CHAPITRE 5

IMPLÉMENTATION DES MODÈLES DÉTAILLÉS

5.1 Introduction

La deuxième partie du projet de recherche consiste à réaliser la simulation en temps réel de modèles détaillés avec commande externe. Ce chapitre présente la démarche utilisée pour réaliser cette simulation. Une première partie introduit les caractéristiques de cette simulation et expose les stratégies développées pour en faire la réalisation. Les parties suivantes présentent le développement réalisé et les résultats obtenus.

5.2 Simulation en temps réel des modèles détaillés

Les modèles détaillés présentent des onduleurs triphasés possédant des ponts de six interrupteurs qui génèrent des discontinuités. D'autre part, les commandes des modèles détaillés possèdent des hystérésis de courant qui génèrent des signaux de référence pour les onduleurs sous forme d'impulsions. Puisque les impulsions sont générées à une fréquence élevée et qu'elles sont asynchrones à la simulation, elles peuvent survenir à l'intérieur d'un pas de calcul de la simulation en temps réel. La principale caractéristique de la simulation en temps réel des modèles détaillés est de faire la correction des événements se produisant à l'intérieur des pas de calcul.

La simulation en temps réel est réalisée à l'aide d'un simulateur développé par le professeur Bruno De Kelper de l'École de technologie supérieure dans le cadre de son projet de doctorat. Ce simulateur, qui utilise la méthode des commutations précises, est conçu pour corriger les événements se produisant à l'intérieur des pas de calcul de la simulation en temps réel et fonctionne dans l'environnement de MATLAB et d'xPC

Target. La principale contribution du projet de maîtrise est de faire la réalisation et l'intégration d'une interface entre le simulateur et la commande externe.

L'interface doit être intégrée au simulateur et doit traiter les signaux provenant de la commande externe. L'interface doit déterminer deux informations essentielles pour faire le traitement des impulsions des signaux de la commande, soit les instants précis où les signaux changent d'états ainsi que les états correspondants aux changements. L'interface matérielle est composée des deux cartes PCI-MIO-16E-4. Puisqu'il n'existe pas de pilote dans xPC Target permettant un tel traitement, un pilote spécifique a été développé.

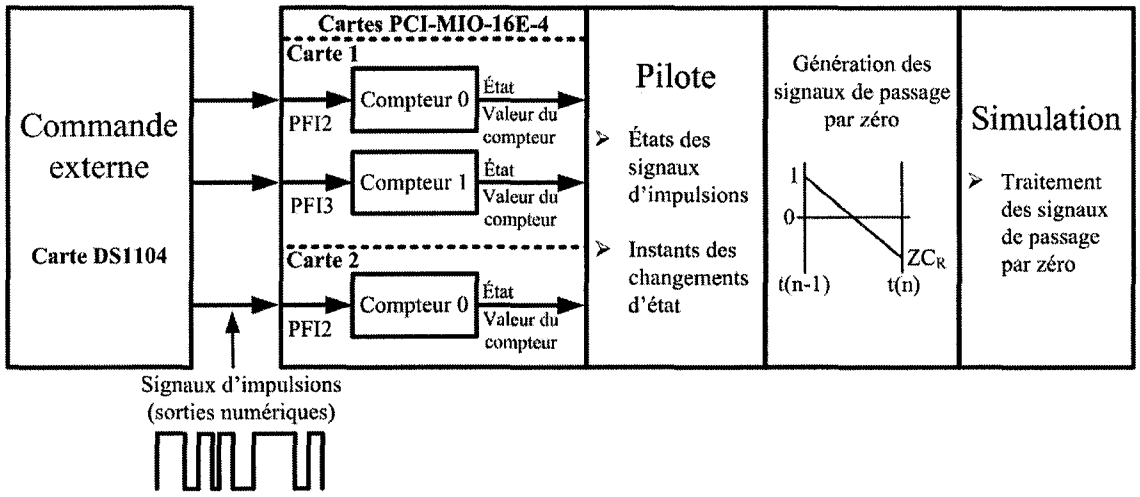

Figure 55 Schéma de principe de la simulation en temps réel avec commande externe

La pilote développé utilise les compteurs des cartes pour déterminer les informations nécessaires. Trois des six signaux d'impulsions provenant de la commande sont connectés à trois des quatre compteurs disponibles sur les cartes PCI-MIO-16E-4. Les trois autres signaux sont générés de façon logicielle puisqu'ils sont complémentaires. Grâce aux fonctionnalités des compteurs, le pilote génère les instants où les signaux changent d'états ainsi que les états correspondants aux changements. Les détails sur le développement du pilote se trouvent dans la section 5.3. La figure 55 présente le schéma

de principe de la stratégie utilisée pour la réalisation de la simulation en temps réel avec commande externe des modèles d'entraînement électrique détaillés.

Lors de l'intégration du pilote, les informations déterminées doivent être traitées pour corriger les événements se produisant à l'intérieur des pas de calculs de la simulation en temps réel. Le simulateur utilise la technologie de détection de passage par zéro de MATLAB pour détecter les discontinuités. Les informations doivent être traduites sous formes de signaux de passage par zéro pour que le simulateur puisse traiter les impulsions provenant de la commande externe comme les autres discontinuités. Des modifications sont apportées au pilote pour faire la génération de signaux de passage par zéro. Des modifications sont aussi apportées au simulateur pour faire le traitement des signaux de passage par zéro du pilote. La section 5.4 présente tous les détails concernant l'intégration du pilote dans le simulateur, la génération et le traitement des signaux de passage par zéro.

5.3 Développement du pilote

Les cartes PCI-MIO-16E-4 sont munies d'un circuit intégré spécialisé (ASIC) nommé DAQ-STC qui comporte une multitude de fonctionnalités. Dans le but d'obtenir le pilote désiré, les registres des DAQ-STC sont configurés afin d'utiliser les fonctionnalités nécessaires. Le pilote a été réalisé à l'aide des documents [1] et [2] de National Instruments sur la programmation des registres du DAQ-STC de la carte PCI-MIO-16E-4. Le pilote développé détermine les instants où les signaux changent d'états ainsi que les états correspondants aux changements. Le tableau III présente la description des registres utilisés et l'annexe 3 présente la description des bits de ces registres.

Tableau III

Description des registres[1]

Registre	Addresse (*i*=0/1)	Fonction
G*i*_Mode_Register	26/27	Ce registre permet de définir le mode de fonctionnement du compteur. Le registre défini entre autre le fonctionnement du compteur par rapport au signal G*i*_GATE.
G*i*_Command_Register	6/7	Ce registre permet de commander le compteur. Il permet entre autre d'armer et de désarmer le compteur, de faire le chargement du compte à partir des registres de chargement, ainsi que de choisir le sens du compte.
G*i*_Input_Select_Register	36/37	Ce registre détermine la source des signaux G*i*_GATE et G*i*_SOURCE. Il spécifie quelques caractéristiques du signal G*i*_GATE ainsi que la polarité des signaux G*i*_OUT et G*i*_SOURCE.
G*i*_Autoincrement_Register	68/69	Ce registre de 8 bits contient une valeur fixe ajoutée au contenu du registre de chargement A après chaque chargement du compteur pour maintenir une valeur incrémentée du compte.
Interrupt_A_Enable_Register	73	Ce registre permet de sélectionner les signaux et les conditions du groupe A qui déclenchent une demande d'interruption.

1.L'indice *i* représente l'indice d'un compteur (compteur 0 ou compteur 1). Les deux compteurs de la première carte sont utilisés et le compteur 0 de la deuxième carte est utilisé.

Tableau III (suite)

Registre	Addresse (*i*=0/1)	Fonction
Interrupt_B_Enable_Register	75	Ce registre permet de sélectionner les signaux et les conditions du groupe B qui déclenchent une demande d'interruption.
Interrupt_A_Ack_Register	2	Ce registre permet de remettre à 0 les conditions d'interruptions et de faire la reconnaissance des demandes d'interruptions du groupe A.
Interrupt_B_Ack_Register	3	Ce registre permet de remettre à 0 les conditions d'interruptions et de faire la reconnaissance des demandes d'interruptions du groupe B.
IO_Bidirection_Pin_Register	57	Ce registre permet de sélectionner la direction en entrée ou en sortie des dix broches PFI.
G*i*_Load_A_Registers	28-29/32-33	Ce registre de chargement contient une valeur modifiable qui peut être chargée dans le compteur lorsqu'il est arrêté. Ce registre comprend une partie haute de 8 bits et une partie basse de 16 bits.
Joint_Status_1_Register	27	Ce registre fourni le statut de quelques signaux et conditions. Il fourni entre autre le statut des signaux G0_GATE et G1_GATE.
G*i*_HW_Save_Registers	8-9/10-11	Ce registre sauvegarde la valeur du compte sur les fronts actifs du signal G*i*_GATE. Ce registre comprend une partie haute de 8 bits et une partie basse de 16 bits.

