Etude d'une suspension à câbles

L'ONERA étudie une suspension active pour des essais de maquette d'avion en soufflerie. La maquette est tenue par 9 câbles dont les tensions sont asservies par des moteurs électriques.

La figure montre deux configurations géométriques qui ont été étudiées.

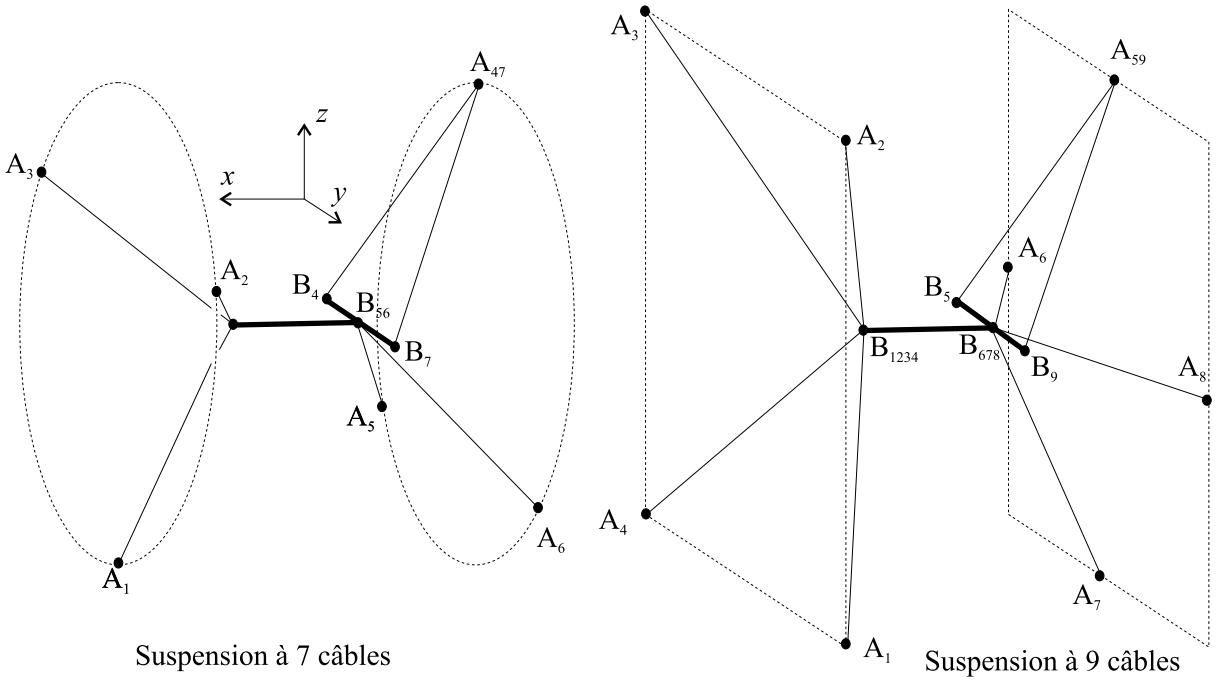

FIG. 1 – Les deux configurations de suspension

Dans le cadre de ce bureau d'étude, nous nous limiterons à l'analyse de la suspension à 7 câbles. La maquette est fixée à un support en forme de T dont la pointe avant est au voisinage du centre de gravité de la maquette et dont la barre du T est située à l'arrière de la maquette. Le fichiers `inidata1.m` donne les caractéristiques géométriques de la configuration à $n_q = 7$ câbles, avec en particulier les composantes dans le repère fixe $\mathcal{R}_A$ des points d'ancrages fixes A_i, et les composantes dans le repère maquette $\mathcal{R}_B$ des points d'ancrages B_i.

OUTILS D'ANALYSE DE LA SUSPENSION

1) Modèles géométrique et cinématique analytiques.

