optimisation

Gabriel HAYOUN, Chloé GOMICHON

May 8, 2022

Part I
Etude du problème d'optimisation

1 Q1

On a, en notant la facture d'électricité $f_{electricity}$

$$\begin{aligned} f_{electricity} &= \int_{t_0}^{t_f} c(t)P(t)dt \\ &= \sum_{i=0}^{N-1} \int_{t_i}^{t_{i+1}} c_i P_i dt \\ &= \sum_{i=0}^{N-1} c_i P_i \Delta t \\ &= \Delta t \sum_{i=0}^{N} c_i P_i \end{aligned}$$

Donc on a bien

$$\boxed{f_{electricity} = \Delta t \sum_{i=0}^{N} c_i P_i}$$

2 Q2

On isole le système constitué de l'habitation. On va appliquer le premier principe entre t et t+dt. On sait que :
- le système reçoit l'énergie $rP(t)dt$ avec r un rendement
- il échange avec l'air extérieur suivant la loi de conducto-convection de Newton ce qui correspond à une énergie $h(T_{ext} - T(t))Sdt$ avec h le coefficient de conducto-convection et S la surface d'échange.

Le premier principe s'écrit alors, en écrivant c la capacité thermique équivalente du système :

$$c(T(t+dt) - T(t)) = rP(t)dt + h(T_{ext}(t) - T(t))Sdt$$

De sorte que :

$$cT'(t) = rP(t) + hS(T_{ext}(t) - T(t))$$

Si l'on résout cette équation différentielle sur un intervalle de type $[t_i, t_{i+1}]$, sur lequel la puissance est constante égale à P_i et la température extérieure constante égale à T_i^e, on peut réecrire

$$cT'(t) = bP_i + Sh(T_i^e - T(t))$$

Ou encore :

$$T'(t) + \frac{Sh}{c}T = \frac{r}{c}P + \frac{Sh}{c}T_i^e \quad (1)$$

La solution homogène associée à cette équation différentielle est :

$$T_h(t) = e^{-\frac{Sh}{c}(t-t_i)}T_i$$

Il nous faut donc identifier les valeurs de b, k et h telles que $\frac{1-e^{-\frac{Sh}{c}(t-t_i)}}{k+h}(bP_i + hT_i^e)$ soit solution particulière de (1). On remarque que $b = \frac{r}{c}$, $k = \frac{Sh}{c}$ et $h = 0$ conviennent. Cependant, cela n'est pas satisfaisant car h n'est pas nul dans les exemples. En fait, on trouverait deux valeurs pour k et h si dans (1), le coefficient précédent $T(t)$ et celui précédant T_i^e étaient différents. Nous n'avons pas trouvé quel phénomène physique amènerait à cela.

3 Q3

On doit minimiser $f_{electricity}$, qui est fonction à la fois de la puissance fournie aux convecteurs et de la température de l'habitat.

Ainsi, on a en notant $c = \begin{pmatrix} c_0 \\ \vdots \\ c_N \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ et $x = \begin{pmatrix} P_0 \\ \vdots \\ P_N \\ T_0 \\ \vdots \\ T_N \end{pmatrix}$,

on remarque que la fonction à minimiser est f définie ainsi :

$$\boxed{\begin{array}{rccc} f: & \mathcal{R}^{2(N+1)} & \longrightarrow & \mathcal{R} \\ & x & \longmapsto & \Delta t c^T x \end{array}}$$

On impose $\forall i \in [\![0, N]\!],\ 0 \leq P_i \leq P_M$ et $\forall j \in \mathcal{I}_{occ},\ T_m \leq T_j \leq T_M$.
En notant, $\mathcal{I}_{occ} = \{j_1, ..., j_{\tilde{n}}\}$, on peut définir c_{in} :

$$\boxed{\begin{array}{rccc} & \mathcal{R}^{2(N+1)} & \longrightarrow & \mathcal{R}^{2(N+1+\tilde{n})} \\ c_{in}: & x & \longmapsto & \begin{pmatrix} -P_0 \\ \vdots \\ -P_N \\ P_0 - P_M \\ \vdots \\ P_N - P_M \\ T_m - T_{j_1} \\ \vdots \\ T_m - T_{j\tilde{n}} \\ T_{j_1} - T_M \\ \vdots \\ T_{j\tilde{n}} - T_M \end{pmatrix} \end{array}}$$

