REPUBLIQUE DU SENEGAL

Ecole Polytechnique de THIES

PROJET DE FIN D'ETUDES

en vue de l'obtention du diplôme d'ingénieur de conception en genie civil

TITRE: CONCEPTION D'UN LOGICIEL POUR SIMULER PAR LA METHODE DES ELEMENTS FINIS LES ECOULEMENTS EN MILIEU POREUX SATURE

(PHASE II)

AUTEUR: KODJOVI MAWUMETO DEGUE

DIRECTEUR: GERARD A. R. SOUMA

CO-DIRECTEUR: AMADOU SARR

Juin 89.

à mon père.

à ma mère

à mes frères et sœurs

à mes amis

REMERCIEMENTS

A Monsieur Gérard SOUMA professeur,
à Monsieur Amadou SARR professeur,
à Monsieur Massamba DIENE professeur,
à tout le personnel du centre de calcul,
à tout ceux qui de près ou de loin ont contribué
au bon déroulement de ce travail,
je voudrais adresser mes sincères remerciements

SOMMAIRE

Le présent projet de fin d'étude est la phase II du projet concernant la conception d' un logiciel pour simuler, par la méthode des éléments finis, l'écoulement en milieux poreux saturé. Pour nous, il s'agit de reprendre le projet et d'écrire le logiciel en TURBO C au lieu du TURBO PASCAL 3 (langage utilisé pour la phase I) et de faire l'extension pour la résolution des écoulements permanents tridimensionnels.

Les écoulements en milieux poreux saturés sont régis par l'équation de POISSON. La formulation par éléments finis de cette équation permet d'obtenir des systèmes d'équations qui se prêtent à un traitement numérique .

Ce rapport comprend sept grandes parties :

-Dans la première partie nous présentons le modèle physique, nous démontrons dans cette partie l'équation de diffusivité et présentons les conditions aux limites .

-Dans la deuxième partie nous exposons le méthode des éléments finis. Nous présentons l'approximation par éléments finis, la discrétisation du milieu, le choix des fonctions d'intégration et introduisons les éléments de référence .

-Dans la troisième partie nous présentons le modèle numérique. Il s'agit de la formulation intégrale, la discrétisation de la forme intégrale, du choix des fonctions de pondération, de la discrétisation du domaine et formulation matricielle et enfin de la méthode d'intégration numérique.

-Dans la quatrième partie nous abordons l'assemblage et la résolution suivant la méthode frontale. Il s'agira de présenter la méthode frontale.

-Dans la cinquième partie nous présentons le logiciel, les différents programmes ses spécificités de calcul.

-Dans la sixième partie nous passons à la précision des calculs.

-Dans la septième partie nous présentons les differents tests de vérification effectués

Nous nous sommes intéressés ,dans l'élaboration de ce logiciel, à l'écoulement permanent. Néammoins dans les développements mathématiques nous avons intégré la phase transitoire. Le logiciel bien que orienté vers le traitement de l'écoulement hydrodynamique peut bien s'adapter à la résolution de tout phénomène régi par l'équation de POISSON.

TABLE DES MATIERES

CHAPITRE I

INTODUCTION

Condition essentielle au développement de la vie, la disponibilité des ressources en eau a aussi été un des principaux facteurs d'essor des civilisations, que ce soit pour l'irrigation agricole, le transport maritime ou fluvial, la production d'énergie la consommation domestique ou industrielle. Ainsi la présence de l'eau a toujours justifié l'implantation des communautés humaines sur les rives des étendues ou des cours d'eau. Les Grecs plaçaient l'eau parmi les quatre éléments fondamentaux, avec l'air, la terre et le feu. Avec le temps, et avec l'accroissement des besoins des populations et des populations elles-mêmes, le problème du manque d'eau se pose avec un peu plus d'accuité. 96.5 % de l'eau de la planète est contenu dans les océans et les mers à fortes teneurs en sel. L'eau souterraine représente les deux tiers des ressources en eau douce de la terre.

Pour cela plusieurs théories ont été élaborées fondées sur les principes de la mécanique des fluides moyennant quelques hypothèses dues au fait qu'il s'agit d'un écoulement en milieu poreux saturé. Cela conduit à une équation aux dérivées partielles du type elliptique dont la résolution mathématique pose de nombreux problèmes. Pour pallier à cela plusieurs modèles numériques ont été développés parmi lesquels:

- La méthode des éléments finis
- La méthode des différences finies

- La méthode des volumes finis

- La méthode des éléments frontiéres

La méthode des éléments finis est de nos jours la plus répandue en raison de sa grande malléabilité et de sa bonne précision. Elle fut proposée par TURNER , CHOUGH , MARTIN et TOPP en 1956

Si les premières applications de la méthode des éléments finis ont été orientées vers la résolution des équations d'équilibre en élasticité ou en élastoplasticité, il est apparu que cette méthode pouvait s'adapter à la résolution des problèmes physiques régis par des systèmes d'équations aux dérivées partielles. En particulier, les problèmes se ramenant à la résolution des équations de Laplace ou de Poisson tels que:

- les problèmes de répartition de la charge hydraulique ou des débits dans les écoulements en milieu poreux,
- les problèmes de répartition de la température ou du potentiel électrique peuvent être traités aisément par la méthode des éléments finis.

Après avoir établi l'équation régissant les écoulements en milieux poreux saturés connue sous le nom d'équation de diffusivité, nous passerons à la définition du problème et situerons le problème dans son contexte réel. Nous présenterons de façon générale de la méthode des éléments finis. Nous aborderons ensuite la formulation par éléments finis. Nous pourrons ensuite présenter comment se déroule l'assemblage

et la résolution. Ensuite on présentera le logiciel qui sera écrit en TURBO C VERSION 1.0 . Nous terminerons cette présentation par une analyse critique des résultats obtenus, phase dans laquelle nous discuterons des problèmes relatifs à la précision des calculs .

CHAPITRE II

FORMULATION MATHEMATIQUE DU MODELE PHYSIQUE

On montre en mécanique et thermodynamique des fluides, que tout problème d'écoulement de fluide Newtonien se ramène à la détermination de six(6) inconnues:

ρ : masse volumique du fluide

P : la pression

Θ : la température

Vx , Vy, Vz: Les composantes du champ de vitesse $\vec{V}$ qui sont des fonctions du temps et du point de l'espace.

Nous allons utiliser les coordonnées d'Euler , i.e. un repère fixe par rapport au laboratoire (ou au terrain) et chercher à exprimer ces six inconnues en fonction des variables spatio-temporelles.

II-1 **Equation de continuité** (conservation de la matière)

La forme différentielle de cette équation est:

$$\operatorname{div}(\rho\vec{V}) + \partial\rho/\partial t = 0 \qquad (2\text{-}1)$$

Cette équation est valable pour un fluide homogène dans un volume fixe par rapport au repère du laboratoire.

Mais pour un écoulement en milieu poreux l'équation peut être modifiée de la façon suivante:

Soit $\vec{U}$ la vitesse réelle du fluide dans chacun des pores du milieu poreux et soient ρ la masse volumique du fluide à cette échelle et w la porosité ponctuelle (w=1 dans un pore et w=0 dans un grain). A cette échelle l'équation de continuité ordinaire s'applique à l'intérieur des pores

Définissons des quantités macroscopiques ou moyennes dans le milieu poreux que nous noterons V, ρ , et m.

$\vec{V}$: vitesse fictive moyenne, ou de filtration

ρ : masse volumique moyenne

m : porosité du milieu qui est la moyenne des porosités ponctuelles.

Ces moyennes sont définies soit par intégration dans l'espace soit par une définition probabiliste.

On arrive ainsi à dériver (démonstration longue et complexe que nous ne présenterons pas) l'équation (2-2). Pour ceux qui sont intéressés ils peuvent consulter la référence (1).

$$\operatorname{div}(\rho\vec{V}) + \partial(m\rho)/\partial t = 0 \qquad (2-2)$$

Cette équation exprime la conservation de la matière au sein d'un volume fermé. Mais en hydrogéologie il faut souvent lui ajouter un terme de source correspondant au prélevement (ou injection) d'eau qu'on peut réaliser dans le milieu. Le terme de source q représentera le débit volumétrique du fluide injecté (positif) ou prélevé (négatif) l'équation s'écrit alors sous la forme :

$$\operatorname{div}(\rho\vec{V}) + \partial(m\rho)/\partial t + \rho q = 0 \qquad (2\text{-}3)$$

II-2 LA LOI DE DARCY

En dimension 1 la loi de Darcy s'exprime comme suit: $\vec{V} = K.\vec{i}$

$\vec{i}$: grandient hydraulique dans cette direction

K : perméabilité

On démontre que K varie en fonction inverse de la viscosité dynamique du fluide .

$K = K\rho g/\mu$ K : perméabilité intrinsèque

La forme générale de la loi de Darcy s'exprime par:

$$\vec{V} = -K/\mu\ (\vec{\text{grad}}\ P + \rho g\ \vec{\text{grad}}\ z) \qquad (2\text{-}4)$$

$\vec{V}$: vitesse par unité de surface (vitesse de Darcy)
c'est une vitesse fictive du fluide ($m^3/s\ m^2$)

K : perméabilité intrinsèque indépendant du fluide(m^2)

μ : viscosité dynamique du fluide(Kg/m s)

ρ : densité du fluide (Kg/m^3)

g : accélération de la pesenteur (m/s^2)

z : élevation au niveau du datum(z=0) (m)

P : pression hydraulique (N/m^2)

L'équation de Darcy peut être dérivée de l'équation de Navier-stokes, équation qui exprime la conservation du moment moyennant les hypothèses suivantes:

-les seules forces agissant sur le fluide sont les forces de

pression (différence de pression) et les forces de gravité.

-les termes d'inertie peuvent être négligés, d'où une vitesse suffisamment petite.

-la résistance à l'écoulement est dominé et est proportionnelle au flux volumétrique ou vitesse de Darcy.

II-3 Equation des écoulements en milieu poreux saturé.

A partir des deux équations établies ci-dessus nous allons pouvoir développer l'équation des écoulements en milieu poreux saturé:

Reprenons pour commencer l'équation de Darcy.

$$\vec{V} = -K/\mu\ (\vec{grad}\ P + \rho g.\vec{grad}\ z) = -K/\mu\ (\vec{grad}\ P - \rho.\vec{g})$$

l'axe des z est dirigé vers le haut alors que g va vers le bas donc $g\ \vec{grad}\ z = -\vec{g}$

$\vec{grad}\ P$ et $-\rho\vec{g}$ sont des forces par unité de volume ($Kgm/s\ m^3$)

posons $\vec{F} = \vec{grad}\ P - \rho.\vec{g}$

$$\vec{V} = -K/\mu\ .\ \vec{F}$$

$$rot\ \vec{F} = rot\ \vec{grad}\ P - rot(\rho.\vec{g})$$

$$= rot\ \vec{grad}\ P - (\vec{grad}\ \rho).\vec{g} - .rot\ \vec{g}$$

or $rot\ \vec{grad}\ P = 0$ et $rot\ \vec{g} = 0$ (g=cste)

donc $rot\ \vec{F} = -(\vec{grad}\ )\rho.\vec{g}$

$rot\ \vec{F} = 0 \iff$

-soit $\vec{grad}\ \rho\ //\ \vec{g} \Longrightarrow$ existence d'une stratification horizontale de la densité

-soit $\vec{grad}\ \rho = 0 \Longrightarrow \rho$ =cste ou a une variation négligeable

Hypothèse: posons ρ =constante.

rot $\vec{F} = 0 \Longrightarrow \vec{F}$ dérive d'un potentiel hydraulique

$$\vec{F} = \vec{grad}\, P + \rho . g . \vec{grad}\, z = \vec{grad}\,(P + \rho . g . z)$$

$$= \rho . g . \vec{grad}\,(P/\rho . g + z)$$

Posons $h = P/(\rho . g) + z$

Donc $\vec{F} = \rho . g . \vec{grad}\, h$

d'où $$\boxed{\vec{V} = - K . \rho . g/\mu . \vec{grad}\, h} \qquad (2\text{-}5)$$

posons $[K] = K\rho g/\mu$

[K] :est un tenseur de perméabilité

l'équation (2-5) devient:

$$\boxed{\vec{V} = - [K] . \vec{grad}\, h} \qquad (2\text{-}5\text{-}a)$$

Reprenons l'équation de continuité pour un écoulement en milieu poreux saturé (équation 2-3).

$$div (\rho . \vec{V}) + \partial(\rho . m)/\partial t + \rho . q = 0$$

Bien que la compressibilité de l'eau et la compressibilité du squelette solide de l'aquifère soient faibles, un changement de pression P produit une faible variation de la porosité, m et de la masse de fluide stockée par unité de volume, ρ:

$m = m(P)$; $\rho = \rho(P)$

Le terme $\partial (\rho . m)/\partial t$ peut-être développé comme suit

$$\partial (\rho . m)/\partial t = \rho . \partial m/\partial t + m . \partial \rho/\partial t$$

$$= \rho . \partial m/\partial P . \partial P/\partial t + m . \partial \rho/\partial P . \partial P/\partial t$$

$= (\rho . \partial m/\partial P + m . \partial \rho/\partial P) . \partial P/\partial t = S^m_{(p)} . \partial P/\partial t$

où $S^m_{(p)} = (\rho . \partial m/\partial P + m . \partial \rho/\partial P)$ (s^2/m^2) est la masse de fluide libérée de l'emmagasinement (ou ajouter à l'emmagasinement) dans une unité de volume d'acquifére par unité croissante ou décroissante de pression

Soit β la compressibilité du fluide de l'eau définie par:

$$\beta = 1/\rho . \partial \rho/\partial P$$

Soit α la compressibilité du squelette solide définie par:

$$\alpha = 1/(1-m) . \partial m/\partial P$$

On aura: $S^m_{(p)} = \rho . ((1-m)\alpha + m\beta)$

On sait que $h = P/\rho . g + z \Longrightarrow P = \rho . g . (h-z)$

$\partial P/\partial t = \rho . g \, \partial h/\partial t + (h-z) . g . \partial \rho/\partial t$ (g=cste)

$\rho . g . \partial h/\partial t = \partial h/\partial t - (h-z) . g . \partial \rho/\partial t = \partial P/\partial t . (1 - (h-z) . g . \partial \rho/\partial P)$

$\partial \rho/\partial P \approx 0 \Longrightarrow$ le terme $(h-z) . g . \partial \rho/\partial P$ négligeable devant 1

d'où $\rho . g . \partial h/\partial t \approx \partial P/\partial t$

d'où $\partial(\rho . m)/\partial t = S^m_{(p)} . \rho . g . \partial h/\partial t = \rho . S . \partial h/\partial t$

où $S = g . S^m_{(p)}$ est le coefficient d'emmagasinement spécifique

$\text{div} (\rho . \vec{V}) = \rho . \text{div} \vec{V}$ (ρ=constante)

d'où on a $\rho . \text{div} \vec{V} + \rho . S . \partial h/\partial t + \rho . q = 0$

$\Longrightarrow \text{div} \vec{V} + S . \partial h/\partial t + q = 0$

or équation (2-5-a) $\vec{V} = - [K] . \overrightarrow{\text{grad}} h$

d'où

$$\text{div} (- [K] . \overrightarrow{\text{grad}} h) + S . \partial h/\partial t + q = 0 \qquad (2\text{-}6)$$

Cette équation est l'équation des écoulements en milieu poreux saturé encore connue sous le nom d'équation de diffusivité

Pour un cas permanent $\partial h/\partial t = 0$

d'où $$\boxed{\text{div} (- [K] . \overrightarrow{\text{grad}}\, h) + q = 0} \qquad (2\text{-}7)$$

II-4 CONDITIONS AUX LIMITES ET CONDITIONS INITIALES

Ces équations (2-6) et (2-7) sont définies dans un domaine Ω. La résolution de l'équation (2-7) nécessite la connaissance des conditions aux limites. Sur le contour Γ du domaine Ω trois types de conditions peuvent être imposées

1- Condition de flux imposée ou condition de NEWMAN

$-K.\partial h/\partial n = q_s(p)$ sur le contour Γ_1

2- Condition de potentiel imposé ou condition de DIRICHLET

$h = \bar{h}(p)$ sur le contour Γ_2

3- Condition mixte ou de CAUCHY

$K.\partial h/\partial n = E(p).(h - h_e)$ sur le contour Γ_3

avec $\Gamma_1 \cup \Gamma_2 \cup \Gamma_3 = \Gamma$ et $\Gamma_1 \cap \Gamma_2 \cap \Gamma_3 = \emptyset$

$\bar{h}(p)$: potentiel imposé au point p

$q_s(p)$: flux imposé au point p

$E(p)$: coefficient d'échange au point p

h_e : potentiel extérieur

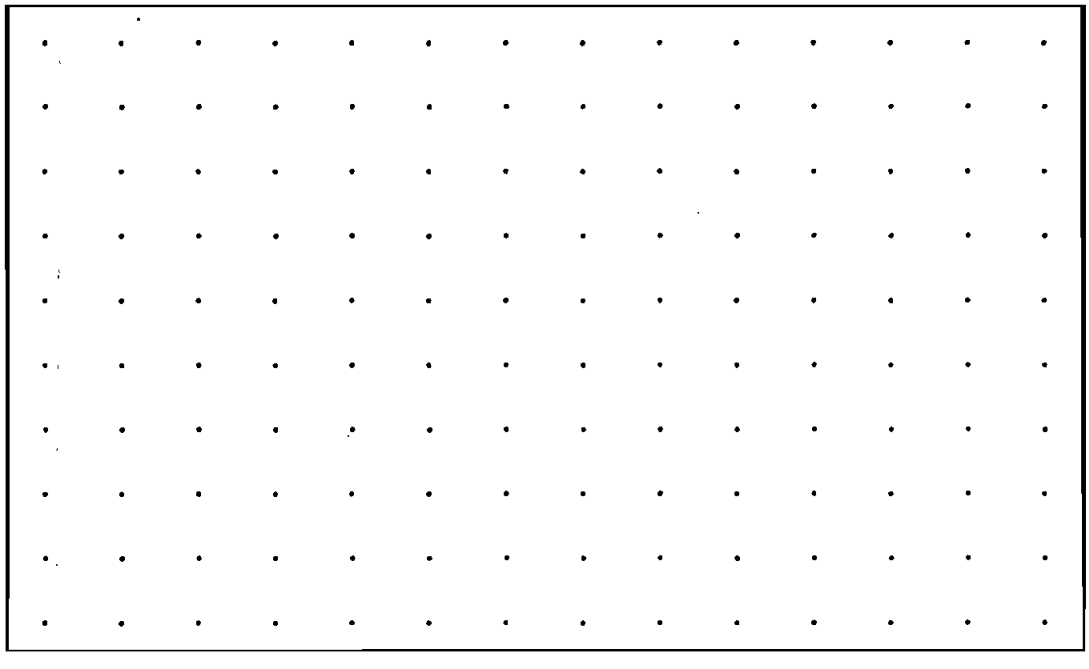
h : potentiel

$\vec{n}$: normale à la surface dirigée positivement vers l'extérieur

A ces conditions aux limites, on ajoute, pour des problèmes transitoires (équation 2-6), les conditions initiales qui sont généralement la donnée d'une charge à un temps t=0

II -5 FORMULATION DU PROBLEME

Soit un domaine Ω sur lequel on connaît en quelque points suffisament denses la charge



En un point quelconque où h est inconnue on peut trouver une approximation de h par interpolation (linéaire ou pas)

d'où $h_{app} = f(h_i)$ h_i charge connue en un point p

En remplaçant h par h_{app} dans l'équation 2-6

on aura $\boxed{\operatorname{div}\left(- [K].\overrightarrow{\operatorname{grad}}\, h_{app} \right) + S \,.\partial h_{app}/\partial t + q \neq 0}$

Posons

$$L = \operatorname{div}\left(- [K] \quad \overrightarrow{\operatorname{grad}}\, . \right) S \frac{\partial .}{\partial t} + q$$

L est un opérateur différentiel

On a: $L(h_{app}) = R$

R est appelé résidu

La meilleure approximation de h sera celle qui abaissera R à une valeur minimale sur le domaine Ω. Une manière évidente de réaliser cette condition consiste à utiliser le fait que si R est identiquement nul partout. On a alors:

$\boxed{\int_{\Omega} \Psi L(h_{app}) d\Omega = 0}$ quel que soit la fonction Ψ (2-9)

Cette manière de faire est connue sous le nom de de la méthode des résidus pondérés:

$$\boxed{\int_{\Omega} \Psi L(h_{app}) d\Omega = 0} \quad (2\text{-}10)$$

CHAPITRE III

METHODE DES ELEMENTS FINIS

L'ingénieur est amené de plus en plus à réaliser des projets complexes, coûteux, et soumis à des containtes de sécurité de plus en plus sévères. Parmi ces projets d'envergure on retrouve entre autre la gestion des nappes souterraines. Pour dominer ces projets, l'ingénieur a besoin de modèles qui lui permettent de simuler le comportement de systèmes physiques. Les sciences de l'ingénieur permettent de décrire le comportement de systèmes physiques grâce à des équations aux dérivées partielles.

L'une des méthodes utilisées de nos jours pour la résolution des systèmes d'équations aux dérivées partielles est la méthode des éléments finis. La méthode des éléments finis consiste à utiliser une approximation simple des variables inconnues pour transformer les équations aux dérivées partielles en équations algébriques. Elle fait appel aux trois (3) domaines suivants:

- -science de l'ingénieur pour construire les équations aux dérivées partielles;
- -méthodes numériques pour construire et résoudre les équations algébriques;
- -programmation et informatique pour exécuter efficacement les

calculs sur ordinnateur.

L'approche par éléments finis comprend deux principales étapes:

-la première est une discrétisation du milieu qui consiste à subdiviser le milieu continu (domaine de travail) en une série de sous domaines appelés éléments.

-la deuxième étape est une autre discrétisation de la fonction inconnue (dans notre cas la fonction ' h '). Elle consiste à remplacer la valeur exacte de h par une valeur approchée (H_{app}) la valeur de la fonction inconnue h en un point X d'un élément sera fonction des coordonnées de l'élément. Le principe des éléments finis est de chercher à rendre ($h - h_{app}$) suffisamment petit.

On passe ensuite au traitement élément par élément de l'équation. Finalement on procèdera à l'assemblage des équations traduisant le comportement de chaque élément. On obtient ainsi une équation générale linéaire ou non du système global

III-1 DISCRETISATION DU MILIEU

Elle consiste à subdiviser le domaine en des sous-domaines appelés éléments. Chaque élément doit être défini analytiquement de manière unique en fonction des coordonnées des noeuds géométriques qui appartiennent à cet élément. Cette partition du domaine en éléments doit respecter les deux règles suivantes:

-deux éléments distincts ne peuvent avoir en commun que des

points sur leur frontière commune, si elle existe, cette condition exclut le recouvrement de deux éléments

-L'ensemble de tous les éléments doit constituer un domaine aussi proche que possible du domaine Ω. Cela exclut en particulier les trous entre éléments.

Pour que les conditions ci-dessus soient respectées, la partition doit se faire de la manière suivante:

- les noeuds géométriques sont situés sur la frontière de l'élément et sont communs à plusieurs éléments.
- la frontière d'un élément à deux ou trois dimensions est formée par un ensemble de courbes ou de surfaces. Chaque portion de frontière doit être définie de manière unique à partir des coordonnées des seuls noeuds géométriques situés sur cette portion de frontière.

FORME DES ELEMENTS

Nous travaillerons uniquement avec des éléments de forme quadratique

III-2 Discrétisation de la fonction inconnue

L'approche par éléments finis consiste à remplacer la fonction exacte h(inconnue) par une fonction approchée h_{app} de telle sorte que $h_{app}-h$ soit assez petite pour l'objectif visé.

Pour construire une fonction approchée nous pouvons:

-choisir un ensemble fini de fonction dépendant de n paramètres $h(X, a_1, \ldots , a_n)$

-déterminer les paramètres $a_1, \ldots , a_n$ pour minimiser $h_{app}-h$.

Les fonctions $h(X, a_1, \ldots , a_n)$ sont choisies de manière à

être facile à évaluer sur ordinnateur, à intégrer ou dériver explicitement.

La fonction approchée h_{app} est souvent linéaire en a_i

$h_{app}=\Sigma\ P_i(x).a_i$ ou $P_i(x)$ sont des fonctions connues linéairement indépendantes et aussi indépendantes des a_i. Nous choisirons comme paramètres a_i les valeurs de la fonction h en n points distincts qui seront des noeuds de coordonnées

$X_1, \ldots, X_n$

$$h_{app}(X)=<N_1(X)\ \ N_2(X)\ \ldots\ N_n(X)>\{h_n\}=<N(X)>\{h_n\}$$

c'est ce qu'on appelle approximation nodale

$N_i(X)$ sont des fonctions d'interpolation

pour $h(X_i)=h_i \Longrightarrow N_j(X_i)=0$ si $i \neq j$.

$N_j(X_i)=1$ si $i=j$

En un point quelconque X: $\sum_{i=1}^{i=n} N_i(X) = 1$

avec n: nombre de noeuds du domaine

X: point quelconque d'un élément du domaine

Mais la construction de N_J devient difficile lorsque le nombre de noeuds n devient important. Le problème se complique encore si le domaine Ω a une forme complexe et si h_{app} doit satisfaire des conditions aux limites sur la frontière de Ω. Pour faire face à ce problème le domaine Ω est sudivisé en sous-domaines appelés éléments. L'approximation nodale se fait alors par élément. Sur un élément :

$$h^e{}_{app}(X)=<N^e(X)>\{h^e{}_n\}$$

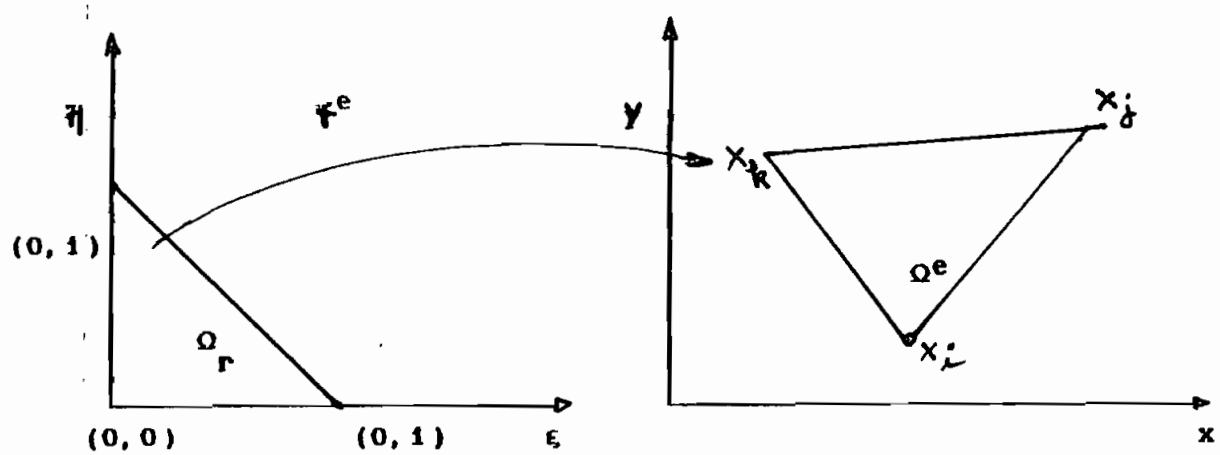
le " e " pour dire que c'est sur l'élément.

Les fonctions $h^e{}_{app}$ sur chaque élément Ω^e sont construite de

manière à être continue sur Ω^e et entre les éléments. Il faut donc que les fonctions $N_i(X)$ soient continues sur les éléments et entre les éléments. Dans certaines conditions il est nécessaire que les fonctions $N_i(X)$ aient des dérivées continues jusqu'à l'ordre S-1 (S étant l'ordre maximal de dérivabilité). N_{app} est différent sur chaque élément. Cette manière de faire ne facilite pas le traitement numérique surtout quand le nombre d'éléments devient important. Pour cela on a introduit le concept d'élément de référence.

III- 3 Elément de référence

Ce concept permet, d'une part de simplifier l'expression mathématique à intégrer et d'autre part de faciliter le traitement numérique du probléme. Un élément de référence Ω_r est un élément de forme trés simple, répéré dans un espace de référence, qui peut être transformé en chaque élément réel Ω^e par une transformation géométrique T^e. Il permet de lier tout élément du même type à un unique élément de référence. Par exemple dans le cas d'un triangle:



$\xi = \langle \xi\ \eta \rangle$
Elément de référence

$X = \langle x\ y \rangle$
Elément réel

ξ_i est un point de l'élément de référence

X_i est un point de l'élément réel

La transformation τ^e définie les coordonnées X^e de chaque point de l'élément réel à partir des coordonnées du point corespondant dans l'élément de référence .

$$\tau^e : \xi \longrightarrow X^e = X^e(\xi)$$

La transformation τ^e dépend des coordonnées des noeuds géomètriques qui définissent l'élément réel. Pour respecter les règles de partition du domaine en élément, la transformation τ^e doit être bijective, les noeuds géométriques de l'élément de référence doivent être transformés en des noeuds de l'élément réel et une frontière de l'élément réel doit être l'image d'une frontière de l'élément de référence.

Utilisons une transformation τ^e linéaire par rapport aux coordonnées $\{X_n\}$ des noeuds géométriques de l'élément réel Ω^e choisie identique pour toutes les coordonnées.

$$\tau^e : \xi \longrightarrow X^e(\xi) = \langle \theta(\xi) \rangle \{X_n\} .$$

On peut choisir les fonctions θ_i telles que les conditions précitées soient satisfaites. Elles sont habituellement des polynômes en ξ appelées fonction de transformation géométrique. Elles sont construites de la même manière que les fonctions d'interpolation $N(\xi)$

$N(\xi)$ est une fonction d'interpolation sur l'élément de référence.

Sur l'élément réel $h^e_{app} = \langle N(X) \rangle \{h_n\}$

Sur l'élément de référence $h^r_{app} = \langle N(\xi) \rangle \{h_n\}$

Pour des points qui correspondent dans la transformation $h^e_{app}=h^r_{app}$. h^e_{app} peut donc être remplacée par h^r_{app}. Les mêmes fonctions $N(\xi)$ peuvent être utilisées pour tous les éléments possédant le même élément de référence caractérisé par la forme, les noeuds géométriques et les noeuds d'interpolation.

III-4 Elément isoparamétrique

Un élément est dit isoparamétrique si les fonctions de transformations géométriques $\theta(\xi)$ sont identiques aux fonctions d'interpolations $N(\xi)$. Ceci implique que les noeuds géométriques soient confondus avec les noeuds d'interpolations. $\theta(\xi) \equiv N(\xi)$.

En choisissant des éléments isoparamétriques on aura donc qu'une seule fonction $N(\xi)$ ce qui offre une plus grande facilité dans les calculs et surtout pour le traitement à l'ordinateur.

CHAPITRE IV

MODELE NUMERIQUE :

(Formulation par élément finis)

La méthode des éléments finis discrétise la formulation intégrale de l'équation de diffusivité pour conduire à un système d'équations algébriques qui fournit une solution approchée du système.

Equation de diffusivité équation (2-6):

$$\operatorname{div}(-[K]\overrightarrow{\operatorname{grad}}\, h) + S\,(\partial h/\partial t)+q = 0 \qquad (4-1)$$

A partir de cette équation aux dérivées partielles on passe à une formulation intégrale en utilisant la méthode des résidus pondérés

IV-1 Formulation intégrale

Soit $L= \operatorname{div}(-K.\overrightarrow{\operatorname{grad}}\, .) + S\,(\partial\, .\, /\partial t) + q$ un opérateur différentiel. L'équation $L(h)=0$ correspond à l'équation de diffusivité. Si h est une solution exacte $L(h)=0$ mais en remplaçant h par son approximation nodale on aura $L(h_{app})=R\neq 0$ R est appelé résidu. La meilleure approximation sera celle qui en un certain sens abaisserait le résidu à une valeur minimale en tout point du domaine Ω. Une manière évidente de réaliser cette condition consiste à utiliser le fait que si R est identiquement nul partout, alors

$$\int_{\Omega} \Psi L(h_{app})\, d\Omega = 0 \qquad (4-2)$$

Cette manière de faire est connue sous le nom de la méthode des résidus pondérés

Ψ est une fonction quelconque des coordonnées

Elle consiste à chercher des fonctions h qui annulent la forme intégrale:

$$W(X)= \int_{\Omega} <\Psi> L(h) d\Omega \quad (4-3)$$

pour toute fonction de pondération Ψ appartenant à un ensemble Eh des solutions qui satisfont les conditions aux limites et qui sont dérivables jusqu'à l'ordre m (m=2 dans notre cas). Soit n le nombre de points où l'on désire calculer h. En choisissant n fonctions de pondération indépendantes (Ψ_1, ... , Ψ_1) on aura le système d'équations suivant:

$$\begin{aligned} W_1(h) &= \int_{\Omega} \Psi_1 L(h_{app}) d\Omega = 0 \\ &\vdots \\ W_n(h) &= \int_{\Omega} \Psi_n L(h_{app}) d\Omega = 0 \end{aligned} \qquad (4-4)$$

qui constitue un système de n équations à n inconnues. La valeur de h obtenue est une valeur approchée. Si on augmente le nombre de points n, h_{app} se rapproche plus de h exacte. Si à la limite n est infini on aura $h_{app}=h$. Dans notre cas on choisira un nombre fini de points qui ne seront autre que les noeuds de l'élément.

Reprenons l'équation de diffusivité.

$div(-K \overrightarrow{grad}\, h) + S\,(\partial h/\partial t) + q = 0$ qu'on peut encore écrit:

$-\partial\,(K^{ij}\partial h/\partial x_j)\,/\partial x_i + S\,\partial h/\partial t + q = 0$

avec: x_i coordonnée suivant l'axe i d'un point x du domaine.

K^{ij} terme de la matrice constituant le tenseur de perméabilité.

