CM08/P18/TD3 - Introduction à l'optimisation : problèmes transitoires

1 Optimisation d'une seule température

On considère un réacteur batch non isotherme avec deux réactions parallèles d'ordre 1

$$A \xrightarrow{k_1} B, \quad A \xrightarrow{k_2} C, \quad k_1 = k_{10}\exp\left(-\frac{E_1}{RT}\right), \quad k_2 = k_{20}\exp\left(-\frac{E_2}{RT}\right),$$

dont la cinétique sur l'intervalle de temps $[0,t_f]$ est donnée par les deux équations différentielles

$$A' = -(k_1+k_2)A, \quad A(0) = A_0, \tag{1}$$
$$B' = k_1 A, \quad B(0) = 0, \tag{2}$$

où on a noté A,B les concentrations respectives. On veut maximiser la concentration $B(t_f)$ en agissant sur le profil de température en fonction du temps, tout en respectant $T \leq T_{max}$.

1.1 Simplification des équations différentielles

Montrer qu'en posant

$$A(t) = A_0\, a(t), \quad B = A_0 b(t), \quad \beta = \frac{E_2}{E_1}, \quad \alpha = \frac{k_{20}}{k_{10}^{\beta}},$$

on peut écrire (1)-(2) sous la forme équivalente

$$a' = -f(k_1)a, \qquad a(0) = 1, \tag{3}$$
$$b' = k_1 a, \qquad b(0) = 0, \tag{4}$$

où on a noté $f(k_1) = k_1 + \alpha k_1^{\beta}$.

1.2 Résolution du problème direct

On prend les constantes suivantes :

$k_{10} = 10^6\,\mathrm{h}^{-1}$, $k_{20} = 5.10^{11}\,\mathrm{h}^{-1}$, $E1 = 41840\,\mathrm{J.mol}^{-1}$, $E2 = 83680\,\mathrm{J.mol}^{-1}$, $T = 400\,\mathrm{K}$, $t_f = 1\mathrm{h}$.

1. Recopiez le script MATLAB donné à la page suivante sur Le listing 1, qui calcule la solution de (3)-(4) et sauvez-le sous le nom `main.m` dans un dossier nommé `optTemp` (selon la version de MATLAB, sauvez si nécessaire les fonctions `solution` et `rhs` dans des fichiers séparés, dans le même répertoire) :
2. Ajoutez à ce script les instructions graphiques pour tracer les concentrations $a(t), b(t), c(t)$.

1.3 Optimisation

Avec les données précédentes, on veut maximiser $b(t_f)$ par rapport à T (ce qui revient à maximiser par rapport à k_1), sans contrainte pour l'instant.

1. Ecrivez une fonction `cost(k1,alpha,beta,tf)` calculant la fonction à minimiser en fonction de k_1.
2. Modifiez le script précédent pour calculer la température optimale en utilisant `fminunc`.

```
R = 8.314; k10 = 1e6; k20 = 5e11;
E1 = 41840; E2 = 83680;
beta = E2/E1; alpha = k20/k10^beta;
T = 400;
k1 = k10*exp(-E1/R/T);
tf = 1;

[t,X] = solution(k1,alpha,beta,tf);

function [t,X] = solution(k1,alpha,beta,tf)
  s = ode45(@(t,X) rhs(t,X,k1,alpha,beta),[0 tf],[1 0]);
  t = s.x;
  X = s.y;
end

function out=rhs(t,X,k1,alpha,beta)
  a   = X(1);
  k2  = alpha*k1^beta;
  out = a * [-(k1+k2); k1];
end
```

Listing 1: script `optTemp/main.m`

2 Optimisation d'un profil de température

On s'intéresse maintenant à maximiser $b(t_f)$ par rapport à un profil de température en fonction du temps. Pour cela on découpe l'intervalle $[0,t_f]$ en N sous-intervalles de longueur constante $[t_k,t_{k+1}]$ où $t_k = (k-1)h$, $k = 1 \ldots N$ et $h = t_f/N$ et on considère que k_1 est une fonction constante par morceaux sur ces intervalles (voir la figure 1, pour $N = 5$) : Les seules inconnues sont donc les valeurs de k_1 dans les

