

Développement d'un algorithme pour évaluer la traînée  
électromagnétique causée par une courbe 3D paramétrée le  
long d'une orbite de satellite

Par

Martin BEAUVAIS

RAPPORT DE PROJET

MONTREAL, LE 6 DECEMBRE 2024

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

© Tous droits réservés, Martin Beauvais, 2024



# TABLE DES MATIERES

LISTE DES ILLUSTRATIONS .....	V
LISTE DES TABLEAUX .....	V
LISTE DES ABREVIATIONS .....	VI
LISTE DES SYMBOLES ET UNITES DE MESURE .....	VII
1 INTRODUCTION .....	1
1.1 Contexte actuel .....	1
1.2 Revue littéraire.....	3
2 METHODOLOGIE.....	12
2.1 Objectifs du projet .....	12
2.2 Python .....	13
2.3 Module ppigrf.....	14
2.4 Logiciel GMAT .....	16
2.5 Méthodes suivies .....	17
2.5.1 Hypothèses de calculs .....	17
2.5.2 Calcul de la force électromagnétique.....	17
2.5.3 Repères liés à la Terre .....	20
2.5.4 Repère lié au satellite .....	22
3 RESULTATS.....	24
3.1 Code de l'outil.....	24
3.1.1 Module ForceElectro .....	24
3.1.2 Module ChampMagnétique .....	26
3.1.3 Module Satellite .....	26
3.1.4 Module Câble .....	28
3.1.5 Module Matériaux .....	28
3.2 Code de validation et d'application de la méthode :.....	29
3.2.1 Validation de la méthode .....	29
3.2.2 Premier code de test :.....	34
3.2.3 Deuxième code de test : .....	35
4 DISCUSSION.....	39
4.1 Limites du code :.....	39

4.2	Perspectives et pistes d'améliorations : .....	40
4.3	Impacts du projet : .....	41
CONCLUSION .....		42
BIBLIOGRAPHIE .....		43
ANNEXE I Module ForceElectro .....		45
ANNEXE II Module ChampMagnétique .....		50
ANNEXE III Module Satellite .....		51
ANNEXE IV Module Câble .....		54
ANNEXE V Module Matériaux .....		57
ANNEXE VI Code Test_Force_Electro .....		64
ANNEXE VII Code Test_Cable_rectiligne .....		65
ANNEXE VIII Code Test_Courbe_Parametree .....		67
ANNEXE IX Simulation GMAT .....		70

## LISTE DES ILLUSTRATIONS

Figure 1-1 : Régions où peuvent avoir lieu les orbites de stockage [2].....	2
Figure 1-2 : Détails du concept du Terminator Tether [1] .....	5
Figure 1-3 : Evolution des câbles électrodynamiques [4] .....	7
Figure 1-4 : Modes opératoire du kit de désorbitage E.T.PACK lors de son vol d'essai [4] .....	9
Figure 2-1 : Schéma du satellite et de son câble .....	18
Figure 2-2 : Schéma de changement de repère entre le repère sphérique ECEF et le repère du satellite.....	22
Figure 3-1 : Résultats 1er code de test .....	33
Figure 3-2 : Résultats du code Test_Cable_rectiligne .....	35
Figure 3-3 : Câble circulaire .....	36
Figure 3-4 : Résultats du code Test_Courbe_Parametree pour le premier cas .....	36
Figure 3-5 : Câble défini par morceaux.....	37
Figure 3-6 : Résultats du code Test_Courbe_Parametree pour le deuxième cas .....	37
Figure 3-7 : Résultats du code Test_Courbe_Parametree pour le troisième cas .....	38

## LISTE DES TABLEAUX

Tableau 2-1 : Liste des bibliothèques Python utilisées et leur version .....	14
Tableau 3-1 : Tableau comparatif des résultats du 1er code de test .....	33

## LISTE DES ABREVIATIONS

API	Interface de Programmation d'Application
ECEF	<i>Earth Centred Earth Fixed</i>
EDT	<i>ElectroDynamic Tether</i>
ESA	Agence Spatiale Européenne
GEO	<i>Geostationaty Earth Orbit</i>
GMAT	<i>General Mission Analysis Tool</i>
IADC	Inter-agence sur les débris spatiaux
IGRF	<i>International Geomagnetic Reference Field</i>
LEO	<i>Low Earth Orbit</i>
LWT	<i>Low Work function Tether</i>
NASA	<i>National Aeronautics and Space Administration</i>
ONU	Organisation des Nations Unies
SSO	<i>Sun-Synchronous Orbit</i>
UTC	Temps Universel Coordonné

## LISTE DES SYMBOLES ET UNITES DE MESURE

<u>Symbole</u>	<u>Unité</u>	<u>Définition</u>
$a$	$[km]$	Demi grand axe d'une ellipse
$\vec{B}$	$[T]$	Champ magnétique
$B_r$	$[T]$	Composante radiale du champ magnétique, en sphérique
$B_\theta$	$[T]$	Composante selon $\theta$ du champ magnétique, en sphérique
$B_\varphi$	$[T]$	Composante selon $\varphi$ du champ magnétique, en sphérique
$\vec{E}$	$[V.m^{-1}]$	Champ électrique
$e$		Excentricité de l'orbite
$\vec{F}$	$[N]$	Force de Lorentz
$F_x$	$[N]$	Composante selon $x$ de la force de Lorentz
$F_y$	$[N]$	Composante selon $y$ de la force de Lorentz
$F_z$	$[N]$	Composante selon $z$ de la force de Lorentz
$G = 6.6743 \times 10^{-11}$	$[m^3.kg^{-1}.s^{-2}]$	Constante de Gravitation universelle
$G_T$		Centre de gravité de la Terre
$G_S$		Centre de gravité du satellite
$\vec{I}$	$[A]$	Vecteur intensité du courant
$I = \ \vec{I}\ $	$[A]$	Intensité du courant
$i$	$[deg]$	Inclinaison de l'orbite
$\vec{L}$	$[m]$	Vecteur longueur du câble (dans le cas où le câble est rectiligne)
$L$	$[m]$	Longueur du câble
$M_T = 5.972168 \times 10^{24}$	$[kg]$	Masse de la Terre
$R$	$[\Omega]$	Résistance électrique du câble
$R_T = 6378100$	$[m]$	Rayon de la Terre

<b><u>Symbole</u></b>	<b><u>Unité</u></b>	<b><u>Définition</u></b>
$R_{Earth \rightarrow Sat}$		Matrice de changement de base entre le repère ECEF et le repère lié au satellite
$\vec{r}$	$[km]$	Vecteur unitaire dans la direction 1 des coordonnées sphériques
$r$	$[km]$	Rayon
$S$	$[m^2]$	Surface de section du câble
$t$	$[s]$	Temps
$U$	$[V]$	Tension électrique
$\vec{V}$	$[m.s^{-1}]$	Vecteur vitesse
$V$	$[m.s^{-1}]$	Vitesse
$\vec{x}$	$[m]$	Vecteur unitaire dans la direction 1 des coordonnées cartésiennes
$x$	$[m]$	Position en abscisse
$\vec{y}$	$[m]$	Vecteur unitaire dans la direction 2 des coordonnées cartésiennes
$y$	$[m]$	Position en ordonnée
$\vec{z}$	$[m]$	Vecteur unitaire dans la direction 3 des coordonnées cartésiennes
$z$	$[m]$	Position en côte
$\alpha$	$[deg]$	Angle du câble par rapport à la verticale
$\gamma$	$[deg]$	Angle de rotation entre les repères ECEF et du satellite à un instant $t$
$\vec{\theta}$	$[deg]$	Vecteur unitaire dans la direction 2 des coordonnées sphériques
$\theta$	$[deg]$	Angle de colatitude en coordonnées sphériques
$\mu = GM_T$ $= 3.986004 \times 10^{14}$	$[m^3.s^{-2}]$	Paramètre Gravitationnel Standard de la Terre
$\nu$	$[deg]$	Anomalie vraie



<u>Symbole</u>	<u>Unité</u>	<u>Définition</u>
$\rho$	$[kg.m^{-3}]$	Masse volumique
$\rho_r$	$[\Omega.m]$	Résistivité électrique
$\vec{\varphi}$	$[deg]$	Vecteur unitaire dans la direction 3 des coordonnées sphériques
$\varphi$	$[deg]$	Angle de longitude en coordonnées sphériques
$\Omega$	$[deg]$	Longitude du nœud ascendant
$\omega$	$[deg]$	Argument du périégée

<u>Unités de mesure</u>	<u>Signification</u>
$A$	Ampère
$deg$ ou $^{\circ}$	Degré
$kg$	Kilogramme
$h$	Heure
$L$	Litre
$m$	Mètre
$km$	Kilomètre
$\mu m$	Micromètre
$min$	Minute
$\Omega$	Ohm
$s$	Seconde
$T$	Tesla
$nT$	Nanotesla
$V$	Volt



# 1 INTRODUCTION

## 1.1 Contexte actuel

Cet essai s'inscrit dans un projet universitaire dont le but est de développer un système de désorbitation pour les satellites en orbite terrestre basse. En effet, l'industrie spatiale étant en plein essor, de plus en plus de satellites sont envoyés en orbite basse. Ces orbites basses, dites LEO, pour *Low Earth Orbit*, font parties des orbites les plus demandées avec les orbites géostationnaires (dit GEO, *Geostationary Earth Orbit*). Les orbites LEO étant très utiles pour des missions d'observation de la Terre et pour la télécommunication, de plus en plus d'états, d'agences spatiales et de compagnies privées, telle StarLink, envoient leurs satellites sur ces orbites.

En 2000, on comptait plus de 7000 objets de plus de 10cm en orbite basse, dont seulement 400 étaient des satellites en opération [1]. En 2008, on comptait plus de 11 000 objets en orbite autour de la Terre [2], toutes orbites confondues, et aujourd'hui ce nombre s'élève à plus de 34 000 objets [3]. Parmi tous ces objets, très peu sont des satellites encore en opération. La grande majorité de ces objets, près de 25 000, sont des débris, qui peuvent être des étages de fusée encore en orbite, des satellites en fin de vie ou bien des débris plus petits issus de la collision de ces différents objets.

Le problème réside ici dans la taille et le nombre de ces débris. Sachant que chaque collision entre ces objets entraîne la création de nouveau débris de plus en plus petits, et donc une croissance exponentielle du nombre de débris, ceci a pour conséquence de rendre l'utilisation de certaines orbites basses très risquée voire complètement inexploitable.

L'accumulation de débris spatiaux autour de la Terre étant un problème assez important dans la mesure où l'on envoie toujours plus de satellites en LEO, les agences du monde entier commence à émettre des plans et des initiatives pour limiter le nombre de débris à long terme. En imposant un temps maximum durant lequel un satellite peut rester en orbite une fois sa mission terminée. Parmi ces agences, on peut compter la NASA ainsi que l'ESA, mais

également des organisations qui ne sont pas des agences telle que l'ONU. Mais certains pays, tel que le Canada, ne font pas partie des pays qui ont accepté ces mesures [2].

L'Inter-agence sur les débris spatiaux (IADC) existe donc dans le but de lier les agences entre elles afin de trouver des solutions communes [2].

Dans l'objectif de pallier ce problème, la NASA a émis une norme, que la plupart des agences du monde se sont accordées à suivre, qui concerne la fin de vie de toutes les missions spatiales. Une fois leurs missions terminées, les satellites doivent libérer rapidement leur orbite pour la rendre de nouveau disponible. Pour cela, le satellite doit pouvoir se désorbiter ou pouvoir se mettre sur une orbite dite de stockage, c'est-à-dire une orbite contrôlée qui ne gêne en rien les orbites fréquemment utilisées [1][2].

Pour les orbites de stockage, la NASA a défini en 1995 les zones où elles pouvaient se trouver, de telle sorte qu'elles n'interfèrent jamais avec les orbites avec le plus de demandes, c'est-à-dire les orbites LEO, GEO et les orbites en régime semi-géostationnaire [2].

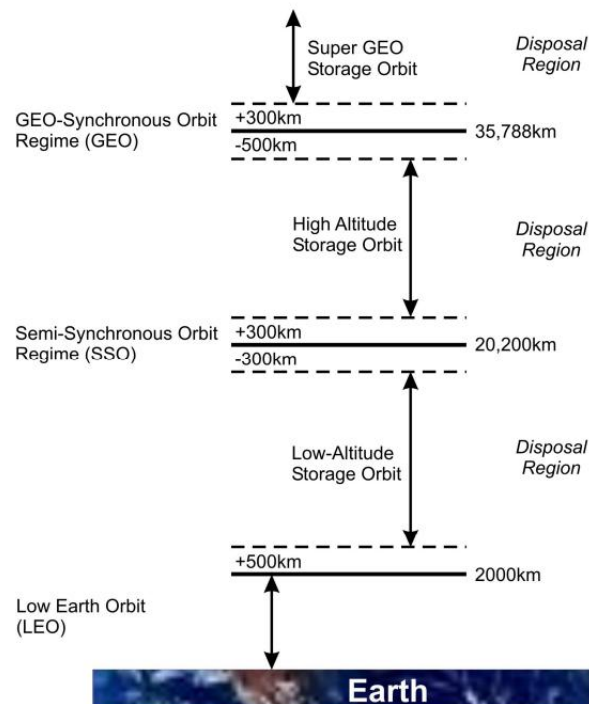


Figure 1-1 : Régions où peuvent avoir lieu les orbites de stockage [2]

Concernant la désorbitation des satellites, la norme prévoit deux moyens. Le premier, en utilisant un dispositif actif de désorbitation, tel que la propulsion chimique ou électrique. Le second, en utilisant un dispositif passif de désorbitation. Dans ce cas, le but du dispositif est essentiellement d'augmenter la traînée du satellite, pour que celui-ci perde petit à petit de la vitesse et donc de l'altitude. Ces dispositifs sont cependant peu répandus car encore en phase de recherche pour beaucoup [1][2]. Dans tous les cas, la désorbitation du satellite doit se faire en moins de 25 ans. Mais cette durée pourrait être réduite dans les années à venir.

Ce projet universitaire s'inscrit donc dans ce contexte et a pour but d'étudier une solution possible de dispositif passif d'augmentation de la traînée. Soit en augmentant sa traînée aérodynamique, soit en augmentant sa traînée électromagnétique. On se concentre ici, plus sur le second aspect et sur la technologie de câble électrodynamique.

## 1.2 Revue littéraire

Dès les années 1970, un concept de câble électrodynamique pour générer une traînée électromagnétique a été pensée [2]. Cependant, il faudra attendre la fin des années 90 pour voir les premiers prototypes partir dans l'espace.

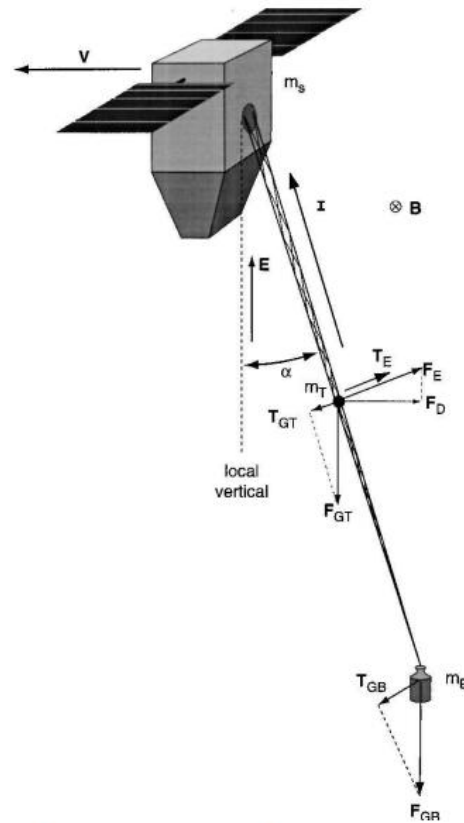
L'objectif de ce concept est d'utiliser à notre avantage le milieu spatial présent autour de la Terre et plus précisément le champ magnétique terrestre. En effet, en orbite basse plus particulièrement, les satellites se situent encore dans l'ionosphère, une couche de l'atmosphère terrestre caractérisée par une forte ionisation de l'air. En conséquence, les satellites "baignent" dans un plasma ambiant. De plus, dans cette zone, le champ magnétique terrestre est encore suffisamment fort. En conséquence, dans un tel environnement, un élément conducteur en mouvement, comme un câble en aluminium ou en cuivre par exemple, va générer un champ électrique  $\vec{E}$  selon la relation suivante :  $\vec{E} = \vec{V} \times \vec{B}$ . Ce champ électrique va à son tour générer un courant électrique au sein de ce conducteur et ce courant va à son tour réagir avec le champ magnétique ambiant pour créer une force électrodynamique, dite force de Lorentz [1][4].

Un des premiers gros projets qui a vu le jour dans le but de développer, d'améliorer et d'utiliser cette technologie a été le projet Terminator Tether [1]. Réaliser en 2000, le but de ce projet est d'utiliser un câble en aluminium qui serait déployé derrière le satellite (par rapport au sens de son mouvement), et dans lequel on imposerait un courant électrique, ou en laissant l'environnement induire un courant, pour générer cette force électrodynamique pour augmenter la traînée du satellite. En réalisant cela, un tel dispositif devrait donc permettre de désorbiter plus rapidement le satellite comparé à un satellite où on n'agirait pas dessus et où on laisserait le temps faire.

