Présentation de MEFlab / MEFtave

C'est un ensemble de scripts[1] MATLAB / OCTAVE permettant d'illustrer les différents chapitres du cours éléments finis, cet ensemble est ouvert et évolutif.

Vous pouvez l'utiliser tel quel comme un applicatif du cours et ne traiter que les exemples proposés dans les différents chapitres, il est alors inutile de lire ce document.

Vous pouvez, à partir de l'étude des scripts proposés, développer vos propres scripts pour d'autres problèmes, que ceux abordés dans le cadre de ce cours.

Ce document présente : Analyse des scripts éléments finis
Description des scripts de données

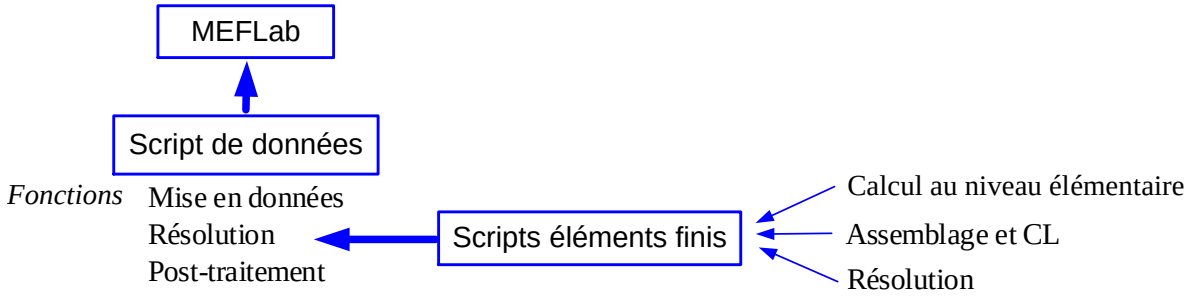
Le script d'initialisation *MEFlab /MEFtave*

L'intérêt de ce script est de permettre une utilisation rapide et directe de tous les autres scripts. Sous MATLAB comme sous OCTAVE il permet de définir les chemins d'accès aux différents répertoires ce qui est indispensable pour pouvoir utiliser les scripts éléments finis que nous proposons. Sous OCTAVE il y a en plus le choix du kit graphique "qt", "fltk" ou "gnuplot", par défaut le Kit "qt" est sélectionné. Le script *MEFtave* supprime aussi l'affichage de tous les "warnings"

```
L'organisation des répertoires est la suivante :
    [Data]              Scripts MEFlab des exercices proposés sur le site
    [Dessin]            Fonctions d'affichage graphique
    [Elements]          Fonctions élémentaires (matrices Ke Me Fe)
    [Generaux]          Scripts éléments finis de résolution du problème.
    [Sol_analytique]    Solutions analytiques connues (comparaison)
    [Work]              Scripts Matlab indépendants des routines MEFlab
```

Le script d'initialisation n'est exécuté qu'en début de session de travail, vous pouvez ensuite lancer n'importe quel script situé dans un des répertoires [Data] ou [Work]

L'organigramme général des scripts *MEFlab* est le suivant :



Analyse des scripts éléments finis

Les scripts éléments finis sont placés dans les répertoires **[Generaux]** et **[Elements]**

Ces scripts illustrent le cours EF, analyser ces scripts sans avoir étudié le cours est illusoire. Ne pas analyser la structure de ces programmes vous fera inévitablement passer à coté de certaines compétences.

[1] Fichier M-file « nom.m », contenant une séquence d'instructions MATLAB / OCTAVE qui sera exécutée en tapant « nom » dans la fenêtre de commande.

Les variables globales :

Les variables globales sont communes à tous les scripts éléments finis. Ces variables globales sont définies par un « pré-processeur ». C'est le premier rôle du script de mise en données du problème (l'analyse de la mise en données est proposée dans le paragraphe suivant).

```
global nddln nnod nddlt nelt nnode ndim ncld
global Coord Connec Typel Nprop Prop Ncl Vcl F

%   nddln : nb de ddl[2] par noeud
%   nnod  : nb de noeuds
%   nddlt : nb de ddl total (= ndln*nnod)
%   nelt  : nb d'éléments
%   nnode : nb de noeuds par élément
%   ndim  : dimension du problème (1D,2D ou 3D)
%   ncld  : nb de conditions en déplacement imposé (Dirichlet)
%
%   Coord(nnod,ndim)  : Tableau des coordonnées des noeuds
%   Connec(nelt,nnode): Tableau de connectivités des éléments
%   Typel(nelt)       : Type des éléments (barre, poutre, …)
%   Nprop(nelt)       : N° de la caractéristique de chaque élément
%   Prop(nprop,ncar)  : Tableau des caractéristiques mécaniques (…)
%   Ncl(nddlt)        : vaut 1 si le champ est imposé (0 sinon)
%   Vcl(nddlt)        : valeurs des déplacements imposés
%   F(nddlt)          : vecteur des charges nodales données
```

