REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

ECOLE NATIONALE POLYTECHNIQUE

Département de Génie Civil

Projet de Fin d'Etudes

Pour l'Obtention du Diplôme d'Ingénieur d'Etat en Génie Civil

THEME

ETUDE ET OPTIMISATION DES STRUCTURES SPATIALES EN TREILLIS

Etudié par :
Y.AOUES
A. HADJ HENNI

Proposé et dirigé par :
Dr K. SILHADI
Mr. B.MEZAZIGH

Remerciements

On ne saurait témoigner toute la reconnaissance qu'on doit à Monsieur le Professeur K.SCHITTKOWSKI (de l'université de Beyreuth) pour son aide. Qu'il trouve ici nous respectueux remerciements.

On tient à témoigner notre profonde gratitude à Monsieur K.SILHADI et Monsieur B.MEZAZIGH pour nous avoir encadrés tout au long de la préparation de ce projet .

On est également très reconnaissant à Messieurs les membres de jury d'avoir examiner ce travail.

Enfin, on remercie tous ceux qui ont contribué de prés ou de loin à la réalisation de ce travail.

A la mémoire de mon grand-père.

Anis Rédha Hadj Henni.

Sujet : Etude et optimisation des structures spatiales en treillis.

Résumé : L'optimisation des structures spatiales en treillis, en comportement linéaire, est réalisé par un programme qui met en ouvre la méthode Séquentielle quadratique et un code de calcul des treillis basé sur une formulation en éléments finis.
Le programme élaboré est testé pour de nombreuses structures, il constitue un outil d'assistance à la conception et au pré-dimensionnement des structures en barres.

Subject : Study and optimization of spatial strusses structures

Abstract : The optimization of the spatial strusses structurs, with a linear behaviour, is realize by a programm how implement the Sequential Quadratic Method and a study code of strusses based on the finite element method.
This programm has been tested for a lot of structures, it constitute an assistant to conception and engineering of strusses structures.

الموضوع : دراسة و تخفيض كتل الهياكل الشبكية الفضائية

ملخص تخفيض كتل الهياكل الشبكية الفضائية ذو التصرف الخطي، تحقق بإنجاز برنامج آلي استعمل الطريقة الرباعية التسلسلية و ذالك باذماج برنامج آلي لحساب الهياكل الشبكية بطريقة العناصر المحددة ، هذا البرنامج تم تجريبه على عدة هياكل شبكية،و يمثل أداة مساعدة في حساب أبعاد الهيكل الشبكي .

Sommaire

Introduction :

Les structures en treillis constituent une part importante de la construction métallique civile et industrielle. En plus de leur rôle de résistance, elles présentent la particularité d'être légères et de mise en œuvre simple.

Néanmoins, il est intéressent de penser à optimiser ce type de structure à l'aide d'un outil performant. En effet, le concepteur ; dans les limites de ses capacités humaines ; ne peut tester efficacement un grand nombre de configurations dimensionnelles (et géométriques) de manière à obtenir *la solution optimale* à son problème d'*optimisation* et de *conception*. Nous devons alors recourir à des outils de calcul performants et à des modèles mathématiques fiables qui assurent la *minimisation* du *poids* de la structure étudiée.

Notre présent travail, met en œuvre un tel outil d'optimisation à travers le développement d'un programme (OPTISTRUCT) de calcul et d'optimisation des structures en treillis.

La minimisation du poids de la structure est assuré par l'algorithme de la méthode séquentielle quadratique, sous les contraintes de résistance des barres à la traction et à la compression. L'optimisation fait appel, comme nous le verrons tout au long de l'algorithme, à un programme de calcul des structures spatiales en treillis basé sur la méthode des éléments finis.

Enfin nous étudierons des exemples de structures en deux et trois dimensions, à travers lesquelles on jugera de la puissance de notre programme qui constitue, comme nous le verrons, un outil d'aide à la conception et au pré-dimensionnement des structures en treillis.

INTRODUCTION :

Les techniques de calcul des structures ont connu ces vingt dernières années un développement considérable, motivé par les besoins des industries de pointe et soutenu par les progrès effectués dans le domaine des ordinateurs. Ainsi la méthode des éléments finis (MEF) est communément utilisée aujourd'hui pour l'analyse des structures dans de nombreux secteurs de l'industrie : Aérospatial, nucléaire, génie civil, construction navale, mécanique...

Par ailleurs, il est intéressant de remarquer que la MEF appliquée au calcul des structures est une technique récente, à caractère pluridisciplinaire, car elle met en œuvre les connaissances de trois disciplines de base :

- la mécanique des structures : élasticité, résistance des matériaux, dynamique, plasticité.

- l'analyse numérique : méthodes d'approximation, résolution des systèmes linéaires, problèmes aux valeurs propre...

- l'informatique appliquée : techniques de programmation, de développement et de maintenance de grands logiciels.

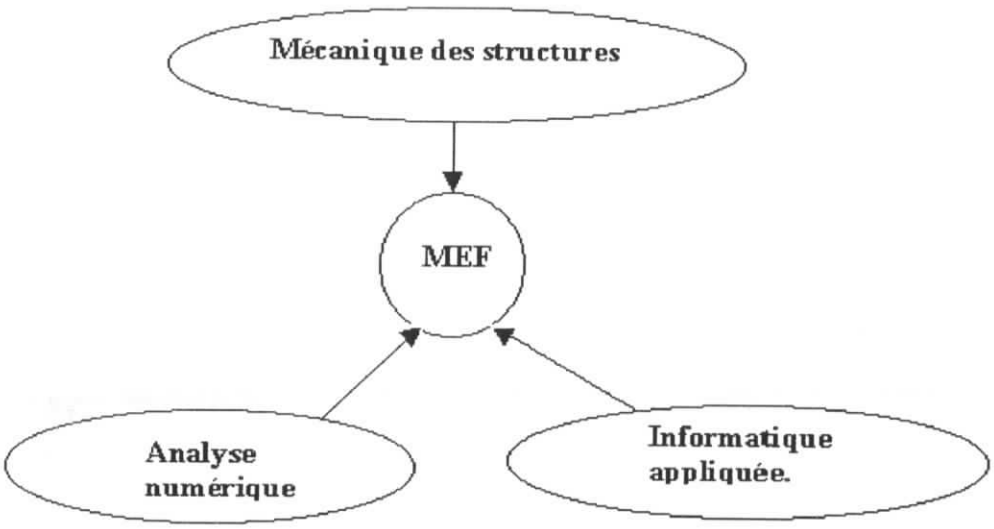


Figure I.1 : les domaines intervenant dans la méthode des éléments finis

Nous donnons dans ce qui suit un petit historique sur la méthode des éléments finis, ensuite les fondements de base de cette méthode qui sont les équations de la mécanique des solides, les méthodes d'approximation et l'approche énergétique des structures, afin de formuler les matrices élémentaires qui sont l'outil de calcul des structures discrétisées, en nous intéressant plus particulièrement au cas des structures en treillis.

I.1. HISTORIQUE :

Les bases de la MEF reposent d'une part sur la formulation énergétique de la mécanique des structures et d'autres part sur les méthodes d'approximation. En ce qui concerne les théorèmes énergétiques de l'élasticité leur formulation a été effectuée au XIX siècle ; en 1819 *NAVIER* définit une méthode d'étude des systèmes hyperstatiques basée sur l'application des conditions d'équilibres et de compatibilité, puis *MAXEWLL* en 1864 et *CASTIGLIANO* en 1878 établissent de façon complètes les théorèmes de l'énergie.

C'est au début du 20 ^{éme} siècle que des résultats fondamentaux dans le domaine des méthodes d'approximations sous l'impulsion de *RITZ* 1908 et *GALERKIN* en 1915, puis en 1943 *COURANT* établit les bases de la MEF. En donnant les premiers éléments pour la résolution de certains problèmes des milieux continus en effectuant une discrétisation spatiale du domaine, en utilisant les méthodes d'approximation variationnelles.

Dans les années 50, l'industrie de l'aéronautique a contribué au développement des méthodes matricielles permettant de traiter les problèmes des structures assez complexes avec les calculateurs disponibles a l'époque. Parmi les contributions les plus importantes on peut citer *LEVY* en 1947 pour la méthode des forces. En 1955 *TURNER* et *CLOUGH* présentent une approche unifiée de la méthode des déplacements et l'introduction du concept de l'élément fini, ces publications particulièrement importantes représentent véritablement le début de la MEF comme technique de calcul des structures complexes.

A partir de cette date, la MEF a connu un développement intense sous l'impulsion de l'industrie aérospatiale et grâce aussi a la disponibilité des premiers ordinateurs. La méthode des déplacements va être choisie de façon universelle comme technique de résolution matricielle de préférence a la méthode des forces. Par ailleurs le domaine d'application de la MEF se limite au début à la statique linéaire, puis s'étend progressivement a la dynamique linéaire, au flambement linéarisé, et l'analyse non linéaire.

Cet historique serrait incomplet si l'on omettait de mentionner le développement de programmes généraux et de logiciels d'analyse et de calcul des structures par la MEF (SAP2000, ROBO, ANSYS...)

I.2. Fondements de la méthode des éléments finis :

I.2.1. Rappels des équations de la mécanique des solides :

Le problème de la mécanique des solides revient à déterminer les deux champs inconnus : le champ de déplacements u_i et de contraintes σ_{ij} sous l'effet des forces de volume f_i et forces de surface ϕ_i. Le phénomène de déformation du corps solide est régi par des équations de champ aux dérivées partielles avec des conditions mixtes aux limites.

a) Relations déformations –déplacements ou relations cinématiques [1] :

$$\varepsilon_{ij}=\frac{1}{2}(u_{i,j}+u_{j,i}) \tag{1}$$

tels que :

u_i : composantes des déplacements.

ε_{ij} : composantes du tenseur des déformations.

$u_{i,j}$ dérivée partielle du déplacement u_i par rapport à la coordonnée x_j .

on peut écrire aussi :

$$\varepsilon_{ij}=\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}+\frac{\partial u_j}{\partial x_i}\right) \tag{2}$$

b) Equations d'équilibre :

$$\frac{\partial \sigma_{ij}}{\partial x_i}+f_j=0 \tag{3}$$

$$\sigma_{ij}=\sigma_{ji} \tag{4}$$

telles que :

σ_{ij} :composantes du tenseur des contraintes.

f_i : composantes des forces de volume.

c) Equations de compatibilité :

$$\varepsilon_{ij,kl}+\varepsilon_{kl,ij}-\varepsilon_{ik,jl}-\varepsilon_{jl,ik}=0 \tag{5}$$

$\varepsilon_{ij,kl}$ dérivée seconde des composantes du tenseur des déformations par rapport aux coordonnées x_k et x_l .

R*emarque* : ces équations sont les conditions d'intégrabilité des déformations qui assurent l'unicité des déplacements mais elles sont obligatoires que dans le cas de la résolution des problèmes d'élasticité en fonction des contraintes.

d) Relations contraintes-déformations ou relations d'élasticité :
dans le cas des matériaux à comportement élastique et linéaire, on a de façon générale :

$$\sigma_{ij} = C_{ijkl}\ \varepsilon_{kl} \tag{6}$$

C_{ijkl} : composantes du tenseur de l'élasticité.

Dans le cas de l'élasticité unidimensionnelle on obtient la relation de *HOOK* :

$$\sigma = E.\varepsilon \tag{7}$$

E : module de Young

e) Conditions aux limites cinématiques : sur une partie $S_u \in S$, des déplacements $\overline{u_i}$ sont imposés et qu'aucune composante du tenseur des contraintes σ_{ij} n'est imposée.(figure I.1)

$$u_i = \overline{u_i} \qquad \forall\ M \in S_u \tag{8}$$

tels que :
S : surface extérieure du solide.
$\overline{u_i}$: déplacement imposé.
M : un point dans le domaine du solide.

f) Conditions aux limites de type équilibre: sur la partie complémentaire S_σ de S des efforts extérieurs ϕ_i sont imposés et qu'aucun déplacement u_i n'est imposé. (figure I.1)

$$\sigma_{ij}.n_j = \phi_i \qquad \forall M \in S_\sigma \tag{9}$$

n_j : composantes de la normale unitaire à S
ϕ_i : composantes des forces de surface.

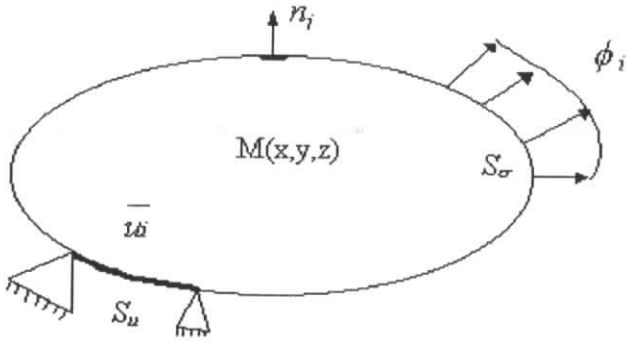


Figure I.2 : représentation des forces de surface sur une structure

Pour résoudre un problème d'élasticité, deux démarches sont envisageables suivant que l'on formule complètement le problème en fonction de l'un des champs inconnus, soit le champ de déplacements u_i, soit le champ de contraintes σ_{ij}. La première approche en fonction des déplacements est appelée *approche cinématique*, l'autre approche en fonction des contraintes est appelée approche équilibre.

1) Champ de déplacements cinématiquement admissible :

On appelle champ de déplacements cinématiquement admissible tout champ $u_i(M)$ satisfaisant [1]:

- les conditions de continuité tel que le champ $u_i(M)$ est de classe :
  - C^0 pour les problèmes de l'élasticité.
  - C^1 pour les problèmes de poutres, plaques.
- conditions cinématiques : c'est-à-dire les conditions aux limites sur S_u soit $u_i = \overline{u_i}$

2) champ de contraintes statiquement admissible :

On appelle champ de contraintes statiquement admissible tout champ de contraintes $\sigma_{ij}(M)$ satisfaisant les conditions suivantes :

- continuité.
- conditions statiques ou de type équilibre :
  - équilibre dans le volume V : $\sigma_{ij,i} + f_i = 0$
  - conditions aux limites sur S_σ : $\sigma_{ij} \cdot n_j = \phi_i$

La nécessité de résoudre ces équations aux dérivées partielles peut se faire en utilisant des techniques numériques, basées sur des méthodes d'approximation adéquates pour discrétiser les structures. Ainsi, on donnera un petit rappel sur les méthodes d'approximation.

I.2.2. Rappel sur les méthodes d'approximation :

Un modèle mathématique d'un système physique fait intervenir plusieurs variables ou fonctions dites exactes $U_{ex}(X)$: températures, déplacement, vitesses.....
celles-ci sont représentées par des fonctions approchées $U(X)$ telle que l'erreur $E(X)$ soit assez petite [2] :

$$E(X) = \| U(X) - U_{ex}(X) \| \qquad (10)$$

Pour construire une fonction approchée nous pouvons choisir un ensemble fini de fonctions dépendantes de n paramètres (a_i)

$$U(X, a_1, a_2, a_3, \ldots\ldots\ldots a_n)$$

telle que :

$$U(X) = P_1(X)\ a_1 + P_2(X)\ a_2 + \ldots\ldots\ldots P_n(X)\ a_n \qquad (11)$$

Où : $P_1, P_2 \ldots\ldots P_n$ sont des fonctions connues linéairement indépendantes, tels que des polynômes ou des fonctions trigonométriques. Ces fonctions sont indépendantes des a_1 , a_2, $\ldots a_n$ qui sont les paramètres généraux ou coordonnées généralisées de l'approximation.

Le principe des méthodes d'approximation consiste à remplacer la résolution d'un problème continu à un nombre infini d'inconnues (ou de degré de liberté) par celle d'un problème avec un nombre fini d'inconnues : les paramètres a_i qu'on détermine en définissant la *meilleure* approximation qui minimise un critère (l'énergie potentielle ,l'erreur...) .

Les méthodes d'approximation peuvent être classées dans deux catégories principales :

- méthodes universelles : utilisables dans le cas d'une formulation locale du problème c'est à dire sous forme d'équations aux dérivées partielles avec des conditions aux limites associées. On peut citer : la méthodes de *GALERKIN* ,moindres carrés ...
- Méthodes variationnelles : utilisables dans le cas d'une formulation variationnelle du problème comme la méthode de *RAYLEIGH-RITZ.*

Cette dernière consiste à chercher une meilleure approximation du champ de déplacements réels appartenant au sous espace de dimension finie n des fonctions cinématiquement admissibles engendrées par les fonctions base φ_i (M) :

$$U(M) = \sum_{i=1}^{n} a_i\ \varphi_i(M) \qquad (12)$$

Les fonctions base doivent être cinématiquement admissibles c'est à dire continues et satisfaire les conditions aux limites cinématiques.

Cette meilleure approximation est celle qui minimise l'énergie potentielle totale qui est la condition de stationnarité pour l'état d'équilibre.

La méthode des éléments finis est une extension de la méthode de *RAYLRIEGH-RITZ*. Elle consiste à substituer au champ complet cherché un assemblage de champs partiels limités à des petits domaines (les éléments finis)

I.2.3. Présentation de la méthode des éléments finis :

Nous présentons dans cette section le concept d'élément fini et la méthode d'approximation par éléments finis, qui nous permettront de définir les fonctions approchées et les conditions nécessaires que doivent vérifier ces fonctions afin d'assurer la convergence de l'approximation.

I.2.3.1. Le concept d'élément fini :

La méthode de *RAYLRIEGH-RITZ* est limitée à des géométries relativement simples des structures, car il n'est pas toujours facile de trouver des fonctions base qui conviennent à des géométries complexes.

Le principe de base de la méthode des éléments finis consiste à subdiviser la structure en sous domaines de forme simple appelés *éléments finis*. On va alors définir une approximation de la solution, (déplacements et / ou contraintes) non pas pour l'ensemble de la structure mais pour chacun de ces éléments constitutifs. Le choix des déplacements nodaux comme coordonnées généralisées, permet alors d'exprimer simplement les conditions de continuité de la solution entre éléments adjacents, ainsi que les conditions d'équilibre inter-éléments et finalement de résoudre le problème à l'aide de la méthode des déplacements [1].

Il y a plusieurs formulations d'éléments finis en mécanique des structures. On peut citer les plus utilisées :

- Formulation *déplacement* : dans laquelle on se donne une approximation du champ de déplacements, le critère variationnel étant celui de l'énergie potentielle totale, le choix des coordonnées généralisées sont les déplacements nodaux pour chaque élément [1].
- Formulation *contrainte* ou équilibre : dans laquelle on se donne une approximation du champ de contraintes, le critère variationnel utilisé est celui de l'énergie potentielle totale complémentaire [1].
- Formulation *hybrides* dans laquelle on définit la solution en termes d'une approximation d'une part du champ de contraintes internes en équilibre, d'autre part des déplacements sur la frontière de l'élément, le critère variationnel utilisé est celui de l'énergie potentielle complémentaire totale [1].

Dans notre étude en à utiliser une formulation type déplacements.

I.2.3.2. Approximation par éléments finis :

Choisissons les paramètres a_i dans la relation (11) les valeurs de la fonction exacte U en n points appelés nœuds de coordonnées : X_1 , X_2 ,.....X_n. On impose que la fonction approchée U coïncide avec la fonction exacte U_{ex} en ces nœuds [2].

$$U_{ex}(X_i) = U(X_i) = q_i \qquad i = 1 , n \tag{13}$$

La fonction approchée peut donc s'écrire [2]:

$$U(X) = \sum_{i=1}^{n} N_i(X)\ q_i \qquad (14)$$

les paramètres q_i sont appelés les variables nodales

les fonctions $N_i(x)$ sont les fonctions d'interpolation

la relation (11) définit une approximation non nodale

la relation (14) définit une approximation nodale

L'approximation nodale possède deux propriétés fondamentales qui découle de (14) et (13)

a) d'après la relation (13), les fonctions $N_i(X)$ vérifient :

$$N_j(X_i) = \begin{cases} 0 & si \quad j \neq i \\ 1 & si \quad j = i \end{cases} \qquad (15)$$

b) l'erreur d'approximation définie par (10) s'annule en tous les nœuds X_i.

$$E(X_i) = 0 \qquad (16)$$

La construction de la fonction approchée $U(X)$ est difficile lorsque le nombre n de nœuds et donc de variables nodales q_i devient très important. Ceci a lieu si le domaine V à une forme complexe.

Si la fonction $U(X)$ doit satisfaire des conditions aux limites sur la frontière de V on définit alors la méthode d'approximation nodale par sous domaine [2] qui consiste à :

a) identifier un ensemble de sous domaine V^e du domaine V

b) définir une fonction approchée $u^e(X)$ différente sur chaque sous domaines V^e par la méthode d'approximation nodale. Chaque fonction $u^e(X)$ peut dépendre des variables nodales d'autres sous domaines.

La méthode d'approximation par éléments finis est une méthode particulière d'approximation par sous domaine qui présente les particularités suivantes :

- l'approximation nodale sur chaque domaine V^e ne fait intervenir que les variables nodales attachées à des nœuds situés sur V^e et sur sa frontière.
- les fonctions approchées $u^e(X)$ sur chaque sous domaine V^e sont construites de manière à être continues sur V^e et satisfaire les conditions de continuité entres les différents sous domaines.

Ces sous domaines sont appelés des éléments. Les coordonnées X_i de ces nœuds sont les coordonnes nodales, les valeurs $q_i = u^e(X_i) = U_{ex}(X_i)$ sont les variables nodales (où degré de liberté).

L'approximation par éléments finis présente deux aspects distincts :

- définir analytiquement la géométrie de tous les éléments.
- construire les fonctions d'interpolation $N_i(X)$ correspondantes à chaque élément

I.2.3.3. Propriétés des fonctions approchées u(X) :

Par soucis d'alléger les notations nous remplaçons la notation des fonctions approchées $u^e(X)$ par $u(X)$.

Ces fonctions qu'on peut construire par la formule (14) peuvent s'écrire sous forme matricielle :

$$\{u\} = [N] \cdot \{q\} \qquad (17)$$

$[N]$ matrice des fonctions d'interpolation

$\{q\}$ vecteur des déplacements nodaux

$\{u\}$ vecteur des déplacements à l'intérieur des éléments.

Ces fonctions d'interpolation doivent être choisies d'une façon à assurer les critères de convergence, qui garantissent que la solution approchée tend vers la solution exacte lorsque que la taille de l'élément tend vers zéro.

Ces critères sont :

a) les fonctions $u(X)$ doivent être cinématiquement admissibles.

b) continuité sur l'élément : pour l'obtention d'une fonction approchée $u(X)$ continue sur l'élément ainsi que ces dérivées jusqu'à l'ordre s, il faut utiliser des fonctions $N_i(X)$ continues et à dérivées continues jusqu'à l'ordre s.

c) continuité entre les éléments : si nous désirons que $u(X)$ et ses dérivées jusqu'à l'ordre s soient continues sur la frontière commune à deux éléments, il faut que $u(X)$ et ses dérivées jusqu'à l'ordre s dépendent de manière unique des seules variables nodales associées aux nœuds de cette frontière. Considérons d'abord la continuité de $u(x)$ sur une frontière (continuité de classe C^0).

$$u(X) = \sum_{i=1}^{n} N_i(X) \cdot q_i$$

Les produits $N_i(X) \cdot q_i$ doivent être nuls si q_i n'est pas une variable nodale associée à un nœud de cette frontière, d'où $N_i(X) = 0$ lorsque X est situé sur une frontière et q_i n'est pas une variable nodale de cette frontière.

La condition pour que $\frac{\partial u(X)}{\partial X} = \sum_{i=1}^{n} \frac{\partial N_i(X)}{\partial X} q_i$ est que $\frac{\partial N_i(X)}{\partial X} = 0$ lorsque X est situé sur une frontière et q_i n'est pas une variable nodale de cette frontière.

La notion de continuité sur les frontières entre les éléments est une notion clé dans la méthode des éléments finis.

Nous présentons ci dessous le étapes fondamentales de calcul des structures afin d'arriver à une formulation élémentaire au niveau de l'élément fini, (matrice de rigidité, vecteur force élémentaire) puis à une formulation globale au niveau de la structure complète, (matrice de rigidité globale, vecteur force globale)afin d'aboutir à la résolution numérique du système d'équations algébriques.

I.3. Etapes de calcul des structures par la méthode des éléments finis :

Le calcul des structures par la M.E.F s'effectue par les étapes suivantes :

a) modélisation et discrétisation de la structure.

b) Calcul des propriétés des éléments : calcul de la matrice de rigidité élémentaire, des vecteurs de charges extérieurs des différents éléments finis.

c) assemblage des matrices de rigidité élémentaires et des vecteurs force élémentaire.

d) résolution du système algébrique obtenu après discrétisation de la structure, et calcul des contraintes.

e) interprétation des résultats.

Dans ce qui suit nous allons détailler ces différentes étapes et nous résumons ces étapes dans la figure I.3

I.3.1. Modélisation et discrétisation de la structure :

Cette étape est extrêmement importante car elle consiste à choisir le modèle mathématique de calcul, et la subdivision de la structure en éléments finis, de manière à approximer le mieux possible sa géométrie.

