ECOLE POLYTECHNIQUE DE THIES
DEPARTEMENT DE GENIE MECANIQUE



# PROJET DE FIN D'ETUDES

SUJET : PROGRAMME DE SIMULATION NUMERIQUE D'ECOULEMENTS EN RESEAUX SOUS PRESSION A PREPROCESSEUR ET POST PROCESSEUR GRAPHIQUES

AUTEUR : KAMBIKA ZODI

DIRECTEUR : AMADOU SARR

ANNEE SCOLAIRE 1989-1990

A mon pére,

A ma mére,

A ma famille,

A mes amis et à tous ce qui me sont chers,

je dédie ce travail.

## REMERCIEMENTS

A Mr Amadou Sarr ,mon directeur de projet,
professeur à l'école polytechinique de THIES.
A tous ce qui m'ont soutenu dans mes études,
qu'ils trouvent ici ma reconnaissance.

## SOMMAIRE

La présente étude a pour but,la mise au point d'un programme interactif modulaire de simulation numérique des écoulements dans les réseaux hydrauliques en régime permanent comportant un interface graphique utilisateur.L'approche adoptée consiste à développer les équations de base de la mécanique des fluides (équation de continuité,équation d'énergie)et a les appliquer dans un réseau de conduites pouvant comporter divers singularités (pompes,réservoir,etc,..) afin de trouver des algorithmes adéquats à la programmation.Un réseau hydraulique peut avoir plusieurs configurations(maillé,étoilé,ramifié) ;ainsi des modèles mathématiques seront développés dans cette étude pour tenir compte de ces particularités.

Dans cette étude ,il sera présenté ce qui suit:

-Les développements des équations de base

-La géométrie du système de conduites

-Les modèles mathématiques utilisées

-L'utilisation et la description du programme

-Le listing du programme

## TABLE DES MATIERES

# INTRODUCTION

La simulation numérique des écoulements dans les réseaux hydrauliques est d'une importance manifeste;en effet on rencontre les résaux hydrauliques dans plusieurs applications comme la distribution d'eau urbaine,les systèmes de distribution d'eaux usées en hydraulique appliquée ,on le rencontre aussi dans les systèmes de distributions de certain fluides comme par exemple les carburants dans les oléoducs ou l'air dans les conduites de climatisation,etc...

La connaissance des débits et pressions présente un intérêt vital lors de la conception des ces systèmes,elle permet de déterminer les dimensions à donner aux éléments ainsi que le choix des éléments de contrôle et de régulation des variables d'écoulements pour un fonctionnement optimal du système.

Ce projet étudie le régime permanent dans les réseaux,régime dans lequel les variables d'écoulements restent inchangées par rapport au temps.Le modèle mathématique régissant ce régime est connu depuis fort longtemps,ce modèle est analogue à celui des calculs des réseaux électriques connue sous l'appellation des équations de KIRCHOFF . La résolution de ces équations pour des grands résaux exige l'utilisation des méthodes numériques adéquates .Cette étude consiste à mettre au point un programme de calculs des ces réseaux en régime permanent par un assemblage des différentes méthodes proposées dans la littérature (Hardy-Cross,Linéarisation,etc...).

## CHAPITRE 1 : ETABLISSEMENTS DES EQUATIONS DE BASE

### 1.1 LES REGIMES D'ECOULEMENTS

Le régime d'écoulement est déterminé à l'aide d'un nombre sans dimension appelé nombre de REYNOLDS ,il est défini comme suit:

$$Re = \frac{D*V}{mhu} \qquad (1.1)$$

Ce nombre comme son nom l'indique fut introduit pour la première fois par REYNOLDS en 1833 en réalisant une expérience sur les régimes d'écoulements des fluides visqueux en conduite circulaire

D:diamètre du tube en m

V:vitesse de l'écoulement en m/s

mhu:viscosité cinématique du fluide en $m^2/s$

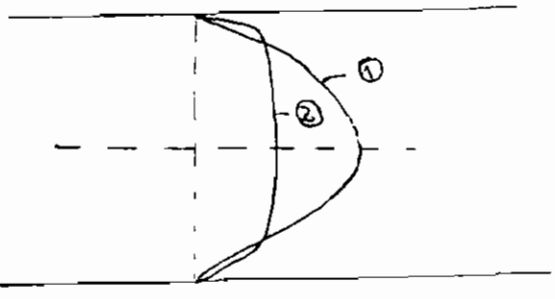
L'expérience de Reynolds consistait à injecter un colorant dans une conduite dans laquelle circulait un fluide ,pour visualiser la trajectoire des particules-fluide en régime permanent ou les lignes d'émission en régime transitoire

Généralement pour un nombre de Reynolds inférieur à 2000 on a un écoulement laminaire et au delà un regime turbulent,toutefois pour des nombres de Reynolds proche de 2000,on observe une zone

de transition laminaire-turbulent

Ce nombre mesure l'importance relative des forces d'inertie par rapport aux forces de viscosité

En observant les profils de vitesse à travers une section droite de l'écoulement ,on remarque que pour le régime turbulent,le profil est plus plat avec un gradient de vitesse plus important au voisinage des parois et que la vitesse moyenne est inférieure par rapport au régime laminaire




① régime laminaire
② régime turbulent

Fig 1.1

Quand le nombre de Reynolds augmente on observe

-une augmentation des frottements aux parois due au gradient de la vitesse à la paroi

$$\left(\frac{dV}{dr}\right)_{r=D/2} \qquad (1.2)$$

-une augmentation des frottements entre les particules

fluides(dissipation de vitesse) due aux mouvements de fluctuations plus importants

La turbulence est un phénomène aléatoire et diffusif

## 1.2 EQUATION DE CONTINUITE

Elle exprime la conservation de la masse d'un domaine fluide,elle s'écrit

$$\frac{dm}{dt} = \int \left(\frac{\delta\sigma}{\delta t} + (\sigma \vec{V})\right) dw = 0 \qquad (1.3)$$

sous forme intégrale

Elle s'écrit aussi

$$\frac{d\sigma}{dt} + \sigma(\vec{\nabla} \vec{V}) = 0 \qquad (1.4)$$

qui est la forme différentielle

Dans les deux cas D représente le domaine fluide considéré

$\sigma$ : masse volumique du fluide

V : vitesse à travers une section

Appliquons l'équation de continuité avec les hypothèses suivantes

-Écoulement d'un fluide incompressible

-Écoulement stationnaire

-Écoulement unidimensionnel

et considérons un domaine fluide cylindrique à section droite

(dans une conduite circulaire)

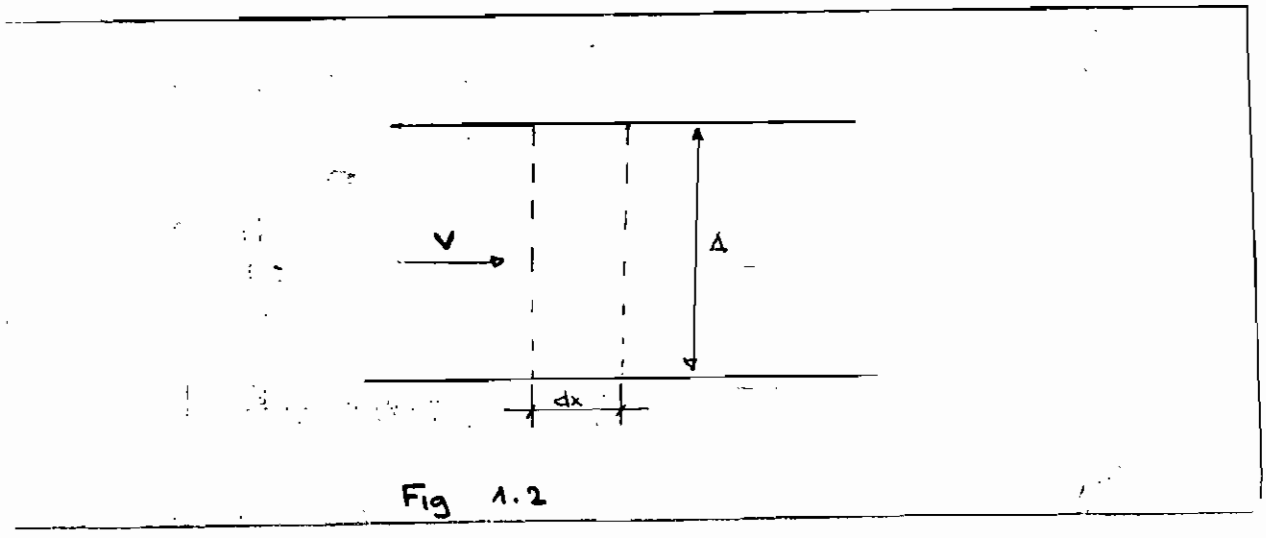



Fig 1.2

L'équation de continuité s'écrit dans ce cas

$$\sigma \int \vec{V} dw = \sigma \int \vec{V}.\vec{n}\, dA = \sigma VA \quad (1.5)$$

A étant l'aire de la section

VA est le débit volume à travers la conduite

$\sigma$VA est le débit masse à travers la conduite

Entre deux sections 1 et 2 de la conduite, nous pouvons écrire la relation de continuité comme suit

$$\sigma V_1 A_1 = \sigma V_2 A_2 \quad \text{ou} \quad A_1 V_1 = A_2 V_2 \quad (1.6)$$

## 1.3 ÉQUATION D'ENERGIE

Elle exprime la conservation d'énergie qui dans une conduite

présente trois composantes essentielles:

-Énergie cinétique

-Énergie potentielle

-Énergie de pression

**Théorème**

La variation de l'énergie cinétique d'un tube de courant (cylindre compris entre deux sections droites de la conduite et dont les surfaces latérales sont en chaque points tangentes à la vitesse)est égale au travail des forces intérieures et extérieures.

Considérons les hypothèses émises ci-haut et appliquons le théorème de l'énergie cinétique sur le domaine fluide de la figure 1.3

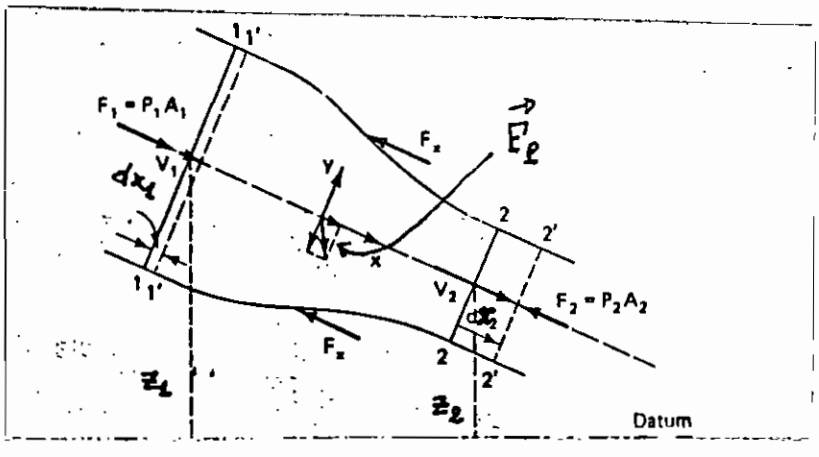



Fig 1.3

$A_1$ , $A_2$ : Aires de sections 1 et 2

$V_1$ , $V_2$ : Vitesses en 1 et 2

$z_1$ , $z_2$ : Hauteurs géométriques par rapport à un niveau arbitraire(datum)

$P_1$ ,$P_2$ : Pressions en 1 et 2

$\sigma$ : masse volumique du fluide

$\Gamma$ : poids volumique du fluide

g : accélération de la pesanteur

Q : débit-volume

M : masse du fluide

-Travail des forces des pressions durant dt

section 1

$$P_1A_1dx_1 = P_1A_1V_1dt = P_1Qdt \qquad (1.7)$$

section 2

$$P_2A_2dx_2 = P_2A_2V_2dt = P_2Qdt \qquad (1.8)$$

-Tavail des forces massiques

$$gA_1V_1dt(z_1-z_2) = gQdt(z_1-z_2) \qquad (1.9)$$

-Variation de l'énergie cinétique

$$1/2MV_2^2 - 1/2MV_1^2 = 1/2\sigma A_1V_1dt(V_2^2-V_1^2) \qquad (1.10)$$

$$= 1/2\sigma Qdt(V_2^2-V_1^2)$$

Nous avons donc(théorème de l'énergie cinétique)

$$P_1Qdt-P_2Qdt + gQdt(z_1-z_2) + W_{frott} = 1/2\ Qdt(V_2^2-V_1^2) \qquad (1.11)$$

Par unité de poids (division par gQdt) on trouve

$$\frac{P1}{\Gamma} - \frac{P2}{\Gamma} + z1-z2 + hf = 1/(2g)(V_2^2-V_1^2) \qquad (1.12)$$

$$\frac{P_1}{\Gamma} + \frac{V_1^2}{2g} + z_1 = \frac{P_1}{\Gamma} + \frac{V_2^2}{2g} + z_2 + h_f \qquad (1.13)$$

$h_f = W_{frott}/\sigma gQdt$ représente le travail des forces de frottements par unité de poids,elle a les dimensions d'une hauteur(m)

Cette équation est appéllée par abus de langage:

'équation de Bernouilli avec pertes des charges' ou

'équation de Bernouilli pour fluides réels'

## 1.4 EQUATIONS DE PERTES DE CHARGES DANS UNE CONDUITE

Il s'agit de modéliser les pertes des charges dues aux frottements ,les formules utilisées sont soit empiriques ou semi-empiriques à cause de la turbulence qui est le régime constamment rencontré dans les écoulements en conduite

### 1.4.1 Formule de DARCY-WEISBASH

Cette formule pratique est établie par ses deux auteurs à partir des observations expérimentales suivantes

-indépendance de $h_f$ par rapport à la pression absolue

-linéarité de $h_f$ par rapport à la longueur L

-proportionnalité de $h_f$ au diamètre D à une puissance négative ($D^{-\alpha}$)

-proportionnalité de $h_f$ vis à vis de la rugosité si l'écoulement est turbulent

$$h_f = f(L/D)*V^2/(2g) \qquad (1.14)$$

f étant le coefficient de frottement(sans dimension)

### 1.4.2 Coefficient de frottement (f)

**-Écoulement laminaire**

Considérons une conduite horizontale de section constante

$P_a, P_b$ : pressions aux sections 1 et 2

$\mu$ : viscosité dynamique du fluide

mhu : viscosité cinématique du fluide

$$hf = \frac{P_a - P_b}{\Gamma} \qquad (1.15)$$

$V = D^2/(32mhuL)*(P_a-P_b)$ (Formule de Hagen-Poiseuille)

$= D^2/(32mhuL)*$ hf (1.16)

$hf = (32\mu/D)*(L/D)*mhu$ (1.17)

en identifiant avec la formule de Darcy-Weisbash on trouve

$$f = \frac{64}{mhuD}\mu = \frac{64mhu}{VD} = \frac{64}{Re} \qquad (1.18)$$

En écoulement laminaire f ne depend que du nombre de Reynolds et suit une loi très simple

**-Écoulement turbulent**

Formule de Colebrook

Soit une conduite circulaire de rayon r

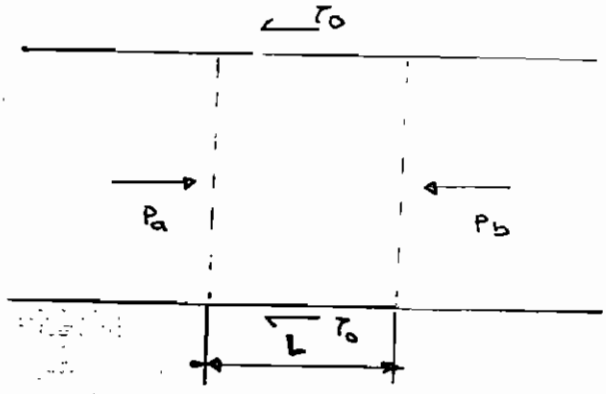



Fig 1.4

L'équation exprimant l'équilibre des forces en régime permanent sur le volume de contrôle (accélération nulle) s'écrit

$$F = P \qquad (1.19)$$

avec P : force de pression

$$P = \Delta p \pi r_o^2 \qquad (1.20)$$

F : Force des frottements visqueux(aux parois)

$$F = \tau_o 2\pi r_o L \qquad (1.21)$$

$\Delta p$ est la différence de pression entre la section 1 et la section 2

$\tau_o$ est la contrainte de cisaillement

$p\pi r_o = \tau_o 2\pi r_o L$ entraîne $\tau_o = \frac{p}{L} * \frac{r_o}{2}$ (1.22)

Cette équation est valable aussi bien pour un régime laminaire que pour un régime turbulent

Considérons l'équation de Darcy-Weibash

$h_f = f(L/2r_o)*v^2/(2g)$ (1.23) v:vitesse moyenne sur la section

$$p = \Gamma h_f = f(L/2r_o)* v^2/2 \qquad (1.24)$$

en éliminant p entre (1.22) et (1.24) on trouve

$$\sqrt{(\tau_o/\sigma)} = \sqrt{(f/8)}*v \qquad (1.25)$$

pour trouver la vitesse moyenne utilisons le profil logarithmique de Prandtl en l'intégrant sur une section droite

$$\frac{u(y)}{u_*} = \frac{1}{A} + \ln\frac{u_* y}{mhu} + B \qquad (1.26)$$

$u(y)$ : vitesse en y ($0<y<2r_o$)

$u_*$ : vitesse de frottement

A et B sont des constantes expérimentales

plaque mince A=0.417 ,B=5.84

conduite lisse A=0.40 ,B=5.5

en remplaçant la valeur de v dans (1.26) on trouve

$1/\sqrt{f} = A_s + B_s \ln(Re\sqrt{f})$ (1.27)

$A_s, B_s$ :constantes expérimentales

En utilisant les données de Nikuradse pour les conduites lisses on a

$$1/\sqrt{f} = 0.86\ln(Re\sqrt{f}) - 0.8 \qquad (1.28)$$

Pour les conduites rugueuses en régime turbulent on aura

$$1/\sqrt{f} = F_2(m,e/D) + B_r\ln(e/D) \qquad (1.29)$$

$F_2$ étant en général une constante pour une forme et une distribution de rugosité donnée

e/D est la rugosité relative du matériau de la conduite

Pour les courbes de Nikuradse cette équation devient

$$1/\sqrt{f} = 1.14 - 0.86\ln(e/D) \qquad (1.30)$$

En additionnant les deux équations on retrouve la formule de Colebrook qui s'écrit

$$1/\sqrt{f} = -0.86\ln\left(\frac{e/D}{3.7} + \frac{2.51}{Re\sqrt{f}}\right) \qquad (1.31)$$

En utilisant quelques constatations expérimentales on peut utiliser les formes réduites de cette équation qui sont

Conduite hydrauliquement lisse

$$1/\sqrt{f} = -0.86\ln(2.51/Re\sqrt{f}) \qquad (1.32)$$ (indépendant de e/D)

Conduite hydrauliquement rugueuse

$$1/\sqrt{f} = -0.86\ln(3.7D/e) \qquad (1.33)$$

Le coefficient de frottement peut également se trouver en utilisant le diagramme de L.F. MOODY

### 1.4.2 FORMULE DE PERTES DE CHARGES EMPIRIQUES

Ces formules donnent des relations explicites entre le coefficient de frottement et les pertes de charges contrairement à la formule de Darcy-Weisbash qui exige une procédure itérative

pour les calculs pratiques.

**FORMULE DE HAZEN-WILLIAMS**

Cette formule est très répandue aux Etats-Unis dans le calcul des systèmes de distribution d'eau.

Elle est considérée valide pour des conduites de diamètres supérieurs à 5 cm et des vitesses moyennes inférieures à 3 m/s

$$v = 0.85 C_{HW} R_h^{0.63} (hf/L)^{0.54} \qquad (1.34)$$

$C_{HW}$ : coefficient de Hazen-Williams

$R_h$ : rayon hydraulique qui est égal au rapport de l'aire de la section sur son périmètre mouillé (pour une conduite circulaire $R_h = D/4$)

Le coefficient de Hazen-Williams n'est pas une fonction des conditions d'écoulement(nombre de Reynolds);sa valeur varie de 140 pour les matériaux lisses et bien alignés à 80 pour les conduites à surface très irrégulières.

**FORMULE DE MANNING**

C'est une formule initialement développée et couramment utilisée dans l'étude des canaux.Elle est aussi utilisée en écoulement dans les conduites.

$$v = 1/n R_h^{2/3} (h_f/L)^{1/2} \qquad (1.35)$$

n est le coefficient de Manning pour différents matériaux.

### 1.4.3 PERTES DES CHARGES SINGULIÈRES

Elles expriment les pertes de charges causées par des irrégularités locales dans la section d'écoulement

Leurs valeurs par unité de poids traversant la conduite sont

exprimées par une hauteur de fluide égale au rapport de l'énergie dissipée pour traverser une singularité sur le poids du fluide

Les phénomènes qui sont à la base de ces pertes de charges sont

-Changement de la section de l'écoulement du tube de courant

-Mouvements secondaires(décollement,...)

