

LES CENTRALES SOLAIRES ORBITALES

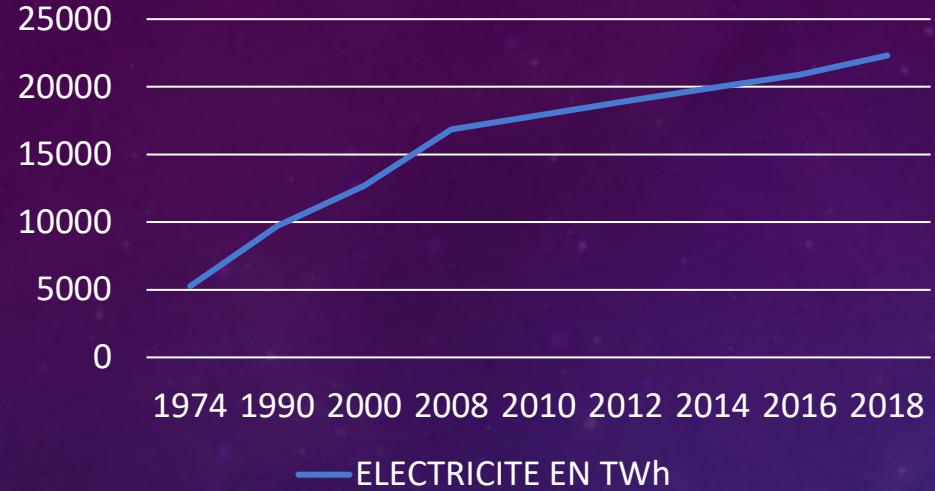
FUTURE SOURCE D'ÉNERGIE POUR
LES VILLES ?



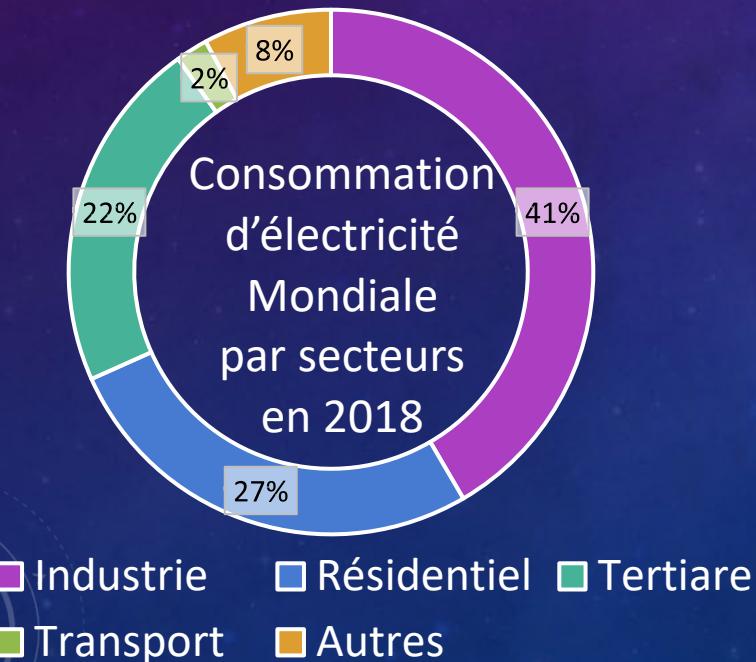
MOHAMED EMINÉ BASSOUM

N°11681

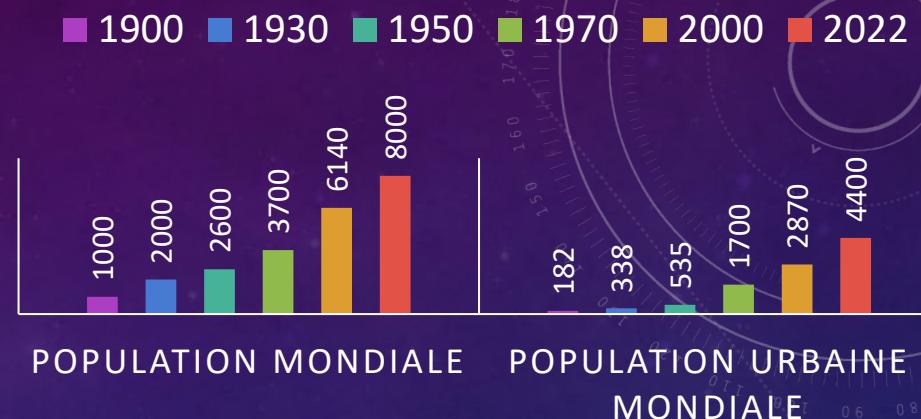
Consommation Mondiale d'Energie



SOURCE : GLOBAL ELECTRICITY REVIEW



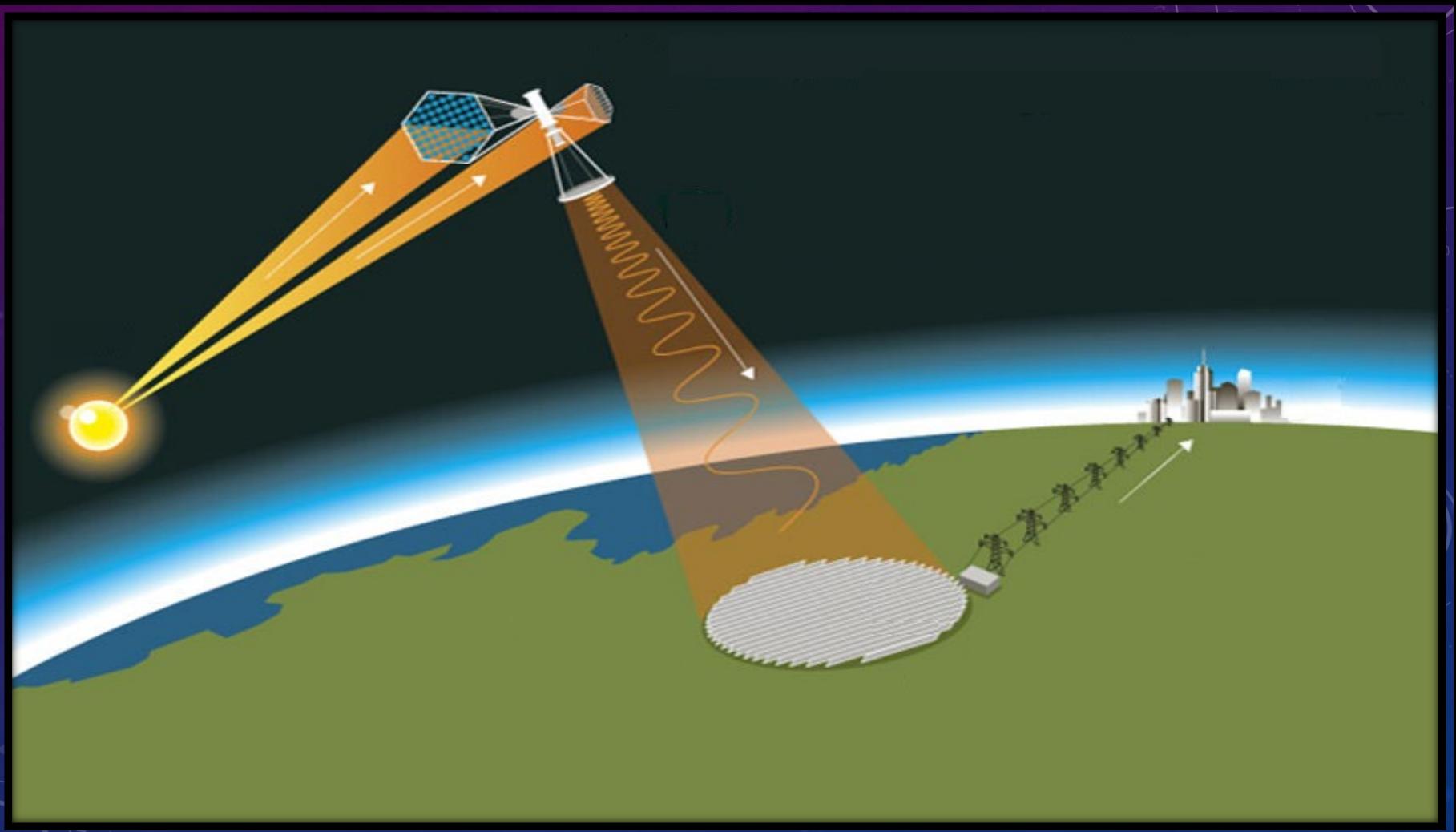
EVOLUTION DE LA POPULATION



SOURCE: BANQUE MONDIALE DE DONNÉES

- Remarque 1 : La croissance de la population mondiale et urbaine est parallèle.
- Remarque 2 : En 2022, l'urbanisation atteint 55% de la population mondiale.
- Remarque 3 : Environ 90% de la consommation électrique se concentre en zone urbaine.

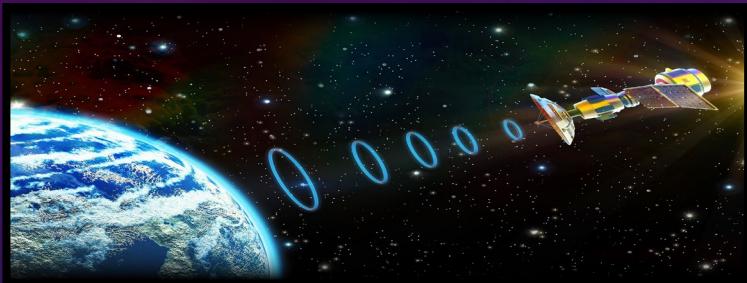
CONCEPT :



Source	Propreté	Dangerosité	Fiabilité	Suivie de charge
Faucille	Émissions de CO2	pollution de l'air et Effet de Serre	Variable	disponibilité des ressources
Nucléaire	Déchet nucléaire	Risques liés à la radioactivité	Élevée	dépendante de la capacité des réacteurs
Eolienne	Propre	danger potentiel pour les oiseaux	Très Faible	disponibilité du vent
Solaire sur terre	Propre	Non Dangereuse		Dépendance météorologique
Hydraulique	Propre	Impact environnemental sur les écosystèmes aquatiques	Élevée	sècheresse
Solaire Spatiale	Propre	Non Dangereuse		peut varier en fonction des conditions spatiales

PLAN

- Déterminer le meilleur moyen de transmettre l'énergie des centrales solaires orbitales sur terre.



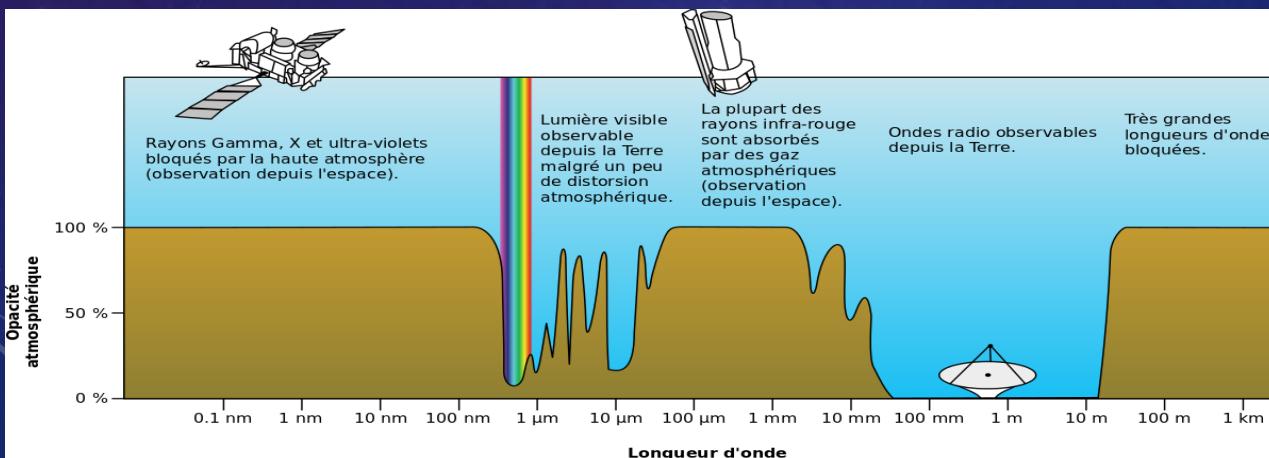
- Etablir un modèle afin de trouver la meilleure orbite.
- Evaluer le nombre et les dimensions des satellites nécessaires pour répondre aux demandes énergétiques d'une ville.



