des contours de phases, dont on admet la valeur de phase (-N) pour les contours droits et (N-360) pour les contours gauches, en sachant par ailleurs qu'un contour de phase se compose de trois portions :

Portion (1): qui utilise les équations (4.21) et (4.23) définies dans l'intervalle $U \in \left[\frac{-R}{2}, \frac{-1}{2} - R\right]$

Portion (2): qui utilise les équations (4.22) et (4.24) définies dans l'intervalle $U \in \left[\frac{-1}{2} - R, \frac{-1}{2} + R\right]$

Portion (3): qui utilise les équations (4.21) et (4.24) définies dans l'intervalle $U \in \left[\frac{-1}{2} + R, 0\right]$

Finalement pour obtenir l'abaque, on fait varier la variable U dans son domaine de définition avec un pas qui prend en compte des spécifications qui s'énoncent comme suit :

- Le tracé doit être le moins discrétisé possible.
- Le temps d'exécution doit être le plus petit possible.

4.4.4.6. MENU « ABAQUE »

4.4.4.6.1. LA PROCEDURE tracageAbaque ( )

Le tracé se fera en validant le menu « Abaque » avec une échelle standard. Pour cela on a réalisé les fonctions *TraceContourAmpl ( )*, *TraceContourPhase ( )* et ainsi que la procédure *TraceBlackBO ( )* pour obtenir le lieu. Si le traçage du lieu de Black est dans le domaine non visuel, un message s'affiche comme suit : "Le graphe est dans le domaine non visuel". La structure suivante montre le fonctionnement de cette procédure :

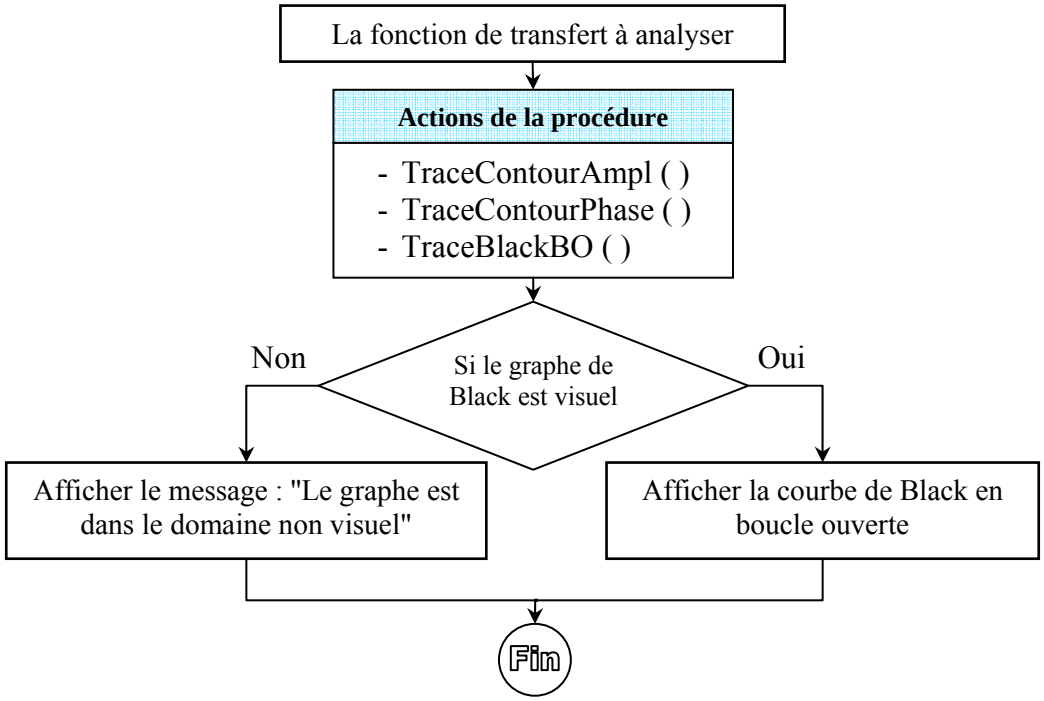


Figure 4.12: schéma fonctionnel expliquant le traçage de l'Abaque

On outre, on aura la possibilité d'utiliser l'option de zoom et de défilement en l'activant ou en le désactivant. Le manipulateur pourra également visualiser une partie du lieu de *Black* en fixant les pulsations de départ et d'arrivée.

4.4.5.LA METHODE D'EVANS

Puisque les performances d'un système asservi dépendent directement de son gain en boucle ouverte *K*, élément de réglage dont on dispose une fois déterminés les autres éléments du système, il est intéressant de connaître l'évolution dans le plan de *Laplace* des pôles de *F*(*p*) (la fonction de transfert en boucle fermée) lorsque le gain *K*, ou le coefficient *K* qui lui est proportionnel, varie de zéro à l'infini. Les pôles de *F*(*p*) décrivent alors un certain lieu géométrique, gradué en valeur de *K* et orienté dans le sens des valeurs croissantes de *K*. Ce lieu est appelé lieu d'*Evans*, ou lieu des racines (de l'équation caractéristique), ou lieu des pôles (de la fonction de transfert en boucle fermée du système).

Soit la fonction de transfert en boucle ouverte : $G(p)=\frac{N(p)}{D(p)}$

Sa fonction de transfert en boucle fermée s'écrira comme suit: $F(p)=\frac{k.N(p)}{D(p)+k.N(p)}$

Les pôles de la fonction de transfert en boucle fermée sont les racines de l'équation caractéristique $D(p)+k.N(p)=0$ qui permettront de construire le lieu des racines dans le plan complexe en fonction de la valeur de gain k.

$K=0 \Rightarrow D(p)=0$, Les racines de l'équation caractéristique sont les racines de D(p) qui sont également les pôles de la fonction de transfert en boucle ouverte.

$K \to \infty \ (K >> 0) \Rightarrow D(p) << k.N(p)$, c'est–à–dire que les racines de l'équation caractéristique tendent vers celles du polynôme N(p) qui sont les zéros de la fonction de transfert en boucle ouverte.

On conclu que lorsque k croit de zéro à l'infini, les lieux des pôles en boucle fermée partent des pôles de la fonction de transfert en boucle ouverte et approchent de ses zéros où ils aboutissent.

Lors de la programmation on a rencontré plusieurs problèmes dans le traçage du lieu d'*Evans*, bien que on va présenter le cas le plus important avec sa solution.

Les résultats obtenus par la méthode numérique "Laguerre" utilisée pour résoudre l'équation caractéristique sont classés d'une manière désordonnée, ce qui peut provoquer une permutation entre les lieus des pôles. Pour approfondir la compréhension du problème, on utilisera l'exemple du deuxième ordre illustré dans la figure ci-dessous :

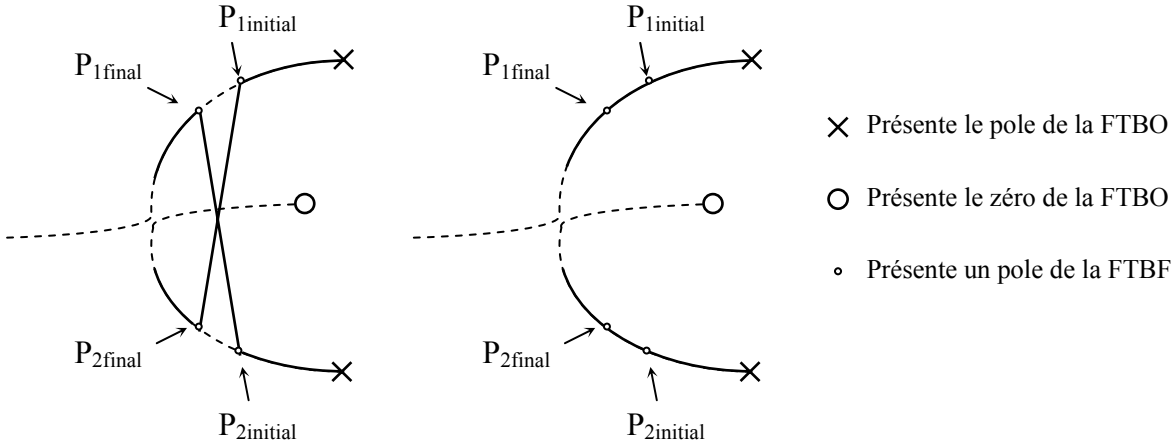


Figure 4.13 : Le traçage du lieu d'Evans
Gauche: tracé avant la solution de problème
Droite: tracé après la solution de problème

On a pour valeur de gain:

- $k=k_1$ les racines $P_{1initial}$ et $P_{2initial}$
- $k= k_2 > k_1$ les racines P_{1final} et P_{2final}

La question qui se pose est: quelles sont les conditions exigées pour que P_{1final} reste dans la série des pôles de $P_{1initial}$, et de même pour que P_{2final} reste dans la série des pôles de $P_{2initial}$?

La solution réside dans le calcul de la plus petite distance entre chaque pôle actuel et les pôles précédents. Cette distance trouvée va alors conditionner la relation entre le pole actuel et le pole précédent. A partir de cette exemple on trouve que :

- La distance entre P_{1final} et $P_{1initial}$ est petite par rapport à la distance entre P_{1final} et $P_{2initial}$

  $\Rightarrow P_{1final}$ sera intégré dans la série de $P_{1initial}$.

- La distance entre P_{2final} et $P_{2initial}$ est petite par rapport à la distance entre P_{2final} et $P_{1initial}$

  $\Rightarrow P_{2final}$ sera intégré dans la série de $P_{2initial}$.

La figure (4.13.gauche) illustre la solution trouvée.

NB : On applique la même technique pour les systèmes d'ordre supérieur.

4.4.5.2. MENU « EVANS »

Dans ce menu on a réalisé deux procédures essentielles. La première est la procédure *TraceEvans ( )* et la seconde est la procédure *Classement ( )* qui classe les pôles nouveaux aux séries qui leurs correspondent à savoir les pôles précédents.

Ci-dessous est représenté un algorithme qui résume de façon générale la procédure de traçage :

Début

Var : k, $k\min$, $k\max$, Pas : Réels;

$N(p)$, $D(p)$: Chaîne de caractères;

$VecPoles_i$, $VecPoles_f$: Vecteurs réels;

Lire : $k\min$, $k\max$, Pas, $N(p)$, $D(p)$;

$k = 0$;

$Equation = D(p) + k.N(p)$;

$VecPoles_i = Laguerre(Equation)$;

$k = k\min$;

Répéter

$k = k + Pas$;

$Equation = D(p) + k.N(p)$;

$VecPoles_f = \textbf{Laguerre}\ (Equation)$;

$\textbf{Classement}(VecPoles_f, VecPoles_i)$;

Tracer les pôles $(VecPoles_f)$;

$VecPoles_i = VecPoles_f$;

Jusqu'à $k \geq k\max$

Fin

4.4.5.2.1. SOUS MENU « OPTIONS »

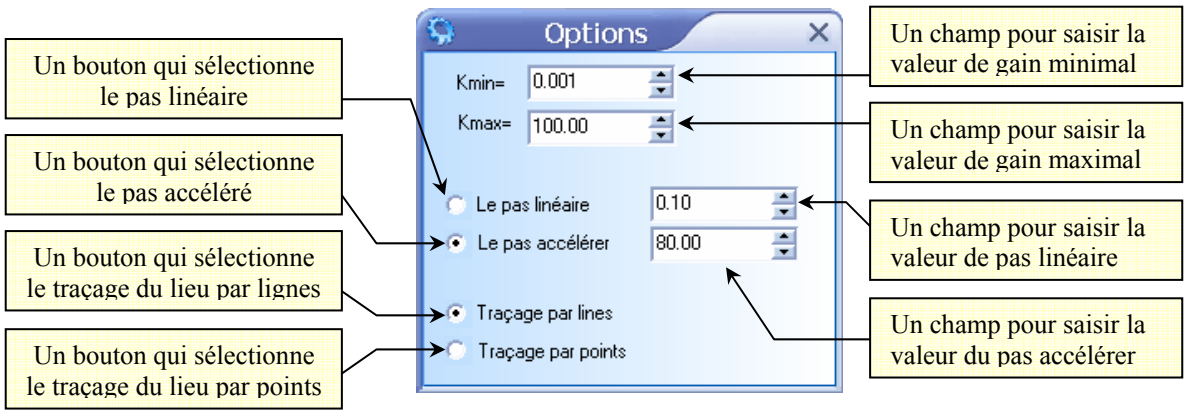


Figure 4.14: interface de la fenêtre Options d'Evans

4.4.6. LA REPRESENTATION TRIDIMENSIONNELLE DE BODE

L'objectif de ce menu est de présenter une vue en perspective de l'une des caractéristiques de la fonction de transfert (Gain, Amplitude ou la phase) en fonction de la variable de *Laplace* p qui définit un plan d'axes σ et $j\omega$ puisque $p = \sigma + j\omega$ comme l'indique la figure ci-dessous. On pourra ainsi représenter un graphe en trois dimensions avec une des caractéristiques de $G(p)$.

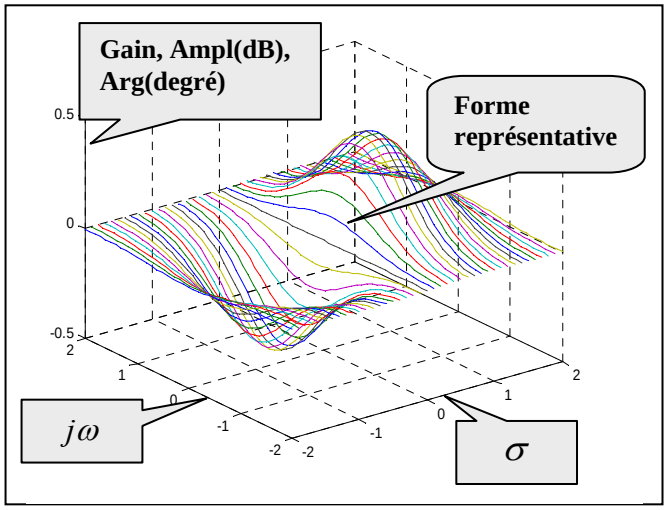


Figure 4.12: La visualisation 3D de graphes

4.4.6.3. INTERPRETATION DE LA COURBE TRIDIMENSIONNELLE

Dans la pratique, seul le graphe de gain en décibel nous permet d'expliquer le comportement fréquentiel du système. On a $G(p)=N(p)/D(p)$:

Si p est un pôle alors :
- $|G(pôle)| \to \infty$
- $|G(pôle)|$ (en dB) $\to \infty$

Fin Si

Si p est un Zéro alors :
- $|G(Zéro)| \to 0$
- $|G(Zéro)|$ (en dB) $\to -\infty$

Fin Si

Une vue tridimensionnelle de gain permettra de voir comment leur tapis sera soulevé et abaissé sachant qu'un pôle engendre un soulèvement du tapis et formera ce qu'on appellera un

pilier de la fonction et qu'un zéro engendre un rabaissement vers le bas créant ainsi un puit dans le tapis. Cette combinaison pilier-puit explique la forme plus ou moins désordonnée du gain. C'est cette même forme qui développera l'allure du diagramme de *Bode* qui comportera ou non des fréquences de résonance induites conditionnées par la présence de pôles à la proximité de l'axe des imaginaires. Plus le pôle lui est proche, plus l'amplitude de la résonance sera élevée ; et, à la limite, si le pole est imaginaire, la résonance sera confondue avec la source d'instabilité qui est le pole.

Le zéro, quand à lui, a la propriété d'absorber toute énergie qui est injectée à l'entrée du système pourvu que cette énergie soit principalement composée de fréquences voisines de la fréquence d'absorbation induite par le zéro à l'instar de la fréquence de résonance induite par le pôle.

En conclusion, les pôles et les zéros créent la forme et la vie du système.

4.4.6.4. MENU « BODE3DIMENSIONS »

4.4.6.4.1. LA PROCEDURE TraceModullog3D ( )

Début

Var : ω , σ : réels

Lire : $\omega(initial)$, $\omega(final)$, $\sigma(initial)$, $\sigma(final)$, $G(j\omega)$

$\omega = \omega(initial)$; $\sigma = \sigma(initial)$;

Répéter

$\omega = \omega + 0.1;$

Répéter

$\sigma = \sigma + 0.1;$

- Calcul **Modullog** $G(\sigma + j\omega)$

- Tracer le point (ω , σ , **Modullog** $G(\sigma + j\omega)$) ;

Jusqu'à $\sigma \geq \sigma(final)$;

Jusqu'à $\omega \geq \omega(final)$;

Fin

4.4.6.4.2. LA PROCEDURE TraceArgument3D ( )

Début

Var : ω , σ réels

Lire : $\omega(initial)$, $\omega(final)$, $\sigma(initial)$, $\sigma(final)$, $G(j\omega)$

$\omega = \omega(initial)$; $\sigma = \sigma(initial)$;

Répéter

$\omega = \omega + 0.1$;

Répéter

$\sigma = \sigma + 0.1$;

- Calcul **Argument** $G(\sigma + j\omega)$

- Tracer le point (ω , σ , **Argument** $G(\sigma + j\omega)$) ;

Jusqu'à $\sigma \geq \sigma(final)$;

Jusqu'à $\omega \geq \omega(final)$;

Fin

4.4.6.4.3. LA PROCEDURE TraceModule3D ( )

Début

Var : ω , σ réels

Lire : $\omega(initial)$, $\omega(final)$, $\sigma(initial)$, $\sigma(final)$, $G(j\omega)$

$\omega = \omega(initial)$; $\sigma = \sigma(initial)$;

Répéter

$\omega = \omega + 0.1$;

Répéter

$\sigma = \sigma + 0.1$;

- Calcul **Module** $G(\sigma + j\omega)$

- Tracer le point (ω , σ , **Module** $G(\sigma + j\omega)$) ;

Jusqu'à $\sigma \geq \sigma(final)$;

Jusqu'à $\omega \geq \omega(final)$;

Fin

4.5. CONCLUSION

L'élaboration du lieu d'*Evans* ainsi que dans une moindre mesure, les abaques de *Black-Nichols* nous ont donné du fil à retordre, vu la difficulté à détecter les racines complexes d'un polynôme et de la méthode de traçage de l'abaque.

On a ainsi essayé de compléter la méthode de Laguerre par l'apport de procédures secondaires très utiles telles que le pas accéléré, qui s'est avéré à l'usage être un outil incontrôlable dans le tracé des courbes sur ordinateur. Toutefois, il peut subsister des cas où la méthode se montrera divergente.

CHAPITRE V

PERFORMANCES D'UN SYSTEME ASSERVI LINEAIRE

5.1.INTRODUCTION

Malgré le grand intérêt théorique de disposer de critères précis pour juger les performances d'un système asservi, l'importance pratique de tels critères qui n'apparaissent qu'à un certain stade d'évolution de la technique. Pendant des années, le domaine d'automatique réalise pour résoudre des problèmes simples, d'excellents asservissements sans avoir à sa disposition de critères rationnels. Aujourd'hui il est impérieux de poser les problèmes de façon plus précise, à cause des perfectionnements de la technique et de l'application des notions relatives aux servomécanismes à des problèmes nouveaux.

Le présent chapitre fait le point d'un certain nombre de critères couramment utilisés en matière de systèmes asservis.

5.2.LA STABILITE

Au point de vue physique, on détermine la stabilité d'un système par sa réponse aux signaux d'entrée ou aux parasites. Intuitivement, un système stable est un système qui reste au repos à moins qu'on ne l'excite au moyen d'une source extérieure, et qui revient au repos dès que toutes les excitations cessent. Autrement dit, on peut définir avec précision la stabilité en relation avec la réponse du système à l'impulsion unité. Un système est dit stable si « la réponse à l'impulsion unité tend vers zéro quand le temps tend vers l'infini ».

Mathématiquement parlant, la condition nécessaire et suffisante de la stabilité d'un système asservi est que « tous les pôles de sa fonction de transfert en boucle fermée aient leur partie réelle négative ». Cette condition exige que les racines de l'équation caractéristique $1+G(p)=0$ ne possèdent pas de partie réelle positive ou nulle.

5.2.1.CRITERES DE STABILITE

Nous allons présenter quelques techniques pour aborder la stabilité d'un système. Il faut noter qu'il en existe bien d'autres…

Pour qu'un système linéaire soit stable au sens strict, il faut que sa fonction de transfert en boucle fermée ne comporte pas de pôle à partie réelle positive ou nulle.

$$F(p)=\frac{S(p)}{E(p)}=\frac{G(p)}{1+G(p)}$$

Alors : $$1+G(p)=0 \qquad (5.1)$$

Tous les pôles doivent donc être à partie réelle strictement négative.

Pour montrer si cette condition est vérifiée, on dispose de deux sortes de critères :

- Un critère algébrique, le critère de *Routh*, qui répond à la question en dispensant de l'effort d'une représentation graphique, mais sans permettre de chiffrer les marges de stabilité
- Des critères graphiques, qui présentent un avantage et un inconvénient exactement inversés par rapport au cas précédant.

5.2.1.1. CRITERE ALGEBRIQUE DE *ROUTH*

Le critère de *Routh* permet de déterminer rapidement la stabilité ou l'instabilité, à partir des coefficients de l'équation caractéristique. Parmi les critères algébriques celui-ci est souvent le plus utilisé dans le cas des systèmes linéaires.

Soit $F(p) = \dfrac{G(p)}{1+G(p)}$ la fonction de transfert de système en boucle fermée.

Les pôles de $F(p)$ sont les racines de l'équation $1+G(p)=0$

Un examen assez simple de $1+G(p)=0$ permet de savoir si certaines de ses racines sont à partie réelle positive ou nulle, rendant le système instable.

On écrit l'équation caractéristique sous forme polynomiale :

$$1+G(p) = a_n p^n + a_{n-1} p^{n-1} + ... + a_1 p + a_0 \qquad \text{avec } a_n > 0. \tag{5.2}$$

Le critère s'énonce alors de la façon suivante (nous ne le démontrerons pas) :

- **1er examen :** *'Condition suffisante pour les systèmes de premier et seconde ordre'*

Si les a_i ne sont pas tous de même signe ou si certains sont nuls, l'équation caractéristique a des racines à droite dans le plan complexe, donc à partie réelle positive. Le système est donc instable.

- **2ème examen :** *'Condition nécessaire et suffisante pour juger la stabilité'*

Si tous les a_i sont positifs, on ne peut connaître la place des pôles qu'après examen de la première colonne du tableau de *Routh* dont la construction est expliquée ci-après.

Les deux premières lignes du tableau sont écrites à l'aide des coefficients de l'équation caractéristique.

Les autres lignes sont formées de termes calculés à partir de ces coefficients.

						Calculs
On pose	P^n		a_n	a_{n-2}	a_{n-4}	a_{n-6}	$A_1 = \frac{a_{n-1}a_{n-2} - a_n a_{n-3}}{a_{n-1}}$
	P^{n-1}		a_{n-1}	a_{n-3}	a_{n-5}	a_{n-7}	$A_2 = \frac{a_{n-1}a_{n-4} - a_n a_{n-5}}{a_{n-1}}$
On détermine	P^n-2		A_1	A_2	A_3		$A_3 = \frac{a_{n-1}a_{n-6} - a_n a_{n-7}}{a_{n-1}}$
	P^{n-3}		B_1	B_2	B_3		$B_1 = \frac{A_1 a_{n-3} - a_{n-1}A_2}{A_1}$
	…		…	…	…		$B_2 = \frac{A_1 a_{n-5} - a_{n-1}A_3}{A_1}$
	P^2		M_1	M_2	M_3		……
	P^1		N_1	N_2	N_3		$C_1 = \frac{N_1 M_2 - M_1 N_2}{N_1}$
	P		C_1	C_2	C_3		
On analyse							

Figure 5.1 : tableau de *Routh*

Selon le critère de *Routh* la condition nécessaire et suffisante de stabilité est que tous les coefficients de la première colonne soient de même signe.

Le critère algébrique de *Routh* permet de savoir de façon simple et rapide si un système est stable ou non. Il nous renseigne sur la stabilité mais non sur la robustesse de cette stabilité. De plus sa mise en œuvre nécessite de connaître l'expression de la fonction de transfert.

5.2.1.2. CRITERE GRAPHIQUE : *R*EVERS

Dans ce cas, on s'intéresse à la fonction de transfert en boucle ouverte (FTBO) ce qui va nous permettre de voir si la fonction de transfert en boucle fermée (FTBF) est stable ou non.

On trace le lieu géométrique défini par la FTBO en prenant $p=j\omega$ et on observe la position de ce lieu par rapport au point –1.

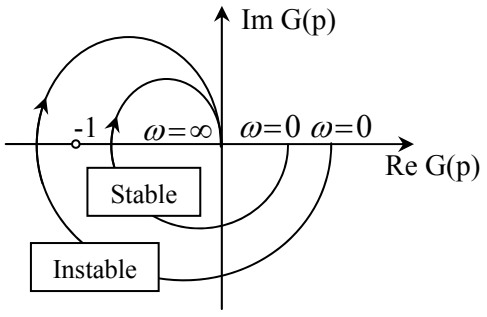


Figure 5.2 : Critère de Revers dans le plan complexe

Si, en parcourant le lieu de transfert dans le sens des ω croissants, et que le point critique (-1,0) reste toujours sur la gauche, le système sera stable.

Ce critère n'est pas rigoureux, mais est largement suffisant pour les fonctions de transfert ayant un sens physique. Cependant, si on veut être rigoureux, on peut utiliser le **critère de Nyquist**, Quoique ce dernier soit trop complexe pour le travail que nous cherchons à réaliser… Pour un système stable, les risques d'instabilité augmentent quand le lieu se rapproche de –1.

On peut donc considérer qu'un système est stable ou non et s'arrêter là. Néanmoins, cette distinction n'est pas suffisante. En effet, un système peut évoluer au cours du temps (dérive thermique, variation de la charge….), ce qui peut modifier son comportement et donc éventuellement conduire à un fonctionnement instable. C'est pour éviter ce genre d'inconvénient que l'on définit souvent une **marge de stabilité** pour les systèmes stables (robustesse de la stabilité). C'est le cas notamment de la marge de stabilité que nous présentons comme suit :

5.2.2. LA MARGE DE STABILITE

Il est essentiel de bien voir la différence entre la stabilité proprement dite et la marge de stabilité (marge de gain et phase), où la stabilité est une condition impérative, et la marge de stabilité est essentiellement une sécurité, que l'on prend en cas, par exemple, de petites variations de réglage, ou de retards dont on n'a pas tenu compte dans les équations. C'est aussi un critère de performance dans la mesure où cela garantit en général un bon amortissement.

5.2.2.1. MARGE DE GAIN : MG

La marge de gain est une mesure de la stabilité relative. On appelle marge de gain l'inverse du module en décibel de la fonction de transfert en boucle ouverte, prise pour les fréquences $\omega_{\pi i}$ correspondant à un angle de phase de -180 degrés. C'est-à-dire,

$$\boxed{MG_i \equiv 20\log\frac{1}{\left|G(j\omega_{\pi_i})\right|} = -20\log\left|G(j\omega_{\pi_i})\right| \qquad [décibels]} \tag{5.3}$$

Où $\arg G(j\omega_\pi) = -180$ dégrés = -π radians et ω_π se nomme *fréquence d'inversion* de la phase.

5.2.2.2. MARGE DE PHASE : MP

La marge de phase MP, est une mesure de la stabilité relative. On appelle marge de phase l'angle de la fonction de transfert en boucle ouverte pour ω_{Ci} correspondant à un gain unité augmenté de 180 degrés. C'est-à-dire :

$$\boxed{MP_i = 180 + \arg G(j\omega_{Ci}) \quad [\text{degrés}]} \tag{5.4}$$

Où $\left|G(j\omega_{Ci})\right|=1$ et ω_C se nomme fréquence de coupure.

