Titre: Title:	Initialisation automatique des systèmes de contrôle pour la simulation des transitoires de réseau
Auteur: Author:	Diane Desjardins
Date:	2019
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Desjardins, D. (2019). Initialisation automatique des systèmes de contrôle pour la simulation des transitoires de réseau [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/4012/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Initialisation automatique des systèmes de contrôle pour la simulation des transitoires de réseau

DIANE DESJARDINS
Département de génie électrique

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie électrique

Août 2019

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Initialisation automatique des systèmes de contrôle pour la simulation des transitoires de réseau

présenté par **Diane DESJARDINS**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Keyhan SHESHYEKANI, président
Jean MAHSEREDJIAN, membre et directeur de recherche
Houshang KARIMI, membre et codirecteur de recherche
Omar SAAD, membre et codirecteur de recherche
Christian DUFOUR, membre

REMERCIEMENTS

Je tiens à remercier vivement mon directeur de recherche Jean Mahseredjian pour m'avoir donné l'opportunité de travailler sur ce sujet qui m'a beaucoup intéressé et pour m'avoir suivi et aiguillé tut le long de ma maîtrise. Je souhaite aussi remercier mon codirecteur Houshang Karimi pour les informations et exemples qu'il m'a apporté sur le contrôle et les micro-réseaux. Enfin je voudrais remercier mon deuxième codirecteur Omar Saad pour son apport sur la simulation des fonctions de transfert.

Je tiens aussi à saluer tous les étudiants de la chaire, son post-doc et le professeur Aki Ametani. La bonne entente qui règne au sein du laboratoire a pour sûr participé au bon déroulement de ma maîtrise.

Enfin, je remercie Bob, Théodore et Gwenael pour leur soutient jour et nuit.

RÉSUMÉ

Un réseau électrique est normalement en régime permanent. Ainsi c'est dans cet état que commencent la plupart des simulations de réseau dans le domaine du temps. Cependant, si la partie purement électrique d'un réseau peut être initialisée via un calcul d'écoulement de puissance, il n'y a pas de solution équivalente pour les systèmes de contrôle.
Ce mémoire s'applique à toute méthode de calcul de type EMT (*Electromagnetic Transient*). Celui-ci possède une fonctionnalité pour modéliser les systèmes de contrôle sous forme de diagramme-blocs dont les variables peuvent être initialisées manuellement. Une erreur ou un manque à ce niveau va créer un régime transitoire virtuel en début de simulation, le temps que le système converge vers son régime permanent. Afin d'éviter ce problème, ce mémoire va présenter un algorithme visant à initialiser automatiquement un système de contrôle à partir des quelques données disponibles grâce au calcul d'écoulement de puissance.

Les fonctions de transfert, composants majeurs des systèmes de contrôle, sont étudiées en premier lieu. Une méthode de simulation réduisant les problèmes d'instabilité numérique est proposée avec l'initialisation correspondante. Son utilité est démontrée sur un cas présentant une erreur numérique sur la simulation faite avec le logiciel EMTP (*Electromagnetic Transient Program*) à très faible pas de temps. La démarche proposée est enfin généralisée aux fonctions de transfert à plusieurs entrées et sorties. Ensuite, cette approche est combinée à la modélisation d'autres éléments présents dans les systèmes de contrôle afin de construire un procédé de simulation avec une initialisation en régime permanent. Le principe est de construire une équation matricielle représentant le comportement du système de contrôle. Cela fait intervenir les variables de ce dernier au pas de temps précédent. Cet historique est initialisé en régime permanent grâce aux données obtenues de l'écoulement de puissance. L'implémentation de la méthode donne un algorithme permettant de simuler une grande variété de diagramme-blocs avec un départ en régime permanent. L'efficacité du procédé d'initialisation est, finalement, démontré dans le cas d'un micro-réseau. Ce dernier change de contrôleur lors de la déconnexion du réseau principal. Il est montré qu'initialiser correctement le nouveau contrôleur permet d'éviter de fortes variations de tensions mesurées dans le cas contraire.

ABSTRACT

As electrical networks are usually in steady-state, it is typically from this state that most time-domain simulations are started. However, if the power parts can be initialized via a power flow calculation, there is no equivalent solution for control systems. The electrical network time-domain simulation software Electromagnetic Transient Program (EMTP) has a functionality to model control systems in the form of block-diagrams whose variables can be initialized manually. An error or a lack at this level will create a virtual transient at the beginning of simulation while the system converges to its steady-state. To avoid this problem, this thesis will present an algorithm which automatically initialize a control system from data available from the steady-state solution.This works is fully applicable to other EMT-type simulation methods and its validity is compared with EMTP (Electromagnetic Transient Program).

Transfer functions are studied first as they are major components of control systems. A simulation method that reduces numerical instability is proposed with the corresponding initialization. Its utility is demonstrated on a case presenting a numerical error on EMTP simulation made with a very small time step. The proposed approach is finally generalized to transfer functions with several inputs and outputs. Then, this method is combined with the modeling of other elements present in control systems in order to build a simulation process with steady-state initialization. The principle is to build a matrix equation representing the control system behavior. This involves using variables of the latter at the previous time step. This history is initialized in steady state thanks to data obtained from the power flow. The method implementation gives an algorithm allowing simulation of block-diagrams from steady-state initialization. The effectiveness of the initialization method is finally demonstrated in the case of a microgrid. The latter changes controller when it disconnects from the main network. It is shown that correctly initializing the new controller avoids large voltages variations measured otherwise.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

EMTP	*Electromagnetic Transient Program*
MIMO	*Multi-Input Multi-Output* (Multi-Entrées Multi-Sorties)
PI	Proportionnel Intégral
VSC	*Voltage Source Converter* (Convertisseur source de tension)

LISTE DES ANNEXES

CHAPITRE 1 INTRODUCTION

1.1 Mise en contexte

La planification, le développement et l'analyse d'un réseau électrique ne peuvent se faire sans l'aide d'outils de simulation. Ils permettent d'étudier le comportement du réseau et les limites de sa stabilité ou encore prévenir un mal-fonctionnement s'il est soumis à un incident ou à une modification. Ils viennent palier à l'impossibilité de faire des tests directement sur le réseaux. La simulation d'un réseau électrique nécessite non seulement la modélisation de ses composants électriques mais aussi des éléments de contrôle. Ces derniers visent à réguler les différentes grandeurs du réseau (tension, courant, fréquence). Les composants typiques sont les excitatrices des machines synchrones, les contrôles des liaisons HTCC et des systèmes électroniques de puissance.

Les applications de la simulation des réseaux électriques et de leurs systèmes de contrôle nécessitent une précision adaptée. Selon les échelles et le but visé, la modélisation peut rester simple ou au contraire être raffinée pour être au plus proche des grandeurs réellement mesurées. Mais il faut en aucun cas que la simulation soit perturbée par des problèmes d'instabilité numérique. Pour cela des modèles plus robustes sont nécessaires. Cependant de tels modèles nécessitent des calculs plus lourds et un temps de simulation plus long. Ce qui peut être problématique dans le cas d'un réseau de taille conséquente.

Les différents types de simulations et modélisations sont très nombreux. Dans certains cas une étude en régime permanent peut être voulue, alors que dans d'autres cas une analyse dans le domaine du temps est nécessaire. Dans ce cas il est souvent souhaité de commencer la simulation en régime permanent. En effet il s'agit de l'état normal du système qu'il faut modéliser avant de simuler l'effet de perturbations. Si la simulation n'est pas initialisée en régime permanent, il faudra alors attendre que le système converge vers cet état avant de faire les tests voulus. Ce temps de simulation dédié à l'atteinte du régime permanent représente du temps de calcul inutile. Une bonne initialisation est donc importante.

Cette maîtrise se place dans le cadre des simulations des réseaux électriques de type EMT (*Electromagnetic Transient*). Celui-ci possède une fonctionnalité pour modéliser les systèmes de contrôle des réseaux grâce à un module spécialisé de simulations. Dans ce logiciel les variables du circuit électrique sont initialisées automatiquement à partir d'un calcul de régime permanent. Cependant les variables du système de contrôle doivent, elles, être initialisées manuellement. Cela pose deux problèmes. Tout d'abord, alors qu'il est souhaité que la si-

mulation démarre en régime permanent, une erreur au niveau de l'initialisation va créer un faux régime transitoire en début de simulation. De plus ces variables peuvent s'avérer très nombreuses sur des systèmes de taille conséquente, rendant l'initialisation manuelle pénible pour l'utilisateur. Ainsi, le besoin d'une initialisation automatique se fait sentir pour prévenir ces éventualités.

1.2 Problématique

La réalisation d'une initialisation automatique dans le logiciel EMTP, et de type EMT, doit prendre en compte plusieurs particularités D'une part, Il est important de garder en tête la variétés des éléments pouvant être modélisés (fonctions de transfert, forme d'état, gain, sommateur...). En effet il est possible que l'utilisateur définisse ses propres systèmes, ainsi une solution générique pouvant s'adapter à une multitude d'éléments de configuration est nécessaire. D'autre part, il est fondamental de bien comprendre la façon dont sont modélisés les système de contrôle au sein du logiciel. Sans cela, l'initialisation développée a peu de chance d'être compatible avec le fonctionnement du logiciel. Cependant dans certains cas, la simulation du contrôle entraîne des problèmes d'instabilité. La résolution du problème d'initialisation est alors l'opportunité de revisiter la modélisation de certains éléments. Cette dernière doit rester conforme à la méthode utilisée pour la simulation du contrôle dans EMTP. De plus, elle ne doit pas détériorer la précision du logiciel et permettre d'éviter certaines instabilités numériques.

1.3 Revue de littérature

1.3.1 Cas d'initialisation de systèmes de contrôle

Il existe de nombreux cas d'initialisations manuelles où les variables sont calculées préalablement à la simulation pour que celle-ci commence en régime permanent. C'est le cas pour les éoliennes ([**?**], [**?**]). Dans ces exemples, un calcul d'écoulement de puissance est appliqué sur la partie électrique. Il s'agit d'une simulation préalable permettant d'obtenir la tension et le courant de tous les nœuds d'un réseau électrique. Du point de vue du contrôle, cela permet d'obtenir quelques données en régime permanent, souvent la sortie du système de commande. Ensuite, les équations du système de contrôle, de la génératrice et de la dynamique des éoliennes sont déroulées pour calculer une à une toutes les variables en régime permanent. Ce type d'initialisation est spécifique au cas étudié et ne peut pas être généralisé.

Des études ont été menées afin d'initialiser des types de système de contrôle dans EMTP.

Il s'agit d'initialisation dite "semi-automatique". C'est le cas pour le contrôleur de machines synchrones ([**?**]). Comme pour les exemples précédents, le calcul des variables du système de contrôle en régime permanent est basé sur les résultats de l'écoulement de puissance. Ce dernier donne la sortie du système de contrôle, ce qui permet de déduire les autres variables en déroulant les équation du diagramme-bloc du système à rebours. Bien que plus complexe, l'initialisation d'une liaison courant-continu ([**?**]) suit le même principe. Ce type d'initialisation n'est pas adapté à un système de contrôle quelconque que pourrait définir l'utilisateur.

Enfin, un cas d'initialisation automatique a été développé avec le langage Modelica ([**?**]). Les équations du système sont manipulées pour créer un matrice Jacobienne afin d'initialiser le système d'après les données de l'écoulement de puissance. Toutefois, l'algorithme n'est pas applicable au logiciel EMTP dû aux particularités du langage.

1.3.2 Méthode de simulation des systèmes de contrôles dans EMTP

Le logiciel EMTP permet la simulation des transitoires électromécaniques, électromagnétiques et des systèmes de contrôle. Son fonctionnement est détaillé par [**?**].

Le principe de la simulation des système de contrôle réside dans la résolution à chaque pas de temps d'un système matriciel représentant le comportement du système. L'approche actuelle permet d'avoir une solution simultanée du système de contrôle ([**?**]). Elle fait appel à la méthode de Newton en introduisant la Jacobienne du système.

1.4 Objectifs de recherche

L'objectif principal de ce mémoire est de créer un algorithme permettant une simulation du système de contrôle avec une initialisation automatique des variables en régime permanent. Pour ce faire, il faudra tout d'abord, redéfinir la modélisation des fonctions de transfert afin de résoudre certains problème d'instabilité numérique. Ensuite, les problèmes d'initialisation en régime permanent de la simulation des diagrammes-blocs représentant un système de contrôle associé à un réseau électrique devront être résolus.

1.5 Méthodologie

Étude de la simulation d'une fonction de transfert

La modélisation des systèmes de contrôle avec EMTP fait très souvent intervenir des fonctions de transfert. Il est donc nécessaire de les étudier en profondeur. De plus des cas

d'erreurs numériques ont été observées sur des fonctions de transfert simples (d'ordre faible), montrant des limites de la méthode actuelle de simulation de ces fonctions. Le but de cette première étape est de proposer une nouvelle méthode de simulation pour les fonctions de transfert présentant une meilleure stabilité ainsi que les équations d'initialisation adaptées.

Les fonctions de transfert scalaires dans le domaine de Laplace (temps continu) ont été étudiées. Plusieurs méthodes de modélisation ont été testées afin de sélectionner la plus performante. Ensuite le processus a été généralisé aux fonctions matricielles et aux fonctions de transfert en temps discrétisé (domaine en Z).

Création d'un algorithme d'initialisation des fonctions de contrôle linéaires

A partir de l'étude d'une fonction de transfert, un algorithme est créé permettant de simuler un ensemble de fonctions de transfert agencées ensemble avec l'initialisation appropriée. Cela nécessite de définir une méthode de simulation du diagramme-bloc en accord avec la méthode de simulation utilisée sur EMTP (résolution d'un système matriciel). Ensuite le fonctionnement de l'algorithme a été conçu pour générer les éléments nécessaire à la simulation voulue (matrices du système à résoudre à chaque pas de temps et variables initialisées). Afin de pouvoir tester l'algorithme sur des cas réels et de généraliser son utilisation d'autres fonctions ont été étudiées afin qu'elles puissent être traités : sommes, gains, formes d'état et limiteurs. De plus l'algorithme a tout d'abord été implémenté pour des variables scalaires avant d'être modifié pour être en mesure de traiter les variables vectorielles.

Etude d'un cas pratique

Afin de valider l'utilité de l'initialisation et l'efficacité de la méthode développée, un cas pratique va être étudié. Il s'agit de l'initialisation d'un contrôleur de micro-réseau lorsque celui-ci est déconnecté du réseau principal.

1.6 Plan du mémoire

La structure du mémoire suit les grandes étapes de recherche présentées dans la méthodologie. Ainsi, le prochain chapitre aborde les fonctions de transfert. Une méthode de simulations pour celles-ci sera présentée et son utilité sera démontrée sur un exemple. Ensuite le troisième chapitre présentera l'algorithme d'initialisation. Son fonctionnement global sera expliqué avant d'être illustré sur un exemple. Enfin, le quatrième chapitre, proposera l'application pratique de l'initialisation sur un micro-réseau.

CHAPITRE 2 SIMULATION D'UNE FONCTION DE TRANSFERT

Dans ce chapitre, une manière de simuler les fonctions de transfert va être définie de façon à résoudre des problèmes d'instabilité rencontrés dans EMTP et les logiciels utilisant des approches similaires. Dans un premier temps, la simulation d'une fonction de transfert scalaire dans le domaine de Laplace sera définie. Ensuite la méthode proposée sera testée sur un cas particulier de fonction de transfert simple dont la simulation pose problème avec EMTP. Enfin la méthode choisie sera généralisée aux fonctions de transfert matricielles et au domaine en Z.

2.1 Fonction de transfert du domaine de Laplace scalaire

Le but de l'étude est de déterminer une méthode performante pour simuler une fonction de transfert scalaire en temps continu (domaine de Laplace). Pour cela, une transformation d'une fonction de transfert à une forme d'état va être choisie ainsi qu'une manière de passer du temps continu au temps discrétisé.

2.1.1 Forme d'état testées

Le but est de transformer la fonction de transfert :

$$H(s) = \frac{Y(s)}{U(s)} = \frac{b_0 s^n + b_1 s^{n-1} + \cdots + b_{n-1} s + b_n}{s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n} \tag{2.1}$$

en la forme d'état :

$$\begin{cases} \dot{\mathbf{X}} = \mathbf{A}\mathbf{X} + \mathbf{B}u \\ y = \mathbf{C}\mathbf{X} + Du \end{cases} \tag{2.2}$$

Les variables internes de cette dernière miment les dérivations successives de l'équation différentielle représentée par la fonction de transfert. Trois passages de la fonction de transfert vers une forme d'état ont été étudiés.

Première transformation

Exemple : ([?])

Soit la fonction de transfert :

$$H(s) = \frac{Y(s)}{U(s)} = \frac{4s^2 + 5s + 6}{s^2 + 2s + 3} \tag{2.3}$$

Dans le domaine du temps, cela donne :

$$\ddot{y} + 2\dot{y} + 3y = 4\ddot{u} + 5\dot{u} + 6u \tag{2.4}$$

On pose les états intermédiaires x_1 et x_2 suivants avec β_0 et β_1 des constantes dont la valeur sera choisie par la suite :

$$x_1 = y - \beta_0 u \tag{2.5}$$
$$x_2 = \dot{y} - \beta_0 \dot{u} - \beta_1 u = \dot{x}_1 - \beta_1 u \tag{2.6}$$

On veut exprimer $\dot{x}_2$ en fonction de x_1, x_2 et u seulement. On dérive l'expression de x_2 :

$$\dot{x}_2 = \ddot{y} - \beta_0 \ddot{u} - \beta_1 \dot{u} \tag{2.7}$$

On va tout d'abord remplacer $\ddot{y}$ par une expression faisant intervenir x_1, x_2 et u et ses dérivées. Pour cela on utilise l'équation différentielle qui donne :

$$\ddot{y} = 4\ddot{u} + 5\dot{u} + 6u - 2\dot{y} - 3y \tag{2.8}$$
$$\ddot{y} = 4\ddot{u} + 5\dot{u} + 6u - 2(x_2 + \beta_0 \dot{u} + \beta_1 u) - 3(x_1 + \beta_0 u) \tag{2.9}$$

En injectant dans l'expression de $\dot{x}_2$, on a :

$$\dot{x}_2 = -2x_2 - 3x_1 + (4 - \beta_0)\ddot{u} + (5 - 2\beta_0 - \beta 1)\dot{u} + (6 - 2\beta_1 - 3\beta_0)u \tag{2.10}$$

Afin de faire $\dot{u}$ et $\ddot{u}$, on choisit :

$$\beta_0 = 4 \tag{2.11}$$
$$\beta_1 = 5 - 2\beta_0 = -3 \tag{2.12}$$
$$\beta_2 = 6 - 2\beta_1 - 3\beta_0 = 0 \tag{2.13}$$

On obtient alors :

$$\dot{x}_2 = -2x_2 - 3x_1 \tag{2.14}$$

D'après la définition des état x_1 et x_2, on a :

$$y = x_1 + \beta_0 u = x_1 + 4u \tag{2.15}$$
$$\dot{x}_1 = x_2 + \beta_1 u = x_2 - 3u \tag{2.16}$$

La forme d'état obtenue est :

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -3 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -3 \\ 0 \end{bmatrix} u \\ \quad y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 4u \end{cases} \tag{2.17}$$

Cas général :

On part de la fonction de transfert dans le cas général :

$$H(s) = \frac{Y(s)}{U(s)} = \frac{b_0 s^n + b_1 s^{n-1} + \cdots + b_{n-1} s + b_n}{s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n} \tag{2.18}$$

où les a_i et b_i sont respectivement les coefficients du dénominateur et du numérateur de la fonction de transfert. On pose alors les états suivant :

$$x_1 = y - \beta_0 u \tag{2.19}$$
$$\text{avec } \beta_0 = b_0 \tag{2.20}$$
$$x_i = y^{(i-1)} - \sum_{j=0}^{i-1} \beta_j u^{i-1-j},\ 2 \leq i \leq n \tag{2.21}$$
$$\text{avec } \beta_i = b_i - \sum_{j=0}^{i-1} a_{i-j} \beta_j, 1 \leq i \leq n \tag{2.22}$$

De la même façon que pour l'exemple, cela donne une relation simple entre $\dot{x}_i$ et x_{i+1}, et l'expression de $\dot{x}_n$ est obtenue à partir de l'équation différentielle représentée par la fonction de transfert. On obtient ainsi :

$$\dot{x}_i = x_{i+1} - a_{n-i}x_n + \beta_i u,\ 1 \leq i \leq n-1 \tag{2.23}$$

$$\dot{x}_n = y^{(n)} - \sum_{j=0}^{n-1} \beta_j u^{n-j} = -\sum_{i=1}^{n} a_{n-i+1}x_i + \beta_n u \tag{2.24}$$

Soit sous forme matricielle :

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ -a_n & -a_{n-1} & \cdots & -a_1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_n \end{bmatrix} u \\ y = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + b_0 u \end{cases} \tag{2.25}$$

Deuxième transformation

([**?**])

Exemple :

On reprend la même fonction de transfert que précédemment.

