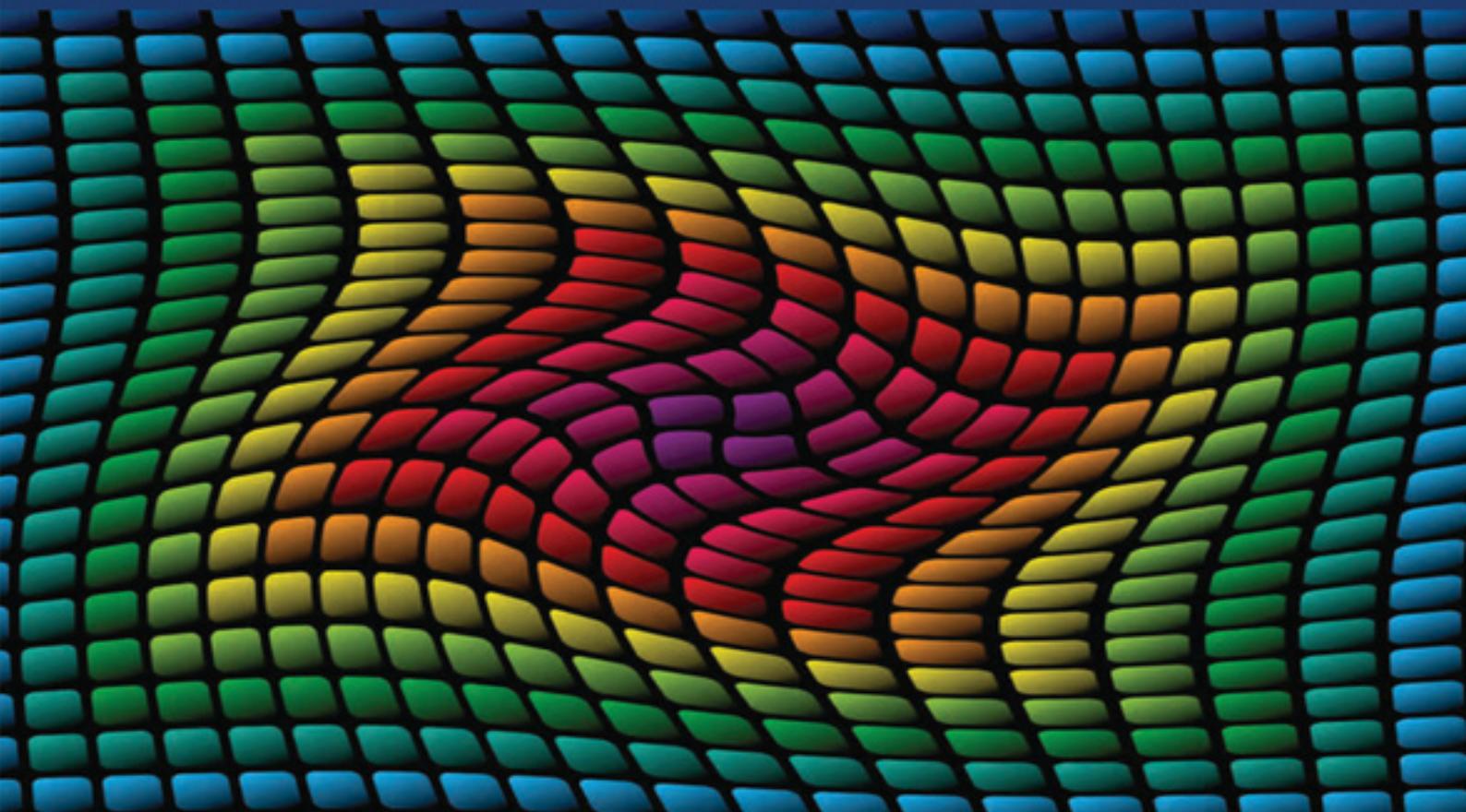


NUMERICAL METHODS SERIES

Finite Element Method



**Gouri Dhatt, Gilbert Touzot
Emmanuel Lefrançois**

ISTE

WILEY

Finite Element Method

Finite Element Method

Gouri Dhatt
Gilbert Touzot
Emmanuel Lefrançois

Series Editor
Piotr Breitkopf



First published 2012 in Great Britain and the United States by ISTE Ltd and John Wiley & Sons, Inc.

Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act 1988, this publication may only be reproduced, stored or transmitted, in any form or by any means, with the prior permission in writing of the publishers, or in the case of reprographic reproduction in accordance with the terms and licenses issued by the CLA. Enquiries concerning reproduction outside these terms should be sent to the publishers at the undermentioned address:

ISTE Ltd
27-37 St George's Road
London SW19 4EU
UK

www.iste.co.uk

John Wiley & Sons, Inc.
111 River Street
Hoboken, NJ 07030
USA

www.wiley.com

© ISTE Ltd 2012

The rights of Gouri Dhatt, Gilbert Touzot and Emmanuel Lefrançois to be identified as the author of this work have been asserted by them in accordance with the Copyright, Designs and Patents Act 1988.

Library of Congress Control Number: 2012946444

British Library Cataloguing-in-Publication Data
A CIP record for this book is available from the British Library
ISBN 978-1-84821-368-5

Printed and bound in Great Britain by CPI Group (UK) Ltd., Croydon, Surrey CR0 4YY



TABLE OF CONTENTS

Introduction	1
0.1 The finite element method	1
0.1.1 General remarks	1
0.1.2 Historical evolution of the method	2
0.1.3 State of the art	3
0.2 Object and organization of the book	3
0.2.1 Teaching the finite element method	3
0.2.2 Objectives of the book	4
0.2.3 Organization of the book	4
0.3 Numerical modeling approach	6
0.3.1 General aspects	6
0.3.2 Physical model	7
0.3.3 Mathematical model	9
0.3.4 Numerical model	10
0.3.5 Computer model	13
Bibliography	16
Conference proceedings	17
Monographs	18
Periodicals	19
Chapter 1. Approximations with finite elements	21
1.0 Introduction	21
1.1 General remarks	21
1.1.1 Nodal approximation	21
1.1.2 Approximations with finite elements	28
1.2 Geometrical definition of the elements	33
1.2.1 Geometrical nodes	33
1.2.2 Rules for the partition of a domain into elements	33
1.2.3 Shapes of some classical elements	35
1.2.4 Reference elements	37
1.2.5 Shapes of some classical reference elements	41
1.2.6 Node and element definition tables	44
1.3 Approximation based on a reference element	45
1.3.1 Expression of the approximate function $u(x)$	45
1.3.2 Properties of approximate function $u(x)$	49

1.4 Construction of functions $N(\xi)$ and $\bar{N}(\xi)$	54
1.4.1 General method of construction	54
1.4.2 Algebraic properties of functions N and \bar{N}	59
1.5 Transformation of derivation operators	61
1.5.1 General remarks	61
1.5.2 First derivatives	62
1.5.3 Second derivatives	65
1.5.4 Singularity of the Jacobian matrix	68
1.6 Computation of functions N , their derivatives and the Jacobian matrix	72
1.6.1 General remarks	72
1.6.2 Explicit forms for N	73
1.7 Approximation errors on an element	75
1.7.1 Notions of approximation errors	75
1.7.2 Error evaluation technique	80
1.7.3 Improving the precision of approximation	83
1.8 Example of application: rainfall problem	89
Bibliography	95
Chapter 2. Various types of elements	97
2.0 Introduction	97
2.1 List of the elements presented in this chapter	97
2.2 One-dimensional elements	99
2.2.1 Linear element (two nodes, C^0)	99
2.2.2 High-precision Lagrangian elements: (continuity C^0)	101
2.2.3 High-precision Hermite elements	105
2.2.4 General elements	109
2.3 Triangular elements (two dimensions)	111
2.3.1 Systems of coordinates	111
2.3.2 Linear element (triangle, three nodes, C^0)	113
2.3.3 High-precision Lagrangian elements (continuity C^0)	115
2.3.4 High-precision Hermite elements	123
2.4 Quadrilateral elements (two dimensions)	127
2.4.1 Systems of coordinates	127
2.4.2 Bilinear element (quadrilateral, 4 nodes, C^0)	128
2.4.3 High-precision Lagrangian elements	129
2.4.4 High-precision Hermite element	134
2.5 Tetrahedral elements (three dimensions)	137
2.5.1 Systems of coordinates	137
2.5.2 Linear element (tetrahedron, four nodes, C^0)	139
2.5.3 High-precision Lagrangian elements (continuity C^0)	140
2.5.4 High-precision Hermite elements	142
2.6 Hexahedric elements (three dimensions)	143
2.6.1 Trilinear element (hexahedron, eight nodes, C^0)	143
2.6.2 High-precision Lagrangian elements (continuity C^0)	144
2.6.3 High-precision Hermite elements	150

2.7 Prismatic elements (three dimensions)	150
2.7.1 Element with six nodes (prism, six nodes, C^0)	150
2.7.2 Element with 15 nodes (prism, 15 nodes, C^0)	151
2.8 Pyramidal element (three dimensions)	152
2.8.1 Element with five nodes	152
2.9 Other elements	153
2.9.1 Approximation of vectorial values	153
2.9.2 Modifications of the elements	155
2.9.3 Elements with a variable number of nodes	156
2.9.4 Superparametric elements	158
2.9.5 Infinite elements	158
Bibliography	160
Chapter 3. Integral formulation	161
3.0 Introduction	161
3.1 Classification of physical systems	163
3.1.1 Discrete and continuous systems	163
3.1.2 Equilibrium, eigenvalue and propagation problems	164
3.2 Weighted residual method	172
3.2.1 Residuals	172
3.2.2 Integral forms	173
3.3 Integral transformations	174
3.3.1 Integration by parts	174
3.3.2 Weak integral form	177
3.3.3 Construction of additional integral forms	179
3.4 Functionals	182
3.4.1 First variation	182
3.4.2 Functional associated with an integral form	183
3.4.3 Stationarity principle	187
3.4.4 Lagrange multipliers and additional functionals	188
3.5 Discretization of integral forms	194
3.5.1 Discretization of W	194
3.5.2 Approximation of the functions \mathbf{u}	197
3.5.3 Choice of the weighting functions ψ	198
3.5.4 Discretization of a functional (Ritz method)	205
3.5.5 Properties of the systems of equations	208
3.6 List of PDEs and weak expressions	209
3.6.1 Scalar field problems	210
3.6.2 Solid mechanics	213
3.6.3 Fluid mechanics	217
Bibliography	229
Chapter 4. Matrix presentation of the finite element method	231
4.0 Introduction	231
4.1 The finite element method	231
4.1.1 Finite element approach	231
4.1.2 Conditions for convergence of the solution	243

4.1.3 Patch test	256
4.2 Discretized elementary integral forms W^e	264
4.2.1 Matrix expression of W^e	264
4.2.2 Case of a nonlinear operator \mathfrak{L}	267
4.2.3 Integral form W^e on the reference element	269
4.2.4 A few classic forms of W^e and of elementary matrices	274
4.3 Techniques for calculating elementary matrices	274
4.3.1 Explicit calculation for a triangular element (Poisson's equation)	274
4.3.2 Explicit calculation for a quadrangular element (Poisson's equation) .	279
4.3.3 Organization of the calculation of the elementary matrices by numerical integration	280
4.3.4 Calculation of the elementary matrices: linear problems	282
4.4 Assembly of the global discretized form W	297
4.4.1 Assembly by expansion of the elementary matrices	298
4.4.2 Assembly in structural mechanics	303
4.5 Technique of assembly	305
4.5.1 Stages of assembly	305
4.5.2 Rules of assembly	305
4.5.3 Example of a subprogram for assembly	307
4.5.4 Construction of the localization table LOCE	308
4.6 Properties of global matrices	310
4.6.1 Band structure	310
4.6.2 Symmetry	314
4.6.3 Storage methods	314
4.7 Global system of equations	318
4.7.1 Expression of the system of equations	318
4.7.2 Introduction of the boundary conditions	318
4.7.3 Reactions	321
4.7.4 Transformation of variables	321
4.7.5 Linear relations between variables	323
4.8 Example of application: Poisson's equation	324
4.9 Some concepts about convergence, stability and error calculation	329
4.9.1 Notations	329
4.9.2 Properties of the exact solution	330
4.9.3 Properties of the solution obtained by the finite element method .	331
4.9.4 Stability and locking	334
4.9.5 One-dimensional exact finite elements	337
Bibliography	343
Chapter 5. Numerical Methods	345
5.0 Introduction	345
5.1 Numerical integration	346
5.1.1 Introduction	346
5.1.2 One-dimensional numerical integration	348
5.1.3 Two-dimensional numerical integration	360
5.1.4 Numerical integration in three dimensions	368

TABLE OF CONTENTS ix

5.1.5 Precision of integration	372
5.1.6 Choice of number of integration points	375
5.1.7 Numerical integration codes	379
5.2 Solving systems of linear equations	384
5.2.1 Introduction	384
5.2.2 Gaussian elimination method	385
5.2.3 Decomposition	391
5.2.4 Adaptation of algorithm (5.44) to the case of a matrix stored by the skyline method	399
5.3 Solution of nonlinear systems	404
5.3.1 Introduction	404
5.3.2 Substitution method	407
5.3.3 Newton–Raphson method	411
5.3.4 Incremental (or step-by-step) method	420
5.3.5 Changing of independent variables	421
5.3.6 Solution strategy	424
5.3.7 Convergence of an iterative method	426
5.4 Resolution of unsteady systems	429
5.4.1 Introduction	429
5.4.2 Direct integration methods for first-order systems	431
5.4.3 Modal superposition method for first-order systems	463
5.4.4 Methods for direct integration of second-order systems	466
5.4.5 Modal superposition method for second-order systems	476
5.5 Methods for calculating the eigenvalues and eigenvectors	480
5.5.1 Introduction	480
5.5.2 Recap of some properties of eigenvalue problems	481
5.5.3 Methods for calculating the eigenvalues	488
Bibliography	502
Chapter 6. Programming technique	505
6.0 Introduction	505
6.1 Functional blocks of a finite element program	506
6.2 Description of a typical problem	507
6.3 General programs	508
6.3.1 Possibilities of general programs	508
6.3.2 Modularity	511
6.4 Description of the finite element code	512
6.4.1 Introduction	512
6.4.2 General organization	513
6.4.3 Description of tables and variables	517
6.5 Library of elementary finite element method programs	521
6.5.1 Functional blocks	521
6.5.2 List of thermal elements	530
6.5.3 List of elastic elements	538
6.5.4 List of elements for fluid mechanics	545

6.6 Examples of application	549
6.6.1 Heat transfer problems	550
6.6.2 Planar elastic problems	558
6.6.3 Fluid flow problems	566
Appendix. Sparse solver	577
Sofiane HADJI	
7.0 Introduction	577
7.1 Methodology of the sparse solver	578
7.1.1 Assembly of matrices in sparse form: row-by-row format	579
7.1.2 Permutation using the “minimum degree” algorithm	584
7.1.3 Modified column–column storage format	587
7.1.4 Symbolic factorization	589
7.1.5 Numerical factorization	590
7.1.6 Solution of the system by descent/ascent	592
7.2 Numerical examples	593
Bibliography	595
Index	597

Introduction

0.1 The finite element method

0.1.1 GENERAL REMARKS

Modern technological advances challenge engineers to carry out increasingly complex and costly projects, which are subject to severe reliability and safety constraints. These projects cover domains such as space travel, aeronautics and nuclear applications, where reliability and safety are of crucial importance. Other projects are connected with the protection of the environment, for example control of thermal, acoustic or chemical pollution, water course management, management of groundwater and weather forecasting. For a proper understanding, analysts need mathematical models that allow them to simulate the behavior of complex physical systems. These models are then used during the design phase of the projects.

Engineering sciences (mechanics of solids and fluids, thermodynamics, etc.) are used to describe the behavior of physical systems in the form of partial differential equations. Today, the finite element method has become one of the most frequently used methods for solving such equations. This method requires intensive use of a computer, and can be applied to solve almost all problems encountered in practice: steady or transient problems in linear and nonlinear regions for one-, two- and three-dimensional domains. Moreover, it can be successfully adapted for use in the heterogeneous environments and domains of complex forms often encountered in practice by engineers.

The finite element method consists of using a simple approximation of unknown variables to transform partial differential equations into algebraic equations. It draws on the following three disciplines:

- engineering sciences to describe physical laws (partial differential equations);
- numerical methods for the elaboration and solution of algebraic equations;
- computing tools to carry out the necessary calculations efficiently using a computer.

0.1.2 HISTORICAL EVOLUTION OF THE METHOD

Structural mechanics allows us to analyze frames and trusses. The behavior of each truss or beam element is represented by an elementary stiffness matrix constructed using knowledge of the strength of materials. Using these matrices, we are able to construct a system of algebraic equations verifying the conditions of displacement continuity and balance of forces at the nodes. The solution of the system of equations corresponding to applied loads leads to the displacements of all nodes in the structure. The emergence of computers and the requirements of the aeronautical industry led to rapid developments in the field of structural mechanics in the 1950s. The concept of finite elements was introduced by Turner, Clough, Martin and Topp [TUR 56] in 1956: they represented an elastic two-dimensional domain by an assembly of triangular panels across which displacements are presumed to vary in a linear manner. The behavior of each panel is represented by an elementary stiffness matrix. Structural mechanics tools are then employed to obtain nodal displacements under different applied loads and boundary conditions.

We should also highlight the work carried out by Argyris and Kelsey [ARG 60], who employed the notion of energy in structural analysis. The basic ideas involved in the finite element method, however, appeared earlier, in an article published by Courant in 1943 [COU 43].

From 1960 onward, finite element method developed rapidly in a number of directions:

- The method was reformulated, based on energetic and variational considerations, in the general form of weighted residuals or weak formulations [ZIE 65; GRE 69; FIN 75; ARA 68].
- A number of authors created high-precision elements [FEL 66], curved elements and isoparametric elements [ERG 68; IRO 68].
- The finite element method was recognized as a general method of solution for partial differential equations. It thus came into use in solving nonlinear and transient problems of structures, as well as in other fields, such as soil and rock mechanics (geomechanics), fluid mechanics and thermodynamics [PRO 01–PRO 13].
- A mathematical basis for finite element method was established using concepts of functional analysis [PRO 14–PRO 15].

Since 1967, many books have been published on the finite element method [MON 01–MON 29]. We wish to highlight, in particular, the three editions of the book by Zienkiewicz [MON 02], which are available throughout the world. We may refer to Pironneau, Gérardin, Imbert, Batoz-Dhatt and

Dhatt-Touzot, among others, for books available in French. An exhaustive list of reference works in the domain of finite elements may be easily obtained using an Internet search engine.

0.1.3 STATE OF THE ART

The finite element method (FEM) is nowadays widely used in industrial applications, including aeronautical, aerospace, automobile, naval and nuclear construction fields, and in applications of fluid mechanics, including tidal studies, sedimentary transportation, the study of thermal or chemical pollution phenomena and fluid-structure interactions. A number of general-purpose computer codes are available for industrial users of the finite element method, such as IDEAS, SAMCEF, NASTRAN, ABAQUS, FIDAP, MARC, ANSYS, ADINA, LSDYNA, ASTER and CASTEM.

In order for the finite element method to be effective in industrial applications, computer codes must be used to assist in the preparation of data and the interpretation of results. These pre- and post-processing tools are usually integrated into more general computer-aided design (CAD) software packages, such as IDEAS, CASTOR or CATIA.

0.2 Object and organization of the book

0.2.1 TEACHING THE FINITE ELEMENT METHOD

The finite element method is now widely taught at both undergraduate and postgraduate level. The teaching of the finite element method requires a multidisciplinary approach involving different aspects:

- understanding of the physical problem and intuitive knowledge of the nature of the solution being sought;
- representation of the physical phenomenon in the form of partial differential equations and weak “variational” or “integral” formulations;
- discretization techniques to produce a discrete or algebraic model;
- matricial organization of data;
- numerical methods for integration of functions with several variables solution of linear and nonlinear algebraic equations;
- computer programming tools for handling massive data files and for creating user friendly graphic interface.

4 FINITE ELEMENT METHOD

It is hard to conceive a balanced formation in all these diverse disciplines. Moreover, suitable teaching software must be used that includes certain characteristics of general industrial codes. Finally the practical aspects of implementing the finite element method in computer codes must not be overlooked.

0.2.2 OBJECTIVES OF THE BOOK

This book attempts to simplify the teaching of the finite element method by smoothing out certain difficulties. It has been developed by engineers to solve engineering problems. Thus, the presentation of the book is primarily addressed to this audience. The mathematical knowledge required is limited to the domains of differential and matrix calculus.

This book is intended for readers who wish to understand the finite element method and apply it to solve engineering problems using a computer. Moreover, it should be of use to students and researchers in applied sciences and as well as to practicing engineers who wish to go beyond the basic level of knowledge implied by the use of “black box” programs.

0.2.3 ORGANIZATION OF THE BOOK

This volume is organized into six chapters, each providing a relatively independent presentation of various concepts of the finite element method as well as the corresponding numerical and programming techniques. Examples are provided for illustrative purposes, accompanied by simple programs written using Matlab[®].

Chapter 1

This chapter presents the approximation of continuous functions over subdomains in terms of nodal values and introduces the concepts of interpolation functions, reference elements, geometrical transformations and approximation error.

Chapter 2

This chapter presents the interpolation functions for classical elements in one, two and three dimensions.

Chapter 3

This chapter gives a description of the weighted residual method that allows us to obtain weak formulations (known as integral formulations) associated with partial differential equations (known as strong formulations).

Chapter 4

This chapter presents the matrix formulation of the finite element method, which consists of discretizing the integral formulation from Chapter 3, using the approximations of unknown functions from Chapters 1 and 2. We introduce notions of elementary matrices and vectors, assembly and global matrices and vectors.

Chapter 5

This chapter describes the numerical methods needed to construct and solve the systems of algebraic equations formed in Chapter 4: numerical methods of integration, methods for the solution of linear and nonlinear algebraic systems domain, methods for integrating first- and second-order propagation systems in time domain and methods for calculating the eigenvalues and vectors.

Chapter 6

This chapter provides a brief overview of the finite difference and finite volume methods as well as the computing techniques that are characteristic of the finite element method using a simple program written in Matlab[®].

Figure 0.1 shows the logical sequence of these chapters. Note that Chapters 1, 3 and 4 are devoted to the fundamental concepts underlying the finite element method, while Chapters 2 and 5 are intended as reference chapters, and finally Chapter 6 presents a simple program for illustrative purposes.

6 FINITE ELEMENT METHOD

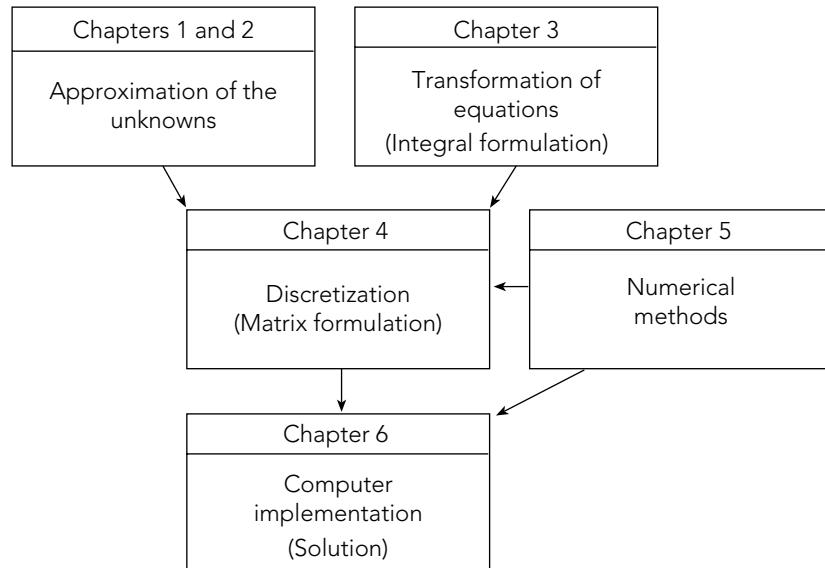


Figure 0.1. Logical flow of the chapters

0.3 Numerical modeling approach

0.3.1 GENERAL ASPECTS

This section gives a brief introduction to the different concepts to be covered in the following chapters.

Numerical modeling is used to simulate the behavior of physical systems using computers. This involves:

- description, in engineering terms, of the physical system in question and the problem under study (**physical model**);
- translation of the engineering problem into a mathematical form (**mathematical model**);
- construction of a **numerical model** (or algebraic model) that can be solved using a computer, and which uses discretization methods such as the finite element method;
- development of a computer code to simulate the behavior of the physical system (**computer model**).

A variety of errors may be introduced into different models or during the passage from one model to another. Three main types of errors are encountered:

- Errors in the choice of the mathematical model, representing the difference between the exact solution to the mathematical model and the real behavior of the physical system.
- Discretization errors, representing the difference between the exact solution to the mathematical model and the exact solution to the numerical model.
- Computer-based errors due to the limited precision of the calculations carried out by the computer and, potentially, programming errors.

Modeling specialists should be able to master these errors so that the solution provided by the software is reasonably close to the real behavior of the physical system under study (*a priori* unknown). In practice, it is often necessary to carry out the steps described above more than once until a satisfactory solution is produced.

0.3.2 PHYSICAL MODEL

The description of a physical system includes:

- a representation of its geometry;
- the selection of unknown variables for which we wish to evaluate spatio-temporal variations;
- the physical laws governing the system's behavior;
- values for the physical properties that are assumed to be known;
- applied forces, boundary conditions and, where applicable, initial conditions for transient problems.

EXAMPLE 0.1. Thermal equilibrium of a truss (physical model)

- | | |
|--|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| | <ul style="list-style-type: none"> — <i>Geometry: rectilinear truss of length L and rectangular section A oriented in the direction x (Figure 0.2).</i> |
|--|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

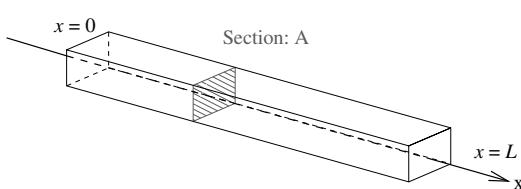


Figure 0.2. Rectilinear truss of constant section

— *Unknown variables:*

- the temperature (in degrees Celsius or Kelvin), $T(x)$;
- the thermal flux component as a function of x (W/m^2), $q(x)$;

The problem represents steady state flow; variables are thus independent of time.

— *Physical laws:*

- conservation of thermal flow as a function of x ;
- Fourier's law of thermal behavior relating the temperature gradient to the flow.

— *Physical properties: thermal conductivity, k ($W/\text{°C}\cdot m$)*.

— *Thermal load from the Joule effect (electrical current): f (W/m^3)*.

— *Boundary conditions:*

- *Imposed temperature:* $T(0) = T_0$.
- *Imposed flux:* $q(L) = q_L$, W/m^2 .

Objectives

Obtain $T(x)$ and $q(x)$ that verify the physical laws and boundary conditions. One possible application would be to estimate heat loss through the wall of a dwelling for which we wish to improve the insulation, i.e. limit the thermal flow.

0.3.3 MATHEMATICAL MODEL

This model is obtained by expressing the laws of conservation and constitutive laws in the form of partial differential equations. As this problem is to be solved using the finite element method, it will also be necessary to give an integral formulation (or weak formulation).

EXAMPLE 0.2. Application to the thermal equilibrium of a truss (mathematical model)

— Law of conservation of thermal flow written as a function of x for Example 0.1:

$$\frac{d(qA)}{dx} - f_0 A = 0, \quad f_0 > 0: \text{source of volumetric heat}$$

— Fourier's constitutive law: $q(x) = -k \frac{dT(x)}{dx}$.

— Boundary conditions:

$$T(x=0) = T_0, \quad : \text{Dirichlet}$$

$$q(x=L) = -k \frac{dT}{dx} \Big|_{x=L} = q_L > 0 \quad (\text{loss}) : \text{Neumann}$$

These relations may also be written in the form (for a constant section A):

$$\frac{d}{dx} \left(k \frac{dT(x)}{dx} \right) + f_0 = 0,$$

The exact solution to the mathematical model in the present case (where k is constant) is:

$$T(x) = T_0 - \frac{q_L}{k} x + \frac{f_0}{k} \left(Lx - \frac{x^2}{2} \right).$$

The integral form obtained using the weighted residual method is written as:

$$W = \int_0^L \frac{d\delta T}{dx} k \frac{dT}{dx} dx + W_{\text{Neu}} - \int_0^L \delta T \cdot f_0 dx = 0,$$

with $W_{\text{Neu}} = \delta T(L) \cdot q_L$ and $\delta T(x=0) = 0$.

where $\delta T(x)$ is any test function.

The solution to the problem is the function $T(x)$, such that $W=0$ for all test functions.

0.3.4 NUMERICAL MODEL

The numerical model associated with the mathematical model is obtained using a discretization method, such as:

- the finite difference method, or
- the finite element method.

In this case, we will illustrate the numerical model using the finite difference method.

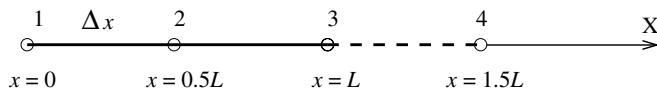
EXAMPLE 0.3. Application to the thermal equilibrium of a truss (numerical model based on the finite difference method)

For cases where “ k ” is constant, the differential equation governing the thermal equilibrium of the truss is written as:

$$k \frac{d^2T(x)}{dx^2} + f_0 = 0,$$

$$T(x=0) = 0 \text{ and } k \frac{dT}{dx} \Big|_L = -q_L.$$

Let us take a set of equidistant discretization points (known as nodes) across the domain. This may be illustrated using three equidistant nodes.



This set of nodes is known as a mesh. A fourth, “fictional” node has been added in order to give the same level of spatial precision for the boundary condition at $x = L$.

The equilibrium relationship is applied at each node, “ i ”:

$$k \frac{d^2T(x)}{dx^2} \Big|_i + f_0 = 0, \quad i = 1, 2, 3.$$

Let us associate an unknown variable with each node in the mesh, so that:

$$T(x_1 = 0) = T_1, \quad T(x_2 = \Delta x) = T_2,$$

$$T(x_3 = L) = T_3, \quad T(x_4 = L + \Delta x) = T_4,$$

where $\Delta x = L/2$ is the distance between two successive nodes.

The finite difference method consists of rewriting the derivatives in discrete form, so that:

$$\begin{aligned}\frac{d^2T}{dx^2}\Big|_x &= \frac{d^2T}{dx^2}\Big|_i \approx \frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta x^2}, \\ \frac{dT}{dx}\Big|_x &= \frac{dT}{dx}\Big|_i \approx \frac{T_{i+1} - T_{i-1}}{2\Delta x}.\end{aligned}$$

We thus obtain the discrete form of the thermal equilibrium equation at node "i":

$$k \frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta x^2} + f_0 = 0, \quad i = 2, \dots, 3.$$

Its application to nodes 2 and 3, associated with the boundary condition on node 1, translates as:

$$\begin{aligned}T_1 &= T_0, \\ T_1 - 2T_2 + T_3 + \frac{\Delta x^2 f_0}{k} &= 0, \\ T_2 - 2T_3 + T_4 + \frac{\Delta x^2 f_0}{k} &= 0.\end{aligned}$$

The boundary condition for $x = L$ gives

$$\frac{T_4 - T_2}{2\Delta x} = \frac{-q_L}{k} \Rightarrow T_4 = T_2 - \frac{2\Delta x q_L}{k}.$$

Organizing these relations in a matrix form leads to

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} T_0 \\ f_0 \Delta x^2 / k \\ 0.5 f_0 \Delta x^2 / k - q_L \Delta x / k \end{bmatrix},$$

which gives

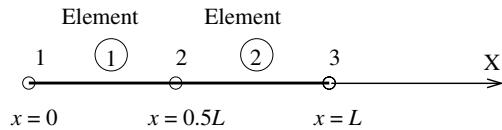
$$\begin{bmatrix} T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} T_0 + 1.5 f_0 \Delta x^2 / k - q_L \Delta x / k \\ T_0 + 2 f_0 \Delta x^2 / k - 2 q_L \Delta x / k \end{bmatrix}.$$

Remark

In this case (where f_0 is constant), we observe that the solution to the numerical model coincides with that of the mathematical model at the nodes.

EXAMPLE 0.4. Approach based on the finite element method

The finite element method consists of constructing a discrete representation of the integral form W of Example 0.2. To do this, we first select a set of two elements as illustrated below.



The integral form is written as:

$$W = \sum_{e=1}^2 W^e + W_{\text{Neu}} = 0,$$

where $W^e = \int_{x_i}^{x_{i+1}} \frac{d\delta T(x)}{dx} k \frac{dT(x)}{dx} dx - \int_{x_i}^{x_{i+1}} \delta T(x) f_0 dx$ and $W_{\text{Neu}} = \delta T(L) q_L$.

For each element, we choose a linear approximation of the solution function $T(x)$ and the test function $\delta T(x)$ with $\delta T(x=0)=0$.

For element 1: $L^e = x_2 - x_1 = \frac{L}{2}$.

$$\begin{aligned} T(x) &= \left(\frac{x_2 - x}{L^e} \right) T_1 + \left(\frac{x - x_1}{L^e} \right) T_2, \quad x_1 = 0, \quad x_2 = L/2, \\ \delta T(x) &= \left(\frac{x_2 - x}{L^e} \right) \delta T_1 + \left(\frac{x - x_1}{L^e} \right) \delta T_2. \end{aligned}$$

where $T_1 = T_0$ and $\delta T_1 = 0$.

The elementary integral form associated with element 1 is then written as:

$$W^1 = \langle \delta T_1 \delta T_2 \rangle \left(\frac{k}{L^e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} - f_0 \frac{L^e}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right).$$

For element 2: $L^e = x_3 - x_2 = \frac{L}{2}$.

The approach taken is equivalent to that used for element 1. The elementary integral form is written as:

$$W^2 = <\delta T_2 \delta T_3> \left(\frac{k}{L^e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} T_2 \\ T_3 \end{Bmatrix} - f_0 \frac{L^e}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} \right).$$

The flow term is expressed as $W_{\text{Neu}} = \delta T_3 \cdot q_L$.

After assembly, the integral form W is written as:

$$W = <\delta T_1 \delta T_2 \delta T_3> \left(\frac{2k}{L} \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \end{Bmatrix} - f_0 \frac{L}{4} \begin{Bmatrix} 1 \\ 2 \\ 1 \end{Bmatrix} + \begin{Bmatrix} 0 \\ 0 \\ q_L \end{Bmatrix} \right) = 0.$$

Introducing the boundary condition at node 1, we obtain the following system:

$$\frac{2k}{L} \begin{bmatrix} 2 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} T_2 \\ T_3 \end{Bmatrix} = f_0 \frac{L}{4} \begin{Bmatrix} 2 \\ 1 \end{Bmatrix} - \begin{Bmatrix} -2kT_0 / L \\ q_L \end{Bmatrix},$$

leading to $\begin{Bmatrix} T_2 \\ T_3 \end{Bmatrix} = \begin{Bmatrix} T_0 + 1.5 f_0 \Delta x^2 / k - q_L \Delta x / k \\ T_0 + 2 f_0 \Delta x^2 / k - 2 q_L \Delta x / k \end{Bmatrix}$.

Remark

In this particular case, the finite difference and finite element methods provide exactly identical solutions at the nodes.

0.3.5 COMPUTER MODEL

The two programs given in Figures 0.3 and 0.4 are written in the Matlab[©] programming language and cover Examples 0.3 and 0.4, respectively.

14 FINITE ELEMENT METHOD

```
%----- initialization
clear all

%----- geometry
L=1; % length (m)
nnt=20; % number of nodes
dx=L/(nnt-1); % discretization dx

%----- properties
kd=2; % thermal conductivity
f0=50; % heat production per unit of length

%----- boundary conditions
T0=30; % Dirichlet at node 1
qL=10; % Neumann at node nnt

%----- construction of system of equations
vkg=zeros(nnt,nnt); % initialization of the vkg global matrix
vfg=zeros(nnt,1); % initialization of the vfg global vector
%----- node loop
if(nnt>2)
    for i=2:nnt-1
        vfg(i)=f0*dx^2/kd;
        vkg(i,[i-1, i, i+1])=[-1, 2, -1];
    end
end
%----- Dirichlet boundary condition (x=0)
vkg(1,1)=1; vfg(1)=T0;

%----- Neumann boundary condition (x=L)
vkg(nnt,[nnt-1 nnt])=[-1 1]; vfg(nnt)=0.5*f0*dx^2/kd-qL*dx/kd;
%----- solution
vsol=vkg\vfg;
%----- display numerical solution
plot([0:nnt-1]*dx,vsol)
```

Figure 0.3. 1D program using the finite difference method (Example 0.3)

```

%----- initialization
clear all
%----- geometry
L=1; % length (m)
nnt=20; % number of nodes
Le=L/(nnt-1); % discretization dx=Le
%----- properties
kd=2; % thermal conductivity
f0=50; % heat production per unit of length
%----- boundary conditions
T0=30; % Dirichlet at node 1
qL=10; % Neumann at node nnt
%----- construction of system of equations
vkg=zeros(nnt,nnt); % initialization of vkg
vfg=zeros(nnt,1); % initialization of vfg
c=kd/Le;
%----- elementary matrix and vector
vke=[c, -c; -c c];
vfe=f0*Le/2*[1; 1]
%----- element loop
for ie=1:nelt
    vfg([ie ie+1])= vfg([ie ie+1])+vfe;
    vkg([ie ie+1], [ie ie+1])= vkg([ie ie+1], [ie ie+1])+vke;
end
%----- Dirichlet boundary condition (x=0)
vkg(1,:)=zeros(1,nnt); vkg(1, 1)=1; vfg(1)=T0
%----- Neumann boundary condition (x=L)
vkg(nnt)= vfg(nnt -qL;
%----- solution
vsol=vkg\vgf;
%----- display numerical solution and exact solution
plot([0:nnt-1]*Le,vsol)
hold on
x=0:L/100:L;
solexact=-0.5*f0/kd*x.^2+(f0*L-qL)/kd*x;
plot(x,solexact)

```

Figure 0.4. 1D program using the finite element method (Example 0.4)

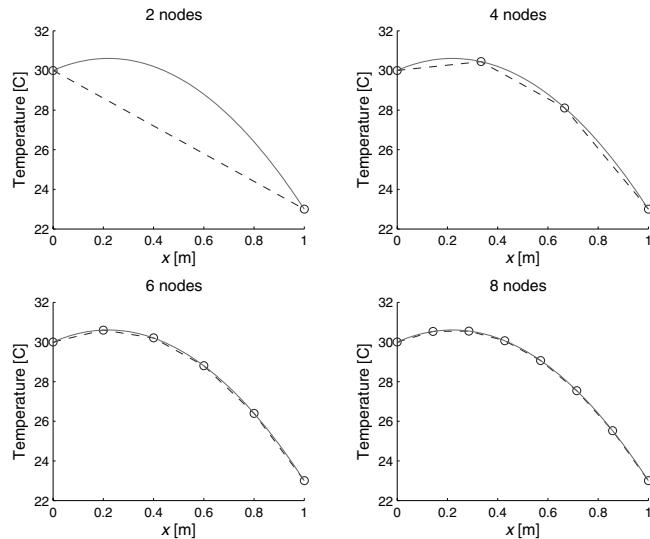


Figure 0.5. Exact solutions and solutions obtained using the finite element method

Figure 0.5 shows the results of the program based on the finite element method for different numbers of nodes. Each illustration shows the exact solution to the problem (continuous curve) and the numerical solution (dotted line).

Bibliography

- [ARA 68] DE ARANTES E OLIVEIRA E.R., “Theoretical foundations of the finite element method”, *International Journal of Solids and Structures*, vol. 4, pp. 929–952, 1968.
- [ARG 60] ARGYRIS J.H., KELSEY S., *Energy Theorems and Structural Analysis*, Butterworth, London, 1960.
- [COU 43] COURANT R., “Variational methods for the solution of problems of equilibrium and vibrations”, *Bulletin of the American Mathematical Society*, vol. 49, pp. 1–23, 1943.
- [ERG 68] ERGATOUDIS J.G., IRONS B.M., ZIENKIEWICZ O.C., “Three-dimensional analysis of Arch Dams and their foundations”, *Symposium on Arch Dams*, Institution of Civil Engineers, London, March 1968.
- [FEL 66] FELIPPRA C.A., Refined finite element analysis of linear and non-linear two-dimensional structures, Report UC SESM 66-22, Department of Civil Engineering, University of California, Berkeley, CA, October 1966.
- [FIN 75] FINLAYSON B.A., “Weighted residual methods and their relation to finite element methods in flow problems”, *Finite Elements in Fluids*, vol. 2, pp. 1–31, 1975.
- [GRE 69] GREENE R.E., JONES R.E., McLAY R.W., STROME D.R., “Generalized variational principles in the finite-element method”, *American Institute of Aeronautics and Astronautics Journal*, vol. 7, no. 7, pp. 1254–1260, 1969.

- [HEN 43] MCHENRY D., "A lattice analogy of the solution of plane stress problems", *Journal of Institution of Civil Engineers*, vol. 21, pp. 59–82, 1943.
- [HOF 56] HOFF N.J., *Analysis of Structures*, Wiley, New York, 1956.
- [HRE 41] HRENNIKOFF A., "Solutions of problems in elasticity by the framework Method", *Journal of Applied Mechanics*, vol. 8, A169–A175, 1942.
- [IRO 68] IRONS B.M., ZIENKIEWICZ O.C., "The isoparametric finite element system – a new concept in finite element analysis", *Proceedings, Conference on Recent Advances in Stress Analysis*, Royal Aeronautical Society, London, 1968.
- [TUR 56] TURNER M.J., CLOUGH R.W., MARTIN H.C., TOPP L.J., "Stiffness and deflection analysis of complex structures", *Journal of Aeronautical Science*, vol. 23, pp. 805–823, 1956.
- [ZIE 65] ZIENKIEWICZ O.C., HOLISTER G.S., *Stress Analysis*, Wiley, New York, 1965.

Conference proceedings

- [PRO 01] *Proceedings of the 1st, 2nd and 3rd Conferences on Matrix Methods in Structural Mechanics*, Wright-Patterson A.F.B., Ohio, 1965, 1968, 1971.
- [PRO 02] HOLLAND I., BELL K. (eds), *Finite Element Methods in Stress Analysis*, Tapir, Trondheim, Norway, 1969.
- [PRO 03] *Proceedings of the 1st, 2nd, 3rd and 4th Conferences on Structural Mechanics in Reactor Technology*, 1971, 1973, 1975, 1977.
- [PRO 04] *Symposium on Applied Finite Element Methods in Civil Engineering*, Vanderbilt University, Nashville, ASCE, 1969.
- [PRO 05] GALLAGHER R.H., YAMADA Y., ODEN J.T. (eds), *Recent Advances in Matrix Methods of Structural Analysis and Design*, University of Alabama Press, Huntsville, 1971.
- [PRO 06] DE VEUBEKE B.F. (ed.), *High Speed Computing of Elastic Structures*, University of Liège, 1971.
- [PRO 07] BREBBIA C.A., TOTTENHAM H. (eds), *Variational Methods in Engineering*, Southampton University, 1973.
- [PRO 08] FENVES S.J., PERRONE N., ROBINSON J., SCHNOBRICH W.C. (eds), *Numerical and Computational Methods in Structural Mechanics*, Academic Press, New York, 1973.
- [PRO 09] GALLAGHER R.H., ODEN J.T., TAYLOR C., ZIENKIEWICZ O.C. (eds), *International Symposium on Finite Element Methods in Flow Problems*, Wiley, 1974.
- [PRO 10] BATHE K.J., ODEN J.T., WUNDERLICH W. (eds), *Formulation and Computational Algorithms in Finite Element Analysis (U.S.: Germany Symposium)*, MIT Press, 1977.
- [PRO 11] GRAY W.G., PINDER G.F., BREBBIA C.A. (eds), *Finite Elements in Water Resources*, Pentech Press, London, 1977.
- [PRO 12] ROBINSON J. (ed.), *Finite Element Methods in Commercial Environments*, Robinson and Associates, Dorset, England, 1978.
- [PRO 13] GLOWINSKI R., RODIN E.Y., ZIENKIEWICZ O.C. (eds), *Energy Methods in Finite Elements Analysis*, Wiley, 1979.
- [PRO 14] AZIZ A.K. (ed.), *The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations*, Academic Press, New York, 1972.
- [PRO 15] WHITEMAN J.R. (ed.), *The Mathematics of Finite Elements and Applications*, Academic Press, London, 1973.

Monographs

- [MON 01] PRZEMIENIECKI J.S., *Theory of Matrix Structural Analysis*, McGraw-Hill, New York, 1968.
- [MON 02] ZIENKIEWICZ O.C., *The Finite Element Method: The Basis (Vol. 1), Solid Mechanics (Vol. 2) & Fluid Mechanics (Vol. 3)*, 5th ed., Butterworth Heinemann, 2000.
- [MON 03] DESAI C.S., ABEL J.F., *Introduction to the Finite Element Method*, Van Nostrand Reinhold, New York, 1972.
- [MON 04] ODEN J.T., *Finite Elements of Non-Linear Continua*, McGraw-Hill, New York, 1972.
- [MON 05] MARTIN H.C., CAREY G.F., *Introduction to Finite Element Analysis*, McGraw-Hill, New York, 1973.
- [MON 06] NORRIE D.J., DE VRIES G., *The Finite Element Method*, Academic Press, New York, 1973.
- [MON 07] ROBINSON J., *Integrated Theory of Finite Element Methods*, Wiley, London, 1973.
- [MON 08] STRAND G., FIX O.J., *Analysis of the Finite Element Methods*, Prentice-Hall, New Jersey, 1973.
- [MON 09] URAL O., *Finite Element Method, Basic Concepts and Applications*, Intext Educational Publishers, 1973.
- [MON 10] COOK R.D., *Concepts and Applications of Finite Element Analysis*, Wiley, 1974.
- [MON 11] GALLAGHER R.H., *Finite Element Analysis Fundamentals*, Prentice-Hall, 1975.
- [MON 12] HUEBNER K.H., *The Finite Element Method for Engineers*, Wiley, 1975.
- [MON 13] WASHIZU K., *Variational Methods in Elasticity and Plasticity*, Pergamon Press, 1976.
- [MON 14] BATHE K.J., WILSON E.L., *Numerical Methods in Finite Element Analysis*, Prentice-Hall, 1976.
- [MON 15] CHEUNG Y.K., *Finite Strip Method in Structural Analysis*, Pergamon Press, 1976.
- [MON 16] CONNOR J.J., BREBBIA C.A., *Finite Element Technique for Fluid Flow*, Butterworth Co., 1976.
- [MON 17] SEGERLIND L.J., *Applied Finite Element Analysis*, Wiley, 1976.
- [MON 18] MITCHELL A.R., WAIT R., *The Finite Element Method in Partial Differential Equations*, Wiley, 1977.
- [MON 19] PINDER G.F., GRAY G.W., *Finite Element Simulation in Surface and Sub-Surface Hydrology*, Academic Press, 1977.
- [MON 20] TONG P., ROSSETOS J., *Finite Element Method: Basic Techniques and Implementation*, MIT Press, 1977.
- [MON 21] CHUNG T.J., *Finite Element Analysis in Fluid Dynamics*, McGraw-Hill, 1978.
- [MON 22] CIARLET P.G., *The Finite Element Method for Elliptic Problems*, North-Holland, 1978.
- [MON 23] IRONS B.M., AHMAD S., *Techniques of Finite Elements*, Ellis Horwood, Chichester, England, 1978.
- [MON 24] DESAI C.S., *Elementary Finite Element Method*, Prentice-Hall, 1979.
- [MON 25] ZIENKIEWICZ O.C., *La Méthode des Elements Finis* (trans.), Pluralis, France, 1976.
- [MON 26] GALLAGHER R.H., *Introduction aux Elements Finis* (trans. J.L. Claudon), Pluralis, France, 1976.
- [MON 27] ROCKEY K.C., EVANS H.R., GRIFFITHS D.W., *Elements Finis* (trans. C. Gomez), Eyrolles, France, 1978.
- [MON 28] ABSI E., *Méthode de calcul numérique en élasticité*, Eyrolles, 1979.
- [MON 29] IMBERT J.F., *Analyse des structures par éléments finis*, CEPADUES Ed. France, 1979.

Periodicals

- [IJ 01] *International Journal for Numerical Methods in Engineering*, (ZIENKIEWICZ O.C., GALLAGHER, R.H., eds), Wiley.
- [IJ 02] *International Journal of Computers and Structures* (LIEBOWITZ H., ed.), Pergamon Press.
- [CM 01] *Computer Methods in Applied Mechanics and Engineering* (ARGYRIS J.H., ed.), North Holland.
- [IJ 03] *International Journal of Computers and Fluids* (TAYLOR C., ed.), Pergamon Press.
- [IJ 04] *International Journal of Numerical Methods in Geotechnics* (DESAI C.S., ed.), Wiley.
- [RE 01] *Revue Européenne des Eléments Finis* (DHATT G., ed.), Hermès-Sciences.

CHAPTER 1

Approximations with finite elements

1.0 Introduction

This chapter is devoted to the study of approximation techniques allowing the replacement of a continuous system by an equivalent discrete system. We first describe **nodal approximations** over a domain V , before introducing the notion of nodal approximation over subdomains, known as **approximation with finite elements**. We present as well the technique of subdividing a domain into elements.

We present the notion of **reference elements** and their **geometrical transformation (mapping)** facilitating the construction of interpolation functions for elements with complicated geometrical shapes.

We shall then describe the general technique to construct interpolation functions for a reference element. The transformation from a reference element into a real element is characterized by the **Jacobian matrix**.

A brief section is devoted to the study of approximation errors. The chapter ends with an application of finite element approximation to estimate the total precipitation over a region from a few discrete measurements of rainfall.

1.1 General remarks

1.1.1 NODAL APPROXIMATION

A mathematical model of a physical system normally involves a number of variables, or functions, $u_{ex}(x)$ representing temperatures, velocities, thicknesses, etc. These are represented by “approximate” functions, $u(x)$, such that the difference:

$$e(x) = u(x) - u_{ex}(x) \quad (1.1)$$

is “small” enough for the desired objective.

To construct an approximation $u(x)$, we can

- choose an expression composed of n functions with n parameters, denoted by a_i :

$$u(x, a_1, a_2, \dots, a_n);$$

- determine parameters a_1, a_2, \dots, a_n to satisfy condition (1.1) according to a plausible criterion, for example making $u_{ex}(x)$ and $u(x)$ coincide at n points x_1, x_2, \dots, x_n , i.e. cancelling $e(x)$ at these n points. The functions $u(x, a_1, a_2, \dots, a_n)$ should be simple and smooth enough for numerical evaluation, derivation and integration within the domain.

The approximation $u(x)$ may be employed, among others, to obtain:

- an approximate representation of a function that is difficult to evaluate, or known only at certain points;
- an approximate solution of a differential or partial differential equation.

These two possibilities are illustrated in the following examples.

EXAMPLE 1.1. Approximation of a physical quantity $u(x)$

Assume that a temperature, $u(x)$, can only be measured at three points:

x	$u_{ex}(x)$
0	20 °C
0.5	25 °C
1	22 °C

We may, however, require an approximate value for u_{ex} at points that are different from the points of measurement. We want our approximation to coincide with the measured values at every point of measurement.

Let us take an approximation in the form of a quadratic polynomial:

$$u_{ex}(x) \approx u(x, a_1, a_2, a_3) = a_1 + a_2 x + a_3 x^2$$

such that

$$\begin{aligned} u_{ex}(x=0) &= u(x=0) = a_1 &= 20 \\ u_{ex}(x=0.5) &= u(x=0.5) = a_1 + 0.5 a_2 + 0.25 a_3 &= 25 \\ u_{ex}(x=1) &= u(x=1) = a_1 + a_2 + a_3 &= 22 \end{aligned}$$

hence

$$a_1 = 20; \quad a_2 = 18; \quad a_3 = -16.$$

$$u_{ex}(x) \approx u(x) = 20 + 18 x - 16 x^2.$$

Thus, for example, at point $x = 0.7$:

$$u(x = 0.7) = 20 + 12.6 - 7.84 = 24.76$$

EXAMPLE 1.2. Approximation solution of a differential equation

We are looking for a function, u_{ex} , that satisfies the following differential equation:

$$\frac{d^2 u_{ex}(x)}{dx^2} = \frac{11 - 12x}{8} \quad \text{where} \quad 0 \leq x \leq 1$$

with boundary conditions: $u_{ex}(x) = 0$ for $x = 0$ and $x = 1$

The exact solution to this problem is:

$$u_{ex}(x) = (-7x + 11x^2 - 4x^3) / 16.$$

Let us choose an approximation of u_{ex} , which satisfies the boundary conditions, of the form:

$$u_{ex}(x) \approx u(x) = a_1 \sin(\pi x) + a_2 \sin(2\pi x)$$

Let us assume that u satisfies the differential equation at points $x_1 = 0.25$ and $x_2 = 0.75$. That is

$$\left. \frac{d^2 u}{dx^2} \right|_{x_1} = -a_1 \pi^2 \sin(0.25\pi) - 4a_2 \pi^2 \sin(0.5\pi) = \frac{11 - 12x_1}{8} = 1$$

and

$$\left. \frac{d^2 u}{dx^2} \right|_{x_2} = -a_1 \pi^2 \sin(0.75\pi) - 4a_2 \pi^2 \sin(1.5\pi) = \frac{11 - 12x_2}{8} = 0.25.$$

Hence

$$a_1 = -\frac{5}{4\sqrt{2}} \frac{1}{\pi^2} \quad a_2 = -\frac{3}{32} \frac{1}{\pi^2}$$

$$u_{ex}(x) \approx u(x) = -\frac{5}{4\sqrt{2}} \frac{1}{\pi^2} \sin(\pi x) - \frac{3}{32} \frac{1}{\pi^2} \sin(2\pi x).$$

The value of $u(x)$ at point $x = 0.25$, for example, is:

$$u(x = 0.25) = -\frac{23}{32} \frac{1}{\pi^2} = -0.0728,$$

and the exact solution is:

$$u_{ex}(x = 0.25) = -0.0703.$$

In this example, the approximation is employed to discretize the differential equation, i.e. replace it with a set of two algebraic equations with unknown a_1 and a_2 .

Functions $u(x)$ and $u_{ex}(x)$, their second derivatives and the error ($u(x) - u_{ex}(x)$) are given in Figure 1.0.

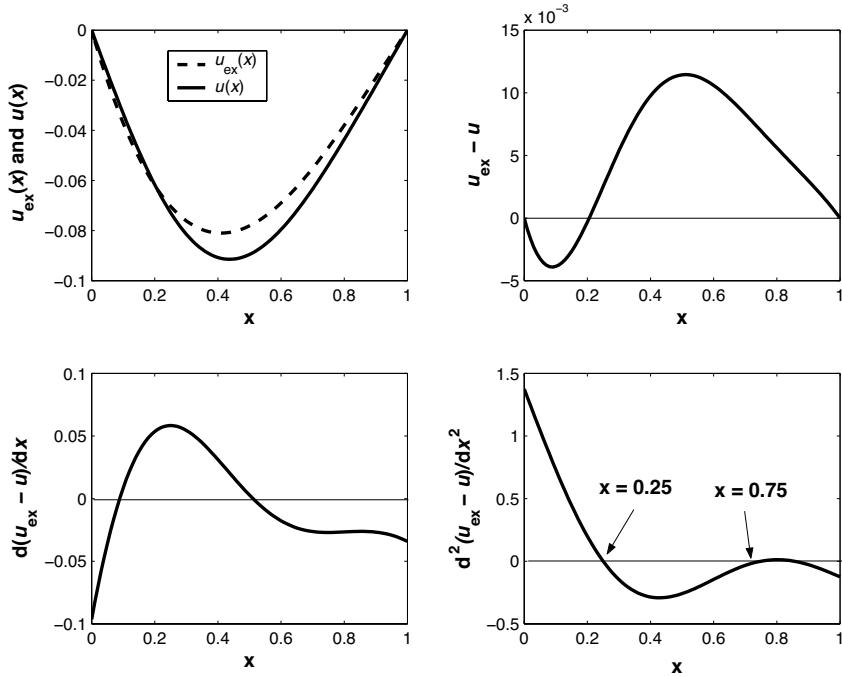


Figure 1.0. Approximation of the solution of a differential equation

As in the previous two examples, the approximate function u is usually a **linear function of a_i** :

$$u(x) = P_1(x) a_1 + P_2(x) a_2 + \cdots + P_n(x) a_n \quad (1.2)$$

thus

$$u(x) = \langle P_1(x) \ P_2(x) \ \dots \ P_n(x) \rangle \begin{Bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{Bmatrix} = \langle P \rangle \{a\} \quad (1.3)$$

where P_1, P_2, \dots, P_n are linearly independent known functions, such as polynomials or trigonometric functions. No function may be constructed by linear combination of the other functions. These functions are independent of the a_i ; and

a_1, a_2, \dots, a_n are the parameters of the approximation.

The a_1, a_2, \dots, a_n parameters usually have no physical meaning. However, we can choose values of the function u at n points (nodes) with coordinates x_1, x_2, \dots, x_n as values for a_i :

$$\begin{aligned} u(x_1) &= u_1 \\ u(x_2) &= u_2 \\ &\dots \\ u(x_n) &= u_n \end{aligned} \quad (1.4)$$

The approximate function (1.2) can now be written as:

$$u(x) = N_1(x) u_1 + N_2(x) u_2 + \cdots + N_n(x) u_n$$

$$u(x) = \langle N_1(x) \ N_2(x) \ \dots \ N_n(x) \rangle \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{Bmatrix} = \langle N \rangle \{u_n\} \quad (1.5)$$

Definitions

- Parameters a_i are the **generalized parameters** of the approximation.
- Parameters u_i are the **nodal parameters** or **nodal variables** of the approximation.
- Equation (1.3) defines a **non-nodal approximation** (see Examples 1.1 and 1.5).
- Equation (1.5) defines a **nodal approximation** (see Example 1.3).
- Functions $P(x)$ are called **basis functions** of the approximation.
- Functions $N(x)$ are **interpolation functions**, also called shape functions.

Nodal approximation possesses the following fundamental properties, which are derived from equations (1.4) and (1.5): as $u(x_i) = u_i$, the functions N_i satisfy:

$$N_j(x_i) = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j. \end{cases} \quad (1.6)$$

The approximation error:

$$e(x) = u(x) - u_{ex}(x). \quad (1.7)$$

vanishes at all nodes, x_i , if we choose $u(x_i) = u_{ex}(x_i)$:

$$e(x_i) = 0. \quad (1.8)$$

EXAMPLE 1.3. Lagrange-type four-noded approximation

Let us consider an arbitrary function, $u_{ex}(x)$, known only at four points, which we approximate by:

$$u(x) = N_1(x) u_1 + N_2(x) u_2 + N_3(x) u_3 + N_4(x) u_4$$

where N_i are Lagrange polynomials of a third degree of the form:

$$N_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^4 \frac{(x - x_j)}{(x_i - x_j)}.$$

These polynomials satisfy equation (1.6).

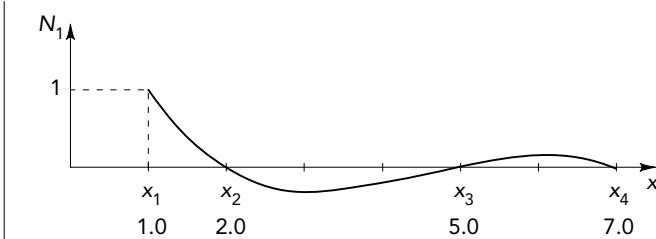
For example, N_1 is written as:

$$N_1(x) = \frac{(x - x_2)(x - x_3)(x - x_4)}{(x_1 - x_2)(x_1 - x_3)(x_1 - x_4)}$$

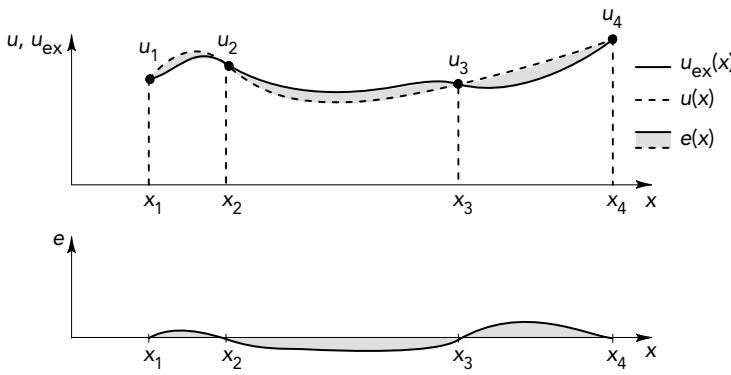
If $x_1 = 1.0$, $x_2 = 2.0$, $x_3 = 5.0$ and $x_4 = 7.0$ the graph of function $N_1(x)$ is written as:

$$N_1 = -\frac{1}{24} (x - 2)(x - 5)(x - 7)$$

x	1	1.5	2	3	4	5	6	7
N_1	1	$\frac{77}{192}$	0	$-\frac{1}{3}$	$-\frac{1}{4}$	0	$\frac{1}{6}$	0



Functions $u_{ex}(x)$, $u(x)$ and the error $e(x)$ are presented schematically as:



The nodal approximation method of a function $u_{ex}(x)$ of one variable can be directly extended to an approximation of a multivariable function, for example in the case of a function with three variables:

$$u_{ex}(x, \gamma, z) = u_{ex}(\mathbf{x})$$

where $\mathbf{x} = \langle x \ \gamma \ z \rangle$ and \mathbf{x} belong to domain V .

The approximate function $u(\mathbf{x})$ is written, in the form used in equation (1.5):

$$u(x, \gamma, z) = u(\mathbf{x}) = \langle N_1(\mathbf{x}) \ N_2(\mathbf{x}) \ \dots \ N_n(\mathbf{x}) \rangle \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{Bmatrix} = \langle N \rangle \{u_n\} \quad (1.9)$$

and must satisfy a relation of type (1.4):

$$u(\mathbf{x}_i) = u_i$$

where $\mathbf{x}_i = \langle x_i \ \gamma_i \ z_i \rangle$ and $i = 1, 2, \dots, n$ are the coordinates of the nodes.

1.1.2 APPROXIMATIONS WITH FINITE ELEMENTS [ODE 71; ZIE 00; STR 73; HUG 87; IMB 84; KIK 86]

The construction of an approximate function $u(\mathbf{x})$ gets very difficult when the number of nodes, n , and therefore of parameters, u_i , increases. More complexity is introduced if the domain V has a complex shape and if function $u(\mathbf{x})$ must satisfy boundary conditions on the boundary of V , as in Example 1.2.

The **method of nodal approximation by subdomains** simplifies the construction of $u(\mathbf{x})$ and is very easily implemented in computer calculations.

This method consists of:

- identifying a set of subdomains, V^e , which together represent the domain V ;
- defining a different approximate function, $u^e(\mathbf{x})$, on each of subdomain V^e using the nodal approximation method. Each function $u^e(\mathbf{x})$ may potentially depend on the nodal variables belonging to other subdomains, as is the case for Spline-type approximations [STR 73].

The **finite element approximation method** is a particular type of approximation over subdomains with the following specific characteristics:

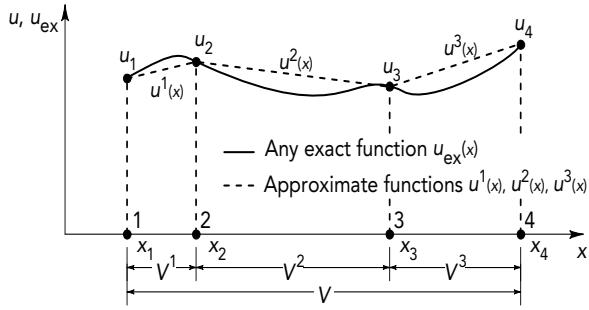
- The nodal approximation over each subdomain V^e depends only on the nodal variables attached to nodes located in V^e and on its boundary.
- The approximate functions $u^e(\mathbf{x})$ over each subdomain V^e are constructed so as to be continuous over V^e and satisfy conditions of continuity between the different subdomains.

Definitions

- The subdomains, V^e , are known as **finite elements**.
- The points at which the approximate function $u^e(\mathbf{x})$ coincides with the function $u(x)$ are called **interpolation nodes** or **nodal points**.
- The coordinates of these nodes, \mathbf{x}_i , are called **nodal coordinates**.
- The values $u_i = u^e(\mathbf{x}_i) = u_{ex}(\mathbf{x}_i)$ are called **nodal variables**.

Finite element approximation presents two distinct steps:

- The geometry of all elements must be defined at the outset, something which may be complicated depending on the shapes involved.
- Appropriate interpolation functions $N_i(\mathbf{x})$ must be constructed for each element while verifying a certain order of continuity between elements.

EXAMPLE 1.4. One-dimensional finite element approximation

Geometrical definition of elements:

Nodes: 1, 2, 3, 4.

Nodal coordinates : x_1, x_2, x_3, x_4 .

Total domain $V : x_1 \leq x \leq x_4$.

Elements $V^1 : x_1 \leq x \leq x_2$

$V^2 : x_2 \leq x \leq x_3$

$V^3 : x_3 \leq x \leq x_4$.

Construction of approximate functions $u^e(x)$:

Nodal variables: u_1, u_2, u_3, u_4 .

Linear approximate functions $u^e(x)$ for each element.

Element 1 (domain V^1):

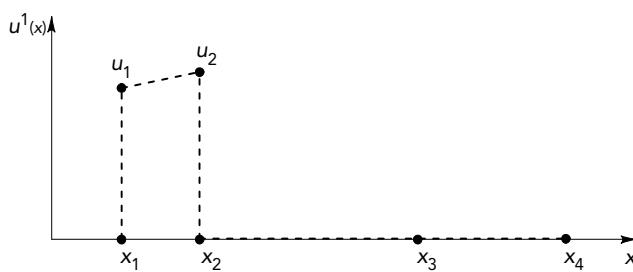
$$u^1(x) = N_1 u_1 + N_2 u_2$$

where N_1 and N_2 are linear functions of x satisfying (1.6):

$$N_1 = \frac{x - x_2}{x_1 - x_2} \quad N_1(x_1) = 1 \quad N_1(x_2) = 0$$

$$N_2 = \frac{x - x_1}{x_2 - x_1} \quad N_2(x_1) = 0 \quad N_2(x_2) = 1$$

$$x_1 \leq x \leq x_2$$



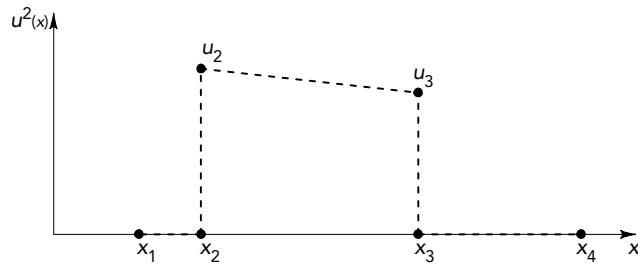
Element 2 (domain V^2):

$$u^2(x) = N_1 u_2 + N_2 u_3$$

where

$$N_1 = \frac{x - x_3}{x_2 - x_3}; \quad N_2 = \frac{x - x_2}{x_3 - x_2}$$

$$x_2 \leq x \leq x_3$$



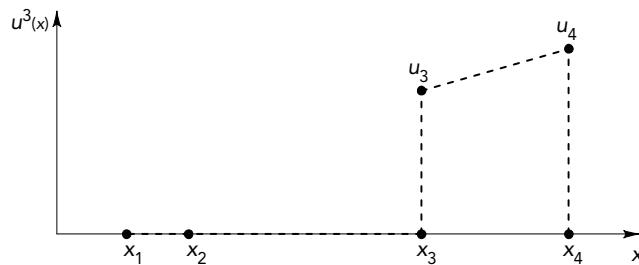
Element 3 (domain V^3):

$$u^3(x) = N_1 u_3 + N_2 u_4$$

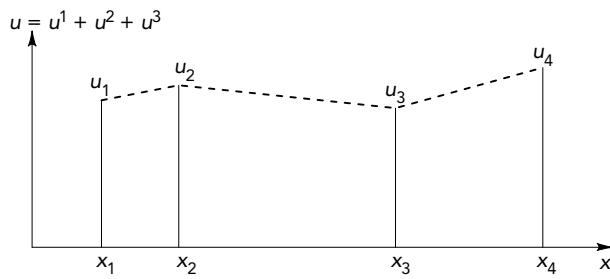
where

$$N_1 = \frac{x - x_4}{x_3 - x_4}; \quad N_2 = \frac{x - x_3}{x_4 - x_3}$$

$$x_3 \leq x \leq x_4.$$

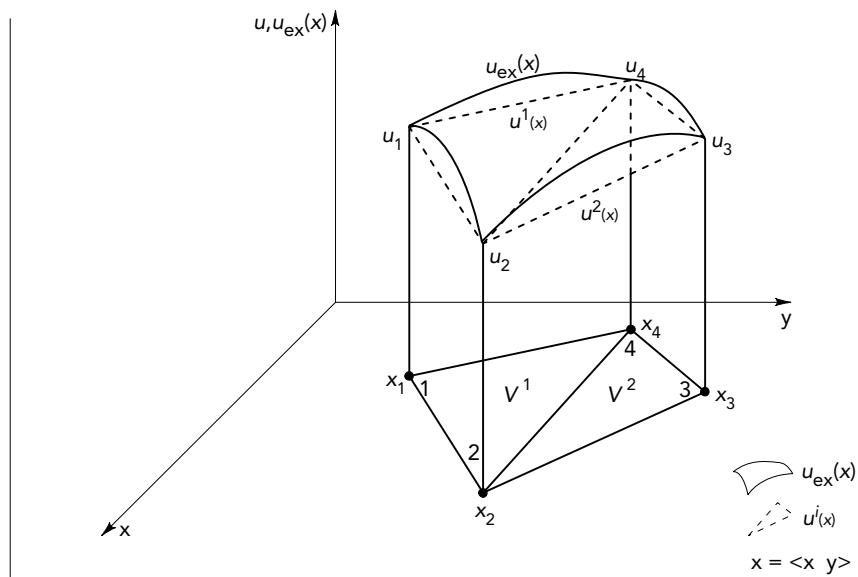


Functions $u^e(x)$ and $N_i(x)$ are different for each element V^e ; these functions are null outside the element V^e . The union of functions $u^1(x), u^2(x)$ and $u^3(x)$ gives the approximate function $u(x)$ over the total domain V :



In Example 1.4 above, it was relatively simple to define the elements and construct approximate functions $u^e(x)$ and interpolation functions $N_i(x)$ for each element. However, in the two-dimensional example that follows, the analytical definition of elements and the construction of these functions are already more complicated.

EXAMPLE 1.5. Two-dimensional linear finite element approximation



Geometrical definition of elements:

Nodes: 1, 2, 3, 4.

Nodal coordinates : $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$.

Total domain, V : quadrilateral 1–2–3–4.

Elements V^1 : triangle 1–2–4

V^2 : triangle 2–3–4.

Construction of approximation functions $u^e(\mathbf{x})$:

Nodal variables: u_1, u_2, u_3, u_4 .

Linear approximation functions $u^e(\mathbf{x})$ over each element.

Element 1 (domain V^1):

$$u^1(\mathbf{x}) = N_1(\mathbf{x}) u_1 + N_2(\mathbf{x}) u_2 + N_3(\mathbf{x}) u_4$$

$u^1(\mathbf{x})$ is a linear function in x and y taking a value equal to u_1 , u_2 , u_4 at points \mathbf{x}_1 , \mathbf{x}_2 , \mathbf{x}_4 . This function is null outside domain V^1 .

Function N_1 is a linear function in x and y taking values of 1 at \mathbf{x}_1 and 0 at \mathbf{x}_2 and \mathbf{x}_4 . N_1 is null outside domain V^1 .

Element 2 (domain V^2):

u^2 is a linear function in x and y taking a value equal to u_2 , u_3 , u_4 at points \mathbf{x}_2 , \mathbf{x}_3 , \mathbf{x}_4 . This function is null outside domain V^2 .

Function N_1 is a linear function taking values of 1 at \mathbf{x}_2 and 0 at \mathbf{x}_3 and \mathbf{x}_4 . N_1 is null outside domain V^2 .

$$u^2(\mathbf{x}) = N_1(\mathbf{x}) u_2 + N_2(\mathbf{x}) u_3 + N_3(\mathbf{x}) u_4$$

Figure 1.1 summarizes the various methods of approximation mentioned so far.

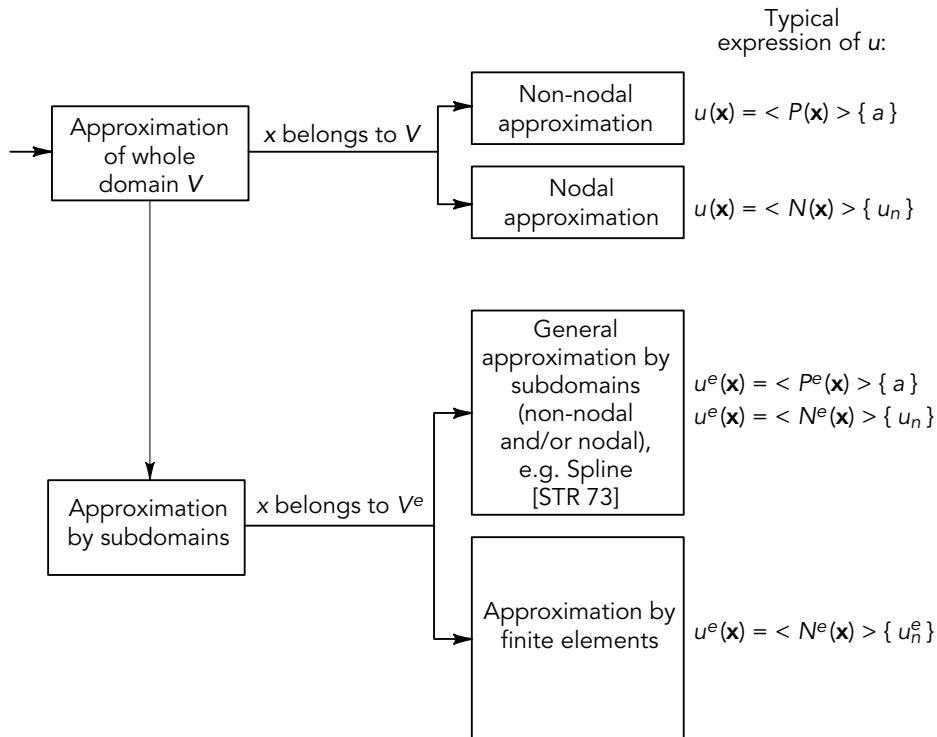


Figure 1.1. Approximation methods

The following two sections describe systematic methods for the analytical definition of elements with complex shapes and for the construction of the corresponding approximate functions.

1.2 Geometrical definition of the elements

1.2.1 GEOMETRICAL NODES

A set of points, \bar{n} , is selected in the domain V to define the geometry of the elements. These points, called **geometrical nodes**, may sometimes coincide with the interpolation nodes. We will then replace the domain V by a set of elements, V^e , of relatively simple shapes. Each element V^e must be analytically and uniquely defined in terms of the coordinates of the geometrical nodes belonging to that element, which are situated in V^e and on its boundary.

EXAMPLE 1.6. One-dimensional domain

In Example 1.4, nodes 1, 2, 3 and 4 are geometrical nodes chosen in domain V . Each element V^e is defined from the coordinates of the two geometrical nodes located at its extremities. For example, element 2 is defined by:

$$x_2 \leq x \leq x_3.$$

EXAMPLE 1.7. Two-dimensional triangular domain

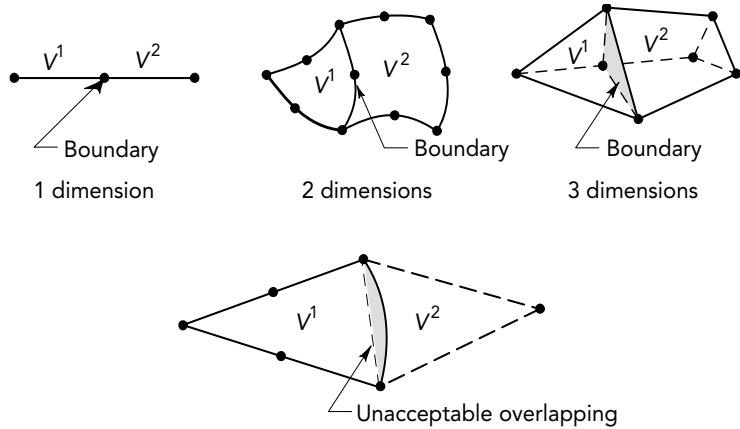
In Example 1.5, nodes 1, 2, 3 and 4 are geometrical nodes. Each element is defined from the coordinates of the three geometrical nodes situated at its vertices. For example, element 1 is defined by the condition that \mathbf{x} belongs to the triangle whose vertices are $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_4$.

The analytical expression translating this condition is complex; we will give a simple explicit form in Example 1.9 using a geometrical transformation.

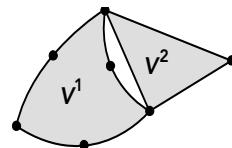
1.2.2 RULES FOR THE PARTITION OF A DOMAIN INTO ELEMENTS

The division of a domain V into elements V^e should satisfy the following two rules:

- a) Two distinct elements can have common points only on their common boundaries, if such boundaries exist. This condition excludes overlapping between elements. Common boundaries can be points, lines or surfaces.

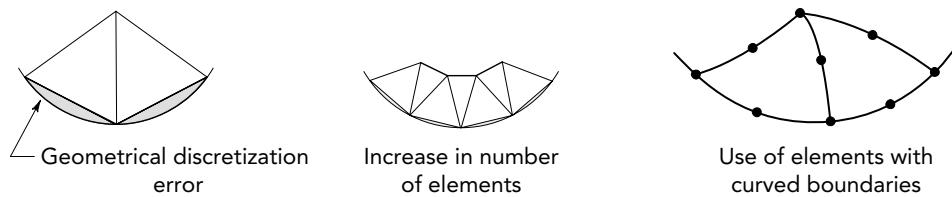


- b) The set of elements V^e must constitute a domain that is as close as possible to the given domain V . There should be no “holes” between elements:



Unacceptable hole between elements

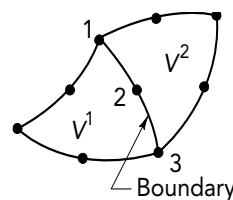
When the boundary of a domain V is constituted by lines or surfaces that are more complex than those that define the boundaries between elements, a certain level of error cannot be avoided. This is known as the **geometrical discretization error**. It can be decreased by reducing the size of the elements, or by using elements with curved boundaries:



The two previous rules are satisfied if the elements are constructed as follows:

- Each element is uniquely defined by the coordinates of the geometrical nodes situated in that element. These geometrical nodes are usually situated on the boundaries of the element and could be common to other adjacent elements.
- The boundary of a two- or three-dimensional element is composed of a set of curves or surfaces. Each portion of the boundary must be uniquely defined from the coordinates of the only geometrical nodes located on this part of the boundary. Thus, the portions of boundary common to two different elements are defined identically for both elements.

EXAMPLE 1.8. Boundary between two elements



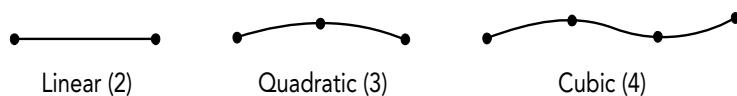
Boundary 1-2-3 must be uniquely defined from the coordinates of nodes 1, 2 and 3. It is possible to choose the parabola passing through these three nodes.

1.2.3 SHAPES OF SOME CLASSICAL ELEMENTS

We now show the shapes of some classical one-, two- or three-dimensional elements.

Each element is given a name suggestive of its shape and the type of curve or surface bounding that element. Furthermore, we state the number of geometrical nodes necessary to define the element.

a) One-dimensional elements



b) Two-dimensional elements

These are triangles or quadrilaterals, the sides of which are polynomial curves of the first, second or third degree.

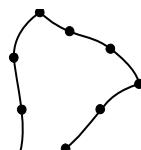
Triangular elements:



Linear (3)

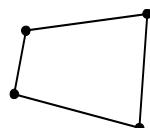


Quadratic (6)

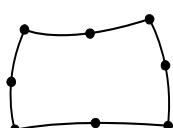


Cubic (9)

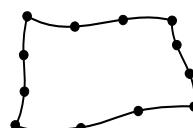
Quadrilateral elements:



Linear (4)



Quadratic (8)

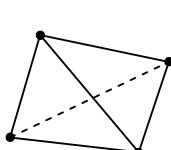


Cubic (12)

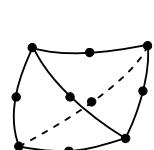
c) Three-dimensional elements

These are tetrahedrons, hexahedrons or prisms, the faces of which are polynomial surfaces of the first, second or third degree.

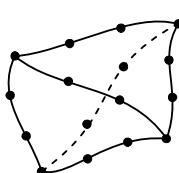
Tetrahedral elements:



Linear (4)

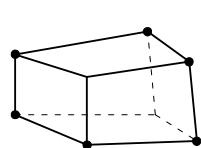


Quadratic (10)

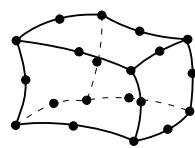


Cubic (16)

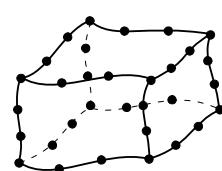
Hexahedral elements:



Linear (8)

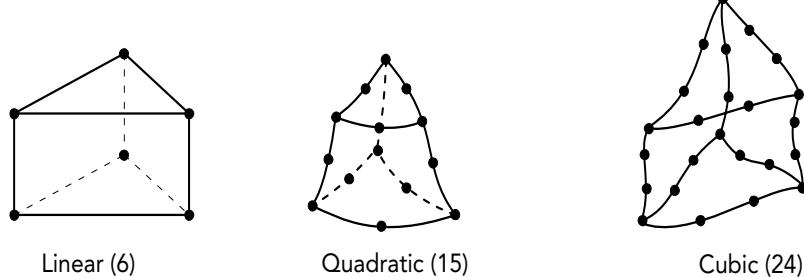


Quadratic (20)



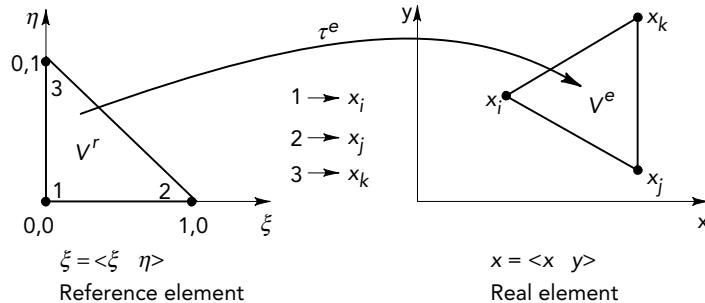
Cubic (32)

Prismatic elements:



1.2.4 REFERENCE ELEMENTS

To simplify the analytical definition for elements of complex shapes, an element of reference is introduced. Such a **reference element** V^r is an element of a very simple shape, identified in a **reference space**, which may be transformed into any real element V^e by a geometrical transformation τ^e . For example, in the case of a triangle:



Transformation τ^e defines the coordinates \mathbf{x}^e of each point of the real element in terms of the coordinates ξ of the corresponding point in the reference element.

$$\tau^e: \xi \rightarrow \mathbf{x}^e = \mathbf{x}^e(\xi) \quad (1.10)$$

Transformation τ^e depends on the form and location of the real element, and thus the coordinates of the geometrical nodes that define it. There is therefore a different transformation τ^e for each real element:

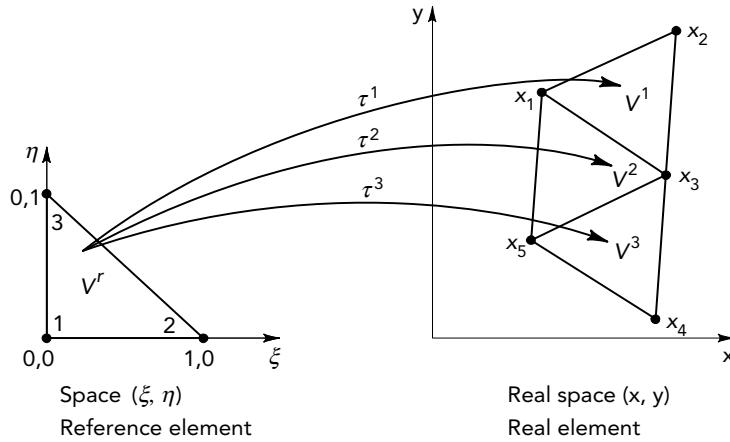
$$\tau^e: \xi \rightarrow \mathbf{x}^e = \mathbf{x}^e(\xi, \mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k, \dots) \quad (1.11)$$

where $\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k, \dots$ are the coordinates of the geometrical nodes belonging to element e .

These transformations τ^e must satisfy the rules established in section 1.2.2. To do this, each transformation τ^e is chosen so as to have the following properties:

- It is bijective at every point ξ located on the reference element or its boundary; each point V^r thus corresponds to a single point V^e and vice versa.
- The geometrical nodes of the reference element correspond to the geometrical nodes of the real element.
- Every portion of the boundary of the reference element, defined by the geometrical nodes of that boundary, corresponds to the portion of the boundary of the real element defined by the corresponding nodes.

We can think that a single reference element V^r (e.g. a three-node triangle) maps into all the real elements V^e of the same type (three-node triangle) using different transformations τ^e :



$$\text{Element 1 } \tau^1: \xi \rightarrow x^1 = x^1(\xi, x_1, x_3, x_2)$$

$$\text{Element 2 } \tau^2: \xi \rightarrow x^2 = x^2(\xi, x_1, x_5, x_3)$$

$$\text{Element 3 } \tau^3: \xi \rightarrow x^3 = x^3(\xi, x_5, x_4, x_3)$$

From now on, to simplify the notation used, the superscript e , used to characterize an element, will be removed. We will use a transformation τ that is linear in relation to the coordinates $\{\mathbf{x}_n\}$ of the geometrical nodes of the real element V^e :

$$\tau: \xi \rightarrow \mathbf{x}(\xi) = [\bar{N}(\xi)] \{ \mathbf{x}_n \}. \quad (1.12)$$

Furthermore, identical transformation functions will be used for all three coordinates:

$$\begin{aligned}x(\xi) &= \langle \bar{N}(\xi) \rangle \{x_n\} \\y(\xi) &= \langle \bar{N}(\xi) \rangle \{y_n\} \\z(\xi) &= \langle \bar{N}(\xi) \rangle \{z_n\}.\end{aligned}$$

For example, in the case of a three-nodes triangle $\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k$:

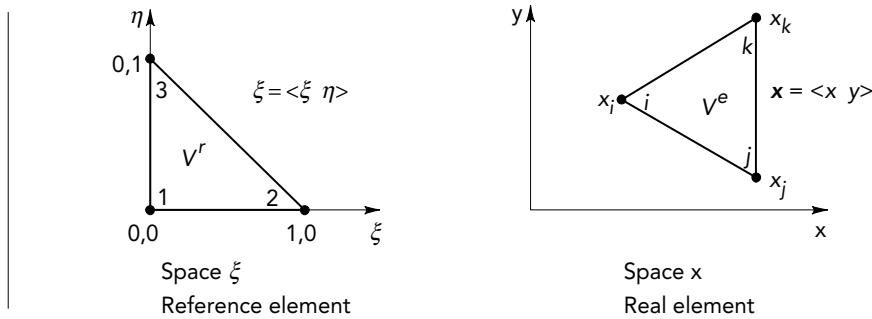
$$\begin{aligned}x(\xi, \eta) &= \bar{N}_1(\xi, \eta) x_i + \bar{N}_2(\xi, \eta) x_j + \bar{N}_3(\xi, \eta) x_k = \langle \bar{N} \rangle \begin{Bmatrix} x_i \\ x_j \\ x_k \end{Bmatrix} \\y(\xi, \eta) &= \bar{N}_1(\xi, \eta) y_i + \bar{N}_2(\xi, \eta) y_j + \bar{N}_3(\xi, \eta) y_k = \langle \bar{N} \rangle \begin{Bmatrix} y_i \\ y_j \\ y_k \end{Bmatrix}\end{aligned}$$

where (ξ, η) belongs to V^r .

The functions \bar{N}_i , normally chosen as polynomials in ξ , are called **geometrical transformation functions**. We may consider (1.12) as a nodal approximation by subdomains for functions $x(\xi)$ and $y(\xi)$. Functions must be such that transformation (1.12) satisfies the three properties mentioned in section 1.2.4. They can be constructed in the same way as the interpolation functions $N(\xi)$, which will be described in sections 1.3 and 1.4.

With the geometrical transformation τ , it is now possible to replace the analytical definition of each element V^e in the space of \mathbf{x} by an easier analytical definition of the reference element V^r in the space of ξ . From now on, we will work exclusively in the domain of ξ and use functions denoted $u(\xi)$ instead of $u(\mathbf{x})$. The relationship between ξ and \mathbf{x} is defined by equation (1.12). Functions $u(\xi)$ and $u(\mathbf{x})$ are different but take the same value at corresponding points in the transformation. We thus obtain $u(\mathbf{x}) = u(\mathbf{x}(\xi))$, which, for reasons of simplicity, we will denote $u(\mathbf{x}) = u(\xi)$.

EXAMPLE 1.9. Analytical definition of a three-noded triangular element



The reference element is defined analytically as:

$$\xi + \eta \leq 1$$

$$\xi \geq 0$$

$$\eta \geq 0.$$

Let us consider the linear transformation in (ξ, η) , such that:

$$x(\xi, \eta) = \langle 1 - \xi - \eta \quad \xi \quad \eta \rangle \begin{Bmatrix} x_i \\ x_j \\ x_k \end{Bmatrix}$$

$$y(\xi, \eta) = \langle 1 - \xi - \eta \quad \xi \quad \eta \rangle \begin{Bmatrix} y_i \\ y_j \\ y_k \end{Bmatrix}$$

This satisfies the three following properties:

- The vertex nodes of V^r with coordinates $\langle 0 \ 0 \rangle$, $\langle 1 \ 0 \rangle$ and $\langle 0 \ 1 \rangle$, transform into the vertex nodes of V^e with coordinates \mathbf{x}_i , \mathbf{x}_j and \mathbf{x}_k . For example:

$$x(\xi = 0, \eta = 0) = \langle 1 \ 0 \ 0 \rangle \begin{Bmatrix} x_i \\ x_j \\ x_k \end{Bmatrix} = x_i.$$

- Each boundary of V^r transforms into the corresponding boundary of V^e . For example, the boundary passing through nodes $\langle 1 \ 0 \rangle$ and $\langle 0 \ 1 \rangle$, the equation for which is $1 - \xi - \eta = 0$, transforms into the boundary of V^e passing through \mathbf{x}_j and \mathbf{x}_k , for which the parametric equation is:

$$x = \langle 0 \quad \xi \quad 1 - \xi \rangle \begin{Bmatrix} x_i \\ x_j \\ x_k \end{Bmatrix} = \xi x_j + (1 - \xi) x_k$$

$$y = \langle 0 \quad \xi \quad 1 - \xi \rangle \begin{Bmatrix} y_i \\ y_j \\ y_k \end{Bmatrix} = \xi y_j + (1 - \xi) y_k.$$

Note that this equation is linear in ξ and η and depends only on the coordinates \mathbf{x}_j and \mathbf{x}_k of the nodes situated on that boundary.

— Transformation τ is bijective if the matrix $[J]$ is non-singular:

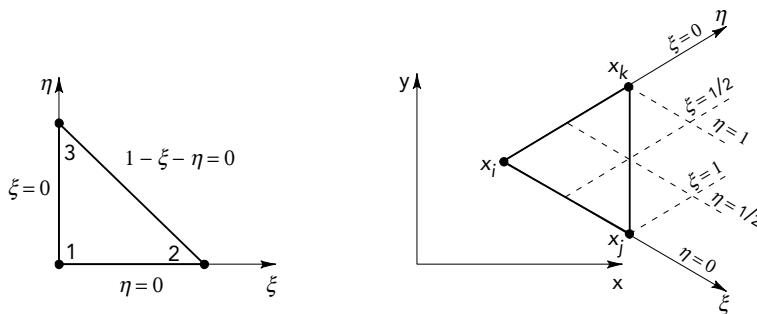
$$[J] = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} x_j - x_i & y_j - y_i \\ x_k - x_i & y_k - y_i \end{bmatrix}$$

$$\det(J) = (x_j - x_i)(y_k - y_i) - (x_k - x_i)(y_j - y_i)$$

This determinant is equal to twice the area of the triangle, so it can vanish only if the three vertices of the element are aligned. More generally, the interior angles of all the triangles and quadrilaterals must be less than 180° so that $\det(J)$ does not vanish [STR 73; HUG 87].

Remarks

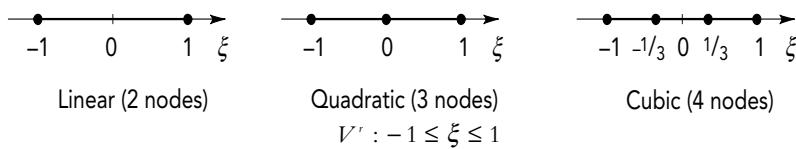
- Reference elements are sometimes called **parent elements**.
- The geometrical transformation τ can be viewed as a simple **change of variables** $x \rightarrow \xi$.
- ξ may also be viewed as a **system of local coordinates** attached to each element.



1.2.5 SHAPES OF SOME CLASSICAL REFERENCE ELEMENTS

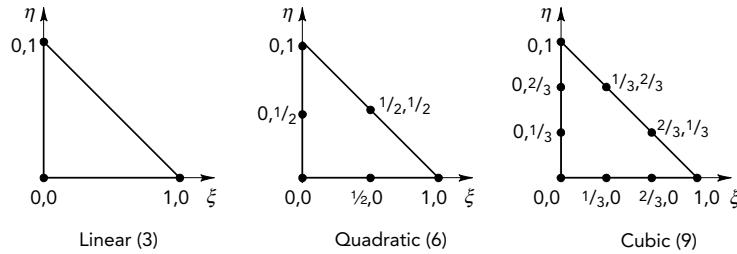
We now illustrate the shape and analytical definition of the reference elements corresponding to the classical elements covered in section 1.2.3.

a) One-dimensional reference elements



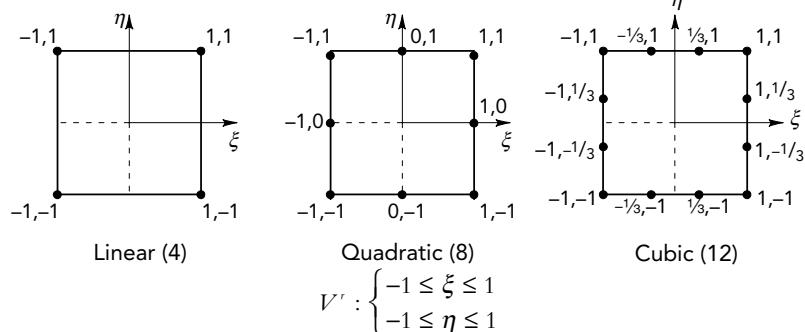
b) Two-dimensional reference elements

Triangular elements:



$$V^r : \begin{cases} \xi + \eta \leq 1 \\ \xi \geq 0 \\ \eta \geq 0 \end{cases}$$

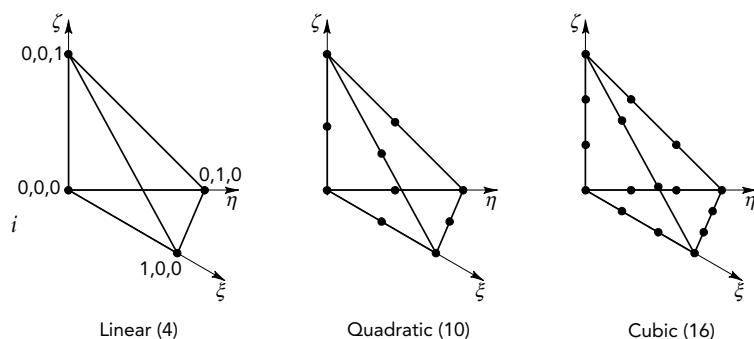
Square elements:



$$V^r : \begin{cases} -1 \leq \xi \leq 1 \\ -1 \leq \eta \leq 1 \end{cases}$$

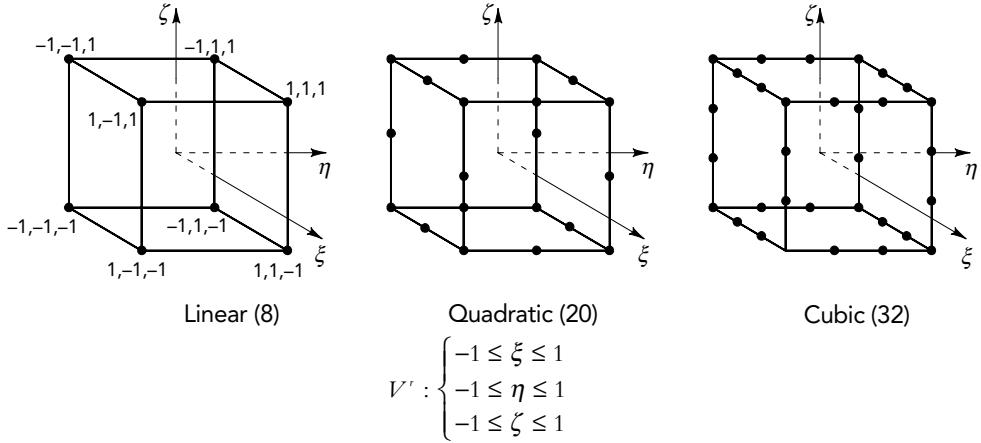
c) Three-dimensional reference elements

Tetrahedral elements:

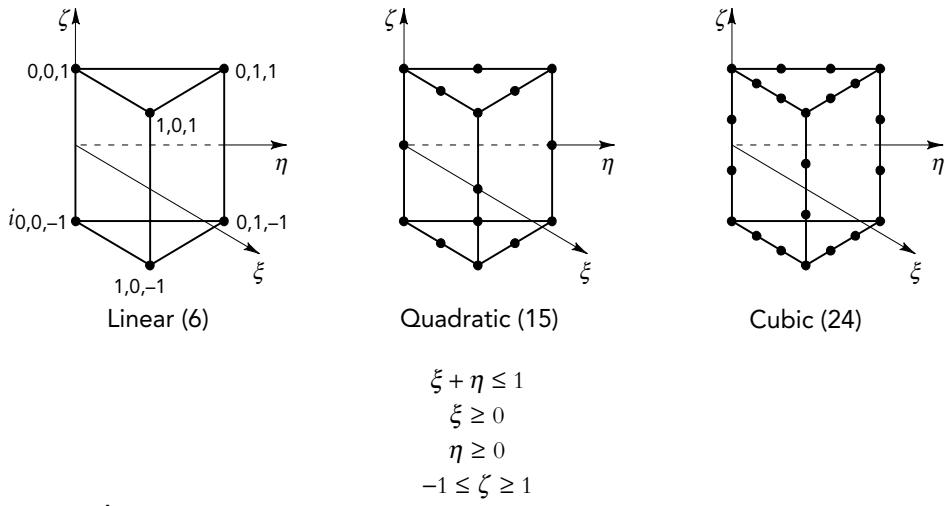


$$V^r : \begin{cases} \xi + \eta + \zeta \leq 1 \\ \xi \geq 0 \\ \eta \geq 0 \\ \zeta \geq 0 \end{cases}$$

Cubic elements:



Prismatic elements:



Remarks

- In quadratic reference elements, the nodes situated on the edges of the element are located in the middle of these edges. In cubic elements, the nodes are situated one-third and two-thirds of the way along the edges.
- The geometrical transformation functions $\bar{N}(\xi)$ have not been mentioned explicitly for the reference elements above. The construction of these functions is identical to that for interpolation functions, $N(\xi)$, which will be discussed in section 1.4.

In the following section, we present the standard organization of geometrical data tables, in a form that is well suited to programming.

1.2.6 NODE AND ELEMENT DEFINITION TABLES

Let us number the geometrical nodes sequentially from 1 to \bar{n} , then define the coordinates of each node using a single global coordinate system adapted to the problem. The coordinates will then be stored in a CORG (*COoRdonnées Globales*) global coordinates table. For a two-dimensional problem, this table takes the following form:

	Nodes					
	1	2	3	...	\bar{n}	
x	x_1	x_2	x_3	...	$x_{\bar{n}}$	
y	y_1	y_2	y_3	...	$y_{\bar{n}}$	

CORG table

Let us number the elements sequentially from 1 to n_{el} , then define each element using the list of numbers of its geometrical nodes. This list is stored in a CONEC (*CONnECtivité*) connectivity table.

	Elements						
	1	2	...	e	...	n_{el}	
1				i_1			
2				i_2			
3			...	i_3	...		
\vdots				\vdots			
\bar{n}^e				$i_{\bar{n}^e}$			

CONEC table

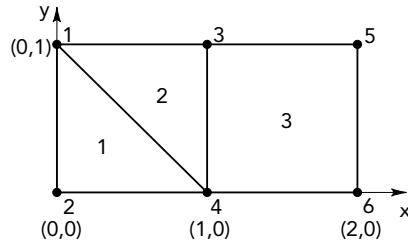
where \bar{n}^e is the maximum number of geometrical nodes per element; $i_1, i_2, i_3, \dots, i_{\bar{n}^e}$ are the numbers of the nodes of element e .

Together, the CORG and CONEC tables are sufficient to completely define the transformation, τ , for all the elements, i.e. to construct the functions $\bar{N}(\xi)$ and the vector of nodal coordinates for each element:

$$\{\mathbf{x}_1\} = \left\{ \begin{array}{c} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{\bar{n}^e} \end{array} \right\}$$

EXAMPLE 1.10. CORG and CONEC tables for a two-dimensional problem

Consider a domain represented by two triangular elements and one square element:



The nodes are numbered 1–6.

The elements are numbered 1–3.

Nodes						
	1	2	3	4	5	6
CORG table: x	0	0	1	1	2	2
	1	0	1	0	1	0

Elements						
	1	2	3			
CONEC table: Nodes	1	1	3	3		
	2	2	1	4		
	3	4	4	6		
	4	—	—	5		

Remark

It is important to choose a numbering convention in order to establish the list of nodes for an element, for example the trigonometric sense, as this has an influence on the sign of the determinant of the Jacobian geometrical transformation matrix.

1.3 Approximation based on a reference element**1.3.1 EXPRESSION OF THE APPROXIMATE FUNCTION $u(x)$**

Let us choose a set of n **interpolation nodes** of coordinates \mathbf{x}_i in the domain V , which may or may not coincide with the geometrical nodes. For each

element V^e let us use a nodal approximation of type (1.5) for the exact function $u_{\text{ex}}(\mathbf{x})$:

$$u_{\text{ex}}(\mathbf{x}) \approx u(\mathbf{x}) = \langle N_1(\mathbf{x}) \ N_2(\mathbf{x}) \ \dots \ N_{n^e}(\mathbf{x}) \rangle \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n^e} \end{Bmatrix} = \langle N(\mathbf{x}) \rangle \{u_n\} \quad (1.13)$$

where \mathbf{x} belongs to V^e ;

u_1, u_2, \dots, u_{n^e} are the values of u_{ex} at the interpolation nodes n^e of the element, i.e. nodal variables; and $N(\mathbf{x})$ are the **interpolation functions on the real element**.

Now replace the approximation on the real element by the corresponding approximation on the reference element:

$$u_{\text{ex}}(\xi) \approx u(\xi) = \langle N(\xi) \rangle \{u_n\} \quad (1.14)$$

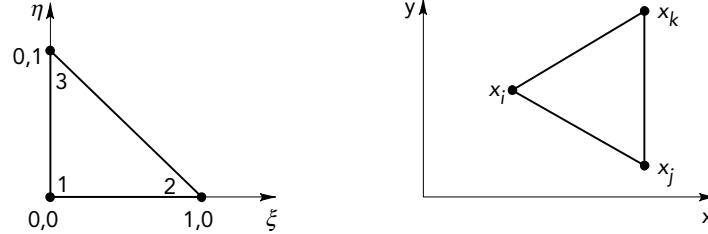
with (1.12):

$$\tau: \xi \rightarrow \mathbf{x}(\xi) = [\bar{N}(\xi)] \{\mathbf{x}_n\}$$

where $\{u_n\}$ are the nodal variables of the element, $\langle N(\xi) \rangle$ are the interpolation functions on the reference element.

Remarks

- In general, functions $N(\mathbf{x})$ are used only for the most simple elements. They are usually replaced by $N(\xi)$ functions where \mathbf{x} and ξ are related by the transformation τ defined in (1.12).
- In expression (1.13), functions $N(\mathbf{x})$ depend on the nodal coordinates of the element and are different for each real element. On the other hand, in equation (1.14) functions $N(\xi)$ are independent of the geometry of the real element V^e . The same functions $N(\xi)$ may therefore be used for all elements having the same reference element, characterized by:
 - its shape
 - its geometrical nodes
 - its interpolation nodes.

EXAMPLE 1.11. *Interpolation functions for a three-noded triangle*

In this case, the three nodes are both interpolation and geometrical nodes. The nodal variables are:

$$\{u_n\} = \begin{Bmatrix} u_i \\ u_j \\ u_k \end{Bmatrix}$$

The linear interpolation on the real element, which takes the form (1.5), composed of three monomials ($1, x$ and y), is written as:

$$u(x, y) = \langle N_1(x, y) \quad N_2(x, y) \quad N_3(x, y) \rangle \begin{Bmatrix} u_i \\ u_j \\ u_k \end{Bmatrix}$$

where

$$N_1(x, y) = \frac{1}{2A} [(y_k - y_j)(x_j - x) - (x_k - x_j)(y_j - y)]$$

$$N_2(x, y) = \frac{1}{2A} [(y_i - y_k)(x_k - x) - (x_i - x_k)(y_k - y)]$$

$$N_3(x, y) = \frac{1}{2A} [(y_j - y_i)(x_i - x) - (x_j - x_i)(y_i - y)]$$

$$2A = (x_k - x_j)(y_i - y_j) - (x_i - x_j)(y_k - y_j).$$

Note that functions $N_i(x, y)$ depend on the coordinates of the nodes.

The interpolation on the reference element, is:

$$u(\xi, \eta) = \langle N_1(\xi, \eta) \quad N_2(\xi, \eta) \quad N_3(\xi, \eta) \rangle \begin{Bmatrix} u_i \\ u_j \\ u_k \end{Bmatrix}$$

$$N_1(\xi, \eta) = 1 - \xi - \eta$$

$$N_2(\xi, \eta) = \xi$$

$$N_3(\xi, \eta) = \eta.$$

The value $u(\xi, \eta)$ obtained by interpolation on the reference element is identical to the value $u(x, y)$ obtained by interpolation on the real element, provided points (ξ, η) and (x, y) are related by transformation τ :

$$\tau: \begin{cases} x(\xi, \eta) = \langle \bar{N}_1 \quad \bar{N}_2 \quad \bar{N}_3 \rangle \begin{Bmatrix} x_i \\ x_j \\ x_k \end{Bmatrix} \\ y(\xi, \eta) = \langle \bar{N}_1 \quad \bar{N}_2 \quad \bar{N}_3 \rangle \begin{Bmatrix} y_i \\ y_j \\ y_k \end{Bmatrix} \end{cases}$$

where $\bar{N}_1 \equiv N_1 = 1 - \xi - \eta$, $\bar{N}_2 \equiv N_2 = \xi$, $\bar{N}_3 \equiv N_3 = \eta$.

Let us verify that $u(\xi_0, \eta_0) = u(x_0, y_0)$ if (ξ_0, η_0) corresponds to (x_0, y_0) in transformation τ . Let us take, for example:

$$\xi_0 = \frac{1}{4}, \quad \eta_0 = \frac{1}{2}$$

so that

$$u(\xi_0, \eta_0) = \left\langle \frac{1}{4} \quad \frac{1}{4} \quad \frac{1}{2} \right\rangle \begin{Bmatrix} u_i \\ u_j \\ u_k \end{Bmatrix} = \frac{1}{4} (u_i + u_j + 2 u_k)$$

and

$$x_0(\xi_0, \eta_0) = \left\langle \frac{1}{4} \quad \frac{1}{4} \quad \frac{1}{2} \right\rangle \begin{Bmatrix} x_i \\ x_j \\ x_k \end{Bmatrix} = \frac{1}{4} (x_i + x_j + 2 x_k)$$

$$y_0(\xi_0, \eta_0) = \frac{1}{4} (y_i + y_j + 2 y_k)$$

Thus substituting into the expressions of $N_1(x, y)$, $N_2(x, y)$ and $N_3(x, y)$:

$$\begin{aligned} N_1(x_0, y_0) &= \frac{1}{4} \\ N_2(x_0, y_0) &= \frac{1}{4} \\ N_3(x_0, y_0) &= \frac{1}{2}. \\ \text{Therefore } u(x_0, y_0) &= \left\langle \frac{1}{4} \quad \frac{1}{4} \quad \frac{1}{2} \right\rangle \begin{Bmatrix} u_i \\ u_j \\ u_k \end{Bmatrix} = u(\xi_0, \eta_0). \end{aligned}$$

1.3.2 PROPERTIES OF APPROXIMATE FUNCTION $u(x)$

a) Interpolating property of the nodal approximation

The approximate function $u(x)$ takes the value u_i at nodes with coordinates \mathbf{x}_i :

$$u(\mathbf{x}_i) = u_i = \langle N_1(\mathbf{x}_i) \quad N_2(\mathbf{x}_i) \quad \dots \quad N_{n^e}(\mathbf{x}_i) \rangle \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n^e} \end{Bmatrix}.$$

$$\text{Hence } N_j(\mathbf{x}_i) = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j. \end{cases} \quad (1.15)$$

Similarly, using the approximation on the reference element:

$$u(\xi_i) = u_i = \langle N_1(\xi_i) \quad N_2(\xi_i) \quad \dots \quad N_{n^e}(\xi_i) \rangle \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n^e} \end{Bmatrix}.$$

$$\text{Hence } N_j(\xi_i) = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j. \end{cases} \quad (1.16)$$

EXAMPLE 1.12. Fundamental property of the interpolation functions of a three-noded triangle

In Example 1.1, we verify, for example, that for

$$\mathbf{x} = \mathbf{x}_k = \langle x_k \ y_k \rangle$$

$$N_1(\mathbf{x} = \mathbf{x}_k) = 0; \ N_2(\mathbf{x} = \mathbf{x}_k) = 0; \ N_3(\mathbf{x} = \mathbf{x}_k) = 1$$

and for $\xi = \xi_3 = \langle 0 \ 1 \rangle$

$$N_1(\xi = \xi_3) = 0; \ N_2(\xi = \xi_3) = 0; \ N_3(\xi = \xi_3) = 1.$$

b) Continuity over the element

If the approximate function $u(\mathbf{x})$ is required to be continuous over the element together with all its derivatives up to the order s , functions $N_i(\mathbf{x})$ with continuous derivatives up to the order s , must be used.

c) Inter-element continuity

If the approximate function $u(\mathbf{x})$ and its derivatives up to the order s are required to be continuous on a common boundary with another element, then $u(\mathbf{x})$ and its derivatives up to the order s must depend solely on the nodal variables associated with the nodes on this boundary.

Let us consider the continuity of $u(\mathbf{x})$ over a boundary (continuity C^0):

$$u(\mathbf{x}) = \langle N_1(\mathbf{x}) \ N_2(\mathbf{x}) \ \dots \ N_{n^e}(\mathbf{x}) \rangle \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n^e} \end{Bmatrix}.$$

Products $N_i(\mathbf{x}) u_i$ must be null if u_i is not a nodal variable associated with a node on this boundary.

Hence: $N_i(\mathbf{x}) = 0$ (1.17a)

when $\left| \begin{array}{l} \mathbf{x} \text{ is situated on a boundary, and} \\ u_i \text{ is not a nodal variable of this boundary.} \end{array} \right.$

Similarly, for the reference element:

$$N_i(\xi) = 0$$

when ξ is situated on the boundary, and
 u_i is not a nodal variable of this boundary.

The condition for $\frac{\partial u(\mathbf{x})}{\partial x}$ to be continuous across a boundary is written in a similar manner:

$$\frac{\partial u(\mathbf{x})}{\partial x} = \left\langle \frac{\partial N_1(\mathbf{x})}{\partial x} \quad \frac{\partial N_2(\mathbf{x})}{\partial x} \dots \frac{\partial N_{n^e}(\mathbf{x})}{\partial x} \right\rangle \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n^e} \end{Bmatrix}$$

where

$$\frac{\partial N_i(\mathbf{x})}{\partial x} = 0 \quad (1.17b)$$

when \mathbf{x} is situated on a boundary, and
 u_i is not a nodal variable of this boundary.

The previous condition may be written for the reference element, in two dimensions:

$$\frac{\partial N_i(\xi)}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial N_i(\xi)}{\partial \eta} \frac{\partial \eta}{\partial x} = 0$$

when ξ is situated on the boundary, and
 u_i is not a nodal variable of this boundary.

The notion of inter-element continuity has played a major role in the development of the finite element method. It is linked to the notion of a compatible or non-compatible element. The type of continuity required is problem dependent and will be discussed in Chapters 3 and 4.

EXAMPLE 1.13. Continuity on the boundaries of a three-noded triangle

Consider the edge $\mathbf{x}_j - \mathbf{x}_k$ of element of Example 1.11:

$$x - x_j = (\gamma - \gamma_j) \frac{(x_k - x_j)}{(y_k - y_j)}.$$

We verify that this expression nullifies function $N_1(x, \gamma)$ corresponding to node i .

The corresponding side $(1, 0) - (0, 1)$ of the reference element has the equation:

$$1 - \xi - \eta = 0.$$

This relation nullifies $N_1(\xi, \eta)$. Consequently, the linear function $u^e(x)$ is continuous on edge $\mathbf{x}_j - \mathbf{x}_k$ since, for this side, it depends only on u_j and u_k and not on u_i :

$$u = \xi u_j + (1 - \xi) u_k.$$

d) Complete polynomial interpolation functions

The approximation error (1.7) can be reduced by increasing the number of elements thus reducing the size of these elements. Depending on the problem being studied, we may wish to reduce the error $u - u_{\text{ex}}$ and possibly also the errors on derivatives.

To ensure that error $u - u_{\text{ex}}$ tends toward zero as the dimensions of the element are reduced, expression (1.3) of u must contain a non-zero constant term (see section 1.5). Approximation u can then exactly represent the function $u_{\text{ex}} = \text{constant}$ for each element.

To ensure that error $\frac{\partial u}{\partial x} - \frac{\partial u_{\text{ex}}}{\partial x}$ tends toward zero with the decrease of element size, expressions (1.3) and (1.13) of u must independently contain the monomial x . Thus, the approximation $\frac{\partial u}{\partial x}$ can exactly represent the function $\frac{\partial u_{\text{ex}}}{\partial x} = \text{constant}$ for each element. In general, if the errors on u and all its derivatives up to order s are to tend toward zero with decrease in element size, expression (1.3) must contain a complete polynomial of order s . Furthermore, if u and its derivatives to the order $s - 1$ are continuous across inter-element boundaries, the error tends toward zero at any point in the domain V , including on the boundaries. When the continuity conditions are not satisfied, we must ensure that the discontinuities along the boundaries do not prevent errors from tending toward zero. This may be verified using the “patch test” technique [STR 73; IRO 72; DEV 74] described in section 4.1.3.

When transformation τ is linear, the conclusions with respect to approximation $u(\mathbf{x})$ on a real element transpose directly into approximation $u(\xi)$ on the reference element: the expression $u(\xi)$ must include a complete polynomial of order s in ξ, η and ζ . When transformation τ is not linear, the condition of a complete polynomial in x, y and z is transposed into a condition of a complete polynomial in ξ, η and ζ if and only if $\langle N \rangle \equiv \langle \bar{N} \rangle$ and $s \leq 1$ [CIA 78].

EXAMPLE 1.14. Complete polynomial for a three-noded triangular element

We verify that expressions $u(x, y)$ and $u(\xi, \eta)$ in Example 1.11 contain complete polynomials of order 1 in x, y and ξ, η . Thus, the errors on u and its first derivatives tend toward zero when the size of each element tends toward zero.

We verify that approximations $u(x, y)$ and $u(\xi, \eta)$ are able to represent the exact function $u_{\text{ex}}(x, y) = \text{constant} = u_0$. To do this, we place $u_i = u_j = u_k = u_0$ into expressions $u(x, y)$ and $u(\xi, \eta)$. We obtain:

$$\begin{aligned} u(x, y) &= (N_1(x, y) + N_2(x, y) + N_3(x, y))u_0 = u_0 \\ u(\xi, \eta) &= (N_1(\xi, \eta) + N_2(\xi, \eta) + N_3(\xi, \eta))u_0 = u_0 \end{aligned}$$

as the functions N verify:

$$\begin{aligned} N_1(x, y) + N_2(x, y) + N_3(x, y) &= 1 \\ N_1(\xi, \eta) + N_2(\xi, \eta) + N_3(\xi, \eta) &= 1 \end{aligned}$$

Definitions

- If the function $u(\mathbf{x})$ is continuous across boundaries between elements, the approximation is of **type (or class) C^0** . If $u(\mathbf{x})$ and its first derivatives are continuous, then the approximation is of **type C^1** . If $u(\mathbf{x})$ and its derivatives up to the order α are continuous, then the approximation is of **type C^α** .
- An element is said to be **isoparametric** if the geometrical transformation functions $\bar{N}(\xi)$ are identical to the interpolation functions $N(\xi)$. This implies that the geometrical nodes are the same as the interpolation nodes [ZIE 00].
- We say that an element is **pseudo-isoparametric** if functions $\bar{N}(\xi)$ and $N(\xi)$ are different polynomials using the same monomials.
- If the order of the polynomials used in $\bar{N}(\xi)$ is lower than that of the polynomials in $N(\xi)$, the element is **subparametric**. It is said to be **superparametric** in the opposite case. Superparametric elements are not usually recommended as they do not possess property (d) for convergence (see section 2.8.4).
- The number of nodal variables u_i associated with the total number of interpolation nodes of the element is called the number of degrees of freedom of the element and is written n_d .

The practical description of the interpolation nodes is identical to that used for the geometrical nodes. Their coordinates are stored in the same CORG table described in section 1.2.6. Similarly, the list of numbers of the two types of node of each element is placed in a CONEC table, as described in section 1.2.6. The actual distinction between geometrical nodes and interpolation nodes is made, if necessary, in the subprograms that calculate the various $\bar{N}(\xi)$ and $N(\xi)$ functions.

1.4 Construction of functions $N(\xi)$ and $\bar{N}(\xi)$

The geometrical transformation functions $\bar{N}(\xi)$ and the interpolation functions $N(\xi)$ have identical properties. They can sometimes be constructed directly with polynomials having the properties described in sections 1.2.4 and 1.3.2. These are often classic Lagrange or Hermite polynomials; however, there is no systematic manual technique that may be used to construct them. Only by experience we have been able to find functions $N(\xi)$ that correspond to a certain number of classic elements.

In the following sections, we will propose a general numerical method that is valid for all element types.

1.4.1 GENERAL METHOD OF CONSTRUCTION

a) Choice of a polynomial basis

Let us write $u(\xi)$ for the reference element in the form of a linear combination of known independent functions, $P_1(\xi), P_2(\xi), \dots$, which are most frequently independent monomials. The choice of functions $P_i(\xi)$ is one of the basic operations in the finite element method:

$$u(\xi) = \langle P_1(\xi) \quad P_2(\xi) \quad \dots \quad P_{n_d}(\xi) \rangle \begin{Bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{n_d} \end{Bmatrix} = \langle P(\xi) \rangle \{ a \} \quad (1.18)$$

The set of functions $P(\xi)$ constitutes the **polynomial basis** of the approximation. The number of terms in this set must be equal to the number of nodal variables or the number of degrees of freedom n_d of the element. A complete polynomial basis is usually preferred, but this is possible only for certain values of n_d . The following table lists the number of monomials needed to construct complete polynomials.

Degree of polynomial r	1 dimension n_d	2 dimensions n_d	3 dimensions n_d
1	2	3	4
2	3	6	10
3	4	10	20
4	5	15	35
5	6	21	56

EXAMPLE 1.15. Complete and incomplete polynomial bases

Number of dimensions	Degree of polynomial r	Polynomial bases $\langle P \rangle$	n_d
Complete bases			
1	1	$\langle 1 \ \xi \rangle$ (linear)	2
1	2	$\langle 1 \ \xi \ \xi^2 \rangle$ (quadratic)	3
2	1	$\langle 1 \ \xi \ \eta \rangle$ (linear)	3
2	2	$\langle 1 \ \xi \ \eta \ \xi^2 \ \xi\eta \ \eta^2 \rangle$ (quadratic)	6
3	1	$\langle 1 \ \xi \ \eta \ \zeta \ \rangle$ (linear)	4
3	2	$\langle 1 \ \xi \ \eta \ \zeta \ \xi^2 \ \xi\eta \ \eta^2 \ \eta\zeta \ \zeta^2 \ \xi\zeta \rangle$ (quadratic)	10
Incomplete bases			
2	2	$\langle 1 \ \xi \ \eta \ \xi\eta \rangle$ (bilinear)	4
3	3	$\langle 1 \ \xi \ \eta \ \zeta \ \xi\eta \ \eta\zeta \ \xi\zeta \ \xi\eta\zeta \rangle$ (trilinear)	8

To construct geometrical transformation functions, \bar{N} , we select expressions of \mathbf{x} in the same way, of the form:

$$\begin{aligned} x(\xi) &= \langle \bar{P}(\xi) \rangle \{a_x\} \\ y(\xi) &= \langle \bar{P}(\xi) \rangle \{a_y\} \\ z(\xi) &= \langle \bar{P}(\xi) \rangle \{a_z\} \end{aligned} \quad (1.19)$$

The number of functions $\bar{P}(\xi)$ and coefficients $\{a_x\}$, $\{a_y\}$ and $\{a_z\}$ is equal to the number of geometrical nodes of the element, \bar{n}^e .

Definitions

- Coefficients $\{a\}$ are called **generalized variables** of the element to distinguish them from **nodal variables** $\{u_n\}$.
- Expression $u(\xi) = \langle P(\xi) \rangle \{a\}$ defines the **generalized approximation** to be distinguished from the **nodal approximation** $u(\xi) = \langle N(\xi) \rangle \{u_n\}$.
- Coefficients $\{a_x\}$, $\{a_y\}$ and $\{a_z\}$ are sometimes called **generalized coordinates** of the element to distinguish them from the **nodal coordinates** $\{x_n\}$, $\{y_n\}$ and $\{z_n\}$ of the geometrical nodes.

b) Relationships between generalized variables and nodal variables

Let us point out that at each interpolation node of coordinates $\{\xi_i\}$, function $u(\xi)$ takes the nodal value $u_i = u(\xi_i)$:

$$\begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n_d} \end{Bmatrix} = \{u_n\} = \begin{Bmatrix} \langle P_1(\xi_1) & P_2(\xi_1) & \cdots & P_{n_d}(\xi_1) \rangle \\ \langle P_1(\xi_2) & P_2(\xi_2) & \cdots & P_{n_d}(\xi_2) \rangle \\ \cdots \\ \langle P_1(\xi_{n_d}) & P_2(\xi_{n_d}) & \cdots & P_{n_d}(\xi_{n_d}) \rangle \end{Bmatrix} \{a\}$$

$$\{u_n\} = [P_n] \{a\} \quad (1.20)$$

thus, inverting **nodal matrix** $[P_n]$ of order n_d :

$$\{a\} = [P_n]^{-1} \{u_n\} \quad (1.21)$$

In order to go from (1.20) to (1.21), $[P_n]$ should not be singular. If $[P_n]$ is singular, this implies that it will not be possible to express the parameters $\{a\}$ of relationship (1.18) in a unique manner as a function of the nodal variables $\{u_n\}$. This depends on the choice of polynomial basis and the coordinates $\{\xi_i\}$ of the nodes of the reference element. Since $[P_n]$ is independent of the geometry of the real element, the property of singularity of $[P_n]$ is a property of the reference element and not of the real element.

In a similar manner, we write equations (1.19) for the geometrical nodes:

$$\begin{aligned} \{x_n\} &= [\bar{P}_n][a_x] \\ \{\gamma_n\} &= [\bar{P}_n][a_y] \\ \{z_n\} &= [\bar{P}_n][a_z] \end{aligned} \quad (1.22)$$

thus, after inversion of $[\bar{P}_n]$:

$$\begin{aligned} \{a_x\} &= [\bar{P}_n]^{-1} [x_n] \\ \{a_y\} &= [\bar{P}_n]^{-1} [\gamma_n] \\ \{a_z\} &= [\bar{P}_n]^{-1} [z_n] \end{aligned} \quad (1.23)$$

c) Expressions of functions N and \bar{N}

Substituting (1.21) into (1.18):

$$\begin{aligned} u(\xi) &= \langle P(\xi) \rangle [P_n]^{-1} \{u_n\} \\ \text{or} \quad u(\xi) &= \langle N(\xi) \rangle \{u_n\} \end{aligned} \quad (1.24)$$

$$\text{hence} \quad \langle N(\xi) \rangle = \langle P(\xi) \rangle [P_n]^{-1}.$$

In a similar manner, we obtain the functions \bar{N} :

$$\begin{aligned} x(\xi) &= \langle \bar{N}(\xi) \rangle \{x_n\} \\ y(\xi) &= \langle \bar{N}(\xi) \rangle \{y_n\} \\ z(\xi) &= \langle \bar{N}(\xi) \rangle \{z_n\} \end{aligned} \quad (1.25)$$

where

$$\langle \bar{N}(\xi) \rangle = \langle \bar{P}(\xi) \rangle [\bar{P}_n]^{-1}.$$

d) Derivative of function $u(\xi)$

Differentiating (1.24), we obtain:

$$\begin{Bmatrix} u_{,\xi} \\ u_{,\eta} \\ u_{,\zeta} \end{Bmatrix} = \begin{bmatrix} \langle P_{,\xi} \rangle \\ \langle P_{,\eta} \rangle \\ \langle P_{,\zeta} \rangle \end{bmatrix} [P_n]^{-1} \{u_n\} = \begin{bmatrix} \langle N_{,\xi} \rangle \\ \langle N_{,\eta} \rangle \\ \langle N_{,\zeta} \rangle \end{bmatrix} \{u_n\} = [B_\xi] \{u_n\}. \quad (1.26)$$

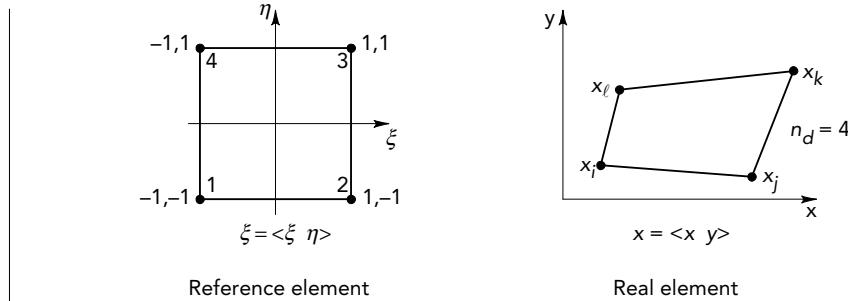
e) Summary of operations required to construct $\langle N \rangle$

- Choice of polynomial basis $\langle P(\xi) \rangle$
- Evaluation of nodal matrix $[P_n] = [P_j(\xi_i)]$; $i, j = 1, 2, \dots, n_d$
- Inversion of nodal matrix $[P_n]$
- Computation of $\langle N \rangle$ at desired points, ξ :

$$\langle N(\xi) \rangle = \langle P(\xi) \rangle [P_n]^{-1}.$$

It is important to note that these operations should only be carried out once for all of the real elements that have the same reference element.

EXAMPLE 1.16. Construction of functions $N(\xi)$ for an isoparametric four-noded quadrilateral element



Since the element is isoparametric, geometrical nodes and interpolation nodes are identical.

a) Choice of the polynomial basis

We have $n_d = 4$ nodal variables, and therefore we cannot use a complete polynomial. The best choice, respecting the symmetry and the continuity of u between elements, is a linear basis in terms of ξ and η :

$$\langle P \rangle = \langle 1 \quad \xi \quad \eta \quad \xi\eta \rangle.$$

Note that $u(\xi) = \langle P \rangle \{a\}$ becomes linear on each side $\xi = \pm 1$ and $\eta = \pm 1$, thus ensuring inter-element continuity.

b) Evaluation of $[P_n]$

Let us evaluate $\langle P(\xi) \rangle$ for each of the four nodes of coordinates ξ_i :

$$[P_n] = \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \quad \begin{aligned} \{\xi_n\} &= \begin{cases} -1 \\ 1 \\ 1 \\ -1 \end{cases} \\ \{\eta_n\} &= \begin{cases} -1 \\ -1 \\ 1 \\ 1 \end{cases} \end{aligned}$$

Remark: In this case, the matrix is orthogonal.

c) Inversion of $[P_n]$

$$[P_n]^{-1} = \frac{1}{4} [P_n]^T = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}$$

d) Expression of $\langle N \rangle$

$$\begin{aligned} \langle N \rangle &= \langle N_1 \ N_2 \ N_3 \ N_4 \rangle = \langle P \rangle [P_n]^{-1} \\ \langle N \rangle &= \left\langle \frac{1-\xi-\eta+\xi\eta}{4}, \ \frac{1+\xi-\eta-\xi\eta}{4}, \ \frac{1+\xi+\eta+\xi\eta}{4}, \ \frac{1-\xi+\eta-\xi\eta}{4} \right\rangle \\ \langle N \rangle &= \frac{1}{4} \langle (1-\xi)(1-\eta); \ (1+\xi)(1-\eta); \ (1+\xi)(1+\eta); \ (1-\xi)(1+\eta) \rangle. \end{aligned}$$

The element is isoparametric:

$$\langle \bar{N} \rangle \equiv \langle N \rangle$$

1.4.2 ALGEBRAIC PROPERTIES OF FUNCTIONS N AND \bar{N}

a) Each interpolation function $N_i(\xi)$ is formed by the scalar product of the polynomial basis $\langle P(\xi) \rangle$ and column i of matrix $[P_n]^{-1}$:

$$N_i(\xi) = \langle P(\xi) \rangle \{C_i\} \quad (1.27)$$

where $[P_n]^{-1} = [\{C_1\} \ \{C_2\} \ \dots \ \{C_i\} \ \dots]$. (1.28)

Function $N_i(\xi)$ is thus a linear combination of the functions $\langle P(\xi) \rangle$, the coefficients being the terms from column $\{C_i\}$ of $[P_n]^{-1}$. The matrix $[P_n]^{-1}$ may be considered as a convenient method of storage for the coefficients of all the functions N_i .

b) Relation (1.24)

$$\langle N(\xi) \rangle = \langle P(\xi) \rangle [P_n]^{-1}$$

may be written by multiplying both members on the right by $[P_n]$:

$$\langle N(\xi) \rangle [P_n] = \langle P(\xi) \rangle \quad (1.29)$$

Using definition (1.20) of $[P_n]$:

$$\sum_{i=1}^{n_d} N_i(\xi) P_j(\xi_i) \equiv P_j(\xi) \quad j = 1, 2, \dots, n_d \quad (1.30)$$

This relation is characteristic of the algebraic structure of functions N_i . It shows that the terms $P_j(\xi)$ belong to the polynomial basis used to construct functions N_i .

Let us suppose that a set N_i has been created empirically. We can check that the approximation $u(\xi) = \langle N \rangle \{u_n\}$ includes any polynomial $p(\xi)$ by verifying that:

$$\sum_{i=1}^{n_d} N_i(\xi) p(\xi_i) \equiv p(\xi). \quad (1.31)$$

When studying convergence, we need to know which monomials are included in the function $u(\xi)$. For example, to verify that the monomials 1, ξ and η are present in $u(\xi)$, we ensure that:

$$\begin{aligned} \sum_{i=1}^{n_d} N_i(\xi) &= 1 \\ \sum_{i=1}^{n_d} N_i(\xi) \xi_i &= \xi \\ \sum_{i=1}^{n_d} N_i(\xi) \eta_i &= \eta. \end{aligned}$$

EXAMPLE 1.17. Properties of functions N of a linear quadrilateral

We can verify that the functions N_i of Example 1.16 satisfy relations (1.30) corresponding to the monomials 1, ξ , η , $\xi\eta$:

$$\begin{aligned} \sum_{i=1}^4 N_i(\xi, \eta) &= N_1 + N_2 + N_3 + N_4 = 1 \\ \sum_{i=1}^4 N_i(\xi, \eta) \xi_i &= -N_1 + N_2 + N_3 - N_4 = \xi \\ \sum_{i=1}^4 N_i(\xi, \eta) \eta_i &= -N_1 - N_2 + N_3 + N_4 = \eta \\ \sum_{i=1}^4 N_i(\xi, \eta) \xi_i \eta_i &= N_1 - N_2 + N_3 - N_4 = \xi\eta \end{aligned}$$

c) Differentiation of (1.30) gives:

$$\sum_{i=1}^{n_d} \frac{\partial N_i(\xi)}{\partial \xi} P_j(\xi_i) \equiv \frac{\partial P_j(\xi)}{\partial \xi} \quad (1.32)$$

Such expressions, together with (1.15), (1.16), (1.17) and (1.30), are useful to verify the accuracy of explicit forms of functions N_i and their derivatives.

1.5 Transformation of derivation operators [ZIE 00]

1.5.1 GENERAL REMARKS

The equations of the physical problem under study are written in the real domain; they involve unknown functions u_{ex} and their derivatives in \mathbf{x} : $\frac{\partial u_{\text{ex}}}{\partial x}, \frac{\partial u_{\text{ex}}}{\partial y}$ etc. Since approximation (1.13) is often very complicated using the real element, we systematically apply approximation (1.14) on the reference element:

$$u_{\text{ex}} \approx u(\xi) = \langle N(\xi) \rangle \{u_n\} \quad (1.33)$$

together with transformation (1.12):

$$\begin{aligned} \tau: \xi &\rightarrow \mathbf{x} = \mathbf{x}(\xi) = [\bar{N}(\xi)] \{x_n\} \\ \mathbf{x} &= \langle x \ y \ z \rangle \\ \xi &= \langle \xi \ \eta \ \zeta \rangle \end{aligned} \quad (1.34)$$

Transformation τ being bijective:

$$\tau^{-1}: \mathbf{x} \rightarrow \xi = \xi(\mathbf{x}). \quad (1.35)$$

While τ^{-1} always exists, it is easy to construct explicitly only if τ is linear, for example in the case of the three-noded triangular element in Example 1.11. In the case of the quadrilateral four-noded element in Example 1.16, construction of τ^{-1} is already complicated. If we have an explicit form of (1.35), we can replace $\xi(\mathbf{x})$ in (1.33) to obtain an approximation for the real element:

$$u(\xi(\mathbf{x})) = \langle N(\xi(\mathbf{x})) \rangle \{u_n\} = \langle N(\mathbf{x}) \rangle \{u_n\} = u(\mathbf{x}).$$

In reality, it is more convenient to perform all operations in the space of reference elements. All expressions that involve derivatives of u in terms of x, y and z are transformed into derivatives in terms of ξ, η and ζ using the transformation matrix, known as the Jacobian matrix $[J]$.

1.5.2 FIRST DERIVATIVES

Let us use the chain rule to calculate the derivatives in ξ of a function from its derivatives using \mathbf{x} :

$$\begin{Bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \\ \frac{\partial}{\partial \zeta} \end{Bmatrix} = \begin{Bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{Bmatrix} \begin{Bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{Bmatrix} \quad (1.36a)$$

which is written as:

$$\{\partial_{\xi}\} = [J] \{\partial_x\} \quad (1.36b)$$

where $[J]$ is the Jacobian matrix of the geometrical transformation. The terms of $[J]$ may be obtained easily by derivation of (1.34). Similarly, the derivatives in \mathbf{x} of a function may be obtained from the derivatives in ξ :

$$\begin{Bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} \end{Bmatrix} = \begin{Bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} & \frac{\partial \zeta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} & \frac{\partial \zeta}{\partial y} \\ \frac{\partial \xi}{\partial z} & \frac{\partial \eta}{\partial z} & \frac{\partial \zeta}{\partial z} \end{Bmatrix} \begin{Bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \\ \frac{\partial}{\partial \zeta} \end{Bmatrix} \quad (1.37a)$$

so that

$$\{\partial_x\} = [j] \{\partial_{\xi}\}. \quad (1.37b)$$

Substituting (1.37b) into (1.36b), we obtain:

$$[j] = [J]^{-1}. \quad (1.38)$$

Matrix $[j]$ is used in practice since we must express derivatives of u in x , y and z based on derivatives of u in ξ , η and ζ . As the terms of $[j]$ are derived from relationship (1.35), which is not explicitly known, we use expression (1.38) to calculate $[j]$ using the terms of $[J]$. We have supposed the transformation τ to be bijective, and consequently the inverse of $[J]$ exists at all points in the reference element.