Le modèle géométrique analytique $\mathbf{q} = H(D_{AB})$ permet de calculer les longueurs des câbles $q_i = \|A_iB_i\|$ en fonction des éléments de la matrice de déplacement D_{AB} définie par

$$D_{AB} = \left(\begin{array}{ccc|c} & C_{AB} & & (AB)_A \\ \hline 0 & 0 & 0 & 1 \end{array} \right)$$

Les longueurs q_i sont obtenues par :

$$q_i = \| -(AA_i)_A + (AB)_A + C_{AB}\,(BB_i)_B \|$$

Le modèle cinématique permet de calculer les vitesses d'allongement des câbles $\dot{q}_i$ en fonction de la vitesse $\vec{V}$ de translation du point B et de la vitesse de rotation $\vec{\Omega}$ de la suspension. Pour exprimer ce modèle (pseudo-jacobienne du modèle géométrique), posons :

$$\vec{u}_i = \frac{1}{q_i}\overrightarrow{A_iB_i}$$

En observant que :

$$\dot{q}_i = \vec{u}_i \cdot \overrightarrow{\frac{d}{dt}\,(A_iB_i)}$$

et que :

$$\overrightarrow{\frac{d}{dt}\left(A_iB_i\right)} = \frac{d}{dt}\left(\overrightarrow{A_iA} + \overrightarrow{AB} + \overrightarrow{BB_i}\right) = \vec{V} + \vec{\Omega} \times \overrightarrow{BB_i}$$

Il vient :

$$\dot{q}_i = \vec{u}_i \cdot \vec{V} + \vec{u}_i \cdot \left(\vec{\Omega} \times \overrightarrow{BB_i}\right) = \vec{u}_i \cdot \vec{V} + \left(\overrightarrow{BB_i} \times \vec{u}_i\right) \cdot \vec{\Omega}$$

D'où :

$$\dot{\mathbf{q}} = \mathbf{P} \begin{pmatrix} V \\ \Omega \end{pmatrix}$$

avec :

$$\mathbf{P}^T = \left(\begin{array}{cccc} u_1 & u_2 & \cdots & u_{n_q} \\ \hline w_1 & w_2 & \cdots & w_{n_q} \end{array}\right)$$

et :

$$\vec{w}_i = \overrightarrow{BB_i} \times \vec{u}_i$$

Pour analyser la suspension, écrire l'algorithme correspondant à la fonction suivante :
`function [Q, P] = Dab2QetP(Dab`
qui renvoie le vecteur $\mathbf{q}$ à n_q composantes et la matrice cinématique $\mathbf{P}$ de dimension $n_q \times 6$ en fonction de la situation D_{AB} du repère maquette $\mathcal{R}_B$ par rapport au repère fixe $\mathcal{R}_A$.

2) Calcul des tensions dans les câbles

Le modèle "statique" donne le torseur résultant en B de l'ensemble des forces de tensions $\vec{F}_i = -f_i\vec{u}_i$ des câbles. Posons :

$$F_X = \begin{pmatrix} F = \sum_{i=1}^{n_q} \vec{F}_i = -\sum_{i=1}^{n_q} f_i\vec{u}_i \\ M_B = \sum_{i=1}^{n_q} \overrightarrow{BB_i} \times \vec{F}_i = -\sum_{i=1}^{n_q} \overrightarrow{BB_i} \times f_i\vec{u}_i \end{pmatrix} = - \begin{pmatrix} \sum_{i=1}^{n_q} f_i\vec{u}_i \\ \sum_{i=1}^{n_q} f_i\vec{w}_i \end{pmatrix}$$

En notant $F_q^T = \begin{pmatrix} f_1 & f_2 & \cdots & f_{n_q} \end{pmatrix}$, il vient :

$$F_X = -\mathbf{P}^T F_q$$

Si $\Delta = \left|\mathbf{P}^T\mathbf{P}\right| \neq 0$, la solution de norme minimale de l'équation précédente est donnée par :

$$F_q = -\mathbf{P}\left(\mathbf{P}^T\mathbf{P}\right)^{-1} F_X$$

Cette solution F_q de norme minimale peut présenter des composantes f_i négatives (câbles détendus). Elle est alors inadmissible. Pour la rendre admissible, on ajoute à cette solution une quantité $\lambda\mathbf{n}$ telle que $\mathbf{P}^T\mathbf{n} = \mathbf{0}$ et telle que la composante la plus petite de $F_q$soit égale à $f_{\min}$ (valeur fournie dans les fichiers `inidata1.m`) :

$$F_q = -\mathbf{P}\left(\mathbf{P}^T\mathbf{P}\right)^{-1} F_X + \lambda\mathbf{n}$$

