Grand projet : Trace au sol d'un satellite

Chris de Claverie, Valentin Steichen

Introduction **2**

Recherches et résultats **3**

Tableur 3

Spot 3 3

Notre cas 4

Algorithmes de corrections 5

Difficultés 5

Correction sur T 5

Correction sur L0 5

Algorithmes de calcul (pseudo-code) 6

Question additionnelle : Recherche de position 7

Dichotomie 7

Calcul direct 7

Application en HTML 8

API Google maps 8

HTML, CSS 8

Code Javascript 9

Exportation en tant qu'application indépendante 9

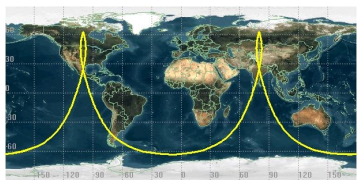
Conclusion **10**

Introduction

Le projet de trace au sol d'un satellite nous permet de comprendre comment repérer un objet céleste en orbite autour de son corps attracteur. Connaître la position d'un satellite à un instant précis a divers intérêts, tels que la définition des conditions de visibilité du celui-ci ou bien de déterminer les besoins nécessaires à la réalisation d'une mission spatiale. Il est important de noter que de connaître uniquement la trace du satellite est évidemment insuffisant. D'autres données sont nécessaires pour mener une étude, telles que l'empreinte orbitale ou tout simplement le choix de l'orbite.

Les objectifs de ce projet sont multiples. Tout d'abord, il est nécessaire de déterminer les couples latitude-longitude pour un ensemble d'anomalies vraies notées v. Pour ce faire, il nous faut d'abord déterminer avec précision les paramètres orbitaux du corps étudiés, qui dans notre cas sont donnés. Dans un deuxième temps, il faudra déterminer un procédure invariante nous permettant de retrouver les couples de coordonnées. Finalement, nous devrons les reporter sur un planisphère du type Mercator.

En fonction du type d'orbite choisie, les traces au sol relatives peuvent prendre différentes formes. Par exemple, pour un satellite géostationnaire, la trace au sol sera théoriquement un point sur le planisphère, même si en réalité un léger déplacement aura lieu. Voici quelques exemples de traces.



Afin de répondre aux objectifs du projet, nous allons réaliser un programme informatique permettant d'obtenir toutes les données nécessaires à partir des paramètres orbitaux du satellite concerné. Le programme devra être fonctionnel pour n'importe quelles valeurs d'entrées et devra nous donner en retour, un tableau regroupant toutes les valeurs souhaitées ainsi qu'une simulation de la trajectoire du satellite sur un planisphère.

