Bouclier solaire

TIPE 2023

LECLAND Erwan candidat n°16870

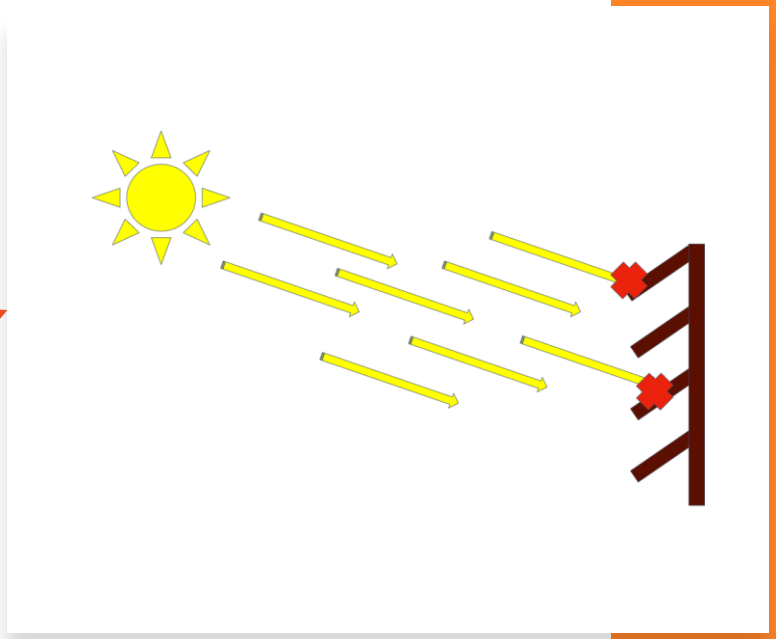
Sommaire :

I.Introduction

- Inauguré le 2 avril 2009 à Dijon
- Premier bâtiment « à énergie positive »
- Echauffement limité en été grâce à un bouclier solaire

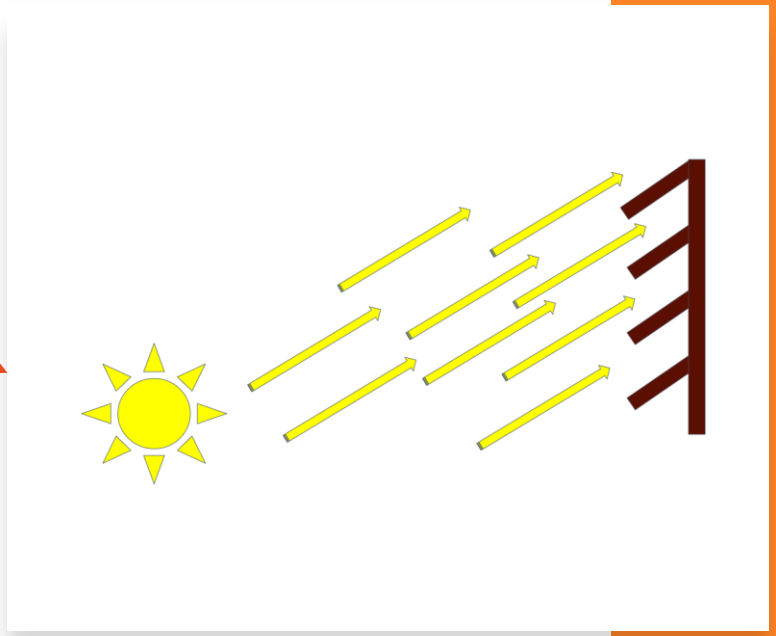
I.Introduction

Fonctionnement en été



Le bouclier bloque les rayons provenant du Soleil s'il est trop haut

Fonctionnement en hiver



I.Introduction

Problématique :

Au regard de l'importance de la gestion énergétique dans un bâtiment en ville, comment limiter les apports solaires thermiques dans une tour de bureaux ?

Objectifs :

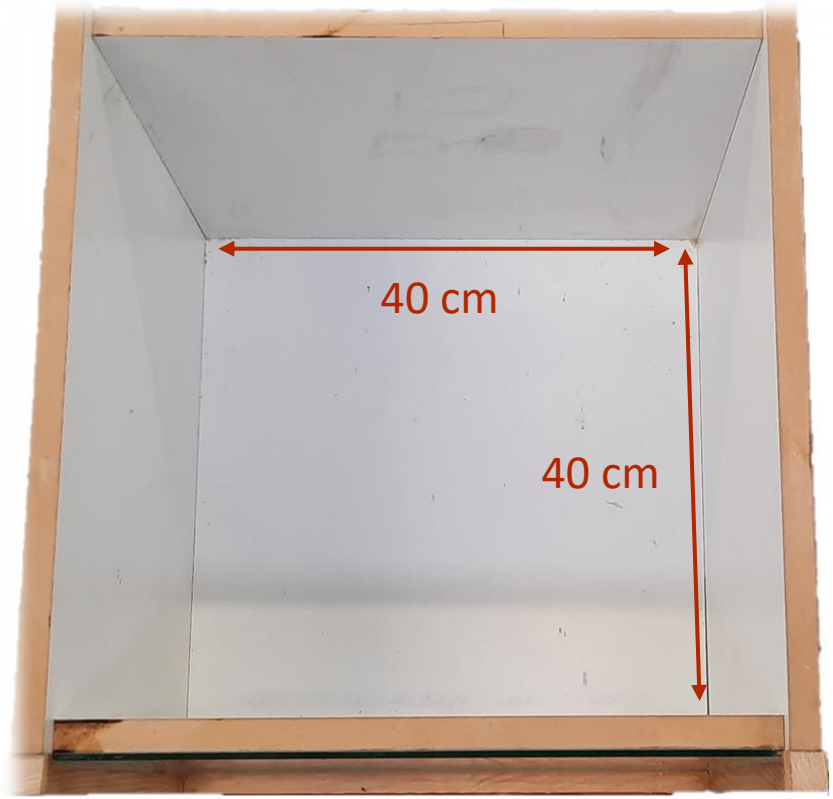
-Conception et création d'une maquette de bouclier solaire

-Modélisation de la course du soleil

-Mesure sur la maquette

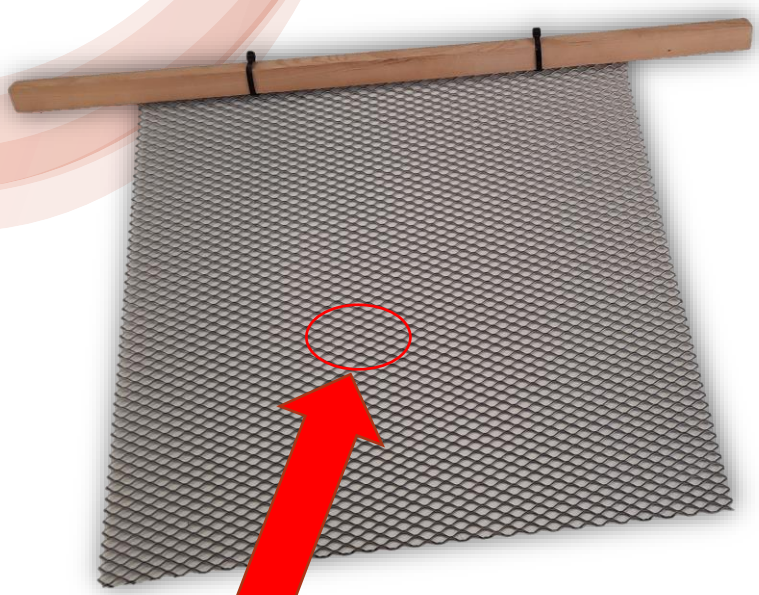
II. Conception et création d'une maquette de bouclier solaire

