BE Optimisation Structurale 2A - Calcul Scientifique

Stéphane Grihon (EADS)
assistés de Antoine Merval (Transiciel), Eric Juliani(ONERA) Manuel Samuelides (SUPAERO)

18 et 21 Mars 2005

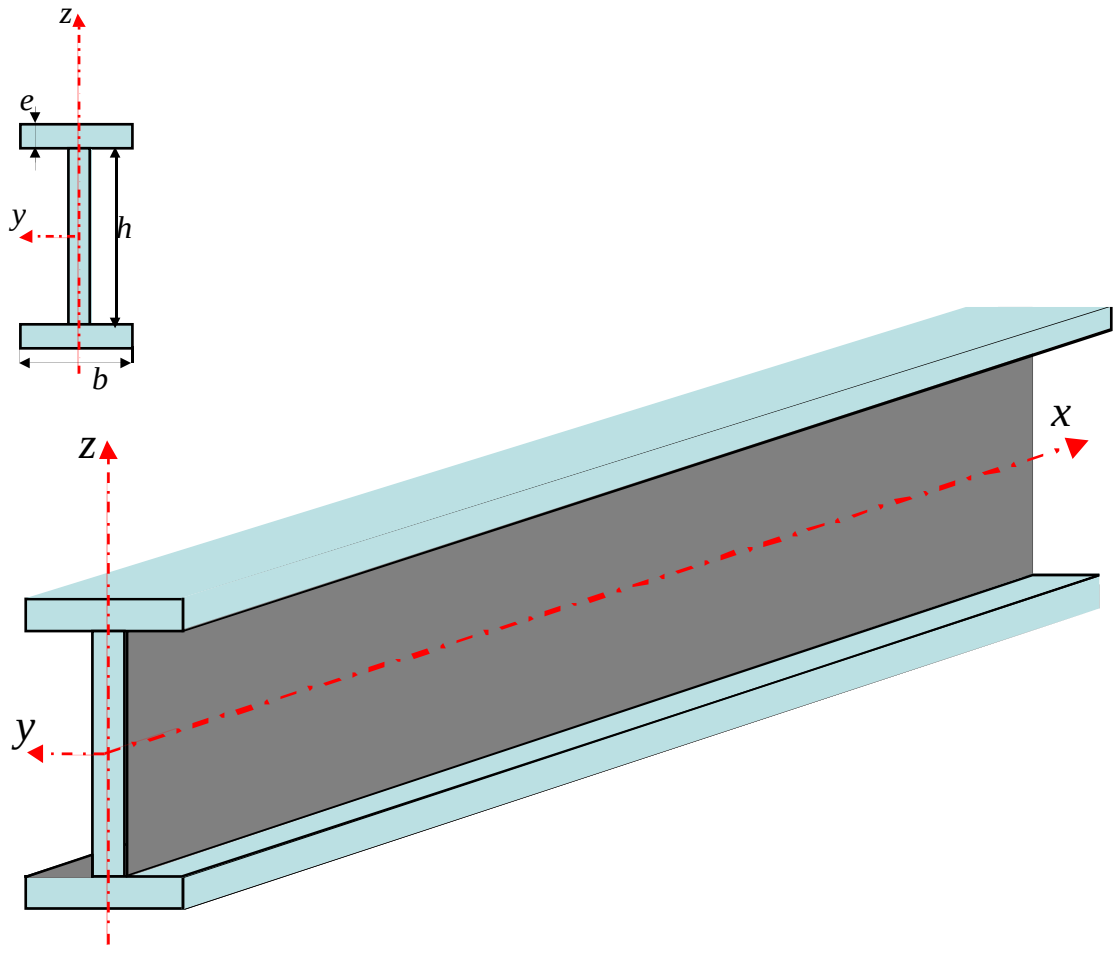

FIG. 1: Schéma d'un raidisseur en I

L'objectif est de traiter le problème de dimensionnement de l'extrados d'un caisson de voilure de gros porteur. Il s'agit de donner dans cette phase de dimensionnement détaillée une forme précise aux raidisseurs de ce panneau. Bien entendu, pour un jeu defforts donné, il existe une infinité de solutions au problème : il suffit de "sur-dimensionner" les raidisseurs pour tenir la charge imposée. Mais il s'agit de rechercher parmi les raidisseurs "admissibles" celui de masse minimale.