I.3.1.1. Notion des structures discrètes :

C'est une structure composée d'éléments discrets (poutres, barres) de forme géométrique simple interconnectés en des points de la structure qu'on appellera 'nœuds ' réticulés. On

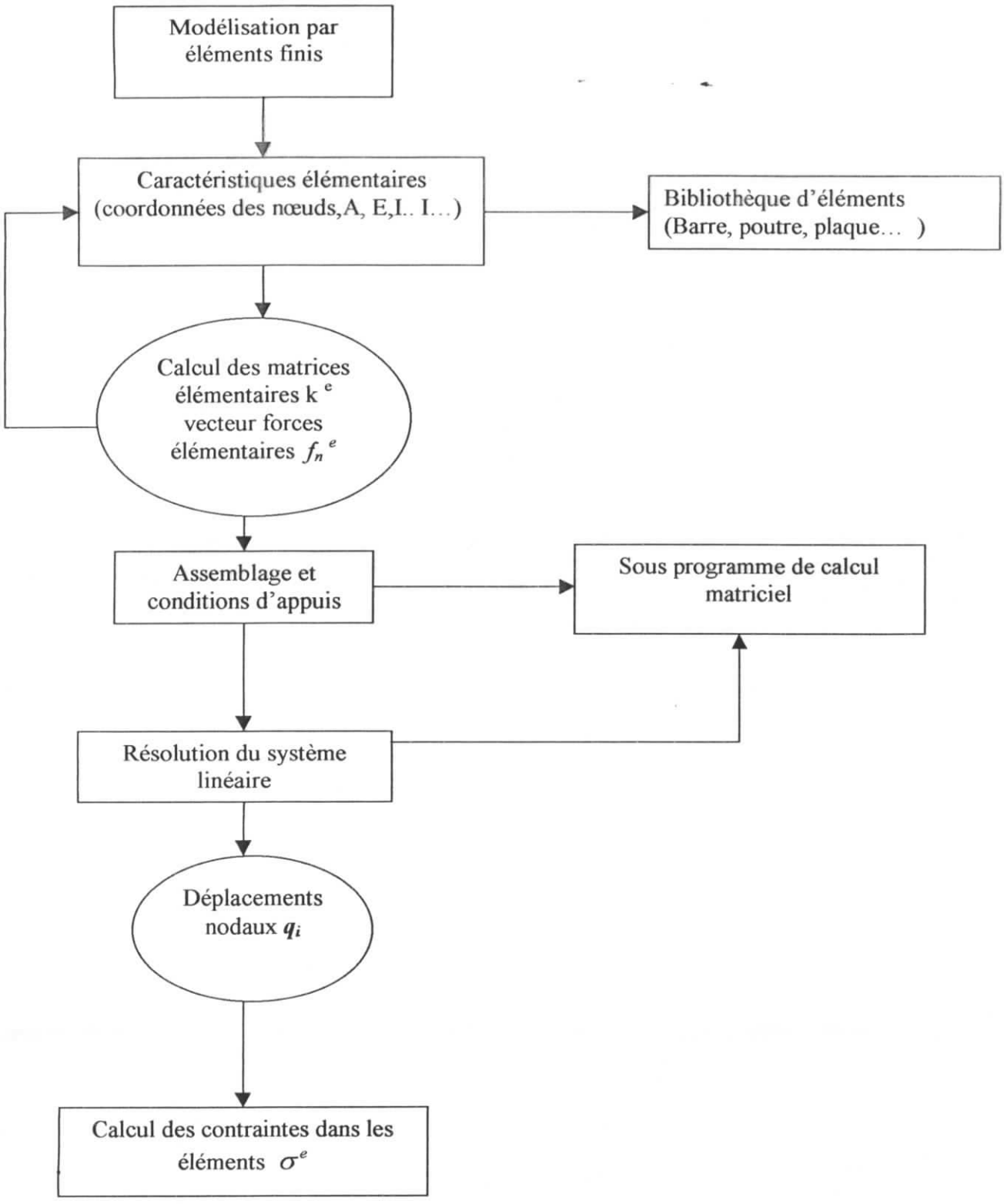


Figure I.2 :Organigramme simplifié de traitement pour l'analyse statique linéaire

caractérise le comportement de cette structure par un nombre fini de paramètres inconnus, qui sont les déplacements aux nœuds.

Cette notion de structure discrète s'applique de façon immédiate aux cas des structures qui nous intéressent, à savoir :

- Les treillis constitués de barres et de nœuds non rigides, où l'élément *barre* ne travaille qu en compression et traction et où des rotations d'ensemble sont permises dans le plan de l'élément.

La modélisation des treillis impose que les charges extérieures appliquées soient concentrées aux nœuds.

Nous allons dans ce que suit introduire la formulation énergétique d'un élément fini afin de formuler l'expression de l'énergie de déformation et l'énergie potentielle totale en fonction des déplacements.

I.3.1.2. Formulation énergétique d'un élément fini :

D'après le théorème des travaux virtuels [1] :

$$\delta \Psi = \delta W \tag{18}$$

soit :

$$\int_V \{\sigma\}^T \{\delta\varepsilon\}\, dV = \int_V \{f\}^T \{\delta u\}\, dV + \int_{S\sigma} \{\phi\}^T \{\delta u\}\, dS \tag{19}$$

tels que :

$\{\delta u\}$: vecteur des déplacements virtuels

$\{u\}$: vecteurs des déplacements

$\{\sigma\}^T$: transposé du vecteur des contraintes

$\{f\}^T$: transposé du vecteur des forces de volume

$\{\phi\}^T$: transposé du vecteur des forces de surface

$\{\delta\varepsilon\}$: vecteur des déformations virtuelles.

Le travail des forces appliquées [1] est :

$$W = -\int_V \{u\}^T \{f\} dV - \int_S \{u\}^T \{\phi\} dS \tag{20}$$

W : travail des forces appliquées

$\{u\}^T$:transposé du vecteur des déplacements

$\{\phi\}$: vecteur des forces de surface

L'énergie de déformation s'écrit en fonction du potentiel de déformation [1]:

$$\Psi = \int_V \Phi(\varepsilon)\, dV = \int_V \frac{1}{2}\{\varepsilon\}^T [E]\{\varepsilon\}\, dV \qquad (21)$$

tels que :

$\Phi(\varepsilon)$ est le potentiel de déformation

Ψ : énergie de déformation

$[E]$: matrice constitutive de l'élément

$\{\varepsilon\}$vecteur des déformations.

Pour un état d'équilibre stable, l'énergie potentielle totale est dite *stationnaire* c'est à dire que l'énergie potentielle atteint un minimum absolu [1].

$$\delta\Pi = \delta(\Psi - W) = 0 \qquad (22)$$

l'énergie potentielle totale s'écrit simplement :

$$\Pi = \Psi - W \qquad (23)$$

soit sous la forme intégrale suivante :

$$\Pi = \int_V \Phi(\varepsilon)\, dV - \int_V \{f\}^T\{u\}\, dV - \int_{S\sigma} \{\phi\}^T\{u\}\, dS \qquad (24)$$

Π : énergie potentielle totale.

Selon ces les relations (21) et (20) on peut écrire d'une manière plus simple :

$$\Pi = \int_V \frac{1}{2}\{\varepsilon\}^T[E]\{\varepsilon\}\, dV - \int_V \{u\}^T\{f\}dV - \int_{S\sigma} \{u\}^T\{\phi\}\, dS \qquad (25)$$

Si le champ de déplacements est approché par :

$$\{u\}=[N]\{q\}$$

$\{q\}$: vecteur des déplacements nodaux

$[N]$ matrice des fonctions d'interpolation

et d'après la relation (2) on obtient :

$$\{\varepsilon\}=[B]\{q\} \tag{26}$$

telle que :

$$[B]=[N]\left[\frac{\partial}{\partial x}\right] \tag{27}$$

On obtient finalement la forme discrétisé de l'énergie de déformation et de l'énergie potentielle totale [3]:

$$\Psi = \sum_{i=1}^{n}\Psi_i^e = \frac{1}{2}\sum_{i=1}^{n} \{q\}_i^T [k]_i \{q\}_i \tag{28}$$

$$\Pi = \sum_{i=1}^{n} \Pi_i^e = \frac{1}{2}\sum_{i=1}^{n} \{q\}_i^T [k] \{q\}_i - \sum_{i=1}^{n} \{q\}_i^T [r_e]_i \tag{29}$$

l'indice de sommation indique que nous incluons la contribution de tous les éléments de la structure

avec :

$$[k] = \iiint_{V^e}[B]^T[E][B]\, dV \tag{30}$$

$[k]$ est la matrice de rigidité élémentaire.

$$[r_e] = \int_{V^e} [N]^T\{F\}\, dV + \int_{S^e} [N]^T\{\phi\}\, dS \tag{31}$$

$[r_e]$ est le vecteur du chargement élémentaire. On note que V^e et S^e sont respectivement le volume et la surface de l'élément.

I.3.2. Calcul des propriétés des éléments :

La section suivante est consacrée à la présentation et la formulation de la matrice de rigidité d'un élément fini type barre (ou de treillis) qui permet d'analyser des structures tridimensionnelles constituées de poutres articulées aux deux extrémités et qui ne transmettent que des efforts de traction ou de compression aux nœuds d'assemblage. Il s'agit d'utiliser l'élément à deux nœuds qui est défini d'abord dans un repère local puis défini dans un repère tridimensionnel.

I.3.2.1. Formulation de la matrice de rigidité d'une barre dans le repère local :

Soit la barre représentée sur la figure I.4, de longueur L parallèle à l'axe Ox, de caractéristiques constantes, module d'élasticité longitudinal E, de section A. Cet élément est composé de deux nœuds avec un seul degré de liberté par nœud : le déplacement longitudinal

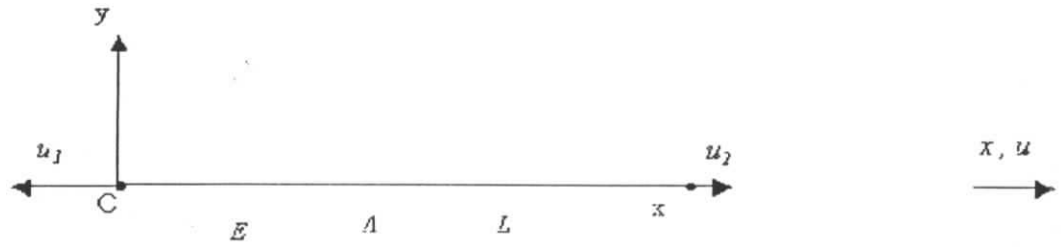


Figure I.4 : Elément type barre à deux nœuds dans le repére local

D'après la théorie de l'élasticité [3] le champ de déplacements à l'intérieur de l'élément est linéaire. Donc on peut écrire

$u = \alpha_1 + \alpha_2 x$

nous vérifions que ce champ des déplacements satisfait les conditions énumérées dans le paragraphe I.2.3.3.

L'approximation du champ de déplacements[3,5] sera alors :

$$u = \begin{bmatrix} \frac{L-x}{x} & \frac{x}{L} \end{bmatrix} \begin{Bmatrix} q_1 \\ q_2 \end{Bmatrix}$$

où :

$$[N] = \begin{bmatrix} \frac{L-x}{L} & \frac{x}{L} \end{bmatrix} \qquad (32)$$

Et d'après la relation (20) :

$$[k] = \int_{V^e} [B]^T [E] [B] \, dV$$

La relation (27) nous permet d'avoir la matrice $[B]$

$$[B] = \begin{bmatrix} -\frac{1}{L} & \frac{1}{L} \end{bmatrix}$$

donc :

$$[k] = EA \int_0^L \begin{bmatrix} -\frac{1}{L} \\ \\ \frac{1}{L} \end{bmatrix} \begin{bmatrix} -\frac{1}{L} & \frac{1}{L} \end{bmatrix} dx = \frac{EA}{L^2} \int_0^L \begin{bmatrix} 1 & -1 \\ \\ -1 & 1 \end{bmatrix} dx$$

On obtient finalement la matrice de rigidité de dimension 2×2 pour un élément barre

$$[k] = \begin{bmatrix} \frac{EA}{L} & -\frac{EA}{L} \\ -\frac{EA}{L} & \frac{EA}{L} \end{bmatrix} \qquad (33)$$

I.3.2.2. Formulation du vecteur force élémentaire dans un repère locale :

Dans le cas des structures discrètes de type treillis ou les forces extérieures sont appliquées qu'aux nœuds (pas de force de volume). Le travail des forces extérieures donnée par la relation (20) devient après discrétisation de la structure :

$$W = \sum_{i=1}^{n} W_i^e = \sum_{i=1}^{n} \{q\}_i^T [r_e]_i \qquad (34)$$

Avec :

$$[r_e] = \int_{S^e} [N]^T \{\phi\} dS \qquad (35)$$

Le travail des forces appliquées de chaque élément est :

$$W^e = \{q\}^T [r_e] \qquad (36)$$

$\{q\}$ déplacements nodaux de l élément .

Les forces extérieures sont appliquées aux nœuds de l'éléments et d'après la relation (37) on obtient finalement [5] :

$$[r_e] = \int_{S^e} [N]^T \{\phi\} dS = \{f_n\} \qquad (37)$$

$$\{f_n\} = \begin{Bmatrix} f_{n1} \\ f_{n2} \end{Bmatrix} \qquad (38)$$

$\{f_n\}$ forces concentrées aux nœuds de l'élément barre a deux nœuds

I.3.2.3. Formulation de la matrice de rigidité et du vecteur force élémentaire dans le repère global :

Une structure spatiale, de type treillis, est constituée d'un assemblage de barres dans l'espace, les variables nodales sont les trois composantes du vecteur déplacement exprimées dans un repère global commun (X Y Z). Dans la section précédente la matrice de rigidité et

le vecteur des forces sont définis dans le repère local ($x\ y\ z$) l'opération de l'assemblage nécessite la transformation des caractéristiques élémentaires dans les repères locaux dans un repère global.(figure I.5)

Dans le repère local. : $[q]_{loc} = \langle u_1\ v_1\ w_1\ ,\ u_2\ v_2\ w_2 \rangle$

Dans le repère global : $[q]_{glo} = \langle U_1\ V_1\ W_1\ ,\ U_2\ V_2\ W_2 \rangle$

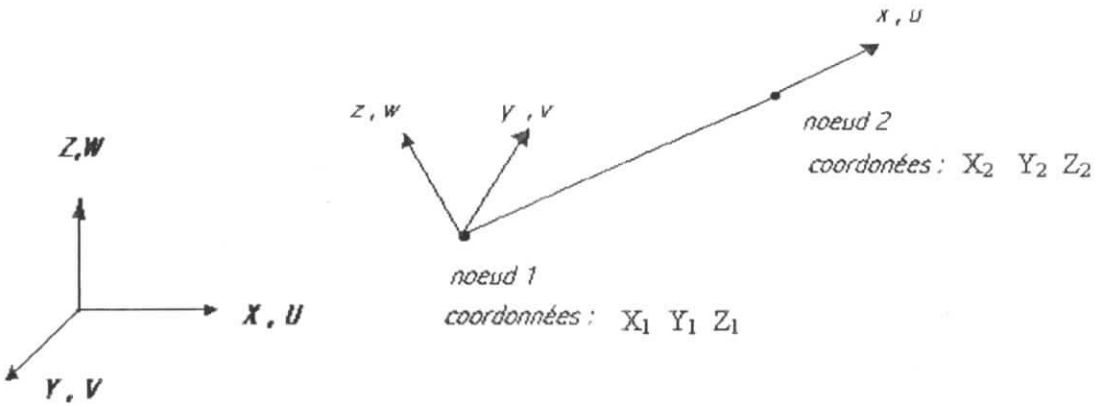


figure I.5 :repère globale et repère locale

Les variables locales s'expriment en fonction des variables globales et d'une matrice de rotation.

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = [M] \begin{bmatrix} U \\ V \\ W \end{bmatrix} \tag{39}$$

Les composantes de la matrice rotation qui sont les cosinus directeurs des axes locaux [6] s'expriment ainsi :

$$[M] = \begin{bmatrix} cos(x,X) & cos(x,Y) & cos(x,Z) \\ cos(y,X) & cos(y,Y) & cos(y,Z) \\ cos(z,X) & cos(z,Y) & cos(z,Z) \end{bmatrix} \tag{40}$$

$$[M] = [M]^{-1}$$

Les cosinus directeurs de l'axe x et la longueur L sont :

$$M_{11}=\frac{X_2-X_1}{L}$$

$$M_{12}=\frac{Y_2-Y_1}{L}$$

$$M_{13}=\frac{Z_2-Z_1}{L}$$

La matrice de rigidité élémentaire dans le repère local est :

$$[k] = \frac{EA}{L}\begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 \\ & & 0 & 0 & 0 & 0 \\ & & & 1 & 0 & 0 \\ & & & & 0 & 0 \\ sym & & & & & 0 \end{bmatrix} \quad (41)$$

Le vecteur force élémentaire dans le repère local est :

$$\langle f_n\rangle=\langle f_1\ 0\ 0\ f_2\ 0\ 0\ \rangle \quad (42)$$

La matrice de rigidité élémentaire dans le repère global est :

$$[k]_{glo}=[T]^T[k]_{loc}[T] \quad (43)$$

Le vecteur force élémentaire dans le repère global est :

$$\langle f_n\rangle_{glo}=[T]^T\langle f_n\rangle_{loc} \quad (44)$$

telle que $[T]$ est une matrice de 6×6 et qui a la forme suivante :

$$[T]=\begin{bmatrix} [M] & -[M] \\ -[M] & [M] \end{bmatrix} \quad (45)$$

I.3.3. Assemblage des matrices de rigidité élémentaires :

Cette étape consiste à chercher pour la structure complète l'expression matricielle des énergies de déformation et du travail des forces appliquées en fonctions des déplacements inconnues *en tous nœuds de la structure.* Ceci nécessite l 'assemblage des caractéristiques élémentaires (matrice de rigidité, vecteur forces) pour *tous les éléments.*

L'énergie potentielle totale et l'énergie de déformation de la structure est donné par la formule (28) et(29)

$$\Pi=\sum_{i=1}^{n}\Pi^{e}=\frac{1}{2}\sum_{i=1}^{n}\{q\}_i^{T}[k]\{q\}_i-\sum_{i=1}^{n}\{q\}_i^{T}[f_e]_i$$

$$\Psi=\sum_{i=1}^{n}\Psi_i^{e}=\frac{1}{2}\sum_{i=1}^{n}\{q\}_i^{T}[k]_i\{q\}_i$$

qu'on peut écrire [3] :

$$\Pi=\frac{1}{2}\{q\}^{T}[K]\{q\}-\{q\}^{T}\{F\} \quad (46)$$

$$\Psi=\frac{1}{2}\{q\}^{T}[K]\{q\} \quad (47)$$

$[K]$ est la matrice de rigidité globale

$\{F\}$ est le vecteur force global

$\{q\}$ déplacements nodaux de la structure

$$[K]=\sum_{i=1}^{n}[k]_i \quad (48)$$

$$\{F\}=\sum_{i=1}^{n}\{f_n\}_i \quad (49)$$

I.3.4. Résolution et calcul des contraintes :

I.3.4.1 Résolution :

Après discrétisation de la structure on a pu formuler l'énergie potentielle et l'énergie de déformation sous une forme de sommation. Ainsi on utilise le théorème de CASTIGLIANO et résoudre le système d'équations algébrique obtenu [1].

D'après le principe des travaux virtuels, on à l'équilibre

$$\delta\Psi=\delta W$$

Dans le cas d'un système de forces ponctuelles, le travail des forces appliquées sur la structure s'écrit simplement :

$$W=\{q\}^{T}\{F\} \quad (50)$$

D'où $\delta W=F_i\,\delta q_i$

On à donc : $F_i=\frac{\delta\Psi}{\delta q_i}$

Et si on exprime l'énergie de déformation en fonction seulement des déplacements aux nœuds

$$F_i=\frac{\partial \Psi(q_i)}{\partial q_i} \tag{51}$$

On peut répéter le raisonnement pour tous les degrés de liberté on obtient donc n équations qui constituent les équations d'équilibre des nœuds soit :

$$F=\nabla_u \quad \Psi \tag{52}$$

($\nabla_u \Psi$ est le gradient de l'énergie de déformation par rapport aux déplacements u)

On à d'autre part : $\Psi=\frac{1}{2}\{q\}^T[K]\{q\}$

Et après application du théorème de *CASTIGLIANO* :

$$\nabla_q \Psi=[K]\{q\}=\{F\} \tag{53}$$

Donc le système algébrique obtenu est :

$$[K]\{q\}=\{F\} \tag{54}$$

Pour la résolution de ce système algébrique par les méthodes numériques adéquates(méthode d'élimination de *GAUSS*, méthode de décomposition...) il faut éliminer les déplacements imposés. (ou les degrés de liberté imposés)Ainsi nous obtiendrons une matrice $[K]$ non singulière qui nous permet de résoudre le système algébrique et d'obtenir les déplacements des nœuds (ou les degrés de libertés)recherchés .

I.3.4.2 calcul des contraintes dans les barres :

Après avoir obtenu les déplacements des nœuds de toute la structure, nous pouvons extraire les déplacements des nœuds de chaque élément et de les exprimer dans le repère local. On utilisant la relation (39) ,afin de calculer les déformations de la barre d'apres la relation (26) :

$$\{\varepsilon\}=[B]\{q\}$$

Ainsi le calcul des contraintes dans les barres peut s'effectué on utilisant la relation de *HOOK* pour un matériau à comportement élastique et linéaire :

$$\sigma = E.\varepsilon$$

INTRODUCTION :

La programmation mathématique se propose d'étudier les problèmes d'optimisation afin de concevoir et de mettre en œuvre des algorithmes de résolution de tels problèmes.

Cette discipline mathématique indépendante est nouvelle car elle remonte historiquement à la découverte de la méthode du *simplexe* en 1947 appliquée à l'économie et n'a cessée depuis d'évoluer pour traiter (comme toute autre théorie globale) des problèmes aussi variés que complexes.

C'est ainsi que les problèmes d'optimisation linéaire ont été abordés en premier pour aboutir, vers 1960 et durant toute cette décennie, à la résolution des problèmes non linéaires.

Les algorithmes élaborés durant cette période faisaient appel aux dérivées des fonctions, c'est alors ensuite, vers 1970, que des méthodes d'optimisation non différentiable ont vu le jour. La discipline s'est aussi enrichie des autres branches des mathématiques et de la recherche opérationnelle mais son évolution rapide n'a été possible que grâce à l'outil informatique. En effet, l'ordinateur permet de traiter des problèmes faisants intervenir des milliers de variables.

Il nous a donc paru intéressant d'aborder dans ce chapitre des méthodes variées, pas toujours générales, mais qui nous renseignent sur l'évolution de la programmation mathématique. C'est ainsi que nous commencerons par donner une méthode d'optimisation unidimensionnelle (intéressante pour bien cerner le problème de l'optimisation). Nous introduirons ensuite l'optimisation non linéaire sans contraintes, pour nous attarder enfin sur l'optimisation non linéaire avec contrainte (qui est le type d'optimisation adopté dans notre présent travail), nous donnerons les conditions nécessaires et suffisantes d'optimalité ainsi que les principales méthodes de résolution (méthodes primales et duales).

Ce développement nous aidera à mieux aborder la méthode SQP, que nous donnerons par la suite, avec laquelle nous avons traité notre problème d'optimisation des treillis spatiales.

II.1. Définition de la programmation mathématique [7] :

D'une façon très générale la programmation mathématique est un problème d'*optimisation* sous contraintes dans R^n de la forme :

$$(P) \quad \begin{cases} \text{Minimiser } f(x) \\ \text{sous les contraintes :} \\ g_i(x) \geq 0 \quad i=1,\ldots\ldots\ldots,m \\ x \in S \subset R^n \end{cases}$$

Ici le vecteur $x \in R^n$ a pour composantes $x_1, x_2, \ldots\ldots, x_n$ qui sont les inconnues du problème. La fonction f est appelée la *fonction objectif* et l'ensemble des conditions $g_i(x) \geq 0$ (i=1,....,m) et $x \in S$ sont les *contraintes* du problème.

Dans ce qui suit nous n'envisagerons que le cas de la minimisation de la fonction objectif. Ceci n'est pas restrictif car le problème de la recherche d'un maximum d'une fonction se ramène à la recherche du minimum de $h = -f$.

On appelle *solution* du problème (P) tout vecteur x vérifiant les contraintes, c'est à dire tel que :

$$g_i(x) \geq 0 \quad i=1,\ldots\ldots\ldots,m \quad \text{et} \quad x \in S \subset R^n$$

On appelle *solution optimale* (optimum global) du problème (P) une solution qui minimise f parmi l'ensemble de toutes les solutions.

II.2. Mise en forme des problèmes d'optimisation :

Nous distinguons entre les problèmes d'optimisation différents types selon la nature du problème réel (ou physique). Il est donc nécessaire qu'un problème soit tout d'abord identifié puis classé selon sa nature pour pouvoir ensuite choisir la méthode de résolution la plus adéquate parmi toutes celles qui existent. La mise en forme d'un problème d'optimisation doit procéder selon les étapes suivantes :

a) Identifier et déterminer de manière unique la *fonction objectif* et les *variables* du problème, le choix peut ; dans certains cas ; être grand mais doit amener à une stratégie de résolution claire. En effet les *variables* doivent être bien connues et mesurées et doivent mener à des *contraintes* (si elles existent) aisément formulées et/ou calculées (ceci est particulièrement important lors d'un calcul numérique assisté par ordinateur)

b) Procéder à la classification du problème parmi toutes les familles connues. Cette identification dépend du type de problème à traiter, nous donnons ci-dessus les principales classes et critères :

- Problème unidimensionnel ou multidimensionnel
- Problème avec ou sans contraintes.

- Problème linéaire ou non linéaire. Ici la non linéarité peut porter sur la *fonction objectif* ou sur les *contraintes* (si elles existent) . Ce critère, une fois les fonctions déterminées, est fixé par la nature du problème à traiter et il n'est pas possible de le contourner.

c) Choisir entre les différentes méthodes d'optimisation car pour un même problème plusieurs algorithmes peuvent s'appliquer. Le choix doit être dicté par l'efficacité, la vitesse de convergence et la mise en œuvre - plus au moins délicate – de l'algorithme. Aussi, le choix doit, quand cela est possible, prendre en ligne de compte les conséquences pratiques tels que le type et le nombre de fonctions à calculer (gradients, jacobiens...), le temps de calcul pour les structures importantes et la précision désirée pour l'évaluation de la solution.

II.3. Présentation des méthodes d'optimisation :

II.3.1. Optimisation unidimensionnelle :

Les méthodes d'optimisation unidimensionnelle où la fonction objectif est à une seule variable se formulent de la manière suivante [7]:

Déterminer $\alpha \geq 0$ minimisant :

$$g(\alpha)=f(x^0+\alpha.d)$$

ou $x^0 = (x_1^0, x_2^0, \ldots\ldots, x_n^0)^T$ est le dernier point obtenu

et $d = (d_1, d_2, \ldots\ldots\ldots, d_n)^T$ une direction de déplacement.

Ces méthodes peuvent, selon le choix, utiliser ou non les dérivées des fonctions *objectif* et *contrainte* (si cette dernière existe).

Parmi les méthodes utilisant les dérivées nous pouvons citer :

- la méthode de *Newton-Raphson*, la méthode de la sécante, la méthode de la dichotomie avec dérivées.

Et parmi celles qui n'utilisent pas les dérivées :

- Méthode de l'interpolation quadratique, méthode des fonctions unimodales, la méthode de la dichotomie sans dérivées, la méthode du nombre d'or.

Toutes ces méthodes algorithmiques de résolution adoptent la même stratégie. Posons nous, par exemple, à l'itération k ou le point courant est x^k le point à l'itération $k+1$ est obtenu par :

$$x^{k+1} = x^k + \alpha.d$$

où d est la direction de déplacement

et α le pas du déplacement. Il est, selon la méthode de résolution, obtenu en fonction des dérivées de la fonction g , à minimiser, ou non.

L'optimisation unidimensionnelle peut être, comme toutes les autres familles, conjuguée à des contraintes qui délimitent l'ensemble des solutions. Détaillons ce point important qui nous servira par la suite à mieux cerner les problèmes multidimensionnels avec contraintes.