Le type ***« Typel »*** caractérise le type d'élément utilisé pour résoudre le problème physique EF-treillis, EF-portique, EF-élasticité plane, EF-thermique, etc.

Le tableau des propriétés ***« Prop »*** dépendra bien entendu du type des éléments utilisés, et de la nature du problème traité statique, dynamique linéaire non linéaire, etc.

Le vecteur ***« Vcl »*** indique si sur le *ddl* concerné il y a une condition de Dirichlet (champ imposé). Le vecteur ***« Vcl »*** représente les valeurs du champ imposé (par défaut il est initialisé à zéro). Et le vecteur ***« F »*** donne les valeurs des flux imposés (conditions de Neumann).

Les calculs :

L'objectif des 3 scripts « ***statiqueU, statiqueUR, statique*** » est la résolution d'un problème linéaire statique ou stationnaire. Ils peuvent être utilisés pour le calcul statique des structures treillis ou portiques telles que celles que vous avez eu l'occasion de traiter à la main en TD et en TA. Mais aussi pour les problèmes d'élasticité plane ou les problèmes stationnaires de la physique.

StatiqueU	: ne donne en sortie que les valeurs nodales du champ.
StatiqueUR	: même script complété par le calcul des réactions (flux inconnus).
Statique	: équivalent au précédent avec un autre algorithme de résolution[3].

Le script ***«vibrations »*** donne les fréquences et modes de vibration d'une structure modélisée en éléments finis.

Comprendre le fonctionnement de ces scripts vous permettra par la suite de développer vos propres scripts adaptés au type de problème que vous aurez à traiter.

[2] ddl : « degrés de liberté » ce sont les variables nodales du modèle éléments finis
[3] Utilise l'algorithme de la méthode du terme unité sur la diagonale proposé par *Dhatt & Touzot*.

Analysons le script « StatiqueUR »

L'assemblage des matrices élémentaires

Boucle caractéristique de la méthode des éléments finis:

```
for iel=1:nelt                          %boucle sur les éléments
    [Ke,Fe] = feval(Typel(iel,:),iel);  %calcul des matrices élémentaires
    loce=[];
    for i=1:nnode                       %localisation des ddl de l'élément
          loce=[loce,(Connec(iel,i)-1)*nddln+[1:nddln]];
    end
    K(loce,loce)=K(loce,loce) + Ke;     %assemblage
    Fg(loce)=Fg(loce) + Fe;
end
```

Analyse:

L'assemblage des matrices élémentaires nécessite de localiser la position des ddl de l'élément dans le vecteur global. La boucle interne consiste à placer dans le vecteur ***«loce»*** la position des ddl de chaque nœud de l'élément.

Exemple

Soit l'élément 7-9 d'un treillis 2D

Les variables élémentaires (ddl) sont : $\langle u_7 \quad v_7 \quad u_9 \quad v_9 \rangle$

Qui occupent respectivement les positions $[13 \quad 14 \quad 17 \quad 18]$ ***= loce***

Les instructions MATLAB permettent de manipuler globalement les matrices, ce qui rend très simple la création du vecteur « `loce` *».*

Exercice :

Ces lignes de programmation conviennent pour des éléments de même type

nnode = Cte

Proposez une simple modification de la boucle qui permette de créer le vecteur ***«loce»*** pour des éléments ayant un nombre de nœuds différents.

Indication : «nnode» sera le nombre maxi de nœuds des éléments utilisés. [4]

L'assemblage consiste ensuite à placer matrice et vecteur élémentaires dans la matrice globale, en manipulant globalement les matrices *merci Matlab*.