Pour les besoins du calcul, le panneau est découpé selon son "motif" périodique : le "super-raidisseur". Il s'agit du raidisseur auquel on ajoute la peau sur les demi-pas adjacents. Mais pour simplifier le processus

de dimensionnement dans ce bureau d'études, on ne considèrera pas la peau et on se concentrera ici sur l'optimisation d'un raidisseur (l'extension se ferait selon le même principe).
On se placera dans les hypothèses simplificatrices suivantes :

H1- section en I symétrique (talon et semelle identiques).

H2- épaisseur de profil uniforme

H3- hypothèse de profil mince pour le calcul de l'inertie

Voir figures ci-avant.

1. La contrainte de **flambage global** est que la charge extrème $P_{\max}$ n'excède pas la **charge d'Euler** donnée par la formule

$$P_c = \frac{\pi^2.E.I}{L^2}$$

où

- E est le module de Young du matériau considéré
- I est le moment d'inertie de la section du raidisseur par rapport l'axe Oy où O est le centre de gravité de la section du raidisseur,
- L est la longueur du raidisseur [1]

Montrer que cette contrainte équivaut à une contrainte de borne sur I

$$\textbf{(FG)} : I \geq I_{\min}$$

où l'on explicitera $I_{\min}$.
2. Calculer I en fonction des dimensions (e, h, b).
Exprimer I en fonction des nouvelles variables (e, S_a, S_t) où S_a est la surface de l'âme et S_t la surface du talon.
3. On cherche la section de raidisseur de surface minimale pour minimiser le poids de la structure. Ecrire le problème d'optimisation (problème P1) correspondant dans les variables (e, h, b) et la formulation équivalente dans les variables (e, S_a, S_t).
4. Soit une configuration admissible telle que la contrainte (FG) ne soit pas saturée. Montrer qu'il existe une configuration admissible de section inférieure. En déduire que s'il existe une solution au problème P1, la contrainte (FG) est saturée pour la configuration solution.
5. En déduire qu'il n'existe pas de solution physique réaliste au problème (P1).
6. On se propose de rétablir le réalisme en introduisant une seconde contrainte de borne

$$\textbf{(EM)} : e \geq e_{\min}$$

On note (P2) le problème (P1) augmenté de la contrainte (EM). Montrer que si (P2) a une solution,la contrainte (EM) est saturée à cette solution. Ecrire un système équivalent portant sur les variables (S_a, e) dont on montrera que les contraintes sont saturées.
Indication : On éliminera S_t sans oublier la contrainte de positivité pesant sur S_t

[1] on considérera que les raidisseurs s'appuient sur les nervures, l'appui est simple et le pas inter-nervures est L.

On montrera alors que la solution du problème (P2) est :

$$\begin{cases} e = e_{\min} \\ h = \left[\frac{12 I_{\min}}{e_{\min}}\right]^{\frac{1}{3}} \\ b = 0 \end{cases}$$

7. Vérifier que cette solution satisfait le théorème de Kuhn-Tucker et notament que les paramètres KKT sont positifs ou nuls
8. Montrer que en cas de charge élevée le rapport $\frac{e}{h}$ baisse dangereusement. Que peut-il se passer dans ces conditions ? La contrainte de flambage local, (**FL**) est

$$\frac{P_{\max}}{S} \leq K \left(\frac{e}{h}\right)^2$$

où K est une constante qui dépend aussi de la géométrie du talon. On note (P3) le problème consistant à minimiser S sous les contraintes (FG) et (FL) et les contraintes de positivité. On pose $\alpha = \frac{e}{h}$ (variable d'aspect). Ecrire le problème (P3) dans le système de variables (α, S_a, S_t).
9. Chercher par l'application du théorème de Kuhn-Tucker l'expression d'une solution possible en ne saturant pas les contraintes de positivité.
10. Le théorème de Kuhn-Tucker ne donne que des conditions nécessaires que doit vérifier une solution. Montrer que le domaine admissible en (α, S_a, S_t) est fermé et que la fonction objectif est à niveaux compacts. En déduire qu'il existe une solution au problème (P3) et que par conséquent la configuration trouvée à la question précédente est bien solution.
11. Comparer avec les performances en objectif pour le problème (P3) du raidisseur optimal en I et du raidisseur âme mince optimal.