Le pilote conçu utilise les fonctionnalités des modules des compteurs d'usage général (GPCT) des DAQ-STC. Le GPCT consiste en deux compteurs (compteur 0 et compteur 1) indépendants de 24 bits associés à des registres de chargement et de sauvegarde. Les compteurs comportent trois signaux d'entrée, soit la source (G*i*_SOURCE), la grille (G*i*_GATE) et le contrôle de sens du compte (G*i*_UP_DOWN). Le G*i*_UP_DOWN permet de sélectionner le sens du compte en incrémentation ou en décrémentation. Dans le cadre du projet, le sens du compte des compteurs est incrémental. Le signal G*i*_SOURCE, qui représente l'horloge du compteur, est relié à la base de temps interne du DAQ-STC, soit le signal G_IN_TIMEBASE1 qui est un signal de forme carrée d'une fréquence de 20 MHz. Les compteurs incrémentent donc avec la fréquence de l'horloge qui est de 20 MHz. Finalement, le signal G*i*_GATE est relié à un signal d'impulsion provenant de la commande. Les trois signaux d'impulsion de la commande sont connectés à trois des quatre signaux G*i*_GATE des compteurs via les broches PFI (Programmable Function Inputs). Les deux compteurs de la première carte sont utilisés et le compteur 0 de la deuxième carte est utilisé.

Le pilote doit tout d'abord effectuer l'initialisation des registres des compteurs. Le tableau IV présente la configuration des registres pour l'initialisation des cartes. Le pilote doit ensuite configurer les registres des cartes pour obtenir les fonctionnalités désirées. Le tableau V présente la configuration des registres pour le pilote. Selon la configuration des registres, les signaux G*i*_GATE permettent, via des registres de sauvegarde et de statut, de déterminer les états et les instants des changements d'état des signaux d'impulsion. Les bits G*i*_Gate_St situés dans les registres Joint_Status_1_Register fournissent les états des signaux G*i*_GATE. Les instants des changements d'état sont fournis par les registres G*i*_HW_Save_Registers, qui contiennent la sauvegarde des comptes. La sauvegarde d'un compte est effectuée de façon matérielle par la carte lorsqu'il survient un changement d'état sur le signal G*i*_GATE correspondant. La sauvegarde des comptes est faite sur les fronts montants et descendants des signaux G*i*_GATE. Les bits G*i*_Gate_St et les registres G*i*_HW_Save_Registers sont lus à la fin de chaque pas de calcul de la simulation

afin de déterminer les nouveaux états et les instants de changement d'état des signaux d'impulsions.

Au début de chaque pas de calcul, les valeurs des comptes sont remises à zéro. De cette façon, les valeurs des comptes recueillies à la fin d'un pas de calcul représentent les délais entre le début du pas et les événements. Les valeurs des comptes sont des valeurs binaires d'une résolution de 24 bits qui incrémentent à la fréquence des compteurs de 20 MHz, soit une période de 50 ns. Pour obtenir les temps des événements en secondes, les valeurs des comptes sont multipliées par la période des compteurs. Cette relation est présentée par l'équation (5.1).

$$t_{\text{événement}} = \text{Valeur du compte} \cdot T_{\text{compteur}} \tag{5.1}$$

Tableau IV

Initialisation des registres[1]

Registre	Valeur (*i*=0/1)	Explication
G*i*_Mode_Register	0x0000/0x0000	Initialisation de tous les bits du registre.
G*i*_Command_Register	0x0000/0x0000	Initialisation de tous les bits du registre.
G*i*_Input_Select_Register	0x0000/0x0000	Initialisation de tous les bits du registre.
G*i*_Autoincrement_Register	0x0000/0x0000	Initialisation de tous les bits du registre.
Interrupt_A_Enable_Register	0x0000/0x0000	Initialisation de tous les bits du registre.

1.L'indice *i* représente l'indice d'un compteur (compteur 0 ou compteur 1). Les deux compteurs de la première carte sont utilisés et le compteur 0 de la deuxième carte est utilisé.

Tableau IV (suite)

Registre	Valeur (*i*=0/1)	Explication
Interrupt_B_Enable_Register	0x0000/0x0000	Initialisation de tous les bits du registre.
G*i*_Command_Register	0x0100/0x0100	Synchronisation du signal G*i*_GATE avec le signal G*i*_SOURCE.
Interrupt_A_Ack_Register	0xC060	Remise à 0 des conditions d'interruptions G0_Gate_Interrupt_St et G0_TC_St, ainsi que des conditions d'erreur G0_TC_Error_St et G0_Gate_Error_St.
Interrupt_B_Ack_Register	0xC006	Remise à 0 des conditions d'interruptions G1_Gate_Interrupt_St et G1_TC_St, ainsi que des conditions d'erreur G1_TC_Error_St et G1_Gate_Error_St.

Tableau V

Configuration des registres[1]

Registre	Valeur (*i*=0/1)	Explication
IO_Bidirection_Pin_Register	0x0000	Sélection de toutes les broches PFI en entrée.
G*i*_Load_A_Registers	0x0000-0x0000 / 0x0000-0x0000	Affectation de la valeur 0 dans le registre de chargement A.

1.L'indice *i* représente l'indice d'un compteur (compteur 0 ou compteur 1). Les deux compteurs de la première carte sont utilisés et le compteur 0 de la deuxième carte est utilisé.

Tableau V (suite)

Registre	Valeur (*i*=0/1)	Explication
G*i*_Mode_Register	0x007F/0x007F	• Les fronts montants et descendants du signal G*i*_GATE sont actifs. • Sélection du mode front. Le compteur est contrôlé par les fronts actifs du signal G*i*_GATE.
G*i*_Command_Register	0x0120/0x0120	• Synchronisation du signal G*i*_GATE avec le signal G*i*_SOURCE. • Sélection du sens du compte en incrémentation.
G*i*_Input_Select_Register	0x0180/0x0200	• Sélection d'une broche PFI comme source du signal G*i*_GATE (PFI2 pour le compteur 0 et PFI3 pour le compteur 1). • Sélection du signal G_IN_TIMEBASE1 comme source du signal G*i*_SOURCE. • Sélection du front montant pour le front actif du signal G*i*_SOURCE. • Les autres options sont mises hors fonction.
Interrupt_A_Enable_Register	0x0000	Les signaux et conditions du groupe A ne génèrent pas d'interruption.
Interrupt_B_Enable_Register	0x0000	Les signaux et conditions du groupe B ne génèrent pas d'interruption.

Tableau V (suite)

Registre	Valeur (*i*=0/1)	Explication
G*i*_Mode_Register	0x007F/0x007F	• Le registre de chargement A est toujours utilisé pour le chargement du compteur. • Le compteur n'est pas chargé lors d'une condition sur le signal G*i*_GATE. • Sélection du niveau haut pour le niveau actif du signal G*i*_GATE. • À la fin du compte, le compteur recommence à compter à partir de 0. • Il n'y a pas de désarmement matériel du compteur. • Le signal G*i*_GATE ne démarre pas et n'arrête pas le compteur.

5.4 Intégration du pilote dans le simulateur

Le pilote développé pour les compteurs des cartes PCI-MIO-16E-4 est intégré dans le simulateur avec une S-function. Une S-function décrit un bloc Simulink en langage machine et peut être écrite en plusieurs langages. Dans le projet, la S-function est écrite en langage C et doit être compilée comme un fichier MEX. Une S-function utilise une syntaxe d'appel de fonction qui permet d'interagir avec l'engin de calcul de la simulation. La S-function est intégrée dans Simulink par un bloc S-function, qui permet de faire interagir la S-function avec les autres blocs de Simulink.

L'implémentation du pilote se fait à travers les diverses fonctions de la S-function. La description des fonctions utilisées pour le pilote est présentée par le tableau VI. Comme mentionné dans le tableau, la fonction mdlStart permet de faire l'initialisation et la configuration des registres du pilote pour obtenir les fonctionnalités désirées. La fonction

mdlOutputs, qui est appelée à tous les pas de calculs de la simulation, met à jour les trois signaux de sortie de la S-function qui représentent les états des trois signaux d'impulsions provenant de la commande externe. Les signaux de sortie de la S-function sont mis à jour avec des variables nommées MODE. Chaque signal de sortie est associé à sa propre variable MODE. Ces variables sont aussi mises à jour dans la fonction mdlOutputs et alternent entre les valeurs 0 et 1 lorsqu'il y a détection de passage pas zéro sur leur signal de passage par zéro respectif. Un algorithme de détection de passage par zéro est donc implémenté dans la fonction mdlOutputs pour traiter les signaux de passage par zéro générés par la fonction mdlZeroCrossings. La figure 56 présente la séquence d'appel des fonctions durant la simulation.

