Système de contrôle d'attitude et d'orbite :

Projet 3A

2014/2015

LAAMARTI Ouaïs.

BENGMAIH Ilyass.

3A Spécialité Architecture des systèmes physiques.

EIDD Paris.

Encadré par :

INCHAUSPE Henri

Remerciements:

Nous souhaitons tout d'abord remercier chaleureusement M. Henri INCHAUSPE pour son aide et ses explications ainsi que son accompagnement tout au long de notre projet.

Nous souhaitons ensuite remercier M.Hubert HALLOIN pour sa présence et son encadrement qui nous ont été d'une aide précieuse et qui nous ont permis d'avancer dans le bon sens dans notre travail..

Nous aimerions par la même occasion remercier, nos camarades qui avaient travaillé sur le même sujet l'année précédente et qui se sont montré très disponible pour nous aider à mieux cerner leur travail.

Pour finir, un remerciement spécial pour M. Gérard ROUSSET, notre directeur des études qui nous a encouragés lors du choix du sujet. Nous le remercions aussi pour son suivi et ses conseils très instructifs qui nous ont aidés à mener à bien le travail qui nous a été confié.

Sommaire :

Résumé du document

Le système de contrôle d'attitude et d'orbite, est un ensemble d'équipements et de logiciels de bord qui assure la fonction de pilotage d'un engin spatial en vue de lui imposer l'attitude voulue et d'ajuster son orbite aux exigences de la mission. Le système de contrôle d'attitude est composé de plusieurs capteurs (pour déterminer sa position), actionneurs (pour modifier l'orientation) et d'un logiciel.

Ce document présente deux grandes parties de l'étude sur ce système. Une première partie qui décrit le fonctionnement du système (Schéma Blocs), et une deuxième partie qui détaille le contenu de ces blocs.

Dans la première partie, il est surtout question de modélisation du système afin de mieux cerner son fonctionnement et comprendre les objectifs de l'étude.

Dans la deuxième partie, on s'est intéressé au contenu des blocs notamment celui où l'on modélise la dynamique du satellite, pour pouvoir ensuite tester des lois de contrôle simples et voir la réponse du système à ces lois C'est la partie qui nous permettra de qualifier objectivement l'aptitude du satellite à suivre son orbite et effectuer les missions souhaitées.

Introduction générale

Dans le cadre de nos études, nous avons eu l'opportunité d'effectuer notre projet de 3A sur le projet IGOSAT qui est un projet de lancement d'un Nano satellite au sein de l'université en collaboration avec le CNES (Centre Nationale d'études Spatiales) et le laboratoire APC (AstroParticules et Cosmologie).

Notre mission a consisté à prendre part à ce projet important pour l'école, qui portait sur l'étude du système de contrôle d'attitude et d'orbite (SCAO). Notre travail consistait dans un premier temps à comprendre ce qui a déjà été fait par nos camarades d'un côté et appréhender le fonctionnement du système de l'autre. La deuxième partie nous avons simulé le système en utilisant le logiciel MATLAB (mis à notre disposition par l'école).

Durant cette période nous avons mis en pratique nos connaissances du logiciel MATLAB, et développé nos compétences en modélisation numérique et en mécanique spatiale. Ce projet nous a aussi permis d'acquérir des connaissances générales sur le matériel utilisé dans les engins spatiaux et leur fonctionnement, tels que les actionneurs et capteurs

Ce document donne un aperçu détaillé sur le contexte du projet et sur la problématique d'un côté, et sur le travail effectué et les résultats obtenus de l'autre.

Présentation du projet :

IGOSAT :

Le projet IGOSAT est un projet à but scientifique et pédagogique, qui a pour objectif d'envoyer un Nanosatellite dans un horizon relativement proche (2017/2018). Le gros du travail est fait par des étudiants de l'université Paris. Ces derniers, encadrés par une équipe d'enseignant qui supervisent, orientent et valident les travaux effectués par les étudiants, ont la possibilité d'intervenir sur plusieurs aspect de l'étude selon leurs préférences à savoir :

- ✓ SCAO (Système de contrôle d'attitude et d'orbite)
- ✓ Télécommunication (Station Bord et Station Sol)
- ✓ Mécanique/Thermique.

Parties prenantes :

Plusieurs personnes et organismes prennent part à ce projet, parmi lesquelles nous pouvons citer :

- ✓ CNES (Centre National d'études Spatiales)
- ✓ Laboratoire APC (Laboratoire AstroParticules et Cosmologie)
- ✓ Etudiants de l'université Paris 7
- ✓ Etudiants de l'Ecole d'ingénieur Denis Diderot (EIDD)

Chacune de ces parties apporte une part de contribution au projet selon son positionnement et son domaine d'expertise, pour pouvoir à la fin arriver à bout du projet.

Présentation de la mission:

Contexte :

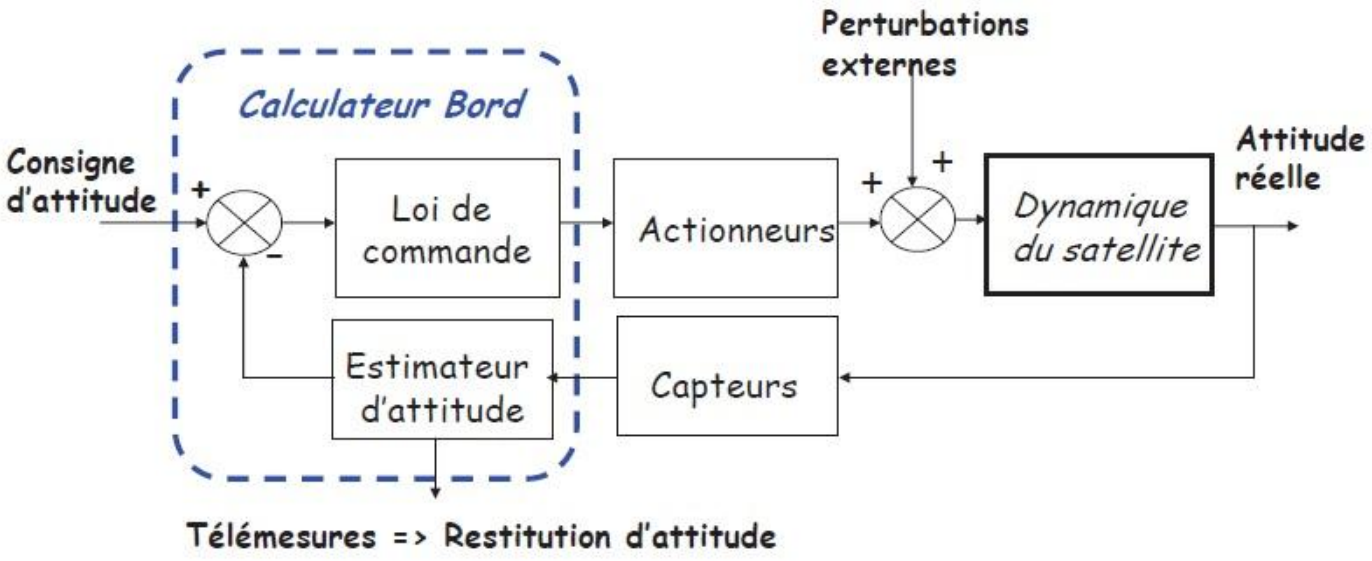
Dans le cadre de notre projet de 3ème Année, mon camarade et moi avions le choix entre plusieurs projets dont nous pouvons citer :

- ✓ Télécommunications sol
- ✓ SCAO
- ✓ Kit Arduino
- ✓ …

Ces projets étaient tous aussi intéressants les uns que les autres, ils faisaient intervenir chacun des aspects différents de notre spécialité. Le choix dépendait beaucoup des préférences de chacun de nous et devait se faire tout en pensant à projeter les travaux qui seront réalisés dans notre projet professionnel afin de s'y investir et en profiter comme il se doit.