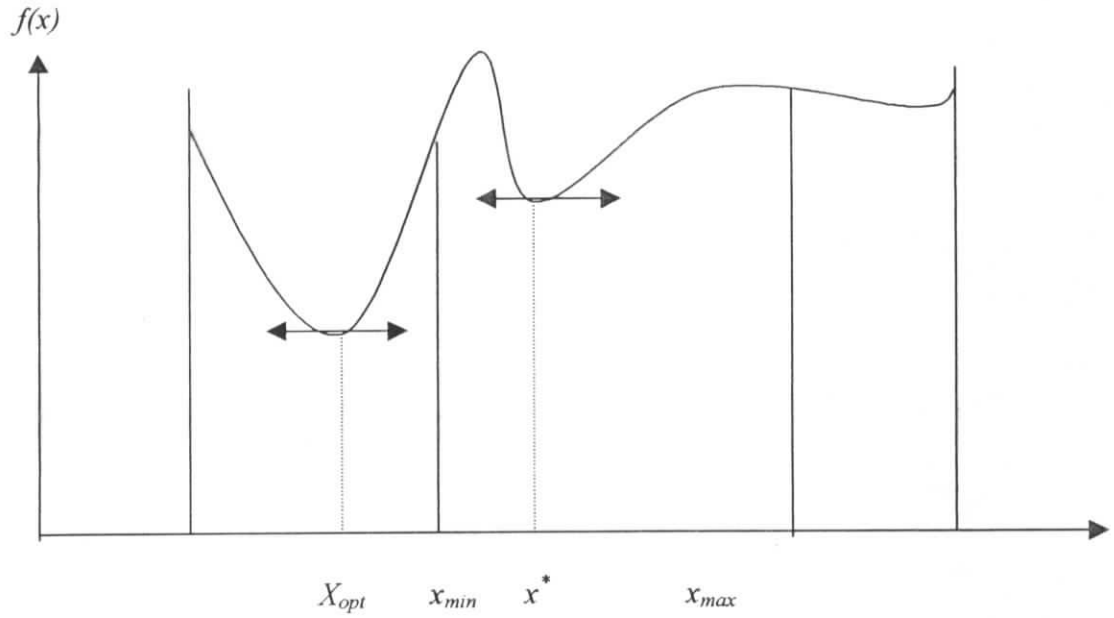


Figure II.1 ***Optimum local et optimum global***

La figure(1) ci-dessus montre, pour une fonction à une seule variable $f(x)$ possédant un optimum global x_{opt} , que les contraintes exprimées de la manière suivante :

$$x_{min} \leq x \leq x_{max}$$

délimitent le champs de recherche du minimum, l'ensemble :

$$S=\{x / x \in R \; ; \; x_{min} \leq x \leq x_{max} \}$$

sera l'ensemble des solutions comme défini plus haut. Il apparaît clairement que la solution optimale trouvée sous ces conditions ne coïncidera que dans un cas particulier avec l'optimum global (sous la condition que x_{opt} appartienne à S) et le minimum ainsi déterminé sera local sur S. Il est noté sur la figure(1) x^*.

II.3.2. Optimisation non linéaire sans contraintes :

Il existe un grand nombre de méthodes adaptées à la résolution de ce type de problème elles font, selon le cas, appel ou non au calcul des dérivées, parmi elles nous pouvons citer :

a) Les méthodes numériques pour les fonctions différentiables,
méthode du gradient, méthode de la plus forte pente (accélérée ou non), méthode du second ordre, méthodes de directions conjuguées, méthode de Newton...,etc.
b) Les méthodes d'optimisation des fonctions convexes non partout différentiables,
c) Les méthodes d'optimisation sans dérivées,
Parmi elles nous citerons les méthodes de relaxation cycliques et l'algorithme de Powell.

Cette classe de problème est traité à l'aide de méthodes spécialement développées pour tenir compte de la spécificité des problème non linéaires. En effet les conditions nécessaires et suffisantes d'optimalité sont ici différentes de celles que nous allons voir en détail dans le prochain paragraphe. Toutes fois, il est intéressant de donner un aperçu des conditions d'optimalité afin de mettre en avant la différence entre les problèmes avec et sans contraintes.

En supposant une fonction $f(x)$, $x \in \mathrm{R}^n$, à dérivées premières $\partial f / \partial x_i$ et seconde $\partial^2 f / \partial x_i \partial x_j$ continues sur $\mathbf{R}^n$, alors une condition *nécessaire* pour que x^* soit un minimum (local ou global) de f est [7]:

a) $\nabla f(x^*) = \vec{0}$ (condition de stationnarité)
b) le hessien $\nabla^2 f(x^*)$ est une matrice semi-définie positive,
c'est à dire que : $\forall y \in R^n$, $y^t . \nabla^2 f(x^*) . y \geq 0$

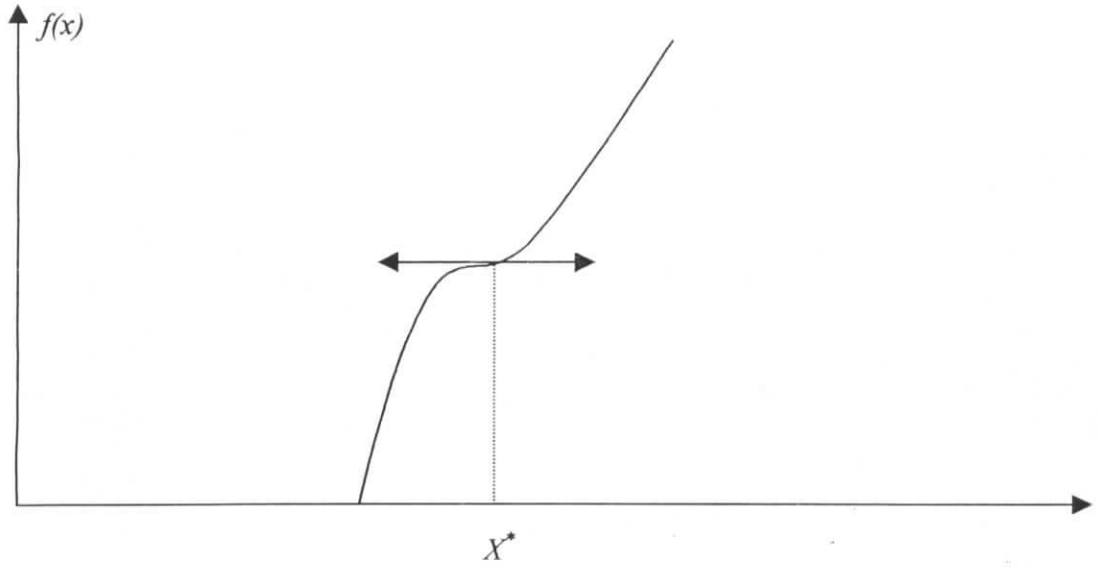


Figure II.2 ***Point stationnaire, non optimal***

On voit dans la figure ci-dessus que la stationnarité seule n'implique pas l'optimalité, d'ou la condition (b).
De même et sous les mêmes conditions sur f et ses dérivées énoncées plus haut, une condition *nécessaire* et *suffisante* pour que x^* soit un minimum local est :

a) $\nabla f(x^*) = 0$ (stationnarité)
b) le hessien $\nabla^2 f(x^*)$ est une matrice définie positive,
c'est à dire que : $\forall y \in R^n$, $y^t . \nabla^2 f(x^*) . y > 0$

La condition (b) revient à dire que la fonction f est *convexe* au voisinage de x^*.

II.3.3. Optimisation non linéaire avec contraintes :

On s'intéresse ici au problème suivant :

$$(P) \quad \begin{cases} \text{Minimiser } f(x) \\ \text{sous les contraintes :} \\ g_i(x) \geq 0 \quad i = 1, \ldots\ldots\ldots, m \\ x \in S \subset R^n \end{cases}$$

Les fonctions f et g_i ($i = 1, \ldots\ldots, m$) sont supposées continues et différentiables, on notera X l'ensemble des solutions de (P) c'est à dire :

$$X = \{ x \in S / g_i(x) \geq 0 \ , \forall i = 1, \ldots\ldots\ldots, m \}$$

Introduisons avant d'énoncer les conditions nécessaires d'optimalité, les notions de directions admissibles et de qualification des contraintes.

Nous supposons que X n'est pas vide. Dire que $x^0 \in X$ est un optimum local de (P) implique que $f(x)$ ne peut décroitre lorsque x décrit un arc de courbe Γ partant de x^0 et contenu dans l'ensemble X des solutions. Un tel arc de courbe Γ sera dit *admissible* en x^0 et sera défini par une fonction φ ($R^+ \longrightarrow R^n$) continument différentiable du paramètre $\theta \geq 0$:

$$\varphi(\theta) = [\varphi_1(\theta), \varphi_2(\theta), \ldots\ldots, \varphi_n(\theta)]$$

et tel que:

a) $\varphi(0) = x^0$
b) pour tout $\theta > 0$ suffisament petit, $\varphi(\theta) \in X$.

On appellera alors *direction admissible* en x^0 tout vecteur

$$y = \frac{d\varphi}{d\theta}(0) = \left[\frac{d\varphi_1}{d\theta}(0), \ldots\ldots\ldots, \frac{d\varphi_n}{d\theta}(0) \right]$$

tangent à un arc de courbe $\varphi(\theta)$ admissible en x^0 .
Une particularité de y est:

$$\nabla g_i^T(x^0).y \leq 0 \quad (\forall i = 1,2,\ldots.. m)$$

On dit que le domaine X défini par les contraintes $g_i(x) \leq 0$ satisfait en $x^0 \in X$ l'hypothése de *qualification des contraintes* (Kuhn et Tucker, Abadie) si et seulement si:

a) toutes les fonctions g_i sont linéaires.
b) Toutes les fonctions g_i sont convexes et X a un interieur non vide.
c) Les gradients $\nabla g_i(x^0) (i \in I^0)$ des contraintes saturées en x^0 sont linéairement indépendants.

Remarquons enfin que les contraintes saturées en x^0 vérifient $\nabla g_i = 0$ $(i \in I^0)$, I^0 étant l'ensemble des indices des contraintes saturées.

II.3.3.1. Les conditions nécessaires d'optimalité de Kuhn et Tucker:

Le théorème suivant est fondamental et donne, sous l'hypothése de qualification des contraintes, une condition nécessaire d'optimalité locale pour un problème d'optimisation avec contraintes du type (P).

Théorème 1 [7]:

On suppose que les fonctions f et g_i $(i=1,2,\ldots\ldots,n)$ sont continument différentiables, et que l'hypothése de qualification des contraintes est vérifiée en $x^0 \in X$, avec:

$$X=\{x \in \mathrm{R}^n / \; g_i(x) \leq 0 \; , \forall i=1,2,\ldots\ldots,m\}$$

Alors une condition nécessaire pour que x^0 soit un optimum local de (P) est qu'il existe des nombres $\lambda_i \geq 0$ $(i=1,2,\ldots\ldots\ldots,n)$ appelés multiplicateurs de *Kuhn* et *Tucker*, tels que:

$$\begin{cases} \nabla f(x^0) + \sum_i \lambda_i \nabla g_i(x^0) = 0 \\ \lambda_i g_i(x^0) = 0 \end{cases}$$

II.3.3.2. Les conditions suffisantes d'optimalité ; Points cols et fonctions de Lagrange:

Nous allons étudier maintenant des conditions *suffisantes* d'optimalité pour des problèmes du type (P) c'est à dire:

$$(P)\quad \begin{cases} \text{Minimiser } f(x) \\ \text{sous les contraintes :} \\ g_i(x) \geq 0 \quad i = 1, \ldots\ldots\ldots, m \\ x \in S \subset R^n \end{cases}$$

Ici S désigne un ensemble de R^n qui pourra par exemple être un ensemble de coordonnées entières (cas de la programmation entière).

Associons à chaque contrainte $i \in I$ un nombre réel $\lambda_i \geq 0$ appelé multiplicateur de *Lagrange*. La *fonction de Lagrange* associé au problème (P), est par définition la fonction :

$$L(x,\lambda) = f(x) + \sum_i \lambda_i g_i(x)$$

Définition [7] :

Soit $\bar{x} \in S$ et $\bar{\lambda} \geq 0$.

On dit que $(\bar{x},\bar{\lambda})$ est un point-col de $L(x,\lambda)$ si :

a) $L(\bar{x},\bar{\lambda}) \leq L(x,\bar{\lambda}) \qquad \forall x \in S$

b) $L(\bar{x},\lambda) \leq L(\bar{x},\bar{\lambda}) \qquad \forall \lambda \geq 0$

Théorème 2 [7]: (Propriété caractéristique des points cols)

Soit $\bar{x} \in S$ et $\bar{\lambda} \geq 0$, $(\bar{x},\bar{\lambda})$ est un point-col pour $L(x,\lambda)$ si et seulement si :

a) $L(\bar{x},\bar{\lambda}) = \underset{X \in S}{Min} L(x,\bar{\lambda})$

b) $g_i(\bar{x}) \leq 0 \qquad (i = 1,2,\ldots..,m)$

c) $\bar{\lambda}_i g_i(\bar{x}) = 0 \qquad (i = 1,2,\ldots..,m)$

On arrive enfin au théorème de la suffisance de la condition de point-col.

Théorème 3 [7]:

Si $(\bar{x},\bar{\lambda})$ est un point-col de $L(x,\lambda)$ alors $\bar{x}$ est un optimum global de (P).

Démonstration

Il est intéressent de donner ici la démonstration de ce théoréme car elle découle directement de la propriété caractéristique de points-col.

La condition (a) du théoréme 2 entraine :

$$L(\bar{x},\bar{\lambda}) = \underset{X \in S}{Min} L(x,\bar{\lambda}) \Rightarrow f(\bar{x}) + \sum_i \bar{\lambda}_i g_i(\bar{x}) \leq f(x) + \sum_i \bar{\lambda}_i g_i(x) \qquad (\forall x \in S)$$

d'autre part

$$\bar{\lambda}_i g_i(\bar{x}) = 0 \quad (\forall i) \Rightarrow \quad f(\bar{x}) \leq f(x) + \sum_i \bar{\lambda}_i g_i(x) \qquad (\forall x \in S)$$

et comme $\overline{\lambda} \geq 0$ on a :

$$f(\overline{x}) \leq f(x) \qquad \forall x \in S \text{ tel que: } g_i(x) \leq 0 \quad (i = 1,2,.....,m)$$

d'où le théoréme.

Ce résultat est très général et s'applique à n'importe quel programme mathématique convexe, non convexe, où f et g_i sont différentiables ou non, S ensemble continu ou au contraire discret, fini ou dénombrable... d'où son importance dans les méthodes d'optimisation non linéaires.

II.3.4. Méthodes d'optimisation non linéaires avec contraintes [7]:

La plupart des méthodes existantes en programmation non linéaire sous contraintes peuvent se rattacher à deux grandes familles :

- méthodes *primales* (ou directes)
- méthodes *duales* (ou utilisant la notion de dualité).

Les méthodes *primales* se caractérisent par le fait qu'elles opérent directement sur le problème, elles engendrent une séquence de solutions (c'est à dire de points satisfaisant les contraintes) en assurant une décroissance monotone de la fonction à minimiser. Elles présentent un avantage important : si le processus itératif est interrompu elles procurent une solution approchée satisfaisant les contraintes. Par contre, elles sont de mise en œuvre délicate et la convergence globale est souvent difficile à obtenir.

Par opposition, les méthodes *duales* sont plus robustes et la convergence globale est souvent plus facile à obtenir ; en contre partie elles présentent l'inconvénient de ne fournir une solution primale réalisable qu'en fin de convergence.

Nous commencerons par présenter, dans ce qui suit, quelques méthodes de la première famille, puis de la seconde. La liste des méthodes données ci-dessous n'est pas exhaustive ni représentative du point de vu des problèmes traités. Néanmoins il est très intéressant de les présenter, même brièvement, pour pouvoir mieux aborder des méthodes plus générales (méthodes duales) mais qui trouvent leurs origines dans des procédés de résolution moins globaux.

Ajoutons enfin, avant d'y revenir plus tard, que la méthode SQP, utilisée pour traiter notre problème d'optimisation, est une méthode duale.

II.3.4.1. Méthodes primales :

a) Méthode de directions réalisables :

Le problème à résoudre est le suivant :

$$(P) \quad \begin{cases} \text{Minimiser } f(x) \\ \text{sous les contraintes :} \\ g_i(x) \geq 0 \quad i = 1, \ldots\ldots\ldots, m \\ x \in \mathrm{R}^n \end{cases}$$

L'idée est d'adapter le principe des méthodes d'optimisation sans contraintes pour tenir compte des limitations imposées par la présence des contraintes.

On choisit un point de départ satisfaisant les contraintes et on cherche une direction 'y' de déplacement *réalisable* c'est à dire :

(a) un petit déplacement dans cette direction ne fait pas sortir de l'ensemble X des solutions.

(b) La fonction $f(x)$ diminue strictement.

On se déplace alors d'une certaine distance dans cette direction pour obtenir un nouveau point, meilleur que le précédent ;on cherchera le minimum de $f(x)$ dans cette direction sans sortir de l'ensemble X.

Les points successifs sont alors obtenus comme suit :

$$x^{k+1} = x^k + \alpha . y$$

La convergence de cette méthode dépend de manière très critique de la direction y choisie. Il a donc fallu plusieurs reformulations du problème avant que *Topkis* et *Veinott* (1967) suggèrent la formulation suivante :

$$\begin{cases} \text{Minimiser } \xi \\ \text{Sous les contraintes :} \\ \nabla f^T(x^0).y - \xi \leq 0 \\ g_i(x^0) + \nabla g_i^T(x^0).y - u_i \xi \leq 0 \end{cases}$$

Où les inconnues sont y_i (les composantes du vecteur déplacement) ξ, u_i des paramètres positifs fixés et x^0 le point courant auquel on associe la direction 'y' solution optimale du problème reformulé ci-dessus et qui est une *direction réalisable*.

b) Méthode du gradient projeté :

L'idée est ici de projeter à chaque étape le déplacement sur la frontière du domaine des solutions afin d'assurer que le nouveau point obtenu appartienne à l'ensemble des solutions

réalisables. Pour une méthode de gradient on est conduit à projeter le gradient sur la frontière du domaine. Ceci donne un cheminement le long de la frontière dans la direction de la plus forte pente « relative » c'est à dire autorisée par les contraintes. L'un des premiers algorithmes bâtis sur ce schéma est le gradient projeté de *Rosen* (1960) conçu pour des contraintes linéaires et, généralisé pour les contraintes non linéaires.

Considérons le problème général suivant :

$$(P') \quad \begin{cases} \text{Minimiser } f(x) \\ \text{sous les contraintes :} \\ g_i(x) \geq 0 \quad i \in I_1 \\ g_i(x) = 0 \quad i \in I_2 \\ x \in \mathrm{R}^n \end{cases}$$

où I_1 est l'ensemble des indices des contraintes inégalités et I_2 l'ensemble des indices des contraintes égalités.

Soit x un point courant réalisable et notons $I^0(x)$ les indices des contraintes saturées en x (une contrainte g_i est dite saturée en x si $g_i(x) = 0$). La direction de déplacement 'y' au point x est ici calculée comme la projection de $-\nabla f(x)$ sur le sous-espace vectoriel S^0 tangent à la surface definie par:

$$g_i(x) = 0 \quad (\forall i \in I^0(x)).$$

Si on appelle $g^0(x)$ le vecteur $[\, g_{i_1}, g_{i_2}, \ldots\ldots\ldots, g_{i_q}]^{\mathrm{T}}$ où $\{i_1, i_2, \ldots\ldots\ldots, i_q\} = I^0(x)$

et $\dfrac{\partial g^0}{\partial x}(x)$ la matrice définie par $J = \begin{pmatrix} \dfrac{\partial g_{i_1}}{\partial x_1} & \cdots & \dfrac{\partial g_{i_q}}{\partial x_1} \\ \vdots & \cdots & \vdots \\ \dfrac{\partial g_{i_1}}{\partial x_n} & \cdots & \dfrac{\partial g_{i_q}}{\partial x_n} \end{pmatrix}$

jacobien de g^0 évalué en x.

Alors on voit que le sous espace S^0 est défini par l'ensemble des vecteurs 'y' tels que :

$$J.y = 0$$

On obtient alors :

$$\bar{y} = -P^0.\nabla f(x)$$

avec $P^0 = I - \left[\dfrac{\partial g^0}{\partial x}\right]^T \left(\left[\dfrac{\partial g^0}{\partial x}\right] \cdot \left[\dfrac{\partial g^0}{\partial x}\right]^T \right)^{-1} \left[\dfrac{\partial g^0}{\partial x}\right]$ et I est la matrice unité.

On peut maintenant donner l'algorithme du gradient projeté :

(*a*) A l'itération k = 0 on est en x^0

(*b*) A l'itération courante k on est en x^k.

Déterminer l'ensemble $I^0(x)$ des indices des contraintes saturées. Poser $L^0 = I^0(x^k)$.

(*c*) Soit J la matrice jacobienne dont les lignes correspondent aux contraintes saturées $i \in L^0$.

Calculer la matrice de projection P^0.

Calculer $y^k = -P^0.\nabla f(x)$.

Si $y^k = 0$ aller en (*e*).

(*d*) Si $y^k \neq 0$ déterminer $\alpha_{max} = Max \left\{ \alpha / x^k + \alpha y^k \in X \right\}$

puis x^{k+1} tel que $f(x^{k+1}) = \underset{0 \leq \alpha \leq \alpha_{max}}{Max} \left\{ f(x^k + \alpha y^k) \right\}$

Faire $k \leftarrow k+1$ et retourner en (*b*)

(*e*) Soit $u = -\left[A^0 \cdot A^{0T}\right]^{-1} . A^0 . \nabla f(x^k)$

Si $u \geq 0$,

Fin : x^k satisfait les conditions de *Khun* et *Tucker*.

Sinon soit u_i la composante la plus négative de u.

Faire $L^0 \leftarrow L^0 - \{ i \}$ et retourner en (*c*).

Il est important de faire remarquer que la méthode du gradient projeté n'est pas très performante pour les problèmes à contraintes non linéaires.

c) Méthode du gradient réduit généralisé (GRG) :

Cette méthode (*Abadie* et *Carpentier* 1969, *Abadie* et *Guigou* 1970) combine les idées des méthodes de linéarisation et celles du gradient réduit de Wolf. La méthode est applicable pour les fonctions objectif et les contraintes non linéaires. Nous donnerons dans ce qui suit l'essentiel des principes de la méthode.

Le problème à traiter est formulé comme suit, où les contraintes non linéaires sont des contraintes d'égalité:

$$(P) \quad \begin{cases} \text{Minimiser } f(x) \\ \text{sous les contraintes :} \\ g_i(x) = [g_1(x), g_2(x), \ldots\ldots, g_m(x)]^T = 0 \\ x \geq 0 \end{cases}$$

On note $J=\{1,2,\ldots\ldots,n\}$ l'ensemble des indices des variables. Soit x^0 une solution réalisable (solution courante), c'est à dire vérifiant $g(x^0) = 0$ et $x \geq 0$.

Un sous ensemble de variables $B \subset J$, ou $|B| = m$, est appelé une *base* si et seulement si le jacobien:

$$\frac{\partial g}{\partial x_B}(x^0) = \left[\frac{\partial g_i}{\partial x_j}(x^0)\right]_{\substack{i=1,\ldots,m \\ j \in B}} \quad \text{est } \textit{régulière}.$$

Le vecteur x peut alors se partitionner en

$$x = [x_B , x_N]$$

où x_B sont les variables de base (variables indépendantes)
et x_N sont les variables hors-base .

A partir de x^0 on va rechercher une direction de déplacement permettant de diminuer la fonction f tout en satisfaisant les contraintes. Avant de définir la direction de déplacement $y = [y_B , y_N]$, introduisons la notion de *gradient réduit généralisé* u_N.

La variation df de f, pour un déplacement infinitésimal dx compatible avec les contraintes s'écrit :

$$df = u_N \cdot dx_N = \left[\frac{\partial f}{\partial x_N} - \left(\frac{\partial f}{\partial x_B}\right)\left(\frac{\partial g}{\partial x_B}\right)^{-1}\left(\frac{\partial g}{\partial x_N}\right)\right] dx_N$$

La direction de déplacement y_N relative aux variables indépendantes sera alors :

$$\begin{cases} y_j = 0 & \text{si } u_j > 0 \text{ et } x_j = 0 \quad (j \in N) \\ y_j = -u_j & \text{sinon } (j \in N) \end{cases}$$

La direction de déplacement y_B suivant les variables de base se déduit alors de y_N par :

$$y_B = -\left(\frac{\partial g}{\partial x_B}\right)^{-1} \cdot \frac{\partial g}{\partial x_N} \cdot y_N$$

A ce stade de développement de l'algorithme la direction de déplacement est tel que :

- si $y_N = 0$ les conditions de Kuhn et Tucker sont satisfaites au point courant x^0.
- si $y_N \neq 0$, l'algorithme se poursuit et on cherche $\hat{\theta} \geq 0$ minimisant la fonction de θ :

$$\psi(\theta) = f(x^0 + \theta y)$$

Ce qui donne le *pas* avec lequel la recherche va se faire dans la direction y. Mais les contraintes de positivité sur x, imposent :

$$x_j + \theta y_j \geq 0 \qquad (\forall j) \qquad \text{d'où} \quad \theta \leq \theta_{\max} = \underset{y_j < 0}{Min}\left\{-\frac{x_j}{y_j}\right\}$$

On cherchera donc $\hat{\theta} \geq 0$ minimisant $\psi(\theta)$ sur $[0, \theta_{\max}]$ (minimisation unidimensionnelle). On arrive alors à déterminer la direction de recherche et le pas optimal de la recherche et ainsi le point à l'itération suivante.

Cette procédure combinée à d'autres conditions mène à la résolution du problème d'optimisation posé plus haut. Toutes fois la convergence de la méthode vers un point

satisfaisant les conditions nécessaires de Kuhn et Tucker est très difficile à prévoir et n'est pas garantie.

d) Méthode de linéarisation par les plans sécants de *Kelley* (1960) :

Elle s'applique à des problèmes convexes de la forme :

$$(P') \quad \begin{cases} \text{Minimiser } f(x) \\ \text{sous les contraintes :} \\ g_i(x) \geq 0 \quad i=1,\ldots\ldots\ldots,m \\ x \in \mathrm{R}^n \end{cases}$$

Où le problème est non linéaire (fonctions f et/ou g_i) et les fonctions différentiables. Remarquons que l'on peut toujours supposer que la fonction f à minimiser est *linéaire*, dans le cas contraire (P') peut toujours s'écrire, en introduisant une variable supplémentaire y :

$$(P') \quad \begin{cases} \text{Minimiser } y \\ \text{sous les contraintes :} \\ f(x) - y \geq 0 \\ g_i(x) \geq 0 \quad i=1,\ldots\ldots\ldots,m \\ x \in \mathrm{R}^n, y \in \mathrm{R} \end{cases}$$

qui est un problème convexe avec fonction objectif linéaire dans R^{n+1}. Et en notant $y = c.x = \sum_{j=1}^{n} c_j x_j$ le problème se ramène à la forme suivante :

$$(P'') \quad \begin{cases} \text{Minimiser } c.x = \sum_{j=1}^{n} c_j x_j \\ \text{sous les contraintes :} \\ g_i(x) \geq 0 \quad i=1,\ldots\ldots\ldots,m \\ x \in \mathrm{R}^n \end{cases}$$

on note $X=\{x \in \mathrm{R}^n / g_i(x) \leq 0 \ , \ \forall i=1,2,\ldots\ldots,m\}$ l'ensemble des solutions de (P''). Le principe de la méthode des plans sécants est le suivant, à l'itération k :

(*a*) Approximer l'ensemble des solutions X par un polytope Q^k tel que $X \subset Q^k$.