-accélérations locales

On peut écrire, entre deux sections 1 et 2 délimitant une singularité,la relation suivante:

$$H_{12} = (\Delta E_{12})/\sigma g Q_v \Delta t \qquad (1.36)$$

$\Delta E_{12}$ étant l'énergie dissipée durant $\Delta t$

$Q_v$ le débit du fluide traversant les deux sections

Coefficient de pertes de charges singulières

On définit,un coefficient sans dimension,appelé coefficient de pertes de charges singulières comme étant le rapport de la perte de charge sur le niveau de fluide correspondant à la pression dynamique($v^2/2g$)

On exprime également ce coefficient par rapport à la pression dynamique à l'amont de l'obstacle,mais on peut le referencier également à la pression dynamique à l'aval et aussi à deux coefficients ,l'un correspondant à la pression dynamique à l'amont et l'autre à la pression dynamique à l'aval.

$$h_l = KV_1^2/2g \qquad \text{amont} \qquad (1.37)$$

$$hl = K'V_1^2/2g \qquad \text{aval} \qquad (1.38)$$

$$hl = K_1V_1^2/2g + K_2V_2^2/2g \qquad \text{amont+aval} \qquad (1.39)$$

On appelle Longueur de conduite équivalente à une

singularité, la longueur de conduite, $L_e$, qui aurait provoquée la même perte de charge que la singularité pour le débit considéré

C'est un moyen très pratique utilisé en calcul numérique

En considérant la formule de Darcy-Weibash on trouve pour $L_e$

$$fL_eV^2/2gD = KV^2/2g \qquad L_e = KD/f \quad (1.40)$$

## CHAPITRE 2 DESCRIPTION DES ELEMENTS GEOMETRIQUES DU RESEAU

Un réseau hydraulique est un assemblage de plusieurs conduites comportant également plusieurs autres composantes que nous définirons dans ce chapitre.

Un réseau hydraulique peut être maillé,étoilé,ramifié ou encore une combinaison de ces différentes configurations.

### 2.1 GEOMETRIE DU SYSTEME DES CONDUITES

**-Noeud de jonction:**

C'est le point de connexion de deux ou plusieurs conduites.
Un débit extérieur peut être injecté ou retiré (demande du noeud) à un noeud de jonction

**-Noeud à pression fixe:**

C'est un noeud où la pression est connue et a une valeur fixe
exemple:-connexion à un reservoir,à une source.

-connexion à une conduite large de pression connue.

**-Boucle élémentaire**

C'est un circuit fermé de conduites ne contenant aucun autre.Le nombre de noeuds d'une boucle élémentaire est égal au nombre de conduites formant cette boucle.Une boucle élémentaire ne contient que des noeuds de jonction.

**-Boucle extérieure**

C'est une boucle fictive qui part d'un noeud à niveau fixe pour aboutir à un autre noeud à niveau fixe,elle est constituée des conduites appartenant au circuit et d'une conduite imaginaire reliant les deux noeuds à niveau fixe.

**-Conduite**

Les conduites constituent les éléments principaux du réseau ,une conduite dans notre cas aura une section constante (diamètre) et une longueur bien définie,elles relient les différents noeuds(noeud de jonction,noeud à niveau fixe) du réseau.

La figure 2.1 montre ce différents éléments

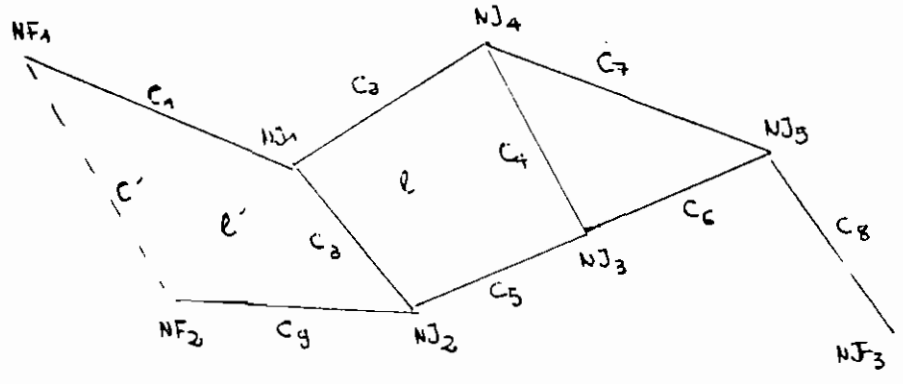



Fig 2.1

$NF_i$ : Noeuds fixes

$NJ_i$ : Noeuds de jonction

$C_i$ : Conduites

l : Boucle élémentaire($C_1, C_2, C_3, C_4, C_5$)

l' : Boucle extérieure($C_1, C_3, C_9, C'$)

C' : Conduite imaginaire

## 2.2 CONFIGURATIONS GEOMETRIQUES DU RESEAU

**-Réseau maillé**

C'est un réseau formé de plusieurs boucles élémentaires,il comporte généralement un réservoir qui alimente le réseau.Pour respecter l'équation de continuité,il doit exister des demandes au niveau de certains noeuds;la somme de ces demandes sera donc égale au débit du réservoir à travers le réseau.

**-Réseau ramifié**

C'est un réseau formé de plusieurs noeuds à niveau fixe ainsi que des noeuds de jonction;les conduites sont reliés aux différents noeuds de jonction sans former des boucles élémentaires mais par contre on trouve dans cette configuration des boucles extérieures.

**-Réseau étoilé**

C'est un réseau ramifié qui ne contient qu'un noeud de jonction mais il peut avoir plusieurs noeuds à niveau fixe.

La figure 2.2 nous montre des exemples de différentes configurations.

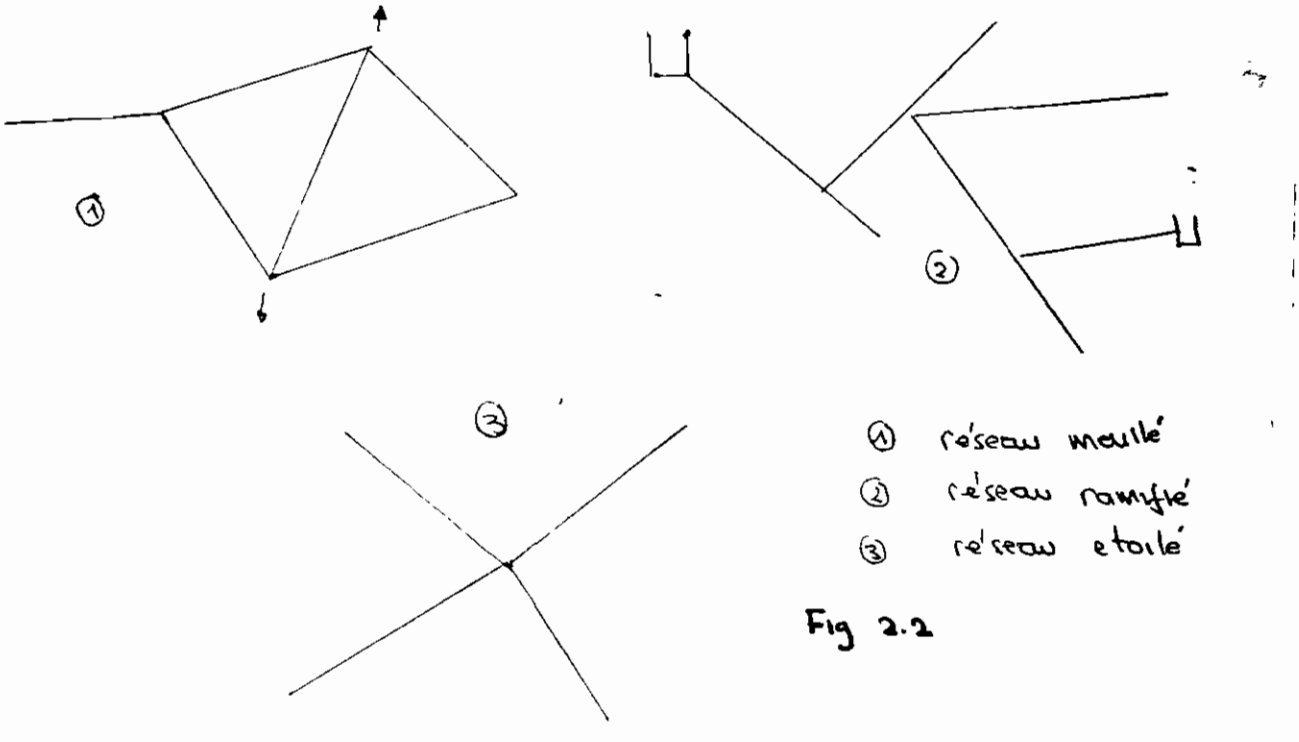



Fig 2.2

## 2.3 ÉLÉMENTS DE CONTROLES

Ces éléments sont introduits dans un réseau pour contrôler et/ou réguler l'état de certaines variables de l'écoulement (pression,débit);ils introduisent des pertes de charges supplémentaires dans le réseau.

**-Clapet antiretour**

Le clapet antiretour laisse passer l'écoulement dans un sens bien déterminé tout en bloquant le passage de l'écoulement dans le sens contraire

**-Régulateur de pression**

Il sert à maintenir dans une conduite une pression de service $P_R$ inférieure à la pression en amont du noeud où il est placé

Un régulateur de pression peut être modelisé comme sur la figure 2.3 par deux noeuds

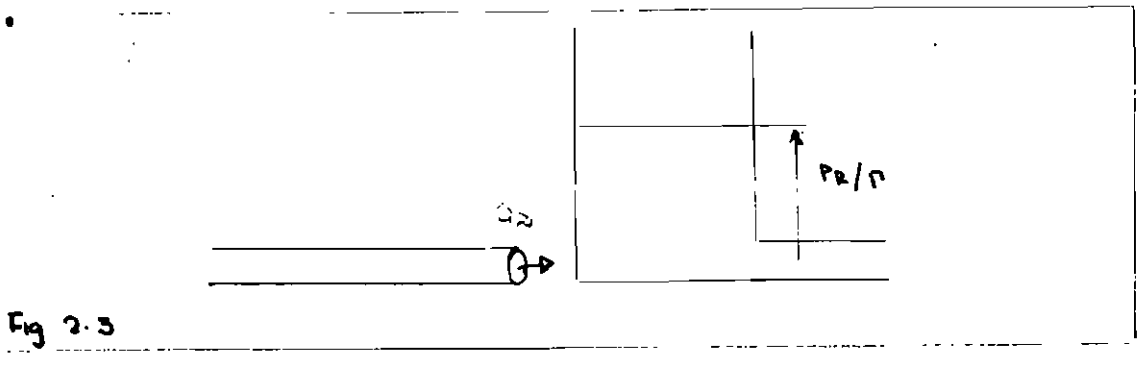



Fig 2.3

-Le noeud amont est un noeud de jonction de débit égal au débit traversant le régulateur

-Le noeud aval est un noeud à niveau fixe dont le niveau est la somme $P_R/\Gamma + z_R$

$z_R$ étant l'élévation géométrique du noeud.

Deux situations peuvent se présenter:

-Si le régulateur est placé entre une région à pression élevée et une autre de pression basse,il ne peut contrôler la pression aval(basse) si celle-ci est supérieure à la pression de service($P_R$);dans ce cas l'écoulement se fait en sens inverse du régulateur.Un clapet antiretour doit donc être placé en aval du régulateur pour bloquer l'écoulement dans ce sens

-La pression amont devient inférieure à la pression de service;dans ce cas le régulateur augmente la pression en aval.

**-Régulateur de débit**

Il sert à maintenir la pression amont en un endroit indiqué en limitant si nécessaire le débit à travers le régulateur

Il peut être modèlisé par un écoulement en sens inverse d'un régulateur de pression et une valve pouvant fonctionner en trois modes

1-La valve est entièrement ouverte et la pression amont est supérieure à la pression de service

2-La valve est étranglée et la pression amont est ajustée à la pression de service

3-La valve est fermée et la pression amont chute en deçà de la pression de service

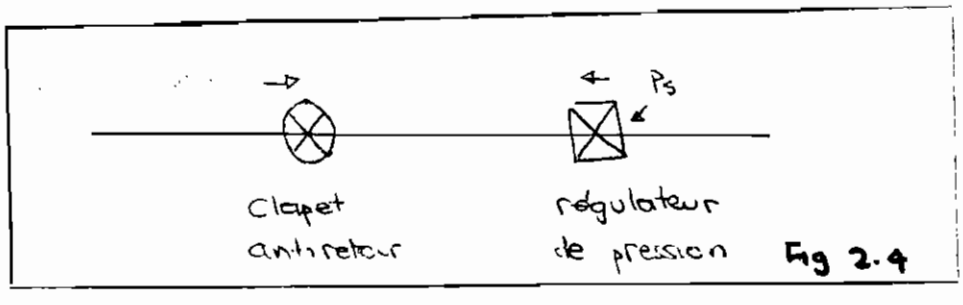


Fig 2.4

## CHAPITRE 3 CALCUL DES SINGULARITES

Un réseau hydraulique peut comporter plusieurs singularités .Pour arriver à résoudre sur ordinateur ce réseau,il faut trouver des modèles mathématiques adéquats à ces différentes singularités.La combinaison de ces modèles mathématiques au modèle mathématique général du système nous donnera les équations physiques du réseau,qui, résolues nous, fournirons les variables inconnues de l'écoulement.

### 3.1 POMPE

La pompe est une machine hydraulique qui transforme l'énergie mécanique venant d'un moteur,d'une turbine,etc,...en une énergie hydraulique fournie au fluide sous forme de pression.Une pompe peut être caractérisée complètement en spécifiant sa puissance ou sa courbe caractéristique.

La courbe caractéristique d'une pompe est l'ensemble des valeurs (H,Q) dans laquelle Q est le débit fourni à une hauteur de charge H.Le point de fonctionnement d'une pompe est le couple (H,Q) avec lequel la pompe opère dans le réseau,il est fonction des caractéristiques du réseau et de la pompe;généralement pour une application donnée, on choisit la pompe qui donne un rendement optimal au point de fonctionnement pompe-réseau considéré

L'incorporation d'une pompe dans le réseau se traduit par l'ajout d'un terme supplémentaire dans l'équation exprimant les

pertes de charges dans la conduite sur laquelle elle est placée. Il faut donc trouver une expression analytique reliant le débit à la hauteur de charge H=f(Q).

1-Pour une pompe dont la puissance est connue on sait que cette puissance P peut s'écrire:

$$P = \Gamma QH$$

Q et H étant respectivement le débit et la hauteur de charge au point de fonctionnement.

On peut donc écrire :

$H = Z/Q$ Z étant une fonction de la puissance de la pompe et de son rendement ∩ $Z = nP$.

2-La courbe caractéristique de la pompe est donnée sous forme des points expérimentaux (H,Q)

Dans ce cas il faut trouver une fonction f telle que H=f(Q) et qui approche le mieux possible la courbe caractéristique expérimentale.

La figure 3.1 représente l'allure générale de la courbe caractéristique d'une pompe centrifuge

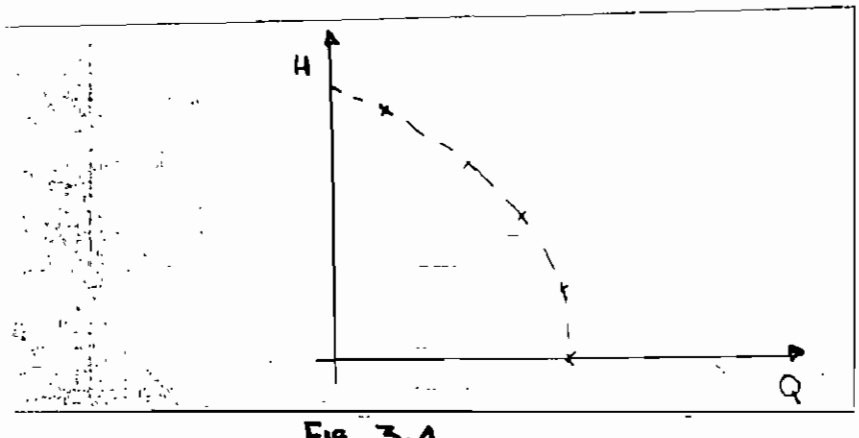



Fig 3.1

Cette courbe peut être approchée assez fidèlement par une relation

de la forme: $H = H_1 - \alpha Q^{\beta}$ (2.1)

$\alpha$ et $\beta$ sont des constantes qui dépendent de la distribution des points expérimentaux $(H_i, Q_i)$

On peut écrire $H_1 - H = \alpha Q^{\beta}$ (2.2)

En prenant le logarithme de deux membres on a :

$\ln(H_1 - H) = \ln\alpha - \beta \ln Q$ (2.3)

posons $\ln(H_1 - H) = y$ (2.4)

et $\ln Q = x$ on aura dans ce cas

$y = ax+b$ avec $a = \beta$ et $b = \ln\alpha$ (2.5)

a et b sont trouvés par la méthode de régression linéaire(moindres carrées) à partir des couples de ponts $(x_i, y_i)$

$b = cov(x,y)/var(x)$ (2.6)

$cov(x,y)$ est la covariance statistique de x et y

$cov(x,y) = \Sigma(x_i-x)(y_i-y)$ avec $x = \Sigma x_i/n$ et $y = \Sigma y_i/n$ (2.7)

n est le nombre de points

$var(x)$ est la variance statistique de x

$var(x) = \Sigma(x_i-x)^2$ d'où (2.8)

$b = \Sigma(x_i-x)(y_i-y)/ \Sigma(x_i-x)^2$ et (2.9)

$a = y - bx$ (2.10)

On aura finalement $\beta = a$ (2.11)

$\alpha = \exp(b)$ (2.12)

## 3.2 RESERVOIR(en régime permanent)

Les réservoirs sont des éléments fréquemment rencontrés dans les réseaux hydrauliques.Ils sont placés en amont ou en aval d'une

conduite

**-Réservoir en amont :**

considérons un reservoir placé en amont d'une conduite i

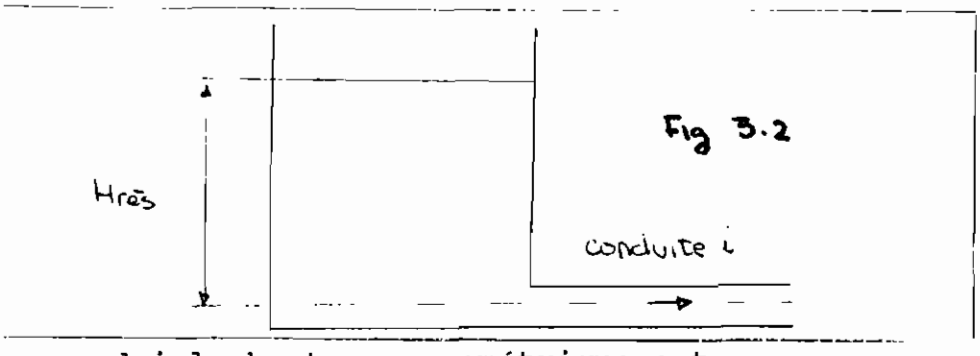



Fig 3.2

Au noeud i,la hauteur manométrique est

$$H_i = H_{rés} - (1+k)Q_i^2/2gA_i^2 \quad \text{avec} \qquad (2.13)$$

k : Coefficient de pertes de charges singulières à la sortie du réservoir

$A_i$ : section de la conduite.

**-Réservoir en aval**

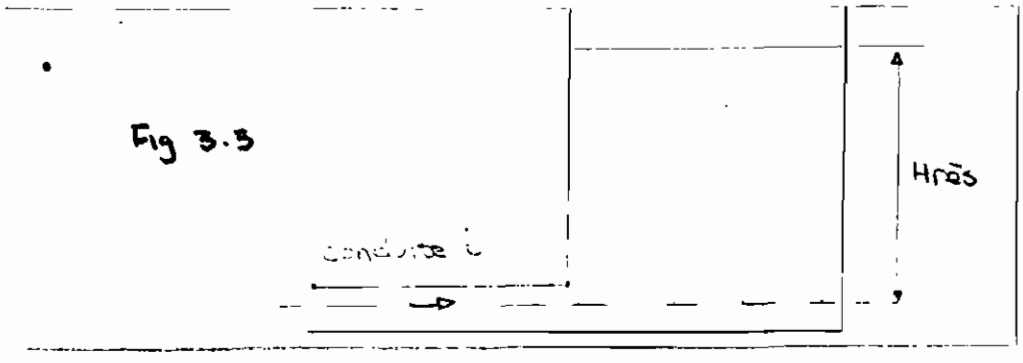



Fig 3.3

Dans ce cas la hauteur manométrique est

$$H_i = H_{rés} - (1-k)Qi^2/2gA_i^2 \qquad (2.14)$$

Dans les deux cas nous pouvons considérer un réservoir comme un noeud à niveau fixe $H_i$

**NOTE :**

Si le sens de l'écoulement est inversé dans les deux cas ,la valeur

de k est considérée comme négative.

### 3.3 VANNE

L'équation mathématique décrivant la manipulation d'une vanne en régime permanent est :

$$Q = (C_d\ A_v)(2gH)^{1/2} \qquad (2.15)$$

avec $C_d$ : coefficient de pertes de charges

$A_v$ : section de passage de la vanne,cette section varie selon le degré d'ouverture de la vanne.

## CHAPITRE 4: MODÈLE MATHEMATHIQUE GENERAL EN REGIME PERMANENT.

### 4.1 ÉQUATIONS DE BASE DU RESEAU.

Il s'agit dans ce chapitre de trouver le modèle mathématique régissant les écoulements en régime permanent .
Le problème est de trouver les débits dans toutes les conduites pour un ensembles de débits entrant et/ou sortant du réseau et ensuite de trouver les lignes d'énergie à travers le réseau.Les caractéristiques géométriques et physiques des conduites sont connues (longueur, diamètre, rugosité,etc..)ainsi que les propriétés physiques du fluide

Le modèle mathématique utilisé dans le calcul des réseaux hydrauliques sous pression est analogue au modèle mathématique de calcul des réseaux électriques en electrotechnique,ce modèle est appelé "Modèle de KIRCHOFF".