Expression of $[j] = [J]^{-1}$

We show explicit forms of the inverse of $[J]$ in one, two or three dimensions:

— One dimension:

$$[J] = J_{11}; \quad [j] = [J]^{-1} = \frac{1}{J_{11}}. \quad (1.39)$$

— Two dimensions:

$$[J] = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}; \quad [J]^{-1} = \frac{1}{\det(J)} \begin{bmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11} \end{bmatrix} \quad (1.40)$$

$$\det(J) = J_{11} J_{22} - J_{12} J_{21}.$$

— Three dimensions:

$$[J] = \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix}$$

$$[J]^{-1} = \frac{1}{\det(J)} \begin{bmatrix} J_{22} J_{33} - J_{32} J_{23}; & J_{13} J_{32} - J_{12} J_{33}; & J_{12} J_{23} - J_{13} J_{22} \\ J_{31} J_{23} - J_{21} J_{33}; & J_{11} J_{33} - J_{13} J_{31}; & J_{21} J_{13} - J_{23} J_{11} \\ J_{21} J_{32} - J_{31} J_{22}; & J_{12} J_{31} - J_{32} J_{11}; & J_{11} J_{22} - J_{12} J_{21} \end{bmatrix} \quad (1.41)$$

$$\det(J) = J_{11} (J_{22} J_{33} - J_{32} J_{23}) + J_{12} (J_{31} J_{23} - J_{21} J_{33}) +$$

$$+ J_{13} (J_{21} J_{32} - J_{31} J_{22})$$

Computations of the terms of $[J]$

The terms of $[J]$ are obtained following (1.36a) by differentiating relation (1.12) in terms of ξ . We rewrite equation (1.12) in the form:

$$\langle x \ y \ z \rangle = \langle \bar{N}(\xi) \rangle [\{x_n\} \ \{y_n\} \ \{z_n\}] \quad (1.42)$$

$\{x_n\} \ \{y_n\} \ \{z_n\}$ being the x, y and z geometrical node coordinates of the element.
The Jacobian matrix is written as:

$$[J] = \begin{bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \\ \frac{\partial}{\partial \zeta} \end{bmatrix} \langle x \ y \ z \rangle = \begin{bmatrix} \langle \bar{N}_{,\xi} \rangle \\ \langle \bar{N}_{,\eta} \rangle \\ \langle \bar{N}_{,\zeta} \rangle \end{bmatrix} [\{x_n\} \ \{y_n\} \ \{z_n\}] \quad (1.43)$$

$$(3 \times \bar{n}^e) \quad (\bar{n}^e \times 3)$$

It is obtained as the product of two matrices, one containing the derivatives of the geometrical transformation functions in terms of ξ and the other containing the coordinates of the geometrical nodes of the element, \bar{n}^e .

Transformation of an integral

The change of variables (1.34) allows us to change the integration of a function f on the real element V^e to a simpler integration using the reference element V^r :

$$\int_{V^e} f(\mathbf{x}) dx dy dz = \int_{V^r} f(\mathbf{x}(\xi)) \det(J) d\xi d\eta d\zeta \quad (1.44)$$

$\det(J)$ being the determinant of the Jacobian matrix $[J]$. The volume element dV is the mixed product:

$$dV = (\vec{dx} \times \vec{dy}) \cdot \vec{dz}$$

In orthonormal Cartesian coordinates:

$$d\vec{x} = dx \cdot \vec{i}; \quad d\vec{y} = dy \cdot \vec{j}; \quad d\vec{z} = dz \cdot \vec{k}$$

where $\vec{i}, \vec{j}, \vec{k}$ are unitary vectors carried by the axes. So:

$$dV = dx \ dy \ dz$$

In the curvilinear referential (ξ, η, ζ) :

$$dV = (\vec{d\xi} \times \vec{d\eta}) \cdot \vec{d\zeta}.$$

The components for these vectors in a Cartesian referential are:

$$\begin{aligned} \vec{d\xi} &= (J_{11} \vec{i} + J_{12} \vec{j} + J_{13} \vec{k}) d\xi \\ \vec{d\eta} &= (J_{21} \vec{i} + J_{22} \vec{j} + J_{23} \vec{k}) d\eta \\ \vec{d\zeta} &= (J_{31} \vec{i} + J_{32} \vec{j} + J_{33} \vec{k}) d\zeta \end{aligned}$$

The mixed product is thus written as:

$$dV = \det(J) d\xi d\eta d\zeta.$$

EXAMPLE 1.18. Jacobian matrix of a four-noded quadrilateral element

Using (1.43):

$$[J] = \frac{1}{4} \begin{bmatrix} -(1-\eta) & (1-\eta) & (1+\eta) & -(1+\eta) \\ -(1-\xi) & -(1+\xi) & (1+\xi) & (1-\xi) \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \\ x_4 & y_4 \end{bmatrix}$$

$$[J] = \frac{1}{4} \begin{bmatrix} -x_1 + x_2 + x_3 - x_4 & -y_1 + y_2 + y_3 - y_4 \\ +\eta(x_1 - x_2 + x_3 - x_4) & +\eta(y_1 - y_2 + y_3 - y_4) \\ -x_1 - x_2 + x_3 + x_4 & -y_1 - y_2 + y_3 + y_4 \\ +\xi(x_1 - x_2 + x_3 - x_4) & +\xi(y_1 - y_2 + y_3 - y_4) \end{bmatrix}$$

$$\det(J) = A_0 + A_1\xi + A_2\eta$$

$$A_0 = \frac{1}{8}[(y_4 - y_2)(x_3 - x_1) - (y_3 - y_1)(x_4 - x_2)]$$

$$A_1 = \frac{1}{8}[(y_3 - y_4)(x_2 - x_1) - (y_2 - y_1)(x_3 - x_4)]$$

$$A_2 = \frac{1}{8}[(y_4 - y_1)(x_3 - x_2) - (y_3 - y_2)(x_4 - x_1)]$$

In the particular case where the element is rectangular with sides $x_2 - x_1 = 2a$ and $y_4 - y_1 = 2b$:

$$x_1 = x_4 \quad x_2 = x_3$$

$$y_1 = y_2 \quad y_3 = y_4$$

$$[J] = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}, \quad \det(J) = a b.$$

and

$$\int_{V^e} f(\mathbf{x}) dx dy = \int_{-1}^1 \int_{-1}^1 f(\mathbf{x}(\xi)) a b d\xi d\eta.$$

1.5.3 SECOND DERIVATIVES

We now look for expressions of second derivatives with respect to \mathbf{x} using the **first and second derivatives with respect to ξ** . By differentiating relation (1.37a) as a function of \mathbf{x} , we obtain the following relationship:

$$\begin{bmatrix} \frac{\partial^2}{\partial x^2} \\ \frac{\partial^2}{\partial y^2} \\ \frac{\partial^2}{\partial z^2} \\ \frac{\partial^2}{\partial x \partial y} \\ \frac{\partial^2}{\partial y \partial z} \\ \frac{\partial^2}{\partial x \partial z} \end{bmatrix} = [T_1] \begin{bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \\ \frac{\partial}{\partial \zeta} \end{bmatrix} + [T_2] \begin{bmatrix} \frac{\partial^2}{\partial \xi^2} \\ \frac{\partial^2}{\partial \eta^2} \\ \frac{\partial^2}{\partial \zeta^2} \\ \frac{\partial^2}{\partial \xi \partial \eta} \\ \frac{\partial^2}{\partial \eta \partial \zeta} \\ \frac{\partial^2}{\partial \xi \partial \zeta} \end{bmatrix} \quad (1.45a)$$

which is written as:

$$\{\partial_x^2\} = [T_1]\{\partial_\xi\} + [T_2]\{\partial_\xi^2\} \quad (1.45b)$$

a) Computation of $[T_1]$

Matrix $[T_1]$ involves the derivatives in terms of ξ of the terms of $[j]$ defined in (1.37a). $[T_1]$ is nullified if $[j]$ is constant, i.e. if the functions $\bar{N}(\xi)$ are linear. As we do not usually have an explicit expression of $[j]$, we now propose a method for the evaluation of $[T_1]$ without using the derivatives of $[j]$.

By differentiating (1.36), we obtain:

$$\{\partial_\xi^2\} = [C_1]\{\partial_x\} + [C_2]\{\partial_x^2\} \quad (1.46a)$$

Using (1.37b):

$$\{\partial_\xi^2\} = [C_1][j]\{\partial_\xi\} + [C_2]\{\partial_x^2\}. \quad (1.46b)$$

Let us insert (1.46b) into (1.45b):

$$\{\partial_x^2\} = ([T_1] + [T_2][C_1][j])\{\partial_\xi\} + [T_2][C_2]\{\partial_x^2\}.$$

Hence, the two relationships:

$$[T_2][C_2] = [I] \text{ therefore } [T_2] = [C_2]^{-1} \quad (1.47a)$$

$$[T_1] + [T_2][C_1][j] = 0 \text{ therefore } [T_1] = -[T_2][C_1][j]. \quad (1.47b)$$

Matrices $[T_2]$ and $[C_1]$ are given explicitly by (1.48) and (1.49) and matrix $[I]$ is the identity matrix.

b) Computation of $[T_2]$ and $[C_1]$

Matrix $[T_2]$, defined by (1.45b), is expressed directly as a function of the terms of $[j]$ given in (1.38) and (1.43):

$$[T_2] = \begin{bmatrix} j_{11}^2 & j_{12}^2 & j_{13}^2 & | & 2 j_{11} j_{12} & 2 j_{12} j_{13} & 2 j_{13} j_{11} \\ j_{21}^2 & j_{22}^2 & j_{23}^2 & | & 2 j_{21} j_{22} & 2 j_{22} j_{23} & 2 j_{23} j_{21} \\ j_{31}^2 & j_{32}^2 & j_{33}^2 & | & 2 j_{31} j_{32} & 2 j_{32} j_{33} & 2 j_{33} j_{31} \\ \hline j_{11} j_{21} & j_{12} j_{22} & j_{13} j_{23} & | & j_{11} j_{22} & j_{12} j_{23} & j_{11} j_{23} \\ j_{21} j_{31} & j_{22} j_{32} & j_{23} j_{33} & | & +j_{12} j_{21} & +j_{13} j_{22} & +j_{13} j_{21} \\ j_{31} j_{11} & j_{32} j_{12} & j_{33} j_{13} & | & +j_{21} j_{32} & +j_{22} j_{33} & j_{21} j_{33} \\ & & & | & +j_{22} j_{31} & +j_{23} j_{32} & +j_{23} j_{31} \\ & & & | & +j_{31} j_{12} & j_{32} j_{13} & j_{31} j_{13} \\ & & & | & +j_{32} j_{11} & +j_{33} j_{12} & +j_{33} j_{11} \end{bmatrix} = [C_2]^{-1} \quad (1.48)$$

Matrix $[C_1]$ defined by (1.46a) is written as:

$$[C_1] = \begin{bmatrix} \frac{\partial}{\partial \xi} \langle J_{11} \ J_{12} \ J_{13} \rangle \\ \frac{\partial}{\partial \eta} \langle J_{21} \ J_{22} \ J_{23} \rangle \\ \frac{\partial}{\partial \zeta} \langle J_{31} \ J_{32} \ J_{33} \rangle \\ \hline \frac{1}{2} \left(\frac{\partial}{\partial \eta} \langle J_{11} \ J_{12} \ J_{13} \rangle + \frac{\partial}{\partial \xi} \langle J_{21} \ J_{22} \ J_{23} \rangle \right) \\ \frac{1}{2} \left(\frac{\partial}{\partial \zeta} \langle J_{21} \ J_{22} \ J_{23} \rangle + \frac{\partial}{\partial \eta} \langle J_{31} \ J_{32} \ J_{33} \rangle \right) \\ \frac{1}{2} \left(\frac{\partial}{\partial \xi} \langle J_{31} \ J_{32} \ J_{33} \rangle + \frac{\partial}{\partial \zeta} \langle J_{11} \ J_{12} \ J_{13} \rangle \right) \end{bmatrix} \quad (1.49)$$

Note that the expression of $[C_2]$ is identical to that of $[T_2]$ if we replace the terms of $[j]$ with the corresponding terms of $[J]$ in $[T_2]$. In two dimensions, we

obtain:

$$[T_2] = \begin{bmatrix} j_{11}^2 & j_{12}^2 & 2j_{11}j_{12} \\ j_{21}^2 & j_{22}^2 & 2j_{21}j_{22} \\ j_{11}j_{21} & j_{12}j_{22} & j_{11}j_{22} + j_{12}j_{21} \end{bmatrix} \quad (1.50a)$$

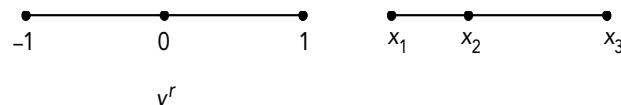
$$[C_1] = \begin{bmatrix} \frac{\partial}{\partial \xi} \langle J_{11} \ J_{12} \rangle \\ \frac{\partial}{\partial \eta} \langle J_{21} \ J_{22} \rangle \\ \frac{1}{2} \left(\frac{\partial}{\partial \eta} \langle J_{11} \ J_{12} \rangle + \frac{\partial}{\partial \xi} \langle J_{21} \ J_{22} \rangle \right) \end{bmatrix} \quad (1.50b)$$

1.5.4 SINGULARITY OF THE JACOBIAN MATRIX

The singularity of $[J]$ at a point in the reference element implies that the transformation τ is not bijective. This singularity appears when the element is deformed considerably. It is advisable to check that the determinant of $[J]$ is positive at all points, ξ , of the reference element.

EXAMPLE 1.19. Singularity in a one-dimensional quadratic transformation

Let us consider a one-dimensional quadratic isoparametric element:



$$\langle N \rangle = \langle \bar{N} \rangle = \left\langle \frac{1}{2} \xi (\xi - 1) \quad -(\xi - 1) (\xi + 1) \quad \frac{1}{2} \xi (\xi + 1) \right\rangle$$

$$[J] = \left\langle \frac{2\xi - 1}{2} \quad -2\xi \quad \frac{2\xi + 1}{2} \right\rangle \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix}.$$

Let us take $x_1 = 0$ and $x_3 = l$. In this case, the determinant of $[J]$ is expressed as a function of x_2 :

$$\det(J) = (\xi + 0.5)l - 2\xi x_2.$$

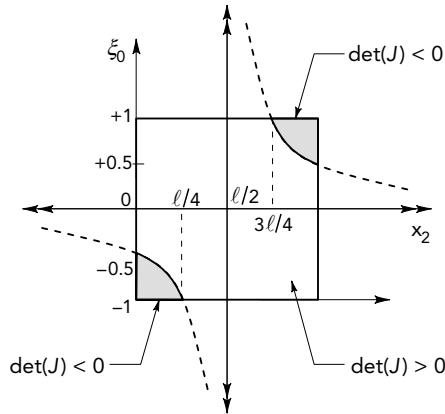
The determinant of $[J]$ vanishes at point:

$$\xi_0 = \frac{0.5l}{2x_2 - l}.$$

Point ξ_0 is only located inside the reference element if:

$$-1 \leq \xi_0 \leq 1$$

so that $-1 \leq \frac{0.5l}{2x_2 - l} \leq 1 \quad \text{with} \quad 0 \leq x_2 \leq l$.

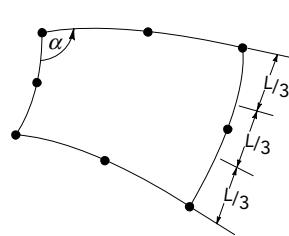


When $\frac{l}{4} < x_2 < \frac{3l}{4}$, the determinant $\det(J)$ does not vanish on the element. When

$x_2 = \frac{l}{4}$, there is a singularity of $[J]$ in $\xi = -1$.

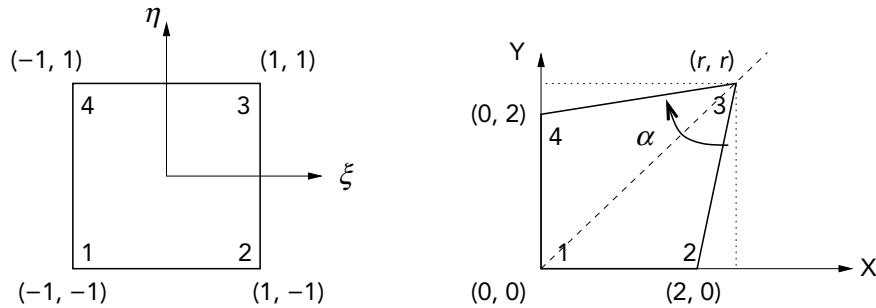
To avoid this singularity in an eight-noded element (section 9.2), Zienkiewicz [ZIE 00, p. 205] proposes the following conditions:

- The four angles α must be less than 180° .
- The middle nodes must be located in the central third of each edge.



EXAMPLE 1.20. Singularity of matrix [J] in the case of a four-noded quadrilateral

In the specific case which follows, the value of $\det(J)$ varies as a function of the position of node 3:



with: $x_1 = 0, y_1 = 0; x_2 = 2, y_2 = 0; x_4 = 0, y_4 = 2;$

and $x_3 = y_3 = r.$

$$\text{We obtain: } x(\xi, \eta) = \frac{1}{4}(1 + \xi)(2 + r - \eta(2 - r));$$

$$y(\xi, \eta) = \frac{1}{4}(1 + \eta)(2 + r - \xi(2 - r));$$

The Jacobian matrix is then (see Example 1.18):

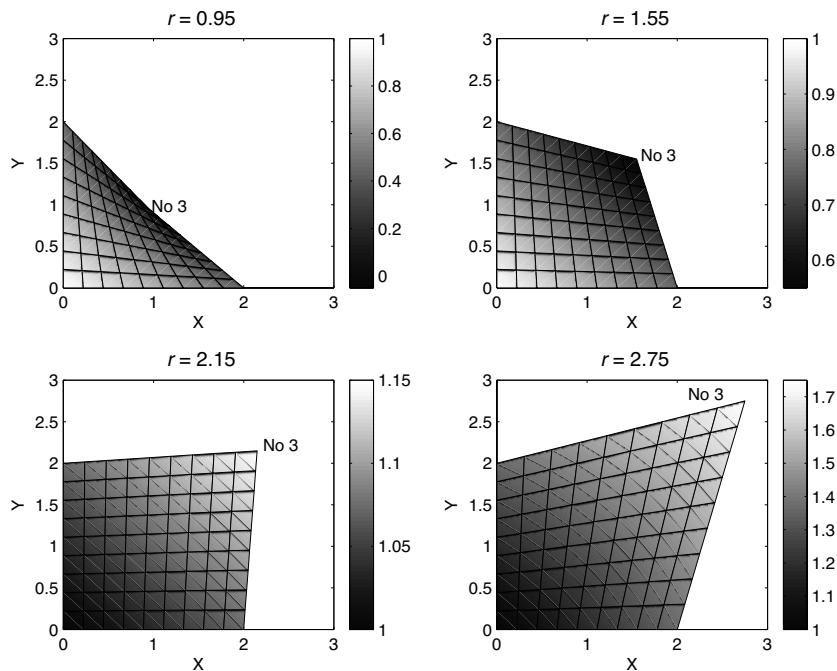
$$[J] = \frac{1}{4} \begin{bmatrix} a - b\eta & -b(1 + \eta) \\ -b(1 + \xi) & a - b\xi \end{bmatrix}, \quad a = 2 + r, \quad b = 2 - r,$$

$$\text{and } \det(J) = \frac{1}{2} \left(r - \left(1 - \frac{r}{2} \right) (\xi + \eta) \right).$$

Depending on the value of r , the determinant of J takes the following values:

r	$\det(J)$	Remark
0.5	$\frac{1}{4}(1 - 1.5\xi - 1.5\eta)$	Positive for $(\xi + \eta) < 2/3$
1	$\frac{1}{2}(1 - 0.5\xi - 0.5\eta)$	Null at node 3
2	1	

The area of the element is equal to $2r$. In this case, we note that the determinant remains positive for: $(\xi + \eta) < \frac{2r}{2-r}$; $\alpha < 180^\circ$.



These figures illustrate the “mappings” of the Jacobian of J and for different positions of node 3 ($r = 0.95, 1.55, 2.15$ and 2.75). We consider that the transformation is singular for any change in the sign of the determinant of J .

1.6 Computation of functions N , their derivatives and the Jacobian matrix

1.6.1 GENERAL REMARKS

In the subsequent chapters, we need approximations of:

$$u(\mathbf{x}), \frac{\partial u(\mathbf{x})}{\partial x}, \frac{\partial u(\mathbf{x})}{\partial y} \text{ etc.}$$

These approximations are used to evaluate integrals over the volume of a real element (see the example in section 1.8 and Chapter 4):

$$k = \int_{V^e} f \left(u(\mathbf{x}), \frac{\partial u(\mathbf{x})}{\partial x}, \dots \right) dV^e. \quad (1.51a)$$

These integrals are replaced by integrals of the reference element using relations (1.37) and (1.44):

$$k = \int_{V^r} f \left(u(\xi), \frac{\partial u(\xi)}{\partial \xi}, \dots [j] \right) \det(J(\xi)) dV^r. \quad (1.51b)$$

These integrals, with complex expressions, are themselves evaluated by a numerical integration technique that will be examined in detail in section 5.1:

$$k \approx \sum_r w_r f \left(u(\xi_r), \frac{\partial u(\xi_r)}{\partial \xi}, \dots [j(\xi_r)] \right) \det(J(\xi_r)) \quad (1.51c)$$

where ξ_r are the coordinates of a set of integration points on the reference elements, for example the quadrature points w_r are numerical integration coefficients:

$$\begin{aligned} u(\xi_r) &= \langle N(\xi_r) \rangle \{u_n\} \\ \frac{\partial u(\xi_r)}{\partial \xi} &= \langle \frac{\partial N(\xi_r)}{\partial \xi} \rangle \{u_n\}. \end{aligned}$$

$[j(\xi_r)]$ and $\det(J(\xi_r))$ are the inverse of the Jacobian matrix and its determinant evaluated at point ξ_r .

Note that the expressions of $\langle N(\xi_r) \rangle$ and $\langle \frac{\partial N(\xi_r)}{\partial \xi} \rangle$ are independent of the geometry of the real element. They depend only on characteristics of the reference element:

- coordinates of the interpolation nodes, $\{\xi_i\}$;
- the polynomial basis, $\langle P(\xi) \rangle$;
- the coordinates of the integration points, $\{\xi\}$.

We can consequently evaluate $\langle N(\xi_r) \rangle$ and its derivatives **once and for all** for each element, i.e. for each *type* of element.

The Jacobian matrix $[j(\xi_r)]$ and its determinant use the expressions of the first derivatives $\langle \frac{\partial N(\xi_r)}{\partial \xi} \rangle$ and the coordinates of the geometrical nodes of each real element. We therefore need to calculate them for each real element. If the element is not isoparametric, we must use $\langle \bar{N} \rangle \neq \langle N \rangle$. Figure 1.2 shows the organization of the calculations of $\langle N \rangle$, $[J]$ and k .

Note that the efficiency of the method used to calculate functions N and their derivatives is not critical as this calculation is carried out only once for each type of reference element.

1.6.2 EXPLICIT FORMS FOR N

In the early years of the development of the finite element method, the concepts of reference elements and numerical integration were not widely used. In general, users formulated functions, $N(x)$, and elementary integrals of the type (1.51a) explicitly for the real element V^e .

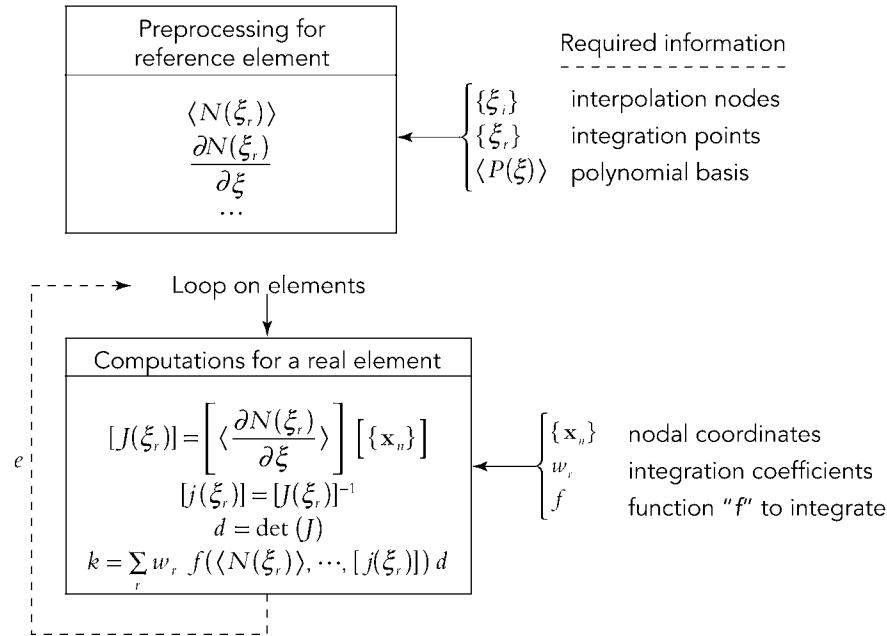


Figure 1.2. Organization of calculations of N , $[J]$ and k

Through systematic use of the reference element, we can replace complicated functions, $\langle N(\mathbf{x}) \rangle$, with simpler $\langle N(\xi) \rangle$ functions. Moreover, these functions can be constructed automatically using a computer program.

If we know the explicit form of $\langle N(\xi) \rangle$ for a reference element, we can manually derive these expressions and evaluate them at each integration point, $\{\xi_r\}$. The manual construction and differentiation of $\langle N \rangle$ functions and the creation of the calculation program are long and delicate procedures and are often a source of errors: for example, for a 32-node three-dimensional element (section 2.6.3), we need to construct 128 cubic and quadratic functions.

1.7 Approximation errors on an element

1.7.1 NOTIONS OF APPROXIMATION ERRORS [HUG 87; IMB 84; KIK 86]

In this section, we introduce the fundamental principals for estimating the truncation or approximation errors for an element. We identify the conditions necessary to force the errors in u and its derivatives to tend toward zero when the dimensions, l , of the element tend toward zero. We also obtain an approximate expression of the error in the form $e \leq c l^\alpha$, where c and α are constants depending on the type of approximation used.

The approximation error at any point \mathbf{x} in the real element V^e is defined by (1.7):

$$e(\mathbf{x}) = u(\mathbf{x}) - u_{ex}(\mathbf{x}). \quad (1.52)$$

The error at point ξ in the reference element is:

$$e(\xi) = u(\xi) - u_{ex}(\xi). \quad (1.53)$$

For two points, \mathbf{x} and ξ , which correspond to one another in transformation (1.12), errors $e(\mathbf{x})$ and $e(\xi)$ take the same value. To characterize the maximum error in the element, we use the **maximum norm** of function $e(\mathbf{x})$:

$$|e| = \text{Maximum on } V^e \text{ of } |e(\mathbf{x})|. \quad (1.54)$$

Remarks

- The so-called quadratic norm L^2 is calculated as:

$$\|e\|^2 = \int_{V^e} e^2(x) dV \quad \text{and} \quad \|e\| = \left(\int_{V^e} e^2(x) dV \right)^{1/2}$$

- The root mean square, L^2 , norm is given by:

$$\|e\|_m = \|e\| / \sqrt{V^e}, \quad \text{where } V^e \text{ is the volume of the element.}$$

- The energy norm is an L^2 norm weighted by the constitutive matrix, and is often used in mechanics:

$$\|e\|_{energy} = \left(\int_{V^e} e.H.e dV \right)^{1/2}, \quad \text{where } H \text{ is the elasticity matrix.}$$

We also have the following relationship: $\|e\| \leq |e| \sqrt{V^e}$.

For cases in one dimension, we have:

$$\|e\| \leq |e| \sqrt{h}, \quad \text{where } h \text{ is the length of the element.}$$

Define the error for each derivative of order s by:

$$e_s(\mathbf{x}) = D^s(e(\mathbf{x})) = \frac{\partial^s e(\mathbf{x})}{\partial^a x \partial^b y \partial^c z}, \quad a+b+c=s, \quad a, b, c, s \geq 0.$$

The corresponding norm is:

$$|e|_s = \text{Maximum on } V^e \text{ of } |D^s(e(\mathbf{x}))| \quad (1.55)$$

for all values of a, b, c such that $a+b+c=s$.

The semi-norm L^2 is often used in the finite element method:

$$\|e\|_s^2 = \sum_{a+b+c=s} \int_{V^e} (D^s(e(\mathbf{x})))^2 dV^e. \quad (1.56)$$

According to Strang [STR 73, pp. 142–144], the norms $|e|_s$ and $\|e\|_s^2$ may be presented in the following standard forms:

$$|e|_s \leq c l^{n-s} |u_{ex}(\mathbf{x})|_n \quad (1.57a)$$

$$\|e\|_s^2 \leq C^2 l^{2(n-s)} |u_{ex}(\mathbf{x})|_n^2 \quad (1.57b)$$

where

- all of the norms are real numbers;
- c and C depend on the type of element and approximation used. C is a function of c and of V^e ;
- the polynomial basis of the approximation is complete up to the order $n-1$;
- l is linked to the maximum dimension of the element: for example, l might be the radius of the circle circumscribing a triangular element;
- all derivatives of the approximation $u(\mathbf{x})$ up to the order s are bounded;
- $|u_{ex}(\mathbf{x})|_n$ is norm (1.55) of $u_{ex}(\mathbf{x})$ with $s=n$.

To estimate error $e(\xi)$, we develop the function u_{ex} in the neighborhood of point ξ in a Taylor series. For one dimension we have:

$$u_{ex}(\xi+h) = u_{ex}(\xi) + h \frac{\partial u_{ex}}{\partial \xi} \Big|_{\xi} + \dots + \frac{h^{n-1}}{(n-1)!} \frac{\partial u_{ex}^{n-1}}{\partial \xi^{n-1}} \Big|_{\xi} + \frac{h^n}{n!} R_n \quad (1.58)$$

where

$$R_n = \frac{\partial^n u_{ex}}{\partial \xi^n} \Bigg|_{\bar{\xi} \text{ on } [\xi, \xi+h]}.$$

Assume that the element has n interpolation nodes with coordinates $\xi_1, \xi_2, \dots, \xi_n$. For a value of $h = \xi_i - \xi$, expression (1.58) takes the value u_i :

$$\begin{aligned} u_{ex}(\xi_i) &= u_i = u_{ex}(\xi) + (\xi_i - \xi) \frac{\partial u_{ex}}{\partial \xi} \Bigg|_{\xi} + \\ &\dots + \frac{(\xi_i - \xi)^{n-1}}{(n-1)!} \frac{\partial^{n-1} u_{ex}}{\partial \xi^{n-1}} \Bigg|_{\xi} + \frac{(\xi_i - \xi)^n}{n!} R_i \end{aligned} \quad (1.59)$$

where

$$R_i = \frac{\partial^n u_{ex}}{\partial \xi^n} \Bigg|_{\bar{\xi}_i \text{ on } [\xi_i, \xi]}.$$

By inserting the previous expression of u_i corresponding to each node into $\{u_n\}$, the approximate function:

$$u(\xi) = \langle N(\xi) \rangle \{u_n\}$$

is written as:

$$\begin{aligned} u(\xi) &= \left(\sum_i N_i \right) u_{ex}(\xi) + \left(\sum_i N_i \cdot (\xi_i - \xi) \right) \frac{\partial u_{ex}}{\partial \xi} \Bigg|_{\xi} + \\ &\dots + \frac{1}{(n-1)!} \left(\sum_i N_i \cdot (\xi_i - \xi)^{n-1} \right) \frac{\partial^{n-1} u_{ex}}{\partial \xi^{n-1}} \Bigg|_{\xi} + \frac{1}{n!} \sum_i N_i \cdot (\xi_i - \xi)^n \cdot R_i. \end{aligned} \quad (1.60)$$

This relationship links the value $u(\xi)$ to the exact value $u_{ex}(\xi)$ at point ξ in the reference element. If the approximation $u(\xi)$ includes the monomials $1, \xi, \xi^2, \dots, \xi^{n-1}$, we know, from (1.31), that:

$$\begin{aligned} \sum_i N_i &= 1 \\ \sum_i N_i \cdot (\xi_i - \xi) &= \sum_i N_i \xi_i - \sum_i N_i \xi = 0 \\ \dots & \\ \sum_i N_i \cdot (\xi_i - \xi)^{n-1} &= 0 \end{aligned} \quad (1.61)$$

$$\text{Hence} \quad e(\xi) = u(\xi) - u_{ex}(\xi) = \frac{1}{n!} \sum_i N_i \cdot (\xi_i - \xi)^n R_i. \quad (1.62a)$$

The last expression involves the derivatives in ξ of order n of the function u_{ex} . These derivatives are linked to the derivatives in x by relation (1.36a):

$$\frac{\partial}{\partial \xi} = \frac{\partial x}{\partial \xi} \cdot \frac{\partial}{\partial x}.$$

For a linear transformation, τ :

$$\frac{\partial x}{\partial \xi} = \text{constant} = \frac{l}{2};$$

l being the length of the element. Thus

$$\begin{aligned} \frac{\partial^n}{\partial \xi^n} &= \left(\frac{l}{2}\right)^n \frac{\partial^n}{\partial x^n} \\ R_i &= \left(\frac{l}{2}\right)^n \frac{\partial^n u_{ex}}{\partial x^n} \Big|_{x(\bar{\xi}_i)}. \end{aligned}$$

Expression (1.62a) becomes:

$$e(x) = \frac{1}{n!} \left(\frac{l}{2}\right)^n \sum_i N_i \cdot (\xi_i - \xi)^n \frac{\partial^n u_{ex}}{\partial x^n} \Big|_{x(\bar{\xi}_i)}. \quad (1.62b)$$

This error tends toward zero when l tends toward zero, provided $n \geq 1$. We are also able to construct the errors for the derivatives u_{ex} in the same way.

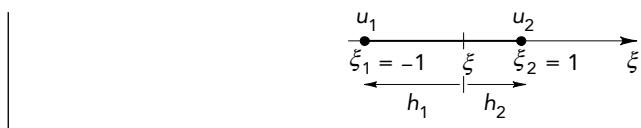
The norm of (1.62b) can be written in its generalized form (1.57a) for $s = 0$:

$$|e|_0 \leq c l^n |u_{ex}(x)|_n.$$

Remark

- The expression of $e(x)$ is obtained using $u_i = u_{ex}(x_i)$. In the context of a formulation using finite elements, this estimation always remains valid even if $u_i \neq u_{ex}(x_i)$ (see [STR 73]).

EXAMPLE 1.21. Error for a one-dimensional linear element



$$u_{ex}(\xi = -1) = u_1 = u_{ex}(\xi) + h_1 \frac{\partial u_{ex}}{\partial \xi} \Big|_{\xi} + \frac{h_1^2}{2} R_1$$

where

$$h_1 = -1 - \xi \quad R_1 = \frac{\partial^2 u_{ex}}{\partial \xi^2} \Big|_{\bar{\xi} \text{ on } [-1, \xi]}$$

$$u_{ex}(\xi = 1) = u_2 = u_{ex}(\xi) + h_2 \frac{\partial u_{ex}}{\partial \xi} \Big|_{\xi} + \frac{h_2^2}{2} R_2$$

where

$$h_2 = 1 - \xi \quad R_2 = \frac{\partial^2 u_{ex}}{\partial \xi^2} \Big|_{\bar{\xi} \text{ on } [\xi, 1]}$$

The approximate function over the element V^r is written as:

$$u(\xi) = \langle N_1(\xi) \quad N_2(\xi) \rangle \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \frac{1-\xi}{2} u_1 + \frac{1+\xi}{2} u_2$$

or, using the previous expressions of u_1 and u_2 :

$$u(\xi) = u_{ex}(\xi) + \frac{1}{4} (1 - \xi^2) ((1 + \xi) R_1 + (1 - \xi) R_2).$$

The error is:

$$e(\xi) = u(\xi) - u_{ex}(\xi) = \frac{1}{4} (1 - \xi^2) ((R_1 + R_2) + \xi(R_1 - R_2))$$

and the norm of the maximum verifies:

$$\|e\|_0 \leq \frac{1}{2} \operatorname{Max} \left| \frac{\partial^2 u_{ex}}{\partial \xi^2} \right|_{V^r},$$

where R_1 and R_2 are upper bounded by $\operatorname{Max} \left| \frac{\partial^2 u_{ex}}{\partial \xi^2} \right|_{V^r}$.

Since $(1 - \xi^2) \leq 1$ for all ξ on V^r :

$$R_1 + R_2 + \xi(R_1 - R_2) \leq 2 \operatorname{Max} \left| \frac{\partial^2 u_{ex}}{\partial \xi^2} \right|_{V^r}.$$

To express this error as a function of the derivatives with respect to x for the real element, we use the following geometrical transformation:

$$x = \left\langle \frac{1-\xi}{2} \quad \frac{1+\xi}{2} \right\rangle \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix}.$$

Hence

$$\frac{\partial u}{\partial \xi} = \frac{\partial x}{\partial \xi} \frac{\partial u}{\partial x} = \frac{x_2 - x_1}{2} \frac{\partial u}{\partial x} = \frac{l}{2} \frac{\partial u}{\partial x}$$

$$\frac{\partial^2 u}{\partial \xi^2} = \frac{l^2}{4} \frac{\partial^2 u}{\partial x^2}$$

Therefore

$$|e|_0 \leq \frac{l^2}{8} \operatorname{Max} \left| \frac{\partial^2 u_{ex}}{\partial x^2} \right|_{V^e}.$$

This expression is of the form (1.57a) with:

$$c = \frac{1}{8}, \quad n = 2, \quad s = 0.$$

We can demonstrate in a similar manner that the error in the derivative is:

$$e_1 = \frac{\partial u}{\partial x} - \frac{\partial u_{ex}}{\partial x}$$

$$= \frac{2}{l} \left(\frac{\partial u}{\partial \xi} - \frac{\partial u_{ex}}{\partial \xi} \right)$$

$$|e|_1 \leq \frac{l}{2} \operatorname{Max} \left| \frac{\partial^2 u_{ex}}{\partial x^2} \right|_{V^e}.$$

This expression is of the form (1.57a) with:

$$c = \frac{1}{2}, \quad n = 2, \quad s = 1$$

1.7.2 ERROR EVALUATION TECHNIQUE

We describe here a systematic technique for estimating the error in a one-dimensional finite element approximation, which may be generalized for two or three dimensions, as we will demonstrate in section 2.3.2.

Let us develop the function $u_{ex}(\xi)$ in a Taylor series in the neighborhood of the point $\xi = 0$:

$$u_{ex}(\xi) = u_{ex}(0) + \xi \frac{\partial u_{ex}}{\partial \xi} \Big|_0 + \dots + \frac{\xi^{n-1}}{(n-1)!} \frac{\partial^{n-1} u_{ex}}{\partial \xi^{n-1}} \Big|_0 + \frac{\xi^n}{n!} R \quad (1.63)$$

$$R = \frac{\partial^n u_{ex}}{\partial \xi^n} \Big|_{\xi \text{ on } [0, \xi]}$$

$$u_{\text{ex}}(\xi) = \langle 1 \quad \xi \quad \xi^2 \quad \dots \quad \xi^{n-1} \rangle \left\{ \begin{array}{c} u_{\text{ex}} \\ \frac{\partial u_{\text{ex}}}{\partial \xi} \\ \vdots \\ \frac{1}{(n-1)!} \frac{\partial^{n-1} u_{\text{ex}}}{\partial \xi^{n-1}} \end{array} \right\}_{\xi=0} + \xi^n \frac{1}{n!} R \quad (1.64a)$$

which, in vectorial notation, is written as:

$$u_{\text{ex}}(\xi) = \langle P \rangle \{ \partial u_{\text{ex}} \} + \xi^n \frac{1}{n!} R. \quad (1.64b)$$

The finite element approximate function is written with the help of (1.18):

$$u(\xi) = \langle P \rangle \{ a \}. \quad (1.65)$$

The values of u and u_{ex} are the same at nodes n :

$$\{u_n\} = [P_n] \{a\} = [P_n] \{\partial u_{\text{ex}}\} + \frac{1}{n!} \left\{ \begin{array}{c} \xi_1^n R_1 \\ \xi_2^n R_2 \\ \dots \\ \xi_n^n R_n \end{array} \right\} \quad (1.66)$$

$$R_i = \left. \frac{\partial^n u_{\text{ex}}}{\partial \xi^n} \right|_{\bar{\xi} \text{ on } [0, \xi_i]}.$$

$$\text{Hence } \{\partial u_{\text{ex}}\} = \{a\} - \frac{1}{n!} [P_n]^{-1} \left\{ \begin{array}{c} \xi_1^n R_1 \\ \xi_2^n R_2 \\ \dots \\ \xi_n^n R_n \end{array} \right\} \quad (1.67)$$

Substituting (1.67) into (1.64b) and using (1.65) and (1.24):

$$e(\xi) = u(\xi) - u_{\text{ex}}(\xi) = \frac{1}{n!} \langle N \rangle \left\{ \begin{array}{c} \xi_1^n R_1 \\ \xi_2^n R_2 \\ \dots \\ \xi_n^n R_n \end{array} \right\} - \frac{1}{n!} \xi^n R. \quad (1.68)$$

The maximum norm of the error $e(\xi) = u(\xi) - u_{\text{ex}}(\xi)$ is written by upper bounding R_1, R_2, \dots, R_n by $R = \text{Max} \left| \frac{\partial^n u_{\text{ex}}}{\partial \xi^n} \right|_{V^r}$:

$$|e|_0 \leq \frac{1}{n!} \operatorname{Max} \left| \langle N \rangle \begin{Bmatrix} \xi_1^n \\ \xi_2^n \\ \vdots \\ \xi_n^n \end{Bmatrix} - \xi^n \right|_{V'} \cdot \operatorname{Max} \left| \frac{\partial^n u_{ex}}{\partial \xi^n} \right|_{V'} ; \quad (1.69)$$

To bring in the geometry of the real element (shape and dimension), the derivatives in terms of ξ must be replaced by derivatives in terms of x , using the results of section 1.5.

The error in the derivatives is obtained by differentiating (1.68). For example, for the first derivative:

$$e_1(\xi) = \frac{\partial u}{\partial x} - \frac{\partial u_{ex}}{\partial x} = \left(\frac{\partial u}{\partial \xi} - \frac{\partial u_{ex}}{\partial \xi} \right) \frac{\partial \xi}{\partial x} \quad (1.70)$$

$$e_1(\xi) = \left(\frac{1}{n!} \langle \frac{\partial N}{\partial \xi} \rangle \begin{Bmatrix} \xi_1^n R_1 \\ \xi_2^n R_2 \\ \dots \\ \xi_n^n R_n \end{Bmatrix} - \frac{1}{(n-1)!} \xi^{n-1} R \right) \frac{\partial \xi}{\partial x}. \quad (1.71)$$

EXAMPLE 1.22. Error for a one-dimensional linear element (general technique)

$$\langle N \rangle = \left\langle \frac{1-\xi}{2} \frac{1+\xi}{2} \right\rangle \quad n=2.$$

Let us use (1.69):

$$\begin{aligned} |e|_0 &\leq \frac{1}{2} \operatorname{Max} \left| \left\langle \frac{1-\xi}{2} \frac{1+\xi}{2} \right\rangle \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} - \xi^2 \right| \cdot \operatorname{Max} \left| \frac{\partial^2 u_{ex}}{\partial \xi^2} \right| \\ &\leq \frac{1}{2} \operatorname{Max} |\xi^2 - 1| \cdot \operatorname{Max} \left| \frac{\partial^2 u_{ex}}{\partial \xi^2} \right| \\ &\leq \frac{l^2}{8} \operatorname{Max} \left| \frac{\partial^2 u_{ex}}{\partial x^2} \right| \\ |e|_1 &\leq \frac{1}{2} \operatorname{Max} \left| \left\langle -\frac{1}{2} \frac{1}{2} \right\rangle \begin{Bmatrix} 1 \\ 1 \end{Bmatrix} - 2\xi \right| \cdot \operatorname{Max} \left| \frac{\partial^2 u_{ex}}{\partial \xi^2} \right| \cdot \operatorname{Max} \left| \frac{\partial \xi}{\partial x} \right| \\ &\leq \frac{l}{2} \operatorname{Max} \left| \frac{\partial^2 u_{ex}}{\partial x^2} \right| \end{aligned}$$

This expression is of the form (1.57a) with:

$$c = \frac{1}{2}, \quad n = 2, \quad s = 1.$$

EXAMPLE 1.23. Simple error estimation technique

For the linear element in Example 1.22, we have:

$$u_{\text{ex}}(\xi) = u(0) + \frac{\partial u}{\partial \xi}(0)\xi + \frac{\xi^2}{2} \left(\frac{\partial^2 u}{\partial \xi^2} \right)_{-1 \leq \xi \leq 1}.$$

The linear approximation is:

$$u(\xi) = \frac{1-\xi}{2} u_1 + \frac{1+\xi}{2} u_2 = \frac{u_1 + u_2}{2} + \xi \left(\frac{u_1 - u_2}{2} \right).$$

The approximation error is then written as:

$$\begin{aligned} u_{\text{ex}}(\xi) &= u(\xi) + \frac{\xi^2}{2} \left(\frac{\partial^2 u}{\partial \xi^2} \right)_{-1 \leq \xi \leq 1} \\ |e|_0 &= |u(\xi) - u_{\text{ex}}(\xi)| \leq \frac{1}{2} \text{Max} \left(\frac{\partial^2 u}{\partial \xi^2} \right) = \frac{l^2}{8} \text{Max} \left(\frac{\partial^2 u}{\partial x^2} \right)_{x_1 \leq x \leq x_2}. \end{aligned}$$

This error is identical to that found in Example 1.22.

For a quadratic approximation (see 2.2.2.1):

$$\begin{aligned} u_{\text{ex}}(\xi) &= u(\xi) + \frac{\xi^3}{6} \left(\frac{\partial^3 u}{\partial \xi^3} \right)_{-1 \leq \xi \leq 1} \\ |e|_0 &= |u(\xi) - u_{\text{ex}}(\xi)| \leq \frac{1}{6} \text{Max} \left(\frac{\partial^3 u}{\partial \xi^3} \right) = \frac{l^3}{48} \text{Max} \left(\frac{\partial^3 u}{\partial x^3} \right)_{x_1 \leq x \leq x_2}. \end{aligned}$$

This estimation technique gives a value greater than that obtained using (1.62b):

$$|e|_0 \leq \frac{l^3}{72\sqrt{3}} \text{Max} \left(\frac{\partial^3 u}{\partial x^3} \right), \quad (\text{see 2.2.c}).$$

1.7.3 IMPROVING THE PRECISION OF APPROXIMATION

To ensure the convergence of the approximation of derivatives up to the order s , the polynomial basis must be complete up to the order $(n-1)$ and it must also verify the consistency condition $n > s$.

To improve the precision of an approximation, we must reduce the errors defined in (1.57). To do this, we must either:

- decrease l , i.e. the dimension of each element, or
- increase n , i.e. use an approximation with a polynomial basis that is complete up to a higher order.

We can therefore use several techniques:

- a) Decrease the size of each element, with a consequent increase in the number of elements needed to represent the whole domain V (h -technique).
- b) Increase the order of the approximation polynomial (p -technique). This may be done:
 - by increasing the number of interpolation nodes per element, whilst preserving a nodal variable u_i for each node, leading to a Lagrangian-type family of elements (see section 2.2.2);
 - by increasing the number of nodal variables per node, while preserving the number of nodes. The additional nodal variables are the values of the derivatives $\frac{\partial u_{ex}}{\partial \mathbf{x}}, \frac{\partial^2 u_{ex}}{\partial \mathbf{x}^2}, \dots$, at the nodes, which lead to a Hermite-type family of elements (see section 2.2.3);
 - by combining the two methods described above (see 2.2.4.1);
 - by adding additional null interpolation functions $P_l(\xi)$ at all the interpolation nodes and on the boundaries:

$$u(\xi) = \langle N(\xi) \rangle \{u_n\} + \langle P_l(\xi) \rangle \{a_I\} \quad (1.72)$$

where $P_l(\xi_i) = 0$; $P_l(\xi) = 0$ if ξ is on the boundary of V^e .

This expression is in effect a combination of nodal approximation and non-nodal approximation for each element (see section 2.2.4.2).

EXAMPLE 1.24. Approximation error of $u_{ex}(x)$

Nodal collocation (single element):

Carrying out an approximation by nodal collocation of $u_{ex}(x)$:

$$u_{ex}(x) = x^n, \quad 0 \leq x \leq h,$$

we have, for a single element of length h :

$$\begin{aligned} u_1 &= 0, \quad u_2 = (h)^n \\ u(x) &= \frac{x}{h}(h)^n, \quad u_{ex} = x^n \\ e(x) &= u(x) - u_{ex}(x) = \left(\frac{x}{h}(h)^n - x^n \right) \end{aligned} \quad (A)$$

Using Example 1.22:

$$|e|_0 \leq \frac{h^2}{8} \operatorname{Max} \left(\frac{\partial^2 u_{ex}}{\partial x^2} \right) = \frac{h^2}{8} n(n-1)h^{n-2}. \quad (B)$$

Using relationship (A), the L^2 norm of the error is:

$$\|e\| = \left(\int_0^h (h^{n-1}x - x^n)^2 dx \right)^{1/2} = \left(\frac{1}{3} + \frac{1}{1+2n} - \frac{2}{2+n} \right)^{1/2} h^n \sqrt{h}.$$

For $h = 1$, $n = 2$, $\|e\| = \frac{1}{\sqrt{30}}$ and $|e|_0 = \frac{1}{4}$.

Nodal collocation (two elements):

For the case where $u_{ex}(x) = x^2$.

Element 1:

$$\begin{aligned} u^{(1)}(x) &= \frac{x}{h}(h)^2, \quad u_{ex}(x) = x^2; \quad 0 \leq x \leq h = 1/2, \\ e^{(1)}(x) &= \left(\frac{x}{h}h^2 - x^2 \right), \quad |e^{(1)}|_0 = \frac{h^2}{4} = 1/16. \end{aligned}$$

Element 2:

$$\begin{aligned} u^{(2)}(x) &= \left(1 - \frac{x}{h} \right)(h)^2 + \frac{x}{h}(2h^2), \quad u_{ex}(x) = (x+h)^2; \quad 0 \leq x \leq h = 1/2, \\ e^{(2)}(x) &= (xh - x^2), \quad |e^{(2)}|_0 = \frac{h^2}{4} = 1/16. \end{aligned}$$

Note that where $n=2$, the error is the same for each element. If, on the other hand, we take $u_{ex}(x) = x^n$, $n > 2$, the error then varies from one element to another.

Least squares method (one element):

Calculating u_i using the least squares method comes down to writing:

$$W = \int_0^h \delta u(u(x) - u_{ex}(x)) dx = 0$$

where $u(x) = \left(1 - \frac{x}{h}\right)u_1 + \frac{x}{h}u_2$ and $\delta u(x) = \left(1 - \frac{x}{h}\right)\delta u_1 + \frac{x}{h}\delta u_2$.

We therefore need to solve: $\frac{h}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \begin{Bmatrix} \int_0^h \left(1 - \frac{x}{h}\right)x^n dx \\ \int_0^h \frac{x}{h}x^n dx \end{Bmatrix}$.

Where $n=2$, we have:

$$\begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \begin{bmatrix} 4 & -2 \\ -2 & 4 \end{bmatrix} \begin{Bmatrix} \frac{1}{12} \\ \frac{1}{4} \end{Bmatrix} h^2 = \begin{Bmatrix} -\frac{1}{6} \\ \frac{5}{6} \end{Bmatrix} h^2$$

$$e(x) = \left(-\left(1 - \frac{x}{h}\right)\frac{h^2}{6} + \frac{5}{6}\frac{x}{h}h^2 - x^2\right); \quad 0 \leq x \leq h$$

$$\text{Max}|e(x)| = \frac{h^2}{12}.$$

However, following equation (B), we have: $\text{Max}|e(x)| = \frac{h^2}{4}$.

Remarks

- The error norm obtained using the least squares method is lower than that obtained by the collocation method.
- The least squares method results in a negative value for u_1 .

Least squares method (two elements):

We have: $u_{ex}(x) = x^2$.

Element 1:

$$u^{(1)}(x) = \frac{x}{h}(h)^2, \quad u_{ex}(x) = x^2, \quad 0 \leq x \leq h = 1/2,$$

$$e^{(1)}(x) = \left(\frac{x}{h}h^2 - x^2\right), \quad |e^{(1)}|_0 = \frac{h^2}{4} = 1/16.$$

Element 2:

$$u^{(2)}(x) = \left(1 - \frac{x}{h}\right)(h)^2 + \frac{x}{h}(2h^2), \quad u_{ex}(x) = (x+h)^2, \quad 0 \leq x \leq h = 1/2,$$

$$e^{(2)}(x) = (xh - x^2), \quad |e^{(2)}|_0 = \frac{h^2}{4} = 1/16.$$

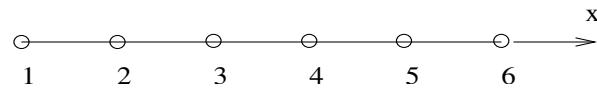
Remark

- The error is the same for each element. If we choose $u_{ex}(x) = x^n, n > 2$, instead, the error would vary from one element to another.

EXAMPLE 1.25. Adaptation of a one-dimensional mesh by displacement of nodes (r-technique)

The principle of this method is based on the reduction of the size of the elements to reduce the error while conserving the same number of elements (by displacing nodes).

Let us take the estimator used in Example 1.23 to modify the position of nodes and obtain a uniform distribution of the error across each of the elements. We will use $u_{ex}(x) = x^3$ and a uniform mesh made up of five elements.



The initial position of the nodes is given by the vector:

$$\langle x_n \rangle = \langle 0 \ 0.2 \ 0.4 \ 0.6 \ 0.8 \ 1 \rangle.$$

The error $|e_0| \sqrt{h}$ for the initial mesh, using a nodal collocation technique, is calculated using:

$$|e_0| \sqrt{h} = \frac{h^{5/2}}{8} \text{Max} \left| \frac{d^2 u}{dx^2} \right| = \frac{3}{4} h^{5/2} \text{Max}(x), \quad h = 0.2.$$

The error for each element is thus:

$$E = |e_0| \sqrt{h} = \langle 0.016 \ 0.0322 \ 0.0483 \ 0.0644 \ 0.0805 \rangle, \quad \sum_{i=1}^5 E_i = 0.2415.$$

The size of the elements is then modified as a function of the error, while preserving the same number of elements. The length of each element is decreased or increased

depending on whether the associated error is large or small. The size of the element is thus modified in the following manner:

$$h^{\text{modif}} = \langle h_i / E_i^{0.4} \rangle \times L / \sum_{i=1}^5 (h_i / E_i^{0.4}), \quad L: \text{total length.}$$

After the first iteration, the nodal coordinates are:

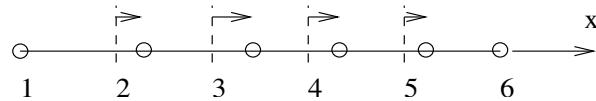
$$\langle x_n \rangle = \langle 0 \ 0.2856 \ 0.502 \ 0.686 \ 0.85 \ 1 \rangle,$$

and the error is:

$$E = |e_0| \sqrt{h} = \langle 0.056 \ 0.0492 \ 0.0448 \ 0.0417 \ 0.0392 \rangle, \quad \sum_{i=1}^5 E_i = 0.2309.$$

After four cycles, the nodal coordinates are:

$$\langle x_n \rangle = \langle 0 \ 0.2699 \ 0.4834 \ 0.6704 \ 0.841 \ 1 \rangle,$$



giving elementary sizes:

$$\langle h \rangle = \langle 0.2699 \ 0.2135 \ 0.1870 \ 0.1706 \ 0.159 \rangle,$$

The error for each element at this point is:

$$E = |e_0| \sqrt{h} = \langle 0.046 \ 0.0458 \ 0.0456 \ 0.0455 \ 0.0454 \rangle, \quad \sum_{i=1}^5 E_i = 0.2283.$$

Figure 1.3 is the Matlab[©] program for this example.

```
% Mesh adaptation using r-technique
% Based on estimator from example 1.24: Eo=h*h/8*Uxx
%----- geometrical data
L=1.; % total length
nelt=5; % number of elements
n=3; % degree of exact solution function
ncycle=20; % number of iterative cycles
h=L/nelt; % uniform element length
vx=[0:1:nelt].*h; % nodal coordinates
vh=ones(1,nelt).*h; % elementary sizes
%----- Adaptation loop
for ic=1:ncycle
```

```

vu=vx.^n; % nodal solution by collocation
ve0=(n*(n-1))/8*vh.^2.*vx(2:nelt+1).^(n-2);% error
ve0h=abs(ve0.*vh.^0.5);
%----- convergence test
emax=max(ve0h);emin=min(ve0h);
if((emax-emin)/emax<0.05) break; end
%-----mesh adaptation
v1=(1./ve0h);
vh=vh.*^(v1.^0.4);
vx=[0,v1]*L;
for j=1:nelt
    vx(j+1)=vx(j)+vx(j+1);% new coordinate
end
vh=vx(2:nelt+1)-vx(1:nelt);% new lengths
end

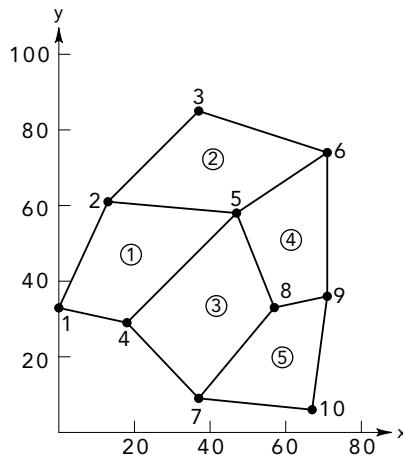
```

Figure 1.3. Matlab[®] program for mesh adaptation

1.8 Example of application: rainfall problem

The finite element method of approximation is most often used for the discretization of partial differential equations. It may, however, be used to approximate a function known only at certain points of measurement.

In this section, we calculate the total rainfall for a region A using the measurements obtained by rain gages situated at certain points:



The total rainfall Q is defined from the rainfall $u(x, y)$ at all points by:

$$Q = \int_A u(x, y) dA. \quad (1.73)$$

The rainfall u_i is known for 10 stations of coordinates x_i, y_i . The data used in this example are taken from an article by Akin [AKI 71]. We will use a finite element approximation technique to evaluate (1.73).

a) Choice of nodes and elements

We select stations 1, 2, ..., 10 as geometrical nodes and interpolation nodes. Their coordinates make up the CORG table:

Node	x_i (km)	y_i (km)
1	0.0	33.3
2	13.2	62.3
3	39.3	84.5
4	22.2	30.1
5	49.9	57.6
6	78.8	78.2
7	39.3	10.0
8	59.7	34.3
9	73.9	36.2
10	69.8	5.1

Region A is divided into five quadrilateral elements defined by the CONEC table:

Element	Nodes			
	i	j	k	l
1	1	4	5	2
2	2	5	6	3
3	4	7	8	5
4	5	8	9	6
5	7	10	9	8

The rainfall vector at the nodes (nodal values of u) is known in this problem (units: depth of rainfall in cm):

$$\{u_n\}^T = \langle 4.62 \ 3.81 \ 4.76 \ 5.45 \ 4.90 \ 10.35 \ 4.96 \ 4.26 \ 18.36 \ 15.69 \rangle$$

b) Approximation of $u(x, y)$ over each element

Choose a bilinear quadrilateral element as described in Example 1.16. For each element e , the approximate function $u(\xi, \eta)$ is:

$$u(\xi, \eta) = \langle P \rangle [P_n]^{-1} \{u_n\} \quad (1.74)$$

The geometrical transformation is:

$$\begin{aligned} x(\xi, \eta) &= \langle P \rangle [P_n]^{-1} \{x_n\} \\ y(\xi, \eta) &= \langle P \rangle [P_n]^{-1} \{y_n\} \end{aligned}$$

where

$$\begin{aligned} \{u_n\}^T &= \langle u_i \quad u_j \quad u_k \quad u_l \rangle \\ \{x_n\}^T &= \langle x_i \quad x_j \quad x_k \quad x_l \rangle \\ \{y_n\}^T &= \langle y_i \quad y_j \quad y_k \quad y_l \rangle \end{aligned}$$

i, j, k and l being the numbers of the four nodes of the element, given by the CONEC table.

The determinant of the Jacobian matrix $\det(J)$ has already been evaluated in Example 1.18 in the form:

$$\det(J) = A_0 + A_1 \xi + A_2 \eta. \quad (1.75)$$

c) Evaluation of Q

The total rainfall Q is the sum of the rainfall Q^e over each element:

$$\begin{aligned} Q &= \sum_{e=1}^5 Q^e & (1.76) \\ Q^e &= \int_{A^e} u(x, y) dA \\ &= \int_{-1}^1 \int_{-1}^1 u(\xi, \eta) \det(J) d\xi d\eta. \end{aligned}$$

Hence, after replacing u by approximation (1.74):

$$\begin{aligned} Q^e &= \int_{-1}^1 \int_{-1}^1 \langle P \rangle [P_n]^{-1} \{u_n\} \det(J) d\xi d\eta \\ Q^e &= \int_{-1}^1 \int_{-1}^1 (A_0 + A_1 \xi + A_2 \eta) \langle 1 \ \xi \ \eta \ \xi\eta \rangle d\xi d\eta \cdot [P_n]^{-1} \{u_n\}. \end{aligned} \quad (1.77)$$

After explicit integration, we organize Q^e in the form:

$$Q^e = \langle A_0 \ \frac{A_1}{3} \ \frac{A_2}{3} \rangle \begin{Bmatrix} u_i + u_j + u_k + u_l \\ -u_i + u_j + u_k - u_l \\ -u_i - u_j + u_k + u_l \end{Bmatrix} \quad (1.78)$$

where coefficients A_0 , A_1 and A_2 are functions of the coordinates of the nodes, given in Example 1.18, and u_i, u_j, u_k, u_l are the values of rainfall at the four nodes of the element, extracted from $\{U_n\}$.

Finally, the table below shows the numerical calculation of (1.76) that uses (1.78), Example 1.18 and the CORG, CONEC and $\{U_n\}$ tables.

Element	A_0 (km ²)	A_1	A_2	Q^e (cm km ²)
1	228.18	1.64	55.04	4,261.41
2	241.65	-5.70	12.99	5,771.07
3	217.56	-25.18	-14.01	4,272.97
4	182.79	-65.72	29.70	6,954.45
5	159.37	15.94	-66.85	6,983.87

The total rainfall is $Q = \sum Q^e = 28,243.78$ cm km².

The approximate total area is $A = 4 \sum A_0 = 4,118.21$ km².

The average height of rainfall is $u_m = \frac{Q}{A} = 6.86$ cm.

Important results

Nodal approximation of a function:

$$u(x) = \langle N \rangle \{u_n\}. \quad (1.9)$$

Transformation of the reference element into a real element:

$$\tau: \xi \rightarrow x(\xi) = [\bar{N}(\xi)] \{x_n\}. \quad (1.12)$$

Approximation of u on the reference element:

$$u(\xi) = \langle N(\xi) \rangle \{u_n\}. \quad (1.14)$$

Properties of the interpolation functions:

$$N_j(\xi_i) = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \quad (1.16)$$

$$\sum_{i=1}^{n_d} N_i(\xi) p(\xi_i) = p(\xi). \quad (1.31)$$

Construction of interpolation functions:

$$u(\xi) = \langle P(\xi) \rangle \{a\} \quad (1.18)$$

$$\{u_n\} = [P_n] \{a\} \quad (1.20)$$

$$\langle N(\xi) \rangle = \langle P(\xi) \rangle [P_n]^{-1}. \quad (1.24)$$

Transformation of first derivatives:

$$\{\partial_x\} = [j] \{\partial_\xi\} \quad (1.37b)$$

$$[j] = [J]^{-1} \quad (1.38)$$

$$[J] = \begin{bmatrix} \langle \bar{N}_{,\xi} \rangle \\ \langle \bar{N}_{,\eta} \rangle \\ \langle \bar{N}_{,\zeta} \rangle \end{bmatrix} \begin{bmatrix} \{x_n\} & \{y_n\} & \{z_n\} \end{bmatrix}. \quad (1.43)$$

Transformation of an integral:

$$\int_{V^e} f(\mathbf{x}) dx dy dz = \int_{V^e} f(\mathbf{x}(\xi)) \det(J) d\xi d\eta d\zeta. \quad (1.44)$$

Approximation errors:

$$|e|_s \leq c l^{n-s} |u_{ex}(\mathbf{x})|_n. \quad (1.57a)$$

Notations

$\langle a \rangle = \langle a_1 \ a_2 \ \dots \ a_n \rangle$	general parameters of an approximation (or generalized variables)
$\langle a_x \rangle, \langle a_y \rangle, \langle a_z \rangle$	generalized coordinates of the element
$e(x) = u(x) - u_{ex}(x)$	approximation error
$[J], [j], \det(J)$	Jacobian matrix, its inverse and its determinant
n	number of interpolation nodes
n_d	number of degrees of freedom of an element
n^e	number of interpolation nodes of an element

n_{el}	number of elements
\bar{n}	number of geometrical nodes
\bar{n}^e	number of geometrical nodes of an element
$\langle N(x) \rangle = \langle N_1(x) \ N_2(x) \ \dots \rangle$	nodal interpolation functions over the real element
$\langle N(\xi) \rangle = \langle N_1(\xi) \ N_2(\xi) \ \dots \rangle$	interpolation functions over the reference element
$\langle \bar{N}(\xi) \rangle = \langle \bar{N}_1(\xi) \ \bar{N}_2(\xi) \ \dots \rangle$ [P_n], [\bar{P}_n]	geometrical transformation functions nodal interpolation and geometrical transformation matrices
$\langle P(x) \rangle = \langle P_1(x) \ P_2(x) \ \dots \rangle$	polynomial basis of the approximation over the real element
$\langle P(\xi) \rangle = \langle P_1(\xi) \ P_2(\xi) \ \dots \rangle$	polynomial basis of the approximation over the reference element
$\langle \bar{P}(\xi) \rangle$	polynomial basis of the geometrical transformation
$[T_1], [T_2], [C_1], [C_2]$	transformation matrices for second derivatives
$u(x)$	approximate functions
$u_{ex}(x)$	exact functions
$u^e(x)$ or sometimes $u(x)$	approximate function over an element
$\langle u_n \rangle = \langle u_1 \ u_2 \ \dots \rangle$	nodal parameters or nodal variables
V	domain under study
V^e	domain corresponding to the element e
V^r	reference element domain
$\mathbf{x} = \langle x \ y \ z \rangle$	Cartesian coordinates of a point
$\mathbf{x}_i = \langle x_i \ y_i \ z_i \rangle$	coordinates of node i (geometrical or interpolation)
$\langle \mathbf{x}_n \rangle$	coordinates of the nodes of a real element
$\langle \bar{\mathbf{x}}_n \rangle$	coordinates of the geometrical nodes
$\langle \partial_x \rangle, \langle \partial_\xi \rangle$	differential operators:
	$\langle \frac{\partial}{\partial x} \ \frac{\partial}{\partial y} \ \frac{\partial}{\partial z} \rangle$ and $\langle \frac{\partial}{\partial \xi} \ \frac{\partial}{\partial \eta} \ \frac{\partial}{\partial \zeta} \rangle$
$\langle \partial_x^2 \rangle, \langle \partial_\xi^2 \rangle$	second-order differential operators defined by (1.45a) and (1.45b)
$\xi = \langle \xi \ \eta \ \zeta \rangle$	coordinates of a point of the reference element

$\langle \xi_n \rangle$	coordinates of the nodes of a reference element
$\langle \bar{\xi}_n \rangle$	coordinates of the geometrical nodes of a reference element
τ^e or τ	geometrical transformation for element e
$\ \cdot \ _s$	maximum and root mean square norm for derivatives of order s of a function
CONEC	connectivity table
CORG	nodal coordinates table

Remarks

A vector a can be represented in three different ways:

- \mathbf{a}
- $\langle a \rangle$ row vector
- $\{a\}$ column vector

A matrix T , its transpose and its inverse are represented: $[T]$, $[T]^T$ and $[T]^{-1}$.

Bibliography

- [AKI 71] AKIN J.E., “Calculation of mean areal depth of precipitation”, *Journal of Hydrology*, vol. 12, pp. 363–376, 1971.
- [CIA 78] CIARLET P.G., *The Finite Element Method for Elliptic Problems*, North-Holland, 1978.
- [DEV 74] DE VUËBEKE F.B., “Variational principles and the patch test”, *International Journal for Numerical Methods in Engineering*, vol. 8, pp. 783–801, 1974.
- [HUG 87] HUGUES T.J., *The Finite Element Method, Linear, Static and Dynamic Analysis*, Prentice-Hall, 1987.
- [IMB 84] IMBERT J.F., *Analyse des structures par éléments finis*, Cepadues Editions, Toulouse, 1984.
- [IRO 72] IRONS B.M., RAZZAQUE A.A., “Experience with the patch test” in *Mathematical Foundations of the Finite Element Method*, Academic Press, pp. 55è–567, 1972.
- [KIK 86] KIKUCHI N., *Finite Element Method in Mechanics*, Cambridge University Press, 1986.
- [ODE 71] ODEN J.T., *Finite Elements of Non-Linear Continua*, McGraw-Hill, New York, 1971.
- [STR 73] STRANG G., FIX O.J., *Analysis of the Finite Element Method*, Prentice-Hall, New Jersey, 1973.
- [ZIE 00] ZIENKIEWICZ O.C., *The Finite Element Method: The Basis (Vol. 1), Solid Mechanics (Vol. 2) & Fluid Mechanics (Vol. 3)*, 5th ed., Butterworth-Heinemann, 2000.

CHAPTER 2

Various types of elements

2.0 Introduction

In the first chapter, we detailed the finite element approximation method. In particular, we discussed the notions of a reference element and an interpolation function. In this chapter, we will present the interpolation functions of the various reference elements frequently used in practice. A particular type of reference element is defined by:

- its shape (e.g. triangular);
- the coordinates $\{\xi_n\}$ of its \bar{n} geometric nodes;
- the coordinates $\{\xi_n\}$ of its n interpolation nodes;
- the number of degrees of freedom n_d ;
- the definition of its nodal variables $\{u_n\}$;
- the polynomial basis of the approximation $\langle P \rangle$;
- the type of continuity of u satisfied on the boundary of the element: (see definition in section 1.3.2).

From the information in the previous chapter, we can construct the interpolation functions $\langle N(\xi) \rangle$ and their derivatives with respect to ξ , η and ζ :

$$u = \langle N \rangle \{u_n\} \quad (1.9)$$

$$\langle N \rangle = \langle P \rangle [P_n]^{-1} \quad (1.24)$$

where $[P_n]$ is defined in [1.20]. For non-isoparametric elements, we use the same technique to construct the geometrical transformation functions $\langle \bar{N} \rangle$, which enable us to construct the Jacobian matrix using (1.43). For isoparametric elements $\langle \bar{N} \rangle \equiv \langle N \rangle$.

The interpolation functions of most of the elements in this chapter are given in the works by Batoz and Dhatt [BAT 90], Mitchell and Wait [MIT 77], and Zienkiewicz and Taylor [ZIE 00].

2.1 List of the elements presented in this chapter

A summary of the characteristics of the various elements described in this chapter is given below.

ONE-DIMENSIONAL ELEMENTS

	Degree of the polynomial basis	Continuity (see definition, section 1.3.2)	Number of nodes n	Number of degrees of freedom n_d	Section
Lagrangian elements	1	C^0	2	2	2.2.1
	2	C^0	3	3	2.2.2.1
	3	C^0	4	4	2.2.2.2
	$n - 1$	C^0	n	n	2.2.2.3
Hermite elements	3	C^1	2	4	2.2.3.1
	5	C^2	2	6	2.2.3.2
Lagrange-Hermite elements	4	C^1	3	5	2.2.4.1
Hermite element with non-nodal degree of freedom	4	C^1	2	5	2.2.4.2

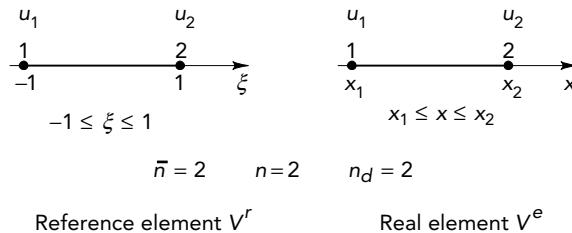
TWO-DIMENSIONAL ELEMENTS

	Degree of the polynomial basis	Continuity (see definition, section 1.3.2)	Number of nodes n	Number of degrees of freedom n_d	Section
TRIANGLES					
Lagrange	1	C^0	3	3	2.3.2
	2	C^0	6	6	2.3.3.1
	r	C^0	$\frac{(r+1)(r+2)}{2}$	$\frac{(r+1)(r+2)}{2}$	2.3.3.2
	3	C^0	10	10	2.3.3.3
	3 (incomplete)	C^0	9	9	2.3.3.4
	1	semi- C^0	3	3	2.3.3.6
Hermite	3	semi- C^1	4	10	2.3.4.1
	3 (incomplete)	semi- C^1	3	9	2.3.4.2
	5	C^1	3	18	2.3.4.3
QUADRILATERALS					
Lagrange	1	C^0	4	4	2.4.2
	2	C^0	9	9	2.4.3.1
	3	C^0	16	16	2.4.3.3
Lagrange (incomplete)	2	C^0	8	8	2.4.3.2
	3	C^0	12	12	2.4.3.4
Hermite	3	semi- C^1	4	12	2.4.4.1
Hermite (rectangle)	3	C^1	4	16	2.4.4.2

THREE-DIMENSIONAL ELEMENTS

	Degree of the polynomial basis	Continuity (see definition, section 1.3.2)	Number of nodes n	Number of degrees of freedom n_d	Section
TETRAHEDRA					
Lagrange	1	C^0	4	4	2.5.2
	2	C^0	10	10	2.5.3.1
	3	C^0	20	20	2.5.3.2
Hermite	3	semi- C^1	8	20	2.5.4
HEXAHEDRA					
Lagrange	1	C^0	8	8	2.6.1
	2	C^0	27	27	2.6.2.1
Lagrange (incomplete)	2	C^0	20	20	2.6.2.2
	3	C^0	32	32	2.6.2.3
Hermite	3	semi- C^1	8	32	2.6.3
PRISMS					
Lagrange (incomplete)	1	C^0	6	6	2.7.1
	2	C^0	15	15	2.7.2

2.2 One-dimensional elements

2.2.1 LINEAR ELEMENT (two nodes, C^0)Reference element V^r Real element V^e

The geometric nodes and the interpolation nodes 1 and 2 are identical. By convention, the nodes of the reference element and the real element are numbered from left to right.

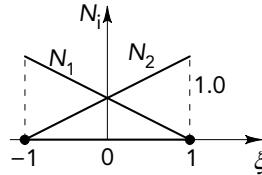
$$\langle P \rangle = \langle 1 \quad \xi \rangle \quad (2.1a)$$

$$[P_n] = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}; \quad [P_n]^{-1} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \quad (2.1b)$$

	$\frac{1}{c}\{N\}$	$\frac{1}{c}\{\partial N/\partial \xi\}$	c
1	$1-\xi$	-1	
2	$1+\xi$	1	1/2

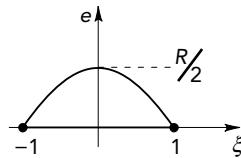
$$[J] = \frac{\partial x}{\partial \xi} = \frac{x_2 - x_1}{2} = \frac{l}{2}; \quad [j] = [J]^{-1} = \frac{2}{l}. \quad (2.1c)$$

The functions N have the following form:



The truncation error, calculated in Example 1.21, verifies that:

$$e(\xi) \leq \frac{1}{2} (1 - \xi^2) R, \quad \text{where } R = \text{Max} \left| \frac{\partial^2 u_{ex}}{\partial \xi^2} \right|_{V^e}$$



$$|e|_0 \leq \frac{l^2}{8} \text{ Max} \left| \frac{\partial^2 u_{ex}}{\partial x^2} \right|_{V^e} \quad (2.1d)$$

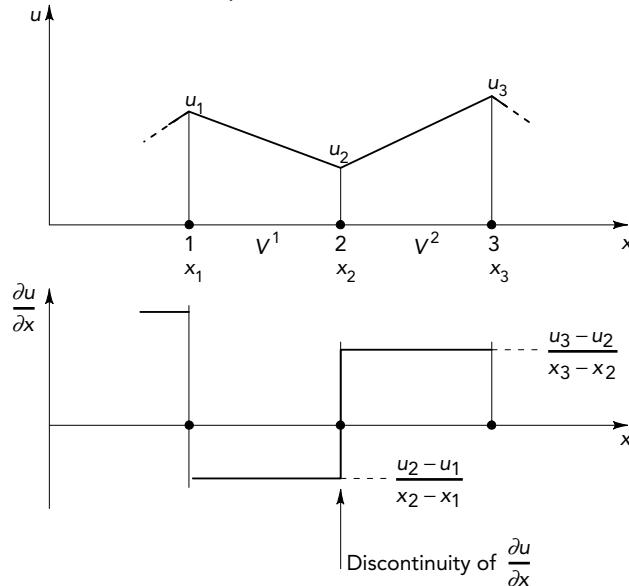
where

$$l = x_2 - x_1.$$

The error for $\partial u / \partial x$ is given by:

$$|e|_1 \leq \frac{l}{2} \operatorname{Max} \left| \frac{\partial^2 u_{ex}}{\partial x^2} \right|_{V_e}. \quad (2.1e)$$

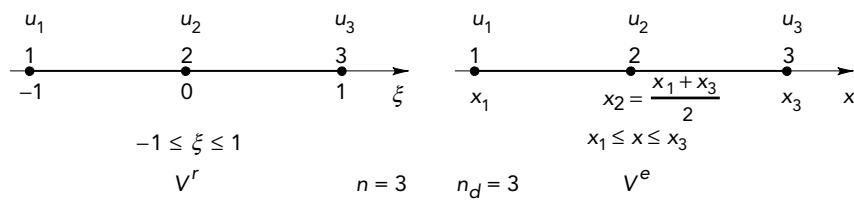
The functions u and $\partial u / \partial x$ are continuous within the element but only u is continuous across the boundary of the element:



2.2.2 HIGH-PRECISION LAGRANGIAN ELEMENTS: (continuity C^0)

This family of elements is obtained by increasing the number of interpolation nodes while keeping only one variable u_i per node. The geometric nodes, the functions \bar{N}_i , and the Jacobian matrix $[J]$ remain similar to those in section 2.2.1. Hence these elements are subparametric.

2.2.2.1 Quadratic element with equidistant nodes (three nodes, C^0)

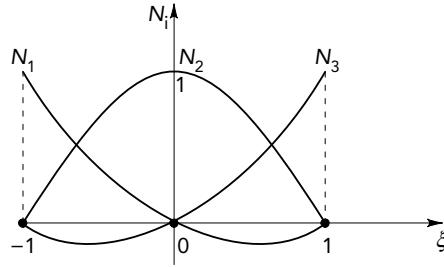


$$\langle P \rangle = \langle 1 \quad \xi \quad \xi^2 \rangle \quad (2.2a)$$

$$[P_n] = \begin{bmatrix} 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}; \quad [P_n]^{-1} = \frac{1}{2} \begin{bmatrix} 0 & 2 & 0 \\ -1 & 0 & 1 \\ 1 & -2 & 1 \end{bmatrix} \quad (2.2b)$$

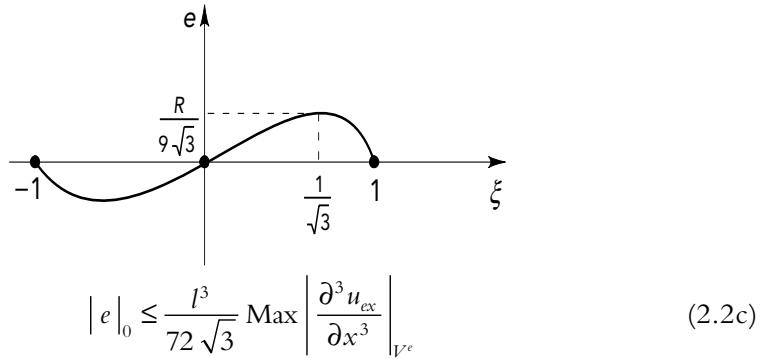
	$\frac{1}{c} \{N\}$	$\frac{1}{c} \{\partial N / \partial \xi\}$	c
1	$-\xi(1-\xi)$	$-1+2\xi$	
2	$2(1-\xi^2)$	-4ξ	$1/2$
3	$\xi(1+\xi)$	$1+2\xi$	

The functions N have the following form:



The truncation error from (1.68) verifies that:

$$e(\xi) \leq \frac{1}{6} \xi(1-\xi^2) R, \quad \text{where } R = \text{Max} \left| \frac{\partial^3 u_{ex}}{\partial \xi^3} \right|_{V^e}$$



$$|e|_1 \leq \frac{l^2}{12} \text{Max} \left| \frac{\partial^3 u_{ex}}{\partial x^3} \right|_{V^e}. \quad (2.2d)$$

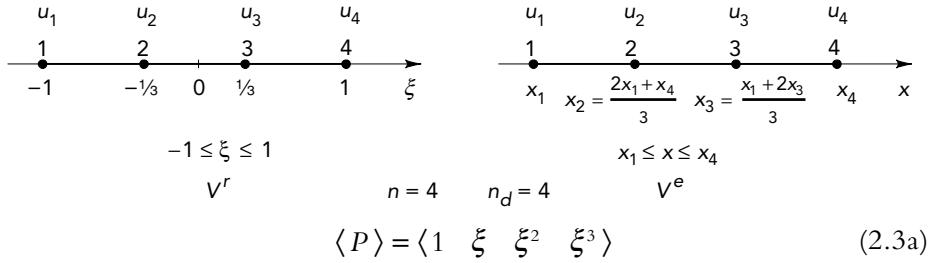
Similarly as for linear elements, u and $\partial u / \partial x$ are continuous within the element and only u is continuous across the boundary of the element.

The approximation of a quadratic element with a generalized internal variable “ a ” is (see section 2.2.4.2):

$$u(\xi) = N_1(\xi)u_1 + N_2(\xi)u_2 + P.a$$

$$N_1(\xi) = \frac{1-\xi}{2}, \quad N_2(\xi) = \frac{1+\xi}{2}, \quad P(\xi) = (1-\xi^2)$$

2.2.2.2 Cubic element with equidistant nodes (four nodes, C^0)



$$[P_n]^{-1} = \frac{1}{16} \begin{bmatrix} -1 & 9 & 9 & -1 \\ 1 & -27 & 27 & -1 \\ 9 & -9 & -9 & 9 \\ -9 & 27 & -27 & 9 \end{bmatrix} \quad (2.3b)$$

	$\frac{1}{c}\{N\}$	$\frac{1}{c}\{\partial N / \partial \xi\}$	c
1	$-(1-\xi)(1-9\xi^2)$	$1+18\xi-27\xi^2$	
2	$9(1-\xi^2)(1-3\xi)$	$-27-18\xi+81\xi^2$	
3	$9(1-\xi^2)(1+3\xi)$	$27-18\xi-81\xi^2$	$1/16$
4	$-(1+\xi)(1-9\xi^2)$	$-1+18\xi+27\xi^2$	

$$e(\xi) \leq \frac{1}{9} (-1+10\xi^2-9\xi^4) \frac{1}{24} \operatorname{Max} \left| \frac{\partial^4 u_{ex}}{\partial \xi^4} \right|_{V^r}$$

$$\left| e \right|_0 \leq \frac{l^4}{1,944} \operatorname{Max} \left| \frac{\partial^4 u_{ex}}{\partial x^4} \right|_{V^e} \quad (2.3c)$$

$$\left| e \right|_1 \leq \frac{l^3}{108} \operatorname{Max} \left| \frac{\partial^4 u_{ex}}{\partial x^4} \right|_{V^e}. \quad (2.3d)$$

2.2.2.3 General element with n nodes (n nodes, C^0)

The functions N of an element with n interpolation nodes are $(n - 1)$ -degree Lagrange polynomials. They are written as follows:

$$N_i(\xi) = \prod_{\substack{j=1 \\ j \neq i}}^n \frac{(\xi_j - \xi)}{(\xi_j - \xi_i)}. \quad (2.4a)$$

When the nodes are equidistant:

$$\xi_i = -1 + 2 \frac{i-1}{n-1}$$

$$N_i(\xi) = \prod_{\substack{j=1 \\ j \neq i}}^n \frac{(2j-n-2)-(n-1)\xi}{2(j-i)}. \quad (2.4b)$$

These functions can also be obtained using the general method in section 1.5 with:

$$\langle P \rangle = \langle 1 \quad \xi \quad \xi^2 \quad \dots \quad \xi^{n-1} \rangle \quad (2.4c)$$

$$\langle \xi_n \rangle = \langle -1; -1 + \Delta; -1 + 2\Delta; \dots; -1 + (n-1)\Delta \rangle$$

where

$$\Delta = \frac{2}{n-1}.$$

The errors are of the form (1.57a):

$$|e|_0 = C_0 l^n \operatorname{Max} \left| \frac{\partial^n u_{ex}}{\partial x^n} \right|_{V^e} \quad (2.4d)$$

$$|e|_1 = C_1 l^{n-1} \operatorname{Max} \left| \frac{\partial^n u_{ex}}{\partial x^n} \right|_{V^e}. \quad (2.4e)$$

All the elements thus far presented exhibit type C^0 continuity: if the basis $\langle P \rangle$ is of degree $n-1$, the function u and its derivatives up to the order $n-1$ are continuous within the element, and only u is continuous across the boundary of the element.

2.2.3 HIGH-PRECISION HERMITE ELEMENTS

These elements are obtained by increasing the number of nodal variables attached to each node: the variables u_i are supplemented with the values at the nodes of the derivatives of u_{ex} :

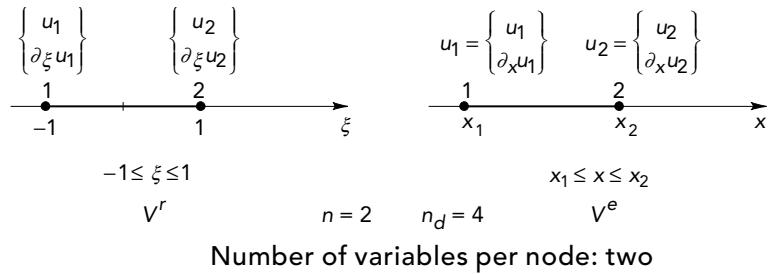
$$\left. \frac{\partial u_{ex}}{\partial x} \right|_{\text{node } i}; \quad \left. \frac{\partial^2 u_{ex}}{\partial x^2} \right|_{\text{node } i} \quad \text{etc.}$$

We will use the following notation:

$$\begin{aligned} \partial_x u_i &= \left. \frac{\partial u}{\partial x} \right|_{x=x_i} & \partial_x^2 u_i &= \left. \frac{\partial^2 u}{\partial x^2} \right|_{x=x_i} \\ \partial_\xi u_i &= \left. \frac{\partial u}{\partial \xi} \right|_{\xi=\xi_i} & \partial_\xi^2 u_i &= \left. \frac{\partial^2 u}{\partial \xi^2} \right|_{\xi=\xi_i}. \end{aligned}$$

The geometric nodes, the functions \bar{N} , and the Jacobian matrix $[J]$ remain identical to those of the linear element in section 2.2.1.

2.2.3.1 Cubic element (two nodes, C^1)



Number of variables per node: two

$$\langle P \rangle = \langle 1 \ \xi \ \xi^2 \ \xi^3 \rangle \quad (2.5a)$$

$$[P_n] = \begin{bmatrix} \langle P(\xi_1) \rangle \\ \langle \frac{\partial P}{\partial \xi}(\xi_1) \rangle \\ \hline \langle P(\xi_2) \rangle \\ \langle \frac{\partial P}{\partial \xi}(\xi_2) \rangle \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 0 & 1 & -2 & 3 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \end{bmatrix}$$

$$[P_n]^{-1} = \frac{1}{4} \begin{bmatrix} 2 & 1 & 2 & -1 \\ -3 & -1 & 3 & -1 \\ 0 & -1 & 0 & 1 \\ 1 & 1 & -1 & 1 \end{bmatrix} \quad (2.5b)$$

$$u(\xi) = \langle N \rangle \{u_n\}_\xi \quad \text{or} \quad u(\xi) = \langle N \rangle \{u_n\} \quad (2.5c)$$

where, in both cases, each function N_i differs only by one multiplication factor ([see 2.5d]).

	$\frac{1}{c} \{N\}$	$\frac{1}{c} \{\partial N / \partial \xi\}$	$\frac{c}{\text{for } \{u_n\}_\xi}$	$\frac{c}{\text{for } \{u_n\}}$
1	$(1-\xi)^2 (2+\xi)$	$-3(1-\xi^2)$		$1/4$
2	$(1-\xi^2)(1-\xi)$	$(-1+\xi)(1+3\xi)$	$1/4$	$l/8$
3	$(1+\xi)^2 (2-\xi)$	$3(1-\xi^2)$		$1/4$
4	$(-1+\xi^2)(1+\xi)$	$(-1-\xi)(1-3\xi)$		$l/8$

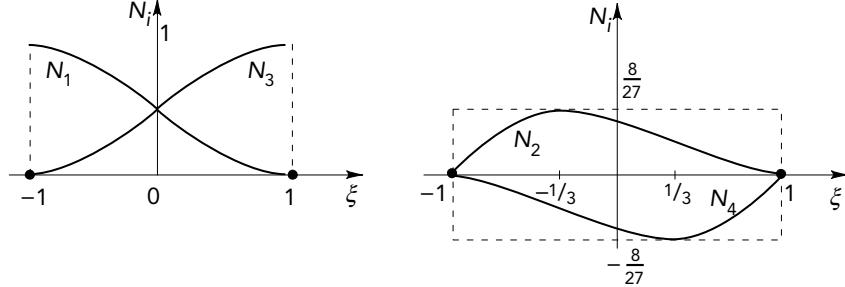
Equation (1.30) yields: $\langle N \rangle [P_n] = \langle P \rangle$.

The nodal variables on the reference element and the real element are different, owing to the differentiations with respect to ξ and x :

$$\begin{aligned} \{u_n\}_\xi &= \begin{Bmatrix} u_1 \\ \frac{\partial_\xi u_1}{u_2} \\ u_2 \\ \frac{\partial_\xi u_2}{u_2} \end{Bmatrix}; \quad \{u_n\} = \begin{Bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{Bmatrix} = \begin{Bmatrix} u_1 \\ \frac{\partial_x u_1}{u_2} \\ u_2 \\ \frac{\partial_x u_2}{u_2} \end{Bmatrix} \\ \{u_n\}_\xi &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{l}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{l}{2} \end{bmatrix} \{u_n\}; \quad l = x_2 - x_1 \end{aligned} \quad (2.5d)$$

It is the nodal variables $\{u_n\}$ containing the derivatives in terms of x that are kept as final variables for the problem.

The plots of the functions N are shown:



Using the method from section 1.7.2, we get:

$$e(\xi) \leq \frac{1}{24} (1 - \xi^2)^2 \operatorname{Max} \left| \frac{\partial^4 u_{ex}}{\partial \xi^4} \right|_{V^r} \quad (2.5e)$$

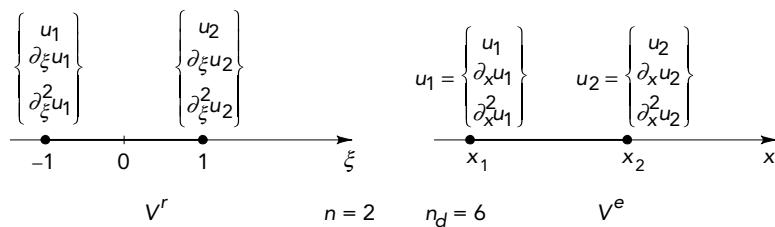
$$\| e \|_0 \leq \frac{l^4}{384} \operatorname{Max} \left| \frac{\partial^4 u_{ex}}{\partial x^4} \right|_{V^e} \quad (2.5f)$$

$$\| e \|_1 \leq \frac{l^3}{72\sqrt{3}} \operatorname{Max} \left| \frac{\partial^4 u_{ex}}{\partial x^4} \right|_{V^e} \quad (2.5f)$$

Note that the order of magnitude of the error for this element is similar to that of the four-node Lagrangian element. However, the coefficient $1/384$ of $\| e \|_0$ is greater than for the Lagrangian element ($1/1,944$), although the coefficient $1/72\sqrt{3}$ of $\| e \|_1$ is smaller than for the Lagrangian element ($1/108$). This element has C^1 continuity: u and $\partial u / \partial x$ are continuous within the element and across the boundary of the element.

2.2.3.2 Fifth-order element (two nodes, C^2)

There are now three nodal variables per node: u_i , $\partial_x u_i$, and $\partial_x^2 u_i$



$$\langle P \rangle = \langle 1 \quad \xi \quad \xi^2 \quad \xi^3 \quad \xi^4 \quad \xi^5 \rangle \quad (2.6a)$$

$$[P_n] = \begin{bmatrix} \langle P(\xi_1) \rangle \\ \langle \frac{\partial P}{\partial \xi}(\xi_1) \rangle \\ \langle \frac{\partial^2 P}{\partial \xi^2}(\xi_1) \rangle \\ \hline \langle P(\xi_2) \rangle \\ \langle \frac{\partial P}{\partial \xi}(\xi_2) \rangle \\ \langle \frac{\partial^2 P}{\partial \xi^2}(\xi_2) \rangle \end{bmatrix}; \quad [P_n]^{-1} = \frac{1}{16} \begin{bmatrix} 8 & 5 & 1 & 8 & -5 & 1 \\ -15 & -7 & -1 & 15 & -7 & 1 \\ 0 & -6 & -2 & 0 & 6 & -2 \\ 10 & 10 & 2 & -10 & 10 & -2 \\ 0 & 1 & 1 & 0 & -1 & 1 \\ -3 & -3 & -1 & 3 & -3 & 1 \end{bmatrix} \quad (2.6b)$$

$$u(\xi) = \langle N \rangle \{ u_n \}_\xi$$

	$\frac{1}{c} \{ N \}$	$\frac{1}{c} \{ \partial N / \partial \xi \}$	c for $\{ u_n \}_\xi$	c for $\{ u_n \}$
1	$(1-\xi)^3 (8+9\xi+3\xi^2)$	$-15(1-\xi^2)^2$		$1/16$
2	$(1-\xi^3)(1+\xi)(5+3\xi)$	$-(1-\xi)^2 (1+3\xi) (7+5\xi)$		$l/32$
3	$(1-\xi)^3 (1+\xi)^2$	$-(1-\xi)^2 (1+\xi) (1+5\xi)$	$1/16$	$l^2/64$
4	$(1+\xi)^3 (8-9\xi+3\xi)$	$15(1-\xi^2)^2$		$l/16$
5	$(1+\xi)^3 (-1+\xi)(5-3\xi)$	$-(1+\xi)^2 (1-3\xi) (7-5\xi)$		$l/32$
6	$(1+\xi)^3 (1-\xi)^2$	$(1+\xi)^2 (1-\xi) (1-5\xi)$		$l^2/64$

$$e(\xi) \leq \frac{1}{720} (1 - \xi^2)^3 \operatorname{Max} \left| \frac{\partial^6 u_{ex}}{\partial \xi^6} \right|_{V^r}$$

$$|e|_0 \leq \frac{1}{720} \frac{l^6}{64} \operatorname{Max} \left| \frac{\partial^6 u_{ex}}{\partial x^6} \right|_{V^e} \quad (2.6c)$$

$$|e|_1 \leq \frac{1}{720} \frac{l^5}{8\sqrt{3}} \operatorname{Max} \left| \frac{\partial^6 u_{ex}}{\partial x^6} \right|_{V^e}. \quad (2.6d)$$

This element exhibits type C^2 continuity: u and its first two derivatives are continuous both within the element and across the boundary.

2.2.4 GENERAL ELEMENTS

We can construct general elements at will by combining the following techniques:

- Increasing the number of interpolation nodes and using a variable number of degrees of freedom at each node. Thus, the element can belong to both the Lagrangian and Hermite families simultaneously.
- Adding a non-nodal approximation to the nodal approximation described above. This adds degrees of freedom that are not related to the nodes, but are related to the element.

The geometric nodes $\langle \bar{N} \rangle$ and $[J]$ are still the same as those in section 2.2.1.

2.2.4.1 Fourth-order Lagrange–Hermite element (three nodes, C^1)

$$\begin{array}{ccccccc} \left\{ \begin{array}{l} u_1 \\ \partial_\xi u_1 \end{array} \right\} & u_2 & \left\{ \begin{array}{l} u_3 \\ \partial_\xi u_3 \end{array} \right\} & u_1 = \left\{ \begin{array}{l} u_1 \\ \partial_x u_1 \end{array} \right\} & u_2 & u_3 = \left\{ \begin{array}{l} u_3 \\ \partial_x u_3 \end{array} \right\} \\ \hline \text{---} & \text{---} & \text{---} & \text{---} & \text{---} & \text{---} \\ -1 & 0 & 1 & x_1 & x_2 & x \\ \end{array}$$

$V^r \quad n=3 \quad n_d=5 \quad V^e$

$$\langle P \rangle = \langle 1 \ \xi \ \xi^2 \ \xi^3 \ \xi^4 \rangle \quad (2.7a)$$

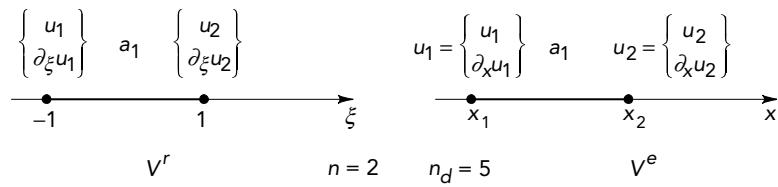
$$[P_n] = \begin{bmatrix} \langle P(\xi_1) \rangle \\ \langle \frac{\partial P}{\partial \xi}(\xi_1) \rangle \\ \hline \langle P(\xi_2) \rangle \\ \hline \langle P(\xi_3) \rangle \\ \langle \frac{\partial P}{\partial \xi}(\xi_3) \rangle \end{bmatrix}; \quad [P_n]^{-1} = \frac{1}{4} \begin{bmatrix} 0 & 0 & 4 & 0 & 0 \\ -3 & -1 & 0 & 3 & -1 \\ 4 & 1 & -8 & 4 & -1 \\ 1 & 1 & 0 & -1 & 1 \\ -2 & -1 & 4 & -2 & 1 \end{bmatrix} \quad (2.7b)$$

	$\frac{1}{c} \{N\}$	$\frac{1}{c} \{\partial N / \partial \xi\}$	c for $\{u_n\}_\xi$	c for $\{u_n\}$
1	$-\xi(1-\xi)^2(3+2\xi)$	$(1-\xi^2)(-3+8\xi)$		$1/4$
2	$-\xi(1-\xi)(1-\xi^2)$	$(1-\xi)(-1+\xi+4\xi^2)$		$1/8$
3	$4(1-\xi^2)^2$	$-16\xi(1-\xi^2)$	$1/4$	$1/4$
4	$\xi(1+\xi)^2(3-2\xi)$	$(1-\xi^2)(3+8\xi)$		$1/4$
5	$-\xi(1+\xi)(1-\xi^2)$	$(1+\xi)(-1-\xi+4\xi^2)$		$1/8$

The error is: $|e|_0 \leq \frac{l^5}{3,840} \text{Max} \left| \frac{\partial^5 u}{\partial x^5} \right|$.

2.2.4.2 Hermite element with one internal degree of freedom (two nodes, C^1)

To the nodal variables of the element in section 2.2.2.1, we add a generalized variable a_1 :



$$\langle P \rangle = \langle 1 \ \xi \ \xi^2 \ \xi^3 \ \xi^4 \rangle \quad (2.8a)$$

$$[P_n] = \begin{bmatrix} \langle P(\xi_1) \rangle \\ \langle \frac{\partial P}{\partial \xi}(\xi_1) \rangle \\ \hline \langle P(\xi_2) \rangle \\ \langle \frac{\partial P}{\partial \xi}(\xi_2) \rangle \\ \hline \langle 0 0 0 0 1 \rangle \end{bmatrix}; \quad [P_n]^{-1} = \frac{1}{4} \begin{bmatrix} 2 & 1 & 2 & -1 & 4 \\ -3 & -1 & 3 & -1 & 0 \\ 0 & -1 & 0 & 1 & -8 \\ 1 & 1 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 4 \end{bmatrix} \quad (2.8b)$$

$$u(\xi) = \langle N_1 \ N_2 \ N_3 \ N_4 \ P_1 \rangle \{u_n\}_\xi.$$

The functions N_1 to N_4 correspond to the functions N_1 , N_2 , N_3 and N_4 from section 2.2.3.1. In addition:

$$P_1 = (1 - \xi^2)^2$$

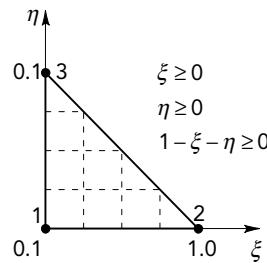
P_1 and $\partial P_1 / \partial \xi$ are eliminated at the two nodes. P_1 is identical to the function N_3 from the previous section.

$$\{u_n\}_\xi = \begin{Bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ a_1 \end{Bmatrix}_\xi; \quad \{u_n\} = \begin{Bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ a_1 \end{Bmatrix}; \quad \{u_n\}_\xi = \begin{Bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{Bmatrix} \{u_n\} \quad (2.8c)$$

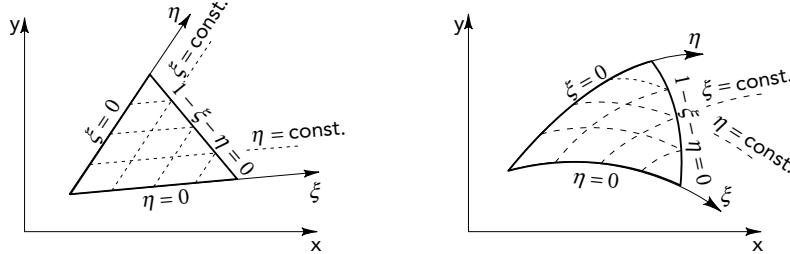
2.3 Triangular elements (two dimensions)

2.3.1 SYSTEMS OF COORDINATES

For all triangular elements, we use the following reference element:



The coordinates (ξ, η) can be interpreted as curvilinear coordinates for the real element:



The barycentric coordinates L_1, L_2 and L_3 are often used to mark a point 0 of a straight-sided triangle:

$$L_1 = \frac{A_1}{A}; \quad L_2 = \frac{A_2}{A}; \quad L_3 = \frac{A_3}{A} \quad (2.9a)$$

$$A = A_1 + A_2 + A_3$$

$$L_1 + L_2 + L_3 = 1$$

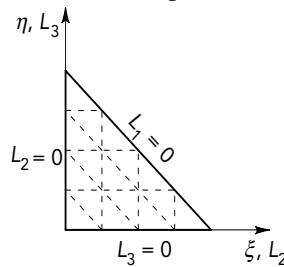
A_1, A_2 and A_3 are the areas of the subtriangles 0-2-3, 0-3-1 and 0-1-2.

A is the area of the triangle 1-2-3.

The coordinates L_1, L_2 and L_3 are linked to the coordinates (ξ, η) by the following relation:

$$L_1 \equiv 1 - \xi - \eta, \quad L_2 \equiv \xi, \quad L_3 \equiv \eta. \quad (2.9b)$$

The reference element can be used to represent the space L_1, L_2, L_3 :



The variables beneath the integral symbol are altered as follows:

$$\int \int_{x \ y} f(x, y) dx dy = \int \int_{\xi \ \eta} f(\xi, \eta) |J| d\xi d\eta$$

By convention, we number the nodes of the reference and real elements in the positive sense of rotation on the oriented surface, starting with a corner node. The following notation is used:

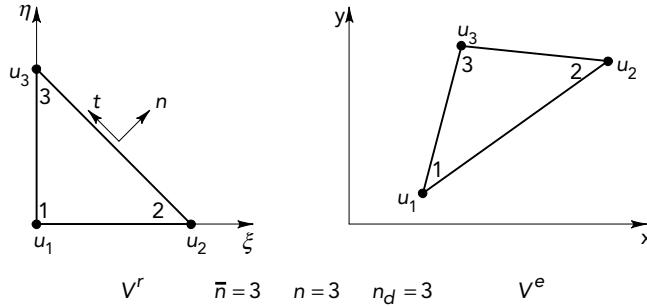
$$u(\xi, \eta) = \langle P(\xi, \eta) \rangle \{a\} = \langle N(\xi, \eta) \rangle \{u_n\}.$$

For an isoparametric element: $x(\xi, \eta) = \langle N(\xi, \eta) \rangle \{x_n\}$,

$$y(\xi, \eta) = \langle N(\xi, \eta) \rangle \{y_n\}.$$

The Jacobian matrix is defined by equation (1.36b).

2.3.2 LINEAR ELEMENT (triangle, three nodes, C^0)



The geometric nodes and the interpolation nodes are identical.

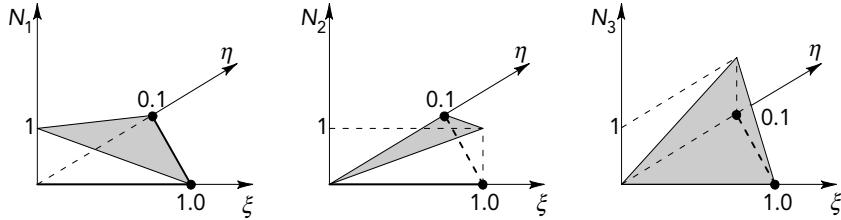
$$\langle P \rangle = \langle 1 \ \xi \ \eta \rangle \quad (2.10a)$$

$$[P_n] = \begin{bmatrix} \langle P(\xi_1) \rangle \\ \langle P(\xi_2) \rangle \\ \langle P(\xi_3) \rangle \end{bmatrix}; \quad [P_n]^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \quad (2.10b)$$

	$\{N\}$	$\{\partial N/\partial \xi\}$	$\{\partial N/\partial \eta\}$
1	$1 - \xi - \eta$	-1	-1
2	ξ	1	0
3	η	0	1

$$[J] = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{bmatrix}; \det(J) = 2 A = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1). \quad (2.10c)$$

The plots of the functions N are shown:



The truncation error can be obtained from equation (1.68), extended to two-dimensional space.

$$e(\xi, \eta) = \frac{1}{2} \langle \xi(1-\xi); -\xi\eta; \eta(1-\eta) \rangle \left\{ \begin{array}{l} \frac{\partial^2 u_{ex}}{\partial \eta^2} \\ 2 \frac{\partial^2 u_{ex}}{\partial \xi \partial \eta} \\ \frac{\partial^2 u_{ex}}{\partial \eta^2} \end{array} \right\}_{\bar{\xi} \text{ sur } V^r}$$

$$|e|_0 \leq C_0 l^2 \operatorname{Max} |D_x^2 u_{ex}|_{V^e} \quad (2.10d)$$

$$\text{where } \operatorname{Max} |D_x^2 u_{ex}|_{V^e} = \operatorname{Max}_{V^e} \left(\left| \frac{\partial^2 u_{ex}}{\partial x^2} \right|; \left| \frac{\partial^2 u_{ex}}{\partial x \partial y} \right|; \left| \frac{\partial^2 u_{ex}}{\partial y^2} \right| \right).$$

We can show that ([SYN 57; ODE 72, p. 130; ZLA 73]):

$$|e|_1 \leq C_1 \frac{l}{\sin \theta} \operatorname{Max} |D_x^2 u_{ex}|_{V^e} \quad (2.10e)$$

where l is the largest dimension of the element and θ the largest interior angle of the triangular element.

The term $l/\sin \theta$ appears when $\partial u/\partial \xi$ is transformed into $\partial u/\partial x$, which involves the inverse of the Jacobian matrix.

The function $u(x, y)$ and its first derivatives are continuous within the element. The function $u(x)$ is continuous across the boundary of the element, but its first derivatives are not. The tangential derivative $\partial u/\partial t$ is continuous, and only the normal derivative $\partial u/\partial n$ is discontinuous (see section 2.3.3.1).

Remark

The approximation in (x, y) (see Example 1.11) can be obtained by choosing:

$$\langle P \rangle = \langle 1 \ x \ y \rangle.$$

We have:

$$[P_N] = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix}; \det(P_N) = 2.A$$

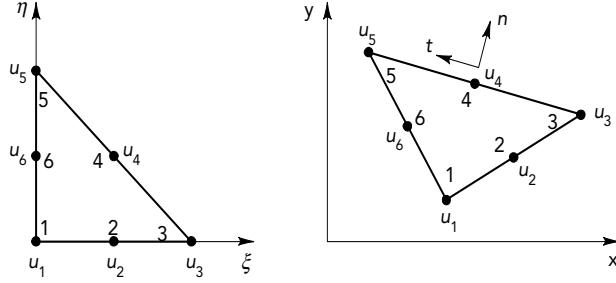
and

$$\langle N(x, y) \rangle = \langle P \rangle [P_N]^{-1}.$$

2.3.3 HIGH-PRECISION LAGRANGIAN ELEMENTS (continuity C^0)

These elements are obtained by increasing the number of interpolation nodes on the boundary, and/or inside the element discussed in the previous section. The same geometric nodes, the functions \bar{N} , and the constant Jacobian matrix $[J]$ are preserved. These elements are thus said to be subparametric. Curvilinear elements will be introduced in section 2.3.3.5.

2.3.3.1 Quadratic element (triangle, six nodes, C^0)



$$n = 6 \quad n_d = 6$$

$$\langle P \rangle = \langle 1 \ \xi \ \eta \ \xi^2 \ \xi\eta \ \eta^2 \rangle \quad (2.11a)$$

$$\langle \xi_i \rangle = \left\langle 0 \ 0; \frac{1}{2} \ 0; 1 \ 0; \frac{1}{2} \ \frac{1}{2}; 0 \ 1; 0 \ \frac{1}{2} \right\rangle$$

$$\langle \mathbf{x}_i \rangle = \langle x_1 \ y_1; x_2 \ y_2; x_3 \ y_3; x_4 \ y_4; x_5 \ y_5; x_6 \ y_6 \rangle$$

where $\mathbf{x}_2 = \frac{1}{2}(\mathbf{x}_1 + \mathbf{x}_3); \mathbf{x}_4 = \frac{1}{2}(\mathbf{x}_3 + \mathbf{x}_5); \mathbf{x}_6 = \frac{1}{2}(\mathbf{x}_5 + \mathbf{x}_1)$

$$[P_n]^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -3 & 4 & -1 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & -1 & 4 \\ 2 & -4 & 2 & 0 & 0 & 0 \\ 4 & -4 & 0 & 4 & 0 & -4 \\ 2 & 0 & 0 & 0 & 2 & -4 \end{bmatrix} \quad (2.11b)$$

	$\{N\}$	$\{\partial N / \partial \xi\}$	$\{\partial N / \partial \eta\}$
1	$-\lambda(1-2\lambda)$	$1-4\lambda$	$1-4\lambda$
2	$4\xi\lambda$	$4(\lambda-\xi)$	-4ξ
3	$-\xi(1-2\xi)$	$-1+4\xi$	0
4	$4\xi\eta$	4η	4ξ
5	$-\eta(1-2\eta)$	0	$-1+4\eta$
6	$4\eta\lambda$	-4η	$4(\lambda-\eta)$

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

$$|e|_0 \leq C_0 l^3 \operatorname{Max} |D_x^3 u_{ex}| \quad (2.11c)$$

$$|e|_1 \leq C_1 \frac{l^2}{\sin \theta} \operatorname{Max} |D_x^3 u_{ex}| \quad (\text{see [ODE 72], p. 134}). \quad (2.11d)$$

The element's continuity is C^0 : u and $\partial u / \partial t$ are continuous on each side, but $\partial u / \partial n$ is discontinuous.

For example, on the side 3-5, $\eta = 1 - \xi$:

$$u_{3-5} = \langle 0; 0; -\xi(1-2\xi); 4\xi(1-\xi); (-1+\xi)(-1+2\xi); 0 \rangle \{u_n\}.$$

This expression depends only on the variables u_3 , u_4 and u_5 linked to the side 3-5. The function u is therefore continuous on this side. The parameter ξ is related to the local coordinate t :

$$t = \frac{l}{2} - l\xi \quad 0 \leq \xi \leq 1 \quad -\frac{l}{2} \leq t \leq \frac{l}{2}$$

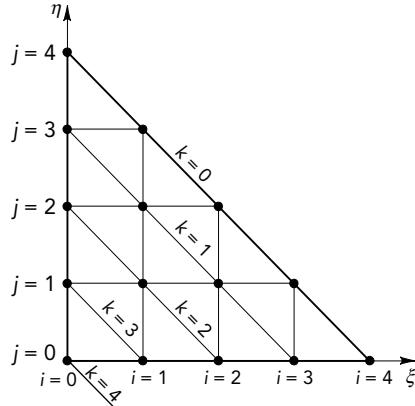
where l is the length of the side 3-5. From this we derive u_{3-5} :

$$\frac{\partial u_{3-5}}{\partial t} = \frac{\partial u}{\partial \xi} \frac{\partial \xi}{\partial t} = -l \langle 0; 0; -1+4\xi; 4-8\xi; -3+4\xi; 0 \rangle \{u_n\}.$$

Similarly, this expression depends only on u_3 , u_4 and u_5 ; thus $\partial u / \partial t$ is continuous on the side 3-5. Conversely, $\partial u / \partial n$ is a linear combination of $(\partial u / \partial \xi)_{3-5}$ and $(\partial u / \partial \eta)_{3-5}$, which depends on all the nodal variables u_i . $\partial u / \partial n$ is therefore not continuous on the side 3-5.

2.3.3.2 Complete polynomial element of order r (triangle, n nodes, C^0)

The r -order complete polynomial contains $n = (r+1)(r+2)/2$ terms. Hence we need n nodes with one degree of freedom. We place three r nodes at regular intervals along the boundary, and the remaining nodes within the element. For instance, for $r = 4$:



A node can be identified by three integers i, j and k , linked to the coordinates of the nodes $\xi_i \eta_j$ by the following expressions:

$$\xi_i = \frac{i}{r} \quad \eta_j = \frac{j}{r} \quad 0 \leq i + j \leq r$$

and

$$k = r - i - j.$$

For the example illustrated above, the three peak nodes are marked by the indicators: $(0, 0, 4; 4, 0, 0; 0, 4, 0)$.

This means we can explicitly construct the interpolation functions corresponding to each node (i, j and k) as the product of the equations of the straight lines passing through all the nodes except (i, j) :

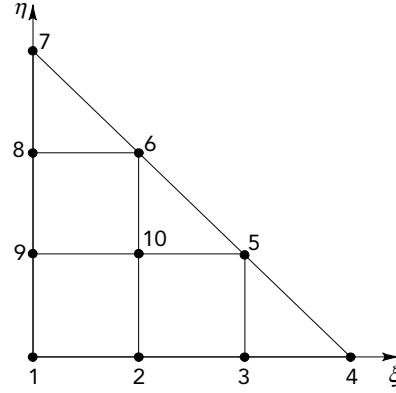
$$N(i, j, k) = \prod_{l=0}^{i-1} \frac{l-r \xi}{l-i} \prod_{m=0}^{j-1} \frac{m-r \eta}{m-j} \prod_{n=0}^{k-1} \frac{n-r(1-\xi-\eta)}{n-k}. \quad (2.12a)$$

Note that we can also construct these functions using the method in section 1.4.1. The error is similar in form to (2.10d):

$$|e|_0 \leq C_0 l^{r+1} \operatorname{Max} |D_x^{r+1} u_{ex}|_{V^e} \quad (2.12b)$$

$$|e|_1 \leq C_1 \frac{l^r}{\sin \theta} \operatorname{Max} |D_x^{r+1} u_{ex}|_{V^e}. \quad (2.12c)$$

2.3.3.3 Complete cubic element (triangle, 10 nodes, C^0)



$$n = 10 \quad n_d = 10$$

The functions N of this element are constructed by directly applying the method from the previous section, where $r = 3$:

$$\langle P \rangle = \langle 1 \ \xi \ \eta \ \xi^2 \ \xi\eta \ \eta^2 \ \xi^3 \ \xi^2 \ \eta \ \xi\eta^2 \ \eta^3 \rangle \quad (2.13a)$$

$$[P_n]^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -5,5 & 9 & -4,5 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -5,5 & 0 & 0 & 0 & 0 & 0 & 1 & -4,5 & 9 & 0 \\ 9 & -22,5 & 18 & -4,5 & 0 & 0 & 0 & 0 & 0 & 0 \\ 18 & -22,5 & 4,5 & 0 & -4,5 & -4,5 & 0 & 4,5 & -22,5 & 27 \\ 9 & 0 & 0 & 0 & 0 & 0 & -4,5 & 18 & -22,5 & 27 \\ -4,5 & 13,5 & -13,5 & 4,5 & 0 & 0 & 0 & 0 & 0 & 0 \\ -13,5 & 27 & -13,5 & 0 & 13,5 & 0 & 0 & 0 & 13,5 & -27 \\ -13,5 & 13,5 & 0 & 0 & 0 & 13,5 & 0 & -13,5 & 27 & -27 \\ -4,5 & 0 & 0 & 0 & 0 & 0 & 4,5 & -13,5 & 13,5 & 0 \end{bmatrix} \quad (2.13b)$$

	$\frac{1}{c} \{ N \}$	$\frac{1}{c} \{ \partial N_i / \partial \xi \}$	$\frac{1}{c} \{ \partial N / \partial \eta \}$	c
1	$\lambda(-1 + 3\lambda)(-2 + 3\lambda)$	$-2 + 18\lambda - 27\lambda^2$	$-2 + 18\lambda - 27\lambda^2$	
2	$9\lambda\xi(-1 + 3\lambda)$	$9\lambda(-1 + 3\lambda - 6\xi) + 9\xi$	$-9\xi(-1 + 6\lambda)$	
3	$9\lambda\xi(-1 + 3\xi)$	$9\xi(1 + 6\lambda - 3\xi) - 9\lambda$	$-9\xi(-1 + 3\xi)$	
4	$\xi(-1 + 3\xi)(-2 + 3\xi)$	$2 - 18\xi + 27\xi^2$	0	
5	$9\xi\eta(-1 + 3\xi)$	$9\eta(-1 + 6\xi)$	$9\xi(-1 + 3\xi)$	
6	$9\xi\eta(-1 + 3\eta)$	$9\eta(-1 + 3\eta)$	$9\xi(-1 + 6\eta)$	$1/2$
7	$\eta(-1 + 3\eta)(-2 + 3\eta)$	0	$2 - 18\eta + 27\eta^2$	
8	$9\lambda\eta(-1 + 3\eta)$	$-9\eta(-1 + 3\eta)$	$9\eta(1 + 6\lambda - 3\eta) - 9\lambda$	
9	$9\lambda\eta(-1 + 3\lambda)$	$-9\eta(-1 + 6\lambda)$	$9\lambda(-1 + 3\lambda - 6\eta) + 9\eta$	
10	$54\xi\eta\lambda$	$54\eta(\lambda - \xi)$	$54\xi(\lambda - \eta)$	

with

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

for barycentric coordinates

$$L_1 = \lambda, \quad L_2 = \xi, \quad L_3 = \eta.$$

2.3.3.4 Incomplete cubic element (triangle, nine nodes, C^0)

If we wish to avoid interior nodes like node 10 in the previous element, there are two techniques that we can use:

- Constructing the functions N from an incomplete polynomial basis with nine terms by eliminating $\xi^2\eta$ or $\xi\eta^2$ in (2.13a). The combination $\xi^2\eta + \xi\eta^2$ cannot be used in $[P_n]$, because it renders $\langle P \rangle$ singular. The functions N are then constructed in accordance with section 1.4.1.
- Expressing the nodal variable u_{10} in the form of a linear combination of the variables $u_1 \dots u_9$ [MIT 77]:

$$u_{10} = \frac{1}{4}(u_2 + u_3 + u_5 + u_6 + u_8 + u_9) - \frac{1}{6}(u_1 + u_4 + u_7). \quad (2.14a)$$

The coefficients $1/4$ and $-1/6$ are obtained such that the modified functions $\langle N_1, N_2 \dots N_9 \rangle$ include all the terms of a quadratic polynomial so that (1.31) is satisfied, and must respect the symmetry of the reference triangle:

$$\begin{aligned} \{N\} &= \{N\}^{(1)} + \{a\} N_{10}; \quad \left\{ \frac{\partial N}{\partial \xi} \right\} = \left\{ \frac{\partial N}{\partial \xi} \right\}^{(1)} + \{a\} \frac{\partial N_{10}}{\partial \xi}; \\ \left\{ \frac{\partial N}{\partial \eta} \right\} &= \left\{ \frac{\partial N}{\partial \eta} \right\}^{(1)} + \{a\} \frac{\partial N_{10}}{\partial \eta} \end{aligned} \quad (2.14b)$$

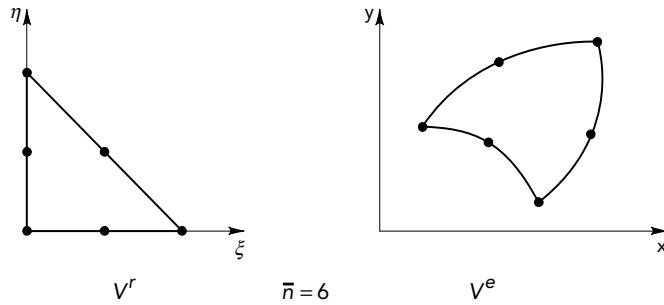
where the functions $\{N\}^{(1)}$, $\{\partial N / \partial \xi\}^{(1)}$ and $\{\partial N / \partial \eta\}^{(1)}$ are the first nine functions in the table above, and

$$a^T = \langle -\frac{1}{6} \quad \frac{1}{4} \quad \frac{1}{4} \quad -\frac{1}{6} \quad \frac{1}{4} \quad \frac{1}{4} \quad -\frac{1}{6} \quad \frac{1}{4} \quad \frac{1}{4} \rangle. \quad (2.14c)$$

2.3.3.5 Curvilinear elements

Elements with curvilinear sides are used to represent a domain with curvilinear boundaries. Such elements can be created by systematically increasing the number of geometric nodes on the boundary of the element.

a) Element with parabolic boundaries



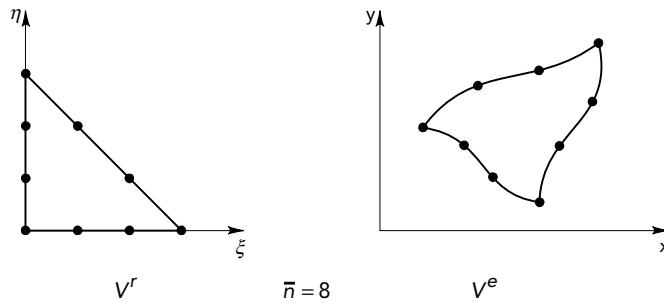
$$\begin{aligned} x &= \langle \bar{N} \rangle \{ \bar{x}_n \} \\ y &= \langle \bar{N} \rangle \{ \bar{y}_n \} \end{aligned}$$

where functions $\langle \bar{N} \rangle$ are identical to functions $\langle N \rangle$ from section 2.3.3.1;

$\{ \bar{x}_n \}$ and $\{ \bar{y}_n \}$ are the coordinates of the six geometric nodes of the real element.

This element is isoparametric if we use the approximation of u defined in section 2.3.3.1. The Jacobian matrix of this element is defined according to equation (1.43).

b) Element with cubic boundaries

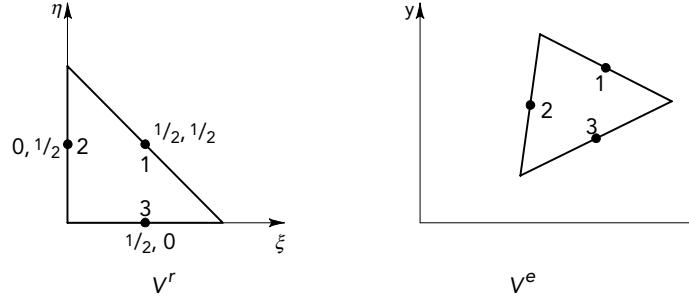


The geometric interpolation functions $\langle \bar{N} \rangle$ are identical to the interpolation functions $\langle N \rangle$ from section 2.3.3.4.

Note that the following condition must be respected absolutely, so the geometric distortion should not be so severe as to negate it.

$$\det(J) > 0.$$

2.3.3.6 Non-compatible element (triangle, three nodes) semi- C^0



$$\langle P \rangle = \langle 1 \quad \xi \quad \eta \rangle \quad (2.15a)$$

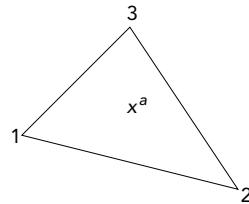
$$[P_n] = \begin{bmatrix} P(\xi_1) \\ P(\xi_2) \\ P(\xi_3) \end{bmatrix}; \quad [P_n]^{-1} = \begin{bmatrix} -1 & 1 & 1 \\ 2 & -2 & 0 \\ 2 & 0 & -2 \end{bmatrix} \quad (2.15b)$$

$\{N\}$	$\{\partial N / \partial \xi\}$	$\{\partial N / \partial \eta\}$
1 $-1 + 2\xi + 2\eta$	2	2
2 $1 - 2\xi$	-2	0
3 $1 - 2\eta$	0	-2

This element does not preserve interelement continuity of the function u at the nodes [BAT 90].

2.3.3.7 Element with “bubble” function [ARN 84]

In fluid mechanics, a special element with a so-called “bubble” function is sometimes used:



$$u(\xi, \eta) = \sum_{i=1}^3 N_i u_i + N_1 N_2 N_3 \cdot a \quad \text{where} \quad N_1 = 1 - \xi - \eta; \quad N_2 = \xi; \quad N_3 = \eta,$$

where a is a generalized internal variable. There is an equivalent three-dimensional element (see section 2.5.2):

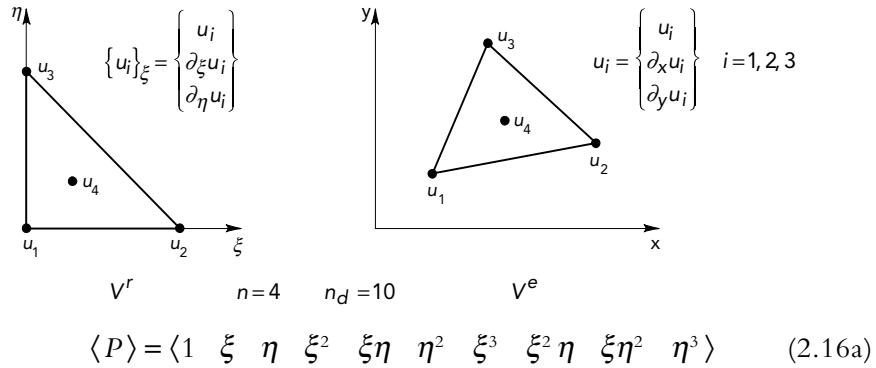
$$u(\xi, \eta, \zeta) = \sum_{i=1}^4 N_i u_i + N_1 N_2 N_3 N_4 \cdot a,$$

where $N_1 = 1 - \xi - \eta - \zeta; N_2 = \xi; N_3 = \eta; N_4 = \zeta.$

2.3.4 HIGH-PRECISION HERMITE ELEMENTS

The geometry of the element can be linear (section 2.2.1), quadratic (section 2.3.3.5a) or cubic (section 2.3.3.5b). In the latter two cases, the number of geometric nodes is greater than the number of functional interpolation nodes.

2.3.4.1 Complete cubic element (triangle, four nodes, semi- C^1)



$$[P_n] = \begin{bmatrix} \langle P(\xi_i) \rangle \\ \langle \frac{\partial P}{\partial \xi}(\xi_i) \rangle \\ \langle \frac{\partial P}{\partial \eta}(\xi_i) \rangle \\ \dots \\ \langle P(\xi_4) \rangle \end{bmatrix} \quad i = 1, 2, 3 \quad (2.16b)$$

$$u(\xi) = \langle N \rangle \{u_n\}_\xi.$$

	$\{N\}$	$\{\partial N / \partial \xi\}$	$\{\partial N / \partial \eta\}$
Node 1	1 $\lambda^2(3-2\lambda)-7a$	$6\lambda(-1+\lambda)-7b$	$6\lambda(-1+\lambda)-7c$
	2 $\xi\lambda^2-a$	$\lambda(\lambda-2\xi)-b$	$-2\xi\lambda-c$
	3 $\eta\lambda^2-a$	$-2\lambda\eta-b$	$\lambda(\lambda-2\eta)-c$
Node 2	4 $\xi^2(3-2\xi)-7a$	$6\xi(1-\xi)-7b$	$-7c$
	5 $\xi^2(-1+\xi)+2a$	$\xi(-2+3\xi)+2b$	$2c$
	6 $\xi^2\eta-a$	$2\xi\eta-b$	ξ^2-c
Node 3	7 $\eta^2(3-2\eta)-7a$	$-7b$	$6\eta(1-\eta)-7c$
	8 $\xi\eta^2-a$	η^2-b	$2\xi\eta-c$
	9 $\eta^2(-1+\eta)+2a$	$2b$	$\eta(-2+3\eta)+2c$
Node 4	10 $27a$	$27b$	$27c$

where $\lambda = 1 - \xi - \eta$; $a = \xi\eta\lambda$; $b = \eta(\lambda - \xi)$; $c = \xi(\lambda - \eta)$

$$\langle u_n \rangle_\xi = \langle \langle u_1 \rangle_\xi \quad \langle u_2 \rangle_\xi \quad \langle u_3 \rangle_\xi \quad \langle u_4 \rangle \rangle$$

$$\langle u_n \rangle = \langle \mathbf{u}_1 \quad \mathbf{u}_2 \quad \mathbf{u}_3 \quad \mathbf{u}_4 \rangle$$

$$\{u_n\}_\xi = [T] \{u_n\}$$

$$[T] = \begin{bmatrix} [T_1] & & \\ & [T_2] & \\ & & [T_3] \\ & & & 1 \end{bmatrix} \quad \text{where} \quad [T_i] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & [J(\xi_i)] \\ 0 & & \end{bmatrix} \quad i = 1, 2, 3. \quad (2.16c)$$

2.3.4.2 Incomplete cubic element (triangle, three nodes, semi- C^0)

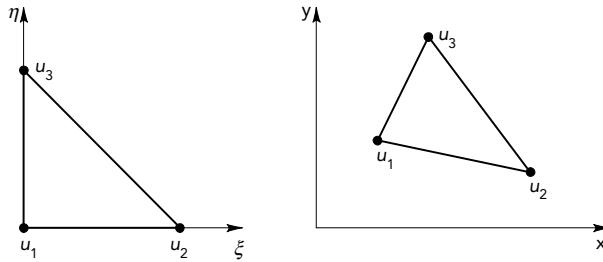
The central node of the previous element can be eliminated by expressing nodal variable u_4 as a linear combination of variables u_1, u_2, u_3 and their derivatives [MIT 77]:

$$\begin{aligned} u_4 = & \frac{1}{3}(u_1 + u_2 + u_3) + \frac{1}{18}(\partial_\xi u_1 - 2\partial_\xi u_2 + \partial_\xi u_3) \\ & + \frac{1}{18}(\partial_\eta u_1 + \partial_\eta u_2 - 2\partial_\eta u_3) \end{aligned}$$

The coefficients $1/3, -1/9$ and $1/18$ are obtained by imposing the following condition: the modified functions $\langle N_1 \ N_2 \ \dots \ N_9 \rangle$ must contain all the terms of a quadratic polynomial such that equation [1.31] is satisfied and must respect the symmetry of the triangle.

Functions N and their derivatives are obtained using a technique similar to that used in section 2.3.3.4. Function u and its first derivatives are continuous at the three nodes. u and $\partial u / \partial t$ are continuous across boundaries, but $\partial u / \partial n$ is generally discontinuous.

2.3.4.3 Fifth-order element (triangle, three nodes, C^1)



$$\langle u_i \rangle_{\xi} = \langle u_i \partial_{\xi} u_i \partial_{\eta} u_i \partial_{\xi\xi} u_i \partial_{\xi\eta} u_i \partial_{\eta\eta} u_i \rangle; \quad \langle u_i \rangle = \langle u_i \partial_x u_i \partial_y u_i \partial_{xx} u_i \partial_{xy} u_i \partial_{yy} u_i \rangle \quad i = 1, 2, 3$$

The polynomial basis $\langle P \rangle$ is complete up to the fourth order and contains three fifth-order terms such that the normal derivative on each side of the reference element varies cubically in ξ and η :

$$\begin{aligned} \langle P \rangle = & \langle 1 \ \xi \ \eta; \ \xi^2 \ \xi\eta \ \eta^2; \ \xi^3 \ \xi^2\eta \ \xi\eta^2 \ \eta^3; \\ & \xi^4 \ \xi^3\eta \ \xi^2\eta^2 \ \xi\eta^3 \ \eta^4; \ \xi^5 - 5\xi^3\eta^2; \\ & \xi^2\eta^3 - \xi^3\eta^2; \ \eta^5 - 5\xi^3\eta^2 \rangle. \end{aligned} \quad (2.17a)$$

Matrix $[P_n]^{-1}$ of dimension (18×18) can thus be inverted to obtain the coefficients of the 18 functions $\langle N \rangle$. Suppose that the nodal variables are organized as follows:

$$\langle u_n \rangle = \langle \mathbf{u}_1; \ \mathbf{u}_2; \ \mathbf{u}_3 \rangle$$

where $\mathbf{u}_1, \mathbf{u}_2$ and \mathbf{u}_3 are the variables attached to the nodes of the reference element, which thus entails derivatives in ξ and η . The interpolation functions are therefore as follows [MIT 77]:

$$\text{Node 1} \left\{ \begin{array}{l} N_1 = \lambda^2(10\lambda - 15\lambda^2 + 6\lambda^3 + 30\xi\eta(\xi + \eta)) \\ N_2 = \xi\lambda^2(3 - 2\lambda - 3\xi^2 + 6\xi\eta) \\ N_3 = \eta\lambda^2(3 - 2\lambda - 3\eta^2 + 6\xi\eta) \\ N_4 = \frac{1}{2}\xi^2\lambda^2(1 - \xi + 2\eta) \\ N_5 = \xi\eta\lambda^2 \\ N_6 = \frac{1}{2}\eta^2\lambda^2(1 + 2\xi - \eta) \end{array} \right.$$

$$\text{Node 2} \left\{ \begin{array}{l} N_7 = \xi^2(10\xi - 15\xi^2 + 6\xi^3 + 15\eta^2\lambda) \\ N_8 = \frac{\xi^2}{2}(-8\xi + 14\xi^2 - 6\xi^3 - 15\eta^2\lambda) \\ N_9 = \frac{\xi^2\eta}{2}(6 - 4\xi - 3\eta - 3\eta^2 + 3\xi\eta) \\ N_{10} = \frac{\xi^2}{4}(2\xi(1-\xi)^2 + 5\eta^2\lambda) \\ N_{11} = \frac{\xi^2\eta}{2}(-2 + 2\xi + \eta + \eta^2 - \xi\eta) \\ N_{12} = \frac{\xi^2\eta^2\lambda}{4} + \frac{\xi^3\eta^2}{2} \end{array} \right.$$

$$\text{Node 3} \left\{ \begin{array}{l} N_{13} = \eta^2(10\eta - 15\eta^2 + 6\eta^3 + 15\xi^2\lambda) \\ N_{14} = \frac{\xi\eta^2}{2}(6 - 3\xi - 4\eta - 3\xi^2 + 3\xi\eta) \\ N_{15} = \frac{\eta^2}{2}(-8\eta + 14\eta^2 - 6\eta^3 - 15\xi^2\lambda) \\ N_{16} = \frac{\xi^2\eta^2\lambda}{4} + \frac{\xi^2\eta^3}{2} \\ N_{17} = \frac{\xi\eta^2}{2}(-2 + \xi + 2\eta + \xi^2 - \xi\eta) \\ N_{18} = \frac{\eta^2}{4}(2\eta(1-\eta)^2 + 5\xi^2\lambda) \end{array} \right.$$

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

The transition from the nodal variables of the reference element to the nodal variables of the real element is achieved using a transformation matrix T , constructed based on the Jacobian matrix and the matrices $[C_1]$ and $[C_2]$ from (1.46), evaluated at each node i :

$$\{u_i\}_\xi = [T_i]\{u_i\}. \quad (2.17b)$$

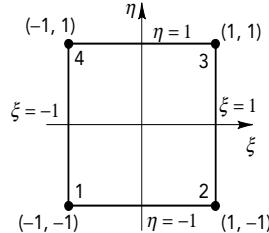
For instance, for an element with straight sides, whose Jacobian matrix is constant:

$$[T_i] = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & J_{11} & J_{12} & 0 & 0 & 0 \\ 0 & J_{21} & J_{22} & 0 & 0 & 0 \\ 0 & 0 & 0 & J_{11}^2 & 2J_{11}J_{12} & J_{12}^2 \\ 0 & 0 & 0 & J_{11}J_{21} & J_{12}J_{21} + J_{11}J_{22} & J_{12}J_{22} \\ 0 & 0 & 0 & J_{21}^2 & 2J_{21}J_{22} & J_{22}^2 \end{bmatrix} \quad (2.17c)$$

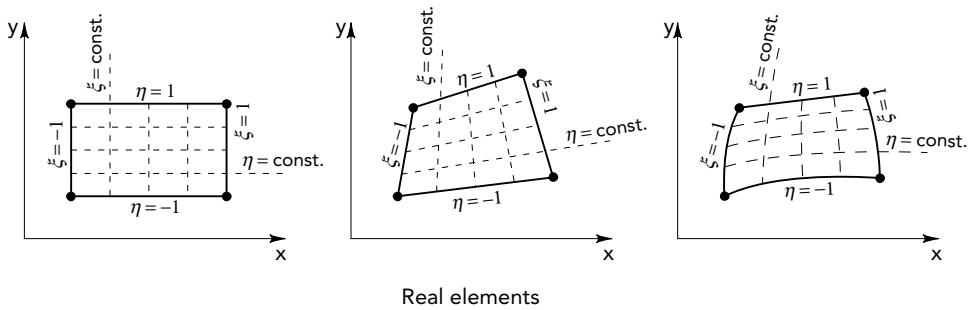
2.4 Quadrilateral elements (two dimensions)

2.4.1 SYSTEMS OF COORDINATES

For all quadrilateral elements, we use the following reference element:



The coordinates (ξ, η) can be interpreted as curvilinear coordinates for the real element:



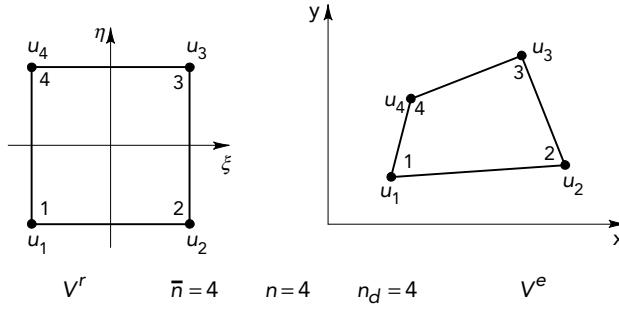
By convention, nodes are numbered in the positive sense of rotation on the oriented surface of the element, starting with a corner node.