SCAO (Système de contrôle d'attitude et d'orbite) :

La SCAO est l'ensemble d'équipements et de logiciels qui permettent le pilotage du nanosatellite, le but étant de lui imposer une attitude voulue et d'ajuster son orbite selon les besoins et les contraintes de la mission. Cela se fait en utilisant une boucle d'asservissement.



Le schéma bloc ci-dessus représente l'architecture SCAO.

Intérêts :

Après de nombreuses discussions avec les proposants et nos responsables de formation, nous avons fait le choix de travailler sur la partie SCAO car le projet nous semblait ambitieux avec beaucoup de défis à relever et faisait intervenir plusieurs aspects de notre formation tels que l'automatique, la modélisation et l'informatique industrielle (notamment l'utilisation de Matlab).

Contraintes :

Le sujet étant d'une importance majeure pour le projet IGOSAT, il nous demandait une forte implication et un travail continu afin de pouvoir avancer. La complexité de notre tâche fut dans la gestion du temps pour assurer une continuité de travail malgré les différents projets, cours et examens qu'on a en parallèle.

Une deuxième difficulté fut celle de devoir établir les équations de mouvements qui a demandé une bonne phase de recherche bibliographique.

Avec l'aide de notre encadrant M.Henri Inchauspé, et du chef du projet IGOSAT M.Hubert HALLOIN, nous avons pu gérer ces deux contraintes et mener à bien notre projet.

1.Chapitre 1 : Prise en main du problème :

1.1. Analyse de l'existant :

Une fois le sujet choisi, pour que nous puissions mieux appréhender la problématique du sujet, il fallait d'abord que l'on comprenne, au-delà des définitions de base, l'intérêt de la partie SCAO pour un satellite.

Pour cela on a dû lire de la documentation tantôt sur ce qui nous a été fourni au début du projet et tantôt en cherchant sur internet.

Ensuite, il a fallu avoir un aperçu clair sur le travail qu'il y avait à faire pour pouvoir établir des échéances raisonnables et définir un plan de travail clair et précis.

Nous avons donc lu les rapports de nos camarades qui ont travaillé sur cette même partie auparavant pour ne pas avoir à réinventer la roue. Une fois qu'on a fait le point sur ce qui a déjà été fait, on s'est mis d'accord sur la fréquence de travail. On a essayé de toujours consacrer deux demi-journées par semaine de travail (dont le Vendredi Matin qui était déjà réservé pour ce projet initialement).

Le premier rapport définissait une partie des composants physiques (Hardware) nécessaires pour la SCAO. Parmi ces composants on cite :

- ✓ Magnéto coupleurs : qui servent d'actionneurs pour appliquer les lois de contrôle.
- ✓ Senseurs solaires
- ✓ Antenne GPS
- ✓ …

Dans le deuxième rapport il était question de spécifications des besoins de la mission.

1.2. Outils:

Lors de notre travail, nous avons eu besoin d'utiliser plusieurs outils (spécialement des outils informatiques) dont on cite :

- ✓ Matlab : c'est un outil que nous avions l'habitude d'utiliser lors de notre formation à l'EIDD. Pour ce projet, toutes les parties de programmation (équations, résolution et linéarisation) ont été faite sur Matlab.
- ✓ Scilab : les positions et vitesses angulaires (que l'on va détailler plus loin dans ce rapport) qui nous ont été très utiles pour démarrer notre travail, ont été calculé grâce à ce logiciel. Nous avons donc dû le prendre en main.

2.Chapitre 2 : Etude de la dynamique (Approche Mathématique) :

2.1. Point de départ :

Pour bien réussir la SCAO il est essentiel de bien étudier et modéliser la dynamique du système. C'est donc une partie centrale de l'étude que nous avons mené.

Pour ce faire, nous nous sommes basés sur les calculs d'orbite et du référentiel de consigne effectués sur Scilab/Celestlab par M.Hubert HALLOIN :

- ✓ Calcul des positions et vitesses linéaires
- ✓ Calcul des positions et vitesses angulaires

On veut exprimer le mouvement angulaire sous forme matricielle en passant par les équations d'état. Pour comprendre la marche à suivre nous avons essayé de le faire pour le mouvement linéaire (Translations).

L'équation du départ pour le mouvement du centre de masse **(3ddl en translation)** est**:**

$$M.\frac{d^2\vec{v}(x,y,z)}{dt^2} = \sum \vec{F}$$

Pour faciliter la résolution numérique des équations, il est important d'exprimer ces dernières sous forme matricielle :

$$\dot{X} = A.X + B\vec{U}$$

Avec :

$$X = \begin{pmatrix} \dot{x} \\ x \\ \dot{y} \\ y \\ \dot{z} \\ z \end{pmatrix} \quad \rightarrow \quad \dot{X} = \begin{pmatrix} \ddot{x} \\ \dot{x} \\ \ddot{y} \\ \dot{y} \\ \ddot{z} \\ \dot{z} \end{pmatrix}$$

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \qquad B\vec{U} = \frac{1}{M}. \begin{pmatrix} F_x \\ 0 \\ F_Y \\ 0 \\ F_z \\ 0 \end{pmatrix}$$

A : Matrice d'évolution
B : Matrice de commande
U : Input (Forces appliquées sur le satellite)
X : Vecteur d'état.

L'intérêt d'utiliser la représentation d'état est que cette dernière permet de modéliser un système dynamique sous forme matricielle en utilisant des variables d'état. Cette représentation, peut être linéaire ou non, continue ou discrète et permet de déterminer l'état du système à n'importe quel instant futur si l'on connaît l'état à l'instant initial et le comportement des variables qui influent sur le système.

2.2 Améliorations:

Comme signalé précédemment cette partie représente le gros de notre étude.
Les grandes étapes de calcul sont décrites ci-dessous :

- ❑ Calcul d'orbite et du référentiel de consigne (fait par M.Halloin)
- ❑ Conversion des angles d'Euler en matrices de rotation
- ❑ Mise en place des équations de mouvements par rapport à J2000 puis par rapport au référentiel de consigne
- ❑ Implémentation de ces équations sous forme matricielle :

➔ Transcription des équations de mouvements sous matlab.
➔ Résolution de ces équations à l'aide Runge-kutta(4) "ode45".
➔ linéarisation des équations.
➔ résolution sous Matlab.

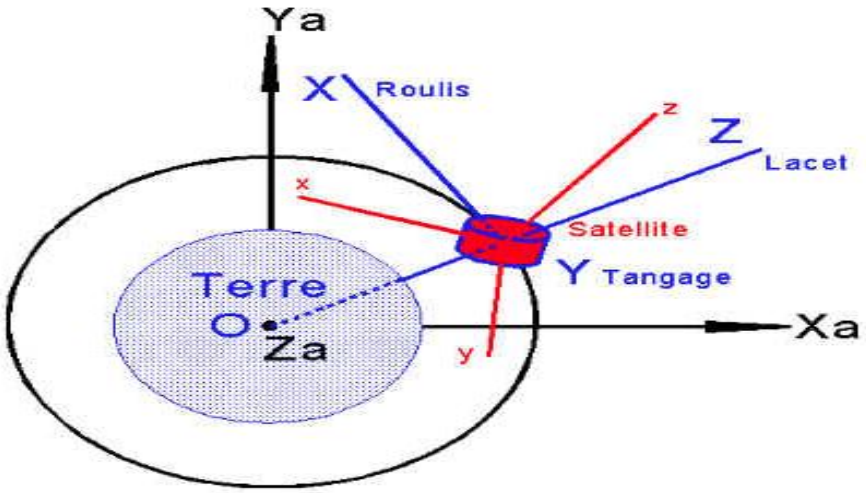
- **Utilité du SSM (State Space Model)**

Le but de la boucle d'asservissement de la SCAO est de comparer la consigne avec l'attitude mesurée et minimiser l'erreur entre ces deux grandeurs. Pour ce faire il est nécessaire de modéliser la dynamique dans le référentiel de consigne, car pour la linéarisation des équations il faut se référer au point d'équilibre. Toujours est-il que c'est plus simple d'exprimer d'abord ces équations dans le référentiel J2000, puis passer au référentiel de consigne en effectuant un changement de repère. Ceci sera détaillé plus loin dans le rapport.