De façon plus détaillée, le Terminator Tether se compose comme suit : Un câble en aluminium est attaché à l'une de ses extrémités au satellite hôte. Son autre extrémité, quant à elle, est attachée à un ballaste de plusieurs kilogrammes, dont le but est de maintenir le câble le plus droit possible en utilisant notamment le gradient de gravité. La Figure 1-2 ci-dessous, montre un tel système. Sur son principe de fonctionnement, le câble va ensuite être parcouru par un courant électrique et le plasma ambiant va permettre de fermer le circuit électrique ainsi créé. Ce courant réagit ensuite avec le champ magnétique ambiant pour créer la force électrodynamique, qui est perpendiculaire au câble et au champ magnétique car la force de Lorentz est défini comme :

$$\vec{F} = L * \vec{I} \times \vec{B} \quad (1.1)$$

Où  $L$  est la longueur du câble parcouru par le courant  $I$ . La force ainsi générée est une force 3D dont le sens et la direction va dépendre du sens et de la direction du courant électrique qui parcourt le câble et du champ magnétique ambiant. Sachant que seule la composante de cette force qui est parallèle à la vitesse du satellite a réellement de l'impact sur la traînée générée.



**Fig. 4 Gradient and electrodynamic forces and torques on the tethered system.**

*Figure 1-2 : Détails du concept du Terminator Tether [1]*

Cependant, ce système possède plusieurs inconvénients majeurs et non négligeables. Le premier est la dimension d'un tel système. En effet, dans le projet du Terminator Tether, le câble utilisé est un câble de plusieurs kilomètres de long (généralement plus de 5 km). Cette longueur entraîne plusieurs problèmes. Avec une telle longueur, on augmente le risque d'entrer en collision avec un satellite qui serait sur une orbite très proche, ce qui peut être le cas en LEO. De plus, cette longueur peut entraîner des instabilités vis-à-vis de l'angle  $\alpha$  (comme décrit dans la Figure 1-2 ci-dessus). Du fait du gradient de gravité qui va s'exercer sur le câble et son ballaste, un angle trop proche de  $0^\circ$  ou de  $45^\circ$  va respectivement contraindre le câble à se maintenir à un angle de  $0^\circ$  ou de  $90^\circ$ . Dans les deux cas, le système devient alors inutile car ne génère plus de composante parallèle à la vitesse du satellite. Selon ce projet, l'efficacité est optimale lorsque  $\alpha = 35.26^\circ$  [1].

Le second inconvénient majeur est le poids d'un tel dispositif. Du fait de sa longueur et de sa composition en aluminium, qui est relativement lourd, et de la présence du ballaste, un tel dispositif peut facilement représenter plus de 5% de la masse totale du satellite [1].

Plus récemment, entre 2019 et 2022, un projet européen avait pour but de développer un kit de désorbitage basé sur la technologie du *low-work-function tether* (LWT). Ce projet, nommé E.T.PACK, pour *Electrodynamic Tether technology for PAssive Consumable-less deorbit Kit*, a été financé par la Commission Européenne. Le but de ce projet était de concevoir un module de désorbitage que l'on viendrait fixer sur un satellite avant son lancement en orbite. Ce module contiendrait à son bord tout l'équipement nécessaire pour assurer, une fois la mission du satellite terminée, la désorbitation du satellite en utilisant un câble électrodynamique pour générer une traînée électrodynamique. Ce câble, qui est d'une technologie similaire à celui du Terminator Tether, est cependant différent d'un point de vue énergétique [4].

Là où la technologie utilisée par le Terminator Tether nécessitait un dispositif actif pour capter et/ou émettre les électrons dans le plasma pour générer le courant, E.T.PACK utilise une nouvelle technologie, développée en 2012, qui fonctionne sur le même principe mais de façon complètement passive et autonome. Cette technologie, celle des *Low-Work-function Tether*, utilise un câble qui, contrairement à celui du Terminator Tether, est composé de plusieurs matériaux différents. Certains vont naturellement capter les électrons présents dans le plasma et d'autres vont naturellement émettre des électrons dans le milieu. Le plus gros avantage ici, c'est que les LWT ne nécessite pas d'être alimenté électriquement pour fonctionner [4].

La Figure 1-3 ci-après montre les différences que l'on peut retrouver entre les différents câbles électrodynamiques qui ont été pensés depuis 1965.



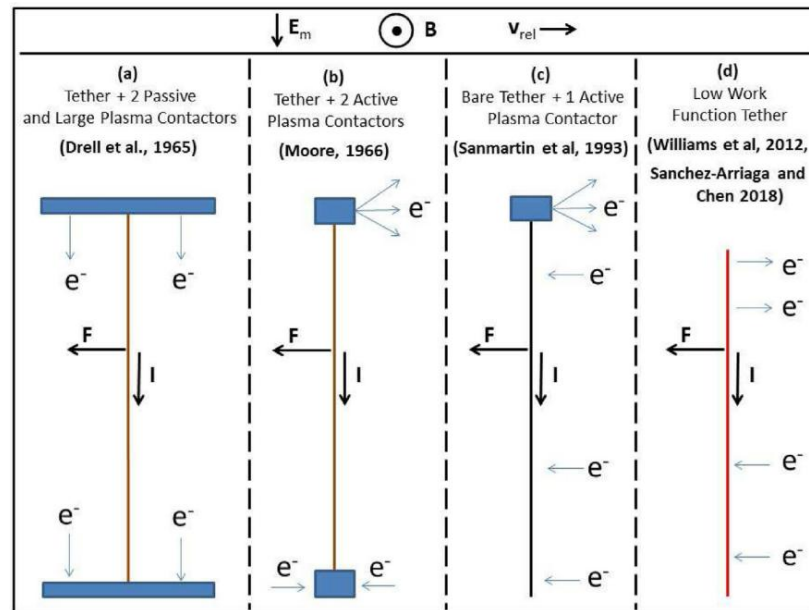


Figure 1-3 : Evolution des câbles électrodynamiques [4]

Les premiers câbles électrodynamique pensés, (a) et (b), avaient comme défaut majeur d'être très encombrants et de consommer beaucoup d'énergie. En effet, le câble (a) nécessitait de larges plaques métalliques, jouant le rôle de cathode et d'anode, pour capter et émettre de façon passive les électrons. Le câble (b) a réglé ce problème d'encombrement en remplaçant l'anode et la cathode passives par des dispositifs actifs, c'est-à-dire par des dispositifs qui nécessite d'être alimenté électriquement pour fonctionner. Bien que fonctionnel, ceci est difficilement implémentable dans un satellite où l'énergie est très limitée par la capacité de la batterie et des panneaux solaires. En fin de vie, ils sont beaucoup moins efficaces et délivrent donc beaucoup moins d'énergie qu'en début de mission.

Ce problème a en partie été réglé par le câble (c), qui est la technologie utilisée par le Terminator Tether. Ici, le câble capte lui-même les électrons du plasma ambiant et le système ne nécessite donc plus la présence d'un capteur d'électrons actif. Cependant, l'émission des électrons se fait encore par le biais d'un émetteur actif. Cette technologie, bien moins gourmande en énergie que celle du câble (b), doit tout de même être alimentée électriquement pour fonctionner.

La technologie LWT (câble (d)), vient régler ce dernier problème en proposant un système entièrement passif et qui nécessite donc aucun apport d'énergie. Ici, c'est le câble qui capte et émet les électrons grâce aux matériaux qui le compose et dont les propriétés physiques font qu'ils ont une tendance naturelle à capter ou émettre des électrons. Dans le projet E.T.PACK, le matériau sélectionné est le *Cl2A7 : e<sup>-</sup> electrider* [4], mais d'autres matériaux, tels que le *nitrogen doped ultra-nanocrystalline diamond* [4] et le *potassium-intercalated carbon nanotubes* [4] sont de très bons candidats.

Pour être aligné avec le secteur spatial et ses nombreuses contraintes, le projet s'est appuyé de l'ESA et des maîtres d'œuvres européens pour définir des requis que le système doit respecter. Parmi ces requis, on retrouve le fait que le système doit permettre la désorbitation, de façon complètement autonome, d'un satellite de 500 kg en SSO (*Sun-Synchronous Orbit*) à 850 km d'altitude et ce en moins de 24 mois. Par de façon complètement autonome, on signifie ici que le système ne doit pas utiliser les systèmes du satellite hôte. Le système doit ainsi être équipé de tous les composants dont il a besoin pour réussir sa mission de désorbitation [4].

Un autre requis est que le système doit pouvoir, avec quelques modifications, être capable de désorbiter un satellite ayant une masse comprise entre 200 et 1000 kg [4]. Ainsi, il a été défini que la masse du système ne devait pas excéder plus de 5% de la masse du satellite hôte, et que les dimensions nominales du système devraient être contenues dans un volume de 20 L et avoir un poids maximal de 25 kg [4].

Le projet E.T.PACK prévoit de lancer dans les années à venir un satellite équipé d'un de leur prototype pour un vol d'essai. Ce vol d'essai est prévu en plusieurs parties, qui vont permettre de tester différents modes d'utilisation du système. Ces modes sont représentés dans la Figure 1-4 ci-après.

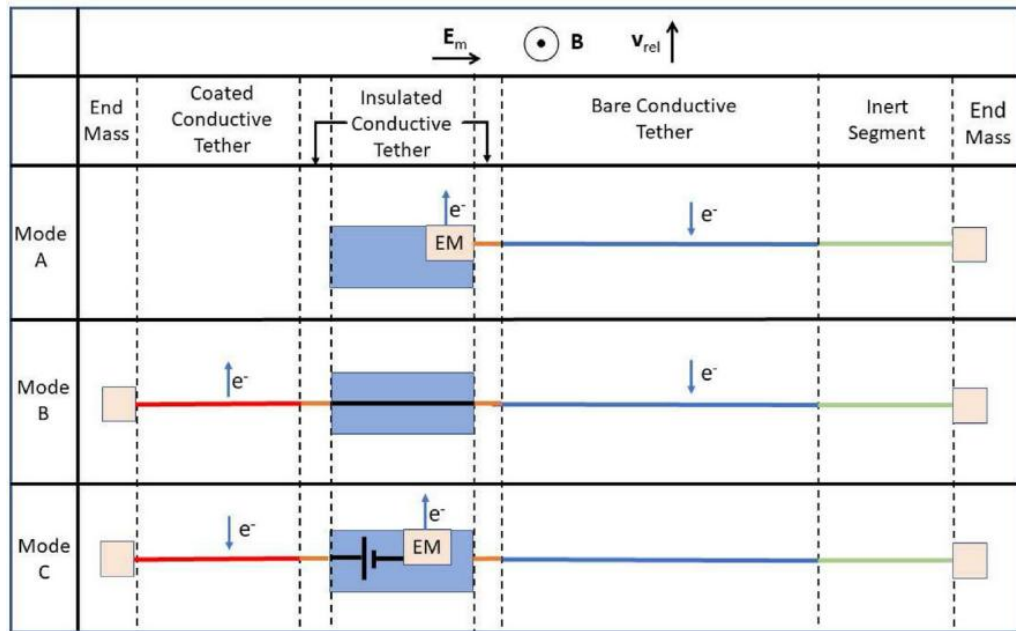


Fig. 4. Modes of operation of the Kit in a future demonstrations flight.

Figure 1-4 : Modes opératoire du kit de désorbitage E.T.PACK lors de son vol d'essai [4]

Dans ces essais, les dimensions des différentes parties du câble sont les suivantes : les parties orange, qui sont des câbles conducteurs isolés électriquement (le but étant de limiter la formation d'arc électrique entre le câble et le satellite et d'isoler les systèmes du satellite) ne devrait pas faire plus de quelques mètres. La partie bleue devrait faire 2 km de long. La partie verte ne devrait faire qu'un seul kilomètre de long. Cette partie du câble non conductrice est là pour plusieurs raisons. La première c'est de limiter la formation d'arc électrique entre le câble conducteur et le ballaste ainsi qu'avec le plasma ambiant. La seconde est pour des raisons de stabilité, cette longueur supplémentaire permet de garantir une certaine stabilité au câble. La partie rouge quant à elle ne devrait faire qu'un seul kilomètre également [4]. Le câble ne ferait alors plus que 3 km de long, sur sa partie la plus grande, comparé au plus de 5 km de long du Terminator Tether, cela limitera le risque de collision avec d'autres satellites.

Dans un rapport plus récent du projet E.T.PACK, on décrit avec plus de détails le prototype qui servira pour les tests durant le vol d'essai. Ce système, nommé EDT-D (EDT pour *Electrodynamique Tether*) sera légèrement différent sur sa conception que celui décrit plus

haut. Le but premier de la mission étant de tester la faisabilité d'un tel produit, ce système sera équipé d'un câble de seulement 500 m de long. Il sera également équipé de tous les équipements nécessaires pour garantir son autonomie vis-à-vis du satellite. Ainsi, il sera équipé de systèmes d'éjection, de tous les systèmes de navigation (ordinateur de bord, *torque rods*, *sunsensors*, ...), de panneaux solaires présents sur deux de ses faces, d'un système de communication complet pour pouvoir échanger directement avec le sol (télémétrie et commande) ainsi que du déployeur mécanique du câble [5].

Il sera totalement autonome et il sera équipé d'un dispositif ON/OFF sur le courant qui parcourt le câble. Le but ici est d'éviter les collisions avec d'autres débris ou satellites actifs. Le prototype fait 12U pour 24 kg et le produit final devrait en théorie faire les mêmes dimensions et le même poids mais cette fois-ci avec un câble de quelques kilomètres [5].

Par ailleurs, le câble utilisé ici n'est plus vraiment un câble à proprement parlé mais plus un ruban, cette forme offrant des avantages d'un point de vue compacité du système tout en gardant les mêmes propriétés électriques qu'un câble de section circulaire. Dans le EDT-D, le câble de 500 m est divisé en 3 parties. Une première de quelques mètres où le câble est entièrement isolé électriquement ; une deuxième de  $400\text{ m} \times 40\text{ }\mu\text{m}$  d'aluminium et enfin environ  $90\text{ m} \times 50\text{ }\mu\text{m}$  de ruban non conducteur [5].

Par le passé, plusieurs missions ont déjà été envoyées en orbite pour tester différentes technologies de câbles électrodynamiques. Des missions, telles que Tether Electrodynamic Propulsion CubeSat Experiment (TEPCE, 2019), la démonstration du Terminator Tape Module (2019), DEorbiting SpaceCraft using Electrodynamic Tethers (DESCENT, 2020) ou encore Miniature Tether Electrodynamics Experiment (MiTEE, 2021) ont déjà permis de montrer la faisabilité du système et son efficacité [5].

De plus, il a été montré lors de précédentes missions, qu'un système utilisant un câble électrodynamique peut produire de la poussée électrodynamique plutôt que la traînée. En effet, en fonction du sens du courant électrique qui parcourt le câble, la force électrodynamique

résultante n'aura pas le même sens ni la même direction [5]. Cela montre qu'avoir un contrôle sur le sens du courant permet de choisir si le système produit de la poussée ou de la traînée.

Dans les deux cas, ceci est un gros avantage pour un satellite. La capacité à générer de la traînée permet de se désorbiter plus rapidement une fois la mission terminée. Et la capacité à produire de la poussée permet plusieurs choses. Premièrement, cela permet d'augmenter la durée de vie du satellite sur des orbites très basses en produisant une force de poussée qui va venir contrer la traînée aérodynamique. Également, cela permet d'avoir un autre moyen de générer de la poussée en plus des différents propulseurs électriques et chimiques. Permettant dans certains cas d'emporter moins de carburant et donc moins de masse, ce qui peut laisser la place pour rajouter des instruments de mesure par exemple.

Par ailleurs, ces systèmes de câbles électrodynamiques utilisent principalement la présence du champ magnétique terrestre pour fonctionner, cela impose quelques limites aux systèmes. En effet, le champ magnétique perdant en intensité à mesure que l'on s'éloigne de la surface de la Terre, les câbles électrodynamiques perdent de leurs intérêts, car ils génèrent une force de plus en plus faible. En d'autres termes, ils ont besoin d'un câble de plus en plus long pour continuer à avoir la même efficacité.

De plus, le champ magnétique terrestre peut, dans une première approximation, être modélisé par un dipôle magnétique donc l'axe est décalé de  $11.5^\circ$  environ par rapport à l'axe de rotation de la Terre. Cela implique que sur certaines orbites, le système devient moins efficace. En effet, pour les orbites ayant une inclinaison supérieure à  $78.5^\circ$ , le satellite va, à un moment donné de la journée, passer dans une zone où le champ magnétique change de sens. Ce qui va induire un courant dans le sens contraire à la normale. Produisant ainsi de la poussée au lieu de la traînée, et inversement [1].

## 2 METHODOLOGIE

### 2.1 Objectifs du projet

L'objectif principal de ce projet est d'étudier et de développer à terme un système de désorbitation pour satellite en LEO en utilisant la technologie des câbles électrodynamiques pour augmenter sa traînée.

Dans cet objectif, on cherche ici à développer un outil permettant de calculer et d'estimer la traînée électromagnétique d'un tel système. Cet outil doit permettre à terme de pouvoir valider numériquement la faisabilité et la validité d'un tel système. Pour cela, le code sera réalisé sous Python.

L'outil doit donc être capable de réaliser plusieurs choses de façon indépendante et de pouvoir en même temps être lisible pour réaliser des calculs plus complexes. Tout d'abord, il doit être capable de calculer une force de Lorentz qui s'applique à un câble connaissant, ou non, le courant qui le parcourt, ainsi que le champ magnétique ambiant et la vitesse de déplacement du câble. L'outil doit pouvoir prendre en charge des formes de câbles simples, tels que des segments, ou bien plus complexes telles que des courbes paramétrées en trois dimensions. Il doit également pouvoir donner des propriétés physiques au câble, en pouvant choisir parmi différents matériaux utilisés dans l'industrie spatiale, tels que les alliages d'aluminium, le cuivre ainsi que les fibres de carbone.

Ensuite, puisque l'on s'intéresse ici à un câble qui serait attaché à un satellite en LEO, l'outil doit être capable de modéliser le champ magnétique terrestre au point où se situe le satellite et son câble. Pour cela, on utilise ici le module Python `ppgrf`, dont on détail les spécificités plus bas. A l'aide de ce module, l'outil doit donc pouvoir estimer la force de Lorentz générée par un câble connaissant l'endroit précis où se situe le satellite autour de la Terre.