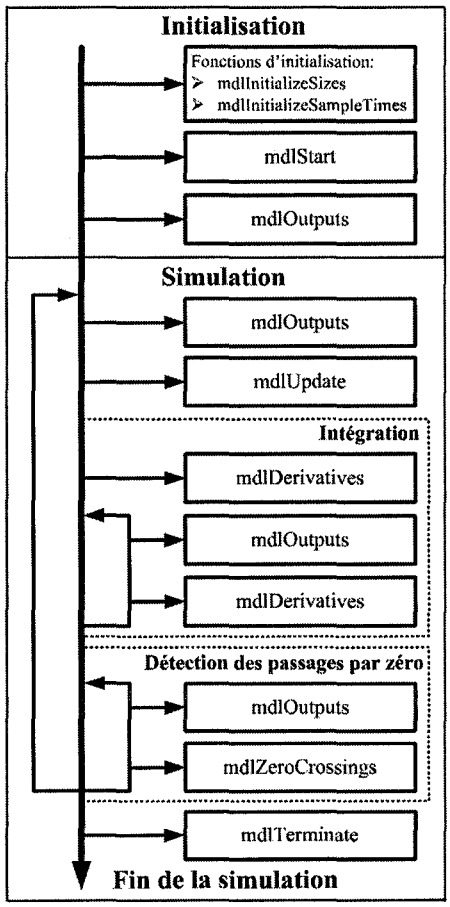

Figure 56 Séquence d'appel des fonctions de la simulation

Les signaux de passage par zéro ont aussi l'utilité de fournir les instants des changements d'état des signaux d'impulsions. Les détails sur la génération des signaux de passage par zéro sont présentés dans la section 5.4.1. Ces signaux doivent aussi être traités dans l'engin de calcul de la simulation. Les détails sur le traitement des signaux de passage par zéro sont présentés dans la section 5.4.2.

Tableau VI

Fonctions de la S-function

Fonction	Description
mdlInitializeSizes	Cette fonction spécifie le nombre d'entrées, de sorties, d'états, de paramètres et autres caractéristiques de la S-function. Elle est la première fonction appelée par Simulink.
mdlInitializeSampleTimes	Cette fonction spécifie les périodes d'échantillonnage auxquelles la S-function opère.
mdlStart	Cette fonction initialise les vecteurs d'état. Elle permet aussi de faire d'autres initialisations requises par la S-function. Pour le pilote, elle est utilisée pour faire l'initialisation et la configuration des registres des cartes PCI-MIO-16E-4, comme présenté dans les tableaux IV et V.
mdlOutputs	Cette fonction est appelée à tous les pas de calcul de la simulation par Simulink pour mettre à jour les signaux de sortie de la S-function et pour stocker les résultats dans les matrices de signaux de sortie. Pour le pilote, elle permet de mettre à jour l'information sur l'état des signaux d'impulsions provenant de la commande.

Tableau VI (suite)

Fonction	Description
mdlZeroCrossings	Cette fonction est appelé à tous les pas de calcul de la simulation par Simulink pour mettre à jour le vecteur des signaux de passage par zéro. Pour le pilote, elle permet de générer les signaux de passage par zéro associés aux changements d'état des signaux d'impulsions provenant de la commande.
mdlTerminate	Cette fonction performe les actions nécessaires à la terminaison de la simulation. Pour le pilote, elle permet d'effectuer la réinitialisation des cartes PCI-MIO-16E-4.

5.4.1 Génération des signaux de passage par zéro

La fonction mdlZeroCrossings est utilisée pour fournir à Simulink les signaux pour lesquels la détection des passages par zéro est nécessaire. Normalement, ce sont des signaux continus à l'entrée de la S-function ou des signaux générés à l'interne qui croisent le zéro lorsqu'une discontinuité survient dans la fonction mdlOutputs. Conséquemment, les signaux de passage par zéro sont utilisés pour localiser les discontinuités dans le pas de calcul selon les points où ils croisent le zéro. Pour fournir les signaux de passage par zéro à Simulink, la fonction mdlZeroCrossings met à jour le vecteur de passage par zéro ssGetNonsampleZCs(S).

Pour le pilote, les signaux de passage par zéro associés aux changements d'état des signaux d'impulsions provenant de la commande sont générés à l'interne de la fonction mdlZeroCrossings de la S-function. Les signaux générés croisent le zéro lorsqu'il y a un changement d'état sur leur signal d'impulsion respectif. Les points où il y a croisement du zéro représentent les instants dans un pas de calcul où s'est produit les changements

d'état. La figure 57 présente le schéma du principe utilisé pour la génération des signaux de passage par zéro.

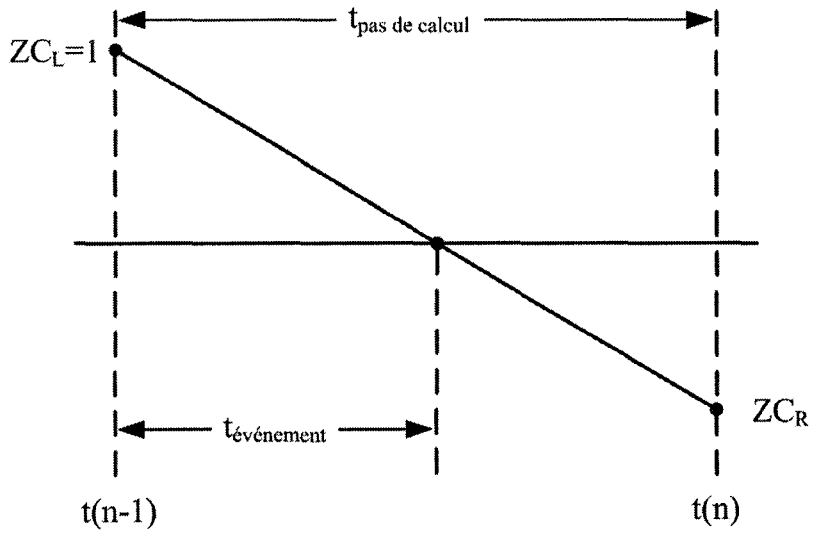

Figure 57 Signal de passage par zéro

$$K = \frac{t_{événement}}{t_{pas\ de\ calcul}} - 1 \tag{5.2}$$

$$ZC_R = \frac{K \cdot ZC_L}{K + ZC_L} \tag{5.3}$$

Selon la figure 57, dans un signal de passage par zéro, il y a deux points particuliers aux extrémités de chaque pas de calcul, soit le point gauche (ZC_L) au début du pas de calcul et le point droit (ZC_R) à la fin du pas de calcul. Ces deux points sont reliés par une droite. Au début de chaque pas de calcul, le ZC_L est placé à 1. Si un événement survient à l'intérieur d'un pas de calcul, le ZC_R est calculé pour que l'interpolation de la droite au niveau de l'abscisse représente le temps où l'événement s'est produit. Tout d'abord, la pente de la droite (K) entre le ZC_L et le ZC_R est obtenue par l'équation (5.2). Cette pente est calculée avec la proportion du pas de calcul où est survenu l'événement et utilise le

temps de l'événement déterminé par l'équation (5.1). Ensuite, le ZC_R est déterminé par l'équation (5.3), qui représente l'équation de la droite.

Dans le cas où il ne survient pas d'événement à l'intérieur d'un pas de calcul, le ZC_R est calculé pour représenter la proportion du temps restant dans le pas de calcul. Ce calcul est présenté par l'équation (5.4). Cette opération est réalisée car le temps d'exécution est normalement plus rapide que le temps du pas de calcul. Cette situation est présentée par la figure 58, qui montre un temps d'exécution plus court que le temps du pas de calcul. Le temps de l'événement utilisé par l'équation (5.4) est le compte du compteur à la fin du temps d'exécution, ce qui permet d'indiquer la proportion du temps restant dans le pas de calcul. Par exemple, selon la figure 58, le temps d'exécution correspond à 40% du temps du pas de calcul réel. Le signal de passage par zéro résultant dans le cas où il n'y a pas d'événement à l'intérieur du pas de calcul est présenté par la figure 59, où le ZC_R représente la proportion du temps restant dans le pas de calcul, soit 60%.

$$ZC_R = 1 - \frac{t_{événement}}{t_{pas\ de\ calcul}} \tag{5.4}$$

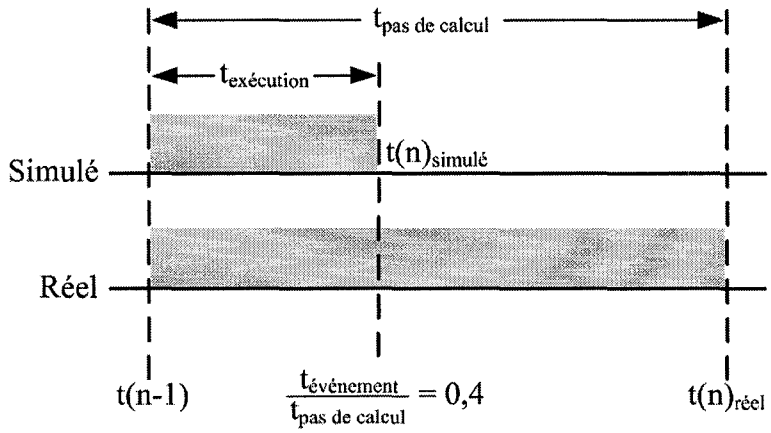

Figure 58 Pas de calcul et temps d'exécution

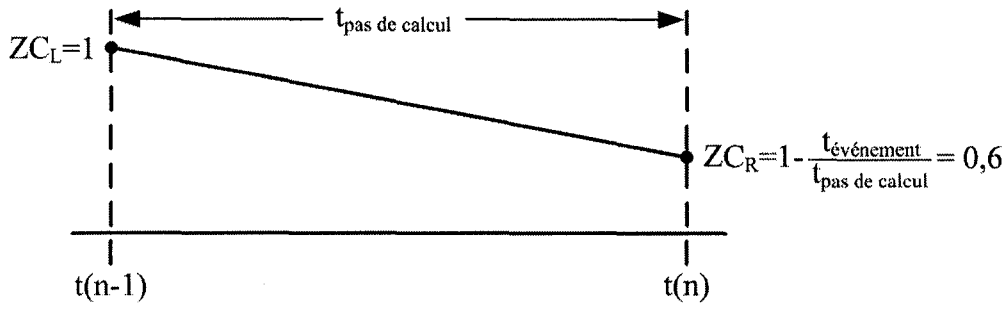

Figure 59 Signal de passage par zéro sans événement

5.4.2 Traitement des signaux de passage par zéro

Durant la simulation, l'engin de calcul de Simulink gère les appels de fonction du modèle. Le modèle Simulink appelle ensuite les fonctions des blocs Simulink et des S-functions selon la demande de l'engin de calcul. L'engin de calcul appelle tout d'abord les fonctions d'initialisation et de démarrage telles que mdlInitializeSizes, mdlInitializeSampleTimes et mdlStart. Par la suite, il démarre la boucle de simulation et fait des appels aux fonctions telles que mdlOutputs, mdlUpdate, mdlDerivatives et mdlZeroCrossings. Il termine ensuite la simulation en appelant les fonctions de terminaison mdlTerminate. La séquence d'appel des fonctions durant la simulation est présentée par la figure 56.

