Commande Optimale d'un convertisseur de puissance Buck-Boost

Compte Rendu

I - Introduction.

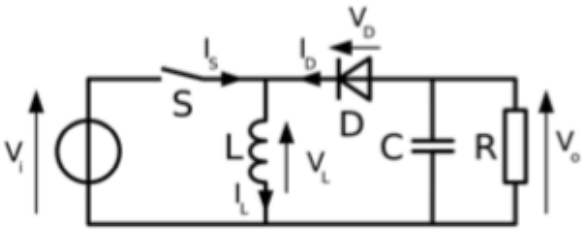

Figure 1 : Schéma électrique du convertisseur Buck-Boost.

Un convertisseur de puissance continu-continu est un dispositif permettant d'adapter la tension délivrée à une charge à partir d'une tension d'entrée constante. On considère un convertisseur Buck-Boost (abaisseur - survolteur). La charge en sortie est assimilée à une résistance R.

Le pilotage de ce dispositif s'effectue par l'ouverture et la fermeture à fréquence élevée de l'interrupteur S. Le système a donc 2 configurations de fonctionnement :

- Lorsque $s = 1$, l'interrupteur est fermée, l'énergie est stocké dans l'inductance et la capacité se décharge dans la résistance.
- Lorsque $s = 0$, l'interrupteur est ouvert, l'inductance est reliée à la charge et à la capacité. Il en résulte un transfert de l'énergie accumulée dans l'inductance vers la capacité et la charge.

En considérant les variables d'états : $x_1 = i_L$ le courant dans l'inductance et $x_2 = v_0$ la tension dans la capacité et en notant v_i la tension d'entrée supposée constante, les équations de ce modèle non linéaire peuvent alors s'écrire $x = (x_1, x_2)$:

$\dot{x} = s(A_1 x + B_1 v_i) + (1 - s)(A_2 x + B_2 v_i))$ avec $s \in [0; 1]$ la variable de commande et :

$$A_1 = \begin{pmatrix} 0 & 0 \\ 0 & -1/RC \end{pmatrix} \qquad A_2 = \begin{pmatrix} 0 & -1/L \\ 1/C & -1/RC \end{pmatrix}$$

$$B_1 = \begin{pmatrix} 1/L \\ 0 \end{pmatrix} \qquad B_2 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

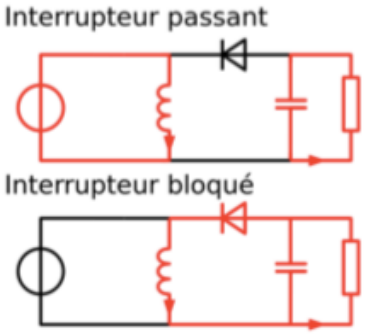

Figure 2 : Fonctionnement du convertisseur.

Paramètres : $V_i = 10V, R = 2\Omega, L = 0.3H, C = 0.1F$

II - Analyse du fonctionnement.

D'après le sujet on effectue un pilotage du circuit avec une MLI (modulation de largeur d'impulsion). Pour une période de découpage T_s fixée et petite devant les constantes de temps du circuit, la commande est déterminée par un rapport cyclique $\alpha(t) \in [0;1]$ suivant la loi :

$$s = \begin{cases} 1 & si \quad k\,.\,T_s \le t \le k\,.\,T_s + \alpha\,.\,T_s \\ 0 & si \quad k\,.\,T_s + \alpha\,.\,T_s < t \le (k+1)\,.\,T_s \end{cases}$$

Un simple calcul montre que la valeur moyenne de $s(t)$ sur la période T_s est égale à $\alpha(t) \in [0;1]$. On voit donc que la commande s peut être assimilée à une variable continue comprise entre les valeurs 0 et 1 si T_s est choisi suffisamment petit. Par la suite, on peut donc relaxer le domaine de commande $\{0;1\}$ à l'intervalle $[0;1]$ et simplifier le problème par la recherche une commande $s(t) \in [0;1]$.

Il faut donc à ce moment calculer la valeur de s qui nous permet de délivrer une tension de sortie de 15 volts.