De plus on a : $P_N = 0,\ \ T_0 = T_{in}$ et

$$\forall i \in [\![1, N]\!],\ T_i = e^{-(k+h)\Delta t} T_{i-1} + \frac{1 - e^{-(k+h)\Delta t}}{k+h}(bP_{i-1} + hT^e_{i-1})$$

On a donc pour c_{eq},

$$\boxed{\begin{array}{rccc} & \mathcal{R}^{2(N+1)} & \longrightarrow & \mathcal{R}^{N+2} \\ c_{eq}: & x & \longmapsto & \begin{pmatrix} P_N \\ T_0 - T_{in} \\ e^{-(k+h)\Delta t} T_0 + \frac{1-e^{-(k+h)\Delta t}}{k+h}(bP_0 + hT^e_0) - T_1 \\ \vdots \\ e^{-(k+h)\Delta t} T_{N-1} + \frac{1-e^{-(k+h)\Delta t}}{k+h}(bP_{N-1} + hT^e_{N-1}) - T_N \end{pmatrix} \end{array}}$$

4 Q4

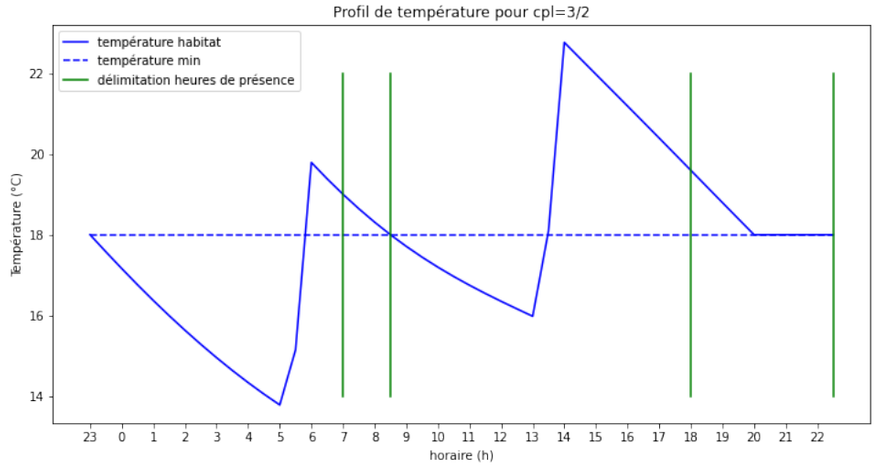
On remarque que f, c_{in} et c_{eq} sont linéaires donc convexes.
En fait, **on se trouve le cadre d'un problème LP** : on cherche à minimiser $c^T x$ avec c un vecteur constant et x la variable de décision, le tout sous contraintes affines.

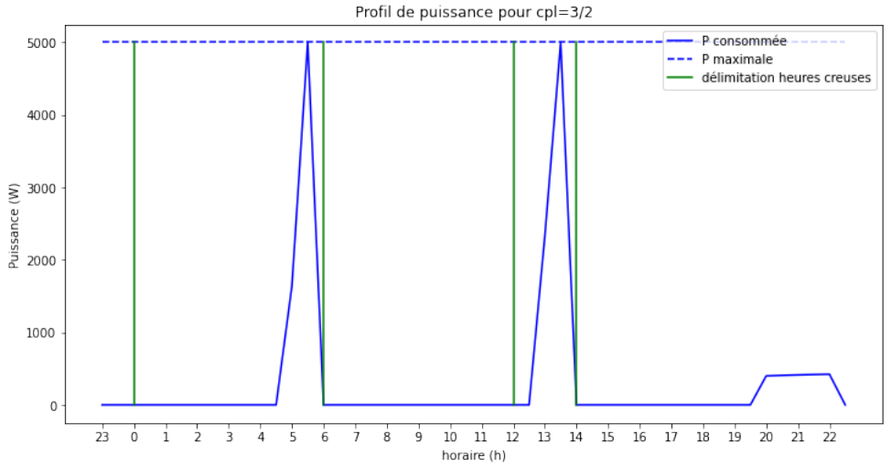
Part II
Etude et résolution numérique du problème individuel

5 Q5

Dans le cadre d'un problème LP, le calcul du gradient et la Hessienne de la fonction à minimiser est aisé donc les méthodes classiques de descente sont une possibilité (gradient à pas constant, à pas optimal, voire même stochastique en cas de trop grande dimension, Newton...). Cependant, **le plus adapté à la particularité du problème est l'algorithme du simplexe, spécifique au cas LP**.