On établit ainsi l'analogie suivante entre les différentes grandeurs:

| Réseau électrique | Réseau hydraulique |
|---|---|
| Intensité du courant | Débit de l'écoulement |
| Tension entre deux points | Pertes de charges |

On aboutit ainsi aux lois suivantes:

**-Loi des noeuds:**

La somme de débits à noeud de jonction est nulle.Cette somme est une somme algébrique ,elle tient compte des signes des

différents débits.La loi des noeuds exprime l'équation de continuité à travers tous les noeuds du réseau.

**-Loi des mailles:**

La somme des pertes de charges sur une boucle fermée (boucle élémentaire et/ou boucle extérieure) est nulle.Cette loi exprime la conservation de l'énergie du réseau.

**-Pertes de charges:**

La formule exprimant les pertes de charges doit être vérifiée le long de chaque conduite.

Considérons un réseau comportant

j noeuds de jonction

f noeuds fixes

l boucles élémentaires

Pour un noeud de jonction nous pouvons écrire

$\Sigma Q_i = 0$ (j équations)

ce qui peut encore s'énoncer comme suit : la somme des débits arrivant à un noeud de jonction est égale à la somme des débits partant de ce même noeud (en incluant la demande du noeud).

L'équation d'énergie dans une boucle élémentaire s'écrit :

$\Sigma H_{fi} = 0$ (l équations)

-pertes de charges linéaires:

Elles peuvent s'écrirent: $H_{fi} = K_i Q_i^n$

avec n dépendant de la formule utilisée pour exprimer les pertes de charges linéaires dans une conduite ;Ki dépend [illegible] caractéristiques de la conduite et aussi de la formule de pertes de charges utilisée.

**1-Formule de Darcy-Weibash:**

$H_{fi} = f_i(L_i/D_i)V_i^2/2g$ (4.1)

la vitesse dans la conduite i en fonction du débit est telle que:

$Q_i = A_iV_i$ $A_i$ étant l'aire de la section. (4.2)

D'où $V_i = 4Q_i/\pi D_i^2$ (4.3)

$K_i$ devient donc : $K_i = (8L_if_i)/(2g\pi^2D_i^5)$ $n_i = 2$ (4.4)

**2-Formule de Hazen-Williams**

$V_i = 0.85C_{HWi}R_{hi}^{0.63}(H_{fi}/L_i)^{0.54}$ (4.5)

$V_i = 4Q_i/\pi D_i^2$ et $R_{hi} = D_i/4$ (4.6)

$K_i$ devient donc :

$K_i = [4L_i(4/D_i)^{0.63}/0.85C_{HWi}D_i^2\pi]^{1/0.54}$ $ni = 1/0.54$ (4.7)

**3-Formule de Manning**

$V_i = 1/n_iR_{hi}^{2/3}(H_{fi}/L_i)^{1/2}$ (4.8)

$K_i = [4L_in_i/\pi D_i^2(D_i/4)^{2/3}]^2$ $n_i = 2$ (4.9)

-Pertes de charges singulières

Pour une singularité elle s'écrit :

$H_{fi} = K_i'Q_i^n$ (4.10) $K_i'$ dépend de la longueur équivalente de la singularité rapportée à la conduite i

En incorporant une pompe dans la conduite i l'équation d'énergie devient:

$\Sigma[H_{fi} - f_i(Q_i)] = 0$ (4.11)

ou $\Sigma(K_iQ_i^n - \alpha_iQ_i^{\beta i}) = 0$ (l équations) (4.12)

Pour les f noeuds fixes du réseau on peut trouver f-1 boucles extérieure indépendantes et écrire l'équation d'énergie :

$\Sigma(K_iQ_i^n - \alpha_iQ_i^{\beta i}) = \Delta E$ (f-1 équations) (4.13)

E étant la différence de niveau entre les deux noeuds fixes de la boucle.

Finalement le modèle mathématique général du système en régime permanent est :

$$\Sigma Q_i = 0 \qquad \text{(j équations)} \qquad (4.14)$$

$$\Sigma K_i Q_i^n - \alpha_i Q^{\beta i}) = 0 \qquad \text{(l équations)} \qquad (4.15)$$

$$\Sigma K_i Q_i^n - \alpha_i Q^{\beta i}) = \Delta E \qquad \text{(f-1 équations)} \qquad (4.16)$$

Pour déterminer les débits dans toutes les conduites du réseau il faut résoudre ce système d'équations.on doit avoir p=j+l-f-1 ,comme étant le nombre de conduites

Il s'agit d'un système d'équations non linéaires de p équations à p inconnues,diverses méthodes de résolution numériques de ces équations sont présentées dans la littérature spécialisée Nous allons en exposer quelques unes.

## 4.2 METHODE DE LINEARISATION.

Le modèle mathématique est formé de j équations linéaires en Q et l+f-1 équations non linéaires;une des techniques de résolution consiste à linéariser les l+f-1 équations de façon à obtenir un système d'équations linéaires qu'on résoud facilement. Nous exposerons deux variantes de cette méthode:

### 4.2.1 : METHODE LINEAIRE DE WOODS ET CHARLES.

Woods et Charles ont suggéré la linéarisation de l'équation d'énergie de la manière suivante:

$$H_{fi} = K_i Q_i^n - \alpha_i Q^{\beta i} \qquad (4.17)$$

$$= (K_i Q_{io}^{n-1} - \alpha_i Q_{io}^{\beta-1}) Q^i = K'_i Q_i \qquad (4.18)$$

$Q_{io}$ étant un débit aproximatif dans la conduite i

Nous obtenons ainsi le système d'équations suivant:

$$\Sigma Q_i = 0 \qquad (j \text{ équations}) \qquad (4.19)$$

$$\Sigma K'_i Q_i = 0 \qquad (1 \text{ équations}) \qquad (4.20)$$

$$\Sigma K'_i Q_i = \Delta E \qquad (f-1 \text{ équations}) \qquad (4.21)$$

C'est un système d'équations linéaires qui peut être resolus par différentes méthodes connues(Gauss-Jordan,Gauss-Seidel,etc,..)

Algorithme de résolution

1-Au départ on suppose $Q_{io} = 1$ dans toutes les conduites

Dans ce cas $K'_i = K_i - \alpha_i \qquad (4.22)$

2-Résolution du système d'équations linéaires formée par la méthode de Gauss Seidel.

3-Les débits trouvés en 2 sont utilisés comme débits approximatifs pour la deuxième itération.Après deux itérations on utilise la formule suivante pour le débit approximatif:

$Q_{io} = (Q_{io-1} - Q_{io-2})/2$ afin d'améliorer la convergence en diminuant les oscillations autour de la solution.

La procédure est réiterée jusqu'à l'obtention de la précision souhaitée(0.1 l/s dans le programme).Un des avantage de cette méthode est qu'il n'est pas nécessaire de donner des débits initiaux pour résoudre les équations.

**4.2.2** Une autre méthode de linéarisation transforme l'équation d'énergie de la manière suivante:

$$H_i = f(Q_i) = K_i Q_i^{n} - \alpha_i Q_i^{\beta i} \qquad (4.23)$$

Développons la fonction $f(Q_i)$ autour de $Q=Q_i$ par la formule de Taylor au premier ordre

$$f(Q) = f(Q_i) - (\delta f/\delta Q)_{Q=Qi}(Q-Q_i) \qquad (4.24)$$

soit $$f'(Q_i) = G_i = (\delta f/\delta Q)_{Q=Qi} \qquad (4.25)$$

$$= nK_i Q_i^{n-1} - \beta_i \alpha_i Q_i^{\beta i-1} \qquad (4.26)$$

autour d'une boucle l'équation d'énergie s'écrit:

$$\Sigma f(Q_i) = 0 \text{ ou } \Sigma f(Q_i) = \Delta E \qquad (4.27)$$

$$\Sigma f(Q) = \Sigma(H_i + G_i Q - G_i Q_i) = 0 \text{ d'où} \qquad (4.28)$$

$\Sigma G_i Q = \Sigma(G_i Q_i - H_i)$ (4.29) pour toutes les boucles du réseau

Nous obtenons le système d'équations linéaires suivant:

$$\Sigma Q_i = 0 \qquad \text{(j équations)} \qquad (4.29)$$

$$\Sigma G_i Q = \Sigma(G_i Q_i - H_i) \qquad \text{(1 équations)} \qquad (4.30)$$

$$\Sigma G_i Q = \Sigma(G_i Q_i - H_i + \Delta E) \qquad \text{(f-1 équations)} \qquad (4.31)$$

Pour la résolution numérique ,il faut trouver au départ un ensemble de débits initiaux en respectant la condition de continuité;ensuite il faut utiliser une procédure d'approximation successive dans laquelle les débits trouvés à une itération i servent des débits approximatifs pour l'itération suivante i+1.

La procédure est continuée jusqu'à l'obtention de la précision souhaitée. Nous remarquons que dans cette méthode ,il faut au préalable déterminer les débits initiaux,ce qui constitue un désavantage.

## 4.3 BALANCEMENT DU RESEAU

## (HARDY-CROSS)

Soit le modèle mathématique décrivant le réseau hydraulique

$$\Sigma Q_i = 0 \qquad \text{(j équations)} \qquad (4.32)$$

$$\Sigma(K_i Q_i^n - \alpha_i Q_i^{\beta i}) = 0 \qquad \text{(1 équations)} \qquad (4.33)$$

$$\Sigma(K_i Q_i^n - \alpha_i Q_i^{\beta i}) = \Delta E \qquad \text{(f-1 équations)} \qquad (4.34)$$

Le raisonnement de la méthode de Hardy-Cross est le suivant

Au début, on suppose des valeurs des débits dans toutes les conduites de sorte que l'équation de continuité soit satisfaite à chaque noeud de jonction.

Soit la conduite i et $Q_{io}$ le débit supposé et $Q_i$ le débit réel l'erreur par rapport à la valeur réelle est :

$$\Delta Q_i = Q_i - Q_{io} \qquad (4.35)$$

Sur une boucle, la somme des pertes de charges doit être nulle mais ne l'est pas car les débits supposés ne sont pas les valeurs réelles.

En faisant une approximation de Taylor au premier ordre de $Q_i^n$ et de $Q_i^{\beta i}$ au voisinage de $Q = Q_{io}$ on trouve

$$H_f = H_{fi} - H_{fio} = K_i[Q_i^n - Q_{io}^n] - \alpha_i[Q_i^{\beta i} - Q_{io}^{\beta i}] \qquad (4.36)$$

$$Q_i^n = Q_{io}^n + (\delta Q_i^n/\delta Q_i)_{Qi=Qio}(Q_i - Q_{io}) \qquad (4.37)$$

$$= Q_{io}^n + nQ_{io}^{n-1}\Delta Q_i \qquad (4.38)$$

$$Q_i^{\beta i} = Q_{io}^{\beta i} + (\delta Q_i^{\beta i}/\delta Q_i)_{Qi=Qio}(Q_i - Q_{io}) \qquad (4.39)$$

$$= Q_{io}^{\beta i} + \beta i Q_{io}^{\beta i-1}\Delta Q_i \qquad (4.40)$$

D'où $H_{fi} = [nK_iQ_{io}^{n-1} - \alpha_i\beta_iQ_{io}^{\beta i-1}]\Delta Q_i$ (4.41)

Si on connaissait $H_f$ on aurait pu calculer $\Delta Q$ mais il n'en est pas ainsi car $H_f$,la perte de charge réelle est inconnue.

On peut cependant calculer l'écart de la perte de charge sur une boucle par rapport à la solution réelle

$$\Sigma H_i = \Sigma[H_{fi} - H_{fio}] \quad (4.42)$$

or $\Sigma H_{fi} = 0$ (équation d'énergie) (4.43)

D'où $\Sigma H_{fi} = -\Sigma H_{fio}$ (4.44)

$\Sigma H_{fio}$ peut être calculé en appliquant la formule de la perte de charges à chaque conduite avec le débit supposé $Q_{io}$

$$-\Sigma H_{fio} = \Sigma[nK_iQ_{io}^{n-1} - \alpha_i\beta_iQ_{io}^{\beta i-1}]\Delta Qi = \Sigma H_{fi} \quad (4.45)$$

Cette équation montre que $\Sigma Hf_{io}$ est du aux écarts des débits dans les conduites par rapport à la solution exacte

$$\Sigma H_{fio} = \Sigma[K_iQ_{io}^{n} - \alpha_iQ_{io}^{\beta i-1}] \quad (4.46)$$

$$= \Sigma[K_iQ_{io}^{n-1} - \alpha_iQ_{io}^{\beta i-1}]Q_{io} \quad (4.47)$$

D'où l'équation suivante :

$$\Sigma[nK_iQ_{io}^{n-1} - \alpha_iQ_{io}^{\beta i-1}]\Delta Q_i = -\Sigma[K_iQ_{io}^{n-1} - \alpha_iQ_{io}^{\beta i-1}]Q_{io} \quad (4.48)$$

$|\Delta Qi|$ a la même valeur dans chaque conduite de la boucle

$$Q\Sigma[nK_i|Q_{io}|^{n-1} - \alpha_i\beta_i|Q_{io}|^{\beta i-1}] \quad (4.48)$$

$$= -\Sigma[K_i|Q_{io}|^{n-1} - \alpha_i|Q_{io}|^{\beta i-1}]Q_{io} \quad (4.49)$$

D'où la valeur de $|\Delta Q|$ pour une boucle:

$$\Delta Q = \frac{-\Sigma[K_i|Q_{io}|^{n-1} - \alpha_i|Q_{io}|^{\beta i-1}]Q_{io}}{\Sigma[nK_i|Q_{io}|^{n-1} - \alpha_i\beta_i-1|Q_{io}|^{\beta i-1}]} \quad (4.50)$$

Pour une boucle extérieure ,en considerant la différence de niveau $\Delta E$ ,on a:

$$\Delta Q = \frac{-\Sigma[K_i|Q_{io}|^{n-1} - \alpha_i|Q_{io}|^{\beta i-1}]Q_{io} - E}{-\Sigma[nK_i|Q_{io}|^{n-1} - \alpha_i\beta_i-1|Q_{io}|^{\beta i-1}} \qquad (4.51)$$

**NOTE:**

Q calculé avec la formule ci-haut n'est pas la valeur exacte de $Q-Q_o$ ,puisque cette formule est obtenue en négligeant les termes d'ordre supérieur ou égal à deux en $\Delta Q$ dans le développement de $Q_o + \Delta Q$.

Comme conséquence de la remarque précédente $Q_1 = Q_o + \Delta Q$ n'est pas la solution exacte du problème ,mais une amélioration de la solution $Q_o$ permettant de mieux approcher la solution exacte

**-Algorithme de la méthode:**

1-Supposer le débit et son sens dans chaque conduite de sorte à respecter l'équation de continuité à travers tous les noeuds du réseau

2-Pour chaque boucle du réseau ,calculer le dénominateur et le numérateur de la formule de $\Delta Q$.

3-Si la somme des pertes des charges calculées sur chaque boucle et les valeurs de $\Delta Q$ sont suffisamment petites,la solution est atteinte sinon on remplace les débits supposés au départ $Q_o$ par $Q_o + \Delta Q$ et on reprend le processus à partir de 1

**NOTE :**

Si une conduite appartient à deux boucles adjacentes ,on lui applique Q pour les deux boucles

**Note sur le calcul des pertes des charges :**

Pour les formules de pertes de charges explicites(Hazen-

Williams,Manning);le coefficient $K_i$ ne dépend pas du débit,il ne dépend que des caractéristiques géométriques et physiques de la conduite,il est donc calculé une fois et reste constant durant toute la simulation.

L'utilisation de la formule de Darcy-Weibash,qui est implicite,et dépend du débit exige une procédure itérative que nous allons décrire

**Algorithme :**

Pour chaque $Q_i$:

1-On calcule une valeur de départ du coefficient de frottement en utilisant la formule DE BLASIUS

$$f_{i1} = 0.3164Re^{-1/4} \qquad (4.52)$$ ce qui entraîne que

$$f_{i1} = 4[0.3164(1/\pi Dmhu)^{-1/4}]Q_i^{-1/4} \qquad (4.53)$$

2-Avec la valeur de $f_{i1}$ on calcule $f_{i2}$ par la formule :

$$f_{i2} = 0.86/[\ln(e/3.7D + 1.97mhu/Q_iDf_{i1})]^2 \qquad (4.54)$$

ainsi de suite jusqu'à avoir

$|f_{ik} - f_{ik-1}| < \mu$ $\mu$ étant un nombre petit arbitraire

Le $f_i$ est utilisé dans l'équation des pertes des charges

Ainsi pour chaque itération,des nouvelles valeurs du coefficient de frottement sont calculées (dépendant du débit).

La méthode de Hardy_Cross,même adaptée à l'ordinateur présente les désavantages suivants:

-Nécessité d'estimer les débits initiaux.

-Rythme de convergence lent.

-Solution non assurée.

La méthode que nous proposons pour calculer le réseau est une

combinaison de la méthode de Hardy-Cross et de la méthode de linéarisation.La méthode de linéarisation est utilisée pour déterminer des débits initiaux assez proches des débits réels en utilisant un coefficient de frottement arbitraire dans toutes les conduites.Le débits calculés par la méthode de linéarisation sont utilisés pour balancer le réseau(Hardy-Cross) ce qui à notre sens contribuera à accélérer la convergence et à limiter les oscillations autour de la solution.

## CHAPITRE 5_ REGIME QUASI-STATIQUE

### 5.1 GENERALITES SUR LE REGIME INSTATIONNAIRE

Le régime instationnaire se définit comme un régime dans lequel les variables de l'écoulement(pression,vitesse,etc,..) dépendent du temps.Il est régit par des équations aux dérivées partielles requérant des méthodes numériques puissantes pour les résoudre.

En considérant le taux de variation des variables d'écoulements par rapport au temps ,il est possible ,de classer le régime instationnaire en trois grandes classes qui sont :

**1-Régime quasi-statique:**

Ici le taux de variation de la masse du fluide est continue dans le temps,mais l'accélération et les forces produisant ces accélérations restent négligeables.Dans ce cas les équations du régime permanent peuvent être appliquées avec une précision acceptable.

Exemple : -Remplissage continu d'un réservoir

-Vidange continu d'un réservoir

**2**-Les oscillations de la masse fluide où le taux de variation par rapport au temps de la vitesse est assez important pour produire une accélération notable du fluide,mais les forces d'accélération restent tout de même assez petites pour pouvoir ignorer la compressibilité du fluide

Exemple : Écoulement des fluides dans les machines à piston.

3-Écoulement des fluides où le temps de changement de vitesse du fluide est comparable à la période d'oscillations des ondes de vitesses dans le système.Dans ce cas les conduites sont considérées comme étant élastique et la compressibilité du fluide est considérée dans l'établissement des équations.Ce régime est connu sous le nom du 'Régime transitoire',les équations régissant ces phénomènes sont résolues généralement sur ordinateur en utilisant des méthodes numériques(méthode des différences finies, méthode des caractéristiques,etc,..)

Exemples : -Effet de la fermeture brusque d'une vanne dans un réseau hydraulique.

-Perte de puissance d'une pompe dans un réseau hydraulique.

Dans le cadre de cette étude nous développerons uniquement le régime quasistatique.

## 5.2 THEORIE DE LA COLONNE DE FLUIDE RIGIDE

Etudions la simulation d'un réservoir en régime quasistatique en appliquant la théorie de la colonne de fluide rigide

-Considérons le cas d'un réservoir se vidant dans un réseau hydraulique à travers une conduite.Les caractéristiques géométriques et physiques (longueur , diamètre , rugosité, etc , ...de toutes les conduites sont connues

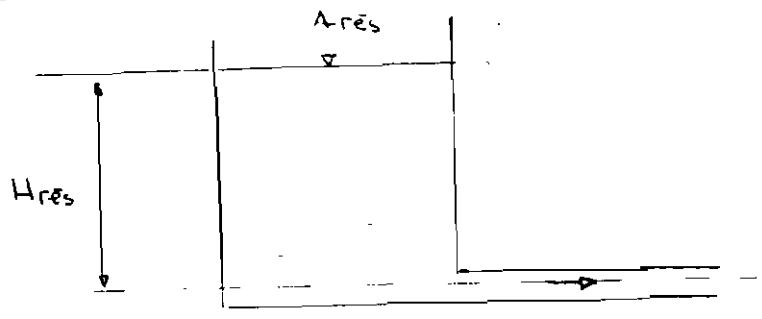



Fig 5.1

- $Q_{in}$ : débit exterieur
- $L_i$ : longueur de la conduite i
- $D_i$ : diamètre de la conduite i
- $A_i$ : aire de la conduite i
- $A_{rés}$ : Aire du réservoir (fonction de H)
- $f_i$ : coefficient de perte de charge de la conduite i
- $K_i$ : coefficient des pertes de charges singulières de la conduite i

L'équation de continuité s'exprime en égalant le taux de variation par rapport au temps de la hauteur du réservoir et le débit traversant la conduite :

$$A_{rés}(-\delta H/\delta t)+Q_{in} = A_i V_i \qquad (5.1)$$

en supposant $A_{rés}$ très grand par rapport à $A_i$

La hauteur manométrique est :

$$H = (f_i L_i/D_i + K_i)V_i^2/2g \qquad (5.2)$$

En tirant la valeur de $V_i$ de l'équation de la hauteur manométrique et en la remplaçant dans l'équation de continuité on a :

$$A_{rés}(-\delta H/\delta t) = C\sqrt{H} \qquad (5.3)$$

avec $C=A_i[2g/(f_iL_i)/D_i + K_i]^{1/2}$ (5.4)

Cette équation peut être intégrée entre deux valeurs de H :$H_1$ et $H_2$ pour trouver le temps que met le fluide pour atteindre $H_2$ à partir de la position $H_1$.On peut trouver une relation de la hauteur du réservoir par rapport au temps en fixant une hauteur initiale $H_{rés}$

$$\int_{H_{rés}}^{H} A_{rés}\, dH/\sqrt{H} = -Ct \qquad (5.5)$$

En intégrant cette dernière équation on trouve

$$t= 2/C_i[H_{rés}^{1/2} - H^{1/2}]\, A_{rés} \qquad (5.6)$$

Ainsi la hauteur atteinte par le réservoir après un temps t est : -réservoir en amont:

$$H = [H_{rés}^{1/2} - tC_i/2]^2 \qquad (5.7)$$

-réservoir en aval :

$$H = [H_{rés}^{1/2} - tC_i/2]^2 \qquad (5.8)$$

Soit un réseau hydraulique avec réservoir,le problème ici consiste de déterminer les débits dans toutes les conduites après un temps de simulation donnée $\Delta t$ ,il suffit dans ce cas de calculer les hauteurs de tous les réservoirs au temps voulu et d'appliquer les équations du régime permanent avec les hauteurs trouvées.
(Au temps t=0,les hauteurs initiales sont connues).On peut ainsi trouver les variables de l'écoulement après un temps donné .