La forme intégrale s'écrit:

$$W(h)=\int_\Omega \Psi(-\partial\ (K^{ij}\partial h/\partial x_j)/\partial x_i)d\Omega + \int_\Omega \Psi(q+S\ (\partial h/\partial t)\ )d\Omega.$$

En intégrant par partie nous obtenons

$$W(h)=\int \Psi(-\partial(K^{ij}\partial h/\partial x_j)/\partial x_i)d\Omega=\int(\partial\Psi/\partial x_i)(K^{ij}\partial h/\partial x_j)d\Omega$$

$$+\int_\Gamma \Psi(-K^{ij}\partial h/\partial x_j).n_i\,d\Gamma$$

Le terme $q_s=(-K^{ij}(\partial h/\partial x_j)).n_i$ représente un flux à travers la limite Γ du domaine Ω.

q_s est un flux surfacique quand Ω est espace 3-D et un flux linéaire quand Ω représente un espace 2-D.

d'où:

$$W(h)=\int_\Omega(\partial\Psi/\partial x_i)(-K^{ij}\partial h/\partial x_j)d\Omega+\int_\Gamma \Psi q_s\,d\Gamma+\int_\Omega \Psi(\ q+\ S(\partial h/\partial t))d\Omega=0 \quad (4\text{-}5)$$

L'intégration par partie diminue les conditions de dérivabilité sur la variable h et fait apparaître un flux (q_s), mais augmente les conditions de dérivabilité de Ψ.

IV-2 Choix de la fonction de pondération

La méthode des résidus pondérés fournit selon le choix des fonctions de pondération Ψ tout un ensemble de formulation intégrale:

- formulation du type collocation par points:

  $\Psi=1$ en un point donné i et $\Psi=0$ partout ailleurs. Ceci équivaut en fait à satisfaire l'équation de diffusivité en n points distincts

- formulation du type collocation par sous domaines:

  $\Psi=1$ sur un sous-domaine particulier et zéro partout ailleurs. Ceci équivaut à annuler W(h) sur un nombre de sous-domaine suffisant pour fournir le nombre nécessaire d'équations simultanées.

- formulation du type moindres carrés;

  La méthode des moindres carrés consiste à minimiser l'expression $W_m = \int R.R \, dv$ (4-6)

  par rapport aux paramètres $h_1, h_2, \ldots, h_n$, R étant le résidu: $R=L(h_{app})$

  Cette méthode est peu utilisée car elle ne permet pas l'intégration par partie, et impose des conditions plus strictes sur l'approximation de h que la méthode de Galerkin. Par contre elle conduit à un système symétrique et défini positif quelque soit l'opérateur L

*formulation du type galerkin:

$\Psi = \partial h$ variation des fonctions inconnues h .

Soit $\Psi = \partial h = \langle N \rangle \{\partial h_n\}$

où $\{\partial h\}$ représente un vecteur colonne des variables nodales. $\Psi = \langle N \rangle \{\partial h_n\} = \langle \partial h_n \rangle \{N\}_{Ni}$. On choisit $\langle \partial h \rangle$ du, type $\langle 0_1, \ldots, 1_i, \ldots, 0_n \rangle$. On aura donc $\phi = \langle 0_1, \ldots, 1_i, \ldots, 0_n \rangle \{N_i\} = N_i$ d'où $\Psi = N_i$. On prend donc Ψ égale à une des fonctions d'approximation nodales.

On utilisera la formulation du type galerkin car conduisant à la meilleure approximation. De plus on aura toujours à manipuler moins de fonctions de différents types d'où la souplesse dans la formulation intégrale.

On a n fonctions d'interpolation indépendantes on peut donc écrire les n équations suivantes:

$$\begin{aligned} W_1 &= \int_\Omega N_1 L(h) d\Omega = 0 \\ &\vdots \\ W_n &= \int_\Omega N_n L(h) d\Omega = 0 \end{aligned} \qquad (4\text{-}7)$$

Les inconnues sont les inconnues nodales qui sont au nombre de n. Donc on a n équations à n inconnues que nous noterons:

$$W_h = \begin{bmatrix} W_1 \\ \vdots \\ W_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$W(h)=\int_{\Omega}\begin{bmatrix} N_1 \\ . \\ N_n \end{bmatrix} L(h)d\Omega=\begin{bmatrix} 0 \\ . \\ . \\ 0 \end{bmatrix} \quad \text{ou plus simplement} \quad W(h)=\int_{\Omega}\{N\}L(h)d\Omega=0 \qquad (4\text{-}8)$$

$$W(h)=\int_{\Omega}\frac{\partial\{N\}}{\partial x_i}\left(K^{ij}\frac{\partial h}{\partial x_j}\right)d\Omega+\int_{\Gamma^e}\{N\}q_s\,d\Gamma^e+\int_{\Omega}\{N\}\left(q+S\frac{\partial h}{\partial t}\right)d\Omega=0 \qquad (4\text{-}9)$$

V-3 Discrétisation du domaine et formulation matricielle

Le domaine Ω est divisé en sous-domaines, appelés élément Ω^e, et, la forme intégrale sur le domaine Ω est égale à la somme des formes intégrales sur les différents éléments:

$W(h)=\sum_{\text{élément}} W^e(h)$ où $W^e(h)$ représente la forme intégrale du résidu pondéré élémentaire.

Sur le domaine élémentaire, on aura:

$$W^e=\int_{\Omega^e}\frac{\partial\{N\}^e}{\partial x_i}\left(K^{eij}\frac{\partial h^e}{\partial x_j}\right)d\Omega^e+\int_{\Gamma^e}\{N\}^e.q_s.d\Gamma^e+\int_{\Omega^e}\{N\}^e\left(q+S\frac{\partial h^e}{\partial t}\right)d\Omega \qquad (4\text{-}10)$$

En remplaçant h^e par $h^e_{app}=\langle N\rangle^e\{h_n\}^e$ N est une fonction d'interpollation qui ne dépend uniquement que des coordonnées qui sont fixes dans le temps, h_n est indépendant des coordonnées mais dépend du temps.

$$\frac{\partial h^e_{app}}{\partial x_i}=\frac{\partial\langle N\rangle^e}{\partial x_i}\{h_n\}^e=\left\langle \frac{\partial N_1}{\partial x_i} \quad \ldots \quad \frac{\partial N_n}{\partial x_i}\right\rangle^e\{h_n\}^e$$

$$\frac{\partial h^e_{app}}{\partial t} = \langle N \rangle^e \frac{\partial \{h_n\}^e}{\partial t} = \langle N \rangle^e \begin{bmatrix} \partial h_1/\partial t \\ \vdots \\ \partial h_n/\partial t \end{bmatrix}^e$$

d'où

$$W^e = \int_{\Omega^e} \frac{\partial \{N\}^e}{\partial x_i} (K^{ij} \frac{\partial \langle N \rangle^e}{x_j} \{h_n\}^e) d\Omega^e + \int_{\Gamma^e} \{N\}^e q \, S \, d\Gamma^e$$

$$+ \int_{\Omega^e} \{N\}^e (q + S\langle N \rangle^e \partial \{h_n\}^e / \partial t) \, d\Omega^e \quad (4\text{-}11)$$

$\{h_n\}^e$ et $\dfrac{\partial \{h_n\}^e}{\partial t}$ sont indépendantes des coordonnées. Donc

$$W^e(h) = \left[\int_{\Omega^e} \frac{\partial \{N\}^e}{\partial x_i} (K^{ij} \frac{\partial \langle N \rangle^e}{\partial x_j} d\Omega^e \right] \{h_n\} + q_s \int_{\Gamma^e} \{N\}^e d\Gamma^e +$$

$$\int_{\Omega^e} \{N\}^e q \, d\Omega^e + \left[\int_{\Omega^e} \{N\}^e S \langle N \rangle^e d\Omega^e \right] \frac{\partial \{h_n\}^e}{\partial t} \quad (4\text{-}12)$$

Posons $[G]^e = \int_{\Omega^e} \frac{\partial \{N\}^e}{\partial x_i} (K^{ij} \frac{\partial \langle N \rangle}{\partial x_j}) d\Omega^e$ (4-13)

$$[f]^e = q_s \int_{\Gamma^e} \{N\}^e d\Gamma^e + \int_{\Omega^e} \{N\}^e q \, d\Omega^e \quad (4\text{-}14)$$

$$[m]^e = S \int_{\Omega^e} \{N\}^e \langle N \rangle^e d\Omega^e \quad (4\text{-}15)$$

La forme intégrale s'écrit:

$$W^e(h)=[k]^e\{h_n\}^e+[f]^e+[m]^e\,\partial\{h_n\}^e/\partial t \quad (4\text{-}16)$$

où $[G]^e$ est appelé matrice de rigidité élémentaire;

$[f]^e$ est le vecteur de sollicitation élémentaire;

$[m]^e$ est la matrice de masse élémentaire;

$\{h_n\}$ est le vecteur élémentaire des variables nodales;

Sur la totalité du domaine.

$$W(h)=\sum_{\text{éléments}}\left([G]^e\{h_n\}^e+[f]^e+[m]^e\frac{\partial\{h_n\}^e}{\partial t}\right)=0 \quad (4\text{-}17)$$

$$=[G].\{h_n\}+\{F\}+[M]\frac{\partial\{h_n\}}{\partial t}=0$$

[G] est la matrice de rigidité globale;

{F} est le vecteur global des sollicitations;

[M] est la matrice de masse globale;

$\{h_n\}$ est le vecteur globale des variables nodales;

Reprenons la matrice de rigidité élémentaire:

$$[G]^e=\int_{\Omega^e}\frac{\partial\{N\}^e}{\partial x_i}K^{ij}\frac{\partial\langle N\rangle^e}{\partial x_j}d\Omega^e$$

Le terme $\dfrac{\partial\langle N\rangle^e}{\partial x_j}$ est une matrice d (d étant le nombre de dimension de l'espace) lignes n (n étant le nombre de noeud de l'élément) colonnes

qu'on notera:

$$[B]=\begin{bmatrix} \dfrac{\partial \langle N\rangle^e}{\partial x_1} \\ \vdots \\ \dfrac{\partial \langle N\rangle^e}{\partial x_d} \end{bmatrix} \qquad [B]^T=\left\{\frac{\partial \{N\}^e}{\partial x_i}\right\}$$

d'où:

$$[G]^e=\int_{\Omega^e} B^T\,[K]\,B\,d\Omega^e \qquad (4\text{-}19)$$

IV-4 Calcul de la matrice [B]

$[B]=\left\{\dfrac{\partial \langle N\rangle^e}{\partial x_j}\right\}$ où $N=N(\xi)$.

Un point de l'espace réel est relié à un point de l'espace de référence par la transformation τ^e:

$\tau^e: \xi \longrightarrow X^e(\xi)=\langle N(\xi)\rangle\{X_n^e\}^e$ et par le fait qu'on utilise des éléments isoparamétriques on aura $\Phi \equiv N$. On aura donc:

$$[B] = \begin{bmatrix} \dfrac{\partial \langle N \rangle^e}{\partial x_1} \\ \dfrac{\partial \langle N \rangle^e}{\partial x_2} \\ \vdots \\ \dfrac{\partial \langle N \rangle^e}{\partial x_d} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial \langle N \rangle^e}{\partial \eta_1} \cdot \dfrac{\partial \eta_1}{\partial x_1} + \dfrac{\partial \langle N \rangle^e}{\partial \eta_2} \cdot \dfrac{\partial \eta_2}{\partial x^2} + \ldots + \dfrac{\partial \langle N \rangle^e}{\partial \eta_d} \cdot \dfrac{\partial \eta_d}{\partial x_d} \\ \dfrac{\partial \langle N \rangle^e}{\partial \eta_1} \cdot \dfrac{\partial \eta_1}{\partial x_2} + \dfrac{\partial \langle N \rangle^e}{\partial \eta_2} \cdot \dfrac{\partial \eta_2}{\partial x_2} + \ldots + \dfrac{\partial \langle N \rangle^e}{\partial \eta_d} \cdot \dfrac{\partial \eta_d}{\partial x_2} \\ \vdots \\ \dfrac{\partial \langle N \rangle^e}{\partial \eta_1} \cdot \dfrac{\partial \eta_1}{\partial x_d} + \dfrac{\partial \langle N \rangle^e}{\partial \eta_2} \cdot \dfrac{\partial \eta_2}{\partial x_d} + \ldots + \dfrac{\partial \langle N \rangle^e}{\partial \eta_d} \cdot \dfrac{\partial \eta_d}{\partial x_d} \end{bmatrix}$$

$$= \begin{bmatrix} \dfrac{\partial \eta_1}{\partial x} & \dfrac{\partial \eta_2}{\partial x} & \cdots & \dfrac{\partial \eta_d}{\partial x} \\ \dfrac{\partial \eta_1^1}{\partial x_2} & \dfrac{\partial \eta_2^1}{\partial x_2} & \cdots & \dfrac{\partial \eta_d^1}{\partial x_2} \\ \vdots & \vdots & \cdots & \vdots \\ \dfrac{\partial \eta_1}{\partial x_d} & \dfrac{\partial \eta_2}{\partial x_d} & \cdots & \dfrac{\partial \eta_d}{\partial x_d} \end{bmatrix} * \begin{bmatrix} \dfrac{\partial \langle N \rangle^e}{\partial \eta} \\ \dfrac{\partial \langle N^i \rangle^e}{\partial \eta_2} \\ \vdots \\ \dfrac{\partial \langle N \rangle^e}{\partial \eta_d} \end{bmatrix} = [J]^{-1} * [B_e] \qquad (4\text{-}20)$$

$$[B]^T = [B_e]^T [[J]^{-1}]^T$$

* Pour des éléments réels dans un espace réel de imension d (noté d-D) utilisant des éléments de référence dans un espace de référence de dimension d'(noté d'-D) avec d=d':

d=d' ⟹ la matrice Jacobienne [J] est carrée

La matrice de rigidité [G] s'écrit:

$$[G]^e = \int_{\Omega^e} [B_e]^T [[J]^{-1}]^T [K] [J]^{-1} [B_e] d\Omega^e$$

avec le changement de variable $d\Omega^e = \det(J) d\xi$

d'où

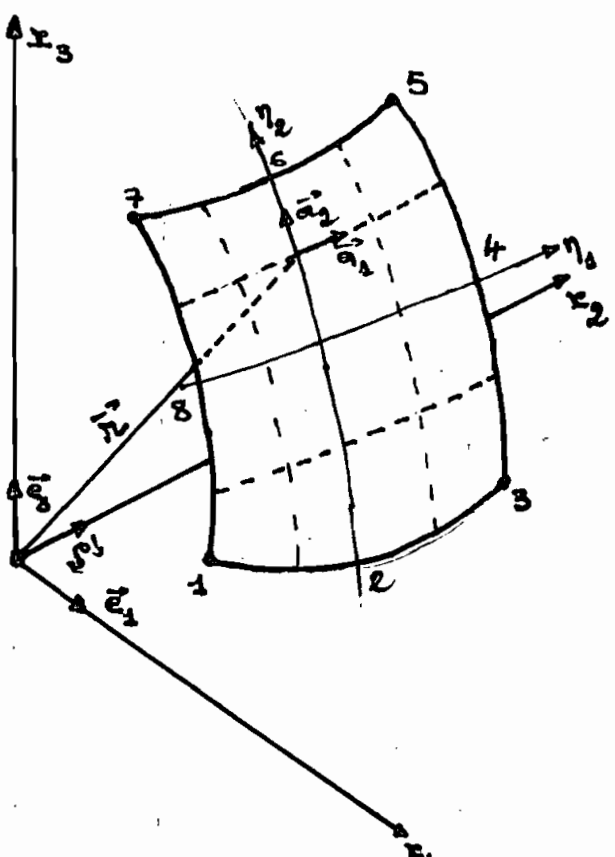
$$[G]^e=\int_{\xi}[B_e]^T[[J]^{-1T}K[J]^{-1}[B_e]\det(J)d\xi \quad (4\text{-}22)$$

* Pour des éléments réels dans un espace de référence d'-D avec d'<d.

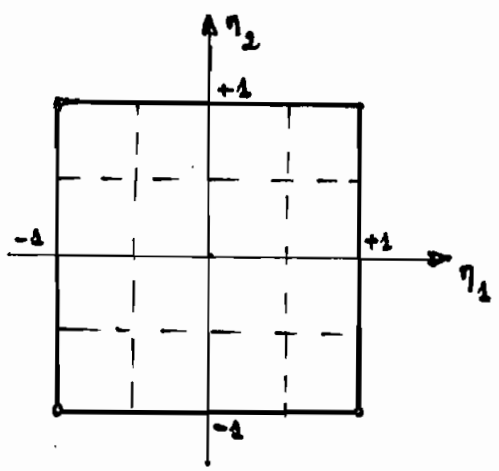
On a [J] qui n'est pas une matrice carré donc non-inversible. Ce qui rend impossible l'utilisation de la formule ci-dessus.

Pour contourner ce problème, on propose une méthode plus générale basée sur la représentation des gradients en coordonnées curvilignes.

Pour d=3 et d'=2 on a la situation suivante:



Espace réel.



Espace de référence.

Représentation du gradient en coordonnées curvilignes.

soit un point P $\vec{r}(\eta_1, \eta_2) = x_1\vec{e}_1 + x_2\vec{e}_2 + x_3\vec{e}_3$

avec $x_i = \langle N \rangle \{X_n\}$.

Dans un système global orthogonal: $\vec{grad}\ h = \dfrac{\partial h}{\partial x_j}\vec{e}_j$

Espace tridimensionnel: base orthonormée $(\vec{e}_i, \vec{e}_j, \vec{e}_k)$

Espace bidimensionnel: base covariante $(\vec{a}_1 = \dfrac{\partial \vec{r}}{\partial \eta_1}, \vec{a}_2 = \dfrac{\partial \vec{r}}{\partial \eta_2})$

Le tenseur métrique covariant est noté $g_{ik} = \{\vec{a}_i . \vec{a}_k\}$

$$\vec{grad}\ h = \frac{\partial \vec{r}}{\partial \eta_k} . g^{ik} \frac{\partial h}{\partial \eta_i} \quad \text{avec } g^{ik} = [g_{ik}]^{-1} \qquad (4\text{-}23)$$

Si on remplace h par happ $= \langle N \rangle \{h_n\}$

$$\vec{grad}\ h_{app} = \frac{\partial \vec{r}}{\partial n_k} . g^{ik} \frac{\partial \langle N \rangle}{\partial n_i} \{h_n\} \qquad (4\text{-}24)$$

$$\frac{\partial \vec{r}}{\partial \eta_k} = \frac{\partial x_1}{\partial \eta_k}\vec{e}_1 + \frac{\partial x_2}{\partial \eta_k}\vec{e}_2 + \frac{\partial x_3}{\partial \eta_k}\vec{e}_3$$

g^{ik} est un tenseur métrique contravariant.

$$\vec{grad}\ h_{app} = \vec{e}_j \left[\frac{\partial x_1}{\partial \eta_k} . g^{ik} \frac{\partial \langle N \rangle}{\partial \eta_i} \right] \{h_n\} \qquad (4\text{-}25)$$

Reprenons grad h_{app} en système global

$$\vec{grad}\ h_{app} = \vec{e}_j \frac{\partial \langle N \rangle}{\partial x_j} \{h_n\} \quad \text{où } \frac{\partial \langle N \rangle}{\partial x_j} \text{ est la ligne j de}$$

la matrice B que nous noterons $B^j{}_n$

En comparant les deux expressions on a:

$$B^j_n = \frac{\partial \langle N \rangle}{\partial x_j} = \left[\frac{\partial x_j}{\partial \eta_k} . g^{ik} . \frac{\partial \langle N \rangle}{\partial \eta_i} \right]$$

On calcule ainsi la matrice B sans avoir a calculer [J]

IV-5 Expression de $d\Omega^e$

En 1-D $\quad dL = \left| \frac{\partial r}{\partial \eta_1} \right| d\eta_1 = \det(g_{ik})^{1/2} d\eta$

En 2-D $\quad dS = \left| \frac{\partial r}{\partial \eta_1} * \frac{\partial r}{\partial \eta_2} \right| d\eta_1 d\eta_2 = \det(g_{ik})^{1/2} d\eta_1 d\eta_2$

En 3-D $\quad dV = \left| \left(\frac{\partial r}{\partial \eta_1} . \partial r / \partial \eta_2 \right) . \partial r / \partial \eta_3 \right| d\eta_1 d\eta_2 d\eta_3$

$= (\det g_{ik})^{1/2} d\eta_1 d\eta_2 d\eta_3$

C'est cette derniére méthode plus générale que nous utiliserons pour le calcul de B. Ce qui nous permet d'utiliser des éléments 1-D ou 2-D en espace 3-D

IV-6 Méthode d'intégration numérique

Vue la complexité des équations il n'est pas toujours possible d'avoir une intégration directe des équations. Les matrices $[G]^e$, $\{f\}^e$ et $[m]^e$ faisant intervenir des polynômes ou des fonctions rationnelles compliqués, leur intégration analytique n'est facile que si elles sont constituées de termes polynômiaux simples. On a donc recours à des méthodes numériques d'intégration.

La méthode numérique d'intégration consiste, d'une façon générale à utiliser r points d'intégration η_i et r coefficients de pondération w_i sur l'élément de référence de manière à intégrer exactement des polynômes d'ordres $m \leq 2*r - 1$. Soit une fonction $f(\eta)$

$$\int_{-1}^{+1} f(\eta)d\eta = \sum_{i=1}^{n} w_i f(\eta^i) \qquad (4\text{-}29)$$

Pour une fonction polynômiale un choix suffisant de nombre de points d'intégration peut conduire à des solutions exactes. Mais quand on traite des éléments compliqués de formes on obtient des intégrales de fonctions rationnelles dont l'intégration numérique ne donne qu'une valeur approchée. La précision dépend du choix des points d'intégration et de leur nombre. Ce choix conduit à différentes méthodes :

- méthode de GAUSS
- méthode de NEWTON-Côtes
- méthode directe

La méthode de Gauss est l'une des plus utilisées, pour sa simplicité et sa bonne précision. Nous allons donc utiliser la méthode de Gauss.

IV-6-1 Intégration de Gauss à une dimension

Sur l'élément de référence :

$$\int_{-1}^{+1} f(\eta)d\eta = \sum_{i=1}^{n} w_i f(\eta^i) \text{ avec } f(\eta) = a_1 + a_2\eta + \ldots\ldots + a_{2r}.\eta^{2r-1} \quad (4\text{-}30)$$

Nous utiliserons trois points d'intégration. Ce qui nous donnera des résultats exacts si la fonction à intégrer est un polynôme de dégré inférieur ou égal à (2*3-1)=5

Ces trois points sont:

$\eta_1 = -(3/5)^{1/2}$ $\quad w_1 = 5/9$
$\eta_2 = 0.0$ $\quad w_2 = 8/9$
$\eta_3 = (3/5)^{1/2}$ $\quad w_3 = 5/9$

$$\int_{-1}^{+1} f(\eta)d\eta = w_1 f(\eta_1) + w_2 f(\eta_2) + w_3 f(\eta_3) \qquad (4\text{-}31)$$

IV-6-2 Intégration de GAUSS à deux dimensions

Elle consiste à utiliser une intégration numérique à une dimension dans chaque direction. Si on utilise r_1 points dans le sens η_1 et r_2 points dans le sens η_2 on intègre exactement le produit d'un polynôme en η_1 d'ordre 2r-1 et d'un polynôme en η_2 d'ordre 2r-1. On utilise donc $r=r_1*r_2$ points d'intégration.

Pour un élément de référence carré nous utiliserons trois points dans chaque direction soit 9 points d'intégration. Les points sont les mêmes que pour 1-D dans chaque sens.

$$\int_{-1}^{+1}\int_{-1}^{+1} f(\eta_1,\eta_2)d\eta_1 d\eta_2 = \sum_{i=1}^{3}\sum_{j=1}^{3} w_i w_j f(\eta^i_1,\eta^j_2) \quad (4\text{-}32)$$

Pour un élément référence triangulaire on va utiliser la méthode de Gauss avec sept (7) points d'intégration.

$$\int_{-1}^{+1}\int_{-1}^{1-\eta_1} f(\eta_1,\eta_2)d\eta_1 d\eta_2 = \sum_{i=1}^{7} w_i f(\eta^i_1,\eta^i_2) \quad (4\text{-}33)$$

IV-6-3 Intégration de Gauss à trois dimensions

Pour des éléments de référence cubique on procède à une intégration numérique à une dimension dans chacune des trois directions.

On a:

$$\int_{-1}^{+1}\int_{-1}^{+1}\int_{-1}^{+1} f(\eta_1,\eta_2,\eta_3)d\eta_1 d\eta_2 d\eta_3$$

$$=\sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3} w_1 w_2 w_3 f(\eta^i{}_1,\eta^j{}_2,\eta^k{}_3) \qquad (4\text{-}34)$$

Elle intègre exactement le monôme $\eta_1{}^i\eta_2{}^j\eta_3{}^k$ tel que $i\leq 2R_1-1$, $j\leq 2r_2-1$; $k\leq 2r_3-1$

Pour des éléments de référence tétraédrique on va utiliser la méthode de Gauss à quinze (15) points d'intégation

$$\int_0^1\int_0^{1-\xi}\int_0^{1-\xi-\eta} f(\xi,\eta,\zeta).d\zeta.d\eta.d\eta = \sum_{i=1}^{15} w_i.f(\xi_i,\eta_i,\zeta_i) \quad (4\text{-}35)$$

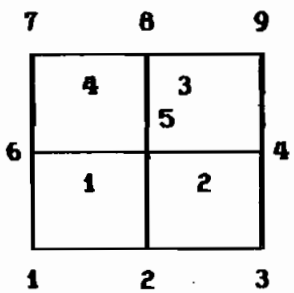
CHAPITRE V

Assemblage et résolution

Le passage des autres étapes nous permet de pouvoir calculer la matrice de rigidité élémentaire de chaque élément. La taille de cette matrice est (n^exn^e) où n^e est le nombre de noeuds de l'élément. La matrice de rigidité globale a une taille (n.n) où n est le nombre total de noeuds du domaine ($n < \Sigma ne_i$ car plusieurs éléments ont des noeuds en commun). La matrice globale s'obtient en faisant la sommation des matrices élémentaires des différents éléments. En un noeud donné tous les éléments mitoyens apportent leur contribution et ceux qui ne sont pas en contact ont des contributions nulles. Cette opération est appelée assemblage.

V-1 Exemple d'assemblage

Domaine.



Supposons que la matrice de rigidité de tout les éléments est:

$$[G]^e = \begin{bmatrix} 4 & 1 & 1 & 1 \\ 1 & 4 & 1 & 1 \\ 1 & 1 & 4 & 1 \\ 1 & 1 & 1 & 4 \end{bmatrix}, \quad [f] = \begin{bmatrix} 4 \\ 4 \\ 4 \\ 4 \end{bmatrix}$$

élément1

	1	2	5	6
1	4	1	1	1
2	1	4	1	1
5	1	1	4	1
6	1	1	1	4

élément2

	2	3	4	5
2	4	1	1	1
3	1	4	1	1
4	1	1	4	1
5	1	1	1	4

élément3

	5	4	9	8
5	4	1	1	1
4	1	4	1	1
9	1	1	4	1
8	1	1	1	4

élément4

	6	5	8	7
6	4	1	1	1
5	1	4	1	1
8	1	1	4	1
7	1	1	1	4

Assemblage: matrice globale

[G] * {h} = {F}

	1	2	3	4	5	6	7	8	9			
1	4 (1)	1 (1)	0	0	1 (1)	1 (1)	0	0	0	h_1		4
2	1 (1)	8 (1,2)	1 (2)	1 (2)	2 (1,2)	1 (1)	0	0	0	h_2		8
3	0	1 (2)	4 (2)	1 (2)	1 (2)	0	0	0	0	h_3		4
4	0	1 (2)	1 (2)	8 (2,3)	2 (2,3)	0	0	1 (3)	1 (3)	h_4		8
5	1 (1)	2 (1,2)	1 (2)	2 (2,3)	16 tous	2 (1,4)	1 (4)	2 (3,4)	1 (3)	h_5	=	16
6	1 (1)	1 (1)	0	0	2 (1,4)	8 (1,4)	1 (4)	1 (4)	0	h_6		8
7	0	0	0	0	1 (4)	1 (4)	4 (4)	1 (4)	0	h_7		4
8	0	0	0	1 (3)	2 (3,4)	1 (4)	1 (4)	8 (3,4)	1 (3)	h_8		8
9	0	0	0	1 (3)	1 (3)	0	0	1 (3)	4 (3)	h_9		4

Les chiffres entre parenthèses indiquent l'élément d'ou provient la contribution. Cette matrice contient beaucoup de zéros. Le traitement numérique d'une telle matrice entraine des opération inutiles avec des zéros "0". En plus ces zéros occupent des places en mémoire donc une sollicitation intensive de la mémoire vive. Pour optimiser le traitement, des techniques d'assemblage et de résolution ont été développées. Parmi ces méthodes nous distinguons:

-la méthode des bandes

-la méthode frontale

On utilisera la méthode frontale qui consiste à éliminer après assemblage et élimination de Gauss, les variables correspondantes à des noeuds qui ne recevrons plus de contribution. Les lignes et colonnes ainsi éliminées seront occupées, par la suite, par des variables qui apparaissent pour la première fois lors de l'assemblage de la matrice de l'élément suivant. Pour cela on proccède élément par élément. La matrice résidant en mémoire est la matrice de travail. Sa dimension varie d'un minimum (dimension du premier élément traité) à un maximum. Sa dimension est égale à la largeur du front. On définit la largeur du front comme étant égale au nombre des noeuds situés sur l'élément en assemblage et sur la frontière séparant le domaine des éléments déjà assemblés du domaine des éléments non encore assemblés. Tous les noeuds composants la largeur frontale sont appelés " variables actives ". On évite ainsi de travailler avec des zéros et on réduit considérablement la taille de la matrice de travail résidant en mémoire.

V-2 METHODE D'ELIMINATION DE GAUSS

Elle est l'une des plus anciennes. Supposons que tout l'assembage est fait et que le terme de droite est connu. L'objectif de l'élimination de Gauss est de réduire la matrice n x n en une matrice triangulaire supérieur (tous les termes sous la diagonale sont nuls). Considérons la $r^{ième}$ élimination de Gauss.

Nous aurons la matrice suivante:

$$\begin{array}{c} \\ r \\ i \end{array}\begin{array}{c} \begin{array}{cc} r & j \end{array} \\ \begin{bmatrix} g_{rr} & g_{rj} \\ g_{ir} & g_{ij} \end{bmatrix} \end{array} X \begin{bmatrix} h_r \\ h_i \end{bmatrix} = \begin{bmatrix} q_r \\ q_i \end{bmatrix} \Longrightarrow \begin{bmatrix} g_{rr} & g_{rj} \\ 0 & g^*_{ij} \end{bmatrix} X \begin{bmatrix} h_r \\ h_j \end{bmatrix} = \begin{bmatrix} q_r \\ q^*_i \end{bmatrix}$$

Avec: $$g^*_{ij} = g_{ij} - \frac{g_{ir}}{g_{rr}} \cdot g_{rj}$$

$$q^*_i = q_i - \frac{g_{ir}}{g_{rr}} \cdot q_r$$

Reprenons l'exemple précédent et réalisons l'élimination de gauss.

Après élimination la dernier matrice devient:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \end{array}\begin{array}{c} \begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array} \\ \begin{bmatrix} 4 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 7.75 & 1 & 1 & 1.75 & 0.75 & 0 & 0 & 0 \\ 0 & 0 & 3.871 & .871 & .771 & -.097 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7.675 & 1.60 & -.075 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 14.866 & 1.616 & 1 & 1.792 & .792 \\ 0 & 0 & 0 & 0 & 0 & 7.499 & .891 & .815 & -.076 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3.827 & .783 & -.044 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7.405 & .792 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3.742 \end{bmatrix} \end{array} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \end{bmatrix} = \begin{bmatrix} 4 \\ 7 \\ 3.097 \\ 6.400 \\ 11.466 \\ 5.216 \\ 2.609 \\ 4.684 \\ 2.138 \end{bmatrix}$$

La solution de ce système d'équation se trouve en faisant la

subtitution inverse. On part de la $9^{ième}$ ligne et on calcul h_9 (h_9=2.138/3.742=0.571) puis h_8 ainsi de suite jusqu'à h_1. On trouve :

h_i=0.571 pour i=1 à 9

V-3 PRESENTATION DE LA METHODE FRONTALE

Si nous regardons l'élimination de Gauss on voit bien que pour pouvoir faire l'élimination on a seulement besoin de connaître avec exactitude g_{ir}, g_{rr}, et q_r qui correspondent au noeud r. Si on a déjà assemblé tous les éléments auxquels appartient le noeud r, ces coefficients sont donc connus avec exactitude. On peut donc commencer l'élimination de Gauss avant même d'avoir toutes les contribution de q_i et de g_{ij}. Puisque la ligne r et la colonne r n'auront plus de contributions on peut donc les enlever de la matrice de travail. Voyons comment ce concept est appliqué dans la méthode frontale.

Considérons un domaine Ω subdivisé en K_p éléments. Chaque élément Ω_i ayant n_i noeuds. Pour chaque élément, on calcule une matrice GeK qui sera assemblée dans la matrice de travail résident en mémoire. Pour chaque noeud on calcule son nombre d'apparition qu'on met dans Id(Noeud). Lors de l'assemblage on crée un vecteur ICODE qui est le code d'apparition des noeuds encore dans la matrice de travail. On fait un test sur ID qui donne les quatre cas de figure suivante:

ID(noeud) > 1 ⟹ ICODE(noeud)=ID(noeud)

et ID(noeud)= - ID(noeud)+1

ID(noeud) = 1 ⟹ ICODE(noeud)=1

ID(noeud) =-1 ⟹ ICODE(noeud)=0

ID(noeud) <-1 ⟹ ICODE(noeud)=-1 et ID(noeud)=ID(noeud)+1

ICODE sert à identifier l'apparition des différents noeuds.

- ICODE(noeud)= n >1 1ière de n apparitions
- ICODE(noeud)=-1 apparition intermédiaire
- ICODE(noeud)=1 première et dernière apparition
- ICODE(noeud)=0 dernière apparition

A chaque dernière apparition on procède à une élimination de Gauss et à l'élimination de la colonne de la ligne correspondante. Et on stocke dans un fichier les données suivantes

- numéro de l'élément;
- coefficients de l'équation éliminée;
- la liste des noeuds dans la matrice de travail;
- taille de la matrice de travail;
- le terme sollicitation correspondant à ce noeud.

