Devoir_3

February 29, 2016

1 Devoir #3 ENR382

1.1 Modules à importer

In [216]:
```
# Python
import numpy as np
import scipy as sp
import scipy.constants as const
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import pylab
import sys

import plotly
from plotly.offline import plot
init_notebook_mode()

# Externe
import solar_mod as sm
import properties_mod as pm
```

```
<IPython.core.display.HTML object>
```

1.2 Question 1

1.2.1 Importation et apperçu des données du devoir #2

In [217]:
```
data_d2 = pd.read_csv('../dev2/data_devoir2.csv')
print(data_d2)
#data_d2.head()
```

```
I  Ibn        Rb        omen        thez
0     0    0  2.728682 -185.932358  111.134497
1     0    0  2.746104 -170.932358  110.852047
2     0    0  2.958693 -155.932358  107.937769
3     0    0  3.590436 -140.932358  102.703818
4     0    0  6.351798 -125.932358   95.609050
5    27   98  0.000000 -110.932358   87.122329
6    99  110  0.000000  -95.932358   77.652628
7   280  411  0.109534  -80.932358   67.541652
8   432  499  0.487013  -65.932358   57.093909
9   543  423  0.673706  -50.932358   46.638033
```

```
10  636  387  0.775716  -35.932358   36.652838
11  802  579  0.830688  -20.932358   28.070626
12  796  521  0.854040   -5.932358   22.861140
13  791  519  0.851393    9.067642   23.530648
14  791  578  0.822137   24.067642   29.671890
15  609  348  0.759068   39.067642   38.665950
16  396  142  0.643653   54.067642   48.804472
17  339  175  0.429968   69.067642   59.290404
18  226  285  0.000000   84.067642   69.692861
19   83   86  0.000000   99.067642   79.695532
20   16    0  0.000000  114.067642   88.989998
21    0    0  5.245744  129.067642   97.222014
22    0    0  3.393977  144.067642  103.967806
23    0    0  2.892482  159.067642  108.752307
```

1.2.2 Données du problème

```
In [218]: T_pc = 52.0          # Temperature de la plaque (Celsius)
          T_pk = T_pc+273      # Temperature de la plaque (Kelvin)
          T_ac = 18            # Temperature ambiante (celsius)
          T_ak = T_ac+273      # Temperature ambiante (kelvin)
          h_w = 6.0              # coefficient de convection extérieur
          Lair  = 0.030        # epaisseur d'air
          alpha_n = 0.95       # Coef d'absortion solaire
          g = 9.8
          rhog = 0.4
          KL = 0.0125          # Propriété du vitrage
          n2=1.526             # indice de réfraction de la vitre
          n1=1                 # indice de réfraction de l'air
          eps_p = 0.17         # Emissivité de la plaque
          eps_c = 0.88          # Emissivitée du verre
          N = 1                # Nombre de vitrage
          beta = 60          # Angle à plat
          gam = 0              # plein sud
          phi  = 45.0 +30.0/60.0  # latitude  Montreal (45 deg 30 min nord)
```

1.2.3 Sélection des données de la 12e tranche

```
In [219]: tranche = 11
          I = data_d2.I[tranche]
          Ibn = data_d2.Ibn[tranche]
          n =sm.jour_mois_jour_annee(12,'juin')
          thez = data_d2.thez[tranche]
          delt  = sm.decl_solaire(n)
          omen = data_d2.omen[tranche]
          Ib = Ibn * sm.cosd(thez)
          Id = I-Ib
          Rb = sm.calcul_Rb(phi,n,omen,beta,gam)

          It,Ib,Id,Ir = sm.modele_isotropique(I,Ib,Id,beta,Rb,rhog)

          print('omen =',omen)
          print('Rb = ',Rb)
          print('It = ',It)
          print('Ib = ',Ib)
```

```
          print('Id = ',Id)
          print('Ir =',Ir)
          print('thez = ',thez)
```

```
omen = -20.932357638
Rb =  0.83068791593
It =  722.922746543
Ib =  424.391149777
Id =  218.331596766
Ir = 80.2
thez =  28.0706255912
```