Dans le cas de la simple redondance ($n_q = 7 = 6 + 1$) en configuration régulière ($\left|\mathbf{P}^T\mathbf{P}\right| \neq 0$), $\mathbf{n}$ est de dimension 1. La macro Matlab `n=null(A)` fournit ce vecteur (matrice $n \times 1$). Si le vecteur $\mathbf{n}$ a toutes ses composantes de même signe (nous les choisirons positives dans ce cas), on voit qu'en ajoutant un $\lambda\mathbf{n}$ avec λ assez grand, on peut rendre n'importe qu'elle composante de F_q supérieure ou égale à $f_{\min}$. Pour rendre la plus petite composante du résultat égale à $f_{\min}$, on prendra :

$$\lambda = \max_i \left(\frac{f_{\min} - f_i}{n_i}\right)$$

Si le vecteur $\mathbf{n}$ n'a pas toutes ses composantes de même signe, cette méthode est inapplicable.

Pour analyser la suspension, écrire l'algorithme correspondant à la fonction suivante :
`function [Fq, Det, nmin] = DabetFx2Fq(Dab,Fx)`
qui fournit la solution de $F_X = -\mathbf{P}^T F_q$ telle que $\min(f_i) = f_{\min}$. Pour surveiller l'évolution des composantes de $\mathbf{n}$, la fonction renvoi `nmin` $= \min(n_i)$.

3) Inversion itérative du modèle $\mathbf{q} = H(D_{AB})$

Cette inversion se fait à l'aide d'une adaptation de l'algorithme de Newton-Raphson :

Initialisations :

$C_{AB} = D_{AB\_ini}(1:3, 1:3)$

$(\mathrm{AB})_A = D_{AB\_ini}(1:3, 4)$

$\Delta q_{moy}^{old} = \infty$, $I = 0$

1. Génération de la matrice D_{AB}, calcul de $\mathbf{q} = H(D_{AB})$, calcul de $\Delta\mathbf{q} = \mathbf{q}^{mes} - \mathbf{q}$.
2. Examen si convergence : $\Delta q_{moy} = \frac{1}{n_q} \|\Delta q\|$. Si $\Delta q_{moy} < \varepsilon$: Fin, solution trouvée.
3. Examen si divergence : Si $\Delta q_{moy} > \Delta q_{moy}^{old}$: Fin, divergence de l'algorithme.
4. Mémorisation : $\Delta q_{moy}^{old} = \Delta q_{moy}$
5. Calcul correction linéaire : $\begin{pmatrix} \Delta B \\ \Delta \mathcal{A} \end{pmatrix} = \left(\mathbf{P}^T\mathbf{P}\right)^{-1} \mathbf{P}^T \Delta\mathbf{q}$
6. Saturation homothétique des déplacemernts correctifs :
   $\Delta d_{\max} = \max_n \left(|\Delta B \cdot e_n|\right)$; $\lambda = \Delta d_{\max}/\Delta d_{\lim}$. Si $\lambda > 1$, alors $\Delta B \leftarrow \frac{1}{\lambda}\Delta B$
   $\Delta r_{\max} = \max_n \left(|\Delta \mathcal{A} \cdot e_n|\right)$; $\lambda = \Delta r_{\max}/\Delta r_{\lim}$. Si $\lambda > 1$, alors $\Delta \mathcal{A} \leftarrow \frac{1}{\lambda}\Delta \mathcal{A}$
   ($\vec{e}_n = \vec{i}_A$ ou $\vec{j}_A$ ou $\vec{k}_A$)
7. Calcul rotation corrective : $s^2 = \|\Delta\mathcal{A}\|^2$; $R = \mathbf{I}_{3\times3} + \widetilde{\Delta\mathcal{A}} + \frac{1-\sqrt{1-s^2}}{s^2}\widetilde{\Delta\mathcal{A}}^2$
8. Correction situation : $C_{AB} \leftarrow R\,C_{AB}$ et $(\mathrm{AB})_A \leftarrow (\mathrm{AB})_A + \Delta B$
9. $I \leftarrow I + 1$. Retour en 1 si I inférieur au nombre maximal d'itérations.

Pour analyser la suspension, écrire l'algorithme correspondant à la fonction suivante :
`function [Dab, iter] = Qmes2Dab(Qmes, itermax, Dab_ini)` : Cette fonction fournit la situation D_{AB} du repère maquette $\mathcal{R}_B$ par rapport au repère fixe $\mathcal{R}_A$ qui correspond à un vecteur `Qmes` $= \mathbf{q}^{mes}$ de longueur des câbles.