(*b*) résoudre le problème de programmation linéaire:

$$\begin{cases} \text{Minimiser } c.x \\ x \in Q^k. \end{cases}$$

Soit x^k une solution optimale de ce problème.

(c) Tant que x^k n'est pas une solution optimale de (P''), rajouter aux contraintes qui définissent Q^k, une contrainte supplémentaire de la forme :

$$g_i(x^k) + \nabla g_i^T(x^k).(x - x^k) \leq 0$$

(ou i est un indice tel que $g_i(x^k) > 0$, par exemple l'indice i tel que $g_i(x^k)$ soit maximal).

(d) Passer à l'itération $k+1$, avec :

$$Q^{k+1} = Q^k \{x/\ g_i(x^k) + \nabla g_i^T(x^k).(x - x^k) \leq 0\}$$

On démontre, sous les conditions de convexité et de différentiabilité des fonctions, que la suite x^k converge vers la solution du problème (P'') c'est à dire l'optimum de (P'). Cette méthode est intéressante dans la mesure ou elle introduit la notion de linéarisation des fonctions (que nous rencontrerons dans la méthode SQP), et qu'elle représente une méthode utilisable pour résoudre des sous-problèmes d'optimisation.

II.3.4.2. Conclusions sur la convergence des méthodes primales :

Les principaux algorithmes décrits jusqu'à maintenant et applicables aux cas général se caractérisent tous par une mise en ouvre délicate et une convergence pas toujours garantie. De plus la vitesse de convergence vers la solution dépend grandement, pour les méthodes différentiables, de l'éstimation des dérivées premiéres des fonctions du problème étudié, sans exploiter les informations du second ordre ce qui limite, pour certain problèmes, l'éfficacité de ces méthodes. Plus généralement il est préférable d'adopter une des méthodes duales que nous allons décrire dans ce qui suit.

II.3.4.3. Méthodes duales [7] :

Le principe commun à ces méthodes consiste à ramener le problème initial à la résolution d'une suite de problèmes d'optimisation sans contraintes. Nous introduirons le notion fondamentale de *dualité* et montrerons comment cette famille exploite les informations données par les fonctions de *Lagrange*.

Mais avant d'aborder tout cela, introduisons brièvement les méthodes de résolution dites de *pénalités* dont la compréhension est nécessaires pour pouvoir assimiler le principe des méthodes duales. Ajoutons enfin que les méthodes exposées ci-dessous (duales) sont généralement, à une certaine étape de l'évolution des algorithmes, combinées avec d'autres méthodes de résolution de problèmes sans contraintes, d'optimisation unidimensionnelle, d'optimisation linéaire, de pénalités …

II.3.4.3.1. Les méthodes de pénalités :

Considérons le problème d'optimisation suivant :

$$(P) \quad \begin{cases} \text{Minimiser } f(x) \\ \text{sous les contraintes :} \\ g_i(x) \le 0 \quad i = 1, \ldots\ldots\ldots, m \\ x \in \mathbb{R}^n \end{cases}$$

Les contraintes sont, par commodité, négatives mais cela n'impose aucune restriction car pour des contraintes de positivité nous n'aurons qu'à remplacer les contraintes $g_i(x)$ par $h_i(x) = -g_i(x)$.

Soit h ($(R \to R)$ la fonction définie par :

$$\begin{cases} h(y) = 0 & \text{si} \quad y \le 0 \\ h(y) = +\infty & \text{si} \quad y \succ 0 \end{cases}$$

et considérons le problème sans contrainte (problème *pénalisé*) :

$$(PP) \quad \begin{cases} \text{Minimiser } \varphi(x) = f(x) + H(x) \\ x \in R^n \end{cases}$$

où la fonction H est la fonction de *pénalisation* et est définie par :

$$\forall x : \quad H(x) = \sum_{1}^{m} h\left(g_i(x)\right)$$

On démontre que résoudre le problème (P) est équivalent à la résolution du problème d'optimisation sans contraintes (PP).

Pour contourner des problèmes de résolutions de (PP) plusieurs formulations des fonctions de pénalisation ont été proposées, nous ne donnerons que la méthode de *pénalités intérieures* proposées par *Fiacco* et *McCormick* (1968).

Supposons que :

- l'interieur de X (ensemble des solutions du problème (P)) est non vide.
- tout point de la frontiére de X est limite d'une suite de points appartenant à l'intérieur de X.

Soit alors la fonction de *pénalisation intérieure* $B(x)$ (aussi appelée fonction barrière) définie par :

$$B(x) = -\sum_{1}^{m} \frac{1}{g_i(x)}$$

cette fonction vérifie :

- $B(x) \geq 0$
- $B(x) \rightarrow +\infty$ lorsque x tend vers la frontiére de X
- $B(x)$ est continue sur l'intérieur de X lorsque les fonctions g_i le sont.

Considérons alors la fonction :

$$\psi(x,t) = f(x) + t.B(x)$$

ou t, réel strictement positif, est appelé le *coefficient de pénalité*.

Le principe de la méthode de pénalité intérieure est de choisir une pénalité t_1 et de minimiser la fonction $\psi(x,t_1)$ à partir d'un point $x^0 \in \text{int}(X)$, on obtiendra alors un point $x^1 \in \text{int}(X)$ si le produit $t_1.B(x^1)$ est assez petit alors x^1 est une bonne approximation de l'optimum de f sur X et les calculs s'arrêtent ; sinon on choisira une valeur $t_2 < t_1$ et on recherchera à nouveau l'optimum de la fonction $\psi(x,t_2)$ en partant cette fois-ci de x^1. Le processus est répété jusqu'à obtention d'une approximation acceptable de l'optimum de (P).

Remarquons que cette méthode est dite de pénalité intérieur en raison du fait que les différents points x^k appartiennent tous à X.

II.3.4.3.2. Dualité Lagrangienne classique :

Considérons un problème d'optimisation du type :

$$(P) \quad \begin{cases} \text{Minimiser } f(x) \\ \text{sous les contraintes :} \\ g_i(x) \leq 0 \quad i=1,\ldots\ldots\ldots,m \\ x \in \mathbb{R}^n \end{cases}$$

La fonction de Lagrange $L(x,\lambda)$ associée est :

$$L(x,\lambda) = f(x) + \sum_{i \in I} \lambda_i g_i(x)$$

On a vu dans le chapitre précédant que le problème (P) peut être résolu si l'on sait déterminer un point-col de la fonction de Lagrange c'est à dire un couple $(\bar{x},\bar{\lambda})$ vérifiant :

$$\begin{cases} L(\bar{x},\bar{\lambda}) = \underset{x \in S}{Min} L(x,\bar{\lambda}) \\ g(\bar{x}) \leq 0 \\ \bar{\lambda}_i . g_i(\bar{x}) = 0 \qquad i = 1,\ldots\ldots\ldots\ldots,m \end{cases}$$

Définissons maintenant pour $\lambda \geq 0$ la fonction $w(\lambda)$ par

$$w(\lambda) = \underset{x \in S}{Min} \{L(x,\lambda)\}$$

Alors la recherche d'un point-col se fait en résolvant le problème :

$$(D) \qquad \begin{cases} \underset{\lambda}{Max}\ w(\lambda) = \underset{\lambda}{Max} \left\{ \underset{x \in S}{Min} L(x,\lambda) \right\} \\ \lambda \in R^{m+} \end{cases}$$

(D) est appelé le *problème dual* de (P). Par opposition (P) est appelé le *problème primal* et w est la *fonction duale*.

De la définition du problème dual et de la fonction duale, il résulte une série de propriétés utilisées pour résoudre les problèmes d'optimisation de la forme (P), nous ne donnerons donc qu'une propriété, celle qui nous paraît la plus fondamentale :

Propriété [7] (Théoréme de la dualité)

(a) Si le problème (P) admet un point-col (x^*,λ^*) alors, on a:

$$\text{Max(D)} = w(\lambda^*) = f(x^*) = \text{Min (P)}$$

Autrement dit, la valeur optimal du problème primal (P) est égale à la valeur optimal du problème dual (D).

(b) Réciproquement, s'il existe x^* solution de (P) et $\lambda^* \geq 0$ tels que :

$$w(\lambda^*) = f(x^*)$$

alors (P) admet un point-col et (x^*,λ^*) est un tel point.

Lorsque les problèmes n'admettant pas de points-col il est toujours possible d'obtenir une solution approchée du problème (P) en résolvant le *dual* (D). Ceci est surtout vrai pour les problèmes pratiques où la solution peut être approchée à ε prés, ceci ;bien entendu; sous certaines conditions mathématiques qui seraient trop longues à aborder ici. Ajoutons aussi que pour résoudre le problème *dual* il est possible d'utiliser, selon le cas, l'une des méthodes de résolution primales car le problème de minimisation est linéaire sans contraintes et les algorithmes de résolution s'y rattachants sont de mise en oeuvre simple et efficace, ceci explique; entre autres; le succés des méthodes duales.

II.3.4.3.3. Programmation non convexe et lagrangien augmenté :

Pour résoudre les problèmes d'optimisation posés jusqu'ici , nous avons décrit deux types de méthodes (qui remplacent le problème par une séquence de problèmes d'optimisation sans contraintes):

- *les méthodes de pénalités*
- *les méthodes utilisant la dualité lagrangienne*.

Ces méthodes présentent l'inconvénient que pour des fonctions non convexes leur utilisation devient très délicates, c'est pour cela qu'une approche modifiée (plus utilisée) est de combiner les deux méthodes citées plus haut, nous vérons que ceci est le cas pour la méthode SQP que nous présenterons dans le prochain chapitre. Mais tout d'abord définissons ce qu'est un problème convexe et l'importance de cette notion fondamentale dans les méthodes dites classiques.

Programmes convexes :

On dit qu'un problème de programmation mathématique est convexe s'il consiste à minimiser une *fonction convexe* sur un *domaine convexe*.

$$\begin{cases} f(x) \text{ est convexe} \\ \text{sous les contraintes :} \\ g_i(x)\ (i=1,\ldots\ldots\ldots,m) \text{ convexe} \\ S \subset \mathrm{R}^n \text{ est convexe} \end{cases}$$

Avec les définitions suivantes :

- Un ensemble $S \subset \mathrm{R}^n$ est dit *convexe* si et seulement si :

$$\forall x \in S$$
$$\forall y \in S \Rightarrow \lambda x + (1-\lambda) y \in S$$
$$\forall \lambda (0 \le \lambda \le 1)$$

D'une façon générale (et géométriquement), on peut dire que S est convexe si et seulement si pour deux points quelconques x et y pris de S, le segment $[x, y]$ tout entier est contenu dans S.

- On dit qu'une fonction $f : R^n \to R^n$ définie sur S convexe, est *convexe*, si elle vérifie :
  $\forall x \in S$, $\forall y \in S$, $\forall \lambda (0 \le \lambda \le 1)$:

$$f(\lambda x + (1-\lambda)y) \le \lambda f(x) + (1-\lambda) f(y)$$

Alors la *propriété fondamentale* d'un problème convexe est la suivante:
Pour un problème convexe, tout optimum local est un optimum global.

Il ressort de ce qui précede qu'un problème non convexe a soit des fonctions non convexes et/ou est défini sur un ensemble non convexe. Ce cas est souvent rencontré en pratique; il est donc nécéssaire de définir des méthodes applicables aux problèmes non convexes et assurant de bons résultats, on parle alors de programmation non convexe applicable méme pour les problèmes qui le sont.

C'est en combinant les approches de *pénalités* et de *dualité* que le problème général :

$$(P) \quad \begin{cases} \text{Minimiser } f(x) \\ \text{sous les contraintes :} \\ g_i(x) \le 0 \quad i = 1, \ldots\ldots\ldots, m \\ x \in R^n \end{cases}$$

revient à minimiser suivant x à chaque étape le *lagrangien augmenté* $L(x,\lambda,r)$:

$$L(x,\lambda,r) = f(x) + \sum_{1}^{m} G(g_i(x), \lambda_i, r)$$

ou r désigne la pénalité (fixe à chaque itération)
g_i les contraintes du problème
λ_i le vecteur des multiplicateurs (connu à chaque itération)
G une fonction de (g_i, λ_i, r) définie différement selon les algorithmes.

Remarquons que la résolution de ce programme peut se faire à l'aide de l'une des méthode de minimisation vues précedament, ceci en définissant le dual du problème (P) et en revenant à la notion de point-col (qui prend un sens quelque peu différent dans le cas non convexe).

Ajoutons enfin que cette approche peut étre intégrée dans une famille de méthodes plus générale appelée *méthodes des multiplicateurs*. Ces méthodes utilisent des représentations lagrangiens généralisés pour formuler les problèmes duals.

II.4 Méthode SQP

INTRODUCTION :

Les méthodes d'optimisations, nous l'avons vu, n'ont cessé d'évoluer pour arriver à traiter des problèmes de plus en plus variés et complexes. C'est dans ce grand mouvement de développement théorique des méthodes de résolution et avec l'apport considérable apporté par des calculateurs de plus en plus puissants, que la méthode SQP a vu le jour à la fin des années 1970 et le début des années 1980 et n'a cessé de s'enrichir depuis.

La programmation quadratique séquentielle, ou méthodes SQP (Sequential Programming Method), est l'un des plus puissants programmes parmi tous les algorithmes dont nous disposons aujourd'hui pour résoudre les problèmes différentiables et non linéaires de la forme (1), c'est ainsi l'un des algorithmes les plus utilisés pour la résolution de problèmes d'optimisation pratiques.

L'origine théorique des méthodes SQP est parfaitement décrite par Stoer (1985) et Spellucci (1993) , leurs convergence mise en évidence et vérifiée par Han (1976,1977), Powell (1978) et Schittkowski (1983) et étudiées d'un point de vue plus pratique par Papalambros, Wilde (1988) et Edgar, Himmelblau (1988). Leur excellente performance numérique a été testée et comparée avec les autres méthodes par Schittkowski (1980).

Cette méthode peut être qualifiée, comme nous allons le voir, de synthèse des méthodes dite classiques. En effet la méthode SQP est une méthode duale, différentiable et qui fait appel aux méthodes et concepts introduits dans le chapitre précédent.

Nous commencerons, dans ce qui suit, par donner les fondements théoriques de la méthode, nous aborderons ensuite la convergence de la méthode et finirons par donner l'algorithme SQP. Il sera donner en fin de chapitre quelques conclusions et remarques concernants les différentes méthodes d'optimisation.

II.4.1. Méthode de la Programmation quadratique séquentielle :

Considérons le problème d'optimisation général suivant : minimiser une fonction objective f sous contraintes d'égalité et d'inégalité non linéaires

$$x\in R^n : \quad \begin{cases} min\ f(x) \\ g_j = 0\ ; \quad j=1,\ldots\ldots,m_e \\ g_j \geq 0\ ; \quad j=m_e+1,\ldots\ldots,m \\ x_l \leq x \leq x_u \end{cases} \tag{1}$$

où x est un vecteur de dimension N. Il est supposé que toutes les fonctions $f(x)$ et $g_j(x)$ du problème ($j=1,\ldots\ldots,m$)sont continûment différentiable sur R^n mais nous ne supposons aucune structure mathématique supplémentaire des fonctions modèles, les fonctions peuvent être convexes ou non.

La programmation quadratique séquentielle est une méthode générale de résolution des problèmes d'optimisation non linéaire , sous les conditions suivant :

- *Le problème n'est pas trop grand.*
- *Les fonctions et gradients peuvent être évaluées avec une assez haute précision.*
- *Le problème est régulier et bien évalué.*

Afin de faciliter la notation dans ce paragraphe nous ne noterons pas les bornes inférieures et supérieures, nous obtenons ainsi la formulation suivante du problème :

$$x\in R^n : \quad \begin{cases} \min\ f(x) \\ g_j = 0\ ; \quad j=1,\ldots\ldots,m_e \\ g_j \geq 0\ ; \quad j=m_e+1,\ldots\ldots,m \end{cases} \tag{2}$$

On suppose que les fonctions du problèmes $f(x)$ et $g_j(x), j=1,\ldots\ldots,m,$ sont toutes continûment différentiables sur R^n et qu'aucune restriction n'est faite sur la structure mathématique de ces fonctions.

L'idée de base de la méthode est de formuler et de résoudre à chaque itération un sous problème de *programmation quadratique* (fonction objectif non linéaire et contraintes linéaires) qui est obtenu par la linéarisation des contraintes et l'approximation quadratique de la fonction de Lagrange :

$$L(x,\lambda) = f(x) - \sum_{1}^{m} \lambda_j g_j(x) \tag{3}$$

où $x\in R^n$ est la variable primale et $\lambda = (\lambda_1, \ldots\ldots\ldots, \lambda_m)^T \in R^m$ le vecteur multiplicateur. On voit

bien que la formulation puis la résolution du problème dual constitue la base de la méthode SQP.

Afin de formuler le sous problème de programmation quadratique nous procéderons à l'approximation de la solution à une itération donnée $x_k \in R^n$, à l'approximation $v_k \in R^n$ du vecteur multiplicateur (constitué des multiplicateurs) et de $B_k \in R^{n \times n}$ une approximation du *Hessien* de la fonction de *Lagrange* , notons que le *hessien* est défini comme suit :

$$B = \begin{pmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \cdots & \frac{\partial h_1}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial h_n}{\partial x_n} & \cdots & \cdots & \frac{\partial h_n}{\partial x_n} \end{pmatrix} \qquad \text{et} \qquad h_i = \frac{\partial L}{\partial x_i}$$

où on désigne par h_i ($i = 1, \ldots\ldots, n$) le gradient de la fonction de *Lagrange* par rapport à la *i-éme* variable et par x_i ($i = 1, \ldots\ldots, n$) les variables du problème. Le *hessien* comporte donc des éléments du second ordre qui nous renseignent sur la courbure de la fonction de *Lagrange*

Avant d'aller plus loin dans la formulation de l'algorithme de résolution, précisons que le hessien (B_k) de la fonction de Lagrange est approximé de manière à bien évaluer les dérivées du second ordre (la convergence et l'efficience de la méthode SQP en dépend grandement), ceci en utilisant l'approche standard de calcul du Hessien par la formule dite quasi-Newtonienne BFGS (des noms de ses auteurs) développée indépendamment par Broyden, Fletcher, Goldfard et Shanno.

Notre problème est alors amené à la résolution du '*problème de programmation quadratique*' suivant :

$$d \in R^n \quad : \quad \begin{cases} \min \frac{1}{2} d^T B_k d + \nabla f(x)^T d \\ \nabla g_j(x_k)^T d + g_j(x_k) = 0, \; j = 1, \ldots\ldots, m_e \\ \nabla g_j(x_k)^T d + g_j(x_k) \geq 0, \; j = m_e + 1, \ldots\ldots, m \end{cases} \qquad (4)$$

où 'd' est le *vecteur direction* variable de ce sous problème de minimisation et soit 'd_k' la solution optimal de ce problème, alors la nouvelle itération 'x_{k+1}' est obtenue par :

$$\begin{pmatrix} x_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ v_k \end{pmatrix} + \alpha_k \begin{pmatrix} d_k \\ \lambda_k - v_k \end{pmatrix} \tag{5}$$

où $\alpha_k \in]0,1]$ est pas à l'itération donnée.

La matrice B_k est définie positive mais malgré cela il est possible que le problème (4) ne soit pas résolu du fait d'une mauvaise formulation ou mise en œuvre des contraintes. Pour remédier à cette possibilité d'échec de la résolution une variable additionnelle $\delta \in R$ est introduite, ce qui nous mène vers une programmation quadratique modifiée.

Le pas 'α_k' est nécessaire en (5) pour renforcer la convergence globale de la méthode SQP, c'est à dire l'approximation d'un point satisfaisant la condition nécessaire d'optimalité de Karush-Kuhn-Tucker quand l'algorithme démarre d'un point initial arbitraire. En pratique il n'est demandé de fournir à l'algorithme que le point initial $x_0 \in R^n$, v_0 étant pris égal à zéro et B_0 égale à la matrice identité.

On voit bien, comme dans le reste des méthodes d'optimisation faisant appel aux différentielles des fonctions et contraintes étudiées, que la convergence de la méthode SQP dépend de la direction de recherche 'd_k' et du pas 'α_k' à l'itération 'k'.

La direction de recherche 'd_k' est déterminée, comme nous l'avons vu, par la résolution d'un problème d'optimisation quadratique ainsi la direction de recherche se trouve elle même être optimale parmi toutes les directions possibles (c'est à dire, d'un point de vu mathématique, une infinité).La résolution pratique d'un tel problème peut se faire par différentes méthodes adaptées à la formulation quadratique (4) car, comme nous le voyons bien, la fonction but est non linéaire mais les contraintes d'optimisation, elles, sont linéaires. Nous pouvons dés lors opter pour l'une des méthodes mentionnées dans les paragraphes précédents.

Il ne nous reste donc que le paramètre pas 'α_k' à définir de manière efficiente. Le pas doit satisfaire la condition de décroissance d'une fonction dite 'Mérite' formulée de la manière suivante :

$$\phi_r(\alpha) = \psi_r\left[\begin{pmatrix} x \\ v \end{pmatrix} + \alpha \begin{pmatrix} d \\ u - v \end{pmatrix}\right] \tag{6}$$

Et ou $\psi_r(x,v)$ est une fonction pénalité convenablement choisie comme étant la fonction de 'Lagrange augmentée' :

$$\psi_r(x,v) = f(x) - \sum_{j \in J} (v_j g_j(x) - \frac{1}{2} r_j g_j(x)^2) - \frac{1}{2} \sum_{j \in K} v_j^2 / r_j \tag{7}$$

avec : $J=\{1,\ldots,m_e\} \cup \{j : m_e<j<m , g_j(x)\leq v_j/r_j\}$

$K=\{1,\ldots,m\}$

$f(x)$ la fonction but

$g_j(x)$ la *j-éme* contrainte

r_j la *j-éme* pénalité

La fonction but se trouve alors pénalisée dés qu'à une itération 'k' la solution quitte le domaine des solutions. La pénalité correspondante r_j , $j=1,\ldots,m$ qui contrôle le degré de violation des contraintes doit être choisie de manière à garantir une direction de descente de la fonction mérite, cette exigence exprimée mathématiquement devient ;

$$\phi'_{r_k}(0)=\nabla\psi_{r_k}(x_k,v_k)\begin{pmatrix} d_k \\ u_k-v_k \end{pmatrix}<0 \qquad (8)$$

A ce stade de l'algorithme et posant

$$\alpha_k=\sigma\beta^i$$

'i' étant une variable, β et σ des constantes. Le paramètre pas α_k est déterminé de manière unique en satisfaisant la condition générale d'Armijo, cette condition est fondamentale car elle impose au pas :

- de ne pas être trop grand, sinon l'algorithme risque d'avoir un comportement oscillatoire.
- de ne pas être trop petit, sinon l'algorithme risque de converger prématurément.

Cette condition s'énonce comme suit :

$$\phi_r(\sigma\beta^i)\leq\phi_r(0)+\sigma\beta^i\mu\phi'_r(0) \qquad (9)$$

où les constantes sont de l'ordre suivant $0<\mu<0.5$, $0<\beta<1$ et $0<\sigma\leq1$.

Pratiquement l'algorithme commence (à chaque itération) par poser $i=0$ et diminuer i jusqu'à ce que (9) soit satisfaite pour la première fois alors et seulement alors $i=i_k$ et le pas de l'itération courante k sera :

$$\alpha_k=\sigma\beta^{i_k}$$

Le pas étant déterminé il est nécessaire d'apporter quelques commentaires concernent la stratégie de calcul de ce dernier car en réalité le paramètre μ (dit paramètre test) de la condition d'Armijo est très petit, de l'ordre de $\mu = 0.0001$, de même le choix du paramètre β (dit de réduction) est en pratique lié à la valeur de la pente de la fonction mérite à l'itération courante. Ainsi si la valeur de β est très petite la ligne de recherche se terminera très rapidement mais, comme souligné plus haut, il risque d'y avoir convergence prématurée

d'une part et un très grand nombre d'itération d'autre part (le calcul sera alors coûteux). Aussi, une valeur élevée de β nécessite un nombre élevé d'appel des fonctions de calcul (évaluation des fonctions but et contraintes ainsi que les gradients qui leur sont associés).

C'est pour ces raison qu'un compromis est réalisé en adoptant une interpolation polynomiale (quadratique) à partir de $\phi_r(0)$, $\phi'_r(0)$ et $\phi_r(\alpha_i)$, avec α_i le pas actuel de la ligne de recherche, il est facile de minimiser cette interpolation et de prendre comme pas définitif de calcul le maximum de cette valeur et de celle fournie par la condition d'Armijo. On voit bien que le choix du pas ne peut remplir les conditions citées plus haut.

II.4.2. Convergence superlinéaire de la méthode SQP :

On dit qu'un algorithme converge de manière superlinéaire quand il satisfait la condition suivante :

$$\frac{\left\|x^{k+1}-x^{*}\right\|}{\left\|x^{k}-x^{*}\right\|}\rightarrow 0 \quad \text{quand} \quad k\rightarrow\infty$$

Ou k désigne le nombre d'itération, x^k et x^{k+1} deux points consécutifs et x^* la solution optimale.

La méthode SQP a une convergence superlinéaire au voisinage de la solution, c'est une propriété essentielle car elle nous renseigne directement sur la vitesse de convergence de l'algorithme et c'est aussi un paramètre de comparaison ente différents algorithme.

II.4.3. L'algorithme de la méthode SQP :

L'algorithme de la méthode SQP est alors le suivant :

(a) à l'itération *0* on est en x^0, avec $v_0 = 0$, $B_0 = I$ et $u_0 > 0$.
(b) à l'itération k on est en x^k.
(c) calculer f, ∇f, g_i et B_k à l'itération courante k.
(d) formuler le problème quadratique à l'itération courante :

$$d\in R^n \quad : \quad \begin{cases} \min \frac{1}{2} d^T B_k d + \nabla f(x)^T d \\ \nabla g_j(x_k)^T d + g_j(x_k) = 0, \; j=1, \ldots, m_e \\ \nabla g_j(x_k)^T d + g_j(x_k) \geq 0, \; j=m_e+1, \ldots, m \end{cases}$$

résoudre le problème à l'aide d'une méthode primale ou duale (gradient généralisé).

(e) formuler la fonction mérite :

$$\phi_r(\alpha) = \psi_r\left[\begin{pmatrix} x_k \\ v_k \end{pmatrix} + \alpha_k \begin{pmatrix} d_k \\ u_k - v_k \end{pmatrix}\right]$$

en fonction de d_k , x_k , u_k , v_k .

(f) fixer β et σ , constantes à l'itération k, poser $i = 0$.

