Ecole Normale Supérieure de Cachan
Rapport de l'étude du flambage d'Euler de poutres droites
Département de Génie Mécanique

Etude du flambage d'Euler de poutres droites

Bertin Morgan
Rivoire Pascal

Cachan, le 22 février 2009

Table des matières

Table des figures

Première partie

Introduction

Le flambage ou flambement est un phénomène d'instabilité d'un matériau, qui soumis à une force de compression, a tendance à fléchir et à se déformer dans une direction perpendiculaire à la force de compression. L'objectif de notre étude et donc de programmer la recherche de charges critiques de flambage (charge à partir desquelles la structure flambe). Cette étude permet donc de connaître certaines contraintes de mise en place de structures réelles. On pourra citer par exemple les rails de chemin de fer et les chateaux d'eau.
Vous pourrez allez consulter en annexe le programme en langage C++ et le programme en fonctionnement.

Deuxième partie

Construction du problème

1 Les données du problèmes

La première partie de notre programme consiste à récuperer les différentes données qui seront utiles à la réalisation du problème. En effet certaines données peuvent être assimilées à des caractéristiques propres au problème tel que le matériau, la forme de la poutre ou encore le type des liaisons qui contraignent la poutre. Cette première partie permet donc de recueillir tous ses renseignements.

1.1 les caractéristiques techniques du matériau

La première caractéristique importante est l'élasticité du matériau, caractérisé par son module d'élasticité $E(Mpa)$ Ceci est réaliser par le sous programme *donnes*

1.2 les caractéristiques géométriques de la poutre

En effet le flambage est directement lié au moment quadratique de la poutre à une section droite donnée. De plus nous souhaitons considérer une poutre qui ne posséde pas une section constante pour pouvoir étudier des formes plus complexes, nous avons donc choisi de prendre une poutre dont la section dépend linéairement de la hauteur. Mais elle peut tout aussi avoir une section constante, c'est l'utilisateur qui décide de la forme de sa poutre avec la contrainte de forme que l'on a choisi.

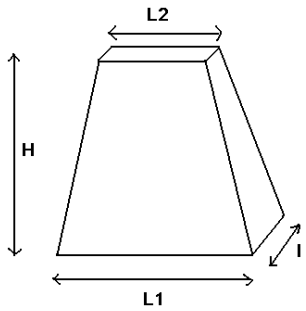

FIG. 1 – Poutre

Nous obtenons donc la relation suivantes pour l'aire à une section droite "i" donnée :

$$\Gamma_i = (L_2 + \frac{H - X}{X} \cdot (L_1 - L_2)) \cdot L \tag{1}$$

Une fois que nous avons obtenu l'aire, il nous faut calculer le moment quadrique pour chaque partie "i" de notre poutre, il vient donc :

$$I_i = \frac{\Gamma_i((\frac{\Gamma_i}{L})^2 + L^2)}{3} \tag{2}$$

Nous avons ensuite décidé pour des raisons de présentation de présenter l'aire de la section i : Γ_i et le moment quadratique de cette même section I_i sous la forme d'un tableau à deux colonnes.

1.3 les Différentes conditions aux limites possibles

Nous demandons à l'utilisateur de définir son problème, c'est à dire de quelle manière la poutre est solicitée, ainsi nous avons choisi quatre types de conditions limites :

CAS 1	Bas de la poutre	Encastrement
	Haut de la poutre	Néant
CAS 2	Bas de la poutre	Rotule
	Haut de la poutre	Rotule mobile en translation
CAS 3	Bas de la poutre	Encastrement
	Haut de la poutre	Rotule mobile en translation
CAS 4	Bas de la poutre	Encastrement
	Haut de la poutre	Encastrement mobile en translation

1.4 Le pas de discrétisation

Nous allons transformer notre problème continu en un problème discret. Pour cela il est nécessaire de discrétiser la poutre en N éléments. Nous demandons donc à l'utilisateur de définir le pas de discrétisation du problème et donc de choisir la précision des calculs de flambage.

1.5 La précision du schéma itératif par la méthode de puissance inverse

Nous reviendrons dans la suite de notre étude sur se sujet, mais nous demandons à l'utilisateur de saisir la précision de résultat qu'il souhaite obtenir. On lui donc de rentrer un ϵ qui pourra servir à arrêter le schéma itératif de la fin du programme.