ANALYSE DE LA SUSPENSION

A l'aide des outils précédents :

1. En configuration origine $D_{AB} = \mathbf{I}_{4\times4}$
   - Donner la longueur des câbles,
   - Donner le déterminant de $\mathbf{P}^T\mathbf{P}$,
   - Donner la tension des câbles pour équilibrer le poids de la maquette (appliqué en B) : solution de norme minimale, puis solution dont la plus petite composante est égale à $f_{\min}$.
2. Rechercher la situation D_{AB} qui correspond au vecteur $\mathbf{q}^{mes}$ qui figure dans le fichier `inidata1.m`.
   - Donner $(\mathrm{AB})_A$ et les angles cap, assiette et roulis de la maquette,
   - Donner la longueur $\mathbf{q}$ des câbles pour la situation D_{AB} trouvée. Comparer à $\mathbf{q}^{mes}$.
   - Donner le déterminant de $\mathbf{P}^T\mathbf{P}$.
3. Etude du volume de travail
   - Tracer long de l'axe des x, l'évolution de $\det\left(\mathbf{P}^T\mathbf{P}\right)$, $\min\left(n_i\right)$ et $\max\left(F_{qi}\right)$ où F_q est le vecteur des tensions, à composante minimale égale à $f_{Q\,\min}$, qui équilibre le poids de la maquette. Ce tracé sera limité au domaine admissible ($\det\left(\mathbf{P}^T\mathbf{P}\right) \neq 0$ et $\min\left(n_i\right) > 0$).
   - Idem le long de l'axe des y.
   - Idem le long de l'axe des z.
   - Donner les intervalles pour lesquels la tension maximale dans les brins est inférieure à la valeur $f_{Q\,\max}$.

Annexes

Fichier `inidata1.m`

```
% inidata1.m : Initialise les vecteurs globaux AAi et BBi
% position des points d'ancrages fixes et mobiles

global Nq AAi BBi Poids FqMin FqMax Qmes dLsat dAsat MyEps itermax
Poids = 5*9.81 ; % Maquette de 5 Kg
FqMin = 2*9.81 ; % Tension minimale des câbles 2 Kg
FqMax = 20*9.81 ; % Tension maximale des câbles 20 Kg
Largeur = 2;     % Largeur de la veine
R = 1;   % Rayon de la veine
LongT = 0.6 ;    % Longueur du T
LargT = 0.2 ;    % Largeur du T
dLsat = 0.1 ;    % Pas maximum autorisé en translation pour l'inversion itérative
dAsat = 0.1 ;    % Pas maximum autorisé en rotation pour l'inversion itérative
MyEps = 1.e-5 ;  % Précision désirée pour l'inversion de Q=H(Dab)
                 % utilisée également pour l'approximation R=1+Ã
itermax = 20;   % Nombre max d'itérations pour l'inversion

% Points d'ancrages fixes
Nq = 7 ;
dy = R*cos(pi/3);
dz = R*sin(pi/3);
AAi=zeros(3,Nq);
x=Largeur/2;
AAi(:,1)=[x;  0;-R];
AAi(:,2)=[x; dy;dz];
AAi(:,3)=[x;-dy;dz];
x=-Largeur/2;
AAi(:,4)=[x;  0;  R];
AAi(:,5)=[x;-dy;-dz];
AAi(:,6)=[x; dy;-dz];
AAi(:,7)= AAi(:,4);

% Points d'ancrages sur le T
BBi=zeros(3,Nq);
x=-LongT;
d= LargT/2.;
BBi(:,4)=[x;-d;0];
BBi(1,5)=x;
BBi(1,6)=x;
BBi(:,7)=[x;d;0];

% Position mesurée pour X, Y, Z, Psi, Theta, Phi ????
Qmes = [1.494992,1.255067,1.332363,0.926188,1.299455,1.244291,0.912307]';
```

