**Utiliser MATLAB/Simulink pour réguler l’intensité lumineuse d’une pièce**

**(Software In the Loop)**

# Introduction

## Presentation

La commande logique n’est pas adaptée pour le pilotage des systèmes dynamiques continus, ces derniers nécessitent une commande spécifique à leurs dynamiques et à celle des perturbations affectant la sortie. Cette commande générée par le correcteur est en fonction de la consigne, de la mesure effective et du temps.

Cette activité montre comment asservir les systèmes dynamiques à commandes analogiques simulés dans HOME I/O (ex : chauffage, lumière), vers une valeur de consigne grâce à un régulateur implémenté au sein du logiciel ***Matlab/Simulink*** et exécuté de manière temps réel avec HOME I/O selon uneArchitecture : ***Software in The Loop***. De plus La simulation des perturbations (contexte météorologique évolutif, trajectoire du soleil) au sein d’HOME I/O nous permet d’analyser la régulation pressentie et plus particulièrement les performances du correcteur en terme **rejet des perturbations**.

## Remarque

Le correcteur implémenté dans Simulink est exécuté de façon temps réel, de ce fait il partage la même base de temps que HOME I/O ou la vôtre. En revanche, l’accélération du temps dans HOME I/O ne sera pas perçue au sein de Simulink… La régulation présentera la même constante de temps, malgré la base de temps plus rapide dans HOME I/O.

## http://stidditec.free.fr/matlab/MATLAB-simulink.JPGIntérêts pédagogiques

Nous verrons dans cette activité comment déléguer la partie commande d’une lumière analogique à un régulateur discrétisé de type Proportionnel Intégral Dérivée (PID), exécuté au sein de ***MATLAB/Simulink***. Cette thématique de travaux regroupe de nombreux intérêts pédagogiques :

* Identifier les perturbations et leurs impacts, introduire la science de la régulation,
* Introduire la théorie des systèmes dynamiques et de l’automatique,
* Se représenter les contraintes temps réel,
* Se familiariser avec le logiciel de calcul scientifique largement répandu : ***MATLAB***,
* Comprendre la conception et le fonctionnement de systèmes embarqués destiné à opérer dans un environnement spécifique.

# Prérequis

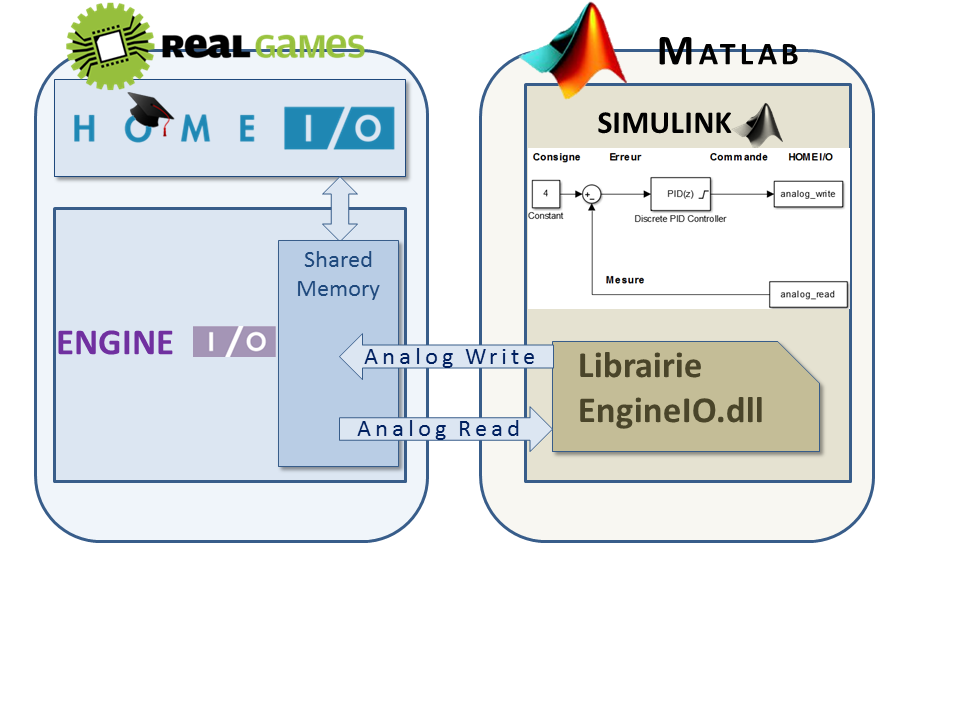
## Logiciels informatiques

La réalisation de cette activité, présuppose une version à jour et installée des logiciels suivants :