2 Construction du problème discret

2.1 Construction des la matrice de K et KG

2.1.1 Présentation du problème

La résolution du problème de flambage est liée à la résolution de l'équation :

$$\int_0^L (P \cdot \frac{d\delta_\nu}{dx} \cdot \frac{d\nu}{dx} + E \cdot I \cdot \frac{d^2\nu}{dx^2} \cdot \frac{d^2\delta_\nu}{dx^2})dx = 0 \tag{3}$$

Pour cela, nous allons discrétiser la poutre en N éléments et on cherchera une résolution approchée sur chaque élément. La flèche sur un élément est definie par :

$$v_e(x) = v.g_1(x) + \theta_1 \cdot h_1(x) + v_2 \cdot g_2(x) + \theta_2 \cdot h_2(x) \tag{4}$$

avecv_1, v_2 valeurs de la flèche et θ_1 et θ_2 ses dérivées aux extrémités de l'élément i. Cette fonction v_e peut également s'écrire sous la forme :

$$v_e(x) = \Phi^t \cdot V_E \tag{5}$$

avec

$$V_E = \begin{pmatrix} v_1 \\ \theta 1 \\ v_2 \\ \theta 2 \end{pmatrix}^t$$

On définit alors la matrice de raideur de flexion sur l'élément i :

$$K = \int_{x_i}^{x_{i+1}} \delta V_E^t \cdot E \cdot I \cdot (\frac{d^2\Phi^t}{dx^2} \cdot \frac{d^2\Phi}{dx^2}) \cdot V_E dx$$

La matrice de raideur de flexion géométrique sur l'élément i :

$$KG = \int_{x_i}^{x_{i+1}} \delta V_E^t \cdot (\frac{d\Phi^t}{dx} \cdot \frac{d\Phi}{dx}) \cdot V_E dx$$

2.1.2 Sous programme *raideur*

Pour faire calculer K et KG par le programme, on a rentré termes à termes leurs coefficients, fonctions dépendant de i. Cela correspond à 16 coefficients définis par la relation

$$v_e(x) = \Phi^t \cdot V_E$$

On obtient ainsi les matrices de K et KG sur l'élément i. Ensuite, il s'agit de faire calculer K et KG sur chaque sous intervalle et de les assembler dans une matrice de rigidité globale. On obtient des matrices de taille 2*p+1.

Présentation de l'algorithme d'assemblage :

Nous avons décidé de réaliser les deux matrices en une seule boucle. Ainsi, aprés avoir calculé les intégrales avec un logiciel de calcul formel, nous rentrons ces intégrales qui sont fonctions de l'abcisse. A chaque boucle, on incrémente i, qui est le pas de discrétisation choisi par l'utilisateur, ce qui nous permet de calculer les termes des matrices K et KG sur chaque partie discrétisée de la poutre. Dans cette boucle il y a ensuite les boucles simple de construction des matrices qui pour une ligne k et une colonne j rentrent les valeurs calculées précedemment dans les matrices de rigidité K et KG. Enfin, il est important de visualiser les résultats.

2.2 Simplification des matrices grâce aux conditions initiales

Nous allons utiliser les conditions aux limites pour réduire la dimension de ces matrices. En effet, à cause des conditions aux extrémités de la poutre, elles ont des lignes et de colonnes inutiles, or on voudra par la suite inverser ces matrices, et on ne peut pas inverser une matrice ayant des lignes ou colonnes nulles. Pour cela,on utilise les sous programmes : EL, RRM, ERM, EEM qui suppriment les ligne nulles en haut, en bas, et les colonnes nulles à gauche, et à droite.

2.2.1 Sous programme EL

EL signifie, Encastrement-Libre. Ce sous programme permet de réduire la matrice de rigidé pour le premier cas étudié. Donc, pour ce cas, le vecteur déplacement s'écrit de la façon suivante :

$$V = \begin{pmatrix} v_1 = 0 \\ \theta_1 = 0 \\ v_2 \\ \theta_2 \\ . \\ . \\ v_n \\ \theta_n \end{pmatrix}$$