TP matlab

Calculer (si possible) par une méthode approchée la configuration optimale du problème (P3) pour les valeurs numériques suivantes. Comparer avec la solution analytique

$$\begin{cases} \text{Pas inter-nervure :} & L = 503\text{mm} \\ \text{Module de Young :} & E = 75000\text{Mpa} \\ \text{Coefficient de Poisson :} & \nu = 0,33 \\ \text{Constante de flambage local :} & K = \frac{4\pi^2 E}{12(1-\nu^2)} \\ \text{Charge extrême :} & P_{\max} \in \{10^5, 2.10^5, ..., 9.10^5\} \end{cases}$$

Les logiciels matlab suivants sont fournis aux étudiants sous "distrib" :

- **optistrucvf** : Ce logiciel dispose les calculs, présente les résultats Il faut toutefois régler les options du programme de minimisation **fmincon** (voir le help de matlab)
- **ContraintesNL** et **Surface** : Ces fonctions ont le format correct pour pouvoir être appelées par "fmincon". Elles sont ocmplètes pour faire gagner du temps aux étudiants
- **FlanbGlob** et **Flambloc** : Ces fonctions renvoyant les contraintes et leurs gradients sont à programmer par les étudiants.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  OptiStruc
%
%   Ce script complet met en place le calcul du BE Optistruc
%   Il teste un etat initial pour voir s'il est admissible et effectue
%   la procedure d'optimisation pour une serie de cas de charge
%
%   Variables d'entr?e du script
%   ----------------------------
%   L=503  mm       pas internervure= longueur du raidisseur
%   E=75000 mPa     module de Young
%   nu=0.33         coefficient de Poisson
%   K=K = 4*pi^2*E/(12*(1-nu^2)) coefficient de flambage local
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           FONCTIONS UTILISEES
%           %%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  function [S, GradS] = Surface(ehb)
%
%   Ce script incomplet calcule la surface du raidisseur et son gradient
%   Les etudiants s'attacheront a comprendre le fonctionnement
%   de fmincon et fixeront les options et les bornes de l'optimisation
%
%   Variables d'entree
%   ------------------
%   Etat du raidisseur ehb (vecteur colonne (1,3)
%   ehb(1)  ?paisseur
%   ehb(2)  hauteur de l'me
%   ehb(3)  largeur de la semelle et du talon
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%function[ContFL,PasDeContEg,gradContFL,PasDeGradContEg]=ContraintesNL(ehb)
%
%   Cette fonction renvoie les contraintes pour fmincon
% ( On n'a pas de contraintes egalite dans notre cas )
%
%   Variables d'entree
%   --------------------
%   ehb = (e;h;b)
%
%   Variables de sortie
%   ---------------------
%   ContraintesIn = [ CFG ;CFL ]
%   GradContraintesIn = [d(CFG)/de   d(CFL)/de ;
%                        d(CFG)/dh    d(CFL)/dh ;
%                        d(CFG)/db    d(CFL)/db ]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%  function [CFG, GradCFG] = FlambGlob(Inertiemin,ehb)
%
%   Ce script calcule la contrainte de flambage global sous la forme CFG0
%   et son gradient
%
%   Variables d'entree
%   ------------------
%   Cas de charge:
%   Inertiemin
%   Etat du raidisseur ehb (vecteur colonne (1,3):
%   ehb(1)  epaisseur
%   ehb(2)  hauteur de l'me
%   ehb(3)  largeur de la semelle et du talon
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  function [CFL, GradCFL] = FlambLoc(K,Pmax,ehb)
%
%   Ce script calcule la contrainte de flambage local sous la forme CFL0
%   et son gradient
%
%   Variables d'entree
%   ------------------
%   Constantes:
%   K       constante de flambage local
%   Cas de charge:
%   Pmax
%   Etat du raidisseur ehb (vecteur colonne (1,3):
%   ehb(1)  epaisseur
%   ehb(2)  hauteur de l'me
%   ehb(3)  largeur de la semelle et du talon
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

e = ehb(1);
h = ehb(2);
b = ehb(3);

