

## 4 Simulation du déploiement d'un câble sous-marin

Le programme sous Matlab est testé sur la simulation d'un câble tracté par un navire et dont l'extrémité immergée est libre. Initialement le câble est disposé verticalement (solution statique) et le navire accélère uniformément en droite ligne d'une vitesse nulle à une vitesse de 0.8 m/s en 30s (solution dynamique). Ensuite il maintient sa vitesse jusqu'à atteindre le régime permanent.

Les paramètres généraux du câble sont entrés dans *mParametres.m* et la trajectoire/vitesse du navire dans *fNavire.m*. Un premier programme de simulation *mStatique.m* (cf. annexes) calcule la configuration statique du câble. Le second *mDynamique.m* (cf. annexes) s'en sert comme conditions initiales de la récurrence (8). A chaque itération de la simulation, les vecteurs d'état des nœuds du câble sont enregistrés dans le fichier *Résultat.txt*.

Les systèmes d'équations statiques et dynamiques sont écrits dans les fichiers *fSysStatique.m* et *fSysDynamique.m* qui utilisent les fichiers *fSysRecStatique.m* et *fSysRecDynamique.m* de manière récurrente (cf. annexes).

Les matrices jacobiniennes des systèmes statique et dynamique doivent être écrites dans les fichiers *fSysJacStatique.m* et *fSysJacDynamique.m* qui utilisent les fichiers *fSysJacRecStatique.m* et *fSysJacRecDynamique.m* de manière récurrente (cf. annexes).

### 4.1 Schéma de Newton-Raphson

Un schéma de Newton sert à résoudre les systèmes d'équations non linéaires. Le problème est de bien séparer les inconnues du problème des fonctions explicites du temps (conditions limites aux nœuds extrêmes du câble). A chaque itération temporelle de la simulation, on fait tourner l'algorithme de Newton :

Soit  $\mathbf{F}(\mathbf{X}) = 0$  le système non linéaire (8) à résoudre et  $\mathbf{J}$  sa matrice jacobienne

Initialisations  $\mathbf{X}_0 = \mathbf{X}_{\text{ancien}}$

Tant que (résidu > précision souhaitée) faire

$$\mathbf{X}_{\text{nouveau}} = \mathbf{X}_{\text{ancien}} - \mathbf{J}^{-1}[\mathbf{F}(\mathbf{X}_{\text{ancien}})]^* \mathbf{F}(\mathbf{X}_{\text{ancien}})$$

$$\text{résidu} = \text{norme}[\mathbf{F}(\mathbf{X}_{\text{nouveau}})]$$

$$\mathbf{X}_{\text{ancien}} = \mathbf{X}_{\text{nouveau}}$$

Fin

La matrice jacobienne  $\mathbf{J}$  a été calculée analytiquement. Elle correspond à la dérivation de l'équation de mouvement (7) par rapport aux variables d'état de  $\mathbf{Y}$ . J'ai essayé de la discréteriser par la méthode des différences finies puis de l'intégrer au schéma de Newton.