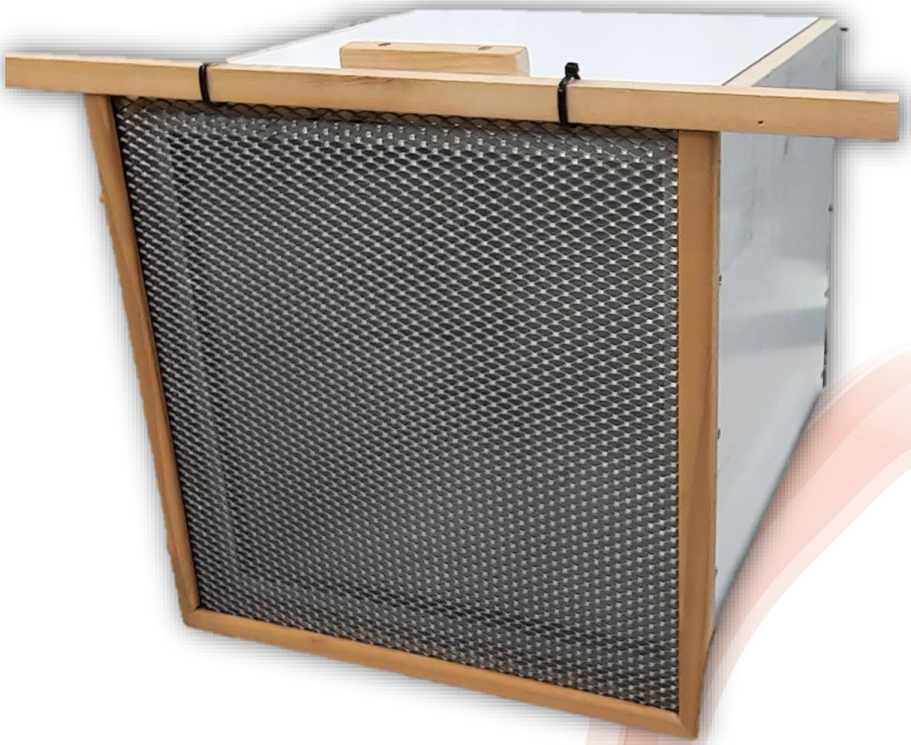
Bois de récupération avec revêtement en blanc

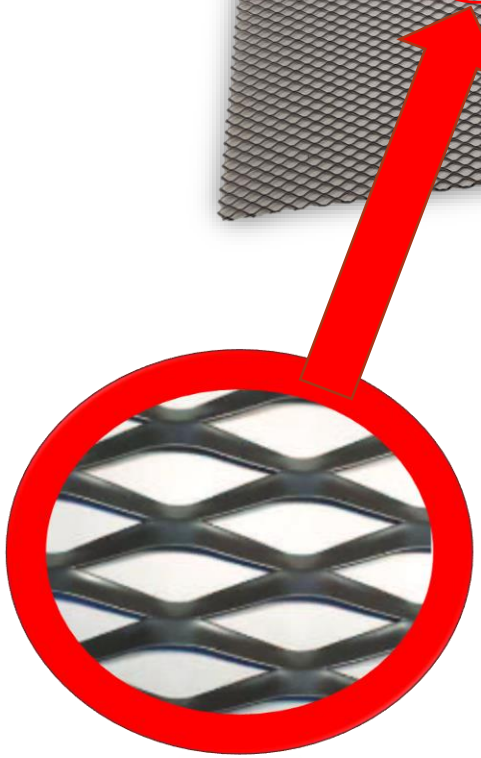


Simple vitrage

II. Conception et création d’une maquette de bouclier solaire



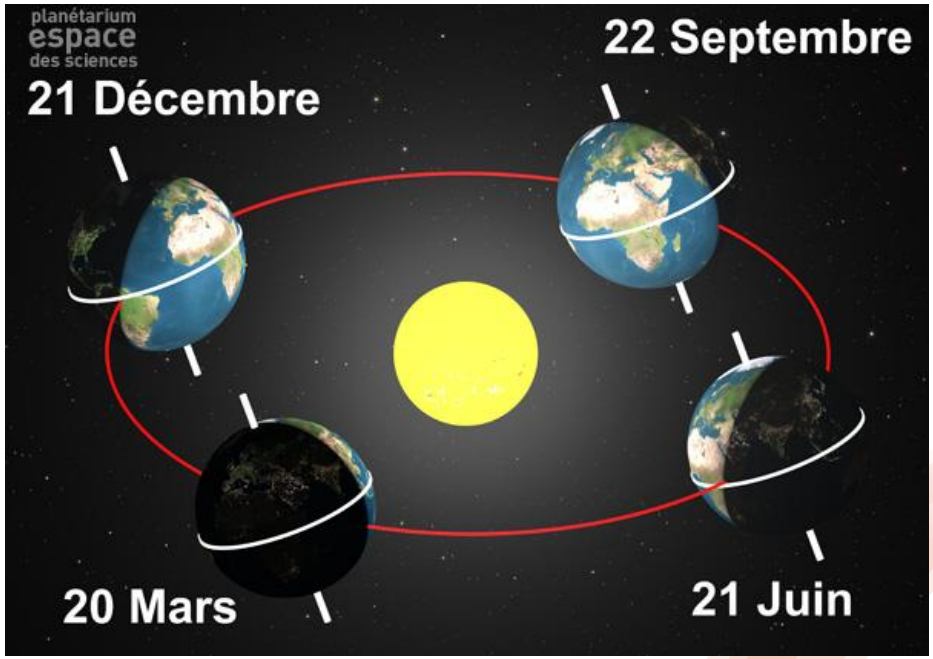




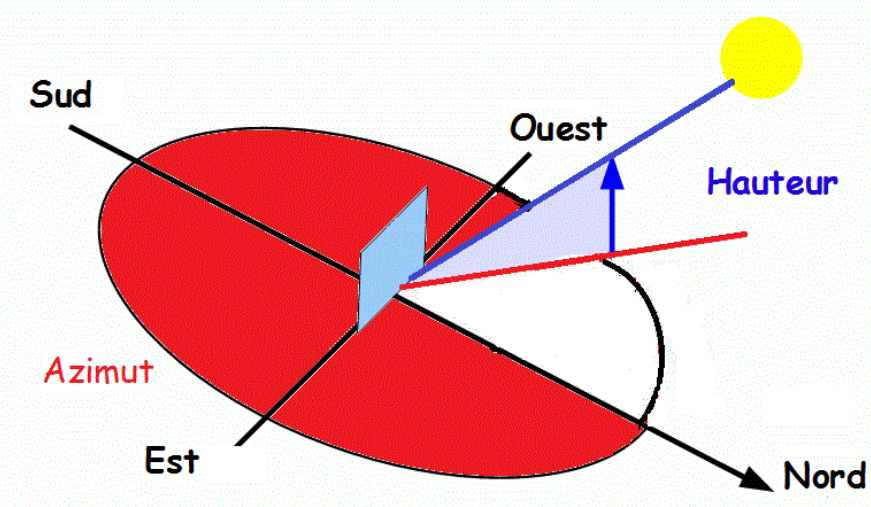
Modèle final proposé

III. Modélisation de la course du soleil

21 juin, le Soleil est au plus haut à midi

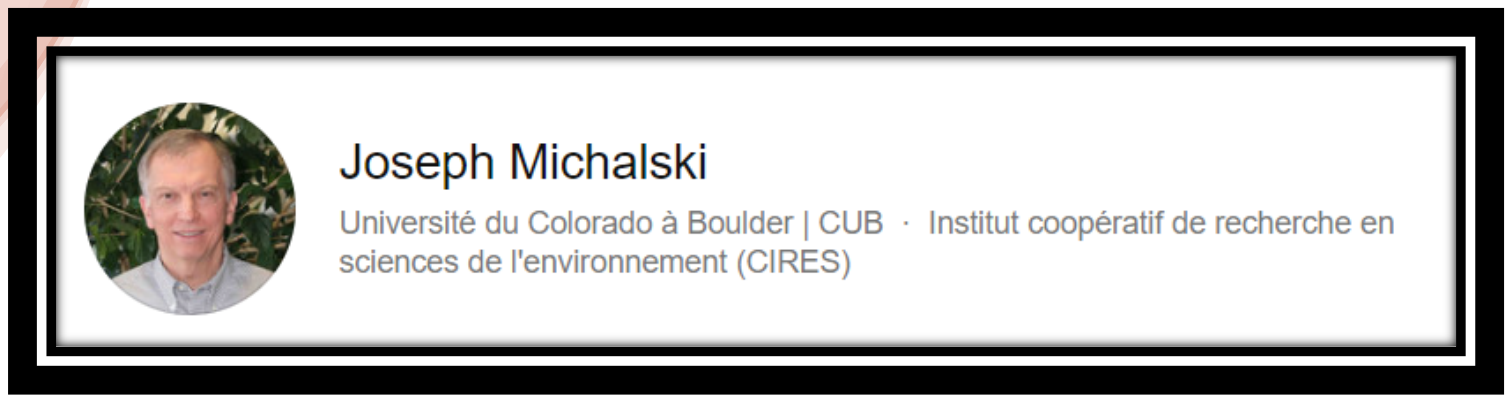


https://www.espace-sciences.org/planetarium/article/le-solstice-d-ete#:~:text=Cette%20variation%20de%20hauteur%20du,au%20jour%20le%20plus%20long.