L'engin de calcul fait un appel aux fonctions mdlOutputs avant de démarrer la boucle de simulation pour déterminer les états initiaux des signaux. La boucle de simulation est ensuite démarrée au temps 0. La simulation avance tout d'abord jusqu'à la fin du premier pas de calcul (t_R), comme le présente la figure 60. Tous les pas de calcul de la simulation sont associés à un temps de départ (t_L) et à un temps de fin (t_R). La simulation vérifie ensuite le vecteur de passage par zéro pour déterminer si un ou plusieurs événements sont survenus à l'intérieur du pas de calcul.

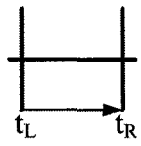

Figure 60 Avancement de la simulation à l'intérieur d'un pas de calcul

Si aucun événement n'est détecté à l'intérieur du pas de calcul, comme le présente l'exemple de la figure 61, la simulation qui a avancée à l'instant t_R ne nécessite pas de correction et débute alors un second pas de calcul. La figure 62 présente le signal de passage par zéro d'un pas de calcul pour lequel aucun événement n'a été détecté. Dans ce cas la simulation a avancé à l'instant t_R et le ZC_R correspond à la proportion du temps restant dans le pas de calcul.

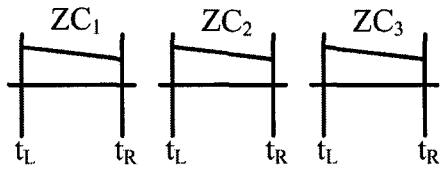

Figure 61 Aucune détection d'événement à l'intérieur d'un pas de calcul

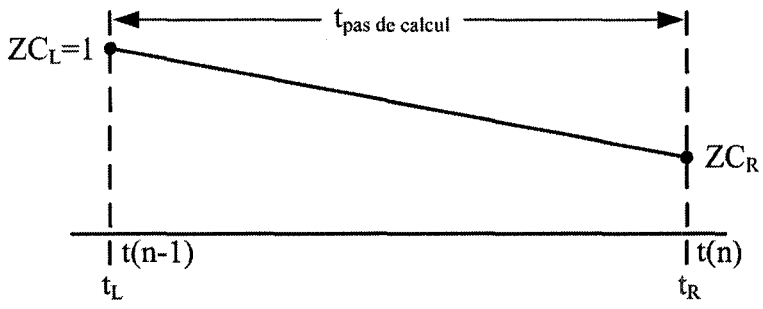

Figure 62 Signal de passage par zéro sans événement

Si un ou plusieurs événements sont détectés à l'intérieur du pas de calcul, comme le présente l'exemple de la figure 63, une logique détermine l'instant du premier événement détecté parmi tous les événements survenus. Dans l'exemple de la figure 63, le premier événement est détecté sur le signal de passage par zéro ZC_1. La figure 64 présente l'agrandissement du signal de passage par zéro ZC_1 dans lequel c'est produit le premier événement. La simulation qui avait avancé à l'instant t_R, retourne à l'instant du premier événement ($t_{événement}$), comme présenté par les figures 65 et 66. La simulation est par la suite corrigée en considérant le nouvel état du signal correspondant au premier événement.

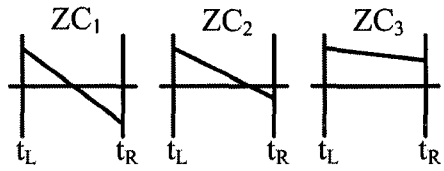

Figure 63 Détection de plusieurs événements à l'intérieur d'un pas de calcul

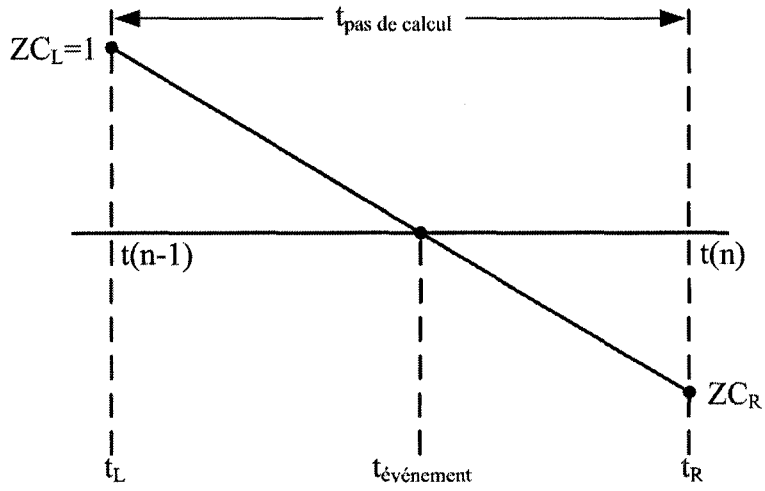

Figure 64 Signal de passage par zéro avec événement

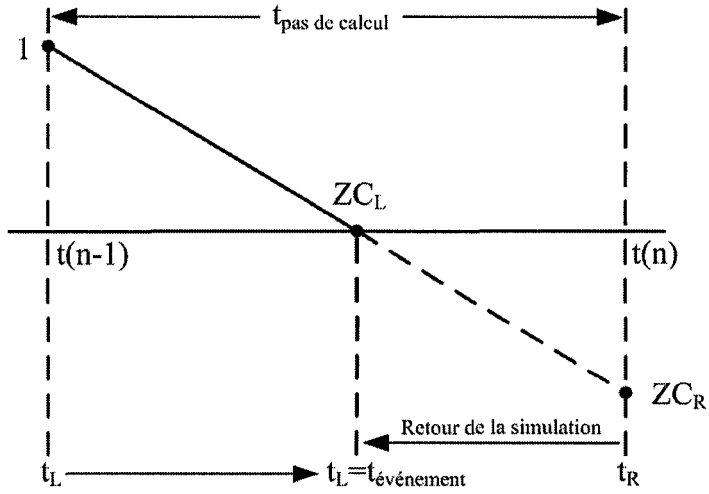

Figure 65 Retour de la simulation à l'instant $t_{événement}$

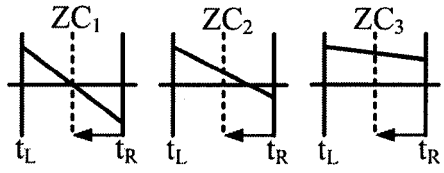

Figure 66 Retour de la simulation

Comme présenté par la figure 65, la simulation place le nouveau t_L à l'instant du premier événement ($t_{événement}$). De plus, les nouvelles valeurs de ZC_L à cet instant correspondent aux valeurs d'interpolation sur les droites où est survenu le premier événement. La même situation se produit pour les autres signaux de passage par zéro comme le présente la figure 66. Les signaux de passage par zéro associés aux signaux d'impulsions provenant de la commande nécessitent une correction avant que la simulation puisse avancer de nouveau à l'instant t_R. Cette correction est nécessaire afin que les droites entre les nouvelles valeurs des ZC_L et les prochaines valeurs des ZC_R croisent le zéro adéquatement afin de correspondre aux instants des changements d'états dans le cas où d'autres impulsions se produiraient. La correction de ces signaux de passage par zéro se

fait au nouveau t_L comme présenté par la figure 67. Pour que les droites croisent le zéro correctement, les nouvelles valeurs des ZC_L doivent correspondre à la proportion restante du pas de calcul et non aux valeurs d'interpolation sur les droites. Les nouvelles valeurs des ZC_L sont déterminées selon l'équation (5.4), qui présente le calcul utilisé pour déterminer la proportion restante du pas de calcul. Dans ce cas-ci, le temps de l'événement utilisé dans l'équation est le nouvel instant t_L. L'exemple de la figure 68 présente les trois signaux de passage par zéro associés aux signaux d'impulsions de la commande suite à la correction.