COMPARAISON DES MODES DE TRANSMISSION

Micro-onde

- Efficacité de conversion relativement élevée
- Transmission sur de longues distances avec peu de pertes
- Robuste face aux conditions atmosphériques



Lasers

- Densité d'énergie plus élevée
- Faisceau plus ciblé et concentré
- Propagation plus directe et stable



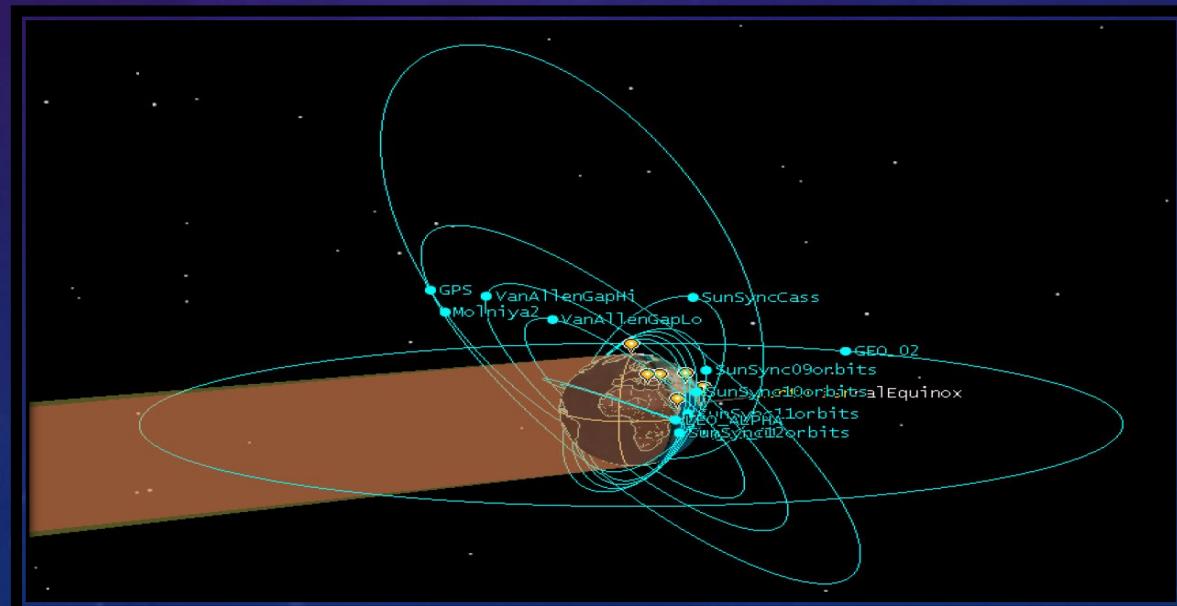
USNRL LASER 20%
eff @ 400m



JPL micro-onde 54%
eff @ 1,5km

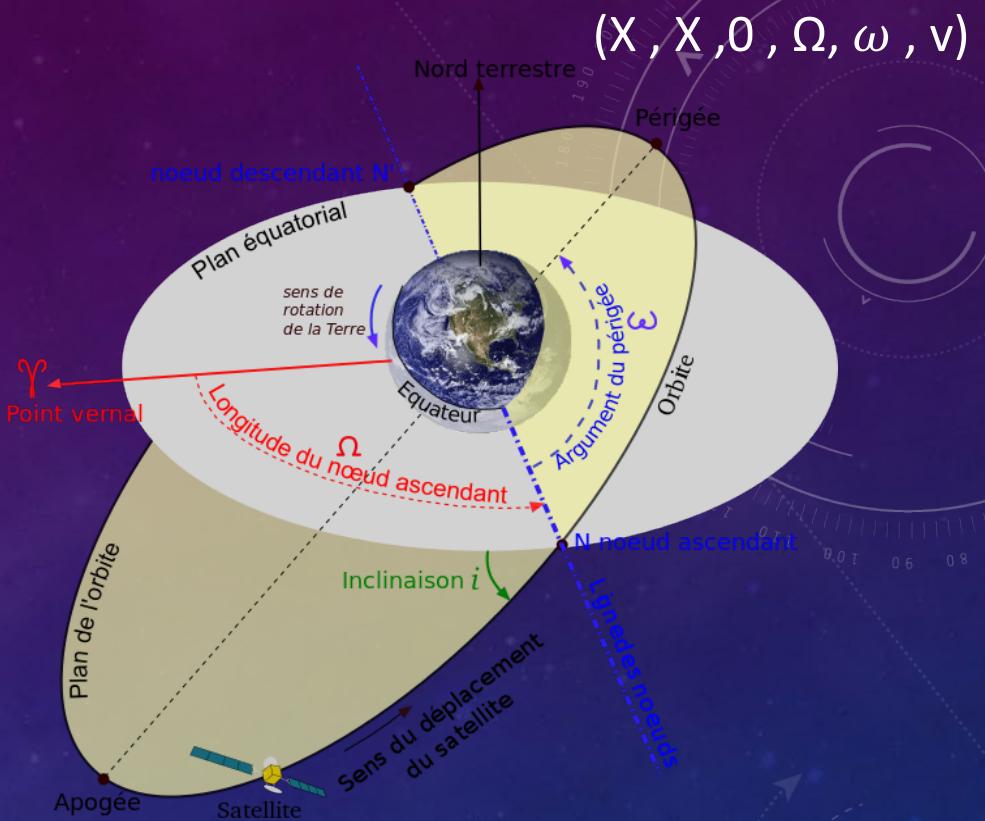
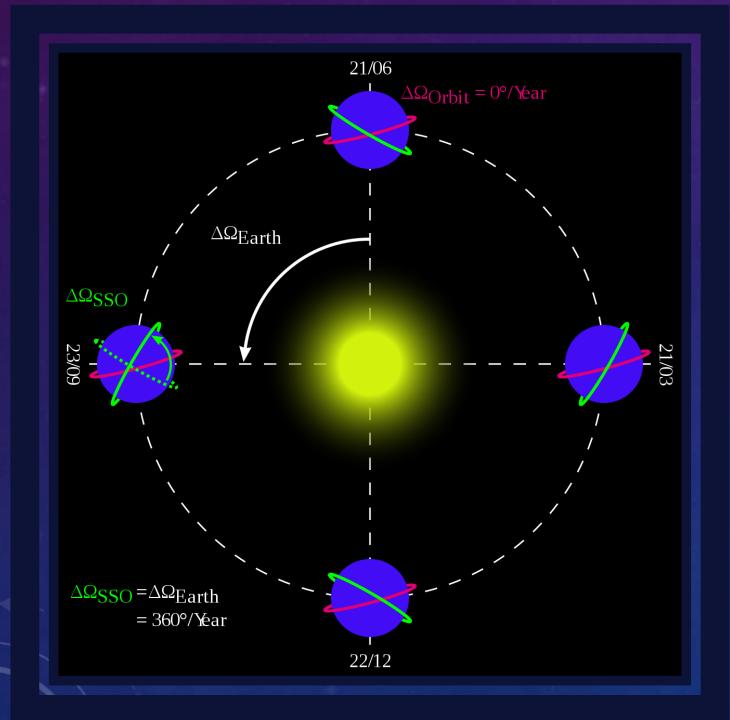
Orbit Description	Apogee Altitude (km)	Perigee Altitude (km)	Circular Altitude (km)	Inclination (deg)	No. Orbits/Day	Period (min)
LEO	500.0	500.0	500.0	28.5	15.2	94.6
Sun Sync Repeat 1	1,676.5	1,676.5	1,676.5	102.9	12.0	119.9
Low MEO	2,158.6	2,158.6	2,158.6	28.5	11.0	130.8
Sun Sync Repeat 2	2,158.6	2,158.6	2,158.6	105.9	11.0	130.8
Sun Sync Repeat 3	2,719.9	2,719.9	2,719.9	110.1	10.0	143.9
Sun Sync Repeat 4	3,383.6	3,383.6	3,383.6	116.0	9.0	160.0
HEO Elliptical	7,414.0	963.0	4,188.5	116.6	8.0	180.2
Low Van Allen Gap	7,000.0	7,000.0	7,000.0	55.0	5.6	256.7
High Van Allen Gap	12,000.0	12,000.0	12,000.0	55.0	3.5	413.2
Molniya	39,850.5	500.0	20,175.3	63.5	2.0	717.7
GPS	20,200.0	20,200.0	20,200.0	55.0	2.0	718.7
GEO	35,786.0	35,786.0	35,786.0	<1	1.0	1,436.1

Rapport US NAVY 2019



HÉLIOSYNCHRONISME

$$\dot{\Omega}_{Helio} = \frac{360^\circ}{365,24} deg/jour$$



$$\dot{\Omega} = -\frac{3}{2} J_2 \left(\frac{R_e}{P} \right)^2 n \cos(i) \text{ avec } p = (1 - e^2)$$

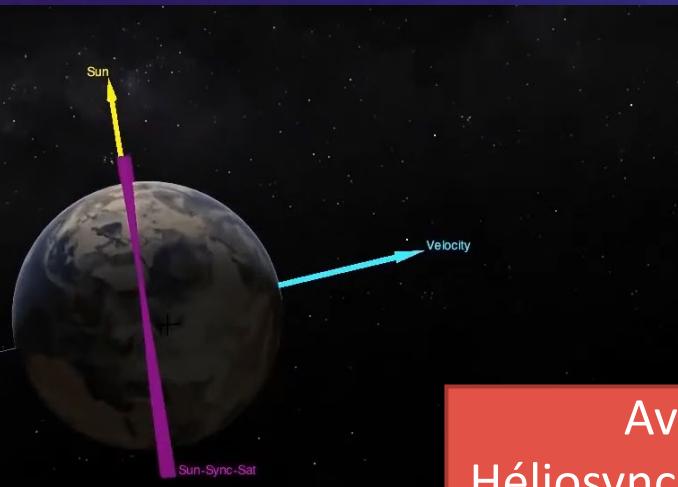
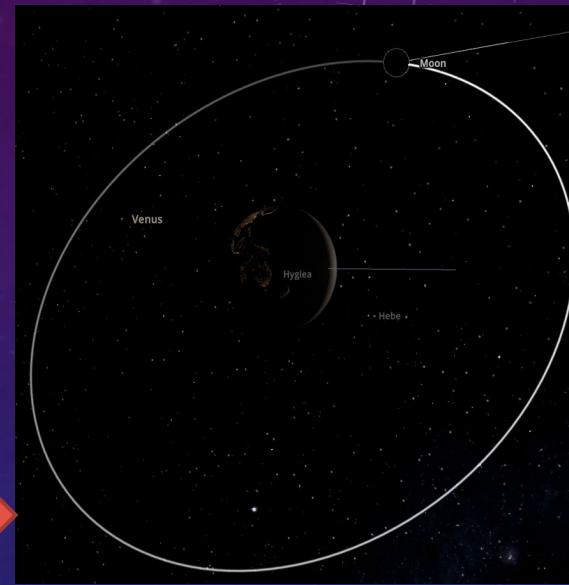
$$\dot{\Omega}_{Helio} = -\frac{3}{2} J_2 \left(\frac{R_e}{a} \right)^2 \sqrt{\frac{\mu}{a^3}} \cos(i)$$

$$i(a) = \arccos\left(\frac{2\dot{\Omega}}{-3 \cdot J_2 \left(\frac{R_e}{P} \right)^2 \sqrt{\frac{\mu}{a^3}}}\right)$$