En régime permanent, on a : $\dot{x} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$:

Donc : $\dot{x} = s\,.\left[\begin{pmatrix} 0 & 0 \\ 0 & -1/RC \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 1/2 \\ 0 \end{pmatrix}\right] + (1-s)\left[\begin{pmatrix} 0 & -1/L \\ 1/C & -1/RC \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right]$

En isolant s, on trouve : $s_{ref} = \dfrac{\frac{V_0}{L}}{\frac{v_i}{L} + \frac{v_0}{L}} = 0.6$ et $i_{L_{ref}} = 1.875$ A

Ainsi : $x_{ref} = \begin{pmatrix} i_{L_{ref}} \\ v_{0_{ref}} \end{pmatrix} = \begin{pmatrix} 1.875 \\ 15 \end{pmatrix}$

III -Algorithme FBS (Forward Bachward Sweep).

Après avoir obtenu les valeurs cherchées à cet état d'équilibre, on veut minimiser le critère suivant à l'état initiale $x_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$:

$$J = \int_0^T (x - x_{ref})^T Q (x - x_{ref}) + \rho (s - s_{ref})^2 dt + \frac{1}{2}(x(T) - x_{ref})^T Q_f (x(T) - x_{ref})$$

Les réglages sont les suivants : $Q_f = Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$; $\rho = 1$; $T = 5$

Pour cela, on va utiliser l'algorithme FBS qui est un « balayage avant-arrière » basé sur les conditions nécessaires du principe du minimum permettant de résoudre des problèmes du type :

$$\begin{aligned} \dot{x} &= f(x,u) \\ x(0) &= x_0 \\ \min_u J(u) &= \int_0^T L(x,u)dt + \phi(x(T)) \end{aligned}$$

A partir d'une commande u donnée et quelconque ($u(t) \in [0;1]$) :

- Intégration sur $[0;T]$ du sytème : $\dot{x} = f(x,u)$ et $x(0) = x_0$
- Application de la condition de transversalité : $\lambda(T) = \frac{d\phi}{dx}(x(T))$
- Intégration sur $[0;T]$ du système adjoint : $\dot{\lambda} = -\frac{dH(x,u,\lambda)}{dx}$
- Calcul de $v = \arg\min_u H(x,u,\lambda)$
- Mise à jour de u : $v = \arg\min_{u_k} J\{(u_k), k = 0,1,2,...\}$ avec $u_k = \prod(u + \beta^k(v-u))$ où $\prod(x)$ désigne la projection sur $[0;1]$ et $\beta \in (0;1)$. De plus, $\prod(x) = \max(\min(1,x),0)$
- Si le text est OK, alors on termine sinon on retourne au point 1.

On a donc au départ : $\begin{cases} \dot{x} = f(x,u) \\ x(0) = x_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ J = \int_0^T L(x,u)dt + \phi(x(T)) \\ H = \lambda^T f(x,u) + L(x,u) \end{cases}$

Ainsi on écrit notre Hamiltonien H : $H = \lambda^T[s(A_1x + B_1v_i) + (1-s)(A_2x + B_2v_i)] + L(x,s)$

et $\dot{x} = [s(A_1x + B_1v_i) + (1-s)(A_2x + B_2v_i)]$.

On discrétise notre problème :

$h = T/N$ et $\begin{cases} x_{k+1} = x_k = h(x_k, h_k) \\ x_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \\ \lambda_k = \lambda_{k+1} - h[s(A_1 - A_2)^\lambda + A_2^T\lambda + Q(x - x_{ref})] \end{cases}$

De plus : $\frac{dH}{ds} = h[(A_1x + B_1v_i) - (A_2x + B_2v_i)] + \rho(s - s_{ref})$.