On peut représenter graphiquement ces marges de stabilité dans les diagrammes fréquentiels :

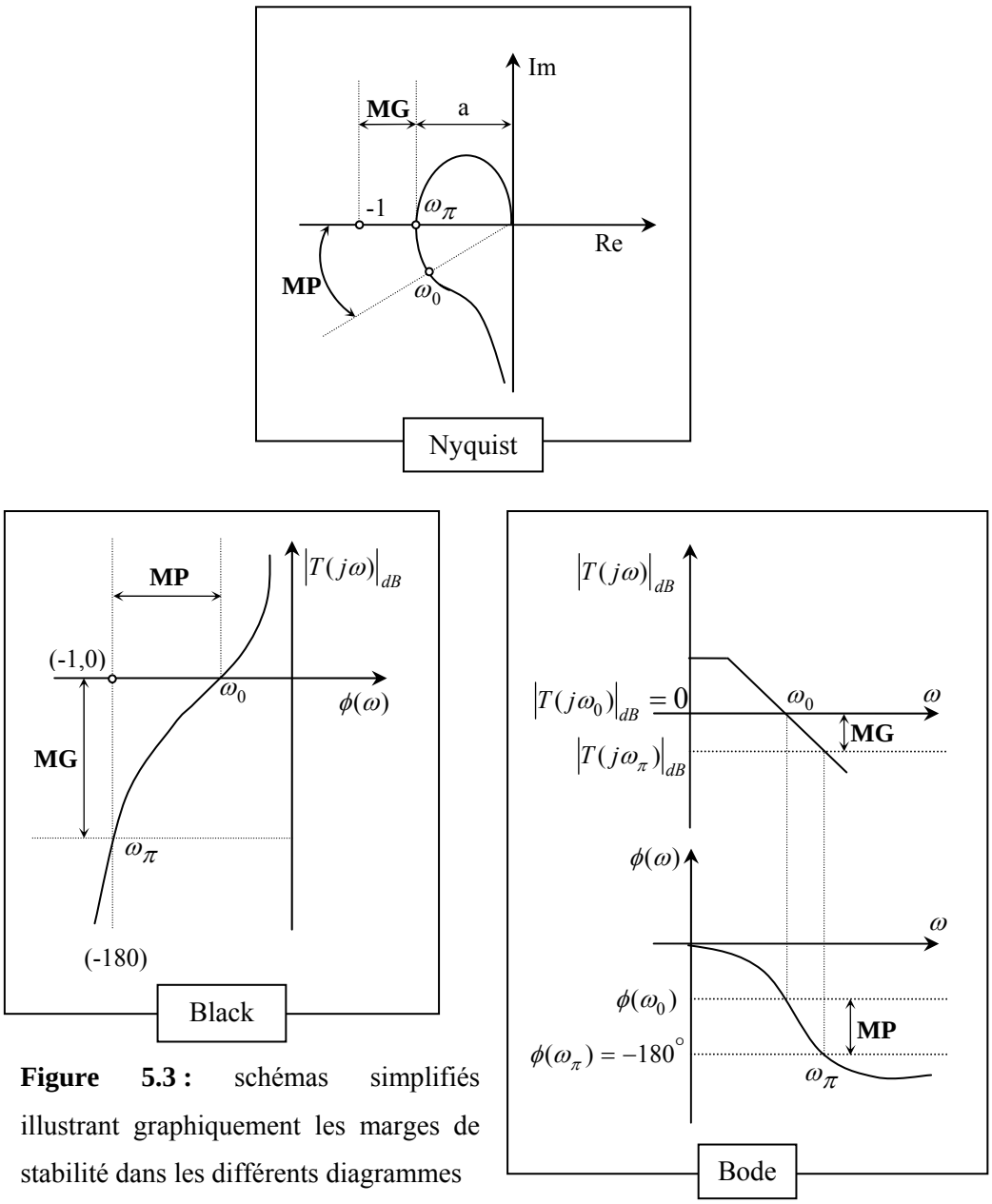


Figure 5.3 : schémas simplifiés illustrant graphiquement les marges de stabilité dans les différents diagrammes

5.3.LA PRECISION

La précision est l'aptitude d'un système à atteindre la valeur visée. Elle est caractérisée par l'écart entre la valeur attendue et la valeur effectivement atteinte par la grandeur de sortie :

$$\varepsilon(t)=Y_d(t)-Y(t) \tag{5.5}$$

Où $Y_d(t)$ valeur désirée, Y(t) valeur réelle.

Par définition, le système sera d'autant plus précis que son signal d'écart ou d'erreur $\varepsilon(t)$ est faible.

En général, on distingue deux types de précision. L'une est la précision statique (erreur en régime permanent) qui est égal à : $\lim_{t\to\infty} \varepsilon(t) = \varepsilon(\infty)$ et l'autre c'est la précision dynamique dont l'erreur est déterminée pour une valeur de temps donnée (définie).

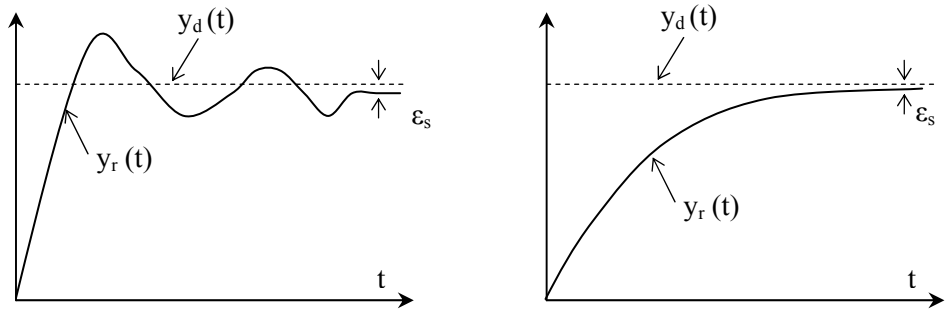


Figure 5.4 : Schéma simplifié qui représente la précision statique

Nous nous intéresserons, dans ce qui suit, à l'erreur statique.

5.3.1. ERREUR EN REGIME PERMANENT

5.3.1.1. L'ERREUR EN BOUCLE OUVERTE

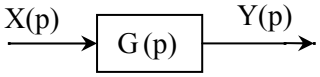


Figure 5.5 : schéma simplifié représentant le système en boucle ouverte

On a :

$$Y(p) = X(p).G(p) \Rightarrow E(p) = X(p) - Y(p) = X(p)\big(1 - G(p)\big)$$

Et par application du théorème de la valeur finale on trouve :

$$\varepsilon(\infty) = \lim_{t\to\infty} \varepsilon(t) = \lim_{p\to 0} p.E(p)$$

On trouve :

$$\boxed{\varepsilon(\infty) = \lim_{p\to 0} p.X(p)\big(1 - G(p)\big)} \qquad (5.6)$$

5.3.1.2. L'ERREUR EN BOUCLE FERMEE : (A RETOUR UNITAIRE)

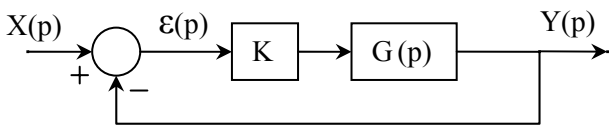


Figure 5.6 : Schéma simplifié représentant le système en boucle fermée

On sait que :

$$Y(p) = \frac{K.G(p)}{1+K.G(p)} X(p)$$

$$E(p) = X(p) - Y(p)$$

$$E(p) = X(p)\left(1 - \frac{K.G(p)}{1+K.G(p)}\right)$$

$$E(p) = X(p)/(1+K.G(p))$$

Et par application du théorème de la valeur finale on trouve :

$$\boxed{\varepsilon(\infty) = \lim_{p\to 0} p.X(p)\frac{1}{(1+K.G(p))}} \quad (5.7)$$

5.3.2. INFLUENCE DE L'ENTREE SUR L'ERREUR STATIQUE

- **Signal en échelon :** l'erreur est appelée erreur de position et est notée ε_P ;

On définit la constante d'erreur de position (ou d'échelon) K_p d'un système de la manière suivante :

$$K_p \equiv \lim_{p\to 0} G(p) \quad (5.8)$$

- **Signal en rampe :** l'erreur est appelée erreur de traînage et est notée ε_t ;

On définit la constante d'erreur de vitesse K_v (ou de Rampe) d'un système stable comme suit :

$$K_V \equiv \lim_{p\to 0} pG(p) \quad (5.9)$$

- **Signal en parabole :** l'erreur est appelée erreur en accélération et est notée ε_a ;

On définit la constante d'erreur d'accélération K_a d'un système stable de la manière suivante :

$$K_a \equiv \lim_{p\to 0} p^2 G(p) \quad (5.10)$$

5.4. LE DILEMME STABILITE-PRECISION

Sachant que plus le gain en boucle ouverte est élevé, plus raide est l'asservissement, et meilleures sont les performances de l'asservissement aux basses fréquences, notamment sa précision en régime statique.

En conclusion, le choix du gain en boucle ouverte résulte d'un dilemme :

► ou bien on choisit K faible pour être tranquille du côté de la stabilité, mais l'asservissement est mou et peu précis.

► Ou bien, pour améliorer la précision statique, on raidit l'asservissement en augmentant K et on tombe alors sur le pompage.

C'est le classique dilemme stabilité-précision ou raideur-pompage, qui était connu des praticiens bien avant qu'on applique les notions de lieux de transfert aux systèmes asservis.

5.4.1. REGLAGE PRATIQUE DE K « CAS DU RETOUR UNITAIRE »

La manière de trancher ce dilemme est affaire de cas d'espèces. Toutefois dans la grande majorité des cas, on s'arrête au compromis consistant à choisir le gain en boucle ouverte qui donne pour le système en boucle fermée un facteur de résonance Q de 1.3 environ, c'est-à-dire 1.2<Q<1.5, où sa formule se résume comme suit : $Q = |F(j\omega)|_{\max} / |F(0)|$

Le gain en question, que nous noterons K_{opt} est parfois appelé gain optimal (sous entendu pour le compromis stabilité-précision). Rappelons que prescrire Q=1.3 revient à fixer une marge de stabilité correspondant aux ordres de grandeur habituellement admis (45 à 50° pour la marge de phase, 10 à 15 dB pour la marge de gain).

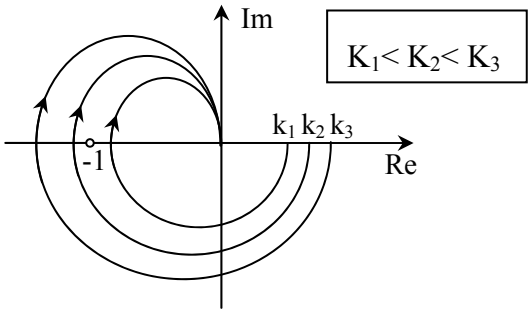


Figure 5.7 : Influence de k sur le lieu de Nyquist

5.5. LE DEPASSEMENT

Le dépassement est la différence maximum existant entre les solutions transitoires et les solutions de régime permanent pour un signal d'entrée en fonction en échelon unitaire. C'est une mesure de la stabilité relative et on l'exprime souvent en pourcentage de la valeur finale de la grandeur de sortie (de la solution de régime permanent).

On définit le premier dépassement par :

$$D_1(\%) = \frac{Y(t_1) - Y(\infty)}{Y(\infty)} \times 100\% = \frac{\Delta_1}{Y(\infty)} \times 100\% \qquad (5.11)$$

Avec $Y(\infty)$ la valeur finale de la sortie et $Y(t_1)$ la valeur de la sortie à l'instant du premier dépassement.
On définit de la même manière les autres dépassements.

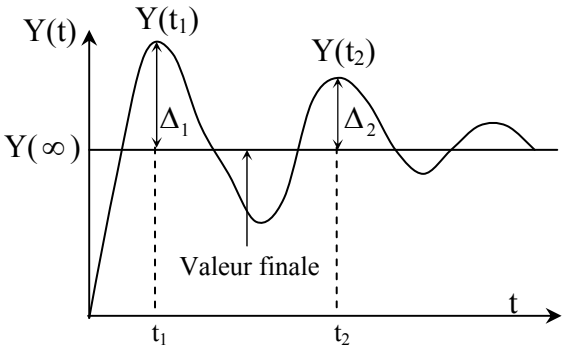


Figure 5.8 : Schéma illustrant le dépassement

5.6.L'AMORTISSEMENT

On caractérise le degré d'amortissement d'un régime transitoire par les dépassements successifs de la réponse unitaire. On considère principalement le premier dépassement que l'on appelle *rebondissement*. Pour que le rebondissement soit acceptable, on admet habituellement un dépassement de 30 à 50%.

Lors du passage d'une valeur à une autre de la grandeur de sortie, le comportement du système peut être tel que la réponse présente des oscillations.

Si ces oscillations sont trop prononcées, elles dénotent alors pour le système un manque de stabilité (réponse 1) (voir figure. 5.9).

La réponse 2 propose un bon compromis entre amortissement et rapidité pour un système oscillant.

Pour la réponse 3, un amortissement trop important conduit à une perte significative de rapidité pour le système.

Il faut compléter cette information par le dépassement qui caractérise l'amplitude des oscillations qui, pour certaines applications doit impérativement être nulle.

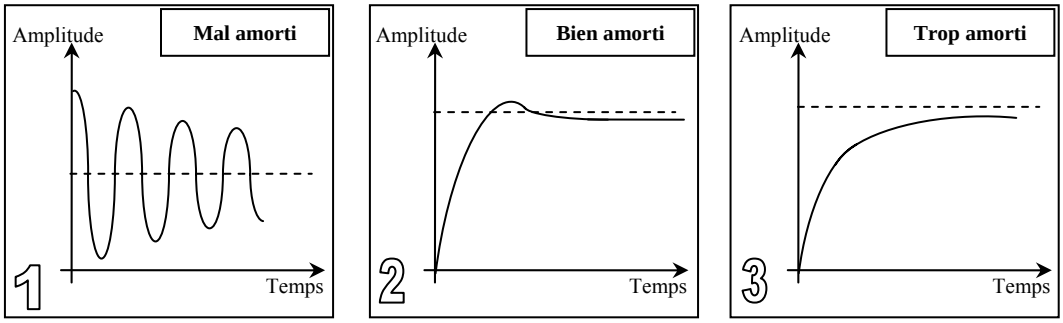


Figure 5.9 : schémas présentant les différents cas d'amortissement d'un système

5.7.LA RAPIDITE

Les critères d'évaluation de la rapidité d'un système asservi sont :

5.7.1.LE TEMPS DE MISE EN ROUTE Tm

Le temps de mise en route est défini comme étant le temps que met la réponse indicielle pour aller de 10 à 90 % de sa valeur finale.

5.7.2.LE TEMPS DE REPONSE Tr

On le définit comme le temps que met la réponse indicielle pour atteindre et rester à l'intérieur d'un pourcentage donné de sa valeur finale (±2 ou ±5%)

5.8.LA BANDE PASSANTE BP

La bande passante d'un système se définit comme étant l'intervalle des fréquences (du signal d'entrée) pour lequel le système a une réponse satisfaisante.

On détermine ce comportement satisfaisant en se reportant aux caractéristiques de ce système particulier.

On définit souvent la bande passante d'un système comme étant l'intervalle de fréquences pour lequel le gain ne diffère pas plus de -3dB de sa valeur en une fréquence déterminée. Pour plusieurs systèmes asservis, on prend pour cette fréquence zéro. La bande passante dans ce cas là est égale à la fréquence de coupure ω_c .

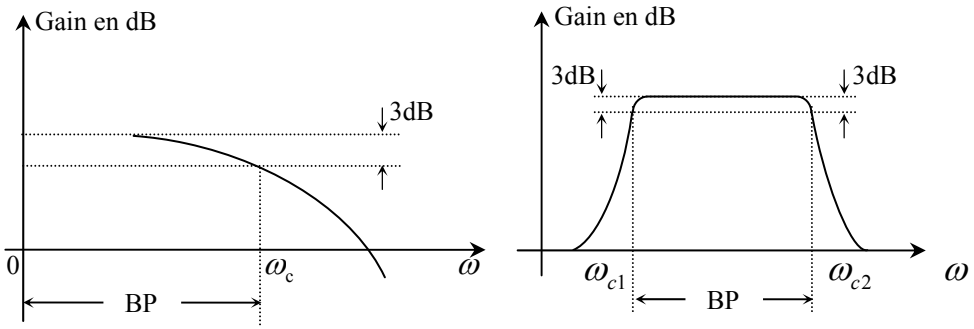


Figure 5.10 : Schéma illustre le calcul de la bande passante

5.9.ACUITE ET FREQUENCE DE RESONANCE

L'acuité A_R de résonance, qui est une mesure de la stabilité relative, est la valeur maximum du gain (module) de la réponse fréquentielle en boucle fermée. C'est-à-dire : $\boxed{A_R \equiv \max_{\omega} |F(j\omega)|}$

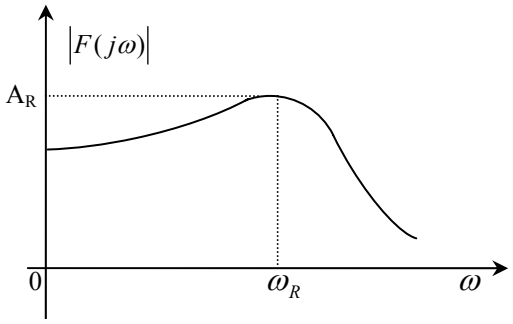


Figure 5.11 : Schéma illustrant l'acuité et la fréquence de résonance

La fréquence de résonance ω_R est la fréquence correspondant à A_R

5.10. MENU « PERFORMANCES »

5.10.1. LA PROCEDURE ErreurStatique ( )

5.10.1.1. LES FONCTIONS UTILISEES

- Fonction Limite ( )

5.10.1.2. DESCRIPTION DE LA PROCEDURE

Pour calculer la limite de l'expression d'erreur en boucle ouverte et en boucle fermée dans le domaine Laplacien on utilise la fonction *Limite ( )*.

Du point de vu programmation lorsque « P » tend vers zéro, on remplace le zéro par une valeur très petite ($\varepsilon = 10^{-300}$), puis en faisant varier le signal d'entrée du système X(P) on trouve :

- Pour un signal X(P) en échelon unitaire (X(p)=1/P), l'erreur calculée est appelée erreur de position.
- Pour un signal X(P) en rampe ($X(P)=1/P^2$), l'erreur calculée est appelée erreur de traînage.
- Pour un signal parabole ($X(P)=1/P^3$), l'erreur déterminée est appelée erreur en accélération.

On peut également calculer la précision d'un système en n'appliquant un signal d'entrée X(P) quelconque. Pour cette raison l'utilisateur peut saisir textuellement l'expression du signal d'entrée sur une cellule.

5.10.1.3. STRUCTURE FONCTIONNELLE

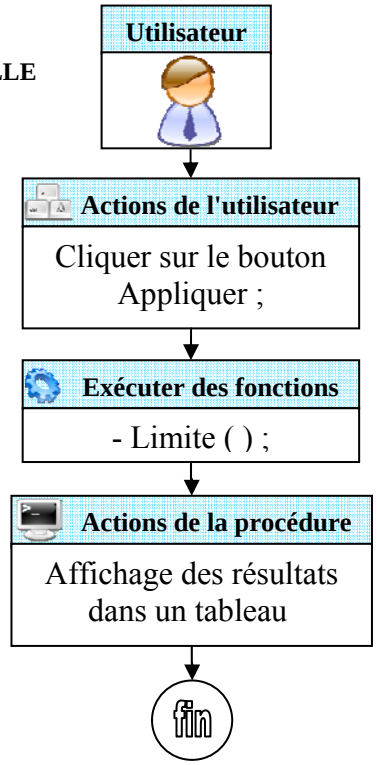


Figure 5.12 : Schéma fonctionnel illustrant la structure d'exécution de la procédure *ErreurStatique ( )*

5.10.2. LA PROCEDURE TESTDESTABILITE ( )

5.10.2.1. LES FONCTIONS UTILISEES

- Fonction TestDeStabilite ( )

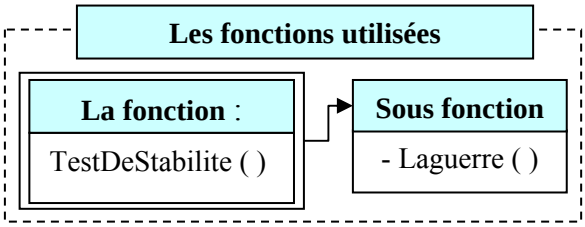


Figure 5.13 : schéma simplifié illustrant le fonctionnement de la procédure *TestDeStabilite ( )*

5.10.2.2. DESCRIPTION DE LA PROCEDURE

On premier lieu, on calcule les pôles de la fonction de transfert par application de la fonction *Laguerre ( )* puis on réalise le traitement sur les parties réelles et imaginaires de ces pôles suivant les conditions :

- Si tous les pôles ont des parties réelles négatives : le système est stable
- Si un seul pôle a une partie réelle positive : le système est instable
- Si un seul pôle a une partie réelle nulle : le système est marginalement stable.

Les résultats obtenus s'affichent sous forme de messages dans la barre de messages

5.10.3. LA PROCEDURE CALMARGESTABILITE ( )

5.10.3.1. LES FONCTIONS UTILISEES

- Fonction CalMargeGain ( )
- Fonction CalMargePhase ( )

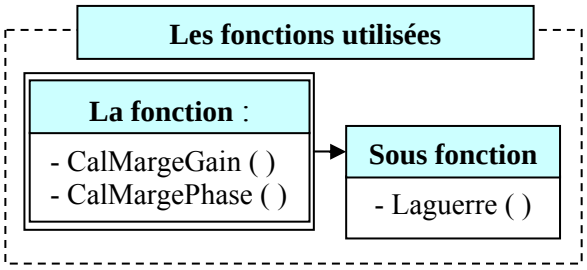


Figure 5.14 : schéma simplifié illustrant le fonctionnement de la procédure *CalMargeStabilite ( )*

5.10.3.2. DESCRIPTION DE LA PROCEDURE

A) Calcul des marges de gain

Pour déterminer les marges de gain de la fonction de transfert en boucle ouverte on a utilisé la fonction *CalMargeGain ( )*. Sur les principes de laquelle on a :

► Calculé les fréquences qui vérifient l'équation $ArgG(\omega_{\pi i}) = -180^{\circ}$

► Recherché parmi ces fréquences les fréquences réelles et positives $\omega_{\pi i}$

► Calculé les marges de gain suivant la formule : $MG_i = -20Log_{10}\left|G(j\omega_{\pi_i})\right|$

Notons bien que le calcul des racines $\omega_{\pi i}$ est effectué par application de la fonction *Laguerre ()*.

B) Calcul des marges de phase

Le calcul des marges de phase est réalisé par application de la fonction *CalMargePhase ( )* qui se traduit par les étapes suivantes :

► Détermination des fréquences en résolvant l'équation $\left|G(j\omega)\right|_{dB} = 0$ qui est équivalente à $\left|G(j\omega)\right| = 1$ c'est-à-dire :

$$\left|\boldsymbol{N}(\boldsymbol{j}\omega)\right| = \left|\boldsymbol{D}(\boldsymbol{j}\omega)\right| \Leftrightarrow \left(\left(\mathrm{Re}\,\boldsymbol{N}(\omega)\right)^2 + \left(\mathrm{Im}\,\boldsymbol{N}(\omega)\right)^2\right) - \left(\left(\mathrm{Re}\,\boldsymbol{D}(\omega)\right)^2 + \left(\mathrm{Im}\,\boldsymbol{D}(\omega)\right)^2\right) = 0$$

► Recherche parmi ces fréquences des fréquences réelles et positives $\omega_{c i}$

► Calcul des marges de phase suivant la formule suivante : $MP_i = 180^{\circ} + ArgG(j\omega_{c_i})$

5.10.4. SOUS MENU « DILEMME STABILITE-PRECISION »

5.10.4.1. LES FONCTIONS UTILISEES :

- Fonction Limite ( )
- Fonction TestDeStabilite ( )

5.10.4.2. DESCRIPTION

Dans ce sous menu on a représenté graphiquement les valeurs des erreurs statiques pour un signal d'entrée et la stabilité pour un intervalle de valeur de gain k avec un pas de "0.1". Cette représentation est réalisée pour la fonction de transfert en boucle ouverte et en boucle fermée à retour unitaire avec un gain k.

Bien entendu, l'erreur statique du système en boucle ouverte est indépendant de la valeur de gain, c'est-à-dire que sa représentation est une droite horizontale.

On a représenté le cas d'un système instable par une bande rouge et le cas de stabilité par une autre verte.

Les valeurs des erreurs statiques et le jugement de la stabilité sont obtenus par l'application de la procédure *ErreurStatique ( )* qui utilise la fonction *Limite ( )*, et la fonction *TestDeStabilite ( )*.

5.11. CONCLUSION

Après avoir analysé un système asservi linéaire (analyse fréquentielle et temporelle), il devient nécessaire de juger ses performances qui ont constitué l'objectif de cette partie du mémoire.

Les critères utilisés dans notre logiciel qui sont la précision et la stabilité absolue et relative sont considérés comme essentiels. Les autres critères plus difficiles à appliquer en regard du temps imparti à la réalisation du logiciel, ont été écartés délibérément.

CHAPITRE VI

SYNTHESE DE SYSTEMES ASSERVIS LINEAIRES

6.1.INTRODUCTION

Les systèmes asservis doivent répondre aux exigences technologiques telles que la précision, la rapidité et la stabilité. Ces qualités fondamentales sont difficiles à réaliser en même temps. En effet, c'est au dépend de la stabilité qu'il est possible d'améliorer la précision ; ce dilemme stabilité-précision a toujours été le problème essentiel de l'automaticien.

La synthèse des systèmes asservis consiste à déterminer une structure susceptible d'améliorer ce dilemme. On y arrive en adjoignant aux systèmes qui ne répondent pas aux indices de qualité un dispositif appelé « régulateur ».

Dans la plupart des dispositifs de régulation et dans beaucoup de cas d'applications, on utilise des régulateurs standard « régulateur P, I, PI, PD, PID », en raison de la simplicité de leur algorithme de réglage et des expériences acquises avec ce type de régulateur.

Ce chapitre est consacré à l'étude des différents types de régulateurs classiques analogiques qui ont pour rôle de modifier l'évolution temporelle et fréquentielle du signal de façon à ce que le système en boucle fermée ait les caractéristiques désirées.

6.2.DEFINITION D'UN REGULATEUR

Un correcteur est un assemblage d'objets servant à matérialiser une structure mathématique représentée par une équation différentielle. Ainsi la fonction remplie par un correcteur a pour effet d'introduire des pôles et des zéros dans une boucle, afin d'en modifier la dynamique globale décrite en termes de performances (Précision, stabilité, amortissement, robustesse, …).

6.3.SCHEMAS DE PRINCIPE DE REGULATION

Le choix entre une correction avec un régulateur placé en série ou en parallèle dépend essentiellement du problème lui-même ainsi que de la technologie utilisée. Lors de synthèse, il faut choisir habituellement le plus simple à réaliser.

En fonction du mode de connexion du régulateur au système asservi, il est possible de distinguer les régulateurs bronchés en cascade, ou en réaction :

6.3.1.REGULATION EN CASCADE

Dans ce type de régulation, le régulateur est bronché à la chaîne d'action après le comparateur (Fig 6.1) :

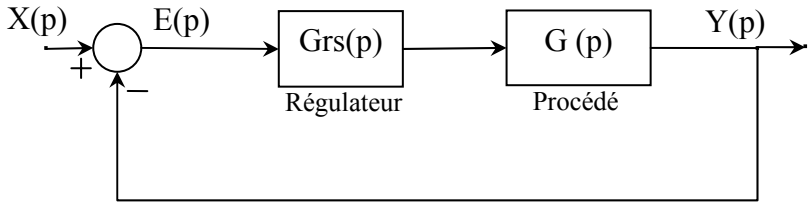


Figure 6.1 : Régulation en cascade

Où :

Grs(p) : la fonction de transfert du régulateur.

G(p) : la fonction de transfert de l'objet à régler.