S=e*h+2*e*b;
CFL=Pmax-K*S*(e/h)^2;
if nargout>1
    grad1=[3*e^2*(h+2*b)/h^2;e^3*(-1/h^2-4*b/h^3);2*e^3/h^2];
    GradCFL=-K*grad1;
end

clear all
close all

global L;
```

```
global E;
global nu;
global K;
global Pmax;
global Inertiemin;


L = 503;                          % pas internervure= longueur du raidisseur
E = 75000;                        % module de Young
nu = 0.33;                        % coefficient de Poisson
K = 4*pi^2*E/(12*(1-nu^2));       % coefficient de flambage local

n=20;                             % nombre de cas de charge ?tudi?s
Chargemax=(1:n)*50000;
eSQP=zeros(1,n);hSQP=eSQP;bSQP=eSQP;SSQP=eSQP;
eKKT=zeros(1,n);hKKT=eKKT;bKKT=eKKT;SKKT=eKKT;
disp('************************************************************************');
disp('********************** Debut du calcul *********************************');

EtatInit=[10;80;20];
disp(['Raidisseur initial:     ','epaisseur ' num2str(EtatInit(1)) ', ...
      hauteur ' num2str(EtatInit(2)) 'largeur ' num2str(EtatInit(3))]);

[S, GradS] = Surface(EtatInit);
disp(['Surface: ',num2str(S)]);
disp(' ');
% Bornes de l'optimisation
lbnd=?????????????;
ubnd=?????????????;
% Options de l'optimisation
options = optimset('Display','?????','GradConstr','??','GradObj','??',...
    'LargeScale','???','MaxIter',xxxx,'MaxFunEvals',xxx,...
    'TolCon',xxx,'TolFun',xxx,'TolX',xxx);

for k=1:n

disp('************************************************************************');
    % Cas de charge
    %%%%%%%%%%%%%
    Pmax=Chargemax(k);                    % Cas de charge
    Inertiemin=Pmax*L^2/(pi^2*E);

    disp(' ');
    disp(['Cas de charge: Pmax=' num2str(Pmax)]);
    disp(' ');
```

```
    % Optimisation numerique
    %%%%%%%%%%%%%%%%%%%%%%%
[BestEtat,SurfaceOpt,exitflag,output,KuhnTucker]=fmincon(@Surface,EtatInit,...
        [],[],[],[],lbnd,ubnd,@ContraintesNL,options);
    [S, GradS] = Surface(BestEtat);
    [CFG, GradCFG] = FlambGlob(Inertiemin,BestEtat);
    [CFL, GradCFL] = FlambLoc(K,Pmax,BestEtat);
disp(['Raidisseur optimise par le calcul:  ','epaisseur',num2str(BestEtat(1)) ',.
 hauteur ' num2str(BestEtat(2)) ' largeur' num2str(BestEtat(3))]);
    disp(['Surface: ',num2str(S),'  Flambage global: ',num2str(CFG),...
'  Flambage local: ',num2str(CFL)]);
    disp(' ');
    eSQP(k)=BestEtat(1);
    hSQP(k)=BestEtat(2);
    bSQP(k)=BestEtat(3);
    SSQP(k)=S;

    % Optimisation Kuhn-Tucker
    %%%%%%%%%%%%%%%%%%%%%%%%%
    alpha5=3*Pmax^2/(32*K^2*Inertiemin);
    alpha=alpha5^.2;
    e4=6*Inertiemin*alpha^3;
    e=e4^.25;
    h=e/alpha;
    b=h/6;
    EtatKKT=[e;h;b];
    [S, GradS] = Surface(EtatKKT);
    [CFG, GradCFG] = FlambGlob(Inertiemin,EtatKKT);
    [CFL, GradCFL] = FlambLoc(K,Pmax,EtatKKT);
    disp(['Raidisseur optimise par Kuhn-Tucker:     ',...
        'epaisseur ' num2str(EtatKKT(1)) ', hauteur ' num2str(EtatKKT(2)) ....
' largeur ' num2str(EtatKKT(3))]);
    disp(['Surface: ',num2str(S),'  Flambage global: ',num2str(CFG),...
          '  Flambage local: ',num2str(CFL)]);
    eKKT(k)=e;
    hKKT(k)=h;
    bKKT(k)=b;
    SKKT(k)=S;
    disp(' ');
    disp('*****************************************************************');
    disp(' ');
end
     disp(' ');
     disp('*****************************************************************');
     disp('******************           Fin du calcul       **************');
     disp('*****************************************************************');
```