On inverse le modèle traditionnel pour avoir des valeurs positives d'Azimut

III. Modélisation de la course du soleil



https://www.researchgate.net/profile/Joseph-Michalsky

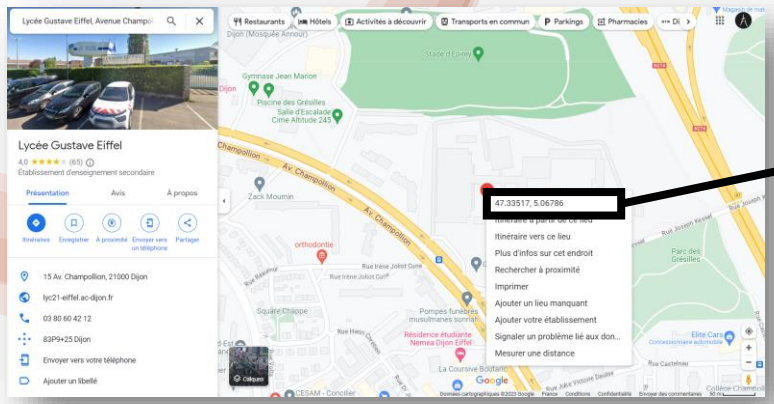
<u>The astronomical almanac algorithm for approximate solar position (1950-2050)</u> :

-Utilise des équations pour approximer la hauteur du Soleil

-Utilise des équations pour approximer l'azimut du Soleil

-Précision annoncée de 0,01°

III. Modélisation de la course du soleil

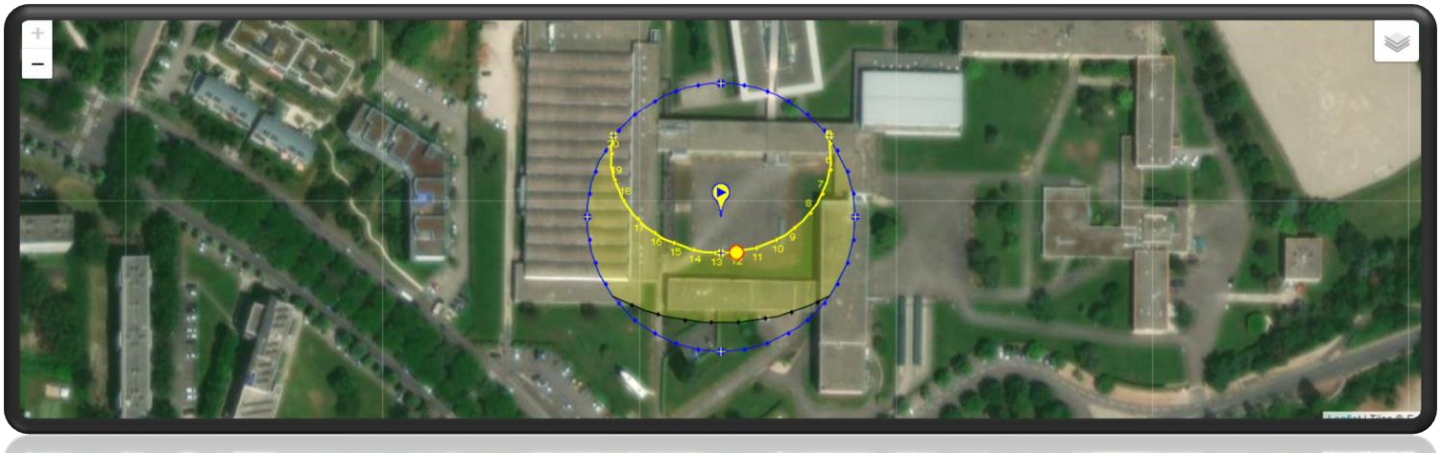


47.33514, 5.06795

Transfert des coordonnés

https://www.google.fr/maps/place/Lyc%C3%A9e+Gustave+Eiffel/@47.3350201,5.0654054,17z/data=!3m1!4b1!4m6!3m5!1s0x47f29e0319fec997:0x198ca025835186da!8m2!3d47.3350201!4d5.0679803!16s%2Fg%2F11b6h_cpm1?entry=ttu

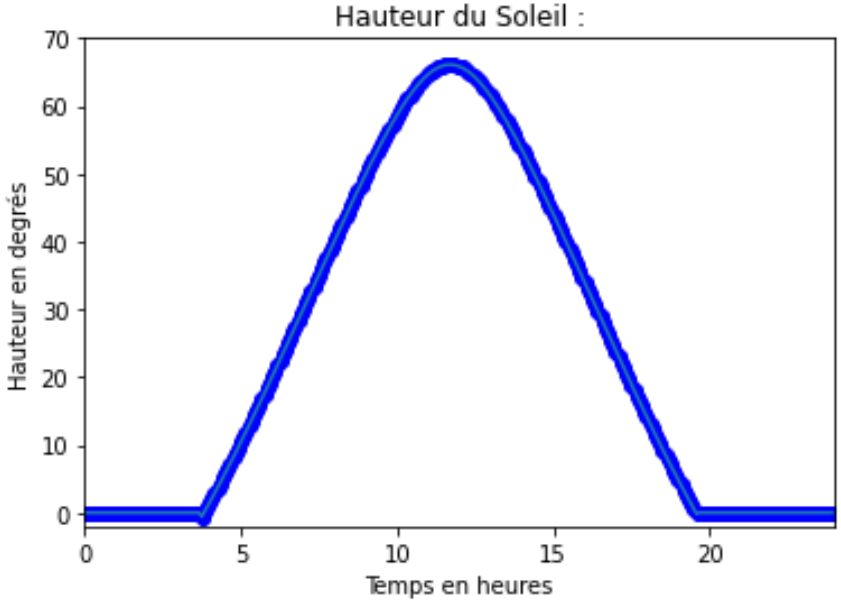
https://www.sunearthtools.com/dp/tools/pos_sun.php?lang=fr#top

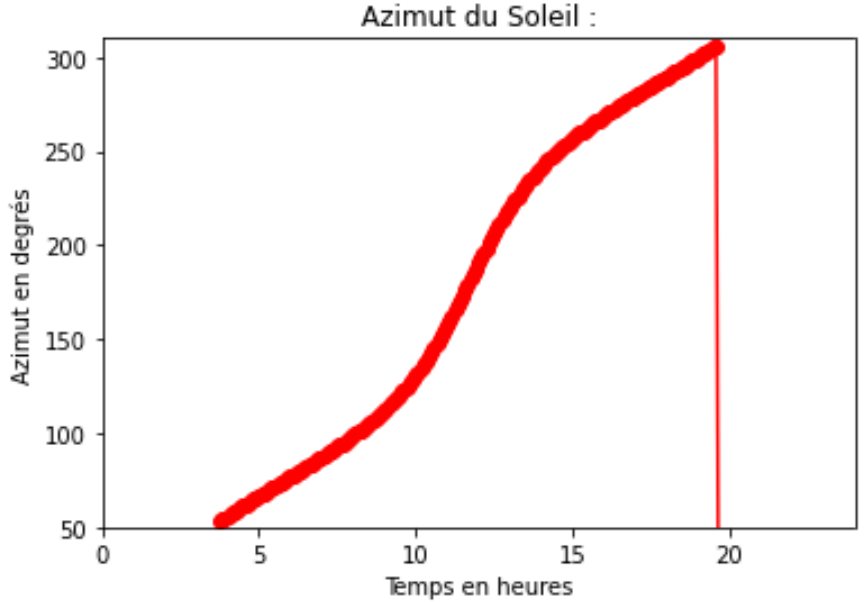


Utilisation des équations

III. Modélisation de la course du soleil

```
Quel jour souhaitez vous ? : 21/06
Indiqué l'heure à laquel vous souhaité l'Azimut et la Hauteur : 12:00
L'Azimut est  190.37
La hauteur est  65.82
```





III. Modélisation de la course du soleil

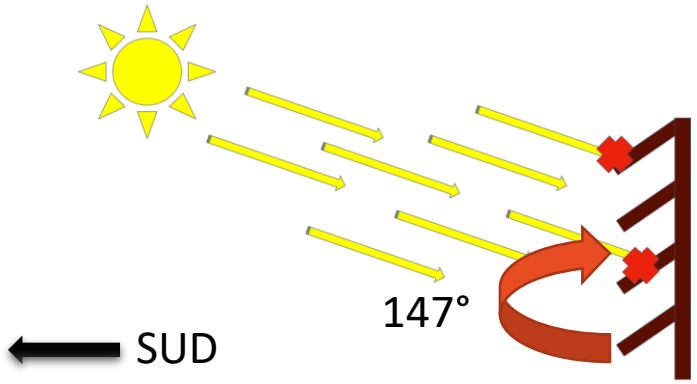
```
La hauteur max que le soleil a atteind en 2022 est 66.1
La hauteur min que le soleil a atteind en 2022 est 0
L'azimut max que le soleil a atteind en 2022 est 307.02
L'azimut min que le soleil a atteind en 2022 est 53.13
```

Recherche des valeurs maximums(et minimums)

Les approximations sont imparfaites :
A midi le 21/06, la hauteur était de 65,82, hors la hauteur max atteinte est de 66,1.