Recherches et résultats

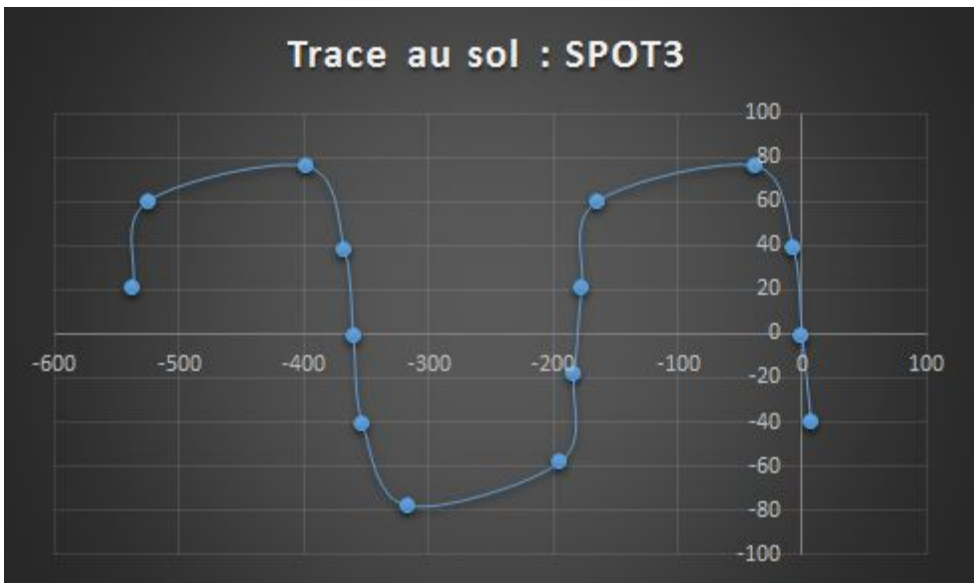
Tableur

Spot 3

Dans un premier temps, il nous a fallu, à l'aide d'une feuille de calcul Excel, retrouver les données du satellite SPOT3 et dont les valeurs nous étaient données. Cette étape nous a permis de garantir l'exactitude des valeurs obtenues pour un autre satellite. Ci-dessous est présent le détail de la feuille Excel.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1																
2	a (km)	7203,4														
3	e (sans unité)	0,0000932														
4	i (°)	98,7066		i (rad)	1,7218818		Vc (rad)	1,57088953								
5	μ (km^3 . S^(-2))	398600														
6	L(Ω) (°)	-25		L(Ω) (rad)	-0,4361111		Tp (s)	1334,22541								
7	w (°)	1,378														
8																
9																
10	V en °	-120	-80	-40	0	40	80	120	160	200	240	280	320	360	400	440
11	V en rad	-2,093333333	-1,3955556	-0,6977778	0	0,69777778	1,39555556	2,09333333	2,79111111	3,48888889	4,186666667	4,884444444	5,582222222	6,28	6,977777778	7,675555556
12	t en s	-691,179946	-16,999549	658,640031	1334,22541	2009,81078	2685,45036	3359,78718	4035,42655	4711,25132	5387,047356	6061,228139	6736,86806	7412,453572	8088,038815	8763,678055
13	la en rad	-0,705477735	-0,0173555	0,67103997	1,32643331	1,04557623	0,36364808	-0,3258446	-1,0089394	-1,3551233	-0,7086	-0,02050456	0,667913891	1,323931632	1,04861385	0,366791757
14	la en degré	-40,44139882	-0,994904	38,4672595	76,0375782	59,9374911	20,8460681	-18,678991	-57,837291	-77,682228	-40,6203822	-1,17542037	38,28805746	75,89417001	60,11162196	21,02627906
15	Lo en rad	0,130033132	0,00264258	-0,1211715	-0,6567928	-0,2658325	-0,0579731	0,05146478	0,24425821	0,76832323	0,130862533	0,003122172	-0,12039179	-0,6486835	-0,26774956	-0,05852266
16	Lo avec degré	7,454128572	0,15148521	-6,9461363	-37,65054	-164,76119	-176,6767	-182,95021	-194,00206	-315,95599	-352,498326	-359,821022	-366,90144	-397,185678	-524,651299	-536,645198
17	Lo en rad avec	0,130033132	0,00264258	-0,1211715	-0,6567928	-2,8741675	-3,0820269	-3,1914648	-3,3842582	-5,5116768	-6,14913747	-6,27687783	-6,40039179	-6,9286835	-9,15225044	-9,36147734
18	Ls en rad	2,582396906	-0,3624266	-3,3097716	-6,6686973	-11,709376	-14,740767	-17,668291	-20,684615	-25,636338	-29,0979836	-32,0431585	-34,9902049	-38,3418015	-43,3886723	-46,4214288
19	Ls en degré	-14,66595499	-24,777683	-34,69047	-68,209813	161,864596	147,133924	138,050676	124,18366	-0,586207	-39,9443568	-50,0761392	-59,9717237	-93,0709014	136,6485392	121,839477

Tableau regroupant toutes les valeurs obtenues pour SPOT3.



Avec les résultats précédents, nous avons obtenu la trace ci-dessus.

Notre cas

Suite à cela, nous avons pu obtenir une seconde trace correspondant cette fois au satellite qui nous était attribué. Il est important de passer par cette phase de calcul car les résultats que nous avons obtenus ont pu être par la suite comparés avec ceux trouvés à l'aide de notre programme.