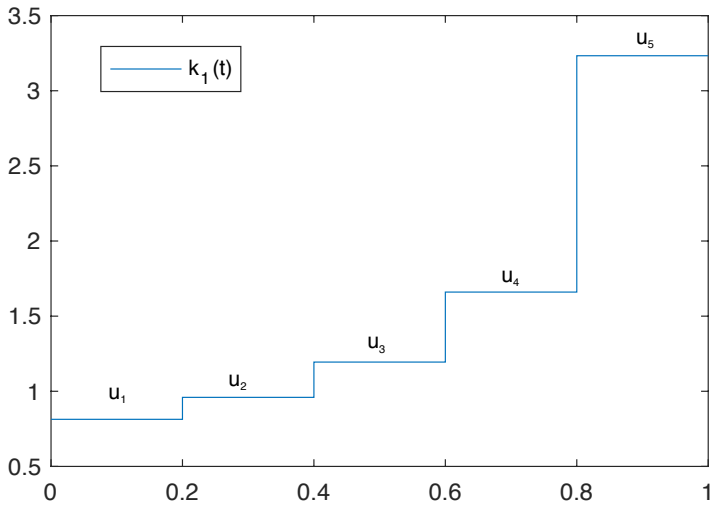

FIGURE 1 – fonction constante par morceaux

intervalles successifs notées ici u_k :

$$\forall t \in [t_k,t_{k+1}], k_1(t) = u_k, k = 1 \ldots N,$$

Le vecteur $u = (u_1, \ldots, u_N)$ est maintenant la nouvelle inconnue de notre problème d'optimisation.

```
R = 8.314; k10 = 1e6; k20 = 5e11;
E1 = 41840; E2 = 83680;
beta = E2/E1; alpha = k20/k10^beta;
T = 400;
N = 5;
tf = 1;
u = k10*exp(-E1/R/T)*ones(N,1);

[t,X] = solution(u,alpha,beta,tf);

function [t,X] = solution(u,alpha,beta,tf)
  N = length(u);
  h = tf/N;
  s = ode45(@(t,X) rhs(t,X,u,1,alpha,beta),[0 h],[1;0]);
  for k=2:length(u)
    s = odextend(s,@(t,X) rhs(t,X,u,k,alpha,beta),k*h);
  end
  t = s.x;
  X = s.y;
end

function out = rhs(t,X,u,k,alpha,beta)
  a = X(1);
  k1 = u(k);
  k2 = alpha*k1^beta;
  out = a * [-(k1+k2); k1];
end
```

Listing 2: script `optTempProfile/main.m`

2.1 Résolution du problème direct

1. Recopiez le script MATLAB donné ci-dessus sur le listing 2, qui calcule la solution de

$$a' = -f(k_1(t))a, \qquad a(0) = 1, \tag{5}$$
$$b' = k_1(t)a, \qquad b(0) = 0, \tag{6}$$

   en enchainant les résolutions sur les intervalles successifs $[t_k, t_{k+1}]$ à l'aide de la fonction `odextend` : ceci est nécessaire à cause des discontinuités de $k_1(t)$ (ici, pour $N = 5$, en t_2, t_3, t_4, t_5). Sauvez-le sous le nom `main.m` dans un dossier nommé `optTempProfile` (selon la version de MATLAB, sauvez si nécessaire les fonctions `solution` et `rhs` dans des fichiers séparés, dans le même répertoire).
2. Ajoutez les instructions graphiques pour représenter les concentrations $a(t), b(t), c(t)$ ainsi que $k_1(t)$ (pour $k_1(t)$ vous pouvez utiliser la fonction `stairs`).

2.1.1 Optimisation

Avec les données précédentes, on veut maximiser $b(t_f)$ par rapport au profil de température, ce qui revient à maximiser par rapport à u, tout en maintenant la température en dessous de $T_{max} = 400\,K$.

1. Ecrivez une fonction `cost(u,alpha,beta,tf)` calculant la fonction à minimiser en fonction de u.
2. Modifiez le script `main.m` pour calculer le profil de température optimal en utilisant `fmincon`.
3. Une fois que votre script donne les résultats attendus, augmentez la valeur de N. Que constatez-vous ?