## CHAPITRE 6: DESCRIPTION ET UTILISATION DU PROGRAMME

### 6.1 PRESENTATION DES MODELES MATHEMATIQUES

**-Hypothèses**

-Ecoulement d'un fluide incompressible

-Ecoulement stationnaire

-Ecoulement unidimensionnel

**-Modèles mathématiques**

Le modèle mathématique utilisé est établi dans le chapitre 4 ,il s'agit de résoudre le système d'équations suivant:

$$\Sigma Q_i = 0 \qquad \text{(j équations)}$$

$$\Sigma(K_i Q_i^{n} - \alpha_i Q_i^{\beta i}) = 0 \qquad \text{(l équations)}$$

$$\Sigma(K_i Q_i^{n} - \alpha_i Q_i^{\beta i}) = \Delta E \qquad \text{(f-1 équations)}$$

Ce modèle est trouvé en appliquant les équations fondamentales de la mécanique de fluide ( équation de continuité ,équation de l'énergie cinetique) à un réseau hydraulique comportant :

-j noeuds de jonctions

-l boucles élémentaires

-f-1 noeuds à pression fixe

$\alpha_i$ et $\beta_i$ sont fonctions de la courbe caractéristique de la pompe se trouvant sur la conduite i ( voir chapitre 3,section 3.2)

$K_i$ est fonction des caractéristiques géométriques et physiques de la conduite i (longueur,diamètre,rugosité,...)

$$K_i = (8L_i f_i)/(2g\pi^2 D_i^5)$$

n est égal à 2 pour la formule de Darcy-Weibash

Les lignes de niveau sont trouvés en appliquant l'équation de

Bernouilli avec pertes de charges en partant d'un noeud à pression fixe

### 6.2 METHODES ET ALGORITHMES DE RESOLUTIONS

La méthode utilisée dans le programme pour résoudre le modèle mathématique est une combinaison de la méthode de linéarisation et de Hardy-Cross

**-Calculs des débits initiaux**

Les débits initiaux sont calculés par la méthode de linéarisation avec un coefficient de frottement arbitraire de f= 0.025 dans toutes les conduites du réseau (voir chapitre 4,section 4.2)

**-Méthode de Hardy-Cross**

La méthode de Hardy-Cross est utilisé pour balancer le débits fournis par la méthode de linéarisation (respectant l'équation de continuité à travers tous les noeuds du réseau)

Ici le coefficient de frottement est calculé à chaque itération par la formule de Colebrook en partant de la rélation de Blasius(voir algorithme page 36)

Les réservoirs et les pompes sont traités dans des modules indépendants.

### 6.3 STRUCTURE GENERAL DU PROGRAMME

**'HYDNET'** est un programme de simulation numérique des écoulements en réseaux hydrauliques en régime permanent,il est interactif càd qu'il possède des menus déroulant permettant à l'utilisateur de saisir facilement son réseau et d'en faire la simulation

Le programme est écrit en TURBOPASCAL 4.0 et profite ainsi de

la grande structuration du langage ; en effet chaque traitement important est réalisé dans un module indépendant.Les différents modules communiquent entre eux des variables.

Aussi la structuration du programme permet la modification des modules qui sont parfaitement indépendants les uns des autres .On peut aussi y ajouter des nouveaux modules.

Dans le paragraphe suivant nous décrirons brièvement les unités que comporte le programme dont le listing est donné en annexe

**CRTHYD** est le module de saisie des données sur le clavier

**SIMHYD1** calcule les débits initiaux

**SIMHYD2** calcule les débits à l'aide de la méthode de Hardy-Cross

**SIMHYD3** calcule les lignes de niveaux dans le réseau

**GRAPHYD1** est le préproprecesseur graphique du programme ,qui permet à l'utilisateur de visualiser son réseau sous forme graphique

**NUMHYD** affiche les résultats numériques du programme

Tous ces modules sont gérés par le programme principal **MAINHYD**

Le programme comporte également certains utilitaires qui sont :

**DECLAR ,DECLAR1 :** déclarations de toutes les variables globales du programme

**MATHTOOLS** est un ensemble des routines mathématiques utilisées dans le programme

**GAUSSIDL** résoud des systèmes d'équations par la méthode de Gauss-Seidel

### 6.4 INTERFACES AVEC L'USAGER

L'usager doit au préalable représenter son réseau physiquement ,il doit numéroter l'ensemble des éléments composant son réseau,ainsi il doit rentrer au clavier les différentes caractéristiques qui sont

-Nombre de noeuds de jonction

-Nombre de conduites

-Nombre de noeuds à pression fixe

-Nombre de boucles élémentaires

-Nombre de réservoirs

Pour une conduite les informations essentielles sont :

-Numéro de la conduite

-Longueur

-Diamètre

-rugosité absolue

-nombre des pompes ( puissance ou courbe caractéristique )

Il doit également constitué ses boucles et fournir au programme les numéros des noeuds composant les différentes boucles.

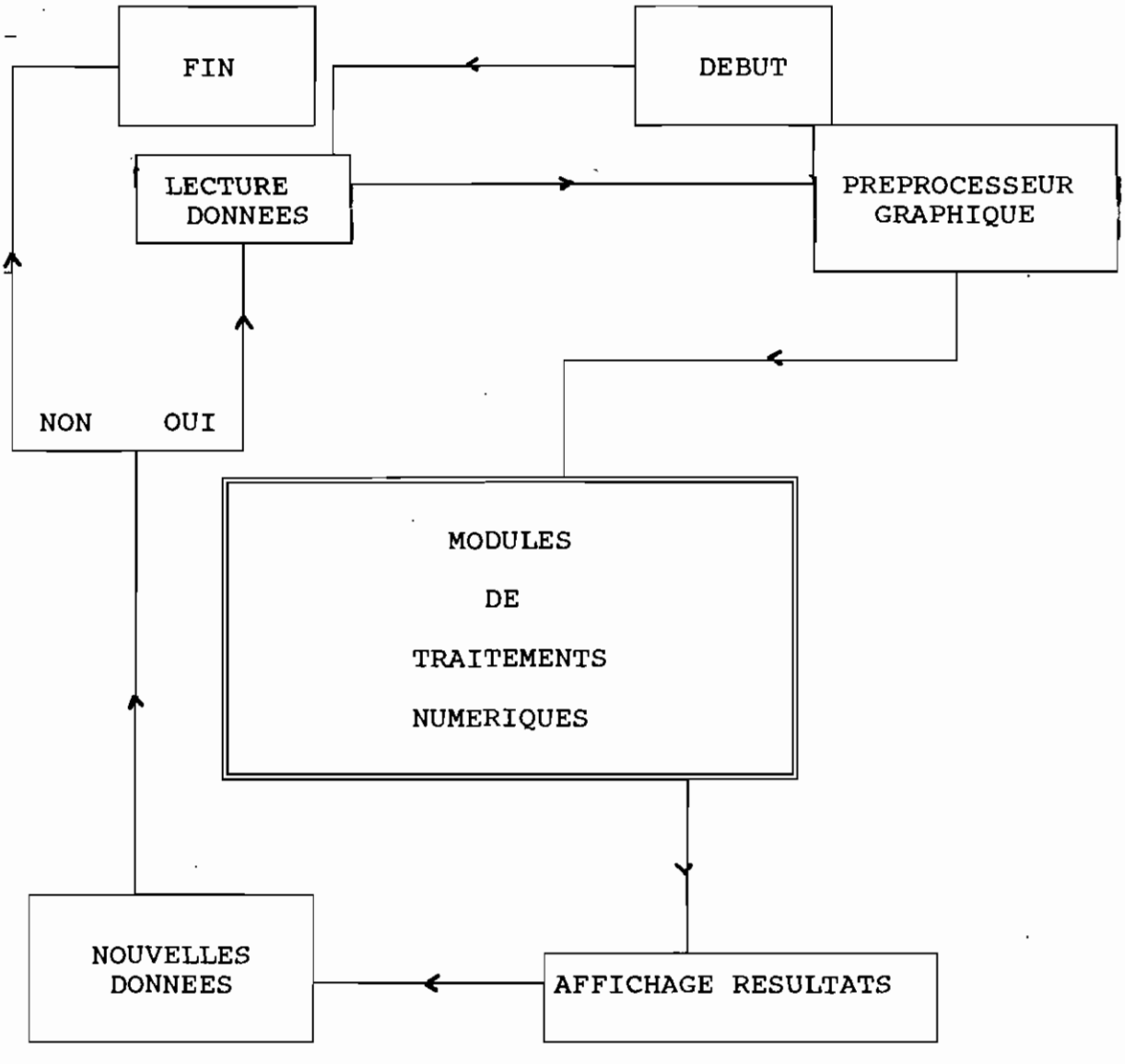
Pour un réservoir ,il doit spécifier le numéro du noeud à pression fixe correspondant.Il indiquera aussi les débits extérieures aux noeuds ainsi que leurs élévations géométriques et la pression (pour un noeud à pression fixe)

Les données relatives à un réseau peuvent être stockées sur disque, il suffira dans ce cas de donner un nom au fichier qui servira à sauvegarder ce réseau.

L'étape suivante consiste à la vérification par l'utilisateur

du réseau rentré au clavier, cette étape est réalisée par le préprocesseur graphique du programme.Ici un écran graphique est proposé à l'utilisateur .A partir des différentes commandes fournies et affichées à l'écran, l'usager place ses différents noeuds à l'écran de façon judicieuse; ainsi le réseau saisi sera représenté graphiquement donnant à l'utilisateur la possibilité de vérifier les données saisies au clavier et de recommencer la saisie en cas de non conformité.Enfin les différents résultats numériques seront affichés à l'écran

### 6.5 ALGORITHME GENERAL DU PROGRAMME



## CONCLUSION ET RECOMMANDATIONS

Cette étude nous a permis de mettre au point un logiciel de simulation des réseaux hydrauliques en régime permanent .Le programme permet de traiter les différentes configurations possible d'un réseau(maillé,étoilé ,ramifié) et comportant différentes singularités (pompe,réservoirs,etc..).

Les équations mathématiques régissant ces écoulements (équation de continuité,équation d'énergie) ainsi que les fonctionnement des éléments de base des réseaux ont étés modélisées afin de trouver des algorithmes pouvant être programmé sur ordinateur.L'interface graphique du programme constitue un outil important pour le traitement des réseaux.

Le programme présente certaines instabilités pour la résolution des systèmes d'équations ;les raisons de cette instabilité peuvent être dues à la programmation de certaines formulations mathématiques qui pourraient nécessiter un temps suffisamment grand pouvant dépasser celui qui nous est alloué dans le cadre de cette étude.

Mais toutefois l'extension et la continuation du programme reste toujours possible grâce a la grande souplesse adoptée pour la programmation,en effet chaque traitement est isolé dans un module qui le décrit fidèlement ,les extensions peuvent se peuvent par des modules supplémentaires à ajouter aisément au

programme

En ajoutant des modules pour les traitements du régime transitoire ,ce programme deviendrait un outil complet et indispensable pour la conception des réseaux hydrauliques ,outil académique mais aussi professionnel.