* **HOME I/O v1.2.0** : Logiciel de simulation des parties opératives d’une maison. L’application intègre 174 systèmes simulés d’une maison. En revanche dans le cadre de l’asservissement seuls les systèmes dynamiques correspondant au chauffage et à l’éclairage peuvent être utilisés avec un régulateur de type PID.
* **MATLAB 2012** ou plus récent : Logiciel de calcul scientifique, incluant Simulink : une interface de simulation des systèmes dynamiques représentée sous forme de schéma blocs. Nous utiliserons Simulink dans le cadre de l’implémentation d’un régulateur destiné à la régulation temps réel des perturbations générées par l’application HOME I/O.

# Présentation générale

## Architecture informatique



La régulation et l’asservissement de systèmes dynamiques sous HOME I/O repose sur l’architecture informatique **SIL** ci-dessus. Celle-ci fait apparaitre 2 entités exécutées au sein de la même machine hôte :

* La simulation des parties opératives, composé de HOME I/O et de la technologie ENGINE I/O permettant la connexion des capteurs et actionneurs simulés avec d’autres logiciels (tels que Simulink dans le cadre de cette activité, ou CONNECT I/O autrement),
* L’exécution temps réel du correcteur, au sein de ***MATLAB/Simulink*** grâce à la mesure émanant de ENGINE I/O et à la mise à jour de la commande - générée par le correcteur sous Simulink - dans la mémoire partagée

## Blocs de communication

Le modèle Simulink récupère les mesures issues de capteurs et met à jour les commandes destinées aux actionneurs, auprès de variables partagées avec HOME I/O via la librairie EngineIO.dll. Cette librairie chargée en mémoire par les 2 blocs « Analog\_Write » et « Analog\_Read » à l’initialisation de la simulation, permet l’écriture et la lecture d’informations à l’intérieure d’une mémoire partagée avec HOME I/O. Ces blocs appartiennent tous 2 à la famille « Level 2 Matlab S-Function » qui autorise la programmation et la configuration désirée du bloc, selon un fichier script Matlab associé, du même nom.

Lors des itérations successives de la boucle de régulation, ces blocs implémentent au sein de leurs scripts Matlab, une méthode spécifique à la lecture ou à l’écriture disponible dans la librairie « EngineIO.dll ».

Dans notre cas, on lit l’information capteur au sein de la fonction output du bloc « AnalogRead » grâce à la commande :

*GetFloat(0, EngineIO.MemoryType.Input)*

De même, on écrit la commande actionneur au sein de la fonction output du bloc « AnalogWrite » grâce à la commande :

*GetFloat(0, EngineIO.MemoryType.Output)*

Le premier argument de cette fonction est l’adresse de la case mémoire partagée, dans laquelle l’action d’écriture ou de lecture doit être initiée. Le second argument est l’orientation du flux, c’est-à-dire qu’il détermine si l’action est une lecture ou une écriture.

Par défaut, les scripts associés à ces blocs réfèrent à la commande et au capteur de lumière analogique de la zone A, ainsi qu’au capteur de lumière extérieure. En revanche, il est possible de changer l’adressage de la fonction «GetFloat» pour accéder à d’autres entrées ou sorties de systèmes dynamiques (ex : chauffage, lumière des autres pièces,…). Referez-vous à la feuille de calcul : « Memory Addresses » de la documentation du SDK d’HOME I/O pour la correspondance des dispositifs avec leurs adresses mémoire respectives.

## Correcteur

Les correcteurs adaptés pour le contrôle des systèmes simulés dans HOME I/O sont de type PID. Ces régulateurs sont largement répandus du fait de leur simplicité de mise en œuvre et de leurs performances satisfaisantes dans une majorité d’applications.

En plus de mener la sortie du système vers valeur de consigne en régime permanent - avec un temps de réponse satisfaisant – le correcteur doit être en mesure de rejeter les perturbations, c’est-à-dire d’amener le système vers une erreur statique nulle en régime permanent.

## Contraintes temps réel

Dans la mesure où le correcteur dispose d’une certaine dynamique adaptée au système qu’il commande, celui-ci doit partager la même base de temps du système simulé avec lequel il est couplé. C’est pourquoi l’exécution de modèle Simulink doit respecter la contrainte d’exécution temps réel.