6 Q6-commentaires sur les courbes

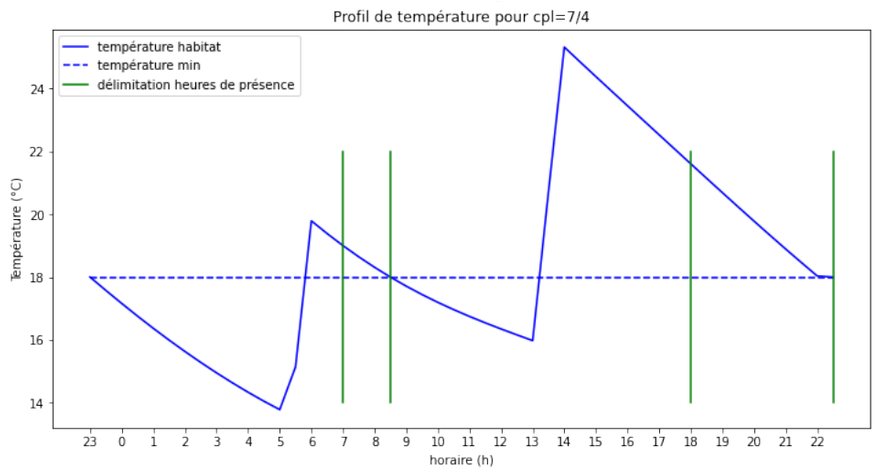


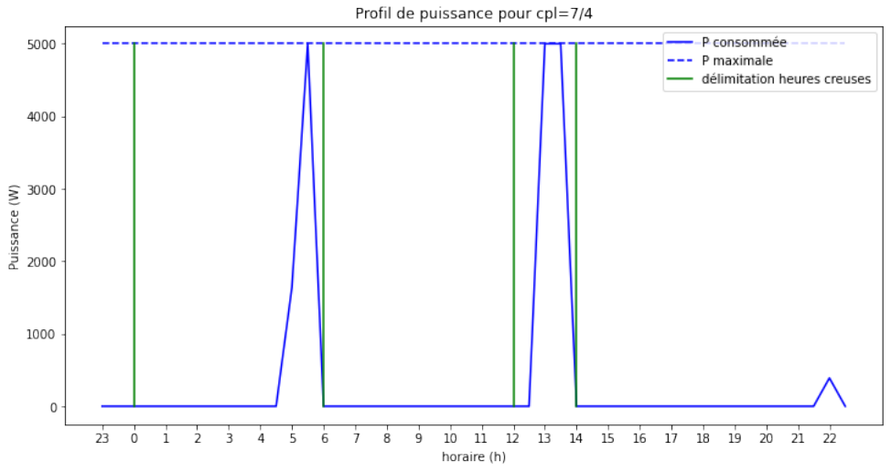


Commentaires généraux
On remarque sur les courbes que la température du foyer respecte bien les contraintes.
De plus, au niveau du chauffage on remarque que l'utilisateur pour minimiser sa facture devrait allumer son chauffage durant les heures creuses de telle sorte que la grande majorité de la puissance consommée le soit en heure creuse. On remarque donc des pics de chauffage juste avant la fin des heures creuses. Ils permettent de chauffer en prévision de la présence des habitants tout en payant moins cher.
Cependant, sans chauffage, la température baisse spontanément et il faut, pour maintenir le confort de l'utilisateur et éventuellement rallumer le chauffage à partir de 19h environ même s'il ne s'agit pas d'une heure creuse.

