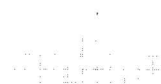


TECHNICAL REPORT 84-49



FEM 301 – A Three Dimensional Model for Groundwater Flow Simulation

L. Kiraly

January 1985

Centre d'Hydrogéologie, Université de Neuchâtel

The computer program presented in appendix A3 is by now out of date, but the description of the frontal method is always valid.

Nagra

Nationale
Genossenschaft
für die Lagerung
radioaktiver Abfälle

Cédra

Société coopérative
nationale
pour l'entreposage
de déchets radioactifs

Cisra

Società cooperativa
nazionale
per l'immagazzinamento
di scorie radioattive

Der vorliegende Bericht wurde im Auftrag der Nagra erstellt. Die Autoren haben ihre eigenen Ansichten und Schlussfolgerungen dargestellt. Diese müssen nicht unbedingt mit denjenigen der Nagra übereinstimmen.

Le présent rapport a été préparé sur demande de la Cédra. Les opinions et conclusions présentées sont celles des auteurs et ne correspondent pas nécessairement à celles de la Cédra.

This report was prepared as an account of work sponsored by Nagra. The viewpoints presented and conclusions reached are those of the author(s) and do not necessarily represent those of Nagra.

ABSTRACT

The purpose of this report is to describe program FEM 301. Program FEM 301 is a three-dimensional, finite element numerical model which solves the steady-state groundwater flow equation. Following a brief description of the mathematical model upon which the numerical model is based, the finite element approximation to the flow equation is presented. The solution method employed by FEM 301 is the frontal elimination technique of IRONS (1970).

The general structure of program FEM 301 is described as are the required input files and the generated output file. The listing of FEM 301 is included as Annex A3. As numerous pre- and post-processing routines have been developed specifically for use with FEM 301 they are also presented.

Several sample problems and verification tests depict the application of FEM 301 to typical hydrogeologic problems. The principal application of FEM 301 has been, however, to evaluate the deep crystalline groundwater flow regime in central northern Switzerland. This application is described in a companion report NTB 84-50 (Kimmeier et al., 1985).

The responsibility for the hydrogeological investigation program lies with the Geological Division of Nagra, headed by Dr. M. Thury. The project leader is Dr. W. Kanz; Scientific advisors: Dr. R. Andrews and Dr. P. Zuidema.

RESUME

Le présent rapport contient la description du programme de calcul FEM 301. Il s'agit d'un modèle numérique à éléments finis, qui permet de simuler les écoulements permanents tridimensionnels dans la zone saturée. Après une brève présentation des équations différentielles fondamentales, on présente la solution approchée obtenue par la méthode des éléments finis. FEM 301 utilise la technique de l'élimination frontale développée par B.M. IRONS (1970).

La description de la structure générale de FEM 301 et celle des fichiers d'entrée et de sortie sont accompagnées du listing complet du programme (voir ANNEXE A3). Certains programmes auxiliaires, spécialement développés pour FEM 301, sont également présentés.

Quelques exemples, utilisés pour la vérification du programme, illustrent l'application de FEM 301 à des problèmes hydrogéologiques caractéristiques. Toutefois, la principale application de FEM 301 a consisté dans la simulation des écoulements souterrains profonds dans le socle cristallin de la Suisse septentrionale. Les résultats détaillés de ces travaux sont présentés dans le rapport NTB 84-50 (Kimmeier et al., 1985).

La responsabilité du programme de recherches hydrogéologiques incombe à la section "Géologie" de la Cédra, dirigée par M. Thury. Le chef de projet est W. Kanz. Conseillers techniques: R. Andrews et P. Zuidema.

ZUSAMMENFASSUNG

Das in diesem Bericht beschriebene Rechenprogramm FEM 301 ist ein dreidimensionales numerisches Modell, beruhend auf der Methode der finiten Elemente und dient der Lösung stationärer Grundwasserfliessprobleme. Anschliessend an eine kurze Beschreibung der mathematischen Grundlagen wird die Näherungslösung mit Hilfe der finiten Element-Methode dargestellt. Das Programm FEM 301 verwendet die "frontal elimination"-Technik nach IRONS (1970).

Neben der Programmstruktur werden auch die entsprechenden Ein- und Ausgabe-Datensätze beschrieben (das gesamte Rechenprogramm findet sich in Anhang A3). Ebenfalls aufgeführt sind die zahlreichen verschiedenen, speziell für den Gebrauch von FEM 301 entwickelten Pre- und Postprocessing-Verfahren.

Etliche Problembeispiele zeigen die Anwendung von FEM 301 auf typische hydrogeologische Fragestellungen. Hauptsächlich wurde das Programm FEM 301 jedoch dafür eingesetzt, die Grundwasserfliessverhältnisse im tieferen Kristallin der zentralen Nordschweiz wiederzugeben. Diese Anwendung wird im gleichzeitig erscheinenden NTB 84-50 (Kimmeier et al., 1985) behandelt.

Das hydrogeologische Untersuchungsprogramm der Nagra ist dem Bereich Geologie, Bereichsleiter Dr. M. Thury, zugeordnet. Projektleiter ist Dr. W. Kanz, der auch die Redaktion des vorliegenden Berichtes besorgte. Fachliche Berater waren Dr. R. Andrews und Dr. P. Zuidema.

TABLE OF CONTENTS

	<u>Page</u>
ABSTRACT	I
RESUME	II
ZUSAMMENFASSUNG	III
TABLE OF CONTENTS	IV
LIST OF FIGURES	VI
LIST OF TABLES	VIII
1. INTRODUCTION	1
1.1 Program Purpose	1
1.2 Program Overview	1
1.3 Program Assumptions and Limitations	4
1.4 Applications of the Program	5
1.5 Structure of this Report	5
1.6 Acknowledgements	6
2. FLOW EQUATIONS USED IN THE MODEL	7
2.1 Darcy's Law	7
2.2 Continuity Equation	10
3. FINITE ELEMENT FORMULATION OF THE FLOW EQUATION	13
3.1 Element Types used in the Model	13
3.2 Galerkin Formulation of the Flow Equation for Finite Elements	19
3.3 Calculation of the Gradient Matrix	21
3.4 Numerical Integration over the Curvilinear Elements	25
4. SOLUTION METHOD	28
4.1 The Gaussian Elimination Method	28
4.2 The Frontal Elimination Method	32
5. CODE DESCRIPTION OF FEM 301	42
5.1 Input Files	42
5.2 Output File	49
5.3 The Flow Chart of FEM 301	53

TABLE OF CONTENTS (continued)

	<u>Page</u>
6. PRE- AND POST-PROCESSING ROUTINES	57
6.1 Pre-Processing Routines	57
6.1.1 ELEBIDI	59
6.1.2 TABLETTE	59
6.1.3 FERN	59
6.1.4 ESTIMZ	60
6.1.5 ELETRIDI	60
6.1.6 OPERZ	61
6.1.7 INSEREZ	61
6.1.8 CALCULZ	61
6.1.9 UJZ	61
6.1.10 NEWFRONT	62
6.1.11 PRECIPIT	62
6.2 Post-Processing Routines	63
6.2.1 GRAD	63
6.2.2 FEDP320	64
6.2.3 DEBSEG	67
6.2.4 QSURF2	67
6.2.5 DEBALI	67
7. SAMPLE PROBLEMS AND VERIFICATION TESTS	70
7.1 Example Problem 1	70
7.2 Verification of FEM 301	74
7.2.1 Well in a Two-Dimensioned Bounded Aquifer	74
7.2.2 Steady Flow to a Well in a Leaky, Confined Aquifer	81
7.3 Remarks on the behaviour of the elements	85
7.3.1 Pinched-sided elements	85
7.3.2 Too large elements	88
8. REFERENCES	95
Annex A1 Nomenclature	A1-1
Annex A2 Principal variables in Program FEM 301	A2-1
Annex A3 Program FEM 301 listing	A3-1
Annex A4 Program LINBC listing	A4-1

LIST OF FIGURES

	<u>Page</u>
Figure 3-1 Typical Finite Element Network with 8-node and 6-node Elements	14
Figure 3-2 Global and Local Coordinates of 1-, 2-, and 3-Dimensional Elements	16
Figure 3-3 Representation of Gradients in Curvilinear Coordinates	22
Figure 4-1 Schematic Representation of the Gauss Elimination Method	29
Figure 4-2 Partitioning of Matrixes in the Gauss Elimination Method	29
Figure 4-3 Simple 2-Dimensional Element Network	31
Figure 4-4a Assembled Coefficient Matrix for the Simple Network	31
Figure 4-4b Position of the Coefficient Matrix Required during Gaussian Elimination	31
Figure 4-5 Symbolic Elimination by the Frontal Method for a Simple Network	37
Figure 4-6 Matrix CGR for each Element in Figure 4-3	38
Figure 4-7 Rediscretization of the Simple Network	39
Figure 5-1 Structure of the Element Input File	44
Figure 5-2 Ordering of Node Numbers	45
Figure 5-3 Structure of the Coordinate Input File	47
Figure 5-4 Structure of the Parameter Input File	48
Figure 5-5 Structure of the Output File	50
Figure 5-6 Flow Chart for Program FEM 301	52
Figure 6-1 Pre-Processing Routines used to create Element and Coordinate Files	58
Figure 6-2 Example Output from Program GRAD	65
Figure 6-3 Viewing Orientations for Two-Dimensional Plots using FEDP320	66
Figure 6-4 Example Output from DEBALI	69
Figure 7-1 Three-Dimensional fractured Block Model with 1-D, 2-D, and 3-D Elements	71

LIST OF FIGURES (continued)

	<u>Page</u>
Figure 7-2/3 Calculated Potentials of the fractured Block Model	72
Figure 7-4 Aquifer Geometry for Two-Dimensional, Steady State Flow to a Well	76
Figure 7-5 Finite Element Meshes for Two-Dimensional Verification Test	78
a) Mesh A	
b) Mesh B	
c) Mesh C	
Figure 7-6 Simulated Equipotentials for Steady State Flow to a Well in a bounded Aquifer	79
Figure 7-7 Vertical Cross-Section of Flow to a Well in a leaky, confined Aquifer	82
Figure 7-8 Results of the three-dimensional bar example with varying element shapes	86
a) Pyramide-Tetrahedron assembly	
b) "Pinched-sided" tetrahedron assembly	
c) curved element sides	
Figure 7-9 Structure of the blocks used in Figures 7-8a, b, c	87
Figure 7-10 Boundary conditions for large element example	89
Figure 7-11 Results of large element example	
a) "Exact" solution	90
b) Single element solution	91
Figure 7-12 Result file for single element solution	92

LIST OF TABLES

	<u>Page</u>
Table 7-1 Comparison of analytical solution and FEM 301 for flow in a bounded aquifer	80
Table 7-2 Comparison of analytical and numerical results for a well in a leaky aquifer, Run PUITSV03	83
Table 7-3 Comparison of analytical and numerical results for a well in a leaky aquifer, Run PUITSV05	84

1. INTRODUCTION

1.1 Program Purpose

FEM 301 simulates steady state, two or three dimensional groundwater flow by the finite element method. The program solves the steady state groundwater flow equation given appropriate boundary conditions. The program allows for the incorporation of one- or two-dimensional elements within a three-dimensional network.

FEM 301 may be applied to any steady state groundwater flow problem. The program's ability to handle discrete discontinuities makes it particularly well suited for analyzing groundwater flow in highly heterogeneous geologic media (e.g. crystalline rocks or karstic limestones). The solution method employed in the program allows for the incorporation of over ten thousand nodal points which makes the modeling of very large and complex regional groundwater systems possible.

The principal reason for NAGRA support for the development of FEM 301 and its accessory programs has been the aim to apply this program to the analysis of the deep crystalline groundwater flow regime in northern Switzerland. This application is described in NAGRA NTB 84-50 "Simulation des Ecoulements Souterrains entre les Alpes et la Forêt-Noire par Modèle Mathématique" (Kimmeier et al., 1985).

1.2 Program overview

The program FEM 301 solves the partial differential equation describing steady state groundwater flow in a porous or fractured media. The flow equation arises from a combination of Darcy's Law (expressing momentum conservation) and the continuity equation (expressing mass conservation). The flow equation is solved by the Galerkin finite element method.

Solution method

In the finite element method, the unknown function (head) is approximated over a subset of the entire global domain (referred to as elements) by using known type of functions (interpolation functions). In the present program only quadratic interpolation functions are employed.

The Galerkin weighted residual approach allows to transform the partial differential equation describing the flow into a set of integral equations which contain derivatives of the first order only. The integration is performed numerically over each element yielding a set of linear equations, which is solved by the frontal elimination technique of B. IRONS (1970).

Element shapes and dimensions

The entire model domain must be discretized into 1-, 2-, or 3-dimensional elements. The program allows for the use of either triangles or rectangles for 2-dimensional elements and either tetrahedron, triangular prisms, or cubes for 3-dimensional elements. These elements may undergo further deformation (e.g. reducing element sides or faces to a point by assigning the same coordinates to more than one nodal point) although care must be taken to ensure compatibility with neighbouring elements. As quadratic interpolation functions are utilized in the program, the user must specify three nodes along each element edge. In addition, nodes on faces and in the interior of elements must be used if Lagrangian elements are employed. The number of element shapes allows for considerable flexibility in designing a hydrogeologic model.

The ability to use varying dimensioned elements allows the modeling of discrete planar or unidimensional features without significantly increasing the number of three-dimensional elements required.

Input required

Input to program FEM 301 consists of three individual files. The element file defines the element topology (element shape, the number of nodes, and the node numbers) and the permeability class and infiltration class for each element of the modeled region. The coordinate file defines the spatial coordinates (x, y, z) of each node. The parameter file defines the permeability (m/s) of each permeability class (or permeability times width of 2-d elements or permeability times area of 1-d elements), the infiltration rate (m/s) for each infiltration class, the heads at constant head nodes, and the volumetric recharge/discharge (m^3/s) at nodes representing injection/withdrawal wells.

Output generated

The output generated from a simulation using FEM 301 consists of one file containing an echo of the parameter file and a table of calculated heads and volumetric recharge/discharge (only at constant head nodes or specified recharge/discharge nodes) at each node in the modeled domain. In addition, the sum of all debits from the model is presented as a check of the total water balance for the entire model.

A series of post-processing routines have been developed to portray the calculated results. These include the possibility of contouring the heads along two-dimensional (x-y, x-z, y-z) cross-sectional planes or in three-dimensional block diagrams. Only contouring along element faces is presently possible. Programs are also available to sum volumetric recharges over specified subdomains of the model and dividing by the area of influence to obtain infiltration rates. In addition volumetric discharges may be summed over varying river segments to obtain the simulated groundwater discharge component at river gauging stations.

Hydraulic gradients may be calculated anywhere in the three-dimensional element network using the post-processing routine GRAD.

Special features

FEM 301 contains several features to allow the modeler to rapidly introduce changes in the conceptual model. In addition the ability to use variably dimensioned elements and the ease with which the element topology can be modified (due to using the frontal solution method), allow the modeler considerable flexibility. The special features of FEM 301 are enumerated below:

1. The use of permeability classes, allows the assigned permeability over large areas in the model to be varied rapidly. Thus aquifer heterogeneity may be defined over groups of elements rather than element by element.
2. The use of infiltration classes, allows the assigned infiltration rates to be varied easily.
3. Anisotropic permeability fields may be modeled.
4. The implementation of the frontal solution method allows an efficient solution of very large 3-D problems.
5. Quadratic elements allows for the use of curvilinear element sides.
6. A resolution routine allows the boundary conditions (either specified head or infiltration) to be varied without requiring the reformation of the coefficient matrix. That is the right hand side vector is updated but the same coefficient matrix is used in the solution routine. This decreases computational time by about 90 % and allows for rapid testing of various boundary condition hypotheses. (Note: If the boundary condition type is changed a completely new solution is required).

1.3 Program assumptions and limitations

As with any numerical approximation of a mathematical model or mathematical representation of a physical phenomena, several approximations are made in the formulation of FEM 301. The user must be aware of these assumptions as they may limit his or her ability to apply the calculational tool to the real world.

The mathematical model, represented by the steady-state groundwater flow equation assumes:

1. Darcy's Law is applicable to defining the volumetric fluid flux in porous media at the macro-scopic scale. This implies that:
 - only pressure gradients and gravity are driving forces for water movement
 - the flow velocity is small enough that inertial terms may be neglected
 - the flow resistance is linearly proportional to the volume flux.
2. Spatial variations in density are small (i.e. the density is constant)
3. Temporal variations are negligible (i.e. steady-state)

The numerical model, represented by Galerkin finite element formulation assumes:

1. The hydraulic heads may vary only as quadratic functions within each element
2. Element sides cannot be distorted more than the degree of the polynomial approximation of the head.

Limitations related to the application of program FEM 301 include:

1. The code is restricted to saturated flow and assumes constant permeability within an element.
2. Too large elements could generate locally unrealistic results in regions where the real potential field cannot be described by a quadratic function.
3. Gradients are not calculated directly, but require the use of a post-processing routine.
4. There is no check that the surface integral of normal fluxes to each element equal zero.

1.4 Applications of the program

FEM 301 could be used for a wide range of steady-state groundwater flow problems. It is particularly suited for problems involving one-dimensional or two-dimensional discontinuities in an otherwise continuous (isotropic or anisotropic) porous media.

The principal application of FEM 301 has been to define the regional groundwater flow system in the crystalline bedrock of northern Switzerland. This work has been conducted for NAGRA with the purpose of determining the likely groundwater flow paths from possible HLW repository sites to the biosphere, the hydraulic gradients along these travel paths, and the variability in these predictions based on uncertain input parameters. These results are reported in Kimmeier and others (1985; NTB 84-50). Other issues addressed for NAGRA using this program include (NOTE: these have not been published) the determination of steady-state inflows to an underground repository, the evaluation of how permeable and close a 2-dimension feature must be before impacting the local flow directions in the vicinity of a repository, and the evaluation of possible explanations for the low heads observed in the crystalline at Schaffisheim.

1.5 Structure of this report

This report has been divided into 6 chapters, in addition to the Introduction. Chapter 2 describes the generation of the general groundwater flow equation from Darcy's Law and the continuity equation. The flow equation to be solved effectively defines the mathematical model.

Chapter 3 formulates the Galerkin finite element approximation of the groundwater flow equation. The element types available for use in FEM 301 are described. The method of calculating the gradient matrix and the method of numerically integrating the volume, surface, or length integrals are also presented.

Chapter 4 defines the method used to solve the linear system of global equations. Prior to presenting the frontal solution method, the general Gaussian elimination procedure is described. A simple two-dimensional element network is used to illustrate the differences in the two methods and the advantages of the frontal solution method.

Chapter 5 describes the structure of FEM 301 including a simplified flow diagram. In addition, descriptions of the input and output files (including examples of each) are also presented.

Chapter 6 defines the pre- and post-processing routines, respectively, developed specifically for use with FEM 301. These routines assist the user in developing the requisite input files for the program and interpreting the output from the program.

Chapter 7 completely describes a simple example problem to assist the reader in understanding the use of the code. In addition, two verification problems are presented. The verification tests consist of steady flow to a well in a bounded two-dimensional aquifer and steady flow to a well across a leaky aquitard.

Five annexes have been included for the interested reader. Annex A1 lists the nomenclature used in the text. Annex A2 defines the principal variables used in the listing of FEM 301 and Annex A3 contains the listing itself. Annex A4 presents the analytical solution routine (LINBC) used in the two-dimensional verification test.

1.6 Acknowledgements

The author would like to express his most sincere thanks to all those who helped him to complete this report. Special thanks are due to Dr. Robert Andrews (INTERA Inc., Houston, USA) who not only corrected the english text, but actively participated in writing the Introduction, as well as Chapter 6 and part of Chapter 7. In particular, he carried out the verification tests described in section 7.2.

Many of the pre- and post-processing programs presented in Chapter 6 have been developed by F. Piccini, scientific collaborator at the Centre d'Hydrogéologie of Neuchâtel. Without these contributions it would be nearly impossible to generate the data files and analyse the results for very large 3-D problems.

2. THE FLOW EQUATIONS USED FOR THE MODEL

In modelling groundwater flow on a large regional scale, without thermal effects or solute transport, the fundamental equations of interest arise from two sources:

1. The Darcy flow laws for porous media, which effectively express momentum conservation.
2. The continuity equation which expresses mass conservation.

This chapter will not be, however, devoted to the detailed derivation of Darcy's law and the continuity equation, as more rigorous developments can be found in HUBBERT (1940, 1957), De WIEST (1965), BEAR (1972, 1979), DAGAN (1972), and De MARSILLY (1981). Our aim is to present the general form of these equations, insisting rather on the simplifying assumptions which allow us to obtain the classical "diffusivity equation" used for many practical hydrogeological problems.

2.1 Darcy's law

Darcy's law is an expression describing the "conductive" volume flux in porous media at macroscopic scale. It can be derived from the Navier-Stokes equations expressing momentum conservation (HUBBERT, 1957; DAGAN, 1972; De MARSILLY, 1981) if we make the following hypotheses:

- the only driving forces acting on the fluid are the pressure gradient and the gravity
 - inertial terms can be neglected (the flow velocity is sufficiently small)
 - resistance to flow is dominated by viscous shear and is proportional to the volume flux or "Darcy velocity"
- $$\vec{q} \left[\text{m}^3/\text{s} \cdot \text{m}^2 \right].$$

Under these assumptions we obtain Darcy's law in the form of equation 2-1:

$$\vec{q} = - \frac{\bar{k}}{\mu} (\vec{\text{grad}} p + \rho g \vec{\text{grad}} z) = - \frac{\bar{k}}{\mu} (\vec{\text{grad}} p - \rho \vec{g}) \quad 2-1$$

where

$\vec{q}$ = volume flux ($\text{m}^3/\text{s} \cdot \text{m}^2$)

$\bar{k}$ = intrinsic or geometric permeability (m^2)

μ = dynamic viscosity ($\text{kg}/\text{m} \cdot \text{s}$)

p = pressure ($\text{kg}/\text{m} \cdot \text{s}^2$)

ρ = density of water (kg/m^3)

g = acceleration due to gravity (m/s^2)

z = elevation above a datum level (m)

The volume flux $\vec{q}$ [$\text{m}^3/\text{s} \cdot \text{m}^2$] is the volume of water flowing per unit time through a unit cross-sectional area of the porous medium normal to the direction of flow. As only a few % of this cross-sectional area allows the water to flow (the "interconnected pores"), the volume flux $\vec{q}$ is **not** the mean groundwater velocity, in spite of such misleading names as "Darcy velocity" or "seepage velocity".

The vectors $\vec{\text{grad}} p$ and $\rho \vec{g}$ are forces per unit volume of water $\left[\frac{\text{kg} \cdot \text{m}}{\text{s}^2} \cdot \frac{1}{\text{m}^3} \right]$, and their resultant $\vec{F} = \vec{\text{grad}} p - \rho \vec{g}$ represents the general driving force inducing the groundwater flow. If the force field $\vec{F}$ is irrotational, it can be derived from a force potential, or hydraulic potential h . The condition for the force field $\vec{F}$ to be irrotational is:

$$\begin{aligned} \text{rot } \vec{F} &= \text{rot} (\vec{\text{grad}} p - \rho \vec{g}) = 0 \quad \text{or} \\ \text{rot } \vec{F} &= \text{rot} \cdot \vec{\text{grad}} p - \vec{\text{grad}} \rho \times \vec{g} - \rho \cdot \text{rot } \vec{g} = 0 \end{aligned}$$

As $\text{rot } \vec{\text{grad}} p = 0$ and $\text{rot } \vec{g} = 0$, we have the condition for the existence of a hydraulic potential:

$$\text{rot } \vec{F} = \vec{\text{grad}} \rho \times \vec{g} = 0 \quad 2-2$$

Equation 2-2 is satisfied only in two cases:

1. In hydrostatic conditions, if there is a horizontal stratification of the densities ($\vec{\text{grad}} \rho$ is parallel to $\vec{g}$).
2. In aquifers where $\vec{\text{grad}} \rho = 0$ (the density of water is constant) or, at least, $\vec{\text{grad}} \rho$ can be neglected (the space variations of the density are small).

We have to mention that HUBBERT (1940, 1957) proposed to express the **g**eneral driving force by

$$\vec{E} = \frac{1}{\rho} \vec{\text{grad}} p - \vec{g} \left[\frac{\text{kg} \cdot \text{m}}{\text{s}^2} \cdot \frac{1}{\text{kg}} \right]$$

which is a force per unit mass of water. An important advantage of this expression is that the force field $\vec{E}$ remains irrotational (and can be derived from a force potential) even for compressible fluids, if the density ρ of the fluid depends on the pressure p only. The condition for $\vec{E}$ to be irrotational is

$$\text{rot } \vec{E} = \vec{\text{grad}} \left(\frac{1}{\rho} \right) \times \vec{\text{grad}} p + \frac{1}{\rho} \text{rot} \cdot \vec{\text{grad}} p - \text{rot } \vec{g} = 0$$

As $\text{rot } \vec{g} = 0$ and $\text{rot} \cdot \vec{\text{grad}} p = 0$, we have:

$$\text{rot } \vec{E} = \vec{\text{grad}}\left(\frac{1}{\rho}\right) \times \vec{\text{grad}} p = 0 \quad 2-2a$$

and equation 2-2a is satisfied in the following cases:

1. $\vec{\text{grad}} p = 0$ (= constant pressure field).
2. $\vec{\text{grad}} (1/\rho) = 0$, i.e. the space variations of the density are zero or negligible ($\rho = \text{constant}$).
3. $\vec{\text{grad}} p$ is parallel to $\vec{\text{grad}} (1/\rho)$, i.e. the surfaces of equal pressure are parallel to the surfaces of equal density. This occurs when the density $\rho = \rho(p)$ depends on the pressure p only, and in this case the driving force $\vec{E}$ may be expressed as the gradient of HUBBERT's force potential:

$$\vec{E} = \vec{\text{grad}} \left[\int_0^p \frac{dp}{\rho(p)} + gz \right] = \vec{\text{grad}} h$$

Obviously, $h \left[\frac{\text{kg} \cdot \text{m}^2}{\text{s}^2} \cdot \frac{1}{\text{kg}} \right]$ is an energy density per unit mass of water.

If the groundwater flow is not coupled with heat and/or solute transport (that is, if the density is not related to temperature effects and/or solute concentration), there will be no use to introduce variable water densities into the flow model. Thus we admit an important simplifying hypothesis for the regional and local models: we suppose that in the modelled volume the **spatial variations of the density are small** and we can neglect $\vec{\text{grad}} \rho$.

By hypothesis:

$$\rho \approx \text{const.} \quad 2-3$$

By putting $\rho \approx \text{const.}$ we allow the force field $\vec{F}$ or $\vec{E}$ and the flow field $\vec{q}$ to be irrotational and we can define the hydraulic potential, as well as Darcy's law, in different ways:

$$\vec{q} = \frac{\bar{k}}{\mu} \cdot \vec{\text{grad}} (p + \rho gz) = \bar{K}_1 \cdot \vec{\text{grad}} h_1 \quad 2-4$$

$$\vec{q} = \frac{\rho \bar{k}}{\mu} \cdot \vec{\text{grad}} \left(\frac{p}{\rho} + gz \right) = \bar{K}_2 \cdot \vec{\text{grad}} h_2 \quad 2-5$$

$$\vec{q} = \frac{\rho g \bar{k}}{\mu} \cdot \vec{\text{grad}} \left(\frac{p}{\rho g} + z \right) = \bar{K}_3 \cdot \vec{\text{grad}} h_3 \quad 2-6$$

The **potentials** h_1 [kg/m · s²], h_2 [m²/s²], and h_3 [m] are energy densities per unit volume of water, per unit mass of water, and per unit weight of water. The **gradients** represent the driving forces:

$\text{grad } h_1$ [kg/m² · s²], $\text{grad } h_2$ [m/s²], and $\text{grad } h_3$ [1] are forces per unit volume of water, per unit mass of water, and per unit weight of water. As $\vec{q}$ must be in each case a volume flux, the definition of the **volume conductivities** or **hydraulic conductivities** $\bar{K}$, will depend on the chosen potential h_i :

$$\bar{K}_1 = \frac{\bar{k}}{\mu} \left[\text{m}^3 \text{s/kg} \right] \quad \text{for } h_1$$

$$\bar{K}_2 = \frac{\rho \bar{k}}{\mu} = \frac{\bar{k}}{\nu} \left[\text{s} \right] \quad \text{for } h_2$$

$$\bar{K}_3 = \frac{\rho g \bar{k}}{\mu} = \frac{g}{\nu} \bar{k} \left[\text{m/s} \right] \quad \text{for } h_3$$

where $\nu = \mu/\rho = \text{kinematic viscosity (m}^2/\text{s)}$.

In our models we will use throughout equation 2-6, more familiar to hydrogeologists, where the potential ("head") is defined in [m] and the volume conductivity ("permeability" of the hydrogeologists) is given in [m/s] .

2.2 The continuity equation

For our groundwater flow problem (saturated porous medium) we express the mass balance by the following equation (BEAR, 1979):

$$\frac{\partial(\epsilon \cdot \rho)}{\partial t} + \text{div } (\rho \vec{q}) + \rho Q = 0 \quad 2-7$$

where:

ρ = density of water [kg/m³]

ϵ = porosity of the porous medium [1]

q = volume flux (m³/s · m²) as expressed by equation 2-6.

Q = volume source/sink term [m³/s m³]

In order to obtain the classical "diffusivity equation" used for many practical problems in hydrogeology, we have to introduce some simplifications in the first and second terms of the mass balance equation.

The **first term** is the change of mass of the groundwater per unit time and per unit volume of porous medium. If this term is not zero, we have to admit that density ρ and ϵ can change, even if slightly, with time. Although the compressibility of the water and the compressibility of the solid skeleton of the aquifer are small, changes in water pressure p will produce small changes in density ρ and in porosity ϵ (BEAR, 1979; pp. 85.86), thus changing the mass of water stored in a unit volume of the porous medium.

Let $\epsilon = \epsilon(p)$ and $\rho = \rho(p)$. The first term can be expressed as:

$$\frac{\partial(\epsilon \cdot \rho)}{\partial t} = \left[\epsilon \frac{\partial \rho}{\partial p} + \rho \frac{\partial \epsilon}{\partial p} \right] \frac{\partial p}{\partial t} = S_{(p)}^m \cdot \frac{\partial p}{\partial t} \quad 2-8$$

where $S_{(p)}^m \left[s^2/m^2 \right]$ can be considered as a "mass storativity" related to pressure change: it is the mass of water released from storage (or added to) in a unit volume of aquifer per unit decline (or rise) in pressure.

Let us define the compressibility of the water $\beta \left[ms^2/kg \right]$ and the vertical compressibility of the solid matrix $\alpha \left[ms^2/kg \right]$ by

$$\beta = \frac{1}{\rho} \frac{\partial \rho}{\partial p} \quad \text{and} \quad \alpha = \frac{1}{(1-\epsilon)} \cdot \frac{\partial \epsilon}{\partial p}$$

Then $\frac{\partial \rho}{\partial p} = \rho \beta$ and $\frac{\partial \epsilon}{\partial p} = (1-\epsilon) \alpha$, and we can express the mass storativity as:

$$S_{(p)}^m = \rho \left[(1-\epsilon) \alpha + \epsilon \beta \right] \quad 2-9$$

However, as we want to use the potential $h = \frac{p}{\rho g} + Z$ instead of the pressure p , we will express $\partial p / \partial t$ in 2-8 by the time derivative of h :

$$p = \rho g(h-z) \Rightarrow \frac{\partial p}{\partial t} = \rho g \frac{\partial h}{\partial t} + (h-z)g \frac{\partial \rho}{\partial p} \cdot \frac{\partial p}{\partial t} \quad \text{or}$$

$$\rho g \frac{\partial h}{\partial t} = \frac{\partial p}{\partial t} \left[1 - \underbrace{(h-z)g \frac{\partial \rho}{\partial p}}_{\text{small as compared to 1}} \right]$$

The second term in the brackets is small as compared to 1, even if $(h-z)$ is of the order of 10'000 m! Let us take the values:

$$\rho = 10^3 \left[kg/m^3 \right], \quad g = 10 \left[m/s^2 \right], \quad (h-z) = 10^4 \left[m \right], \\ \text{and } \beta = 5 \cdot 10^{-10} \left[ms^2/kg \right]$$

Then:

$$g(h-z) \cdot \frac{\partial \rho}{\partial p} = g \rho \beta (h-z) = 10 \cdot 10^3 \cdot 5 \cdot 10^{-10} \cdot 10^4 = \\ = 5 \cdot 10^{-2}$$

So we have, with a good approximation:

$$\frac{\partial p}{\partial t} \approx \rho g \frac{\partial h}{\partial t} \quad 2-10$$

Substituting 2-10 in equation 2-8, we obtain for the first term of the mass balance equation:

$$\frac{\partial(\varepsilon \rho)}{\partial t} \approx \rho g S_{(p)}^m \frac{\partial h}{\partial t} = \rho S \frac{\partial h}{\partial t} \quad 2-11$$

where S [1/m] is the **volume storativity** or "specific storativity" related to hydraulic head changes: it is the volume of water released from storage in a unit volume of aquifer per unit decline in hydraulic head. Making use of 2-9, the volume storativity can be expressed by

$$S = g S_{(p)}^m = \rho g [(1-\varepsilon) \alpha + \varepsilon \beta] \quad 2-12$$

The **second term** of the mass balance equation is the divergence of the convective mass flux. Developing this term we have

$$\text{div}(\rho \vec{q}) = \rho \cdot \text{div} \vec{q} + \vec{\text{grad}} \rho \cdot \vec{q}$$

As $\rho = \rho(p)$, the term $\vec{\text{grad}} \rho$ is not quite equal to zero. However, the compressibility of the water is so small (about $5 \cdot 10^{-10}$ [ms²/kg]) that $\vec{\text{grad}} \rho$ can be neglected again, as compared to the other terms. With this approximation we obtain for the second term of the mass balance equation

$$\text{div}(\rho \vec{q}) \approx \rho \cdot \text{div} \vec{q} \quad 2-13$$

Replacing the simplified time derivative term (equation 2-11) and the simplified convective divergence term (equation 2-13) in equation 2-7, we have

$$\rho S \frac{\partial h}{\partial t} + \rho \cdot \text{div} \vec{q} + \rho Q = 0$$

Dividing by the density ρ we obtain

$$S \frac{\partial h}{\partial t} + \text{div} \vec{q} + Q = 0 \quad \text{or} \quad 2-14$$

$$S \frac{\partial h}{\partial t} + \text{div} \left(- \frac{\rho g \bar{k}}{\mu} \vec{\text{grad}} h \right) + Q = 0$$

with $h = (p/\rho g) + z$. This is the classical "diffusivity equation" or "volume balance equation" used by hydrogeologists for describing simple groundwater flow. For steady state flow the time derivative is zero and 2-14 simplifies to

$$\text{div} \left(- \frac{\rho g \bar{k}}{\mu} \vec{\text{grad}} h \right) + Q = 0 \quad 2-15$$

which is the basic equation that will be used in the regional and local finite element models.

3. FINITE ELEMENT FORMULATION OF THE FLOW EQUATION

The finite element method was originally developed as a tool for structural engineering analysis, but the theory and formulation have been progressively so refined and generalized that the method has been applied successfully to such other fields as groundwater flow and heat or solute transport. As there is a wide variety of publications describing the finite element techniques in detail (ZIENKIEWICZ, 1971; DESAI and ABEL, 1972; PINDER and GRAY, 1977; SEGERLIND, 1976, etc.), we will give in the following sections only a rather general presentation of the method.

In a quite general way, the basis of the finite element method is the representation of a volume under study by an assemblage of subdivisions called **finite elements** (Figure 3-1). These elements are interconnected at joints which are called **nodes** or **nodal points**. Simple functions (linear, quadratic or cubic polynomials) are chosen to approximate the unknown distribution of the actual field variable (in our problem: the hydraulic potentials or heads) over each finite element. If the actual field variable is replaced by the known type of **polynomial approximation** in the governing differential equation, this latter equation, multiplied by an appropriate weighting function, can be integrated over each element yielding a set of N simultaneous, generally linear, equations for the N nodal points. The solution of the linear system will give the approximate values of the field variable at the N nodal points. As the polynomial approximation (or interpolation function) is known for each element, the knowledge of the **nodal values** is sufficient to calculate the approximate value of the field variable at any point of the elements in the whole volume of interest.

3.1 Element types used in the model

One dimensional (1-D), two dimensional (2-D), and three dimensional (3-D) elements may be assembled to represent the shape of the flow region and the heterogeneous and/or anisotropic structure of the permeability field. Each element is defined simultaneously in the **global space** by the **global coordinates** of the nodal points, and in a **local space**, by the **local coordinates** of the nodes (Figure 3-2 and ZIENKIEWICZ, 1971).

In the **local space** all elements have a very simple and regular shape, due to the fact that the normalized local coordinates s_i^1 and the "natural" areal or volume coordinates L_j^j take their values on the intervals: $[-1; +1]$ and $[0; \frac{1}{2}; 1]$, respectively. (Note that in the expressions s_i^1 and L_j^j the letters i and j are not exponents, but simply superscripts.)

SAMPLE ELEMENT NETWORK

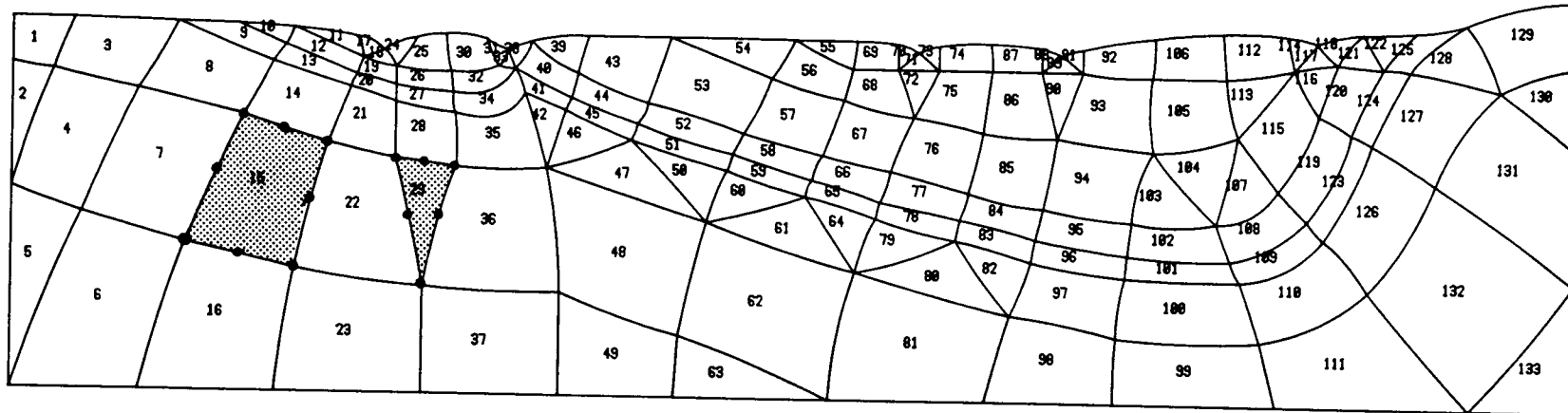


Figure 3-1: Typical Finite Element Network with 8-Node and 6-Node Elements

By definition each element **corner** is a nodal point having the local coordinates of $s^i = (-1 \text{ or } +1)$ or $L^j = (0 \text{ or } 1)$. If an element has only corner nodes, there will be two nodal points on each element side and only a linear interpolation of the nodal values can be performed. This type of element is called a "linear element". By introducing mid-side nodes between the corners we can create a "quadratic element": there will be three nodal points on each element side (or edge) which allow a quadratic interpolation of the nodal values. In the finite element code FEM 301 we use **exclusively quadratic elements** (Fig. 3-2), with quadratic interpolation functions.

The interpolation functions $N_n(s^i)$ depend only on the local coordinates. They are quadratic polynomials in the local coordinates s^i or L^j (see ZIENKIEWICZ, 1971). If a given element has NP nodal points, there will be NP interpolation functions which must satisfy the following conditions:

- for any point s^i ($-1 \leq s^i \leq +1$): $\sum N_n(s^i) = 1$; $n = 1, NP$.
- for a given nodal point m , $N_n = 0$ if $n \neq m$, and $N_n = 1$ if $n = m$ (at a nodal point all interpolation functions have value zero, excepting one!).

Now, let h be an unknown type of function over an element. If we know the nodal values h^n at the nodal points, we can approximate h by a quadratic function h^* at any point s^i of the element:

$$h^*(s^i) = N_n(s^i) \cdot h^n ; n = 1, NP \quad 3-1$$

(Note that in equation 3-1 we make use of the summation convention: $N_n h^n = \sum_{n=1}^{NP} N_n \cdot h_n$)

Next we have to define the "real" shape and size of the elements in the "real", **global space**. The global space is an ordinary three dimensional space related to an orthonormal cartesian reference system. This reference system, defined by the base vectors $\vec{e}_i = \{\vec{e}_1, \vec{e}_2, \vec{e}_3\}$ and the global coordinates $x^i = \{x^1, x^2, x^3\}$ will be the same for all elements of the model.