The variables beneath the integral symbol are altered as follows:

$$\int \int f(x, y) dx dy = \int_{-1}^1 \int_{-1}^1 f(\xi, \eta) |J| d\xi d\eta$$

2.4.2 BILINEAR ELEMENT (quadrilateral, 4 nodes, C^0)

This element is described in Examples 1.16 and 1.18.



The geometric nodes and the interpolation nodes are identical; the element is isoparametric:

$$\langle P \rangle = \langle 1 \ \xi \ \eta \ \xi\eta \rangle \quad (2.18a)$$

$$[P_n] = \begin{bmatrix} \langle P(\xi_1) \rangle \\ \langle P(\xi_2) \rangle \\ \langle P(\xi_3) \rangle \\ \langle P(\xi_4) \rangle \end{bmatrix} \quad [P_n]^{-1} = \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \quad (2.18b)$$

	$\frac{1}{c}\{N\}$	$\frac{1}{c}\{\partial N/\partial \xi\}$	$\frac{1}{c}\{\partial N/\partial \eta\}$	c
1	$(1-\xi)(1-\eta)$	$-1+\eta$	$-1+\xi$	
2	$(1+\xi)(1-\eta)$	$1-\eta$	$-1-\xi$	
3	$(1+\xi)(1+\eta)$	$1+\eta$	$1+\xi$	$1/4$
4	$(1-\xi)(1+\eta)$	$-1-\eta$	$1-\xi$	

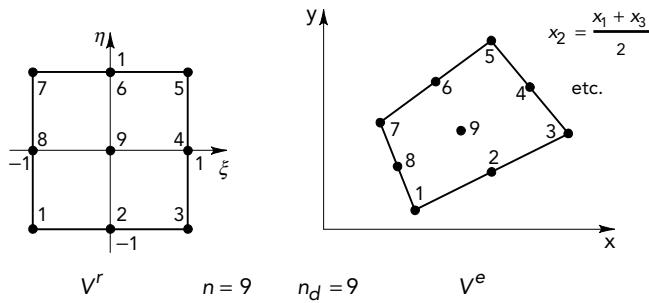
An explicit expression for the Jacobian matrix $[J]$ can be found in Example 1.18. The truncation error can be obtained using equation (1.68), extended to two-dimensional spaces.

2.4.3 HIGH-PRECISION LAGRANGIAN ELEMENTS

For the elements discussed in this section, we will use the geometric nodes, the functions \bar{N} , and the Jacobian matrix of the bilinear element from section 2.4.2. Curvilinear elements will be described in section 2.4.3.5.

2.4.3.1 Complete quadratic element (quadrilateral, nine nodes, C^0)

This element uses a one-dimensional quadratic Lagrangian approximation in both directions ξ and η . It is often used in fluid mechanics.



$$\begin{aligned} \langle P \rangle &= \langle \xi^i \eta^j; i = 0, 1, 2; j = 0, 1, 2 \rangle \\ &= \langle 1 \ \xi \ \eta \ \xi^2 \ \xi\eta \ \eta^2 \ \xi^2\eta \ \xi\eta^2 \ \xi^2\eta^2 \rangle. \end{aligned} \quad (2.19)$$

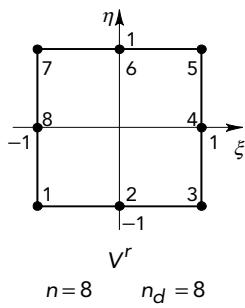
The functions $N(\xi, \eta)$ of this element, which are given below, are the products of the functions $N(\xi)$ and $N(\eta)$ corresponding to the one-dimensional, three-node Lagrangian element discussed in section 2.2.2.1.

The continuity of this element is of type C^0 .

	$\{N\}$	$\{\partial N/\partial\xi\}$	$\{\partial N/\partial\eta\}$
1	$\frac{(1-\xi)(1-\eta)\xi\eta}{4}$	$\frac{(1-2\xi)(1-\eta)\eta}{4}$	$\frac{(1-\xi)(1-2\eta)\xi}{4}$
2	$\frac{-(1-\xi^2)(1-\eta)\eta}{2}$	$(1-\eta)\xi\eta$	$\frac{-(1-\xi^2)(1-2\eta)}{2}$
3	$\frac{-(1+\xi)(1-\eta)\xi\eta}{4}$	$\frac{-(1+2\xi)(1-\eta)\eta}{4}$	$\frac{-(1+\xi)(1-2\eta)\xi}{4}$
4	$\frac{(1+\xi)(1-\eta^2)\xi}{2}$	$\frac{(1+2\xi)(1-\eta^2)}{2}$	$-(1+\xi)\xi\eta$
5	$\frac{(1+\xi)(1+\eta)\xi\eta}{4}$	$\frac{(1+2\xi)(1+\eta)\eta}{4}$	$\frac{(1+\xi)(1+2\eta)\xi}{4}$
6	$\frac{(1-\xi^2)(1+\eta)\eta}{2}$	$-(1+\eta)\xi\eta$	$\frac{(1-\xi^2)(1+2\eta)}{2}$
7	$\frac{-(1-\xi)(1+\eta)\xi\eta}{4}$	$\frac{-(1-2\xi)(1+\eta)\eta}{4}$	$\frac{-(1-\xi)(1+2\eta)\xi}{4}$
8	$\frac{-(1-\xi)(1-\eta^2)\xi}{2}$	$\frac{-(1-2\xi)(1-\eta^2)}{2}$	$(1-\xi)\xi\eta$
9	$(1-\xi^2)(1-\eta^2)$	$-2(1-\eta^2)\xi$	$-2(1-\xi^2)\eta$

2.4.3.2 Incomplete quadratic element (quadrilateral, eight nodes, C^0)

This element, very frequently used in its isoparametric form, has only eight nodes, which are situated on the boundary of the element:



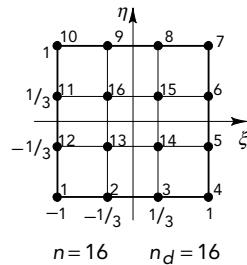
$$\langle P \rangle = \langle 1 \ \xi \ \eta \ \xi^2 \ \xi\eta \ \eta^2 \ \xi^2\eta \ \xi\eta^2 \rangle. \quad (2.20)$$

This element also exhibits type C^0 continuity.

	$\{N\}$	$\{\partial N / \partial \xi\}$	$\{\partial N / \partial \eta\}$
1	$\frac{-(1-\xi)(1-\eta)(1+\xi+\eta)}{4}$	$\frac{(1-\eta)(2\xi+\eta)}{4}$	$\frac{(1-\xi)(\xi+2\eta)}{4}$
2	$\frac{(1-\xi^2)(1-\eta)}{2}$	$-(1-\eta)\xi$	$\frac{-(1-\xi^2)}{2}$
3	$\frac{-(1+\xi)(1-\eta)(1-\xi+\eta)}{4}$	$\frac{(1-\eta)(2\xi-\eta)}{4}$	$\frac{-(1+\xi)(\xi-2\eta)}{4}$
4	$\frac{(1+\xi)(1-\eta^2)}{2}$	$\frac{(1-\eta^2)}{2}$	$-(1+\xi)\eta$
5	$\frac{-(1+\xi)(1+\eta)(1-\xi-\eta)}{4}$	$\frac{(1+\eta)(2\xi+\eta)}{4}$	$\frac{(1+\xi)(\xi+2\eta)}{4}$
6	$\frac{(1-\xi^2)(1+\eta)}{2}$	$-(1+\eta)\xi$	$\frac{(1-\xi^2)}{2}$
7	$\frac{-(1-\xi)(1+\eta)(1+\xi-\eta)}{4}$	$\frac{(1+\eta)(2\xi-\eta)}{4}$	$\frac{-(1-\xi)(\xi-2\eta)}{4}$
8	$\frac{(1-\xi)(1-\eta^2)}{2}$	$\frac{-(1-\eta^2)}{2}$	$-(1-\xi)\eta$

2.4.3.3 Complete cubic element (quadrilateral, 16 nodes, C^0)

This is a Lagrangian element with four nodes in directions ξ and η .



$$\langle P \rangle = \langle \xi^i \eta^j; i = 0, 1, 2, 3; j = 0, 1, 2, 3 \rangle \quad (2.21)$$

	$\{N(\xi, \eta)\}$	$\{\partial N(\xi, \eta)/\partial \xi\}$	$\{\partial N(\xi, \eta)/\partial \eta\}$
1	$N_1(\xi) \cdot N_1(\eta)$	$B_1(\xi) \cdot N_1(\eta)$	$N_1(\xi) \cdot B_1(\eta)$
2	$N_2(\xi) \cdot N_1(\eta)$	$B_2(\xi) \cdot N_1(\eta)$	$N_2(\xi) \cdot B_1(\eta)$
3	$N_3(\xi) \cdot N_1(\eta)$	$B_3(\xi) \cdot N_1(\eta)$	$N_3(\xi) \cdot B_1(\eta)$
4	$N_4(\xi) \cdot N_1(\eta)$	$B_4(\xi) \cdot N_1(\eta)$	$N_4(\xi) \cdot B_1(\eta)$
5	$N_4(\xi) \cdot N_2(\eta)$	$B_4(\xi) \cdot N_2(\eta)$	$N_4(\xi) \cdot B_2(\eta)$
6	$N_4(\xi) \cdot N_3(\eta)$	$B_4(\xi) \cdot N_3(\eta)$	$N_4(\xi) \cdot B_3(\eta)$
7	$N_4(\xi) \cdot N_4(\eta)$	$B_4(\xi) \cdot N_4(\eta)$	$N_4(\xi) \cdot B_4(\eta)$
8	$N_3(\xi) \cdot N_4(\eta)$	$B_3(\xi) \cdot N_4(\eta)$	$N_3(\xi) \cdot B_4(\eta)$
9	$N_2(\xi) \cdot N_4(\eta)$	$B_2(\xi) \cdot N_4(\eta)$	$N_2(\xi) \cdot B_4(\eta)$
10	$N_1(\xi) \cdot N_4(\eta)$	$B_1(\xi) \cdot N_4(\eta)$	$N_1(\xi) \cdot B_4(\eta)$
11	$N_1(\xi) \cdot N_3(\eta)$	$B_1(\xi) \cdot N_3(\eta)$	$N_1(\xi) \cdot B_3(\eta)$
12	$N_1(\xi) \cdot N_2(\eta)$	$B_1(\xi) \cdot N_2(\eta)$	$N_1(\xi) \cdot B_2(\eta)$
13	$N_2(\xi) \cdot N_2(\eta)$	$B_2(\xi) \cdot N_2(\eta)$	$N_2(\xi) \cdot B_2(\eta)$
14	$N_3(\xi) \cdot N_2(\eta)$	$B_3(\xi) \cdot N_2(\eta)$	$N_3(\xi) \cdot B_2(\eta)$
15	$N_3(\xi) \cdot N_3(\eta)$	$B_3(\xi) \cdot N_3(\eta)$	$N_3(\xi) \cdot B_3(\eta)$
16	$N_2(\xi) \cdot N_3(\eta)$	$B_2(\xi) \cdot N_3(\eta)$	$N_2(\xi) \cdot B_3(\eta)$

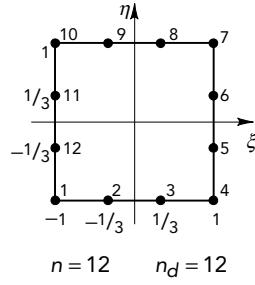
where
$$\begin{aligned} \langle N_i(\xi) \rangle &= \frac{1}{16} \langle -(1-\xi)(1-9\xi^2); 9(1-\xi^2)(1-3\xi); \\ &\quad 9(1-\xi^2)(1+3\xi); -(1+\xi)(1-9\xi^2) \rangle \\ &= \langle N_1(\xi) \quad N_2(\xi) \quad N_3(\xi) \quad N_4(\xi) \rangle \end{aligned}$$

$$\begin{aligned} \langle B_i(\xi) \rangle &= \frac{1}{16} \langle 1+18\xi-27\xi^2; -27-18\xi+81\xi^2; \\ &\quad 27-18\xi-81\xi^2; -1+18\xi+27\xi^2 \rangle \\ &= \langle B_1(\xi) \quad B_2(\xi) \quad B_3(\xi) \quad B_4(\xi) \rangle \end{aligned}$$

We can construct a general Lagrangian element with $n \times n$ nodes by using the $(n-1)$ -order Lagrange polynomials given in section 2.2.2.3, in both directions ξ and η .

2.4.3.4 Incomplete cubic element

This element, often used in its isoparametric form, has 12 nodes on its boundary.



$$\langle P \rangle = \langle 1 \ \xi \ \eta \ \xi^2 \ \xi\eta \ \eta^2 \ \xi^3 \ \xi^2\eta \ \xi\eta^2 \ \eta^3 \ \xi^3\eta \ \xi\eta^3 \rangle$$

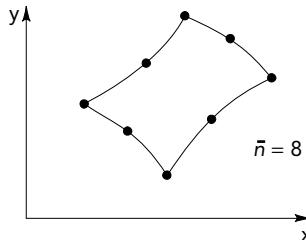
$\frac{1}{c} \{N\}$	$\frac{1}{c} \{\partial N / \partial \xi\}$	$\frac{1}{c} \{\partial N / \partial \eta\}$	c
1 $(1 - \xi)(1 - \eta)\lambda$	$(1 - \eta) \left(\frac{10}{9} + 2\xi - 3\xi^2 - \eta^2 \right)$	$(1 - \xi) \left(\frac{10}{9} + 2\eta - \xi^2 - 3\eta^2 \right)$	
2 $(1 - 3\xi)(1 - \xi^2)(1 - \eta)$	$(1 - \eta)(-3 - 2\xi + 9\xi^2)$	$(-1 + \xi^2)(1 - 3\xi)$	
3 $(1 + 3\xi)(1 - \xi^2)(1 - \eta)$	$(1 - \eta)(3 - 2\xi - 9\xi^2)$	$(-1 + \xi^2)(1 + 3\xi)$	
4 $(1 + \xi)(1 - \eta)\lambda$	$(1 - \eta) \left(-\frac{10}{9} + 2\xi + 3\xi^2 + \eta^2 \right)$	$(1 + \xi) \left(\frac{10}{9} + 2\eta - \xi^2 - 3\eta^2 \right)$	
5 $(1 + \xi)(1 - 3\eta)(1 - \eta^2)$	$(1 - \eta^2)(1 - 3\eta)$	$(1 + \xi)(-3 - 2\eta + 9\eta^2)$	
6 $(1 + \xi)(1 + 3\eta)(1 - \eta^2)$	$(1 - \eta^2)(1 + 3\eta)$	$(1 + \xi)(3 - 2\eta - 9\eta^2)$	9/32
7 $(1 + \xi)(1 + \eta)\lambda$	$(1 + \eta) \left(-\frac{10}{9} + 2\xi + 3\xi^2 + \eta^2 \right)$	$(1 + \xi) \left(-\frac{10}{9} + 2\eta + \xi^2 + 3\eta^2 \right)$	
8 $(1 + 3\xi)(1 - \xi^2)(1 + \eta)$	$(1 + \eta)(3 - 2\xi - 9\xi^2)$	$(1 - \xi^2)(1 + 3\xi)$	
9 $(1 - 3\xi)(1 - \xi^2)(1 + \eta)$	$(1 + \eta)(-3 - 2\xi + 9\xi^2)$	$(1 - \xi^2)(1 - 3\xi)$	
10 $(1 - \xi)(1 + \eta)\lambda$	$(1 + \eta) \left(\frac{10}{9} + 2\xi - 3\xi^2 - \eta^2 \right)$	$(1 - \xi) \left(-\frac{10}{9} + 2\eta + \xi^2 + 3\eta^2 \right)$	
11 $(1 - \xi)(1 + 3\eta)(1 - \eta^2)$	$(-1 + \eta^2)(1 + 3\eta)$	$(1 - \xi)(3 - 2\eta - 9\eta^2)$	
12 $(1 - \xi)(1 - 3\eta)(1 - \eta^2)$	$(-1 + \eta^2)(1 - 3\eta)$	$(1 - \xi)(-3 - 2\eta + 9\eta^2)$	

$$\lambda = \left(-\frac{10}{9} + \xi^2 + \eta^2 \right). \quad (2.22)$$

2.4.3.5 Curvilinear elements

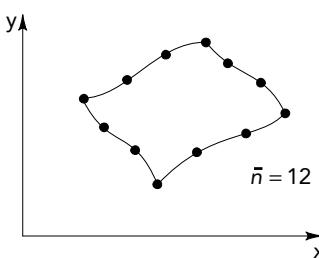
We can construct curvilinear quadrilaterals by increasing the number of geometric nodes on the boundary of the element.

a) Quadratic-sided element



The functions \bar{N} are identical to the functions N from section 2.4.3.2.

b) Cubic-sided element

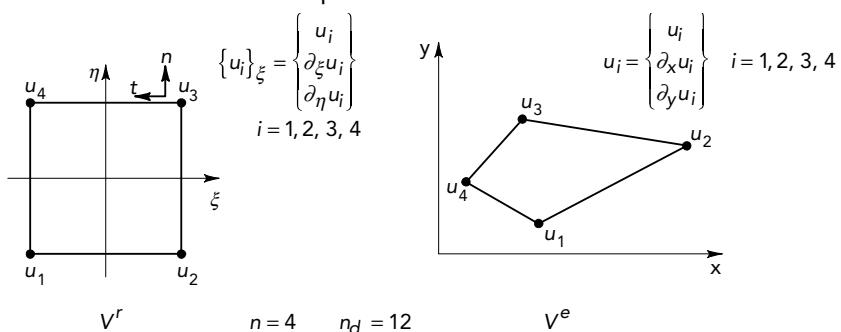


The functions \bar{N} are identical to the functions N from section 2.4.3.4.

2.4.4 HIGH-PRECISION HERMITE ELEMENT

The geometry of the element can again be linear, quadratic or cubic, as is the case for Lagrangian elements.

2.4.4.1 Cubic element (quadrilateral, four nodes, semi- C^1)



$$\langle P \rangle = \langle 1 \ \xi \ \eta \ \xi^2 \ \xi\eta \ \eta^2 \ \xi^3 \ \xi^2\eta \ \xi\eta^2 \ \eta^3 \ \xi^3\eta \ \xi\eta^3 \rangle \quad (2.23a)$$

	$\frac{1}{e} \{N\}$	$\frac{1}{e} \{\partial N / \partial \xi\}$	$\frac{1}{e} \{\partial N / \partial \eta\}$	e
Node 1	1 2 3	$a(\alpha - \xi - \eta)$ $a(1 - \xi^2)$ $a(1 - \eta^2)$	$(1 - \eta)(-3 + 3\xi^2 + \eta^2 + \eta)$ $-a(1 + 3\xi)$ $(-1 + \eta)(1 - \eta^2)$	$(1 - \xi)(-3 + \xi^2 + 3\eta^2 + \xi)$ $(-1 + \xi)(1 - \xi^2)$ $-a(1 + 3\eta)$
	4 5 6	$b(\alpha + \xi - \eta)$ $-b(1 - \xi^2)$ $b(1 - \eta^2)$	$(1 - \eta)(3 - 3\xi^2 - \eta^2 - \eta)$ $-b(1 - 3\xi)$ $(1 - \eta)(1 - \eta^2)$	
	7 8 9	$c(\alpha + \xi + \eta)$ $-c(1 - \xi^2)$ $-c(1 - \eta^2)$	$(1 + \eta)(3 - 3\xi^2 - \eta^2 + \eta)$ $-c(1 - 3\xi)$ $(-1 - \eta)(1 - \eta^2)$	
Node 4	10 11 12	$d(\alpha - \xi + \eta)$ $d(1 - \xi^2)$ $-d(1 - \eta^2)$	$(1 + \eta)(-3 + 3\xi^2 + \eta^2 - \eta)$ $-d(1 + 3\xi)$ $(1 + \eta)(1 - \eta^2)$	$(1 - \xi)(3 - \xi - \xi^2 - 3\eta^2)$ $(1 - \xi)(1 - \xi^2)$ $-d(1 - 3\eta)$

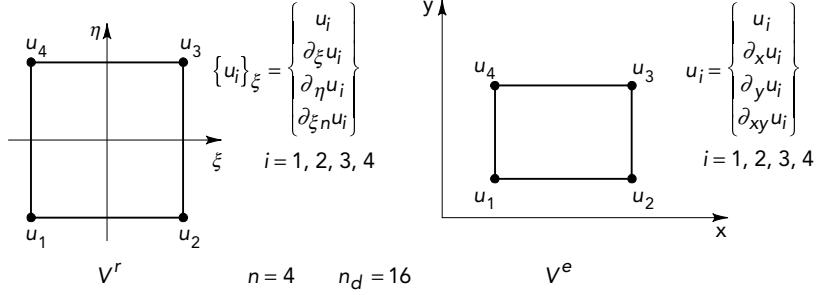
where $a = (1 - \xi)(1 - \eta)$; $b = (1 + \xi)(1 - \eta)$; $c = (1 + \xi)(1 + \eta)$;
 $d = (1 - \xi)(1 + \eta)$; $\alpha = 2 - \xi^2 - \eta^2$.

The functions N correspond to the nodal variables $\{u_i\}_\xi$. The transformation between the two sets of nodal variables is thus written as:

$$\begin{aligned} \{u_n\}_\xi &= \begin{Bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \mathbf{u}_4 \end{Bmatrix}_\xi \quad \{u_n\} = \begin{Bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_3 \\ \mathbf{u}_4 \end{Bmatrix}_x \\ \{u_n\}_\xi &= \begin{bmatrix} [T_1] & & & \\ & [T_2] & & \\ & & [T_3] & \\ & & & [T_4] \end{bmatrix} \{u_n\} \text{ where } [T_i] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & [J(\xi)] & \\ 0 & & \end{bmatrix}. \quad (2.23b) \end{aligned}$$

On the sides, u and $\partial u / \partial t$ are continuous, but $\partial u / \partial n$ is not. However, the values of $\partial u / \partial x$ and $\partial u / \partial y$ at each node are identical for all elements linked to these nodes.

2.4.4.2 Rectangular element (rectangle, four nodes, C^1)



This element exhibits complete C^1 continuity. However, it must be rectangular and parallel to the global system of reference because, in general, the transformation of the nodal variable $\frac{\partial^2}{\partial \xi \partial \eta} u_i$ will involve the three second derivatives in \mathbf{x} :

$$\langle P \rangle = \langle \xi^i \eta^j; \quad i = 0, 1, 2, 3; \quad j = 0, 1, 2, 3 \rangle. \quad (2.24)$$

We can easily get the functions N by using the one-dimensional Hermite approximation (see section 2.2.3.1) in both directions.

	$\{N(\xi, \eta)\}$	$\{\partial N(\xi, \eta)/\partial \xi\}$	$\{\partial N(\xi, \eta)/\partial \eta\}$
Node 1	$\begin{cases} 1 & N_1(\xi) \cdot N_1(\eta) \\ 2 & N_2(\xi) \cdot N_1(\eta) \\ 3 & N_1(\xi) \cdot N_2(\eta) \\ 4 & N_2(\xi) \cdot N_2(\eta) \end{cases}$	$\begin{cases} B_1(\xi) \cdot N_1(\eta) \\ B_2(\xi) \cdot N_1(\eta) \\ B_1(\xi) \cdot N_2(\eta) \\ B_2(\xi) \cdot N_2(\eta) \end{cases}$	$\begin{cases} N_1(\xi) \cdot B_1(\eta) \\ N_2(\xi) \cdot B_1(\eta) \\ N_1(\xi) \cdot B_2(\eta) \\ N_2(\xi) \cdot B_2(\eta) \end{cases}$
	$\begin{cases} 5 & N_3(\xi) \cdot N_1(\eta) \\ 6 & N_4(\xi) \cdot N_1(\eta) \\ 7 & N_3(\xi) \cdot N_2(\eta) \\ 8 & N_4(\xi) \cdot N_2(\eta) \end{cases}$	$\begin{cases} B_3(\xi) \cdot N_1(\eta) \\ B_4(\xi) \cdot N_1(\eta) \\ B_3(\xi) \cdot N_2(\eta) \\ B_4(\xi) \cdot N_2(\eta) \end{cases}$	$\begin{cases} N_3(\xi) \cdot B_1(\eta) \\ N_4(\xi) \cdot B_1(\eta) \\ N_3(\xi) \cdot B_2(\eta) \\ N_4(\xi) \cdot B_2(\eta) \end{cases}$
	$\begin{cases} 9 & N_3(\xi) \cdot N_3(\eta) \\ 10 & N_4(\xi) \cdot N_3(\eta) \\ 11 & N_3(\xi) \cdot N_4(\eta) \\ 12 & N_4(\xi) \cdot N_4(\eta) \end{cases}$	$\begin{cases} B_3(\xi) \cdot N_3(\eta) \\ B_4(\xi) \cdot N_3(\eta) \\ B_3(\xi) \cdot N_4(\eta) \\ B_4(\xi) \cdot N_4(\eta) \end{cases}$	$\begin{cases} N_3(\xi) \cdot B_3(\eta) \\ N_4(\xi) \cdot B_3(\eta) \\ N_3(\xi) \cdot B_4(\eta) \\ N_4(\xi) \cdot B_4(\eta) \end{cases}$
	$\begin{cases} 13 & N_1(\xi) \cdot N_3(\eta) \\ 14 & N_2(\xi) \cdot N_3(\eta) \\ 15 & N_1(\xi) \cdot N_4(\eta) \\ 16 & N_2(\xi) \cdot N_4(\eta) \end{cases}$	$\begin{cases} B_1(\xi) \cdot N_3(\eta) \\ B_2(\xi) \cdot N_3(\eta) \\ B_1(\xi) \cdot N_4(\eta) \\ B_2(\xi) \cdot N_4(\eta) \end{cases}$	$\begin{cases} N_1(\xi) \cdot B_3(\eta) \\ N_2(\xi) \cdot B_3(\eta) \\ N_1(\xi) \cdot B_4(\eta) \\ N_2(\xi) \cdot B_4(\eta) \end{cases}$

$$\begin{aligned} \langle N_i(\xi) \rangle &= \frac{1}{4} \langle (1-\xi)^2 (2+\xi); (1-\xi^2)(1-\xi); (1+\xi)^2(2-\xi); \\ &\quad (-1+\xi^2)(1+\xi) \rangle \\ &= \langle N_1(\xi) \quad N_2(\xi) \quad N_3(\xi) \quad N_4(\xi) \rangle \end{aligned}$$

$$\begin{aligned}\langle B_i(\xi) \rangle &= \frac{1}{4} \langle -3(1-\xi^2); (-1+\xi)(1+3\xi); 3(1-\xi^2); \\ &\quad (-1-\xi)(1-3\xi) \rangle \\ &= \langle B_1(\xi) \ B_2(\xi) \ B_3(\xi) \ B_4(\xi) \rangle.\end{aligned}$$

The transformation of the nodal variables is similar to (2.23b).

$$[T_i] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{a}{2} & 0 & 0 \\ 0 & 0 & \frac{b}{2} & 0 \\ 0 & 0 & 0 & \frac{ab}{4} \end{bmatrix}$$

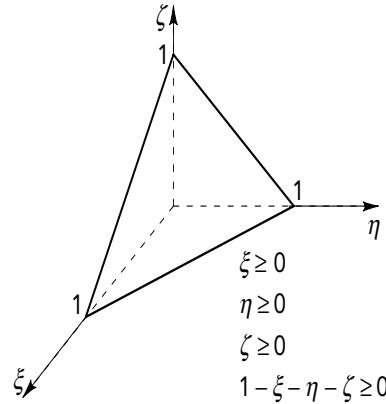
where

$$a = x_2 - x_1, \quad b = y_4 - y_1.$$

2.5 Tetrahedral elements (three dimensions)

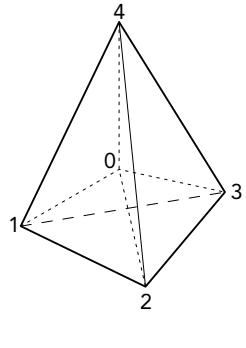
2.5.1 SYSTEMS OF COORDINATES

For all tetrahedral-shaped elements, we use the following reference elements:



Just like for triangles, the coordinates (ξ, η, ζ) can be interpreted as curvilinear coordinates for the real element. The surface $\xi = \text{constant}$ (or $\eta = \text{constant}$ or $\zeta = \text{constant}$) is plane, parallel to the faces of the element, in the case of an element with straight sides.

The barycentric coordinates L_1, L_2, L_3 and L_4 are sometimes used to represent a point 0 for a straight-sided tetrahedron.



$$\begin{aligned}
 L_1 &= \frac{V_1}{V} \\
 L_2 &= \frac{V_2}{V} \\
 L_3 &= \frac{V_3}{V} \\
 L_4 &= \frac{V_4}{V}
 \end{aligned} \tag{2.25a}$$

$V = V_1 + V_2 + V_3 + V_4$; $L_1 + L_2 + L_3 + L_4 = 1$; V_1, V_2, V_3 and V_4 are respectively the volumes of the tetrahedra:

0-2-3-4, 0-1-3-4, 0-1-2-4 and 0-1-2-3.

V is the volume of the whole tetrahedron.

The barycentric coordinates are linked to the coordinates ξ, η and ζ by:

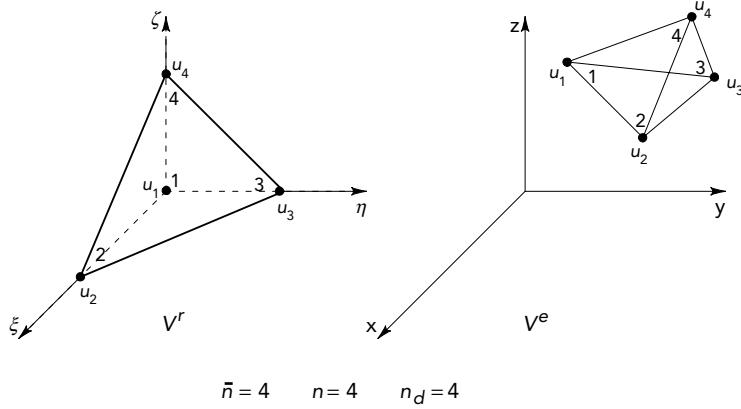
$$\begin{aligned}
 L_1 &\equiv 1 - \xi - \eta - \zeta \\
 L_2 &\equiv \xi \\
 L_3 &\equiv \eta \\
 L_4 &\equiv \zeta.
 \end{aligned} \tag{2.25b}$$

Note that the order of the numbering must be consistent between the reference element and the real element. Here the first three nodes are taken in the positive sense of rotation about a unit normally oriented toward the interior of the element.

The variables beneath the integral symbol are altered as follows:

$$\int_x \int_y \int_z f(x, y, z) dx dy dz = \int_0^{1-\zeta} \int_0^{1-\zeta-\eta} \int_0^1 f(\xi, \eta, \zeta) |J| d\xi d\eta d\zeta$$

2.5.2 LINEAR ELEMENT (tetrahedron, four nodes, C^0)



The four geometric nodes are identical to the interpolation nodes (i.e. the element is isoparametric):

$$\langle P \rangle = \langle 1 \ \xi \ \eta \ \zeta \rangle. \quad (2.26a)$$

$$[P_n] = \begin{bmatrix} \langle P(\xi_1) \rangle \\ \langle P(\xi_2) \rangle \\ \langle P(\xi_3) \rangle \\ \langle P(\xi_4) \rangle \end{bmatrix}; \quad [P_n]^{-1} = \frac{1}{4} \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix} \quad (2.26b)$$

$\{N\}$	$\{\partial N / \partial \xi\}$	$\{\partial N / \partial \eta\}$	$\{\partial N / \partial \zeta\}$
1 $1 - \xi - \eta - \zeta$	-1	-1	-1
2 ξ	1	0	0
3 η	0	1	0
4 ζ	0	0	1

The Jacobian matrix is thus expressed:

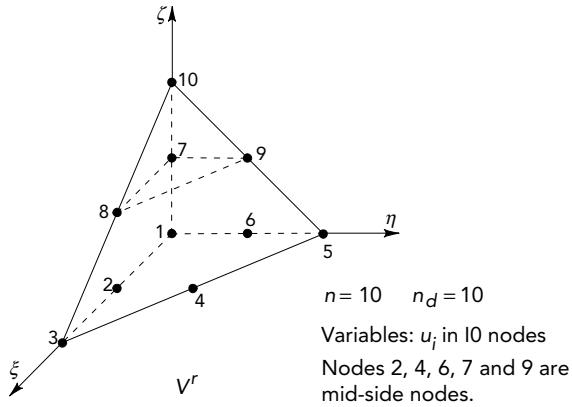
$$[J] = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \\ x_4 - x_1 & y_4 - y_1 & z_4 - z_1 \end{bmatrix} \quad (2.26c)$$

$$\det(J) = 6V$$

where V is the volume of the real element.

2.5.3 HIGH-PRECISION LAGRANGIAN ELEMENTS (continuity C^0)

2.5.3.1 Complete quadratic element (tetrahedron, 10 nodes, C^0)



We keep the geometry of the linear tetrahedral element. The $\langle P \rangle$ is a complete quadratic polynomial basis:

$$\langle P \rangle = \langle 1 \ \xi \ \eta \ \zeta \ \xi^2 \ \xi\eta \ \eta^2 \ \eta\zeta \ \zeta^2 \ \xi\zeta \rangle. \quad (2.27)$$

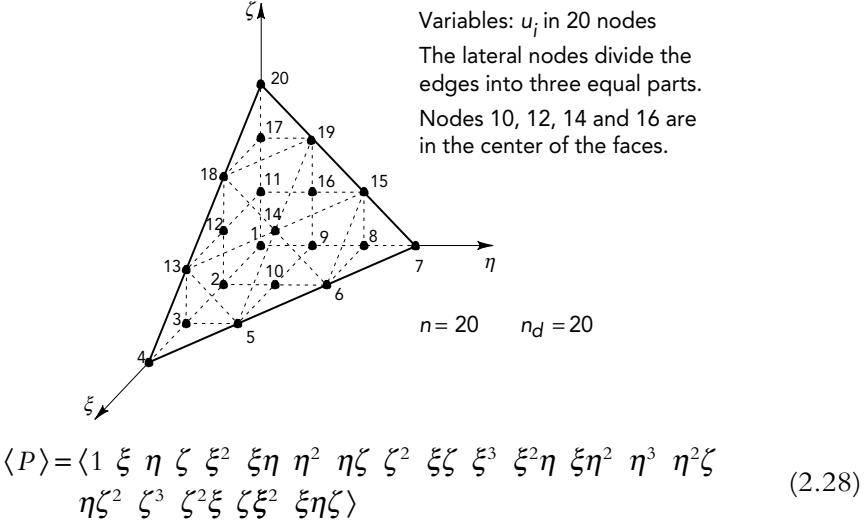
We can easily calculate the functions N based on those of the quadratic triangle (section 2.3.3.1).

	$\{N\}$	$\{\partial N / \partial \xi\}$	$\{\partial N / \partial \eta\}$	$\{\partial N / \partial \zeta\}$
1	$-\lambda(1 - 2\lambda)$	$1 - 4\lambda$	$1 - 4\lambda$	$1 - 4\lambda$
2	$4\xi\lambda$	$4(\lambda - \xi)$	-4ξ	-4ξ
3	$-\xi(1 - 2\xi)$	$-1 + 4\xi$	0	0
4	$4\xi\eta$	4η	4ξ	0
5	$-\eta(1 - 2\eta)$	0	$-1 + 4\eta$	0
6	$4\eta\lambda$	-4η	$4(\lambda - \eta)$	-4η
7	$4\zeta\lambda$	-4ζ	-4ζ	$4(\lambda - \zeta)$
8	$4\xi\zeta$	4ζ	0	4ξ
9	$4\eta\zeta$	0	4ζ	4η
10	$-\zeta(1 - 2\zeta)$	0	0	$-1 + 4\zeta$

where

$$\lambda = 1 - \xi - \eta - \zeta.$$

2.5.3.2 Complete cubic element (tetrahedron, 20 nodes, C^0)



The $\langle P \rangle$ is a complete cubic polynomial basis in ξ, η and ζ .

The interpolation functions N are obtained from the functions given in section 2.4.3.3, replacing λ with $1 - \xi - \eta - \zeta$: the functions N_1 to N_{10} are thus identical. The functions $N_{11}, N_{12}, N_{13}, N_{17}, N_{18}$ and N_{20} are expressed like the functions $N_9, N_{10}, N_5, N_8, N_6$ and N_7 in section 2.3.3.3, with η being replaced by ζ . The functions N_{15}, N_{16} and N_{19} are expressed like the functions N_5, N_{10} and N_6 in section 2.3.3.3, with ξ being replaced by ζ . The function N_{14} is $27 \xi\eta\zeta$.

The interpolation functions are as follows:

- For the four peak nodes – 1, 4, 7 and 20:

$$N_i = \frac{1}{2} (3a - 1)(3a - 2)a, \text{ where } a = \lambda, \xi, \eta \text{ and } \zeta, \text{ respectively.}$$

- For the 12 nodes in between the segments:

$$N_i = \frac{9}{2} ab(3a - 1).$$

Node	2	3	5	6	8	9	17	11	15	19	13	18
<i>a</i>	λ	ξ	ξ	η	η	λ	ζ	λ	η	ζ	ξ	ζ
<i>b</i>	ξ	λ	η	ξ	λ	η	λ	ζ	ζ	η	ζ	ξ

- For the four nodes in the center of the faces – 10, 16, 12 and 14:

$$N_i = 27\lambda\xi\eta; 27\lambda\zeta\eta; 27\lambda\xi\zeta; 27\zeta\xi\eta.$$

Remark

- We can eliminate nodes 10, 12, 14 and 16 from the faces using a technique similar to that in section 2.3.3.4 to obtain an element with 16 nodes.

2.5.3.3 Curvilinear elements

We can construct elements with incurvate faces using the functions \bar{N} from sections 2.5.3.1 and 2.5.3.2.

2.5.4 HIGH-PRECISION HERMITE ELEMENTS

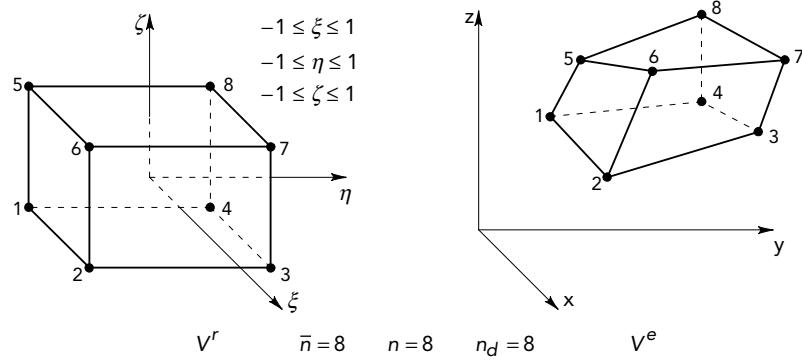
Just as for triangles, a semi- C^1 -type element uses the following nodal variables:

- $u_i \partial_\xi u_i \partial_\eta u_i \partial_\zeta u_i$ at the four peaks;
- u_i at the center of each face.

This element thus has eight nodes, 20 degrees of freedom, and uses a complete cubic polynomial basis. It is also possible to use an incomplete polynomial with 16 terms and avoid the nodes situated on the faces. For instance, we can eliminate the monomials $\xi^2\eta, \eta^2\zeta, \zeta^2\xi$ and $\xi\eta\zeta$.

2.6 Hexahedric elements (three dimensions)

2.6.1 TRILINEAR ELEMENT (hexahedron, eight nodes, C^0)



This element includes a variable u_i at each of its eight nodes. The geometric nodes are identical to the interpolation nodes.

$$\langle P \rangle = \langle 1 \ \xi \ \eta \ \zeta \ \xi\eta \ \eta\zeta \ \xi\zeta \ \eta\xi\zeta \rangle. \quad (2.29)$$

The functions N are the products of the functions N of the one-dimensional linear element.

	$\frac{1}{c} \{N\}$	$\frac{1}{c} \{\partial N / \partial \xi\}$	$\frac{1}{c} \{\partial N / \partial \eta\}$	$\frac{1}{c} \{\partial N / \partial \zeta\}$	c
1	$a_2 b_2 c_2$	$-b_2 c_2$	$-a_2 c_2$	$-a_2 b_2$	
2	$a_1 b_2 c_2$	$b_2 c_2$	$-a_1 c_2$	$-a_1 b_2$	
3	$a_1 b_1 c_2$	$b_1 c_2$	$a_1 c_2$	$-a_1 b_1$	
4	$a_2 b_1 c_2$	$-b_1 c_2$	$a_2 c_2$	$-a_2 b_1$	
5	$a_2 b_2 c_1$	$-b_2 c_1$	$-a_2 c_1$	$a_2 b_2$	$1/8$
6	$a_1 b_2 c_1$	$b_2 c_1$	$-a_1 c_1$	$a_1 b_2$	
7	$a_1 b_1 c_1$	$b_1 c_1$	$a_1 c_1$	$a_1 b_1$	
8	$a_2 b_1 c_1$	$-b_1 c_1$	$a_2 c_1$	$a_2 b_1$	

where

$$a_1 = 1 + \xi; \quad a_2 = 1 - \xi$$

$$b_1 = 1 + \eta; \quad b_2 = 1 - \eta$$

$$c_1 = 1 + \zeta; \quad c_2 = 1 - \zeta.$$

2.6.2 HIGH-PRECISION LAGRANGIAN ELEMENTS (continuity C^0)

For the elements discussed in this section, we will keep the functions \bar{N} from the previous element.

2.6.2.1 Complete quadratic element (hexahedron, 27 nodes, C^0)

This element uses a one-dimensional quadratic Lagrange approximation in all three directions ξ , η and ζ .

$$n = 27 \quad n_d = 27$$

$$\langle P \rangle = \langle \xi^i \eta^j \zeta^k; \quad i=0,1,2; \quad j=0,1,2; \quad k=0,1,2 \rangle. \quad (2.30a)$$

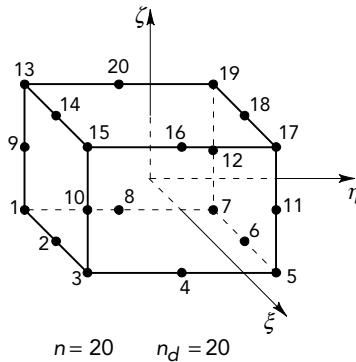
The coordinates ξ_i , η_i , ζ_i of the nodes comprise 27 triplets formed by combining the values $-1, 0$ and 1 . The functions N are of the form:

$$N(\xi, \eta, \zeta) = N(\xi) \cdot N(\eta) \cdot N(\zeta) \quad (2.30b)$$

where $N(\xi)$, $N(\eta)$ and $N(\zeta)$ are identical to the functions $N(\xi)$ given in section 2.2.2.1.

2.6.2.2 Incomplete quadratic element (hexahedron, 20 nodes, C^0)

This element is very frequently used, especially in its isoparametric form:



$$\begin{aligned} \langle P \rangle = & \langle 1 \xi \eta \zeta; \xi^2 \xi\eta \eta^2 \eta\zeta \zeta^2 \xi\zeta; \\ & \xi^2 \eta \xi\eta^2 \eta^2 \zeta \eta\zeta^2 \xi\xi^2 \xi^2 \xi\eta\zeta; \xi^2 \eta\zeta \xi\eta^2 \xi\eta\zeta^2 \rangle. \end{aligned} \quad (2.31)$$

The functions N_i and their derivatives are as follows:

— Corner nodes:

Node i	1	3	5	7	13	15	17	19
ξ_i	-1	1	1	-1	-1	1	1	-1
η_i	-1	-1	1	1	-1	-1	1	1
ζ_i	-1	-1	-1	-1	1	1	1	1

$$N_i = \frac{1}{8} (1 + \xi \xi_i) (1 + \eta \eta_i) (1 + \zeta \zeta_i) (-2 + \xi \xi_i + \eta \eta_i + \zeta \zeta_i)$$

$$\frac{\partial N_i}{\partial \xi} = \frac{1}{8} \xi_i (1 + \eta \eta_i) (1 + \zeta \zeta_i) (-1 + 2 \xi \xi_i + \eta \eta_i + \zeta \zeta_i)$$

$$\frac{\partial N_i}{\partial \eta} = \frac{1}{8} \eta_i (1 + \xi \xi_i) (1 + \zeta \zeta_i) (-1 + \xi \xi_i + 2 \eta \eta_i + \zeta \zeta_i)$$

$$\frac{\partial N_i}{\partial \zeta} = \frac{1}{8} \zeta_i (1 + \xi \xi_i) (1 + \eta \eta_i) (-1 + \xi \xi_i + \eta \eta_i + 2 \zeta \zeta_i).$$

— Nodes on the sides parallel to the axis ξ :

Node i	2	6	14	18
$\xi_i = 0 ; \eta_i$	-1	1	-1	1

$$N_i = \frac{1}{4} (1 - \xi^2) (1 + \eta \eta_i) (1 + \zeta \zeta_i)$$

$$\frac{\partial N_i}{\partial \xi} = -\frac{1}{2} \xi (1 + \eta \eta_i) (1 + \zeta \zeta_i)$$

$$\frac{\partial N_i}{\partial \eta} = \frac{1}{4} \eta_i (1 - \xi^2) (1 + \zeta \zeta_i)$$

$$\frac{\partial N_i}{\partial \zeta} = \frac{1}{4} \zeta_i (1 - \xi^2) (1 + \eta \eta_i).$$

— Nodes on the sides parallel to the axis η :

Node i	4	8	16	20
$\eta_i = 0$; $\begin{matrix} \xi_i \\ \zeta_i \end{matrix}$	1	-1	1	-1

$$N_i = \frac{1}{4} (1 + \xi \xi_i) (1 - \eta^2) (1 + \zeta \zeta_i)$$

$$\frac{\partial N_i}{\partial \xi} = \frac{1}{4} \xi_i (1 - \eta^2) (1 + \zeta \zeta_i)$$

$$\frac{\partial N_i}{\partial \eta} = -\frac{1}{2} \eta (1 + \xi \xi_i) (1 + \zeta \zeta_i)$$

$$\frac{\partial N_i}{\partial \zeta} = \frac{1}{4} \zeta_i (1 + \xi \xi_i) (1 - \eta^2).$$

— Nodes on the sides parallel to the axis ζ :

Node i	9	10	11	12
$\zeta_i = 0$; $\begin{matrix} \xi_i \\ \eta_i \end{matrix}$	-1	1	1	-1

$$N_i = \frac{1}{4} (1 + \xi \xi_i) (1 + \eta \eta_i) (1 - \zeta^2)$$

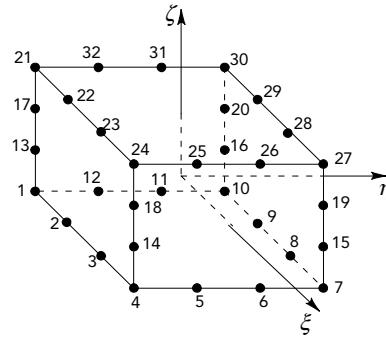
$$\frac{\partial N_i}{\partial \xi} = \frac{1}{4} \xi_i (1 + \eta \eta_i) (1 - \zeta^2)$$

$$\frac{\partial N_i}{\partial \eta} = \frac{1}{4} \eta_i (1 + \xi \xi_i) (1 - \zeta^2)$$

$$\frac{\partial N_i}{\partial \zeta} = \frac{1}{2} \zeta (1 + \xi \xi_i) (1 + \eta \eta_i)$$

2.6.2.3 Incomplete cubic element (hexahedron, 32 nodes, C^0)

This element has eight corner nodes and 24 mid-side nodes dividing each edge into three equal parts.



The $\langle P \rangle$ basis is a complete cubic polynomial basis (20 terms), with the addition of the following 12 terms:

$$\begin{aligned} & \xi^3\eta \quad \xi\eta^3 \quad \eta^3\xi \quad \eta\xi^3 \quad \xi\xi^3 \quad \xi^3\xi \\ & \xi^2\eta\xi \quad \xi\eta^2\xi \quad \xi\eta\xi^2 \\ & \xi^3\eta\xi \quad \xi\eta^3\xi \quad \xi\eta\xi^3. \end{aligned} \quad (2.32)$$

The functions and their derivatives are as follows:

— Corner nodes:

Node i	1	4	7	10	21	24	27	30
ξ_i	-1	1	1	-1	-1	1	1	-1
η_i	-1	-1	1	1	-1	-1	1	1
ζ_i	-1	-1	-1	-1	1	1	1	1

$$N_i = \frac{9}{64} (1 + \xi \xi_i) (1 + \eta \eta_i) (1 + \zeta \zeta_i) \left(-\frac{19}{9} + \xi^2 + \eta^2 + \zeta^2 \right)$$

$$\frac{\partial N_i}{\partial \xi} = \frac{9}{64} (1 + \eta \eta_i) (1 + \zeta \zeta_i) \left(\xi_i \left(-\frac{19}{9} + 3 \xi^2 + \eta^2 + \zeta^2 \right) + 2 \xi \right)$$

$$\frac{\partial N_i}{\partial \eta} = \frac{9}{64} (1 + \xi \xi_i) (1 + \zeta \zeta_i) \left(\eta_i \left(-\frac{19}{9} + \xi^2 + 3 \eta^2 + \zeta^2 \right) + 2 \eta \right)$$

$$\frac{\partial N_i}{\partial \zeta} = \frac{9}{64} (1 + \xi \xi_i) (1 + \eta \eta_i) \left(\zeta_i \left(-\frac{19}{9} + \xi^2 + \eta^2 + 3 \zeta^2 \right) + 2 \zeta \right).$$

— Nodes on the sides parallel to the axis ξ :

Node i	2	3	8	9	22	23	28	29
ξ_i	$-\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$-\frac{1}{3}$	$-\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$-\frac{1}{3}$
η_i	-1	-1	1	1	-1	-1	1	1
ζ_i	-1	-1	-1	-1	1	1	1	1

$$N_i = \frac{81}{64} (1 - \xi^2) \left(\frac{1}{9} + \xi \xi_i \right) (1 + \eta \eta_i) (1 + \zeta \zeta_i)$$

$$\frac{\partial N_i}{\partial \xi} = \frac{81}{64} (1 + \eta \eta_i) (1 + \zeta \zeta_i) \left(\xi_i - \frac{2 \xi}{9} - 3 \xi^2 \xi_i \right)$$

$$\frac{\partial N_i}{\partial \eta} = \frac{81}{64} \eta_i (1 - \xi^2) \left(\frac{1}{9} + \xi \xi_i \right) (1 + \zeta \zeta_i)$$

$$\frac{\partial N_i}{\partial \zeta} = \frac{81}{64} \zeta_i (1 - \xi^2) \left(\frac{1}{9} + \xi \xi_i \right) (1 + \eta \eta_i).$$

— Nodes on the sides parallel to the axis η :

Node i	5	6	11	12	25	26	31	32
ξ_i	1	1	-1	-1	1	1	-1	-1
η_i	$-\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$-\frac{1}{3}$	$-\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$-\frac{1}{3}$
ζ_i	-1	-1	-1	-1	1	1	1	1

$$N_i = \frac{81}{64} (1 + \xi \xi_i) (1 - \eta^2) \left(\frac{1}{9} + \eta \eta_i \right) (1 + \zeta \zeta_i)$$

$$\frac{\partial N_i}{\partial \xi} = \frac{81}{64} \xi_i (1 - \eta^2) \left(\frac{1}{9} + \eta \eta_i \right) (1 + \zeta \zeta_i)$$

$$\frac{\partial N_i}{\partial \eta} = \frac{81}{64} (1 + \xi \xi_i) (1 + \zeta \zeta_i) \left(\eta_i - \frac{2 \eta}{9} - 3 \eta^2 \eta_i \right)$$

$$\frac{\partial N_i}{\partial \zeta} = \frac{81}{64} \zeta_i (1 + \xi \xi_i) (1 - \eta^2) \left(\frac{1}{9} + \eta \eta_i \right).$$

— Nodes on the sides parallel to the axis ζ :

Node i	13	14	15	16	17	18	19	20
ξ_i	-1	1	1	-1	-1	1	1	-1
η_i	-1	-1	1	1	-1	-1	1	1
ζ_i	$-\frac{1}{3}$	$-\frac{1}{3}$	$-\frac{1}{3}$	$-\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$

$$N_i = \frac{81}{64} (1 + \xi \xi_i) (1 + \eta \eta_i) (1 - \zeta^2) \left(\frac{1}{9} + \zeta \zeta_i \right)$$

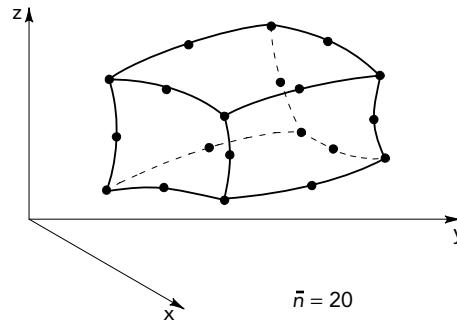
$$\frac{\partial N_i}{\partial \xi} = \frac{81}{64} \xi_i (1 + \eta \eta_i) (1 - \zeta^2) \left(\frac{1}{9} + \zeta \zeta_i \right)$$

$$\frac{\partial N_i}{\partial \eta} = \frac{81}{64} \eta_i (1 + \xi \xi_i) (1 - \zeta^2) \left(\frac{1}{9} + \zeta \zeta_i \right)$$

$$\frac{\partial N_i}{\partial \zeta} = \frac{81}{64} (1 + \xi \xi_i) (1 + \eta \eta_i) \left(\zeta_i - \frac{2 \zeta}{9} - 3 \zeta^2 \zeta_i \right).$$

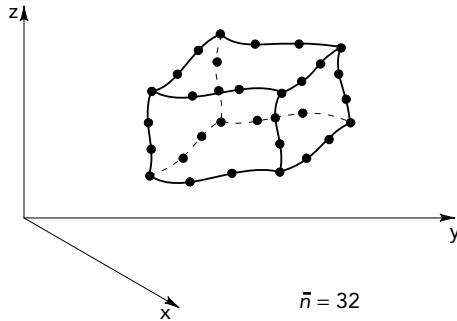
2.6.2.4 Curvilinear elements

a) Elements with quadratic surfaces



The functions \bar{N} are identical to the functions N from section 2.6.2.2.

b) Elements with cubic surfaces



The functions \bar{N} are identical to the functions N from section 2.6.2.3.

2.6.3 HIGH-PRECISION HERMITE ELEMENTS

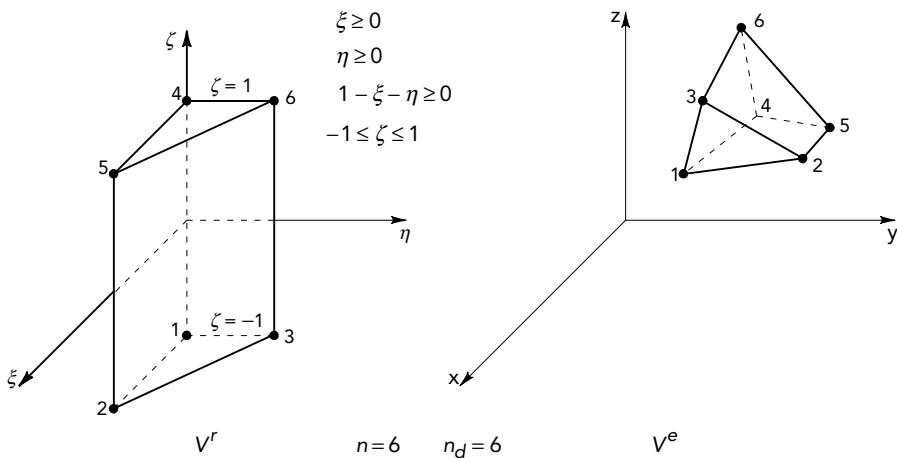
Once again, it is possible to construct a high-precision semi- C^1 element with eight nodes using the polynomial basis from section 2.6.2.3 and four nodal variables per node:

$$u_i \ \partial_\xi u_i \ \partial_\eta u_i \ \partial_\zeta u_i.$$

Three-dimensional C^1 elements are rarely used, given the very high number of degrees of freedom.

2.7 Prismatic elements (three dimensions)

2.7.1 ELEMENT WITH SIX NODES (prism, six nodes, C^0)



$$\langle P \rangle = \langle 1 \ \xi \ \eta \ \zeta \ \xi\zeta \ \eta\zeta \rangle \quad (2.33)$$

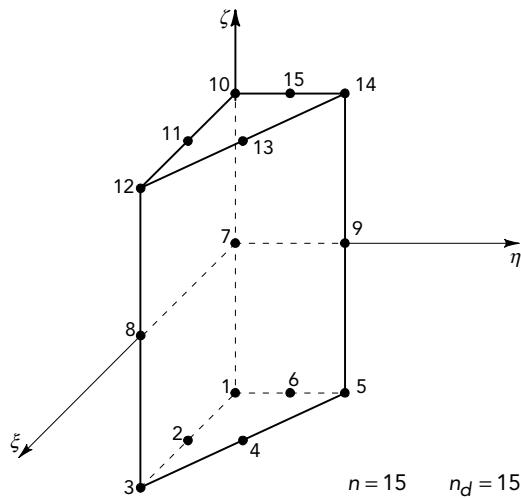
$\{N\}$	$\{\partial N / \partial \xi\}$	$\{\partial N / \partial \eta\}$	$\{\partial N / \partial \zeta\}$
1	λa	$-a$	$-a$
2	ξa	a	0
3	ηa	0	a
4	λb	$-b$	$-b$
5	ξb	b	0
6	ηb	0	b

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

$$a = \frac{1 - \zeta}{2}$$

$$b = \frac{1 + \zeta}{2}.$$

2.7.2 ELEMENT WITH 15 NODES (prism, 15 nodes, C^0)



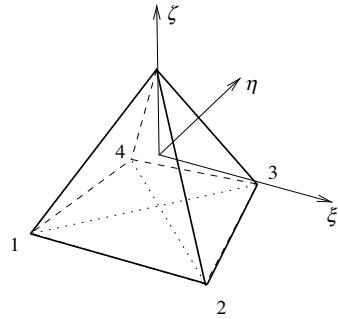
$$\langle P \rangle = \langle 1 \ \xi \ \eta \ \zeta \ \xi^2 \ \eta^2 \ \zeta^2 \ \xi\eta \ \xi\zeta \ \eta\zeta \ \xi^2\zeta \ \eta^2\zeta \ \xi\zeta^2 \ \eta\zeta^2 \ \xi\eta\zeta \rangle. \quad (2.34)$$

The functions N are given in accordance with:

$$\begin{aligned} N = & \left\langle a\lambda(2\lambda - \zeta - 2) \ 4a\lambda\xi \ a\xi(2\xi - \zeta - 2) \ 4a\eta\xi \ a\eta(2\eta - \zeta - 2) \ 4a\eta\lambda \right. \\ & (1 - \zeta^2)\lambda \ (1 - \zeta^2)\xi \ (1 - \zeta^2)\eta \\ & b\lambda(2\lambda + \zeta - 2) \ 4b\lambda\xi \ b\xi(2\xi + \zeta - 2) \ 4b\eta\xi \ b\eta(2\eta + \zeta - 2) \ 4b\eta\lambda \rangle \\ a = & \frac{1 - \xi}{2}, \quad b = \frac{1 + \xi}{2}. \end{aligned}$$

2.8 Pyramidal element (three dimensions)

2.8.1 ELEMENT WITH FIVE NODES



$$\frac{1 - \zeta}{2} \leq \xi \leq \frac{1 + \zeta}{2}; \quad \frac{1 - \zeta}{2} \leq \eta \leq \frac{1 + \zeta}{2}; \quad -1 \leq \zeta \leq 1.$$

The functions N are given by:

$$\begin{aligned} \langle N \rangle = & \left\langle \frac{1}{8}(1 - \xi)(1 - \eta)(1 - \zeta) \ \frac{1}{8}(1 + \xi)(1 - \eta)(1 - \zeta) \right. \\ & \frac{1}{8}(1 + \xi)(1 + \eta)(1 - \zeta) \ \frac{1}{8}(1 - \xi)(1 + \eta)(1 - \zeta) \\ & \left. \frac{1}{2}(1 - \zeta) \right\rangle \end{aligned}$$

The geometry is defined by:

$$\langle x \ y \ z \rangle = \langle N \rangle [x_n \ y_n \ z_n] = \langle N \rangle [X_n].$$

The Jacobian matrix is calculated by:

$$[J] = \begin{bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \\ \frac{\partial}{\partial \zeta} \end{bmatrix} \langle x(\xi, \eta, \zeta) \ y(\xi, \eta, \zeta) \ z(\xi, \eta, \zeta) \rangle.$$

For an element of reference of coordinates to the nodes:

$$(-1, -1, -1), \quad (+1, -1, -1), \quad (+1, +1, -1), \quad (-1, +1, -1), \quad (0, 0, 1)$$

the Jacobian matrix is written by:

$$[J] = \begin{bmatrix} \frac{1-\zeta}{2} & 0 & 0 \\ 0 & \frac{1-\zeta}{2} & 0 \\ -\frac{\xi}{2} & -\frac{\eta}{2} & 1 \end{bmatrix} \text{ where } |J| = \frac{(1-\zeta)^2}{4}; \quad \text{Volume} = \frac{8}{3}.$$

2.9 Other elements

2.9.1 APPROXIMATION OF VECTORIAL VALUES

If we wish to construct an approximation for the domain V of a vectorial value:

$$\mathbf{u} = \begin{Bmatrix} u \\ v \\ p \end{Bmatrix},$$

we will use a finite element approximation for each component:

$$u = \langle N_u \rangle \{u_n\}; \quad v = \langle N_v \rangle \{v_n\}; \quad p = \langle N_p \rangle \{p_n\}, \quad (2.35a)$$

which can be written as:

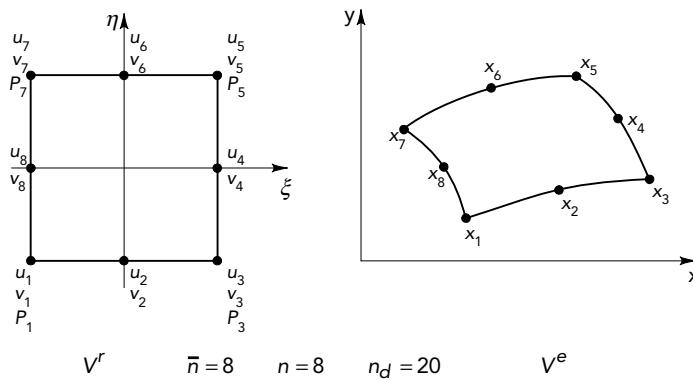
$$\mathbf{u} = \begin{Bmatrix} u \\ v \\ p \end{Bmatrix} = \begin{bmatrix} \langle N_u \rangle & & 0 \\ & \langle N_v \rangle & \\ 0 & & \langle N_p \rangle \end{bmatrix} \begin{Bmatrix} \{u_n\} \\ \{v_n\} \\ \{p_n\} \end{Bmatrix}, \quad (2.35b)$$

$$\mathbf{u} = [N] \{u_n\}.$$

When the components have similar natures or characteristics, we often assign them the same functions N , e.g. $\langle N_u \rangle \equiv \langle N_v \rangle$.

EXAMPLE 2.1. *Quadrilateral element with eight nodes
for fluid mechanics*

In certain problems in two-dimensional fluid mechanics, we have to construct an approximation of a velocity field with components u and v , and of a pressure field p . For the purposes of this problem, it is desirable to use a linear approximation for p and a quadratic approximation for u and v .



We choose the following approximation:

$$\begin{Bmatrix} u \\ v \\ p \end{Bmatrix} = \begin{bmatrix} \langle N_u \rangle_{(1 \times 8)} & 0 & 0 \\ 0 & \langle N_u \rangle_{(1 \times 8)} & 0 \\ 0 & 0 & \langle N_p \rangle_{(1 \times 4)} \end{bmatrix} \begin{Bmatrix} u_n \\ v_n \\ p_n \end{Bmatrix}_{(8 \times 1) \quad (8 \times 1) \quad (4 \times 1)}$$

$$\begin{Bmatrix} x \\ y \end{Bmatrix} = \begin{bmatrix} \langle \bar{N} \rangle & 0 \\ 0 & \langle \bar{N} \rangle \end{bmatrix} \begin{Bmatrix} \{x_n\} \\ \{y_n\} \end{Bmatrix}$$

where $\langle N_u \rangle \equiv \langle \bar{N} \rangle$ is given in section 2.4.3.2

$\langle N_p \rangle$ is given in section 2.4.2.

$$\begin{aligned}
 \{u_n\}^T &= \langle u_1 \dots u_8 \rangle \\
 \{v_n\}^T &= \langle v_1 \dots v_8 \rangle \\
 \{p_n\}^T &= \langle p_1 \ p_3 \ p_5 \ p_7 \rangle \\
 \{x_n\}^T &= \langle x_1 \dots x_8 \rangle \\
 \{\gamma_n\}^T &= \langle \gamma_1 \dots \gamma_p \rangle.
 \end{aligned}$$

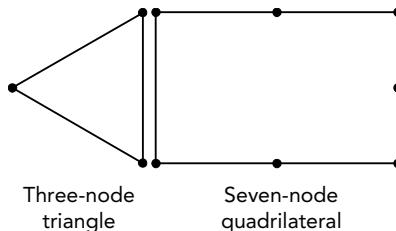
Remark

- An approximation with nine terms for (u, v) is sometimes used by adding an internal node (section 2.4.3.1).

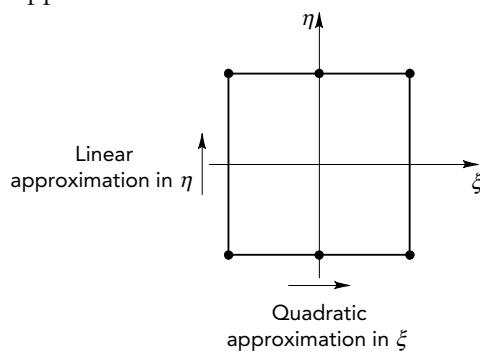
2.9.2 MODIFICATIONS OF THE ELEMENTS

It is sometimes useful to have elements that present different numbers of nodes on their various sides. Thus, the interpolation function will be of varying degrees on the various sides. This enables us, for example, to:

- link different types of elements;



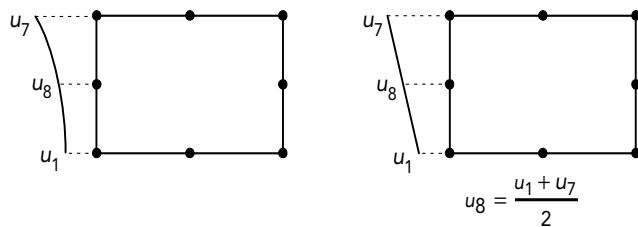
- use a degree of approximation that is different in the directions ξ, η :



These elements are constructed by transforming conventional elements, by eliminating nodes. To this end, we have to introduce linear relations between the nodal variables so as to eliminate certain variables.

EXAMPLE 2.2. Quadratic quadrilateral element with seven nodes

We work from the element with eight nodes and write that $u(\xi, \eta)$ is linear on the side 1-7-8.



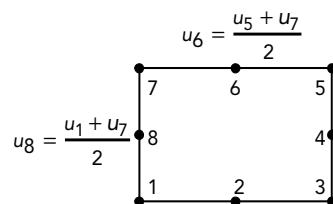
For the element with eight nodes:

$$\langle N \rangle = \langle N_1 \ N_2 \ \dots \ N_8 \rangle.$$

For the element with seven nodes (1 3 4 5 6 7):

$$\langle N \rangle = \left\langle \left(N_1 + \frac{N_8}{2} \right) N_2 \ N_3 \ N_4 \ N_5 \ N_6 \left(N_7 + \frac{N_8}{2} \right) \right\rangle.$$

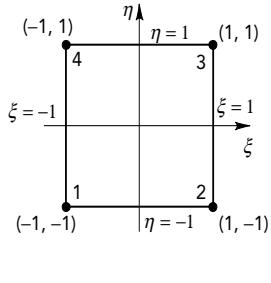
For an element with six nodes (1 2 3 4 5 7):



$$\langle N \rangle = \left\langle \left(N_1 + \frac{N_8}{2} \right) N_2 \ N_3 \ N_4 \ \left(N_5 + \frac{N_6}{2} \right) \left(N_7 + \frac{N_6}{2} + \frac{N_8}{2} \right) \right\rangle.$$

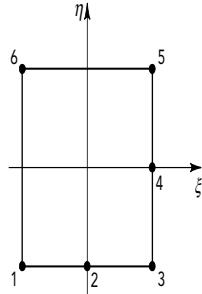
2.9.3 ELEMENTS WITH A VARIABLE NUMBER OF NODES [BAT 76]

So far, we have presented the interpolation functions of linear and quadratic elements with one, two and three dimensions separately. We can also construct functions N of quadratic elements by adding terms to the functions N of the linear elements; each of these terms corresponds to the addition of one node to one side of the element. Consider the example of a quadrilateral with four nodes.



$$\begin{aligned}
 N_1 &= \frac{1}{4}(1-\xi)(1-\eta) \\
 N_2 &= \frac{1}{4}(1+\xi)(1-\eta) \quad (\text{section 2.4.2}) \\
 N_3 &= \frac{1}{4}(1+\xi)(1+\eta) \\
 N_4 &= \frac{1}{4}(1-\xi)(1+\eta)
 \end{aligned}$$

Let us add a mid-side node to sides $\eta = -1$ and $\xi = 1$; the corresponding interpolation functions N^* are:



— corner nodes:

$$\begin{aligned}
 N_1^* &= N_1 - \frac{a}{2} \\
 N_3^* &= N_3 - \frac{a}{2} - \frac{b}{2} \\
 N_5^* &= N_5 - \frac{b}{2} \\
 N_6^* &= N_6.
 \end{aligned} \tag{2.36}$$

— side nodes:

$$\begin{aligned}
 N_2^* &= a \\
 N_4^* &= b
 \end{aligned}$$

where

$$\begin{aligned}
 a &= \frac{1}{2}(1-\xi^2)(1-\eta) \\
 b &= \frac{1}{2}(1+\xi)(1-\eta^2)
 \end{aligned}$$

are the interpolation functions N_2^* , and N_4^* , the side nodes of the eight-node quadratic element given in section 2.4.3.2.

2.9.4 SUPERPARAMETRIC ELEMENTS

Up until now we have used isoparametric ($N \equiv \bar{N}$) and subparametric elements (quadratic or cubic straight-sided elements). The elements are superparametric when the degree of \bar{N} is greater than that of N .

These elements are not commonly used, because they pose a problem of convergence: if we want the errors $|e|_1$ on the first derivatives $\partial u / \partial x, \partial u / \partial y, \partial u / \partial z$, to tend toward zero as the size of the element tends toward zero, the approximation of u must contain a first-order complete polynomial in 1 and \mathbf{x}, y, z .

We search for the condition so that the approximation u contains a linear polynomial of the form:

$$u_0(x, y) = a_1 + a_2 x + a_3 y. \quad (2.37a)$$

The two-dimensional geometric transformation:

$$\begin{aligned} x &= \langle \bar{N} \rangle \{x_n\} \\ y &= \langle \bar{N} \rangle \{\gamma_n\} \end{aligned} \quad (2.37b)$$

enables us to express u_0 in terms of ξ, η :

$$\begin{aligned} u_0(\xi, \eta) &= a_1 + a_2 \langle \bar{N} \rangle \{x_n\} + a_3 \langle \bar{N} \rangle \{\gamma_n\} \\ &= \langle \bar{N} \rangle \left\{ \begin{array}{c} b_1 \\ b_2 \\ \vdots \\ b_n \end{array} \right\} \text{ where } \begin{array}{l} b_i = a_1 + a_2 x_i + a_3 \gamma_i \\ \sum_i \bar{N}_i = 1. \end{array} \end{aligned} \quad (2.37c)$$

In order for the approximation:

$$u(\xi, \eta) = \langle N \rangle \{u_n\}$$

to contain the expression $u_0(\xi, \eta)$, every function \bar{N}_i must be a linear combination of the functions N_i . If $\langle P \rangle$ and $\langle \bar{P} \rangle$ are the polynomial bases corresponding to the functions $\langle N \rangle$ and $\langle \bar{N} \rangle$, the basis $\langle \bar{P} \rangle$ must be included in the basis $\langle P \rangle$. For superparametric elements, this condition is not satisfied.

2.9.5 INFINITE ELEMENTS

For functions monotonically converging toward zero at infinity, it is possible to map the infinite region into a reference element using a singular mapping function.

Consider the following one-dimensional geometric transformation that transforms node 2 of the reference element into a point of the real element, situated in infinity:



$$x = x_1 + \alpha \frac{1+\xi}{1-\xi}. \quad (2.38a)$$

We use the conventional linear approximation on the reference element with $u_2 = 0$ at infinity:

$$u(\xi) = \left\langle \frac{1-\xi}{2} \frac{1+\xi}{2} \right\rangle \begin{cases} u_1 \\ u_2 = 0 \end{cases} = \frac{1-\xi}{2} u_1. \quad (2.38b)$$

Using the transformation (2.38a)

$$\xi = \frac{\alpha(x - x_1) - 1}{\alpha(x - x_1) + 1}$$

we obtain

$$u(x) = \frac{1}{1 + \alpha(x - x_1)} u_1. \quad (2.38c)$$

This approximation of u tends toward zero as x tends toward zero at infinity in $1/\alpha x$. It is possible to modify the form of $u(x)$ by multiplying the interpolation function $N_1(\xi)$ with a function $f(\xi)$ that becomes null for $\xi = 1$; for instance:

$$f(\xi) = \exp \left[-\frac{\beta(1+\xi)}{1-\xi} \right].$$

Thus

$$u(\xi) = \exp \left[-\frac{\beta(1+\xi)}{1-\xi} \right] \cdot \frac{1-\xi}{2} \cdot u_1$$

$$u(x) = e^{-\frac{\beta(x-x_1)}{\alpha}} \frac{1}{1 + \alpha(x - x_1)} u_1.$$

The same technique can be applied to two-dimensional elements. Other choices of functions are offered in [BET 77].

Bibliography

- [ARN 84] ARNOLD D.N., BREZZI F., FORTIN M., “A stable finite element for Stokes equations”, *Calcolo*, vol. 21, no. 4, pp. 337–344, 1984.
- [BAT 76] BATHE K.J., WILSON E.L., *Numerical Methods in Finite Element Analysis*, Prentice-Hall, 1976.
- [BAT 90] BATOZ J.L., DHATT G., *Modélisation des structures par éléments finis*, Vol. 1, Hermès, 1990.
- [BET 77] BETTESS P., “Infinite elements”, *International Journal for Numerical Methods in Engineering*, vol. 11, pp. 53–64, 1977.
- [MIT 77] MITCHEL A.R., WAIT R., *The Finite Element Method in Partial Differential Equations*, Wiley, 1977.
- [ODE 72] ODEN J.T., *Finite Elements of Non-Linear Continua*, McGraw-Hill, New York, 1972.
- [SYN 57] SYNGE J.L., *The Hypercircle Method in Mathematical Physics*, Cambridge University Press, 1957.
- [ZIE 00] ZIENKIEWICZ O.C., *The Finite Element Method: The Basis (Vol. 1), Solid Mechanics (Vol. 2) & Fluid Mechanics (Vol. 3)*, 5th edition, Butterworth-Heinemann, 2000.
- [ZLA 73] ZLAMAL M., “Some recent advances in the mathematics of finite elements”, *Mathematics of Finite Elements and Applications*, Academic Press, pp. 59–81, 1973.

CHAPTER 3

Integral formulation

3.0 Introduction

The first two chapters have been devoted to the approximation of functions using the finite element method, and to the description of the most commonly used elements. In this chapter, we devote our attention to **the construction of integral formulations** (variational formulations or weak formulations) associated with partial differential equations describing the behavior of physical systems. The finite element method, outlined in Chapter 4, is then employed to discretize the integral formulation leading to a system of algebraic equations which are numerically solved giving a reasonably accurate solution to the problem (Figure 3.1).

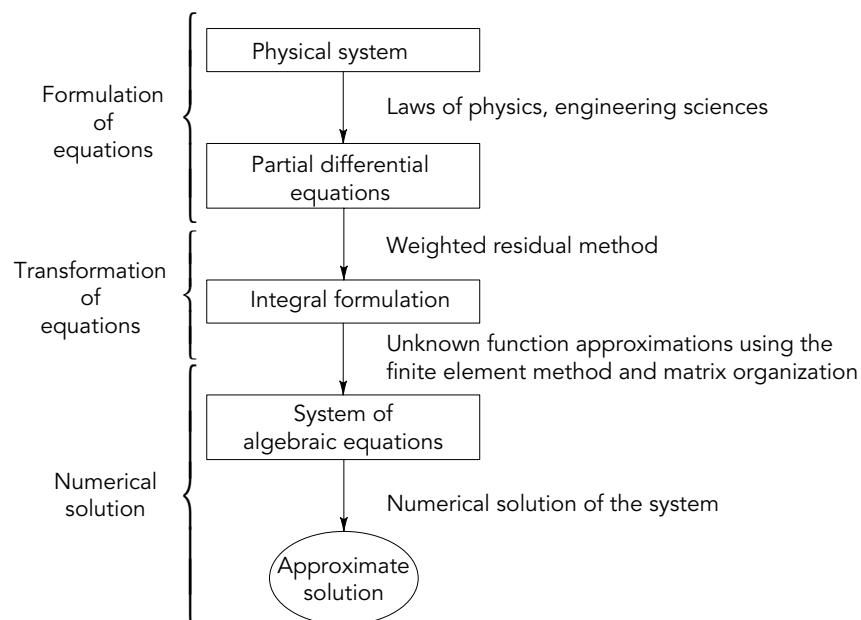


Figure 3.1. Transformation of the equations of a physical system

We first broadly classify physical problems into discrete and continuous systems. Then, we present the **weighted residual method** that enables us to transform a system of partial differential equations describing the behavior of a continuous system into an **integral formulation**. **Integration by parts** is often employed to obtain integral formulations with lower order derivatives that are easier to use.

In solid mechanics, the notion of a **functional** is often employed to directly construct an integral formulation using the principle of stationarity of the functional of energy. The **Lagrange multiplier method** furnishes modified functionals either **mixed** or **complementary**, which can be useful for certain problems. Let us stress that the notion of a functional is not necessary if we can describe the behavior of a physical system by a set of partial differential equations. We may then systematically employ the weighted residual method to obtain the integral formulation associated with the partial differential equations.

The weighted residual method, depending on the **choice of the weighting functions**, provides a whole ensemble of integral formulations:

- the **Galerkin** formulation, or **Ritz** formulation if we use the notion of a functional. This is the most widely used method;
- a point or subdomain **collocation** formulation;
- a **least squares** formulation;
- a **boundary-integral** formulation.

The method of undetermined parameters consists of replacing the unknown functions in one of the above integral formulations with approximations of type (1.2) that depend on a finite number of parameters. This method is called the finite element method if we use the finite element approximation defined in section 1.1.2. Thus we obtain a discretized expression of an integral formulation that constitutes the system of algebraic equations whose numerical solution is an approximate solution to the original problem.

The organization of this chapter is represented diagrammatically in Figure 3.2.

Remarks

- The description of the behavior of a physical problem in the form of partial differential equations (PDEs) is called “strong” formulation.
- The expression of a physical problem in integral form is called “weak” expression.
- The solution of a weak model corresponds to an approximate or “weak” solution in terms of continuity.

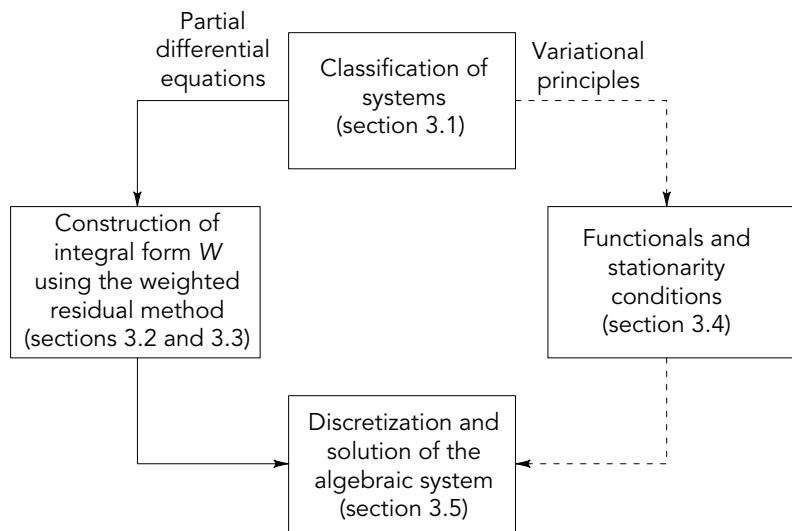


Figure 3.2. Organization of Chapter 3

3.1 Classification of physical systems [CRA 56; COL 66; KRE 88; FAR 82]

3.1.1 DISCRETE AND CONTINUOUS SYSTEMS

The behavior of a physical system is characterized by a set of variables that may depend on the coordinates of space $\mathbf{x} = (x, y, z)$ and time t . The system is said to be **stationary** if none of the variables are time dependent.

A number of variables \mathbf{d} of the system are known *a priori*: material properties, geometrical dimensions, applied forces, boundary conditions, etc. Other variables \mathbf{u} are unknown: displacements, velocities, temperatures, stresses, etc.

A mathematical model of the system results in relations between \mathbf{u} and \mathbf{d} using physical laws. These relations form a system of equations in \mathbf{u} to be solved. The **number of degrees of freedom** of the system is the number of parameters required to define \mathbf{u} at a given time t .

A system is **discrete** if the number of degrees of freedom is finite; it is **continuous** if the number of degrees of freedom is infinite.

The behavior of a discrete system is represented by a set of **algebraic equations**. That of a continuous system is most often represented by a system of **partial**

differential equations or integro-differential equations accompanied by appropriate spatial and temporal boundary conditions.

The algebraic equations of discrete systems can be solved directly by numerical methods as described in Chapter 5. On the other hand, the equations of continuous systems cannot generally be solved directly. They must be discretized, i.e. replaced with algebraic equations. The finite element method is one of the methods that may be used to carry out this discretization.

3.1.2 EQUILIBRIUM, EIGENVALUE AND PROPAGATION PROBLEMS

Discrete and continuous systems can be subdivided into three broad categories, that we briefly describe below

a) Equilibrium or boundary value problems

For such problems, we are interested in the unknown values \mathbf{u} in a steady-state system. For a discrete system, the governing equations can generally be written in matrix form:

$$[K]\{U\} = \{F\} \quad (3.1a)$$

where: $[K]$ is a matrix, symmetrical or otherwise, dependent upon $\{U\}$ or not;

$\{U\}$ are the unknown variables;

$\{F\}$ are the known applied forces.

The behavior of a continuous system is described by partial differential equations:

$$\begin{aligned} \mathcal{L}(\mathbf{u}) + \mathbf{f}_V &= 0 && \text{over a domain } V \\ \mathcal{C}(\mathbf{u}) &= \mathbf{f}_S && \text{across the boundary } S \text{ of } V \end{aligned} \quad (3.1b)$$

where: \mathcal{L} and \mathcal{C} are differential operators characterizing the system;

\mathbf{u} are the unknown functions;

\mathbf{f}_V are known functions associated with the applied forces;

\mathbf{f}_S are functions associated with the boundary conditions.

b) Eigenvalue problems

This is a subclass of equilibrium problems governed by homogeneous equations in which we evaluate \mathbf{u} corresponding to critical values of certain

parameters λ , called eigenvalues. The corresponding homogeneous equations are written as:

- For a discrete system:

$$[K]\{U\} = \lambda[M]\{U\} \quad (3.2a)$$

where $[M]$ is a mass matrix and λ are unknown eigenvalues.

- For a continuous system:

$$\begin{aligned} \mathcal{L}_1(\mathbf{u}) &= \lambda \mathcal{L}_2(\mathbf{u}) \quad \text{over the domain } V \\ \mathcal{C}_1(\mathbf{u}) &= \lambda \mathcal{C}_2(\mathbf{u}) \quad \text{across the boundary } S \text{ of } V \end{aligned} \quad (3.2b)$$

where $\mathcal{L}_1, \mathcal{L}_2, \mathcal{C}_1$ and \mathcal{C}_2 are differential operators.

c) Propagation or initial value problems

These consist of evaluating $\mathbf{u}(\mathbf{x}, t)$ for $t > t_0$, in a non-steady-state system, when $\mathbf{u}(\mathbf{x}, t_0)$ is known.

- For a discrete system:

$$[M] \frac{d^2}{dt^2} \{U\} + [C] \frac{d}{dt} \{U\} + [K]\{U\} = \{F(t)\} \quad \text{for } t > t_0 \quad (3.3a)$$

with initial conditions:

$$\{U\} = \{U_0\} \text{ and } \frac{d}{dt}\{U\} = \{\dot{U}_0\} \quad \text{for } t = t_0$$

where $[C]$ is the damping matrix.

- For a continuous system:

$$\begin{aligned} m \frac{\partial^2 \mathbf{u}}{\partial t^2} + c \frac{\partial \mathbf{u}}{\partial t} + \mathcal{L}(\mathbf{u}) + \mathbf{f}_V &= 0 \quad \text{over } V \\ \mathcal{C}(\mathbf{u}) &= \mathbf{f}_S \quad \text{over } S \end{aligned} \quad (3.3b)$$

with initial conditions:

$$\mathbf{u} = \mathbf{u}_0 \text{ and } \frac{\partial \mathbf{u}}{\partial t} = \dot{\mathbf{u}}_0 \quad \text{for } t = t_0.$$

Figure 3.3 summarizes the classification system that will be used.

We now introduce a few terms frequently employed to characterize the equations of physical systems:

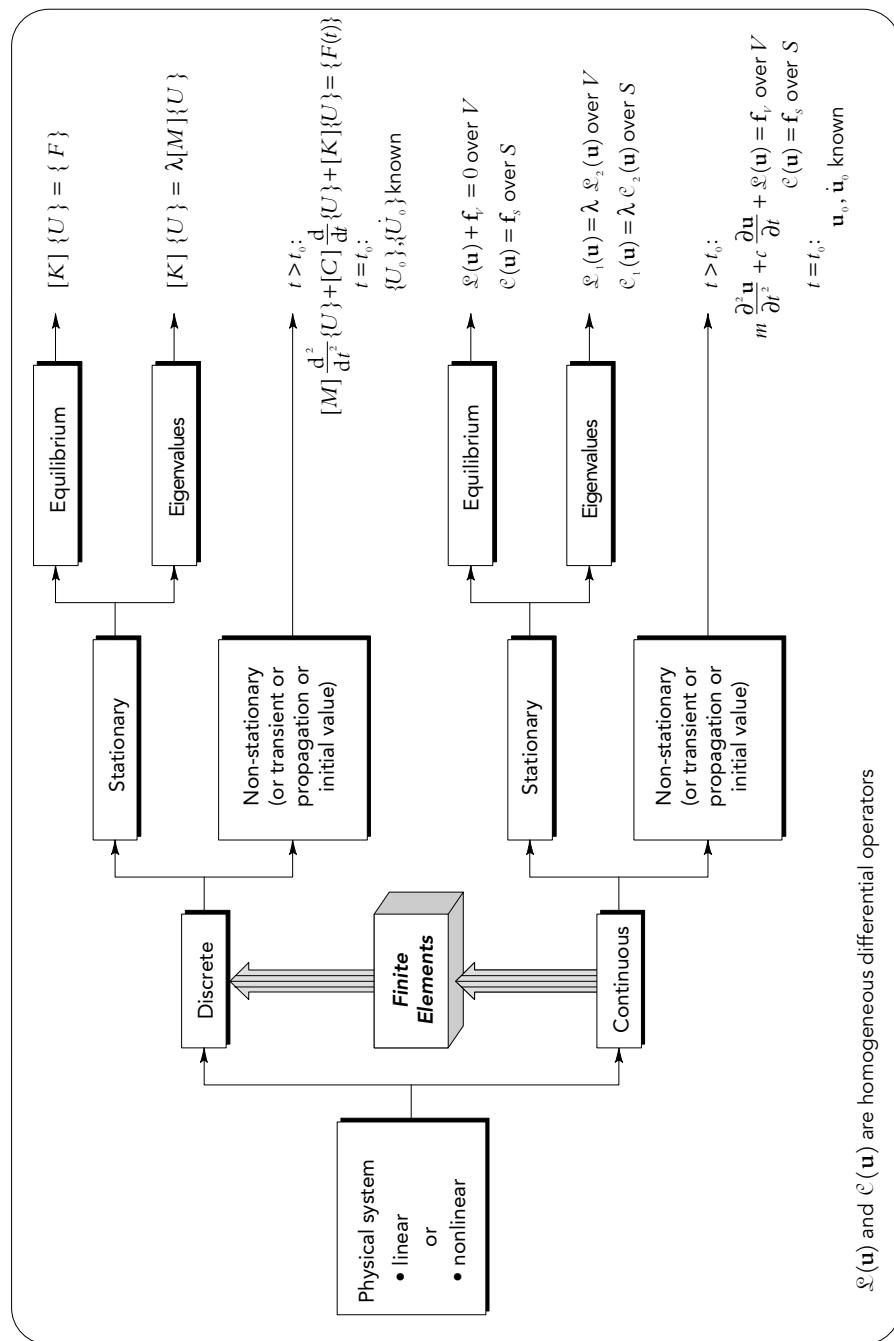


Figure 3.3. Classification of physical systems

- A **discrete system is linear** if the terms of $[K]$, $[M]$, $[C]$ and $\{F\}$ are constants, independent of \mathbf{u} .
- A **continuous system is linear** if the expressions $\mathcal{L}(\mathbf{u})$ and $\mathcal{C}(\mathbf{u})$ are linear in u and its derivatives. In addition, f_V, f_S, m, c are independent of \mathbf{u} and its derivatives. Thus, we can write:

$$\mathcal{L}(\mathbf{u}) = [\mathcal{L}] \{u\}; \quad \mathcal{C}(\mathbf{u}) = [\mathcal{C}] \{u\}$$

where $[\mathcal{L}]$ and $[\mathcal{C}]$ are matrices of differential operators independent of \mathbf{u} . For instance, for the Laplacian operator:

$$\mathcal{L}(u) = \Delta u = \frac{\partial^2 \mathbf{u}}{\partial x^2} + \frac{\partial^2 \mathbf{u}}{\partial y^2} = \left[\frac{\partial}{\partial x^2} + \frac{\partial}{\partial y^2} \right] \cdot u$$

- A system of partial differential equations is of **order m** if it contains derivatives of \mathbf{u} up to order m .
- A differential operator \mathcal{L} is said to be **homogeneous** if:

$$\mathcal{L}(u = 0) = 0.$$

- A system of linear partial differential equations:

$$[\mathcal{L}] \{u\} + \{f_V\} = 0$$

is homogeneous if:

$$\{f_V\} = 0$$

and the boundary conditions

$$[\mathcal{C}] \{u\} = \{f_S\}$$

are homogeneous if:

$$\{f_S\} = 0.$$

- A system of linear partial differential equations is **self-adjoint** or **symmetrical** if:

$$\int_V \langle u \rangle [\mathcal{L}] \{v\} dV = \int_V \langle v \rangle [\mathcal{L}] \{u\} dV \quad (3.4a)$$

where u and v are sufficiently derivable functions on V , which satisfy the homogeneous boundary conditions:

$$\mathcal{C}(\mathbf{u}) = \mathcal{C}(\mathbf{v}) = 0. \quad (3.4b)$$

- A linear system of partial differential equations is **positive** if:

$$\int_V \langle u \rangle [\mathcal{L}] \{u\} dV \geq 0 \quad (3.4c)$$

for any function \mathbf{u} satisfying (3.4b).

If (3.4c) is null for $\mathbf{u} = 0$ only, the system is positive definite.

- The matrix $[K]$ of a discrete system is symmetrical if:

$$\langle V \rangle [K] \{U\} = \langle U \rangle [K] \{V\} \quad \forall \{U\}, \{V\} \text{ so that } [K] = [K]^T.$$

- A matrix $[K]$ is positive definite if the quadratic form satisfies:

$$\langle U \rangle [K] \{U\} > 0 \quad \forall \{U\} \neq \{0\}. \quad (3.4d)$$

- Physical phenomena may be classified into three distinct types: elliptic, parabolic and hyperbolic. The first is representative of diffusion-type stationary problems, with the other two being associated with transient and propagation problems:

$\Delta u + f = 0$: elliptical (stationary, diffusion)

$\frac{\partial u}{\partial t} - \Delta u = 0$: parabolic (transient, diffusion)

$\frac{\partial^2 u}{\partial t^2} - \Delta u = 0$: hyperbolic (dynamic)

$\frac{\partial u}{\partial t} + \vec{a} \cdot \vec{\nabla} u = 0$: hyperbolic (transient, diffusion)

A hyperbolic problem allows for a general solution of the type $A \times e^{\alpha(s-c.t)}$ where c is a propagation rate. A problem regulated by the general relations of compressible fluid mechanics has the particular property of being parabolic for subsonic flows (two directions of propagation on a one-dimensional plane) and hyperbolic for supersonic flows (only one direction of propagation). Parabolic or hyperbolic problems are generally associated with the notion of characteristics that expresses a favored particular direction of information flow in space-time (x, y and t on a two-dimensional plane). This may prove useful, for example, for determining the number of boundary conditions to be imposed on the inputs and outputs of the field depending on whether the flow therein is subsonic or supersonic [COR 00].

EXAMPLE 3.1. Two-dimensional continuous problems

Equilibrium problem

The following Poisson equation corresponds to a two-dimensional steady-state continuous system:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f_V = 0 \quad \text{on } V.$$

That equation governs many physical problems; for instance, the problem of steady-state heat conduction u in a two-dimensional, homogeneous and isotropic medium. The solution becomes unique after one of the following two types of boundary conditions is applied on every point of the boundary S of the domain V :

- Condition on u (the so-called Dirichlet boundary condition):

$$u = u_S \text{ on } S_u$$

where S_u represents the part of S on which that condition is imposed.

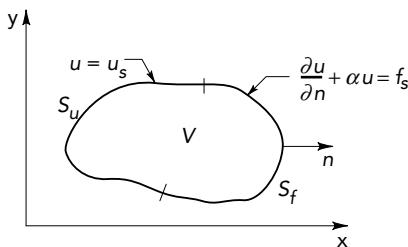
- Condition on $\frac{\partial u}{\partial n}$ or flux condition q_n :

$$\frac{\partial u}{\partial n} + \alpha u = f_S \text{ on } S_f$$

where S_f represents the part of boundary S on which that condition is imposed.

If $\alpha \neq 0$, this is called the Cauchy boundary condition.

If $\alpha = 0$, this is called the Neumann boundary condition.



The governing equation and its boundary conditions can be written in matrix form as:

$$\left[\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right] \cdot u + f_V = 0 \quad \text{so that } [\mathcal{L}] \{u\} + \{f_V\} = 0 \text{ on } V.$$

$$\left[\frac{\partial}{\partial n} + \alpha \right] \cdot u + f_S \quad \text{so that } [\mathcal{C}_f] \{u\} = \{f_S\} \text{ on } S_f.$$

$$[1] \cdot u = u_S \quad \text{so that } [\mathcal{C}_u] \{u\} = \{f_u\} \text{ on } S_u.$$

Solving such an equilibrium problem involves finding the function u that satisfies all three of these equations. The system is self-adjoint because we can show for homogeneous boundary conditions, using integration by parts, that:

$$\int_V u \left[\frac{\partial^2 \nu}{\partial x^2} + \frac{\partial^2 \nu}{\partial y^2} \right] dV = \int_V \nu \left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right] dV$$

when u and v satisfy the homogeneous boundary conditions:

$$\left. \begin{array}{l} \frac{\partial u}{\partial n} + \alpha u = 0 \\ \frac{\partial v}{\partial n} + \alpha v = 0 \end{array} \right\} \text{on } S_f; \quad \left. \begin{array}{l} u = 0 \\ v = 0 \end{array} \right\} \text{on } S_u.$$

We can also show, in a similar fashion, that the system is positive definite because:

$$\int_V u \left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right] dV > 0 \quad \text{if } \alpha \geq 0$$

for any non-null value of u that satisfies the aforementioned homogeneous boundary conditions.

Eigenvalue problem

The Helmholtz equation is:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \lambda u = 0 \quad \text{on } V.$$

It is associated with Neumann or Dirichlet homogeneous boundary conditions. The solution of this problem consists of calculating both the parameter λ and the function u .

For instance, this equation can define the eigenmodes u and the eigenfrequencies $\sqrt{\lambda}$ of vibration of an elastic membrane under tension. It also applies to calculating the electromagnetic waves and vibrations of fluids in acoustics.

Transient problem

A typical transient problem can be described as:

$$-\frac{\partial u}{\partial t} + \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f(t) = 0 \quad \text{on } V.$$

with boundary conditions for $t > t_0$:

$$\frac{\partial u}{\partial n} + \alpha u = f_s \text{ on } S_f$$

$$u = u_S \text{ on } S_u.$$

and initial conditions for $t = t_0$:

$$u = u_0 \text{ on } V.$$

For instance, this equation governs a transient temperature distribution u in a two-dimensional medium.

Solving the problem entails finding the function u at everytime $t > t_0$ that satisfies the above three equations.

EXAMPLE 3.2. Navier-Stokes equation

A transient, two-dimensional, incompressible laminar viscous flow is governed by the Navier-Stokes equations:

$$\begin{aligned} \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + \frac{1}{\rho} \frac{\partial p}{\partial x} - \frac{\mu}{\rho} \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) + f_x &= 0 \\ \frac{\partial v}{\partial t} + u \frac{\partial v}{\partial x} + v \frac{\partial v}{\partial y} + \frac{1}{\rho} \frac{\partial p}{\partial y} - \frac{\mu}{\rho} \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) + f_y &= 0 \\ \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0 \end{aligned}$$

where: u and v are the components of the velocity vector field;

p is the static pressure;

f_x , f_y are the forces applied per unit mass;

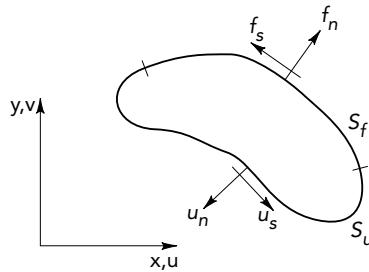
μ is the dynamic viscosity of the fluid;

ρ is its density (constant).

The boundary conditions are, for example, ($t > t_0$):

$$\left. \begin{aligned} -p + \mu \frac{\partial u_n}{\partial n} &= \rho f_n \\ \mu \frac{\partial u_s}{\partial n} &= \rho f_s \end{aligned} \right\} \text{on } S_f \text{ (loads)} \\ \left. \begin{aligned} u_n &= u_{nS} \\ u_s &= u_{sS} \end{aligned} \right\} \text{on } S_u \text{ (kinematics).} \\$$

where n and s are the normal and tangential directions to the boundary.



The initial conditions are ($t = t_0$):

$$u = u_0; \quad v = v_0; \quad p = p_0 \quad \text{on } V.$$

The above nonlinear equations can be written in matrix form:

$$\{\dot{u}\} + [\mathcal{L}] \{u\} + \{f_V\} = 0$$

where:

$$\{\dot{u}\} = \begin{pmatrix} \frac{\partial u}{\partial t} \\ \frac{\partial v}{\partial t} \\ 0 \end{pmatrix}; \quad \{u\} = \begin{pmatrix} u \\ v \\ p \end{pmatrix}; \quad \{f_V\} = \begin{pmatrix} f_x \\ f_y \\ 0 \end{pmatrix}$$

$$[L] = \begin{bmatrix} u \frac{\partial}{\partial x} + v \frac{\partial}{\partial y} - \frac{\mu}{\rho} \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) & 0 & \frac{1}{\rho} \frac{\partial}{\partial x} \\ 0 & u \frac{\partial}{\partial x} + v \frac{\partial}{\partial y} - \frac{\mu}{\rho} \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right) & \frac{1}{\rho} \frac{\partial}{\partial y} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & 0 \end{bmatrix}$$

The stationary formulation is obtained by suppressing the term $\{\dot{u}\}$. We get the formulation of the Stokes problem by ignoring all the nonlinear terms.

3.2 Weighted residual method [FIN 72]

3.2.1 RESIDUALS

Consider a steady-state continuous physical system whose behavior is represented by a system of partial differential equations, linear or nonlinear, of order m :

$$\mathcal{L}(\mathbf{u}) + \mathbf{f}_V = 0 \quad \text{on domain } V \tag{3.5a}$$

The boundary conditions are:

$$C(\mathbf{u}) = \mathbf{f}_S \quad \text{on the boundary } S. \quad (3.5b)$$

Functions \mathbf{u} constitute a solution to the equilibrium problem if they satisfy both (3.5a) and (3.5b).

We use the term **residual** to denote the quantity $\mathbf{R}(\mathbf{u})$ defined by:

$$\mathbf{R}(\mathbf{u}) = \mathcal{L}(\mathbf{u}) + \mathbf{f}_V \quad (3.6)$$

which, of course, becomes zero when \mathbf{u} is a solution to (3.5). The residual is a vector when (3.5a) is a system of differential equations.

3.2.2 INTEGRAL FORMS

The **weighted residual method** involves finding functions \mathbf{u} that nullify the following **integral form**:

$$W(\mathbf{u}) = \int_V \langle \psi \rangle \{ R(\mathbf{u}) \} dV = \int_V \langle \psi \rangle \{ \mathcal{L}(\mathbf{u}) + f_V \} dV = 0 \quad (3.7)$$

for any **weighting function** ψ that belongs to a set of functions E_ψ , \mathbf{u} belonging to the ensemble E_u of acceptable solutions that satisfy the **boundary conditions** (3.5b) and that are differentiable up to the m th order.

Any solution \mathbf{u} that satisfies (3.5a) and (3.5b) also satisfies (3.7), whatever the choice of E_ψ . However, the solution \mathbf{u} to (3.7) depends on the choice of E_ψ . For instance, if the set E_ψ comprises all the Dirac distributions $\delta(\mathbf{x})$ on V , then the functions \mathbf{u} , which satisfy (3.7), also satisfy (3.5a), because the residual R is then null at every point of V . If the set E_ψ is finite, the solution \mathbf{u} , which satisfies the weak formulation (3.7), is an approximate solution to the problem: it does not satisfy (3.5a) exactly at all points of V .

The function $u(x)$ is called a **solution function**, and $\psi(x)$ is called a **weighting function** or a **test function**.

EXAMPLE 3.3. Integral form of Poisson's equation

The integral form of Poisson's equation from Example 3.1 is written as:

$$W = \int_V \psi(x, y) \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f_V \right) dV = 0$$

where u is twice differentiable and must satisfy all the boundary conditions on S_u and S_f , $\psi(x, y)$ is an arbitrary function.

EXAMPLE 3.4. Integral form of the Navier-Stokes equations

The integral form of the stationary Navier-Stokes equations from Example 3.2 can be written in matrix form as:

$$W = \int_V \langle \psi_u(x, y) \psi_v(x, y) \psi_p(x, y) \rangle \{[\mathcal{L}] \begin{Bmatrix} u \\ v \\ p \end{Bmatrix} + f_V\} dV = 0$$

where: u and v are twice differentiable, p is once differentiable;
 u , v and p satisfy all the boundary conditions on S_u and S_f ;
 $[\mathcal{L}]$ is defined in Example 3.2;
 ψ_u , ψ_v and ψ_p are arbitrary test functions.

3.3 Integral transformations

3.3.1 INTEGRATION BY PARTS [KRE 88]

Integration by parts can be used to transform an integral form such as (3.7) so as to lower the order of highest derivatives contained in the integrand. Let us first recap the formulae for integration by parts:

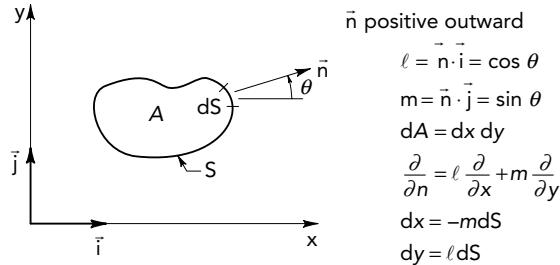
a) One dimension

$$\int_{x_1}^{x_2} \psi \frac{du}{dx} dx = - \int_{x_1}^{x_2} \frac{d\psi}{dx} u dx + (\psi u) \Big|_{x_1}^{x_2} \quad (3.8a)$$

$$\int_{x_1}^{x_2} \psi \frac{d^2u}{dx^2} dx = - \int_{x_1}^{x_2} \frac{d\psi}{dx} \frac{du}{dx} dx + \left(\psi \frac{du}{dx} \right) \Big|_{x_1}^{x_2} \quad (3.8b)$$

b) Two dimensions

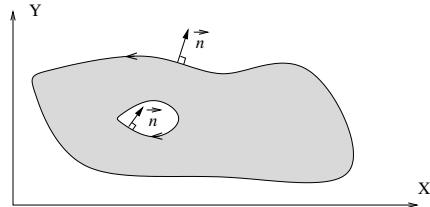
$$\begin{aligned} \int_A \psi \frac{\partial u}{\partial x} dx dy &= - \int_A \frac{\partial \psi}{\partial x} u dx dy + \oint_S \psi u dS \\ &= - \int_A \frac{\partial \psi}{\partial x} u dx dy + \oint_S \psi u l dS \end{aligned} \quad (3.9a)$$



$$\begin{aligned} \int_A \psi \frac{\partial u}{\partial y} dx dy &= - \int_A \frac{\partial \psi}{\partial y} u dx dy - \oint_S \psi u dx \\ &= - \int_A \frac{\partial \psi}{\partial y} u dx dy + \oint_S \psi u m ds \end{aligned} \quad (3.9b)$$

Remarks

- The normal \vec{n} is oriented toward the outside of the domain.
- Integration on the contour is carried out in the positive sense of rotation on the oriented triangular plane.
- If the domain is multi-convex, the direction of the path of the contour integral is such that the domain is always located on the left.



Using the relations (3.9a) and (3.9b), we obtain the divergence formula (called the Gaussian formula):

$$\int_A \psi \cdot \operatorname{Div}(\mathbf{u}) dA = - \int_A \nabla \psi \cdot \mathbf{u} dA + \oint_S \psi \mathbf{u} \cdot \mathbf{n} dS,$$

where:

$$\mathbf{u} = \langle u(x, y) \ v(x, y) \rangle, \quad \operatorname{Div}(\mathbf{u}) = \nabla \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y},$$

$$\nabla \psi = \left\langle \frac{\partial \psi}{\partial x} \ \frac{\partial \psi}{\partial y} \right\rangle, \quad \nabla \psi \cdot \mathbf{u} = \frac{\partial \psi}{\partial x} u + \frac{\partial \psi}{\partial y} v,$$

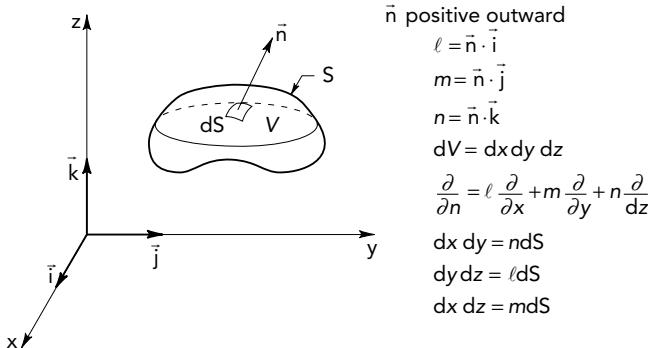
$$\mathbf{u} \cdot \mathbf{n} = u_n = u.l + v.m.$$

If $\psi = 1$, we have: $\int_A \operatorname{Div}(\mathbf{u}) dA = \oint_S \mathbf{u} \cdot \mathbf{n} dS.$

For a Laplacian operator, we obtain:

$$\begin{aligned} \int_A \psi \Delta u \, dA &= - \int_A \nabla \psi \cdot \nabla u \, dA + \oint_S \psi u_n \, dS. \\ \int_A (\psi \Delta u - u \Delta \psi) \, dx \, dy &= \oint_S \left(\psi \frac{\partial u}{\partial n} - u \frac{\partial \psi}{\partial n} \right) \, dS. \end{aligned} \quad (3.9c)$$

c) Three dimensions



The divergence formula is written as:

$$\int_V \psi \cdot \operatorname{Div}(\mathbf{u}) dV = - \int_V \nabla \psi \cdot \mathbf{u} dV + \oint_A \psi \mathbf{u} \cdot \mathbf{n} dA, \quad (3.10)$$

where, in the system of Cartesian coordinates:

$$\mathbf{u} = \langle u(x, y, z) \ v(x, y, z) \ w(x, y, z) \rangle, \quad \operatorname{Div}(\mathbf{u}) = \nabla \cdot \mathbf{u} = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}.$$

For the Laplacian operator, we have:

$$\int_V \psi \Delta u \, dV = - \int_V \nabla \psi \cdot \nabla u \, dV + \oint_A \psi \frac{\partial u}{\partial \mathbf{n}} \, dA,$$

where: $\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}$,

$$\nabla \psi \cdot \nabla u = \frac{\partial \psi}{\partial x} \frac{\partial u}{\partial x} + \frac{\partial \psi}{\partial y} \frac{\partial u}{\partial y} + \frac{\partial \psi}{\partial z} \frac{\partial u}{\partial z}.$$

3.3.2 WEAK INTEGRAL FORM

The integration by parts in (3.7) gives us the so-called **weak** integral forms that offer the following advantages:

- The maximum order of the derivatives of \mathbf{u} appearing in the integral form decreases. Thus, the derivability conditions of \mathbf{u} are less stringent.
- Some of the boundary conditions that appear in the weak form may be taken into account in the integral formulation, rather than being identically satisfied by \mathbf{u} .

On the other hand, integration by parts introduces derivatives of ψ . Hence, the derivability conditions of ψ increase. Furthermore, ψ may have to satisfy conditions on part of the boundary in order to eliminate certain contour terms. The solution function which satisfies the weak integral form represents an approximate solution to equation (3.5) with reduced derivability conditions. For example, a polygonal line may approximate an arbitrary curve as closely as we wish, without being differentiable at its corners.

EXAMPLE 3.5. Weak integral form of the Poisson equation

In the integral form, Example 3.3, the solution function, u must:

- *be twice differentiable;*
- *satisfy all the boundary conditions on S_f and S_u .*

The test functions ψ are not subject to any conditions.

After an integration by parts:

$$W = - \int_V \left(\frac{\partial \psi}{\partial x} \frac{\partial u}{\partial x} + \frac{\partial \psi}{\partial y} \frac{\partial u}{\partial y} - \psi f_V \right) dV + \int_{S_f} \psi \frac{\partial u}{\partial n} dS + \int_{S_u} \psi \frac{\partial u}{\partial n} dS = 0.$$

The functions ψ and u must be once differentiable. Now we have terms involving contour integrations on S_f and S_u . Instead of requiring the solution function u to satisfy exactly the boundary condition on S_f :

$$\frac{\partial u}{\partial n} = f_s - \alpha u \quad \text{on } S_f$$

we may satisfy it in a weak sense:

$$\int_{S_f} \psi \frac{\partial u}{\partial n} dS \quad \text{with} \quad \int_{S_f} \psi (f_s - \alpha u) dS.$$

In addition, we can eliminate the contour integral on S_u by imposing:

$$\psi = 0 \quad \text{on } S_u.$$

The weak integral form is then written:

$$W = - \int_V \left(\frac{\partial \psi}{\partial x} \frac{\partial u}{\partial x} + \frac{\partial \psi}{\partial y} \frac{\partial u}{\partial y} - \psi f_V \right) dV + \int_{S_f} \psi (f_s - \alpha u) dS = 0 \quad (1)$$

where u and ψ must satisfy the boundary conditions:

$$u = u_S; \quad \psi = 0 \quad \text{on } S_u.$$

After two rounds of integration by parts of the integral form, Example 3.3

$$W = \int_V \left(\left(\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} \right) u + \psi f_V \right) dV + \oint_S \left(\psi \frac{\partial u}{\partial n} - \frac{\partial \psi}{\partial n} u \right) dS = 0 \quad (2)$$

If we choose test or weighting functions that satisfy $\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2}$ at all points of V and if $f_V = 0$, then equation (2) leads to:

$$W = \oint_S \left(\psi \frac{\partial u}{\partial n} - \frac{\partial \psi}{\partial n} u \right) dS = 0. \quad (3)$$

The introduction of the boundary conditions finally yields:

$$W = \int_{S_u} \left(\psi \frac{\partial u}{\partial n} - \frac{\partial \psi}{\partial n} u_s \right) dS + \int_{S_f} \left(\psi (f_s - \alpha u) - \frac{\partial \psi}{\partial n} u \right) dS = 0.$$

This expression constitutes the basis for the boundary integral method.

Summary

Formulation	Conditions on \mathbf{u}			Conditions on Ψ		
	Order of derivation	Condition on S_f	Condition on S_u	Order of derivation	Condition on S_f	Condition on S_u
Partial differential equations	2	$\frac{\partial \mathbf{u}}{\partial n} + \alpha \mathbf{u} = \mathbf{f}_s$	$\mathbf{u} = \mathbf{u}_s$	none	none	none
Integral form from Example 3.3	2	$\frac{\partial \mathbf{u}}{\partial n} + \alpha \mathbf{u} = \mathbf{f}_s$	$\mathbf{u} = \mathbf{u}_s$	none	none	none
↓						
Integral form (1)	1	none	$\mathbf{u} = \mathbf{u}_s$	1	none	$\Psi = 0$
↓						
If $\mathbf{f}_V = 0$ Integral form (3) (No volume integral)	none	$\frac{\partial \mathbf{u}}{\partial n} + \alpha \mathbf{u} = \mathbf{f}_s$	$\mathbf{u} = \mathbf{u}_s$	Ψ satisfies $\Delta \Psi \equiv 0$ (no conditions on S)		

For an m -order differential system such as (3.5) and for its integral form (3.7), the acceptable functions \mathbf{u} must be m -times differentiable and must satisfy all the boundary conditions. After s rounds of integration by parts, we can choose the following conditions on \mathbf{u} and Ψ :

- \mathbf{u} must be differentiable $m - s$ times;
- Ψ must be differentiable s times;
- \mathbf{u} satisfies only the boundary conditions containing derivatives up to the order $m - s - 1$;
- Ψ is zero on the boundaries on which \mathbf{u} has to satisfy the previous boundary conditions.

The boundary conditions that contain derivatives whose order is greater than or equal to $m - s$ are then taken into account in the integral formulation; they are satisfied in the weak sense.

3.3.3 CONSTRUCTION OF ADDITIONAL INTEGRAL FORMS

In practice, the system of equations $\mathcal{L}(\mathbf{u}) + \mathbf{f}_V = 0$ is often constructed by elimination of a number of variables \mathbf{q} , involving various field equations:

$$\begin{aligned} \mathbf{L}_e(\mathbf{q}) - \mathbf{f}_V &= 0: \text{equilibrium or conservation laws} \\ \mathbf{L}_c(\mathbf{q}, \mathbf{u}) &= 0: \text{constitutive laws} \end{aligned} \quad (3.11)$$

The operator \mathcal{L} is generally of a higher order than \mathcal{L}_e and \mathcal{L}_c .

It is sometimes helpful to construct integral forms directly based on (3.11) to make the variables \mathbf{q} appear explicitly as unknown, and to decrease the derivability conditions of \mathbf{u} .

EXAMPLE 3.6. Construction of the Poisson equation

A two-dimensional heat flow problem may be written as:

— the conservation of the heat flux \mathbf{q} , f_V being a source of heat per unit volume

$$\mathcal{L}_e(\mathbf{q}) - f_V = \frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} - f_V = 0$$

— the relation between heat flux and temperature gradient (Fourier's Law)

$$\mathcal{L}_c(\mathbf{q}, \mathbf{u}) = 0 \begin{cases} q_x + \frac{\partial u}{\partial x} = 0 \\ q_y + \frac{\partial u}{\partial y} = 0 \end{cases}$$

where $u(x, y)$ is the temperature at point (x, y) .

Poisson's equation is obtained by eliminating q_x and q_y :

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f_V = 0.$$

Let us apply the weighted residual method directly to the operators $(\mathcal{L}_e, \mathbf{f}_V)$ and \mathcal{L}_c to obtain a mixed integral form:

$$W_r = \int_V \langle \psi_u \rangle \{ \mathbf{L}_e(\mathbf{q}) - f_V \} dV + \int_V \langle \psi_q \rangle \{ \mathbf{L}_c(\mathbf{q}, \mathbf{u}) \} dV = 0 \quad (3.12a)$$

Because ψ_u and ψ_q are independent:

$$\begin{aligned} \int_V \langle \psi_u \rangle \{ L_e(\mathbf{q}) - f_V \} dV &= 0 \\ \int_V \langle \psi_q \rangle \{ L_e(\mathbf{q}, \mathbf{u}) \} dV &= 0 \end{aligned} \quad (3.12b)$$

where \mathbf{u} and \mathbf{q} satisfy all the boundary conditions on S_u and S_f .

EXAMPLE 3.7. Mixed integral form for Poisson's equation

$$\begin{aligned} W_r = \int_V \left(\psi_u \left(\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} - f_V \right) + \psi_{q_x} \left(q_x + \frac{\partial u}{\partial x} \right) \right. \\ \left. + \psi_{q_y} \left(q_y + \frac{\partial u}{\partial y} \right) \right) dV = 0. \end{aligned}$$

As weighting functions, let us use Galerkin functions, i.e. of the same nature as u , q_x and q_y , and denoted as δu , δq_x and δq_y :

$$\psi_u = \delta u, \quad \psi_{q_x} = \delta q_x, \quad \psi_{q_y} = \delta q_y.$$

Then:

$$\begin{aligned} W_r = \int_V \left(\delta u \left(\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} - f \right) + \delta q_x \left(q_x + \frac{\partial u}{\partial x} \right) \right. \\ \left. + \delta q_y \left(q_y + \frac{\partial u}{\partial y} \right) \right) dV = 0, \end{aligned}$$

where $u = u_s$ on S_u and $\frac{\partial u}{\partial n} = f_s - \alpha u$ on S_f .

3.4 Functionals [MIK 64; MIK 71]

We will show that using the weighted residual method is equivalent to rendering a functional stationary in certain cases. For instance, in the case of solid mechanics, this functional might be the total potential energy of the system. This enables us to directly obtain an integral formulation from the functional's stationarity conditions, which is helpful when it is simpler to express the energy functional than to express the partial differential equations (3.5).

3.4.1 FIRST VARIATION

A functional π is a function of an ensemble of functions and their derivatives:

$$\pi = \pi\left(\mathbf{u}, \frac{\partial \mathbf{u}}{\partial x}, \dots\right). \quad (3.13a)$$

If π is differentiable, its first variation is defined by:

$$\delta\pi = \frac{\partial\pi}{\partial \mathbf{u}} \delta\mathbf{u} + \frac{\partial\pi}{\partial\left(\frac{\partial \mathbf{u}}{\partial x}\right)} \delta\left(\frac{\partial \mathbf{u}}{\partial x}\right) + \dots \quad (3.13b)$$

where: $\delta\mathbf{u}$, $\delta\left(\frac{\partial \mathbf{u}}{\partial x}\right)$ are arbitrary variations of \mathbf{u} and $\frac{\partial \mathbf{u}}{\partial x}$

$$\frac{\partial\pi}{\partial \mathbf{u}}$$

$$\frac{\partial\pi}{\partial\left(\frac{\partial \mathbf{u}}{\partial x}\right)}$$

The variation operator δ has the following properties:

$$\begin{aligned} \delta\left(\frac{\partial \mathbf{u}}{\partial x}\right) &= \frac{\partial(\delta\mathbf{u})}{\partial x} \\ \delta(\delta\mathbf{u}) &= 0 \\ \delta\left(\int_V u \, dV\right) &= \int_V \delta u \, dV \end{aligned} \quad (3.14)$$

$$\delta(u + v) = \delta u + \delta v$$

$$\delta(u v) = u \delta v + v \delta u = \delta(v u)$$

$$\delta(c u) = c \delta u \quad (c = \text{constant}).$$

If $u = \sum N_i(x)u_i$, we then have $\delta u = \sum N_i(x)\delta u_i$, with the variation δu being expressed as a function of its nodal values δu_i .

EXAMPLE 3.8. One-dimensional functional

Consider the following one-dimensional functional:

$$\pi\left(u, \frac{du}{dx}\right) = \int_{x_1}^{x_2} \left(\frac{1}{2} \left(\frac{du}{dx} \right)^2 - u f \right) dx, \quad f \text{ is constant.}$$

Hence, its first variation (3.13b) is:

$$\delta\pi = \delta \int_{x_1}^{x_2} \left(\frac{1}{2} \left(\frac{du}{dx} \right)^2 - u f \right) dx$$

or, if we use the properties (3.14):

$$\delta\pi = \int_{x_1}^{x_2} \left(\delta \left(\frac{du}{dx} \right) \frac{du}{dx} - \delta u f \right) dx = \int_{x_1}^{x_2} \left(\frac{d(\delta u)}{dx} \frac{du}{dx} - \delta u f \right) dx.$$

The second variation of π :

$$\delta^2\pi = \delta(\delta\pi) = \int_{x_1}^{x_2} \left(\delta \left(\frac{du}{dx} \right) \right)^2 dx = \int_{x_1}^{x_2} \left(\frac{d(\delta u)}{dx} \right)^2 dx$$

because

$$\delta(\delta u) = 0.$$

3.4.2 FUNCTIONAL ASSOCIATED WITH AN INTEGRAL FORM

For certain problems defined by (3.5a) and (3.5b), it is possible to construct a functional $\pi\left(\mathbf{u}, \frac{\partial \mathbf{u}}{\partial x}, \dots\right)$ such that:

$$\delta\pi \equiv W = 0 \tag{3.15a}$$

where W is a particular integral form, called a Galerkin form, obtained by choosing $\psi = \delta\mathbf{u}$ in equation (3.7) and integrating by parts if necessary:

$$W = \int_V \langle \delta\mathbf{u} \rangle \{ \mathcal{L}(\mathbf{u}) + f_V \} dV = 0. \tag{3.15b}$$

In particular, this is possible if:

- \mathcal{L} and \mathcal{C} are linear and all the derivatives of \mathcal{L} are of an even-numbered order;
- f_S and f_V are independent of \mathbf{u} .

These conditions are sufficient for a functional to exist, but are not necessary.

EXAMPLE 3.9. Functional for Poisson's equation

$$\mathcal{L}(u) + f_V = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f_V = 0.$$

This operator contains only second-order derivatives; f_V is constant. Thus a functional can be associated with this problem.

The integral form (1) from Example 3.5 is:

$$W = \int_V \left(\frac{\partial \psi}{\partial x} \frac{\partial u}{\partial x} + \frac{\partial \psi}{\partial y} \frac{\partial u}{\partial y} - \psi f_v \right) dV + \int_{S_f} \psi (\alpha u - f_s) dS = 0.$$

By choosing $\psi \equiv \delta u$, as a weighting function, we get:

$$\begin{aligned} W &= \int_V \left(\frac{\partial(\delta u)}{\partial x} \frac{\partial u}{\partial x} + \frac{\partial(\delta u)}{\partial y} \frac{\partial u}{\partial y} - \delta u f_v \right) dV \\ &\quad + \int_{S_f} \delta u (\alpha u - f_s) dS = 0. \end{aligned}$$

If we define a functional π in the form:

$$\begin{aligned} \pi \left(u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y} \right) &= \int_V \left(\frac{1}{2} \left(\frac{\partial u}{\partial x} \right)^2 + \frac{1}{2} \left(\frac{\partial u}{\partial y} \right)^2 - u f_v \right) dV \\ &\quad + \int_{S_f} \left(\frac{1}{2} \alpha u^2 - u f_s \right) dS \end{aligned}$$

we can readily verify that:

$$\delta \pi \equiv W = 0.$$

Equation (3.15) can be considered a stationarity condition for the functional π : a solution \mathbf{u} that renders W null also renders the functional π stationary at that point. This is minimum, maximum or stationary depending on whether the second variation $\delta^2 \pi$ is positive, negative or null for all weighting functions δu :

$$\delta^2 \pi \left(\mathbf{u}, \frac{\partial \mathbf{u}}{\partial x}, \dots \right) = \frac{\partial^2 \pi}{\partial \mathbf{u}^2} \delta \mathbf{u} \delta \mathbf{u} + \frac{\partial^2 \pi}{\partial \left(\frac{\partial \mathbf{u}}{\partial x} \right)^2} \delta \left(\frac{\partial \mathbf{u}}{\partial x} \right) \delta \left(\frac{\partial \mathbf{u}}{\partial x} \right) + \dots \quad (3.16)$$

For instance, the second variation of π in Example 3.9 is:

$$\delta^2 \pi = \int_V \left(\left(\frac{\partial(\delta u)}{\partial x} \right)^2 + \left(\frac{\partial(\delta u)}{\partial y} \right)^2 \right) dV + \int_{S_f} \alpha(\delta u)^2 dS.$$

This value is always positive for a non-null δu and a positive α . The solution \mathbf{u} of $\delta \pi = W = 0$ therefore renders π minimum.

A function $\pi \left(\mathbf{u}, \frac{\partial \mathbf{u}}{\partial x}, \dots \right)$ is said to be **linear** if its expression is linear in $\mathbf{u}, \frac{\partial \mathbf{u}}{\partial x}, \dots$; for example:

$$\pi \left(\mathbf{u}, \frac{\partial \mathbf{u}}{\partial x} \right) = \int_V \left(a_1 u + a_2 \frac{\partial u}{\partial x} \right) dV. \quad (3.17)$$

A functional is said to be **quadratic** if its expression is quadratic in $\mathbf{u}, \frac{\partial \mathbf{u}}{\partial x}, \dots$; for example:

$$\pi = \int_V \left(a_1 \left(\frac{\partial u}{\partial x} \right)^2 + a_2 u^2 \right) dV \quad (3.18)$$

In practice, we sometimes use the term “quadratic functional” to denote a functional that has both a quadratic and a linear part to it. A purely quadratic function can be written in matrix form:

$$\pi = \frac{1}{2} \int_V \langle \mathbf{u} \frac{\partial \mathbf{u}}{\partial x} \dots \rangle [D] \begin{Bmatrix} \mathbf{u} \\ \frac{\partial \mathbf{u}}{\partial x} \\ \vdots \end{Bmatrix} dV \quad (3.19)$$

where $[D]$ is a symmetrical matrix, independent of \mathbf{u} .

The first and second variations are then given by:

$$\delta \pi = \int_V \langle \delta \mathbf{u} \frac{\partial(\delta \mathbf{u})}{\partial x} \dots \rangle [D] \begin{Bmatrix} \mathbf{u} \\ \frac{\partial \mathbf{u}}{\partial x} \\ \vdots \end{Bmatrix} dV \quad (3.20a)$$

$$\delta^2 \pi = \int_V \langle \delta u \frac{\partial(\delta u)}{\partial x} \dots \rangle [D] \begin{Bmatrix} \delta u \\ \frac{\partial(\delta u)}{\partial x} \\ \vdots \end{Bmatrix} dV. \quad (3.20b)$$

If the matrix $[D]$ is positive definite, i.e. if all its eigenvalues are positive, then the second variation $\delta^2 \pi$ is also positive.

Let u_{ex} be the exact solution such that $\delta \pi_{ex} = 0$. Consider a function u that is not a solution to the problem, and let $\delta u = u - u_{ex}$. We can write the quadratic form:

$$\pi(u) = \pi(u_{ex}) + \delta\pi(u_{ex}) + \delta^2\pi(u_{ex}) \text{ where } \delta\pi(u_{ex}) = 0.$$

If $\delta^2\pi(u) > 0$, $\forall \delta u$, then $\pi(u) = \pi(u_{ex}) + \delta^2\pi > \pi(u_{ex})$.

The functional allows a minimum in u_{ex} .

EXAMPLE 3.10. Minimum of a quadratic form

Consider the algebraic relation:

$$k.u_{ex} - f = 0, \quad u_{ex} = f/k \quad (A)$$

We associate the following quadratic form with it:

$$\pi = \frac{1}{2} k.u^2 - u.f \text{ so that } \delta\pi = \delta u(k.u - f)$$

such that (A) corresponds to $\delta\pi = 0$, $\forall \delta u$.

The expression of π in the vicinity of u_{ex} can be written as:

$$\pi(u) = \pi(u_{ex}) + \delta\pi|_{u_{ex}} + \delta^2\pi$$

$$\text{so that } \pi(u) = -\frac{1}{2}k.u_{ex}^2 - u_{ex}.f + \frac{k}{2}.\delta u^2 = -\frac{1}{2}\frac{f^2}{k} + \frac{k}{2}.\delta u^2,$$

$$\text{where } \delta u = u - u_{ex} \text{ and } \delta\pi|_{u_{ex}} = 0,$$

$$\pi(u) > \left(\pi|_{u_{ex}} = -\frac{1}{2}\frac{f^2}{k} \right), \quad \forall \delta u.$$

The functional allows a minimum in $u_{ex} = f/k$ whose value is $-f^2/(2k)$.

EXAMPLE 3.11. Matrix form of the functional formulation of Poisson's equation

The functional π from Example 3.9 can be written in matrix form:

$$\begin{aligned}\pi = & \frac{1}{2} \int_V \left\langle \frac{\partial u}{\partial x} \frac{\partial u}{\partial y} \right\rangle \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{cases} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{cases} dV - \int_V u f_V dV \\ & + \int_{S_f} \left(\frac{\alpha u^2}{2} - u f_s \right) dS.\end{aligned}$$

Its first variation is:

$$\begin{aligned}W = \delta\pi = & \int_V \left\langle \frac{\partial(\delta u)}{\partial x} \frac{\partial(\delta u)}{\partial y} \right\rangle \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{cases} \frac{\partial(\delta u)}{\partial x} \\ \frac{\partial(\delta u)}{\partial y} \end{cases} dV - \int_V \delta u f_v dV \\ & + \int_{S_f} (\delta u \alpha u - \delta u f_s) dS.\end{aligned}$$

Its second variation is:

$$\begin{aligned}\delta W = \delta^2 \pi = & \int_V \left\langle \frac{\partial(\delta u)}{\partial x} \frac{\partial(\delta u)}{\partial y} \right\rangle \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{cases} \frac{\partial(\delta u)}{\partial x} \\ \frac{\partial(\delta u)}{\partial y} \end{cases} dV \\ & + \int_{S_f} \alpha(\delta u)^2 dS.\end{aligned}$$

In this case, $[D] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. This matrix is positive definite because its eigenvalues are equal to 1. Consequently, $\delta^2 \pi \geq 0$.

3.4.3 STATIONARITY PRINCIPLE

Let us write the partial differential equations (3.5), separating the boundary into two parts; the condition on S_f and S_u :

$$\mathcal{L}(\mathbf{u}) + \mathbf{f}_V = 0 \quad \text{on } V \tag{3.21a}$$

$$\mathcal{C}_f(\mathbf{u}) = \mathbf{f}_s \quad \text{on } S_f \tag{3.21b}$$

$$\mathcal{C}_u(\mathbf{u}) = \mathbf{f}_u \quad \text{on } S_u. \tag{3.21c}$$

The integral form obtained using the weighted residual method is:

$$W(\mathbf{u}) = \int_V \langle \psi \rangle \{ \mathcal{L}(\mathbf{u}) + f_V \} dV = 0 \quad \text{for any value of } \psi \quad (3.22)$$

where:

$$\mathcal{C}_f(\mathbf{u}) = \mathbf{f}_S \quad \text{on } S_f$$

$$\mathcal{C}_u(\mathbf{u}) = \mathbf{f}_u \quad \text{on } S_u.$$

By choosing $\psi = \delta \mathbf{u}$ and integrating by parts, we can in certain cases (with the so-called **conservative** systems) construct a functional π that is stationary for the solution \mathbf{u} sought:

$$\delta\pi(\mathbf{u}) \equiv W(\mathbf{u}) = 0 \quad (3.23)$$

where:

$$\mathcal{C}_u(\mathbf{u}) = f_u \quad \text{on } S_u. \quad (3.24)$$

The **stationarity principle** can be stated thus:

Out of all the possible solution functions \mathbf{u} (derivability and boundary conditions on S_u), the one that satisfies equations (3.21a and 3.21b) renders the functional π stationary.

3.4.4 LAGRANGE MULTIPLIERS AND ADDITIONAL FUNCTIONALS

In the functional π , the only unknown variables in the problem are the functions \mathbf{u} that must satisfy continuity conditions and boundary conditions on S_u .

The **Lagrange multiplier method** can be used to construct other functionals π^* whose stationarity conditions constitute new integral formulations that may exhibit the following characteristics:

- introduction of additional physical variables as unknowns;
- less strict derivability conditions and boundary conditions on \mathbf{u} .

Let us first introduce the concept of a Lagrange multiplier, using a simple example.