En écrivant, la fonction de transfert du système en chaîne ouverte, on voit clairement une modification de la transmittance du système asservi corrigé :

$$Gt(p) = Grs(p).G(p) \tag{6.1}$$

6.3.2.REGULATION PAR CHAINE DE REACTION PRINCIPALE. (Fig.6.2)

Dans ce type de correction, le régulateur peut être installé comme suit :

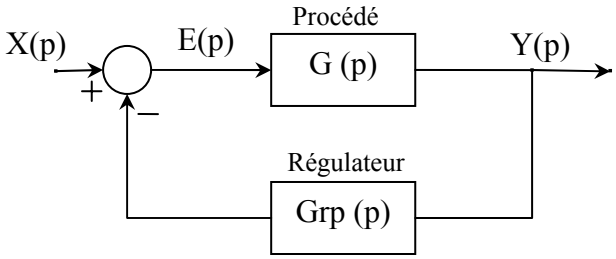


Figure 6.2 : Régulation en réaction principale

La fonction de transfert en boucle ouverte du système corrigé est égale à :

$$Gt(p) = \frac{G(p)}{1 + G(p).Grp(p)} \tag{6.2}$$

Où Grp(p) est la fonction de transfert du régulateur.

6.4.LES REGULATEURS CLASSIQUES ANALOGIQUES

On peut classer les régulateurs analogiques classiques selon leurs actions comme suit:

6.4.1.REGULATEUR PROPORTIONNEL P

L'action proportionnelle représente l'action minimale indispensable du réseau régulateur. Elle correspond à un gain constant K_p.

L'action proportionnelle est donnée par la loi de commande suivante :

$$U(t) = K_p e(t) \tag{6.3}$$

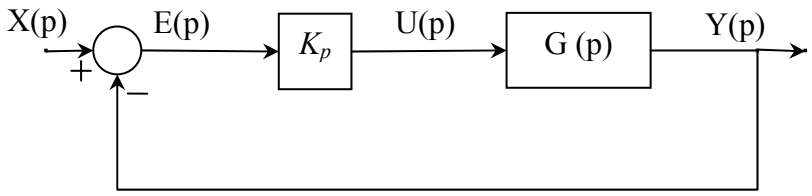


Figure 6.3 : Régulateur proportionnel

Ce contrôleur est caractérisé par :

- Un gain K_p inférieur à l'unité qui permet d'accroître la stabilité mais, en même temps engendre un décroissement de la précision.
- Un gain K_p supérieur à l'unité qui permet d'accroître la précision mais cette fois, engendre un décroissement de la stabilité.

6.4.2.REGULATEUR INTEGRAL I

L'action intégration est donnée par la loi de commande suivante :

$$U(t) = \frac{1}{T_i}\int_0^t e(t)dt \tag{6.4}$$

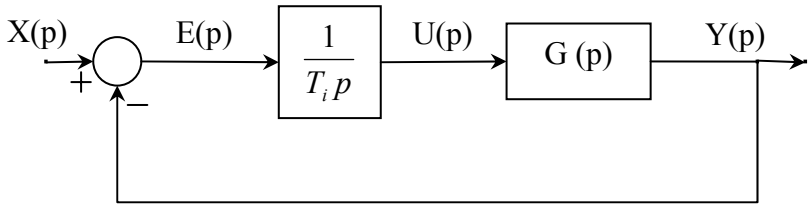


Figure 6.4 : Régulateur intégrateur

L'action intégrale augmente le gain et diminue la phase aux faibles fréquences. Si T_i augmente, la contribution de l'intégration diminue. Le système rejette plus lentement les perturbations.

Inversement si T_i diminue, le système réagit plus rapidement sans laisser au système le temps de démarrer progressivement, le dépassement sur la sortie s'amplifie mais revient à zéro en régime permanent.

6.4.3.REGULATEUR DERIVATEUR D

Cette action permet d'anticiper l'évolution du système et donc d'augmenter sa rapidité. Elle accroît le gain et la phase aux hautes fréquences. Généralement, les actions intégrales et dérivées ne s'emploient jamais seules mais toujours en combinaison avec l'action proportionnelle.

L'action dérivation est donnée par la loi de commande suivante :

$$U(t) = T_d \frac{de(t)}{dt} \tag{6.5}$$

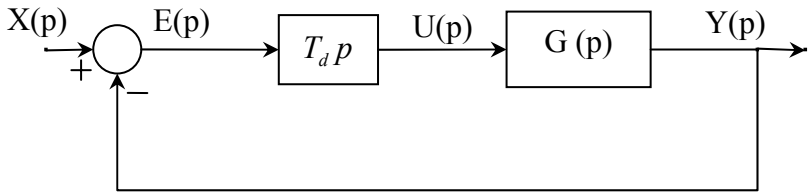


Figure 6.5 : Régulateur intégrateur

6.4.4.REGULATEUR PROPORTIONNEL-INTEGRAL PI

L'action de commande est donné par :

$$U(t) = K_P\left(\delta(t) + \frac{1}{T_i}\int_0^t e(t)dt\right) \tag{6.6}$$

D'où: $$Grs(p) = K_p\left(1 + \frac{1}{T_i p}\right) \tag{6.7}$$

Où K_p est la constante proportionnelle et T_i la constante de l'action intégrale, les constantes précédentes sont ajustables. La constante T_i ajuste l'action intégrale, tandis qu'un changement de K_p affecte l'action proportionnelle et l'action intégrale.

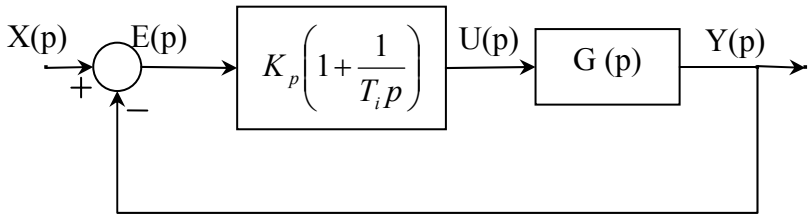


Figure 6.6 : Régulateur proportionnel-intégrateur (PI)

La commande PI réagit aux variations brusques de l'erreur et assure un rattrapage progressif de la consigne.

6.4.5.REGULATEUR PROPORTIONNEL-DERIVATEUR PD

L'action de commande est donnée par :

$$U(t) = K_P\left(e(t) + T_d \frac{de(t)}{dt} \right) \tag{6.8}$$

D'où : $$Grs(p) = K_p\left(1 + T_d\, p\right) \tag{6.9}$$

Où K_p est la constante proportionnelle et T_d la constante de l'action dérivée. Cette dernière est appelée quelque fois la vitesse de commande où l'amplitude de la commande X(p) est proportionnelle à la vitesse (taux de changement) du signal erreur.

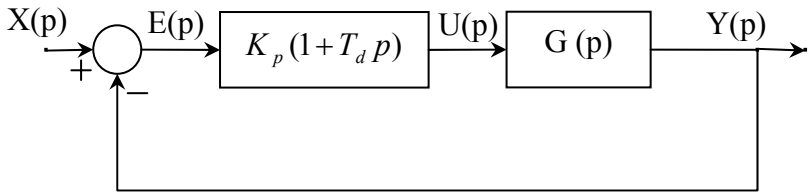


Figure 6.7 : Régulateur proportionnel-dérivateur (PD)

L'action dérivée permet une correction rapide de l'erreur, donc elle assure une amélioration de la stabilité et de la rapidité du système globalement. Il y a donc, en augmentant K_p un accroissement de la stabilité et de la précision du système en statique et en dynamique.

Malgré que l'action dérivée a l'avantage d'être anticipative, son inconvénient est qu'elle amplifie les signaux de bruit et peut causer des saturations au procédé.

Notons que l'action D ne doit être jamais utilisée toute seule, car elle est efficace seulement dans la partie transitoire de la réponse.

6.4.6.REGULATEUR PROPORTIONNEL-INTEGRAL-DERIVATEUR PID

L'action de commande est donnée par :

$$U(t) = K_P\left(\delta(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d \frac{de(t)}{dt} \right) \tag{6.10}$$

D'où : $$Grs(p) = K_p\left(1 + T_d\, p + \frac{1}{T_i\, p}\right) \tag{6.11}$$

K_p : est la constante proportionnelle,

T_i : la constante de temps de l'action intégrale,

T_d : la constante de temps de l'action dérivée.

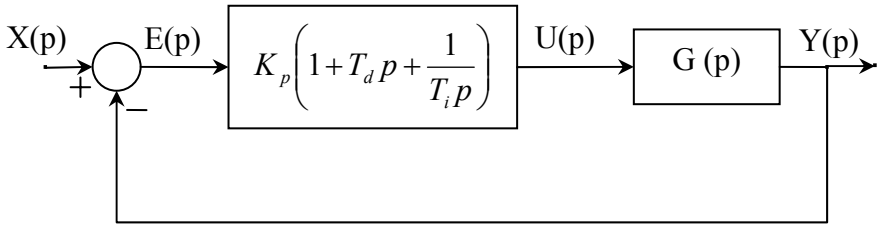


Figure 6.8 : Régulateur proportionnel-intégrateur-dérivateur (PID)

L'avantage de l'action PID est qu'elle assure l'avantage de chaque action : P, I, D (vitesse, précision et stabilité).

Ce contrôleur combine les avantages de PI et PD. Le terme de dérivation a une action stabilisatrice et permet d'améliorer la rapidité du système. Le terme intégral permet d'effacer l'influence des perturbations constantes. Lorsque le procédé possède lui-même une intégration, il apporte un degré de précision supplémentaire.

6.5.MENU « REGULATION »

6.5.1.LA PROCEDURE CalculFTRegulie ( )

L'idée de cette procédure c'est de calculer et afficher les fonctions de transfert des régulateurs suivant leurs types et la fonction totale du système corrigé en série ou bien en réaction ou les deux méthodes de régulation en même temps. Pour cela on a utilisé dans le calcul les fonctions *CalculGrs ( )*, *CalculGrp ( )* et *CalculGt ( ).*

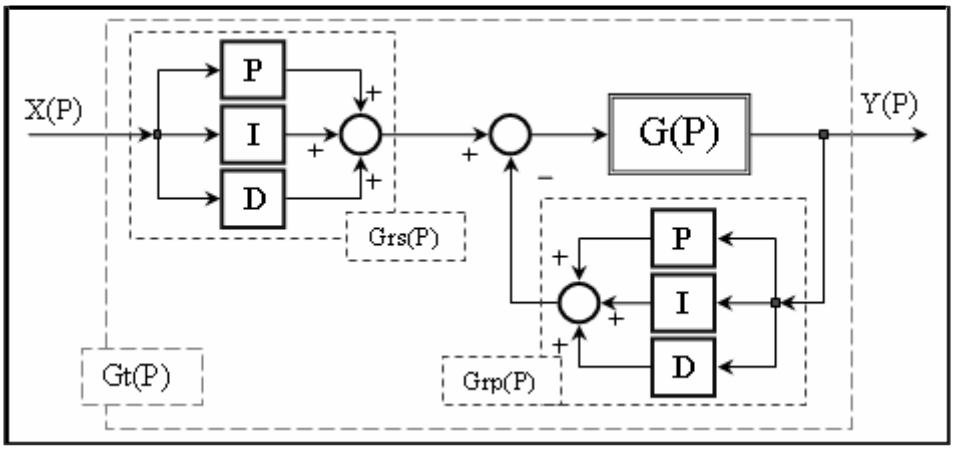


Figure 6.9 : schéma de régulation utilisé dans notre logiciel

Où :

Grs(p) : est la fonction de transfert du régulateur en série

Grp(p) : est la fonction de transfert du régulateur en réaction principale

Gt(p) : est la fonction de transfert total du système en boucle ouverte corrigé.

Il existe pour construire le schéma bloc de système avec régulation soixante douze (72) possibilités. Chaqu'une de ces possibilités présente un schéma qui contient un emplacement de chaque type de régulateur (P, I, D,…) suivant la méthode de régulation (cascade ou en réaction) choisie ou sélectionnée par l'utilisateur.

Pour mieux expliquer cette procédure, on l'illustre par le schéma fonctionnel suivant :

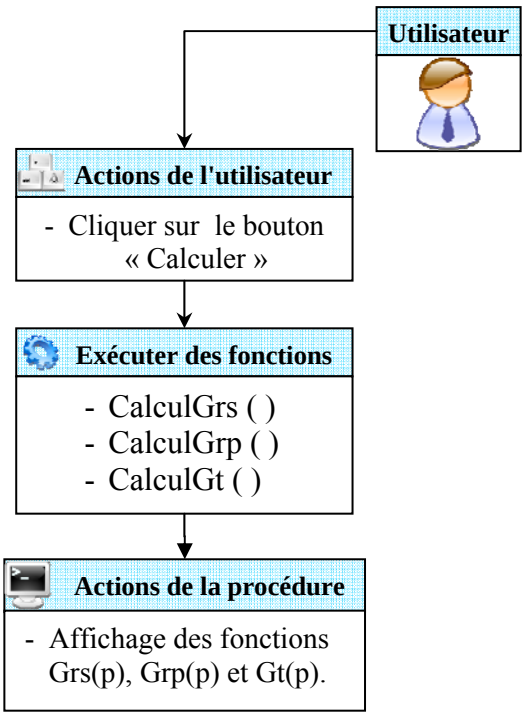


Figure 6.9 : schéma fonctionnel expliquant la procédure *CalculFTRegulie ( )*

6.6.CONCLUSION

On a essayé dans ce chapitre de mettre en évidence les principaux aspects concernant les régulateurs, ainsi que la procédure de conception des différentes techniques de régulation.

Parmi les techniques citées plus haut, la régulation de type PID est la plus répandue ; elle couvre une large gamme d'applications industrielles. Elle doit sa grande diffusion à la simplicité de sa synthèse et de sa mise en œuvre.

Le régulateur PID assure un excellent compromis entre la qualité de réglage et la simplicité d'emploi, grâce à quelques réglages simples qui permettent l'ajustement facile de ses paramètres aux caractéristiques du système à régulier.

Pour atteindre de bonnes performances pour la régulation du *procédé*, on a jugé utile d'exploiter la technique de régulation PID qui possède les propriétés requises pour une bonne correction.

CHAPITRE VII

EXEMPLE D'APPLICATION

7.1. INTRODUCTION

On a choisi d'étudier le groupe **générateur-moteur** (*Ward-Léonard*) de façon à obtenir des performances satisfaisantes dans le but de commander notre système en boucle fermée.

7.2. DEFINITION DU SYSTEME

Sachant que le système est caractérisé par la fonction de transfert comme suit :

7.2.1. LA REPRESENTATION PAR LA FONCTION DE TRANSFERT

$$G(P) = \frac{18520}{P^3+316*P^2+15785*P+94976}$$

7.2.2. LA REPRESENTATION PAR POLES ET ZEROS

Les Zéros

NB) Le système n'a aucun zéro

Les Poles	
P 1 =	-256
P 2 =	-53
P 3 =	-7

7.2.3. LA REPRESENTATION PAR L'ESPACE D'ETAT

Matrice A		
0	1	0
0	0	1
-94976	-15731	-316

Matrice B
0
0
1

Matrice C		
18520	0	0

Matrice D
0

7.2.3.1. L'OBSERVABILITE DU SYSTEME

La matice d'observabilité		
18520	0	0
0	18520	0
0	0	18520

Le déterminant de la matrice d'observabilité est :

6352182208000

Le système est donc :

Complètement observable

7.2.3.2. LA CONTROLABILITE DU SYSTEME

La matice controlabilité		
0	0	1
0	1	-316
1	-316	84125

Le déterminant de la matrice de contrôlabilité est :

-1

Le système est donc :