```
     disp(' ');

% Representation graphique des resultats
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(1);
plot(Chargemax,eKKT,'k-',Chargemax,eSQP,'r--');
legend('Calcul analytique', 'Optimisation approchee')
title('Epaisseur en fonction de la charge');
figure(2);
plot(Chargemax,hKKT,'k-',Chargemax,hSQP,'r--');
legend('Calcul analytique', 'Optimisation approchee')
title('Hauteur en fonction de la charge');
figure(3);
plot(Chargemax,bKKT,'k-',Chargemax,bSQP,'r--');
legend('Calcul analytique', 'Optimisation approchee')
title('Largeur semelle en fonction de la charge');
figure(4);
plot(Chargemax,SKKT,'k-',Chargemax,SSQP,'r--');
legend('Calcul analytique','Optimisation approchee');
title('Surface en fonction de la charge');


function [ContraintesIn, NoEq, GradContraintesIn, NoGradEq] = ContraintesNL(ehb)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%function [ContFL, PasDeContEg, gradContFL, PasDeGradContEg] = ContraintesNL(ehb)
%
%   Cette fonction complete renvoie les contraintes pour fmincon
% ( On n'a pas de contraintes egalite dans notre cas )
%
%   Variables d'entree
%   --------------------
%   ehb = (e;h;b)
%
%   Variables de sortie
%   ---------------------
%   ContraintesIn = [ CFG ;CFL ]
%   GradContraintesIn = [d(CFG)/de   d(CFL)/de ;
%                        d(CFG)/dh    d(CFL)/dh ;
%                        d(CFG)/db    d(CFL)/db ]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global L E nu K Pmax Inertiemin;

NoEq = 0;
NoGradEq = zeros(3,1);
```

```
ContraintesIn = zeros(2,1);
GradContraintesIn = zeros(3,2);

[ContraintesIn(1),GradContraintesIn(:,1)] = FlambGlob(Inertiemin,ehb);
[ContraintesIn(2),GradContraintesIn(:,2)] = FlambLoc(K,Pmax,ehb);


function [S, GradS] = Surface(ehb)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  function [S, GradS] = Surface(ehb)
%
%   Cette fonction complete calcule la surface du raidisseur et son gradient
%
%   Variables d'entree
%   ------------------
%   Etat du raidisseur ehb (vecteur colonne (1,3)
%   ehb(1)  epaisseur
%   ehb(2)  hauteur de l'me
%   ehb(3)  largeur de la semelle et du talon
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

e = ehb(1);
h = ehb(2);
b = ehb(3);

S=e*h+2*e*b;
if nargout>1
    GradS=[h+2*b;e;2*b];
end


function [CFG, GradCFG] = FlambGlob(Inertiemin,ehb)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  function [CFG, GradCFG] = FlambGlob(Inertiemin,ehb)
%
%   Cette fonction incomplete calcule la contrainte de flambage global CFG
%   sous la forme CFG0 et son gradient.
%   Les etudiants programmeront le calcul de la contrainte et de son
%   gradient
%
%   Variables d'entree
%   ------------------
%   Cas de charge:
```

```
%   Inertiemin
%   Etat du raidisseur ehb (vecteur colonne (1,3):
%   ehb(1)  epaisseur
%   ehb(2)  hauteur de l'me
%   ehb(3)  largeur de la semelle et du talon
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

e = ehb(1);
h = ehb(2);
b = ehb(3);


function [CFL,GradCFL] = FlambLoc(K,Pmax,ehb)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  function [CFL, GradCFL] = FlambLoc(K,Pmax,ehb)
%
%   Cette fonction  incomplete calcule la contrainte de flambage global CFL
%   sous la forme CFL0 et son gradient.
%   Les etudiants programmeront le calcul de la contrainte et de son
%   gradient
%
%
%   Variables d'entree
%   ------------------
%   Constantes:
%   K       constante de flambage local
%   Cas de charge:
%   Pmax
%   Etat du raidisseur ehb (vecteur colonne (1,3):
%   ehb(1)  epaisseur
%   ehb(2)  hauteur de l'me
%   ehb(3)  largeur de la semelle et du talon
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

e = ehb(1);
h = ehb(2);
b = ehb(3);
```