3 Utilisation de la dérivée exacte pour l'optimisation

3.1 Cas d'une température constante

On considère maintenant a et b comme des fonctions de t et de k_1, et on note

$$S_a(t) = \frac{\partial a}{\partial k_1}(t), S_b(t) = \frac{\partial b}{\partial k_1}(t),$$

les dérivées de a et de b par rapport à k_1. En dérivant formellement (3)-(4) par rapport à k_1, on peut montrer que ces dérivées peuvent être calculées en même temps que $a(t)$ et $b(t)$ en résolvant le système d'équations différentielles

$$a' = -f(k_1)a, \qquad a(0) = 1, \tag{7}$$
$$b' = k_1 a, \qquad b(0) = 0, \tag{8}$$
$$S_a' = -f(k_1)S_a - f'(k_1)a, \qquad S_a(0) = 0, \tag{9}$$
$$S_b' = k_1 S_a + a, \qquad S_b(0) = 0. \tag{10}$$

1. Dupliquez le répertoire `optTemp` sous le nom `optTempWithGrad` et modifiez la fonction `rhs` pour prendre en compte les deux nouvelles équations différentielles (9)-(10)
2. En notant bien que la dérivée de la fonction économique à maximiser est

   $$\frac{\partial b}{\partial k_1}(t_f) = S_b(t_f),$$

   utiliser ce résultat pour accélérer le script `main.m` réalisant l'optimisation de k_1, notamment en utilisant le champ `gradObj` des options passées à la fonction `fminunc`.

3.2 Cas d'un profil de température

Comme précédemment, on considère a et b comme des fonctions de t et du vecteur u, et on note

$$S_{a,i}(t) = \frac{\partial a}{\partial u_i}(t), S_{b,i}(t) = \frac{\partial a}{\partial u_i}(t), i = 1 \dots n,$$

les dérivées partielles de a et de b par rapport à $u_1, \dots, u_N$.

$$\frac{\partial b}{\partial u_k}(t_f), k = 1 \dots N.$$

Si on écrit les équations différentielles (3)-(4) sur l'intervalle $[t_k, t_{k+1}]$ et que l'on ajoute leur dérivées par rapport à u_i, $i = 1 \dots N$, cela donne les $2N+2$ équations différentielles, pour $k = 1 \dots N$,

$$a' = -f(u_k)a, \tag{11}$$
$$b' = u_k a, \tag{12}$$
$$S_{a,i}' = -f(u_k)S_{a,i} - \delta_{ik} f'(u_k)a, \qquad i = 1 \dots N \tag{13}$$
$$S_{b,i}' = u_k S_{a,i} + \delta_{ik} a, \qquad i = 1 \dots N \tag{14}$$

où on a noté $\delta_{ik} = 1$ si $i = k$ et 0 sinon et les conditions initiales sont données par

$$a(0) = 1, b(0) = 0,$$
$$S_{a,i}'(0) = S_{b,i}'(0) = 0, i = 1 \dots n.$$

1. Dupliquez le répertoire `optTempProfile` sous le nom `optTempProfileWithGrad` et modifiez la fonction `rhs` pour prendre en compte les $2N$ nouvelles équations différentielles données par (13)-(14).

2. En notant bien que les dérivées partielles de la fonction économique à maximiser sont égales à

$$\frac{\partial b}{\partial u_i}(t_f) = S_{b,i}(t_f), i = 1 \dots N,$$

utiliser ce résultat pour accélérer le script réalisant l'optimisation de u, notamment en utilisant le champ `gradObj` des options passées à la fonction `fmincon`.