Nous disposions des paramètres orbitaux suivants :

$$a = 12\,000\,km\ ;\ e = 0.13\ ;\ i = 40°\ ;\ \omega = -20°\ ;\ L_{\Omega} = -50°$$

μ étant le paramètre gravitationnel standard (ici géocentrique), celui-ci reste constant pour n'importe quel satellite en orbite autour de la Terre.

Algorithmes de corrections

Difficultés

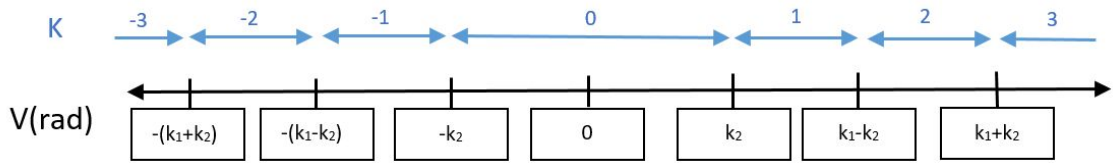
Les algoritmes de correction sur T, le temps de passage, et L_0, la longitude en terre fixe, sont donnés dans certains cas (les plus courants), de la façon suivante :

Corrections sur T		**Corrections sur Lo**
-2π-Vc < v < -2π+Vc	→ -2π+	-ω-270° < v < -ω-90° → +180°- / -180°+
-2π+Vc < v < -Vc	→ -π-	-ω-90° < v < -ω+90° → 0°+ / 0°+
-Vc < v < +Vc	→ 0+	-ω+90° < v < -ω+270° → -180°- / +180°-
Vc < v < 2π-Vc	→ +π-	
2π-Vc < v < 2π+Vc	→ +2π+	avec correction rétrograde / prograde

Il nous faut alors adapter ces corrections pour un cas général.

1. Correction sur T

La correction K qui s'applique sur T devra être telle que montrée sur le schéma suivant :



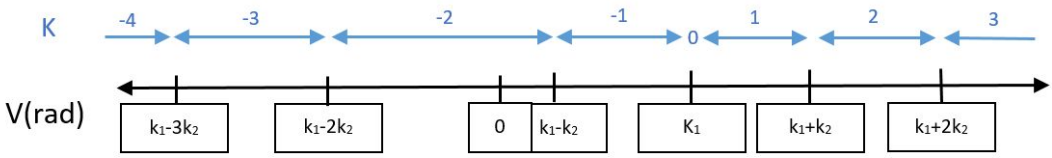
avec, ici, $k_1 = 2 * \pi$ et $k_2 = V_C$

La correction sera ensuite appliquée ainsi : $t = \sqrt{\frac{a^3}{\mu}} * (K\pi + (-1^K * t_{int}))$

avec $t_{int} = asin(\frac{\sqrt{1-e^2}*sin(v)}{1+e*cos(v)}) - e * \frac{\sqrt{1-e^2}\,sin(v)}{1+e*cos(v)}$

2. Correction sur L_0

La correction K qui s'applique sur L_0 devra être telle que montré sur le schema suivant :



avec, ici, $k_1 = -\omega - \frac{\pi}{2}$ et $k_2 = \pi$, avec ces deux valeurs en radians.
La correction sera ensuite appliquée ainsi :

- $L_0 = K\pi + (-1^K * L_{0_{int}})$ dans le cas prograde
- $L_0 = K\pi - (-1^K * L_{0_{int}})$ dans le cas rétrograde

avec $L_{0_{int}} = asin(sin(\omega + v) * \frac{cos(i)}{cos(L_a)})$

Algorithmes de calcul

Les algorithmes qui suivront seront exprimés en pseudo-code.