Matrice et vecteur élémentaires

Le script *Matlab* effectuant le calcul porte le nom ***Typel(iel)*** , tous ces scripts sont regroupés dans le répertoire **`[Elements]`**

Exemple ***«barre_ke»***

Calcul de la matrice raideur élémentaire et du vecteur force généralisé d'une charge répartie pour un élément barre.

Les variables globales utilisées par ce script sont : la table de coordonnées des nœuds (`Coord`), la table de connectivité des éléments (`Connec`), les tables des caractéristiques mécaniques (`Nprop` et `Prop`) donnant les valeurs de la raideur ***ES*** et les pressions linéiques ***fx*** et ***fy*** appliquées sur l'élément, et la dimension du problème (`ndim`).

Regardez et analyser ce script, vous identifierez aisément les résultats de cours que nous avons utilisés pour effectuer les calculs à la main.

Vous pouvez modifier ce script pour faire afficher les matrices élémentaires si vous souhaitez utiliser ce programme pour vérifier vos calculs à la main.

---

[4] *Correction : voir un des scripts "statique" réellement proposé*

Les autres scripts proposés concernent:

«*poutre_ke*»	pour le calcul statique des portiques, EF poutre (1D, 2D et ...)
«*barre_keme*»	Pour les calcul dynamique des treillis et portiques
«*poutre_keme*»	(calcul des matrices masse et raideur)
«*barre_stress*»	Pour le calcul des efforts intérieurs
«*poutre_stress*»	(contraintes sur les éléments)
«*T3_ep*» «*T3_stress*»	EF triangle à 3 noeuds pour l'élasticité plane en statique
«*Q4_ep*» «*Q4_stress*»	EF quadrilatère à 4 noeuds pour l'élasticité plane en statique
«*Q4_th*»	EF quadrilatère à 4 noeuds de thermique stationnaire

Vous devez étudier ces scripts avant de les utiliser.

La prise en compte des conditions aux limites

```
F = F + Fg;
ir = 0;
for i=1:nddlt
    if ( Ncl(i) == 1 )                  %déplacements imposés dans F
          F = F - K(:,i)*Vcl(i); ir=ir+1;
    end
end
for i=nddlt:-1:1
    if ( Ncl(i) == 1 )
          Kr(ir,:) = K(i,:);            %pour le calcul des réactions
          Kr(:,i) = []; R(ir,1) = -F(i); ir = ir-1;
          K(i,:) = []; K(:,i) = [];     %suppression ligne colonne dans K
          F(i)=[];                      %suppression ligne dans F
    end
end
```

On commence par sommer les charges nodales données (variable globale «*F*») avec le vecteur des charges réparties sur les éléments (variable «*Fg*»).

L'algorithme proposé ici est basé sur une résolution par bloc du système d'équations, c'est la méthode vue en cours et utilisée en TD.

$$\begin{bmatrix} [K_{11}] & [K_{12}] \\ [K_{21}] & [K_{22}] \end{bmatrix} \begin{Bmatrix} \{U_i\} \\ \{U_d\} \end{Bmatrix} = \begin{Bmatrix} \{F_d\} \\ \{F_i\} \end{Bmatrix}$$

Le premier bloc d'équations nous donne le vecteur des déplacements nodaux inconnus:

$$\{U_i\} = [K_{11}]^{-1}\{\{F_d\} - [K_{12}]\{U_d\}\}$$ C'est le système réduit

En reportant dans le second nous obtenons le vecteur des efforts de liaison inconnus:

$$\{F_i\} = \left[K_{22} - K_{21}K_{11}^{-1}K_{12}\right]\{U_d\} + \left[K_{21}K_{11}^{-1}\right]\{F_d\}$$

Dans le script nous modifions dans un premier temps le vecteur du chargement en tenant compte des déplacements imposés. Notez que cette opération est inutile si tous les déplacements imposés sont nuls.

Variables globales utilisées

Nddlt : nombre de degré de liberté total
Ncl : vecteur de dimension nddlt qui vaut 1 si le ddl est imposé
Vcl : vecteur de dimension nddlt valeurs des déplacements imposés

La boucle suivante remonte le système d'équations pour supprimer les lignes et les colonnes de K et les lignes de F permettant ainsi d'obtenir le système réduit (ou premier bloc du système global). Avant cette opération nous stockons dans une matrice Kr les éléments de K qui seront utiles pour calculer les réactions aux appuis (second bloc du système global).