2.3 ***Repères de travail :***

Rappelons brièvement que tout repérage à base d'angles présente des configurations spéciales où un angle n'est pas défini, entraînant des difficultés de programmation ou de continuité de cet angle.

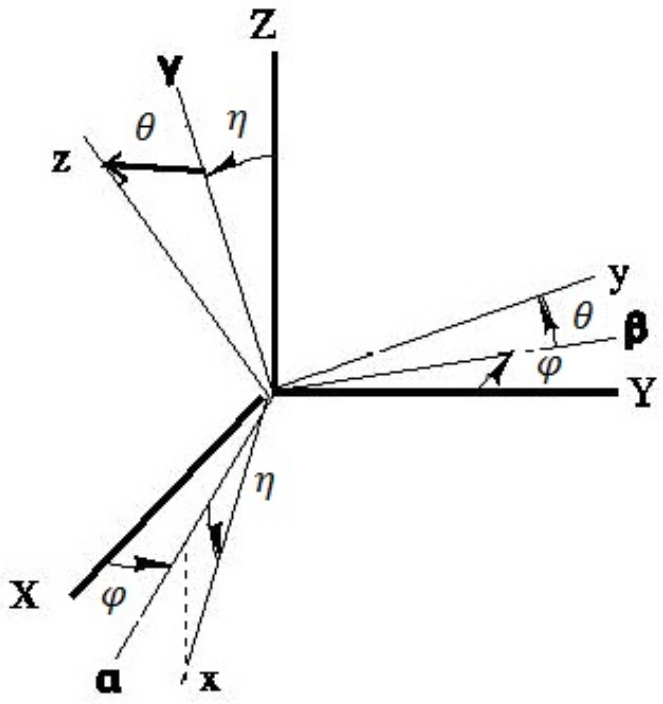
La question ne concerne que la connaissance de l'attitude du satellite par rapport à un repère de consigne ou de travail.



Xa Ya Za un repère absolu ou inertiel.

XYZ est le repère de consigne, pour nous ce sera le repère orbital : X suivant la vitesse, Y suivant le moment cinétique orbital, Z le zénith.

xyz est un repère lié au satellite, adapté à sa géométrie. Ci-dessous le repérage par angles de Cardan.



2.4 ***Dynamique dans J2000 :***

Dans ce qui suit, nous allons détailler les grandes étapes de calcul pour établir les équations de mouvement et leur résolution.

Afin d'étudier l'attitude du satellite, il nous faut écrire l'équation du mouvement dans le référentiel du satellite par rapport à J2000.Pour cela on utilise le théorème du moment dynamique :

$$I.\dot{\vec{\omega}}_{S/J} + \vec{\omega}_{S/J} \times I.\vec{\omega}_{S/J} = \overrightarrow{\mathcal{M}}$$

I : moment d'inertie.

$\vec{\omega}_{S/J}$: Vitesse angulaire du satellite par rapport à J2000.

On l'écrit sous forme matricielle :

$$M.\dot{X} = A.X + B\vec{U}$$

Avec :

$$X = \begin{pmatrix} \overrightarrow{\alpha s_{/J}} \\ \overrightarrow{\omega s_{/J}} \end{pmatrix} \quad \rightarrow \quad \dot{X} = \begin{pmatrix} \dot{\overrightarrow{\alpha s_{/J}}} \\ \dot{\overrightarrow{\omega s_{/J}}} \end{pmatrix}$$

Et

$$\overrightarrow{\alpha s_{/J}} = \begin{pmatrix} \theta_{S/J} \\ \eta_{S/J} \\ \varphi_{S/J} \end{pmatrix} \quad \rightarrow \quad \dot{\overrightarrow{\alpha s_{/J}}} = \begin{pmatrix} \dot{\theta}_{S/J} \\ \dot{\eta}_{S/J} \\ \dot{\varphi}_{S/J} \end{pmatrix}$$

$\overrightarrow{\alpha s_{/J}}$: Vecteur positon du satellite par rapport à J2000.

$\dot{\overrightarrow{\alpha s_{/J}}}$: Dérivée du vecteur position

Pour calculer la vitesse angulaire on procède de la façon suivante *:*

On a :

$$\overrightarrow{\omega s_{/J}} = \dot{\varphi}\overrightarrow{e_z} + \dot{\eta}R_{\varphi}\overrightarrow{e_y} + \dot{\theta}R_{\eta}R_{\varphi}\overrightarrow{e_x}$$

(On a considéré que les rotations autour des axes s'effectuent dans l'ordre suivant : $\overrightarrow{e_z}$, $\overrightarrow{e_y}$ et $\overrightarrow{e_x}$).

$$R_\varphi = \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$ Matrice de rotation autour de $\overrightarrow{e_z}$

$$R_\eta = \begin{pmatrix} \cos\eta & 0 & \sin\eta \\ 0 & 1 & 0 \\ -\sin\eta & 0 & \cos\eta \end{pmatrix}$$ Matrice de rotation autour de $\overrightarrow{e_y}$

$$R_\theta = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{pmatrix}$$ Matrice de rotation autour de $\overrightarrow{e_x}$

Le but c'est de pouvoir exprimer la vitesse angulaire sous la forme suivante :

$$\overrightarrow{\omega s_{/J}} = Bs_{/J}\left(\overrightarrow{\alpha s_{/J}}\right).\dot{\overrightarrow{\alpha s_{/J}}}$$

Par identification on trouve :

$$\overrightarrow{\omega s_{/J}} = \begin{pmatrix} cos\eta s_{/J} cos\varphi s_{/J} & -sin\varphi s_{/J} & 0 \\ sin\varphi s_{/J} & cos\varphi s_{/J} & 0 \\ -sin\eta s_{/J} cos\varphi s_{/J} & 0 & 1 \end{pmatrix} \begin{pmatrix} \dot{\theta} \\ \dot{\eta} \\ \dot{\varphi} \end{pmatrix}$$

$$Bs_{/J}\left(\overrightarrow{\alpha s_{/J}}\right) = \begin{pmatrix} cos\eta s_{/J} cos\varphi s_{/J} & -sin\varphi s_{/J} & 0 \\ sin\varphi s_{/J} & cos\varphi s_{/J} & 0 \\ -sin\eta s_{/J} cos\varphi s_{/J} & 0 & 1 \end{pmatrix}$$

On en déduit l'accélération angulaire

$$\dot{\overrightarrow{\omega s_{/J}}} = (Bs_{/J}\left(\dot{\overrightarrow{\alpha s_{/J}}}\right).\dot{\overrightarrow{\alpha s_{/J}}})$$

Pour résoudre numériquement sur Matlab cette équation on a eu besoin de transformer le produit vectoriel en produit scalaire :

$$\vec{\omega}_{S_{/J}} \times I.\vec{\omega}_{S_{/J}} \quad \rightarrow \quad \tilde{\omega}.I.\vec{\omega}_{S_{/J}}$$

Il faut donc coder une fonction qui prend un vecteur en entrée et donne une matrice en sortie, le principe mis en place est le suivant :

$$X \times Y = \begin{pmatrix} A \\ B \\ C \end{pmatrix} \times \begin{pmatrix} D \\ E \\ F \end{pmatrix} = \begin{pmatrix} BF - CE \\ AF - CD \\ AE - BD \end{pmatrix} = \begin{pmatrix} 0 & -C & B \\ -C & 0 & A \\ -B & A & 0 \end{pmatrix} \begin{pmatrix} D \\ E \\ F \end{pmatrix}$$

$$\tilde{X} = \begin{pmatrix} 0 & -C & B \\ -C & 0 & A \\ -B & A & 0 \end{pmatrix}$$

C'est-à-dire que : $X \times Y = \tilde{X}.Y$

Et on déduit la matrice M :

$$M = \begin{pmatrix} Bs_{/J}(\overrightarrow{\alpha s_{/J}}) & 0 \\ \widetilde{\omega s_{/J}}.I.Bs_{/J}(\overrightarrow{\alpha s_{/J}}) & I \end{pmatrix}$$