Making use of the quadratic interpolation functions $N_n(s^i)$ we can define **curvilinear, distorted elements** in the global space: if we give the global coordinates x^{ni} of the nodal points, the element sides (or edges) will be pieces of quadratic curves (parabolic curves) passing through the given nodal points. The interpolation scheme:

$$x^i(s^j) = N_n(s^j) \cdot x^{ni}; n = 1, NP \quad 3-2$$

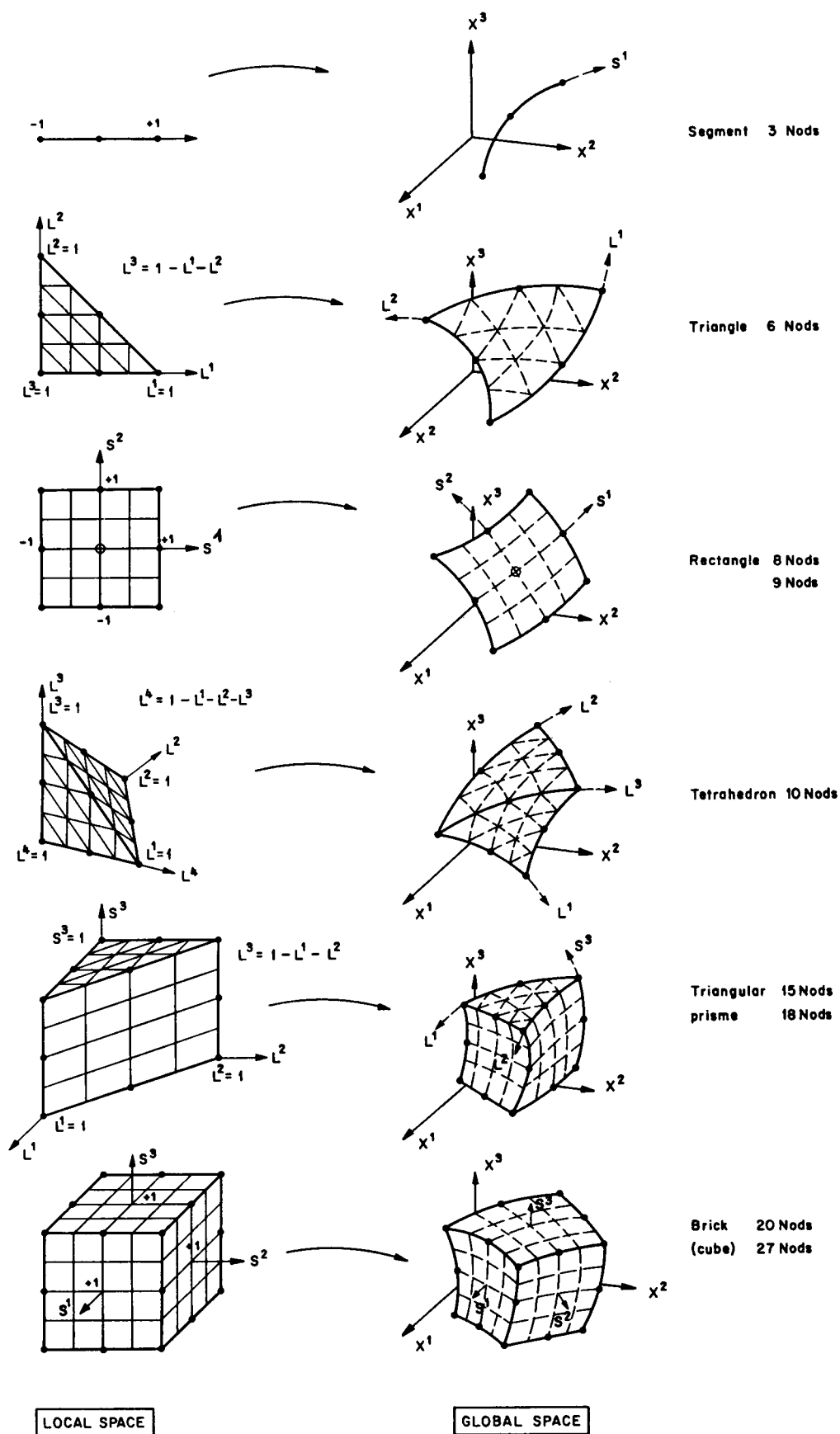


Figure 3-2: Global and Local Coordinates of 1-, 2-, and 3-Dimensional Elements

may be considered as the mapping of a straight-sided "parent" element defined in the local space into a curved-sided element represented in the global space. In Figure 3-2 it appears clearly that **in the global space the local coordinates s^i or L^j are distorted to a new set of curvilinear coordinates.** The orientation of this local coordinate system changes from element to element, and from point to point, even in the interior of the same element. In the global space the interpolation scheme 3-2 may be interpreted simply as a coordinate transformation between the curvilinear local coordinates s^i or L^j , and the cartesian global coordinates x^k . Uniqueness of the transformation is assured if the element is not unreasonably distorted. A set of curvilinear local coordinates will correspond to a set of global cartesian coordinates, and only one such set.

When 1-D or 2-D elements are mapped into the 3-D global space, the number of the independent local coordinates s^i or L^j will be less than the number of the global coordinates x^k , and the transformation matrix

$$D_i^k = \frac{\partial x^k}{\partial s^i} \quad 3-3$$

will not be any more an invertible square matrix. This point will be important when calculating the hydraulic gradient $\partial h / \partial x^k$ in 1-D and 2-D curvilinear elements (see section 3.3).

We have seen that the interpolation functions $N_n(s^i)$ play a very important role in:

- defining the "real" geometry of the elements in the global, cartesian coordinates (equation 3-2)
- defining functions of a given order (linear, quadratic, etc.) over the "real" elements for the unknown (equation 3-1).

If the geometry of an element and the distribution of a field variable over the element are described by the same interpolation functions $N_n(s^i)$, then we call this element **"isoparametric"**.

In the finite element code FEM 301 we use exclusively **quadratic isoparametric elements**, which are listed below:

Dimension of elements	Name and local coordinates	Number of nodes
1-D	Segment (s^1)	3
2-D	Triangle (L^1, L^2, L^3) with $L^3 = 1-L^1-L^2$	6
2-D	Rectangle (s^1, s^2)	8
2-D	Rectangle (s^1, s^2), (Lagrange)	9
3-D	Tetrahedron (L^1, L^2, L^3, L^4) with $L^4 = 1-L^1-L^2-L^3$	10
3-D	Triangular prism (L^1, L^2, L^3, s^3) with $L^3 = 1-L^1-L^2$	15
3-D	Triangular prism (Lagrange) (L^1, L^2, L^3, s^3) with $L^3 = 1-L^1-L^2$	18
3-D	Brick (s^1, s^2, s^3)	20
3-D	Brick (s^1, s^2, s^3), (Lagrange)	27

In the global space, elements may undergo further deformations. Some element sides (edges) or faces can be reduced to a "point" by giving the same node number (that is the same global coordinates) to more than one nodal point. In such situations care must be taken, however, to assure the compatibility with neighbouring elements (C_0 -continuity).

3.2 The GALERKIN formulation of the flow equation for finite elements

Although the governing differential equation solved by code FEM 301 is equation 2-15, we will give here the Galerkin formulation for the more general transient equation 2-14:

$$S \frac{\partial h}{\partial t} + \text{div} (-\bar{K} \text{ grad } h) + QV = 0$$

or in another form:

$$S \frac{\partial h}{\partial t} + \frac{\partial}{\partial x^i} (-K^{ij} \frac{\partial h}{\partial x^j}) + QV = 0 \quad 3-4$$

where:

$h = (p/\rho g) + z = \text{hydraulic head or potential (m)}$

$\bar{K} = K^{ij} = (\rho g k^{ij})/\mu = \text{volume conductivity tensor (m/s)}$

$S = \text{volume storativity (specific storativity) (1/m)}$

$QV = \text{volume source/sink term (m}^3/\text{s} \cdot \text{m}^3)$

The solution must satisfy the following boundary conditions:

- $h(x^i) = H(x^i)$ on S_1 (prescribed head boundary)
- $\vec{q}(x^i) \cdot \vec{n}(x^i) = q^j n_j = QS(x^i)$ on S_2 (prescribed normal flux boundary) where: $\vec{n} = n_j$ is the normal vector to boundary S_2 , $\vec{q} = q^j = -\bar{K} \vec{\text{grad}} h$ is the volume flux as defined by Darcy's law, and QS is the normal flux through boundary S_2 .

Equations 2-14 and 3-4 may be expressed by

$$L(h) = 0 \quad 3-5$$

where L is a differential operator. As $h(x^i)$ is an unknown type of function, we will approximate it over each element by a known type of polynomial:

$$h^* = N_n h^{*n}, \quad n = 1, NP$$

where N_n are the interpolation functions as defined in section 3.1, and h^{*n} are the approximate head values at the NP nodal points. If we replace h by h^* , equation 3-5 will not be satisfied exactly:

$$L(h^*) = L(N_n h^{*n}) = R \neq 0$$

where R is a residual, different from zero. In order to obtain the "best" approximate solution for the nodal values h^{*n} , we want the function h^* to satisfy two conditions:

1. the approximate solution h^* must satisfy the boundary conditions as defined above;
2. the integral of the weighted residuals R over an element (and over the entire flow region) has to be zero:

$$\int_V W \cdot L(h^*) dV = 0 \quad \text{or} \quad \int_V W_m \cdot L(N_n h^{*n}) dV = 0; \quad n=1, NP; m=1, NP$$

where the W_m are some kind of weighting functions depending on space coordinates s^i or x^j .

This procedure is the quite general "weighted residual" approach, as the choice of the weighting functions W_m is left open. If we choose the interpolation functions N_m as weighting functions ($W_m = N_m$), we obtain the GALERKIN^m formulation of the flow problem:

$$\int_V N_m \cdot L(N_n \cdot h^{*n}) dV = 0 \quad \text{or} \quad \int_V N_m \cdot S \frac{\partial h}{\partial t} dV + \int_V N_m \cdot \frac{\partial}{\partial x^i} (-K^{ij} \frac{\partial N_n}{\partial x^j} h^{*n}) dV + \int_V N_m \cdot QV \cdot dV = 0 \quad 3-6$$

In the second volume integral we may "eliminate" the second derivatives by making use of GREEN's first identity:

$$\int_V N_m \cdot \frac{\partial}{\partial x^i} (-K^{ij} \frac{\partial N_n}{\partial x^j} h^{*n}) dV = \int_V \frac{\partial N_m}{\partial x^i} K^{ij} \frac{\partial N_n}{\partial x^j} h^{*n} dV + \int_S N_m \underbrace{(-K^{ij} \frac{\partial N_n}{\partial x^j} h^{*n}) \cdot n_i}_{\text{normal flux QS on the boundary}} dS$$

So, the second integral in 3-6 is split into a volume integral which contains only first derivatives, and in a surface integral which automatically incorporates the normal flux QS on the boundary (either prescribed or calculated during solution). Substituting this result in 3-6, and supposing that QV and QS are constant over an element, we obtain:

$$\int_V S \cdot N_m \cdot N_n \cdot \frac{\partial h^{*n}}{\partial t} dV + \int_V \frac{\partial N_m}{\partial x^i} K^{ij} \frac{\partial N_n}{\partial x^j} h^{*n} dV + QV \int_V N_m dV + QS \int_S N_m dS = 0$$

Where $NV_m = \int_V N_m \cdot dV$ is the volume of influence, and $NS_m = \int_S N_m \cdot dS$ is the surface of influence of the m-th nodal point.

This is already a system of linear equations which can be written in the very simple form of:

$$C_{nm} \frac{\partial h^{*n}}{\partial t} + A_{nm} h^{*n} + QT_m = 0 \quad 3-7$$

$$\text{with: } C_{nm} = \int_V S \cdot N_m \cdot N_n dV \quad 3-8a$$

$$A_{nm} = \int_V \frac{\partial N_m}{\partial x^i} K^{ij} \frac{\partial N_n}{\partial x^j} dV \quad 3-8$$

$$QT_m = \int_V QV N_m \cdot dV + QS \int_S N_m \cdot dS \quad 3-9$$

The program **FEM 301** solves 3-7 without the time-dependent term (transient flow is simulated by another code): $\partial h^n / \partial t = 0$

The matrixes A_{mn} and C_{mn} are NP by NP symmetrical square matrixes which depend only on the geometry of the elements, and on the permeability tensor K_{ij}^1 and the volume storativity S .

The vector Q^T represents the total source/sink terms for the region of influence of each nodal point. In a 3-D model Q^T is generally zero and Q^T becomes the vector of the nodal discharges (or nodal infiltrations) on the boundaries. By solving the linear system 3-7 we obtain the nodal head h^n . The nodal discharge or infiltration Q^T is calculated for nodes where the head h^n is prescribed.

In order to obtain the correct numerical values for the coefficient matrix A_{nm} and for the "constant vector" Q^T_m , we have still two problems to deal with:

- the calculation of the "gradient matrix"
- the numerical integration over the curved-sided, distorted elements.

3.3 Calculation of the gradient matrix $B_{jn} = \partial N_n / \partial x^j$

In the orthonormal global system the hydraulic gradient is defined by

$$\vec{\text{grad}} h = \frac{\partial h}{\partial x^i} \vec{e}^i = J_i \vec{e}^i$$

Recalling that $h = N_n(s^j) \cdot h^n$ (in the following sections we will drop the sign $*$ from h^{*n}), we obtain for J_i :

$$J_i = \frac{\partial N_n}{\partial x^i} h^n = B_{in} \cdot h^n$$

We have to emphasize that multiplying the nodal heads h^n by the gradient matrix $B_{in} = \partial N_n / \partial x^i$ yields, for any point s^j of an element, the **Cartesian components J_i** of the gradient vector. The principal problem in evaluating the numerical value of $\partial N_n(s^j) / \partial x^i$ is that the coefficients $N_n(s^j)$ are given as functions of the local coordinates s^j only, but we have to form the partial derivatives with respect to the global, cartesian coordinates x^i ! To overcome this difficulty, most standard textbooks (ZIENKIEWICZ, 1971; DESAI and ABEL, 1972; etc.) propose a very simple technique. By the chain rule we have

$$\frac{\partial N_n}{\partial s^j} = \frac{\partial N_n}{\partial x^i} \cdot \frac{\partial x^i}{\partial s^j} \quad \text{with} \quad \frac{\partial x^i}{\partial s^j} = \frac{\partial N_n}{\partial s^j} x^{ni} \quad 3-10$$

and

$$\frac{\partial N_n}{\partial x^i} = \frac{\partial N_n}{\partial s^j} \left[\frac{\partial x^i}{\partial s^j} \right]^{-1} = B_{in} \quad 3-11$$

This method works well when the number of the independent local coordinates s^j is the same as the number of the cartesian coordinates x^i (3-D elements in a 3-d global space), that is, when $\partial x^i / \partial s^j$ is an invertible square matrix. In this case we can calculate the numerical value of $\partial N_n / \partial x^i$ at any point s^j of an element by using only the derivatives of $N_n(s^j)$ with respect to the local coordinates s^j (the $\partial N_n / \partial s^j$ are easy to compute), and the global coordinates x^{in} of the nodal points (which are fixed by the geometry of the elements). As a matter of fact, the gradient matrix B_{in} depends only on the geometry of the elements!

When 1-D or 2-D elements are mapped into a 3-D global space, the number of the independent local coordinates s^j will be less than the number of the global coordinates x^i , the transformation matrix $\partial x^i / \partial s^j$ will not be any more an invertible square matrix, and equation 3-11 cannot be used to calculate the gradient matrix $\partial N_n / \partial x^i$. We propose a more general method (KIRALY, 1979) which may be applied to 1-D, 2-D and 3-D elements mapped in a 3-D global space. This method is based on the well known representation of the gradients in curvilinear coordinates (see for example: KLINGBEIL, 1966; TEICHMANN, 1964), and is illustrated in figure 3-3. In the curvilinear local system we express the gradient vector (at any point s^k of the element) by the **covariant base vectors** $\vec{a}_k$ and the **contravariant components** of the gradient, JL^k :

$$\vec{\text{grad}} h = \vec{a}_k JL^k \quad 3-12$$

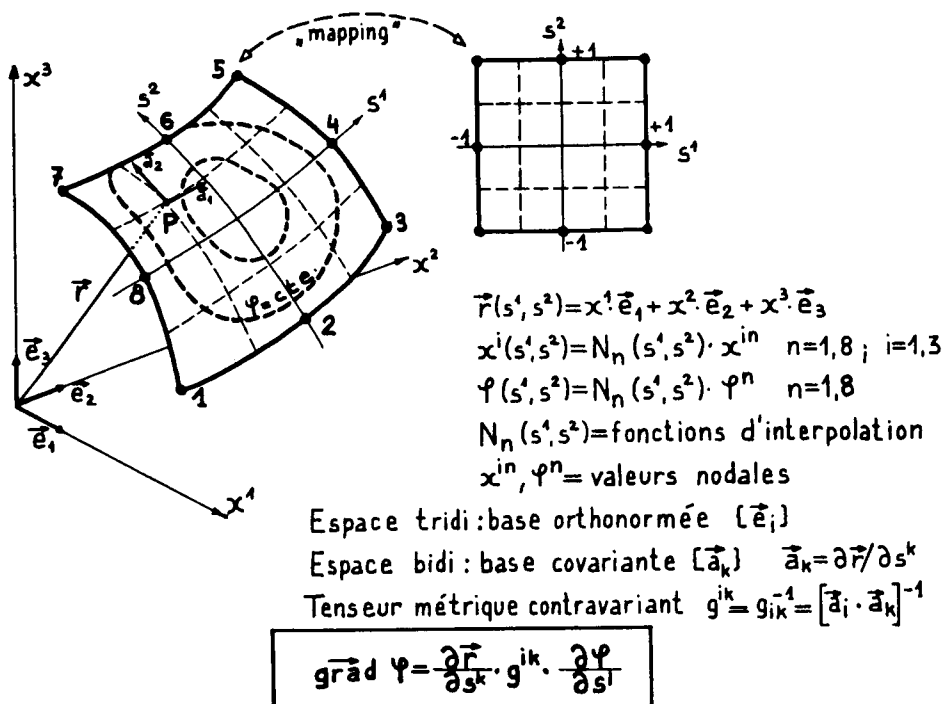


Figure 3-3: Representation of Gradients in Curvilinear Coordinates

Note that the covariant base vectors $\vec{a}_k$ are tangent to the curvilinear coordinate lines and their orientations vary from point to point. They are defined by:

$$\vec{a}_k = \frac{\partial \vec{r}}{\partial s^k} = \vec{e}_j \frac{\partial x^j}{\partial s^k} = \vec{e}_j \frac{\partial N_n}{\partial s^k} x^{jn} \quad \frac{\partial x^j}{\partial s^k} \quad 3-13$$

The covariant base vectors $\vec{a}_k$ are not only easily calculated by 3-13 but, in addition, they are expressed directly in the global system. The matrix $\partial x^j / \partial s^k$ contains the cartesian components of the $\vec{a}_k$.

The contravariant components of the gradient are almost as easily obtained:

$$JL^k = g^{ik} JL_i = g^{ik} \frac{\partial h}{\partial s^i} = g^{ik} \frac{\partial N_n}{\partial s^i} h^n \quad 3-14$$

where $JL_i = \partial h / \partial s^i$ are the covariant components of the gradient. As we need the contravariant components JL^k , we have to multiply JL_i by the **contravariant metric tensor** g^{ik} . The symmetric square matrix g^{ik} is computed by making use of the covariant base vectors $\vec{a}_k$:

$$g^{ik} = [g_{ik}]^{-1} = [\vec{a}_i \cdot \vec{a}_k]^{-1} \quad 3-15$$

where g_{ik} is the covariant metric tensor (a symmetric square matrix, thus invertible). Replacing the values of $\vec{a}_k$ (equation 3-13) and JL^k (equation 3-14) in equation 3-12, we have

$$\vec{\text{grad}} h = \vec{e}_j \left[\frac{\partial x^j}{\partial s^k} \cdot g^{ik} \cdot \frac{\partial N_n}{\partial s^i} \right] \cdot h^n = \vec{e}_j \cdot B_n^j \cdot h^n = \vec{e}_j J^j$$

The term in brackets is clearly the gradient matrix and $J^j = B_n^j h^n$ are the cartesian components of the gradient vector in the global system (in a cartesian system $J^j = J_j$ and $\vec{e}^j = \vec{e}_j$!). Finally B_n^j is expressed as

$$B_n^j = \frac{\partial x^j}{\partial s^k} g^{ik} \frac{\partial N_n}{\partial s^i} = x^{mj} \frac{\partial N_m}{\partial s^k} g^{ik} \frac{\partial N_n}{\partial s^i} \quad 3-16$$

The only "extra work" which we have to do with respect to equation 3-11 (the "standard method") is the computation of g^{ik} and g_{ik} , but this extra work is largely compensated by the generality thus gained.

To make the method more "transparent" let us define the gradient matrix for the 8-node element in figure 3-3. To do this, we introduce a more explicit notation for the variables:

$$s = s^1, t = s^2; \quad x = x^1, y = x^2, z = x^3$$

$X_1 \dots X_8, Y_1 \dots Y_8, Z_1 \dots Z_8$ are the global coordinates of the nodal points and $[B]$ is the gradient matrix $\partial N_n / \partial x^j$:

$$[B]_{3 \times 8} = \begin{bmatrix} \frac{\partial x}{\partial s} & \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial s} & \frac{\partial y}{\partial t} \\ \frac{\partial z}{\partial s} & \frac{\partial z}{\partial t} \end{bmatrix} \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix}^{-1} \begin{bmatrix} \frac{\partial N_1}{\partial s} & \frac{\partial N_2}{\partial s} & \dots & \frac{\partial N_8}{\partial s} \\ \frac{\partial N_1}{\partial t} & \frac{\partial N_2}{\partial t} & \dots & \frac{\partial N_8}{\partial t} \end{bmatrix}$$

with
$$g_{11} = \left(\frac{\partial x}{\partial s} \right)^2 + \left(\frac{\partial y}{\partial s} \right)^2 + \left(\frac{\partial z}{\partial s} \right)^2$$

$$g_{22} = \left(\frac{\partial x}{\partial t} \right)^2 + \left(\frac{\partial y}{\partial t} \right)^2 + \left(\frac{\partial z}{\partial t} \right)^2$$

$$g_{12} = g_{21} = \frac{\partial x}{\partial s} \cdot \frac{\partial x}{\partial t} + \frac{\partial y}{\partial s} \cdot \frac{\partial y}{\partial t} + \frac{\partial z}{\partial s} \cdot \frac{\partial z}{\partial t}$$

and
$$\begin{bmatrix} \frac{\partial x}{\partial s} & \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial s} & \frac{\partial y}{\partial t} \\ \frac{\partial z}{\partial s} & \frac{\partial z}{\partial t} \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & \dots & x_8 \\ y_1 & y_2 & \dots & y_8 \\ z_1 & z_2 & \dots & z_8 \end{bmatrix} \begin{bmatrix} \frac{\partial N_1}{\partial s} & \frac{\partial N_2}{\partial t} \\ \frac{\partial N_2}{\partial s} & \frac{\partial N_2}{\partial t} \\ \dots & \dots \\ \frac{\partial N_8}{\partial s} & \frac{\partial N_8}{\partial t} \end{bmatrix}$$

Again, the numerical value of $\partial N_n / \partial x^j$ may be calculated at any point s^i of an element if we know the formulas for the derivatives $\partial N_n(s^i) / \partial s^i$, and the global coordinates x^{jn} for the nodal points n (= the geometry of the element).

Once the value of the gradient matrix is determined we can evaluate the expression

$$\frac{\partial N_m}{\partial x^i} K^{ij} \frac{\partial N_n}{\partial x^j}$$

in equation 3-8. Our next problem is to integrate it over the curvilinear elements.

3.4 Numerical integration over the curvilinear elements

Expressions 3-8a, 3-8 and 3-9 have to be integrated over the curvilinear, distorted, "real" elements. As the integration limits would be much too complicated if expressed with respect to cartesian coordinates x^i , the integrals will be calculated in the local coordinate system. In this system the element sides or element faces are not only parallel to the local coordinate lines or coordinate surfaces, but the integration limits would simply be (-1 and +1) or (0 and +1).

If we integrate in the local system, the infinitesimal quantities dL (length in 1-D elements), dS (surface in 2-D elements) and dV (volume in 3-D elements) must be expressed by the covariant metric tensor g_{ik} , given by equation 3-15 (or the covariant base vectors $\vec{a}_k$ given by equation 3-13) and the differentials ds^k of the contravariant local coordinates. Bearing in mind that all 1-D, 2-D and 3-D elements are mapped in a 3-D global space, we have:

$$\begin{aligned}
 \text{1-D element: } dL &= \left(\det g_{jk} \right)^{1/2} ds^1 && (\text{or: } |\vec{a}_1| ds^1) \\
 \text{2-D element: } dS &= \left(\det g_{jk} \right)^{1/2} ds^1 ds^2 && (\text{or: } |\vec{a}_1 \times \vec{a}_2| ds^1 ds^2) \\
 \text{3-D element: } dV &= \left(\det g_{jk} \right)^{1/2} ds^1 ds^2 ds^3 \\
 &&& (\text{or: } |(\vec{a}_1 \times \vec{a}_2) \cdot \vec{a}_3| ds^1 ds^2 ds^3) \\
 \text{or: } dV &= \det \left[\partial x^i / \partial s^k \right] ds^1 ds^2 ds^3
 \end{aligned}$$

When the number of local coordinates s^k is the same as the number of global coordinates x^i , the determinant of $\partial x^i / \partial s^k$ may be used to express dV . However, using the metric tensor g_{jk} allows us to write a far more general expression for the integrals in the local coordinate system:

$$A_{nm} = \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} \frac{\partial N_m}{\partial x^i} K^{ij} \frac{\partial N_n}{\partial x^j} (\det g_{lk})^{1/2} ds^1 ds^2 ds^3 \quad 3-17$$

$$C_{nm} = \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} S \cdot N_m \cdot N_n \cdot (\det g_{lk})^{1/2} ds^1 ds^2 ds^3 \quad 3-17a$$

$$Q^T_m = QV \int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} N_m \cdot (\det g_{lk})^{1/2} ds^1 ds^2 ds^3 + QS \int_{-1}^{+1} \int_{-1}^{+1} N_m \cdot (\det g_{lk})^{1/2} ds^1 ds^2 \quad 3-18$$

Clearly enough, simple, double and triple integrals will apply to 1-D, 2-D and 3-D elements in spite of the fact that all elements are mapped in a 3-D global space.

While the limits of integration are simple, the explicit form of the integrand is not. Excepting the simplest elements (straight-sided triangle and tetrahedron), algebraic integration usually defies our mathematical skill and numerical integration has to be carried out. This is not a severe penalty and has the advantage that general programs, not tied to a particular element, can be written for various classes of problems. Details on numerical integration techniques can be found in ZIENKIEWICZ (1971) and in standard textbooks on numerical analysis (SCHEID, 1968).

In FEM 301 we apply the very popular GAUSS-LEGENDRE integration formulae. Let $f(y)$ be a polynomial function depending only on y , and being defined over the interval $y = -1$; $y = +1$. The main idea in the Gaussian integration is to calculate the numerical value of $f(y)$ at N_i discrete points, of coordinate y_n (called integration points or Gaussian points) and calculate the integral by the weighted sum of the N_i values $f(y_n)$:

$$\int_{y=-1}^{+1} f(y) dy = \sum_{n=1}^{N_i} W_n \cdot f(y_n)$$

where W_n is the weighting coefficient for the n -th integration point. If there are N_i integration points within the interval $[-1; +1]$, the result will be "exact" for all polynomials of degree $(2 N_i - 1)$ or less. With two Gaussian points we can integrate a cubic polynomial, and a quintic polynomial can be integrated with three points. **In FEM 301 we use 3 integration points in each local coordinate direction**, with the following local coordinates y_n and weighting coefficients W_n :

$$\begin{array}{ll} y_1 = - (3/5)^{1/2} & W_1 = 5/9 \\ y_2 = 0.0 & W_2 = 8/9 \\ y_3 = + (3/5)^{1/2} & W_3 = 5/9 \end{array}$$

Now, in order to show the operations more explicitly for multiple integrals, let us define the local coordinates by $s = s^1$, $t = s^2$, and $u = s^3$. Let f be a function, depending on local coordinates, which we have to integrate over 1-D, 2-D and 3-D elements (for example: f = the integrand in equations 3-17, 3-17a or 3-18). The results (for example: A_{nm} , C_{nm} , QT_m) will be obtained by the following operations:

1-D element: $f = f(s)$; 3 Gaussian points.

$$\int_{-1}^{+1} f(s) ds = \sum_{p=1}^3 f(s_p) \cdot W_p$$

2-D elements: $f = f(s, t)$: 9 Gaussian points.

$$\int_{-1}^{+1} \int_{-1}^{+1} f(s, t) ds dt = \sum_{p=1}^3 \sum_{q=1}^3 f(s_p, t_q) \cdot w_p \cdot w_q$$

3-D element: $f = f(s, t, u)$; 27 Gaussian points.

$$\int_{-1}^{+1} \int_{-1}^{+1} \int_{-1}^{+1} f(s, t, u) ds dt du = \sum_{p=1}^3 \sum_{q=1}^3 \sum_{r=1}^3 f(s_p, t_q, u_r) \cdot w_p \cdot w_q \cdot w_r$$

where $s_p = y_p$, $t_q = y_q$, and $u_r = y_r$ are the local coordinates of the Gaussian integration points in a given element.

So, we obtain the element matrixes A_{nm} , C_{nm} , Q^T_m by calculating the numerical value of the integrand in equations 3-17, 3-17a and 3-18 for each integration point and then forming the weighted sum of these numerical values for all Gaussian points of an element. The numerical integration over triangles or tetrahedrons is carried out in a similar way, excepting that we use 7 points for a triangle and 15 points for a tetrahedron and the coordinates of the integration points are defined by the surface or volume coordinates L^j .

Once the element matrixes are evaluated, we can form equation 3-7 for each element (the time dependent term is not considered here):

$$A_{mn} \cdot h^n = Q^T_m, \quad n = 1, NP; \quad m = 1, NP.$$

If there are NP nodal points in an element, there will be NP linear equations formed. Each of these represents the weighted continuity equation 2-15, integrated over the volume of influence of a nodal point in the given element. As most nodal points are common for several elements, the complete equation of the m-th node must contain the sum of all possible element contributions. By adding the element contributions to each equation we obtain, finally, the linear system 3-19, which represents the finite element formulation of the flow problem for the entire model:

$$\left[\sum_{LM=1}^{NLM} A_{nm} \right] \{ h^n \} = \left\{ \sum_{LM=1}^{NLM} Q^T_m \right\} \quad 3-19$$

where NLM is the number of elements in the whole model. The method of solving 3-19 is presented in chapter 4.

4. THE SOLUTION METHOD

Let us rewrite equation 3-19 in the simpler form of

$$M_{ik} \cdot h^i = Q_k \quad 4-1$$

where $M_{ik} = A_{ik}$ is the assembled "grand" matrix and $Q_k = Q_k^T$, for simplicity. The coefficient matrix M_{ik} is a positive, symmetric sparse matrix, generally with a banded structure. Two methods of solution have been highly developed for the solution of such systems of simultaneous equations:

- iteration, in which a successive approximation technique is used to converge on the true solution;
- direct solution, in which an "exact" (within round-off accuracy) solution is sought.

The method of "conjugate gradients" and the Gauss-Seidel procedure are frequently used for iterative solution and the Gauss elimination procedure for direct solution (a good overview of these methods can be found in SCHWARZ, 1980).

In FEM 301 we apply the **frontal solution method of B.M. IRONS (1970)**, which is a particularly useful modification of the Gauss elimination procedure. The frontal method is very efficient for large 3-D problems. To make clear the advantages of this method, we will first briefly describe the classical Gauss elimination procedure.

4.1 The Gauss elimination method

This method is one of the oldest, and still perhaps the best, for the treatment of linear systems. In the simplest case the equations are completely formed, that is, all element contributions are assembled in the "grand" coefficient matrix M_{ki} and in the right hand side "constant" vector Q_k . The main objective of the Gauss elimination method is to **reduce the N by N coefficient matrix M_{ki} to a triangular form M_{ki}^* in which all coefficients below the main diagonal are zero.** At the same time the constant vector Q_k is transformed into an appropriate form Q_k^* so, that the new system $M_{ki}^* \cdot h^i = Q_k^*$ will be equivalent (= giving the same solutions for h^i) to the original problem of equation 4-1 (see Figure 4-1).

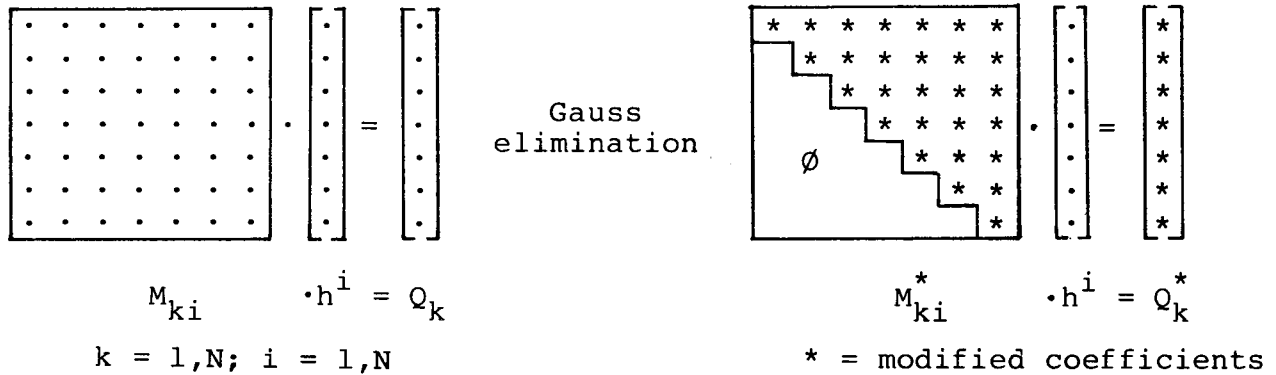


Figure 4-1

Schematic Representation of the Gauss Elimination Method

The "triangularization" procedure is based on the fact that we may multiply an equation by a constant, and then subtract (or add) it from (or to) another equation without changing the solution of the linear system. Let the original matrix be represented in the partitioned form of Figure 4-2:

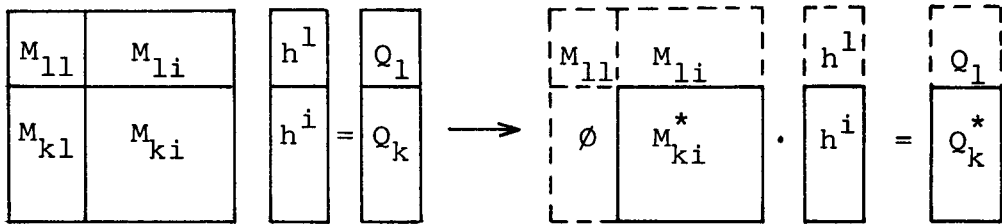


Figure 4-2

Partitioning of Matrixes in the Gauss Elimination Method

In Figure 4-2 M_{11} is a 1 by 1 matrix (the diagonal term)

M_{1i} is a 1 by (N-1) matrix (first row)

M_{k1} is a (N-1) by 1 matrix (first column)

M_{ki} is a (N-1) by (N-1) matrix

If we successively multiply the first equation by the constants M_{k1}/M_{11} and subtract it from the corresponding k-th equation, the modified coefficients M_{ki}^* and Q_k^* will be given by

$$M_{ki}^* = M_{ki} - M_{1i} \cdot M_{k1} \cdot M_{11}^{-1}$$

$$Q_k^* = Q_k - Q_1 \cdot M_{k1} \cdot M_{11}^{-1}$$

$i = 2, N$ and $k = 2, N$

Obviously, M_{k1}^* would be zero:

$$M_{k1}^* = M_{k1} - M_{11} M_{k1} M_{11}^{-1} = \emptyset$$

As we have zeros in the first column below the diagonal, the first variable (or unknown) h^1 is eliminated from all equations, excepting the first one.

This procedure may be repeated by partitioning the reduced, $(N-1)$ by $(N-1)$ matrix M_{ki} in the same way, and, step by step, we will obtain the triangular matrix sought for. This is the **forward elimination** phase of the Gauss method, where the general algorithm for the elimination of the n -th equation is given by 4-2:

$$M_{ki}^* = M_{ki} - M_{ni} \frac{M_{kn}}{M_{nn}}$$

$$Q_k^* = Q_k - Q_n \frac{M_{kn}}{M_{nn}} \quad 4-2$$

$$i = (n+1), N \quad \text{and} \quad k = (n+1), N$$

Note, that in 4-2, the coefficients without * could result (and generally they do) from previous transformations.

As the coefficients below the main diagonal are zero, the k -th equation (row) will contain only $(N-k+1)$ unknowns (the first $(k-1)$ coefficients of this row being zero, the corresponding variables are "eliminated" from the equation). Obviously, the last (N -th) equation will contain only one unknown $h^{(N)}$ which can easily be calculated from

$M_{(N,N)}^* \cdot h^{(N)} = Q_{(N)}^*$. Substituting the value of $h^{(N)}$ into the vector h^i , we can solve the $(N-1)$ -th equation

$$M_{(N-1,N-1)}^* \cdot h^{(N-1)} + M_{(N,N)}^* \cdot h^{(N)} = Q_{(N-1)}^*$$

which contains, again, only one unknown: $h^{(N-1)}$, easily determined. Stepping backwards, each equation can be solved for the values $h^{(N-2)}$, $h^{(N-3)}$, h^1 , and this procedure represents the **back substitution** phase of the Gauss solution method.

We have presented here the simplest direct method in which all terms of the matrix M_{ki} are stored and operated upon.

However, it is possible to take advantage of the **symmetry** that exists in the coefficient matrix by restricting operations to the upper triangular area. Furthermore, the typical assembled "grand" matrix contains many zero terms, and in particular there is a distance from the diagonal beyond which no terms exist (Figure 4-4a). This is called **banding** of the matrix and the distance from the diagonal term to the last term in any row is called the **half-bandwidth**, or **semi-bandwidth**.

Figure 4-3 shows a simple element network with 9 elements and 49 nodal points. The completely assembled matrix equation is shown in Figure 4-4a, where the dots represent the non-zero contributions of all elements to the coefficient matrix M_{ki} . This latter is actually a 49 by 49 banded and symmetric sparse matrix, with a maximum semi-bandwidth of $b = 17$.

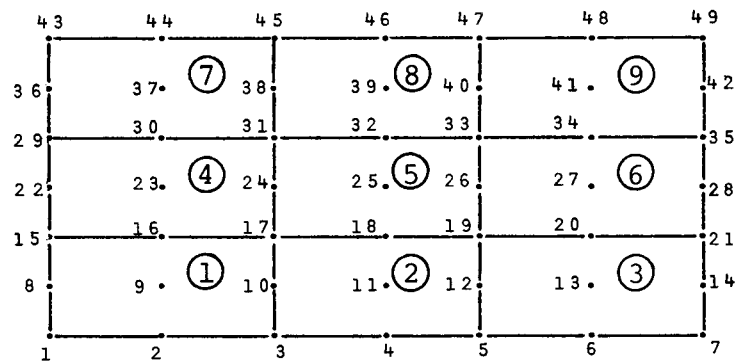


Figure 4-3
Simple 2-Dimensional Element Network

19 = node number
⑤ = element number

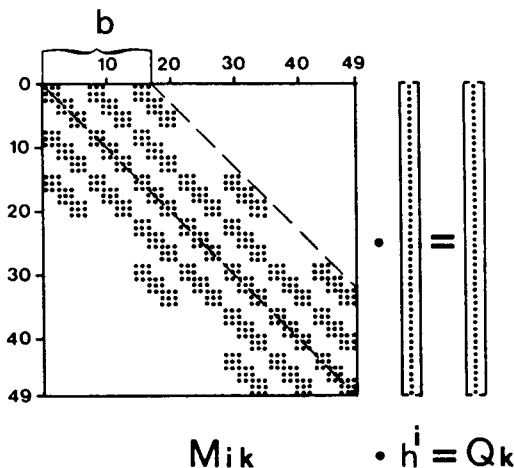


Figure 4-4a
Assembled coefficient matrix
for Figure 4-3

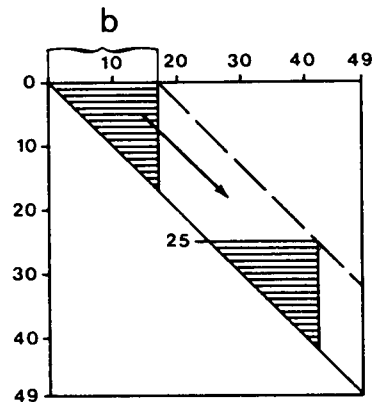


Figure 4-4b
Position of the triangular
area during elimination
of equations 1 and 25

Given the maximum semi-bandwidth b , the forward elimination technique modifies only a triangle of coefficients, which effectively moves diagonally downwards as elimination proceeds. Figure 4-4b shows clearly that only coefficients in the shaded triangular area will be modified during elimination of the successive equations. The number of coefficients, NC , in the triangular area depends on the maximum semi-bandwidth b :

$$NC = \frac{(b \cdot b - b)}{2} + b$$

The semi-bandwidth (as well as the core space needed for the solution) will depend on the **node numbering in the model**. It will be the largest spread of node numbers plus one in any element. For example, in element 1 of Figure 4-3 the lowest node number is 1, the highest is 17, and the semibandwidth is $(17-1)+1 = 17$ (in this particular example it is the same for each element).

The core requirement is a serious drawback of the classical Gauss elimination method when applied to finite element models. In large 3-D problems, using quadratic or higher order elements, the maximum semi-bandwidth could be as large as 3000 or 4000, even if the node numbering is already optimized. In addition, introducing new elements and new nodal points into an "old" model is cumbersome as generally we have to re-number all nodal points trying to keep the bandwidth as small as possible. Some of these difficulties are reduced or completely eliminated in the frontal solution method.

4.2 The frontal elimination method

The frontal elimination method is fully described in B.M. IRONS (1970). The same paper contains a FORTRAN version of the main routine in his frontal solution program. The fundamental principles of the frontal method are implied by the Gaussian process itself. Let us examine the general algorithm for the elimination of the n -th equation, given by formula 4-2:

$$\begin{aligned} M_{ki}^* &= M_{ki} - \left[M_{ni} \frac{M_{kn}}{M_{nn}} \right] \\ Q_k^* &= Q_k - \left[Q_n \cdot \frac{M_{kn}}{M_{nn}} \right] \end{aligned} \quad 4-2$$

$$k = (n+1), N \quad \text{and} \quad i = (n+1), N$$

where M_{ki} and Q_k are coefficients resulting from the $(n-1)$ -th elimination step. As a matter of fact, M_{ki} or Q_k is the sum of contributions arising from two sources:

- element contributions (in their "virgin", unmodified form $M_{ki} = \sum_{LM} A_{ki}$ and $Q_k = \sum_{LM} Q_k^T$, see equations 4-1 and 3-19).
- contributions arising from the elimination process itself ("process contributions") in the form of $[M_{ni} \cdot M_{kn}/M_{nn}] = GM_{ik}$ and $[Q_n \cdot M_{kn}/M_{nn}] = GQ_k$.