Il n'est donc pas utile pour le calcul de conserver les lignes 1 et 2 et les colonnes 1 et 2 des matrices de rigidité K et K_G, le sous programme permet de faire cela. La première boucle permet de conserver les termes des matrices strictement supérieur à 1 pour supprimer les lignes et les colonnes,

$$K = \begin{pmatrix} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & . & . & \bullet \\ \bullet & \bullet & K_{22} & . & . & K_{2n} \\ \bullet & \bullet & K_{32} & . & . & K_{3n} \\ \bullet & \bullet & . & . & . & . \\ \bullet & \bullet & . & . & . & . \\ \bullet & \bullet & K_{(n-1)2} & . & . & K_{(n-1)n} \\ \bullet & \bullet & K_{n2} & . & . & K_{nn} \end{pmatrix}$$

la deuxième boucle permettant de réinitialiser la position des termes à partir de i=j=0 pour une normalisation de l'écriture des matrices dans le programme. En effet nous avons décidé d'écrire les matrices à partir d'une incrémentation i=j=0.

$$K = \begin{pmatrix} K_{22} & . & . & K_{2n} \\ K_{32} & . & . & K_{3n} \\ . & . & . & . \\ . & . & . & . \\ K_{(n-1)2} & . & . & K_{(n-1)n} \\ K_{(n)2} & . & . & K_{nn} \end{pmatrix} = \begin{pmatrix} K_{00} & . & . & K_{0(n-2)} \\ K_{10} & . & . & K_{1(n-2)} \\ . & . & . & . \\ . & . & . & . \\ K_{(n-3)0} & . & . & K_{(n-3)(n-2)} \\ K_{(n-2)0} & . & . & K_{(n-2)(n-2)} \end{pmatrix}$$

2.2.2 Sous programme RRM

RRM signifie, Rotule-rotule mobile. Ce sous programme permet de réduire la matrice de rigidé pour le second cas étudié. Donc, pour ce cas, le vecteur déplacement s'écrit de la façon suivante :

$$V = \begin{pmatrix} v_1 = 0 \\ \theta_1 \\ v_2 \\ \theta_2 \\ . \\ . \\ v_n = 0 \\ \theta_n \end{pmatrix}$$

Il n'est donc pas utile pour le calcul de conserver les lignes 1 et n-1 et les colonnes 1 et n-1 des matrices de rigidité K et K_G, le sous programme permet de faire cela. La première boucle permet de conserver les termes des matrices strictement supérieurs à 0. La seconde boucle permet de supprimer les termes de la ligne n-1 et de la colonne n-1.

$$K = \begin{pmatrix} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & K_{11} & K_{12} & . & K_{1(n-2)} & \bullet & K_{1n} \\ \bullet & K_{21} & K_{22} & . & K_{2(n-2)} & \bullet & K_{2n} \\ \bullet & . & . & . & . & \bullet & . \\ \bullet & K_{(n-2)1} & . & . & K_{(n-2)(n-2)} & \bullet & K_{(n-2)n} \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & K_{n1} & K_{n2} & . & K_{n(n-2)} & \bullet & K_{nn} \end{pmatrix}$$

La partie du programme suivante permet de normaliser les parties de matrice obtenues à 0. La première partie décale les termes pour $i \in [1..n-2]$ et $j \in [1..n-2]$ pour décaler les termes à 0. Ensuite on décale les termes pour $j = n$ $i \in [1..n-2]$ en $j = n-2$ et les termes pour $i = n$ et $j \in [1..n-2]$ en $i = n-2$. Enfin on décale le terme K_{nn} en $K_{(n-2)(n-2)}$.

$$K = \begin{pmatrix} K_{11} & . & . & K_{1n} \\ K_{21} & . & . & K_{2n} \\ . & . & . & . \\ . & . & . & . \\ K_{(n-2)1} & . & . & K_{(n-2)n} \\ K_{n1} & . & . & K_{nn} \end{pmatrix} = \begin{pmatrix} K_{00} & . & . & K_{0(n-2)} \\ K_{10} & . & . & K_{1(n-2)} \\ . & . & . & . \\ . & . & . & . \\ K_{(n-3)0} & . & . & K_{(n-3)(n-2)} \\ K_{(n-2)0} & . & . & K_{(n-2)(n-2)} \end{pmatrix}$$