EXAMPLE 3.12. Lagrange multiplier

The extremum of the function:

$$\pi_1(u, v) = u^2 + v^2$$

is defined by the condition:

$$\delta\pi_1 = 2u\delta u + 2v\delta v = 0 \text{ for any value of } \delta u \text{ and } \delta v.$$

Hence $u = v = 0 \quad \pi_1(0, 0) = 0.$

Suppose we are looking for the minimum of π_1 with the condition:

$$g(u, v) = u - v + 2 = 0.$$

One method is to use $g(u, v) = 0$ to eliminate v from the expression of π_1 :

$$\pi(u) = 2u^2 + 4u + 4$$

$$\delta\pi = 4(u+1)\delta u = 0$$

$$u = -1 \quad v = 1 \quad \pi(-1, 1) = 2.$$

The Lagrange multiplier method consists of rendering the following expression stationary:

$$\pi^*(u, v, \lambda) = \pi_1(u, v) + \lambda g(u, v) = u^2 + v^2 + \lambda(u - v + 2)$$

where λ is the Lagrange multiplier corresponding to the condition $g = 0$.

The stationarity condition is written as:

$$\delta\pi^* = \frac{\partial\pi^*}{\partial u}\delta u + \frac{\partial\pi^*}{\partial v}\delta v + \frac{\partial\pi^*}{\partial\lambda}\delta\lambda = 0 \text{ for any value of } \delta u, \delta v \text{ and } \delta\lambda$$

so:

$$\delta\pi^* = (2u + \lambda)\delta u + (2v - \lambda)\delta v + (u - v + 2)\delta\lambda = 0$$

Hence: $2u + \lambda = 0 \quad u = -1$

$$2v - \lambda = 0 \quad \text{so} \quad v = 1$$

$$u - v + 2 = 0 \quad \lambda = 2.$$

This method avoids the elimination but here leads to a function π^* with three variables u, v and λ , whereas the function π is dependent only upon u and v .

To generalize the results from the above example we look for functions \mathbf{u} that will render a functional $\pi_1(\mathbf{u}, \mathbf{q})$ stationary, subject to the conditions:

$$\begin{aligned} g_1(\mathbf{u}, \mathbf{q}) &= 0 \\ g_2(\mathbf{u}, \mathbf{q}) &= 0 \\ \dots &\quad \text{on } V \\ g_m(\mathbf{u}, \mathbf{q}) &= 0 \end{aligned} \tag{3.25}$$

where \mathbf{q} are the physical variables such as loads and flow rates.

One method is to eliminate m variables among \mathbf{u} and \mathbf{q} in the functional π_1 using (3.25). For example, if the number of relations (3.25) is equal to the number of variables \mathbf{q} , we can obtain a functional π that is dependent only on \mathbf{u} .

The “Lagrange multiplier” method consists of introducing m Lagrange multipliers $\lambda_1, \lambda_2, \dots, \lambda_m$, and rendering the generalized functional stationary:

$$\pi^*(\mathbf{u}, \mathbf{q}, \boldsymbol{\lambda}) = \pi_1(\mathbf{u}, \mathbf{q}) + \int_V (\lambda_1 g_1(\mathbf{u}, \mathbf{q}) + \dots + \lambda_m g_m(\mathbf{u}, \mathbf{q})) dV. \tag{3.26}$$

The stationarity conditions of π^* include the conditions (3.25):

$$\frac{\partial \pi^*}{\partial \mathbf{u}} = 0; \quad \frac{\partial \pi^*}{\partial \mathbf{q}} = 0; \quad \frac{\partial \pi^*}{\partial \lambda_i} = g_i = 0, \quad i = 1, 2, \dots, m. \tag{3.27}$$

The unknown functions pass from \mathbf{u}, \mathbf{q} to \mathbf{u}, \mathbf{q} and $\boldsymbol{\lambda}$. The functional π^* is not positive definite, even if π is. Figure 3.4 illustrates the relationships between $\delta\pi$, $\delta\pi_1$ and $\delta\pi^*$.

EXAMPLE 3.13. Generalized functional for Poisson's equations

Consider the functional from Example (3.9):

$$\pi(u) = \int_V \left(\frac{1}{2} \left(\frac{\partial u}{\partial x} \right)^2 + \frac{1}{2} \left(\frac{\partial u}{\partial y} \right)^2 - u f_V \right) dV + \int_{S_f} \left(\frac{1}{2} \alpha u^2 - u f_S \right) dS.$$

It can also be written with the two variables q_x and q_y (see Example 3.6):

$$\begin{aligned}\pi_1(u, q_x, q_y) = & \int_V \left(\frac{1}{2} (q_x^2 + q_y^2) - u f_V \right) dV \\ & + \int_{S_f} \left(\frac{1}{2} \alpha u^2 - u f_S \right) dS\end{aligned}$$

with the following conditions:

$$\begin{aligned}q_x + \frac{\partial u}{\partial x} &= 0 \\ q_y + \frac{\partial u}{\partial y} &= 0.\end{aligned}$$

Rendering π stationary is equivalent to rendering π_1 stationary conditionally. By eliminating q_x and q_y in π_1 using the two conditions, we get π . We can also use the Lagrange multiplier method to define the modified functional:

$$\begin{aligned}\pi^*(u, q_x, q_y, \lambda_1, \lambda_2) = & \int_V \left(\frac{1}{2} (q_x^2 + q_y^2) - u f_V + \lambda_1 \left(q_x + \frac{\partial u}{\partial x} \right) \right. \\ & \left. + \lambda_2 \left(q_y + \frac{\partial u}{\partial y} \right) \right) dV + \int_{S_f} \left(\frac{1}{2} \alpha u^2 - u f_S \right) dS.\end{aligned}$$

The stationarity condition of π^* is:

$$\delta \pi^* = \frac{\partial \pi^*}{\partial u} \delta u + \frac{\partial \pi^*}{\partial q_x} \delta q_x + \frac{\partial \pi^*}{\partial q_y} \delta q_y + \frac{\partial \pi^*}{\partial \lambda_1} \delta \lambda_1 + \frac{\partial \pi^*}{\partial \lambda_2} \delta \lambda_2 = 0.$$

Using integration by parts, we get the stationarity conditions:

$$\begin{aligned}\frac{\partial \lambda_1}{\partial x} + \frac{\partial \lambda_2}{\partial y} + f_v &= 0; \\ q_x + \lambda_1 &= 0; \quad q_y + \lambda_2 = 0; \\ q_x + \frac{\partial u}{\partial x} &= 0; \quad q_y + \frac{\partial u}{\partial y} = 0; \\ \text{and } \alpha.u - f_s + \lambda_1.l + \lambda_2.m &= 0 \quad \text{on } S_f.\end{aligned}$$

where l and m are the direction cosines of the normal to S_f .

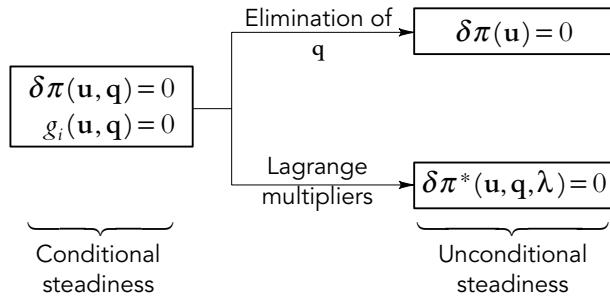


Figure 3.4. Transformation of a functional with conditions into another functional with no conditions

Lagrange multipliers often have physical meaning: for instance, they might be flow rates, fluxes, stresses, etc.

We can construct **mixed functionals** π_r by eliminating the Lagrange multipliers from the functional π^* , using equations (3.27). In structural mechanics, the so-called Hellinger–Reissner functionals [WAS 82] are mixed functionals.

EXAMPLE 3.14. “Mixed” functional from Example 3.12

Let us use one of the stationarity conditions of π^* from Example 3.12 to express λ in the form:

$$\lambda = 2v.$$

The “mixed” function is obtained by substituting the expression of λ into the functional π^* from Example 3.12, so that:

$$\pi_r(u, v) = u^2 - v^2 + 2uv + 4v.$$

The stationarity conditions of π_r are thus: $u = -1$; $v = 1$

and

$$\pi_r(-1, 1) = 2.$$

EXAMPLE 3.15. Mixed functional for Poisson’s equation

Let us use the relations $\frac{\partial \pi^*}{\partial q_x} = 0$ and $\frac{\partial \pi^*}{\partial q_y} = 0$ from Example 3.13 to eliminate λ_1 and λ_2 in π^* :

$$\lambda_1 = -q_x; \quad \lambda_2 = -q_y.$$

The functional π^* becomes the mixed functional π_r :

$$\begin{aligned}\pi_r(u, q_x, q_y) = & - \int_V \left(\frac{1}{2} (q_x^2 + q_y^2) + q_x \frac{\partial u}{\partial x} + q_y \frac{\partial u}{\partial y} + u f_V \right) dV \\ & + \int_{S_f} \left(\frac{1}{2} \alpha u^2 - u f_S \right) dS.\end{aligned}$$

After integration by parts of $\frac{\partial \pi_r}{\partial u} \delta u$, the stationarity conditions are:

$$\begin{aligned}\frac{\partial \pi_r}{\partial q_x} \delta q_x &= - \int_V \left(q_x + \frac{\partial u}{\partial x} \right) \delta q_x dV = 0 \\ \frac{\partial \pi_r}{\partial q_y} \delta q_y &= - \int_V \left(q_y + \frac{\partial u}{\partial y} \right) \delta q_y dV = 0 \\ \frac{\partial \pi_r}{\partial u} \delta u &= \int_V \left(\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} - f_V \right) \delta u dV + \int_{S_f} (\alpha u - f_S - q_n) \delta u dS = 0\end{aligned}$$

where

$$q_n = q_x l + q_y m$$

$$u = u_S; \delta u = 0 \text{ on } S_u.$$

Another form of π_r is obtained using integration by parts of the terms $q_x \frac{\partial u}{\partial x}$ and $q_y \frac{\partial u}{\partial y}$:

$$\begin{aligned}\pi_r^*(u, q_x, q_y) = & - \int_V \left(\frac{1}{2} (q_x^2 + q_y^2) - \left(\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} - f_V \right) u \right) dV \\ & + \int_{S_f} \left(\frac{1}{2} \alpha u^2 - u f_S - u q_n \right) dS - \int_{S_u} q_n u_s dS.\end{aligned}$$

In the functional π in Example 3.9, the only variable is u . However, the functionals π_r and π_r^* depend on three independent variables u , q_x and q_y .

A complementary functional π_c can also be obtained after eliminating \mathbf{u} from π_r by explicitly satisfying constraints in \mathbf{q} .

EXAMPLE 3.16. Complementary functional for Poisson's equation, when $\alpha = 0$

If q_x and q_y are chosen to satisfy the following conditions:

$$\begin{aligned}\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} - f_V &= 0 \quad \text{on } V \\ q_n + f_S &= 0 \quad \text{on } S_f\end{aligned}$$

the functional π_r^* from Example 3.14 becomes the complementary functional

$$\pi_c = - \int_V \frac{1}{2} (q_x^2 + q_y^2) dV - \int_{S_u} q_n u_s dS.$$

This functional depends only on the two variables q_x and q_y . Its stationarity conditions are:

$$\begin{aligned}\frac{\partial \pi_c}{\partial q_x} \delta q_x &= - \int_V \delta q_x \cdot q_x dV - \int_{S_u} \delta q_x \cdot l u_s dS = 0 \\ \frac{\partial \pi_c}{\partial q_y} \delta q_y &= - \int_V \delta q_y \cdot q_y dV - \int_{S_u} \delta q_y \cdot m u_s dS = 0.\end{aligned}$$

Figure 3.5 shows the relationships between π , π^* , π_r and π_c and their interpretation in linear elasticity.

3.5 Discretization of integral forms

3.5.1 DISCRETIZATION OF W

In sections 3.2 and 3.3, we replaced the solution of the partial differential equations (3.5) with the search for a solution function \mathbf{u} that nullifies the integral form (3.7):

$$W = \int_V \psi \cdot R(\mathbf{u}) dV = \int_V \psi \cdot (\mathcal{L}(\mathbf{u}) + f_V) dV = 0 \quad (3.28)$$

for any weighting function ψ .

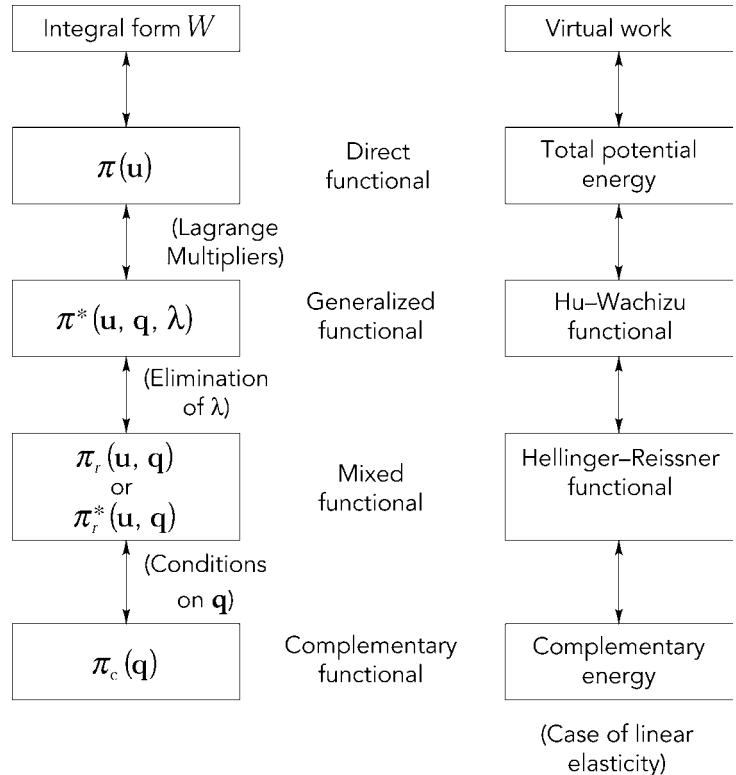


Figure 3.5. Various types of functionals

In order to construct an approximate solution \mathbf{u} , we discretize (3.28) in two stages:

- Choose an approximation of the unknown functions \mathbf{u} with n parameters. This approximation may be nodal or non-nodal, over the entire domain or over the subdomains (see section 1.1). The **undetermined coefficient method** [CRA 56] uses the non-nodal approximation (1.3). The finite element method uses the finite element approximation described in section 1.1.2. In all cases, \mathbf{u} can be written as:

$$\mathbf{u} = \mathbf{u}(a_1, a_2, \dots, a_n). \quad (3.29)$$

Expression (3.28) becomes:

$$W = \int_V \psi \cdot (\mathcal{L}(u(a_1, a_2, \dots, a_n) + f_V)) \, dV = 0 \quad (3.30)$$

for any ψ .

- Choose a set of n independent weighting functions $\psi_1, \psi_2, \dots, \psi_n$. Let us stress that the number of weighting functions must be equal to the number of parameters in approximation (3.29). The choice of the type of functions ψ_i leads to different methods: collocation, Galerkin (the most widely used), least squares, etc. Equations (3.30) can be written as:

$$\begin{aligned} W_1 &= \int_V \psi_1(\mathcal{L}(\mathbf{u}(a_1, a_2, \dots, a_n) + f_V)) dV = 0 \\ W_2 &= \int_V \psi_2(\mathcal{L}(\mathbf{u}(a_1, a_2, \dots, a_n) + f_V)) dV = 0 \\ &\dots \\ W_n &= \int_V \psi_n(\mathcal{L}(\mathbf{u}(a_1, a_2, \dots, a_n) + f_V)) dV = 0 \end{aligned} \quad (3.31)$$

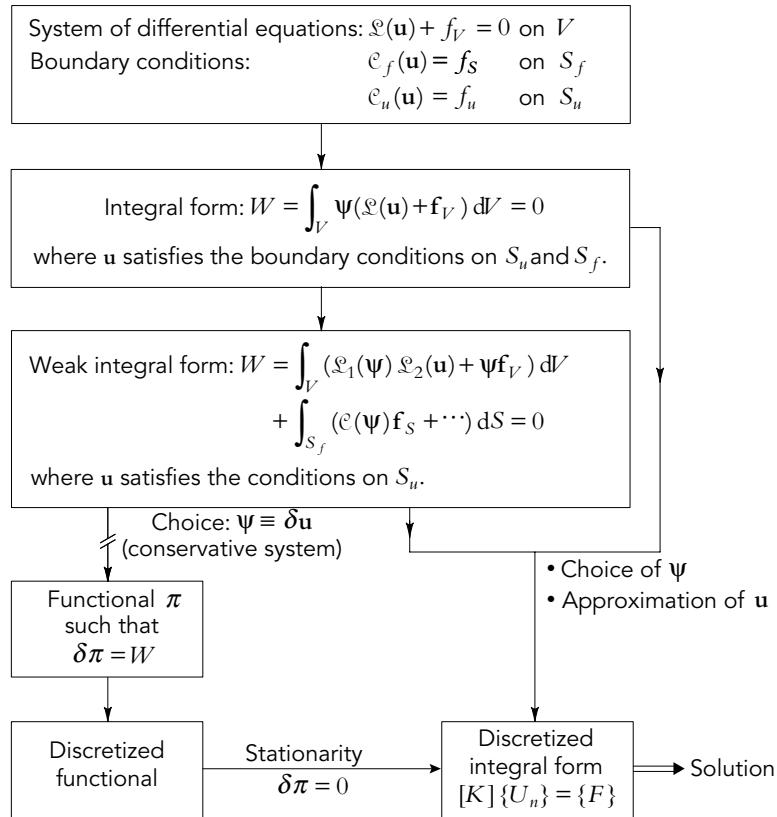


Figure 3.6. Relationships between systems of differential equations, integral forms and functionals

After integration, we obtain a system of algebraic equations whose solution gives us the parameters of the approximation of \mathbf{u} :

$$[K]\{a\} = \{F\}. \quad (3.32)$$

Figure 3.6 summarizes the various operations necessary to obtain an approximate solution using the weighted residual method.

3.5.2 APPROXIMATION OF THE FUNCTIONS \mathbf{u}

The functions \mathbf{u} may be approximated by one of the methods described in Chapter 1, satisfying the **derivability conditions** and the **boundary conditions** required by the integral form being used:

- Non-nodal approximation over the entire domain V (relation (1.3)):

$$\mathbf{u} = \mathbf{u}(\mathbf{x}, a_1, a_2, \dots, a_n) = \langle P_1 \quad P_2 \quad \dots \quad P_n \rangle \begin{Bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{Bmatrix} \quad (3.33)$$

- Nodal approximation over the entire domain V (relation (1.5))

$$\mathbf{u} = \mathbf{u}(\mathbf{x}, u_1, u_2, \dots, u_n) = \langle N_1 \quad N_2 \quad \dots \quad N_n \rangle \begin{Bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{Bmatrix} \quad (3.34)$$

- Finite element approximation (section 1.1.2).

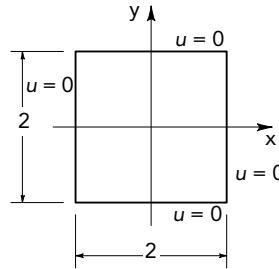
In the following examples, we use non-nodal approximations on V such as (3.33) to illustrate the methods corresponding to the various choices of weighting function Ψ . For the rest of this book, we will use finite element approximations.

EXAMPLE 3.17. Non-nodal approximation of u over a square

Consider the Poisson equation defined over a square ($f_V = f = \text{constant}$):

$$\mathcal{L}(u) + f_V = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f = 0 \quad \text{on the square}$$

$$u = 0 \quad \text{on } S \quad \begin{cases} x = \pm 1 \\ y = \pm 1. \end{cases}$$



An approximation of u which satisfies the boundary conditions and the symmetry of the problem is:

$$u = \langle P_1 \quad P_2 \rangle \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} = \langle P \rangle \{a\}$$

where

$$\begin{aligned} P_1 &= (x^2 - 1)(y^2 - 1) \\ P_2 &= (x^2 - 1)(y^2 - 1)(x^2 + y^2) = P_1(x^2 + y^2). \end{aligned}$$

Hence:

$$\mathcal{L}(u) = \mathcal{L}(\langle P \rangle \{a\}) = \mathcal{L}(P_1) a_1 + \mathcal{L}(P_2) a_2$$

with

$$\begin{aligned} \mathcal{L}(P_1) &= 2(x^2 + y^2 - 2) \\ L(P_2) &= 2(6x^2 - 1)(y^2 - 1) + 2(6y^2 - 1)(x^2 - 1) + \\ &\quad + 2(x^4 - x^2) + 2(y^4 - y^2). \end{aligned}$$

3.5.3 CHOICE OF THE WEIGHTING FUNCTIONS ψ

Depending on the choice of ψ , equation (3.31) leads us to different methods, which are illustrated in Figure 3.7.

3.5.3.1 Point collocation method

The function $\psi_i(\mathbf{x})$ is the Dirac distribution $\delta(\mathbf{x}_i)$ at the point \mathbf{x}_i , called the collocation point. The integral form (3.7) is written as:

$$W = \int_V \delta(\mathbf{x}_i) R(\mathbf{x}, \mathbf{u}) dV = R(\mathbf{x}_i, \mathbf{u}) = 0. \quad (3.35)$$

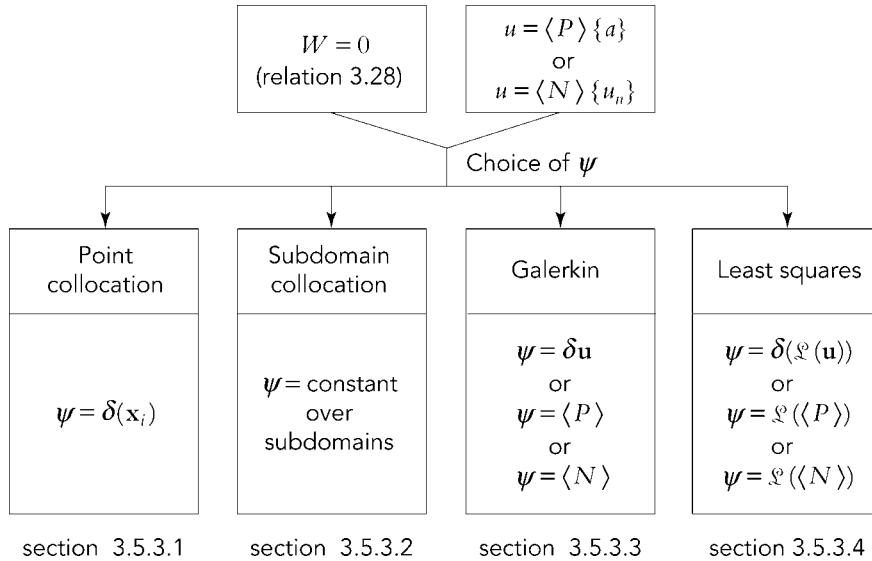


Figure 3.7. Different undetermined coefficient methods depending on the choice of ψ

Equation (3.31) becomes:

$$W_i(\mathbf{a}) = (\mathcal{L}(\mathbf{u}(\mathbf{x}, a_1, a_2, \dots, a_n)) + \mathbf{f}_V)_{\mathbf{x}=\mathbf{x}_i} = 0 \quad i = 1, 2, \dots, n$$

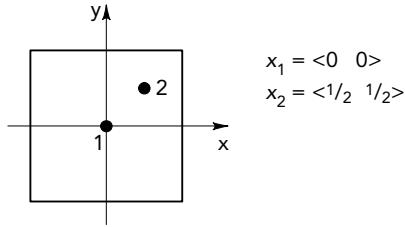
so by using approximation (3.33) of u :

$$\begin{aligned} W_i(\mathbf{a}) &= (\mathcal{L}(\langle P \rangle \{a\}) + f_V)_{\mathbf{x}=\mathbf{x}_i} = 0 \\ &= (\langle \mathcal{L}(P) \rangle \{a\} + f_V)_{\mathbf{x}=\mathbf{x}_i} = 0. \end{aligned} \tag{3.36}$$

The precision of the solution depends on the number and position of the collocation points \mathbf{x}_i , which must respect the symmetry of the problem. The number of collocation points is equal to the number n of parameters a_i . In practice, this method is not often used, because it is difficult to implement in conjunction with a finite element approximation. In addition, it gives us a non-symmetrical system of equations. On the other hand, it does have the advantage of avoiding integration over the volume, which may be interesting for certain nonlinear problems. The quality of the solution can be improved by choosing a number of collocation points greater than n and using the least squares technique.

EXAMPLE 3.18. *Solution of Poisson's equation with the point collocation method*

Let us use point collocation to solve the problem defined in the previous example. The collocation points chosen are:



The weighting functions are:

$$\psi_1 = \delta(\mathbf{x}_1); \quad \psi_2 = \delta(\mathbf{x}_2).$$

With the approximation of u from the previous example, the system of equations (3.36) is written as:

$$W_1 = \langle \mathcal{L}(P_1) \quad \mathcal{L}(P_2) \rangle_{\mathbf{x}=\mathbf{x}_1} \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} + f(\mathbf{x}_1) = 0$$

$$W_2 = \langle \mathcal{L}(P_1) \quad \mathcal{L}(P_2) \rangle_{\mathbf{x}=\mathbf{x}_2} \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} + f(\mathbf{x}_2) = 0.$$

Thus, by using the results from the previous example:

$$W_1 = -4 a_1 + 4 a_2 + f = 0 \quad \begin{cases} a_1 = 0.2976 f \\ a_2 = 0.0476 f \end{cases}$$

$$W_2 = -3 a_1 - \frac{9}{4} a_2 + f = 0$$

The value of u at the center of the square is:

$$u(\mathbf{x}_1) = \langle P_1(\mathbf{x}_1) \quad P_2(\mathbf{x}_1) \rangle \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} = a_1$$

$$u_c = u(\mathbf{x}_1) = 0.2976 f.$$

The "exact" value from a 14-term Fourier series (Example 3.24) is:

$$u_c \approx 0.2947 f.$$

The value of u_c obtained with the approximation using a single parameter $u = P_1(\mathbf{x}) a_1$ would be:

- with the collocation point \mathbf{x}_1 : $u_c = 0.25 f$;
- with the collocation point \mathbf{x}_2 : $u_c \approx 0.333 f$.

3.5.3.2 Subdomain collocation method

Let us choose n subdomains V^i and take the following as a function ψ_i :

$$\psi_i = \begin{cases} 1 & \text{if } \mathbf{x} \text{ belongs to } V^i \\ 0 & \text{if } \mathbf{x} \text{ otherwise } V^i. \end{cases} \quad (3.37)$$

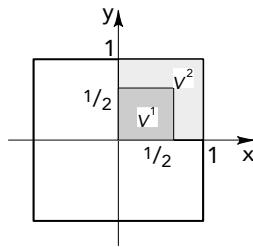
Relation (3.31) is written in the form of n equations:

$$W_i(\mathbf{a}) = \int_{V^i} (\langle \mathcal{L}(P) \rangle \{a\} + f_V) dV = 0. \quad (3.38)$$

The precision of the solution depends on the choice of the subdomains V^i . These must respect the symmetries. The number of subdomains must match the number of parameters a_i . This method is not very widely used, because it is difficult to choose the subdomains. Because it requires integrations over V , it is preferable to employ the Galerkin method.

EXAMPLE 3.19. Solution of Poisson's equation with the subdomain collocation method

We again refer to the problem described in Example 3.17; the following subdomains are selected:



System (3.38) becomes:

$$\begin{aligned} W_1 &= \int_{V^1} \langle L(P_1) - L(P_2) \rangle dV \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} + \int_{V^1} f dV \\ &= -0.9167 a_1 + 0.3875 a_2 + 0.25 f = 0 \end{aligned}$$

$$\begin{aligned} W_2 &= \int_{V^2} \langle L(P_1) - L(P_2) \rangle dV \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} + \int_{V^2} f dV \\ &= -1.75 a_1 - 3.5875 a_2 + 0.75 f = 0. \end{aligned}$$

Hence:

$$a_1 = 0.2994 f$$

$$a_2 = 0.0630 f.$$

The value of u at the center is:

$$u_c = a_1 = 0.2994 f.$$

The value obtained with the one-parameter approximation $u = P_1(x, y) a_1$, and if we integrate over the whole domain, would be $u_c = 0.375 f$.

3.5.3.3 Galerkin method

The weighting functions Ψ comprise all the variations $\delta\mathbf{u}$ of the functions \mathbf{u} :

$$\Psi = \delta\mathbf{u} = \langle P \rangle \{\delta a\} \quad \text{for any value of } \{\delta a\} \quad (3.39)$$

where $\{\delta a\}$ are the first variation of the approximation parameters $\{a\}$.

Equation (3.31) becomes:

$$W = \int_V \delta u (\mathcal{L}(\mathbf{u}) + f_V) dV = 0 \quad (3.40)$$

$$W = \langle \delta a \rangle \int_V \{P\} (L(\langle P \rangle \{a\}) + f_V) dV = 0. \quad (3.41)$$

Because W must vanish for any value of $\{\delta a\}$, the latter equation is equivalent to the n algebraic equations:

$$\begin{aligned} W_1(a) &= \int_V P_1 (\langle \mathcal{L}(P) \rangle \{a\} + f_V) dV = 0 \\ &\vdots \\ W_n(a) &= \int_V P_n (\langle \mathcal{L}(P) \rangle \{a\} + f_V) dV = 0. \end{aligned} \quad (3.42)$$

This system is symmetrical if the operator \mathcal{L} is self-adjoint.

EXAMPLE 3.20. *Solution of Poisson's equation using the Galerkin method without integration by parts*

Using the functions P_1 and P_2 from Example 3.17, we get the following expression of (3.42):

$$\begin{aligned} W_1 &= \int_V \langle P_1 \cdot \mathcal{L}(P_1) - P_1 \cdot \mathcal{L}(P_2) \rangle dV \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} + \int_V P_1 f dV = 0 \\ W_2 &= \int_V \langle P_2 \cdot \mathcal{L}(P_1) - P_2 \cdot \mathcal{L}(P_2) \rangle dV \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} + \int_V P_2 f dV = 0 \end{aligned}$$

From this, we get the symmetrical system:

$$\begin{aligned} 5.689 a_1 + 1.9505 a_2 &= 1.7778 f \\ 1.9505 a_1 + 2.3839 a_2 &= 0.7111 f \\ a_1 &= 0.2922 f \quad a_2 = 0.0592 f \\ u_c &= 0.2922 f. \end{aligned}$$

If we had used a one-parameter solution $u = P_1(x, y) a_1$, we would have obtained:

$$u = 0.3125 f.$$

For a three-parameter solution using the same functions P_1, P_2 and $P_3 = P_1 \cdot x^2 \cdot y^2$, we would have found:

$$\begin{aligned} a_1 &= 0.2949 f \quad a_2 = 0.0401 f \quad a_3 = 0.1230 f, \\ u_c &= 0.2949 f. \end{aligned}$$

Integration by parts can generally be used to transform (3.42), as demonstrated in section 3.3:

$$\begin{aligned} W_1(\mathbf{a}) &= \int_V \mathcal{L}_1(P_1) (\langle \mathcal{L}_2(P) \rangle \{a\}) dV - \int_V P_1 f_V dV - \int_{S_f} P_1 f_S dS = 0 \\ \vdots &\quad \vdots \quad \vdots \quad \vdots \\ W_n(\mathbf{a}) &= \int_V \mathcal{L}_1(P_n) (\langle \mathcal{L}_2(P) \rangle \{a\}) dV - \int_V P_n f_V dV - \int_{S_f} P_n f_S dS = 0. \end{aligned} \tag{3.43}$$

The solutions to (3.42) and (3.43) are identical if the functions $\langle P \rangle$ are identical and all the conditions required by (3.42) are met.

However, because the admissibility conditions required by (3.43) are less restrictive, we can use “simpler” functions $\langle P \rangle$ for (3.43) than for (3.42). Of all the methods which we have described, it is the Galerkin method in the form (3.43) that is most widely used.

EXAMPLE 3.21. *Solution of Poisson's equation using the Galerkin method, with integration by parts*

Using the form W obtained in Example 3.5, we find:

$$\mathcal{L}_1 = \left\langle \frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right\rangle; \quad \mathcal{L}_2 = \begin{Bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{Bmatrix}.$$

Expression (3.43) becomes ($f_s = \alpha = 0$)

$$W_1 = \int_V \left\langle \frac{\partial P_1}{\partial x} \frac{\partial P_1}{\partial x} + \frac{\partial P_1}{\partial y} \frac{\partial P_1}{\partial y}, \frac{\partial P_1}{\partial x} \frac{\partial P_2}{\partial x} + \frac{\partial P_1}{\partial y} \frac{\partial P_2}{\partial y} \right\rangle dV \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} - \int_V P_1 f dV = 0.$$

$$W_2 = \int_V \left\langle \frac{\partial P_2}{\partial x} \frac{\partial P_1}{\partial x} + \frac{\partial P_2}{\partial y} \frac{\partial P_1}{\partial y}, \frac{\partial P_2}{\partial x} \frac{\partial P_2}{\partial x} + \frac{\partial P_2}{\partial y} \frac{\partial P_2}{\partial y} \right\rangle dV \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} - \int_V P_2 f dV = 0.$$

This system of equations and its solution are identical to those in Example 3.20. If we choose a single function over one-fourth of the domain illustrated in Example 3.17, then we have:

$$u(x, y) = P_1(x, y)a_1, \quad P_1(x, y) = (1-x)(1-y), \quad 0 \leq x \leq 1, \quad 0 \leq y \leq 1,$$

where:

$$a_1 = 0.375 \quad \text{and} \quad u_c = 0.375.$$

3.5.3.4 Least squares method

The least squares method consists of minimizing the expression:

$$\pi_m = \int_V R \cdot R dV \tag{3.44}$$

with respect to parameters a_1, a_2, \dots, a_n , where R is the residual function:

$$R = \mathcal{L}(u) + f_V = \langle \mathcal{L}(P) \rangle \{a\} + f_V. \tag{3.45}$$

The stationarity conditions of (3.44) are:

$$W = \delta \pi_m (a_1, a_2, \dots, a_n) = 0$$

$$\begin{aligned} W_1(\mathbf{a}) &= \int_V \mathcal{L}(P_1) (\langle \mathcal{L}(P) \rangle \{a\} + f_V) dV = 0 \\ &\vdots \\ W_n(\mathbf{a}) &= \int_V \mathcal{L}(P_n) (\langle \mathcal{L}(P) \rangle \{a\} + f_V) dV = 0. \end{aligned} \quad (3.46)$$

This method is not widely used because it does not allow for integration by parts, and therefore imposes stricter conditions on the approximation of \mathbf{u} than the Galerkin method. Conversely, it gives us a system which is symmetrical and positive definite no matter what the operator \mathcal{L} may be.

EXAMPLE 3.22. Solution of the Poisson equation with the least squares method

Using the functions P_1 and P_2 from Example 3.17, relation (3.46) is expressed as follows, for the problem defined in Example 3.18:

$$\begin{aligned} W_1 &= \int_V \langle \mathcal{L}(P_1) \cdot \mathcal{L}(P_1) - \mathcal{L}(P_1) \cdot \mathcal{L}(P_2) \rangle dV \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} + \int_V \mathcal{L}(P_1) f dV = 0 \\ W_2 &= \int_V \langle \mathcal{L}(P_2) \cdot \mathcal{L}(P_1) - \mathcal{L}(P_2) \cdot \mathcal{L}(P_2) \rangle dV \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} + \int_V \mathcal{L}(P_2) f dV = 0 \\ 31.2889 a_1 + 25.1937 a_2 &= 10.6667 f \\ 25.1937 a_1 + 87.4463 a_2 &= 12.8000 f \\ a_1 &= 0.2904 f \quad a_2 = 0.0627 f, \\ u_c &= 0.2904 f. \end{aligned}$$

With the one-parameter approximation $u = P_1(x, y) a_1$, we would find:

$$u_c = 0.3409 f.$$

Using the three-parameter approximation and the functions P_1, P_2 and $P_1 \cdot x^2 \cdot y^2$, we would have:

$$\begin{aligned} a_1 &= 0.2949 f \quad a_2 = 0.0385 f \quad a_3 = 0.1562 f \\ u_c &= 0.2949 f. \end{aligned}$$

3.5.4 DISCRETIZATION OF A FUNCTIONAL (Ritz method)

In the Ritz method, we discretize a functional π using an approximation of \mathbf{u} such as (3.33) and then write its stationarity conditions with respect to the approximation parameters:

$$\pi(\mathbf{u}) = \pi(\mathbf{u}(a_1, a_2, \dots, a_n))$$

$$\delta\pi(a_1, a_2, \dots, a_n) = \frac{\partial\pi}{\partial a_1} \delta a_1 + \frac{\partial\pi}{\partial a_2} \delta a_2 + \dots + \frac{\partial\pi}{\partial a_n} \delta a_n = 0, \quad \forall \delta a_i.$$

From this, we get the n equations:

$$\frac{\partial\pi}{\partial a_1} = 0; \quad \frac{\partial\pi}{\partial a_2} = 0; \quad \dots; \quad \frac{\partial\pi}{\partial a_n} = 0. \quad (3.47)$$

The functions \mathbf{u} must satisfy the conditions required by the functional. If the functional π exists, its first variation is identical to a Galerkin integral form W (3.43):

$$\delta\pi = W = 0. \quad (3.48)$$

Thus, the solution obtained using the Ritz method is identical to that obtained using the Galerkin method. Figure 3.8 shows that, in fact, the Ritz and Galerkin methods are identical.

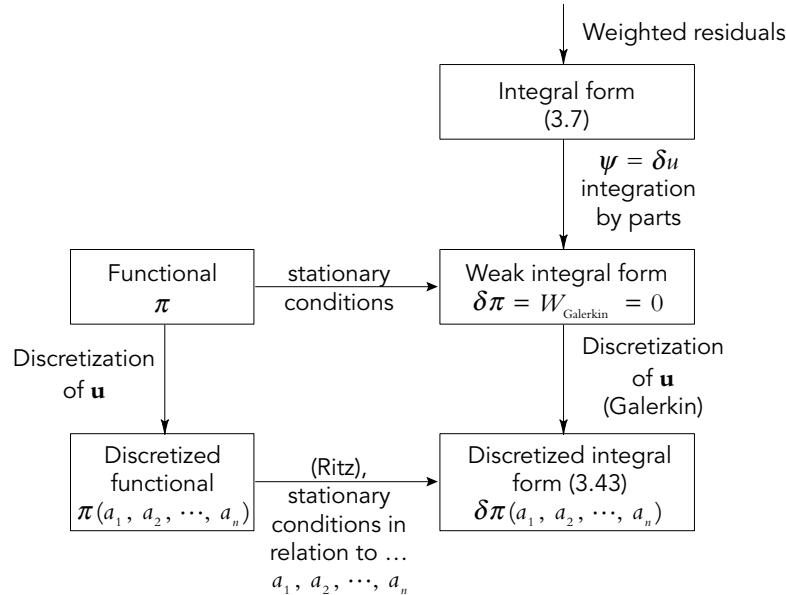


Figure 3.8. Ritz and Galerkin methods

EXAMPLE 3.23. Solution of Poisson's equation using the Ritz method

The expression of π is given in Example 3.11 ($\alpha = f_s = 0$). Using the approximation of u given in Example 3.17, we get:

$$\begin{aligned}\pi(a_1, a_2) &= \frac{1}{2} \langle a_1 \quad a_2 \rangle \int_V \begin{bmatrix} \frac{\partial P_1}{\partial x} & \frac{\partial P_1}{\partial y} \\ \frac{\partial P_2}{\partial x} & \frac{\partial P_2}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial P_1}{\partial x} & \frac{\partial P_2}{\partial x} \\ \frac{\partial P_1}{\partial y} & \frac{\partial P_2}{\partial y} \end{bmatrix} dV \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} \\ &\quad - \langle a_1 \quad a_2 \rangle \int_V \begin{Bmatrix} P_1 \\ P_2 \end{Bmatrix} f dV.\end{aligned}$$

Then $\delta\pi = 0$ gives:

$$\begin{aligned}& \int_V \begin{bmatrix} \frac{\partial P_1}{\partial x} \frac{\partial P_1}{\partial x} + \frac{\partial P_1}{\partial y} \frac{\partial P_1}{\partial y} & \frac{\partial P_1}{\partial x} \frac{\partial P_2}{\partial x} + \frac{\partial P_1}{\partial y} \frac{\partial P_2}{\partial y} \\ Sym. & \frac{\partial P_2}{\partial x} \frac{\partial P_2}{\partial x} + \frac{\partial P_2}{\partial y} \frac{\partial P_2}{\partial y} \end{bmatrix} dV \begin{Bmatrix} a_1 \\ a_2 \end{Bmatrix} \\ & - \int \begin{Bmatrix} P_1 & f \\ P_2 & f \end{Bmatrix} dV = 0\end{aligned}$$

This system of equations is identical to that from Example 3.21, which corresponds to the Galerkin method.

EXAMPLE 3.24. Comparison of results from the various methods of solution

The analytical solution of Poisson's equation is obtained by a development into a series:

$$u(x, y) = \sum_{m=1,3,\dots} \sum_{n=1,3,\dots} u_{mn} \sin\left(\frac{m\pi x}{a}\right) \sin\left(\frac{n\pi y}{b}\right).$$

We also have:

$$f_o = \sum_{m=1,3,\dots} \sum_{n=1,3,\dots} f_{mn} \sin\left(\frac{m\pi x}{a}\right) \sin\left(\frac{n\pi y}{b}\right),$$

$$\text{where } f_{mn} = \frac{16f_o}{\pi^2 mn}, \quad 0 \leq x \leq a, \quad 0 \leq y \leq b.$$

The terms u_{mn} are calculated using Poisson's equation for each pair (m, n) :

$$u_{mn} = \frac{16 f_o}{\pi^2 mn \left(\frac{\pi^2}{a^2} m^2 + \frac{\pi^2}{b^2} n^2 \right)}.$$

The value of the solution at the center ($x = y = 1$) is:

$$u(x=1, y=1) = \frac{64}{\pi^4} \sum_{m=1,3,\dots} \sum_{n=1,3,\dots} \frac{(-1)^{(m+3)/2} \cdot (-1)^{(n+3)/2}}{mn(m^2 + n^2)}.$$

Let us sum up the values of $\frac{u_c}{f}$ obtained by various methods employed above to solve the Poisson equation from Example 3.17.

$$P_1 = (x^2 - 1)(y^2 - 1); \quad P_2 = P_1(x^2 + y^2); \quad P_3 = P_1 \cdot x^2 \cdot y^2.$$

Functions	Point collocation Example 3.18	Subdomain collocation Example 3.19	Galerkin or Ritz Examples 3.20 and 3.23	Least squares Example 3.22	Exact (Fourier series) (14 terms)
P_1	0.2500	0.3750	0.3125	0.3409	
P_1, P_2	0.2976	0.2994	0.2922	0.2904	
P_1, P_2, P_3			0.2949	0.2949	0.2947

3.5.5 PROPERTIES OF THE SYSTEMS OF EQUATIONS

All the methods of solution lead to a system of algebraic equations like:

$$[K]\{a\} = \{F\}. \quad (3.49)$$

Figure 3.9 summarizes the properties of such a system, for each method of solution.

Methods	Terms K_{ij}	F_i	Conditions on $u = \langle P \rangle \{a\}$, and therefore on $\langle P \rangle$	Properties of $[K]$
Point collocation	$\mathcal{L}(P_j)$ in $\mathbf{x} = \mathbf{x}_i$	$-f_V(\mathbf{x}_i)$	on S_u on S_f	non-symmetrical
Subdomain collocation	$\int_{V^i} \mathcal{L}(P_j) \text{ in } \mathbf{x} = \mathbf{x}_i$	$-\int_V f_V(\mathbf{x}_i)$	on S_u on S_f	non-symmetrical
Galerkin	$\int_V P_i \mathcal{L}(P_j) dV$	$-\int_V P_i f_V dV$	on S_u on S_f	symmetrical if \mathcal{L} is self-adjoint
Galerkin after integration by parts	$\int_V \langle \mathcal{L}_1(P_i) \rangle \{ \mathcal{L}_2(P_j) \} dV$	$\int_V P_i f_V dV$ $+ \int_{S_f} P_i f_S dS$	on S_u	symmetrical if \mathcal{L} is self-adjoint $\mathcal{L}_1 = \mathcal{L}_2$
Least squares	$\int_V \mathcal{L}(P_i) \mathcal{L}(P_j) dV$	$-\int_V \mathcal{L}(P_i) f_V dV$	on S_u on S_f	symmetrical and positive definite
Ritz (if the functional exists)	$\int_V \langle \mathcal{L}_1(P_i) \rangle \{ \mathcal{L}_1(P_j) \} dV$	$\int_V P_i f_V dV$ $+ \int_{S_f} P_i f_S dS$	on S_u	symmetrical

Figure 3.9. Properties of the system of equations from the undetermined coefficient method

3.6 List of PDEs and weak expressions

In this section, we present a non-exhaustive list of commonly encountered partial differential equations and their associated Galerkin (variational) weak expressions found in mechanics (fluid mechanics, structural mechanics, heat mechanics, etc.).

We will begin with equations for a scalar field (temperature, concentration, etc.), followed by vector field equations (solid mechanics, fluid mechanics, etc.), starting with a monodimensional model before extending it to other dimensions.

3.6.1 SCALAR FIELD PROBLEMS

3.6.1.1 Monodimensional approach

The general relation of transport-diffusion of a scalar quantity in a monodimensional domain with a non-uniform section is written as:

$$\frac{\partial}{\partial t}(\rho A u) + \frac{\partial}{\partial x}(\rho A a u) + \frac{\partial}{\partial x}(q_d \cdot A) + b u - f_A = 0, \quad 0 \leq x \leq L, \forall t > 0 \quad (3.50)$$

where:

- the constitutive law : $q_d = -k \frac{\partial u}{\partial x}$,
- the boundary conditions: $u(x=0,t) = u_s$ and $q_d(x=L,t) = -f_s$,
- and the initial condition: $u(x,t=0) = u_o(x)$.

The Galerkin weak expression is given by:

$$W = \int_0^L \left[\delta u(x) \left(\frac{\partial}{\partial t}(\rho A u) + \frac{\partial}{\partial x}(\rho A a u) + b u - f_A \right) + \frac{\partial \delta u}{\partial x} k \frac{\partial u}{\partial x} A \right] dx \quad (3.51)$$

$$- \delta u(L) f_s = 0, \quad \forall x \in [0, L], \quad \forall \delta u(x) / \delta u(0) = 0.$$

This scalar relation can be applied to various physical problems:

Heat transport

where: $u(x,t)$: temperature	$(^\circ C)$
ρ	: specific heat capacity \times area	$(J / ^\circ C \cdot m^3)$
a	: velocity of flow, zero for a solid	(m/s)
k	: conductivity coefficient \times area (Fick)	$(W / ^\circ C \cdot m)$
b	: convective exchange coefficient \times area	$(W / ^\circ C \cdot m)$
f_A	: production (source or sink) \times area	(W/m)
$A(x)$: cross-section (area)	(m^2)

The most common boundary conditions are of the type:

- Dirichlet (imposed temperature): $T(x=\bar{x}) = \bar{T}$.
- Neumann (imposed flux): $q_d = -k \frac{\partial u}{\partial x}(x=\bar{x}) = -f_s$.

- Cauchy (convective flux): $q_d = -k \frac{\partial u}{\partial x}(x = \bar{x}) = h(T(\bar{x}) - T_\infty)$,
- h : convective exchange coefficient
- T_∞ : temperature of the surrounding medium.

Transport–diffusion of a pollutant-type concentration

where: $u(x, t)$: concentration	$(\text{g}/\text{m}^3 \text{ of fluid})$
ρ	: 1	$-$
a	: velocity of flow	(m/s)
k	: diffusion coefficient \times area (Fick)	(m^2/s)
b	: term of deposit \times area	$(\text{m}^2/\text{s}^{-1})$
f	: production \times area (source or sink)	$(\text{g}/\text{m} - \text{s})$

3.6.1.2 Two-dimensional approach

The extension to the two-dimensional case (constant thickness) is given by:

$$\frac{\partial}{\partial t}(\rho u) + \mathbf{Div}(\rho \mathbf{a} u + \mathbf{q}_d) + bu - f = 0, \quad 0 \leq x \leq L, \quad (3.52)$$

where:

- the constitutive law : $\mathbf{q}_d = -[\mathbf{H}] \nabla u$,
- the boundary conditions: $u = u_s$ on S_u and $\mathbf{q}_d \cdot \mathbf{n} = -f_s$ on S_f ,
- and the initial condition: $u(x, y, t = 0) = u_o(x, y)$.

The Galerkin weak expression is given by:

$$W = \int_A \left[\delta u \left(\frac{\partial}{\partial t}(\rho u) + \mathbf{Div}(\rho \mathbf{a} u) + bu - f \right) + \langle \nabla \delta u \rangle [\mathbf{H}] \{ \nabla u \} \right] dA \quad (3.53)$$

$$- \int_{S_f} \delta u f_s dS = 0,$$

$$\forall \delta u(x, y) \text{ where } u = u_s \text{ and } \delta u(x, y) = 0 \text{ on } S_u.$$

where:

ρ	: specific heat capacity	$(J / {}^\circ C \cdot m^3)$
$[H]$: Fick matrix or conductivity matrix	$(W / {}^\circ C \cdot m)$
$\mathbf{a}(x, y, t)$: velocity vector of components (a_x, a_y)	(m / s)
b	: exterior exchange	$(W / {}^\circ C \cdot m^3)$
f	: production \times thickness	(W / m^3)
\mathbf{n}	: normal external to the domain	
∇	: gradient operator $\left(\frac{\partial}{\partial x} \frac{\partial}{\partial y} \right)$	

The extension to three dimensions gives relations similar to (3.52) and (3.53).

In the particular case where the stationary hypothesis ($\partial/\partial t = 0$) can be used, relation (3.53) above, extended to three-dimensional space and where $a = 0$, gives us the biharmonic relation:

$$\mathbf{Div} \mathbf{q}_d - f_v = 0, \forall (x, y, z) \in V, \quad (3.54)$$

where:

$$\mathbf{q}_d = \begin{Bmatrix} q_x \\ q_y \\ q_z \end{Bmatrix} = -[H] \{ \nabla u \} = \begin{Bmatrix} -k_x \cdot \partial u / \partial x \\ -k_y \cdot \partial u / \partial y \\ -k_z \cdot \partial u / \partial z \end{Bmatrix},$$

$$u = u_s \text{ on } S_u \text{ and } \mathbf{q}_d \cdot \mathbf{n} = \alpha \cdot u - f_s \text{ on } S_f$$

and:

$$W = \int_V \langle \nabla \delta u \rangle [H] \{ \nabla u \} dV - \int_V \delta u f_v dS - \int_{S_f} \delta u (f_s - \alpha u) dS = 0,$$

$$\forall \delta u(x, y, z) / u = u_s \text{ and } \delta u(x, y, z) = 0 \text{ on } S_u.$$

This general relation can be applied in such diverse domains of physics as:

- Thermal $u \equiv T(x, y, z)$: temperature
- Flows in porous media $u \equiv P_H(x, y, z)$: load pressure
- Electrostatics $u \equiv \varphi(x, y, z)$: potential function
- Diffusion of a pollutant $u \equiv C(x, y, z)$: concentration
- Incompressible perfect fluid $u \equiv \psi(x, y, z)$: stream function

3.6.2 SOLID MECHANICS

3.6.2.1 Monodimensional approach (constant section)

a) Truss element in traction

The general relation of the axial dynamics of a truss element with a constant section (only traction and compression) is written as:

$$\rho A \frac{\partial^2 u}{\partial t^2} - \frac{\partial \sigma_A}{\partial x} - f_A = 0, \quad 0 \leq x \leq L, \quad \forall t > 0, \quad (3.55)$$

with:

- the constitutive law : $\sigma_A = EA \frac{\partial u}{\partial x}$,
- the boundary conditions : $u(x=0, t) = u_s$ and $\sigma(x=L, t) = f_s$,
- and the initial conditions: $u(x, t=0) = u_o$ and $\frac{\partial u}{\partial t}(x, t=0) = v_o$.

where:	$u(x, t)$: axial shift	(m)
	σ_A	: axial load \times area = normal force	(N)
	$\frac{\partial u}{\partial x}$: axial deformation	–
	E	: Young's elasticity modulus	(N / m^2)
	ρ	: density	(kg / m^3)
	f_A	: lineic stress	(N / m)
	A	: section	(m^2)

The Galerkin weak formulation is given by:

$$W = \int_0^L \left[\delta u \left(\rho A \frac{\partial^2 u}{\partial t^2} - f_A \right) + \frac{\partial \delta u}{\partial x} AE \frac{\partial u}{\partial x} \right] dx - (\delta u \cdot f_s)_{x=L} = 0, \quad (3.56)$$

$\forall \delta u(x)$ with $u(x=0) = \bar{u}$, $\delta u(x=0) = 0$.

b) Beam with bending

In the particular case of a slender beam (Bernoulli's hypothesis), the equilibrium equation is written as:

$$\rho A \frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 M}{\partial x^2} - f = 0, \quad 0 \leq x \leq L, \quad \forall t > 0, \quad (3.57)$$

with:

- the constitutive law : $M = -EI \frac{\partial^2 u}{\partial x^2}$, I : moment of inertia
- the boundary conditions : $u(x=0, t) = u_s$ and $\frac{\partial u}{\partial x}(x=0, t) = 0$,
 $M(x=L, t) = M_s$ and $\frac{\partial M}{\partial x}(x=L, t) = T_s$,
- and the initial conditions : $u(x, t=0) = u_o$ and $\frac{\partial u}{\partial t}(x, t=0) = v_o$.

where:	$u(x, t) = w(x, t)$: transversal shift	(m)
	M	: moment	(N-m)
	T	: shearing force	(N)
	ρ	: density	(kg / m ³)
	f	: lineic stress	(N / m)

The weak formulation is given by:

$$W = \int_0^L \left[\delta u \left(\rho A \frac{\partial^2 u}{\partial t^2} - f \right) + \frac{\partial^2 \delta u}{\partial x^2} EI \frac{\partial^2 u}{\partial x^2} \right] dx + (\delta u \cdot M_s + \delta u \cdot T_s)_{x=L} = 0, \quad (3.58)$$

where, in $x=0$: $u=0$, $\delta u=0$

$$\frac{\partial u}{\partial x}=0, \quad \frac{\partial \delta u}{\partial x}=0.$$

3.6.2.2 Two-dimensional approach (constant thickness)

The two-dimensional approach gives us the same types of equations with vectorial quantities (constant thickness):

$$\rho \frac{\partial^2}{\partial t^2} \mathbf{u}(\mathbf{x}, t) - \operatorname{Div}[\boldsymbol{\sigma}] - \mathbf{f}_v = 0, \quad \forall \mathbf{x} \in A, \quad \forall t > 0. \quad (3.59)$$

with:

- the boundary conditions: $\mathbf{u}(\mathbf{x}, t) = \mathbf{u}_s(t)$ on S_u ,

$$[\boldsymbol{\sigma}] \cdot \mathbf{n} = \mathbf{f}_s \text{ on } S_f,$$

- the initial conditions: $\mathbf{u}(\mathbf{x}, t=0) = \mathbf{u}_o(\mathbf{x})$,

$$\frac{\partial}{\partial t} \mathbf{u}(\mathbf{x}, t=0) = \dot{\mathbf{u}}_o(\mathbf{x}).$$

the constitutive law:

$$\{\boldsymbol{\sigma}\} = [H]\{\boldsymbol{\varepsilon}\}, \text{ where } [\boldsymbol{\varepsilon}] = \frac{1}{2} (\nabla \mathbf{u} + \nabla \mathbf{u}^T),$$

In two dimensions (2D):

$$\begin{aligned} \mathbf{u} &= \langle u(x, y, t) \ v(x, y, t) \rangle && : \text{(displacement vector)} && (m) \\ \langle \boldsymbol{\sigma} \rangle &= \langle \sigma_x \ \sigma_y \ \sigma_{xy} \rangle && : \text{load tensor} && (N/m^2) \\ \langle \boldsymbol{\varepsilon} \rangle &= \left\langle \frac{\partial u}{\partial x} \ \frac{\partial v}{\partial y} \ \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right) \right\rangle && : \text{strain tensor} && (-) \\ [H] & && : \text{matrix of the material properties} && (N/m^2) \end{aligned}$$

The matrix of the material properties differs depending on whether the problem is formulated for plane stress or plane strain conditions:

$$\begin{aligned} [H] &= \frac{E}{1-v^2} \begin{bmatrix} 1 & v & 0 \\ v & 1 & 0 \\ 0 & 0 & \frac{1-v}{2} \end{bmatrix} : \text{plane stress,} && (3.60) \\ [H] &= \frac{E}{(1+v)(1-2v)} \begin{bmatrix} 1-v & v & 0 \\ v & 1-v & 0 \\ 0 & 0 & \frac{1-2v}{2} \end{bmatrix} : \text{plane strains.} \end{aligned}$$

The boundary conditions are generally of the following types:

— Dirichlet (or kinematic): $\mathbf{u}(x, y, t) = \mathbf{u}_s(t)$ on S_u

— Neumann (or mechanical): $[\sigma] \cdot \mathbf{n} = \mathbf{f}_s$ on S_f .

The weak formulation requires a vector weighting function and is written thus:

$$\begin{aligned} W = & \int_A \left[\delta \mathbf{u}(\mathbf{x}) \rho \frac{\partial^2 \mathbf{u}}{\partial t^2} + \langle \delta \boldsymbol{\varepsilon} \rangle [H] \{ \boldsymbol{\varepsilon} \} \right] dA - \int_A \delta \mathbf{u}(\mathbf{x}) \mathbf{f}_v dA \\ & + \int_{S_f} \delta \mathbf{u}(\mathbf{x}) \mathbf{f}_s dS = 0, \\ \text{with } \delta \mathbf{u}(\mathbf{x}) &= 0 \text{ on } S_u. \end{aligned} \quad (3.61)$$

3.6.2.3 Three-dimensional approach

The extension of the weak formulation into three-dimensional space is written as:

$$\begin{aligned} W = & \int_V \left[\delta \mathbf{u}(\mathbf{x}) \rho \frac{\partial^2 \mathbf{u}}{\partial t^2} + \langle \delta \boldsymbol{\varepsilon} \rangle [H] \{ \boldsymbol{\varepsilon} \} \right] dV - \int_V \delta \mathbf{u}(\mathbf{x}) \mathbf{f}_v dV \\ & + \int_{S_f} \delta \mathbf{u}(\mathbf{x}) \mathbf{f}_s dA = 0, \\ \text{with } \delta \mathbf{u}(\mathbf{x}) &= 0 \text{ on } S_u. \end{aligned} \quad (3.62)$$

where: $\mathbf{u}(x, y, z, t) = \langle u(x, y, z, t) \ v(x, y, z, t) \ w(x, y, z, t) \rangle$

$$\langle \boldsymbol{\sigma} \rangle = \langle \sigma_x \ \sigma_y \ \sigma_z \ \sigma_{xy} \ \sigma_{yz} \ \sigma_{xz} \rangle$$

$$\langle \boldsymbol{\varepsilon} \rangle = \left\langle \frac{\partial u}{\partial x} \ \frac{\partial v}{\partial y} \ \frac{\partial w}{\partial z} \ \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \right) \left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \left(\frac{\partial w}{\partial x} + \frac{\partial u}{\partial z} \right) \right\rangle$$

and:

$$[H] = \frac{E}{(1+v)(1-2v)} \begin{bmatrix} 1-v & v & v & 0 & 0 & 0 \\ v & 1-v & v & 0 & 0 & 0 \\ v & 0 & 1-v & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1+2v}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2v}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2v}{2} \end{bmatrix}$$

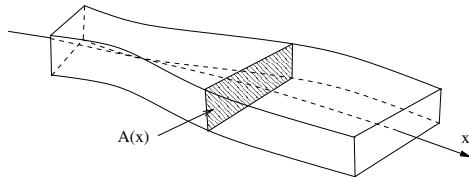
Symmetric

3.6.3 FLUID MECHANICS

This section covers the main relations found in fluid mechanics, respectively, presenting the monodimensional and then two-dimensional relations. These relations express the laws of conservation of mass and momentum. In the case of compressible flows, there is an additional law of energy conservation, supplemented by a perfect gas-type law of state [CAN 90; CHA 97].

3.6.3.1 One-dimensional flow with free surface

The flow of an incompressible fluid in a monodimensional channel with a variable section as illustrated below:



is governed by the relations:

$$\begin{aligned} b \frac{\partial h}{\partial t} + \frac{\partial(b H u)}{\partial x} &= 0 \\ \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + \frac{g}{bH} \frac{\partial I}{\partial x} + g(S_f - S_o) - v \frac{\partial^2 u}{\partial x^2} &= 0 \end{aligned} \quad (3.63)$$

where: $I = \int_0^H (H - z) \frac{\partial b H}{\partial z} dz$; $A(x) = b H$

b	: width of the channel	(m)
h	: relative height of the water	(m)
H	: height of the water in relation to the bottom	(m)
u	: velocity	(m / s)
g	: gravity	(m / s ²)
S_f	: friction slope	(–)
S_o	: slope of the bottom	(–)
v	: viscosity	(m ² / s)

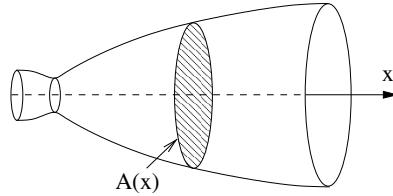
The boundary conditions relate to the flow rate and the height of the water on entering and exiting the domain.

The associated weak formulation is written as:

$$\begin{aligned} W_h &= \int_0^L \left(\delta h b \frac{\partial h}{\partial t} - \frac{\partial \delta h}{\partial x} b H u \right) dx + [\delta h b H u]_0^L = 0, \\ W_{SL} &= \int_0^L \delta u \left(\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + \frac{g}{bH} \frac{\partial I}{\partial x} + g(S_f - S_o) \right) dx \\ &\quad - \int_0^L \frac{\partial \delta u}{\partial x} v \frac{\partial u}{\partial x} dx + [\delta u v \frac{\partial u}{\partial x}]_0^L = 0. \end{aligned} \quad (3.64)$$

3.6.3.2 Monodimensional compressible flow

A monodimensional compressible flow in a pipe with variable section, as illustrated below in the case of the duct in a rocket engine:



is governed by:

$$\begin{aligned} \frac{\partial(\rho A)}{\partial t} + \frac{\partial(\rho A u)}{\partial x} &= 0 \\ \frac{\partial(\rho A u)}{\partial t} + \frac{\partial(\rho A u^2 + A p)}{\partial x} &= p \frac{\partial A}{\partial x} \\ \frac{\partial(\rho A e)}{\partial t} + \frac{\partial(\rho A e + A p)u}{\partial x} &= 0 \\ p = \rho r T; e = C_v \cdot T + \frac{1}{2} u^2; \end{aligned} \quad (3.65)$$

where:

$u(x, t)$: velocity	(m / s)
$p(x, t)$: pressure	(N / m^2)
$T(x, t)$: temperature	(K)
e	: total mass energy	(J / kg)
ρ	: density	(kg / m^3)
r	: perfect gas constant	$(m^2 / s^2 - K)$
C_v	: calorific capacity	$(J / kg - K)$
$A(x)$: section	(m^2)

The boundary conditions are governed by characteristics theory [DON 03, p. 164]. The number of conditions to be imposed is equal to the number of characteristics entering into the domain (see section 3.6.3.4 for further details). By rewriting the equilibrium equations in the following summarized form:

$$\frac{\partial}{\partial t} \mathbf{U} + \frac{\partial}{\partial x} \mathbf{F} = \mathbf{0}, \text{ where } \mathbf{U} = \begin{Bmatrix} \rho A \\ \rho A u \\ \rho A e \end{Bmatrix} \text{ and } \mathbf{F} = \begin{Bmatrix} \rho A u \\ \rho A u^2 + p \\ (\rho A e + p) u \end{Bmatrix}.$$

The associated weak formulation is written thus:

$$W = \int_0^L \delta \mathbf{U} \cdot \frac{\partial}{\partial t} \mathbf{U} dx - \int_A \delta \mathbf{U}_{,x} \cdot \mathbf{F} dx + [\delta \mathbf{U} \cdot \mathbf{F}]_0^L = 0. \quad (3.66)$$

3.6.3.3 Two-dimensional (2D) Saint-Venant equations

The equations in fluid mechanics, applied to a free-surface two-dimensional flow, are written as:

$$\begin{aligned} \frac{\partial h}{\partial t} + \nabla \cdot (H \mathbf{u}) &= 0 \\ \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} + g \nabla h - \frac{1}{\rho} \nabla \cdot [\tau] + \frac{g}{C^2} \frac{|\mathbf{u}|}{H} \mathbf{u} &= \mathbf{0} \\ [\tau] &= 2\mu \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{1}{2} \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \\ sym & \frac{\partial v}{\partial y} \end{bmatrix} \text{ where } \mathbf{u} = \begin{Bmatrix} \mathbf{u} \\ \mathbf{v} \end{Bmatrix}. \end{aligned} \quad (3.67)$$

where:

$\mathbf{u} = (u, v)$: velocity vector	(m / s)
$h(x, y, t)$: relative height of the water	(m)
$H(x, y, t)$: height of the water in relation to the bottom	(m)
g	: gravity	(m / s^2)
ρ	: density	(kg / m^3)
$[\tau]$: viscous stress tensor	(N / m^2)
C	: Chézy's coefficient	$(m^{1/2} / s)$
μ	: dynamic viscosity	(N / m^2)

The boundary conditions relate to the flow rate and the height of the water upon entering and exiting the domain. A no-slip condition, $\vec{u} = \vec{0}$, is imposed on the bottoms. The associated weak formulation is written as:

$$\begin{aligned} W_h &= \int_A \delta h \frac{\partial h}{\partial t} dA - \int_A \nabla \delta h \cdot (H \mathbf{u}) dA + \int_S \delta h H \mathbf{u} \cdot \mathbf{n} = 0, \\ W_{SV} &= \int_A \delta \mathbf{u} \cdot \left(\frac{\partial \mathbf{u}}{\partial t} + g \nabla h + \frac{g}{C^2} \frac{|\mathbf{u}|}{H} \mathbf{u} \right) + \int_A \frac{1}{\rho} [\varepsilon_{NS}] : [\tau] dA = 0, \quad (3.68) \\ \text{where } [\varepsilon_{NS}] &= \frac{1}{2} (\nabla \delta \mathbf{u} + (\nabla \delta \mathbf{u})^T). \end{aligned}$$

3.6.3.4 Incompressible Navier-Stokes equations (2D)

Internal incompressible flows are governed by the relations:

$$\begin{aligned} \nabla \cdot \mathbf{u} &= \frac{p}{\lambda} \\ \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \times \mathbf{u}) + \frac{1}{\rho} \nabla p - \frac{1}{\rho} \nabla \cdot [\tau] - \mathbf{g} &= \mathbf{0} \quad (3.69) \end{aligned}$$

where:

$\mathbf{u} = (u, v)$: velocity vector	(m/s)
$p(x, y, t)$: pressure	(m)
λ	: penalty coefficient	$(-)$
\mathbf{g}	: gravity vector	(m/s^2)
ρ	: density	(kg/m^3)
$[\tau]$: viscous stress tensor	(N/m^2)

The boundary conditions relate to the adherence to the walls $\vec{u} = \vec{0}$. The conditions upon entering and exiting the domain stem from the relation between flow rate and pressure, or indeed between pressure upstream and pressure downstream. The associated weak formulation is written as:

$$\begin{aligned} W_m &= \int_A \delta p \left(\nabla \cdot \mathbf{u} - \frac{p}{\lambda} \right) dA = 0, \\ W_{NS} &= \int_A \delta \mathbf{u} \cdot \left(\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \times \mathbf{u}) - \mathbf{g} \right) dA - \int_A \frac{1}{\rho} (\nabla \cdot \delta \mathbf{u} p - [\varepsilon_{NS}] : [\tau]) dA \\ &\quad + \oint_S \frac{1}{\rho} \delta u \cdot (p \mathbf{n} - [\tau] \cdot \mathbf{n}) dS = 0, \end{aligned} \quad (3.70)$$

where $[\varepsilon_{NS}] = \frac{1}{2} \left(\nabla \delta \mathbf{u} + (\nabla \delta \mathbf{u})^T \right)$.

3.6.3.5 Compressible two-dimensional Euler equations

If the flow is compressible, a further relation is introduced, which is associated with the coupling with the thermal behavior. Supplemented by a perfect gas law of state, these relations (for a non-viscous fluid) are generally grouped together in the following vector form:

$$\begin{aligned} \frac{\partial}{\partial t} \begin{Bmatrix} \rho \\ \rho u \\ \rho v \\ \rho e \end{Bmatrix} + \frac{\partial}{\partial x} \begin{Bmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ (\rho e + p)u \end{Bmatrix} + \frac{\partial}{\partial y} \begin{Bmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ (\rho e + p)v \end{Bmatrix} &= \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{Bmatrix}, \\ p &= \rho r T; e = C_v T + \frac{1}{2}(u^2 + v^2); \\ \text{so that } \frac{\partial}{\partial t} \mathbf{U} + \frac{\partial}{\partial x} \mathbf{F} + \frac{\partial}{\partial y} \mathbf{G} &= \mathbf{0}. \end{aligned} \quad (3.71)$$

where:

$\mathbf{u} = (u, v)$: components of the velocity vector	(m/s)
$p(x, y, t)$: pressure	(N/m^2)
$T(x, y, t)$: temperature	(K)
e	: total mass energy	(J/kg)
ρ	: density	(kg/m^3)
r	: perfect gas constant	$(m^2/s^2 - K)$
C_v	: calorific capacity	$(J/kg - K)$

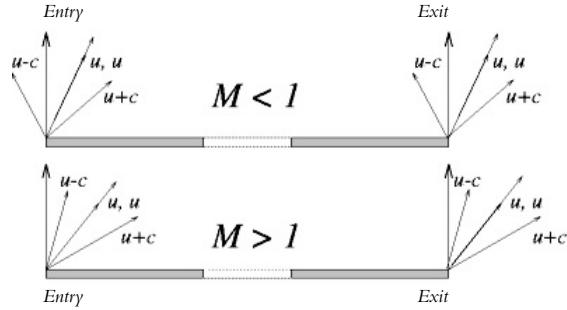
The boundary conditions relate to the slip condition walls $\mathbf{u} \cdot \mathbf{n} = 0$, and are governed by the characteristics theory [DON 03, p. 164] for boundaries such as the entry and exit points. This theory is based on knowledge of the characteristics of propagation of information in *space-time* (x, y, t) . The number of conditions to be imposed is a function of the number of characteristics entering the domain. For a two-dimensional compressible flow, there are four. For a boundary with exterior normal \mathbf{n} and local celerity c , they are associated with the speed of propagation:

$$(\mathbf{u} \cdot \mathbf{n}, \mathbf{u} \cdot \mathbf{n}, \mathbf{u} \cdot \mathbf{n} + c, \mathbf{u} \cdot \mathbf{n} - c).$$

Depending on the nature of the flow (see the following figure), we can distinguish the following cases:

- Subsonic entry, comprising three input characteristics: the pressure and the two velocity components are imposed.
- Supersonic entry, comprising four input characteristics: the pressure, the temperature and the two velocity components are imposed.
- Subsonic exit, comprising a single input characteristic: the pressure is imposed.

- Supersonic exit, with no input characteristics: no boundary condition need be imposed.



Subsonic case (Mach = $M < 1$); Supersonic case (Mach = $M > 1$)

The associated weak formulation is written:

$$W = \int_A \delta \mathbf{U} \cdot \frac{\partial}{\partial t} \mathbf{U} dA - \int_A (\delta \mathbf{U}_{,x} \mathbf{F} + \delta \mathbf{U}_{,y} \mathbf{G}) dA + \oint_S \delta \mathbf{U} \cdot (\mathbf{F} n_x + \mathbf{G} n_y) dS = 0, \quad (3.72)$$

where (n_x, n_y) are the components of the normal to the boundaries.

Remark

The extension to the axisymmetrical case is written thus, while keeping the vectors \mathbf{U} , \mathbf{F} and \mathbf{G} unchanged:

$$\frac{\partial}{\partial t} \mathbf{U} + \frac{\partial}{\partial z} \mathbf{F} + \frac{1}{r} \frac{\partial}{\partial r} (r \mathbf{G}) = \begin{pmatrix} 0 \\ 0 \\ p \\ 0 \end{pmatrix}, \quad r : \text{radial distance from the } z \text{ axis}$$

the relations being projected into a framework (r, z) , with z on the axis of the domain and r the radial distance from that axis.

Important results

Integral form of the weighted residual method:

$$W = \int_V \psi R(\mathbf{u}) dV = \int_V \psi (\mathcal{L}(\mathbf{u}) - f_V) dV = 0. \quad (3.7)$$

First variation of a functional:

$$\delta\pi = \frac{\partial\pi}{\partial\mathbf{u}} \delta\mathbf{u} + \frac{\partial\pi}{\partial\left(\frac{\partial\mathbf{u}}{\partial x}\right)} \delta\left(\frac{\partial\mathbf{u}}{\partial x}\right) + \dots \quad (3.13b)$$

Functional associated with an integral form:

$$\pi \text{ such that } \delta\pi \equiv W = 0 \quad (3.15a)$$

$$W = \int_V \delta u (\mathcal{L}(\mathbf{u}) - f_V) dV = 0. \quad (3.15b)$$

Discretized integral form:

$$W_i = \int_V \psi_i (\mathcal{L}(\mathbf{u}(a_1, a_2, \dots, a_n)) + f_V) dV = 0 \quad (3.31)$$

$$i = 1, 2, \dots, n.$$

Point collocation:

$$W_i(\mathbf{a}) = (\langle \mathcal{L}(P) \rangle \{a\} + f_v)_{\mathbf{x}=\mathbf{x}_i} = 0. \quad (3.36)$$

Subdomain collocation

$$W_i(\mathbf{a}) = \int_{V_i} (\langle \mathcal{L}(P) \rangle \{a\} + f_V) dV = 0. \quad (3.38)$$

Galerkin (after integration by parts) or Ritz:

$$W_i(\mathbf{a}) = \int_V \mathcal{L}_1(P_i) \langle \mathcal{L}_2(P) \rangle \{a\} dV$$

$$- \int_V P_i f_V dV - \int_{S_f} P_i f_S dS = 0. \quad (3.43)$$

Least squares:

$$W_i(\mathbf{a}) = \int_V \mathcal{L}(P_i) (\langle \mathcal{L}(P) \rangle \{a\} + f_V) dV = 0. \quad (3.46)$$

Notations

$\{a\}, a_1, a_2, \dots$	parameters of the approximation of \mathbf{u}
c	damping coefficient
$[C]$	damping matrix for a discrete system
E_u	set of acceptable functions \mathbf{u}
E_ψ	set of weighting functions
$\mathbf{f}_V, \mathbf{f}_S$	load vector for volume and surface
$\{F\}$	load vector of a discrete system
$[H]$	matrix of properties of the material
$[K]$	global or “stiffness” matrix of a discrete system
l, m, n	components of the unit vector normal to the boundary of the domain
$\mathcal{L}(u), \mathcal{C}(u)$	differential operators defining the equations and boundary conditions of a continuous physical system
m	mass per unit volume
$[M]$	mass matrix of a discrete system
\mathbf{q}	physical variables such as flow rate and stresses
R	residual corresponding to a partial differential equation
S_u, S_f	parts of the boundary of the domain on which \mathbf{u} and \mathbf{f} are known
\mathbf{u}	unknown variables of a physical system
W, W_1, W_2, \dots	integral forms
$\delta u, \delta \pi$	first variation of a function, and of a functional
$\delta^2 u, \delta^2 \pi$	second variation of a function, and of a functional
$\delta(\mathbf{x})$	Dirac distribution corresponding to point \mathbf{x}
Δ	Laplacian operator
$\lambda, \lambda_1, \lambda_2, \dots$	eigenvalues or Lagrange multipliers
$\psi, \psi_1, \psi_2, \dots$	weighting functions
$\pi, \pi_1, \pi^*, \pi_r, \pi_c$	functionals

Additional data

UNITS

Length (L): meter (m) Volume: m^3 1 liter = $10^{-3}m^3 = 1,000 \text{ cm}^3$

Time (t): seconds (s) Velocity (L / t): m/s Acceleration (L / t^2): m / s^2

Frequency (1 / s): cycle / s (Hertz) Angle: radian (L / L)

Angular velocity: radian / s

Temperature (T): $K = 273.15 + {}^\circ C$ ${}^\circ F = 32 + 1.8 {}^\circ C$

MECHANICAL UNITS

Mass (M): kilogram (kg)

Density ρ : kg / m^3

Force (ML / t^2): Newton (kg·m / s^2)

Dyne: (gm - cm / s^2)

Force (ML / t^2): kgf (9.806 N)

Volumic force: N / m^3

Stress, pressure: N / m^2 (Pascal: Pa)

Elasticity modulus (E): N / m^2

Energy, work (ML^2 / t^2):

Power: N - m / s (Watt: W)

N - m (Joule: J)

Absolute viscosity μ : N - s / m^2 (Pa - s) Kinematic viscosity $\nu = \frac{\mu}{\rho}$: m^2 / s

THERMAL UNITS

Thermal flux (q): W / m²

Heat conductivity (k): W / m - K

Heat capacity (c_p): W / kg - K

Heat diffusivity $\left(\frac{k}{\rho c_p}\right)$: m² / s

Convective coefficient (h): W/m² – K Heat production: W / m³

VALUES OF SOME QUANTITIES

$$g = 9.81 \text{ m} / \text{s}^2$$

$$\text{Atmospheric pressure: } 1 \text{ atm} = 101.325 \text{ Pa} \quad 1 \text{ truss} = 10^5 \text{ Pa}$$

$$\text{Density of air at } 20^\circ\text{C: } 1.024 \text{ kg} / \text{m}^3 \quad c_p(\text{air}) = 1,004 \text{ J} / \text{kg} - \text{K}$$

$$\text{Conductivity of air: } 25.63 \times 10^{-3} \text{ W} / \text{m} - \text{K}$$

$$\mu_{\text{air}} = 1.81 \times 10^{-5} \text{ Ns} / \text{m}^2 \quad v_{\text{air}} = 1.50 \times 10^{-5} \text{ m}^2 / \text{s}$$

$$\text{Density of water at } 20^\circ\text{C: } 998.20 \text{ kg} / \text{m}^3 \quad c_p(\text{water}) = 4,182 \text{ J} / \text{kg} - \text{K}$$

$$\text{Conductivity of air: } 0.6 \text{ W} / \text{m} - \text{K}$$

$$\mu_{\text{water}} = 100.5 \times 10^{-5} \text{ Ns} / \text{m}^2 \quad v_{\text{water}} = 0.1007 \times 10^{-5} \text{ m}^2 / \text{s}$$

THERMAL AND MECHANICAL PROPERTIES

Material/ substance	ρ (kg / m ³)	C_p (J / kg - C)	k (W / m - C)	$k/\rho C_p$ (10 ⁵ m ² /s)	E (GPa)
Aluminum	2,707	896	204	8.411	72
Iron (5%)	7,850	460	59	1.634	100 – 150
Steel					200
Cr 0 %	7,897	452	73	2.045	
20 %	7,689	460	22	0.622	
Tungsten 10 %	8,314	419	48	1.378	
Copper (Cu)	8,954	383.1	386	11.253	120
Bronze (Cu - Al)					
(95 / 5) %	8,666	410	83	2.338	
Brass (Cu - Zn)					
(70 / 30) %	8,522	385	111	3.383	
Silver	10,524	234	415	16.852	
Cement					
Portland	1,500	≈ 800	0.29		
Granite	≈ 2,640	≈ 800	1.7–4	0.08–0.19	
Pine wood	430	≈ 2,800	0.120	0.0093	15
Water	998.2	4,182	0.604	0.0145	2.19
Air	1.204	1,004	0.025	2.076	

Bibliography

- [CAN 90] CANDEL S., *Mécanique des fluides*, Dunod Université, 1990.
- [CHA 97] CHASSAING P., *Mécanique des fluides – Eléments d'un premier parcours*, Cépaduès Editions, 1997.
- [COL 66] COLLATZ L., *The Numerical Treatment of Differential Equations*, Springer-Verlag, 1966.
- [COR 00] W. CORMACK R., *Numerical Computation of Compressible Viscous Flow*, Stanford University course, 2000.
- [CRA 56] CRANDALL S.H., *Engineering Analysis*, McGraw-Hill, 1956.
- [DON 03] DONEA J., HUERTA A., *Finite Element Method for Flow Problems*, Wiley, 2003.
- [FAR 82] FARLOW S.J., *Partial Differential Equations*, Wiley, 1982.
- [FIN 72] FINLAYSON B.A., *The Method of Weighted Residuals and Variational Principles*, Academic Press, 1972.
- [KRE 88] KREYSZIG E., *Advanced Engineering Mathematics*, Wiley, 1988.
- [MIK 64] MIKHLIN S.C., *Variational Methods in Mathematical Physics*, Macmillan, 1964.
- [MIK 71] MIKHLIN S.C., *The Numerical Performance of Variational Methods*, Wolters-Noordhoff, 1971.
- [WAS 82] WASHIZU K., *Variational Methods in Elasticity and Plasticity*, 3rd ed., Pergamon, 1982.

CHAPTER 4

Matrix presentation of the finite element method

4.0 Introduction

This chapter describes the finite element method and the various stages necessary for its implementation. A particular emphasis is placed on matrix organization, because matrix algebra expressions are very easy to transform into computer codes. To begin with, we define the finite element method as a discretization process for Galerkin integral expressions; it involves replacing the global integral form W with a summation of **elementary integral forms** W^e . Then, every integral form is discretized using finite element approximations. This leads to the definition of the **global** and **elementary matrices**. We go on to discuss the convergence conditions (toward exact solutions) and present the “patch test” technique, which is useful for verifying the convergence of a finite element model.

Then, we describe the **matrix organization** of the discretized elementary integral forms, using variational forms of the problems described in section 3.6. The **assembly technique**, which is typical of the finite element method, allows us to superpose (or merge) the elementary matrices and vectors into global matrices and vectors. Then, we examine the properties of the global matrix and the various **techniques for its storage**: particularly the “skyline” storage method.

Finally, we describe the different ways of introducing **boundary conditions** into the final system of equations and the operations for transforming the variables. The chapter ends with a detailed example of the finite element method applied to the solution of Poisson’s equation.

4.1 The finite element method

4.1.1 FINITE ELEMENT APPROACH

The finite element method consists of using a finite element approximation (see section 1.1.2) of the unknown functions \mathbf{u} to discretize an integral form W (section 3.2), and then solving the resulting system of algebraic equations.

In this section, we will briefly describe the various stages of the solution process, which will then be examined in detail in the rest of the chapter.

A mathematical model, written in the form of a weak formulation (see Chapter 3), is used to represent the behavior of a physical system. This model includes the conservation and constitutive laws and the boundary conditions, written in a weak formulation. The finite element method is usually based on a Galerkin/Ritz weak formulation. It is also possible to use a collocation or least squares expression, for instance, for certain types of problems [BA2 90; HUG 87; ZIE 00].

The solving of a physical problem by the finite element method involves the following stages (see section 0.3):

- a) Construction of the weak expression of the mathematical model (Chapter 3):

$$W = \iint_A \cdots dA + \oint_S \delta u \cdot f_s ds = 0,$$

where $\oint_S \delta u \cdot f_s ds = \underbrace{\int_S \delta u \cdot f_R ds}_{S_u \text{ known displacements}} + \underbrace{\int_S \delta u \cdot f_{ext} ds}_{S_f \text{ known forces}}$

where A is the domain of calculation and S_u and S_f the parts of the contour, where the external displacements and surface loads or forces, respectively, are imposed.

- b) Representation of the domain of calculation V by a set of finite elements of simple geometric forms (Chapters 1 and 2):

$$V = \sum_e V^e; \quad \mathbf{x}^e = \sum_i \bar{\mathbf{N}}_i(\xi, \eta, \dots) \mathbf{x}_i, \quad (4.1)$$

where:

ξ, η, \dots : elementary coordinates in the reference space;

\mathbf{x}^e : vector of the elementary coordinates;

\mathbf{x}_i : elementary nodal coordinates;

$\bar{\mathbf{N}}_i$: geometric interpolation function.

- c) Approximation of the solution function and the test function on each element:

$$\begin{aligned}\mathbf{u}^e(\mathbf{x}, t) &= \sum_i \mathbf{N}_i(\xi, \eta, \dots) \mathbf{u}_i(t), \\ \delta \mathbf{u}^e(\mathbf{x}) &= \sum_i \mathbf{N}_i(\xi, \eta, \dots) \delta \mathbf{u}_i,\end{aligned}\quad (4.2)$$

where:

- $\mathbf{u}_i(t)$: nodal variables of the solution function;
 $\delta \mathbf{u}_i$: nodal variables of the test function.

- d) Discretization of the elementary weak forms (Chapter 4):

$$W^e \Rightarrow W_h^e.$$

using equations (4.1) and (4.2), the quantity W_h^e is written, in the case of a dynamics problem, for instance:

$$\begin{aligned}W_h^e &= \langle \delta u \rangle \left([m] \left\{ \frac{d^2 u_i}{dt^2} \right\} + [c] \left\{ \frac{du_i}{dt} \right\} + [k] \{u_i\} - \{f\} \right) \\ &= \langle \delta u \rangle \{r\}\end{aligned}\quad (4.3)$$

where:

- $[m], [c]$ and $[k]$: elementary mass, damping and stiffness matrices;
 $\{f\}$: elementary load vector;
 $\{r\}$: elementary residual vector.

- e) Assembly of the set of elementary contributions (Chapter 4):

$$\begin{aligned}W_h &= \sum_e W_h^e, \\ &= \langle \delta U \rangle \left([M] \left\{ \frac{d^2 U}{dt^2} \right\} + [C] \left\{ \frac{dU}{dt} \right\} + [K] \{U\} - \{F\} \right) = 0, \quad \forall \{\delta U\}.\end{aligned}$$

the solution of the problem involves finding $\{U(t)\}$ such that:

$$\{R\} = [M] \left\{ \frac{d^2 U}{dt^2} \right\} + [C] \left\{ \frac{dU}{dt} \right\} + [K] \{U\} - \{F\} = 0, \quad (4.4)$$

where:

$[M]$, $[C]$ and $[K]$: global mass, damping and stiffness matrices;

$\{F\}$: global load and reaction vector;

$\{U\}$: global vector of the unknown nodal functions;

$\{R\}$: global residual vector.

- f) Time discretization (Chapter 5): if the problem is transient (first or second order in time), the time derivatives are then discretized by way of explicit or implicit finite difference schemes.
- g) Solution (Chapter 5): after the phase of time discretization and the introduction of Dirichlet boundary conditions such that:

$$\delta u = 0 \text{ on } S_u,$$

the solution is then calculated for successive times. In the case of a stationary problem, we have:

$$\{R\} = [K]\{U\} - \{F\} = \{0\}, \quad (4.5)$$

$$\text{so } \{U_{sol}\} = [K]^{-1}\{F\}.$$

- h) Post-processing: calculation of the elementary values such that:

$$\langle \partial U / \partial x \ \partial U / \partial y \rangle \dots$$

calculation of the reactions at the supports: $\{F_R\} = [K]\{U_{sol}\} - \{F\}$,

where $[K]$ is the global matrix that is not modified by the Dirichlet boundary conditions. The vector $\{F_R\}$ is non-null on the degrees of freedom associated with the imposed displacements and null everywhere else.

- i) Computer programming (Chapter 6): all the stages of the solution are processed by a computer using a software package: entering of the geometry, meshing, computation of the elementary contributions, assembly, solution and, finally, presentation of the results in the form of curves, color maps or even animations.

Example 4.1 illustrates the different stages of this approach for a monodimensional scalar field problem.

EXAMPLE 4.1. ‘Simple’ one-dimensional finite element problem

Consider a monodimensional domain with constant section (such as a pipe or a channel). The evolution over time of a scalar variable $u(x, t)$ is regulated by:

$$c \frac{\partial u}{\partial t} + a \frac{\partial u}{\partial x} - \frac{\partial}{\partial x} \left(k \frac{\partial u}{\partial x} \right) + bu - f_v = 0, \quad 0 \leq x \leq L, \quad \forall t > 0$$

with:

- the boundary conditions: $u(x=0, t) = u_s$
and $-k \frac{\partial u}{\partial x}(x=L, t) = -(\alpha \cdot u(L, t) - f_s)$,
- the initial condition : $u(x, t=0) = u_o(x)$.

where:
 $u(x, t)$: scalar quantity (concentration, temperature, etc.),
 c : accumulation coefficient (such as capacity),
 a : velocity of flow in the medium (positive or negative),
 k : diffusion coefficient,
 b : coefficient linked to loss/production,
 f_s, f_v : surfacic and volumic loads or sources/
sinks (> 0 if there is production),
 α : coefficient of heat exchange with the outside.

a) The associated weak variational form is written as:

$$\begin{aligned} W &= \int_0^L \left[\delta u \left(c \frac{\partial u}{\partial t} + a \frac{\partial u}{\partial x} + bu - f_v \right) + \frac{d \delta u}{dx} k \frac{\partial u}{\partial x} \right] dx + \left[\delta u (\alpha \cdot u - f_s) \right]_{x=L} = 0, \\ &= W_m + W_c + W_p - W_f + W_d + W_{CL} = 0, \\ &\quad 0 \leq x \leq L, \quad u(0) = u_s. \end{aligned}$$

where $\delta u(x)$ is a test function such that $\delta u(0) = 0$.

b–c) The domain of calculation is made up of “n” two-node elements:



such that:

$$V = \sum_{e=1}^n V^e, \quad W = \sum_{e=1}^n W^e$$

The test functions and solution functions are approximated on an arbitrary element by:

$$u(x, t) = \langle N(x) \rangle \{u_n(t)\} \quad \text{and} \quad \delta u(x) = \langle N(x) \rangle \{\delta u_n\},$$

where:

$$\langle N(x) \rangle = \left\langle 1 - \frac{x}{L^e}, \frac{x}{L^e} \right\rangle, \quad \{u_n(t)\} = \begin{Bmatrix} u_e \\ u_{e+1} \end{Bmatrix}, \quad \{\delta u_n\} = \begin{Bmatrix} \delta u_e \\ \delta u_{e+1} \end{Bmatrix},$$

and $\frac{du}{dx} = \langle B \rangle \{u_n(t)\}$, with $\langle B \rangle = \left\langle -\frac{1}{L^e}, \frac{1}{L^e} \right\rangle$, where L^e is the length of the element

The functions $N_i(x)$, $i = 1, 2$ are called approximation functions.

d) The discretized elementary weak formulation W_h^e is written:

$$W_h^e = W_m^e + W_c^e + W_d^e + W_p^e - W_f^e,$$

where:

$$W_m^e = \langle \delta u_n \rangle c[m] \left\{ \frac{du_n}{dt} \right\} \quad \text{where } [m] = \int_0^{L^e} \{N\} \langle N \rangle dx = \frac{L^e}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix},$$

$$W_c^e = \langle \delta u_n \rangle [k_c] \{u_n\} \quad \text{where } [k_c] = a \int_0^{L^e} \{N\} \langle B \rangle dx = \frac{a}{2} \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix},$$

$$W_k^e = \langle \delta u_n \rangle [k_d] \{u_n\} \quad \text{where } [k_d] = k \int_0^{L^e} \{B\} \langle B \rangle dx = \frac{k}{L^e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix},$$

$$W_p^e = \langle \delta u_n \rangle b[m] \{u_n\}$$

$$\text{and } W_f^e = \langle \delta u_n \rangle \{f\}, \quad \text{where } \{f\} = f_v \int_0^{L^e} \{N\} dx = \frac{f_v L^e}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}, \quad (f_v = \text{const.}).$$

In the particular case of a stationary problem, the elementary form can be reduced to:

$$W_h^e = \langle \delta u_n \rangle ([k] \{u_n\} - \{f\}),$$

$$\text{with: } [k] = ([k_c] + [k_d] + b[m]) = \begin{bmatrix} \frac{bL^e}{3} - \frac{a}{2} + \frac{k}{L^e} & \frac{bL^e}{6} + \frac{a}{2} - \frac{k}{L^e} \\ \frac{bL^e}{6} - \frac{a}{2} - \frac{k}{L^e} & \frac{bL^e}{3} + \frac{a}{2} + \frac{k}{L^e} \end{bmatrix}.$$

e) Assembly phase: Choose two meshes, respectively, comprising two and four elements, and the following numerical values:

$$\frac{\partial u}{\partial t} = 0, \quad L = 4, \quad a = 1, \quad k = 4, \quad b = 0.5, \quad f_s = 0, \quad \alpha = 0, \quad u_s = 0 \quad \text{and} \quad f_v = 1.$$

The exact solution is given by:

$$u_{ex}(x) = Ae^{-x/4} + Be^{x/2} + 2, \quad A = -1.9514 \text{ and } B = -0.0486.$$

Two-element mesh: we have: $L^e = 2$, $u_1 = 0$ and $\delta u_1 = 0$.

The local Péclet number is defined by: $P_e = \frac{aL^e}{k} = \frac{1}{2}$.

The elementary quantities are:

$$[k^{(1)}] = [k^{(2)}] = \begin{bmatrix} 11/6 & -4/3 \\ -7/3 & 17/6 \end{bmatrix}, \quad \{f^{(1)}\} = \{f^{(2)}\} = \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}.$$

Following assembly: $W = W^1 + W^2 + W_{CL} = 0$,

$$\text{i.e.: } \langle \delta u_1 \quad \delta u_2 \quad \delta u_3 \rangle \left(\begin{bmatrix} 11/6 & -4/3 & 0 \\ -7/3 & 28/6 & -4/3 \\ 0 & -7/3 & 17/6 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \\ u_3 \end{Bmatrix} - \begin{Bmatrix} 1 \\ 2 \\ 1 \end{Bmatrix} \right) = 0,$$

$\forall \langle \delta u_n \rangle$ where $\delta u_1 = 0$.

After the introduction of the boundary condition at node 1 (section 4.7.2), we obtain the solution: $u_2 = \frac{9}{13} = 0.6923$ and $u_3 = \frac{12}{13} = 0.9231$.

The exact solution at these nodes is: $u_2 = 0.6844$ and $u_3 = 0.9232$.

Calculation of the gradients on both of the elements gives:

$$\frac{du}{dx}^{(1)} = \frac{9}{26}, \quad \frac{du}{dx}^{(2)} = \frac{3}{26}.$$

The value of the reaction vector is: $\{F_R\} = [K]\{U_{sol}\} - \{F\} = \left\langle -\frac{25}{13} \quad 0 \quad 0 \right\rangle^T$.

Four-element mesh: we have: $L^e = 1$, $u_1 = 0$ and $\delta u_1 = 0$.

The local Péclet number is defined by: $P_e = \frac{aL^e}{k} = \frac{1}{4}$.

The elementary matrices and vectors are:

$$[k] = \begin{bmatrix} 11/3 & -41/12 \\ -53/12 & 14/3 \end{bmatrix}, \quad \{f\} = \frac{1}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}, \quad e = 1, \dots, 4$$

$$W = W^1 + W^2 + W^3 + W^4 + W_{CL} = 0$$

After the introduction of the boundary conditions by elimination of the corresponding rows and columns, the system is written as:

$$\begin{bmatrix} 25/3 & -41/12 & 0 & 0 \\ -53/12 & 25/3 & -41/12 & 0 \\ 0 & -53/12 & 25/3 & -41/12 \\ 0 & 0 & -53/12 & 14/3 \end{bmatrix} \begin{Bmatrix} u_2 \\ u_3 \\ u_4 \\ u_5 \end{Bmatrix} = \frac{1}{2} \begin{Bmatrix} 1 \\ 2 \\ 2 \\ 1 \end{Bmatrix}$$

The solution of the system gives us:

$$u_2 = 0.4013, u_3 = 0.6862, u_4 = 0.8622, u_5 = 0.9231.$$

The exact solution is equal to:

$$u_2^{ex} = 0.4001, u_3^{ex} = 0.6844, u_4^{ex} = 0.8605, u_5^{ex} = 0.9232.$$

Calculating the gradients on each of the four elements gives:

$$\frac{du}{dx}^{(1)} = 0.4013, \frac{du}{dx}^{(2)} = 0.2849, \frac{du}{dx}^{(3)} = 0.176, \frac{du}{dx}^{(4)} = 0.061.$$

The reactions are: $\{F_R\} = [K]\{U_{sol}\} - \{F\} = \langle -1.871 \quad 0 \quad 0 \quad 0 \quad 0 \rangle^T$

The corresponding program written in Matlab[®] is shown in Figure 4.0.

```
%----- Simple 1D finite element program
% 1D scalar problem: transport diffusion production
% Example 4.1
% au,x - k u,xx + b u - f = 0
% x=0 u = u0 ou u,n =- alpha* u + fs
% x=L u = uL ou u,n =- alpha* u + fs
% a: velocity m/s k: diffusion m^2/s
% b: 1/s production
% f: incoming source fs: incoming source on S
```

```

%      exact solution: A e^r1*x + B e^r2*x + f/b
%-----
clear all
clf
% Read: parameters: a k b f
a = 5 ; k = 4 ; b = .5 ; f = 0;
fprintf(' a=%6.3f  k=%6.3f  b=%6.3f  f=%6.3f \n',a,k,b,f)
% Read: boundary conditions on two nodes x = 0 and x = L
kcond=[1 1]; %subscript CL 1=Dirichlet ; 2=Neumann/Cauchy
vcond=[ 0 0;1 0]; % val vcond[1,2:] u0 or fs alpha
disp(' cond lim 1=Dirichlet 2= Neumann-cauchy cond');disp(kcond);
disp(' cond lim value vcond( fs alfa)');disp(vcond);
% Read: finite element data
long = 4; nelt= 4 ; % length of domain and number of elements
nnt=nelt+1;ndlt=nnt;
fprintf(' ----- long=%6.3f   nelt=%5i   nnt=%5i \n',[long,nelt,nnt])
kconec = [1:nelt;2:nelt+1]'; % connectivity table
vcor= [0: nnt-1] * long/nelt; % coordinates
disp('-----connectivity table');disp(kconec);
disp('-----table of coordinates vcor');disp(vcor)
%-----
% Finite element calculation: vke vfe and assembly
%---- initialization
vkg=zeros(ndlt);vfg=zeros(ndlt,1);
%---- Loop on the elements
for ie=1:nelt
    fprintf(' element ie = %5i\n',ie)
    xL=abs(vcor(ie+1)-vcor(ie)); % length of an element
    kloce=kconec(ie,:); % localization vector
    fprintf(' kloce = %5i %5i\n',kloce)
    % calculation of vke and vfe
    vke=[ (-a/2)+(k/xL)+ (b*xL/3) , (a/2)+(-k/xL)+(b*xL/6) ;
          (-a/2)+(-k/xL)+(b*xL/6), (a/2)+(k/xL)+(b*xL/3) ];
    vfe=[1 1]*f*xL/2;

```

```

    disp(' vke ');disp(vke);
    disp(' vfe ');disp(vfe);
    % assembly in vkg and vfg
    vkg(kloce,kloce)= vkg(kloce,kloce)+ vke;
    vfg(kloce) = vfg(kloce) + vfe';
    end
%-----
% Introduction of boundary conditions x=0 and x=L
% Technique 4.7.2-b : unit term on diagonal
vnc=[1,nnt]; % numbers of the CL nodes
for ic = 1 :2
    nc=vnc(ic);
    if(kcond(1) == 1) % Dirichlet
        vfg=vfg - vkg(:,nc)* vcond(ic); % taking account of non-zero val
        vkg(nc,:)= zeros(1,nnt); % technique: unit on diagonal
        vkg(:,nc) = zeros(nnt,1);
        vkg(nc,nc)=1; vfg(nc)= vcond(ic);
    end
    if(kcond(ic) >= 2) % Neumann or Cauchy
        vkg(nc,nc)= vkg(nc,nc) + vcond(ic,2); % stiffness Cauchy alpha
        vfg(nc) = vfg(nc) + vcond(ic,1)
    % Neumann stress
    end
end
%-----
% Solution
vsol = vkg\vf;
disp(' ----- Solution vsol ');disp(vsol);
%-----
% Exact solution

```

```
%----- Dirichlet case
% Solution A exp r1*x + B exp r2 * x
if(kcond(1) == 1 & kcond(2)==1)
    r1= (a - (a^2+4*b*k)^.5)/(2*k);
    r2= (a + (a^2+4*b*k)^.5)/(2*k);
    ab=[1 1;exp(r1*long) exp(r2*long)]; % calculation of A B with vcond
    ab= ab \ vcond(:,1);
    vexac = ab(1)*exp(r1*vcor)+ ab(2)*exp(r2*vcor);
else break; % supplement for Neumann-Cauchy
end
disp('root and a b');disp([r1,r2]);disp(Vab);
disp(' ----- Exact solution vexac ');disp(vexac);
%-----
% Display of numerical and exact solutions
plot(vcor, vsol)
hold on
plot(vcor,vexac)
% Calculation of the gradients
grade=zeros(nelt,1);
for ie=1:nelt
    xL=abs(vcor(ie+1)-vcor(ie)); % elementary length
    grade(ie)=(vsol(ie+1)-vsol(ie))/xL;
end
grade
```

Figure 4.0. Simple one-dimensional finite element program
(Example 4.1)

Matrix expression of W^e

Most often, we perform integrations by parts (see section 3.3) to decrease the order of the derivatives as much as possible. Thus, the expression of W involves derivatives of δu and of the contour integrals. The terms W^e can then be written in matrix form (stationary system):

$$W^e = \int_{V^e} (\langle \delta(\partial u^e) \rangle [H] \{ \partial u^e \} - \delta u^e \cdot f_V) dV - \int_{S_f^e} \delta u^e \cdot f_S dS \quad (4.6)$$

where:

$$\begin{aligned} \langle \partial u^e \rangle &= \langle u^e \quad \frac{\partial u^e}{\partial x} \quad \dots \quad \frac{\partial^2 u^e}{\partial x^2} \quad \dots \rangle \\ \langle \delta(\partial u^e) \rangle &= \langle \delta u^e \quad \delta \left(\frac{\partial u^e}{\partial x} \right) \quad \dots \quad \delta \left(\frac{\partial^2 u^e}{\partial x^2} \right) \quad \dots \rangle \end{aligned}$$

where:

- [H] is a matrix independent of \mathbf{u}^e and of its derivatives for linear operators \mathcal{L} . [H] is a function of \mathbf{u}^e and of its derivatives for nonlinear operators \mathcal{L} .
- f_V, f_S are the volume and surface forces or loads.
- V^e is the volume of the element.
- S_f^e is the portion of the boundary of the element e (if it exists) that coincides with the boundary S_f of V , on which integration by parts reveals a contour integral.

EXAMPLE 4.2. Matrix expression of W^e (Poisson's equation)

For Poisson's equation, presented in Example 3.1, the elementary integral expression W^e is written as follows by eliminating the indices e from u^e and δu^e :

- before integration by parts (see Example 3.3):

$$W = \sum_e W^e = \sum_e \int_{V^e} \delta u \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f_V \right) dV = 0;$$

- After integration by parts (see Example 3.5):

$$\begin{aligned} W = \sum_e W^e &= \sum_e \left(\int_{V^e} (\langle \delta(\partial u) \rangle [H] \{ \partial u \} - \delta u f_V) dV - \right. \\ &\quad \left. - \int_{S_f^e} \delta u (f_S - \alpha u) dS \right) = 0 \end{aligned}$$

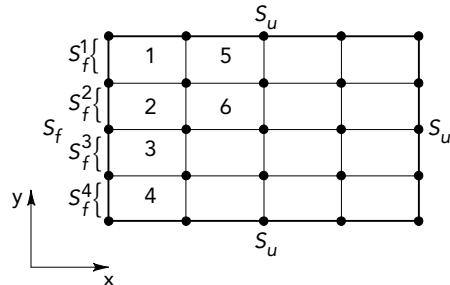
where:

$$\langle \delta(\partial u) \rangle = \left\langle \delta \left(\frac{\partial u}{\partial x} \right) \quad \delta \left(\frac{\partial u}{\partial y} \right) \right\rangle$$

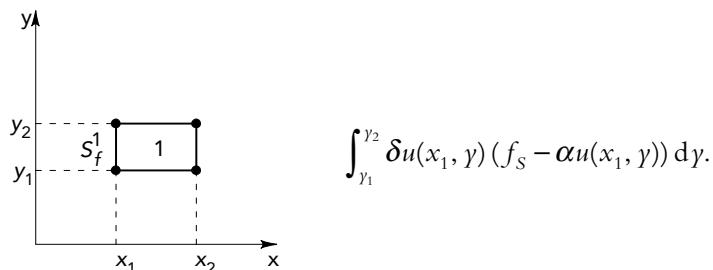
$$\langle \partial u \rangle = \left\langle \frac{\partial u}{\partial x} \quad \frac{\partial u}{\partial y} \right\rangle$$

$$[H] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Consider the rectangular domain V divided into rectangular elements V^e . Its boundary S is divided into two sections S_u and S_f :



The contour integral thus only exists on one side of the elements 1, 2, 3 and 4; for element 1, it is written as:



4.1.2 CONDITIONS FOR CONVERGENCE OF THE SOLUTION [BA1 90; ZIE 00; HUG 87; STR 73; BAT 76]

The finite element method yields an approximate solution. To guarantee convergence toward the exact solution as the size of the elements decreases, the approximation must satisfy the following conditions:

- consistency,
- interelement continuity,
- stability of the discretized system.

4.1.2.1 Consistency

Consistency has to do with the correct representation of the terms of expression W in the space of finite element approximation. For the approximate solution to tend toward the exact solution as the size h of the elements tends toward zero, the approximation errors for all the terms of W^e must be of the order h^n where $n \geq 1$. We have seen in section 1.7 that the approximation of \mathbf{u} must use at least one complete polynomial basis of up to order m to ensure the convergence of the m -order derivatives of \mathbf{u} .

For instance, for a one-dimensional problem defined by:

$$W^e = \int_0^{L^e} \left(\frac{\partial \delta u}{\partial x} \cdot k \cdot \frac{\partial u}{\partial x} + \delta u \cdot b \cdot u \right) dx$$

the approximation of u and δu must at least use the complete linear polynomial basis $(1, x)$.

For the following two-dimensional variational expression:

$$W^e = \int_A \left(\frac{\partial \delta u}{\partial x} \frac{\partial u}{\partial x} + \frac{\partial \delta u}{\partial y} \frac{\partial u}{\partial y} \right) dxdy$$

the approximation of u and δu must at least use the complete linear polynomial basis $(1, x, y)$.

For the case:

$$W^e = \int_A \left(\frac{\partial^2 \delta u}{\partial x^2} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 \delta u}{\partial y^2} \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 \delta u}{\partial x \partial y} \frac{\partial^2 u}{\partial x \partial y} \right) dxdy \quad (4.7a)$$

the approximation of u and δu must at least use the complete quadratic polynomial basis $(1, x, y, x^2, xy, y^2)$.

In order to satisfy the consistency criterion, the approximation must correctly represent the constant state of all the terms of W^e . In deformable solid mechanics, this constant state is expressed by the presence of motions of rigid bodies and their states of constant deformation.

4.1.2.2 Continuity

The previous local condition must be joined by a global condition relating to the continuity of the approximations of \mathbf{u} and of its derivatives between the elements, so that it can be written as:

$$W \equiv \sum_e W^e.$$

The expression of W^e contains terms of derivatives on u and δu up to the m th order.

The approximate function \mathbf{u} over the entire domain V must satisfy the derivability conditions of the integral expression $W:u$ and all its derivatives up to the m th order must be bounded. If u and its derivatives up to the $m - 1$ order are continuous on the elements and on the interelement boundaries (approximation C^{m-1}), the above condition is satisfied; in this case an element is said to be **conform or compatible**. For instance, for the expression (4.7a), a compatible element must ensure the continuity of u , $\frac{\partial u}{\partial x}$ and $\frac{\partial u}{\partial y}$ at every point of V .

While it is easy to satisfy the conditions of continuity on each element, it is sometimes difficult to satisfy them on the interelement boundaries – particularly when derivatives of the order higher than 1 appear in W . An element is said to be **non-compatible** when it does not satisfy the required continuity conditions. In this case:

$$W = \sum_e W^e + W^d \quad (4.7b)$$

where W^d is a term due to the interelement discontinuities, which does not appear in the terms W^e .

In order for the convergence of the approximate solution to be correct, W^d must be null, or be limited and tend toward zero with decreasing size of the elements for any function u and its constant derivatives up to the m th order.

4.1.2.3 Stability

After finite element discretization, the weak formulation $W(u, \delta u)$ of transient problems is expressed in the general discrete (or algebraic) form:

$$W^h = \langle \delta U \rangle \left([M] \left\{ \frac{d^2 U}{dt^2} \right\} + [C] \left\{ \frac{d U}{dt} \right\} + [K] \{U\} - \{F\} \right) = 0, \quad \forall \langle \delta U \rangle.$$

so: $[M] \left\{ \frac{d^2 U}{dt^2} \right\} + [C] \left\{ \frac{d U}{dt} \right\} + [K] \{U\} - \{F\} = 0.$

In the case of a stationary problem, the system is limited to:

$$[K] \{U\} = \{F\}.$$

To ensure convergence, in addition to the conditions of consistency and continuity, it is necessary to verify the following points:

- The matrix $[K]$ must be invertible (non-singular) after introduction of necessary Dirichlet-type boundary conditions (nullification of the motions of rigid bodies). If the expression of $W(u, \delta u)$ is positive definite or elliptical, i.e. such that:

$$W(u,u) > 0, \forall u \neq 0,$$

a consistent and continuous approximation generally guarantees convergence.

- Spurious oscillations must be avoided for any mesh. This problem is often encountered in transport/diffusion-type problems. When the local Péclet number, defined by:

$$P_e = a \cdot L^e / k,$$

where (see Example 4.1): a : convection coefficient,

k : diffusion coefficient,

L^e : length of the element,

is greater than a critical value yet to be determined, spurious oscillations may appear for certain types of boundary conditions. Then we have to ensure the positivity of the discrete scheme (see Example 4.3 and section 4.9) to stabilize the scheme.

- The absence of the numerical phenomenon of “locking” must be verified (section 4.9.4). This phenomenon, usually encountered with problems where the approximation has to respect certain additional constraints, results in numerical behavior of the system which is totally incongruous with the behavior which is actually observable, or at least expected. For instance, for thin structures, the discretized system may be dominated by parasite shear terms leading to highly rigid behavior instead of reproducing the flexible bending behavior. When constraint is introduced by way of a penalty parameter, the rank of the matrix associated with that constraint must be lower than the rank of the complete matrix in order to avoid any locking.
- The absence of spurious modes must be verified. These modes appear as additional modes of rigid body displacements. They generally appear in the case of mixed models, in case of reduced integrations, in incompressible fluid mechanics, etc.

We must ensure that the rank of the elementary matrix satisfies:

$$\text{rank}([k]) = \text{dimension}([k]) - (\text{number of rigid body modes}),$$

The rank of a matrix is defined by the number of non-null eigenvalues. Elastic problems exhibit a single mode of rigid body motion in one dimension, three in two dimensions and six in three-dimensional space. A Laplacian-type problem has a single rigid body mode no matter what the geometric dimension of the problem.

Section 4.9 touches on some aspects of the stability of a discretized system. In the case of a transient problem, it is also necessary to ensure that the time scheme is both stable and consistent (see Chapter 5).

EXAMPLE 4.3. Rigid body modes

Rigid body modes are characterized by fields $u(x) \neq 0$ such that:

$$W_{\text{int}}(\delta u, u) = W(u, u) = 0,$$

where $W_{\text{int}}(\delta u, u)$ is the variational form associated with the internal stresses. They are also called zero energy modes. Consider the following examples:

a) $W_{\text{int}} = \int_0^L \frac{d\delta u}{dx} \cdot \frac{du}{dx} dx,$

W_{int} is zero for the set of solutions $u(x)$ that satisfy $\frac{du}{dx} = 0$. The only rigid body mode is then defined by $u(x) = \text{constant}$. The elementary matrix for a two-node element is:

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

Its rank is equal to 1 (number of independent columns) and we can indeed easily verify that:

$$\text{rank}([k]) = \text{dimension}([k]) - (\text{number of rigid body modes})$$

Spurious modes appear when we have:

$$\text{dimension}([k]) - \text{rank}([k]) > (\text{number of rigid body modes}).$$

b) $W_{\text{int}} = \int_0^L \frac{d^2\delta u}{dx^2} \cdot \frac{d^2u}{dx^2} dx,$

W_{int} is zero for the set of solutions $u(x)$ that satisfy $\frac{d^2u}{dx^2} = 0$,

i.e.: $u(x) = ax + b$, so two modes in total.

c) The variational form associated with the internal stresses for a bending beam is written as:

$$W_{\text{int}} = \int_0^L \left(\frac{d\delta\beta}{dx} H_f \frac{d\beta}{dx} + G \left(\delta\beta + \frac{d\delta w}{dx} \right) \left(\beta + \frac{dw}{dx} \right) \right) dx,$$

The two rigid body modes are associated with: $\frac{d\beta}{dx} = 0$ and $\beta + \frac{dw}{dx} = 0$, so $\beta = \text{constant} = a$ and $w = b - ax$.

In order to eliminate these two modes, therefore, it is possible either to impose β and w at a node or w at two nodes.

d) Let us return to the variational form (limited to just the stationary terms) in Example 4.1:

$$W_{\text{int}} = \int_0^L \left[\delta u \left(a \frac{\partial u}{\partial x} + bu \right) + \frac{d\delta u}{dx} k \frac{\partial u}{\partial x} \right] dx$$

This equation has no rigid body modes at all because there is no solution field $u(x) \neq 0$ that nullifies W .

Indeed, in the case where $u(x) = \text{constant} = u_0$, we have:

$$W_{\text{int}} = \int_0^L \delta u \cdot b \cdot u_0 dx \neq 0, \quad \forall \delta u.$$

For $L = 2$, $a = 1$, $k = 4$, $b = 0.5$, the associated matrix is Example 4.1:

$$[k] = \begin{bmatrix} 11/6 & -4/3 \\ -7/3 & 17/6 \end{bmatrix}, \quad \det(k) = 25/12,$$

with the rank of $[k]$ being 2 here. In the particular case where $b = 0$, we have a single rigid body mode defined by $u(x) = \text{constant}$ and $W_{\text{int}} = 0$.

The associated matrix is: $[k] = \begin{bmatrix} 3/2 & -3/2 \\ -5/2 & 5/2 \end{bmatrix}, \quad \det(k) = 0$,

The rank of $[k]$ is now 1.

For the case where $W_{\text{int}} = \int_0^L \delta u_{,x} \cdot a \cdot u dx$, there is a single rigid body mode

$u(x) = \text{constant} = u_0$. The above reasoning can indeed be applied to the quadratic form:

$$W_{\text{int}} = \int_0^L u_{,x} \cdot a \cdot u dx = \langle u_1 \quad u_2 \rangle \frac{a}{2} \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix},$$

with the rank of $[k]$ being equal to 1.

e) Consider a two-dimensional planar linear elasticity problem:

$$W = \int_A \langle \delta \epsilon \rangle [H] \{ \epsilon \} dA$$

with: $\{ \epsilon \} = \begin{Bmatrix} \partial u / \partial x \\ \partial v / \partial y \\ \partial u / \partial y + \partial v / \partial x \end{Bmatrix}$ (deformations),

$$[H] = \frac{E}{1-v^2} \begin{bmatrix} 1 & v & 0 \\ v & 1 & 0 \\ 0 & 0 & \frac{1-v}{2} \end{bmatrix}$$
 (elastic properties).

There are three rigid body modes associated with null deformations, two translational motions:

$$u = u_0 \quad \text{and} \quad v = v_0$$

and a rotation in the sense of small rotations: $u = cy$ and $v = -cx$.

4.1.2.4 Concept of positivity

The absence of spurious oscillations in any discrete scheme, whether stationary or transient, is a gauge of its numerical stability. The concept of positivity is based on a heuristic approach. If we limit ourselves to a monodimensional approach, we can express it as follows:

Any value assumed by a solution function at a node “j” contained between the neighboring nodes “j–1” and “j+1”, must be positive (or negative, respectively) if the values attached to those neighboring nodes are positive (or negative, respectively).

This condition is all the more important to satisfy when the solution presents “steep” profiles (high gradients, stationary or transient) such as shocks. In practical terms, the concept is dealt with slightly differently depending on whether or not the problem is stationary.

a) Positivity in the case of a stationary scheme

In order to illustrate this concept, let us look at the stationary system from Example 4.1, written in the form:

$$[K] \{ U \} = \{ 0 \}.$$

The right-hand side of this equation is taken to be null because it has no effect whatsoever on the stability of the system (see the remark later on).

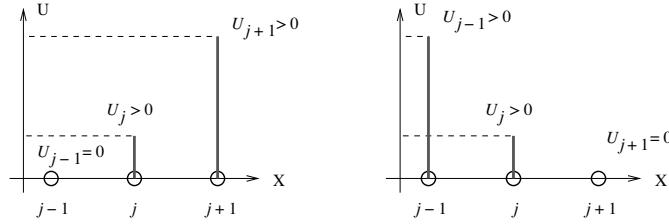
In general, the row “ j ” of the system, which expresses the local equilibrium at the node “ j ”, can be written in the following recurring form:

$$a_{j-1} \cdot u_{j-1} + a_j \cdot u_j + a_{j+1} \cdot u_{j+1} = 0, \quad (4.7c)$$

where the coefficients a_{j-1} , a_j and a_{j+1} depend on the discretization scheme. In order to ensure the positivity of the scheme, we then have to verify that:

$$\begin{cases} a_{j-1} > 0 \\ a_j < 0 \quad \text{or indeed} \\ a_{j+1} > 0 \end{cases} \quad \begin{cases} a_{j-1} < 0 \\ a_j > 0 \\ a_{j+1} < 0 \end{cases} \quad (4.7d)$$

The obtention of these criteria can be demonstrated quite simply by injecting equation (4.7c) with different profiles for u_{j-1} , u_j and u_{j+1} . Three profiles are sufficient, of which, two are illustrated here:



- The first profile is expressed by $u_{j-1} = 0$ and $u_{j+1} > 0$. Hence, equation (4.7c) is rewritten as:

$$a_j \cdot u_j + a_{j+1} \cdot u_{j+1} = 0.$$

In order to ensure that $u_j > 0$, a_j and a_{j+1} must bear opposite signs.

- Based on the same approach, the second profile illustrates the fact that a_j and a_{j-1} must also have opposite signs.

The choices of possible signs are then given by equation (4.7d).

Remarks:

- The same reasoning with negative profiles leads to the same criteria.

- The presence of a non-zero right-hand side does not affect the positivity of the scheme. Indeed, the positivity is assured when any disturbance introduced at a given time decreases over the course of time (or little by little in the case of a stationary problem). Thus, it is possible to posit:

$$\tilde{u}_k \leftarrow u_k + \varepsilon_k, \quad k = j-1, \quad j, \quad j+1,$$

where ε_k is the disturbance at node k . By injecting this disturbed solution into the discrete equation (4.7c), we get a recurrence equation whose right-hand side is 0:

$$\begin{aligned} & a_{j-1} \cdot u_{j-1} + a_j \cdot u_j + a_{j+1} \cdot u_{j+1} = f_j, \\ \Leftrightarrow & \underbrace{\left(a_{j-1} \cdot \tilde{u}_{j-1} + a_j \cdot \tilde{u}_j + a_{j+1} \cdot \tilde{u}_{j+1} - f_j \right)}_{=0} + a_{j-1} \cdot \varepsilon_{j-1} + a_j \cdot \varepsilon_j + a_{j+1} \cdot \varepsilon_{j+1} = 0, \\ \Leftrightarrow & a_{j-1} \cdot \varepsilon_{j-1} + a_j \cdot \varepsilon_j + a_{j+1} \cdot \varepsilon_{j+1} = 0. \end{aligned}$$

EXAMPLE 4.4. Application of the concept of positivity to Example (4.1)

The row associated with the node “ j ” in Example (4.1) is written as:

$$\begin{aligned} & a_{j-1} \cdot u_{j-1} + a_j \cdot u_j + a_{j+1} \cdot u_{j+1} = 0, \\ \text{where: } & a_{j-1} = \frac{bL^e}{6} - \frac{a}{2} - \frac{k}{L^e}, \\ & a_j = \frac{2bL^e}{3} + \frac{2k}{L^e}, \\ & a_{j+1} = \frac{bL^e}{6} + \frac{a}{2} - \frac{k}{L^e}. \end{aligned}$$

By dividing by k/L^e , we introduce the dimensionless Prandtl and Péclet numbers:
 $P_r = b \cdot L^{e^2} / k$, $P_e = a \cdot L^e / k$,

such that:

$$\left(\frac{P_r}{6} - \frac{P_e}{2} - 1 \right) \cdot u_{j-1} + \left(\frac{2}{3} P_r + 2 \right) \cdot u_j + \left(\frac{P_r}{6} + \frac{P_e}{2} - 1 \right) \cdot u_{j+1} = 0.$$

Consider the following different cases:

- Convection/diffusion, $b=0$ so $P_r=0$. The positivity is then assured for: $|P_e| < 2$.

The absolute value means that we can consider both positive and negative values for the velocity "a". The criterion can be rewritten in a more explicit form:

$$L^e < \frac{2k}{|a|},$$

expressing the maximum size of the elements to be used in order to ensure a stable scheme. Depending on the physics of the problem being studied, this condition may prove too restrictive (such is the case for low values of "k" and high velocity flows). A "shock-capturing" technique, or the addition of numerical diffusion k_{num} , is preferable, in order to ensure a reasonable size of the elements and therefore a reasonable mesh size:

$$k_{tot} = k + k_{num}, \text{ where } k_{num} = \frac{\Delta x |a|}{2},$$

the positivity is thus assured $\forall \Delta x \geq L^e$.

ii) Diffusion/production, $a = 0$. If $b > 0$, the positivity is assured this time for:

$$P_r < 6, \text{ so } L^e < \sqrt{\frac{6k}{b}}.$$

The use of a diagonal mass matrix, such that:

$$[m] = \frac{L^e}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

leads to an unconditionally positive scheme:

$$\frac{-k}{L^e} \cdot u_{j-1} + \left(bL^e + 2\frac{k}{L^e} \right) u_j - \frac{k}{L^e} \cdot u_{j+1} = 0.$$

This technique of diagonalizing the mass matrix is frequently used in transient thermics.

iii) Diffusion, $a = 0$ and $b = 0$: the scheme obtained is always stable and positive.

iv) Vibration problem, $a = 0$, $k = E$, $b = -\rho\omega^2$, where:

E : elastic modulus (N / m^2),

ρ : mass density (kg/m^3),

ω : pulsation of the forced regime (rad/s),

associated with the equilibrium equation: $E \frac{d^2u}{dx^2} + \rho\omega^2 \cdot u = 0$.

The discretized equation is written thus:

$$a_{j-1} \cdot u_{j-1} + a_j \cdot u_j + a_{j+1} \cdot u_{j+1} = 0,$$

where: $a_{j-1} = a_{j+1} = -\left(\frac{\rho\omega^2 L^e}{6} + \frac{E}{L^e}\right)$,

$$a_j = \left(\frac{2E}{L^e} - \frac{2\rho\omega^2 L^e}{3}\right).$$

The scheme will be positive if: $L^e < \frac{\sqrt{3E/\rho}}{\omega}$.

In terms of wave propagations, these expressions allow the following physical values to appear:

$$\begin{aligned} c &= \sqrt{E/\rho} && : \text{celerity of the wave (m / s)} \\ T &= 1/f = 2\pi/\omega && : \text{period of the forced vibration (s)} \\ \lambda &= cT && : \text{wavelength (m)} \end{aligned}$$

The stability criterion can be rewritten as:

$$L^e < \sqrt{3} \frac{\lambda}{2\pi}.$$

This criterion stipulates that six elements of length $L^e \approx \lambda/6$ are needed per wavelength in order to verify the positivity of the scheme while also maintaining the hope of obtaining a reasonable degree of precision. In the case of a diagonalized mass matrix, this criterion becomes more restrictive:

$$L^e < \sqrt{2} \frac{\lambda}{2\pi}.$$

Remarks

- Although it is defined by a heuristic approach, this criterion ensures that the discretized schema being studied will produce no spurious oscillation. This criterion is commonly used to develop stable schemes for compressible flow problems in the presence of shockwaves [HIR 88; HIR 90].
- The positivity of a scheme is not a necessary condition. It is possible to use schemes that are stable but non-positive to obtain solutions of perfectly acceptable quality.

b) Positivity in the case of a transient scheme

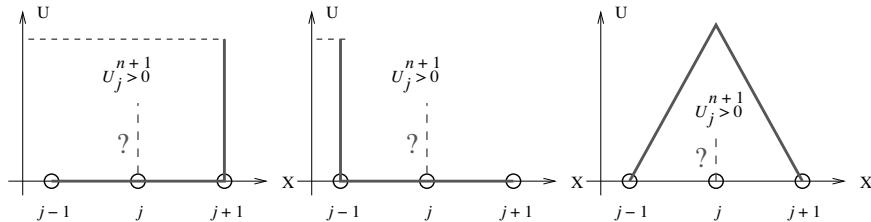
In the case of a transient discrete scheme, at node j we have the general recurrence relation:

$$u_j^{n+1} = a_{j-1} \cdot u_{j-1}^n + a_j \cdot u_j^n + a_{j+1} \cdot u_{j+1}^n. \quad (4.7e)$$

where the subscripts n and $n+1$ are associated with two successive instants of calculation. The positivity of the scheme is thus assured in accordance with the following criteria:

$$a_{j-1} > 0, \quad a_j > 0 \quad \text{and} \quad a_{j+1} > 0. \quad (4.7f)$$

The demonstration is similar to that for the previous case, and based on the fact that the solution u_j^{n+1} depends solely on the profile of the solution at time n on the three nodes $j-1$, j and $j+1$. Three “shock” profiles are sufficient to test the positivity of the solution. These profiles are chosen so that the value u_j^{n+1} is positive. They are illustrated as:



Thus, for the first case, equation (4.7e) is written as:

$$u_j^{n+1} = a_{j+1} \cdot u_{j+1}^n.$$

From this, it can be deduced that the coefficient a_{j+1} must be positive in order to ensure that $u_j^{n+1} > 0$. The same reasoning can be applied to the other two profiles for the coefficients a_{j-1} and a_j .

Remarks:

- The same reasoning with negative profiles leads us to the same criteria.
- The profiles are shown on the x -axis for simplicity's sake, but of course the criteria remain valid for “shifted” profiles (in this case, we would simply change the variables).
- The presence of a non-zero right-hand side does not affect the positivity of the scheme.

EXAMPLE 4.5. Stability and positivity

Consider the following one-dimensional transport scalar equation:

$$\frac{\partial u}{\partial t} + a \frac{\partial u}{\partial x} = 0, \quad 0 \leq x \leq L.$$

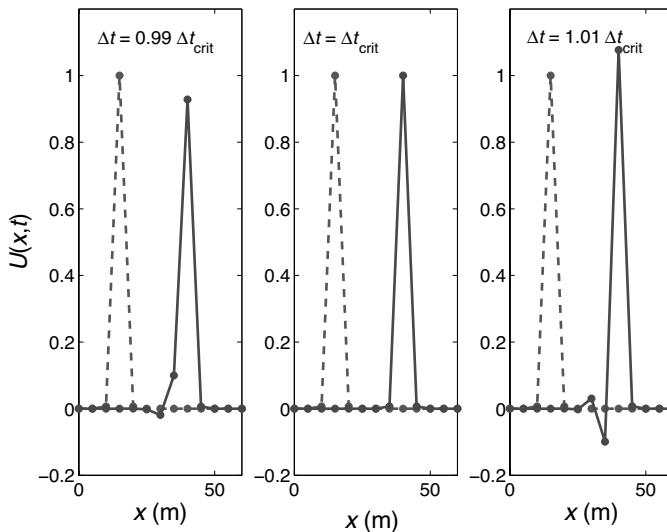
A spatially centered and time explicit discretization, with the addition of a term of numerical diffusion, gives us the scheme:

$$u_j^{n+1} = u_j^n - \frac{a \Delta t}{2 \Delta x} (u_{j+1}^n - u_{j-1}^n) + \underbrace{\frac{a \Delta t^2}{2 \Delta x^2} (u_{j+1}^n - 2u_j^n + u_{j-1}^n)}_{\text{numerical diffusion}}$$

A stability analysis using the Neumann decomposition technique (Chapter 5) yields

the criterion: $\Delta t \leq \frac{\Delta x}{a} = \Delta t_{\text{crit}}$.

For its part, the positivity of the scheme is assured for $\Delta t = \frac{\Delta x}{a}$ and gives the exact solution. For reasons of machine precision, this criterion cannot be exactly satisfied (rounding errors). Also, for a time-step smaller than the critical time-step, oscillations appear upstream of a shock profile but maintain a “moderate” amplitude (hence the stability). The figures illustrate the transport of a Gaussian (“bell-shaped” profile) for three time-steps, respectively, equal to 99%, 100% and 101% of the critical time-step. The curve shown as a dotted line is the initial solution; the solid line is the solution after 30 iterations. The second case gives the exact solution to the problem.



4.1.3 PATCH TEST

The patch test technique [IRO 72; ZIE 00] is useful for verifying the convergence of a numerical model. The pragmatic approach that it offers enables us, in particular, to look at the following aspects:

- the convergence of a model,
- the presence or absence of spurious modes or locking,
- the correct programming of the software written,
- the quality of the convergence.

However, the verification of patch tests using a finite element model is by no means a sufficient criterion to attest to the quality of the solution on any mesh.

Two patch test techniques are proposed: a numerical approach [IRO 72] and a variational approach [ZIE 00]. It is important to use the numerical patch test method to verify the quality of the calculation of the elementary quantities and of the programming.

Numerical patch test

We have shown in the previous section that the approximation of \mathbf{u} on each element must use a complete polynomial basis up to the m th order. We have to verify that this condition is also satisfied by the approximation of \mathbf{u} over the entire domain V . In theory, this condition is satisfied for conformable finite elements if the approximation is complete up to the m th order on each of the elements. On the contrary, the numerical integration of elementary quantities may introduce certain errors. The numerical patch test can be carried out in two ways: a kinematic approach and a mechanical approach.

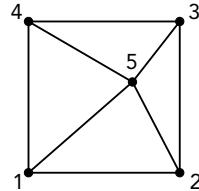
Let us choose an m -order polynomial $P_m(\mathbf{x})$ that is a solution to the problem:

$$u(\mathbf{x}) = P_m(\mathbf{x}).$$

The problem is generally defined by the partial differential equation:

$$\mathcal{L}(P_m) + f_v(\mathbf{x}) = 0 \text{ or indeed } W(\delta u, P_m(\mathbf{x})) = 0.$$

Choose an arbitrary domain comprising a few elements (a patch of elements) that includes at least one internal node. For simplicity's sake, it is common to consider a rectangular domain (like a “brick” in three dimensions) with a decentered internal node, as in the example illustrated:



a) Kinematic or Dirichlet patch test

Boundary conditions on u are imposed at the external nodes (nodes 1, 2, 3 and 4 for the above example). The values to be imposed are $u_i = P_m(\mathbf{x}_i)$ where \mathbf{x}_i are the coordinates of the nodes.

Because P_m is a solution to the problem, $W(P_m) = 0$, it is then sufficient to verify that the solution obtained using the finite element method leads to $u_5 = P_m(\mathbf{x}_5)$.

We can instead use the global matrix $[K]$ and to calculate the nodal values of the residual defined by:

$$\{R\} = [K] \{u_i = P_m(\mathbf{x}_i)\}.$$

where u_i are the values at five nodes.

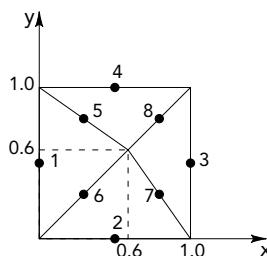
The residue value at the internal nodes (node 5), therefore, has to be zero.

b) Mechanical or Neumann patch test

This test is used to verify the absence of spurious modes and to test the convergence of the discrete model. The technique involves imposing a condition on the solution (Dirichlet) to eliminate the rigid body modes. Everywhere else, a Neumann-type condition is imposed, in accordance with $u(\mathbf{x}) = P_m(\mathbf{x})$ on the boundary.

EXAMPLE 4.6. Numerical patch test for the Laplace equation

Let us apply the test to the non-compatible three-node element described in section 2.3.3.6. We will use the following domain:



For the Laplace equation, $m = 1$. The integral expression W is given in Example 4.1, where $f_V = f_S = \alpha = 0$.

$$W = \int_A \left(\frac{\partial \delta u}{\partial x} \frac{\partial u}{\partial x} + \frac{\partial \delta u}{\partial y} \frac{\partial u}{\partial y} \right) dx dy - \int_{S_f} \delta u \cdot f ds = 0$$

where $\frac{\partial u}{\partial n} = f$ on S_f .

Let us choose: $P_m(x, y) = a_0 + a_1 x + a_2 y$

where, for instance: $a_0 = a_1 = 1$ $a_2 = 0$.

a) Kinematic or Dirichlet patch test

We impose u as boundary conditions at the external nodes 1, 2, 3 and 4.

$$u_1 = P_m(0, 0, 5) = 1,0; u_2 = 1,5; u_3 = 2,0; u_4 = 1,5.$$

The solution of this problem using the finite element method (see the next part of this chapter) gives us the values of u at the interior nodes: u_5, u_6, u_7 and u_8 . The patch test consists of verifying that:

$$\begin{aligned} u_5 &= 1,3 = P_m(0, 3, 0, 8); u_6 = 1,3 = P_m(0, 3, 0, 3); \\ u_7 &= 1,8 = P_m(0, 8, 0, 3); u_8 = 1,8 = P_m(0, 8, 0, 8). \end{aligned}$$

b) Mechanical or Neumann patch test

The Neumann condition is imposed on the contour of the domain. However, it is necessary to impose a Dirichlet condition on one of the nodes to eliminate the one rigid body mode in the problem, that being:

$$u_1 = P_m(0, 0, 5) = 1.$$

We impose the following flux conditions on the contour:

$$\text{Side 2: } \frac{\partial P_m}{\partial n} / \partial y = -\frac{\partial P_m}{\partial y} = 0$$

$$\text{Side 3: } \frac{\partial P_m}{\partial n} / \partial x = \frac{\partial P_m}{\partial x} = 1$$

$$\text{Side 4: } \frac{\partial P_m}{\partial n} / \partial y = \frac{\partial P_m}{\partial y} = 0$$

The nodal load on node 3 is equal to 1, whereas it is 0 on the other nodes. The numerical solution for all eight nodes is then given by:

$$\langle u \rangle = \langle 1.0 \quad 1.5 \quad 2.0 \quad 1.5 \quad 1.3 \quad 1.3 \quad 1.8 \quad 1.8 \rangle.$$

Given that this solution does indeed correspond to $u(\mathbf{x}) = P_m(\mathbf{x})$, the patch test is conclusive.

Remarks

- While it is possible to perform this patch test with a single element, it is preferable to consider a patch of elements.
- The verification of the mechanical patch test guarantees the convergence of the model but does not guarantee the quality of the solution and the absence of numerical oscillations.

Variational method

This method consists simply of verifying, at the level of the Galerkin integral forms (3.43), that:

$$W(P_m) = \sum_e W^e(P_m) = 0. \quad (4.8a)$$

By integration by parts, this expression is transformed into a sum of integrals on the contour of each element:

$$\sum_e \int_{S^e} \dots = 0. \quad (4.8b)$$

The expression of each integral $\int_{S^e} \dots$ is divided into two parts:

- The conformable part such that its integral on each side of the element involves only the nodal variables on the side in question.
- The non-conformable part such that its integral on each side of the element involves nodal variables that do not belong to that side.

During the course of assembly, the conformable parts of (4.8b) vanish. In order to verify that (4.8b) is satisfied, it is sufficient to consider only the non-conformable part. If we demonstrate that

$$\oint_{S^e} (\text{non-conformable part}) \cdot ds = 0 \quad (4.8c)$$

for each element, equation (4.8b) is verified, no matter what mesh is being used (in the case where $\partial P_m / \partial n$ is constant).