Lorsque HOME I/O est en mode de temps réel (c’est-à-dire temps non accéléré), ce dernier partage la même base de temps que Simulink. Cette contrainte imposée à Simulink est mené grâce à un bloc S-Function « Parser Real Time » qui limite la vitesse d’exécution pour que celle-ci puisse être observée comme étant synchronisée avec le temps réel.

# Exemple applicatif

## Présentation

### Introduction

L’objectif de cet exercice est de concevoir un correcteur capable de générer une commande analogique adaptée au pilotage de l’intensité lumineuse d’une pièce (**suivie de consigne**) et à la compensation des modifications de rayonnement lumineux induit par l’environnement de la maison (**rejet de perturbation**).



### Spécifications

Le correcteur à l’étude dans cet exercice doit répondre aux contraintes de fonctionnement du système avec lequel il est couplé. Les contraintes prises en compte dans la conception sont les suivantes :

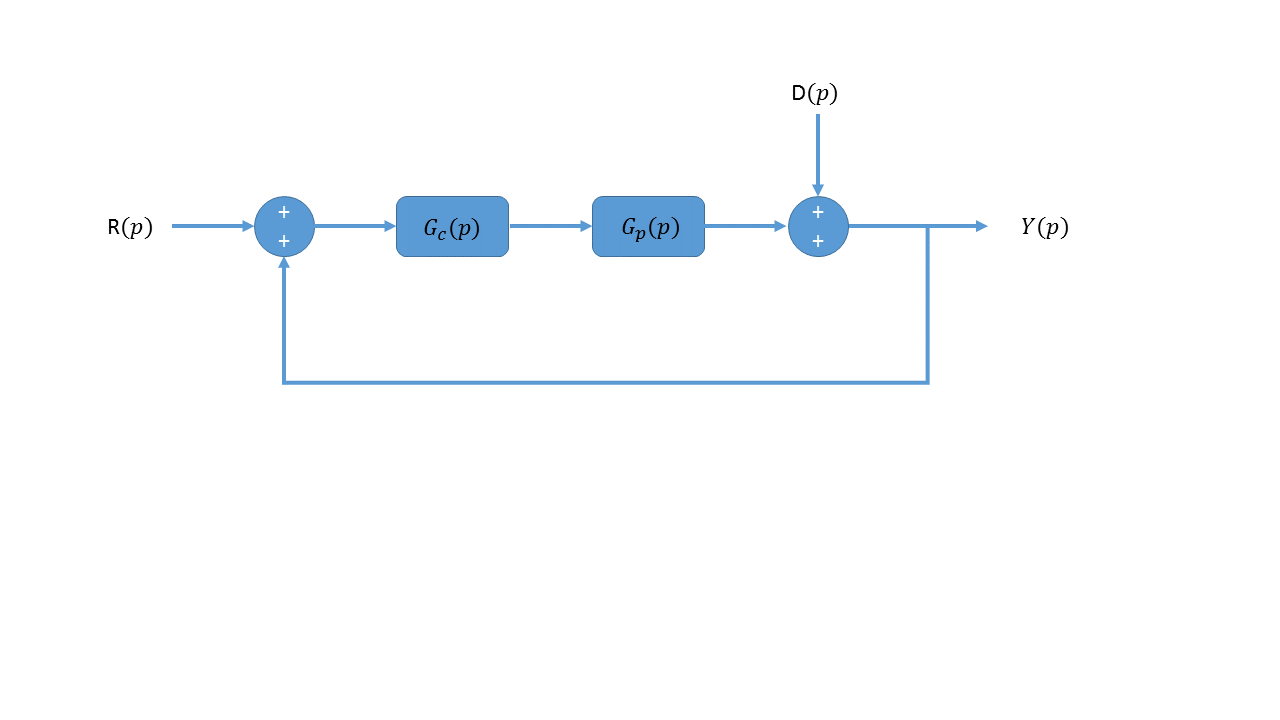
* La fréquence d’échantillonnage relatif à HOME I/O est de : ,
* Le retard due à la communication entre HOME I/O et Simulink est de : = 0,2s,
* Le modèle mathématique de la lumière, transformant une commande de type tension (en Volts) en une intensité lumineuse est un simple gain de ½. Son fonctionnement n’introduit pas de dynamique, la transformation est considérée instantanée. La fonction de transfert de l’actionneur est alors : ,
* Le capteur de luminosité ne présente lui aussi aucune dynamique et son gain est de 1. Le retour est donc unitaire,
* Le correcteur doit fonctionner de manière temps réel pour partager la même base de temps que HOME I/O et être synchronisé avec sa dynamique,
* Les erreurs de mesures et le bruit sont négligeables dans la simulation, ces phénomènes ne seront pas pris en compte dans la conception du correcteur.