Enfin, afin de pouvoir avoir des données concrètes sur les orbites, les coordonnées du satellite par rapport à la Terre ainsi qu'une date précise, l'outil doit pouvoir se lier à GMAT, un logiciel *open source* de la NASA, dont on détaille les spécificités plus bas. La connexion à GMAT doit

également permettre de valider les calculs analytiques. GMAT étant un outil puissant et reconnu de simulation d'orbite.

En résumé, l'objectif est ici de créer un outil numérique qui doit pouvoir calculer et retourner en sortie une force électromagnétique générée par un câble électrodynamique. Ceci doit pouvoir être fait dans des cas simples et plus complexes, ainsi que de pouvoir simuler des cas réels.

## 2.2 Python

Dans le cadre de ce projet, le code sera réalisé sous Python, car ce langage possède beaucoup de bibliothèques et de modules qui permettent d'avoir des outils simples mais néanmoins puissants dans plusieurs domaines (calculs matriciels, affichage, banques de données, *etc.*). Dans le cadre de ce projet, on utilise Python sous sa version 3.9.20.

Parmi les nombreuses bibliothèques disponibles, nous utiliserons plus particulièrement les bibliothèques suivantes :

- **Numpy** : c'est un outil mathématique très puissant qui permet de réaliser entre autres des calculs matriciels en permettant de créer facilement des vecteurs ainsi que des matrices ;
- **Datetime** : Dans cette bibliothèque, nous utiliserons seulement le module *datetime*, qui permet de définir une date ;
- **Astropy** : Cette bibliothèque est un outil très puissant de modélisation et de calcul d'orbite. Puisque, dans notre cas, on utilise GMAT pour la modélisation des orbites, *astropy* est principalement utilisé ici pour sa banque de données des constantes physique telles que la masse de la Terre ou son rayon ;
- **Math** : Cette bibliothèque contient plein de constantes et de fonctions mathématiques. Moins puissante que numpy mais fonctionnant pareille, on l'utilise principalement ici pour les fonctions trigonométriques et pour  $\pi$  ;

- **Ppigrf** : Cette bibliothèque contient une banque de données de mesures réelles du champ magnétique terrestre, ainsi qu'un modèle prévisionnel. On détaille plus cette bibliothèque dans la section suivante ;
- **Matplotlib** : Celle-ci est un puissant outil de représentation de graphiques et de courbes.

On regroupe dans le tableau ci-dessous les bibliothèques que l'on utilise ainsi que leur version utilisée.

*Tableau 2-1 : Liste des bibliothèques Python utilisées et leur version*

Bibliothèques Python	Version
<i>Numpy</i>	1.26.4
<i>Datetime</i>	Version courante
<i>Astropy</i>	5.3.4
<i>Math</i>	Version courante
<i>Ppigrf</i>	1.1.0
<i>Matplotlib</i>	3.9.2

## 2.3 Module ppigrf

*Ppigrf* [6] est un module Python qui contient les données passées de l'état du champ magnétique terrestre. Cette banque de données se base sur le modèle IGRF du champ magnétique terrestre. Mis à jour régulièrement, ce module modélise très bien le champ magnétique terrestre et donne des valeurs très proche de la réalité.

Le *International Geomagnetic Reference Field* (IGRF) est un modèle international de modélisation du champ magnétique terrestre. Il se base sur des données empiriques obtenues par observation depuis la Terre ainsi que de mesures réalisées par des satellites en orbite. Il se base également sur un modèle mathématique qui permet de prévoir l'état du champ magnétique



terrestre [7]. Mis à jour régulièrement, tous les 5 ans environ [6], la dernière mise à jour date du 19 novembre 2024 [7]. Le module *ppigrf* utilise donc les données de ce modèle.

Le module *ppigrf* est très simple d'utilisation. Il permet, connaissant la position ainsi que la date, de connaître la valeur du champ magnétique terrestre en ce point et à cet instant. Les coordonnées de la position peuvent être données soit dans les coordonnées géodésiques soit en coordonnées géocentriques. Les coordonnées géodésiques demandent de connaître la longitude, la latitude ainsi que la distance du centre de la Terre pour connaître la position. Alors que les coordonnées géocentriques utilisent un système de coordonnées sphériques usuel dont le repère cartésien de référence est le repère terrestre fixe. Ce repère également appelé repère ECEF, pour *Earth-Centred Earth-Fixed* [8], est décrit plus en détails dans la section 2.5.3.

Dans notre cas on utilisera les coordonnées géocentriques pour définir la position de nos points de calculs. Ce faisant, on contraint également en partie ici l'utilisation du repère ECEF pour nos calculs et le code.

Dans notre cas, l'utilisation du module *ppigrf* se fera via la fonction :

$$igrf\_gc(r, \theta, \varphi, t)$$

Qui, en entrée, prend les coordonnées sphériques  $(r, \theta, \varphi)$  du point, exprimées dans le repère géocentrique, ainsi qu'une date  $t$  exprimée au format *datetime*.

En sortie, la fonction *igrf\_gc()* renvoie les composantes radiale ( $B_r$ ), Sud ( $B_\theta$ ) et Est ( $B_\varphi$ ) du champ magnétique. Ces composantes sont renvoyées en nanotesla [nT] [6].

## 2.4 Logiciel GMAT

Développé par la NASA en 2007, *General Mission Analysis Tool* (GMAT) est un logiciel *open source* de calcul et de modélisation d'orbite autour de la Terre mais également dans tout le système solaire [9]. Ce logiciel très puissant, permet de modéliser de façon fidèle des orbites en considérant des aspects que l'on néglige habituellement à cause de leurs faibles impacts. En effet, il permet de prendre en compte dans les calculs les différentes forces de gravités issues des autres astres du système solaire, tel que le Soleil, la Lune ou encore Mars. Il permet également de prendre en compte la traînée aérodynamique et ce même pour des orbites terrestres à très hautes altitudes, telles que les orbites GEO. D'autres événements propres au spatial, tels que le temps de contact avec une station au sol, le temps de passage dans l'ombre d'un astre ou bien une manœuvre de changement d'orbite, sont également implémentés dans ce logiciel. Cependant, dans sa version actuelle, GMAT ne permet pas de modéliser le champ magnétique terrestre, ni celui des autres astres du système solaire tels que le champ magnétique du soleil, ou encore celui des planètes gazeuses.

Par ailleurs, GMAT possède une API qui permet de lier le logiciel à Python ou à Matlab. GMAT permettant également de modéliser une force qui s'exercerait sur un satellite. Force que l'on peut rentrer manuellement lors de la création d'une simulation. Connaissant cela, il est donc tout à fait possible, à l'aide d'un code Python, de modéliser et d'ajouter à une simulation GMAT une force électromagnétique. C'est là l'objectif de ce projet, écrire un code Python qui permet de modéliser une force électrodynamique pour ensuite implémenter cette force aux calculs de GMAT à l'aide de l'API.

## 2.5 Méthodes suivies

On décrit ici les méthodes de calcul et les réflexions logiques utilisées pour réaliser ce projet.

### 2.5.1 Hypothèses de calculs

Dans ce projet, on fait ici plusieurs hypothèses dans le but de simplifier le problème.

On suppose ici que les orbites considérées sont circulaires ou quasi circulaires. En réalité et d'après les lois de Kepler, les orbites sont des ellipses, dont la Terre occupe l'un des foyers (dans le cas d'orbites autour de la Terre). Les orbites circulaires ne sont qu'un cas particulier des orbites elliptiques, lorsque l'excentricité est nulle. Par ailleurs, la plupart des satellites présent en LEO ont des orbites quasi circulaire, dont l'excentricité est très proche de zéro. Ce qui justifie cette hypothèse.

Les orbites étant considérées comme circulaire, on peut donc considérer que la vitesse du satellite sur son orbite est constante et qu'elle est défini par :

$$V = \sqrt{\frac{\mu}{r^3}} \quad (2.1)$$

On considère également que la Terre est une sphère, car cette considération est prise par le module *ppgrf* lorsque l'on utilise la fonction *igrf\_gc()* [6].

On considère également que les orbites sont entièrement décrites par les élément képlériens  $a$ ,  $e$ ,  $i$ ,  $\Omega$ ,  $\omega$  et  $\nu$ .

### 2.5.2 Calcul de la force électromagnétique

Pour le calcul de la force de Lorentz, on distingue dans tout le projet deux cas distincts mais semblables en plusieurs points sur la méthode de calculs. Le premier cas est celui où l'on considère que le courant électrique qui parcourt le câble est naturellement induit par le mouvement du satellite dans le plasma ambiant autour de la Terre. On nomme ce cas, le cas induit. Pour ce cas, la méthode suivie est celle utilisée par le projet du Terminator Tether [1], que l'on détaille plus bas. Le second cas est celui où l'on impose le courant électrique qui

parcourt le câble sans prendre en compte le courant induit par le mouvement. On nomme ce cas, le cas imposé.

Dans les deux cas, afin de simplifier les calculs on définit un repère lié au satellite, que l'on définit plus en détails dans la section suivante. Ce repère, cartésien, est défini de telle sorte que l'axe  $\vec{x}$  soit dans le même sens que le vecteur vitesse du satellite. L'axe  $\vec{z}$  est quant à lui défini comme la verticale vers le haut par rapport à la surface de la Terre, ou en d'autres termes, colinéaire et de sens opposé au vecteur gravité.

Dans ce repère on définit les paramètres suivant en accord avec la Figure 2-1 :

- $\vec{V} = V\vec{x}$ , le vecteur vitesse du satellite ;
- $\vec{L}$ , le vecteur longueur du câble, défini dans le cas où le câble est un segment rectiligne, partant de l'extrémité libre du câble est pointant en direction du satellite ;
- $L$ , la longueur du câble et la norme de  $\vec{L}$  si elle a lieu d'être ;
- $\vec{I}$ , le vecteur intensité du courant, orienté de la même façon que  $\vec{L}$ , avec sa norme  $I$  l'intensité du courant qui parcourt le câble.

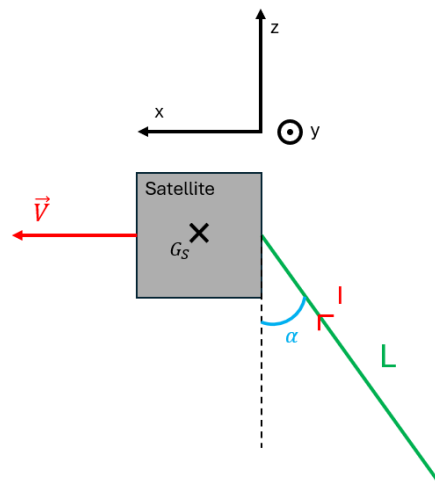


Figure 2-1 : Schéma du satellite et de son câble

Dans le cas induit, on suit la méthode suivante :

- On commence par calculer le champ électrique  $\vec{E}$  induit par le mouvement du satellite dans le plasma ambiant soumis au champ magnétique  $\vec{B}$  :

$$\vec{E} = \vec{V} \times \vec{B} \quad (2.2)$$

- Ensuite, on calcule la tension électrique  $U$  qui parcourt le câble conducteur et qui est induite par la présence du champ électrique  $\vec{E}$  autour du câble :

$$U = \vec{E} \cdot \vec{L} \quad (2.3)$$

- On définit ensuite l'intensité électrique  $I$  qui parcourt le câble connaissant la résistance électrique  $R$  du câble à l'aide de la loi d'Ohm, puis le vecteur intensité du courant associé :

$$I = \frac{U}{R} \quad (2.4)$$

$$\vec{I} = I * \frac{\vec{L}}{L} \quad (2.5)$$

- Enfin, on calcule la force de Lorentz qui s'exerce sur le câble dans ces conditions :

$$\vec{F} = L(\vec{I} \times \vec{B}) \quad (2.6)$$

Dans le cas imposé, puisque l'on connaît déjà l'intensité  $I$  du courant qui parcourt le câble, la méthode commence directement par la définition du vecteur  $\vec{I}$  au niveau de l'équation (2.5), puis on calcule la force de Lorentz ainsi générée à l'aide de l'équation (2.6).

Dans le cas, où l'on voudrait calculer la force de Lorentz générée par un câble ayant une forme quelconque, les méthodes décrites ci-dessus restent applicables si l'on discrétise le câble par des segments rectilignes.

### 2.5.3 Repères liés à la Terre

Dans ce projet, on utilise deux repères liés à la Terre, différents dans leur définition et dans leurs applications. Ces deux repères sont le repère terrestre ECEF et le système de coordonnées équatorial.

Le premier, le repère ECEF est défini comme suit [10] :

- C'est un repère cartésien ;
- L'origine se situe au niveau du centre de gravité de la Terre
- L'axe  $\vec{x}$  se situe dans le plan de l'équateur et pointe en direction du méridien de Greenwich ;
- L'axe  $\vec{y}$  est défini dans le plan de l'équateur tel que  $\vec{x} \times \vec{y} = \vec{z}$  ;
- L'axe  $\vec{z}$  correspond à l'axe de rotation de la Terre et il pointe vers le pôle Nord ;
- Ce repère tourne avec la Terre, il effectue donc une rotation complète en un jour dit sidéral, soit en 23 h 56 min 4 s [11].

Dans ce repère, on définit également un système de coordonnées sphériques usuel  $S(G_T, \vec{r}, \vec{\theta}, \vec{\varphi})$ , que l'on appellera simplement repère sphérique par la suite. Ce repère permet notamment de savoir la position du satellite par rapport à la Terre, mais c'est également le repère dans lequel les composantes du champ magnétique terrestre nous sont données par le module *ppigrf*. C'est également dans ce repère qu'est défini le repère du satellite présenté dans la section suivante.

Le second repère, le système de coordonnées équatorial est défini comme suit [12] :

- C'est un repère cartésien ;
- C'est un repère dit céleste, c'est-à-dire qu'il est centré sur la Terre mais qu'il ne tient pas compte de la rotation de celle-ci ;
- L'origine se situe au niveau du centre de gravité de la Terre
- L'axe  $\vec{x}$  se situe dans le plan de l'équateur et pointe en direction du point Vernal. Le point Vernal correspond à la direction du soleil au moment de l'équinoxe de printemps.

Cela correspond également au point d'intersection entre le plan de l'équateur et le plan de l'écliptique (plan de l'orbite terrestre), au moment où le Soleil passe de l'hémisphère céleste Sud à l'hémisphère céleste Nord (ou nœud ascendant). Ce qui correspond à l'équinoxe de printemps [13] ;

- L'axe  $\vec{y}$  est défini dans le plan de l'équateur tel que  $\vec{x} \times \vec{y} = \vec{z}$  ;
- L'axe  $\vec{z}$  correspond à l'axe de rotation de la Terre et il pointe vers le pôle Nord ;
- Ce repère tourne avec la Terre autour du Soleil, il effectue donc une rotation complète en 1 an.

Ce repère est principalement utilisé pour décrire les paramètres képlériens d'une orbite autour de la Terre, la longitude du nœud ascendant  $\Omega$  étant défini par rapport au point Vernal. On l'utilisera donc pour cela.

On peut également noter que le repère terrestre ECEF est en rotation dans le système de coordonnées équatoriales. Mais que le passage d'un repère à un autre est relativement compliqué puisqu'il dépend de la date et de l'heure à laquelle on veut effectuer le changement. Cependant, le logiciel GMAT permet de faire le passage de l'un à l'autre assez facilement. En effet, lorsque l'on fait une simulation sur GMAT, plusieurs repères sont déjà implémentés et considérés pour réaliser les calculs de la position du satellite en cartésien et en sphérique, ainsi que de l'état de ses paramètres képlériens. Les deux repères GMAT qui nous intéressent ici sont le repère *EarthMJ2000Eq* et *EarthFixed*, qui correspondent respectivement au système de coordonnées équatoriales et au repère terrestre ECEF. Le seul inconvénient, c'est que GMAT ne donne pas d'outil de conversion pour passer d'un repère à un autre mais donne simplement les valeurs de la position et des képlériens dans les différents repères et à la date de fin de la simulation.

### 2.5.4 Repère lié au satellite

Afin de faciliter les calculs de la force électromagnétique générée par un câble, on définit un repère lié au satellite. Repère dans lequel on pourra définir plus aisément le câble en lui-même : longueur, forme, position, *etc.*

On définit ici et on appelle repère du satellite le repère défini comme tel :

- C'est un repère cartésien en mouvement par rapport au repère terrestre ECEF ;
- L'origine du repère correspond au centre de gravité du satellite ;
- L'axe  $\vec{x}$  est défini de telle sorte que le vecteur vitesse du satellite soit orienté dans le même sens, *i.e.* tel que :  $\vec{V} = V\vec{x}$  ;
- L'axe  $\vec{z}$  est défini comme la verticale vers le haut par rapport à la surface de la Terre, mais également tel que  $\vec{z} = \vec{r}$ , où  $\vec{r}$  est le vecteur rayon des coordonnées sphériques défini dans le repère terrestre ECEF ;
- L'axe  $\vec{y}$  est défini tel que :  $\vec{x} \times \vec{y} = \vec{z}$

A un instant  $t$  donné, le satellite étant en un point de coordonnées sphériques  $(r, \theta, \varphi)$  dans le repère terrestre ECEF, on peut, en ce point et à cet instant, définir un changement de repère pour passer du repère ECEF au repère du satellite. A cet instant, on sait que le repère du satellite est une rotation d'un angle  $\gamma$  du repère défini par  $(G_S, \vec{r}, \vec{\theta}, \vec{\varphi})$ , selon la Figure 2-2 ci-dessous.