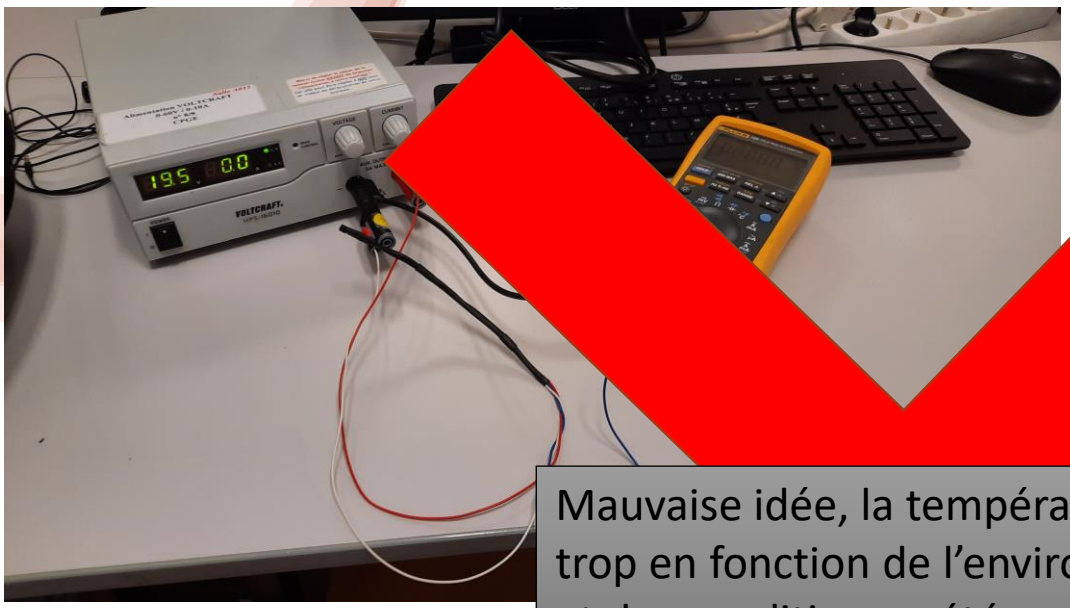
```
L'angle optimal est de 57.2923404255319 degré
```

Etude sur l'angle optimal (on bloque complètement les rayons au-dessus de 57°)



On en déduit que le bouclier doit avoir des pales orientées à 147° vers le Sud.

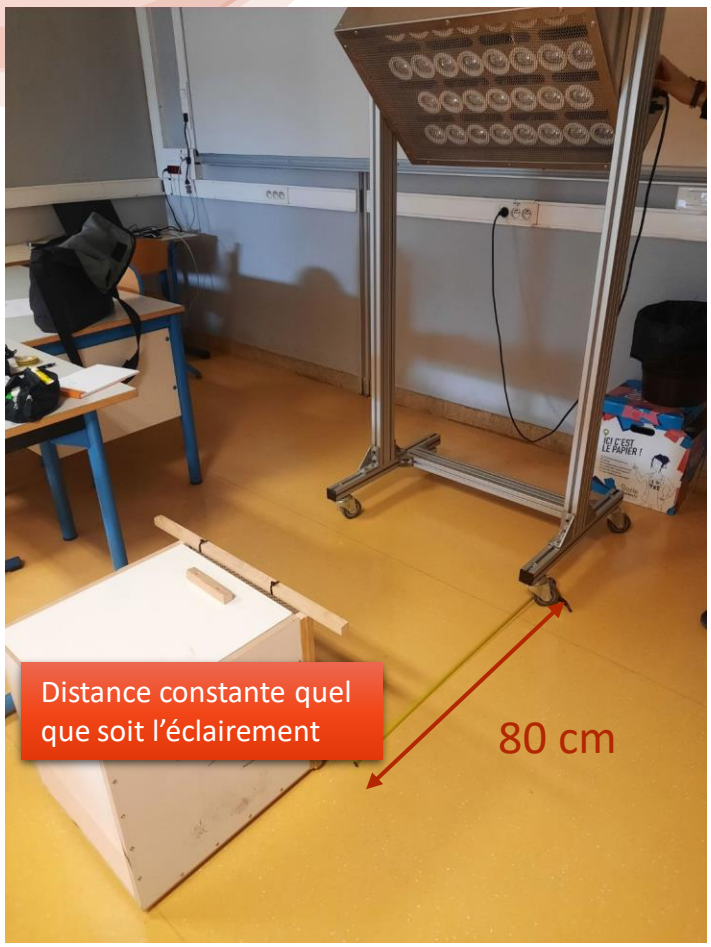
IV. Mesure sur la maquette



Mauvaise idée, la température varie trop en fonction de l'environnement et des conditions météo.

IV. Mesure sur la maquette

Disposition de la mesure :

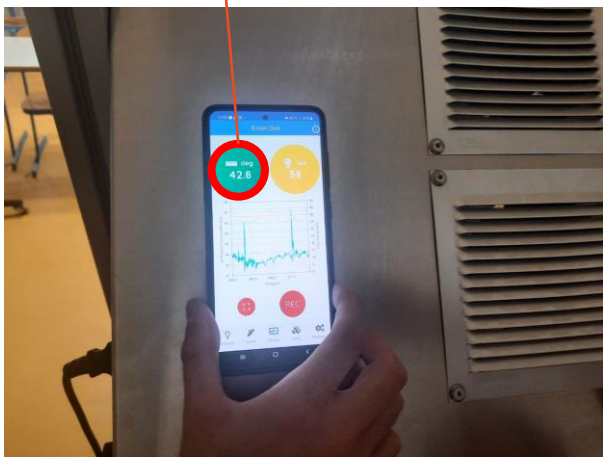


Simule le Soleil :