En plus d’intégrer les contraintes ci-dessus, le régulateur doit présenter des performances suffisantes pour respecter les exigences requises par l’exercice :

* Un temps de réponse optimal (en mode accéléré sous HOME I/O les perturbations surviennent très rapidement). En revanche, ce temps de réponse est limité par la fréquence d’échantillonnage de HOME I/O d’après la règle suivante :
* Une robustesse, face à des retards qui subsistent de manière variables malgré qu’ils soient centrés sur 0,2s
* Une erreur statique nulle en régime permanent, en vue d’obtenir une commande précise
* Un rejet de perturbation satisfaisant menant à une compensation rapide de celle-ci

## Conception

### Modèle en boucle fermée



Avec : R(p) = Dynamique de la consigne

D(p) = Dynamique des perturbations

Y(p) = Dynamique de l’intensité mesurée

### Théories

En appliquant le théorème de superposition, il est facile de montrer la contribution de la consigne (R(p)) et des perturbations (D(p)) sur la sortie, et donc la mesure :

D’après l’équation ci-dessus on montre que

* La sensibilité du correcteur vis-à-vis des perturbations est moindre lorsque le terme : diminue, c’est-à-dire que le gain de retour de la boucle :augmente
* Le suivi de trajectoire vis-à-vis de la consigne est meilleur lorsque le terme augmente.

### Choix du Correcteur

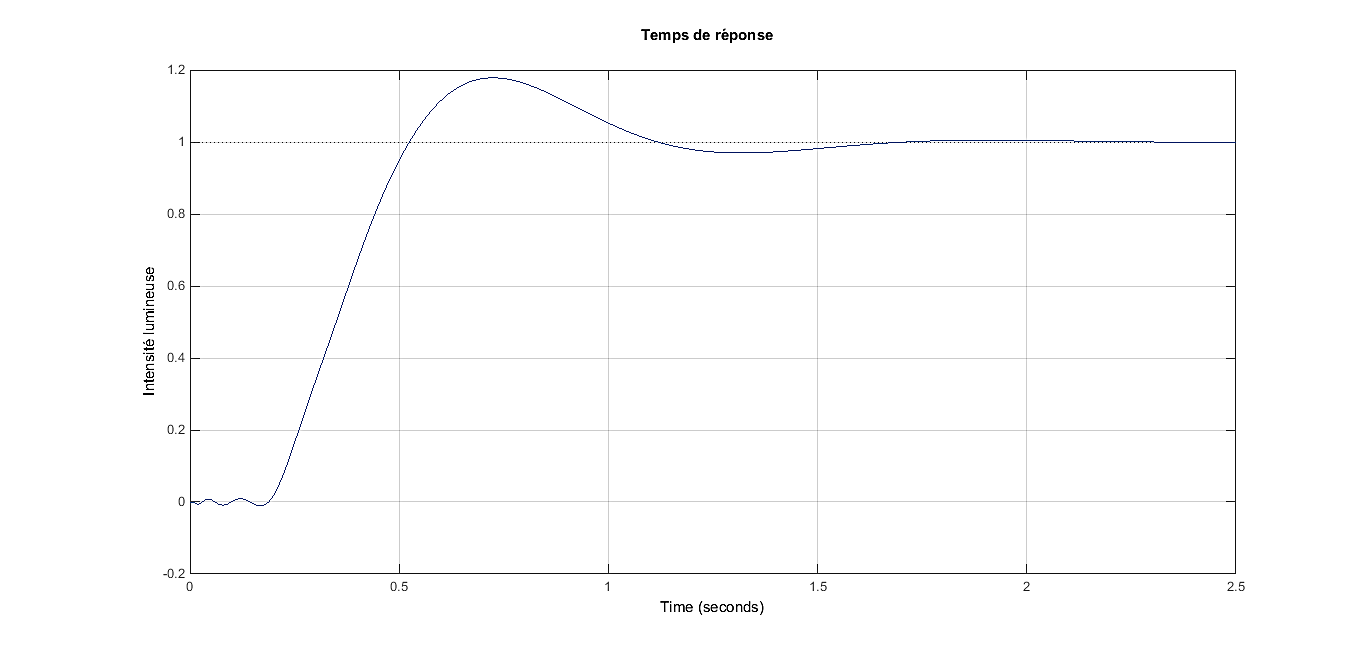
Nous utilisons pour l’asservissement un régulateur de type PID, dont 3 paramètres de réglages relatifs à l’action Proportionnel, Intégrale et dérivée sont disponibles. En revanche le système à contrôler ne possède pas de constante de temps ni d’intégrateur pur. De ce fait l’utilisation de l’action dérivée au sein du correcteur, rapprocherait le système de l’instabilité et diminuerait sa robustesse. Nous n’utiliserons pas son action dans le correcteur.