2.2.3 Sous programme ERM

ERM signifie, Encastrement-rotule mobile. Ce sous programme permet de réduire la matrice de rigidé pour le troisième cas étudié. Donc, pour ce cas, le vecteur déplacement s'écrit de la façon suivante :

$$V = \begin{pmatrix} v_1 = 0 \\ \theta_1 = 0 \\ v_2 \\ \theta_2 \\ . \\ . \\ v_n = 0 \\ \theta_n \end{pmatrix}$$

Il n'est donc pas utile pour le calcul de conserver les lignes 1,2 et n-1 et les colonnes 1,2 et n-1 des matrices de rigidité K et K_G, le sous programme permet de faire cela. La première boucle permet de conserver les termes des matrices strictement supérieur à 0. La seconde boucle permet de supprimer les termes de la ligne n-1 et de la colonne n-1.

$$K = \begin{pmatrix} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & K_{22} & . & K_{2(n-2)} & \bullet & K_{2n} \\ \bullet & \bullet & . & . & . & \bullet & . \\ \bullet & \bullet & K_{(n-2)2} & . & K_{(n-2)(n-2)} & \bullet & K_{(n-2)n} \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & K_{n2} & . & K_{n(n-2)} & \bullet & K_{nn} \end{pmatrix}$$

La partie du programme suivante permet de normaliser les parties de matrice obtenues à 0. La première partie décale les termes pour $i \in [2..n-2]$ et $j \in [2..n-2]$ pour décaler les termes à 0. Ensuite on décale les termes pour $j = n$ $i \in [2..n-2]$ en $j = n-2$ et les termes pour $i = n$ et $j \in [2..n-2]$ en $i = n-2$. Enfin on décale le terme K_{nn} en $K_{(n-3)(n-3)}$.

$$K = \begin{pmatrix} K_{22} & . & . & K_{2n} \\ K_{32} & . & . & K_{3n} \\ . & . & . & . \\ . & . & . & . \\ K_{(n-2)2} & . & . & K_{(n-2)n} \\ K_{n2} & . & . & K_{nn} \end{pmatrix} = \begin{pmatrix} K_{00} & . & . & K_{0(n-3)} \\ K_{10} & . & . & K_{1(n-3)} \\ . & . & . & . \\ . & . & . & . \\ K_{(n-4)0} & . & . & K_{(n-4)(n-3)} \\ K_{(n-3)0} & . & . & K_{(n-3)(n-3)} \end{pmatrix}$$

2.2.4 Sous programme EEM

EEM signifie, Encastrement-encastrement mobile. Ce sous programme permet de réduire la matrice de rigidé pour le troisième cas étudié. Donc pour ce cas, le vecteur déplacement s'écrit de la façon suivante :

$$V = \begin{pmatrix} v_1 = 0 \\ \theta_1 = 0 \\ v_2 \\ \theta_2 \\ . \\ . \\ v_n \\ \theta_n = 0 \end{pmatrix}$$

Il n'est donc pas utile pour le calcul de conserver les lignes 1,2 et n et les colonnes 1,2 et n des matrices de rigidité K et K_G, le sous programme permet de faire cela. La première boucle permet de conserver les termes des matrices strictement supérieurs à 0. La seconde boucle permet de supprimer les termes de la ligne n-1 et de la colonne n-1.

$$K = \begin{pmatrix} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & K_{22} & . & K_{2(n-2)} & K_{2(n-1)} & \bullet \\ \bullet & \bullet & . & . & . & . & \bullet \\ \bullet & \bullet & K_{(n-2)2} & . & K_{(n-2)(n-2)} & K_{(n-2)(n-1)} & \bullet \\ \bullet & \bullet & K_{(n-1)2} & . & K_{(n-1)(n-2)} & K_{(n-1)(n-1)} & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \end{pmatrix}$$

La partie du programme suivante permet de normaliser les parties de matrice obtenues à 0. La première partie décale les termes pour $i \in [2..n-2]$ et $j \in [2..n-2]$pour décaler les termes à 0. Ensuite on décale les termes pour $j = n$ et $i \in [2..n-2]$ en $j = n-2$ et les termes pour $i = n$ et $j \in [2..n-2]$ en $i = n-2$. Enfin on décale le terme K_{nn} en $K_{(n-3)(n-3)}$.