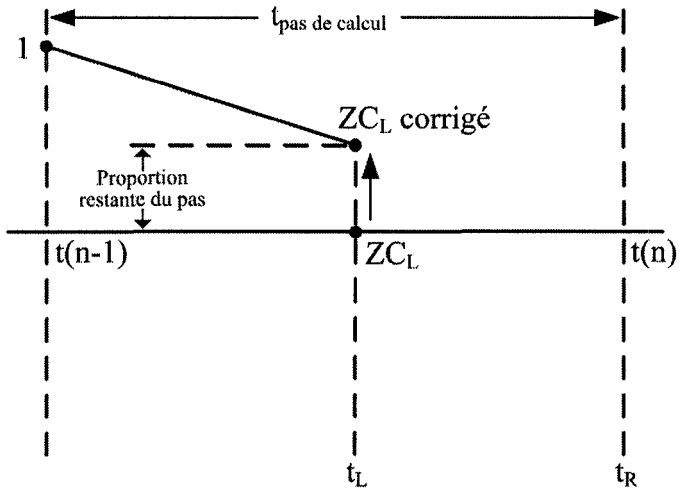

Figure 67 Correction d'un signal de passage par zéro

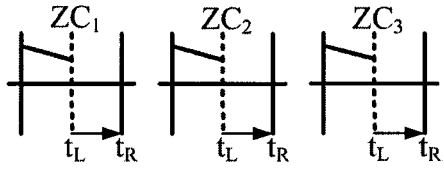

Figure 68 Correction des signaux de passage par zéro

Par la suite, la simulation avance à nouveau jusqu'à t_R et recommence le même traitement s'il y a détection d'événements. La simulation débute un second pas de calcul seulement lorsqu'elle a avancée à l'instant t_R et qu'elle ne détecte plus d'événement dans le pas de calcul. La figure 69 présente le signal de passage par zéro avec la détection d'un deuxième événement à l'intérieur du même pas de calcul. Cette figure démontre le fonctionnement adéquat de la génération des signaux de passage par zéro lorsque la correction des valeurs des ZC_L est réalisée.

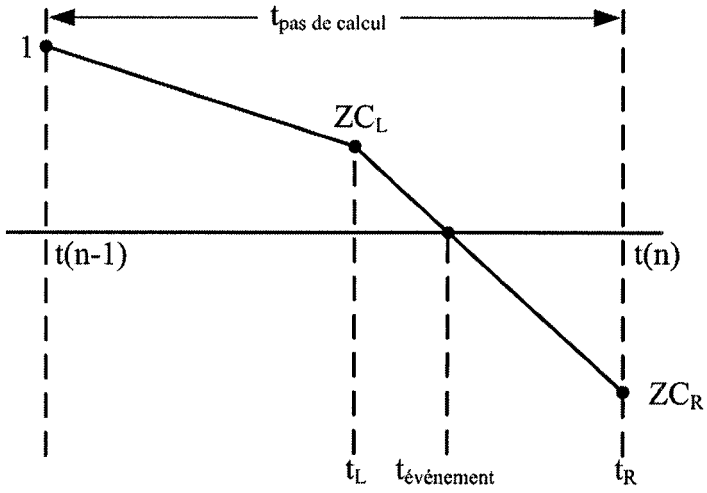

Figure 69 Signal de passage par zéro avec un deuxième événement

Le pas de calcul de la simulation en temps réel est fixe contrairement au temps d'exécution qui est variable et normalement plus court. Ceci représente la problématique majeure de la simulation en temps réel avec commande externe car les impulsions provenant de la commande peuvent survenir à l'intérieur du pas de calcul, mais après la fin du temps d'exécution. La figure 70 présente le schéma de comparaison entre le temps d'exécution et le pas de calcul pour la simulation en temps réel. Ce schéma montre le temps d'exécution sur l'axe du temps simulé et le pas de calcul sur l'axe du temps réel. S'il n'y a pas de modification apportée à l'engin de calcul, les impulsions qui surviennent dans cette période ne sont pas traitées. Afin de traiter ces impulsions, une

logique a été ajoutée dans la fonction mdlZeroCrossings de la S-function du pilote pour vérifier l'état des signaux d'impulsions au début des pas de calcul. Ceci permet de déterminer si les états des signaux d'impulsions au début d'un pas de calcul sont différents des états des signaux d'impulsions à la fin du temps d'exécution du pas de calcul précédent. En cas de différence, les changements d'état correspondants sont forcés, ce qui produit des passages par zéro au début du pas de calcul. La simulation est alors corrigée avec une erreur.

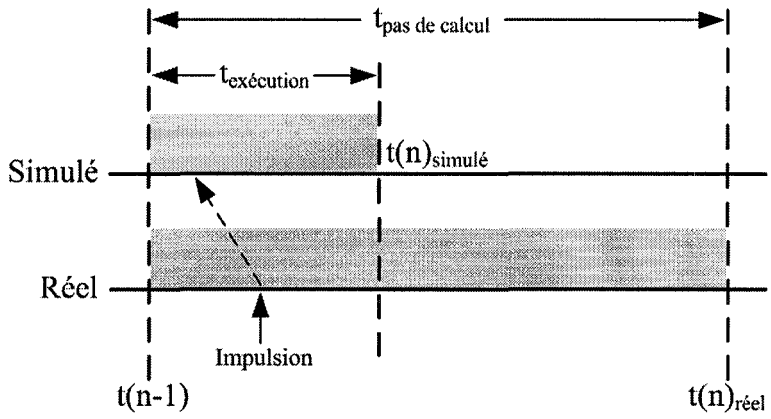

Figure 70 Correction d'une impulsion à l'intérieur du temps d'exécution

Dans la situation présentée par la figure 70, lorsqu'une impulsion survient à l'intérieur du temps d'exécution, la correction de la simulation s'effectue au temps correspondant à cette impulsion. Par contre, dans la situation présentée par la figure 71, lorsqu'une impulsion survient à l'intérieur du pas de calcul mais après la fin du temps d'exécution, la logique ajoutée permet de faire la correction de la simulation au début du nouveau pas de calcul. L'instant de la correction de la simulation correspond à la fin du temps d'exécution et non au temps correspondant à l'impulsion, ce qui introduit une erreur dans la simulation.

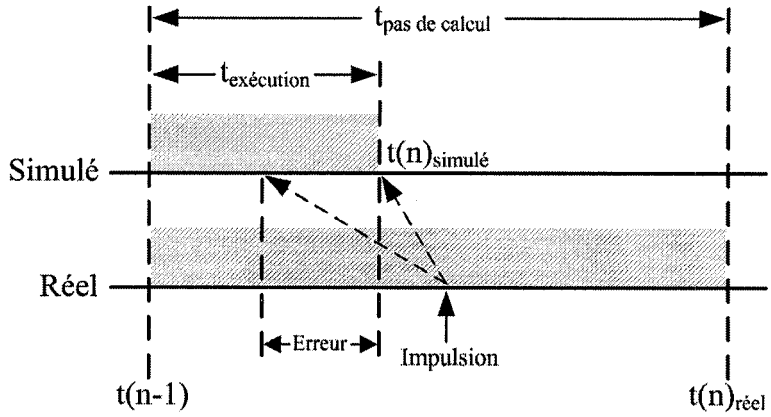

Figure 71 Correction d'une impulsion à l'extérieur du temps d'exécution

5.5 Résultats et analyse

Cette section présente les résultats obtenus pour la simulation en temps réel des modèles détaillés avec commande externe. La simulation a été réalisée à partir du modèle AC3 détaillé. Afin de comparer les résultats obtenus pour la simulation en temps réel du modèle, une simulation en temps différé a été réalisée. La partie de l'électronique de puissance et de la machine asynchrone est simulée avec la méthode des commutations précises du professeur De Kelper dans les deux cas. La figure 72 présente le schéma Simulink utilisé pour réaliser la simulation en temps différé du modèle AC3 détaillé avec commutations précises.

La commande du modèle détaillé est pratiquement identique à celle du modèle à valeur moyenne comme présentée dans la section 2.2.1. La seule différence se situe au niveau de la commande à flux orienté. La différence est que la commande fournit des signaux d'impulsions pour les interrupteurs de l'onduleur triphasé, contrairement à la commande du modèle à valeur moyenne qui fournit des références de courant aux sources de courant contrôlées de l'onduleur non standard. La figure 73 présente la commande à flux

orienté du modèle détaillé. La génération des signaux d'impulsions est réalisée avec des hystérésis de courant.