V-4 EXEMPLE DE CALCUL PAR LA METHODE FRONTALE

Reprenons l'exemple précédent et résolvons le par la méthode frontale

La matrice

$$GEK=\begin{bmatrix} 4 & 1 & 1 & 1 \\ 1 & 4 & 1 & 1 \\ 1 & 1 & 4 & 1 \\ 1 & 1 & 1 & 4 \end{bmatrix} \quad \text{et} \quad FEK = \begin{bmatrix} 4 \\ 4 \\ 4 \\ 4 \end{bmatrix}$$

Nous avons au total neuf noeuds. Calculons le nombre totale d'apparition de chaque noeud.

Id[1]=1 Id[2]=2 Id[3]=1 Id[4]=2 Id[5]=4

Id[6]=2 Id[7]=1 Id[8]=2 Id[9]=1

V-4-1 Assemblage du 1er élément

L'élément 1 est composé des noeuds suivants: 1, 2, 5, 6

Id[1]=1 ⟹ Icode[1]=1 ID[1]=1

Id[2]=2 ⟹ Icode[2]=2 ID[2]=2

Id[5]=4 ⟹ Icode[5]=4 ID[5]=4

Id[6]=2 ⟹ Icode[6]=2 ID[6]=2

On assemble donc la matrice GEK de l'élément dans la matrice de travail MTR VEK

Noeud	ligne	MTR	FSG
1	1	4 1 1 1	4
2	2	1 4 1 1	4
5	3	1 1 4 1	4
6	4	1 1 1 4	4

VEK est un vecteur qui, à chaque noeud, associe une ligne dans la matrice de travail .

VEKinv est un vecteur qui, à une ligne, associe le noeud qui l'occupe

FSG vecteur de sollicitation globale. En réalité ce nom est inapproprié car l'assemblage n'est pas totalement terminé. Mais puisqu'il sert à recevoir la contribution de chaque élément ce nom n'est pas tellement abusif.

Icode[1]=1 donc on peut éliminer la première ligne et la première colonne après avoir effecctué un pas d'élimination de Gauss avec pour pivot le terme MTR[0][0]

On enregistre sur fichier les informations suivantes:

/noeud/coefficients /taille/ variables / FSG/

/ 1 / 4 1 1 1 / 4 / 1 2 5 6 / 4/

Après élimination nous obtenons:

Noeud	ligne	MTR			
			2	5	6
-	1	0	0	0	0
2	2	0	3,75	0,75	0,75
5	3	0	0,75	3,75	0,75
6	4	0	0,75	0,75	3,75

	0
	3
	3
	3

Après élimination on réarrange la matrice MTR:

$$MTR=\begin{bmatrix} 3.75 & 0.75 & 0.75 \\ 0.75 & 3.75 & 0.75 \\ 0.75 & 0.75 & 3.75 \end{bmatrix} \quad FSG=\begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix}$$

V-4-2 Asemblage du second élément

L'élément 2 est composé des noeuds suivants 2 3 4 5

ID[2]=-1 ⟹ Icode[2]=0 ID[2]=0

ID[3]=1 ⟹ Icode[3]=1 ID[3]=1

ID[4]=2 ⟹ Icode[4]=2 ID[4]=-1

Id[5]=-3 ⟹ Icode[5]=-2 ID[4]=-1

On obtient l'assemblage suivant

Noeud ligne MTR

		2	5	6	3	4	FSG
2	1	7,75	1,75	0,75	1	1	7
5	2	1,75	7,75	0,75	1	1	7
6	3	0,75	0,75	3,75	0	0	3
3	4	1	1	0	4	1	4
4	5	1	1	0	1	4	4

On note que le noeud 2 est à sa dernière apparition. On l'élimine donc de la matrice de travail.

On enregistre sur fichier les informations suivantes :

/ 2 / 7,75 1,75 0.75 1 1 / 5 / 2 5 6 3 4 / 7 /

La matrice de travail devient:

Noeud ligne MTR FSG

2	1	0	0	0	0	0
5	2	0	7,35	0,58	0,77	0,77	5,42
6	3	0	0,58	3,68	-0,10	-0,10	2,32
3	4	0	0,77	-0,10	3,87	0,87	3,10
4	5	0	0,77	-0,10	0,87	3,87	3,10

On réarrange la matrice de travail qui devient:

Noeud ligne MTR FSG

5	1	7,35	0,58	0,77	0,77	5,42
6	2	0,58	3,68	-0,10	-0,10	2,32
3	3	0,77	-0,10	3,87	0,87	3,10
4	4	0,77	-0,10	0,87	3,87	3,10

On note aussi que le noeud 3 est à sa première et dernière apparition. On l'élimine de la matrice de travail.

On enregistre sur fichier les informations suivantes:

/ 3 / 0,77 -0,10 3,87 0,87 / 4 / 5 6 3 4 / 3,10 /

La matrice de travail devient après réarrangement:

Noeud Ligne MTR

$$\begin{bmatrix} 5 \\ 6 \\ 4 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad \begin{matrix} \\ 5 \\ 6 \\ 4 \end{matrix} \begin{bmatrix} & 5 & 6 & 4 \\ 5 & 7,20 & 0,60 & 0,60 \\ 6 & 0,60 & 3,675 & -0,075 \\ 4 & 0,60 & -0,075 & 3,675 \end{bmatrix} \quad \begin{bmatrix} 4,80 \\ 2,40 \\ 2,40 \end{bmatrix}$$

V -4-3 Assemblage du 3è élément

L'élément 3 est composé des noeuds suivants: 5, 4, 9, 8

Id[4]=-1 Icode[4]=0

Id[5]=-2 Icode[5]=-1 Id[5]=-1

Id[8]=2 Icode[8]=2 Id[8]=-1

Id[9]=1 Icode[9]=1

On obtient l'assemblage suivant:

Noeud ligne MTR

$$\begin{bmatrix} 5 \\ 6 \\ 4 \\ 9 \\ 8 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{bmatrix} \quad \begin{bmatrix} 11,20 & 0,60 & 1,60 & 1 & 1 \\ 0,60 & 3,675 & -0,075 & 0 & 0 \\ 1,60 & -0,075 & 7,675 & 1 & 1 \\ 1 & 0 & 1 & 4 & 1 \\ 1 & 0 & 1 & 1 & 4 \end{bmatrix} \quad \begin{bmatrix} 8,80 \\ 2,40 \\ 6,40 \\ 4,00 \\ 4,00 \end{bmatrix}$$

Icode[9]=1 ⟹ Noeud 9 est à sa première et dernière apparition . On peut donc l'éliminer.

On enregistre sur fichier les informations suivantes:

/ 9 / 1 0 1 1 4 / 5 / 5 6 4 8 9 / 4 /

La matrice de travail devient:

Noeud	ligne	MTR				FSG
5	1	10,95	0,60	1,35	0,75	0	7,80
6	2	0,60	3,675	-0,075	0	0	2,40
4	3	1,35	-0,075	7,425	0,75	0	5,40
9	4	0,75	0	0,75	3,75	0	3,00
8	5	0	0	0	0	0	0

Qu'on réarrange:

Noeud	ligne	MTR				FSG
5	1	10,95	0,60	1,35	0,75	7,80
6	2	0,60	3,675	-0,075	0	1,40
4	3	1,35	-0,075	7,425	0,75	5,40
8	4	0,75	0	0,75	3,75	3,00

Icode[4]=0 ⟹ le noeud 4 est à sa dernière apparition; on peut l'éliminer aussi.

on enregistre sur fichier les informations suivantes:

/ 4 / 1,35 -0,075 7,425 0,75 / 4 / 5 6 4 8 / 5,40/

La matrice de travail devient après élimination et réarrangement:

Noeud	ligne	MTR			
5	1	10,70	0,614	0,614	7,818
6	2	0,614	3,674	0,008	2,455
8	3	0,614	0,008	3,674	2,455

V-4-4 Asemblage du 4^{è} élément

L'élément 4 est composé des noeuds suivants: 6 5 8 7

Id[6]=-1 ⟹ Icode[6]=0

Id[5]=-1 ⟹ Icode[5]=0

Id[8]=-1 ⟹ Icode[8]=0

Id[7]=1 ⟹ Icode[7]=1

On obtient l'assemblage suivant:

Noeud	ligne	MTR				FSG
5	1	14,70	1,614	1,614	1	10,818
6	2	1,614	7,674	1,008	1	6,455
8	3	1,614	1,008	7,674	1	6,455
7	4	1	1	1	4	4

Icode[5]=0 ⟹ le noeud 5 est à sa dernière apparition. On peut donc l'éliminer.

On enregistre sur fichier les informations suivantes:

/ 5 / 14,70 1,614 1,614 1 / 4 / 5 6 8 7 / 10,818/

la matrice de travail devient après réarrangement

Noeud	ligne	MTR			FSG
6	1	7,497	0,831	0,890	5,267
8	2	0,831	7,497	0,890	5,267
7	3	0,890	0,890	0,890	3,264

Icode[6]=0 le noeud 6 est à sa dernière apparition. On peut l'éliminer .

On enregistre sur fichier les informations suivantes:

/ 6 / 7,497 0,831 0,890 / 3 / 6 8 7 / 5,267 /

La matrice de travail devient après réarrangement:

Noeud	ligne	MTR		FSG
8	1	7,405	0,791	4,683
7	2	0,791	3,718	2,639

Icode[8] = 0. Le noeud 8 aussi est à sa dernière apparition. On peut donc l'éliminer.

On enregistre sur fichier les informations suivantes:

/ 7 / 7,405 0,791 / 2 / 8 7 / 4,683 /

La matrice devient après élimination et assemblage:

Noeud ligne

[7] [1] [3,700] [2,138]

Icode[7] =1. Le noeud 7 est à sa première et dernière apparition. On peut donc l'éliminer.

On enregistre les informations suivantes:

/ 7 / 3,700 / 1 / 2 / 2,138 /

On vient donc de terminer l'assemblage et l'élimination par la méthode frontale.

V-4-5 REMARQUES

Le programme écrit fonctionne de la manière décrite ici. Mais précisons que dans le programme, si Id[noeud]>0 alors le noeud est à sa première appariton. On lui crée une place dans la matrice de travail MTR. La taille de la matrice de travail est donc augmentée de 1. Pour savoir si un noeud est à sa dernière apparition, on fait un test sur Icode[ligne]. Si Icode[ligne] =0 ou Icode[ligne]=1 alors le noeud est à sa dernière apparition et se trouve ainsi éliminé de la matrice de travail MTR. On réarrange la MTR et la taille de la matrice de travail sera réduite de 1.

Après l'assemblage et l'élimination par la méthode frontale on obtient une matrice triangulaire supérieur. On peut donc résoudre facilement par la méthode de substitution inverse.

V-5 SUBSTITUTION INVERSE

Soit le système d'équations $[G]\{h_n\}=\{q_n\}$ où [G] est une matrice triangulaire supérieure obtenue après la triangularisation, $\{h_n\}$

le vecteur des inconnues et $\{q_n\}$ le second membre obtenu après la triangularisation.

on a un système de la forme:

$$\begin{bmatrix} K_{11} & K_{12} & \cdots & K_{1n} \\ 0 & K_{22} & \cdots & K_{2n} \\ \cdot & & \cdots & \\ \cdot & & & \\ 0 & \cdot & \cdots & K_{nn} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ \cdot \\ \cdot \\ h_n \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ \cdot \\ \cdot \\ q_n \end{bmatrix}$$

La résolution de ce système se fait en partant de la dernière équation et de proche en proche on détermine toutes les inconnues:

$$h_n = q_n / K_{nn}$$

$$h_{n-1} = (q_{n-1} - K_{(n-1)n} h_n / K_{(n-1)(n-1)}$$

$$h_i = (q_i - \sum_{K=i+1}^{n} K_{iK} q_K) / K_{ii} \quad \text{pour } i<n$$

Dans notre cas la substitution inverse s'applique en partant de la dernière équation stockée pour aller vers la première. Ainsi en arrivant à une ligne donnée on connaît toutes les charges qui la composent lors de son élimination.

V-5-1 EXEMPLE

Reprenons l'exemple précédent et par substitution inverse déterminons les inconnues h. En lisant la dernière équation stockée on a:

$3,700\ h_7 = 2,138 \Longrightarrow h_7 = 0,578$

$7,405\ h_8 + 0,791\ h_7 = 4,683 \Longrightarrow h_8 = 0,571$

$7,497\ h_6 + 0,831\ h_8 + 0,890\ h_7 = 5,157 \Longrightarrow h_6 = 0,572$

$14.70\ h_5 + 1.614\ h_6 + 1.614\ h_8 + h_7 = 10.818 \Longrightarrow h_4 = 0.571$

$1.35\ h_5 - 0.075\ h_6 + 7.425\ h_4 + 0.75\ h_8 = 5.40 \Longrightarrow h_4 = 0.572$

$h_5 + h_4 + h_8 + 4\ h_9 = 4 \Longrightarrow h_9 = 0.572$

$0.77\ h_5 - 0.10\ h_6 + 3.67\ h_3 + 0.67\ h_4 = 3.10 \Longrightarrow h_3 = 0.574$

$7.75\ h_2 + 1.75\ h_5 + 0.75\ h_6 + h_3 + h_4 = 7 \Longrightarrow h_2 = 0.571$

$4\ h_1 + h_2 + h_5 + h_6 = 4 \Longrightarrow h_1 = 0.571$

V-6 PARTICULARITES DU PROBLEME IMPOSE PAR LES CONDITIONS AUX LIMITES

L'objectif de cette résolution est la détermination des charges et des débits aux différents noeuds du réseau. Les charges P/y + z dépendent du référentiel choisi: elles sont donc relatives. Il est important, dans les conditions aux limites, qu'une charge au moins soit connue en un point donné. Dans ce cas c'est le débit qui devient l'inconnu. L'élimination se fait autrement car l'inconnue n'est plus dans le premier membre mais dans le second. La triangularisation présentée ci-dessus par élimination de Gauss est pour les charges inconnues. Cette élimination est connue sous le nom d'élimination standard de Gauss. Si la charge est connue on parle d'élimination non standard de Gauss.

V-6-1 ELIMINATION NON STANDARD DE GAUSS

Soit le système d'équations suivant:

$$\begin{bmatrix} a_{11} & a_{12} & \cdot & \cdot & a_{1n} \\ a_{21} & a_{22} & \cdot & \cdot & a_{2n} \\ \cdot & \cdot & \cdot & \cdot & \\ \cdot & \cdot & \cdot & \cdot & \\ a_{n1} & a_{n2} & \cdot & \cdot & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_n \end{bmatrix}$$

Si x_r est connue le système d'équations précedent peut-être scindé en deux système d'équation:

$$y_r = a_{r1}x_1 + a_{r2}x_2 + \cdot \cdot \cdot + a_{rn}x_n$$

et

$$\begin{bmatrix} a_{11} & a_{12} & \cdot \cdot \cdot & a_{1n} \\ a_{21} & a_{22} & \cdot \cdot \cdot & a_{2n} \\ \cdot & & & \\ a_{(r-1)1} & a_{(r-1)2} & \cdot \cdot \cdot & a_{(r-1)n} \\ a_{(r+1)1} & a_{(r+1)2} & \cdot \cdot \cdot & a_{(r+1)n} \\ \cdot & \cdot & \cdot & \cdot \\ a_{n1} & a_{n2} & \cdot \cdot \cdot & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ x_{r-1} \\ x_{r+1} \\ \cdot \\ x \end{bmatrix} =$$

$$\begin{bmatrix} y_1 - a_{1r}x_r \\ y_2 - a_{2r}x_r \\ \cdot \\ y_{r-1} - a_{(r-1)r}.x_r \\ y_{r+1} - a_{(r+1)r}.x_r \\ \cdot \\ y_n - a_{nr}.x_r \end{bmatrix}$$

On remarque que la résolution du second système permet de résoudre le premier. C'est l'élimination non-standard de Gauss. Cette élimination affecte seulement les termes du second membre

CHAPITRE VI

PRESENTATION DU LOGICIEL

Ce logiciel traite du problème des écoulements en milieu poreux saturé.

Pour une question de commodité nous avons subdivisé le programme en quatre parties (quatre programmes)

-Un programme de saisie des données " SAISI "

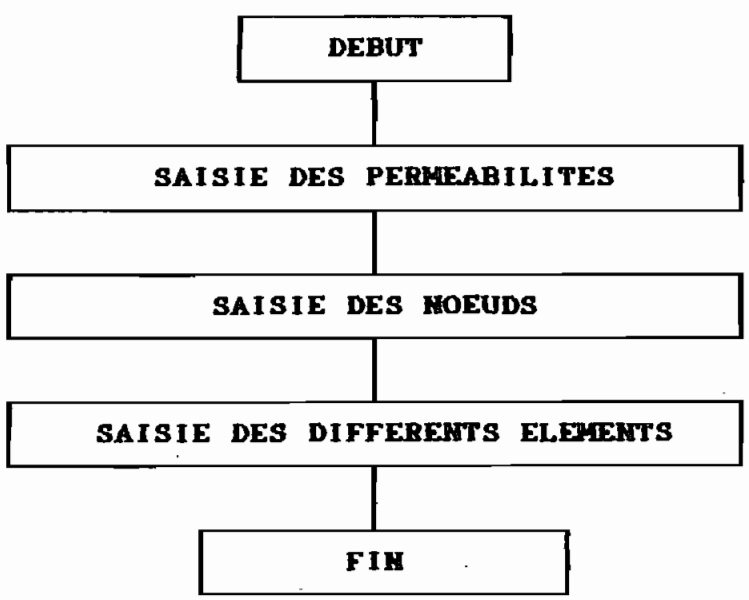
-Un programme de résolution " RESOL "

-Un programme de sortie des données et des résultats de calcul " SORTI "

-Un dernier programme qui lance les trois premiers suivant l'option " EMPSATEPT "

VI-1 SAISIE DES DONNEES

Organigramme du programme



VI-1-1 SAISIE DES PERMEABILITES

Ce sous-programme saisit les caractéristiques du sols. Il s'agit des perméabilités du sol. En réalité on devrait parler de matrice de perméabilités du milieu, qui est un tenseur. Elle est symétrique. Nous la saisirons toujours comme une matrice 3x3 et en raison de la symétrie on ne saisit que la moitié supérieure soit seulement 6 données.

$$[K] = \begin{bmatrix} K1 & K2 & K3 \\ & K4 & K5 \\ SYM & & K6 \end{bmatrix}$$

Sur un élément, on considère que la perméabilité est constante. Pour pouvoir simuler l'hétérogénéité du sol, on peut le subdiviser en plusieurs classes de perméabilité. Un élément ne peut avoir qu'une classe de perméabilité. La subdivision du domaine doit donc être faite en tenant compte de la stratification du milieu. Les différentes classes de perméabilités saisies sont stockées dans un fichier appelé clasper qui est un fichier texte à accès séquentiel.

VI-1-2 SAISIE DES NOEUDS

Ce deuxième sous-programme fait la saisie des coordonnées des différents noeuds (X, Y, Z), des conditions aux limites imposées (Hhcx) :

- Hhcx=-1 charge imposée
- Hhcx=0 débit imposé

La condition aux limites imposée est déterminée par la valeur que prendra Hhcx. Hhcx ne peut prendre que ces deux valeurs mentionnées ci-dessus.

- Les coordonées (X, Y, Z) sont en mètre (m)
- La charge hydraulique (h) est en (m)
- le débit (q) est en (m^3/s)

Les données sont stockées dans un fichier nommé CORG qui est un fichier texte à accès séquentiel

VI-1-3 Saisie des différents éléments

Dans ce sous-programme on saisit les données suivantes :

- Le type de l'élément. Il peut prendre les valeurs suivantes: 23, 36, 48, 49, 410, 820, 827.

  Le type, c'est pour distinguer les différentes catégories d'éléments. Le premier numéro correspond au nombre de sommets de l'élément, et le reste correspond au nombre de noeuds composant l'élément.
- La classe.

  Pour dire à quelle classe appartient l'élément en question.
- L'alimentation distribuée Ald.

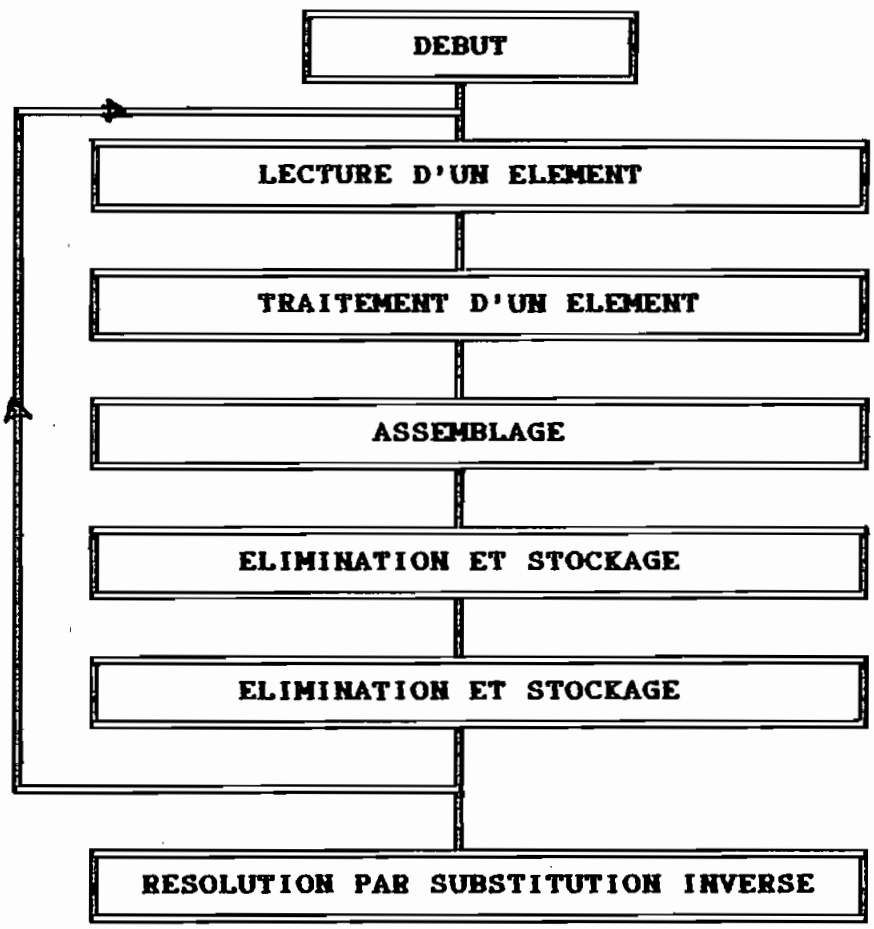
  Pour simuler des conditions d'alimentation uniformement distribuée sur tout l'élément.
  - * Ald < 0, prélèvement uniforme sur tout le domaine
  - * Ald > 0, apport uniforme sur tout le domaine.

- Le numéro des différents noeuds de l'élément.

Les données sont stockées dans un fichier nommé LOCE. Ce fichier est un fichier binaire à accès séquentiel.

VI-2 RESOLUTION " RESOL "

ORGANIGRAMME



VI-2-1 Lecture d'un élément

Il s'agit de parcourir enregistrement par enregistrement tout le fichier LOCE. On fait la lecture d'un élément (un enregistrement) puis on passe au traitement de cet élément.

VI-2-2 Traitement d'un élément

Suivant le type d'élément on passe à l'un des sous programme suivant:

- TYPE23 pour les éléments de type 23
- TYPE36 pour les éléments de type 36
- TYPE48 pour les éléments de type 48
- TYPE49 pour les éléments de type 49
- TYPE410 pour les éléments de type 410
- TYPE820 pour les éléments de type 820
- TYPE827 pour les éléments de type 827

Le sous programme TYPE23 contient la procédure pour calculer la matrice B, ainsi que les fonction d'interpollations. Les sous programmes TYPE36, TYPE48, TYPE49, font appel au sous programme "matcovar2d" pour le calcul de la matrice B. Les fonctions d'intégration sont contenues dans une librairie. Les sous-programmes TYPE410, TYPE820, TYPE827, font appel au sous-programme "matcovar3d" pour le calcul de la matrice B. Les fonctions d'intégration sont contenues dans une librairie. Ces différentes fontions font appel ensuite aux mêmes sous-programmes. Soit:

- Conver qui transfere une matrice a dans une matrice B
- Transpo qui calcule la tranposée d'une matrice
- Prodmat qui fait le produit de deux matrices
- Sommat qui fait la somme de deux matrices

Pour chaque point d'intégration nous calculons la matrice élémentaire K^e et le vecteur Fk^e. A chaque passage nous évaluons la matrice K^e et le vecteur Fk^e aux point de Gauss donné puis

nous faisons la sommation terme à terme de toutes ces évaluations Nous obtenons ainsi finalement la matrice élémentaire, notée Gek et le vecteur de sollicitation élémentaire, notée Fek qui seront assemblés dans la matrice globale de travail MTR et le vecteur de sollicitation global de travail FSG.

VI-2-3 ASSEMBLAGE

Le processus d'assemblage comprend deux parties:

-création de place dans la matrice de travail pour les nouveaux noeuds qui sont apparus. Cela concerne le sous-programme "REDIMMTR"

-assembige de la matrice Gek dans la matrice MTR dans le sous-programme "Assemblage"

Après l'assemblage de la nouvelle matrice on passe à l'élimination des variables qui apparaissent pour la dernière fois c'est le processus d'assembage et d'élimination.

VI-2-4 ELIMINATION ET STOCKAGE

On élimine les variables qui sont à leur dernière apparition. Suivant la valeur de Hhcx ont passe à l'élimination standard de GAUSS ou à l'élimination non standard de GAUSS. On stocke les données du noeud éliminé dans un fichier "EFRONT". Après avoir éliminer tous les noeuds qui apparaissent pour la dernière fois, on recommence le processus en lisant l'élément suivant et ainsi de suite juisqu'à la lecture de tous les éléments.

VI-2-5 RESOLUTION PAR SUBTITUTION INVERSE

Dans cette partie du programme on calcule les charges et les débits. On part de la dernière équation stockée et on remonte le fichier " EFRONT " vers le début. Pour un noeud à charge connue, on calcule le débit à ce noeud, et à un noeud où on connait le débit, on calcule la charge.

Ceci met fin au programme qui fait la resolution.

VI-3- SORTIE " SORTI "

Ce programme, comme son nom l'indique, fait la sortie des données et des résultats suivant l'option à l'écran ou sur imprimante.

VI-4 EMPSAT (Ecoulement en Milieu Poreux SATuré)

Ce programme est le programme principal qui lance les trois premiers suivant l'option choisie. Il est clair que les trois programmes sont dépendents et qu'on ne peut pas lancer la résolution sans avoir saisi les données.

CHAPITRE VIII

PRECISION DES CALCULS

Les sources d'erreurs sont nombreuses. On peut les regroupés en deux grands groupes:

- -les erreurs liées à l'approximation par élément finis
- -les erreurs liées au traitement numérique à l'ordinateur.

VII-1 LES ERREURS LIEES A L'APPROXIMATION PAR ELEMENT FINIS

VII-1-1 ERREURS DE DISCRETISATION GEOMETRIQUE

Lorsque le domaine qu'on étudie a des frontières complexes et différntes des frontiéres des éléments utilisés dans la modélisation du domaine, on a une erreur de discrétisation géométrique. Cette erreur peut être réduite en diminuant la taille des éléments à frontière plus complexes. Dans notre cas, à cause du fait qu'on à utiliser que des éléments quadratique, on doit subdiviser le domaine en éléments dont les frontiéres peuvent être assimiler à des fonctions quadratiques. On aura ainsi des éléments plus ou moins grand, suivant la complexité du domaine à modeliser, et qui n'aurons pas forcement la même taille.

VII-1-2 ERREUR D'APPROXIMATION

C'est l'erreur qu'on a commise en remplaçant h par h_{app}. L'erreur d'approximation en tout point x de l'élément réel Ω^e est définie par :

$$e(x)=h(x) - h_{app}(x).$$

L'erreur au point ξ de l'élément de référence est:

$$e(\xi)=h(\xi) - h_{app}(\xi).$$

En deux points x et ξ qui correspondent par la transformation τ^e définie dans le paragraphe: **ELEMENT DE REFERENCE**, les erreurs e(x) et $e(\xi)$ prennent la même valeur.

$$|e_{max}| = |e| = \text{Maximum sur } \Omega^e \text{ de } |e(x)|$$

Une expression approchée de l'erreur est : $e \leq C.L^{\alpha}$, où C et α sont des constantes qui dépendent du type d'approximation utilisé (type d'élément) et L la taille de l'élément. Elle tend vers zéro quand la taille de l'élément tend vers zéro.

Pour améliorer la précision de l'approximation il faut:

- -Soit diminuer L, donc la dimension de chaque élément
- -Soit augmenter n, i.e. utiliser une approximation dont la base polynômiale soit complète jusqu'à un ordre plus élévé.

VII-2 - ERREURS DUES AU TRAITEMENT NUMERIQUE.

VII-2-1 ERREURS D'INTEGRATION

L'intégration exacte des matrices élémentaire et des vecteurs sollicitations élémentaires nécessite l'intégration exacte de chacune de leurs termes. Ceci n'est possible, avec la méthode d'intégration numérique de Gauss que si ces termes sont

des polynômes de dégré fini. Ce qui est en général le cas lorsque la matrice jacobienne est une constante. Cette erreur augmente avec la distortion de l'élément. Le nombre de points de GAUSS choisi détermine l'ordre de précision de cette intégration numérique (approximative) par rapport à une intégration analitique (exacte) si elle existe. Pour une intégration sur un segment ou on à utiliser r points d'intégration on estime l'erreur à :

$$e = \frac{2^{(2.r+1)}\ (r!)^4}{(2r+1).[(2r)!]^3} \cdot \frac{d^{(2.r)}\ Y}{d\xi^{2.r}}$$

Y étant la fonction à intégrer sur le domaine Ω.

Pour un nombre suffisant de points d'intégration, cette erreur devient très négligeable.

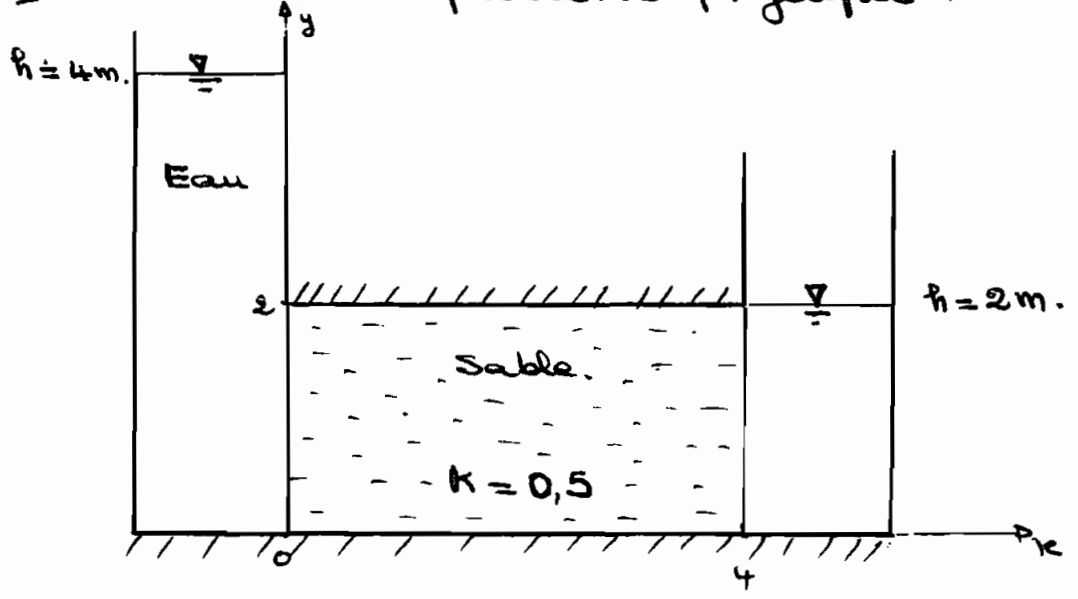
VII-2-1 ERREURS LIEES A LA MACHINE.

Ces erreurs de l'utilisation de valeurs approchées pour représenter des valeurs exactes ou des expressions mathématiques exactes. Les plus importantes sont les erreurs d'arrondis et les erreurs de troncatures. Ces erreurs sont négligeables quand la précision des données sont compatibles avec la précision de l'ordinnateur. En simple précision ces erreurs peuvent rapidement devenir très important à cause surtout de la faiblesse de la perméabilité. Mais elles sont négligeable en double précision avec virgule flottante

CHAPITRE VIII

TESTS DE VERIFICATION.

I- Soit le problème physique suivant:



On doit déterminer le débit en supposant une épaisseur de sable de 1 m.

1- Solution analytique.

Loi de Darcy. $Q = K \cdot A \cdot \frac{\Delta h}{L}$

On a $K = 0,5 \ m/s$

$A = 2 \times 1 = 2 \ m^2$

$\Delta h = 4 - 2 = 2 m.$

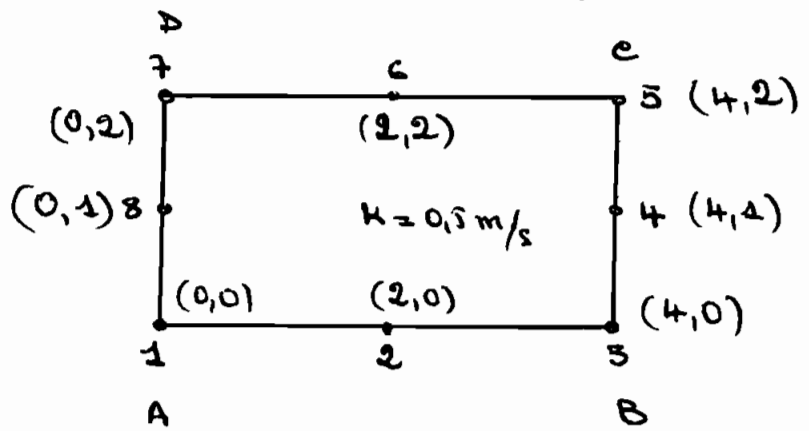
$L = 4 m.$

$Q = 0,5 \times 2 \times \frac{2}{4} = 0,5 \ m^3/s$

$$Q = 0,5 \ m^3/s.$$

2. Solution par la méthode numérique.

2-1. Modelisation par un élément Type 48.



Aux noeuds 1, 8, 7 la charge est connue egale à 4 m.

Aux noeuds 3, 4, 5 la charge est également connue égale à 2 m.

Aux autres noeuds par le fait d'a la frontiere imperméable $q_s = 0$ (Condition de Newman.)