$$K = \begin{pmatrix} K_{22} & . & . & K_{2n} \\ K_{32} & . & . & K_{3n} \\ . & . & . & . \\ . & . & . & . \\ K_{(n-2)2} & . & . & K_{(n-2)n} \\ K_{(n-1)2} & . & . & K_{(n-1)(n-1)} \end{pmatrix} = \begin{pmatrix} K_{00} & . & . & K_{0(n-3)} \\ K_{10} & . & . & K_{1(n-3)} \\ . & . & . & . \\ . & . & . & . \\ K_{(n-4)0} & . & . & K_{(n-4)(n-3)} \\ K_{(n-3)0} & . & . & K_{(n-3)(n-3)} \end{pmatrix}$$

2.3 Affichage des matrices simplifiées

On peut remarquer que les matrices simplifées en fonction des conditions limites n'ont pas la même taille. Nous avons décidé pour pallier à ce problème de créer une variable u qui selon sa valeur influera sur la taille de la matrice affichée. Donc dans le sous programme matrice, il y aura trois informations : le pas de discrétisation, la matrice à afficher et la variable u.

cas	taille	valeur de u
1	(n-3)	0
2	(n-3)	0
3	(n-4)	-1
4	(n-4)	-1

(6)

Pour ainsi obtenir une matrice de taille $2 \cdot p + u$.

3 Résolution du problème discret

3.1 Présentation

En faisant afficher les matrices K et K_G simplifiées grâce aux conditions limites, on se rend compte qu'elles sont symétriques, définies positives et à diagonale dominante. On peut donc utiliser la methode de décomposition de cholesky pour effectuer des calculs. Cela nous arrange d'autant plus que le problème

d'équations initial peut se mettre sous la forme :

$$(K + p \cdot K_G) \cdot V = 0$$

Ce qui s'écrit également :

$$(K_G^{-1} \cdot K + p \cdot ID) \cdot V = 0$$

ce qui revient à dire que $-p \cdot ID$ est valeur propre de $K_G^{-1} \cdot K$ associée au vecteur propre V. Nous allons donc chercher à calculer $K_G^{-1} \cdot K$, puis ses valeurs propres.

3.2 Construction des matrices de cholesky

3.2.1 Présentation

On décompose la matrice A de la manière suivante :

$$\underline{\underline{A}} = \underline{\underline{L}} \cdot \underline{\underline{L}}^T$$

Ainsi il est plus facile de calculer l'inverse de la matrice A, en effet une matrice triangulaire est plus facile à inverser qu'une matrice entière.

$$\underline{\underline{A}}^{-1} = (\underline{\underline{L}} \cdot \underline{\underline{L}}^T)^{-1} = \underline{\underline{L}}^{-T} \cdot \underline{\underline{L}}^{-1}$$

3.2.2 Relations de calcul

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2}$$

$$l_{ij} = a_{ji} - (\sum_{k=1}^{i-1} l_{ik} l_{jk}) \cdot \frac{1}{l_{ii}}$$

Le calcul des termes de $\underline{\underline{L}}$ se fait colonne par colonne, de haut en bas. Le nombre d'opérations est de $\frac{n^3}{3} + \frac{n^2}{2} + \frac{n}{6}$ soit environ la moitié du nombre d'opérations à effectuer par la méthode de Gauss.

3.2.3 Programme cholesky

Les données nécessaires à la réalisation de cette tâche sont le pas de discrétisation, la matrice $\underline{\underline{K_G}}$ et une matrice $\underline{\underline{L}}$. Le programme est construit sur deux boucles imbriquées pour contruire chaques termes de la matrice $\underline{\underline{L}}$. A l'intérieur de ces boucles, il y a deux parties distinctes, la première pour les termes de la diagonal et la seconde pour les termes en dessous. La particularité de ce programme est que pour respecter l'ordre de calcul de la relation de cholesky, pour chaque boucle i variant de 0 à $2p$ j varie de i à 0.