Complètement contrôlable

7.3. LES PERFORMANCES DU SYSTEME

7.3.1. LA STABILITE ABSOLUE

LE SYSTÈME EST STABLE

7.3.2. LA STABILITE RELATIVE

7.3.2.1. LES MARGES DE GAIN

LES MARGES DE GAIN [dB]			
w1	125.423283324908	MG1	48.4084899190465

7.3.2.2. LES MARGES DE PHASE

Il N'Y A PAS DES MARGES DE PHASE			

7.3.2.3. LE DILEMME STABILITE-PRECISION

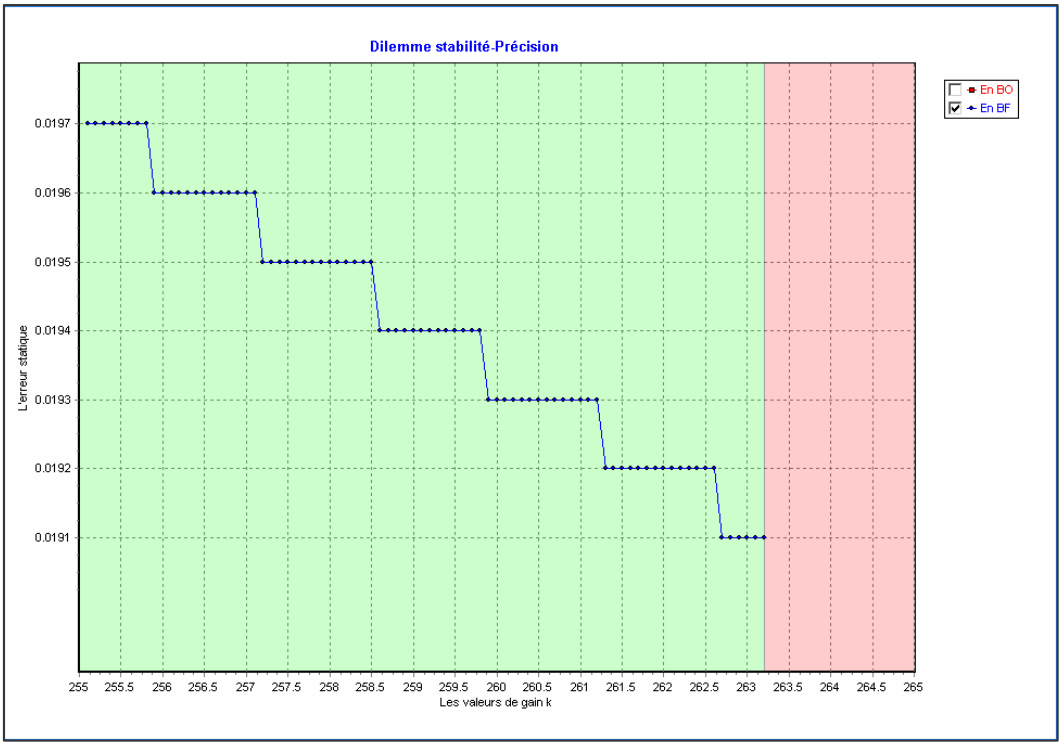


Figure 7.1 : présentant le dilemme stabilité-précision

7.4. L'ANALYSE FREQUENTIELLE

7.4.1. DIAGRAMME DE BODE

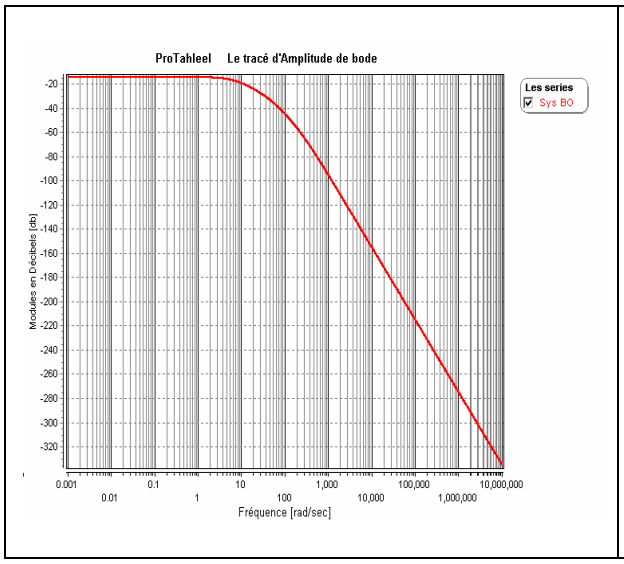


Figure 7.2 : diagramme d'amplitude

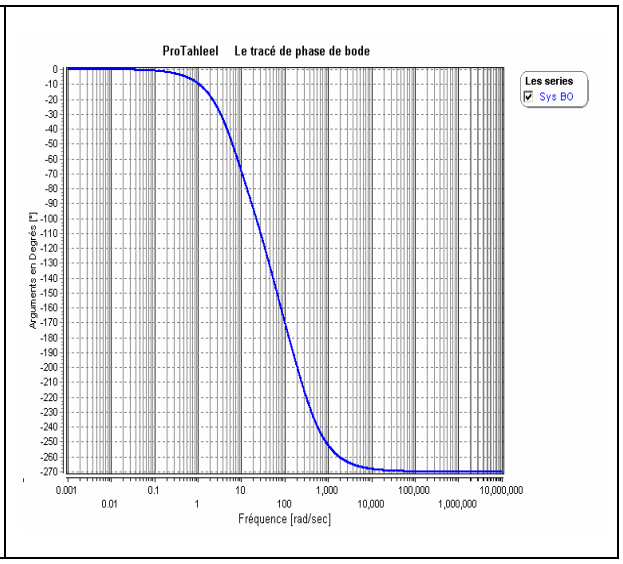


Figure 7.3 : diagramme de phase

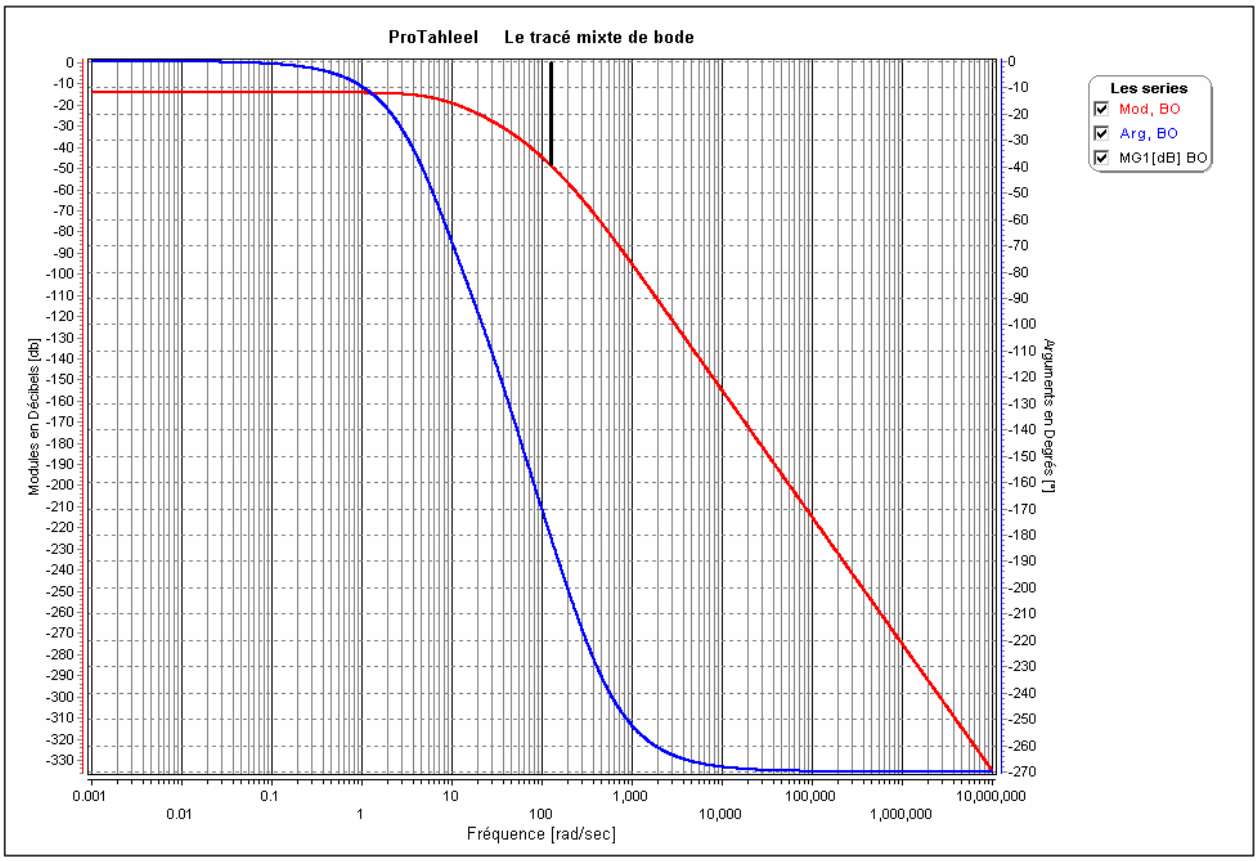


Figure 7.4 : Diagramme mixte de Bode

7.4.2. LIEU DE NYQUIST

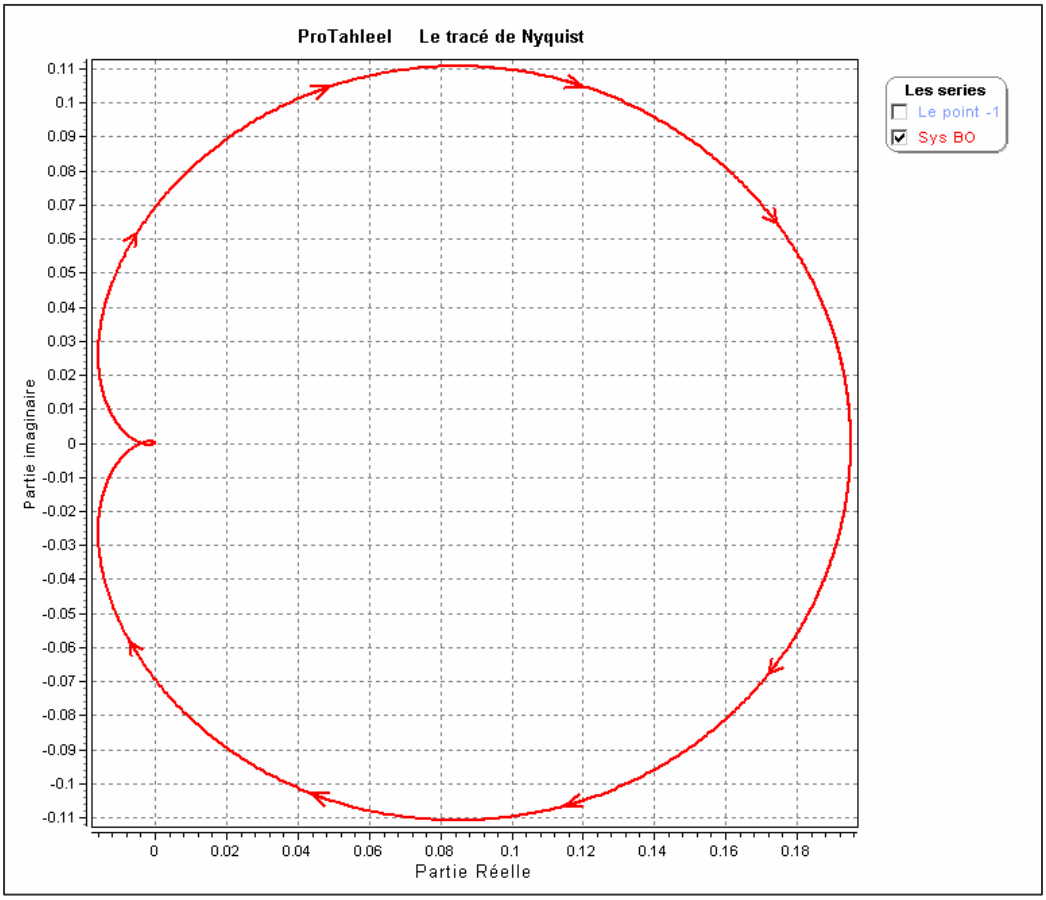


Figure 7.5 : Le lieu de Nyquist

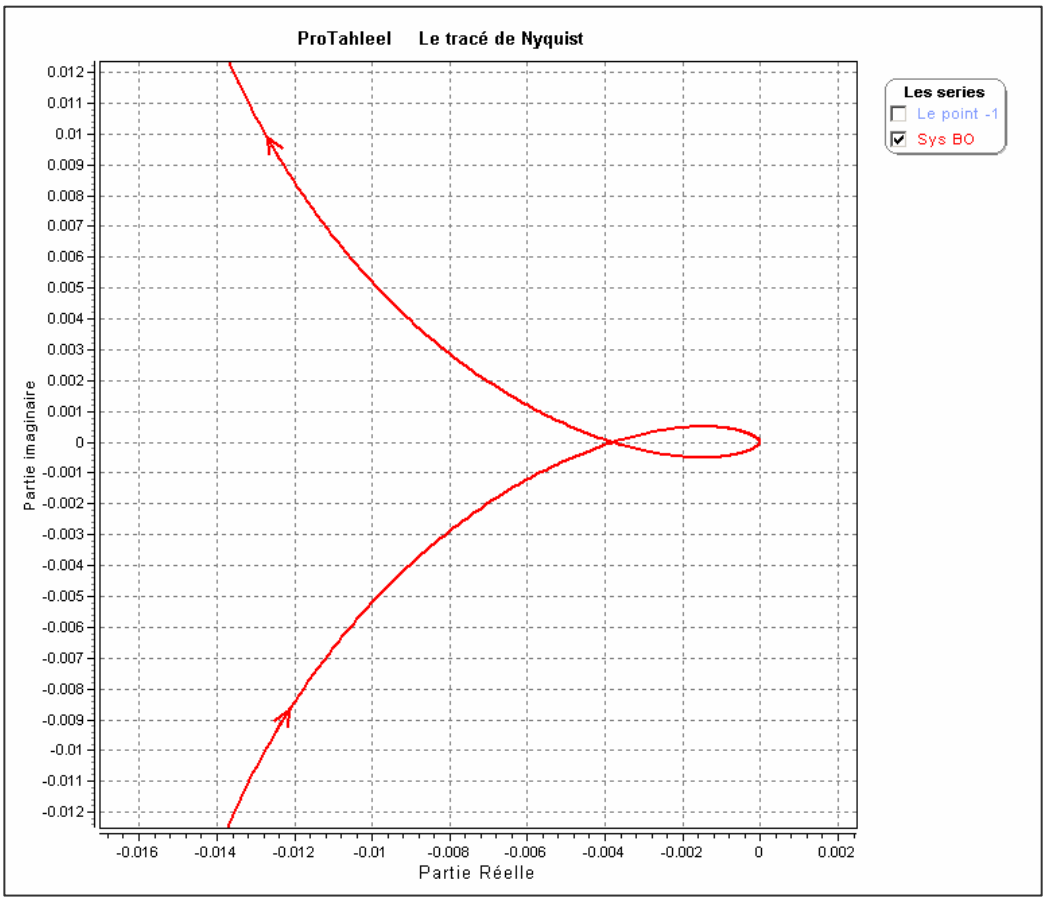


Figure 7.6 : zoom du lieu de Nyquist

7.4.3. LIEU DE BLACK-NICHOLS ET L'ABAQUE

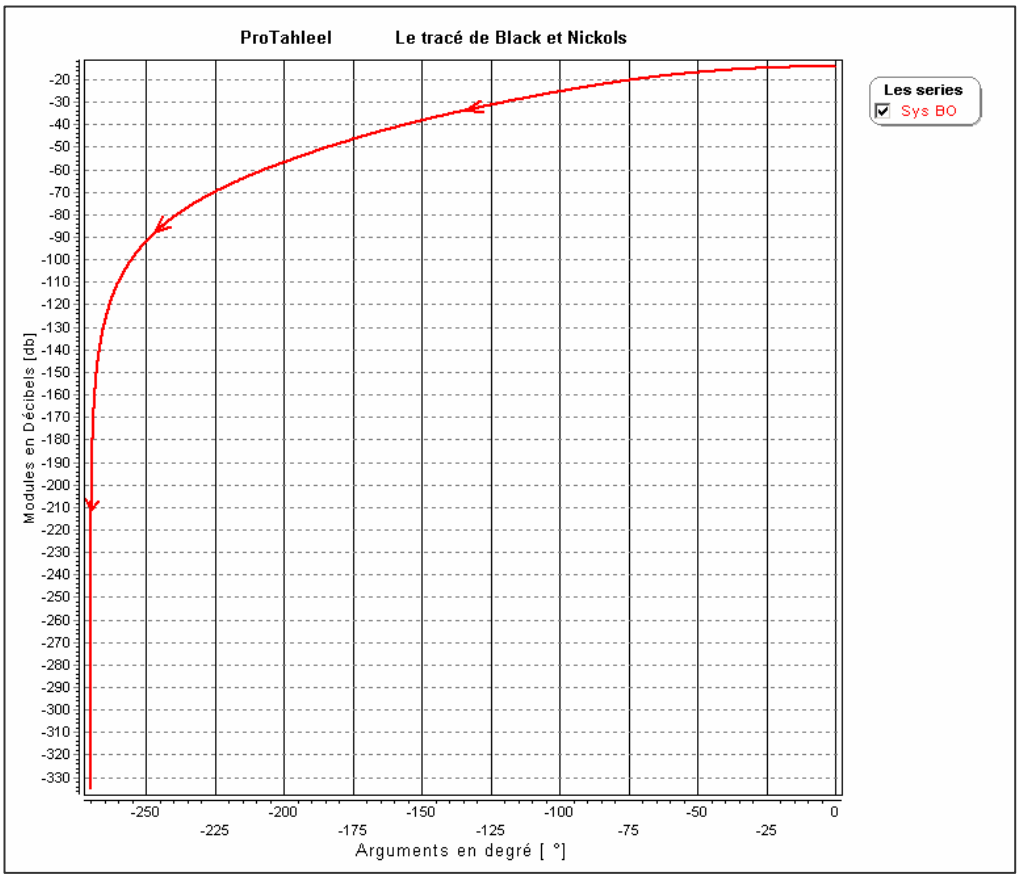


Figure 7.7 : Le lieu de Black-Nichols

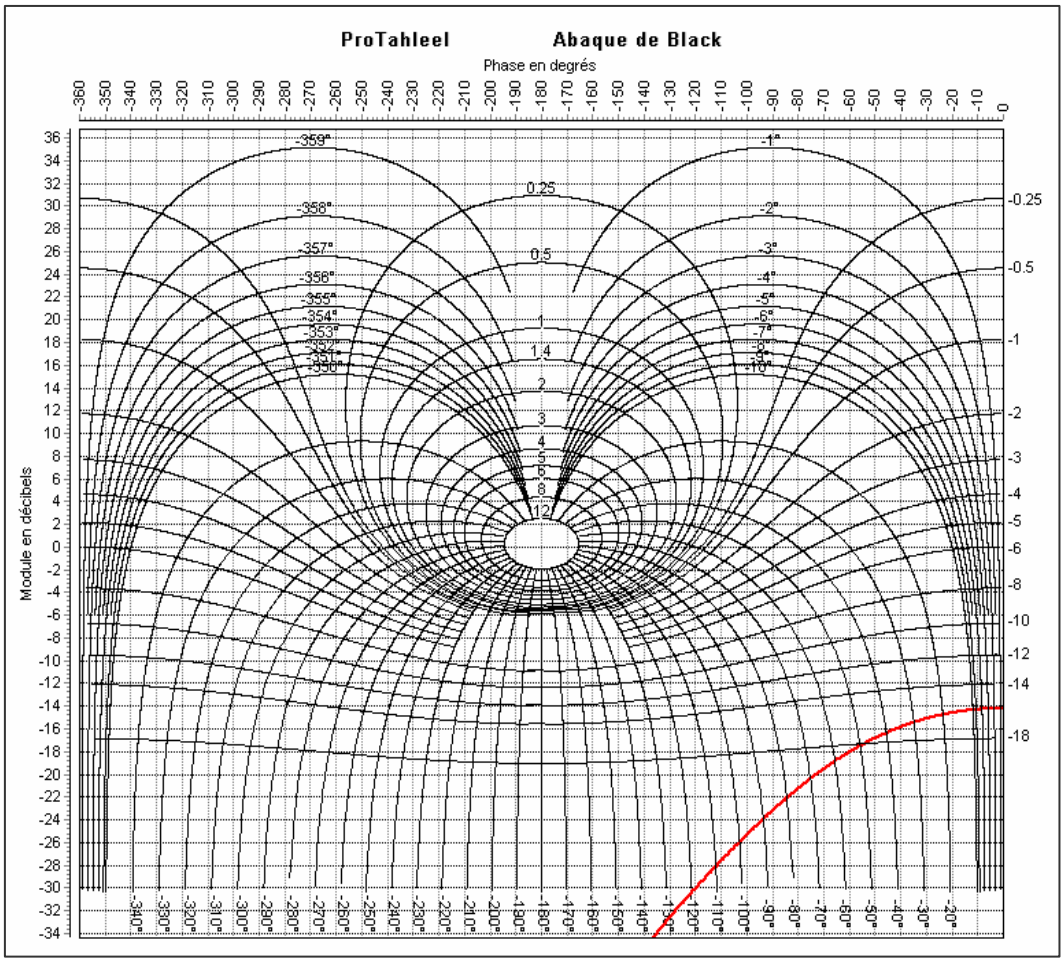


Figure 7.8 : L'abaque de Black

7.4.4. LA REPRESENTATION EN 3D DES TRACES DE BODE

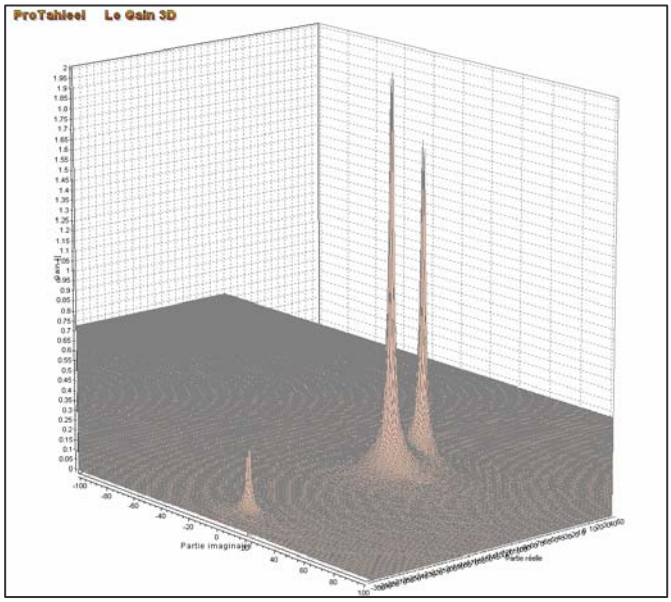


Figure 7.9 : Courbe de gain en 3D

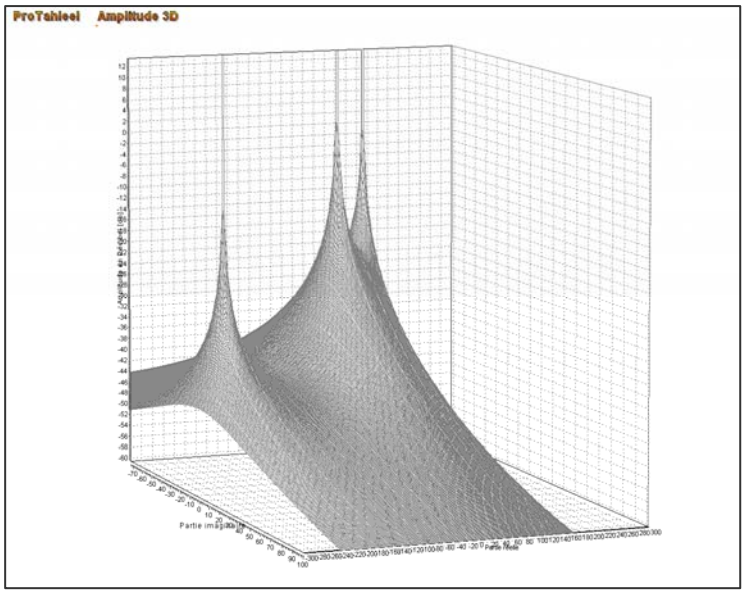


Figure 7.10 : Courbe d'amplitude en 3D

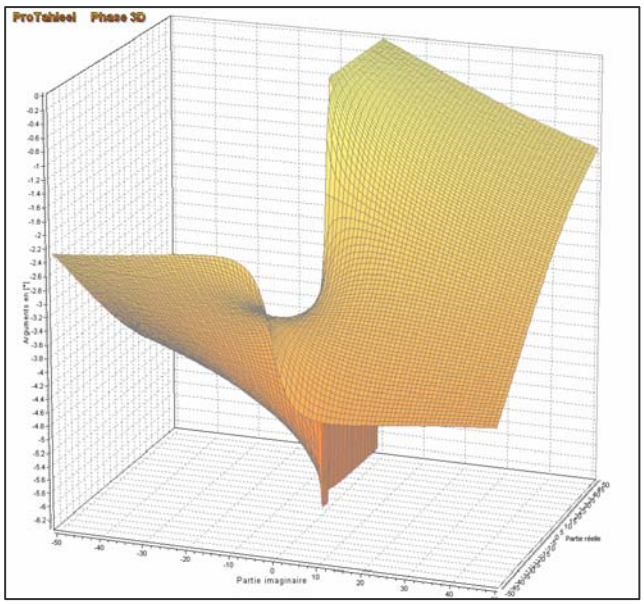


Figure 7.11 : Courbe de Phase en 3D

7.5. L'ANALYSE TEMPORELLE

Le schéma représentatif de la commande se présente comme suit :

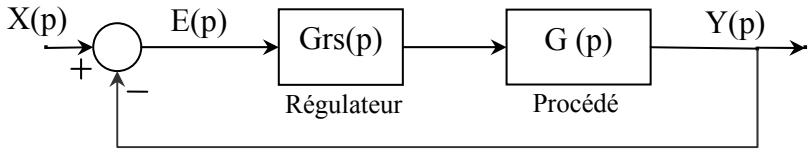


Figure 7.12 : Régulation en cascade

En boucle fermée, on constate que la réponse du système suit l'allure de l'entrée avec une erreur en régime permanent importante, de l'ordre de 80% pour une entrée en échelon, comme le montre la (Fig.7.14). La solution, qui vient directement à l'esprit pour éliminer cette erreur permanente, consiste à amplifier la sortie du système.

La détermination du k optimal doit tenir compte des considérations suivantes :

- Erreur en régime permanent faible.
- Régime transitoire marqué par un minimum de perturbations et avec un premier pic (dépassement) le plus atténué possible.
- Le système doit présenter une stabilité absolue et relative idéale.

D'après les différents tracés obtenus pour différentes valeurs du gain kp (Fig.7.14), on rempli le tableau (fig.7.13) par l'erreur statique, le dépassement et le temps de monté.

			Les performances du système		
		Paramètres du régulateur	**Erreur statique [%]**	**Dépassement[%]**	**Temps de monté [sec]**
Système en **BO**		———	80.50	———	———
Système en **BF**		———	83.68	———	———
TYPE DE REGULATEUR	P	Kp = 10	33.90	———	———
		Kp = 30	14.60	———	———
		Kp = 150	3.31	62.70	0.02
	PD	Kp = 150	3.30	7.90	0.011
		Td = 0.02			
	PID	Kp = 150	0	8.10	0.011
		Ti = 1			
		Td = 0.02			

Figure 7.13 : Tableau présentant quelques performances

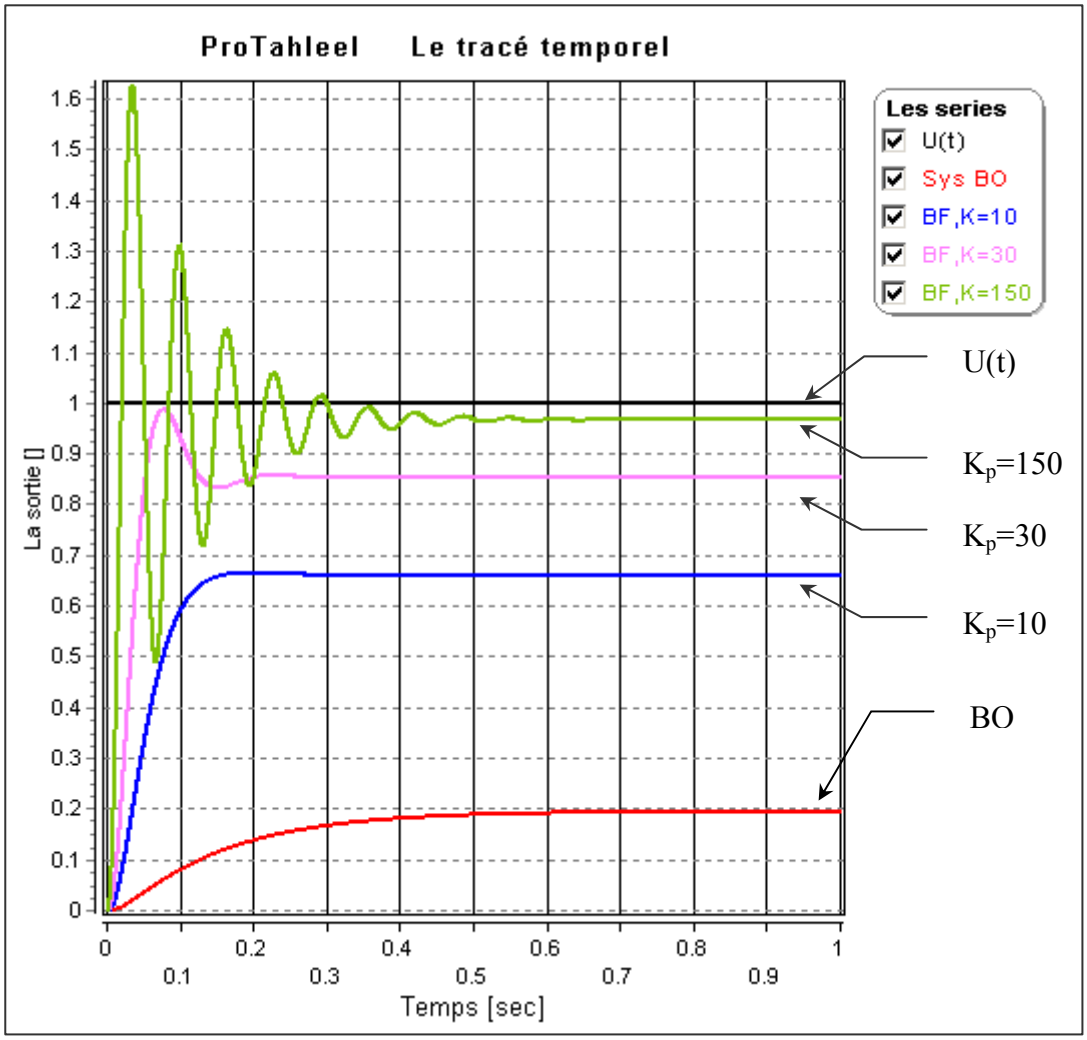


Figure 7.14 : Réponses indicielles pour les différentes valeurs de gain k_p

Sur la base de ces quelques valeurs de k, on a aperçu un dilemme de stabilité-précision. Pour cela on a réalisé d'autres types de régulation PD et PID, grâce à quelques réglages simples expérimentaux à l'aide de notre logiciel et qui permettent l'ajustement facile des paramètres de ces régulateurs.

A partir de l'observation de la table (Fig.7.13) et les traces (Fig.7.15) on a remarqué les résultats suivants :

- Le régulateur PD a amélioré les performances acquises en comparaison avec la régulation proportionnelle P.

- Le régulateur PID a augmenté la précision statique par rapport au régulateur PD.

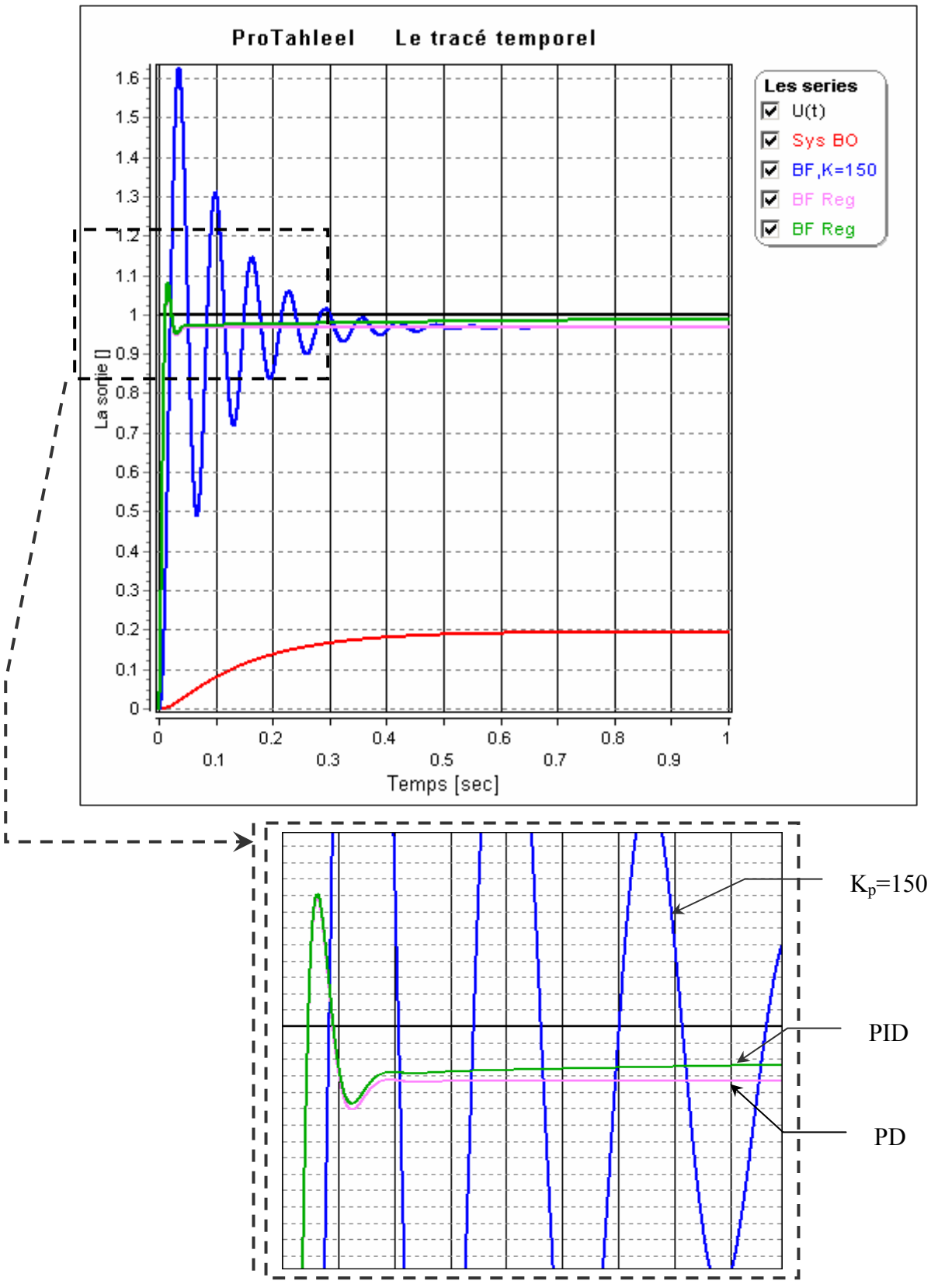


Figure 7.15 : Réponse indicielles avec des régulateur (P, PD, PID)

7.6. CONCLUSION

En conclusion, les performances du *procédé* sont satisfaisantes par la régulation qui été exploitée par le régulateur PID qui possède les propriétés requises pour une bonne correction.

CONCLUSION GENERALE

CONCLUSION GENERALE

Tout au long de notre travail notre souci constant a été de rechercher à travers les étapes de construction de notre projet les moyens les plus à même de rendre l'utilisation de ce logiciel la plus simple possible avec en perspective une performance satisfaisante. Cette étude montre combien un logiciel est indispensable surtout quand il peut répondre aux besoins des utilisateurs, en leur permettant de visualiser et imprimer les différents résultats nécessaires pour argumenter leur étude, et de faciliter ainsi dans une large mesure la compréhension de la théorie des systèmes.

Malgré les suggestions qui nous ont été faites en matière de méthodes de régulation en vue d'améliorer l'efficacité de notre logiciel, nous nous somme trouvé contraint, faute de temps à n'utilisé que la régulation en cascade et en réaction.

Ces remarques pourraient dans l'avenir constituer une base pour l'élaboration d'une nouvelle version de "ProTahleel " car il ne faut pas perdre de vue q'un logiciel est un outil informatique qui doit continuer à se développer avec le temps au risque de devenir inefficace.

En conclusion, il est à espérer que ce logiciel soit exploité au maximum de ses possibilités, que ce soit dans les séances de travaux pratiques ou encore dans les projets de recherche.

BIBLIOGRAPHIE

[1] J.J.DI STEFANO, A.R.STUBBERUD, I.J.WILLIAMS, « Systèmes asservis1, cours et problèmes », série Schum, Edition McGrow-Hill, Paris 1974.

[2] J.J.DI STEFANO, A.R.STUBBERUD, I.J. WILLIAMS, « Systèmes asservis2, cours et problèmes », série Schum, Edition McGrow-Hill, Paris 1975.

[3] Rachid IKNI, Lahcène BENBAOUCHE, « Asservissements linéaires continus », Edition OPU, Algérie 1990.

[4] Jean-Poul Houtier, Jean-Pierre Caron, «systèmes Automatique, commande des processus» , Tome2, Edition Ellipses, Paris, 1997

[5] F.de CARFOET, C.FOULARD, J.CALVET « Asservissements linéaires continus », Edition Dunod, Paris 1976.

[6] Jean-Charles.GILLE, Paul.DECAULNE, Marc.PELEGRIN, « Théorie et calcul des asservissements linéaires », Edition Dunod, Paris 1987.

[7] Jean-pierre DEMAILLY, « Analyse numérique et équations différentielles », Edition OPU, Algérie 1994.

[8] Patrick SIARRY, « Automatique de Base », Edition Berti, Algérie 1993.

[9] Zerouni MED LAID, MELGANI FARID, « Elaboration d'un logiciel d'analyse des systèmes continus », PFE d'ingénieur en électronique option contrôle, Batna 1994.

[10] Francis MILSANT, « Asservissement linéaires », Edition Eyrolles, Paris 1986.

[11] Francis MILSANT, « Cours d'Electrotechnique, Machines électrique », Edition Berti, Algérie 1993.

[12] Hiham BELAID, Schahrazed AOUBID, Samira BOUKHALFA, « Commande d'un processus physique par PC », PFE d'ingénieur en électronique, Batna 2001.