D’après la fréquence d’échantillonnage des données de HOME I/O, on a choisi une constante de temps optimale pour le réglage des paramètres du correcteur. En revanche, dans ce cas, la constante de temps choisit du système en boucle fermé est égale à la durée du retard qui subsiste dans la communication des données (). Il est donc nécessaire de choisir une constante de temps plus longue que le retard (nécessaire pour satisfaire la robustesse du correcteur). De plus, l’action proportionnelle (impliquant une transformation instantanée de l’erreur en commande) est inadaptée au control des systèmes à retard pure (sauf si celui-ci est compensé par un ***prédicteur de Smith*** par exemple). On choisit par exemple assurant robustesse (face aux retards) et rapidité (en régime transitoire). On détermine le gain de l’action intégrale par placement de pole :

## Validation

### Rapidité

On trace la réponse indicielle du système formé par le correcteur et l’actionneur en linéarisant préalablement le retard pure grâce à une fonction « Pade » d’ordre 6. On constate que la constante de temps du système sans le retard respecte bien celle calculée en phase de conception.



s

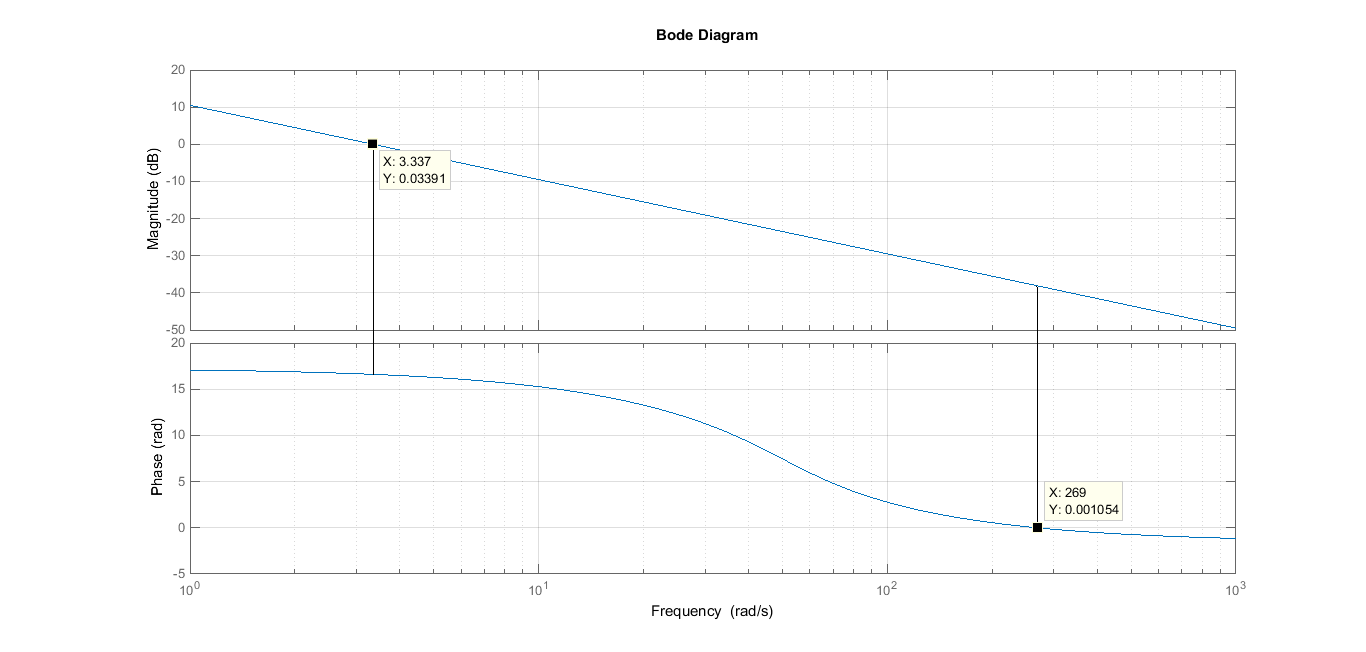
### Robustesse

Un système est considéré robuste s’il présente les caractéristiques suivantes :