EXAMPLE 4.7. Variational patch test for the Laplace equation

The elementary integral form from Example 4.1 is written as:

$$W^e = \int_{V^e} \left(\delta \left(\frac{\partial u}{\partial x} \right) \frac{\partial u}{\partial x} + \delta \left(\frac{\partial u}{\partial y} \right) \frac{\partial u}{\partial y} \right) dV.$$

We integrate by parts:

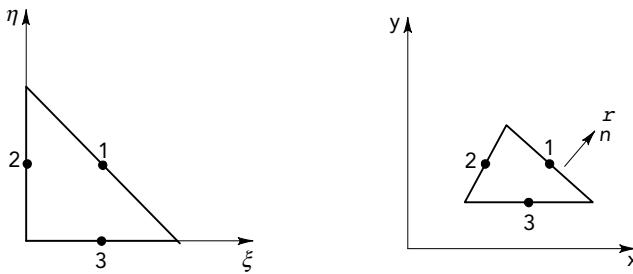
$$W^e = - \int_{V^e} \delta u \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) dV + \int_{S^e} \delta u \frac{\partial u}{\partial n} dS.$$

We replace u with the polynomial P_m defined in Example 4.6.

Because $\frac{\partial^2 P_m}{\partial x^2} + \frac{\partial^2 P_m}{\partial y^2} = 0$ and $\frac{\partial P_m}{\partial n} = l$:

$$W^e = \int_{S^e} \delta u l dS$$

where l is the directing cosine of the normal external to the element. Consider an isolated element:



$$\delta u = \begin{pmatrix} -1 + 2\xi + 2\eta & 1 - 2\xi & 1 - 2\eta \end{pmatrix} \begin{pmatrix} \delta u_1 \\ \delta u_2 \\ \delta u_3 \end{pmatrix} = \langle N \rangle \{ \delta u_n \}.$$

Calculate the value of W^e on side 1, e.g. ($\eta = 1 - \xi$):

$$W^e = \langle \delta u_n \rangle \int_0^1 \begin{Bmatrix} 1 \\ 1 - 2\xi \\ -1 + 2\xi \end{Bmatrix} lh d\xi$$

where h is the length of side 1:

$$W^e = \langle \delta u_n \rangle \begin{Bmatrix} lh \\ 0 \\ 0 \end{Bmatrix} = \delta u_1 \cdot lh.$$

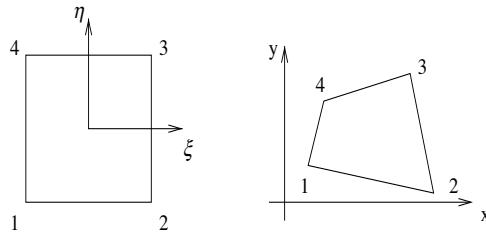
The only non-zero term in W^e is that which involves δu_1 ; it is a conformable term that vanishes with the corresponding term of the neighboring element because δu_1 and h are the same, and the sign of l changes.

In the same way, we can show that this property is verified on the other two sides. Note that for this triangle, the terms that render u discontinuous on each side have a zero integral on the corresponding side:

$$\int_{\text{each side}} N_i \, dS = 0$$

if the node i is not on the side in question.

EXAMPLE 4.8. Wilson–Taylor element [ZIE 00]



$$u(\xi, \eta) = \sum_{i=1}^4 N_i(\xi, \eta) u_i + \sum_{i=1}^2 P_i(\xi, \eta) \alpha_i,$$

$$\delta u(\xi, \eta) = \sum_{i=1}^4 N_i(\xi, \eta) \delta u_i + \sum_{i=1}^2 P_i(\xi, \eta) \delta \alpha_i,$$

$$\langle N_i \rangle = \frac{1}{4} \langle (1-\xi)(1-\eta) \quad (1+\xi)(1-\eta) \quad (1+\xi)(1+\eta) \quad (1-\xi)(1+\eta) \rangle,$$

$$P_1 = (1-\xi^2), \quad P_2 = (1-\eta^2); \quad \text{non-conformable modes.}$$

The internal modes P_1 and P_2 are non-conformable because they are non-null on the contour of the element. In order to ensure convergence, the non-conformable modes must satisfy (4.8c) for $u(x, y) = P_m(x, y) = a_0 + a_1x + a_2y$.

Hence:

$$\oint_{S^e} (1-\xi^2) \, ds = 0, \quad \oint_{S^e} (1-\eta^2) \, ds = 0.$$

For any element:

$$\oint_S (1-\xi^2) \, ds = \frac{2}{3} (l_{1-2} - l_{4-3}) \neq 0, \quad l_{1-2} \text{ and } l_{4-3} \text{ are the lengths of the sides } 1-2 \text{ and } 4-3$$

$$\oint_S (1-\eta^2) \, ds = \frac{2}{3} (l_{2-3} - l_{1-4}) \neq 0.$$

If the element is rectangular, in this case we have $l_{1-2} = l_{4-3}$ and $l_{2-3} = l_{1-4}$, and therefore indeed $W^d = 0$.

In order to construct an arbitrary quadrilateral which satisfies $W^d = 0$, we have to ensure that:

$$\frac{\partial P_i}{\partial x} \det(J) = \frac{\partial P_i}{\partial \xi} c_1 + \frac{\partial P_i}{\partial \eta} c_2, \text{ where } c_1 \text{ and } c_2 \text{ are constants.}$$

We can then employ the following technique (see Example 1.18 for the computation of J):

$$\begin{Bmatrix} \frac{\partial P_i}{\partial x} \\ \frac{\partial P_i}{\partial y} \end{Bmatrix} = [j(\xi = 0, \eta = 0)] \begin{Bmatrix} \frac{\partial P_i}{\partial \xi} \\ \frac{\partial P_i}{\partial \eta} \end{Bmatrix} \frac{\det(J^0)}{\det(J)},$$

We can then verify that:

$$\int_A \frac{\partial P_i}{\partial x} dA = 0 \text{ and } \int_A \frac{\partial P_i}{\partial y} dA = 0,$$

so that for $P_i = (1 - \xi^2)$,

$$\int_{-1}^1 \int_{-1}^1 - (j_{11}^0 \cdot 2\xi) \det(J^0) d\xi d\eta = 0 \text{ etc.}$$

The matrix $[k]$ is then given by:

$$[k] = \int_{6 \times 6}^{A'} [B]^T [H] [B] \cdot |J| d\xi d\eta,$$

where $[B] = \begin{bmatrix} [j] & \left\{ \begin{array}{l} \partial N_i / \partial \xi \\ \partial N_i / \partial \eta \end{array} \right\}, i = 1, \dots, 4 \\ \left\{ \begin{array}{l} \partial N_i / \partial \xi \\ \partial N_i / \partial \eta \end{array} \right\}, i = 1, 2 \end{bmatrix} \frac{\det(J^0)}{\det(J)} \begin{bmatrix} \partial P_i / \partial \xi \\ \partial P_i / \partial \eta \end{bmatrix}, i = 1, 2 \right]$

Using a (2×2) Gaussian point numerical integration (see section 5.1), we get:

$$[k] = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix}, \text{ where the dimensions of } [k_{22}] \text{ are } (2 \times 2).$$

It is possible to locally eliminate the coefficients α_1 and α_2 using a static condensation in order to obtain a matrix $[k]$ whose dimensions are (4×4) :

$$[k] = [k_{11} - k_{12} \cdot k_{22}^{-1} \cdot k_{21}].$$

Remarks

- The matrix $[k_{11}]$ corresponds to the stiffness of a classical Q4 element.
- It is common not to take account of the equivalent load terms associated with P_i , namely:

$$\int_A P_i f \cdot dA = 0.$$

It is also possible to construct a patch test for transport/diffusion problems and nonlinear problems. To do so, we need only choose a linear value for $P_m(x, y, z)$ corresponding to a field with constant gradients. We then define a source term $f_v(x, y, z)$ that satisfies the partial differential equation associated with the problem. Then we need to introduce enough Dirichlet conditions to eliminate the rigid body modes. The finite element calculation that results from this must then give a solution that corresponds exactly to the chosen field $P_m(x, y, z)$.

It is possible to use this technique to evaluate the precision of a model. Indeed, we need only choose a polynomial field whose order is greater than m and adjust $f_v(x, y, z)$ to satisfy the partial differential equations. We then solve the problem for a patch test mesh or for a finer mesh created by a mesh generation tool. It is also possible to evaluate the computation error by comparing the numerical solution calculated with the chosen solution $P_m(x, y, z)$.

EXAMPLE 4.9. Example of a patch test for a transport/diffusion problem

Consider the problem described in Example 4.1:

$$a \frac{\partial u}{\partial x} - \frac{\partial}{\partial x} \left(k \frac{\partial u}{\partial x} \right) + bu - f_v = 0, \quad 0 \leq x \leq 4.$$

where: $a = 1, k = 4$ and $b = 0.5$.

We choose the following solution function with a constant gradient:

$$u_m(x) = P_m(x) = \alpha_0 + \alpha_1 x.$$

In order for $P_m(x)$ to be a solution to the problem we readjust the source term such that:

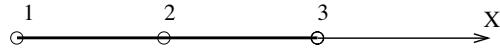
$$f(x) = a \frac{du_m}{dx} + b u_m = b \alpha_0 + a \alpha_1 + b x \alpha_1,$$

where:

$$\alpha_0 = 1, \alpha_1 = 2 \text{ and } P_m(x) = 1 + 2x,$$

$$\text{That is } f(x) = (2a+b) + 2bx = 2.5 + x.$$

We use the three-node mesh illustrated as:



Before solving the patch test, we have to fix the values of the solution at nodes 1 and 3, in accordance with the solution chosen, so that:

$$u_1 = P_m(x=0) = 1 \text{ and } u_3 = P_m(x=4) = 9.$$

$$\text{The exact solution at node 2 is } u_2 = P_m(x=2) = 5.$$

The elementary terms are then written:

$$[k^{(1)}] = [k^{(2)}] = \begin{bmatrix} 11/6 & -4/3 \\ 7/3 & 17/6 \end{bmatrix}, \{f^{(1)}\} = \begin{Bmatrix} 19/6 \\ 23/6 \end{Bmatrix}, \{f^{(2)}\} = \begin{Bmatrix} 31/6 \\ 35/6 \end{Bmatrix}.$$

Following assembly and introduction of the boundary conditions on nodes 1 and 3, we finally get:

$$\frac{28}{6} u_2 = \left(\frac{7}{3} + 12 + 9 \right), \text{ or in other words } u_2 = 5.$$

The patch test has succeeded.

4.2 Discretized elementary integral forms W^e

4.2.1 MATRIX EXPRESSION OF W^e

In order to obtain the discretized form (4.4) of W^e , we introduce into (4.6) the approximations over each element e : of u , of δu and of their derivatives (see Chapters 1 and 2):

$$\begin{aligned}
u &= \langle N \rangle \{u_n\}; & \frac{\partial u}{\partial x} &= \left\langle \frac{\partial N}{\partial x} \right\rangle \{u_n\} \\
&\vdots && \\
\delta u &= \langle N \rangle \{\delta u_n\}; & \delta \left(\frac{\partial u}{\partial x} \right) &= \left\langle \frac{\partial N}{\partial x} \right\rangle \{\delta u_n\}. \\
&\vdots &&
\end{aligned} \tag{4.9a}$$

Thus:

$$\begin{aligned}
\{\partial u\} &= \begin{Bmatrix} u \\ \frac{\partial u}{\partial x} \\ \vdots \end{Bmatrix} = \begin{Bmatrix} \langle N \rangle \\ \langle \frac{\partial N}{\partial x} \rangle \\ \dots \end{Bmatrix} \{u_n\} = [B] \{u_n\} \\
\{\delta(\partial u)\} &= \begin{Bmatrix} \delta u \\ \delta \left(\frac{\partial u}{\partial x} \right) \\ \vdots \end{Bmatrix} = \begin{Bmatrix} \langle N \rangle \\ \langle \frac{\partial N}{\partial x} \rangle \\ \dots \dots \end{Bmatrix} \{\delta u_n\} = [B_\delta] \{\delta u_n\}.
\end{aligned} \tag{4.9b}$$

For self-adjoint operators \mathcal{L} :

$$[B_\delta] \equiv [B].$$

Equation (4.6) is written in discretized form, using (4.9), as:

$$W^e = \langle \delta u_n \rangle \left(\int_{V^e} [B_\delta]^T [H][B] dV \{u_n\} - \int_{V^e} \{N\} f_v dV - \int_{S_f^e} \{N\} f_s dS \right). \tag{4.10a}$$

Thus: $W^e = \langle \delta u_n \rangle ([k]\{u_n\} - \{f\})$,

with:

$$[k] = \int_{V^e} [B_\delta]^T [H] [B] dV \tag{4.10b}$$

$$\{f\} = \int_{V^e} \{N\} f_v dV + \int_{S_f^e} \{N\} f_s dS. \tag{4.10c}$$

EXAMPLE 4.10. Discretized expression of the form W^e from Example 4.2
(Poisson's equation)

The approximation of u on the element is written as:

$$u = \langle N \rangle \{u_n\}$$

$$\{\partial u\} = \begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial u}{\partial y} \end{pmatrix} = \begin{bmatrix} \left\langle \frac{\partial N}{\partial x} \right\rangle \\ \left\langle \frac{\partial N}{\partial y} \right\rangle \end{bmatrix} \{u_n\} = [B] \{u_n\}$$

$$\{\delta(\partial u)\} = \begin{pmatrix} \delta\left(\frac{\partial u}{\partial x}\right) \\ \delta\left(\frac{\partial u}{\partial y}\right) \end{pmatrix} = \begin{bmatrix} \left\langle \frac{\partial N}{\partial x} \right\rangle \\ \left\langle \frac{\partial N}{\partial y} \right\rangle \end{bmatrix} \{\delta u_n\} = [B_\delta] \{\delta u_n\} = [B] \{\delta u_n\}.$$

Note that, in this case, $[B_\delta] \equiv [B]$ because the Laplacian operator is self-adjoint.

$$[k] = \int_{V^e} [B]^T [H] [B] dV + \int_{S_f^e} \alpha \{N\} \langle N \rangle dS.$$

The matrix $[k]$ is symmetrical.

$$\{f\} = \int_{V^e} \{N\} f_V dV + \int_{S_f^e} \{N\} f_S dS.$$

Remark

The contour term appears only for elements with a side shared with the boundary S_f . In the case of a concentrated load at point $\mathbf{x} = \mathbf{x}_i$ of the surface, the function f_V is a Dirac distribution $\delta(\mathbf{x}_i)$;

$$f_V(\mathbf{x}_i) = f_i \delta(\mathbf{x}_i)$$

the corresponding vector $\{f\}$ is written as:

$$\{f\} = \{N(\mathbf{x}_i)\} f_i.$$

4.2.2 CASE OF A NONLINEAR OPERATOR \mathcal{L}

For linear problems, $[k]$ and $\{f\}$ are independent of \mathbf{u}_n . On the contrary, for nonlinear problems, $[k]$ is dependent on \mathbf{u}_n but can be decomposed into the sum of a constant matrix $[k_l]$ and a matrix $[k_{nl}(\mathbf{u}_n)]$, which is a function of \mathbf{u}_n :

$$[k(\mathbf{u}_n)] = [k_l] + [k_{nl}(\mathbf{u}_n)] \quad (4.11)$$

EXAMPLE 4.11. Beam with large displacements

The integral form corresponding to a rectilinear beam with a rectangular section $h \times b$ subjected to a large transversal displacement $w(x)$ is (small deformations and moderate rotations):

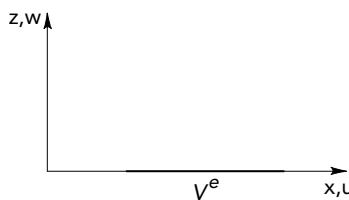
$$W^e = E h b \int_{V^e} \delta \boldsymbol{\varepsilon} \cdot \boldsymbol{\varepsilon} \, dx + \frac{E h^3 b}{12} \int_{V^e} \delta \boldsymbol{\kappa} \cdot \boldsymbol{\kappa} \, dx - \int_{V^e} \delta w \cdot f_V \, dx$$

where:

$$\begin{aligned} \boldsymbol{\varepsilon} &= u_{,x} + \frac{1}{2} w_{,x}^2 & \delta \boldsymbol{\varepsilon} &= \delta(u_{,x}) + w_{,x} \delta(w_{,x}) \\ \boldsymbol{\kappa} &= -w_{,xx} & \delta \boldsymbol{\kappa} &= -\delta(w_{,xx}) \end{aligned}$$

$$\begin{aligned} W^e &= E h b \int_{V^e} \left(\delta(u_{,x}) \left(u_{,x} + \frac{1}{2} w_{,x}^2 \right) + \delta(w_{,x}) \left(w_{,x} u_{,x} + \frac{1}{2} w_{,x}^3 \right) \right) dx + \\ &\quad + \frac{E h^3 b}{12} \int_{V^e} \delta(w_{,xx}) w_{,xx} \, dx - \int_{V^e} \delta w f_V \, dx. \end{aligned}$$

Note that here, W^e is the first variation of the total potential energy.



u, w are the axial and transversal displacements of a point on the beam;
 E, h, b are Young's modulus, the height and the thickness of the beam;
 f_V is the transversal load per unit length of the beam.

Express W^e in matrix form by separating the linear parts W_l^e and the nonlinear parts W_{nl}^e :

$$W_l^e = \int_{V^e} (\langle \delta(\partial u) \rangle [H_l] \{\partial u\} - \delta w \cdot f_V) dx$$

where:

$$\{\partial u\} = \begin{Bmatrix} u_{,x} \\ w_{,x} \\ w_{,xx} \end{Bmatrix} \quad \{\delta(\partial u)\} = \begin{Bmatrix} \delta(u_{,x}) \\ \delta(w_{,x}) \\ \delta(w_{,xx}) \end{Bmatrix}$$

$$[H_l] = \begin{bmatrix} Eh b & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{Eh^3 b}{12} \end{bmatrix}$$

$$W_{nl}^e = \int_{V^e} \langle \delta(\partial u) \rangle [H_{nl}] \{\partial u\} dx$$

$$[H_{nl}] = Eh b \begin{bmatrix} 0 & \frac{1}{2}w_{,x} & 0 \\ w_{,x} & \frac{1}{2}w_{,x}^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Let us use two different approximations for u and w :

$$u = \langle N_u \rangle \{u_n\}$$

$$w = \langle N_w \rangle \{w_n\}.$$

Thus:

$$\{\partial u\} = \begin{bmatrix} \langle N_{u,x} \rangle & 0 \\ 0 & \langle N_{w,x} \rangle \\ 0 & \langle N_{w,xx} \rangle \end{bmatrix} \begin{Bmatrix} u_n \\ w_n \end{Bmatrix} = [B] \{u_n\}$$

$$\{\delta(\partial u)\} = [B] \{\delta u_n\}.$$

The linear and nonlinear matrices are written as:

$$[k_l] = \int_{V^e} [B]^T [H_l] [B] dx$$

$$[k_{nl}] = \int_{V^e} [B]^T [H_{nl}] [B] dx \quad (\text{non-symmetrical matrix})$$

where $[H_{nl}]$ is expressed as a function of $\{w_n\}$:

$$[H_{nl}] = E h b \begin{bmatrix} 0 & \frac{1}{2} \langle N_{w,x} \rangle \{w_n\} & 0 \\ \langle N_{w,x} \rangle \{w_n\} & \frac{1}{2} (\langle N_{w,x} \rangle \{w_n\})^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

The matrix $[k_{nl}]$ is symmetrical if we use another expression of $[H_{nl}]$, while keeping the same matrix $[B]$:

$$[H_{nl}] = E h b \begin{bmatrix} 0 & \frac{1}{2} w_{,x} & 0 \\ \frac{1}{2} w_{,x} & \frac{1}{2} w_{,x}^2 + \frac{1}{2} u_{,x} & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

4.2.3 INTEGRAL FORM W^e ON THE REFERENCE ELEMENT

In Chapters 1 and 2, we constructed interpolation functions $N(\xi)$ on the reference element. The expressions (4.6) and (4.10) of W^e contain:

- derivatives in \mathbf{x} of \mathbf{u} and $\delta\mathbf{u}$;
- nodal variables \mathbf{u}_n and $\delta\mathbf{u}_n$ which, for Hermite elements, include values at the nodes of derivatives in \mathbf{x} of \mathbf{u} and $\delta\mathbf{u}$;
- integrations on the real element V^e .

Hence, we have to transform the derivations in \mathbf{x} into derivations in ξ , and the integration on V^e into an integration on the reference element V^r .

4.2.3.1 Transformation of the derivations in \mathbf{x}

The derivatives $u_{,x}, u_{,y}, u_{,z}, u_{,xx}, \dots$ are expressed as functions of $u_{,\xi}, u_{,\eta}, u_{,\zeta}, u_{,\xi\xi}, \dots$ and terms of the inverse of the Jacobian matrix $[j] = [J]^{-1}$ of the geometrical transformation, in accordance with the relations in section 1.5. For instance, in one dimension:

$$\begin{aligned} u(\xi) &= \langle N(\xi) \rangle \{u_n\} \\ \frac{du}{dx} &= \frac{d\xi}{dx} \frac{du}{d\xi} = \frac{d\xi}{dx} \langle \frac{dN(\xi)}{d\xi} \rangle \{u_n\}. \end{aligned}$$

The expression (4.9b) of the matrix $[B]$ can be reorganized:

$$[B] = [Q][B_\xi] \quad (4.12)$$

where: $[Q]$ is a transformation matrix containing terms of $[j] = [J]^{-1}$;
 $[B_\xi]$ is a matrix similar to $[B]$ but which involves derivatives in ξ of the functions $N(\xi)$ instead of derivatives in \mathbf{x} of the functions $N(\mathbf{x})$.

EXAMPLE 4.12. Transformation of the matrix $[B]$ of Poisson's integral formula

$$[B] = \begin{bmatrix} \left\langle \frac{\partial N}{\partial x} \right\rangle \\ \left\langle \frac{\partial N}{\partial y} \right\rangle \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} \end{bmatrix} \begin{bmatrix} \left\langle \frac{\partial N}{\partial \xi} \right\rangle \\ \left\langle \frac{\partial N}{\partial \eta} \right\rangle \end{bmatrix} = [Q][B_\xi].$$

In this case

$$[Q] = [j] = [J]^{-1}.$$

Thus:

$$[k] = \int_{V^e} [B_\xi]^T [Q]^T [H] [Q] [B_\xi] dV.$$

4.2.3.2 Transformation of the nodal variables

For a Hermite element, the nodal variables $\{u_n\}_\xi$ on the reference element and $\{u_n\}_x$ on the real element are linked by:

$$\begin{aligned} \{u_n\}_\xi &= [T] \{u_n\}_x \\ \{\delta u_n\}_\xi &= [T] \{\delta u_n\}_x. \end{aligned}$$

The transformation matrix $[T]$ contains terms of $[J]$ calculated at the nodes (see section 2.3.4.1).

4.2.3.3 Transformation of the domain of integration

The volume integral on V^e is replaced with the volume integral on the reference element V^r (see sections 1.6.1 and 1.44):

$$\int_{V^e} \dots dV = \int_{V^r} \dots \det(J) d\xi d\eta d\zeta. \quad (4.13)$$

The limits of integration in ξ for the conventional reference elements are:

- **One dimension**

$$\int_{\xi=-1}^{\xi=1} \dots \det(J) d\xi.$$

- **Two dimensions**

Triangle	$\int_{\xi=0}^{\xi=1} \int_{\eta=0}^{\eta=1-\xi} \dots \det(J) d\eta d\xi$
Quadrilateral	$\int_{\xi=-1}^{\xi=1} \int_{\eta=-1}^{\eta=1} \dots \det(J) d\eta d\xi$

- **Three dimensions**

Tetrahedron	$\int_{\xi=0}^{\xi=1} \int_{\eta=0}^{\eta=1-\xi} \int_{\zeta=0}^{\zeta=1-\xi-\eta} \dots \det(J) d\zeta d\eta d\xi$
Hexahedron	$\int_{\xi=-1}^{\xi=1} \int_{\eta=-1}^{\eta=1} \int_{\zeta=-1}^{\zeta=1} \dots \det(J) d\zeta d\eta d\xi$
Prism	$\int_{\xi=0}^{\xi=1} \int_{\eta=0}^{\eta=1-\xi} \int_{\zeta=-1}^{\zeta=1} \dots \det(J) d\zeta d\eta d\xi$

4.2.3.4 Transformation of the differential element dS of the contour integrals

a) Curvilinear integral

The integral

$$I = \int_S \dots dS$$

is written as a function of a curvilinear coordinate s on the curve S

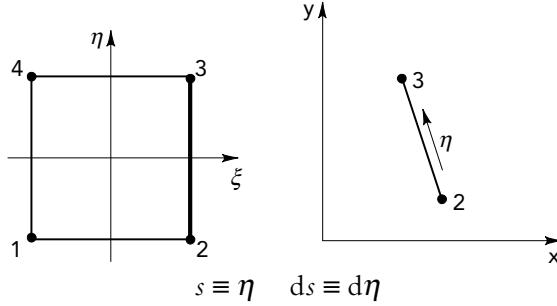
$$I = \int_{s_1}^{s_2} \dots J_s ds. \quad (4.14a)$$

The coordinate s is generally one of the variables ξ, η or ζ . The curve S corresponds to one of the sides or corners of the reference element, on which a point is defined by the parameter s :

$$\begin{aligned} x &= \langle N(s) \rangle \{x_n\} \quad \text{etc.} \\ J_s &= \sqrt{x_{,s}^2 + y_{,s}^2 + z_{,s}^2}. \end{aligned} \quad (4.14b)$$

EXAMPLE 4.13. Contour integral for a four-node element

For the side $\xi = 1$ of the quadrilateral element presented in Example 1.16:



$$\langle N(s) \rangle = \langle N(\xi = 1, \eta) \rangle = \langle 0 \ \frac{1-\eta}{2} \ \frac{1+\eta}{2} \ 0 \rangle$$

$$\text{That is, } x(\eta) = \frac{1-\eta}{2}x_2 + \frac{1+\eta}{2}x_3 \text{ and } y(\eta) = \frac{1-\eta}{2}y_2 + \frac{1+\eta}{2}y_3$$

$$x_{,\eta} = \langle N_{,\eta}(\xi = 1, \eta) \rangle \{x_n\} = \frac{1}{2}(x_3 - x_2)$$

$$y_{,\eta} = \frac{1}{2}(y_3 - y_2)$$

$$J_s = \sqrt{x_{,\eta}^2 + y_{,\eta}^2} = \sqrt{\left(\frac{x_3 - x_2}{2}\right)^2 + \left(\frac{y_3 - y_2}{2}\right)^2}$$

$$I = \int_{-1}^1 \dots J_s d\eta.$$

b) Three-dimensional surface integral

The integral

$$\int_S \dots dS$$

is written as a function of the surface coordinates s_1 and s_2 , which are generally (ξ, η) or (η, ζ) :

$$\int_S \dots J_s ds_1 ds_2. \quad (4.15a)$$

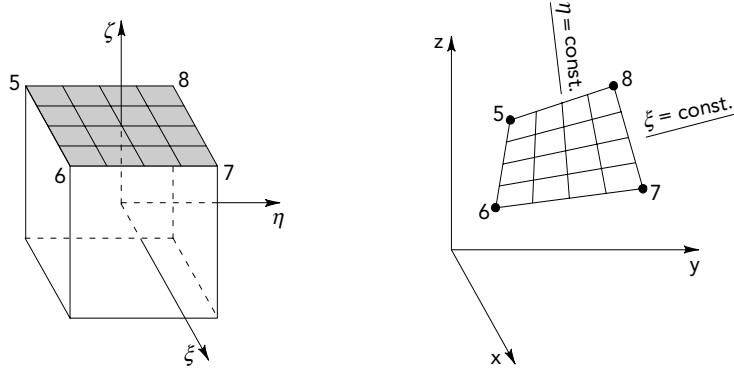
The surface S is one of the faces of the reference element. On this face, a point is identified by two parameters s_1 and s_2 :

$$x = \langle N(s_1, s_2) \rangle \{x_n\}, \quad y = \langle N \rangle \{y_n\}, \quad z = \langle N \rangle \{z_n\}.$$

$$J_S = \left| \frac{\partial \mathbf{x}}{\partial s_1} \wedge \frac{\partial \mathbf{x}}{\partial s_2} \right|, \\ = \sqrt{(\gamma_{,s_1} z_{,s_2} - z_{,s_1} \gamma_{,s_2})^2 + (z_{,s_1} x_{,s_2} - x_{,s_1} z_{,s_2})^2 + (x_{,s_1} \gamma_{,s_2} - \gamma_{,s_1} x_{,s_2})^2}. \quad (4.15b)$$

EXAMPLE 4.14. Contour integral for an eight-node element

On the face $\zeta = 1$ of the eight-node hexahedral element presented in section 2.6.1:



$$s_1 \equiv \xi \quad s_2 \equiv \eta \\ ds_1 \equiv d\xi \quad ds_2 \equiv d\eta$$

$$N(s_1, s_2) = N(\xi, \eta, \zeta = 1) = \\ \frac{1}{4} \langle 0; 0; 0; 0; (1-\xi)(1-\eta); (1+\xi)(1-\eta); \\ (1+\xi)(1+\eta); (1-\xi)(1+\eta) \rangle$$

$$N_{,\xi}(\xi, \eta, \zeta = 1) = \frac{1}{4} \langle 0; 0; 0; 0; -(1-\eta); (1-\eta); (1+\eta); -(1+\eta) \rangle$$

$$N_{,\eta}(\xi, \eta, \zeta = 1) = \frac{1}{4} \langle 0; 0; 0; 0; -(1-\xi); -(1+\xi); (1+\xi); (1-\xi) \rangle$$

$$\langle x_{,\xi} \quad y_{,\xi} \quad z_{,\xi} \rangle = \langle N_{,\xi} \rangle [\{x_n\} \{y_n\} \{z_n\}] \\ \langle x_{,\eta} \quad y_{,\eta} \quad z_{,\eta} \rangle = \langle N_{,\eta} \rangle [\{x_n\} \{y_n\} \{z_n\}]$$

J_S is given by (4.15b) and

$$l = \int_{-1}^1 \int_{-1}^1 \dots J_S(\xi, \eta) d\xi d\eta.$$

4.2.3.5 Expressions of $[k]$, $[m]$ and $\{f\}$ on the reference element

If we use integration over the reference element V^r , expressions (4.10b) and (4.10c) of the matrix $[k]$ and the vector $\{f\}$ are written as:

$$[k] = \int_{V^r} [B_\delta]^T [H] [B] \det(J) d\xi d\eta d\zeta \quad (4.16a)$$

$$\{f\} = \int_{V^r} \{N\} f_V \det(J) d\xi d\eta d\zeta + \int_{S_f^r} \{N\} f_s J_s ds_1 ds_2 \quad (4.16b)$$

$$[m] = \int_{V^r} \{N\} \langle N \rangle \det(J) d\xi d\eta d\zeta \quad (4.16c)$$

where: $[B_\delta] = [Q_\delta][B_\xi]$ and $[B] = [Q][B_\xi]$.

$[Q]$ and $[B_\xi]$ are defined by (4.12);

$[Q_\delta]$ and $[B_{\delta\xi}]$ are similar to $[Q]$ and $[B_\xi]$, only differing from them for non-self-adjoint operators;

J_s is defined by (4.14b) or (4.15b).

The integrations of (4.16) are usually performed numerically using the methods demonstrated in section 5.1 (in particular, see equation (5.4) and Example 5.4).

4.2.4 A FEW CLASSIC FORMS OF W^e AND OF ELEMENTARY MATRICES

The integral forms W^e generally comprise of a sum of several terms; for instance:

$$W^e = \int_{V^e} \left(\delta \left(\frac{\partial u}{\partial x} \right) \cdot \frac{\partial u}{\partial x} + \delta \left(\frac{\partial u}{\partial y} \right) \cdot \frac{\partial u}{\partial y} \right) dV.$$

Figure 4.1 describes the most commonly encountered terms, and the corresponding matrices $[B]$ and $[H]$.

Figure 4.2 represents the explicit form of the elementary matrix (4.16a) for two one-dimensional elements, and for classic forms of W^e .

4.3 Techniques for calculating elementary matrices

4.3.1 EXPLICIT CALCULATION FOR A TRIANGULAR ELEMENT (Poisson's equation)

Let us apply the results from the previous sections to construct the elementary matrices, in a scenario where the integrations can be performed explicitly. We will use the three-node triangular element from section 2.3.2 to evaluate the matrices $[k]$, $[m]$ and the vector $\{f\}$ formulated in Examples 4.10 and 4.12.

Terms	$[B_\xi]^T$	$[H]$	$[B]$	Properties of $[k]$
Symmetrical quadratic $\int_{\Gamma^e} \delta_u \cdot u \, dV$	$\{N\}$	1	$\langle N \rangle$	constant symmetrical
$\int_{\Gamma^e} \delta \left(\frac{\partial u}{\partial x} \right) \cdot \frac{\partial u}{\partial x} \, dV$	$\left\{ \frac{\partial N}{\partial x} \right\}$	1	$\langle \frac{\partial N}{\partial x} \rangle$	constant symmetrical
$\int_{\Gamma^e} \delta \left(\frac{\partial^2 u}{\partial x^2} \right) \cdot \frac{\partial^2 u}{\partial x^2} \, dV$	$\left\{ \frac{\partial^2 N}{\partial x^2} \right\}$	1	$\langle \frac{\partial^2 N}{\partial x^2} \rangle$	constant symmetrical
$\int_{\Gamma^e} \delta \left(\frac{\partial^m u}{\partial x^m} \right) \cdot \frac{\partial^m u}{\partial x^m} \, dV$	$\left\{ \frac{\partial^m N}{\partial x^m} \right\}$	1	$\langle \frac{\partial^m N}{\partial x^m} \rangle$	constant symmetrical
Non-symmetrical quadratic $\int_{\Gamma^e} \delta_u \cdot \frac{\partial u}{\partial x} \, dV$	$\{N\}$	1	$\langle \frac{\partial N}{\partial x} \rangle$	constant non-symmetrical
$\int_{\Gamma^e} \delta \left(\frac{\partial^m u}{\partial x^m} \right) \cdot \frac{\partial^m u}{\partial x^m} \, dV$	$\left\{ \frac{\partial^m N}{\partial x^m} \right\}$	1	$\langle \frac{\partial^m N}{\partial x^m} \rangle$	constant non-symmetrical if $m \neq n$
Nonlinear				
$\int_{\Gamma^e} \delta_u \cdot u \cdot \frac{\partial u}{\partial x} \, dV$	$\{N\}$	$\langle \frac{\partial N}{\partial x} \rangle \{u_n\}$	$\langle N \rangle$	function of $\{u_n\}$ symmetrical
$\int_{\Gamma^e} \delta \left(\frac{\partial u}{\partial x} \right) \cdot \frac{\partial u}{\partial x} \cdot \frac{\partial u}{\partial x} \, dV$	$\left\{ \frac{\partial N}{\partial x} \right\}$	$\langle \frac{\partial N}{\partial x} \rangle \{u_n\}$	$\langle \frac{\partial N}{\partial x} \rangle$	function of $\{u_n\}$ non-symmetrical
$\int \delta \left(\frac{\partial u}{\partial x} \right) \cdot \frac{\partial u}{\partial x} \cdot \frac{\partial u}{\partial x} \, dV$	$\left\{ \frac{\partial^m N}{\partial x^m} \right\}$	$\langle \frac{\partial^m N}{\partial x^m} \rangle \{u_n\}$	$\langle \frac{\partial^m N}{\partial x^m} \rangle$	function of $\{u_n\}$ symmetrical
$\int \delta \left(\frac{\partial^m u}{\partial x^m} \right) \cdot H \left(u, \frac{\partial u}{\partial x}, \dots \right) \cdot \frac{\partial^m u}{\partial x^m} \, dV$	$H(\{u_n\}) \left\{ \frac{\partial^m N}{\partial x^m} \right\}$	$\langle \frac{\partial^m N}{\partial x^m} \rangle$	function of $\{u_n\}$ non-symmetrical if $m \neq n$	

Figure 4.1. Classic forms of W^e

				Remarks
Quadratic contour terms $\int_{S^e} \delta u \cdot u \, dS$	$\{N\}$	1	$\langle N \rangle$	constant
Linear volumic terms (volumic stresses) $\int_{V^e} \delta u \cdot f_v \, dV$	$\{N\}$	1	f_v	$W^e = \langle \delta u_u \rangle \{f\}$ $\{f\} = \int_{V^e} \{N\} f_v \, dV$
Linear contour terms (surface stresses) $\int_{S_f^e} \delta u \cdot f_s \, dS$	$\{N\}$	1	f_s	$W^e = \langle \delta u_u \rangle \{f\}$ $\{f\} = \int_{S_f^e} \{N\} f_s \, dS$
Transient terms $\int_{V^e} \delta u \cdot \frac{\partial u}{\partial t} \, dV$	$\{N\}$	1	$\langle N \rangle$	$W^e = \langle \delta u_u \rangle [\epsilon] \left\{ \frac{du_u}{dt} \right\}$ $[\epsilon] = \int_{V^e} \{N\} \langle N \rangle \, dV$
$\int_{V^e} \delta u \cdot \frac{\partial^2 u}{\partial t^2} \, dV$	$\{N\}$	1	$\langle N \rangle$	$W^e = \langle \delta u_u \rangle [m] \left\{ \frac{d^2 u_u}{dt^2} \right\}$ $[m] = \int_{V^e} \{N\} \langle N \rangle \, dV$

Figure 4.1. (Continued)

Two-node linear element (section 2.2.1)

$$\int_{x_1}^{x_2} \delta u \cdot u \, dx = \langle \delta u_1 \quad \delta u_2 \rangle \frac{l}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \langle \delta u_n \rangle [m] \{u_n\}$$

$$\int_{x_1}^{x_2} \delta \left(\frac{\partial u}{\partial x} \right) \cdot \frac{\partial u}{\partial x} \, dx = \langle \delta u_1 \quad \delta u_2 \rangle \frac{1}{l} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \langle \delta u_n \rangle [k] \{u_n\}$$

$$\int_{x_1}^{x_2} \delta u \cdot \frac{\partial u}{\partial x} \, dx = \langle \delta u_1 \quad \delta u_2 \rangle \frac{1}{2} \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} u_1 \\ u_2 \end{Bmatrix} = \langle \delta u_n \rangle [k_c] \{u_n\}$$

$$l = x_2 - x_1.$$

Remark: $[m]$, $[k]$ and $[k_c]$ are used to define the mass matrix, stiffness matrix and transport matrix of a one-dimensional truss element.

Two-node Hermite cubic element (section 2.2.3.1)

$$\int_{x_1}^{x_2} \delta u \cdot u \, dx =$$

$$= \langle \delta u_n \rangle \frac{l}{420} \begin{bmatrix} 156 & 22l & 54 & -13l \\ & 4l^2 & 54 & -3l^2 \\ & & 156 & -22l \\ & & & 4l^2 \end{bmatrix}_{\text{Sym.}} \{u_n\} = \langle \delta u_n \rangle [m] \{u_n\}, \quad \text{rank}(m) = 4$$

$$\int_{x_1}^{x_2} \delta \left(\frac{\partial u}{\partial x} \right) \cdot \frac{\partial u}{\partial x} \, dx =$$

$$= \langle \delta u_n \rangle \frac{1}{30l} \begin{bmatrix} 36 & 3l & -36 & 3l \\ & 4l^2 & -3l & -l^2 \\ & & 36 & -3l \\ & & & 4l^2 \end{bmatrix}_{\text{Sym.}} \{u_n\} = \langle \delta u_n \rangle [k_m] \{u_n\}, \quad \text{rank}(k_m) = 3$$

$$\int_{x_1}^{x_2} \delta \left(\frac{\partial^2 u}{\partial x^2} \right) \frac{\partial^2 u}{\partial x^2} \, dx =$$

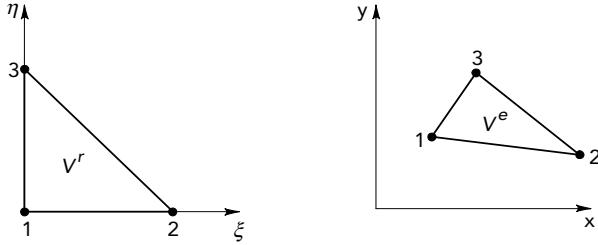
$$= \langle \delta u_n \rangle \frac{1}{l^3} \begin{bmatrix} 12 & 6l & -12 & 6l \\ & 4l^2 & -6l & -2l^2 \\ & & 12 & -6l \\ & & & 4l^2 \end{bmatrix}_{\text{Sym.}} \{u_n\} = \langle \delta u_n \rangle [k_f] \{u_n\}, \quad \text{rank}(k_f) = 2$$

$$\langle \delta u_n \rangle = \langle \delta u_1 \quad \delta u_{1,x} \quad \delta u_2 \quad \delta u_{2,x} \rangle; \quad \langle u_n \rangle = \langle u_1 \quad u_{1,x} \quad u_2 \quad u_{2,x} \rangle$$

$$l = x_2 - x_1.$$

Remark: $[m]$, $[k_m]$ and $[k_f]$ are used to define the mass matrix, axial stiffness matrix and bending stiffness matrix of a one-dimensional slender beam element.

Figure 4.2. Explicit forms of elementary matrices for two one-dimensional elements



$$\langle N \rangle = \langle 1 - \xi - \eta \quad \xi \quad \eta \rangle$$

$$[j] = \frac{1}{2A} \begin{bmatrix} y_3 - y_1 & -(y_2 - y_1) \\ -(x_3 - x_1) & x_2 - x_1 \end{bmatrix};$$

$$\det(J) = 2A = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) \quad (4.17a)$$

$$[B_\xi] = \begin{bmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}; \quad [B] = [j][B_\xi] = \frac{1}{2A} \begin{bmatrix} y_2 - y_3 & y_3 - y_1 & y_1 - y_2 \\ x_3 - x_2 & x_1 - x_3 & x_2 - x_1 \end{bmatrix}$$

$$[H] = d \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (4.17b)$$

where d is the isotropic conductivity coefficient which is equal to one for the Laplace equation.

$$[k] = \int_0^1 \int_0^{1-\xi} d[B]^T [B] \det(J) d\eta d\xi.$$

The matrix $[B]$ being constant: $[k] = A \cdot d \cdot [B]^T [B] =$

$$[k] = \frac{d}{4A} \begin{bmatrix} (y_3 - y_2)^2 & (y_3 - y_2)(y_1 - y_3) & (y_2 - y_1)(y_3 - y_2) \\ + (x_3 - x_2)^2 & +(x_3 - x_2)(x_1 - x_3) & +(x_2 - x_1)(x_3 - x_2) \\ \hline & (y_3 - y_2)^2 & (y_1 - y_3)(y_2 - y_1) \\ & +(x_3 - x_2)^2 & +(x_1 - x_3)(x_2 - x_1) \\ \hline & & (y_2 - y_1)^2 \\ & & +(x_2 - x_1)^2 \end{bmatrix} \quad (4.18)$$

symmetrical

The rank of $[k]$ is 2.

In the case where the real element is homothetic of the reference element:

$$x_1 = \gamma_1 = 0; \quad x_2 = a; \quad \gamma_2 = 0; \quad x_3 = 0; \quad \gamma_3 = a \quad \text{and} \quad A = \frac{a^2}{2}$$

$$[k] = \frac{d}{2} \begin{bmatrix} 2 & -1 & -1 \\ & 1 & 0 \\ \text{Sym.} & & 1 \end{bmatrix}. \quad (4.19)$$

The mass matrix is written (4.6b) as:

$$[m] = \int_0^1 \int_0^{1-\xi} \{N\} \langle N \rangle \det(J) d\eta d\xi \quad (4.20a)$$

$$[m] = \frac{A}{12} \begin{bmatrix} 2 & 1 & 1 \\ & 2 & 1 \\ \text{Sym.} & & 2 \end{bmatrix}. \quad (4.20b)$$

In accordance with (4.16b), the vector $\{f\}$ is written as follows, if f_V is constant and f_S is zero:

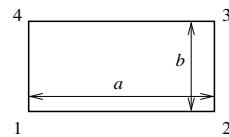
$$\{f\} = \frac{A f_V}{3} \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix}. \quad (4.20c)$$

Remark

We have:
$$\int_0^1 \int_0^{1-\xi} \xi^m \cdot \eta^n d\xi d\eta = \frac{m! n!}{(m+n+2)!}$$

4.3.2 EXPLICIT CALCULATION FOR A QUADRANGULAR ELEMENT (Poisson's equation)

For an element with rectangular geometry and with sides a and b as illustrated:



The elementary mass matrix and the elementary load vector are given by:

$$[m] = \frac{ab}{36} \begin{bmatrix} 4 & 2 & 1 & 2 \\ & 4 & 2 & 1 \\ \text{Sym.} & & 4 & 2 \\ & & & 4 \end{bmatrix}, \quad \{f\} = \frac{f_i ab}{4} \begin{Bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{Bmatrix}. \quad (4.21a)$$

In the case of an isotropic material with coefficient d , the stiffness matrix is given by:

$$[k] = d \frac{ab}{6} \begin{bmatrix} \frac{2}{a^2} + \frac{2}{b^2} & -\frac{2}{a^2} + \frac{1}{b^2} & -\frac{1}{a^2} - \frac{1}{b^2} & \frac{1}{a^2} - \frac{2}{b^2} \\ & \frac{2}{a^2} + \frac{2}{b^2} & \frac{1}{a^2} - \frac{2}{b^2} & -\frac{1}{a^2} - \frac{1}{b^2} \\ \text{Sym.} & & \frac{2}{a^2} + \frac{2}{b^2} & -\frac{2}{a^2} + \frac{1}{b^2} \\ & & & \frac{2}{a^2} + \frac{2}{b^2} \end{bmatrix}. \quad (4.21b)$$

The rank of the matrix $[m]$ is 4 and that of $[k]$ is 3.

4.3.3 ORGANIZATION OF THE CALCULATION OF THE ELEMENTARY MATRICES BY NUMERICAL INTEGRATION

For most elements, we have to use numerical integration, which will be presented in detail in section 5.1, to calculate the elementary matrices and vectors. The corresponding stages of calculation are as follows.

- a) Operations common to all elements of the same type (i.e. with the same reference element):

- calculation of the coordinates ξ_r and weights w_r corresponding to the integration points;
 - calculation of the functions N, \bar{N} and of their derivatives in ξ at the integration points (for isoparametric elements $N \equiv \bar{N}$).
- b) Operations needed to calculate the matrix $[k]$ of each element (4.16a)
where $[B_\delta] = [B]$:
- initialize $[k]$ at zero;
 - for each integration point ξ_r :
 - calculate the Jacobian matrix $[J]$ based on the derivatives in ξ of the functions \bar{N} and of the coordinates of the nodes of the element (1.43), as well as its inverse and its determinant (see 1.39–1.41);
 - calculate the derivatives of the functions N in \mathbf{x} from the derivatives in (1.37b);
 - construct the matrices $[B]$ and $[H]$;
 - accumulate in $[k]$ the product: $[B]^T [H] [B] \det(J) w_r$.
- c) Operations needed to calculate the mass matrix $[m]$ (4.16c):
- initialize $[m]$ at zero;
 - for each integration point ξ_r :
 - calculate the Jacobian matrix and its determinant;
 - accumulate in $[m]$ the product: $\{N\} \langle N \rangle \det(J) w_r$.
- d) Operations needed to calculate the load vector $\{f\}$ corresponding to a constant f_V (4.16b):
- initialize $\{f\}$ at zero;
 - for each integration point ξ_r :
 - calculate the Jacobian matrix and its determinant;
 - accumulate in $\{f\}$ the product: $\{N\} f_V \det(J) w_r$.

- e) Operations needed to calculate the residual $\{r\}$ based on the solution $\{u_n\}$ (4.3):
 - initialize the residual $\{r\}$ with $\{f\}$ calculated in step (d);
 - for each integration point ξ_r :
 - construct the matrices $[B]$, $[H]$ and $[J]$ like in step (b) above;
 - accumulate in $\{r\}$ the product: $-[B]^T [H] [B] \{u_n\} w_r \det(J)$.
- f) Operations needed to calculate the gradients $\{\partial u\}$ at the integration points based on the solution $\{u_n\}$ (4.9b):
 - for each integration point ξ_r :
 - construct the matrix $[B]$ like in step (b) above;
 - calculate the gradient: $\{\partial u\} = [B] \{u_n\}$.

A number of programs for finite element initiation are presented in Chapter 6. They are written in the programming language Matlab[®]. The structure of the programs is modular, comprising a set of scripts (also called subprograms).

In the following sections, we present the details of how to calculate the elementary terms of the variational forms given in section 3.6.

4.3.4 CALCULATION OF THE ELEMENTARY MATRICES: LINEAR PROBLEMS

This section presents the calculations for the elementary quantities for the problems laid out in section 3.6. Thus, we give the expressions of the mass matrix $[m]$, stiffness matrix $[k]$ and load vector $\{f\}$ such that:

$$W_m^e = \begin{cases} \langle \delta u_n \rangle [m] \{ \dot{u}_n \} & (\text{transient}) \\ \text{or} \\ \langle \delta u_n \rangle [m] \{ \ddot{u}_n \} & (\text{dynamic}) \end{cases} \quad W_k^e = \langle \delta u_n \rangle [k] \{ u_n \}; \quad W_f^e = \langle \delta u_n \rangle \{ f \},$$

where $\{u_n\}$, $\{\dot{u}_n\}$ and $\{\ddot{u}_n\}$ are the vectors of the elementary nodal unknowns and of their first and second time derivatives. The computer implementation is presented in Chapter 6.

4.3.4.1 One-dimensional scalar transport/diffusion problem

The elementary weak expression (section 3.6.1) is given by (3.51) with:

$$\begin{aligned} W_k^e &= \int_0^{L^e} \left[\frac{d\delta u(x)}{dx} k \frac{\partial u(x)}{\partial x} A + \delta u(x) \left(\frac{\partial}{\partial x} (\rho A a u) + bu \right) \right] dx; \\ W_m^e &= \int_0^{L^e} \delta u(x) \rho A \frac{\partial u}{\partial t} dx; \quad W_f^e = \int_0^{L^e} \delta u(x) f_A dx; \end{aligned}$$

Using a two-node linear element with two degrees of freedom (L2):

$$\langle N \rangle = \begin{pmatrix} 1 - \frac{x}{L^e} & \frac{x}{L^e} \end{pmatrix}; \quad \langle u_n \rangle = \begin{pmatrix} u_1 & u_2 \end{pmatrix}; \quad L^e = x_2 - x_1; \quad \text{section } A,$$

we get the elementary terms:

$$\begin{aligned} [m] &= \frac{\rho A L^e}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}; \quad [k] = \frac{k A}{L^e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} + \frac{\rho A a}{2} \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} + \frac{b L^e}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}; \\ \{f\} &= f_A \frac{L^e}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}. \end{aligned}$$

The elementary gradient is calculated in accordance with: $\left. \frac{\partial u}{\partial x} \right|_e = \frac{u_2 - u_1}{L^e}$.

4.3.4.2 One-dimensional problem of an elastic truss

The weak elementary expression (section 3.6.2) is given by (3.55) with:

$$\begin{aligned} W_m^e &= \int_0^{L^e} \delta u(x) \rho A \frac{\partial^2 u}{\partial t^2} dx; \quad W_k^e = \int_0^{L^e} \frac{d\delta u(x)}{dx} EA \frac{\partial u(x)}{\partial x} dx; \\ W_f^e &= \int_0^{L^e} \delta u(x) f_A dx. \end{aligned}$$

The elementary quantities for a two-node linear element (L2) are:

$$[m] = \frac{\rho A L^e}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}; \quad [k] = \frac{E A}{L^e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}; \quad \{f\} = f_A \frac{L^e}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}.$$

4.3.4.3 One-dimensional problem of a thin beam bending [BA1 90]

The weak elementary expression is given by (3.58) with:

$$W_m^e = \int_0^{L^e} \delta u(x) \rho A \frac{\partial^2 u}{\partial t^2} dx; \quad W_k^e = \int_0^{L^e} \frac{\partial^2 \delta u(x)}{\partial x^2} EI \frac{\partial^2 u(x)}{\partial x^2} dx;$$

$$W_f^e = \int_0^{L^e} \delta u(x) f_A dx.$$

For a two-node element (with the use of a reference element) and with four degrees of freedom per element:

$$\langle u_n \rangle = \begin{pmatrix} u_1 & \frac{\partial u_1}{\partial x} & u_2 & \frac{\partial u_2}{\partial x} \end{pmatrix}; \quad L^e = x_2 - x_1;$$

$$\langle N \rangle = \frac{1}{4} \begin{pmatrix} 2 - 3\xi + \xi^3 & \frac{L^e}{2}(1 - \xi^2)(1 - \xi) & 2 + 3\xi - \xi^3 & \frac{L^e}{2}(-1 + \xi^2)(1 + \xi) \end{pmatrix};$$

$$\langle N_{,xx} \rangle = \frac{1}{L^{e^2}} \begin{pmatrix} 6\xi & L^e(3\xi - 1) & -6\xi & L^e(3\xi + 1) \end{pmatrix}$$

where $\langle N_{,xx} \rangle = \frac{4}{L^{e^2}} \langle N_{,\xi\xi} \rangle$ and $\frac{dx}{d\xi} = \frac{L^e}{2}$.

The elementary quantities are:

$$[m] = \frac{\rho A L^e}{420} \begin{bmatrix} 156 & 22L^e & 54 & -13L^e \\ & 4L^{e^2} & 13L^e & -3L^{e^2} \\ & & 156 & -22L^e \\ Sym & & & 4L^{e^2} \end{bmatrix}; \quad [k] = \frac{EI}{L^{e^3}} \begin{bmatrix} 12 & 6L^e & -12 & 6L^e \\ & 4L^{e^2} & -6L^e & -2L^{e^2} \\ & & 12 & -6L^e \\ Sym & & & 4L^{e^2} \end{bmatrix};$$

$$\langle f \rangle = \frac{f_A L^e}{2} \begin{pmatrix} 1 & \frac{L^e}{6} & -1 & \frac{L^e}{6} \end{pmatrix}.$$

The deformation is calculated by: $u_{,xx} = \langle N_{,xx} \rangle \{u_n\}$.

Remark

In the case of a thick beam (with transverse shear), the stiffness matrix is written as [BA2 90]:

$$[k] = \frac{EI}{L^e^3(1-\phi)} \begin{bmatrix} 12 & 6L^e & -12 & 6L^e \\ & (4+\phi)L^e^2 & -6L^e & -(2-\phi)L^e^2 \\ & & 12 & -6L^e \\ Sym & & & (4+\phi)L^e^2 \end{bmatrix}; \quad \phi = \frac{12EI}{L^e^2 k G A}$$

where ϕ is the coefficient of correction of transverse shearing and A the section of the beam.

Remark

The nodal variables are $\langle w_1 \ w_{1,x} \text{ or } -\beta_1 \ w_2 \ w_{2,x} \text{ or } -\beta_2 \rangle$

4.3.4.4 Two-dimensional scalar Laplacian problem

For a constant thickness h , the elementary quantities are written:

$$W_m^e = h \int_A \delta u \rho \frac{\partial u}{\partial t} dA; \quad W_k^e = \int_A \langle \delta \varepsilon \rangle [H] \{ \varepsilon \} dA; \quad W_f^e = h \int_A \delta u f dA;$$

$$[H] = h \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}; \quad \{ \varepsilon \} = \begin{bmatrix} \partial u / \partial x \\ \partial u / \partial y \end{bmatrix}.$$

a) Triangular element (T3) with three degrees of freedom

The approximation is given by:

$$\langle N \rangle = \langle 1 - \xi - \eta \quad \xi \quad \eta \rangle;$$

$$[j] = [J]^{-1} = \frac{1}{2A} \begin{bmatrix} y_{31} & -y_{21} \\ -x_{13} & x_{21} \end{bmatrix}; \quad 2A = |J| = y_{31} \cdot x_{21} - y_{21} \cdot x_{13}$$

where $x_{ij} = x_i - x_j$.

The elementary quantities are written as:

$$[m] = \frac{\rho A h}{12} \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}; \quad [k] = A h [B]^T [H] [B]; \quad \{f\} = \frac{f A h}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix};$$

$$[B] = \frac{1}{2A} \begin{bmatrix} y_{23} & y_{31} & y_{12} \\ -x_{23} & -x_{31} & -x_{12} \end{bmatrix};$$

The calculation of the gradients is obtained by: $\{\boldsymbol{\varepsilon}\} = [\mathbf{B}]\{u_n\}$.

The boundary terms are calculated on L2-type elements such as:

$$W_k^{L2} = \int_0^L \delta u \alpha h u ds; \quad W_f^{L2} = \int_0^L \delta u f_s h ds;$$

$$[k_{L2}] = \frac{\alpha h L}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}; \quad \{f_{L2}\} = \frac{f_s h L}{2} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}; \quad L: \text{length of the truss element.}$$

b) Quadrilateral element (Q4) with four degrees of freedom

The approximation on this element (see Example 1.18 and section 2.4.2) is described by:

$$\langle N \rangle = \frac{1}{4} \langle (1-\xi)(1-\eta) \quad (1+\xi)(1-\eta) \quad (1+\xi)(1+\eta) \quad (1-\xi)(1+\eta) \rangle;$$

The gradient matrix is defined by:

$$[\mathbf{B}] = [j][B_\xi] \text{ where } [B_\xi] = \begin{bmatrix} \langle N_{,\xi} \rangle \\ \langle N_{,\eta} \rangle \end{bmatrix} = \frac{1}{4} \begin{bmatrix} -(1-\eta) & (1-\eta) & (1+\eta) & -(1+\eta) \\ -(1-\xi) & -(1+\xi) & (1+\xi) & (1-\xi) \end{bmatrix};$$

The Jacobian matrix is calculated by:

$$[J] = [B_\xi] \llbracket \{x_n\} : \{\gamma_n\} \rrbracket.$$

The use of a numerical integration technique with (2×2) Gaussian points (see section 5.1):

$$\xi_i = \pm \frac{1}{\sqrt{3}}, \quad \eta_i = \pm \frac{1}{\sqrt{3}}, \quad w_i = 1;$$

gives the following elementary terms.

$$[m] = \sum_{i=1}^4 \left(\rho h \{N\} \langle N \rangle w_i \det(J) \right)_{(\xi_i, \eta_i)};$$

$$[k] = \sum_{i=1}^4 \left([B]^T [H] w_i \det(J) [B] \right)_{(\xi_i, \eta_i)};$$

$$\{f\} = \sum_{i=1}^4 \left(\{N\} w_i \det(J) f h \right)_{(\xi_i, \eta_i)}.$$

The gradients are calculated by: $\{\boldsymbol{\varepsilon}\} = [\mathbf{B}]\{u_n\}$.

Remarks

- For each integration point, the calculations are performed in the following order: the Jacobian matrix, its inverse, the functions $\langle N \rangle$, and finally the matrix $[B]$.
- T6-type elements (triangles with six nodes), Q8 (quadrilaterals with eight nodes), etc., are defined using the corresponding approximation functions N_i (see Chapter 2), in conjunction with an appropriate numerical integration technique (Chapter 5). This approach is also valid for defining three-dimensional elements.

c) Two-dimensional transport/diffusion/production-type problem

The mass matrix and the load vector are explicitized in section 4.3.4.4a. The weak expression associated with the stiffness matrix is written:

$$W_k^e = \int_{A^e} (\langle \delta \epsilon \rangle [H] \{ \epsilon \} + \delta u \cdot \operatorname{div}(a h u) + \delta u b h u) dA; \quad (4.22)$$

where $\{a\} = \begin{Bmatrix} a_x \\ a_y \end{Bmatrix}$ (constant).

The elementary stiffness matrix is the sum of three contributions:

$$\begin{bmatrix} k \end{bmatrix}_{3 \times 3} = \begin{bmatrix} k_d \\ \text{diffusion} \end{bmatrix} + \begin{bmatrix} k_c \\ \text{transport} \end{bmatrix} + \begin{bmatrix} k_p \\ \text{production} \end{bmatrix},$$

where:

$$\begin{bmatrix} k_d \end{bmatrix} = A[B]^T [H] [B]; \begin{bmatrix} k_c \end{bmatrix} = \frac{A h}{3} \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix} \begin{Bmatrix} a_x & a_y \end{Bmatrix} [B]; \begin{bmatrix} k_p \end{bmatrix} = \frac{b A h}{12} \begin{Bmatrix} 2 & 1 & 1 \\ 2 & 1 & 1 \\ \text{sym.} & & 2 \end{Bmatrix}.$$

The matrix $[B]$ is given in section 4.3.4.4a.

4.3.4.5 Planar elastic problem

The integral expression corresponding to two-dimensional linear elasticity is written as [WAS 75]:

$$W_m^e = \int_{V^e} \langle \delta \mathbf{u} \rangle \rho \frac{\partial^2}{\partial t^2} \{ \mathbf{u} \} dV; W_k^e = \int_{V^e} \langle \delta \boldsymbol{\varepsilon} \rangle [H] \{ \boldsymbol{\varepsilon} \} dV; \\ W_f^e = \int_{V^e} \langle \delta \mathbf{u} \rangle \begin{Bmatrix} f_{Vx} \\ f_{Vy} \end{Bmatrix} dV + \int_{S_f^e} \langle \delta \mathbf{u} \rangle \begin{Bmatrix} f_{Sx} \\ f_{Sy} \end{Bmatrix} dS.$$

where:

$\langle u \rangle = \langle u \quad v \rangle$ are the displacements of a point;

$\langle \boldsymbol{\varepsilon} \rangle = \langle \frac{\partial u}{\partial x}; \frac{\partial v}{\partial y}; \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \rangle$ are the small deformations;

f_{Vx}, f_{Vy} are the forces per unit volume in directions x and y ;

f_{Sx}, f_{Sy} are the forces applied to S_f^e per unit surface.

$[H] = h \begin{bmatrix} d_1 & d_2 & 0 \\ d_2 & d_1 & 0 \\ 0 & 0 & d_3 \end{bmatrix}$ is the matrix of the properties of the material.

$$d_1 = \frac{E(1-\alpha\nu)}{(1+\nu)(1-\nu-\alpha\nu)}$$

$$d_2 = \frac{\nu d_1}{(1-\alpha\nu)}$$

$$d_3 = \frac{E}{2(1+\nu)}$$

where E is Young's modulus, ν Poisson's coefficient, α is equal to 0 for plain stress ($\sigma_z = 0$) and to 1 for plain strain ($\epsilon_z = 0$). $\{\boldsymbol{\sigma}\} = [H]\{\boldsymbol{\varepsilon}\}$ represents the load vector.

In order to generalize the matrix expression of the elementary quantities for T3, T6, Q4 and Q8 elements, we use the following notations:

$\langle u_n \rangle = \langle \dots \quad u_i \quad v_i \quad \dots \rangle, i = 1, \dots, n_d$ where n_d is the number of nodes (3, 6, 4 or 8).

$$\begin{Bmatrix} \mathbf{u} \\ \mathbf{v} \end{Bmatrix} = \left[\dots \begin{array}{c|c} N_i & 0 \\ \hline 0 & N_i \end{array} \dots \right]_{i=1, \dots, n_d} \begin{Bmatrix} \mathbf{u}_n \\ \mathbf{v}_n \end{Bmatrix} = [N] \{u_n\};$$

$$\begin{aligned}
[J] &= \left[\dots \begin{array}{|c|} \hline N_{i,\xi} \\ \hline N_{i,\eta} \\ \hline \end{array} \dots i = 1, \dots, n_d \right] \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_{n_d} & y_{n_d} \end{bmatrix} = [B_\xi][x_n]; \\
\{\varepsilon\} &= \left[\dots \begin{array}{|cc|} \hline N_{i,x} & 0 \\ 0 & N_{i,y} \\ \hline N_{i,y} & N_{i,x} \\ \hline \end{array} \dots i = 1, \dots, n_d \right] \{u_n\} = [B]\{u_n\}; \\
N_{i,x} &= j_{11} \cdot N_{i,\xi} + j_{12} \cdot N_{i,\eta}; \quad N_{i,y} = j_{21} \cdot N_{i,\xi} + j_{22} \cdot N_{i,\eta}.
\end{aligned} \tag{4.23b}$$

a) Triangular element (T3) with six degrees of freedom

The geometric description of the element is given in section 4.3.4.4a. The mass matrix comprises the terms of the mass matrix of the element with three degrees of freedom, i.e.:

$$[m] = \begin{bmatrix} m_{11} & 0 & \dots & m_{13} & 0 \\ 0 & m_{11} & \ddots & 0 & m_{13} \\ & & \ddots & \vdots & \\ Sym & & & m_{33} & 0 \\ & & & 0 & m_{33} \end{bmatrix}_{6 \times 6}.$$

The elementary stiffness matrix is written as:

$$[k] = A[B]^T[H][B] \text{ where } [B] = \frac{1}{2A} \begin{bmatrix} y_{23} & 0 & y_{31} & 0 & y_{12} & 0 \\ 0 & x_{32} & 0 & x_{13} & 0 & x_{21} \\ x_{32} & y_{23} & x_{13} & y_{31} & x_{21} & y_{12} \end{bmatrix}$$

$$2A = y_{31} \cdot x_{21} - y_{21} \cdot x_{31},$$

The volumic load vector is:

$$\langle f \rangle = \frac{A h}{3} \langle f_{Vx} \ f_{Vy} \ f_{Vx} \ f_{Vy} \ f_{Vx} \ f_{Vy} \rangle.$$

The boundary term is integrated on an L2-type contour element:

$$\langle f^{L2} \rangle = \frac{L h}{2} \langle f_{Sx} \ f_{Sy} \ f_{Sx} \ f_{Sy} \rangle, \quad L: \text{length of the contour element.}$$

The deformations are calculated by: $\{\varepsilon\} = [B]\{u_n\}$.

b) Quadrilateral element (Q4) with eight degrees of freedom

The geometric description of the element is given in section 4.3.4.4b. The matrix comprises the terms of the mass matrix of the element with four degrees of freedom, i.e.:

$$[m]_{8 \times 8} = \begin{bmatrix} m_{11} & 0 & \dots & m_{14} & 0 \\ 0 & m_{11} & \ddots & 0 & m_{14} \\ & & \ddots & & \vdots \\ Sym & & & m_{44} & 0 \\ & & & 0 & m_{44} \end{bmatrix}.$$

The elementary stiffness matrix and the load vector are obtained by numerical integration on four points, i.e.:

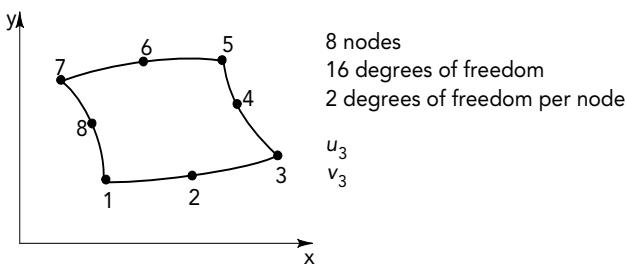
$$[k]_{8 \times 8} = \sum_{i=1}^4 [B(\xi_i, \eta_i)]^T ([H] \omega_i \det(J))_{(\xi_i, \eta_i)} [B(\xi_i, \eta_i)];$$

$$\langle f \rangle = \frac{A}{4} \langle f_{Vx} \ f_{VY} \ f_{Vx} \ f_{VY} \ f_{Vx} \ f_{VY} \ f_{Vx} \ f_{VY} \rangle; \ A = \frac{1}{2} (\gamma_{42} \cdot x_{31} - \gamma_{31} \cdot x_{42}).$$

The matrix $[B]$ is given in section 4.3.4.4b. The deformations are calculated by: $\{\varepsilon\} = [B]\{u_n\}$.

c) Quadrilateral element (Q8) with 16 degrees of freedom

Let us use the eight-node element from section 2.4.3.2:



The functions $\{N_i\}$ are defined in section 2.4.3.2. A numerical integration technique leads to the calculation of the elementary quantities:

$$\begin{aligned} [m]_{(16 \times 16)} &= \sum_{i=1}^9 \left([N]^T (\rho h \cdot \det(J) w_i) [N] \right)_{\xi_i, \eta_i}; \quad [J] = [B_\xi] \begin{bmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_8 & y_8 \end{bmatrix}; \\ [k]_{(16 \times 16)} &= \sum_{i=1}^4 \left([B]^T ([H] \cdot \det(J) w_i) [B] \right)_{\xi_i, \eta_i}; \\ \text{and } \{f\}_{(16 \times 1)} &= \sum_{i=1}^4 \left([N]^T \begin{Bmatrix} f_{Vx} \\ f_{Vy} \end{Bmatrix} \det(J) w_i \right)_{\xi_i, \eta_i}. \end{aligned} \quad (4.23c)$$

The deformations at a point are calculated by: $\{\varepsilon(\varepsilon, \eta)\} = [B(\varepsilon, \eta)]\{u_n\}$.

Remarks

- The matrix $[k]$ has a rank of 12 with three rigid body modes and a single spurious mode that vanishes when the two elements are assembled. Four integration points are sufficient; it is uncommon for nine points to be used.
- The matrix $[m]$ has a rank of eight and has eight spurious modes with four integration points. Hence, we have to use nine integration points to obtain a correct rank.

4.3.4.6 Stokes flow problem: Q4 element

The equilibrium relations are given in section 3.6.3.4. The Stokes problem is defined by considering these relations as stationary and restricted to linear terms only. The various terms making up the associated weak expression are written as:

$$\begin{aligned}
W_m &= - \int_A \delta p \left(\operatorname{Div}(\mathbf{u}) - \frac{p}{\lambda} \right) dA = 0, \quad \delta p \text{ is a scalar test function} \\
W_{NS} &= \int_A \left(\mu (\nabla \delta u \cdot \nabla u + \nabla \delta v \cdot \nabla v) - \operatorname{Div}(\delta \mathbf{u}) \cdot p - \delta u \cdot f_x - \delta v \cdot f_y \right) dA \\
&\quad + \oint_{S_f} \delta \mathbf{u} \cdot \mathbf{n} p dS = 0,
\end{aligned}$$

$\forall \delta \mathbf{u} = (\delta u, \delta v) / \delta \mathbf{u} = \mathbf{0}$ on S_u .

where \mathbf{n} is the normal outside the contour of the domain. The finite element chosen is a Q4 element for the velocity field, with the pressure being taken to be constant across the element:

$$\begin{aligned}
u(\xi, \eta) &= \sum_{i=1}^4 N_i(\xi, \eta) u_i; \quad \delta u(\xi, \eta) = \sum_{i=1}^4 N_i(\xi, \eta) \delta u_i; \\
v(\xi, \eta) &= \sum_{i=1}^4 N_i(\xi, \eta) v_i; \quad \delta v(\xi, \eta) = \sum_{i=1}^4 N_i(\xi, \eta) \delta v_i;
\end{aligned}$$

p is constant for each element.

We choose a different order of approximation for the components of the velocity and pressure, in order to avoid a numerical locking (see Example 4.23).

The elementary stiffness matrix is written in the form of four matrices:

$$[k] = \begin{bmatrix} u_1 & v_1 & \cdots & \cdots & u_4 & v_4 & p \\ u_1 & & & & & & \\ v_1 & & k_{uu} & & & -k_{up} & \\ \vdots & & 8 \times 8 & & & 8 \times 1 & \\ u_4 & & & & & & \\ v_4 & & & & & & \\ p & & & -k_{up} & & A^e / \lambda & \end{bmatrix}, \quad \begin{Bmatrix} u_1 \\ v_1 \\ \vdots \\ u_n \\ v_n \\ p \end{Bmatrix} = \begin{Bmatrix} u_1 \\ v_1 \\ \vdots \\ u_4 \\ v_4 \\ p \end{Bmatrix}$$

The matrix $[k_{uu}]$ is identical to that presented for the Laplacian problem in section 4.3.4.5b:

$$[k_{uu}] = \begin{bmatrix} k_{11} & 0 & \cdots & k_{14} & 0 \\ 0 & k_{11} & \cdots & 0 & k_{14} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ k_{44} & 0 & \cdots & k_{44} & 0 \\ sym. & & & 0 & k_{44} \end{bmatrix} \text{ and } \{k_{up}\} = \int_{-1}^1 \int_{-1}^1 \begin{Bmatrix} \partial N_1 / \partial x \\ \partial N_1 / \partial y \\ \vdots \\ \partial N_4 / \partial x \\ \partial N_4 / \partial y \end{Bmatrix} |J| d\xi d\eta = \frac{1}{2} \begin{Bmatrix} -\gamma_{42} \\ x_{42} \\ -\gamma_{13} \\ x_{13} \\ -\gamma_{24} \\ x_{24} \\ -\gamma_{31} \\ x_{31} \end{Bmatrix}.$$

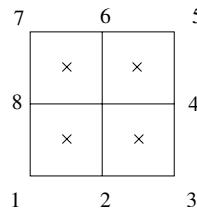
A condensation of the pressure helps to reduce the elementary stiffness matrix to an (8×8) matrix such that:

$$[k] = \begin{bmatrix} k_{uu} - \frac{\lambda}{p} \{k_{up}\} \langle k_{up} \rangle \end{bmatrix}_{8 \times 8}.$$

The pressure is then calculated during the phase of post-processing of the results on the velocity field, by the equation:

$$p = \lambda \operatorname{Div}(u) = \lambda \langle k_{up} \rangle \begin{Bmatrix} u_n \\ v_n \end{Bmatrix}.$$

This mixed element satisfies the conditions of convergence and stability in most situations that will be encountered. However, a phenomenon of locking in incompressibility may arise for a certain type of boundary conditions. Indeed, the stability of the problem requires that the rank of $[k_{uu}]$ be higher than the rank of $\{k_{up}\} = \{k_{up}\} \langle k_{up} \rangle$, following the introduction of the boundary conditions: the number of unknowns p must remain strictly less than the number of unknowns (u, v) , after the boundary conditions are introduced. Consider the patch illustrated here, comprising four Q4 elements:



For a scenario where the velocities are known all along the boundary, the only genuine unknowns that remain are four pressure values (one per element) and two velocity components (the central node). Hence, this case leads to a serious problem of locking by way of the penalty coefficient λ .

In practice, though, it is preferable to rewrite the Stokes expression in the following manner:

$$\begin{aligned} \operatorname{Div}(\bar{\sigma}) - \nabla p + \mathbf{f}_v &= \mathbf{0}; \quad \operatorname{Div}(\mathbf{u}) - \frac{p}{\lambda} = 0, \\ \bar{\sigma} = \{\sigma\} &= [H]\{\varepsilon\}, \text{ where } [H] = \mu \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

The stiffness matrix $[k_{uu}]$ is then similar to that of two dimensional planar elasticity in view of the above matrix $[H]$. This formulation, besides being more rigorous, offers the advantage of being able to correctly introduce the boundary conditions. Thus, the approach is similar to that of incompressible elasticity:

$$\begin{aligned} W^e &= \int_{A^e} \langle \delta\varepsilon \rangle [H] \{\varepsilon\} dA - \int_{A^e} \left(\operatorname{Div}(\delta\mathbf{u}) p + \delta p \left(\operatorname{Div}(\mathbf{u}) - \frac{p}{\lambda} \right) \right) dA \\ &\quad - \int_{A^e} \langle \delta u \rangle \begin{Bmatrix} f_{Vx} \\ f_{V\gamma} \end{Bmatrix} dA - \int_{S_f^e} \left(\langle \delta u \rangle \begin{Bmatrix} f_{Sx} \\ f_{Sy} \end{Bmatrix} - \delta \mathbf{u} \cdot \mathbf{n} p \right) ds \\ \text{where } \langle \varepsilon \rangle &= \left\langle \frac{\partial u}{\partial x} \quad \frac{\partial v}{\partial y} \quad \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right\rangle. \end{aligned}$$

Remark

The choice of stable finite elements for incompressible problems is fairly restricted (see Example 4.29).

4.3.4.7 Three-dimensional Laplacian-type elements: T4, H8

The elementary weak formulation associated with a Laplacian-type problem comprises the terms:

$$W_m^e = \int_V \delta u(\mathbf{x}) \rho \frac{\partial \mathbf{u}}{\partial t} dV; W_k^e = \int_V \langle \delta \boldsymbol{\varepsilon} \rangle [H] \{ \boldsymbol{\varepsilon} \} dV; W_f^e = \int_V \delta u(\mathbf{x}) \mathbf{f}_v dV;$$

$$\{ \boldsymbol{\varepsilon} \} = \begin{bmatrix} \partial u / \partial x \\ \partial u / \partial y \\ \partial u / \partial z \end{bmatrix}; [H] = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix}.$$

a) T4 element with four degrees of freedom (section 2.5)

This tetrahedral element is described by:

$$\langle N \rangle = \langle 1 - \xi - \eta - \zeta \quad \xi \quad \eta \quad \zeta \rangle;$$

$$[J] = \begin{bmatrix} x_{21} & y_{21} & z_{21} \\ x_{31} & y_{31} & z_{31} \\ x_{41} & y_{41} & z_{41} \end{bmatrix}; |J| = 6V; [B] = [j] \begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}.$$

The elementary quantities are given by:

$$[m] \frac{\rho V}{20} \begin{bmatrix} 2 & 1 & 1 & 1 \\ & 2 & 1 & 1 \\ Sym & & 2 & 1 \\ & & & 2 \end{bmatrix}; [k] = \frac{|J|}{6} [B]^T [H] [B]; \{ f \} = \frac{f_v}{4} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix};$$

$$\text{where } \int_{V'} \xi^m \eta^n \zeta^p d\xi d\eta d\zeta = \frac{m! n! p!}{(m+n+p)!}.$$

b) H8 element with eight degrees of freedom (section 2.6)

This hexahedral element is described by:

$$\langle N \rangle = \frac{1}{8} \langle \dots (1 + \xi_i \xi)(1 + \eta_i \eta)(1 + \zeta_i \zeta) \dots i = 1, \dots, 8 \rangle;$$

$$[J] = \left[\dots \begin{array}{|c|} \hline N_{i,\xi} \\ \hline N_{i,\eta} \\ \hline N_{i,\zeta} \\ \hline \end{array} \dots i = 1, \dots, 8 \right] \begin{bmatrix} x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots \\ x_8 & y_8 & z_8 \end{bmatrix} = [B_\xi] [x_n];$$

The elementary quantities are given by:

$$\begin{aligned} [m] &= \sum_{i=1}^8 (\{N\} \rho \langle N \rangle |J| \omega_i)_{\xi_i, \eta_i, \zeta_i}; \quad [k] = \sum_{i=1}^8 ([B]^T [H] [B] |J| \omega_i)_{\xi_i, \eta_i, \zeta_i}; \\ \{f\} &= f_v \sum_{i=1}^8 (\{N\} |J| \omega_i)_{\xi_i, \eta_i, \zeta_i}; \quad \xi_i = \pm \frac{1}{\sqrt{3}}, \quad \eta_i = \pm \frac{1}{\sqrt{3}}, \quad \zeta_i = \pm \frac{1}{\sqrt{3}}, \quad \omega_i = 1. \end{aligned}$$

4.3.4.8 Three dimensional element with elasticity

The terms that make up the associated weak formulation are (see section 3.6.2.3):

$$\begin{aligned} W_m^e &= \int_V \delta u(\mathbf{x}) \rho \frac{\partial^2 \mathbf{u}}{\partial t^2} dV; \quad W_k^e = \int_V \langle \delta \boldsymbol{\varepsilon} \rangle [H] \{ \boldsymbol{\varepsilon} \} dV; \quad W_f^e = \int_V \delta u(\mathbf{x}) \mathbf{f}_v dV; \\ \{\mathbf{u}\} &= \begin{Bmatrix} u \\ v \\ w \end{Bmatrix}; \quad \langle \boldsymbol{\varepsilon} \rangle = \left\langle \frac{\partial u}{\partial x} \quad \frac{\partial v}{\partial y} \quad \frac{\partial w}{\partial z} \quad \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \quad \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \quad \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right\rangle; \end{aligned}$$

$$[H] = \left[\begin{array}{ccc|ccc} d_1 & d_2 & d_2 & 0 & 0 & 0 \\ & d_1 & d_2 & 0 & 0 & 0 \\ & & d_1 & 0 & 0 & 0 \\ \hline & & & d_3 & 0 & 0 \\ sym & & & & d_3 & 0 \\ & & & & & d_3 \end{array} \right] \text{ where } \begin{cases} d_1 = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \\ d_2 = \frac{\nu E}{(1+\nu)(1-2\nu)} \\ d_3 = \frac{E}{2(1+\nu)} \end{cases}$$

In order to generalize the matrix expression of the elementary quantities for T4, H8 (etc.) elements, we use the following notations:

$$\langle u_n \rangle = \langle \dots \ u_i \ v_i \ w_i \ \dots \rangle, \quad i = 1, \dots, n_d \text{ where } n_d \text{ is the number of nodes (four or eight).}$$

$$\begin{Bmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{w} \end{Bmatrix} = \left[\dots \begin{array}{ccc|c} N_i & 0 & 0 & \dots \\ 0 & N_i & 0 & i = 1, \dots, n_d \\ 0 & 0 & N_i & \end{array} \right] \begin{Bmatrix} \mathbf{u}_n \\ \mathbf{v}_n \\ \mathbf{w}_n \end{Bmatrix} = [N] \{u_n\};$$

$$[J] = \begin{bmatrix} \cdots & | N_{i,\xi} | & \\ | N_{i,\eta} | & \cdots i = 1, \dots, n_d & \\ | N_{i,\zeta} | & & \end{bmatrix} \begin{bmatrix} x_1 & y_1 & z_1 \\ \vdots & \vdots & \vdots \\ x_{n_d} & y_{n_d} & z_{n_d} \end{bmatrix} = [B_\xi][x_n];$$

$$\{\varepsilon\} = [B]\{u_n\};$$

where:

$$[B] = \begin{bmatrix} N_{i,x} & 0 & 0 \\ 0 & N_{i,y} & 0 \\ 0 & 0 & N_{i,z} \\ N_{i,y} & N_{i,x} & 0 \\ N_{i,z} & 0 & N_{i,x} \\ 0 & N_{i,z} & N_{i,y} \end{bmatrix} \cdots i = 1, \dots, n_d \text{ and } \begin{bmatrix} N_{i,x} \\ N_{i,y} \\ N_{i,z} \end{bmatrix} = [j] \begin{bmatrix} N_{i,\xi} \\ N_{i,\eta} \\ N_{i,\zeta} \end{bmatrix}. \quad (4.23d)$$

a) T4 element with 12 degrees of freedom

The mass matrix comprises three blocks from the mass matrix given in section 4.3.4.7a. The stiffness matrix and the volumic load vector are equivalent to those given in section 4.3.4.7a where the matrix $[B]$ is defined by equation (4.23d) by considering $n_d = 4$.

b) H8 element with 24 degrees of freedom

The mass matrix comprises three blocks from the mass matrix given in section 4.3.4.7b. The stiffness matrix and the volumic load vector are equivalent to those given in section 4.3.4.7b where the matrix $[B]$ is defined by equation (4.23d) by considering $n_d = 8$.

4.4 Assembly of the global discretized form W

Assembly is the operation that consists of constructing the global matrix $[K]$ and the global load vector $\{F\}$ from the elementary matrices $[k]$ and the elementary load vectors $\{f\}$.

4.4.1 ASSEMBLY BY EXPANSION OF THE ELEMENTARY MATRICES

Each elementary integral expression W^e is written in the discretized form (4.3):

$$W^e = \langle \delta u_n \rangle ([k] \{u_n\} - \{f\})$$

where: $[k]$ is the elementary matrix of the element e ;

$\{f\}$ is the load vector of the element; it is the sum of the volumic and surface loads.

The vectors $\langle \delta u_n \rangle$ and $\{u_n\}$ are different for each element because they contain the nodal variables of the element e .

Let $\langle \delta U_n \rangle$ and $\{U_n\}$ represent the vectors formed by all the nodal variables of the entire domain V and which appear in (4.4).

$\langle \delta u_n \rangle$ and $\{u_n\}$ contain the terms from $\langle \delta U_n \rangle$ and $\{U_n\}$ which correspond to the element e :

Global variables:

$$\langle \delta U_n \rangle = \langle \delta u_1 \quad \dots \quad \delta u_i \quad \dots \quad \delta u_j \quad \dots \quad \delta u_k \quad \dots \quad \delta u_n \rangle$$

↓ ↓ ↓

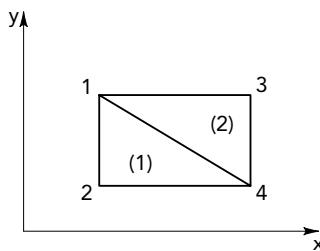
Elementary variables:

$$\langle \delta u_n \rangle = \langle \delta u_i \quad \delta u_j \quad \delta u_k \rangle$$

where $\delta u_i, \delta u_j, \delta u_k$ are the nodal variables of the element.

EXAMPLE 4.15. Elementary vector $\{u_n\}$ and global vector $\{U_n\}$

Consider the domain V represented by the two triangular elements with a single degree of freedom per node:



The global vectors are:

$$\begin{aligned}\langle \delta U_n \rangle &= \langle \delta u_1 \quad \delta u_2 \quad \delta u_3 \quad \delta u_4 \rangle \\ \langle U_n \rangle &= \langle u_1 \quad u_2 \quad u_3 \quad u_4 \rangle.\end{aligned}$$

The elementary vectors of element (1) are:

$$\begin{aligned}\langle \delta u_n^{(1)} \rangle &= \langle \delta u_1 \quad \delta u_2 \quad \delta u_4 \rangle \\ \langle u_n^{(1)} \rangle &= \langle u_1 \quad u_2 \quad u_4 \rangle.\end{aligned}$$

The elementary vectors of element (2) are:

$$\begin{aligned}\langle \delta u_n^{(2)} \rangle &= \langle \delta u_1 \quad \delta u_4 \quad \delta u_3 \rangle \\ \langle u_n^{(2)} \rangle &= \langle u_1 \quad u_4 \quad u_3 \rangle.\end{aligned}$$

The elementary integral expressions are written as:

$$\begin{aligned}W^{(1)} &= \langle \delta u_n^{(1)} \rangle ([k^{(1)}] \{u_n^{(1)}\} - \{f^{(1)}\}) \\ W^{(2)} &= \langle \delta u_n^{(2)} \rangle ([k^{(2)}] \{u_n^{(2)}\} - \{f^{(2)}\})\end{aligned}$$

Note that a nodal variable u_n (or δu_n) often appears in several elementary vectors, because a node may belong to several different elements: such is the case for nodes 1 and 4 in Example 4.15. A nodal variable must be expressed in the same frame for all the elements.

The global discretized integral form W is the sum of the discretized elementary forms W^e (4.3). This operation is called **assembly**:

$$\begin{aligned}W &= \sum_{\text{elements}} W^e \\ W &= \sum_{\text{elements}} \langle \delta u_n \rangle ([k] \{u_n\} - \{f\}).\end{aligned}$$

We seek to put this expression in the form of (4.4):

$$W = \langle \delta U_n \rangle ([K] \{U_n\} - \{F\}).$$

In order to do so, we need only rewrite the elementary expressions W^e as a function of $\{U_n\}$ and $\langle \delta U_n \rangle$:

$$W^e = \langle \delta U_n \rangle ([K^e] \{U_n\} - \{F^e\}). \quad (4.24)$$

The matrix $[K^e]$ is constructed by expanding the matrix $[k]$ by inserting rows and columns of zeros. The dimensions of $[k]$ are equal to the number of degrees of freedom of the element; those of $[K^e]$ are equal to the total number of degrees of freedom.

Similarly, $\{F^e\}$ is constructed by inserting zeros into $\{f\}$. Let us now detail the operations of expansion of $[k]$ into $[K^e]$ and of $\{f\}$ into $\{F^e\}$:

a) Expansion of $[k]$

The expansion of $[k]$ takes place in two stages; one involves replacing $\{u_n\}$ with $\{U_n\}$ and the other consists of replacing $\langle \delta u_n \rangle$ with $\langle \delta U_n \rangle$. For instance, consider the expression

$$W^e = \langle \delta u_I \quad \delta u_J \rangle \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \begin{Bmatrix} u_I \\ u_J \end{Bmatrix} = \langle \delta u_n \rangle [k] \{u_n\}. \quad (4.25a)$$

The global vector of nodal variables is:

$$\langle U_n \rangle = \langle u_1 \quad u_2 \quad \dots \quad \underline{u_I} \quad u_{I+1} \quad \dots \quad \underline{u_J} \quad u_{J+1} \quad \dots \quad u_n \rangle.$$

The subscripts I and J denote the position in the global vector.

— Replacement of $\{u_n\}$ with $\{U_n\}$

In order for W^e to remain unchanged if we replace $\{u_n\}$ with $\{U_n\}$, we have to replace $[k]$, of dimensions (2×2) , with a matrix $[k']$, of dimensions $(2 \times n)$, whose column I is $\begin{Bmatrix} k_{11} \\ k_{21} \end{Bmatrix}$, column J is $\begin{Bmatrix} k_{12} \\ k_{22} \end{Bmatrix}$ and all other columns are null:

$$W^e = \langle \delta u_I \quad \delta u_J \rangle \begin{bmatrix} 0 & 0 & \dots & \begin{Bmatrix} k_{11} \\ k_{21} \end{Bmatrix} & 0 & \dots & \begin{Bmatrix} k_{12} \\ k_{22} \end{Bmatrix} & 0 & \dots & 0 \\ 0 & 0 & \dots & \begin{Bmatrix} k_{11} \\ k_{21} \end{Bmatrix} & 0 & \dots & \begin{Bmatrix} k_{12} \\ k_{22} \end{Bmatrix} & 0 & \dots & 0 \end{bmatrix}_{(2 \times n)} \begin{Bmatrix} u_1 \\ \vdots \\ u_I \\ u_{I+1} \\ \vdots \\ u_J \\ u_{J+1} \\ \vdots \\ u_n \end{Bmatrix} \quad (4.25b)$$

Note that if $I > J$, the columns of $[k]$ will be inverted in (4.25b).

— Replacement of $\langle \delta u_n \rangle$ with $\langle \delta U_n \rangle$

In order for W^e to remain unchanged if we replace $\langle \delta u_n \rangle$ with $\langle \delta U_n \rangle$, again we have to replace the new matrix $[k']$ with the matrix $[K^e]$ of dimensions $(n \times n)$, whose row I is the first row of $[k']$, row J is the second row of $[k']$ and all other rows are null:

$$\langle \delta U_n \rangle = \langle \delta u_1 \quad \delta u_2 \quad \dots \quad \underline{\delta u_I} \quad \delta u_{I+1} \quad \dots \quad \underline{\delta u_J} \quad \delta u_{J+1} \quad \dots \quad u_n \rangle.$$

$$\begin{aligned}
W^e = \langle \delta U_n \rangle & \begin{bmatrix} 0 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \dots & k_{11} & \dots & k_{12} & \dots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \dots & k_{21} & \dots & k_{22} & \dots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \dots & 0 & \dots & 0 & \dots & 0 \end{bmatrix} & \leftarrow \text{row } I \\
& \begin{array}{c} \uparrow \\ \text{column } I \end{array} \quad \begin{array}{c} \uparrow \\ \text{column } J \end{array} & \{U_n\} = \langle \delta U_n \rangle [K^e] \{U_n\} \\
& & \leftarrow \text{row } J \\
& & (n \times n) & & & & (4.25c)
\end{aligned}$$

b) Expansion of $\{f\}$

Consider the expression

$$W^e = \langle \delta u_I \ \delta u_J \rangle \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} = \langle \delta u_n \rangle \{f\}. \quad (4.26a)$$

In order for W^e to remain unchanged if we replace $\langle \delta u_n \rangle$ with $\langle \delta U_n \rangle$, we have to replace $\{f\}$ of dimension 2 with the vector $\{F^e\}$ of dimension n where the I th term is f_1 , the J th term is f_2 and all other terms are null:

$$W^e = \langle \delta U_n \rangle \begin{Bmatrix} 0 \\ \vdots \\ f_1 \\ \vdots \\ f_2 \\ \vdots \\ 0 \end{Bmatrix} = \langle \delta U_n \rangle \{F^e\}. \quad (4.26b)$$

EXAMPLE 4.16. Expansion of $[k]$ and $\{f\}$ from Example 4.15

In Example 4.15, the elementary form of element (1) is written as:

$$W^{(1)} = \langle \delta u_1 \quad \delta u_2 \quad \delta u_4 \rangle \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix}^{(1)} \begin{Bmatrix} u_1 \\ u_2 \\ u_4 \end{Bmatrix} - \begin{Bmatrix} f_1 \\ f_2 \\ f_4 \end{Bmatrix}^{(1)}$$

or by using the extended matrix $[K^{(1)}]$ and the extended vector $\{F^{(1)}\}$

$$W^{(1)} = \langle \delta u_1 \quad \delta u_2 \quad \delta u_3 \quad \delta u_4 \rangle \begin{pmatrix} k_{11} & k_{12} & 0 & k_{13} \\ k_{21} & k_{22} & 0 & k_{23} \\ 0 & 0 & 0 & 0 \\ k_{31} & k_{32} & 0 & k_{33} \end{pmatrix}^{(1)} \underbrace{\begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}}_{[K^{(1)}]} - \underbrace{\begin{Bmatrix} f_1 \\ f_2 \\ 0 \\ f_3 \end{Bmatrix}}_{\{F^{(1)}\}}^{(1)}.$$

In the case of element (2):

$$W^{(2)} = \langle \delta u_1 \quad \delta u_4 \quad \delta u_3 \rangle \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix}^{(2)} \begin{Bmatrix} u_1 \\ u_4 \\ u_3 \end{Bmatrix} - \begin{Bmatrix} f_1 \\ f_2 \\ f_3 \end{Bmatrix}^{(3)}.$$

In its extended form:

$$W^{(2)} = \langle \delta u_1 \quad \delta u_2 \quad \delta u_3 \quad \delta u_4 \rangle \begin{pmatrix} k_{11} & 0 & k_{13} & k_{12} \\ 0 & 0 & 0 & 0 \\ k_{31} & 0 & k_{33} & k_{32} \\ k_{21} & 0 & k_{23} & k_{22} \end{pmatrix}^{(2)} \underbrace{\begin{Bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{Bmatrix}}_{[K^{(2)}]} - \underbrace{\begin{Bmatrix} f_1 \\ 0 \\ f_3 \\ f_2 \end{Bmatrix}}_{\{F^{(2)}\}}^{(2)}.$$

Remark: The subscripts that appear in $[K^e]$ and $\{F^e\}$ represent the position of each term in $[k]$ and $\{f\}$. However, the subscripts in $\langle \delta u_n \rangle$ and $\{u_n\}$ represent the number of the nodal variable.

We obtain the global integral form for adding together the expressions (4.24), factoring $\langle \delta U_n \rangle$ and $\{U_n\}$:

$$\begin{aligned} W &= \sum_e W^e = \sum_e \langle \delta U_n \rangle ([K^e] \{U_n\} - \{F^e\}) \\ &= \langle \delta U_n \rangle \left(\left[\sum_e [K^e] \right] \{U_n\} - \left\{ \sum_e \{F^e\} \right\} \right) \\ &= \langle \delta U_n \rangle ([K] \{U_n\} - \{F\}) \end{aligned} \tag{4.27a}$$

where:

$$[K] = \sum_e [K^e]; \quad \{F\} = \sum_e \{F^e\}. \quad (4.27b)$$

The global matrix $[K]$ is therefore the sum of the extended elementary matrices $[K^e]$. The global vector $\{F\}$ is the sum of the extended elementary vectors $\{F^e\}$.

EXAMPLE 4.17. Global matrix from Example 4.16

The global matrix is:

$$[K] = [K^{(1)}] + [K^{(2)}]$$

$$[K] = \begin{bmatrix} k_{11}^{(1)} + k_{11}^{(2)} & k_{12}^{(1)} & k_{13}^{(2)} & k_{13}^{(1)} + k_{12}^{(2)} \\ k_{21}^{(1)} & k_{22}^{(1)} & 0 & k_{23}^{(1)} \\ k_{31}^{(2)} & 0 & k_{33}^{(2)} & k_{32}^{(2)} \\ k_{31}^{(1)} + k_{21}^{(2)} & k_{32}^{(1)} & k_{23}^{(2)} & k_{33}^{(1)} + k_{22}^{(2)} \end{bmatrix}.$$

The global load vector is:

$$\{F\} = \{F^{(1)}\} + \{F^{(2)}\} = \begin{Bmatrix} f_1^{(1)} + f_1^{(2)} \\ f_2^{(1)} \\ f_3^{(2)} \\ f_3^{(1)} + f_2^{(2)} \end{Bmatrix}.$$

4.4.2 ASSEMBLY IN STRUCTURAL MECHANICS

Historically, the notion of assembly was first used for problems in structural mechanics, in which the element is in fact a spring, a truss or a beam. For each element e , considered to be isolated, we have the relation linking the displacements $\{u_n\}$ and the applied forces:

$$[k]\{u_n\} - \{f\} = \{p\} \quad (4.28)$$

where: $\{f\}$ are the known external forces applied to the element, identical to those in (4.4);

$\{p\}$ are the internal forces due to the action of the other elements on the element e ;

$[k]$ is the elementary stiffness matrix.

Assembly consists of constructing the global system of equations:

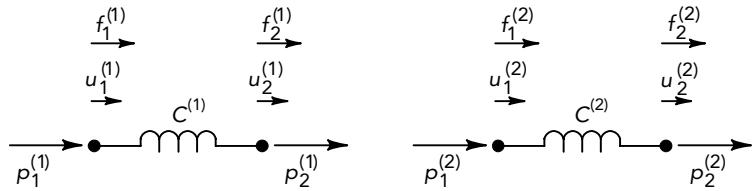
$$[K] \{U_n\} = \{F\}$$

using

- continuity of the displacements at the nodes;
- equilibrium of the forces which, at each node i , is expressed by $\sum_e p_i^e = 0$.

EXAMPLE 4.18. Assembly of two springs

Consider two springs of unit stiffness

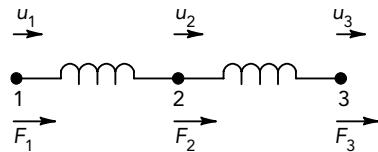


The equations (4.28) corresponding to each spring are:

$$\text{Spring 1: } \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} u_1^{(1)} \\ u_2^{(1)} \end{Bmatrix} - \begin{Bmatrix} f_1^{(1)} \\ f_2^{(1)} \end{Bmatrix} = \begin{Bmatrix} p_1^{(1)} \\ p_2^{(1)} \end{Bmatrix};$$

$$\text{Spring 2: } \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} u_1^{(2)} \\ u_2^{(2)} \end{Bmatrix} - \begin{Bmatrix} f_1^{(2)} \\ f_2^{(2)} \end{Bmatrix} = \begin{Bmatrix} p_1^{(2)} \\ p_2^{(2)} \end{Bmatrix}.$$

The assembled structure is as follows:



Continuity of the displacements implies:

$$\begin{aligned} u_1^{(1)} &= U_1 & u_1^{(2)} &= U_2 \\ u_2^{(1)} &= U_2 & u_2^{(2)} &= U_3. \end{aligned}$$

Equilibrium of the forces is written as, at the three nodes:

$$p_1^{(1)} = 0 \quad p_2^{(1)} + p_1^{(2)} = 0 \quad p_2^{(2)} = 0$$

which, following expansion of the elementary relations in a form comparable to (4.25c) and (4.26b), is written as:

$$\begin{bmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} - \begin{Bmatrix} f_1^{(1)} \\ f_2^{(1)} \\ 0 \end{Bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} - \begin{Bmatrix} 0 \\ f_1^{(2)} \\ f_2^{(2)} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}$$

$$[K^{(1)}] \{U_n\} - \{F^{(1)}\} + [K^{(2)}] \{U_n\} - \{F^{(2)}\} = 0$$

so:

$$\begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} = \begin{Bmatrix} f_1^{(1)} \\ f_2^{(1)} + f_1^{(2)} \\ f_2^{(2)} \end{Bmatrix}.$$

$$[K] \{U_n\} = \{F\}.$$

4.5 Technique of assembly

4.5.1 STAGES OF ASSEMBLY

We saw in section 4.4 that assembly comprises two stages:

- construction of the extended matrix $[K^e]$ and the extended vector $\{F^e\}$ of each element in accordance with (4.25c) and (4.26b);
- addition of the extended matrices and vectors (4.27b).

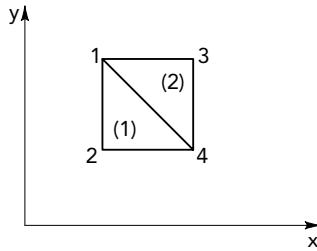
In practice, these two stages are carried out simultaneously so as to avoid explicitly constructing $[K^e]$ and $\{F^e\}$.

4.5.2 RULES OF ASSEMBLY

So as to standardize the operations of assembly, let us define, for each element, the **Elementary LOCalization (LOCE) table**, which gives the position of each term from $\{u_n\}$ in $\{U_n\}$ and therefore also the position of each term from $\langle \delta u_n \rangle$ in $\langle \delta U_n \rangle$. If there is only one degree of freedom per node, this table is identical to the connectivity (CONEC) table defined in section 1.2.6. The dimension of the LOCE table is equal to the number of degrees of freedom of the element n_{de} .

EXAMPLE 4.19. Definition of the elementary localization table

In the case of assembly of the following two triangles:



the connectivity (CONEC) table is:

elements	nodes		
1	1	2	4
2	1	4	3

a) If there is one degree of freedom per node u :

$$\langle U_n \rangle = \langle u_1 \quad u_2 \quad u_3 \quad u_4 \rangle$$

- element 1

$$\begin{aligned} \langle U_n \rangle &= \langle u_1 \quad u_2 \quad u_4 \rangle \\ \text{LOCE} &= \langle 1 \quad 2 \quad 4 \rangle \end{aligned}$$

- element 2

$$\begin{aligned} \langle U_n \rangle &= \langle u_1 \quad u_4 \quad u_3 \rangle \\ \text{LOCE} &= \langle 1 \quad 4 \quad 3 \rangle \end{aligned}$$

b) If there are two degrees of freedom per node u and v :

$$\langle U_n \rangle = \langle u_1 \quad v_1 \quad u_2 \quad v_2 \quad u_3 \quad v_3 \quad u_4 \quad v_4 \rangle$$

- element 1

$$\begin{aligned} \langle U_n \rangle &= \langle u_1 \quad v_1 \quad u_2 \quad v_2 \quad u_4 \quad v_4 \rangle \\ \text{LOCE} &= \langle 1 \quad 2 \quad 3 \quad 4 \quad 7 \quad 8 \rangle \end{aligned}$$

• *element 2*

$$\begin{aligned}\langle U_n \rangle &= \langle u_1 \quad v_1 \quad u_4 \quad v_4 \quad u_3 \quad v_3 \rangle \\ \text{LOCE} &= \langle 1 \quad 2 \quad 7 \quad 8 \quad 5 \quad 6 \rangle\end{aligned}$$

Let us discuss in greater detail the operation of expansion (4.25c) of an arbitrary elementary matrix $[k]$ into a matrix $[K^e]$ using the localization table LOCE: each term k_{ij} from $[k]$ is transferred into K_{IJ}^e from $[K^e]$ so that:

$$\begin{aligned}I &= \text{LOCE}(i) \quad i = 1, n_{de} \\ J &= \text{LOCE}(j) \quad j = 1, n_{de}\end{aligned}$$

or indeed:

$$K_{IJ}^e \equiv K_{\text{LOCE}(i), \text{LOCE}(j)}^e \equiv k_{ij}. \quad (4.29a)$$

Similarly, each term f_i from $\{f\}$ is transferred into F_I^e from $\{F^e\}$ so that:

$$F_I^e \equiv F_{\text{LOCE}(i)}^e \equiv f_i. \quad (4.29b)$$

The general algorithms that are used to carry out the two stages of the assembly are as follows:

- initialize the terms of $[K]$ and $\{F\}$ at zero;
- for each element e :
 - add each term k_{ij} from its elementary matrix to the term K_{IJ}^e from the global matrix:

$$\begin{aligned}K_{IJ} &= K_{IJ} + k_{ij} \quad i = 1, 2, \dots, n_{de} \\ &\quad j = 1, 2, \dots, n_{de}\end{aligned}$$

where:

$$I = \text{LOCE}(i)$$

$$J = \text{LOCE}(j).$$

- add each term f_i from the elementary load vector to the term F_I^e from the global vector:

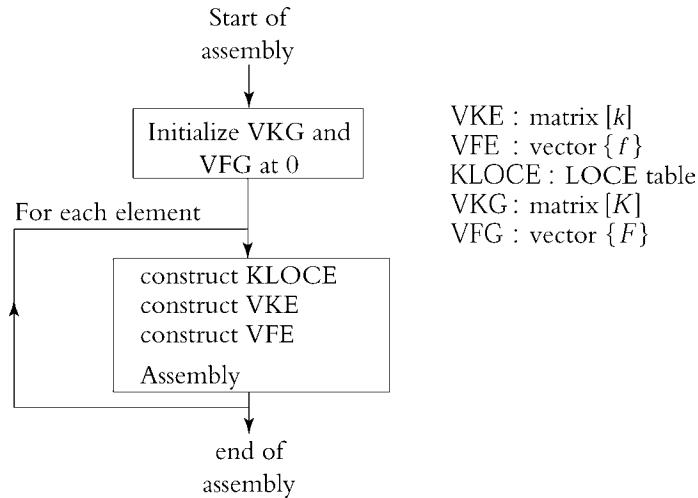
$$F_I = F_I + f_i \quad i = 1, 2, \dots, n_{de}$$

where:

$$I = \text{LOCE}(i).$$

4.5.3 EXAMPLE OF A SUBPROGRAM FOR ASSEMBLY

In Figure 4.6, we present a simple subprogram for assembly of the matrix $[k]$ and the vector $\{f\}$ of an element. This subprogram runs in the following manner:



The “Assembly” phase in the element loop is expressed by the lines of code (here written in Matlab[©]):

$$\begin{aligned}
 \text{VKG(KLOCE, KLOCE)} &= \text{VKG(KLOCE, KLOCE)} + \text{VKE} \\
 \text{VFG(KLOCE)} &= \text{VFG(KLOCE)} + \text{VFE}
 \end{aligned}$$

4.5.4 CONSTRUCTION OF THE LOCALIZATION TABLE LOCE

a) Case of one degree of freedom per node

The localization table LOCE is identical to the connectivity table CONEC which defines the element (see section 1.2.6).

b) Case of n_{dn} degrees of freedom at each node (u, v, \dots)

Suppose that the variables are organized in the form:

$$\begin{aligned}
 \{u_n\}^T &= \langle u_i \quad v_i \quad \dots; \quad u_j \quad v_j \quad \dots; \quad \dots \rangle \\
 \{U_n\}^T &= \langle u_1 \quad v_1 \quad \dots; \quad u_2 \quad v_2 \quad \dots; \quad u_3 \quad v_3 \quad \dots; \quad \dots; \quad u_n \quad v_n \rangle
 \end{aligned}$$

where: i, j, \dots are the numbers of the n_e nodes of the element e ;
 n is the total number of nodes.

The total number of degrees of freedom of an element is:

$$n_{de} = n_e \times n_{dn} .$$

The KLOCE table is constructed for each element based on the CONEC table using the algorithm given in Figure 4.3, written in Matlab[©]:

```

j=0;    % Initialization of a counter to zero
%Loop on the 'nnel' nodes of the element
for in=1: nnel
    ido=(kconec(in)-1)*ndln;
    %Loop on the 'ndln' unknowns of the node 'in'
    for id=1: ndln
        j=j+1;
        kloce(j)=id+ido;
    end
end

```

Figure 4.3. Construction of the KLOCE: constant number of DoF per node

c) Case of a variable number of degrees of freedom per node

The number of degrees of freedom of each node must be stored in a table, DLNC. In practice, for reasons of efficiency, this table is organized in a “cumulative” way: DLNC ($I + 1$) represents the sum of the numbers of degrees of freedom of the nodes 1, 2, ..., $I - 1$, I . The dimensions of the DLNC table are $n + 1$. The number of degrees of freedom of the node I is, therefore:

$$\text{DLNC}(I + 1) - \text{DLNC}(I).$$

The KLOCE table is constructed, for each element, using the algorithm in Figure 4.4:

```

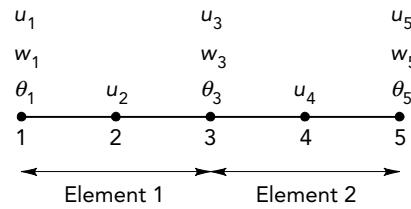
j=0;    % Initialization of a counter to zero
%Loop on the 'nnel' nodes of the element
for in=1: nnel
    ii=kconec(in);
    ido=kdlnc(in);           % number of DoF associated with the node 'in'
    idln=kdlnc(ii+1)-ido;
    %Loop on the 'idln' unknowns of the node 'in'
    for id=1: idln
        j=j+1;
        kloce(j)=id+ido;
    end
end

```

Figure 4.4. Construction of the KLOCE: variable number of DoF per node

EXAMPLE 4.20. Localization table for an assembly with a variable number of degrees of freedom

Consider the assembly of two elements with seven degrees of freedom and three nodes:



total number of nodes:	$n = 5$
number of elements:	$n_{el} = 2$
number of nodes per element:	$n_e = 3$
number of degrees of freedom per element:	$n_{de} = 7$
total number of degrees of freedom:	$n_d = 11.$

Connectivity table (CONEC)

elements	nodes		
1	1	2	3
2	3	4	5

Table of numbers of degrees of freedom per cumulative node (DLNC):

$$\langle 0 \ 3 \ 4 \ 7 \ 8 \ 11 \rangle$$

Localization table (LOCE):

- element 1 $\langle 1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \rangle$
- element 2 $\langle 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \rangle.$

4.6 Properties of global matrices

4.6.1 BAND STRUCTURE

The global matrix $[K]$ is constructed by addition of the very sparse expanded elementary matrices $[K^e]$:

$$[K] = \sum_{\text{elements}} [K^e].$$

In accordance with the assembly rule from section 4.5, the only non-null terms in $[K^e]$ are such that:

$$K_{IJ}^e \equiv k_{ij}^e \quad (4.30)$$

where:

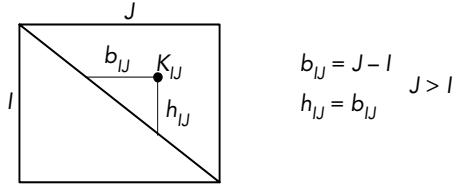
$$I = \text{LOCE}(i) \quad i = 1, 2, \dots, n_{de}$$

$$J = \text{LOCE}(j) \quad j = 1, 2, \dots, n_{de}.$$

Consequently, a term K_{IJ} is different from zero if and only if there is an element that simultaneously involves the nodal variables u_I and u_J .

The assemble rule is symmetrical in I and J ; if there is a non-null term K_{IJ} ..., then there is also a non-null term K_{JI} . Thus we need only to study the structure (the topology) of the upper half of $[K]$, for which $J > I$. Note that this does not imply that the $[K]$ is symmetrical ($K_{ij} = K_{ji}$).

Let us define the **horizontal distance** b_{IJ} and the **vertical distance** h_{IJ} of a non-null term K_{IJ} in relation to the diagonal in $[K]$:



$$\begin{aligned} b_{IJ} &= J - I & J > I \\ h_{IJ} &= b_{IJ} \end{aligned}$$

In view of equation (4.30), b_{IJ} corresponding to the term k_{ij} of the element e is written as:

$$b_{IJ}^e = J - I = \text{LOCE}(j) - \text{LOCE}(i) \quad J > I.$$

The **elementary bandwidth** b_i^e of the row I in $[K^e]$ is $\text{Max}(b_{IJ}^e)$ for all the **non-zero** terms on that row I :

$$\boxed{\begin{array}{l} i = 1, 2, \dots, n_{de} \\ | \\ I = \text{LOCE}(i) \\ | \\ j = 1, 2, \dots, n_{de} \\ | \\ b_i^e = \max_j (\text{LOCE}(j) - I) = \max_j (\text{LOCE}(j)) - I. \end{array}} \quad (4.31a)$$

Similarly, the **elementary band height** h_j^e of the column J in $[K^e]$ is $\text{Max}(h_{IJ}^e)$ for all the **non-zero** terms in that column:

$$\boxed{\begin{array}{l} j = 1, 2, \dots, n_{de} \\ | \\ J = \text{LOCE}(j) \\ | \\ i = 1, 2, \dots, n_{de} \\ | \\ h_j^e = \max_i (J - \text{LOCE}(i)) = J - \min_i (\text{LOCE}(i)). \end{array}} \quad (4.31b)$$

The bandwidth b_I of the row I of the global matrix $[K]$ is:

$$b_I = \text{Max}_e (b_I^e) \quad (4.32a)$$

for all elements e . On the row I , the terms K_{Ij} are null for $J > b_I + 1$. The band height h_J of the column J in the global matrix $[K]$ is:

$$h_J = \text{Max}_e (h_J^e) \quad (4.32b)$$

for all elements e . Finally, the bandwidth b and the band height h of the global matrix $[K]$ is:

$$\begin{aligned} b &= \text{Max}_I (b_I) && \text{for all rows } I \\ h &= \text{Max}_J (h_J) && \text{for all columns } J. \end{aligned} \quad (4.33)$$

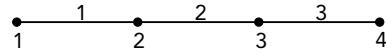
Because of the symmetry in I and J ,

$$b = h.$$

Let us stress that the bandwidths and band heights defined above do not take account of the diagonal term. Thus, for a diagonal matrix, $b = h = 0$.

EXAMPLE 4.21. Bandwidths and band heights for an assembly of three one-dimensional elements

Consider the three two-node elements, with one degree of freedom per node:



- Element 1 LOCE = $\langle 1 \ 2 \rangle$

$$b_I^{(1)} = \langle 1 \ 0 \ 0 \ 0 \rangle$$

$$h_J^{(1)} = \langle 0 \ 1 \ 0 \ 0 \rangle.$$

- Element 2 LOCE = $\langle 2 \ 3 \rangle$

$$b_I^{(1)} = \langle 0 \ 1 \ 0 \ 0 \rangle$$

$$h_J^{(1)} = \langle 0 \ 0 \ 1 \ 0 \rangle.$$

- Element 3 LOCE = $\langle 3 \ 4 \rangle$

$$b_I^{(1)} = \langle 0 \ 0 \ 1 \ 0 \rangle$$

$$h_J^{(1)} = \langle 0 \ 0 \ 0 \ 1 \rangle.$$

- For the assembled matrix:

$$b_I = \langle 1 \ 1 \ 1 \ 0 \rangle$$

$$h_J = \langle 0 \ 1 \ 1 \ 1 \rangle$$

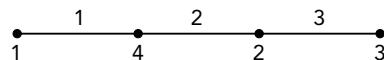
$$b = h = 1$$

$$[K] = \begin{bmatrix} x & x & 0 & 0 \\ x & x & x & 0 \\ 0 & x & x & x \\ 0 & 0 & x & x \end{bmatrix}.$$

The band structure of the matrix $[K]$ is an important characteristic of the finite element method. It enables savings to be made both in terms of storage of the matrix and of the resolution operations of the final system of equations. The bandwidth b_I of each row of $[K]$ depends on the LOCE table of each element, and therefore on the connectivity table of the elements and consequently on the order of numbering of the nodes. Although the number of non-null terms in $[K]$ remains constant, the bandwidth can change significantly with the order of numbering of the nodes.

In the case of the so-called *sparse* storage, only the non-zero terms in the matrix are stored, which greatly reduces the memory space to be allocated to the storage of the matrix. Conversely, this technique necessitates the construction of additional vectors for localization of the non-zero terms and the rewriting of the main matrix computation algorithms such as the matrix/vector product, for instance.

EXAMPLE 4.22. Renumbering of the nodes from Example 4.21



$$[K] = \begin{bmatrix} x & 0 & 0 & x \\ 0 & x & x & x \\ 0 & x & x & 0 \\ x & x & 0 & x \end{bmatrix}$$

$$b = h = 3.$$

The matrix $[K]$ contains the same number of non-zero terms as in Example 4.21; however, the bandwidth has increased from one to three.

Practical rule

To minimize the bandwidth, we have to minimize the difference of the numbers of nodes belonging to the same element.

4.6.2 SYMMETRY

In numerous problems (\mathcal{L} self-adjoint), the matrices $[k]$ are symmetrical; hence so too is the matrix $[K]$:

$$K_{ij} = K_{ji}.$$

This property also facilitates significant savings in terms of storage and resolution of the system of equations.

4.6.3 STORAGE METHODS

a) Full non-symmetrical matrix

A full non-symmetrical matrix, of dimensions $(n \times n)$, occupies n^2 actual numbers (generally double precision requiring $n \times n \times 8$ bytes) in computer memory.

b) Full symmetrical matrix

In this case, it is sufficient to store the upper triangle of the matrix in a VK , e.g. by descending columns:

$$[K] = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ K_{12} & K_{22} & K_{23} \\ K_{13} & K_{23} & K_{33} \end{bmatrix}; \quad VK = \langle K_{11} \ K_{12} \ K_{22} \ K_{13} \ K_{23} \ K_{33} \rangle$$

$$K_{ij} \equiv VK_I \quad \text{if } I = \frac{J(J-1)}{2} + i, \quad J \geq I. \quad (4.34)$$

$\frac{n(n+1)}{2}$ real words have to be stored.

c) Non-symmetrical matrix band

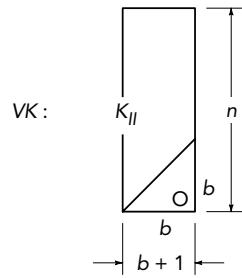
We store the matrix in a rectangular table VK of dimensions $n(2b+1)$:

$$[K] = \quad VK: \quad K_{ij} = VK_{ij} \quad \text{if} \quad \begin{cases} i = I \\ j = J - I + 1 + b. \end{cases} \quad (4.35)$$

We have to store $n(2b+1)$ real words, including $b(b+1)$ useless null values.

d) Symmetrical matrix band

In this case:



$$K_{ij} = VK_{ij} \quad \text{if } \begin{cases} i = I \\ j = J - I + 1 \\ J \geq I \end{cases} \quad (4.36)$$

We have to store $n(b+1)$ real words, including $b(b+1)/2$ useless null values.

e) Non-symmetrical “skyline” matrix

The “skyline” storage method consists of storing the terms of $[K]$ by rows and columns of variable lengths. We choose to use three storage tables:

VKGD contains the diagonal terms;

VKGS contains the terms of the upper triangle of $[K]$, organized by descending columns (without the diagonal terms);

VKGI contains the terms of the lower triangle of $[K]$, organized by rows from left to right (without the diagonal terms).

For the matrix:

$$[K] = \begin{bmatrix} K_{11} & K_{12} & 0 & K_{14} & 0 \\ K_{21} & K_{22} & K_{23} & K_{24} & 0 \\ 0 & K_{32} & K_{33} & K_{34} & K_{35} \\ K_{41} & K_{42} & K_{43} & K_{44} & 0 \\ 0 & 0 & K_{53} & 0 & K_{55} \end{bmatrix} \quad \text{“skyline”} \quad (4.37)$$

$$\begin{aligned}
 [K] = & \left[\begin{array}{ccccc}
 0 & K_{12} & 0 & K_{14} & 0 \\
 0 & 0 & K_{23} & K_{24} & 0 \\
 0 & 0 & 0 & K_{34} & K_{35} \\
 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0
 \end{array} \right] + \left[\begin{array}{ccccc}
 K_{11} & 0 & 0 & 0 & 0 \\
 0 & K_{22} & 0 & 0 & 0 \\
 0 & 0 & K_{33} & 0 & 0 \\
 0 & 0 & 0 & K_{44} & 0 \\
 0 & 0 & 0 & 0 & K_{55}
 \end{array} \right] + \\
 & \text{Terms placed in VKGS} \quad \text{Terms placed in VKGD} \\
 & + \left[\begin{array}{ccccc}
 0 & 0 & 0 & 0 & 0 \\
 K_{21} & 0 & 0 & 0 & 0 \\
 0 & K_{32} & 0 & 0 & 0 \\
 K_{41} & K_{42} & K_{43} & 0 & 0 \\
 0 & 0 & K_{53} & 0 & 0
 \end{array} \right] \\
 & \text{Terms placed in VKGI} \quad (4.38)
 \end{aligned}$$

$$\begin{aligned}
 \text{VKGS} &= \langle K_{12}; K_{23}; K_{14} \ K_{24} \ K_{34}; K_{35} \ 0 \rangle \\
 \text{VKGI} &= \langle K_{21}; K_{32}; K_{41} \ K_{42} \ K_{43}; K_{53} \ 0 \rangle \quad (4.39) \\
 \text{VKGD} &= \langle K_{11} \ K_{22} \ K_{33} \ K_{44} \ K_{55} \rangle.
 \end{aligned}$$

The skyline is the envelope of the corners of the columns of varying heights. It is symmetrical in relation to the diagonal, to the envelope of the left-hand extremities of the rows, whether or not $[K]$ is symmetrical. It is defined by the table of column heights h_J (4.32b); for the matrix (4.37):

$$h_J = \langle 0 \ 1 \ 1 \ 3 \ 2 \rangle. \quad (4.40)$$

The null terms in $[K]$ that fall outside the two envelopes are not stored; the null terms within the envelopes are stored, as are the terms in position (4,5) and (5,4) in the matrix (4.37).

In order to define the position of a term K_{ij} in the VKGS and VKGI tables, we use the table of “Localization of the Beginnings of the Columns” (KLD), of dimension $n+1$, defined by:

$$\begin{aligned}
 \text{KLD}(1) &= 1; \text{KLD}(2) = 1 \\
 \text{KLD}(I) &= \text{KLD}(I-1) + h_J(I-1) \quad I = 3, 4, \dots, n+1.
 \end{aligned} \quad (4.41)$$

In the case of (4.40):

$$\text{KLD} = \langle 1 \ 1 \ 2 \ 3 \ 6 \ 8 \rangle.$$

Thus, a term K_{ij} is placed:

- If $I = J$ in VKGD (I)
 - If $I < J$ in VKGS (l) where $l = \text{KLD}(J+1) - J + I$
 - If $I > J$ in VKGI (l) where $l = \text{KLD}(I+1) - I + J$.
- (4.42)

The storage space needed is:

- for VKGD: n real words
- for VKGS/VKGI: $\text{KLD}(n+1) - 1$ real words

and therefore in total:

$$n + 2(\text{KLD}(n+1) - 1) \text{ real words.} \quad (4.43)$$

EXAMPLE 4.23. Skyline storage of a non-symmetrical matrix

The matrix $[K]$ from Example 4.22 is:

$$[K] = \begin{bmatrix} K_{11} & 0 & 0 & K_{14} \\ 0 & K_{22} & K_{23} & K_{24} \\ 0 & K_{32} & K_{33} & 0 \\ K_{41} & K_{42} & 0 & K_{44} \end{bmatrix} \quad n = 4.$$

In this case:

$$\begin{aligned} h_J &= \langle 0 \ 0 \ 1 \ 3 \rangle \\ \text{KLD} &= \langle 1 \ 1 \ 1 \ 2 \ 5 \rangle \\ \text{VKGS} &= \langle K_{23} \ K_{14} \ K_{24} \ 0 \rangle \\ \text{VKGI} &= \langle K_{32} \ K_{41} \ K_{42} \ 0 \rangle \\ \text{VKGD} &= \langle K_{11} \ K_{22} \ K_{33} \ K_{44} \rangle. \end{aligned}$$

In accordance with equation (4.42), the term K_{24} is in:

$$\text{VKGS } (l) \text{ where } l = \text{KLD}(4+1) - 4 + 2 = 3.$$

The space required is:

$$4 + 2(\text{KLD}(5) - 1) = 12 \text{ real words.}$$

f) Symmetrical skyline matrix

The storage is identical to that of a non-symmetrical matrix for the diagonal and the upper triangle. The VKGI table is not used.

The storage space needed is:

$$n + \text{KLD}(n+1) - 1 \text{ real words.}$$

Note that for a diagonal matrix, it is sufficient simply not to use the VKGS table. This is often the case for mass matrices.

4.7 Global system of equations

4.7.1 EXPRESSION OF THE SYSTEM OF EQUATIONS

Following assembly, the global integral form is written as (4.5b):

$$W = \langle \delta U_n \rangle ([K] \{U_n\} - \{F\}) = 0.$$

The problem lies in finding $\{U_n\}$ which nullifies W for any value of $\langle \delta U_n \rangle$ while satisfying the boundary conditions on S_u defined in section 3.3.2 and which imposes constraints on $\{U_n\}$ and $\langle \delta U_n \rangle$. In discretized form, these conditions are written as:

$$\delta U_i = 0 \quad (4.44a)$$

$$U_i = \bar{U}_i \quad (4.44b)$$

for all the imposed degrees of freedom U_i whose value is \bar{U}_i .

Thus, the algebraic system:

$$[K] \{U_n\} = \{F\} \quad (4.45)$$

must be solved in $\{U_n\}$ after modification of the matrix $[K]$ and the vector $\{F\}$ to take account of the conditions (4.44).

4.7.2 INTRODUCTION OF THE BOUNDARY CONDITIONS

The conditions (4.44) can be introduced into system (4.45) in a number of different ways:

a) Dominant diagonal method

The matrix $[K]$ is assembled without taking account of the boundary conditions; then each relation $U_i = \bar{U}_i$ is introduced by replacing:

- K_{ii} with $K_{ii} + \alpha$, where α is a number that is very large in relation to all the terms K_{ij} ;
- F_i with $\alpha \bar{U}_i$.

$$\begin{bmatrix} K_{11} & \dots & K_{1i} & \dots & K_{1n} \\ \vdots & & \vdots & & \vdots \\ K_{i1} & \dots & K_{ii} + \alpha & \dots & K_{in} \\ \vdots & & \vdots & & \vdots \\ K_{n1} & \dots & K_{ni} & \dots & K_{nn} \end{bmatrix} \begin{Bmatrix} U_1 \\ U_i \\ U_n \end{Bmatrix} = \begin{Bmatrix} F_1 \\ \alpha \bar{U}_i \\ F_n \end{Bmatrix} \quad (4.46)$$

The equation i is written as:

$$\alpha U_i + \left(\sum_{j=1}^n K_{ij} U_j \right) = \alpha \bar{U}_i. \quad (4.47)$$

It admits the approximate solution:

$$U_i \approx \bar{U}_i \quad \text{if } \alpha \bar{U}_i \gg \sum_{j=1}^n K_{ij} U_j.$$

In practice, in programs, we can choose $\alpha = 10^7 \cdot \text{Max } |K_{ij}|$ or $10^{15} \cdot \text{Max } |K_{ij}|$ depending on whether the computer's degree of precision is to seven or 15 decimal places. This leads to an error on U_i , which is of the same order as the precision of the computer.

This method is very simple to implement because we only need to change the two terms K_{ii} and F_i , but it may cause problems when the matrix $[K]$ is inaccurately conditioned and when certain components of $\{U_n\}$ are large.

b) Method using the unit term on the diagonal

This method involves modifying the vector $\{F\}$ and then the matrix $[K]$ for each relation $U_i = \bar{U}_i$:

$$F_j = F_j - K_{ji} \bar{U}_i \quad j = 1, 2, \dots, n \quad j \neq i$$

$$F_i = \bar{U}_i$$

$$K_{ij} = K_{ji} = 0 \quad j = 1, 2, \dots, n \quad j \neq i$$

$$K_{ii} = 1$$

$$\begin{bmatrix} K_{11} & \dots & K_{1,i-1} & 0 & K_{1,i+1} & \dots & K_{1n} \\ \vdots & & \vdots & & \vdots & & \vdots \\ K_{i-1,1} & \dots & K_{i-1,i-1} & 0 & K_{i-1,i+1} & \dots & K_{i-1,n} \\ 0 & \dots & 0 & 1 & 0 & \dots & 0 \\ K_{i+1,1} & \dots & K_{i+1,i-1} & 0 & K_{i+1,i+1} & \dots & K_{i+1,n} \\ \vdots & & \vdots & & \vdots & & \vdots \\ K_{n1} & \dots & K_{n,i-1} & 0 & K_{n,i+1} & \dots & K_{nn} \end{bmatrix} \begin{Bmatrix} U_1 \\ U_{i-1} \\ U_i \\ U_{i+1} \\ U_n \end{Bmatrix} = \begin{Bmatrix} F_1 - K_{1i} \bar{U}_i \\ F_{i-1} - K_{i-1,i} \bar{U}_i \\ \bar{U}_i \\ F_{i+1} - K_{i+1,i} \bar{U}_i \\ \vdots \\ F_n - K_{ni} \bar{U}_i \end{Bmatrix} \quad (4.48)$$

This method does not pose the same numerical problems; however, it is more complex to program.

c) Method of elimination of equations

This method consists of restructuring the matrix $[K]$ so as to eliminate the equations corresponding to the imposed degrees of freedom U_i . It offers the advantage of reducing the number of unknowns of the system.

The restructuring of $[K]$ and $\{F\}$ corresponding to $U_i = \bar{U}_i$ gives us equation (4.48) in which the row i and the column i have vanished.

In industrial codes, the assembly process is modified to prevent the construction of such relations altogether, thus avoiding costly global matrix manipulations.

EXAMPLE 4.24. Introduction of boundary conditions

The system of equations corresponding to Example 4.22 is written as:

$$\begin{bmatrix} K_{11} & 0 & 0 & K_{14} \\ 0 & K_{22} & K_{23} & K_{24} \\ 0 & K_{23} & K_{33} & 0 \\ K_{14} & K_{24} & 0 & K_{44} \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix}.$$

The condition $U_1 = \bar{U}_1$ gives us the following three modified forms of $[K]$ and $\{F\}$:

— dominant diagonal term:

$$\begin{bmatrix} K_{11} + 10^{15} & 0 & 0 & K_{14} \\ 0 & K_{22} & K_{23} & K_{24} \\ 0 & K_{23} & K_{33} & 0 \\ K_{14} & K_{24} & 0 & K_{44} \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{Bmatrix} = \begin{Bmatrix} 10^{15} \cdot \bar{U}_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix}$$

— unit diagonal term:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & K_{22} & K_{23} & K_{24} \\ 0 & K_{23} & K_{33} & 0 \\ 0 & K_{24} & 0 & K_{44} \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{Bmatrix} = \begin{Bmatrix} \bar{U}_1 \\ F_2 \\ F_3 \\ F_4 - K_{14} \bar{U}_1 \end{Bmatrix}$$

— deletion of equation 1:

$$\begin{bmatrix} K_{22} & K_{23} & K_{24} \\ K_{23} & K_{33} & 0 \\ K_{24} & 0 & K_{44} \end{bmatrix} \begin{Bmatrix} U_2 \\ U_3 \\ U_4 \end{Bmatrix} = \begin{Bmatrix} F_2 \\ F_3 \\ F_4 - K_{14} \bar{U}_1 \end{Bmatrix}; \quad U_1 = \bar{U}_1.$$

4.7.3 REACTIONS

When we impose the value of a degree of freedom U_1 , the right-hand side in the equation, F_i , becomes an unknown variable called **reaction** in mechanics. This reaction is calculated following the solution of the system, by the equation:

$$F_i = \sum_{j=1}^n K_{ij} U_j. \quad (4.49)$$

Another way of introducing the boundary conditions is to include the reactions F_i in the list of unknowns. Equation (4.48) becomes:

$$\begin{bmatrix} K_{11} & \dots & K_{1,i-1} & 0 & K_{1,i+1} & \dots & K_{1n} \\ \vdots & & \vdots & & \vdots & & \vdots \\ K_{i-1,1} & \dots & K_{i-1,i-1} & 0 & K_{i-1,i+1} & \dots & K_{i-1,n} \\ K_{i1} & \dots & K_{i,i-1} & -1 & K_{i,i+1} & \dots & K_{in} \\ K_{i+1,1} & \dots & K_{i+1,i-1} & 0 & K_{i+1,i+1} & \dots & K_{i+1,n} \\ \vdots & & \vdots & & \vdots & & \vdots \\ K_{n1} & \dots & K_{n,i-1} & 0 & K_{n,i+1} & \dots & K_{nn} \end{bmatrix} \begin{Bmatrix} U_1 \\ U_{i-1} \\ F_i \\ U_{i+1} \\ \vdots \\ U_n \end{Bmatrix} = \begin{Bmatrix} F_1 - K_{1i} \bar{U}_i \\ \vdots \\ F_{i-1} - K_{i-1,i} \bar{U}_i \\ -K_{ii} \bar{U}_i \\ F_{i+1} - K_{i+1,i} \bar{U}_i \\ \vdots \\ F_n - K_{ni} \bar{U}_i \end{Bmatrix} \quad (4.50)$$

We could solve this system directly to obtain both $\{U_n\}$ and the reactions. However, it should be noted that the matrix (4.50) is not symmetrical, even if the unmodified $[K]$ is.

4.7.4 TRANSFORMATION OF VARIABLES

Suppose it is necessary to transform the variables $\{\delta U_n\}$ and $\{U_n\}$ as follows:

$$\{\delta U_n\} = [R] \{\delta U'_n\} \quad \{U_n\} = [R] \{\delta U'_n\} \quad (4.51)$$

where $[R]$ is an arbitrary transformation matrix, which may be triangular. Let us write (4.51) in its integral form (4.5b):

$$W = \langle \delta U'_n \rangle ([K'] \{U'_n\} - \{F'\}) = 0$$

where: $[K'] = [R]^T [K] [R]$ and $\{F'\} = [R]^T \{F\}$. (4.52)

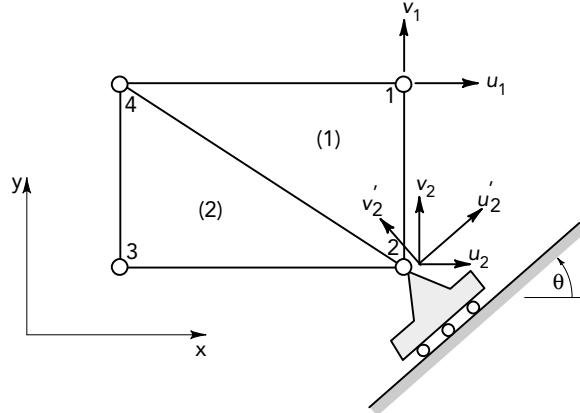
Such a transformation can be used:

- to change the framework of the nodal variables;
- to express a nodal variable as a function of other variables, or more generally to introduce linear relations between variables.

The transformation (4.52) of the global variables can also be performed **at the elementary level**; in particular, this enables us to use a local framework that simplifies the construction of $[k]$ and $\{f\}$.

EXAMPLE 4.25. Rotation of the variables for two-dimensional solid elements

Consider an assembly of two elements whose unknown nodal values are displacement components u, v . A change in reference system is needed at node 2 to impose the condition that this node slides along a sloping plane:



The global system of equations is written as:

$$\langle U_n \rangle = \langle u_1 \ v_1 \ u_2 \ v_2 \ u_3 \ v_3 \ u_4 \ v_4 \rangle$$

$$[K] \quad \langle U_n \rangle = \{F\}$$

$$(8 \times 8) \quad (8 \times 1)$$

The condition imposed on node 2 is written: $v'_2 = 0$, where v'_2 is the component of displacement perpendicular to the plane of motion.

$$\begin{Bmatrix} u_2 \\ v_2 \end{Bmatrix} = \begin{bmatrix} c & -s \\ s & c \end{bmatrix} \begin{Bmatrix} u'_2 \\ v'_2 \end{Bmatrix} \quad c = \cos \theta \quad s = \sin \theta$$

The matrix of transformation of all the variables is written as:

$$[R] = \begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & c & -s & & & & \\ & & s & c & & & & \\ & & & & 1 & & & \\ & & & & & 1 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{bmatrix}$$

$$\langle U'_n \rangle = \langle u_1 \ v_1 \ u'_2 \ v'_2 \ u_3 \ v_3 \ u_4 \ v_4 \rangle.$$

The matrices $[K']$ and $\{F'\}$ are obtained by (4.52). We then need only solve $[K']\{U'_n\} = \{F'\}$ with the condition $v'_2 = 0$.