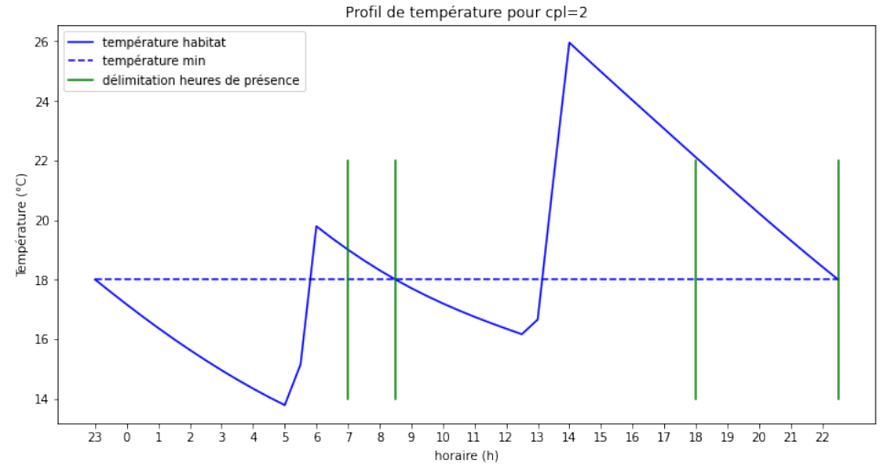
Influence du prix en heure pleine

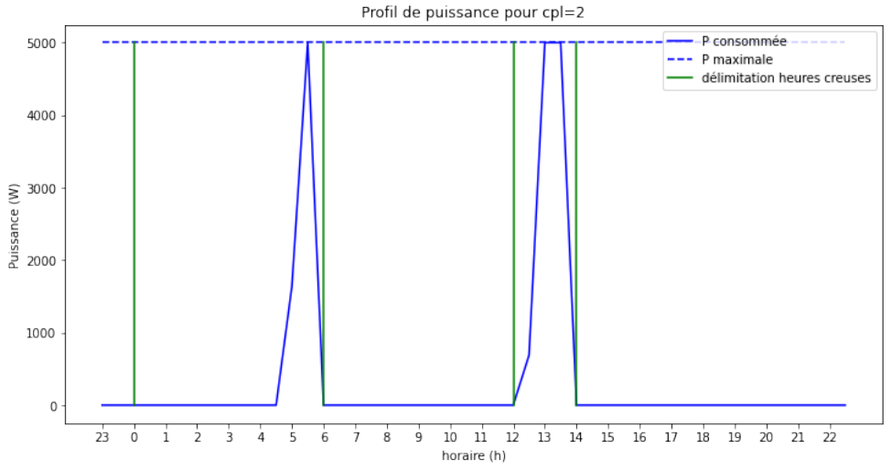
pour $cpl = \frac{7}{4}$ **:**





pour $cpl = 2$ **:**





On remarque que plus le prix en heure pleine augmente, plus la puissance de chauffage sur l'intervalle d'heure creuse $[12h, 14h]$ est élevé et plus la puissance délivrée autour de $19h$ diminue jusqu'à disparaître. Quand c_{pl} vaut 2, on ne chauffe plus qu'en heure creuse.
En effet, le coût des heures pleines devient tellement important qu'il est alors plus avantageux de chauffer beaucoup en heure creuse (quitte à s'approcher de la température maximale en sortie de période creuse) pour que, ensuite, malgré la baisse spontanée de température en l'absence de chauffage on reste au-dessus de la température minimiale : on évite alors de réchauffer autour de 19h (en heure pleine) et l'on fait des économies.

7 Q7

Les heures creuses permettent au régulateur du réseau RTE d'inciter les utilisateurs à consommer aux moments où il n'y a pas de tension d'approvisionnement. Les utilisateurs ne chauffent ainsi plus uniquement lorsqu'ils en ressentent le besoin, tous en même temps, mais certains se mettent à chauffer en prévision de leur présence durant les heures creuses. Cela répartit de façon plus homogène dans le temps la demande.

Part III
Régulation collective

8 Q8

8.1

On écrit cette fois la variable de décision x telle que : $x = \begin{pmatrix} P_0^1 \\ \vdots \\ P_N^1 \\ P_0^2 \\ \vdots \\ P_N^2 \\ \vdots \\ P_N^{n_l} \\ T_0^1 \\ \vdots \\ T_N^1 \\ T_0^2 \\ \vdots \\ T_N^2 \\ \vdots \\ T_N^{n_l} \end{pmatrix}$

On note c_r le vecteur suivant : $c_r = \begin{pmatrix} c_0 \\ \vdots \\ c_N \end{pmatrix}$ Le vecteur associé au coût est cette fois la concaténation de n_l vecteurs c_r puis de $(N+1)(n_l-1)$ composantes nulles. On note ce vecteur associé à la situation collectif c_{coll}.
Avec cette écriture, la forme de la fonction à minimiser est exactement la même.

Cherchons la forme de contrainte inégalité adaptée dans ce cas collectif notée $c_{in,coll}$ analogue à c_{in} exprimée dans le cas précédent.