Comme vous pouvez le voir l'intérêt de MATLAB est de pouvoir manipuler globalement les matrices, ce qui nous donne une programmation simple et efficace.

Dans le script «*vibrations*» il suffit d'éliminer les lignes et colonnes de K et M.

Il existe d'autres méthodes pour prendre en compte les conditions aux limites en déplacement sans avoir à réduire le système d'équations ce qui évite ensuite d'avoir à réintroduire les déplacements imposés dans le vecteur solution. La plus efficace est celle du terme unité sur la diagonale voir (*G. Dhatt - G. Touzot & E Lefrancois : méthode des éléments finis. Hermes Lavoisier, 2005*).

Exercice :

Étudier l'algorithme de la méthode du terme unité sur la diagonale proposé par *Dhatt & Touzot*, puis programmer le script correspondant.

Correction : regardez le script statique.

La résolution *Sol =K\ F en statique*

Ayant une matrice symétrique définie positive la méthode utilisée est celle de Choleski.

Attention si vous laissez des modes rigides le programme affichera un message d'erreur indiquant que la matrice est singulière, c'est donc à vous de modifier le jeu de données pour éliminer les modes rigides de la structure.

Pour calculer les «*n*» premières fréquences et modes propres de vibrations d'une structure nous avons les commandes MATLAB suivantes :

```
[modes,omega] = eigs(K,M,n,'sm');
f = sqrt(diag(omega))/(2*pi);
```

Le script se termine en replaçant les valeurs imposées dans le vecteur *U* et en calculant les réactions (flux) inconnus correspondants.

L'étude attentive de ces scripts vous permettra d'approfondir vos connaissances en EF et de vous perfectionner dans l'utilisation de *MATLAB*.

Sauf à développer la bibliothèque des programmes ou éléments finis proposés vous n'aurez pas à modifier ces scripts pour utiliser *MEFlab*.

Description des scripts de données

Ces scripts se déroulent en trois temps :

Définition des données du problème : caractéristiques de la structure
Calcul résolution du problème
Post-traitement calculs complémentaires & analyse des résultats (graphiques)

La mise en données se termine logiquement par une représentation graphique du maillage script «*plotstr*» dans les premiers scripts on vous demande si vous souhaitez poursuivre le calcul, avant l'appel du script de résolution du problème.

Vous pouvez supprimer ce test pour gagner du temps lorsque vous développerez vos scripts de données.

Ce sont les lignes suivantes :

```
disp('Les variables globales sont initialisees');
disp('Fin de lecture des donnees');
% trace du maillage pour validation des donnees
plotstr
reponse = input('Voulez-vous continuer? O/N [O]: ','s');
if isempty(reponse) | reponse =='O'
    U = zeros(nddlt,1);
    R = zeros(nddlt,1);
    [U(:,1),R(:,1)] = statiqueUR;   % ----- resolution du probleme
```

Post traitement

Le post traitement effectue une mise en forme des résultats (formats d'impression) et liste les variables à afficher.

Vous trouverez quasiment systématiquement

Tracé de la déformée fonction «*plotdef*» en mécanique, représentation du champ calculé aux nœuds pour un problème de physique «*plot_therm*». Éventuellement un listing des champs nodaux (déplacements, températures etc.) et des flux inconnus (efforts nodaux, quantité de chaleur, etc.).

En dynamique : tracé des modes de vibrations

En statique le calcul des contraintes sur les éléments est systématiquement effectué car comme vous l'avez vu en cours l'analyse de la discontinuité sur ces résultats nous donne une information sur la qualité de notre modèle numérique (discrétisation).

Exemple, la fonction «*barre_stress*» affiche l'effort normal calculé dans les éléments barre.

Comparaison avec une solution analytique

Lorsqu'elle existe il peut être intéressant de programmer la solution analytique du problème pour la comparer aux résultats du modèle éléments finis, lorsque nous utilisons un script pour ces comparaisons nous l'avons mis dans le répertoire **[Sol_analytique]**

Il est tout aussi simple de compléter les lignes du script de données pour programmer, à partir des résultats du modèle éléments finis, les calculs que vous souhaitez effectuer pour répondre aux objectifs de votre étude.