Fichier `Mat2Carol.m`

```
% [Carol] = Mat2Carol(MatRot, Carol)
%
% Extraction des angles de Cap, assiette et Roulis (ordre ZYX)
% de la matrice MatRot
% Pour abs(assiette) > pi/2 - 1.E-5   conservation de la somme
% ou de la différence des valeurs passées en référence, qui
% sont également utilisées pour envoyer la solution dans la même
% nappe.
%
function [Carol] = Mat2Carol(a,Carol)

tet = -asin(a(3,1));
if abs(tet) < pi/2-1.e-7
   psi = atan2(a(2,1),a(1,1));
   phi = atan2(a(3,2),a(3,3));
else
   if tet > 0.
      som = Carol(1)+Carol(3);
      dif = atan2(-a(1,2),a(2,2));
   else
      dif = Carol(1)-Carol(3);
      som = atan2(-a(1,2),a(2,2));
   end
   psi = (som+dif)/2.;
   phi = (som-dif)/2.;
end
% On met dans la même nappe que la solution précédente
if abs(Carol(2)) > pi/2
   if psi < 0., psi = psi+pi; else psi = psi-pi; end
   if phi < 0., phi = phi+pi; else phi = phi-pi; end
   if tet > 0., tet = pi-tet; else tet =-pi-tet; end
end
Carol(1)=psi;
```

```
Carol(2)=tet;
Carol(3)=phi;
return
```

Fichier `manti.m`

```
% function M = manti(V)
% Matrice antisymétrique associée au vecteur V
function M = manti(V)
M=[ 0  -V(3) V(2);
   V(3)  0  -V(1);
  -V(2) V(1)  0  ];
return;
```

Fichiers solutions proposés

Fichier `Dab2QetP.m`

```
% function [Q P] = Dab2QetP(Dab)
% Calcule le vecteur des longueurs Q
% et la matrice cinématique P pour la situation Dab
function [Q, P] = Dab2QetP(Dab)
global AAi BBi Nq
AB = Dab(1:3,4);
Cab = Dab(1:3,1:3);
Q = zeros(Nq,1);
P = zeros(Nq,6);
for i=1:Nq
   BBi_a = Cab*BBi(:,i);
   AiBi = -AAi(:,i)+ AB + BBi_a;
   Qi = sqrt(AiBi'*AiBi);
   Q(i) = Qi;
   if Qi == 0; Ui =[1;0;0]; % Direction sans importance
   else Ui = AiBi/Qi ; end
   P(i,:)=[Ui; cross(BBi_a,Ui)]';
end
```

Fichier `Qmes2Dab.m`

```
% [Dab iter] = Qmes2Dab(Qmes, itermax, Dab_ini) : Résolution itérative de Q=H(Dab)
%
% Le nombre d'itération est limité à itermax. Si ce nombre
% est atteint, Dab = [] est renvoyé.
% La plus grande variation de Qi est limitée à dQsat,
% Il y a convergence quand max(|Qmesi-Qi|) < MyEps. Le résultat est
% renvoyé avec iter > 0
% Dans le cas contraire, il y a arrêt si l'erreur augmente. La valeur
% iter renvoyée est négative (iter < 0)
% Si on atteint le nombre max d'itérations, on renvoi iter = 0
%
% MyEps est également utilisé comme limite d'approximation
% pour négliger (1-cos(MyEps)).
%
% Le système étant sur-déterminé, on pourra examiner au retour
% l'écart entre Q = H(Dab) et Qmes.
%
function [Dab, iter] = Qmes2Dab(Qmes, itermax, Dab_ini)
global Nq dLsat dAsat MyEps
% Initialisation de la situation
Cab = Dab_ini(1:3,1:3);
AB = Dab_ini(1:3,4) ;
dQold = inf ;
I3 = eye(3) ;
for iter = 1:itermax
   % Calcul longueurs et matrice cinématique
   Dab = [Cab AB; 0 0 0 1];
   [Q P] = Dab2QetP(Dab) ;
   % Calcul erreur par rapport aux longueurs mesurées
   dQ = Qmes - Q ;
   dQmoy = norm(dQ)/Nq ;
   % Examen si convergence
   if dQmoy  < MyEps, return; end
   % Arrêt si divergence
   if dQmoy > dQold,
      iter = -iter; return;
   end;
   dQold = dQmoy ;
   % Erreur de situation correspondante
   dW = (P'*P)\(P'*dQ) ;
   dW = inv(P'*P)*(P'*dQ) ;
```

```
   % Extraction erreur de translation et rotation
   dM = dW(1:3,1);
   dA = dW(4:6,1);
   % On sature les erreurs à dLsat et dAsat
   dMmax = max(abs(dM)); coef = dMmax/dLsat ;
   if coef > 1., dM = dM/coef ; end
   dAmax = max(abs(dA)); coef = dAmax/dAsat ;
   if coef > 1., dA = dA/coef ; end
   % Correction position
   AB = AB + dM;
   Anti = manti(dA);
   % Carré du sinus angle écart orientation
   s2 = dA'*dA ;
   R = I3 + Anti + ((1-sqrt(1-s2))/s2)*Anti*Anti ;
   Cab = R*Cab ;
end
iter = 0
return
```