Les rayons se propagent en ligne droite

Envoie un éclairement choisi sur la vitre de la boite

Orientation à 43° et pas de possibilité de pivoter la lampe



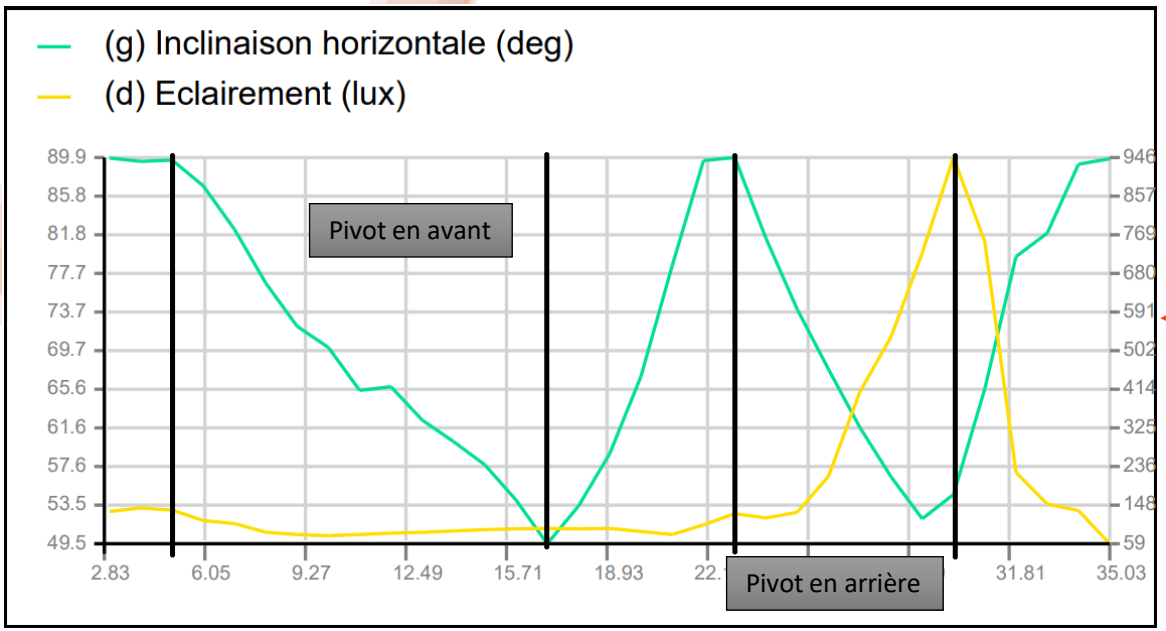
IV. Mesure sur la maquette

Prise de mesure :
On fait tanguer la boite en avant et en arrière

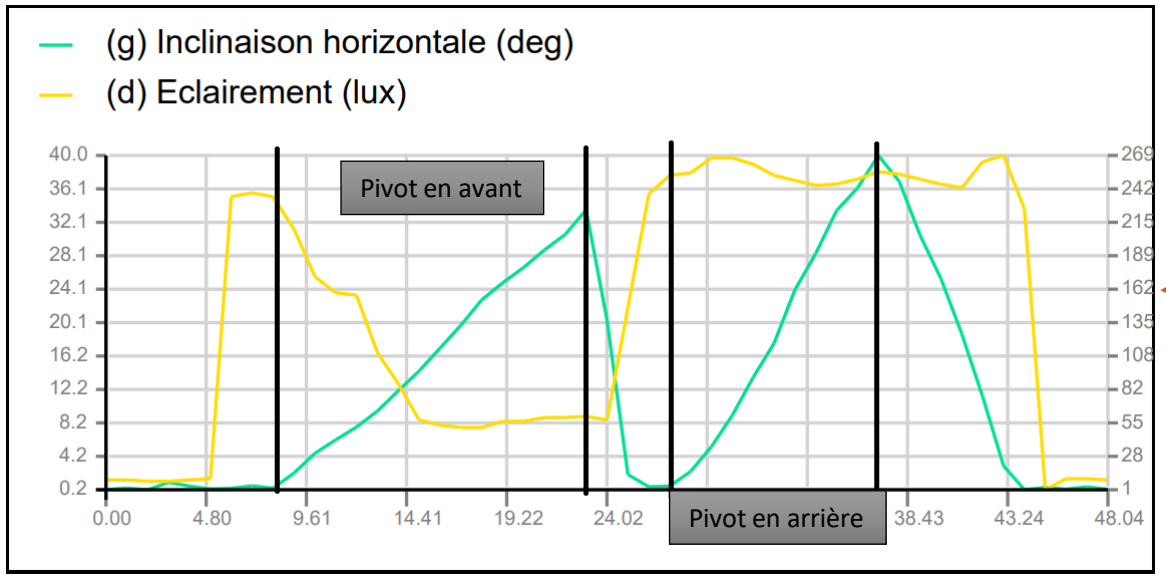
Placement :

Etalonnage :

IV. Mesure sur la maquette



Fond de la boite
(planche parallèle à la vitre)

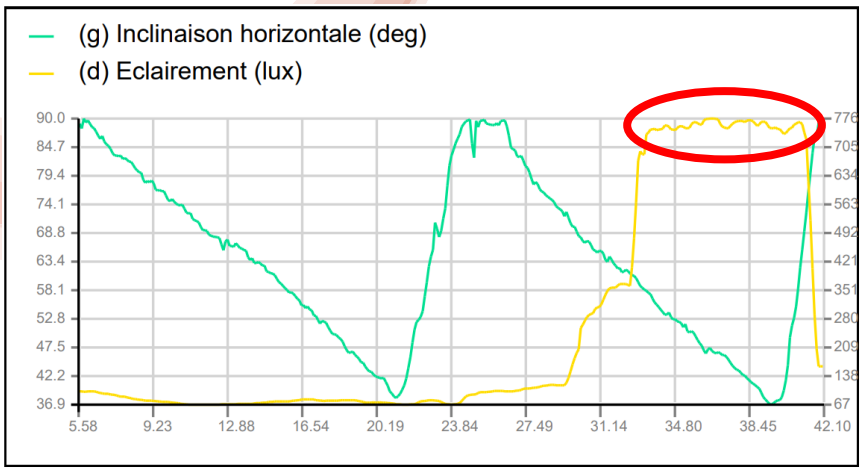


Sol de la boite

Essaie à 1200 lux

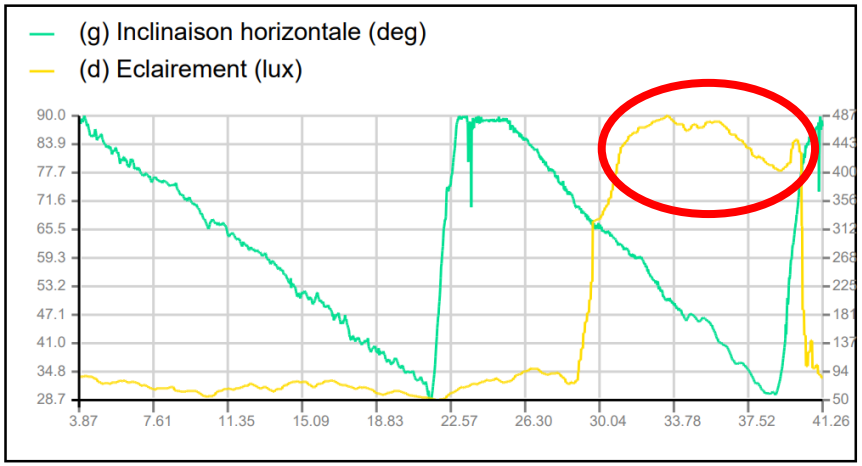
IV. Mesure sur la maquette

Sur le fond de la boite
(planche parallèle à la vitre)



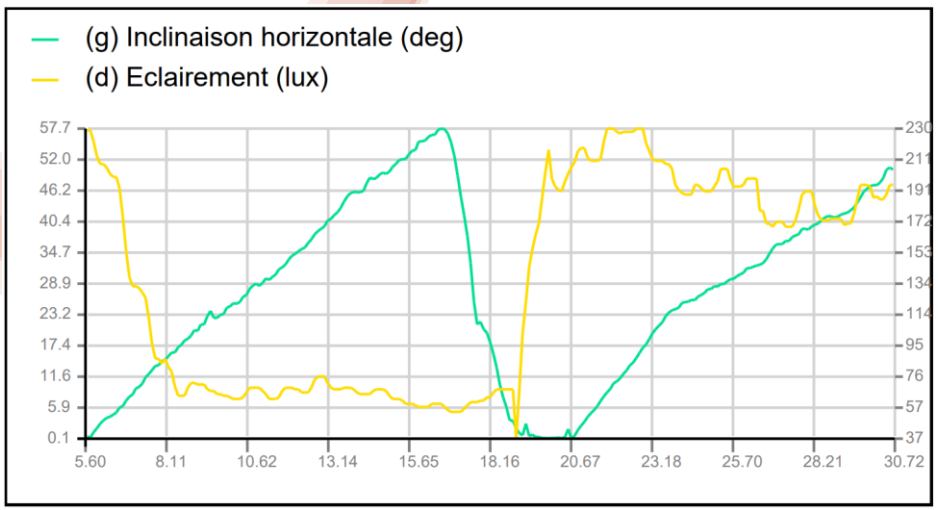
Essaie a 1000 lux

Après de nombreux essais, on se rend compte que le bouclier protège au minima environ 20% des rayons du Soleil entrant .



Essaie a 600 lux

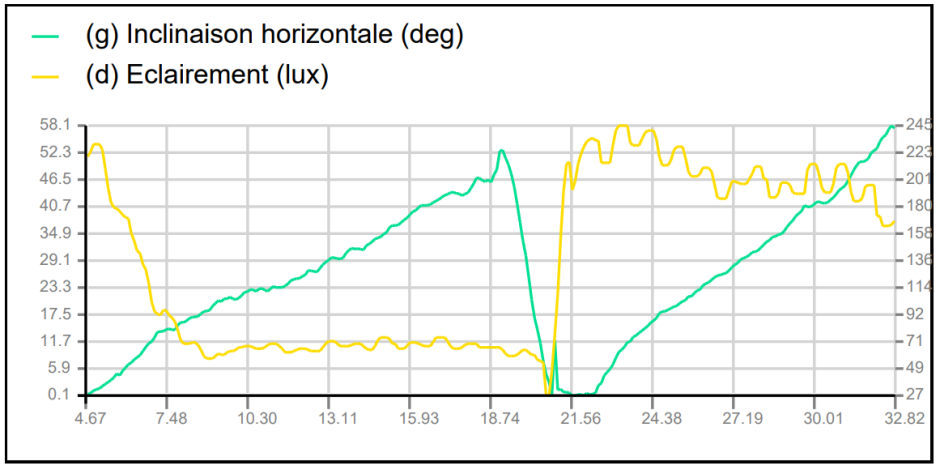
IV. Mesure sur la maquette



Sur le sol de la boite

Essaie a 1000 lux

Après de nombreux essais, on se rend compte que le bouclier protège de façon assez équivalente le sol de la boite quel que soit l'éclairage entrant.



Essaie a 600 lux

V. Conclusion et piste de poursuite

Conclusion :

- Le bouclier doit être placé sur la façade Sud du bâtiment
- Pour éviter les fortes chaleurs en été il doit être orienter de 147° (face au Sud)
- Si on utilise un espacement et une épaisseur identique des mailles du bouclier, toutes l'année les rayons ne rentreront qu'à 80%.