On a donc la même équation différentielle :

$$\ddot{y} + 2\dot{y} + 3y = 4\ddot{u} + 5\dot{u} + 6u \tag{2.26}$$

On pose :

$$x_2 = y - 4u \tag{2.27}$$

$$x_1 = \dot{y} + 2y - 4\dot{u} - 5u \tag{2.28}$$

En dérivant la dernière expression :

$$\dot{x}_2 = \dot{y} - 4\dot{u} \tag{2.29}$$

Soit, d'après les expression de x_1 et x_2 :

$$\dot{x}_2 = x_1 - 2x_2 + (5 - 2 \cdot 4)u \tag{2.30}$$

De plus, en dérivant l'expression de x_1, on a :

$$\dot{x}_1 = \ddot{y} + 2\dot{y} - 4\ddot{u} - 5\dot{u} \tag{2.31}$$

D'après l'équation différentielle, cela donne :

$$\dot{x}_1 = 6u - 3y \tag{2.32}$$

D'après la définition de x_2 :

$$\dot{x}_1 = -3x_2 + (6 - 3 \cdot 4)u \tag{2.33}$$

On encore :

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & -3 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -6 \\ -3 \end{bmatrix} u \\ \quad y = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 4u \end{cases} \tag{2.34}$$

Cas général On repart encore une fois de la fonction de transfert dans le cas général 2.1.

On pose les états :

$$x_n = y - b_0 u \tag{2.35}$$

$$x_{n-i} = y^{(i)} - \sum_{j=1}^{i} a_j y^{(i-j)} - \sum_{j=0}^{i} b_j u^{(i-j)},\ 1 \leq i \leq n \tag{2.36}$$

où les a_i et b_i sont les coefficients de la fonctions de transfert. Soit :

$$\dot{x}_{i+1} = x_i - a_{n-i}x_n + (b_{n-i} - a_{n-i}b_0)u \tag{2.37}$$

$$\dot{x}_1 = y^{(n)} + \sum_{j=1}^{n-1} a_j y^{(n-j)} - \sum_{j=0}^{n-1} b_j u^{(n-j)} = -a_n x_n + (b_n - a_n b_0)u \tag{2.38}$$

Soit sous forme matricielle :

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 0 & -a_n \\ 1 & 0 & \cdots & 0 & -a_{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -a_1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_n - a_n b_0 \\ \vdots \\ b_1 - a_1 b_0 \end{bmatrix} u \\ y = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + b_0 u \end{cases} \tag{2.39}$$

Troisième transformation

Exemple : [**?**]
On reprend la même fonction de transfert que précédemment.

On a toujours :

$$\ddot{y} + 2\dot{y} + 3y = 4\ddot{u} + 5\dot{u} + 6u \tag{2.40}$$

De façon similaire à la méthode précédente, on pose :

$$x_1 = y - 4u \tag{2.41}$$

$$x_2 = \dot{y} + 2y - 4\dot{u} - 5u \tag{2.42}$$

Cela donne :

$$\dot{x}_1 = x_2 - 2x_1 + (5 - 2 \cdot 4)u \tag{2.43}$$

$$\dot{x}_2 = \ddot{y} + 2\dot{y} - 4\ddot{u} - 5\dot{u} = 6u - 3y \tag{2.44}$$

$$\dot{x}_2 = -3x_1 + (6 - 3 \cdot 4)u \tag{2.45}$$

On encore :

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -2 & 1 \\ -3 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} -3 \\ -6 \end{bmatrix} u \\ \quad y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + 4u \end{cases} \tag{2.46}$$

Cas général Encore une fois on reprendre la fonction de transfert générale 2.1.

On pose :

$$x_1 = y - b_0 u \tag{2.47}$$

$$x_{n-i} = y^{(i)} - \sum_{j=1}^{i} a_j y^{(i-j)} - \sum_{j=0}^{i} b_j u^{(i-j)},\ 1 \leq i \leq n \tag{2.48}$$

Soit :

$$\dot{x}_{i+1} = x_i - a_{n-i} x_n + (b_{n-i} - a_{n-i} b_0) u \tag{2.49}$$

$$\dot{x}_n = y^{(n)} + \sum_{j=1}^{n-1} a_j y^{(n-j)} - \sum_{j=0}^{n-1} b_j u^{(n-j)} = -a_n x_n + (b_n - a_n b_0) u \tag{2.50}$$

Soit sous forme matricielle :

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} -a_1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n-1} & 0 & \cdots & 1 \\ -a_n & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 - a_1 b_0 \\ \vdots \\ b_n - a_n b_0 \end{bmatrix} u \\ \quad y = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + b_0 u \end{cases} \tag{2.51}$$

Remarque : Les formes d'état proposées ici sont toutes observables. Il existe aussi des formes contrôlables mais le choix a été fait de privilégier les formes ci-dessus dans le but de faciliter l'obtention de l'état initial si la sortie est connue. En effet, la propriété d'observabilité des formes d'état devrait faciliter l'obtention de l'entrée en régime permanent à partir de sa sortie, lors du calcul d'initialisation à rebours qui est l'un des objectifs de ce mémoire.

2.1.2 Passage au temps discret

Sachant une entrée

$$U = \begin{bmatrix} u(t_0) & u(t_0 + \Delta t) & u(t_0 + 2\Delta t) & \cdots & u(t_f) \end{bmatrix} \tag{2.52}$$

et une fonction de transfert

$$H(s) = \frac{Y(s)}{U(s)} = \frac{b_0 s^n + b_1 s^{n-1} + \cdots + b_{n-1} s + b_n}{s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n} \tag{2.53}$$

on veut obtenir la sortie

$$Y = \begin{bmatrix} y(t_0) & y(t_0 + \Delta t) & y(t_0 + 2\Delta t) & \cdots & y(t_f) \end{bmatrix} \tag{2.54}$$

en passant par l'une des formes d'états présentées précédemment.

Par discrétisation et inversion

On veut discrétiser la forme d'état en temps continu :

$$\begin{cases} \dot{\mathbf{X}}(t) = \mathbf{A}\mathbf{X}(t) + \mathbf{B}u(t) \\ y(t) = \mathbf{C}\mathbf{X}(t) + Du \end{cases} \tag{2.55}$$

Pour cela on utilise la discrétisation trapézoïdale :

$$X(t) \to \frac{1}{2}(X_t + X_{t-\Delta t}), \quad \dot{X}(t) \to \frac{1}{\Delta t}(X_t - X_{t-\Delta t}), \quad U(t) \to \frac{1}{2}(U_t + U_{t-\Delta t}) \tag{2.56}$$

Soit,

$$\begin{cases} \frac{1}{\Delta t}\mathbf{X}_t - \frac{1}{\Delta t}\mathbf{X}_{t-1} = \frac{1}{2}\mathbf{A}X_t + \frac{1}{2}\mathbf{A}\mathbf{X}_{t-\Delta t} + \frac{1}{2}\mathbf{B}(U_t + U_{t-\Delta t}) \\ Y_t = \mathbf{C}\mathbf{X}_t + DU_t \end{cases} \tag{2.57}$$

En regroupant les termes et en nommant $\mathbf{I}_n$ la matrice identité de taille n :

$$\begin{cases} \mathbf{X}_t = (\mathbf{I}_n - \frac{\Delta t}{2}\mathbf{A})^{-1}(\mathbf{I}_n + \frac{\Delta t}{2}\mathbf{A})\mathbf{X}_{t-\Delta t} + \frac{1}{2}(\mathbf{I}_n - \frac{\Delta t}{2}\mathbf{A})^{-1}\mathbf{B}(U_t + U_{t-\Delta t}) \\ Y_t = \mathbf{C}\mathbf{X}_t + DU_t \end{cases} \tag{2.58}$$

On pose :

$$\mathbf{E} = (\mathbf{I}_n - \frac{\Delta t}{2}\mathbf{A})^{-1}(\mathbf{I}_n + \frac{\Delta t}{2}\mathbf{A}) \tag{2.59}$$

$$\mathbf{F} = \frac{1}{2}(\mathbf{I}_n - \frac{\Delta t}{2}\mathbf{A})^{-1}\mathbf{B} \tag{2.60}$$

La réponse de la fonction de transfert initiale est donc calculée à chaque pas de temps en effectuant :

$$\begin{cases} \mathbf{X}_t = \mathbf{E}\mathbf{X}_{t-\Delta t} + \mathbf{F}(U_t + U_{t-\Delta t}) \\ \mathbf{Y}_t = \mathbf{C}\mathbf{X}_t + DU_t \end{cases} \tag{2.61}$$

Pour vérifier l'inversibilité de $\mathbf{I}_n - \mathbf{A}$, on peut vérifier que

$$det(\mathbf{I}_n - \frac{\Delta t}{2}\mathbf{A}) = 1 + \sum_{i=0}^{n} a_i \left(\frac{\Delta t}{2}\right)^i \text{ est différent de } 0 \tag{2.62}$$

Par transformation bilinéaire

Les fonctions de transfert vues jusqu'ici appartiennent au domaine de Laplace. Elles correspondent à la traduction dans le domaine fréquentielle d'équation différentielle en temps continu. Cependant il existe aussi des fonctions de transferts représentant des équations discrètes. On parle alors du domaine en Z et ces fonctions de transfert sont notées :

$$H(z) = \frac{\beta_n z^{-n} + \cdots + \beta_1 z^{-1} + \beta_0}{\alpha_n z^{-n} + \cdots + \alpha_1 z^{-1} + \alpha_0} \tag{2.63}$$

avec z la variable du domaine en Z, équivalent de s pour le domaine de Laplace. n est l'ordre de la fonction de transfert, et les α_i et β_i les coefficients du dénominateur et du numérateur. Cela correspond à l'équation en temps discret à l'instant $p\Delta t$:

$$\alpha_n y_{p-n} + \cdots + \alpha_1 y_{p-1} + \alpha_0 y_p = \beta_n u_{p-n} + \cdots + \beta_1 u_{p-1} + \beta_0 u_p \tag{2.64}$$

Afin de passer d'une fonction de transfert dans le domaine de Laplace à son équivalent dans le domaine en Z, on utilise la transformation bilinéaire ([**?**]).

Ici, nous allons tout d'abord appliquer la transformation bilinéaire à notre fonction e transfert $H(s)$, puis multiplier le numérateur et le dénominateur par z^n, et enfin transformer la fonction de transfert discrétisée $H(z)$ obtenue en une forme d'état. En effet, des formes d'état

discrètes similaires à celle présentées précédemment peuvent être appliqués au fonctions de transfert dans le domaine en Z en définissant des états discrets de la même manière qu'ont été définis les états intermédiaires en temps continu. Multiplier par z^n permet de travailler avec des états discrets de la forme $x_{t+1} = g * x_t + h$ (où g et h sot des constantes quelconques) au lieu d'un état de la forme $x_{t-1} = g * x_t + h$.

$$H(z) = \frac{\beta_n z^{-n} + \cdots + \beta_1 z^{-1} + \beta_0}{\alpha_n z^{-n} + \cdots + \alpha_1 z^{-1} + \alpha_0} \rightarrow H(z) = \frac{\beta_n + \cdots + \beta_1 z^{n-1} + \beta_0 z^n}{\alpha_n + \cdots + \alpha_1 z^{n-1} + \alpha_0 z^n} \tag{2.65}$$

$$\rightarrow \begin{cases} \mathbf{X}_{p+1} = \mathbf{A}\mathbf{X}_p + \mathbf{B}u_p \\ y_p = \mathbf{C}\mathbf{X}_p + Du_p \end{cases} \tag{2.66}$$

2.1.3 Tests effectués et observations

Le but de cette série de tests est de déterminer quelle forme d'état et quel passage au temps discret (discrétisation de la forme d'état ou transformation bilinéaire) choisir afin d'avoir la meilleure stabilité numérique.

Entrées testées :

- $U(t) = A\cos(\omega t)$ (2.67)
- $U(t) = \begin{cases} 0 \text{ si } t \leq 10\Delta t \\ 1 \text{ sinon} \end{cases}$ (2.68)

Fonctions de transfert testées :

Tirées de la littérature [**?**] :

- $H(s) = \dfrac{0.33s + 50}{s + 0.005}$ (2.69)
- $H(s) = \dfrac{s + 100}{s + 10000}$ (2.70)
- $H(s) = \dfrac{56850}{s^2 + 30.16s + 56850}$ (2.71)

Autres :

$$\bullet\ H(s) = \frac{1}{s^2 + 10^3 s + 10^9} \tag{2.72}$$

$$\bullet\ H(s) = \frac{1}{s^4 + 10^3 s^3 + 10^9 s^2 + 10^5 s + 10^{11}} \tag{2.73}$$

$$\bullet\ H(s) = \frac{b}{\sum_{k=0}^{n} \binom{n}{k} s^k} \tag{2.74}$$

Initialisation :

Afin de pouvoir exécuter le premier pas de calcul l'état intermédiaire X doit être initialisé ainsi que l'entrée fictive précédant le premier pas de calcul : $u_{t_1 - \Delta t}$. Le but de cette partie étant de tester la stabilité des méthodes proposées, l'initialisation n'a pas d'importance. Ces deux variables seront donc initialisées à zéro.

Résultats :

Tous les tests ont été faits à partir du fichier `FTvEtat_Etude_Stabilite.m`.

Pour de faibles degrés de dérivations, les écarts entre les solutions des différentes méthodes sont négligeables : de l'ordre de 10^{-16} à 10^{-13}, voire nuls.

Pour des ordres de dérivations plus élevés, fonction de transfert 2.74, on commence à observer des différences notables. La figure 2.1 présente la réponse à un échelon de la fonction 2.74 à un ordre 5 pour un pas de temps $\Delta t = 1\ ms$. La méthode via transformation bilinéaire donne un résultat bien différent que celui obtenue par discrétisation de la forme d'état, qui est le résultat attendu. En diminuant le pas de temps à 0.1 ms, figure 2.2, le résultat de la méthode utilisant la transformation bilinéaire diverge.

Pour la méthode de calcul via inversion, figure 2.3, il faut atteindre des ordres de dérivations exagérément élevés pour avoir une différence. Avec l'entrée échelon et $n = 80$, on a une différence d'environ 2% entre les sorties des différentes formes d'état. Pour $n = 90$, la troisième forme d'état présentée (équation 2.51) conduit à une sortie divergente. Pour $n = 100$, on obtient les résultats de la figure 2.3, ou la première (équation 2.25) et la troisième formes d'état présentées précédemment donnent des résultats divergents. Une possible explication est que la 2e forme d'état soit moins propice, par sa forme, à des erreurs numériques.

Remarque sur les résultats : L'expression de la dérivée via la discrétisation trapézoïdale est une approximation qui conduit donc à une (très faible) erreur. Dans le cas de la discrétisation de la forme d'état, il n'y a qu'une dérivée par état intermédiaire donc l'approximation

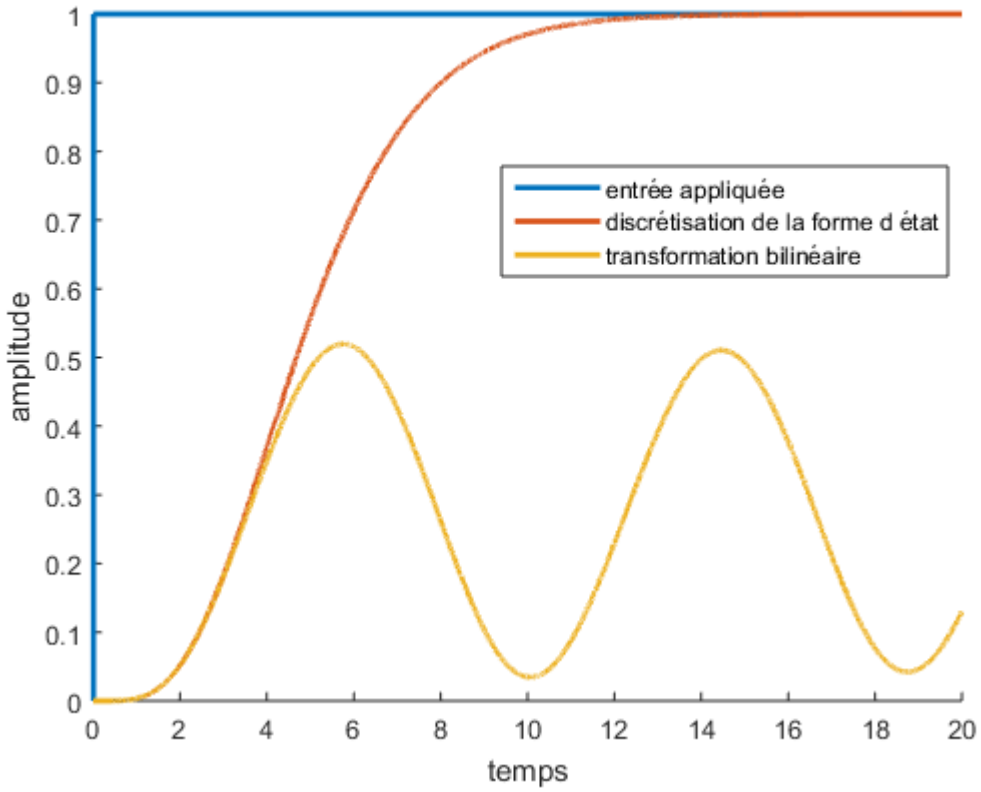

Figure 2.1 Réponse à un échelon de la fonction $H(s) = \frac{b}{\sum_{k=0}^{5} \binom{5}{k} s^k}$ avec la méthode de la discrétisation de la forme d'état et la méthode de la transformation bilinéaire, $\Delta t = 1\ ms$