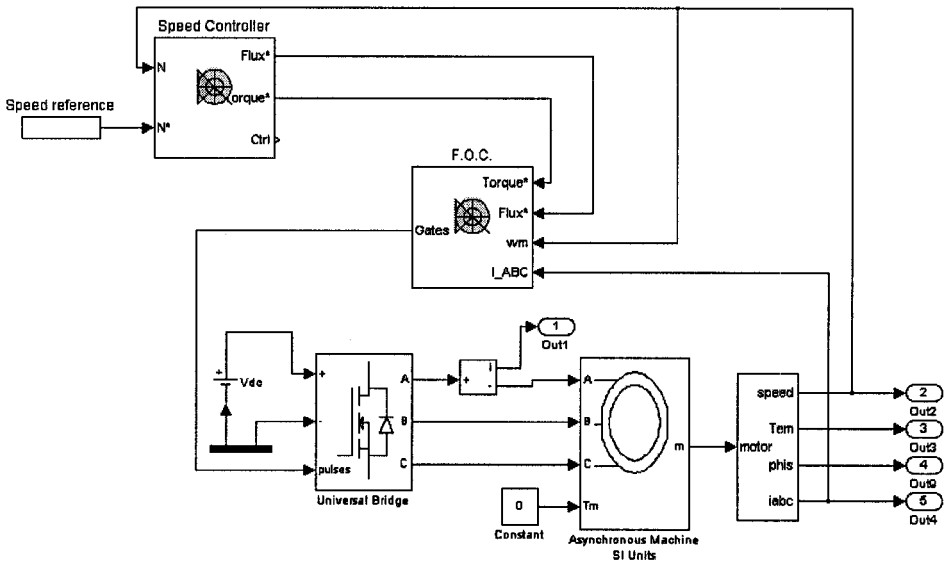

Figure 72 Schéma Simulink de simulation en temps différé

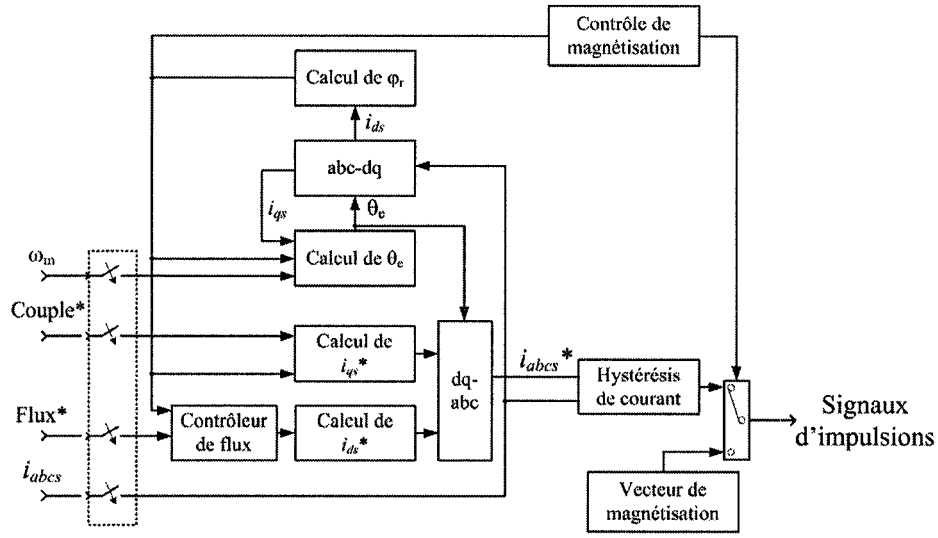

Figure 73 Commande à flux orienté

La simulation réalisée est la même pour les deux cas. La référence de vitesse est de 55 RPM et le couple mécanique est nul pour toute la durée de la simulation. La figure 74 présente les courants statoriques (i_{abcs}) obtenus pour la période de 0 s à 0,04 s de la simulation en temps différé. La figure 75 présente les signaux d'impulsions 1, 3 et 5 de la commande pour la même période de temps. Les signaux 2, 4 et 6 sont complémentaires. Les résultats montrent que les courants sont élevés durant la période de magnétisation pour faire augmenter le flux dans la machine. La fréquence des impulsions pendant cette période est relativement faible. La fréquence des impulsions augmente considérablement à partir de 0,015 s.

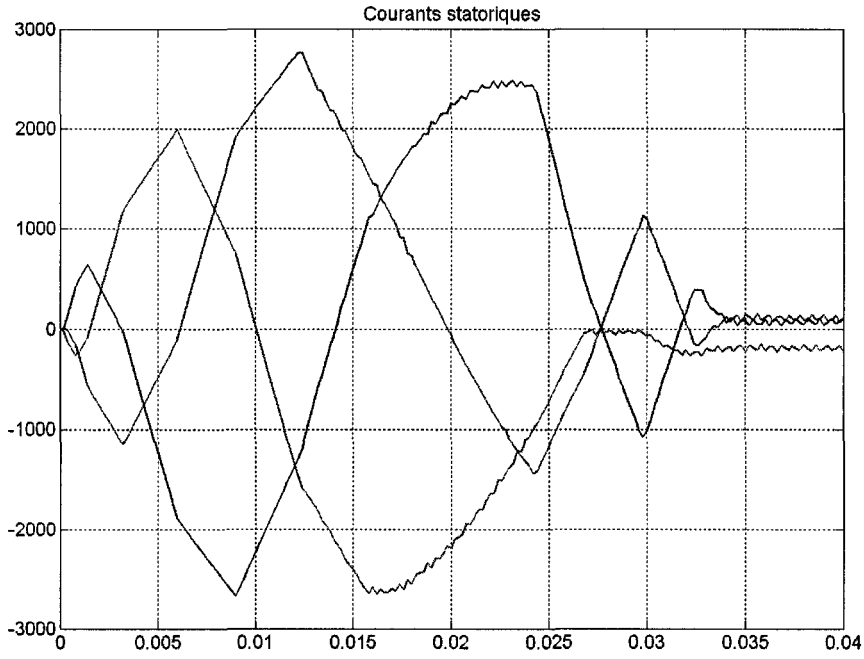

Figure 74 Résultats de simulation en temps différé

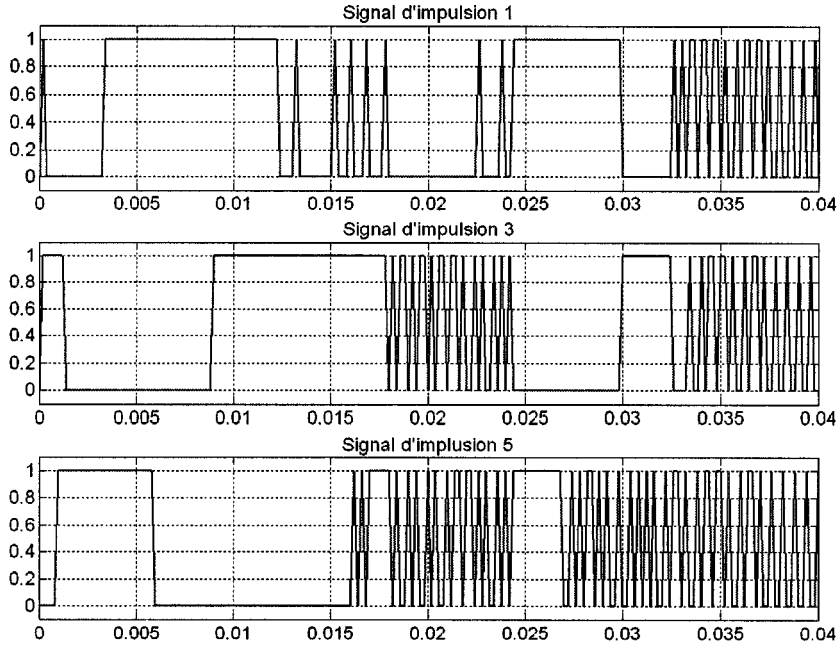

Figure 75 Signaux d'impulsions de la commande

Pour la simulation en temps réel du modèle détaillé avec commande externe, la commande est implémentée dans la carte DS1104 de la même façon que présentée dans la section 3.2. La figure 76 présente le schéma Simulink de la commande du modèle AC3 détaillé. Contrairement au modèle à valeur moyenne qui utilisent des sorties analogiques pour transmettre les références des courants statoriques, le modèle détaillé utilisent les sorties numériques pour transmettre les signaux d'impulsions. Les blocs de Simulink pour les sorties numériques de la carte DS1104 nécessitent comme information les numéros des sorties utilisées et leurs niveaux initiaux. Les sorties numériques utilisées sont les sorties 1, 2 et 3. Le signal d'impulsions 1 est transmis par la sortie 1, le signal d'impulsions 3 est transmis par la sortie 2 et le signal d'impulsions 5 est transmis par la sortie 3. Les niveaux initiaux des trois sorties sont de 0 V. Des blocs de conversion de type sont nécessaires avant les blocs de sortie numérique pour convertir le type double des signaux en type booléen. Les entrées analogiques 5 et 6 sont utilisées pour recevoir les courants statoriques i_{as} et i_{bs}. Le courant i_{cs} est obtenu selon l'équation (2.18).

L'entrée analogique 8 est utilisée pour recevoir la vitesse mécanique du rotor (ω_m). Les gains appliqués aux entrées ont été ajustés pour faire correspondre les plages de valeurs avec les plages de la simulation de l'électronique de puissance et de la machine. Ces gains ont été déterminés grâce aux plages de valeurs observées dans les résultats de la simulation en temps différé, comme expliqué dans la section 3.2.1.1.