## 8.2 Programmes Matlab

```
% mDynamique.m
%
% EOLLIVIER
% 01/04/01
%
% Résolution du problème non linéaire d'un câble élastique par la
% méthode implicite des différences finies
% d(Y)/ds - M(Y)*d(Y)/dt - P(Y) = 0
%
% Y_ki = [ epsilon_ki
%           f2_ki
%           f3_ki
%           v1_ki
%           v2_ki
%           v3_ki
%           beta0_ki
%           beta1_ki
%           beta2_ki
%           beta3_ki
%           omega2_ki
%           omega3_ki ]
%
% k = 1..Np
% y_ki désigne la valeur de la variable y au kème noeud à l'itération i
%
clear all
cd Z:\MFiles
mParametres
%
% CONDITIONS INITIALES = PROBLEME STATIQUE
%
disp('RESOLUTION DU PROBLEME STATIQUE')
mStatique
[R0, theta0, v0, alpha0, Vx0, Vy0, beta0] = fNavire(0);

Y = zeros(12,Np);
Y(1:3,1:Np) = X(1:3,1:Np);
Y(4,1) = 2*Vx0*(beta0(2,1)*beta0(3,1) - beta0(1,1)*beta0(4,1)) + ...
          2*Vy0*(beta0(2,1)*beta0(4,1) + beta0(1,1)*beta0(3,1));
Y(5,1) = Vx0*(beta0(1,1)^2 - beta0(2,1)^2 + beta0(3,1)^2 - beta0(4,1)^2) + ...
          2*Vy0*(beta0(3,1)*beta0(4,1) - beta0(1,1)*beta0(2,1));
Y(6,1) = 2*Vx0*(beta0(3,1)*beta0(4,1) + beta0(1,1)*beta0(2,1)) + ...
          Vy0*(beta0(1,1)^2 - beta0(2,1)^2 - beta0(3,1)^2 + beta0(4,1)^2);
Y(7:10,1:Np) = X(4:7,1:Np);
Y(11:12,1:Np) = X(9:10,1:Np);

save Y

% INITIALISATION
Y0 = zeros(12,Np-1);
%Y0(1:12,2:Np-1) = Y(1:12,2:Np-1);
%Y0(1:3,1) = Y(1:3,1);
%Y0(4:12,1) = Y(4:12,Np);

%
% RESULTAT DE LA SOLUTION STATIQUE DANS Resultat.txt
%
fichier = fopen('Resultat.txt','w');
for j = 1:12
    for k = 1:Np
```

```

        fprintf(fichier,'%14.10f\t',Y(j,k));
    end
    fprintf(fichier,'\r');
end
fprintf(fichier,'\r');

for i = 1:floor(T/dt)
    % -----
    % RESOLUTION DU SYSTEME fSysDynamique(Y) = 0
    %
    % fSysDynamique.m
    % fRecDynamique.m
    %
    % -----
    % CONDITIONS LIMITES = FONCTIONS EXPLICITES DU TEMPS
    %

    [R, theta, V, alpha, Vx, Vy, beta] = fNavire(dt*i);
    Y2 = zeros(12,1);

    % EXTREMITE DU CABLE
    % efforts
    Y2(1,1) = 0;
    Y2(2,1) = 0;
    Y2(3,1) = 0;

    % AMARRAGE AU NAVIRE
    % vitesse
    Y2(4,1) = 2*Vx*(beta(2,1)*beta(3,1) - beta(1,1)*beta(4,1)) + ...
               2*Vy*(beta(2,1)*beta(4,1) + beta(1,1)*beta(3,1));
    Y2(5,1) = Vx*(beta(1,1)^2 - beta(2,1)^2 + beta(3,1)^2 - beta(4,1)^2) + ...
               2*Vy*(beta(3,1)*beta(4,1) - beta(1,1)*beta(2,1));
    Y2(6,1) = 2*Vx*(beta(3,1)*beta(4,1) + beta(1,1)*beta(2,1)) + ...
               Vy*(beta(1,1)^2 - beta(2,1)^2 - beta(3,1)^2 + beta(4,1)^2);

    % orientation
    Y2(7,1) = beta(1,1);
    Y2(8,1) = beta(2,1);
    Y2(9,1) = beta(3,1);
    Y2(10,1) = beta(4,1);

    % courbure
    Y2(11,1) = 0;
    Y2(12,1) = 0;

    save Y2

    % RESOLUTION DU SYSTEME
    disp('RESOLUTION DU PROBLEME DYNAMIQUE')
    options = optimset('Display','iter');
    [Y1, fevalDynamique] = fsolve('fSysDynamique',Y0,options);

    % FUSION INCONNUES/FONCTIONS EXPLICITES
    Y = zeros(12,Np);
    Y(1:12,2:Np-1) = Y1(1:12,2:Np-1);
    Y(1:3,1) = Y1(1:3,1);
    Y(4:12,1) = Y2(4:12,1);
    Y(1:3,Np) = Y2(1:3,1);
    Y(4:12,Np) = Y1(4:12,1);

    %
    % -----
    % SAUVEGARDE DE LA SOLUTION DANS Y.mat
    %

    save Y

```

```

% -----
% ECRITURE DE LA SOLUTION DANS Resultat.txt
%

for j = 1:12
    for k = 1:Np
        fprintf(fichier, '%14.10f\t', Y(j,k));
    end
    fprintf(fichier, '\r');
end
fprintf(fichier, '\r');
fclose(fichier)
disp('THE END')