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

La forme finale de l'équation est la suivante :

$$\begin{pmatrix} Bs_{/J}(\overrightarrow{\alpha s_{/J}}) & 0 \\ \widetilde{\omega s_{/J}}.I.Bs_{/J}(\overrightarrow{\alpha s_{/J}}) & I \end{pmatrix} \begin{pmatrix} \dot{\overrightarrow{\alpha s_{/J}}} \\ \dot{\overrightarrow{\omega s_{/J}}} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \overrightarrow{\alpha s_{/J}} \\ \overrightarrow{\omega s_{/J}} \end{pmatrix} + B\vec{U}$$

2.5 ***Dynamique dans le référentiel de consigne :***

Équation de mouvement dans le réferentiel du satellite par rapport au référentiel orbital:

On rappelle l'équation de mouvement dans le référentiel du satellite par rapport à J2000

$$I.\dot{\vec{\omega}}_{s_{/J}} + \vec{\omega}_{s_{/J}} \times I.\vec{\omega}_{s_{/J}} = \overrightarrow{\mathcal{M}}$$

On effectue un changement de réf :

$$\vec{\omega}_{s_{/J}} = \vec{\omega}_{s_{/O}} + \vec{\omega}_{O_{/J}}$$

$$\dot{\vec{\omega}}_{s_{/J}} = \dot{\vec{\omega}}_{O_{/J}} + \dot{\vec{\omega}}_{s_{/O}} + \vec{\omega}_{O_{/J}} \times \vec{\omega}_{s_{/O}}$$

On remplace dans l'équation de mouvement :

$$I.(\dot{\vec{\omega}}_{o/_J} + \dot{\vec{\omega}}_{s/_o} + \vec{\omega}_{o/_J} \times \vec{\omega}_{s/_o}) + \left(\vec{\omega}_{s/_o} + \vec{\omega}_{o/_J}\right) \times I.\left(\vec{\omega}_{s/_o} + \vec{\omega}_{o/_J}\right) = \overrightarrow{\mathcal{M}}$$

On exprime cette équation sous forme matricielle :

$$\begin{pmatrix} B(\vec{\alpha}_{s/_o}) & 0 \\ Z_{s/_o} & I \end{pmatrix} \cdot \begin{pmatrix} \dot{\overrightarrow{\alpha_{s/_o}}} \\ \dot{\overrightarrow{\omega_{s/_o}}} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \overrightarrow{\alpha_{s/_o}} \\ \overrightarrow{\omega_{s/_o}} \end{pmatrix} + \begin{pmatrix} 0 \\ \vec{T} \end{pmatrix}$$

Avec :

$$B_{s/_o}\left(\overrightarrow{\alpha_{s/_o}}\right) = \begin{pmatrix} cos\eta_{s/_o} cos\varphi_{s/_o} & -sin\varphi_{s/_o} & 0 \\ sin\varphi_{s/_o} & cos\varphi_{s/_o} & 0 \\ -sin\eta_{s/_o} cos\varphi_{s/_o} & 0 & 1 \end{pmatrix}$$

$$\overrightarrow{\omega_{s/_o}} = B_{s/_o}\left(\overrightarrow{\alpha_{s/_o}}\right).\dot{\overrightarrow{\alpha_{s/_o}}} \quad \rightarrow \quad \dot{\overrightarrow{\omega_{s/_o}}} = (B_{s/_o}\left(\dot{\overrightarrow{\alpha_{s/_o}}}\right).\dot{\overrightarrow{\alpha_{s/_o}}})$$

$$\dot{\overrightarrow{\omega_{s/_o}}} = (B_{s/_o}\left(\dot{\overrightarrow{\alpha_{s/_o}}}\right).\dot{\overrightarrow{\alpha_{s/_o}}})$$

$$\dot{\overrightarrow{\alpha_{s/_o}}} = \begin{pmatrix} \dot{\theta}_{s/_o} \\ \dot{\eta}_{s/_o} \\ \dot{\varphi}_{s/_o} \end{pmatrix} \rightarrow \overrightarrow{\alpha_{s/_o}} = \begin{pmatrix} \theta_{s/_o} \\ \eta_{s/_o} \\ \varphi_{s/_o} \end{pmatrix}$$

$$Z_{s/_o} = \tilde{\omega}_{s/_o}.I.B_{s/_o}\left(\overrightarrow{\alpha_{s/_o}}\right) + \tilde{\omega}_{o/_J}.I.B_{s/_o}\left(\overrightarrow{\alpha_{s/_o}}\right) + I.\tilde{\omega}_{o/_J}.B_{s/_o}\left(\overrightarrow{\alpha_{s/_o}}\right) - \left(\widetilde{I.\tilde{\omega}_{o/_J}}\right).B_{s/_o}\left(\overrightarrow{\alpha_{s/_o}}\right)$$

$$\vec{T} = \overrightarrow{\mathcal{M}} - I.\dot{\vec{\omega}}_{o/_J} - \tilde{\omega}_{o/_J}.(I.\vec{\omega}_{o/_J})$$

2.6 ***Perspectives:***

Cette partie est pratiquement bouclée, ce qui reste à faire est d'arriver à faire compiler le programme de résolution de la discrétisation du SSM (Space state model) linéaire décrite ci-dessus. Nous n'avons pas eu le temps de bien faire tourner le modèle.

3. Chapitre 3 : Etude de la dynamique (Simulation/Modélisation) :

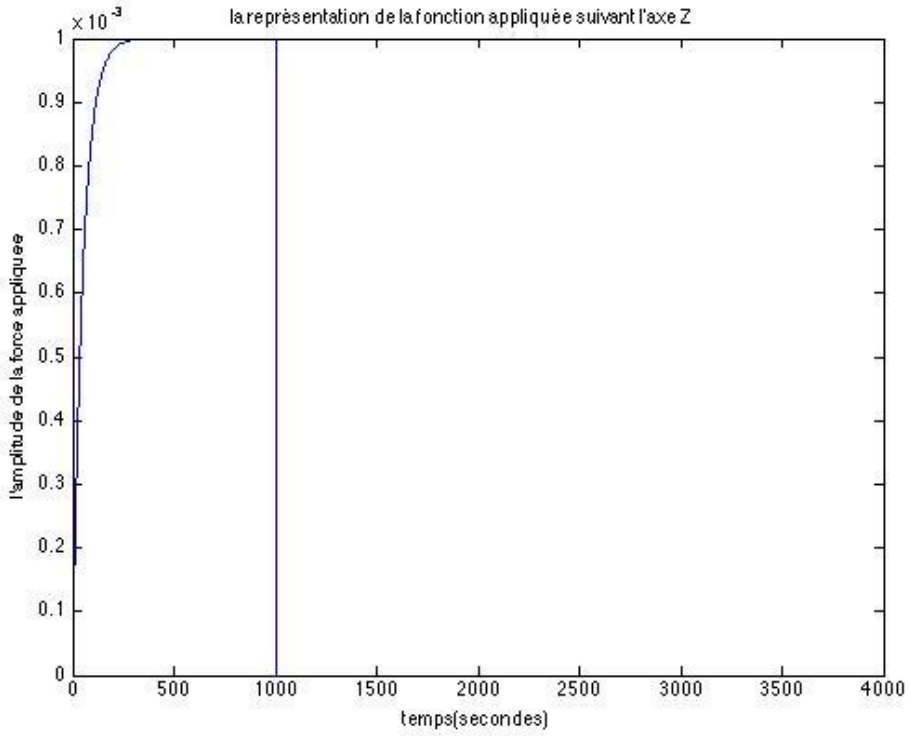
3.1. Simulation numérique sous Matlab :

Lors de la simulation de la dynamique du système sous Matlab, nous avons eu besoin de programmer plusieurs fonctions, certaines pour simplifier le calcul, d'autres pour la résolution ou encore linéarisation.
Les Scripts de ces programmes sont fournis en annexes de ce document.