On note, pour j entre 1 et n_l $c_{in,P^j} = \begin{pmatrix} -P_0^j \\ \vdots \\ -P_N^j \\ P_0^j - P_M \\ \vdots \\ P_N^j - P_M \end{pmatrix}$ et $c_{in,T^j} = \begin{pmatrix} T_m - T_{j_0}^j \\ \vdots \\ T_m - T_{j\,\tilde{n}}^j \\ T_{j_0}^j - T_M \\ \vdots \\ T_{j\,\tilde{n}}^j - T_M \end{pmatrix}$

La contrainte inégalité associée au problème collectif est la concaténation des vecteurs de type c_{in,P^j} et de ceux de type c_{in,T^j}.

La contrainte égalité, elle, va être plus profondément modifiée. On va noter cette fois-ci

$$c_{eq,j} = \begin{pmatrix} P_N^j \\ T_0^j - T_{in}^j \\ e^{-(k+h+\sum_{k\neq j} h_{jk})\Delta t} T_0^j + \frac{1-e^{-(k+h+\sum_{k\neq j} h_{jk}))\Delta t}}{k+h+\sum_{k\neq j} h_{jk})} (bP_0^j + hT_0^{j\,e} + \sum_{k\neq j} h_{jk} T_0^k)) - T_1 \\ \vdots \\ e^{-(k+h+\sum_{k\neq j} h_{jk}))\Delta t} T_{N-1}^j + \frac{1-e^{-(k+h+\sum_{k\neq j} h_{jk}))\Delta t}}{k+h+\sum_{k\neq j} h_{jk})} (bP_{N-1}^j + hT_{N-1}^{j\,e} + \sum_{k\neq j} h_{jk} T_{N-1}^k))) - T_N^j \end{pmatrix}$$

De sorte que l'on cherche à annuler la contrainte formée de la concaténation des $c_{eq,j}$.

8.2

L'écriture de la fonction à minimiser est toujours de la forme $c^T x$ avec c un vecteur constant et x la variable de décision, et les contraintes sont toujours affines. Ainsi, on est toujours dans le cadre LP : **les propriétés du problème d'optimisation sont inchangées**.

D'un point de vue pratique, cette solution prend en compte les transferts entre les habitations et peut donc améliorer la précision de calcul. En effet, si l'on ne prend pas en compte cette dépendance, on peut d'une part surchauffer une maison en ne prenant pas en compte qu'elle reçoit de la chaleur des voisins (ce qui couterait au passage de l'argent) ou au contraire sous-chauffer une maison (ce qui nuit au confort) en oubliant qu'elle cède une partie de sa chaleur aux voisins.

9 Q9

9.1

Pour ce qui est de la variable de décision x :
Le vecteur x peut s'écrire sous la forme :

$$x = \begin{pmatrix} P_0^1 \\ \vdots \\ P_N^1 \\ T_0^1 \\ \vdots \\ T_N^1 \\ P_0^2 \\ \vdots \\ P_N^2 \\ T_0^{n_l} \\ \vdots \\ T_N^{n_l} \\ P_0^{n_l} \\ \vdots \\ P_N^{n_l} \end{pmatrix} = \begin{pmatrix} x_1 \\ \vdots \\ x_{n_l} \end{pmatrix} \quad avec \quad x_i = \begin{pmatrix} P_0^i \\ \vdots \\ P_N^i \\ T_0^i \\ \vdots \\ T_N^i \end{pmatrix}$$

Pour ce qui est de la fonction à minimiser : Le coût total est écrit sous la forme :

$$c = \sum_{j=1}^{n_l} \sum_{i=0}^{N} (\Delta t c_i P_i^j + \epsilon (P_i^j)^2)$$

Cela peut se réécrire : $c = \sum_{j=1}^{n_l} f_j(x_j)$ avec f_j la fonction qui à x un vecteur de R^{N+1} de la forme x_j associe le réel $\sum_{i=0}^{N} (\Delta t c_i P_i^j + \epsilon (P_i^j)^2)$.