La correction sur le temps de passage est donc calculée comme suit :

```
SOIT x = RADIANS(v), K1 = 2π, K2 = Vc
SOIT K = 0

SI x > 0
ALORS
    TANT QUE x > K1-K2
        x = x - K1
        K = K + 1
    FIN TANT QUE
K = K * 2
SI x > K2
ALORS
    K = K + 1
FIN SI
FIN SI
```

La correction sur la longitude en terre fixe est donc calculée comme suit :

```
SOIT x = RADIANS(v), K1 = -RADIANS(omega) - RADIANS(90), K2 = RADIANS(180);
SOIT K = 0;

SI x > K1
ALORS
    TANT QUE  x > K1+K2
        x = x - K2;
        K = K + 1;
    FIN TANT QUE
SINON SI x < K1
ALORS
    TANT QUE x < k1-k2
        x = x + K2;
        K = K - 1;
    FIN TANT QUE

    K = K - 1
SINON
    K = 0
FIN SI
```

Question additionnelle : Recherche de position

Ayant déjà effectué le projet Satellite en première année pour l'un d'entre nous, une question additionnelle nous a été donnée : à partir des paramètres orbitaux et d'un temps de passage, retrouver la latitude et longitude du point survolé par le mobile. Deux options s'offrent à nous : la recherche par dichotomie, simple, mais plus lente et moins précise, et le calcul direct, théoriquement plus complexe.

Dichotomie

Le calcul par dichotomie est simple : en prenant des bornes pour *v*, nous calculons le temps de passage *t* correspondant à la borne supérieure et inférieure, ains que celui de la moyenne entre les deux bornes. Comparant ces temps de passage calculés à celui recherché, nous affinons la recherche en divisant l'intervalle en deux, soit entre la borne inférieure et la valeur moyenne, ou entre la valeur moyenne et la borne supérieure. Après assez d'itérations, une précision correcte peut être atteinte, sans avoir eu recours à de complexes calculs mathématiques

Calcul direct

Nous n'avons pas été en mesure de mener l'étude mathématique jusqu'au bout - mais avons des pistes de résolution qui pourraient nous permettre de trouver *v* directement.

En effet, nous savons que :

$$t(v) = \sqrt{\frac{a^3}{\mu}}(arcsin(\frac{\sqrt{1-e^2}sin(v)}{1+e*cos(v)}) - e*\frac{\sqrt{1-e^2}sin(v)}{1+e*cos(v)}) + T_p$$

Nous avons donc *t* selon *v*, et cherchons *v* selon *t*. Il s'agit de trouver la fonction réciproque de *t*.

Premièrement, posons : $X(v) = \frac{\sqrt{1-e^2}sin(v)}{1+e*cos(v)}$.
On a alors : $t(v) = \sqrt{\frac{a^3}{\mu}}(arcsin(X(v)) - e*X(v)) + T_p$

Ceci est de la forme : $t(v) = A\,arcsin(X(v)) + BX(v) + C$
avec $A = \sqrt{\frac{a^3}{\mu}}$, $B = -eA$, $C = T_p$

Cette forme semble être solution d'une équation différentielle … ?

Application en HTML

La façon la plus simple, rapide et élégante d'implémenter une interface utilisateur est en utilisant la puissance des langages web. Notre logiciel se présentera donc comme une page web, affichable par un navigateur. Cette méthode a ses inconvénients : il faut adapter notre code pour les différents navigateurs, ce qui demande beaucoup de travail. Nous verrons en dernière partie comment ce problème a été résolu simplement.

API Google maps

L'affichage d'une carte sera faite grâce à Google maps et son interface de programmation, aussi appelée *Application Programming Interface* (API). Celle-ci permet d'utiliser un langage d'animation de pages web, le Javascript, pour contrôler le comportement d'une carte.

Ainsi, vous trouverez le code d'initialisation de la carte dans le fichier Javascript principal, nommé *main.js*. Ce code se situe dans une fonction nommée *initMap()*, première fonction appelée dans le fichier - voir appel de la fonction à la dernière ligne. Tout ce qui se situe avant la dernière ligne ne sont que des déclarations, expliquant comment telle ou telle fonction doit agir, sans jamais les faire agir.