# ANNEXE

LISTING DU PROGRAMME

```
PROGAM MAINHYD

USES CRT,
     GESTECR,
     DOS

(*$M 16384,0,0 *)
(*spécification de la taille du tas*)
TYPE
   activites=(permanent,quasistatique,quitter);
   EnregActiv=record
       Nomproc:string[20];
       ligne,
       colonne:byte;
       chaineMenu:string[50]
     end;
 CONST
   poscol=20;
   affichoption=' P Q S ';

   carnul=#0;
   entree=#13;
   sonnerie=#7;

   EnHaut=#72;
   AGauche=#75;
   ADroite=#77;
   EnBas=#80;

   activite:array[activites] of enregactiv=
     ((nomproc:'permanent1';ligne:8;colonne:poscol;
       chainemenu:'Simulation en regime permanant '),
      (nomproc:'quasistatique1';ligne:10;colonne:poscol;
       chainemenu:'Simulation en regime quasistatique '),
       (nomproc:'';ligne:12;colonne:poscol;
       chainemenu:'Sortir '));

   carmenu:set of char=
            ['P','Q','S',carnul,entree];

   codesfleches:set of char=
            [enhaut,agauche,adroite,enbas];
VAR
  fait:boolean;
  selection:activites;

FUNCTION Majuscules(chsaisie:string):string;
  VAR
    i:integer;
    chres:string;
  begin
    chres:='';
    for i:=1 to length(chsaisie) do
      chres:=chres+upcase(chsaisie[i]);
    majuscules:=chres
  end;



PROCEDURE Surbrillance;
  begin
    reversevideo(true);
    with activite[selection] do
      begin
```

```
        gotoxy(colonne,ligne);
        writeln(majuscules(chainemenu))
      end;
    reversevideo(false);
    gotoxy(60,17)
  end;

PROCEDURE Menuinitial;
  VAR
    option:activites;
  begin
    clrscr;

    Reversevideo(true);
    gotoxy(poscol-5,4);
    writeln(' * SIMULATION NUMERIQUE DES ECOULEMENTS * ');
    reversevideo(false);

    for option :=permanent to quitter do
      with activite[option] do
         begin
           gotoxy(colonne,ligne);
           writeln(chainemenu);
         end;

     selection :=permanent;
     surbrillance;
     gotoxy(poscol-12,16);
     write('Utiliser');
     reversevideo(true);
     write(#24,' ',#25,' ',#26,' ',#27);
     reversevideo(false);
     write(' ou ');
     reversevideo(true);
     write(affichoption);
     reversevideo(false);
     write(' pour selectionner une option , ');

     gotoxy(poscol-10,17);
     write('puis appuyer sur <Entrée> pou confirmer.)');
   end;

 PROCEDURE Selectionne(var signalfin:boolean);
   CONST
     premcar='PQS';
   VAR
     carsaisi:char;
   PROCEDURE Continue;
     VAR
       espace:char;
     begin
       gotoxy(10,25);
       write('Appuyer sur la barre d''espacement pour revenir au menu. ');
       repeat
         espace:=readkey
       until espace=' ';
     end;

   PROCEDURE EnleverSurbrillance;
     begin
       with activite[selection] do
         begin
           gotoxy(colonne,ligne);
           writeln(chainemenu)
         end
```

```
    end;

PROCEDURE Choixsuivant;
  begin
    enleversurbrillance;
    if selection=quitter then
      selection:=permanent
    else
      selection:=succ(selection);
    surbrillance
   end;

 PROCEDURE ChoixPrecedent;
   begin
     enleversurbrillance;
     if selection=permanent then
       selection:=quitter
     else
       selection:=pred(selection);
     surbrillance
   end;

   begin(*selectionne*)
     signalfin:=false;
     repeat
       carsaisi:=upcase(readkey);
       if not (carsaisi in carmenu) then
         write(sonnerie)
     until (carsaisi in carmenu);

  case carsaisi of
    'P','Q','S' :
    begin
      enleversurbrillance;
      selection:=
            activites(pos(carsaisi,premcar)-1);
      surbrillance
    end;

    carnul :
      begin
        carsaisi:=readkey;
        if carsaisi in codesfleches then
          case carsaisi of
            enhaut,agauche:choixprecedent;
            enbas,adroite:choixsuivant
          end
        else
          write(sonnerie)
       end;

 entree:

 begin
   if selection=quitter then
     signalfin:=true
   else
     begin
        if selection=permanent then
          exec('b:\gestcrt1.exe','');
         if selection=quasistatique
           then write('ZOOI');
        continue;
        clrscr;
        menuinitial
```

```
      end
    end
  end
end;

begin
 menuinitial;
 repeat
   selectionne(fait);
 until fait;
 clrscr
 end.
```

```
PROGRAM MAINHYD1

USES CRT,
     DECLAR1,
     GESTECR,
     CRTHYD,
     SIMHYD1,
     SIMHYD2,
     SIMHYD3,
     GRAPHYD1,
     NUMHYD,
     DOS;

(*$M 65520,0,D*)
(*spécification de la taille du tas*)

TYPE
   activites=(saisie_clavier,sauvegarde,lecture,quitter);
   EnregActiv=record
       Nomproc:string[20];
       ligne,
       colonne:byte;
       chaineMenu:string[50]
     end;
 CONST
   poscol=20;
   affichoption=' C F U S ';

   carnul=#0;
   entree=#13;
   sonnerie=#7;

   EnHaut=#72;
   AGauche=#75;
   ADroite=#77;
   EnBas=#80;

   activite:array[activites] of enregactiv=
     ((nomproc:'saisie_clavier1';ligne:6;colonne:poscol;
       chainemenu:'saisie au Clavier '),
      (nomproc:'sauvegarde1';ligne:8;colonne:poscol;
       chainemenu:'sauvegarder  sur Fichier'),
       (nomproc:'lecture1';ligne:10;colonne:poscol;
       chainemenu:'Utiliser un fichier'),
       (nomproc:'';ligne:12;colonne:poscol;
       chainemenu:'Sortir'));

   carmenu:set of char=
            ['C','F','U','S',carnul,entree];

   codesfleches:set of char=
            [enhaut,agauche,adroite,enbas];
VAR
  fait:boolean;
  selection:activites;

FUNCTION Majuscules(chsaisie:string):string;
  VAR
    i:integer;
    chres:string;
  begin
    chres:='';
    for i:=1 to length(chsaisie) do
      chres:=chres+upcase(chsaisie[i]);
```

```
    majuscules:=chres
  end;


PROCEDURE Surbrillance;
  begin
    reversevideo(true);
    with activite[selection] do
      begin
        gotoxy(colonne,ligne);
        writeln(majuscules(chainemenu))
      end;
    reversevideo(false);
    gotoxy(60,17)
  end;

PROCEDURE Menuinitial;
  VAR
    option:activites;
  begin
    clrscr;

    Reversevideo(true);
    gotoxy(poscol-5,3);
    writeln(' * SIMULATION EN REGIME PERMANANT * ');
    reversevideo(false);

    for option :=saisie_clavier to quitter do
      with activite[option] do
         begin
           gotoxy(colonne,ligne);
           writeln(chainemenu);
         end;

     selection :=saisie_clavier;
     surbrillance;
     gotoxy(poscol-12,16);
     write('Utiliser');
     reversevideo(true);
     write(#24,' ',#25,' ',#26,' ',#27);
     reversevideo(false);
     write(' ou ');
     reversevideo(true);
     write(affichoption);
     reversevideo(false);
     write(' pour selectionner une option , ');

     gotoxy(poscol-10,17);
     write('puis appuyer sur <Entrée> pour  confirmer.)');
   end;

 PROCEDURE Selectionne(var signalfin:boolean);
   CONST
     premcar='CFUS';
   VAR
     carsaisi:char;
   PROCEDURE Continue;
     VAR
       espace:char;
     begin
       gotoxy(10,25);
       write('Appuyer sur la barre d''espacement pour revenir au menu. ');
       repeat
         espace:=readkey
```

```
    until espace=' ';
  end;

PROCEDURE EnleverSurbrillance;
  begin
    with activite[selection] do
      begin
        gotoxy(colonne,ligne);
        writeln(chainemenu)
      end
    end;

PROCEDURE Choixsuivant;
  begin
    enleversurbrillance;
    if selection=quitter then
      selection:=saisie_clavier
    else
      selection:=succ(selection);
    surbrillance
   end;

 PROCEDURE ChoixPrecedent;
   begin
     enleversurbrillance;
     if selection=saisie_clavier then
       selection:=quitter
     else
       selection:=pred(selection);
     surbrillance
   end;

   begin(*selectionne*)
     signalfin:=false;
     repeat
       carsaisi:=upcase(readkey);
       if not (carsaisi in carmenu) then
         write(sonnerie)
     until (carsaisi in carmenu);

  case carsaisi of
    'C','F','U','S' :
    begin
      enleversurbrillance;
      selection:=
            activites(pos(carsaisi,premcar)-1);
      surbrillance
    end;

    carnul :
      begin
        carsaisi:=readkey;
        if carsaisi in codesfleches then
          case carsaisi of
            enhaut,agauche:choixprecedent;
            enbas,adroite:choixsuivant
          end
        else
          write(sonnerie)
       end;

 entree:

 begin
   if selection=quitter then
```

```
      signalfin:=true
    else
      begin
        if selection=saisie_clavier  then
          begin
            saisieclavier;
            debitinitial;
            continue;
            affichage(q1);
            clrscr;
           menuinitial;
          end;
        if selection=sauvegarde then
          begin
            saisieclavier;

            ecriturefichier;
            continue;
            clrscr;
            menuinitial;
          end;
        if selection=lecture then
          begin
            clrscr;
            lecturefichier;
            debitinitial;
            affichage(q1);
            continue;
            clrscr;
            menuinitial;
          end;
       end
     end
  end
end;

begin
 menuinitial;
 repeat
   selectionne(fait);
 until fait;
 clrscr
 end.
```

```
UNIT DECLAR;
  (*declaration des types*)
  INTERFACE

    type                          (* display_menu*)
      menu_rec= record
        num_choices : integer;
        menu_width : integer;
        choices  :array[1..14] of char;
        descriptions:array[1..14] of string;
        title:string;
        prompt:string;
      end;

  tab=array[1..20,1..20] of real;
 vcol=array[1..20] of real;

  IMPLEMENTATION
         end.
```

```
UNIT DECLAR1

   INTERFACE

   USES DECLAR ,
        (*$U b:\graphyd\graph.tpu*)GRAPH;

   const g = 9.81;

   type numero = 0..10;
        ncond  = 0..30;
        pompe  = (pompe1,pompe2);
        tab1=array[1..2,1..10] of real;
        tab2=array[1..10] of real;
        tnmatrix=array[1..20,1..20] of real;
        tnvector=array[1..20] of real;
        forme  = (cylindrique,tronconique,rectangulaire);
        tabcond=array[1..30] of real;
        pu=array[1..10] of real;

        Noeudfixe = record
                      pression,
                      elevation : real;
                            no : numero;
                    end;

        Pipe = record
                 no,
                 no1,
                 no2        :numero;
                 longueur,
                 diametre,
                 rugosite,
                 pertch     :real;
                 npompe     :numero;
                   case po:char of
                     '1' : (puissancep:pu);
                     '2' : (data:tab1);
                end;

        NoeudJonction = record
                          no        : numero;
                          debitext  : real;
                          elevation : real;
                      end;


         Tank = record
                  no              : numero;
                  hautmin,
                  hautmax,
                  debitext:real;
                   case fo:char of
                     'c'   : (diam:real);
                     't'   : (diam1,diam2:real);
                     'r' : (cote1,cote2:real);
                  end;

        Loop = record
                  no     : numero;
                  nloop  : numero;
                  nloop1 : array[numero] of numero;
                  sens   : array[numero] of boolean;
               end;
```

```
        Loopout= record
                   no : numero;
                   nloopout:numero;
                   nloopout1:array[numero] of numero;
                   nof:array[1..2] of numero;
                   sens:array[numero] of boolean;
                 end;


    var
      j,f,l,r,np : numero;
      p          : integer;
      npoints    : integer;


      NF         : File of NoeudFixe;
      Conduite   : File of pipe;
      NJ         : File of NoeudJonction;     (*fichier donnees*)
      Reservoir  : File of Tank;
      Boucle     : File of Loop;
      Boucleext:   File of loopout;

      Nomfichier : string[30];
      i,k,m,n,o,q,s,t: integer;

      pipe1      : pipe;
      nj1        : noeudjonction;
      nf1        : noeudfixe;
      res1       : tank;
      bouc1      : loop;
      boucext1   : loopout;

      pipe2      : array[ncond] of pipe;
      nj2        : array[numero] of noeudjonction;
      nf2        : array[numero] of noeudfixe;
      res2       : array[numero] of tank;
      bouc2      : array[numero] of loop;
      boucext2   : array[numero] of loopout;

      r1,r2,r3,r4,hfe:vcol;
      a:tnmatrix;
     Energie,db,q2:vcol;b,q1:tnvector;
      q11,b1:tab;

     point:array[1..20] of pointtype;
     dessin:file of pointtype;

     x:tab;y,y1,h:vcol;
     erreur:boolean;
     alpha,beta:real;
     deltaq,deltaq1:vcol;
   precision:real;
   mhu:real;

implementation
  end.
```

```
UNIT CRTHYD;
 INTERFACE
   USES  CRT,
         DOS
         DECLAR,
         DECLAR1,
         GESTECR;

     procedure InitialisationFichier;
     procedure EcritureFichier;
     procedure LectureFichier;
     procedure SaisieClavier;


 IMPLEMENTATION

procedure InitialisationFichier;
  begin

    begin(*creation du fichier de données*)
      clrscr;
      write('nom du fichier  ');
      readln(nomfichier);
      assign(NF,nomfichier+'\nf.dat');
      assign(NJ,nomfichier+'\nj.dat');
      assign(conduite,nomfichier+'\cond.dat');
      assign(reservoir,nomfichier+'\res.dat');
      assign(boucle,nomfichier+'\bouc.dat');
      assign(boucleext,nomfichier+'\boucext.dat');
     end;

    begin (*initialisation fichier NoeudJonction*)
      rewrite(nj);
        with nj1 do
         begin
           debitext:=0 ;elevation:=0;
           for i:=1 to j do
             begin
               no:=i;
               write(nj,nj1);
             end;(*for*)
          end;(*with*)
      close(nj);
     end;(*fin initialisation*)

 begin(*initialisationfichier noeudfixe*)
   rewrite(nf);
     clrscr;

     with nf1 do
       begin
        pression:=0;elevation:=0;
        for i:=1 to f do
          begin
            no:=i;
            write(nf,nf1);
          end;(*for*)
       end;(*with*)
  end;(*fin initialisation*)

 begin(*initialisation fichier conduite*)
   rewrite(conduite);
     clrscr;
     with pipe1 do
```

```
        begin
          no1:=0;no2:=0;
          longueur:=0;
          diametre:=0;
          rugosite:=0;
          pertch:=0;
          npompe:=np;
           for k:=1 to np  do
             begin
              case po of
               '1': puissancep[k]:=0;
               '2': begin
                     for i:=1 to 2 do
                       for k:=1 to 10 do
                         data[i,k]:=0;
                    end;
              end;(*case*)
           end;(*for k*)
         end;(*with*)
       end;(*conduite*)

  begin(*initialisation fichier boucle*)
     rewrite(boucle);
       with bouc1 do
         for i:=1 to l do
           begin
             no:=i;
             nloop:=0;
             for k:=1 to nloop do
               nloop1[k]:=0;
             write(boucle,bouc1);

         end;(*for i*)
       end;(*boucle*)

 begin(*initialisation fichier boucle exterieur*)
   rewrite(boucleext);
     with boucext1 do
       for i:=1 to f-1  do
         begin
           no:=i;
           nloopout:=0;
           for k:=1 to nloopout do
            ' nloopout1[k]:=0;
          for k:=1 to 2 do
           nof[k]:=0;
             write(boucleext,boucext1);

         end;(*for i*)
end;(*boucleext*)


begin(*initialisation fichier reservoir*)
  rewrite(reservoir);
    with res1 do
      for i:=1 to r do
        begin
          no:=i;
          hautmin:=0;
          hautmax:=0;
          case fo of
           'c':diam:=0;
           't': begin
                  diam1:=0;
                  diam2:=0;
```

```
                end;
          'r': begin
                 cote1:=0;
                 cote2:=0;
               end;
           end;(*case*)
        end;(*for i*)
      end;(*reservoir*)

   end;(*initialisationFichier*)


  procedure ecritureFichier;
   begin

clrscr;
cadre(5,5,75,20,2);
reversevideo(true);
gotoxy(19,7);
writeln('*** SAUVERGARDE  SUR  FICHIER ***');
reversevideo(false);
repeat
gotoxy(10,12);
write('nom du repertoire :  ');

gotoxy(40,12);
clreol;
readln(nomfichier);
(*$I-*);
mkdir(nomfichier);
(*$I+*);
if ioresult =0 then
  begin
  gotoxy(10,30);
  writeln('le fichier de données  ',nomfichier:33+length(nomfichier),' est creé ');
  gotoxy(20,40);
  writeln('Appuyez sur  <Return>  pour continuer  ');
  readln;
  end
  else
  begin
  gotoxy(10,30);
  writeln('le fichier de données  ',nomfichier,' existe ');
  gotoxy(20,40);
  writeln('Appuyez sur  <Return>  pour continuer ');
  readln;
end;
until ioresult=0;



begin(*conduite*)
      assign(conduite,nomfichier+'\cond.dat');
      rewrite(conduite);
      for i:=1 to p do
        begin
          pipe1:=pipe2[i];
          seek(conduite,i-1);
          write(conduite,pipe1);
          end;(*for i*)
      close(conduite);
    end;(*conduite*)
```

```
    begin(*noeud de jonction*)
      assign(nj,nomfichier+'\nj.dat');
      rewrite(nj);
      for i:=1 to j do
        begin
          nj1:=nj2[i];
          seek(nj,i-1);
          write(nj,nj1);
        end;(*for i*)
     close(nj);
   end;(*noeud de jonction*)

   begin(*noeud fixe*)
     assign(nf,nomfichier+'\nf.dat');
     rewrite(nf);
     for i:=1 to f do
       begin
         nf1:=nf2[i];
         seek(nf,i-1);
         write(nf,nf1);
      end;(*for i*)
      close(nf);
    end;(*noeud fixe*)

   begin(*reservoir*)
     assign(reservoir,nomfichier+'\res.dat');
     rewrite(reservoir);
     for i:=1 to r do
       begin
         res1:=res2[i];
         seek(reservoir,i-1);
         write(reservoir,res1);
       end;(*for i*)
    close(reservoir);
  end;(*reservoir*)

begin(*boucle*)
  assign(boucle,nomfichier+'\bouc.dat');
  rewrite(boucle);
  for i:=1 to l do
    begin
      bouc1:=bouc2[i];
      seek(boucle,i-1);
      write(boucle,bouc1);
    end;(*for i*)
  close(boucle);
end;(*boucle*)

begin(*boucle exterieure*)
  assign(boucleext,nomfichier+'\boucext.dat');
  rewrite(boucleext);
  for i:=1 to f-1 do
    begin
      boucext1:=boucext2[i];
      seek(boucleext,i-1);
      write(boucleext,boucext1);
    end;(*for i*)
   close(boucleext);
 end;(*boucle exterieure*)


  end;(*ecritureFichier*)

procedure lecturefichier;
```

```
begin
 clrscr;
 write('nom du repertoire :  ');
 readln(nomfichier);

begin(*conduite*)
  assign(conduite,nomfichier+'\cond.dat');
  reset(conduite);
  p:=filesize(conduite);
  for i:=1 to p do
    begin
      seek(conduite,i-1);
      read(conduite,pipe1);
      pipe2[i]:=pipe1;
    end;(*for i*)
 end;(*conduite*)

 begin(*noeud de jonction*)
   assign(nj,nomfichier+'\nj.dat');
   reset(nj);
   j:=filesize(nj);
   for i:=1 to j do
     begin
       seek(nj,i-1);
       read(nj,nj1);
       nj2[i]:=nj1;
     end;(*for i*)
  end;(*noeud de jonction*)

   begin(*noeud fixe*)
     assign(nf,nomfichier+'\nf.dat');
     reset(nf);
     f:=filesize(nf);
     for i:=1 to f do
       begin
         seek(nf,i-1);
         read(nf,nf1);
         nf2[i]:=nf1;
       end;(*for i*)
   end;(*noeud fixe*)

   begin(*reservoir*)
     assign(reservoir,nomfichier+'\res.dat');
     reset(reservoir);
     r:=filesize(reservoir);
     for i:=1 to r do
       begin
         seek(reservoir,i-1);
         read(reservoir,res1);
         res2[i]:=res1;
       end;(*for i*)
  end;(*reservoir*)

  begin(*boucle*)
    assign(boucle,nomfichier+'\bouc.dat');
    reset(boucle);
    l:=filesize(boucle);
    for i:=1 to l do
      begin
        seek(boucle,i-1);
        read(boucle,bouc1);
        bouc2[i]:=bouc1;
      end;(*for i*)
 end;(*boucle*)
```

```
   begin(*boucle exterieure*)
     assign(boucleext,nomfichier+'\boucext.dat');
     reset(boucleext);
     f:=filesize(boucleext)+1;
     for i:=1 to f-1 do
       begin
         seek(boucleext,i-1);
         read(boucleext,boucext1);
         boucext2[i]:=boucext1;
       end;(*for i*)
  end;(*boucle exterieure*)

   end;(*lectureFichier*)


procedure saisieclavier;
  var sens1,a:char; a1:boolean;
  begin (*programme principal*)
    begin
      repeat
      clrscr;
      cadre(1,1,79,24,2);
      reversevideo(true);
      gotoxy(23,6);
      write('nombre de conduites           ');
      reversevideo(false);
      readln(p);
      reversevideo(true);
      gotoxy(23,8);
      write('nombre de noeud de jonction    ');
      reversevideo(false);
      readln(j);
      reversevideo(true);
      gotoxy(23,10);
      write('nombre de noeud à niveau fixe  ');
      reversevideo(false);
      readln(f);
      reversevideo(true);
      gotoxy(23,12);
      write('nombre de boucle elementaire  ');
      reversevideo(false);
      readln(l);
      reversevideo(true);
      gotoxy(23,14);
      write('nombre de reservoir           ');
      reversevideo(false);
      readln(r);
      reversevideo(true);
      gotoxy(32,20);
      write('Exact OUI/NON ?  ');
      reversevideo(false);
      a:=readkey;
      if a='o' then
      a1:=true else a1:=false;

   until a1
end;


     clrscr;

  for i:=1 to j do
    repeat
      cadre(26,4,52,14,1);
      reversevideo(true);
```

```
      gotoxy(30,6);
      writeln('NOEUD DE JONCTION ',i);
      reversevideo(false);
      with nj2[i] do
        begin
          gotoxy(28,8);
          write('debit exterieur  ');
          readln(debitext);
          gotoxy(28,10);
          write('elevation  ');
          readln(elevation);
          writeln;
          no:=i;
        end;(*with*)
     reversevideo(true);
     gotoxy(30,12);
     write('Exact OUI/NON ?  ');
     a:=readkey;
     if a='o' then a1:=true else a1:=false;
     reversevideo(false);
     clrscr;
   until a1;

clrscr;

 for i:=1 to f do
   repeat
     cadre(23,4,54,14,1);
     reversevideo(true);
     gotoxy(27,6);
     writeln('NOEUD FIXE  ',i,'  numero ',i+j);
     reversevideo(false);
     with nf2[i] do
       begin
         gotoxy(25,8);
         write('pression  ');
         readln(pression);
         gotoxy(25,10);
         write('elevation  ');
         readln(elevation);
         writeln;
         no:=i+j;
      end;(*with*)
      reversevideo(true);
      gotoxy(27,12);
      write('Exact OUI/NON ?  ');
      a:=readkey;
      if a='o' then a1:=true else a1:=false;
      reversevideo(false);
      clrscr;
    until a1 ;

    clrscr;

    for i:=1 to p do
      repeat
        clrscr;
        cadre(1,4,79,24,2);
        reversevideo(true);
        gotoxy(35,6);
        writeln('CONDUITE  ',i);
        reversevideo(false);
        with pipe2[i] do
          begin
            no:=i;
```

```
        gotoxy(18,8);
        write('noeuds extremes no1 ---> no2  ');
        readln(no1,no2);
        gotoxy(18,10);
        write('longueur ');
        readln(longueur);
        gotoxy(18,12);
        write('diametre ');
        readln(diametre);
        gotoxy(18,14);
        write('rugosite ');
        readln(rugosite);
        gotoxy(18,16);
        write('somme de pertes de charges singulieres ');
        readln(pertch);
        gotoxy(18,18);
        write('nombre de pompes ');
        readln(npompe);
      end;(*with*)
   reversevideo(true);
   gotoxy(32,22);
   write('Exact OUI/NON ?  ');
   reversevideo(false);
   a:=readkey;
   if a='o' then a1:=true else a1:=false;
 until a1;

for i:=1 to p do
  with pipe2[i] do
    for k:=1 to npompe do
      begin
        repeat
          clrscr;
          cadre(4,3,35,11,1);
          reversevideo(true);
          gotoxy(15,4);
          writeln('CONDUITE ',i);
          gotoxy(15,5);
          writeln('POMPE    ',k);
          reversevideo(false);
          gotoxy(5,7);
          writeln('puissance ou caracteristique');
          gotoxy(5,8);
          write('tapez 1 ou 2 ');
          po:=readkey;
          reversevideo(true);
          gotoxy(10,10);
          write('Exact OUI/NON ?  ');
          reversevideo(false);
          a:=readkey;
          if a='o' then a1:=true else a1:=false;
        until a1;
        case po of
             '1' :
               repeat
                 cadre(3,2,72,17,2);
                 cadre(36,12,71,16,1);
                 gotoxy(47,13);
                 write('Puissance ?  ');
                 gotoxy(50,14);
                 clreol;
                 readln(puissancep[k]);
                 reversevideo(true);
                 gotoxy(50,15);
                 write('Exact OUI/NON ?  ');
```

```
                reversevideo(false);
                a:=readkey;
                if a='o' then a1:=true else a1:=false;
              until a1;

            '2' :
              repeat
                cadre(2,12,80,23,1);
                gotoxy(5,13);
                write('nombre de points ?  ');
                gotoxy(45,13);
                writeln('maximum 10 points ');
                gotoxy(50,14);
                clreol;
                readln(npoints);
                gotoxy(30,15);
                writeln('entrez (H,Q) ');
                  for q:=1 to npoints do
                    begin
                      cadre(1,2,80,24,2);
                      cadre(3,16,79,22,1);
                      reversevideo(true);
                      gotoxy(4+8*(q-1),17);
                      writeln('H',q);
                      reversevideo(false);
                      gotoxy(4+8*(q-1),18);
                      clreol;
                      readln(data[1,q]);
                      reversevideo(true);
                      gotoxy(4+8*(q-1),20);
                      writeln('Q',q);
                      reversevideo(false);
                      gotoxy(4+8*(q-1),21);
                      clreol;
                      readln(data[2,q]);
                    end;(*for q*)
                reversevideo(true);
                gotoxy(27,22);
                write('Exact OUI/NON ?  ');
                reversevideo(false);
                a:=readkey;
                if a='o' then a1:=true else a1:=false;
            until a1;
          end;(*case*)
      end;(*for k*)

begin(*reservoir*)
  clrscr;
  for i:=1 to r do
    begin
      repeat
        clrscr;
        cadre(9,5,67,18,1);
        reversevideo(true);
        gotoxy(34,6);
        writeln('RESERVOIR  ',i);
        reversevideo(false);
        with res2[i] do
        begin
          gotoxy(15,8);
          write('numero du noeud fixe correspondant ');
          readln(no);
          gotoxy(15,10);
          write('hauteur maximale                   ');
          readln(hautmax);
```

```
            gotoxy(15,12);
            write('hauteur minimale                    ');
            readln(hautmin);
            gotoxy(15,14);
            write('debit exterieur                     ');
            readln(debitext);
            gotoxy(15,16);
            writeln('forme du reservoir                ');
            gotoxy(15,17);
            reversevideo(true);
            write('cylindrique(C),tronconique(T),rectangulaire(R)
            reversevideo(false);
            fo:=readkey;
            case fo of

             'c':    begin
                       cadre(8,4,68,24,2);
                       cadre(13,19,49,24,1);
                       gotoxy(20,20);
                       write('diametre    ');
                       readln(diam);
                     end;

            't':    begin
                       cadre(8,4,68,24,2);
                       cadre(13,19,49,24,1);
                       gotoxy(20,20);
                       write('diametre 1  ');
                       readln(diam1);
                       gotoxy(20,21);
                       write('diametre 2  ');
                       readln(diam2);
                    end;

            'r':    begin
                       cadre(8,4,68,24,2);
                       cadre(13,19,49,24,1);
                       gotoxy(20,20);
                       write('coté 1      ');
                       readln(cote1);
                       gotoxy(20,21);
                       write('coté 2      ');
                       readln(cote2);
                    end;
              end;(*case*)
           end;(*with*)
           reversevideo(true);
           gotoxy(18,23);
           write('Exact OUI/NON ?  ');
           reversevideo(false);
           a:=readkey;
           if a='o' then a1:=true else a1:=false;
      until a1;
  end;(*for*)
end;(*reservoir*)

begin(*boucle*)
 for i:=1 to l do
   repeat
     clrscr;
     cadre(6,5,72,17,1);
     reversevideo(true);
     gotoxy(35,4);
     writeln('BOUCLE  ',i);
     reversevideo(false);
```

```
      with bouc2[i] do
        begin
          gotoxy(7,6);
            write('nombre des noeuds ');
            readln(nloop);
            for k:=1 to nloop do
              if k<=5 then
                begin
                  gotoxy(7,6+2*k);
                  write('cond ',k,' : ');
                  readln(nloop1[k]);
                end(*if then k*)
              else
                begin
                  gotoxy(22,6+2*(k-5));
                  write('cond ',k,' : ');
                  readln(nloop1[k]);
                end;(*if else k*)
              for k:=1 to nloop do
                begin
                  cadre(16,18,61,23,1);
                  cadre(5,3,73,24,2);
                  reversevideo(true);
                  gotoxy(30,18);
                  writeln('conduite  ',k);
                  reversevideo(false);
                  gotoxy(20,20);
                  writeln('donnez le sens du débit dans la conduite');
                  gotoxy(20,21);
                  writeln('no1--->no2 (1)  no2--->no1 (2)  ');
                  gotoxy(20,22);
                  clreol;
                  write('Tapez 1 ou 2  ');
                  readln(sens1);
                  if sens1='1' then
                     sens[k]:=true
                       else
                          if sens1='2' then