To be more explicit, we show the element contributions (summation over LM = elements), and the process contributions (summation over EQ = equations) separately:

$$M_{ki} = \sum_{LM} A_{ki} - \sum_{EQ} GM_{ki} \quad \text{and} \quad Q_k = \sum_{LM} Q_k^T - \sum_{EQ} GQ_k.$$

In the classical band algorithm, element contributions are first collected (the equations are assembled in their unmodified form), and then begins the elimination of the variables in a sequential order, generating the process contributions. In Figure 4-3, for example, before eliminating the variable 8 (h^8), we have to eliminate the first 7 variables and we have to assemble, at least, the first 3 elements.

However, as it was pointed out by B.M. IRONS (1970), it does not matter in what order the additions or subtractions are made, provided the coefficients of the n -th equation, about to be eliminated, are correct: i.e. provided that M_{ni} , M_{kn} , M_{nn} and Q_n in formulae 4-2 are fully summed (= contain all necessary element contributions). If this condition is satisfied, all process contributions GM_{ki} and GQ_k will automatically be correct, too, and we may subtract them from the remaining coefficients M_{ki} , Q_k even if these are not yet fully summed. The lacking element contributions may be added after the elimination of the n -th equation. This is the **basic idea of the frontal method**, and the frontal process alternates between **accumulation of element coefficients (assembly)**, and **elimination of the variables**, so that the "virgin" equations rarely appear, fully summed yet unmodified.

As assembling element contributions and eliminating variables go hand in hand, each time we pick up an element, we have to examine the "appearances" of the variables related to the nodal points. When a variable appears for the first time in an element (i.e. it was not contained in the previous elements), the "construction" of the corresponding equation is started on core. When a variable appears in an element,

but this is not its first or last appearance (intermediate appearances), we proceed with adding the element contributions to the coefficients of the corresponding equation. Finally, when a variable appears for the last time in an element (i.e. there are no subsequent elements containing this variable), the coefficients of the corresponding equation are fully summed and we may eliminate the variable. As there will be no more contributions to this equation, we may discard it from core, and store it on a file.

Between its first and last appearances a variable is called "active" and its equation remains on core. However, this equation (let us say, the n -th equation) will **contain only the coefficients M_{ni} ($= M_{in}$) which relate to the other currently active variables**. Indeed, there is no reason to relate variable n to another variable m which is not active. If m is already eliminated, the coefficient M_{nm} ($= M_{mn}$) would be zero, and if m has not appeared yet, the coefficient M_{nm} would be zero, too. Now, it is possible that by assembling further elements m becomes active before n is eliminated, but **only then** will be time to introduce the term M_{nm} ($= M_{mn}$), **not before!**

Most technical peculiarities encountered in the frontal solution method are only consequences of the few basic ideas mentioned above:

- a. Only the currently active variables will have their equation on core. The number NA of these currently active variables is called the **frontwidth**, and it depends on the order in which the elements are presented (i.e. on the element numbering).
- b. Each equation will contain only NA terms (these terms relate only to the other active variables!), and the active coefficients M_{ki} on core will form an NA by NA symmetric square matrix (called CGR in the program). As CGR is symmetric, we may operate in the upper triangular part only, where we need

$$NC = \frac{(NA \cdot NA - NA)}{2} + NA$$

places for assembling the current element contributions, and for eliminating the variables which appear for the last time.

- c. As a consequence of point b, core requirements will not depend any more on the node numbering, but **only on the frontwidth NA** , i.e. on the element numbering. Just as node numbering is critical in a band algorithm, element numbering will be critical in the frontal solution method.

As a matter of fact, ordering of the nodal points (and of the variables) is irrelevant to the frontal technique. The node numbers are only "names" identifying the particular nodal points or variables.

- d. When a variable appears for the first time in an element, its equation may occupy the first free column (and row) encountered in CGR (the associated node number does not play any role in choosing the position of an equation). The element contributions can still be added to the correct coefficients, provided we have a "pointer vector" (called MVBL in the program) which indicates the "names" (i.e. the numbers) of the equations occupying the successive columns (and rows) in CGR.
- e. The elimination of the variables is not sequential. The equation to be eliminated could occupy any row and column in the coefficient matrix. After having substracted the "process contributions" from the remaining coefficients, all terms of the eliminated equation may be discarded from CGR, leaving free rows and columns for the new equations. In contrast to the band algorithm, the coefficients need not be shifted diagonally after each elimination step. If an equation is entered, for example, in row 10 and column 10, it will remain there until it is eliminated (see Figures 4-5 and 4-6).
- f. As the elimination of the variables is carried out elementwise, the **solutions** also will appear element by element (but in reverse order) during the back-substitution phase.
- g. The central point in the frontal method is the correct elementwise assembling, eliminating and solving of the equations. Thus, each element data set should carry informations about the "appearance" of the nodal points (first, intermediate, last), and about the position of the corresponding equations in the coefficient matrix CGR (row or column). These informations are generated by a "**symbolic elimination**" routine (called LSYMB in the program) **before numerical calculations are carried out**. The results are stored in short integer vectors (LDEST in the program) which will accompany each element data set, and which will allow us to update the "pointer vector" MVBL (see point d) each time an element is processed.
- h. Contributions to the right hand-side of the equations (source/sink terms GT_k and process contributions GQ_k) are assembled and eliminated in a vector called RHS^k . It has the same length as the rows of CGR, and the equations are stored in the same order as in the rows of CGR.

In order to illustrate some of the concepts mentioned in points a to h, let us apply the frontal elimination method to the simple element network of Figure 4-3. The updated pointers MVBL are shown in Figure 4-5, and Figure 4-6 shows the structure of the coefficient matrix CGR each time an element was processed (to be more clear, we present the whole square matrix).

In Figure 4-5, the vectors LVB(k), $k = 1, KR$ represent the "variable names" (= node numbers) for each element defined by KR nodal points (in this example, $KR = 9$ for each element). The first step in the **symbolic elimination** is to place the node numbers of the first element into MVBL (see Figure 4-5). The value of LDEST(k) is defined as the "address" of variable LVB(k) in the pointer vector MVBL and this value will simply indicate in which row and column of CGR the equation of variable LVB(k) has to be assembled. For example, if $k = 8$ in element 1, the equation of variable $LVB(8) = 16$ must be assembled in the $LDEST(8) = 8$ -th row and column of CGR (Figures 4-5 and 4-6). Quite generally: an element contribution $A(k,i)$, related to variables LVB(k) and LVB(i), must be added to CGR [LDEST(k), LDEST(i)]. Scanning the elements (from 2 to 9) shows that variables 1, 2, 8 and 9 appear for the last time in element 1. As they have to be eliminated, a special code is generated for the corresponding terms in LDEST, and hence the LDEST vector carries all information we need, how to add the element contributions to CGR, and which variables to eliminate afterwards. Finally, we have to discard the eliminated "variable names" from MVBL (i.e. clear columns 1, 2, 4 and 5), in order to make place for the new ones.

Now we may take element 2 and enter the new node numbers into the free places of MVBL; nodes 4 and 5 in columns 1 and 2, nodes 11 and 12 in columns 4 and 5, and nodes 18 and 19 in columns 10 and 11. The LDEST vector is created as before by recording the addresses of the variables in MVBL. By scanning the elements (from 3 to 9) for the last appearances equations 3, 4, 10 and 11 may be eliminated, thus a special code is generated for the corresponding terms in LDEST. The **symbolic elimination** is achieved by processing each element in the same way, and by keeping the LDEST vectors and the node numbers (= LVB vectors) on a file.

During the symbolic elimination we record the **maximum frontwidth** (= maximum number of active variables). It is 13 for the element network of Figure 4-3, so there will be a maximum of 91 coefficients in the upper triangle of CGR. With the band algorithm we obtained a maximum semi-bandwidth of 17 for the same element network, and we would

need 153 coefficients to achieve the elimination procedure in an upper triangular matrix (see Figures 4-4a and 4-4b). The difference between the core requirements would be, naturally, much more important if the two procedures (frontal method and band algorithm) were applied to large 3-D problems.

ELEM NBR.	NOD NUMBERS - LVB									ACTIVE VARIABLES - MVBL												
	L D E S T VECTOR									1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	2	3	8	9	10	15	16	17													
	①*	②*	3	④*	⑤*	6	7	8	9	①	②	3	⑧	⑨	10	15	16	17				
2	3	4	5	10	11	12	17	18	19													
	③*	①*	2	⑥*	④*	5	9	10	11	④	5	③	⑪	12	⑩	15	16	17	18	19		
3	5	6	7	12	13	14	19	20	21													
	②*	①*	③*	⑤*	④*	6*	11	12	13	⑥	⑤	⑦	⑬	⑫	⑭	15	16	17	18	19	20	21
4	15	16	17	22	23	24	29	30	31													
	⑦*	⑧*	9	①*	②*	3	4	5	6	⑫	⑬	24	29	30	31	⑮	⑯	17	18	19	20	21
5	17	18	19	24	25	26	31	32	33													
	⑨*	⑩*	11	③*	①*	2	6	7	8	⑮	26	⑭	29	30	31	32	33	⑰	⑱	19	20	21
6	19	20	21	26	27	28	33	34	35													
	⑪*	⑫*	⑬*	②*	①*	③*	8	9	10	⑰	⑱	⑲	29	30	31	32	33	34	35	⑲	⑳	㉑
7	29	30	31	36	37	38	43	44	45													
	④*	⑤*	6	①*	②*	3	⑪	⑫*	13	⑳	㉑	38	⑲	⑳	31	32	33	34	35	㉒	㉓	45
8	31	32	33	38	39	40	45	46	47													
	⑥*	⑦*	8	③*	①*	2	⑬*	④*	5	㉒	40	㉓	④	47	㉔	㉕	33	34	35			㉖
9	33	34	35	40	41	42	47	48	49													
	⑧*	⑨*	⑩*	②*	①*	③*	⑤*	④*	⑥*	㉔	㉕	㉖	④	④	④		㉗	㉘	㉙			

Figure 4-5: Symbolic Elimination of the Variables for the Element Network of Figure 4-3.

- ⑲ : variable which has to be eliminated
- ①* : special code in LDEST for last appearance of the corresponding variable

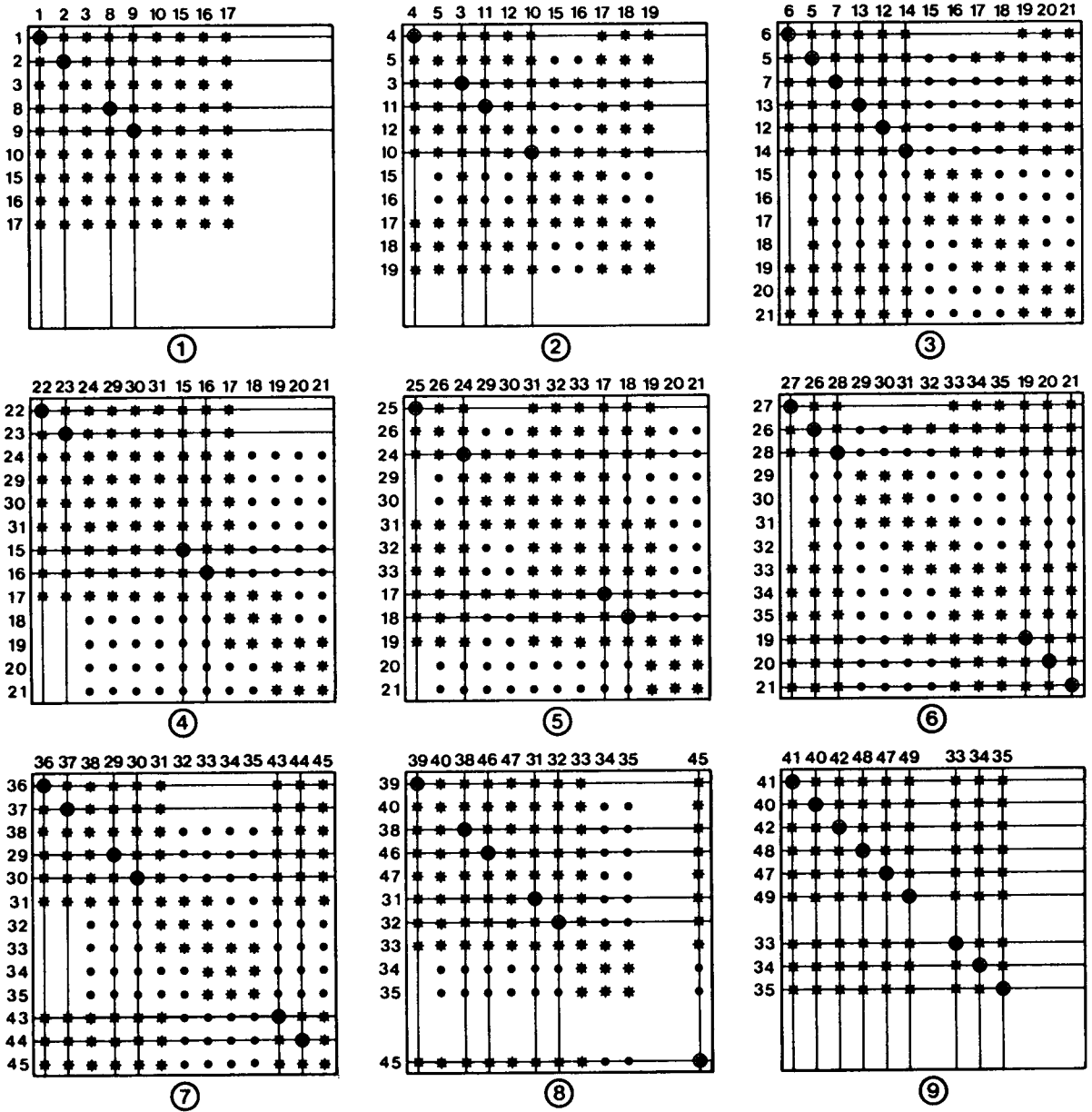


Figure 4-6: Matrix CGR for each element in Figure 4-3 (element contributions are assembled and last appearing variables are eliminated)

The difference between the two solution methods also appears when we introduce changes in the topology of the element network. Suppose we would like to replace element 2 in Figure 4-3 by a finer discretization (i.e. by 6 small elements) such as shown in Figure 4-7.

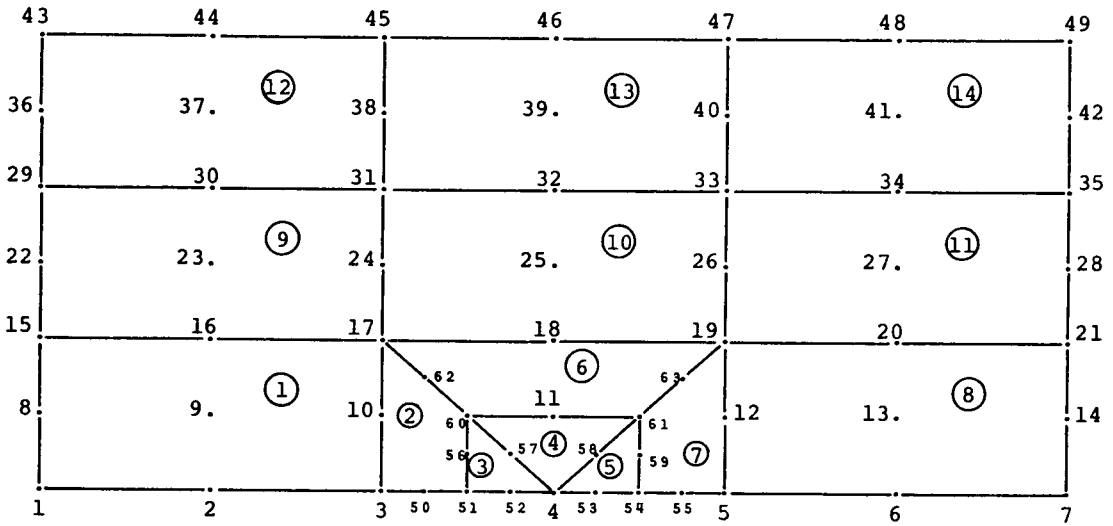


Figure 4-7: Rediscretization of the Simple Network

23 = nodal point

(10) = element number

New nodal points: 50,51,52,53,54,55,56,57,58,59,60,61,62,63.
 New elements: 2,3,4,5,6,7 for element 2 in Figure 4-3.
 Maximum frontalwidth: 13. (old frontwidth: 13)

With the new elements we have introduced 14 new nodal points (from 50 to 63), but we have not changed the old node numbers (from 1 to 49). Applying the symbolic elimination procedure to the elements in Figure 4-7 yields a maximum frontwidth of 13, just as before. Solving the same problem with the band algorithm would require the complete renumbering of the nodal points, and even in this case, the maximum semi-bandwidth would be greater than before (about 30 instead of 17).

Once the symbolic elimination has been achieved and the size of CGR defined, the **numerical calculations** may be carried out, element by element. The first step in the numerical processing of an element is to update the pointer vector MVBL by making use of the LDEST and LVB vectors:

$$\text{MVBL} [\text{LDEST}(k)] = \text{LVB}(k); \quad k = 1, \text{KR}$$

At the same time the current length of MVBL is recorded. This is called KURPA in the program and it represents the length of the equations in CGR.

Next, the element contributions $A(k,i)$, $QT(k)$, related to variables $\text{LVB}(k)$, $\text{LVB}(i)$, will be added to the coefficient matrix CGR and the right hand-side "constant vector" RHS. The addresses contained in LDEST ensure the correct assembling of the coefficients:

$$\begin{aligned} \text{CGR}[\text{LDEST}(k), \text{LDEST}(i)] &= \text{CGR}[\text{LDEST}(k), \text{LDEST}(i)] + A(k,i) \\ \text{RHS}[\text{LDEST}(k)] &= \text{RHS}[\text{LDEST}(k)] + QT(k) \end{aligned}$$

k and $i = 1, \text{KR}$.

Figure 4-6 shows the element contributions to CGR each time an element has been processed. Once the element coefficients are assembled, the $\text{LDEST}(k)$, $k = 1, \text{KR}$ are decoded, searching for last appearances. Suppose that for a given k the variable $\text{LVB}(k)$ has to be eliminated and its equation is in the $\text{LDEST}(k) = n$ -th row and column of CGR. The process contributions to the remaining coefficients of CGR are given by

$$\text{CGR}(1,j) = \text{CGR}(1,j) - \text{CGR}(n,j) \cdot \frac{\text{CGR}(1,n)}{\text{CGR}(n,n)}$$

$$\text{RHS}(1) = \text{RHS}(1) - \text{RHS}(n) \cdot \frac{\text{CGR}(1,n)}{\text{CGR}(n,n)}$$

$1, j = 1, (n-1)$ and $1, j = (n+1)$, KURPA (= length of the equations).

If, for a given variable $\text{LVB}(k) = \text{MVBL}(n)$ there is a fixed head $H [\text{MVBL}(n)]$ boundary condition, we use

$$\begin{aligned} \text{CGR}(1,j) &= \text{CGR}(1,j) : \text{no process contributions!} \\ \text{RHS}(1) &= \text{RHS}(1) - H [\text{MVBL}(n)] \cdot \text{CGR}(1,n) \end{aligned}$$

because these formulae allow us to calculate the nodal discharge (or infiltration) $QT[\text{MVBL}(n)]$ directly on back-substitution. The matrixes shown in Figure 4-6 give a good idea about the frontal elimination procedure as opposed to the band algorithm. Note how pure process contributions fill up the matrix CGR during elimination of the equations!

For each eliminated equation the following record is written on a direct access file:

MVBL(n), n, RHS(n), KURPA, (CGR(n,i), i = 1, KURPA)

and finally, the n-th row and column of CGR will be filled up with zeros.

In the **back-substitution phase** the eliminated equations are read in reverse order, and the calculated heads $H[MVBL(n)]$ are placed in position n in a "running variable" vector HH. At any time, therefore, the composition of this vector reflects the composition of MVBL (= currently active variables) and of CGR at the corresponding time during the elimination; thus in back-substituting we can take a direct scalar product of the coefficients CGR(n,i) into the running variable vector HH(i):

$$H[MVBL(n)] \rightarrow HH(n) = [RHS(n) - \sum_i CGR(n,i) * HH(i)] / CGR(n,n)$$

with $i = 1, (n-1)$ and $i = (n+1)$, KURPA. If $H[MVBL(n)] = HH(n)$ is a known fixed head, we calculate directly the nodal discharge (or infiltration) by

$$QT[MVBL(n)] = \sum_i CGR(n,i) * HH(i) - RHS(n)$$

with $i = 1, KURPA$.

5. CODE DESCRIPTION OF FEM 301

FEM 301 simulates steady state, three-dimensional ground-water flow in heterogeneous and/or anisotropic geologic media, using a finite element method with frontal solution technique. The simulated process considered here is described in chapter 2, and may be expressed by:

$$\text{div } (-\bar{K} \cdot \vec{\text{grad}} h) + Q = 0$$

where h = hydraulic head [m]

$\bar{K}$ = volume conductivity tensor [m/s]

Q = volume source/sink term [m³/s · m³]

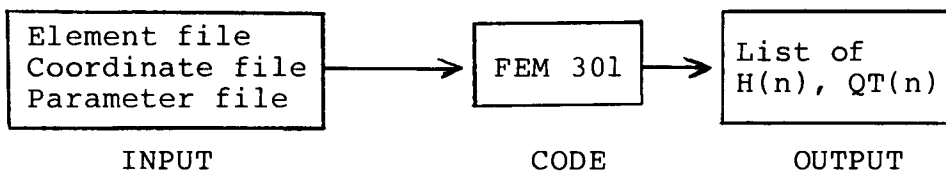
The finite element formulation of the flow equation, described in chapter 3, yields a set of simultaneous linear equations of the form

$$\begin{bmatrix} M_{mn} \end{bmatrix} \{h^n\} = \{QT_m\}$$

which is solved by the frontal method described in chapter 4, either for h^n (if QT_m is imposed), or for QT_m (if h^n is imposed).

FEM 301 does not incorporate any pre-processing or post-processing programs (these depend on the user's needs and on the available hardware); it is an independent bloc using simple input files (element topology, nodal coordinates, element permeabilities and boundary conditions), and producing a simple output file (nodal heads and nodal discharges or infiltrations). The code is written in FORTRAN 77, but it contains some features which are typical to the VAX 780 and VAX 750 environment (such as DO WHILE () loops, for example). FEM 301 is supposed to be run on machines with a virtual memory.

The aim of the present chapter is to describe the input files, the output file and the FEM 301 code itself:



5.1 Input files

The data necessary for defining the problem to be solved are:

- the geometry of the model, i.e. the element topology (list of the nodal points¹ which compose an element), and the global coordinates x^1 (= x, y, z) of the nodal points.

- the element properties; i.e. the permeability values attributed to each element.
- the boundary conditions which may be either nodal heads [m] and nodal discharges [m^3/s], or distributed source/sink intensities, such as infiltration rates over an element face [$\text{m}^3/\text{s} \cdot \text{m}^2$] or, less likely, some kind of source intensity distributed in the whole volume of an element [$\text{m}^3/\text{s} \cdot \text{m}^3$].

The above mentioned data are stored in three input files:

- a. "element file" (element topology)
- b. "coordinate file" (global coordinates for each nodal point)
- c. "parameter file" (element properties and boundary conditions)

As reading of the files is list-directed, the records contained in the input files, need not be entered in a fixed format.

The structure of the element file is illustrated by Figure 5-1. Data transmission begins with the record following immediately the keyword **"ELEMENTS:"**, so comments may be introduced, at any length, in the lines preceding the keyword. Each record is composed of the following integer variables:

LM, NQ, NP, KR, NAR, (LVB(k), k = 1, KR)

LM = element number

NQ = number indicating to which infiltration class belongs the given element

NP = number indicating to which permeability class belongs the given element

KR = number of nodal points in the element

NAR = number of element sides or element edges

LVB(k) = node numbers ("variable names") of the KR nodal points

The order in which the node numbers are listed in LVB is very important, and this order is given for each element by Figure 5-2. Note that enumeration of the nodal points starts always at a "coin" and obeys the "corkscrew principle": turn left and move upwards, or turn right and move downwards.

The infiltration classes NQ and the permeability classes NP reflect the qualitative hydrogeologic structure in the model. That is they indicate which elements belong to a given aquifer or to a given aquitard, to a given lithologic facies or to a given fault zone, etc. The actual numerical values used for the simulation will be defined for each infiltration class and permeability class in the "parameter file". Generally, the number of the permeability and in-

COMMENTAIRES:

Fractured bloc with 1-D, 2-D and 3-D elements.

(This is the "element file").

ELEMENTS:

1	1 1 27 12	1	6	11	12	13	8	3	2	7
		36	41	46	47	48	43	38	37	42
		71	76	81	82	83	78	73	72	77
2	1 1 27 12	3	8	13	14	15	10	5	4	9
		38	43	48	49	50	45	40	39	44
		73	78	83	84	85	80	75	74	79
3	1 1 27 12	11	16	21	22	23	18	13	12	17
		46	51	56	57	58	53	48	47	52
		81	86	91	92	93	88	83	82	87
4	1 1 27 12	13	18	23	24	25	20	15	14	19
		48	53	58	59	60	55	50	49	54
		83	88	93	94	95	90	85	84	89
5	1 1 27 12	21	26	31	32	33	28	23	22	27
		56	61	66	67	68	63	58	57	62
		91	96	101	102	103	98	93	92	97
6	1 1 27 12	23	28	33	34	35	30	25	24	29
		58	63	68	69	70	65	60	59	64
		93	98	103	104	105	100	95	94	99
-07	3 3 09 4	071	072	073	108	143	142	141	106	107
7	1 1 27 12	71	76	81	82	83	78	73	72	77
		106	111	116	117	118	113	108	107	112
		141	146	151	152	153	148	143	142	147
207	3 3 09 4	081	082	083	118	153	152	151	116	117
307	3 3 09 4	073	078	083	118	153	148	143	108	113
-08	3 3 09 4	073	074	075	110	145	144	143	108	109
8	1 1 27 12	73	78	83	84	85	80	75	74	79
		108	113	118	119	120	115	110	109	114
		143	148	153	154	155	150	145	144	149
208	3 3 09 4	083	084	085	120	155	154	153	118	119
9	1 1 27 12	81	86	91	92	93	88	83	82	87
		116	121	126	127	128	123	118	117	122
		151	156	161	162	163	158	153	152	157
209	3 3 09 4	091	092	093	128	163	162	161	126	127
309	3 3 09 4	083	088	093	128	163	158	153	118	123
10	1 1 27 12	83	88	93	94	95	90	85	84	89
		118	123	128	129	130	125	120	119	124
		153	158	163	164	165	160	155	154	159
210	3 3 09 4	093	094	095	130	165	164	163	128	129
11	1 1 27 12	91	96	101	102	103	98	93	92	97
		126	131	136	137	138	133	128	127	132
		161	166	171	172	173	168	163	162	167
211	3 3 09 4	093	098	103	138	173	168	163	128	133
12	1 1 27 12	93	98	103	104	105	100	95	94	99
		128	133	138	139	140	135	130	129	134
		163	168	173	174	175	170	165	164	169
-13	3 3 09 4	141	142	143	178	213	212	211	176	177
13	1 1 27 12	141	146	151	152	153	148	143	142	147
		176	181	186	187	188	183	178	177	182
		211	216	221	222	223	218	213	212	217
113	2 2 09 4	211	216	221	222	223	218	213	212	217
213	3 3 09 4	151	152	153	188	223	222	221	186	187
313	3 3 09 4	143	148	153	188	223	218	213	178	183
413	4 4 03 1	213	178	143						
513	4 4 03 1	143	148	153						
613	4 4 03 1	153	188	223						
-14	3 3 09 4	143	144	145	180	215	214	213	178	179
14	1 1 27 12	143	148	153	154	155	150	145	144	149
		178	183	188	189	190	185	180	179	184
		213	218	223	224	225	220	215	214	219
114	2 2 09 4	213	218	223	224	225	220	215	214	219
214	3 3 09 4	153	154	155	190	225	224	223	188	189
15	1 1 27 12	151	156	161	162	163	158	153	152	157
		186	191	196	197	198	193	188	187	192
		221	226	231	232	233	228	223	222	227
115	2 2 09 4	221	226	231	232	233	228	223	222	227
215	3 3 09 4	161	162	163	198	233	232	231	196	197
315	3 3 09 4	153	158	163	198	233	228	223	188	193
415	4 4 03 1	153	158	163						
515	4 4 03 1	163	198	233						
16	1 1 27 12	153	158	163	164	165	160	155	154	159
		188	193	198	199	200	195	190	189	194
		223	228	233	234	235	230	225	224	229
116	2 2 09 4	223	228	233	234	235	230	225	224	229
216	3 3 09 4	163	164	165	200	235	234	233	198	199
17	1 1 27 12	161	166	171	172	173	168	163	162	167
		196	201	206	207	208	203	198	197	202
		231	236	241	242	243	238	233	232	237
117	2 2 09 4	231	236	241	242	243	238	233	232	237
217	3 3 09 4	163	168	173	208	243	238	233	198	203
417	4 4 03 1	163	168	173						
18	1 1 27 12	163	168	173	174	175	170	165	164	169
		198	203	208	209	210	205	200	199	204
		233	238	243	244	245	240	235	234	239
118	2 2 09 4	233	238	243	244	245	240	235	234	239

Figure 5-1: Structure of the Element Input File

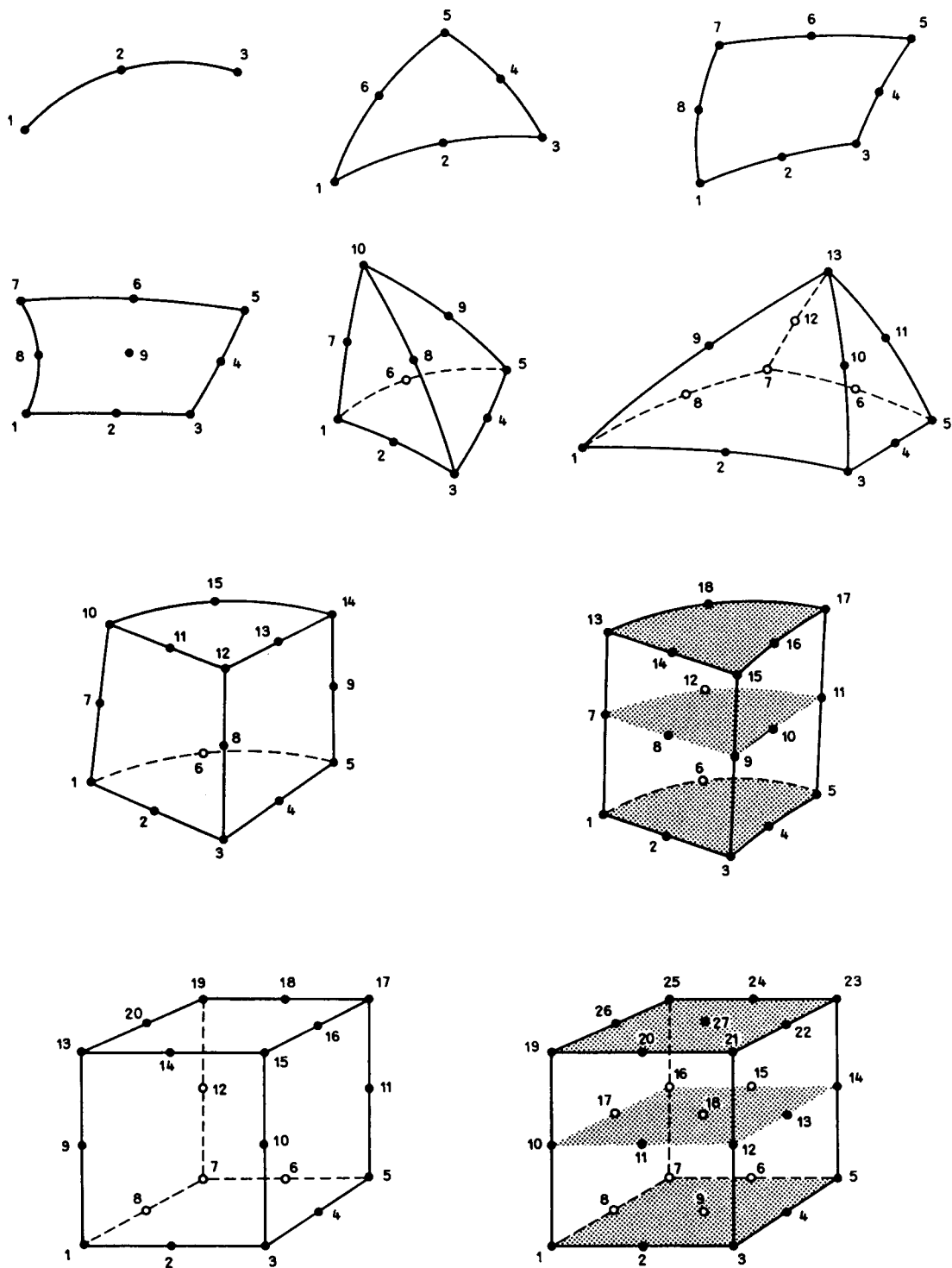


Figure 5-2 Ordering of Node Numbers

filtration classes (a few tens or a few hundreds) will be much less than the number of elements (several thousands), thus by changing a few numerical values we can change the element permeabilities in the whole model. The total number of elements is automatically recorded while reading the file.

The structure of the **coordinate** file, illustrated by Figure 5-3, is more simple. Data input begins when the **keyword** "COORDONNEES:" is encountered. The first record following the keyword contains the scale factors ECHX, ECHY, ECHZ which transform the x, y, z coordinate units into meters. If x and y are given in kilometers and z in meters, the scale factors will be ECHX = 1000, ECHY = 1000 and ECHZ = 1.

Each record following the scale factors is of the form:

NOD, X(NOD), Y(NOD), Z(NOD)

where NOD = node number, and X, Y, Z are the global coordinates of NOD.

The structure of the **parameter** file is shown by Figure 5-4. This file contains three data groups, each of them beginning with a different keyword, and ending with a zero followed by a slash: "0/".

- a. The permeability values [m/s] have to be defined for each permeability class occurring in the element file. The records following the keyword "PERMEABILITES:" may contain 1 or 6 permeability values, each record being terminated by a slash "/":

NP K1/

NP K1, K2, K3, K4, K5, K6/

where NP is the number indicating the permeability class. In the first case, the permeability is considered as isotropic; i.e. there is only one value for the NP-th class. In the second case, the 6 values represent the upper triangular part of a symmetric permeability tensor for 3-D elements, column by column:

$$K = \begin{bmatrix} K1 & K2 & K4 \\ & K3 & K5 \\ \text{Sym} & & K6 \end{bmatrix}$$

- b. The infiltration rates (given in $[m^3/s \cdot m^2]$ for a 2-D element face) have to be defined for each infiltration class occurring in the element file. The records following the keyword "ALIMENTATIONS DISTRIBUEES:" are of the form

NQ QD

where NQ = number indicating the infiltration class
QD = value of the infiltration rate.

FIGURE 5-3: STRUCTURE OF THE COORDINATE INPUT FILE

COMMENTAIRES:

Fractured bloc with 1-D, 2-D and 3-D elements.

(This is the "coordinate file").

COORDONNEES:

250.0	250.0	50.0	/ECHX,ECHY,ECHZ
1	1.0	1.0	1.0
2	1.0	2.0	1.0
3	1.0	3.0	1.0
4	1.0	4.0	1.0
5	1.0	5.0	1.0
6	2.0	1.0	1.0
7	2.0	2.0	1.0
8	2.0	3.0	1.0
9	2.0	4.0	1.0
10	2.0	5.0	1.0
11	3.0	1.0	1.0
12	3.0	2.0	1.0
13	3.0	3.0	1.0
14	3.0	4.0	1.0
15	3.0	5.0	1.0
16	4.0	1.0	1.0
17	4.0	2.0	1.0
18	4.0	3.0	1.0
19	4.0	4.0	1.0
20	4.0	5.0	1.0
21	5.0	1.0	1.0
22	5.0	2.0	1.0
23	5.0	3.0	1.0
24	5.0	4.0	1.0
25	5.0	5.0	1.0
26	6.0	1.0	1.0
27	6.0	2.0	1.0
28	6.0	3.0	1.0
29	6.0	4.0	1.0
30	6.0	5.0	1.0
31	7.0	1.0	1.0
32	7.0	2.0	1.0
33	7.0	3.0	1.0
34	7.0	4.0	1.0
35	7.0	5.0	1.0
36	1.0	1.0	2.0
37	1.0	2.0	2.0
38	1.0	3.0	2.0
39	1.0	4.0	2.0
40	1.0	5.0	2.0
41	2.0	1.0	2.0
42	2.0	2.0	2.0
43	2.0	3.0	2.0
44	2.0	4.0	2.0
45	2.0	5.0	2.0
46	3.0	1.0	2.0
47	3.0	2.0	2.0
48	3.0	3.0	2.0
49	3.0	4.0	2.0
50	3.0	5.0	2.0
51	4.0	1.0	2.0
52	4.0	2.0	2.0
53	4.0	3.0	2.0
54	4.0	4.0	2.0
55	4.0	5.0	2.0
56	5.0	1.0	2.0
57	5.0	2.0	2.0
58	5.0	3.0	2.0
59	5.0	4.0	2.0
60	5.0	5.0	2.0
61	6.0	1.0	2.0
62	6.0	2.0	2.0
63	6.0	3.0	2.0
64	6.0	4.0	2.0
65	6.0	5.0	2.0
66	7.0	1.0	2.0
67	7.0	2.0	2.0
68	7.0	3.0	2.0
69	7.0	4.0	2.0
70	7.0	5.0	2.0
71	1.0	1.0	3.0
72	1.0	2.0	3.0
73	1.0	3.0	3.0
74	1.0	4.0	3.0
75	1.0	5.0	3.0
76	2.0	1.0	3.0
77	2.0	2.0	3.0
78	2.0	3.0	3.0
79	2.0	4.0	3.0
80	2.0	5.0	3.0
81	3.0	1.0	3.0
82	3.0	2.0	3.0
83	3.0	3.0	3.0
84	3.0	4.0	3.0
85	3.0	5.0	3.0
86	4.0	1.0	3.0
87	4.0	2.0	3.0
88	4.0	3.0	3.0
89	4.0	4.0	3.0
90	4.0	5.0	3.0
91	5.0	1.0	3.0
92	5.0	2.0	3.0
93	5.0	3.0	3.0
94	5.0	4.0	3.0
95	5.0	5.0	3.0
96	6.0	1.0	3.0
97	6.0	2.0	3.0
98	6.0	3.0	3.0
99	6.0	4.0	3.0
100	6.0	5.0	3.0
101	7.0	1.0	3.0
102	7.0	2.0	3.0
103	7.0	3.0	3.0
104	7.0	4.0	3.0
105	7.0	5.0	3.0
106	1.0	1.0	4.0
107	1.0	2.0	4.0
108	1.0	3.0	4.0
109	1.0	4.0	4.0
110	1.0	5.0	4.0
111	2.0	1.0	4.0
112	2.0	2.0	4.0
113	2.0	3.0	4.0
114	2.0	4.0	4.0
115	2.0	5.0	4.0
116	3.0	1.0	4.0
117	3.0	2.0	4.0
118	3.0	3.0	4.0
119	3.0	4.0	4.0
120	3.0	5.0	4.0
121	4.0	1.0	4.0
122	4.0	2.0	4.0
123	4.0	3.0	4.0
124	4.0	4.0	4.0
125	4.0	5.0	4.0
126	5.0	1.0	4.0
127	5.0	2.0	4.0
128	5.0	3.0	4.0
129	5.0	4.0	4.0
130	5.0	5.0	4.0
131	6.0	1.0	4.0
132	6.0	2.0	4.0
133	6.0	3.0	4.0
134	6.0	4.0	4.0
135	6.0	5.0	4.0
136	7.0	1.0	4.0
137	7.0	2.0	4.0
138	7.0	3.0	4.0
139	7.0	4.0	4.0
140	7.0	5.0	4.0
141	1.0	1.0	5.0
142	1.0	2.0	5.0
143	1.0	3.0	5.0
144	1.0	4.0	5.0
145	1.0	5.0	5.0
146	2.0	1.0	5.0
147	2.0	2.0	5.0
148	2.0	3.0	5.0
149	2.0	4.0	5.0
150	2.0	5.0	5.0
151	3.0	1.0	5.0
152	3.0	2.0	5.0
153	3.0	3.0	5.0
154	3.0	4.0	5.0
155	3.0	5.0	5.0
156	4.0	1.0	5.0
157	4.0	2.0	5.0
158	4.0	3.0	5.0
159	4.0	4.0	5.0
160	4.0	5.0	5.0
161	5.0	1.0	5.0
162	5.0	2.0	5.0
163	5.0	3.0	5.0
164	5.0	4.0	5.0
165	5.0	5.0	5.0
166	6.0	1.0	5.0
167	6.0	2.0	5.0
168	6.0	3.0	5.0
169	6.0	4.0	5.0
170	6.0	5.0	5.0
171	7.0	1.0	5.0
172	7.0	2.0	5.0
173	7.0	3.0	5.0
174	7.0	4.0	5.0
175	7.0	5.0	5.0
176	1.0	1.0	6.0
177	1.0	2.0	6.0
178	1.0	3.0	6.0
179	1.0	4.0	6.0
180	1.0	5.0	6.0
181	2.0	1.0	6.0
182	2.0	2.0	6.0
183	2.0	3.0	6.0
184	2.0	4.0	6.0
185	2.0	5.0	6.0
186	3.0	1.0	6.0
187	3.0	2.0	6.0
188	3.0	3.0	6.0
189	3.0	4.0	6.0
190	3.0	5.0	6.0
191	4.0	1.0	6.0
192	4.0	2.0	6.0
193	4.0	3.0	6.0
194	4.0	4.0	6.0
195	4.0	5.0	6.0
196	5.0	1.0	6.0
197	5.0	2.0	6.0
198	5.0	3.0	6.0
199	5.0	4.0	6.0
200	5.0	5.0	6.0
201	6.0	1.0	6.0
202	6.0	2.0	6.0
203	6.0	3.0	6.0
204	6.0	4.0	6.0
205	6.0	5.0	6.0
206	7.0	1.0	6.0
207	7.0	2.0	6.0
208	7.0	3.0	6.0
209	7.0	4.0	6.0
210	7.0	5.0	6.0
211	1.0	1.0	7.0
212	1.0	2.0	7.0
213	1.0	3.0	7.0
214	1.0	4.0	7.0
215	1.0	5.0	7.0
216	2.0	1.0	7.0
217	2.0	2.0	7.0
218	2.0	3.0	7.0
219	2.0	4.0	7.0
220	2.0	5.0	7.0
221	3.0	1.0	7.0
222	3.0	2.0	7.0
223	3.0	3.0	7.0
224	3.0	4.0	7.0
225	3.0	5.0	7.0
226	4.0	1.0	7.0
227	4.0	2.0	7.0
228	4.0	3.0	7.0
229	4.0	4.0	7.0
230	4.0	5.0	7.0
231	5.0	1.0	7.0
232	5.0	2.0	7.0
233	5.0	3.0	7.0
234	5.0	4.0	7.0
235	5.0	5.0	7.0
236	6.0	1.0	7.0
237	6.0	2.0	7.0
238	6.0	3.0	7.0
239	6.0	4.0	7.0
240	6.0	5.0	7.0
241	7.0	1.0	7.0
242	7.0	2.0	7.0
243	7.0	3.0	7.0
244	7.0	4.0	7.0
245	7.0	5.0	7.0

COMMENTAIRES:

Fractured bloc with 1-D, 2-D and 3-D elements.
(This is the "parameter file").