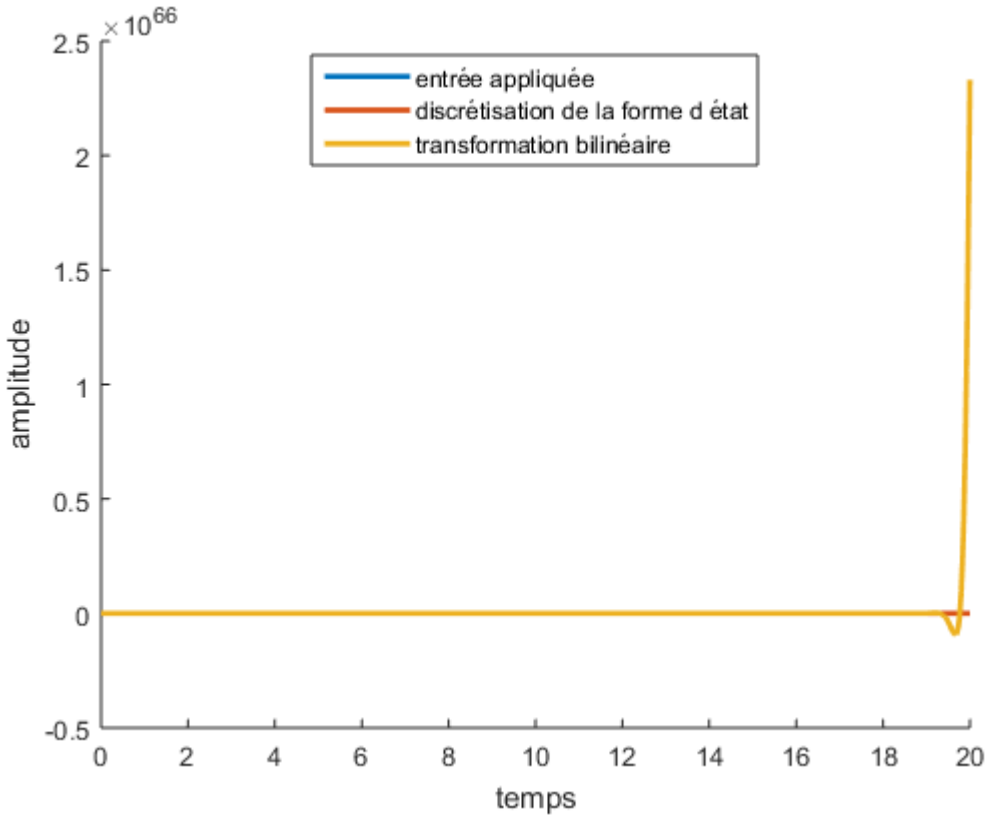

Figure 2.2 Réponse à un échelon de la fonction $H(s) = \frac{b}{\sum_{k=0}^{5} \binom{5}{k} s^k}$ avec la méthode de la discrétisation de la forme d'état et la méthode de la transformation bilinéaire, $\Delta t = 0.1\ ms$

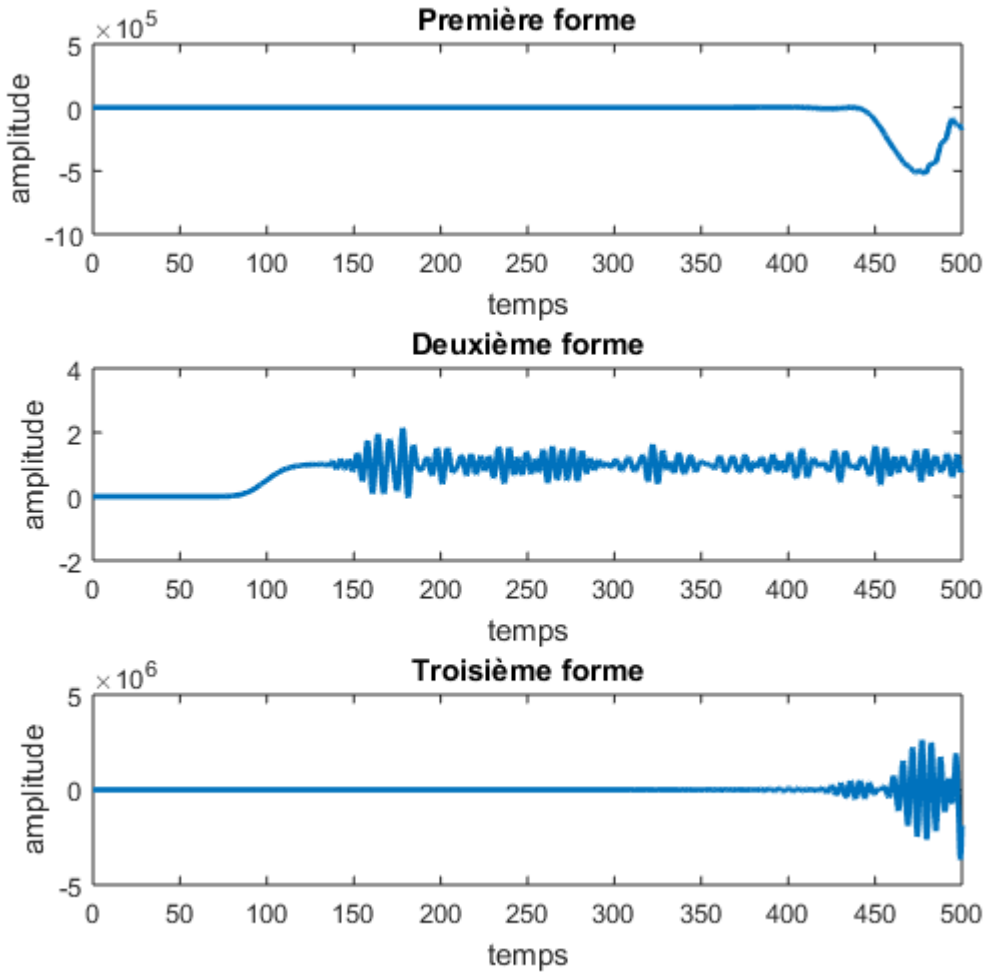

Figure 2.3 Réponse à un échelon de la fonction $H(s) = \frac{b}{\sum_{k=0}^{n} \binom{n}{k} s^k}$ 2.74 avec $n = 100$ avec la méthode de la discrétisation de la forme d'état, pour les trois forme d'état présentée, $\Delta t = 10\ ms$

est faite n fois avec n l'ordre maximal de la fonction de transfert initiale. Cependant dans le cas de la transformation bilinéaire, pour chaque coefficient d'ordre p l'approximation est faite p fois afin de transformer la dérivée d'ordre p (s^p) en une somme d'éléments. En effet une première approximation est faite pour exprimer la dérivée d'ordre p en fonction de la dérivée d'ordre $p-1$, puis on recommence pour exprimer la dérivée d'ordre $p-1$ en fonction de la dérivée d'ordre $p-2$. Et ainsi de suite jusqu'à obtenir uniquement des termes utilisant la fonction non-dérivée. Pour une fonction de transfert d'ordre n, on peut donc avoir jusqu'à $n \cdot (n+1)$ approximations puisqu'il faut le faire non seulement pour le terme d'ordre n mais aussi pour tous les termes d'ordre inférieur. Il est donc logique que passer à la forme d'état avant de discrétiser soit la méthode la plus stable.

Remarque sur la discrétisation de $Y_t = CX_t + DU_t$: Les tests effectués pour de nombreux ordres de dérivations et des pas de temps différents donne une erreur relative maximale de 10^{-11} (obtenue pour un pas de 10^{-6}) entre les deux possibilités de calcul de la sortie (avec Δ_t le pas de temps de la simulation) :

$$Y_t = \mathbf{C}\mathbf{X}_t + DU_t \tag{2.75}$$

$$\text{ou} \quad Y_t = \mathbf{C}(\mathbf{X}_t + \mathbf{X}_{t-\Delta t}) + D(U_t + U_{t-\Delta t}) - Y_{t-\Delta t} \tag{2.76}$$

La première forme de calcul de Y_t est conservée dans la suite.

Conclusion

Il est clair que le meilleur passage au temps discret est la méthode via discrétisation et inversion. En ce qui concerne la forme d'état, la deuxième transformation proposée 2.39 permet une convergence jusqu'à des ordres de dérivation plus élevés que les deux autres. Cependant pour les fonctions de transfert issus de cas réels, l'ordre de dérivation maximal est bien trop faible pour avoir un écart notable entre les différentes formes d'état testées.

2.2 Démonstration de la méthode sur un exemple

La fonction de transfert suivante va être étudiée et simulée avec Matlab, EMTP et la méthode décrite précédemment, afin de montrer l'avantage de cette dernière.

$$H(s) = \frac{80s^2}{s^3 + 82s^2 + 161s + 80} \tag{2.77}$$

2.2.1 Réponse analytique à un échelon

$$Y(s) = H(s) \cdot \frac{1}{s} = \frac{80s}{s^3 + 82s^2 + 161s + 80} \tag{2.78}$$

Factorisation du dénominateur :

$$Y(s) = \frac{80s}{(s+1)^2(s+80)} \tag{2.79}$$

Décomposition en élément simple :

$$Y(s) = \frac{80}{6241}\left(\frac{80s+1}{(s+1)^2} - \frac{80}{s+80}\right) \tag{2.80}$$

Ce qui peut s'écrire :

$$Y(s) = \frac{80}{6241}\left[80\left(\frac{1}{s+1} - \frac{1}{(s+1)^2}\right) + \frac{1}{(s+1)^2} - \frac{80}{s+80}\right] \tag{2.81}$$

Passage dans le domaine du temps :

$$y(t) = \frac{80}{6241}\left[80\left(\exp^{-t} - t\exp - t\right) + t\exp^{-t} - 80\exp^{-80t}\right] \tag{2.82}$$

$$\tag{2.83}$$

La réponse analytique a un échelon de la fonction de transfert H (2.77) dans le domaine du temps est :

$$y(t) = \frac{80}{6241}\left[(89 - 79t)\exp^{-t} - 80\exp^{-80t}\right] \tag{2.84}$$

2.2.2 Comparaison entre la discrétisation avec Matlab, EMTP et la méthode proposée

D'une part, les fonctions Matlab `c2d` avec la méthode `'tustin'` (appelée Matlab tustin dans les tableaux de résultats) et `bilin` avec la méthode `'BwdRec'` (appelée Matlab BwdRec) ont été utilisées pour des pas de temps de $50\mu s$ et $1\mu s$. D'autre part, la même réponse a été simulée avec EMTP avec l'option de simulation `trapezoidal` puis avec `Backward Euler`. La simulation pour cette dernière méthode diverge pour les deux pas de temps. Enfin la réponse

a été simulée avec la méthode de simulation des fonctions de transfert proposée suite à la première partie (passage à la forme d'état 2.39 + discrétisation trapézoïdale). Les erreurs par rapport à la réponse analytique obtenues sont répertoriées dans le tableau 2.1. Le résultat des simulations est présenté sur les figures 2.4 et 2.5. Pour un pas de temps de $1\mu s$ les simulations avec EMTP et Matlab ne sont pas fidèles à la réponse analytique. Au contraire de la méthode choisie dans le section précédente de ce mémoire (passage à la forme d'état 2.39 + discrétisation trapézoïdale) qui donne toujours un résultat proche de la théorie. Les simulations sur EMTP ont été faites sur le fichier `Ex_FTS.ecf` et les simulations MatLab sur le fichier `Ex_FTS.m`.

2.2.3 Fonction de transfert décomposée

Comme on a pu le voir lors du calcul de la réponse analytique, on peut aussi écrire la fonction de transfert sous la forme :

$$H(s) = \frac{80s^2}{(s+80)(s+1)^2} = \frac{80}{s+80}\frac{s}{s+1}\frac{s}{s+1} \tag{2.85}$$

On effectue la simulation avec EMTP avec 3 blocs "fonction de transfert". La simulation avec la méthode trapézoïdale ne présente plus d'erreur importante pour $dt = 1\mu s$ et la simulation avec la méthode d'intégration numérique *backward euler* converge. Les simulations équivalentes sont réalisées avec Matlab. Les erreurs avec la réponse analytique sont présentées dans le tableau 2.2, les différences avec les résultats précédents se trouvent dans le tableau 2.3. Les courbes obtenues sont celles de la figure 2.6. On n'observe plus de différences notables entre les différentes méthodes.

Tableau 2.1 Erreurs entre la réponse analytique à un échelon de 2.77 et la réponse simulée avec les différentes méthodes

	Erreur absolue maximale	
	dt = $50\mu s$	dt = $1\mu s$
Matlab tustin	$1.995908 \cdot 10^{-3}$	$4.455995 \cdot 10^{-2}$
Matlab BwdRec	$3.983665 \cdot 10^{-3}$	$7.999344 \cdot 10^{-5}$
EMTP trapezoidal	$2.0 \cdot 10^{-2}$	$1.7 \cdot 10^{-1}$
Méthode proposée	$2.0 \cdot 10^{-2}$	$3.9998 \cdot 10^{-5}$

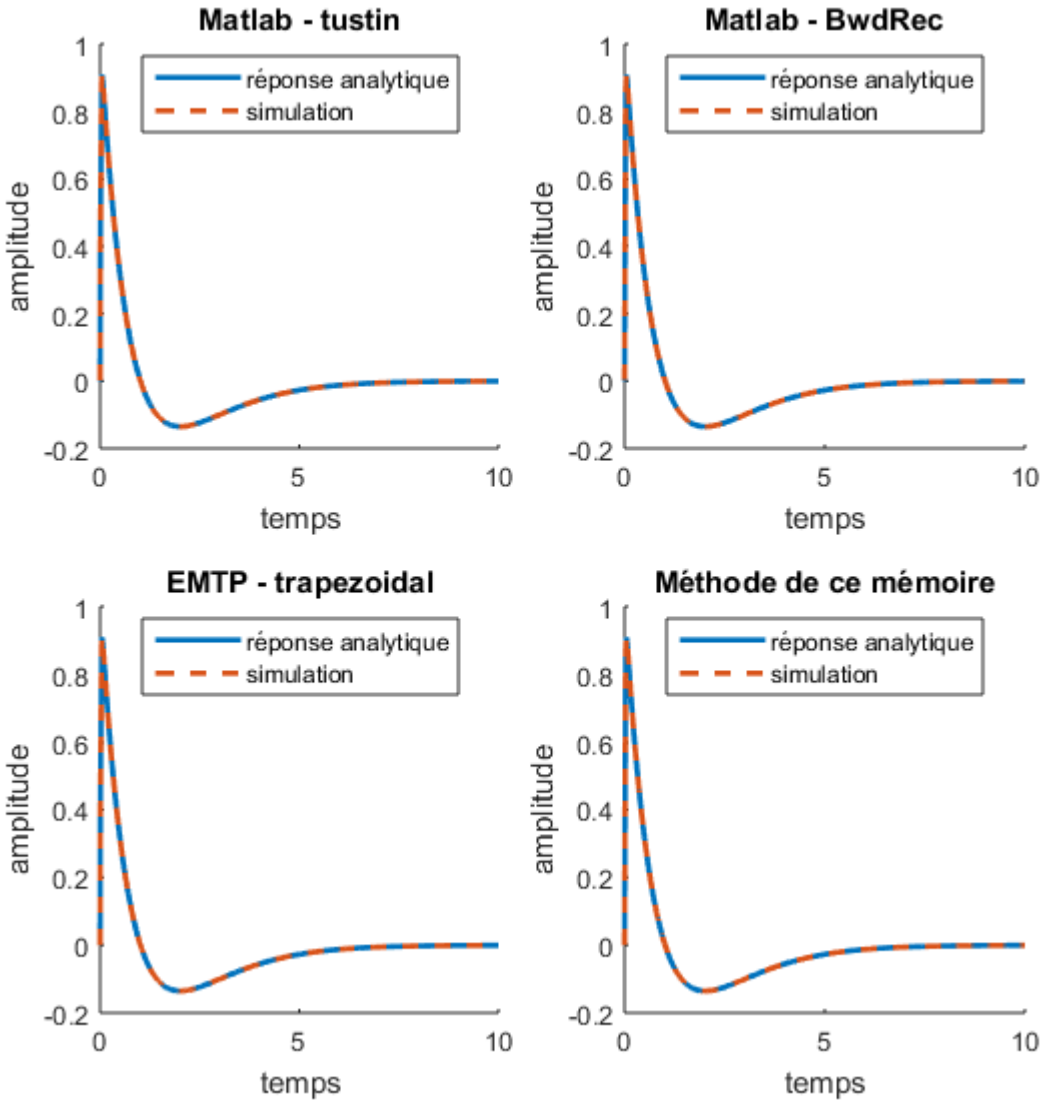

Figure 2.4 Simulation de 2.77 avec les différentes méthodes pour un pas de temps de $50\mu s$

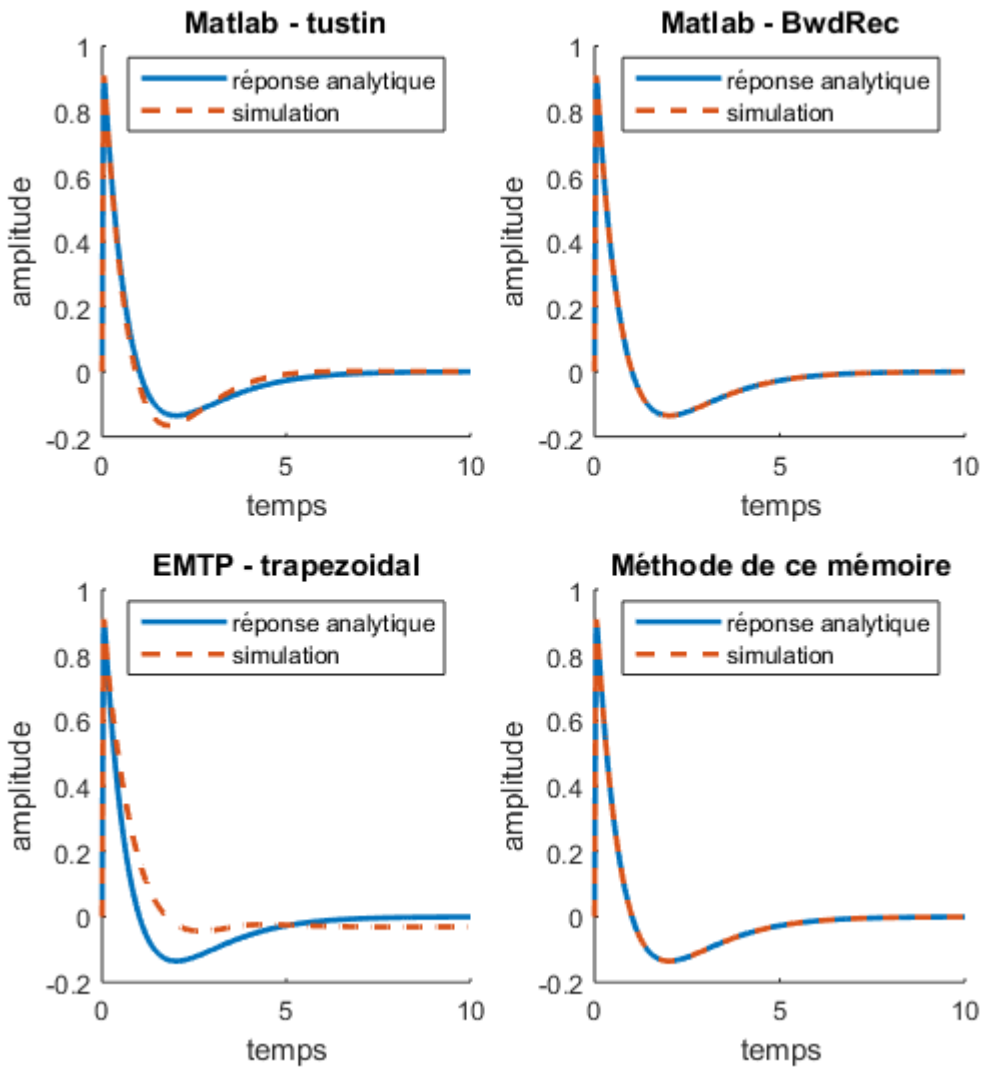

Figure 2.5 Simulation de 2.77 avec les différentes méthodes pour un pas de temps de $1\mu s$