Pour ce qui est des contraintes :
On a pour chaque utilisateur la contrainte égalité suivante :

$$c_{eq,j} : \begin{pmatrix} P_N^j \\ T_0^j - T_{in}^j \\ e^{-(k+h+\sum_{k\neq j} h_{jk})\Delta t} T_0^j + \frac{1-e^{-(k+h+\sum_{k\neq j} h_{jk}))\Delta t}}{k+h+\sum_{k\neq j} h_{jk})} (bP_0^j + hT_0^{j\,e} + w_0) - T_1 \\ \vdots \\ e^{-(k+h+\sum_{k\neq j} h_{jk}))\Delta t} T_{N-1}^j + \frac{1-e^{-(k+h+\sum_{k\neq j} h_{jk}))\Delta t}}{k+h+\sum_{k\neq j} h_{jk})} (bP_{N-1}^j + hT_{N-1}^{j\,e} + w_{N-1}) - T_N^j \end{pmatrix}$$

Il faut avant tout noter que cette contrainte est indépendante des autres utilisateurs puisque la famille $(w_i)_{0\leq i\leq N-1}$ est une donnée. De plus, elle peut être traitée comme une contrainte inégalité en concaténant la $c_{eq,j}$ et $-c_{eq,j}$.

Les contraintes inégalités, elles, ont toujours été propres à un utilisateur.

Ainsi, les contraintes peuvent bien s'écrire sous la forme $c(x) \leq 0$ avec $c(x) = \sum_{j=1}^{n_l} c_j(x_j)$, en écrivant $c_i(x_i)$ le vecteur contrainte propre à l'utilisateur i dont les composantes sont soit nulles, soit celles de $c_{eq,j}$,$-c_{eq,j}$ ou $c_{in,j}$. Les lignes nulles sont celles qui seront non nulles pour un autre utilisateur de sorte qu'en faisant la somme, le vecteur $c(x)$ n'a aucune composante nulle.

9.2

On peut proposer plusieurs **avantages** pour cette méthode :

- la résolution des problèmes un par un évite les propagations d'erreurs
- on décompose le problème global en n sous-problèmes et cela peut être un gain de temps. En effet, si l'on décompose le problème global en n sous-problèmes, on peut imaginer lancer la résolution des n sous problèmes en parallèle sur n machines, qui communiquent, puis centraliser le résultat, et le temps de calcul sera donc uniquement le temps de résolution d'un des petits problèmes.
- décomposer en n sous-problèmes peut permettre de gagner en performance en identifiant pour chacun d'eux l'algorithme de résolution le plus adapté.
- cela permet aussi d'identifier les sous-problèmes influençant le plus le résultat final ou les plus sensibles à la modification d'un paramètre.

Les éventuels **inconvénients** sont les suivants :

- cette solution nécessite la connaissance de la somme. De manière générale, il faut trouver un moyen de coordonner les sous-problèmes pour bien amener à la résolution du problème global.

9.3 Remarques sur notre code Python

Remarques sur notre code : Pour la valeur de w_i nous avons pris les valeurs trouvés avec la résolution faite à la question précédente. Nous avons tenté de résoudre des deux façons indiquées sur Moodle (que vous avons appelées V1 et V2). La première, avec ajout d'un terme quadratique pour éviter la divergence converge mais ne donne pas des résultats très satisfaisants (cf commentaires sur les courbes faits juste après). La deuxième ne marche pas (code défaillant), pourriez-vous, si vous en avez le temps, nous indiquer ce qui pose problème selon vous ? Merci !

```
def c1(x) :
    c1 = []
    #contraintes égalitées
    c1 += [x[n-1]]
    c1 +=[-x[n-1]]
    c1 += [x[n]-T_in]
    c1 += [-(x[n]-T_in)]
    J = [i for i in range(4)]
    I =[]

    for i in range (1,n):
        c1 +=[np.exp(-(k+h+h)*deltaT)*x[n+i-1]
              +(1-np.exp(-(k+h+h)*deltaT))/(k+h+h)*(b*x[i-1]+h*Text[i-1]+h*T2[i-1])
                -x[n+i]]
        I +=[len(c1)-1]
    for i in range (4,len(c1)):
        c1 += [-c1[i]]
        I +=[len(c1)-1]

    #contraintes inégalitées
    for i in range (n):
        c1+=[-x[i]]
        J+=[len(c1)-1]
        c1+=[x[i]-P_M]
        J+=[len(c1)-1]
    for i in I_occ :
        c1+=[T_m1-x[n+i]]
        J+=[len(c1)-1]
        c1+=[-T_M1+x[n+i]]
        J+=[len(c1)-1]
    return c1

def c2 (y):
    c2 = []
    #contraintes égalitées
    c2 += [y[n-1]]
    c2 += [y[n]-T_in]
    c2 += [-y[n-1]]
    c2 += [-(y[n]-T_in)]
    for i in range (1,n):
        c2 +=[np.exp(-(k+h+h)*deltaT)*y[n+i-1]
              +(1-np.exp(-(k+h+h)*deltaT))/(k+h+h)*(b*y[i-1]+h*Text[i-1]+h*T1[i-1])
                -y[n+i]]
    for i in range (4,len(c2)):
        c2 += [-c2[i]]

    #contraintes inégalitées
    for i in range (n):
        c2+=[-y[i]]
        c2+=[y[i]-P_M]
    for i in I_occ :
        c2+=[T_m2-y[n+i]]
        c2+=[-T_M2+y[n+i]]
    return c2
```