Ci-dessous les lignes de post-traitement d'un script treillis

```
                                        %----- format d'impression des vecteurs
form =' %8.3e   %8.3e   %8.3e  '; format = [form(1:8*nddln),' \n'];
disp(' ');disp('------- déplacements nodaux sur (x,y,z) ----------');
fprintf(format,U)
plotdef(U)
                                        %-----  post-traitement
disp(' ');disp('------- Efforts aux appuis  ----------');
fprintf(format,R(:,1));
[Rx,Ry,Rz] = feval('resultante',R);         %----- résultantes et réactions

disp(' ');
fprintf('La résultante des charges nodales     en (x,y,z) est : %8.3e    %8.3e    %8.3e
\n',Fx,Fy,Fz);
fprintf('La résultante des charges réparties  en (x,y,z) est : %8.3e   %8.3e   %8.3e
\n',-Rx-Fx,-Ry-Fy,-Rz-Fz);
fprintf('La résultante des efforts aux appuis en (x,y,z) est : %8.3e    %8.3e    %8.3e
\n',Rx,Ry,Rz);

disp(' ');disp('------- Contraintes sur les éléments ----------');
for iel=1:nelt                          %----- boucle sur les éléments
    loce=[]; for i=1:nnode loce=[loce,(Connec(iel,i)-1)*nddln+[1:nddln]];end
    Ue=U(loce);
    feval('barre_stress',iel,Ue);
end
```

Données du problème

Cette première partie du script sert à définir les valeurs des variables globales, du modèle éléments finis. Il vous faudra modifier ces lignes pour les adapter à la nouvelle structure que vous souhaitez étudier.

Regardons le script du treillis étudié dans le chapitre de cours.

treillis_cours.m

```
Coord=[ 0 , 0 ; ...          % définition des coordonnées des nœuds X , Y
        2*h , 0 ; ...
        h , h ];
[nnod,ndim]=size(Coord);     % dimension du tableau
nddln=2;  nddlt=nddln*nnod;  % Nombre de ddl par nœuds et total

Connec=[ 1 , 2 ; ...         % définition de la matrice de connectivité
         1 , 3 ; ...
         2 , 3 ];
[nelt,nnode]=size(Connec);   % dimension du tableau

Typel = 'barre_ke';          % définition du type des éléments
for i=1:nelt
    Typel = str2mat('barre_ke',Typel);
end
                             % définition des caractéristiques mécaniques
Nprop=[1;1;1];               % pour chaque élément N° de la propriété
Prop=[ 100*sqrt(2) 0 0];     % tableau des différentes valeurs de ES fx fy

                             % définition des CL en déplacement
CL=[ 1 , 1 , 1 ; ...         % N° du nœud, type sur u et v (1 ddl imposé ,
     2 , 0 , 1 ];                                        0 ddl libre)
Ncl=zeros(1,nddlt);
Vcl=zeros(1,nddlt);          % Valeurs des déplacements imposés
%Vcl(2)=1;                   % à utiliser pour imposer une valeur non nulle

for i=1:size(CL,1)           % Variables globales associées
   for j=1:nddln
       if CL(i,1+j)==1 Ncl(1,(CL(i,1)-1)*nddln+j)=1; end
   end
end
                             % définition des charges nodales
Charg=[ 3  40.  0 ];         %  N° du noeud , Fx , Fy
F=zeros(nddlt,1);            %  vecteur sollicitation
for iclf=1:size(Charg,1)
    noeud=Charg(iclf,1);
    for i=1:nddln
       F((noeud-1)*nddln+i)=F((noeud-1)*nddln+i) + Charg(iclf,i+1);
    end
end
```

Il est simple de modifier cette partie du script pour créer un nouveau jeu de données.

On peut programmer tous les calculs si on souhaite paramétrer la discrétisation de la structure pour mener une étude de convergence, c'est ce qui est fait dans d'autres scripts de données.

Un regard sur ces scripts vous permettra de voir différentes options pour gagner ensuite du temps lors de la réalisation de vos propres scripts.

Bilan

Si vous avez lu en détail cette présentation et traité les différents exercices proposés vous avez compris le fonctionnement de l'application *MEFlab*, et le principe d'utilisation des différents scripts proposés.

A vous de jouer.

Si vous effectuez des développements sous *MEFlab* n'hésitez pas à m'en faire part pour les mettre en ligne et les partager.