Varier $i<0$ jusqu'à satisfaire une première fois la condition d'Armijo :

$$\phi_r(\sigma\,\beta^i) \le \phi_r(0) + \sigma\,\beta^i\,\mu\,\phi'_r(0)$$

Poser $\alpha_k = \sigma\,\beta^i$

(g) Si le couple (x_k , v_k) est un point-col de la fonction de lagrange, alors le vecteur x_k est un optimum global du problème.

Sinon faire : $\begin{pmatrix} x_{k+1} \\ v_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ v_k \end{pmatrix} + \alpha_k \begin{pmatrix} d_k \\ u_k - v_k \end{pmatrix}$ et retourner en *(b)*.

II.5. Conclusions sur les méthodes d'optimisation :

Nous avons, dans les paragraphes précédents, abordé différentes méthodes et approches pour la résolution des problèmes d'optimisation. Nous avons pu voir que le type de problème : linéaire ou non linéaire, différentiable ou non, convexe ou pas... détermine tout d'abord la classe de méthodes à choisir, puis la méthode approprié. Ce second choix est souvent dicté par les conditions de convergence de la méthode, de la difficulté de sa mise en œuvre, du nombre d'appels de fonction nécessaires à son exécution et de son coût (en temps et moyens).

Une autre conclusion est que les différentes méthodes sont souvent imbriquées et liées. Ceci est surtout vrai pour les problèmes non linéaires avec contraintes qui peuvent faire appel aux algorithmes de résolution des problèmes linéaires, sans contraintes, unidimensionnelle... Il est donc indispensable d'aborder ces méthodes, d'un double point de vu pédagogique et pratique, avant de traiter les méthodes non linéaires avec contraintes.

Ajoutons pour terminer, que la méthode SQP se révèle être, comme nous allons le vérifier, un excellent moyen d'optimisation de part sa robustesse (la variété des problèmes traités par

cette méthode avec efficacité étant très grande) et sa puissance de convergence. En effet la méthode SQP est une synthèse des différentes approches de la programmation mathématique dans la mesure ou elle fait appel aux principales méthodes développés précédemment et donc à l'efficacité de chacune d'elles.

III.1. Dérivation numérique :

Introduction :

De nombreux concepts en physique et en génie utilisent les dérivées, et, comme les données numériques sont souvent des valeurs de la fonction elle-même et non sa dérivée, il est souvent nécessaire d'estimer la dérivée de manière approchée. Ceci est vrai dans notre cas car la méthode d'optimisation adoptée pour traiter notre problème fait appel au calcul du gradient de la fonction objectif et au jacobien des contraintes.

Nous commencerons par formuler la dérivation approchée que nous avons adopté et la commenterons, pour ensuite montrer comment elle a été mise en œuvre du point de vu algorithmique (en mettant l'accent sur le cas qui nous intéresse à savoir l'optimisation des structures en treillis)

III.1.2. Etablissement de la formule de dérivation [9,12,13] :

Soit la fonction $f(x)$ définie de R^n dans R , supposée continue et différentiable jusqu'à l'ordre *cinq* et soit :

$$\frac{\partial f}{\partial x_i}$$

sa dérivée partielle par rapport à la *i-éme* composante du vecteur x. Nous allons établir, à partir du développement en série de *Taylor* de la fonction f au voisinage de x, l'expression de la dérivée partielle première de la fonction.

Pour cela développons $f(x)$ jusqu'au cinquième ordre au voisinage de x pour différents pas tous de la forme $k\Delta$, avec $k \in I = \{-1, +1, -2, +2\}$, le développement en série de *Taylor* étant le suivant :

$$f(x_0+\Delta) = f(x_0) + \frac{\Delta}{1!} f'(x_0) + \ldots\ldots\ldots + \frac{h^n}{n!} f^{(n)}(x_0) + \theta(0)$$

d'ou

$$f(x_i-\Delta) = f(x_i) - \Delta \frac{\partial f}{\partial x_i} + \frac{\Delta^2}{2} f^{(2)}(x_i) - \frac{\Delta^3}{6} f^{(3)}(x_i) + \frac{\Delta^4}{24} f^{(4)}(x_i) - \frac{\Delta^5}{120} f^{(5)}(x_i) + \theta(0)$$

$$f(x_i+\Delta) = f(x_i) + \Delta \frac{\partial f}{\partial x_i} + \frac{\Delta^2}{2} f^{(2)}(x_i) + \frac{\Delta^3}{6} f^{(3)}(x_i) + \frac{\Delta^4}{24} f^{(4)}(x_i) + \frac{\Delta^5}{120} f^{(5)}(x_i) + \theta(0)$$

$$f(x_i-2\Delta)=f(x_i)-2\Delta\frac{\partial f}{\partial x_i}+4\frac{\Delta^2}{2}f^{(2)}(x_i)-8\frac{\Delta^3}{6}f^{(3)}(x_i)+16\frac{\Delta^4}{24}f^{(4)}(x_i)$$
$$-32\frac{\Delta^5}{120}f^{(5)}(x_i)+\theta(0)$$

$$f(x_i+2\Delta)=f(x_i)+2\Delta\frac{\partial f}{\partial x_i}+4\frac{\Delta^2}{2}f^{(2)}(x_i)+8\frac{\Delta^3}{6}f^{(3)}(x_i)+16\frac{\Delta^4}{24}f^{(4)}(x_i)$$
$$+32\frac{\Delta^5}{120}f^{(5)}(x_i)+\theta(0)$$

avec $\theta(0)$ une fonction qui tend vers zéro quand h tend vers zéro.

en effectuant quelques opérations sur les développement; on aura :

$$16[f(x_i+\Delta)-f(x_i-\Delta)]=32\Delta f'(x_i)+\frac{16}{3}\Delta^3 f^{(3)}(x_i)+\frac{16}{60}\Delta^5 f^{(5)}(x_i)+\theta(0) \quad (1)$$

$$2[f(x_i+\Delta)-f(x_i-\Delta)]=-8\Delta f'(x_i)-\frac{16}{3}\Delta^3 f^{(3)}(x_i)-\frac{16}{60}\Delta^5 f^{(5)}(x_i)+\theta(0) \quad (2)$$

en faisant (1)+(2) on aura :

$$f'(x_i)=\frac{16f(x_i+\Delta)-16f(x_i-\Delta)+2f(x_i-2\Delta)-2f(x_i+2\Delta)}{24\Delta}+\frac{1}{30}\Delta^4 f^{(5)}(x_i)$$

qui est l'expression de la dérivée partielle de la fonction par rapport à la *i-éme* variable et où le terme :

$$R(x_i)=\frac{1}{30}\Delta^4 f^{(5)}(x_i)$$

est le *reste* ou l'*erreur* sur la dérivée. Cette quantité ne donne pas l'erreur exacte (il faudrait pour cela développer la fonction à un ordre infini) mais nous renseigne fidèlement sur l'ordre de grandeur de cette erreur. Nous voyons bien que le reste est de l'ordre du produit $\Delta^4 f^{(5)}(x_i)$, c'est à dire du quatrième ordre.

III.1.3 Erreur sur la dérivée et choix du pas [9,15] :

Les erreurs lors du calcul de la dérivée d'une fonction de manière approchée sur ordinateur sont de deux origines différentes :

- une imprécision due à l'approximation de la dérivée, dite de troncation et notée ε_t à laquelle on associe l'erreur e_t.
- une imprécision due à la précision de stockage sur la mémoire des variables qu'on notera ε_r à laquelle on associe l'erreur e_r.

L'erreur de troncation e_r est le plus grand terme du développement en série de Taylor de la fonction f, on a vu que ce terme était :

$$R(x_i) = \frac{1}{30}\Delta^4 f^{(5)}(x_i)$$

On voit bien e_r est de l'ordre de $e_r \sim |\Delta^4 f^{(5)}(x_i)|$.

L'erreur due à la machine est quant à elle évaluée par la formule suivante $e_r = \varepsilon_f |f(x_i)/h|$ où ε_f est la précision avec laquelle la fonction est calculée .

Le choix du pas doit donc minimiser la somme $e_r + e_r$, ceci s'obtient en dérivant cette somme et en résolvant l'équation : $d(e_r + e_r)/dx = 0$.

La dérivée de la somme est :

$$\frac{d(e_r + e_t)}{dx} = -\frac{\varepsilon_f |f(x_i)|}{\Delta^2} + 4\Delta^3 |f^{(5)}(x_i)|$$

elle s'annule pour la valeur de : $\Delta = (\varepsilon_f)^{1/5} \left(\frac{|f(x)|}{4|f^{(5)}(x)|} \right)^{1/5}$

cette valeur optimale du pas fournit l'erreur fractionnelle suivante :

$$(e_r + e_t) = \varepsilon_f^{\frac{4}{5}}$$

La valeur de ε_f est prise, pour une fonction simple, égale à la précision de calcul de la machine, qui est, en double précision, égale à 10^{-14} , la précision de calcul de la dérivée sera alors de l'ordre de 10^{-11}.

Cette haute précision est en pratique très dure à atteindre pour deux raisons :

- l'expression du pas donnée plus haut fait intervenir les dérivées de la fonction du cinquième ordre, l'évaluation de ces dérivées est alors délicat à la mise en œuvre et engendre, lors des calculs successives des dérivées, des erreurs qui se cumulent, d'ou une perte considérable en précision. De plus il n'est pas intéressant de calculer les dérivées cinquièmes pour arriver à évaluer la dérivée première d'une fonction.
- la seconde difficulté est que la précision de calcul de la fonction f n'est pas toujours égale à la précision de la machine. Ceci est vrai pour les fonctions qui font intervenir, durant leur évaluation, plusieurs étapes de calcul et accumulent par la suite les erreurs successives dues à la machine.

C'est pour ces raisons que notre choix du pas est quelques peu différent de ce qui a été donné plus haut. En effet, pour un programme de calcul des structures par éléments finis intégré dans une procédure d'optimisation de structures, l'usage est de prendre :

$$\Delta = \|x\| h$$

où $\|x\|$ désigne la norme du vecteur, et h une constante prise égale à 10^{-7}.

Cette formule s'inspire de l'étude du pas optimal pour une approximation de la dérivée du premier ordre :

$$h \approx \sqrt{\varepsilon_f} \cdot \sqrt{\frac{f}{f''}} \approx \sqrt{\varepsilon_f}\, x_c$$

La valeur de $h=10^{-7}$ est prise égale à la racine carrée de la précision de la machine. La norme de x est prise à la place de x_c car les deux valeurs sont proches de plus on évite de la sorte de calculer la dérivée seconde de f.

Ce choix, combiné avec la formule approchée de la dérivée que nous avons adopté plus haut, mène à une bonne précision de calcul et fourni, comme nous le verrons plus tard, des résultats convaincants.

III.1.4. Calcul de la matrice jacobienne :

Le jacobien étant une matrice constituée selon ses colonnes par les gradients de fonctions $f_i(x)$ $(i=1, \ldots\ldots\ldots, n)$, il est simple de le calculer. En effet si J désigne cette matrice, alors :

$$J = (\nabla f_1, \nabla f_2 \ldots\ldots\ldots, \nabla f_n)$$

en utilisant la formule de la dérivée donnée ci-dessus pour chaque fonction f_i nous arrivons à construire le jacobien désiré avec la même précision que celle du gradient.

III.1.5. Algorithme de dérivation :

La procédure adoptée pour le calcul du gradient et du jacobien adopte la formulation donné plus haut. Il peut sembler que le nombre de fonctions calculées est élevé à chaque itération (au nombre de quatre) mais cela est nécessaire si l'on veut atteindre un bon niveau de précision, chose indispensable pour le fonctionnement et la convergence de la méthode d'optimisation SQP.

Pour mieux cerner la procédure, nous donnons ci-dessous l'algorithme de dérivation :

(a) Initialiser le jacobien, poser $J(i,j) = 0$

(b) lire le vecteur $x(i)$ $(i = 1,2, \ldots\ldots\ldots, n)$

faire $\Delta = \|x\|.h$

(c) $j = 1$

calculer, pour chaque fonction f_j le gradient ∇f_j :

$$f_j(x_i - \Delta),\ f_j(x_i + \Delta),\ f_j(x_i - 2\Delta),\ f_j(x_i + 2\Delta)$$

$$\nabla f_j(i) = \frac{\partial f_j}{\partial x_i} = \frac{16f(x_i+\Delta) - 16f(x_i-\Delta) + 2f(x_i-2\Delta) - 2f(x_i+2\Delta)}{24\Delta}$$

(*d*) affecter à la *j-éme* colonne du jacobien le gradient ∇f_j

(*e*) $j = j+1$, retourner en (*c*).

Terminons ce paragraphe par quelques remarques relatives au calcul des fonction f_j aux points perturbés. Le calcul de la quantité $\nabla f_j(i)$ revient, dans notre cas d'optimisation et de calcul des treillis, à :

- perturber d'une quantité Δ' (égale au pas adopté) la variable d'optimisation (selon les cas, l'aire de la section droite ou les dimensions de cette section),
- constituer le vecteur *variables* perturbé et appeler un sous programme de calcul de structures qui, en retour, donne les contraintes (de compression ou de traction) dans chaque barre,
- sélectionner la *i-éme* composante qui va entrer dans l'évaluation du gradient,
- refaire l'opération pour le pas suivant car pour chaque élément du gradient il est nécessaire d'appeler le sous programme de calcul quatre fois.

C'est ainsi, de gradient en gradient, qu'on construit le jacobien des contraintes d'optimisation. Ce point, important, sera abordé plus en détail dans la section consacrée à la mise en ouvre du sous-programme d'optimisation.

III.2. Résolution des systèmes d'équations par la méthode d'élimination de Gauss [2]:

Cette méthode est utilisée dans la résolution des systèmes d'équations, elle est constituée de deux étapes :

- Triangularisation :

Cette étape consiste à transformer le système d'équations

$$[k]\{u_n\} = \{F\}$$

en un système triangulaire $\begin{bmatrix} & & S \\ & & \\ 0 & & \end{bmatrix}\{u_n\} = \{F'\}$.

- Résolution du système triangulaire supérieur précédent.

Cette étape consiste à calculer les inconnues $u_{n'}$ de la dernière à la première, par résolution du système triangulaire.

III.2.1. Triangularisation :

Pour résoudre un système de la forme suivante :

$$\begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1n} \\ k_{12} & k_{22} & \cdots & k_{2n} \\ . & . & \cdots & . \\ k_{n1} & & \cdots & k_{nn} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ . \\ u_n \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ . \\ F_n \end{Bmatrix}$$

La triangularisation consiste à « éliminer » successivement les inconnues u_s ,$s = 1,2,\ldots\ldots,n-1$ dans les équations $s+1$ à n. L'élimination de u_s se fait de la manière suivante :

- exprimer u_s en fonction de u_{s+1} , u_{s+2},……, u_n et F_s en utilisant l'équation s ;
- reporter l'expression de u_s précédente dans les équations $s+1$, $s+2$, ……,n.

Après élimination de u_s cette inconnue n'apparaît plus dans les équations $s+1$, ……, n ; il y a donc des zéros dans la colonne s sous la diagonale.

Après élimination des inconnues U_1 à U_{n-1} , la matrice $[k]$ est triangulaire supérieure, puisqu'elle ne comporte plus que des zéros sus la diagonale.

L'élimination de chaque inconnue u_s modifie $[k]$ et $\{F\}$. Notons $[k^s]$ et $\{F^s\}$ la matrice et le second membre après élimination des inconnues $1,2,3,\ldots,s$ la matrice $[k^0]$ étant la matrice initiale :

$[K]=[K^0]$ et $\{F^0\}=\{F\}$ système original

↓ éliminer u_1 dans les équations 2 à n

$[K^1]$ et $\{F^1\}$

↓ éliminer u_2 dans les équations 3 à n

$[K^2]$ et $\{F^2\}$

↓

.

.

↓ éliminer u_s dans les équations $s+1$ à n

$[K^s]$ et $\{F^s\}$

↓

.

.

↓ éliminer U_{n-1} dans les équations n

$[S]=[K^{n-1}]$ et $\{F^{n-1}\}=\{F'\}$ système triangulaire.

Pour éliminer, par exemple, la variable U_1 du système, nous utilisons la première équation sous la forme :

$$u_1 = \frac{1}{k_{11}}(F_1 - k_{12}u_2 \dots - k_{1n}u_n)$$

Reportons cette expression de u_1 dans les équations du système :

$$\begin{bmatrix} k_{11} & k_{12} & \dots & k_{1n} \\ 0 & k_{22} - \frac{k_{21}}{k_{11}}k_{12} & \dots & k_{2n} - \frac{k_{21}}{k_{11}}k_{1n} \\ . & . & & . \\ 0 & k_{n2} - \frac{k_{n1}}{k_{12}}k_{12} & \dots & k_{nn} - \frac{k_{n1}}{k_{11}}k_{1n} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ . \\ . \\ u_n \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 - \frac{k_{21}}{k_{11}}F_1 \\ . \\ F_n - \frac{k_{n1}}{k_{11}}F_1 \end{Bmatrix}$$

et ainsi de suite pour le reste des inconnues ; le système triangulaire final s'écrit :

$$\begin{bmatrix} k_{11} & \dots & \dots & \dots & \dots & \dots & k_{1n} \\ 0 & k'_{22} & \dots & \dots & \dots & \dots & k'_{2n} \\ . & 0 & k'_{33} & . & . & . & . \\ . & . & 0 & . & . & . & . \\ . & . & . & 0 & k'_{ss} & \dots & k'_{sn} \\ . & . & . & . & 0 & . & . \\ 0 & 0 & 0 & 0 & 0 & . & k'_{nn} \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ . \\ u_s \\ . \\ u_n \end{Bmatrix} = \begin{Bmatrix} F \\ F'_2 \\ . \\ F'_s \\ . \\ F'_n \end{Bmatrix}$$

où k'_s est le terme modifié de la matrice de rigidité.
et F'_s est le second membre de l'équation modifié.

III.2.2. Algorithme de triangularisation :

L'algorithme de triangularisation (de construction successive des matrices $[K^i]$ sera donc le suivant ;

$s = 1,2,\dots,n-1$
  $i = s+1, s+2, \dots\dots\dots, n$
    $c = \frac{k_{is}}{k_{ss}}$
    $F_i = F_i - cF_s$
      $j = s+1, s+2, \dots, n$
        $k_{ij} = k_{ij} - ck_{sj}$

III.2.3. Résolution du système triangulaire supérieur :

La résolution du système triangulaire se fait à partir de la dernière équation, en calculant successivement u_n , u_{n-1} , ... , u_1 :

$$u_n = k'_{nn} F'_n$$

$$u_{n-1} = k'_{n-1,n-1} (F'_{n-1} - k'_{n-1,n} u_n)$$

.....

$$u_1 = k'_{11} (F'_1 - k'_{12} u_2 - k_{13} u_3 - ..... - k'_{1n} u_n)$$

où les termes k'_{ij} et F'_i désignent les termes modifiées par la triangularisation, la résolution se fait don directement sur le système modifié.

Remarque :

L'algorithme donné ci-dessus doit être modifié pour tenir compte du cas ou l'élément k_{ii} (élément diagonal de la matrice de rigidité initiale non modifiée) est nul. En effet si cet élément, appelé *pivot*, l'algorithme de triangularisation ne fonctionne plus, il faudra alors permuter les lignes et les colonnes pour tenir compte de la symétrie de la matrice, s'il y 'a symétrie, et poursuivre l'opération.

IV.1. Mise en œuvre de l'optimisation des structures en treillis :

Le programme que nous développé calcule et optimise les structures spatiales et planes en treillis. Le calcul de la structure est basé la méthode des éléments finis et mis en œuvre à travers un programme de calcul qui fournit :

- les déplacements aux nœuds de la structure en barres
- les contraintes, de traction ou de compression, dans les barres de la structure.

Les éléments barres de la structure sont supposés linéaires et travailler uniquement à la traction et à la compression suivant l'axe de chacune d'elles. Aussi nous nous plaçons dans le domaine élastique des petites déformations, le comportement de la structure est don *linéaire* durant tout le processus d'optimisation.

L'optimisation de la structure est, quant à elle, basée sur la méthode SQP (Sequantiel quadratic programming) donnée dans le chapitre précédent. L'optimisation porte sur la minimisation de la fonction poids de la structure (fonction objectif). Les variables du problème ont été choisi comme étant les aires des sections droites des barres, leur nombre sera donc égal au nombre de barres de la structures.

Le problème d'optimisation est non linéaire (d'où le choix de la méthode SQP) mais aussi soumis à des contraintes d'optimisation. En effet les contraintes exercées sur les barres ne doit pas dépasser, en traction et en compression, des valeurs limites de résistance (que l'utilisateur doit introduire). Le nombre de contraintes d'optimisation est alors égale au double du nombre des barres constituants la structure à étudier. Notre problème est donc le suivant :

$$(P)\quad \begin{cases} \text{Minimiser } f(x)=\gamma \sum_{1}^{m} x_i l_i \\ \text{sous les contraintes :} \\ \sigma_{i\,limt} - \sigma_i(x) \geq 0 \quad i=1,\ldots\ldots\ldots\ldots,m \\ \sigma_i(x) + \sigma_{i\,limc} \geq 0 \quad i=1,\ldots\ldots\ldots\ldots,m \\ x \in S \subset R^m \end{cases}$$

où

x: vecteur, à m composantes, des sections des barres

$f(x)$: est la *fonction objectif,* poids de la structure.

m : est le nombre de barre

γ : le poids volumique du matériau constituant les barres

$\sigma_i(x)$ la contrainte dans la barre i, qui est une fonction non linéaire de x

$\sigma_{i\,limt}$: vecteur des contraintes limites de traction pour chaque barre, ses composantes peuvent être les mêmes.

$\sigma_{i\,limc}$: vecteur des contraintes limites de compression pour chaque barre, ses composantes peuvent être les mêmes.

S : ensemble des intervales où les différentes sections x_i peuvent être comprises.

Le code de calcul des structures intégré dans la procédure d'optimisation est complémentaire de celle-ci. En effet, le calcul des fonctions et de leurs dérivées se fait, à chaque itération, en appelant le sous-programme de calcul des structures spatiales en treillis STRUCT.

Les méthodes numérique, de dérivation et de résolution des systèmes d'équations, utilisées dans le programme sont celles données dans le chapitre précédant.

V. 1

Introduction

Le programme *OPTISTRUCT* que nous avons réalisé consiste à calculer les déplacements et les contraintes dans les éléments des structures spatiales en treillis et à optimiser la fonction poids de celles-ci. Nous avons intégré dans ce programme le code de calcul *MEF* conçu par *G.DHATT* et *G.TOUZOT* basé sur la ***méthode des éléments finis***, avec lequel la structure est calculée.

Nous donnerons dans ce chapitre l'organisation du programme ainsi que la manière dont les différents sous-programmes s'articulent, en mettant plus particulièrement l'accent sur le sous-programme d'optimisation. Nous aborderons aussi quelques techniques de programmations utilisées dans le développement du programme (allocation en mémoire, gestion des fichiers...), le tout enrichi d'organigrammes qui facilitent la compréhension du programme.

IV.3. Organisation générale du programme OPTISTRUCT :

Commençons, pour fixer les idées, par donner l'organigramme général du programme que nous avons développé. L'ensemble des sous-programmes sont appelés par le programme principal OPTISTRUCT selon un enchaînement qui permet, selon le besoin de l'utilisateur, de calculer la structure (déplacements aux nœuds et contraintes dans les barres) puis d'optimiser cette dernière. Nous mettrons, tout spécialement, l'accent sur le sous-programme d'optimisation .

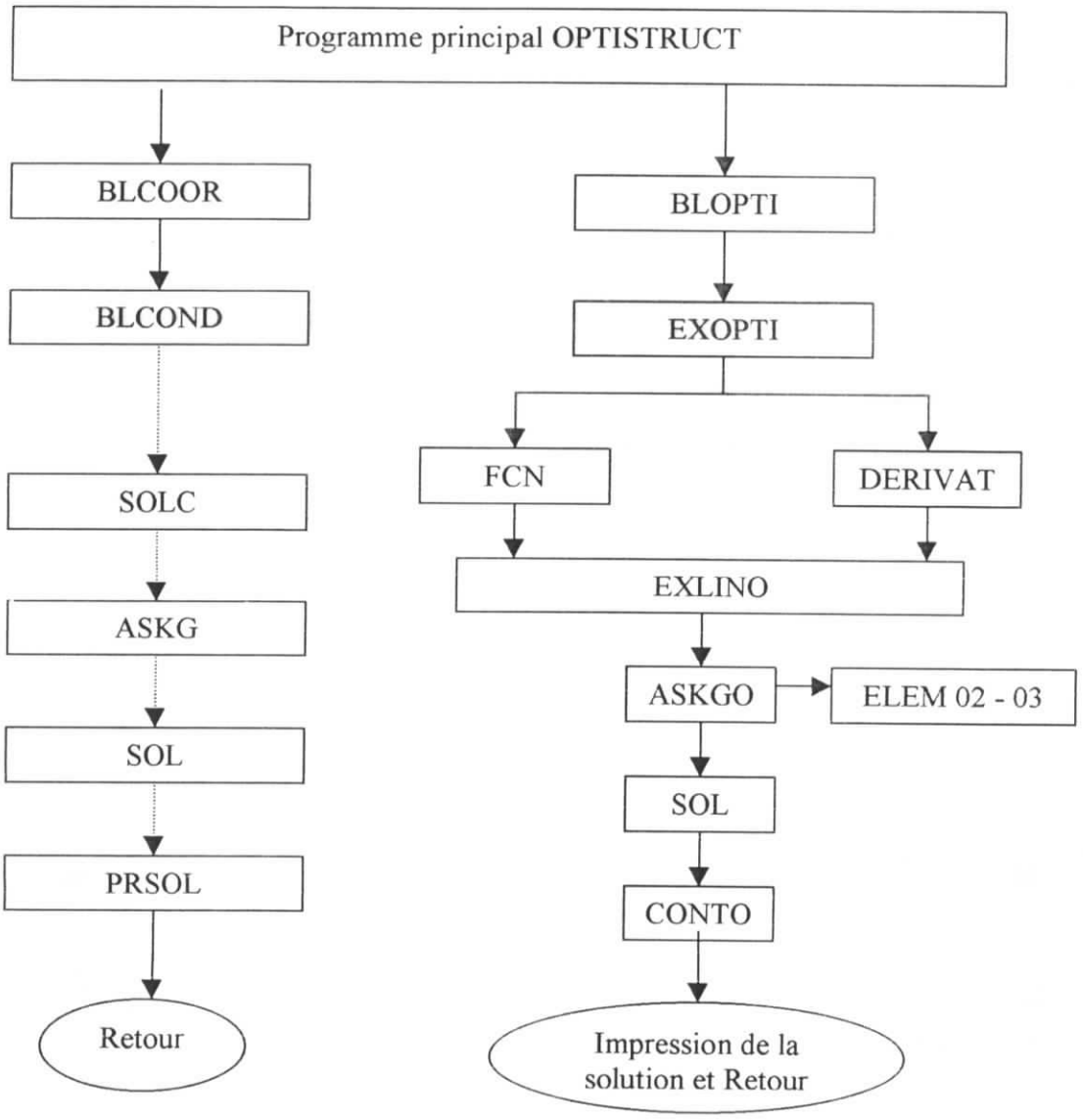


Figure IV.1 Organigramme d'appel de sous-programmes

On voit bien, à travers l'organigramme donné ci-dessus, que le programme comprend deux grandes parties, l'une, qu'on appellera STRUCT, est consacrée à l'étude de la structure et la seconde, appelés OPTI, à l'optimisation de cette dernière son optimisation. Nous donnons dans le paragraphe suivant les principaux blocs fonctionnels de la partie étude de la structure qui s'inspire du programme *MEF* [2].