* Marge de Gain > 15db (correspond au gain supplémentaire maximum que l'on peut donner au système en BO sans risquer de le rendre instable en BF)
* Marge de Phase > (permet de déterminer le retard maximal sans déstabiliser le système en BF)

D’après le diagramme de bode ci-dessous, on remarque que le système est robuste et satisfait le cahier des charges :

* La marge de gain, Mg = -38db ,
* La marge de phase, Mp = 16+3,14 = 19,14rad,
* La marge de retard, Mr = Mg/3,37 = 5s,



-38 db

16 rad

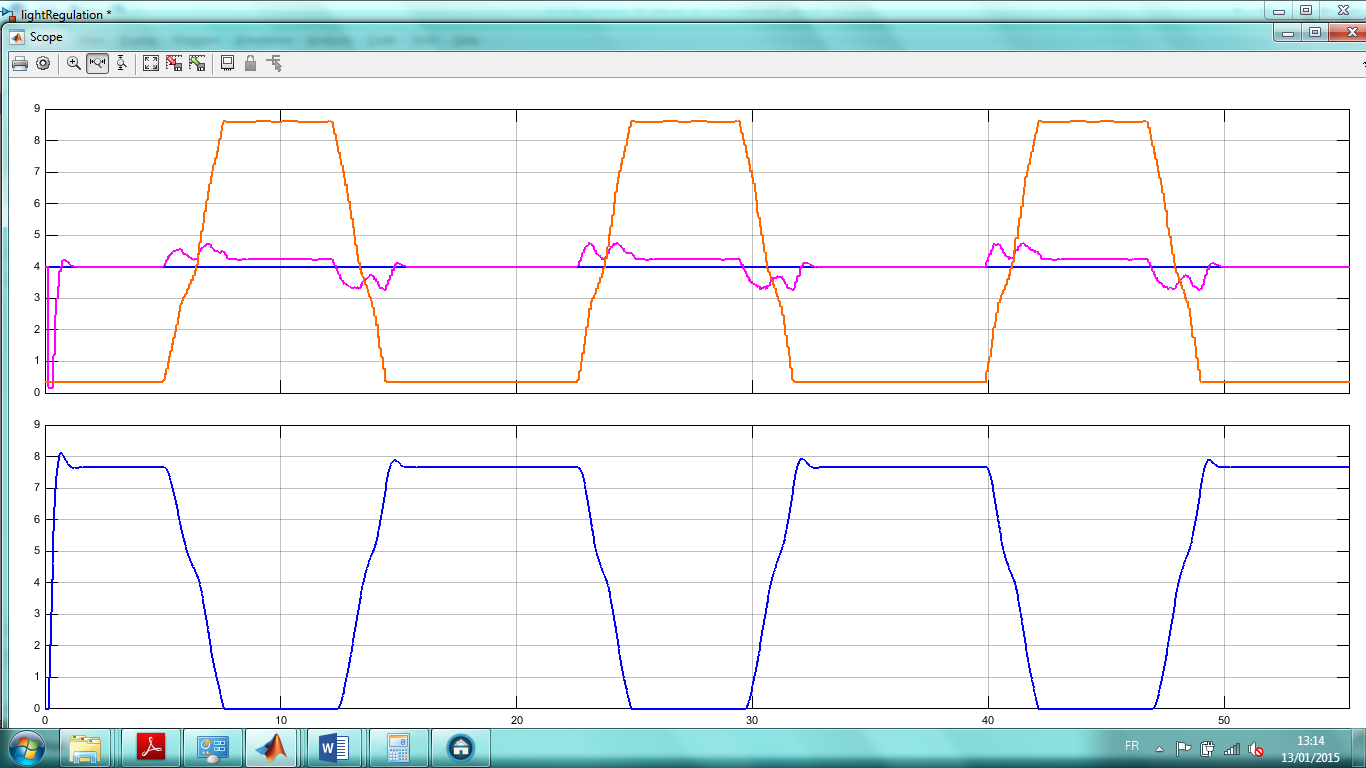
## Tests

### Scénario

La conception du correcteur est conforme aux exigences définies dans cet exercice. La phase de test permet de s’assurer que le correcteur marche correctement dans l’environnement pour lequel il a été conçu. Pour ce faire, on mène un scénario d’application au sein de HOME I/O :