4.7.5 LINEAR RELATIONS BETWEEN VARIABLES

The transformation of variables (4.51) enables us to introduce a linear relation between several variables:

$$a_i U_i + a_j U_j + a_k U_k + \dots = U'_i = g. \quad (4.53)$$

We transform the old variables $U_i, U_j, U_k \dots$ in $U'_i, U_j, U_k \dots$ by the matrix:

$$[R]_{(n \times n)} = \begin{bmatrix} & j & i & k & \\ 1 & & & & \\ & 1 & \ddots & 1 & \\ & & & \ddots & \\ \dots & -\frac{a_j}{a_i} & \dots & \frac{1}{a_i} & \dots & -\frac{a_k}{a_i} \\ & & & & \ddots & \\ & & & & 1 & \ddots & 1 \end{bmatrix} \quad (4.54)$$

using the equations (4.52), and then impose the condition $U'_i = g$. The matrix $[K]$ is then modified in accordance with:

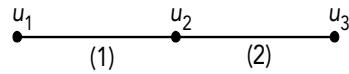
$$[K'] = [R]^T [K''] = [R]^T [K] [R], \quad (4.55)$$

and the vector $\{F\}$:

$$\{F'\} = [R]^T \{F\}. \quad (4.56)$$

EXAMPLE 4.26. Linear relations between variables of a one-dimensional problem

Consider the following assembly of two elements:



The global system of equations is written as:

$$\begin{bmatrix} K_{11} & K_{12} & 0 \\ K_{21} & K_{22} & K_{23} \\ 0 & K_{32} & K_{33} \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \end{Bmatrix}.$$

In order to impose $U_1 - U_3 = U'_3$, we will use the transformation (4.54) whereby $a_i = -1, a_j = 1, a_k = 0$:

$$\begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} = [R] \begin{Bmatrix} U_1 \\ U_2 \\ U'_3 \end{Bmatrix}; \quad [R] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & -1 \end{bmatrix}.$$

Then, in accordance with (4.55) and (4.56):

$$[K'] = \begin{bmatrix} K_{11} + K_{33} & K_{12} + K_{32} & -K_{33} \\ K_{21} + K_{23} & K_{22} & -K_{23} \\ -K_{33} & -K_{32} & K_{33} \end{bmatrix}$$

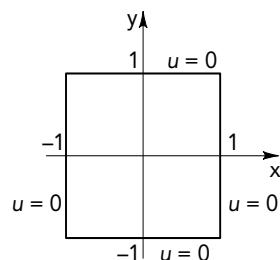
$$\{F'\} = \begin{Bmatrix} F_1 + F_3 \\ F_2 \\ -F_3 \end{Bmatrix}.$$

4.8 Example of application: Poisson's equation

Consider the problem already studied in Example 3.17: let us solve the following equation using the finite element method:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f_v = 0$$

This equation is defined on a square whose sides measure two, and associated with the boundary conditions: $u = 0$ on all four sides of the square.



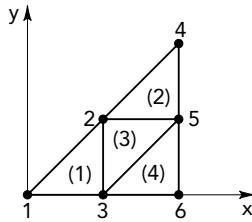
These equations may correspond to a thermal equilibrium problem, or a membrane equilibrium problem. The elementary integral form was obtained in Example 4.1, and can be written:

$$\begin{aligned}
 W^e &= \int_{V^e} \langle \delta(\partial u) \rangle \{ \partial u \} dV - \int_{V^e} \delta u f_V dV \\
 \langle \delta(\partial u) \rangle &= \left\langle \delta \left(\frac{\partial u}{\partial x} \right) \quad \delta \left(\frac{\partial u}{\partial y} \right) \right\rangle \\
 \langle \partial u \rangle &= \left\langle \frac{\partial u}{\partial x} \quad \frac{\partial u}{\partial y} \right\rangle.
 \end{aligned}$$

To get a solution using the finite element method, we perform the following operations:

a) Choice of mesh

We use the three-node triangular element described in section 4.3.1. In view of the symmetries, we choose the following mesh, which corresponds to one eighth of the domain:



The table of coordinates is:

	Nodes	1	2	3	4	5	6
VCOR	x	0	0,5	0,5	1,0	1,0	1,0
	y	0	0,5	0	1,0	0,5	0

The connectivity table is:

CONEC	Elements	1	2	3	4
	node 1	3	5	2	6
	node 2	2	4	3	5
	node 3	1	2	5	3

b) Dirichlet boundary conditions

$$U_4 = U_5 = U_6 = 0.$$

c) Neumann boundary conditions

The axes of symmetry respectively linking nodes 1–6 and nodes 1–4 are associated with the Neumann condition $\frac{\partial u}{\partial n} = 0$. This type of condition makes a contribution to the load vector for cases of a non-null value to be imposed.

d) Loads or sources

We can assume $f_V = \text{constant}$ over the whole domain, which corresponds, for instance:

- for a membrane equilibrium problem, to a distributed transverse pressure of constant value;
- for a thermal equilibrium problem, to a distributed heat source of constant value.

e) Elementary matrices and vectors

All the matrices $[k]$ are identical to the matrix (4.19) where $d = 1$, because we chose the first node of each triangle at the right-angled corner, like in section 4.3.1.

$$[k] = \frac{1}{2} \begin{bmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \quad \{f\} = \frac{A \cdot f_V}{3} \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix}, \quad \text{where } A = 1/8.$$

f) Assembly

Assembly of the four matrices $[k]$ and the use of the boundary conditions give us the final system:

$$\frac{1}{2} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 4 & -2 \\ -1 & -2 & 4 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} = \frac{f_V}{24} \begin{Bmatrix} 1 \\ 3 \\ 3 \end{Bmatrix}.$$

After resolution:

$$U_1 = 0,3125 f_V; \quad U_2 = 0,1771 f_V; \quad U_3 = 0,2292 f_V.$$

Consider the scenario of $f_v = 0$ and a concentrated force at node 1, whose value is f_C .

- for a membrane equilibrium problem, this corresponds to a force concentrated at node 1;
- for a heat equilibrium problem, it corresponds to a heat source concentrated at node 2.

The matrix of the system remains unchanged; however, the right-hand side vector becomes:

$$\{F\} = \begin{Bmatrix} f_C \\ 0 \\ 0 \end{Bmatrix}.$$

The corresponding solution is:

$$U_1 = 3,0 f_c; \quad U_2 = 0,5 f_c; \quad U_3 = 1,0 f_c.$$

Programs in Matlab[®] are given in Figures 4.5 and 4.6: the first prepares the geometric data of the problem (data1t3.m); the second deals with the solution of the problem by the finite element method (poisson2D.m).

```
% Script to prepare the data from the example in section 4.8
% 1/8th of the domain in question
% Coordinates table
vcor=[      0.0      0.0
             0.5      0.5
             0.5      0.0
             1.0      1.0
             1.0      0.5
             1.0      0.0];
% Connectivity table
kconec=[      3      2      1
               5      4      2
               2      3      5
               6      5      3];
nnt=size(vcor,1); % number of nodes
nelt=size(kconec,1); % number of elements
ndln=1; % number of DoF per node
ndlt=nnt*ndln; % total number of DoF
% Physical property: conductivity and volumic stress: vpfv
vpref=[1]; vpfv=1;
```

```
% Dirichlet boundary conditions: nodes 4, 5 and 6
kcond=[0 0 0 1 1 1];
vcond=[0 0 0 0 0 0];
% Display of the mesh
for ie=1:nelt
    x=vcor([kcone(ie,:);kcone(ie,1)],1)';
    y=vcor([kcone(ie,:);kcone(ie,1)],2)';
    plot(x,y,'o-')
    hold on
end
```

Figure 4.5. Program to prepare the data (section 4.8)

```
% Simple finite element program
% Resolution of Poisson's equation in 2D: Example section 4.8
% Use of the data script data1t3.m
%
% Reading of geometric data
data1t3
% Initialization of the global matrices and vectors
vkg=zeros(ndlt); % global stiffness matrix
vfg=zeros(ndlt,1); % global load vector
% Loop of assembly on finite elements
for ie=1:nelt
    fprintf('----- element ie = %5i\n',ie)
    vc当地=vcor(kcone(ie,:)); % elementary coordinates
    kloce=kcone(ie,:); % vector of localization of the DoF
    % Calculation of the elementary quantities
    detj= (vc当地(2,1)-vc当地(1,1))*(vc当地(3,2)-vc当地(1,2)) ...
        -(vc当地(3,1)-vc当地(1,1))*(vc当地(2,2)-vc当地(1,2));
    % Gradient matrix [B] (2X3)
    vb=[ (vc当地([2 3 1],2)-vc当地([3 1 2],2))'
        (vc当地([3 1 2],1)-vc当地([2 3 1],1))' ]./detj;
    % Elementary stiffness matrix
    vke=vb'*(0.5*detj*vprel(1))*vb;
    % Elementary load vector
    vfe=(vpfv*detj/6)*[1 1 1]';
    % Assembly
```

```

vkg(kloce,kloce)=vkg(kloce,kloce)+vke;
vfg(kloce)=vfg(kloce)+vfe;
end
% Introduction of the Dirichlet boundary conditions
% Method of the unit term on the diagonal
for ic=1:ndlt
    if(kcond(ic)==1)
        vfg=vfg-vkg(:,ic)*vcond(ic);
        vkg(ic,:)=zeros(1,ndlt);
        vkg(ic,ic)=1; vfg(ic)=vcond(ic);
    end
end
% Resolution
vsol=vkg\vfg;
vsol'

```

Figure 4.6. Problem-solving program for Poisson's equation (section 4.8)

4.9 Some concepts about convergence, stability and error calculation

Here, we present some interesting properties for solutions obtained using the finite element method discretizing Galerkin type integral forms. The notions of locking and exact one-dimensional finite elements are also discussed in this section.

4.9.1 NOTATIONS

- A bilinear form will be denoted $W_{\text{int}}(\delta u, u)$. The function u represents the solution function; the functions v, w and δu are represent the space of test functions admissible for a continuum. In finite element space, we will denote the solution as u^h and v^h, w^h , the test functions from the approximation space.
- A quadratic form is positive definite (or elliptical) if:

$$W_{\text{int}}(v, v) > 0, \forall v \neq 0. \quad (4.57)$$

The quadratic form $W(u + \Delta u, u + \Delta u)$ can be developed in accordance with:

$$W_{\text{int}}(u + \Delta u, u + \Delta u) = W_{\text{int}}(u, u) + 2W_{\text{int}}(\Delta u, u) + W_{\text{int}}(\Delta u, \Delta u), \quad (4.58)$$

with the following symmetry property:

$$W_{\text{int}}(\Delta u, u) = W_{\text{int}}(u, \Delta u).$$

The following equations illustrate a few of the most commonly encountered linear and quadratic forms:

$$W_{\text{int}}(\delta u, \delta u) = \int_0^L \frac{d\delta u}{dx} \cdot \frac{d\delta u}{dx} dx \quad : \text{quadratic}$$

$$W_{\text{int}}(\delta u, \delta u) = \int_A \langle \delta \epsilon \rangle [H] \{ \delta \epsilon \} dA \quad : \text{quadratic}$$

$$W_{\text{ext}}(\delta u, f) = \int_0^L \delta u \cdot f dx \quad : \text{linear}$$

- The energy norm is defined by: $\frac{1}{2} W_{\text{int}}(u, u)$.
- In the finite element approximation space, we write as:

$$W_{\text{int}}(\delta u^h, \delta u^h) = \int_A \langle \delta \epsilon^h \rangle [D] \{ \delta \epsilon^h \} dA = \langle \delta u_n \rangle [k] \{ \delta u_n \}.$$

- The space L^2 is defined as the ensemble of function u^h such that:

$$\int_V (u^h)^2 dV < \infty.$$

- The space C^1 is a set of continuous functions such that:

$$u^h, \frac{\partial u^h}{\partial x}, \frac{\partial u^h}{\partial y}, \frac{\partial u^h}{\partial z} \in L^2.$$

4.9.2 PROPERTIES OF THE EXACT SOLUTION

It is possible to associate a functional Π with a positive definite bilinear form $W = 0$, such that the exact solution u corresponds to the minimum of Π (Chapter 3). Thus:

$$W(\delta u, u) = W_{\text{int}}(\delta u, u) - W_{\text{ext}}(\delta u, f) = 0, \quad \forall \delta u. \quad (4.59)$$

We then have: $W = \delta \Pi = 0, \quad \forall \delta u$

where: $\Pi(v, v) = \frac{1}{2} W_{\text{int}}(v, v) - W_{\text{ext}}(v, f)$.

The minimum of Π corresponding to the solution is $v = u$. Let us posit that $v = u + w$ and develop:

$$\Pi(u + w, u + w) = \frac{1}{2} W_{\text{int}}(u + w, u + w) - W_{\text{ext}}(u + w, f).$$

Using equation (4.58):

$$\begin{aligned}\Pi(v, v) &= \frac{1}{2}W_{\text{int}}(u, u) - W_{\text{ext}}(u, f) \\ &\quad + (W_{\text{int}}(w, u) - W_{\text{ext}}(w, f)) + \frac{1}{2}W_{\text{int}}(w, w),\end{aligned}$$

and equation (4.59) with $\delta u = w$, we get:

$$\begin{aligned}\Pi(v, v) &= \Pi(u, u) + \frac{1}{2}W_{\text{int}}(w, w), \\ \Pi(v, v) &\geq \Pi_{\text{exact}}, \text{ for } W_{\text{int}} \text{ positive definite.}\end{aligned}\tag{4.60}$$

Remark

v, w and δu are arbitrary functions of the admissible space.

4.9.3 PROPERTIES OF THE SOLUTION OBTAINED BY THE FINITE ELEMENT METHOD

The solution obtained by the finite element method u^h is such that:

$$W(v^h, u^h) = W_{\text{int}}(v^h, u^h) - W_{\text{ext}}(v^h, f) = 0, \quad \forall v^h \text{ where } v^h = \delta u^h.\tag{4.61}$$

The exact solution u , given by equation (4.59), satisfies:

$$W(v^h, u) = W_{\text{int}}(v^h, u) - W_{\text{ext}}(v^h, f) = 0, \quad \forall v^h \text{ where } v^h = \delta u.\tag{4.62}$$

The space of admissible functions for a continuous domain contains all the finite element solutions. By subtracting (4.61) from (4.62), we get:

$$\begin{aligned}W_{\text{int}}(v^h, u - u^h) &= 0, \\ \Leftrightarrow W_{\text{int}}(v^h, e^h) &= 0, \text{ where } e^h = u - u^h, \text{ the error in the solution.}\end{aligned}\tag{4.63}$$

This last equation expresses the fact that the solution obtained by the finite element method using a Galerkin integral form is the best there can be in the approximation space in question. It is also possible to demonstrate that:

$$W_{\text{int}}(u^h, u^h) \leq W_{\text{int}}(u, u).\tag{4.64}$$

Indeed, we can write:

$$\begin{aligned} W_{\text{int}}(u, u) &= W_{\text{int}}(u^h + e^h, u^h + e^h), \\ &= W_{\text{int}}(u^h, u^h) + \underbrace{2W_{\text{int}}(e^h, u^h)}_{=0} + W_{\text{int}}(e^h, e^h). \end{aligned}$$

From equation (4.63), we deduce (4.64). This equation stipulates that the exact energy norm $W_{\text{int}}(u, u)$ (also called internal energy) is always greater than that corresponding to the solution found by the finite element method, $W_{\text{int}}(u^h, u^h)$ (not to be confused with the functional Π).

Let us also show (see [STR72], p. 40) the following relation:

$$W_{\text{int}}(u - u^h, u - u^h) \leq W_{\text{int}}(u - v^h, u - v^h), \quad (4.65)$$

where v^h represents the admissible function space and u^h the solution function given by the finite element method in that space. Thus:

$$\begin{aligned} W_{\text{int}}(u - u^h - w^h, u - u^h - w^h) &= \\ W_{\text{int}}(u - u^h, u - u^h) - 2W_{\text{int}}(w^h, u - u^h) + W_{\text{int}}(w^h, w^h). \end{aligned}$$

Equation (4.63) can also be written as:

$$W_{\text{int}}(v^h, e^h) = W_{\text{int}}(w^h, e^h) = 0, \text{ where } e^h = u - u^h.$$

because v^h and w^h are arbitrary functions of the approximation space. By choosing $v^h = u^h + w^h$, we finally get equation (4.65).

The norm of the error $W_{\text{int}}(u - u^h, u - u^h)$ of the solution yielded by the finite element method is the smallest of the norms of $W_{\text{int}}(u - v^h, u - v^h)$.

This property is remarkable for calculations to estimate the approximation error in the finite element method. The error estimator $\|u - u^h\|$ given in Chapter 1 was not associated with a finite element solution but rather with an arbitrary function v^h of the approximation space. Equation (4.65) stipulates, therefore, that the estimators presented in this chapter give an upper bound for the error calculation.

EXAMPLE 4.27. Expressions of $W_{\text{int}}(v^h, u^h)$

| Consider the integral form:

$$W(v^h, u^h) = W_{\text{int}}(v^h, u^h) - W_{\text{ext}}(v^h, f) = 0, \quad \forall v^h$$

where: $W_{\text{int}}(v^h, u^h) = \int_A \langle \boldsymbol{\epsilon}(v) \rangle [H] \{ \boldsymbol{\epsilon}^h \} dA,$

$$W_{\text{ext}}(v^h, f) = \int_A v^h f dx.$$

For a Laplacian problem:

$$\langle \boldsymbol{\epsilon}^h \rangle = \begin{pmatrix} \frac{\partial u^h}{\partial x} & \frac{\partial u^h}{\partial y} \end{pmatrix}, \quad [H] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

$$\langle \boldsymbol{\epsilon}(v) = \boldsymbol{\delta}\boldsymbol{\epsilon} \rangle = \begin{pmatrix} \frac{\partial v^h}{\partial x} & \frac{\partial v^h}{\partial y} \end{pmatrix} = \begin{pmatrix} \frac{\partial \delta u^h}{\partial x} & \frac{\partial \delta u^h}{\partial y} \end{pmatrix}, \text{ where } v^h = \delta u^h.$$

The corresponding internal energy is:

$$\frac{1}{2} W_{\text{int}}(u, u) = \frac{1}{2} \int_A \left[\left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} \right)^2 \right] dA,$$

$$\frac{1}{2} W_{\text{int}}(u^h, u^h) = \frac{1}{2} \int_A \left[\left(\frac{\partial u^h}{\partial x} \right)^2 + \left(\frac{\partial u^h}{\partial y} \right)^2 \right] dA,$$

where $W_{\text{int}}(u^h, u^h) \leq W_{\text{int}}(u, u).$

Equation (4.63) is written as:

$$W_{\text{int}}(v^h, e^h) = \int_A \left(\frac{\partial \delta u^h}{\partial x} \frac{\partial(u - u^h)}{\partial x} + \frac{\partial \delta u^h}{\partial y} \frac{\partial(u - u^h)}{\partial y} \right) dA = 0,$$

where $\boldsymbol{\delta}u^h = v^h.$

Take $v^h = u_I$, the function that coincides with the exact solution at the nodes of the element. Equation (4.65) is written as:

$$W_{\text{int}}(e^h, e^h) = \int_A \left(\left(\frac{\partial(u - u^h)}{\partial x} \right)^2 + \left(\frac{\partial(u - u^h)}{\partial y} \right)^2 \right) dA,$$

$$\leq \int_A \left(\left(\frac{\partial(u - u_I)}{\partial x} \right)^2 + \left(\frac{\partial(u - u_I)}{\partial y} \right)^2 \right) dA.$$

where u^h is the finite element solution.

4.9.4 STABILITY AND LOCKING

In section 4.1.2, we presented the criteria of convergence toward the exact solution for a finite element solution. In brief, they are as follows:

- The approximation is consistent (complete polynomial basis up to the m th order for a variational form accepting maximum orders of the derivatives up to the m th order) and continuous.
- The discretized relation admits a solution stripped of all spurious oscillation and presents no locking for any mesh configuration. However, this criterion is not always simple to satisfy.

For a variational form with positive definite internal energy and in the absence of constraints such as incompressibility or zero shear, the conditions of consistency and stability are sufficient to ensure convergence.

For a scalar transport/diffusion problem (rendered non-symmetrical by the transport term), the criterion of positivity ensures the absence of numerical oscillations (section 4.1.2). The taking into account of constraints such as incompressibility requires that the discrete model satisfies the Babuška–Brezzi condition [STR 73; HUG 87]. This condition is very often stringent in real-world situations such as an arbitrary mesh and various boundary conditions. Thus, many stable elements are the fruit of a more heuristic approach, which frequently consists of verifying the rank of the different submatrices of an element, and then of a patch of elements with the strictest of boundary conditions possible. We present two examples to illustrate this kind of approach.

EXAMPLE 4.28. Shear locking

Consider the integral form associated with a thick beam with terms of transverse shear γ [BAZ 90]:

$$W_{\text{int}}^e = \int_0^L \left(\frac{\partial \delta \beta}{\partial x} H_f \frac{\partial \beta}{\partial x} + \delta \gamma \cdot G \cdot h \cdot \gamma \right) dx$$

where $\gamma = \frac{\partial \beta}{\partial x} + \frac{\partial w}{\partial x}$; $H_f = \frac{Eh^3}{12}$; $G = \frac{E}{2(1+\nu)}$; width = 1

E : Young's modulus; ν : Poisson coefficient;

h : thickness.

Using an element with two nodes and two degrees of freedom per node (transverse displacement w and rotation), we obtain the following discrete form:

$$\begin{aligned}
W_h &= \langle \delta u_n \rangle \left(\frac{H_f}{L} [k_f] + G.h.L [k_c] \right) \{u_n\}, \quad \text{where } \{u_n\} = \begin{pmatrix} w_1 \\ \beta_1 \\ w_2 \\ \beta_2 \end{pmatrix}, \\
[k_f] &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ sym & 0 & 0 & 1 \end{bmatrix}, \quad [k_c] = \begin{bmatrix} 1/L^2 & -1/2L & -1/L^2 & -1/2L \\ 1/3 & 1/2L & 1/6 & 0 \\ sym & 1/L^2 & 1/2L & 1/3 \end{bmatrix}, \\
rank(k_f) &= 1, \quad rank(k_c) = 2.
\end{aligned}$$

Following assembly and introduction of the boundary conditions:

$$\left(\frac{H_f}{L} [K_f] + G.h.L [K_c] \right) \{U\} = \{F\}.$$

For a thin beam, we have: $h/L \ll 1$ and $GhL \gg H_f/L$, with physical behavior controlled by bending.

The finite element solution:

$$\{U\} = \frac{1}{GhL} [K_c]^{-1} \{F\}, \quad \text{for an invertible } [K_c],$$

expresses a numerical behavior that is exclusively controlled by shear instead of bending: the numerical model exhibits severe locking where shearing is preponderant in relation to bending, because: $GhL \gg H_f/L$.

Thus, it is necessary to control the influence of $[K_c]$, i.e. reduce its rank so that bending remains dominant for $h/L \ll 1$. The simplest approach is to choose constant γ and $\delta\gamma$:

$$\gamma = \left(\beta + \frac{\partial w}{\partial x} \right) \Big|_{x=L/2} = \frac{\beta_1 + \beta_2}{2} + \frac{w_2 - w_1}{L}.$$

The matrix $[k_c]$ is then written as:

$$\begin{aligned}
G.h.L [k_c] &= G.h.L \begin{bmatrix} 1/L^2 & -1/2L & -1/L^2 & -1/2L \\ 1/4 & 1/2L & 1/4 & 0 \\ sym & 1/L^2 & 1/2L & 1/4 \end{bmatrix}, \\
rank(k_c) &= 1, \quad rank(k_f + k_c) = 2.
\end{aligned}$$

This element reduces the effect of shear locking for $h/L \ll 1$, with the ranks of $[K_c]$ and $[K_f + K_c]$ being respectively equal to n_{el} and $(2 \times n_{el} - 2)$ where n_{el} is the number of elements.

The element $Q4\gamma$ [BA2 90] generalizes this principle for a plate element. We can verify the absence of locking by way of patch tests.

EXAMPLE 4.29. Locking in incompressibility

Consider the Stokes problem defined by the following integral form:

$$W_{\text{int}} = \int_A \left(\langle \delta \boldsymbol{\varepsilon} \rangle [H] \{ \boldsymbol{\varepsilon} \} - \text{Div}(\boldsymbol{\delta u}) p - \delta p \cdot \text{Div}(\mathbf{u}) + \delta p \frac{p}{\lambda} \right) dA.$$

A finite element discretization of this form, with $p \in L^2$ and $\mathbf{u} \in C^0$, enables us to define the following elementary stiffness matrix (see section 4.3.4.5):

$$[k] = \begin{bmatrix} & & | & \\ & k_{uu} & | & -k_{up} \\ & \hline & -k_{up} & | & -A/\lambda \end{bmatrix}.$$

Let us use the notation N_{uv} for the number of unknown velocity variables; N_p for the number of unknown pressure variables; and N_{CL} for the number of boundary conditions imposed. The resulting scheme is stable under the following conditions:

– at the elementary level:

$$\text{rank}(k_{uu}) > N_p.$$

– for a patch of elements with the nodal velocities imposed all along the contour:

$$N_{uv} - N_{CL} > N_p.$$

It is also vital to ensure that the number of independent columns of $[k_{up}]$ or $[K_{up}]$ be equal to N_p . The Babuška–Brezzi condition is written as:

$$\frac{|\langle u^h \rangle [k_{up}] \{ p^h \}|}{|u^h| \| p^h \|} > 0, \quad \forall u^h, p^h \text{ of the approximation space},$$

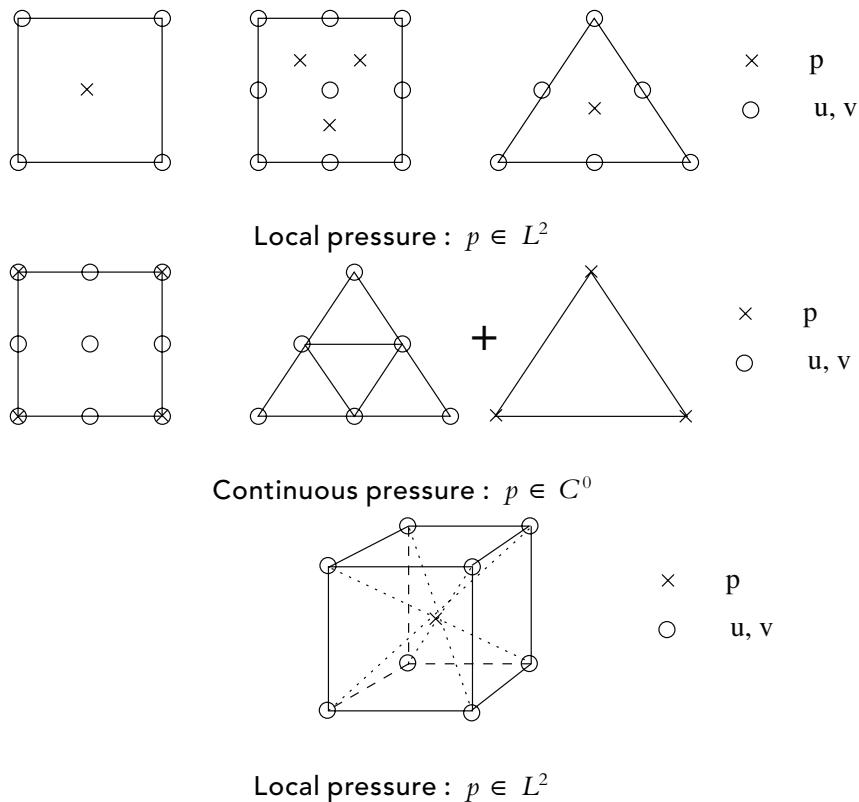
or indeed, for an assembled system with boundary conditions:

$$\text{rank} \left(\begin{bmatrix} K_{pu} \\ K_{uu} \end{bmatrix} \begin{bmatrix} K_{uu} \end{bmatrix}^{-1} \begin{bmatrix} K_{up} \end{bmatrix} \right) = N_p,$$

with $\text{rank} \left(\begin{bmatrix} K_{uu} \end{bmatrix} \right) > N_p$.

In practice, this condition is equivalent to ensuring that the number of velocity variables is greater than the number of pressure variables, following the introduction of the boundary conditions.

The figures illustrate a few elements that are commonly used in incompressible fluid mechanics.



However, it remains difficult to satisfy the stability conditions in certain scenarios.

4.9.5 ONE-DIMENSIONAL EXACT FINITE ELEMENTS

For stationary, monodimensional problems with constant coefficients, it is possible to obtain a finite element formulation for which the solution at the nodes is exact.

Consider the monodimensional problem defined by:

$$\mathcal{L}(u) + f_v = 0. \quad (4.66)$$

The associated weak expression is written as:

$$W = \int_0^L (\mathcal{L}_1(\delta u) \cdot \mathcal{L}_2(u) - \delta u f_v) dx + [\delta u \dots]_0^L = 0, \quad \forall \delta u.$$

The exact solution associated with this problem also satisfies:

$$W = \int_0^L (\mathcal{L}_1(\delta u^h) \cdot \mathcal{L}_2(u) - \delta u^h f_v) dx + [\delta u^h \dots]_0^L = 0, \quad \forall \delta u^h, \quad (4.67)$$

because $\delta u^h \in \delta u$.

In addition, the finite element solution satisfies:

$$W^h = \int_0^L (\mathcal{L}_1(\delta u^h) \cdot \mathcal{L}_2(u^h) - \delta u^h f_v) dx + [\delta u^h \dots]_0^L = 0, \quad \forall \delta u^h, \quad (4.68)$$

The difference between the two expressions (4.67) and (4.68) is written as:

$$W - W^h = \int_0^L \mathcal{L}_1(\delta u^h) \cdot \mathcal{L}_2(u - u^h) dx = 0, \quad (4.69)$$

Put differently, taken to the elementary level:

$$W_e - W_e^h = \int_0^{L_e} \mathcal{L}_1(\delta u^h) \cdot \mathcal{L}_2(u - u^h) dx.$$

The successive application of one or more rounds of integration by parts (depending on the degree of the derivatives in question) gives us:

$$W_e - W_e^h = \int_0^{L_e} \mathcal{L}(\delta u^h) \cdot (u - u^h) dx + [\dots (u - u^h)]_0^{L_e}. \quad (4.70)$$

The choice of δu^h such that:

$$\mathcal{L}(\delta u^h) = 0, \text{ on each element} \quad (4.71)$$

leads us to:

$$W_e - W_e^h = \langle \delta u_n \rangle [k] \{ u_n - u_n^h \}. \quad (4.72)$$

Following assembly on the elements and introduction of the boundary conditions, we finally get:

$$[K]\{U_n - U_n^h\} = \{0\}, \quad (4.73)$$

Hence:

$$\{U_n\} = \{U_{exact}\} = \{U_n^h\}_{(at\ the\ nodes)}.$$

EXAMPLE 4.30. Exact solution (monodimensional problem)

a) Consider the diffusion-type scalar problem whose elementary variational form is given below:

$$W_e = \int_0^{L_e} (\delta u_{,x}^h \cdot u_{,x} - \delta u \cdot f) dx, \text{ with the notation } u_{,x} \equiv \frac{du}{dx}.$$

This being the case, equation (4.69) is written as:

$$W_e - W_e^h = \int_0^{L_e} \delta u_{,x}^h \cdot (u_{,x} - u_{,x}^h) dx.$$

Integration by parts gives us:

$$W_e - W_e^h = \int_0^{L_e} \delta u_{,xx}^h \cdot (u - u^h) dx + [\delta u_{,x}^h \cdot (u - u^h)]_0^{L_e}.$$

For a two-node linear truss element, we have:

$$\delta u_{,xx} = 0 \text{ and } [\delta u_{,x} \cdot (u - u^h)]_0^{L_e} = \langle \delta u_1 \quad \delta u_2 \rangle \frac{1}{L_e} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} u_1 - u_1^h \\ u_2 - u_2^h \end{Bmatrix},$$

where u_1 and u_2 are the exact nodal values.

Following assembly and introduction of the boundary conditions, we get:

$$[K]\{U_n - U_n^h\} = \{0\}, \text{ hence } \{U_{exact}\} = \{U_n^h\}.$$

b) Let us now consider the transport/diffusion problem defined by:

$$W_e = \int_0^{L_e} (\delta u_{,x}^h \cdot a \cdot u_{,x} + \delta u_{,x}^h \cdot k \cdot u_{,x} - \delta u \cdot f) dx,$$

and:

$$W_e - W_e^h = \int_0^{L_e} (\delta u^h \cdot a + \delta u_{,x}^h \cdot k) (u_{,x} - u_{,x}^h) dx.$$

Integration by parts leads to:

$$\begin{aligned} W_e - W_e^h &= - \int_0^{L_e} (a \cdot \delta u_{,x}^h + k \cdot \delta u_{,xx}^h) \cdot (u - u^h) dx \\ &\quad + \left[(\delta u^h \cdot a + \delta u_{,x}^h \cdot k) (u - u^h) \right]_0^{L_e}. \end{aligned}$$

Choosing a value of δu^h such that:

$$a \cdot \delta u_{,x}^h + k \cdot \delta u_{,xx}^h = 0, \text{ on each element}$$

enables us to get the exact solution at the nodes:

$$\delta u^h = N_1 \cdot \delta u_1 + N_2 \cdot \delta u_2,$$

$$\text{with: } N_1(x) = 1 - \frac{e^{-P_e \frac{x}{L_e}} - 1}{e^{-P_e} - 1} \text{ and } N_2(x) = \frac{e^{-P_e \frac{x}{L_e}} - 1}{e^{-P_e} - 1},$$

$$\text{where: } P_e = \frac{a L_e}{k} \text{ (local P\'eclet number).}$$

Thus:

$$\begin{aligned} W_e - W_e^h &= \langle \delta u_1 \quad \delta u_2 \rangle \begin{bmatrix} k \\ k \end{bmatrix} \begin{Bmatrix} u_1 - u_1^h \\ u_2 - u_2^h \end{Bmatrix}, \\ \text{where } \begin{bmatrix} k \\ k \end{bmatrix} &= \frac{P_e / L_e}{e^{-P_e} - 1} \begin{bmatrix} 1 & -1 \\ -e^{-P_e} & e^{-P_e} \end{bmatrix} + \begin{bmatrix} -a & 0 \\ 0 & a \end{bmatrix}. \end{aligned}$$

The use of this elementary matrix enables us to obtain the exact solution at the nodes. For a mesh composed of two elements with the boundary conditions

$u_1 = 0$ and $u_3 = 1$, where: $L = 2$, $L_e = 1$, $a = k = 1$, we have:

$$P_e = 1 \text{ and } \frac{1}{e^{-1} - 1} ((1 + e^{-1}) u_2 - u_3) = 0,$$

$$\text{Hence: } u_2 = \frac{1}{1 + e^{-1}} = \text{exact solution.}$$

IMPORTANT RESULTS

Integral form:

$$W = \sum_{e=1}^{n_e} W^e = 0. \quad (4.2a)$$

Discretized elementary integral form:

$$W^e = \langle \delta u_n \rangle ([k] \{u_n\} - \{f\}). \quad (4.4)$$

Discretized global integral form:

$$W = \langle \delta U_n \rangle ([K] \{U_n\} - \{F\}) = 0. \quad (4.5b)$$

Elementary mass matrix:

$$[m] = \int_{V_e} \{N\} \langle N \rangle dV. \quad (4.6b)$$

Elementary (or stiffness) matrix:

$$[k] = \int_{V_e} [B_\delta]^T [H] [B] dV \quad (\text{see also 4.16a}). \quad (4.10b)$$

Elementary load vector:

$$\{f\} = \int_{V_e} \{N\} f_V dV + \int_{S_f} \{N\} f_S dS \quad (\text{see also 4.16b}). \quad (4.10c)$$

Transformation of the matrix $[B]$:

$$[B] = [Q] [B_\xi]. \quad (4.12)$$

Rules of assembly:

$$K_{IJ}^e \equiv k_{ij} \quad \begin{array}{l} I = \text{LOCE}(i) \\ J = \text{LOCE}(j). \end{array} \quad (4.29a)$$

Transformation of the variables:

$$[K'] = [R]^T [K] [R] \quad \{F'\} = [R]^T [F]. \quad (4.52)$$

Notations

b, h	width and height of the band [K]
$[B]$	matrix linking the gradients in \mathbf{x} to the nodal variables
$[B_\delta]$	matrix linking the variations of the gradients to the variations of the nodal variables
$[B_\xi]$	matrix linking the gradients in ξ to the nodal variables
$[H]$	matrix of the physical properties
e	indicator of an element
$\{f\}$	elementary load vector (or right-hand side or equivalent forces)
$\{F\}$	global load vector
$\{F^e\}$	extended elementary vector
$[k]$	elementary (or stiffness) matrix of an element
$[K]$	global matrix
$[K^e]$	extended elementary matrix
$[m]$	mass matrix of an element
n	total number of nodes
n_b	number of blocks of a global matrix segmented on disk
n_d	total number of degrees of freedom
n_{de}	number of degrees of freedom per element
n_{dn}	number of degrees of freedom per node
n_e	number of nodes per element
n_{el}	total number of elements
$[Q]$	matrix of transformation of the gradient
$[T], [R], [r]$	matrices of transformation of the nodal variables
$\{u_n\}$	nodal variables of an element
$\{U_n\}$	set of all the nodal variables
W	global integral form
W^e	elementary integral form
$\{\delta u_n\}$	variation of the nodal variables of an element
$\{\delta U_n\}$	variation of the ensemble of nodal variables
ξ_r, w_r	coordinates and weights of the numerical integration points.

Bibliography

- [BA1 90] BATOZ J.L., DHATT G., *Modélisation des structures par éléments finis*, Vol. 1, Hermès, 1990.
- [BA2 90] BATOZ J.L., DHATT G., *Modélisation des structures par éléments finis*, Vol. 2, Hermès, 1990.
- [BAT 76] BATHE K.J., WILSON E.L., *Numerical Methods in Finite Element Analysis*, Prentice Hall, 1976.
- [HIR 88] HIRSH C., *Numerical Computation of Internal and External Flows- Vol. 1: Fundamentals of Numerical Discretization*, Wiley and Sons, 1988.
- [HIR 90] HIRSH C., *Numerical Computation of Internal and External Flows- Vol. 2: Computational Methods for Inviscid and Viscous Flows*, Wiley and Sons, 1990.
- [HUG 87] HUGHES T.J., *The Finite Element Method, Linear, Static and Dynamic Analysis*, Prentice Hall, 1987.
- [HUB 84] HUBERT G., Modélisation d'écoulements de fluides incompressibles par la méthode des éléments finis, Doctoral thesis, Compiègne University of Technology, 1984.
- [IRO 72] IRONS B.M., RAZZAQUE A.A., “Experience with the Path Test”, in *Mathematical Foundations of the Finite Element Method*, Academic Press, pp. 557–587, 1972,
- [STR 73] STRANG G., FIX G., *An Analysis of Finite Element Method*, Prentice Hall, 1973.
- [WAS 75] WASHIZU K., *Variational Methods in Elasticity and Plasticity*, 2 ed., Pergamon Press, 1975.
- [ZIE 00] ZIENKIEWICZ O.C., *The Finite Element Method: The Basis (Vol. 1), Solid Mechanics (Vol. 2) & Fluid Mechanics (Vol. 3)*, 5th ed., Butterworth Heinermann, 2000.

CHAPTER 5

Numerical Methods

5.0 Introduction

The effective implementation of the finite element method described in previous chapters requires the use of various numerical methods to build the elementary matrices and solve the resultant systems of algebraic equations (Figure 5.1). This chapter describes the most commonly used numerical methods, but does not give an exhaustive presentation of all available methods.

We first review the **numerical integration** methods used to construct elementary matrices and vectors by integration over the reference element. Numerical integration formulae are proposed for various one-, two- and three-dimensional domains.

The second section is devoted to the solution of linear systems of equations. The **Gaussian elimination** method and the corresponding **decomposition** techniques are then described and subsequently adapted to matrices stored by the skyline technique.

The **substitution** and **Newton–Raphson** type methods are then described which enable us to solve systems of nonlinear equations by iteratively solving a series of linear equations.

Then we describe methods of solving first- and second-order **unsteady** problems. Different schemes of time discretization, explicit and implicit, are proposed which transform an unsteady system into a steady state system for each time interval.

Finally, the last section is devoted to the extraction of **eigenvalues** and **eigenvectors** by iterative methods such as **inverse iteration** and **subspace** methods.

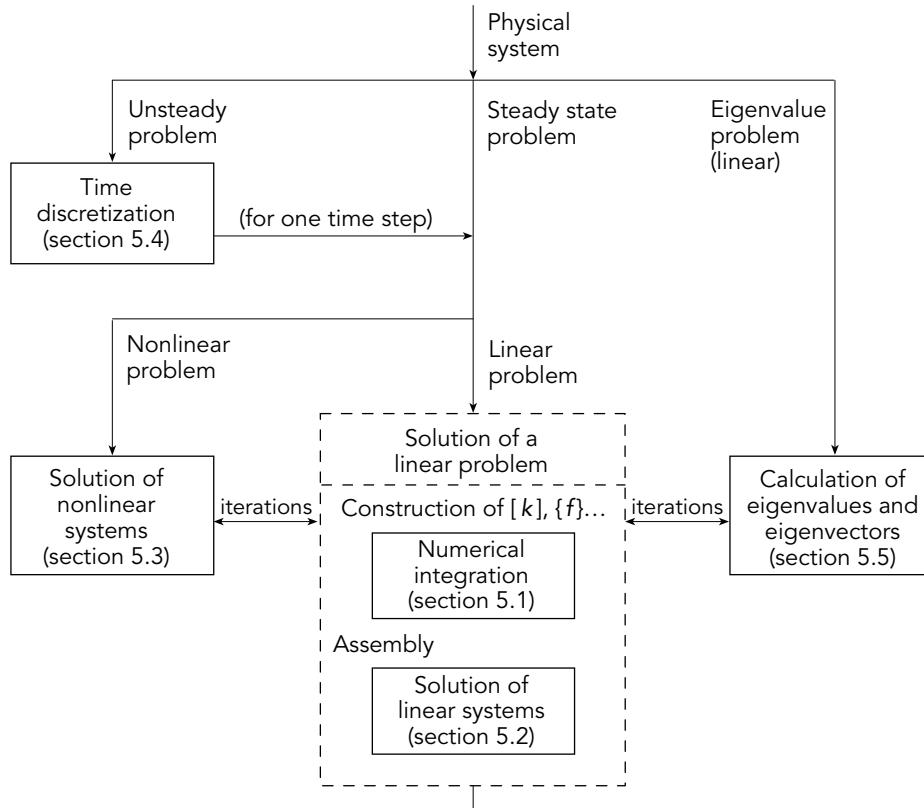


Figure 5.1. Numerical methods used in the finite element method

5.1 Numerical integration

5.1.1 INTRODUCTION

In the finite element method, the elementary matrix $[k]$ and the elementary load vector $\{f\}$ are expressed as integrals over lines, areas and volumes (4.10b, 4.10c) defined over the geometry of the real element V^e :

$$\begin{aligned}[k] &= \int_{V^e} [B_\delta]^T [H][B] dV \\ \{f\} &= \int_{V^e} \{N\} f_V dV + \int_{S_f^e} \{N\} f_S dS. \end{aligned} \tag{5.1a}$$

On the reference element, these integrals become (4.16a) and (4.16b):

$$\begin{aligned}[k] &= \int_{V^r} [B_\delta(\xi)]^T [H(\xi)] [B(\xi)] \det(J(\xi)) dV^r \\ \{f\} &= \int_{V^r} \{N(\xi)\} f_V \det(J(\xi)) dV^r + \int_{S_f^r} \{N(\xi_s)\} f_S dS\end{aligned}\quad (5.1b)$$

where: V^r is the volume of the reference element;
 S_f^r is the part of the boundary of the reference element to which the load f_S is applied;
 ξ_s represents the coordinates ξ on the boundary S_f^r ;
 $dS = J_S ds_1 ds_2$ (see section 4.2.3.4);
 $[J]$ is the Jacobian matrix of the geometric transformation (1.5.2);

or, in more compact terms:

$$\begin{aligned}[k^*] &= \int_{V^r} [k^*] dV^r \\ \{f^*\} &= \int_{V^r} \{f_V^*\} dV^r + \int_{S_f^r} \{f_S^*\} dS\end{aligned}\quad (5.2)$$

where $[k^*] = [B_\delta(\xi)]^T [H(\xi)] [B(\xi)] \det(J(\xi))$
 $\{f_V^*\} = \{N(\xi)\} f_V \det(J(\xi))$
 $\{f_S^*\} = \{N(\xi_s)\} f_S$.

The terms $[k^*]$, $\{f_V^*\}$ and $\{f_S^*\}$ are polynomials or complicated rational fractions. It is difficult to integrate them explicitly unless they consist of polynomial terms. Below, we present the explicit integrals of monomials on the classical reference elements:

One dimension

$$\int_{-1}^1 \xi^i d\xi = \begin{cases} 0 & \text{if } i \text{ is odd} \\ \frac{2}{i+1} & \text{if } i \text{ is even.} \end{cases} \quad (5.3a)$$

Two dimensions

- Square reference element:

$$\int_{-1}^1 \int_{-1}^1 \xi^i \eta^j d\xi d\eta = \begin{cases} 0 & \text{if } i \text{ or } j \text{ is odd} \\ \frac{4}{(i+1)(j+1)} & \text{if } i \text{ and } j \text{ are even.} \end{cases} \quad (5.3b)$$

- **Triangular reference element:**

$$\int_0^1 \int_0^{1-\xi} \xi^i \eta^j d\eta d\xi = \frac{i! j!}{(i+j+2)!}. \quad (5.3c)$$

Three dimensions

- **Cubic reference element:**

$$\begin{aligned} & \int_{-1}^1 \int_{-1}^1 \int_{-1}^1 \xi^i \eta^j \zeta^k d\xi d\eta d\zeta = \\ &= \begin{cases} 0 & \text{if } i, j \text{ or } k \text{ is odd} \\ \frac{8}{(i+1)(j+1)(k+1)} & \text{if } i \text{ and } j \text{ and } k \text{ are even.} \end{cases} \end{aligned} \quad (5.3d)$$

- **Tetrahedral reference element:**

$$\int_0^1 \int_0^{1-\xi} \int_0^{1-\xi-\eta} \xi^i \eta^j \zeta^k d\zeta d\eta d\xi = \frac{i! j! k!}{(i+j+k+3)!} \quad (5.3e)$$

For general cases where explicit integration is not possible, we employ the numerical integration formulas having the following generic form:

$$\begin{aligned} [k] &= \sum_{i=1}^r w_i [k^*(\xi_i)] \\ \{f\} &= \sum_{i=1}^r w_i [f_V^*(\xi_i)] \end{aligned} \quad (5.4)$$

where: ξ_i are the coordinates of the r **integration points**;

w_i are the **weighting coefficients** (or weights) corresponding to each integration point.

For complex elements, geometries, particularly curvilinear elements, numerical integration (5.4) only gives an approximate value of $[k]$ and $\{f\}$.

5.1.2 ONE-DIMENSIONAL NUMERICAL INTEGRATION [DAV 75; STR 71; KOP 61]

5.1.2.1 Gaussian quadrature method

The Gaussian method is a widely used numerical integration method in which r pairs of weights w_i and coordinates ξ_i are determined so as to accurately integrate polynomials of order $m \leq 2r - 1$.

Replace the integral of a polynomial function $y(\xi)$ with a linear combination of its values at the integration points ξ_i :

$$\begin{aligned} \int_{-1}^1 y(\xi) d\xi &= w_1 y(\xi_1) + w_2 y(\xi_2) + \cdots + w_r y(\xi_r) \\ &= \sum_{i=1}^r w_i y(\xi_i). \end{aligned} \quad (5.5)$$

We determine the $2r$ coefficients w_i and ξ_i so that (5.5) is satisfied precisely for the following polynomial:

$$y(\xi) = a_1 + a_2 \xi + \cdots + a_{2r} \xi^{2r-1}.$$

Substituting this expression into (5.5):

$$\begin{aligned} a_1 \int_{-1}^1 d\xi + a_2 \int_{-1}^1 \xi d\xi + \cdots + a_{2r} \int_{-1}^1 \xi^{2r-1} d\xi &= a_1 (w_1 + w_2 + \cdots + w_r) + \\ &+ a_2 (w_1 \xi_1 + w_2 \xi_2 + \cdots + w_r \xi_r) + \cdots + \\ &+ a_{2r} (w_1 \xi_1^{2r-1} + w_2 \xi_2^{2r-1} + \cdots + w_r \xi_r^{2r-1}). \end{aligned} \quad (5.6)$$

For (5.6) to be identically verified for arbitrary values of a_1, a_2, \dots, a_{2r} , the following $2r$ relations must be satisfied:

$$\begin{aligned} \int_{-1}^1 \xi^\alpha d\xi &= \frac{2}{\alpha+1} = \sum_{i=1}^r w_i \xi_i^\alpha \quad \alpha = 0, 2, 4, \dots, 2r-2 \\ \int_{-1}^1 \xi^\alpha d\xi &= 0 = \sum_{i=1}^r w_i \xi_i^\alpha \quad \alpha = 1, 3, 5, \dots, 2r-1. \end{aligned} \quad (5.7)$$

That is:

$$\begin{aligned} 2 &= w_1 + w_2 + \cdots + w_r \\ 0 &= w_1 \xi_1 + w_2 \xi_2 + \cdots + w_r \xi_r \\ \frac{2}{3} &= w_1 \xi_1^2 + w_2 \xi_2^2 + \cdots + w_r \xi_r^2 \\ &\dots \\ 0 &= w_1 \xi_1^{2r-1} + w_2 \xi_2^{2r-1} + \cdots + w_r \xi_r^{2r-1} \end{aligned}$$

This system of $2r$ equations is linear in w_i and nonlinear in ξ_i ; the solution of the system is subject to the following conditions:

$$\left. \begin{array}{l} w_i > 0 \\ -1 < \xi_i < 1 \end{array} \right\} i = 1, 2, \dots, r.$$

Remark

The abscissas ξ_i and the associated weight values w_i are selected symmetrically around the origin $\xi = 0$.

EXAMPLE 5.1. Calculation of coefficients of the two-point Gaussian method

In this case $r = 2$; equation (5.5) is written as:

$$\int_{-1}^1 \gamma(\xi) d\xi = w_1 \gamma(\xi_1) + w_2 \gamma(\xi_2).$$

For this approximation to be exact for a polynomial of order $2 r - 1 = 3$, the equations (5.7) must be satisfied:

$$\begin{aligned} 2 &= w_1 + w_2 \\ 0 &= w_1 \xi_1 + w_2 \xi_2 \\ \frac{2}{3} &= w_1 \xi_1^2 + w_2 \xi_2^2. \\ 0 &= w_1 \xi_1^3 + w_2 \xi_2^3. \end{aligned}$$

The solution of this system is:

$$w_1 = w_2 = 1; \quad \xi_1 = -\xi_2 = \frac{1}{\sqrt{3}}.$$

The abscissas, ξ_i , are always selected within the domain $(-1 < \xi_i < 1)$ for the Gaussian integration method. Whereas, the Gauss–Lobatto integrations scheme includes points situated at the ends of the domain, i.e. $\xi_i = \pm 1$, the internal points being determined by the Gaussian quadrature method. However, in order to obtain a similar degree of precision of integration, the Gauss–Lobatto scheme requires the use of an extra point. Using this method can sometimes prove more effective for the integration of elementary terms: integration with (2×2) Gauss–Lobatto points on a Q4-type element enables us to obtain the elementary diagonal mass matrix directly (reduced integration).

EXAMPLE 5.2. Gauss–Lobatto method

We demonstrate, in this example, the calculation of the integration weights to obtain a correct integration, where the degree of the polynomial is a function of the number of points considered.

Two-point integration (order 1): $\xi_i = \pm 1$.

The two weights w_1 and w_2 for accurate integration of a linear polynomial are calculated

according to: $w_1 + w_2 = \int_{-1}^1 d\xi = 2$,

$$w_1 \xi_1 + w_2 \xi_2 = \int_{-1}^1 \xi d\xi = 0, \text{ where } \xi_1 = -1, \xi_2 = 1,$$

Thus $w_1 = w_2 = 1$.

Three-point integration (order 3):

The unknowns are the three integration weights and the coordinate of the third point. The system to be solved is given by:

$$w_1 + w_2 + w_3 = \int_{-1}^1 d\xi = 2,$$

$$-w_1 + w_2 + w_3 \xi_3 = \int_{-1}^1 \xi d\xi = 0,$$

$$w_1 + w_2 + w_3 \xi_3^2 = \int_{-1}^1 \xi^2 d\xi = \frac{2}{3},$$

$$-w_1 + w_2 + w_3 \xi_3^3 = \int_{-1}^1 \xi^3 d\xi = 0, \text{ where } \xi_1 = -1, \xi_2 = 1,$$

The solution is: $w_1 = w_2 = \frac{1}{3}$, $w_3 = \frac{4}{3}$ and $\xi_3 = 0$.

Four-point integration (order 5):

The unknowns are w_1 , w_2 , (w_3, ξ_3) and (w_4, ξ_4) where $\xi_1 = -1$ and $\xi_2 = 1$. For reasons of symmetry we posit: $w_1 = w_2$, $w_3 = w_4$ and $\xi_3 = -\xi_4$. The system to be solved is written as:

$$w_1 + w_2 + w_3 + w_4 = 2,$$

$$w_1 + w_2 + w_3 \xi_3^2 + w_4 \xi_4^2 = \frac{2}{3},$$

$$w_1 + w_2 + w_3 \xi_3^4 + w_4 \xi_4^4 = \frac{2}{5},$$

r	ξ_i	w_i	Error	Maximum degree of polynomials integrated precisely
1	0	2	$\frac{1}{3} \frac{d^2 y}{d\xi^2}$	1
2	$\pm 0.577350269189626 (\pm 1/\sqrt{3})$	1	$\approx 0.7 \times 10^{-2} \frac{d^4 y}{d\xi^4}$	3
3	$\pm 0.774596669241483 (\pm \sqrt{3}/5)$	$0.8888888888888889 (8/9)$ $0.5555555555555556 (5/9)$	$\approx 0.7 \times 10^{-4} \frac{d^6 y}{d\xi^6}$	5
4	$\pm 0.339981043584856 \left(\pm \sqrt{\frac{3-2\sqrt{6/5}}{7}} \right)$ $\pm 0.861136311594053 \left(\pm \sqrt{\frac{3+2\sqrt{6/5}}{7}} \right)$	$0.652145154862546 \left(\frac{1}{2} + \frac{1}{6\sqrt{6/5}} \right)$ $0.347854845137454 \left(\frac{1}{2} - \frac{1}{6\sqrt{6/5}} \right)$	$\approx 0.3 \times 10^{-6} \frac{d^8 y}{d\xi^8}$	7

$$\int_{-1}^1 y(\xi) d\xi = \sum_{i=1}^r w_i y(\xi_i)$$

Figure 5.2. One-dimensional Gaussian numerical integration

r	ξ_i	w_i	Error	Maximum degree of polynomials integrated precisely
5	0	0.56888 888888 888889 (128/225)		
	$\pm 0.53846 93101 05683 \left(\pm \frac{1}{3} \sqrt{5 - 4 \sqrt{5/14}} \right)$	$0.47862 86704 99366 \left(\frac{161}{450} + \frac{13}{180\sqrt{5/14}} \right)$	$\approx 0.8 \times 10^{-9} \frac{d^{10}y}{d\xi^{10}}$	9
	$\pm 0.90617 98459 38664 \left(\pm \frac{1}{3} \sqrt{5 + 4 \sqrt{5/14}} \right)$	$0.23692 68850 56189 \left(\frac{161}{450} - \frac{13}{180\sqrt{5/14}} \right)$		
6	$\pm 0.23861 91860 83197$ $\pm 0.66120 93864 66265$ $\pm 0.93246 95142 03152$	0.46791 39345 72691 0.36076 15730 48139 0.17132 44923 79170	$\approx 1.5 \times 10^{-12} \frac{d^{12}y}{d\xi^{12}}$	11
7	0 $\pm 0.40584 51513 77397$ $\pm 0.74153 11855 99394$ $\pm 0.94910 79123 42759$	0.41795 91836 73469 0.38183 00505 05119 0.27970 53914 89277 0.12948 49661 68870	$\approx 2.1 \times 10^{-15} \frac{d^{14}y}{d\xi^{14}}$	13

Figure 5.2. (Continued)

we have: $w_1 = w_2 = \frac{1}{6}$, $w_3 = \frac{5}{6}$, $\xi_3 = -\xi_4 = 1/\sqrt{5}$.

Five-point integration (order 7):

The unknowns are $w_1, w_2, (w_3, \xi_3), (w_4, \xi_4)$ and (w_5, ξ_5) where $\xi_1 = -1, \xi_2 = 1$.

For reasons of symmetry: $w_1 = w_2, w_4 = w_5, \xi_4 = -\xi_5$ and $\xi_3 = 0$.

The system to solve is written as:

$$w_1 + w_2 + w_3 + w_4 + w_5 = 2,$$

$$w_1 + w_4 \xi_4^2 = \frac{1}{3},$$

$$w_1 + w_4 \xi_4^4 = \frac{1}{5},$$

$$w_1 + w_4 \xi_4^6 = \frac{1}{7},$$

That is: $w_1 = w_2 = \frac{1}{10}, w_3 = \frac{32}{45}, w_4 = w_5 = \frac{49}{90}, \xi_4 = -\xi_5 = \sqrt{3/7}$.

The abscissas ξ_i , which are the results of (5.7), are also the roots of the r -order Legendre polynomial (see [DAV 75], p. 88):

$$P_r(\xi) = 0$$

defined by the recurrence formula:

$$\begin{aligned} P_0(\xi) &= 1 \\ P_1(\xi) &= \xi \\ P_k(\xi) &= \frac{2}{k} \frac{k-1}{k} \xi P_{k-1}(\xi) - \frac{k-1}{k} P_{k-2}(\xi); \quad k = 2, 3, \dots, r. \end{aligned} \tag{5.8}$$

The weights w_i are written as:

$$w_i = \frac{2(1-\xi_i^2)}{[r P_{r-1}(\xi_i)]^2}; \quad i = 1, 2, \dots, r. \tag{5.9a}$$

The integration error is of the form:

$$e = \frac{2^{2r+1} (r!)^4}{(2r+1)[(2r)!]^3} \frac{d^{2r} \gamma}{d\xi^{2r}}. \tag{5.9b}$$

EXAMPLE 5.3. Calculation of the coefficients with the two-point Gaussian method (Legendre polynomials)

Let us go back to the values of ξ_i and w_i obtained in Example 5.1 using the second-order Legendre polynomial. If $r = 2$, the polynomials (5.8) are:

$$\begin{aligned} P_0 &= 1 \\ P_1 &= \xi \\ P_2 &= \frac{3}{2}\xi^2 - \frac{1}{2}. \end{aligned}$$

The roots of $P_2 = 0$ are:

$$\xi_i = \pm \frac{1}{\sqrt{3}}.$$

According to (5.9a), the roots are:

$$w_i = \frac{2\left(1 - \frac{1}{3}\right)}{\left(2 \cdot \frac{1}{\sqrt{3}}\right)^2} = 1.$$

Now let us use the two-point Gaussian method to integrate the polynomial $y = 1 + \xi^2 + \xi^3 + \xi^4$:

$$l_{\text{app}} = \left(1 + \frac{1}{3} + \frac{1}{3\sqrt{3}} + \frac{1}{9}\right) + \left(1 + \frac{1}{3} - \frac{1}{3\sqrt{3}} + \frac{1}{9}\right) = \frac{26}{9}.$$

From (5.9b), the integration error is:

$$e = \frac{1}{135} \frac{d^4 y}{d\xi^4} = \frac{1}{135} \cdot 24 = \frac{8}{45}.$$

Indeed:

$$\left| l_{\text{app}} - l_{\text{exact}} \right| = \left| \frac{26}{9} - \int_{-1}^1 y(\xi) d\xi \right| = \left| \frac{130}{45} - \frac{138}{45} \right| = \frac{8}{45} \leq e.$$

Figure 5.2 gives coefficients w_i and ξ_i for one-, two- and seven-points integrations [KOP 61]. The abscissas ξ_i are symmetrical around $\xi = 0$; the weights w_i corresponding to two symmetrical points are equal.

EXAMPLE 5.4. Integration of $[k]$ and $\{f\}$ using the one-dimensional Gaussian method

Let us use the two-point Gaussian method to integrate the expressions (5.1b):

$$\begin{aligned}[k] &\approx [B_\delta(\xi_1)]^T [H] \cdot w_1 \cdot \det(J(\xi_1)) [B(\xi_1)] + \\ &+ [B_\delta(\xi_2)]^T [H] \cdot w_2 \cdot \det(J(\xi_2)) [B(\xi_2)]\end{aligned}$$

$$\begin{aligned}\{f_V\} &\approx \{N(\xi_1)\} (w_1 f_V \cdot \det(J(\xi_1))) + \\ &+ \{N(\xi_2)\} (w_2 f_V \cdot \det(J(\xi_2)))\end{aligned}$$

where

$$\xi_1 = -\xi_2 = \frac{1}{\sqrt{3}}; \quad w_1 = w_2 = 1.$$

It is also possible to construct numerical integration methods that involve a weighting function $p(\xi)$:

$$\int_{-1}^1 p(\xi) \gamma(\xi) d\xi = \sum_{i=1}^r w_i \gamma(\xi_i). \quad (5.10a)$$

For example, the Gauss–Jacobi method corresponds to:

$$p(\xi) = (1 - \xi). \quad (5.10b)$$

By choosing $p(\xi)$ in the form of $\frac{1}{(\xi - \xi_0)^\alpha}$, we can construct methods appropriate to the integration of singular functions at point ξ_0 .

5.1.2.2 Newton–Cotes method [DAV 75; STR 71]

If the abscissas ξ_i of the integration points are specified *a priori*, we are left with r coefficients w_1, \dots, w_r still to be determined so that (5.5) exactly integrates an $r - 1$ -order polynomial. In the Newton–Cotes method the points ξ_i are regularly spaced and symmetrical around $\xi = 0$:

$$\xi_i = 2 \frac{i-1}{r-1} - 1, \quad i = 1, 2, \dots, r. \quad (5.11a)$$

To calculate the coefficients w_i , let us represent $\gamma(\xi)$ by an $r - 1$ -order Lagrange polynomial that takes the values $\gamma(\xi_i)$ at the r integration points ξ_i . The Lagrange interpolation functions N_i were calculated in section 2.2.2.3:

$$\gamma(\xi) = \sum_{i=1}^r N_i(\xi) \gamma(\xi_i). \quad (5.11b)$$

Thus:

$$\int_{-1}^1 \gamma(\xi) = \sum_{i=1}^r \left(\int_{-1}^1 N_i(\xi) d\xi \cdot \gamma(\xi_i) \right) = \sum_{i=1}^r w_i \gamma(\xi_i).$$

The weights w_i are therefore the integrals of the Lagrange interpolation functions N_i :

$$w_i = \int_{-1}^1 N_i(\xi) d\xi. \quad (5.11c)$$

The weights corresponding to two symmetrical points with respect to $\xi = 0$ are equal.

The integration error is of the form:

$$\begin{aligned} e &= C_r \left(\frac{2}{r-1} \right)^{r+2} \frac{d^{r+1} \gamma}{d\xi^{r+1}} \quad \text{if } r \text{ is odd} \\ e &= C_r \left(\frac{2}{r-1} \right)^{r+1} \frac{d^r \gamma}{d\xi^r} \quad \text{if } r \text{ is even.} \end{aligned} \quad (5.12)$$

It is therefore preferable to use an odd number of integration points. The coefficient C_r can be calculated by integrating the approximation error of the Lagrange interpolation given by (1.68).

Figure 5.3 shows the coefficients of the Newton–Cotes method for two, three, ..., seven and nine points.

For a given number of integration points, the maximum degree of the polynomials integrated exactly by the Newton–Cotes method is much lower than that obtained by the Gaussian method (see comparisons in Figure 5.4). However, the former method sometimes allows the integration points and the interpolation points to coincide. The integration of terms containing the interpolation functions N_i is then simplified since N_i is zero for all integration points other than ξ_i . This technique can be used for “mass matrices”:

$$[m] = \int_{V_e} \{N\} \langle N \rangle dV$$

and for load vectors:

$$\{f\} = \int_{V_e} \{N\} f_V dV$$

r	ξ_i	w_i	Error	Maximum degree of polynomials integrated precisely
2	± 1	1	$\frac{1}{6} \frac{d^2 \gamma}{d\xi^2}$	1
3	0	$4/3$	$\frac{1}{90} \frac{d^4 \gamma}{d\xi^4}$	3
	± 1	$1/3$		
4	$\pm 1/3$	$3/4$	$\frac{2}{405} \frac{d^4 \gamma}{d\xi^4}$	3
	± 1	$1/4$		
5	0	$12/45$	$\approx 0.6 \times 10^{-5} \frac{d^6 \gamma}{d\xi^6}$	5
	$\pm 1/2$	$32/45$		
	± 1	$7/45$		
6	$\pm 1/5$	$50/144$	$\approx 3.7 \times 10^{-5} \frac{d^6 \gamma}{d\xi^6}$	5
	$\pm 3/5$	$75/144$		
	± 1	$19/144$		
7	0	$272/420$	$\approx 3.2 \times 10^{-7} \frac{d^8 \gamma}{d\xi^8}$	7
	$\pm 1/3$	$27/420$		
	$\pm 2/3$	$216/420$		
	± 1	$41/420$		
9	0	$-4,540/14,175$	$\approx 1.2 \times 10^{-9} \frac{d^{10} \gamma}{d\xi^{10}}$	9
	$\pm 1/4$	$10,496/14,175$		
	$\pm 1/2$	$-928/14,175$		
	$\pm 3/4$	$5,888/14,175$		
	± 1	$989/14,175$		

$$\int_{-1}^1 \gamma(\xi) d\xi = \sum_{i=1}^r w_i \gamma(\xi_i)$$

Figure 5.3. One-dimensional Newton–Cotes numerical integration

		Number of integration points r				Newton–Cotes					Gauss				
		2	3	4	5	1	2	3	4	5					
Monomials	Exact integral														
1	$\int_{-1}^1 1 \cdot d\xi = 2$	2	2	2	2	2	2	2	2	2	2	2	2	2	2
ξ	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ξ^2	2/3	2	2/3	2/3	2/3	0	2/3	2/3	2/3	2/3					
ξ^3	0	0	0	0	0	0	0	0	0	0					
ξ^4	2/5	2	2/3	14/27	2/5	0	2/9	2/5	2/5	2/5					
ξ^5	0	0	0	0	0	0	0	0	0	0					
ξ^6	2/7	2	2/3	122/243	1/3	0	2/27	6/25	2/7	2/7					

Figure 5.4. Comparison of numerical integrations of one-dimensional monomials, between -1 and 1 , using the Newton–Cotes and Gaussian methods

Remarks

Numerical integration methods have been discussed for the reference element. It is easy to transpose them into the real element using (1.44).

$$\int_{x_1}^{x_2} y(x) dx = \frac{x_2 - x_1}{2} \int_{-1}^1 y(\xi) d\xi \quad (5.13)$$

where

$$x = \frac{1-\xi}{2} x_1 + \frac{1+\xi}{2} x_2.$$

The r -point Gauss method precisely integrates a polynomial of order $2r - 1$. The r -points Newton–Cotes method precisely integrates a polynomial of order $r - 1$. In practice, we need to integrate high-order polynomials or non-polynomial functions (rational fractions). By the previous methods, we then obtain an approximate integration that is more accurate when the number of integration points is high.

EXAMPLE 5.5. *Integration of a non-polynomial function using the Gaussian and Newtonian methods*

The exact value of

$$l = \int_{-1}^1 \frac{1}{1+\xi^2} d\xi$$

is

$$l = \frac{\pi}{2} \approx 1.5708.$$

The Gaussian and Newton–Cotes methods give the following values of l :

Number of points	Gauss		Newton–Cotes	
	value	relative error (%)	value	relative error (%)
1	2	27	2	27
2	1.5	4.5	1	36
3	1.5833	0.8	1.6666	6
4	1.5686	0.1	1.6	2

Rather than using an integration method with many integration points, we can split the integration domain into several subdomains; we then use a simple integration method in each subdomain. This technique is particularly useful for continuous functions $\gamma(\xi)$ in the entire subregion.

5.1.3 TWO-DIMENSIONAL NUMERICAL INTEGRATION [DAV 75; STR 71]

a) “Product” methods

These methods involve using a one-dimensional numerical integration in both directions ξ and η . If we use r_1 points in direction ξ and r_2 points in direction η , the Gaussian method precisely integrates the product of a $2r_1 - 1$ -order polynomial in ξ and a $2r_2 - 1$ -order polynomial in η .

The “product” method uses $r = r_1 \cdot r_2$ points; it integrates all monomials

$$\begin{aligned} \xi^i \eta^j \text{ such that } 0 \leq i \leq 2r_1 - 1 \\ 0 \leq j \leq 2r_2 - 1. \end{aligned}$$

b) Direct methods

It is also possible to directly extend the methods from section 5.1.2 to two-dimensional space:

$$\iint_V y(\xi, \eta) d\xi d\eta = \sum_{i=1}^r w_i y(\xi_i, \eta_i). \quad (5.14)$$

In particular, we can construct Gaussian methods, which precisely integrate the m -order monomials:

$$\xi^i \eta^j \text{ such that } i + j \leq m.$$

Such methods often use fewer points than “product” methods. They are discussed in detail by Stroud [STR 71].

For square reference elements, the “product” methods are most often used, while for the triangular elements, direct methods are more common.

5.1.3.1 Square reference element

The “product” method is written as:

$$\int_{-1}^1 \int_{-1}^1 y(\xi, \eta) d\xi d\eta = \sum_{i=1}^{r_1} \sum_{j=1}^{r_2} w_i w_j y(\xi_i, \eta_j) \quad (5.15)$$

where: w_i, w_j are the coefficients given in Figures 5.2 (Gauss) or 5.3 (Newton–Cotes);

ξ_i, η_j are the coordinates of the corresponding integration points.

Figure 5.5 shows some formulae based on the direct method.

EXAMPLE 5.6. Integration of $[k]$ using the two-dimensional Gaussian method and organization of the calculations

Let us use the ($r = 2 \times 2$ point) Gaussian “product” method to integrate the expression (5.1b) of $[k]$:

$$\begin{aligned}
[k] \approx & [B_\delta(\xi_1 \eta_1)]^T \cdot ([H] \cdot (w_1 \cdot w_1 \cdot \det(J(\xi_1 \eta_1)))) [B(\xi_1 \eta_1)] + \\
& + [B_\delta(\xi_1 \eta_2)]^T \cdot ([H] \cdot (w_1 \cdot w_2 \cdot \det(J(\xi_1 \eta_2)))) [B(\xi_1 \eta_2)] + \\
& + [B_\delta(\xi_2 \eta_1)]^T \cdot ([H] \cdot (w_2 \cdot w_1 \cdot \det(J(\xi_2 \eta_1)))) [B(\xi_2 \eta_1)] + \\
& + [B_\delta(\xi_2 \eta_2)]^T \cdot ([H] \cdot (w_2 \cdot w_2 \cdot \det(J(\xi_2 \eta_2)))) [B(\xi_2 \eta_2)].
\end{aligned}$$

where

$$w_1 = w_2 = 1$$

$$\xi_1 = \eta_1 = -\xi_2 = -\eta_2 = \frac{1}{\sqrt{3}}.$$

Let us note that, for reasons of efficiency, it is preferable to multiply the matrix $[H]$ by the scalar $w_i w_j \det(J)$, rather than multiply $[B_\delta]^T [H] [B]$ by this scalar. Indeed, $[H]$ is smaller than $[k]$.

Let us consider an eight-node element corresponding to the Laplace equation. $[B]$ measures 2×8 and $[H]$ is 2×2 and symmetrical. Then for each integration point, we need a number of multiplications equal to:

$$\begin{array}{ll}
2 & \text{for } c = w_1 \cdot w_2 \cdot \det(J) \\
+ 3 & \text{for } [H \cdot c] \\
+ 32 & \text{for } [H \cdot c] [B] \\
+ 72 & \text{for } [B]^T ([H \cdot c] [B]) \text{ (symmetrical)} \\
\hline
109 & \text{in Total}
\end{array}$$

Thus, for four integration points we need $4 \times 109 = 436$ multiplications, regardless of how $[B]$, $[H]$ and $\det(J)$ are constructed. The number of operations may be reduced if we take account of the zeros present in $[B]$ and $[H]$.

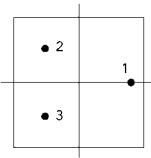
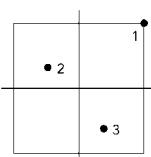
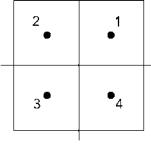
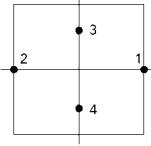
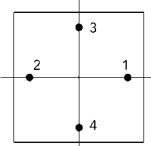
Furthermore, given that the matrix $[H]$ is positive definite, we can decompose it into upper and lower triangles (5.38):

$$[H] = [L] [L]^T$$

then calculate:

$$([L]^T [B])^T ([L]^T [B]).$$

This further reduces the number of operations needed.

	Order m	Number of points r	Coordinates ξ_i, η_i	Weight w_i
	2	3	$\sqrt{2/3}$ $-\frac{1}{\sqrt{6}}$	$4/3$ $4/3$
	2	3	1 $-5/9$ $1/3$	$4/7$ $27/14$ $3/2$
	3	4 2×2 -point product method	$\pm \frac{1}{\sqrt{3}}$	$\pm \frac{1}{\sqrt{3}}$
	3	4	± 1 0	0 $\pm \frac{1}{\sqrt{2}}$
	3	4	$\pm \sqrt{2/3}$ 0	0 $\pm \sqrt{2/3}$

$$\int_{-1}^1 \int_{-1}^1 y(\xi, \eta) d\xi d\eta \approx \sum_{i=1}^r w_i y(\xi_i, \eta_i).$$

Formulas integrating exactly the polynomials of order m
(including monomials $\xi^i \eta^j$ such as $i + j \leq m$).

Figure 5.5. Direct integration formulae on a square

	Order m	Number of points r	Coordinates ξ_i, η_i	Weight w_i
	5	7	0 0 0 $\pm\sqrt{14/15}$ $\pm\sqrt{3/5}$ $\pm\sqrt{3/5}$	$8/7$ $20/63$ $20/36$
	5	7	0 0 $-r$ $-r$ $+r$ $+r$ $+s$ $-t$ $-s$ $+t$ $+t$ $-s$ $-t$ $+s$	$\left. \begin{array}{l} 8 \\ 7 \end{array} \right\} 100/168$ $\left. \begin{array}{l} 20/48 \end{array} \right\}$
			$r = \sqrt{7/15} \approx 0.683$ $s = \sqrt{\frac{7 + \sqrt{24}}{15}} \approx 0.89$ $t = \sqrt{\frac{7 - \sqrt{24}}{15}} \approx 0.374$	

Figure 5.5. (Continued)

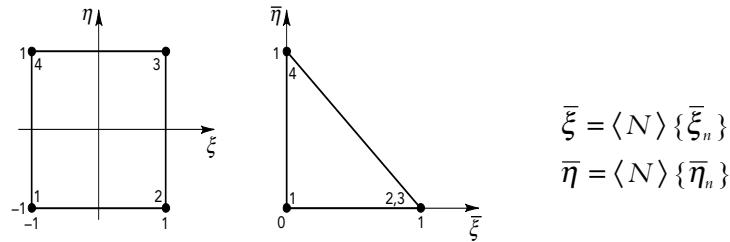
5.1.3.2 Triangular reference element

a) Gauss–Radau method [IRO 66]

The “product” method consists of first transforming the integral on the triangle:

$$I = \int_0^1 \int_0^1 u(\bar{\xi}, \bar{\eta}) d\bar{\eta} d\bar{\xi} \quad (5.16)$$

into an integral on a square (5.15) by changing the variables. Let us use the geometric transformation of the points (ξ, η) of the square element into points $(\bar{\xi}, \bar{\eta})$ of the triangular element:



where

$$\begin{aligned}\langle \bar{\xi}_n \rangle &= \langle 0 \ 1 \ 1 \ 0 \rangle \\ \langle \bar{\eta}_n \rangle &= \langle 0 \ 0 \ 0 \ 1 \rangle.\end{aligned}$$

$\langle N \rangle$ are the functions given in Example 1.16.

Thus:

$$\begin{aligned}\bar{\xi} &= \frac{1+\xi}{2} \\ \bar{\eta} &= \frac{1-\xi}{2} \ \frac{1+\eta}{2} \\ d\bar{\xi} \ d\bar{\eta} &= \det(J) d\xi \ d\eta = \frac{1-\xi}{2} d\xi \ d\eta.\end{aligned}\tag{5.17}$$

The integral (5.16) becomes:

$$l = \frac{1}{2} \int_{-1}^1 (1-\xi) \int_{-1}^1 y(\bar{\xi}(\xi), \bar{\eta}(\xi, \eta)) d\eta \ d\xi.$$

Let us use the Gaussian method to integrate numerically in η :

$$l = \int_{-1}^1 (1-\xi) \left[\sum_{j=1}^{r_2} \frac{w_j}{2} y(\bar{\xi}(\xi), \bar{\eta}(\xi, \eta_j)) \right] d\xi.\tag{5.18}$$

Let us now integrate in ξ using the Gauss–Jacobi method (5.10b) whose weights are w'_i and the abscissas of the integration points are ξ'_i :

$$l = \sum_{i=1}^{r_1} \sum_{j=1}^{r_2} w'_i \frac{w_j}{2} y(\bar{\xi}(\xi'_i), \bar{\eta}(\xi'_i, \eta_j)).\tag{5.19}$$

This method, called the Gauss–Radau method, is rarely used because its integration points are not localized in such a way as to respect the symmetry of the triangle. On the other hand, it can be effective when the variations $y(\xi, \eta)$ are very great in the vicinity of the node A of the triangle.

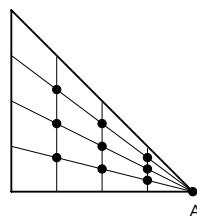


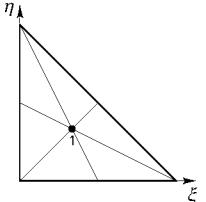
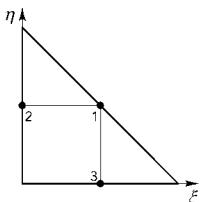
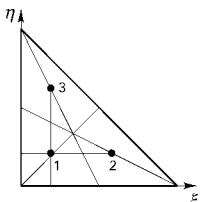
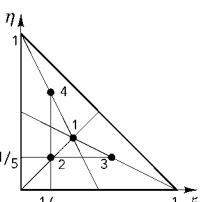
Figure 5.6 gives the weight and coordinates of the integration points of the Gauss–Radau method.

Integration order in ξ^i or η^j	Number of points $r \times r$	RI	WI	SJ	AJ
1	1×1	0.5	1.0	0.3333333333	0.75
3	2×2	0.2113248654 0.7886751346	0.5 0.5	0.1550510257 0.6449489743	0.3764030627 0.5124858262
5	3×3	0.1127016654 0.5 0.8872983346	0.2777777778 0.4444444444 0.2777777778	0.0885879595 0.4094668644 0.7876594618	0.2204622112 0.3881934688 0.3288443200
7	4×4	0.0694318442 0.3300094782 0.6699905218 0.9305681558	0.1739274226 0.3260725774 0.3260725774 0.1739274226	0.0571041961 0.2768430136 0.5835904324 0.8602401357	0.1437135608 0.2813560151 0.3118265230 0.2231039011
9	5×5	0.0469100770 0.2307653449 0.5 0.7692346551 0.9530899230	0.1184634425 0.2393143353 0.2844444444 0.2393143353 0.1184634425	0.0398098571 0.1980134179 0.4379748102 0.6954642734 0.9014649142	0.1007941926 0.2084506672 0.2604633916 0.2426935942 0.1598203766
$\int_0^1 \int_0^{1-\xi} \gamma(\xi, \eta) d\eta d\xi = \sum_{j=1}^r \sum_{i=1}^r WJ(j) \cdot WI(i) \cdot \gamma(\xi_j, \eta_{ij})$					
where $WJ(j) = AJ(j) (1 - SJ(j))$ $\xi_j = SJ(j)$ $\eta_{ij} = RI(i) (1 - SJ(j)).$					
$RI \text{ and } WI \text{ are the coefficients of the Gaussian numerical integration at interval } (0,1):$ $\int_0^1 \gamma(\xi) d\xi = \sum_{i=1}^r WI(i) \gamma(RI(i)).$					

Figure 5.6. “Product” formula for integration on a triangle (Gauss–Radau)

b) Direct method [STR 71; HAM 56]

Figure 5.7 gives the formulae of order $m = 1, 2, \dots, 6$, which precisely integrate monomials $\xi^i \eta^j$ for which $i + j \leq m$. These formulae are often called “Hammer” formulae.

	Order m	Number of points r	Coordinates		Weight w_i
			ξ_i	η_i	
	1	1	1/3	1/3	1/2
	2	3	1/2 0 1/2	1/2 1/2 0	1/6
	2	3	1/6 2/3 1/6	1/6 1/6 2/3	1/6
	3	4	1/3 1/5 3/5 1/5	1/3 1/5 1/5 3/5	-27/96 25/96

$$\int_0^1 \int_0^{1-\xi} y(\xi, \eta) d\eta d\xi \approx \sum_{i=1}^r w_i y(\xi_i, \eta_i).$$

Formulas precisely integrating m -order polynomials
 $(\xi^i \eta^j \text{ where } i + j \leq m)$.

Figure 5.7. Direct formulae for integration over a triangle (Hammer)

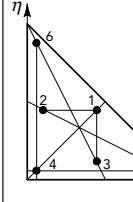
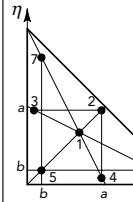
	Order m	Number of points r	Coordinates		Weight w_i
			ξ_i	η_i	
	4	6	a $1 - 2a$ a b $1 - 2b$ b	a a $1 - 2a$ b b $1 - 2b$	0.111690794839005 0.054975871827661
	5	7	$1/3$ a $1 - 2a$ a b $1 - 2b$ b	$1/3$ a a $1 - 2a$ b b $1 - 2b$	$\frac{9}{80}$ $A = \frac{155 + \sqrt{15}}{2400}$ $= 0.0661970763942530$ $\frac{31}{240} - A =$ $= 0.0629695902724135$
	6	12	a $1 - 2a$ a b $1 - 2b$ b c d $1 - (c+d)$ $1 - (c+d)$ c d	a a $1 - 2a$ b b $1 - 2b$ d c c c $1 - (c+d)$ $1 - (c+d)$	0.025422453185103 0.058393137863189 0.041425537809187

Figure 5.7. (Continued)

5.1.4 NUMERICAL INTEGRATION IN THREE DIMENSIONS [STR 71; HAM 56; IRO 71; HEL 72]

5.1.4.1 Cubic reference element

The “product” method is written as:

$$\int_{-1}^1 \int_{-1}^1 \int_{-1}^1 y(\xi, \eta, \zeta) d\xi d\eta d\zeta = \sum_{i=1}^{r_1} \sum_{j=1}^{r_2} \sum_{k=1}^{r_3} w_i w_j w_k y(\xi_i, \eta_j, \zeta_k) \quad (5.20)$$

where w_i , w_j and w_k are the coefficients from Figure 5.2 (Gauss) or 5.3 (Newton–Cotes);

ξ_i, η_j, ζ_k are the coordinates of the integration points in Figures 5.2 and 5.3.

The $r_1 \times r_2 \times r_3$ -point Gaussian method precisely integrates monomials $\xi^i \eta^j \zeta^k$ such that $i \leq 2 r_1 - 1, j \leq 2 r_2 - 1, k \leq 2 r_3 - 1$.

Order m	Number of points r	Coordinates ξ_i η_i ζ_i	Weight w_i
2	4	$0 \quad \pm \sqrt{\frac{2}{3}} \quad -\frac{1}{\sqrt{3}}$ $\pm \sqrt{\frac{2}{3}} \quad 0 \quad \frac{1}{\sqrt{3}}$	2
3	6	$\frac{1}{\sqrt{6}} \quad \pm \frac{1}{\sqrt{2}} \quad -\frac{1}{\sqrt{3}}$ $-\frac{1}{\sqrt{6}} \quad \pm \frac{1}{\sqrt{2}} \quad \frac{1}{\sqrt{3}}$ $-\sqrt{\frac{2}{3}} \quad 0 \quad -\frac{1}{\sqrt{3}}$ $\sqrt{\frac{2}{3}} \quad 0 \quad \frac{1}{\sqrt{3}}$	$\frac{4}{3}$
3	6	$\pm 1 \quad 0 \quad 0$ $0 \quad \pm 1 \quad 0$ $0 \quad 0 \quad \pm 1$	$\frac{4}{3}$
5	14	$\pm a \quad 0 \quad 0$ $0 \quad \pm a \quad 0$ $0 \quad 0 \quad \pm a$ $\pm b \quad \pm b \quad \pm b$ $a = \sqrt{\frac{19}{30}} \quad b = \sqrt{\frac{19}{33}}$	$\frac{320}{361} \quad \frac{121}{361}$
7	34	$\pm a \quad 0 \quad 0$ $0 \quad \pm a \quad 0$ $0 \quad 0 \quad \pm a$ $\pm a \quad \pm a \quad 0$ $0 \quad \pm a \quad \pm a$ $\pm a \quad 0 \quad \pm a$ $\pm b \quad \pm b \quad \pm b$ $\pm c \quad \pm c \quad \pm c$	0.295747599451303 0.094101508916324 0.412333862271436 0.224703174765601

$$\int_{-1}^1 \int_{-1}^1 \int_{-1}^1 y(\xi, \eta, \zeta) d\xi d\eta d\zeta = \sum_{i=1}^r w_i y(\xi_i, \eta_i, \zeta_i).$$

Figure 5.8. Formulae for direct integration over a cube

Figure 5.8 shows some formulas of order $m = 2, 3, 5$ and 7 obtained by the direct method, which precisely integrate the monomials $\xi^i \eta^j \zeta^k$ such that $i + j + k = m$.

5.1.4.2 Tetrahedral reference element

The “product” method, which is an extension of the Gauss–Radau method into three-dimensional space, is rarely used.

Figure 5.9 shows formulas of orders $m = 1, 2, 3$ and 5 obtained by the direct method.

Order m	Number of points r	Coordinates			Weight ω_i
		ξ_i	η_i	ζ_i	
1	1	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{6}$
2	$a = \frac{4}{5 - \sqrt{5}}$ $b = \frac{5 + 3\sqrt{5}}{20}$	a a a b a b	a a b a a a	a a b a a a	$\frac{1}{24}$
3	$a = \frac{1}{4}$ $b = \frac{1}{6}$ $c = \frac{1}{2}$	a b b c b c b	a b b c b c b	a b b c b c b	$-\frac{2}{15}$ $\frac{3}{40}$

$$\int_0^1 \int_0^{1-\xi} \int_0^{1-\xi-\eta} y(\xi, \eta, \zeta) d\xi d\eta d\zeta = \sum_{i=1}^r \omega_i y(\xi_i, \eta_i, \zeta_i)$$

Figure 5.9. Formulae for direct integration over a tetrahedron

Order m	Number of points r	Coordinates			Weight ω_i
		ξ_i	η_i	ζ_i	
4	11	a	a	a	$-\frac{74}{5625}$
		$\frac{11}{14}$	$\frac{1}{14}$	$\frac{1}{14}$	
		$\frac{1}{14}$	$\frac{11}{14}$	$\frac{1}{14}$	
		$\frac{1}{14}$	$\frac{1}{14}$	$\frac{11}{14}$	
		$\frac{1}{14}$	$\frac{1}{14}$	$\frac{1}{14}$	
		a	a	a	
		a	b	a	
		a	b	b	
		b	a	a	
		b	b	a	
		b	a	b	
5	15	a	a	a	$\frac{112}{5670}$
		b_i	b_i	b_i	
		c_i	b_i	b_i	
		b_i	c_i	b_i	
		b_i	b_i	c_i	
		e	d	d	
		d	e	d	
		d	d	e	
		d	e	e	
		e	d	e	
		e	e	d	

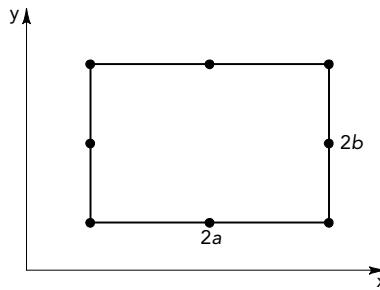
$$\int_0^1 \int_0^{1-\xi} \int_0^{1-\xi-\eta} \gamma(\xi, \eta, \zeta) d\xi d\eta d\zeta = \sum_{i=1}^r \omega_i \gamma(\xi_i, \eta_i, \zeta_i)$$

Figure 5.9. (Continued)

5.1.5 PRECISION OF INTEGRATION

The exact integration of elementary matrices and load vectors requires the exact integration of each of their terms. This is only possible, with the integration methods discussed above, if these terms are polynomials, which is usually the case when the Jacobian matrix is constant.

EXAMPLE 5.7. *Choice of number of integration points for an eight-node rectangular element corresponding to the Laplace equation*



$$[k] = \int_{-1}^1 \int_{-1}^1 [B]^T [H] [B] \det(J) d\xi d\eta$$

$$[B] = \frac{1}{\det(J)} \begin{bmatrix} b \langle \frac{\partial N}{\partial \xi} \rangle \\ a \langle \frac{\partial N}{\partial \eta} \rangle \end{bmatrix}; [H] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The functions $\langle N \rangle$ are those from section 2.4.3.2; they contain the eight monomials:

$$1 \quad \xi \quad \eta \quad \xi^2 \quad \xi\eta \quad \eta^2 \quad \xi^2\eta \quad \xi\eta^2.$$

The functions $\frac{\partial N}{\partial \xi}$ and $\frac{\partial N}{\partial \eta}$ therefore contain the monomials:

For $\frac{\partial N}{\partial \xi}$: 0 1 0 2\xi \eta 0 2\xi\eta \eta^2

For $\frac{\partial N}{\partial \eta}$: 0 0 1 0 \xi 2\eta \xi^2 2\xi\eta

The product $[B]^T [H] [B]$ therefore contains all monomials to the fourth order ($\xi^i \eta^j$; $i + j \leq 4$). With the Gaussian integration method, we need 3×3 points to precisely integrate $[k]$. Very often we use a 2×2 -point integration for this element which gives improved overall results. This is called reduced integration scheme.

Consider a load vector of the form:

$$\{f\} = \int_{-1}^1 \int_{-1}^1 \{N\} f \det(J) d\xi d\eta.$$

If f is constant, we must use 2×2 Gaussian integration points or a direct third-order method to integrate exactly.

For a load vector corresponding to a given f on the side $\xi = 1$ of the element, which is the same as the boundary of the domain:

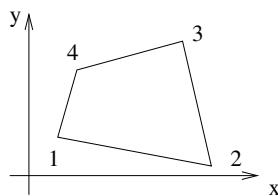
$$\{f\} = \int_{-1}^1 \{N(\xi=1, \eta)\} f b d\eta.$$

In this case, it is sufficient to use two Gaussian points.

If the element is deformed (quadrilaterals, curvilinear sides), the geometric transformation is not linear and the Jacobian matrix is a polynomial function of ξ . The terms to be integrated to obtain $[k]$ are rational fractions. It is no longer possible to integrate these terms exactly. For a given number of integration points, the precision of integration decreases as the deformation of the element increases. Indeed, we can develop the inverse of the denominator into an infinite series, which must be truncated to a greater order when the deformation is great. Each term to be integrated is then the polynomial product of the numerator by the truncated series.

EXAMPLE 5.8. Expression of $[k]$ for a deformed element

Let us deform the element in the previous example with a transformation in which the Jacobian matrix is a linear function of ξ, η .



The Jacobian matrix is given in Example 1.18:

$$[J] = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} = \begin{bmatrix} a_1 + b_1 \eta & a_2 + b_2 \eta \\ a_3 + b_1 \xi & a_4 + b_2 \xi \end{bmatrix}$$

$$\det(J) = A_0 + A_1 \xi + A_2 \eta.$$

Then:

$$[B] = \begin{bmatrix} \langle \frac{\partial N}{\partial x} \rangle \\ \langle \frac{\partial N}{\partial y} \rangle \end{bmatrix} = \frac{1}{\det(J)} \begin{bmatrix} J_{22} \langle \frac{\partial N}{\partial \xi} \rangle - J_{12} \langle \frac{\partial N}{\partial \eta} \rangle \\ -J_{21} \langle \frac{\partial N}{\partial \xi} \rangle + J_{11} \langle \frac{\partial N}{\partial \eta} \rangle \end{bmatrix}$$

$$= \frac{1}{\det(J)} [B']$$

$$[k] = \int_{-1}^1 \int_{-1}^1 \frac{1}{\det(J)} [B']^T [H] [B'] d\xi d\eta.$$

The terms of $[k]$ are rational fractions: the numerator contains terms up to ξ^4 and η^4 ; the denominator is linear in ξ, η . The term $\frac{1}{\det(J)}$ can be developed as follows:

$$\frac{1}{\det(J)} = \frac{1}{A_0 + A_1 \xi + A_2 \eta} = \frac{1}{A_0} \left(1 + \frac{A_1}{A_0} \xi + \frac{A_2}{A_0} \eta \right)^{-1}$$

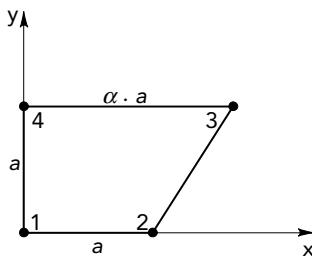
$$= \frac{1}{A_0} \left(1 - \frac{A_1}{A_0} \xi - \frac{A_2}{A_0} \eta + \dots \right).$$

The precision of the numerical integration is influenced by $\frac{A_1}{A_0}$ and $\frac{A_2}{A_0}$, that characterize the deformation of the element.

The following example shows the influence of the deformation of the element on the precision of the numerical integration.

EXAMPLE 5.9. Numerical integration over a four-node trapezoidal element

Consider the following element:



The Jacobian matrix of the geometric transformation, given in Example 1.18, is written as:

$$\det(J) = A_0 + A_1 \xi + A_2 \eta$$

$$A_0 = a^2 \frac{\alpha+1}{8}$$

$$A_1 = 0$$

$$A_2 = a^2 \frac{\alpha-1}{8}.$$

Integrate numerically the following expression:

$$\begin{aligned} l &= \int_{-1}^1 \int_{-1}^1 \frac{1}{\det(J)} d\xi d\eta = \frac{16}{a^2} \int_{-1}^1 \frac{1}{(\alpha+1) + (\alpha-1)\eta} d\eta \\ &= \frac{16}{a^2} \frac{1}{\alpha-1} \operatorname{Log} \alpha. \end{aligned}$$

The values $\frac{a^2 l}{16}$ are as follows, for different values of α and for a number r of integration points:

α	2	3	5	10
r				
1	0.66666	0.50	0.33333	0.18182
2	0.69231	0.54545	0.39130	0.23404
3	0.69312	0.54902	0.40067	0.24962
exact	0.69315	0.54931	0.40236	0.25584

5.1.6 CHOICE OF NUMBER OF INTEGRATION POINTS [ZIE 00; STR 73]

The choice of the number of integration points depends on the type of element being used and the elementary matrix being constructed (e.g. $[k]$ or $[m]$). In practice, we usually choose as small a number of points as possible to decrease the amount of computation necessary. Experience has shown [ZIE 00] that reduced integration can yield better results than exact integration. On the other hand, for each type of element, there is a minimum number of integration points below which the matrix $[K]$ remains singular in spite of the introduction of boundary conditions. For an isoparametric eight-node quadrilateral element (section 2.4.3.2), we need a minimum of 2×2 integration points to calculate $[k]$ and 3×3 points to calculate $[m]$. For a Hermite quadrilateral element with four nodes (section 2.4.4.2), we need 3×3 points to calculate $[k]$.

According to Zienkiewicz [ZIE 00], for isoparametric elements, the number of integration points must facilitate precise integration of $\det(J)$: that is, the volume of the real element. This criterion may be insufficient in some cases – for instance, it leads us to use a single point for the four-node isoparametric quadrilateral (section 2.4.2), which can make $[K]$ singular when the number of elements is low.

In fact, each integration point of an element has one or more associated relationships between the nodal variables of the element. In order for the global matrix $[K]$ not to be singular, the total number of integration points must be such that the number of corresponding relations is at least equal to the number of unknowns in the problem, after introduction of minimum number boundary conditions. In solid mechanics, it corresponds to rigid body motions which are six for three-dimensional problems.

Remarks

- For an exact integration, the rank of the elementary matrix $[k]$ is:

$$\rho(k) = ndl - n_R,$$

where ndl is the number of degrees of freedom and n_R the number of rigid modes.

- The number of non-zero eigenvalues of $[k]$ is equal to its rank. It has n_R null eigenvalues.
- A reduced integration of the matrix $[k]$ introduces n_p parasitic modes with null energy, that is:

$$\rho(k) = ndl - n_R - n_p.$$

The number of null eigenvalues is therefore $n_R + n_p$.

- We have to verify the absence of parasitic modes after assembling the global matrix to ensure that it is non-singular.

EXAMPLE 5.10. Reduced integration

Consider the eight-node elastic element presented in section 4.3.4.5c. 3×3 -point integration leads to a correct ranking of:

$$\rho(k) = 16 - 3(\text{rigid modes}) = 13.$$

as it involves 27 relations which are greater than the required 13 relations.

Any reduced integration involving less than 13 relations will produce parasitic modes. We have thus to verify the absence of these modes on a set of elements.

– One-point integration leads to a rank of three and the presence of 10 parasitic modes:

$$[k] = \left([B]^T [H] \boldsymbol{\omega}_i \cdot \det(J) [B] \right)_{\xi_i, \eta_i}^{3 \times 3}, \quad \text{with } \rho(k) = \rho(H).$$

– 2×2 -point integration leads to a rank of 12 and the presence of a single parasitic mode:

$$\begin{aligned} [k] &= \sum_{\xi_i, \eta_i} \left([B]^T [\bar{H}] [B] \right)^{3 \times 16}, \quad \text{with } [\bar{H}] = [H] \boldsymbol{\omega}_i \cdot \det(J_i) \\ &= [B_1^T \ B_2^T \ B_3^T \ B_4^T] \begin{bmatrix} \bar{H}_1 & & & \\ & \bar{H}_2 & & \\ & & \bar{H}_3 & \\ & & & \bar{H}_4 \end{bmatrix}_{12 \times 12} \begin{bmatrix} B_1^T \\ B_2^T \\ B_3^T \\ B_4^T \end{bmatrix}_{12 \times 16}, \quad \text{with } \rho(k) = \rho(H). \end{aligned}$$

The parasitic mode disappears after assembly of the two elements; hence, this reduced integration can be deemed sufficient.

– As for the integration of the mass matrix $[m]$, it requires a minimum of 3×3 points.

EXAMPLE 5.11. Selective reduced integration

The phenomenon of locking appears for constrained problems such as:

- thin structures (beams, plates, shells) with low or zero transverse shearing:
- incompressible fluids: $\operatorname{Div}(\vec{V}) = 0$ with \vec{V} : velocity field.

The use of selective reduced integration provides an equation without locking.

Let us consider the equation: $[k] = [k_1] + \alpha[k_c]$, where α is a very large coefficient associated with the term responsible for the locking. After assembly and introduction of boundary conditions, we obtain:

$$[K] = [K_1] + \alpha[K_c].$$

If the matrix $[K_c]$ is non-singular, a locking appears, the extent of which is directly related to the value of α . The technique consists of using a reduced integration to render $[K_c]$ singular.

Let us illustrate this approach with two examples:

a) A problem of a thin beam with shear, defined by:

$$W^e = W_f^e + W_c^e,$$

$$\text{with } W_f^e = \int_0^{L^e} \delta\beta_{,x} H_f \beta_{,x} dx \text{ and } W_c^e = \int_0^{L^e} (\delta\beta + \delta w_{,x}) H_c (\beta + w_{,x}) dx$$

$$\text{where } H_c = Gh \gg H_f = \frac{Eh^3}{12}, \quad G = \frac{E}{2(1+\nu)}.$$

E : Elastic modulus

ν : Poisson's ratio

h : thickness such that $h/L^e \ll 1$

β : rotation

w : transverse displacement (arrow)

For a two-node element, we have:

$$[k] = H_f ([k_f] + \alpha [k_c]), \text{ with } \alpha = \frac{H_c}{H_f} \gg 1.$$

$$\text{After assembly: } [K] = H_f ([K_f] + \alpha [K_c]).$$

As the matrix $[K_c]$ is non-singular, the solution is dominated by highly rigid shear term which is null for thin beams. We have thus the presence of shear locking. In order to eliminate this phenomenon, the matrix $[K_c]$ is evaluated with a reduced integration using a single point. Its rank, which is initially two, then becomes equal to one. This reproduces the mixed element with $\gamma = \beta + w_{,x} = \text{const.}$

b) A two-dimensional Stokes problem with penalty on the pressure field is defined by:

$$W^e = W_1^e + \lambda W_c^e,$$

$$\text{with } W_1^e = \int_{A^e} \langle \delta \boldsymbol{\epsilon} \rangle [H] \{ \boldsymbol{\epsilon} \} dA \text{ and } W_c^e = \int_{A^e} \text{Div}(\boldsymbol{\delta u}) \cdot \text{Div}(\boldsymbol{u}) dA.$$

By the same reasoning, it is possible to show that a 2×2 -point integration for W_1^e and a single integration point for W_c^e will reduce penalty term locking for a Q4 element.

5.1.7 NUMERICAL INTEGRATION CODES

It is desirable to organize numerical integration codes which are valid for the “product” methods or direct methods, and for the various types of one-, two- or three-dimensional elements. Let us express all the numerical integration formulae in the general form:

$$I = \sum_{l=1}^{\text{IPG}} w_l \gamma(\xi_l). \quad (5.21)$$

One-dimensional:

$$w_l = w_i, \quad \text{IPG} = r.$$

Two-dimensional (Figure 5.10):

- “product” methods: $w_l = w_i w_j, \quad \text{IPG} = r_1 \cdot r_2$
- direct methods: $w_l = w_i, \quad \text{IPG} = r.$

Three-dimensional (Figure 5.11):

- “product methods”: $w_l = w_i w_j w_k, \quad \text{IPG} = r_1 \cdot r_2 \cdot r_3$
- direct methods: $w_l = w_i, \quad \text{IPG} = r$

where: ξ_l represents the coordinates of the l th integration point;

w_l is the corresponding weight at integration point number l ;

IPG is the total number of integration points.

The calculation of the integral therefore reduces in each case to a simple loop:

```

Integral = 0
  └─ For   l = 1   at  IPG
      └─ Integral = Integral + w_l · γ(ξ_l).

```

In the programs (Figures 5.10 and 5.11), we place weights w_l in the **weighting coefficient table VCPG** of length IPG , and the coordinates ξ_l in the **integration points coordinates table VKSI** of length $\text{IPG} \times \text{NDIM}$, where NDIM represents the number of dimensions of a problem (one, two or three).

$$\begin{aligned} \text{VCPG} &= \langle w_1 \ w_2 \ \dots \ w_{\text{IPG}} \rangle \\ \text{VKSI} &= \langle \xi_1 \ \eta_1 \ \zeta_1; \ \xi_2 \ \eta_2 \ \zeta_2; \ \dots; \ \xi_{\text{IPG}} \ \eta_{\text{IPG}} \ \zeta_{\text{IPG}} \rangle. \end{aligned}$$

```
% Numerical integration for a reference triangle
% Direct symmetrical quadrature formulae
% 1 pt--order 1; 4 pts order 3; 6 pts order 4; 7 pts order 5; 12 pts order 6
%----- Input: np number of points
np = input('Give number of points (1, 3, 4, 6, 7 or 12) :\n');
%----- Output: integrated and exact value ksi^m eta^n
% m, n: polynomial order
% Vksi: abscissas ksi eta; vcog: weight
% table of values Figure 5.7
vksi1=[1/3; 1/3]; vcpg1= 1/2; % 1 point order 1
vksi3=[1/6 2/3 1/6 ; 1/6 1/6 2/3]; vcpg3=[1/6 1/6 1/6]; % 3 points order 2
%----- 4 points order 3
vksi4=[1/3 1/5 3/5 1/5; 1/3 1/5 1/5 3/5];
vcpg4=[-9/32 25/96 25/96 25/96];
%----- 6 points order 4
a= ( 8-sqrt(10) + sqrt(38-44*sqrt(2/5)) )/18;
b= ( 8-sqrt(10) - sqrt(38-44*sqrt(2/5)) )/18;
aw=( 620 + sqrt(213125-53320*sqrt(10)) )/7440;
bw=( 620 - sqrt(213125-53320*sqrt(10)) )/7440;
vksi6=[a,(1-2*a),a,b,(1-2*b),b;a,a,(1-2*a),b,b,(1-2*b)];
vcpg6=[aw,aw,aw,bw,bw,bw];
%----- 7 points order 5
a= ( 6+sqrt(15) )/21; b= ( 6-sqrt(15) )/21;
aw=(155+sqrt(15))/2400;bw=(155-sqrt(15))/2400;
vksi7=[1/3,a,(1-2*a),a,b,(1-2*b),b;1/3,a,a,(1-2*a),b,b,(1-2*b)];
vcpg7=[9/80,aw,aw,aw,bw,bw,bw];
%----- 12 points order 6
a=0.063089014491502;b=0.249286745170910;
c=0.310352451033785;d=0.053145049844816;
vksi12=[a,(1-2*a),a,b,(1-2*b),b,c,d,1-c-d,(1-c-d),c,d;
a,a,(1-2*a),b,b,(1-2*b),d,c,c,d,(1-c-d),(1-c-d)];
vcpg12=[0.025422453185103*[1 1 1],0.058393137863189*[1 1 1],...
0.041425537809187*[1 1 1 1 1]];
%----- Integration of each monomial depending on the number of points np
nmax=6;
disp('----- number of points retained: 1 3 4 6 7 12')
disp('----- integration order : 1 2 3 4 5 6 ')
if sum(np==[1 3 4 6 7 12]) == 0,
```

```

disp(' no formula for the nb of chosen points'), break, end
% fprintf(1,'pas=%5i dt= %12.6f tpas=%12.6f\n',ipas,dt,tpas)
disp('----- numerical integration with exact val ')
fprintf(1,' nb de points = %5i\n',np)
disp(' m n vnum vexact')
for i=0:nmax
    for j=0:nmax
        if(np==1 & i+j <=0)
            vnum=vcpg1; vexact=.5;
            fprintf(1,'%5i%5i %15.12f %15.12f\n',i,j,vnum,vexact)
        elseif(np==3 & i+j <=2)
            vnum=sum( (vksi3(1,:).^i).*(vksi3(2,:).^j).*vcpg3 );
            vexact=(gamma(i+1)*gamma(j+1))/gamma(1+i+j+2);
            fprintf(1,'%5i%5i %15.12f %15.12f\n',i,j,vnum,vexact)
        elseif(np==4 & i+j <=3)
            vnum=sum( (vksi4(1,:).^i).*(vksi4(2,:).^j).*vcpg4);
            vexact=(gamma(i+1)*gamma(j+1))/gamma(1+i+j+2);
            fprintf(1,'%5i%5i %15.12f %15.12f\n',i,j,vnum,vexact)
        elseif(np==6 & i+j <=4)
            vnum=sum( (vksi6(1,:).^i).*(vksi6(2,:).^j).*vcpg6 );
            vexact=(gamma(i+1)*gamma(j+1))/gamma(1+i+j+2);
            fprintf(1,'%5i%5i %15.12f %15.12f\n',i,j,vnum,vexact)
        elseif(np==7 & i+j <=5)
            vnum=sum( (vksi7(1,:).^i).*(vksi7(2,:).^j).*vcpg7 );
            vexact=(gamma(i+1)*gamma(j+1))/gamma(1+i+j+2);
            fprintf(1,'%5i%5i %15.12f %15.12f\n',i,j,vnum,vexact)
        elseif(np==12 & i+j <=6)
            vnum=sum( (vksi12(1,:).^i).*(vksi12(2,:).^j).*vcpg12 );
            vexact=(gamma(i+1)*gamma(j+1))/gamma(1+i+j+2);
            fprintf(1,'%5i%5i %15.12f %15.12f\n',i,j,vnum,vexact)
        end
    end
end

```

Figure 5.10. Numerical integration program: triangular element

```
% Numerical integration for a reference tetrahedron
% Direct symmetrical Quadrature FormulaE
%----- Input : np number of points
np=input('Give number of points (1, 4, 5, 11 or 15) :\n');
%----- Exit: integrated value and exact ksi^m eta^n
% m, n : polynomial order
% Vksi : abcissa ksi eta ; vcog : weight
% Table of values Figure 5.9
vksi1=[1/4;1/4]; vcpg1=1/6; % 1 point order 1
%----- 4 points order 2
a=(5-sqrt(5))/20;b=(5+3*sqrt(5))/20;
vksi4=[a,a,a,b;a,a,b,a;a,b,a,a];vcpg4=[1 1 1 1]/24;
%----- 5 points order 3
vksi5=[1/4 1/6 1/6 1/6 1/6;1/4 1/6 1/6 1/2 1/6;1/4 1/6 1/2 1/6 1/6];
vcpg5=[-2/15 3/40 3/40 3/40 3/40];
%----- 11 points order 4
a=(1+sqrt(5/14))/4;b=(1-sqrt(5/14))/4;
vksi11=[1/4 11/14 1/14 1/14 1/14 a a a b b b ;
        1/4 1/14 11/14 1/14 1/14 a b b a b a;
        1/4 1/14 1/14 11/14 1/14 b a b a a b ];
vcpg11=[-74/5625, [1 1 1 1]*343/45000,[1 1 1 1 1]*56/2250];
%----- 15 points order 5
b2=(7+sqrt(15))/34; b1=(7-sqrt(15))/34;
c1=(13+3*sqrt(15))/34;c2=(13-3*sqrt(15))/34;
d= (5+sqrt(15))/20; e=(5-sqrt(15))/20;
vksi15=[1/4 b1 c1 b1 b1 b2 c2 b2 b2 e d d d e e;
        1/4 b1 b1 c1 b1 b2 b2 c2 b2 d e d e d e;
        1/4 b1 b1 b1 c1 b2 b2 b2 c2 d d e e d];
vcpg15=[8/405,    [1 1 1 1]*(2665+14*sqrt(15))/226800, ...
          [1 1 1 1]*(2665-14*sqrt(15))/226800, ...
          [1 1 1 1 1]*5/567];
%----- Integration of each monomials depending on the number of points np
nmax=5;
disp('----- number of points retained: 1 4 5 11 15 ')
disp('----- integration order : 1 2 3 4 5 ')
if sum(np==[1 4 5 11 15]) == 0,
    disp(' no formula for the chosen number of points), break, end
    disp('----- numerical integration with exact value ')
```

```

fprintf(1,' nb de points = %5i\n',np)
disp(' m   n   p   vnum      vexact')
for i=0:nmax
    for j=0:nmax
        for k=0:nmax
            vexact=(gamma(i+1)* ...
                      gamma(j+1)*gamma(k+1))/ gamma(1+i+j+k+3);
            if(np==1 & i+j+k <=0)
                vnum=vcpg1;
            fprintf(1,'%5i%5i%5i15.12f %15.12f\n',i,j,k,vnum,vexact)
            end
            if(np==4 & i+j+k <=2)
                vnum=sum( (vksi4(1,:).^i).*(vksi4(2,:).^j).*...
                           (vksi4(3,:).^k).*vcpg4);
            fprintf(1,'%5i%5i%5i15.12f %15.12f\n',i,j,k,vnum,vexact)
            end
            if(np==5 & i+j+k <=3)
                vnum=sum( (vksi5(1,:).^i).*(vksi5(2,:).^j).*...
                           (vksi5(3,:).^k).*vcpg5 );
            fprintf(1,'%5i%5i%5i15.12f %15.12f\n',i,j,k,vnum,vexact)
            end
            if(np==11 & i+j+k <=4)
                vnum=sum( (vksi11(1,:).^i).*(vksi11(2,:).^j).*...
                           (vksi11(3,:).^k).*vcpg11 );
            fprintf(1,'%5i%5i%5i15.12f %15.12f\n',i,j,k,vnum,vexact)
            end
            if(np==15 & i+j+k <=5)
                vnum=sum( (vksi15(1,:).^i).*(vksi15(2,:).^j).*...
                           (vksi15(3,:).^k).*vcpg15 );
            fprintf(1,'%5i%5i%5i15.12f %15.12f\n',i,j,k,vnum,vexact)
            end
        end
    end
end

```

Figure 5.11. Numerical integration program: tetrahedral element

5.2 Solving systems of linear equations [CAR 69; BAT 79]

5.2.1 INTRODUCTION

Solving the system of equations:

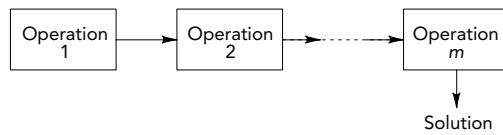
$$[K]\{U_n\} = \{F\} \quad (5.22)$$

is an important step in the finite element method. This system is linear when $[K]$ is independent of $\{U_n\}$.

The number n of unknowns U_n is proportional to the total number of interpolation nodes and to the number of degrees of freedom per node. The accuracy and scope of application of the finite element method is limited by the size of the systems of equations, which can now be solved cost-effectively on easily available computers.

Methods of solving linear systems can be classified into two categories:

- a) Direct methods that lead to the solution with a number of operations known *a priori*:



- b) Iterative methods that lead to the solution by way of a series of improvements to an approximate solution; the number of iterations required is difficult to predict and depends on the conditioning of matrix $[K]$.

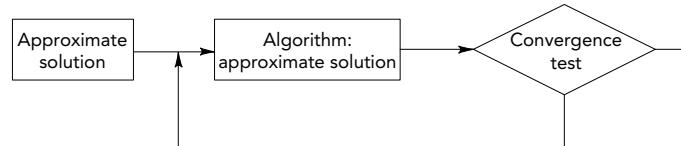


Figure 5.12. General iteration procedure

The earliest finite element codes used iterative methods (Gauss–Seidel), because they are simpler to program and require less memory space than direct methods (see section 5.3).

Many contemporary programs use direct methods derived from the Gaussian method, because they generally require far fewer operations than iterative methods. On the other hand, they are generally more prone to rounding errors due to the limited accuracy with which the computer performs arithmetic operations. We will only discuss direct methods in this chapter.

5.2.2 GAUSSIAN ELIMINATION METHOD

This very frequently used method comprises two stages:

a) Triangularization

This stage consists of transforming the system of equations (5.22) into a triangular system:

$$\begin{bmatrix} & & S \\ 0 & & \end{bmatrix} \{U_n\} = \{F'\}. \quad (5.23)$$

b) Solution of the upper triangular system

This step consists of calculating the unknowns U_n from the last to the first, by solving the triangular system (5.23) (this method is known as *back-substitution*).

5.2.2.1 Triangularization

$$\begin{bmatrix} K_{11} & K_{12} & \dots & K_{1n} \\ K_{21} & K_{22} & \dots & K_{2n} \\ \vdots & \vdots & \dots & \vdots \\ K_{n1} & K_{n2} & \dots & K_{nn} \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ \vdots \\ U_n \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ \vdots \\ F_n \end{Bmatrix}. \quad (5.24)$$

Triangularization consists of successively “eliminating” the unknowns U_s , $s = 1, 2, \dots, n-1$ in the equations $s+1$ to n . The elimination of U_s is carried out as follows:

- express U_s in terms of $U_{s+1}, U_{s+2}, \dots, U_n$ and F_s using equation s ;
- substitute the previous expression U_s into the equations $s+1, s+2, \dots, n$.

After eliminating U_s , this unknown no longer appears in the equations $s+1$ to n ; there are therefore zeros in column s below the diagonal. After eliminating the unknowns U_1 to U_{n-1} , the matrix $[K]$ is upper triangular, as it now contains only zeros below the diagonal.

The elimination of each unknown U_s modifies $[K]$ and $\{F\}$. Let $[K^s]$ and $\{F^s\}$ represent the matrix and the right-hand side of the equation after elimination of the unknowns $1, 2, 3, \dots, s$, where matrix $[K^0]$ is the initial matrix $[K]$:

$[K] = [K^0]$ and $\{F^0\} = \{F\}$ ORIGINAL SYSTEM

\downarrow eliminate U_1 in equations 2 to n

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

\downarrow eliminate U_2 in equations 3 to n

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

\downarrow

⋮

\downarrow eliminate U_s in equations $s+1$ to n

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

\downarrow

⋮

\downarrow eliminate U_{n-1} in equation n

$[S] = [K^{n-1}]$ and $\{F^{n-1}\} = \{F'\}$ TRIANGULAR SYSTEM

To eliminate the variable U_1 from system (5.24), we use the first equation:

$$U_1 = \frac{1}{K_{11}} (F_1 - K_{12} U_2 - \dots - K_{1n} U_n)$$

and use this expression U_1 in the equations 2, 3, ..., n :

$$\begin{bmatrix} K_{11} & K_{12} & & K_{1n} \\ 0 & K_{22} - \frac{K_{21}}{K_{11}} K_{12} & \cdots & K_{2n} - \frac{K_{21}}{K_{11}} K_{1n} \\ \vdots & \vdots & & \vdots \\ 0 & K_{n2} - \frac{K_{n1}}{K_{11}} K_{12} & \cdots & K_{nn} - \frac{K_{n1}}{K_{11}} K_{1n} \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ \vdots \\ U_n \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 - \frac{K_{21}}{K_{11}} F_1 \\ \vdots \\ F_n - \frac{K_{n1}}{K_{11}} F_1 \end{Bmatrix}$$

which we write as:

$$\begin{bmatrix} K_{11} & K_{12} & & K_{1n} \\ 0 & K_{22}^1 & \cdots & K_{2n}^1 \\ \vdots & \vdots & & \vdots \\ 0 & K_{n2}^1 & \cdots & K_{nn}^1 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ \vdots \\ U_n \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2^1 \\ \vdots \\ F_n^1 \end{Bmatrix}$$

where

$$K_{ij}^1 = K_{ij} - K_{i1} \cdot K_{11}^{-1} \cdot K_{1j} \quad i, j = 2, 3, \dots, n.$$

$$F_i^1 = F_i - K_{i1} \cdot K_{11}^{-1} \cdot F_1$$

After eliminating $U_1 \ U_2 \ \dots \ U_{s-1}$:

$$\left[\begin{array}{cccccc} K_{11} & K_{12} & \cdots & K_{1s} & \cdots & K_{1n} \\ 0 & K_{22}^1 & \cdots & K_{2s}^1 & \cdots & K_{2n}^1 \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & K_{ss}^{s-1} & \cdots & K_{sn}^{s-1} \\ 0 & 0 & \cdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & K_{ns}^{s-1} & \cdots & K_{nn}^{s-1} \end{array} \right] \begin{Bmatrix} U_1 \\ U_2 \\ \vdots \\ U_s \\ \vdots \\ U_n \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2^1 \\ \vdots \\ F_s^{s-1} \\ \vdots \\ F_n^{s-1} \end{Bmatrix}$$

$$[K^{s-1}] \{U_n\} = \{F^{s-1}\}.$$

After eliminating U_s :

$$[K^s] \{U_n\} = \{F^s\}$$

the modified terms being:

$$\begin{aligned} K_{ij}^s &= K_{ij}^{s-1} - K_{is}^{s-1} (K_{ss}^{s-1})^{-1} K_{sj}^{s-1} \\ F_i^s &= F_i^{s-1} - K_{is}^{s-1} (K_{ss}^{s-1})^{-1} F_s^{s-1} \quad i, j = s+1, s+2, \dots, n. \end{aligned} \quad (5.25)$$

The final triangular system is written as:

$$\left[\begin{array}{cccccc} K_{11} & \cdots & \cdots & \cdots & \cdots & \cdots & K_{1n} \\ 0 & K_{22}^1 & \cdots & \cdots & \cdots & \cdots & K_{2n}^1 \\ \vdots & 0 & K_{33}^2 & \cdots & \cdots & \cdots & K_{3n}^2 \\ \vdots & \vdots & 0 & \ddots & & & \vdots \\ \vdots & \vdots & \vdots & & K_{ss}^{s-1} & \cdots & K_{sn}^{s-1} \\ \vdots & \vdots & \vdots & & 0 & & \vdots \\ \vdots & \vdots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & \cdots & K_{nn}^{n-1} \end{array} \right] \begin{Bmatrix} U_1 \\ U_2 \\ \vdots \\ U_s \\ \vdots \\ U_n \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2^1 \\ \vdots \\ F_s^{s-1} \\ \vdots \\ F_n^{n-1} \end{Bmatrix}$$

$$\begin{cases} K^{n-1} \\ 0 \\ S \\ 0 \end{cases} \{U_n\} = \{F^{n-1}\}$$

that is

$$\begin{cases} K^{n-1} \\ 0 \\ S \\ 0 \end{cases} \{U_n\} = \{F'\}.$$

In practice, we construct the successive matrices $[K^1], [K^2], \dots$ in the original matrix $[K]$. The algorithm is as follows:

$$\boxed{\begin{array}{l} s = 1, 2, \dots, n-1 \\ \quad i = s+1, s+2, \dots, n \\ \quad c = K_{is} K_{ss}^{-1} \\ \quad F_i = F_i - c F_s \\ \quad j = s+1, s+2, \dots, n \\ \quad K_{ij} = K_{ij} - c K_{sj}. \end{array}} \quad (5.26)$$

Remark

- For a symmetrical system, the index j ranges from i to n . Moreover, $K_{ji} = K_{ij}$.

EXAMPLE 5.12. Triangularization of a non-symmetrical system

$$\begin{bmatrix} 2 & 4 & 8 \\ 4 & 11 & 25 \\ 6 & 18 & 46 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} = \begin{Bmatrix} 34 \\ 101 \\ 180 \end{Bmatrix}$$

$$[K] \{U_n\} = \{F\}.$$

After eliminating U_1 :

$$\begin{bmatrix} 2 & 4 & 8 \\ 0 & 11 - \frac{4}{2} \cdot 4 = 3 & 25 - \frac{4}{2} \cdot 8 = 9 \\ 0 & 18 - \frac{6}{2} \cdot 4 = 6 & 46 - \frac{6}{2} \cdot 8 = 22 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} = \begin{Bmatrix} 34 \\ 101 - \frac{4}{2} \cdot 34 = 33 \\ 180 - \frac{6}{2} \cdot 34 = 78 \end{Bmatrix}.$$

After eliminating U_1 and U_2 :

$$\begin{bmatrix} 2 & 4 & 8 \\ 0 & 3 & 9 \\ 0 & 0 & 22 - \frac{6}{3} \cdot 9 = 4 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} = \begin{Bmatrix} 34 \\ 33 \\ 78 - \frac{6}{3} \cdot 33 = 12 \end{Bmatrix}.$$

Hence:

$$\begin{bmatrix} 2 & 4 & 8 \\ 0 & 3 & 9 \\ 0 & 0 & 4 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} = \begin{Bmatrix} 34 \\ 33 \\ 12 \end{Bmatrix}.$$

$$[S] \{U_n\} = \{F'\},$$

$$\det(K) = \det(S) = S_{11} \times S_{22} \times S_{33} = 24.$$

Algorithm (5.2) no longer works if, during the course of triangularization, the pivot K_{ss} is zero. In this case, we have to exchange row s with another row $i > s$ such that $K_{is} \neq 0$. In the case of a symmetrical system, we can preserve the symmetry if we exchange columns i and s equally, which implies a modification of the order of unknowns; for this to be possible, the diagonal term K_{ii} must be non-zero.

If, when $K_{ss} = 0$, all terms K_{si} ($i > s$) or all terms K_{is} ($i > s$) are zero, matrix K is singular; the system cannot be solved.

The determinant of K is the product of the diagonal terms of the triangular matrix $[S]$.

EXAMPLE 5.13. Finding a non-zero pivot

Consider the system:

$$\begin{bmatrix} 0 & 4 & 8 \\ 4 & 0 & 6 \\ 8 & 6 & 0 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} = \begin{Bmatrix} 2 \\ 4 \\ 12 \end{Bmatrix}.$$

The first pivot being zero, swap the first and second row:

$$\begin{bmatrix} 4 & 0 & 6 \\ 0 & 4 & 8 \\ 8 & 6 & 0 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} = \begin{Bmatrix} 4 \\ 2 \\ 12 \end{Bmatrix}.$$

This matrix is no longer symmetrical. This being the case, we cannot exchange columns one and two, as this will restore a zero pivot in the first row.

After eliminating U_1 :

$$\begin{bmatrix} 4 & 0 & 6 \\ 0 & 4 & 8 \\ 0 & 6 & -12 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} = \begin{Bmatrix} 4 \\ 2 \\ 4 \end{Bmatrix}.$$

Then eliminating U_2 :

$$\begin{bmatrix} 4 & 0 & 6 \\ 0 & 4 & 8 \\ 0 & 0 & -24 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} = \begin{Bmatrix} 4 \\ 2 \\ 1 \end{Bmatrix},$$

$$\det(K) = -\det(S) = -4 \times 4 \times (-24) = 384,$$

the negative sign resulting from the inversion of the rows.

5.2.2.2 Solution of an upper triangular system

The upper triangular system obtained by (5.26) can be solved by successively calculating $U_n U_{n-1} \dots U_1$, hence the appellation *back-substitution*:

$$\begin{aligned} U_n &= S_{nn}^{-1} F'_n \\ U_{n-1} &= S_{n-1,n-1}^{-1} (F'_{n-1} - S_{n-1,n} U_n) \\ &\dots \\ U_1 &= S_{11}^{-1} (F'_1 - S_{12} U_2 - S_{13} U_3 - \dots - S_{1n} U_n). \end{aligned}$$

The practical algorithm works directly on the matrices $[K]$ and $\{F\}$ modified by triangularization:

$$\begin{aligned} U_n &= K_{nn}^{-1} F_n \\ i = n-1, n-2, \dots, 1 \\ U_i &= K_{ii}^{-1} \left(F_i - \sum_{j=i+1}^n K_{ij} U_j \right). \end{aligned} \tag{5.27}$$

EXAMPLE 5.14. Solving a triangular system

Let us solve the triangular system obtained in Example 5.12:

$$\begin{bmatrix} 2 & 4 & 8 \\ 0 & 3 & 9 \\ 0 & 0 & 4 \end{bmatrix} \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix} = \begin{Bmatrix} 34 \\ 33 \\ 12 \end{Bmatrix}$$

$$U_3 = \frac{12}{4} = 3$$

$$U_2 = \frac{1}{3} (33 - 9 \times 3) = 2$$

$$U_1 = \frac{1}{2} (34 - 4 \times 2 - 8 \times 3) = 1.$$

5.2.3 DECOMPOSITION

5.2.3.1 Introduction

We will matrixially reformulate the Gaussian elimination operations that transform matrix $[K]$ into an upper triangular matrix $[S]$. This will enable us:

- a) to show that in fact the Gauss method decomposes $[K]$ in the form:

$$[K] = \begin{bmatrix} & & 0 \\ & \ddots & \\ L & & \end{bmatrix} \begin{bmatrix} & & S \\ & \ddots & \\ 0 & & \end{bmatrix} = [L][S] \quad (5.28)$$

where: $[L]$ is a lower triangular matrix with diagonal units,

$[S]$ is the upper triangular matrix obtained by Gaussian elimination in section 5.2.2.1;

- b) to carry out the operations involving $[K]$ and $\{F\}$ during elimination separately. This allows us, after triangularization of $[K]$, to successively solve the system with several vectors on the right-hand side;
- c) to construct resolution algorithms adapted for the storage of $[K]$ by way of the skyline method introduced in section 4.6.3.

5.2.3.2 Matrix formulation of the Gauss elimination

The elimination operation (5.25) of the unknown U_s transforms $[K^{s-1}]$ into $[K^s]$. This transformation can be written in matrix form:

$$[K^s] = ([I] + \begin{bmatrix} 0 & & & \\ & \ddots & & 0 \\ & & 0 & \\ & & -l_{s+1,s} & \\ 0 & & \vdots & \ddots \\ & & -l_{n,s} & 0 \end{bmatrix}) [K^{s-1}] = [l^s] [K^{s-1}] \quad (5.29)$$

where: $[I]$ is the unit matrix

$$l_{is} = K_{is}^{s-1} (K_{ss}^{s-1})^{-1} \quad i = s+1, s+2, \dots, n.$$

The matrix $[l^s]$ is lower triangular, its diagonal terms are equal to 1 and only the column s is non-null.

Eliminating the unknowns U_1, U_2, \dots, U_{n-1} (algorithm 5.26) is equivalent to successively applying operation (5.29) with $s = 1, 2, \dots, n-1$. Hence:

$$\begin{aligned} [K^{n-1}] &= \begin{bmatrix} & & S \\ & \searrow & \\ 0 & & \end{bmatrix} = [l^{n-1}] [l^{n-2}] \dots [l^1] [K] \\ \begin{bmatrix} & & S \\ & \searrow & \\ 0 & & \end{bmatrix} &= \begin{bmatrix} & & 0 \\ & \searrow & \\ l & & \end{bmatrix} [K] = [l] [K] \end{aligned} \quad (5.30)$$

EXAMPLE 5.15. Decomposition of the matrix of Example 5.12

$$[K] = \begin{bmatrix} 2 & 4 & 8 \\ 4 & 11 & 25 \\ 6 & 18 & 46 \end{bmatrix}.$$

We use expression (5.29)

$$s = 1$$

$$[K^1] = [l^1] [K]$$

$$s = 2$$

$$[K^2] = [l^2] [K^1] = [l^2] [l^1] [K] = [l] [K]$$

$$[l^1] = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}; \quad [K^1] = \begin{bmatrix} 2 & 4 & 8 \\ 0 & 3 & 9 \\ 0 & 6 & 22 \end{bmatrix}$$

$$[l^2] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix}; \quad [K^2] = \begin{bmatrix} 2 & 4 & 8 \\ 0 & 3 & 9 \\ 0 & 0 & 4 \end{bmatrix} = [S]$$

$$[l] = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -3 & -2 & 1 \end{bmatrix}.$$

The decomposition (5.28) of K is obtained by inverting the triangular matrix $[l]$ defined by (5.30):

$$\begin{aligned}[K] &= [l]^{-1} [S] = [l^1]^{-1} [l^2]^{-1} \dots [l^{n-1}]^{-1} [S] \\ &= [L^1] [L^2] \dots [L^{n-1}] [S] = \begin{bmatrix} & & & 0 \\ L & & & \\ & & & \\ & & & \end{bmatrix} [S] \\ &= [L] [S].\end{aligned}\quad (5.31)$$

5.2.3.3 Properties of the triangular matrices $[l^s]$

Here we present the properties of the triangular matrices $[l^s]$ and $[L^s]$ that appear in (5.30) and (5.31) and for which the diagonal terms are all equal to 1:

- the product of two lower (or upper) triangular matrices is a lower (or an upper) triangular matrix;
- the determinant of a triangular matrix is equal to the product of its diagonal terms;
- the inverse of a lower (or upper) triangular matrix is a lower (or upper) triangular matrix;
- The topology (bandwidth, skyline) of $[S]$ (or of $[L]$) is identical to that of the upper (or lower) part of $[K]$.
- the inverse of the matrices $[l^s]$ is written as:

$$[l^s]^{-1} = [L^s] = -[l^s] + 2[I] \quad (5.32)$$

which is tantamount to changing the sign of the terms beneath the diagonal;

- the product of two matrices $[L^i][L^j]$, where $i \leq j$, is written as:

$$[L^i][L^j] = [L^i] + [L^j] - [I] \quad (5.33)$$

which is tantamount to adding the non-diagonal terms of the two matrices, while keeping the diagonal terms equal to 1.

EXAMPLE 5.16. Matrices $[L]$ and $[S]$ from Example 5.15

Let us apply relations (5.31) to the matrices obtained in Example 5.15, using the properties (5.32) and (5.33):

$$[K] = [L^1][L^2][S] = [L][S]$$

where

$$[L^1] = [l^1]^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix}$$

$$\begin{aligned}
 [L^2] &= [l^2]^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} \\
 [L] &= [L^1][L^2] = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix} \\
 [S] &= [K^2] = \begin{bmatrix} 2 & 4 & 8 \\ 0 & 3 & 9 \\ 0 & 0 & 4 \end{bmatrix}.
 \end{aligned}$$

From this we derive the final decomposition:

$$\begin{bmatrix} 2 & 4 & 8 \\ 4 & 11 & 25 \\ 6 & 18 & 46 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 4 & 8 \\ 0 & 3 & 9 \\ 0 & 0 & 4 \end{bmatrix}$$

$[K]$ $[L]$ $[S]$

5.2.3.4 Various forms of the decomposition of $[K]$

We will systematically use the decomposition (5.31) of $[K]$, known as the Doolittle decomposition,

$$[K] = [L][S]. \quad (5.34)$$

Three other decompositions are sometimes used:

a) LDU decomposition

We decompose $[S]$ in the form of a product of a diagonal matrix $[D]$ and of an upper triangular matrix with unit diagonal coefficients $[U]$:

$$[S] = \begin{bmatrix} D \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} U \\ \vdots \\ 0 \end{bmatrix}$$

$$\begin{aligned} U_{ii} &= 1 \\ U_{ij} &= S_{ij}/S_{ii} \quad j > i \\ D_{ii} &= S_{ii}. \end{aligned}$$

Then

$$[K] = [L][D][U]. \quad (5.35)$$

b) Crout decomposition

$$[K] = [L'][U] \quad (5.36)$$

where

$$[L'] = [L][D]$$

For symmetrical matrices, equation (5.35) becomes:

$$[K] = [L][D][U] = [K]^T = [U]^T [D][L]^T$$

and therefore:

$$\begin{aligned} [U] &= [L]^T \\ [K] &= [L][D][L]^T. \end{aligned} \quad (5.37)$$

c) Cholesky decomposition

When $[K]$ is positive definite ($S_{ii} > 0$) we can write (5.37) in the form of the Cholesky decomposition:

$$[K] = [L_c][L_c]^T \quad (5.38)$$

where:

$$[L_c] = [L] \begin{bmatrix} & & & 0 \\ & & \sqrt{S_{ii}} & \\ & 0 & & \\ 0 & & & \end{bmatrix}.$$

5.2.3.5 Solving a system by decomposition

The system to be solved

$$[K]\{U_n\} = \{F\}$$

is written as follows, using decomposition (5.34):

$$[L][S]\{U_n\} = \{F\}.$$

This system is solved in two stages:

$$\begin{array}{l} [L]\{F'\} = \{F\} \quad \text{Lower triangular system} \\ \downarrow \\ [S]\{U_n\} = \{F'\} \quad \text{Upper triangular system} \end{array} \quad (5.39)$$

Note that $\{F'\}$ is identical to the right-hand side of the equation obtained following the use of the Gaussian elimination (5.26).

EXAMPLE 5.17. Resolution by decomposition

Let us solve the system from Example 5.12 using the decomposition of $[K]$ obtained in Example 5.16:

— Stage 1: lower triangular system

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix} [L] \{F'\} = \begin{bmatrix} 34 \\ 101 \\ 180 \end{bmatrix} \Rightarrow \{F'\} = \begin{bmatrix} 34 \\ 33 \\ 12 \end{bmatrix}$$

— Stage 2: upper triangular system

$$\begin{bmatrix} 2 & 4 & 8 \\ 0 & 3 & 9 \\ 0 & 0 & 4 \end{bmatrix} [S] \{U_n\} = \begin{bmatrix} 34 \\ 33 \\ 12 \end{bmatrix} \Rightarrow \{U_n\} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

5.2.3.6 Decomposition algorithms

A decomposition algorithm is used to calculate the terms of $[L]$ and $[S]$ from the terms of $[K]$; these terms are stored in $[K]$ in the form:

$$\begin{bmatrix} S_{11} & S_{12} & \dots & S_{1n} \\ L_{21} & & & \\ L_{n1} & \dots & L_{n,n-1} & S_{nn} \end{bmatrix}. \quad (5.40)$$

The Gaussian elimination algorithm (5.26) can be considered a decomposition algorithm so long as the terms L_{is} are stored in the lower part of $[K]$:

$$\left| \begin{array}{l} s = 1, 2, \dots, n-1 \\ \quad \left| \begin{array}{l} i = s+1, s+2, \dots, n \\ \quad \left| \begin{array}{l} K_{is} = K_{is} K_{ss}^{-1} \text{ (column } s \text{ of } L) \\ \quad (L_{is}) \\ \quad \left| \begin{array}{l} j = s+1, s+2, \dots, n \\ \quad \left| \begin{array}{l} K_{ij} = K_{ij} - K_{is} K_{sj} \\ \quad (L_{is})(S_{sj}) \end{array} \right. \end{array} \right. \end{array} \right. \end{array} \right. \quad (5.41)$$

For a given value of s , this algorithm

- creates the column s of L (beneath the diagonal);
- creates the row $s+1$ of S (to the right of the diagonal);
- modifies the terms K_{ij} $i, j > s$.