Enfin pour définir les deux parties nous avons mis en place la fonction if :

partie	cas
1	$i = j$
2	$i < j$

(7)

Ensuite dans chaque partie il faut vérifier si le calcul de la somme est possible, en effet si :
Partie 1 : pour $i = j$

$$(i - 1) < 1$$

$$L_{ij} = M_{ij}$$

$$(i - 1) == 1$$

$$L_{ij} = \sqrt{M_{ij} - L_{i1}^2}$$

$$(i-1) > 1$$

$$L_{ij} = \sqrt{M_{ij} - \sum_{k=1}^{i-1} l_{ik}^2}$$

Partie 2 : pour $i < j$

$$(i-1) < 1$$

$$L_{ij} = \frac{M_{ij}}{L_{ij}}$$

$$(i-1) == 1$$

$$L_{ij} = \frac{M_{ij} - L_{j1}^2}{L_{ij}}$$

$$(i-1) > 1$$

$$L_{ij} = M_{ij} - \frac{z}{L_{ij}}$$

En effet si on a $(i-1) < 1$ la somme ne se calcul pas.
Ce programme nous donne donc la matrice $\underline{\underline{\mathrm{L}}}$.
Il faut ensuite calculer sa transposée $\underline{\underline{\mathrm{L}}}^{-T}$.
Pour cela, nous allons utiliser le sous programme transpmat.

3.3 Programme de transposition

Pour transposer une matrice, il suffit de se rappeler que :

$$L_{ij}^T = L_{ji}$$

et de mettre cette relation dans une double boucle for pour faire calculer tous les éléments de la transposée.

Maintenant que l'on connait la matrice L et sa transposée, il nous faut calculer leur inverse.

3.4 Programme d'inversion de matrice triangulaire

Ce programme calcule l'inverse d'une matrice triangulaire par la méthode du pivot de Gauss. Dans ce programme, on a fait intervenir une variable temporaire pour faciliter les calculs. Une fois que l'on a inversé les matrices L et L^T , on peut calculer la matrice K_G^{-1}. Pour cela, il suffit de d'effectuer le produit matriciel : $\underline{\underline{\mathrm{L}}}^{-T} \cdot \underline{\underline{L}}^{-1}$. Pour cela, on utilise le programme produitmatrice.

3.5 Programme de calcul d'un produit matriciel

Ainsi, nous faisons calculer K_G^{-1} à partir de L et de sa transposée. Ensuite, nous réutilisons ce programme pour faire calculer $K_G^{-1} \cdot K$.

Il nous faut maintenant obtenir les valeurs propres de cette matrice. On va pour cela utiliser la méthode des puissances.

3.6 Calcul des valeurs propres

3.6.1 Présentation du problème

Nous avons obtenu précédement la relation :

$$(K_G^{-1} \cdot K + p \cdot ID) \cdot V = 0$$

On a donc à résoudre un problème du type

$$\underline{\underline{A}} \cdot \underline{x_k} = \lambda_k \cdot \underline{x_k}$$

où les x_k et λ_k sont les vecteurs et valeurs propres de A.
Ainsi dans notre projet :

$$\underline{\underline{A}} = \underline{\underline{K_G}}^{-1} \cdot \underline{\underline{K}}$$

Nous allons donc utiliser la méthode des puissances pour résoudre notre problème. Le principe est le suivant :

On déifnit une récurrence de la manière suivante :
- on choisit x_0 arbitrairement.
- on a la relation de récurrence suivante : $x_k = A \cdot x_{k-1}$

On réalise une mise en mémoire :

$$\underline{x_k} = \frac{\underline{x_z}}{\|\underline{x_k}\|}$$

On trouve alors :

$$\lambda_k = \frac{\underline{x_k}^{-T} \cdot \underline{\underline{A}} \cdot \underline{x_k}}{\underline{x_k}^{-T} \cdot \underline{x_k}}$$

Et si $\exists \epsilon$, $\| \lambda_k - \lambda_{k-1} < \epsilon$ alors λ_k est la plus grande valeur propre.

3.6.2 Sous programme valeurpropre

On initialise tout d'abord le vecteur $\mathrm{x}_0 = \begin{pmatrix} 1 \\ 0 \\ . \\ . \\ 0 \end{pmatrix}$

Ensuite on fait de même avec la première valeur propre, $\lambda_0 = 0$

Nous avons décider de faire fonctionner notre programme par une boucle while, car notre schéma itératif ne doit s'arrêté que si la précision demandé a été atteinte. Nous calculons ensuite le vecteur x_k dans deux boucles for(). Nous calculons le produit scalaire de deux vecteurs comme la somme des produits des coefficients correspondant deux à deux et ainsi normer le vecteur x_k. Pour simplifié le calcul de λ_k nous avons décidé de calculer le numérateur et le dénominateur séparement pour pouvoir calculer le produit scalaire de manière plus facile et faire ensuite le quotient de ces deux valeurs. On trouve ainsi λ_k On calcul ensuite l'erreur ϵ_{erreur} en valeur absolu , on incrémente la variable k et on test $\epsilon_{erreur} > \epsilon$. On trouve ainsi la plus grande valeur propre qui est la solution à notre problème.