A la page suivante on peut voir les données et les résultats de calcul obtenus.

***** NOMBRE TOTAL D'ELEMENT ********** 1
***** NOMBRE TOTAL DE NOEUD ********** 8
***** NOMBRE TOTAL DE PERMEABILITE ***** 1

** PERMRABILITE**

CLASSE	K1	K2	K3	K4	K5	K6
1	0.50000	0.00000	0.00000	0.50000	0.00000	0.00000

*** CONNECTIVITE ***

ELT	Nx	TYPE	CLASSE	ALD	ND1	ND2	ND3	ND4	ND5	ND6	ND7	ND8	ND9
1	48		1	0.000	1	2	3	4	5	6	7	8	

*** COORDONNEES ET CONDITIONS AUX LIMITES ****

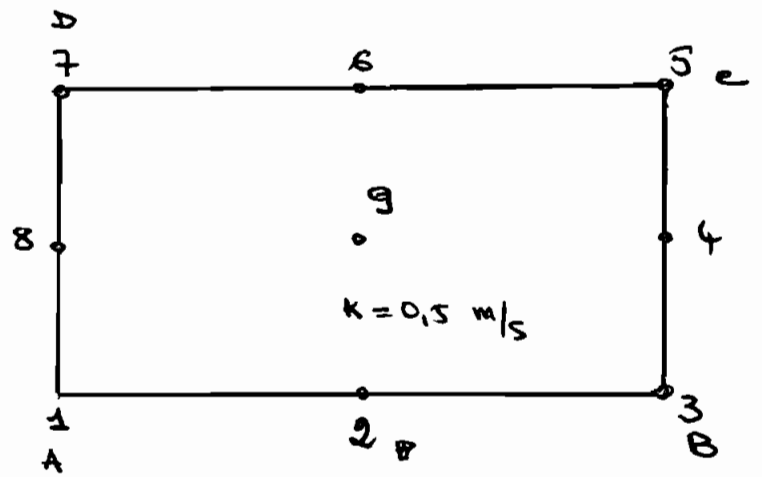
NOEUD	Nx	XCOORD (m)	YCOORD (m)	ZCOORD (m)	Hhcx	Valh (m)	DEBIT (m3/s)
1		0.000	0.000	0.000	-1	4.000	0.0000
2		2.000	0.000	0.000	0	0.000	0.0000
3		4.000	0.000	0.000	-1	2.000	0.0000
4		4.000	1.000	0.000	-1	2.000	0.0000
5		4.000	2.000	0.000	-1	2.000	0.0000
6		2.000	2.000	0.000	0	0.000	0.0000
7		0.000	2.000	0.000	-1	4.000	0.0000
8		0.000	1.000	0.000	-1	4.000	0.0000

****
RESULTATS CALCULES

VARIABLE Nx	CHARGE CALCULEE (m)	DEBIT CALCULE (m3 / s)
1	4.000000	0.08333
2	3.000000	0.00000
3	2.000000	-0.08333
4	2.000000	-0.33333
5	2.000000	-0.08333
6	3.000000	0.00000
7	4.000000	0.08333
8	4.000000	0.33333

******* SOMME DES DEBITS ENTRANT : 4.999989867e-001
******* SOMME DES DEBITS SORTANT : -4.999989867e-001
******* SOMME DE TOUT LES DEBITS : 0.000000000e+000

2.2 - Modélisation par un élément Type 49 avec simulation d'une alimentation uniforme linéairement distribuée



$h = 2m$ aux noeuds 3, 4, 5

Sur AD on a une alimentation distribuée de $0,25\ m^3/s.m$.

Les 2 pages suivantes nous montre les données et les resultats obtenus.

***** NOMBRE TOTAL D'ELEMENT ********** 2
***** NOMBRE TOTAL DE NOEUD ********** 9
***** NOMBRE TOTAL DE PERMEABILITE ***** 2

** PERMRABILITE**

CLASSE	K1	K2	K3	K4	K5	K6
1	0.50000	0.00000	0.00000	0.50000	0.00000	0.00000
2	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

*** CONNECTIVITE ***

ELT Nx	TYPE	CLASSE	ALD	ND1	ND2	ND3	ND4	ND5	ND6	ND7	ND8	ND9
1	49	1	0.000	1	2	3	4	5	6	7	8	9
2	23	2	0.250	7	8	1						

*** COORDONNEES ET CONDITIONS AUX LIMITES ****

NOEUD Nx	XCOORD (m)	YCOORD (m)	ZCOORD (m)	Hhcx	Valh (m)	DEBIT (m3/s)
1	0.000	0.000	0.000	0	0.000	0.0000
2	2.000	0.000	0.000	0	0.000	0.0000
3	4.000	0.000	0.000	-1	2.000	0.0000
4	4.000	1.000	0.000	-1	2.000	0.0000
5	4.000	2.000	0.000	-1	2.000	0.0000
6	2.000	2.000	0.000	0	0.000	0.0000
7	0.000	2.000	0.000	0	0.000	0.0000
8	0.000	1.000	0.000	0	0.000	0.0000
9	2.000	1.000	0.000	0	0.000	0.0000

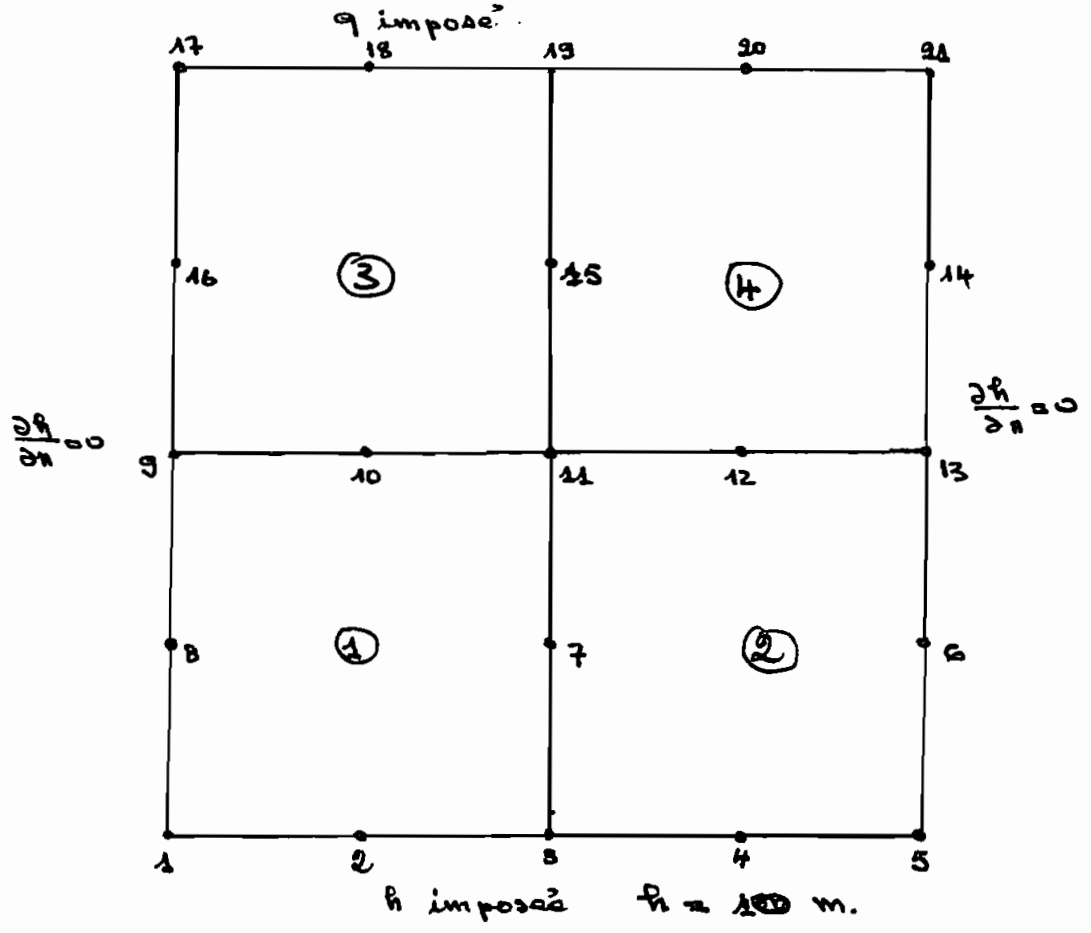
****

RESULTATS CALCULES

VARIABLE N.x	CHARGE CALCULEE (m)	DEBIT CALCULE (m3 / s)
1	4.000000	0.08333
2	3.000000	0.00000
3	2.000000	-0.08333
4	2.000000	-0.33333
5	2.000000	-0.08333
6	3.000000	0.00000
7	4.000000	0.08333
8	4.000000	0.33333
9	3.000000	0.00000

******* SOMME DES DEBITS ENTRANT : 4.999989867e-001
******* SOMME DES DEBITS SORTANT : -4.999989867e-001
******* SOMME DE TOUT LES DEBITS : 0.000000000e+000

II



$K = 1\ m/s$

Domaine 1000 m x 1000 m.

Domaine subdivisé de type 48.

Noeuds 1, 2, 3, 4, 5 charge imposée = 10. m.

Noeuds 17, 18, 19, 20, 21 débit imposé

$q_{17} = 0,083333.\ m^3/s$

$q_{18} = 0,333333.\ m^3/s$

$q_{19} = 0,166667.\ m^3/s$

$q_{20} = 0,333333\ m^3/s$

$q_{21} = 0,083333.\ m^3/s.$

***** NOMBRE TOTAL D'ELEMENT ********** 4
***** NOMBRE TOTAL DE NOEUD ********** 21
***** NOMBRE TOTAL DE PERMEABILITE ***** 1

** PERMRABILITE**

CLASSE	K1	K2	K3	K4	K5	K6
1	1.00000	0.00000	0.00000	1.00000	0.00000	0.00000

*** CONNECTIVITE ***

ELT Nx	TYPE	CLASSE	ALD	ND1	ND2	ND3	ND4	ND5	ND6	ND7	ND8	N
1	48	1	0.000	1	2	3	7	11	10	9	8	
2	48	1	0.000	3	4	5	6	13	12	11	7	
3	48	1	0.000	9	10	11	15	19	18	17	16	
4	48	1	0.000	11	12	13	14	21	20	19	15	

*** COORDONNEES ET CONDITIONS AUX LIMITES ****

NOEUD Nx	XCOORD (m)	YCOORD (m)	ZCOORD (m)	Hhcx	Valh (m)	DEBIT (m3/s)
1	0.000	0.000	0.000	-1	10.000	0.0000
2	250.000	0.000	0.000	-1	10.000	0.0000
3	500.000	0.000	0.000	-1	10.000	0.0000
4	750.000	0.000	0.000	-1	10.000	0.0000
5	1000.000	0.000	0.000	-1	10.000	0.0000
6	1000.000	250.000	0.000	0	0.000	0.0000
7	500.000	250.000	0.000	0	0.000	0.0000
8	0.000	250.000	0.000	0	0.000	0.0000
9	0.000	500.000	0.000	0	0.000	0.0000
10	250.000	500.000	0.000	0	0.000	0.0000
11	500.000	500.000	0.000	0	0.000	0.0000
12	750.000	500.000	0.000	0	0.000	0.0000
13	1000.000	500.000	0.000	0	0.000	0.0000
14	1000.000	750.000	0.000	0	0.000	0.0000
15	500.000	750.000	0.000	0	0.000	0.0000
16	0.000	750.000	0.000	0	0.000	0.0000
17	0.000	1000.000	0.000	0	0.000	0.0833
18	250.000	1000.000	0.000	0	0.000	0.3333
19	500.000	1000.000	0.000	0	0.000	0.1667
20	750.000	1000.000	0.000	0	0.000	0.3333
21	1000.000	1000.000	0.000	0	0.000	0.0833

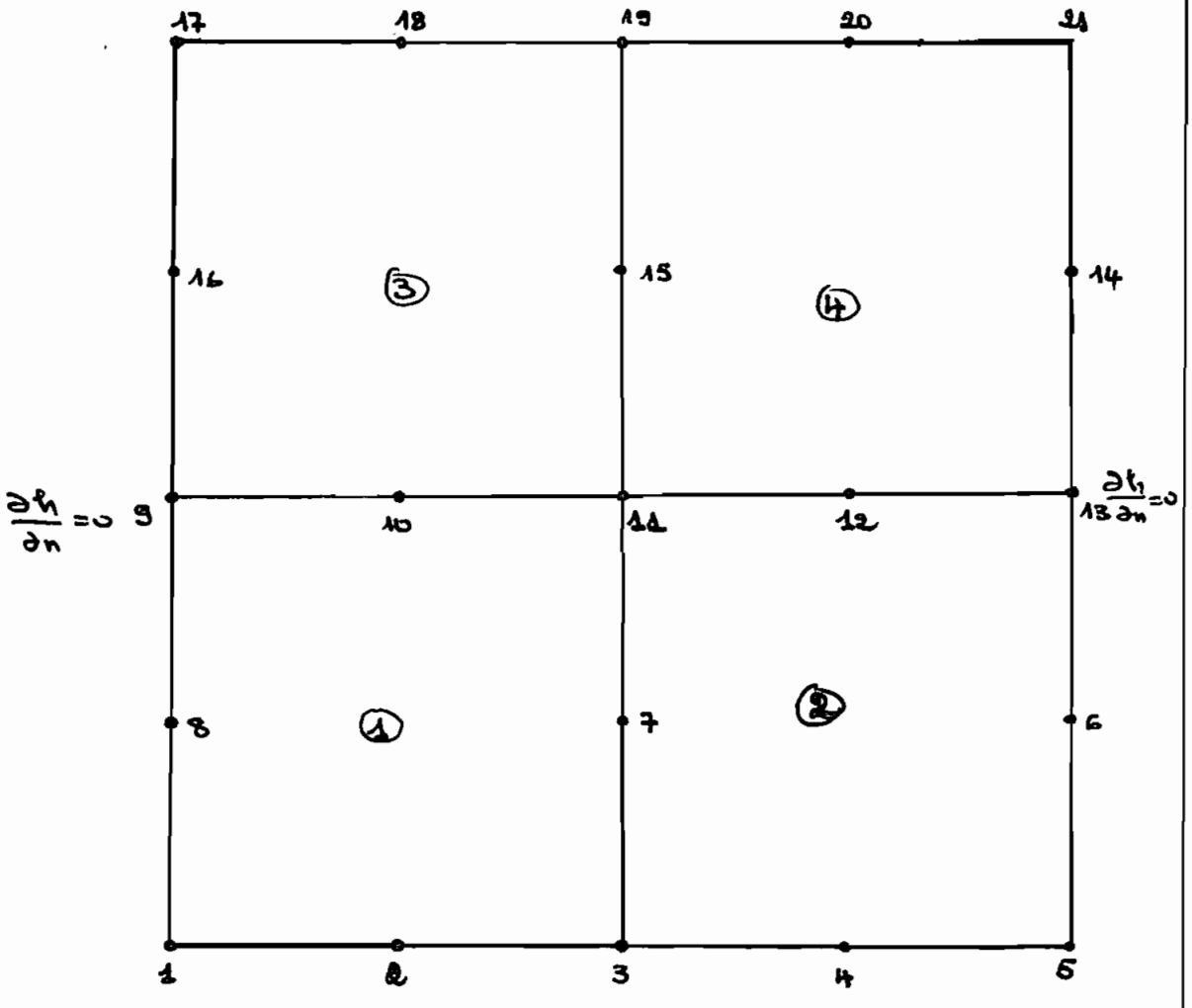
****

RESULTATS CALCULES

VARIABLE Nx	CHARGE CALCULEE (m)	DEBIT CALCULE (m3 / s)
1	10.000000	-0.08333
2	10.000000	-0.33333
3	10.000000	-0.16667
4	10.000000	-0.33333
5	10.000000	-0.08333
6	10.250000	0.00000
7	10.250000	0.00000
8	10.250001	0.00000
9	10.500000	0.00000
10	10.500000	0.00000
11	10.499999	0.00000
12	10.500000	0.00000
13	10.500000	0.00000
14	10.750000	0.00000
15	10.749999	0.00000
16	10.749999	0.00000
17	10.999998	0.08333
18	11.000000	0.33333
19	10.999999	0.16667
20	11.000000	0.33333
21	11.000000	0.08333

******* SOMME DES DEBITS ENTRANT : 9.999989271e-001
******* SOMME DES DEBITS SORTANT : -1.000000954e+000
******* SOMME DE TOUT LES DEBITS : -2.026557922e-006

III.



Elément 5 linéaire . 17 , 18 , 19.

Elément 6 linéaire 19 , 20 , 21.

Segment 17 - 21 Alimentation distribuée de $0{,}001\ m^3/_{sm}$

Noeuds 1, 2, 3, 4, 5 Charge imposée de 10. m.

$K = 1\ m/_s$.

***** NOMBRE TOTAL D'ELEMENT ********** 6
***** NOMBRE TOTAL DE NOEUD ********** 21
***** NOMBRE TOTAL DE PERMEABILITE ***** 2

** PERMRABILITE**

CLASSE	K1	K2	K3	K4	K5	K6
1	1.00000	0.00000	0.00000	1.00000	0.00000	0.00000
2	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000

*** CONNECTIVITE ***

ELT Nx	TYPE	CLASSE	ALD	ND1	ND2	ND3	ND4	ND5	ND6	ND7	ND8	ND9
1	48	1	0.000	1	2	3	7	11	10	9	8	
2	48	1	0.000	3	4	5	6	13	12	11	7	
3	48	1	0.000	9	10	11	15	19	18	17	16	
4	48	1	0.000	11	12	13	14	21	20	19	15	
5	23	2	0.001	17	18	19						
6	23	2	0.001	19	20	21						

*** COORDONNEES ET CONDITIONS AUX LIMITES ****

NOEUD Nx	XCOORD (m)	YCOORD (m)	ZCOORD (m)	Hhcx	Valh (m)	DEBIT (m3/s)
1	0.000	0.000	0.000	-1	10.000	0.0000
2	250.000	0.000	0.000	-1	10.000	0.0000
3	500.000	0.000	0.000	-1	10.000	0.0000
4	750.000	0.000	0.000	-1	10.000	0.0000
5	1000.000	0.000	0.000	-1	10.000	0.0000
6	1000.000	250.000	0.000	0	0.000	0.0000
7	500.000	250.000	0.000	0	0.000	0.0000
8	0.000	250.000	0.000	0	0.000	0.0000
9	0.000	500.000	0.000	0	0.000	0.0000
10	250.000	500.000	0.000	0	0.000	0.0000
11	500.000	500.000	0.000	0	0.000	0.0000
12	750.000	500.000	0.000	0	0.000	0.0000
13	1000.000	500.000	0.000	0	0.000	0.0000
14	1000.000	750.000	0.000	0	0.000	0.0000
15	500.000	750.000	0.000	0	0.000	0.0000
16	0.000	750.000	0.000	0	0.000	0.0000
17	0.000	1000.000	0.000	0	0.000	0.0000
18	250.000	1000.000	0.000	0	0.000	0.0000
19	500.000	1000.000	0.000	0	0.000	0.0000
20	750.000	1000.000	0.000	0	0.000	0.0000
21	1000.000	1000.000	0.000	0	0.000	0.0000

****
RESULTATS CALCULES

VARIABLE Nx	CHARGE CALCULEE (m)	DEBIT CALCULE (m3 / s)
1	10.000000	-0.08333
2	10.000000	-0.33333
3	10.000000	-0.16667
4	10.000000	-0.33334
5	10.000000	-0.08333
6	10.250001	0.00000
7	10.250001	0.00000
8	10.250001	0.00000
9	10.500000	0.00000
10	10.500002	0.00000
11	10.500002	0.00000
12	10.500002	0.00000
13	10.500001	0.00000
14	10.750002	0.00000
15	10.750002	0.00000
16	10.750001	0.00000
17	11.000001	0.08333
18	11.000001	0.33333
19	11.000002	0.16667
20	11.000002	0.33333
21	11.000002	0.08333

******* SOMME DES DEBITS ENTRANT : 9.999989271e-001
******* SOMME DES DEBITS SORTANT : -1.000002980e+000
******* SOMME DE TOUT LES DEBITS : -4.053115845e-006

CHAPITRE IX

CONCLUSION ET RECOMMANDATION

Les difficultés rencontrées lors de la conception de ce logiciel sont trés variées. La première, et l'une des plus importantes, a été la maitrise du langage C (Langage de programmation). A ce niveau les principales difficultés ont été la gestion des fichiers. Ce handicap a été finalement levé. Le second problème a résidé dans ce que nous pourrons appeler des faiblesses de la version 1.0 du langage C. Nous voulons parler de la mauvaise gestion des adresses et l'absence totale de message d'erreurs, en cas d'erreurs, lors de l'exécution du programme.

Mise à part ces faiblesses le langage c est un langage remarquable et très rapide dans l'exécution. En dépit de ces difficultés nous avons écrit un logiciel qui, vus les résultats satisfaisants obtenus, fonctionne très bien. Ce logiciel est capable de traiter les écoulements 1-D, 2-D et 3-D. La précision des calculs est acceptable. Par rapport à tout cela nous pourrons dire que les objectifs de ce projet de fin d'étude ont été dans la majeur parti atteints. Ce logiciel prépare, pour les écoulements 2-D et 1-D, un fichier que pourra utiliser le logiciel CONTOUR, si l'on désire visualiser les lignes équipotentielles (i.e la surface libre de la nappe).

RECOMMANDATION

Afin de rendre ce logiciel assez perfomant, un peu plus flexible et capable de traiter les écoulements transitoires, il est vivement souhaité que ce projet soit poursuivi dans une troisième phase. Cette troisième phase doit concerner les domaines suivants:

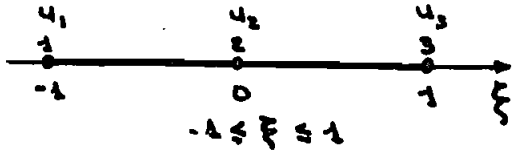
- -amélioration de la saisie des données;
- -la résolution des problèmes d'écoulement transitoire;
- -mutiplier les texts de vérification afin de vérifier toutes les facettes du logiciel et d'avoir un logiciel vraiment adapter aux divers types de problème qui risquent de se produire dans la réalité

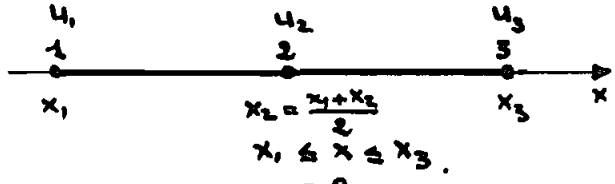
ANNEXES

Tableau des fonctions d'interpolations utilisées dans le logiciel.

1- Eléments quadratique à noeuds équidistants

(3 noeuds, C°) Type 23.



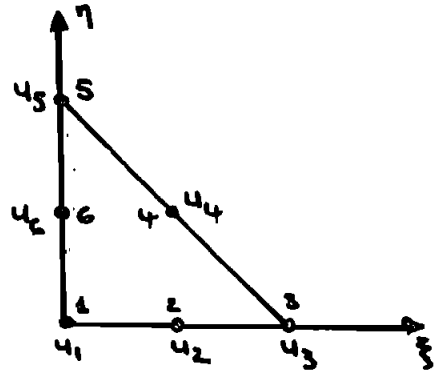


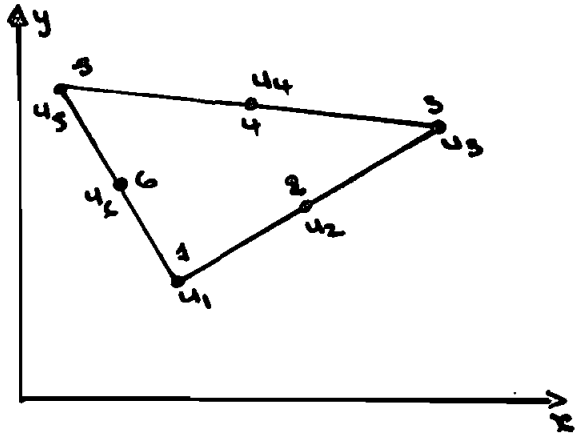
Ω^R $n = 3$ $n_d = 3$ Ω^e

i	Ni	$\partial N_i/\partial \xi$
1	$-\frac{1}{2}\xi(1-\xi)$	$\frac{1}{2}(-1+2\xi)$
2	$1-\xi^2$	-2ξ
3	$\frac{1}{2}\xi(1+\xi)$	$\frac{1}{2}(1+2\xi)$

2- Eléments quadratique (Triangle, 6 noeuds, C°)

Type 36





$n = 6$ $n_d = 6$

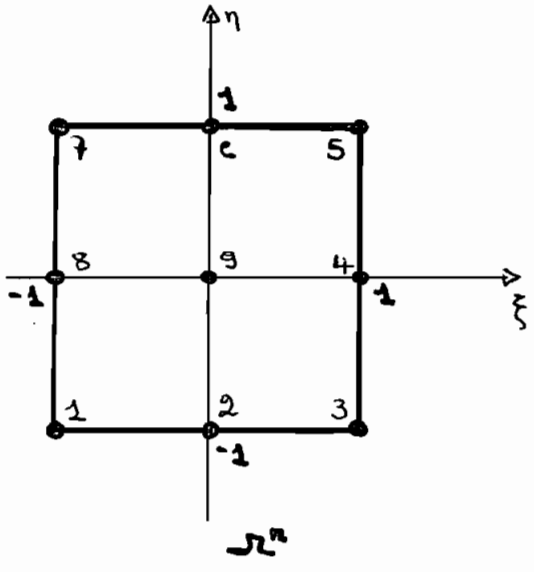
Ω^R Ω^e

Elément de référence. Elément réel.

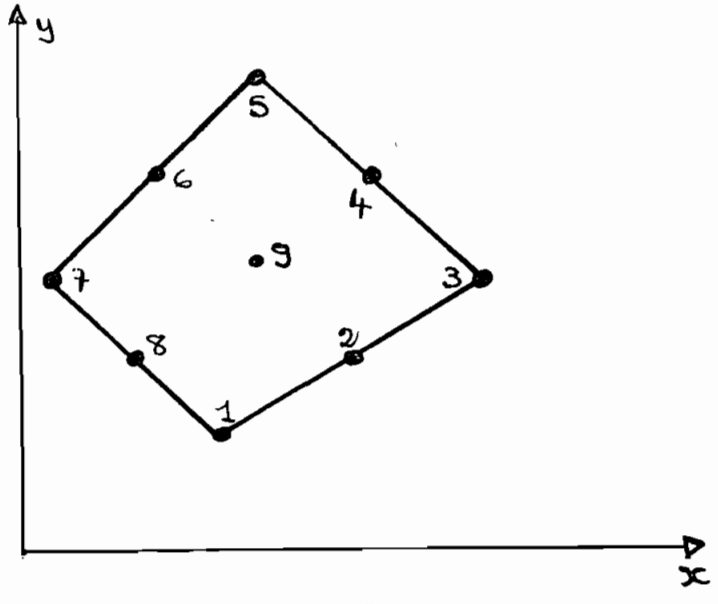
i	N_i	$\partial N_i/\partial \xi$	$\partial N_i/\partial \eta$
1	$-\lambda(1-2\lambda)$	$1-4\lambda$	$1-4\lambda$
2	$4\cdot\xi\cdot\lambda$	$4(\lambda-\xi)$	-4ξ
3	$-\xi(1-2\cdot\xi)$	$-1+4\xi$	0
4	$4\cdot\xi\cdot\eta$	4η	4ξ
5	$-\eta\cdot(1-2\cdot\eta)$	0	$-1+4\cdot\eta$
6	$4\eta\cdot\lambda$.	$-4\cdot\eta$	$4\cdot(\lambda-\eta)$

avec $\lambda = 1-\xi-\eta$

3- <u>Elément quadratique complet.</u> (quadrilatère, 9 noeuds, C^0). Type 49



$\mathcal{R}^n$



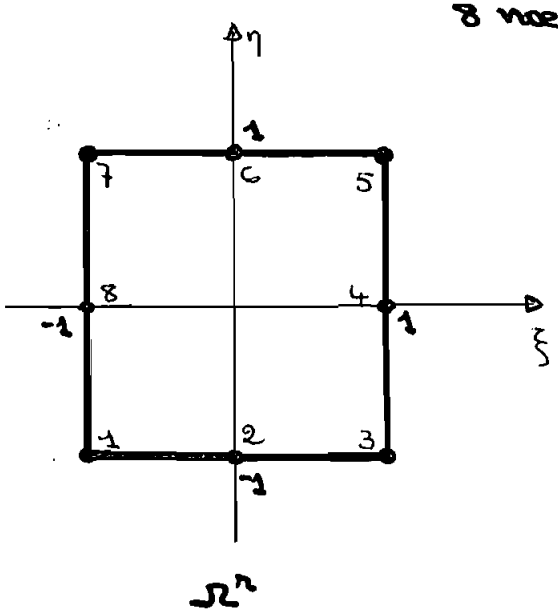
$\mathcal{R}^e$

$n = 9 \qquad n_d = 9.$

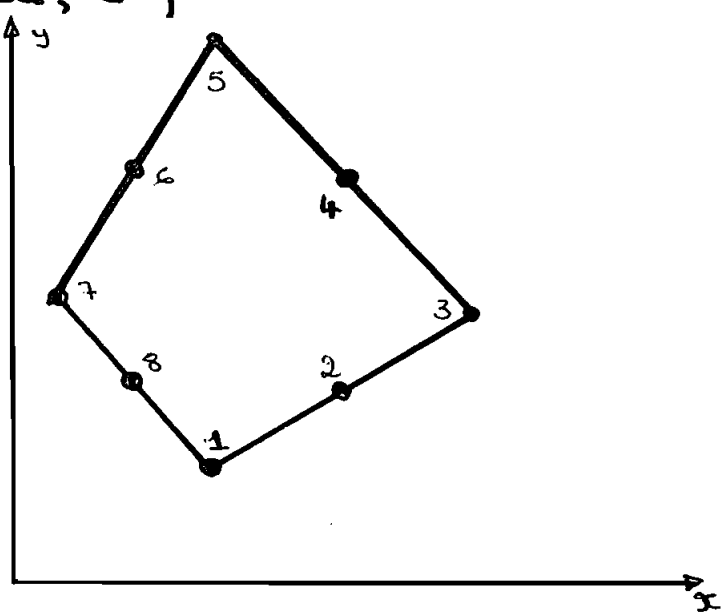
Elément de référence. Elément réel

i	Ni	$\partial N_i/\partial \xi$	$\partial N_i/\partial \eta$
1	$-1/4(1-\xi)(1-\eta)\cdot\xi\eta$	$-1/4\cdot(1-2\cdot\xi)(1-\eta)\cdot\eta$	$-1/4\cdot(1-\xi)(1-2\eta)\xi$
2	$-1/2\cdot(1-\xi^2)(1-\eta)\eta$	$(1-\eta)\cdot\xi\eta$	$-1/2(1-\xi^2)(1-2\eta)$
3	$-1/4\cdot(1+\xi)(1-\eta)\xi\eta$	$-1/4\cdot(1+2\xi)(1-\eta)\cdot\eta$	$-1/4(1+\xi)(1-2\eta)\xi$
4	$1/2\cdot(1+\xi)(1-\eta^2)\xi$	$1/2\cdot(1+2\xi)(1-\eta^2)$	$-(1+\xi)\xi\eta$
5	$1/4(1+\xi)(1+\eta)\cdot\xi\cdot\eta$	$1/4\cdot(1+2\cdot\xi)(1+\eta)\cdot\eta$	$1/4(1+\xi)(1+2\eta)\xi$
6	$1/2\cdot(1-\xi^2)(1+\eta)\eta$	$-(1+\eta)\xi\cdot\eta$	$1/2\cdot(1-\xi^2)(1+2\eta)$
7	$-1/4(1-\xi)(1+\eta)\xi\cdot\eta$	$-1/4\cdot(1-2\xi)(1+\eta)\eta$	$-1/4(1-\xi)(1+2\eta)\xi$
8	$-1/2(1-\xi)(1-\eta^2)\xi$	$-1/2(1-2\xi)(1-\eta^2)$	$(1-\xi)\xi\eta$
9	$(1-\xi^2)(1-\eta^2)$	$-2(1-\eta^2)\xi$	$-2(1-\xi^2)\eta$

4- Elément quadratique incomplet (quadrilatère, 8 noeuds, C°)



$\mathcal{R}^n$



$\mathcal{R}^e$

$n=8 \quad n_d=8$

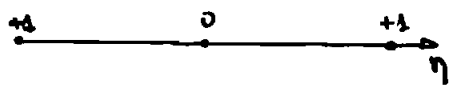
Elément de référence. Elément réel.

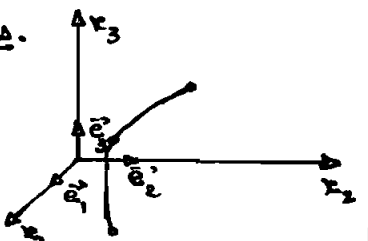
i	N_i	$\partial N_i/\partial \xi$	$\partial N_i/\partial \eta$
1	$-1/4(1-\xi)(1-\eta)(1+\xi+\eta)$	$-1/4(1-\eta)(2\cdot\xi+\eta)$	$-1/4\cdot(1-\xi)(\xi+2\cdot\eta)$
2	$1/2(1-\xi^2)(1-\eta)$	$-(1-\eta)\xi$	$-1/2\cdot(1-\xi^2)$
3	$-1/4\cdot(1+\xi)(1-\eta)(1-\xi+\eta)$	$-1/4(1-\eta)(2\xi-\eta)$	$-1/4(1+\xi)(\xi-2\eta)$
4	$1/2(1+\xi)(1-\eta^2)$	$-1/2(1-\eta^2)$	$-(1+\xi)\eta$
5	$-1/4(1+\xi)(1+\eta)(1-\xi-\eta)$	$-1/4(1+\eta)(2\cdot\xi+\eta)$	$-1/4\cdot(1+\xi)(\xi+2\cdot\eta)$
6	$1/2(1-\xi^2)(1+\eta)$	$-(1+\eta)\xi$	$1/2(1-\xi^2)$
7	$-1/4(1-\xi)(1+\eta)(1+\xi-\eta)$	$-1/4\cdot(1+\eta)(2\cdot\xi-\eta)$	$-1/4\cdot(1-\xi)(\xi-2\cdot\eta)$
8	$1/2\cdot(1-\xi)(1-\eta^2)$	$-1/2(1-\eta^2)$	$-(1-\xi)\eta$

Caractéristiques des éléments de référence.