V1 : résolution avec ajout d'un terme quadratique

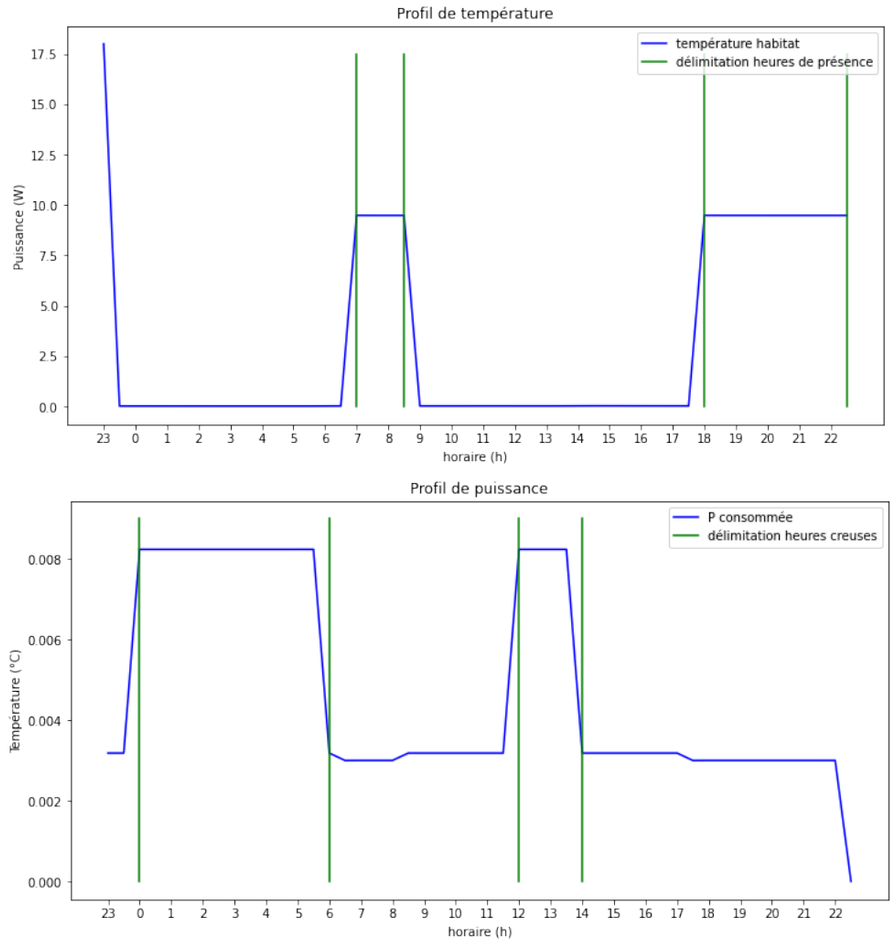
```
eps = 0.01
rho = 0.1
Niter = 40
n = 48

# construction du vecteur coût
c = np.array([cpl for i in range(n)]) # heures pleines

# heures creuses
for i in range (2,2+6*2):
    c[i] = ccr
for i in range (26,30):
    c[i] = ccr
c= np.concatenate((c,np.zeros(n)))

# initialisation de lambda utilisé dans le lagrangien
lambdak = np.ones((2*n+2*(n+1)+2*len(I_occ)))
lambdak1 = np.ones((2*n+2*(n+1)+2*len(I_occ)))
k=0

while (np.linalg.norm(lambdak1 - lambdak)>eps or k ==0) and k+1<Niter :
    k+=1

    lambdak = np.copy(lambdak1)

    # résolution pour l'habitat 1

    x = opti.variable(2*n) # variable de décision pour l'habitat 1

    f1 = 0
    for i in range (n):
        f1 += deltaT*c[i]*x[i]+eps*(x[i]**2+x[n+i]**2)
    for i in range (len(c1(x))) :
        f1 += lambdak[i]*c1(x)[i]
    opti.minimize(f1)

    opti.subject_to()
    x0 = np.concatenate((np.array([2500. for i in range(n)]),
                                np.array([T_in for i in range (n)])))
    opti.set_initial(x,x0)
    opti.solver('ipopt')
    sol = opti.solve()
    x = sol.value(x)

    # idem pour l'habitat 2

    y = opti.variable(2*n) # variable de décision pour l'habitat 2

    f2 = 0
    for i in range (n):
        f2 += deltaT*c[i]*y[i]+eps*(y[i]**2+y[n+i]**2)
    for i in range (len(c2(y))) :
        f2 += lambdak[i]*c2(y)[i]
    opti.minimize(f2)

    opti.subject_to()
    x0 = np.concatenate((np.array([2500. for i in range(n)]),
                                np.array([T_in for i in range (n)])))
    opti.set_initial(y,x0)
    opti.solver('ipopt')
    sol = opti.solve()
    y = sol.value(y)

    for i in range (len(lambdak)): # mise à jour du multiplicateur de Lagrande
        lambdak1[i]=max(0,lambdak[i]+rho*(c1(x)[i]+c2(y)[i]))
```