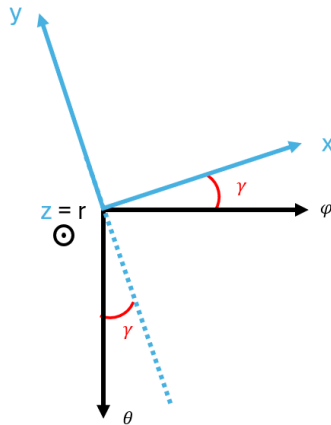


Figure 2-2 : Schéma de changement de repère entre le repère sphérique ECEF et le repère du satellite



Par définition, l'axe  $\vec{x}$  est colinéaire et de même sens que le vecteur vitesse du satellite sur son orbite, que l'on suppose circulaire. Par ailleurs, cette orbite est entièrement décrite par les six paramètres képlériens  $(a, e, i, \Omega, \omega, \nu)$ , et on sait que la vitesse est tangente en tout point à la trajectoire circulaire du satellite. On sait donc qu'au niveau du nœud ascendant, le vecteur vitesse  $\vec{V}$  du satellite fait un angle  $\gamma$  avec le plan de l'équateur qui correspond à l'inclinaison  $i$  de l'orbite du satellite. A l'inverse, au niveau du nœud descendant, on a  $\gamma = -i$ . De plus, au niveau du périgée et de l'apogée on a  $\gamma = 0^\circ$ . On voit donc ici que l'angle  $\gamma$  varie au cours d'une orbite. Variation qui semble être sinusoïdale. Par ailleurs, on sait qu'au périgée on a  $\nu = 0^\circ$ , qu'au nœud descendant on a  $\nu = 90^\circ$ , qu'à l'apogée on a  $\nu = 180^\circ$  et qu'au nœud ascendant on a  $\nu = 270^\circ$ . De là, on en déduit donc et on suppose par la suite que :

$$\gamma = -i \sin \nu \quad (2.7)$$

## 3 RESULTATS

### 3.1 Code de l'outil

Le code développé dans le cadre de ce projet se découpe en plusieurs morceaux distincts qui, pour certains, peuvent être utilisés de façon indépendante. Le code est ainsi composé de cinq modules ayant chacun un but bien précis : le calcul de la force électrodynamique ; la détermination du champ magnétique terrestre ; les calculs liés aux différents changements de repères ; la détermination des paramètres liés au câble et la détermination des paramètres physiques du matériau qui compose le câble.

#### 3.1.1 Module ForceElectro

Ce module permet de calculer la force de Lorentz résultante générée par un câble en mouvement baigné dans un champ magnétique et parcouru par un courant électrique. Il contient trois fonctions qui suivent toutes la méthodologie décrite dans la section 2.5.2, mais pour des câbles définis différemment.

La première fonction, contrairement aux deux autres, peut fonctionner de façon indépendante vis-à-vis des autres modules. Les deux dernières fonctions calculant directement la force de Lorentz générée par un câble soumis au champ magnétique terrestre. Elles utilisent donc le module *ppigrf* et une fonction définit dans le module *ChampMagnetique* décrit plus bas.

La première fonction est la fonction *ForceElectro( )*. Elle permet de calculer la force électromagnétique générée par un câble très simple défini comme un segment rectiligne. En entrée cette fonction prend une valeur du champ magnétique. Ce qui permet de pouvoir utiliser cette fonction avec des valeurs réelles du champ magnétique terrestre, grâce au module *ppigrf*, ou bien avec des valeurs quelconques. Cette fonction prend également en entrée la vitesse et la résistance électrique du câble. Également, et comme dans les autres fonctions de ce module, on spécifie en entrée la valeur du courant électrique qui parcourt le câble, soit en donnant une

valeur numérique pour calculer dans le cas imposé, soit en laissant la valeur sur *None* pour calculer dans le cas induit.

Par ailleurs, le fait que cette fonction utilise un câble rectiligne défini entre deux points, cette fonction peut être utilisée comme brique principale dans des codes où l'on chercherait à discrétiser un câble, ou bien parce que le câble est défini par morceaux. Ceci peut aussi être intéressant à utiliser si l'on souhaite décrire un câble réel, car en orbite, le câble aura tendance à osciller et à avoir une forme courbe plutôt que rectiligne.

Les deux autres fonctions sont les fonctions *Discret( )* et *Parametre( )*, qui fonctionnent exactement de la même manière, mais qui ont des arguments différents en entrée.

En argument d'entrée, la fonction *Discret( )* va prendre plusieurs choses. Tout d'abord la position du satellite, en sphérique, autour de la Terre dans le repère terrestre ECEF. On prendra également une date, défini au format *datetime*, ainsi que l'inclinaison de l'orbite du satellite et la position du satellite sur son orbite grâce à son anomalie vraie. Tout ceci servira essentiellement à définir la valeur du champ magnétique terrestre qui sera utilisé dans les calculs. On a ensuite besoin des informations sur le câble, notamment le courant qui le parcourt, sa résistance électrique, sa vitesse de déplacement, le nombre de point de discrétisation ainsi de la définition du câble en lui-même. En effet, tout comme la fonction *ForceElectro( )*, cette fonction fait les calculs pour un câble rectiligne défini entre deux points. Mais ici, on vient rajouter des points sur le câble pour inclure dans le calcul les variations du champ magnétique terrestre le long du câble et ainsi avoir une plus grande précision.

Pour la fonction *Parametre( )*, les arguments d'entrées sont quasiment les mêmes. On retrouve la position du satellite, données en sphériques dans le repère terrestre ECEF, la date donnée au format *datetime*, l'inclinaison de l'orbite, l'anomalie vraie du satellite, la résistance électrique du câble, le courant qui le parcourt et la vitesse de déplacement. Cependant, cette fonction fait le calcul de la force de Lorentz mais pour des câbles définis par des courbes paramétrées en 3D. Ce qui permet ici de définir un câble de toutes les formes que l'on veut. Pour cela, on a

donc besoin en entrée de trois listes qui contiennent respectivement les abscisses, les ordonnées et les côtes des points qui composent le câble. Étant donnée la définition complexe du câble utilisée ici, le calcul se fait de façon infinitésimale entre chaque nœud qui définit le câble.

Dans tous les cas, ces fonctions permettent de calculer la force de Lorentz. Des codes de tests qui utilisent ces fonctions et qui montre leur fonctionnement sont décrits plus bas dans la section 3.2. De plus, le code de ces trois fonctions est présenté en ANNEXE I (*cf.* p.45) et on y décrit plus en détails leur fonctionnement.

### 3.1.2 Module ChampMagnétique

Ce module permet de connaître la valeur du champ magnétique terrestre en un point donné et à une date donnée. Pour cela, il utilise le module *ppigrf*, avec la fonction *igrf\_gc()*. Cette fonction fonctionne dans le repère terrestre ECEF et utilise un système de coordonnées sphériques. En entrée, elle prend les coordonnées du point ainsi qu'une date, ici exprimée au format *datetime*. En sortie, on obtient les composantes du champ magnétique terrestre. Comme la fonction *igrf\_gc()* donne les valeurs du champ magnétique en nanotesla [nT], dans notre fonction on réalise en plus la conversion en tesla [T] pour que ce soit plus simple d'utilisation par la suite.

Ce module fonctionne de façon indépendante vis-à-vis des autres modules de ce projet.

Le code de ce module est montré dans l'ANNEXE II (*cf.* p.50).

### 3.1.3 Module Satellite

Ce module contient plusieurs fonctions qui permettent de réaliser les différents changements de repère entre le repère terrestre ECEF et celui du satellite, pour des points ainsi que pour des vecteurs.

La première fonction, la fonction  $Sat2Earth()$ , et sa réciproque  $Earth2Sat()$ , permettent toutes deux de réaliser un changement de coordonnées pour un point dans le repère du satellite vers le repère terrestre ECEF et inversement.

En paramètres d'entrée, la fonction  $Sat2Earth()$  va prendre les coordonnées  $(x, y, z)$  du point, définis dans le repère du satellite, ainsi que l'inclinaison de l'orbite du satellite, son anomalie vraie et la position du satellite dans le repère terrestre ECEF. On obtient donc à la fin, les coordonnées  $(r, \theta, \varphi)$  du point, dans le repère terrestre ECEF.

Pour la fonction  $Earth2Sat()$ , c'est l'inverse. On prend en entrée, en plus de l'inclinaison de l'orbite, de l'anomalie vraie et de la position du satellite dans ECEF, les coordonnées  $(r, \theta, \varphi)$  du point dans le repère terrestre ECEF. Et on obtient en sortie les coordonnées  $(x, y, z)$  du point dans le repère du satellite.

Les deux autres fonctions, la fonction  $Sat2Earth\_cm()$  et sa réciproque  $Earth2Sat\_cm()$ , fonctionnent exactement de la même manière. Elles possèdent les mêmes arguments d'entrée, à la seule différence qu'elles n'ont pas besoin de la position du satellite dans le repère ECEF, car ces fonctions réalisent les changements de repère pour des vecteurs.

Pour ces quatre fonctions, on se base sur la Figure 2-2 pour réaliser la matrice  $R_{Earth \rightarrow Sat}$  de changement de bases ci-dessous :

$$R_{Earth \rightarrow Sat} = \begin{bmatrix} 0 & -\sin \gamma & \cos \gamma \\ 0 & -\cos \gamma & -\sin \gamma \\ 1 & 0 & 0 \end{bmatrix} \quad (3.1)$$

Ce module fonctionne de façon indépendante vis-à-vis des autres modules de ce projet.

Le code de ce module est montré dans l'ANNEXE III (cf. p.51).

### 3.1.4 Module Câble

Ce module contient plusieurs fonctions qui permettent de connaître des informations sur le câble si l'on connaît ses dimensions ainsi que les caractéristiques du matériau qui le compose.

Ainsi, ce code contient des fonctions qui permettent de connaître la résistance électrique du câble, son volume ainsi que sa masse. Il contient également une fonction qui permet de tracer dans un repère 3D et ainsi, avoir une représentation d'un câble qui serait défini par une liste de points.

Enfin, ce module possède également deux fonctions qui permettent de transformer un câble défini par une matrice contenant uniquement les coordonnées des points aux extrémités du câble en trois listes contenant respectivement les abscisses, les ordonnées et les côtes des points discrétisés du câble, et inversement.

Ce module fonctionne de façon indépendante vis-à-vis des autres modules de ce projet.

Le code de ce module est montré dans l'ANNEXE IV (*cf.* p.54).

### 3.1.5 Module Matériaux

Ce module contient plusieurs *class* dont chacune représente un matériau différent. Pour chacune de ces *class*, on définit plusieurs fonctions qui permettent d'obtenir différents paramètres et propriétés propre au matériau, comme sa masse volumique, sa résistivité électrique ou encore son module de Young et des limites à la rupture.

Parmi les matériaux intégrés, on peut compter le cuivre à l'état pure, les aluminium 6061-T6 et 7075-T6, qui sont tous trois des matériaux fréquemment utilisés dans l'industrie spatiale.

Ces différentes *class* sont montrées dans l'ANNEXE V (*cf.* p.57).

## 3.2 Code de validation et d'application de la méthode :

### 3.2.1 Validation de la méthode

Ce premier code de test a pour but de valider la méthode utilisée. On prend ici un cas simple et on comparera les résultats donnés par le code avec ceux d'un calcul analytique.

Pour ce test, les différents paramètres sont :

- Une vitesse  $\vec{V} = \begin{bmatrix} 7.5 \\ 0 \\ 0 \end{bmatrix} m/s$  ;
- Un champ magnétique  $\vec{B} = \begin{bmatrix} 0 \\ 0.001 \\ 0 \end{bmatrix} T$  ;
- Une intensité  $I = 1.5 A$  (pour le cas imposé) ;
- Un câble de longueur  $L = 40 m$  ;
- Le câble fait un angle  $\alpha = 30^\circ$  avec la verticale, selon le schéma montré en Figure 2-1 ;
- On a donc :  $\vec{L} = \begin{bmatrix} 20\sqrt{3} \\ 0 \\ 20 \end{bmatrix} m$  ;
- On utilise un câble en aluminium 7075 T6, de  $D = 5 mm$  de diamètre, on a donc :
  - $\rho = 2810 kg/m^3$  ;
  - Résistivité électrique  $\rho_r = 5.15 * 10^{-8} \Omega.m$  ;
  - Surface de section du câble :

$$S = \pi \left( \frac{D}{2} \right)^2 = \pi \left( \frac{5 * 10^{-3}}{2} \right)^2 = 1.963 * 10^{-5} m^2 \quad (3.2)$$

- Résistance électrique du câble :

$$R = \rho_r \frac{L}{S} = 5.15 * 10^{-8} * \frac{40}{1.963 * 10^{-5}} = 0.105 \Omega \quad (3.3)$$

Dans le cas induit, on a donc :

$$\vec{E} = \vec{V} \times \vec{B}$$

*i.e.* :

$$\vec{E} = \begin{bmatrix} 7.5 \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0.001 \\ 0 \end{bmatrix}$$

*i.e.* :

$$\boxed{\vec{E} = \begin{bmatrix} 0 \\ 0 \\ 7.5 * 10^{-3} \end{bmatrix} V.m^{-1}} \quad (3.4)$$

Puis, on a :

$$U = \vec{E} \cdot \vec{L}$$

*i.e.* :

$$U = \begin{bmatrix} 0 \\ 0 \\ 7.5 * 10^{-3} \end{bmatrix} \cdot \begin{bmatrix} 20\sqrt{3} \\ 0 \\ 20 \end{bmatrix}$$

*i.e.* :

$$\boxed{U = 0.15 V} \quad (3.5)$$

Puis on a :

$$\vec{I} = \frac{U \vec{L}}{R L}$$

*i.e.* :

$$\vec{I} = \frac{0.15}{0.105 * 40} * \begin{bmatrix} 20\sqrt{3} \\ 0 \\ 20 \end{bmatrix}$$



*i.e.* :

$$\vec{I} = \begin{bmatrix} \frac{5\sqrt{3}}{7} \\ 0 \\ 5 \\ \frac{5}{7} \end{bmatrix} A \quad (3.6)$$

Et :

$$I = \|\vec{I}\| = \frac{10}{7} \approx 1.4286 A$$

Finalement, on a donc :

$$\vec{F} = L(\vec{I} \times \vec{B})$$

*i.e.* :

$$\vec{F} = 40 * \begin{bmatrix} \frac{5\sqrt{3}}{7} \\ 0 \\ 5 \\ \frac{5}{7} \end{bmatrix} \times \begin{bmatrix} 0 \\ 0.001 \\ 0 \end{bmatrix}$$

*i.e.* :

$$\vec{F} = \begin{bmatrix} -0.02857 \\ 0 \\ 0.04949 \end{bmatrix} N \quad (3.7)$$

Dans le cas imposé, on a :

$$\vec{I} = I \frac{\vec{L}}{L}$$

*i.e.* :

$$\vec{I} = \frac{1.5}{40} * \begin{bmatrix} 20\sqrt{3} \\ 0 \\ 20 \end{bmatrix}$$

*i.e.* :

$$\boxed{\vec{I} = \begin{bmatrix} \frac{3\sqrt{3}}{4} \\ 0 \\ 0.75 \end{bmatrix} A} \quad (3.8)$$

Finalement, on a donc :

$$\vec{F} = L(\vec{I} \times \vec{B})$$

*i.e.* :

$$\vec{F} = 40 * \begin{bmatrix} \frac{3\sqrt{3}}{4} \\ 0 \\ 0.75 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0.001 \\ 0 \end{bmatrix}$$

*i.e.* :

$$\boxed{\vec{F} = \begin{bmatrix} -0.03 \\ 0 \\ 0.05196 \end{bmatrix} N} \quad (3.9)$$

A l'aide du code mis en ANNEXE VI (*cf.* p.64), on teste avec la fonction *ForceElectro()* le calcul de la force électrodynamique dans un cas simple comme celui-ci. Les résultats obtenus sont ceux montrés dans la Figure 3-1 ci-dessous.

```

Cas induit : F =

[-0.02859459  0.          0.04952729] N
Norme = 0.05718918665515399

Cas imposé : F =

[-0.03        0.          0.05196152] N
Norme = 0.05999999999999999

```

Figure 3-1 : Résultats 1er code de test

On regroupe dans le tableau ci-dessous, les différentes valeurs obtenues analytiquement et numériquement dans les deux cas considérés.

Tableau 3-1 : Tableau comparatif des résultats du 1er code de test

		Calculs analytiques	Calculs numériques	Erreur relative
Cas induit				
Force de Lorentz	$F_x$	$-0.02857\text{ N}$	$-0.02859459\text{ N}$	0.09 %
	$F_y$	$0\text{ N}$	$0\text{ N}$	0 %
	$F_z$	$0.04949\text{ N}$	$0.04952729\text{ N}$	0.075 %
Norme		$0.057144\text{ N}$	$0.057189187$	0.08 %
Cas imposé				
Force de Lorentz	$F_x$	$-0.03\text{ N}$	$-0.03\text{ N}$	0 %
	$F_y$	$0\text{ N}$	$0\text{ N}$	0 %
	$F_z$	$0.05196\text{ N}$	$0.05196152\text{ N}$	0.003 %
Norme		$0.0599986$	$0.059999999\text{ N}$	0.002 %

On remarque ici que les erreurs relatives sont quasiment toutes nulles, ce qui montre ici que les résultats analytiques et numériques sont quasiment identiques. Ce qui montre que le code réalisé ici, et plus particulièrement la fonction *ForceElectro()*, suivent bien la méthodologie que l'on souhaitait suivre et qu'ils donnent les résultats attendus.

### 3.2.2 Premier code de test :

Dans ce premier code de test, on cherche à montrer la validité et la cohérence des différentes fonctions de calcul de la force électromagnétique présentes dans le module ForceElectro. Pour cela, on va comparer les résultats de ces fonctions, dans le cas induit et dans le cas imposé, pour un câble en orbite autour de la Terre.

Pour ce test, on définit donc un câble en aluminium 6061-T4 de 4 km de long et de 5 mm de diamètre. On suppose également ici que le câble est un segment rectiligne compris dans le plan  $xz$  du satellite et qu'il fait un angle de  $\alpha = 30^\circ$  avec la verticale du satellite.