```

```

function GG = fSysDynamique(YY)
% SYSTEME DYNAMIQUE NON LINEAIRE
% YY_ki = [ Y_1i , ... Y_ki ... , Y_Npi ]
% GG_ki = [ G_1i , ... G_ki ... , G_Npi ]'
%
% k = 1..Np
% Y_ki désigne le vecteur d'état au kème noeud à l'itération i
% G_ki désigne le kème système d'équations non linéaires

% -----
% PARAMETRES DU PROBLEME
%

mParametres

[m, n] = size(YY);
if (m == 12)&(n == Np-1)
    GG = zeros(m*(Np-1),1);
    %
    % EQUATIONS DU CABLE
    %

    % ETAT DES NOEUDS A L'INSTANT PRECEDENT
    load Y.mat
    YY1 = Y;

    % FONCTIONS EXPLICITES DU TEMPS
    load Y2.mat
    YY2 = Y2;

    %
    % SYSTEME D'EQUATIONS NON LINEAIRES
    %

    % AMARRAGE AU NAVIRE = EFFORTS INCONNUS
    [M11, P11] = fRecDynamique(YY1(1:m,1));
    [M21, P21] = fRecDynamique(YY1(1:m,2));
    [M12, P12] = fRecDynamique([YY1(1:3,1);YY2(4:12,1)]); % [INCONNUES; CONNUES]
    [M22, P22] = fRecDynamique(YY1(1:m,2));

    GG(1:m,1) = 2*dt*(YY1(1:m,2)-[YY1(1:3,1);YY2(4:12,1)]+YY1(1:m,2)-YY1(1:m,1)) -
    ...
    ds*((M11+M12)*([YY1(1:3,1);YY2(4:12,1)]-YY1(1:m,1))+(M21+M22)*(YY1(1:m,2)-
    YY1(1:m,2))) - ...
    dt*ds*(P11 + P12 + P21 + P22);

    % NOEUDS DU CABLE = EFFORTS ET DEPLACEMENTS INCONNUS
    for k = 3:Np-1
        [M11, P11] = fRecDynamique(YY1(1:m,k-1));
        [M21, P21] = fRecDynamique(YY1(1:m,k));
        [M12, P12] = fRecDynamique(YY1(1:m,k-1));
        [M22, P22] = fRecDynamique(YY1(1:m,k));

        GG((k-2)*m+1:(k-1)*m,1) = 2*dt*(YY1(1:m,k)-YY1(1:m,k-1)+YY1(1:m,k)-YY1(1:m,k-
        1)) - ...
        ds*((M11+M12)*(YY1(1:m,k-1)-YY1(1:m,k-1))+(M21+M22)*(YY1(1:m,k)-YY1(1:m,k))) -
        ...
        dt*ds*(P11 + P12 + P21 + P22);
    end

    % EXTREMITE DU CABLE = DEPLACEMENTS INCONNUS
    [M11, P11] = fRecDynamique(YY1(1:m,Np-1));
    [M21, P21] = fRecDynamique(YY1(1:m,Np));
    [M12, P12] = fRecDynamique(YY1(1:m,Np-1));
    [M22, P22] = fRecDynamique([YY2(1:3,1);YY2(4:12,1)]); % [CONNUES; INCONNUES]

```

```

    GG((Np-2)*m+1:(Np-1)*m,1) = 2*dt*([YY2(1:3,1);YY(4:12,1)]-YY(1:m,Np-
1)+YY1(1:m,Np)-YY1(1:m,Np-1)) - ...
    ds*((M11+M12)*(YY(1:m,Np-1)-YY1(1:m,Np-
1))+(M21+M22)*([YY2(1:3,1);YY(4:12,1)]-YY1(1:m,Np))) - ...
    dt*ds*(P11 + P12 + P21 + P22);
else
    error('! mauvais argument !')
end