* Consigne d’intensité lumineuse à 4, pour que l’actionneur soit en mesure de l’atteindre.
* lumière analogique du salon en mode externe pour recevoir la commande issue du correcteur (Actionneur de la boucle de régulation)
* Volets roulant ouverts (Maximiser les perturbations extérieurs)
* Informations d’intensité lumineuse intérieure en mode externe, situé dans le salon (Capteur intervenant dans la boucle de régulation)
* Information d’intensité lumineuse extérieure en mode externe, situe près de la porte de garage (Visualisation de perturbations extérieures)
* Test sur 3 jours complets pour l’alternance du jour et de la nuit
* Temps accéléré au maximum pour simuler 3 jours en moins d’une minute (cas le plus défavorable pour le correcteur vis-à-vis de la dynamique des perturbations à compenser)

### Résultats



*Commande*

*Consigne*

*Mesure intérieure*

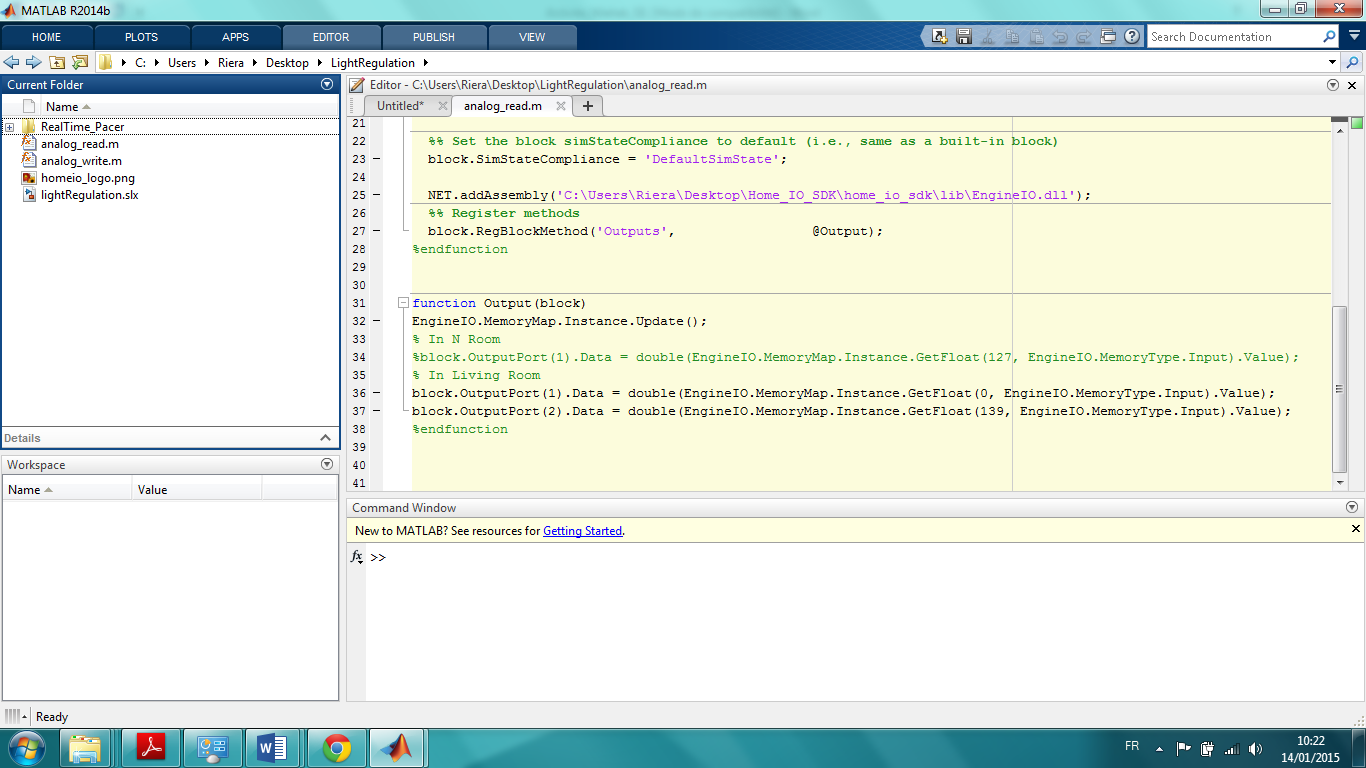
*Mesure extérieure*

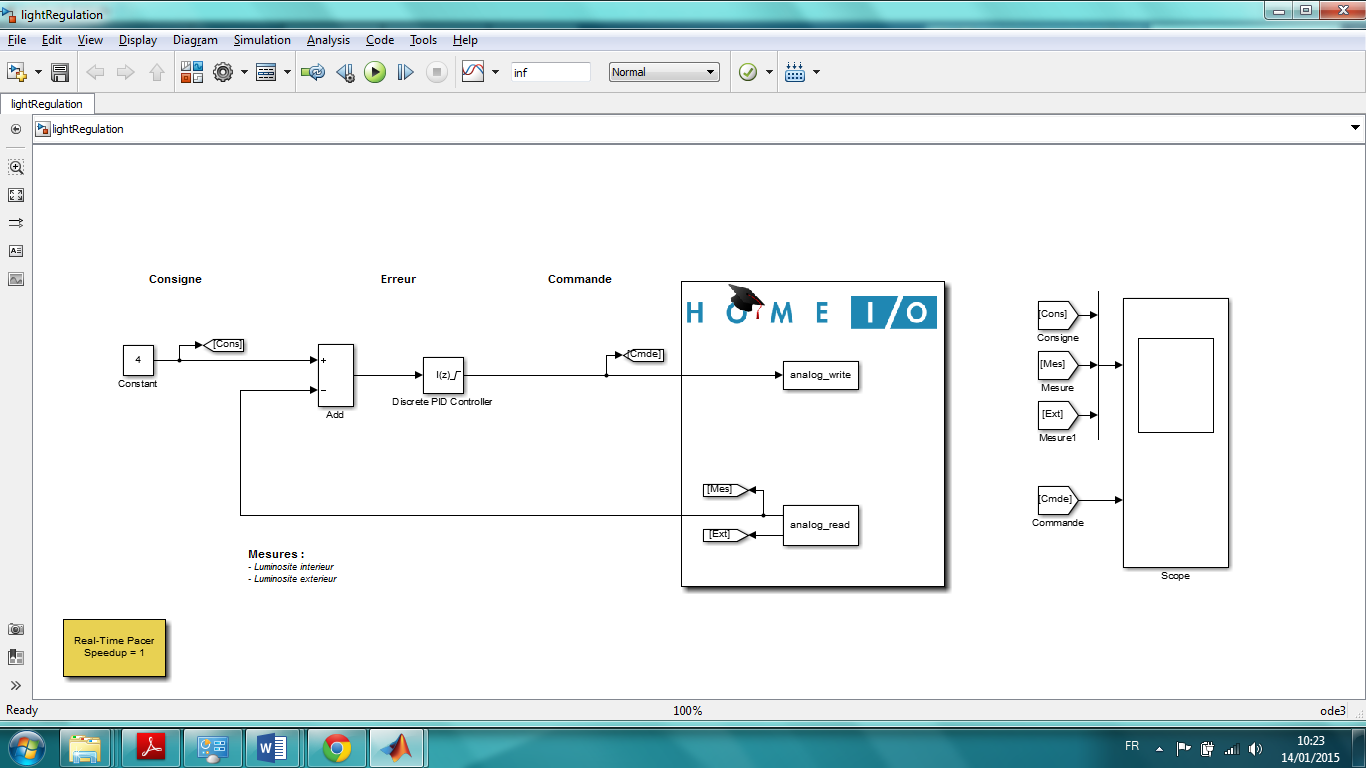
D’après les courbes tirées de la simulation sous Matlab on remarque :

* La précision en régime permanent (erreur statique nulle),
* L’effet des perturbations sur la mesure et la compensation de celle-ci,
* La saturation inférieure de l’actionneur lumière (commande négative impossible, une lumière ne génère pas d’obscurité)

### Mise en œuvre

1. Lancer HOME I/O et charger la sauvegarde : « Light Regulation\_2015\_1\_7\_16\_43\_51 »
2. Lancer MATLAB et se placer dans le dossier « LightRegulation » afin de le définir comme emplacement de travail
3. Cliquer sur « Real Time Pacer » >> « Add to Path » >> « Selected Folders and Subfolders » afin d’ajouter la librairie d’exécution temps réel à l’environnement de travail.



1. Cliquer sur « Run »  pour compiler et exécutera le modèle Simulink