V. Conclusion et piste de poursuite

Piste de poursuite :

- Placer des panneaux sur la façade Est et Ouest incliné à l'Horizontal pour stopper les rayons grâces aux informations sur l'Azimut
- Pousser la réflexion sur un éclairage adaptatif pour la lumière à l'intérieur du bâtiment en fonction de la lumière extérieur
- Voir si d'autre disposition des mailles du métal déployer pourrait atténuer encore plus les rayons entrants

VI. Annexe

```
import matplotlib.pyplot as plt#importe un module de trassage de courbe
import matplotlib.image as mpimg#importe un module de gestion d'image
img = mpimg.imread("azimut.gif")#ouvre l'image
plt.imshow(img)#place l'image
plt.show()#montre l'image

import csv #Importe le module csv

lecteur = csv.DictReader(open('jour''.csv','r')) #Définit quel ficher lire
for ligne in lecteur: #Boucle for pour balayer le tableur
    jour = dict(ligne) #création du dictionnaire jour

n = str(input("Quel jour souhaitez vous ? : ")) #Question posé à l'utilisateur

q=str(input ("Indiqué l'heure à laquel vous souhaité l'Azimut et la Hauteur : "))#Question posé à l'utilisateur

lecteur = csv.DictReader(open('cs2021A''.csv','r')) #Définit quel ficher lire
dictionaire = [dict(ligne) for ligne in lecteur] #Boucle for pour balayer le tableur et créer une liste de dictionaires

j=dictionaire[int(jour[n])] #Dictionaire tiré de la liste, du jour qui nous interesse seulement

print("L'Azimut est ",j.get(q))#Affiche l'Azimut

lecteur = csv.DictReader(open('cs2021E''.csv','r')) #Définit quel ficher lire
dictionaire = [dict(ligne) for ligne in lecteur] #Boucle for pour balayer le tableur et créer une liste de dictionaires

j=dictionaire[int(jour[n])] #Dictionaire tiré de la liste, du jour qui nous interesse seulement

print("La hauteur est ",j.get(q))#Affiche la hauteur

x=[]#créé une liste x qui vas contenir les valeurs de la hauteur et de l'azimut de la journée selectionné

for val in j.values():#balayage par le nombre de valeur de j
    if val == "--":# si la valeur est un --
        val=0#-- devient 0
        x.append(val)#ajoute un 0 a x
    else:#sinon
        x.append(val)#on ajoute la valeur normalement a la liste
x.remove(x[0])#on retire la première valeur qui est la date et qui ne nous interessera pas dans cette liste
```

VI. Annexe

```
40
41  h=[]#créé une liste h qui vas contenir les valeurs de la hauteur de la journée selectionné
42  a=[]#créé une liste a qui vas contenir les valeurs de l'azimut de la journée selectionné
43
44  for k in range(len(x)):#l'action vas être répété le nombre de fois qu'il y a de valeur dans x
45      if (k%2)==0:#si la position de l'élement de la liste est pair
46          h.append(float(x[k]))#on l'ajoute à la liste h la valeur pair de  x
47      else:#sion(elle est forcement impair)
48          a.append(float(x[k]))#on l'ajoute à la liste a la valeur impair de x
49
50  heure=[]#créé une liste heure qui vas contenir les valeurs l'heure de la journée selectionné
51
52  for k in range(len(h)):#va dénombrer le nombre d'éléménts que vas contenir les liste h(et a car si le tableur est corectement
53      heure.append(k*1/12)#convertion des minutes en heures(il y a 12 valeur par heure car 1 toutes les 5 minutes)
54
55  plt.title('Hauteur du Soleil :')#titre du graphique
56  plt.ylabel('Hauteur en degrés')#titre de l'axe verticale
57  plt.xlabel('Temps en heures')#titre de l'axe horizontal
58  plt.axis([0,24, -2, 70])#limite des valeurs(entre 0 et 24 heures et entre -2(-2car les valeurs peuvent parfois être negatives
59  plt.plot(heure,h,'bo')#place les point de la hauteur en fonction de l'heure
60  plt.plot(heure,h)#trace la courbe de la hauteur en fonction de l'heure
61  plt.show()#montre le résutat
62
63  plt.title('Azimut du Soleil :')#titre du graphique
64  plt.ylabel('Azimut en degrés')#titre de l'axe verticale
65  plt.xlabel('Temps en heures')#titre de l'axe horizontal
66  plt.axis([0,24, 50, 310])#limite des valeurs(entre 0 et 24 heures et entre 50 et 70 d'azimut)
67  plt.plot(heure,a,'ro')#place les point de l'azimut en fonction de l'heure
68  plt.plot(heure,a,'r')#trace la courbe de l'azimut en fonction de l'heure
69  plt.show()#montre le résutat
```

Ligne 52 : #va dénombrer le nombre d'éléments que vas contenir les liste h(et a car si le tableur est correctement dimensionné, les listes sont de même dimension)

Ligne 58 : #limite des valeurs(entre 0 et 24 heures et entre -2(-2car les valeurs peuvent parfois être négatives dans le tableur) et 70 de hauteur)

VI. Annexe

```python
import csv #Importe le module csv

hmax=[]#créé une liste hmax qui vas contenir les valeurs de la hauteur max
amax=[]#créé une liste amax qui vas contenir les valeurs de l'azimut max
amin=[]#créé une liste amin qui vas contenir les valeurs de l'azimut min

lecteur = csv.DictReader(open('jour''.csv','r')) #Définit quel ficher lire
for ligne in lecteur: #Boucle for pour balayer le tableur
    jour = dict(ligne) #création du dictionnaire jour

lecteur = csv.DictReader(open('cs2021A''.csv','r')) #Définit quel ficher lire
dictionaire = [dict(ligne) for ligne in lecteur] #Boucle for pour balayer le tableur et créer une liste de dictionaires

lecteur = csv.DictReader(open('cs2021E''.csv','r')) #Définit quel ficher lire
dictionaire = [dict(ligne) for ligne in lecteur] #Boucle for pour balayer le tableur et créer une liste de dictionaires


for n in range(365):#répète l'action les 365 jours de l'année
    j=dictionaire[n] #Dictionaire tiré de la liste, du jour qui nous interesse seulement
    x=[]#créé une liste x qui vas contenir les valeurs de la hauteur et de l'azimut de la journée selectionné
    for val in j.values():#balayage par le nombre de valeur de j
        if val == "--":# si la valeur est un --
            val=0#-- devient 0
            x.append(val)#ajoute un 0 a x
        else:#sinon
            x.append(val)#on ajoute la valeur normalement a la liste
    x.remove(x[0])#on retire la première valeur qui est la date et qui ne nous interessera pas dans cette liste

    h=[]#créé une liste h qui vas contenir les valeurs de la hauteur de la journée selectionné
    a=[]#créé une liste a qui vas contenir les valeurs de l'azimut de la journée selectionné

    for k in range(len(x)):#l'action vas être répété le nombre de fois qu'il y a de valeur dans x
        if (k%2)==0:#si la position de l'élement de la liste est pair
            h.append(float(x[k]))#on l'ajoute à la liste h la valeur pair de  x
        else:#sion(elle est forcement impair)
            a.append(float(x[k]))#on l'ajoute à la liste a la valeur impair de x


    azi=sorted(a)#créé la liste azi composer des valeur trié de la liste a de la plus petite à la plus grande valeur
    amax.append(azi[287])#prend la plus grande valeur de azi et l'ajoute à amax
```

VI. Annexe

```
    hau=sorted(h)#créé la liste hau composer des valeur trié de la liste h de la plus petite à la plus grande valeur
    hmax.append(hau[287])#prend la plus grande valeur de hau et l'ajoute à hmax

    azimin=[]#créé une liste amin qui vas contenir les valeurs de l'azimut min
    for az in range(288): #répète l'action les 288 valeurs de la journé
        p=azi[az] #tire la valeur p, soit la valeur numéro az de la liste azi
        if p==0: #si p est égal à 0
            p=0 #p est égal à 0
        else: #sinon
            amin.append(p) #on ajoute la valeur à la liste amin


hmax=sorted(hmax) #trie de la liste hmax de la plus petite à la plus grande valeur
amax=sorted(amax) #trie de la liste amax de la plus petite à la plus grande valeur
amin=sorted(amin) #trie de la liste amin de la plus petite à la plus grande valeur
print("La hauteur max que le soleil a atteind en 2022 est",hmax[364]) #Affiche le résultat de la hauteur max
print("La hauteur min que le soleil a atteind en 2022 est 0") #Affiche le résultat de la hauteur min
print("L'azimut max que le soleil a atteind en 2022 est",amax[364]) #Affiche le résultat de l'azimut max
print("L'azimut min que le soleil a atteind en 2022 est",amin[0]) #Affiche le résultat de l'azimut min
```

```
La hauteur max que le soleil a atteind en 2022 est 66.1
La hauteur min que le soleil a atteind en 2022 est 0
L'azimut max que le soleil a atteind en 2022 est 307.02
L'azimut min que le soleil a atteind en 2022 est 53.13
```