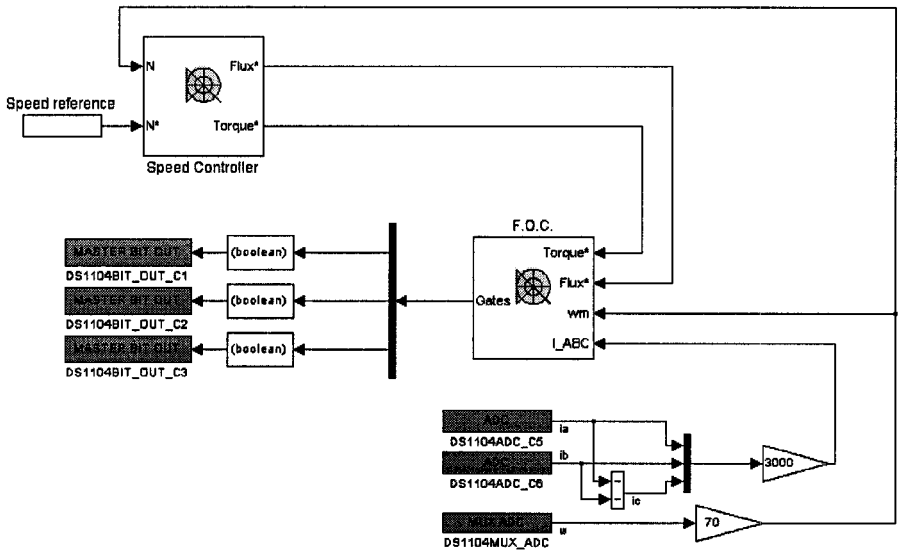

Figure 76 Schéma Simulink de la commande

La figure 77 présente le schéma Simulink de l'électronique de puissance et de la machine asynchrone. Des blocs S-function sont utilisés dans le modèle pour intégrer la S-function du pilote des cartes PCI-MIO-16E-4. La S-function est intégrée avec deux blocs. Le bloc Pilote_carte_1 est le pilote pour les deux compteurs de la première carte et le bloc Pilote_carte_2 est le pilote pour le compteur 0 de la deuxième carte. Les blocs des sorties analogiques des cartes PCI-MIO-16E-4 utilisés sont configurés de la même façon que dans la section 3.3.1.1. Les sorties de la première carte transmettent les courants statoriques i_{as} et i_{bs}. Les plages de fonctionnement de ces sorties sont de -10 V à 10 V et leurs valeurs initiales sont nulles. La première sortie de la deuxième carte transmet la vitesse mécanique du rotor (ω_m). La plage de fonctionnement de cette sortie est de 0 V à

10 V et sa valeur initiale est nulle. Les gains appliqués aux sorties ont été ajustés pour faire correspondre les plages de valeurs avec les plages de la commande externe.

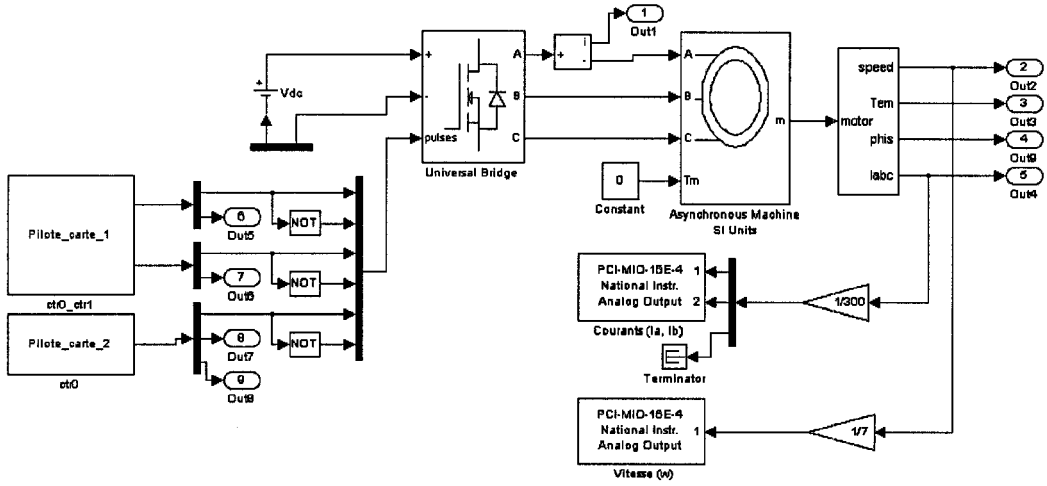

Figure 77 Schéma Simulink de l'électronique de puissance et de la machine

Comme mentionné précédemment, la simulation réalisée en temps réel avec commande externe pour le modèle AC3 détaillé est la même que celle en temps différé. Le pas de calcul utilisé pour obtenir le fonctionnement de la simulation pour la période de 0 s à 0,04 s est de 200 µs. La figure 78 présente les courants statoriques (i_{abcs}) obtenus pour cette période. La comparaison de ces résultats avec ceux de la figure 74 montrent quelques similitudes, mais démontre que le système n'est pas fonctionnel. En effet, durant la période de magnétisation, où la fréquence des impulsions est relativement faible, le comportement des courants statoriques est très similaire au comportement des courants de la simulation en temps différé. Par contre, la simulation n'est plus fonctionnelle à partir de 0,015 s, qui correspond au temps où la fréquence des impulsions augmente considérablement. Il est évident que les impulsions provenant de la commande ne sont pas toutes traitées. Par contre, le début de la simulation démontre que la correction des événements est fonctionnelle.

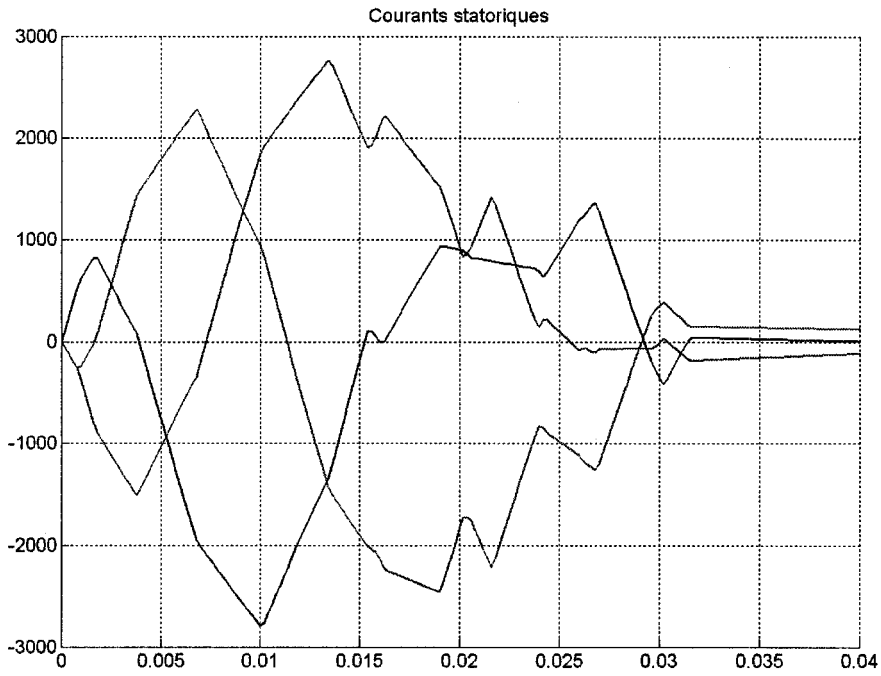

Figure 78 Résultats de simulation en temps réel

La figure 79 présente l'état des variables MODE et les signaux de passage par zéro associés aux signaux d'impulsions provenant de la commande pour la période de 0 s à 0,02 s. Les variables MODE représentent l'état des signaux d'impulsions. L'observation des variables MODE démontre que certaines impulsions ne sont pas traitées. En effet, par la comparaison des variables MODE avec les impulsions de la figure 75, il peut être observé que l'impulsion survenant au temps de 12,5 ms sur le signal d'impulsion 1 de la figure 75 n'est pas détectée par le compteur 0 de la carte 1 sur la figure 79. De plus, l'observation des signaux de passage par zéro à partir du temps de 12 ms sur la figure 79 montre que la simulation n'est plus fonctionnelle, car ces signaux sont incohérents. Grâce à la correction des impulsions, les signaux de passage par zéro ne devraient pas croiser le zéro .

La figure 79 présente aussi les TET de la simulation, qui sont multipliés par un gain de 25 000 afin d'être visualisés. La plage de variation des TET observée pour cette période est de 47 µs à 180 µs. D'après la figure 79, les TET de 47 µs sont obtenus lorsqu'il n'y a pas de traitement d'impulsion. Lors du traitement d'une impulsion, la plage de variation des TET est de 105 µs à 180 µs. Cette situation est inquiétante car le traitement d'une impulsion nécessite entre 58 µs et 133 µs. La figure 79 montre aussi que le TET moyen augmente considérablement à partir du temps de 12 ms, lorsque la simulation n'est plus fonctionnelle.