Fichier `DabetFx2Fq.m`

```
% [Fq Det nmin] = DabetFx2Fq(Dab, Fx)
% Calcul pour la situation Dab de
% Fq : forces qui équilibrent Fx avec la composante
% minimale à FqMin : Renvoi Fq = [], si impossible
% Det : déterminant de P'*P
% nmin : composante minimale du noyau dans la direction ppale
function [Fq, Det, nmin] = DabetFx2Fq(Dab,Fx)
global FqMin Nq
Fq = [] ;
[Q P] = Dab2QetP(Dab) ;
A = P'; AAt = A*P ;
Det = det(AAt) ;
Noy = null(A) ;
if max(Noy) < 0, Noy = -Noy; end ;
nmin = min(Noy) ;
if Det < eps | nmin <= 0, return ; end
iP = -P*inv(AAt);
Fq = iP*Fx;
% [fNeg i] = min(Fq);
% coef = (FqMin - fNeg )/Noy(i) ;
coef = max((FqMin-Fq)./Noy) ;
Fq = Fq + coef*Noy;
return
```

Fichier `main.m`

```
inidata1 ; disp('Jeu de données inidata1'); disp(' ') ;
global Poids FqMin FqMax Qmes itermax Nq
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     ANALYSE DE LA CONFIGURATION INITIALE     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Dab = eye(4);
disp('ANALYSE DE LA CONFIGURATION INITIALE');disp(' ');
% Longueur des câbles et matrice cinématique correspondante
[Q P] = Dab2QetP(Dab) ;
disp(['Longueur initiale des câbles : ']);
disp(Q');
A = P'; AAt = A*P ;
Det = det(AAt) ;
disp(['Déterminant AAt : ' num2str(Det)]); disp(' ');
if Det < 10*eps
   disp('Configuration initiale singulière.');
   disp('On arrête tout !!!') ;
   return ;
end
% Pseudo-inverse de Moore-Penrose de Pt
iP = P*inv(AAt);
% Tension des câbles de Norme minimale qui équilibre le poids
disp('Recheche de la tension des câbles qui équilibrent le poids');
FxPoids = [0 0 -Poids 0 0 0]';
Fq = iP*FxPoids;
% Impression des index des câbles non tendus
Ineg = Fq < 0 ;
if sum(Ineg) ~= 0
   disp('La solution de norme minimale a des câbles non tendus : ');
disp(Fq');
end
% Analyse du noyau. Est-il coopératif ?
Noy = null(A) ; Noy = Noy*(Noy'*ones(Nq,1)) ;
if sum(Noy < 0) ~= 0
   disp('Il y a des composantes du noyau négatives !!');
   disp('Le noyau n''est pas coopératif. Rien à faire !!');
```

```
else
   disp('Le noyau est coopératif : ');
   disp(Noy');
% Si existe des câbles non tendus, on les tend au minimum
[fNeg i] = min(Fq);
if fNeg < FqMin
    %coef = (FqMin - fNeg )/Noy(i) ;
coef = max((FqMin-Fq)./Noy) ;
    Fq = Fq + coef*Noy;
end
disp(['Tension minimale acceptée : ' num2str(FqMin)]);
disp(['Tension des câbles : ']);
disp(Fq');
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%     RECHERCHE DE LA SITUATION COURANTE       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
disp('RECHERCHE DE LA SITUATION COURANTE');disp(' ');
% Calcul Situation pour Q = Qmes
[Dab iter] = Qmes2Dab(Qmes, itermax, eye(4)) ;
if (iter == 0)
   disp(['Echec résolution après ' num2str(itermax) ' itérations !']); disp(' ');
else
   if iter < 0
      disp(['Divergence après ' num2str(-iter) ' itérations !']); disp(' ');
   else
      disp(['Convergence après ' num2str(iter) ' itérations !']); disp(' ');
   end
end

disp(['Position trouvée : ', num2str(Dab(1:3,4)')]);
disp(['Attitude zyx trouvée : ', num2str(Mat2Carol(Dab(1:3,1:3),zeros(1,3)))]);
[Q P] = Dab2QetP(Dab) ;
disp(['Longueur mesurée des câbles : ']);
disp(Qmes');
disp(['Longueur trouvée des câbles : ']);
disp(Q');