We can construct other algorithms by identifying the terms of the product $[L][S]$ with the terms of $[K]$. For instance, we can obtain an algorithm that successively constructs a row of L and a column of S . This algorithm lends itself well to the storage of $[K]$ by the skyline method described in section 4.6.3 (see section 5.2.4.1).

$$\begin{bmatrix} 1 & & & \\ L_{21} & 1 & & \\ L_{31} & L_{32} & 1 & \\ \dots & & & \end{bmatrix} \begin{bmatrix} S_{11} & S_{12} & S_{13} & \dots \\ & S_{22} & S_{23} & \dots \\ & & S_{33} & \dots \\ & & & \ddots \end{bmatrix} = \begin{bmatrix} K_{11} & K_{12} & K_{13} & \dots \\ K_{21} & K_{22} & K_{23} & \dots \\ K_{31} & K_{32} & K_{33} & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix} \quad (5.42)$$

$$s=1: 1 \cdot S_{11} = K_{11}$$

$$\text{hence: } S_{11} = K_{11}$$

$$s=2: L_{21} S_{11} = K_{21}$$

$$L_{21} = K_{21} (S_{11})^{-1}$$

$$1 \cdot S_{12} = K_{12}$$

$$S_{12} = K_{12}$$

$$L_{21} S_{12} + S_{22} = K_{22}$$

$$S_{22} = K_{22} - L_{21} S_{12}$$

$$s=3: L_{31} S_{11} = K_{31}$$

$$L_{31} = K_{31} (S_{11})^{-1}$$

$$L_{31} S_{12} + L_{32} S_{22} = K_{32}$$

$$L_{32} = (K_{32} - L_{31} S_{12}) (S_{22})^{-1}$$

$$1 \cdot S_{13} = K_{13}$$

$$S_{13} = K_{13}$$

$$L_{21} S_{13} + S_{23} = K_{23}$$

$$S_{23} = K_{23} - L_{21} S_{13}$$

$$L_{31} S_{13} + L_{32} S_{23} + S_{33} = K_{33}$$

$$S_{33} = K_{33} - L_{31} S_{13} - L_{32} S_{23}.$$

For any value of s :

$$\begin{aligned} L_{si} &= \left(K_{si} - \sum_{m=1}^{i-1} L_{sm} S_{mi} \right) S_{ii}^{-1} \quad i = 1, 2, \dots, s-1 \\ S_{js} &= K_{js} - \sum_{m=1}^{j-1} L_{jm} S_{ms} \quad j = 1, 2, \dots, s. \end{aligned} \tag{5.43}$$

Given that the matrices L and S are stored in $[K]$ in the form (5.40), this latest algorithm is written as:

$$\begin{aligned}
 s &= 2, 3, \dots, n-1 \\
 i &= 1, 2, \dots, s-1 \\
 m &= 1, 2, \dots, i-1 \\
 K_{si} &= K_{si} - K_{sm} K_{mi} && \text{row of } L \\
 K_{is} &= K_{is} - K_{im} K_{ms} && \text{column of } S \\
 K_{si} &= K_{si} K_{ii}^{-1} && \text{normalizes the row of } L \\
 m &= 1, 2, \dots, s-1 \\
 K_{ss} &= K_{ss} - K_{sm} K_{ms} && \text{diagonal term.}
 \end{aligned} \tag{5.44}$$

EXAMPLE 5.18. Decomposition of the matrix from Example 5.12 using algorithm (5.44)

$$[K] = \begin{bmatrix} 2 & 4 & 8 \\ 4 & 11 & 25 \\ 6 & 18 & 46 \end{bmatrix}.$$

After step $s = 2$

$$[K] = \begin{bmatrix} 2 & 4 & 8 \\ 2 & 3 & 25 \\ 6 & 18 & 46 \end{bmatrix}.$$

After step $s = 3$

$$[K] = \begin{bmatrix} 2 & 4 & 8 \\ 2 & 3 & 9 \\ 3 & 2 & 4 \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} & S_{13} \\ L_{21} & S_{22} & S_{23} \\ L_{31} & L_{32} & S_{33} \end{bmatrix}.$$

Following decomposition by algorithm (5.44), the matrix $[K]$ contains the terms of $[L]$ and $[S]$. We then have to solve the two triangular systems (5.39):

— Stage 1 (lower triangular system)

$$\boxed{i = 2, \dots, n} \quad F_i = F_i - \sum_{j=1}^{i-1} K_{ij} F_j. \quad (5.45)$$

— Stage 2 (upper triangular system)

$$\boxed{i = n-1, n-2, \dots, 1} \quad F_n = K_{nn}^{-1} F_n$$

$$\boxed{F_i = K_{ii}^{-1} \left(F_i - \sum_{j=i+1}^n K_{ij} F_j \right).} \quad (5.46)$$

5.2.4 ADAPTATION OF ALGORITHM (5.44) TO THE CASE OF A MATRIX STORED BY THE SKYLINE METHOD

5.2.4.1 Skyline matrix residing in central memory

When we wish to avoid the operations relating to null terms in $[K]$ outside the skyline (see section 4.6.3e), algorithm (5.44) must be modified as follows:

$$\boxed{s = 2, 3, \dots, n}$$

$$\boxed{i = i_{0s}, \dots, s-1}$$

$$\boxed{m = \text{Max}(i_{0i}, i_{0s}), \dots, i-1}$$

$$\boxed{K_{si} = K_{si} - K_{sm} K_{mi}}$$

$$\boxed{K_{is} = K_{is} - K_{im} K_{ms}}$$

$$\boxed{K_{si} = K_{i_s} K_{ii}^{-1}}$$

$$\boxed{m = i_{0s}, \dots, s-1}$$

$$\boxed{K_{ss} = K_{ss} - K_{sm} K_{ms}} \quad (5.47)$$

where i_{0i} and i_{0s} are the row numbers of the upper terms in columns i and s (or indeed the column numbers of the left-hand terms in rows i and s).

In view of (4.32b) and (4.41):

$$i_{0i} = i - h_J(i) = i - \text{KLD}(i+1) + \text{KLD}(i) \quad (5.48)$$

$$i_{0s} = s - h_J(s) = s - \text{KLD}(s+1) + \text{KLD}(s)$$

Figure 5.13 shows the positions of the different terms of $[K]$ involved in algorithm (5.47). In order to use the vector storage method of $[K]$ from section 4.6.3e, we need only calculate the positions in the VKGS, VKGD and VKGI tables of those terms of $[K]$ that appear in (5.47). This brings the table of KLD pointers (4.42) into play.

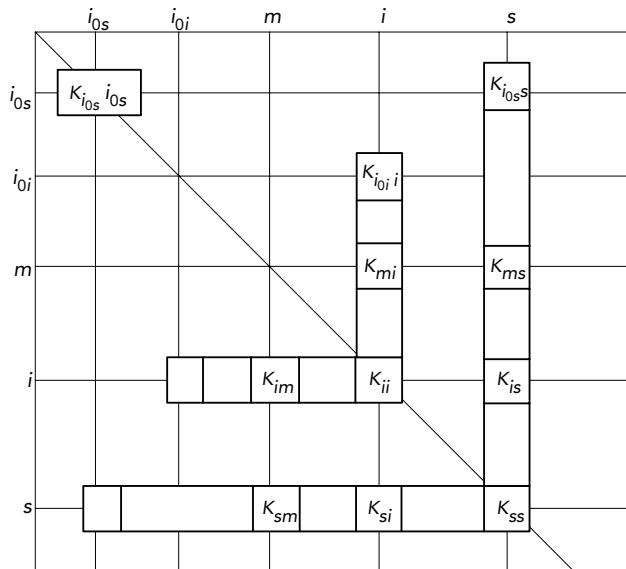


Figure 5.13. Position of the terms of $[K]$ involved in algorithm (5.47)

In the case of a symmetrical system, algorithm (5.47) is modified to avoid operating on the lower triangle of $[K]$ and storing the corresponding terms:

$$\begin{aligned}
 & s = 2, 3, \dots, n \\
 & \quad \left| \begin{array}{l} i = i_{0s}, +1, \dots, s-1 \\ m = \text{Max}(i_{0i}, i_{0s}), \dots, i-1 \\ K_{is} = K_{is} - K_{im} K_{ms} \\ c = 0 \end{array} \right. \\
 & \quad \left| \begin{array}{l} m = i_{0s}, \dots, i-1 \\ c = c + K_{ms}^2 / K_{mm} \\ K_{ms} = K_{ms} / K_{mm} \\ K_{ss} = K_{ss} - c. \end{array} \right. \tag{5.49}
 \end{aligned}$$

5.2.4.2 Current trends

The evolution over the course of the past 20 years in computation capacities – in terms of RAM, floating point operations per second and processor power – means that today we can envisage computations with very large numbers of degrees of freedom (several million unknowns). However, computing techniques based on better management of IT resources mean that this type of massive computation can be carried out on standard computers, rather than having to resort to specialized computers such as CRAY or similar ones.

a) Sparse matrix storage

The so-called sparse matrix storage technique, an extension of the SKYLINE approach, consists of only storing the non-null terms of the matrices and vectors, thereby facilitating a better use of the memory storage capacities. However, this approach requires us to redefine the subprograms used to carry out the basic matrix operations, e.g. the matrix vector product. This enables us to envisage handling very large problems on machines with relatively reduced capacities [SAA 03].

By way of example, we cite the operators on sparse matrices in Matlab[©]. These only require that we declare the matrices and vectors involved in the form:

<code>vmg=sparse(vmg);</code>	<code>% global mass matrix</code>
<code>vkg=sparse(vkg);</code>	<code>% global stiffness matrix</code>
<code>vfg=sparse(vfg);</code>	<code>% global force vector</code>

No other modification is required in the program.

b) Iterative solution methods

In most three-dimensional problems, it is too costly in terms of memory space and CPU time to solve the system direct methods. Another alternative is therefore to use iterative resolution methods [HAD 96; SAA 03], thereby avoiding the problem of storing the data in the form of global matrices.

Iterative techniques for solving nonlinear problems (see section(5.3) are based on transforming the problem into a series of linear problems in the form $[A] \{x\} = \{b\}$ such that the subspace of linear solutions converges toward the nonlinear solution sought. The success of such an iterative strategy depends on

the construction of the linearized problem (Newton; Newton–Raphson). The principle of preconditioning of a matrix $[A]$ consists of replacing the previous system with the equivalent system $[A'] \{x'\} = \{b'\}$, such that the conditioning of $[A']$ is far smaller than that of $[A]$. The convergence of these methods is therefore directly linked to the choice of the appropriate preconditioning, the renumbering of the variables to force convergence, and finally, the choice of a “correct” stop criterion in order to accelerate the convergence.

Of the main methods used, we can cite the iterative methods *Conjugate Gradient* (CG), *BiConjugate Gradient* (BICG), *Conjugate Gradient Squared* (CGS), *BiConjugate Gradient Stabilized* (BICGSTAB), *Transpose-Free Quasi Minimal Residual* (TFQMR), *Full Orthogonal Method* (FOM) and *Generalized Minimal Residual* (GMRES).

c) Distributed calculation techniques

Another approach is to use distributed calculation techniques. Of the existing techniques, we might cite for example, the iterative substructuring domain decomposition method.

This consists of subdividing the calculation domain into a number N of subdomains (submeshes) that will be worked separately and simultaneously by N different processors. An interface is thus defined by the set of boundaries common to all the subdomains. Given that the solution on each of these subdomains is dependent upon those of the neighboring subdomains, an additional stage of processing of the interface is necessary. The solution of the entire problem can thus be characterized by the following stages:

- treatment of the internal variables in the subdomains (simultaneous activity of the N processors);
- assembly and solution of the interface problem (a single processor but distributed solution of the interface problem is a possibility);
- working back from the “interface” solution to the internal variables (simultaneous activity of the N processors).

Thus, the gain in performance offered by a distributed approach in comparison to a conventional approach on the whole domain is directly linked to the yield, defined by:

$$\eta_N = \frac{T_{\text{sequential}}}{T_{\text{decomposition}}}$$

where $T_{\text{sequential}}$ is the computation time taken to solve the overall problem with a single processor, and $T_{\text{decomposition}}$ is the time corresponding to a distributed, decomposition-based approach.

This yield depends greatly on the computation time needed to solve the interface problem. The main problems processed nowadays can thus be classified into two clearly distinct categories:

- problems of linear/nonlinear statics, temporal schemes based on implicit schemes, etc.
- temporal schemes based on explicit schemes, iterative resolution methods, etc.

The former category covers most elliptical problems, in which the solution at one node depends on the solution at all other nodes in the mesh. The interface problem therefore requires a “condensation” of the information (stiffness, for instance) across the entire domain of computation (similar to the “superelement” technique).

The second category encompasses the so-called wave propagation problems, where the solution at a node and at a given time is a function only of those calculated previously at the neighboring nodes. Here, the interface problem is greatly reduced. Thus, the best performances are usually obtained for the second category of problems with a performance gain close to the number of processors used.

d) Computation platform

Today, it is no longer necessary to use machines that are massively parallel, expensive and require a dedicated administration. Economical solutions based on *clusters* of standard machines administrated in a Linux environment and connected by a high-speed network are commonplace. These solutions offer many parallel and easy-to-use computation environments, the best-known of which are PVM (*Parallel Virtual Machine*) [PVM 94] and MPI (*Message Passing Interface*). These environments, which are available for free, enable the user to group together a set of heterogeneous virtual machines on a single physical machine. They provide the tools needed for transfer of data such as vectors or matrices between processors (the message passing technique). These environments also enable us to set up message passing between copies of a code duplicated as many times as need be, and to update the data shared between two distinct calculation codes, devoted to applications of a different nature – for instance, a pressure field and the velocity field of a flexible wall in analyzing the fluid/structure coupling.

5.3 Solution of nonlinear systems

5.3.1 INTRODUCTION

Nonlinearities appear in the formulation of physical problems for two reasons:

- The physical parameters that are supposedly independent of \mathbf{U}_n in a linear model, such as Young's modulus, the coefficients of conductivity and viscosity, etc., may become functions of \mathbf{U}_n . Such is the case, for instance, in plasticity, in non-Newtonian flows, in unsaturated porous mediums, etc.
- Terms that are nonlinear in relation to the unknowns of the problem appear in the partial differential equations, even when the physical properties are independent of \mathbf{U}_n . For example, in the Navier–Stokes equations (Example 3.2), the following terms appear:

$$u \frac{\partial u}{\partial x} + v \frac{\partial u}{\partial y} + \dots$$

and in elasticity with large displacements (Example 4.11):

$$\varepsilon_x = \frac{\partial u}{\partial x} + \frac{1}{2} \left(\frac{\partial v}{\partial x} \right)^2.$$

The finite element method gives us a discretized formulation of the nonlinear problems, which can be written in the form (4.5b):

$$W = \langle \delta \mathbf{U}_n \rangle ([K(\mathbf{U}_n)] \{ \mathbf{U}_n \} - \{ \mathbf{F} \}) = 0 \quad \text{for any } \langle \delta \mathbf{U}_n \rangle \quad (5.50)$$

or if we replace \mathbf{U}_n with \mathbf{U} for the sake of simplicity of the notations:

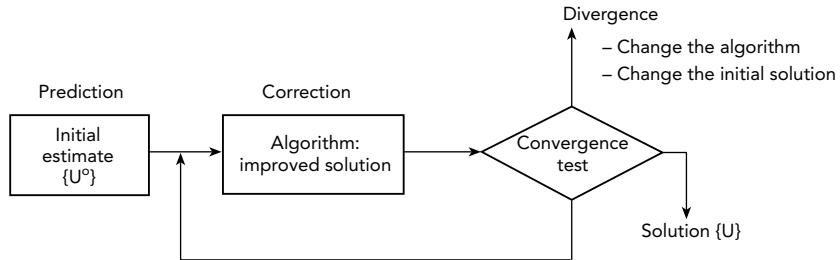
$$[K(\mathbf{U})] \{ \mathbf{U} \} = \{ \mathbf{F} \} \quad \text{or} \quad \{ R(\mathbf{U}) \} = \{ \mathbf{F} \} - [K(\mathbf{U})] \{ \mathbf{U} \} = 0. \quad (5.51a)$$

In certain cases (plasticity), there is only an incremental form of (5.51):

$$[K(\mathbf{U})] \{ \Delta \mathbf{U} \} = \{ \Delta \mathbf{F} \}. \quad (5.51b)$$

Solving the nonlinear system (5.51) involves finding a vector $\{ \mathbf{U} \}$ that renders the residual $\{ R(\mathbf{U}) \}$ as close as possible to zero.

The search for the solution $\{U\}$ is carried out in an iterative manner:



Most of the algorithms lead to the resolution of a system of linear equations at each iteration. Several factors must be taken into account when choosing a resolution algorithm:

- the type of nonlinearity: localized or otherwise, predominant or otherwise;
- the existence of one or several solutions;
- the availability of a method for constructing an approximate solution;
- the precision and rapidity of convergence sought;
- the danger of divergence.

In practice, there is no general method that is valid in all cases; based on experience, we have to adapt the resolution strategy to a given category of problems, using a combination of the following three linearization methods:

- substitution method,
- Newton–Raphson method,
- incremental method.

These three approaches stem from a general method called the “fixed point method” [KRE 88]. Consider the general expression of a nonlinear problem, in the form:

$$\{R(U)\} = \{F\} - [K(U)]\{U\} = \{0\}.$$

The fixed point method establishes a recurrence relation in the form:

$$\{U^{i+1}\} = \{g(U^i)\}, \quad (5.52)$$

$$\begin{aligned} \text{with } \{g(U^i)\} &= \{U^i\} + \omega^i [K_c^i]^{-1} \{R^i\}, \\ &= \{U^i\} + \omega^i [K_c^i]^{-1} (\{F\} - [K^i] \{U^i\}), \end{aligned}$$

where $\{U^i\}$, ω^i denote the solution vector at the iteration “ i ” and a relaxation factor for monitoring the convergence. The correction matrix $[K_c]$ is a function of linearization technique chosen: substitution, Newton–Raphson or something else. The “fixed point” is the solution to the problem toward which the method has to converge:

$$\{U^{i+1}\} = \{U^i\} = \{U\},$$

and which satisfies:

$$\{U\} = \{g(U)\} = \{U\} + \omega^i [K_c]^{-1} \{R(U)\}, \text{ so that } \{R(U)\} = \{0\}.$$

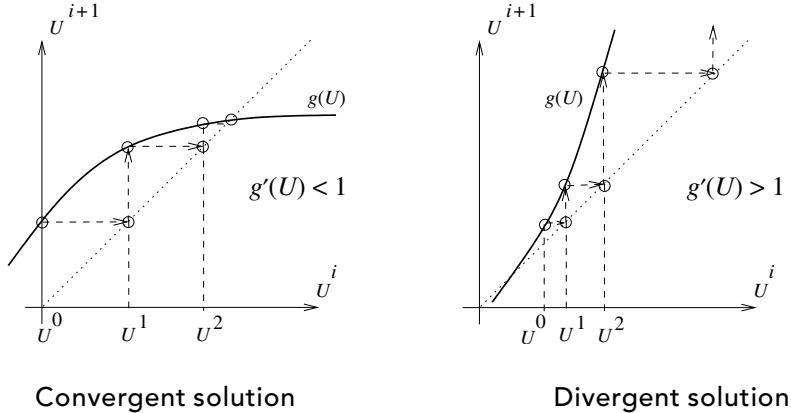


Figure 5.14. Fixed point method

Figure 5.14 respectively illustrates the cases:

- of a scheme converging toward the “fixed point” solution ($g'(u) < 1$),
- of a divergent scheme ($g'(u) > 1$).

5.3.2 SUBSTITUTION METHOD

This method consists of constructing a series of solutions $\{U^0\}, \{U^1\} \dots \{U^i\}$; $\{U^i\}$ is calculated from the previous value $\{U^{i-1}\}$ by solving the linear system:

$$[K(U^{i-1})]\{U^i\} = \{F\}; \quad i = 1, 2, 3 \dots \quad (5.53)$$

which can be written in incremental form by introducing the **residual** $\{R^i\}$:

$$\begin{aligned} \{R^i\} &= \{R(U^{i-1})\} = \{F\} - [K(U^{i-1})]\{U^{i-1}\} \\ &[K(U^{i-1})]\{\Delta U^i\} = \{R^i\} \\ &\{U^i\} = \{U^{i-1}\} + \{\Delta U^i\}. \end{aligned} \quad (5.54)$$

As for the fixed point method, the correction matrix is given by $[K_c] = [K(U^i)]$.

Since the vector $\{U^{i-1}\}$ is known, we can construct the elementary matrices $[k(u^{i-1})]$, assemble them to get $[K(U^{i-1})]$ and finally solve the linear system (5.54) for $\{\Delta U^i\}$ using one of the methods from section 5.2.

The algorithm corresponding to (5.54) is given in Figure 5.15:

Choose an approximate solution $\{U^0\}$, whose value may be equal to zero.

Construct $\{F\}$ by assembling the elementary vectors $\{f\}$.

$i = 1, 2, \dots$ (for each iteration)

For each element

Extract the values $\{u^{i-1}\}$ from $\{U^{i-1}\}$

Calculate $[k(u^{i-1})]$

Calculate the elementary residual $\{r\} = \{f\} - [k]\{u^{i-1}\}$

Assemble:

$[k]$ in $[K]$

$\{r\}$ in $\{R^i\}$.

solve: $[K]\{\Delta U^i\} = \{R^i\}$

Construct the new estimation of the solution:

$$\{U^i\} = \{U^{i-1}\} + \omega \{\Delta U^i\} \quad (5.55)$$

Calculate the norm $\|n\|$ of $\{\Delta U^i\}$ or $\|m\|$ of $\{R^i\}$.

Convergence test using $\|n\|$ or $\|m\|$.

Figure 5.15. Substitution algorithm

Remarks

a) Over-relaxation and under-relaxation

The over-relaxation factor ω , used in (5.55), can often improve the speed of convergence. Its optimal value depends on the problem; it is determined by numerical experimentation. In plasticity problems, for instance, it is between 1.7 and 1.9. The method without over-relaxation corresponds to $\omega = 1$.

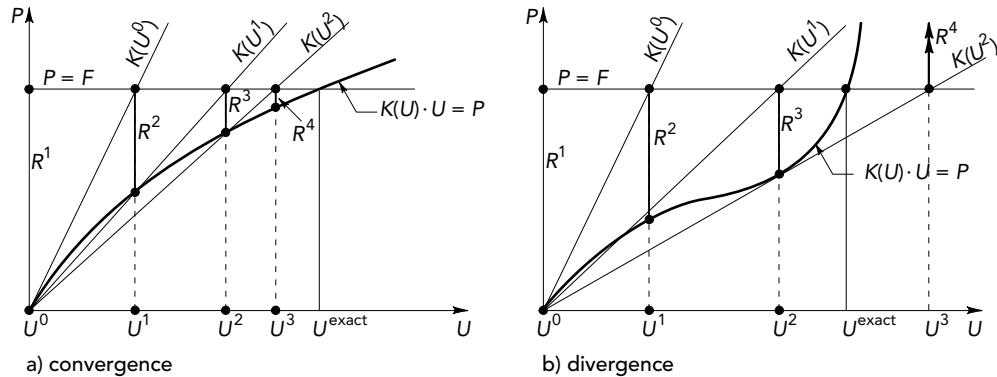
b) Norms

To test the convergence at iteration i , we can use either the maximum norm:

$$\|n\| = \max_j |\Delta U_j|^i \quad \text{or} \quad \|m\| = \max_j |R_j|^i \quad (5.56)$$

or the least squares norm:

$$\|n\| = \sqrt{\langle \Delta U^i \rangle \{ \Delta U^i \}} \quad \text{or} \quad \|m\| = \sqrt{\langle R^i \rangle \{ R^i \}}. \quad (5.57)$$



SUBSTITUTION METHOD: (Algorithm: 5.55)

Figure 5.16. Substitution method

In practice, we often use relative norms:

$$\|n\| = \max_j \left| \frac{\Delta U_j}{U_j} \right|^i \quad (5.58)$$

If U_j is very small in relation to the average value of the terms in $\{U\}$, or equal to zero, we can replace U_j with:

$$\|n\| = \frac{\sqrt{\langle \Delta U^i \rangle \{ \Delta U^i \}}}{\sqrt{\langle U^i \rangle \{ U^i \}}} \quad (5.59)$$

The iterative process is stopped when

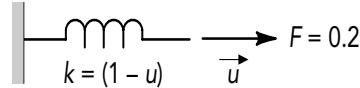
$$\|n\| < \varepsilon$$

with, for instance, for the norm (5.59): $\varepsilon = 10^{-3}$ to 10^{-6} .

For a one-variable problem, algorithm (5.55) is illustrated in Figure 5.16, so as to highlight the possibilities for divergence.

EXAMPLE 5.19. Nonlinear spring: substitution method

Consider a spring subjected to a force $F = 0.2$, whose stiffness k depends on the extension U of the spring (physical nonlinearity):



so that:

$$(1 - U) \cdot U = F.$$

The fixed point method is written:

$$R(U) = F - (1 - U)U$$

$$\text{and } U^{i+1} = g(U^i) = F / (1 - U^i)$$

$$\text{where } g'(U^i) = F / (1 - U^i)^2.$$

Let us take $U^0 = 0$ as an initial estimation, and use algorithm (5.55):

Iteration i	U^{i-1}	$k = 1 - U^{i-1}$	$R^i = F - k(U^{i-1}) \cdot U^{i-1}$	$\Delta U^i = k^{-1} R^i$	$U^i = U^{i-1} + \Delta U^i$	$\ n\ $ (5.58)
1	0	1	0.2	0.2	0.2	1
2	0.2	0.8	0.04	0.05	0.25	0.2
3	0.25	0.75	0.0125	0.0167	0.2667	0.06
4	0.2667	0.733	0.0044	0.006	0.2727	0.02

Exact solution: $0.5 \mp \sqrt{0.05} = 0.2764$ and 0.7236 .

Modified Newton–Raphson algorithm

We reformulate (5.54) by decomposing $[K]$ into a sum of a constant matrix $[K_l]$ and a matrix $[K_{nl}]$, which is a function of U :

$$([K_l] + [K_{nl}(U^{i-1})])\{\Delta U^i\} = \{R^i\}. \quad (5.60)$$

In algorithm (5.55), we have to assemble and decompose $[K]$ at each iteration, which is very costly. By ignoring $[K_{nl}]$ in (5.60), we get:

$$\begin{aligned} [K_l]\{\Delta U^i\} &= \{R^i\} \\ \{U^i\} &= \{U^{i-1}\} + \{\Delta U^i\}. \end{aligned} \quad (5.61)$$

In this case, the fixed point method is written thus:

$$\{U^{i+1}\} = \{g(U^i)\} \quad \text{where} \quad \{g(U^i)\} = \{U^i\} + [K_l]^{-1} \cdot (\{F\} - [K(U^i)]\{U^i\}).$$

The matrix $[K_l]$ is only decomposed once. For each iteration, we need only calculate $\{R^i\}$ and then evaluate $\{\Delta U^i\}$ based on the already decomposed $[K_l]$. The algorithm corresponding to (5.61) is as follows:

Choose an approximate solution $\{U^0\}$, whose value may be equal to zero.

Construct $\{F\}$ by assembling the elementary vectors $\{f\}$.

Construct $[K_l]$ by assembling the elementary “linear” matrices $[k_l]$.

Decompose $[K_l]$.

$i = 1, 2, \dots$ (for each iteration)

For each element

Extract $\{u^{i-1}\}$ from $\{U^{i-1}\}$

Calculate the residual $\{R^i\}$ by assembling the elementary residuals

$$\{r\} = \{f\} - [k]\{u^{i-1}\} \quad (5.62)$$

Solve $[K_l]\{\Delta U^i\} = \{R^i\}$ using the decomposed $[K_l]$

Calculate $\{U^i\} = \{U^{i-1}\} + \{\Delta U^i\}$ (assuming that $\omega = 1$)

Calculate $\|n\|$

Convergence test using $\|n\|$.

This algorithm is illustrated for the case of a one-variable problem in Figure 5.17.

EXAMPLE 5.20. Nonlinear spring: modified Newton–Raphson method

In the previous example:

$$k = (1 - U) = k_l + k_{nl}(U)$$

where

$$k_l = 1$$

$$k_{nl} = -U.$$

In this case, the fixed point method is written as: $U^{i+1} = g(U^i) = (F + U^{i^2})$.

Let us use algorithm (5.62):

Iteration i	U^{i-1}	k_l	$k = 1 - U^{i-1}$	$R^i = F - k \cdot U^{i-1}$	$\Delta U^i = k_l^{-1} R^i$	$U^i = U^{i-1} + \Delta U^i$	$\ n\ $ (5.58)
1	0	1	1	0.2	0.2	0.2	1
2	0.2	1	0.8	0.04	0.04	0.24	0.167
3	0.24	1	0.76	0.0176	0.0176	0.2576	0.068
4	0.2576	1	0.7424	0.0088	0.0088	0.2664	0.033

In algorithm (5.62) the matrix $[K_l]$ is assembled and decomposed only once, whereas in algorithm (5.55), we have to assemble and decompose $[K]$ at each iteration.

Algorithm (5.62) is often employed in the case of slight nonlinearities. However, for highly nonlinear problems, the Newton–Raphson method, presented in the following section, is more commonly used than (5.55) because it generally converges more quickly.

5.3.3 NEWTON–RAPHSON METHOD

Suppose that at iteration $i - 1$ we have obtained an approximation U^{i-1} of the solution such that the residual is non-null.

$$\{R(U^{i-1})\} = \{F\} - [K(U^{i-1})]\{U^{i-1}\} \neq \{0\}. \quad (5.63)$$

At iteration i we seek an approximation U^i of the solution such that:

$$\{R(U^i)\} = \{R(U^{i-1} + \Delta U^i)\} \approx \{0\}. \quad (5.64)$$

The algorithm is obtained by developing this residual into a Taylor series in the vicinity of U^{i-1} :

$$\{R(U^{i-1} + \Delta U^i)\} = \{R(U^{i-1})\} + \left[\frac{\partial R}{\partial U} \right]_{U=U^{i-1}} \{\Delta U^i\} + \dots = \{0\}. \quad (5.65)$$

Hence, by ignoring those terms whose order is greater than one:

$$\begin{aligned} -\left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}} \right] \{ \Delta \mathbf{U}^i \} &= \{ R(\mathbf{U}^{i-1}) \} \\ [K_t(\mathbf{U}^{i-1})] \{ \Delta \mathbf{U}^i \} &= \{ R(\mathbf{U}^{i-1}) \} \\ \{ \mathbf{U}^i \} &= \{ \mathbf{U}^{i-1} \} + \{ \Delta \mathbf{U}^i \}. \end{aligned} \quad (5.66)$$

In this case, the fixed point method is written as:

$$\{ \mathbf{U}^{i+1} \} = \{ g(\mathbf{U}^i) \} \quad \text{where} \quad \{ g(\mathbf{U}^i) \} = \{ \mathbf{U}^i \} + [K_t(\mathbf{U}^i)]^{-1} \cdot (\{ F \} - [K(\mathbf{U}^i)] \{ \mathbf{U}^i \}).$$

The expression of the tangent matrix $[K_t(\mathbf{U}^{i-1})]$ is obtained by deriving expression (5.51) from the residual:

$$[K_t(\mathbf{U})] = -\left[\frac{\partial \mathbf{R}}{\partial \mathbf{U}} \right] = -\left[\frac{\partial \mathbf{F}}{\partial \mathbf{U}} \right] + [K(\mathbf{U})] + \left[\frac{\partial [K(\mathbf{U})]}{\partial \mathbf{U}} \{ \mathbf{U} \} \right]. \quad (5.67)$$

If F is independent of \mathbf{U} :

$$[K_t(\mathbf{U})] = [K(\mathbf{U})] + \left[\frac{\partial [K(\mathbf{U})]}{\partial \mathbf{U}} \{ \mathbf{U} \} \right] \quad (5.68)$$

or indeed, if $(K_t)_{ij}$ and K_{ij} are the components of the matrices $[K_t]$ and $[K]$:

$$(K_t)_{ij} = K_{ij} + \sum_l \frac{\partial K_{il}}{\partial U_j} U_l.$$

The algorithm corresponding to (5.66) is similar to algorithm (5.55); however, $[K]$ is replaced by $[K_t]$. It is represented graphically in Figure 5.17, in the case of a single variable.

EXAMPLE 5.21. Nonlinear spring: Newton–Raphson method

In the previous example:

$$k_t = k + \frac{\partial k}{\partial U} U = (1 - U) + (-1) U = 1 - 2U.$$

Iteration i	U^{i-1}	$k_t = 1 - 2U^{i-1}$	$k = 1 - U^{i-1}$	$R^i = F - kU^{i-1}$	$\Delta U^i = k_t^{-1} R^i$	$U^i = U^{i-1} + \Delta U^i$	$\ n \ $ (5.58)
1	0	1	1	0.2	0.2	0.2	1
2	0.2	0.6	0.8	0.04	0.0667	0.2667	0.25
3	0.2667	0.466	0.7333	0.0044	0.0095	0.2762	0.03

The formulation corresponding to the fixed point method is:

$$U^{i+1} = g(U^i) = \frac{F - U^{i^2}}{(1 - 2U^i)}.$$

The three methods presented in Examples 5.19 and 5.21 can be summarized in the following general form:

$$U^{i+1} = g(U^i) = U^i + K_c^{-1}(F - U^i(1 - U^i)),$$

where:

$$\text{Substitution: } K_c = (1 - U^i)$$

$$\text{Modified Newton: } K_c = 1$$

$$\text{Newton Raphson: } K_c = (1 - 2U^i)$$

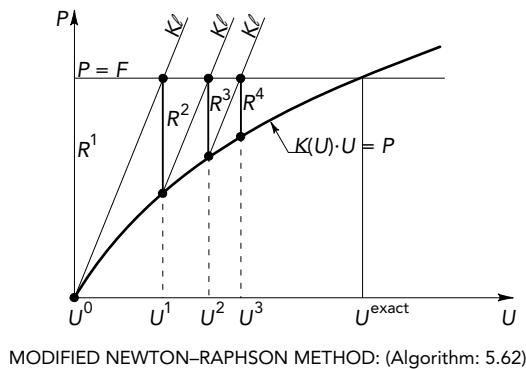
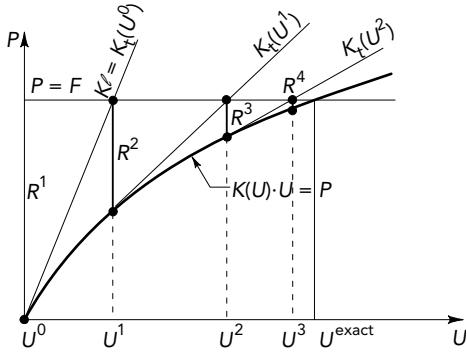
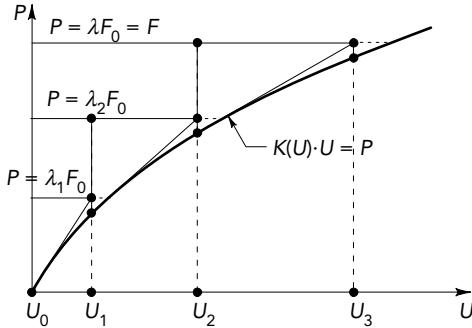


Figure 5.17. Newton method



NEWTON-RAPHSON METHOD: (Algorithm: 5.66)



ONE-ITERATION INCREMENTAL NEWTON-RAPHSON METHOD: (Algorithm: 5.78)

Figure 5.17. (Continued)

Construction of $[K_t]$

The global tangent matrix $[K_t]$ is obtained, in practice, by calculating the elementary tangent matrices $[k_t]$. However, it is tricky to use an expression such as (5.67) to construct $[k_t]$, because we have to derive the explicit expression of $[k(u)]$ in relation to the nodal variables $\{u\}$. It is simpler to work from the non-discretized integral form (4.3):

$$W(\mathbf{u}) = \sum_e W^e = \sum_e \int_{V^e} \delta(\partial \mathbf{u}) R(\mathbf{u}) dV = 0. \quad (5.69)$$

We develop W into a Taylor series in the vicinity of \mathbf{u}^{i-1} to obtain an expression corresponding to (5.65):

$$W(\mathbf{u}^i) = W(\mathbf{u}^{i-1}) + \Delta(W)_{\mathbf{u}=\mathbf{u}^{i-1}} + \dots = 0 \quad (5.70)$$

where $\Delta(W) = \frac{\partial W}{\partial \mathbf{u}} \Delta \mathbf{u}$ is the first variation of W .

Using a finite element approximation of \mathbf{u} , we obtain discretized expressions of $W(\mathbf{u}^{i-1})$ and of $\Delta(W)$, which can be written as:

$$\begin{aligned} W(U^{i-1}) &= \langle \delta U_n \rangle ([K(U^{i-1})] \{U_n^{i-1}\} - \{F\}) \\ &= -\langle \delta U_n \rangle \{R(U^{i-1})\} \end{aligned} \quad (5.71)$$

$$\Delta W(U^{i-1}) = \langle \delta U_n \rangle [K_t(U^{i-1})] \{\Delta U^i\} \quad (5.72)$$

Equation (5.70) is written, in its discrete form:

$$\langle \delta U_n \rangle ([K_t(U^{i-1})] \{\Delta U^i\} - \{R(U^{i-1})\}) = 0 \text{ for any } \langle \delta U_n \rangle \quad (5.73)$$

which is identical to (5.66).

Hence, we can obtain $[K_t]$ by directly discretizing ΔW . Figure 5.18 highlights the two methods for constructing $[K_t]$.

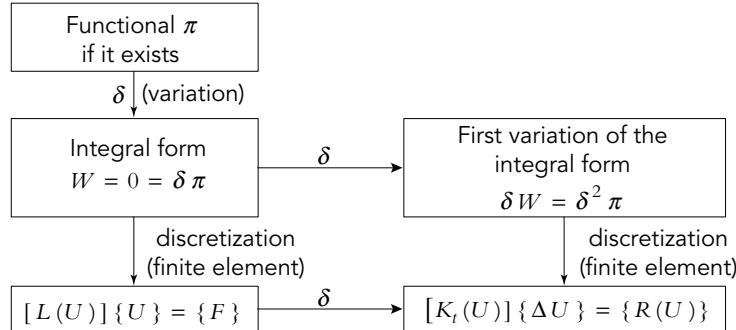


Figure 5.18. Methods for constructing $[K_t]$

EXAMPLE 5.22. The matrices $[k]$ and $[k_t]$ for a beam

The elementary integral form of a beam with large displacements was obtained in Example 4.11:

$$W^e = \int \left(\langle \delta \varepsilon \rangle [H] \{\varepsilon\} - \langle \delta u \quad \delta w \rangle \begin{Bmatrix} f_u \\ f_w \end{Bmatrix} \right) dx = \langle \delta u_n \rangle \{r_n^e\}$$

where:

$$\{\boldsymbol{\varepsilon}\} = \begin{Bmatrix} \boldsymbol{\varepsilon}_m \\ \boldsymbol{\kappa} \end{Bmatrix} = \begin{Bmatrix} \frac{du}{dx} + \frac{1}{2} \left(\frac{dw}{dx} \right)^2 \\ -\frac{d^2 w}{dx^2} \end{Bmatrix}$$

$$[H] = \begin{bmatrix} EA & 0 \\ 0 & EI \end{bmatrix}$$

$$\Delta(W^e) = \int (\langle \delta\varepsilon \rangle [H] \{\delta\varepsilon\} + \langle \Delta(\delta\varepsilon) \rangle [H] \{\varepsilon\}) dx = \langle \delta u_n \rangle [k_t^e] \{\Delta u_n\}$$

where:

$$\{\delta\varepsilon\} = \begin{Bmatrix} \left(\frac{d(\delta u)}{dx} \right) + \frac{dw}{dx} \left(\frac{d(\delta w)}{dx} \right) \\ -\left(\frac{d^2(\delta w)}{dx^2} \right) \end{Bmatrix}, \quad \{\Delta\varepsilon\} = \begin{Bmatrix} \left(\frac{d(\Delta u)}{dx} \right) + \frac{dw}{dx} \left(\frac{d(\Delta w)}{dx} \right) \\ -\left(\frac{d^2(\Delta w)}{dx^2} \right) \end{Bmatrix}$$

$$\{\Delta(\delta\varepsilon)\} = \begin{Bmatrix} \left(\frac{d(\delta w)}{dx} \right) \left(\frac{d(\Delta w)}{dx} \right) \\ 0 \end{Bmatrix}$$

$$\text{where: } \Delta(\delta u) = \Delta(\delta w) = 0.$$

By discretizing using finite elements:

$$u = \langle N_u \rangle \{u_n\}, \quad \delta u = \langle N_u \rangle \{\delta u_n\}, \\ w = \langle N_w \rangle \{w_n\}, \quad \delta w = \langle N_w \rangle \{\delta w_n\}.$$

$$\{\boldsymbol{\varepsilon}\} = ([B_l] + [B_{nl}]) \begin{Bmatrix} u_n \\ w_n \end{Bmatrix} = [B] \{u^e\}$$

$$[B_l] = \begin{bmatrix} \langle N_{u,x} \rangle & 0 \\ 0 & -\langle N_{w,xx} \rangle \end{bmatrix}$$

$$[B_{nl}] = \begin{bmatrix} 0 & \frac{1}{2} (\langle N_{w,x} \rangle \{w_n\} \langle N_{w,x} \rangle) \\ 0 & 0 \end{bmatrix}$$

$$\{\delta\varepsilon\} = ([B_l] + 2[B_{nl}]) \{\delta u^e\}, \quad \{\Delta\varepsilon\} = ([B_l] + 2[B_{nl}]) \{\Delta u^e\}.$$

The expression of the residual is: $\{r^e\} = \int_0^{L^e} [B_l^T + 2B_{nl}^T] [H] \{\varepsilon\} dx$.

Hence: $W^e = \langle \delta u^e \rangle ([k] \{u^e\} - \{f^e\})$

$$[k] = \int ([B_l] + 2[B_{nl}])^T [H] ([B_l] + [B_{nl}]) dx.$$

This non-symmetrical matrix can be reorganized into a symmetrical form (see Example 4.11):

$$[k] = [k_l] + [k_{nl}^1] + [k_{nl}^2] + [k_\sigma]$$

$$[k_l] = \int [B_l]^T [H] [B_l] dx$$

$$[k_{nl}^1] = \int ([B_{nl}]^T [H] [B_l] + [B_l]^T [H] [B_{nl}]) dx$$

$$[k_{nl}^2] = \int [B_{nl}]^T [H] [B_{nl}] dx$$

$$[k_\sigma] = \begin{bmatrix} 0 & 0 \\ 0 & \int \{N_{w,x}\} (EA \varepsilon_m) \langle N_{w,x} \rangle dx \end{bmatrix}$$

$$\Delta W^e = \langle \delta u^e \rangle [k_l] \{\Delta u^e\}$$

$$[k_t] = [k_l] + 2[k_{nl}^1] + 4[k_{nl}^2] + [k_\sigma].$$

EXAMPLE 5.23. Matrix $[k_t]$ and residual vector for an incompressible Navier-Stokes problem

The elementary integral form with penalty on the pressure is written (see Chapter 3) as:

$$W^e = \iint_{A^e} \left(\begin{pmatrix} \delta u & \delta v \end{pmatrix} \begin{cases} \frac{\partial u^2}{\partial x} + \frac{\partial(uv)}{\partial y} \\ \frac{\partial(uv)}{\partial x} + \frac{\partial v^2}{\partial y} \end{cases} + \langle \delta \varepsilon \rangle [H] \{\varepsilon\} \right. \right. \\ \left. \left. - \operatorname{Div}(\delta \mathbf{u}) p + \delta p \left(\frac{p}{\lambda} - \operatorname{Div}(\mathbf{u}) \right) \right) dA \right)$$

$$\Delta W^e = \iint_{A^e} \left(\begin{array}{c} \langle \delta u \quad \delta v \rangle \\ \end{array} \right) \left\{ \begin{array}{l} 2 \frac{\partial u \Delta u}{\partial x} + \frac{\partial (\Delta u \cdot v + u \Delta v)}{\partial y} \\ \frac{\partial (\Delta u \cdot v + u \Delta v)}{\partial x} + 2 \frac{\partial v \Delta v}{\partial y} \end{array} \right\} + \langle \delta \varepsilon \rangle [H] \{ \Delta \varepsilon \} - \text{Div}(\delta \mathbf{u}) \Delta p + \delta p \left(\frac{\Delta p}{\lambda} - \text{Div}(\Delta \mathbf{u}) \right) dA$$

where “ u, v ” are the two components of the velocity field, “ p ” the pressure field, λ the penalty coefficient and:

$$\{ \varepsilon \} = \begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial y} \\ \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{pmatrix}, \quad [H] = \mu \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The velocity field is approximated for a Q4 element with a local approximation for pressure field (constant across the element):

$$\begin{aligned} u &= \sum N_i u_i, \quad v = \sum N_i v_i, \quad p = p_o, \\ u^2 &= \sum N_i u_i^2, \quad v^2 = \sum N_i v_i^2, \quad uv = \sum N_i u_i v_i, \\ u \Delta u &= \sum N_i u_i \Delta u_i, \quad v \Delta v = \sum N_i v_i \Delta v_i, \quad u \Delta v = \sum N_i u_i \Delta v_i, \quad \dots \\ \{ \varepsilon \} &= [B] \{ u_n \}, \quad \text{Div}(\mathbf{u}) = \langle B_{div} \rangle \{ u_n \}, \\ [B] &= \begin{bmatrix} N_{i,x} & 0 & \dots & i = 1, \dots, 4 \\ \dots & N_{i,y} & \dots & \\ N_{i,y} & N_{i,x} & & \end{bmatrix}, \\ \langle B_{div} \rangle &= \langle \dots \quad N_{i,x} \quad N_{i,y} \quad \dots \quad i = 1, \dots, 4 \rangle. \end{aligned}$$

The tangent matrix and the residual vector are calculated by (2×2) -point Gaussian numerical integration. The procedure is described by the following algorithm:

- Initialization: $\{r^e\}_{(8 \times 1)} = \{0\}$, $[k_t^e]_{(8 \times 8)} = [0]$, $\{k_{up}^e\}_{(8 \times 1)} = \{0\}$, $A^e = 0$.
- Loop on the Gaussian integration points:
 - Calculation of $[J]$, $[j]$ and $\det(J) = |J|$
 - $c = \omega_i |J|$
 - Calculation of: $\langle N \rangle$, $\langle N_x \rangle$, $\langle N_y \rangle$, $[B]$, $\langle B_{div} \rangle$
 - $A^e = A^e + c$
 - $\{\bar{N}_i\} = \{r^e\} + c \times \begin{pmatrix} \vdots \\ N_i (\langle N_x \rangle \{u_i^2\} + \langle N_y \rangle \{u_i v_i\}) \\ N_i (\langle N_x \rangle \{u_i v_i\} + \langle N_y \rangle \{v_i^2\}) \\ \vdots \end{pmatrix} + [B]^T [H][B]\{u_n\}$
 - $\bar{N}_i = N_i \times c$
 - $[k_t^e] = [k_t^e] + \begin{bmatrix} \vdots & & & \\ \bar{N}_i (2N_{j,x} u_j + N_{j,y} u_j) & \bar{N}_i N_{j,y} u_j & & i=1, \dots, 4 \\ \dots & \bar{N}_i N_{j,x} v_j & \bar{N}_i (2N_{j,y} v_j + N_{j,x} v_j) & \dots j=1, \dots, 4 \\ & & \vdots & \end{bmatrix} + c \times [B]^T [H][B]$
 - $\{k_{up}^e\} = \{k_{up}^e\} + \{B_{div}\} \times c$
- End of loop
- $[k_t^e] = [k_t^e] - \left(\frac{\lambda}{A^e}\right) \{k_{up}^e\} \langle k_{up}^e \rangle$
- $p_o = \left(\frac{\lambda}{A^e}\right) \langle k_{up}^e \rangle \{u_n\}$
- $\{r^e\} = \{r^e\} - \{k_{up}^e\} p_o$

If there is a functional π , the matrices $[k]$ and $[K_t]$ are symmetrical; indeed if:

$$\pi = \pi(U_1, U_2, \dots, U_n) \quad (5.74)$$

where U_1, U_2, \dots, U_n are the nodal variables,

$$\Delta W = \Delta(\delta \pi) = \langle \delta U_n \rangle [K_t] \{\Delta U_n\} \quad (5.75)$$

and

$$(K_t)_{ij} = \frac{\partial^2 \pi}{\partial U_i \partial U_j} = (K_t)_{ji} = \frac{\partial^2 \pi}{\partial U_j \partial U_i}.$$

5.3.4 INCREMENTAL (OR STEP-BY-STEP) METHOD

The initial solution plays an important role in the previous iterative methods. Depending on the choice of this solution, the methods may diverge or converge toward unacceptable solutions.

The incremental method consists of replacing the solution of

$$[K(U)]\{U\} = \lambda\{F_0\} = \{F\} \quad (5.76)$$

with the successive solution of

$$[K(U_j)]\{U_j\} = \lambda_j\{F_0\} \quad (5.77)$$

where:

$$\lambda_j = \lambda_1, \lambda_2, \dots, \lambda.$$

The initial solution used to calculate U_j is the solution U_{j-1} obtained at the previous stage. Each stage constitutes a nonlinear problem that is solved by one or more iterations of the Newton–Raphson method or the modified Newton–Raphson method.

The incremental method, using **one iteration of the Newton–Raphson** method at each stage, is written as follows for a given level of force λ_j :

$$\begin{aligned} \{R(U_{j-1})\} &= \lambda_{j-1}\{F_0\} - [K(U_{j-1})]\{U_{j-1}\} \\ [K_t(U_{j-1})]\{\Delta U_j\} &= \{R(U_{j-1})\} + (\lambda_j - \lambda_{j-1})\{F_0\} \\ \{U_j\} &= \{U_{j-1}\} + \{\Delta U_j\}. \end{aligned} \quad (5.78)$$

This algorithm is presented graphically for one variable in Figure 5.17.

The incremental method using several iterations of the Newton–Raphson method is written, for a given level of force λ_j :

$$\begin{aligned} [K_t(U_j^{i-1})]\{\Delta U_j^i\} &= \{R(U_j^{i-1})\} + (\lambda_j - \lambda_{j-1})\{F_0\} \\ \{U_j^i\} &= \{U_j^{i-1}\} + \{\Delta U_j^i\} \quad i = 2, 3, \dots \end{aligned} \quad (5.79)$$

For $i = 1$, we use (5.78) directly.

EXAMPLE 5.24. Nonlinear spring: incremental method

Applying the weighting $F = 0.2$ from the previous example in the form of two increases:

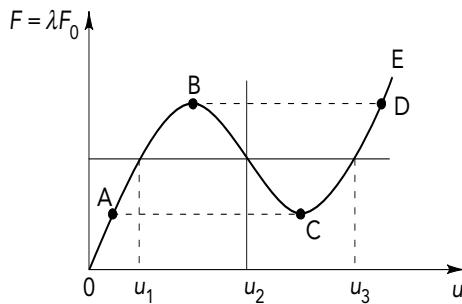
$$\lambda_1 = 0.5 \quad \lambda_2 = 1 \quad F_0 = 0.2.$$

We will use algorithm (5.79):

Step j	$\lambda_j F_0$	Iteration i	U_{j-1} or (U_j^{i-1})	$k(U_{j-1})$	R	$k_t(U_{j-1})$	ΔU	U_j^i
1	0.1	1	0	1	0.1	1	0.1	0.1
2	0.2	1	0.1	0.9	0.11	0.8	0.137	0.2375
	0.2	2	0.2375	0.7625	0.019	0.525	0.0362	0.2735

5.3.5 CHANGING OF INDEPENDENT VARIABLES [BAT 79]

Up until now, we have assumed the forces to be given, with the unknowns being the nodal variables $\{U_n\}$. In certain problems, there are several solutions $\{U_n\}$ corresponding to a given force vector $\lambda \{F_0\}$:



The methods presented hitherto – for instance the Newton–Raphson method – can provide the solution in the regions OAB and EDC . To get the solution in the domain BC , we can consider a component U_l as being known, and the level of force λ as unknown. Indeed, between B and C there is only one vector $\lambda \{F\}$ that corresponds to a given value \bar{U}_l of U_l .

Algorithm (5.79) is written as:

$$[K_t]\{\Delta U\} = \{R\} + \Delta \lambda \{F_0\} \quad (5.80)$$

We transfer $\Delta \lambda$ into the vector of unknowns, and impose $\Delta U_l = 0$:

$$\begin{array}{c} \text{column } l \\ \downarrow \\ \left[\begin{array}{ccc|c} K_{t_{11}} & \dots & K_{t_{1n}} & \\ \vdots & & \vdots & \\ K_{t_{1n}} & \dots & K_{t_{nn}} & \end{array} \right] \left\{ \begin{array}{c} \Delta U_1 \\ \vdots \\ \Delta U_{l-1} \\ \Delta \lambda \\ \Delta U_{l+1} \\ \vdots \\ \Delta U_n \end{array} \right\} = \{R\} \end{array} \quad (5.81)$$

and:

$$U_l = \bar{U}_l.$$

Replacing column l of $[K_t]$ with $\{F_0\}$ changes the structure of this matrix: it becomes asymmetrical and its bandwidth changes. In order to avoid this inconvenience, we can solve (5.80) twice, with different right-hand sides, using the same matrix $[K_t]$.

$$[K_t]\{\Delta U^R\} = \{R\} \quad (5.82)$$

$$[K_t]\{\Delta U^F\} = \{F_0\}.$$

The solution to (5.80) is thus:

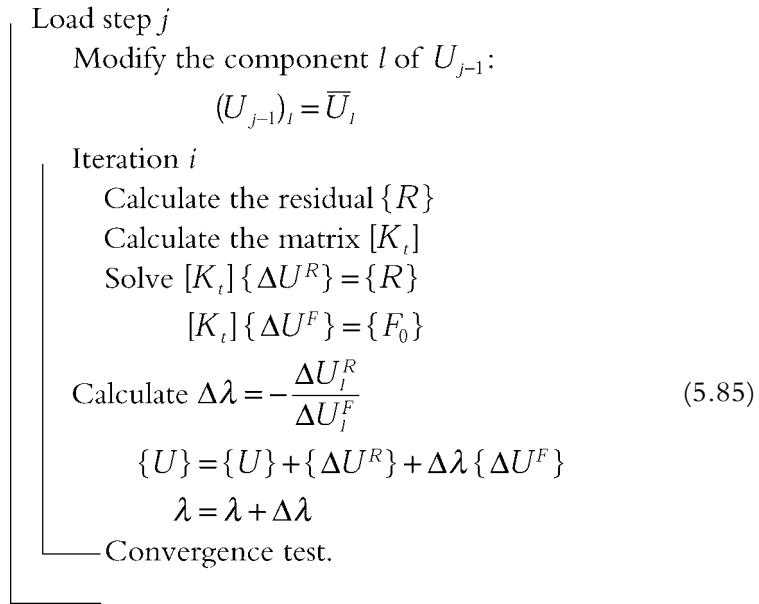
$$\{\Delta U\} = \{\Delta U^R\} + \Delta \lambda \{\Delta U^F\}. \quad (5.83)$$

The unknown $\Delta \lambda$ is obtained by writing:

$$\Delta U_l = 0 = \Delta U_l^R + \Delta \lambda \Delta U_l^F$$

$$\Delta \lambda = -\frac{\Delta U_l^R}{\Delta U_l^F}. \quad (5.84)$$

The corresponding algorithm is written as:



Remark

- It is possible to generalize this technique by privileging the choice of arc length Δs as a control variable instead of U_l . Let U_{j-1}, λ_{j-1} be the solution at step $(j-1)$. We look for the solution at step “ j ” such that:

$$\begin{aligned}\|U_j^i - U_{j-1}\| &= \Delta s, \\ U_j^i &= U_{j-1}^{i-1} + \Delta U^R + \Delta\lambda \cdot \Delta U^F.\end{aligned}$$

Thus we get $\Delta\lambda$ by:

$$\left| U_j^{i-1} - U_{j-1} + \Delta U^R + \Delta\lambda \cdot \Delta U^F \right| = \Delta s.$$

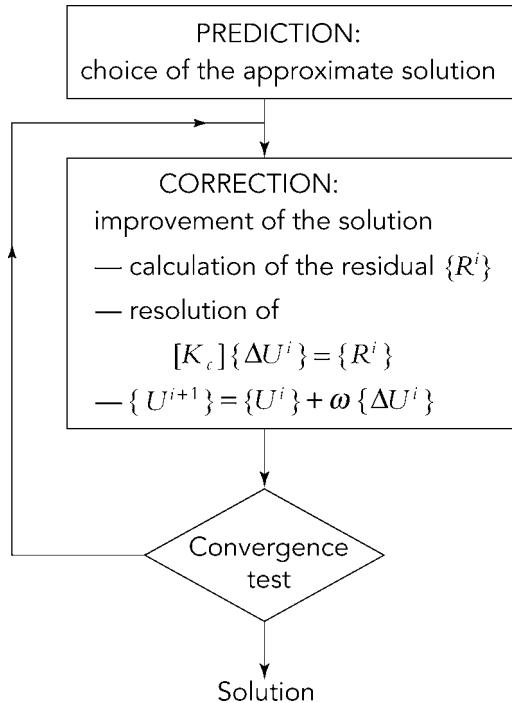
The norm can be calculated by:

$$\left(U_j^{i-1} - U_{j-1} + \Delta U^R \right)^2 + \Delta\lambda^2 \cdot (\Delta U^F)^2 = \Delta s^2.$$

At iteration $i = 1$, we have: $\Delta U^R = 0$, $U_j^{i-1} - U_{j-1} = 0$ and $\Delta\lambda = \frac{\Delta s}{\|\Delta U^F\|}$.

5.3.6 SOLUTION STRATEGY

All the above methods can be boiled down to a single algorithm that, for a given level of force, is represented as follows:



Whatever the method used, the expression of the residual $\{R\}$ remains the same, because it is characteristic of the equation to be solved. On the contrary, the expression of the matrix $[K]$ varies from one method to another, and influences the speed of convergence:

$$[K_e] \equiv [K] \text{ for the substitution method}$$

$$[K_e] \equiv [K_t] \text{ for the modified Newton-Raphson method}$$

$$[K_e] \equiv [K_t] \text{ for the Newton-Raphson method}$$

Iterative methods for linear problems can become a special case of the above procedure by a particular choice of $[K_t]$. The over-relaxation factor ω can often improve the speed of convergence.

For example,

$$[K_c] \equiv \begin{bmatrix} \ddots & & & 0 \\ & K_{ii} & \ddots & \\ 0 & & \ddots & \end{bmatrix} \text{ for the Jacobi method}$$

$$[K_c] \equiv \begin{bmatrix} K_{11} & 0 & \dots & 0 \\ K_{21} & K_{22} & & \\ \vdots & & \ddots & \vdots \\ K_{nl} & \dots & \dots & K_{nn} \end{bmatrix} \text{ for the Gauss-Seidel method.}$$

Figure 5.19 details the algorithm shared between the different methods, in which we have to choose:

- the number and value of each increase in force (or force step) $\Delta\lambda$;
- the maximum number of iterations per force step and the convergence criterion (type of norm and precision required);
- the type of matrix used at each iteration to calculate the correction $\{\Delta U\}$; depending on the value of IMETH, we might use, for example, $[K_l]$ from (5.61), $[K(U^{i-1})]$ from (5.54) and $[K_r(U^{i-1})]$ from (5.68);
- the prediction technique used at the beginning of each increase in force;
- the over- or under-relaxation coefficient ω . A small value of ω offers better control of the convergence in certain cases.

All these decisions depend on the problem in question: it is sometimes necessary to increase load in small steps, for reasons of convergence or because the formulation of the problem is incremental in nature, such as with plasticity; in this case, we may only need a reduced number of iterations per step. Note that the algorithm in Figure 5.19 is practically identical to that in Figure 5.21, which corresponds to the resolution of unsteady and nonlinear problems.

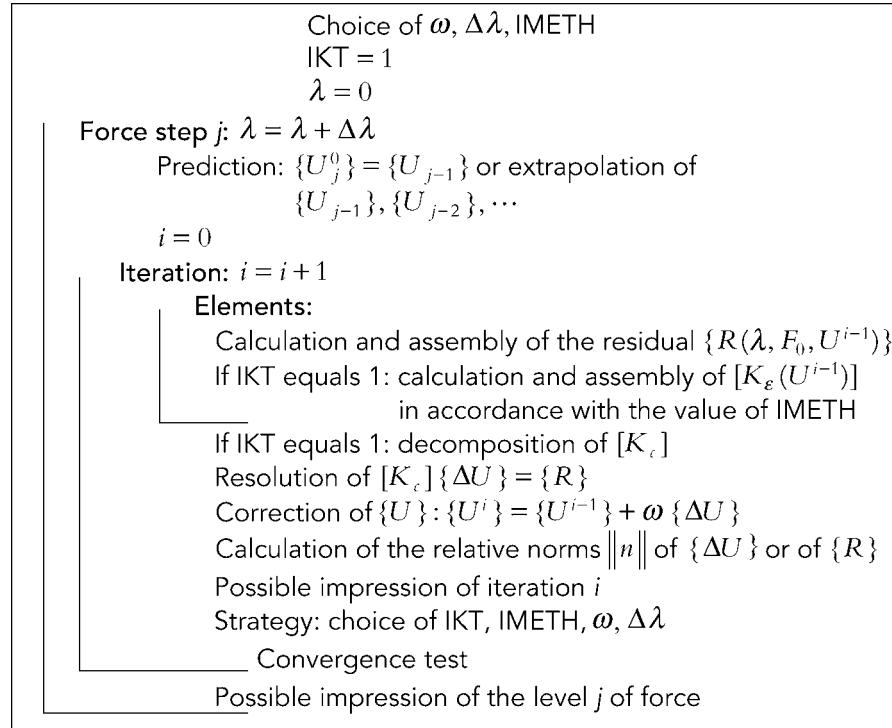


Figure 5.19. General algorithm for solving nonlinear, Newton–Raphson-type problems

5.3.7 CONVERGENCE OF AN ITERATIVE METHOD

An iterative scheme is written, in its general form (section 5.3.1):

$$U^{i+1} = g(U^i),$$

The exact solution satisfies:

$$U = g(U).$$

By subtraction, we express the error between the calculated and exact solutions in the following way:

$$e^{i+1} = U^{i+1} - U = g(U^i) - g(U).$$

A development into a Taylor series enables us to write:

$$g(U) = g(U^i) + \frac{\partial g}{\partial U} \Big|_i (U - U^i) + \frac{\partial^2 g}{\partial U^2} \Big|_i \frac{(U - U^i)^2}{2} + \dots$$

leading to:

$$\begin{aligned} e^{i+1} &= U^{i+1} - U = g(U^i) - g(U^i) - \frac{\partial g}{\partial U} \Big|_{U^i} (U - U^i) - \frac{\partial^2 g}{\partial U^2} \Big|_{U^i} \frac{(U - U^i)^2}{2} \dots \\ &= g'(U^i) e^i + \frac{1}{2} g''(U^i) (e^i)^2 + \dots \end{aligned}$$

A necessary condition to ensure the convergence of the scheme is:

$$|g'(U^i)| < 1 \quad \text{or} \quad |g''(U^i)| < 1 \quad \text{if} \quad g'(U^i) = 0.$$

If we consider the form of the iterative scheme given by equation (5.52):

$$\begin{aligned} \{U^{i+1}\} &= \{U^i\} + [K_c^i]^{-1} (\{F\} - [K^i(U^i)] \{U^i\}), \\ &= \{U^i\} + [K_c^i]^{-1} \{R(U^i)\}. \end{aligned}$$

The exact solution satisfies:

$$\begin{aligned} \{U\} &= \{U\} + [K_c^i]^{-1} (\{F\} - [K(U)] \{U\}), \\ \text{because } \{F\} - [K(U)] \{U\} &= \{R(U)\} = \{0\}. \end{aligned}$$

The error is written:

$$\{e^{i+1}\} = \{e^i\} + [K_c^i]^{-1} (\{R(U^i)\} - \{R(U)\}).$$

A development into a Taylor series of the residual $\{R(U)\}$ around $\{U^i\}$ then yields:

$$\begin{aligned} \{e^{i+1}\} &= \{e^i\} + [K_c^i]^{-1} \left([K_T(U^i)] \{U - U^i\} - \frac{1}{2} \frac{\partial^2 \{R(U)\}}{\partial \{U\}^2} \Big|_{\{U^i\}} \cdot \{U - U^i\}^2 - \dots \right) \\ \text{where: } [K_T] &= - \frac{\partial \{R(U)\}}{\partial \{U\}} \Big|_{\{U^i\}}. \end{aligned}$$

which can be rewritten in the form:

$$\begin{aligned}\{e^{i+1}\} &= \left([I] - [K_c^i]^{-1} [K_T(U^i)] \right) \{e^i\} - \frac{1}{2} [K_c^i]^{-1} \frac{\partial^2 \{R(U)\}}{\partial \{U\}^2} \Big|_{\{U^i\}} \cdot \{e^i\}^2 - \dots \\ &= [g'(U^i)] \{e^i\} - \frac{1}{2} [K_c^i]^{-1} \frac{\partial^2 \{R(U)\}}{\partial \{U\}^2} \Big|_{\{U^i\}} \cdot \{e^i\}^2 - \dots\end{aligned}$$

The convergence of the scheme is then assured, on condition that:

$$|\rho([I] - [K_c^i]^{-1} [K_T^i])| < 1,$$

where ρ , called the spectral radius, denotes the largest of the eigenvalues (in absolute value) of the matrix $[g'(U^i)]$.

For the case described in Example (5.21), depending on the method chosen, we then have:

$$K_T^i = (1 - 2U^i),$$

$$\text{and Substitution } K_c^i = (1 - U^i); \quad g' = \frac{U^i}{1 - U^i}$$

$$\text{Modified Newton } K_c^i = 1; \quad g' = 2U^i$$

$$\text{Newton-Raphson } K_c^i = (1 - 2U^i); \quad g' = 0$$

The order of the convergence is

$$\{e^{i+1}\} = \left([I] - [K_c^i]^{-1} [K_T(U^i)] \right) \{e^i\} - \frac{1}{2} [K_c^i]^{-1} \frac{\partial^2 \{R(U)\}}{\partial \{U\}^2} \Big|_{\{U^i\}} \cdot \{e^i\}^2 - \dots$$

A first-order convergence is written in the form: $\{e^{i+1}\} = [A]\{e^i\}$.

In the case of the Newton-Raphson scheme, we have: $[K_c] = [K_T]$,

which gives us:

$$\{e^{i+1}\} = -\frac{1}{2} [K_T^i]^{-1} \frac{\partial^2 \{R(U)\}}{\partial \{U\}^2} \Big|_{\{U^i\}} \cdot \{e^i\}^2 - \dots$$

Thus, this method presents a second-order convergence, or quadratic for solutions near to the exact solution, as the norm of $\{e^i\}$ must remain “small” in order to ensure convergence.

5.4 Resolution of unsteady systems [BAT 76;

CRA 56; CLO 75]

5.4.1 INTRODUCTION

Spatial discretization of a propagation problem using the finite element method leads to a system of differential equations at time t , usually of the first or second order:

First order

$$[C]\{\dot{U}\} + [K]\{U\} = \{F\} \text{ for } t > t_0 \quad (5.86)$$

with

$$\{U(t_0)\} = \{U_0\}.$$

Second order

$$[M]\{\ddot{U}\} + [C]\{\dot{U}\} + [K]\{U\} = \{F\} \text{ for } t > t_0 \quad (5.87)$$

with

$$\{U(t_0)\} = \{U_0\}; \quad \{\dot{U}(t_0)\} = \{\dot{U}_0\}$$

where:

$$\{\dot{U}\} = \frac{\partial}{\partial t}\{U\}; \quad \{\ddot{U}\} = \frac{\partial^2}{\partial t^2}\{U\} = \frac{\partial}{\partial t}\{\dot{U}\}$$

$[M]$ is the “mass” matrix;

$[C]$ is the “damping” or “thermal capacity” matrix, etc.;

$[K]$ is the “stiffness” or “heat conductivity” matrix, etc.;

$\{F\}$ is the forces vector,

$\{U(t)\}$ is the solution vector sought at time t .

In a linear system, $[M]$, $[C]$, $[K]$ and $\{F\}$ are independent of $\{U\}$ and of its derivatives. In addition, in many physical systems, the matrices $[M]$, $[C]$ and $[K]$ are independent of the time, which implies that the system’s physical parameters do not depend on time either.

In a nonlinear system, $[K]$ and more infrequently $[C]$ and $[M]$ depend on $\{U\}$ and its derivatives.

It is always possible to transform a second-order system (5.87) into a first-order system such as (5.86), as in Example 5.25. However, this operation is rarely used for large systems, because it involves reorganizing huge matrices. In addition, there are efficient methods to solve second-order systems directly.

EXAMPLE 5.25. Transformation of a second-order differential system into a first-order system

In order to transform:

$$[M]\{\ddot{U}\} + [C]\{\dot{U}\} + [K]\{U\} = \{F\}$$

we posit that:

$$\{\dot{U}\} = \{V\}$$

$$[M]\{\dot{V}\} + [C]\{V\} + [K]\{U\} = \{F\}$$

$$[I]\{\dot{U}\} - [I]\{V\} = 0.$$

Hence

$$\begin{bmatrix} [M] & 0 \\ 0 & [I] \end{bmatrix} \begin{Bmatrix} \dot{V} \\ \dot{U} \end{Bmatrix} + \begin{bmatrix} [C] & [K] \\ -[I] & 0 \end{bmatrix} \begin{Bmatrix} V \\ U \end{Bmatrix} = \begin{Bmatrix} F \\ 0 \end{Bmatrix}$$

$$[C']\{\dot{U}'\} + [K']\{U'\} = \{F'\}.$$

where $[I]$ is a unitary matrix.

The number of unknowns has doubled in this new formulation. Furthermore, the matrix $[K']$ is not symmetrical.

The objective is to find a set of functions $\{U(t)\}$ that satisfy (5.86) or (5.87) at all times t , and the initial conditions imposed at $t = t_0$. We present two types of methods:

- direct integration or step-by-step method;
- modes superposition methods.

Direct integration methods consist of numerically constructing a sequential set of solution vectors at the successive times $t_0 + \Delta t$, $t_0 + 2 \Delta t$, ..., $t_0 + n \Delta t$, ...

$$\{U(t_0)\} \rightarrow \{U(t_0 + \Delta t)\} \rightarrow \{U(t_0 + 2 \Delta t)\} \rightarrow \dots \rightarrow \{U(t_0 + n \Delta t)\} \quad (5.88)$$

These methods use finite difference approximations of the derivatives $\{\dot{U}\}$ and $\{\ddot{U}\}$. It is also possible to use a finite element approximation to discretize the differential equation at t [ZIE 00].

However, in superposition methods, we begin by transforming the system of coupled equations (5.86) or (5.87) into a system of uncoupled modal equations. Each modal equation is then integrated explicitly or numerically. The solution sought is a linear combination of the solutions of the uncoupled equations.

5.4.2 DIRECT INTEGRATION METHODS FOR FIRST-ORDER SYSTEMS

5.4.2.1 Explicit Eulerian method

Algorithm

First-order differential systems can be written in the general form:

$$\begin{aligned}\{\dot{U}\} &= \{f(\{U\}, t)\} \text{ for } t > t_0 \\ \{U(t_0)\} &= \{U_0\}.\end{aligned}\quad (5.89)$$

In the case of system (5.86):

$$\{f\} = [C]^{-1} (\{F\} - [K]\{U\}). \quad (5.90)$$

To begin with, we present the Eulerian method adapted to systems written in the form of (5.89); then we adapt it to systems of the form (5.86). We discretize $\{\dot{U}\}$ using the left-offset finite difference formula:

$$\{\dot{U}(t)\} = \{\dot{U}_t\} \approx \frac{1}{\Delta t} (\{U_{t+\Delta t}\} - \{U_t\}). \quad (5.91)$$

The relation (5.89) thus becomes:

$$\{U_{t+\Delta t}\} = \{U_t\} + \Delta t \{f(\{U_t\}, t)\}. \quad (5.92)$$

This Eulerian recurrence formula is said to be **explicit**, because $\{f\}$ does not involve the unknown $\{U_{t+\Delta t}\}$.

EXAMPLE 5.26. Solving a one-variable first-order system using the explicit Eulerian method

Consider the one-variable equation:

$$\left| \begin{array}{l} \frac{du}{dt} + u = 0 \quad t > 0 \quad \text{so that} \quad \frac{du}{dt} = f(u, t) = -u. \\ u_0 = 1 \quad t = 0 \end{array} \right.$$

The algorithm (5.92) leads to:

$$u_{t+\Delta t} = u_t - \Delta t \cdot u_t = (1 - \Delta t) u_t$$

so at time $t = n\Delta t$:

$$u_{n\Delta t} = (1 - \Delta t)^n u_0.$$

The value of $u_{n\Delta t}$ remains bounded as n tends toward infinity if:

$$\begin{aligned} |1 - \Delta t| &\leq 1: \\ -1 &\leq 1 - \Delta t \text{ so } \Delta t \leq 2. \end{aligned}$$

In addition, in order to avoid oscillations of the solution, the following condition must be satisfied:

$$0 \leq 1 - \Delta t \text{ so } \Delta t \leq 1.$$

t	0	0.1	0.2	0.3	0.4	0.5	0.6
$\Delta t = 0.2$	1	—	0.800	—	0.640	—	0.512
$\Delta t = 0.1$	1	0.900	0.810	0.729	0.656	0.590	0.534
<i>Exact</i> (e^{-t})	1	0.905	0.819	0.741	0.670	0.607	0.549

In practice, in order to solve (5.86), Euler's algorithm is reorganized using (5.90), as below:

$$[C]\{U_{t+\Delta t}\} = \Delta t \{F_t\} + ([C] - \Delta t [K])\{U_t\}. \quad (5.93)$$

The incremental form is written as:

$$\begin{aligned} [C]\{\Delta U\} &= \Delta t(\{F_t\} - [K]\{U_t\}) = \{R_t\} \\ \{U_{t+\Delta t}\} &= \{U_t\} + \{\Delta U\}. \end{aligned} \quad (5.94)$$

The relation (5.93) or (5.94) is solved using the method described in section 5.2.3.1 for constant $[C]$. The operations corresponding to algorithm (5.94) are detailed in Figure 5.20. The numerical efficiency of (5.94) is greatly improved if we can choose a diagonal matrix $[C]$.

```

 $t = t_0$ 
Define  $\{U_0\}$ ,  $\Delta t$ 
Construct the matrix  $[C]$ 
Triangularize  $[C]$ 
For each time-step
 $t = t + \Delta t$ 
Construct  $\{R_t\} = \Delta t(\{F_t\} - [K]\{U_t\})$ 
Based on the triangularized  $[C]$ , solve
 $[C]\{\Delta U\} = \{R_t\}$ 
Calculate:  $\{U_{t+\Delta t}\} = \{U_t\} + \{\Delta U\}$ 

```

Figure 5.20. Explicit Eulerian algorithm: incremental form (5.94)

Remark

If $[C]$ is symmetrical, we can write (sections 5.2.3.4 and 5.2.3.5):

$$[C] = [L][S] = [L][D][L]^T.$$

Stability

As we showed in Example 5.26, the explicit Eulerian algorithm is only stable if Δt is less than a critical value Δt_c . Indeed, (5.93) can be rewritten as:

$$\begin{aligned} \{U_{t+\Delta t}\} &= [B]\{F_t\} + [A]\{U_t\} \\ [B] &= \Delta t[C]^{-1}; \quad [A] = [I] - \Delta t[C]^{-1}[K] \end{aligned} \tag{5.95}$$

so that by recurrence:

$$\begin{aligned} \{U_{t_0+n\Delta t}\} &= [A]^n \{U_0\} + [A]^{n-1} [B]\{F_0\} + [A]^{n-2} [B]\{F_{\Delta t}\} + \cdots + \\ &\quad + [B]\{F_{t_0+(n-1)\Delta t}\}. \end{aligned} \tag{5.96}$$

For $\{U_{t_0+n\Delta t}\}$ to be limited as n tends toward infinity, the spectral radius $\rho(A)$ of the matrix A must be less than or equal to one; it is defined by:

$$\rho(A) = \max |\lambda_i| \leq 1 \tag{5.97}$$

where λ_i are the eigenvalues of $[A]$. If l_i are the eigenvalues of $[C]^{-1}[K]$:

$$\lambda_i = 1 - \Delta t l_i. \tag{5.98}$$

The stability condition (5.97) is finally written, assuming $[C]$ and $[K]$ to be positive definite ($l_i > 0$):

$$\begin{aligned} |\lambda_i| &\leq 1 \\ -1 \leq 1 - \Delta t l_{\max} &\text{ so that } \Delta t \leq \frac{2}{l_{\max}} \\ l_{\max} &= \max(l_i). \end{aligned} \quad (5.99)$$

In addition, in order to avoid oscillations of the $\{U\}$ (see Example 5.26), the following condition must be satisfied:

$$\begin{aligned} \lambda_i &> 0, \\ 0 \leq 1 - \Delta t l_{\max} &\text{ so that } \Delta t \leq \frac{1}{l_{\max}} = \Delta t_c. \end{aligned} \quad (5.100)$$

Note that the stability condition is not a criterion of the precision of the solution. In addition to the stability condition (5.100), a value of Δt must be chosen for the truncation error in the finite difference formula (5.91) to be acceptable. Depending on the case, the minimal admissible value of Δt may be guided simultaneously by the stability criterion and by an acceptable degree of precision.

EXAMPLE 5.27. Resolution of a two-variable first-order system using the explicit Eulerian method

Let us solve system (5.86) using the explicit Eulerian method, with:

$$[C] = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}; \quad [K] = \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}; \quad \{F\} = \begin{Bmatrix} 2 \\ 3 \end{Bmatrix}$$

The stability condition (5.100) is:

$$\Delta t \leq \frac{1}{l_{\max}}$$

where l_{\max} is the largest eigenvalue of

$$\begin{aligned} [C]^{-1}[K] &= \left[\begin{array}{cc} 1 & 1 \\ 1 & 2 \end{array} \right]^{-1} \left[\begin{array}{cc} 1 & 1 \\ 1 & 3 \end{array} \right] \\ &= \left[\begin{array}{cc} 2 & -1 \\ -1 & 2 \end{array} \right] \left[\begin{array}{cc} 1 & 1 \\ 1 & 3 \end{array} \right] = \left[\begin{array}{cc} 1 & -1 \\ 0 & 2 \end{array} \right] \end{aligned}$$

$l_{\max} = 2$. Gives

$$\Delta t \leq \frac{1}{2}.$$

We choose $\Delta t = 0.1$ and apply the algorithm of Figure 5.20:

$$\begin{aligned} t &= 0 \\ \{U_0\} &= \{0\} \quad \Delta t = 0.1 \\ [C] &= \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}. \end{aligned}$$

Step 1:

$$\begin{aligned} t &= 0.1 \\ \{R_t\} &= 0.1 \left(\begin{bmatrix} 2 \\ 3 \end{bmatrix} - \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0.2 \\ 0.3 \end{bmatrix} \\ [C]\{\Delta U\} &= \begin{bmatrix} 0.2 \\ 0.3 \end{bmatrix} \rightarrow \{\Delta U\} = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} \\ \{U_1\} &= \{U_0\} + \{\Delta U\} = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}. \end{aligned}$$

Step 2:

$$\begin{aligned} t &= 0.2 \\ \{R_t\} &= 0.1 \left(\begin{bmatrix} 2 \\ 3 \end{bmatrix} - \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix} \right) = \begin{bmatrix} 0.18 \\ 0.26 \end{bmatrix} \\ [C]\{\Delta U\} &= \begin{bmatrix} 0.18 \\ 0.26 \end{bmatrix} \rightarrow \{\Delta U\} = \begin{bmatrix} 0.10 \\ 0.08 \end{bmatrix} \\ \{U_2\} &= \{U_1\} + \{\Delta U\} = \begin{bmatrix} 0.20 \\ 0.18 \end{bmatrix}. \end{aligned}$$

The solution obtained is

t	u_1	u_2
0	0	0
0.1	0.1	0.1
0.2	0.20	0.18
0.3	0.298	0.244
0.4	0.393	0.295
0.5	0.483	0.336
0.6	0.568	0.369
0.7	0.648	0.395
0.8	0.723	0.416
0.9	0.792	0.433
1.0	0.856	0.446
2.0	1.263	0.494
3.0	1.416	0.499
4.0	1.471	0.5
5.0	1.489	0.5
6.0	1.5	0.5
10.0	1.5	0.5

Note that for an infinite t , $\{U\}$ tends toward the solution of

$$[K]\{U\} = \{F\}.$$

The exact solution of the problem is given in Example 5.36.

Nonlinear problems

The explicit Eulerian method applies directly to nonlinear problems in which $[K]$ is a function of $\{U\}$ but $[C]$ is constant. Since the equations are evaluated at time t , the matrix $[K]$ is explicitly calculated. We need only replace $[K]$ with $[K(U_t)]$ in (5.93) and (5.94); for instance, (5.94) becomes:

$$\begin{aligned}[C]\{\Delta U\} &= \Delta t(\{F_t\} - [K(U_t)]\{U_t\}) \\ \{U_{t+\Delta t}\} &= \{U_t\} + \{\Delta U\}.\end{aligned}\tag{5.101}$$

Because of the nonlinearity of $[K]$, the stability criterion is more difficult to explicitly define than in the case of a linear problem.

5.4.2.2 Implicit Eulerian method

Algorithm

This method consists of writing (5.89) **at time $t + \Delta t$** , and using the right-offset finite difference formula:

$$\{\dot{U}_{t+\Delta t}\} \approx \frac{1}{\Delta t}(\{U_{t+\Delta t}\} - \{U_t\}).\tag{5.102}$$

The implicit Euler recurrence formula for system (5.89) is written as :

$$\{U_{t+\Delta t}\} = \{U_t\} + \Delta t \{f(\{U_{t+\Delta t}\}, t + \Delta t)\}. \quad (5.103)$$

In this case, the expression of $\{f\}$ involves the unknown vector $\{U_{t+\Delta t}\}$. The adaptation of (5.103) to systems such as (5.86) can be written as:

$$[\bar{K}] \{U_{t+\Delta t}\} = [\bar{R}_{t+\Delta t}] \quad (5.104)$$

where:

$$[\bar{K}] = [C] + \Delta t [K]$$

$$\{\bar{R}_{t+\Delta t}\} = \Delta t \{F_{t+\Delta t}\} + [C] \{U_t\}$$

or in incremental form

$$[\bar{K}] \{\Delta U\} = \{\bar{R}_{t+\Delta t}\} - [\bar{K}] \{U_t\} = \{R_{t+\Delta t}\} \quad (5.105)$$

where:

$$\{R_{t+\Delta t}\} = \Delta t (\{F_{t+\Delta t}\} - [K] \{U_t\})$$

$$\{U_{t+\Delta t}\} = \{U_t\} + \{\Delta U\}.$$

The algorithm corresponding to (5.105) is that given in Figure 5.21, in which $\alpha = 1$.

EXAMPLE 5.28. Resolution of a one-variable first-order problem using the implicit Eulerian method

The implicit algorithm (5.104), corresponding to Example 5.26, is written as:

$$(1 + \Delta t) u_{t+\Delta t} = u_t$$

$$\text{or} \quad u_{t+\Delta t} = \frac{1}{1 + \Delta t} u_t$$

$$\text{so that} \quad u_{n\Delta t} = \left(\frac{1}{1 + \Delta t} \right)^n u_0.$$

Because $\frac{1}{1 + \Delta t} < 1$ for any $\Delta t > 0$, the algorithm is unconditionally stable. The values of $u(t)$ are:

t	0	0.1	0.2	0.3	0.4	0.5	0.6
$\Delta t = 0.2$	1	—	0.833	—	0.694	—	0.579
$\Delta t = 0.1$	1	0.909	0.826	0.751	0.683	0.621	0.565
<i>Exact</i> (e^{-t})	1	0.905	0.819	0.741	0.670	0.607	0.549

EXAMPLE 5.29. Resolution of a two-variable, first-order problem using the implicit Eulerian method

Let us use algorithm (5.105) to solve the system defined in Example 5.27:

$$[\bar{K}] = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} + 0.1 \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 1.1 & 1.1 \\ 1.1 & 2.3 \end{bmatrix}.$$

Step 1:

$$\{R_{t+\Delta t}\} = 0.1 \left(\begin{bmatrix} 2 \\ 3 \end{bmatrix} - \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \right) = \begin{bmatrix} 0.2 \\ 0.3 \end{bmatrix}$$

$$\begin{bmatrix} 1.1 & 1.1 \\ 1.1 & 2.3 \end{bmatrix} \{\Delta U\} = \begin{bmatrix} 0.2 \\ 0.3 \end{bmatrix} \rightarrow \{\Delta U\} = \begin{bmatrix} 0.0984848 \\ 0.0833333 \end{bmatrix}$$

$$\{U_1\} = \begin{bmatrix} 0.0984848 \\ 0.0833333 \end{bmatrix}.$$

Solution:

t	u_1	u_2
0	0	0
0.1	0.0985	0.0833
0.2	0.1943	0.1528
0.3	0.2867	0.2106
0.4	0.3751	0.2589
0.5	0.4591	0.2991
0.6	0.5385	0.3326
0.7	0.6132	0.3605
0.8	0.6833	0.3837
0.9	0.7487	0.4031
1.0	0.8097	0.4192
2.0	1.2158	0.4870
3.0	1.3875	0.4979
4.0	1.4562	0.4997
5.0	1.4830	0.4999
6.0	1.4934	0.5
10.0	1.5	0.5

EXAMPLE 5.30. Notion of the precision of a scheme

Consider the relation: $\frac{d\bar{u}}{dt} = F - k\bar{u}$, where $\bar{u}(t=0) = \bar{u}_o$.

For a constant value of F , this relation can be rewritten in the homogeneous form:

$$\frac{du}{dt} = -ku, \quad \text{where } u = \left(\bar{u} - \frac{F}{k} \right) \quad \text{and} \quad u(t=0) = u_0 = \bar{u}_0 - \frac{F}{k}.$$

Consider a numerical scheme in the form:

$$u_{t+\Delta t} = A \cdot u_t$$

where: $A = 1 - k\Delta t$: explicit

$$= \frac{1}{1 + k\Delta t} : \text{implicit}$$

$$= \frac{1 - k\Delta t/2}{1 + k\Delta t/2} : \text{Crank–Nicholson (see section 5.4.2.3)}$$

The exact solution is: $u_{t+\Delta t} = u_t \cdot e^{-k\Delta t} = A_{ex} \cdot u_t$ where $A_{ex} = e^{-k\Delta t}$.

We determine the constant k_h associated with the numerical scheme, such that we have:

$$e^{-k_h \Delta t} = A \quad \text{and} \quad \frac{du}{dt} = -k_h \cdot u.$$

The numerical scheme is therefore representative of a physical system controlled by a value of k_h different from k . In order to ensure the convergence of the scheme, we must verify that: $(k_h - k) = o(\Delta t^n)$, $n \geq 1$.

In the case of the explicit scheme, we have:

$$e^{-k_h \Delta t} = (1 - k\Delta t), \quad \text{so} \quad k_h = k \left(1 + k \frac{\Delta t}{2} + \dots \right); \quad k_h > k.$$

(This is obtained by taking log of both sides and expanding in a Taylor series)
For the implicit case:

$$e^{-k_h \Delta t} = \frac{1}{(1 + k\Delta t)}, \quad \text{so} \quad k_h = k \left(1 - k \frac{\Delta t}{2} + k^2 \frac{\Delta t^2}{3} - \dots \right); \quad k_h < k.$$

Hence, the explicit scheme underestimates the solution, whereas the implicit scheme overestimates it. Thus: $u_{t+\Delta t} < u_{t+\Delta t}^{ex}$: explicit (Example 5.22)

$$u_{t+\Delta t} > u_{t+\Delta t}^{ex} : \text{implicit (Example 5.24)}$$

For the particular case where $u_0 = 0$ and $F > 0$, the numerical solution is now overestimated by the explicit scheme and underestimated by the implicit scheme (Examples 5.28, 5.29 and 5.36).

Stability

Expression (5.104) can be written in a form similar to (5.95):

$$\begin{aligned} \{U_{t+\Delta t}\} &= [B]\{F_{t+\Delta t}\} + [A]\{U_t\} \\ [B] &= \Delta t([C] + \Delta t[K])^{-1} \\ [A] &= ([I] + \Delta t[C]^{-1}[K])^{-1}. \end{aligned} \quad (5.106)$$

The stability condition uses the spectral radius (5.181) of $[A]$:

$$\rho(A) \leq 1 \quad (5.107)$$

so

$$\text{Max} |1 + \Delta t \cdot l_i| \geq 1 \quad (5.108)$$

l_i being the eigenvalues of $[C]^{-1}[K]$ that are positive if $[C]$ and $[K]$ are positive definite; hence (5.107) is always verified. Thus, the implicit Eulerian method is unconditionally stable.

Nonlinear problems

For nonlinear problems in which $[C]$ is constant and $[K]$ is a function of $\{U\}$, for each time-step, we have to solve the nonlinear problem (5.104), which is of the form (5.51). For this, we use one of the methods of section 5.3. The correction $\{\Delta U^i\}$ of $\{U_{t+\Delta t}\}$ over the course of an iteration is obtained by solving the system:

$$\begin{aligned} [K_{nl}]\{\Delta U^i\} &= \{R_{nl}\}, \\ \text{with } \{U_{t+\Delta t}^i\} &= \{U_{t+\Delta t}^{i-1}\} + \{\Delta U^i\}, \end{aligned} \quad (5.109)$$

where

$$\begin{aligned} \{R_{nl}\} &= \{\bar{R}_{t+\Delta t}\} - [\bar{K}(U_{t+\Delta t}^{i-1})]\{U_{t+\Delta t}^{i-1}\} \\ &= \Delta t(\{F_{t+\Delta t}\} - [K(U_{t+\Delta t}^{i-1})]\{U_{t+\Delta t}^{i-1}\}) + [C](\{U_t\} - \{U_{t+\Delta t}^{i-1}\}). \end{aligned} \quad (5.110)$$

For $i = 1$: $\{U_{t+\Delta t}^0\} = \{U_t\}$.

In the substitution method (section 5.3.2)

$$[K_{nl}] = [\bar{K}] = [C] + \Delta t[K(U_{t+\Delta t}^{i-1})]. \quad (5.111)$$

In the Newton–Raphson method (section 5.3.3)

$$[K_{nl}] = [K_t] = [\bar{K}] + \Delta t \left[\frac{\partial K}{\partial U} \cdot U \right]_{t+\Delta t}^{i-1}. \quad (5.112)$$

In practice, constructing $[K_t]$ by derivation of $[K]$ is often a complicated task. It is preferable to discretize the first variation $\Delta(W)$ of the integral form W (see section 5.3.3), $[K_t]$ being associated with the expression of $\Delta W(U_{t+\Delta t}^{i-1})$ (see Figure 5.18).

5.4.2.3 Semi-implicit Eulerian method

Algorithm

This method consists of writing (5.89) at time $t + \alpha \Delta t$, where $0 \leq \alpha \leq 1$. Euler's formula is then:

$$\{U_{t+\Delta t}\} = \{U_t\} + \Delta t \{f(\{U_{t+\alpha \Delta t}\}, t + \alpha \Delta t)\} \quad (5.113)$$

where: $\{U_{t+\alpha \Delta t}\} = \alpha \{U_{t+\Delta t}\} + (1 - \alpha) \{U_t\}$.

We come back to the explicit Eulerian method when $\alpha = 0$, and the implicit Eulerian method when $\alpha = 1$.

The (5.104)-type algorithm is written as:

$$[\bar{K}] \{U_{t+\Delta t}\} = \{\bar{R}_{t+\Delta t}\} \quad (5.114)$$

where:

$$[\bar{K}] = [C] + \alpha \Delta t [K]$$

$$\{\bar{R}_{t+\Delta t}\} = \Delta t (\alpha \{F_{t+\Delta t}\} + (1 - \alpha) \{F_t\} - (1 - \alpha) [K] \{U_t\}) + [C] \{U_t\}.$$

The (5.105)-type algorithm is written as:

$$[\bar{K}] \{\Delta U\} = \{R_{t+\Delta t}\} \quad (5.115)$$

where:

$$\begin{aligned} \{R_{t+\Delta t}\} &= \Delta t (\alpha \{F_{t+\Delta t}\} + (1 - \alpha) \{F_t\} - [K] \{U_t\}) \\ \{U_{t+\Delta t}\} &= \{U_t\} + \{\Delta U\}. \end{aligned}$$

The operations required to implement this algorithm are described in Figure 5.21.

Stability

Expression (5.114) can be written in a form similar to (5.95):

$$\{U_{t+\Delta t}\} = [B] (\alpha \{F_{t+\Delta t}\} + (1 - \alpha) \{F_t\}) + [A] \{U_t\} \quad (5.116)$$

$$[B] = [\bar{K}]^{-1} \quad (5.117)$$

$$[A] = [\bar{K}]^{-1} ([C] - (1 - \alpha) \Delta t [K]). \quad (5.118)$$

If $[C]$ and $[K]$ are positive definite, the stability condition $\rho(A) \leq 1$ is written as:

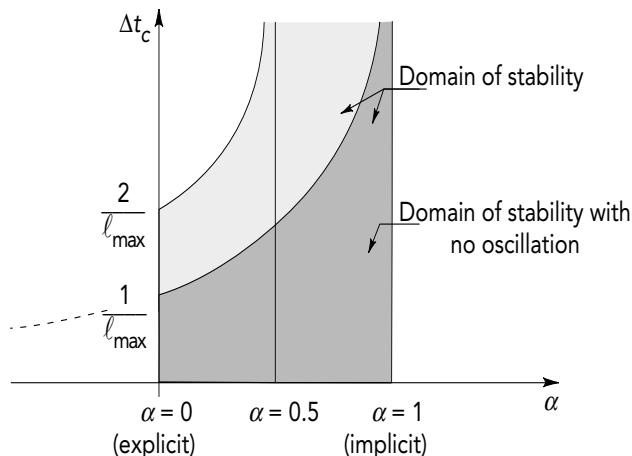
$$(1 - 2\alpha) \Delta t l_{\max} \leq 2. \quad (5.119)$$

where l_{\max} is the largest eigenvalue of $[C]^{-1} [K]$.

This condition is always satisfied when $\alpha \geq 0.5$. For $0 \leq \alpha < 0.5$, condition (5.119) is written as:

$$\Delta t \leq \frac{2}{(1 - 2\alpha) l_{\max}} \equiv \Delta t_c \quad (5.120)$$

The domain of stability with no oscillation is defined by $(1 - \alpha) \Delta t l_{\max} < 1$.



When $\alpha = 0.5$, we get the very widely-used Crank–Nicholson semi-implicit algorithm:

$$\{U_{t+\Delta t}\} = \{U_t\} + \frac{\Delta t}{2} [C]^{-1} \left(\{F_t\} - [K]\{U_{t+\frac{\Delta t}{2}}\} \right),$$

$$\text{where } \{U_{t+\frac{\Delta t}{2}}\} = 0.5 (\{U_t\} + \{U_{t+\Delta t}\}).$$

By choosing:

$$\{U_{t+\frac{\Delta t}{2}}\} = \{U_t\} + \frac{\Delta t}{2} [C]^{-1} (\{F_t\} - [K]\{U_t\})$$

we obtain an explicit version of the Crank–Nicholson algorithm.

Nonlinear problems

The algorithm for solving nonlinear problems is similar to (5.109). The residual corresponding to (5.115) is:

$$\begin{aligned} \{R_{nl}\} = & \Delta t(\alpha\{F_{t+\Delta t}\} + (1-\alpha)\{F_t\} - (1-\alpha)[K(U_t)]\{U_t\} \\ & - \alpha[K(U_{t+\Delta t}^{i-1})]\{U_{t+\Delta t}^{i-1}\}) + [C](\{U_t\} - \{U_{t+\Delta t}^{i-1}\}). \end{aligned} \quad (5.121)$$

The matrix $[K_{nl}]$ is expressed as:

— Substitution method:

$$[K_{nl}] = [\bar{K}] = [C] + \alpha \Delta t [K(U_{t+\Delta t}^{i-1})]. \quad (5.122)$$

— Newton–Raphson method:

$$[K_{nl}] = [K_t] = [\bar{K}] + \alpha \Delta t \left[\frac{\partial K}{\partial U} \cdot U \right]_{t+\Delta t}^{i-1} \quad (5.123)$$

where $[\bar{K}]$ is given by (5.114).

The operations required to implement the above algorithm are described in Figure 5.21. They are similar to the operations corresponding to the Newton–Raphson algorithm given in Figure 5.19. We need only interpret the increase in force as a time-step. If $[C] = 0$, in the present algorithm, we come back to the nonlinear algorithm for unsteady problems shown in Figure 5.19.

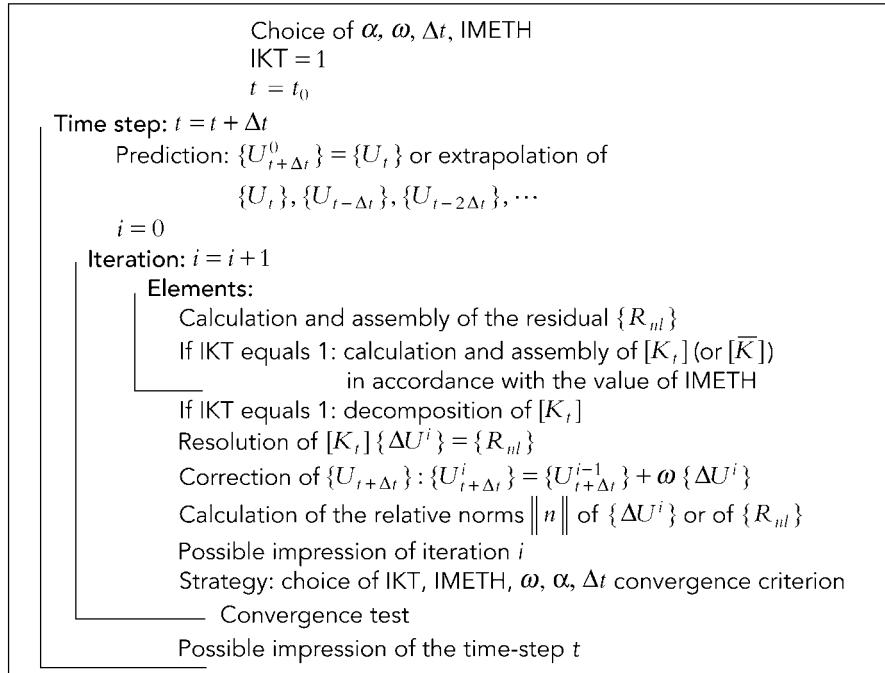


Figure 5.21. Semi-implicit Eulerian algorithm for solving unsteady nonlinear problems, using the Newton–Raphson method

5.4.2.4 Prediction-correction methods [CRA 56, RAL 65]

In these methods, the operations corresponding to each time-step are divided into two phases: the prediction phase gives a preliminary approximation of $\{U_{t+\Delta t}\}$ based on $\{U_t\}$, $\{U_{t-\Delta t}\}$, ..., using an explicit technique; the correction phase improves $\{U_{t+\Delta t}\}$ by way of an implicit technique with one or more iterations.

Prediction formulae (explicit)

For systems similar in form to (5.89), these formulae are written

$$\{U_{t+\Delta t}^0\} = \{U_t\} + \Delta t \sum_{j=1}^n b_j \{f(U_{t+\Delta t-j\Delta t}, t + \Delta t - j \Delta t)\} \quad (5.124)$$

For the formulation (5.86):

$$[C]\{\Delta U\} = \Delta t \sum_{j=1}^n b_j \{R_j\}; \quad \{U_{t+\Delta t}\} = \{U_t\} + \{\Delta U\} \quad (5.125)$$

$$\{R_j\} = \{F_{t+\Delta t-j\Delta t}\} - [K(U_{t+\Delta t-j\Delta t})] \{U_{t+\Delta t-j\Delta t}\}. \quad (5.126)$$

The coefficients b_j depend on the formula chosen:

	n	Error	b_1	b_2	b_3	b_4
Second-order formula (explicit Eulerian method)	1	$O(\Delta t^2)$	1	0	0	0
Third-order formula	2	$O(\Delta t^3)$	$\frac{3}{2}$	$-\frac{1}{2}$	0	0
Fourth-order formula	3	$O(\Delta t^4)$	$\frac{23}{12}$	$-\frac{16}{12}$	$\frac{5}{12}$	0
Fifth-order formula	4	$O(\Delta t^5)$	$\frac{55}{24}$	$-\frac{59}{24}$	$\frac{37}{24}$	$-\frac{9}{24}$

Remark

The coefficients b_j are calculated based on developments into a Taylor series:

$$\begin{aligned}\{U_{t+\Delta t}\} &= \{U_t\} + \Delta t \{\dot{U}_t\} + \frac{\Delta t^2}{2} \{\ddot{U}_t\} + \frac{\Delta t^3}{6} \{\dddot{U}_t\} + \dots \\ \{\dot{U}_{t-\Delta t}\} &= \{\dot{U}_t\} - \Delta t \{\ddot{U}_t\} + \frac{\Delta t^2}{2} \{\dddot{U}_t\} + \dots \\ \{\ddot{U}_{t-2\Delta t}\} &= \{\ddot{U}_t\} - 2\Delta t \{\ddot{U}_t\} + 2\Delta t^2 \{\dddot{U}_t\} + \dots\end{aligned}$$

Second-, third-order formulae (etc.) are obtained by eliminating the terms $\{\ddot{U}_t\}$, $\{\dddot{U}_t\}$, ... and then replacing the expressions of $\{\dot{U}_t\}$, $\{\ddot{U}_{t-\Delta t}\}$, ... with $\{f(U_t)\}$, $\{f(U_{t-\Delta t})\}$, ...

Correction formulae

For systems similar in form to (5.89), these formulae are written as:

$$\{U_{t+\Delta t}\} = \{U_t\} + \Delta t \sum_{j=0}^n b_j \{f(U_{t+\Delta t-j\Delta t}, t + \Delta t - j \Delta t)\}. \quad (5.127)$$

In the case of formulation (5.86), the formula is identical to (5.125); however, the subscript j has an initial value of zero. The corresponding coefficients b_j are:

	n	Error	b_0	b_1	b_2	b_3
Third-order formula (Eulerian, with $\alpha = 0.5$)	1	$0(\Delta t^3)$	$\frac{1}{2}$	$\frac{1}{2}$	0	0
Fourth-order formula	2	$0(\Delta t^4)$	$\frac{5}{12}$	$\frac{8}{12}$	$-\frac{1}{12}$	0
Fifth-order formula	3	$0(\Delta t^5)$	$\frac{9}{24}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$

As $\{U_{t+\Delta t}\}$ appears in $\{f\}$, we have to iterate within the time-step. For the formulation (5.86), at each iteration we have to calculate:

$$[C]\{\Delta U^i\} = \{\bar{R}\}; \quad \{U_{t+\Delta t}^i\} = \{U_t\} + \{\Delta U^i\} \quad (5.128)$$

$$\{\bar{R}\} = \Delta t b_0 \{R_0\} + \Delta t \sum_{j=1}^n b_j \{R_j\} \quad (5.129)$$

$\{R_j\}$ are given by (5.126) and remain constant in each step.

$$\{R_0\} = \{F_{t+\Delta t}\} - [K(U_{t+\Delta t}^{i-1})] \{U_{t+\Delta t}^{i-1}\} \quad (5.130)$$

The combined use of (5.125) and (5.128) is highly effective if $[C]$ is diagonal; in this case, all the operations are performed on vectors. For linear problems, the correction algorithm converges if the eigenvalues of $[C]^{-1} \Delta t b_0 [K]$ are less than one in their absolute value. If $[C]$ is diagonal, this is verified by:

$$\Delta t \leq \frac{1}{b_0 \max_i \sum_j \left| \frac{K_{ij}}{C_{ii}} \right|} \equiv \Delta t_c \quad (5.131)$$

The same algorithm (5.128) applies to nonlinear problems. However, the stability criterion (5.131) may impose a very small value of Δt_c . It is possible to increase this value by modifying algorithm (5.128) as follows:

$$[C'] \{ \Delta U^i \} = \{ \bar{R}' \} \quad (5.132)$$

where: $[C'] = [C] + \Delta t b_0 [K_l]$

$$\{ \bar{R}' \} = \Delta t b_0 \{ R'_0 \} + \Delta t \sum_{j=1}^n b_j \{ R_j \}$$

$$\{ R'_0 \} = \{ F_{t+\Delta t} \} - [K_{nl}(U^{i-1}_{t+\Delta t})] \{ U^{i-1}_{t+\Delta t} \}$$

$[K_l]$ and $[K_{nl}]$ are the linear and nonlinear parts of $[K]$.

The convergence criterion is then:

$$\rho([C']^{-1} \Delta t b_0 [K_{nl}]) \leq 1. \quad (5.133)$$

Note that the matrix $[C]$ from (5.128) and the matrix $[C']$ in (5.132) are assembled and triangularized only once.

All prediction-and-correction methods, with the exception of the Eulerian methods, use the results of previous steps; hence, they pose problems for starting the process for the first time-steps and when Δt changes. Therefore, we have to resort to Eulerian or Runge–Kutta methods (see section 5.4.2.5).

EXAMPLE 5.31. Solution of a first-order, two-variable problem by prediction-and-correction

Let us solve the system defined in Example 5.27 using a fourth-order prediction formula, followed by a fourth-order correction formula. We begin the resolution by using the results obtained in Example 5.32 with the Runge–Kutta formula.

The prediction formula (5.125) is written explicitly:

$$\begin{aligned}\{\Delta U\} &= \Delta t[C]^{-1} \left(\{F\} - \frac{1}{12}[K](23\{U_t\} - 16\{U_{t-\Delta t}\} + 5\{U_{t-2\Delta t}\}) \right) \\ \{U_{t+\Delta t}^0\} &= \{U_t\} + \{\Delta U\}.\end{aligned}$$

The correction formula (5.127) is written thus (for a single iteration):

$$\begin{aligned}\{\Delta U^1\} &= \Delta t[C]^{-1} \left(\{F\} - \frac{1}{12}[K](5\{U_{t+\Delta t}^0\} + 8\{U_t\} - \{U_{t-\Delta t}\}) \right) \\ \{U_{t+\Delta t}^1\} &= \{U_t\} + \{\Delta U^1\}.\end{aligned}$$

For $\Delta t = 0.1$ and 0.4 , we present the solution in the following table:

t	$\Delta t = 0.1$		$\Delta t = 0.4$	
	u_1	u_2	u_1	u_2
0.0	0.0	0.0	0.0	0.0
0.1	0.996667 *	0.906667 *		
0.2	0.197608 *	0.164924 *		
0.3	0.293	0.226		
0.4	0.384	0.275	0.3786667 *	0.2826667 *
0.5	0.471	0.316		
0.6	0.553	0.349		
0.7	0.630	0.377		
0.8	0.702	0.399	0.6984533 *	0.4055324 *
1.2	0.943	0.455	0.941	0.459
1.6	1.116	0.480	1.116	0.482
2.0	1.238	0.491	1.239	0.492
4.0	1.464	0.500	1.466	0.500
6.0	1.495	0.500	1.495	0.500
8.0	1.499	0.500	1.499	0.500
10.0	1.500	0.500	1.500	0.500

(*: solution obtained using the Runge–Kutta method)

5.4.2.5 Explicit Runge–Kutta-type methods [CAR 69; CRA 56; KRE 88]

These methods solve system (5.89) using repeated evaluations of $\{f(\{U\}, t)\}$ within each time-step Δt . They are self starting and pose no problem for step value changes. However, when Δt remains constant, these methods are less efficient than prediction-and-correction methods, because of the evaluations of $\{f\}$ necessary at each step. The general expression of the Runge–Kutta formulae is:

$$\{U_{t+\Delta t}\} = \{U_t\} + \sum_{j=1}^n b_j \{g(U_j, t_j)\} \quad (5.134)$$

with:

$$\sum_{j=1}^n b_j = 1, \quad b_j > 0.$$

n	Error	Formulae
1	$O(\Delta t^3)$	<ul style="list-style-type: none"> $b_1 = 1; t_1 = t + \frac{\Delta t}{2}; \{U_1\} = \{U_t\} + \frac{\Delta t}{2} \{f(U_t, t)\}; \{g_1\} = \Delta t \{f(U_1, t_1)\}$
2	$O(\Delta t^4)$	<ul style="list-style-type: none"> $b_1 = \frac{1}{4}; t_1 = t; \{g_1\} = \Delta t \{f(U_t, t_1)\}$ $b_2 = \frac{3}{4}; t_2 = t + \frac{2\Delta t}{3}; \{U_2\} = \{U_t\} + \frac{2}{3} \Delta t \left\{ f\left(U_t + \frac{1}{3} g_1, t + \frac{\Delta t}{3}\right) \right\}; \{g_2\} = \Delta t \{f(U_2, t_2)\}$
4	$O(\Delta t^5)$	<ul style="list-style-type: none"> $b_1 = \frac{1}{6}; t_1 = t; \{g_1\} = \Delta t \{f(U_t, t_1)\}$ $b_2 = \frac{1}{3}; t_2 = t + \frac{\Delta t}{2}; \{U_2\} = \{U_t\} + \frac{1}{2} \{g_1\}; \{g_2\} = \Delta t \{f(U_2, t_2)\}$ $b_3 = \frac{1}{3}; t_3 = t + \frac{\Delta t}{2}; \{U_3\} = \{U_t\} + \frac{1}{2} \{g_2\}; \{g_3\} = \Delta t \{f(U_3, t_3)\}$ $b_4 = \frac{1}{6}; t_4 = t + \Delta t; \{U_4\} = \{U_t\} + \{g_3\}; \{g_4\} = \Delta t \{f(U_4, t_4)\}$

For a linear system of the form: $[C] \left\{ \frac{dU}{dt} \right\} = \{F\} - [K]\{U\}$,

the application of the Runge–Kutta method is written as:

$$[C] \{U_{t+\Delta t} - U_t\} = \Delta t \left(\sum_{j=1}^n b_j \{F(t_j)\} - [K] \left(\sum_{j=1}^n b_j \{U_j\} \right) \right).$$

EXAMPLE 5.32. Resolution of a first-order, two-variable problem using the Runge–Kutta formula

Let us solve the system defined in Example (5.27) using the fourth-order Runge–Kutta formula.

Formula (5.134) is written explicitly:

$$\{U_{t+\Delta t}\} = \{U_t\} + \frac{\Delta t}{4} [C]^{-1} (4 \{F\} - [K] (\{U_t\} + 3 \{U_2\}))$$

$$\text{where } \{U_2\} = \{U_t\} + \frac{2}{3} \Delta t [C]^{-1} (\{F\} - [K] \{U^*\})$$

$$\text{and } \{U^*\} = U_t + \frac{\Delta t}{3} [C]^{-1} (\{F\} - [K] \{U_t\}).$$

For $\Delta t = 0.1$ and 0.4 , we present the solution in the following table:

t	$\Delta t = 0.1$		$\Delta t = 0.4$	
	u_1	u_2	u_1	u_2
0.0	0.0	0.0	0.0	0.0
0.1	0.996667	0.906667		
0.2	0.197608	0.164924		
0.3	0.293	0.226		
0.4	0.384	0.275	0.3786667	0.2826667
0.5	0.471	0.316		
0.6	0.553	0.349		
0.7	0.630	0.377		
0.8	0.702	0.399	0.6984533	0.4055324
1.2	0.943	0.455	0.941	0.459
1.6	1.117	0.480	1.116	0.482
2.0	1.238	0.491	1.239	0.492
4.0	1.464	0.500	1.464	0.500
6.0	1.495	0.500	1.495	0.500
8.0	1.499	0.500	1.499	0.500
10.0	1.500	0.500	1.500	0.500

5.4.2.6 Lax–Wendroff scheme [LAX 60]

In this section, we present a time-discretization scheme that is particularly well adapted to convection-type problems. This scheme has the following peculiarities:

- It is an explicit Cranck–Nicholson-type formula (section 5.4.2.3).
- It uses a Galerkin formulation to evaluate $U_{t+\Delta t}$ and a collocation-type formulation to evaluation $U_{t+\frac{\Delta t}{2}}$.

For a scalar variable, these problems are thus written in the form:

$$\begin{aligned} \text{— 1 dimension: } & \frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0, \\ \text{— 2 dimensions: } & \frac{\partial u}{\partial t} + \frac{\partial f_x(u)}{\partial x} + \frac{\partial f_y(u)}{\partial y} = 0, \end{aligned} \quad (5.134a)$$

where u is a scalar variable representing a concentration, a temperature, a velocity, etc. The terms of flux, f_x and f_y , are functions of u , either linear or otherwise. For instance, the transport of a scalar quantity u by a two-dimensional flow is written as:

$$\frac{\partial u}{\partial t} + \frac{\partial(a_x u)}{\partial x} + \frac{\partial(a_y u)}{\partial y} = 0,$$

where (a_x, a_y) are the velocity components. The Burgers equation, for its part, describes a nonlinear transport problem.

$$\frac{\partial u}{\partial t} + \frac{\partial(u^2)}{\partial x} = 0.$$

The weak Galerkin form associated with equation (5.134a) after time discretization is written as:

$$W = \iint_A \delta u (u^{t+\Delta t} - u^t) dA - \Delta t \iint_A \left(\frac{\partial \delta u}{\partial x} f_x^\tau + \frac{\partial \delta u}{\partial y} f_y^\tau \right) dA + \oint_S \delta u f_n^\tau dS = 0, \quad (5.134b)$$

where: $f_n^\tau = f_x^\tau l + f_y^\tau m$ and $t < \tau < t + \Delta t$,

where l and m are the direction cosines of the normal to the boundary.

The choice of the time τ conditions the type of scheme:

$$\begin{aligned} \tau = t &: \text{explicit (unstable)} \\ \tau = t + \Delta t &: \text{implicit (stable but diffusive),} \\ \tau = t + \frac{\Delta t}{2} &: \begin{cases} \text{Lax-Wendroff (explicit)} \\ \text{Crank-Nicholson (semi-implicit)} \end{cases} \quad (\text{conditionally stable}). \end{aligned}$$

A finite-element spatial discretization associated with the Lax-Wendroff scheme gives us the form:

$$W = \sum_e W^e = \sum_e \langle \delta u_n \rangle \left([m] \{ \Delta u_n \} - \left\{ r_n \left(t + \frac{\Delta t}{2} \right) \right\} + \left\{ r_n^s \left(t + \frac{\Delta t}{2} \right) \right\} \right) = 0, \quad (5.134c)$$

where $\{ \Delta u_n \} = \{ u_n^{t+\Delta t} \} - \{ u_n^t \}$,

The elementary residual terms are given by:

$$\left\{ r_n \left(t + \frac{\Delta t}{2} \right) \right\} = \Delta t \sum_{I=1}^{NPG} \left(\left\{ N_{,x} \right\}_I \cdot f_x \left(u_n^{t+\frac{\Delta t}{2}} \right)_I + \left\{ N_{,y} \right\}_I \cdot f_y \left(u_n^{t+\frac{\Delta t}{2}} \right)_I \right) w_I \cdot J_I$$

where I : Gaussian integration point;

w_I : weight of integration;

J_I : Jacobian (if it is a reference element).

A collocation-type formulation is used to explicitly calculate $u_n^{t+\frac{\Delta t}{2}}$, so that:

$$W = \iint_A \delta u \left(u^{t+\frac{\Delta t}{2}} - u^t \right) dA + \frac{\Delta t}{2} \iint_A \delta u \left(\frac{\partial f_x^t}{\partial x} + \frac{\partial f_y^t}{\partial y} \right) dA = 0,$$

where δu constant in each element.

$$\text{Thus: } u_n^{t+\frac{\Delta t}{2}} = u_n^t - \frac{\Delta t}{2} \left(\frac{\partial}{\partial x} (f_x(u_n^t)) + \frac{\partial}{\partial y} (f_y(u_n^t)) \right),$$

and $f_n^{t+\frac{\Delta t}{2}} = f_x \left(u_n^{t+\frac{\Delta t}{2}} \right) \cdot l + f_y \left(u_n^{t+\frac{\Delta t}{2}} \right) \cdot m$ on the boundary.

For Dirichlet boundary conditions where $u = \bar{u}$ on part of S , we consider that $\delta u = 0$. The boundary terms associated with a flux condition are, for their part, integrated on contour elements:

$$\left\{ r_n^s \left(t + \frac{\Delta t}{2} \right) \right\} = \Delta t \sum_{I=1}^{NPG} \left(\left\{ N \right\}_I \cdot f_n \left(u_n^{t+\frac{\Delta t}{2}} \right)_I \right) \omega_I \cdot J_I,$$

where $f_n \left(u_n^{t+\frac{\Delta t}{2}} \right)_I = f_x \left(u_n^{t+\frac{\Delta t}{2}} \right)_I \cdot l + f_y \left(u_n^{t+\frac{\Delta t}{2}} \right)_I \cdot m$.

After assembly and taking account of the Dirichlet boundary conditions, we finally obtain:

$$[M] \{ \Delta U \} = \{ R \} \quad \text{and} \quad \{ U_{t+\Delta t} \} = \{ U_t \} + \{ \Delta U \}. \quad (5.134d)$$

Using a diagonal mass matrix $[M_D]$ enables us to solve this equation with a Jacobian iterative method [STR 86]:

$$\{ \Delta U^i \} = [M_D]^{-1} (\{ R \} + ([M_D] - [M]) \{ \Delta U^{i-1} \}).$$

The use of the Lax–Wendroff scheme is summarized by the program written in Matlab[®], given at the end of Example 5.33 and applied to the transport of a Gaussian.

Remark

This scheme can easily be extended to transport/diffusion problems, for which the weak formulation is with $\tau = t + \Delta t/2$:

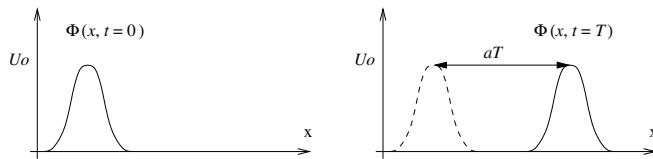
$$W = \iint_A \delta u (u^{t+\Delta t} - u^t) dA - \iint_A \left(\frac{\partial \delta u}{\partial x} f_x^\tau + \frac{\partial \delta u}{\partial y} f_y^\tau \right) dA + \\ \iint_A \left(\frac{\partial \delta u}{\partial x} k_x \frac{\partial u^\tau}{\partial x} + \frac{\partial \delta u}{\partial y} k_y \frac{\partial u^\tau}{\partial y} \right) dA + \oint_S \delta u (f_n^\tau - q_n^\tau) dS = 0, \\ \text{where } u_n^{t+\frac{\Delta t}{2}} = u_n^t + \frac{\Delta t}{2} \left(\frac{\partial}{\partial x} f_x(u_n^t) + \frac{\partial}{\partial y} f_y(u_n^t) \right).$$

EXAMPLE 5.33. Lax–Wendroff scheme for a monodimensional transport problem

Consider the one-dimensional scalar problem regulated by the relation:

$$\frac{\partial u(x, t)}{\partial t} + \frac{\partial(au)}{\partial x} = 0, \quad u(0, t) = U_0, \quad u(x, 0) = \Phi(x); \quad \Phi(0) = U_0.$$

The exact solution is given by: $u_{ex}(x, t) = \Phi(x - at)$, $0 \leq x \leq L$.



The variational form is written as:

$$W = \sum_e W^e + W_s^e = 0, \\ W^e = \int_0^{L^e} \delta u (u^{t+\Delta t} - u^t) dx - \Delta t \int_0^{L^e} \frac{\partial \delta u}{\partial x} (au)^{t+\frac{\Delta t}{2}} dx = 0, \quad W_s^e = (\delta u, au)_L.$$

Discretization with a 2-node element gives us:

$$W^e = \langle \delta u_n \rangle \left([m] \{ \Delta u_n \} - \left\{ r_n \left(t + \frac{\Delta t}{2} \right) \right\} \right) \text{ where } [m] = \frac{L^e}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix};$$

$$\text{and } \left\{ r_n \left(t + \frac{\Delta t}{2} \right) \right\} = a.u_n^{t+\frac{\Delta t}{2}} \begin{Bmatrix} -1 \\ 1 \end{Bmatrix} \text{ where } u_n^{t+\frac{\Delta t}{2}} = \frac{(u_1 + u_2)_t}{2} + \frac{\Delta t}{2} a \frac{(u_2 - u_1)_t}{L^e}.$$

The boundary term is calculated by:

$$r_s^e = a.u_{\frac{t+\Delta t}{2}}^e, \text{ with } u_{\frac{t+\Delta t}{2}}^e \text{ calculated on the last element.}$$

The three figures illustrate the transport of a Gaussian, initially centered at $x = 0.25$. There are three time-steps, respectively equal to $0.8 \Delta t_{crit}$, Δt_{crit} and $1.2 \Delta t_{crit}$ of the critical time-step. These time-steps are respectively associated with CFL numbers of 0.8, 1 and 1.2. The CFL (Courant–Friedrichs–Lowy) number is defined by: $CFL = \frac{\Delta t}{\Delta t_{crit}}$. The positivity criterion $CFL = 1$ is valid here only with the diagonalized mass matrix. The initial solution is shown by a dotted line; the solution yielded by the calculation is shown by a solid line. The first case presents a stable but non-positive solution with a dispersive effect (upstream oscillation). For the second case, the calculated solution is identical to the exact solution. The third is an unstable numerical solution.

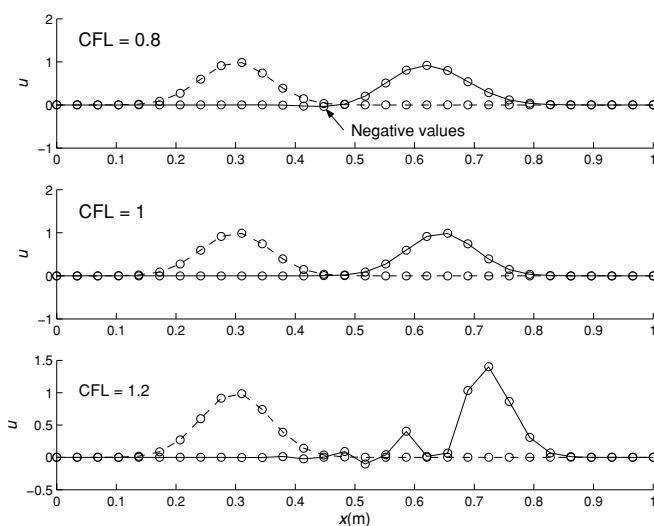


Figure 5.22 shows the Matlab[©] program.

```

clear all
% Geometric properties
nnt=30; nelt=nnt-1; % number of nodes, elements
nnel=2; L=1.0; % nodes per element, length
vcor([1:nnt],1)=[0:L./nelt:L]'; % coordinates
kconec([1:1:nelt],1)=[1:1:(nnt-1)]'; % connectivities
kconec([1:1:nelt],2)=[2:1:nnt]';
npas=10; % number of time-steps
% Initialization: Gaussian curve centered at x=centre
centre=0.3;
vsol=zeros(nnt,1); vflux=zeros(nnt,1); vmg=zeros(nnt);
vsol=exp(-(vcor-centre).^2.*150);
cfl=0.9; % CFL criterion
a=1.0; % velocity of flow
% Calculation of the time-step using the CFL criterion
dt=cfl.*min((L./nelt)./(a.*ones(nnt,1)));
%
hold on; plot(vcor,vsol,'-ob')
%
% Loop on the number of time-steps
for ipas=1:npas
    % Calculation of the flux vector on all the nodes
    vflux(:,1)= a.*vsol(:,1);
    %
    % Stage 1 (local): calculation of u_bar(n+1:2) (constant for each element)
    for ie=1:nelt
        vcore=vcor(kconec(ie,1:nnel),:); % extraction of elementary coordinates
        vsole=vsol(konec(ie,1:nnel),1); % extraction of elementary solution
        vflue=vflux(kconec(ie,1:nnel),:); % extraction of elementary flow
        x = vcore(2,:)-vcore(1,:); xl=sqrt(x * x'); % elementary length.
        vsol1_2(ie)=(vsole(1)+vsole(2))./2.-dt./xl./2.* (vflue(2)-vflue(1));
        vflu1_2(ie)= a.*vsol1_2(ie) ;
    end;
    %
    % Stage 2 (global): construction of the linear system
    vres=zeros(nnt,1);
    %

```

```

for ie=1:nelt
    kloce=[];
    for in=1:nnel
        kloce=[kloce,(kcone(ie, in)-1)+1];
    end
    vcore=vcor(kcone(ie,1:nnel),:);
    vsole=vsol(kcone(ie,1:nnel),1);
    vflux=vflux(kcone(ie,1:nnel),1);
    x = vcore(2 ,:) - vcore(1,:);    xl=sqrt(x * x');
    %
    vrese=dt.*[-vflux1_2(ie); vflux1_2(ie)];
    vres(kloce)=vres(kloce)+vrese;
    %
    if ipas==1           % a single assembly of [M]
        vme=xl./6.*[ 2 1; 1 2]; % consistent mass matrix
        %vme=xl./2.*[ 1 0; 0 1]; % diagonal mass matrix
        vmg(kloce,kloce)=vmg(kloce,kloce)+vme;
    end
    end
    %
    % Introduction of the boundary conditions
    vmg(1,:)=zeros(nnt,1); vres(1)=0.; vmg(:,1)=zeros(nnt,1); vmg(1,1)=1;
    vres(nnt)=vres(nnt)-dt.*vflux(nnt);
    %
    % Resolution and updating of the solution
    vdu=vmg\vrese; vsol=vsol+vdu;
end
%
plot(vcor,vsol,'-og'); xlabel('x(m)'); ylabel('u')

```

Figure 5.22. MATLAB[®] Program to solve the linear Bürgers problem using the Lax–Wendroff scheme

5.4.2.7 Studying stability by Neumann spectral decomposition

The Neumann spectral decomposition method is used to evaluate the stability of a scheme. In order to illustrate this method, let us consider the following monodimensional problem:

$$\frac{\partial u}{\partial t} - k \frac{\partial^2 u}{\partial x^2} = 0. \quad (5.134e)$$

Considering a uniform mesh and discretization by an explicit Eulerian scheme, the discrete relation at node j with the diagonalized mass matrix is written as:

$$u_j^{n+1} = u_j^n + \frac{\Delta t \cdot k}{\Delta x^2} (u_{j-1}^n - 2u_j^n + u_{j+1}^n),$$

or indeed:

$$u_j^{n+1} = r \cdot u_{j-1}^n + (1 - 2r) u_j^n + r \cdot u_{j+1}^n \quad \text{with} \quad r = \frac{\Delta t \cdot k}{\Delta x^2}, \quad (5.134f)$$

where u_j^n : value of u at the node x_j at time t ;

u_j^{n+1} : value of u at the node x_j at time $t + \Delta t$

Applied to all the nodes in the mesh, these relations are written thus in their matrix form: $\{U^{n+1}\} = [A]\{U^n\}$.

The scheme is numerically stable if the spectral radius (the ensemble of all the eigenvalues) of $[A]$ is limited by one (when the Dirichlet boundary conditions have been taken into account):

$$\rho(A) < 1.$$

In practice, this condition is difficult to verify as the calculation of the eigenvalues complicates the approach. The Neumann decomposition method enables us to bypass the calculation of the eigenvalues. The solution at time t is then rewritten in the form of a Fourier series:

$$u(x, t) = v(x) \times u_m(t) = \sum_m e^{im\pi \frac{x}{L}} \times u_m(t), \quad (5.134g)$$

where m is a harmonic mode and $i = \sqrt{-1}$.

Remarks

- For a mesh composed of n_e elements, the position of the node j is calculated by:

$$x_j = j \cdot \Delta x \quad \text{with} \quad \Delta x = \frac{L}{n_e} \quad \text{and} \quad j = 0, 1, \dots, n_e.$$

The summation term present in (5.134g) is then limited by $1 \leq m \leq n_e$:

$$u(x_j, t) = \sum_{m=1}^{n_e} e^{ijp} \times u_m(t), \quad \text{where} \quad p = \frac{m\pi}{n_e}.$$

- An acceptable representation of the different modes requires at least four finite elements per half-wavelength, so that:

$$m \cdot \pi \frac{4}{n_e} \approx \pi, \text{ or } m \leq \frac{n_e}{4} = \frac{L}{4 \Delta x}.$$

Thus, at each node with coordinates $x_j = j \cdot \Delta x$ and for each of the modes m , we have the decomposition:

$$u(x_j, t) = u_m(t) \cdot e^{imj\pi \frac{\Delta x}{L}} = u_m(t) \cdot e^{ijp}, \quad j = 0, 1, \dots, n_e. \quad (5.134h)$$

The Neumann decomposition then consists of separately examining the stability of each mode m . Let us use (5.134h) in equation (5.134f) with:

$$\begin{aligned} u_j^n &= u_m^n \cdot e^{ijp}, \\ u_j^{n+1} &= u(x_j, t + \Delta t) = u_m^{n+1} \cdot e^{ijp}, \\ u_{j-1}^n &= u_m^n \cdot e^{im(j-1)\frac{\pi}{n_e}} = u_j^n \cdot e^{-ip}, \\ u_{j+1}^n &= u_m^n \cdot e^{im(j+1)\frac{\pi}{n_e}} = u_j^n \cdot e^{ip}, \end{aligned}$$

Hence:

$$u_m^{n+1} = \left(r(e^{-ip} + e^{ip}) + (1 - 2r) \right) u_m^n, \text{ where } r = k \frac{\Delta t}{\Delta x^2} \text{ and } n_e = \frac{L}{\Delta x}, \quad (5.134i)$$

which can be written in the form:

$$u_m^{n+1} = \left(1 - 2r(1 - \cos(p)) \right) u_m^n. \quad (5.134j)$$

The stability of the scheme is then assured for: $0 \leq 1 - 2r(1 - \cos(p)) \leq 1$.

Given that the cosine function is bounded by ± 1 , we can deduce from this that:

$$k \frac{\Delta t}{\Delta x^2} \leq \frac{1}{4}. \quad (5.134k)$$

Remarks

- The presence of a source term has no effect whatsoever on the stability of the scheme: indeed, it can be integrated into $u(x, t)$ simply by changing the variable.
- This analysis does not deal with the stability in the vicinity of the initial condition on $u(x, t)$.

By way of comparison, the positivity of the scheme (section 4.1.2) is assured for:

$$r \geq 0 \text{ and } (1 - 2r) \geq 0 \text{ so that } k \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2}. \quad (5.134l)$$

We note that condition (5.134k) is more restrictive than condition (5.134l). If we limit the Neumann analysis to $m \frac{\pi}{n_e} = \frac{\pi}{2}$, the two criteria are then equivalent. Thus, the criterion of positivity of the scheme will be preferred because it is simpler to use.

Of course, a scheme with a consistent mass matrix precludes the use of the positivity criterion: we then use the Neumann decomposition technique. However, it is possible in certain types of problems, parabolic (thermal, diffusion, etc.), to diagonalize the mass matrix.

Remark about the precision of the scheme

By taking the solution of the form $u(x, t) = u_m(t) e^{i p \frac{x}{\Delta x}}$, where $p = \frac{m\pi}{n_e}$, the exact solution of equation (5.134e) is written as:

$$u^{t+\Delta t} = u_m(t) \cdot e^{-r p^2}, \quad r = \frac{k \Delta t}{\Delta x^2},$$

so: $u_{ex,m}^{n+1} = u_m^n \cdot e^{-r p^2}$ (exact).

The solution of the discrete system (5.134f) is given by the relation (5.134j). The error for the solution is then written:

$$e = u_{ex,m}^{n+1} - u_m^{n+1} = [e^{-r p^2} - (1 - 2r(1 - \cos(p)))] u_m^n,$$

This can be simplified with a development into a Taylor series:

$$e = \left(\frac{r^2}{2} - \frac{r}{12} \right) p^4 + \dots$$

Thus, it is possible to reduce the error by decreasing p (by increasing the number of elements, $p = \frac{m\pi}{n_e}$), while keeping r constant to ensure the positivity of the schema — and consequently reducing the time-step Δt such that:

$$\frac{\Delta t}{\Delta x^2} = \text{constant.}$$

EXAMPLE 5.34. Spatial shifting and positivity

Consider the monodimensional transport problem governed by the equation:

$$\frac{\partial u}{\partial t} + a \frac{\partial u}{\partial x} = 0.$$

Discretization by:

- an explicit Eulerian scheme of the time term,
- a space discretization whose second term is respectively left offset (LO), centered (C) and right offset (RO),

gives us the general discrete form: $u_j^{n+1} = \alpha \cdot u_{j-1}^n + \beta u_j^n + \gamma \cdot u_{j+1}^n$

where:

		Coefficients		
		α	β	γ
Type of discretization	C	$\frac{a\Delta t}{2\Delta x}$	1	$-\frac{a\Delta t}{2\Delta x}$
	RO	0	$\left(1 + \frac{a\Delta t}{\Delta x}\right)$	$-\frac{a\Delta t}{\Delta x}$
	LO	$\frac{a\Delta t}{\Delta x}$	$\left(1 - \frac{a\Delta t}{\Delta x}\right)$	0

The positivity of the scheme is assured if the three coefficients α , β and γ are positive or null. The centered discretization (C) of the spatial derivative term cannot be used to ensure the positivity of the scheme.

This choice is also unstable in the sense of the Neumann decomposition. Choosing to offset the discretization, however, enables us to obtain positive schemes depending on the sign of “ a ”, a constant similar to a stream velocity either positive (left to right) or negative (right to left). Thus, the positivity of the scheme is ensured by:

- a left-offset scheme for $a > 0$,
- a right-offset scheme for $a < 0$,

with condition that: $\frac{|a|\Delta t}{\Delta x} < 1$.

This technique is at the basis of the so-called flux splitting techniques.

EXAMPLE 5.35. Diffusive and dispersive schemes

Consider a monodimensional pure convection problem:

$$\frac{\partial u}{\partial t} + a \frac{\partial u}{\partial x} = 0.$$

For the initial condition: $u(x, 0) = u_0 e^{im\pi \frac{x}{L}}$, $m = 1, 2, \dots$

the exact solution is given by: $u(x, t) = u_0 e^{im\pi \frac{x}{L}(x-at)}$, $m = 1, 2, \dots$

The exact solution at time $t + \Delta t$ is deduced from the solution at time t , by introducing phase shifting such that:

$$u(x, t + \Delta t) = u_0 e^{im\pi \frac{x}{L}(x-at)} \cdot e^{-i\Phi} \text{ with the phase } \Phi = \frac{m\pi a}{L} \Delta t.$$

Depending on the size Δx and number of $n_e = \frac{L}{\Delta x}$ of the elements, we have:

$$\Phi = \frac{m\pi}{n_e} C, \text{ with the Courant number: } C = \frac{a\Delta t}{\Delta x}.$$

The exact solution:

$$u(x, t + \Delta t) = u(x, t) e^{-i\Phi},$$

is at constant amplitude over time, but presents a phase shift Φ between the times t and $t + \Delta t$.

— A numerical scheme is diffusive or dissipative if it leads to a decrease in the amplitude of the solution between the times t and $t + \Delta t$.

— A scheme is dispersive if the phase error $\varepsilon_\Phi = \frac{(\Phi - \Phi_{num})}{\Phi}$ is a function of the mode "m".

Explicit scheme:

For an explicit scheme with diagonalization of the mass term, the equilibrium relation discretized at the node "j" is written as:

$$u_j^{n+1} = \frac{C}{2} u_{j-1}^n + u_j^n - \frac{C}{2} u_{j+1}^n. \quad (A-1)$$

This scheme does not satisfy the positivity criterion. For a mode:

$$u(x_j, t) = u_m^n e^{im\pi \frac{x_j}{L}} = u_m^n e^{i j p}; \quad x_j = j \Delta x; \quad \Delta x = \frac{L}{n_e}; \quad i = \sqrt{-1}; \quad p = \frac{m\pi}{n_e},$$

equation (A-1) is written: $u_m^{n+1} = (1 - i C \sin(p)) u_m^n$.

Here, the amplification coefficient is equal to: $A = |1 - i.C. \sin(p)| > 1$.

The scheme is unstable.

Implicit scheme:

The implicit version gives us the discrete relation:

$$-\frac{C}{2} u_{j-1}^{n+1} + u_j^{n+1} \frac{C}{2} u_{j+1}^{n+1} = u_j^n. \quad (A-2)$$

This scheme is not positive, because for $u_j^n = 0$, $u_{j-1}^{n+1} = 0$ and $u_{j+1}^{n+1} > 0$, we have $u_j^{n+1} < 0$.