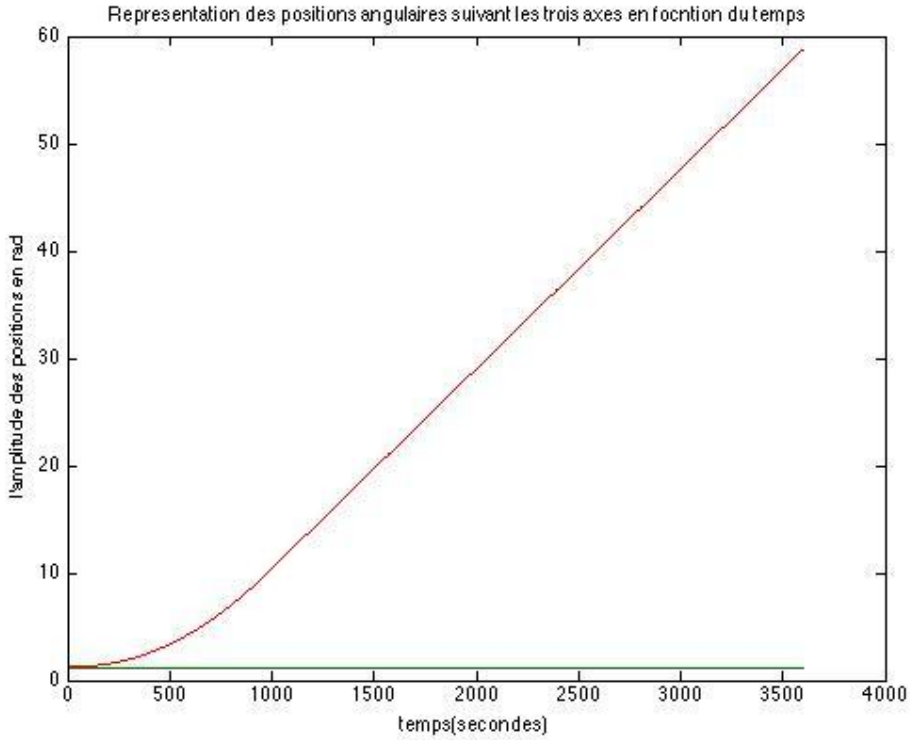
- « Euler2RotMat » : cette fonction permet de transformer les angles d'Euler en matrice de rotation.
- « matrice » : cette fonction prend en entrée un vecteur et donne en sortie une matrice. Cela nous permet de simplifier le calcul, et d'éviter de faire le calcul en utilisant produit vectoriel et rapporte la résolution en scalaire
- « bdealpha » : permet d'avoir le vecteur vitesse en fonction du vecteur position.
- « SSM » permet de résoudre l'équation du mouvement du centre de masse dans le référentiel du satellite par rapport à J2000 (translation).
- « SSMEULER » Cette fonction permet de résoudre l'équation de mouvement du satellite autour de son centre de masse (l'attitude).
  Les résultats obtenus sont exprimés dans le référentiel du satellite (toujours par rapport à J2000).
- « SSMEULERNLIN » qui permet de résoudre l'équation de mouvement du satellite autour de son centre de masse (l'attitude). Sauf que cette fois, les résultats sont exprimés dans le référentiel orbital (ce cas présente un problème de non linéarité)

3.2. Résultats :

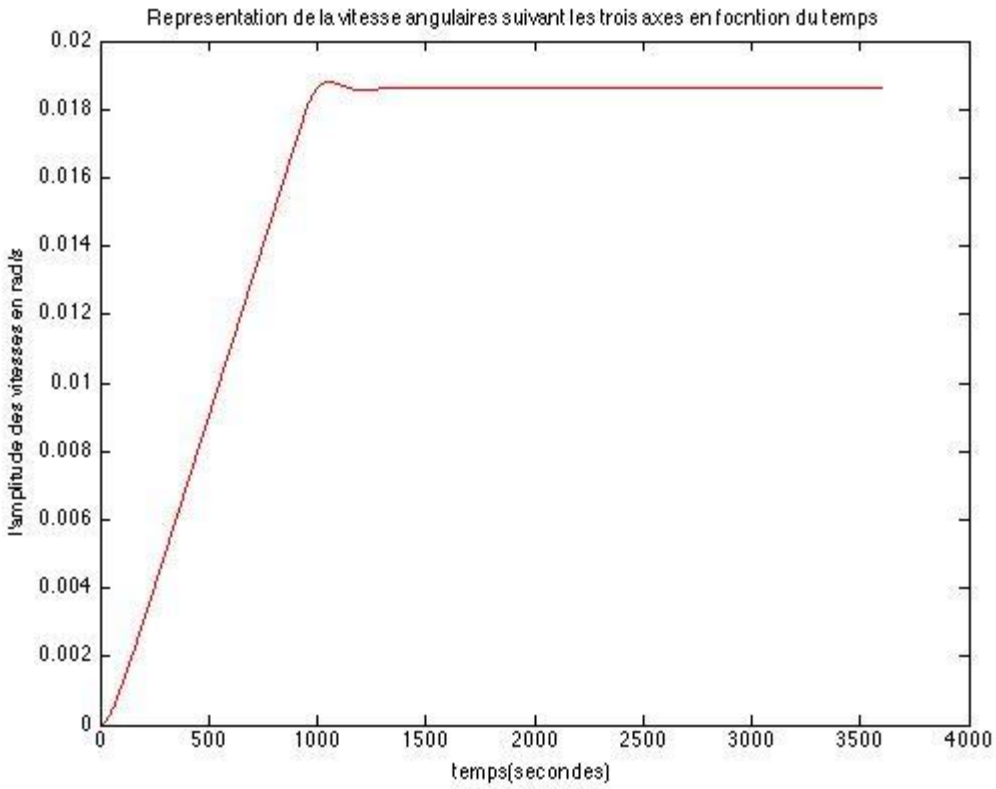
On a appliqué un couple non nul sur le système durant un intervalle de temps (en dehors de l'intervalle le couple est nul) pour le mouvement rotationnel dans le cas linéaire et dans le cas non linéaire.



Le graphe ci-dessus représente le couple appliqué au système autour de l'axe Z pour une durée de 1000 secondes.



Le graphe ci-dessus représente l'évolution (en rouge) de la position angulaire suivant Z. La position angulaire est orientée suivant Z.



Ce graphe (en rouge) représente l'évolution de la vitesse angulaire suivant Z. sur l'intervalle de l'application du couple [0,1000] la vitesse croit, oscille quelque secondes après, puis se stabilise au-delà. Cette réponse correspond à l'attendu de la sollicitation d'entrée.

3.3. Améliorations :

Ce qui reste à améliorer dans cette partie, c'est de faire tourner le programme de discrétisation du SSM (Voir annexe).

4. Chapitre 4 : Contrôleur PID:

Afin de corriger l'erreur calculée entre l'attitude demandée et l'attitude mesurée on utilisera un régulateur PID.

On détaillera ci-dessous le fonctionnement d'un régulateur PID :

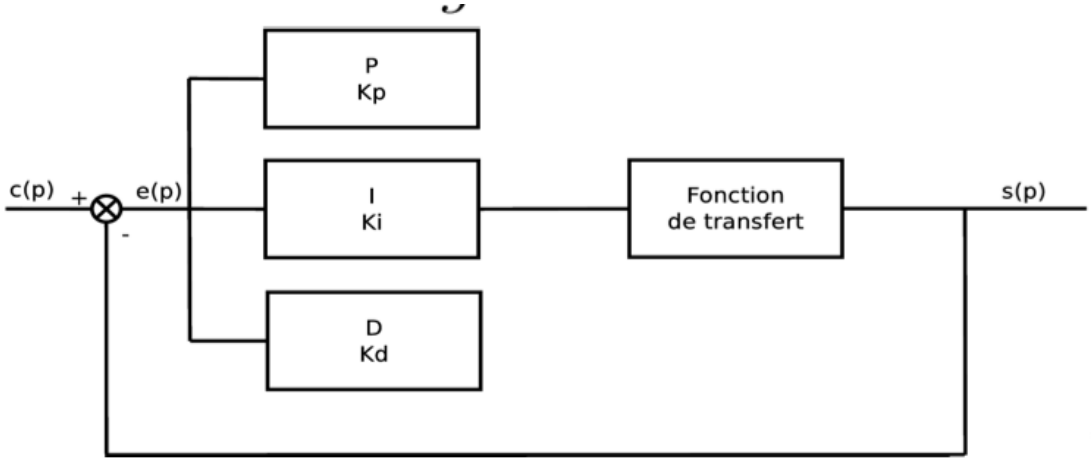
1. Définition :

C'est un système d'auto régulation (boucle fermée), qui cherche à réduire l'erreur entre la consigne et la mesure.

$$e = consigne - mesure$$

Consigne : attitude voulue.
Mesure : attitude mesurée.