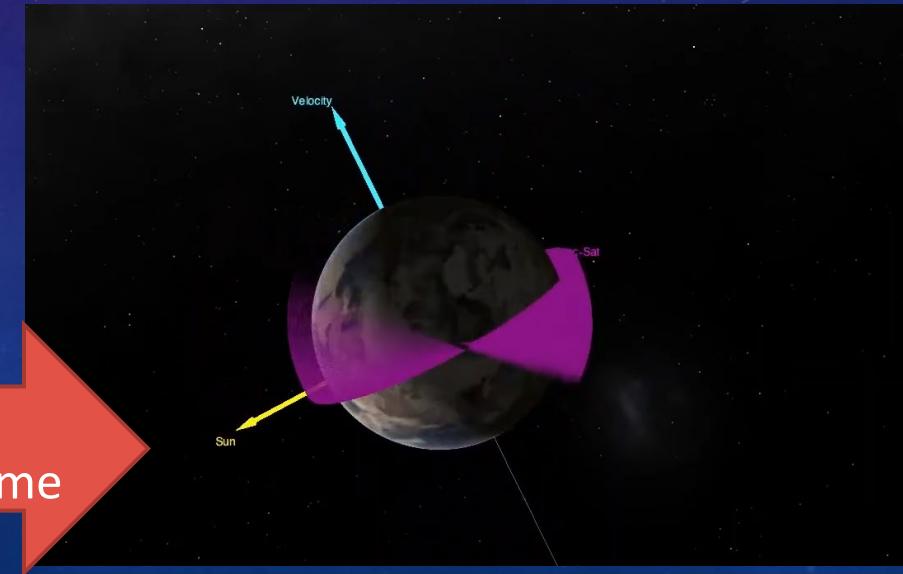
HÉLIOSYNCHRONISME



Sans
Héliosynchronisme

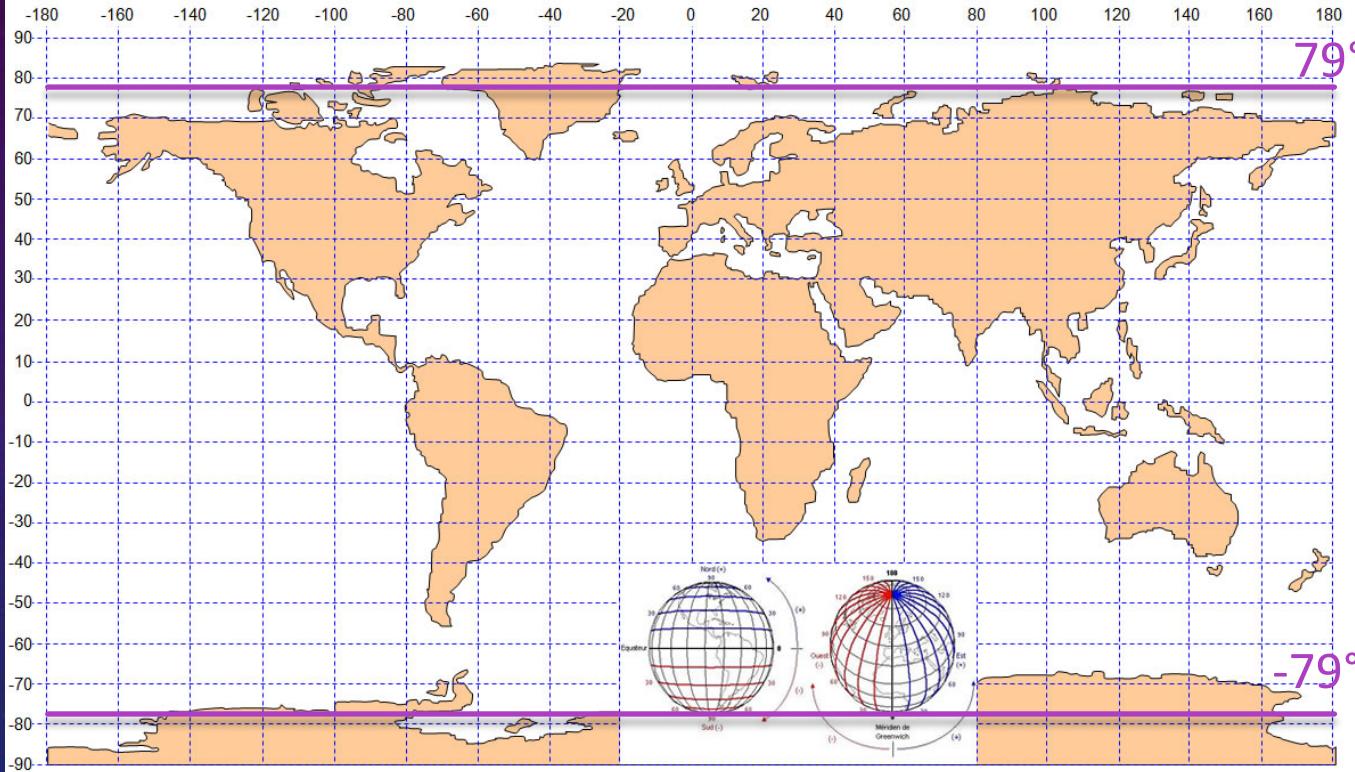


Avec
Héliosynchronisme



DÉTERMINATION DE A ET I

Inclinaison	Altitude °
90	-6371
91	-2490
92	-1640
93	-1059
94	-605,1
95	-226,3
96	101,15
97	391,3
98	652,75
99	891,34
100	1111,2
101	1315,4
102	1506,2
103	1685,5
104	1854,7
105	2014,9



Condition :

Trace au sol répété : $k = \frac{m}{n}$ Avec $n = 2\pi \sqrt{\frac{a^3}{\mu}} \div 24h$ (en sec)

Période orbitale répétée : 19 jour

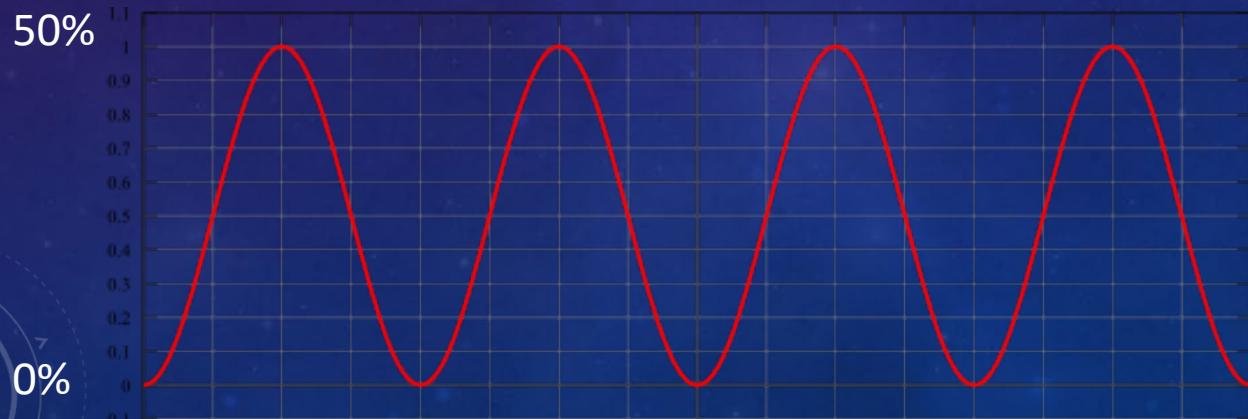
i = 101,4°
 a = 7759 km

DÉTERMINATION DE Ω

Ω	0°	30°	60°	90°	120°	150°	180°
Ombre%							
Pénombre %							
PMS (w/m^2)							

Résultats de la simulation

210°	240°	270°	300°	330°	360°

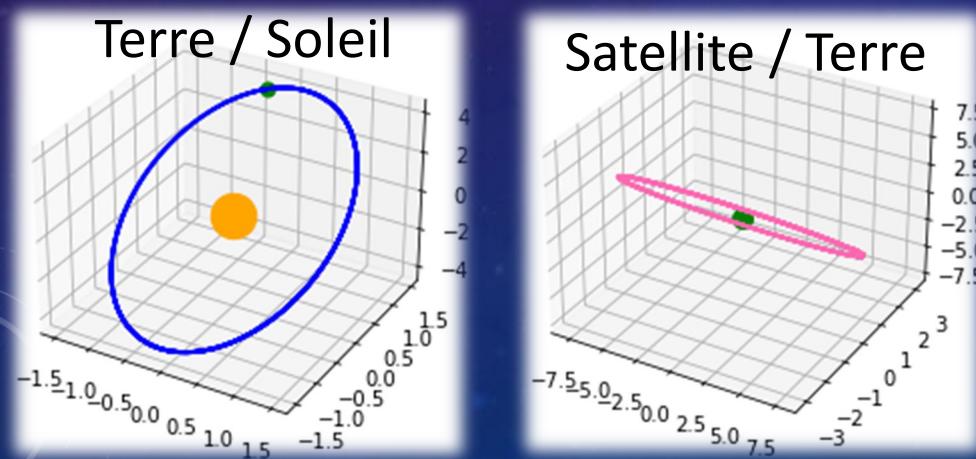


Résultats attendus

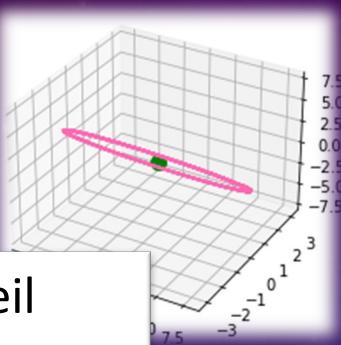
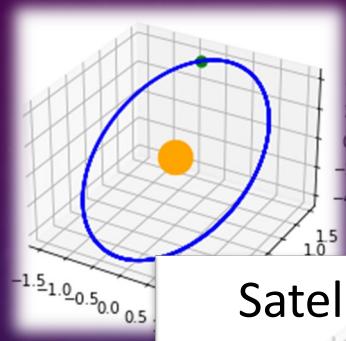
FONCTIONNEMENT DE LA SIMULATION DES ORBITE

- 1ere partie : simulation de l'orbite d'un corps céleste
- On commence par l'équation de mouvement (PFD):

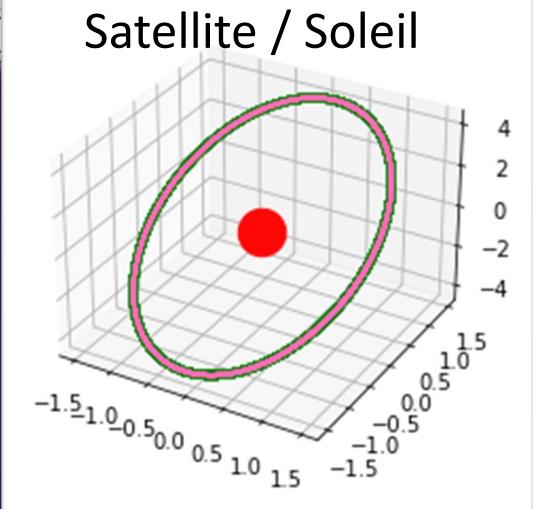
- $\ddot{r} = -\frac{\mu}{r^2}$ avec \ddot{r} selon \vec{ur}
- En passant en coordonnées cartésiennes on peut trouver numériquement la position et la vitesse du point P après un instant dt



SUPERPOSITION ET RÉSULTATS



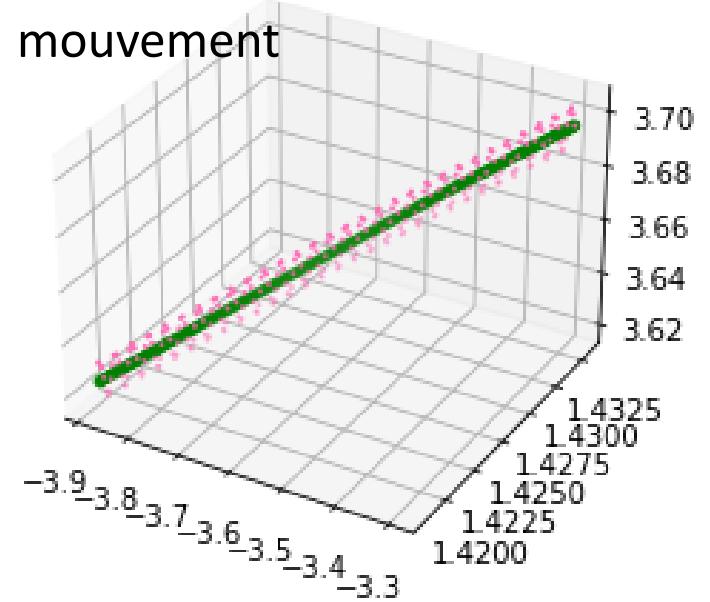
Satellite / Soleil



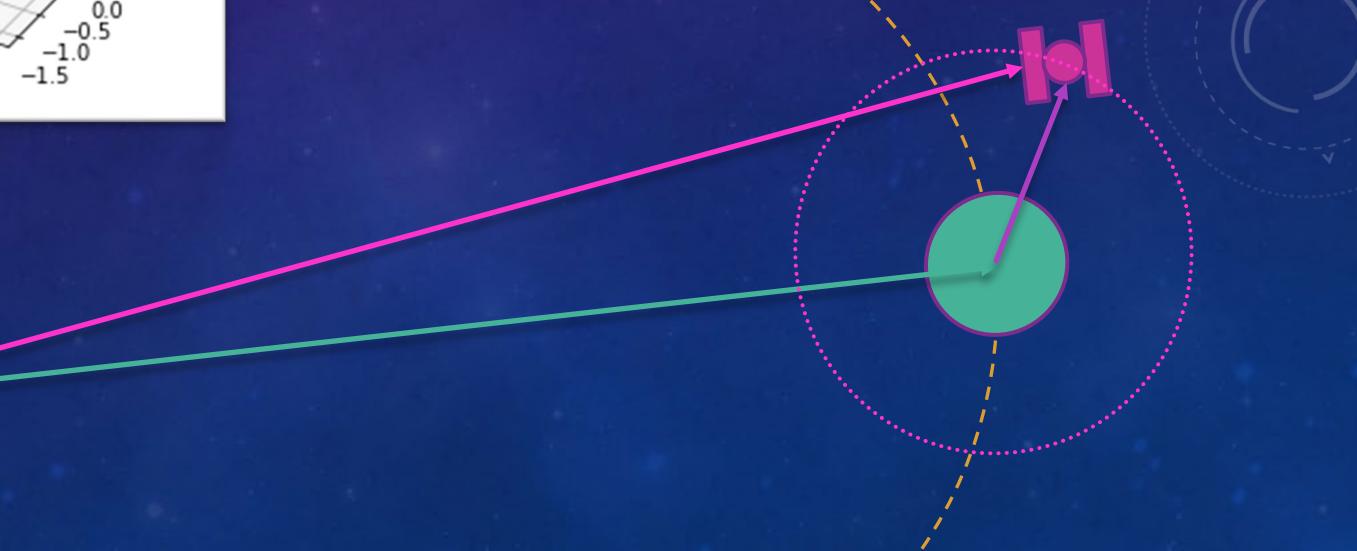
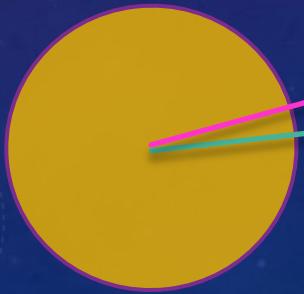
Zoom