Pour le satellite, on suppose qu'il est en orbite circulaire à 800 km d'altitude. On suppose également que le satellite se situe sur son orbite au point défini par les éléments képlériens suivants, définis dans le système de coordonnées équatoriales :

$$(a, e, i, \Omega, \omega, \nu) = (7178.05 \text{ km}, 6.05e^{-5}, 25^\circ, 44^\circ, 179^\circ, 106.8^\circ) \quad (3.10)$$

Ces résultats ont été obtenus à la suite d'une simulation réalisée sur GMAT, dont les détails sont présentés en ANNEXE IX (*cf.* p.70). Simulation dont la date de fin est le 15 juillet 2020 à 15h20 UTC. C'est la date que l'on prendra pour nos calculs du champ magnétique terrestre. On se sert également des résultats obtenus via GMAT pour obtenir la position du satellite dans le repère terrestre ECEF, car GMAT nous donne directement la conversion entre différents repères terrestre usuels. Les résultats de cette simulation sont présentés en ANNEXE IX (*cf.* p.70).

A la suite de cette simulation, on obtient donc les coordonnées en sphérique dans le repère terrestre ECEF, ainsi que l'inclinaison de l'orbite et l'anomalie vraie du satellite dans ce même repère. On a donc :

$$\begin{cases} (r, \theta, \varphi) = (7178.2 \text{ km}, 114^\circ, 168^\circ) \\ i = 25^\circ \\ \nu = 180^\circ \end{cases} \quad (3.11)$$

Ce sont ces données qui sont ensuite utilisées dans notre code de comparaison.

A l'aide de ces données, et du code *Test\_Cable\_rectiligne* (cf. ANNEXE VII p.65), on obtient les résultats suivants :

Comparaisons des fonctions :				
Cas induit :				
	F_x	F_y	F_z	F
ForceElectro() :	[-4.523896521	-2.267499649	2.611949492]	5.694688274 N
Discret() :	[-4.532254013	-2.308006958	2.616774834]	5.719766874 N
Parametre() :	[-4.532329729	-2.308376689	2.61681855 ]	5.71999607 N
Cas imposé :				
	F_x	F_y	F_z	F
ForceElectro() :	[-0.105368854	-0.052813728	0.060836521]	0.132638484 N
Discret() :	[-0.105466138	-0.053707621	0.06089269 ]	0.133099717 N
Parametre() :	[-0.105467019	-0.053715776	0.060893198]	0.133103938 N

Figure 3-2 : Résultats du code *Test\_Cable\_rectiligne*

On peut voir ici plusieurs choses. Tout d'abord, c'est que les trois fonctions donnent toutes des résultats très proches les uns des autres, montrant ici que les trois fonctions calculent bien la même chose. On peut également observer, que lorsque l'on prend en compte la variation du champ magnétique le long du câble, comme avec les fonctions *Discret()* et *Parametre()*, le résultat s'affine et devient plus précis. En contrepartie, on perd en temps de calcul. Ces deux fonctions étant beaucoup plus lentes que la fonction *ForceElectro()*, alors que les résultats sont sensiblement les mêmes. Pour une première approximation, la fonction *ForceElectro()* suffit amplement. Mais pour plus de précision, notamment si on veut prendre en compte la variation du champ magnétique le long du câble, il est préférable d'utiliser les fonctions *Discret()* et *Parametre()*. Cependant, pour des câbles non rectilignes, seule la fonction *Parametre()* est utilisable.

### 3.2.3 Deuxième code de test :

Jusqu'à présent, on considérait que le câble avait une forme relativement simple et qu'il était assimilable à un segment de droite. Or, en réalité le câble a peu de chance d'être parfaitement droit et ce notamment à cause du gradient de gravité qui s'exerce sur le câble. Ce qui fait que le câble aura tendance à osciller autour d'une position d'équilibre au cours d'une orbite. De plus, dans le cadre de la recherche lié à ce projet, il peut être utile de tester des formes de câble

plus complexes mais moins encombrantes spatialement. C'est ce qu'on on montre ici en utilisant la fonction *Parametre( )* sur des exemples de câbles définis par des courbes paramétrées. Pour cela on utilise le code *Test\_Courbe\_parametree* présenté en ANNEXE VIII (cf. p.67).

Pour ces exemples, on utilise les mêmes coordonnées décrites par l'équation (3.10) pour la position du satellite. La date est également la même que dans la section précédente. Pour le câble, on considère qu'il est fait d'aluminium 2024-T3 et qu'il a un diamètre de 5 mm.

Dans le premier exemple, on considère que le câble est en forme de cercle, de 2 km de diamètre, derrière le satellite (cf. Figure 3-3). Il a donc une longueur de 6,2 km environ. Ce câble est paramétré pour faire un angle de  $\alpha = 30^\circ$  avec la verticale du satellite. Avec un tel câble on obtient les résultats présentés dans la Figure 3-4.

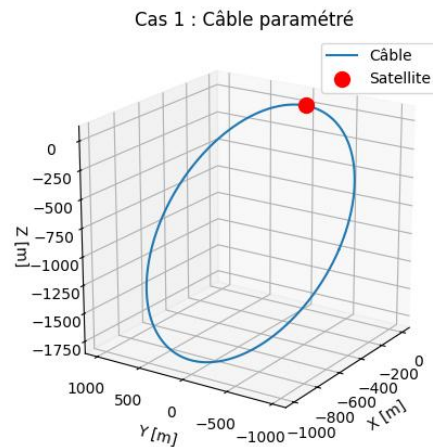


Figure 3-3 : Câble circulaire

```
Cas 1 : Câble paramétré

Cas induit :
  F   = [-0.00010499  0.0002256  0.00011728] N
 ||F|| = 0.0002750892458507528 N

Cas imposé :
  F   = [ 0.00076837 -0.00165105 -0.00085832] N
 ||F|| = 0.0020132260383787675 N
```

Figure 3-4 : Résultats du code *Test\_Courbe\_Parametree* pour le premier cas

On observe ici plusieurs choses. Tout d'abord, malgré la longueur du câble, qui est plus grande que dans les tests de la section précédente, la force électromagnétique générée est très faible. On est ici dans l'ordre des  $10^{-4}$  Newton pour le cas induit et des  $10^{-3}$  Newton dans le cas imposé. De telles forces auront donc un très faible impact sur le satellite. Par ailleurs, on observe également ici que dans le cas imposé, la composante de la force colinéaire à la vitesse de déplacement est positive. Ce qui signifie, qu'ici, on produit de la poussée et non de la traînée.

Pour le deuxième exemple, on définit un câble faisant là aussi un angle de  $30^\circ$  avec la verticale du satellite, mais cette fois-ci défini par morceaux. On le définit ici à l'aide de quatre segments de tel sorte que le câble soit fermé et qu'il forme un carré derrière le satellite, comme montré dans la Figure 3-5. Les résultats sont montrés dans la Figure 3-6.

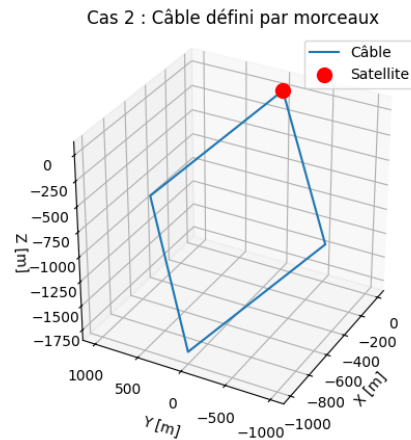


Figure 3-5 : Câble défini par morceaux

```
Cas 2 : Câble défini par morceaux

Cas induit :
F = [-4.54478568e-05  9.46411758e-05  4.91932641e-05] N
||F|| = 0.00011594152436030605 N

Cas imposé :
F = [-0.00050554  0.00105274  0.0005472 ] N
||F|| = 0.001289668184964631 N
```

Figure 3-6 : Résultats du code `Test_Courbe_Parametree` pour le deuxième cas

On observe ici que, similairement au premier cas, la force générée est très faible. Cela est en fait dû à la forme fermée du câble. En effet, bien qu'ici les résultats soient très proches de ceux du cas précédent, notamment parce que les câbles ont des formes et des dimensions très proches, ils restent très faibles. Comme on va le montrer avec le troisième exemple, si on est soumis à un champ magnétique uniforme, un circuit fermé produit une force Lorentz nulle. De façon simpliste, la force produite par la partie gauche du circuit s'annule avec celle produite par la partie droite. Ceci explique la faible force générée dans les deux premiers cas. Le fait que ces forces ne soient pas nulles s'explique ici par le fait qu'on prend en compte les variations spatiales du champ magnétique. Ainsi, celui-ci n'est pas parfaitement symétrique par rapport au câble, ce qui se traduit par la création d'une force nette.

Pour le dernier exemple, on réutilise le câble défini précédemment, mais cette fois-ci, le câble étant assez simple, on va utiliser la fonction *ForceElectro()* afin de pouvoir considérer un champ magnétique uniforme sur toute la longueur du câble. Pour cela, on utilise une boucle *for* et on obtient les résultats présentés dans la Figure 3-7.

```
Cas 3 : Câble soumis à un champ magnétique uniforme
Cas imposé :
F    = [-4.85878427e-16 -1.79821707e-16 -2.54313172e-17] N
||F|| = 5.187103663887181e-16 N
```

Figure 3-7 : Résultats du code *Test\_Courbe\_Parametree* pour le troisième cas

On peut voir ici que, comme on pouvait s'y attendre, la force générée est extrêmement faible. Cette valeur de l'ordre de  $10^{-17}$  N pouvant ici être considérée comme un zéro numérique. Le zéro exact n'étant pas atteint ici à cause des approximations de calculs et par le fait que la discrétisation du câble ne soit pas exactement la même sur toute la longueur du câble.



## 4 DISCUSSION

### 4.1 Limites du code :

Le code développé dans le cadre de ce projet permet de répondre, en partie, aux attentes que l'on s'était fixé. En effet, on arrive bien à calculer la force électromagnétique générée par un câble électrodynamique. Cependant, l'outil développé ici possède quelques limitations.

Tout d'abord, puisque tous les calculs utilisant le champ magnétique terrestre se basent sur la base de données du module *ppigrf* et ne permettent donc pas de faire des simulations pour des dates futures. Bien que le module *ppigrf* possède un modèle prédictif du champ magnétique terrestre, celui-ci est peu fiable à mesure que l'on s'éloigne des dernières dates de mesures effectuées, puisqu'on ne peut pas prévoir les éruptions solaires et les mouvements des pôles magnétiques qui ont tous deux un impact sur le champ magnétique terrestre. En dépit de cela, on peut tout de même estimer les variations futures du champ magnétique solaire en se basant sur les données récoltées durant les derniers cycles solaires.

De plus, on fait également dans le développement de cet outil, beaucoup d'approximations et d'hypothèses qui limitent son utilisation. La première limitation, c'est qu'on ne considère que des orbites circulaires. Bien que ce soit le type d'orbite le plus représenté en LEO, certains satellites ont des orbites elliptiques et l'outil tel que présenté ne permet pas, sans modifications, de traiter leur cas. La deuxième est liée au changement de repère que l'on effectue entre le repère terrestre ECEF et le repère lié au satellite. En effet, bien que ce changement de repère soit exact si on le fait au niveau de centre de masse du satellite, il devient de plus en plus approximatif à mesure que l'on s'éloigne de ce point. La dernière est liée au changement de repère entre le repère terrestre ECEF et le système de coordonnées équatoriales terrestres. Bien que le mouvement entre ces deux repères soit simple, connaître la position de l'un par rapport à l'autre est beaucoup plus compliqué, du fait qu'ils tournent à des vitesses différentes. Le changement de coordonnées entre ces deux repères n'est donc pas traité par cet outil et il faut utiliser le logiciel GMAT pour le faire.

Enfin, le temps de calculs nécessaire lors de l'utilisation des fonctions *Discret()* et *Parametre()* du module ForceElectro peut être relativement long. En effet, ces fonctions utilisant plusieurs boucles d'itérations qui dépendent du nombre de points de discrétisation, un trop grand nombre de points rendra le temps de calcul long. Or, en limitant le nombre de points de discrétisation, on perd en précision sur la définition d'un câble ayant une forme complexe. Ce temps de calcul devient particulièrement préoccupant si on s'intéresse à une orbite complète d'un satellite. Jusqu'ici, on réalisait les calculs en un seul point d'une orbite, et les temps de calculs sont déjà de l'ordre de la minute. Cependant, il peut être intéressant de connaître la force de Lorentz générée sur la totalité d'une orbite. Orbite qui sera discrétisée en plusieurs points et à chacun de ces points on devra réaliser un calcul qui peut prendre une minute. Rendant ainsi le temps de calcul à une heure pour une orbite discrétisée avec seulement 60 points, ce qui est peu.

## 4.2 Perspectives et pistes d'améliorations :

Parmi les points abordés dans le paragraphe précédent, plusieurs peuvent être en partie réglée en réalisant les améliorations suivantes.

Tout d'abord, le temps de calcul peut être réduit en optimisant le code. Dans l'état actuel, beaucoup de calcul ne sont pas réalisés de façon vectorielle et les boucles d'itération ne sont pas écrites pour être performantes mais plutôt pour être compréhensibles.

Une autre amélioration envisageable serait la prise en compte des orbites elliptiques. Cependant, cela pourrait impliquer la refonte de la définition du repère du satellite. Dans notre cas, on se basait sur le fait que sur une orbite circulaire la vitesse de déplacement du satellite est purement tangente à la trajectoire. Or, sur une orbite elliptique, la vitesse de déplacement possède également une composante radiale qu'il faut prendre en compte.

Enfin, une perspective pour cet outil serait de réussir à le lier à GMAT. Dans l'état actuel, bien qu'initialement prévu dans le cadre de ce projet, la liaison avec GMAT n'est pas réalisée. Un manque de temps couplé à des difficultés liées à l'utilisation et à la complexité de l'API Python de GMAT ont rendu impossible la réalisation de cet objectif.

Par ailleurs, le module Matériaux est largement améliorable. Pour le moment il ne prend en compte que quelques métaux fréquemment utilisés dans l'industrie aérospatiale. Dans le cadre de la recherche liée à ce projet, il pourrait être intéressant d'implémenter des matériaux composites et les fibres de carbone. Cependant, ces matériaux étant souvent créés pour un usage spécifique, il est difficile de trouver des composites qui pourraient satisfaire nos besoins dans la mesure où les besoins liés au câble n'ont pas encore été tous entièrement définis.

### **4.3 Impacts du projet :**

Bien que cet outil développé reste encore améliorable et optimisable, il reste tout de même utilisable et exploitable. Les fonctions et modules développés ici fonctionnent et donnent des résultats cohérents avec les attentes de la théorie.

Cet outil permet essentiellement de pouvoir calculer la force électromagnétique générée par un câble électrodynamique. En étant couplé à d'autres outils Python de modélisation de traînée atmosphérique et de calculs d'orbites, il sera donc possible de simuler l'impact d'un système de désorbitation électromagnétique sur un satellite en LEO. Il sera donc possible de tester et de valider ou d'infirmer numériquement des prototypes et autres résultats analytiques théoriques.

## CONCLUSION

Pour conclure, malgré les quelques points encore améliorables et ceux qui n'ont pas pu être traité par manque de temps, l'outil numérique développé durant ce projet est fonctionnel. Les objectifs énoncés au début ont en grande partie été réalisés. L'outil développé est bien capable de calculer des traînées électromagnétique développées par un câble électrodynamique en orbite autour de la Terre. Le champ magnétique terrestre est bien pris en compte et le câble peut avoir des formes diverses, allant d'un simple segment à une courbe 3D paramétrée. Les différents modules développés peuvent fonctionner de façon indépendante ou de façon combinée pour obtenir des résultats plus précis. Et les codes de tests montrent que la méthode suivie donne bien des résultats similaires à ce que pourrait donner un calcul analytique.