V2 : résolution avec décomposition uniquement des contraintes "mixtes"

On récupère I et J, en faisant tourner une fonction identique à c1 appelé c1IJ afin d'avoir I et J en variable global.
J1 et J2 sont identique en effet la construction de c1 et c2 est symétrique.

```
eps = 0.01
rho = 0.1
Niter = 40
n = 48

# construction du vecteur coût

c = np.array([cpl for i in range(n)]) # heures pleines

# heures creuses
for i in range (2,2+6*2):
    c[i] = ccr
for i in range (26,30):
    c[i] = ccr
c= np.concatenate((c,np.zeros(n)))

# initialisation du multiplicateur de Lagrange
lambdak = np.ones((2*n+2*(n+1)+2*len(I_occ)))
lambdak1 = np.ones((2*n+2*(n+1)+2*len(I_occ)))
k=0

while (np.linalg.norm(lambdak1 - lambdak)>eps or k ==0) and k+1<Niter :
    k+=1
    lambdak = np.copy(lambdak1)

    # résolution pour l'habitat 1
    x = opti.variable(2*n)

    f1 = 0
    for i in range (n):
        f1 += deltaT*c[i]*x[i]+eps*x[i]**2
    for i in I : # uniquement pour i dans I
        f1 += lambdak[i]*c1(x)[i]
    opti.minimize(f1)

    for i in J: # uniquement pour i dans J
        opti.subject_to(c1(x)[i]<=0)

    x0 = np.concatenate((np.array([2500. for i in range(n)]),np.array([T_in for i in range (n)])))
    opti.set_initial(x,x0)
    opti.solver('ipopt')
    sol = opti.solve()
    x = sol.value(x)

    # résolution pour l'habitat 1
    y = opti.variable(2*n)

    f2 = 0
    for i in range (n):
        f2 += deltaT*c[i]*y[i]+eps*y[i]**2
    for i in I :
        f2 += lambdak[i]*c2(y)[i]
    opti.minimize(f2)

    for i in J:
        opti.subject_to(c2(y)[i]<=0)

    x0 = np.concatenate((np.array([2500. for i in range(n)]),np.array([T_in for i in range (n)])))
    opti.set_initial(y,x0)
    opti.solver('ipopt')
    sol = opti.solve()
    y = sol.value(y)

    for i in I:
        lambdak1[i]=max(0,lambdak[i]+rho*(c1(x)[i]+c2(y)[i]))
```

9.4 Commentaires sur les courbes obtenues



On remarque que les ordres de grandeur en puissance ne sont absolument pas satisfaisants. En terme de température, l'ordre de grandeur est bon, mais il semble qu'une translation de quelques degrés supplémentaires aurait été plus cohérente : notamment, la contrainte de température minimale n'est pas respectée... En revanche, la puissance est majoritairement développée durant les heures creuses, ce qui est un point positif. La température est bien maximale durant les heures de présence ce qui est aussi un bon point.