PERMEABILITES:

01 5.0E-6/
02 5.0E-6/
03 5.0E-3/
04 1.0E+1/
00/

ALIMENTATIONS DISTRIBUEES:

01 0.0E+0
02 2.0E-8
03 0.0E+0
04 0.0E+0
00/

CONDITIONS NODALES:

213 -1 0.0
00/

Figure 5-4: Structure of the Parameter Input File.

- c. The nodal boundary conditions are preceded by the keyword "CONDITIONS NODALES:". Each record has the general form of:

NOD ID(NOD) VAL(NOD)

where: NOD = node number for which the boundary condition is given

ID(NOD) = 1 for fixed nodal discharge (or nodal infiltration)

= -1 for fixed nodal head

VAL(NOD) = value of the imposed nodal discharge [m^3/s] or imposed nodal head [m], depending on ID(NOD).

All three input files have a simple data structure, so they are easy to update. For example, when a large element is replaced by several smaller ones (in Figures 4-3 and 4-7), the record of the large element is simply deleted from the element file, and the new records are introduced at the same place. The elements need not be renumbered; only the order of the element records is important.

5.2 Output file

The general structure of the output file is shown by Figure 5-5. Output begins with the **control parameters** immediately after the symbolic elimination procedure has been achieved. These control parameters are:

MNNIC : lowest node number

MXNIC : highest node number

MXNQ : number of infiltration classes in the element file

MXNP : number of permeability classes in the element file

MXLM : total number of elements

NDACT : total number of active variables (it is not necessarily MXNIC - MNNIC + 1 !!)

MXPA : maximum frontwidth encountered during the symbolic elimination

If MXPA is greater than some threshold value, the numerical calculations are not carried out and an error message is issued.

The three data groups of the parameter file are then written on the output file in order to check the correct reading in of the values (permeabilities, infiltration rates and nodal boundary conditions).

The calculated results are presented, for each nodal point, in the following form:

NOD ID(NOD) H(NOD) QT(NOD)

FIGURE 5-5: STRUCTURE OF THE OUTPUT FILE

```
*****
* "FEM301" - MODELE A ELEMENTS FINIS *
* ECOULEMENT PERMANENT - TRIDI *
* CENTRE D'HYDROGEOLOGIE (UNIVERSITE DE NEUCHATEL) *
*****
```

FICHIERS DES DONNEES:

```
***** VERT:[KIRALY]FRAC.ELM;4
      VERT:[KIRALY]FRAC.COR;3
      VERT:[KIRALY]FRAC.PAR;3
```

FICHER DE SORTIE:

```
***** VERT:[KIRALY]FRAC.RES;2
```

PARAMETRES DE CONTROLE:

```
*****
```

```
MNNIC= 1      MXNIC= 245      MXNQ= 4      MXNP= 4
MXLM= 48      NDACT= 245      MXPA= 57
```

VALEURS DES CLASSES DE PERM/PORO

```
*****
```

```
1      0.50000E-05
2      0.50000E-05
3      0.50000E-02
4      0.10000E+02
```

ALIMENTATIONS DISTRIBUEES

```
*****
```

```
1      0.0000000E+00
2      0.2000000E-07
3      0.0000000E+00
4      0.0000000E+00
```

CONDITIONS NODALES

```
*****
```

```
1      213 -1      0.0000000E+00
```

```
FICHER DES EQUATIONS: VERT:[KIRALY]FRAC.EQU;2
```

!RESULTATS CALCULES!

```
*****
```

1 0 1.10 0.000E+0	2 0 0.90 0.000E+0	3 0 0.61 0.000E+0
4 0 0.90 0.000E+0	5 0 1.10 0.000E+0	6 0 1.51 0.000E+0
7 0 1.31 0.000E+0	8 0 0.97 0.000E+0	9 0 1.31 0.000E+0
10 0 1.51 0.000E+0	11 0 1.91 0.000E+0	12 0 1.72 0.000E+0
13 0 1.43 0.000E+0	14 0 1.72 0.000E+0	15 0 1.91 0.000E+0
16 0 2.29 0.000E+0	17 0 2.09 0.000E+0	18 0 1.76 0.000E+0
19 0 2.09 0.000E+0	20 0 2.29 0.000E+0	21 0 2.55 0.000E+0
22 0 2.34 0.000E+0	23 0 2.03 0.000E+0	24 0 2.34 0.000E+0
25 0 2.55 0.000E+0	26 0 2.90 0.000E+0	27 0 2.64 0.000E+0
28 0 2.19 0.000E+0	29 0 2.64 0.000E+0	30 0 2.90 0.000E+0
31 0 3.07 0.000E+0	32 0 2.77 0.000E+0	33 0 2.26 0.000E+0
34 0 2.77 0.000E+0	35 0 3.07 0.000E+0	36 0 1.07 0.000E+0
37 0 0.86 0.000E+0	38 0 0.56 0.000E+0	39 0 0.86 0.000E+0
40 0 1.07 0.000E+0	41 0 1.52 0.000E+0	42 0 1.32 0.000E+0
43 0 0.93 0.000E+0	44 0 1.32 0.000E+0	45 0 1.52 0.000E+0
46 0 1.90 0.000E+0	47 0 1.72 0.000E+0	48 0 1.41 0.000E+0
49 0 1.72 0.000E+0	50 0 1.90 0.000E+0	51 0 2.30 0.000E+0
52 0 2.11 0.000E+0	53 0 1.73 0.000E+0	54 0 2.11 0.000E+0
55 0 2.30 0.000E+0	56 0 2.54 0.000E+0	57 0 2.34 0.000E+0
58 0 2.00 0.000E+0	59 0 2.34 0.000E+0	60 0 2.54 0.000E+0
61 0 2.91 0.000E+0	62 0 2.65 0.000E+0	63 0 2.16 0.000E+0
64 0 2.65 0.000E+0	65 0 2.91 0.000E+0	66 0 3.08 0.000E+0
67 0 2.79 0.000E+0	68 0 2.22 0.000E+0	69 0 2.79 0.000E+0
70 0 3.08 0.000E+0	71 0 0.96 0.000E+0	72 0 0.75 0.000E+0
73 0 0.41 0.000E+0	74 0 0.75 0.000E+0	75 0 0.96 0.000E+0
76 0 1.54 0.000E+0	77 0 1.38 0.000E+0	78 0 0.83 0.000E+0

Figure 5-5 (cont.)

79	0	1.38	0.000E+0	80	0	1.54	0.000E+0	81	0	1.88	0.000E+0
82	0	1.70	0.000E+0	83	0	1.33	0.000E+0	84	0	1.70	0.000E+0
85	0	1.88	0.000E+0	86	0	2.31	0.000E+0	87	0	2.15	0.000E+0
88	0	1.63	0.000E+0	89	0	2.15	0.000E+0	90	0	2.31	0.000E+0
91	0	2.52	0.000E+0	92	0	2.33	0.000E+0	93	0	1.93	0.000E+0
94	0	2.33	0.000E+0	95	0	2.52	0.000E+0	96	0	2.92	0.000E+0
97	0	2.70	0.000E+0	98	0	2.02	0.000E+0	99	0	2.70	0.000E+0
100	0	2.92	0.000E+0	101	0	3.09	0.000E+0	102	0	2.84	0.000E+0
103	0	2.08	0.000E+0	104	0	2.84	0.000E+0	105	0	3.09	0.000E+0
106	0	0.95	0.000E+0	107	0	0.75	0.000E+0	108	0	0.37	0.000E+0
109	0	0.75	0.000E+0	110	0	0.95	0.000E+0	111	0	1.60	0.000E+0
112	0	1.43	0.000E+0	113	0	0.80	0.000E+0	114	0	1.43	0.000E+0
115	0	1.60	0.000E+0	116	0	1.88	0.000E+0	117	0	1.71	0.000E+0
118	0	1.31	0.000E+0	119	0	1.71	0.000E+0	120	0	1.88	0.000E+0
121	0	2.38	0.000E+0	122	0	2.21	0.000E+0	123	0	1.61	0.000E+0
124	0	2.21	0.000E+0	125	0	2.38	0.000E+0	126	0	2.52	0.000E+0
127	0	2.34	0.000E+0	128	0	1.91	0.000E+0	129	0	2.34	0.000E+0
130	0	2.52	0.000E+0	131	0	2.99	0.000E+0	132	0	2.76	0.000E+0
133	0	2.00	0.000E+0	134	0	2.76	0.000E+0	135	0	2.99	0.000E+0
136	0	3.17	0.000E+0	137	0	2.90	0.000E+0	138	0	2.05	0.000E+0
139	0	2.90	0.000E+0	140	0	3.17	0.000E+0	141	0	0.93	0.000E+0
142	0	0.75	0.000E+0	143	0	0.25	0.000E+0	144	0	0.75	0.000E+0
145	0	0.93	0.000E+0	146	0	1.69	0.000E+0	147	0	1.52	0.000E+0
148	0	0.78	0.000E+0	149	0	1.52	0.000E+0	150	0	1.69	0.000E+0
151	0	1.87	0.000E+0	152	0	1.72	0.000E+0	153	0	1.25	0.000E+0
154	0	1.72	0.000E+0	155	0	1.87	0.000E+0	156	0	2.47	0.000E+0
157	0	2.31	0.000E+0	158	0	1.58	0.000E+0	159	0	2.31	0.000E+0
160	0	2.47	0.000E+0	161	0	2.51	0.000E+0	162	0	2.35	0.000E+0
163	0	1.85	0.000E+0	164	0	2.35	0.000E+0	165	0	2.51	0.000E+0
166	0	3.09	0.000E+0	167	0	2.87	0.000E+0	168	0	1.96	0.000E+0
169	0	2.87	0.000E+0	170	0	3.09	0.000E+0	171	0	3.26	0.000E+0
172	0	3.00	0.000E+0	173	0	2.00	0.000E+0	174	0	3.00	0.000E+0
175	0	3.26	0.000E+0	176	0	0.92	0.000E+0	177	0	0.76	0.000E+0
178	0	0.13	0.000E+0	179	0	0.76	0.000E+0	180	0	0.92	0.000E+0
181	0	1.83	0.000E+0	182	0	1.66	0.000E+0	183	0	0.83	0.000E+0
184	0	1.66	0.000E+0	185	0	1.83	0.000E+0	186	0	1.89	0.000E+0
187	0	1.74	0.000E+0	188	0	1.26	0.000E+0	189	0	1.74	0.000E+0
190	0	1.89	0.000E+0	191	0	2.61	0.000E+0	192	0	2.45	0.000E+0
193	0	1.63	0.000E+0	194	0	2.45	0.000E+0	195	0	2.61	0.000E+0
196	0	2.54	0.000E+0	197	0	2.37	0.000E+0	198	0	1.86	0.000E+0
199	0	2.37	0.000E+0	200	0	2.54	0.000E+0	201	0	3.23	0.000E+0
202	0	3.01	0.000E+0	203	0	2.01	0.000E+0	204	0	3.01	0.000E+0
205	0	3.23	0.000E+0	206	0	3.39	0.000E+0	207	0	3.12	0.000E+0
208	0	2.05	0.000E+0	209	0	3.12	0.000E+0	210	0	3.39	0.000E+0
211	0	0.89	0.139E-3	212	0	0.79	0.556E-3	213	-1	0.00	-0.300E-1
214	0	0.79	0.556E-3	215	0	0.89	0.139E-3	216	0	1.99	0.556E-3
217	0	1.89	0.222E-2	218	0	0.87	0.111E-2	219	0	1.89	0.222E-2
220	0	1.99	0.556E-3	221	0	1.91	0.278E-3	222	0	1.75	0.111E-2
223	0	1.26	0.556E-3	224	0	1.75	0.111E-2	225	0	1.91	0.278E-3
226	0	2.77	0.556E-3	227	0	2.68	0.222E-2	228	0	1.66	0.111E-2
229	0	2.68	0.222E-2	230	0	2.77	0.556E-3	231	0	2.55	0.278E-3
232	0	2.39	0.111E-2	233	0	1.86	0.556E-3	234	0	2.39	0.111E-2
235	0	2.55	0.278E-3	236	0	3.39	0.556E-3	237	0	3.22	0.222E-2
238	0	2.05	0.111E-2	239	0	3.22	0.222E-2	240	0	3.39	0.556E-3
241	0	3.51	0.139E-3	242	0	3.27	0.556E-3	243	0	2.08	0.278E-3
244	0	3.27	0.556E-3	245	0	3.51	0.139E-3				

SOMME DES DEBITS = -0.1929880E-15

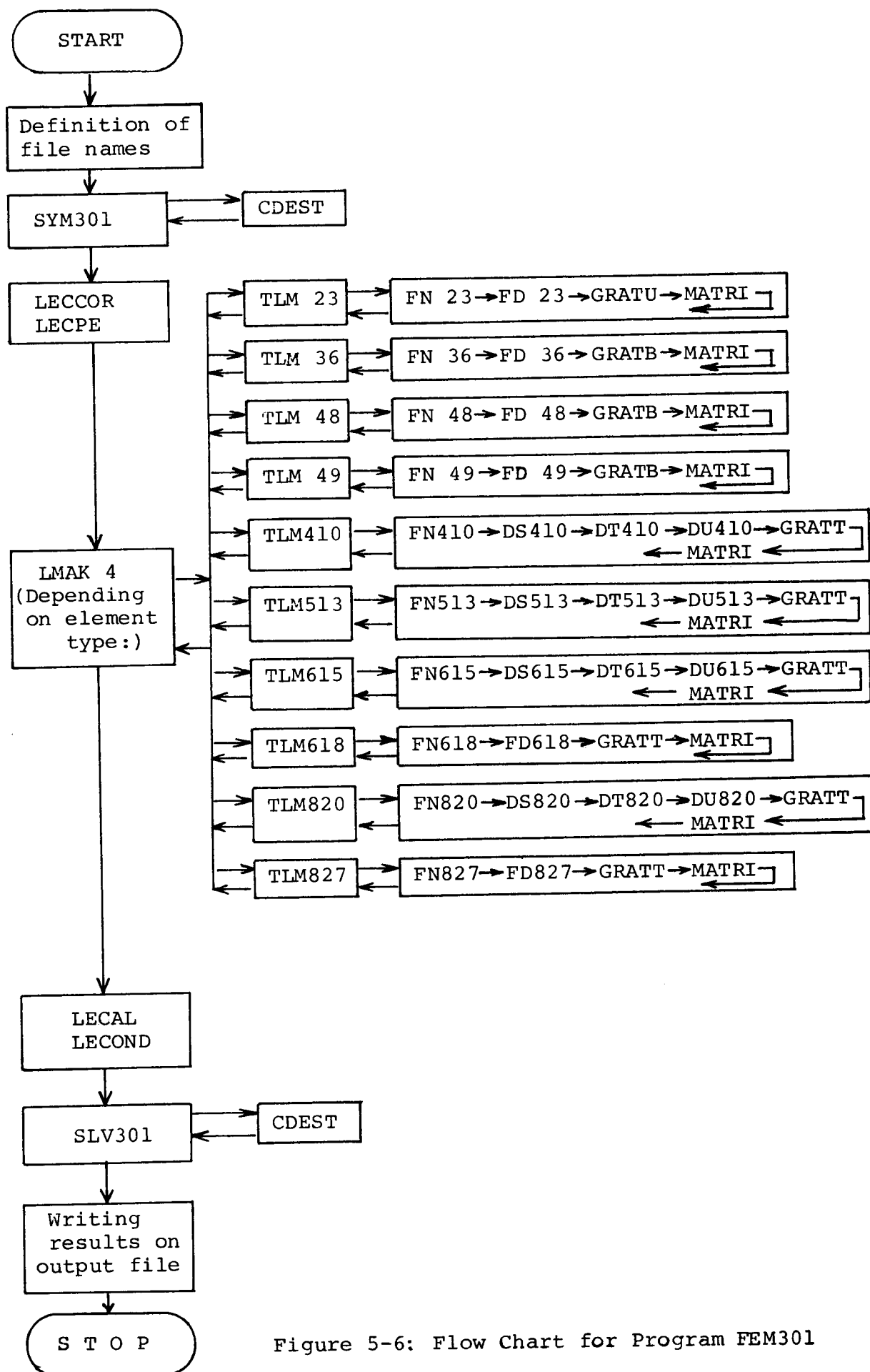


Figure 5-6: Flow Chart for Program FEM301

If $ID(NOD) = \emptyset$ or 1 then $H(NOD)$ is the calculated result; if $ID(NOD)$ is negative then $QT(NOD)$ is the calculated value at the fixed head boundary. Note that $QT(NOD)$ represents the integral of all discharges (or infiltrations) over the volume of influence of a given nodal point, and its interpretation is not always evident.

5.3 Flow chart of FEM 301

The program FEM 301 has been conceived by a "top-down" design approach producing a rather modular structure. By retaining this modularity, the program is simpler to follow and understand, which in itself aids the maintenance and the improvement of the program. The straightforward relationships between program units of the same level and between program units of different levels, are shown by the flow chart of Figure 5-6. As the principal variables are defined in Annex A2, and the listings are presented in Annex A3, we will give here only a rather brief description of the different program units.

The first version of the code was written in FORTRAN IV, and the program was run on an IBM 1130 with 16-K memory. The present version of FEM 301 is supposed to be run on a VAX 780 with virtual memory, and contains some features of FORTRAN 77, as well as some features which are typical to VAX 780 or VAX 750 environment (such as DO WHILE loops, for example). In its original form the code was developed for university research and teaching purposes (simulation of fractured and/or karstic aquifers), thus our main concern was not to make it easily portable or particularly well documented. This is, naturally, a serious drawback when the code is required to be distributed or the listings (reproduced in Annex A3) to be easily understood.

The **main program FEM 301** defines (= opens) the files, calls the first-level subroutines, and writes the control parameters as well as the calculated results onto the result file. The following files are used by FEM 301:

FIELM : element file (formatted input file).
FICOR : coordinate file (formatted input file).
FIPAR : parameter file (formatted input file).
FIDST : unformatted (binary), sequential scratch file. It contains the LVB and LDEST vectors (output of SYM 301).
FIMAT : unformatted, sequential scratch file. It contains the LVB and LDEST vectors, as well as the ALM and CLM element matrixes produced by LMAK4.

FIEQU : unformatted, direct-access file, containing the eliminated equations (output of SLV 301). This file is not deleted after the job has been achieved, thus it may be used for re-resolution purposes later on.

FIRES : formatted output file

The diagram below shows the structure of the main working area in core during symbolic elimination by SYM 301, calculation of the element matrixes by LMAK4, and solving of the equations by SLV 301:

SYM 301:	NIX (I*4)			ID(I*4)	small
TLMAK4 :	X (R*8)	Y (R*8)	Z (R*8)	ID(I*4)	vec-
SLV 301:	CGR (R*8)			H (R*8) Q(R*8)	tors

Presently, the dimensions are 200'000 for NIX, 300'000 for CGR, 99'999 for X, Y, Z, H, Q and ID.

Four of the seven **first-level subroutines** called by the main program are used only for reading the input files:

LECOR : reading in the coordinate file

LECPE : reading in the permeabilities from the parameter file

LECAL : reading in the infiltration rates from the parameter file

LECOND : reading in the nodal boundary conditions from the parameter file.

The calculations described in chapters 3 and 4 are carried out by the remaining three subroutines, which form the **three main blocks of FEM 301**:

SYM 301: achieves the symbolic elimination and generates the LDEST vectors (see section 4.2). SYM 301 reads the element file FIELM (input), and writes the LVB and LDEST vectors on file FIDST (output).

TLMAK4 : calculates the element matrixes by using numerical integration over the elements (see sections 3.3 and 3.4). Required input: global coordinates X, Y, Z; element data from file FIDST; permeability values for each permeability class (from file FIPAR). Output: the complete element data set, written on file FIMAT (see Annex A3). In these element data sets only the vectors ALM and the matrixes CLM are produced by TLMAK4. Note that the vector ALM represents the region of influence of each nodal point in a given element. This region of influence is a length for a 1-D element, a surface for a 2-D element and a volume for a 3-D element.

SLV 301: assembles, eliminates and solves the element equations by the frontal method described in section 4.2. Required input: nodal boundary conditions (from file FIPAR); distributed infiltration rates (from file FIPAR); the complete element data sets (LVB, LDEST and ALM vectors; CLM matrix) from file FIMAT. Output: the eliminated equations on file FIEQU, and the calculated nodal heads H or nodal discharges Q in core.

The **second-level subroutines** are called by SYM 301, TLMAK4 and SLV 301.

CDEST : is a small subroutine which decodes the LDEST vectors. It returns two quantities: LDES indicates the address of an equation in CGR (row or column), and NSTR shows the appearance of the corresponding variable (first, intermediate, last).

TLM ***: these subroutines contain the local coordinates of the integration points and achieve the numerical integration for the different element types. In "TLM***" the first number * indicates the number of corners, and the last two figures ** represent the number of nodal points in the corresponding element. They return the integrated values ALM and CLM for each element.

The **third-level subroutines** are called only by the TLM***.

FN*** : calculates the numerical value of the interpolation functions N_n for a given set of local coordinates (here: for the local coordinates of the integration points). The numbers "****" show the element type.

FD***
DS***
DT***
DU*** } : subroutines computing the numerical value of the derivatives $\partial_j N_n / \partial s^j$ for a given set of local coordinates s^j (here: for the integration points).

GRATU : calculates the numerical value of the gradient matrix $B_{ni} = \partial N_n / \partial x^i$ for a 1-D element in a 3-D global space (see section 3.3), by using the "local derivatives" $\partial N_n / \partial s^j$.

GRATB : calculates the numerical value of the gradient matrix $B_{ni} = \partial N_n / \partial x^i$ for a 2-D element in a 3-D global space.

GRATT : calculates the numerical value of the gradient matrix $B_{ni} = \partial N_n / \partial x^i$ for a 3-D element in a 3-D global space.

MATR4 : computes the numerical value of vector ALM and matrix CLM (= integrands in equations 3-17 and 3-18) at a given integration point. Required input: numerical value of the interpolation functions given by FN***; numerical value of the gradient matrix given by GRATU, GRATB or GRATT; permeability tensor K^{1j} for the given element. The vector ALM and the matrix CLM are returned to TLM*** (see Figure 5-6).

5.4 Command procedures

The command procedure FEM 301.COM (written in DCL command language and presented in Annex A3) allows the user to define the input and output files interactively. He also can chose to execute the job interactively or in batch mode by answering the questions appearing on the screen of the terminal.

FEM 301.COM activates another command procedure, FEM 301A.COM (see Annex A3), which assigns the user given file names to the external unit numbers defined in the main program FEM 301.FOR. If the command procedures are not used, the assignments have to be executed by the user himself.

6. PRE- AND POST-PROCESSING ROUTINES

The generation of the input files and evaluation of the output files of a large numerical model takes a considerable amount of time. In order to reduce this time, a series of pre-processing programs have been developed to assist the modeler in creating the input files. In addition, several post-processing routines have been written to aid the modeler in interpreting and graphically displaying the results. These routines are briefly described below. As most of these programs are specific to the hardware available at the Université de Neuchâtel they may not be transformable to other systems. They are presented here only for completeness. Virtually all of these programs have been designed to be used interactively; that is the user is prompted at the terminal to type the required information.

6.1 Pre-processing routines

As described in Chapter 5, three input files are required to execute program FEM 301. These are the **element file**, the **coordinate file**, and the **parameter file**. The element file defines the element type (e.g. rectangular prism, pinched triangular prism, etc), the permeability and infiltration class, the number of nodes, and the corner and midside node numbers for each element. The coordinate file defines the x, y, and z coordinates of each node (i.e. locates each node in the global, three-dimensional space). The parameter file defines the permeabilities and infiltration rates (in m/s) for each permeability and infiltration class indicated in the element file as well as specified nodal discharge rates (in m³/s) or heads along the boundaries.

Generating these input files for a large, complex two- or three-dimensional problem can be a very laborious procedure. For example, the regional model of groundwater flow in northern Switzerland contains a total of 14'768 nodes, 3'635 three-dimensional elements, 1'738 two-dimensional elements, and 258 one-dimensional elements (NTB 84-50). As a result, a suite of pre-processing routines has been developed to assist the modeler in transgressing from raw geologic and topographic data to the definition of the geometry and element structure of the conceptual model. These programs have been specifically developed for use with FEM 301.

The programs utilized in developing the requisite input data files are depicted in Figure 6-1. These programs are briefly described below. Given the relative simplicity of the parameter file, there are no explicit pre-processing routines for generating this file except the normal text editor.

ELEBIDI : 2-d file of surface element

TABLETTE : digitizing table used for x-y coordinates of
2-d surface elements

FERN : check surface elements and move nodes
if necessary

ESTIMZ : z-coordinates of surface nodes
based on digitized topography

ELETRIDI : generate element file, column file, x-y
coordinate file for all nodes of 3-d grid

ESTIMZ : z-coordinates of element layer boundaries based
on digitized structure contour maps

- . OPERZ : add/subtract thickness to a given
structural elevation
- . INSEREZ: assign constant z-elevation for a
layer (e.g. base of model)
- . CALCULZ: calculate z-coordinate of mid-side
nodes along vertical edges

UJZ : modify z-coordinate of first 3 layers

NEWFRONT : re-order element file to minimize front width

Figure 6-1: Pre-processing routines used to create element
and coordinate files

6.1.1 ELEBIDI

Program ELEBIDI assists the user in creating a two-dimensional element file. The element file consists of the node number (automatically incremented), a default infiltration class, a default permeability class, the number of nodes in the element (3, 6 or 8), and the node numbers. For each successive surface element, the user is prompted to enter the number of nodes in that element followed by the node numbers (in a counter-clockwise direction). The program is not an automatic mesh generator, but requires the user to define the surface mesh and node numbers by hand before proceeding.

6.1.2 TABLETTE

Program TABLETTE assists the user in creating a two-dimensional coordinate file of the surface mesh formed by ELEBIDI. This is accomplished by the use of a digitizing table. Moving the cursor ("mouse") successively from node to node, the user must only enter the node number. The x-y coordinates are defined automatically using the relationship between the size of the table and the scale of the map. The user can type the z-coordinate of the node at this time if a digitized water-table surface or surface topography map is not available.

6.1.3 FERN

Program FERN is a graphic routine developed to plot the finite element grid. Both vertical and horizontal sections may be plotted. The principal use of FERN is to allow for the interactive relocation of individual nodes. Once the grid is plotted on the screen, the user can move the cursor to lie on a node which is misplaced (i.e. the coordinate file is in err, the element file is in err, or the elements are too distorted, etc.). The precursor can then be moved to the proper location for the node in question. By entering a simple command, the new coordinates of this node will be written on the coordinate file. For areas of the mesh with a high density of nodes, FERN allows the user to "zoom" into this area for finer scale node movements. This program is very valuable for checking the geometry of a given mesh and rapidly modifying the coordinate file prior to using the file in a simulation.

6.1.4 ESTIMZ

Program ESTIMZ allows the nodal elevation of a particular surface to be defined based on a digitized map. The nodal values are determined by a linear piecewise interpolation scheme. This could be used for generating the z-coordinates of the nodes in the 2-dimensional surface elements from either topographic or water table maps. However, the detail in topographic maps is generally too great to warrant digitizing. Therefore, the surface elevations are normally input manually except where water table surfaces are available (normally the alluvial aquifers).

ESTIMZ is also used for defining the z-coordinate of element layer boundaries based on digitized structure contour maps. For example, structure contours on the top of an aquifer which is modeled as a discrete layer (either 2-d or 3-d) in a three-dimensional model are used to interpolate the nodal elevations for these nodes corresponding to this surface. This avoids the very tedious procedure of manually defining the z-coordinate! It should be noted that a three-dimensional coordinate file (generated by ELETRIDI) is required prior to defining the actual z-coordinate of the subsurface nodes. Program ISOVALZ is used to control digitized maps prior to use by ESTIMZ.

6.1.5 ELETRIDI

Program ELETRIDI allows for the interactive formation of a three-dimensional mesh from a two-dimensional surficial mesh. The input required for this program consists of the two-dimensional element and coordinates files generated by ELEBIDI and TABLETTE.

Each surface element defines the top of a three-dimensional column. The user must interactively specify the type of elements found in each column, from the surface to the base. In addition, the permeability and infiltration classes and the number of nodes and edges of each element must also be specified. As many columns have the same vertical element topology, these may be predefined by the user so they need not be typed each time that column appears. For example, if an 8-node rectangle exists at the surface, the first 3-d element may be a triangular prism (15 nodes), followed by two bricks (20 nodes) a 2-d rectangle (8 nodes), three more bricks (20 nodes), etc.

The output of program ELETRIDI consists of a three-dimensional element file for all elements of the mesh, a column file which defines the type of elements down each vertical column, and a three-dimensional coordinate file. The program automatically numbers the nodes down each vertical edge and properly orders the nodes for each element corresponding to the "corkscrew principle" (Figure 5-2). The coordinate file contains the x-y coordinates of all nodes. The z-coordinates must be added using programs ESTIMZ, OPERZ, INSEREZ or CALCULZ. Without program ELETRIDI, the creation of irregular three-dimensional meshes of any significant size would be extremely time consuming.

6.1.6 OPERZ

Program OPERZ allows for the addition or subtraction of a thickness to a layer of nodes. For example, if the modeler has a structure contour map for the base of a particular aquifer (corresponding to a three-dimensional layer of the model) he would first use ESTIMZ to define the elevations of all nodes on this boundary (perhaps nodes from 26,000 to 27,234). If this aquifer has a relatively uniform thickness (say 50 m), then program OPERZ simply adds 50 m to these elevations to define the elevations at the top of this aquifer nodes (22,000 to 23,234). In this way, the z-coordinates of the coordinate file can be quickly introduced. OPERZ may also be used with digitized isopach maps to add/subtract variable thicknesses.

6.1.7 INSEREZ

Program INSEREZ allows the user to insert a constant z-coordinate for any layer of a three-dimensional mesh. For example, a constant elevation of -8000 m has been used in the regional and local models described in NTB 84-50 (Kimmeier et al., 1984).

6.1.8 CALCULZ

Program CALCULZ calculates the z-coordinate of mid-side nodes which lie on vertical edges. This routine simply places the node mid way between the top and bottom of the element.

6.1.9 UJZ

Program UJZ allows the elevations of the first three layers of a three-dimensional mesh to be automatically modified. This program is utilized when the structure contour maps, isopachs, surface elevations and surface geology used to define the geometry of the model are inconsistent. For example, if one takes as given the contoured structure map for some geologic horizon and adds constant (or digitized) isopachs for each layer above this horizon when the surface (defined by a topographic map) is reached, the top layer **should** correspond to the mapped surface geology. Unfortunately this is not often the case, primarily due to the fact that the structure maps and isopachs are based on a limited data base which when contoured often leads to inconsistencies. This is especially true when contours are generated by hand or in areas of complex geology. As it is difficult for the modeler to identify the actual source of the inconsistency (i.e. is some layer too thick or thin), UJZ arbitrarily adds or subtracts the appropriate thickness to the first three layers of the model.

Once the entire three-dimensional element files and coordinate files have been generated, program FERN is again used to check the geometry of the mesh. In addition to checking each two-dimensional layer of the mesh, each vertical section is commonly plotted to verify the vertical geometry and to reposition nodes if necessary. While this is a somewhat laborious procedure, it ensures that the prescribed model geometry is exactly what the user intended.

6.1.10 NEWFRONT

Program NEWFRONT restructures the element file to minimize the required front width for the equation solver. This program is very useful to minimize the computational time, especially for large three-dimensional meshes.

6.1.11 PRECIPIT

The infiltration classes are generally segregated based on the permeability of the outcropping geologic layer. This is due to the strong dependence the permeability has on the percent of precipitation which actually recharges the groundwater system. However, it is quite possible that due to topographic and geographic differences the precipitation rates vary over the outcrop area of an individual geologic series. Program PRECIPIT calculates the mean precipitation over the entire area of an infiltration class by using a digitized precipitation map. It should be noted that these values must still be reduced by some percentage to represent the infiltration rate for each class rather than the precipitation rate.

6.2 Post processing routines

A series of post-processing programs have been developed to assist the modeler in his or her interpretation of the results. These routines consist of plotting programs, programs to sum boundary discharges or recharges over particular areas of interest, and calculational routines to evaluate the hydraulic gradient or hydraulic head at any point in the three-dimensional modeled region. The most significant of these routines developed specifically for use with FEM 301 include:

GRAD
FEDP320
DEBSEG
QSURF2
DEBALI

These programs are briefly discussed below. All of these programs have been structured to be interactive in order to be flexible for user application.

The only direct output from FEM 301 consists of the calculated heads at every nodal point and the calculated volumetric recharge or discharge at every constant head node. Without graphically displaying these results or otherwise representing the output, it would be a very tedious process for the modeler to evaluate the results. This is especially true of large two- or three-dimensional models which contain thousands of nodes.

6.2.1 GRAD

Program GRAD calculates the hydraulic gradient ($-\vec{\text{grad}} h$) at a specified number of points in each element the user indicates. The input required for this program consists of an element file (which is the same structure as the element file used in FEM 301 but need only include those elements for which gradients are desired), a coordinate file (which again is the same form as used in FEM 301 but need only include those nodes within the elements for which gradients are desired), and the results file from a FEM 301 simulation. In addition, the user must specify the elements for which gradients are desired and the vertical columns containing these elements.

The user must also define the number of points **along each edge** of the specified elements for which gradients are to be calculated. For example, in a brick element, if the user specifies 3 points per edge, a total of 27 gradients will be calculated; the 8 corner points, the 12 midside

points, the 6 midface points, and the 1 point in the center of the element.

An example of the output from program GRAD is depicted in Figure 6-2. Following a definition of the element number and node numbers within this element, the output consists of the x, y, and z coordinates of the point, the horizontal gradient direction (negative to west, positive to east, 0 is due north), the vertical gradient direction (negative down, positive up, 0 is horizontal), the magnitude of the gradient, and the head at the point.

6.2.2 FEDP320

Program FEDP320 is a graphics routine to permit the contouring of nodal heads (or elevations). The input required for this program is a two-dimensional element file, the three-dimensional coordinate file and a file containing the results from FEM 301. In addition, a file containing the contours to be employed and the scale of the plot is required. The user must interactively define the type of pen(s) and paper to be used. The element file may either be a horizontal or vertical section or a combination of these if the user wishes to create a block diagram. In order to assist the user in creating the two-dimensional element files, programs COUPES, HORIZSURF, and HORIZTRID have been developed. Program COUPES forms a two-dimensional element file containing all vertical element faces beneath a specified segment of nodes at the surface. This segment must, however, be along the sides of the two-dimensional surface elements. That is random sections cannot, at present, be readily specified. Program HORIZSURF extracts the two-dimensional surface elements (e.g. for plotting surface heads). Program HORIZTRID extracts two-dimensional elements along any structural surface (e.g. the base of an aquifer) to a three-dimensional mesh. This is simply done by searching for all nodes within a certain (min., max.) interval.

Several different viewing orientations are possible with program FEDP320. These are summarized in Figure 6-3, where x, y, and z correspond to the coordinates of the mesh and s and t correspond to the plot coordinates. At present angles a and b are 7° and 41° , respectively. Most of the vertical sections plotted in NTB 84-50 are using orientations 1 (looking towards the northeast) or 4 (looking towards the southeast).

FEDP320 develops contours by using a piecewise linear interpolation between punctual values. Each face of elements included within the area to be contoured is divided into 6 segments (7 points) on an edge. At these points (49 for a rectangular element) the value of head (or elevation) is evaluated exactly using the basis functions. Between these points, the values are linearly interpolated. The number of segments is somewhat arbitrarily chosen to maximize detail yet minimize computational costs.

Figure 6-2: Example Output from Program "GRAD".

```

*****
* PROGRAMME "GRAD" - MODELE A ELEMENTS FINIS      *
* CALCUL DES VECTEURS GRADIENTS (-grad H)         *
* CENTRE D'HYDROGEOLOGIE (UNIVERSITE DE NEUCHATEL) *
*****

```

NUMERO DE LA COLONNE : 1

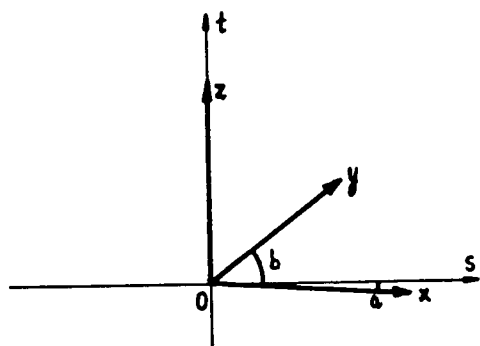
=====

FICHER DES RESULTATS: FRAC.RES

=====

1	1	27	1	6	11	12	13	8	3	2	7
			36	41	46	47	48	43	38	37	42
			71	76	81	82	83	78	73	72	77
1	250.000	250.000	50.000		-69.1	-4.54	0.18E-02			1.098	
2	250.000	250.000	100.000		-72.4	34.20	0.24E-02			1.067	
3	250.000	250.000	150.000		-78.8	45.69	0.40E-02			0.960	
4	500.000	250.000	50.000		-72.9	-1.49	0.17E-02			1.509	
5	500.000	250.000	100.000		-77.3	-8.63	0.17E-02			1.517	
6	500.000	250.000	150.000		-94.8	-14.52	0.19E-02			1.535	
7	750.000	250.000	50.000		-70.7	-3.89	0.17E-02			1.908	
8	750.000	250.000	100.000		-70.9	11.94	0.15E-02			1.903	
9	750.000	250.000	150.000		-71.3	38.29	0.12E-02			1.876	
10	250.000	500.000	50.000		-59.7	-4.25	0.19E-02			0.896	
11	250.000	500.000	100.000		-62.9	32.62	0.26E-02			0.864	
12	250.000	500.000	150.000		-70.3	42.30	0.44E-02			0.754	
13	500.000	500.000	50.000		-56.6	3.86	0.20E-02			1.311	
14	500.000	500.000	100.000		-55.7	-17.74	0.22E-02			1.324	
15	500.000	500.000	150.000		-53.2	-31.56	0.28E-02			1.377	
16	750.000	500.000	50.000		-59.4	-2.25	0.19E-02			1.719	
17	750.000	500.000	100.000		-55.3	5.56	0.18E-02			1.717	
18	750.000	500.000	150.000		-33.0	17.72	0.14E-02			1.702	
19	250.000	750.000	50.000		-42.6	-2.12	0.18E-02			0.609	
20	250.000	750.000	100.000		-42.6	46.39	0.28E-02			0.561	
21	250.000	750.000	150.000		-42.0	61.37	0.47E-02			0.408	
22	500.000	750.000	50.000		-44.3	-3.61	0.23E-02			0.965	
23	500.000	750.000	100.000		-40.8	28.32	0.29E-02			0.934	
24	500.000	750.000	150.000		-31.7	39.83	0.46E-02			0.826	
25	750.000	750.000	50.000		-56.5	-0.92	0.25E-02			1.429	
26	750.000	750.000	100.000		-54.3	20.45	0.27E-02			1.406	
27	750.000	750.000	150.000		-49.7	34.03	0.35E-02			1.333	

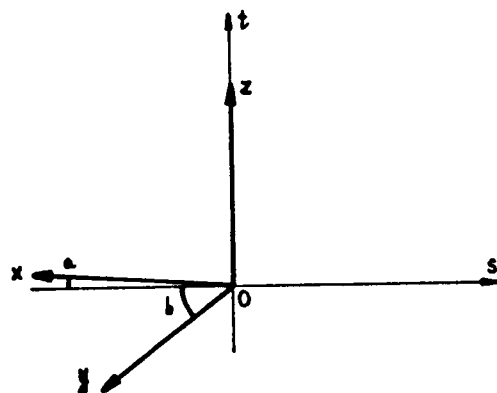
ORIENTATION=1



$$s = x \cdot \cos(a) + 0.5 \cdot y \cdot \cos(b)$$

$$t = -x \cdot \sin(a) + 0.5 \cdot y \cdot \sin(b) + z$$

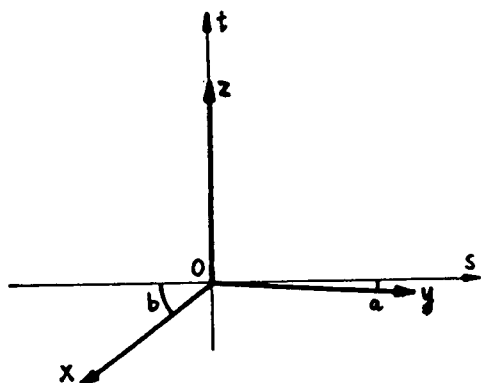
ORIENTATION=2



$$s = -x \cdot \cos(a) - 0.5 \cdot y \cdot \cos(b)$$

$$t = x \cdot \sin(a) - 0.5 \cdot y \cdot \sin(b) + z$$

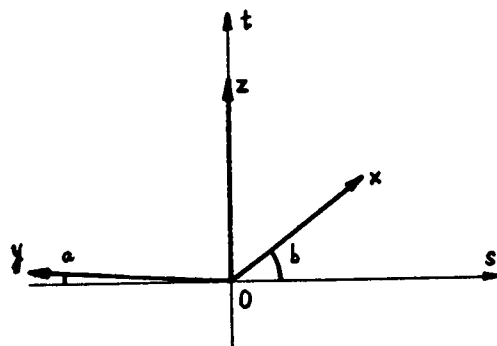
ORIENTATION=3



$$s = y \cdot \cos(a) - 0.5 \cdot x \cdot \cos(b)$$

$$t = -y \cdot \sin(a) - 0.5 \cdot x \cdot \sin(b) + z$$

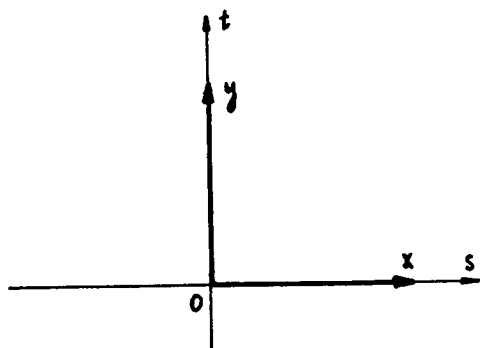
ORIENTATION=4



$$s = -y \cdot \cos(a) + 0.5 \cdot x \cdot \cos(b)$$

$$t = y \cdot \sin(a) + 0.5 \cdot x \cdot \sin(b) + z$$

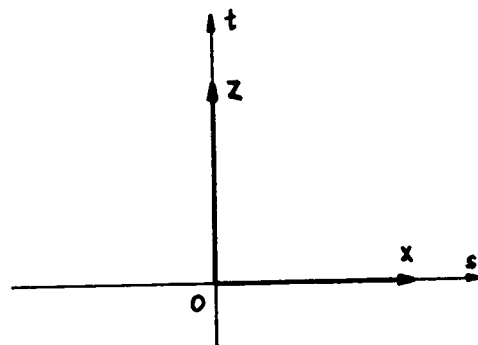
ORIENTATION=5



$$s = x$$

$$t = y$$

ORIENTATION=6



$$s = x$$

$$t = z$$

Figure 6-3 Viewing Orientations for Two-Dimensional Plots using FEDP320

6.2.3 DEBSEG

Program DEBSEG sums the nodal volumetric fluxes calculated at constant head nodes for a specified group of nodes. This nodal grouping may be along various segments of a river or for a particular geologic formation (either at the surface or along lateral boundaries). The fluxes are summed separately for positive and negative values and an overall total. It should be noted that the finite element model conserves mass for the entire model but care must be taken in interpreting volumetric inflows/outflows over subdomains of the model.