Tableau 2.2 Erreurs entre la réponse analytique de 2.85 et la réponse simulée avec les différentes méthodes

	Erreur absolue maximale	
	dt $= 50\mu s$	dt $= 1\mu s$
Matlab tustin	$4.54874 \cdot 10^{-5}$	$9.091219 \cdot 10^{-7}$
Matlab BwdRec	$3.983665 \cdot 10^{-3}$	$7.999344 \cdot 10^{-5}$
EMTP trapezoidal	$2.0 \cdot 10^{-2}$ *	$3.9998 \cdot 10^{-5}$
EMTP backward euler	$2.0 \cdot 10^{-2}$ *	$3.9998 \cdot 10^{-5}$
ma méthode	0.0020 *	$3.9998 \cdot 10^{-5}$

** : si on coupe le début de la simulation (première demi-seconde) on obtient une erreur de l'ordre de* 10^{-5}

Tableau 2.3 Différences entre les réponses simulées de 2.77 et de 2.85 avec les différentes méthodes

	Différence absolue maximale	
	dt $= 50\mu s$	dt $= 1\mu s$
Matlab tustin	$1.995908 \cdot 10^{-3}$	$4.455989 \cdot 10^{-2}$
Matlab BwdRec	$8.985312 \cdot 10^{-13}$	$4.547582 \cdot 10^{-11}$
EMTP trapezoidal	$3.8914 \cdot 10^{-6}$	$1.7 \cdot 10^{-1}$

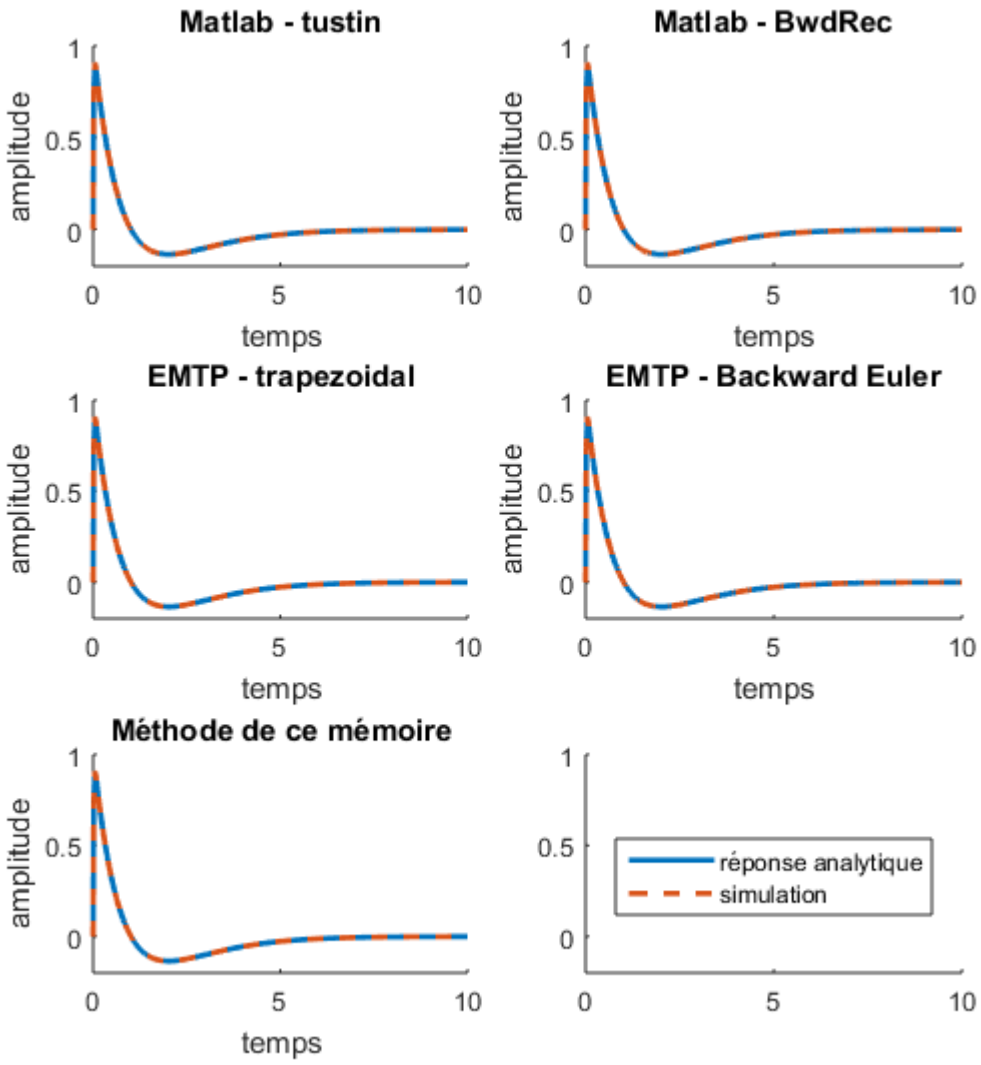

Figure 2.6 Simulation de l'équation découplée 2.85 avec les différentes méthodes pour le pas de temps $dt = 1\mu s$

Conclusion de l'exemple

La méthode choisie en première partie s'est montrée plus performante que la méthode implémentée actuellement sur EMTP pour simuler la fonction de transfert donnée. Une autre solution est de décomposer la fonction de transfert problématique en un produit de fonction plus simple. Cependant, le logiciel EMTP doit pouvoir simuler de façon performante des fonctions de transfert arbitraires rentrées par l'utilisateur qui ne seront pas forcément décomposables.

2.3 Généralisation aux fonctions de transfert matricielle et en temps discret

Dans cette partie la méthode de simulation des fonctions de transfert scalaires dans le domaine de Laplace va être généralisée dans un premier temps aux systèmes multi-entrées multi-sorties puis au domaine en Z (temps discrétisé). L'initialisation de la simulation en régime permanent sera aussi étudiée.

2.3.1 Passage à la forme d'état d'une fonction de transfert matricielle

Le but est d'obtenir une forme d'état à partir de la fonction de transfert matricielle $H(s)$ de taille $r \cdot m$, tel que $Y(s) = H(s) \cdot U(s)$ (m entrées, r sorties). Chaque coefficient de H est une fonction de transfert scalaire.

$$[\mathbf{H}(s)]_{ij} = h_{ij}(s) = \frac{y_{ij}(s)}{u_j(s)} \tag{2.86}$$

$$y_i(s) = \sum_{j=1}^{m} y_{ij}(s) \tag{2.87}$$

Exemple : Soit la fonction de transfert à 2 entrées et 3 sorties suivante :

$$H(s) = \begin{bmatrix} \frac{1}{1+s+s^2} & \frac{2}{1+2s+s^2} \\ \frac{3+s}{1+s+s^2} & \frac{4+s}{2+2s+s^2} \\ \frac{5+s+s^2}{3+s+s^2} & \frac{6+s+s^2}{3+2s+s^2} \end{bmatrix}$$

Nommons l'entrée $\mathbf{U} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$ et la sortie $\mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$. Si on se concentre sur y_1, on a :

$$y_1(s) = y_{1,1}(s) + y_{1,2}(s) = \frac{1}{1+s+s^2}u_1(s) + \frac{2}{1+2s+s^2}u_2(s)$$

où $y_{1,1}$ représente la dépendance de y_1 en u_1 et y_2 la dépendance en u_2. On va tout d'abord étudier $y_{1,1} = h_{1,1}(s)u_1(s)$ et $y_{1,2} = h_{1,2}(s)u_2(s)$ séparément pour les transformer en forme d'état en utilisant la forme d'état scalaire (2.39) choisie en première partie. Et ensuite on va reconstruire la sortie $y_1(t) = y_{1,1}(t) + y_{1,2}(t)$ avec les sorties des formes d'état obtenues.

Pour toute la fonction de transfert MIMO, une forme d'état est associée à chaque coefficient en utilisant la forme choisie dans la première partie 2.39.

$$(1,1) \Rightarrow \begin{cases} \dot{X}_{11} = \begin{bmatrix} 0 & -1 \\ 1 & -1 \end{bmatrix} X_{11} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} U_1 \\ Y_{11} = \begin{bmatrix} 0 & 1 \end{bmatrix} X_{11} \end{cases} \qquad (1,2) \Rightarrow \begin{cases} \dot{X}_{12} = \begin{bmatrix} 0 & -1 \\ 1 & -2 \end{bmatrix} X_{12} + \begin{bmatrix} 2 \\ 0 \end{bmatrix} U_2 \\ Y_{12} = \begin{bmatrix} 0 & 1 \end{bmatrix} X_{12} \end{cases}$$

(2.88)

$$(2,1) \Rightarrow \begin{cases} \dot{X}_{21} = \begin{bmatrix} 0 & -2 \\ 1 & -1 \end{bmatrix} X_{21} + \begin{bmatrix} 3 \\ 1 \end{bmatrix} U_1 \\ Y_{21} = \begin{bmatrix} 0 & 1 \end{bmatrix} X_{21} \end{cases} \qquad (2,2) \Rightarrow \begin{cases} \dot{X}_{22} = \begin{bmatrix} 0 & -2 \\ 1 & -2 \end{bmatrix} X_{22} + \begin{bmatrix} 4 \\ 1 \end{bmatrix} U_2 \\ Y_{12} = \begin{bmatrix} 0 & 1 \end{bmatrix} X_{22} \end{cases}$$

(2.89)

$$(3,1) \Rightarrow \begin{cases} \dot{X}_{31} = \begin{bmatrix} 0 & -3 \\ 1 & -1 \end{bmatrix} X_{31} + \begin{bmatrix} 5 \\ 1 \end{bmatrix} U_1 \\ Y_{31} = \begin{bmatrix} 0 & 1 \end{bmatrix} X_{31} + U_1 \end{cases} \qquad (3,2) \Rightarrow \begin{cases} \dot{X}_{32} = \begin{bmatrix} 0 & -3 \\ 1 & -2 \end{bmatrix} X_{32} + \begin{bmatrix} 6 \\ 1 \end{bmatrix} U_2 \\ Y_{32} = \begin{bmatrix} 0 & 1 \end{bmatrix} X_{32} + U_2 \end{cases}$$

(2.90)

où, par exemple, la forme d'état (2,1) représente l'effet de l'entrée u_1 sur la sortie y_2. Les coefficients de la sortie $\mathbf{Y}$ sont alors la somme de l'effet de chaque entrée :

$$Y_1 = Y_{11} + Y_{21}, \quad Y_2 = Y_{21} + Y_{22}, \quad Y_3 = Y_{31} + Y_{32} \tag{2.91}$$

On en déduit le système complet :

$$\begin{bmatrix} \dot{X}_{11,1} \\ X_{11,2} \\ \vdots \\ X_{32,2} \end{bmatrix} = \begin{bmatrix} 0 & -1 & & & & & & & & & & \\ 1 & -1 & & & & & & & & & & \\ & & 0 & -1 & & & & & & & & \\ & & 1 & -2 & & & & & & & & \\ & & & & 0 & -2 & & & & & & \\ & & & & 1 & -1 & & & & & & \\ & & & & & & 0 & -2 & & & & \\ & & & & & & 1 & -2 & & & & \\ & & & & & & & & 0 & -3 & & \\ & & & & & & & & 1 & -1 & & \\ & & & & & & & & & & 0 & -3 \\ & & & & & & & & & & 1 & -2 \end{bmatrix} \begin{bmatrix} X_{11,1} \\ X_{11,2} \\ \vdots \\ X_{32,2} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 2 \\ 0 & 0 \\ 3 & 0 \\ 1 & 0 \\ 0 & 4 \\ 0 & 1 \\ 5 & 0 \\ 1 & 0 \\ 0 & 6 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \tag{2.92}$$

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 & & & & & & & & \\ & & & & 0 & 1 & 0 & 1 & & & & \\ & & & & & & & & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{11,1} \\ \vdots \\ X_{32,2} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \tag{2.93}$$

Cas général : Pour chaque coefficients $h_{ij}(s) = \frac{b_{0,ij}s^{n_{ij}}+\cdots+b_{n,ij}}{s^{n_{ij}}+\cdots+a_{n,ij}}$, d'ordre de dérivation maximal n_{ij}, on définit une forme d'état scalaire suivant une la méthode définie précédemment :

$$\begin{cases} \dot{\mathbf{X}}_{ij} = \mathbf{A}_{ij}\mathbf{X}_{ij} + \mathbf{B}_{ij}U_j \\ Y_{ij} = \mathbf{C}_{ij}\mathbf{X}_{ij} + D_{ij}U_{ij} \end{cases} \qquad \mathbf{X}_{ij} \in \mathbb{R}^{n_{ij}}, \mathbf{A}_{ij} \in \mathbb{R}^{n_{ij}\cdot n_{ij}}, \mathbf{B}_{ij} \in \mathbb{R}^{n_{ij}}, \mathbf{C}_{ij} \in \mathbb{R}^{1\cdot n_{ij}}, D_{ij} \in \mathbb{R} \tag{2.94}$$

La forme d'état s'obtient en construisant le système :

$$\begin{bmatrix} \dot{\mathbf{X}}_{11} \\ \dot{\mathbf{X}}_{12} \\ \vdots \\ \dot{\mathbf{X}}_{21} \\ \vdots \\ \dot{\mathbf{X}}_{rm} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11} & & & & & \\ & \mathbf{A}_{21} & & & & \\ & & \ddots & & & \\ & & & \mathbf{A}_{21} & & \\ & & & & \ddots & \\ & & & & & \mathbf{A}_{rm} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{11} \\ \mathbf{X}_{12} \\ \vdots \\ \mathbf{X}_{21} \\ \vdots \\ \mathbf{X}_{rm} \end{bmatrix} + \begin{bmatrix} \mathbf{B}_{11} & \\ & \ddots \\ \mathbf{B}_{21} & \\ & \ddots \\ \ddots & \\ & \mathbf{B}_{rm} \end{bmatrix} \mathbf{U} \tag{2.95}$$

$$\mathbf{Y} = \begin{bmatrix} \mathbf{C}_{11} & \cdots & \mathbf{C}_{1m} & & & & & & \\ & & & \mathbf{C}_{21} & \cdots & \mathbf{C}_{2m} & & & \\ & & & & & & \ddots & & \\ & & & & & & & \mathbf{C}_{r1} & \cdots & \mathbf{C}_{rm} \end{bmatrix} \begin{bmatrix} \mathbf{X}_{11} \\ \vdots \\ \mathbf{X}_{21} \\ \vdots \\ \mathbf{X}_{rm} \end{bmatrix} + \begin{bmatrix} D_{11} & \cdots & D_{1m} \\ \vdots & & \vdots \\ D_{r1} & \cdots & D_{rm} \end{bmatrix} \mathbf{U} \tag{2.96}$$

On revient au système :

$$\begin{cases} \dot{\mathbf{X}} = \mathbf{A}\mathbf{X} + \mathbf{B}\mathbf{U} & \mathbf{X} \in \mathbb{R}^n, \mathbf{U} \in \mathbb{R}^m, \mathbf{Y} \in \mathbb{R}^r \\ \mathbf{Y} = \mathbf{C}\mathbf{X} + \mathbf{D}\mathbf{U} & \mathbf{A} \in R^{n \cdot n}, \mathbf{B} \in \mathbb{R}^{n \cdot m}, \mathbf{C} \in \mathbb{R}^{r \cdot n}, \mathbf{D} \in \mathbb{R}^{r \cdot m} \end{cases} \qquad n = \sum_{i=1}^{r} \sum_{j=1}^{m} n_{ij} \tag{2.97}$$

Cette forme d'état multi-entrée multi-sortie est une possibilité parmi plusieurs. En effet selon la façon de procéder à partir de la fonction de transfert originelle on peut obtenir plusieurs forme d'état différente qui auront le même comportement entrée-sortie. Cette forme a été choisie pour sa facilité d'implémentation.

Forme minimale

La forme d'état obtenue à partir d'une fonction de transfert matricielle peut atteindre une taille très importante. Il existe une méthode permettant de la réduire, afin d'obtenir un système ayant le même comportement entrée-sortie que le système initial mais avec la plus petite taille possible. Cet algorithme est détaillé dans l'article [**?**]. Il s'agit d'une méthode construisant une nouvelle forme d'état à partir des vecteurs propres de la matrice $A - BKC$ et de sa transposée, où A, B et C sont les matrices de notre forme d'état initiale et K une matrice aléatoire respectant $||BKC|| \simeq ||A||$. Les vecteur propres correspondant à une valeur propres qui est aussi une valeur propres de A sont éliminés. Soit X les vecteur propres restant liés à la matrice $A-BKC$, Y ceux liés à sa transposée $(A-BKC)^T$ et Δ la matrice diagonale des vecteur propre de $A - BKC$ qui ne sont pas vecteur propres de A. On calcule la forme

d'état minimale selon :

$$C_{minimal} = C * X \tag{2.98}$$
$$B_{minimal} = Y' * B \tag{2.99}$$
$$A_{minimal} = \Delta - B_{minimal} * K * C_{minimal} \tag{2.100}$$

Notons que cette nouvelle forme d'état est observable et contrôlable. Cependant elle peut avoir des coefficient complexe dû à l'utilisation des vecteurs propres (qui peuvent être complexe). L'article propose aussi une méthode garantissant un résultat réel en se basant sur les parties réelles et imaginaire des vecteurs propres.

2.3.2 Initialisation en régime permanent d'une fonction de transfert du domaine de Laplace

Dans la plupart des simulations, le réseau électrique est considéré en régime permanent à l'instant initial. Les tensions et intensités en tout point de la partie électrique du réseau sont calculées grâce à un calcul d'écoulement de puissance. Du point de vue du système de contrôle, cela donne une ou plusieurs variables en régime permanent. Il s'agit le plus souvent des variables de sortie, il faut donc propager l'information à rebours pour initialiser tout le système.

Cas d'une fonction de transfert avec une seule entrée et sortie

Soit la fonction de transfert quelconque :

$$H(s) = \frac{Y(s)}{U(s)} = \frac{b_0 s^n + b_1 s^{n-1} + \cdots + b_{n-1} s + b_n}{s^n + a_1 s^{n-1} + \cdots + a_{n-1} s + a_n} \tag{2.101}$$

On note x_{RP} la valeur de la variable x en régime permanent.