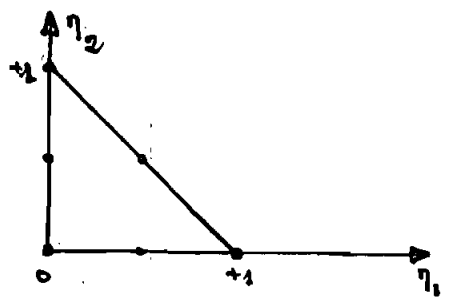
Eléments de référence. Eléments réel en espace 3-D

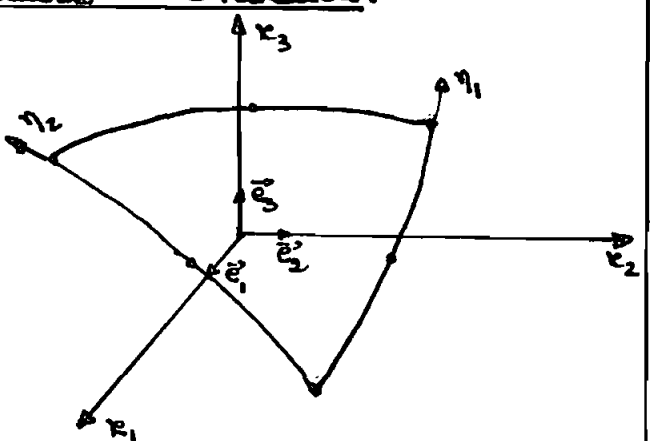
Segment à 3 noeuds.



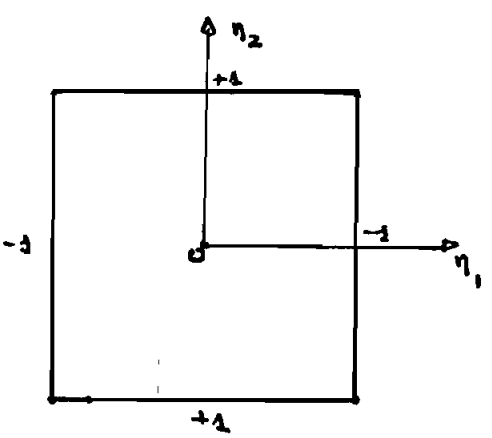


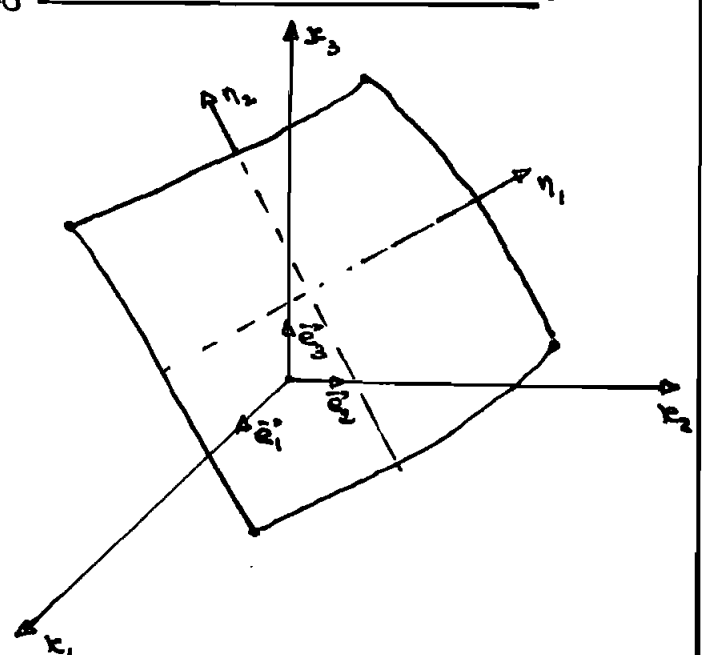
Elément triangulaire à 6 noeuds.





Elément rectangulaire à 8 ou 9 noeuds.

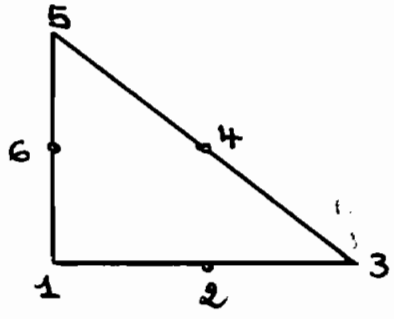




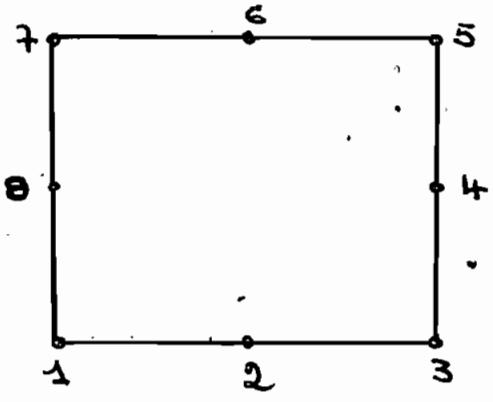
ORDRE DES NOEUDS DANS LES ELEMENTS.

TYPE 23

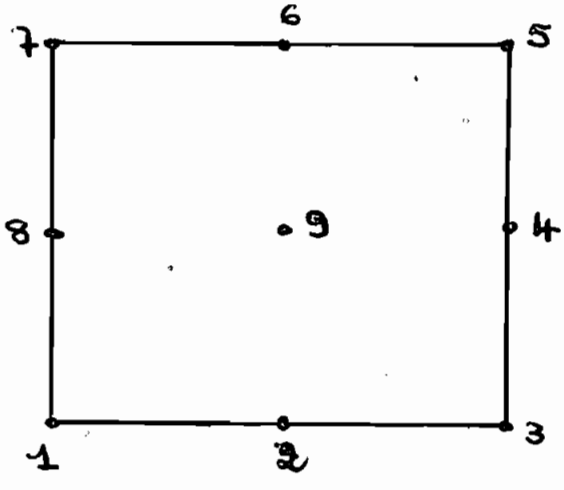
TYPE 36



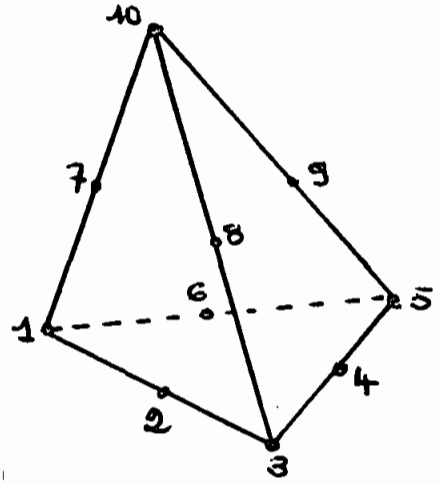
TYPE 48



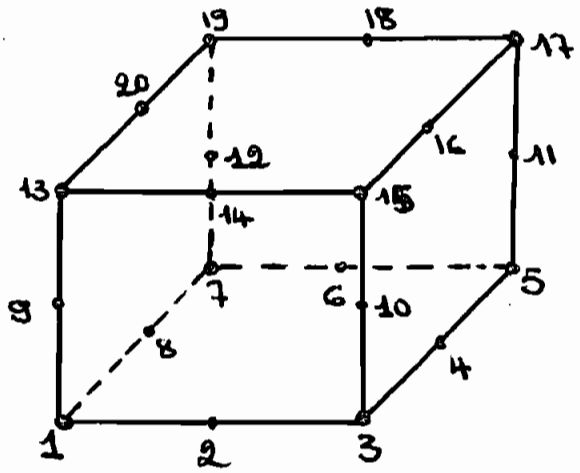
TYPE 49



TYPE 4 10



TYPE 8 20.



TYPE 8 27

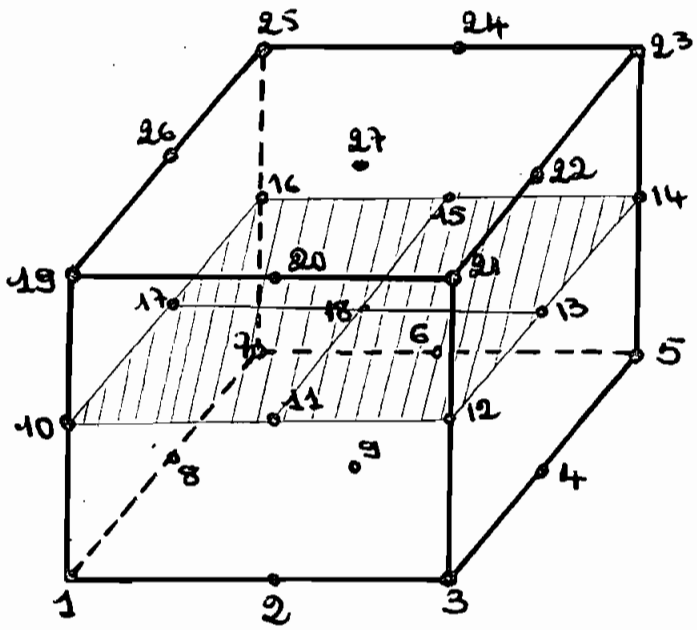


Tableau des Eléments et des points d'intégrations.

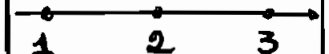
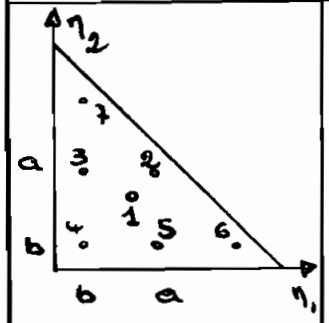
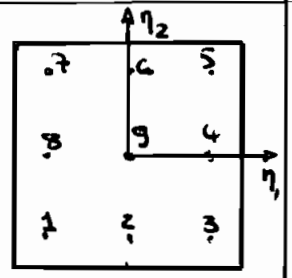
Elément	nombre de point d'intégration (n)	Coordonnées η_1	Coordonnées η_2	Coefficient de pondération w
	$n = 3$	$-\sqrt{3/5}$ $+0$ $+\sqrt{3/5}$		$5/9$ $8/9$ $5/9$
	$n = 7$ $a = \frac{6+\sqrt{18}}{21}$ $b = \frac{4}{7} - a$	$1/3$ a $1-2a$ a b $1-2b$ b	$1/3$ a a $1-2a$ b b $1-2b$	$9/80$ $A = \frac{155-\sqrt{15}}{2400}$ $\frac{31}{240} - A$
	$n = 9$ $a = -\sqrt{3/5}$ $b = 0$ $c = \sqrt{3/5}$	a b c c a b a b	a a a b c c c b b	$25/81$ $40/81$ $25/81$ $40/81$ $25/81$ $40/81$ $25/81$ $40/81$ $64/81$

Tableau des éléments et des points d'intégration : (suite)

Elément	Point d'intégration n	Coordonnées η_1	η_2	η_3	Coefficient de pondération ω
Elément de référence tétraédrique.	$n = 15$	a	a	a	$8/405$
	$a = 1/4$	b_i	b_i	b_i	$\left.\begin{matrix} w_1 \\ w_2 \end{matrix}\right\} = \dfrac{2\,665 \mp 14\sqrt{15}}{226.800}$
	$b_1 = \dfrac{7+\sqrt{15}}{34}$	b_i	b_i	c_i	
	$b_2 = \dfrac{7-\sqrt{15}}{34}$	b_i	c_i	b_i	
	$c_1 = \dfrac{13-3\sqrt{15}}{34}$	c_i	b_i	b_i	
	$c_2 = \dfrac{13+3\sqrt{15}}{34}$	d	d	e	$5/567$
	$d = \dfrac{5-\sqrt{15}}{20}$	d	e	d	
	$e = \dfrac{5+\sqrt{15}}{20}$	e	d	d	
		d	e	e	
		e	d	e	
		e	e	d	
Elément de référence Cubique.	$n = 27$	a	a	a	125/729
	$a = -\sqrt{3/5}$	a	a	b	200/729
	$b = 0$	a	a	e	125/729
	$e = \sqrt{3/5}$	a	b	a	200/729
		a	b	b	320/729
		a	b	e	200/729
		a	e	a	125/729
		a	e	b	200/729
		a	e	e	125/729
		b	a	a	200/729
		b	a	b	320/729
		b	a	e	200/729
		b	b	a	320/729
		b	b	b	512/729
		b	b	e	320/729
		b	e	a	200/729
		b	e	b	320/729
		b	e	e	200/729

Tableau des éléments et des points d'intégration (suite)

Elément	Point d'intégration r	Coordonnées η_1	η_2	η_3	Coefficient de pondération w.
Elément de référence Cubique (suite)	$r = 27$ $a = -\sqrt{3/5}$ $b = 0$ $c = \sqrt{3/5}$	c	a	a	125/729
		c	a	b	200/729
		c	a	c	125/729
		c	b	a	200/729
		c	b	b	320/729
		c	b	c	200/729
		c	c	a	125/729
		c	c	b	200/729
		c	c	c	125/729

Liste des notations.

ρ : masse volumique du fluide.

P : pression

θ : température

$\vec{V}$: vitesse de filtration du fluide.

m : porosité du milieu.

k : perméabilité

K : perméabilité intrinsèque

$\vec{g}$: accélération de la pesenteur.

μ : viscosité dynamique du fluide.

z : élevation au niveau du datum

$[K]$: tenseur de perméabilité

h : charge hydraulique (m)

t : temps

α : compressibilité du squelette solide.

β : compressibilité du fluide.

S : Coefficient d'emmagasinement specifique.

q : terme de source (Debit volumetrique du fluide)

Γ : Contour du Domaine

Ω : Domaine

R : Résidu.

L : Opérateur

φ : fonction de ponderation quelconque.

N_i : fonctions d'interpolation

$\tilde{N}_i$: fonctions de transformation géométrique.

τ^e : transformation géométrique.

$[G]^e$: matrice de rigidité élémentaire

$[f]^e$: Vecteur de sollicitation élémentaire

$[m]^e$ matrice de masse élémentaire

$\{h_n\}$ Vecteur élémentaire des variables nodales.

$[G]$ matrice de rigidité globale.

$[F]$ Vecteur de sollicitation globale.

$[M]$ matrice de masse globale.

$\{h\}$ Vecteur globale des variables nodales.

$[J]$ matrice Jacobienne.

g^{ik} : tenseur métrique contravariant.

g_{ik} : tenseur metrique covariant.

δ : poids volumique du fluide.

*****

NOMENCLATURE

Fe	Fonctions d'interpolation
A	Dérivées des fonctions d'interpolation par rapport aux variables (η_1, η_2, η_3) de l'espace de référence
nd	Nombre de noeuds de l'élément réel
nbnd	Nombre de noeuds total du domaine
nbelt	Nombre total d'éléments du domaine
H	Charge à un noeud donné du domaine
Q	Débit à un noeud donné du domaine
Id	Tableau contenant le nombre d'apparution des différents noeuds du domaine
Ndi	Tableau des noeuds d'un élément donné
VEK	Vecteur qui à un noeud associe une place dans la matrice de travail
VEKin	Vecteur qui à une position donné de la matrice de travail associe un noeud
hc	Vecteur élémentaire des conditions aux limites
Hhcx	Vecteur des conditions aux limites des noeuds activés
MTR	Matrice de travail résident en mémoire
Gek	Matrice de rigidité élémentaire
FSG	Vecteur de sollicitation des noeuds activés
FSe	Vecteur de sollicitation élémentaire
Max	Taille maximale de la :argeur de front
classe	Classe de perméabilité élémentaire
Ald	Alimentation ditribuer sur l'élément
Ty	Le type de l'élément de référence

lon	Dimension de matrice de travail
gik	Matrice des tenseurs contravariant
E	Matrice des dérivées de coordonées dans l'espace réel par rapport aux variables (η_1 , η_2 , η_3)
det	Déterminant de la matrice des tenseurs covariant
B	Matrice des dérivées des fonction d'interpolation par rapport aux variables (x , y , z)
X, Y, Z	Vecteur des coordonnées des noeuds de l'élément réel
Ke	Tenseur de perméabilité
n1, n2, n3	Points d'intégration
wi	Coefficient de pondération