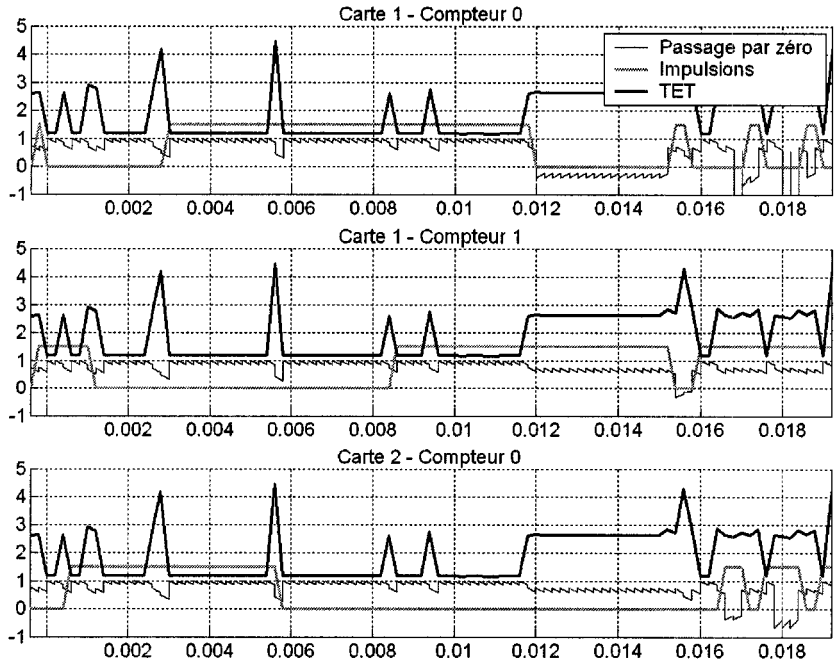

Figure 79 États des impulsions, passage par zéro et TET

Les résultats démontrent clairement que l'engin de calcul de la simulation n'est pas adapté correctement pour le traitement des signaux d'impulsions provenant de la commande. Une raison qui peut être à l'origine du problème est que l'engin de calcul exécute la boucle de simulation une seule fois par pas de calcul. Il n'est donc pas adapté

pour traiter des commutations multiples, c'est à dire lorsque plusieurs impulsions se produisent à l'intérieur d'un pas de calcul.

Par exemple, si l'engin de calcul n'a pas détecté d'événement durant le temps d'exécution, la simulation attend la fin du pas de calcul pour débuter un second pas de calcul. Comme le présente la figure 80, s'il survient plus d'une impulsion sur un même signal durant cette attente, seulement la dernière impulsion sera traitée par la logique mentionnée précedemment. Dans ce cas, la simulation est erronée car elle ne traite pas toutes les impulsions. Selon le comportement des résultats à partir du temps de 0,015 s pour une fréquence élevée des impulsions, il est très probable que cette situation soit à l'origine du problème.

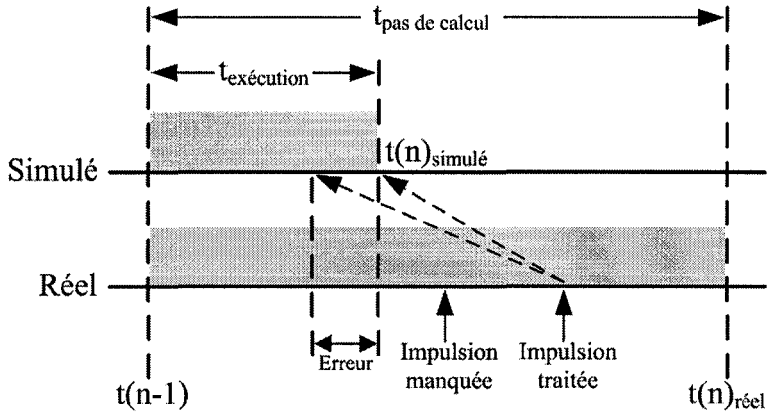

Figure 80 Impulsions multiples

Pour corriger ce problème, il faudrait modifier l'engin de calcul afin de permettre plusieurs passages dans la boucle de simulation. Cette solution permettrait de réduire le temps d'attente entre la fin du temps d'exécution et la fin du pas de calcul, comme présenté par la figure 81. La simulation pourrait ainsi détecter et traiter plus d'une impulsion dans le pas de calcul. Par contre, l'implémentation d'une telle solution n'est pas simple car l'engin devrait gérer le nombre de passages dans la boucle de simulation à

chaque pas de calcul. De plus, une autre problématique à considérer est que le temps d'exécution d'une boucle de simulation est variable, comme démontré par les TET obtenus à la figure 79.

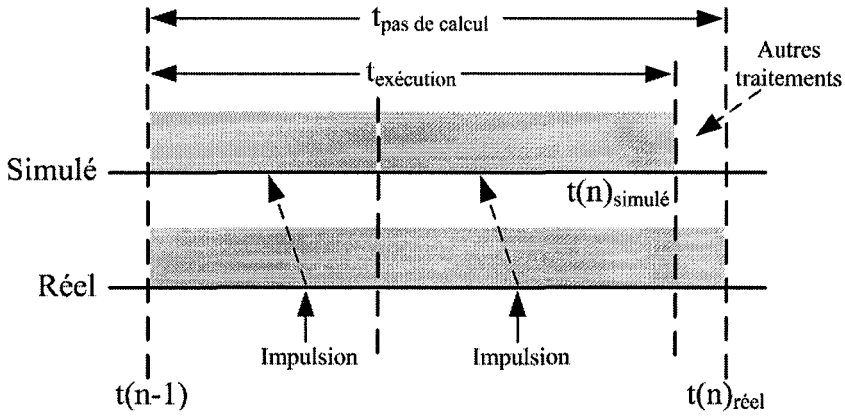

Figure 81 Réduction du temps d'attente

Comme le présente la figure 81, l'engin de calcul devrait aussi limiter la fin du temps d'exécution pour laisser un temps d'attente minimal nécessaire à la simulation pour effectuer d'autres traitements, telle que la sauvegarde des données des signaux. Dans le cas où des impulsions se produiraient dans ce temps d'attente minimal, la logique en place corrigerait la simulation avec une légère imprécision au début du pas de calcul suivant, en supposant que le risque que plusieurs impulsions surviennent sur un même signal dans cette période est très faible. Une meilleure solution serait de développer un pilote permettant de générer des interruptions pour les impulsions se produisant à l'intérieur de cette période, comme présenté par la figure 82. Cette solution assurerait le traitement de toutes les impulsions et permettrait de corriger la simulation avec plus de précision.

Le pilote par interruptions permettrait aussi de traiter les impulsions se produisant près de la fin du temps d'exécution, comme le présente la figure 82. Ces impulsions sont

problématiques car la simulation n'a pas le temps nécessaire pour les traiter avant la fin du temps d'exécution. Le pilote fournirait donc l'information nécessaire pour traiter ces impulsions avec précision au début du pas de calcul suivant.

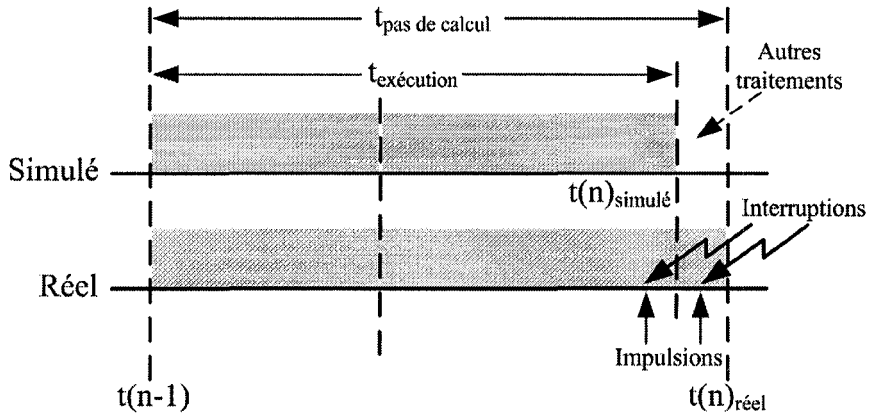

Figure 82 Traitement des impulsions par interruption

Finalement, un autre problème de la simulation est qu'elle ne traite pas les commutations simultanées, soit lorsque plusieurs événements se produisent au même instant. Par exemple, le changement d'état d'un signal d'impulsion entraîne le changement d'état d'un interrupteur dans le convertisseur, ce qui peut ensuite produire une chaîne de changements d'état des interrupteurs du convertisseur. Tous ces changements ce produisent au même instant dans la simulation. Le traitement d'une telle situation est problématique car le temps d'exécution est limité. Le traitement de plusieurs événements dans un même pas de calcul entraîne l'augmentation du temps d'exécution. La simulation devrait donc limiter le nombre d'événements à traiter dans cette situation, afin de ne pas excéder la limite du temps d'exécution.

5.6 Conclusion

Ce chapitre a présenté la démarche utilisée pour réaliser la simulation en temps réel avec commande externe des modèles d'entraînement électrique détaillés. Une première partie a introduit les caractéristiques de cette simulation et exposé les stratégies développées pour en faire la réalisation. Les parties suivantes ont présenté le développement du pilote pour les cartes PCI-MIO-16E-4, l'intégration du pilote dans le simulateur et les travaux effectués sur la génération et le traitement des signaux de passage par zéro. La dernière partie a présenté les résultats obtenus et les problématiques non résolues de la simulation en temps réel avec commande externe des modèles détaillés. Finalement, des solutions afin de pouvoir résoudre ces problématiques ont été proposées.