6.2.4 QSURF2

Program QSURF2 transforms the nodal recharge rates (in m^3/s) to nodal infiltration rates (in mm/yr) by dividing the calculated flux by the area of influence of each node. The area of influence is calculated in program BALIM (see below). The calculated infiltration rate is not corrected for the fact that when a node is on the boundary of two geologic series, the values of inflow should be weighted by the permeability of each series.

Program BALIM calculates the area of influence for each surface node. The total area and the individual areas for each element the node is contained within are presented separately.

As in the case of program DEBSEG, care must be taken in interpreting the results from program QSURF2. Any individual nodal flux has little meaning. Only summed fluxes may be rationally analyzed. Even so, the calculated infiltration rates from DEBSEG give a feel for the spatial variation of the output.

6.2.5 DEBALI

Program DEBALI calculates the total recharge to and discharge from each geologic series (defined on the basis of the infiltration class) at the surface of a three-dimensional model. The total recharge consists of both the applied infiltration rate times the surface area and the positive nodal inflows. The individual nodal fluxes (m^3/s) are proportioned to the appropriate geologic series based on the area of influence of each series at each node (by program BALIM) and the permeability of each element sharing the node. Hence if a node is at the boundary of an aquifer and an aquitard, the calculated nodal recharge or discharge is assigned to the aquifer. In addition to calculating the volumetric totals for each geologic series, DEBALI also calculates the percent of the total recharge and discharge which occurs within each series.

The input to program DEBALI consists of files containing the mean precipitation rate over each geologic series (from program PRECIPIT), the areas of influence for each surface node (from program BALIM), the parameter file, the segments to be summed, and a result file from FEM 301. One-dimensional elements at the surface which correspond to alluvial aquifers are incorporated into the adjacent two-dimensional elements for summation purposes. One-dimensional elements which correspond to the surface expression of two-dimensional elements in the subsurface (whether faults or two-dimensional aquifers) are summed separately rather than added to the adjacent two-dimensional surface elements.

An example output from program DEBALI is reproduced in Figure 6-4. By comparing the assigned infiltration rate (AD) and the calculated mean precipitation rate (Pm), the percent of precipitation which actually infiltrates (% Pm) may be calculated. The recharge (R) represents the total volumetric infiltration over each infiltration class (CAD). While the positive and negative fluxes (D+ and D-) are broken out in program DEBALI; as before, these numbers by themselves have no physical meaning but the sum of D+ and D- over large areas should represent the net inflow or outflow. The percents of total recharge and total discharge (% Rt and % Dt) indicate the relative boundary fluxes to each geologic series. The sum of the recharges and discharges for all classes should equal zero, with the exception of lateral boundary inflows and outflows which are summed using program DEBSEG.

Figure 6-4: Example Output from "DEBALI".

Fichier des donnees consultes:
Fichier des precipitations moy.: PRECIPITA.TAB;3
Fichier des resultats : E15V02.RES;1
Fichier des segments : E13FNT.SEG;3
Fichier des surfaces : E15V01.SUR;2
Fichier des colonnes : E15.COL;4
Fichier des parametres : E15V02.PAR;3
Dictionnaire des elts. : F15DEBALI.DIC;6

Legende:
CAD = Classe d'alimentation distribuee
AD = Valeur d'alimentation distribuee [mm/an]
Pm = Precipitations moyennes [mm/an]
%Pm = 100*AD/Pm
D- = Debits negatifs REPARTIS CUMULES [m3/s]
D+ = Debits positifs REPARTIS CUMULES [m3/s]
R = Recharge [m3/s]
Rs = R + D+ [m3/s]
Rt = Somme des Rs [m3/s]
Dt = Somme des D- [m3/s]
%Rt = 100*Rs/Rt
%Dt = 100*D-/Dt
BILAN = Rs + D- [m3/s]

CAD	AD	Pm	%Pm	R	D+	Rs=R+D+	%Rt	D-	%Dt	BILAN=Rs+D-
2	225	1887	11.92	14.02	22.97	36.99	17.70	-39.32	18.79	-2.34
10	225	1843	12.21	2.10	4.68	6.78	3.24	-4.07	1.94	2.71
20	300	1662	18.05	0.65	2.21	2.86	1.37	-1.08	0.51	1.78
30	300	1920	15.63	0.00	0.28	0.28	0.13	-0.08	0.04	0.20
31	0	1715	0.00	0.00	0.00	0.00	0.00	-0.04	0.02	-0.04
40	151	1944	7.77	1.43	2.16	3.58	1.72	-3.53	1.69	0.05
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
401	0	992	0.00	0.00	0.00	0.00	0.00	-0.02	0.01	-0.02
420	75	987	7.60	0.21	0.02	0.24	0.11	-0.23	0.11	0.01
440	0	1153	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TOTAUX				115.96	93.03	209.00	100.00	-209.32	100.00	

7. SAMPLE PROBLEMS AND VERIFICATION TESTS

The previous chapters have dealt with the mathematical and numerical framework that is the basis for FEM 301. In addition, the auxiliary programs which have been developed to assist the user in generating the required input and interpreting the output file have also been described. The purpose of the present chapter is to illustrate the application of FEM 301 to a few simple groundwater flow problems. Two of these applications are verification tests of the program to compare the simulated results with analytical solutions of the steady state groundwater flow equation. While these examples are not exhaustive, they should give the reader a feel for the program's capabilities.

7.1 Example Problem 1:

A Fractured Block with 1-D, 2-D and 3-D elements

In order to show the potential user of FEM 301 an example of the input and output files, a simple three-dimensional block model with one-, two-, and three-dimensional elements has been created. This model is illustrated in Figure 7-1. This model is totally hypothetical and was only constructed to show the flexibility of FEM 301.

The model consists of 18 three-dimensional, Lagrangian (i.e. 27 node) elements, each 50 m thick and having edges 500 m long. The model is divided into three layers. The model contains 18 two-dimensional Lagrangian (i.e. 9 node) elements which correspond to vertical faults. These faults exist only in the top two layers. In addition, 6 two-dimensional elements are used on the top of the model to correspond to a surficial aquifer. Six one-dimensional elements are also incorporated in the model, as illustrated in Figure 7-2.

The element and coordinate files for this block model are presented in Figures 5-1 and 5-3, respectively. The three-dimensional elements numbered from 1 to 18 are permeability class 1 and infiltration class 1 and have 27 nodes and 12 edges. Note the "corkscrew" numbering of element 1 (the lower left hand corner of the model). The two-dimensional faults numbered in the 200's and 300's are permeability class 3 and infiltration class 3 and have 9 nodes and 4 edges. The two dimensional surface elements numbered from 113 to 118 are permeability class 2 and infiltration class 2. The one-dimensional elements numbered in the 400's, 500's and 600's are of permeability class 4 and infiltration class 4. The coordinates of the model start at $x = 250$ m, $y = 250$ m, $z = 50$ m in the lower left hand corner.

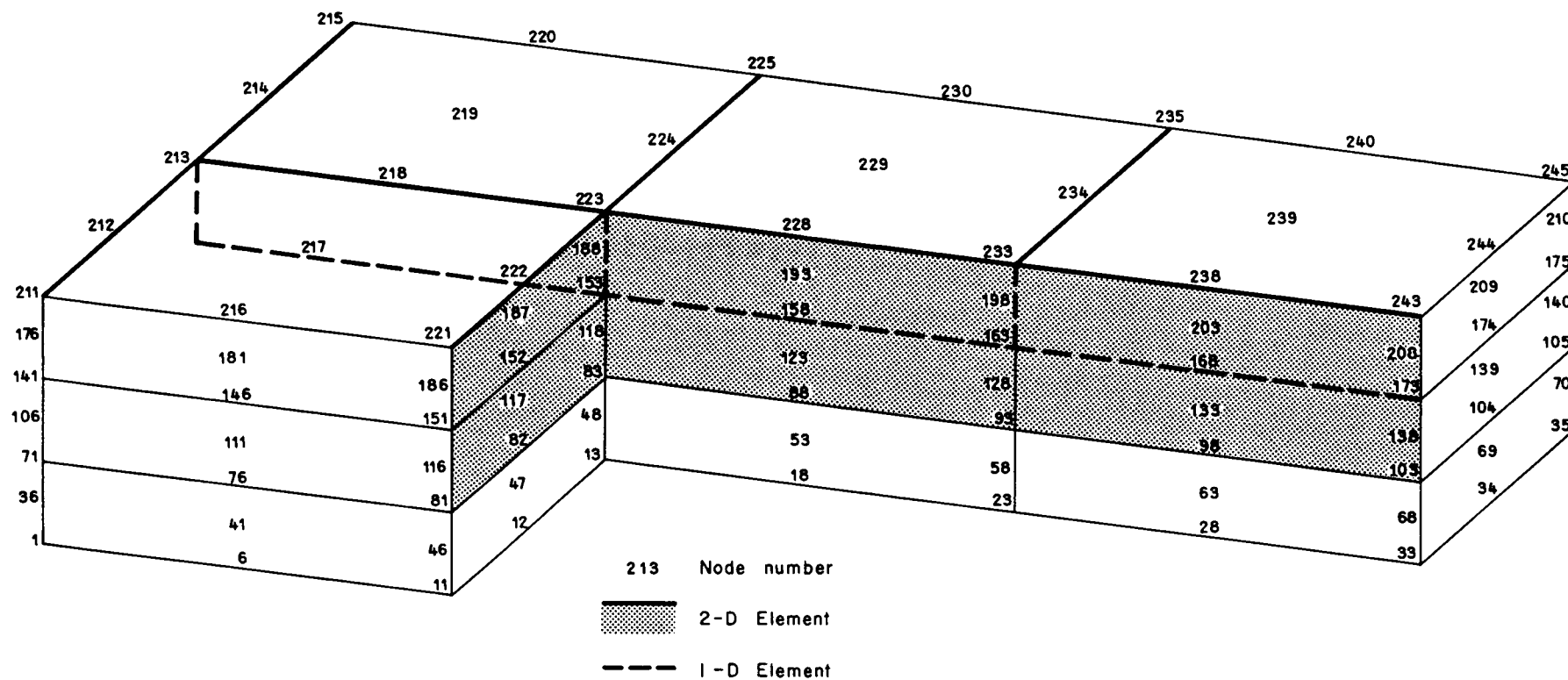


Figure 7-1: Three-Dimensional Fractured Block Model with 1-D, 2-D and 3-D Elements

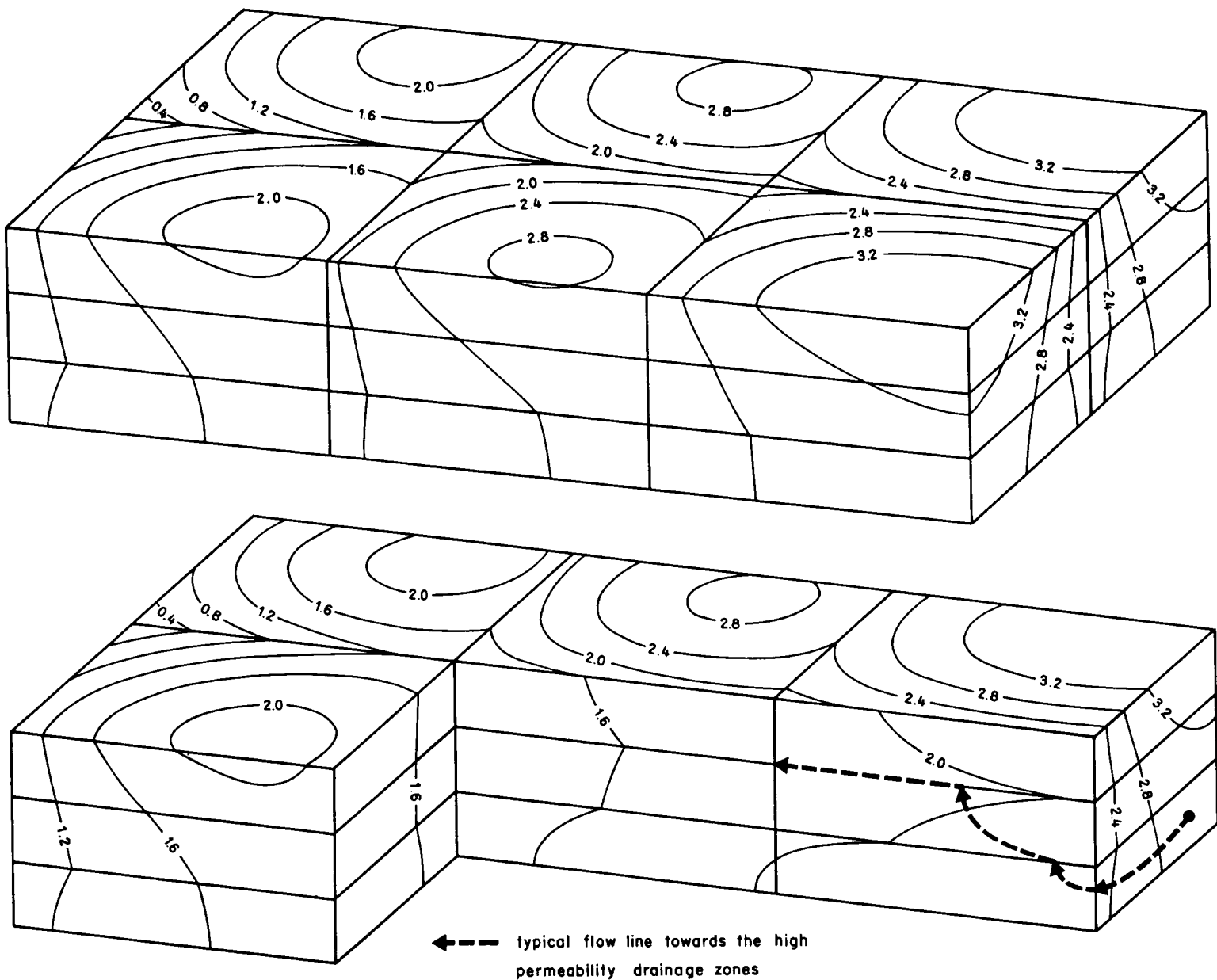


Figure 7-2/3: Calculated Potentials of the Fractured Block Model

The parameter file used as input to example problem 1 is illustrated in Figure 5-4. The permeability of the three-dimensional elements (class 1) is $5\text{E-}6$ m/s. The permeability times thickness of the surface elements (class 2) is $5\text{E-}6$ m²/s. The permeability times width of the vertical faults (class 3) is $5\text{E-}3$ m²/s. The permeability times the cross-sectional area of the one-dimensional elements (class 4) is 10 m³/s. Infiltration is applied over the entire surface (class 2) at the rate of $2\text{E-}8$ m/s. All of this water must leave the system at node 213 where a constant head of $0.$ m has been assigned.

The output file generated from FEM 301 using the above input files is illustrated in Figure 5-5. The statistics of the run are printed under PARAMETRES DE CONTROLE; with the node numbers varying from a minimum of 1 (MNNIC) to a maximum of 245 (MXNIC), the number of infiltration classes (MXNQ) and permeability classes (MXNP) being equal to 4, the number of elements (MXLM) equal to 48, the total number of active variables (NDACT) equal to 245, and the maximum front width (MXPA) equal to 57. Following the printing of the input values from the parameter file (i.e. the permeabilities, infiltration rates, and nodal boundary conditions), the calculated results are presented. The calculated nodal heads (m) are printed in the third column and the calculated nodal volumetric recharge (+) or discharge (-) rates (m³/s) are printed in the fourth column. As expected, the nodal discharge rate at node 213 is 0.03 m³/s ($= 2\text{E-}8$ m/s * 1000 m * 1500 m). It should be noted that the sum of the recharge volumes at the other surface nodes is 0.029708 m³/s. The difference is due to the fact that the infiltration over the surface area of influence of node 213 (which should be $0.278\text{E-}3$ m³/s as at nodes 221, 225, etc.) is **not** incorporated in the discharge value printed for node 213. That is the volumetric discharge from node 213 is the total discharge not the net discharge.

In order to better illustrate the calculated flow field, Figure 7-2 depicts the equipotentials drawn on a cut away block diagram of the model. As expected, the one- and two-dimensional permeable features have a considerable effect on the overall flow regime.

7.2 VERIFICATION OF FEM 301

In order to generate confidence in the appropriateness of the numerical model and the correctness of the coding, it is beneficial to compare the results of a physical problem solved by FEM 301 to the results generated by alternative solution procedures. This comparison of results is commonly referred to as code verification. Verification may take the form of either comparing results from the numerical solution to results from an analytical solution of the same physical problem, or by comparing the solutions generated by two different computer codes for the same problem (code-to-code comparison).

FEM 301 has been verified by comparing numerically generated results to analytical solutions for two groundwater flow problems involving wells in bounded aquifers. For one of these problems, the results have also been compared to those generated using a finite difference numerical model. In all cases, the comparisons are very favorable.

Additional verification tests of FEM 301 are planned. An international study for comparing groundwater hydrology models and modelling strategies has been proposed by the Swedish Nuclear Power Inspectorate (SKI). Appropriate problems defined in this study (HYDROCOIN) will be analyzed using FEM 301. This will enable comparison to a broad suite of programs utilized in other countries. In addition to the above, the unique capabilities of FEM 301 (e.g. the ability to handle one-, two-, and three-dimensional elements) will be tested by solving a limited subdomain of an "infinite" fractured network. While it is difficult to completely test all of the capabilities of the present program, we feel these tests to be sufficient to provide a high degree of confidence in the numerical accuracy of the simulated results.

7.2.1 Well in a two-dimensional bounded aquifer

As the present version of FEM 301 was developed to address steady state groundwater flow in porous or fractured media, the physical problems to be analyzed must have steady state solutions, i.e. the volume of water which flows into the modeled region must equal the volume of water which flows out. One such problem may be generated by placing a well in a rectangular aquifer bounded on 3 sides by impermeable boundaries and on the fourth side by a constant head (recharge) boundary. At steady state, the volume of water discharged by the well should equal the volume of water which inflows across the recharge boundary. The physical description of this problem and analytical solution to this problem are described below, followed by the results generated by FEM 301 for a particular parameter set.

Two-dimensional groundwater flow in a finite dimension aquifer may be physically represented by Figure 7-4. The aquifer is $y_1 - y_0$ units wide and $x_1 - x_0$ units long with a well located at x_{wi}, y_{wi} pumping at the rate of GL_j . The transient flow equation is described by:

$$S \frac{\partial H}{\partial t} = T_{xx} \frac{\partial^2 H}{\partial x^2} + T_{yy} \frac{\partial^2 H}{\partial y^2} + \sum_{i=1}^n GL_i \delta(x - x_{wi}) \delta(y - y_{wi})$$

where S is the storativity (L^0)
H is the hydraulic head (L)
t is time
 T_{xx}, T_{yy} are the anisotropic, homogeneous transmissivities (L^2/t)
 δ is the Kronecker delta
n is the number of wells

Given the geometry indicated on Figure 7-4, this equation is subject to the following boundary conditions:

$$H = H_0 \quad x = x_0, \quad t \geq 0$$

$$\frac{\partial H}{\partial x} = 0 \quad x = x_1$$

$$\frac{\partial H}{\partial y} = 0 \quad y = y_0$$

$$\frac{\partial H}{\partial y} = 0 \quad y = y_1$$

$$H = H_0 \quad t = 0$$

The analytical solution to this problem is (Hydrologisch Colloquium, 1964):

$$H = H_0 + 2 (x_1 - x_0) / T_{xx} / (y_1 - y_0) \sum_{k=0}^{\infty} \sum_{j=0}^{\infty} \sum_{i=1}^n GL_i \cdot B_k \cos \left[\beta_k (y_{wi} - y_0) / (y_1 - y_0) \right] \cos \left[\beta_k (y - y_0) / (y_1 - y_0) \right] \cdot \sin \left[\alpha_j (x_{wi} - x_0) / (x_1 - x_0) \right] \sin \left[\alpha_j (x - x_0) / (x_1 - x_0) \right] \cdot \frac{(1 - [\exp - (\alpha_j^2 + \beta_k^2 / P_1) t] T_{xx} / S / (x_1 - x_0)^2)}{(\alpha_j^2 + \beta_k^2 / P_1)}$$

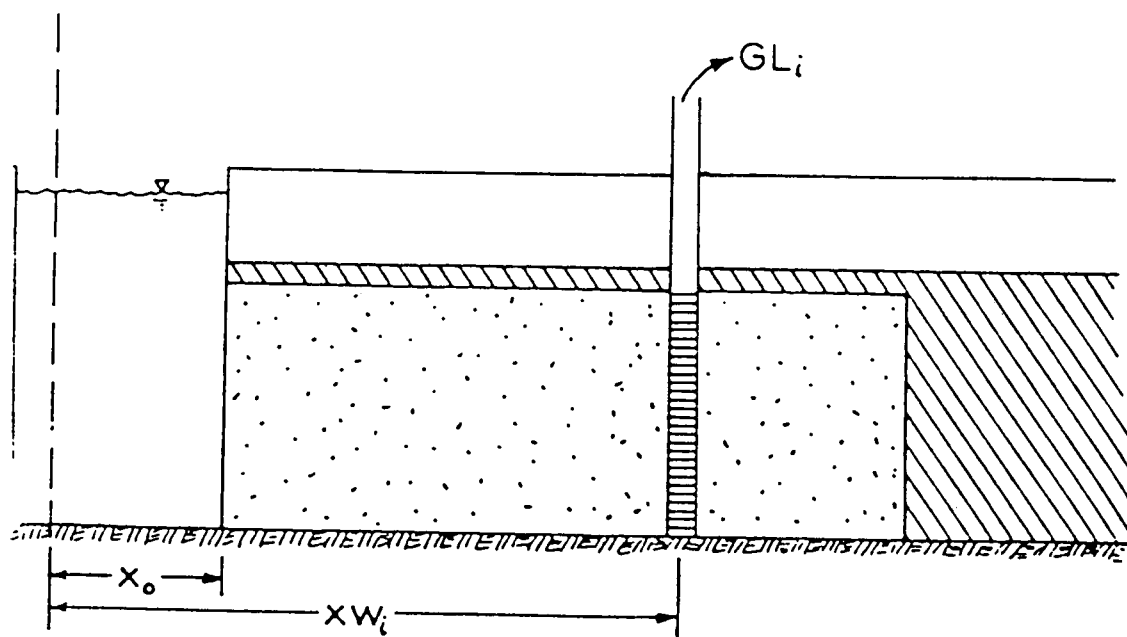
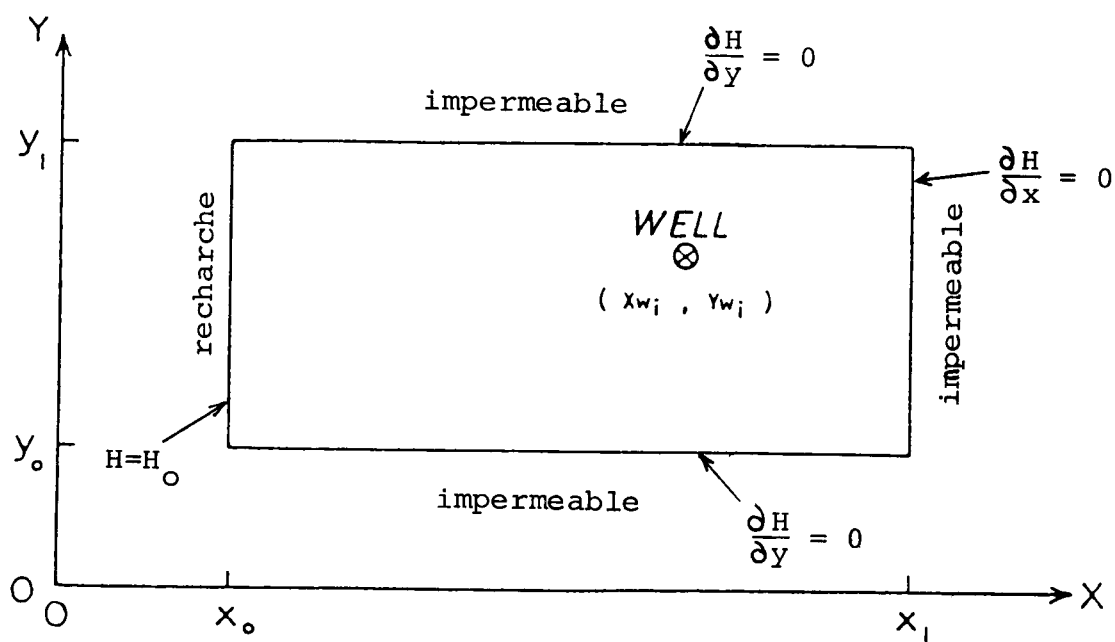


Figure 7-4: Aquifer geometry for Two-Dimensional, State Flow to a Well

$$\alpha_j = (2j + 1)\pi/2 \quad j = 0, 1, \dots$$

$$\beta_k = k\pi \quad k = 0, 1, \dots$$

$$P_1 = T_{xx}(y_1 - y_0)^2 / T_{yy}(x_1 - x_0)^2$$

$$B_k = \begin{cases} 1 & k = 0 \\ 2 & k > 0 \end{cases}$$

While the summations over indices k and j are to infinity (as a result of incorporating all image wells across the boundaries) for all practical purposes these summations may be stopped at a finite number with an infinitesimal effect on the calculated heads. This solution is the transient solution which may be modified to the steady state solution (t equal infinity) by removing the $\exp[\quad]$ term. A small computer program (LINBC) has been developed by INTERA Technologies, Inc. to calculate the heads based on the above analytical solution. This program is reproduced in Appendix A4.

In order to test the ability of FEM 301 to reproduce the results calculated from the analytical solution of the steady state flow equation, a particular parameter set has been chosen. Any other parameters could have been selected and the relative comparison would be the same. The assigned transmissivity of the aquifer (T_{xx}) is $8.02 \times 10^{-4} \text{ m}^2/\text{sec}$. The applied pumping rate (GL_j) has been assigned a value of $2.294 \times 10^{-3} \text{ m}^3/\text{sec}$. The well is located in the center of a rectangular aquifer 304.8 m ($y_1 - y_0$) by 609.6 m ($x_1 - x_0$). The constant boundary head (H_0) at $x = x_0$ is 61.53 m .

The modeled region has been discretized into three separate finite element meshes in order to evaluate the effect of the discretization on the simulated results. These meshes are illustrated in Figure 7-5a, 7-5b, and 7-5c for meshes A, B, and C, respectively.

While results could be presented for all nodes, a partial selection has been made for comparison purposes. These nodes are illustrated on Figure 7-5a. The calculated potential surface for all meshes and the analytical solution is illustrated in Figure 7-6. As these surfaces are indistinguishable, a more detailed comparison is contained within Table 7-1. These results show excellent agreement between the analytical and numerical solutions except at the well itself (node 87). It should be noted, however, that at the well the analytical solution also deteriorates. This same problem was run using SWENT (INTERA, 1983), in which the nodal points of finite element mesh A were made to correspond to the grid block centers of the finite difference mesh used in SWENT. Again the agreement in Table 7-1 is quite good.

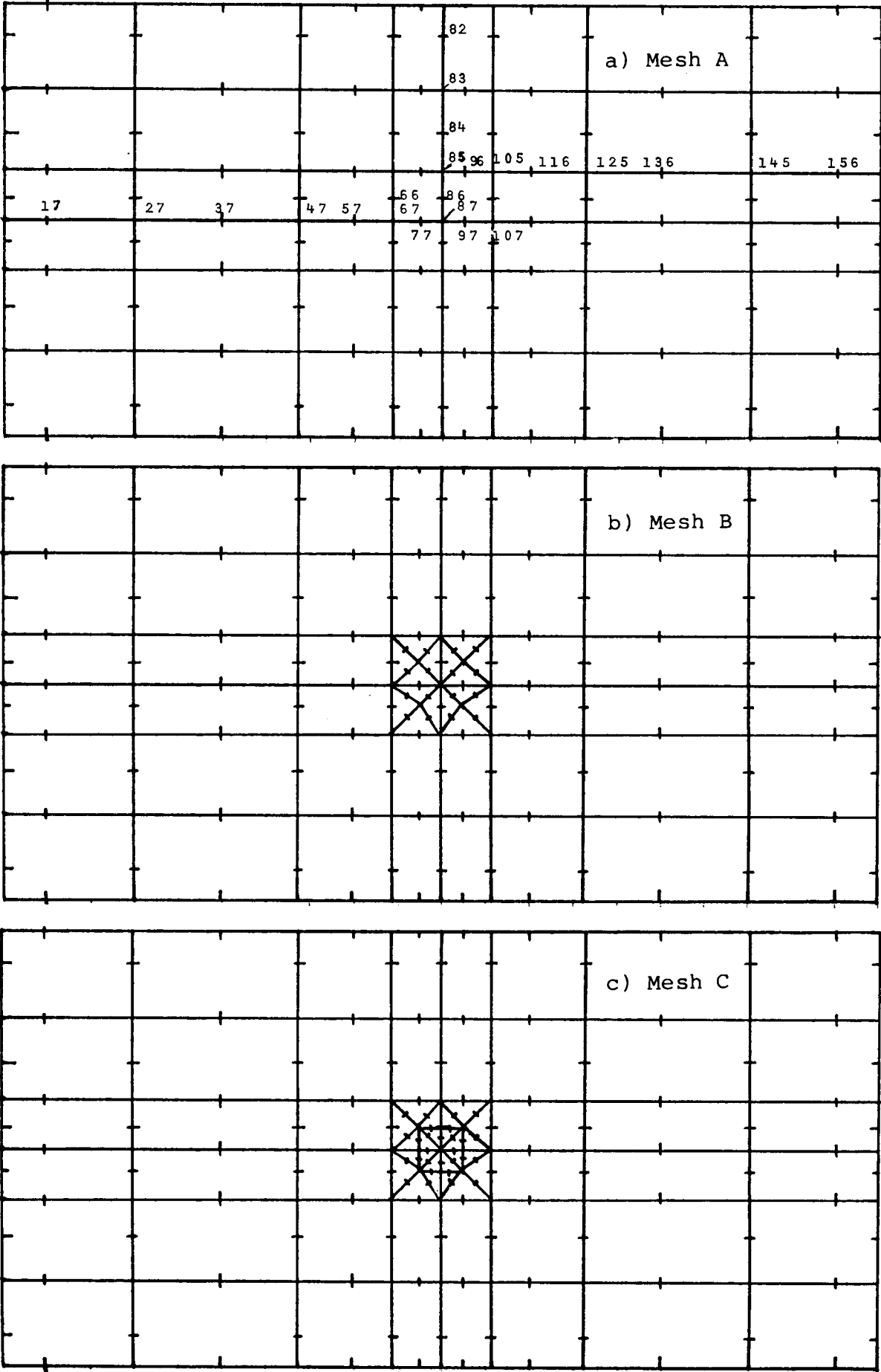


Figure 7-5 Finite Element Meshes for Two-Dimensional Verification Test

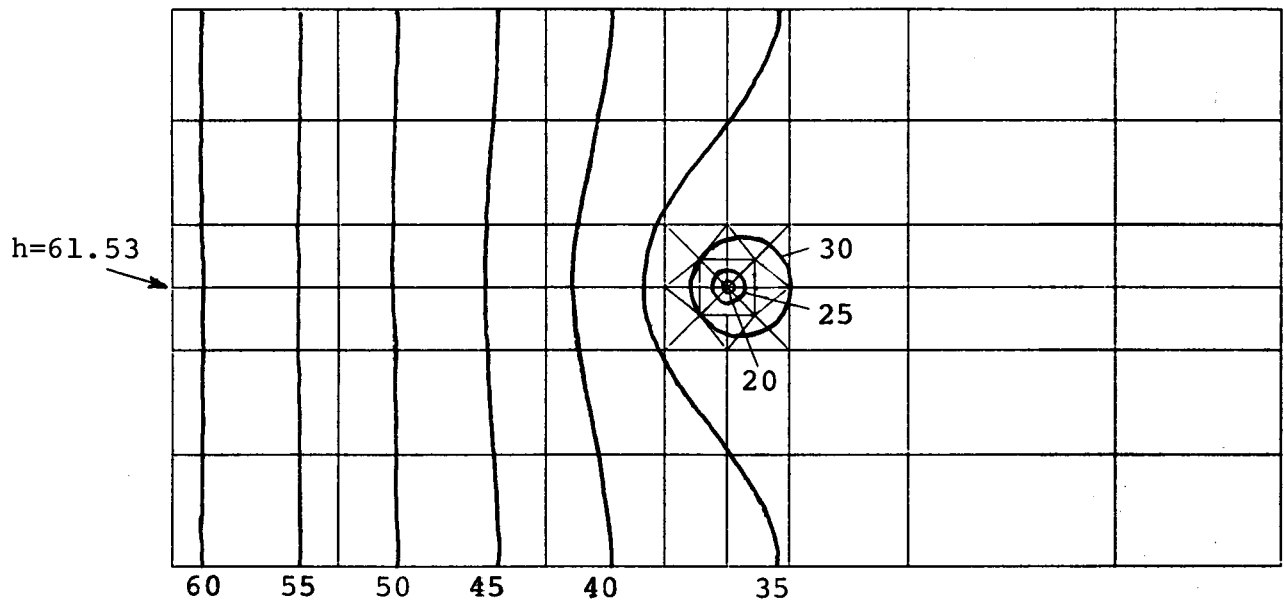


Figure 7-6: Simulated Equipotentials for Steady State Flow to a Well in a bounded Aquifer

Table 7-1 Comparison of Analytical Solution and FEM 301 for flow in a bounded aquifer

Node	Analytical	Mesh A	Mesh B	Mesh C	SWENT
17	58.67	58.66	58.66	58.66	58.86
27	52.91	52.90	52.89	52.89	52.87
37	47.04	47.04	47.04	47.04	47.17
47	41.61	41.52	41.58	41.58	41.54
57	37.14	37.21	37.14	37.15	37.14
67	33.07	33.43	32.96	33.05	32.82
77	28.46	27.57	28.24	28.60	27.90
87	18.52	17.94	16.34	15.26	20.08
97	27.02	26.14	26.82	27.17	26.56
107	29.85	30.22	29.73	29.83	29.71
66	33.58	33.19	33.47	33.46	33.46
86	27.62	26.82	27.49	27.85	27.25
106	30.34	29.97	30.25	30.25	30.22
82	35.97	35.94	35.95	35.95	36.11
83	35.13	35.02	35.10	35.10	35.19
84	33.68	33.77	33.69	33.70	33.71
85	31.26	31.64	31.16	31.25	31.04
65	34.55	34.72	34.57	34.55	34.47
76	32.41	32.15	32.42	32.42	32.42
96	30.98	30.72	31.00	30.99	30.99
105	31.33	31.50	31.35	31.33	31.25
116	31.92	32.02	31.93	31.93	31.98
125	32.49	32.45	32.49	32.49	32.42
136	32.79	32.77	32.78	32.78	32.85
145	32.90	32.89	32.88	32.88	32.84
156	32.90	32.91	32.91	32.91	32.87

7.2.2 Steady flow to a well in a leaky, confined aquifer

During pumping from an aquifer partially confined by a semi-pervious aquitard, leakage into the aquifer occurs across the leaky layer(s). A schematic diagram of the flow system is illustrated in Figure 7-7. When the permeability of the aquitard (K') is much less than the permeability of the aquifer (K), an approximate solution may be generated by assuming the flow in the aquitard is vertical and flow in the aquifer is essentially horizontal. After a certain period of time, a steady state flow system is attained with flow in the aquifer being maintained entirely by the leakage (assuming the head in the upper aquifer is maintained at a constant level). Given the flow domain is axisymmetric the resulting flow equation may be written:

$$\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial h}{\partial r} \right) + \frac{h_o - h}{\lambda^2} = 0$$

where $\lambda^2 = \frac{B' B K}{K'}$ is referred to as a leakage factor.

The general solution to this equation is (Bear, 1979, pg. 313):

$$s(r) = h_o - h(r) = \alpha I_0(r/\lambda) + \beta K_0(r/\lambda)$$

where α and β are constants to be derived from the boundary conditions, $I_0(x)$ and $K_0(x)$ are Modified Bessel functions of the first kind and second kind of zero order, and $s(r)$ is the drawdown in the aquifer.

If the aquifer is effectively infinite, the boundary condition becomes $h = h_o$ at $r = \infty$, hence $\alpha = 0$. The second boundary condition is the constant discharge rate at the well, hence

$$Q_w = 2 \pi r_w B K \left. \frac{\partial h}{\partial r} \right|_{r=r_w} = 2 \pi r_w B K \beta \frac{K_1(r_w/\lambda)}{\lambda}$$

$$\text{thus } \beta = \frac{Q_w}{2 \pi T (r_w/\lambda) K_1(r_w/\lambda)}$$

where K_1 is the Modified Bessel function of the second kind and first order. Substituting yields

$$s(r) = h_o - h(r) = \frac{Q_w}{2 \pi T} \frac{K_0(r/\lambda)}{(r_w/\lambda) K_1(r_w/\lambda)}$$

Given that in practice the radius of the well is very small, $r_w \ll 1$, the denominator in this equation is effectively one, and thus may be simplified to:

$$s(r) = h_o - h(r) = \frac{Q_w}{2\pi T} K_o(r/\lambda)$$

A series of runs have been made with FEM 301 in order to compare the simulated and analytical solutions. In some cases the assumptions required for the analytical solution to be valid are not met (e.g. the drawdown in the aquifer extends to the boundary implying the aquifer is not infinite, or there is horizontal flow in the aquitard, or there is vertical flow in the aquifer) thus making a comparison of results meaningless. However, in the problem described below the analytical approximation appears to be valid.

The system modeled is a three-layered radial geometry. Due to the steeper hydraulic gradients near the well, a much finer discretization is required in the center of the modeled domain. An exponential function was employed to define the radial extent of each element. The horizontal layers consist from top down of a very permeable aquifer, a low permeability aquitard, and a high permeability aquifer which is pumped. The thickness of each layer is 100 m. To evaluate the effect of varying radial distances to the boundary and different leakage factors, two runs have been made. In run PUITSV03 the permeability of the uppermost layer (K_1) is 10^{-3} m/sec, the permeability of layer 2 (K_2) is 10^{-7} m/sec, the permeability of layer 3 (K_3) is 10^{-3} m/sec, and the radial distance to the boundary is 5987.41 m. In run PUITSV05, $K_1 = 10^{-3}$ m/sec, $K_2 = 10^{-10}$ m/sec, $K_3 = 10^{-3}$ m/sec, and the radial distance to the boundary is 5987.41 m. In both runs a drawdown of 100 m has been assigned at the well. The results from both simulations and the comparison to the analytical solutions are presented in Tables 7-2 and 7-3. Again the comparison is excellent.