En régime permanent $t \to \inf \Leftrightarrow s \to 0$. Ainsi, sachant la sortie en régime permanent, la fonction de transfert donne la consigne :

$$u_{RP} = \frac{a_n}{b_n} y_{RP} \tag{2.102}$$

De plus, la forme d'état devient en régime permanent, de même que pour le cas scalaire on utilise un régime particulier stationnaire ($\dot{\mathbf{X}} = 0$) :

$$\begin{cases} 0 = \mathbf{A}\ \mathbf{X} + \mathbf{B}u \\ y = \mathbf{C}\mathbf{X} + Du \end{cases} \tag{2.103}$$

on obtient, si A est inversible :

$$X_{RP} = -A^{-1}Bu_{RP} = -A^{-1}B\frac{a_n}{b_n}y_{RP} \tag{2.104}$$

Dans le cas des formes d'état de la section précédente 2.39, A est inversible si et seulement si a_n est non nul. De plus on peut facilement calculer l'état avec les formules suivantes. En particulier, pour la forme retenue :

$$\begin{cases} x_i = \dfrac{a_{n-i}b_n - a_n b_{n-i}}{b_n}y, \quad 1 \leq i \leq n-1 \\ x_n = \dfrac{b_n - a_n b_0}{b_n}y \end{cases} \tag{2.105}$$

Cas d'une fonction de transfert avec des entrées et sorties multiples

De la même façon que pour le cas scalaire, la forme d'état 2.97 en régime permanent donne, si A est inversible :

$$X_0 = -A^{-1}BU_0 \tag{2.106}$$

Si la matrice A a été construite selon la méthode exposée précédemment 2.95, selon les propriétés des matrices diagonales par blocs, on a :

$$det(A) = \prod_{i,j=1}^{r,m} det(A_{i,j}) = \prod_{i,j=1}^{r,m} (-1)^{n_{ij}} a_{n_{ij}} \tag{2.107}$$

Si on se place dans le cas où on ne connais pas l'entrée U_0 à appliquer, mais la sortie Y_0, l'initialisation se fait en résolvant le système matriciel :

$$\begin{bmatrix} 0 \\ Y_0 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} X_0 \\ U_0 \end{bmatrix} \tag{2.108}$$

Cependant la résolution peut poser problème. si le nombre r d'entrées est égal du nombre m

de sorties, on peut espérer avoir une matrice inversible. Si on a plus d'entrées que de sorties ($r > m$), le système est sur-dimensionné, on peut le résoudre en utilisant une partie des équations. Mais le cas inverse ($m > r$), la résolution est impossible.

Remarque : le régime permanent supposé ici est en fait un régime stationnaire : $\dot{X} = 0$, ce qui n'est pas applicable à un régime d'oscillations forcées. Cependant dans le cas où tensions et intensités sont étudiées dans le domaine dq0, elles sont bien constantes en régime permanent donc cette estimations est valide.

2.3.3 Fonction de transfert en temps discret

Les fonctions de transfert étudiées ici correspondent à l'équivalent dans le domaine fréquentiel d'un temps discrétisé, contrairement au domaine de Laplace correspondant à un temps continu. Elles s'écrivent sous la forme :

$$H(z) = \frac{Y(z)}{U(z)} = \frac{\alpha_n z^{-n} + \alpha_{n-1} z^{-(n-1)} + \cdots + \alpha_1 z^{-1} + \alpha_0}{\beta_n z^{-n} + \beta_{n-1} z^{-(n-1)} + \cdots + \beta_1 z^{-1} + \beta_0} \tag{2.109}$$

et correspondent dans le domaine temporel à l'équation :

$$\alpha_n y_{t-n\Delta t} + \cdots + \alpha_1 y_{t-\Delta t} + \alpha_0 y_t = \beta_n u_{t-n\Delta t} + \cdots + \beta_1 u_{t-\Delta t} + \beta_0 u_t \tag{2.110}$$

Le passage du domaine de Laplace à une fonction de transfert en Z peut se faire via la transformation bilinéaire (cf [**?**]), mais ici les fonctions étudiées seront directement discrétisées. En effet comme nous l'avons vu dans la première partie dans la sous-section sur la transformation bilinéaire 2.1.2, la simulation d'une fonction de transfert du domaine de Laplace est plus performante sans passer par la transformation bilinéaire.

Simulation avec initialisation en régime permanent

Puisque la fonction de transfert correspond déjà à un temps discrétisé, il suffit de lui appliquer directement la transformation en forme d'état 2.39 choisie en première partie, comme expliqué à l'équation 2.66. La forme d'état obtenue est donc directement discrétisée :

$$\begin{cases} \mathbf{X}_{p+1} = \mathbf{A}\ \mathbf{X}_p + \mathbf{B} u_p \\ \quad y_p = \mathbf{C}\mathbf{X}_p + D u_p \end{cases} \tag{2.111}$$

Les fonctions de transfert matricielles se traitent de la même façon en prenant la forme

d'état 2.95 explicitée plus haut.

L'initialisation en régime permanent se fait en posant $X_{p+1} = X_p$, il faut alors résoudre le système suivant où I représente la matrice identité :

$$\begin{cases} 0 = (\mathbf{A} - \mathbf{I})\mathbf{X}_0 + \mathbf{B}U_0 \\ Y_0 = \mathbf{C}\mathbf{X}_0 + DU_0 \end{cases} \tag{2.112}$$

On retrouve les mêmes problèmes de dimensionnement que dans le cas du temps continu.

Changement de pas de temps

Dans le cas où la fonction de transfert fournie par l'utilisateur ne correspond pas au pas de temps pour lequel il désire la simuler, il faut procéder à un changement de pas de discrétisation. Pour cela une façon de faire est d'appliquer l'inverse de la transformation bilinéaire décrite par [**?**] pour retrouver une fonction de transfert du domaine de Laplace.

Soit $\vec{\alpha}$ le vecteur des coordonnées dans le domaine en Z et $\vec{a}$ le vecteur des coordonnées dans le domaine de Laplace.

$$\vec{\alpha} = \begin{bmatrix} \alpha_0 & \alpha_0 & \cdots & \alpha_n \end{bmatrix}^T \tag{2.113}$$

$$\text{et } \vec{a} = \begin{bmatrix} a_0 & a_1 & \cdots a_n \end{bmatrix}^T \tag{2.114}$$

$$\text{on a : } \vec{\alpha} = DH \cdot T \cdot \vec{a} \tag{2.115}$$

où DH est une matrice obtenue à partir de produits de coefficients binomiaux qui vérifie :

$$DH^{-1} = \frac{1}{2^n} \cdot DH \tag{2.116}$$

$$\text{et } T = \begin{bmatrix} \left(\frac{2}{\Delta t}\right)^0 & & \\ & \ddots & \\ & & \left(\frac{2}{\Delta t}\right)^n \end{bmatrix} \tag{2.117}$$

On peut en déduire :

$$\vec{a} = \frac{1}{2^n} \cdot \begin{bmatrix} \left(\frac{\Delta t}{2}\right)^0 & & \\ & \ddots & \\ & & \left(\frac{\Delta t}{2}\right)^n \end{bmatrix} \cdot DH \cdot \vec{\alpha} \tag{2.118}$$

Une fois la fonction de transfert dans le domaine de Laplace obtenue, on applique le passage à la forme d'état et la discrétisation pour pouvoir la simuler. On pourrait aussi appliquer la transformation bilinéaire correspondant au pas de temps voulu mais comme nous l'avons vu en première partie cette façon de faire est moins performante.

la figure 2.7 montre un exemple de diminution du pas de temps effectué sur la fonction de transfert suivante, extraire d'un contrôleur définit par [**?**], associée au pas de temps $\Delta t = 0.1$. (fichier `Ex_FTZ_changedt.m`)

$$H(z) = \frac{10^{-5} \cdot (-10.39z^{-6} - 6.98z^{-5} + 9.1z^{-4} + 0.7z^{-3} - 2.36z^{-2} + 1.95z^{-1} - 0.63)}{z^{-6} - 2.8z^{-5} + 3.26z^{-4} - 2.16z^{-3} + 0.93z^{-2} - 0.25z^{-1} + 0.03} \tag{2.119}$$

La transformation bilinéaire inverse a été appliqué à cette fonction de transfert puis elle a été transformée en la forme d'état (2.39) choisie en première partie. Enfin cette forme d'état a été discrétisée selon la discrétisation trapézoïdale pour être simulée au pas de temps voulu. Ici, pour tester différents pas de temps, la fonction de transfert dans le domaine de Laplace a été reprise et discrétisée au pas de temps voulu à chaque test.

Conclusion du chapitre

Dans ce chapitre, une méthode de simulation des fonctions de transfert du domaine de Laplace a été choisie afin d'avoir la meilleure stabilité possible. Pour être simulée, la fonction de transfert est transformée en forme d'état. Différentes formes sont possibles et leurs simulations diffèrent pour des ordres très élevés. Cependant le choix de discrétiser la forme d'état obtenue ou d'utiliser la transformation bilinéaire pour faire passer la fonction de transfert initiale au temps discret à une importance. Les tests effectués ont montré qu'il est préférable de, premièrement, construire une forme d'état à partir de la fonction de transfert du domaine de Laplace, puis de discrétiser le système obtenu en utilisant la discrétisation trapézoïdale. Afin de montrer son utilité, cette méthode a été testée sur un exemple posant problème sur EMTP et a donné des résultats prometteurs. Enfin, une façon de généraliser le processus aux fonctions de transfert matricielles a été proposée. Puis, l'initialisation des fonctions de transfert du domaine de Laplace en régime permanent a été établie, ainsi que ses limitations. Finalement, le cas des fonctions de transfert du domaine en Z a été étudié afin de pouvoir les simuler, les initialiser et traiter le changement de pas de temps.

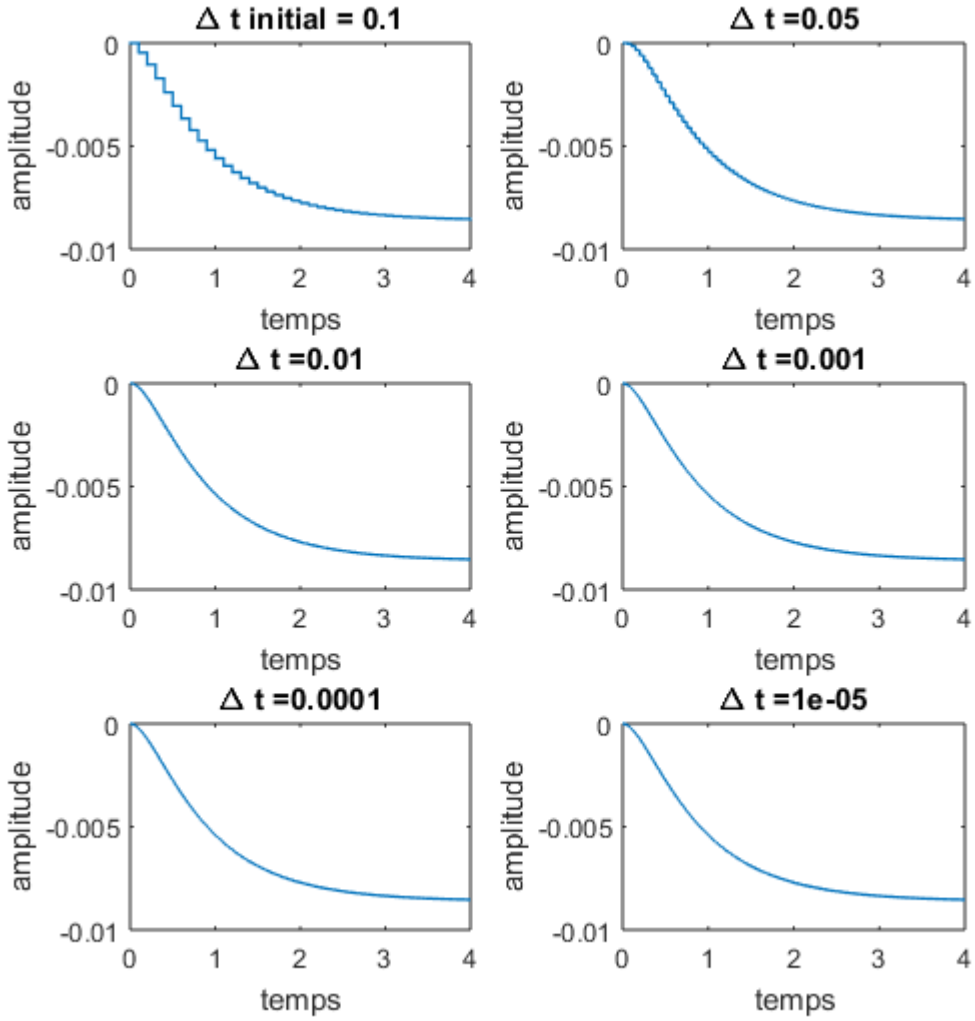

Figure 2.7 Diminution du pas de temps pour la fonction de transfert discrétisé 2.119

CHAPITRE 3 ALGORITHME D'INITIALISATION D'UN SYSTÈME DE CONTRÔLE

Le but de ce chapitre est de proposer un algorithme permettant la simulation d'un système de contrôle conforme à la méthode utilisée actuellement dans le logiciel EMTP incluant une initialisation automatique des variables en régime permanent basée sur une ou plusieurs données préalables. Dans le cadre de cette maîtrise, les systèmes traités sont linéaires, avec comme fonctions traitées : les gains, les sommes, les formes d'état, les fonctions de transfert.

Ce chapitre présentera la méthode utilisée par l'algorithme, en étudiant notamment son fonctionnement et la modélisation des différents blocs traités, pour enfin illustrer le tout avec un exemple. L'algorithme est le fichier `System_Creation.m`.

3.1 Théorie et implémentation de l'algorithme

Dans cette partie, premièrement, les choix théoriques fait pour la simulation et l'initialisation d'un système de contrôle seront explicités, ainsi que leurs répercutions sur les attentes et les fonctionnalités de l'algorithme. Ensuite le fonctionnement de celui-ci sera expliqué en précisant quelles sont les données nécessaires à lui fournir et les grandes étapes qu'il effectue. Enfin, la modélisation des différents éléments traités par l'algorithme sera définie.

3.1.1 Méthodes de simulation et d'initialisation choisies et leurs implications

Dans le logiciel EMTP les systèmes de contrôle sont simulés en résolvant à chaque pas de temps une équation matricielle de la forme :

$$\mathbf{A} \cdot \mathbf{X}_t = \mathbf{b}_t \tag{3.1}$$

où X_t est un vecteur inconnu contenant toutes les variables. Dans le cas des systèmes linéaires, la matrice $\mathbf{A}$ est constante, cependant le vecteur b_t représentant l'historique et les entrées du système doivent être réévalué à chaque pas de temps. Pour éviter des calculs inutiles, ce vecteur est décomposé de façon à faire directement intervenir l'historique $\mathbf{X}_{t-dt}$ et les entrées u_t du système. L'équation matricielle devient alors :

$$\mathbf{A} \cdot \mathbf{X}_t = \mathbf{H} \cdot \mathbf{X}_{t-dt} + \mathbf{E} \cdot u_t \tag{3.2}$$

où H contient les coefficients multiplicatifs des termes de l'historique $\mathbf{X}_{t-dt}$ et E contient les coefficients multiplicatifs des entrées. Le but de l'algorithme présenté ici est donc, d'une part, de générer les matrices $\mathbf{A}$, $\mathbf{H}$ et $\mathbf{E}$ afin de permettre la simulation du système ; et d'autre part, d'initialiser le vecteur historique $\mathbf{X}_{t-dt}$ afin que la simulation commence en régime permanent.

La construction des matrices nécessite que l'algorithme modélise les différentes fonctions pouvant être utilisées par l'utilisateur (ici : fonction de transfert, forme d'état, somme et gain) sous une forme adaptée. C'est-à-dire que le comportement de l'élément va être exprimé par une équation scalaire ou matricielle en temps discret reliant les variables du système au temps t, et dans certain cas leur historique à l'instant $t - dt$ avec dt le pas de temps de la simulation. Ainsi les fonctions de transfert ne pourront pas être intégrées telles quelles et devront être transformées en une forme d'état, conformément à la méthode proposée au chapitre précédent. Ces équations iront, unes à unes, remplir les lignes des matrices souhaitées. La construction des matrices est achevée par l'ajout des lignes représentant les variables de $\mathbf{X}$ qui font partie des entrées du système u. Il s'agit d'inclure dans le système matricielle des équations du type $\mathbf{X}_{t,1} = u_t$, avec u la variable d'entrée du diagramme-bloc.

L'initialisation du vecteur historique $\mathbf{X}_{t-dt}$ se fait grâce à la résolution d'un autre système matriciel.

$$\mathbf{A}_i \cdot \mathbf{X}_0 = \mathbf{b}_i \tag{3.3}$$

Les matrices de ce dernier sont construites par l'algorithme en parallèle des matrices dédiées à la simulation en utilisant cette fois-ci l'équation de chaque élément en régime permanent. Le système est ensuite complété en ajoutant les équations représentant les variables connues en régime permanent.

Pour remplir sa mission, l'algorithme a besoin d'être en mesure de repérer l'entrée (ou les entrées) et la sortie d'un élément du système de contrôle modélisé. De plus il doit pouvoir repérer une variable en particulier au sein du vecteur $\mathbf{X}_t$ regroupant toutes les variables. Enfin il doit être capable identifier la ou les variable(s) correspondant à l'entrée (ou les entrées) du système de contrôle. Cela permettra d'assigner correctement ces variables à u en construisant la matrice $\mathbf{E}$ définie à l'équation 3.2

3.1.2 Principe de fonctionnement de l'algorithme

Dans EMTP les systèmes de contrôle sont modélisés sont la forme de diagramme-bloc. Pour les caractériser afin de générer les matrices permettant leur simulation et initialiser leurs variables, certaines données sont nécessaires :

- **Liste des éléments du système :** contenant la caractérisation de chacun d'entre-eux

  - Entrée : objet chaîne de caractère permettant de reconnaître qu'il s'agit d'une entrée du système, aucune autre donnée n'est nécessaire.
  - Gain : valeur du facteur multiplicatif.
  - Sommation : liste des signes de la somme ou de la soustraction.
  - Forme d'état : matrices du système d'équations.
  - Fonction de transfert : coefficients du numérateur et du dénominateur.
- **Matrice d'adjacence M :** permettant de connaître les entrées et les sorties d'un élément du diagramme. Si $\mathbf{M}_{ij} = 1$ alors l'élément du diagramme-bloc en j^e position de la liste des élément est une sortie de l'élément en i^e position.
- **Données en régime permanent :** valeur(s) de la (des) variable(s) connue(s) pour l'initialisation en régime permanent. Il s'agit la plupart du temps des sorties du système de contrôle, obtenu grâce à un écoulement de puissance sur la modélisation du réseau électrique connecté au système de contrôle étudié. Afin que l'équation matricielle liée à l'initialisation soit inversible, il faut que le nombre de variables connues en régime permanent soit égal au nombre de variables d'entrée du système.
- **Pas de temps de la simulation :** pour la discrétisation des équations de certains éléments.

Basé sur ces données, l'algorithme se décompose en quatre étapes :

1. La liste des blocs est parcourue une première fois afin, d'une part, de connaître le nombre total de variables du système. Et d'autre part pour repérer la position des variables de sortie de chaque élément du diagramme-bloc au sein des vecteurs $\mathbf{X}_t$ et $\mathbf{X}_{t-dt}$. Cela se fait en créant une liste, que nous appellerons L_{ind}, contenant les indices de leur position. Cette étape est cruciale pour pouvoir générer convenablement les matrices et récupérer les données voulues durant la simulation.
2. La liste des blocs est parcourue une seconde fois afin de remplir les matrices $\mathbf{A}$,$\mathbf{H}$ et $\mathbf{A}_i$ definies par les équations 3.2 et 3.3. Les blocs sont un à un modélisés sous la forme d'une équation en temps discrétisé reliant différentes variables de $\mathbf{X}_t$ et $\mathbf{X}_h$. Les coefficients des matrices sont remplis correctement à l'aide de L_{ind} et de la matrice d'adjacence.
3. Les matrices $\mathbf{A}$ et $\mathbf{A}_i$ sont ensuite complétées ainsi que $\mathbf{E}$ et $\mathbf{b}_i$ (définis par les équations 3.2 et 3.3) générés en traitant les équations d'entrée et des variables connues en régime permanent.
4. Enfin, le système $\mathbf{A}_i \cdot \mathbf{X}_0 = \mathbf{b}_i$ est résolu afin d'initialiser l'historique en régime permanent.