3. Quel facteur d'accélération obtenez vous pour $N = 5$ et pour des valeurs supérieures. Que peut-on en conclure ?

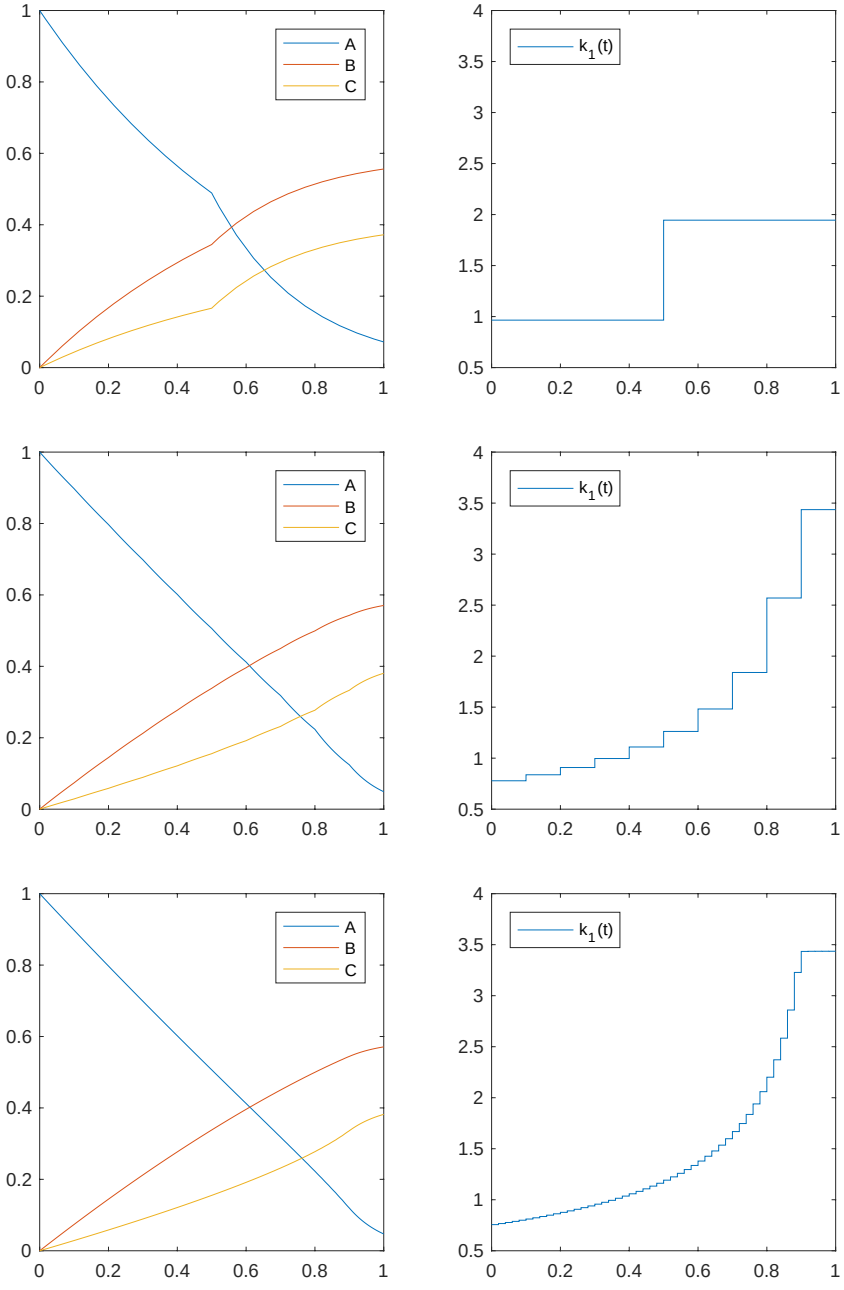

FIGURE 2 – Résultats obtenus pour $N = 2, 10, 50$.

```
R=8.314; k10=1e6; k20=5e11;
E1=10000*4.184; E2=20000*4.184;
beta=E2/E1;
alpha=k20/k10^beta;
T=400;
k1sup=k10*exp(-E1/R/T);

N=10;
tf=1;
lb=zeros(N,1);
ub=ones(N,1)*k1sup;

options = optimoptions('fmincon','display','iter');
options.SpecifyObjectiveGradient=true;

u=fmincon(@(u) cost(u,tf,alpha,beta),ub,[],[],[],[],lb,ub,[],options);

[t,X]=solution(u,tf,alpha,beta);

subplot(1,2,1)
plot(t,X(1:2,:),t,1-X(1,:)-X(2,:))
legend A B C
subplot(1,2,2)
stairs(linspace(0,tf,N+1),[u;u(end)])
legend('k_1(t)','location','northwest')

function [f,g]=cost(u,tf,alpha,beta)
  [~,X]=solution(u,tf,alpha,beta);
  f=-X(2,end);
  g=-X(4:2:end,end);
end

function [t,X]=solution(u,tf,alpha,beta)
  N=length(u);
  h=tf/N;
  s=ode45(@(t,X) rhs(t,X,u,1,alpha,beta),[0 h],[1;0;zeros(2*N,1)]);
  for k=2:length(u)
    s=odextend(s,@(t,X) rhs(t,X,u,k,alpha,beta),k*h);
  end
  t=s.x;
  X=s.y;
end

function out=rhs(~,X,u,k,alpha,beta)
  N=length(u);
  out=zeros(2*N+2,1);
  a=X(1);
  Sa=X(3:2:end);
  k1=u(k);
  k2=alpha*k1^beta;
  out(1:2) = a*[-(k1+k2); k1];
  out(3:end) = kron(Sa,[-(k1+k2); k1]);
  out(2*k+1:2*k+2) = out(2*k+1:2*k+2)+[-(1+alpha*beta*k1^(beta-1))*a;a];
end
```

Listing 3: Solution `optTempProfileWithGrad/main.m`