Satellite orbitant la terre en mouvement



13



DÉTECTION DES ZONES D'OMBRES

1^{ère} méthode :
(difficile à implémenter
Temps d'exécution élevé)



Calcul de puissance :

$$\text{Loi de Stephan : } PST = \sigma \times T^4 \times 4\pi R_s^2$$

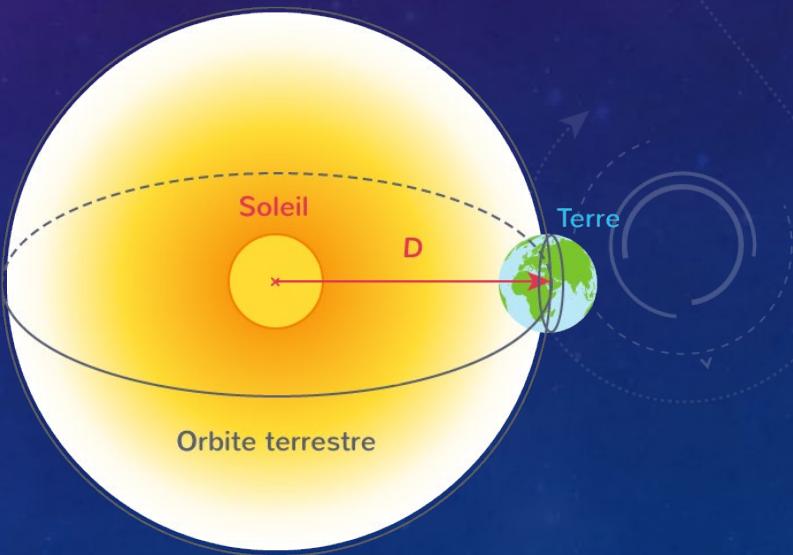
Si SAT est Dans une zone d'ombre :

Passe

Sinon :

$$ES = ES + PST / 4\pi R_{sat}^2 \times dt$$

A la fin : PMS = ES / une Année



Sphère sur laquelle la puissance solaire est répartie

$$S_{\text{sphère}} = 4 \times \eta \times D^2$$

DÉTECTION DES ZONES D'OMBRES

2^{ème} méthode :

Vérifier si le point q se trouve entre les plans des deux facettes circulaires du cylindre:

$$(\vec{q} - \vec{p1}) \cdot (\vec{p2} - \vec{p1}) \geq 0$$
$$(\vec{q} - \vec{p2}) \cdot (\vec{p2} - \vec{p1}) \leq 0$$

Vérifier si le point q se trouve à l'intérieur du cylindre:

$$\frac{|(\vec{q} - \vec{p1}) \cdot (\vec{p2} - \vec{p1})|}{|\vec{p2} - \vec{p1}|} \leq r$$

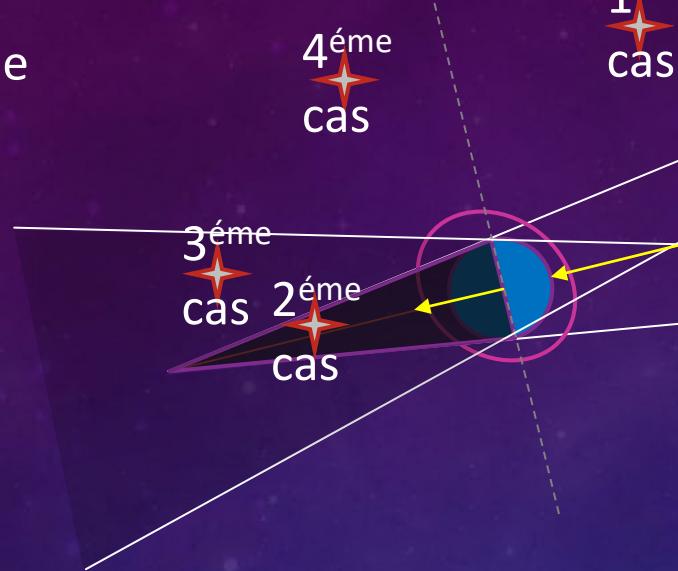
100% |██████████| 39421/39421 [00:01<00:00, 38471.68it/s]
lumiere% = 95.2961265795
ombre% = 4.42971805432
penombra% = 0.27415536610

Orbite de l'ISS



DÉTECTION DES ZONES D'OMBRES

3^{ème} méthode
 {Nasa JPL} :



1^{er} cas SAT entre la terre et le soleil

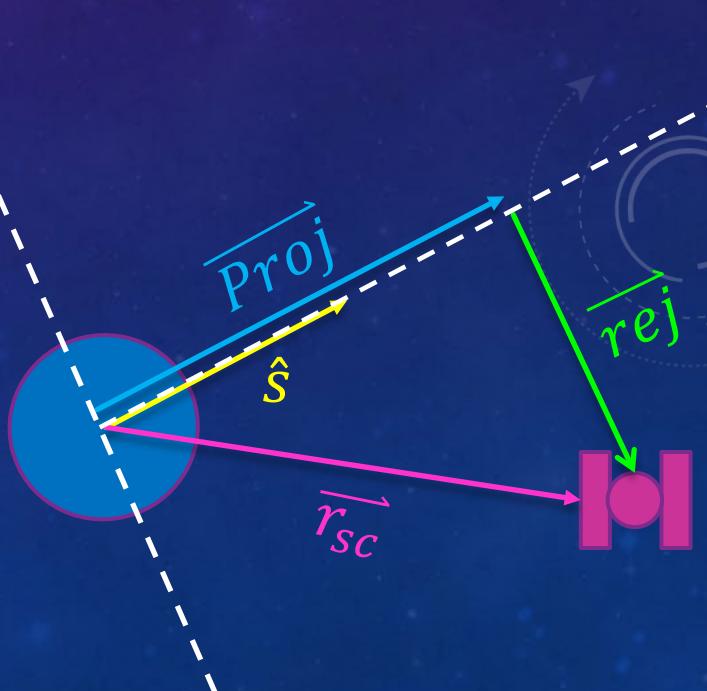
$$Proj_{scalaire} = \overrightarrow{r_{sc}} \cdot \hat{s}$$

Si $Proj_{scalaire} \leq 0$:

SAT éclairé

Sinon on définit :

$$\begin{aligned} \overrightarrow{Proj} &= Proj_{scalaire} \cdot \hat{s} \\ \overrightarrow{rej} &= \overrightarrow{r_{sc}} - \overrightarrow{Proj} \end{aligned}$$



DÉTECTION DES ZONES D'OMBRES

3^{ème} méthode

{Nasa JPL} :

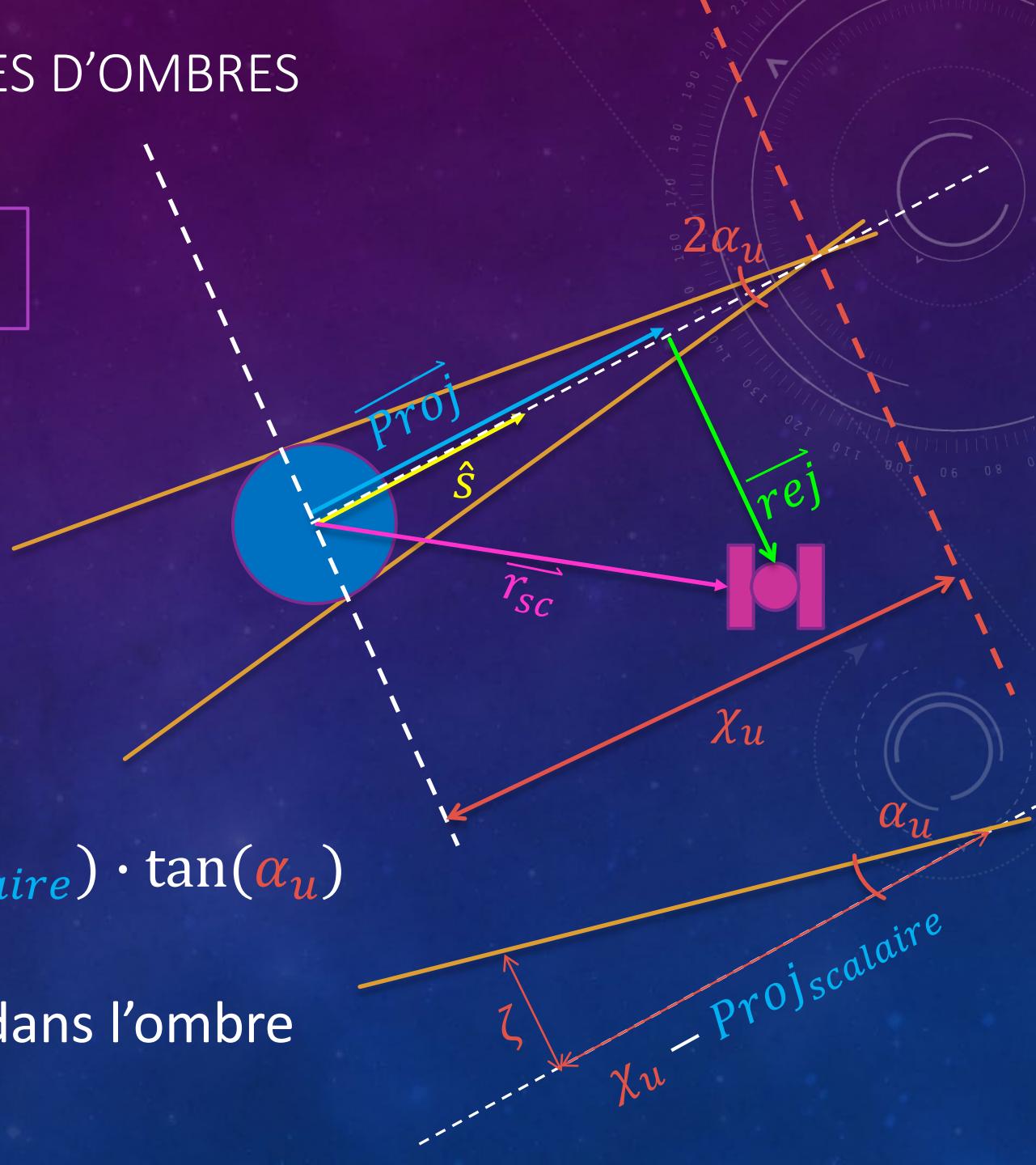
2^{ème} cas :
ombre

$$\chi_u = \frac{R_T \cdot \|\vec{s}\|}{R_S - R_T}$$

$$\alpha_u = \sin^{-1}\left(\frac{R_T}{\chi_u}\right)$$

$$\zeta = (\chi_u - \text{Proj scalaire}) \cdot \tan(\alpha_u)$$

Si $\|\overrightarrow{\text{rej}}\| \leq \zeta$: SAT dans l'ombre



DÉTECTION DES ZONES D'OMBRES

3^{ème} méthode
 {Nasa JPL} :