Pour stabiliser le système on vient réinjecter négativement l'erreur mesurée dans la consigne.



2. ***<u>À quoi ça sert ?</u>***

Atteindre la valeur souhaitée du vecteur position, vecteur vitesse ou bien vecteur accélération.

- Régulation : Pour minimiser rapidement les perturbations (bruit d'actuation, couple solaire, couple gravitationnel….).
- Poursuite : Pour s'adapter rapidement aux nouvelles consignes.

3. ***<u>Proportionnel :</u>***

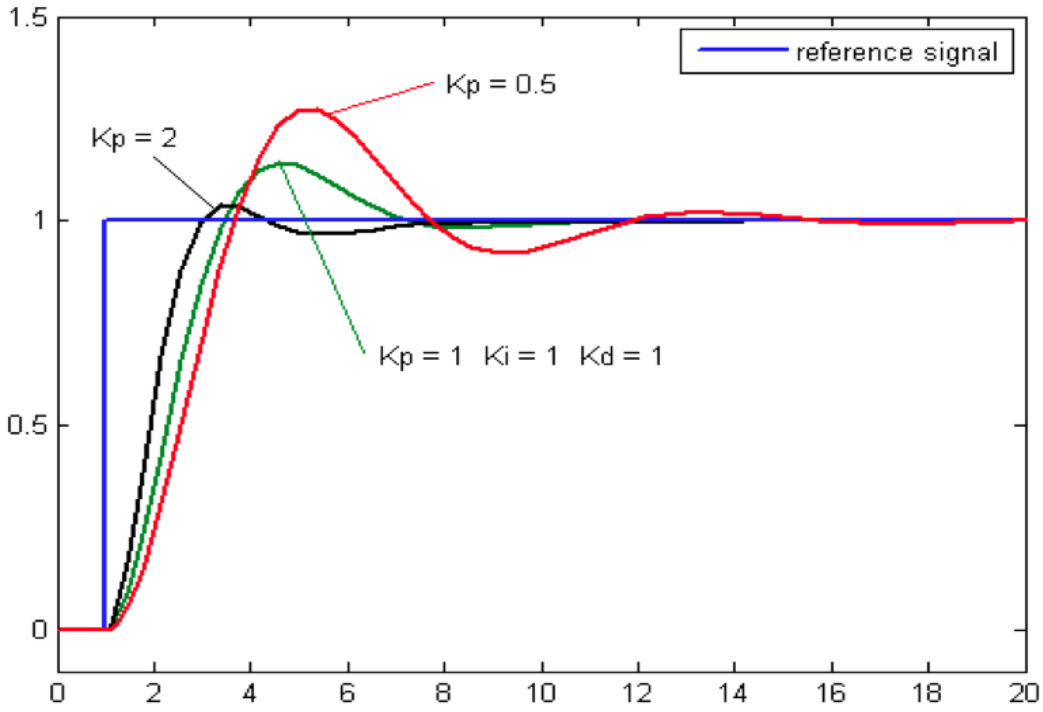
L'erreur est multipliée par une constante K_p

$$U(t) = K_p.e(t)$$

Plus K_p est grande, plus la réponse est rapide.
C'est ce qu'on appelle l'erreur statique.

On applique cela sur une fonction Heaviside et on remarquera bien l'utilité d'avoir utilisé K_p plus grand.

On trace en temporel l'influence de la constante K_p :



4. ***Intégral :***

Dans le cas d'une régulation par intégration, l'erreur est intégrée sur un intervalle de temps, puis multipliée par une constante K_i.

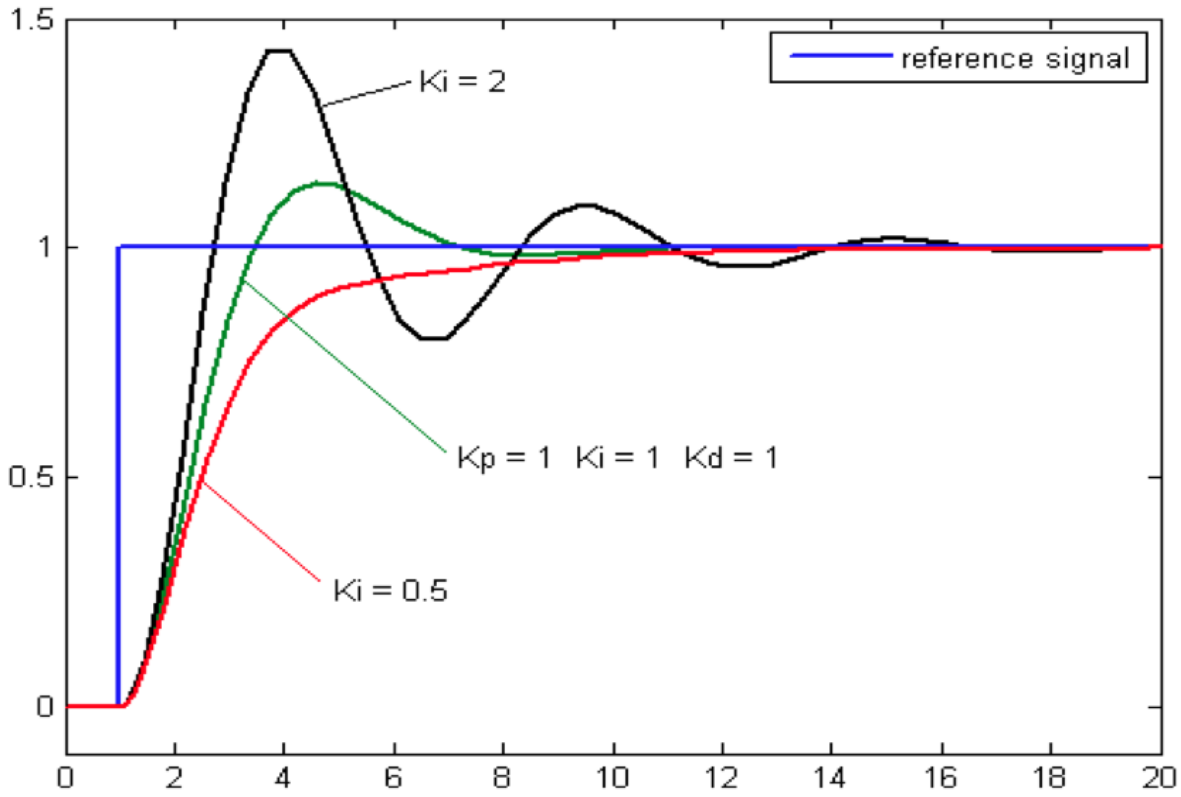
$$U(t) = K_i \int_0^{\tau} e(\tau)\, d\tau$$

Sinon dans le domaine de Laplace on divise par P :

$$U(p) = K_i \frac{e(p)}{p}$$

Pour corriger l'erreur dans le cas d'une intégration on doit augmenter la constante K_i. Parce plus K_i est élevée plus l'erreur statique est corrigée.