```

PARAMETRES DE SIMULATION

INITIALISATION

INITIALISATION DES DIFFERENTES VARIABLES

```

% initialisation de Y1
M1 = zeros(12,6);
M1(1,1) = M0/2;
M1(2,1) = M0/2;
M1(3,1) = M0/2;
M1(4,1) = M0/2;
M1(5,1) = 1;

M2 = zeros(12,1);
M2(1,1) = 2*Y(1,1)*Y(1,1);
M2(2,1) = 2*M0*Y(1,1)*Y(1,1);
M2(3,1) = 2*Y(3,1)*Y(3,1);

M3 = zeros(12,1);
M3(1,1) = Y(1,1)/M0;
M3(2,1) = -2*Y(1,1)*Y(1,1)/M0;
M3(3,1) = -2*Y(3,1)*Y(3,1)/M0;

M4(1,1) = 0;
M4(2,1) = -2*Y(1,1)*Y(1,1);
M4(3,1) = -2*Y(3,1)*Y(3,1);

% initialisation de YY
YY1(1,1) = 2*M0*Y(1,1)*Y(1,1) - Y(1,1)*Y(1,1)/M0;
YY1(2,1) = 2*M0*Y(2,1)*Y(2,1) - Y(2,1)*Y(2,1)/M0;
YY1(3,1) = -2*M0*Y(3,1)*Y(3,1) + Y(3,1)*Y(3,1)/M0;

YY2(1,1) = 0;
YY2(2,1) = 2*Y(2,1)*Y(2,1);
YY2(3,1) = 2*Y(3,1)*Y(3,1);

% initialisation de YY2
YY2(1,1) = 2*M0*Y(1,1)*Y(1,1) - Y(1,1)*Y(1,1)/M0;
YY2(2,1) = 2*M0*Y(2,1)*Y(2,1) - Y(2,1)*Y(2,1)/M0;
YY2(3,1) = 2*M0*Y(3,1)*Y(3,1) - Y(3,1)*Y(3,1);

```

```

function [M, P] = fRecDynamique(Y)
% RECURRENCE DU SYSTEME DISCRET DYNAMIQUE
% Y_ki = [ epsilon_ki = Y(1,1)
%           f2_ki     = Y(2,1)
%           f3_ki     = Y(3,1)
%           v1_ki     = Y(4,1)
%           v2_ki     = Y(5,1)
%           v3_ki     = Y(6,1)
%           beta0_ki   = Y(7,1)
%           betal_ki   = Y(8,1)
%           beta2_ki   = Y(9,1)
%           beta3_ki   = Y(10,1)
%           omega2_ki  = Y(11,1)
%           omega3_ki  = Y(12,1) ]
%
% k = 2..Np
% y_ki désigne la valeur de la variable y au kème noeud à l'itération i
%
```

```

% -----
% PARAMETRES DU PROBLEME
%
```

mParametres

```

% -----
% METHODE IMPLICITE DES DIFFERENCES FINIES
%

% matrice M(Y)
M1 = zeros(12,6);
M1(1,4) = Mc/EA;
M1(2,5) = Mc + Ma;
M1(3,6) = Mc + Ma;
M1(4,1) = 1;

M2 = zeros(12,1);
M2(1,1) = 2*Mc*(Y(10,1)*Y(5,1) - Y(9,1)*Y(6,1))/EA;
M2(2,1) = 2*Mc*(Y(8,1)*Y(6,1) - Y(10,1)*Y(4,1));
M2(3,1) = 2*Mc*(Y(9,1)*Y(4,1) - Y(8,1)*Y(5,1));

M2(4,1) = 0;
M2(5,1) = -2*Y(10,1)*(1 + Y(1,1));
M2(6,1) = 2*Y(9,1)*(1 + Y(1,1));

M3 = zeros(12,1);
M3(1,1) = 2*Mc*(Y(9,1)*Y(5,1) + Y(10,1)*Y(6,1))/EA;
M3(2,1) = -2*Mc*(Y(9,1)*Y(4,1) + Y(7,1)*Y(6,1));
M3(3,1) = -2*Mc*(Y(7,1)*Y(5,1) - Y(10,1)*Y(4,1));

M3(4,1) = 0;
M3(5,1) = -2*Y(9,1)*(1 + Y(1,1));
M3(6,1) = -2*Y(10,1)*(1 + Y(1,1));

M4 = zeros(12,1);
M4(1,1) = 2*Mc*(Y(7,1)*Y(6,1) - Y(8,1)*Y(5,1))/EA;
M4(2,1) = 2*Mc*(Y(8,1)*Y(4,1) + Y(10,1)*Y(6,1));
M4(3,1) = -2*Mc*(Y(10,1)*Y(5,1) + Y(7,1)*Y(4,1));

M4(4,1) = 0;
M4(5,1) = 2*Y(8,1)*(1 + Y(1,1));
M4(6,1) = -2*Y(7,1)*(1 + Y(1,1));

M5 = zeros(12,1);
M5(1,1) = -2*Mc*(Y(8,1)*Y(6,1) + Y(7,1)*Y(5,1))/EA;
M5(2,1) = 2*Mc*(Y(7,1)*Y(4,1) - Y(9,1)*Y(6,1));
M5(3,1) = 2*Mc*(Y(9,1)*Y(5,1) + Y(8,1)*Y(4,1));