1.2.4 Calculs du problème

Calcul des angles réfléchi et diffus

```
In [220]: the_g = sm.angle_reflechi(beta)
          the_d = sm.angle_diffus(beta)
          #the = abs(thez-beta)

          the = sm.normale_solaire(delt,phi,omen,beta,gam)

          print('the = ',the)
          print('the_g = ',the_g)
          print('the_d = ',the_d)
```

```
the =  42.8638179755
the_g =  64.9668
the_d =  56.7612
```

Calcul du coéfficient absorbeur pour faible longueur d'ondes

```
In [221]: tau_al_b = sm.Calcul_tau_al(the,alpha_n,KL,n2,n1,N)
          tau_al_g = sm.Calcul_tau_al(the_g,alpha_n,KL,n2,n1,N)
          tau_al_d = sm.Calcul_tau_al(the_d,alpha_n,KL,n2,n1,N)

          print('tau_al_b =', tau_al_b)
          print('tau_al_g =', tau_al_g)
          print('tau_al_d =', tau_al_d)
```

```
tau_al_b = 0.832787232782
tau_al_g = 0.688465506364
tau_al_d = 0.772853382109
```

Calcul de la radiation totale transmise

```
In [222]: Ibt = Ib*tau_al_b
          Idt = Id*tau_al_d
          Irt = Ir*tau_al_g
          S =  Ibt+Idt+Irt

          print('Radiation directe transmise =',Ibt)
          print('Radiation diffuse transmise=',Idt)
          print('Radiation réfléchie transmise =',Irt)
          print('Radiation totale transmise (S) =',S)
```

```
Radiation directe transmise = 353.42753124
Radiation diffuse transmise= 168.738312982
Radiation réfléchie transmise = 55.2149336104
Radiation totale transmise (S) = 577.380777832
```

Calcul des pertes par model itératif Pour une plaque simple, le coéfficient de perte vers le haut est exprimé grâce à cette formule.

Premièrement, il faut trouver chacun des termes du dénominateur. Les formules suivantes seront utilisées. Pour le coéfficient de radiation de la plaque interne jusqu'à la surface externe, Pour ce qui est de la radiation de la surface externe vers l'air ambiante, La convection entre les deux surfaces ($ h_{c,p-c} $) sera déterminé grâce à la fonction suivante, Pour enfin trouver le coéfficient de température de la face externe du capteur.

In [238]:
```
T_cc = 40 # Température initiale (hypothèse)
T_old = 1
converg = []
x_it = 0
x_conv = []

while (abs(T_old-T_cc) > 0.0001):

    converg.append(T_cc)
    T_old = T_cc

    T_ck = T_cc + 273.0
    T_avg = (T_ck+T_pk)/2

    hr_pc = (const.sigma * (T_pk**2+T_ck**2) * (T_pk+T_ck))/((1/eps_p)+(1/eps_c)-1)

    hr_ca = eps_c*const.sigma*(T_ck**2+T_ak**2)*(T_ck+T_ak)

    nu = pm.air_prop('nu',T_avg)
    al = pm.air_prop('al',T_avg)
    k = pm.air_prop('k',T_avg)
    Ra = const.g*(1/T_avg)*abs(T_ck-T_pk)*Lair**3/(nu*al)

    f1 = max(0,1.0-1708.0/(Ra*sm.cosd(beta)))
    f2 = 1.0-1708*(sm.sind(1.8*beta))**1.6/(Ra*sm.cosd(beta))
    f3 = max(0,(Ra*sm.cosd(beta)/5830)**(1.0/3.0)-1.0)
    Nu = 1.0 + 1.44*f1*f2+f3
    hc_pc = Nu * k/Lair

    T_p = T_avg-273.0


    Ut = ((1/(hc_pc + hr_pc))+(1/(h_w+hr_ca)))**-1

    T_cc = T_p - ((Ut*(T_p-T_ac))/(hc_pc+hr_pc))
    x_conv.append(x_it)
    x_it = x_it + 1

q = Ut*(T_pc-T_ac)

#plotly.offline.iplot({
#"data": [{
```

```
#    "x": x_conv,
#    "y": converg
#}],
#"layout": {
#    "title": "Convergence de la température externe"
#}
#})
```

```
print('Le coéfficient de perte vers le haut =',Ut, 'W/m² °C')
print('Les pertes vers le haut =', q, 'W/m²')
print('Ce qui est capté',total_iteratif,'W/m²' )
print('La température de la surface extérieure =',T_cc,'°C')
```

```
Le coéfficient de perte vers le haut = 2.8585751874 W/m² °C
Les pertes vers le haut = 97.1915563716 W/m²
Ce qui est capté 480.189221461 W/m²
La température de la surface extérieure = 23.0521700256 °C
```

Calcul des pertes par modèle empirique Klein En plus de calculer avec la méthode itérative, la méthode Klein est utilisé afin de comparer les résultats.