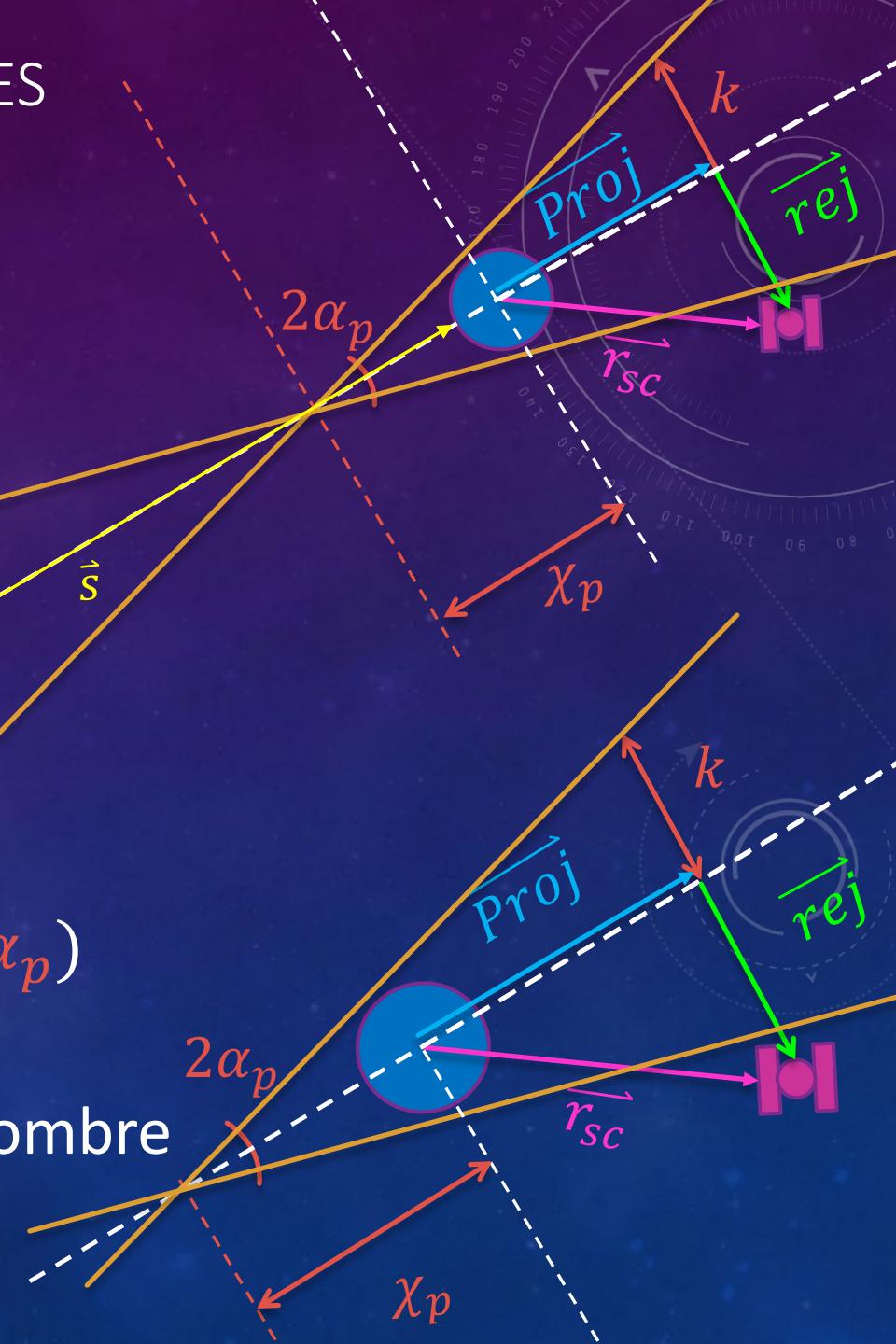
3^{ème} cas :
 pénombre

$$\chi_p = \frac{R_T \cdot \|\vec{s}\|}{R_S + R_T}$$

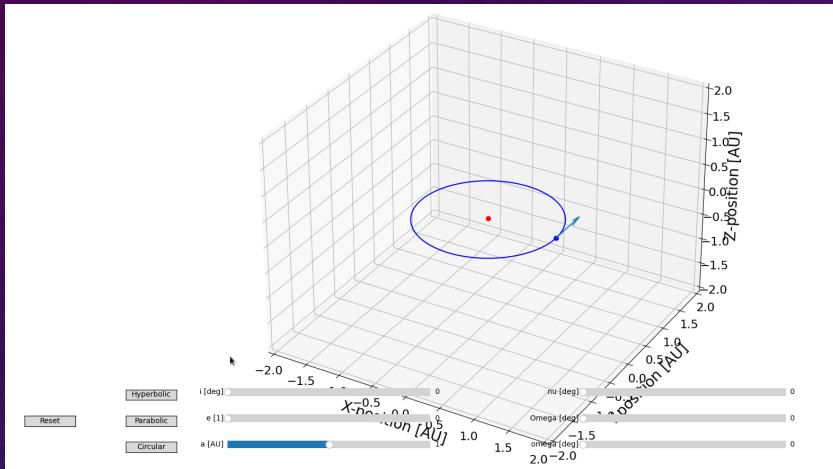
$$\alpha_p = \sin^{-1}\left(\frac{R_T}{\chi_p}\right)$$

$$k = (\chi_p + Proj_{scalaire}) \cdot \tan(\alpha_p)$$

Si $\|\vec{rej}\| \leq k$: SAT dans la pénombre



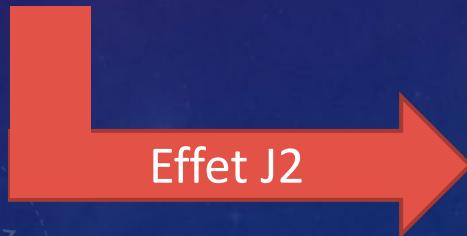
PROBLÈMES RENCONTRÉS ET TENTATIVES DE CORRECTION



Condition
initiale

Mass	
M	m
$1.98891691172467 \times 10^{30}$ kg	$1.69863768365072 \times 10^{27}$ kg
State vectors	
R_x	617244712358 m
R_y	-431694791368 m
R_z	-12036457087 m
V_x	7.320 km/s
V_y	11.329 km/s
V_z	-0.211 km/s
Orbital elements	
a	778194564622.574 m
e	0.048425472281527694
i	1.305185101881517
Ω	100.49431512298086
ω	275.36540415423883
M_0	313.3830012166574
ν	309.1791558055481
l	325.0388750827678
q	740510125303.805 m
Q	81587903941.34 m
P	374239090.6392552 s

$$P = \frac{3 \cdot J_2 \cdot \mu \cdot R_{terre}^2}{2 \cdot R^5} \left[\left(5 \frac{Z^2}{R^2} - 1 \right) (X\hat{I} + Y\hat{J}) + Z \left(5 \frac{Z^2}{R^2} - 1 \right) \hat{K} \right]$$



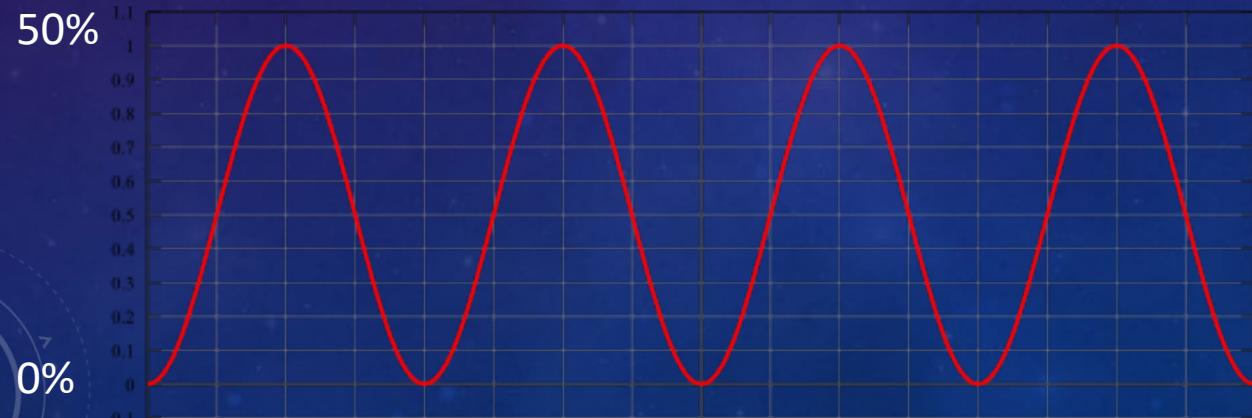
```
100% [██████████] 39421/39421 [00:01<00:00, 38075.35it/s] inlight% = 65.89127622333274
ombre% = 33.72314248750666
penombra% = 0.38558128916059964
OMBRE% = 34.10872377666726
Energie surfacique = 181080819.0930161J/m²
Puissance surfacique moyenne = 918.7023114229273W/m²
Puissance% = 67.2234320247415%
```

DÉTERMINATION DE Ω

Ω	0°	30°	60°	90°	120°	150°	180°
Ombre%	16	16,2	17	17,2	17,35	17,35	17,2
Pénombre %	0,4	0,38	0,4	0,35	0,38	0,42	0,39
PMS (w/m ²)	1160	1159	1149	1145	1143	1142	1145

Résultats de la simulation

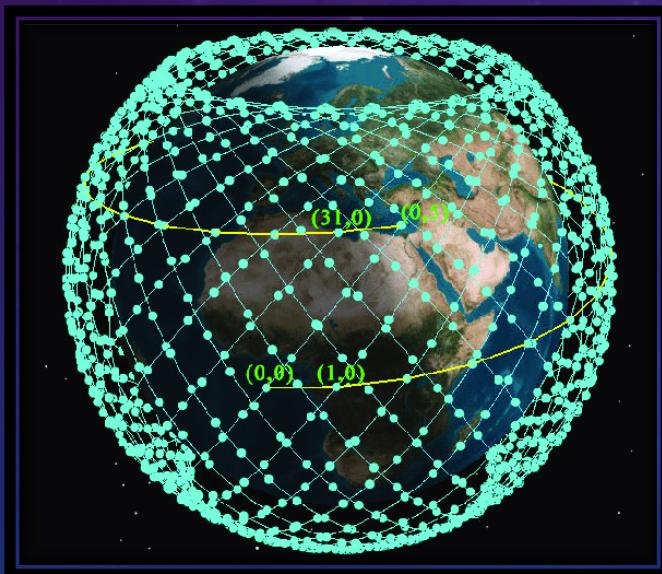
210°	240°	270°	300°	330°	360°
17,1	17,15	17,09	16,94	16,81	16,3
0,38	0,42	0,35	0,4	0,34	0,36
1147	1145	1147	1148	1151	1158



Résultats attendus

NOMBRE DE SATELLITE NÉCESSAIRE POUR UNE COUVERTURE PERMANENTE

- Période orbitale répétée : 19 jour
- Rapport US Navy 2019 : temps de passage $\sim 11\text{mn}$ pour une orbite d'altitude $\sim 1300\text{km}$
- Il faut donc 2487 Satellite pour une couverture permanente en Energie électrique.



Constellation de STARLINK



Concept de Relay

CHOIX DE LA CONSTELLATION

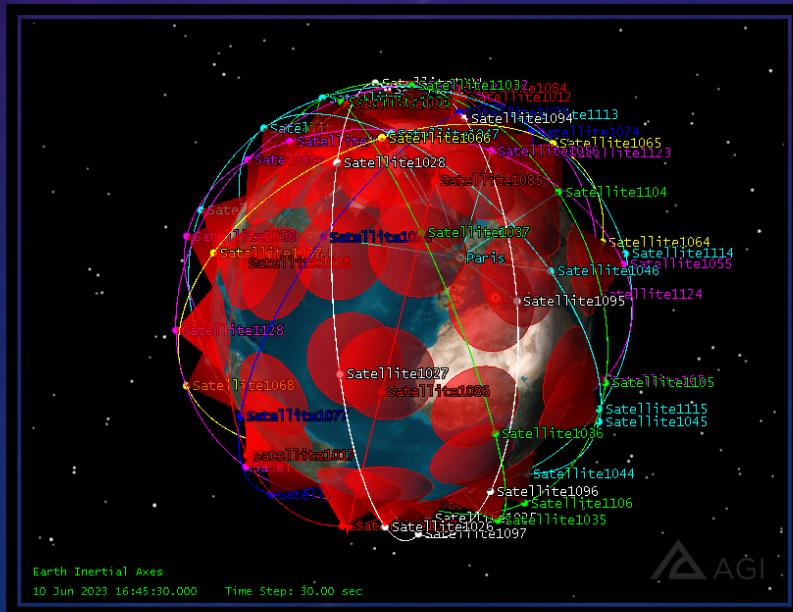
2 types de constellation se présentent pour les orbites héliosynchrones basse :

- Constellation Fleures
- Constellation Walker delta (utilisé pour le système GPS)

pour avoir une couverture avec moins de satellite on utilisera la Walker delta.

Sur STK on peut configurer une Walker 100/30/30

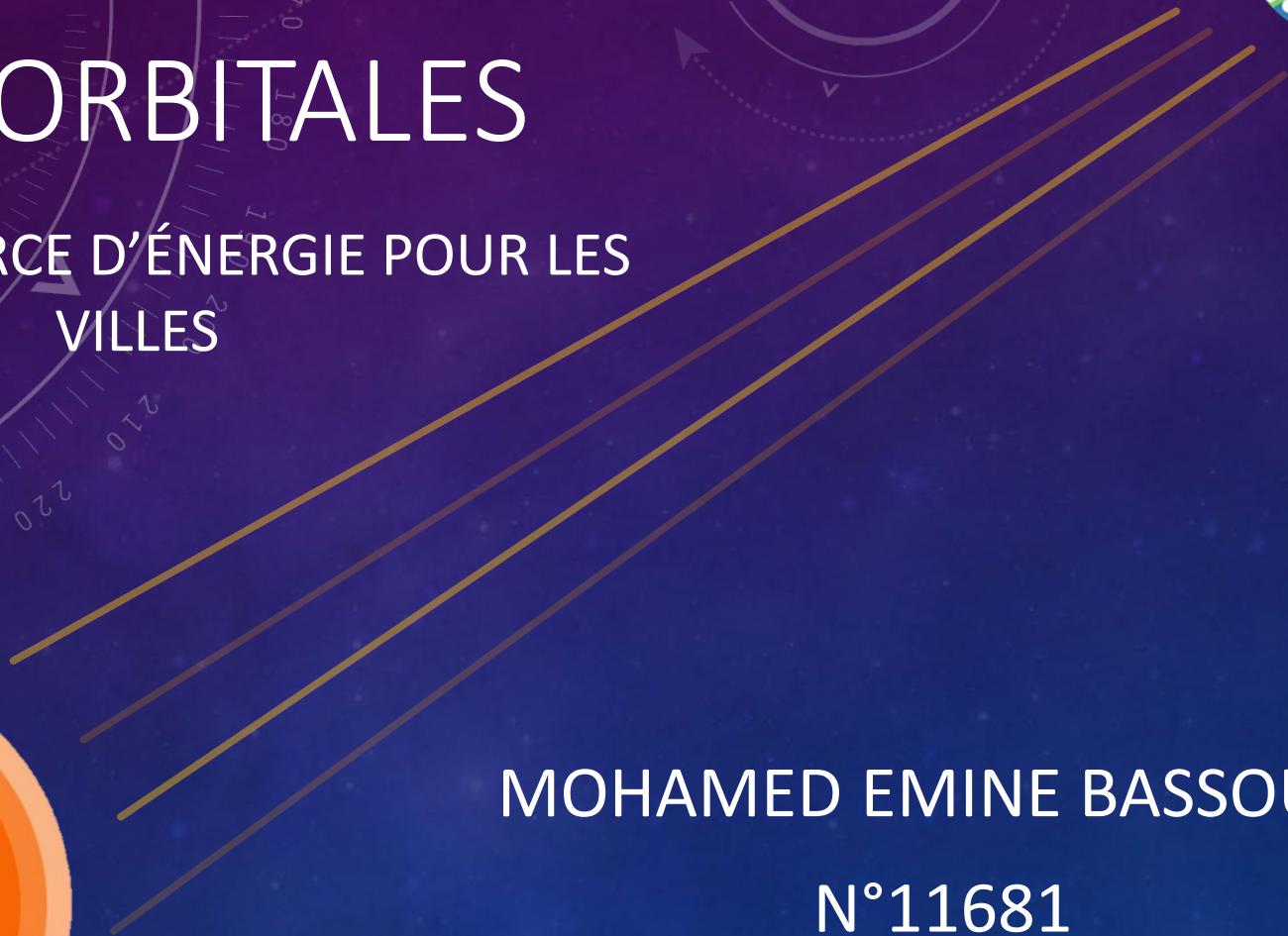
Résultat : couverture par au moins 2 satellites pendant toute la durée des 19 jours