Remarques :

- Les blocs 'entrée' sont nécessaires au fonctionnement de l'algorithme afin de représenter les variables d'entrée correspondantes. Par construction, tous les blocs ont une variable de sortie présente dans X_t.
- Un bloc n'a qu'une variable de sortie (scalaire ou vectorielle), mais cette variable peut être l'entrée de plusieurs blocs.
- Hormis les variables internes à la simulation des fonctions de transfert et des formes d'état, toutes les variables doivent avoir la même taille. Par exemple, si la variable d'entrée est un vecteur à 2 éléments, alors cela doit être le cas pour la variable de sortie de chaque bloc. De plus, les deux éléments d'une de ces variables doivent être connu en régime permanent pour l'initialisation.

3.1.3 Étude des fonctions traitées

Cette partie traite de la modélisation des éléments pouvant être présents dans le système de contrôle défini par l'utilisateur. L'algorithme parcours les blocs de la liste un par un et rempli les lignes des matrices à partir des équations présentées ici.

Gain et Sommation

Il s'agit des éléments les plus simples traités ici. Ils représentent les équations :

$$\text{Gain} : X_{sortie} = K \cdot X_{entree} \tag{3.4}$$

$$\text{Somme} : X_{sortie} = \sum_{i \in entrees} (\pm 1) \cdot X_i \tag{3.5}$$

Ces équations font directement intervenir les variables de X_t pour la simulation et X_0 pour l'initialisation. Elles peuvent directement être injectées dans les matrices en construction.

Elles formeront alors, au sein des matrices A et A_i une ligne de la forme :

$$\text{Gain} : [\cdots K \cdots -1 \cdots] \begin{bmatrix} \vdots \\ X_{entree1} \\ \vdots \\ X_{sortie} \\ \vdots \end{bmatrix} = 0 \tag{3.6}$$

$$\text{Somme} : [\cdots +1 \cdots -1 \cdots -1 \cdots] \begin{bmatrix} \vdots \\ X_{entree1} \\ \vdots \\ X_{entree2} \\ \vdots \\ X_{sortie} \\ \vdots \end{bmatrix} = 0 \tag{3.7}$$

La position de la variable de sortie du bloc dans X_t est obtenue grâce à la liste L_{ind}. Par exemple, s'il s'agit du j^e élément de la liste des blocs du système de contrôle alors : $x_{sortie} = X_t[L_{ind}(j)]$. L'entrée du bloc gain ou les entrées du bloc sommation sont obtenues à partir de la matrice d'adjacence. Toujours pour le même j^e élément, la j^e colonne de la matrice d'adjacence est parcourue pour relevé les indices des "1". S'il y a un 1 à la k^e ligne de la j^e colonne alors l'élément k est une entrée du bloc j. L'indice de la variable correspondante dans X_t est obtenue également par $L_{ind}(k)$, soit $x_{entree} = X_t[L_{ind}(k)]$.

Forme d'état

Forme d'état en temps continu : Il s'agit d'un lien entre une entrée $u = x_{entree}$ et une sortie $y = x_{sortie}$ faisant intervenir un état intermédiaire nommé $\mathbf{X}_{fe}$ (fe pour forme d'état) dans un système linéaire de la forme :

$$\begin{cases} \dot{\mathbf{X}}_{fe} = \mathbf{A}\ \mathbf{X}_{fe} + \mathbf{B}u \\ \quad y = \mathbf{C}\mathbf{X}_{fe} + Du \end{cases} \tag{3.8}$$

La simulation nécessite de discrétiser ce système afin d'exprimer $\dot{X}$ en fonction de X_t et X_{t-dt}. Pour cela on utilise la discrétisation trapézoïdale qui utilise les approximations suivantes :

$$\dot{X}(t) \to \frac{X_t - X_{t-dt}}{dt} \qquad X(t) \to \frac{X_t + X_{t-dt}}{2} \tag{3.9}$$

Et on obtient le système discrétisé, où $\mathbf{I}$ est la matrice identité :

$$\begin{cases} \left(\mathbf{I} - \dfrac{dt}{2}\mathbf{A}\right)\mathbf{X}_{fe,t} - \dfrac{dt}{2}Bu_t = \left(\mathbf{I} + \dfrac{dt}{2}\mathbf{A}\right)\mathbf{X}_{fe,t-dt} + \dfrac{dt}{2}\mathbf{B}u_{t-dt} \\ \mathbf{C}\mathbf{X}_{fe,t} + Du_t - y_t = 0 \end{cases} \tag{3.10}$$

avec d'un côté les variables à l'instant t et de l'autre celles à l'instant $t - dt$. Comme pour les blocs gain et sommation, la matrice d'adjacence et la liste des indices L_{ind} sont utilisées pour remplir correctement les matrices A et H. Cela donne les lignes :

$$\begin{bmatrix} \cdots & \left(\mathbf{I} - \frac{dt}{2}\mathbf{A}\right) & \cdots & -\frac{dt}{2}\mathbf{B} & \cdots & 0 & \cdots \\ \cdots & \mathbf{C} & \cdots & D & \cdots & -I & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ \mathbf{X}_{fe} \\ \vdots \\ x_{entree} \\ \vdots \\ x_{sortie} \\ \vdots \end{bmatrix} \tag{3.11}$$

$$= \begin{bmatrix} \cdots & \left(\mathbf{I} + \frac{dt}{2}\mathbf{A}\right) & \cdots & \frac{dt}{2}\mathbf{B} & \cdots \\ \cdots & 0 & \cdots & 0 & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ \mathbf{X}_{fe,h} \\ \vdots \\ x_{entree,h} \\ \vdots \end{bmatrix}$$

L'initialisation se fait en régime stationnaire : $\dot{X} = 0$. Il s'agit d'une hypothèse pas toujours vérifiée qui a été discuté à la remarque 2.3.2 à la page 31. Aucune discrétisation n'est nécessaire, le système obtenu peut directement être exprimé sous la forme appropriée.

$$\begin{cases} \mathbf{A}\ \mathbf{X}_{fe} + \mathbf{B}u = 0 \\ \mathbf{C}\mathbf{X}_{fe} + Du - y = 0 \end{cases} \tag{3.12}$$

Ce qui donne les lignes dans le système d'initialisation final, avec $\mathbf{I}$ la matrice identité :

$$\begin{bmatrix} \cdots & \mathbf{A} & \cdots & \mathbf{B} & \cdots & \mathbf{0} & \cdots \\ \cdots & \mathbf{C} & \cdots & \mathbf{D} & \cdots & -\mathbf{I} & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ \mathbf{X}_{interne} \\ \vdots \\ \mathbf{X}_{entree} \\ \vdots \\ \mathbf{X}_{sortie} \\ \vdots \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{3.13}$$

Forme d'état discrétisée : Si l'utilisateur souhaite simuler une forme d'état déjà discrétisée :

$$\begin{cases} \mathbf{X}_{fe,t} = \mathbf{A}\ \mathbf{X}_{fe,t-dt} + \mathbf{B}u_{t-dt} \\ \quad y_t = \mathbf{C}\mathbf{X}_{fe,t} + Du_t \end{cases} \tag{3.14}$$

Le système d'initialisation devient alors :

$$\begin{cases} (\mathbf{A} - \mathbf{I})\mathbf{X}_{fe,0} + Bu_0 = 0 \\ \mathbf{C}\mathbf{X}_{fe,0} + Du_0 - y_0 = 0 \end{cases} \tag{3.15}$$

Ces systèmes peuvent directement être injectés dans les matrices $\mathbf{A}$ et $\mathbf{H}$ définies par l'équation 3.2 avec u la variable d'entrée de la forme d'état (et non du diagramme-bloc en entier) et y sa variable de sortie.

Fonction de transfert

La relation entre l'entrée u et la sortie y du bloc est de la forme :

Temps continu (domaine de Laplace) :

$$H(s) = \frac{b_0 s^n + b_1 s^{n-1} + \cdots + b_n}{s^n + a_1 s^{n-1} + \cdots + a_n} \tag{3.16}$$

Temps discrétisé (domaine en Z) :

$$H(z) = \frac{\beta_0 z^n + \beta_1 z^{n-1} + \cdots + \beta_n}{z^n + \alpha_1 z^{n-1} + \cdots + \alpha_n} \tag{3.17}$$

Pour simuler une telle fonction de transfert, on utilise la méthode du chapitre précédent, en définissant une forme d'état ayant le même comportement entrée-sortie : ([**?**])

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} 0 & 0 & \cdots & 0 & -a_n \\ 1 & 0 & \cdots & 0 & -a_{n-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -a_1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_n - a_n b_0 \\ \vdots \\ b_1 - a_1 b_0 \end{bmatrix} u \\ y = \begin{bmatrix} 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + b_0 u \end{cases} \tag{3.18}$$

Le nombre d'états internes est égal à l'ordre du dénominateur de l'élément. Dans le cas de fonction de transfert Multi-Entrées Multi-Sortie, on utilise aussi la méthode du chapitre précédent.

On applique ensuite le même processus que pour les formes d'états ci-dessus.

3.2 Démonstration du fonctionnement sur un exemple simple

Afin d'expliquer son fonctionnement, l'algorithme présenté dans ce chapitre va être appliqué pas à pas sur l'exemple fictif de la figure 3.1. Il s'agit d'une boucle de rétroaction comprenant une fonction de transfert et un gain. Le but est de pouvoir simuler ce système en commençant directement en régime permanent, et pour cela on veut obtenir les matrices du système

$$\mathbf{A} \cdot \mathbf{X}_t = \mathbf{H} \cdot \mathbf{X}_h + \mathbf{E} \cdot u_t \quad \text{avec} \quad \mathbf{X}_t = \begin{bmatrix} x_1 & x_2 & x_3 & x_{ft} & x_4 \end{bmatrix}' \tag{3.19}$$

où x_{ft} est l'état intermédiaire qui résulte de la transformation de la fonction de transfert H en forme d'état. L'algorithme doit aussi donné l'historique initialisé en régime permanent.

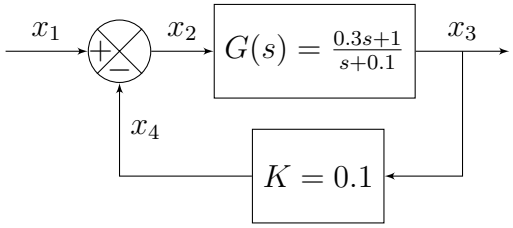

Figure 3.1 Diagramme-bloc de l'exemple traité

Pour cela, il construit et résout le système :

$$\mathbf{A}_i \cdot \mathbf{X}_{h,0} = \mathbf{b}_i \tag{3.20}$$

3.2.1 Données d'entrée de l'algorithme

Afin de pouvoir caractériser le système de contrôle et calculer les matrices nécessaires à la simulation et à l'initialisation, les données suivantes sont données à l'algorithme :

- la liste des blocs : `liste_blocs = {'entrée',sommateur,G,K}` avec
  - `sommateur` = [+1 -1], la liste des signes
  - `H` = `tf`([0.3 1],[1 0.1]), un objet Matlab contenant les coefficients de la fonction de transfert
  - `K` = 0.1, la valeur du gain
- la matrice d'adjacence :

$$\mathbf{M} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{array} \tag{3.21}$$

  Cette matrice indique, par exemple, que la sortie du bloc n°1 (le bloc Entrée) est le bloc n°2 (le bloc Sommateur) car $\mathbf{M}_{1,2} = 1$.
- les données connues en régime permanent. Ici il n'y a qu'une entrée, donc une seule donnée est nécessaire. On suppose connue en régime permanent $x_3 = 5$.
- le pas de temps : dt = 0.001

3.2.2 Application pas à pas de l'algorithme

1^e étape : taille du système et indice des variables de sortie

Le but de cette étape est de déterminer le nombre total de variables intervenant dans le système, c'est à dire le nombre de variables de sortie des blocs ajouté au nombre de variables internes des formes d'état et fonctions de transfert. De plus, les indices de la variable de sortie de chaque bloc dans le vecteur $\mathbf{X}_t$ sont relevés ainsi que l'indice de l'entrée du système. Cela permettra un remplissage des matrices **A,H,E** et $\mathbf{A}_i$ plus aisé. Pour ce faire, la liste des blocs est parcourue en mettant à jour pas à pas le nombre de variables et la liste des indices.

- nombre_variable = 0
- liste_indices = [0 0 0 0]
- indice_entree = 0

1. bloc 'entrée'
   - nombre_variable = 1 *(implémentation de 1 car 1 variable de sortie)*
   - liste_indices = [1 0 0 0]
   - indice_entree = 1 *(l'entrée est reconnue en testant s'il s'agit d'une chaîne de caractère)*
2. bloc sommateur
   - nombre_variable = 2
   - liste_indices = [1 2 0 0]
3. bloc G
   - nombre_variable = 4 *(implémentation de 2 pour 1 variable de sortie et 1 variable interne nécessaire à la simulation (nombre donné par l'ordre de la fonction de transfert))*
   - liste_indices = [1 2 3 0]
4. bloc K
   - nombre_variable = 5
   - liste_indices = [1 2 3 5]

2^e étape : construction des matrices

La taille du système étant connue, les matrices vides A, H et A_i sont créées. Elles vont être remplies ligne par ligne en parcourant une nouvelle fois la liste des blocs.

Chaque bloc va passer une série de test afin de reconnaître de quel élément il s'agit. Ensuite, à l'aide de la matrice d'adjacence et de la liste des indices, l'algorithme va déterminer quels sont les indices des entrées et de la sortie du bloc dans le vecteur des variables $\mathbf{X}_t$. Cela permet de savoir quelles sont les colonnes des coefficients à remplir dans les matrices $\mathbf{A}$, $\mathbf{H}$ et $\mathbf{A}_i$ pour modéliser l'élément traité. La ligne à remplir est, elle, indiquée par un marqueur mis à jour à chaque fois qu'une ligne est complétée.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{A}_i = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

`ligne_traitée = 1` *(Marqueur de la ligne à remplir)*

1. `bloc 'entrée' : rien à faire`
2. `bloc sommateur = [+1 -1]`

$$\text{L'équation du bloc est } (+1)x_1 + (-1)x_4 - x_2 = 0 \tag{3.22}$$

- `Reconnaissance du bloc : liste de double` $\Rightarrow$ `bloc de sommation`
- `Entrées du bloc = [`x_1`,`x_4`]` *(Lecture de la colonne de la matrice d'adjacence correspondant au numéro du bloc traité, ici : 2).*
- `Indices entrées du bloc = [liste_indice(1),liste_indices(4)]=[1,5]` *(On sait maintenant que les coefficients des variables d'entrées vont être remplis dans les colonnes 1 et 5).*
- `Indice de sortie du bloc = liste_indice(2) = 2` *(Le coefficient de la variable de sortie va être rempli dans la* 2^e *colonne).*
- `Implémentation de l'équation du bloc dans les matrices` (`ligne_traitée` *indique que la première ligne de la matrice va être complétée.)*

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{A}_i = \begin{bmatrix} 1 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- `ligne_traitée = 2` *(Passage à la ligne suivante)*

3. `bloc G = tf([0.3 1],[1 0.1])`

$$G(s) = \frac{x_3(s)}{x_2(s)} = \frac{0.3s + 1}{s + 0.1} \tag{3.23}$$

- `Reconnaissance du bloc : objet tf` $\Rightarrow$ `fonction de transfert`

- Passage à la forme d'état

$$\begin{cases} \dot{x}_{tf} = -0.1 \cdot x_{tf} + 0.97 \cdot x_2 \\ x_3 = 1 \cdot x_{tf} + 0.3 \cdot x_2 \end{cases} \tag{3.24}$$

- Sous Matlab, une fonction de transfert en temps continu est définit avec un pas de temps nul. C'est ainsi que l'algorithme reconnaît il s'agit d'une fonction de transfert en temps continu. La discrétisation trapézoïdale est appliquée :

$$\begin{cases} 1.005 \cdot x_{tf,t} - 0.0485 \cdot x_{2,t} = 0.9950 \cdot x_{tf,t-dt} + 0.0485 \cdot x_{2,t-dt} \\ 1 \cdot x_{tf,t} + 0.3 \cdot x_{2,t} - x_{3,t} = 0 \end{cases} \tag{3.25}$$

- Initialisation

$$\begin{cases} -0.1 \cdot x_{tf} + 0.97 \cdot x_2 = 0 \\ 1 \cdot x_{tf} + 0.3 \cdot x_2 - x_3 = 0 \end{cases} \tag{3.26}$$

- Indice entrée (x_2) = 2, Indice sortie (x_3) = 3.
  (Par construction, les états intermédiaires nécessaire à la simulation de la fonction de transfert sont placés juste après la variable de sortie dans X_t. Ici il n'y a qu'un état intermédiaire, et il se trouve à l'indice 4.)
- Implémentation des matrices :

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 0 & 0 & -1 \\ 0 & 0.3 & -1 & 1 & 0 \\ 0 & -0.0485 & 0 & 1.005 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0485 & 0 & 0.995 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{A}_i = \begin{bmatrix} 1 & -1 & 0 & 0 & -1 \\ 0 & 0.3 & -1 & 1 & 0 \\ 0 & 0.97 & 0 & -0.1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- ligne_traitée = 4
  (2 lignes ont été remplie, ainsi la prochaine ligne vide est la 4^e.)

4. bloc K = 0.1

$$0.1 \cdot x_3 - x_4 = 0 \tag{3.27}$$

- Reconnaissance du bloc : double ⇒ gain
- Indice entrée = 3, indice sortie = 5
- Implémentation des matrices :

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 0 & 0 & -1 \\ 0 & 0.3 & -1 & 1 & 0 \\ 0 & -0.0485 & 0 & 1.005 & 0 \\ 0 & 0 & 0.1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0485 & 0 & 0.995 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\mathbf{A}_i = \begin{bmatrix} 1 & -1 & 0 & 0 & -1 \\ 0 & 0.3 & -1 & 1 & 0 \\ 0 & 0.97 & 0 & -0.1 & 0 \\ 0 & 0 & 0.1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- ligne_traitée = 5

3^e étape : ajout des équations des variables connues

Les équations liées aux variables connues sont implémentés : la dernière ligne des matrices $\mathbf{A}$ et $\mathbf{A}_i$ sont remplies, la matrice $\mathbf{E}$ et le vecteur $\mathbf{b}_i$ sont construits. Ici, $\mathbf{E}$ est aussi un vecteur car le diagramme-bloc n'a qu'une entrée scalaire.

Pour l'initialisation il s'agit de l'équation $x_3 = 5$, car c'est la donnée connue en régime permanent.

- indice de la variable connue = 3
- remplissage de $\mathbf{A}_i$: $\mathbf{A}_i$(ligne_traitée, indice variable connue) = 1

$$\mathbf{A}_i = \begin{bmatrix} 1 & -1 & 0 & 0 & -1 \\ 0 & 0.3 & -1 & 1 & 0 \\ 0 & 0.97 & 0 & -0.1 & 0 \\ 0 & 0 & 0.1 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

- construction de $\mathbf{b}_i$: vecteur nul sauf $\mathbf{b}_i$(ligne_traitée) = donnée

$$\mathbf{b}_i = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 5 \end{bmatrix}$$

Pour la simulation il s'agit de désigner x_1 comme l'entrée du système.