FICHIERS

Clasper	Fichier des perméabilités
Loce	Fichier des connectivités
Corg	Fichier des coordonnées des noeuds
EFRONT	Fichier des équations éliminés
RESUL	Ficihier des résultats intermediaire claiculé lors de la saisie des données
RESULTAT	Fichier des résultats

```
/*  *-------------------------------------------------------------
    *                        REPUBLIQUE DU SENEGAL
    *
    *                   ECOLE POLYTECHNIQUE DE THIES
    *
    *                   DEPARTEMENT GENIE CIVIL
    *
    *                  PROJET DE FIN D'ETUDE
    *
    *       TITRE:    CONCEPTION D'UN LOGICIEL POUR SIMULER
    *                   PAR LA METHODE DES ELEMENTS FINIS
    *                    LES ECOULEMENTS EN MILIEU POREUX
    *                       SATURE
    *
    *
    *       AUTEUR:    KODJOVI MAWUMETO DEGUE
    *
    *
    *       DIRECTEUR: GERARD ANDRE ROBERT SOUMA
    *       Professeur d'hydrogéologie
    *
    *    CO-DIRECTEUR: AMADOU SARR
    *     Professeur d'hydraulique
    *
    *************************************************************************
*/                  /* EMPSAT*/
#include <ctype.h>
#include <stdio.h>
#include <dos.h>
#include <process.h>
#include <fonccom.c>
main()
{
int n=0, stat;
char ch;
```

```
gotoxy(2,5);
puts("                                                                    ");
gotoxy(3,5);
puts("                                                                    ");
gotoxy(4,5);
puts("                          REPUBLIQUE DU SENEGAL                     ");
gotoxy(5,5);
puts("                                                                    ");
gotoxy(6,5);
puts("                       ECOLE POLYTECHNIQUE DE THIES                 ");
gotoxy(7,5);
puts("                                                                    ");
gotoxy(8,5);
puts("                        DEPARTEMENT GENIE CIVIL                     ");
gotoxy(9,5);
puts("                                                                    ");
gotoxy(10,5);
puts("                     PROJET DE FIN D'ETUDE                          ");
gotoxy(11,5);
puts("                                                                    ");
gotoxy(12,5);
puts("          TITRE:    CONCEPTION D'UN LOGICIEL POUR SIMULER           ");
gotoxy(13,5);
puts("                    PAR LA METHODE DES ELEMENTS FINIS LES           ");
gotoxy(14,5);
puts("                    ECOULEMENTS EN MILIEU POREUX SATURE             ");
gotoxy(15,5);
puts("                                                                    ");
gotoxy(16,5);
puts("          AUTEUR:    KODJOVI MAWUMETO DEGUE                        ");
gotoxy(17,5);
puts("                                                                    ");
gotoxy(18,5);
puts("       DIRECTEUR:  GERARD ANDRE ROBERT SOUMA                        ");
gotoxy(19,5);
puts("                     Professeur d'hydrogéologie                     ");
gotoxy(20,5);
puts("                                                                    ");
gotoxy(21,5);
puts("    CO-DIRECTEUR:  AMADOU SARR                                      ");
gotoxy(22,5);
puts("                     Professeur d'hydraulique                       ");
gotoxy(23,5);
puts("                                                                    ");

 gotoxy(24,20);
 puts("  PESER SUR UNE TOUCHE POUR CONTINUER "); getch();
 do {
gotoxy(24,25); printf("\n\n\n\n\n");
cls();
```

```
gotoxy(5,15);
puts("                                                              ");
gotoxy(6,15);
puts("                                                              ");
gotoxy(7,15);
puts("                      MENU                                    ");
gotoxy(8,15);
puts("                                                              ");
gotoxy(9,15);
puts("        < S >    SAISIE DES DONNEES                           ");
gotoxy(10,15);
puts("                                                              ");
gotoxy(11,15);
puts("        < R >  RESOLUTION (TRAITEMENT DES DONNEES)            ");
gotoxy(12,15);
puts("                                                              ");
gotoxy(13,15);
puts("        < I >  IMPRESSION DES RESULTATS ET DES                ");
gotoxy(14,15);
puts("                       DONNEES                                ");
gotoxy(16,15);
puts("        < E >     SORTIR DU PROGRAMME                         ");
gotoxy(15,15);
puts("                                                              ");
gotoxy(17,15);
puts("                                                              ");
gotoxy(18,15);
puts("                                                              ");
gotoxy(19,15);
puts("                                                              ");
gotoxy(20,15);
puts("          *****  VOTRE CHOIX ******                          ");
gotoxy(21,15);
puts("                                                              ");
gotoxy(22,15);
puts("                                                              ");
gotoxy(23,15);
puts("                                                              ");
gotoxy(20,55); ch=getch();
ch=toupper(ch);
while(ch!='S' && ch!='R' && ch!='I' && ch!='E')
     {
     printf("\a"); gotoxy(18,55); ch=getch();
     ch=toupper(ch);
     }
```

```
gotoxy(24,20); cls();
if(ch=='S')
  {
   n=1;
     system("SAISI");
       continue;
   }
 else if(ch=='R')
    {
         if(n!=1) continue;
             n=2;
             system("RESOL");
    }
  else if(ch=='I')
    {
         if(n!=2) continue;
               system("SORTI");
    }
  } while(ch!='E');
```

```
      /*******************      FONCCOM.C   **********************/

void gotoxy(int x,int y)
                /*Positionne le curseur à la position indiqué */
  {
    union REGS reg;

    reg.h.ah = 2;
    reg.x.dx = (x << 8) + y;
    reg.h.bh = 0;
    int86(0x10, &reg, &reg);
    }
void cls(void)  /* efface l'écran */
   {
   printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
   printf("\n\n\n\n\n\n\n\n\n\n\n\n\n\n");
   gotoxy(1,1);
    }
```

```
            /************   FNOM.C   ****************/


int delete_file(char near *name)
  {
      union REGS regs; struct SREGS sregs;
      int ret;

       regs.h.ah=0x41;
       regs.x.dx=(unsigned) name;
       sregs.ds=_DS;
       ret=intdosx(&regs,&regs,&sregs);
       return(regs.x.cflag ? ret:0);
```

```
 /*  ************ RESOL.C ******************
        ***********************************
        *      Projet de fin d'études:      *
        ***********************************
 DIRECTEUR : GERARD SOUMA

 CO-DIRECTEUR: AMADOU SARR

 AUTEUR : DEGUE KODJOVI MAWUMETO

 TITRE :      Conception d'un logiciel pour simuler par
           la méthode des éléments finis les écoulements
            en milieu poreux saturée
    */

void Assemblage(double Gek[27][27],double FSe[27],int VEK[230],int ntd,
                double MTR[60][60],double FSG[67],int Ndi[27]);
int AssemElimStock(int Id[230],float H[230],float Q[230],int nbet);
void conver(float C[][],int nbl,int nbc,float B[][]);
void ElnonstGauss(double MTR[][],int k,int lon,float ht,double FSG[]);
void ElstGauss(double MTR[][],int k,int la,double FSG[]);
void foncIcode(int Ndi[],int Id[],int la,int VEK[],int Icode[]);
void Inidouble(double Wi[],int nbi);
void Inifloat(float Wi[],int nbi);
void IniInt(int Wi[],int nbi);
float matcovar3d(float A[][],float X[],float Y[],float Z[],int nd,
                 float B[][]);
float matcovar2d(float A[][],float X[],float Y[],float Z[],int nd,
                 float B[][]);
int ModifMTR(double MTR[][],double FSG[],int VEK[],int VEKin[],
                 int k,int la,int Hhcx[],int Icode[]);
int nbnoeud(int Type);
void prodmat(float A[][],float A1[][],int nblA1,int nblB1,
                 int nbcB1,float C[][]);
int RedimMTR(int NDi[],int VEe[],int ntd,int lon,int VEK[],
                 int VEKin[],int Hhcx[],int hc[]);
void Sommat(float C[][],double Gek[][],int nd);
void subtinv(float Q[],float H[],int *nbd,int *MaX);
void transpo(float C[][],int nbl,int nbc,float A[][]);
void Type23(int classe,float Ald,int Ndi[],double Gek[][],double FSe[],
                 int Hhcx[],float Q[],int Id[]);
void Type36(int classe,float Ald,int Ndi[],double Gek[][],double FSe[],
                 int Hhcx[],float Q[],int Id[]);
void Type48(int classe,float Ald,int Ndi[],double Gek[][],double FSe[],
                 int Hhcx[],float Q[],int ID[]);
void Type49(int classe,float Ald,int Ndi[],double Gek[][],double FSe[],
                 int Hhcx[],float Q[],int Id[]);
void Type410(int classe,float Ald,int Ndi[],double Gek[][],double FSe[],
                 int Hhcx[],float Q[],int Id[]);
void Type620(int classe,float Ald,int Ndi[],double Gek[][],double FSe[],
                 int Hhcx[],float Q[],int Id[]);
void Type627(int classe,float Ald,int Ndi[],double Gek[][],double FSe[],
                 int Hhcx[],float Q[],int Id[]);
```

```
    struct NDS {
            int Num;
            int Ty;
            int classe;
            float Ald;
            unsigned int Ndi[27];
            };
     struct SUB {
             int no;
             float Coef[60];
             int Noeud[60];
             float fg,co;
             int Hhcx,lon;
               } ;

#include <math.h>
#include <fcntl.h>
#include <stdlib.h>
#include <stdio.h>
#include <dos.h>
#include <io.h>
#include "A:PTYPE49.C"
#include "A:PTYPE48.C"
#include "A:PTYPE36.C"
#include "A:FNOM.C"
 main()
   {
 int nd=0,MaX=0;
 int i,j,Id[230];
 int nbnd,nbet=0;
 float H[230],Q[230];
 int fd,fd5;
    FILE *f,*f5;

 Inifloat(Q,230);Inifloat(H,230);IniInt(Id,230);
    fd=open("A:RESUL",O_CREAT|O_TEXT,0x0100|0x0080);
    f=fdopen(fd,"rt");
     fscanf(f,"%d %d %d",&nbnd,&nbet,&nd);

       for(i=0; i<nbnd; i++)
          fscanf(f,"%d %f %f",&Id[i],&H[i],&Q[i]);
          fclose(f);
                  MaX=AssemElimStock(Id,H,Q,nbet);
        subtinv(Q,H,&nbnd,&MaX);
        puts("  \n ****** FIN  D'EXECUTION  ******");
        delete_file("a:RESULTAT");
        fd5=open("A:RESULTAT",O_CREAT|O_TEXT,0x0100|0x0080);
        f5=fdopen(fd5,"r+t");
        for(i=1; i<=nbnd; i++)
        fprintf(f5,"%d  %f  %f\n",i,H[i-1],Q[i-1]);
        fclose(f5);
        }
          /* Fin de la fonction principale */
```

```
void Assemblage(double Gek[27][27], double FSe[27], int VEK[230], int ntd,
                    double MTR[60][60], double FSG[67], int Ndi[27])
       {
/* Assemblage de la nouvelle Matrice Gek à la Matrice de travail MTR */

       int i, j, n=0, d=0;

          for(i=0; i<ntd; i++) {
            for(j=0; j<ntd; j++) {
           n=0;
           n=VEK[Ndi[i]-1]-1;
           d=0;
           d=VEK[Ndi[j]-1]-1;
            MTR[n][d]+=Gek[i][j];
            }
      FSG[n]+=FSe[i];
         }
       }
int AssemElimStock(int Id[230], float H[230], float Q[230], int nbet)
   {
 int lon=0, MaX=0, Ndi[27], nd=0;
 int i, j, k, VEK[230], VEKin[60], hc[27];
 int Icode[60], Hhcx[60];
double MTR[60][60], FSG[60], Gek[27][27], FSe[27];
 struct NDS xt;
 struct SUB yts;
 int fd4, fd2;
 FILE *f4, *f2;
    IniInt(VEK, 230); IniInt(VEKin, 60); IniInt(Icode, 60);
    IniInt(Hhcx, 60); Inidouble(MTR, 60*60);
    Inidouble(FSG, 60);

   delete_file("c:EFRONT");
   fd4=open("C:EFRONT", O_TRUNC|O_CREAT|O_BINARY, 0x0100|0x0080);
   f4=fdopen(fd4, "r+");
             IniInt(Ndi, 27);

   fd2=open("A:LOCE", O_BINARY, 0x0100|0x0080);
   f2=fdopen(fd2, "rb");

      puts("    DEBUT DE L'EXECUTION ");
      rewind(f2);
      for(i=1; i<=nbet; i++) {
    IniInt(hc, 27); IniInt(Ndi, 27); Inidouble(Gek, 27*27); Inidouble(FSe, 27);
      xt.Num=0; xt.Ty=0; xt.classe=0; xt.Ald=0;
      for(j=0; j<27; j++) xt.Ndi[j]=0;
       /* Lecture dans le fichier LOCE */
       printf("  \r élément N°%d d'une serie de %d ", i, nbet);
       fread(&xt, sizeof(xt), 1, f2);
      nd=nbnoeud(xt.Ty);
        for(j=0; j<nd; j++) {
             Ndi[j]=xt.Ndi[j];
                  }
```

```
      if(nd==3)  Type23(xt.classe,xt.Ald,Ndi,Gek,FSe,hc,Q,Id);
      else if(nd==6)  Type36(xt.classe,xt.Ald,Ndi,Gek,FSe,hc,Q,Id);
      else if(nd==8)  Type48(xt.classe,xt.Ald,Ndi,Gek,FSe,hc,Q,Id);
      else if(nd==9)  Type49(xt.classe,xt.Ald,Ndi,Gek,FSe,hc,Q,Id);
      else if(nd==10) Type410(xt.classe,xt.Ald,Ndi,Gek,FSe,hc,Q,Id);
      else if(nd==20) Type620(xt.classe,xt.Ald,Ndi,Gek,FSe,hc,Q,Id);
      else  Type627(xt.classe,xt.Ald,Ndi,Gek,FSe,hc,Q,Id);

      lon=RedimMTR(Id,Ndi,nd,lon,VEK,VEKin,Hhcx,hc);
      if(MaX<lon) MaX=lon;
      Assemblage(Gek,FSe,VEK,nd,MTR,FSG,Ndi);
      foncIcode(Ndi,Id,nd,VEK,Icode);
      for(j=0; j<lon; j++) {
      if(Icode[j]==0 || Icode[j]==1) {
      /*Enregistrement dans un fichier EFRONT du numéro du noeud
des coefficients dans la matrice MTR du numéro des noeuds composants
       de la valeur de la charge H du Debit et de Hhcx */

     yts.no=0;
     yts.no=VEKin[j];
     yts.co=0;
     yts.co=MTR[j][j];
     yts.fg=0;
     yts.fg=FSG[j];
     yts.Hhcx=0;
     yts.Hhcx=Hhcx[j];
     yts.lon=0;
     yts.lon=lon;
     Inifloat(yts.Coef,60);  IniInt(yts.Noeud,60);

              for(k=0; k<lon; k++) {
                  yts.Coef[k]=MTR[j][k];
                  yts.Noeud[k]=VEKin[k];
                           }
      fwrite(&yts,sizeof(yts),1,f4);

      if(Hhcx[j]==-1)
        ElnonstGauss(MTR,j,lon,H[VEKin[j]-1],FSG);
     else    ElstGauss(MTR,j,lon,FSG);
       lon=ModifMTR(MTR,FSG,VEK,VEKin,j,lon,Hhcx,Icode);
       j--;
            }
        }
        }
        fclose(f2);
        fclose(f4);
   return(MaX);
   }
```

```
void conver(float Ci[27][27], int nbl, int nbc, float Bi[27][27])
   {
      int i,j;
         for(i=0; i<=(nbl-1); i++) {
           for(j=0; j<=(nbc-1); j++) {
                Bi[i][j]=0.0;
                  Bi[i][j]=Ci[i][j];
                      Ci[i][j]=0.0;
                }
            }
    }
void ElstGauss(double MTR[60][60], int k, int lon, double FSG[60])
             /* Elimination standard de GAUSS */
    {
     int i,j;
     double b, ai, c;
     ai=MTR[k][k];
     if(ai==0.0) {puts("** division par zero**"); exit(1); }
      for(i=0; i<lon; i++)  {
      if(i!=k) {
      b=0; b=(MTR[i][k]/ai);

       for(j=0; j<lon; j++) {
        c=0;
        c=b*MTR[k][j];
         MTR[i][j]=MTR[i][j]-c;
                }
         c=0;
         c=b*FSG[k];
        FSG[i]=FSG[i]-c;
         }
      }
    }
void ElnonstGauss(double MTR[60][60], int k, int lon, float ht, double FSG[60])
      {
      /* Elimination non standard de Gauss  */

       int i,j;
       double b;
        for(i=0; i<lon; i++) {
          if(i!=k) {
           b=0;
           b=MTR[i][k]*ht;
            FSG[i]=FSG[i]-b;
            }
        }
       }
```

```
void foncIcode(int Ndi[27], int Id[230], int la, int VEK[230], int Icode[60])
    /* Calcul du code d'apparution des variables */
       {
        int i, n=0;

           for(i=0; i<la; i++) {
       if(Id[Ndi[i]-1]>1) {
              n=VEK[Ndi[i]-1]-1;
                  Icode[n]=Id[Ndi[i]-1];
                  n=-Id[Ndi[i]-1];
                  Id[Ndi[i]-1]=-Id[Ndi[i]-1]+1;
                }
       else  if(Id[Ndi[i]-1]==1) {
            n=VEK[Ndi[i]-1]-1;
                  Icode[n]=1;
             }
       else  if(Id[Ndi[i]-1]==-1) {
               n=VEK[Ndi[i]-1]-1;
                  Icode[n]=0;
            }
       else   if(Id[Ndi[i]-1]<-1) {
             n=VEK[Ndi[i]-1]-1;
                  Icode[n]=-1;
                  Id[Ndi[i]-1]++;
              }
            }
       }
void Inifloat(float Wi[], int nbi)
  {
   int i;
      for(i=0; i<nbi; i++)
            Wi[i]=0.0;
  }
void IniInt(int Wi[], int nbi)
  {
   int i;
      for(i=0; i<nbi; i++)
            Wi[i]=0;
  }
void Inidouble(double Wi[], int nbi)
  {
     int i;
     for(i=0; i<nbi; i++)
     Wi[i]=0;
  }
```

```
float matcovar3d(float A[27][27],float X[27],float Y[27],float Z[27],
                 int nd,float B[27][27])
  {
  float E[3][3],gik[3][3],a1=0.0,a2=0.0,a3=0.0,a4=0.0,a5=0.0,a6=0.0;
  float C[27][27];
  double det;
  int i,j,k;

 det=0.0;
 Inifloat(E,3*3); Inifloat(gik,3*3); Inifloat(C,27*27);

     for(i=0; i<=(nd-1); i++) {
          E[0][0]+=A[i][0]*X[i];
          E[1][0]+=A[i][0]*Y[i];
          E[2][0]+=A[i][0]*Z[i];
          E[0][1]+=A[i][1]*X[i];
          E[1][1]+=A[i][1]*Y[i];
          E[2][1]+=A[i][1]*Z[i];
          E[0][2]+=A[i][2]*X[i];
          E[1][2]+=A[i][2]*Y[i];
          E[2][2]+=A[i][2]*Z[i];
    }
    a1=(E[0][0]*E[0][0]);
    a1+=(E[1][0]*E[1][0]);
    a1+=(E[2][0]*E[2][0]);
    a2=(E[0][0]*E[0][1]);
    a2+=(E[1][0]*E[1][1]);
    a2+=(E[2][0]*E[2][1]);
    a3=(E[0][0]*E[0][2]);
    a3+=(E[1][0]*E[1][2]);
    a3+=(E[2][0]*E[2][2]);
    a4=(E[0][1]*E[0][1]);
    a4+=(E[1][1]*E[1][1]);
    a4+=(E[2][1]*E[2][1]);
    a5=(E[0][1]*E[0][2]);
    a5+=(E[1][1]*E[1][2]);
    a5+=(E[2][1]*E[2][2]);
    a6=(E[0][2]*E[0][2]);
    a6+=(E[1][2]*E[1][2]);
    a6+=(E[2][2]*E[2][2]);

       det=(a1*(a4*a6-a5*a5));
       det+=(a2*(a5*a3-a2*a6));
       det+=(a3*(a2*a5-a4*a3));

            gik[0][0]=(a4*a6-a5*a5)/det;
            gik[1][0]=gik[0][1]=(a5*a3-a2*a6)/det;
            gik[2][0]=gik[0][2]=(a2*a5-a4*a3)/det;
            gik[1][1]=(a1*a6-a3*a3)/det;
            gik[1][2]=gik[2][1]=(a2*a3-a1*a5)/det;
            gik[2][2]=(a1*a4-a2*a2)/det;
```

```
          for(i=0; i<3; i++) {
              for(j=0; j<nd; j++) {
                      C[i][j]=0;
                       for(k=0; k<3; k++)
                        C[i][j]+=gik[i][k]*A[j][k];
                    }
              }
    for(i=0; i<3; i++) {
              for(j=0; j<nd; j++) {
                      B[i][j]=0;
                       for(k=0; k<3; k++)
                        B[i][j]+=E[i][k]*C[k][j];
                    }
              }

   a1=0;
  a1=det; return(a1);
  }

 float matcovar2d(float A1[27][27],float X[9],float Y[9],float Z[9],int nd,
                       float B[27][27])
   {
   int i,j;
   float E1=0,E2=0,E3=0,E4=0,E5=0,E6=0,gik[2][2];
   float n1=0,n2=0,n3=0,n4=0,n5=0,n6=0;
   double det;
   det=0;
      for(i=0; i<nd; i++)
          {

              E1+=(A1[i][0]*X[i]);
              E2+=(A1[i][1]*X[i]);
              E3+=(A1[i][0]*Y[i]);
              E4+=(A1[i][1]*Y[i]);
              E5+=(A1[i][0]*Z[i]);
              E6+=(A1[i][1]*Z[i]);
          }
     n1=E1*E1; n2=E2*E2; n3=E3*E3;
     n4=E4*E4; n5=E5*E5; n6=E6*E6;

det=(n1+n3+n5);
det=det*(n2+n4+n6);
n1=n2=n3=0;
n1=E1*E2; n2=E3*E4; n3=E5*E6;n4=0;
n4=n1+n2+n3;
det=det-n4*n4;
Inifloat(gik,2*2);
if(det==0) {puts("*****Division par zero******"); exit(1); }
n1=0; n1=E2*E2+E4*E4+E6*E6;
gik[0][0]=n1/det;
n2=0; n2=E1*E2+E3*E4+E5*E6;
gik[0][1]=-n2/det;
gik[1][0]=-n2/det;
```

```
n3=0;  n3=E1*E1+E3*E3+E5*E5;
gik[1][1]=n3/det;
 for(i=0;  i<nd;  i++)
    {
   n1=0;  n1=A1[i][0]*gik[0][0];
   n2=0;  n2=A1[i][1]*gik[1][0];
                  B[0][i]=E1*(n1+n2);
  n3=0;  n3=A1[i][0]*gik[0][1];
  n4=0;  n4=A1[i][1]*gik[1][1];
                  B[0][i]+=E2*(n3+n4);
  n5=0;  n5=A1[i][1]*gik[1][0];
                  B[1][i]=E3*(n1+n5);
  n6=0;  n6=A1[i][1]*gik[1][1];
                  B[1][i]+=E4*(n3+n6);
                  B[2][i]=E5*(n1+n2);
                  B[2][i]+=E6*(n3+n6);
   }
            n1=0;   n1=det;
      return(n1);
    }
int ModifMTR(double MTR[60][60],double FSG[60], int VEK[230], int VEKin[60],
                    int k, int la, int Hhcx[60], int Icode[60])
       {
    /* k est la ligne qu'on vient d'éliminer */
    /* VEKin[] Vecteur qui a une position donnée de MTR associe un noeud
       VEK     Vecteur qui à un noeud associe une position dans MTR */
            int i,j;
      for(i=k;  i<=(la-1);  i++) {

       VEKin[i]=VEKin[i+1];
       Hhcx[i]=Hhcx[i+1];
       Icode[i]=Icode[i+1];
       VEK[VEKin[i]-1]=i+1;
       FSG[i]=FSG[i+1];
           }
          FSG[la-1]=0;
     for(i=k;  i<(la-1);  i++) {
         for(j=0;  j<la;  j++)  {
         MTR[i][j]=MTR[i+1][j];
         }
      }
     for(i=0;  i<la;  i++) {
         for(j=k;  j<(la-1);  j++)  {
         MTR[i][j]=MTR[i][j+1];
         }
      }
     for(i=0;  i<la;  i++) {
         MTR[la-1][i]=0;    MTR[i][la-1]=0;
          }
       la--;
       return(la);
   }
```

```
int nbnoeud(int Type)
   {
      int nd=0;

      if(Type==23)  nd=3;
      else if(Type==36)  nd=6;
      else if(Type==48)  nd=8;
      else if(Type==49)  nd=9;
      else if(Type==410) nd=10;
      else if(Type==620) nd=20;
      else nd=27;
      return (nd);
   }
void prodmat(float A1[27][27],float B1[27][27],int nblA1,int nblB1,
               int nbcB1,float C1[27][27])
     {
     int i=0,j=0,k=0;
     float a=0;
       for(i=0; i<=nblA1-1; i++) {
          for(j=0; j<=nbcB1-1; j++)   {
              C1[i][j]=0.0;
               for(k=0; k<=(nblB1-1); k++) {
                         a=0.0;  a=A1[i][k]*B1[k][j];
                             C1[i][j]=C1[i][j]+a;
                             }
               }
          }
     }
int RedimMTR(int Id[230],int VEe[27],int ntd,int lon,int VEK[230],
                     int VEKin[60],int Hhcx[60],int hc[27])
      {
/*
Crée une place dans la matrice de travail MTR pour
       les nouvelles variables
*/

       int i,la;
       la=lon;
        for(i=1; i<=ntd; i++) {
        if(Id[VEe[i-1]-1]>0) {
         la++;
         VEK[VEe[i-1]-1]=la;
         Hhcx[la-1]=hc[i-1];
         VEKin[la-1]=VEe[i-1];
         }
       }
       return(la);
     }
```

```
void Sommat(float Ci[27][27],double Gek[27][27],int nd)
        {
        int i,j;
          for(i=0; i<nd; i++) {
            for(j=0; j<nd; j++)
                    Gek[i][j]+=Ci[i][j];
           }
        }
        /* Processus d'assemblage et d'élimination de Gauss */

void subtinv(float Q[230],float H[230],int *nbd,int *MaX)
 {
 /* Ce sous programme fait la resolution du système par la méthode de la
    la substitution inverse. Cela consiste à lire dans le fichier RES_EL en
    partant de la fin jusqu'au début du  fichier. */

    float TablCoef[60][230],TCoef[2][230];
    int i,j,fd4,nbnd,n=0;
    double xi,som;
    struct SUB yts;
    int Tabl[3][230],Tablnoeud[60][230];
    FILE *f4;
    nbnd=*nbd; n=*MaX;
    fd4=open("C:EFRONT",O_BINARY,0x0100|0x0080);
    f4=fdopen(fd4,"rb");
    rewind(f4);
    IniInt(Tabl,3*nbnd); IniInt(Tablnoeud,n*nbnd);
    Inifloat(TablCoef,n*nbnd); Inifloat(TCoef,2*nbnd);
    for(j=nbnd-1; j>=0; j--) {

    yts.no=0; yts.co=0; yts.Hhcx=0; yts.lon=0; yts.fg=0;
    IniInt(yts.Noeud,60); Inifloat(yts.Coef,60);

     fread(&yts,sizeof(yts),1,f4);

   Tabl[0][j]=yts.no;  Tabl[1][j]=yts.Hhcx; Tabl[2][j]=yts.lon;
   TCoef[0][j]=yts.fg; TCoef[1][j]=yts.co;

   for(i=0; i<yts.lon; i++) {

      TablCoef[i][j]=yts.Coef[i];
      Tablnoeud[i][j]=yts.Noeud[i];
           }
               }
                 fclose(f4);
     for(i=0; i<nbnd; i++)  {
     som=0;
    if(Tabl[1][i]==0) {
            for(j=0; j<Tabl[2][i]; j++) {
              if(Tablnoeud[j][i]!=Tabl[0][i]) {
                    n=Tablnoeud[j][i]-1;
                    som+=(TablCoef[j][i]*H[n]); }
                  }
```

```
          xi=0; xi=TCoef[0][i]-som;
               H[Tabl[0][i]-1]+=xi/TCoef[i][i];
        }
      else  if(Tabl[i][i]==-1) {
                      for(j=0; j<Tabl[2][i]; j++) {
                        som+=TablCoef[j][i]*H[Tablnoeud[j][i]-1]; }
                        Q[Tabl[0][i]-1]+=(som-TCoef[0][i]);

                }
            }
        }
void transpo(float C1[27][27], int nbl, int nbc, float A1[27][27])
   {
   int i,j;
          for(i=0; i<nbl; i++) {
             for(j=0; j<nbc; j++) {
               A1[j][i]=0.0;
               A1[j][i]=C1[i][j];
          }
      }
   }
void Type23(int classe,float Ald, int Ndi[27],double Gek[27][27],
                double FSe[27], int Hhcx[27],float Q[230], int Id[230])

/* Elèment linéaire à 3 noeuds */
    {
    float X[3],Y[3],Z[3],gik=0,Ke[27][27];
    float A[27][27],B[27][27],n1=0,w1=0,E1=0,x=0,y=0;
    float E2=0,E3=0,z=0,v=0,d=0;
    float debit[3],K[6]={0,0,0,0,0,0};
    int i,j,h=0,n=0;
    int fd3,fd;
    double det;
    float C[27][27];
    FILE *f,*f3;
    fd3=open("A:CORG",O_TEXT);
    f3=fdopen(fd3,"rt");
     fd=open("A:clasper",O_TEXT);
        f=fdopen(fd,"rt");
        do {
        fscanf(f,"%d %f %f %f %f %f %f",&i,&K[0],&K[1],&K[2],&K[3],&K[4],&K[5]);
         } while(classe!=i);
    Inifloat(X,3); Inifloat(Y,3); Inifloat(Z,3);
    Inifloat(Ke,27*27); Inifloat(debit,3);
    Inifloat(A,27*27); Inifloat(B,27*27); Inifloat(C,27*27);
    do {
    fscanf(f3,"%d %f %f %f %d %f %f \n",&j,&x,&y,&z,&h,&v,&d);
    for (i=0; i<=2; i++) {
       if(Ndi[i]==j) {
       X[i]=x; Y[i]=y; Z[i]=z; Hhcx[i]=h; debit[i]=d; n++;
       }
       }
       } while(n!=3);
```

```
  fclose(f);
  fclose(f3);
    i=0;
  for(i=1; i<=3; i++) {
     E1=0; E2=0; E3=0;   x=5;y=8;z=9;
    if(i==1) { n1=-sqrt(0.6); w1=x/z; }
    if(i==2) { n1=0.0; w1=y/z; }
    if(i==3) { n1=sqrt(0.6); w1=x/z; }
      A[0][0]=0.5*(-1+2*n1);
      A[1][0]=-2*n1;
      A[2][0]=0.5*(1+2*n1);
        for(j=0; j<3; j++) {
         x=0;   x=A[j][0]*X[j];
         E1+=x;
          y=0;  y=A[j][0]*Y[j];
         E2+=y;
         z=0;  z=A[j][0]*Z[j];
         E3+=z;
         }
      x=E1*E1;  y=E2*E2;  z=E3*E3;
         det=x+y+z;
         if(det==0.0) {printf("division par zero"); exit(1); }
         gik=1/det;

         for(j=1; j<=3; ++j) {

         B[0][j-1]=E1*A[j-1][0]*gik;
         B[1][j-1]=E2*A[j-1][0]*gik;
         B[2][j-1]=E3*A[j-1][0]*gik;

           }
   x=sqrt(det);y=w1*K[0];
Ke[0][0]=y*x;z=w1*K[1];
Ke[0][1]=Ke[1][0]=z*x;  z=0;  z=w1*K[3];
Ke[1][1]=z*x;  z=0;  z=w1*K[2];
Ke[2][0]=Ke[0][2]=z*x;  z=0;  z=w1*K[4];
Ke[1][2]=Ke[2][1]=z*x;  z=0;  z=w1*K[5];
Ke[2][2]=z*x;
      prodmat(Ke, B, 3, 3, 3, C);
      conver(C, 3, 3, B);
      prodmat(A, B, 3, 3, 3, C);
      Sommat(C, Gek, 3);
       /* Calcul du vecteur sollicitation élèmentaire */
          if(Ald!=0.0) {
          FSe[0]+=-0.5*n1*(1-n1)*w1*Ald*sqrt(det);
          FSe[1]+=(1-n1*n1)*w1*Ald*sqrt(det);
          FSe[2]+=0.5*n1*(1+n1)*w1*Ald*sqrt(det);
          }
       }
        for(i=0; i<=2; i++){
             Q[Ndi[i]-1]+=FSe[i];
             if(Id[Ndi[i]-1]>0) FSe[i]=FSe[i]+debit[i];}
  }
```

```
void Type36(int classe,float Ald, int Ndi[27],double Gek[27][27],
                double FSe[27], int Hhcx[27],float Q[230], int Id[230])
          /* triangle à 6 noeuds */
  {
  /* Lecture  des données relatif à l'élèment de numéro num */

  float X[9],Y[9],Z[9],Fe[6];
  float A[27][27],B[27][27],Ke[27][27];
  float wi=0,K[6]={0,0,0,0,0,0};
  float debit[6],x=0,y=0,z=0,v=0,d=0;
  int i,j,fd3,fd,h=0,n=0;
  double det;
  FILE *f,*f3;
  float C[27][27];
  fd3=open("A:CORG",O_TEXT);
      f3=fdopen(fd3,"rt");
       fd=open("A:clasper",O_TEXT);
        f=fdopen(fd,"rt");
        do {
        fscanf(f,"%d %f %f %f %f %f %f",&i,&K[0],&K[1],&K[2],&K[3],&K[4],&K[5]);
         } while(classe!=i);
   Inifloat(X,9); Inifloat(Y,9); Inifloat(Z,9);
Inifloat(Ke,27*27); Inifloat(debit,6);
    do {
    fscanf(f3,"%d %f %f %f %d %f %f \n",&j,&x,&y,&z,&h,&v,&d);
    for (i=0; i<=5; i++) {
       if(Ndi[i]==j) {
       X[i]=x; Y[i]=y; Z[i]=z; Hhcx[i]=h; debit[i]=d; n++;
       }
       }
       } while(n!=6);
       fclose(f);
          fclose(f3);

for(i=0; i<=6; i++) {
det=0.0; Inifloat(A,27*27); Inifloat(Fe,6);
Inifloat(B,27*27); Inifloat(C,27*27);
  wi=PRTYPE36(i,A,Fe);
   x=matcovar2d(A,X,Y,Z,6,B);
     det=x;
        x=sqrt(det); y=wi*K[0];
     Ke[0][0]=y*x; z=wi*K[1];
     Ke[0][1]=Ke[1][0]=z*x; z=0; z=wi*K[3];
     Ke[1][1]=z*x; z=0; z=wi*K[2];
     Ke[2][0]=Ke[0][2]=z*x; z=0; z=wi*K[4];
     Ke[1][2]=Ke[2][1]=z*x; z=0; z=wi*K[5];
     Ke[2][2]=z*x;


    transpo(B,3,6,A);
    prodmat(Ke,B,3,3,6,C);
    conver(C,3,6,B);
```

```
    prodmat(A, B, 6, 3, 6, C);
    Sommat(C, Gek, 6);
    det=x;
     /* Calcul du vecteur de Sollicitation élèmentaire */
         if(Ald!=0.0) {
         for(j=0; j<=5; j++)
            FSe[j]+=Fe[j]*wi*Ald*sqrt(det);
           }
       }         /* Fin de la boucle sur i */

                 for(i=0; i<=5; i++) {
                    Q[Ndi[i]-1]+=FSe[i];
                    if(Id[Ndi[i]-1]>0) FSe[i]=FSe[i]+debit[i]; }
    }
     /* fin du traitement sur l'élément du type 36 */

void Type48(int classe,float Ald, int Ndi[27],double Gek[27][27],
              double FSe[27], int Hhcx[27],float Q[230], int Id[230])
   {
   /* Lecture des donnée relatives à cet élément */

   float X[9],Y[9],Z[9],Fe[8];
   float A[27][27],B[27][27],Ke[27][27];
   float wi=0,x=0,y=0,z=0,d=0,v=0;
   float debit[8],K[6]={0,0,0,0,0,0};
   int i,j,h=0,n=0;
   double det;
   float C[27][27];
   int fd3,fd;
   FILE *f,*f3;
    fd3=open("A:CORG",O_TEXT);
      f3=fdopen(fd3,"rt");
       fd=open("A:clasper",O_TEXT);
        f=fdopen(fd,"rt");
        do {
    fscanf(f,"%d %f %f %f %f %f %f",&i,&K[0],&K[1],&K[2],&K[3],&K[4],&K[5]);
         } while(classe!=i);
      Inifloat(X,9); Inifloat(Y,9); Inifloat(Z,9); Inifloat(Ke,27*27);
      Inifloat(debit,8);
    do {
    fscanf(f3,"%d %f %f %f %d %f %f \n",&j,&x,&y,&z,&h,&v,&d);
    for (i=0; i<=7; i++) {
       if(Ndi[i]==j) {
       X[i]=x; Y[i]=y; Z[i]=z; Hhcx[i]=h; debit[i]=d; n++;
       }
       }
       } while(n!=8);
       fclose(f);
         fclose(f3);

     for(i=1; i<=9; i++) {
   det=0.0; Inifloat(B,27*27); Inifloat(C,27*27);
   Inifloat(Ke,27*27); Inifloat(A,27*27); Inifloat(Fe,8);
```

```
   wi=PRTYPE48(i,A,Fe);
 x=matcovar2d(A,X,Y,Z,8,B);
   det=x;
        x=sqrt(det);y=wi*K[0];
     Ke[0][0]=y*x;z=wi*K[1];
     Ke[0][1]=Ke[1][0]=z*x;  z=0;  z=wi*K[3];
     Ke[1][1]=z*x;  z=0;  z=wi*K[2];
     Ke[2][0]=Ke[0][2]=z*x;  z=0;  z=wi*K[4];
     Ke[1][2]=Ke[2][1]=z*x;  z=0;  z=wi*K[5];
     Ke[2][2]=z*x;

  transpo(B,3,8,A);
  prodmat(Ke,B,3,3,8,C);
  conver(C,3,8,B);
  prodmat(A,B,8,3,8,C);
  Sommat(C,Gek,8);
  det=x;
 /* Calcul du vecteur sollicitation élèmentaire */
   if(Ald!=0.0)  {
   for(j=0; j<=7; j++)
      FSe[j]+=Fe[j]*wi*Ald*sqrt(det);

     }
      }

          for(i=0; i<=7; i++) {
            Q[Ndi[i]-1]+=FSe[i];
            if(Id[Ndi[i]-1]>0)  FSe[i]=FSe[i]+debit[i]; }
  }
void Type49(int classe,float Ald,int Ndi[27],double Gek[27][27],double FSe[27],in
   {
   float X[9],Y[9],Z[9];
   float A[27][27];
   float B[27][27],Ke[27][27];
   float wi=0.0,Fe[9],K[6]={0,0,0,0,0,0};
   float debit[9],x=0,y=0,z=0,v=0,d=0;
   int i,j,fd3,fd,h=0,n=0;
   float C[27][27];
   double det;
   FILE *f3,*f;
   fd3=open("A:CORG",O_TEXT);
   f3=fdopen(fd3,"rt");
     fd=open("A:clasper",O_TEXT);
        f=fdopen(fd,"rt");
        do {
        fscanf(f,"%d %f %f %f %f %f %f",&i,&K[0],&K[1],&K[2],&K[3],&K[4],&K[5]);
         } while(classe!=i);
Inifloat(X,9); Inifloat(Y,9); Inifloat(Z,9);
Inifloat(Ke,27*27); Inifloat(debit,9);
```

```
  do {
  fscanf(f3,"%d %f %f %f %d %f %f \n",&j,&x,&y,&z,&h,&v,&d);
  for (i=0; i<=8; i++) {
     if(Ndi[i]==j) {
     X[i]=x; Y[i]=y; Z[i]=z; Hhcx[i]=h; debit[i]=d; n++;
     }
     }

     } while(n!=9);
      fclose(f);
      fclose(f3);

  for(i=1; i<=9; i++) {
det=0.0; x=0; Inifloat(A,27*27); Inifloat(Fe,9); Inifloat(B,27*27);
Inifloat(C,27*27);
  w1=PRTYPE49(i,A,Fe);
 x=matcovar2d(A,X,Y,Z,9,B);
     det=x;
      x=sqrt(det);y=w1*K[0];
   Ke[0][0]=y*x;z=w1*K[1];
   Ke[0][1]=Ke[1][0]=z*x; z=0; z=w1*K[3];
   Ke[1][1]=z*x; z=0; z=w1*K[2];
   Ke[2][0]=Ke[0][2]=z*x; z=0; z=w1*K[4];
   Ke[1][2]=Ke[2][1]=z*x; z=0; z=w1*K[5];
   Ke[2][2]=z*x;

  transpo(B,3,9,A);
  prodmat(Ke,B,3,3,9,C);
  conver(C,3,9,B);
  prodmat(A,B,9,3,9,C);
  Sommat(C,Gek,9);
  /* Calcul du vecteur sollicitation élèmentaire */
  if(Ald!=0.0) {
  for(j=0; j<=8; j++)
    FSe[j]+=Fe[j]*w1*Ald*x;

          }
        }

            for(i=0; i<=8; i++) {
                  Q[Ndi[i]-1]+=FSe[i];

                  if(Id[Ndi[i]-1]>0) FSe[i]=FSe[i]+debit[i]; }
   }
   /* Calcul des éléments tridimensionnels */
```

```
void Type410(int classe,float Ald,int Ndi[27],double Gek[27][27],
                double FSe[27],int Hhcx[27],float Q[230],int Id[230])
  {
  float X[27],Y[27],Z[27],Fe[10];
  float A[27][27],wi=0.0,bt=0,K[6]={0,0,0,0,0,0};
  float Ke[27][27],x=0,y=0,z=0,v=0,d=0;
  int i,fd3,fd5,fd,j,h=0,n=0;
  FILE *f3,*f5,*f;
  float B[27][27],C[27][27];
  double det;
  float debit[10];
     fd3=open("A:CORG",O_TEXT);
     f3=fdopen(fd3,"rt");
     fd5=open("A:DATA410",O_TEXT);
     f5=fdopen(fd5,"rt");
     fd=open("A:clasper",O_TEXT);
     f=fdopen(fd,"rt");
       do {
    fscanf(f,"%d %f %f %f %f %f %f",&i,&K[0],&K[1],&K[2],&K[3],&K[4],&K[5]);
        } while(classe!=i);
  Inifloat(X,27); Inifloat(Y,27); Inifloat(Z,27);
   Inifloat(Ke,27*27); Inifloat(debit,10);
   do {
   fscanf(f3,"%d %f %f %f %d %f %f \n",&j,&x,&y,&z,&h,&v,&d);
   for (i=0; i<=9; i++) {
      if(Ndi[i]==j) {
      X[i]=x; Y[i]=y; Z[i]=z; Hhcx[i]=h; debit[i]=d; n++;
      }
      }
      } while(n!=10);
      fclose(f);
     fclose(f3);


  for(i=0; i<=14; i++) {
  det=0.0; Inifloat(A,27*27); Inifloat(Fe,10);
   Inifloat(B,27*27); Inifloat(C,27*27);
    for(j=0; j<=9; j++)
       fscanf(f5,"%e %e %e %e %e",&A[j][0],&A[j][1],&A[j][2],&Fe[j],&wi);

  x=matcovar3d(A,X,Y,Z,10,B);
       det=x;
       x=sqrt(det); y=wi*K[0];
    Ke[0][0]=y*x; z=wi*K[1];
    Ke[0][1]=Ke[1][0]=z*x; z=0; z=wi*K[3];
    Ke[1][1]=z*x; z=0; z=wi*K[2];
    Ke[2][0]=Ke[0][2]=z*x; z=0; z=wi*K[4];
    Ke[1][2]=Ke[2][1]=z*x; z=0; z=wi*K[5];
    Ke[2][2]=z*x;

  transpo(B,3,10,A);
  prodmat(Ke,B,3,3,10,C);
  conver(C,3,10,B);
```

```
   prodmat(A, B, 10, 3, 10, C);
   Sommat(C, Gek, 9);

if(Ald!=0.0) {
   bt=Ald*sqrt(det);
   for(j=0; j<=9; j++)
   FSe[j]+=Fe[j]*wi*bt;

        }
      }
      fclose(f5);
          for(i=0; i<=9; i++) {
                Q[Ndi[i]-1]+=FSe[i];
                     if(Id[Ndi[i]-1]>0) FSe[i]=FSe[i]+debit[i]; }
   }
void Type620(int classe,float Ald, int Ndi[27],double Gek[27][27],
                double FSe[27], int Hhcx[27],float Q[230], int Id[230])
   {
      float X[27], Y[27], Z[27], Fe[20];
      float A[27][27], B[27][27], Ke[27][27];
      float wi=0, bt=0.0, K[6]={0, 0, 0, 0, 0, 0};
      float x=0, y=0, z=0, d=0, v=0;
      int i, fd3, fd, fd5, j, h=0, n=0;
      FILE *f3, *f, *f5;
      double det;
      float C[27][27], debit[20];
            fd3=open("A:CORG", O_TEXT);
     f3=fdopen(fd3, "rt");
     fd=open("A:clasper", O_TEXT);
     f=fdopen(fd, "rt");
     fd5=open("A:DATA620", O_TEXT, 0x0100);
     f5=fdopen(fd5, "rt");
        do {
     fscanf(f, "%d %f %f %f %f %f %f", &i, &K[0], &K[1], &K[2], &K[3], &K[4], &K[5]);
        } while(classe!=i);
     Inifloat(X, 27); Inifloat(Y, 27); Inifloat(Z, 27);
     Inifloat(Ke, 27*27); Inifloat(debit, 20);
    do {
    fscanf(f3, "%d %f %f %f %d %f %f \n", &j, &x, &y, &z, &h, &v, &d);
    for (i=0; i<=19; i++) {
       if(Ndi[i]==j) {
       X[i]=x; Y[i]=y; Z[i]=z; Hhcx[i]=h; debit[i]=d; n++;
       }
       }
       } while(n!=20);
       fclose(f);
        fclose(f3);

          for(i=0; i<=26; i++) {
     det=0.0; Inifloat(A, 27*27); Inifloat(Fe, 20);
     Inifloat(B, 27*27); Inifloat(C, 27*27);
        for(j=0; j<20; j++)
      fscanf(f5, "%e %e %e %e %e", &A[j][0], &A[j][1], &A[j][2], &Fe[j], &wi);
```

```
      x=matcovar3d(A, X, Y, Z, 20, B);
           det=x;
      x=sqrt(det); y=w1*K[0];
    Ke[0][0]=y*x; z=w1*K[1];
    Ke[0][1]=Ke[1][0]=z*x;  z=0;  z=w1*K[3];
    Ke[1][1]=z*x;  z=0;  z=w1*K[2];
    Ke[2][0]=Ke[0][2]=z*x;  z=0;  z=w1*K[4];
    Ke[1][2]=Ke[2][1]=z*x;  z=0;  z=w1*K[5];
    Ke[2][2]=z*x;

        transpo(B, 3, 20, A);
        prodmat(Ke, B, 3, 3, 20, C);
        conver(C, 3, 20, B);
        prodmat(A, B, 20, 3, 20, C);
        Sommat(C, Gek, 20);
           if(Ald!=0.0) {
                bt=Ald*sqrt(det);
                for(j=0;  j<=19;  j++)
                 FSe[j]+=Fe[j]*w1*bt;

          }

        }
        fclose(f5);
     for(i=0;  i<=19;  i++) {
       Q[Ndi[i]-1]+=FSe[i];
                    if(Id[Ndi[i]-1]>0) FSe[i]=FSe[i]+debit[i]; }
}
void Type627(int classe, float Ald, int Ndi[27], double Gek[27][27], double FSe[27],
  {
  float X[27], Y[27], Z[27], Ke[27][27];
  float A[27][27], B[27][27], Fe[27];
  float w1=0, bt=0, x=0, y=0, z=0, d=0, v=0;
  double det;
  float C[27][27], debit[27], K[6]={0, 0, 0, 0, 0, 0};
  int i, fd3, fd5, fd, j, h=0, n=0;
  FILE *f, *f3, *f5;
   fd3=open("A:CORG", O_TEXT);
   f3=fdopen(fd3, "rt");
    fd5=open("A:DATA627", O_TEXT);
    f5=fdopen(fd5, "rt");
       fd=open("A:clasper", O_TEXT);
       f=fdopen(fd, "rt");
       do {
       fscanf(f, "%d %f %f %f %f %f %f", &i, &K[0], &K[1], &K[2], &K[3], &K[4], &K[5]);
        } while(classe!=i);
   Inifloat(X, 27);  Inifloat(Y, 27);  Inifloat(Z, 27);
   Inifloat(Ke, 27*27);  Inifloat(debit, 27);
```

```
do {
fscanf(f3,"%d %f %f %f %d %f %f",&j,&x,&y,&z,&h,&v,&d);
for (i=0; i<=26; i++) {
    if(Ndi[i]==j) {
    X[i]=x; Y[i]=y; Z[i]=z; Hhcx[i]=h; debit[i]=d; n++;
    }
    }
    } while(n!=27);
    fclose(f);
  fclose(f3);

    for(i=0; i<=26; i++) {
det=0.0; Inifloat(A,3*27); Inifloat(Fe,27);
Inifloat(B,27*3); Inifloat(C,27*27);

     for(j=0; j<=26; j++)
         fscanf(f5,"%e %e %e %e %e",&A[j][0],&A[j][1],&A[j][2],&Fe[j],&wi);

        x=matcovar3d(A,X,Y,Z,27,B);
             det=x;
    x=sqrt(det); y=wi*K[0];
 Ke[0][0]=y*x; z=wi*K[1];
 Ke[0][1]=Ke[1][0]=z*x; z=0; z=wi*K[3];
 Ke[1][1]=z*x; z=0; z=wi*K[2];
 Ke[2][0]=Ke[0][2]=z*x; z=0; z=wi*K[4];
 Ke[1][2]=Ke[2][1]=z*x; z=0; z=wi*K[5];
 Ke[2][2]=z*x;

         transpo(B,3,27,A);
         prodmat(Ke,B,3,3,27,C);
         conver(C,3,27,B);
         prodmat(A,B,27,3,27,C);
         Sommat(C,Gek,27);
         det=x;
              if(Ald!=0.0) {
              bt=Ald*sqrt(det);
              for(j=0; j<=26; j++)
               FSe[j]+=Fe[j]*bt*wi;
               }
         }
         fclose(f5);
         for(i=0; i<=26; i++) {
         Q[Ndi[i]-1]+=FSe[i];
                   if(Id[Ndi[i]-1]>0) FSe[i]=FSe[i]+debit[i]; }
}
```

```
   /*  ************ SAISI.C ******************
          ***********************************
          *      Projet de fin d'étude:      *
          ***********************************
  DIRECTEUR : GERARD SOUMA

  CO-DIRECTEUR: AMADOU SARR

  TITRE :      Conception d'un logiciel pour simuler par
           la méthode des élèments finis les écoulements
           en milieu poreux saturée
 */
#include <fcntl.h>
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include "A:fonccom.c"
#include "A:FNOM.c"
     /*  **** Declaration des fonctions *****  */

 void calcul_Q_H(int ncx,float Dt,float h,int n,float Qd[],float Hd[]);
 void cls(void);
 char controle_entree_donnees(int *nbnd,int *nc,char *option,char kh);
 int delete_file(char near *name);
 int element(int nc,int nb,int Ide[]);
 void Exist_noeud(int nbnd,float Hd[230],float Qd[230]);
 void foncId(int ntd,int Ndi[],int Ide[]);
 void gotoxy(int x,int y);
 int  nbnoeud(int Type);
 int  noeud(float He[],float Qe[]);
 int permeabilite(void);
 void setcursor( int xt,int yt);

    struct NDS {
            int Num;
            unsigned int Ty;
            int classe;
            float Ald;
            unsigned int Ndi[27];
                 } ;
```

```
main()
   {
   int Ide[230], fd, i, nc=0, nbnd=0, nbet=0;
   float Hd[230], Qd[230];
   char ch, *name, kh;
   FILE *f;
   setcursor(0, 12);
        name="PERMEABILITES"; kh='P';
        ch=controle_entree_donnees(&nbnd, &nc, name, kh);
  if(ch=='m') nc=permeabilite();
          name="COORDONNEES"; kh='C';
          ch=controle_entree_donnees(&nbnd, &nc, name, kh);
  if(ch=='m') nbnd=noeud(Hd, Qd);
       else Exist_noeud(nbnd, Hd, Qd);
     nbet=element(nc, nbnd, Ide);
   setcursor(0, 2);

   delete_file("a:RESUL");
   fd=open("A:RESUL", O_CREAT|O_TRUNC|O_TEXT, 0x0100|0x0080);
   f=fdopen(fd, "r+t");

      fprintf(f, "%d %d %d\n", nbnd, nbet, nc);
      for(i=0; i<nbnd; i++)
           fprintf(f, "%d %f %f\n", Ide[i], Hd[i], Qd[i]);
           fclose(f);
    }
    /* Fin de la fonction principale */

int permeabilite(void) /* saisi des perméabilités */
    {
    int fd1=0, ncper=0, i=0, j=0, n=1;
    char op;
    float K[6];
    FILE *f1;
    do {
    n=1;
     delete_file("A:clasper");
    fd1=open("a:clasper", O_TRUNC|O_CREAT|O_TEXT, 0x0100|0x0080);
    f1=fdopen(fd1, "r+t");

    cls(); gotoxy(10, 5);
    puts("Donner le nombre total de classe de perméabilité ...");
  gotoxy(10, 61); scanf("%d", &ncper); cls();
  gotoxy(6, 10);
puts("La matrice de perméabilité est une matrice symétrique");
 gotoxy(8, 10);
     puts("la saisi se fait seulement sur la moitié supérieure de");
gotoxy(10, 10);
```

```
     puts(" la matrice de la manière suivante:  ┌                  ┐");
gotoxy(11,10);
     puts("                                     |                  |");
gotoxy(12,10);
     puts("                                     |K1      K2      K3|");
gotoxy(13,10);
     puts("                                     |                  |");
gotoxy(14,10);
     puts("                                     |        K4      K5|");
gotoxy(15,10);
     puts("                                     |                  |");
gotoxy(16,10);
     puts("                                     |SYM               K6|");
gotoxy(17,10);
     puts("                                     |                  |");
gotoxy(18,10);
     puts("                                     └                  ┘");
gotoxy(24,30); puts (" Presser une touche quelconque pour continuer");
      getch();
      cls();

        /* création du fichier clas-per */
     gotoxy(5,5);
     puts(" classe      K1          K2          K3          K4
                                                K5          K6 ");
       for(i=1; i<=ncper; i++)
       {
           gotoxy(6+n,7);
           printf("%d",i);
              for(j=1; j<=6; j++)
              {
              gotoxy(6+n,7+11*j);
              scanf("%f",&K[j-1]);
              }
              /* Ecriture sur le fichier clas-perm */

      fprintf(f1,"%d %f %f %f %f %f %f\n",i,K[0],K[1],K[2],K[3],K[4],K[5]);

           n++; if(n%16==0) { n=1;