Type d'entreprise	Cout approximative (\$CAN/hre)
Petite entreprise conventionnelle	500 \$
Communication cellulaire	65 500 \$
Billetterie téléphonique(générale)	115 000 \$
Réservation de billets d'avion (Système informatisé)	144 000 \$
Transactions par carte de crédits	4 120 000 \$
Transaction par courtiers de placement	10 350 000 \$
Usines de fabrication de puces micro-processeurs	50 000 000 \$

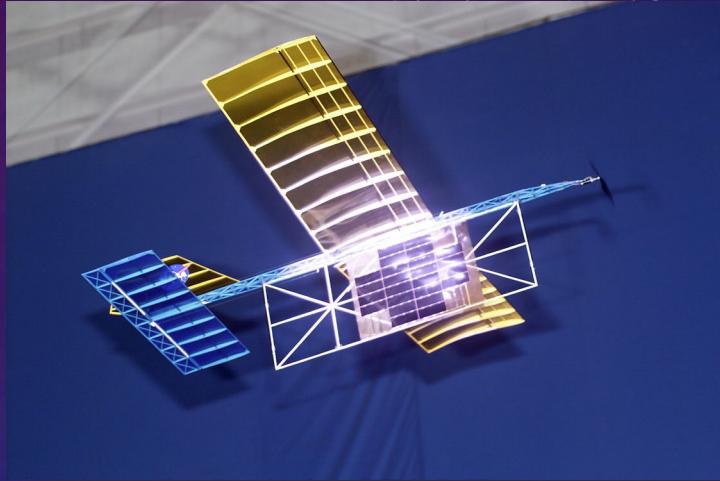
LES CENTRALES SOLAIRES ORBITALES

FUTURE SOURCE D'ÉNERGIE POUR LES
VILLES



MOHAMED EMINE BASSOUM
N°11681

AUTRES CAS PRATIQUE



Deployable Power Generation and Distribution System (DPGDS)



MEP-PU-810A Prime Power Unit



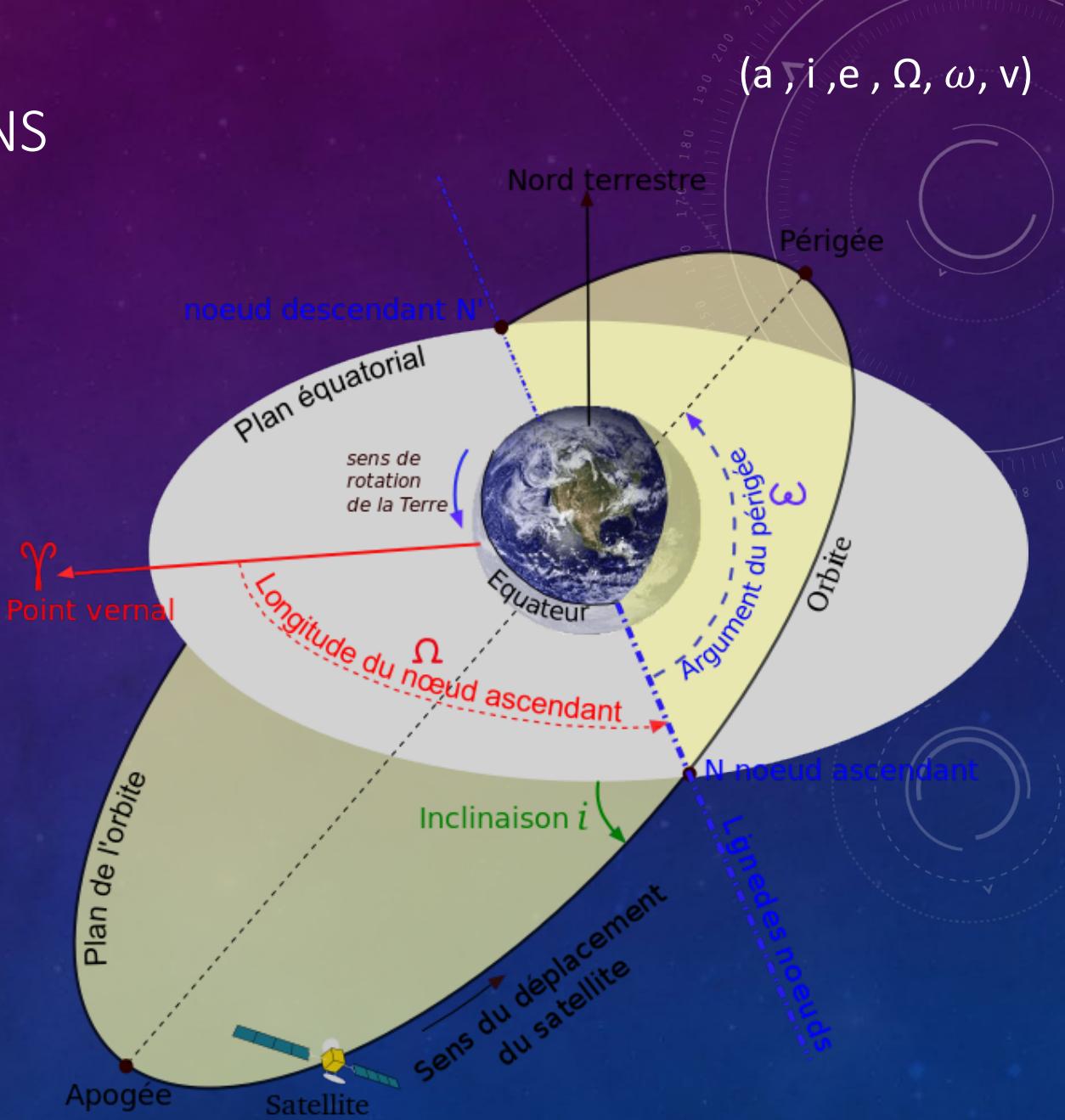
MEP-PU-810B Prime Power Unit

Système / Sous-système	{TRL X/9} maturité technologique	Notes
Communication	9	des décennies d'héritage
Générateur d'électricité	9	PV
Système de gestion thermique	6	Reduction de masse probablement nécessaire
Transmission et Réception d'énergie	6/5	Démo Sous-échelle en RF Démo Laser au labo
Sécurité	5/4	Démo préliminaire effectuer pour RF/Laser
Système de lancement	8	Fusé Falcon 9 ou Heavy réutilisable
Centrale solaire	5/4	RF / Laser

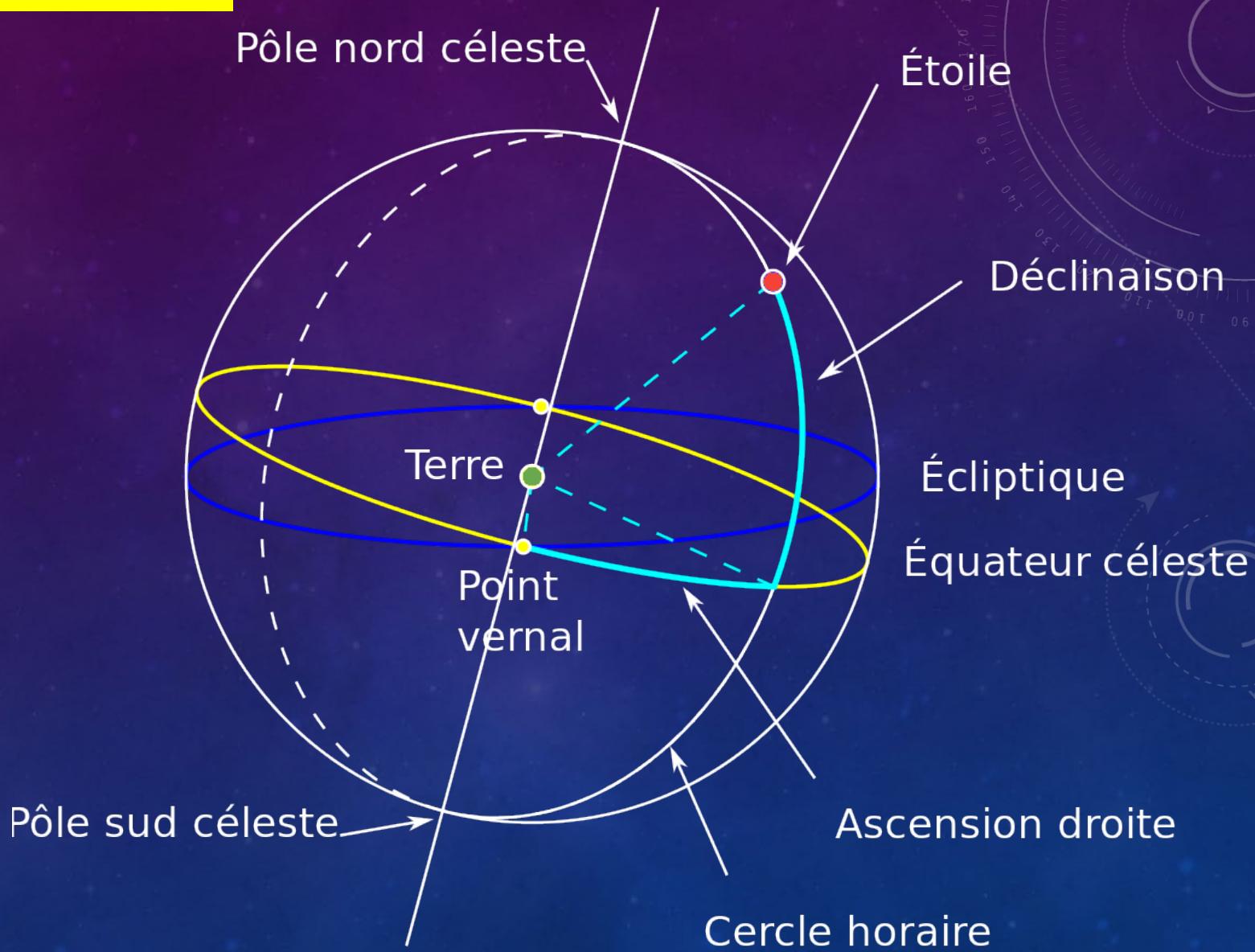
($a, \gamma, i, e, \Omega, \omega, v$)

ÉLÉMENT KÉPLÉRIENS

- a : demi-grand axe
- e : excentricité
- i : inclinaison
- Ω : Longitude du nœud ascendant
- ω : argument du périgée
- v : anomalie vrai