- Implémentation de $\mathbf{A}$: $\mathbf{A}$(ligne_traitée,indice_entrée) = 1

$$\mathbf{A} = \begin{bmatrix} 1 & -1 & 0 & 0 & -1 \\ 0 & 0.3 & -1 & 1 & 0 \\ 0 & -0.0485 & 0 & 1.005 & 0 \\ 0 & 0 & 0.1 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

- construction de $\mathbf{E}$: vecteur nul sauf $\mathbf{E}$(ligne_traitée) = 1

$$\mathbf{E} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

4^e étape : résolution de l'initialisation

Le système $\mathbf{A}_i \cdot \mathbf{X}_0 = \mathbf{b}_i$ est résolu afin d'obtenir l'historique initialisé. Ici on obtient :

$$\mathbf{X}_0 = \begin{bmatrix} 1 \\ 0.5 \\ 5 \\ 4.85 \\ 0.5 \end{bmatrix} \tag{3.28}$$

3.2.3 Simulation et résultat

La simulation se fait effectuant, à chaque pas de temps :

$$\mathbf{X}_t = (\mathbf{A}^{-1}\mathbf{H}) \cdot \mathbf{X}_h + (\mathbf{A}^{-1}\mathbf{E}) \cdot u \tag{3.29}$$

Avec u l'entrée du système, et $\mathbf{X}_h = \mathbf{X}_0$ à la première itération. Les matrices $\mathbf{A}$, $\mathbf{H}$ et $\mathbf{E}$ étant constantes, le calcul de $\mathbf{A}^{-1}\mathbf{H}$ et $\mathbf{A}^{-1}\mathbf{E}$ ne se fait qu'une seule fois avant la simulation. Enfin la sortie x_3 est extraite de $\mathbf{X}_t$ grâce à la liste d'indices générée par l'algorithme.

Le rendu de la simulation de l'exemple traité ici est donnée sur la figure 3.2 (fichier `Ex_algorithme.m`). L'initialisation est bien menée : la simulation démarre en régime permanent. De plus, le système prend plus de 30s à établir son régime permanent. Ainsi, sans initialisation automatique, il aurait fallu simuler inutilement le comportement du système sur 30s avant d'effectuer les perturbations voulues, ce qui impose du temps de calcul supplémentaire.

Conclusion du chapitre

Ce chapitre a présenté un algorithme générant les matrices nécessaires à la simulation d'un système de contrôle linéaire ainsi que l'initialisation des variables associées en régime permanent. Cela se fait à partir de données préalables, notamment un certain nombre de variables connues en régime permanent. De plus la création des matrices requiert une modélisation particulière des éléments du système, comme la transformation des fonctions de transfert en forme d'état. Enfin l'initialisation des variables se fait par la résolution d'un système matriciel construit simultanément aux matrices permettant la simulation. Un tel algorithme permet d'éviter un faux régime transitoire en début de simulation où le système converge vers son régime permanent.

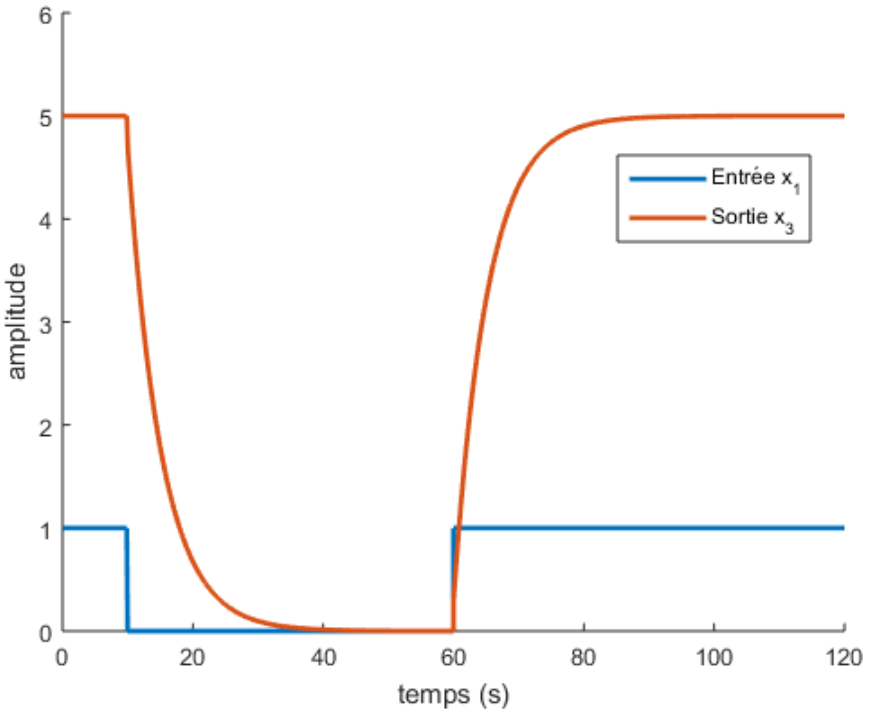

Figure 3.2 Simulation de l'exemple traité via les matrices générées par l'algorithme

CHAPITRE 4 CAS PRATIQUE : CHANGEMENT DE CONTRÔLEUR DANS UN MICRO-RÉSEAU

L'initialisation en régime permanent n'est pas seulement utile pour gagner du temps en début de simulation. Elle peut aussi permettre de démarrer un élément du système de contrôle en pleine opération tout en lui imposant une sortie donnée. Ce chapitre montrera une application de l'initialisation en régime permanent dans ce contexte sur un cas tiré de l'article : H. Karimi, E.J. Davison, R. Iravani, "Multivariable Servomechanism Controller for Autonomous Operation of a Distributed Generation Unit : Design and Performance Evaluation", *IEEE Transaction on Power Delivery* (Vol. 25, n°2), 2010 (voir [**?**]).

Dans ce chapitre, le micro-réseau étudié sera tout d'abord présenté. Ensuite, un contrôleur sera construit dans le cas où le micro-réseau est connecté au réseau principal. Enfin, l'initialisation du contrôleur dans le cas où le micro-réseau est isolé sera développée et son utilité démontrée.

4.1 Présentation du micro-réseau

L'étude se porte sur le micro-réseau schématisé sur la figure 4.1 initialement connecté à un réseau principal. Ce micro-réseau comporte une unité de production décentralisée connectée au réseau via un convertisseur source de tension (VSC pour *Voltage source converter*). Ce convertisseur est relié au point de connexion avec le réseau principal par un transformateur élévateur. Une charge est connectée au même point, elle est modélisée par un circuit RLC parallèle triphasé. Le système est initialement connecté à un réseau principal modélisé par une source parfaite de tension. La tension en sortie de VSC est alors commandé par un contrôleur proportionnel-intégral (PI), à définir, via le contrôle du courant traversant le système. Lorsque la connexion cesse, le système bascule sur un contrôleur à entrées et sorties multiples (MIMO pour *Multi-Input Multi-Output)*, décrit dans l'article. Les paramètres de l'ensemble sont résumés sur le tableau 4.1.

Le but de l'étude est d'initialiser le contrôleur MIMO au moment de l'isolation du micro-réseau tel que la valeur de la commande générée soit la même que la commande générée par le contrôleur PI juste avant la déconnexion. Cela permettra d'avoir une transition moins abrupte entre les deux états 'connecté au réseau' et 'isolé'. Les calculs et simulations viennent des fichiers `MSC.m` et `TwoControllers.ecf` et des données contenues dans `controller_feb16.mat`.

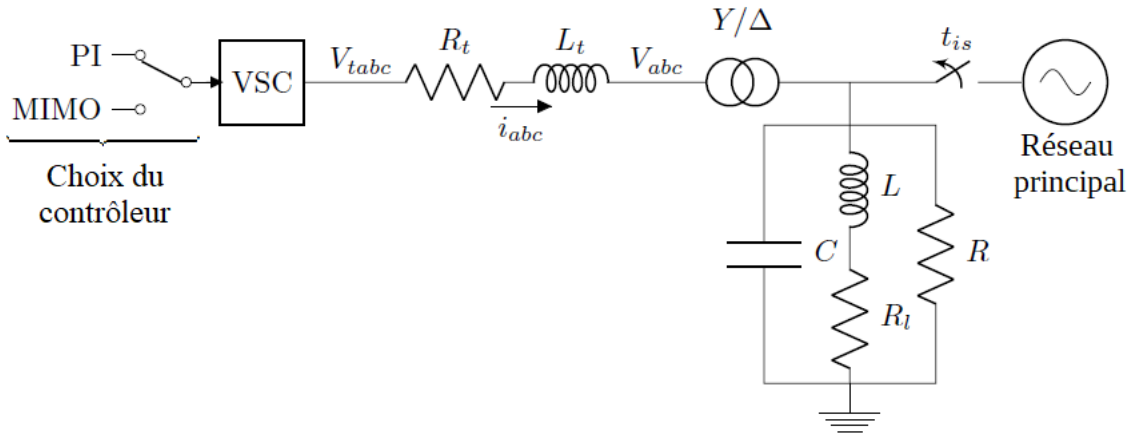

Figure 4.1 Schéma du micro-réseau étudié

Tableau 4.1 Paramètres du micro-réseau étudié et de la simulation effectuée

Paramètre	Valeur
Résistance du filtre du convertisseur VSC	$R_t = 1.5$ mΩ
Inductance du filtre du convertisseur VSC	$L_t = 300$ μH
Résistance de la charge	$R = 76$ Ω
Inductance de la charge	$L = 111.9$ mH
Résistance propre	$R_l = 0.3$ Ω
Capacité de la charge	$C = 62.86$ μF
Ratio du transformateur	0.6/13.8
Tension du réseau principal	13.8 kV RMS LL
Fréquence du réseau	$f = 60$ Hz
Instant de l'isolation	$t_{is} = 0.5$ s
Commande du contrôleur PI	$I_{d,ref} = 400$ I $I_{q,ref} = 0$ I
Commande du contrôleur MIMO	$V_{d,ref} = 424.3$ V $V_{q,ref} = 245.6$ V
Pas de temps de la simulation	dt = 1 μs

4.2 Construction du contrôleur PI du mode 'connecté'

Le but du contrôleur PI est de réguler la tension de la charge à partir de l'erreur entre le courant parcourant le micro-réseau et une valeur de référence. La construction de ce contrôleur passe tout d'abord par la modélisation du système sous forme d'une fonction de transfert. Avec les notations de la figure 4.1 :

$$V_{tabc} - V_{abc} = R_t i_{abc} + L_t \frac{di_{abc}}{dt} \tag{4.1}$$

dans le domaine dq0, cela donne :

$$V_{tdq} - V_{dq} = R_t i_{dq} + L_t \left[\frac{di_{dq}}{dt} + j\omega i_{dq} \right] \tag{4.2}$$

décomposition en partie réelle et imaginaire

$$\begin{cases} V_{td} - V_d + L_t \omega i_q = R_t i_d + L_t \dfrac{di_d}{dt} \\ V_{tq} - V_q - L_t \omega i_d = R_t i_q + L_t \dfrac{di_q}{dt} \end{cases} \tag{4.3}$$

soit :

$$\begin{cases} u_d = R_t i_d + L_t \dfrac{di_d}{dt} \\ u_q = R_t i_q + L_t \dfrac{di_q}{dt} \end{cases} \tag{4.4}$$

en posant :

$$u_d = V_{td} - V_d + L_t \omega i_q \tag{4.5}$$

$$u_q = V_{tq} - V_q - L_t \omega i_d \tag{4.6}$$

Cela donne 2 équations identiques et découplées. Prenons l'équation pour u_d dans le domaine de Laplace :

$$u_d = (R_t + sL_t) i_d \tag{4.7}$$

on a la même équation pour u_q. On a donc la fonction de transfert du micro-réseau :

$$G(s) = \frac{I_d(s)}{U_d(s)} = \frac{1}{L_t s + R_t} \tag{4.8}$$

cela implique un contrôleur PI de la forme suivante. Le diagramme-bloc de la boucle de contrôle est représenté sur la figure 4.2.

$$C(s) = K\frac{s + \frac{R_t}{L_t}}{s} \tag{4.9}$$

avec K choisi pour avoir une marge de phase de 60° à l'aide de l'outil matlab `sisotool`, figure 4.3, on obtient :

$$K = 0.2 \tag{4.10}$$

Ce choix de la constante K prend en compte la présence du filtre passe-bas H ci-dessous à l'entrée du contrôleur. Ce filtre permet d'éliminer les harmoniques de haute fréquence du courant (générés par des commutations par exemple).

$$H(s) = \frac{1000}{s + 1000} \tag{4.11}$$

Il est important de noter que la sortie du contrôleur PI donne u_d, mais la commande voulue est la tension $V_{td} = u_d + V_d - L_t\omega i_q$ d'après 4.5. Un calcul en sortie du contrôleur est donc effectué afin d'extraire la tension V_{td}.

L'implémentation du contrôleur sur EMTP est présenté sur la figure 4.4.

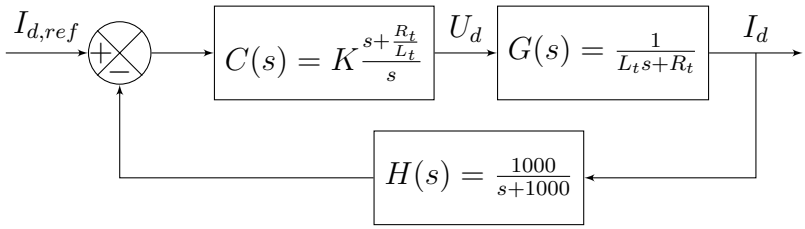

Figure 4.2 Diagramme-bloc de la boucle de contrôle

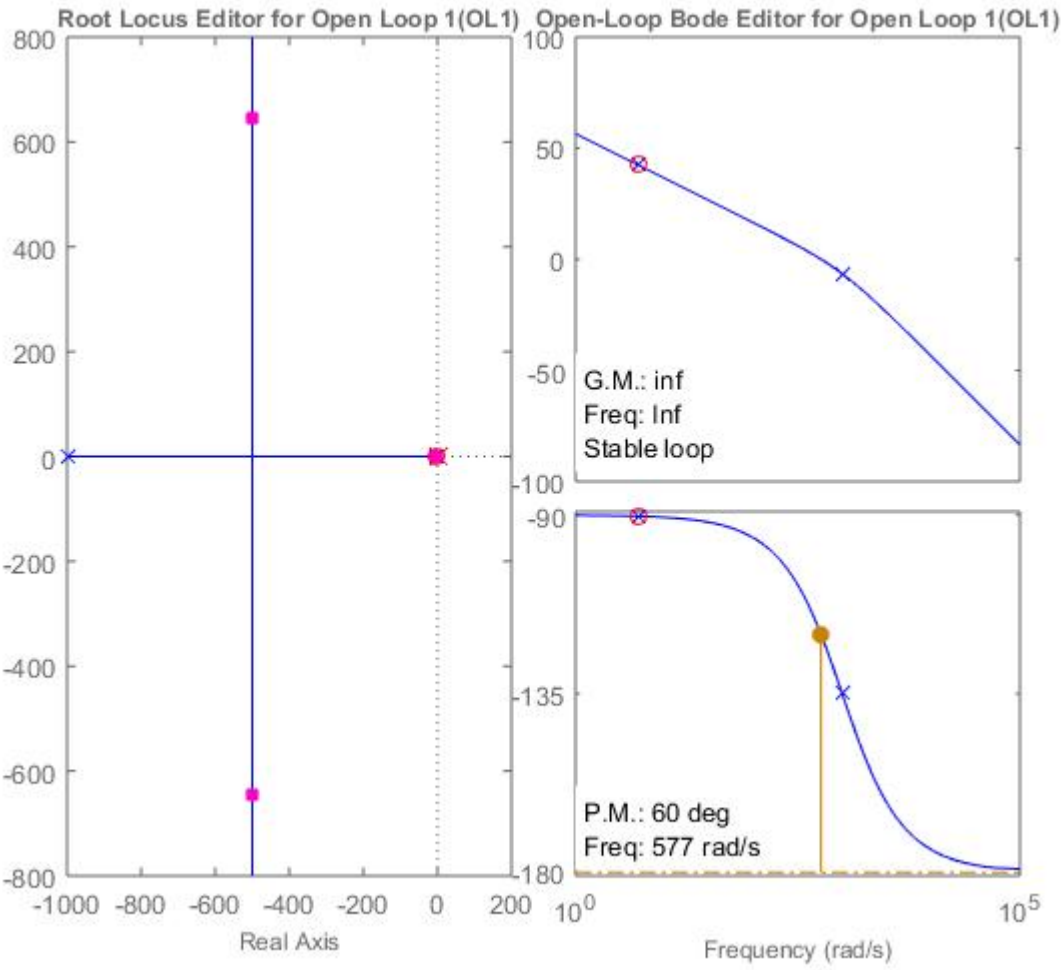

Figure 4.3 Performances du contrôleur PI avec $K = 0.2$

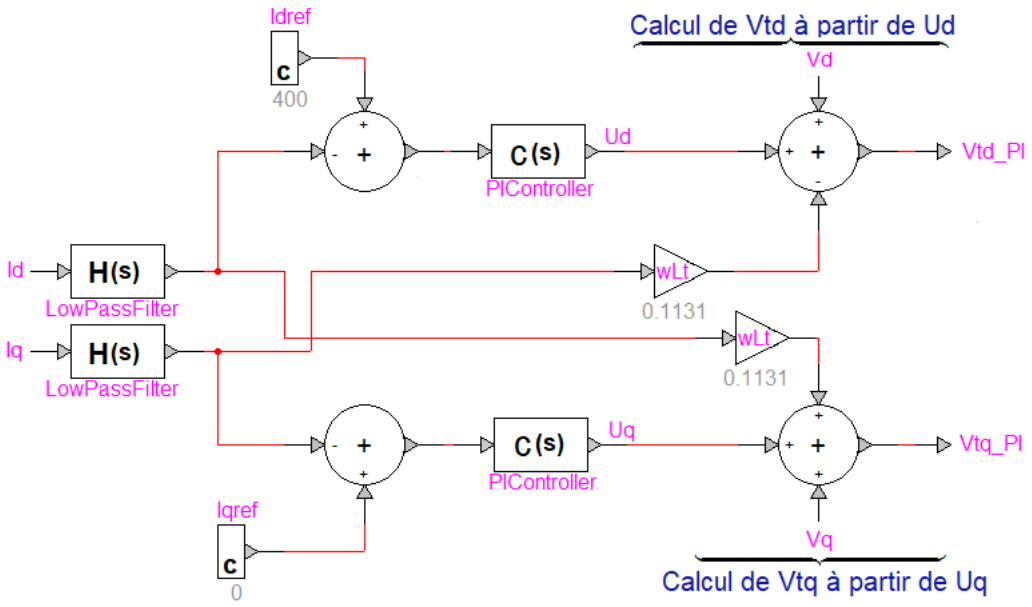

Figure 4.4 Implémentation du contrôleur PI sur EMTP

4.3 Initialisation du contrôleur MIMO du mode 'isolé'

Le contrôleur MIMO est la forme d'état :

$$\begin{cases} \dot{\beta} = \mathbf{A}_c\beta + \mathbf{B}_c\mathbf{e} \\ \mathbf{V}_{tdq} = \mathbf{C}_c\beta \end{cases} \tag{4.12}$$

où $\mathbf{V}_{tdq} = \begin{bmatrix} V_{td} \\ V_{tq} \end{bmatrix}$ et $\mathbf{e} = \begin{bmatrix} V_d \\ V_q \end{bmatrix} - \begin{bmatrix} V_{d,ref} \\ V_{q,ref} \end{bmatrix}$ est l'erreur sur la tension au primaire du transformateur par rapport à une référence défini dans le cas où le micro-réseau est isolé. β est de taille 12, et les valeurs des matrices $\mathbf{A}_c$, $\mathbf{B}_c$ et $\mathbf{C}_c$ sont explicitées dans l'article de référence [**?**].