ANNEXES

- Annexe I
- Annexe II
- Annexe III

Annexe I

Présentation des menus du logiciel

❖ MENU PRINCIPAL

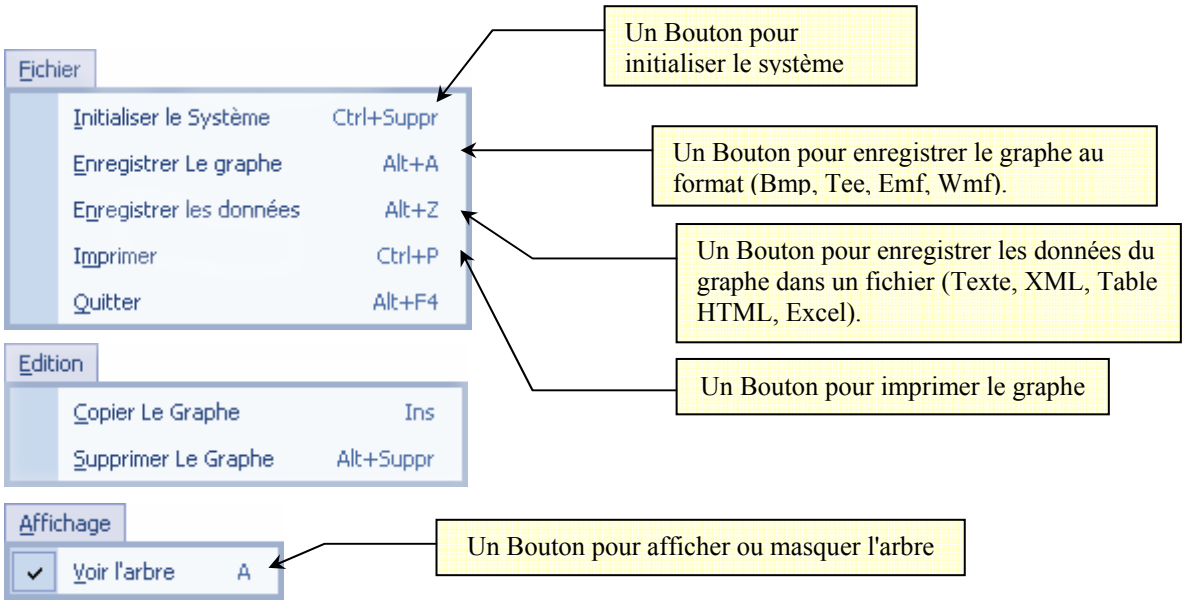
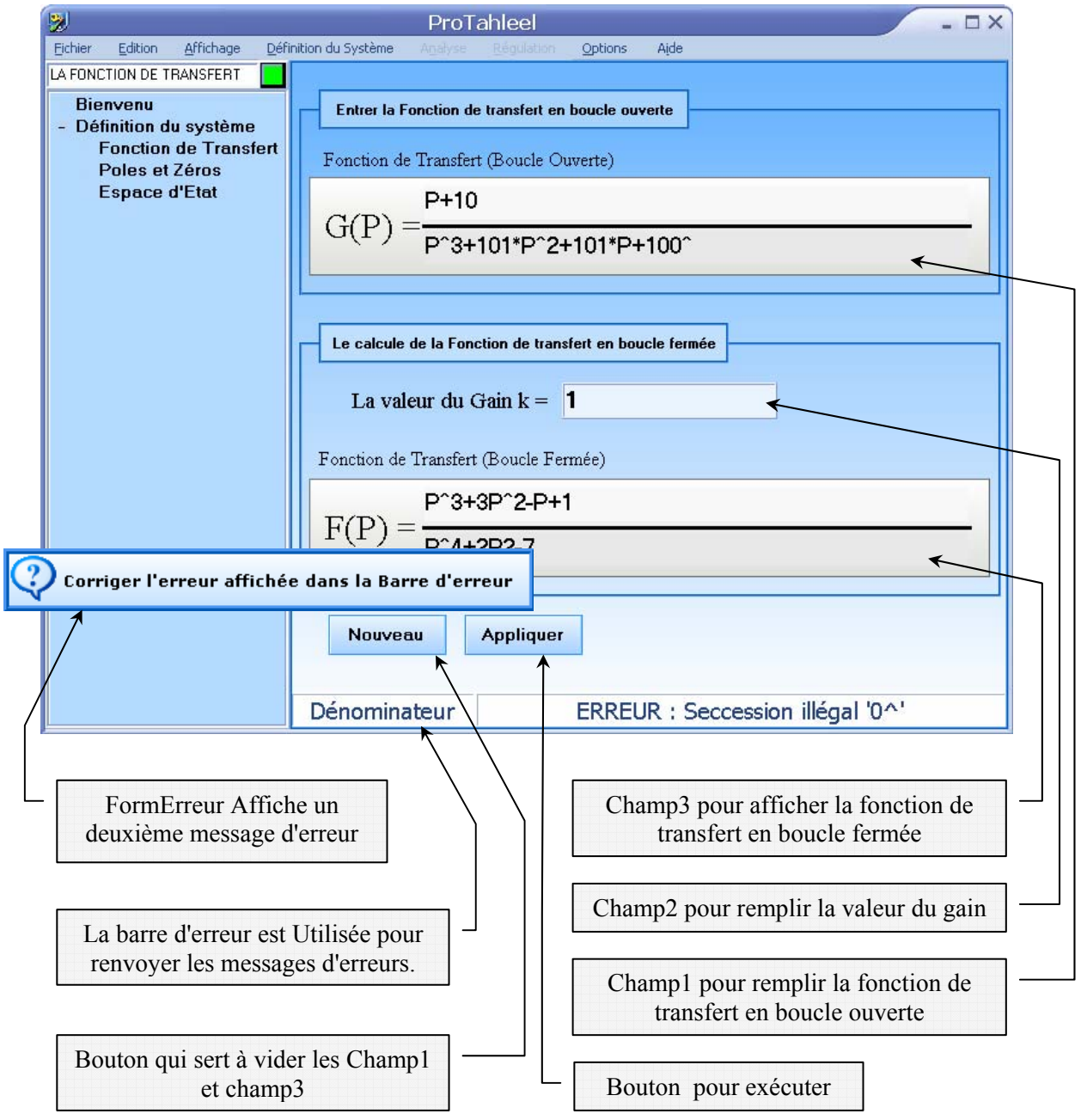
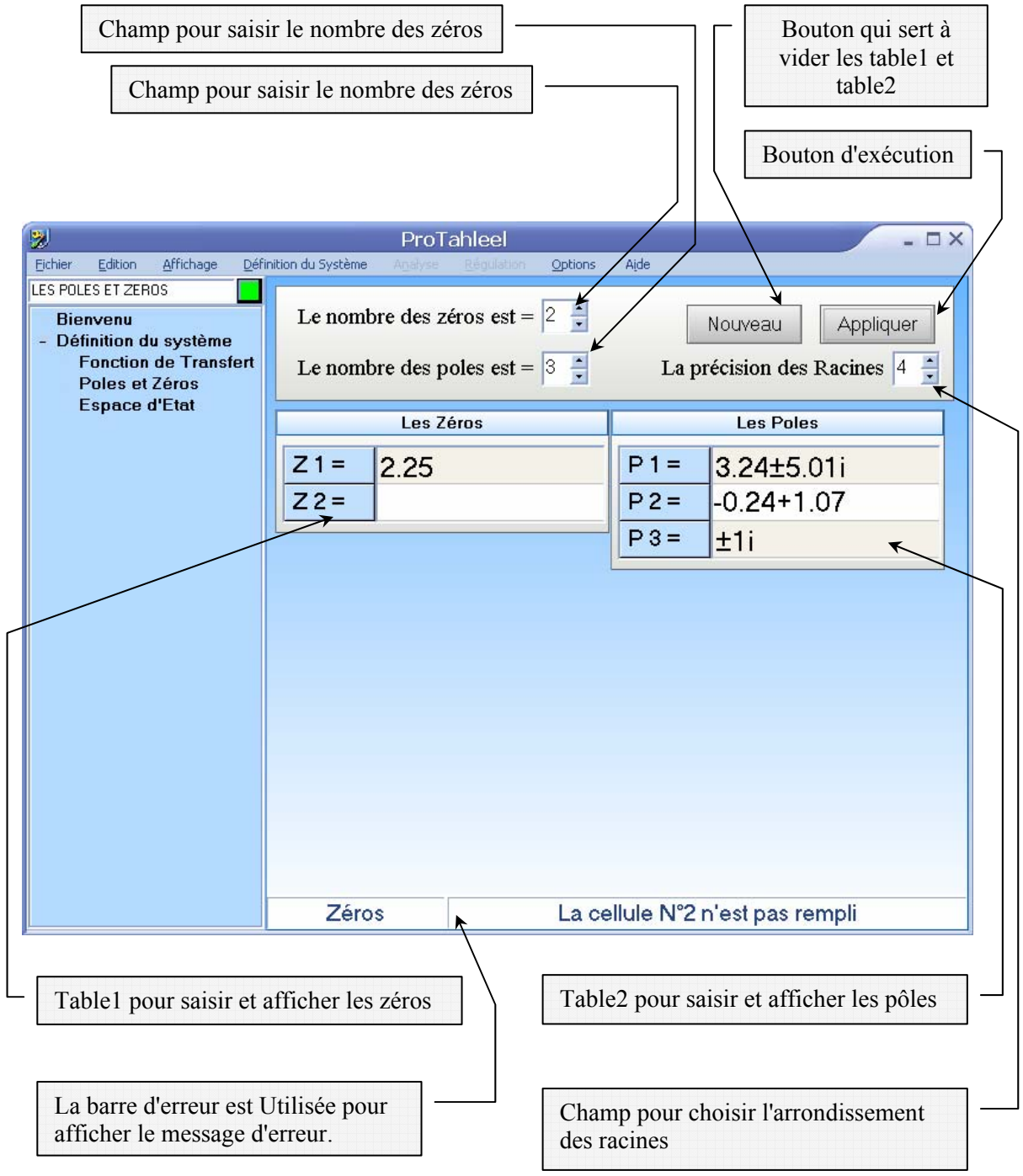


Figure AI.4 : présentant quelques boutons essentiels de menu barre

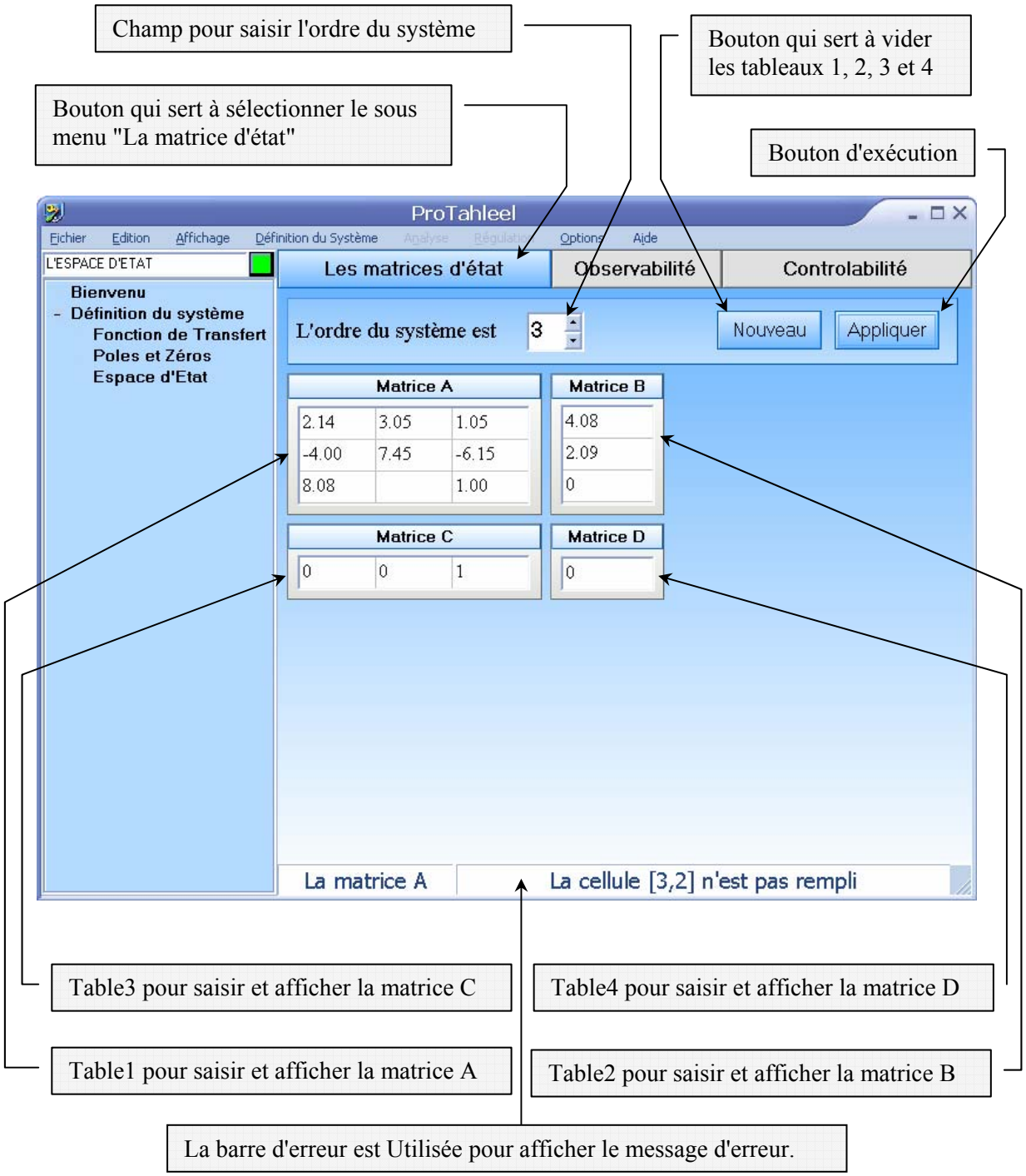
❖ L'INTERFACE DU MENU DE LA FONCTION DE TRANSFERT



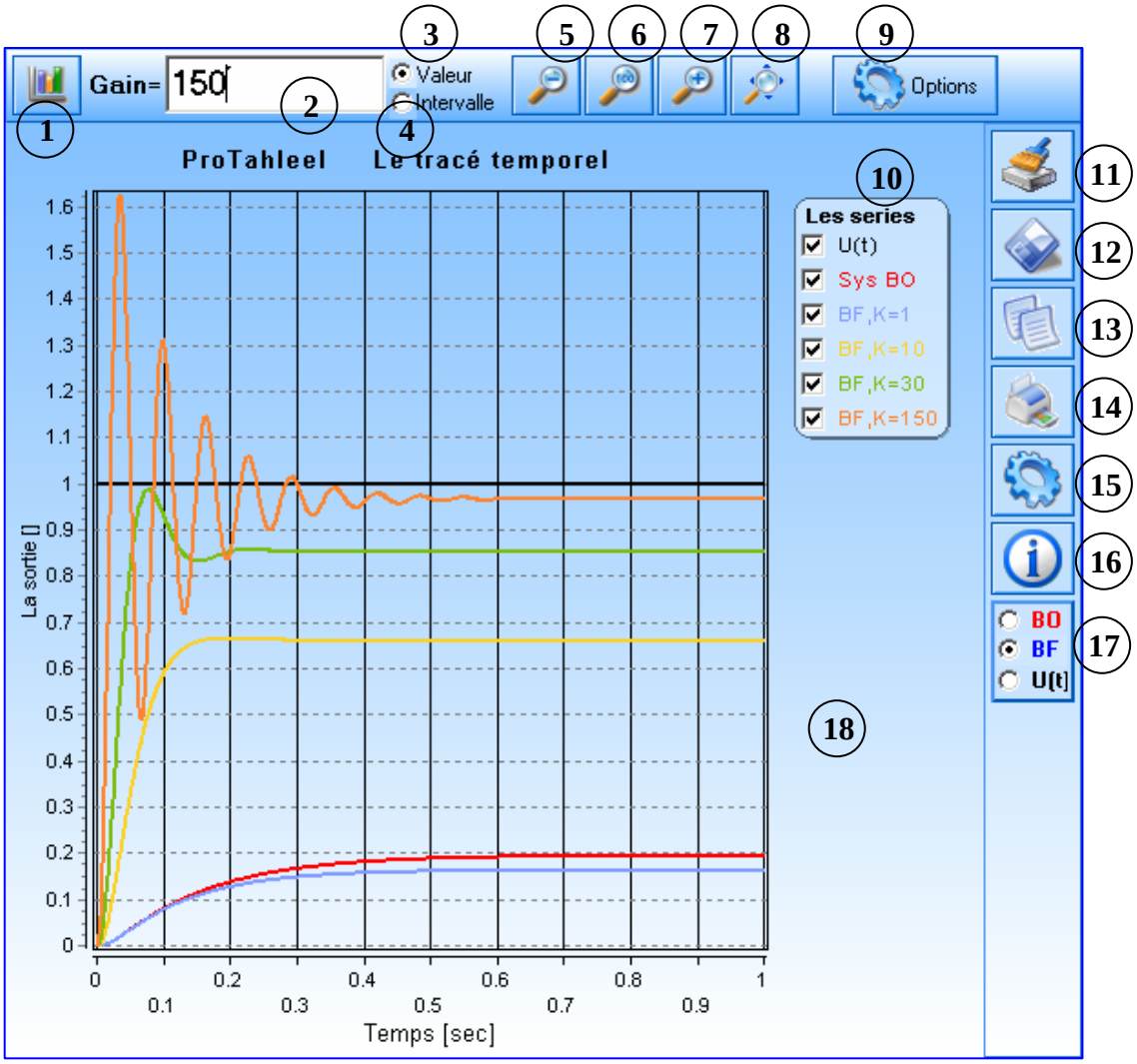
❖ L'INTERFACE DU MENU POLES ET ZEROS



❖ L'INTERFACE DU MENU DE L'ESPACE D'ETAT

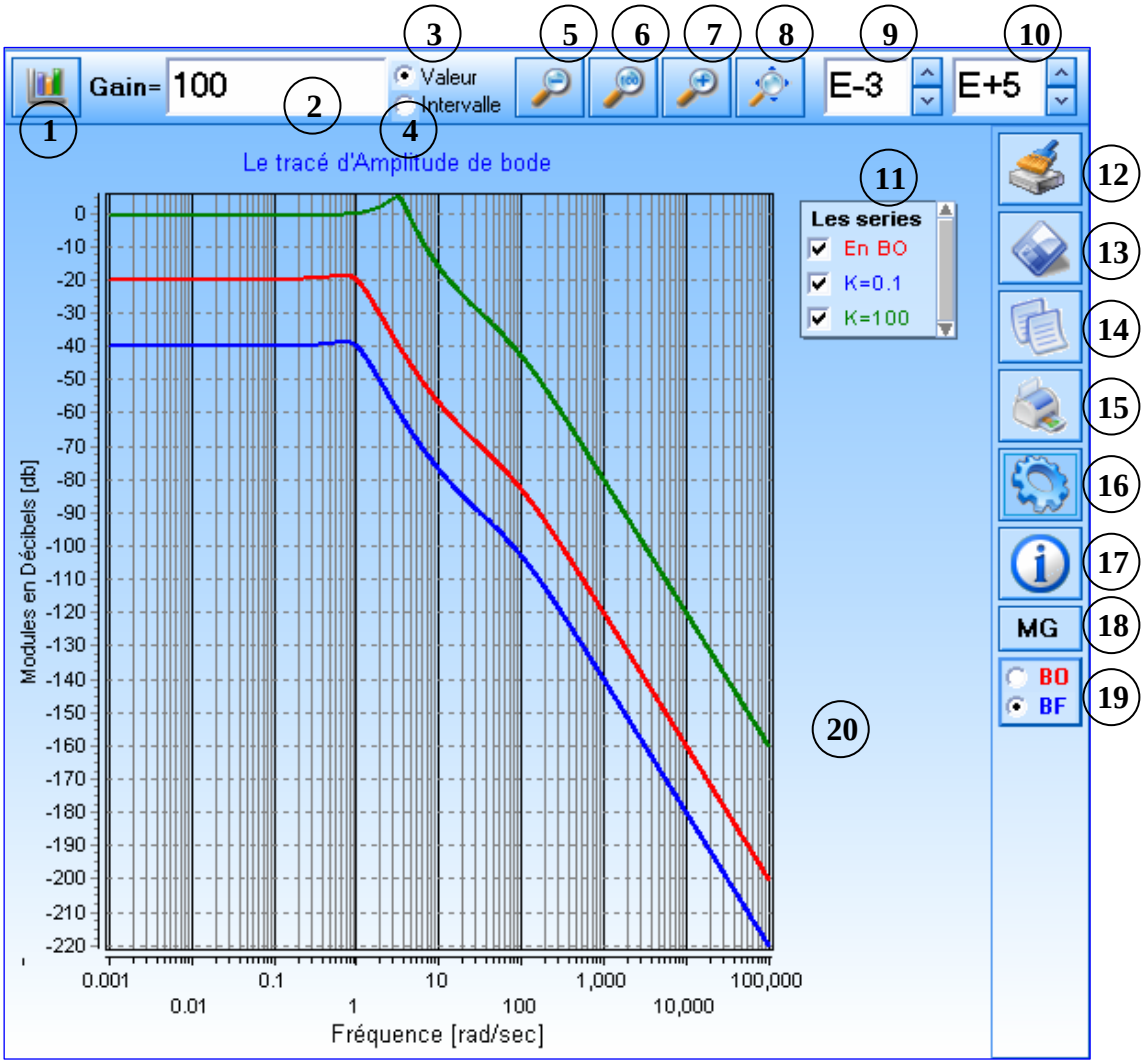


❖ **INTERFACE DU MENU ANALYSE TEMPORELLE**



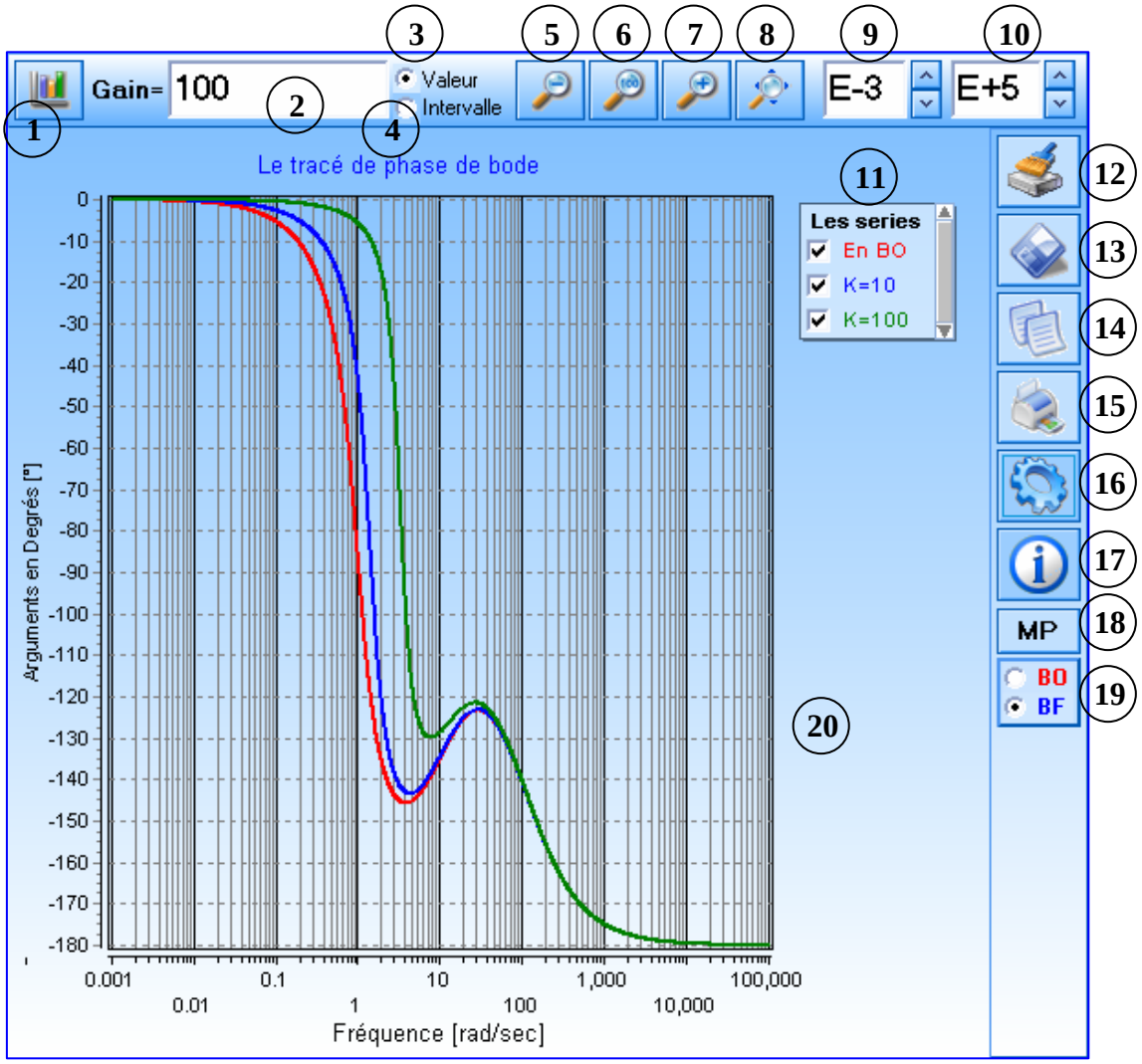
1. Bouton de traçage
2. Champ de saisir du gain k
3. Bouton pour sélectionner le type valeur
4. Bouton pour sélectionner le type intervalle
5. Bouton zoom en avant (minimisé le graphe)
6. Bouton zoom 100%
7. Bouton zoom en arrière (agrandir le graphe)
8. Bouton qui agrandi ou minimisé le menu
9. Bouton qui affiche le sous menu options
10. La légende du graphe
11. Bouton pour supprimer le graphe
12. Bouton d'enregistrement du graphe
13. Bouton pour copier le graphe
14. Bouton pour imprimer le graphe
15. Bouton pour afficher les options de graphe
16. Bouton qui affiche la légende
17. Bouton pour sélectionner la nature du système
18. Surface du graphe (plan de tracé).

❖ INTERFACE DU MENU D'AMPLITUDE DE BODE



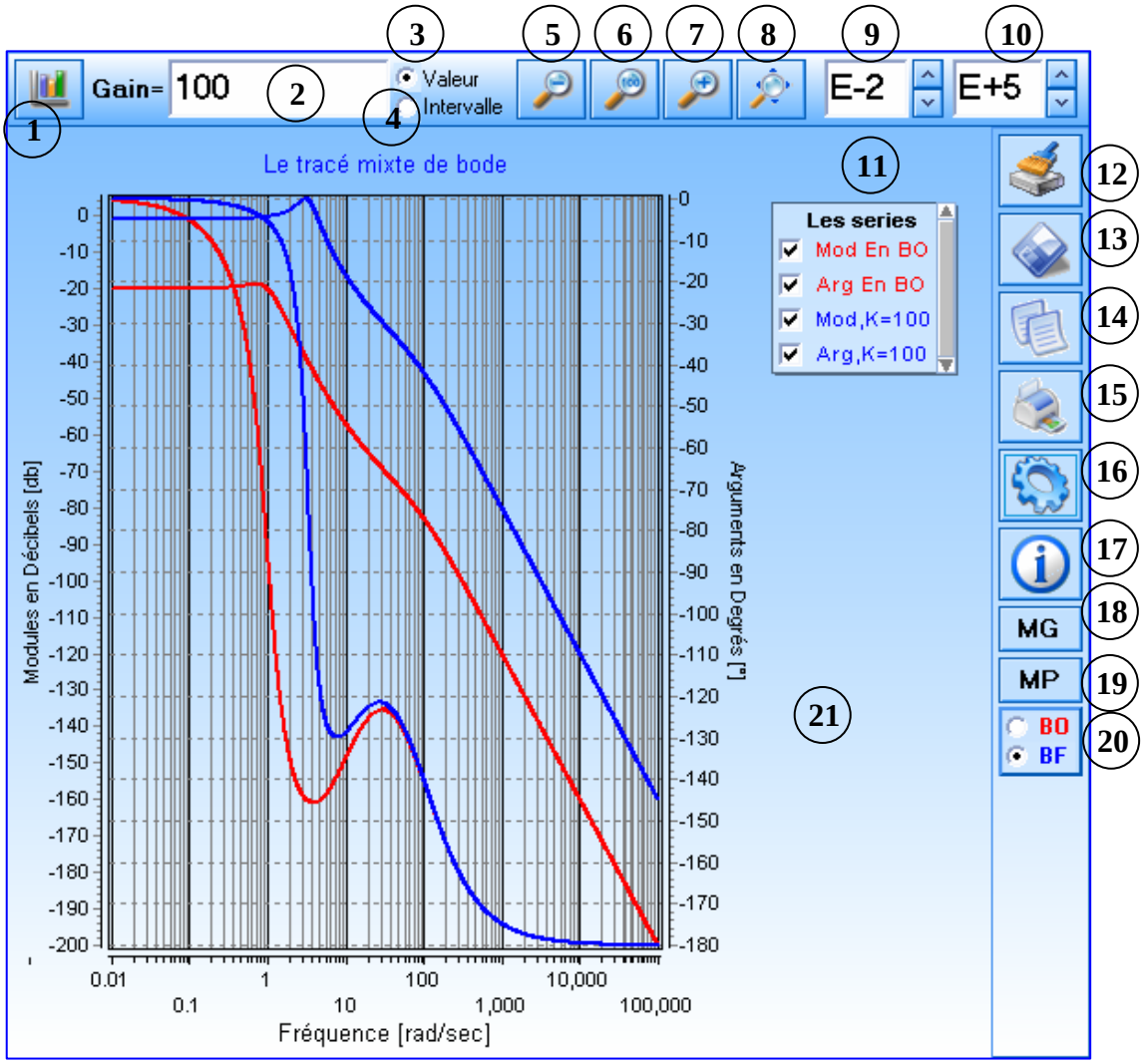
1. Bouton de traçage
2. Champ de saisir du gain k
3. Bouton pour sélectionner le type valeur
4. Bouton pour sélectionner le type intervalle
5. Bouton zoom en avant (minimisé le graphe)
6. Bouton zoom 100%
7. Bouton zoom en arrière (agrandir le graphe)
8. Bouton qui agrandi ou minimisé le menu
9. Champ de saisie du ω_{min} de l'intervalle
10. Champ de saisie du ω_{max} de l'intervalle
11. La légende du graphe
12. Bouton pour supprimer le graphe
13. Bouton d'enregistrement du graphe
14. Bouton pour copier le graphe
15. Bouton pour imprimer le graphe
16. Bouton pour afficher les options de graphe
17. Bouton qui affiche la légende
18. Bouton qui trace la marge de gain
19. Bouton pour sélectionner la nature du système
20. Surface du graphe (plan de tracé).