```

```

M5(4,1) = 0;
M5(5,1) = 2*Y(7,1)*(1 + Y(1,1));
M5(6,1) = 2*Y(8,1)*(1 + Y(1,1));

M6 = zeros(12,2);

M = [M1 M2 M3 M4 M5 M6];

% matrice P(Y)
if nargout > 1
P = zeros(12,1);

P(1,1) = (Y(12,1)*Y(2,1) - Y(11,1)*Y(3,1) - ...
W*(Y(7,1)^2 + Y(8,1)^2 - Y(9,1)^2 - Y(10,1)^2))/EA;
P(2,1) = -Y(12,1)*EA*Y(1,1) - ...
2*W*(Y(8,1)*Y(9,1) + Y(7,1)*Y(10,1));
P(3,1) = Y(11,1)*EA*Y(1,1) - ...
2*W*(Y(8,1)*Y(10,1) - Y(7,1)*Y(9,1));

P(4,1) = Y(12,1)*Y(5,1) - Y(11,1)*Y(6,1);
P(5,1) = -Y(12,1)*Y(4,1);
P(6,1) = Y(11,1)*Y(4,1);

P(7,1) = -0.5*(Y(9,1)*Y(11,1) + Y(10,1)*Y(12,1));
P(8,1) = 0.5*(Y(10,1)*Y(11,1) - Y(9,1)*Y(12,1));
P(9,1) = 0.5*(Y(7,1)*Y(11,1) + Y(8,1)*Y(12,1));
P(10,1) = 0.5*(-Y(8,1)*Y(11,1) + Y(7,1)*Y(12,1));

P(11,1) = (1 + Y(1,1))*Y(3,1)/EI;
P(12,1) = -(1 + Y(1,1))*Y(2,1)/EI;
end

```

```

% mStatique.m
%
% EOllivier
% 01/04/01
%
% Résolution du problème non linéaire d'un câble élastique par la
% méthode implicite des différences finies
% d(X_k0)/dt - R(X_k0) = 0
%
% X_k0 = [ epsilon_k0
%           f2_k0
%           f3_k0
%           beta0_k0
%           beta1_k0
%           beta2_k0
%           beta3_k0
%           omega1_k0
%           omega2_k0
%           omega3_k0 ]
%
% k = 1..Np
% x_k0 désigne la valeur de la variable x au kème noeud en équilibre statique
cd Z:\MFiles

% -----
% PARAMETRES DU PROBLEME
%

mParametres

% -----
% CONDITIONS INITIALES
%

X0 = zeros(10,Np);
feval0 = fSysStatique(X0);

% -----
% RESOLUTION DU SYSTEME fSysStatique(X) = 0
%
% fSysStatique.m
% fRecStatique.m

% RESOLUTION DU SYSTEME
Xold = X0;
residu = 1;
residuMin = 1e-8;

iteration = 0;
iterationMax = 10;

% schéma de Newton
while (residu > residuMin)&(iteration < iterationMax)
    [F, J] = fSysStatique(Xold);
    if det(J) ~= 0;
        % résolution du système linéaire: Xnew = Xold - J^-1[F(Xold)]*F(Xold)
        delta1X = inv(J)*F;
        delta2X = zeros(10,Np);

        % reshape de la matrice delta1X -> delta2X
        for l = 1:Np
            delta2X(1:Np,l) = delta1X((l-1)*10+1:l*10,1);
        end

        Xnew = Xold - delta2X;
    else
        residu = 1;
    end
    residu = norm(F);
    iteration = iteration + 1;
end

```

```

Xold = Xnew;
residu = norm(F);
iteration = iteration + 1;
else
    error('! pas de solution !');
end
end
if (iteration >= iterationMax)
    error('! pas convergent !');
else
    X = Xnew
    fevalStatique = fSysStatique(Xnew)
end