POINT VERNAL



NON SPHÉRICITÉ DE LA TERRE

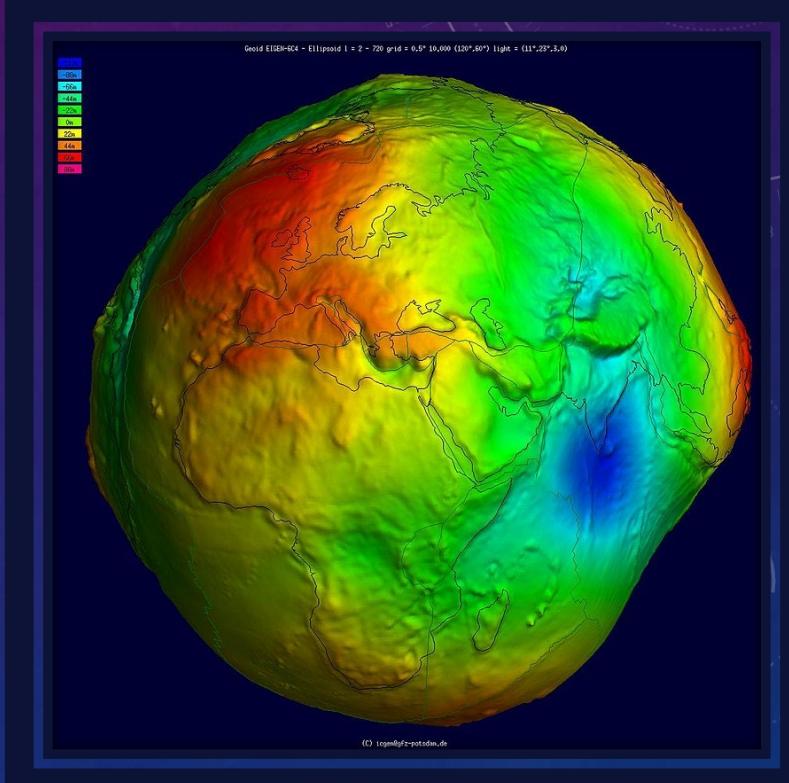
- On sait que le potentiel $V = -\frac{\mu}{r}$
avec $\mu = M \times G$
- Or le vrai potentiel comporte des termes perturbateur $\rightarrow V = -\frac{\mu}{r} + \Phi(r, \varphi)$
- (Roger Battin 1999) :

$$\Phi(r, \varphi) = \frac{\mu}{r} \sum_{k=2}^{\infty} J_k \left(\frac{R}{r}\right)^k P_k(\cos(\varphi))$$

- J_2 étant le terme dominant on peut donc conserver :

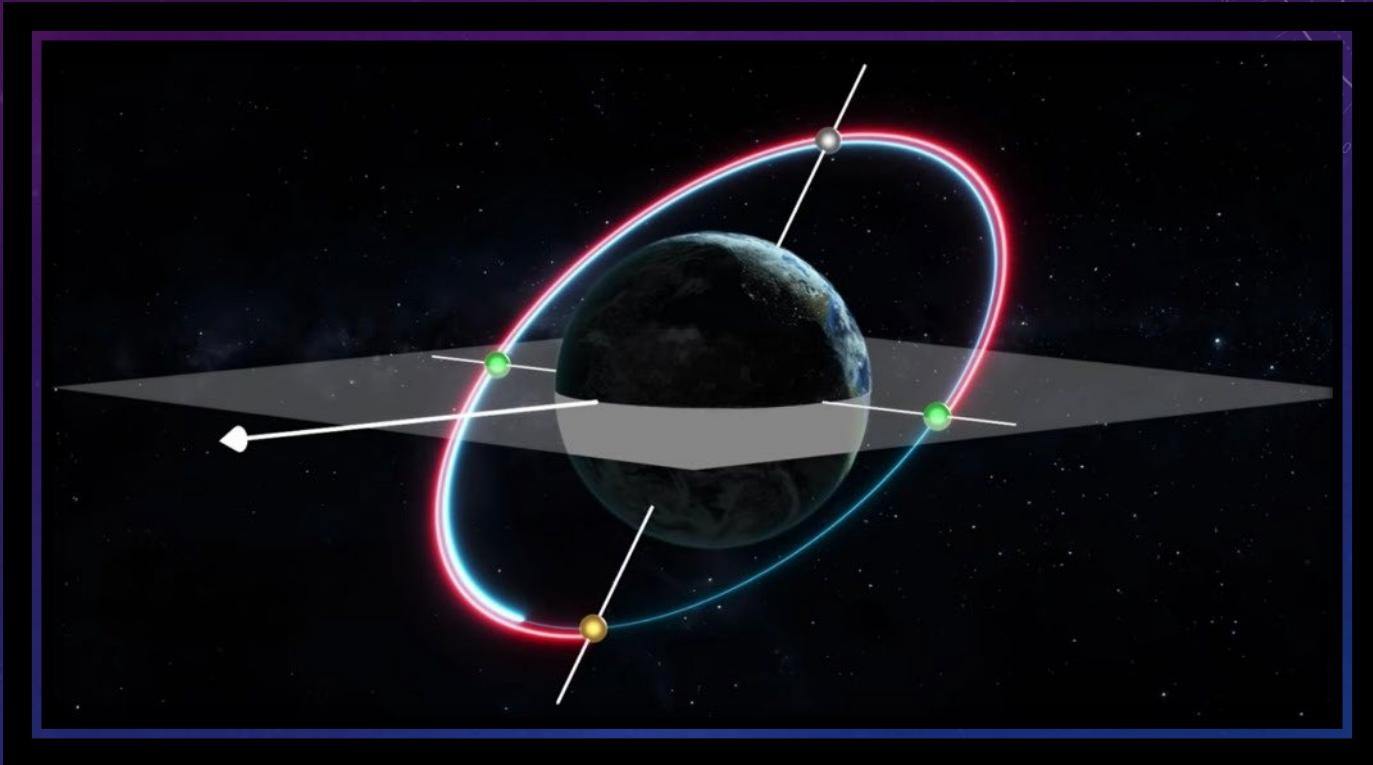
$$\Phi(r, \varphi) = \frac{\mu}{r} J_2 \left(\frac{R}{r}\right)^2 (\cos^3(\varphi) - 1)$$

(Formule de Rodrigues)



(a , i , e , Ω , ω , v)

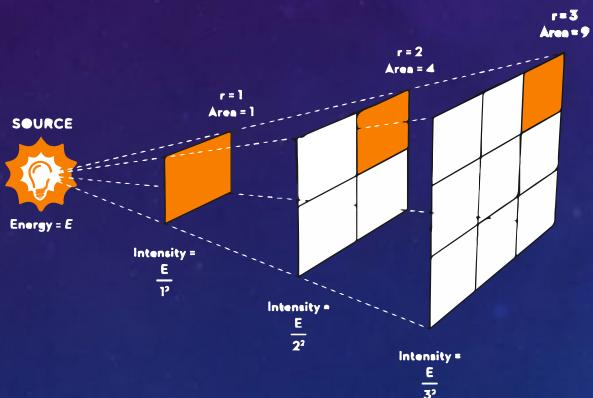
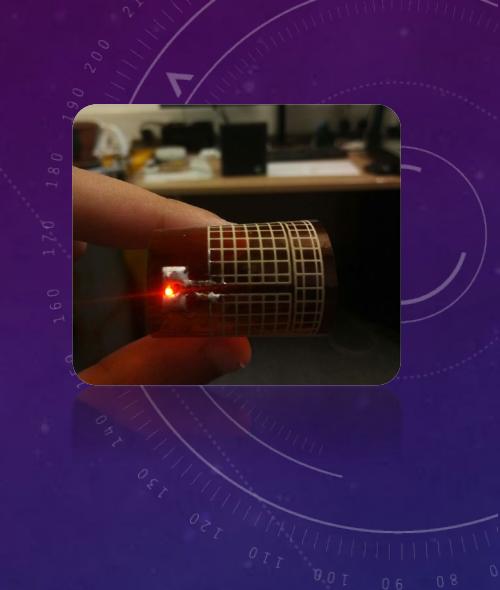
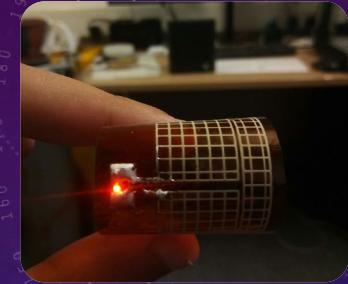
CONSÉQUENCE DE LA NON-SPHÉRICITÉ DE LA TERRE



$$\omega_p = -\frac{3}{2} \frac{{R_E}^2}{(a(1-e^2))^2} J_2 \omega \cos i$$

RÉCEPTION DE L'ENERGIE SUR TERRE

Energie RF



Inverse Square Law



Safety Standard

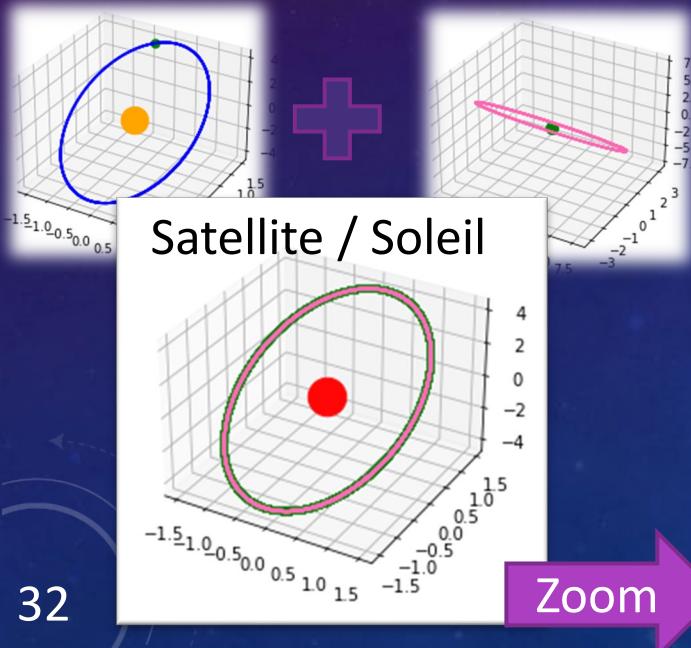
Safety Standard	W/m ²
IEEE Std C95.1™-2005 for 2 GHz to 100 GHz uncontrolled area	10
IEEE Std C95.1™-2005 for 3 GHz to 300 GHz controlled area	100
ANSI Z136.1-2014 limit for 1400nm to 1000μm	1,000

PSEUDO-CODE DE LA SIMULATION DES ORBITES + SUPERPOSITION

```

184 #ajustement temporelle
185 k = round(len(pt[0])/len(p[0]))+1
186 p[0] *= k; p[1] *= k; p[2] *= k
187 del p[0][len(pt[0]):]
188 del p[1][len(pt[1]):]
189 del p[2][len(pt[2]):]
190
191 #Superposition des orbites terre / sat
192 print("superpositionnement")
193 for i in tqdm(range(len(pt[0]))):
194     p[0][i] += pt[0][i]
195     p[1][i] += pt[1][i]
196     p[2][i] += pt[2][i]

```



```

18 def plan(a,p01,p02):
19     R = m.sqrt(p01**2+p02**2)
20     if p01 == 0 :
21         a1 = 0; a2 = a
22     elif p02 == 0 :
23         a1 = a; a2 = 0
24     else :
25         theta = abs(m.asin(p02/R))
26         a1 = abs(a * m.cos(theta))
27         a2 = abs(a * m.sin(theta))
28         if p01 > 0 :
29             a1 = -a1
30         if p02 > 0 :
31             a2 = -a2
32     return(a1,a2)

```

```

76 Dt = 5
77 ts = 160
78 print("Creation de l'orbite terrestre")
79 for n in tqdm(range(365*24*60*60//Dt)):
80
81     R = m.sqrt(p0xt**2+p0yt**2+p0zt**2)
82     a = Ms * G / (R**2)
83     if p0xt == 0 : #probleme plan sur Y, Z
84         ax = 0
85         ay , az = plan(a,p0yt,p0zt)
86     elif p0yt == 0 : # probleme plan sur Z, X
87         ay = 0
88         az , ax = plan(a,p0zt,p0xt)
89     elif p0zt == 0 : #probleme plan sur X,Y
90         az = 0
91         ax , ay = plan(a,p0xt,p0yt)
92     else :
93         phi = m.atan2(p0yt,p0xt)
94         theta = m.atan2(m.sqrt(p0xt**2+p0yt**2),p0zt)
95         ax = abs(a * m.cos(phi) * m.sin(theta))
96         ay = abs(a * m.sin(phi) * m.sin(theta))
97         az = abs(a * m.cos(theta))
98
99     if p0xt > 0 :
100         ax = -ax
101
102     if p0yt > 0 :
103         ay = -ay
104
105     if p0zt > 0 :
106         az = -az
107
108     p0xt = p0xt + v0xt * Dt + (ax * Dt**2)/2
109     p0yt = p0yt + v0yt * Dt + (ay * Dt**2)/2
110     p0zt = p0zt + v0zt * Dt + (az * Dt**2)/2
111
112     v0xt = v0xt + ax * Dt
113     v0yt = v0yt + ay * Dt
114     v0zt = v0zt + az * Dt
115
116     pt[0].append(p0xt)
117     pt[1].append(p0yt)
118     pt[2].append(p0zt)