disp('ANALYSE DE LA SITUATION COURANTE');disp(' ');
Det = det(P'*P) ;
disp(['Déterminant AAt : ' num2str(Det)]); disp(' ');
if Det < 10*eps
   disp('Configuration trouvée singulière.');
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%      ETUDE  DU DOMAINE ATTEIGNABLE           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
letmin='xyz';
letmaj='XYZ';
disp ('Domaine opérationnel :');
figure(1);
pas = 0.002;
ps10=pas/10.;
for ix = 1:3
    TabDet = [];
    TabX = [];
    TabNmin = [];
    TabFmax = [];
    Dab = eye(4);
    % Excursion positive
    while 1==1
        Dab(ix,4) = Dab(ix,4) + pas ;
        [Fq Det nmin] = DabetFx2Fq(Dab,FxPoids) ;
        if isempty(Fq) break ; end
        TabX = [TabX Dab(ix,4)];
        TabDet = [TabDet Det] ;
        TabNmin = [TabNmin nmin];
        fmax = max(Fq);
        TabFmax = [TabFmax fmax];
        if fmax > FqMax, continue; end
        xmax = Dab(ix,4);
    end
    % Excursion négative
    Dab(ix,4) = pas ;
    while 1==1
        Dab(ix,4) = Dab(ix,4) - pas ;
    [Fq Det nmin] = DabetFx2Fq(Dab,FxPoids) ;
        if isempty(Fq) break ; end
        TabX = [Dab(ix,4) TabX];
        TabDet = [Det TabDet] ;
        TabNmin = [nmin TabNmin];
        fmax = max(Fq);
        TabFmax = [fmax TabFmax];
        if fmax > FqMax, continue; end
        xmin = Dab(ix,4);
```

```
    end
    subplot(3,3,ix); plot(TabX,TabDet);     grid on; if ix==1, ylabel('Det. Pt*P'); end
    subplot(3,3,ix+3); plot(TabX,TabNmin);  grid on; if ix==1, ylabel('Ni min'); end
    imin = find((TabX <= xmin+ps10) & (xmin-ps10 <= TabX) );
    imax = find((TabX <= xmax+ps10) & (xmax-ps10 <= TabX) );
    subplot(3,3,ix+6);
    plot(TabX(1:imin),TabFmax(1:imin),':'); hold on;
    plot(TabX(imin:imax),TabFmax(imin:imax));
    plot(TabX(imax:end),TabFmax(imax:end),':');
    V=axis; axis([V(1) V(2) 0 FqMax*5]);
    hold off;
    grid on; if ix==1, ylabel('Fqi max'); end
    xlabel(['Déplacement selon O' letmin(ix)]);
    disp ([num2str(TabX(imin)) ' <= ' letmaj(ix) ' <= ' num2str(TabX(imax))]);
end
```

Listing résultat

```
>> main
Jeu de données inidata1

ANALYSE DE LA CONFIGURATION INITIALE

Longueur initiale des câbles :
    1.4142    1.4142    1.4142    1.0817    1.0770    1.0770    1.0817

Déterminant AAt : 0.0017379

Recheche de la tension des câbles qui équilibrent le poids
La solution de norme minimale a des câbles non tendus :
  -34.5160   20.1214   20.1214    2.5410    2.9216    2.9216    2.5410

Le noyau est coopératif :
    1.0599    0.6119    0.6119    1.0133    1.1650    1.1650    1.0133

Tension minimale acceptée : 19.62
Tension des câbles :
   19.6200   51.3768   51.3768   54.2986   62.4301   62.4301   54.2986

RECHERCHE DE LA SITUATION COURANTE

Convergence après 4 itérations !

Position trouvée : 0.050007     0.10001        0.15
Attitude zyx trouvée : 0.050003     0.10006     0.15002
Longueur mesurée des câbles :
    1.4950    1.2551    1.3324    0.9262    1.2995    1.2443    0.9123

Longueur trouvée des câbles :
    1.4950    1.2551    1.3324    0.9262    1.2995    1.2443    0.9123

ANALYSE DE LA SITUATION COURANTE

Déterminant AAt : 0.0011977

Domaine opérationnel :
-0.236 <= X <= 0.818
-0.13 <= Y <= 0.13
-0.554 <= Z <= 0.866
```