```

```

function [FF, JJ] = fSysStatique(XX)
% SYSTEME STATIQUE NON LINEAIRE
% XX = [ X_10, ... X_k0 ... , X_Np0 ]
% FF = [ F_10, ... F_k0 ... , F_Np0 ]'
% JJ = Jacobien du système
%
% k = 1..Np
% X_k0 désigne le vecteur d'état au kème noeud en équilibre statique
% F_k0 désigne le kème système d'équations non linéaires
%
% -----
% PARAMETRES DU PROBLEME
%

mParametres

[m, n] = size(XX);
if (m == 10)&(n == Np)
    FF = zeros(m*Np,1);

%
% SYSTEME D'EQUATIONS NON LINEAIRES
%

for k = 2:Np
    FF((k-1)*m+1:k*m,1) = 2*(XX(1:m,k) - XX(1:m,k-1)) - ...
        ds*(fRecStatique(XX(1:m,k)) + fRecStatique(XX(1:m,k-1)));
end

%
% CONDITIONS LIMITES
%

% EXTREMITE DU CABLE
% efforts
FF(1,1) = XX(1,Np);
FF(2,1) = XX(2,Np);
FF(3,1) = XX(3,Np);

% AMARRAGE AU BATEAU
% orientation
FF(4,1) = XX(4,1) - 1;
FF(5,1) = XX(5,1);
FF(6,1) = XX(6,1);
FF(7,1) = XX(7,1);

% courbure
FF(8,1) = XX(8,1);
FF(9,1) = XX(9,1);
FF(10,1) = XX(10,1);

else
    error('! mauvais argument !')
end

if nargout > 1
    JJ = zeros(m*Np,m*Np);

%
% JACOBIENNE
%

for k = 2:Np
    JJ((k-1)*m+1:k*m,(k-1)*m+1:k*m) = 2*diag(ones(m,1)) - ...
        ds*fRecJacStatique(XX(1:m,k));
end

```

```

JJ((k-1)*m+1:k*m, (k-2)*m+1:(k-1)*m) = -2*diag(ones(m,1)) - ...
ds*fRecJacStatique(XX(1:m,k-1));
end

% -----
% CONDITIONS LIMITES
%

% EXTREMITE DU CABLE
% efforts
JJ(1,m*(Np-1)+1) = 1;
JJ(2,m*(Np-1)+2) = 1;
JJ(3,m*(Np-1)+3) = 1;

% AMARRAGE AU BATEAU
% orientation
JJ(4,4) = 1;
JJ(5,5) = 1;
JJ(6,6) = 1;
JJ(7,7) = 1;

% courbure
JJ(8,8) = 1;
JJ(9,9) = 1;
JJ(10,10) = 1;

```

end

```

function R = fRecStatique(X)
% RECURRENCE DU SYSTEME DISCRET STATIQUE
% X_k0 = [ epsilon_k0 = X(1,1)
%           f2_k0      = X(2,1)
%           f3_k0      = X(3,1)
%           beta0_k0   = X(4,1)
%           betal_k0   = X(5,1)
%           beta2_k0   = X(6,1)
%           beta3_k0   = X(7,1)
%           omega1_k0  = X(8,1)
%           omega2_k0  = X(9,1)
%           omega3_k0  = X(10,1) ]
%
% k = 2..Np
% x_k0 désigne la valeur de la variable x au kème noeud en équilibre statique
%
% -----
% PARAMETRES DU PROBLEME
%

mParametres

% -----
% METHODE IMPLICITE DES DIFFERENCES FINIES
%

R = zeros(10,1);
R = [
(X(10,1)*X(2,1) - X(9,1)*X(3,1) - ...
W*(X(4,1)^2 + X(5,1)^2 - X(6,1)^2 - X(7,1)^2))/EA;
X(8,1)*X(3,1) - X(10,1)*EA*X(1,1) - ...
2*W*(X(5,1)*X(6,1) + X(4,1)*X(7,1));
X(9,1)*EA*X(1,1) - X(8,1)*X(2,1) - ...
2*W*(X(5,1)*X(7,1) - X(4,1)*X(6,1));
-0.5*(X(5,1)*X(8,1) + X(6,1)*X(9,1) + X(7,1)*X(10,1));
0.5*(X(4,1)*X(8,1) + X(7,1)*X(9,1) - X(6,1)*X(10,1));
0.5*(-X(7,1)*X(8,1) + X(4,1)*X(9,1) + X(5,1)*X(10,1));
0.5*(X(6,1)*X(8,1) - X(5,1)*X(9,1) + X(4,1)*X(10,1));
0;
X(8,1)*X(10,1)*(1 - GI/EI) + ((1 + X(1,1))^3)*X(3,1)/EI;
X(8,1)*X(9,1)*(GI/EI - 1) - ((1 + X(1,1))^3)*X(2,1)/EI
];