                            sens[k]:=false;
                end;(*with*)
        end;(*for*)
      reversevideo(true);
      gotoxy(30,24);
      write('Exact OUI/NON ?  ');
      reversevideo(false);
      a:=readkey;
      if a='o' then a1:=true else a1:=false;
    until a1
  end;(*boucle*)

  begin(*boucle exterieure*)
    for i:= 1 to f-1 do
      repeat
        clrscr;
        cadre(3,2,76,15,1);
        reversevideo(true);
        gotoxy(30,2);
        writeln('BOUCLE EXTERIEURE  ',i);
        reversevideo(false);
        with boucext2[i] do
          begin
            gotoxy(7,4);
            write('nombre de noeuds  ');
            readln(nloopout);
            nloopout:=nloopout-1;
```

```
          for k:=1 to nloopout do
            begin
              if k<=5 then
                begin
                  gotoxy(7,4+2*k);
                  write('cond ',k,' :  ');
                  readln(nloopout1[k]);
                end;
              if (k>5) and (k<=10) then
                begin
                  gotoxy(22,4+2*(k-5));
                  write('cond ',k,' : ');
                  readln(nloopout1[k]);
                end;
             if (k>10) and (k<=15) then
                begin
                  gotoxy(37,4+2*(k-10));
                  write('cond ',k,' : ');
                  readln(nloopout1[k]);
                end;
            end;(*for k*)
          for k:=1 to 2 do
            begin
              cadre(2,1,77,24,2);
              cadre(31,16,52,19,1);
              gotoxy(33,16+k);
              write('NOEUD FIXE  ',k,'  ');
              readln(nof[k]);
            end;(*for k*)
          for k:=1 to nloopout do
             begin
               cadre(16,20,61,24,1);
               reversevideo(true);
               gotoxy(30,20);
               writeln('conduite  ',k);
               reversevideo(false);
               gotoxy(17,21);
               writeln('sens du débit dans la conduite');
               gotoxy(17,22);
                writeln('Tapez no1--->no2 (1)  no2--->no1 (2) ');
                gotoxy(17,23);
                clreol;
                write('Tapez 1 ou 2  ');
                readln(sens1);
                if sens1='1' then
                  sens[k]:=true
                      else
                        if sens1='2' then
                           sens[k]:=false
              end;(*for k*)
        end;(*with*)
      reversevideo(true);
      gotoxy(30,25);
      write('Exact OUI/NON ? ');
      reversevideo(false);
      a:=readkey;
      if a='o' then a1:=true else a1:=false;
    until a1;
  end;(*boucle exterieure*)
end;(*saisieclavier*)
end.
```

```
UNIT SIMHYD1;

 INTERFACE
   USES CRT,
        DOS,
        DECLAR1,
        MATHTOOLS,
        GESTECR;

     procedure EquationNoeudJonction;
     procedure CalculConduite(var r1:vcol);
     procedure AssemblageBoucle(con:vcol);
     procedure AssemblageBoucleExt(con:vcol);
     function norme(a,b:vcol;p:integer):real;
     procedure DebitInitial;
     procedure affichage(q11:tnvector);
     procedure EcritureMatrice(a1:tnmatrix);
     procedure EcritureColonne( var b:vcol);
     procedure PompeConduite(donnees:tab1;number:numero;
               var alpha:real;var beta:real);
     procedure CalculPompe(var r2:vcol;var r3:vcol);


IMPLEMENTATION

procedure pompeconduite;
  var logh,logq:tab2;
      i,k:integer;
      alpha1,beta1:real;
  begin
    for i:=2 to number do
      begin
        logh[i]:=ln(donnees[1,1]-donnees[1,i]);
        logq[i]:=ln(donnees[2,i]);
      end;
  moindres_carrees(logq,logh,2,number,alpha1,beta1);
  alpha:=exp(beta1);
  beta:=alpha1;
  end;


    procedure  equationNoeudJONCTION;
      begin               (*equation de continuite*)
        for i:=1 to j do
          begin
            b[i]:=nj2[i].debitext;
           for k:=1 to p do
             begin
              a[i,k]:=0;
              if nj2[i].no=pipe2[k].no1 then
                a[i,k]:= -1
                else
                  if nj2[i].no=pipe2[k].no2 then
                    a[i,k]:=1


             end;(*for k*)
        end;(*for i*)
     end;(*equationNoeudJonction*)

   procedure CalculConduite; (*calcul de r[i]*)
   var
     long,dia4,dia5,ki:real;
```

```
 begin(*procedure*)
   for i:=1 to p do
     begin
       with pipe2[i] do
         begin
           long:=longueur;
           dia4:=puissance(diametre,4);
           dia5:=diametre*dia4;
           ki:=pertch;
         end;(*with*)
     r1[i]:=16/(pi*pi*2*g)*(0.025*long/dia5+ki/dia4);
  end;(*for*)
end;(*procedure*)

procedure calculpompe;
var alpha,beta:real;
    k:integer;
begin
  for k:=1 to p do
    with pipe2[k] do
      if npompe <> 0 then
        begin
          case po of
          '1':  begin
            r2[k]:=puissancep[1];
            r3[k]:=0.0
                 end;
          '2': begin
            r2[k]:=alpha;
            r3[k]:=beta
               end;
           end;(*case*)
        end(*if then*)
          else
       begin
         r2[k]:=0.0;
         r3[k]:=0.0;
       end;(*if else*)
  end;(*calculpompe*)

procedure assemblageBoucle;
  var
   n:integer;
  begin           (*equation d'energie*)
    for i:=1 to l do
      begin
        b[i+j]:=0;
        with bouc2[i] do
          begin
            for k:=1 to p  do
            begin
              a[i+j,k]:=0;
              for o:=1 to nloop do
              if nloop1[o]=pipe2[k].no then
                 if sens[o]=true then
                a[i+j,k]:=r4[k]
                   else
                 if sens[o]=false then
               a[i+j,k]:=-r4[k];
            end;(*for k*)
          end;(*for i*)
        end;(*with*)
        end;(*assemblageBoucle*)

procedure assemblageBoucleExt;
```

```
    begin
      for i:=1 to f-1 do
        begin
          b[i+j+l]:=0;
         with boucext2[i] do
           begin
            begin
             for k:=1 to f do
               if nf2[k].no=nof[1] then
                 b[i+j+l]:=nf2[k].pression;
             for k:=1 to f do
               if nf2[k].no=nof[2] then
                 b[i+j+l]:=b[i+j+l]-nf2[k].pression;
             end;
             b[i+j+l]:=b[i+j+l]/981;
           for k:= 1 to p do
             begin
               a[i+j+l,k]:=0;
               for o:=1 to boucext2[i].nloopout do
                 if nloopout1[o] =pipe2[k].no then
                   if sens[o]=true then
                     a[i+j+l,k]:=r4[k]
                       else
                   if sens[o]=false then
                     a[i+j+l,k]:=-r4[k];
             end;(*for k*)
           end;(*for i*)
         end;(*with*)
        end;(*assemblageBoucleExt*)

function norme;
var k:integer;
    res:real;
begin
  res:=0;
  for k:=1 to p do
    res:=res + sqr(a[i]-b[i]);
 norme:=abs(sqrt(res));
 end;

procedure affichage;
 begin(*affichage*)
   clrscr;
   for i:=1 to p do
     begin
       gotoxy(2,3+i);
       write('Debit ',i);
       gotoxy(30,3+i);
       writeln(q1[i]*1.0e3:9:10)
     end;
 end;

procedure ecriturematrice;
 var i,j:integer;
 begin
   for i:=1 to p do
     begin
       writeln;
       for j:=1 to p do
         write(a1[i,j]:3:3);
     end;
   writeln;
 end;

procedure ecriturecolonne;
```

```
  var i:integer;
  begin
  for i:=1 to p do
    writeln(b[i]:3:3);
  end;



procedure debitinitial;
var
 erreur:byte;k:integer;
 iter:integer;
 (*$I GAUSSIDL.INC *)
 begin
   begin(*itération 1*)  for k:=1 to p do q1[k]:=0.0;
     equationNoeudJonction;
     CalculConduite(r1);
     calculpompe(r2,r3);
     for i:=1 to p do
       r4[i]:=r1[i]+r2[i];
     AssemblageBoucle(r4);
     AssemblageBoucleExt(r4); ecriturematrice(a);readln;
     Gauss_Seidel(p,a,b,1e-6,10000,q1,iter,erreur); affichage(q1);
                                    writeln('1');writeln(iter,' ',erreur);
                                    readln;

     for i:=1 to p do
       q11[1,i]:=q1[i]
   end;(*itération 1*)

   begin(*itération 2*)
     for i:=1 to p do
       with pipe2[i] do
         if npompe <> 0 then
           case po of
           '1':r4[i]:=r1[i]*q1[i]+r2[i]/(q1[i]*q1[i]);
           '2':r4[i]:=r1[i]*q1[i]+r2[i]*puissance(q1[i],r3[i]-1);
           end(*case*)
             else
             r4[i]:=r1[i]*q1[i];

     equationNoeudJonction;
     AssemblageBoucle(r4);
     AssemblageBoucleExt(r4);ecriturematrice(a);readln;
     Gauss_Seidel(p,a,b,1e-6,1000,q1,iter,erreur);affichage(q1);
                                  writeln('2');readln  ;

     for i:=1 to p do
       q11[2,i]:=q1[i];

    end;(*itération 2*)

    begin
    k:=0;
    repeat(*itérations*)
      for i:=1 to p do
        begin
          q11[k+3,i] := ( q11[k+2,i] + q11[k+1,i] )  / 2 ;
          q1[i] := q11[k+3,i];
          q2[i] := q11[k+3,i];
        end;
        for i:=1 to p do
          with pipe2[i] do
            if npompe <> 0 then
```

```
            case po of
            '1' : r4[i]:=r1[i]*q2[i]+r2[i]/(q2[i]*q2[i]);
            '2' : r4[i]:=r1[i]*q2[i]+r2[i]*puissance(q2[i],r3[i]-1);
            end(*case*)
                else
            r4[i]:=r1[i]*q2[i];

      equationNoeudJonction;
      AssemblageBoucle(r4);
      AssemblageBoucleExt(r4);writeln(k+3);readln;ecrituretrice(a);readln;
      Gauss_Seidel(p,a,b,1e-6,1000,q1,iter,erreur);affichage(q1);readln;

      for i:=1 to p do
        q11[k+4,i] := q1[i];
      k:=k+1;
   until k=18
end;
  end;
end.
```

```
UNIT SIMHYD2;
    INTERFACE

  USES DECLAR,
       DECLAR1,
       MATTHOOLS,
       SIMHYD1,
       CRT,
       DOS;


  Function DarcyWeibash(debit,diametre,longueur,
                        mhu,rugosite       :real):real;

  Procedure EquationContinuite;

  Procedure DebitInitialH;

  Procedure CoefficientFrottement(var db:vcol);

  Procedure BalancementBoucle( db:vcol;var deltaq:vcol);

  Procedure BalancementBoucleext( db:vcol;var deltaq:vcol);

  Procedure PertesChargesBoucle(db:vcol;var h:vcol);

  Procedure PertesChargesBoucleExt(db:vcol;var h:vcol);

  Function Maximum(a:vcol;nbre:integer):real;

  Procedure Hardy_Cross;


IMPLEMENTATION
  Function DarcyWeibash;
    var  i     : integer;
         f1,f2 : real;

    begin(*function*)
      f1:=0.3164*puissance(4*abs(debit)/(pi*mhu*diametre),-1/4);
      f2:=0.25*sqr(ln(10))/sqr(ln(abs((rugosite/(diametre*3.7)
            +(2.51*pi*mhu*diametre)/(4*debit*sqrt(f1))))));
      while abs(f2-f1) > 0.001 do
        begin
          f1:=f2;
          f2:= 0.25*sqr(ln(10))/sqr(ln(abs((rugosite/(diametre*3.7)
                  +(2.51*pi*mhu*diametre)/(4*debit*sqrt(f1))))));
        end;(*while*)
      darcyweibash:=f2
   end;(*function*)

procedure EquationContinuite;
  var i:integer;   a1:tab;
      k:integer;   b1:vcol;
  (*$I GAUSSIDL.INC *)
  begin
    for i:=1 to j do
      for k:=1 to j do
        a1[i,k]:=a[i,k];

    for i:=1 to j do
     begin
       b1[i]:=0.0;
       for k:=j+1 to p do
```

```
        b1[i]:=b1[i]+a[i,j+k]*b[i];
     end;(* for i*)

  end;(*equationcontinuite*)

Procedure DebitinitialH;
  var i,k,iter:integer;
      erreur:byte;
  (*$I GAUSSIDL.INC *)
  begin
    calculconduite(r1);
    equationNoeudJonction;
    for i:=1 to l do
      for k:=1 to p do

        a[i+j,k]:=r1[i+j];
    for i:=1 to f-1 do
      for k:=1 to p do
        a[i+j+l,k]:=r1[k]/2;
   ecriturematrice(a);readln;
   Gauss_seidel(p,a,b,1e-6,1000,q1,iter,erreur);

   end;



Procedure CoefficientFrottement;
  var i:integer;
      long,dia4,dia5,dia,ki,rug,frott:real;
  begin
    FOR i:=1 TO p DO
     begin
       WITH pipe2[i] DO
         begin
           long:=longueur;
           dia4:=puissance(diametre,4);
           dia5:=dia4*diametre;
           ki:=pertch;
           dia:=diametre;
           rug:=rugosite;
         end;
      frott:=DarcyWeibash(q1[i],dia,long,mhu,rug);writeln('f=',frott:3:3);readln;
      db[i]:=16/(pi*pi*2*g)*(frott*long/dia5+ki/dia4);
    end;(*for i*)
  end;(*procedure*)

procedure BalancementBoucle;
  var num,denom:vcol;
      i,k:integer;

  begin
    FOR i:=1 TO l DO
      begin
        num[i]:=0.0;
        denom[i]:=0.0;
        deltaq[i]:=0.0;
        WITH bouc2[i] DO
          FOR k:=1 TO p DO
            IF pipe2[k].npompe = 0
                  THEN
                 begin
                FOR o:=1 TO nloop DO
                  begin
                   IF nloop1[o]=pipe2[k].no THEN
                    IF sens[o]=true THEN
```

```
                    begin
                      num[i]:=num[i]+db[k]*abs(q1[k])*q1[k];
                      denom[i]:=denom[i]+db[k]*2*abs(q1[k]);
                    end(*if sens then*)
                       ELSE
                    begin
                      num[i]:=num[i]-db[k]*abs(q1[k])*q1[k];
                      denom[i]:=denom[i]-db[k]*2*abs(q1[k]);
                    end;(*if sens else*)
                  end(*for o*)
                 end(*if pipe2[k].no*)
                  ELSE
                 begin
                FOR o:=1 TO nloop DO
                  begin
                   IF nloop1[o]=pipe2[k].no THEN
                    IF sens[o]=true THEN
                      CASE pipe2[k].po OF
                       '1':  begin
                             end;
                       '2':  begin
                               num[i]:=num[i]+r2[k]*puissance(abs(q1[k]),
                                    r3[k]-1)*q1[k]+db[k]*abs(q1[k])*q1[k];
                               denom[i]:=denom[i]-r2[k]*r3[k]*puissance(abs
                                   (q1[k]),r3[k]-1)+db[k]*2*abs(q1[k]);
                    end;(*case*)
                  end(*if*)
                    (*if sens[o]=true*)ELSE
                      CASE pipe2[k].po OF
                        '1':  begin
                              end;
                        '2':  begin
                                num[i]:=num[i]-r2[k]*puissance(abs(q1[k]),
                                     r3[k]-1)*q1[k]-db[k]*abs(q1[k])*q1[k];
                                denom[i]:=denom[i]+r2[k]*r3[k]*puissance(abs
                                    (q1[k]),r3[k]-1)-db[k]*2*abs(q1[k]);
                              end;
                    end;(*case*)
                  end;(*for o*)
        end;(*if pipe2[k].no else*)
deltaq[i]:=-(num[i])/denom[i];
            end;(*for i*)
      end;(*balancementBoucle*)

procedure BalancementBoucleExt;
    var num,denom:vcol;
      i,k:integer;

  begin
    FOR i:=1 TO f-1 DO
      begin
        num[i]:=0.0;
        denom[i]:=0.0;
        WITH boucext2[i] DO
          FOR k:=1 TO p DO
            IF pipe2[k].npompe = 0
                  THEN
                begin
               FOR o:=1 TO nloopout DO
                 begin
                  IF nloopout1[o]=pipe2[k].no THEN
                   IF sens[o]=true THEN
                     begin
                       num[i]:=num[i]+db[k]*abs(q1[k])*q1[k];
                       denom[i]:=denom[i]+db[k]*2*abs(q1[k]);
```

```
                    end(*if sens then*)
                        ELSE
                    begin
                      num[i]:=num[i]-db[k]*abs(q1[k])*q1[k];
                      denom[i]:=denom[i]-db[k]*2*abs(q1[k]);
                    end;(*if sens else*)
                  end(*for o*)
                 end
                  ELSE
                 begin
                FOR o:=1 TO nloopout DO
                  begin
                   IF nloopout1[o]=pipe2[k].no THEN
                    IF sens[o]=true THEN
                      CASE pipe2[k].po OF
                       '1':  begin
                             end;
                       '2':  begin
                               num[i]:=num[i]+r2[k]*puissance(abs(q1[k]),
                                    r3[k]-1)*q1[k]+db[k]*abs(q1[k])*q1[k];
                               denom[i]:=denom[i]-r2[k]*r3[k]*puissance(abs
                                   (q1[k]),r3[k]-1)+db[k]*2*abs(q1[k]);
                    end;(*case*)
                  end(*if*)
                    (*if sens[o]=true*)ELSE
                      CASE pipe2[k].po OF
                        '1':  begin
                              end;
                        '2':  begin
                                num[i]:=num[i]-r2[k]*puissance(abs(q1[k]),
                                     r3[k]-1)*q1[k]-db[k]*abs(q1[k])*q1[k];
                                denom[i]:=denom[i]+r2[k]*r3[k]*puissance(abs
                                    (q1[k]),r3[k]-1)-db[k]*2*abs(q1[k]);
                              end;
                    end;(*case*)
                  end;(*for o*)
                 end;
num[i]:=num[i]+(boucext2[i].nof[1]-boucext2[i].nof[i])/981;
             '    deltaq[i]:=-(num[i])/denom[i];
               end;(*for i*)
       end;(*balancementBoucleExt*)

Procedure PertesChargesBoucle;
   begin
     FOR i:=1 TO l DO
        begin
          H[i]:=0.0;
          WITH bouc2[i] DO
            begin
              FOR k:=1 TO p DO
                FOR o:=1 TO nloop DO
                  IF nloop1[o]=pipe2[k].no THEN
                    IF sens[o]=true THEN

                      H[i]:=H[i]+db[k]*abs(q1[k])*q1[k]
                  (*IF sens[o]=true*)ELSE
                      H[i]:=H[i]-db[k]*abs(q1[k])*q1[k];
              FOR k:=1 TO p DO
                IF pipe2[k].npompe <> 0 THEN
                  FOR o:=1 TO nloop DO
                    IF sens[o]=true THEN
                      CASE pipe2[k].po OF
                        '1': H[i]:=0.0;
                        '2': H[i]:=H[i]-r2[k]*puissance(abs(q1[k]),
                               r3[k]-1)*q1[k];
```

```
                    end(*case*)
                (*IF sens[o]=true*)ELSE
                    CASE pipe2[k].po OF
                      '1':H[i]:=0.0;
                      '2':H[i]:=H[i]+r2[k]*puissance(abs(q1[k]),
                            r3[k]-1)*q1[k];
                      end;(*case*)
          end;(*with*)
       end;(*for i*)
    end;(*procedure*)

Procedure PertesChargesBoucleExt;
   begin
     FOR i:=1 TO f-1 DO
        begin
          H[i]:=0.0;
          WITH boucext2[i] DO
            begin
              FOR k:=1 TO p DO
                FOR o:=1 TO nloopout DO
                  IF nloopout1[o]=pipe2[k].no THEN
                    IF sens[o]=true THEN
                      H[i]:=H[i]+db[k]*abs(q1[k])*q1[k]
                  (*IF sens[o]=true*)ELSE
                      H[i]:=H[i]-db[k]*abs(q1[k])*q1[k];
              FOR k:=1 TO p DO
                IF pipe2[k].npompe <> 0 THEN
                  FOR o:=1 TO nloopout DO
                    IF sens[o]=true THEN
                      CASE pipe2[k].po OF
                        '1': H[i]:=0.0;
                        '2': H[i]:=H[i]-r2[k]*puissance(abs(q1[k]),
                              r3[k]-1)*q1[k];
                        end(*case*)
                  (*IF sens[o]=true*)ELSE
                      CASE pipe2[k].po OF
                        '1':H[i]:=0.0;
                        '2':H[i]:=H[i]+r2[k]*puissance(abs(q1[k]),
                              r3[k]-1)*q1[k];
                        end;(*case*)
              H[i]:=H[i]+nof[1]-nof[2];
            end;(*with*)
        end;(*for i*)
    end;(*procedure*)

Function Maximum;
  var i:integer;
      resultat:real;
  begin
    resultat:=a[1];
    for i:=2 to nbre do
      if a[i] > resultat then
        resultat:=a[i];
     maximum:=resultat;
   end;


procedure Hardy_cross;
  var ki:integer;db1,deltaq1,deltaq2,h1,h2:vcol;
  begin
  debitinitialh;affichage(q1);readln; ki:=0;
  Repeat

    CoefficientFrottement(db1);
    BalancementBoucle(db1,deltaq1);
```

```
    BalancementBoucleExt(db1,deltaq2);
    FOR i:=1 TO l DO
      WITH bouc2[i] DO
        FOR k:=1 TO p DO
          FOR o:=1 TO nloop DO
            IF nloop1[o]=pipe2[k].no THEN
              CASE sens[o] OF
             true:   q1[k]:=q1[k]+deltaq1[i];
             false:  q1[k]:=q1[k]-deltaq1[i];
              end;(*case*)
             affichage(q1);writeln(ki);readln;


    FOR i:=1 TO f-1 DO
      WITH boucext2[i] DO
        FOR k:=1 TO p DO
          FOR o:=1 TO nloopout DO
            IF nloopout1[o]=pipe2[k].no THEN
              q1[k]:=q1[k]+deltaq2[i];

    PertesChargesBoucle(db1,h1);
    PertesChargesboucleExt(db1,h2);
       ki:=ki+1; writeln(maximum(h1,l));readln;
 until (ki=20) or (abs(maximum(h1,l)) < 0.01)
               or (abs(maximum(deltaq1,l))<0.0001);

end;
end.
```

```
UNIT SIMHYD3;
  INTERFACE
    Uses Declar,
         Declar1,

    Procedure CalculVitesse(var vitesse :vcol);
    Procedure CalculPertesdeCharges(var pertecharge : vcol);
    Procedure EnergieReseau(var energie,niveau:vcol);

  IMPLEMENTATION

    Procedure CalculVitesse;
      begin
        for i:=1 to p do
          with pipe2[i] do
            vitesse[i]:=4*q1[i]/(pi*puissance(diametre,2));
       end;

    Procedure PertesdeCharges;
      begin
        coefficientfrottement(db);
        for i:=1 to p do
          with pipe2[i] do
            pertechage[i]:=(db[i]*longueur/diametre +pertch)
                        *puissance(vitesse[i],2)/(2*g);

    Procedure EnergieReseau;
      begin
        CalculPertesdeCharges(hfe);
        For k:=1 to f-1 do
          with boucext2[k] do
            begin
              for i:=1 to f do
                if nof[1]=nf2[i].no then
                  begin
                    niveau[nf2[i].no]:=nf2[i].pression
                              /gamma;
                    energie[nf2[i].no]:=niveau[nf2[i].no] +
                           nf2[i].elevation;
                  end
                  else
                  if nof[2]:=nf2[i].no then
                    begin
                      niveau[nf2[i].no]:=nf2[i].pression
                                  /gamma;
                      energie[nf2[i].no]:=niveau[nf2[i].no] +
                             nf2[i].elevation
                    end;


              for i:=1 to nloopout do
                if pipe2[nloopout1[i]].no1=nof[1] then
                  begin
                    temp1:=nof[1];
                    temp2:=pipe2[nloopout1[i]].no2
                    temp:=pipe2[nloopout1[i]].no
                    if sens[i]=true then
                      begin
                        niveau[temp2]:=niveau[temp1]+gamma/(2*g)*
                          (puissance(vitesse[temp1],2)
                          -puissance(vitesse[temp2],2))
                          +gamma*(nf2[temp1].elevation-
                          nf2[temp2].elevation) -
                          gamma*hfe[temp];
```

```
          energie[temp2]:=niveau[temp2] +
                    nf2[temp2].elevation;
        end
        else(*if sens[i]=false*)
        begin
          niveau[temp2]:=niveau[temp1]+gamma/(2*g)*
            (puissance(vitesse[temp1],2)
            -puissance(vitesse[temp2],2))
            +gamma*(nf2[temp1].elevation-
            nf2[temp2].elevation) +
            gamma*hfe[temp];

          energie[temp2]:=niveau[temp2] +
                    nf2[temp2].elevation;
        end;
  end;


if pipe2[nloopout1[i]].no2=nof[1] then
  begin
    temp1:=nof[1];
    temp2:=pipe2[nloopout1[i]].no1
    temp:=pipe2[nloopout1[i]].no
    if sens[i]=true then
      begin
        niveau[temp2]:=niveau[temp1]+gamma/(2*g)*
          (puissance(vitesse[temp1],2)
          -puissance(vitesse[temp2],2))
          +gamma*(nf2[temp1].elevation-
          nf2[temp2].elevation) -
          gamma*hfe[temp];

         energie[temp2]:=niveau[temp2] +
                    nf2[temp2].elevation;
        end
        else(*if sens[i]=false*)
        begin
          niveau[temp2]:=niveau[temp1]+gamma/(2*g)*
            (puissance(vitesse[temp1],2)
            -puissance(vitesse[temp2],2))
            +gamma*(nf2[temp1].elevation-
            nf2[temp2].elevation) +
            gamma*hfe[temp];

          energie[temp2]:=niveau[temp2] +
                    nf2[temp2].elevation;
        end;
  end;
```

```
UNIT GRAPHYD1;

  INTERFACE

   uses (*$u b:graphyd\graph.tpu*) GRAPH,
        CRT,declar,declar1,
        crthyd;

const trait:array[1..2,1..6] of char=((#218,#191,#192,#217,#196,#179),
                          (#201,#187,#200,#188,#205,#186));

Procedure ReverseVideo(etat:boolean);
Procedure ligne(direction:boolean;type_t:byte;a,b,c:integer);
  (* direction=true  --> a=x1,b=x2,c=y   horizontal  *)
  (* direction=false --> a=y1,b=y2,c=x   vertical    *)

Procedure Mode_Graphique;

 procedure DDA(x1,y1,x2,y2:integer);

procedure mouvement;

procedure photo(var imecran:pointer);

procedure cliche(imecran:pointer);

procedure ecran;

procedure tracage;

procedure putpompe;

Procedure AfficheConduite;

Procedure AfficheNoeudJonction;

Procedure AfficheNoeudfixe;

Procedure AfficheReservoir;

Procedure CadreGraphique;

Procedure Legende1;

Procedure Legende2;

Procedure Legende4;

Procedure Legende5;

Procedure Legende3;

Procedure Sauvegardedessin;

Procedure LectureDessin;

Procedure PreProcesseurGraphique;

IMPLEMENTATION


procedure reverseVideo;

   begin
```

```
    if etat then
      begin
        textcolor(black);
        textbackground(white);
      end
    else
      begin
        textcolor(white);
        textbackground(black);
      end;
  end;

Procedure ligne;
    var i:integer;
  begin
    if direction then
      begin
        for i:=a to b do
          begin
            gotoxy(i,c);
            write(trait[type_t,5]);
          end;
      end
    else
      begin
        for i:=a to b do
          begin
            gotoxy(c,i);
            write(trait[type_t,6]);
          end;
       end;
   end;



Procedure Mode_Graphique;
    var graphmode,
        graphdriver:integer;
    begin
      graphmode:=detect;
      detectgraph(graphdriver,graphmode);
      initgraph(graphmode,graphdriver,'b:\graphyd\');
      if graphresult <> 0 then
        begin
          writeln('code erreur ',graphresult);
          halt(1);
        end;
     end;

  procedure DDA;
    var
      dx,dy,steps,k:integer;
      x_inc,y_inc:real;
      x,y:real;
    begin
      dx:=x2-x1;
      dy:=y2-y1;
      if abs(dx) > abs(dy) then steps:=abs(dx)
        else steps:=abs(dy);
      x_inc:=x/steps;
      y_inc:=y/steps;
      x:=x1;y:=y1;
      putpixel(round(x),round(y),1);
      for k:=1 to steps do begin
        x:=x+x_inc;
```

```
        y:=y+y_inc;
        putpixel(round(x),round(y),1);
      end;
    end;

procedure mouvement;
 var x,y,xa,ya:integer;
     p,ce:integer;
     car:char;
begin    x:=100;y:=100;
  ce:=getcolor;
  repeat
    p:=getpixel(x,y);
    repeat
      putpixel(x,y,0);
      putpixel(x,y,ce);
      delay(2);
    until keypressed;
    putpixel(x,y,p);
    car:=readkey;
    if car=#0 then
      begin
        car:=readkey;
        case car of
        #75: begin
               x:=x-1;
               car:=readkey;
               if car=#13 then outtextxy(x,y,'+');
             end;

        #77: begin
              x:=x+1;
              car:=readkey;
              if car=#13 then outtextxy(x,y,'+');
             end;
        #72: begin
               y:=y+1;
               car:=readkey;
               if car=#13 then outtextxy(x,y,'+');
             end;
        #80: begin
              y:=y-1;
              car:=readkey;
              if car=#13 then outtextxy(x,y,'+');
             end;
      end;

    end;
         until car='F';

end;

procedure photo(var imecran:pointer);
  var  tailleecran:word;

  begin
    tailleecran:=imagesize(0,0,getmaxx,getmaxy);
    getmem(imecran,tailleecran);
    getimage(0,0,getmaxx,getmaxy,imecran^);
readln;
closegraph;
    restorecrtmode;
  end;

procedure cliche(imecran:pointer);
```

```
var graphmode,graphdriver:integer;
begin
    detectgraph(graphmode,graphdriver);
    initgraph(graphmode,graphdriver,'b:\graphyd\');
    putimage(0,0,imecran^,normalput);
  end;

var imecran:pointer;

procedure ecran;
  var x,y,xa,ya:integer;    car:char; xst1,xst2:string;
  begin
    x:=100;y:=5;
for i:=1 to j+f do
  begin

      outtextxy(x,y,'*');

    car:=readkey;
    if car =#0 then
repeat
  car:=readkey;
  case car of
      #75:begin
            x:=x-10;
            putpixel(x,y,1);
            moveto(x,y);
            str(getx,xst1);
            str(gety,xst2);
            setviewport(200,318,400,340,false);
            outtextxy(200,320,xst1);
            outtextxy(200,336,xst2);
            clearviewport;
            setviewport(0,0,getmaxx,getmaxy,true);
            car:=readkey;
            if car =#13 then
              begin
                outtextxy(x,y,'+');
                moveto(x,y);
                point[i].x:=getx;
                point[i].y:=gety;
              end;
            end;
      #77:begin
           x:=x+10;
           putpixel(x,y,1);
           moveto(x,y);
           str(getx,xst1);
           str(gety,xst2);
           setviewport(200,318,400,340,false);
           outtextxy(200,320,xst1);
           outtextxy(200,336,xst2);
           clearviewport;
           setviewport(0,0,getmaxx,getmaxy,true);
           car:=readkey;
           if car=#13 then
             begin
               outtextxy(x,y,'+');
               moveto(x,y);
               point[i].x:=getx;
               point[i].y:=gety;
             end;
           end;

       #72:begin
```

```
          y:=y-10;
          putpixel(x,y,1);
          moveto(x,y);
          str(getx,xst1);
          str(gety,xst2);
          setviewport(200,318,400,340,false);
          outtextxy(200,320,xst1);
          outtextxy(200,336,xst2);
          clearviewport;
          setviewport(0,0,getmaxx,getmaxy,true);
          car:=readkey;
          if car=#13 then
            begin
              outtextxy(x,y,'+');
              moveto(x,y);
              point[i].x:=getx;
              point[i].y:=gety;
            end;
          end;
      #80:begin
           y:=y+10;
           putpixel(x,y,1);
           moveto(x,y);
           str(getx,xst1);
           str(gety,xst2);
           setviewport(200,318,400,340,false);