VI. Annexe

```
#La hauteur maximum du Soleil est le 21 Juin, soit le 1er jour de l'été
#La fin de l'été est le 23 Septembre
#Il suffit de faire une moyenne de la hauteur du Soleil du 21 Juin au 23
#Les jours concerné sont donc le 172ème et le 266ème de l'année

import csv #Importe le module csv

hmax=[]#créé une liste hmax qui vas contenir les valeurs de la hauteur max

lecteur = csv.DictReader(open('jour''.csv','r')) #Définit quel ficher lire
for ligne in lecteur: #Boucle for pour balayer le tableur
    jour = dict(ligne) #création du dictionnaire jour

lecteur = csv.DictReader(open('cs2021A''.csv','r')) #Définit quel ficher lire
dictionaire = [dict(ligne) for ligne in lecteur] #Boucle for pour balayer le tableur et créer une liste de dictionaires

lecteur = csv.DictReader(open('cs2021E''.csv','r')) #Définit quel ficher lire
dictionaire = [dict(ligne) for ligne in lecteur] #Boucle for pour balayer le tableur et créer une liste de dictionaires

for n in range(172,266):#répète l'action les jours qui nous interessent seulement
    j=dictionaire[n] #Dictionaire tiré de la liste, du jour qui nous interesse seulement
    x=[]#créé une liste x qui vas contenir les valeurs de la hauteur et de l'azimut de la journée selectionné
    for val in j.values():#balayage par le nombre de valeur de j
        if val == "--":# si la valeur est un --
            val=0#-- devient 0
            x.append(val)#ajoute un 0 a x
        else:#sinon
            x.append(val)#on ajoute la valeur normalement a la liste
    x.remove(x[0])#on retire la première valeur qui est la date et qui ne nous interessera pas dans cette liste

    h=[]#créé une liste h qui vas contenir les valeurs de la hauteur de la journée selectionné

    for k in range(len(x)):#l'action vas être répété le nombre de fois qu'il y a de valeur dans x
        if (k%2)==0:#si la position de l'élement de la liste est pair
            h.append(float(x[k]))#on l'ajoute à la liste h la valeur pair de  x


    hau=sorted(h)#créé la liste hau composer des valeur trié de la liste h de la plus petite à la plus grande valeur
    hmax.append(hau[287])#prend la plus grande valeur de hau et l'ajoute à hmax

```

VI. Annexe

```
40
41  somme=0 #initialisation de la somme des valeurs (pour la moyenne)
42
43  for o in range(len(hmax)): #répète 94 fois [nombre de jour]
44      somme = somme+ hmax[o] #ajoute la valeur précedante avec l'actuel
45
46  Résultat= somme/94 #fait la moyenne
47
48  print("L'angle optimal est de",Résultat,"degré") #affiche le résultat
```

```
L'angle optimal est de 57.2923404255319 degré
```

VI. Annexe

- The astronomical almanac algorithm for approximate solar position (1950-2050) :

1. Calcul de la durée écoulée depuis le 1er janvier 2000 (en jours) :

   $\Delta T = (JD - 2451545.0) / 36525.0$

2. Calcul de la position moyenne du soleil :

   -L'obliquité de l'écliptique (ε) est calculée en utilisant une formule polynomiale en fonction de ΔT.

   -L'anomalie moyenne du soleil (M) est calculée en utilisant une formule polynomiale en fonction de ΔT.

   -La longitude moyenne du soleil (L) est calculée en utilisant une formule polynomiale en fonction de ΔT.

   -La longitude apparente du soleil (λ) est obtenue en ajoutant une correction à la longitude moyenne du soleil.

3. Calcul de l'équation du temps :

   -L'équation du temps (E) est calculée en utilisant une formule polynomiale en fonction de ΔT et de la longitude.

4. Calcul de l'azimut et de l'élévation du soleil :

   -L'heure locale est ajustée en fonction de la longitude, de l'équation du temps et du décalage horaire.

   -L'heure solaire moyenne (MST) est calculée en utilisant l'heure locale, la longitude et l'équation du temps.

   -L'heure solaire apparente (AST) est obtenue en ajoutant une correction à l'heure solaire moyenne.

   -L'heure solaire apparente est ensuite utilisée pour calculer l'azimut et l'élévation du soleil en fonction de la latitude, de la longitude et de l'heure solaire apparente.

V. Annexe

Calcul de l'obliquité de l'écliptique (ε) : ε = 23.439291 - 0.0130042 * ΔT - 1.64e-07 * ΔT^2 + 5.04e-07 * ΔT^3

Calcul de l'anomalie moyenne du soleil (M) : M = 357.52911 + 35999.05029 * ΔT - 1.537e-04 * ΔT^2 + 2.56e-07 * ΔT^3

Calcul de la longitude moyenne du soleil (L) : L = 280.46646 + 36000.76983 * ΔT + 3.032e-04 * ΔT^2

Calcul de la longitude apparente du soleil (λ) : λ = L + 1.915 * sin(M) + 0.020 * sin(2M)

Calcul de l'équation du temps (E) : E = (L - 0.0057183 - λ) * 4

Calcul de l'heure locale ajustée : Heure_locale_ajustée = Heure_locale + Longitude / 15 - E / 60 + Décalage_horaire

Calcul de l'heure solaire moyenne (MST) : MST = Heure_locale_ajustée * 15 - Longitude

Calcul de l'heure solaire apparente (AST) : AST = MST + E

Calcul de l'azimut (Az) et de l'élévation (El) du soleil : sin(El) = sin(Latitude) * sin(δ) + cos(Latitude) * cos(δ) * cos(HA) cos(Az) = (sin(El) * sin(Latitude) - sin(δ)) / (cos(El) * cos(Latitude))

où:

ΔT = (JD - 2451545.0) / 36525.0 (JD= Julian date*)

δ : déclinaison du soleil

HA : angle horaire du soleil

*La Julian Date est basée sur le nombre de jours écoulés depuis midi (heure universelle) le 1er janvier de l'an 4713 avant Jésus-Christ (date julienne). Elle est souvent utilisée pour les calculs astronomiques en raison de sa simplicité et de sa précision.

VI. Annexe

```
c   this program calculates the solar position given the year,
day,
c   time, latitude, and longitude
c
c   it outputs solar azimuth, elevation, hour angle,
declination, and
c   air mass
c
c
        program sunpos
        real lat,long, mn
        write(6,'(3x,"This program calculates sun position if you
     # specify time and location!")')
        write(6,'(/)')
        write(6,'(3x,"Type the latitude like this sxx.xx where s
is a
     # sign!")')
        read(5,'(f8.4)')lat
        write(6,'(f8.4)')lat
        write(6,'(3x,"Type the longitude like this sxxx.xx!")')
        read(5,'(f9.4)')long
        write(6,'(f9.4)')long
        write(6,'(3x,"Type the year like this xxxx.!")')
        read(5,'(f5.0)')year
        write(6,'(f5.0)')year
        write(6,'(3x,"Indicate time zone,e.g., est=+5,mst=+7,etc.
(The
     # sign is important)!")')
        read(5,'(i3)')itz
        write(6,'(i3)')itz
        tz=float(itz)
60      write(6,'(3x,"Type the day of year(Feb 1=32) like this
xxx.!")')
        read(5,'(f4.0)')day
        write(6,'(f4.0)')day
50      write(6,'(3x,"Type the local standard time(not daylight
savings
     # time) like this xx.xx.xx., e.g., 12.15.47.!")')
        read(5,'(3f3.0)')hour, mn,sec
        write(6,'(3f3.0)')hour, mn,sec
        tm=((tz+hour)*3600+ mn*60+sec)/3600.
        call sunae(year,day,tm,lat,long,az,el,ha,dec,soldst)
        write(6,'(//)')
        write(6,'("The solar azimuth and elevation are")')
        write(6,100)az,el
100     format(3x,f9.4,3x,f8.4,/)
        write(6,'("The solar hour angle and declination are")')
        write(6,100)ha,dec
        am=armass(el)
        write(6,'("The air mass is")')
        write(6,200)am
200     format(3x,f6.2)
      write(6,'("The solar distance is")')
      write(6,250)soldst
250     format(3x,f7.4)
        write(6,'(//)')
        write(6,'("If you want another time same day, type 9!")')
        read(5,'(i1)')i
        if(i.eq.9)go to 50
        write(6,'("If you want another day, type 99!")')
        read(5,'(i2)')i
        if(i.eq.99)go to 60
        end
c-------------------------------------------------------------
------+
c
|
c   subroutine sunae(year,day,hour,lat,long,az,el,ha,dec,soldst)
|
c
|
c   this subroutine calculates the local azimuth and elevation
of the |
c     sun at a specific location and time using an approximation
to   |
c     equations used to generate tables in The Astronomical
Almanac.  |
c     refraction correction is added so sun position is apparent
one. |
c
|
c   The Astronomical Almanac, U.S. Gov't Printing Office,
Washington, |
c     D.C. (1985).
|
c
|
c   input parameters
|
c     year=year, e.g., 1986
|
c     day=day of year, e.g., feb 1=32
|
c     hour=hours plus fraction in UT, e.g., 8:30 am eastern
```