```

```

function J = fRecJacStatique(X)
% JACOBIEN DU SYSTEME STATIQUE
% X_k0 = [ epsilon_k0 = X(1,1)
%           f2_k0 = X(2,1)
%           f3_k0 = X(3,1)
%           beta0_k0 = X(4,1)
%           beta1_k0 = X(5,1)
%           beta2_k0 = X(6,1)
%           beta3_k0 = X(7,1)
%           omega1_k0 = X(8,1)
%           omega2_k0 = X(9,1)
%           omega3_k0 = X(10,1) ]
%
% k = 2..Np
% x_k0 désigne la valeur de la variable x au kème noeud en équilibre statique
%
% -----
% PARAMETRES DU PROBLEME
%

mParametres

%
% -----
% DERIVATION DES EQUATIONS
%

J = zeros(10,10);
J(1,:) = [
0;
X(10,1)/EA;
-X(9,1)/EA;
-2*W*X(4,1)/EA;
-2*W*X(5,1)/EA;
2*W*X(6,1)/EA;
2*W*X(7,1)/EA;
0;
-X(3,1)/EA;
X(2,1)/EA
];
;

J(2,:) = [
-X(10,1)*EA;
0;
X(8,1);
-2*W*X(7,1);
-2*W*X(6,1);
-2*W*X(5,1);
-2*W*X(4,1);
X(3,1);
];

```

```
0;  
-EA*X(1,1)  
];'
```

```
J(3,:) = [  
X(9,1)*EA;  
-X(8,1);  
0;  
2*W*X(6,1);  
-2*W*X(7,1);  
2*W*X(4,1);  
-2*W*X(5,1);  
-X(2,1);  
EA*X(1,1);  
0  
];'
```

```
J(4,:) = [  
0;  
0;  
0;  
0;  
-0.5*X(8,1);  
-0.5*X(9,1);  
-0.5*X(10,1);  
-0.5*X(5,1);  
-0.5*X(6,1);  
-0.5*X(7,1)  
];'
```

```
J(5,:) = [  
0;  
0;  
0;  
0.5*X(8,1);  
0;  
-0.5*X(10,1);  
0.5*X(9,1);
```

```

0.5*X(4,1);
0.5*X(7,1);
-0.5*X(6,1)
]';

J(6,:) = [
0;
0;
0;
0.5*X(9,1);
0.5*X(10,1);
0;
-0.5*X(8,1);
-0.5*X(7,1);
0.5*X(4,1);
0.5*X(5,1)
]';

J(7,:) = [
0;
0;
0;
0.5*X(10,1);
-0.5*X(9,1);
0.5*X(8,1);
0;
0.5*X(6,1);
-0.5*X(5,1);
0.5*X(4,1)
]';

J(9,:) = [
(3*(1 + X(1,1))^2)*X(3,1)/EI;
0;
((1 + X(1,1))^3)/EI;
0;
0;
0;
0;

```

```
X(10,1)*(1 - GI/EI);  
0;  
X(8,1)*(1 - GI/EI)  
]';  
  
J(10,:) = [  
-(3*(1 + X(1,1))^2)*X(2,1)/EI;  
-((1 + X(1,1))^3)/EI;  
0;  
0;  
0;  
0;  
0;  
0;  
X(9,1)*(GI/EI - 1);  
X(8,1)*(GI/EI - 1);  
0  
]';
```

```

% mParametres.m
%
% EOLLIVIER
% 01/04/01
%
% Paramètres du problème de câble élastique
%

% CABLE
L = 30;
d = 1.676e-2;
rhoc = 1140;
g = 9.81;
W = 2.335;
Mc = W/g + rhoc*pi*(d/2)^2;
E = 2e10;
EA = E*pi*(d/2)^2;
EI = 80;
GI = 100;

% FLUIDE
rho = 1e3;
Cdn = 1.2;
Cdt = 0.15;
Ca = 1;
Ma = Ca*rho*pi*(d/2)^2;

% SIMULATION
Np = 8;
ds = L/Np;
dt = 0.1;
T = 0.1;

```