## BIBLIOGRAPHIE

- [1] Robert L. Forward et Robert P. Hoyt (2000), Terminator Tether™ : A Spacecraft Deorbit Device, *Journal of Spacecraft and Rockets*, Vol. 37 (No. 2), pages 187-197
- [2] Alexander M. Jablonski, (2008), Deorbiting of microsatellites in Low Earth Orbit (LEO) an introduction, *Defence R&D Canada – Ottawa*, Technical Memorandum
- [3] CNES et ESA, (2024), Les débris spatiaux, <https://cnes.fr/dossiers/debris-spatiaux>, consulté le 5 décembre 2024
- [4] G. Sánchez-Arriaga, S. Naghdi, K. Wätzig, J. Schilm, E.C. Lorenzini, M. Tajmar, E. Urgoiti, L. Tarabini Castellani, J.F. Plaza et A. Post, (2020), The E.T.PACK project : Towards a fully passive and consumable-less deorbit kit based on law-work-function tether technology, *Acta Astronomica*, Vol. 177, pages 821-827
- [5] G. Sánchez-Arriaga, L. Tarabini Castellani, E.C. Lorenzini, M. Tajmar, K. Wätzig et A. Post, ( ), Status of development of a deorbit device based on electrodynamic tether technologie in the E.T.PACK project
- [6] Karl M. Laundal, (2024), Pure Python International Geomagnetic Reference Field, <https://github.com/klaundal/ppigrf>, consulté le 5 décembre 2024
- [7] National Centers for Environmental Information, National Oceanic and Atmospheric Administration (NOAA), (2024), International Geomagnetic Reference Field (IGRF), <https://www.ncei.noaa.gov/products/international-geomagnetic-reference-field>, consulté le 5 décembre 2024
- [8] CGRSC, (2024), Système de référence terrestre (SRT), [https://cgrsc.ca/fr/ressources\\_fr/geodetic-reference-systems/terrestrial-reference-system-trs/#:~:text=Le%20Syst%C3%A8me%20de%20r%C3%A9f%C3%A9rence%20terrestre%20est%20un%20syst%C3%A8me,g%C3%A9om%C3%A9trique%20%C3%A0%20axes%20fixes%20ou%20ECF%20%28Earth-Centered%20Earth-Fixed%29](https://cgrsc.ca/fr/ressources_fr/geodetic-reference-systems/terrestrial-reference-system-trs/#:~:text=Le%20Syst%C3%A8me%20de%20r%C3%A9f%C3%A9rence%20terrestre%20est%20un%20syst%C3%A8me,g%C3%A9om%C3%A9trique%20%C3%A0%20axes%20fixes%20ou%20ECF%20%28Earth-Centered%20Earth-Fixed%29), consulté le 5 décembre 2024

[9] NASA Technologie Transfer Program, General Mission Analysis Tool (GMAT) Version R2018a, <https://software.nasa.gov/software/GSC-18094-1>, consulté le 5 décembre 2024

[10] Wikipedia, (2024), Earth-centered, Earth-fixed coordinate system, [https://en.wikipedia.org/wiki/Earth-centered,\\_Earth-fixed\\_coordinate\\_system](https://en.wikipedia.org/wiki/Earth-centered,_Earth-fixed_coordinate_system), consulté le 5 décembre 2024

[11] Wikipédia, (2023), Jour sidéral, [https://fr.wikipedia.org/wiki/Jour\\_sid%C3%A9ral](https://fr.wikipedia.org/wiki/Jour_sid%C3%A9ral), consulté le 5 décembre 2024

[12] Wikipédia, (2024), Equatorial coordinate system, [https://en.wikipedia.org/wiki/Equatorial\\_coordinate\\_system](https://en.wikipedia.org/wiki/Equatorial_coordinate_system), consulté le 5 décembre 2024

[13] Wikipédia, (2024), Point Vernal, [https://fr.wikipedia.org/wiki/Point\\_vernal](https://fr.wikipedia.org/wiki/Point_vernal), consulté le 5 décembre 2024

## ANNEXE I

### Module ForceElectro

```
import numpy as np
import Satellite
import ChampMagnetique as cm
```

```
'''
```

Ce code contient des fonctions qui permettent de calculer la force électrodynamique générée par un câble conducteur baigné dans un champ magnétique.

La fonction ForceElectro() permet de calculer cette force sur un segment rectiligne

La fonction Discret() permet de calculer cette force sur un segment que l'on va discrétiser

pour plus de précision. Cette fonction utilise les données du module ppigrf et réalise donc les calculs avec le champ magnétique terrestre

La fonction Parametre() fonctionne sur le même principe que la fonction Discret() mais cette fois-ci pour des courbes quelconques

```
'''
```

```
#-----
```

```
def ForceElectro(B, cable, I, V, R):
```

```
'''
```

Permet de calculer la force de Lorentz générée par un câble conducteur, de résistance électrique R, parcouru par un courant I, baigné dans un champ magnétique B et se déplaçant à la vitesse V.

- Le câble considéré doit être rectiligne

Attributs :

- B (array) (3,)	Champ magnétique	[T]
- cable (array) (2,3)	Coordonnées des extrémités de câble	[m]
- I (float or None)	Intensité du courant	[A]
- V (float)	Vitesse de déplacement du câble	[m/s]
- R (float)	Résistance électrique du câble	[Ohm]

Sortie :

- F_mag	Force de Lorentz résultante	[N]
---------	-----------------------------	-----

```
'''
```

```
V = np.array([V,0,0])
```

```
# Calcul du vecteur longueur du câble, orienté positivement en direction du satellite
```

```
L = -np.array([cable[1][0]-cable[0][0], cable[1][1]-cable[0][1], cable[1][2]-cable[0][2]])
```

```
l = np.linalg.norm(L) # Norme du vecteur longueur du câble = Longueur du câble
```

```
# Cas du courant induit
```

```
if I == None :
```

```
    E = np.cross(V,B) # Calcul du champ électrique induit
```

```
    U = np.dot(E,L) # Calcul de la tension générée dans le câble
```

```
    U = np.squeeze(U)
```

```
    I = (U/R)*L/l # Calcul du vecteur courant induit
```

```
    F_mag = l*np.cross(I,B) # Calcul de la force électrodynamique
```

```

# Cas du courant imposé
else:
    In = I*L/l # Calcul du vecteur courant
    F_mag = l*np.cross(In,B) # Calcul de la force électrodynamique

return F_mag

#-----

def Discret(r, theta, phi, date, INC, TA, cable, m, I, V, R):
    """
    Permet de calculer la force de Lorentz générée par un câble conducteur,
    de résistance électrique R, parcouru par un courant I, baigné dans un
    champ magnétique et se déplaçant à la vitesse V.

    - Le champ magnétique considéré ici est celui de la Terre

    - Le câble considéré doit être rectiligne et il sera discrétisé en m segments.
      Il doit être exprimé dans le repère du satellite

    - Le satellite, de coordonnées sphériques (r,θ,φ), ainsi que son anomalie vraie
      et l'inclinaison de son orbite doivent être exprimées dans le repère
      terrestre ECEF

    Attributs :
        - r (float)          Coordonnée selon r du satellite          [km]
        - theta (float)      Coordonnée selon θ du satellite          [deg]
        - phi (float)        Coordonnée selon φ du satellite          [deg]
        - date (datetime)    Date de calcul
        - INC (float)        Inclinaison de l'orbite du satellite      [deg]
        - TA (float)         Anomalie vraie du satellite              [deg]
        - cable (array) (2,3) Coordonnées des extrémités de câble      [m]
        - m (int)            Nombre de points de discrétisation        []
        - I (float or None)  Intensité du courant                    [A]
        - V (float)          Vitesse de déplacement du câble          [m/s]
        - R (float)          Résistance électrique du câble            [Ohm]

    Sortie :
        - F_mag              Force de Lorentz résultante              [N]
    """
    # Pas de discrétisation selon x, y et z, dans le repère du satellite
    px = (cable[1][0] - cable[0][0])/m
    py = (cable[1][1] - cable[0][1])/m
    pz = (cable[1][2] - cable[0][2])/m

    Li = [] # Liste des points de discrétisation
    Bi = [] # Liste des champs magnétique aux différents points de discrétisation
    V = np.array([V,0,0]) # Vecteur vitesse
    F_mag = 0 # Force Electrodynamique
    Sat = np.array([r, [theta], [phi]]) # Vecteur position du satellite dans ECEF

    for i in range(m):
        Li.append([cable[0][0] + i*px , cable[0][1] + i*py , cable[0][2] + i*pz])
        P = Satellite.Sat2Earth(Li[i][0], Li[i][1], Li[i][2], INC, TA, Sat)
        B0 = cm.ChampMagnetique(P[0], P[1], P[2], date)
        B0 = Satellite.Earth2Sat_cm(B0[0], B0[1], B0[2], INC, TA)
        Bi.append(B0)

    Li.append(cable[1])

```



```

L = -np.array([Li[1][0]-Li[0][0], Li[1][1]-Li[0][1], Li[1][2]-Li[0][2]])
l = np.linalg.norm(L)

# Cas du courant induit
if I == None :
    U = 0 # Tension dans le câble
    '''
    Cette première boucle a pour but de calculer toutes les tensions induites sur
les segments de discrétisation, dans le but d'en déduire le courant induit qui
parcourt tout le câble
    '''
    for i in range(m):
        E = np.cross(V,Bi[i]) # Calcul du champ électrique induit
        U2 = np.dot(E,L) # Calcul de la tension induite dans le segment i
        U2 = np.squeeze(U2)
        U = U + U2

    I2 = (U/R) # Calcul du courant induit dans le câble
    In = I2* L/l # Calcul du vecteur courant sur les segments
    'Calcul de la somme des forces Electrodynamiques sur chaque segments'
    for i in range(m):
        F_mag = F_mag + l*np.cross(In,Bi[i])

# Cas du courant imposé
else :
    In = I*L/l # Calcul du vecteur courant sur les segments
    for i in range(m):
        F_mag = F_mag + l*np.cross(In,Bi[i])

return F_mag

#-----

def Parametre(r, theta, phi, date, INC, TA, X_cable, Y_cable, Z_cable, I, V, R):
    '''
    Permet de calculer la force de Lorentz générée par un câble conducteur,
    de résistance électrique R, parcouru par un courant I, baigné dans un
    champ magnétique et se déplaçant à la vitesse V.

    - Le champ magnétique considéré ici est celui de la Terre

    - Le câble est ici défini par les trois liste X_cable, Y_cable et Z_cable,
    qui contiennent les coordonnées des points qui définissent le câble,
    exprimées dans le repère du satellite

    - Le calcul se fait ici de façon discrète entre chaque point du câble

    - Le satellite, de coordonnées sphériques (r,θ,φ), ainsi que son anomalie vraie
    et l'inclinaison de son orbite doivent être exprimées dans le repère
    terrestre ECEF

    Attributs :
    - r (float)          Coordonnée selon r du satellite      [km]
    - theta (float)      Coordonnée selon θ du satellite      [deg]
    - phi (float)        Coordonnée selon φ du satellite      [deg]
    - date (datetime)    Date de calcul

```

```

- INC (float)          Inclinaison de l'orbite du satellite      [deg]
- TA (float)           Anomalie vraie du satellite             [deg]
- X_cable (array) (h,1) Liste des abscisses des points du câble [m]
- Y_cable (array) (h,1) Liste des ordonnées des points du câble [m]
- Z_cable (array) (h,1) Liste des côtes des points du câble     [m]
                        h étant la longueur de ces listes
- I (float or None)    Intensité du courant                    [A]
- V (float)            Vitesse de déplacement du câble        [m/s]
- R (float)            Résistance électrique du câble          [Ohm]

Sortie :
... - F_mag           Force de Lorentz résultante              [N]
...
n = len(X_cable) # Nombre de points de discrétisation
Cable = []       # Liste des coordonnées des différents points de discrétisation
Bi = []          # Liste du champ magnétique aux différents points
V = np.array([V,0,0]) # Vecteur vitesse
F_mag = 0        # Force Electrodynamique
Sat = np.array([[r], [theta], [phi]]) # Vecteur position du satellite dans ECEF

for i in range(n):
    Cable.append(np.array([X_cable[i], Y_cable[i], Z_cable[i]]))
    P = Satellite.Sat2Earth(Cable[i][0], Cable[i][1], Cable[i][2], INC, TA, Sat)
    B0 = cm.ChampMagnetique(P[0], P[1], P[2], date)
    B0 = Satellite.Earth2Sat_cm(B0[0], B0[1], B0[2], INC, TA)
    Bi.append(B0)

L = [] # Liste des vecteurs longueurs du câble sur les segments de
discrétisation, orientés positivement en direction du satellite
l = [] # Liste des normes des segments de discrétisations

# Cas du courant induit
if I == None :
    U = 0 # Tension dans le câble
    ...
    Cette première boucle a pour but de calculer toutes les tensions induites sur
les
    segments de discrétisation, dans le but d'en déduire le courant induit qui
parcourt
    tout le câble
    ...
    for i in range(n-1):
        L.append(-np.array([Cable[i+1][0]-Cable[i][0], Cable[i+1][1]-Cable[i][1],
Cable[i+1][2]-Cable[i][2]]))
        l.append(np.linalg.norm(L[i]))

        E = np.cross(V,Bi[i]) # Calcul du champ électrique induit
        U2 = np.dot(E,L[i])   # Calcul de la tension induite dans le segment i
        U2 = np.squeeze(U2)
        U = U + U2

I2 = (U/R) # Calcul du courant induit dans le câble

'Calcul de la somme des forces Electrodynamiques sur chaque segments'
for i in range(n-1):
    In = I2* L[i]/l[i] # Calcul du vecteur courant sur le segment i
    F_mag = F_mag + l[i]*np.cross(In,Bi[i])

# Cas du courant imposé
else :
```

```

    for i in range(n-1):
        L.append(-np.array([Cable[i+1][0]-Cable[i][0], Cable[i+1][1]-Cable[i][1],
Cable[i+1][2]-Cable[i][2]]))
        l.append(np.linalg.norm(L[i]))

        In = I*L[i]/l[i] # Calcul du vecteur courant sur le segment i
        F_mag = F_mag + l[i]*np.cross(In,Bi[i])

return F_mag

```

## ANNEXE II

### Module ChampMagnétique

```
import ppigrf
import numpy as np

def ChampMagnetique(r, theta, phi, date):
    """
    Permet de connaitre le champ magnétique terrestre au point de coordonnées
    sphériques (r,θ,φ), exprimé dans le repère terrestre ECEF, à la date définie
    par le paramètre 'date'

    Attributs :
        - r (float)          Coordonnée selon r du point de calcul          [km]
        - theta (float)      Coordonnée selon θ du point de calcul          [deg]
        - phi (float)        Coordonnée selon φ du point de calcul          [deg]
        - date (datetime)    Date de calcul

    Sorties :
        - B_r (float)        Composante selon r du champ magnétique terrestre [T]
        - B_theta (float)    Composante selon θ du champ magnétique terrestre [T]
        - B_phi (float)      Composante selon φ du champ magnétique terrestre [T]
    """
    [B_r, B_theta, B_phi] = ppigrf.igrf_gc(r, theta, phi, date) # [nT]
    B_r = np.squeeze(B_r) * 10**-9 # [T]
    B_theta = np.squeeze(B_theta) * 10**-9 # [T]
    B_phi = np.squeeze(B_phi) * 10**-9 # [T]

    return B_r, B_theta, B_phi
```

## ANNEXE III

### Module Satellite

```

import numpy as np
from math import cos, sin, pi

def Sat2Earth(X, Y, Z, INC, TA, Sat):
    '''
    Permet d'effectuer un changement de coordonnées entre les coordonnées
    cartésiennes du repère du satellite et les coordonnées sphériques du repère
    ECEF.

    - Fonctionne pour des coordonnées de points

    - Cette fonction est la réciproque de Earth2Sat()

    Attributs :
        - X (float)          Abscisse du point dans le repère du satellite    [m]
        - Y (float)          Ordonnée du point dans le repère du satellite    [m]
        - Z (float)          Côte du point dans le repère du satellite        [m]
        - INC (float)        Inclinaison de l'orbite du satellite              [deg]
        - TA (float)         Anomalie vraie du satellite                     [deg]
        - Sat (array) (3,1) Vecteur position du satellite dans ECEF
                           (coordonnées sphérique) (r,θ,φ)                  [km],[deg],[deg]

    Sorties :
        - r (float)          Coordonnée selon r du point dans le repère ECEF  [km]
        - theta (float)      Coordonnée selon θ du point dans le repère ECEF  [deg]
        - phi (float)        Coordonnée selon φ du point dans le repère ECEF  [deg]
    '''
    INC = INC*pi/180 # [rad]
    TA = TA*pi/180  # [rad]

    P = np.array([[X*10**-3],[Y*10**-3],[Z*10**-3]]) # [km] Vecteur position du point
    dans le repère du satellite

    # Matrice de passage Sphérique vers Cartésien Sat
    R = np.array([[0, -sin(-INC*sin(TA)), cos(-INC*sin(TA))],
                  [0, -cos(-INC*sin(TA)), -sin(-INC*sin(TA))],
                  [1, 0, 0]])
    R2 = np.linalg.inv(R) # Matrice de passage Cartésien Sat vers Sphérique

    # Coordonnées du point P dans le repère Sphérique
    [[r],[theta],[phi]] = np.dot(R2,P) + Sat # [km] Vecteur position du point dans le
    repère terrestre fixe

    return r, theta, phi

#-----

def Earth2Sat(r, theta, phi, INC, TA, Sat):
    '''
    Permet d'effectuer un changement de coordonnées entre les coordonnées
    sphériques du repère ECEF et les coordonnées cartésiennes du repère du satellite.

    - Fonctionne pour des coordonnées de points

```

```

- Cette fonction est la réciproque de Sat2Earth()

Attributs :
- r (float)          Coordonnée selon r du point dans le repère ECEF [km]
- theta (float)      Coordonnée selon  $\theta$  du point dans le repère ECEF [deg]
- phi (float)        Coordonnée selon  $\phi$  du point dans le repère ECEF [deg]
- INC (float)        Inclinaison de l'orbite du satellite [deg]
- TA (float)         Anomalie vraie du satellite [deg]
- Sat (array) (3,1)  Vecteur position du satellite dans ECEF
                    (coordonnées sphérique) (r, $\theta$ , $\phi$ ) [km],[deg],[deg]

Sorties :
- X (float)          Abscisse du point dans le repère du satellite [m]
- Y (float)          Ordonnée du point dans le repère du satellite [m]
- Z (float)          Côte du point dans le repère du satellite [m]
...
INC = INC*pi/180 # [rad]
TA = TA*pi/180  # [rad]

P = np.array([[r],[theta],[phi]]) # Vecteur position du point dans le repère
terrestre fixe

# Matrice de passage Sphérique vers Cartésien Sat
R = np.array([[0, -sin(-INC*sin(TA)), cos(-INC*sin(TA))],
              [0, -cos(-INC*sin(TA)), -sin(-INC*sin(TA))],
              [1, 0, 0]])

# Coordonnées du point P dans le repère du satellite
[[X],[Y],[Z]] = np.dot(R,P-Sat)

return X*10**-3, Y*10**-3, Z*10**-3 # [m]

#-----

def Earth2Sat_cm(r, theta, phi, INC, TA):
    '''
    Permet d'effectuer un changement de coordonnées entre les coordonnées
    sphériques du repère ECEF et les coordonnées cartésiennes du repère du satellite.

    - Fonctionne pour des coordonnées de vecteur

    - Cette fonction est la réciproque de Sat2Earth_cm()

    Attributs :
    - r (float)          Coordonnée selon r du point dans le repère ECEF [km]
    - theta (float)      Coordonnée selon  $\theta$  du point dans le repère ECEF [deg]
    - phi (float)        Coordonnée selon  $\phi$  du point dans le repère ECEF [deg]
    - INC (float)        Inclinaison de l'orbite du satellite [deg]
    - TA (float)         Anomalie vraie du satellite [deg]

    Sorties :
    - X (float)          Abscisse du point dans le repère du satellite [m]
    - Y (float)          Ordonnée du point dans le repère du satellite [m]
    - Z (float)          Côte du point dans le repère du satellite [m]
    ...
    INC = INC*pi/180 # [rad]
    TA = TA*pi/180  # [rad]

    P = np.array([[r],[theta],[phi]]) # Vecteur position du point dans le repère
    terrestre fixe