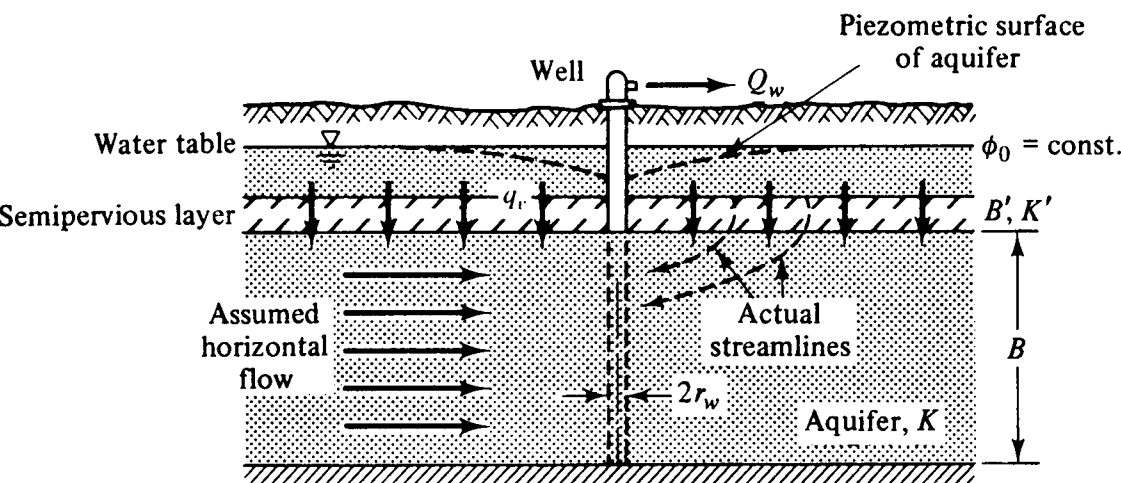


Figure 7-7: Vertical Cross-Section of Flow to a Well in a leaky, confined Aquifer (Bear, 1979)

Table 7-2 Comparison of analytical and numerical results for a well in a leaky aquifer, Run PUITSVØ3.

r(m)	Ko($\frac{r}{\lambda}$)	analytical	numerical	% difference
.068	8.561	83.196	82.94	3.1
.10	8.175	79.445	77.72	2.2
.186	7.554	73.410	71.87	2.1
.272	7.174	69.717	68.13	2.3
.505	6.556	63.711	62.19	2.4
.739	6.175	60.009	58.32	2.8
1.374	5.555	53.983	52.41	2.9
2.009	5.175	50.291	48.58	3.4
3.737	4.554	44.256	42.67	3.6
5.460	4.175	40.573	38.85	4.2
10.151	3.556	34.557	33.01	4.5
14.841	3.177	30.874	29.26	5.2
27.592	2.562	24.898	23.60	5.2
40.343	2.188	21.263	20.03	5.8
75.003	1.591	15.461	14.67	5.1
109.663	1.241	12.060	11.44	5.1
203.880	0.7221	7.017	6.75	3.8
298.096	0.4573	4.444	4.27	3.9
554.202	0.1549	1.505	1.50	0.3
810.308	0.0579	0.563	0.54	4.0
1506.477	0.0048	0.047	0.05	7.1
2202.647	0.0004	0.004	0.01	-
4095.030	-	-	<.01	-
5987.41	-	-	<.01	-

$$K_1 = 10^{-3} \text{ m/s}$$

$$K_2 = 10^{-8} \text{ m/s}$$

$$K_3 = 10^{-7} \text{ m/s}$$

$$\lambda = \left(\frac{B_2 B_3 K_3}{K_2} \right)^{1/2} = 3.1623 \times 10^2$$

$$Q \text{ calculated} = 0.6106 \times 10^{-3} \text{ m}^3/\text{sec}$$

$$s \text{ analytical} = \frac{Q}{2\pi T} \text{Ko} \left(\frac{r}{\lambda} \right)$$

Table 7-3 Comparison of analytical and numerical results for a well in a leaky aquifer, Run PUITSVØ5.

r(m)	Ko($\frac{r}{\lambda}$)	analytical	numerical	% difference
6.8	6.258	76.592	77.62	1.30
10.0	5.872	71.867	71.20	.93
18.6	5.252	64.279	63.81	.73
27.2	4.872	59.628	59.27	.60
50.5	4.253	52.052	51.75	.58
73.9	3.873	47.402	47.04	.76
137.4	3.254	39.826	39.57	.64
200.9	2.876	35.199	34.92	.79
373.7	2.263	27.697	27.52	.64
546.0	1.894	23.181	22.96	.95
1015.1	1.311	16.045	15.92	.78
1484.1	0.9773	11.961	11.80	1.30
2759.2	0.5069	6.204	6.15	.87
4034.3	0.2875	3.519	3.41	3.10
7500.3	0.0726	0.889	0.89	.16
10966.3	0.0203	0.248	0.22	11.50
20388.0	0.0007	0.009	0.01	-
29809.6	-	-	0.01	-
55420.2	-	-	<.01	-
81030.8	-	-	<.01	-
150647.7	-	-	<.01	-
220264.7	-	-	<.01	-
409503.0	-	-	<.01	-
598741.4	-	-	<.01	-

$$K_1 = 10^{-3} \text{ m/s}$$

$$K_2 = 10^{-10} \text{ m/s}$$

$$K_3 = 10^{-7} \text{ m/s}$$

$$\lambda = \left(\frac{B_2 B_3 K_3}{K_2} \right)^{1/2} = 3.1623 \times 10^3$$

$$Q \text{ calculated} = 0.769 \times 10^{-3} \text{ m}^3/\text{s}$$

$$s \text{ analytical} = \frac{Q}{2\pi T} \text{Ko} \left(\frac{r}{\lambda} \right)$$

7.3 Remarks on the behaviour of the elements

The user of FEM 301 should be aware of the fact that in certain circumstances the elements behave "incorrectly" or, better, generate unrealistic results. The aim of this section is to present two examples for such "bad behaviour" of the elements:

- in the first example pinched-sided elements are incorrectly assembled with "normal" elements
- in the second example the size of the element is too large (or, the quadratic function is too simple) to represent the true distribution of the hydraulic heads.

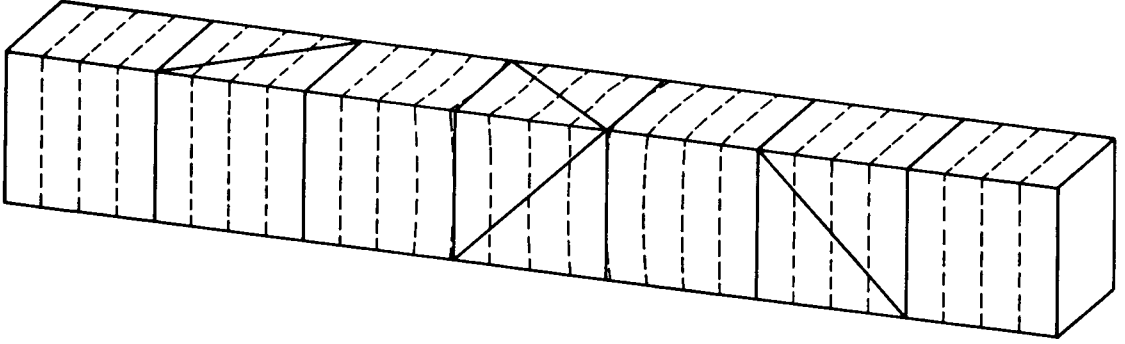
7.3.1 Pinched-sided elements

Figure 7-8a represents a constant section bar composed of 7 blocks, some of which are subdivided into smaller elements:

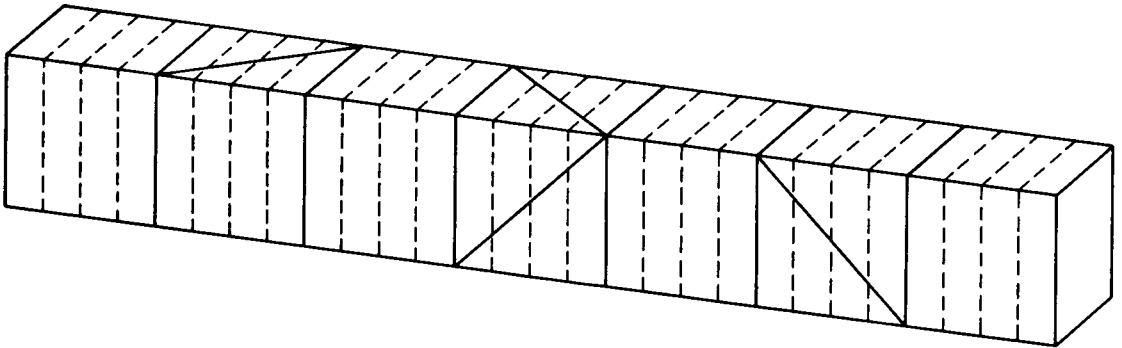
- block 1 = a 20-node brick
- block 2 = 2 triangular prisms
- block 3 = 5 pyramids + 2 tetrahedrons
- block 4 = 5 tetrahedrons
- block 5 = 5 pyramids + 2 tetrahedrons
- block 6 = 2 triangular prisms
- block 7 = a 20-node brick.

The section of the bar is 4 m^2 , its length is 14 m , and the permeability $K = 1 \text{ m/s}$ is given to each of the 25 elements. At the two ends of the bar we impose the constant heads $h = 0 \text{ m}$ (nodes 1, 2, 3, 4, 5, 6, 7, 8), and $h = 14 \text{ m}$ (nodes 113, 114, 115, 116, 117, 118, 119, 120); the four other faces of the bar being no-flow boundaries. The true solution would obviously give unit gradients everywhere in the bar, vertical equipotential surfaces, and the calculated nodal heads should be the same as the x coordinates of the nodal points. FEM 301 was applied to solve this problem. The calculated results are shown in Figure 7-8a (equipotentials). Comparing the calculated heads to the x nodal coordinates indicates an error of 5 to 7 cm for the nodes 48, 50, 54, 56, 65, 67, 71 and 73, an error which seems too important for such a small problem. The above mentioned nodal points are situated on element faces where pyramids are assembled to tetrahedrons and the errors originate from this assembly.

a) Pyramide-Tetrahedron assembly



b) "Pinched-sided" tetrahedron assembly



c) curved element sides

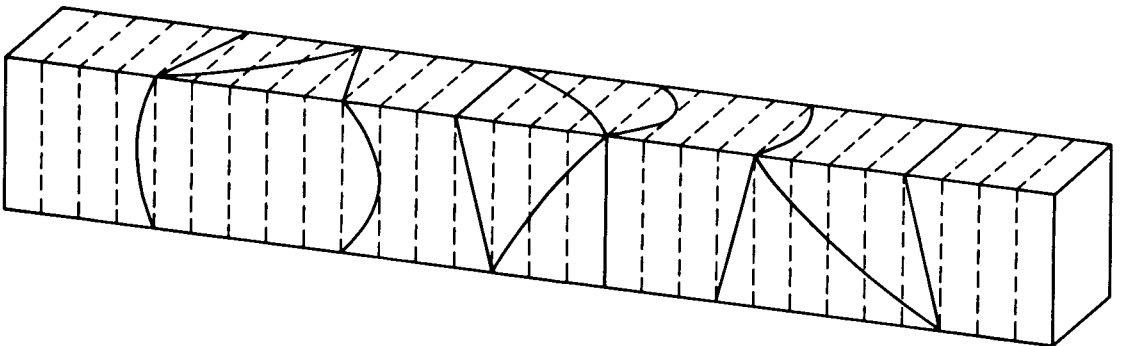


Figure 7-8 Results of the three-dimensional bar
example with varying element shapes

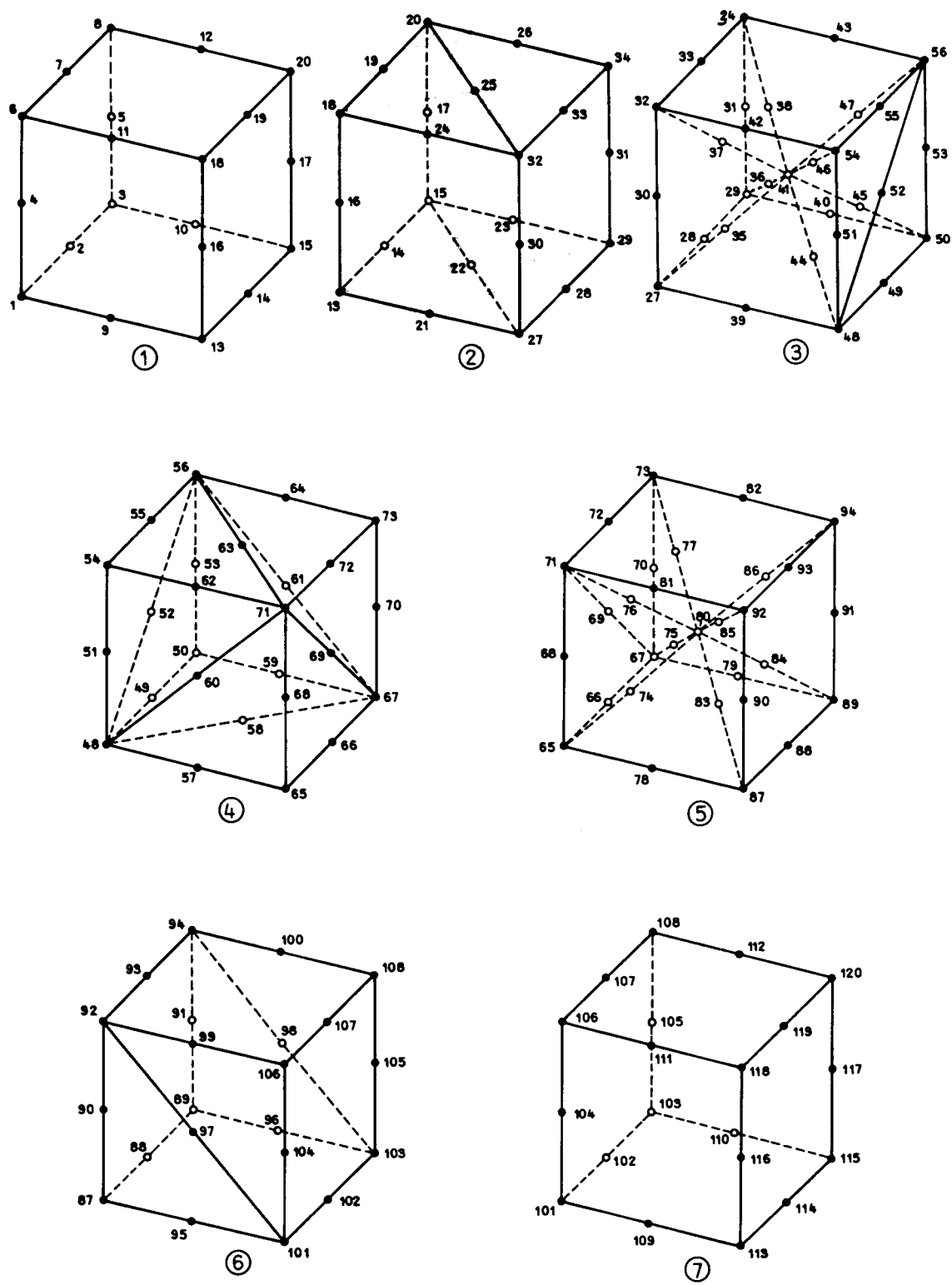


Figure 7-9 Structure of the blocks used in Figures 7-8a, b, c

Pyramids are pinched-sided elements, as they are derived from 20-node bricks by reducing 8 nodes of a face to only one point. As a consequence, the triangular faces of a pyramid will not be "true" triangles, but pinched-sided rectangles, which should be assembled to the same type of face of another element. As tetrahedrons have "true" triangular faces, the assembly pyramide-tetrahedron is not quite compatible, which leads to the above mentioned errors.

This incompatibility is, however, easily corrected by using a "pinched-sided" tetrahedron, derived from a triangular prism by reducing one of the triangular faces to a point. The new results are presented in Figure 7-8b (equipotentials). Comparison of the calculated heads with the x nodal coordinates indicates that the new solution is perfect.

The solution does not change if we use curved-sided elements in the constant section bar (see Figure 7-8c). The results are presented in Figure 7-8c (equipotentials). The calculated nodal heads are again the same as the x coordinates of the nodal points.

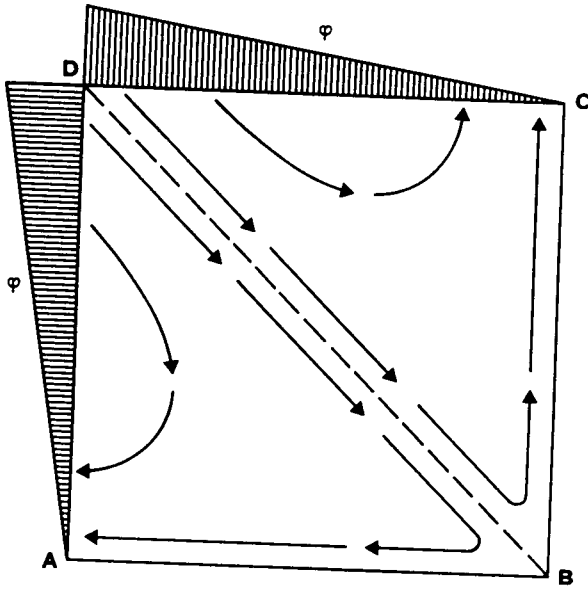
7.3.2 Too large elements

The behaviour of too large elements will be demonstrated by a simple two dimensional example. Figure 7-10 represents a simple 2 km by 2 km square region where the following boundary conditions are defined:

- zero normal flux on side AB and BC (no-flow boundary)
- linearly varying head on side AD and DC: $h = 2 \text{ m}$ at corner D, and $h = 0 \text{ m}$ at corners A and C.

A permeability of $K = 1 \text{ m/s}$ is given to the entire flow region.

Evidently enough, the true solution will show the existence of two flow systems separated by the diagonal D-B. The "exact" potential distribution is shown by Figure 7-11a, obtained by using 100 8-node elements for the entire flow region. Figure 7-11b represents the solution for the same problem, obtained by using only one 8-node element to simulate the potential field. Obviously, the element is too large (or, the quadratic function is too simple) to represent the true solution, and this leads to the appearance of unrealistic features in the potential field of Figure 7-11b:



$$\varphi_A = \varphi_C = 0 \quad [\text{m}]$$

$$\varphi_D = 2 \quad [\text{m}]$$

$$K = 1 \quad [\text{m/s}]$$

$$q_n = 0 \quad \text{on AB and BC.}$$

Figure 7-10: Boundary conditions for large element example

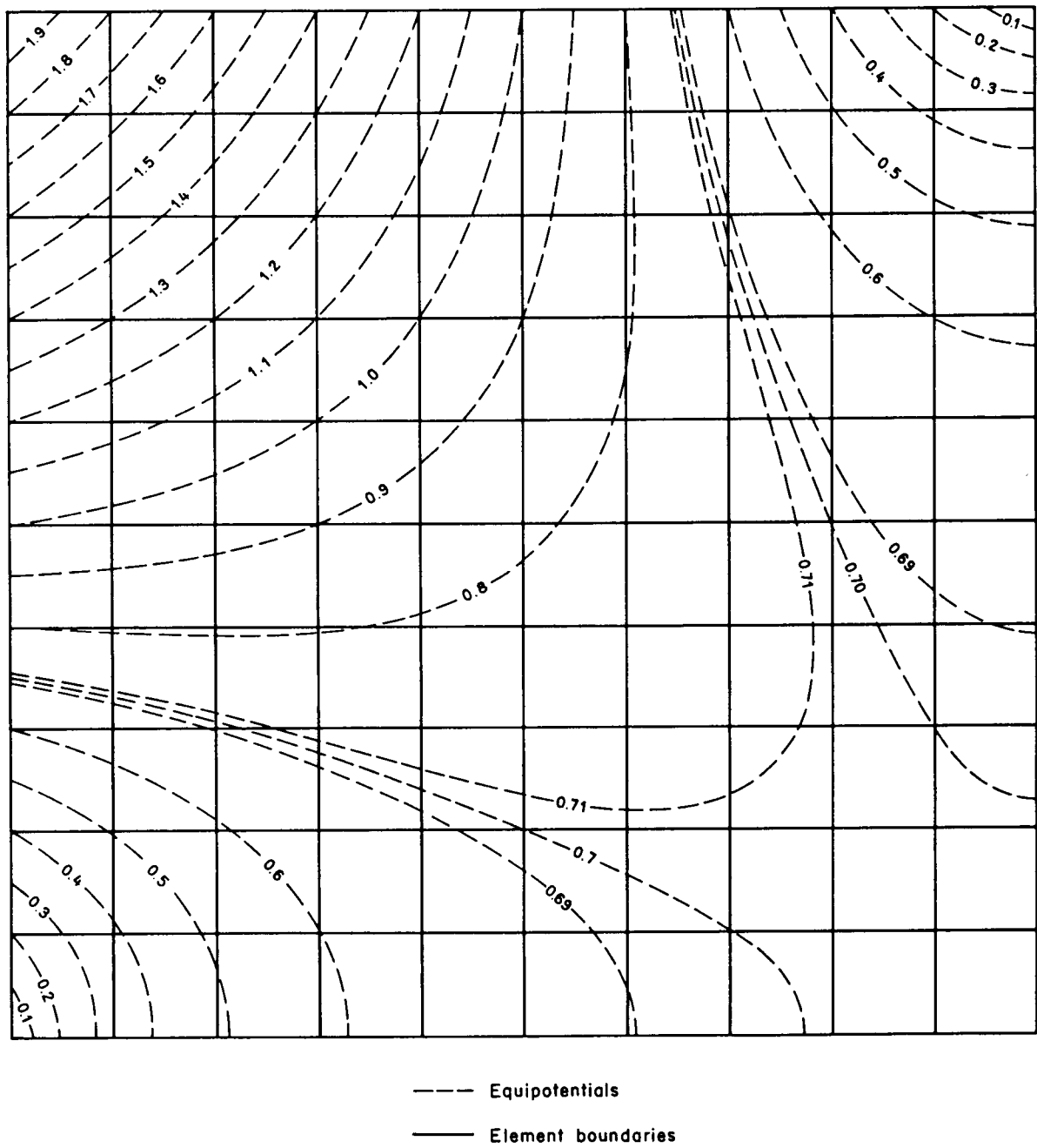


Figure 7-11: Results of large element example
a) "Exact" solution

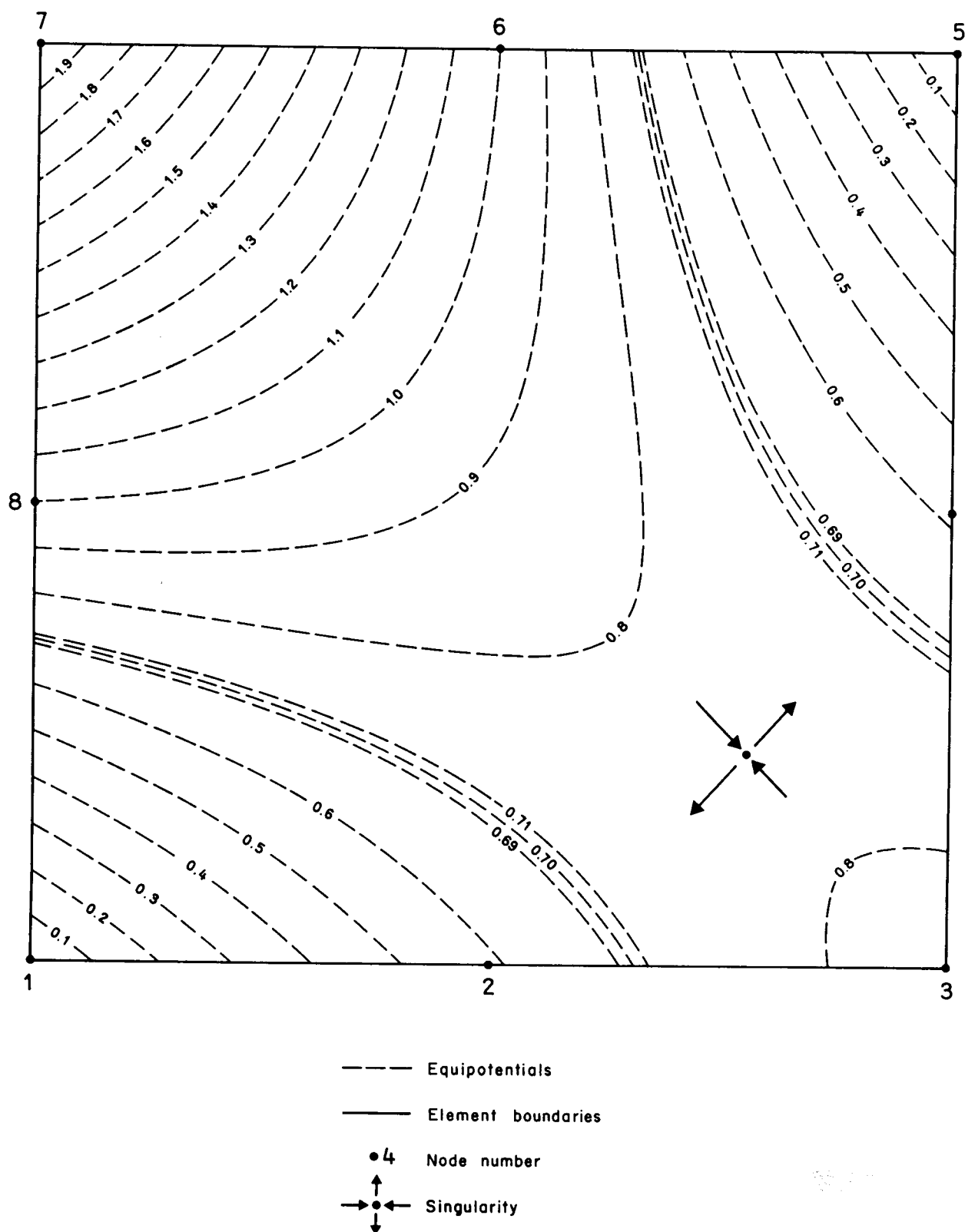


Figure 7-11: Results of large element example
b) Single element solution

```
*****
* "FEM200" - MODELE A ELEMENTS FINIS *
* ECOULEMENT PERMANENT - BIDI *
* CENTRE D'HYDROGEOLOGIE (UNIVERSITE DE NEUCHATEL) *
*****
```

PARAMETRES

```
MNNIC= 1      MXNIC= 8      MXNQ= 1      MXNP= 1
MXLM= 1      NDACT= 8      MXPA= 8
```

VALEURS DES CLASSE DE PERM/PORO

```
1      0.1000E+01      0.1000E+01      0.0
```

ALIMENTATIONS DISTRIBUEES

```
1      0.0000000E+00
```

CONDITIONS NODALES

```
1      1      -1      0.0000000E+00
2      5      -1      0.0000000E+00
3      6      -1      0.1000000E+01
4      7      -1      0.2000000E+01
5      8      -1      0.1000000E+01
```

!RESULTATS CALCULES!

```
1 -1  0.0000 -0.697      2  0  0.5843  0.000      3  0  0.8315  0.000
4  0  0.5843  0.000      5 -1  0.0000 -0.697      6 -1  1.0000  0.449
7 -1  2.0000  0.494      8 -1  1.0000  0.449
```

SOMME DES DEBITS = -0.1387779E-16

Figure 7-12: Result file for single element solution

- a strong singularity appears on the diagonal D-B
- there are 4 flow systems (2 larger and 2 smaller) in the element
- on side AB and BC the gradients are not parallel to the no-flow boundaries, thus there will be a non-zero flux across these sides (in and out).

Integrating the inward and outward normal fluxes over the 4 element sides does not give zero, but a negative value, i.e. more water comes out than goes in across the element sides.

Apparently the mass conservation principle is violated, but only apparently. As a matter of fact, the excess water flowing out of the element simply indicates that the integral of the divergence of the flux is not zero in the interior of the element, but has a positive value, i.e. water is "created" in the element. This is naturally unrealistic (in the true solution $\text{div}(-K \nabla h) = 0$ in the interior of the flow region), but in spite of such unrealistic features the quadratic function, representing the potential field, satisfies the Galerkin finite element formulation given in chapter 3, and does not violate the mass conservation principle.

It is interesting to mention that the nodal discharges of nodes 2, 3 and 4 are zero, indicating that the amount of water "created" in, and the amount of water flowing out, of the region of influence of these nodal points sum up to zero. It is even more interesting to mention that the total discharge (given by the result file) is $0,697 \text{ m}^3/\text{s}$ on boundary C-D if we simulate the problem by only one element, and the total discharge is $0,593 \text{ m}^3/\text{s}$ on the same boundary if we simulate the flow by 100 8-node elements! So, even if the behaviour of the large element is locally unrealistic, the global results represent a useful approximation of the real system.

8. REFERENCES

- /1/ Bear, J., 1972: Dynamics of Fluids in Porous Media, Elsevier, New York, 764 p.
- /2/ Bear, J., 1979: Hydraulics of Groundwater. Mc-Graw Hill, New York, 569 p.
- /3/ Dagan, G., 1972: Some aspects of heat and mass transfer in porous media.
in: Fundamentals of Transport phenomena in porous media, Elsevier, New York, 764 p.
- /4/ De Marsily, G., 1981: Hydrogéologie quantitative, Masson, Paris, 215 p.
- /5/ Desai, C.S., and Abel, J.F, 1972. Introduction to the finite element method. Van Nostrand Reinhold, London, 477 p.
- /6/ De Wiest, R.S.M, 1965. Geohydrology, Wiley, New York, 412 p.
- /7/ Hubbert, M.K., 1940: The theory of ground-water motion, Journal Geology, 48, 785 - 944 p.
- /8/ Hubbert, M.K., 1957: Darcy law and the field equation of flow of underground fluids, Bull. A.J.H.S., No. 5, 23 - 59 p.
- /9/ Hydrologisch Colloquium, 1964: Steady flow of groundwater towards wells, Committee for Hydrological Research, T.N.O., Pub. 10, 179 p.
- /10/ INTERA Environmental Consultants, Inc., 1983: SWENT: A Three-dimensional Finite Difference code for the Simulation of Waste, Energy, and Nuclide Transport, ONWI-457, Office of Nuclear Waste Isolation, Columbus, OH.

- /11/ Irons, B.M., 1970: A frontal solution program for finite element analysis. Int. J. Num. Meth. Eng., Vol. 2, 5 - 32 p.
- /12/ Kimmeier, F., Perrochet, P., Andrews, R., Kiraly, L., 1985: Simulation des Ecoulements Souterrains entre les Alpes et la Forêt Noire par Modèle Mathématique, NAGRA NTB 84-50.
- /13/ Kiraly, L., 1979: Remarques sur la simulation des failles et du réseau karstique par éléments finis dans les modèles d'écoulement. Bull. du Centre d'Hydrogéologie Uni. Neuchâtel, No. 3, 155 - 167 p.
- /14/ Klingbeil, E., 1966: Tensorrechnung für Ingenieure. Bibl. Inst. Mannheim, 194 p.
- /15/ Pinder, G.F. and Gray, W.G., 1977: Finite Element Simulation in Surface and Sub-surface Hydrology, Academic Press, New York, 295 pp.
- /16/ Scheid, F., 1968: Numerical Analysis, Mc Graw-Hill, New York, 422 p.
- /17/ Schwarz, H.R., 1980: Methode der finiten Elemente. B.G. Teubner, Stuttgart, 320 p.
- /18/ Segerlind, L.J., 1976: Applied finite element analysis. Wiley, New York
- /19/ Teichmann, H., 1964: Physikalische Anwendungen der Vektor- und Tensorrechnung. Bibl. Inst. Mannheim, 231 p.
- /20/ Zienkiewicz, O.C., 1971: The finite element method in engineering science. Mc Graw-Hill, London, 521 p.

ANNEX A1: NOMENCLATURE

a_k	Covariant base vectors ($= \partial r / \partial s^k$)
b	Half band-width of assembled coefficient matrix
$\vec{e}_1, \vec{e}_2, \vec{e}_3$	Global unit vectors
$f(y)$	Polynomial function depending only on y
$\vec{g}$	Gravitational acceleration (m/s^2)
g^{ik}	Contravariant tensor = $a_i \cdot a_k^{-1}$
g_{ik}	Covariant tensor = $a_i \cdot a_k$
h_1, h_2, h_3	Hydraulic potentials 1. Energy potential per unit volume water (Kg/ms^2) 2. Energy potential per unit mass water (m^2/s^2) 3. Energy potential per unit weight of water (head) (m)
i, j	Indices
$\frac{=}{k}$	Intrinsic permeability (m^2)
m, n	Nodal indices
$\vec{n}$	Normal vector to boundary S2
p, q, r	Number of Gauss points in the s, t, n directions
$\vec{q}$	Darcy velocity or flux (m^3/sm^2)
$\vec{r}$	Pointer vector in global space
s^i	Local coordinates
s, t, u	Local coordinates ($= s^1, s^2, s^3$)
x^1, x^2, x^3	Global coordinate directions

y_i	Local coordinates of Gauss integration points
z	Elevation above datum (m)
A_{nm}, C_{nm}, Q_{Tm}	Coefficient matrixes for each element
B_{in}	Gradient of the interpolation functions in global space = $\partial N_n / \partial x^i$
D_i^k	Transformation matrix from local to global derivatives = $\partial x^k / \partial s^i$
F	$\text{grad } p - \rho g$ Groundwater driving force ($\text{kg/m}^2 \text{s}^2$)
GM_k	Process contributions
GT_k	Source/sink terms in right-hand side
H	Prescribed head on boundary S1
J	Hydraulic gradient
JL^k	Contravariant component of gradient vector
K	$\bar{k} \rho g / \mu$ Hydraulic conductivity or permeability (m/s)
L^j	Local natural or volume coordinates
$L(h)$	Differential operator $= S \frac{\partial h}{\partial t} + \frac{\partial}{\partial x^i} (-K^{ij} \frac{\partial h}{\partial x^j}) + Q$
M_{ik}	Assembled "grand" coefficient matrix for all elements
N_n	Interpolation functions
P	Fluid pressure (kg/ms^2)
Q_k	Assembled source/sink terms for all elements
Q	Volume source/sink term (m^3/sm^3)
QS	Normal flux through boundary S2
R	Residual from approximating h by h^* $L(h^*) = R$

S	Volume storativity = gS^m (1/m)
S^m	Mass storativity = $\left[\rho(1-\epsilon)\alpha + \epsilon\beta \right]$ (s^2/m^2)
V	Volume of integration
W	Weighting functions (equal to N in the Galerkin method)
α	Compressibility of porous medium ($ms^2/kg = 1/P_a$)
β	Compressibility of water ($ms^2/kg = 1/P_a$)
ϵ	Porosity of porous medium (-)
ν	Kinematic viscosity ($= \mu/\rho$) (m^2/s)
ρ	Fluid density (kg/m^3)
μ	Dynamic viscosity (kg/ms)

CGR	Symmetric matrix containing the active coefficients of M_{ik} in core
EQ	Number of equations
KR	Number of nodal points per element
KURPA	Length of equations in CGR
LDEST	Address of variable LVB in the CGR matrix
LM	Number of element
LSYMB	Symboloc elimination routine
LVB	Node numbers for each assembled element
MVBL	Pointer vector indicating the equations occupying successive columns in CGR (i.e. the active node numbers)
NA	Number of terms in each equation = front width
NC	Number of coefficients required in core for solution routine $= \frac{b^2 - b}{2} + b \text{ in Gauss}$ $= \frac{NA^2 - NA}{2} + NA \text{ in frontal}$
NLM	Total number of elements
NP	Number of nodal points in an element
RHS	Sum of $GM_k + GT_k$ prior to elimination

ANNEX A2: Variable names used in program FEM 301.

Only the most important variable names will be defined here, and only if they are not yet defined in the listing itself. As the subroutine names and the file names are explained in section 5.3., these definitions will not be repeated here.

- (AL,BL,CL,DL): surface or volume local coordinates for triangle and tetrahedron.
- ALM: region of influence (length, surface or volume) for each node of a particular element.
- BX, BY, BZ: lines of the "gradient matrix" (derivatives of the interpolation functions with respect to global coordinates x, y and z).
- CEQ: coefficients of an eliminated equation (stored on file FIEQU)
- CLM: element coefficient matrix. Will be assembled in CGR to form the equations.
- CGR: "grand" coefficient matrix in core during assembly and elimination of the equations.
- DTJ: determinant of the Jacobian matrix, or square root of the determinant of the covariant metric tensor.
- FACX, FACY, FACZ: scaling factors for the x,y,z global coordinates.
- FN: vector of the interpolation functions for a given point in an element.
- FNDS, FNDD, FNDU: derivatives of the interpolation functions with respect to the local coordinates for a given point in an element.
- H: nodal values of the hydraulic head.
- ID: auxiliary vector indicating the active variables, type of boundary conditions, etc.
- IVB: auxiliary vector in SYM301, to check the repetition of the node numbers in an element.
- KOUNT: counts the number of appearances of a variable, in thousands, plus 1000 (in SYM301).
- KR: number of nodal points in an element.
- KURPA: is the current size of MVB, i.e. the number of active variables or the length of an eliminated equation (subroutine SLV301).
- LAS, LAST: the last appearance of a variable is in NIX(LAST). A provisional value of LAST is LAS (in subroutine SYM301).
- LDES, LDEST: the element "destinations" are in LDEST (coded information about appearance and adress of a variable in CGR). LDES is the decoded version of LDEST(K). See subroutine CDEST in Annex 3.
- LVB: vector of the node numbers for a given element.
- MVB: vector of the active variables, i.e. those in CGR or in the running variables RHS used in the back-substitution.
- MXLM: number of elements
- MXND: number of nodal points in the model.
- MXNIC, MNNIC: highest and lowest node number.
- MXNP, MXNQ: highest permeability class number and infiltration class number.
- MXPA: is the maximum size of CGR required, in terms of the number of variables (maximum frontwidth).
- NAR: number of element sides or element edges.
- NEW = N1 to NZ in SYM301 ranges over an element in NIX.
- NFUNC: is a function giving the position of a term (I,J) in a vector representing an upper triangular matrix.
- NP, NQ: permeability class and infiltration class for an element.

NIX: main working area in SYM301. It starts as a list of all LVB-s for successive elements, and it becomes the list of all LDEST vectors.

NIZZ: the last label is NIX(NIZZ).

NSTR: number of appearances of a variable. See comments in CDEST.

NVABZ: the same as MXND.

PIVOT: diagonal term in CGR.

PER: list of all permeability matrixes.

PERM: permeability matrix for an element.

Q: nodal discharge or recharge values.

QD: list of all infiltration rates.

RHS: running variable in SLV301 (right hand sides on elimination, heads for the active variables on back-substitution).

RHSEL: eliminated right hand side term for a given equation.

S, T, U: local coordinates for a given point in an element.

X, Y, Z: list of all global coordinates.