❖ INTERFACE DU MENU PHASE DE BODE



1. Bouton de traçage
2. Champ de saisir du gain k
3. Bouton pour sélectionner le type valeur
4. Bouton pour sélectionner le type intervalle
5. Bouton zoom en avant (minimisé le graphe)
6. Bouton zoom 100%
7. Bouton zoom en arrière (agrandir le graphe)
8. Bouton qui agrandi ou minimisé le menu
9. Champ de saisie du ω_{min} de l'intervalle
10. Champ de saisie du ω_{max} de l'intervalle
11. La légende du graphe
12. Bouton pour supprimer le graphe
13. Bouton d'enregistrement du graphe
14. Bouton pour copier le graphe
15. Bouton pour imprimer le graphe
16. Bouton pour afficher les options de graphe
17. Bouton qui affiche la légende
18. Bouton qui trace la marge de phase
19. Bouton pour sélectionner la nature du système
20. Surface du graphe (plan de tracé).

❖ **INTERFACE DU MENU MIXTE DE BODE**



1 Bouton de traçage

2 Champ de saisir du gain k

3 Bouton pour sélectionner le type valeur

4 Bouton pour sélectionner le type intervalle

5 Bouton zoom en avant (minimisé le graphe)

6 Bouton zoom 100%

7 Bouton zoom en arrière (agrandir le graphe)

8 Bouton qui agrandi ou minimisé le menu

9 Champ de saisie du ω_{min} de l'intervalle

10 Champ de saisie du ω_{max} de l'intervalle

11 La légende du graphe

12 Bouton pour supprimer le graphe

13 Bouton d'enregistrement du graphe

14 Bouton pour copier le graphe

15 Bouton pour imprimer le graphe

16 Bouton pour afficher les options de graphe

17 Bouton qui affiche la légende

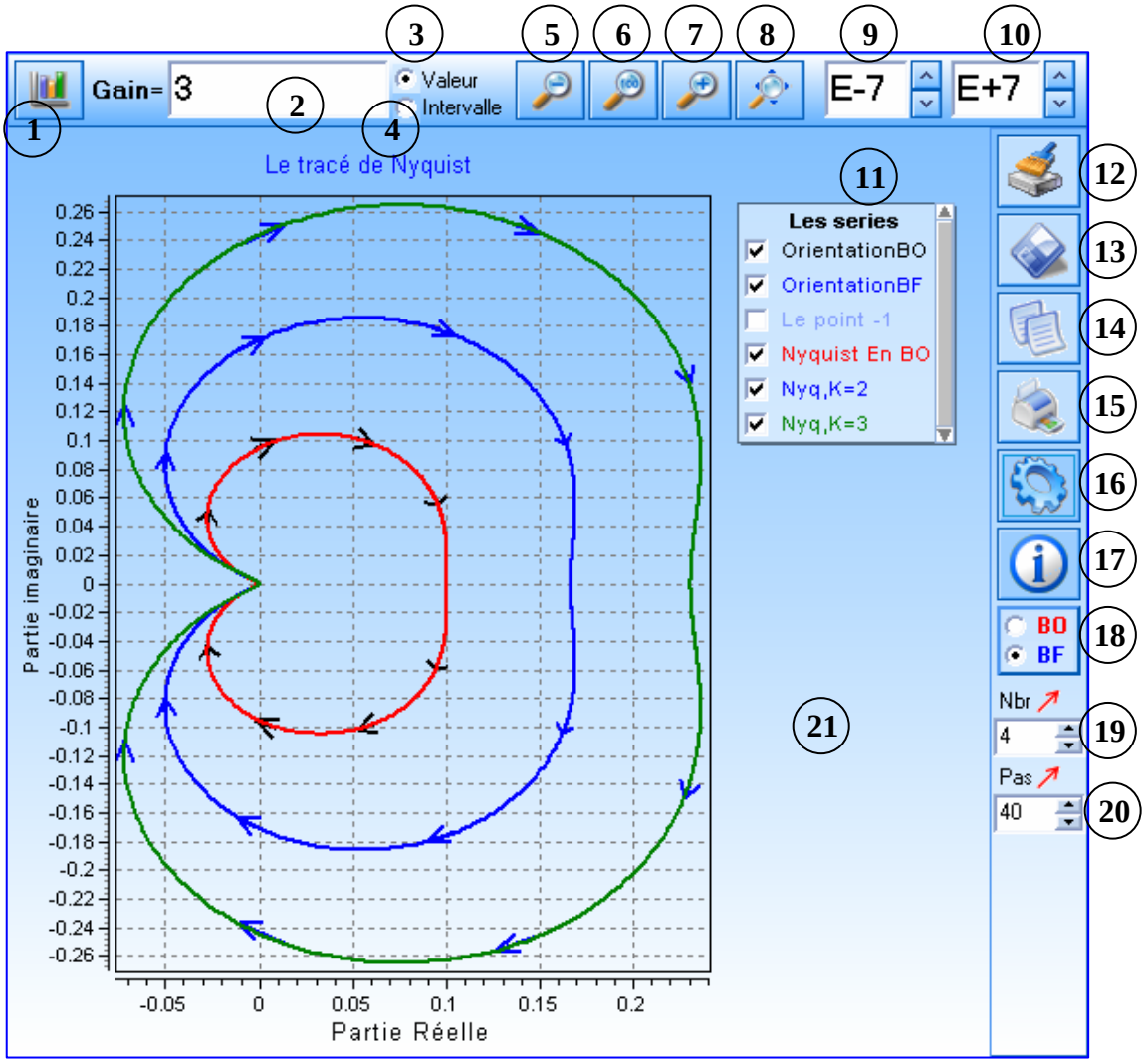
18 Bouton qui trace la marge de gain

19 Bouton qui trace la marge de phase

20 Bouton pour sélectionner la nature du système

21 Surface du graphe (plan de tracé).

❖ **INTERFACE DU MENU NYQUIST**



1 Bouton de traçage

2 Champ de saisir du gain k

3 Bouton pour sélectionner le type valeur

4 Bouton pour sélectionner le type intervalle

5 Bouton zoom en avant (minimisé le graphe)

6 Bouton zoom 100%

7 Bouton zoom en arrière (agrandir le graphe)

8 Bouton qui agrandi ou minimisé le menu

9 Champ de saisie du ω_{min} de l'intervalle

10 Champ de saisie du ω_{max} de l'intervalle

11 La légende du graphe

12 Bouton pour supprimer le graphe

13 Bouton d'enregistrement du graphe

14 Bouton pour copier le graphe

15 Bouton pour imprimer le graphe

16 Bouton pour afficher les options de graphe

17 Bouton qui affiche la légende

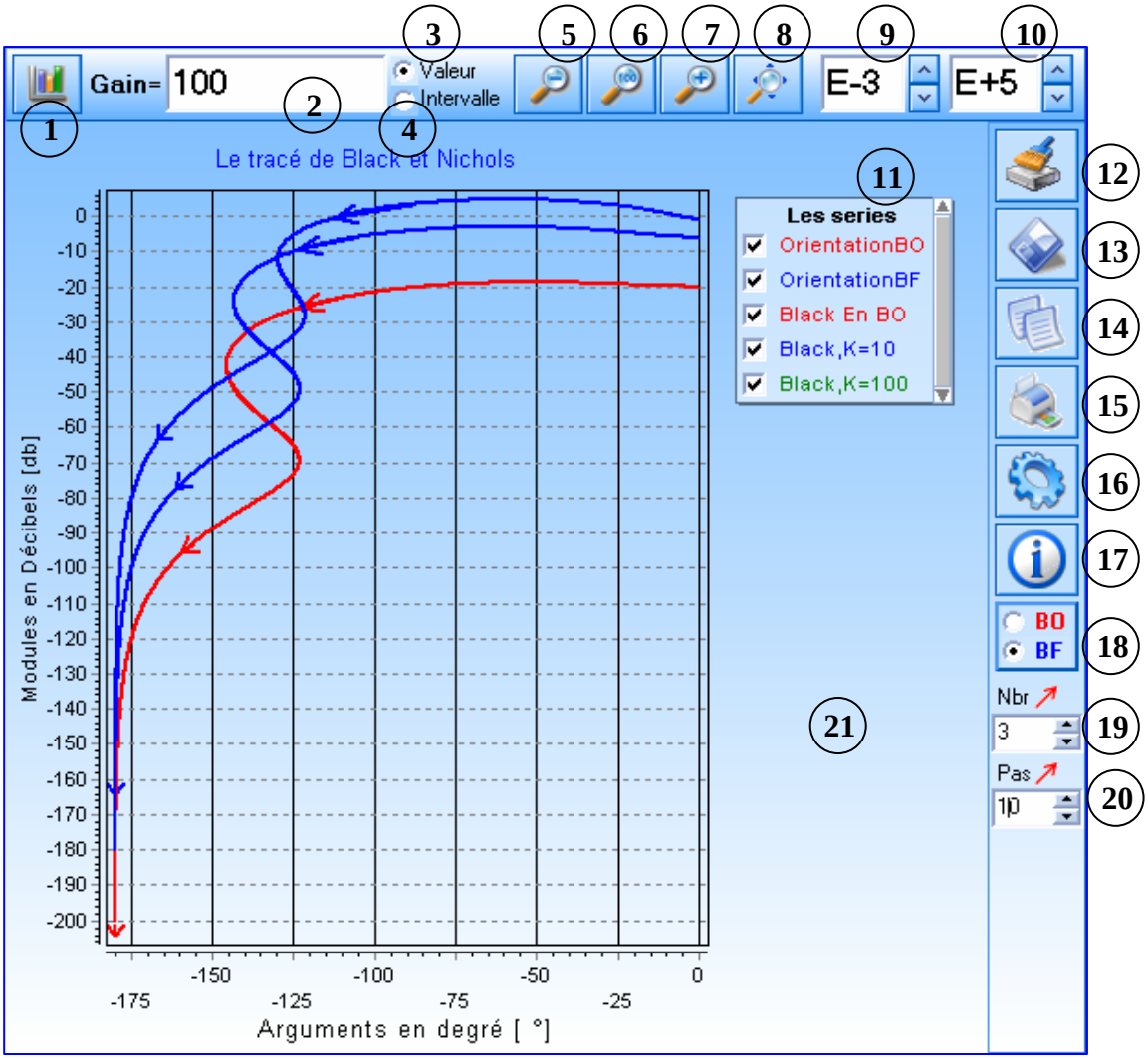
18 Bouton pour sélectionner la nature du système

19 Champ pour saisir le nombre des flèches.

20 Champ pour saisir le pas des flèches.

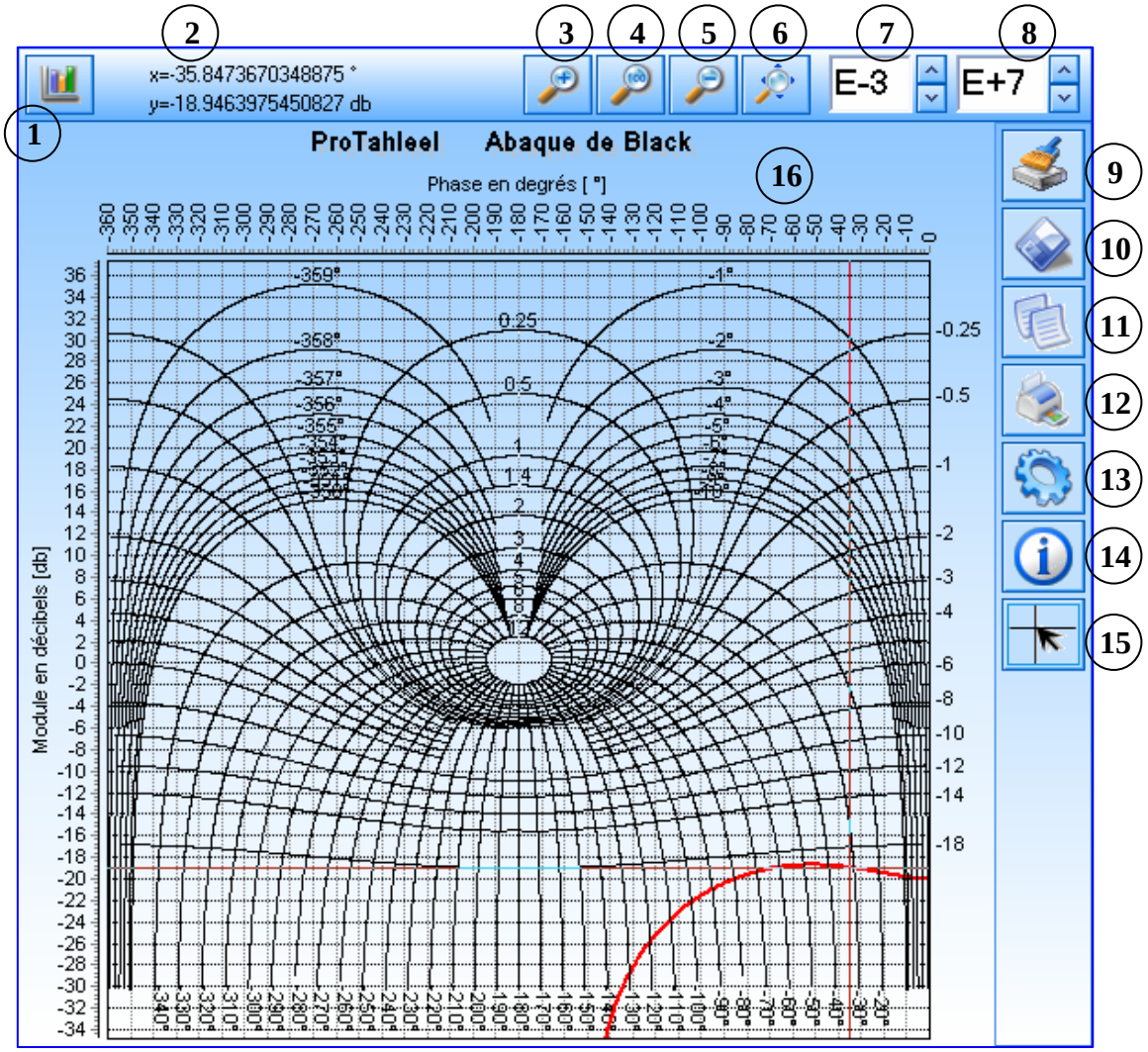
21 Surface du graphe (plan de tracé).

❖ **INTERFACE DU MENU BLACK-NICHOLS**



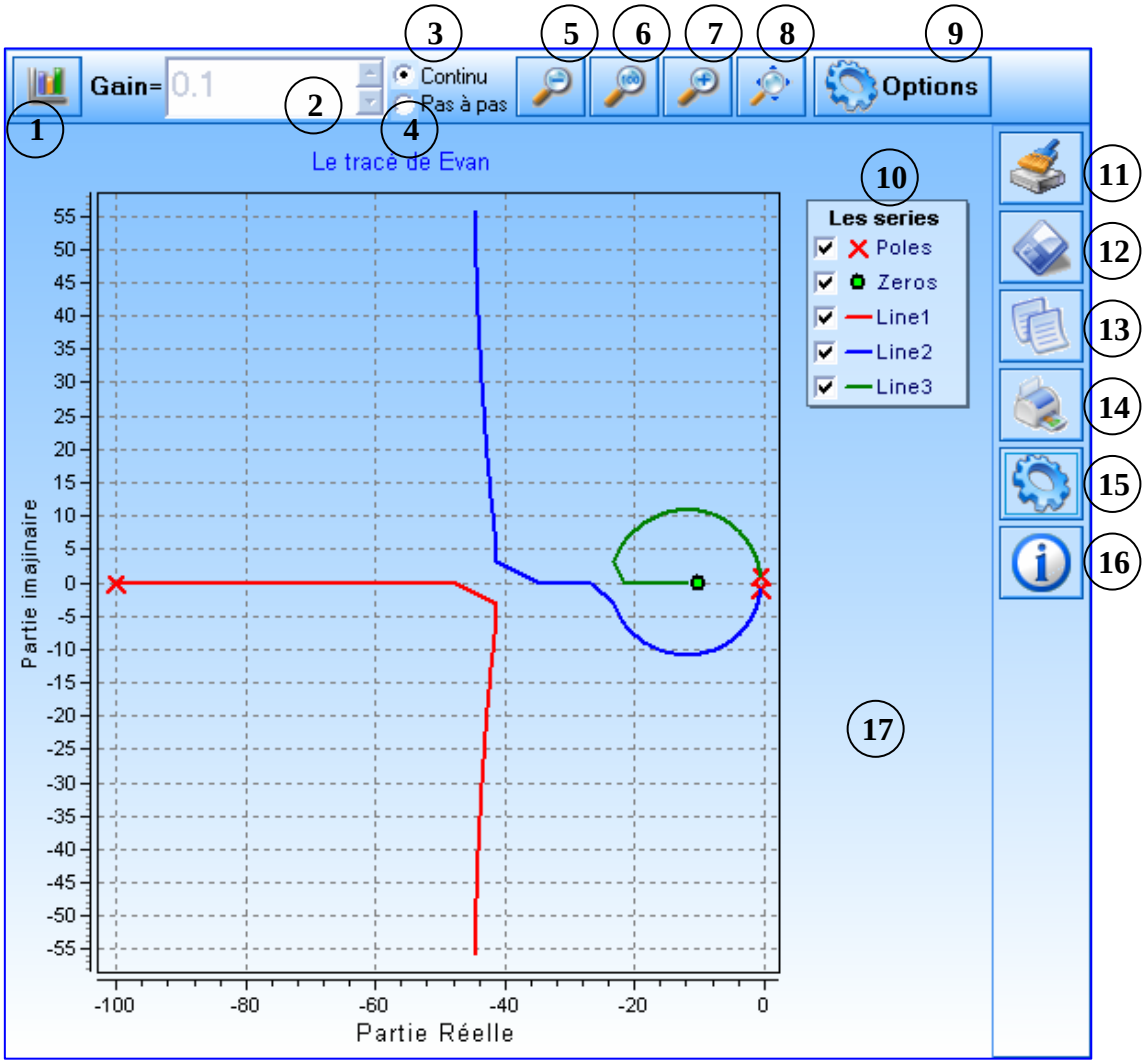
1. Bouton de traçage
2. Champ de saisir du gain k
3. Bouton pour sélectionner le type valeur
4. Bouton pour sélectionner le type intervalle
5. Bouton zoom en avant (minimisé le graphe)
6. Bouton zoom 100%
7. Bouton zoom en arrière (agrandir le graphe)
8. Bouton qui agrandi ou minimisé le menu
9. Champ de saisie du ω_{min} de l'intervalle
10. Champ de saisie du ω_{max} de l'intervalle
11. La légende du graphe
12. Bouton pour supprimer le graphe
13. Bouton d'enregistrement du graphe
14. Bouton pour copier le graphe
15. Bouton pour imprimer le graphe
16. Bouton pour afficher les options de graphe
17. Bouton qui affiche la légende
18. Bouton pour sélectionner la nature du système
19. Champ pour saisir le nombre des flèches.
20. Champ pour saisir le pas des flèches.
21. Surface du graphe (plan de tracé).

❖ **INTERFACE DU MENU L'ABAQUE**



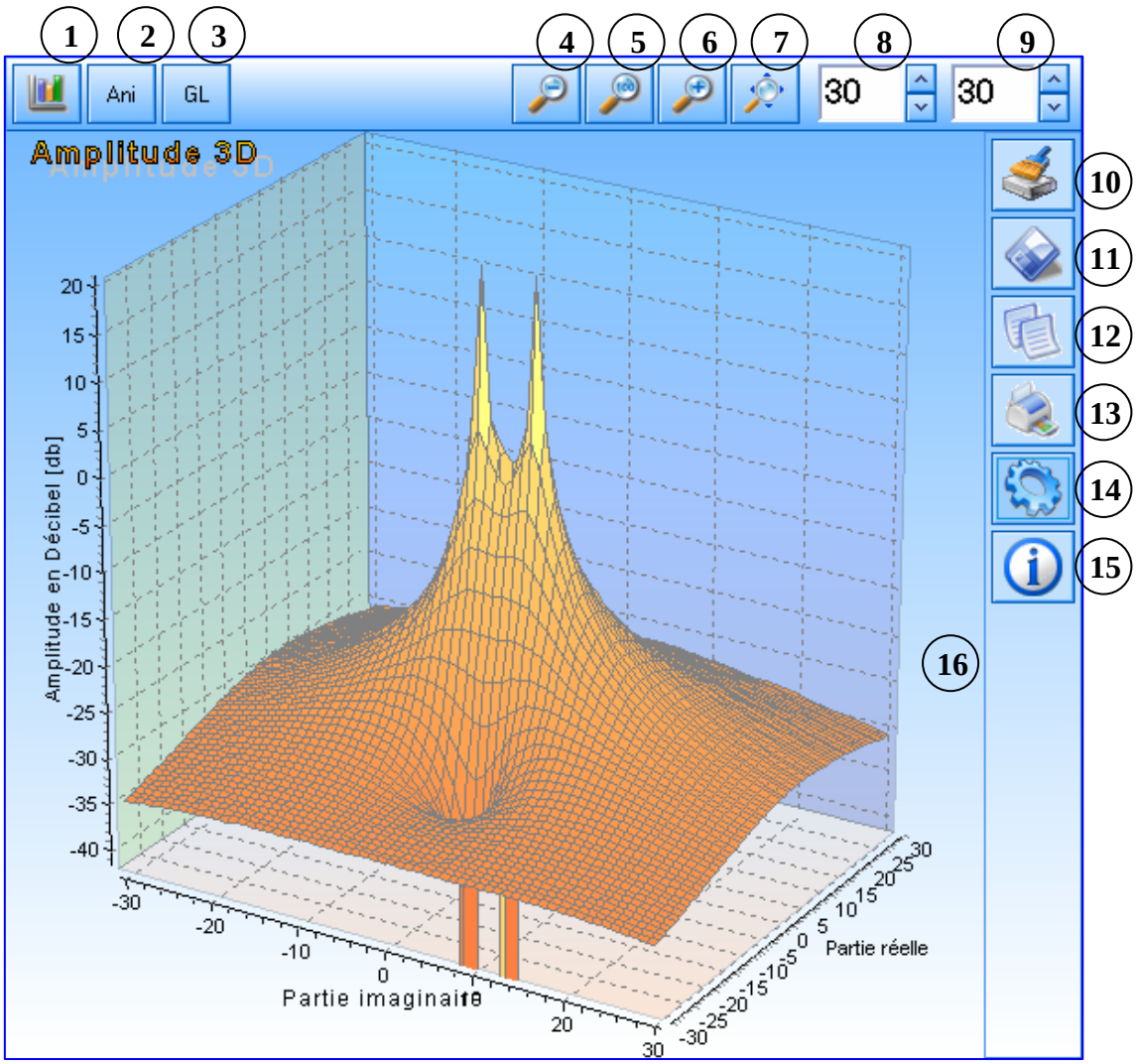
1 Bouton de traçage	**9** Bouton pour supprimer le graphe
2 Champ sert à afficher les valeurs de (x,y).	**10** Bouton d'enregistrement du graphe
3 Bouton zoom en avant (minimisé le graphe)	**11** Bouton pour copier le graphe
4 Bouton zoom 100%	**12** Bouton pour imprimer le graphe
5 Bouton zoom en arrière (agrandir le graphe)	**13** Bouton pour afficher les options de graphe
6 Bouton qui agrandi ou minimisé le menu	**14** Bouton qui affiche la légende
7 Champ de saisie de la ω_{min} de l'intervalle	**15** Bouton qui affiche les cordonnées du plan.
8 Champ de saisie de la ω_{max} de l'intervalle	**16** Surface du graphe (plan de tracé).

❖ INTERFACE DU MENU EVANS



1. Bouton de traçage
2. Champ de saisir du gain k
3. Bouton pour le traçage continu
4. Bouton pour le traçage pas à pas
5. Bouton zoom en avant (minimisé le graphe)
6. Bouton zoom 100%
7. Bouton zoom en arrière (agrandir le graphe)
8. Bouton qui agrandi ou minimisé le menu
9. Bouton pour afficher les options de traçage
10. La légende du graphe
11. Bouton pour supprimer le graphe
12. Bouton d'enregistrement du graphe
13. Bouton pour copier le graphe
14. Bouton pour imprimer le graphe
15. Bouton pour afficher les options de graphe
16. Bouton qui affiche la légende
17. Surface du graphe (plan de tracé).