           outtextxy(200,320,xst1);
           outtextxy(200,336,xst2);
           clearviewport;
           setviewport(0,0,getmaxx,getmaxy,true);
           car:=readkey;
           if car=#13 then
             begin
               outtextxy(x,y,'+');
               moveto(x,y);
               point[i].x:=getx;
               point[i].y:=gety;
             end;
          end
        end;(*case*)


  until car='f';
end;

end;

procedure tracage;
var a1,a2,b1,b2:integer;
begin
    for i:=1 to p do
     begin
       with pipe2[i] do

         for k:=1 to j+f do
           if no1=k then
             begin
               a1:=point[k].x;
               b1:=point[k].y;
             end
           else
             if no2=k then
               begin
                 a2:=point[k].x;
                 b2:=point[k].y;
```

```
            end;
        line(a1,b1,a2,b2);
      end;
    end;


procedure putpompe;

  var a1,a2,b1,b2:integer;
begin
for i:=1 to p do
   begin
     with pipe2[i] do
       if npompe <> 0 then

           for k:=1 to j+f do
            if no1=k then
              begin
                a1:=point[k].x;
                b1:=point[k].y;
              end
            else
              if no2=k then
                begin
                  a2:=point[k].x;
                  b2:=point[k].y;
                end;
         circle(round((a1+a2)/2),round((b1+b2)/2),12);
      end;
   end;

Procedure AfficheConduite;
  begin
    reversevideo(true);
    gotoxy(32,1);
    write('*** CONDUITE ***');
    reversevideo(false);
    ligne(true,2,2,78,5);
    ligne(false,1,3,22,15);
    ligne(false,1,3,22,28);
    ligne(false,1,3,22,41);
    ligne(false,1,3,22,54);
    ligne(false,1,3,22,67);
    gotoxy(4,3);
    write('conduite');
    gotoxy(17,3);
    write('longueur');
    gotoxy(20,4);
    write('m');
    gotoxy(30,3);
    write('diametre');
    gotoxy(32,4);
    write('mm');
    gotoxy(43,3);
    write('rugosité');
    gotoxy(46,4);
    write('mm');
    gotoxy(56,3);
    write('nombre de');
    gotoxy(58,4);
    write('pompe');
    gotoxy(68,3);
    write('pertes de ');
    gotoxy(69,4);
    write('charges');
```

```
    for i:=1 to p do
      with pipe2[i] do
        begin
          gotoxy(5,5+i);
          write(no:4);
          gotoxy(19,5+i);
          write(longueur:4);
          gotoxy(32,5+i);
          write(diametre:4);
          gotoxy(45,5+i);
          write(rugosite:4);
          gotoxy(60,5+i);
          write(npompe:4);
          gotoxy(70,5+i);
          write(pertch:4);
        end;
    end;

Procedure AfficheNoeudJonction;
  begin
    gotoxy(27,1);
    reversevideo(true);
    write('*** NOEUDS DE JONCTION ***');
    reversevideo(false);
    ligne(true,2,5,75,5);
    ligne(false,1,3,20,15);
    ligne(false,1,3,20,45);
    ligne(false,1,3,20,30);
    ligne(false,1,3,20,60);
    gotoxy(7,3);
    write('Noeud');
    gotoxy(18,3);
    write('Débit');
    gotoxy(18,4);
    Write('litre/s');
    gotoxy(32,3);
    write('elevation');
    gotoxy(36,4);
    write('m');
    gotoxy(55,3);
    write('coordonées');
    gotoxy(47,4);
    write('abscisse');
    gotoxy(62,4);
    write('ordonnée');
   for i:=1 to j do
     with nj2[i] do
       begin
         gotoxy(8,5+i);
         write(no:2);
         gotoxy(18,5+i);
         write(debitext:4:1);
         gotoxy(34,5+i);
         write(elevation:4:1);
         gotoxy(48,5+i);
         write(point[i].x:3);
         gotoxy(63,5+i);
         write(point[i].y:3);
       end;
end;

Procedure AfficheNoeudfixe;
  begin
    gotoxy(30,1);
    reversevideo(true);
```

```
    write('*** NOEUDS FIXES ***');
    reversevideo(false);
    ligne(true,2,5,75,5);
    ligne(false,1,3,20,15);
    ligne(false,1,3,20,30);
    ligne(false,1,3,20,45);
    ligne(false,1,4,20,60);
    gotoxy(6,3);
    write('Noeud fixe');
    gotoxy(18,3);
    write('elevation');
    gotoxy(20,4);
    write('m');
    gotoxy(33,3);
    write('pression');
    gotoxy(38,4);
    write('m');
    gotoxy(55,3);
    write('coordonées');
    gotoxy(47,4);
    write('abscisse');
    gotoxy(62,4);
    write('ordonnée');
    for i:=1 to f do
      with nf2[i] do
        begin
          gotoxy(10,5+i);
          write(i:2);
          gotoxy(17,5+i);
          write(elevation:4:1);
          gotoxy(32,5+i);
          write(pression:4:1);
          gotoxy(47,5+i);
          write(point[i+j].x:3);
          gotoxy(62,5+i);
          write(point[i+j].y:3);
        end;
  end;

Procedure AfficheReservoir;
  begin
    gotoxy(30,1);
    reversevideo(true);
    write('***  RESERVOIR ***');
    reversevideo(false);
    ligne(true,2,1,79,5);
    ligne(false,1,3,20,5);
    ligne(false,1,3,20,10);
    ligne(false,1,3,20,25);
    ligne(false,1,3,20,40);
    ligne(false,1,4,20,55);
    ligne(false,1,3,20,60);
    gotoxy(3,3);
    write('no');
    gotoxy(7,3);
    write('no');
    gotoxy(7,4);
    write('NF');
    gotoxy(13,3);
    write('Hmin');
    gotoxy(14,4);
    write('m');
    gotoxy(28,3);
    write('Hmaxi');
    gotoxy(30,4);
```

```
    write('m');
    gotoxy(42,3);
    write('Débit');
    gotoxy(42,4);
    write('litres/s');
    gotoxy(55,3);
    write('forme');
    gotoxy(63,3);
    write('Dimensions');
    for i:=1 to r do
      with res2[i] do
        begin
          gotoxy(2,5+i);
          write(i:2);
          gotoxy(6,5+i);
          write(no:2);
          gotoxy(12,5+i);
          write(hautmin:4:1);
          gotoxy(27,5+i);
          write(hautmax:4:1);
          gotoxy(42,5+i);
          write(debitext);
          gotoxy(57,5+i);
          case fo of
            'c':write('C');
            't':write('T');
            'r':write('R');
          end;(*case*)
          case fo of
            'c': begin
                   gotoxy(70,5+i);
                   write(diam:2:1);
                 end;
            't': begin
                  gotoxy(61,5+i);
                  write(diam1:2:1);
                  gotoxy(71,5+i);
                  write(diam2);
                end;
            'r': begin
                   gotoxy(61,5+i);
                   write(cote1:2:1);
                   gotoxy(71,5+i);
                   write(cote2:2:1);
                 end;
           end;(*case*)
        end(*with*)
     end;

Procedure CadreGraphique;
begin
rectangle(0,0,719,347);
rectangle(2,2,100,313);
rectangle(2,315,717,345);
end;

Procedure Legende1;


begin
  outtextxy(7,10,'COMMANDES');
  line(2,25,100,25);
  outtextxy(4,60,'deplacement');
  outtextxy(4,80,#24+' '+#25+' '+#26+' '+#27);
  outtextxy(6,120,'Placer');
```

```
  outtextxy(8,135,'un');
  outtextxy(5,150,'noeud');
  outtextxy(5,165,'<Return>');
  outtextxy(4,200,'Confirmer');
  outtextxy(5,215,'<Espace>');
  outtextxy(50,320,'abscisse = ');
  outtextxy(50,336,'ordonnee=  ');

end;

Procedure Legende2;
  begin
    setviewport(2,2,99,312,false);
    clearviewport;
    setviewport(0,0,getmaxx,getmaxy,false);
    outtextxy(7,10,'COMMANDES');
    line(2,25,100,25);
    outtextxy(4,50,'Donnees');
    outtextxy(4,65,'Numeriques');
    outtextxy(4,100,'<C>onduite');
    outtextxy(4,150,'<N>oeud');
    outtextxy(8,165,'de');
    outtextxy(4,180,'jonction');
    outtextxy(4,230,'Noeud');
    outtextxy(4,245,'<F>ixe');
    outtextxy(4,295,'<R>eservoir');
  end;
Procedure Legende4;
  begin
    setviewport(2,2,99,312,false);
    clearviewport;
    setviewport(0,0,getmaxx,getmaxy,false);
    outtextxy(7,10,'COMMANDES');
    line(2,25,100,25);
    outtextxy(4,50,'Sauvegarde');
    outtextxy(6,65,'dessin');
    outtextxy(8,80,'<F1>');
    outtextxy(4,130,'Redessiner');
    outtextxy(8,145,'<F2>');
    outtextxy(4,195,'Donnees');
    outtextxy(4,210,'numeriques');
    outtextxy(8,225,'<F3>');
 end;

Procedure Legende5;
  begin
    gotoxy(27,6);
    reversevideo(true);
    write('*** SAUVEGARDE DESSIN ***');
    reversevideo(false);
    ligne(true,2,4,76,8);
    gotoxy(10,12);
    write('Dessin ',nomfichier,+'\dessin.dat');
    gotoxy(10,14);
    write('en cours de sauvegarde');
    ligne(true,2,4,76,20);
   gotoxy(10,17);
   write('Appuyez sur <Return> pour revenir');
   gotoxy(10,18);
   write('à l''interface graphique');
end;



Procedure Legende3;
```

```
  var xst1:string; a,b:integer;
  begin
    for i:=1 to j do
      begin
        str(i,xst1);
        outtextxy(point[i].x+5,point[i].y-5,'NJ'+xst1);
      end;
    for i:=1 to f do
      begin
        str(i,xst1);
        outtextxy(point[i+j].x+5,point[i+j].y-5,'NF'+xst1);
      end;
    for i:=1 to p do
      with pipe2[i] do
        begin
          str(no,xst1);
          for k:=1 to j+f do
            if no1=k then
              a:=no1
            else
              if no2=k then
                b:=no2;
            outtextxy(round((point[a].x+point[b].x)/2),
                      round((point[a].y+point[b].y)/2),xst1);
          end;
  outtextxy(350,325,'Legende');
  outtextxy(450,318,'NJ --> Noeud de jonction');
  outtextxy(450,336,'NF --> Noeud fixe');
  outtextxy(5,250,'Menu suivant');
  outtextxy(5,265,'<Return>');
end;


Procedure sauvegardedessin;
  begin
    assign(dessin,nomfichier+'\dessin.dat');
    rewrite(dessin);
      for i:=1 to j+f do
        begin
          seek(dessin,i-1);
          write(dessin,point[i]);
        end;
     close(dessin);
  end;

Procedure LectureDessin;
  begin
    assign(dessin,nomfichier+'\dessin.dat');
    reset(dessin);
      for i:=1 to j+f do
        begin
          seek(dessin,i-1);
          read(dessin,point[i]);
        end;
end;

Procedure PreprocesseurGraphique;

var car :char;

begin
lecturefichier;
mode_graphique;
cadreGraphique;
```

```
legende1;
ecran;
tracage;
legende3;
readln;
legende4;

REPEAT
car:=readkey;
if car=#0 then
  car:=readkey;
  case car of
  #59: begin
        photo(imecran);
        sauvegardedessin;
        legende5;
        readln;
        cliche(imecran);
        setviewport(2,100,717,313,false);
        clearviewport;
        setviewport(0,0,getmaxx,getmaxy,true);
        cadregraphique;
        tracage;
        legende4;
        legende3;
        car:=readkey;
       end;

#60: begin
       setviewport(2,315,717,313,false);
       clearviewport;
       setviewport(0,0,getmaxx,getmaxy,true);
       cadregraphique;
       legende1;
       ecran;
       tracage;
       legende3;
       legende4;
       car:=readkey;
      end;
#61: begin
       setviewport(2,315,717,313,false);
       clearviewport;
       setviewport(0,0,getmaxx,getmaxy,true);
       cadregraphique;
       legende1;
       tracage;
       legende3;
       legende2;
       car:=readkey;
       case car of
       'C': begin
             photo(imecran);
             affichecondu ite;
             readln;
             clrscr;
             affichenoeudjonction;
             readln;
             clrscr;
             affichenoeudfixe;
             readln;
             clrscr;
             affichereservoir;
             readln;
             cliche(imecran);
```

```
            legende2;
            car:=readkey;
          end;
        'N': begin
            photo(imecran);
            affichenoeudjonction;
            readln;
            afficheconduite;
            readln;
            clrscr;
            affichenoeudfixe;
            readln;
            clrscr;
            affichereservoir;
            cliche(imecran);
            legende2;
            car:=readkey;
          end;
      'F': begin
            photo(imecran);
            affichenoeudfixe;
            readln;
            afficheconduite;
            readln;
            clrscr;
            affichenoeudjonction;
            readln;
            clrscr;
            affichereservoir;
            readln;
            cliche(imecran);
            legende2;
            car:=readkey;
          end;
     'R': begin
            photo(imecran);
            affichereservoir;
            readln;
            clrscr;
            afficheconduite;
            readln;
            clrscr;
            affichenoeudjonction;
            readln;
            clrscr;
            affichenoeudfixe;
            readln;
            cliche(imecran);
            legende2;
            car:=readkey;
          end;
    end;(*case*)
   end;(* #61*)
   end;(*case*)

  until car='a';
  closegraph;
  restorecrtmode;

end;
end.
```

```
UNIT NUMHYD;

  INTERFACE
    uses (*$u b:graphyd\graph.tpu*) GRAPH,
         CRT,declar,declar1,
         crthyd,Graphyd1;

var xst1,xst2,xst3,xst4,xst5 :string;
Procedure Affiche;

IMPLEMENTATION

 Procedure Affiche;

begin
lecturefichier;
Mode_graphique;
  rectangle(0,0,719,347);
  rectangle(2,2,717,345);
  rectangle(4,4,715,343);
  outtextxy(240,30,'** AFFICHAGE DES RESULTATS **');
  line(37,50,678,50);
  line(200,40,300,40);
  line(360,40,300,40);
  line(560,40,300,40);
  line(200,50,200,300);
  line(360,50,360,300);
  line(520,50,520,300);
  outtextxy(80,65,'Conduite');
  outtextxy(215,65,'Debit (litre/s)');
  outtextxy(400,65,'Sens');
  outtextxy(535,65,'Charge (m)');
  line(37,75,678,75);

For i:=1 to p do
  with pipe2[i] do
    begin
      str(no,xst1);
      str(q1[i],xst2);
      str(no1,xst3);
      str(no2,xst4);
      str(energie[i],xst5);
      outtextxy(80,70+15*i,xst1);
      outtextxy(220,70+15*i,xst2);
      outtextxy(400,70+15*i,xst3+' ------>  '+xst4);
      outtextxy(560,70+15*i,xst5);
    end;
outtextxy(100,320,'Appuyez sur <Return> pour visualiser le reseau');
readln;
lecturedessin;
clearviewport;
cadregraphique;
legende3;
tracage;
outtextxy(4,100,'Appuyez sur');
outtextxy(4,150,'  <Return> ');
outtextxy(4,200, 'pour sortir');
readln;


  restorecrtmode;
end;
end.
```

```
UNIT MATHTOOLS;
 (*   Bibliothéque d'utilitaires mathémathiques
      ecrit par Kambika ZOOI,polytechnicien      *)
  INTERFACE
  Uses CRT,DOS,declar1;

   FUNCTION puissance(a,b:real):real;
   (*calcul la puissance b d'un nombre a(>0)*)
   FUNCTION log(arg,base:real):real;
   (*calcul le logarithme de a (base b) *)
   PROCEDURE Iswap(var a,b:integer);
   PROCEDURE Rswap(var a,b:real);
   (*permutation des deus nombres*)
   FUNCTION tan(x:real):real;
   (*tangente d'un nombre en radian*)
   FUNCTION sinus(x:real):real;
   (*sinus de x en degrés*)
   FUNCTION cosinus(x:real):real;
   (*cosinus de x en degrés*)
   FUNCTION rad(nombdeg:real):real;
   (*conversion degrés --> radians*)
   FUNCTION deg(nombrad:real):real;
   (*conversion radians --> degrés*)
   PROCEDURE MOINDRES_CARREES(x,y:tab2;lowerbound,upperbound:integer;
                              var slope:real;
                              var intercept:real);
IMPLEMENTATION
   function puissance;
     var res1:real;
     begin(*puissance*)
       if a>0 then
         begin
           res1:=exp(b*ln(a));
           puissance := res1;
         end
        else
          puissance:=0.0;

     end;(*puissance*)

   function log;
     var res2:real;
     begin(*log*)
       res2:=ln(arg)/ln(base);
       log:=res2;
     end;(*log*)

   procedure Iswap;
     var temp:integer;
     begin
       temp:=a;a:=b;b:=temp;
     end;

   procedure Rswap;
     var temp:real;
     begin
       temp:=a;a:=b;b:=temp;
     end;

   function tan;
     var tan1:real;
     begin
       tan1:=sin(x)/cos(x);
       tan:=tan1;
```

```
  end;

function rad;
  var rrad:real;
  begin
    rrad:=pi*nombdeg/180;
    rad:=rrad;
  end;

function deg;
  var rdeg:real;
  begin
    rdeg:=nombrad*180/pi;
    deg:=rdeg;
  end;

function sinus;
  var rsinus:real;
  begin
    rsinus:=sin(rad(x));
    sinus:=rsinus;
  end;

function cosinus;
  var rcosinus:real;
  begin
    rcosinus:=cos(rad(x));
    cosinus:=rcosinus;
  end;

Procedure Moindres_carrees;
  var
    index,num_elements :integer;
    x_sum,y_sum,xsquared_sum,xy_sum:real;
  begin
    x_sum:=0.0;
    y_sum:=0.0;
    xsquared_sum:=0.0;
    xy_sum:=0.0;
    FOR index := lowerbound TO upperbound DO
      BEGIN
        x_sum:=x_sum+x[index];
        y_sum:=y_sum+y[index];
        xsquared_sum:=xsquared_sum+(x[index]*x[index]);
        xy_sum:=xy_sum+(x[index]*y[index]);
      END;
    num_elements:=abs(upperbound-lowerbound)+1;
    intercept:=(num_elements*xy_sum-x_sum*y_sum)/
               (num_elements*xsquared_sum-x_sum*x_sum);
    slope:=y_sum/num_elements-intercept*x_sum/num_elements;
  end;

 end.
```

```
PROCEDURE Gauss_Seidel(Dimen       : integer;
                       Coefficients : TNmatrix;
                       Constants    : TNvector;
                       Tol          : real;
                       MaxIter      : integer;
                   var Solution     : TNvector;
                   var Iter         : integer;
                   var Error        : byte);

{------------------------------------------------------------------------}
{-                                                                      -}
{-     Turbo Pascal Numerical Methods Toolbox                           -}
{-     (C) Copyright 1986 Borland International.                        -}
{-                                                                      -}
{-                Input: Dimen, Coefficients, Constants, Tol, MaxIter   -}
{-               Output: Solution, Iter, Error                          -}
{-                                                                      -}
{-             Purpose : Calculate the solution of a linear set of      -}
{-                       equations using Gauss - Seidel iteration.      -}
{-                                                                      -}
{-  User-defined Types : TNvector = array[1..TNArraySize] of real       -}
{-                       TNmatrix = array[1..TNArraySize] of TNvector   -}
{-                                                                      -}
{-    Global Variables : Dimen : integer;         Dimen of the square   -}
{-                                                matrix                -}
{-                       Coefficients : TNmatrix; Square matrix         -}
{-                       Constants    : TNvector; Constants of each equation-}
{-                       Tol          : real;     Tolerance in answer   -}
{-                       MaxIter      : integer;  Maximum number of     -}
{-                                                iterations allowed    -}
{-                       Solution     : TNvector; Unique solution to the-}
{-                                                set of equations      -}
{-                       Iter         : integer;  Number of iterations  -}
{-                       Error        : integer;  Flags if something goes-}
{-                                                wrong.                -}
{-                                                                      -}
{-              Errors : 0: No errors;                                  -}
{-                       1: Iter >= MaxIter and matrix not              -}
{-                          diagonally dominant                         -}
{-                       2: Iter >= MaxIter                             -}
{-                       3: Dimen < 1                                   -}
{-                       4: Tol <= 0                                    -}
{-                       5: MaxIter < 0                                 -}
{-                       6: Zero on the diagonal of                     -}
{-                          the Coefficients matrix                     -}
{-                       7: Diverging                                   -}
{-                                                                      -}
{-                 Note: If the Gauss-Seidel iterative method is        -}
{-                       applied to an underdetermined system of equations-}
{-                       (i.e. one of the equations is a linear         -}
{-                       superposition of the others) it will converge  -}
{-                       to a (non-unique) solution. The Gauss-Seidel   -}
{-                       method is unable to distinguish between unique -}
{-                       and non-unique solutions.                      -}
{-                       If you are concerned that  your equations      -}
{-                       may be underdetermined, solve them with        -}
{-                       Gaussian elimination (GAUSELIM.INC or          -}
{-                       PARTPIVT.INC                                   -}
{-                                                                      -}
{-          Version Date: 26 January 1987                               -}
{-                                                                      -}
{------------------------------------------------------------------------}
```

```
const
  TNNearlyZero = 1E-015;     { If you get a syntax error here, you are    }
                             { not running TURBO-87.                      }
                             { TNNearlyZero = 1E-015 if using the 8087    }
                             {                       math co-processor.   }
                             { TNNearlyZero = 1E-07 if not using the 8087 }
                             {                       math co-processor.   }

var
  Guess : TNvector;

procedure TestInput(Dimen        : integer;
                    Tol          : real;
                    MaxIter      : integer;
                var Coefficients : TNmatrix;
                var Constants    : TNvector;
                var Solution     : TNvector;
                var Error        : byte);

{---------------------------------}
{- Input: Dimen, Tol, MaxIter     -}
{-        Coefficients,           -}
{-        Constants               -}
{- Output: Solution, Error        -}
{-                                -}
{- test the input data for errors -}
{- The procedure also finds the   -}
{- solution for the trivial case  -}
{- Dimen = 0.                     -}
{---------------------------------}

begin
  Error := 0;
  if Dimen < 1 then
    Error := 3
  else
    if Tol <= 0 then
      Error := 4
    else
      if MaxIter < 0 then
        Error := 5;
  if (Error = 0) and (Dimen = 1) then
  begin
    if ABS(Coefficients[1, 1]) < TNNearlyZero then
      Error := 6
    else
      Solution[1] := Constants[1] / Coefficients[1, 1];
  end;
end; { procedure TestInput }

procedure TestForDiagDominance(Dimen        : integer;
                           var Coefficients : TNmatrix;
                           var Error        : byte);

{-----------------------------------------------------------------}
{- Input: Dimen, Coefficients                                     -}
{- Output: Error                                                  -}
{-                                                                -}
{- This procedure examines the Coefficients matrix to see if it is -}
{- diagonally dominant. If it is, then the Gauss-Seidel iterative  -}
{- method will converge to a solution of this system of equations; -}
{- if not, then convergence may not be possible with this method   -}
{- and Error = 1 (which is a warning) is returned. If one of the   -}
{- elements on the main diagonal of the Coefficients matrix is     -}
{- zero, then the matrix is singular and cannot be solved and      -}
```

```
{- Error = 6 is returned. In such a case, one of the direct        -}
{- methods for solving systems of equations (e.g. Gaussian         -}
{- elimination) should be used.                                    -}
{------------------------------------------------------------------}

var
  Row, Column : integer;
  Sum : real;

begin
  Row := 0;
  while (Row < Dimen) and (Error < 2) do
  begin
    Row := Succ(Row);
    Sum := 0;
    for Column := 1 to Dimen do
      if Column <> Row then
        Sum := Sum + ABS(Coefficients[Row, Column]);
    if Sum > ABS(Coefficients[Row, Row]) then
      Error := 1;  { WARNING! convergence may not be }
                   { possible because matrix isn't   }
                   { diagonally dominant             }
    if ABS(Coefficients[Row, Row]) < TNNearlyZero then
      Error := 6;  { Singular matrix - can't be solved  }
                   { by the Gauss-Seidel method.        }
  end; { while }
end; { procedure TestForDiagDominance }

procedure MakeInitialGuess(Dimen        : integer;
                       var Coefficients : TNmatrix;
                       var Constants    : TNvector;
                       var Guess        : TNvector);

{------------------------------------------------------------------}
{- Input: Dimen, Coefficients, Constants                           -}
{- Output: Guess                                                   -}
{-                                                                 -}
{- This procedure creates an initial approximation to the solution -}
{- by dividing the Constants terms by the corresponding terms      -}
{- on the main diagonal of the Coefficients matrix.                -}
{------------------------------------------------------------------}

var
  Term : integer;

begin
  FillChar(Guess, SizeOf(Guess), 0);
  for Term := 1 to Dimen do
    if ABS(Coefficients[Term, Term]) > TNNearlyZero then
      Guess[Term] := Constants[Term] / Coefficients[Term, Term];
end; { procedure MakeInitialGuess }

procedure TestForConvergence(Dimen     : integer;
                         var OldApprox : TNvector;
                         var NewApprox : TNvector;
                             Tol       : real;
                         var Done      : boolean;
                         var Product   : real;
                         var Error     : byte);

{------------------------------------------------------------------}
{- Input: Dimen, OldApprox, NewApprox, Tol, Product                 -}
{- Output: Done, Product, Error                                    -}
{-                                                                 -}
{- This procedure determines if the sequence of approximations     -}
```

```
{- has converged. For convergence to occur, the relative difference   -}
{- between each Term of OldApprox and NewApprox must be less than     -}
{- the tolerance, Tol. If so, Done = TRUE is returned.                -}
{-                                                                    -}
{- This procedure also determines if the sequence of approximations   -}
{- is diverging. Product records the total fractional change from     -}
{- the initial guess to the current iteration.  If Product is greater -}
{- than 1E20, then the sequence is assumed to have diverged. If so,   -}
{- Error = 7 is returned.                                             -}
{---------------------------------------------------------------------}

var
  Term : integer;
  PartProd : real;

begin
  Done := true;
  PartProd := 0;
  for Term := 1 to Dimen do
  begin
    if ABS(OldApprox[Term] - NewApprox[Term]) > ABS(NewApprox[Term] * Tol) then
      Done := false;
    if (ABS(OldApprox[Term]) > TNNearlyZero) and (Error = 1) then
      { This is part of the divergence test }
      PartProd := PartProd + ABS(NewApprox[Term] / OldApprox[Term]);
  end;
  Product := Product * PartProd / Dimen;
  if Product > 1E20 then
    Error := 7   { Sequence is diverging }
end; { procedure TestForConvergence }

procedure Iterate(Dimen        : integer;
              var Coefficients : TNmatrix;
              var Constants    : TNvector;
              var Guess        : TNvector;
                  Tol          : real;
                  MaxIter      : integer;
              var Solution     : TNvector;
              var Iter         : integer;
              var Error        : byte);

{-----------------------------------------------------------}
{- Input: Dimen, Coefficients, Constants, Guess, Tol, MaxIter -}
{- Output: Solution, Iter, Error                             -}
{-                                                           -}
{- This procedure performs the Gauss-Seidel iteration and    -}
{- returns either an error or the approximated solution and  -}
{- the number of iterations.                                 -}
{-----------------------------------------------------------}

var
  Done : boolean;
  OldApprox, NewApprox : TNvector;
  Term, Loop : integer;
  FirstSum, SecondSum, Product : real;

begin { procedure Iterate }
  Product := 1;
  Done := false;
  Iter := 0;
  NewApprox := Guess;
  OldApprox := Guess;
  while (Iter < MaxIter) and not(Done) and (Error <= 1) do
  begin
    Iter := Succ(Iter);
```

```
    for Term := 1 to Dimen do
    begin
      FirstSum := 0;
      SecondSum := 0;
      for Loop := 1 to Term - 1 do
        FirstSum := FirstSum + Coefficients[Term, Loop] * NewApprox[Loop];
        for Loop := Term + 1 to Dimen do
          SecondSum := SecondSum + Coefficients[Term, Loop] * OldApprox[Loop];
      NewApprox[Term] := (Constants[Term] - FirstSum - SecondSum) /
                         Coefficients[Term, Term];
    end;
    TestForConvergence(Dimen, OldApprox, NewApprox, Tol, Done, Product, Error);
    OldApprox := NewApprox;
  end; { while }
  if (Iter < MaxIter) and (Error = 1) then
    Error := 0;  { The sequence converged, }
                 {  disregard the warning  }
  if (Iter >= MaxIter) and (Error = 1) then
    Error := 1;  { Matrix is not diagonally dominant; }
                 { convergence is probably impossible }
  if (Iter >= MaxIter) and (Error = 0) then
     Error := 2; { Convergence IS possible;   }
                 { more iterations are needed }
  Solution := NewApprox;
end; { procedure Iterate }

begin  { procedure Gauss_Seidel }
  TestInput(Dimen, Tol, MaxIter, Coefficients, Constants, Solution, Error);
  if Dimen > 1 then
  begin
    TestForDiagDominance(Dimen, Coefficients, Error);
    if Error < 2 then
    begin
      MakeInitialGuess(Dimen, Coefficients, Constants, Guess);
      Iterate(Dimen, Coefficients, Constants, Guess, Tol,
              MaxIter, Solution, Iter, Error);
    end;
  end;
end; { procedure Gauss_Seidel }
```