XN, YN, ZN: global coordinates for the nodal points of a particular element.

```

PROGRAM FEM301
REAL*8 CGR,H,Q,QSUM
REAL*8 X,Y,Z,PER,QD
CHARACTER*80 TITR,LINE
CHARACTER*50 FIELM,FICOR,FIMAT,FIEQU,FIRES,FIPAR,FIDST,NAM
CHARACTER*60 HD1,HD2,HD3,HD4
PARAMETER (IV1=99999,IV2=10000,MXFR=770)
DIMENSION CGR(300000),NIX(200000),X(IV1),Y(IV1),Z(IV1)
DIMENSION PER(6,600),POR(600),QD(600),ID(IV1),H(IV1)
DIMENSION Q(IV1),JQ(IV2),JP(IV2),LMZ(IV2)
EQUIVALENCE (X(1),CGR(1),NIX(1)),(Y(1),CGR(100001)),
1      (Z(1),CGR(200001))
DATA NGAUSS,KTRAN,INDST/3,0,17/
DATA INELM,INCOR,INPAR,INMAT,INEQU,INRES/11,12,13,14,15,16/
DATA HD1/' *****'/
DATA HD2/' * "FEM301" - MODELE A ELEMENTS FINIS */
DATA HD3/' * ECOULEMENT PERMANENT - TRIDI */
DATA HD4/' * CENTRE D'HYDROGEOLOGIE (UNIVERSITE DE NEUCHATEL) */

C      *****
C      * MODELE TRIDI - PERMANENT *
C      *****

680  FORMAT(/5(4X,A60/))
700  FORMAT(A)

OPEN(UNIT=INELM,TYPE='OLD')
OPEN(UNIT=INCOR,TYPE='OLD')
OPEN(UNIT=INPAR,TYPE='OLD')
OPEN(UNIT=INRES,TYPE='NEW',FORM='FORMATTED')
INQUIRE(UNIT=INRES,NAME=FIRES)
I1=INDEX(FIRES,'J')+1
I2=INDEX(FIRES(I1:),'.')+I1-1
FIMAT=FIRES(I1:I2)//'MAT'
FIDST=FIRES(I1:I2)//'DST'
OPEN(UNIT=INDST,NAME=FIDST,TYPE='SCRATCH',
1  FORM='UNFORMATTED',ACCESS='SEQUENTIAL')

702  FORMAT(Q,A)
WRITE(INRES,680)HD1,HD2,HD3,HD4,HD1

INQUIRE(UNIT=INELM,NAME=NAM)
WRITE(INRES,800)NAM
800  FORMAT(/4X,' FICHIERS DES DONNEES: '/
1      4X,' ***** ',A50)
INQUIRE(UNIT=INCOR,NAME=NAM)
WRITE(INRES,805)NAM
805  FORMAT(28X,A50)
INQUIRE(UNIT=INPAR,NAME=NAM)
WRITE(INRES,805)NAM
INQUIRE(UNIT=INRES,NAME=NAM)
WRITE(INRES,810)NAM
810  FORMAT(4X,' FICHIER DE SORTIE: '/
1      4X,' ***** ',A50//)

C      -----
1  CALL SYM301(MXLM,MNNIC,MXNIC,MXNQ,MXNP,NVABZ,MXPA,NIX,
C      JQ,JP,LMZ,ID,ISTOP)
C      -----
WRITE(INRES,815)
815  FORMAT(4X,' PARAMETRES DE CONTROLE: '/

```

```

1      4X, ' *****' )
WRITE(INRES,706)MNNIC,MXNIC,MXNQ,MXNP
706   FORMAT(4X, ' MNNIC=' ,I5,5X, 'MXNIC=' ,I5,5X, 'MXNQ=' ,I3,5X, 'MXNP=' ,I3)
WRITE(INRES,708)MXLM,NVABZ,MXPA
708   FORMAT(4X, ' MXLM=' ,I5,5X, 'NDACT=' ,I5,5X, 'MXPA=' ,I3/)
IF(MXPA.GT.MXFR) THEN
    WRITE(INRES,711)MXFR
711   FORMAT('/ LE FRONT DEPASSE ',I4)
    ISTOP=ISTOP+1
END IF
IF(ISTOP.NE.0) THEN
    WRITE(INRES,812)
812   FORMAT('/ CALCUL INTERROMPU APRES L'ELIMINATION SYMBOLIQUE')
    STOP
END IF
C -----
CALL LECCOR(INCOR,INRES,MXNIC,X,Y,Z,ID)
CALL LECPE(INPAR,INRES,PER)
OPEN(UNIT=INMAT,NAME=FIMAT,TYPE='SCRATCH',
1     FORM='UNFORMATTED',ACCESS='SEQUENTIAL')
C -----
CALL TLM4(X,Y,Z,PER,ISTOP)
C -----
IF(ISTOP.NE.0) THEN
    STOP
END IF
REWIND INMAT
C -----
CALL LECAL(INPAR,INRES,QD)
CALL LECOND(INPAR,INRES,H,Q,ID)
C -----
C Calcul de INITIALSIZE pour le fichier des equations:
A=32.0+8.0*FLOAT(MXPA)
A=(A/512.0)*FLOAT(NVABZ)
INITSZ=IFIX(A)+1
OPEN(UNIT=INEQU,TYPE='NEW',
1   ASSOCIATEVARIABLE=IN15,RECORDTYPE='FIXED',
2   ACCESS='DIRECT',MAXREC=NVABZ,RECORDSIZE=(2*MXPA+8),
3   INITIALSIZE=INITSZ)
INQUIRE(UNIT=INEQU,NAME=FIEQU)
C -----
CALL SLV301(MXPA,H,Q,ID,CGR,QD,INMAT,INEQU,QSUM)
WRITE(INRES,2002)FIEQU
2002  FORMAT(4X, ' FICHER DES EQUATIONS: ',A)
C -----
WRITE(INRES,730)
730   FORMAT('//4X, ' !RESULTATS CALCULES!'/5X,20(1H*))
MXND=0
K=1
DO WHILE (K.LE.MXNIC)
    IDI=ID(K)
    IF(IDI.NE.0) THEN
        MXND=MXND+1
        H(MXND)=H(K)
        Q(MXND)=Q(K)
        IF(IDI.GE.1001) THEN
            IDI=0
        END IF
        ID(MXND)=IDI
        NIX(MXND)=K
    END IF

```

```
      END IF
      K=K+1
    END DO
    WRITE(INRES,732)(NIX(K),ID(K),H(K),Q(K),K=1,MXND)
732   FORMAT(3(4X,I6,I3,F8.2,1X,E10.3))

    WRITE(INRES,734)QSUM
734   FORMAT(//4X,' SOMME DES DEBITS = ',E14.7)
      CLOSE(UNIT=INRES)

    STOP
  END
```

```

      SUBROUTINE SYM301(MXLM,MNNIC,MXNIC,MXNQ,MXNP,NVABZ,MXPA,
1          NIX,JQ,JP,LMZ,ID,ISTOP)

      CHARACTER*80 LINE
      DIMENSION LVB(27),LDEST(27),MVB(999),IVB(27)
      DIMENSION NIX(1),LMZ(1),JQ(1),JP(1),ID(1)
      DATA INELM,INRES,INDST/11,16,17/

C      Symbolic elimination procedure.
C      -----

      8  READ(INELM,700)LINE
700    FORMAT(A)
      IF(INDEX(LINE,'ELEMENTS').EQ.0)GO TO 8

      MXLM=0
      MNNIC=100000
      MXNIC=0
      MXNQ=0
      MXNP=0
      NVABZ=0
      MXPA=1
      NIZZ=0
      ISTOP=0

C      Input of the element file:
12     READ(INELM,*,END=16,ERR=500)LM,NQ,NP,KR,NAR,(LVB(K),K=1,KR)
      MXLM=MXLM+1
      DO 14 K=1,KR
        NIC=LVB(K)
        NIZZ=NIZZ+1
        NIX(NIZZ)=-NIC
        ID(NIC)=ID(NIC)+1
        IF(NIC.LT.MNNIC) MNNIC=NIC
        IF(NIC.GT.MXNIC) MXNIC=NIC
      -----
C      If NAR.GE.0: repetition of a node number in the same
C      element is not allowed.
C      If NAR.LT.0: nod numbers may be repeted in the same ele-
C      ment (elem. edges may be reduced to a point)
      IF(NAR.GE.0.AND.IVB(K).EQ.0) THEN
        KK=K+1
        IREP=1
        DO WHILE (KK.LE.KR)
          IF(NIC.EQ.LVB(KK)) THEN
            IVB(KK)=1
            IREP=IREP+1
          END IF
          KK=KK+1
        END DO
        IF(IREP.NE.1) THEN
          WRITE(INRES,710)LM,NIC,IREP
710      FORMAT(' LM=',I5,' NOD=',I6,' APPARAIT ',I2,' FOIS'/)
          IVB(K)=0
          ISTOP=ISTOP+1
        END IF
      END IF

C      -----
14     CONTINUE
      JQ(MXLM)=NQ

```

```

      JP(MXLM)=NP
      IF(NQ.GT.MXNQ) MXNQ=NQ
      IF(NP.GT.MXNP) MXNP=NP
      LMZ(MXLM)=NIZZ ! last node of an elem. in NIX
GO TO 12
16  CONTINUE
    WRITE(INDST)MXLM

C    The updated MVB vector is used to create the LDEST vectors
C    for each element:
      N1=1
      DO 50 LM=1,MXLM
        NZ=LMZ(LM)
        KR=NZ-N1+1
        DO 40 NEW=N1,NZ
          NIC=NIX(NEW)          ! if NIX(NEW).GT.0: the variable is
          LDES=NIC              ! already in MVB(LDES)
          IF(NIC.GT.0)GO TO 38 ! if not: first appearance
          DO 30 LDES=1,MXPA      ! MXPA is the current length of MVB
            IF(MVB(LDES).EQ.0)GO TO 32 ! free place in MBV?
30          CONTINUE
            LDES=MXPA+1          ! if not: increase the current size
            MXPA=LDES            ! of MVB
32          CONTINUE
            MVB(LDES)=NIC
            KOUNT=1000
            DO 36 LAS=NEW,NIZZ    ! scanning all elements
              IF(NIX(LAS).NE.NIC)GO TO 34 ! for last appearance
              NIX(LAS)=LDES
              KOUNT=KOUNT+1000      ! counting appearances
              LAST=LAS
34          CONTINUE
36          CONTINUE
            NIX(LAST)=LDES+1000 ! code for the last appearance
            NIX(NEW)=LDES+KOUNT ! code for the first appearance
            LDES=LDES+KOUNT
38          CONTINUE
            IND=NEW-N1+1
            LDEST(IND)=LDES ! coded information is in LDEST
40          CONTINUE

      N1=NZ+1
      DO 44 K=1,KR
        CALL CDEST(LDEST,LDES,NSTR,K) ! decoding LDEST
        NIC=-MVB(LDES)
        LVB(K)=NIC
        IF(NSTR.NE.0.AND.NSTR.NE.1)GO TO 42 ! last appearance?
        MVB(LDES)=0 ! if yes: NIC is eliminated from MVB
        NVABZ=NVABZ+1
42      CONTINUE
44      CONTINUE
        WRITE(INDST)JQ(LM),JP(LM),KR,(LVB(K),K=1,KR),(LDEST(K),K=1,KR)
50      CONTINUE
      REWIND INDST
      RETURN

500  TYPE *, 'ERREUR DANS LA LECTURE DES ELEMENTS!'
      ISTOP=100
      RETURN
      END

```

```
SUBROUTINE CDEST(LDEST,LDES,NSTR,KL)
PARAMETER (M1=1,M2=2,M3=1000)
DIMENSION LDEST(M1)
```

```
C -----
C Programme de decodage des adresses pour la methode frontale
C de IRONS(1970). La valeur de LDES donne le numero de la co-
C lonne et de la ligne dans la matrice d'assemblage CGR ou il
C faut placer le coefficient du noeud LVB(KL).
C La valeur de NSTR montre s'il s'agit de la premiere appari-
C tion, de la derniere apparition ou d'une apparition inter-
C mediaire du noeud:
C NSTR=-1: apparition intermediaire
C NSTR= 0: derniere apparition (et elimination!)
C NSTR= 1: premiere et derniere apparition (elimination!)
C NSTR= N: premiere des N apparitions
C -----
C LDES=LDEST(KL)
C NSTR=M1
C DO WHILE (LDES.GE.M3)
C   LDES=LDES-M3
C   NSTR=NSTR+M1
C END DO
C NSTR=NSTR-M2
C END
```

```

SUBROUTINE LECPE(INPAR,INRES,PER)
CHARACTER LINE*80
REAL*8 PER,PERM,ZER
PARAMETER (ZER=0.0D+0)
DIMENSION PER(6,1),PERM(6)

```

C Input of the permeabilities [K]=(k11,k12,k22,k13,k23,k33)
C -----

```

WRITE(INRES,712)
712  FORMAT(/5X,'VALEURS DES CLASSES DE PERM/PORO'/5X,32(1H*))

DO 10 I=1,6
    PERM(I)=ZER
10  CONTINUE

20  READ(INPAR,700)LINE
700  FORMAT(A)
    IF(INDEX(LINE,'PERMEABILITES').EQ.0)GO TO 20

READ(INPAR,*,IOSTAT=IOS)NP,(PERM(I),I=1,6)
DO WHILE (IOS.EQ.0.AND.NP.GE.1)
    IF(PERM(3).EQ.ZER.AND.PERM(6).EQ.ZER) THEN
        PERM(3)=PERM(1)
        PERM(6)=PERM(1)
        WRITE(INRES,714)NP,PERM(1)
714  FORMAT(5X,I5,5X,E14.5)
    ELSE
        WRITE(INRES,716)NP,PERM(1),PERM(2),PERM(4),
1      PERM(3),PERM(5),PERM(6)
716  FORMAT(5X,I5,5X,3E14.5/29X,2E14.5/43X,E14.5)
    END IF
    DO 30 I=1,6
        PER(I,NP)=PERM(I)
        PERM(I)=ZER
30  CONTINUE
    READ(INPAR,*,IOSTAT=IOS)NP,PERM
END DO
REWIND INPAR
END

```

```

SUBROUTINE LECCOR(INCOR,INRES,MXNIC,X,Y,Z,ID)
CHARACTER LINE*80
REAL*8 X,Y,Z,XX,YY,ZZ,FACX,FACY,FACZ
DIMENSION X(1),Y(1),Z(1),ID(1)

```

C Input of the global coordinates X,Y,Z.
C -----

```

10  READ(INCOR,700)LINE
700  FORMAT(A)
    IF(INDEX(LINE,'COORDONNEES').EQ.0)GO TO 10

READ(INCOR,*)FACX,FACY,FACZ
READ(INCOR,*,IOSTAT=IOS)NIC,XX,YY,ZZ
DO WHILE (IOS.EQ.0.AND.NIC.GE.1)
    IF(ID(NIC).GE.1) THEN

```

```

      X(NIC)=XX*FACX
      Y(NIC)=YY*FACY
      Z(NIC)=ZZ*FACZ
      ID(NIC)=ID(NIC)+1000
    END IF
    READ(INCOR,*,IOSTAT=IOS)NIC,XX,YY,ZZ
  END DO

  DO 18 NIC=1,MXNIC
    IF(ID(NIC).NE.0.AND.ID(NIC).LT.1001) THEN
      WRITE(INRES,710)NIC
710    FORMAT(4X,' LE NOEUD',I6,'N'-'A PAS DE COORDONNEES!')
      ISTOP=100
    END IF
18    CONTINUE
    IF(ISTOP.NE.0) THEN
      STOP
    END IF
    CLOSE(UNIT=INCOR)
  END

  SUBROUTINE LECAL(INPAR,INRES,QD)
  CHARACTER*80 LINE
  REAL*8 QD,QDIS
  DIMENSION QD(1)

C    Input of the distributed infiltration rates.
C    -----

    WRITE(INRES,716)
716    FORMAT(/4X,' ALIMENTATIONS DISTRIBUEES'/4X,30(1H*))

    32    READ(INPAR,700)LINE
700    FORMAT(A)
    IF(INDEX(LINE,'ALIM').EQ.0)GO TO 32

    READ(INPAR,*,IOSTAT=IOS)NQ,QDIS
    DO WHILE (IOS.EQ.0.AND.NQ.GE.1)
      QD(NQ)=QDIS
      WRITE(INRES,717)NQ,QDIS
717    FORMAT(5X,I5,5X,E14.7)
      READ(INPAR,*,IOSTAT=IOS)NQ,QDIS
    END DO
    REWIND INPAR
  END

```

```

SUBROUTINE LECOND(INPAR,INRES,H,Q,ID)
CHARACTER LINE*80
REAL*8 H,Q,VAL
DIMENSION H(1),Q(1),ID(1)

```

```

C      Input of the nodal boundary conditions.
C      -----

```

```

WRITE(INRES,718)
718  FORMAT(/4X,' CONDITIONS NODALES'/4X,20(1H*))
MXCON=0

42   READ(INPAR,700)LINE
700  FORMAT(A)
IF(INDEX(LINE,'CONDITIONS').EQ.0)GO TO 42
READ(INPAR,*,IOSTAT=IOS)NIC,IDI,VAL
DO WHILE (IOS.EQ.0.AND.NIC.GE.1)
  IF(IDI.LT.0) THEN
    H(NIC)=VAL
  ELSE
    Q(NIC)=VAL
  END IF
  ID(NIC)=IDI
  MXCON=MXCON+1
  WRITE(INRES,720)MXCON,NIC,IDI,VAL
720  FORMAT(4X,I5,5X,I6,1X,I2,5X,E14.7)
  READ(INPAR,*,IOSTAT=IOS)NIC,IDI,VAL
END DO
CLOSE(UNIT=INPAR)
END

```

```

SUBROUTINE TLMAK4(X,Y,Z,PER,ISTOP)
REAL*8 ALM,CLM,SUM
REAL*8 XN,YN,ZN,X,Y,Z,PER,PERM
DIMENSION XN(27),YN(27),ZN(27),ALM(27),LVB(27),LDEST(27)
DIMENSION CLM(380),X(1),Y(1),Z(1)
DIMENSION PER(6,1),PERM(6)
DATA INMAT,INDST,INRES/14,17,16/

C      CALCUL DES MATRICES ELEMENTAIRES
C      *****

ISTOP=0
READ(INDST)MXLM
WRITE(INMAT)MXLM
DO 100 LM=1,MXLM
  READ(INDST)NQ,NP,KR,(LVB(K),K=1,KR),(LDEST(K),K=1,KR)
  DO 10 I=1,6
    PERM(I)=PER(I,NP)
10  CONTINUE
    LZ=(KR*KR-KR)/2+KR
    DO 40 K=1,KR
      NIC=LVB(K)
      XN(K)=X(NIC)
      YN(K)=Y(NIC)
      ZN(K)=Z(NIC)
      ALM(K)=0.0D+0
40  CONTINUE
      DO 42 K=1,LZ
        CLM(K)=0.0D+0
42  CONTINUE

      IF(KR.EQ.3)THEN
        CALL TLM23(KR,XN,YN,ZN,CLM,ALM,PERM)
      ELSE IF(KR.EQ.6)THEN
        CALL TLM36(KR,XN,YN,ZN,CLM,ALM,PERM)
      ELSE IF(KR.EQ.8)THEN
        CALL TLM48(KR,XN,YN,ZN,CLM,ALM,PERM)
      ELSE IF(KR.EQ.9)THEN
        CALL TLM49(KR,XN,YN,ZN,CLM,ALM,PERM)
      ELSE IF(KR.EQ.10)THEN
        CALL TLM410(KR,XN,YN,ZN,CLM,ALM,PERM)
      ELSE IF(KR.EQ.13)THEN
        CALL TLM513(KR,XN,YN,ZN,CLM,ALM,PERM)
      ELSE IF(KR.EQ.15)THEN
        CALL TLM615(KR,XN,YN,ZN,CLM,ALM,PERM)
      ELSE IF(KR.EQ.18)THEN
        CALL TLM618(KR,XN,YN,ZN,CLM,ALM,PERM)
      ELSE IF(KR.EQ.20)THEN
        CALL TLM820(KR,XN,YN,ZN,CLM,ALM,PERM)
      ELSE IF(KR.EQ.27)THEN
        CALL TLM827(KR,XN,YN,ZN,CLM,ALM,PERM)
      ELSE
        TYPE *, 'ERREUR DANS ELEMENT:',LM
        TYPE *, 'KR=',KR
        STOP 'CALCUL INTERROMPU'
      END IF

      SUM=0.0D+0
      DO 50 K=1,KR
        SUM=SUM+ALM(K)
50  CONTINUE

```

```
50      CONTINUE
        IF(SUM.LE.0.0) THEN
          ISTOP=ISTOP+100
          WRITE(INRES,700)LM,NQ,NP,KR,(LVB(K),K=1,KR)
700      FORMAT(/' VOLUME NUL OU NEGATIF POUR: '/
          1      I5,3X,3I4,5X,6I7/25X,6I7/25X,6I7/25X,6I7)
          END IF

          WRITE(INMAT)NQ,NP,KR,LZ,(LVB(K),K=1,KR),(ALM(K),K=1,KR),
          1      (LDEST(K),K=1,KR),(CLM(L),L=1,LZ)
100     CONTINUE
        END
```

```

SUBROUTINE TLM23(KR,XN,YN,ZN,CLM,ALM,PERM)
  IMPLICIT REAL*8 (A,B,C,D,F,S,W)
  REAL*8 XN,YN,ZN,PERM
  DIMENSION XN(1),YN(1),ZN(1),CLM(1),ALM(1)
  DIMENSION SI3(3),WI3(3),FN(3),FNDS(3),BX(3),BY(3),BZ(3)
  DIMENSION PERM(6)
  DATA SI3/-0.7745966692414834,0.0D+0,0.7745966692414834/
  DATA WI3/0.5555555555555555,0.8888888888888888,
1      0.5555555555555555/

  DO 30 IS=1,3
    S=SI3(IS)
    W=WI3(IS)
    CALL FN23(FN,S)
    CALL FD23(FNDS,S)
    CALL GRATU(KR,FNDS,XN,YN,ZN,BX,BY,BZ,DTJ)
    CALL MATR4(KR,DTJ,W,BX,BY,BZ,CLM,ALM,FN,PERM)
30  CONTINUE
  END

SUBROUTINE TLM36(KR,XN,YN,ZN,CLM,ALM,PERM)
  IMPLICIT REAL*8 (A,B,C,D,F,W)
  REAL*8 XN,YN,ZN,PERM
  DIMENSION AL7(7),BL7(7),CL7(7),WL7(7)
  DIMENSION XN(1),YN(1),ZN(1),CLM(1),ALM(1)
  DIMENSION BX(6),BY(6),BZ(6),FN(6),FNDS(6),FNDS(6),FNDS(6)
  DIMENSION PERM(6)
  DATA AL7/0.3333333333333333,0.0597158717897698,
1      0.4701420641051151,0.4701420641051151,
2      0.7974269853530873,0.1012865073234563,
3      0.1012865073234563/
  DATA BL7/0.3333333333333333,0.4701420641051151,
1      0.0597158717897698,0.4701420641051151,
2      0.1012865073234563,0.7974269853530873,
3      0.1012865073234563/
  DATA CL7/0.3333333333333333,0.4701420641051151,
1      0.4701420641051151,0.0597158717897698,
2      0.1012865073234563,0.1012865073234563,
3      0.7974269853530873/
  DATA WL7/0.1125D+0,3*0.0661970763942531,3*0.0629695902724136/

  DO 30 IT=1,7
    AL=AL7(IT)
    BL=BL7(IT)
    CL=CL7(IT)
    W=WL7(IT)
    CALL FN36(AL,BL,CL,FN)
    CALL FD36(AL,BL,CL,FNDS,FNDS)
    CALL GRATB(KR,FNDS,FNDS,XN,YN,ZN,BX,BY,BZ,DTJ)
    CALL MATR4(KR,DTJ,W,BX,BY,BZ,CLM,ALM,FN,PERM)
30  CONTINUE
  END

```

```

SUBROUTINE TLM48(KR,XN,YN,ZN,CLM,ALM,PERM)
IMPLICIT REAL*8 (A,B,C,D,F,S,T,W)
REAL*8 XN,YN,ZN,PERM
DIMENSION XN(1),YN(1),ZN(1),CLM(1),ALM(1)
DIMENSION SI3(3),WI3(3),FN(8),FNDS(8),FNDT(8)
DIMENSION BX(8),BY(8),BZ(8)
DIMENSION PERM(6)
DATA SI3/-0.7745966692414834,0.0D+0,0.7745966692414834/
DATA WI3/0.5555555555555555,0.8888888888888888,
1      0.5555555555555555/

```

```

DO 50 IT=1,3
  T=SI3(IT)
  WT=WI3(IT)
  DO 40 IS=1,3
    S=SI3(IS)
    W=WT*WI3(IS)
    CALL FN48(S,T,FN,SM,SP,SQ,TM,TP,TQ)
    CALL FD48(S,T,SM,SP,SQ,TM,TP,TQ,FNDS,FNDT)
    CALL GRATB(KR,FNDS,FNDT,XN,YN,ZN,BX,BY,BZ,DTJ)
    CALL MATR4(KR,DTJ,W,BX,BY,BZ,CLM,ALM,FN,PERM)
40  CONTINUE
50  CONTINUE
END

```

```

SUBROUTINE TLM49(KR,XN,YN,ZN,CLM,ALM,PERM)
IMPLICIT REAL*8 (A,B,C,D,F,S,T,W)
REAL*8 XN,YN,ZN,PERM
DIMENSION XN(1),YN(1),ZN(1),CLM(1),ALM(1)
DIMENSION SI3(3),WI3(3),FN(9),FNDS(9),FNDT(9)
DIMENSION BX(9),BY(9),BZ(9)
DIMENSION PERM(6)
DATA SI3/-0.7745966692414834,0.0D+0,0.7745966692414834/
DATA WI3/0.5555555555555555,0.8888888888888888,
1      0.5555555555555555/

```

```

DO 50 IT=1,3
  T=SI3(IT)
  WT=WI3(IT)
  DO 40 IS=1,3
    S=SI3(IS)
    W=WT*WI3(IS)
    CALL FN49(S,T,FN,SQM,SQP,SQ,TQM,TQP,TQ)
    CALL FD49(S,T,SQM,SQP,SQ,TQM,TQP,TQ,FNDS,FNDT)
    CALL GRATB(KR,FNDS,FNDT,XN,YN,ZN,BX,BY,BZ,DTJ)
    CALL MATR4(KR,DTJ,W,BX,BY,BZ,CLM,ALM,FN,PERM)
40  CONTINUE
50  CONTINUE
END

```

```

SUBROUTINE TLM410(KR,XN,YN,ZN,CLM,ALM,PERM)
IMPLICIT REAL*8 (A,B,C,D,F,W)
REAL*8 XN,YN,ZN,PERM
DIMENSION XN(1),YN(1),ZN(1),CLM(1),ALM(1)
DIMENSION BX(10),BY(10),BZ(10)
DIMENSION FN(10),FNDS(10),FNDT(10),FNDU(10)
DIMENSION AL5(5),BL5(5),CL5(5),DL5(5),WL5(5)
DIMENSION PERM(6)
DATA AL5/0.25,0.50,3*0.16666666666666666/
DATA BL5/0.25,0.16666666666666666,0.50,2*0.16666666666666666/
DATA CL5/0.25,2*0.16666666666666666,0.50,0.16666666666666666/
DATA DL5/0.25,3*0.16666666666666666,0.50/
DATA WL5/-0.80,0.45,0.45,0.45,0.45/
DO 20 IP=1,5
  AL=AL5(IP)
  BL=BL5(IP)
  CL=CL5(IP)
  DL=DL5(IP)
  W=-WL5(IP)/6.0
  CALL FN410(AL,BL,CL,DL,A2,B2,C2,D2,FN)
  CALL DS410(AL,BL,CL,DL,A2,B2,C2,D2,FNDS)
  CALL DT410(AL,BL,CL,DL,A2,B2,C2,D2,FNDT)
  CALL DU410(AL,BL,CL,DL,A2,B2,C2,D2,FNDU)
  CALL GRATT(KR,FNDS,FNDT,FNDU,XN,YN,ZN,BX,BY,BZ,DTJ)
  CALL MATR4(KR,DTJ,W,BX,BY,BZ,CLM,ALM,FN,PERM)
20 CONTINUE
END

SUBROUTINE TLM513(KR,XN,YN,ZN,CLM,ALM,PERM)
IMPLICIT REAL*8 (A,B,C,D,F,S,T,U,W)
REAL*8 XN,YN,ZN,PERM
DIMENSION XN(1),YN(1),ZN(1),CLM(1),ALM(1)
DIMENSION SI3(3),WI3(3),BX(20),BY(20),BZ(20)
DIMENSION FN(20),FNDS(20),FNDT(20),FNDU(20)
DIMENSION PERM(6)
DATA SI3/-0.7745966692414834,0.0D+0,0.7745966692414834/
DATA WI3/0.5555555555555555,0.8888888888888888,
1          0.5555555555555555/
DO 70 IU=1,3
  U=SI3(IU)
  WU=WI3(IU)
  DO 60 IT=1,3
    T=SI3(IT)
    WT=WI3(IT)
    DO 50 IS=1,3
      S=SI3(IS)
      W=WU*WT*WI3(IS)
      CALL BRAUX(SM,SP,SQ,TM,TP,TQ,UM,UP,UQ,A1,A2,A3,A4,
1      S,T,U)
      CALL FN513(SM,SP,SQ,TM,TP,TQ,UM,UP,UQ,A1,A2,A3,A4,FN)
      CALL DS513(TM,TP,TQ,UM,UP,UQ,S,FNDS)
      CALL DT513(SM,SP,SQ,UM,UP,UQ,T,FNDT)
      CALL DU513(SM,SP,SQ,TM,TP,TQ,U,FNDU,A1,A2,A3,A4)
      CALL GRATT(KR,FNDS,FNDT,FNDU,XN,YN,ZN,BX,BY,BZ,DTJ)
      CALL MATR4(KR,DTJ,W,BX,BY,BZ,CLM,ALM,FN,PERM)
50 CONTINUE
60 CONTINUE
70 CONTINUE
END

```

```

SUBROUTINE TLM615(KR,XN,YN,ZN,CLM,ALM,PERM)
IMPLICIT REAL*8 (A,B,C,D,F,S,U,W)
REAL*8 XN,YN,ZN,PERM
DIMENSION XN(1),YN(1),ZN(1),CLM(1),ALM(1)
DIMENSION SI3(3),WI3(3),AL7(7),BL7(7),CL7(7),WL7(7)
DIMENSION BX(15),BY(15),BZ(15)
DIMENSION FN(15),FNDS(15),FNDT(15),FNDU(15)
DIMENSION PERM(6)
DATA SI3/-0.7745966692414834,0.0D+0,0.7745966692414834/
DATA WI3/0.5555555555555555,0.8888888888888888,
1      0.5555555555555555/
DATA AL7/0.3333333333333333,0.0597158717897698,
1      0.4701420641051151,0.4701420641051151,
2      0.7974269853530873,0.1012865073234563,
3      0.1012865073234563/
DATA BL7/0.3333333333333333,0.4701420641051151,
1      0.0597158717897698,0.4701420641051151,
2      0.1012865073234563,0.7974269853530873,
3      0.1012865073234563/
DATA CL7/0.3333333333333333,0.4701420641051151,
1      0.4701420641051151,0.0597158717897698,
2      0.1012865073234563,0.1012865073234563,
3      0.7974269853530873/
DATA WL7/0.1125D+0,3*0.0661970763942531,3*0.0629695902724136/

DO 50 IT=1,7
  AL=AL7(IT)
  BL=BL7(IT)
  CL=CL7(IT)
  WT=WL7(IT)
  DO 40 IU=1,3
    U=SI3(IU)
    W=WT*WI3(IU)
    CALL PRAUX(AL,BL,CL,U,UM,UP,UQ,A1,A2,A3)
    CALL FN615(AL,BL,CL,UM,UP,UQ,A1,A2,A3,FN)
    CALL DS615(AL,BL,CL,UM,UP,UQ,FNDS)
    CALL DT615(AL,BL,CL,UM,UP,UQ,FNDT)
    CALL DU615(AL,BL,CL,A1,A2,A3,U,FNDU)
    CALL GRATF(KR,FNDS,FNDT,FNDU,XN,YN,ZN,BX,BY,BZ,DTJ)
    CALL MATR4(KR,DTJ,W,BX,BY,BZ,CLM,ALM,FN,PERM)
40    CONTINUE
50    CONTINUE
END

```

```

SUBROUTINE TLM618(KR,XN,YN,ZN,CLM,ALM,PERM)
IMPLICIT REAL*8 (A,B,C,D,F,S,U,W)
REAL*8 XN,YN,ZN,PERM
DIMENSION XN(1),YN(1),ZN(1),CLM(1),ALM(1)
DIMENSION SI3(3),WI3(3),AL7(7),BL7(7),CL7(7),WL7(7)
DIMENSION BX(18),BY(18),BZ(18)
DIMENSION FN(18),FNDS(18),FNDT(18),FNDU(18)
DIMENSION PERM(6)
DATA SI3/-0.7745966692414834,0.0D+0,0.7745966692414834/
DATA WI3/0.5555555555555555,0.8888888888888888,
1      0.5555555555555555/
DATA AL7/0.3333333333333333,0.0597158717897698,
1      0.4701420641051151,0.4701420641051151,
2      0.7974269853530873,0.1012865073234563,
3      0.1012865073234563/
DATA BL7/0.3333333333333333,0.4701420641051151,
1      0.0597158717897698,0.4701420641051151,
2      0.1012865073234563,0.7974269853530873,
3      0.1012865073234563/
DATA CL7/0.3333333333333333,0.4701420641051151,
1      0.4701420641051151,0.0597158717897698,
2      0.1012865073234563,0.1012865073234563,
3      0.7974269853530873/
DATA WL7/0.1125D+0,3*0.0661970763942531,3*0.0629695902724136/

DO 50 IT=1,7
  AL=AL7(IT)
  BL=BL7(IT)
  CL=CL7(IT)
  WT=WL7(IT)
  DO 40 IU=1,3
    U=SI3(IU)
    W=WT*WI3(IU)
    CALL FN618(AL,BL,CL,U,A2,B2,C2,A,B,C,
1      AA,BB,CC,UM,UQ,UP,FN)
    CALL FD618(A2,B2,C2,A,B,C,AA,BB,CC,U,UM,UQ,UP,
1      FNDS,FNDT,FNDU)
    CALL GRATT(KR,FNDS,FNDT,FNDU,XN,YN,ZN,BX,BY,BZ,DTJ)
    CALL MATR4(KR,DTJ,W,BX,BY,BZ,CLM,ALM,DLM,FN,PERM)
40  CONTINUE
50  CONTINUE
END

```

```

SUBROUTINE TLM820(KR,XN,YN,ZN,CLM,ALM,PERM)
IMPLICIT REAL*8 (A,B,C,D,F,S,T,U,W)
REAL*8 XN,YN,ZN,PERM
DIMENSION XN(1),YN(1),ZN(1),CLM(1),ALM(1)
DIMENSION SI3(3),WI3(3),BX(20),BY(20),BZ(20)
DIMENSION FN(20),FNDS(20),FNDT(20),FNDU(20)
DIMENSION PERM(6)
DATA SI3/-0.7745966692414834,0.0D+0,0.7745966692414834/
DATA WI3/0.5555555555555555,0.8888888888888888,
1      0.5555555555555555/
DO 70 IU=1,3
  U=SI3(IU)
  WU=WI3(IU)
  DO 60 IT=1,3
    T=SI3(IT)
    WT=WI3(IT)
    DO 50 IS=1,3
      S=SI3(IS)
      W=WU*WT*WI3(IS)
      CALL BRAUX(SM,SP,SQ,TM,TP,TQ,UM,UP,UQ,A1,A2,A3,A4,
1      S,T,U)
      CALL FN820(SM,SP,SQ,TM,TP,TQ,UM,UP,UQ,A1,A2,A3,A4,FN)
      CALL DS820(TM,TP,TQ,UM,UP,UQ,S,FNDS)
      CALL DT820(SM,SP,SQ,UM,UP,UQ,T,FNDT)
      CALL DU820(SM,SP,SQ,TM,TP,TQ,U,FNDU,A1,A2,A3,A4)
      CALL GRATT(KR,FNDS,FNDT,FNDU,XN,YN,ZN,BX,BY,BZ,DTJ)
      CALL MATR4(KR,DTJ,W,BX,BY,BZ,CLM,ALM,FN,PERM)
50      CONTINUE
60      CONTINUE
70      CONTINUE
END

```

```

SUBROUTINE TLM827(KR,XN,YN,ZN,CLM,ALM,PERM)
IMPLICIT REAL*8 (A,B,C,D,F,S,T,U,W)
REAL*8 XN,YN,ZN,PERM
DIMENSION XN(1),YN(1),ZN(1),CLM(1),ALM(1)
DIMENSION SI3(3),WI3(3),BX(27),BY(27),BZ(27)
DIMENSION FN(27),FNDS(27),FNDT(27),FNDU(27),SQ(3),TQ(3),UQ(3)
DIMENSION PERM(6)
DATA SI3/-0.7745966692414834,0.0D+0,0.7745966692414834/
DATA WI3/0.5555555555555555,0.8888888888888888,
1      0.5555555555555555/
DO 70 IU=1,3
  U=SI3(IU)
  WU=WI3(IU)
  DO 60 IT=1,3
    T=SI3(IT)
    WT=WI3(IT)
    DO 50 IS=1,3
      S=SI3(IS)
      W=WU*WT*WI3(IS)
      CALL FN827(S,T,U,FN,SQ,TQ,UQ)
      CALL FD827(S,T,U,SQ,TQ,UQ,FNDS,FNDT,FNDU)
      CALL GRATT(KR,FNDS,FNDT,FNDU,XN,YN,ZN,BX,BY,BZ,DTJ)
      CALL MATR4(KR,DTJ,W,BX,BY,BZ,CLM,ALM,FN,PERM)
50      CONTINUE
60      CONTINUE
70      CONTINUE
END

```

```

SUBROUTINE GRATU(KR,FNDS,XN,YN,ZN,BX,BY,BZ,DETJ)
IMPLICIT REAL*8 (B,D,F,G)
REAL*8 XN,YN,ZN
DIMENSION FNDS(1),XN(1),YN(1),ZN(1),BX(1),BY(1),BZ(1)

```

C MATRICE DES GRADIENTS - SEGMENT UNIDIM EN ESPACE TRIDI

```

D11=0.0D+0
D12=0.0D+0
D13=0.0D+0
DO 10 K=1,KR
  FS=FNDS(K)
  D11=D11+FS*XN(K)  ! D1J: composantes du vecteur
  D12=D12+FS*YN(K)  !           de base covariant
  D13=D13+FS*ZN(K)
10  CONTINUE

G11=D11*D11+D12*D12+D13*D13  ! tenseur metrique covariant
GV11=1.0D+0/G11              ! tenseur metrique contravar.
F11=D11*GV11  ! F1J: composantes du vecteur
F12=D12*GV11  !           de base contravariant
F13=D13*GV11

DO 20 K=1,KR
  FS=FNDS(K)
  BX(K)=F11*FS
  BY(K)=F12*FS  ! lignes de la matrice des gradients
  BZ(K)=F13*FS
20  CONTINUE

DETJ=SQRT(G11)  ! element de longueur
END

```

```

SUBROUTINE GRATB(KUREL,FNDS,FNDT,XN,YN,ZN,BX,BY,BZ,DETJ)
IMPLICIT REAL*8 (B,D,F,G)
REAL*8 XN,YN,ZN
DIMENSION FNDS(1),FNDT(1),BX(1),BY(1),BZ(1),XN(1),YN(1),ZN(1),D(6)
EQUIVALENCE (D11,D(1)),(D12,D(2)),(D13,D(3)),(D21,D(4))
EQUIVALENCE (D22,D(5)),(D23,D(6))

```

C MATRICE DES GRADIENTS - FEUILLE BIDI EN ESPACE TRIDI

```

DO 8 I=1,6
  D(I)=0.0D+0
8 CONTINUE

DO 10 I=1,KUREL
  FS=FNDS(I)
  FT=FNDT(I)
  D11=D11+FS*XN(I) ! DIJ: composantes des vecteurs
  D12=D12+FS*YN(I) ! de base covariants
  D13=D13+FS*ZN(I)
  D21=D21+FT*XN(I)
  D22=D22+FT*YN(I)
  D23=D23+FT*ZN(I)
10 CONTINUE

G11=D11*D11+D12*D12+D13*D13 ! GIJ: tenseur metrique
G12=D11*D21+D12*D22+D13*D23 ! covariant
G22=D21*D21+D22*D22+D23*D23
DETJ=G11*G22-G12*G12
DETJV=1.0D+0/DETJ
GV11=G22*DETJV ! GVIJ: tenseur metrique
GV12=-G12*DETJV ! contravariant
GV22=G11*DETJV
F11=D11*GV11+D21*GV12 ! FIJ: composantes des vecteurs
F12=D12*GV11+D22*GV12 ! de base contravariants
F13=D13*GV11+D23*GV12
F21=D11*GV12+D21*GV22
F22=D12*GV12+D22*GV22
F23=D13*GV12+D23*GV22

DO 20 I=1,KUREL
  FS=FNDS(I)
  FT=FNDT(I)
  BX(I)=F11*FS+F21*FT ! lignes de la matrice
  BY(I)=F12*FS+F22*FT ! des gradients
  BZ(I)=F13*FS+F23*FT
20 CONTINUE

DETJ=SQRT(DETJ) ! element de surface
END

```

```

SUBROUTINE GRATT(KUREL,FNDS,FNDT,FNDU,XN,YN,ZN,
1          BX,BY,BZ,DETJ)
  IMPLICIT REAL*8 (B,D,F)
  REAL*8 XN,YN,ZN,XX,YY,ZZ
  DIMENSION D(9),FNDS(1),FNDT(1),FNDU(1),BX(1),BY(1),BZ(1)
  DIMENSION XN(1),YN(1),ZN(1)
  EQUIVALENCE (D11,D(1)),(D12,D(2)),(D13,D(3))
  EQUIVALENCE (D21,D(4)),(D22,D(5)),(D23,D(6))
  EQUIVALENCE (D31,D(7)),(D32,D(8)),(D33,D(9))

C  MATRICE DES GRADIENTS - ELEMENT TRIDI EN ESPACE TRIDI

  DO 8 I=1,9
    D(I)=0.0D+0
8  CONTINUE
  DO 10 I=1,KUREL
    XX=XN(I)
    YY=YN(I)
    ZZ=ZN(I)
    FS=FNDS(I)
    FT=FNDT(I)
    FU=FNDU(I)
    D11=D11+FS*XX      ! DIJ: matrice de transformation
    D12=D12+FS*YY
    D13=D13+FS*ZZ
    D21=D21+FT*XX
    D22=D22+FT*YY
    D23=D23+FT*ZZ
    D31=D31+FU*XX
    D32=D32+FU*YY
    D33=D33+FU*ZZ
10  CONTINUE
    DM11=D22*D33-D23*D32      ! DMIJ: mineures de DIJ
    DM12=-(D21*D33-D23*D31)
    DM13=D21*D32-D22*D31
    DM21=-(D12*D33-D13*D32)
    DM22=D11*D33-D13*D31
    DM23=-(D11*D32-D12*D31)
    DM31=D12*D23-D13*D22
    DM32=-(D11*D23-D13*D21)
    DM33=D11*D22-D12*D21
    DETJ=D11*DM11+D12*DM12+D13*DM13 ! element de volume
    DETJV=1.0D+0/DETT
    DV11=DM11*DETTJV      ! DVIJ: inverse de DIJ a partir des
    DV12=DM21*DETTJV      ! mineures transposees DMIJ
    DV13=DM31*DETTJV
    DV21=DM12*DETTJV
    DV22=DM22*DETTJV
    DV23=DM32*DETTJV
    DV31=DM13*DETTJV
    DV32=DM23*DETTJV
    DV33=DM33*DETTJV
    DO 20 I=1,KUREL
      FS=FNDS(I)
      FT=FNDT(I)
      FU=FNDU(I)
      BX(I)=DV11*FS+DV12*FT+DV13*FU ! lignes de la matrice
      BY(I)=DV21*FS+DV22*FT+DV23*FU ! des gradients
      BZ(I)=DV31*FS+DV32*FT+DV33*FU
20  CONTINUE
  END