IV.3.1. Organisation de la partie étude de la structure :

IV.3.1.1. Caractéristiques d'un programme d'éléments finis :

Tout programme basé sur la méthode des élément finis inclut les principaux blocs fonctionnels suivants :

a) lecture, vérification et organisation des données décrivant le maillage(nœuds et éléments), les paramètres physiques (modules d'élasticité), les sollicitation et conditions aux limites.
b) construction des matrices et vecteurs élémentaires, puis assemblage de ceux-ci pour former la matrice globale et le vecteur global des sollicitations.
c) Résolution du système d'équations après prise en compte des conditions aux limites.
d) Impression des résultats après calcul des variables additionnelles (contraintes, réactions...)

La figure ci-dessous montre l'enchaînement des blocs principaux.

Lecture, vérification, organisation des données
Lire et imprimer : • les coordonnées des nœuds • les connectivités des nœuds • les paramètres physiques • les sollicitations • les conditions aux limites

↓

Construction de la matrice et du vecteur globaux $[K]$ et $\{F\}$
Pour chaque élément: • extraire les information liées à cet élément • construire la matrice et le vecteur élémentaire $[K_e]$ et $\{F_e\}$ • assembler $[K_e]$ et $\{F_e\}$ dans $[K]$ et $\{F\}$

↓

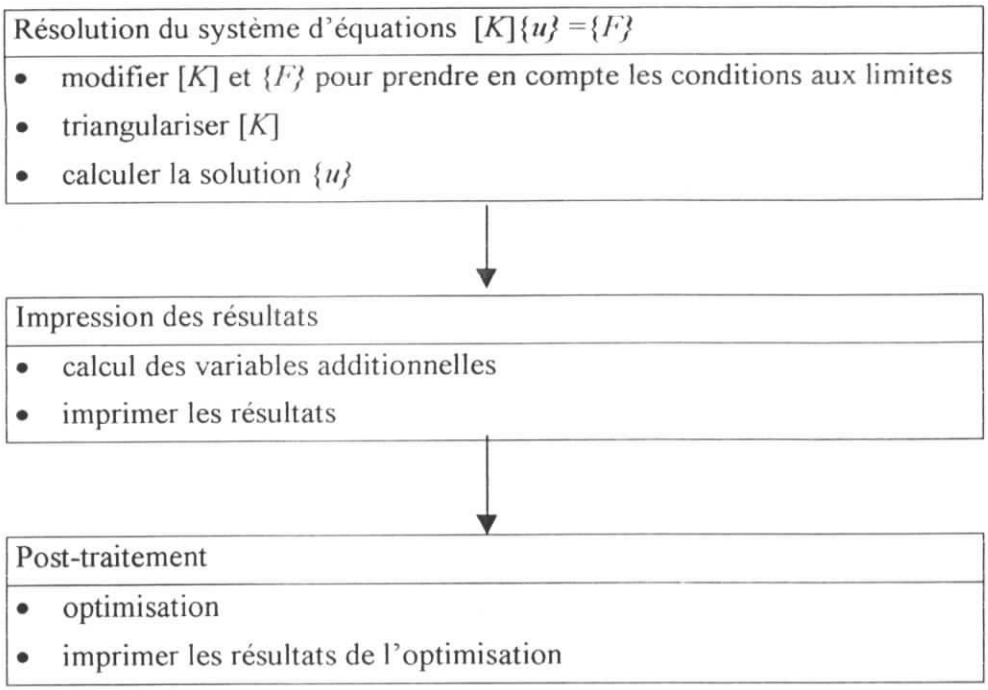


Figure IV.2 Processus de résolution du problème de calcul des treillis avec optimisation

IV.3.1.2. Modularité du programme :

Le sous-programme qui présente les caractéristiques décrites plus haut doit être tel que sa logique soit facile à comprendre et facilement modifiable (afin d'offrir la possibilité à ses utilisateurs de collaborer à son développement par la suite). C'est pour cela que la partie STRUCT du programme général est organisé en une série de sous-programmes fonctionnels modulables.

L'ensemble de ces *sous-programmes* sont appelés par le *programme principal* selon un enchaînement contrôlé par l'utilisateur à travers un *fichier de données* où sont introduits les données du problème, ainsi que l'ordre d'exécution des sous-programmes.

Chaque fonction de la procédure de résolution du problème de calcul de la structure en treillis est exécutée, comme le montre la figure ci-dessous, par un sous-programme BLnnnn [2]:

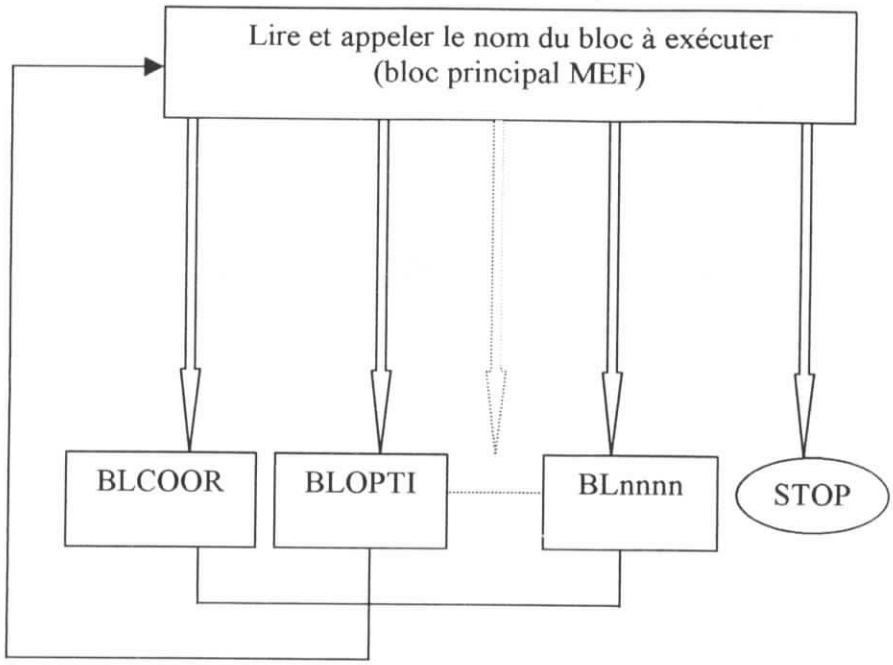


Figure IV.3 Organisation générale de l'appel des sous-programmes fonctionnels de la partie STRUCT

Avant d'aborder l'organisation du programme général attardons nous sur deux techniques de programmation utilisées dans le présent travail à savoir, le stockage (allocation) des données et l'utilisation des fichiers.

IV.3.2. Techniques de programmation :

IV.3.2.1. Allocation pseudo-dynamique et allocation dynamique :

Le programme MEF écrit en langage FORTRAN au début des années quatre vingt utilise une allocation dite *pseudo-dynamique*, que nous avons gardé dans l'écriture de la partie du programme général STRUCT. Ce type d'allocation était pratique quand on sait que les anciennes versions du langage FORTRAN ne permettait pas l'allouer dynamiquement des espaces dans la mémoire.

En effet pour éviter de changer les dimensions des tables (vecteurs et matrices) lorsque la nature et la taille du problème varient la technique permet [2] :

- de dimensionner les tables volumineuses comme des vecteurs et non pas comme des matrices.
- de placer toutes les tables réelles et entières dans une même table unique VA de dimension forfaitaire (que l'on peut modifier).
- de repérer chaque table par la position de son premier terme dans VA.

Le schéma ci-dessous montre comment les tables sont repérées dans la table VA ainsi que sa structure.

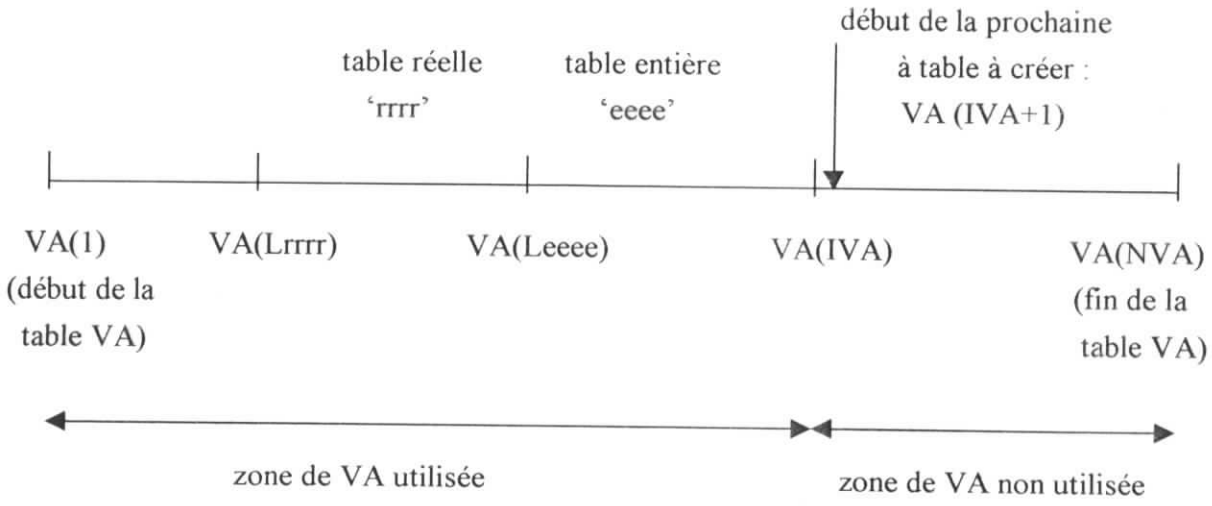


Figure IV.4 Allocation pseudo-dynamique

où Leeee et Lrrrr sont les pointeurs qui indiquent les débuts des tables créées. Remarquons que la création (réservation d'espace) et la suppression (décalage) des tables sont, respectivement, exécutées par les sous-programme ESPACE et VIDE.

Cette technique a été d'un grand secours pour les programmeurs avant que le FORTRAN ne se développe encore plus car elle permet d'économiser de l'espace sur la mémoire de la machine, néanmoins elle présente l'inconvénient d'encombrer le programme et peut être une source d'erreurs lors de l'exécution.

C'est pour cela que nous avons utilisé l'allocation *dynamique* permise par le langage FORTRAN 90. Cette allocation est représenté par une instruction Fortran dont voici la structure [16] :

déclarer le type de la table A à créer
lire la dimension N de la table
allouer l'espace N de la table A dans la mémoire.

L'allocation *dynamique*, nous le voyons bien, est plus fluide car l'économie d'espace mémoire est meilleure et plus intuitive (une table à deux dimensions, matrice, n'est pas, dans ce cas stockée comme un vecteur) et il devient plus simple de parcourir ses éléments. Cette technique a été utilisée dans la création des tables de la partie OPTI du programme général.

IV.3.2.2 Utilisation et gestion des fichiers :

L'utilisation, tout au long de notre programme, des *fichiers* permet [16] :
- de lire les données du problème à étudier selon un format spécifique.
- d'afficher les données et les résultats de l'étude, toujours selon un format précis.
- de stocker des données internes au programme et de les rendre accessibles.

Les deux premières fonctions sont nécessaires à notre programme pour introduire et lire les données et les résultats. La dernière fonction nous a permis de transférer les données internes d'un sous-programme à un autre de manière fluide (les arguments du sous-programme se retrouvants allégés) et interactive, en effet un sous-programme peut alors *lire* et *écrire* sur un même fichier. Cette technique rend donc le programme général plus accessible.

IV.3.3. Lecture des données du problème :

La lecture des données se fait sur un *fichier de données* où ces dernières sont entrées selon un format spécifique [2], et auxquelles nous avons rajouté les informations relatives à l'optimisation.

Les données générales suivantes sont nécessaires pour l'exécution de la partie calcul de la structure et optimisation :

a) les données relatives à la géométrie de la structure :
   - le nombre de nœuds total (*NNT*)
   - le numéro et les coordonnées (planes ou spatiales) de chaque nœud
   - le nombre d'éléments total (*NELT*)
   - le numéro des deux nœuds définissant chaque élément.

b) les données relatives aux propriétés élémentaires :
   - le module d'élasticité du matériau (E).
   - les sections des barres.
   - Les contraintes limites de chaque barre à la compression et à la traction (données exploitées par la partie optimisation, dans le cas où celle-ci est demandée)

c) les données relatives aux charges concentrées appliquées :
   - les directions de chaque charge.
   - les composantes des charges suivant les axes du repère global

- les nœuds d'application.

d) les données nécessaires à l'optimisation de la structure.

- le nombre maximum d'itération (*MAXITN*)
- le nombre de contraintes (*NC*)
- le nombre de variable du problème d'optimisation (*N=NELT*).
- les limites inférieures et supérieures des sections ou la plage des sections (*XLB* et *XUB*).

IV.3.4. Description du bloc optimisation :

Nous allons dans ce qui suit décrire le programme d'optimisation ainsi que les sous-programmes qui s'y rattachent. Mais tout d'abord commençons par donner, dans le tableau suivant, les variables principales de la partie optimisation OPTI;

Nom de la variable	Dimension	Description
Vecteurs du sous-programme BLopti		
X	*N*	Vecteur contenant les sections courantes à une itération k , de dimension *N* (nombre d'éléments)
XGUESS	*N =NELT*	Vecteur contenant les sections courantes à l'itération $k=0$
XLB	*N*	Vecteur qui porte les limites inférieures des sections
XUB	*N*	Vecteur qui porte les limites supérieures des sections
VCLIMT	*NELT*	Vecteur contenant les limites des contraintes de traction dans chaque barre
VCLIMC	*NELT*	Vecteur contenant les limites des contraintes de compression dans chaque barre
Constantes du sous-programme BLopti		
MAXITN	*	Nombre maximum d'itération
NC	*	Nombre de contraintes d'optimisation (= *NELT*)
N	*	Nombre de variables d'optimisation (= *NELT*)
Vecteurs et scalaires du sous-programme FCN		

G	*NELT*	Vecteur contenant les contraintes d'optimisation, extrait du vecteur *VCONT* (contraintes dans les barres)
LONG	*NELT*	Vecteur des longueurs des barres
VCONT	*NELT*	Vecteur contenant les contraintes dans les barres pour les sections correspondants à l'itération *k*
VSEC	*NELT*	Vecteur des sections courantes (où *X*)
F	*	Valeur de la fonction objectif à l'itération *k*
Vecteurs et scalaires du sous-programme DERIVAT		
JACOB	$NC \times N$	Matrice contenant les éléments du jacobien des fonction G_i
VSECPi $(i = \overline{1,4})$	*NELT*	Vecteur contenant les valeurs des sections perturbées
VCONTPi $(i = \overline{1,4})$	*NELT*	Vecteur contenant les valeurs des contraintes perturbées qui correspondent aux section perturbées
DG	$NC \times N$	Matrice contenant les gradients des contraintes d'optimisation *activées* (G_i), elle est extraite de *JACOB*
DF	*	Gradient de la fonction objectif *F*
NOR	*	Norme du vecteur des sections courantes *X*
PAS	*	Pas de la dérivation $PAS = NOR \times 10^{-7}$

Le programme d'optimisation fait appel à un ensemble de sous-programmes donnés dans ce qui suit :

(a) **BLOPTI**

Ce sous-programme est appelé par le programme principal dans le cas où l'optimisation de la structure est demandée par l'utilisateur. Il a pour fonction de lire sur le fichier des données les entrées relatives à l'optimisation et de créer dynamiquement les vecteurs associées. Ce sous-programme appelle ensuite l'exécutable EXOPTI et imprime enfin le poids optimal de la structure ainsi que les sections des barres dans le fichier résultat.

(b) **EXOPTI**

Il s'agit du sous-programme ou l'algorithme d'optimisation est mis en ouvre et qui se charge d'appeler les sous-programmes FCN et DERIVAT. Ces deux sous-programmes, nous allons le voir, sont nécessaire car la méthode d'optimisation SQP repose grandement sur eux. Parmi les arguments d'appel se trouve le vecteur *X* courant à l'itération *k* et ou les

informations complémentaires (valeur de la fonction objectif, jacobien....) doivent être évaluées.

Nous donnons dans ce qui suit l'organigramme général de ce sous-programme :

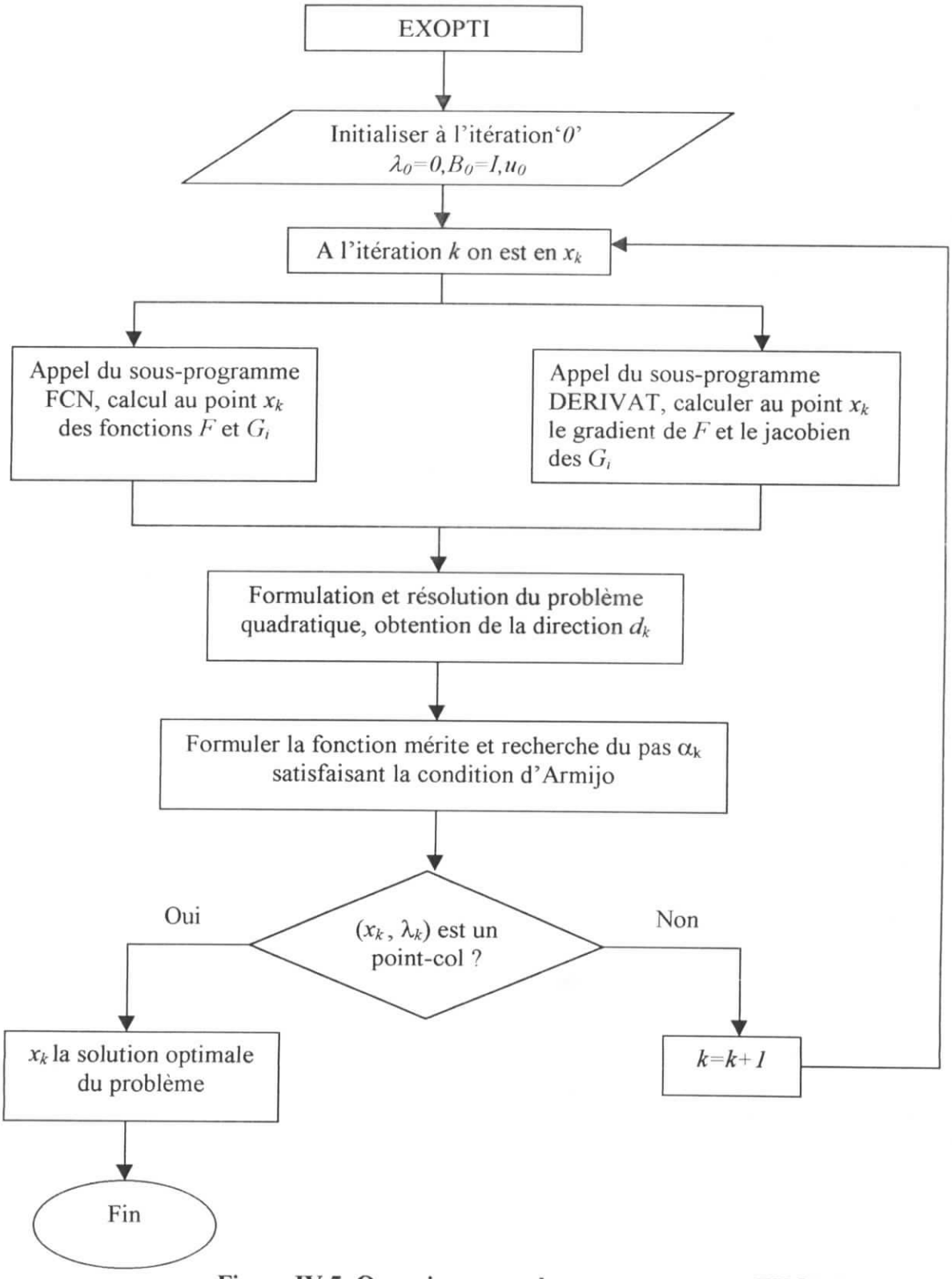


Figure IV.5 Organigramme du sous-programme EXOPTI

(c) **FCN**

Ce sous-programme calcul la valeur de la fonction objectif au point courant (X^*) ainsi que les valeurs des contraintes d'optimisation G_i activées. La fonction objectif est évaluée linéairement à partir du point courant X^*, des longueurs des barres et du poids spécifique du matériau constituant la structure.

Les valeurs des contraintes d'optimisation G_i ne sont pas calculées directement car elles dépendent des contraintes limites (de traction et de compression) mais aussi des contraintes dans les barres à l'itération k , il est donc nécessaire de faire appel à sous-programme de calcul des contraintes, ce sous-programme est nommé EXLINO (EXécution d'une étude LINéaire pour l'Optimisation).

Nous donnons dans ce qui suit l'organigramme général de ce sous-programme :

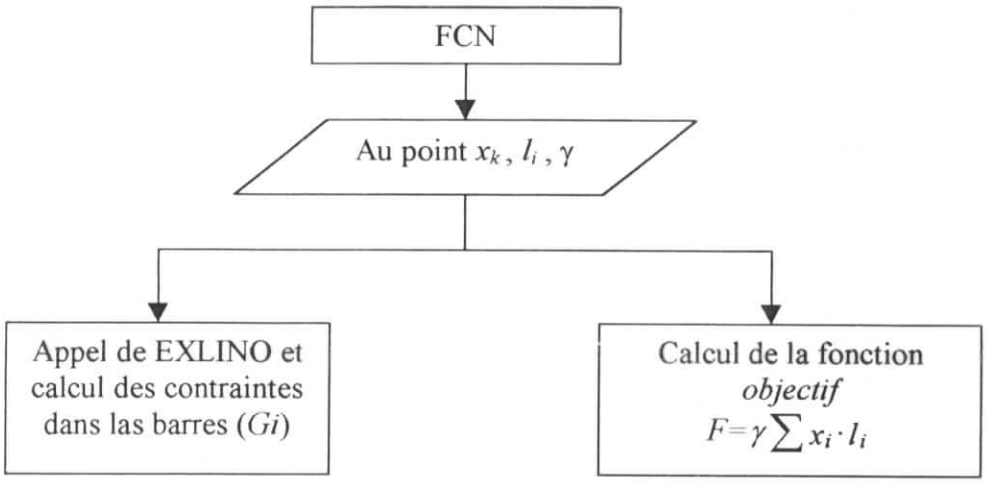


Figure IV.6 Organigramme du sous-programme FCN

(a) EXLINO

Ce sous-programme s'inspire du programme MEF mais n'exécute que les parties assemblage des matrices et vecteurs ; résolution du système d'équations et calcul des contraintes dans les barres, le reste des informations (géométrie de la structure, connectivité, sollicitations...) étants déjà lues et stockées en mémoire lors de l'exécution du programme MEF pour les sections de départ.

La structure de EXLINO est donc la suivante :

(1) lecture des donnés fixes du problème ainsi que les nouvelles sections des barres pour lesquelles le programme va calculer les contraintes, distinction entre le calcul effectué pour le sous-programme FCN et DERIVAT.

(2) appel du sous-programme (ASKGO) d'assemblage de la matrice de rigidité et du vecteur force associé. Ce sous-programme, comme celui du programme MEF, distingue entre les problèmes plans et tridimensionnels ce qui permet, en deux dimensions, de gagner en temps d'exécution et en mémoire.

(3) appel du sous-programme (SOL) de résolution du système d'équations $[K]\{U\}=\{F\}$ ainsi obtenu, les déplacements étants calculés il est possible de calculer les contraintes dans les barres.

(4) calcul des contraintes dans les barres à l'aide du sous-programme CONTO, pour chaque élément, ce sous-programme extrait les déplacements aux nœuds, calcul les forces axiales qui s'exercent sur la barre à l'aide de la matrice de rigidité élémentaire et calcul enfin la contrainte (de compression où de traction) désirée.

(5) retour au programme appelant.

(e) DERIVAT

Ce sous-programme calcule le gradient de la fonction objectif et le jacobien des contraintes d'optimisation. Il attribut ensuite les éléments du jacobien suivant que la contrainte considérée est activée ou non.

Le calcul du gradient de la fonction objectif par rapport aux valeurs des sections courantes, nous l'avons vu, ne pose pas de problème (la fonction poids étant linéaire). Mais en ce qui concerne le jacobien il est nécessaire de faire appel au sous-programme EXLINO pour calculer les contraintes perturbées qui correspondent aux sections des barres perturbées.

L'exécution de EXLINO est la même que précédemment avec la seule différence que les fichiers utilisés pour la dérivation sont spécifiques à ce calcul.

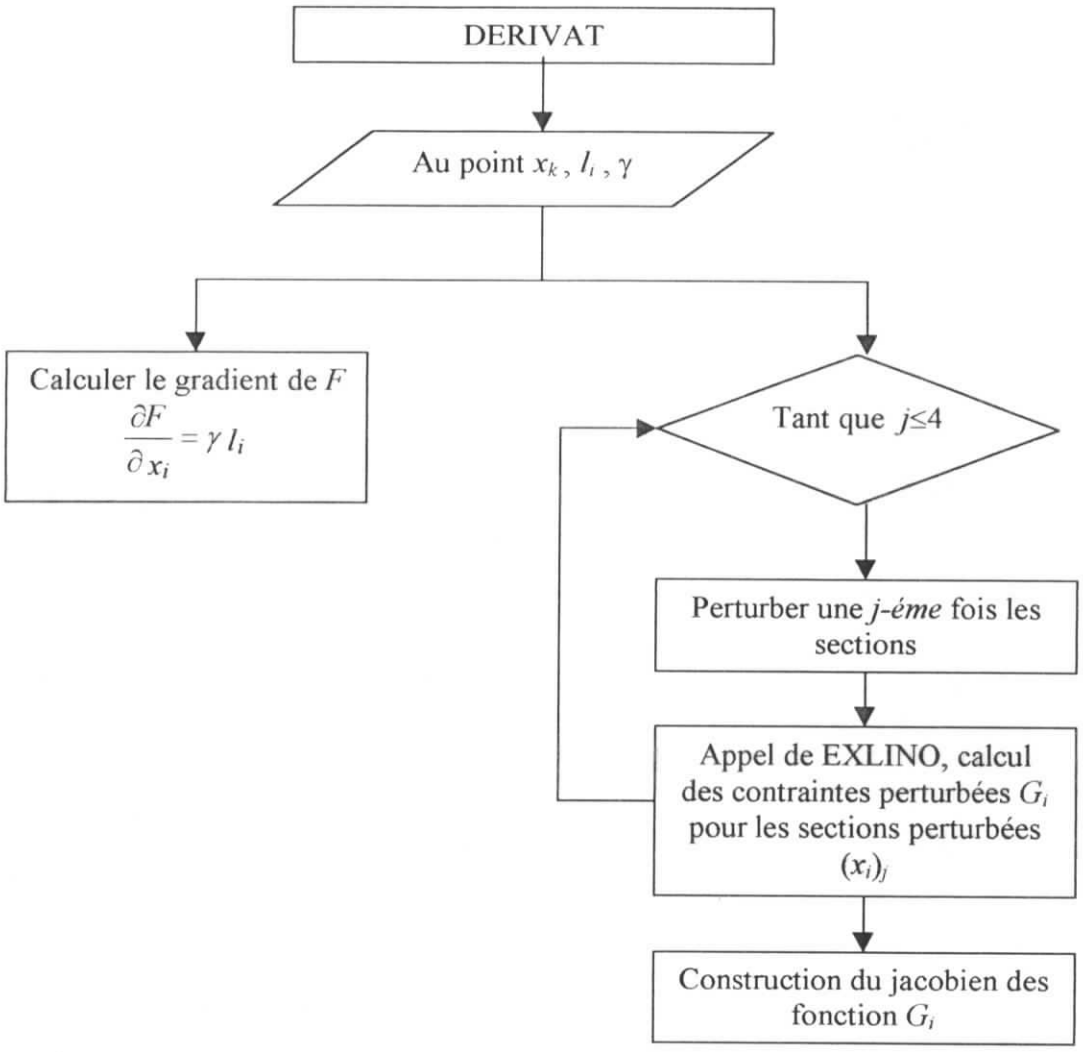


Figure IV.7 Organigramme du sous-programme DERIVAT

(c) **FCN**

Ce sous-programme calcul la valeur de la fonction objectif au point courant (X^*) ainsi que les valeurs des contraintes d'optimisation G_i activées. La fonction objectif est évaluée linéairement à partir du point courant X^*, des longueurs des barres et du poids spécifique du matériau constituant la structure.