❖ **INTERFACE DU MENU D'AMPLITUDE DE BODE EN 3D**



1 Bouton de traçage

2 Bouton qui active l'animation rotative.

3 Bouton qui active l'option (Open GL).

4 Bouton zoom en avant (minimisé le graphe)

5 Bouton zoom 100%

6 Bouton zoom en arrière (agrandir le graphe)

7 Bouton qui agrandi ou minimisé le menu

8 Champ de saisie de ω_{min} et ω_{max} de l'intervalle

9 Saisir de σ_{min} et σ_{max} de l'intervalle

10 Bouton pour supprimer le graphe

11 Bouton d'enregistrement du graphe

12 Bouton pour copier le graphe

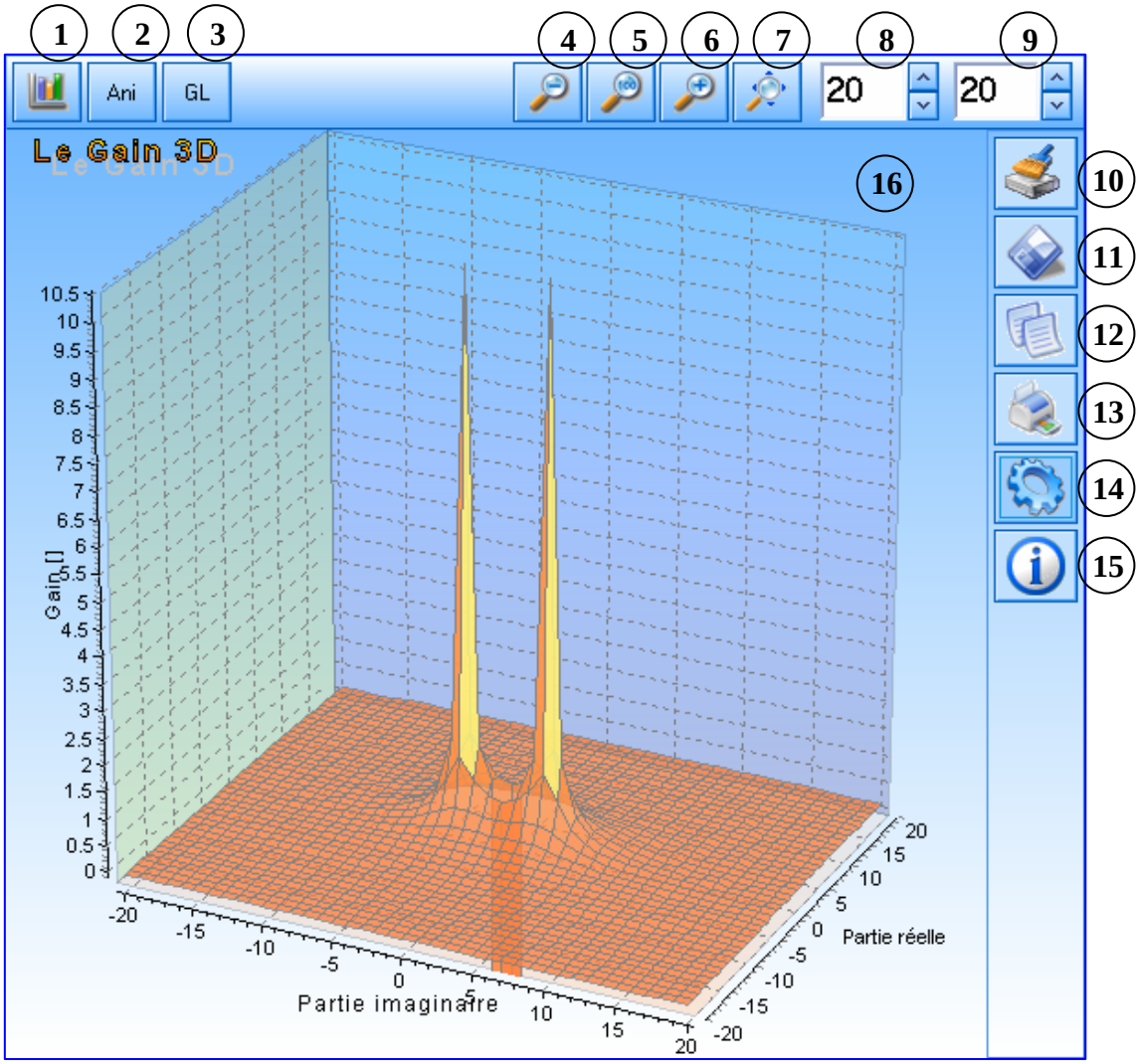
13 Bouton pour imprimer le graphe

14 Afficher les options de graphe

15 Bouton qui affiche la légende

16 Surface où le graphe est tracé

❖ **INTERFACE DU MENU DE GAIN DE BODE EN 3D**



1 Bouton de traçage

2 Bouton qui active l'animation rotative.

3 Bouton qui active l'option (Open GL).

4 Bouton zoom en avant (minimisé le graphe)

5 Bouton zoom 100%

6 Bouton zoom en arrière (agrandir le graphe)

7 Bouton qui agrandi ou minimisé le menu

8 Champ de saisie de ω_{min} et ω_{max} de l'intervalle

9 Saisir de σ_{min} et σ_{max} de l'intervalle

10 Bouton pour supprimer le graphe

11 Bouton d'enregistrement du graphe

12 Bouton pour copier le graphe

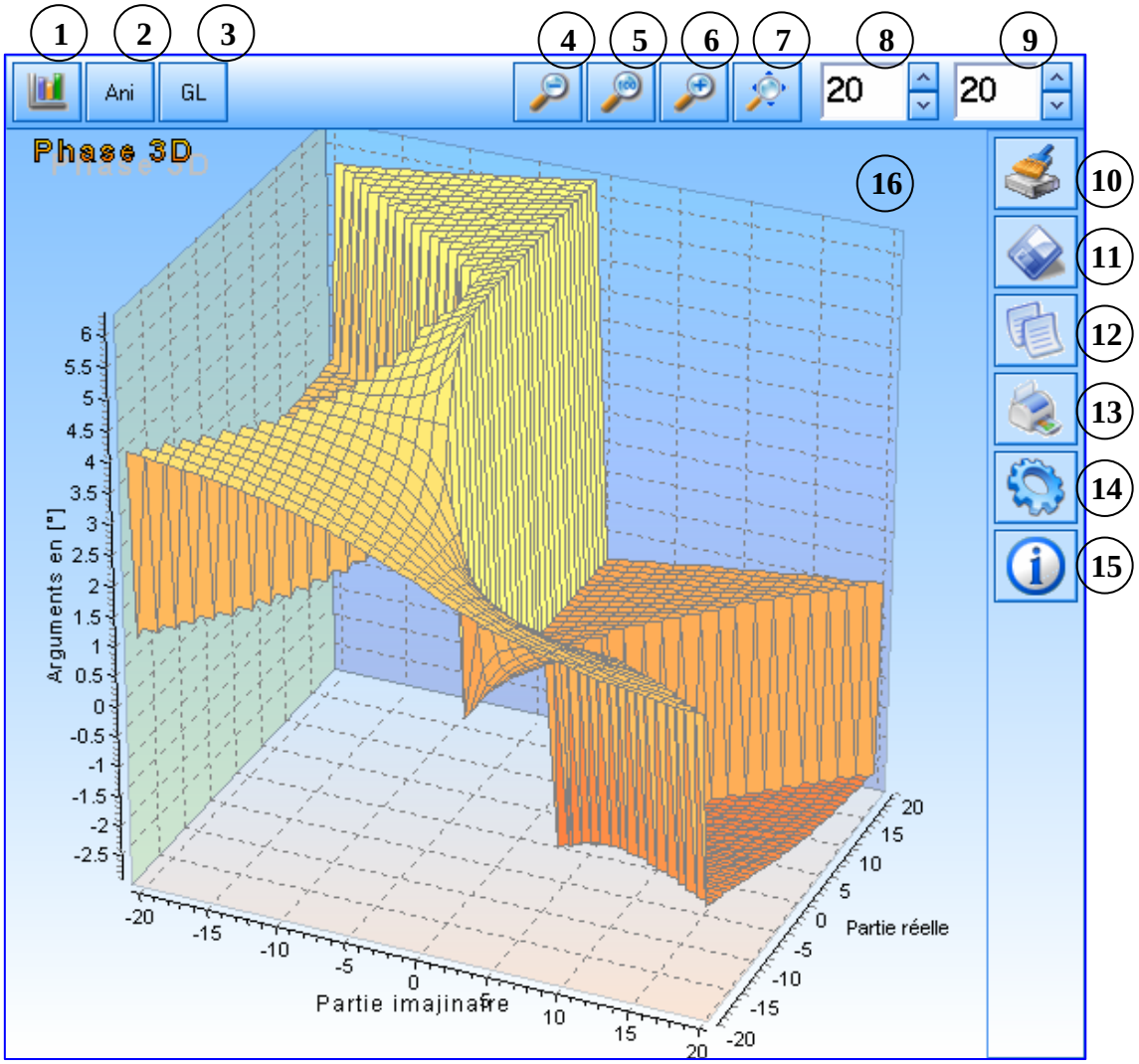
13 Bouton pour imprimer le graphe

14 Afficher les options de graphe

15 Bouton qui affiche la légende

16 Surface où le graphe est tracé

❖ **INTERFACE DU MENU PHASE DE BODE EN 3D**



1 Bouton de traçage

2 Bouton qui active l'animation rotative.

3 Bouton qui active l'option (Open GL).

4 Bouton zoom en avant (minimisé le graphe)

5 Bouton zoom 100%

6 Bouton zoom en arrière (agrandir le graphe)

7 Bouton qui agrandi ou minimisé le menu

8 Champ de saisie de ω_{min} et ω_{max} de l'intervalle

9 Saisir de σ_{min} et σ_{max} de l'intervalle

10 Bouton pour supprimer le graphe

11 Bouton d'enregistrement du graphe

12 Bouton pour copier le graphe

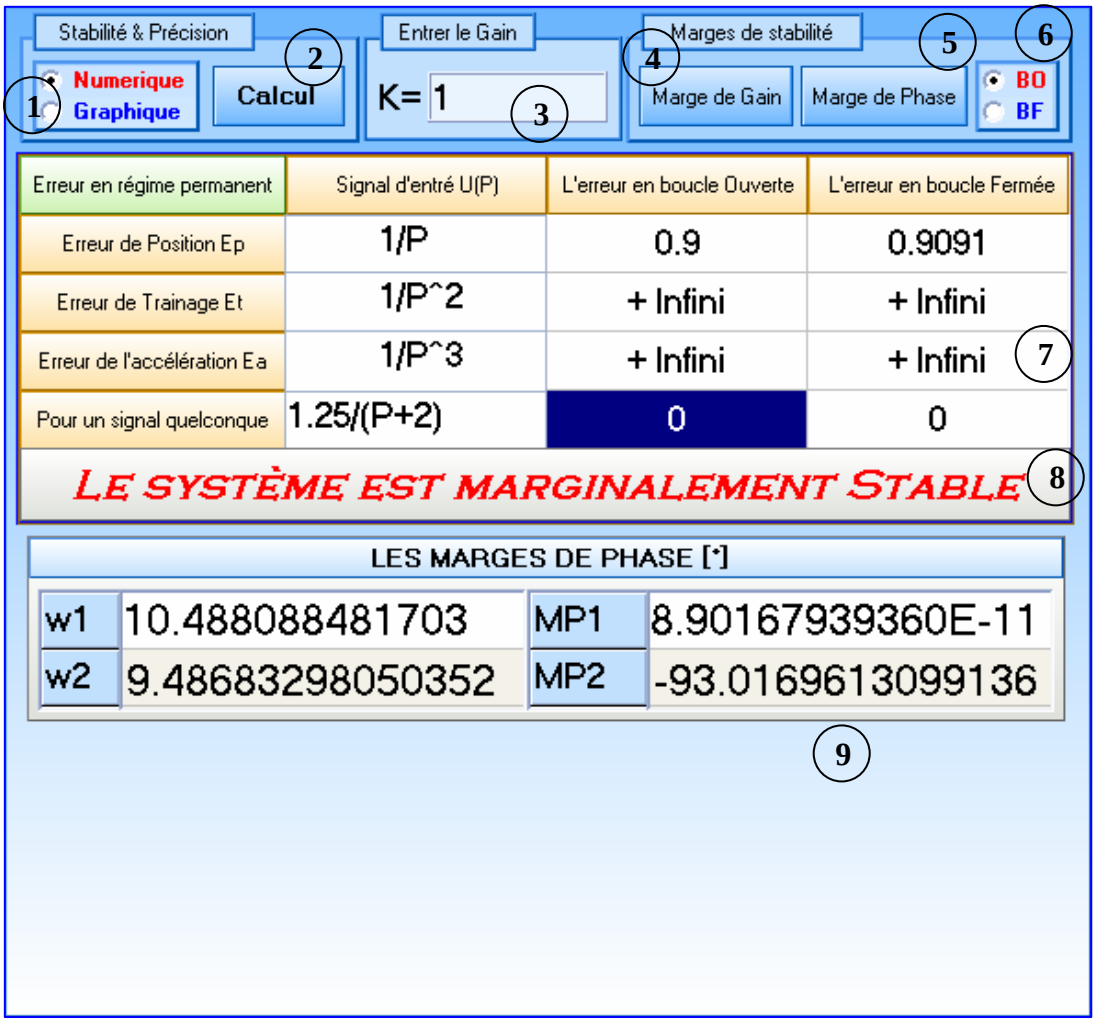
13 Bouton pour imprimer le graphe

14 Afficher les options de graphe

15 Bouton qui affiche la légende

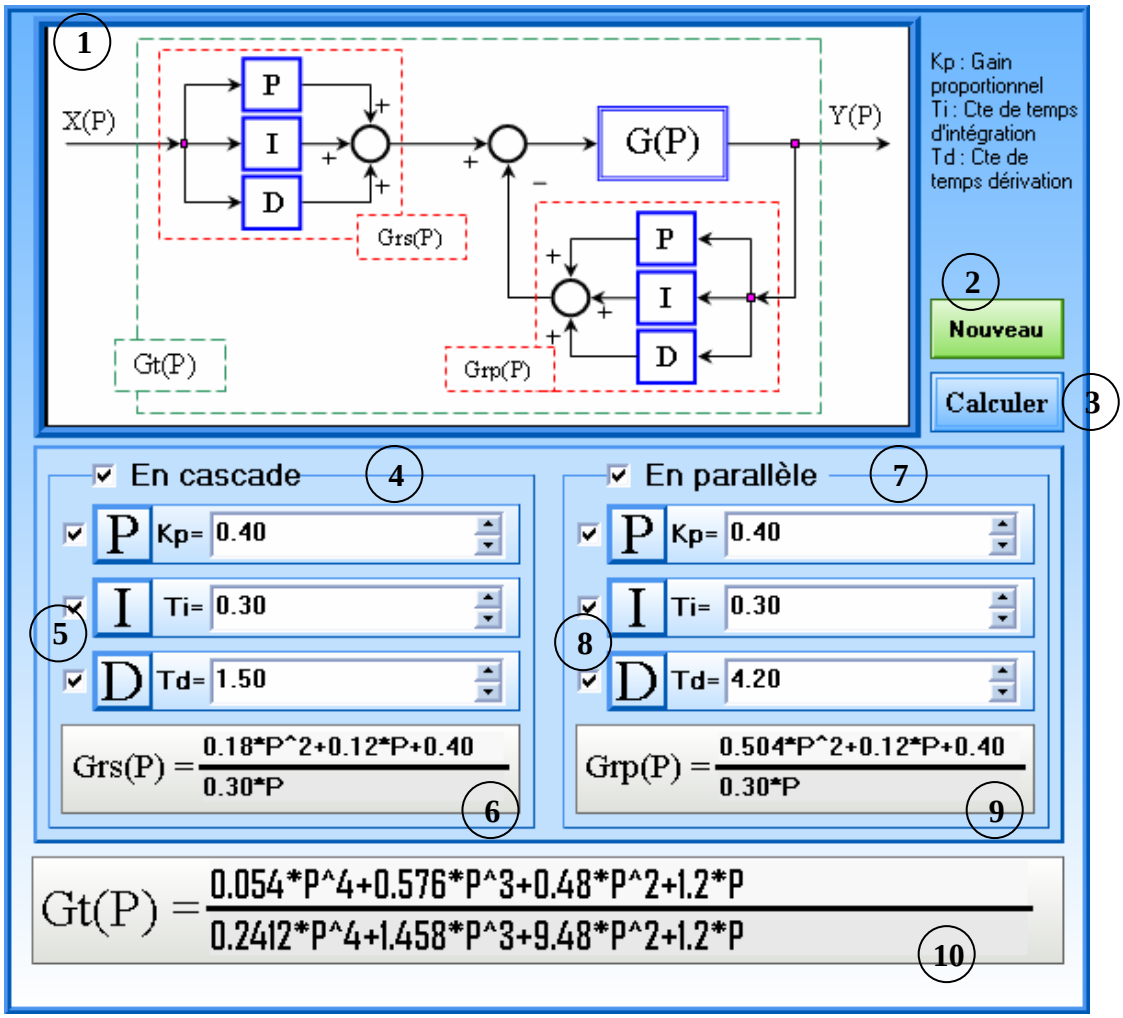
16 Surface où le graphe est tracé

❖ **INTERFACE DU MENU PERFORMANCES**



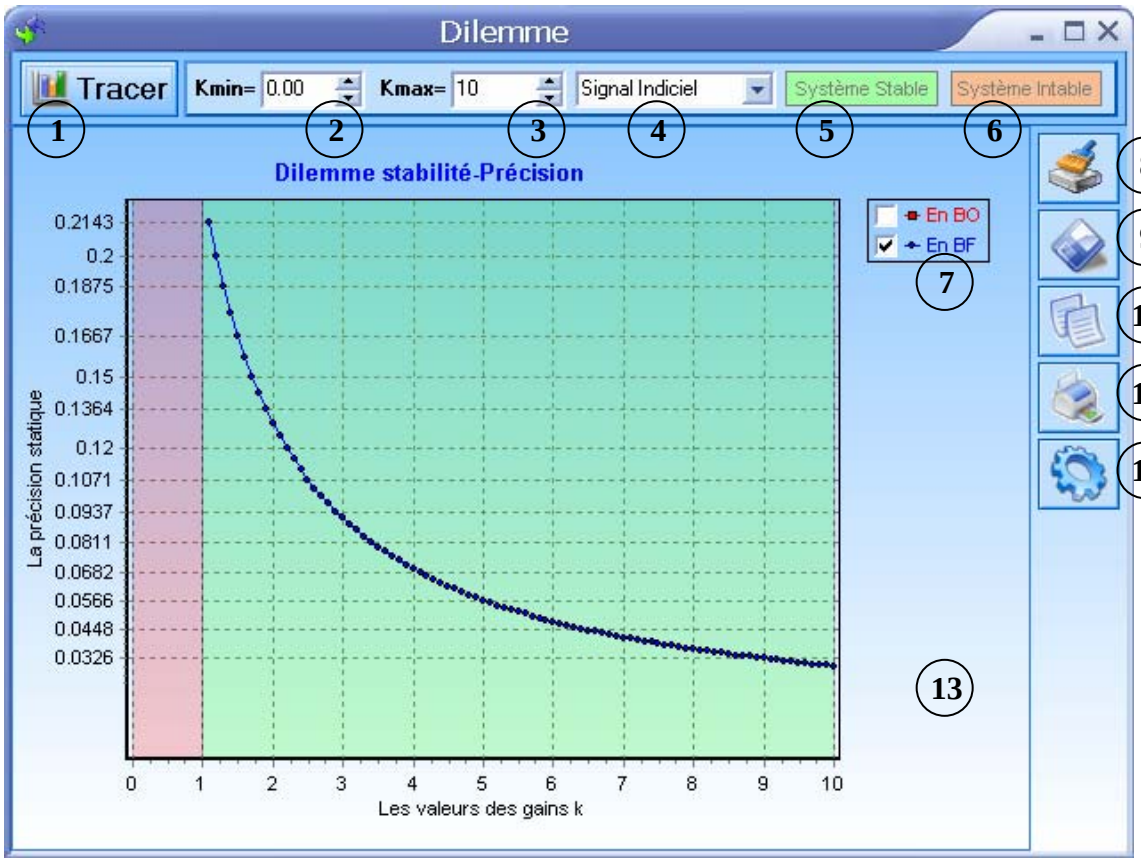
1. Bouton pour sélectionner la manière de calcul.
2. Bouton d'exécution.
3. Champ de saisir du gain k
4. Bouton qui calcul la marge de gain
5. Bouton qui calcul la marge de phase
6. Pour sélectionner la nature du système
7. Tableau qui affiche l'erreur statique.
8. Champ qui affiche la stabilité absolue.
9. Tableau qui affiche la marge de stabilité

❖ **INTERFACE DU MENU DE REGULATION**



1. Figure qui représente la méthode de régulation
2. Bouton qui initialisé tous les champs
3. Bouton d'exécution
4. Sélectionner la correction en cascade
5. Sélectionner et entrer les paramètres des régulateurs.
6. Afficher la FT de la régulation en cascade
7. Sélectionner la correction en cascade
8. Sélectionner et entrer les paramètres des régulateurs.
9. Afficher la FT de la régulation en cascade
10. Afficher la fonction de transfert de la régulation totale en boucle ouverte.

❖ **INTERFACE DU SOUS MENU DILEMME**



1 Bouton de traçage

2 Champ de saisir du gain minimal k_{min}

3 Champ de saisir du gain maximal k_{max}

4 Sélectionnée le type du signal d'entrée

5 Indique que le système est stable.

6 Indique que le système est instable.

7 La légende du graphe

8 Bouton pour supprimer le graphe

9 Bouton d'enregistrement du graphe

10 Bouton pour copier le graphe

11 Bouton pour imprimer le graphe

12 Afficher les options de graphe

13 Surface où le graphe est tracé

❖ MENU IMPRIMER

L'éditeur d'état livré avec ProTahleel permet de réaliser facilement une impression, où cette dernière est consultable à l'écran dans le mode "aperçu", la fenêtre de prévisualisation est paramétrable où on peut de revenir en arrière lors d'une visualisation.

Le principe d'une impression graphique peut varier d'une catégorie d'imprimantes à une autre, même si les idées de base restent les même.

Le principe est basé sur trois idées-forces :

1- Analyse de l'image à imprimer.
2- Méthode d'impression.
3- Sélection des caractères de contrôle selon les options assignées (orientation, marges, …).

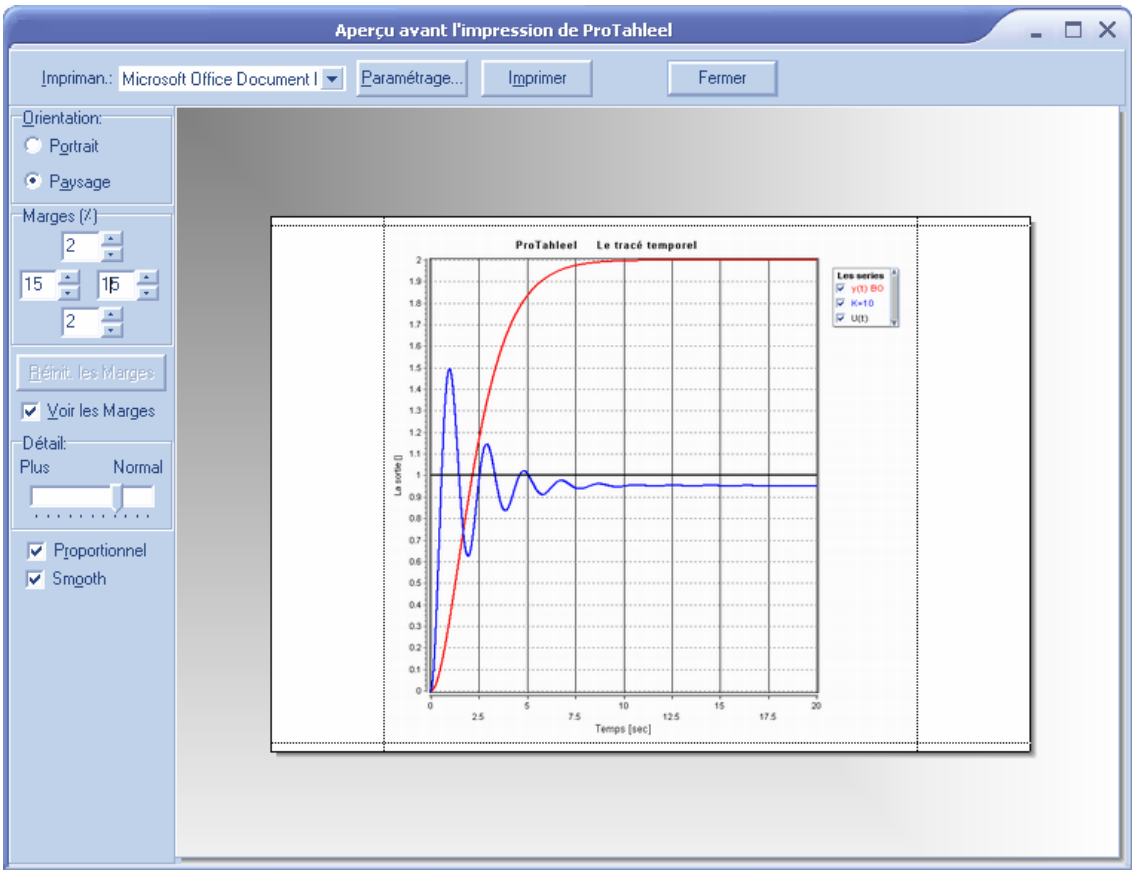


Figure A.I.1 : l'interface de la fenêtre de prévisualisation

❖ **MENU EDITEUR D'OPTIONS POUR LE GRAPHE**

L'éditeur d'options de graphe permet de manipuler les paramètres d'un graphe (couleur, type et épaisseur de graphe,…).

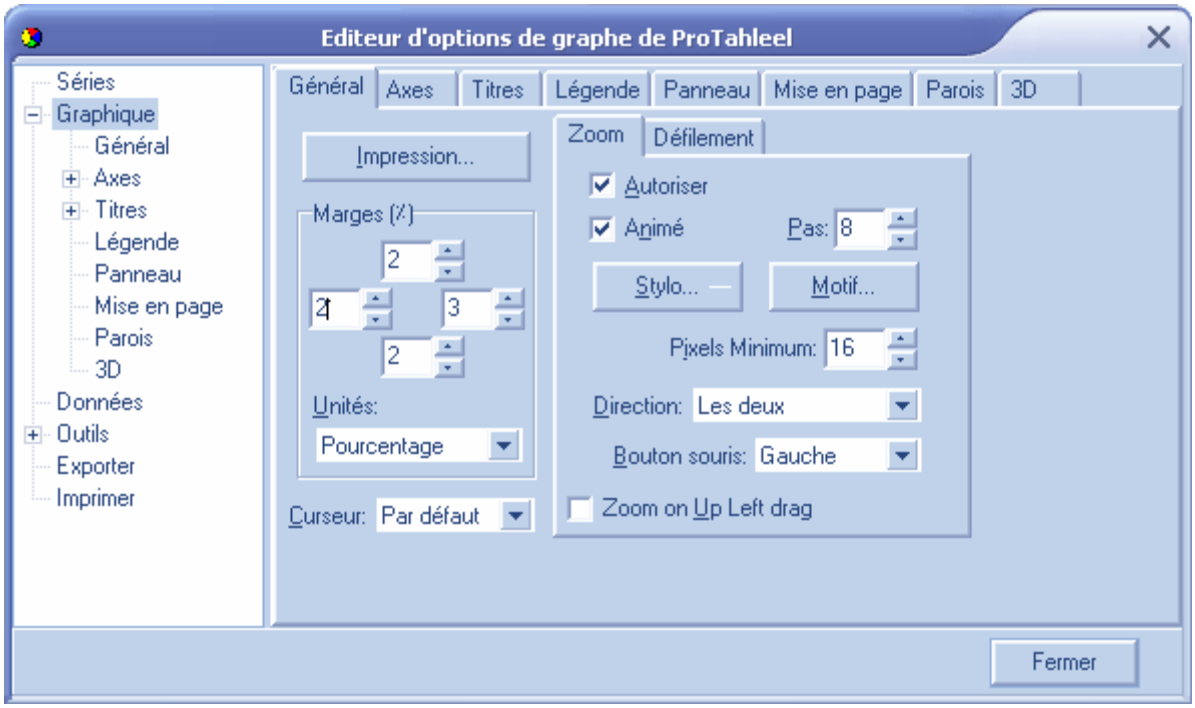


Figure AI.2 : présentant l'éditeur d'options de graphe

Annexe II

Les bibliothèques utilisées

BIBLIOTHEQUES MATHEMATIQUES DES FONCTIONS D'ACCOMPAGNEMENTS :

Ces bibliothèques jouent un rôle primordial puisqu'elles sont utilisées dans tous les menus du logiciel ProTahleel sans exception. Vu leurs nombres, on pourra classer les fonctions comme suites:

```
Type VecteurString =array[1..50] of Shortstring;
     VecteurReel   =array[1..50] of Real;
     TableSting    =array[1..50,1..5] of Shortstring;
     TableReel     =array[1..50,1..5] of Real;
```

❖ **LIBRARY : ERREURENECRITURE**

- **Function** MessageDeErreur(Poly ,variable, langue :Shortstring):Shortstring;
  Détection d'une erreur dans l'écriture du polynôme
- **Function** PositionDeErreur(Poly, variable, Langue :Shortstring):Integer;
  Détection de la position de l'erreur dans l'écriture du polynôme

❖ **LIBRARY : OPERATIONPOLYNOME**

- **Function** VecPoly(Poly, Variable : Shortstring):TableString;
  Transformer le polynôme a un Vecteur
- **Function** DimVecPoly(Poly, Variable :Shortstring):Integer;
  Dimension du Vecteur VecPoly
- **Function** SommePoly(Poly1,Poly2,Variable:Shortstring):ShortString;
  La Somme de Deux polynômes
- **Function** SustractionPoly(Poly1,Poly2,Variable:Shortstring):ShortString;
  La Soustraction de Deux polynômes
- **Function** MultScalPoly(Poly,Variable:Shortstring;gain:real):ShortString;
  La Produit d'un Scalaire a un polynômes
- **Function** VecMultPoly(Poly1,Poly2,Variable:ShortString):VecteurReel;
  Un vecteur résultant de la Multiplication de deux polynômes

- **Function** DimVecMultPoly(Poly1,Poly2,Variable:ShortString):Integer;
  Dimension du vecteur résultant de la Multiplication de deux polynomes
- **Function** MultPoly(Poly1,Poly2,Variable:ShortString):ShortString;
  La Multiplication de deux polynômes
- **Function** DiffPoly(Poly,Variable:Shortstring):Shortstring;
  La dérivée d'un polynôme
- **Function** DimduMatriceA(DenominateurduFT,Variable:ShortString):integer;
  Calcul la Dimension de la matrice d'état A à partir dénominateur de la fonction de transfert
- **Function** MatriceA_du_FT(Num,Deno,Variable:Shortstring):TableR;
  Calcul la matrice d'état A à partir numérateur et dénominateur de la fonction de transfert
- **Function** MatriceB_du_FT(Num,Deno,Variable:Shortstring):TableR;
  Calcul la matrice d'etat B à partir numérateur et dénominateur de la fonction de transfert
- **Function** MatriceC_du_FT(Num,Deno,Variable:Shortstring):TableR;
  Calcul la matrice d'état C à partir numérateur et dénominateur de la fonction de transfert
- **Function** MatriceD_du_FT(Num,Deno,Variable:Shortstring):TableR;
  Calcul la matrice d'état D à partir numérateur et dénominateur de la fonction de transfert
- **Function** DenoduFT(A:TableR;Variable:ShortString;n:Integer):ShortString;
  Calcul le Dénominateur de la fonction de transfert à partir la matrice d'état A
- **Function** NumeduFT(A,B,C,D:TableR;Variable:ShortString;n:Integer):ShortString;
  Calcul le numérateur de la fonction de transfert à partir des matrices d'états A, B, C et D