Pour commencer l'algorithme, on définie toutes nos constantes.

```
global R Vi L C beta kmax Sref T N h t A1 A2 Q Xref rho B1 B2

R=20;
Vi=10;
L=0.3;
C=0.1;
beta=0.5;
kmax=25;
Sref=0.6;
T=5;
N=100;
h=T/N;
t=linspace(0,T,N);
rho=1;
epsilon=1E-6;

A1=[0,0;0,-1/(R*C)];
A2=[0,-1/L ; 1/C,-1/(R*C)];

B1=[1/L;0];
B2=[0;0];

Q=eye(2);

Xref=[1.875;15];


Sstock=[];
Xstock=[];
Jstock=[];
```

Figure 3 : Définition des constantes.

Ensuite on procède à l'initialisation de notre vecteur position et de notre commande.

```
%% Initialisation

S=Sref*ones(1,N);
Sstock=[Sstock;S];

X=discretX(S);
Xstock=[Xstock;X];


Xstock=[Xstock;X];
Sstock=[Sstock;S];

Jold=0;
p=1;
test=1;
```

Figure 4 : Initialisation de x et s.

On fait une boucle *while* pour réaliser toute nos itérations, ici on fixe ce nombre à *p =100*. Cela nous permet de trouver la commande optimal en minimisant le coût *J*.

```
while test && (p<100)

X=discretX(S);
lbd=discretLambda(X,S);
V=discretV(X,lbd);

[X,S,Jmin]=minimisation(X,S,V);

Xstock=[Xstock;X];
Sstock=[Sstock;S];
Jstock=[Jstock;Jmin];

figure(1)
hold on
plot(t,Xmin(1,:),t,Xmin(2,:))
figure(2)
hold on
plot(t,Smin)
figure(3)
hold on
plot(p,Jmin,'+b')

test=abs(Jold-Jmin)>1.e-3;
Jold=Jmin;
p=p+1;

end
```

Figure 5 : Parcours de toutes nos itérations pour trouver le J optimal.

En traçant, la commande s et l'état x_1 et x_2 en fonction du temps à chaque itérations, on obtient :

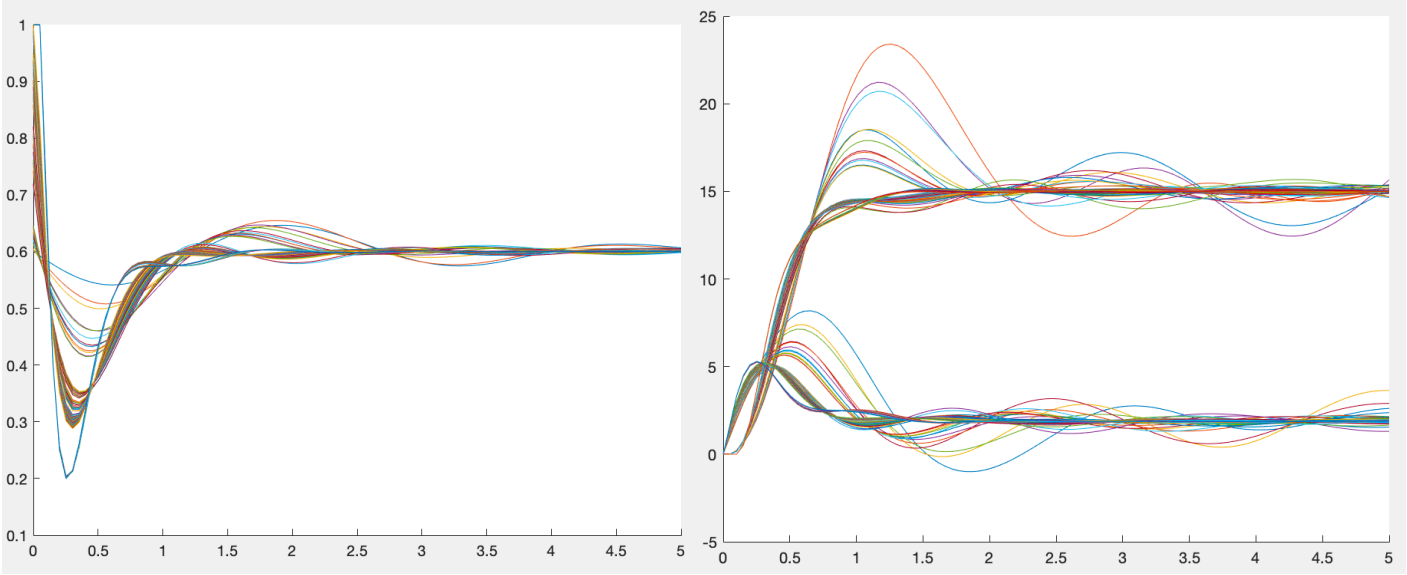

Figure 6 : Les courbes de commande (à gauche) et les courbes d'état (à droite) en fonction du temps à chaque itérations