```

```

# Matrice de passage Sphérique vers Cartésien Sat
R = np.array([[0, -sin(-INC*sin(TA)), cos(-INC*sin(TA))],
              [0, -cos(-INC*sin(TA)), -sin(-INC*sin(TA))],
              [1, 0, 0]])

# Coordonnées du point P dans le repère du satellite
[[X],[Y],[Z]] = np.dot(R,P)

return X, Y, Z # [m]

#-----
def Sat2Earth_cm(X, Y, Z, INC, TA):
    """
    Permet d'effectuer un changement de coordonnées entre les coordonnées
    cartésiennes du repère du satellite et les coordonnées sphériques du repère
    ECEF.

    - Fonctionne pour des coordonnées de points

    - Cette fonction est la réciproque de Earth2Sat()

    Attributs :
        - X (float)          Abscisse du point dans le repère du satellite      [m]
        - Y (float)          Ordonnée du point dans le repère du satellite      [m]
        - Z (float)          Côte du point dans le repère du satellite          [m]
        - INC (float)        Inclinaison de l'orbite du satellite               [deg]
        - TA (float)         Anomalie vraie du satellite                       [deg]

    Sorties :
        - r (float)          Coordonnée selon r du point dans le repère ECEF    [km]
        - theta (float)      Coordonnée selon  $\theta$  du point dans le repère ECEF [deg]
        - phi (float)        Coordonnée selon  $\phi$  du point dans le repère ECEF    [deg]
    """
    INC = INC*pi/180 # [rad]
    TA = TA*pi/180  # [rad]

    P = np.array([[X*10**-3],[Y*10**-3],[Z*10**-3]]) # [km] Vecteur position du point
    dans le repère du satellite

    # Matrice de passage Sphérique vers Cartésien Sat
    R = np.array([[0, -sin(-INC*sin(TA)), cos(-INC*sin(TA))],
                  [0, -cos(-INC*sin(TA)), -sin(-INC*sin(TA))],
                  [1, 0, 0]])
    R2 = np.linalg.inv(R) # Matrice de passage Cartésien Sat vers Sphérique

    # Coordonnées du point P dans le repère Sphérique
    [[r],[theta],[phi]] = np.dot(R2,P) # [km] Vecteur position du point dans le
    repère terrestre fixe

    return r, theta, phi

```

## ANNEXE IV

### Module Câble

```

import matplotlib.pyplot as plt
import matplotlib
matplotlib.use('Qt5Agg')
import numpy as np

def Resistance(L,S,res):
    """
    Permet de calculer la résistance électrique d'un câble dont on connaît les
    dimensions et le matériau qui le compose

    Attributs :
        - L (float)      Longueur du câble                [m]
        - S (float)      Surface de la section du câble    [m²]
        - res (float)    Résistivité électrique du matériau du câble [Ohm.m]

    Sortie :
        - R (float)      Résistance électrique du câble    [Ohm]
    """
    R = res*L/S
    return R
#-----

def Volume(L,S):
    """
    Permet de calculer le volume du câble dont on connaît les dimensions

    Attributs :
        - L (float)      Longueur du câble                [m]
        - S (float)      Surface de la section du câble    [m²]

    Sortie :
        - Vol (float)    Volume du câble                    [m³]
    """
    Vol = L*S
    return Vol
#-----

def Masse(Vol,rho):
    """
    Permet de calculer la masse du câble dont on connaît les caractéristiques

    Attributs :
        - Vol (float)    Volume du câble                    [m³]
        - rho (float)    Masse volumique du matériau        [kg/m³]

    Sortie :
        - m (float)      Masse du câble                     [kg]
    """
    m = Vol*rho
    return m
#-----

def Mat2Liste(cable,m):
    """

```



Permet de discrétiser un câble rectiligne défini par une matrice contenant les points au extrémité du câble.

Cette fonction est la réciproque de Liste2Mat()

Attributs :

- cable (array) (2,3)      Coordonnées des extrémités de câble      [m]
- m (int)                      Nombre de points de discrétisation      []

Sorties :

- x (array) (m,)              Liste des abscisses des points du câble [m]
- y (array) (m,)              Liste des ordonnées des points du câble [m]
- z (array) (m,)              Liste des côtes des points du câble      [m]

```
...
x = np.linspace(cable[0][0],cable[1][0],m)
y = np.linspace(cable[0][1],cable[1][1],m)
z = np.linspace(cable[0][2],cable[1][2],m)
return x,y,z
```

#-----

```
def Liste2Mat(x,y,z):
    '''
```

Permet de transformer un câble rectiligne défini par des listes de coordonnées en une matrice qui contient les coordonnées des extrémités du câble

Cette fonction est la réciproque de Mat2Liste()

Attributs :

- x (array) (m,)              Liste des abscisses des points du câble [m]
- y (array) (m,)              Liste des ordonnées des points du câble [m]
- z (array) (m,)              Liste des côtes des points du câble      [m]

Sortie :

- cable (array) (2,3)      Coordonnées des extrémités du câble      [m]

```
...
cable = np.array([[x[0],y[0],z[0]],[x[-1],y[-1],z[-1]]])
return cable
```

#-----

```
def Graph(x, y, z, titre=None):
    '''
```

Permet de tracer dans un repère 3D un câble défini par des listes de coordonnées

Attributs :

- x (array) (m,)              Liste des abscisses des points du câble [m]
- y (array) (m,)              Liste des ordonnées des points du câble [m]
- z (array) (m,)              Liste des côtes des points du câble      [m]
- titre (str)                      Titre du graphique

Sortie :

- Graph du câble dans le repère du satellite

```
...
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot(x, y, z, label='Câble')
ax.scatter(0, 0, 0, color='red',label='Satellite' ,s=100)
ax.set_xlabel('X [m]')
ax.set_ylabel('Y [m]')
ax.set_zlabel('Z [m]')
```

```
if titre == None :  
    ax.set_title('Câble Paramétré en 3D')  
else :  
    ax.set_title(titre)  
ax.set_box_aspect([1, 1, 1])  
plt.legend()  
plt.show()
```

## ANNEXE V

### Module Matériaux

```
'''
Liste des matériaux disponible :      Class associée
- Al 2024 T3                          Al_2024
- Al 6061 T6                          Al_6061
- Al 7075 T6                          Al_7075
- Al (pure)                          Al
- Cu (pure)                          Cu
- Cu Cold Drawn (pure)               Cu_CD
'''
```

Pour chacune de ces class, la fonction Info() donne toutes les informations concernant les propriétés physiques disponibles, leur valeur, leur unité ainsi que la fonction associée

```
'''
class Al_2024:
    def __init__(self):
        pass

    def MasseVolumique(self):
        return 2780 # [kg/m^3]

    # Propriétés Mécaniques
    def ModuleYoung(self):
        return 73.1 # [GPa]

    def CoeffPoison(self):
        return 0.33

    def ModuleCisaillement(self):
        return 28 # [GPa]

    def LimiteCisaillement(self):
        return 283 # [MPa]

    def LimiteFatigue(self):
        return 138 # [MPa]

    def ResistanceTraction(self):
        return 483 # [MPa]

    def LimiteElastique(self):
        return 345 # [MPa]

    # Propriétés électriques
    def Resistivite(self):
        return 5.82*10**-8 # [Ohm.m]

    def Conductance(self):
        return 1/(5.82*10**-8) # [S/m]

    # Propriétés Thermiques
    def ThConductivite(self):
        return 121 # [W/m.K]
'''
```

```

def ChaleurSpe(self):
    return 875 # [J/kg/K]

def Info(self):
    print('Propriétés Aluminium 2024-T3 :          unité    fonction')

print('-----')
print('Masse Volumique      ρ = 2780      kg/m^3    MasseVolumique()')
print('Module de Young      E = 73.1      GPa      ModuleYoung()')
print('Coefficient de Poisson ν = 0.33      CoeffPoison()')
print('Module de Cisaillement G = 28      GPa')
ModuleCisaillement()
print('Limite de Cisaillement      = 283      MPa')
LimiteCisaillement()
print('Limite en Fatigue      = 138      MPa      LimiteFatigue()')
print('Limite de Résistance σ_u = 483      MPa')
ResistanceTraction()
print('      en Traction')
print('Limite Elastique      σ_Y = 345      MPa      LimiteElastique()')
print('Résistivité électrique      = 5.82e-8      Ω.m      Resistivite()')
print('Conductance électrique      = 17182130.58      S/m      Conductance()')
print('Conductivité Thermique λ = 121      W/m/K      ThConductivite()')
print('Chaleur Spécifique      c = 875      J/kg/K      ChaleurSpe()')
#-----

class Al_6061:
    def __init__(self):
        pass

    def MasseVolumique(self):
        return 2700 # [kg/m^3]

    # Propriétés Mécaniques
    def ModuleYoung(self):
        return 68.9 # [GPa]

    def CoeffPoison(self):
        return 0.33

    def ModuleCisaillement(self):
        return 26 # [GPa]

    def LimiteCisaillement(self):
        return 207 # [MPa]

    def LimiteFatigue(self):
        return 96.5 # [MPa]

    def ResistanceTraction(self):
        return 310 # [MPa]

    def LimiteElastique(self):
        return 276 # [MPa]

    # Propriétés électriques
    def Resistivite(self):
        return 3.99*10**-8 # [Ohm.m]

    def Conductance(self):
        return 1/(3.99*10**-8) # [S/m]

```

```

# Propriétés Thermiques
def ThConductivite(self):
    return 167 # [W/m.K]

def ChaleurSpe(self):
    return 896 # [J/kg/K]

def Info(self):
    print('Propriétés Aluminium 6061-T6,6061-T651 : unité    fonction')

print('-----')
print('Masse Volumique      ρ = 2700      kg/m^3    MasseVolumique()')
print('Module de Young      E = 68.9      GPa      ModuleYoung()')
print('Coefficient de Poison ν = 0.33      CoeffPoison()')
print('Module de Cisaillement G = 26      GPa      ModuleCisaillement()')
print('Limite de Cisaillement = 207      MPa      LimiteCisaillement()')
print('Limite en Fatigue      = 96.5      MPa      LimiteFatigue()')
print('Limite de Résistance σ_u = 310      MPa      ResistanceTraction()')
print('      en Traction')
print('Limite Elastique      σ_Y = 276      MPa      LimiteElastique()')
print('Résistivité électrique = 3.99e-8      Ω.m      Resistivite()')
print('Conductance électrique = 25062656.64      S/m      Conductance()')
print('Conductivité Thermique λ = 167      W/m/K      ThConductivite()')
print('Chaleur Spécifique    c = 896      J/kg/K      ChaleurSpe()')
#-----

class AL_7075:
    def __init__(self):
        pass

    def MasseVolumique(self):
        return 2810 # [kg/m^3]

    # Propriétés Mécaniques
    def ModuleYoung(self):
        return 71.7 # [GPa]

    def CoeffPoison(self):
        return 0.33

    def ModuleCisaillement(self):
        return 26.9 # [GPa]

    def LimiteCisaillement(self):
        return 331 # [MPa]

    def LimiteFatigue(self):
        return 159 # [MPa]

    def ResistanceTraction(self):
        return 572 # [MPa]

    def LimiteElastique(self):
        return 503 # [MPa]

    # Propriétés électriques

```

```

def Resistivite(self):
    return 5.15*10**-8 # [Ohm.m]

def Conductance(self):
    return 1/(5.15*10**-8) # [S/m]

# Propriétés Thermiques
def ThConductivite(self):
    return 130 # [W/m.K]

def ChaleurSpe(self):
    return 960 # [J/kg/K]

def Info(self):
    print('Propriétés Aluminium 7075-T6;7075-T651 : unité    fonction')

print('-----')
print('Masse Volumique      ρ = 2810      kg/m^3    MasseVolumique()')
print('Module de Young      E = 71.7      GPa      ModuleYoung()')
print('Coefficient de Poisson ν = 0.33      CoeffPoison()')
print('Module de Cisaillement G = 26.9      GPa')
ModuleCisaillement()
print('Limite de Cisaillement = 331      MPa')
LimiteCisaillement()
print('Limite en Fatigue      = 159      MPa      LimiteFatigue()')
print('Limite de Résistance σ_u = 572      MPa')
ResistanceTraction()
print('      en Traction')
print('Limite Elastique      σ_Y = 503      MPa      LimiteElastique()')
print('Résistivité électrique = 5.15e-8      Ω.m      Resistivite()')
print('Conductance électrique = 19417475.73      S/m      Conductance()')
print('Conductivité Thermique λ = 130      W/m/K      ThConductivite()')
print('Chaleur Spécifique      c = 960      J/kg/K      ChaleurSpe()')
#-----

class Al:
    def __init__(self):
        pass

    def MasseVolumique(self):
        return 2698.9 # [kg/m^3]

    # Propriétés Mécaniques
    def ModuleYoung(self):
        return 68 # [GPa]

    def CoeffPoison(self):
        return 0.36

    def ModuleCisaillement(self):
        return 25 # [GPa]

    # Propriétés électriques
    def Resistivite(self):
        return 2.7*10**-8 # [Ohm.m]

    def Conductance(self):
        return 1/(2.7*10**-8) # [S/m]

    # Propriétés Thermiques

```

```

def ThConductivite(self):
    return 210 # [W/m.K]

def ChaleurSpe(self):
    return 900 # [J/kg/K]

def Info(self):
    print('Propriétés Aluminium (pure) :          unité    fonction')

print('-----')
print('Masse Volumique      ρ = 2698.9      kg/m^3    MasseVolumique()')
print('Module de Young      E = 68          GPa      ModuleYoung()')
print('Coefficient de Poison  ν = 0.36          CoeffPoison()')
print('Module de Cisaillement G = 25          GPa      ModuleCisaillement()')
print('Résistivité électrique  = 2.7e-8      Ω.m      Resistivite()')
print('Conductance électrique  = 37037037.04 S/m      Conductance()')
print('Conductivité Thermique λ = 210        W/m/K     ThConductivite()')
print('Chaleur Spécifique      c = 900        J/kg/K     ChaleurSpe()')
#-----

class Cu:
    def __init__(self):
        pass

    def MasseVolumique(self):
        return 8930 # [kg/m^3]

    # Propriétés Mécaniques
    def ModuleYoung(self):
        return 110 # [GPa]

    def CoeffPoison(self):
        return 0.343

    def ModuleCisaillement(self):
        return 46 # [GPa]

    def ResistanceTraction(self):
        return 210 # [MPa]

    def LimiteElastique(self):
        return 33.3 # [MPa]

    # Propriétés électriques
    def Resistivite(self):
        return 1.7*10**-8 # [Ohm.m]

    def Conductance(self):
        return 1/(1.7*10**-8) # [S/m]

    # Propriétés Thermiques
    def ThConductivite(self):
        return 385 # [W/m.K]

    def ChaleurSpe(self):
        return 385 # [J/kg/K]

    def Info(self):
        print('Propriétés Cuivre (pure) :          unité    fonction')

```

```

print('-----')
print('Masse Volumique       $\rho$  = 8930      kg/m^3      MasseVolumique()')
print('Module de Young      E = 110          GPa          ModuleYoung()')
print('Coefficient de Poison   $\nu$  = 0.343      CoeffPoison()')
print('Module de Cisaillement G = 46          GPa')
ModuleCisaillement()
print('Limite de Résistance   $\sigma_u$  = 210      MPa')
ResistanceTraction()
print('          en Traction')
print('Limite Elastique       $\sigma_Y$  = 33.3      MPa          LimiteElastique()')
print('Résistivité électrique  = 1.7e-8           $\Omega.m$           Resistivite()')
print('Conductance électrique  = 58823529.41      S/m          Conductance()')
print('Conductivité Thermique   $\lambda$  = 385      W/m/K          ThConductivite()')
print('Chaleur Spécifique     c = 385          J/kg/K          ChaleurSpe()')
#-----

class Cu_CD:
    def __init__(self):
        pass

    def MasseVolumique(self):
        return 8930 # [kg/m^3]

    # Propriétés Mécaniques
    def ModuleYoung(self):
        return 110 # [GPa]

    def CoeffPoison(self):
        return 0.364

    def ModuleCisaillement(self):
        return 46 # [GPa]

    def ResistanceTraction(self):
        return 344 # [MPa]

    def LimiteElastique(self):
        return 333.4 # [MPa]

    # Propriétés électriques
    def Resistivite(self):
        return 1.7*10**-8 # [Ohm.m]

    def Conductance(self):
        return 1/(1.7*10**-8) # [S/m]

    # Propriétés Thermiques
    def ThConductivite(self):
        return 385 # [W/m.K]

    def ChaleurSpe(self):
        return 385 # [J/kg/K]

    def Info(self):
        print('Propriétés Cuivre Cold Drawn (pure) :      unité      fonction')

print('-----')
print('Masse Volumique       $\rho$  = 8930      kg/m^3      MasseVolumique()')
print('Module de Young      E = 110          GPa          ModuleYoung()')