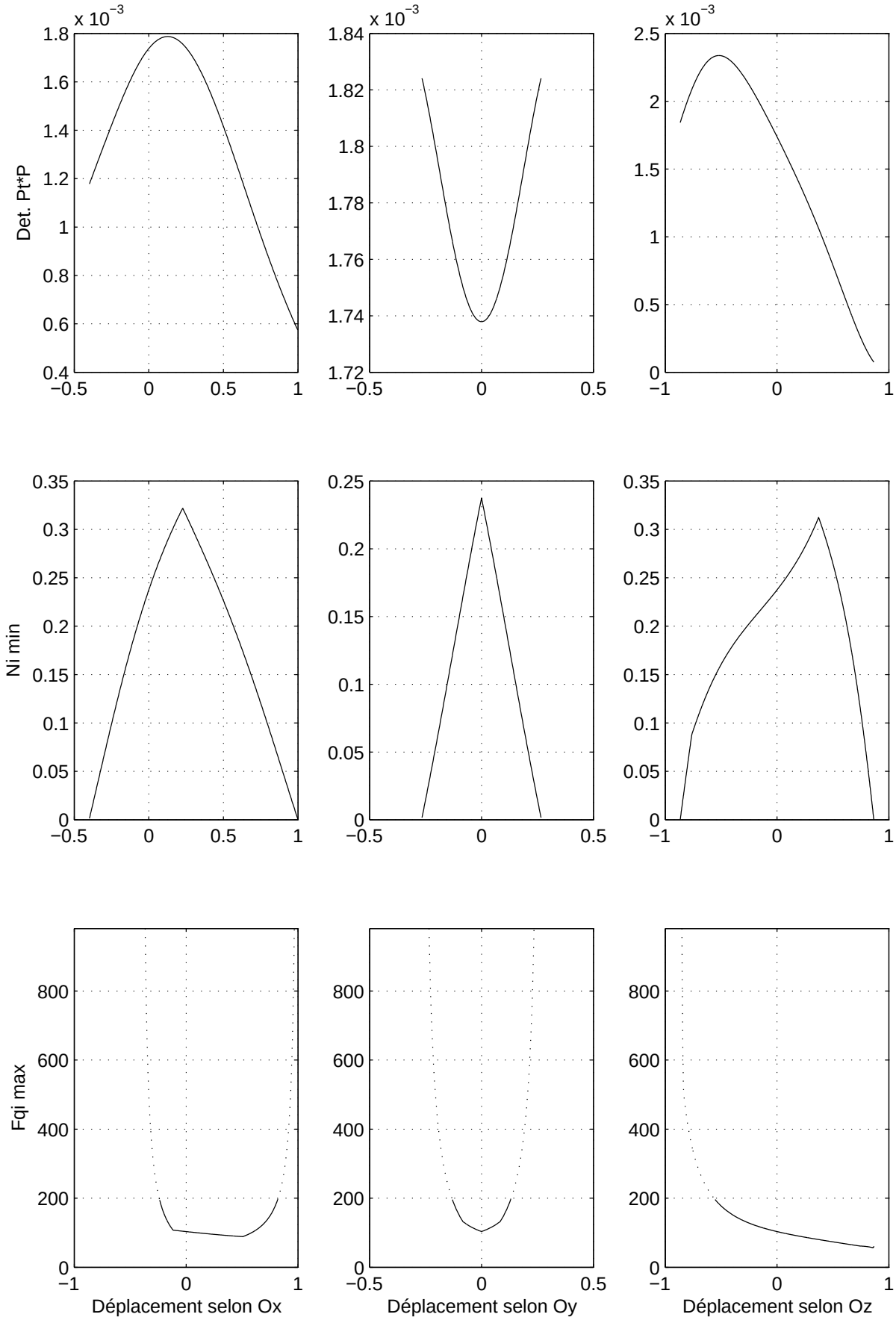

FIG. 2 – Domaine opérationnel