           cls();       gotoxy(5,5);
     puts(" classe      K1          K2          K3          K4
                                                      K5          K6 ");
         }
       }
       fclose(f1);
       gotoxy(24,20);

       puts("  DONNEES CORRECTES 'O' OU 'N'"); op=getch();

     while(op!='o' && op!='O' && op!='n' && op!='N') {

        printf("\a"); gotoxy(24,20); op=getch();
}
```

```
    } while(op=='n' || op=='N');
      return (ncper);
  }
  int nbnoeud(int Type)
  {
  int nd=0;
     if (Type==23)  nd=3;
     else if (Type==36)  nd=6;
     else if (Type==48)  nd=8;
     else if (Type==49)  nd=9;
     else if (Type==410) nd=10;
     else if (Type==620) nd=20;
     else  nd=27;
     return (nd);
  }

int noeud(float Hd[230],float Qd[230])
/* saisi  des caractéristiques de chaque noeuds */
  {
  int i,nbtnd,Hhcx,n=1;
  float x,y,z,ValH,debit;
  char op;
  int fd3;
  FILE *f3;
  do {
  n=1;
  delete_file("A:CORG");
  fd3=open("a:CORG",O_TRUNC|O_CREAT|O_TEXT,0x0100|0x0080);
  f3=fdopen(fd3,"r+t");
  cls();
  /* création du fichier CORG à accés direct */
 gotoxy(10,10);
  puts("Donner le nombre total de noeuds sur le domaine....");
  gotoxy(10,64); scanf("%d",&nbtnd);cls();

   for(i=0; i<=(nbtnd-1); i++)  {
       Qd[i]=0.0; Hd[i]=0.0; }

  gotoxy(2,5);
puts("*************************************************************************************");
  gotoxy(3,5);
puts("noeud N      X         Y          Z         Hhcx      ValH        debit");
  gotoxy(4,5);
puts("            ( m )     ( m )      ( m )    (-1 ou 0 )   ( m )       ( m3/s) ");
  gotoxy(5,5);
puts("*************************************************************************************");
```

```
        for(i=1;  i<=nbtnd;  i++)
          {

          gotoxy(5+n, 8);
          printf("%d", i);
          gotoxy(5+n, 18);
          scanf("%f", &x);
          gotoxy(5+n, 28);
          scanf("%f", &y);
          gotoxy(5+n, 38);
          scanf("%f", &z);
          gotoxy(5+n, 48);
          scanf("%d", &Hhcx);
          while(Hhcx!=-1 && Hhcx!=0)
          {

          Hhcx=0;
         gotoxy(5+n, 48);  printf("\a     "); gotoxy(5+i, 48);
          scanf("%d", &Hhcx);
          }
          gotoxy(5+n, 58);
          scanf("%f", &ValH);
          gotoxy(5+n, 68);
          scanf("%f", &debit);   if(Hhcx==0) ValH=0;  else debit=0;
               /* écriture dans le fichier CORG */
               fprintf(f3, "%d %f %f %f %d %f %f \n", i, x, y, z, Hhcx, ValH, debit);
               calcul_Q_H(Hhcx, debit, ValH, i, Qd, Hd);
             n++;  if((n%16)==0) { n=1;
             cls();
    gotoxy(2, 5);
puts("*************************************************************************");
    gotoxy(3, 5);
puts("noeud N       X          Y          Z         Hhcx       ValH        debit");
    gotoxy(4, 5);
puts("             ( m )     ( m )      ( m )    (-1 ou 0 )    ( m )       ( m3/s ");
    gotoxy(5, 5);
puts("*************************************************************************");
             }
           }
           fclose(f3);
          gotoxy(24, 20);
puts("  DONNEES CORRECTES 'O' OU 'N'"); op=getch();
         while(op!='o' && op!='O' && op!='n' && op!='N') {
         printf("\a");  gotoxy(24, 20);
         op=getch();
         }
        } while(op=='n' || op=='N');
          return (nbtnd);
      }
      /* Traitement des données relatif aux divers type d'éléments */
```

```
int element(int nc, int nb, int Ide[])
/* saisie des differents élèments */
     {
     int nbelt=0, j, n=1, ni=1;
     int nd=0, i, Ndi[27];
     char op;
     struct NDS xt;
     int fd2;
     FILE *f2;
     do {
     n=1; ni=1; nd=0; nbelt=0;
     delete_file("A:LOCE");
     fd2=open("a:LOCE", O_TRUNC|O_CREAT|O_BINARY, 0x0100|0x0080);
     f2=fdopen(fd2, "r+");
     cls(); gotoxy(10, 10);
                    puts("*****************************************");
    gotoxy(12, 10); puts("** Donner le nombre total d'éléments .....");
    gotoxy(14, 10); puts("*****************************************");
    gotoxy(12, 52); scanf("%d", &nbelt);
     cls();
     /*Creation du fichier loce  */
      for(i=0; i<nb;  i++) Ide[i]=0;
     gotoxy(5, 1);
  puts("Numero Type classe  Ald  ND1  ND2  ND3  ND4  ND5  ND6  ND7
                                                ND8  ND9  ND10  ");
        for(i=1;  i<=nbelt;  i++)
         {

         gotoxy(5+ni, 4);
         printf("%d", i);
         xt.Num=i;
         gotoxy(5+ni, 9);
         scanf("%d", &xt.Ty);
         while(xt.Ty!=23 && xt.Ty!=36 && xt.Ty!=48 && xt.Ty!=49 &&
                       xt.Ty!=410 && xt.Ty!=620 && xt.Ty!=627)
          {
         gotoxy(5+ni, 9); printf("       \a"); gotoxy(5+ni, 9);
         scanf("%d", &xt.Ty);
         }
         gotoxy(5+ni, 16); xt.classe=0;
         scanf("%d", &xt.classe);
          while(xt.classe <1||xt.classe >nc)
          {

         gotoxy(5+ni, 16); printf("      \a");
         gotoxy(5+ni, 16); xt.classe=0;
         scanf("%d", &xt.classe);
          }
         gotoxy(5+ni, 22);
          scanf("%f", &xt.Ald);
          nd=nbnoeud(xt.Ty); n=1; j=0;
                  /* determine le nombre de noeud de l'élèment */
```

```
          for(j=0; j<27; j++) xt.Ndi[j]=0;
           for(j=1; j<=nd; j++)
           {
            gotoxy(5+ni,22+5*n);
            scanf("%d",&xt.Ndi[j-1]);
             gotoxy(5+ni,22+5*n);
            printf("%d",xt.Ndi[j-1]);
             while(xt.Ndi[j-1]<1 || xt.Ndi[j-1]>nb)
             {

            gotoxy(5+ni,22+5*n);
            printf("      \a");
            gotoxy(5+ni,22+5*n);
            scanf("%d",&xt.Ndi[j-1]);
            gotoxy(5+ni,22+5*n);
            printf("%d",xt.Ndi[j-1]);
             }
             n++;
             if(j%10==0) { n=1; ni++; }
            }
            fwrite(&xt,sizeof(xt),1,f2);
             ni++;
             if(ni%16==0) { ni=1;
             cls();
    gotoxy(5,1);
    puts("Numero Type classe  Ald  ND1  ND2  ND3  ND4  ND5  ND6  ND7
                                                    ND8  ND9  ND10  ");
           }
            for(j=0; j<=(nd-1); j++) Ndi[j]=xt.Ndi[j];

            foncId(nd,Ndi,Ide);
         }
         fclose(f2);
          gotoxy(24,20);
 puts("  DONNEES CORRECTES 'O' OU 'N'"); op=getch();
        while(op!='o' && op!='O' && op!='n' && op!='N') {
        printf("\a"); gotoxy(24,20);
        op=getch();
        }
       } while(op=='n' || op=='N');
          return(nbelt);
   }

void foncId(int ntd,int Ndi[],int Ide[])
     {
     int i;
       for(i=0; i<=(ntd-1); i++)
                    Ide[Ndi[i]-1]++;
      }
```

```
void calcul_Q_H(int hcx,float Dt,float h,int n,float Qd[230],float Hd[230])
    {
         if(hcx==0) {
              Qd[n-1]=Dt; Hd[n-1]=0.0; }
         if(hcx==-1) {
              Qd[n-1]=0; Hd[n-1]=h;   }
     }
 void setcursor(int xt,int yt)
    {
     union REGS reg;
       reg.h.ah=1;
       reg.x.cx= (xt << 8) + yt;
       int86(0X10,&reg,&reg);
     }
 char controle_entree_donnees(int *nbnd,int *nc,char *option,char kh)
    {
    char ch,*name;
    cls(); gotoxy(3,10);
    if(kh=='C') puts(" SAISIE DES COORDONNEES ");
    if(kh=='P') puts(" SAISIE DES PERMEABILITES ");

    gotoxy(5,5); puts("                    MENU ");
    gotoxy(7,5); printf(" < E >  DONNEES SUR LES %s EXISTANT   ",option);
    gotoxy(9,5); puts(" < C >   CREER A PARTIR DE L'EDITEUR SEE ");
    gotoxy(11,5); puts(" < U >  UTILISER LE PROGRAMME DE SAISI EXISTANT");
    gotoxy(14,20); puts("  ** VOTRE CHOIX ** ");
   gotoxy(14,42); ch=getch();
   while(ch!='e' && ch!='E' && ch!='c' && ch!='C' && ch!='u' && ch!='U') {
       printf("\a"); gotoxy(14,42); ch=getch(); }
   if(ch=='e' || ch=='E')
      {
    cls(); gotoxy(7,5);name=" ";
    if(kh=='P') puts(" DONNER LE NOM DU FICHIER : EXEMPLE < A:clasper >");
    if(kh=='C') puts(" DONNER LE NOM DU FICHIER : EXEMPLE < A:CORG >");
    gotoxy(7,55); scanf("%s",name);
    gotoxy(9,10);
    if(kh=='P')
    {
     puts(" DONNER LE NOMBRE TOTAL DE PERMEABILITES ");
     gotoxy(9,50); scanf("%d",nc);
      }
    if(kh=='C')
    {
     puts(" DONNER LE NOMBRE TOTAL DE NOEUDS ");
    gotoxy(9,50); scanf("%d",nbnd);
    }
    }
    else  if(ch=='c' || ch=='C')
    {
    cls();
```

```
 if(kh=='C')
 {
 gotoxy(10,10); puts(" DONNER LE NOMBRE TOTAL DE NOEUDS ");
 gotoxy(10,45); scanf("%d",nbnd);
 system("see a:CORG");
 }
 if(kh=='P')
 {
 gotoxy(10,10); puts(" DONNER LE NOMBRE TOTAL DE PERMEABILITES ");
 gotoxy(10,52); scanf("%d",nc);
 system("see a:clasper");
 }
 cls();
 }
 else ch='m';
 return(ch);
 }
void Exist_noeud(int nbnd,float Hd[230],float Qd[230])
    {
    float x,y,z;
    int i,hc,fd,n;
    FILE *f;
    fd=open("A:CORG",O_CREAT|O_TEXT,0x0100);
    f=fdopen(fd,"rt");
    for(i=0; i<nbnd; i++)
      {
      fscanf(f,"%d %f %f %f %d %f %f",&n,&x,&y,&z,&hc,&Hd[i],&Qd[i]);
      if(hc==-1) Qd[i]=0;
      else if(hc==0) Hd[i]=0;
      else { cls(); gotoxy(5,5);
      printf("ERREUR SUR LA CONDITION AUX LIMITES DANS LE FICHIER LOCE");
      gotoxy(7,15); printf("SUR LA LIGNE %d",(i+1));
      exit(1);
      }
   }
```

```
/* ************************************************************************
                        SORTI.C
 ************************************************************************
          *    PROJET DE FIN D'ETUDE *
           -------------------------
     DIRECTEUR: GERARD SOUMA
     CO-DIRECTEUR: AMADOU SARR
     AUTEUR : KODJOVI MAWUMETO DEGUE

  TITRE: Conception d'un logiciel pour simuler par la
      la méthode des éléments finis les écoulements en
       milieu poreux saturé

 ************************************************************************
 */
 #include <fcntl.h>
 #include <dos.h>
 #include <stdlib.h>
 #include <stdio.h>
 #include <bios.h>
 #include <io.h>
 #include "fonccom.c"
   struct NDS {
      int Num;
      int Ty;
      int classe;
      float Ald;
      unsigned int Ndi[27];
      };

 main()
   {
int i, j, fd1, fd2, fd3, fd, fd4, fd5;
struct NDS xt;
char *form, ch, *name;
float som1, som2, K[6], x, y, z, h, debit;
int nbnd=0, nbelt=0, nbnc=0, n=0, Hhc;
float Xt[230], Yt[230], Ht[230];
int nd=0, ni=1;
FILE *f1, *f2, *f3, *f, *f4, *f5;

  fd1=open("A:LOCE", O_BINARY);
  fd2=open("A:CORG", O_CREAT|O_TEXT, 0x0100|0x0080);
  fd3=open("A:clasper", O_CREAT|O_TEXT, 0x0100|0x0080);
  fd=open("lpt1", O_CREAT);
  fd4=open("A:RESUL", O_CREAT|O_TEXT, 0x0100|0x0080);
  fd5=open("A:RESULTAT", O_CREAT|O_TEXT);
  f5=fdopen(fd5, "rt");
  f4=fdopen(fd4, "rt");
  f=fdopen(fd, "w");
  f1=fdopen(fd1, "rb");
  f2=fdopen(fd2, "rt");
  f3=fdopen(fd3, "rt");
```

```
      fscanf(f4,"%d %d %d",&nbnd,&nbelt,&nbnc);
      fclose(f4); cls();

 gotoxy(6,15);   puts("                                                    ");
 gotoxy(7,15);   puts("                                                    ");
 gotoxy(8,15);   puts("      CHOIX                                         ");
 gotoxy(9,15);   puts("                                                    ");
 gotoxy(10,15);  puts("       < E >    IMPRESSION A ECRAN                  ");
 gotoxy(11,15);  puts("                                                    ");
 gotoxy(12,15);  puts("       < I >    IMPRESSION SUR IMPRIMANTE           ");
 gotoxy(13,15);  puts("                                                    ");
 gotoxy(14,15);  puts("       ** VOTRE CHOIX **                            ");
 gotoxy(15,15);  puts("                                                    ");
 gotoxy(16,15);  puts("                                                    ");
 gotoxy(17,15);  puts("                                                    ");
 gotoxy(18,15);  puts("                                                    ");

     gotoxy(14,46); ch=getch();

   while(ch!='e' &&ch!='E' && ch!='i' && ch!='I')
     {
     printf("\a"); gotoxy(14,46); ch=getch();
     }
      if(ch=='i' || ch=='I') {
      do {
         n=biosprint(1,90,0);
         if(n==1 || n==8 || n==64) {
        gotoxy(22,20);
printf("\a VERIFIER VOTRE IMPRIMANTE ");
        gotoxy(24,20);
puts(" APPUYER SUR <RETOUR> SI VOUS ETES PRET");
          getch();  }
          if(n==128) {
        gotoxy(22,20);
printf("\a ALLUMER VOTRE IMPRIMANTE ");
        gotoxy(24,20);
puts(" APPUYER SUR <RETOUR> SI VOUS ETES PRET");
          getch();  }
            if(n==32) {
   gotoxy(22,20); printf("\a METTEZ EN PLACE DU PAPIER ");
   gotoxy(24,20); puts(" APPUYER SUR <RETOUR> SI VOUS ETES PRET");
          getch();  }
       } while(n==16);
      gotoxy(22,1); cls();
 fprintf(f," ***** NOMBRE TOTAL D'ELEMENT ***********      %d\n",nbelt);
 fprintf(f," ***** NOMBRE TOTAL DE NOEUD ***********      %d\n",nbnd);
 fprintf(f," ***** NOMBRE TOTAL DE PERMEABILITE *****     %d\n",nbnc);
     fprintf(f,"\n\n\n        ** PERMRABILITE** ");
 fprintf(f,"\n\n\n-------------------------------------------
                                          ----------------------------");
 fprintf(f,"\n  CLASSE        K1          K2          K3          K4
                                                  K5          K6 ");
 fprintf(f,"\n-------------------------------------------------------
                                          ----------------------");
```

```
    for(i=0; i<nbnc; i++) {
     form="\n    %3d     %10.5f %10.5f %10.5f %10.5f %10.5f %10.5f ";
  fscanf(f3,"%d %f %f %f %f %f %f ",&n,&K[0],&K[1],&K[2],&K[3],&K[4],&K[5]);
fprintf(f,form,n,K[0],K[1],K[2],K[3],K[4],K[5]);
     }
 fprintf(f,"\n\n");
     fclose(f3);
  fprintf(f,"\n\n\n\n    *** CONNECTIVITE ***");
 fprintf(f,"\n\n\n\n-------------------------------------------------------
                                                  -------------------------");
 fprintf(f,"\n ELT N° TYPE  CLASSE  ALD   ND1   ND2   ND3   ND4   ND5
                                                  ND6 ND7   ND8   ND9");
 fprintf(f,"\n-------------------------------------------------------------
                                                  ---------------------");
       form="\n  %3d   %3d   %3d%8.3f";
      for(i=0; i<nbelt; i++) {
     fread(&xt,sizeof(xt),1,f1);
      fprintf(f,form,xt.Num,xt.Ty,xt.classe,xt.Ald);
      nd=nbnoeud(xt.Ty);
      for(j=1; j<=nd; j++) {
            fprintf(f,"  %4d",xt.Ndi[j-1]);
            if(j%9==0) fprintf(f,"\n                  ");
              }
       }
       fprintf(f,"\n                        ");
      gotoxy(5,5); puts(" ARRANGER LE PAPIER SI VOUS LE DESIRER");
      getch();
        fprintf(f,"\n\n\n");
       fclose(f1);
 fprintf(f," \n\n\n\n    *** COORDONNEES ET CONDITIONS AUX LIMITES ****");
 fprintf(f,"\n\n-------------------------------------------------------
                                                  -----------------");
 fprintf(f," \n NOEUD N°      XCOORD    YCOORD    ZCOORD    Hhcx
                                                  Valh      DEBIT");
 fprintf(f," \n               ( m )     ( m )     ( m )
                                                  ( m )     ( m3/s )");
 fprintf(f,"\n-------------------------------------------------------
                                                  -------------------");
      form="\n    %4d    %10.3f%10.3f%10.3f    %4d     %8.3f  %8.4f ";
       for(i=0; i<nbnd; i++)  {
       fscanf(f2,"%d %f %f %f %d %f %f",&n,&x,&y,&z,&Hhc,&h,&debit);
     fprintf(f,form,n,x,y,z,Hhc,h,debit);
     Xt[i]=x; Yt[i]=y;
     } fclose(f2);
     fprintf(f,"\n                  ");
      gotoxy(5,5); puts(" ARRANGER LE PAPIER SI VOUS LE DESIRER");
      getch();
     fprintf(f,"\n\n\n");
   fprintf(f,"\n\n\n\n  ***********************************");
   fprintf(f," \n          RESULTATS CALCULES              ");
   fprintf(f," \n ------------------------------------");
 fprintf(f,"\n\n\n-------------------------------------------------------------");
 fprintf(f,"\n   VARIABLE N°       CHARGE CALCULEE         DEBIT CALCULE ");
 fprintf(f," \n                          ( m )               ( m3 / s) ");
```

```
fprintf(f,"\n-------------------------------------------------------------");
        form="\n       %4d           %12.6f          %12.5f    ";
      som1=0; som2=0; x=0; y=0;
for(i=0; i<nbnd; i++) {

fscanf(f5,"%d %e %e",&n,&x,&y);
if(y<0) som1+=y;
if(y>0) som2+=y;
fprintf(f,form,n,x,y);
Ht[i]=x;
}  fclose(f5);
fprintf(f,"\n\n\n\n     ******** SOMME DES DEBITS ENTRANT     : %25.9e",som2);
fprintf(f,"\n     ******** SOMME DES DEBITS SORTANT     : %25.9e",som1);
fprintf(f,"\n     ******** SOMME DE TOUT LES DEBITS     : %25.9e",(som1+som2));
fprintf(f," \n\n                ");
        gotoxy(22,1);
         }

 if(ch=='E' || ch=='e') { printf("\n\n\n\n");cls();

printf("\n\n ***** NOMBRE TOTAL D'ELEMENT ***********      %d\n",nbelt);
printf(" ***** NOMBRE TOTAL DE NOEUD ************      %d\n",nbnd);
printf(" ***** NOMBRE TOTAL DE PERMEABILITE *****      %d\n",nbnc);
    printf("\n\n\n        ** PERMRABILITE** ");
printf("\n\n\n-------------------------------------------------------------
                                                  ----------------");
printf("\n   CLASSE        K1          K2          K3          K4
                                                   K5          K6 ");
printf("\n-------------------------------------------------------------
                                                 -------------------");
   for(i=0; i<nbnc; i++) {
 form="\n   %3d     %10.3f %10.3f %10.3f %10.3f %10.3f %10.3f ";

 fscanf(f3,"%d %f %f %f %f %f %f ",&n,&K[0],&K[1],&K[2],&K[3],&K[4],&K[5]);

printf(form,n,K[0],K[1],K[2],K[3],K[4],K[5]);
    }
printf("\n\n");
    fclose(f3);
    gotoxy(24,10); puts("PESER SUR UNE TOUCHE POUR CONTINUER");
    getch(); cls();
 printf("\n\n\n\n   *** CONNECTIVITE ***");
printf("\n\n\n\n-------------------------------------------------------------
                                                 -------------------");
printf("\n ELT N° TYPE  CLASSE  ALD   ND1   ND2   ND3   ND4   ND5   ND6
                                                  ND7   ND8   ND9");
printf("\n-------------------------------------------------------------
                                                 ----------------");
     form="\n  %3d   %3d   %3d%8.3f";
     for(i=0; i<nbelt; i++) {
    fread(&xt,sizeof(xt),1,f1);
     printf(form,xt.Num,xt.Ty,xt.classe,xt.Ald);
     nd=nbnoeud(xt.Ty);
```

```
      for(j=1; j<=nd; j++) {
            printf("  %4d",xt.Ndi[j-1]);
            if(j%9==0) printf("\n                          ");
              }
       }
        printf("\n\n\n");
       fclose(f1); gotoxy(24,10); puts("PESER SUR UNE TOUCHE POUR CONTINUER");
       getch(); cls();

gotoxy(2,10); printf(" *** COORDONNEES ET CONDITIONS AUX LIMITES ****");
gotoxy(7,5); printf("-------------------------------------------------
                              ------------------------");
gotoxy(8,5); printf("  NOEUD N°      XCOORD     YCOORD     ZCOORD     Hhcx
                          Valh       DEBIT");
gotoxy(9,5); printf("                  ( m )      ( m )      ( m )
                            ( m )       ( m3/s)");
gotoxy(10,5);
printf("-------------------------------------------------------------------
                                        -----------");
       for(i=1; i<=nbnd; i++)  {

       fscanf(f2,"%d %f %f %f %d %f %f",&n,&x,&y,&z,&Hhc,&h,&debit);

       gotoxy(10+ni,9); printf("%4d",n);
       gotoxy(10+ni,16); printf("%10.3f",x);
       gotoxy(10+ni,26); printf("%10.3f",y);
       gotoxy(10+ni,36); printf("%10.3f",z);
       gotoxy(10+ni,48); printf("%4d",Hhc);
       gotoxy(10+ni,58); printf("%8.3f",h);
       gotoxy(10+ni,68); printf("%8.3f",debit);
       Xt[i-1]=x; Yt[i-1]=y;
       ni++;
       if(ni%15==0) { ni=1; printf("\n\n  APPUYER SUR UNE TOUCHE POUR CONTINUER "
       getch();cls();
      gotoxy(7,5); printf("-------------------------------------------------
                               ---------------------------");
   gotoxy(8,5); printf("  NOEUD N°      XCOORD     YCOORD     ZCOORD     Hhcx

                            Valh       DEBIT");
  gotoxy(9,5); printf("                  ( m )      ( m )      ( m )
                              ( m )      ( m3/s)");
      gotoxy(10,5); printf("-------------------------------------------
                                 ---------------------------------");

      }
     }
     gotoxy(24,10);
     puts(" APPUYER SUR UNE TOUCHE POUR CONTINUER ");
     fclose(f2);
     getch(); cls();
     printf("\n\n\n");
```

```
  printf("\n\n\n\n  ************************************");
  printf(" \n          RESULTATS CALCULES              ");
  printf(" \n ---------------------------------------");
printf("\n\n\n----------------------------------------------------------");
printf("\n   VARIABLE N°       CHARGE CALCULEE         DEBIT CALCULE ");
printf("\n                         ( m )                  ( m3 / s)");
printf("\n----------------------------------------------------------");
     form="\n       %4d           %12.6f            %12.4f    ";
     som1=0; som2=0;x=0; y=0;
for(i=1; i<=nbnd; i++) {

fscanf(f5,"%d %e %e ",&n,&x,&y);
if(y<0) som1+=y;
if(y>0) som2+=y;
printf(form,n,x,y);
Ht[i-1]=x;
if(i%13==0) { gotoxy(24,5); printf("\n\n");
          puts("PESER SUR UNE TOUCHE POUR CONTINUER L'IMPRESSION");
             getch(); cls();
 printf("\n\n\n----------------------------------------------------------");
printf("\n   VARIABLE N°       CHARGE CALCULEE         DEBIT CALCULE ");
printf("\n                         ( m )                  ( m3 / s )  ");
printf("\n----------------------------------------------------------");
     }
}  fclose(f5);
printf("\n\n\n\n     SOMME DES DEBITS ENTRANT    : %15.9e",som2);
printf("\n    SOMME DES DEBITS SORTANT    : %15.9e",som1);
printf("\n    SOMME DE TOUT LES DEBITS  : %15.9e",(som1+som2));
printf(" \n\n               ");
       }
   gotoxy(24,5); puts(" APPUYER SUR UNE TOUCHE POUR CONTINUER ");
  getch(); cls();
  gotoxy(5,5); puts(" VOULER VOUS CREER UN FICHIER UNTILISABLE PAR CONTOUR");
  gotoxy(5,62); ch=getch();
  if(ch=='o' || ch=='O') {
   gotoxy(8,5); puts(" DONNER LE NOM DU FICHIER < EXEMPLE A:GRAND >");
   name=" ";
   gotoxy(8,55); scanf("%s",name);

    fd4=open(name,O_RDWR|O_TEXT,0x0100|0x0080);
 if(fd4==-1) fd4=creat(name,0x0100|0x0080);
          f4=fdopen(fd4,"r+t");

   for(i=0; i<nbnd; i++)
   fprintf(f4,"%f %f %f\n",Xt[i],Yt[i],Ht[i]);
         }
         fclose(f4);
     }
```

```
int nbnoeud(int type)
   {
     int nd=0;
     if(type==23) nd=3;
     if(type==36) nd=6;
     if(type==48) nd=8;
     if(type==49) nd=9;
     if(type==410) nd=10;
     if(type==620) nd=20;
     if(type==627) nd=27;
     return(nd);
```

```
      /**********  PTYPE36.C   **********/

float PRTYPE36(int i,float A[27][27],float FSe[6])
{
  float a,b,c,n1=0,n2=0,w1=0;
  float e=0,f1=0,h=0;

 a=4701.42064E-4; /*(6+sqrt(15))/21;*/
 b=1012.86507E-4; /*4/7-a;*/
 c=661.97076E-4; /*(155+sqrt(15))/2400;*/


   if(i==0) { n1=n2=3333.333333E-4 ; w1=0.1125; }
   if(i==1) { n1=n2=a;   w1=c; }
   if(i==2) {n1=597.15871E-4; n2=a; w1=c; }
   if(i==3) {n1=a; n2=597.15871E-4; w1=c; }
   if(i==4) {n1=n2=b ;w1=629.6959E-4; }
   if(i==5) {n1=7974.26985E-4; n2=b ; w1=629.6959E-4;}
   if(i==6) {n1=b ; n2=7974.26985E-4; w1=629.6959E-4;}

                  w1=w1;

   e=4*n1; f1=-3+e; h=4*n2;

   A[0][0]=A[0][1]=f1+h;
        e=2*n1; f1=e+n2; h=1-f1;
   A[1][0]=4*h;
   A[1][1]=-4*n1;
   A[2][0]=-1+4*n1;
   A[2][1]=0.0;
   A[3][0]=4*n2;
   A[3][1]=4*n1;
   A[4][0]=0.0;
          e=4*n2; f1=e-1;
   A[4][1]=f1;
   A[5][0]=-4*n2;
         e=2*n2; f1=n1+e; h=1-f1;
   A[5][1]=4*h;


        e=1-n1-n2;f1=n1+n2;h=e*(2*f1-1);


       FSe[0]=-h;
       FSe[1]=4*n1*e;
             e=1-2*n1;
       FSe[2]=-n1*e;
       FSe[3]=4*n1*n2;
              e=1-2*n2;
       FSe[4]=-n2*e;
              f1=1-n1-n2;
       FSe[5]=4*n2*f1;
               return(w1);
```

```
          /**********     PTYPE48.C    **********/

float PRTYPE48(int i,float A[27][27],float FSe[8])
       {

    float n1,n2,w1;
    float a=0,b=0,c=0,d=0;
          a=25; b=81; c=40; d=64;


      if(i==1) { n1=n2=-sqrt(0.6);       w1=a/b; }
      else if(i==2) {  n1=-sqrt(0.6); n2=0 ; w1=c/b; }
      else if(i==3) {  n1=-sqrt(0.6);n2=-n1; w1=a/b; }
      else if(i==4) {  n1=0.0;n2=-sqrt(0.6); w1=c/b; }
      else if(i==5) {  n1=n2=0.0;            w1=d/b; }
      else if(i==6) {  n1=0.0; n2=sqrt(0.6); w1=c/b; }
      else if(i==7) {  n1=sqrt(0.6); n2=-n1; w1=a/b; }
      else if(i==8) {  n1=sqrt(0.6); n2=0.0; w1=c/b; }
      else {  n1=n2=sqrt(0.6);       w1=a/b; }

       w1=w1;
      a=2*n1+n2;b=1-n2;


    A[0][0]=0.25*b*a;
          a=1-n1; b=n1+2*n2;
    A[0][1]=0.25*a*b;
    A[1][0]=-(1-n2)*n1;
          a=1-n1*n1;
    A[1][1]=-0.5*a;
          a=1-n2;   b=2*n1-n2;
    A[2][0]=0.25*a*b;
          a=1+n1;   b=n1-2*n2;
    A[2][1]=-0.25*a*b;
           a=1-n2*n2;
    A[3][0]=0.5*a;
           a=1+n1;
    A[3][1]=-a*n2;
           a=1+n2; b=2*n1+n2;
    A[4][0]=0.25*a*b;
           a=1+n1; b=n1+2*n2;
    A[4][1]=0.25*a*b;
            a=1+n2;
    A[5][0]=-a*n1;
            a=1-n1*n1;
    A[5][1]=0.5*a;
          a=2*n1-n2; b=1+n2;
    A[6][0]=0.25*b*a;
           a=1-n1; b=n1-2*n2;
    A[6][1]=-0.25*a*b;
           a=1-n2*n2;
    A[7][0]=-0.5*a;
             a=1-n1;
```

```
A[7][1]=-a*n2;
    a=1+n1+n2; b=1-n1; c=1-n2;


FSe[0]=0.25*b*c*a;
    a=1-n1*n1; b=1-n2;
FSe[1]=0.5*a*b;
     a=1+n1; b=1-n2; c=1-n1+n2;
FSe[2]=-0.25*a*b*c;
      a=1+n1; b=1-n2*n2;
FSe[3]=0.5*a*b;
       a=1+n1; b=1+n2; c=1-n1-n2;
FSe[4]=-0.25*a*b*c;
       a=1-n1*n1; b=1+n2;
FSe[5]=0.5*a*b;
       a=1-n1; b=1+n2; c=1+n1-n2;
FSe[6]=-0.25*a*b*c;
        a=1-n1;   b=1-n2*n2;
FSe[7]=0.5*a*b;
           return(w1);
```

```
        /**********     PTYPE49.C    **************/

float PRTYPE49(int i,float A[27][27],float FSe[9])
  {
  int j;
  float n1=0,n2=0,x=0,y=0,w1=0;
  float a=0,b=0,c=0;

 y=81;

   if(i==1) { n1=n2=-sqrt(0.6); x=25;    w1=x/y; }
   if(i==2) { n1=-sqrt(0.6); n2=0 ;x=40;   w1=x/y; }
   if(i==3) { n1=-sqrt(0.6);n2=-n1;x=25;   w1=x/y; }
   if(i==4) { n1=0.0;n2=-sqrt(0.6);x=40;   w1=x/y; }
   if(i==5) { n1=n2=0.0;   x=64;           w1=x/y; }
   if(i==6) { n1=0.0;n2=sqrt(0.6);x=40;   w1=x/y; }
   if(i==7) { n1=sqrt(0.6);n2=-n1; x=25; w1=x/y; }
   if(i==8) { n1=sqrt(0.6);n2=0.0; x=40; w1=x/y; }
   if(i==9) { n1=n2=sqrt(0.6); x=25;      w1=x/y; }

   A[0][0]=0.25*(1-2*n1);
   A[0][0]*=(1-n2)*n2;
   A[0][1]=0.25*(1-n1);
   A[0][1]*=(1-2*n2)*n1;
   A[1][0]=(1-n2)*n1*n2;
   A[1][1]=-(0.5)*(1-n1*n1);
   A[1][1]*=(1-2*n2);
   A[2][0]=-(0.25)*(1+2*n1);
   A[2][0]*=(1-n2)*n2;
   A[2][1]=-0.25*(1+n1);
   A[2][1]*=(1-2*n2)*n1;
   A[3][0]=0.5*(1+2*n1);
   A[3][0]*=(1-n2*n2);
   A[3][1]=-(1+n1)*n1*n2;
   A[4][0]=0.25*(1+2*n1);
   A[4][0]*=(1+n2)*n2;
   A[4][1]=0.25*(1+n1);
   A[4][1]*=(1+2*n2)*n1;
   A[5][0]=-(1+n2)*n1*n2;
   A[5][1]=0.5*(1-n1*n1);
   A[5][1]*=(1+2*n2);
   A[6][0]=-(0.25)*(1-2*n1);
   A[6][0]*=(1+n2)*n2;
   A[6][1]=-(0.25)*(1-n1);
   A[6][1]*=(1+2*n2)*n1;
   A[7][0]=-(0.5)*(1-2*n1);
   A[7][0]*=(1-n2*n2);
   A[7][1]=(1-n1)*n1*n2;
   A[8][0]=-2*(1-n2*n2)*n1;
   A[8][1]=-2*(1-n1*n1)*n2;
```

```
      a=1-n1; b=1-n2; c=a*b*n1;
FSe[0]=0.25*c*n2;
      a=1-n1*n1;  b=1-n2; c=a*b*n2;
FSe[1]=-0.5*c;
      a=1+n1; b=1-n2; c=a*b*n1*n2;
FSe[2]=-0.25*c;
      a=1+n1; b=1-n2*n2; c=a*b*n1;
FSe[3]=0.5*c;
      a=1+n1; b=1+n2; c=a*b*n1*n2;
FSe[4]=0.25*c;
       a=1-n1*n1; b=1+n2; c=a*b*n2;
FSe[5]=0.5*c;
       a=1-n1;  b=1+n2;  c=a*b*n1*n2;
FSe[6]=-0.25*c;
       a=1-n1; b=1-n2*n2; c=a*b*n1;
FSe[7]=-0.5*c;
       a=1-n1*n1; b=1-n2*n2;
FSe[8]=a*b;
return(w1);
  }
```

```
   /************  PTYPE410.C **************/

#include <math.h>
#include <fcntl.h>
#include <stdio.h>
#include <io.h>
   main()
      {

int i,j,fd;
float n1,n2,n3,A[10][3],FSe[10];
float a,b,c,d,c2,a1,b1,b2,c1,d1,e1,w1;
FILE *f;


fd=open("A:DATA410",O_RDWR|O_CREAT|O_TRUNC|O_TEXT,0x0100|0x0080);
if(fd==-1) creat("A:DATA410",0x0100|0x0080);
f=fdopen(fd,"r+t");
  printf("fd=%d\n",fd);getch();

for(i=0;  i<=14;  i++) {

a1=0.25;  b1=3197.9363E-4;  b2=919.7108E-4;
c1=406.1912E-4;  c2=7240.8677E-4;
d1=563.5083E-4;  e1=4436.4917E-4;
          a=8;b=405;c=5;d=567;


       if(i==0) {n1=n2=n3=a1;  w1=a/b;  }
       if(i==1) {n1=n2=n3=b1;       w1=115.1137E-4;  }
       if(i==2) {n1=n2=b1;  n3=c1;  w1=115.1137E-4;  }
       if(i==3) {n1=n3=b1;  n2=c1;  w1=115.1137E-4;  }
       if(i==4) {n1=c1;  n2=n3=b1;  w1=115.1137E-4;  }
       if(i==5) {n1=n2=n3=b2;       w1=119.8951E-4;  }
       if(i==6) {n1=n2=b2;  n3=c2;  w1=119.8951E-4;  }
       if(i==7) {n1=n3=b2;  n2=c2;  w1=119.8951E-4;  }
       if(i==8) {n1=c2;  n2=n3=b2;  w1=119.8951E-4;  }
       if(i==9) {n1=n2=d1;  n3=e1;  w1=c/d;          }
      if(i==10) {n1=n3=d1;  n2=e1;  w1=c/d;          }
      if(i==11) {n1=e1;  n2=n3=d1;  w1=c/d;          }
      if(i==12) {n1=d1;  n2=n3=e1;  w1=c/d;          }
      if(i==13) {n1=n3=e1;  n2=d1;  w1=c/d;          }
      if(i==14) {n1=n2=e1;  n3=d1;  w1=c/d;          }
```

```
          a=1-n1-n2-n3;

         A[0][0]=1-4*a;
         A[0][1]=1-4*a;
         A[0][2]=1-4*a;
         A[1][0]=4*(a-n1);
         A[1][1]=-4*n1;
         A[1][2]=-4*n1;
         A[2][0]=-1+4*n1;
         A[2][1]=0.0;
         A[2][2]=0.0;
         A[3][0]=4*n2;
         A[3][1]=4*n1;
         A[3][2]=0.0;
         A[4][0]=A[4][2]=0.0;
         A[4][1]=-1+4*n2;
         A[5][0]=-4*n2;
         A[5][1]=4*(a-n2);
         A[5][2]=-4*n2;
         A[6][0]=-4*n3;
         A[6][1]=-4*n3;
         A[6][2]=4*(a-n3);
         A[7][0]=4*n3;
         A[7][1]=0.0;
         A[7][2]=4*n1;
         A[8][0]=0.0;
         A[8][1]=4*n3;
         A[8][2]=4*n2;
         A[9][0]=0.0;
         A[9][1]=0.0;
         A[9][2]=-1+4*n3;

FSe[0]=-a*(1-2*a);
FSe[1]=4*n1*a;
FSe[2]=-n1*(1-2*n1);
FSe[3]=4*n1*n2;
FSe[4]=-n2*(1-2*n2);
FSe[5]=4*n2*a;
FSe[6]=4*n3*a;
FSe[7]=4*n1*n2;
FSe[8]=4*n2*n3;
FSe[9]=-n3*(1-2*n3);

for(j=0; j<=9; j++) {

    fprintf(f,"%e %e %e %e %e\n",A[j][0],A[j][1],A[j][2],FSe[j],w1);

 printf("%d %f %f %f %f %f\n", i,A[j][0],A[j][1],A[j][2],FSe[j],w1);
     }

     }
     fclose(f);
```

```
        /*****************  PTYPE620.C ********************/

#include <math.h>
#include <fcntl.h>
#include <stdio.h>
#include <io.h>

main()

 {
 int i,j,fd;
 float n1,n2,n3,w1,A[20][3],FSe[20];
 float a,b,c,d,a1,b1,c1,e;
 FILE *f;

 fd=open("A:DATA620",O_RDWR|O_CREAT|O_TRUNC|O_TEXT,0x0100|0x0080);
 if(fd==-1) creat("A:DATA620",0x0100|0x0080);
 f=fdopen(fd,"r+t");

      for(i=0;  i<=26;  i++) {

          c1=sqrt(0.6);  a1=-sqrt(0.6);b1=0.0;
        a=125;b=200;c=320;d=512;e=729;


       if(i==0) {     n1=n2=n3=a1;        w1=a/e;   }
       if(i==1) {     n1=n2=a1; n3=b1;    w1=b/e;  }
       if(i==2) {     n1=n2=a1; n3=c1;    w1=a/e;   }
       if(i==3) {     n1=n3=a1; n2=0.0;   w1=b/e;  }
       if(i==4) {     n1=a1;  n2=n3=b1;   w1=c/e;  }
       if(i==5) { n1=a1;  n2=b1;  n3=c1;  w1=b/e;  }
       if(i==6) {       n1=n3=a1;n2=c1;   w1=a/e;   }
       if(i==7) {n1=a1;  n2=c1;  n3=b1;   w1=b/e;  }
       if(i==8) {n1=a1;  n2=n3=c1;        w1=a/e;   }
       if(i==9) {n1=b1;  n2=n3=a1;        w1=b/e;  }
      if(i==10) {n1=n3=b1;  n2=a1;        w1=c/e;  }
      if(i==11) {n1=b1;  n2=a1;  n3=c1;   w1=b/e;  }
      if(i==12) {n1=n2=b1;  n3=a1;        w1=c/e;  }
      if(i==13) {n1=n2=n3=b1;             w1=d/e;  }
      if(i==14) {n1=n2=b1;  n3=c1;        w1=c/e;  }
      if(i==15) {n1=b1;  n2=c1;  n3=a1;   w1=b/e;  }
      if(i==16) {n1=n3=b1;  n2=c1;        w1=c/e;  }
      if(i==17) {n1=b1;  n2=n3=c1;        w1=b/e;  }
      if(i==18) {n1=c1;  n2=n3=a1;        w1=a/e;   }
      if(i==19) {n1=c1;  n2=a1;  n3=b1;   w1=b/e;  }
      if(i==20) {n1=n3=c1;  n2=a1;        w1=a/e;   }
      if(i==21) {n1=c1;  n2=b1;  n3=a1;   w1=b/e;  }
      if(i==22) {n1=c1;  n2=n3=b1;        w1=c/e;  }
      if(i==23) {n1=n3=c1;  n2=b1;        w1=b/e;  }
      if(i==24) {n1=n2=c1;  n3=a1;        w1=a/e;   }
      if(i==25) {n1=n2=c1;n3=b1;          w1=b/e;  }
      if(i==26) {n1=n2=n3=c1;             w1=a/e;   }
```

```
A[0][0]=-0.125*(1-n2)*(1-n3)*(-1-2*n1-n2-n3);
A[0][1]=-0.125*(1-n1)*(1-n3)*(-1-n1-2*n2-n3);
A[0][2]=-0.125*(1-n1)*(1-n2)*(-1-n1-n2-2*n3);
A[1][0]=-0.5*n1*(1-n2)*(1-n3);
A[1][1]=-0.25*(1-n1*n1)*(1-n3);
A[1][2]=-0.25*(1-n1*n1)*(1-n2);
A[2][0]=0.125*(1-n2)*(1-n3)*(-1+2*n1-n2-n3);
A[2][1]=-0.125*(1+n1)*(1-n3)*(-1+n1-2*n2-n3);
A[2][2]=-0.125*(1+n1)*(1-n2)*(-1+n1-n2-2*n3);
A[3][0]=0.25*(1-n2*n2)*(1-n3);
A[3][1]=-0.5*n2*(1+n1)*(1-n3);
A[3][2]=-0.25*(1+n1)*(1-n2*n2);
A[4][0]=0.125*(1+n2)*(1-n3)*(-1+2*n1+n2-n3);
A[4][1]=0.125*(1+n1)*(1-n3)*(-1+n1+2*n2-n3);
A[4][2]=-0.125*(1+n1)*(1+n2)*(-1+n1+n2-2*n3);
A[5][0]=-0.5*n1*(1+n2)*(1-n3);
A[5][1]=0.25*(1-n1*n1)*(1-n3);
A[5][2]=-0.25*(1-n1*n1)*(1+n2);
A[6][0]=-0.125*(1+n2)*(1-n3)*(-1-2*n1+n2-n3);
A[6][1]=0.125*(1-n1)*(1-n3)*(-1-n1+2*n2-n3);
A[6][2]=-0.125*(1-n1)*(1+n2)*(-1-n1+n2-2*n3);
A[7][0]=-0.125*(1-n2*n2)*(1-n3);
A[7][1]=-0.5*n2*(1-n1)*(1-n3);
A[7][2]=-0.25*(1-n1)*(1-n2*n2);
A[8][0]=-0.25*(1-n2)*(1-n3*n3);
A[8][1]=-0.25*(1-n1)*(1-n3*n3);
A[8][2]=-0.5*n3*(1-n1)*(1-n2);
A[9][0]=0.25*(1-n2)*(1-n3*n3);
A[9][1]=-0.25*(1+n1)*(1-n3*n3);
A[9][2]=-0.5*n3*(1+n1)*(1-n2);
A[10][0]=0.25*(1+n2)*(1-n3*n3);
A[10][1]=0.25*(1+n1)*(1-n3*n3);
A[10][2]=-0.5*n3*(1+n1)*(1+n2);
A[11][0]=-0.25*(1+n2)*(1-n3*n3);
A[11][1]=0.25*(1-n1)*(1-n3*n3);
A[11][2]=-0.5*n3*(1-n1)*(1+n2);
A[12][0]=-0.125*(1-n2)*(1+n3)*(-1-2*n1-n2+n3);
A[12][1]=-0.125*(1-n1)*(1+n3)*(-1-n1-2*n2+n3);
A[12][2]=0.125*(1-n1)*(1-n2)*(-1-n1-n2+2*n3);
A[13][0]=-0.5*n1*(1-n2)*(1+n3);
A[13][1]=-0.25*(1-n1*n1)*(1+n3);
A[13][2]=0.25*(1-n1*n1)*(1-n2);
A[14][0]=0.125*(1-n2)*(1+n3)*(-1+2*n1-n2+n3);
A[14][1]=-0.125*(1+n1)*(1+n3)*(-1+n1-2*n2+n3);
A[14][2]=0.125*(1+n1)*(1-n2)*(-1+n1-n2+2*n3);
A[15][0]=0.25*(1-n2*n2)*(1+n3);
A[15][1]=-0.5*n2*(1+n1)*(1+n3);
A[15][2]=0.25*(1+n1)*(1-n2*n2);
A[16][0]=0.125*(1+n2)*(1+n3)*(-1+2*n1+n2+n3);
A[16][1]=0.125*(1+n1)*(1+n3)*(-1+n1+2*n2+n3);
A[16][2]=0.125*(1+n1)*(1+n2)*(-1+n1+n2+2*n3);
```

```
A[17][0]=-0.5*n1*(1+n2)*(1+n3);
A[17][1]=0.25*(1-n1*n1)*(1+n3);
A[17][2]=0.25*(1-n1*n1)*(1+n2);
A[18][0]=-0.125*(1+n2)*(1+n3)*(-1-2*n1+n2+n3);
A[18][1]=0.125*(1-n1)*(1+n3)*(-1-n1+2*n2+n3);
A[18][2]=0.125*(1-n1)*(1+n2)*(-1-n1+n2+2*n3);
A[19][0]=-0.25*(1-n2*n2)*(1+n3);
A[19][1]=-0.5*n2*(1-n1)*(1+n3);
A[19][2]=0.25*(1-n1)*(1-n2*n2);


 FSe[0]=0.125*(1-n1)*(1-n2)*(1-n3)*(-2-n1-n2-n3);
 FSe[1]=0.25*(1-n1*n1)*(1-n2)*(1-n3);
 FSe[2]=0.125*(1+n1)*(1-n2)*(1-n3)*(-2+n1-n2-n3);
 FSe[3]=0.25*(1+n1)*(1-n2*n2)*(1-n3);
 FSe[4]=0.125*(1+n1)*(1+n2)*(1-n3)*(-2+n1+n2-n3);
 FSe[5]=0.25*(1-n1*n1)*(1+n2)*(1-n3);
 FSe[6]=0.125*(1-n1)*(1+n2)*(1-n3)*(-2-n1+n2-n3);
 FSe[7]=0.25*(1-n1)*(1-n2*n2)*(1-n3);
 FSe[8]=0.25*(1-n1)*(1-n2)*(1-n3*n3);
 FSe[9]=0.25*(1+n1)*(1-n2)*(1-n3*n3);
FSe[10]=0.25*(1+n1)*(1+n2)*(1-n3*n3);
FSe[11]=0.25*(1-n1)*(1+n2)*(1-n3*n3);
FSe[12]=0.125*(1-n1)*(1-n2)*(1+n3)*(-2-n1-n2+n3);
FSe[13]=0.25*(1-n1*n1)*(1-n2)*(1+n3);
FSe[14]=0.125*(1+n1)*(1-n2)*(1+n3)*(-2+n1-n2+n3);
FSe[15]=0.25*(1+n1)*(1-n2*n2)*(1+n3);
FSe[16]=0.125*(1+n1)*(1+n2)*(1+n3)*(-2+n1+n2+n3);
FSe[17]=0.25*(1-n1*n1)*(1+n2)*(1+n3);
FSe[18]=0.125*(1-n1)*(1+n2)*(1+n3)*(-2-n1+n2+n3);
FSe[19]=0.25*(1-n1)*(1-n2*n2)*(1+n3);


  for(j=0; j<=19; j++) {

  fprintf(f,"%e %e %e %e %e\n",A[j][0],A[j][1],A[j][2],FSe[j],w1);

  printf("%d %f %f %f %f %f\n",i,A[j][0],A[j][1],A[j][2],FSe[j],w1);
 }
}
```

```
/********************  PTYPE627.C   ***************/

#include <math.h>
#include <fcntl.h>
#include <stdio.h>
#include <io.h>
#include <dos.h>

main()
 {
 int j, i, fd;
 float n1, n2, n3, w1, A[27][3], FSe[27], c1, a1, b1;
 float a, b, c, d, e;
 FILE *f;

 fd=open("A:DATA627",O_RDWR|O_CREAT|O_TRUNC|O_TEXT,0x0100|0x0080);
 if(fd==-1) creat("A:DATA627",0x0100|0x0080);
 f=fdopen(fd,"r+t");

 for(i=0; i<=26; i++) {

      c1=sqrt(0.6); a1=-sqrt(0.6);b1=0.0;
     a=125;b=200;c=320;d=512;e=729;

    if(i==0) {    n1=n2=n3=a1;       w1=a/e; }
    if(i==1) {    n1=n2=a1; n3=b1;   w1=b/e; }
    if(i==2) {    n1=n2=a1; n3=c1;   w1=a/e; }
    if(i==3) {    n1=n3=a1; n2=0.0;  w1=b/e; }
    if(i==4) {    n1=a1; n2=n3=b1;   w1=c/e; }
    if(i==5) { n1=a1; n2=b1; n3=c1;  w1=b/e; }
    if(i==6) {      n1=n3=a1;n2=c1;  w1=a/e; }
    if(i==7) {n1=a1; n2=c1; n3=b1;   w1=b/e; }
    if(i==8) {n1=a1; n2=n3=c1;       w1=a/e; }
    if(i==9) {n1=b1; n2=n3=a1;       w1=b/e; }
   if(i==10) {n1=n3=b1; n2=a1;       w1=c/e; }
   if(i==11) {n1=b1; n2=a1; n3=c1;   w1=b/e; }
   if(i==12) {n1=n2=b1; n3=a1;       w1=c/e; }
   if(i==13) {n1=n2=n3=b1;           w1=d/e; }
   if(i==14) {n1=n2=b1; n3=c1;       w1=c/e; }
   if(i==15) {n1=b1; n2=c1; n3=a1;   w1=b/e; }
   if(i==16) {n1=n3=b1; n2=c1;       w1=c/e; }
   if(i==17) {n1=b1; n2=n3=c1;       w1=b/e; }
   if(i==18) {n1=c1; n2=n3=a1;       w1=a/e; }
   if(i==19) {n1=c1; n2=a1; n3=b1;   w1=b/e; }
   if(i==20) {n1=n3=c1; n2=a1;       w1=a/e; }
   if(i==21) {n1=c1; n2=b1; n3=a1;   w1=b/e; }
   if(i==22) {n1=c1; n2=n3=b1;       w1=c/e; }
   if(i==23) {n1=n3=c1; n2=b1;       w1=b/e; }
   if(i==24) {n1=n2=c1; n3=a1;       w1=a/e; }
   if(i==25) {n1=n2=c1;n3=b1;        w1=b/e; }
   if(i==26) {n1=n2=n3=c1;           w1=a/e; }
```

```
A[0][0]=0.125*(-1+2*n1)*n2*(1-n2)*n3*(1-n3);
A[0][1]=0.125*n1*(1-n1)*(-1+2*n2)*n3*(1-n3);
A[0][2]=0.125*n1*(1-n1)*n2*(1-n2)*(-1+2*n3);
A[1][0]=-0.25*(-1+2*n1)*n2*(1-n2)*(1-n3*n3);
A[1][1]=-0.25*n1*(1-n1)*(-1+2*n2)*(1-n3*n3);
A[1][2]=-0.5*n1*(1-n1)*n2*(1-n2)*n3;
A[2][0]=-0.125*(-1+2*n1)*n2*(1-n2)*n3*(1+n3);
A[2][1]=-0.125*n1*(1-n1)*(-1+2*n2)*n3*(1+n3);
A[2][2]=0.125*n1*(1-n1)*n2*(1-n2)*(1+2*n3);
A[3][0]=-0.25*(-1+2*n1)*(1-n2*n2)*n3*(1-n3);
A[3][1]=-0.5*n1*(1-n1)*n2*n3*(1-n3);
A[3][2]=-0.25*n1*(1-n1)*(1-n2*n2)*(-1+2*n3);
A[4][0]=0.5*(-1+2*n1)*(1-n2*n2)*(1-n3*n3);
A[4][1]=n1*(1-n1)*n2*(1-n3*n3);
A[4][2]=n1*(1-n1)*(1-n2*n2)*n3;
A[5][0]=0.25*(-1+2*n1)*(1-n2*n2)*n3*(1+n3);
A[5][1]=0.5*n1*(1-n1)*n2*n3*(1+n3);
A[5][2]=-0.25*n1*(1-n1)*(1-n2*n2)*(1+2*n3);
A[6][0]=-0.125*(-1+2*n1)*n2*(1+n2)*n3*(1-n3);
A[6][1]=0.125*n1*(1-n1)*(1+2*n2)*n3*(1-n3);
A[6][2]=-0.125*n1*(1-n1)*n2*(1+n2)*(-1+2*n3);
A[7][0]=0.25*(-1+2*n1)*n2*(1+n2)*(1-n3*n3);
A[7][1]=-0.25*n1*(1-n1)*(1+2*n2)*(1-n3*n3);
A[7][2]=0.5*n1*(1-n1)*n2*(1+n2)*n3;
A[8][0]=0.125*(-1+2*n1)*n2*(1+n2)*n3*(1+n3);
A[8][1]=-0.125*n1*(1-n1)*(1+2*n2)*n3*(1+n3);
A[8][2]=-0.125*n1*(1-n1)*n2*(1+n2)*(1+2*n3);
A[9][0]=-0.5*n1*n2*(1-n2)*n3*(1-n3);
A[9][1]=-0.25*(1-n1*n1)*(-1+2*n2)*n3*(1-n3);
A[9][2]=-0.25*(1-n1*n1)*n2*(1-n2)*(-1+2*n3);
A[10][0]=n1*n2*(1-n2)*(1-n3*n3);
A[10][1]=0.5*(1-n1*n1)*(-1+2*n2)*(1-n3*n3);
A[10][2]=(1-n1*n1)*n2*(1-n2)*n3;
A[11][0]=0.5*n1*n2*(1-n2)*n3*(1+n3);
A[11][1]=-0.25*(1-n1*n1)*(-1+2*n2)*n3*(1+n3);
A[11][2]=-0.25*(1-n1*n1)*n2*(1-n2)*(1+2*n3);
A[12][0]=n1*(1-n2*n2)*n3*(1-n3);
A[12][1]=(1-n1*n1)*n2*n3*(1-n3);
A[12][2]=0.5*(1-n1*n1)*(1-n2*n2)*(-1+2*n3);
A[13][0]=-2*n1*(1-n2*n2)*(1-n3*n3);
A[13][1]=-2*(1-n1*n1)*n2*(1-n3*n3);
A[13][2]=-2*(1-n1*n1)*(1-n2*n2)*n3;
A[14][0]=-n1*(1-n2*n2)*n3*(1+n3);
A[14][1]=-(1-n1*n1)*n2*n3*(1+n3);
A[14][2]=0.5*(1-n1*n1)*(1-n2*n2)*(1+2*n3);
A[15][0]=0.5*n1*n2*(1+n2)*n3*(1-n3);
A[15][1]=-0.25*(1-n1*n1)*(1+2*n2)*n3*(1-n3);
A[15][2]=0.25*(1-n1*n1)*n2*(1+n2)*(-1+2*n3);
A[16][0]=-n1*n2*(1+n2)*(1-n3*n3);
A[16][1]=0.5*(1-n1*n1)*(1+2*n2)*(1-n3*n3);
A[16][2]=-(1-n1*n1)*n2*(1+n2)*n3;
```

```
A[17][0]=-0.5*n1*n2*(1+n2)*n3*(1+n3);
A[17][1]=0.25*(1-n1*n1)*(1+2*n2)*n3*(1+n3);
A[17][2]=0.25*(1-n1*n1)*n2*(1+n2)*(1+2*n3);
A[18][0]=0.125*(1+2*n1)*n2*(1-n2)*n3*(1-n3);
A[18][1]=-0.125*n1*(1+n1)*(-1+2*n2)*n3*(1-n3);
A[18][2]=-0.125*n1*(1+n1)*n2*(1-n2)*(-1+2*n3);
A[19][0]=-0.25*(1+2*n1)*n2*(1-n2)*(1-n3*n3);
A[19][1]=0.25*n1*(1+n1)*(-1+2*n2)*(1-n3*n3);
A[19][2]=0.5*n1*(1+n1)*n2*(1-n2)*n3;
A[20][0]=-0.125*(1+2*n1)*n2*(1-n2)*n3*(1+n3);
A[20][1]=0.125*n1*(1+n1)*(-1+2*n2)*n3*(1+n3);
A[20][2]=-0.125*n1*(1+n1)*n2*(1-n2)*(1+2*n3);
A[21][0]=-0.25*(1+2*n1)*(1-n2*n2)*n3*(1-n3);
A[21][1]=0.5*n1*(1+n1)*n2*n3*(1-n3);
A[21][2]=0.25*n1*(1+n1)*(1-n2*n2)*(-1+2*n3);
A[22][0]=0.5*(1+2*n1)*(1-n2*n2)*(1-n3*n3);
A[22][1]=-n1*(1+n1)*n2*(1-n3*n3);
A[22][2]=-n1*(1+n1)*(1-n2*n2)*n3;
A[23][0]=0.25*(1+2*n1)*(1-n2*n2)*n3*(1+n3);
A[23][1]=-0.5*n1*(1+n1)*n2*n3*(1+n3);
A[23][2]=0.25*n1*(1+n1)*(1-n2*n2)*(1+2*n3);
A[24][0]=-0.125*(1+2*n1)*n2*(1+n2)*n3*(1-n3);
A[24][1]=-0.125*n1*(1+n1)*(1+2*n2)*n3*(1-n3);
A[24][2]=0.125*n1*(1+n1)*n2*(1+n2)*(-1+2*n3);
A[25][0]=0.25*(1+2*n1)*n2*(1+n2)*(1-n3*n3);
A[25][1]=0.25*n1*(1+n1)*(1+2*n2)*(1-n3*n3);
A[25][2]=-0.5*n1*(1+n1)*n2*(1+n2)*n3;
A[26][0]=0.125*(1+2*n1)*n2*(1+n2)*n3*(1+n3);
A[26][1]=0.125*n1*(1+n1)*(1+2*n2)*n3*(1+n3);
A[26][2]=0.125*n1*(1+n1)*n2*(1+n2)*(1+2*n3);

      FSe[0]=-n1*(1-n1)*n2*(1-n2)*n3*(1-n3);
      FSe[1]=2*n1*(1-n1)*n2*(1-n2)*(1-n3*n3);
      FSe[2]=n1*(1-n1)*n2*(1-n2)*n3*(1+n3);
      FSe[3]=2*n1*(1-n1)*(1-n2*n2)*n3*(1-n3);
      FSe[4]=-4*n1*(1-n1)*(1-n2*n2)*(1-n3*n3);
      FSe[5]=-2*n1*(1-n1)*(1-n2*n2)*n3*(1+n3);
      FSe[6]=n1*(1-n1)*n2*(1+n2)*n3*(1-n3);
      FSe[7]=-2*n1*(1-n1)*n2*(1+n2)*(1-n3*n3);
      FSe[8]=-n1*(1-n1)*n2*(1+n2)*n3*(1+n3);
      FSe[9]=2*(1-n1*n1)*n2*(1-n2)*n3*(1-n3);
     FSe[10]=-4*(1-n1*n1)*n2*(1-n2)*(1-n3*n3);
     FSe[11]=-2*(1-n1*n1)*n2*(1-n2)*n3*(1+n3);
     FSe[12]=-4*(1-n1*n1)*(1-n2*n2)*n3*(1-n3);
     FSe[13]=8*(1-n1*n1)*(1-n2*n2)*(1-n3*n3);
     FSe[14]=4*(1-n1*n1)*(1-n2*n2)*n3*(1+n3);
     FSe[15]=-2*(1-n1*n1)*n2*(1+n2)*n3*(1-n3);
     FSe[16]=4*(1-n1*n1)*n2*(1+n2)*(1-n3*n3);
     FSe[17]=2*(1-n1*n1)*n2*(1+n2)*n3*(1+n3);
     FSe[18]=n1*(1+n1)*n2*(1-n2)*n3*(1-n3);
     FSe[19]=-2*n1*(1+n1)*n2*(1-n2)*(1-n3*n3);
     FSe[20]=-n1*(1+n1)*n2*(1-n2)*n3*(1+n3);
```

```
            FSe[21]=-2*n1*(1+n1)*(1-n2*n2)*n3*(1-n3);
            FSe[22]=4*n1*(1+n1)*(1-n2*n2)*(1-n3*n3);
            FSe[23]=2*n1*(1+n1)*(1-n2*n2)*n3*(1+n3);
            FSe[24]=-n1*(1+n1)*n2*(1+n2)*n3*(1-n3);
            FSe[25]=2*n1*(1+n1)*n2*(1+n2)*(1-n3*n3);
            FSe[26]=n1*(1+n1)*n2*(1+n2)*n3*(1+n3);

            for(j=0; j<=26; j++) {

fprintf(f,"%e %e %e %e %e\n",A[j][0],A[j][1],A[j][2],FSe[j],wi);

printf("%d %f %f %f %f %f\n",i,A[j][0],A[j][1],A[j][2],FSe[j],wi);
                        }
          }

          fclose(f);
```