Les valeurs des contraintes d'optimisation G_i ne sont pas calculées directement car elles dépendent des contraintes limites (de traction et de compression) mais aussi des contraintes dans les barres à l'itération k , il est donc nécessaire de faire appel à sous-programme de calcul des contraintes, ce sous-programme est nommé EXLINO (EXécution d'une étude LINéaire pour l'Optimisation).

Nous donnons dans ce qui suit l'organigramme général de ce sous-programme :

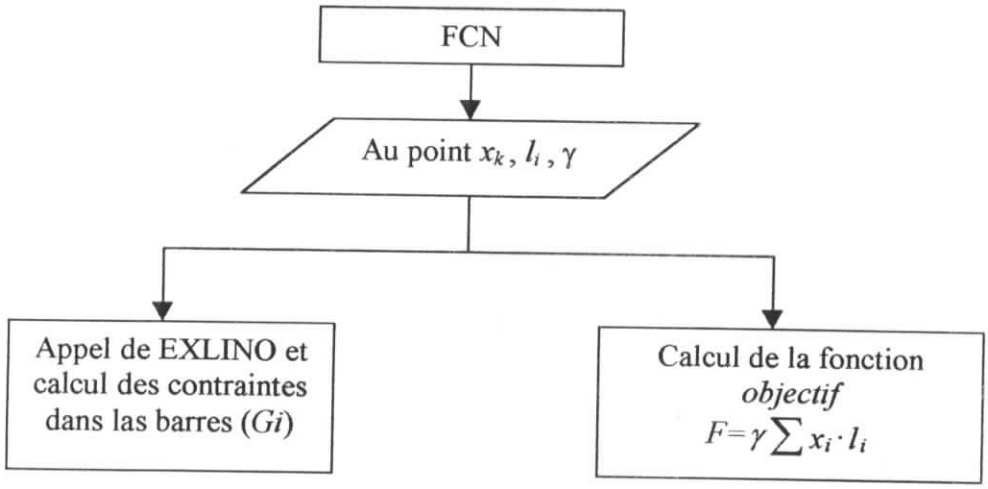


Figure IV.6 Organigramme du sous-programme FCN

(a) EXLINO

Ce sous-programme s'inspire du programme MEF mais n'exécute que les parties assemblage des matrices et vecteurs ; résolution du système d'équations et calcul des contraintes dans les barres, le reste des informations (géométrie de la structure, connectivité, sollicitations...) étants déjà lues et stockées en mémoire lors de l'exécution du programme MEF pour les sections de départ.

La structure de EXLINO est donc la suivante :

(1) lecture des donnés fixes du problème ainsi que les nouvelles sections des barres pour lesquelles le programme va calculer les contraintes, distinction entre le calcul effectué pour le sous-programme FCN et DERIVAT.

(2) appel du sous-programme (ASKGO) d'assemblage de la matrice de rigidité et du vecteur force associé. Ce sous-programme, comme celui du programme MEF, distingue entre les problèmes plans et tridimensionnels ce qui permet, en deux dimensions, de gagner en temps d'exécution et en mémoire.

(3) appel du sous-programme (SOL) de résolution du système d'équations $[K]\{U\}=\{F\}$ ainsi obtenu, les déplacements étants calculés il est possible de calculer les contraintes dans les barres.

(4) calcul des contraintes dans les barres à l'aide du sous-programme CONTO, pour chaque élément, ce sous-programme extrait les déplacements aux nœuds, calcul les forces axiales qui s'exercent sur la barre à l'aide de la matrice de rigidité élémentaire et calcul enfin la contrainte (de compression où de traction) désirée.

(5) retour au programme appelant.

(e) DERIVAT

Ce sous-programme calcule le gradient de la fonction objectif et le jacobien des contraintes d'optimisation. Il attribut ensuite les éléments du jacobien suivant que la contrainte considérée est activée ou non.

Le calcul du gradient de la fonction objectif par rapport aux valeurs des sections courantes, nous l'avons vu, ne pose pas de problème (la fonction poids étant linéaire). Mais en ce qui concerne le jacobien il est nécessaire de faire appel au sous-programme EXLINO pour calculer les contraintes perturbées qui correspondent aux sections des barres perturbées.

L'exécution de EXLINO est la même que précédemment avec la seule différence que les fichiers utilisés pour la dérivation sont spécifiques à ce calcul.

V.I. Exemple à six barres

Cet exemple d'application est une structure en treillis constitué de six barres toutes comprises dans le même plan, voir figure V.I.1

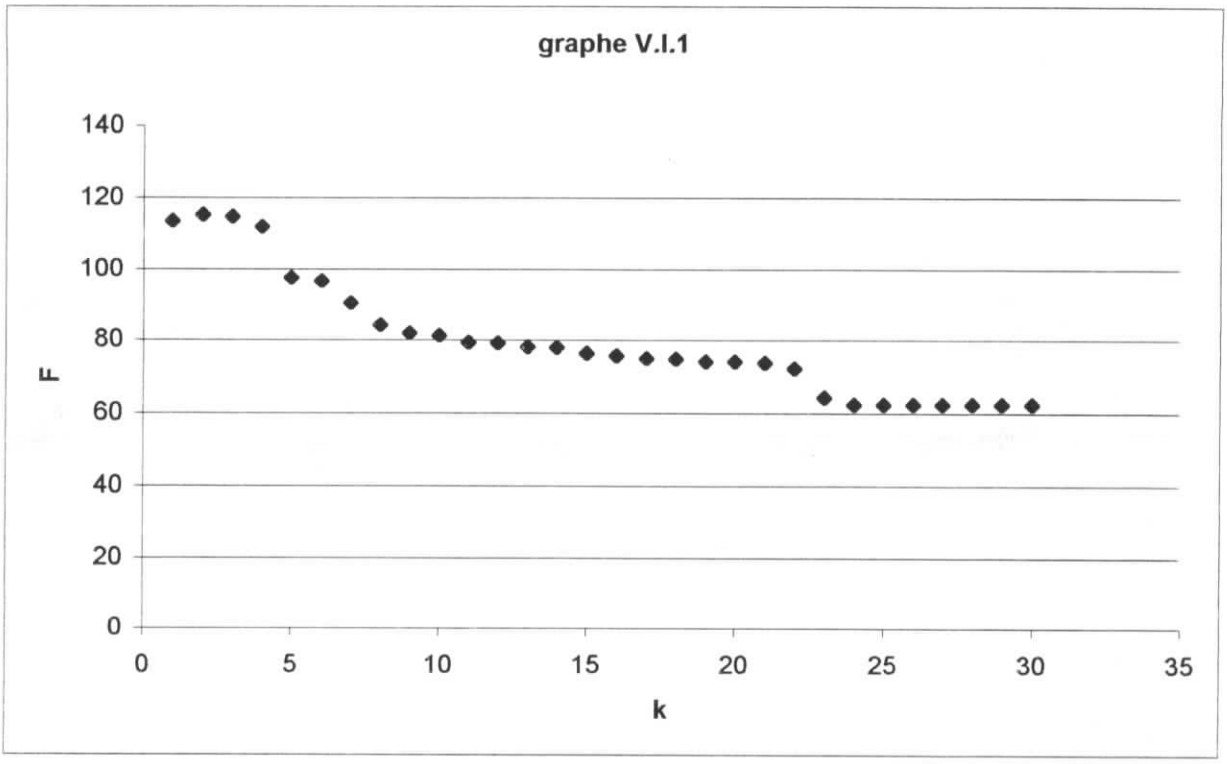
La force est appliquée à la structure au nœud libre (2) suivant la direction de y négatifs.
Le résultat du calcul est donné dans le tableau V.I.1., pour chaque itération(Les sections des barres sont exprimées en mm^2, et le poids à chaque itération en kilogramme).

Tableau V.I.1.

k	A_1	A_2	A_3	A_4	A_5	A_6	P
1	50.00000	50.00000	50.00000	50.00000	50.00000	50.00000 *	113.56881
2	48.50380	49.94058	50.52261	49.94058	50.96461	53.76047*	115.40361
3	47.83813	49.71497	50.24599	49.71497	50.70795	53.99962	114.89521
4	45.13309	48.66784	48.80124	48.66784	49.24276	54.12954	112.02663
5	31.68916	43.43877	41.55335	43.43877	41.86897	54.54892	97.57719
6	30.81523	43.07616	41.22903	43.07616	41.31142	54.64750	96.68474
7	26.63992	40.25058	41.50891	40.25058	35.45441	53.75800*	90.47085
8	21.78522	37.17437	42.21359	37.17437	29.30189	53.72762*	84.30137
9	20.04147	36.06258	42.44259	36.06258	27.14343	53.65296	82.06615
10	19.55463	35.49908	42.27628	35.49908	26.63817	53.61544	81.26363
11	18.42570*	34.05879	41.76662	34.05879	25.58242	53.48391	79.32125
12	18.49770*	33.81791	41.49348	33.81791	25.87030	53.39938*	79.21963
13	18.40467*	32.29838	40.31509*	32.29838	26.63280	53.12976*	78.19832
14	18.48910*	31.74492	40.37884	31.74492	26.52408	53.37349*	78.05906
15	18.79922*	27.76744	39.60402	27.76744	26.60380	54.26875*	76.42468
16	18.95400*	25.91484	39.21630	25.91484	26.71495	54.71556*	75.70643
17	19.10751*	24.21502	38.83168	24.21502	26.84791	55.15871*	75.07097
25	34.63490*	0.01000	0.01483	0.01000	0.01000	99.98248*	62.40455
26	34.63491*	0.01000	0.01482	0.01000	0.01000	99.98250*	62.40455
27	34.63491*	0.01000	0.01480	0.01000	0.01000	99.98251*	62.40456
28	34.63494**	0.01000	0.01472*	0.01000	0.01000	99.98260**	62.40458
30	34.63494**	0.01000	0.01472**	0.01000	0.01000	99.98260**	**62.40458**

(*) les contraintes d'optimisation sont violées c'est à dire que la contrainte dans la barre est supérieure aux contraintes limites.
(**)les contraintes d'optimisation sont saturées c'est a dire que la contrainte dans la barre est aux voisinage de la contraintes limite .

La graphe V.I.1 montre l'évolution de la valeur de la fonction objectif en fonction des itérations.



Interprétations du résultat :

(a) La courbe est, comme le montre la figure, cascadée mais a une allure générale de descente.L'augmentation en poids de la structure entre l'itération 1 et 2 est dû au fait que la contrainte de départ de la barre (6) dépasse la valeur limite de traction. L'algorithme a rattrapé l'écart au bout d'une itération pour éffectuer la recherche dans le domaine des solutions réalisables, ce point sera abordé plus en détail dans la prochaine application.

(b) L'étude du tableau des résultats, tableau V.I.1, montre une diminution constante des sections de barres 2,3,4, et 5 jusqu'à atteindre la valeur minimale($0{,}01mm^2$) de l'intervalle ou les sections optimales doivent être comprises.

Ceci s'explique par le fait que les barres citées plus haut et dans ce cas de chargement, ne travaillent pas. Leur rôle dans la structure, dans ces conditions de sollicitations, est négligeable. L'ensemble de la structure peut se passer de ces barres sans aucunement affecter l'équilibre ni la résistance de la structure.

Ceci est un aspect important du programme et du travail que nous avons effectué, en effet la convergence de la méthode d'optimisation garantie une solution optimale unique (minimisation du poids de la structure) mais peut aussi nous guider dans le dimensionnement des structures d'une part et dans la conception géométrique de la structure.

V.II. Exemple quatre barres dans l'espace :

Le système d'unité utilisé dans cet exemple est le système(lb, in^2) tels que :

1 $in^2 = 6.452\ cm^2$

1 lb = 4.45 N

1kip = 1000 lb = 4.45 KN
1 ksi = 6.89 MPA

V.II.1. 1^{ere} étude :

L'optimisation de ce treillis, en minimisant son poids, à été effectuée par *A.J MORIS* [18], en utilisant la méthode des forces pour le calcul des contraintes dans les barres et une analyse non linéaire. L'optimisation de la fonction poids est effectuée après linéaireisation des contraintes d'optimisation par l'algorithme du simplexe.

Le problème d'optimisation se résume ainsi :

$$\text{Minimiser} \quad f(A) = \gamma \sum_{i=1}^{n} L_i A_i$$

Sous les contraintes d'optimisations :

$$\sigma_c \leq \sigma \leq \sigma_u$$

$$-u_l \leq u \leq u_l$$

σ_c : contrainte limite de compression
σ_u : contrainte limite de rupture.
u_l : déplacement limite.

Cette limitation des déplacements permet un calcul non linéaire c'est à dire l'exploitation de la non linéarité géométrique du matériau telle que la borne supérieure des contraintes d'optimisation est la contrainte limite de rupture.

Caractéristiques de ce treillis :
treillis hyperstatiquement appuyé (voir figureV.1)
σ_u = 25.00 ksi = 172,25 MPA
σ_c = -25.00 ksi = -172,25 MPA
E =10000 ksi = 68900
γ = 0.1 lb/in^2 =2635 kg/m^3
les sections de départ sont $A_0 = \langle 2.50 , 2.50 , 2.50 , 2.50 \rangle$
on limite les déplacements au nœud '5' à $u_l = \mp 0.3$ in
les forces extérieures sont appliquées au nœud '5' telles que :
F_x = 10 kips = 44.5 KN
F_y = 20 kips = 89 KN
F_z = -60 kips = - 267 KN

Les résultats obtenus de *A.J MORIS* [18], après 12 itérations sont donnes dans le tableau V.II.1,nous représentons la variation du poids de ce treillis en fonction des itérations (voir graphe V.II.1). Les contraintes d'optimisation saturées à l'optimum sont, la contrainte de compression dans la barre 3, le déplacement selon z du nœud '5' ainsi que la section minimale dans la barre 1.

Tableau V.II.1

Itération K	Section N°1 (in^2)	Section N°2 (in^2)	Section N°3 (in^2)	Section N°4 (in^2)	Poids (lb)
1	2.50	2.50	2.50	2.50	174.296
2	1.5770	3.3215	2.2070	1.4958	143.449
3	1.4959	3.8950	2.0050	1.3558	143.071
4	1.2671	3.5323	2.2066	1.5674	142.666
5	1.3794	3.5957	1.7941	1.8238	143.038
6	1.0380	3.8972	1.7063	1.9037	141.488
7	0.8831	3.9812	1.4699	2.1829	140.680
8	0.5800	4.0968	1.3888	2.2639	138.243
9	0.5377	3.9960	1.0828	2.5152	135.419
10	0.3399	4.2669	0.8092	2.5152	130.989
11	0.1000	3.4942	0.7316	2.8148	121.521
12	0.1000	4.0108	0.7409	2.4314	120.544

Cette étude nous permettra de faire une étude comparative avec l'étude que nous allons faire, en optimisant le poids de la même structure par l'algorithme de SQP.

V.II.2. 2eme étude :

Le problème d'optimisation, comme nous allons le traiter, est le suivant :

Minimiser $f(A) = \gamma \sum_{i=1}^{n} L_i A_i$

Sous les contraintes d'optimisation :

$$\sigma_c \leq \sigma \leq \sigma_u$$

σ_c : contrainte limite de compression ou de non-flambement (contrainte critique d'Euler)
σ_t : contrainte limite élastique à la traction

Le programme que nous avons élaboré n'effectue qu'un calcul linéaire, sous l'hypothèse de petits déplacements. Par conséquent la limitation sur les déplacements n'est pas obligatoire.

Pour le même treillis étudier par *A.J MORIS* [18], les contraintes limites élastiques du matériau de ce treillis sont :
σ_c = -95 MPA = -13,78 ksi
σ_t = -95 MPA = 1378 ksi
Structure composée de 4 barres dans l 'espace (figure1)

V.II.2.1. Cas 1

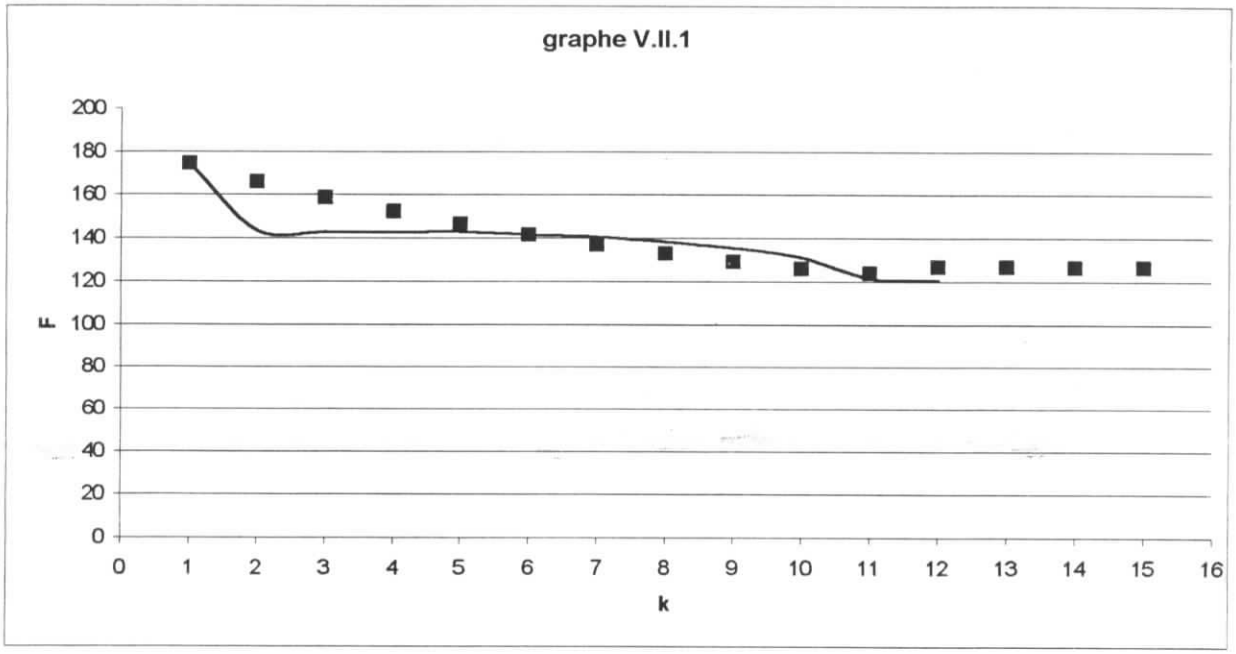
Pour un point de départ des sections incluses dans le domaine des sections réalisables
A(2.5 , 2.5 , 2.5 , 2.5)
XUB=0.1
XLB=10.00

Nous obtenons les sections et le poids du treillis à chaque itérations nous représentons ces résultats dans le tableauV.II.2 . Ainsi nous représentons la variation du poids de ce treillis en fonction des itérations (voir graphe V.II.1).

A : section (in^2)
F : poids total de la structure (lb)
K : numéro de l'itération

Tableau V.II.2

Itération k	Section N° 1 (in^2)	Section N° 2 (in^2)	SectionN°3 (in^2)	Section N° 4 (in^2)	F (Lb)
1	2.5	2.5 *	2.5	2.5	**174.29549**
2	2.260	2.56144 *	2.49822	2.26000	**166.07264**
3	2.044	2.61729 *	2.50142	2.04400	**158.76959**
4	1.8496	2.66806 *	2.50813	1.84960	**152.27539**
5	1.67464	2.71420 *	2.51723	1.67464	**146.49401**
6	1.51718	2.75612 *	2.52787	1.51718	**141.34203**
7	1.37546	2.7942 *	2.53941	1.37546	**136.74674**
8	1.24791	2.82876 *	2.55138	1.24791	**132.64459**
9	1.13312	2.86013 *	2.56342	1.13312	**128.97991**
10	1.02981	2.88859 *	1.13312	1.02981	**125.70381**
11	0.95185	2.92313 *	2.57274	0.97252	**123.62695**
12	0.90197	3.29074 *	2.47261 *	1.01315	**126.71499**
13	0.93355	3.27524 *	2.52504 *	0.94209	**126.51414**
14	0.91944 *	3.29914	2.52478 *	0.93534 *	**126.45499**
15	0.91957**	3.29934**	2.52472**	0.93547**	**126.46154**



——— variation du poids F en fonction des itérations k (étude 1,)

■ variation du poids F en fonction des itérations k (étude 2, cas1)

Les résultats de notre étude sont donnés dans le tableauV.II.1. L'évolution de la convergence est donné sur la graphe V.II.1.

Le nombre d'itération est égal à 14, et la valeur de l'optimum atteint est légèrement supérieure à celle de l'exemple de référence, ceci est du au fait que :

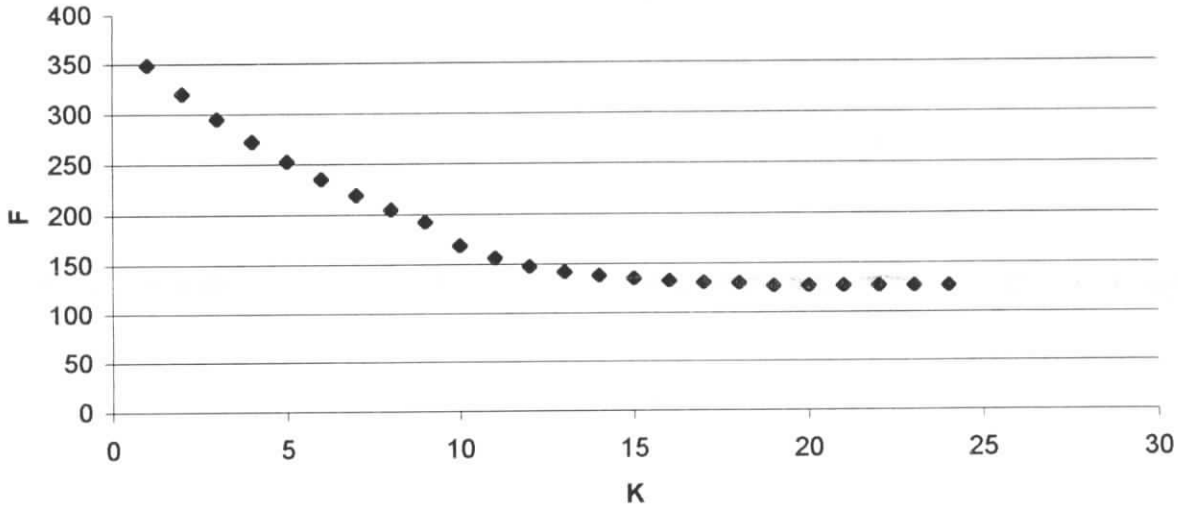
- Le comportement global des deux structures est différent (linéarité de l'une et non-linéarité de l'autre)
- Les contraintes sur les déplacements de l'exemple de référence limites le champs des solutions admissibles.

V.II.2.2. Cas 2

Pour des sections de départ non réalisables c'est à dire qu'elles ne vérifient pas les conditions limites de contraintes. Les sections doivent être incluses dans le domaine $\Omega=[0.1\ ,10.00]$
En posant $A_0=[0.01, 0.01, 0.01, 0.01]$

Le tableau (tableau V.II.2, annexe) représente les sections à chaque itération, nous construisons ainsi le graphe de l'évolution du poids de la structure en fonction des itérations k GrapheV.II.2

graphe V.II.2



On voit bien que l'optimum du poids, atteint dans ce cas, est égal à **126.46154**. Cette valeur est la même que celle du cas précédent, pour un nombre d'itération égal à 24 .