Au moment où le système est isolé du réseau principal, on passe du contrôleur PI au contrôleur MIMO. Sans prendre de mesure particulière le VSC recevra un écart très important dans sa commande. Pour éviter cela, il est souhaité d'initialiser le contrôleur MIMO tel qu'au moment de l'isolation, sa sortie soit la même que la commande du VSC $\mathbf{V}_{tdq}$ générée grâce au contrôleur PI .

La valeur $\mathbf{V}_{tdq,is}$ juste avant l'isolation est supposée connue, et le contrôleur MIMO est

supposé en régime permanent $\dot{\beta} = 0$. On obtient alors le système :

$$\begin{cases} 0 = \mathbf{A}_c\beta_0 + \mathbf{B}_c\mathbf{e}_0 \\ \mathbf{V}_{tdq,is} = \mathbf{C}_c\beta_0 \end{cases} \tag{4.13}$$

La résolution de ce système donne une erreur $\mathbf{e}_0$ nulle et le vecteur d'état intermédiaire β_0 qui garantit d'obtenir la sortie voulue tout en restant en régime permanent tant que l'erreur $\mathbf{e}$ est nulle. La matrice $\mathbf{A}_c$ n'est pas inversible, ainsi $\beta_0 \neq 0$ même avec $\mathbf{A}_c\beta_0 = 0$.

Sous EMTP, cela a été implémenté en initialisant le contrôleur MIMO avec β_0 au début de la simulation. Et tant que le micro-réseau est connecté au réseau principal, on lui applique une entrée (erreur) nulle afin que le contrôleur reste en régime permanent et que β_0 et la sortie $\mathbf{V}_{tdq,is}$ soient conservés. (figure 4.5)

La simulation avec cette initialisation a été comparée dans les figures 4.6 et 4.7 avec le cas sans initialisation où le contrôleur MIMO est, durant toute la durée de la simulation, connecté à l'erreur réelle $e = V_{dq} - V_{dqref}$. Les paramètres de la simulation sont présentés sur le tableau 4.1. Pour ces courbes, le changement de contrôleur se fait au même moment que l'isolation.

Sans initialisation, la grande variation de V_{td} et V_{tq} se répercute sur la tension à la charge V_{dq}. On observe un petit décrochage de celle-ci après $0.5s$: il s'agit du résultat de la discontinuité de la commande. Dans le cas avec initialisation, la continuité de la commande du VSC permet une variation douce de la tension à la charge. Après ce décrochage (ou sans décrochage pour le cas avec initialisation), on observe une sorte de palier. L'explication la plus probable est qu'il s'agit d'une faible variations de la tension le temps que les perturbations dues à l'isolation se propagent au contrôleur et entraînent une modification de sa sortie. Néanmoins, la charge voit de grandes oscillations de tension, deux fois plus importante qu'avec initialisation. Du point de vu de l'usager, c'est particulièrement mauvais, car cela pourrait endommager le matériel électrique branché au réseau. Ainsi dans le cadre d'un système réelle le profil sans initialisation est à éviter. D'où la grande importance de l'initialisation dans le cas présent.

Dans un système réel la permutation des contrôleurs ne se fait pas exactement au moment de l'isolation, mais avec un court délai. Ce cas a été testé avec ce montage sur EMTP pour plusieurs écarts, et les résultats restent les mêmes. Comme le montre les figures 4.8 et 4.9 présentant le comportement des tensions du VSC et à la charge autour de l'isolation avec un délai de 1ms. La discontinuité de la tension au VSC est toujours observée sans initialisation, de même que la lente évolution des tensions après se décrochage.

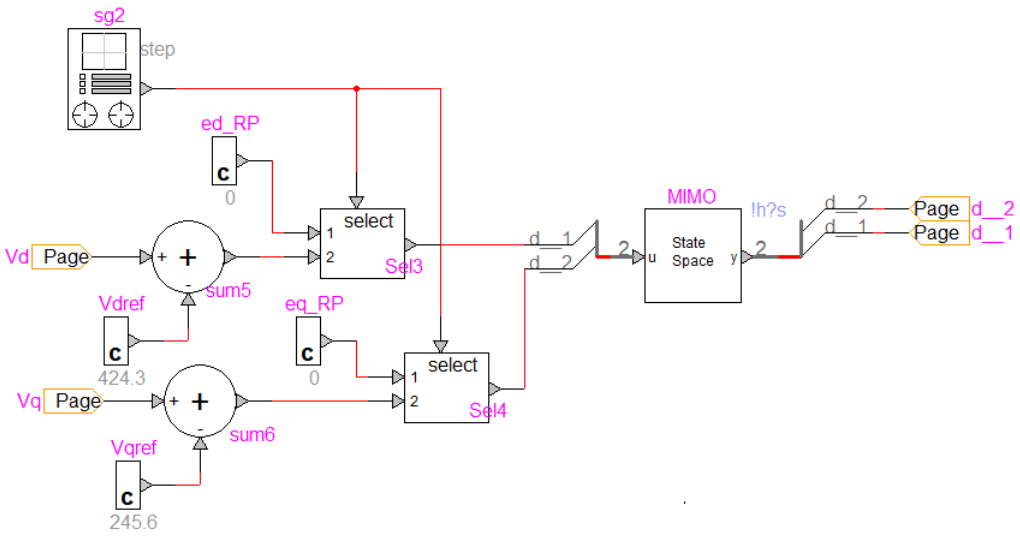

Figure 4.5 Implémentation du contrôleur MIMO avec initialisation sous EMTP, sans délai

On peut observer que la tension ne change pas entre le moment de l'isolation et le changement du contrôleur. Il se trouve que la sortie du contrôleur PI ne se modifie qu'au alentour de 0.5012s (pour une isolation à 0.5s). Cela pourrait être causé par le bilan de puissance PQ qui est nul à la sortie du VSC.

Conclusion du chapitre

Dans ce chapitre, l'utilité de l'initialisation en régime permanent développée dans cette maîtrise a été démontrée sur un cas concret : le changement de contrôleur dans un micro-réseau au moment de sa déconnexion avec le réseau principal. Pour cela le contrôleur PI pour le mode 'connecté au réseau' a été défini afin d'obtenir la valeur de sa sortie au moment de la coupure. Cette dernière a servi de donnée pour l'initialisation à rebours du contrôleur MIMO du mode 'isolé'. Cette initialisation permet de réduire considérablement la variations de la commande et de la tension de la charge.

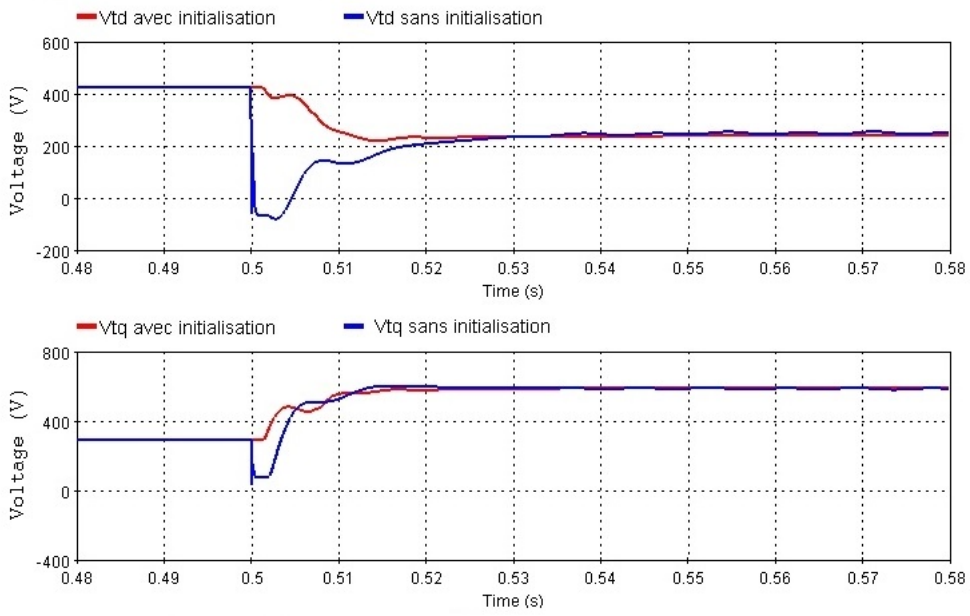

Figure 4.6 Comparaison de la commande du VSC avec et sans initialisation, sans délai entre l'isolation du micro-réseau et le changement du contrôleur

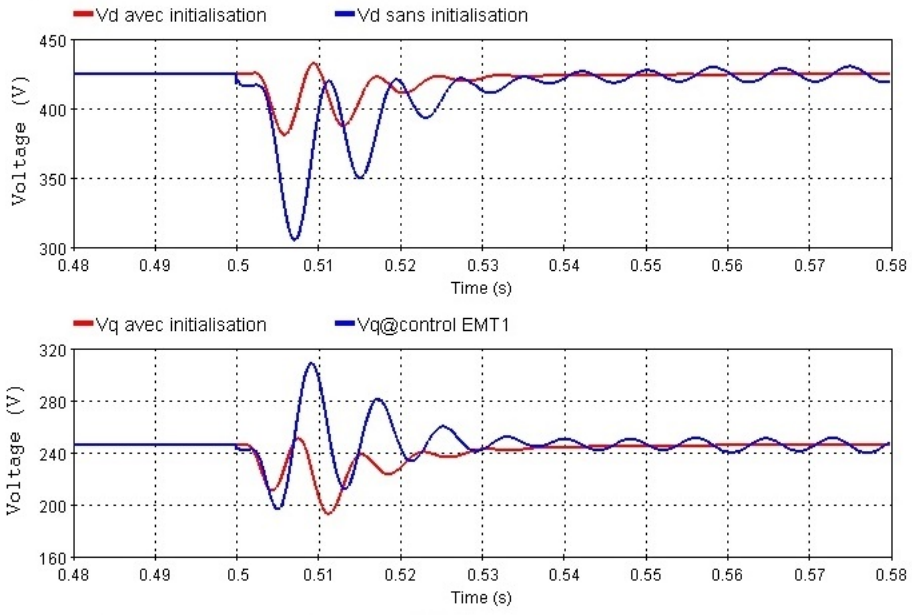

Figure 4.7 Comparaison de tension au primaire du transformateur avec et sans initialisation, sans délai entre l'isolation du micro-réseau et le changement du contrôleur

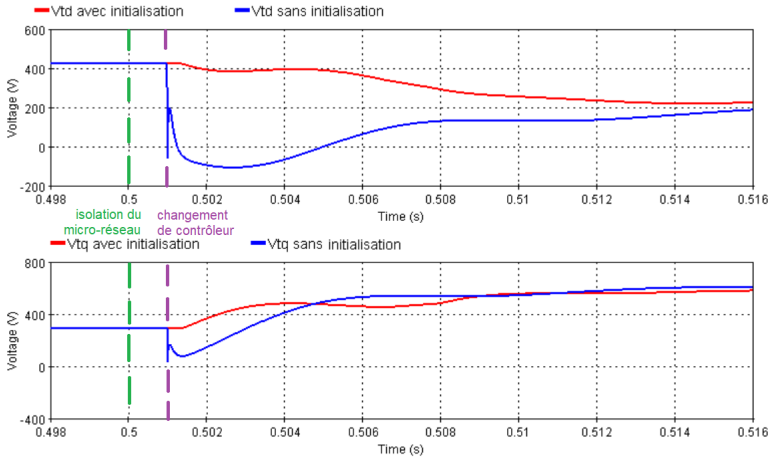

Figure 4.8 Comparaison de tension au primaire du transformateur avec et sans initialisation, avec délai de $1ms$ entre l'isolation du micro-réseau et le changement du contrôleur

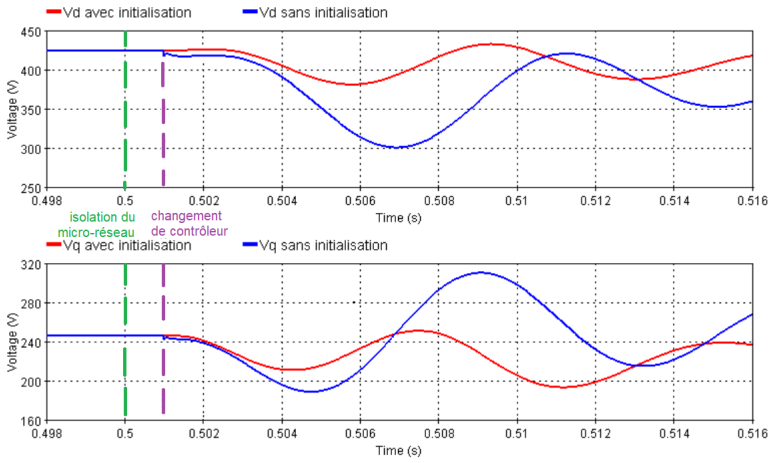

Figure 4.9 Comparaison de la commande du VSC avec et sans initialisation, avec délai de $1ms$ entre l'isolation du micro-réseau et le changement du contrôleur

CHAPITRE 5 CONCLUSION

5.1 Synthèse des travaux

L'objectif principal de ce mémoire était de développer une initialisation automatique pour la simulation des systèmes de contrôle. Ce faisant, il était souhaité de résoudre des problèmes d'instabilité de simulation des fonctions de transfert.

Dans un premier temps, la simulation des fonctions de transfert a été étudié afin, d'une part, proposer une méthode de simulation diminuant les problèmes d'instabilité. Et d'autre part, de présenter l'initialisation en régime permanent correspondante. Cette nouvelle méthode a prouvé son intérêt dans le cas d'une fonction de transfert scalaire du domaine du domaine de Laplace dont la simulation pose problème dans EMTP.

Dans un deuxième temps, l'algorithme développé dans cette maîtrise a été décrit. Il permet d'initialiser en régime permanent des systèmes de contrôle représentés sous la forme de diagramme-bloc. Il prend en compte les éléments fonction de transfert, forme d'état, somme, gain et limiteur. À partir de la caractérisations des blocs, de la matrice d'adjacence du système et de quelques données en régime permanent, le programme construit les matrices $\mathbf{A}$, $\mathbf{H}$ et $\mathbf{E}$ nécessaire à la simulation du diagramme. Celle-ci ce fait en posant, à chaque pas de temps, un calcul de la forme $\mathbf{X}_t = \mathbf{A}^{-1} \cdot \mathbf{H} \cdot \mathbf{X}_h + \mathbf{A}^{-1} \cdot \mathbf{E} \cdot u$. L'algorithme initialise $\mathbf{X}_h$ en régime permanent en générant et résolvant un système secondaire $\mathbf{A}_i \cdot \mathbf{X}_{h,0} = \mathbf{b}_i$ simultanément aux matrices précédentes. Cela permet d'éviter de simuler un régime transitoire inutile en début de simulation lorsqu'on souhaite observer un phénomène sur un système déjà en régime permanent.

Finalement, la démarche d'initialisation a été appliquée sur un cas concret de changement de contrôleur dans un micro-réseau. Lorsque ce dernier est connecté au réseau principal, il ne nécessite que l'utilisation d'un contrôleur PI. Cependant ce dernier n'est pas suffisant pour ajuster la tension lorsque le système est isolé. Ainsi lors de la déconnexion avec le réseau principal, la commande du système est assurée par un autre contrôleur plus complexe. Sans aucune initialisation de ce dernier, la commande que reçoit le convertisseur VSC ainsi que la tension à la charge subissent une variation très importante au moment du changement. L'étude menée dans ce mémoire a montré qu'initialiser le nouveau contrôleur tel que sa sortie soit la même que celle de l'ancien au moment du changement est possible et permet d'éviter des variations trop importantes des tensions.

Pour conclure, l'objectif de ce mémoire a été complété via l'implémentation d'un algorithme

d'initialisation automatique des système de contrôle, pour un certain nombre d'éléments (fonctions de transfert, forme d'état...) toutefois. Cela améliorera le confort de l'utilisateur en lui enlevant la charge d'initialiser à la main toutes les variables. De plus, cela évitera les erreurs d'initialisation et ainsi les faux régimes transitoires en début de simulation. L'usager n'aura plus besoin d'attendre que la simulation converge vers le régime permanent avant d'étudier la réponse aux perturbations qu'il apporte, permettant un gain de temps qui peut être conséquent sur un gros système. En outre, ce mémoire a aussi proposé une façon de simuler les fonctions de transfert qui permet de résoudre des problèmes d'instabilité dans la simulation des fonctions d'ordre élevé ou avec un très faible pas de temps.

5.2 Limitations et améliorations futures

L'initialisation des fonctions de transfert n'est pas applicable au intégrateur $H(s) = \frac{1}{s}$ et, en général, à toute fonction dont le dénominateur peut être factorisé par s. En effet, en régime permanent s tend vers 0, ainsi ces fonctions divergent.

Par sa construction, l'algorithme développé ici présente quelques limitations. Il nécessite, notamment, que toutes les variables du système, hormis les variables internes des fonctions de transfert et des formes d'état, doivent avoir toute la même taille. De plus, il traite principalement des fonctions linéaires. Il reste, donc,encore à élargir la méthode aux fonctions non-linéaires pouvant être présentes dans un système de contrôle. Cela peut se faire en utilisant une matrice Jacobienne, comme cela est fait pour gérer les non-linéarités dans la simulation des systèmes de contrôle avec EMTP (voir [**?**]).

Enfin, la méthode de simulation des fonctions de transfert reste à être implémentée dans le logiciel EMTP.

ANNEXE A MATRICES DU CONTRÔLEUR MIMO

$$A_c = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-154771188 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -154771188 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -10209 & 376 & 15909 & 0 & -15909 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -376 & -10209 & 0 & 15909 & 0 & -15909 \\
90037 & 206 & 1400 & -0.11 & -480 & 1123 & -6275 & 5.24 & -226 & 376 & 221 & 1.09 \\
-1452 & 0.11 & 90037 & 206 & -1123 & -480 & -5.24 & -6275 & -376 & -226 & -1.09 & 221 \\
0 & 0 & 0 & 0 & 0 & 0 & 7875 & -8.16 & 0 & 0 & -3.14 & 376 \\
0 & 0 & 0 & 0 & 0 & 0 & -3.53 & 7875 & 0 & 0 & -376 & -3.14
\end{bmatrix}$$

$$B_c = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 10000 & 0 \\ 0 & 10000 \\ 6272.90 & 0 \\ 0 & 6272.90 \\ -7866.75 & 0 \\ 0 & -7866.75 \end{bmatrix}$$

$$C_c = \begin{bmatrix} 43218 & 99 & 672 & -0.05 & -230 & 539 & -0.04 & 2.51 & -106 & 0 & 106 & 0.52 \\ -697 & 0.05 & 43217 & 99 & -539 & -230 & -2.51 & -0.04 & 0 & -106 & -0.52 & 106 \end{bmatrix}$$