❖ **LIBRARY : LAGUERRE**

- **Function** PartReel(a:ShortString):ShortString;
  Partie réelle un nombre complexe
- **Function** PartImag(a:ShortString):ShortString;
  Partie imaginaire un nombre complexe
- **Function** ConjC(a:ShortString):ShortString;
  Le conjoin d'un nombre complexe
- **Function** AddC(a,b:ShortString):ShortString;
  La somme de deux nombres complexe
- **Function** SoustC(a,b:ShortString):ShortString;
  La soustraction de deux nombres complexe
- **Function** MultC(a,b:ShortString):ShortString;
  La multiplication de deux nombres complexe

- **Function** DivC(a,b:ShortString):ShortString;
  La division de deux nombres complexe
- **Function** CarreC(a:ShortString):ShortString;
  La carre d'un nombre complexe
- **Function** RacineCarreC(a:ShortString):ShortString;
  La racine carre d'un nombre complexe
- **Function** ModuleC(a:ShortString):Double;
  Le module d'un nombre complexe
- **Function** VecPoly(Poly,Variable:Shortstring):VecteurString;
  Vecteur dans polynôme
- **Function** NbrRacines(Poly,Variable:ShortString):integer;
  Nombre des racines d'un polynôme
- **Function** RacinesLaguerre(Poly,Variable:ShortString):VecteurString;
  Calcul les racines d'un polynôme par la méthode de Laguerre
- **Function** Arrondi(R:ShortString; m:integer):ShortString;
  Arrondissement d'un nombre réel
- **Function** ArrondiC(R:ShortString; m:integer):ShortString;
  Arrondissement d'un nombre complexe

❖ **LIBRARY : FONCTIONPOLYNOME**

- **Function** module (Poly,VarEntrer,VarSortie:Shortstring):Shortstring;
  Calcul le Module d'un polynôme
- **Function** moduleDecibel(Poly,VarEntrer,VarSortie:Shortstring):Shortstring;
  Calcul le module en Décibel d'un polynôme
- **Function** TermeArgument(Poly,VarEntrer,VarSortie:Shortstring):Shortstring;
  Calcul l'argument d'un polynôme
- **Function** TermePolynome(Poly,VarEntrer,VarSortie:Shortstring;index:integer):Shortstring;
  Calcul la partie réelle (index=1) ou imaginaire (index=2) d'un polynôme

❖ **LIBRARY : CALLEPOLYNOME**

- **Function** NbrdeRacinesReels(TRacines:VecteurString ;nbrRacine :Integer):Integer;
  Calcul le nombre des racines réelles d'après un vecteur contient des racines
- **Function** NbrdeRacinesImags(TRacines:VecteurString ;nbrRacine :Integer):Integer;
  Calcul le nombre des racines imaginaires d'après un vecteur contient des racines
- **Function** VecRacinesReels(TRacines:VecteurString ;nbrRacine :Integer):VecteurReel;
  Construire le vecteur qui contient les racines réelles à partir du vecteur des racines
- **Function** VecRacinesImags(TRacines:VecteurString ;nbrRacine :Integer):VecteurString;
  Construire le vecteur qui contient les racines imaginaires à partir du vecteur des racines
- **Function** VecPolyRacineReel(VecRacinesReels:VecteurReel;nbrRacineReel:integer): VecteurReel;
  Calcul le vecteur coefficient du polynôme qui correspond au multiplication (P-R1)*(P-R2)*...*(P-Rn) où R1,R2...Rn sont les racines réelles calculées d'après le vecteur des racines.
- **Function** DimVecPolyRacineReel(VecRacinesReels:VecteurReel;nbrRacineReel:integer):integer;
  Calcul la Dimension du vecteur calculé par la fonction VecPolyRacineReel
- **Function** TRacinesImagSeparer(VecRacineImag:VecteurString ;NbrdeRacineImag :Integer): TableReel;
  Construire la table qui contient la partie réelle et imaginaire des racines complexe séparer
- **Function** vecPolyRacineImag(TRacinesImagSeparer:TableReel ;NbrdeRacineImag :Integer):VecteurReel;
  Calcul le vecteur coefficient du polynôme qui correspond au multiplication (P-C1)*(P-C2)*...*(P-Cn) où C1,C2...Cn Sont les racines imaginaires calculées d'après le vecteur des racines.
- **Function** DimvecPolyRacineImag(TRacinesImagSeparer:TableReel; NbrdeRacineImag :Integer):Integer;
  Calcul la dimension du vecteur calcul par la fonction vecPolyRacineImag
- **Function** MultPoly(A,B:VecteurReel; m,n:integer):VecteurReel;
  Calcul le vecteur correspond à la multiplication des deux vecteurs A et B
- **Function** DimMultPoly(A,B:VecteurReel; m,n:integer):Integer;
  Calcul la Dimension du vecteur calcule par la fonction MultPoly
- **Function** Polynome(TRacines:VecteurString; nbrRacine:Integer; Variable: ShortString) :ShortString;
  Calcul le polynôme correspond au vecteur des racines directement.

Annexe III
Code source

Dans cette annexe on a proposé le code source complet de fichier DLL qui contient la fonction Laguerre et les fonctions qui l'accompagnent par Delphi7 :

```
Library Laguerre;

{Remarque importante concernant la gestion de mémoire de DLL : ShareMem doit être la première unité de la clause USES de votre bibliothèque ET de votre projet (sélectionnez Projet-Voir source) si votre DLL exporte des procédures ou des fonctions qui passent des chaînes en tant que paramètres ou résultats de fonction. Cela s'applique à toutes les chaînes passées de et vers votre DLL --même celles qui sont imbriquées dans des enregistrements et classes. ShareMem est l'unité d'interface pour le gestionnaire de mémoire partagée BORLNDMM.DLL, qui doit être déployé avec vos DLL. Pour éviter d'utiliser BORLNDMM.DLL, passez les informations de chaînes avec des paramètres PChar ou ShortString.}

Uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms, Dialogs, StdCtrls, Grids, math, math387;

{$R *.res}

Type
    VecteurString = array [1..100]of ShortString;
    VecteurReel = array[1..100]of Double;
    TableReel = array [1..100,1..5]of Double;
    TableString  =array[1..100,1..5] of Shortstring;
```

// La fonction qui calcul la partie réelle d'un nombre complexe

```
Function PartReel(a:ShortString):ShortString;
Var c1,c2,c3,PR,PI:Shortstring ;
   i,j,n:integer;
   T:Array[1..2]of Shortstring;
Begin
 n:=Length(a);
 j:=1;   T[1]:='';    T[2]:='';
  For i:=1 To n Do
   Begin
    if (a[i]<> ' ')then T[j] := T[j]+ a[i]
    else
     Begin
      j:=j+1;
      T[j] := T[j]+ a[i];
     End;
   End;
 C1:=T[1];
 C2:=T[2];
 n:=Length(C1);
if (C1[n]='i')then
  Begin
   C3:=C1;  C1:=C2;  C2:=C3;
  End;
 n:=Length(C2);
 C3:='';
for i:=1 to n-1 do C3:=C3+C2[i];
 if (C1='')then C1:='0';
 if (C2='')then C2:='0';
 if (C3='')then C3:='0';
 PR:=C1;
 PI:=C3;
 Result:=PR;
End;
Exports PartReel;
```

// La fonction qui calcul la partie imaginaire d'un nombre complexe

```
Function PartImag(a:ShortString):ShortString;
var c1,c2,c3,PR,PI:Shortstring ;
   i,j,n,n1:integer;
   T:Array[1..2]of Shortstring;
Begin
 n:=Length(a);
 j:=1;   T[1]:='';    T[2]:='';
  For i:=1 To n Do
   Begin
    if (a[i]<>' ') then
     T[j] := T[j]+ a[i]
    else
     Begin
      j:=j+1;
      if (j>=2)then j:=2;
     End;
   End;
```

```
 C1:=T[1];
 C2:=T[2];
 n:=Length(C1);
 if (C1[n]='i')then
  Begin
   C3:=C1;  C1:=C2;  C2:=C3;
  End;
 n:=Length(C2);
 C3:='';
 for i:=1 to n-1 do C3:=C3+C2[i];
 if (C1='')then C1:='0';
 if (C2='')then C2:='0';
 if (C3='')then C3:='0';
 PR:=C1;
 PI:=C3;
 Result:=PI;
End;
Exports PartImag;
```

// La fonction qui calcul le conjoint d'un nombre complexe

```
Function ConjC(a:ShortString):ShortString;
Var B,a1,a2,b1,b2:Shortstring ;
Begin
 a1:=PartReel(a);
 a2:=PartImag(a);
 b1:=FloattoStr(StrtoFloat(a1));
 b2:=FloattoStr(-StrtoFloat(a2));
 if (StrtoFloat(b2)<0)then
  B:=b1+' '+b2+'i'
 else
  B:=b1+' +'+b2+'i';
 Result:=B;
End;
Exports ConjC;
```

// La fonction qui calcul la somme de deux nombres complexes

```
Function AddC(a,b:ShortString):ShortString;
Var c,a1,a2,b1,b2,c1,c2:Shortstring ;
Begin
 a1:=PartReel(a);
 a2:=PartImag(a);
 b1:=PartReel(b);
 b2:=PartImag(b);
 c1:=FloattoStr(StrtoFloat(a1)+StrtoFloat(b1));
 c2:=FloattoStr(StrtoFloat(a2)+StrtoFloat(b2));
 if (StrtoFloat(c2)<0)then
  c:=c1+' '+c2+'i'
 else
  c:=c1+' +'+c2+'i';
 Result:=c;
End;
Exports AddC;
```

// La fonction qui calcul la soustraction de deux nombres complexes

```
Function SoustC(a,b:ShortString):ShortString;
Var c,a1,a2,b1,b2,c1,c2:Shortstring ;
Begin
 a1:=PartReel(a);
 a2:=PartImag(a);
 b1:=PartReel(b);
 b2:=PartImag(b);
 c1:=FloattoStr(StrtoFloat(a1)-StrtoFloat(b1));
 c2:=FloattoStr(StrtoFloat(a2)-StrtoFloat(b2));
 if (StrtoFloat(c2)<0)then
  c:=c1+' '+c2+'i'
 else
  c:=c1+' +'+c2+'i';
 Result:=c;
End;
Exports SoustC;
```

// La fonction qui calcul la multiplication de deux nombres complexes

```
Function MultC(a,b:ShortString):ShortString;
Var c,c1,c2:Shortstring ;
   a1,a2,b1,b2:Double;
Begin
 a1:=Strtofloat(PartReel(a));
 a2:=Strtofloat(PartImag(a));
 b1:=Strtofloat(PartReel(b));
 b2:=Strtofloat(PartImag(b));
 c1:=FloattoStr(a1*b1-a2*b2);
 c2:=FloattoStr(a1*b2+a2*b1);
 if (StrtoFloat(c2)<0)then
  c:=c1+' '+c2+'i'
 else
  c:=c1+' +'+c2+'i';
 Result:=c;
End;
Exports MultC;
```

// La fonction qui calcul la divisés de deux nombres complexes

```
Function DivC(a,b:ShortString):ShortString;
Var b1,c,c1,c2:Shortstring ;
   d1,d2,D:Double;
Begin
 b1:=ConjC(b);
 D:=StrtoFloat(PartReel(MultC(b,b1)));
 d1:=StrtoFloat(PartReel(MultC(a,b1)));
 d2:=StrtoFloat(PartImag(MultC(a,b1)));
 c1:=FloattoStr(d1/D);
 c2:=FloattoStr(d2/D);
 if (StrtoFloat(c2)<0)then
  c:=c1+' '+c2+'i'
 else
  c:=c1+' +'+c2+'i';
Result:=c;
End;
Exports DivC;
```

// La fonction qui calcul le carré d'un nombre complexe

```
Function CarreC(a:ShortString):ShortString;
Var c,c1,c2:Shortstring ;
   a1,a2:Double;
Begin
 a1:=Strtofloat(PartReel(a));
 a2:=Strtofloat(PartImag(a));
 c1:=FloattoStr(a1*a1-a2*a2);
 c2:=FloattoStr(2*a1*a2);
```

```
 if (StrtoFloat(c2)<0)then
  c:=c1+' '+c2+'i'
 else
  c:=c1+' +'+c2+'i';
 Result:=c;
End;
Exports CarreC;
```

// La fonction qui calcul la racine carré d'un nombre complexe

```
Function RacineCarreC(a:ShortString):ShortString;
Var c,c1,c2:Shortstring ;
    a1,a2,x,y:Double;
Begin
 a1:=Strtofloat(PartReel(a));
 a2:=Strtofloat(PartImag(a));
 x:=Sqrt((Sqrt(a1*a1+a2*a2)+a1)/2);
 y:=Sqrt((Sqrt(a1*a1+a2*a2)-a1)/2);
 if (a2<0)then y:=-y;
 c1:=FloattoStr(x);
 c2:=FloattoStr(y);
 if (StrtoFloat(c2)<0)then
  c:=c1+' '+c2+'i'
 else
  c:=c1+' +'+c2+'i';
 Result:=c;
End;
Exports RacineCarreC;
```

// La fonction qui calcul le module d'un nombre complexe

```
Function ModuleC(a:ShortString):Double;
Var a1,a2,x:Double;
Begin
 a1:=Strtofloat(PartReel(a));
 a2:=Strtofloat(PartImag(a));
 x:=Sqrt(a1*a1+a2*a2);
 Result:=x;
End;
Exports ModuleC;
```

// Une fonction qui calcul le vecteur des coefficients d'un polynôme écrie textuellement

```
Function
VecPoly(Poly,Variable:Shortstring):VecteurString;
 label PUISSNUM1,PUISSNUM2,CALCULE
,QUITTER;
 var  c,y,PR, PReelle, PI, PImag, Ctemp1, Ctemp2:
Shortstring;
  Tnumerateur,Temp : TableString;
  TB : TableReel;
  T:VecteurString;
  i,j,k,m,b,L,n,u,A,num,jnum,z,Gp :Integer;
Begin
////////////////////// SUPPRIMER LES CELLULES DES
TABLES ////////////////
   FOR j:=1 to 3 do
   FOR I:=1 to 50 do
     Tnumerateur[i,j] := '';

k:=0; m:=0; b:=0; L:=0; n:=0; u:=0; A:=0; num:=0;
jnum:=0; c:='';  y:='';
```

```
//////// SEPARER LES TERMES ET LES
COEFFICIENTS DU NUMERATEUR //////
 num:= Length(Poly);
 jnum:=1;
   For i:=1 To num Do
    Begin
     if (i=1)Then
      IF(Poly[1]='+')or(Poly[1]='-') then jnum:=jnum-1;
     if (Poly[i]<> '+')and(Poly[i]<> '-') then
       Tnumerateur[jnum,1] := Tnumerateur[jnum,1]+
Poly[i]
     else
      Begin
       jnum:=jnum+1;
       Tnumerateur[jnum,1] := Tnumerateur[jnum,1]+
Poly[i];
      End;
    End;
 k:=1;
 b:=1;
 For i:=1 To jnum Do
  Begin
   c := Tnumerateur[i,1]+' ';
   num := Length(c);
 For m:=1 To num Do
   BEGIN
    if (c[m] = '^') then    GOTO PUISSNUM2;
    if (c[1] = Variable) then    Tnumerateur[1,2]:='1';
    if (c[m] <> Variable) AND (c[m] <> ' ') then
     Tnumerateur[K,2]:=Tnumerateur[K,2]+ c[m]
    else
     BEGIN
      k:=k+1;
      if (c[m+1] <> '^') then    GOTO PUISSNUM1;
     End;
   End;
/// DETERMINER LES VALEURS DE PUISSANCE
POUR LES TERMES DU NUMERATEUR  //////
  PUISSNUM1 :
            Tnumerateur[b,3]:= '1';
  PUISSNUM2 :
        For m:= m+1 To num Do
           if ( c[m] <> ' ') then
            Tnumerateur[b,3]:= Tnumerateur[b,3]+
c[m];
     b:=b+1;
        End;
 For m:=1 To jnum Do
  Begin
   y:= Tnumerateur[m,2];
   u:= Length(y);
   IF (u = 1) then
    Begin
     IF (Tnumerateur[m,2] = '+')then
Tnumerateur[m,2] := '+1';
     IF (Tnumerateur[m,2] = '-')then
Tnumerateur[m,2] := '-1';
    End;
  End;
 For i:=1 To jnum Do
  Begin
   A := 0;
   C:= Tnumerateur[i,1];
```

```
  n := Length(C);
  For m:=1 To n Do
   IF (C[m] = Variable)then   A:=A+1;
  IF (A <> 1)then  Tnumerateur[i,3]:= '0';
 End;
 for j:=1 to jnum-1 do
  for i:=j+1 to jnum do
   Begin
    if
(StrtoFloat(Tnumerateur[i,3])<StrtoFloat(Tnumerateur
[j,3]))then
     Begin
      for Z:=1 to 3 do
       Begin
        Temp[j,Z]:=Tnumerateur[j,Z];
        Tnumerateur[j,Z]:=Tnumerateur[i,Z];
        Tnumerateur[i,Z]:=Temp[j,Z];
       End;
     End;
   End;
 for j:=1 to 2 do
  for i:=1 to 50 do  Temp[i,j]:='0';
 for i:=1 to 50 do
  Temp[i,3]:=InttoStr(i-1);
Gp:=0;
for i:=1 to jnum do
 if (StrtoInt(Tnumerateur[i,3])>Gp)then
  Gp:=StrtoInt(Tnumerateur[i,3]);
for i:=1 to jnum do
 for j:=0 to Gp+1 do
  if (StrtoInt(Tnumerateur[i,3])=j)then
    for k:=1 to 3 do
     Temp[j+1,k]:=Tnumerateur[i,k];
//// AFFICHER DANS DES TABLES //////
 for i:=1 to Gp+1 do
  Begin
   Ctemp1:=Temp[i,2];
   n:=Length(Ctemp1);
   IF (Ctemp1[n] = '*')then
    Begin
     Ctemp2:='';
     For m:=1 To n-1 Do
      Ctemp2 := Ctemp2 + Ctemp1[m];
    End
   else
    Ctemp2:=Ctemp1;
   Temp[i,2]:= Ctemp2;
  End;
 For i:=1 to Gp+1 do   T[Gp+2-i]:=Temp[i,2];
 Result:=T;
End;
exports VecPoly;
```

// Une fonction qui calcul le nombre des racines d'un polynôme écrie textuellement

```
Function
NbrRacines(Poly,Variable:ShortString):integer;
var
  c,y,StrR: Shortstring;
  i,j,k,E,m,b,n,NN1,u,A,num,jnum,Gpnum :Integer;
    Tnumerateur : TableString;
 Label PUISSNUM1, PUISSNUM2, CALCULE,
QUITTER;
//////// SEPARER LES TERMES ET LES
COEFFICIENTS DU NUMERATEUR //////
Begin
 num:= Length(Poly);
 jnum:=1;
    For i:=1 To num Do
     Begin
      if (i=1)Then
       IF(Poly[1]='+')or(Poly[1]='-') then jnum:=jnum-1;
      if (Poly[i]<> '+')and(Poly[i]<> '-') then
        Tnumerateur[jnum,1] := Tnumerateur[jnum,1]+
Poly[i]
       else
        Begin
         jnum:=jnum+1;
         Tnumerateur[jnum,1] := Tnumerateur[jnum,1]+
Poly[i];
        End;
     End;
 k:=1;
 b:=1;
 For i:=1 To jnum Do
  Begin
   c := Tnumerateur[i,1]+' ';
   num := Length(c);
 For m:=1 To num Do
  Begin
    if (c[m] = '^') then    GOTO PUISSNUM2;
    if (c[1] = Variable) then    Tnumerateur[1,2]:='1';
    if (c[m] <> Variable) AND (c[m] <> ' ') then
      Tnumerateur[K,2]:=Tnumerateur[K,2]+ c[m]
    else
      Begin
        k:=k+1;
        if (c[m+1] <> '^') then    GOTO PUISSNUM1;
      End;
  End;
// DETERMINER LES VALEURS DE PUISSANCE
POUR LES TERMES DU NUMERATEUR //////
 PUISSNUM1 :
            Tnumerateur[b,3]:= '1';
 PUISSNUM2 :
        For m:= m+1 To num Do
           if ( c[m] <> ' ') then
            Tnumerateur[b,3]:= Tnumerateur[b,3]+ c[m];
    b:=b+1;
        End;
 For m:=1 To jnum Do
  Begin
   y:= Tnumerateur[m,2];
   u:= Length(y);
   IF (u = 1) then
    Begin
     IF (Tnumerateur[m,2] = '+')then
Tnumerateur[m,2] := '+1';
     IF (Tnumerateur[m,2] = '-')then
Tnumerateur[m,2] := '-1';
    End;
  End;
 For i:=1 To jnum Do
  Begin
```

```
   A := 0;
   C:= Tnumerateur[i,1];
   n := Length(C);
   For m:=1 To n Do
    IF (C[m] = Variable)then   A:=A+1;
   IF (A <> 1)then  Tnumerateur[i,3]:= '0';
  End;
/////// la valaur de la grande puisssance ////////
  if (Poly<>'')then
   Begin
    Gpnum:=0;
    for i:=1 to jnum do
     if (StrtoInt(Tnumerateur[i,3])>Gpnum)then
      Gpnum:=StrtoInt(Tnumerateur[i,3]);
 Result:=Gpnum;
End;
End;
Exports NbrRacines;
```

// La fonction qui calcul les racines réelles et complexes d'un polynôme écrie textuellement

```
Function
RacinesLaguerre(Poly,Variable:ShortString;Erreur,
repeter:Double):VecteurString;
 var  i,j,m,rr,k,n:integer;
     R,x,y,y1,y2,G,T,Q,V,H,a,s,T2:ShortString;
     TCoeff,TRes:VecteurString;
     c,c1,b,b1,d1,d2:Double;
 Label Post0, Post1, Post2, Post3, Post4, Post5, Post6,
Post7, Post8;
Begin
  n:=NbrRacines(Poly,Variable);
  TCoeff:=VecPoly(Poly,Variable);
////////// Programme Laguerre ////////////////////////////
  k:=n+1;
Post0:
// R:=inttoStr(Random(10));
  R:='0';
  if(n<1)then Goto Post4;
  rr:=0;
Post1:
  rr:=rr+1;
  ///////////////  SousProgramme de diff ////////
  m:=n+1;
  y:='0';y1:='0';y2:='0';
  for i:=1 to m do
   Begin
    y:=AddC(MultC(y,R),TCoeff[i]);
    if (i<m)then
y1:=AddC(MultC(y1,R),MultC(InttoStr(m-i),
TCoeff[i]));
    if (i<n)then
y2:=AddC(MultC(y2,R),MultC(InttoStr((m-i)*(n-
i)),TCoeff[i]));
   End;
//////////////////////////////////////////////////////
  if(ModuleC(y)<Erreur)then Goto Post2;
  if(rr>repeter)then Goto Post0;
  G:=DivC(y1,y);
  H:=SoustC(CarreC(G),DivC(y2,y));
  T:=MultC(InttoStr(n-1),
SoustC(MultC(InttoStr(n),H), CarreC(G)));
  T:=RacineCarreC(T);
  Q:=SoustC(G,T);
  V:=AddC(G,T);
  if (ModuleC(V)>ModuleC(Q))Then Q:=V;
  a:='1';
  if(Q<>'0 +0i')then a:=DivC(inttoStr(n),Q);
  R:=SoustC(R,a);
  Goto Post1;
Post2:
  S:=PartReel(R);
  T2:='0';
  j:=1;
  TRes[n]:=s;
  if(ModuleC(PartImag(R))>1E-6)then
   Begin
    TRes[n]:=R;
    TRes[n-1]:=ConjC(R);
    s:=AddC(s,s);
    T2:=MultC(R,ConjC(R));
    j:=2;
   End;
/////////// SousProgramme Divpolynome ///////
  TCoeff[2]:=AddC(TCoeff[2],MultC(s,TCoeff[1]));

TCoeff[3]:=AddC(TCoeff[3],SoustC(MultC(s,TCoeff[
2]),MultC(T2,TCoeff[1])));
  For i:=4 to n+1 do

TCoeff[i]:=AddC(TCoeff[i],SoustC(MultC(s,TCoeff[i
-1]),MultC(T2,TCoeff[i-2])));
  n:=n-j;
  Goto post0;
Post4:
//////// Précisés les résultats obtenus  //////
 for i:=1 to k-1 do
  Begin
   c:=StrtoFloat(PartReel(TRes[i]));
   b:=StrtoFloat(PartImag(TRes[i]));

   d1:=abs(Trunc(c))-abs(c);
   d2:=abs(Trunc(b))-abs(b);
   c1:=c;
   b1:=b;
   if (abs(d1)<1E-12)then  c1:=Trunc(c);
   if (1-abs(d1)<1E-12)then  c1:=abs(Trunc(c))+1;
   if (abs(d2)<1E-12)then b1:=Trunc(b);
   if (1-abs(d2)<1E-12)then b1:=Trunc(b)+1;
   if (c1*c<0)then c1:=-c1;
   if (b1*b<0)then b1:=-b1;
   if(c1=0)and(b1=0)then TRes[i]:='0';
   if(c1=0)and(b1<>0)then
   if (b1>0)then TRes[i]:='+'+FloattoStr(b1)+'i'
    else  TRes[i]:=FloattoStr(b1)+'i';
   if(c1<>0)and(b1=0)then TRes[i]:=FloattoStr(c1);
   if(c1<>0)and(b1<>0)then
   if (b1>0)then  TRes[i]:=FloattoStr(c1)+'
+'+FloattoStr(b1)+'i'
    else  TRes[i]:=FloattoStr(c1)+' '+FloattoStr(b1)+'i';
  End;
//////////  fin de la présision //////////
 Result:=TRes;
End;
Exports RacinesLaguerre;
```

MEMOIRE DE FIN D'ETUDES EN VUE DE L'OBTENTION DU DIPLOME

D'INGENIEUR D'ETAT EN GENIE ELECTROTECHNIQUE

OPTION : ELECTROMECANIQUE

Proposé et dirigé par :

Mr. Yahia LAAMARI

Présenté par : Fawzi Mourad BISKER & Badreddine LADJAL

THEME :

DEVELOPPEMENT D'UN LOGICIEL D'ANALYSE ET DE SYNTHESE DES SYSTEMES ASSERVIS LINEAIRES CONTINUS MONOVARIABLES

RESUME :

Le présent travail porte sur l'élaboration d'un logiciel, baptisé "ProTahleel", pour l'analyse des systèmes linéaires, continus et monovariables préalablement modélisés et représentés par leur fonction de transfert, la constellation de leurs pôles et zéros ou dans l'espace d'état. Le logiciel permet également de réaliser les tests de contrôlabilité et d'observabilité.

Ce logiciel propose un large éventail de méthodes d'analyse aussi bien dans le domaine fréquentiel (Bode, Black-Nickols, Nyquist, Evans et représentation tridimensionnelle), que dans le domaine temporel (réponse en boucle ouverte, en boucle fermée, réponse impulsionnelle…).

Il réalise aussi un certain nombre de contrôles suivant des critères couramment utilisés en matière de systèmes asservis pour juger leurs performances.

ProTahleel Possède en plus la possibilité de faire une synthèse simple des systèmes linéaires.

MOTS CLES :

Asservissements linéaires, systèmes continus invariants monovariable, représentation d'état, fonction de transfert, Pôles et zéros, analyse, observabilité, contrôlabilité, Laguerre, Rung-Kutta4, synthèse, régulateur PID.