V.II.2.3. Cas 3 :

Pour des sections de départ réalisables (les contraintes de départ dans les barres vérifient les conditions de résistance), incluses dans le domaine $\Omega=[0.1, 10.00]$

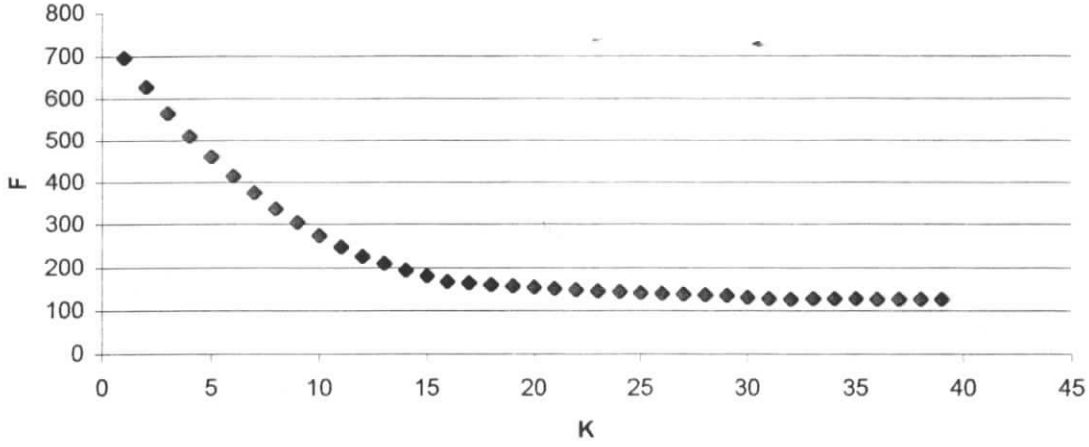
En posant $A=[15.0, 15.0, 15.0, 15.0]$

Le tableau (tableau V.II.3, annexe) représente les sections à chaque itération, nous construisons ainsi le graphe de l'évolution du poids de la structure en fonction des itérations k grapheV.II.3

Nous obtenons, dans ce cas, 39 itérations. Les résultas sont donnés dans le tableau V.II.3 pour les 5 premières et dernières itérations

Là aussi l'optimum du poids, atteint dans ce cas, est égal à **126.46154**. Cette valeur est la même que celle du cas précédent.

graphe V.II.3



V.II.2.4. Cas 4 :

Pour des sections de départ réalisables (les contraintes de départ dans les barres vérifient les conditions de résistance), incluses dans le domaine $\Omega=[0.5 \quad 5.00]$

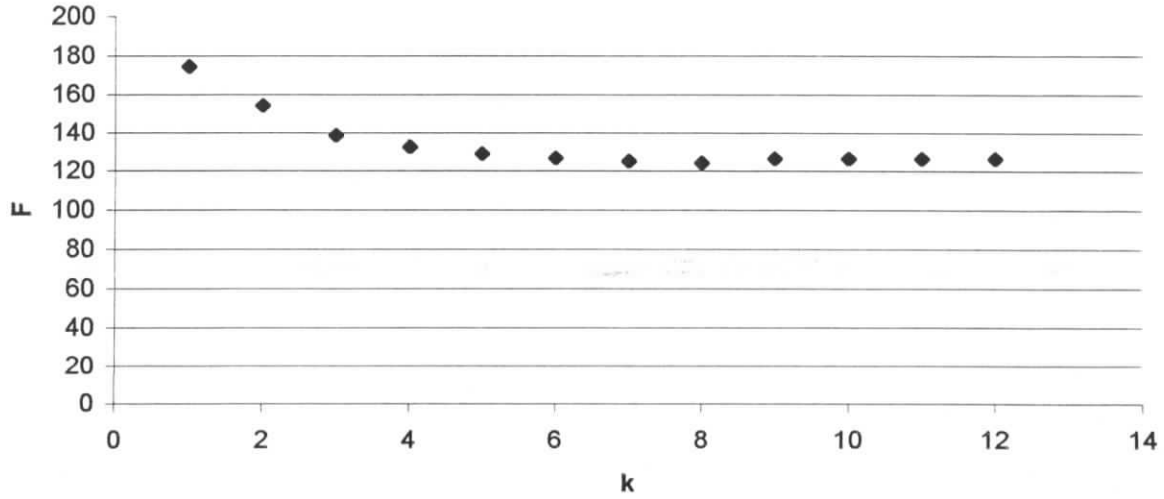
En posant $A=[2.5 , 2.5 , 2.5 , 2.5]$

Le tableau (tableau V.II.4, annexe) représente les sections à chaque itération, nous construisons ainsi le graphe de l'évolution du poids de la structure en fonction des itérations k grapheV.II.4.

Nous obtenons, dans ce cas, 12 itérations. Les résultas sont donnés dans le tableau V.II.4.

Là aussi l'optimum du poids, atteint dans ce cas, est égal à **126.46154**. Cette valeur est la même que celle du cas précédent.

graphe V.II.4



Interprétations des résultats :

Les résultats obtenus pour cette structure spatiale à quatre barres sont tous identiques. En effet les poids optimaux sont tous égaux à la valeur **126.46154lb** . Néanmoins trois remarques s'imposent :

a) Le nombre d'itération vari d'un cas à un autre car ce paramètre dépend du point de départ de l'algorithme d'optimisation. Le nombre d'itération des cas (2) et (3) est élevé en raison de l'éloignement du point de départ de la solution optimale.
Le cas (3) est intéressant, car il montre qu'un point de départ réalisable ne mène pas obligatoirement vers un nombre d'itération et un temps de calcul réduits (k= *39*). Le nombre d'itération minimal est obtenu pour le cas (4) où l'intervalle des solutions (introduit par l'utilisateur du programme) a été *réduit* ou *resserré* autours du vecteur solution, de maniére à orienter l'algorithme plus rapidement vers la solution.
Ceci nous méne à la conclusion suivante :
Il est nécessaire de resserrer l'intervalle des solutions autours de la solution afin de réduire le temps de calcul de la solution du problème d'optimisation.
Ceci est particulièrement vérifié pour les structures volumineuses, où un mauvais conditionnement des paramètres d'entrée peut mener à des temps de calcul très élevés.

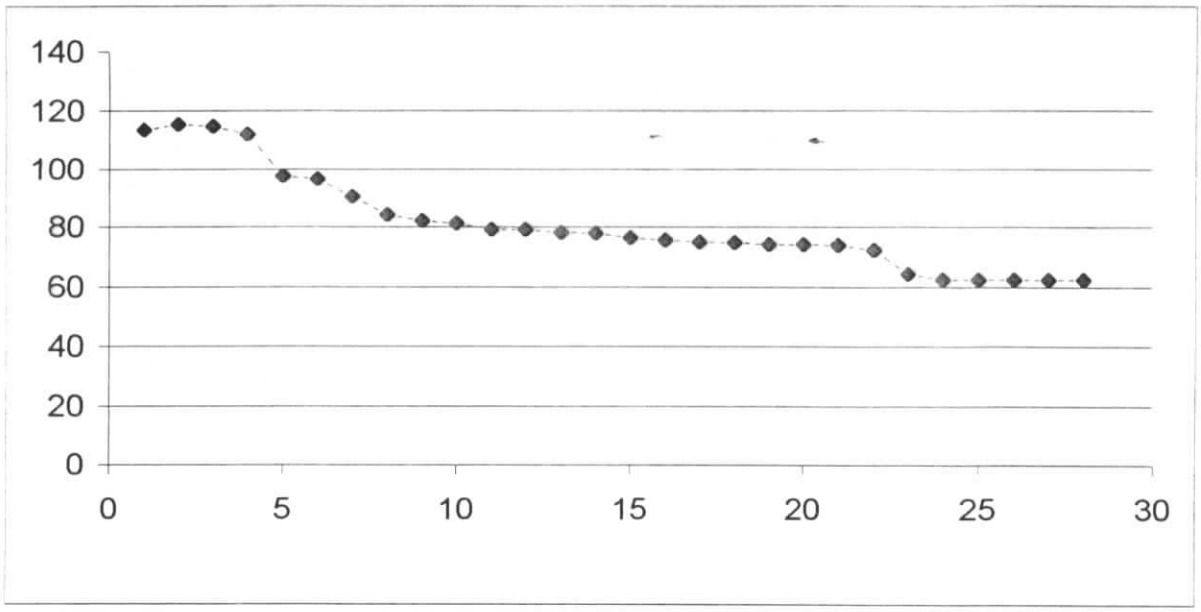
b) L'exemple du cas (1) est représentatif d'une particularité intéressante de la méthodes SQP. Le graphe V.II.1 présente au voisinage de la 11-éme itération un extremum, auquel une valeur de la fonction objectif est associée mais cette valeur ne peut être la solution car les contraintes associées aux sections en ce pont ne vérifient pas les contraintes d'optimisation du problème.
Ceci s'explique par le fait que l'algorithme soit sorti du domaine des solutions *rélisables du problème*. Nous voyons bien que l'algorithme se rattrape à partir de la 12-éme itération pour revenir au domaine des solutions et continuer le recherche de la solution.

c) L'observation des sections des barres optimales d'un cas à un autre montre que ces derniers changent. Ceci est dû au fait que le problème étudié ne présente pas de *vecteur solution* unique, la valeur optimale de la fonction objectif est certes la même, mais la combinaison des A_i composants du vecteur solution peu changer. Un tel problème est dit à *Solution non dominé*.

V.3. Exemple à douze barres :

Le treillis est constitué de douze barres dans l'espace, la structures est soumise à des sollicitations concentrées aux nœuds.
Les solutions du problème doivent être comprises entre les valeurs limites [0.6452 , 30.00] exprimées en in^2 (in^2=*6,452* cm^2). Cet exemple constitue un cas académique et été traité par plusieurs auteurs.
Le but de cet application est traiter un cas spatial plus complexe que le précédent. Les résultats successifs sont donnés dans le tableau V.III.1 en annexe.

Le graphe V.III .1 montre l'évolution de la fonction objectif (poids de la structure) en fonction des itérations.



Le nombre d'itération est égal 28, ce qui pour un exemple comme celui-ci, est raisonnable. Remarquons enfin que le graphique a un comportement asymptotique au voisinage de la solution, la fonction but se rapproche de la solution pour l'atteindre avec suffisamment de précision, et pour rendre un maximum de contraintes d'optimisation saturées.

CONCLUSIONS

Après avoir abordé les bases de la méthode des éléments finis ainsi que les grandes familles de l'optimisation et mi en œuvre la procédure d'optimisation des structures spatiales en treillis, les résultats fiables des applications traitées ont montré la validité du programme *OPTISTRUCT* et la puissance de l'algorithme SQP.

Les conditions de convergence de cet algorithme n'étant pas restrictives, le programme OPTISTRUCT converge vers la solution quel que soit le conditionnement initial du problème. Il constitue enfin un véritable outil d'aide à la conception et au pré-dimensionnement des structures spatiales en treillis et présente un intérêt économique certain ; en minimisant le poids des structures et en améliorant le taux de travail des barres du treillis.

BIBLIOGRAPHIE :

[1] IMBERT J.F. (1979), *Analyse des structures par éléments finis*
Cepadues-Editions, 473p.

[2] DHATT G., TOUZOT G. (1984), *Une présentation de la méthode des éléments finis*
Maloine S.A, Paris, 539p.

[3] BATOZ J.L., DHATT G. (1990), *Modélisation des structures par éléments finis*
(tome 1 et 2), HERMES, Paris, 455p.

[4] GAY D., GAMBELIN J. (1999), *Dimensionnement des structures, une introduction*
HERMES Sciences Publications, Paris, 676p.

[5] KHENNANE A. (1997), *Méthodes des éléments finis*
Enoncé des principes de base
Office des publications universitaires, Alger, 198p.

[6] BROUSSE P. (1981), *Mécanique analytique*
Vuibert Université, Paris, 228p.

[7] MINOUX M. (1983), *Programmation mathématique*
Dunod, Paris, 294p.

[8] SCHITTKOWSKI K. (2001), *NLPQLP : A New Fortran Implementation of a Sequential Quadratic Programming Algorithm – User's Guide-*
Report, Department of mathematics, University of Bayreuth.

[9] SCHITTKOWSKI K., ZILLOBER C., ZOTEMANTEL R. (1994),
Numerical Comparison of Nonlinear Programming Algorithms for Structural Optimization
Structural Optimization, vol. 7, No. 1, p. 1-28.

[10] SCHITTKOWSKI K. (2002), *Test Problems for Nonlinear Programming – User's Guide -*

[11] SCHITTKOWSKI K., ZILLOBER CH. (*2000*), *Nonlinear programming*
Report, Department of mathematics, University of Bayreuth.

[12] BOURDEAU M., GELINAS J. (1982), *Analyse numérique élémentaire*
Gaetan morin éditeur, Québec, 364 p.

[13] SIBONY M., MARDON J.-CI.(1984), *Approximations et équations différentielles* (II)
HERMANN, Paris, 216 p.

[14] NOUGIER J.P. (1983), *Méthodes de calcul numérique*
Masson, Paris, 317p.

[15] Cambridge University Press, Numerical Recipes Software(1988-1992)
NUMERICAL RECIPES : The Art of Scientific Computing
Website *http://www.nr.com*

[16] AIN M. (1993), *savez-vous parler Fortran ?*
De Boeck-Wesmael, Bruxelles, 501p.

[17] COATES R.C, COUTIE M.G., KONG F.K. (1980), *Structural Analysis*
English Language Book Society,
Wokingham, England, 579 p.

[18] ADELI H., KAMAL O. (1985), *Efficient Optimization of Spac trusses*
Computers and Structures
Vol.24, No. 3, pp 501-511 (1986)

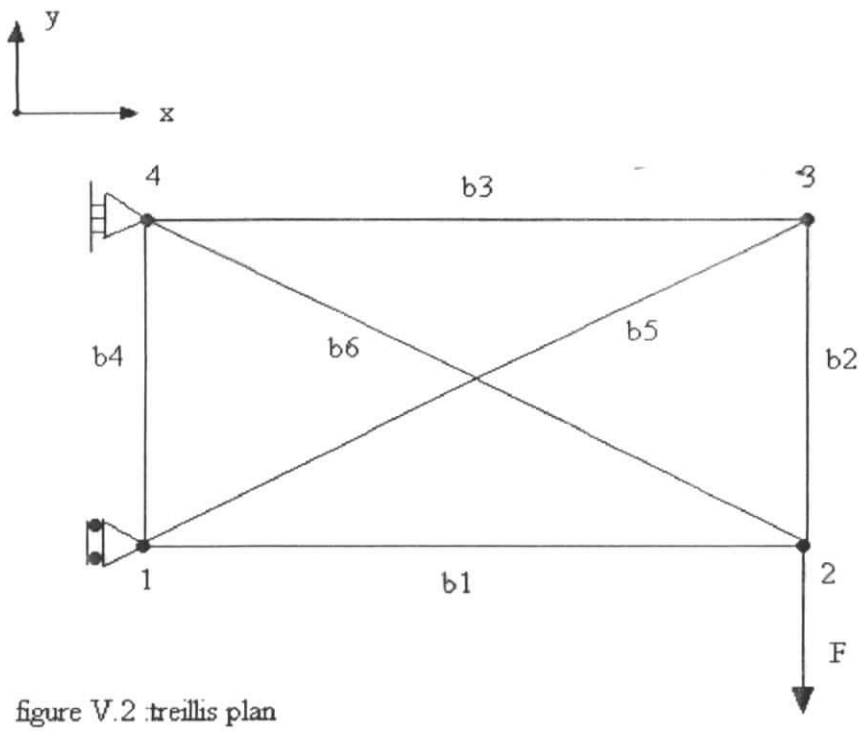


figure V.2 :treillis plan

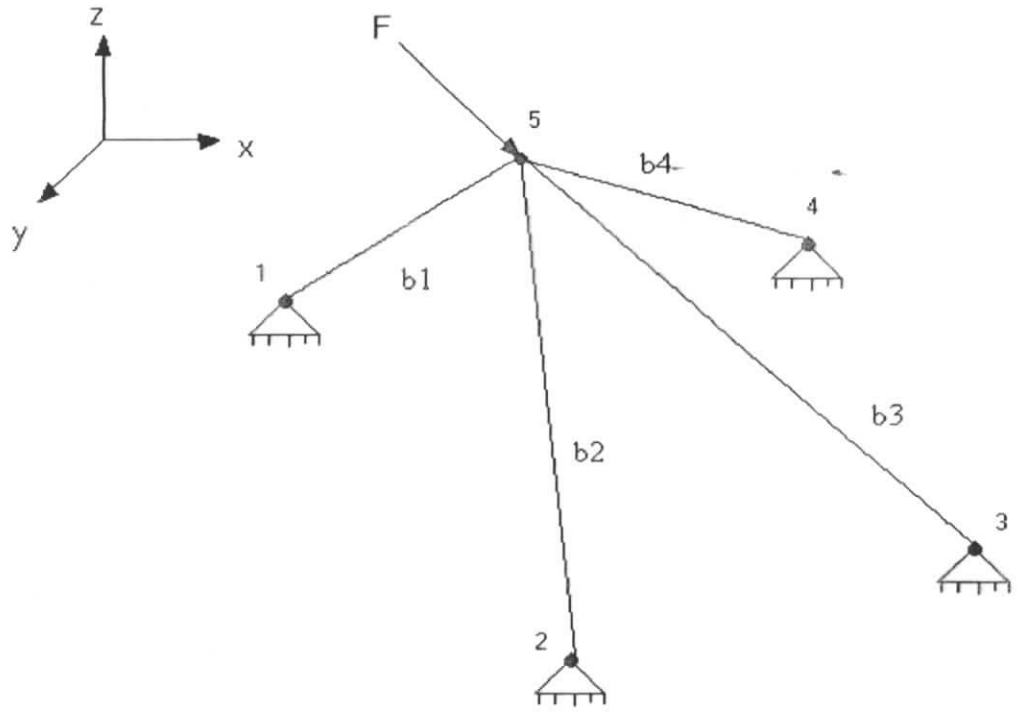


figure V.1: trellis spatial

Tableau V.I.1.

k	A_1	A_2	A_3	A_4	A_5	A_6	P
1	50.00000	50.00000	50.00000	50.00000	50.00000	50.00000 *	113.56881
2	48.50380	49.94058	50.52261	49.94058	50.96461	53.76047*	115.40361
3	47.83813	49.71497	50.24599	49.71497	50.70795	53.99962	114.89521
4	45.13309	48.66784	48.80124	48.66784	49.24276	54.12954	112.02663
5	31.68916	43.43877	41.55335	43.43877	41.86897	54.54892	97.57719
6	30.81523	43.07616	41.22903	43.07616	41.31142	54.64750	96.68474
7	26.63992	40.25058	41.50891	40.25058	35.45441	53.75800*	90.47085
8	21.78522	37.17437	42.21359	37.17437	29.30189	53.72762*	84.30137
9	20.04147	36.06258	42.44259	36.06258	27.14343	53.65296	82.06615
10	19.55463	35.49908	42.27628	35.49908	26.63817	53.61544	81.26363
11	18.42570*	34.05879	41.76662	34.05879	25.58242	53.48391	79.32125
12	18.49770*	33.81791	41.49348	33.81791	25.87030	53.39938*	79.21963
13	18.40467*	32.29838	40.31509*	32.29838	26.63280	53.12976*	78.19832
14	18.48910*	31.74492	40.37884	31.74492	26.52408	53.37349*	78.05906
15	18.79922*	27.76744	39.60402	27.76744	26.60380	54.26875*	76.42468
16	18.95400*	25.91484	39.21630	25.91484	26.71495	54.71556*	75.70643
17	19.10751*	24.21502	38.83168	24.21502	26.84791	55.15871*	75.07097
25	34.63490*	0.01000	0.01483	0.01000	0.01000	99.98248*	62.40455
26	34.63491*	0.01000	0.01482	0.01000	0.01000	99.98250*	62.40455
27	34.63491*	0.01000	0.01480	0.01000	0.01000	99.98251*	62.40456
28	34.63494**	0.01000	0.01472*	0.01000	0.01000	99.98260**	62.40458
30	34.63494**	0.01000	0.01472**	0.01000	0.01000	99.98260**	62.40458

Tableau V.II.1

Itération	Section N°1	Section N°2	Section N°3	Section N°4	Poids
1	2.50	2.50	2.50	2.50	174.296
2	1.5770	3.3215	2.2070	1.4958	143.449
3	1.4959	3.8950	2.0050	1.3558	143.071
4	1.2671	3.5323	2.2066	1.5674	142.666
5	1.3794	3.5957	1.7941	1.8238	143.038
6	1.0380	3.8972	1.7063	1.9037	141.488
7	0.8831	3.9812	1.4699	2.1829	140.680
8	0.5800	4.0968	1.3888	2.2639	138.243
9	0.5377	3.9960	1.0828	2.5152	135.419
10	0.3399	4.2669	0.8092	2.5152	130.989
11	0.1000	3.4942	0.7316	2.8148	121.521
12	0.1000	4.0108	0.7409	2.4314	120.544

Tableau V.II.2

Itération	Section N° 1	Section N° 2	Section N° 3	Section N° 4	F
1	0.10000*	0.10000*	0.10000*	0.10000*	6.97182
2	0.17271*	0.20006*	0.18997*	0.20573*	13.42677
3	0.47805*	0.35251*	0.43426*	0.19940*	24.95519
4	0.55526*	0.71164*	0.71284*	0.54369*	43.52001
5	1.38068*	1.10091*	1.54167*	0.10000*	68.55159
6	0.16087*	2.26036*	1.51905*	1.60425*	95.23914
7	0.10000*	3.27441*	1.55513*	1.97231	116.26747
8	0.38082*	3.63954*	1.93101	1.58863*	124.76445
9	0.41783*	3.70476*	1.95616	1.57446*	126.42291
10	0.41829*	3.70701*	1.95515*	1.57539*	126.46150
11	0.41829**	3.70701**	1.95515**	1.57539**	126.46154

Tableau V.II.3

K	Section N° 1	Section N° 2	Section N° 3	Section N° 4	F
1	10.00000	10.00000	10.00000	10.00000	697.1819
2	9.01000	9.01000	9.01000	9.01000	628.16094
3	8.11900	8.11900	8.11900	8.11900	566.04203
4	7.31710	7.31710	7.31710	7.31710	510.13501
5	6.59539	6.59539	6.59539	6.59539	459.81869
35	0.33128*	3.83566	1.87075	1.68647	127.5093
36	0.33057*	3.81663	1.86508	1.68737	127.15505
37	0.32919*	3.77797*	1.85387*	1.68933	126.44484
38	0.32920*	3.77946*	1.68911	1.68911	126.46153
39	0.32920**	3.77946**	1.85393**	1.68911**	126.46154

Tableau V.II.4.

K	Section N° 1	Section N°2	Section N° 3	Section N° 4	F
1	2.50000	2.50000*	2.50000	2.50000	174.29549
2	1.90947	2.68963*	2.47869	1.90947	154.26169
3	1.41731	2.85802*	2.49843	1.41731	138.40676
4	1.21954	2.93081*	2.51866	1.21954	132.33456
5	1.10624	2.97393*	2.53267	1.10624	128.92056
6	1.02978	3.00362*	2.54301	1.02978	126.64107
7	0.97367	3.02571*	2.55102	0.97367	124.98041
8	0.93740	3.04879*	2.55161	0.94466	124.09174
9	0.90763	3.29801*	2.49946*	0.97087	126.51874
10	0.92915*	3.29106	2.53634*	0.92141*	126.43000
11	0.93054*	3.29042*	2.53719	0.92146*	126.46151
12	0.93054**	3.29042**	2.53719**	0.92146**	126.46154

(*) les contrainte d'optimisation est violées c'est à dire que la contrainte dans la barre est supérieure aux contraintes limites.

(**)la contrainte d'optimisation est saturées c'est a dire que la contrainte dans la barre est aux voisinage des contraintes limites .

Tableau V.III.1.

itération	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}	P
1	5,0	5,0	5,0	5,0	5,0	5,0	5,0	5,0	5,0	5,0	5,0	5,0	**461,8**
2	4.51	4.51	4.51	4.51	4.51	4.51	4.6605	4.6606	4.6606	4.6605	4.6605	4.6605	**419,6**
3	4.0690	4.0690	4.0690	4.0690	4.069	4.069	4.2045	4.2045	4.2045	4.2045	4.2045	4.2045	**378,59 545**
4	3.6721	3.6721	3.6721	3.6721	3.6721	3.6721	3.7940	3.7940	3.7940	3.7940	3.7940	3.7940	**341,65 956**
5	3.3148	3.3148	3.3148	3.3148	3.3148	3.3148	3.4246	3.4246	3.4246	3.4246	3.4246	3.4246	**308,41 726**
6	2.9934	2.9934	2.9934	2.9934	2.9934	2.9934	3.0921	3.0922	3.0922	3.0921	3.0922	3.0922	**278,49 919**
7	2.7058	2.7040	2.7058	2.7059	2.7040	2.7040	2.7929	2.7929	2.7929	2.7929	2.792	2.7929	**251,63 886**
8	2.6921	2.4436	2.6921	2.6925	2.4436	2.4436	2.5236	2.5236	2.5236	2.5236	2.5236	2.5236	**236,28 945**
9	2.6679	2.2092	2.6679	2.6685	2.2092	2.2092	2.2813	2.2813	2.2813	2.2813	2.2813	2.2813	**222,04 744**
10	2.6214	1.9983	2.6214	2.6221	1.9983	1.9983	2.0631	2.0631	2.0631	2.0631	2.0631	2.0631	**208,34 021**
11	2.5750	1.8085	2.5750	2.5756	1.8085	1.8085	1.8668	1.8668	1.8668	1.8668	1.8668	1.8668	**195,83 434**
12	2.5347	1.6376	2.5347	2.5352	1.6376	1.6376	1.6901	1.6901	1.6901	1.6901	1.6901	1.6901	**184,63 493**
13	2.4996	1.4839	2.4996	2.5001	1.4839	1.4839	1.5311	1.5311	1.5311	1.5311	1.5311	1.5311	**174,59 82**
14	2.4690	1.3455	2.4690	2.4695	1.3455	1.3455	1.3880	1.3880	1.3880	1.3880	1.3880	1.3880	**165,59 814**
15	2.4421	1.2209	2.4421	2.4426	1.2209	1.2209	1.2592	1.2592	1.2592	1.2592	1.2592	1.2592	**157,52 360**
16	2.4185	1.1088	2.4185	2.4189	1.1088	1.1088	1.2618	1.2618	1.2618	1.2618	1.2618	1.2618	**152,68 986**
17	2.3977	1.0079	2.3977	2.3981	1.0079	1.0079	1.2641	1.2641	1.2641	1.2641	1.2641	1.2641	**148,35 527**
18	2.3793	0.9171	2.3793	2.3796	0.9171	0.9171	1.2662	1.2660	1.2663	1.2662	1.2663	1.2663	**144,46 654**
19	2.3776	0.9142	2.3776	2.3780	0.9142	0.9142	1.2664	1.2664	1.2664	1.2664	1.2664	1.2664	**144,30 463**
20	2.3473	0.9146	2.3473	2.3476	0.9139	0.9139	1.2699	1.2700	1.2700	1.2699	1.2700	1.2700	**143,28 369**
21	2.3003	0.9156	2.3003	2.3005	0.9133	0.9133	1.2756	1.2757	1.2757	1.2756	1.2757	1.2757	**141,70 446**
22	2.2243	0.9180	2.2243	2.2245 0	0.9118	0.9118	1.2854	1.2856	1.2856	1.2854	1.2856	1.2856	**139,15 991**
23	2.2267	0.9188	2.2267	2.2269	0.9112	0.9112	1.2855	1.2857	1.2857	1.2855	1.2857	1.2857	**139,24 519**
24	2.2267	0.9241	2.2267	2.2269	0.9076	0.9076 4	1.2855	1.2857	1.2857	1.2855	1.2857	1.2857	**139,22 136**
25	2.2267	0.9150	2.2267	2.2269	0.9072	0.9072	1.2855	1.2857	1.2857	1.2855	1.2857	1.2857	**139,10 368**
26	2.2267 6	0.9072 5	2.2267 6	2.2269 3	0.9072 6	0.9072 6	1.2855 0	1.2857 4	1.2857 4	1.2855 0	1.2857 2	1.2857 2	**139,00 973**
27	2.2267	0.9073	2.2267	2.2269	0.9072	0.9072	1.2855	1.2857	1.2857	1.2855	1.2857	1.2857	**139,01 063**
28	2.2267	0.9073	2.2267	2.2269	0.9072	0.9072	1.2855	1.2857	1.2857	1.2855	1.2857	1.2857	**139,01 063**
Active	*	*	*	*	*	*	*	*	*	*	*	*	