On applique cela toujours pour une fonction Heaviside et on en aura comme résultats dans la figure ci-dessous :



5. ***Dérivé :***

Dans le cas d'une régulation par dérivation, l'erreur est dérivée par rapport au temps, puis multipliée par une constante K_d.

$$U(t) = K_d . \frac{de(t)}{dt}$$

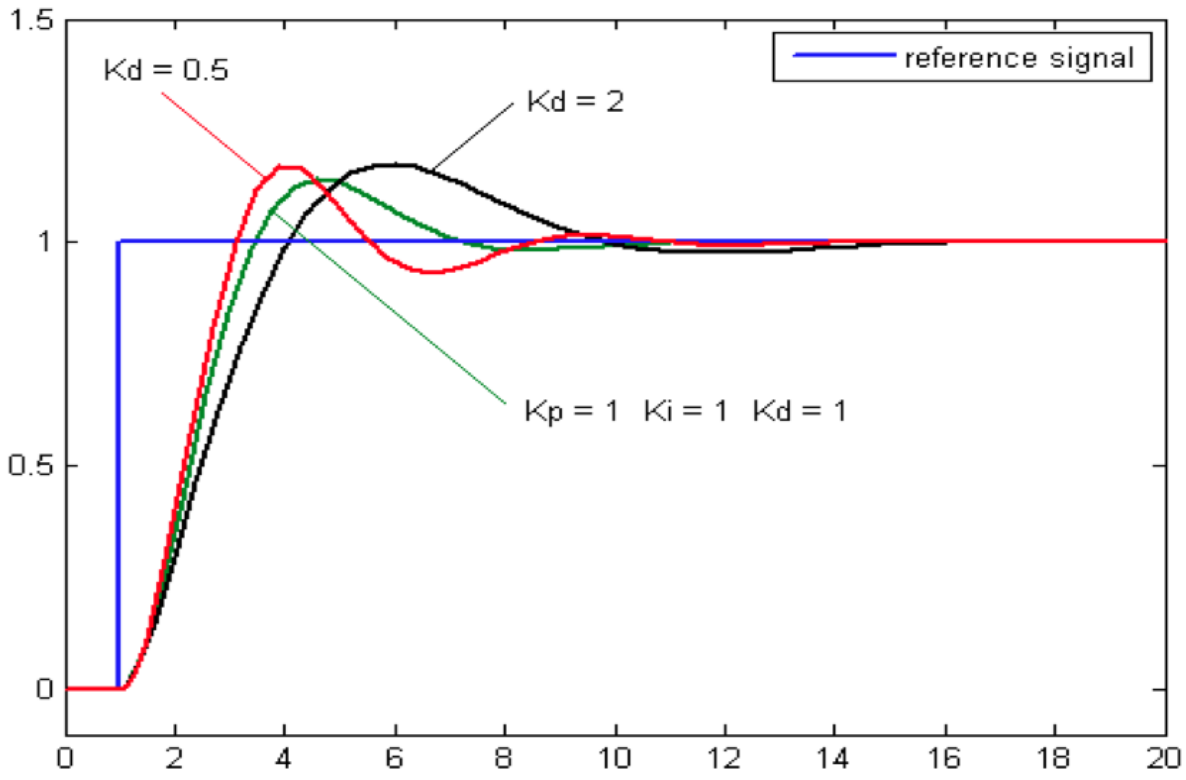
Sinon dans le domaine de Laplace on divise par P :

$$U(p) = K_d . e(p) . p$$

La correction de cette erreur conduit à réduire le dépassement et le temps de stabilisation.

Plus K_d est grande plus le dépassement de l'erreur statique est petit par rapport à la fonction appliqué en entrée.

On observe dans la figure ci-dessous le comportement d'erreur statique sur une fonction Heaviside en fonction de différentes valeurs de K_d :



6. ***Résumé***

On en déduit l'erreur totale :

Dans le domaine temporel :

$$U(t) = K_p.e(t) + K_d.\frac{de(t)}{dt} + K_i \int_0^{\tau} e(\tau)\, d\tau$$

Dans le domaine de Laplace :

$$U(t) = K_p.e(t) + K_d.e(p).p + K_i \frac{e(p)}{p}$$

Le tableau ci-dessous montre l'influence des trois coefficients (K_p, K_d et K_i) sur le temps de monté, temps de stabilisation, dépassement et enfin l'erreur statique.

Coefficient	Temps de montée	Temps de stabilisation	Dépassement	Erreur Statique
Kp	Diminue	Augmente	Augmente	Diminue
Ki	Diminue	Augmente	Augmente	Annule
Kd	–	Diminue	Diminue	-

Conclusion :

Le travail sur ce projet, a été pour nous une bonne occasion de confronter la théorie avec la pratique. En effet, la mise en œuvre des bases théoriques acquises lors de notre formation nous a beaucoup aidés à mener à bien cette mission.

Au début du projet, la complexité du travail nous empêchait de prendre l'initiative. Cette situation n'a pas trop duré. En effet, grâce aux conseils de notre encadrant , nous avons pu affronter cette difficulté et nous sommes arrivés à de bons résultats.

La résolution de la problématique en question demandait une certaine polyvalence, nous avons donc mis en œuvre beaucoup de compétences acquises lors de notre parcours durant nos 3années d'études à l'EIDD telles que l'automatique, informatique, modélisation.

Indépendamment de l'aspect technique, ce sujet nous a permis de faire un pas de plus vers l'accomplissement de notre projet professionnel.

J'ai pu voir de près ce qu'est réellement le métier d'ingénieur, et comment on pouvait travailler en collaboration très étroite avec les autres tout en restant très autonome.

Annexe 1 :

La fonction « Euler2RotMat » :

```
function R = Euler2RotMat(alpha)

   R_z = [cos(alpha(3)), -sin(alpha(3)), 0; ...
          sin(alpha(3)),  cos(alpha(3)), 0; ...
          0,              0,             1];

   R_y = [cos(alpha(2)),  0, sin(alpha(2)); ...
          0,              1, 0; ...
          -sin(alpha(2)), 0, cos(alpha(2))];

   R_x = [1, 0,              0; ...
          0, cos(alpha(1)),  sin(alpha(1)); ...
          0, -sin(alpha(1)), cos(alpha(1))];

   R = R_x*R_y*R_z;

end
```

Le script :

```
DataAng = importdata('IGOSAT_EULER_ANGLES_Mission.txt'); % fichier TXT contenant les vecteurs positions des angles d'euler
angle_x=DataAng(:,3);
angle_y=DataAng(:,5);
angle_z=DataAng(:,7);

alpha = DataAng(:, [3,5,7]);
%%

RotMat =  zeros(3,3, size(alpha,1));
for i = 1:size(alpha,1)
   RotMat(:,:,i) = Euler2RotMat(alpha(i,:));
end
```

Annexes 2 :

La fonction « matrice » :

```
function A=matrice(vecteur)

A=[0, -vecteur(3), vecteur(2);...
   -vecteur(3), 0, vecteur(1);...
   -vecteur(2), vecteur(1), 0;];


end
```

Annexes 3 :

La fonction « SSM » :

```
function Xp=SSM(t, X, U)
A=[0 0 0 0 0 0;...
   1 0 0 0 0 0; ...
   0 0 0 0 0 0; ...
   0 0 1 0 0 0; ...
   0 0 0 0 0 0; ...
   0 0 0 0 1 0];

Xp=A*X+U;

End
```

Le script :

```
Fx=20;% la projection de la force selon x
Fy=20;% la projection de la force selon y
Fz=20;
m=3;
A=[0 0 0 0 0 0;...
   1 0 0 0 0 0; ...
   0 0 0 0 0 0; ...
   0 0 1 0 0 0; ...
   0 0 0 0 0 0; ...
   0 0 0 0 1 0];
U=[(Fx/m); 0; (Fy/m); 0; (Fz/m); 0];


t=1:30:3000;
X0 = zeros(1,6);
%%
[T,Y]=ode45(@(t, X) SSM(t, X, U), t, X0);
%%
figure(1)
plot(T,Y(:,1:3));
title('Representation des positions angulaires suivant les trois axes en focntion du temps');
xlabel('temps(secondes)');
ylabel('l"amplitude des positions en rad');
figure(2)
plot(T,Y(:,4:6));
title('Representation de la vitesse angulaires suivant les trois axes en focntion du temps');
xlabel('temps(secondes)');
ylabel('l"amplitude des vitesses en rad/s');
```