Une fois que l'on sort de cette boucle, soit on atteint :

$|J_{k+1} - J_k| < \epsilon = 10^{-3}$ ou alors nous avons atteint la valeur maximal d'itérations.

Ainsi on en déduit que la commande optimale pour piloter notre système afin d'atteindre x_{ref} tout en minimisant le coût J.

On peut alors tracer les courbes de la commande optimale et de l'évolution de l'état en fonction du temps pour la commande optimale. Ainsi que, l'évolution (décroissante) du coût J, qui correspond bien à la minimisation de ce dernier.

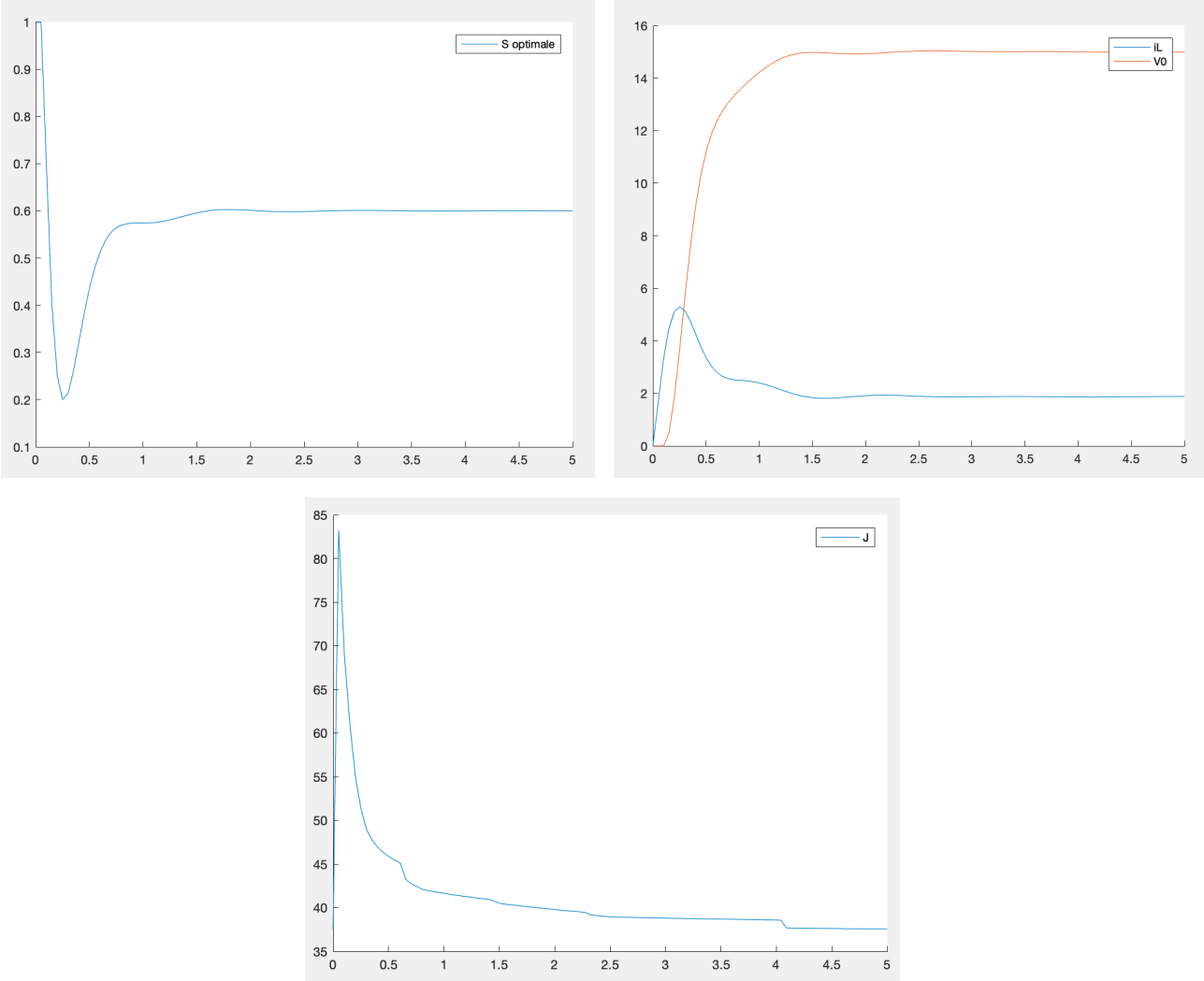

Figure 7 : La courbe de commande optimale, les courbes d'état et l'évolution de J en fonction du temps

IV -Conclusion.