In [237]:
```
pertes_empirique_coef = sm.U_Klein(T_pc,T_ac,beta,h_w,eps_p,eps_c,N)
pertes_empirique = pertes_empirique_coef*(T_pc-T_ac)
total_empirique = S-pertes_empirique
```

```
print('Coefficient de perte vers le haut Klein',pertes_empirique_coef,'W/m² °C')
print('Pertes vers le haut =',pertes_empirique, 'W/m²')
print('Ce qui est capté =',total_empirique, 'W/m²')
```

```
Coefficient de perte vers le haut Klein 0.7025954760782759 W/m² °C
Pertes vers le haut = 23.88824618666138 W/m²
Ce qui est capté = 553.492531646 W/m²
```

Calcul du rendement du capteur

In [226]:
```
Rendement_empirique = total_empirique/It
Rendement_iteratif = total_iteratif/It
print('Le rendement du capteur est de',Rendement_empirique,'pour la méthode empirique')
print('Le rendement du capteur est de',Rendement_iteratif,'pour la méthode itérative')
```

```
Le rendement du capteur est de 0.661970049777 pour la méthode empirique
Le rendement du capteur est de 0.664233106175 pour la méthode itérative
```

1.2.5 Résultats

In [227]:
```
print('a)',S,'J/m²')
print('b)',q, 'W/m²')
print('c)',Rendement_iteratif)
```

```
a) 577.380777832 J/m²
b) 97.1915563716 W/m²
c) 0.664233106175
```

1.3 Question 2

1.3.1 Données du problème

In [228]:
```
H= 0.8
N=8
Y=2
Ac=H*Y # Surface des capteurs
W=0.1 # Distance entre les tubes (m)
D = 0.007 # m
Cb=100.0 # W/mK
P=2*(H+Y) # Perimetre du capteur
deltaa= 0.001 #epaisseur de la plaque (m)
ka=400.0  # W/m2K
mpt=0.016 # Debit total Kg/s
hf=1100.0 #W/m2K
Cp=4180.0   #J/KgK
Rpjoint = 1/Cb    #hypoth?se Conductivite joint est infinie
Ti=18.0
UL=Ut
```

Calcul du rendement d'ailette F

In [229]:
```
m = np.sqrt(UL/(ka*deltaa))
```

In [230]:
```
F = np.tanh((m*(W-D)/2))/(m*(W-D)/2)
```

Calcul du rendement d'absorbeur F'

In [231]:
```
Fp = (1.0/UL)/(W*(1.0/(UL*(D+(W-D)*F))+Rpjoint+(1.0/(hf*const.pi*D))))
print('Fp =',Fp)
```

```
Fp = 0.980912220305
```

Calcul du rendement d'absorbeur F"

In [232]:
```
Fpp1 = (mpt*Cp)/(Ac*UL*Fp)
Fpp2 =  1-np.exp(-(Ac*UL*Fp)/(mpt*Cp))
Fpp = Fpp1 * Fpp2
print(Fpp)
```

```
0.967196774224
```

Calcul du facteur FR

In [233]:
```
Fr = Fpp*Fp
print(Fr)
```

```
0.948735135277
```

Calcul de la température à l'entrée du capteur En isolant Tfi de la formule, on obtient,

In [234]:
```
Qu = Ac*Fr*(S-UL*(Ti-T_ac))
Tfi = T_pc-((Ac)/(Fr*UL))*(1-Fpp)
```

1.3.2 Résultats

```
In [235]: print('a)',Fr)
          print('b)',Tfi,'°C')
```

```
a) 0.948735135277
b) 51.9806472836 °C
```