```

Satellite orbitant la terre en mouvement

PSEUDO-CODE DU DÉTECTEUR DE ZONE D'OMBRE

```

245 for i in tqdm(range(len(tspt[0]))):
246
247     #Déclaration des variables
248     p0xt = tspt[0][i]
249     p0yt = tspt[1][i]
250     p0zt = tspt[2][i]
251     T = np.array([p0xt,p0yt,p0zt])
252
253     p0x = tsp[0][i]
254     p0y = tsp[1][i]
255     p0z = tsp[2][i]
256     PS = np.array([p0x,p0y,p0z])
257     PT = PS - T
258
259     DTSSat = m.sqrt((p0x-p0xt)**2+(p0y-p0yt)**2+(p0z-p0zt)**2)
260     DSS = m.sqrt((p0x)**2+(p0y)**2+(p0z)**2)
261     DTS = m.sqrt((p0xt)**2+(p0yt)**2+(p0zt)**2)
262
263     Us = T/np.linalg.norm(T)
264     projv = np.dot(Us, PT) * Us
265     proj = np.dot(Us, PT)
266     rej = PT - projv
267     nrej = np.linalg.norm(rej)
268
269     # 1er cas le satellite est entre la terre et le soleil :
270     if proj <= 0 :
271         Pss = pst/(4 * 3.14 * (DSS**2) )
272         E += Pss * Dt #energie/ m²
273         l += 1
274
275     else :
276         # 2eme cas le satellite est derrière la terre plusieurs cas ce presentes
277         xu = (rt*DTS) / (rs - rt)
278         au = m.asin(rt/xu)
279         ki = (xu - proj)*m.tan(au)
280         if nrej <= ki : # satellite dans l'ombre
281             E += 0
282             o += 1
283         else :
284             xp = rt*DTS/(rs + rt)
285             ap = m.asin((rt)/(xp))
286             kp = (xp + proj)* m.tan(ap)
287             angle = m.atan2(nrej, proj)
288             if nrej <= kp : #satellite dans la penombre
289                 Pss = pst/(4 * 3.14 * (DSS**2) )
290                 E += Pss * Dt / 2
291                 pen += 1
292             else : # satellite dans la lumiere du soleil
293                 Pss = pst/(4 * 3.14 * (DSS**2) )
294                 E += Pss * Dt #energie/ m²
295                 l += 1
296
297     print(f"inlight% = {l/len(tspt[0])*100}")
298     print(f"ombre% = {o/len(tspt[0])*100}")
299     print(f"penombra% = {pen/len(tspt[0])*100}")
300     print(f"OMBRE% = {(pen+o) /(len(tspt[0])) * 100}")
301     print(f"Energie surfacique = {E}J/m²")
302     print(f"Puissance surfacique moyenne = {E/(len(tspt[0]*Dt))}W/m²")
303     print(f"Puissance% = {(E/(len(tspt[0]*Dt)))/1366.64 * 100}%")

```

Exécution sur l'orbite $\Omega = 0^\circ$

```

100%|██████████| 39421/39421 [00:00<00:00, 41851.39it/s]inlight% =
83.98529210319373
ombre% = 15.85195707871439
penombra% = 0.36275081809188
OMBRE% = 16.01470789680627
Energie surfacique = 230050987.28227782J/m²
Puissance surfacique moyenne = 1167.149424328545W/m²
Puissance% = 85.60284378684547%

```

Exécution sur l'orbite de l'ISS

```

100%|██████████| 39421/39421 [00:01<00:00, 38476.08it/s]inlight% =
65.99274498363816
ombre% = 33.61660028918597
penombra% = 0.39065472717587074
OMBRE% = 34.00725501636184
Energie surfacique = 181367645.29522264J/m²
Puissance surfacique moyenne = 920.1575063809779W/m²
Puissance% = 67.32991178225267%

```

CODE COMPLET

```
#imports
import math as m
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm

def plan(a,p01,p02):
    R = m.sqrt(p01**2+p02**2)
    if p01 == 0 :
        a1 = 0; a2 = a
    elif p02 == 0 :
        a1 = a; a2 = 0
    else :
        theta = abs(m.asin(p02/R))
        a1 = abs(a * m.cos(theta))
        a2 = abs(a * m.sin(theta))
        if p01 > 0 :
            a1 = -a1
        if p02 > 0 :
            a2 = -a2
    return(a1,a2)

#configuration
ua = 149597870700
G = 6.67e-11
J2 = 1.08262668e-3
Mt = 5.972e24
Ms = 2e30
rt = 6371e3
rs = 6.957e8
μ = G * Mt

p0xt = -32968121947
p0yt = 143310330830
p0zt = 3696215645

v0xt = -29.516e3
v0yt = -6.78e3
v0zt = -0.342e3

pt = [[p0xt],
       [p0yt],
       [p0zt]]

p0x = 3765219
```

```
p0y = 5469009
p0z = -1391292
v0x = -3315.81
v0y = 3755.156
v0z = 5802.16

Ts = 93*60

p = [[p0x],
      [p0y],
      [p0z]]

#Energie puissance shadow
H = m.sqrt(p0x**2 + p0y**2 + p0z**2)
E = 0
pst = 3.8651*10**26
#primary sun secondary earth

Dt = 5
ts = 160
print("Creation de l'orbite terrestre")
for n in tqdm(range(365*24*60*60//Dt)):

    R = m.sqrt(p0xt**2+p0yt**2+p0zt**2)
    a = Ms * G / (R**2)
    if p0xt == 0 : #probleme plan sur Y, Z
        ax = 0
        ay , az = plan(a,p0yt,p0zt)
    elif p0yt == 0 : # probleme plan sur Z, X
        ay = 0
        az , ax = plan(a,p0zt,p0xt)
    elif p0zt == 0 : #probeleme plan sur X,Y
        az = 0
        ax , ay = plan(a,p0xt,p0yt)
    else :
        phi = m.atan2(p0yt,p0xt)
        theta = m.atan2(m.sqrt(p0xt**2+p0yt**2),p0zt)
        ax = abs(a * m.cos(phi) * m.sin(theta))
        ay = abs(a * m.sin(phi) * m.sin(theta))
        az = abs(a * m.cos(theta))

        if p0xt > 0 :
            ax = -ax
        if p0yt > 0 :
```

```
if p0yt > 0 :
    ay = -ay

if p0zt > 0 :
    az = -az

p0xt = p0xt + v0xt * Dt + (ax * Dt**2)/2
p0yt = p0yt + v0yt * Dt + (ay * Dt**2)/2
p0zt = p0zt + v0zt * Dt + (az * Dt**2)/2

v0xt = v0xt + ax * Dt
v0yt = v0yt + ay * Dt
v0zt = v0zt + az * Dt

pt[0].append(p0xt)
pt[1].append(p0yt)
pt[2].append(p0zt)

print("creation de l'orbite du Satellite")
for n in tqdm(range(Ts//Dt + 1)):

    R = m.sqrt(p0x**2+p0y**2+p0z**2)
    a = Mt * G / (R**2)

    if p0x == 0 : #probleme plan sur Y, Z
        ax = 0
        ay , az = plan(a,p0y,p0z)
    elif p0y == 0 : # probleme plan sur Z, X
        ay = 0
        az , ax = plan(a,p0z,p0x)
    elif p0z == 0 : #probeleme plan sur X,Y
        az = 0
        ax , ay = plan(a,p0x,p0y)
    else :
        phi = m.atan2(p0y,p0x)
        theta = m.atan2(m.sqrt(p0x**2+p0y**2),p0z)
        ax = abs(a * m.cos(phi) * m.sin(theta))
        ay = abs(a * m.sin(phi) * m.sin(theta))
        az = abs(a * m.cos(theta))

        if p0x > 0 :
            ax = -ax
        if p0y > 0 :
```

CODE COMPLET

```

if p0z > 0 :
    az = -az

# #J2 perturbation
pr = 1.5 * J2 * mu * (rt**2) / R # R**5 depasse 10**100 probleme
pr /= R
pr /= R
pr /= R

pi = pr * p0x * (5 *((p0z**2)/ (R**2)) - 1)
pj = pr * p0y * (5 *((p0z**2)/ (R**2)) - 1)
pk = pr * p0z * (5 *((p0z**2)/ (R**2)) - 3)

ax += pi
ay += pj
az += pk
# #J2 perturbation ends

p0x = p0x + v0x * Dt + (ax * Dt**2)/2
p0y = p0y + v0y * Dt + (ay * Dt**2)/2
p0z = p0z + v0z * Dt + (az * Dt**2)/2

v0x = v0x + ax * Dt
v0y = v0y + ay * Dt
v0z = v0z + az * Dt

p[0].append(p0x)
p[1].append(p0y)
p[2].append(p0z)

#ajustement temporelle

k = round(len(pt[0])/len(p[0]))+1
p[0] *= k; p[1] *= k; p[2] *= k
del p[0][len(pt[0]):]
del p[1][len(pt[1]):]
del p[2][len(pt[2]):]

#Superposition des orbites terre / sat
print("superposition")
for i in tqdm(range(len(pt[0]))):

```

```

#Superposition des orbites terre / sat
print("superposition")
for i in tqdm(range(len(pt[0]))):
    p[0][i] += pt[0][i]
    p[1][i] += pt[1][i]
    p[2][i] += pt[2][i]

#timesplit
    #earth
print("echantillonage de point de l'orbite de la terre")
tspt = [[],[],[]]
for i in tqdm(range(0,len(pt[0])+1,ts)):
    tspt[0].append(pt[0][i])
    tspt[1].append(pt[1][i])
    tspt[2].append(pt[2][i])
    #sat

print("echantillonage de point de l'orbite du satellite")
tsp = [[],[],[]]
for i in tqdm(range(0,len(pt[0])+1,ts)):
    tsp[0].append(p[0][i])
    tsp[1].append(p[1][i])
    tsp[2].append(p[2][i])

#plot :
fig1 = plt.figure(1)
ax = plt.axes(projection='3d')

ax.scatter3D(tspt[0],tspt[1],tspt[2],color = 'green', s = 10)
ax.scatter3D(tsp[0],tsp[1],tsp[2],color = 'hotpink',s = 1)
ax.scatter3D(0,0,0,color = "red", s =500)

#zoom x1

fig2 = plt.figure(2)
ax = plt.axes(projection='3d')
ax.scatter3D(tspt[0][:750],tspt[1][:750],tspt[2][:750],color = 'green', s = 10)
ax.scatter3D(tsp[0][:750],tsp[1][:750],tsp[2][:750],color = 'hotpink',s = 1)

#zoom 2

fig3 = plt.figure(3)
ax = plt.axes(projection='3d')
ax.scatter3D(tspt[0][:50],tspt[1][:50],tspt[2][:50],color = 'green', s = 10)
ax.scatter3D(tsp[0][:50],tsp[1][:50],tsp[2][:50],color = 'hotpink',s = 1)

```

CODE COMPLET

```
#ombre et puissance

E = 0
o = 0
pen = 0
l = 0

for i in tqdm(range(len(tspt[0]))):

    #Declaration des variables
    p0xt = tspt[0][i]
    p0yt = tspt[1][i]
    p0zt = tspt[2][i]
    T = np.array([p0xt,p0yt,p0zt])

    p0x = tsp[0][i]
    p0y = tsp[1][i]
    p0z = tsp[2][i]
    PS = np.array([p0x,p0y,p0z])
    PT = PS - T

    DTSat = m.sqrt((p0x-p0xt)**2+(p0y-p0yt)**2+(p0z-p0zt)**2)
    DSS = m.sqrt((p0x)**2+(p0y)**2+(p0z)**2)
    DTS = m.sqrt((p0xt)**2+(p0yt)**2+(p0zt)**2)

    Us = T/np.linalg.norm(T)
    projv = np.dot(Us, PT) * Us
    proj = np.dot(Us, PT)
    rej = PT - projv
    nrej = np.linalg.norm(rej)
```

```
# 1er cas le satellite est entre la terre et le soleil : lumiere directe
if proj <= 0 :
    Pss = pst/(4 * 3.14 * (DSS**2) )
    E += Pss * Dt #energie/ m2
    l += 1

else :
    # 2eme cas le satellite est entre la terre et la lune plusieurs cas ce presentes
    xu = (rt*DTS) / (rs - rt)
    au = m.asin(rt/xu)
    ki = (xu - proj)*m.tan(au)
    if nrej <= ki : # satellite dans l'ombre
        E += 0
        o += 1
    else :
        xp = rt*DTS/(rs + rt)
        ap = m.asin((rt)/(xp))
        kp = (xp + proj)* m.tan(ap)
        angle = m.atan2(nrej, proj)
        if nrej <= kp : #satellite dans la penombre
            Pss = pst/(4 * 3.14 * (DSS**2) )
            E += Pss * Dt / 2
            pen += 1
        else : # satellite dans la lumiere du soleil
            Pss = pst/(4 * 3.14 * (DSS**2) )
            E += Pss * Dt #energie/ m2
            l += 1
```

```
print(f"lumiere% = {l/len(tspt[0])*100}")
print(f"ombre% = {o/len(tspt[0])*100}")
print(f"penombra% = {pen/len(tspt[0])*100}")
print(f"OMBRE% = {(pen+o) / (len(tspt[0])) * 100}%")
print(f"Energie surfacique = {E}J/m2")
print(f"Puissance surfacique moyenne = {E/(len(tspt[0]*Dt))}W/m2")
print(f"Puissance% = {(E/(len(tspt[0]*Dt)))/1366.64 * 100}%")
```