BIBLOGRAPHIE

1 G. de MARSILY Hydrogéologie quantitative
MASSON 1981

2 GOURI DHATT et TOUZOT Une présentation de la méthode des éléments finis
Les presses de l'université LAVAL 1981

3 O.C. ZIENKIEWICZ la méthode des éléments finis appliquée à l'art de l'ingénieur
PARIS 1973

4 TURBO C REFERENCE GUIDE
THE FASTES, MOST EFFICIENT AND EASY TO USE COMPILER
I.B.M VERSION 1987

5 TURBO C USER'S GUIDE
I.B.M VERSION 1987

6 JEAN-MICHEL DRAPPIER ET ANNIE MAUFFREY
C PAR L'EXEMPLE.
INFORMATIQUE ET ENTREPRISE 1986

7 B. W. KERNIGHAN D. M. RITCHIE

LE LANGAGE C

MASSON 1983

8 DANIEL HILLER

L'EAU ET LE SOL PRINCIPES ET PROCESSUS PHYSIQUES

SCIENCE ET TECHNIQUE 1986

9 A. NENONENE

PROJET DE FIN D'ETUDE E.P.T 1986

10 J. TINSLEY ODEN ET GRAHAM F. CAREY

FINITE ELEMENTS

1- INTRODUCTION

2- A SECOND CAUSE

3- COMPUTIONNAL ASPECTS

4- MATHEMATICAL ASPECTS

5- SPECIAL PROBLEMS IN SOLID FLUID MECHANICS

DANS LA COLLECTION : THE TEXAS FINITE ELEMENT SERIES

11 HENRI VARCOLLIER

Exposés et exemples d'application de calcul tensoriel et de calcul matriciel à l'usage des Ingénieurs et physiciens.

Troisième édition: PRESSES UNIVERSITAIRE DE FRANCE

12 PIERRE ARNAUD RIVIART

Les méthodes d'éléments finis en mécanique des fluides

COLLECTION DE LA DIRECTION DES ETUDES ET RECHERCHES

D'ELECTRICITE EN FRANCE 1981

13 BATHE K.J. and WILSON E.L.

NUMERICAL METHODS IN FINITE ELEMENT ANALYSIS

PRENTICE HALL, ENGLEWOOD CLIFFS, N.S. 1976

14 TECHNICAL REPORT:

Par KIRALY 1979 du centre de recherche CEDRA (SUISSE)

15 G.CASTANY

TRAITE PRATIQUE DES EAUX SOUTERRAINES PARIS 1967