Veuillez noter que l'utilisation de l'API Google Maps doit obligatoirement se faire par Internet, il sera donc impossible d'utiliser notre application hors-ligne, à cause des limitations imposées par Google.

HTML, CSS

La création d'une page web se fait en deux parties : d'abord, la structure de la page, qui sera divisée en blocs. Cette structure sera définie grâce au langage HTML, dans sa version 5 utilisée ici. Dans notre application, vous pourrez retrouver le code HTML dans le fichier principal *map.html*.
C'est aussi dans ce fichier que les appels aux fichiers Javascript et CSS sont faits : c'est le point de départ de notre application.

La deuxième étape de la création d'une page web est la mise en place du style, grâce au langage CSS. Notre style est assez simple et se trouve dans le fichier *main.css*. Les autres fichiers CSS du dossier nous permettent d'assurer un minimum de compatibilité avec les autres navigateurs.

Code Javascript

Le Javascript contient deux fonctions principales : *refresh_mode1()* et *refresh_mode2()*. Le mode 1 est le mode trace : il exécute les calculs nécessaire et trace la courbe de passage

du satellite. Le mode 2, quant à lui, permet la recherche d'un mobile sur cette courbe, grâce à son temps de passage *t*. Le code de ces fonctions est la traduction directe du pseudo-code de la partie *Algorithmes de correction*, ainsi que la gestion des tableaux de données, le traçage de la courbe et la gestion de la carte. Toutes les fonctions Javascript sont commentées afin de permettre à n'importe quel lecteur non-initié de comprendre le déroulement du processus d'exécution.

Le reste du code Javascript présent dans le fichier *main.js* gère l'animation de la page : affichage du tableau de valeurs / de la carte, changement de mode (trace ou recherche), réponse aux boutons, et les fonctions mathématiques de base manquantes du langage Javascript : la conversion degrés-radians et radians-degrés.

Exportation en tant qu'application indépendante

L'application n'ayant été testée que sous chrome, ou plus précisément le moteur web nommé WebKit, les utilisateurs ne disposant pas de ces navigateurs ou ne souhaitant pas les utiliser sont oubliés. Afin de combler ce manque, nous avons décidé d'empaqueter dans une application indépendante et multi-plateforme le navigateur et le code d'affichage grâce à un outil libre de droits, *electron*. Cet outil, disponible sur la plateforme des outils Javascript nommée *NPM* (*www.npmjs.com*), est très simple d'usage et crée des fichiers executables d'environ 400Mb compressibles à moins de 40Mb, beaucoup trop lourds pour être joints à ce document. Ainsi, seules les sources de l'application ont été jointes.

Par conséquent, vous pourrez télecharger l'application complète compressée pour votre plateforme aux adresses suivante :

- Pour Windows :
  http://www.mediafire.com/file/hss2bn0ouoa9gh9/calculateur_satellite-win32-x64.7z
- Pour Mac :
  http://www.mediafire.com/file/rj121nc5vgbbz6y/calculateur_satellite-darwin-x64.7z
- Pour Linux :
  http://www.mediafire.com/file/cnxao30d7xbsoey/calculateur_satellite-linux-x64.7z

Veuillez noter que pour une meilleure compression, le format *7z* a été utilisé, et une application telle que *7-zip* sera nécessaire pour la décompression (*www.7-zip.org*)

Conclusion

Ce projet s'est révélé très enrichissant dans la mesure où il a consisté en une approche concrète du métier d'ingénieur. En effet, la prise d'initiative, le respect des délais seront des aspects essentiels de notre futur métier. De plus, il nous a permis d'appliquer quelques connaissances de mécanique spatiale.

Bien qu'il ne s'agisse pas d'une étude approfondie, le programme que nous avons pu développer dans le cadre de ce projet pourra nous être utile plus tard dans notre cursus, par exemple pour l'étude orbitale de notre CubeSat, dans le cadre de l'association IpsaONE. De plus, bien que le grand projet se termine, nous pensons toujours maintenir le programme à jour et bien sûr continuer à l'améliorer en y ajoutant de nouvelles fonctionnalités.