Annexes 4 :

La fonction « SSMEULER » :

```
function Xp=SSMEULER(t, X, U)

m = 3;% la masse du nanosatellite
a=10;b=10;c=30;
I=[m/12*(b^2+c^2), 0,             0; ...
   0,              m/12*(a^2+c^2), 0; ...
   0,              0,              m/12*(a^2+b^2)];% I c'est la mtrice d'inertie

B=bdealpha(X(1:3));% dÈfinie dans le rapport

Z=matrice(X(4:6))*I*B;

M=[B, zeros(3,3);...
    Z, I;];

A=[zeros(3,3), eye(3,3);...
    zeros(3,3), zeros(3,3);];

tspan=1:1:3600;
U_Interp = interp1(tspan, U.', t);

Xp=M\(A*X + U_Interp.');

End
```

Le script :

```
DataAng = importdata('IGOSAT_EULER_ANGLES_Mission.txt');
DataPos = importdata('IGOSAT_POSITION_VELOCITY.txt');

angle_x=DataAng(:,3);
angle_y=DataAng(:,5);
angle_z=DataAng(:,7);

alpha = DataAng(:, [3,5,7]);
alpha0 = alpha(1,:);
Rot_B2J = Euler2RotMat(alpha0).';

a=1;b=2;c=2;
I=[a, 0, 0;...
    0, b, 0;...
    0, 0, c];

B=bdealpha(alpha(1,:));
w=zeros(3,1);


dw=ones(3,1);
wtild=matrice(w);
```

```
X0 = [alpha(1:3,1);w];

 tspan=1:1:3600;

U=zeros(6,numel(tspan));
U(6, 1:1000) = 1e-3*(1-exp(-0.02*tspan(1:1000)));
U_J = zeros(size(U));
for t = 1:size(U, 2)
    U_J(4:6,t) = Rot_B2J*U(4:6,t);
end

%%

[T,Y]=ode45(@(t, X) SSMEULER(t, X, U), tspan, X0);

%%

figure(1)
plot(T,Y(:,1:3));
title('Representation des positions angulaires suivant les trois axes en focntion du temps');
xlabel('temps(secondes)');
ylabel('l''amplitude des positions en rad');
figure(2)
plot(T,Y(:,4:6));
title('Representation de la vitesse angulaires suivant les trois axes en focntion du temps');
xlabel('temps(secondes)');
ylabel('l''amplitude des vitesses en rad/s');
figure(3)
plot(T,U(6,:));
title('la reprÈsentation de la fonction appliquÈe suivant l''axe Z');
xlabel('temps(secondes)');
ylabel('l''amplitude de la force appliquee');
```

Annexes 5 :

La fonction « SSMEULERNLIN » :

```
function Xp=SSMEULERNLIN(t, X, U, tspan, Alpha_O_J, Omega_O_J, dOmega_O_J)

m = 3;% la masse du nanosatellite
a=10;b=10;c=30;
I=[m/12*(b^2+c^2), 0,            0; ...
   0,            m/12*(a^2+c^2), 0; ...
   0,            0,            m/12*(a^2+b^2)];% I c'est la mtrice d'inertie

B_alpha=bdealpha(X(1:3));
```

```
U_Interp = interp1(tspan, U.', t).';% interpolation de la force appliquÈ % U le vecteur qui contient la force appliquÈe
omega_O_J_O_Interp = interp1(tspan, Omega_O_J,t);% interpolation du vecteur vitesse angulaire du satellite par rapport ‡ J2000 exprimÈe dans le rÈfÈrenciele orbital
domega_O_J_O_Interp = interp1(tspan, dOmega_O_J,t);%l'accÈlÈration angulaire du satellite par rapport ‡ J2000 exprimÈe dans le rÈfÈrenciele orbital

T_B_O = Euler2RotMat(X(1:3));% vecteur qui permet d'exprimer les vecteurs vitesses et accÈlÈrations dans le rÈfÈrence orbital au lieu de les exprimer dans le rÈfÈrentiel du satellite
T_O_B = T_B_O.';
omega_O_J_B_Interp = T_B_O*omega_O_J_O_Interp.';% Interpolation du vecteur vitesse angulaire exprimÈ dans le rÈfÈrence orbital
domega_O_J_B_Interp = T_B_O*domega_O_J_O_Interp.';% Interpolation du vecteur d'accÈlÈration angulaire exprimÈ dans le rÈfÈrence orbital

Z = matrice(X(4:6))*I*B_alpha + matrice(omega_O_J_B_Interp)*I*B_alpha +
I*matrice(omega_O_J_B_Interp)*B_alpha - matrice(I*omega_O_J_B_Interp)*B_alpha;

F_Inertia = T_O_B*(-I*domega_O_J_B_Interp - matrice(omega_O_J_B_Interp)*I*(omega_O_J_B_Interp));
U_Inertia = [F_Inertia];

U_Interp_total = U_Interp + U_Inertia;

M=[B_alpha, zeros(3,3);...
   Z,       I;];

A=[zeros(3,3), eye(3,3);...
   zeros(3,3), zeros(3,3)];

B = [zeros(3,3); T_B_O];

iMA = M\A;% inverse de M multipliÈ par la matrice A
iMB = M\B;% inverse de M multipliÈ par la matrice B

Xp = iMA*X + iMB*U_Interp_total;% l'Équation finale % Xp c'est la derivÈe de X % X c'est vecteur 6, il contient vecteur position et vecteur vitese

fprintf('Omega = [%f, %f, %f] at time t = %f\n', X(4:6), t);

end
```

Le script :

```
DataAng = importdata('IGOSAT_EULER_ANGLES_Mission.txt');
DataPos = importdata('IGOSAT_POSITION_VELOCITY.txt');

Index = find(ismember(DataAng(:,1), 58284));
tspan = DataAng(Index,2);
% angle_x = DataAng(Index,3);
% angle_y = DataAng(Index,5);
% angle_z = DataAng(Index,7);
angle_x = DataAng(Index,4);
angle_y = DataAng(Index,6);
angle_z = DataAng(Index,8);

%Alpha_O_J = DataAng(Index, [3,5,7]);
Alpha_O_J = unwrap(DataAng(Index, [4,6,8]));

N = size(Alpha_O_J, 1);

dt = 30;
dAlpha_O_J = [diff(Alpha_O_J, 1)/dt, ; zeros(1,3)];
Omega_O_J = zeros(N,3);
%dOmega_O_J = zeros(N,3);
for i = 1:N
   Omega_O_J(i,:) = bdealpha(Alpha_O_J(i,:))*dAlpha_O_J(i,:).';
end
dOmega_O_J = [diff(Omega_O_J)/dt; zeros(1,3)];

%%

alpha0 = Alpha_O_J(1,:);
Rot_B2J = Euler2RotMat(alpha0).';

a=1;b=2;c=2;
I=[a, 0, 0;...
   0, b, 0;...
   0, 0, c];

w=zeros(3,1);

X0 = [Alpha_O_J(1:3,1);w];

U=zeros(3,numel(tspan));
 [T,Y]=ode45(@(t, X) SSMEULERNLIN(t, X, U, tspan, Alpha_O_J, Omega_O_J, dOmega_O_J), tspan, X0);
```

Annexes 5 :

La fonction « bdealpha » :

```
function Balpha=bdealpha(alpha)
Balpha=[cos(alpha(2)).*cos(alpha(3)),-sin(alpha(3)),0;...
                sin(alpha(3)), cos(alpha(3)),0;...
       -sin(alpha(2)).*cos(alpha(3)),          0,1;];
 End
```

Le script :

```
DataAng = importdata('IGOSAT_EULER_ANGLES_Mission.txt');
alpha = DataAng(:, [3,5,7]);

 B =  zeros(3,3, size(alpha,1));

 for i = 1:size(alpha,1)
    B = bdealpha(alpha(i,:));
end
```