VI. Annexe

```
daylight  |
c       time is equal to 8.5 + 5(5 hours west of Greenwich)
-1(for    |
c       daylight savings time correction)
|
c     lat=latitude in degrees (north is positive)
|
c     long=longitude in degrees (east is positive)
|
c
|
c   output parameters
|
c     a=sun azimuth angle (measured east from north, 0 to 360
degs)   |
c     e=sun elevation angle (degs)
|
c     plus others, but note the units indicated before return
|
c
|
c-----------------------------------------------------------------
------+
c
      subroutine
sunae(year,day,hour,lat,long,az,el,ha,dec,soldst)
c  work with real variables and define some constants, including
c  one to change between degs and radians
      implicit real (a-z)
      data twopi,pi,rad/6.2831853,3.1415927,.017453293/
c
c   get the current julian date (actually add 2,400,000 for jd)
      delta=year-1949.
      leap=aint(delta/4.)
      jd=32916.5+delta*365.+leap+day+hour/24.
c   1st no. is mid. 0 jan 1949 minus 2.4e6; leap=leap days since
1949
c  the last yr of century is not leap yr unless divisible by 400
      if(amod(year,100.).eq.0.0.and.amod(year,400.).ne.
0.0)jd=jd-1.
c
c   calculate ecliptic coordinates
      time=jd-51545.0
c   51545.0 + 2.4e6 = noon 1 jan 2000
c
c   force mean longitude between 0 and 360 degs
      mnlong=280.460+.9856474*time
      mnlong=mod(mnlong,360.)
      if(mnlong.lt.0.)mnlong=mnlong+360.
c
c   mean anomaly in radians between 0 and 2*pi
      mnanom=357.528+.9856003*time
      mnanom=mod(mnanom,360.)
      if(mnanom.lt.0.)mnanom=mnanom+360.
      mnanom=mnanom*rad
c
c   compute the ecliptic longitude and obliquity of ecliptic in
radians
      eclong=mnlong+1.915*sin(mnanom)+.020*sin(2.*mnanom)
      eclong=mod(eclong,360.)
      if(eclong.lt.0.)eclong=eclong+360.
      oblqec=23.439-.0000004*time
      eclong=eclong*rad
      oblqec=oblqec*rad
c
c   calculate right ascension and declination
      num=cos(oblqec)*sin(eclong)
      den=cos(eclong)
      ra=atan(num/den)
c   force ra between 0 and 2*pi
      if(den.lt.0)then
          ra=ra+pi
      elseif(num.lt.0)then
          ra=ra+twopi
      endif
c
c   dec in radians
      dec=asin(sin(oblqec)*sin(eclong))
c
c   calculate Greenwich mean sidereal time in hours
      gmst=6.697375+.0657098242*time+hour
c   hour not changed to sidereal time since 'time' includes
c   the fractional day
      gmst=mod(gmst,24.)
      if(gmst.lt.0.)gmst=gmst+24.
c
c   calculate local mean sidereal time in radians
      lmst=gmst+long/15.
      lmst=mod(lmst,24.)
      if(lmst.lt.0.)lmst=lmst+24.
      lmst=lmst*15.*rad
c
c   calculate hour angle in radians between -pi and pi
      ha=lmst-ra
```

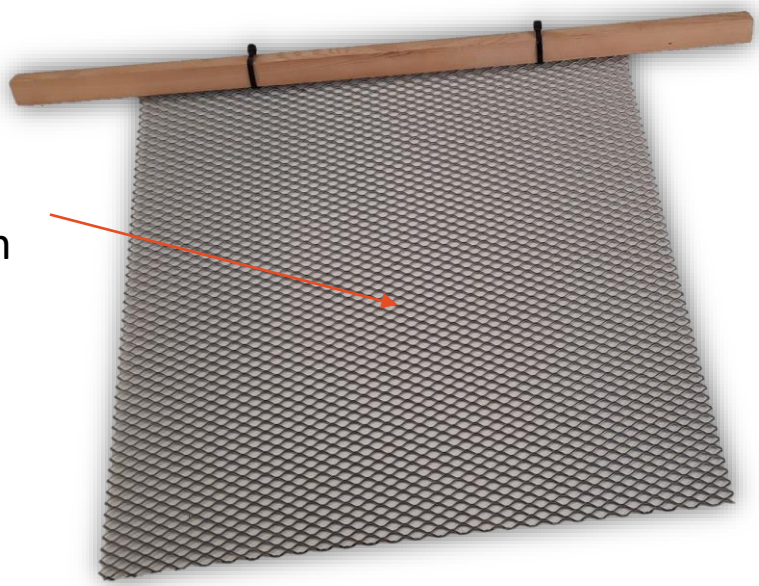
VI. Annexe

```
      if(ha.lt.-pi)ha=ha+twopi
      if(ha.gt.pi)ha=ha-twopi
c
c   change latitude to radians
      lat=lat*rad
c
c   calculate azimuth and elevation
      el=asin(sin(dec)*sin(lat)+cos(dec)*cos(lat)*cos(ha))
      az=asin(-cos(dec)*sin(ha)/cos(el))
c
c   this puts azimuth between 0 and 2*pi radians
      if(sin(dec)-sin(el)*sin(lat).ge.0.)then
      if(sin(az).lt.0.)az=az+twopi
      else
      az=pi-az
      endif
c
c   calculate refraction correction for US stan. atmosphere
c   need to have el in degs before calculating correction
      el=el/rad
c
      if(el.ge.19.225) then
         refrac=.00452*3.51823/tan(el*rad)
      else if (el.gt.-.766.and.el.lt.19.225) then
         refrac=3.51823*(.1594+.0196*el+.00002*el**2)/
     1   (1.+.505*el+.0845*el**2)
      else if (el.le.-.766) then
         refrac=0.0
      end if
c
c   note that 3.51823=1013.25 mb/288 C
      el=el+refrac
c   elevation in degs
c
c   calculate distance to sun in A.U. & diameter in degs
      soldst=1.00014-.01671*cos(mnanom)-.00014*cos(2.*mnanom)
      soldia=.5332/soldst
c
c   convert az and lat to degs before returning
      az=az/rad
      lat=lat/rad
         ha=ha/rad
         dec=dec/rad
c   mnlong in degs, gmst in hours, jd in days if 2.4e6 added;
c   mnanom,eclong,oblqec,ra,and lmst in radians
      return
      end
```

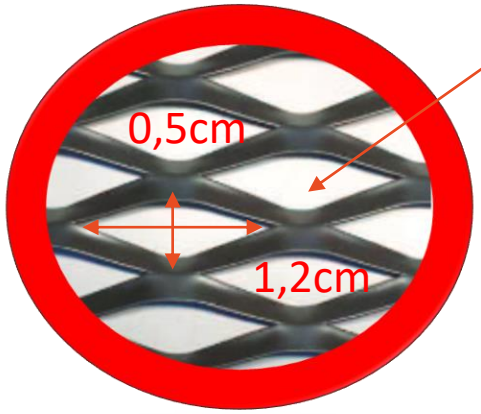
Langage Fortran

VI. Annexe

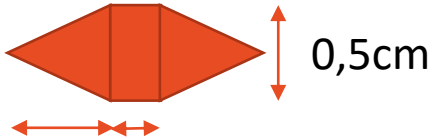
Dans la position optimale :
Aire d'un troue : 0,38cm²
Aire d'une maille : 0,48cm²



Bouclier de 40x40
d'épaisseur 2,8mm



Aire d'un troue : 0,35cm²
Aire d'une maille : 0,80cm²



Epaisseur d'une planche

Espace entre la vitre et le bouclier 2,5 cm

Vitre 40x40 d'épaisseur 0,4 mm