Suite à ce TP, nous avons réussi à minimiser un coût J en trouvant un état et une commande optimale grâce à l'algorithme du FBS. Pour cela nous avons dû tout d'abord trouver la valeur de s et ainsi en déduire notre état en régime permanent : $x_{ref} = \begin{pmatrix} i_{L_{ref}} \\ v_{0_{ref}} \end{pmatrix} = \begin{pmatrix} 1.875 \\ 15 \end{pmatrix}$. Puis après avoir discrétiser notre problème, nous avons utiliser le FBS et ainsi obtenu la commande dont le coût est minimal. Enfin, lorsque notre ρ correspondant à la pondération est grand, nous pouvons alors remarquer que le coût est plus important mais que la commande optimale reste sensiblement identique.

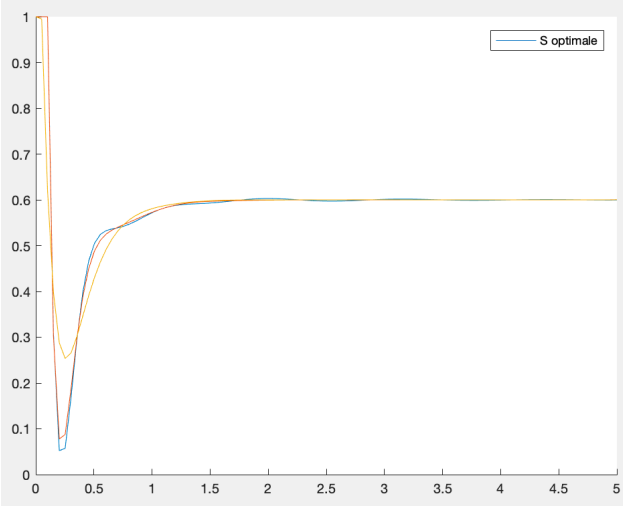

Figure 7 bis : Courbe de commande en fonction de rho (bleu: 0.1, orange: 1 et jaune: 100)