```

```

SUBROUTINE MATR4(KUREL,DETJ,W,BX,BY,BZ,CLM,ALM,FN,PERM)
IMPLICIT REAL*8 (A,B,C,D,F)
REAL*8 W,P,PERM
DIMENSION BX(1),BY(1),BZ(1),FN(1),CLM(1),ALM(1)
DIMENSION PERM(6),P(6)

C      MATRICE DES PERMABILITES POUR MODELE TRIDI

C      Les ALM(I) sont des longueurs pour les elements 1-D,
C      des surfaces pour les elements 2-D et des volumes pour
C      les elements 3-D!
C      CLM(I): triangle superieur de la matrice "elementaire".

CST=DETJ*W
DO 10 I=1,6
    P(I)=PERM(I)*CST
10  CONTINUE

L=0
DO 30 KO=1,KUREL
    BXKO=BX(KO)*P(1)+BY(KO)*P(2)+BZ(KO)*P(4)
    BYKO=BX(KO)*P(2)+BY(KO)*P(3)+BZ(KO)*P(5)
    BZKO=BX(KO)*P(4)+BY(KO)*P(5)+BZ(KO)*P(6)
    DO 20 LI=1,KO
        L=L+1
        CLM(L)=CLM(L)+BXKO*BX(LI)+BYKO*BY(LI)+BZKO*BZ(LI)
20    CONTINUE
    ALM(KO)=ALM(KO)+FN(KO)*CST
30  CONTINUE
END

```

```
SUBROUTINE FN23(FN,S)
REAL*8 FN,S,A
DIMENSION FN(3)
A=S*S
FN(1)=0.5*(A-S)
FN(2)=1.0-A
FN(3)=0.5*(A+S)
END
```

```
SUBROUTINE FD23(FNDS,S)
REAL*8 FNDS,S
DIMENSION FNDS(3)
FNDS(1)=S-0.5
FNDS(2)=-S-S
FNDS(3)=S+0.5
END
```

```
SUBROUTINE FN36(AL,BL,CL,FN)
IMPLICIT REAL*8 (A,B,C,F)
DIMENSION FN(6)
A=AL*BL*2.0
B=BL*CL*2.0
C=CL*AL*2.0
FN(1)=AL-C-A
FN(3)=BL-A-B
FN(5)=CL-B-C
FN(2)=A+A
FN(4)=B+B
FN(6)=C+C
END
```

```
SUBROUTINE FD36(AL,BL,CL,FNDS,FNDT)
IMPLICIT REAL*8 (A,B,C,F)
DIMENSION FNDS(6),FNDT(6)
AS=BL+BL
AT=AL+AL
BS=-AS
BT=CL+CL-AS
CS=CL+CL-AT
CT=-AT
FNDS(1)=1.0-CS-AS
FNDT(1)=0.0
FNDS(3)=0.0
FNDT(3)=1.0-AT-BT
FNDS(5)=-1.0-BT-CS
FNDT(5)=-1.0-BT-CT
FNDS(2)=AS+AS
FNDT(2)=AT+AT
FNDS(4)=BS+BS
FNDT(4)=BT+BT
FNDS(6)=CS+CS
FNDT(6)=CT+CT
END
```

```

SUBROUTINE FN48(S,T,FN,SM,SP,SQ,TM,TP,TQ)
IMPLICIT REAL*8 (S,T,F,A,B,C,D)
DIMENSION FN(8)
CS=0.5*S
CT=0.5*T
SM=0.5-CS
SP=0.5+CS
TM=0.5-CT
TP=0.5+CT
SQ=0.5-S*CS
TQ=0.5-T*CT
A=SQ*TM
B=SP*TQ
C=SQ*TP
D=SM*TQ
FN(1)=SM*TM-D-A
FN(3)=SP*TM-A-B
FN(5)=SP*TP-B-C
FN(7)=SM*TP-C-D
FN(2)=A+A
FN(4)=B+B
FN(6)=C+C
FN(8)=D+D
END

```

```

SUBROUTINE FD48(S,T,SM,SP,SQ,TM,TP,TQ,FNDS,FNDT)
IMPLICIT REAL*8 (S,T,F,A,B,C,D)
DIMENSION FNDS(8),FNDT(8)
AS=-S*TM
AT=-0.5*SQ
BS=0.5*TQ
BT=-T*SP
CS=-S*TP
CT=0.5*SQ
DS=-0.5*TQ
DT=-T*SM
FNDS(1)=-0.5*TM-DS-AS
FNDT(1)=-0.5*SM-DT-AT
FNDS(3)=0.5*TM-AS-BS
FNDT(3)=-0.5*SP-AT-BT
FNDS(5)=0.5*TP-BS-CS
FNDT(5)=0.5*SP-BT-CT
FNDS(7)=-0.5*TP-CS-DS
FNDT(7)=0.5*SM-CT-DT
FNDS(2)=AS+AS
FNDT(2)=AT+AT
FNDS(4)=BS+BS
FNDT(4)=BT+BT
FNDS(6)=CS+CS
FNDT(6)=CT+CT
FNDS(8)=DS+DS
FNDT(8)=DT+DT
END

```

```
SUBROUTINE FN49(S,T,FN,SQM,SQP,SQ,TQM,TQP,TQ)
IMPLICIT REAL*8 (S,T,F,A,B)
DIMENSION FN(9)
A=S*S
B=T*T
SQM=0.5*(A-S)
SQP=0.5*(A+S)
SQ=1.0-A
TQM=0.5*(B-T)
TQP=0.5*(B+T)
TQ=1.0-B
FN(1)=SQM*TQM
FN(2)=SQ*TQM
FN(3)=SQP*TQM
FN(4)=SQP*TQ
FN(5)=SQP*TQP
FN(6)=SQ*TQP
FN(7)=SQM*TQP
FN(8)=SQM*TQ
FN(9)=SQ*TQ
END
```

```
SUBROUTINE FD49(S,T,SQM,SQP,SQ,TQM,TQP,TQ,FNDS,FNDT)
IMPLICIT REAL*8 (S,T,F,D)
DIMENSION FNDS(9),FNDT(9)
DSQM=S-0.5
DSQP=S+0.5
DSQ=-S-S
DTQM=T-0.5
DTQP=T+0.5
DTQ=-T-T
FNDS(1)=DSQM*TQM
FNDT(1)=SQM*DTQM
FNDS(2)=DSQ*TQM
FNDT(2)=SQ*DTQM
FNDS(3)=DSQP*TQM
FNDT(3)=SQP*DTQM
FNDS(4)=DSQP*TQ
FNDT(4)=SQP*DTQ
FNDS(5)=DSQP*TQP
FNDT(5)=SQP*DTQP
FNDS(6)=DSQ*TQP
FNDT(6)=SQ*DTQP
FNDS(7)=DSQM*TQP
FNDT(7)=SQM*DTQP
FNDS(8)=DSQM*TQ
FNDT(8)=SQM*DTQ
FNDS(9)=DSQ*TQ
FNDT(9)=SQ*DTQ
END
```

```
SUBROUTINE FN410(AL,BL,CL,DL,A2,B2,C2,D2,FN)
  IMPLICIT REAL*8 (A,B,C,D,F)
  DIMENSION FN(10)
  A2=AL+AL
  B2=BL+BL
  C2=CL+CL
  D2=DL+DL
  FN(1)=AL*(A2-1.0)
  FN(3)=BL*(B2-1.0)
  FN(5)=CL*(C2-1.0)
  FN(10)=DL*(D2-1.0)
  FN(2)=A2*B2
  FN(4)=B2*C2
  FN(6)=C2*A2
  FN(7)=A2*D2
  FN(8)=B2*D2
  FN(9)=C2*D2
  END
```

```
SUBROUTINE DS410(AL,BL,CL,DL,A2,B2,C2,D2,FNDS)
  IMPLICIT REAL*8 (A,B,C,D,F)
  DIMENSION FNDS(10)
  FNDS(1)=A2+A2-1.0
  FNDS(3)=0.0
  FNDS(5)=0.0
  FNDS(10)=-D2-D2+1.0
  FNDS(2)=B2+B2
  FNDS(4)=0.0
  FNDS(6)=C2+C2
  FNDS(7)=D2+D2-A2-A2
  FNDS(8)=-FNDS(2)
  FNDS(9)=-FNDS(6)
  END
```

```
SUBROUTINE DT410(AL,BL,CL,DL,A2,B2,C2,D2,FNDT)
  IMPLICIT REAL*8 (A,B,C,D,F)
  DIMENSION FNDT(10)
  FNDT(1)=0.0
  FNDT(3)=B2+B2-1.0
  FNDT(5)=0.0
  FNDT(10)=-D2-D2+1.0
  FNDT(2)=A2+A2
  FNDT(4)=C2+C2
  FNDT(6)=0.0
  FNDT(7)=-FNDT(2)
  FNDT(8)=D2+D2-B2-B2
  FNDT(9)=-FNDT(4)
  END
```

```

SUBROUTINE DU410(AL,BL,CL,DL,A2,B2,C2,D2,FNDU)
IMPLICIT REAL*8 (A,B,C,D,F)
DIMENSION FNDU(10)
FNDU(1)=0.0
FNDU(3)=0.0
FNDU(5)=C2+C2-1.0
FNDU(10)=-D2-D2+1.0
FNDU(2)=0.0
FNDU(4)=B2+B2
FNDU(6)=A2+A2
FNDU(7)=-FNDU(6)
FNDU(8)=-FNDU(4)
FNDU(9)=D2+D2-C2-C2
END

```

```

SUBROUTINE FN513(SM,SP,SQ,TM,TP,TQ,UM,UP,UQ,A1,A2,A3,A4,FN)
IMPLICIT REAL*8 (S,T,U,F,A)
DIMENSION FN(20)
C 13-node pyramide from 20-node brick element
FN(2)=SQ*TM*UM
FN(4)=TQ*SP*UM
FN(6)=SQ*TP*UM
FN(8)=TQ*SM*UM
FN(9)=UQ*A1
FN(10)=UQ*A2
FN(11)=UQ*A3
FN(12)=UQ*A4
FN(1)=A1*UM-0.5*(FN(2)+FN(8)+FN(9))
FN(3)=A2*UM-0.5*(FN(4)+FN(2)+FN(10))
FN(5)=A3*UM-0.5*(FN(6)+FN(4)+FN(11))
FN(7)=A4*UM-0.5*(FN(8)+FN(6)+FN(12))
FN(13)=UP-0.5*UQ
END

```

```

SUBROUTINE DS513(TM,TP,TQ,UM,UP,UQ,S,FNDS)
IMPLICIT REAL*8 (S,T,U,F,D)
DIMENSION FNDS(20)
DATA DSM,DSP/-0.5,0.5/
C 13-node pyramide from 20-node brick element
DSQ=-S-S
DSA=DSM*TM
DSB=-DSA
DSC=DSP*TP
DSD=-DSC
FNDS(2)=DSQ*TM*UM
FNDS(4)=DSP*TQ*UM
FNDS(6)=DSQ*TP*UM
FNDS(8)=DSM*TQ*UM
FNDS(9)=UQ*DSA
FNDS(10)=UQ*DSB
FNDS(11)=UQ*DSC
FNDS(12)=UQ*DSD
FNDS(1)=DSA*UM-0.5*(FNDS(2)+FNDS(8)+FNDS(9))

```

```

FNDS(3)=DSB*UM-0.5*(FNDS(4)+FNDS(2)+FNDS(10))
FNDS(5)=DSC*UM-0.5*(FNDS(6)+FNDS(4)+FNDS(11))
FNDS(7)=DSD*UM-0.5*(FNDS(8)+FNDS(6)+FNDS(12))
FNDS(13)=0.0
END

```

```

C
SUBROUTINE DT513(SM,SP,SQ,UM,UP,UQ,T,FNDT)
IMPLICIT REAL*8 (S,T,U,F,D)
DIMENSION FNDT(20)
DATA DTM,DTP/-0.5,0.5/
13-node pyramide from 20-node brick element
DTQ=-T-T
DTA=SM*DTM
DTB=SP*DTM
DTC=SP*DTP
DTD=SM*DTP
FNDT(2)=SQ*DTM*UM
FNDT(4)=SP*DTQ*UM
FNDT(6)=SQ*DTP*UM
FNDT(8)=SM*DTQ*UM
FNDT(9)=UQ*DTA
FNDT(10)=UQ*DTB
FNDT(11)=UQ*DTC
FNDT(12)=UQ*DTD
FNDT(1)=DTA*UM-0.5*(FNDT(2)+FNDT(8)+FNDT(9))
FNDT(3)=DTB*UM-0.5*(FNDT(4)+FNDT(2)+FNDT(10))
FNDT(5)=DTC*UM-0.5*(FNDT(6)+FNDT(4)+FNDT(11))
FNDT(7)=DTD*UM-0.5*(FNDT(8)+FNDT(6)+FNDT(12))
FNDT(13)=0.0
END

```

```

C
SUBROUTINE DU513(SM,SP,SQ,TM,TP,TQ,U,FNDU,A1,A2,A3,A4)
IMPLICIT REAL*8 (S,T,U,F,A,D)
DIMENSION FNDU(20)
DATA DUM,DUP/-0.5,0.5/
13-node pyramide from 20-node brick element
DUQ=-U-U
FNDU(2)=SQ*TM*DUM
FNDU(4)=SP*TQ*DUM
FNDU(6)=SQ*TP*DUM
FNDU(8)=SM*TQ*DUM
FNDU(9)=A1*DUQ
FNDU(10)=A2*DUQ
FNDU(11)=A3*DUQ
FNDU(12)=A4*DUQ
FNDU(1)=A1*DUM-0.5*(FNDU(2)+FNDU(8)+FNDU(9))
FNDU(3)=A2*DUM-0.5*(FNDU(4)+FNDU(2)+FNDU(10))
FNDU(5)=A3*DUM-0.5*(FNDU(6)+FNDU(4)+FNDU(11))
FNDU(7)=A4*DUM-0.5*(FNDU(8)+FNDU(6)+FNDU(12))
FNDU(13)=DUP+U
END

```

```

SUBROUTINE FN615(AL,BL,CL,UM,UP,UQ,A1,A2,A3,FN)
IMPLICIT REAL*8 (A,B,C,U,F)
DIMENSION FN(15)

```

```

FN(2)=A1*UM
FN(4)=A2*UM
FN(6)=A3*UM
FN(7)=AL*UQ
FN(8)=BL*UQ
FN(9)=CL*UQ
FN(11)=A1*UP
FN(13)=A2*UP
FN(15)=A3*UP

```

```

FN(1)=AL*UM-0.5*(FN(2)+FN(6)+FN(7))
FN(3)=BL*UM-0.5*(FN(4)+FN(2)+FN(8))
FN(5)=CL*UM-0.5*(FN(6)+FN(4)+FN(9))
FN(10)=AL*UP-0.5*(FN(11)+FN(15)+FN(7))
FN(12)=BL*UP-0.5*(FN(13)+FN(11)+FN(8))
FN(14)=CL*UP-0.5*(FN(15)+FN(13)+FN(9))

```

```

END

```

```

SUBROUTINE DS615(AL,BL,CL,UM,UP,UQ,FNDS)
IMPLICIT REAL*8 (A,B,C,U,F,D)
DIMENSION FNDS(15)

```

```

DSA1=4.0*BL
DSA2=-DSA1
DSA3=4.0*(CL-AL)

```

```

FNDS(2)=DSA1*UM
FNDS(4)=DSA2*UM
FNDS(6)=DSA3*UM
FNDS(7)=UQ
FNDS(8)=0.0
FNDS(9)=-UQ
FNDS(11)=DSA1*UP
FNDS(13)=DSA2*UP
FNDS(15)=DSA3*UP

```

```

FNDS(1)=UM-0.5*(FNDS(2)+FNDS(6)+FNDS(7))
FNDS(3)=0.0-0.5*(FNDS(4)+FNDS(2)+FNDS(8))
FNDS(5)=-UM-0.5*(FNDS(6)+FNDS(4)+FNDS(9))
FNDS(10)=UP-0.5*(FNDS(11)+FNDS(15)+FNDS(7))
FNDS(12)=0.0-0.5*(FNDS(13)+FNDS(11)+FNDS(8))
FNDS(14)=-UP-0.5*(FNDS(15)+FNDS(13)+FNDS(9))

```

```

END

```

```

SUBROUTINE DT615(AL,BL,CL,UM,UP,UQ,FNDT)
IMPLICIT REAL*8 (A,B,C,U,F,D)
DIMENSION FNDT(15)

```

```

DTA1=4.0*AL
DTA2=4.0*(CL-BL)
DTA3=-DTA1

```

```

FNDT(2)=DTA1*UM
FNDT(4)=DTA2*UM
FNDT(6)=DTA3*UM
FNDT(7)=0.0
FNDT(8)=UQ
FNDT(9)=-UQ
FNDT(11)=DTA1*UP
FNDT(13)=DTA2*UP
FNDT(15)=DTA3*UP

FNDT(1)=0.0-0.5*(FNDT(2)+FNDT(6)+FNDT(7))
FNDT(3)=UM-0.5*(FNDT(4)+FNDT(2)+FNDT(8))
FNDT(5)=-UM-0.5*(FNDT(6)+FNDT(4)+FNDT(9))
FNDT(10)=0.0-0.5*(FNDT(11)+FNDT(15)+FNDT(7))
FNDT(12)=UP-0.5*(FNDT(13)+FNDT(11)+FNDT(8))
FNDT(14)=-UP-0.5*(FNDT(15)+FNDT(13)+FNDT(9))

END

```

```

SUBROUTINE DU615(AL,BL,CL,A1,A2,A3,U,FNDU)
IMPLICIT REAL*8 (A,B,C,U,F,D)
DIMENSION FNDU(15)
DATA DUM,DUP/-0.5,0.5/
DUQ=-U-U

FNDU(2)=A1*DUM
FNDU(4)=A2*DUM
FNDU(6)=A3*DUM
FNDU(7)=AL*DUQ
FNDU(8)=BL*DUQ
FNDU(9)=CL*DUQ
FNDU(11)=A1*DUP
FNDU(13)=A2*DUP
FNDU(15)=A3*DUP

FNDU(1)=AL*DUM-0.5*(FNDU(2)+FNDU(6)+FNDU(7))
FNDU(3)=BL*DUM-0.5*(FNDU(4)+FNDU(2)+FNDU(8))
FNDU(5)=CL*DUM-0.5*(FNDU(6)+FNDU(4)+FNDU(9))
FNDU(10)=AL*DUP-0.5*(FNDU(11)+FNDU(15)+FNDU(7))
FNDU(12)=BL*DUP-0.5*(FNDU(13)+FNDU(11)+FNDU(8))
FNDU(14)=CL*DUP-0.5*(FNDU(15)+FNDU(13)+FNDU(9))

END

```

```

1 SUBROUTINE FN618(AL,BL,CL,U,A2,B2,C2,A,B,C,
  AA,BB,CC,UM,UQ,UP,FN)
IMPLICIT REAL*8 (A,B,C,F,R,U)
PARAMETER (R0=0.0D+0,R1=1.0D+0,R2=2.0D+0,RD=0.5D+0)
DIMENSION FN(18)

```

C

```

-----
A2=R2*AL
B2=R2*BL
C2=R2*CL
U2=U*U
A=AL*(A2-R1)
B=BL*(B2-R1)
C=CL*(C2-R1)

```

```

AA=A2*B2
BB=B2*C2
CC=C2*A2
UM=RD*(U2-U)
UP=RD*(U2+U)
UQ=R1-U2

```

C

```

-----
FN(1)=A*UM
FN(3)=B*UM
FN(5)=C*UM
FN(7)=A*UQ
FN(9)=B*UQ
FN(11)=C*UQ
FN(13)=A*UP
FN(15)=B*UP
FN(17)=C*UP
FN(2)=AA*UM
FN(4)=BB*UM
FN(6)=CC*UM
FN(8)=AA*UQ
FN(10)=BB*UQ
FN(12)=CC*UQ
FN(14)=AA*UP
FN(16)=BB*UP
FN(18)=CC*UP
END

```

```

1  SUBROUTINE FD618(A2,B2,C2,A,B,C,AA,BB,CC,U,UM,UQ,UP,
      FNDS,FNDT,FNDU)
      IMPLICIT REAL*8 (A,B,C,D,F,R,U)
      PARAMETER (R0=0.0D+0,R1=1.0D+0,R2=2.0D+0,RD=0.5D+0)
      DIMENSION FNDS(18),FNDT(18),FNDU(18)

```

C

```

-----
AS=R2*A2-R1
BS=R0
CS=R1-R2*C2
AAS=R2*B2
BBS=-AAS
CCS=R2*(C2-A2)

```

C

```

-----
FNDS(1)=AS*UM
FNDS(3)=BS*UM
FNDS(5)=CS*UM
FNDS(7)=AS*UQ
FNDS(9)=BS*UQ
FNDS(11)=CS*UQ
FNDS(13)=AS*UP
FNDS(15)=BS*UP
FNDS(17)=CS*UP
FNDS(2)=AAS*UM
FNDS(4)=BBS*UM
FNDS(6)=CCS*UM
FNDS(8)=AAS*UQ
FNDS(10)=BBS*UQ
FNDS(12)=CCS*UQ
FNDS(14)=AAS*UP
FNDS(16)=BBS*UP
FNDS(18)=CCS*UP

```

AT=R0
 BT=R2*B2-R1
 CT=R1-R2*C2
 AAT=R2*A2
 BBT=R2*(C2-B2)
 CCT=-AAT

C

 FNDT(1)=AT*UM
 FNDT(3)=BT*UM
 FNDT(5)=CT*UM
 FNDT(7)=AT*UQ
 FNDT(9)=BT*UQ
 FNDT(11)=CT*UQ
 FNDT(13)=AT*UP
 FNDT(15)=BT*UP
 FNDT(17)=CT*UP
 FNDT(2)=AAT*UM
 FNDT(4)=BBT*UM
 FNDT(6)=CCT*UM
 FNDT(8)=AAT*UQ
 FNDT(10)=BBT*UQ
 FNDT(12)=CCT*UQ
 FNDT(14)=AAT*UP
 FNDT(16)=BBT*UP
 FNDT(18)=CCT*UP

C

 DUM=U-RD
 DUP=U+RD
 DUQ=-U-U

C

 FNDU(1)=A*DUM
 FNDU(3)=B*DUM
 FNDU(5)=C*DUM
 FNDU(7)=A*DUQ
 FNDU(9)=B*DUQ
 FNDU(11)=C*DUQ
 FNDU(13)=A*DUP
 FNDU(15)=B*DUP
 FNDU(17)=C*DUP
 FNDU(2)=AA*DUM
 FNDU(4)=BB*DUM
 FNDU(6)=CC*DUM
 FNDU(8)=AA*DUQ
 FNDU(10)=BB*DUQ
 FNDU(12)=CC*DUQ
 FNDU(14)=AA*DUP
 FNDU(16)=BB*DUP
 FNDU(18)=CC*DUP
 END

```

SUBROUTINE FN820(SM,SP,SQ,TM,TP,TQ,UM,UP,UQ,A1,A2,A3,A4,FN)
IMPLICIT REAL*8 (S,T,U,F,A)
DIMENSION FN(20)
FN(2)=SQ*TM*UM
FN(4)=TQ*SP*UM
FN(6)=SQ*TP*UM
FN(8)=TQ*SM*UM
FN(9)=UQ*A1
FN(10)=UQ*A2
FN(11)=UQ*A3
FN(12)=UQ*A4
FN(14)=SQ*TM*UP
FN(16)=TQ*SP*UP
FN(18)=SQ*TP*UP
FN(20)=TQ*SM*UP
FN(1)=A1*UM-0.5*(FN(2)+FN(8)+FN(9))
FN(3)=A2*UM-0.5*(FN(4)+FN(2)+FN(10))
FN(5)=A3*UM-0.5*(FN(6)+FN(4)+FN(11))
FN(7)=A4*UM-0.5*(FN(8)+FN(6)+FN(12))
FN(13)=A1*UP-0.5*(FN(14)+FN(20)+FN(9))
FN(15)=A2*UP-0.5*(FN(16)+FN(14)+FN(10))
FN(17)=A3*UP-0.5*(FN(18)+FN(16)+FN(11))
FN(19)=A4*UP-0.5*(FN(20)+FN(18)+FN(12))
END

```

```

SUBROUTINE DS820(TM,TP,TQ,UM,UP,UQ,S,FNDS)
IMPLICIT REAL*8 (S,T,U,F,D)
DIMENSION FNDS(20)
DATA DSM,DSP/-0.5,0.5/
DSQ=-S-S
DSA=DSM*TM
DSB=-DSA
DSC=DSP*TP
DSD=-DSC
FNDS(2)=DSQ*TM*UM
FNDS(4)=DSP*TQ*UM
FNDS(6)=DSQ*TP*UM
FNDS(8)=DSM*TQ*UM
FNDS(9)=UQ*DSA
FNDS(10)=UQ*DSB
FNDS(11)=UQ*DSC
FNDS(12)=UQ*DSD
FNDS(14)=DSQ*TM*UP
FNDS(16)=DSP*TQ*UP
FNDS(18)=DSQ*TP*UP
FNDS(20)=DSM*TQ*UP
FNDS(1)=DSA*UM-0.5*(FNDS(2)+FNDS(8)+FNDS(9))
FNDS(3)=DSB*UM-0.5*(FNDS(4)+FNDS(2)+FNDS(10))
FNDS(5)=DSC*UM-0.5*(FNDS(6)+FNDS(4)+FNDS(11))
FNDS(7)=DSD*UM-0.5*(FNDS(8)+FNDS(6)+FNDS(12))
FNDS(13)=DSA*UP-0.5*(FNDS(14)+FNDS(20)+FNDS(9))
FNDS(15)=DSB*UP-0.5*(FNDS(16)+FNDS(14)+FNDS(10))
FNDS(17)=DSC*UP-0.5*(FNDS(18)+FNDS(16)+FNDS(11))
FNDS(19)=DSD*UP-0.5*(FNDS(20)+FNDS(18)+FNDS(12))
END

```

```

SUBROUTINE DT820(SM,SP,SQ,UM,UP,UQ,T,FNDT)
IMPLICIT REAL*8 (S,T,U,F,D)
DIMENSION FNDT(20)
DATA DTM,DTP/-0.5,0.5/
DTQ=-T-T
DTA=SM*DTM
DTB=SP*DTM
DTC=SP*DTP
DTD=SM*DTP
FNDT(2)=SQ*DTM*UM
FNDT(4)=SP*DTQ*UM
FNDT(6)=SQ*DTP*UM
FNDT(8)=SM*DTQ*UM
FNDT(9)=UQ*DTA
FNDT(10)=UQ*DTB
FNDT(11)=UQ*DTC
FNDT(12)=UQ*DTD
FNDT(14)=SQ*DTM*UP
FNDT(16)=SP*DTQ*UP
FNDT(18)=SQ*DTP*UP
FNDT(20)=SM*DTQ*UP
FNDT(1)=DTA*UM-0.5*(FNDT(2)+FNDT(8)+FNDT(9))
FNDT(3)=DTB*UM-0.5*(FNDT(4)+FNDT(2)+FNDT(10))
FNDT(5)=DTC*UM-0.5*(FNDT(6)+FNDT(4)+FNDT(11))
FNDT(7)=DTD*UM-0.5*(FNDT(8)+FNDT(6)+FNDT(12))
FNDT(13)=DTA*UP-0.5*(FNDT(14)+FNDT(20)+FNDT(9))
FNDT(15)=DTB*UP-0.5*(FNDT(16)+FNDT(14)+FNDT(10))
FNDT(17)=DTC*UP-0.5*(FNDT(18)+FNDT(16)+FNDT(11))
FNDT(19)=DTD*UP-0.5*(FNDT(20)+FNDT(18)+FNDT(12))
END

```

```

SUBROUTINE DU820(SM,SP,SQ,TM,TP,TQ,U,FNDU,A1,A2,A3,A4)
IMPLICIT REAL*8 (S,T,U,F,A,D)
DIMENSION FNDU(20)
DATA DUM,DUP/-0.5,0.5/
DUQ=-U-U
FNDU(2)=SQ*TM*DUM
FNDU(4)=SP*TQ*DUM
FNDU(6)=SQ*TP*DUM
FNDU(8)=SM*TQ*DUM
FNDU(9)=A1*DUQ
FNDU(10)=A2*DUQ
FNDU(11)=A3*DUQ
FNDU(12)=A4*DUQ
FNDU(14)=SQ*TM*DUP
FNDU(16)=SP*TQ*DUP
FNDU(18)=SQ*TP*DUP
FNDU(20)=SM*TQ*DUP
FNDU(1)=A1*DUM-0.5*(FNDU(2)+FNDU(8)+FNDU(9))
FNDU(3)=A2*DUM-0.5*(FNDU(4)+FNDU(2)+FNDU(10))
FNDU(5)=A3*DUM-0.5*(FNDU(6)+FNDU(4)+FNDU(11))
FNDU(7)=A4*DUM-0.5*(FNDU(8)+FNDU(6)+FNDU(12))
FNDU(13)=A1*DUP-0.5*(FNDU(14)+FNDU(20)+FNDU(9))
FNDU(15)=A2*DUP-0.5*(FNDU(16)+FNDU(14)+FNDU(10))
FNDU(17)=A3*DUP-0.5*(FNDU(18)+FNDU(16)+FNDU(11))
FNDU(19)=A4*DUP-0.5*(FNDU(20)+FNDU(18)+FNDU(12))
END

```

```

SUBROUTINE FN827(S,T,U,FN,SQ,TQ,UQ)
IMPLICIT REAL*8 (A,B,F,S,T,U)
DIMENSION FN(27),SQ(3),TQ(3),UQ(3),NUM(27)
DATA NUM/1,2,3,8,9,4,7,6,5,10,11,12,17,18,13,16,15,14,
1 19,20,21,26,27,22,25,24,23/

A=0.5
B=S*S
SQ(1)=A*(B-S)
SQ(2)=A+A-B
SQ(3)=A*(B+S)
B=T*T
TQ(1)=A*(B-T)
TQ(2)=A+A-B
TQ(3)=A*(B+T)
B=U*U
UQ(1)=A*(B-U)
UQ(2)=A+A-B
UQ(3)=A*(B+U)
L=0
DO 30 IU=1,3
    UQI=UQ(IU)
    DO 20 IT=1,3
        TQI=TQ(IT)
        DO 10 IS=1,3
            L=L+1
            N=NUM(L)
            FN(N)=UQI*TQI*SQ(IS)
10        CONTINUE
20    CONTINUE
30    CONTINUE
END

```

```

SUBROUTINE FD827(S,T,U,SQ,TQ,UQ,FNDS,FNDT,FNDU)
IMPLICIT REAL*8 (A,S,T,U,D,F)
DIMENSION FNDS(27),FNDT(27),FNDU(27),SQ(3),TQ(3),UQ(3)
DIMENSION DSQ(3),DTQ(3),DUQ(3),NUM(27)
DATA NUM/1,2,3,8,9,4,7,6,5,10,11,12,17,18,13,16,15,14,
1 19,20,21,26,27,22,25,24,23/

A=0.5
DSQ(1)=S-A
DSQ(2)=-S-S
DSQ(3)=S+A
DTQ(1)=T-A
DTQ(2)=-T-T
DTQ(3)=T+A
DUQ(1)=U-A
DUQ(2)=-U-U
DUQ(3)=U+A

L=0
DO 30 IU=1,3
    UQI=UQ(IU)
    DUQI=DUQ(IU)
    DO 20 IT=1,3
        TQI=TQ(IT)
        DTQI=DTQ(IT)

```

```

      DO 10 IS=1,3
        L=L+1
        N=NUM(L)
        FNDS(N)=UQI*TQI*DSQ(IS)
        FNDDT(N)=UQI*DTQI*SQ(IS)
        FNDDU(N)=DUQI*TQI*SQ(IS)
10      CONTINUE
20      CONTINUE
30      CONTINUE
      END

```

```

1  SUBROUTINE BRAUX(SM,SP,SQ,TM,TP,TQ,UM,UP,UQ,A1,A2,A3,A4,
      S,T,U)
      IMPLICIT REAL*8 (S,T,U,A)
      C=0.5
      D=1.0
      SM=C*(D-S)
      SP=C*(D+S)
      TM=C*(D-T)
      TP=C*(D+T)
      UM=C*(D-U)
      UP=C*(D+U)
      SQ=D-S*S
      TQ=D-T*T
      UQ=D-U*U
      A1=SM*TM
      A2=SP*TM
      A3=SP*TP
      A4=SM*TP
      END

```

```

SUBROUTINE PRAUX(AL,BL,CL,U,UM,UP,UQ,A1,A2,A3)
IMPLICIT REAL*8 (A,B,C,U)
A1=4.0*AL*BL
A2=4.0*BL*CL
A3=4.0*CL*AL
UM=0.5*(1.0-U)
UP=0.5*(1.0+U)
UQ=1.0-U*U
END

```

```

SUBROUTINE SLV301(MXPA,H,Q,ID,CGR,QD,INMAT,INEQU,QSUM)

REAL*8 CGR,ALM,CLM,CEQ,RHS,Q,H,CE,PIVOT,RHSEL,RHEL
REAL*8 CONST,SUM,HH,QSUM,QD
PARAMETER (IV=800)
DIMENSION H(1),Q(1),ID(1),CGR(1),QD(1),MVB(IV),LVB(27)
DIMENSION LDEST(27),ALM(27),CLM(380),CEQ(IV),RHS(IV)
NFUNC(I,J)=(J*J-J)/2+I

C    ASSEMBLING AND SOLVING THE EQUATIONS (PERMANENT FLOW)

QSUM=0.0D+0
NEQ=0
KURPA=0

LZ=(MXPA*MXPA-MXPA)/2+MXPA
DO 20 I=1,LZ
    CGR(I)=0.0D+0
20  CONTINUE

DO 22 I=1,MXPA
    MVB(I)=0
    RHS(I)=0.0D+0
    CEQ(I)=0.0D+0
22  CONTINUE

READ(INMAT)MXLM
DO 70 LM=1,MXLM
    READ(INMAT)NQ,NP,KR,LZ,(LVB(K),K=1,KR),(ALM(K),K=1,KR),
1    (LDEST(K),K=1,KR),(CLM(L),L=1,LZ)
    QAL=QD(NQ)

    DO 34 K=1,KR
        NIC=LVB(K)
        Q(NIC)=Q(NIC)+QAL*ALM(K)
        QSUM=QSUM+QAL*ALM(K)
        CALL CDEST(LDEST,LDES,NSTR,K)
        MVB(LDES)=NIC
        LVB(K)=LDES
        IF(KURPA.GE.LDES)GO TO 32
        KURPA=LDES
32    CONTINUE
34    CONTINUE

L=0
DO 42 KO=1,KR
    KGR=LVB(KO)
    MGO=NFUNC(0,KGR)
    DO 40 LI=1,KO
        LGR=LVB(LI)
        L=L+1
        CE=CLM(L)
        IF(KGR.GE.LGR)GO TO 36
        MG=NFUNC(KGR,LGR)
        GO TO 38
36    MG=MGO+LGR
38    CONTINUE
        CGR(MG)=CGR(MG)+CE
40    CONTINUE

```

```

C -----
C Est-ce qu'il y a des termes dans le triangle inferieur
C qu'il faudrait ajouter encore au terme diagonal?
C Au lieu d'examiner toutes les lignes de la colonne K0,
C on examine toutes les colonnes de la ligne situee dans
C le triangle superieur (symetrie!).
  LI=K0+1
  LL=L
  DO WHILE (LI.LE.KR)
    LL=LL+LI-1
    IF(LVB(LI).EQ.LVB(KO)) THEN
      CGR(MG)=CGR(MG)+CLM(LL)
    END IF
    LI=LI+1
  END DO
C -----
42 CONTINUE

DO 66 K=1,KR
  CALL CDEST(LDEST,LDES,NSTR,K)
  IF(NSTR.NE.0.AND.NSTR.NE.1)GO TO 64
  NDIAG=NFUNC(0,LDES+1)
  PIVOT=CGR(NDIAG)
C -----
  IF(PIVOT.EQ.0.0) THEN
    WRITE(16,4002)LM
    4002 FORMAT('/ Pivot est zero lors de l'elimination'/
1      ' de l'element ',I5,'. Calcul interrompu!')
    TYPE 4002,LM
    STOP
  END IF
C -----
  CGR(NDIAG)=0.0
  NIC=MVB(LDES)
  MVB(LDES)=0
  NEQ=NEQ+1
  RHSEL=RHS(LDES)+Q(NIC)
  RHS(LDES)=0.0
  IF(ID(NIC).LT.0)GO TO 44
  RHEL=RHSEL
  GO TO 46
44 RHEL=H(NIC)
46 CONTINUE

DO 58 KGR=1,KURPA
  MGO=NFUNC(0,KGR)
  IF(KGR.GE.LDES)GO TO 50
  MG=NDIAG-LDES+KGR
  GO TO 52
50 MG=MGO+LDES
52 CONTINUE
  CONST=CGR(MG)
  CGR(MG)=0.0
  CEQ(KGR)=CONST
  IF(ID(NIC).LT.0)GO TO 56
  CONST=CONST/PIVOT
  DO 54 I=1,KGR
    II=MGO+I
    CGR(II)=CGR(II)-CONST*CEQ(I)
54 CONTINUE
56 CONTINUE

```

```

      RHS(KGR)=RHS(KGR)-CONST*RHEL
58      CONTINUE
      CEQ(LDES)=PIVOT

      WRITE(INEQU`NEQ)NIC,LDES,KURPA,RHSEL,(CEQ(I),I=1,KURPA)

60      IF(MVB(KURPA).NE.0.OR.KURPA.EQ.1)GO TO 62
      KURPA=KURPA-1
      GO TO 60
62      CONTINUE
64      CONTINUE
66      CONTINUE
70      CONTINUE

      MXNEQ=NEQ
      DO 90 N=1,MXNEQ
      NEQ=MXNEQ-N+1
      READ(INEQU`NEQ)NIC,LDES,KURPA,RHSEL,(CEQ(I),I=1,KURPA)
      PIVOT=CEQ(LDES)
      CEQ(LDES)=0.0
      SUM=0.0
      DO 82 I=1,KURPA
      SUM=SUM+CEQ(I)*RHS(I)
82      CONTINUE
      IF(ID(NIC).GE.0)GO TO 84
      RHS(LDES)=H(NIC)
      Q(NIC)=SUM+PIVOT*RHS(LDES)-RHSEL
      QSUM=QSUM+Q(NIC)
      GO TO 86
84      HH=(RHSEL-SUM)/PIVOT
      RHS(LDES)=HH
      H(NIC)=HH
86      CONTINUE
90      CONTINUE

      END
```

```
$!      Command procedure FEM301.COM
$      TYPE SYS$INPUT
```

```
*****
* "FEM301" - MODELE A ELEMENTS FINIS          *
* ECOULEMENT PERMANENT - TRIDI                *
* CENTRE D'HYDROGEOLOGIE (UNIVERSITE DE NEUCHATEL) *
*****
```

Vous devez donner le nom des fichiers suivants:

1. Fichier des elements
2. Fichier des coordonnees
3. Fichier des parametres
4. Fichier des resultats
5. Fichier des equations

ATTENTION! Dans le fichier des elements on lit LM,NQ,NP,KR,NAR
ou NAR est le nombre d'aretes dans l'element!
Si NAR.GE.0: repetition des noeuds non admise dans l'element
Si NAR.LT.0: repetition des noeuds admise (reduction des aretes).

\$A1:

```
$      INQUIRE FIELM "  Nom du fichier des elements"
$      @U:EXIST `FIELM`
$      IF EXIST THEN GOTO A2
$      TYPE SYS$INPUT
```

Ce fichier n'existe pas! Controlez encore une fois
le nom, le type et le numero de version!

```
$      GOTO A1
```

\$A2:

```
$      INQUIRE FICOR "  Nom du fichier des coordonnees"
$      @U:EXIST `FICOR`
$      IF EXIST THEN GOTO A3
$      TYPE SYS$INPUT
```

Ce fichier n'existe pas! Controlez encore une fois
le nom, le type et le numero de version!

```
$      GOTO A2
```

\$A3:

```
$      INQUIRE FIPAR "  Nom du fichier des parametres"
$      @U:EXIST `FIPAR`
$      IF EXIST THEN GOTO A4
$      TYPE SYS$INPUT
```

Ce fichier n'existe pas! Controlez encore une fois
le nom, le type et le numero de version!

```
$      GOTO A3
```

\$A4:

```
$      INQUIRE FIRES "  Nom du fichier des resultats"
$      INQUIRE FIEQU "  Nom du fichier des equations"
$      REP:=`F$LOGICAL("SYS$DISK")`F$DIRECTORY()
$      INQUIRE ANS "EXECUTION INTERACTIVE(I) OU EN BATCH(B)"
$      IF ANS .EQS. "B" THEN GOTO A10
$      @GEOL:[KIRALY.FEM.AUX]FEM301A `FIELM` `FICOR` `FIPAR` `FIRES` -
$      `REP` `FIEQU`
```

```
$EXIT
```

\$A10:

```
$      SUBMIT/PARAMETERS=(`FIELM`,`FICOR`,`FIPAR`,`FIRES`,`REP`, -
$      `FIEQU`) GEOL:[KIRALY.FEM.AUX]FEM301A
```

```
$EXIT
```

```
$ !      Command procedure FEM301A.COM
$ !      Cette procedure est appelee par FEM301.COM qui transmet
$ !      les 5 parametres: P1=fichier des elements
$ !                               P2=fichier des coordonnees
$ !                               P3=fichier des parametres
$ !                               P4=fichier des resultats
$ !                               P5=nom du repertoire utilise
$ !                               P6=fichier des equations
$ !      Dans cette version plusieurs noeuds d'un meme element
$ !      peuvent avoir le meme numero (contraction des aretes).
$      SET DEF 'P5'
$      ASSIGN/USER 'P1' FOR011
$      ASSIGN/USER 'P2' FOR012
$      ASSIGN/USER 'P3' FOR013
$      ASSIGN/USER 'P4' FOR016
$      ASSIGN/USER 'P6' FOR015
$      RUN GEOL:[KIRALY.FEM.AUX]FEM301
$EXIT
```