```



```

    print('Coefficient de Poison    $\nu$  = 0.364           CoeffPoison()')
    print('Module de Cisaillement  G = 46               GPa
ModuleCisaillement()')
    print('Limite de Résistance    $\sigma_u$  = 344         MPa
ResistanceTraction()')
    print('          en Traction')
    print('Limite Elastique        $\sigma_Y$  = 333.4       MPa   LimiteElastique()')
    print('Résistivité électrique   = 1.7e-8             $\Omega \cdot m$  Resistivite()')
    print('Conductance électrique     = 58823529.41       S/m    Conductance()')
    print('Conductivité Thermique     $\lambda$  = 385          W/m/K   ThConductivite()')
    print('Chaleur Spécifique          c = 385             J/kg/K   ChaleurSpe()')

```

## ANNEXE VI

### Code Test\_Force\_Electro

```

import numpy as np
from math import pi
import Force_Electro as fe
import Cable as ca
from Matériaux import AL_7075

# Caractéristiques du matériau
Al = AL_7075()
rho = Al.MasseVolumique() # [kg/m^3] Masse volumique
res = Al.Resistivite() # [Ohm.m] Résistivité électrique du matériaux

# Caractéristiques du câble
cable = np.array([[0,0,0],[-20*np.sqrt(3), 0, -20]]) # Pour avoir L = 40
L = np.linalg.norm(cable) # [m] Longueur du câble

d = 5 # [mm] Diamètre du câble
S = pi*((d*10**-3)/2)**2 # [m^2] Surface de la section du câble

R = ca.Resistance(L, S, res) # [Ohm] Résistance électrique du câble

Vol = ca.Volume(L,S) # [m^3] Volume totale du câble
m = ca.Masse(Vol,rho) # [kg] Masse du câble

V = 7.5 # [m/s] Vitesse de déplacement

B = np.array([0,1,0])*10**-3 # [T] Champ magnétique

I = 1.5 # [A] Intensité du courant

# Calcul de la force électrodynamique dans le cas d'un courant imposé
F1 = fe.ForceElectro(B, cable, I, V, R)
# Calcul de la force électro dynamique dans le cas d'un courant induit
F2 = fe.ForceElectro(B, cable, None, V, R)

print(' -----')
print(' Cas induit : F =')
print('')
print(' ',F2, ' N')
print(' Norme = ',np.linalg.norm(F2))
print(' -----')
print(' Cas imposé : F =')
print('')
print(' ',F1, ' N')
print(' Norme = ',np.linalg.norm(F1))
print(' -----')

```

## ANNEXE VII

### Code Test\_Cable\_rectiligne

```

import numpy as np
from datetime import datetime
from astropy import constants
from math import pi

import Force_Electro as fe
import ChampMagnetique as cm
import Satellite
import Cable as ca
from Materiaux import Al_6061

# Caractéristiques du matériau
Al = Al_6061()
rho = Al.MasseVolumique() # [kg/m^3] Masse volumique
res = Al.Resistivite() # [Ohm.m] Résistivité électrique du matériaux

# Constantes
R_T = constants.R_earth.value # [m] Rayon de la Terre
M_T = constants.M_earth.value # [kg] Masse de la Terre
G = constants.G.value # [m^3/kg/s^2] Constante Universelle de Gravitation
mu = G * M_T # [m^3/s^2] Paramètre Gravitationnel Standard de la Terre

# Caractéristiques du câble
cable = np.array([[0,0,0],[-2650, 0, -3000]])
L = np.linalg.norm(cable) # [m] Longueur du câble
d = 5 # [mm] Diamètre du câble
S = pi*((d*10**-3)/2)**2 # [m^2] Surface de la section du câble
R = ca.Resistance(L, S, res) # [Ohm] Résistance électrique du câble
Vol = ca.Volume(L,S) # [m^3] Volume totale du câble
m = ca.Masse(Vol,rho) # [kg] Masse du câble

np.set_printoptions(precision=9)

# 1er Cas : Avec la fonction ForceElectro() -----

# Position du satellite
r = R_T*10**-3 + 800 # [km]
theta = 114 # [deg]
phi = 168 # [deg]
INC = 25 # [deg]
TA = 180 # [deg]
date = datetime(2020,7,15,15,20,0)

# Vitesse du satellite sur son orbite
V = np.sqrt(mu/(r*10**3)) # [m/s] Calcul de la vitesse du satellite sur son orbite

# Champ Magnétique terrestre au niveau du satellite
B = cm.ChampMagnetique(r, theta, phi, date)
B = Satellite.Earth2Sat_cm(B[0],B[1],B[2],INC,TA) # [T]

# Calcul de la force de Lorentz
F1a = fe.ForceElectro(B,cable, None, V, R) # Cas induit
F1b = fe.ForceElectro(B,cable,1.5,V,R) # Cas imposé

```

```

# 2e Cas : Avec la fonction Discret() -----

# Calcul de la force de Lorentz
F2a = fe.Discret(r, theta, phi, date, INC, TA, cable, 100, None, V, R) # Cas imposé
F2b = fe.Discret(r, theta, phi, date, INC, TA, cable, 100, 1.5, V, R) # Cas imposé

# 3e Cas : Avec la fonction Parametre() -----

# Définition de la courbe paramétrée
[x,y,z] = ca.Mat2Liste(cable, 1000)

# Calcul de la force de Lorentz
F3a = fe.Parametre(r, theta, phi, date, INC, TA, x, y, z, None, V, R) # Cas induit
F3b = fe.Parametre(r, theta, phi, date, INC, TA, x, y, z, 1.5, V, R) # Cas imposé

# Comparaison : -----

print(' -----')
print(' Comparaisons des fonctions :')
print(' -----')
print(' Cas induit :')
print('           F_x           F_y           F_z           ||F||')
print(' ForceElectro() : ',F1a,' ',np.round(np.linalg.norm(F1a),9),' N')
print(' Discret()      : ',F2a,' ',np.round(np.linalg.norm(F2a),9),' N')
print(' Parametre()    : ',F3a,' ',np.round(np.linalg.norm(F3a),9),' N')
print(' -----')
print(' Cas imposé :')
print('           F_x           F_y           F_z           ||F||')
print(' ForceElectro() : ',F1b,' ',np.round(np.linalg.norm(F1b),9),' N')
print(' Discret()      : ',F2b,' ',np.round(np.linalg.norm(F2b),9),' N')
print(' Parametre()    : ',F3b,' ',np.round(np.linalg.norm(F3b),9),' N')
print(' -----')

```

## ANNEXE VIII

### Code Test\_Courbe\_Parametree

```
import numpy as np
from datetime import datetime
from math import pi
from astropy import constants

import Force_Electro as fe
import ChampMagnetique as cm
import Cable as ca
import Satellite
from Materiaux import Al_2024

'/!\ le code peut mettre plusieurs minutes pour tourner dans son intégralité'

# Données_-----

# Constantes
R_T = constants.R_earth.value # [m] Rayon de la Terre
M_T = constants.M_earth.value # [kg] Masse de la Terre
G = constants.G.value # [m^3/kg/s^2] Constante Universelle de Gravitation
mu = G * M_T # [m^3/s^2] Paramètre Gravitationnel Standard de la Terre

# Caractéristiques du matériau
Al = Al_2024()
rho = Al.MasseVolumique()
res = Al.Resistivite()

# Caractéristiques du câble
d = 5 # [mm] Diamètre du câble
S = pi*((d*10**-3)/2)**2 # [m^2] Surface de la section du câble

D = 2000 # [m] Diamètre du cercle formé par le câble
L = pi*D # [m] Longueur du câble

R = ca.Resistance(L, S, res) # [Ohm] Résistance électrique du câble
Vol = ca.Volume(L,S) # [m^3] Volume totale du câble
m = ca.Masse(Vol,rho) # [kg] Masse du câble

# 1er Cas : Câble Paramétré_-----

# Définition de la courbe paramétrée
t = np.linspace(-pi,pi,1000)
x = 500*(np.sin(t)-1)
y = 1000*np.cos(t)
z = 500*np.sqrt(3)*(np.sin(t)-1)
ca.Graph(x,y,z, 'Cas 1 : Câble paramétré') # Représentation 3D de la courbe

# Position du satellite
r = R_T*10**-3 + 800 # [km]
theta = 114 # [deg]
phi = 168 # [deg]
INC = 25 # [deg]
TA = 180 # [deg]
date = datetime(2020,7,15,15,20,0)
```

```

V = np.sqrt(mu/(r*10**3)) # [m/s] Vitesse du satellite sur son orbite

# Calcul de la force de Lorentz générée
F1a = fe.Parametre(r, theta, phi, date, INC, TA, x, y, z, None, V, R) # Cas induit
F1b = fe.Parametre(r, theta, phi, date, INC, TA, x, y, z, 1.5, V, R) # Cas imposé

print(' -----')
print(' Cas 1 : Câble paramétré ')
print('')
print(' Cas induit :')
print('   F   = ',F1a,' N')
print(' ||F|| = ',np.linalg.norm(F1a),' N')
print('')
print(' Cas imposé :')
print('   F   = ',F1b,' N')
print(' ||F|| = ',np.linalg.norm(F1b),' N')
print(' -----')
print('')

# 2e Cas : Câble défini par morceaux-----

# Définition du câble
x1 = np.linspace(0,-1000,501)
x1 = np.delete(x1,-1)
x2 = np.linspace(-1000,0,500)
xx = np.concatenate((x1,x2))

y1 = np.linspace(0,1000,251)
y1 = np.delete(y1,-1)
y2 = np.linspace(1000,0,251)
y2 = np.delete(y2,-1)
y3 = np.linspace(0,-1000,251)
y3 = np.delete(y3,-1)
y4 = np.linspace(-1000,0,250)
yy = np.concatenate((y1,y2,y3,y4))

zz = np.sqrt(3)*xx
ca.Graph(xx,yy,zz, ' Cas 2 : Câble défini par morceaux') # Représentation 3D de la
courbe

# Calcul de la force de Lorentz générée
F2a = fe.Parametre(r, theta, phi, date, INC, TA, xx, yy, zz, None, V, R) # Cas induit
F2b = fe.Parametre(r, theta, phi, date, INC, TA, xx, yy, zz, 1.5, V, R) # Cas imposé

print(' -----')
print(' Cas 2 : Câble définit par morceaux')
print('')
print(' Cas induit :')
print('   F   = ',F2a,' N')
print(' ||F|| = ',np.linalg.norm(F2a),' N')
print('')
print(' Cas imposé :')
print('   F   = ',F2b,' N')
print(' ||F|| = ',np.linalg.norm(F2b),' N')
print(' -----')
print('')

# 3e Cas : Câble soumis à un champ magnétique uniforme-----

# Définition du champ magnétique

```

```

B = cm.ChampMagnetique(r, theta, phi, date)
B = Satellite.Earth2Sat_cm(B[0], B[1], B[2], INC, TA) # [T]

# Définition du câble (le même que le cas 2)
cable = np.array([xx,yy,zz])

F3 = 0
for i in range(len(xx)-1):
    cable_i =
np.array([[cable[0][i],cable[1][i],cable[2][i]], [cable[0][i+1],cable[1][i+1],cable[2]
[i+1]]])
    F3 = F3 + fe.ForceElectro(B, cable_i, 1.5, V, R)

print('-----')
print(' Cas 3 : Câble soumis à un champ magnétique uniforme')
print('')
print(' Cas imposé :')
print('   F   = ',F3,' N')
print(' ||F|| = ',np.linalg.norm(F3),' N')
print('-----')
print('')

```

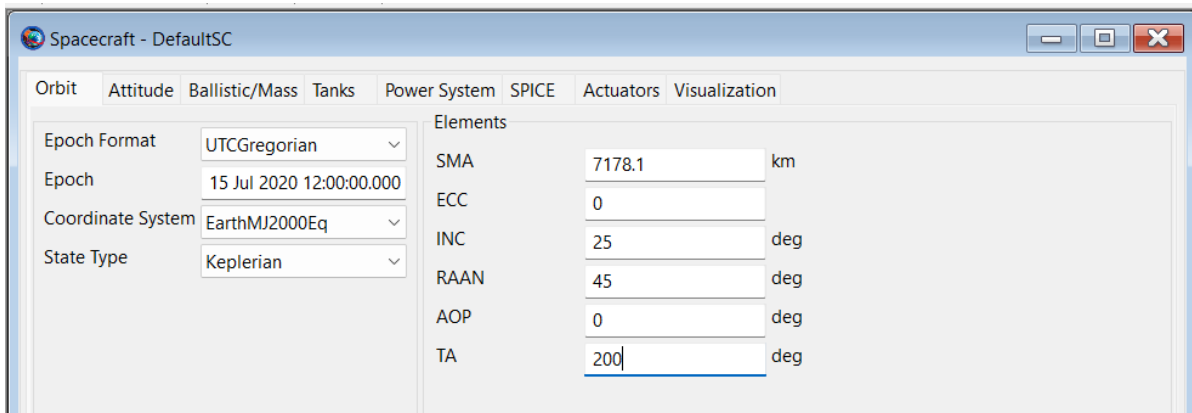
## ANNEXE IX

### Simulation GMAT

Pour les fonctions de test, et afin d'avoir des données réalistes, on décrit ici la simulation réalisée sur GMAT qui a servi à obtenir les données.

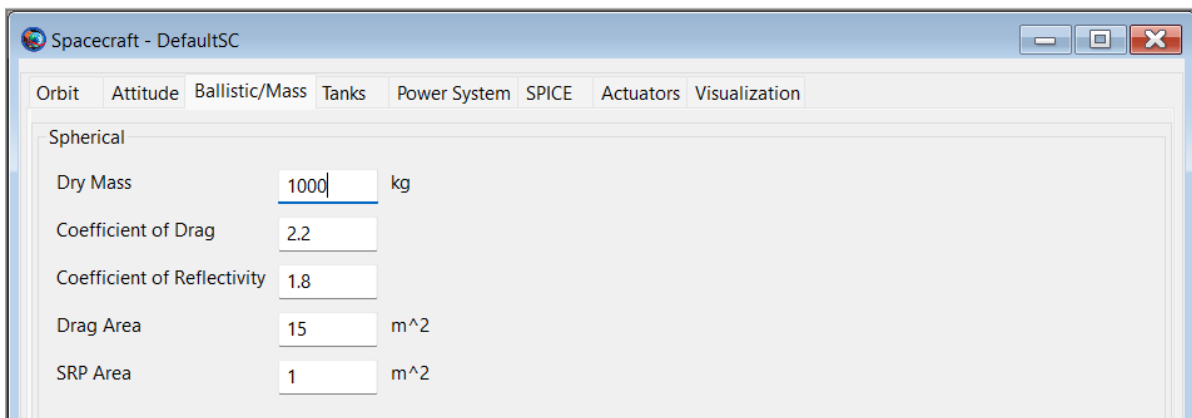
La simulation utilisée ici est très simple. Elle a pour point de départ la mission par défaut dans GMAT. Si un autre fichier est déjà ouvert, il suffit d'appuyer sur l'icône "*new mission*", ce qui ouvrira la mission par défaut de GMAT.

Dans ce fichier, on réalise peu de modifications. On ne change que des paramètres dans la définition du satellite, comme indiqué dans les figures ci-dessous :



The screenshot shows the 'Spacecraft - DefaultSC' window with the 'Elements' tab selected. The 'Orbit' tab is also visible. The 'Epoch Format' is set to 'UTCGregorian', 'Epoch' is '15 Jul 2020 12:00:00.000', 'Coordinate System' is 'EarthMJ2000Eq', and 'State Type' is 'Keplerian'. The 'Elements' section contains the following parameters:

Parameter	Value	Unit
SMA	7178.1	km
ECC	0	
INC	25	deg
RAAN	45	deg
AOP	0	deg
TA	200	deg



The screenshot shows the 'Spacecraft - DefaultSC' window with the 'Spherical' tab selected. The 'Orbit' tab is also visible. The 'Spherical' section contains the following parameters:

Parameter	Value	Unit
Dry Mass	1000	kg
Coefficient of Drag	2.2	
Coefficient of Reflectivity	1.8	
Drag Area	15	m <sup>2</sup>
SRP Area	1	m <sup>2</sup>



On fait ensuite tourner le programme et on note les résultats importants dans le tableau ci-dessous :

	EarthMJ2000Eq	EarthFixed	
Cartesian State			
X	5756,529637	-6412,623899	km
Y	-3140,209086	1393,863259	km
Z	-2920,212651	-2908,880961	km
VX	3,873707472	-1,840939505	km/sec
VY	6,308594659	-6,675032022	km/sec
VZ	0,851206958	0,858773713	km/sec
Keplerian State			
SMA	7178,048936	6390,244633	km
ECC	6,05403E-05	0,123301996	
INC	24,99615681	25,05740514	deg
RAAN	44,17081877	239,1980277	deg
AOP	178,8953564	106,9232886	deg
TA	106,79176	179,9747825	deg
MA	106,7851183	179,9679357	deg
EA	106,7884391	179,9714553	deg
Spherical State			
RMAG	7178,174454	7178,174454	km
RA	-28,61263685	167,7368082	deg
DEC	-24,00512078	-23,90614621	deg
VMAG	7,451746737	6,977291956	km/s
AZI	82,81519034	82,26112016	deg
VFPA	89,99667914	89,99645332	deg
RAV	58,44855571	-105,4185827	deg
DECV	6,559167641	7,069963004	deg

On peut noter ici plusieurs détails. Le premier, c'est que dans le repère *EarthFixed*, la vitesse de déplacement "*VMAG*" du satellite, prend en compte la rotation de la Terre, ce qui n'est pas le cas dans le repère *EarthMJ2000Eq* (système de coordonnées équatoriales). Également, dans les coordonnées données en sphérique (pour les deux systèmes), elles sont données sous le format (Rayon, Ascension droite, Déclinaison) (*RMAG*, *RA*, *DEC*), qui est l'équivalent du système (*Rayon*, *Latitude*, *Longitude*) mais pour un système de coordonnées célestes. Pour transformer ces données en système classique de coordonnées sphériques ( $r, \theta, \varphi$ ) il faut réaliser les opérations suivantes :

$$(r, \theta, \varphi) = (RMAG, -DEC + 90^\circ, RA)$$