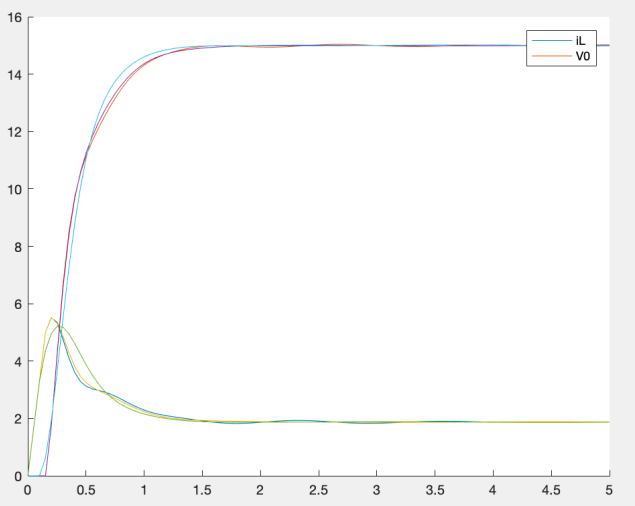

Figure 7 ter : Courbe de l'état en fonction de rho

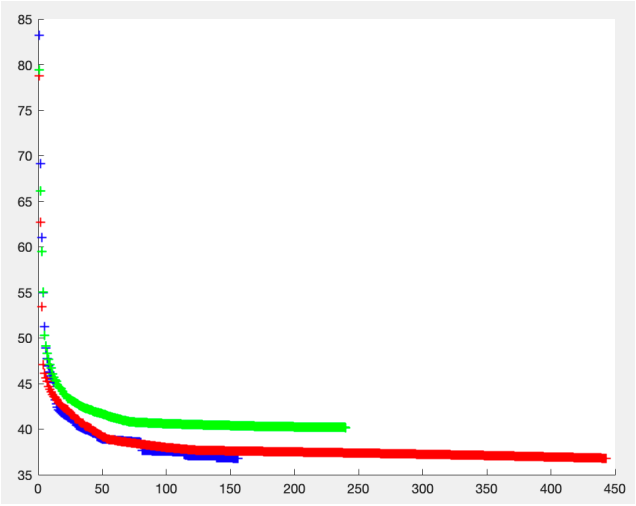

Figure 7 quater : Courbe du coût en fonction de rho (vert: rho=100, rouge: rho=0.1 et bleu: rho=1)

V - ANNEXES

```
function [X] = discretX(S)

global N h Vi A1 A2 B1 B2

X=zeros(2,N);
X(:,1)=[0;0];

for k=1:N-1

   X(:,k+1) = X(:,k) + h*(S(k)*(A1*X(:,k)+B1*Vi)+(1-S(k))*(A2*X(:,k)+B2*Vi));

end

end
```

Figure 8 : Fonction Matlab pour calculer X

```
function [lbd] = discretLambda(X,S)

global Q h A1 A2 Xref N

lbd=zeros(2,N);

lbd(:,N)=Q*(X(:,N)-Xref);

for k=1:N-1

    lbd(:,N-k)=lbd(:,N-k+1) + h*(S(N-k+1)*(A1-A2)'*lbd(:,N-k+1)+A2'*lbd(:,N-k+1)+Q*(X(:,N-k+1)-Xref));

end

end
```

Figure 9 : Fonction Matlab pour calculer lambda

```
function [V] = discretV(X,lbd)

global N A1 A2 B1 B2 Vi rho Sref

V=zeros(1,N);

for k=1:N

    V(k)=Sref+(lbd(:,k)'*((A2-A1)*X(:,k)+(B2-B1)*Vi))/rho;

end

end
```

Figure 10 : Fonction Matlab pour calculer v

```
function [J] = discretJ(X,S)

global N h Xref Q rho Sref

J=0;

for k=1:N

    J=J+(h/2)*((X(:,k)-Xref)'*Q*(X(:,k)-Xref)+rho*(S(k)-Sref)^2);

end

J=J+(1/2)*(X(:,N)-Xref)'*Q*(X(:,N)-Xref);

end
```

Figure 11 : Fonction Matlab pour calculer J

```
function [Xmin,Smin,Jmin] = minimisation(X,S,V)

global N kmax beta

Sp=zeros(kmax,N);
Xp=zeros(2*kmax+2,N);
Jp=zeros(1,kmax);


Xp(1:2,:)=X;
Jp(1)=discretJ(X,S);

for k=1:kmax

    Sp(k,:)=max(min(S+(beta^(k))*(V-S),1),0);
    Xp(2*(k-1)+1:2*k,:)=discretX(Sp(k,:));
    Jp(1,k)=discretJ(Xp(2*k-1:2*k,:),Sp(k,:));

end

[minJ,indiceMin]=min(Jp);
Jmin=minJ;
Smin=Sp(indiceMin,:);
Xmin=Xp(2*indiceMin-1:2*indiceMin,:);

end
```

Figure 12 : Fonction de minimisation de J en fonction des commandes et des état calculés au instant k