**BIBLIOGRAPHIE**


V.L.Streeter andE.B.Wylie, "FLUID MECHANICS",

MacGraw-Hill KOGAKUSHA,LTD ,1975

S.J. Michel,"FLUIDS AND PARTICLE MECHANICS"

1970 PERGAMONON PRESS LTD,Library of congress

A.FOX,"ENGINEERING FLUID MECHANICS"

MacMillan press LTD.

R.L. DAUGHERTIN,J.B FRANZINI,"FLUIDS MECHANICS WITH ENGINEERINGS APPLICATIONS"

MacGraw-Hill KOGAKUSHA LTD

B.CARNAHAN,H.A.LUTHER,J.O.WILKES,"APPLIED NUMERICAL METHODS"

John Wiley & sons,Inc

SHAN S.KUO,"COMPUTER APPLICATIONS OF NUMERICAL METHODS"

Addison,Wesley Publishing Company

Borland Software,"TURBO PASCAL REFERENCE MANUAL"

K.JAMSA & S.NAMEROFF,"TURBO PASCAL Programmer's LIBRARY"

Borland-Osborne/McGraw-Hill

JEFF DUNTERMAN,"COMPLETE TURBO PASCAL"

Scott,Foresman and Company

DON J. WOOD,"COMPUTER ANALYSIS OF FLOW IN PIPE NETWORKS" (USER'S MANUAL")

University of Kentucky,DPT of Civil Engineering

HERBERT SCHILDT,"ADVANCED TURBO PASCAL"

Borland-Osborne/mcGraw-Hill

AMADOU SARR, "COURS DE MECANIQUE DES FLUIDES II"

Ecole Polytechnique de THIES,1988

J.C BERNARD,R.GUARDO,J.LAVOIE,P.SAVARD "INFORMATIQUE I"

,Notes de cours

Ecole Polytechnique de Montreal,Spt 1988

K.JENSEN,N.WIRTH, "PASCAL ,Manuel de l'utilisateur et

Rapport de Définition

EYROLLES 1987