4 Analyse des résultats

Ce projet nous a permi d'élaborer un raisonnement au travers d'un programme, c'est à dire créer différents sous programmes semblables à ceux que l'on a vu en TP et de leur donner un lien cohérent pour résoudre un problème concret, celui du flambage d'Euler des pourtres. Nous nous sommes servi d'informations aqcuises en cours de méthodes numériques. Pour revenir au programme, tous les modèles possèdent des defauts qui peuvent être diminués. Notre projet de par son concept, celui de discrétiser la

poutre en N élements permet une précision dans les calculs tout à fait accéptable. De plus le fait d'utiliser la méthode des puissances permet de minimiser les calculs et donc le temps utilisé par l'ordinateur pour résoudre notre problème. Enfin notre programme permet grâce à la méthode des puissances de régler la précision des calculs. Par contre nous avons un problème pour le calcul de l'inverse dans le second cas, en effet le premier terme de la matrice K_G est toujours nul.

4.1 Exemple mis en oeuvre

Nous avons choisi de tester la précision de notre programme en effectuant trois modélisations d'une même poutre de module de young 200 GPa dans trois configurations différentes. La poutre a les dimensions suivantes :

hauteur	1000
longueur	10
largeur	10

4.1.1 Cas 1 Encastrement-libre

essai	discrétisation	ϵ	itération	résultat
1	2	1.10^{-3}	2	$1.487.10^{-2}$
2	10	1.10^{-3}	2	$4{\cdot}10^{-2}$
1	20	1.10^{-3}	2	$8.1{\cdot}10^{-2}$
1	40	1.10^{-3}	2	$1.6{\cdot}10^{-1}$

essai	discrétisation	ϵ	itération	résultat
1	5	1.10^{-3}	2	$2.01{\cdot}10^{-2}$
2	5	1.10^{-5}	2	$2.01803{\cdot}10^{-2}$
1	5	1.10^{-10}	3	$2.01803{\cdot}10^{-2}$
1	5	1.10^{-20}	3	$2.01803{\cdot}10^{-2}$

4.1.2 Cas 2 Encastrement-rotule mobile

essai	discrétisation	ϵ	itération	résultat
1	2	1.10^{-3}	2	$8.09{\cdot}10^{-3}$
2	10	1.10^{-3}	2	$4.04{\cdot}10^{-2}$
3	20	1.10^{-3}	2	$4.11{\cdot}10^{-2}$
4	40	1.10^{-3}	2	$1.64{\cdot}10^{-1}$

essai	discrétisation	ϵ	itération	résultat
1	5	1.10^{-3}	2	$2.01{\cdot}10^{-2}$
2	5	1.10^{-5}	2	$2.01803{\cdot}10^{-2}$
3	5	1.10^{-10}	3	$2.01803{\cdot}10^{-2}$
4	5	1.10^{-20}	3	$2.01803{\cdot}10^{-2}$

4.1.3 Cas 3 Encastrement-encastrement mobile

essai	discrétisation	ϵ	itération	résultat
1	2	1.10^{-3}	2	$1.48{\cdot}10^{-2}$
2	10	1.10^{-3}	2	$1.48{\cdot}10^{-2}$
1	20	1.10^{-3}	2	$8{\cdot}10^{-2}$
1	40	1.10^{-3}	2	$1.64{\cdot}10^{-1}$

essai	discrétisation	ϵ	itération	résultat
1	5	1.10^{-3}	2	$2.013{\cdot}10^{-2}$
2	5	1.10^{-5}	2	$2.01804{\cdot}10^{-2}$
3	5	1.10^{-10}	13	$2.01803{\cdot}10^{-2}$
4	5	1.10^{-20}	13	$2.01803{\cdot}10^{-2}$

Attention la boucle while du main donne des résultats faux, il faut a chaque fois réinitialiser le programme.

On remarque q'avec nos études le programme converge trés vite.