For a mode “m”, equation (A-2) is written as:

$$u_m^{n+1} = \frac{1}{1 + i.C. \sin(p)} u_m^n = A.u_m^n \cdot e^{-i\Phi_{num}}.$$

As the amplification coefficient “A” is strictly less than one:

$$A = \frac{1}{\sqrt{1 + C^2 \cdot \sin^2(p)}} < 1.$$

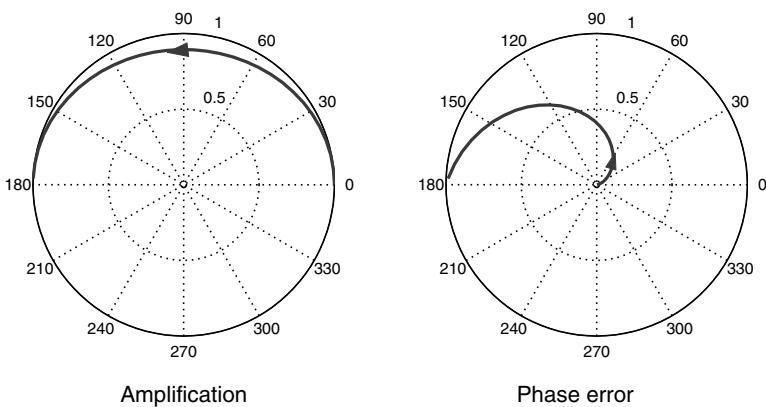
The scheme is stable but diffusive and dispersive.

- It is diffusive because: $A < 1$.
- It is dispersive because the phase shifting error given by:

$$\varepsilon_\Phi = \frac{(\Phi - \Phi_{num})}{\Phi} = \left(1 - \frac{\tan^{-1}(C. \sin(p))}{C. p} \right),$$

is a function of the mode “m”.

The figures illustrate the amplitude of the solution and the phase error for $0 \leq p \leq \pi$, where $C = 1$:



Lax-Wendroff scheme:

In the linear case, the Lax-Wendroff scheme is written as:

$$u_j^{n+1} = \underbrace{\frac{C}{2} u_{j-1}^n + u_j^n - \frac{C}{2} u_{j+1}^n}_{\text{explicit scheme}} + \frac{C^2}{2} (u_{j-1}^n - 2u_j^n + u_{j+1}^n),$$

$$\text{Hence: } u_j^{n+1} = \frac{C}{2} (1+C) u_{j-1}^n + (1-C^2) u_j^n - \frac{C}{2} (1-C) u_{j+1}^n. \quad (A-3)$$

The scheme is not positive. For a mode "m", equation (A-3) is written as:

$$u_j^{n+1} = (1 - C^2 + C^2 \cos(p) - iC \sin(p)) u_j^n \quad \text{where } p = \frac{m\pi}{n_e}, \quad C = \frac{a\Delta t}{\Delta x},$$

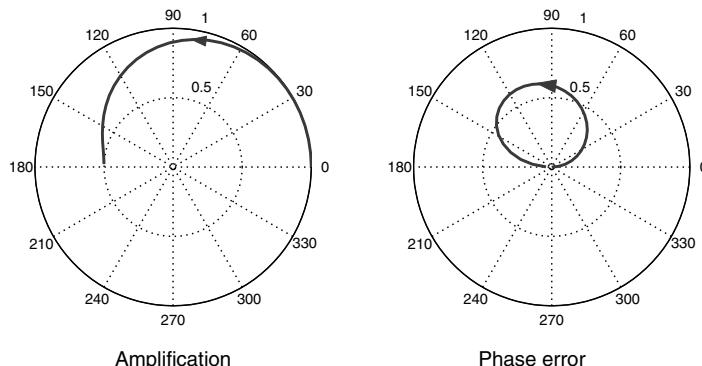
$$\text{Hence: } u_j^{n+1} = \left(1 - 2C^2 \sin^2\left(\frac{p}{2}\right) - iC \sin(p) \right) u_j^n = A \cdot u_j^n \cdot e^{-i\Phi_{num}},$$

$$\text{where: } A = \sqrt{1 - 4(1 - C^2)C^2 \sin^4\left(\frac{p}{2}\right)}, \quad \Phi_{num} = \tan^{-1} \left(\frac{C \sin(p)}{1 - 2C^2 \sin^2\left(\frac{p}{2}\right)} \right).$$

The amplification coefficient is less than one for $C < 1$.

$$\text{The phase error is given by: } \varepsilon_\Phi = \left(1 - \frac{\tan^{-1}(C \cdot \sin(p))}{1 - 2C^2 \sin^2\left(\frac{p}{2}\right)} \right).$$

Thus, the scheme is stable as long as it respects the criterion $C < 1$, which is known as the CFL (Courant-Friedrichs-Levy) criterion, but it also exhibits a diffusive and dispersive nature.



However, for low harmonics ($p < 1$), the diffusion and dispersion remain slight.

For the particular case where $C = 1$, we have $A = 1$, $\varepsilon_\Phi = 0$, the scheme is then “exact” for any wave number m and:

$$u_j^{n+1} = u_{j-1}^n.$$

5.4.3 MODAL SUPERPOSITION METHOD FOR FIRST-ORDER SYSTEMS

This method transforms the coupled system (5.86), here assumed to be linear, into an uncoupled system by the transformation:

$$\{U(t)\} = [X] \{V(t)\} \quad (5.135)$$

where the transformation matrix $[X]$ is constituted by the n eigenvectors $\{X_i\}$ defined by (see section 5.5):

$$([K] - \lambda_i [C]) \{X_i\} = 0 \quad i = 1, 2, \dots, n \quad (5.136)$$

$$[X] = [\{X_1\} \{X_2\} \dots \{X_n\}]. \quad (5.137)$$

The matrix $[X]$ satisfies the orthogonality relations obtained in (5.176):

$$[X]^T [C] [X] = [I]$$

$$[X]^T [K] [X] = [\lambda] = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{bmatrix} \quad (5.138)$$

System (5.86), transformed by (5.135), is written in the form of uncoupled equations:

$$[X]^T [C] [X] \{\dot{V}(t)\} + [X]^T [K] [X] \{V(t)\} = [X]^T \{F(t)\} = \{\bar{F}(t)\}$$

thus, in view of (5.138):

$$\{\dot{V}(t)\} + [\lambda] \{V(t)\} = \{\bar{F}(t)\} \text{ for } t > t_0. \quad (5.139)$$

Each uncoupled equation in this system can be integrated explicitly. The initial conditions $\{V(t_0)\}$ are obtained by premultiplying (5.135) by $[X]^T [C]$:

$$[X]^T [C] \{U(t_0)\} = [X]^T [C] [X] \{V(t_0)\} = \{V(t_0)\}. \quad (5.140)$$

The general solution to the i -th equation in (5.139) is written as:

$$V_i(t) = e^{-\lambda_i(t-t_0)} \left(V_i(t_0) + \int_{t_0}^t e^{\lambda_i(s-t_0)} \bar{F}_i(s) ds \right). \quad (5.141)$$

When the expression of $\{\bar{F}(t)\}$ permits it, (5.141) is integrated explicitly to give $V_i(t)$, $i = 1, 2, \dots, n$. The solution sought $\{U(t)\}$ is a linear combination of the eigenvectors $\{X_i\}$, whose coefficients are $V_i(t)$, according to (5.135).

In practice, it is not necessary to use all the eigenvectors and eigenvalues; we content ourselves with a rectangular matrix $[X]$ containing only the dominant modes, for which $\{X_i\} V_i(t)$ are greatest.

When $\{\bar{F}(t)\}$ is complicated, it is more efficient to integrate each uncoupled equation (5.139) using the direct step-by-step method of section 5.4.2, rather than numerically integrating (5.141).

Let us recap the solution procedure by modal superposition:

- choose the number p of modes that will be used;
- calculate the p smallest eigenvalues of the system (5.136) and the corresponding eigenvectors (see section 5.5);
- construct the vector $\{\bar{F}\}$, then solve p uncoupled equations from (5.139);
- obtain the solution by modal superposition using (5.135).

EXAMPLE 5.36. Resolution of a first-order system by modal superposition

Consider the system defined in Example 5.27:

$$[C]\{\dot{U}\} + [K]\{U\} = \{F\} \quad (\text{for } t > 0);$$

$$[C] = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}; \quad [K] = \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}; \quad \{F\} = \begin{Bmatrix} 2 \\ 3 \end{Bmatrix}$$

$$\{U(t=0)\} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}.$$

The eigenvalues and eigenvectors of

$$([K] - \lambda[C])\{X\} = 0$$

are (see section 5.5 and Example 5.40):

$$\lambda_1 = 1 \quad \{X_1\} = \begin{Bmatrix} 1 \\ 0 \end{Bmatrix}$$

$$\lambda_2 = 2 \quad \{X_2\} = \begin{Bmatrix} 1 \\ -1 \end{Bmatrix}$$

The uncoupled system (5.139) is written as:

$$\{\dot{V}\} + \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \{V\} = \begin{bmatrix} 1 & 0 \\ 1 & -1 \end{bmatrix} \begin{Bmatrix} 2 \\ 3 \end{Bmatrix},$$

so

$$\begin{aligned} \dot{V}_1 + V_1 &= 2 & V_1(t=0) &= 0 \\ \dot{V}_2 + 2V_2 &= -1 & V_2(t=0) &= 0. \end{aligned}$$

The solutions to these two equations are (5.141):

$$\begin{aligned} V_1(t) &= 2(1 - e^{-t}) \\ V_2(t) &= \frac{1}{2}(e^{-2t} - 1). \end{aligned}$$

The solution of the coupled system, therefore, is (5.135):

$$\{U\} = \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix} \begin{Bmatrix} 2(1 - e^{-t}) \\ \frac{1}{2}(e^{-2t} - 1) \end{Bmatrix} = \begin{Bmatrix} \frac{3}{2} - 2e^{-t} + \frac{1}{2}e^{-2t} \\ \frac{1}{2} - \frac{1}{2}e^{-2t} \end{Bmatrix}.$$

t	u_1	u_2
0	0	0
0.1	0.100	0.091
0.2	0.198	0.165
0.3	0.293	0.226
0.4	0.384	0.275
0.5	0.471	0.316
0.6	0.553	0.349
0.7	0.630	0.377
0.8	0.702	0.399
0.9	0.770	0.417
1.0	0.832	0.432
2.0	1.239	0.491
3.0	1.402	0.499
4.0	1.464	0.5
5.0	1.487	0.5
6.0	1.495	0.5
10.0	1.5	0.5
∞	1.5	0.5

5.4.4 METHODS FOR DIRECT INTEGRATION OF SECOND-ORDER SYSTEMS [BAT 76; WIL 77; CLO 75]

5.4.4.1 Central finite difference method

Recall the second-order system (5.87):

$$[M]\{\ddot{U}\} + [C]\{\dot{U}\} + [K]\{U\} = \{F(t)\} \quad t > t_0 \quad (5.142)$$

$\{U\}$ and $\{\dot{U}\}$ being given for $t = t_0$.

The central finite difference method is an **explicit** method that uses the solution at time t , and the following approximations of the derivatives by centered finite differences:

$$\{\ddot{U}\} \approx \frac{1}{\Delta t^2} (\{U_{t+\Delta t}\} - 2\{U_t\} + \{U_{t-\Delta t}\}) \quad (5.143a)$$

$$\{\dot{U}\} \approx \frac{1}{2\Delta t} (\{U_{t+\Delta t}\} - \{U_{t-\Delta t}\}). \quad (5.143b)$$

System (5.142) is then written as:

$$[\bar{K}] \{U_{t+\Delta t}\} = \{R_t\}$$

where:

$$[\bar{K}] = [M] + \frac{\Delta t}{2} [C] \quad (5.144a)$$

$$\begin{aligned} \{R_t\} = \Delta t^2 \{F_t\} + [M](2\{U_t\} - \{U_{t-\Delta t}\}) \\ + \frac{\Delta t}{2} [C](\{U_{t-\Delta t}\} - \Delta t^2 [K]\{U_t\}) \end{aligned}$$

or in its incremental form: $[\bar{K}] \{\Delta U\} = \{R_t\}$,

where:

$$\begin{aligned} \{R_t\} = \Delta t^2 \{F_t\} + [M](\{U_t\} - \{U_{t-\Delta t}\}) + \\ + \frac{\Delta t}{2} [C](-\{U_t\} + \{U_{t-\Delta t}\}) - \Delta t^2 [K]\{U_t\} \end{aligned}$$

and

$$\{U_{t+\Delta t}\} = \{U_t\} + \{\Delta U\}. \quad (5.144b)$$

At time $t = t_0$, $\{U_0\}$ and $\{\dot{U}_0\}$ are known, and $\{\ddot{U}_0\}$ can be evaluated using (5.142). In addition, $\{U_{t_0-\Delta t}\}$ is evaluated by eliminating $\{U_{t+\Delta t}\}$ between the two equations (5.143) written at time $t = t_0$:

$$\{U_{t_0-\Delta t}\} = \{U_0\} - \Delta t \{\dot{U}_0\} + \frac{\Delta t^2}{2} \{\ddot{U}_0\}. \quad (5.145)$$

Figure 5.23 represents the operations necessary to implement the above algorithm. The stability of the solution is ensured for Δt less than a critical value Δt_c that is linked to the minimum characteristic period of resonance T_{\min} of the physical system being studied. In addition, Δt must be small enough for the approximation errors of the discretizations (5.143a) and (5.143b) to be acceptable.

Given that the method is explicit, $[K]$ is not involved in the left-hand side of (5.144a) and (5.144b); for nonlinear problems in which only $[K]$ is dependent on $\{U\}$, these algorithms are still directly applicable using $K(\{U_t\})$. The nonlinearity may necessitate the use of a smaller Δt than in the linear case.

5.4.4.2 Houbolt method [HOU 50]

This implicit method uses the expression of the system (5.142) at time $t + \Delta t$, and the following right-offset finite difference approximations:

$$\{\ddot{U}_{t+\Delta t}\} = \frac{1}{\Delta t^2} (2\{U_{t+\Delta t}\} - 5\{U_t\} + 4\{U_{t-\Delta t}\} - \{U_{t-2\Delta t}\}) \quad (5.146a)$$

$$\{\dot{U}_{t+\Delta t}\} = \frac{1}{6 \Delta t} (11\{U_{t+\Delta t}\} - 18\{U_t\} + 9\{U_{t-\Delta t}\} - 2\{U_{t-2\Delta t}\}). \quad (5.146b)$$

The truncation error in these formulae is in $(\Delta t)^2$. The algorithm thus obtained is written as:

$$[\bar{K}]\{U_{t+\Delta t}\} = \{R_{t+\Delta t}\}$$

$$\text{where: } [\bar{K}] = 2[M] + \frac{11}{6} \Delta t[C] + \Delta t^2[K] \quad (5.147)$$

$$\begin{aligned} \{R_{t+\Delta t}\} &= \Delta t^2 \{F_{t+\Delta t}\} + [M] (5\{U_t\} - 4\{U_{t-\Delta t}\} + \{U_{t-2\Delta t}\}) + \\ &\quad + \Delta t[C] \left(3\{U_t\} - \frac{3}{2}\{U_{t-\Delta t}\} + \frac{1}{3}\{U_{t-2\Delta t}\} \right). \end{aligned} \quad (5.148)$$

By introducing $\{\Delta U\}$:

$$\begin{aligned} [\bar{K}]\{\Delta U\} &= \{R_{t+\Delta t}\} - [\bar{K}]\{U_t\} \\ \{U_{t+\Delta t}\} &= \{U_t\} + \{\Delta U\}. \end{aligned}$$

This algorithm is unconditionally stable but it is not self-starting. The corresponding operations are described in Figure 5.23. To calculate the solution to $t = t_0 + \Delta t$ and $t = t_0 + 2 \Delta t$, we must resort to other methods of integration, e.g. the central finite difference method discussed in the previous section.

For nonlinear problems in which $[M]$ and $[C]$ are constant and $[K]$ is dependent on $\{U\}$, the correction of $\{U_{t+\Delta t}\}$ applied by each iteration of a step is defined by:

$$\begin{aligned} [K_{nl}] \{\Delta U^i\} &= \{R_{t+\Delta t}\} - [\bar{K}(U_{t+\Delta t}^{i-1})] \{U_{t+\Delta t}^{i-1}\} \\ \{U_{t+\Delta t}^i\} &= \{U_{t+\Delta t}^{i-1}\} + \{\Delta U^i\}. \end{aligned} \quad (5.149)$$

In the substitution method:

$$[K_{nl}] = [\bar{K}]. \quad (5.150)$$

In the Newton-Raphson method:

$$[K_{nl}] = [K_t] = [\bar{K}] + \Delta t^2 \left[\frac{\partial K}{\partial U} \cdot U \right]_{t+\Delta t}^{i-1}. \quad (5.151)$$

5.4.4.3 Newmark and Wilson methods [BAT 76; KAT 77; NEW 59]

These implicit self-starting methods enable us to construct the solution at time $t + \Delta t$ from the known vectors $\{U_t\}$, $\{\dot{U}_t\}$ and $\{\ddot{U}_t\}$. They use the following truncated developments:

$$\{\dot{U}_{t+\tau}\} = \{\dot{U}_t\} + \tau((1-a)\{\ddot{U}_t\} + a\{\ddot{U}_{t+\tau}\}) \quad (5.152a)$$

$$\{U_{t+\tau}\} = \{U_t\} + \tau\{\dot{U}_t\} + \frac{\tau^2}{2}((1-b)\{\ddot{U}_t\} + b\{\ddot{U}_{t+\tau}\}). \quad (5.152b)$$

When $a = b = \frac{1}{2}$, these approximations consist of assuming the acceleration to be constant in the interval $t, t + \tau$ and equal to its average value. When $a = \frac{1}{2}$ and $b = \frac{1}{3}$, these approximations consist of assuming the acceleration to vary in a linear fashion over the interval $t, t + \tau$.

Expression (5.142), written at time $t + \tau$, becomes:

$$[\bar{K}] \{U_{t+\tau}\} = \{R_{t+\tau}\} \quad (5.153)$$

where:

$$[\bar{K}] = [M] + \tau a[C] + \frac{\tau^2}{2} b[K]$$

$$\begin{aligned} \{R_{t+\tau}\} &= \frac{\tau^2}{2} b \{F_{t+\tau}\} + [M] \left(\{U_t\} + \tau \{\dot{U}_t\} + \frac{\tau^2}{2} (1-b) \{\ddot{U}_t\} \right) + \\ &+ [C] \left(\tau a \{U_t\} + \frac{\tau^2}{2} (2a-b) \{\dot{U}_t\} + \frac{\tau^3}{2} (a-b) \{\ddot{U}_t\} \right). \end{aligned}$$

Newmark's method

When $\tau = \Delta t$, we get the Newmark method, which is unconditionally stable if:

$$a \geq \frac{1}{2}; \quad b \geq \frac{1}{2} \left(a + \frac{1}{2} \right)^2. \quad (5.154)$$

	Central finite differences	Houbolt	Newmark – Wilson
Preliminary operations Define the parameters:	Δt	Δt	$\Delta t, \theta, a, b$
Assemble and triangularize: $[\bar{K}] = a_0 [M] + a_1 [C] + a_2 [K]$	$a_0 = 1, a_1 = \frac{\Delta t}{2}, a_2 = 0$	$a_0 = 2, a_1 = \frac{11}{16} \Delta t, a_2 = \Delta t^2$	$a_0 = 1, a_1 = \theta \Delta t a,$ $a_2 = \frac{(\theta - \Delta t)^2}{2} b$
Initialize the vectors:	$\{U_{t_0}\}$ known $\{U_{t_0 - \Delta t}\}$ by (5.145)	$\{U_{t_0}\}$ known $\{U_{t_0 - \Delta t}\}$ Another $\{U_{t_0 - 2\Delta t}\}$ method	$\{U_{t_0}\}$ known $\{\dot{U}_{t_0}\}$ known $\{\ddot{U}_{t_0}\}$ by (5.142)
At each time-step			
Calculate the residuals			
Solve the system in $\{U_{t+\Delta t}\}$	(5.144a) Transfer of 2 vectors	(5.147) Transfer of 3 vectors	(5.153) Calculation of $\{U\}$, $\{\dot{U}\}$ and $\{\ddot{U}\}$ by (5.155) – (5.158) in the case of Wilson($\theta = 1$)
Prepare the next step			

Figure 5.23. Operations corresponding to the various methods of direct integration of second-order systems

When $a = \frac{1}{2}$, a stability condition can be written [KAT 77]:

$$\Delta t \leq (l/c) \sqrt{\frac{1}{1-2b}}$$

where: c is the speed of wave propagation in the medium; for an elastic solid:

$$c^2 = \frac{E(1-\nu)}{\rho(1+\nu)(1-2\nu)}, \quad E \text{ being the elastic modulus and } \nu \text{ Poisson's coefficient;}$$

l is the smallest dimension of an element.

When $b = 0$, the method is explicit. The most commonly used values are $a = b = \frac{1}{2}$.

After solving (5.153), we have to calculate $\{\ddot{U}_{t+\Delta t}\}$ and then $\{\dot{U}_{t+\Delta t}\}$ using (5.152).

Wilson's method

Wilson's method corresponds to the case where $\tau = \theta \Delta t$ with $\theta > 1$. Hence, it consists of applying Newmark's method to construct the solution $\{U_{t+\theta\Delta t}\}$ at time $t + \theta \Delta t$ by solving (5.153). Then, we have to calculate the solution in $t + \Delta t$ necessary for the next time-step.

To this end, we successively construct:

— $\{\ddot{U}_{t+\theta\Delta t}\}$ using (5.152b):

$$\{\ddot{U}_{t+\theta\Delta t}\} = \frac{2}{b(\theta\Delta t)^2} (\{U_{t+\theta\Delta t}\} - \{U_t\}) - \frac{2}{b\theta\Delta t} \{\dot{U}_t\} - \left(\frac{1}{b} - 1\right) \{\ddot{U}_t\}. \quad (5.155)$$

— $\{\ddot{U}_{t+\Delta t}\}$ using the hypothesis of linear variation of $\{\ddot{U}\}$:

$$\{\ddot{U}_{t+\Delta t}\} = \{\ddot{U}_t\} + \frac{1}{\theta} (\{\ddot{U}_{t+\theta\Delta t}\} - \{\ddot{U}_t\}). \quad (5.156)$$

— $\{\dot{U}_{t+\Delta t}\}$, using (5.152a) where $\tau = \Delta t$:

$$\{\dot{U}_{t+\Delta t}\} = \{\dot{U}_t\} + \Delta t ((1-a) \{\ddot{U}_t\} + a \{\ddot{U}_{t+\Delta t}\}). \quad (5.157)$$

— $\{U_{t+\Delta t}\}$, using (5.152b) where $\tau = \Delta t$:

$$\{U_{t+\Delta t}\} = \{U_t\} + \Delta t \{\dot{U}_t\} + \frac{\Delta t^2}{2} ((1-b) \{\ddot{U}_t\} + b \{\ddot{U}_{t+\Delta t}\}). \quad (5.158)$$

Increasing the value of θ enables us to increase the damping of the high-frequency modes of vibration. An unconditionally stable method which is frequently used corresponds to $a = \frac{1}{2}$, $b = \frac{1}{3}$, $\theta = 1.4$.

For nonlinear problems, we use relations similar to (5.149)–(5.151), based on the matrix $[\bar{K}]$ and the residual defined by (5.153).

Figure 5.23 represents the operations necessary to implement the Newmark–Wilson algorithm.

EXAMPLE 5.37. Resolution of a two-variable, second-order problem using the central finite difference and Newmark methods

Consider the system:

$$[M]\{\ddot{U}\} + [C]\{\dot{U}\} + [K]\{U\} - \{F\}; \quad \{U_0\} = \{\dot{U}_0\} = 0$$

where:

$$[M] = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}; \quad [C] = [K] = \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}; \quad \{F\} = \begin{Bmatrix} 1 \\ 2 \end{Bmatrix}.$$

Substituting $\{U_0\}$ and $\{\dot{U}_0\}$, we find:

$$[M]\{\ddot{U}_0\} = \begin{Bmatrix} 1 \\ 2 \end{Bmatrix} \quad \{\dot{U}_0\} = \begin{Bmatrix} 0 \\ 1 \end{Bmatrix}.$$

Central finite differences ($\Delta t = 0.5$, algorithm (5.144a))

$$\begin{aligned} \{R_t\} = 0.25\{F_t\} + [M](2\{U_t\} - \{U_{t-\Delta t}\}) + \\ + 0.25[C]\{U_{t-\Delta t}\} - 0.25[K]\{U_t\}. \end{aligned}$$

According to (5.144a):

$$[\bar{K}] = [M] + 0.25[C] = \begin{bmatrix} 1.25 & 1.25 \\ 1.25 & 2.75 \end{bmatrix}.$$

According to (5.145):

$$\{U_{-\Delta t}\} = \begin{Bmatrix} 0 \\ 0.125 \end{Bmatrix}.$$

The results are as follows:

t	u_1	u_2
0	0	0
0.5	0.000	0.125
1.0	0.083	0.292
1.5	0.233	0.417
2.0	0.399	0.486
2.5	0.535	0.514
3.0	0.619	0.519
4.0	0.638	0.508
5.0	0.561	0.501
6.0	0.495	0.499
7.0	0.474	0.500
8.0	0.482	0.500
9.0	0.496	0.500
10.0	0.503	0.500
15.0	0.499	0.500

Newmark: ($\Delta t = 0.5$, $\theta = 1$, $a = b = 0.5$, algorithm (5.153))

$$[\bar{K}] = [M] + 0.25[C] + 0.0625[K] = \begin{bmatrix} 1.3125 & 1.3125 \\ 1.3125 & 2.9375 \end{bmatrix}.$$

$$\begin{aligned} \{R_{t+\Delta t}\} = & 0,0625 \{F_{t+\Delta t}\} + [M] (\{U_t\} + 0,5 \{\dot{U}_t\} + 0,0625 \{\ddot{U}_t\}) \\ & + [C] (0,25 \{U_t\} + 0,0625 \{\dot{U}_t\}). \end{aligned}$$

After each time-step, the velocities and accelerations are obtained by (5.152b) and (5.152a):

$$\{\ddot{U}_{t+\Delta t}\} = -\{\ddot{U}_t\} + 16(\{U_{t+\Delta t}\} - \{U_t\} - 0,5 \{\dot{U}_t\})$$

$$\{\dot{U}_{t+\Delta t}\} = \{\dot{U}_t\} + 0,25(\{\ddot{U}_t\} + \{\ddot{U}_{t+\Delta t}\}).$$

t	u_1	u_2	\dot{u}_1	\dot{u}_2	\ddot{u}_1	\ddot{u}_2
0	0	0	0	0	0	1
0.5	0.018	0.077	0.073	0.308	0.293	0.231
1.0	0.090	0.237	0.213	0.331	0.265	-0.136
1.5	0.219	0.379	0.303	0.239	0.096	-0.235
2.0	0.371	0.471	0.307	0.129	-0.078	-0.201
2.5	0.509	0.515	0.241	0.048	-0.187	-0.126
3.0	0.604	0.528	0.141	0.002	-0.215	-0.058
4.0	0.651	0.515	-0.035	-0.019	-0.120	0.007
5.0	0.581	0.502	-0.090	-0.007	0.003	0.011
6.0	0.504	0.498	-0.057	0.000	0.051	0.003
7.0	0.471	0.499	-0.010	0.001	0.039	0.000
8.0	0.475	0.500	0.014	0.000	0.011	-0.001
9.0	0.491	0.500	0.015	0.000	-0.007	0.000
10.0	0.503	0.500	0.007	0.000	-0.009	0.000
15.0	0.499	0.500	0.000	0.000	0.001	0.000

EXAMPLE 5.38. Stability of central finite difference and Newmark schemes

Consider the following scalar relation:

$$\ddot{u} + \omega^2 u = 0, \text{ with period } T = \frac{2\pi}{\omega}.$$

a) The scheme discretized by a central finite difference technique is written as:

$$u^{t+\Delta t} - (2 - \omega^2 \Delta t^2)u^t + u^{t-\Delta t} = 0.$$

— The positivity of the scheme is assured for:

$$\omega \Delta t < \sqrt{2}, \text{ so that } \Delta t < \frac{T}{\sqrt{2\pi}}.$$

— If we express the solution in the form $u^t = e^{i p t} = \lambda$, the stability analysis by Neumann decomposition is written:

$$\lambda^2 - (2 - \omega^2 \Delta t^2)\lambda + 1 = 0; \lambda = e^{i p \Delta t} = \cos(p \Delta t) + i \sin(p \Delta t),$$

hence:

$$\lambda = \left(1 - \frac{\omega^2 \Delta t^2}{2}\right) \pm i \sqrt{1 - \left(1 - \frac{\omega^2 \Delta t^2}{2}\right)^2}, \text{ where } \omega^2 \Delta t^2 < 2,$$

$$|\lambda| = 1 \text{ (without damping)} \text{ and } \cos(p \Delta t) = 1 - \frac{\omega^2 \Delta t^2}{2}.$$

The scheme is stable on condition that (positive term in the square root):

$$\omega \Delta t < \sqrt{2}, \text{ so that } \Delta t < \frac{T}{\sqrt{2\pi}}.$$

b) The scheme discretized by the Newmark method is written as:

$$\begin{aligned} \left(1 + \frac{\omega^2 \Delta t^2}{4}\right)u^{t+\Delta t} &= \left(1 - \frac{\omega^2 \Delta t^2}{4}\right)u^t + \Delta t \dot{u}^t, \\ \dot{u}^{t+\Delta t} &= \dot{u}^t - \frac{\Delta t \omega^2}{2}(u^t + u^{t+\Delta t}). \end{aligned}$$

These relations can be brought together in the form:

$$\left(1 + \frac{\omega^2 \Delta t^2}{4}\right)u^{t+2\Delta t} - 2\left(1 - \frac{\omega^2 \Delta t^2}{4}\right)u^{t+\Delta t} + \left(1 + \frac{\omega^2 \Delta t^2}{4}\right)u^t = 0.$$

The Neumann stability analysis, if we posit that $u^t = e^{ip t}$ and $\lambda = e^{ip \Delta t}$, gives us:

$$\lambda^2 - 2\left(\frac{1 - \omega^2 \Delta t^2/4}{1 + \omega^2 \Delta t^2/4}\right)\lambda + 1 = 0,$$

hence:

$$\lambda = \left(\frac{1 - \omega^2 \Delta t^2/4}{1 + \omega^2 \Delta t^2/4}\right) \pm i \sqrt{1 - \left(\frac{1 - \omega^2 \Delta t^2/4}{1 + \omega^2 \Delta t^2/4}\right)^2},$$

$$|\lambda| = 1 \text{ and } \cos(p\Delta t) = \left(\frac{1 - \omega^2 \Delta t^2/4}{1 + \omega^2 \Delta t^2/4}\right).$$

Thus, the scheme is unconditionally stable and there is no damping.

Remark

The Neumann decomposition enables us to transform any term from a spatial derivative into a form similar to that given in Example 5.38. For illustrative purposes, consider the following relation:

$$\frac{d^2 u}{dt^2} + k \frac{d^2 u}{dx^2} = 0.$$

By expressing the solution in the form:

$$u(x, t) = u_m e^{ip \frac{x}{\Delta x}},$$

the above relation becomes:

$$\frac{d^2 u_m}{dt^2} + \frac{k p^2}{\Delta x^2} u_m = 0.$$

The rest of the stability analysis is then identical to that presented in Example 5.38.

5.4.4.4 Explicit Runge–Kutta scheme [KRE 88]

The fourth-order Runge–Kutta scheme is often used for rapid dynamic problems with diagonalization of the mass matrix. Consider the dynamic system described by the relation:

$$\begin{aligned} [M_d]\{\ddot{U}\} + [C]\{\dot{U}\} + [K]\{U\} - \{F(t)\} &= \{0\}, \\ \{\ddot{U}\} &= [M_d]^{-1}(\{F(t)\} - [C]\{\dot{U}\} - [K]\{U\}) \\ &= \{f(t, U, \dot{U})\}. \end{aligned}$$

The Runge–Kutta scheme is written as:

$$\begin{aligned} \{U^{t+\Delta t}\} &= \{U^t\} + \Delta t \left\{ \dot{U}^t + \frac{1}{3}(g_1 + g_2 + g_3) \right\}, \\ \{\dot{U}^{t+\Delta t}\} &= \{\dot{U}^t\} + \frac{1}{3}\{g_1 + 2g_2 + 2g_3 + g_4\}, \end{aligned}$$

with:

- $\{g_1\} = \frac{1}{2}\Delta t \{f(t_n, U^t, \dot{U}^t)\},$
- $\{g_2\} = \frac{1}{2}\Delta t \left\{ f\left(t_n + \frac{\Delta t}{2}, U_2, \dot{U}_2\right) \right\},$
where $\{U_2\} = \{U^t\} + \frac{1}{2}\Delta t \left(\{\dot{U}^t\} + \frac{1}{2}\{g_1\} \right),$
 $\{\dot{U}_2\} = \{\dot{U}^t\} + \{g_1\}.$

- $\{g_3\} = \frac{1}{2}\Delta t \left\{ f\left(t_n + \frac{\Delta t}{2}, U_2, \dot{U}_3\right) \right\}$,
where $\{\dot{U}_3\} = \{\dot{U}^t\} + \{g_2\}$.
- $\{g_4\} = \frac{1}{2}\Delta t \left\{ f\left(t_n + \Delta t, U_4, \dot{U}_4\right) \right\}$,
with $\{U_4\} = \{U^n\} + \Delta t \left(\{\dot{U}^t\} + \{g_3\} \right)$,
 $\{\dot{U}_4\} = \{\dot{U}^t\} + 2\{g_3\}$.

5.4.5 MODAL SUPERPOSITION METHOD FOR SECOND-ORDER SYSTEMS [WIL 77]

This method is similar to that described in section 5.4.3 for first-order systems. However, for second-order systems (5.87), we use the eigenvalues and eigenvectors of

$$([K] - \lambda[M])\{X\} = 0. \quad (5.159)$$

Following the transformation (5.135), system (5.87) becomes:

$$\{\ddot{V}(t)\} + [X]^T [C][X]\{\dot{V}(t)\} + [\lambda]\{V(t)\} = \{\bar{F}(t)\}. \quad (5.160)$$

This system is only uncoupled in one of the following hypotheses:

- the damping matrix $[C]$ is null;
- the matrix $[C]$ can be defined as a linear combination of $[M]$ and $[K]$:

$$[C] = c_1[M] + c_2[K]. \quad (5.161)$$

Thus:

$$[X]^T [C][X] = c_1[I] + c_2[\lambda].$$

- the matrix $[X]^T [C][X]$ can be replaced with a diagonal matrix, often of the following form:

$$\begin{bmatrix} 2\xi_1\omega_1 & & & 0 \\ & 2\xi_2\omega_2 & & \\ & & \ddots & \\ & & & 2\xi_i\omega_i \\ 0 & & & \end{bmatrix}; \quad \omega_i = \sqrt{\lambda_i} \quad (5.162)$$

where the coefficient ξ_i is the damping ratio of the mode i over its critical damping $2\omega_i$. Note that the Hypothesis (5.161) is tantamount to choosing only two non-null ξ_i values.

When the equations (5.160) are uncoupled, they are written as:

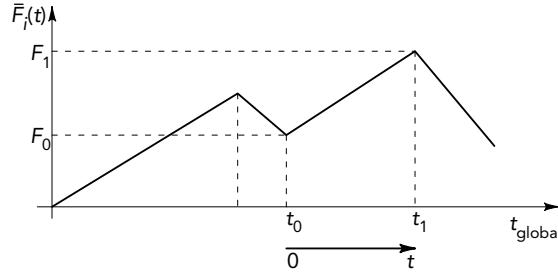
$$\ddot{V}_i(t) + 2\omega_i \xi_i \dot{V}_i(t) + \omega_i^2 V_i(t) = \bar{F}_i(t). \quad (5.163)$$

The general solution is expressed in the form of the Duhamel integral, supposing that $t_0 = 0$:

$$\begin{aligned} V_i(t) &= e^{-\xi_i \omega_i t} (\alpha_1 \sin \bar{\omega}_i t + \alpha_2 \cos \bar{\omega}_i t) + \\ &+ \frac{1}{\bar{\omega}_i} \int_0^t e^{-\xi_i \omega_i (t-s)} \sin \bar{\omega}_i (t-s) \cdot \bar{F}_i(s) ds \end{aligned} \quad (5.164)$$

$$\bar{\omega}_i = \omega_i \sqrt{1 - \xi_i^2}.$$

In practice, the vector $\{\bar{F}_i(t)\}$ is often defined in a linear form in t at intervals:



$$\bar{F}_i(t) = F_0 + \frac{F_1 - F_0}{t_1 - t_0} t = a + bt \quad \text{for } 0 \leq t \leq t_1 - t_0.$$

In this case, the solution to (5.163) is explicit:

$$V_i(t_0 + t) = a_0 + a_1 t + e^{-\xi_i \omega_i t} (a_2 \cos \bar{\omega}_i t + a_3 \sin \bar{\omega}_i t) \quad (5.165)$$

$$a_0 = \frac{a}{\omega_i^2} - 2 \frac{\xi_i b}{\omega_i^3}; \quad a_1 = \frac{b}{\omega_i^2}$$

$$a_2 = V_i(t_0) - a_0; \quad a_3 = \frac{1}{\bar{\omega}_i} (\dot{V}_i(t_0) + \xi_i \omega_i a_2 - a_1).$$

EXAMPLE 5.39. Resolution of a two-variable, second-order problem by modal superposition

Let us use modal superposition to solve the system defined in Example 5.37. The eigenvalues and eigenvectors of:

$$\left(\begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix} + \lambda \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \right) \{X\} = 0$$

have already been obtained in Example 5.37:

$$\begin{aligned} \lambda_1 &= 1 & \{X_1\} &= \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} & [X] &= \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix} \\ \lambda_2 &= 2 & \{X_2\} &= \begin{Bmatrix} 1 \\ -1 \end{Bmatrix} \end{aligned}$$

The system (5.160) is uncoupled because $[C]$ is a linear combination of $[M]$ and $[K]$:

$$[C] = 0 \cdot \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} + 1 \cdot \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}$$

Hence:

$$\{\ddot{V}\} + \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \{\dot{V}\} + \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \{V\} = \begin{Bmatrix} 1 \\ -1 \end{Bmatrix}$$

so:

$$\begin{aligned} \ddot{V}_1 + \dot{V}_1 + V_1 &= 1 \\ \ddot{V}_2 + 2\dot{V}_2 + 2V_2 &= -1. \end{aligned}$$

The initial conditions are obtained, using (5.140):

$$\{V_0\} = [X]^T [M] \{U_0\} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}; \quad \{\dot{V}_0\} = [X]^T [M] \{\dot{U}_0\} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}.$$

The solution of the first uncoupled equation is obtained by (5.165):

$$a = 1, \quad b = 0, \quad \omega_1 = 1, \quad \xi_1 = \frac{1}{2}, \quad \bar{\omega}_1 = \frac{\sqrt{3}}{2}$$

$$a_0 = 1, \quad a_1 = 0, \quad a_2 = -1, \quad a_3 = -\frac{1}{\sqrt{3}}$$

$$V_1 = 1 - e^{-t/2} \left(\cos \frac{\sqrt{3}}{2} t + \frac{\sqrt{3}}{3} \sin \frac{\sqrt{3}}{2} t \right).$$

For the second uncoupled equation:

$$\begin{aligned} a &= -1, \quad b = 0, \quad \omega_2 = \sqrt{2}, \quad \xi_2 = \frac{\sqrt{2}}{2}, \quad \bar{\omega}_2 = 1 \\ a_0 &= -\frac{1}{2}, \quad a_1 = 0, \quad a_2 = \frac{1}{2}, \quad a_3 = \frac{1}{2} \\ V_2 &= -\frac{1}{2} + e^{-t} \left(\frac{1}{2} \cos t + \frac{1}{2} \sin t \right). \end{aligned}$$

The solution of the initial system is:

$$\begin{aligned} \{U\} &= [X] \begin{Bmatrix} V_1 \\ V_2 \end{Bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix} \times \\ &\quad \times \begin{Bmatrix} 1 - e^{-t/2} \left(\cos \frac{\sqrt{3}}{2}t + \frac{\sqrt{3}}{3} \sin \frac{\sqrt{3}}{2}t \right) \\ -\frac{1}{2} + e^{-t} \left(\frac{1}{2} \cos t + \frac{1}{2} \sin t \right) \end{Bmatrix} \\ \{U\} &= \begin{Bmatrix} \frac{1}{2} - e^{-t/2} \left(\cos \frac{\sqrt{3}}{2}t + \frac{\sqrt{3}}{3} \sin \frac{\sqrt{3}}{2}t \right) + e^{-t} \left(\frac{1}{2} \cos t + \frac{1}{2} \sin t \right) \\ \frac{1}{2} - e^{-t} \left(\frac{1}{2} \cos t + \frac{1}{2} \sin t \right) \end{Bmatrix} \end{aligned}$$

t	u_1	u_2
0	0	0
0.5	0.016	0.088
1.0	0.094	0.246
1.5	0.230	0.381
2.0	0.383	0.466
2.5	0.515	0.508
3.0	0.603	0.521
4.0	0.640	0.513
5.0	0.572	0.502
6.0	0.503	0.499
7.0	0.475	0.499
8.0	0.479	0.500
9.0	0.493	0.500
10.0	0.502	0.500
15.0	0.499	0.500

5.5 Methods for calculating the eigenvalues and eigenvectors [BAT 76; CRA 56; WIL 65]

5.5.1 INTRODUCTION

Solving an eigenvalue problem consists of finding the couples λ_i and $\{X_i\}$, which satisfy the following relation:

$$[K]\{X_i\} = \lambda_i[M]\{X_i\}. \quad (5.166)$$

This type of problem appears, for example, in the following domains:

Vibration of structures

The determination of the eigenmodes of vibration of a structure leads to a relation similar to (5.166), in which:

- $[K]$ is the stiffness matrix of the structure;
- $[M]$ is the mass matrix;
- $\{X_i\}$ is the vector of the structure's movements defining the i -th eigenmode of vibration;
- $\lambda_i = \omega_i^2$ is the square of the corresponding pulsation.

Critical buckling load of a structure

The determination of the critical linear buckling load P of a structure also yields (5.166), where:

- $[K]$ is the stiffness matrix of the structure;
- $[M] \equiv [K_G]$ is the geometric matrix (or initial strains) of the structure;
- $\{X_i\}$ is the vector of the structure's movements defining the i -th buckling mode;
- λ_i defines the amplitude of the critical load.

Resolution of first- and second-order unsteady problems

We have seen in sections 5.4.3 and 5.4.5 that the modal superposition method requires us to calculate the eigenvalues and eigenvectors of a system similar to (5.166).

The domain of calculation of eigenvalues and eigenvectors (5.166) is vast. The works of Wilkinson [WIL 65] and Bathe and Wilson [BAT 76] offer a detailed presentation of the available methods. Here, we will content ourselves with

briefly recapping the properties of (5.166) without a demonstration and presenting three very commonly used methods of solution: inverse iteration, the Jacobian method and the subspace method. Numerous other methods have been developed, but they fall beyond the scope of this book.

5.5.2 RECAP OF SOME PROPERTIES OF EIGENVALUE PROBLEMS

5.5.2.1 Simplified formulation

We suppose that the matrices $[K]$ and $[M]$ in (5.166) are symmetrical and real, and that at least one of these two matrices is positive definite. It is then possible to go back to the simplified formulation:

$$[\bar{K}] \{ \bar{X} \} = \lambda \{ \bar{X} \}. \quad (5.167)$$

If $[M]$ is positive definite, we need only decompose it in the form (5.38):

$$[M] = [L][L]^T \quad (5.168)$$

and posit

$$\begin{aligned} \{ \bar{X} \} &= [L]^T \{ X \} \\ [\bar{K}] &= [L]^{-1} [K] [L]^{-1 T}. \end{aligned}$$

If $[K]$ is positive definite and $[M]$ is not, we replace (5.166) with:

$$[M] \{ X \} = \frac{1}{\lambda} [K] \{ X \} = \lambda' [K] \{ X \} \quad (5.169)$$

and then decompose $[K]$ to get the system of the form (5.167).

5.5.2.2 Eigenvalues

There are n real eigenvalues λ_i , distinct or otherwise, which satisfy (5.166). If both matrices are positive definite, the values λ_i are positive. We can order them: $\lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \dots \leq \lambda_n$. We rewrite (5.166) in the form:

$$([K] - \lambda[M]) \{ X \} = 0. \quad (5.170)$$

There are non-null vectors $\{ X \}$ which satisfy (5.170) only if $[K] - \lambda[M]$ is singular. The objective is to find λ 's which satisfy :

$$\det ([K] - \lambda[M]) = P(\lambda) = 0. \quad (5.171)$$

This expression is an n -order characteristic polynomial in λ , which is difficult to construct explicitly for large systems. The search for the eigenvalues λ_i is therefore identical to the search for the roots of an n -order polynomial. There is no direct method for $n > 4$. We have to use iterative methods.

5.5.2.3 Eigenvectors

For each value λ_i there is a corresponding eigenvector $\{X_i\}$ which is a solution of

$$([K] - \lambda_i [M]) \{X_i\} = 0. \quad (5.172a)$$

In order to solve this singular system, we have to fix the value of at least one component of $\{X_i\}$ *a priori*. If $\{X_i\}$ is a solution to (5.172a), so too is $\alpha \{X_i\}$. We normalize the eigenvectors in the following manner:

$$\langle X_i \rangle [M] \{X_i\} = 1. \quad (5.172b)$$

The eigenvectors satisfy the following orthogonality relations:

$$\langle X_i \rangle [K] \{X_j\} = \lambda_i \delta_{ij}; \quad \delta_{ij} = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \quad (5.173a)$$

$$\langle X_i \rangle [M] \{X_j\} = \delta_{ij}. \quad (5.173b)$$

We say that the eigenvectors are “[K]-orthogonal” and “[M]-orthonormal”.

We group all the eigenvectors together in a matrix called a “modal matrix”:

$$[X] = [\{X_1\} \dots \{X_n\} \dots]. \quad (5.174)$$

The relations (5.166) corresponding to the ensemble of eigenvectors are written as:

$$[K][X] = [M][X][\lambda] \quad (5.175)$$

$$[\lambda] = \begin{bmatrix} \lambda_1 & & & & 0 \\ & \lambda_2 & & & \\ & & \ddots & & \\ & & & \lambda_i & \\ & & & & 0 \end{bmatrix}.$$

Thus, the relations (5.173) become:

$$[X]^T [K] [X] = [\lambda] \quad (5.176a)$$

$$[X]^T [M] [X] = [I] \quad (5.176b)$$

or indeed by multiplying the left-hand side of (5.176b) by $[X]$ and the right-hand side by $[X]^{-1}$:

$$[X] [X]^T [M] = [I]. \quad (5.177)$$

5.5.2.4 Spectral decomposition

By multiplying the right-hand side of (5.175) by $[X]^T [M]$ and using (5.177), we get (because $[M]$ is symmetrical):

$$[K] = [M] [X] \cdot [\lambda] \cdot ([M] [X])^T \quad (5.178)$$

or by positing $\{Y_i\} = [M] \{X_i\}$

$$[K] = \sum_{i=1}^n \lambda_i \{Y_i\} \langle Y_i \rangle. \quad (5.179a)$$

The spectral decomposition of $[K]$ is defined by (5.179a) in the particular case where $[M] \equiv [I]$:

$$[K] = \sum_{i=1}^n \lambda_i \{X_i\} \langle X_i \rangle = \sum_{i=1}^n \lambda_i [P_i] \quad (5.179b)$$

where the symmetrical matrices $[P_i]$ are the projectors of $[K]$. In particular, this relation helps to define $[K]$ to the power p :

$$[K]^p = \sum_{i=1}^n \lambda_i^p [P_i] \quad (\text{is positive or negative}). \quad (5.180)$$

The spectral radius of $[K]$ is defined by:

$$\rho(K) = \max |\lambda_i| \leq \|K\|_\infty \quad (5.181)$$

where:

$$\|K\|_\infty = \max_i \left(\sum_{j=1}^n |K_{ij}| \right).$$

We obtain the condition so that $[K]^p$ will remain bounded as p tends toward infinity:

$$\rho(K) < 1. \quad (5.182)$$

Any vector V can be represented in the base of eigenvectors of the system (5.166):

$$\{V\} = \sum_{i=1}^n c_i \{X_i\}. \quad (5.183)$$

Its components c_i are:

$$c_i = \langle X_i \rangle [M] \{V\}. \quad (5.184)$$

5.5.2.5 Transformation of $[K]$ and $[M]$

Let us transform the matrices $[K]$ and $[M]$ as follows:

$$[\bar{K}] = [Q]^T [K] [Q]; \quad [\bar{M}] = [Q]^T [M] [Q]. \quad (5.185a)$$

The eigenvalues and eigenvectors corresponding to $[\bar{K}]$ and $[\bar{M}]$ are defined by:

$$([\bar{K}] - \bar{\lambda}[\bar{M}]) \{\bar{X}\} = 0 \quad (5.185b)$$

or indeed:

$$[Q]^T ([K] - \bar{\lambda}[M]) [Q] \{\bar{X}\} = 0.$$

$$\begin{aligned} \text{Thus: } \det([\bar{K}] - \bar{\lambda}[\bar{M}]) &= \det([Q]^T ([K] - \bar{\lambda}[M]) [Q]) \\ &= \det([Q]^T) \cdot \det([Q]) \cdot \det([K] - \bar{\lambda}[M]) = 0. \end{aligned}$$

Hence, the eigenvalues $\bar{\lambda}$ of the transformed system (5.185b) are identical to the eigenvalues λ of the system (5.170) if $\det([Q]) \neq 0$. Note that we have in fact used this type of transformation in section 5.5.2.1. The eigenvectors $\{\bar{X}\}$ of (5.185b) are linked to those of (5.170) by:

$$\{X\} = [Q] \{\bar{X}\}. \quad (5.185c)$$

EXAMPLE 5.40. Eigenvalues and eigenvectors

Consider the following eigenvalue problem:

$$[K] \{X_i\} = \lambda_i [M] \{X_i\}; \quad [K] = \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}; \quad [M] = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}.$$

The matrices $[K]$ and $[M]$ are positive definite, so the eigenvalues are real and positive; they are roots of the polynomial:

$$\det \left(\begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \right) = (1 - \lambda)(2 - \lambda) = 0.$$

Hence:

$$\lambda_1 = 1$$

$$\lambda_2 = 2.$$

The eigenvectors are solutions of the singular systems:

$$\begin{aligned} \left(\begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix} - 1 \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \right) \{X_1\} = 0 \rightarrow \{X_1\} = \begin{Bmatrix} x_1 \\ 0 \end{Bmatrix} \\ \left(\begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix} - 2 \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \right) \{X_2\} = 0 \rightarrow \{X_2\} = \begin{Bmatrix} x_2 \\ -x_2 \end{Bmatrix}. \end{aligned}$$

The values of x_1 and x_2 are obtained by using the condition of M-orthonormality of the vectors $\{X_1\}$ and $\{X_2\}$ (5.173b):

$$\{X_1\} = \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} \quad \{X_2\} = \begin{Bmatrix} 1 \\ -1 \end{Bmatrix}$$

$$[X] = \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix}.$$

The orthogonality relations (5.176) are written as:

$$\begin{aligned} \begin{bmatrix} 1 & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, \\ \begin{bmatrix} 1 & 0 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \end{aligned}$$

The spectral decomposition (5.179a) of the matrix $[K]$ is:

$$\begin{aligned} \{Y_1\} &= \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{Bmatrix} 1 \\ 0 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}; \\ \{Y_2\} &= \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{Bmatrix} 1 \\ -1 \end{Bmatrix} = \begin{Bmatrix} 0 \\ -1 \end{Bmatrix}; \\ [K] &= 1 \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} + 2 \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}. \end{aligned}$$

5.5.2.6 Rayleigh quotient

The Rayleigh quotient of any vector $\{V\}$ is:

$$R(\{V\}) = \frac{\langle V \rangle [K] \{V\}}{\langle V \rangle [M] \{V\}}. \quad (5.186)$$

If $[K]$ and $[M]$ are positive definite, $R(V) > 0$; in addition, no matter what the vector $\{V\}$ is, the value of $R(\{V\})$ is between the smallest and largest eigenvalues in (5.170):

$$\lambda_1 \leq R(\{V\}) \leq \lambda_n. \quad (5.187)$$

When the vector $\{V\}$ is equal to the eigenvector $\{X_i\}$ of (5.170), the Rayleigh coefficient is equal to the eigenvalue λ_i :

$$R(\{X_i\}) = \lambda_i. \quad (5.188)$$

If $\{V\}$ is a linear combination of the first p eigenvectors, the Rayleigh coefficient is between λ_1 and λ_p :

$$\lambda_1 \leq R\left(\sum_{i=1}^p c_i \{X_i\}\right) \leq \lambda_p \quad (5.189)$$

and the maximum value of R is reached when $c_1 = c_2 = \dots = c_{p-1} = 0$; $c_p = 1$. Suppose that $\{V\}$ is near to the eigenvector $\{X_i\}$ and is written:

$$\{V\} = \{X_i\} + \varepsilon \{Z\} \quad (5.190)$$

where: ε is a very small parameter;
 $\{Z\}$ is any disturbance vector.

The Rayleigh coefficient is then close to λ_i and takes the form:

$$R(\{V\}) = \lambda_i + O(\varepsilon^2). \quad (5.191)$$

This means that the Rayleigh quotient is stationary in relation to ε when V is an eigenvector.

5.5.2.7 Separation of the eigenvalues

Let $\lambda_1, \lambda_2, \dots, \lambda_n$ represent the n eigenvalues of the system (5.170) and l_1, l_2, \dots, l_{n-1} the $n-1$ eigenvalues of the system:

$$[K]^1 \{X\} = l[M]^1 \{X\} \quad (5.192)$$

in which $[K]^1$ and $[M]^1$ are obtained by eliminating the last row and the last column of $[K]$ and $[M]$. The separation property of the eigenvalues is thus written as:

$$\lambda_1 \leq l_1 \leq \lambda_2 \leq l_2 \leq \dots \leq \lambda_{n-1} \leq l_{n-1} \leq \lambda_n. \quad (5.193)$$

More generally, we define $[K]^r$ and $[M]^r$ as the matrices obtained by eliminating the last r rows and columns from $[K]$ and $[M]$. Let $P^r(\lambda)$ be the corresponding characteristic polynomial. The roots of $P^{r-1}(l)$ “separate” the roots of $P^r(\lambda)$, meaning that:

$$\lambda_1 \leq l_1 \leq \lambda_2 \leq \dots \leq l_{n-r-2} \leq \lambda_{n-r-1} \leq l_{n-r-1} \leq \lambda_{n-r}. \quad (5.194)$$

The series of polynomials $P^r(\lambda)$ constitutes a **Sturm series**.

Let us decompose the matrix $[K] - \mu[M]$ using (5.37). Of course, this is only possible if this matrix is not singular, i.e. if μ is not an eigenvalue.

$$[K] - \mu[M] = [L][D][L]^T. \quad (5.195)$$

Let us suppose that the eigenvalues of (5.170) are positive; the number of negative terms in the matrix $[D]$ is equal to the number of eigenvalues less than μ in (5.170).

5.5.2.8 Shifting of the eigenvalues

Transform the matrix $[K]$ of the system (5.170):

$$[\bar{K}] = [K] - a[M]. \quad (5.196)$$

We get:

$$[\bar{K}] \{X\} = \bar{\lambda} [M] \{X\} \text{ or } [K] \{X\} = (\bar{\lambda} + a) [M] \{X\}. \quad (5.197)$$

The eigenvectors of this system are identical to those of (5.170). The eigenvalues $\bar{\lambda}$ have undergone a shift a in relation to the eigenvalues λ of (5.170):

$$\bar{\lambda} = \lambda - a. \quad (5.198)$$

EXAMPLE 5.41. Rayleigh quotient and separation of eigenvalues

Consider matrices $[K]$ and $[M]$ from the previous example. The Rayleigh quotient (5.186) of any vector $\{V\}$ is:

$$\{V\} = \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}; R(\{V\}) = \frac{\langle 1 \ 1 \rangle \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}}{\langle 1 \ 1 \rangle \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix} \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}} = \frac{6}{5} = 1.2.$$

We verify that

$$\lambda_1 \leq R(\{V\}) \leq \lambda_2$$

where

$$\lambda_1 = 1, \quad \lambda_2 = 2.$$

Let us remove the last row and the last column from $[K]$ and $[M]$:

$$[K]^1 = [1] \quad [M]^1 = [1].$$

The eigenvalue of

$$([1] - l[1]) X = 0$$

is: $l = 1$.

We verify the property (5.193):

$$\lambda_1 \leq l \leq \lambda_2.$$

We carry out the decomposition (5.195) with $\mu = 3$

$$[K] - 3[M] = \begin{bmatrix} -2 & -2 \\ -2 & -3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} -2 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

As there are two negative terms in the diagonal matrix, all the eigenvalues are less than 3.

5.5.3 METHODS FOR CALCULATING THE EIGENVALUES

5.5.3.1 Inverse iteration method

Looking for the smallest eigenvalue λ_1

The inverse iteration method enables us to calculate the smallest eigenvalue λ_1 of the system (5.170), and the corresponding eigenvector $\{X_1\}$. $[K]$ has to be positive definite. Otherwise, it is possible to perform a shift (5.196) such that $[\bar{K}]$ is positive definite. The algorithm is shown in Figure 5.24.

Convergence

The speed of convergence of inverse iteration is proportional to $\frac{\lambda_2}{\lambda_1}$, where λ_2 is the first eigenvalue greater than λ_1 . It may become very small when λ_1 and λ_2 are near. A shift “a” improves the convergence if

$$\frac{\lambda_2 - a}{\lambda_1 - a} > \frac{\lambda_2}{\lambda_1}. \quad (5.199)$$

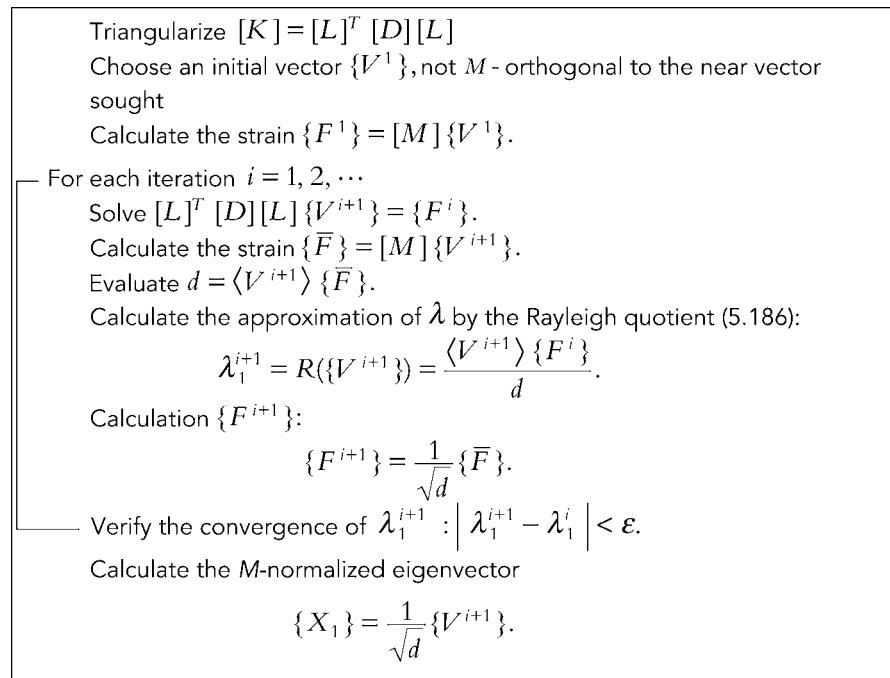


Figure 5.24. Algorithm of the inverse iteration method

The shift “a” may be equal to the Rayleigh ratio $R(\{V^{i+1}\})$. This greatly accelerates the convergence but requires $[K] - a[M]$ to be triangularized at each iteration.

Looking for the largest eigenvalue

The largest eigenvalue can be obtained by using the direct iteration method. This consists of applying the inverse iteration method to the system:

$$[M]\{X\} = \frac{1}{\lambda}[K]\{X\} = \lambda[K]\{X\}. \quad (5.200)$$

This requires that $[M]$ be positive definite. If it is not, we can use a shift such that $[\bar{M}] = [M] - a[K]$ is positive definite.

Looking for an intermediary eigenvalue

Intermediary eigenvalues λ_p , between λ_1 and λ_n can be obtained using inverse iteration, on condition that we use a shift $a \approx \lambda_p$. However, it is sometimes difficult to predict the number p of the eigenvalue to which a given shift a will lead us.

Successively looking for $\lambda_1 \lambda_2 \dots$

A systematic technique for looking for the eigenvalues $\lambda_1 \lambda_2 \lambda_3 \dots$ consists of using the inverse iteration method associated with an orthogonalization technique: suppose we have obtained λ_1 and $\{X_1\}$. To obtain λ_2 and $\{X_2\}$, we perform a new inverse iteration, forcing the vector $\{V^{i+1}\}$ in the above algorithm to remain M -orthogonal to $\{X_1\}$. To do so, we subtract from $\{V^{i+1}\}$ its projection on $\{X_1\}$ using (5.183):

$$\{V^{i+1}\} = \{V^{i+1}\} - c_1 \{X_1\}$$

where:

$$\begin{aligned} c_1 &= \langle X_1 \rangle \{ \bar{Y} \} \\ \{ \bar{Y} \} &= [M] \{ V^{i+1} \}. \end{aligned} \quad (5.201)$$

Similarly, in order to construct λ_p and $\{X_p\}$, we force $\{V^{i+1}\}$ to remain M -orthogonal to $\{X_1\}, \{X_2\}, \dots, \{X_{p-1}\}$:

$$\{V^{i+1}\} = \{V^{i+1}\} - \sum_{j=1}^{p-1} (\langle X_j \rangle \{ \bar{Y} \}) \{X_j\}. \quad (5.202)$$

Note that this orthogonalization technique requires a high degree of numerical precision in calculating the successive λ_i and $\{X_i\}$.

5.5.3.2 Jacobian method

General aspects

The general Jacobian method enables us to calculate the n eigenvalues and eigenvectors of a system whose dimensions are limited (n less than 100), where the matrices are symmetrical and positive definite.

It consists of transforming the matrices $[K]$ and $[M]$ into diagonal matrices using successive transformations:

$$\begin{aligned} [K^1] &= [K] & [M^1] &= [M] \\ [K^2] &= [Q^1]^T [K^1] [Q^1] & [M^2] &= [Q^1]^T [M^1] [Q^1] \\ &\dots &&\dots \\ [K^{k+1}] &= [Q^k]^T [K^k] [Q^k] & [M^{k+1}] &= [Q^k]^T [M^k] [Q^k]. \end{aligned} \quad (5.203)$$

The matrices $[K^{k+1}]$ and $[M^{k+1}]$ tend toward diagonal matrices $[K^d]$ and $[M^d]$ as k tends toward infinity. The eigenvalues and eigenvectors are thus:

$$[\lambda] = [K^d][M^d]^{-1} \text{ or } \lambda_i = K_{ii}^d / M_{ii}^d \quad (5.204)$$

$$[X] = [Q^1] [Q^2] \dots [Q^k] [Q^{k+1}] \begin{bmatrix} & & & 0 \\ & & \frac{1}{\sqrt{M_{ii}^d}} & \\ & 0 & & \\ & & & \ddots \end{bmatrix}.$$

Transformation matrices

Each matrix $[Q^k]$ is chosen so that a term (i, j) , non-diagonal and non-null in $[k^k]$ (and $[M^k]$) is null after transformation (5.203). The matrix $[Q^k]$ has the following structure:

$$[Q^k] = \begin{bmatrix} 1 & & & 0 \\ & 1 & a & \\ & b & 1 & \\ 0 & & & 1 \end{bmatrix} - \text{row } i - \text{row } j. \quad (5.205)$$

column i column j

The coefficients a and b are calculated by writing that $K_{ij}^{k+1} = M_{ij}^{k+1} = 0$ so that, by simply deleting the subscript $k + 1$ on the terms of each matrix:

$$\begin{aligned} a K_{ii} + (1+ab) K_{ij} + b K_{jj} &= 0 \\ a M_{ii} + (1+ab) M_{ij} + b M_{jj} &= 0. \end{aligned} \quad (5.206)$$

In the particular case where:

$$\frac{K_{ii}}{M_{ii}} = \frac{K_{jj}}{M_{jj}} = \frac{K_{ij}}{M_{ij}}$$

the values of a and b are:

$$a = 0 \quad b = -\frac{K_{ij}}{K_{jj}}. \quad (5.207)$$

Note that, generally:

$$\begin{aligned} c_1 &= K_{ii} M_{ij} - M_{ii} K_{ij} \\ c_2 &= K_{jj} M_{ij} - M_{jj} K_{ij} \\ c_3 &= K_{ii} M_{jj} - M_{ii} K_{jj} \end{aligned}$$

$$d = \frac{c_3}{2} + \text{sign}(c_3) \sqrt{\left(\frac{c_3}{2}\right)^2 + c_1 c_2}.$$

Thus

$$a = \frac{c_2}{d} \quad b = -\frac{c_1}{d}. \quad (5.208)$$

When $[M]$ is positive definite, the coefficient $\left(\frac{c_3}{2}\right)^2 + c_1 c_2$ is positive. When $d = 0$, a and b are given by (5.207).

Classic Jacobian method

In the case of a unit matrix $[M]$ (system in the form of (5.167)), the matrices $[Q^k]$ are orthogonal and are written thus (classic Jacobian method):

$$[Q^k] = \begin{bmatrix} 1 & & & \\ & \cos \theta & -\sin \theta & \\ & \sin \theta & \cos \theta & \\ & & & 1 \end{bmatrix} \begin{array}{l} \text{--- row } i \\ \text{--- row } j \end{array} \quad (5.209)$$

$$\text{where: } \operatorname{tg}(2\theta) = \frac{2K_{ij}}{K_{ii} - K_{jj}} \quad \text{if} \quad K_{ii} \neq K_{jj},$$

$$\theta = \frac{\pi}{4} \quad \text{if} \quad K_{ii} = K_{jj}.$$

In the case where $[M] = [I]$, the general formulation (5.208) gives us a matrix $[Q^k]$ similar to (5.209) to a near multiplication factor $\frac{1}{\cos \theta}$.

The algorithm corresponding to the general method above is presented in Figure 5.25.

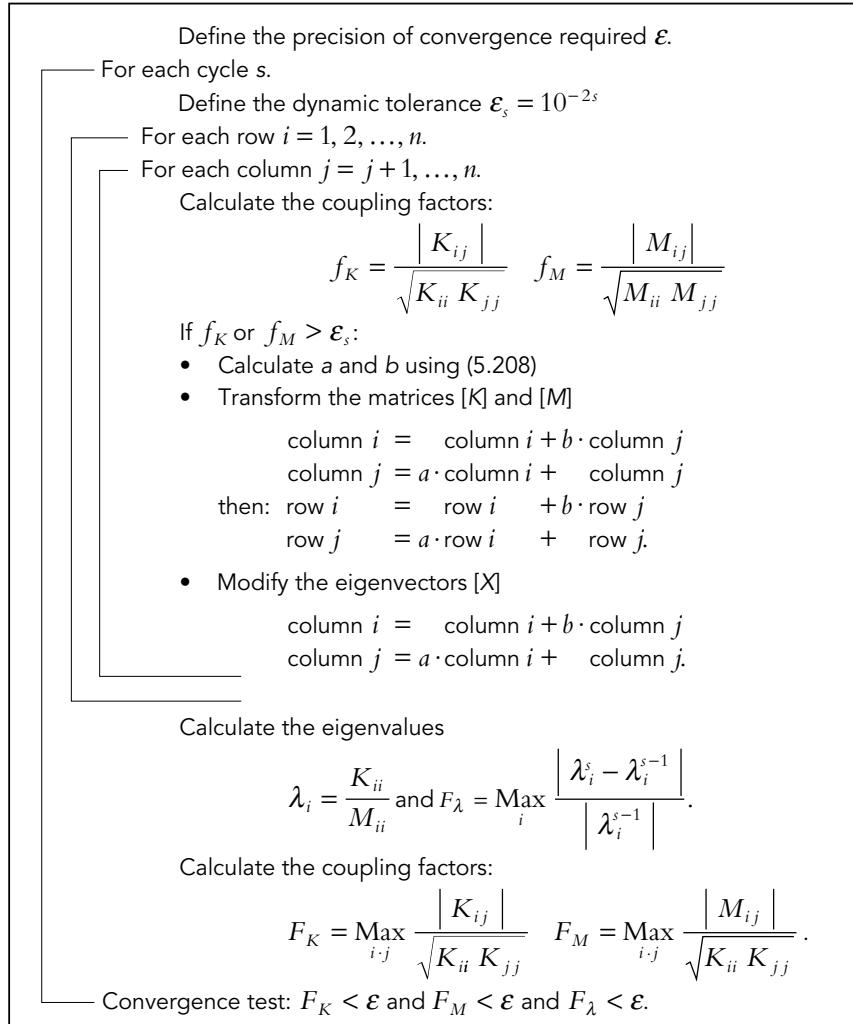


Figure 5.25. Algorithm of the general Jacobian method

5.5.3.3 Ritz method

The Ritz method allows us to transform a large eigenvalue problem into a smaller problem. We can then calculate all the eigenvalues and all the eigenvectors of the reduced system using the Jacobian method.

We assume that each eigenvector of the system (5.170) can be represented in the form of a linear combination of p independent vectors q_i , called Ritz vectors.

$$\{X\} = a_1 \{q_1\} + a_2 \{q_2\} + \cdots + a_p \{q_p\} \quad (5.210)$$

$$\begin{aligned} \{X\} &= [\{q_1\} \{q_2\} \dots \{q_p\}] \begin{Bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{Bmatrix} \\ &\quad (n \times 1) \quad (n \times p) \quad (p \times 1) \end{aligned} \quad (5.211)$$

$$\{X\} = [Q] \{a\}.$$

We look for the coefficients $\{a\}$ such that the vector $\{X\}$ is as close as possible to an eigenvector in (5.179). For this, we seek to render the Rayleigh quotient (5.186) stationary:

$$R(\{X\}) = \frac{\langle X \rangle [K] \{X\}}{\langle X \rangle [M] \{X\}} = \frac{\langle a \rangle [\bar{K}] \{a\}}{\langle a \rangle [\bar{M}] \{a\}} \quad (5.212)$$

$$[\bar{K}] = [Q]^T [K] [Q]$$

$$[\bar{M}] = [Q]^T [M] [Q].$$

The stationarity condition $\delta R = 0$ for any $\langle \delta a \rangle$ is written as:

$$([\bar{K}] - R[\bar{M}]) \{a\} = 0. \quad (5.213)$$

This expression defines an eigenvalue problem with dimension p , whose p eigenvectors $\{A_i\}$ and eigenvalues $\bar{\lambda}_i$ satisfy:

$$[\bar{K}] [A] = [\bar{M}] [A] [\bar{\lambda}] \quad (5.214)$$

where:

$$[A] = [\{A_1\} \{A_2\} \dots \{A_p\}]; \quad [\bar{\lambda}] = \begin{bmatrix} \bar{\lambda}_1 & & 0 \\ & \bar{\lambda}_2 & \\ 0 & & \bar{\lambda}_p \end{bmatrix}.$$

The eigenvalues $\bar{\lambda}_i$ constitute approximations of the eigenvalues of the original system (5.170). The approximations improve when Ritz vectors span a subspace

which contains the eigenvectors sought. In addition, the approximate eigenvalues $\bar{\lambda}_i$ and the exact ones λ_i verify a relation similar to (5.194):

$$\lambda_1 \leq \bar{\lambda}_1; \lambda_2 \leq \bar{\lambda}_2; \dots; \lambda_p \leq \bar{\lambda}_p \leq \lambda_r. \quad (5.215)$$

So as to rapidly obtain the smallest eigenvalues, we can choose, as Ritz vectors, the solutions to:

$$[K]\{q_i\} = \{F_i\}, \quad (5.216)$$

where $\{F_i\}$ are the unit vectors which excite the degree of freedom i corresponding to the smallest values of $\frac{K_{ii}}{M_{ii}}$:

$$\{F_i\} = \begin{Bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \end{Bmatrix} \leftarrow \text{row } i$$

The approximate eigenvectors of (5.170) are obtained from the vectors $\{A_i\}$ using (5.211).

5.5.3.4 Subspace method

This method is used to calculate the first p eigenvalues of a large system. It consists of repeatedly applying the Ritz method whilst improving the Ritz vectors by inverse iteration. The Ritz method forces the vectors $\{X\}$ to remain orthogonal to one another, while the inverse iteration adjusts the Ritz vectorial basis, so as to ensure convergence toward the eigenvectors corresponding to the smallest eigenvalues.

The subspace method involves the following sequence of operations:

- a) Choosing p initial vectors:

$$[X] = [\{X_1\} \{X_2\} \dots \{X_p\}]. \quad (5.217)$$

- b) Carrying out an inverse iteration (Figure 5.23) to simultaneously calculate the p Ritz vectors $\{q_i\}$ by solving:

$$[K]\{q_i\} = [M]\{X_i\} = \{F_i\} \quad i = 1, 2, \dots, p \quad (5.218)$$

$$[K][Q] = [M][X].$$

- c) Applying the Jacobi method to find the eigenvectors and eigenvalues in the Ritz subspace:

$$([\bar{K}] - \bar{\lambda}_i [\bar{M}]) \{A_i\} = 0 \quad (\text{Jacobi})$$

where:

$$\begin{aligned} [\bar{K}] &= [Q]^T [K] [Q] \\ [\bar{M}] &= [Q]^T [M] [Q] \\ \{X_i\} &= [Q] \{A_i\}. \end{aligned} \quad (5.219)$$

- d) Testing the convergence of $\bar{\lambda}_i$ and if need be, repeating operations b, c and d. The method converges toward the p smallest eigenvalues, on condition that the p initial vectors $\{X_i\}$ are not M -orthogonal to one of the p eigenvectors sought. We can ensure that the p eigenvalues found are indeed the p smallest by using the property (5.195) of the Sturm series. In order to do so, we have to decompose the matrix $[K] - (\lambda_p + \varepsilon) [M]$ and check that there are p negative pivots in the decomposition (ε of around 10^{-2} to 10^{-3}).

Remarks

- The matrices $[K]$ and $[M]$ tend toward diagonal matrices, which increases the efficiency of the Jacobian method.
- If we wish to calculate p eigenvalues, it is quicker to use a subspace of dimension q greater than p , and merely verify the convergence of the p smallest eigenvalues. We can use the value $q = \text{Min}(p + 8.2 p)$ given in [BAT 76].
- In practice, the initial vectors are often chosen as follows: $\{X_1\}$ is an arbitrary vector; the other vectors are:

$$\{X_2\} = \begin{cases} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \end{cases} \text{ — row } i_1 \quad \{X_3\} = \begin{cases} 0 \\ \vdots \\ 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{cases} \text{ — row } i_2, \quad \text{etc.}$$

where i_1, i_2, \dots are the indices i corresponding to the smallest successive values of $\frac{K_{ii}}{M_{ii}}$.

Important results

Numerical integration

$$\begin{aligned} [k] &= \sum_{i=1}^r w_i [k^*(\xi_i)] \\ \{f\} &= \sum_{i=1}^r w_i \{f_V^*(\xi_i)\} \end{aligned} \quad (5.4)$$

$$\int_{-1}^1 y(\xi) d\xi = \sum_{i=1}^r w_i y(\xi_i) \quad (5.5)$$

$$\int_{-1}^1 \int_{-1}^1 y(\xi, \eta) d\xi d\eta = \sum_{i=1}^r w_i y(\xi_i, \eta_i) \quad (5.14)$$

or

$$= \sum_{i=1}^{r_1} \sum_{j=1}^{r_2} w_i w_j y(\xi_i, \eta_j) \quad (5.15)$$

$$\int_{-1}^1 \int_{-1}^1 \int_{-1}^1 y(\xi, \eta, \zeta) d\xi d\eta d\zeta = \sum_{i=1}^{r_1} \sum_{j=1}^{r_2} \sum_{k=1}^{r_3} w_i w_j w_k y(\xi_i, \eta_j, \zeta_k) \quad (5.20)$$

Resolution of systems of linear equations

$$[K] \{U_n\} = \{F\} \quad (5.22)$$

Gaussian elimination algorithm

$$[K] = [L][S] \quad (\text{decomposition}) \quad (5.28)$$

$$[K] = [L] [D] [U] \quad (5.35)$$

Decomposition algorithm for matrix stored by the skyline method

— non-symmetrical matrix

— symmetrical matrix

Resolution of systems of nonlinear equations

$$\{R^i\} = \{F\} - [K(U^{i-1})] \{U^{i-1}\} \quad (5.54)$$

— Substitution method

$$[K(U^{i-1})] \{\Delta U^i\} = \{R^i\} \quad (5.54)$$

— Modified Newton–Raphson method

$$[K_I] \{\Delta U^i\} = \{R^i\} \quad (5.61)$$

— Newton–Raphson method

$$[K_I(U^{i-1})] \{\Delta U^i\} = \{R^i\} \quad (5.66)$$

General algorithm. (Figure 5.19)

Resolution of unsteady systems

$$[C] \{\dot{U}\} + [K] \{U\} = \{F(t)\} \quad (5.86)$$

$$[M] \{\ddot{U}\} + [C] \{\dot{U}\} + [K] \{U\} = \{F(t)\} \quad (5.87)$$

— First-order systems

Explicit Eulerian method (5.101)

Implicit Eulerian method (5.105) and (5.109)

Prediction and correction methods (5.124) and (5.127)

Runge–Kutta methods (5.134)

Lax–Wendroff method (5.134c)

Modal superposition method (5.135) and (5.141)

— Second-order systems

Direct methods for second-order systems (Figure 5.23)

Modal superposition method (5.164)

Calculation of eigenvalues and eigenvectors

$$[K] \{X_i\} = \lambda_i [M] \{X_i\} \quad (5.166)$$

$$[X]^T [K] [X] = [\lambda] \quad (5.176a)$$

$$[X]^T [M] [X] = [I] \quad (5.176b)$$

Rayleigh quotient (5.186)

Inverse iteration method (Figure 5.24)

General Jacobian method (Figure 5.25)

Ritz method (5.214)

Subspace method (5.217) to (5.219)

Notations

Numerical integration

a_1, a_2, \dots, a_{2r}	coefficients of a polynomial
$[B], [B_\delta]$	matrices linking the gradients to the nodal variables, and the variations of these values
C_r	constant in the expression of errors
dV^e, dV'	differential volume of the real element and the reference element
$[H]$	matrix of physical properties
e	error
f, f_s, f_V	strains
$\{f\}, \{f_s\}, \{f_V\}$	vectors of elementary strains
$\{f_s^*\}, \{f_V^*\}$	vectors to be integrated to construct $\{f_s\}$ and $\{f_V\}$
$[J]$	Jacobian matrix
$[k]$	elementary matrix
$[k^*]$	matrix to be integrated to construct $[k]$
$[m]$	elementary mass matrix
$[N], \langle N \rangle$	matrix and vector made up of interpolation functions
P_r	Legendre polynomial
r, r_1, r_2, r_3	numbers of integration points and order of polynomials
w_i	weighting coefficient of numerical integration
$\gamma(\xi)$	function to be integrated numerically
$\xi = \langle \xi \quad \eta \quad \zeta \rangle$	coordinates of a point in the reference element
ξ_i	coordinates of a point of numerical integration

Resolution of systems of linear equations

$[D]$	diagonal matrix of the decomposition
$\{F\}$	global strain vector
F_i	component i of $\{F\}$
$\{F'\}$	vector $\{F\}$ after Gaussian elimination
$\{F^s\}$	vector $\{F\}$ after elimination of the first s variables
h_J	height of the column J
i_{0i}, i_{0s}	row numbers of the upper term in columns i and s
$[K]$	global matrix
$[K^s]$	global matrix after elimination of the first s variables
$[l]$	lower triangular matrix equal to $[L]^{-1}$
$[l^i]$	lower triangular matrix corresponding to the elimination of the variable i

$[L]$	lower triangular matrix of the decomposition
$[L^i]$	inverse of $[l^i]$
$[L_c]$	matrix of Choleski decomposition
$[S]$	upper triangular matrix of the decomposition
$\{U_n\}$	global variables
$[U]$	upper triangle of the decomposition $[L][D][U]$

Resolution of systems of nonlinear equations

$\{F\}$	global load vector
$\{F_0\}$	global reference load vector
$\{\Delta F\}$	increase in $\{F\}$
$[K]$	global matrix
$[K_c]$	matrix used to calculate $\{\Delta U\}$
$[K_l]$	global matrix, linear part
$[K_{nl}]$	global matrix, nonlinear part
$[K_t]$	tangent global matrix
$\ m\ , \ n\ $	norm of $\{R\}$ and $\{\Delta U\}$
$\{R\}$	residual
$\{R^i\}$	residual corresponding to $\{U^{i-1}\}$
$\{U\}, \mathbf{U}, \{U_n\}$	global vector of nodal variables
$\{U^i\}$	vector $\{U\}$ at iteration i
$\{\Delta U\}, \{\delta U_n\}$	increment and variation of $\{U\}$
$W, W^e, \delta W$	global and elementary integral form, first variation of W
δ	variation symbol
ε	admissible error on a norm
$\lambda, \Delta\lambda$	load vector parameter, corresponding increment
ω	over-relaxation coefficient
Π	functional

Resolution of unsteady systems

a, b	coefficients of the Newmark-Wilson method
a_0, a_1, a_2	coefficients involved in the direct methods for second-order systems
$[A], [B]$	matrices used in the stability calculations
b_j	coefficients of the prediction-and-correction and Runge-Kutta methods
$[C]$	damping matrix
f	right-hand side of the standard expression $\frac{du}{dt} = f(u, t)$

$\{F\}, \{\underline{F}_t\}$	global load vector at time t
$[K], [\bar{K}], [K_{nl}], [K_t]$	global matrix, modified matrix, nonlinear part of $[K]$, tangent matrix
l_{\max}	largest eigenvalue (in absolute value)
$[M]$	global mass matrix
$\{R\}, \{R_t\}, \{\bar{R}\}, \{R_{nl}\}$	residuals
$t, t_0, \Delta t$	time, initial time, increment in time
$\{U\}, \{U_0\}, \{U_t^i\}$	nodal variables, values at initial time, values at time t after iteration i
$\{\dot{U}\}, \{\ddot{U}\}$	first and second derivatives of U in relation to time
$\{V\}$	solution to the uncoupled equations
$\{X_i\}, [X]$	eigenvectors and modal matrix
$\lambda, \lambda_i, [\lambda]$	eigenvalues
ξ_i	damping factor of mode i
ω_i	frequency corresponding to the eigenvalue λ_i
$\rho(A)$	spectral radius of the matrix A
τ, θ	parameter of the Wilson method

Calculation of eigenvalues and eigenvectors

a	amplitude of a shift
a, b, c_1, c_2, c_3, d	coefficients of the Jacobian method
$a_1, a_2, \dots, a_p, \{a\}$	coefficients of the Ritz vectors
$\{A_i\}$	eigenvectors of the Ritz method
c_i	components of a vector in the base of eigenvectors
$[K], [\bar{K}]$	global matrix, modified matrix
$[K^i]$	matrix $[K]$ modified by i Jacobian iterations
l_i	eigenvalues
$[L]$	lower triangular matrix of the decomposition of $[K]$
$[M], [\bar{M}]$	global mass matrix, modified matrix
$[M^i]$	matrix $[M]$ after iteration i of the Jacobian method
p	exponent of $[K]$, subscript
$P(\lambda)$	characteristic polynomial of a matrix
$[P_i]$	projector of a matrix
$\{q_i\}$	Ritz vectors
$[Q]$	transformation matrix, matrix of Ritz vectors
$[Q^i]$	transformation matrix of iteration i of the Jacobian method
$R(V)$	Rayleigh quotient corresponding to the vector V
V	any vector
$\{X_i\}, [X] \quad [\bar{X}]$	eigenvectors

$\{\bar{Y}\}$	vectors $[M]\{X\}$
λ, λ_i	eigenvalues
μ	shift
$\rho(K)$	spectral radius of $[K]$
θ	Jacobian rotation.

Bibliography

- [BAT 76] BATHE J., WILSON E.L., *Numerical Methods in Finite Element Analysis*, Prentice Hall, 1976.
- [BAT 79] BATOZ J.L., DHATT G.S., “Incremental displacement algorithms for non-linear problems”, *International Journal for Numerical Methods in Engineering*, vol. 14, no. 8, pp. 1262–1267, 1979.
- [CAR 69] CARNAHAN B., HUNTER H.A., WILES O., *Applied Numerical Methods*, Wiley, 1969.
- [CLO 75] CLOUGH R.W., PENZIEN J., *Dynamics of Structures*, McGraw-Hill, 1975.
- [COL 66] COLLATZ L., *The Numerical Treatment of Differential Equations*, Springer-Verlag, 1966.
- [COO 04] COOLS R., “An encyclopaedia of cubature formulas”, *Journal of Complexity*, vol. 9, no. 3, pp. 445–453, 2004.
- [CRA 56] CRANDALL S.H., *Engineering Analysis*, McGraw-Hill, 1956.
- [DAV 75] DAVIS J.J., RABINOWITZ P., *Methods of Numerical Integration*, Academic Press, 1975.
- [DUN 85] DUNAVANT D.A., “High degree efficient symmetrical Gaussian Quadrature Rules for the triangle”, *International Journal for Numerical Methods in Engineering*, vol. 21, pp. 1129–1148, 1985.
- [FEL 00] FELIPPA C.A., “Recent advances in finite element templates”, Chapter 4, *Computational Mechanics for the Twenty-First Century*, B.H.V. Topping, Saxe-Coburn Publications, Edinburgh, pp. 71–98, 2000.
- [FEL 04] FELIPPA C.A., *A Compendium of FEM Integration Rules for CAS Work*, University of Colorado, 2004.
- [GEL 91] GELLERT M., HARBORD R., “Moderate degree cubature formulas for 3-D tetrahedral finite-element approximation”, *Communications in Applied Numerical Methods*, vol. 7, pp. 487–495, 1991.
- [HAD 96] HADJI S., DHATT G., “Méthodes itératives pour les fluides incompressibles”, *Revue Européenne des Eléments Finis*, vol. 6, nos. 5/6, pp. 513–544, 1997.
- [HAM 56] HAMMER P.C., MARLOW O.P., STROUD A.H., “Numerical integration over simplexes and cones”, *Mathematical Tables and Other Aids to Computation*, vol. 10, pp. 130–137, 1956.
- [HEL 72] HELLEN T.K., “Effective quadrature rules for quadratic solid isoparametric finite elements”, *International Journal for Numerical Methods in Engineering*, vol. 4, pp. 597–600, 1972.
- [HOU 50] HOUBOLT J.C., “A recurrence matrix solution for the dynamic response of elastic aircraft”, *Journal of the Aeronautical Sciences*, vol. 17, pp. 540–550, 1950.

- [IRO 66] IRONS B.M., “Engineering application of numerical integration in stiffness method”, *The American Institute of Aeronautics and Astronautics (AIAA) Journal*, vol. 14, pp. 2035–2037, 1966.
- [IRO 71] IRONS B.M., “Quadrature rules for brick based finite elements”, *The American Institute of Aeronautics and Astronautics (AIAA) Journal*, vol. 9, pp. 293–294, 1971.
- [KAT 77] KATONA G., THOMPSON R., SMITH J., Efficiency study of implicit and explicit time integration operators for finite element applications, Report No. R 856, Naval Construction Battalion Center, Port Hueneme, Cal., 1977.
- [KOP 61] KOPAL Z., *Numerical Analysis*, 2nd ed., Chapman and Hall, 1961.
- [KRE 88] KREYSZIG E., *Advanced Engineering Mathematics*, Wiley, 1988.
- [LAX 60] LAX P.D., WENDROFF B., “Systems of conservation laws” *Communications on Pure and Applied Mathematics*, vol. 13, pp. 217–237, 1960.
- [NEW 59] NEWMARK N.M., “A method of computation of structural dynamics”, *ASCE Journal of Engineering Mechanics, Division*, vol. 85, pp. 67–94, 1959.
- [PVM 94] BEGUELIN A., GEIST A., et al., *PVM 3 User’s Guide and Reference Manual*, Oak Ridge National Laboratory, May 1994.
- [RAL 65] RALSTON A., *A First Course in Numerical Analysis*, McGraw-Hill, 1965.
- [SAA 03] SAAD Y., *Iterative Methods for Sparse Linear Systems*, 2nd ed., Ed. SIAM, 2003.
- [STR 71] STROUD A.J., *Approximative Calculation of Multiple Integrals*, Prentice Hall, 1971.
- [STR 73] STRANG G., FIX G., *An Analysis of Finite Element Method*, Prentice Hall, 1973.
- [STR 86] STRANG G., *Introduction to Applied Mathematics*, Wellesley-Cambridge Press, 1986.
- [WIL 77] WILSON E.L., CAL computer analysis language, Report No. UC SESM 77-2, Department of Civil Engineering, University of California, Berkeley, 1977.
- [WIL 65] WILKINSON J.H., *The Algebraic Eigenvalue Problem*, Oxford University Press, 1965.
- [ZIE 00] ZIENKIEWICZ O.C., *The Finite Element Method: The Basis (Vol. 1), Solid Mechanics (Vol. 2) & Fluid Mechanics (Vol. 3)*, 5th ed., Butterworth Heinemann, 2000.

CHAPTER 6

Programming technique

6.0 Introduction

In the previous chapters, we have presented various aspects of the finite element method:

- geometric representation of a domain and finite element approximation (Chapters 1 and 2);
- mathematical modeling of the behavior of a physical system in the form of weak expressions, also called integral or variational functionals or forms (Chapter 3);
- construction of a discrete or algebraic model associated with a weak expression and based on a finite element discretization (Chapter 4);
- presentation of the various numerical methods used to assemble and solve the global algebraic system representing the behavior of a physical system – be it stationary or transient, linear or nonlinear (Chapter 5).

The computer implementation of the method is composed of different functional blocks:

- geometric and physical description of the problem in question;
- calculation of the elementary matrices and vectors and assembly of the global system;
- solution of the system;
- visualization of the results and post-processing.

In the computer market today, there are a certain number of finite element codes, which can simulate the behavior of a variety of physical systems, such as deformable solids, compressible or incompressible fluid mechanics, thermal engineering, electromagnetism, etc. Of these programs, one might cite, in particular, NASTRAN; ANSYS; ABAQUS; NSDYN; FIDAP; FLUENT; IDEAS; SAMCEF; CASTEM; CASTOR; etc.

The actual use of a modeling software involves a certain amount of *savoir faire*, and expertise that lends itself to the problem in question: a good knowledge of the physical principle(s) involved and a general understanding of the finite element method. The large amount of data to be analyzed at the post-processing stage depends on the complexity of the problem at issue, and only a solid understanding of the physical phenomena at play will enable the user to isolate the essential aspects from the mass of output information furnished by the code.

In this chapter, we present a number of aspects of the practical use of the finite element method, and in particular, the basic programming techniques required for its implementation in software form. Finally, we describe a general finite element method program written in Matlab[©] (a scientific calculation programming language, marketed by Mathworks), which serves both to introduce the user to finite elements and to define a base from which it will be possible to expand the domains of application and the complexity of the problems dealt with.

6.1 Functional blocks of a finite element program

Any program based on the finite element method includes a certain number of characteristic functional blocks, intended to:

- a) read, verify and organize the data describing the mesh (nodes and elements), the physical parameters (conductivities, elasticity moduli, etc.), loadings and boundary conditions;
- b) construct the elementary matrices and vectors, assemble the global matrix and the global load vector;
- c) solve the system of equations having taken account of the boundary conditions and the initial conditions in the case of a transient problem;
- d) visualize the results and possibly calculating additional variables (fluxes, constraints, reactions, etc.).

Figure 6.1 shows the logical flow of these different blocks for a linear, stationary calculation.

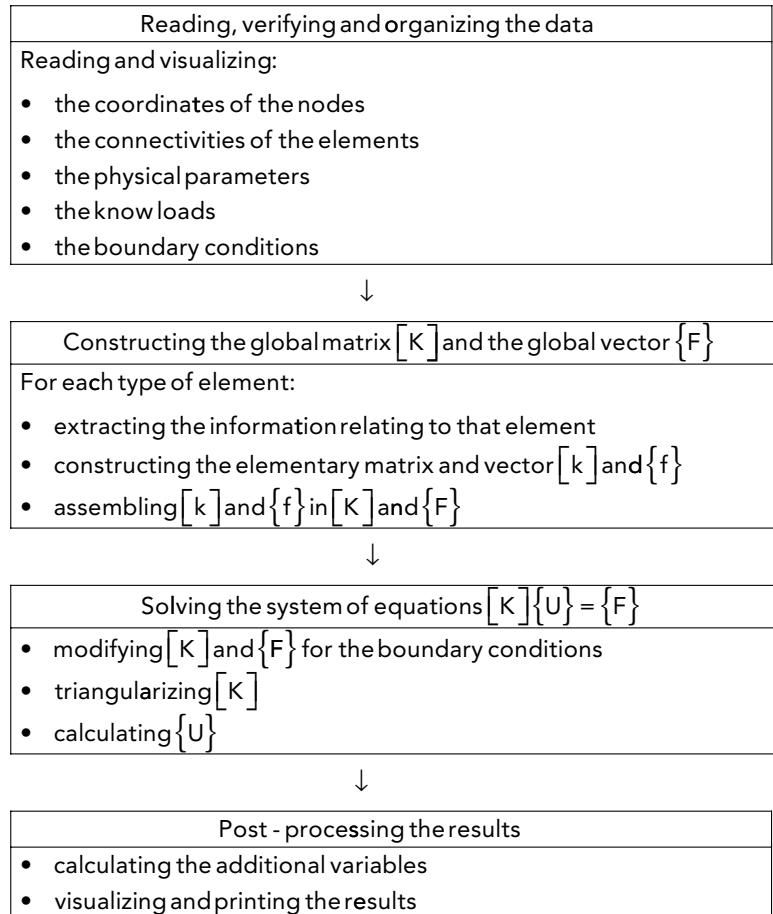


Figure 6.1. Characteristic functional blocks of a finite element program for a linear, stationary calculation

6.2 Description of a typical problem

The first stage of studying a physical system using a finite element code is an accurate description of the problem being studied, at the geometric level and at the level of the physical properties, the known applied forces and the boundary conditions. Thus, we can identify the different stages:

- Nature of the physical behavior
 - structural mechanics: linear or otherwise (physical or geometric), contact, plasticity, etc.
 - fluid mechanics: laminar or turbulent flows, etc.

- thermal engineering: conduction, free or forced convection, radiation, etc.
- and so on.
- Time-related aspect: is this behavior stationary, quasi-stationary or transient? If the problem is transient, we have to choose one or more initial conditions.
- Geometric characteristics: if the geometry of the problem is three-dimensional, would it be possible to transpose the study to a two-dimensional – or even one-dimensional – problem? Would certain symmetries help to simplify the domain being studied? In structural mechanics, these criteria help determine the choice of the type of element to be used: beam, plate, shell, volume, etc.
- Physical properties (conductivity, Young's modulus, Poisson's coefficient, etc.), homogeneous, isotropic materials or otherwise, etc.
- Boundary conditions: identifying the different types of boundary conditions along the edge of the domain of calculation: Dirichlet, Neumann, Cauchy, etc.
- Loads: identifying the nature of the surfacic and/or volumic forces.

In an industrial context, engineers find themselves collaborating with different specialists (designers, experimenters, manufacturers, etc.) to create as precise a description of the problem as possible.

To ensure the quality of the obtained solution, an engineer will be led to reexamine his or her initial choices, such as:

- the finite elements chosen: T3, Q4, etc.;
- the fineness of the mesh with the possibility to adapt the mesh to the gradient fields (constraints) provided by the calculation;
- the choice of the time scheme that influences the stability of the solution: explicit for the study of a rapid transitory problem or implicit when searching for a quasi-stationary solution;
- the algorithm for solving the nonlinear model;
- and so on.

6.3 General programs

6.3.1 POSSIBILITIES OF GENERAL PROGRAMS

A general program must be capable of solving:

- varied problems that arise from various domains: linear or nonlinear elasticity, compressible or incompressible fluid, thermics, harmonic problems, etc.;
- large-scale problems involving large numbers of nodes and elements.

In addition, it must include a library of finite elements and important numerical methods.

Depending on the problem being studied, the number and nature of the nodal variables change, as do the formulations of the elementary matrices and vectors. In addition, for a given problem, we can use several differently shaped types of elements (e.g. triangles and quadrilaterals), and with varying degrees of precision (e.g. a triangle with three or six nodes). Finally, in a given domain of application, it is desirable to be able to deal with one-, two- and three-dimensional problems, be they linear or nonlinear, and stationary or otherwise.

In summary, to deal with the varied problems, a general program must include the following possibilities:

- use of the data provided by computer-aided design (CAD) tools, automatic generation and adaption of meshes;
- one-, two- and three-dimensional problems;
- differing numbers of degrees of freedom (DoF) at each node;
- library of finite elements that can be easily added to;
- elementary and global matrices that are symmetrical or otherwise, linear or nonlinear problems, stationary or transient problems;
- eigenvalue problems;
- post-processing tools for visualizing the results, with the possibility of animation for transient problems.

Numerous industrial problems require the use of a great many elements, nodes and, therefore, degrees of freedom. The total number of unknowns can range from a few hundred (small problems) to several hundred thousand (medium-sized problems), or even several million. The size of the problem depends on the following factors:

- the number of dimensions;
- the number of unknown variables at each node – e.g. the components of the velocity and pressure fields;
- the complexity of the geometry being studied;
- the fineness of the mesh that allows us to determine the solution to a sufficient degree of precision.

For large problems, a number of difficulties present themselves.

a) Description of the problem

Description of the problem includes preparation of the tables of coordinates ('vcor' table) and connectivity ('kconec' table), definition of the physical properties ('vprel' table), loads and boundary conditions ('kcond' and 'vcond' tables). This description may become voluminous and give rise to errors.

Thus, a general program must have tools to assist the preparation and verification of the data: subprograms to automatically generate and trace meshes. Increasingly, these tools constitute interactive preprocessors that are independent of the general program, which often include the preparation of the load vectors, the tables of physical properties and boundary conditions. General CAD programs tend to offer these possibilities.

b) Volume of calculations

For a large problem, the computation time needed to obtain the solution becomes very great – particularly for nonlinear and transient problems. It is important to efficiently program the phases of calculation that are frequently repeated: construction of the elementary matrices, assembly and resolution. In addition, we have to make the correct **choices for all the parameters** that affect the computation time:

- types of elements and numbering of the nodes in the mesh;
- method of numerical integration;
- method of solving the system of equations, particularly for nonlinear systems;
- method of calculating the eigenvalues.

c) Distributed computing techniques

Distributed computing techniques (section 5.2.4.2) are mainly based on a technique whereby the initial domain is broken down into N subdomains of limited sizes (i.e. N sub-meshes). The aim of these techniques is to carry out a maximum number of tasks considered to be 'distributable' simultaneously on N processors. Of these tasks, one might cite:

- the phase of calculation of the elementary matrices and assembly of the partial systems associated with each of the subdomains;
- the phase of partial solution of the system associated with the subdomain: elimination of the unknowns inside the subdomain;
- the phase of redistribution of the solution calculated on the interface shared between all the subdomains (this phase is normally performed on a dedicated processor).

These techniques require the use of appropriate environments, facilitating the rapid exchange of data between the subdomains. They are essentially intended to:

- improve the performance of a computation tool by significantly reducing the computation time in comparison to a calculation carried out on a single processor;
- render **feasible** calculations that require greater memory capacity than that of a single processor.

d) Exploitation of the results

Such programs output the results in the form of vectors and voluminous tables, which are difficult to exploit. Thus, a general tool must have tools for selective representation and three-dimensional visualization of the results with the possibility of animations for transient problems.

These tools may constitute post-processors that are independent of the general finite element program, and must be able to be adapted to the specific needs of each user (GiD, Tecplot, Vigie, etc.).

6.3.2 MODULARITY

A general program, which presents the characteristics described in section 6.3.1, is necessarily voluminous and complex. However, it is preferable that:

- Its logic should be easy to understand.
- It should be easy to modify.
- Several people should be able to work together on its development without each of them having to be minutely familiar with the entirety of the program.
- We should be able to specialize or optimize the program for a given type of application by simply replacing certain subprograms.

To attain these objectives, we have to structure the program in a **modular** manner:

a) Organization of data

- creation of the tables of coordinates and connectivities;
- creation of the tables containing known parameters linked to elements or to nodes (elementary properties and nodal properties);
- creation of the tables defining the boundary conditions.

b) Operations at the elementary level

- determination of the coordinates and weights of the integration points;
- calculation of the interpolation functions and their derivatives;
- calculation of the Jacobian matrix, its inverse and its determinant;
- construction of each elementary matrix and vector:
 $[k], [m], \{f\}, [k_t], \{r\}$, etc.

c) Operations of assembly

- assembly of an elementary vector or matrix $[k], [m], \{f\}, [k_t], \{r\}$ into a global vector or matrix $[K], [M], \{F\}, [K_t], \{R\}$.

d) Solution

- iterative linearization process in the case of a system of nonlinear equations;
- decomposition and resolution of a system of linear equations.

e) Visualization of the results

- visualization of the nodal variables and the various additional results: gradients, reactions, etc.

6.4 Description of the finite element code

6.4.1 INTRODUCTION

In this section, we present a simple finite element program, written in the interactive language Matlab[©]. This program exhibits the modular structure described in section 6.3.

There are many general programs of excellent quality that enable us to solve a set of physical problems commonly found in an industrial setting. The objective here is not a competitive comparison between these software packages, which include a great deal of knowledge accumulated since the 1950s.

The objectives of the finite element code presented here are:

- to help the user properly understand the stages of the implementation of the finite element method;
- to offer a simple structure with programming norms that enable us to quickly construct a program to deal with specific problems.

Thus, we propose to have three functional blocks:

- ‘blin’: linear stationary problems;
- ‘btemp’: temporary linear problems;
- ‘bnlin’: nonlinear problems, stationary or otherwise.

Each of these blocks is a main program that calls a certain number of subprograms written in the form of functions in Matlab[©]. The ensemble of these subprograms constitutes an extendable library of finite elements dedicated to problems in solid mechanics, fluid mechanics, heat transfer, etc. Depending on the user’s needs, it is then possible to quickly develop a new functional block which would only include those subprograms that are appropriate for the intended application.

6.4.2 GENERAL ORGANIZATION

6.4.2.1 Functional blocks

A functional block is a main program that strings together the operations and Matlab[©] functions. In this section, we present three functional blocks that, respectively, solve a stationary linear problem, a linear temporary problem and a nonlinear problem (stationary or otherwise).

The execution of a functional block invariably begins with the reading of a file of data, describing the problem at hand. This operation is performed by the Matlab[©] command:

```
feval(fdata)
```

The variable ‘fdata’ is the name of the file containing the data of the problem. For instance, to deal with the problem of the Poiseuille flow described in section 6.6.3-b, the user need only type the following under the call function (‘>>’) in Matlab[©]:

```
>> fdata='dpoiseuille';
```

The nomenclature of the tables and variables used in the finite element method is brought together in section 6.4.3.

Note that in the following section, description of codes written in Matlab[©] are presented with explanatory comments – **this code description is not intended to be executed**. For the executable code in section 6.5 and later sections, the comments are preceded by a % symbol.

a) Description of functional block for solving a stationary linear system: blin

The algorithm is as follows:

- Reading of the data: feval (fdata)
- Initialization of the matrices **vkg** and **vfg** at zero

% Construction of **vkg** and **vfg**

Loop on the elements ie

- feval('nom_type_elem'_ke,ie): calculation of the values **vke** and **vfe**
- Assembly: **vke** into **vkg** and **vfe** into **vfg**

End of loop on the elements

- Introduction of the boundary conditions into **vkg** and **vfg**

- Solution: **vsol=vkg\vfg**

- Display of the solution as values or color fields

% Calculation of the elementary gradients

Loop on the elements ie

- feval('nom_type_elem'_gr,ie): calculation of the elementary gradient
- Print

End of loop on the elements

b) Functional block for solving a transient linear system: btemp

The algorithm of btemp is written:

- Reading of the data: feval (fdata)
- Initialization of the matrices **vmg**, **vkg**, **vfg** and **vres** at zero

% Construction of **vmg**, **vkg** and **vfg**

Loop on the elements ie

- feval('nom_type_elem'_ke,ie): calculation of the values **vke** and **vfe**
- Assembly: **vke** into **vkg** and **vfe** into **vfg**
- feval('nom_type_elem'_me,ie): calculation of **vme**
- Assembly: **vme** into **vmg**

End of loop on the elements

- Choice of type of scheme: implicit (alpha=1), explicit (alpha=0) ...
- Calculation of: **vmg=vmg+dt*alfa*vkg**
- Introduction of the boundary conditions into **vkg** and **vfg**

% Temporal integration

Loop on the step ipas

- Calculation of the residual: **vres=vfg-vkg*vsol**
- Solution: **vdu=vmg\dt*vres**
- Updating of the solution: **vsol=vsol+vdu**

New step

- Display of the results

c) Functional block for solving a transient nonlinear system: bnlm

The algorithm of bnlm is written:

- Reading of the data: feval (fdata)
- Initialization of the matrices **vkg** and **vres** at zero

% Loop for temporal resolution or iterative steps for stationary case

*** For each step ipas

- Updating: **vsol** or stress or C.L.

** For each iteration: iter

% Construct the tangential matrix **vkg** and the residual **vres**

*Loop on the elements ie

- feval('nom_type_elem'_kt,ie) : calculation of **vke** and **vfe**
- Assembly: **dt*vke** into **vkg** and **dt*vfe** into **vres**
- Calculation of the mass matrix if density > 0

- feval('nom_type_elem'_me,ie) : calculation of **vme**
- Assembly: **vme** into **vkg** and modification of **vres**
- * End of loop on the elements
- Introduction of the boundary conditions into **vkg** and **vres**
- Solution: **vdu** = **vkg****vres**
- Updating: **vsol**=**vsol**+**vdu**
- Stop criterion for the iterative cycle
- ** New iteration
- *** New step

6.4.2.2 Programming norms

When a program has to be developed and modified by several people, programming norms must be defined. We systematically use lower case characters to indicate functions, tables and variables. The norms adopted for writing the finite element method program are as follows:

a) Functional blocks

The name of each functional block is either four or five letters long:

- blin, btemp, bnlm.

b) Names of functions for calculating the elementary quantities

Each function has a name comprising:

- two characters denoting the type of problem: thermal ('th'), elasticity ('el'), fluid ('ns');
- followed by two or three characters indicating the geometry of the element: 'q4', 't3', 'l2' and 'l2b' (for boundary);
- and finally two characters specifying the quantities calculated: 'ke', 'kt' or 'gr' (gradients).

The names chosen for each application are (here the variable ie denotes the number of element being processed):

Heat transfer problem (q4, t3 with l2 boundary):

- [vke,vfe] = th_q4_ke(ie): calculation of $[k]$ and $\{f\}$ with Q4.

- [vke,vfe] = th_t3_ke(ie) : calculation of $[k]$ and $\{f\}$ with T3.
- vme = th_t3_me(ie) : calculation of $[m]$ with T3 (also for Q4).
- vsig = th_t3_gr(ie) : calculation of the flux with T3 (also for Q4).
- [vke vfe] = th_l2b_ke(ie) : calculation of $[k]$ and $\{f\}$, element with boundary L2.

Elasticity problem (q4, t3 with l2 boundary):

- [vke,vfe] = el_q4_ke(ie) : calculation of $[k]$ and $\{f\}$ with Q4.
- [vke,vfe] = el_t3_ke(ie) : calculation of $[k]$ and $\{f\}$ with T3.
- vsig = el_t3_gr(ie) : calculation of the loads with T3 (also for Q4).
- [vke vfe] = el_l2b_ke(ie) : calculation of $[k]$ and $\{f\}$, element with boundary L2.

Navier-Stokes problem (q4, with l2 boundary):

- [vke,vfe] = ns_q4_kt(ie) : calculation of $[k_t]$ and $\{r\}$ with Q4.
- vme = ns_q4_me(ie) : calculation of $[m]$ with Q4.
- [vke vfe] = ns_l2b_kt(ie) : calculation of $[k_t]$ and $\{f\}$, element with boundary L2.

c) Tables

A table has a name comprising at most six lower case letters:

- v..... for a table made up of real numbers (e.g. vcor, vkg...).
- k..... for a table made up of integers (e.g. kcone, kloce,...).

6.4.3 DESCRIPTION OF TABLES AND VARIABLES

a) General nomenclature

In the following, we present an exhaustive list of variables used when writing the various subprograms that make up the finite element method program.

% Alphanumeric variable

fdata : name of the Matlab[©] file (without the file extension) containing the geometric data (finite elements) and physical data (materials, boundary conditions, stresses) of the problem.

%---- Parameters

nnt : total number of nodes;
 ndlt : total number of degrees of freedom (=ndln*nnt)
 nelt : total number of elements
 nnel : number of nodes per element
 ndln : number of degrees of freedom per node
 nprel : number of properties per element;
 ndle : number of degrees of freedom per element (=nnel*ndln)
 ntypel : number of different types of elements
 nprop : number of different groups of elementary properties
 ndim : dimension of the geometry (1, 2 or 3)

%---- Mesh and properties

nomtype : alphanumeric table listing the names of the different types of elements used to solve the problem
 vcor(nnt,ndim) : coordinates table
 kcone(c(nelt,nnel)) : connectivity table

ktypel(1,nelt) : table describing the type of each element

The maximum number of types of elements is ‘ntypel’. This number is used to choose the name of the corresponding function in ‘nomtype’.

vprelg(nprop,npref) : table of elementary properties
 kprop (1,nelt) : table of types of elementary properties

The maximum number is ‘nprop’. It is used to extract the properties in the vprelg table.

kcond(1,ndlt) : table of indices of the Dirichlet condition imposed (=1 imposed, 0 if not)

```

vcond(1,ndlt)           : table of Dirichlet values imposed

% -----Elementary and global matrices and vectors
vke(ndle,ndle)          : elementary stiffness matrix
vme(ndle,ndle)          : elementary mass matrix
vfe(ndle,1)              : elementary load vector
vdle(ndle,1)             : elementary solution vector
kloce(1,ndle)            : elementary localization table for assembly
vkg(ndlt,ndlt)           : global stiffness matrix
vmg(ndlt,ndlt)           : global mass matrix
vfg(ndlt,1)              : global load vector
vfcg(ndlt,1)             : concentrated load vector
vsol(ndlt,1)              : global solution vector
vsol0(ndlt,1)             : initial global solution vector
vres(ndlt,1)              : global residual vector

% -----Parameters for temporal and nonlinear calculation
npas                      : number of steps
dt                         : time-step
niter                     : maximum number of iterations to perform
alfa                       : parameter of scheme – explicit (=0), implicit (=1)
npilot                    : parameter of piloting of the solution by stresses or by
                           boundary conditions

```

b) Global variables

Globally declaring certain variables that are common to all the subprograms makes it easier and less cumbersome to write the programs.

```

% ----- Global declarations
%----- parameters
      global nnt nelt ndlt ndln nnel ndle
      global ntypel nprop nprel
%----- mesh and allocation

```

```

        global vcor kcone c ktypel nomtype
%----- properties
        global vprelg vprel kprop
%----- solution
        global vsol vdle kloce
%----- time / nonlinear
        global vsol0 dt npas alfa npilot niter aval

```

c) File describing the data of the problem

The data associated with a problem are contained in a file called ‘fdata’ and initialized by the user of the program by way of the Matlab[©] command:

```
>> fdata='name of data file';
```

where ‘>>’ represents the invitation (or prompt). The data file is based on the following general organization:

```

% Reading of the types of elements used
% For instance, for two types of thermal finite elements:
    nomtype = ['th_q4'; 'th_l2b'];
% Important: If need be, supplement with one or more spaces so that the
% names occupy the same number of columns.
% Reading of the coordinates table vcor(nnt, ndim)
    vcor = [ x1 y1
              ...
              xn yn];
% Reading of the connectivity table kcone(c,nelt,nnel)
    kcone(c) = [n1 n2 n3 n4
                ...
                ...];
% Reading of the parameters:
    ndln=1; nnel=size(kcone,2); ndle=4 ; nelt= size(kcone,1);
    nnt=size(vcor,1); ndlt=nnt*ndln ; ndim=2;
% Reading of the table defining the type of elements in the mesh: ktypel(1,nelt)
%      1: first name in nomtype, 2: second name in nomtype...
    ktypel = [1 1 2 2 ...];

```

```
% Reading of the table of types of properties: kprop(1,nelt)
% It is possible to have a single type of element with different types of properties
% or different types of elements with a single type of property.
%      1: element associated with the first group of properties,
%      2: element associated with the second group of properties ...
kprop=[1 1 1 2 ...];
% Reading of the table of elementary properties: kprelg(nprop, nprel)
% The properties are, e.g. the density, conductivity, thickness,
% a volumic forces for each type comprising 'nprop' properties.
% Example: the elementary program th_q4_ke uses the following properties:
vprelg= [ ro modulus thickness f; % first group of properties
           ...           ]; % second group of properties
% Reading of the boundary conditions: kcond(1,ndlt),vcond(1,ndlt)
kcond=[ 0 if free and 1 if imposed ...];
vcond=[ read only the non-zero values imposed]
% Reading of vfcg(ndlt,1) if necessary
% Reading of the parameters for 'btemp' or 'bnlin'
dt = ... ; npas = ... ; alfa = ... ; niter = ... ;
```

See sections 6.6.1 – 6.6.3 for detailed examples of data files.

6.5 Library of elementary finite element method programs

In this section, we give the ensemble of subprograms that make up the finite element code. All these programs, written in Matlab[®], have the filename extension '.m'.

6.5.1 FUNCTIONAL BLOCKS

6.5.1.1 Block for 'blin.m'

```
%---- parameters
global nnt nelt ndlt ndln nnel ndle
global ntypel nprop nprel
%---- mesh and allocation
```

```

global vcor kconecktypel
%---- properties
global vprelg vprelkprop
%---- solution
global vsol vdle
%-----
time0=cputime % initialize CPU time
%---- reading of the data fdata='nom_du_fichier'
feval(fdata)
%---- tracing of the mesh
close all
trace_mail(vcor,kconecktypel)
%---- finite element calculation: vke vfe and assembly
%---- initialization
disp(' ===== blin MODULE, finite element method version 1.0 =====');
disp(' ');
vkg=zeros(ndlt); % global stiffness matrix
vfg=zeros(ndlt,1); % global load vector
vfg=vfcg; % updating by concentrated stress
%---- loop on the elements
for ie=1:nelt
    fprintf('---- élément ie = %5i\n',ie)
    %---- localization vector kloce for assembly
    kloce=[];
    for ii=1:nne
        if kconecktypel(ie,ii) > 0
            kloce=[kloce,(kconecktypel(ie,ii)-1)*ndln+[1:ndln] ];end
        end;
        %---- calculation of vke and vfe: example t3_th_ke
        vprelg=vprelg(kprop(ie,:)); % elementary properties
        [vke,vfe]=feval([deblank(nomtype(ktypel(ie,:)),'_ke'),ie]);
        %---- assembly of vke
        vkg(kloce,kloce)=vkg(kloce,kloce) + vke;
        %---- assembly of vfe
        vfg(kloce)=vfg(kloce) + vfe;
    end
    %---- Introduction of the Dirichlet boundary conditions-----
    % Technique demonstrated in section 4.7.2-b: unit term on the diagonal
    for ic = 1 :ndlt
        if(kcond(ic) == 1) % Dirichlet or kinematic
            vfg=vfg - vkg(:,ic)* vcond(ic) ;
    end

```

```

vkg(ic,:)=zeros(1,ndlt);
vkg(:,ic)=zeros(ndlt,1);
vkg(ic,ic)=1; vfg(ic)= vcond(ic);
end
end
%----- solution-----
vsol=vkg\ vfg;
disp(' '); disp(' ----- solution vsol -----');
fprintf(' %12.5e%12.5e%12.5e%12.5e%12.5e\n',vsol)
%----- calculation of the gradients-----
disp(' '); disp(' ----- Flux at Gaussian points -----')
disp(' elem sigx sigy sigxy')
%---- loop on the elements
for ie=1:nelt
    %---- localization vector kloce for assembly
    kloce=[];
    for ii=1:nnel
        if kconec(ie,ii) > 0
            kloce=[kloce,(kconec(ie,ii)-1)*ndln+1:ndln];
        end;
        vdle=vsol(kloce);           % elementary solution
        vprelg=vprelg(kprop(ie),:); % elementary properties
        vsig=feval([deblank(nomtype(ktypel(ie),:)), '_gr'],ie);
    end
    fprintf(1,'Time CPU=%12.6f secondes\n',0)

```

6.5.1.2 Block for 'btemp.m'

```

%---- parameters
global nnt nelt ndlt ndln nnel ndle
global ntypel nprop nprel
%---- mesh and allocation
global vcor kconec ktypel
%---- properties
global vprelg vprel kprop
%---- solution
global vsol vdle
%---- time
global vsol0 dt npas alfa
%-----
time0=cputime          % initialize CPU time

```

```
%----- reading of the data fdata='nom_du_fichier'
feval(fdata)
%----- tracing of the mesh
close all
trace_mail(vcor,kconec)
%----- initialization
disp(' ===== btemp MODULE, finite element method version 1.0 =====');
disp(' ');
vkg=zeros(ndlt); % global stiffness matrix
vmkg=zeros(ndlt); % matrice m+dt*alfa*k
vfg=zeros(ndlt,1); % global load vector
vres=zeros(ndlt,1); % residual vector F-K*U
vfg=vfcg; % updating by concentrated stress
%-----
fprintf(1,' dt= %12.6f npas=% 5i alfa= %12.6f\n',dt,npas,alfa)
%----- construction of vmg and vkg-----
%----- loop on the elements
for ie=1:nelt
    fprintf('---- element ie = %5i\n',ie)
    %----- localization vector kloce for assembly
    kloce=[];
    for ii=1:nnei
        if kconec(ie,ii) > 0
            kloce=[kloce,(kconec(ie,ii)-1)*ndln+[1:ndln] ];end
        end;
        %----- calculation of vke and vfe
        vprelg=vprelg(kprop(ie,:)); % elementary properties
        [vke,vfe]=feval([deblank(nomtype(ktypel(ie,:))),'_ke'],ie);
        %----- assembly of vke
        vkg(kloce,kloce)=vkg(kloce,kloce) + vke;
        %----- assembly of vfe
        vfg(kloce)=vfg(kloce) + vfe;
        %----- calculation of vme:
        [vme]=feval([deblank(nomtype(ktypel(ie,:))),'_me'],ie);
        %----- assembly of vme
        vmkg(kloce,kloce)=vmkg(kloce,kloce) + vme;
    end
    %----- vmkg=vmg+dt*alfa*vkg
    if alfa > 0
        vmkg=vmkg+(dt*alfa)*vkg;
    end
```

```
%----- Introduction of the boundary conditions -----
%
% Technique demonstrated in section 4.7.2b: unit term on the diagonal
%
% For incremental formulation: vcond=0
for ic = 1 :ndlt
    if(kcond(ic) == 1)           % Dirichlet or kinematic
        %vfg=vfg - vmkg(:,ic)* vcond(ic) ;
        vmkg(ic,:) = zeros(1,ndlt);
        vmkg(:,ic) = zeros(ndlt,1);
        vmkg(ic,ic)=1; vfg(ic)= vcond(ic)*0;
    end
end
%----- loop on the steps -----
tpas=0;
iprint=1;                      % print control parameter
vsol=vsol0                      % updating by initial solution
for ipas=1:npas
    tpas=tpas+dt;
    fprintf(1,'pas=%5i dt= %12.6f tpas=%12.6f\n',ipas,dt,tpas)
    vres = vfg - vkg* vsol;      %--- residual at each step
%----- place non-null values of cls in  vsol
    kvec=find(kcond);          %--- imposed terms
    vsol(kvec)=vcond(kvec);
    if size(kvec,2) > 0         %--- non-null boundary conditions on vres
        vres(kvec)=zeros(size(kvec,2),1);
    end
%----- solution
    vdu = vmkg\ (dt*vres);    % incremental solution
    vsol= vsol + vdu;          % updating of the solution
    nres= norm(vres);          % norm of the residual
    ndu= norm(vdu)/norm(vsol); % relative norm of vdu
    fprintf(1,' normes nres= %12.6f ndu=%12.6f\n',nres,ndu);
    if iprint>0
        fprintf(' %12.5e%12.5e%12.5e%12.5e%12.5e%12.5e\n',vsol);disp(' ')
    end;
end
disp(' ----- print the solution DoF values -----')
for i=1:ndlt
    fprintf("%5i %12.5e\n",i,vsol(i));
end
fprintf(1,'time CPU=%12.6f secondes\n',0)
```

6.5.1.3 Bloc 'bnlin.m'

```
%----- parameters
    global nnt nelt ndlt ndln nnel ndle
    global ntypel nprop nprel
%----- mesh and allocation
    global vcor kcone c ktypel nomtype
%----- properties
    global vprelg vprel kprop
%----- solution
    global vsol vdle kloce
%----- times
    global vsol0 dt npas alfa npilot aval
%
time0=cputime           % initialize CPU time
%----- reading of the data fdata='nom_du_fichier'
feval(fdata)
%----- tracing of the mesh
close all
trace_mail(vcor,kcone)
ndmax=1e-6;
%----- initialization
disp(' ===== NLIN MODULE, finite element method version 1.0 =====');
disp(' ');
vkg=zeros(ndlt);        % global stiffness matrix
vfg=zeros(ndlt,1);      % global load vector
vfg=vfcg;               % updating by concentrated stress
%
disp('----- Piloting of the solution npilot 0= load, 1=CL, 2= viscosity -----')
fprintf(' dt= %12.6f npas=% 5i alfa= %12.6f npilot= %5i \n',dt,npas,alfa,npilot)
tpas=0;
iprint=1;                % print control parameter
vsol=vsol0;              % updating by initial solution
%
loop on the steps -----
for ipas=1:npas
    tpas=tpas+dt;         % dt=1 for a stationary problem
    aval=ipas/npas         % increment =1 if npas =1
    fprintf(1,'pas=%5i dt= %12.6f tpas=%12.6f aval=%12.6f\n',ipas,dt,tpas,aval)
    kv=find(kcond);
    if npilot==1           %--- piloting by boundary conditions
        vsol(kv)=aval*vcond(kv);
```

```

else
    vsol(kv)=vcond(kv);
end

%---- loop on the iterations-----
for iter=1:niter
    if npilot==1      %--- piloting by boundary conditions
        vsol(kv)=aval★vcond(kv);
    else
        vsol(kv)=vcond(kv);
    end
    vres=vfg;      %--- residual vector
    if npilot==0      %--- piloting by concentrated stress
        vres=aval★vfg;
    end

%---- Assembly: construct vkg tangent and vres -----
vkg=zeros(ndlt);
%---- loop on the elements
for ie=1:nelt
    fprintf('---- element ie = %5i\n',ie)
    %---- localization vector kloce for assembly
    kloce=[];
    for ii=1:nnel
        if kcone(ie,ii) > 0
            kloce=[kloce,(kcone(ie,ii)-1)*ndln+1:ndln];
        end
    end;
    %---- calculation of vke and vfe: tangential matrix and residual
    vprel=vprelg(kprop(ie,:)); % elementary properties
    [vke,vfe]=feval([deblank(nomtype(ktypel(ie,:)),'_kt'),ie]);
    %---- assembly of vke
    vkg(kloce,kloce)=vkg(kloce,kloce) + dt★vke;
    %---- assembly of vfe
    vres(kloce)=vres(kloce) + dt★vfe;
    %---- calculation of me if vprel(1) >0: transient
    if vprel(1) > 0
        [vme]=feval([deblank(nomtype(ktypel(ie,:)),'_me'),ie]);
        vkg(kloce,kloce)=vkg(kloce,kloce) + vme;
        vres(kloce)=vres(kloce) + vme★(vsol(kloce)-vsol0(kloce))
    end
end

```

```

%----- end of loop on the elements
%----- Introduction of boundary conditions
% Technique demonstrated in section 4.7.2b: unit term on the diagonal
% In incremental formulation: vcond=0
for ic = 1 :ndlt
    if(kcond(ic) == 1)           % Dirichlet or kinematic
        vkg(ic,:)= zeros(1,ndlt);
        vkg(:,ic) = zeros(ndlt,1);
        vkg(ic,ic)=1; vres(ic)= 0;
    end
end
%----- solution
vdu = vkg\vre;           % incremental solution
vsol= vsol + vdu;
nres= norm(vres);         % residual norm
ndu= norm(vdu)/norm(vsol); % norm relative to the incremental solution

fprintf(1,'ipas=%5 iter=%5 norme res=%12.6f ndu=%12.6f\n',ipas,iter,nres,ndu);
if(ndu < ndmax) break;end;
if iprint>0
    fprintf(' %12.5e%12.5e%12.5e%12.5e%12.5e%12.5e\n',vsol);disp(' ')
end;
end
%----- end of iteration loop-----
vsol0=vsol;      % updating for the next step
disp(' ----- print the solution DoF value -----')
for i=1:nnt
    ii=(i-1)*ndln+[1:ndln];
    fprintf("%5i %12.5e %12.5e\n",i,vsol(ii));
end
end
%----- end of loop on steps -----
fprintf(1,'Time CPU=%12.6f secondes\n',0)

```

6.5.1.4 Block for 'trace_mail.m'

This function displays the mesh.

```

function trace_mail(vcor,kconec)
%-----

```

```
% Display of 1- or 2-dimensional meshes
% Input: vcor : nodal coordinates nodales
% kconec : connectivities
% Output: display of the mesh
%-----
[nelt,nne]=size(kconec);
[nnt,ndim]=size(vcor) ;
hold on
%---- trace the mesh and display the numbers of the nodes and of the elements
for ie=1:nelt
    ivec=find( kconec(ie,:) );
    %--- non-null terms
    vcore=vcor(kconec(ie,ivec),:);
    xym=mean(vcore);
    if ndim==1
        line( vcore,zeros(size(ivec)) );
        text(xym,0,int2str(ie)); %--- element
    elseif ndim==2
        line( [vcore(:,1)',vcore(1,1)], [vcore(:,2)',vcore(1,2)] );
        text(xym(1),xym(2),int2str(ie)); %--- element
    end
end
%---- number the nodes
for i=1:nnt
    if ndim==1
        text(vcor(i),0,'*'); %--- node
    elseif ndim==2
        text(vcor(i,1),vcor(i,2) ,int2str(i));
    end
end
```

6.5.1.5 Block for 'degrade.m'

This function uses color shading (*dégradé de couleurs*) to display the obtained solutions for meshes composed of T3 or Q4 elements.

```
function[c] = degrade(vcor,vsol,kconec),
%-----
% Function to display the color shading of a field 'vsol' on a domain
% known by 'vcor' and 'kconec'. Underprint the gridlines in the background.
%-----
```

```

nelt=length(kconec); nnt=length(vsol); nnel=length(kconec(1,:)); ib=0;
for i=1:nelt % extraction of the 2-node elements
    if length(find(kconec(i,:)==0))==2
        ib=ib+1;
    end
end
c=zeros(nnsl,nelt-ib);
for j=1:nelt
    if(length(find(kconec(j,:)==0))~=2)
        vecx(:,j)= vcor([kconec(j,:)],1) ;
        vecy(:,j)= vcor([kconec(j,:)],2) ;
        for i=1:nnel
            c(i,j)=vsol( kconec(j,i) )' ;
        end
    end
end
hold on
fill(vecx,vecy,c)
axis equal
colorbar

```

6.5.2 LIST OF THERMAL ELEMENTS

6.5.2.1 T3 element: 'th_t3_ke.m', 'th_t3_me.m', 'th_t3_gr.m'

```

function [vke,vfe]=th_t3_ke(ie)
%=====
%T3 Laplace/harmonic/thermic element
%      see section 4.3.4.4a
% Calculation of vke and fe
%      Input:    vcore
%      vprel(1=c, 2=k1, 3=k2, 4=thickness, 5=fv)
%      Output:   vke(3, 3)and vfe(3,1)
%=====
%---- parameters
global nnt nelt ndlt ndln nnel ndle
global ntypel nprop nprel
%---- mesh and allocation
global vcor kconec ktypel
%---- properties

```

```

global vprelg vprel kprop
%----- solution
global vsol vdle
%-----
ndle=3;
vcore=vcor(kconec(ie,[1 2 3]),:);% elementary coordinates
%----- calculation of the determinant A=1/2★detj
detj=(vcore(2,1)-vcore(1,1))★(vcore(3,2)-vcore(1,2))...
-(vcore(3,1)-vcore(1,1))★(vcore(2,2)-vcore(1,2));
%----- matrix B (2, 3)
vb=[ (vcore([2 3 1],2)-vcore([3 1 2],2))';
      (vcore([3 1 2],1)-vcore([2 3 1],1))' ] /detj;
%----- matrix H of properties
vh=[vprel(2)★vprel(4) 0;0 vprel(3)★vprel(4)];
%----- matrice vke(3,3)
vke=vb'★(.5★detj★vh)★vb;
%----- vector vfe(3,1); vprel(4 5 6) :thickness, fv
vfe=(vprel(4)★vprel(5)★detj/6)★[1 1 1]';

function [vke]=th_t3_me(ie)
%=====
% T3 Laplace/harmonic/thermic element
% see section 4.3.4.4a
% Calculation of the mass matrix
% Input: vcore
% vprel(1=c, 2=k1, 3=k2, 4=thickness, 5=fv)
% Output: vke(3,3)
%=====
%----- parameters
global nnt nelt ndlt ndln nnel ndle
global ntypel nprop nprel
%----- mesh and allocation
global vcor kconec ktypel
%----- properties
global vprelg vprel kprop
%----- solution
global vsol vdle
%-----
ndle=3;
vcore=vcor(kconec(ie,[1 2 3]),:);% elementary coordinates
%----- calculation of the determinant A=1/2★detj

```

```

detj=(vcore(2,1)-vcore(1,1))★(vcore(3,2)-vcore(1,2))...
-(vcore(3,1)-vcore(1,1))★(vcore(2,2)-vcore(1,2));
%----- mass matrix vke(3,3): vprel(1 4 ) : ro, thickness
vke= (detj★vprel(1)★vprel(4)/24)★[2 1 1;1 2 1;1 1 2];

function vsig=th_t3_gr(ie)
%=====
% Calculation of the gradient
%T3 Laplace/harmonic/thermic element
%       see section 4.3.4.4a
%       Input:    vcore vdle
%                  vprel(1=c, 2=k1, 3=k2, 4=thickness, 5=fv)
%       Output:   qx and qy
%=====

%----- parameters
global nnt nelt ndlt ndln nnel ndle
global ntypel nprop nprel
%----- mesh and allocation
global vcor kconeck typel
%----- properties
global vprelg vprel kprop
%----- solution
global vsol vdle
%-----

ndle=3;
vcore=vcor(kconeck(ie,[1 2 3]),:);      % elementary coordinates
%----- calculation of the determinant A=1/2★detj
detj=(vcore(2,1)-vcore(1,1))★(vcore(3,2)-vcore(1,2))...
-(vcore(3,1)-vcore(1,1))★(vcore(2,2)-vcore(1,2));
%----- matrix B 2x3
vb=[ (vcore([2 3 1],2)-vcore([3 1 2],2 ))';
     (vcore([3 1 2],1)-vcore([2 3 1],1 ))' ] /detj;
%----- matrix H of properties
vh=[vprel(2)★vprel(4) 0;0 vprel(3)★vprel(4)];
%----- calculation of qx qy
vsig=-vh★vb★vdle;
fprintf(' %5i %15.5e %15.5e\n',[ie,vsig])

```

6.5.2.2 Q4 element : 'th_q4_ke.m', 'th_q4_me.m', 'th_q4_gr.m'

```

function [vke,vfe]=th_q4_ke(ie)
%=====
% Q4 Laplace/harmonic/thermic element
%      see section 4.3.4.4b
% Calculation of vke and vfe
% 2x2 numerical integration
%      Input:    vcore
%                  vprel(1=c, 2=k1, 3=k2, 4=thickness, 5=fv)
%      Output:   vke(4, 4) and vfe(4,1)
%=====

%---- parameters
global nnt nelt ndlt ndln nnel ndle
global ntypel nprop nprel
%---- mesh and allocation
global vcor kcone c ktypel
%---- properties
global vprelg vprel kprop
%---- solution
global vsol vdle
%---- 

ndle=4;
vcore=vcor(kcone(ie,[1:4]),:); % elementary coordinates
%---- coordinates of the 4 Gaussian points (PG)
c=1/sqrt(3);vpg=[1,1,1,1] ;npg=4;
ksig=[-c,-c ;-c, c ;c,-c ;c, c];
vke=zeros(ndle);vfe=zeros(ndle,1);
aire=0;
%---- loop on the PG
for ipg=1:npg
    ksi=ksig(ipg,1); eta=ksig(ipg,2);poid=vpg(ipg);
    %---- functions N, vn1=N,ksi N,eta
    vn=.25★[(1-ksi)★(1-eta) (1+ksi)★(1-eta)... 
    (1+ksi)★(1+eta) (1-ksi)★(1+eta)]; %--- vec 1x4
    vn1=.25★[-(1-eta) (1-eta) (1+eta) -(1+eta)
    -(1-ksi) -(1+ksi) (1+ksi) (1-ksi)];%--- vec 2x4
    %---- Jacobian matrix
    vj=vn1★vcore;
    detj=vj(1,1)★vj(2,2)-vj(1,2)★vj(2,1);
    pdet=detj★poid;
    aire=aire+pdet;
end

```

```

vji=[vj(2,2) -vj(1,2); -vj(2,1) vj(1,1)]/detj ;
%----matrix B 2x4 N,x N,y
vb=vji★vn1;
% matrix H of properties
vh=[vprel(2)★vprel(4) 0;0 vprel(3)★vprel(4)];
%---- matrix vke(4,4)
vke=vke+vb'★(vh★pdet)★vb;
%---- vector vfe(3,1);vprel(4 5 6) : thickness fv
if vprel(5) ~= 0
    vfe=vfe+vn'★pdet★vprel(5)★vprel(4);
end
%---- end of integration loop
end

function [vke]=th_q4_me(ie)
%=====
% Q4 Laplace/harmonic/thermic element
%      see section 4.3.4.4b
% Calculation of the mass matrix
% 2x2 numerical integration
%
% Input:      vc当地
%             vprel(1=c, 2=k1, 3=k2, 4=thickness, 5=fv)
% Output:     vke(4, 4)
%=====

%---- parameters
global nnt nelt ndlt ndln nnel ndle
global ntypel nprop nprel
%---- mesh and allocation
global vcor kconec ktypel
%---- properties
global vprelg vprel kprop
%---- solution
global vsol vdle
%---- 

ndle=4;
vc当地=vcor(kconec(ie,[1:4],:)); % elementary coordinates
%---- coordinates of the 4 Gaussian points (PG)
c=1/sqrt(3);vpg=[1,1,1,1] ;npg=4;
ksig=[-c,-c ;-c, c ;c,-c ;c, c];
vke=zeros(ndle);vfe=zeros(ndle,1);
aire=0;
%---- loop on the PG

```

```

for ipg=1:npg
    ksi=ksig(ipg,1); eta=ksig(ipg,2);poid=vgp(ipg);
    %---- functions N, vn1=N,ksi N,eta
    vn=.25★[(1-ksi)★(1-eta) (1+ksi)★(1-eta)...%
    (1+ksi)★(1+eta) (1-ksi)★(1+eta)]; %--- vec 1x4
    %---- Jacobian matrix
    detj=vj(1,1)★vj(2,2)-vj(1,2)★vj(2,1);
    pdet=detj★poid;
    %---- mass matrix vke(4,4):vprel(1 4) :ro thickness
    vke=vke+vn'★(vprel(4)★vprel(1)★pdet)★vn;
%---- end of integration loop
end

function vsig=th_q4_gr(ie)
%=====
% Calculation of the gradient
% Q4 Laplace/harmonic/thermic element
% see section 4.3.4.4a
% 2x2 numerical integration
% Input: vcore vdle
% vprel(1=c, 2=k1, 3=k2, 4=thickness, 5=fv)
% Output: qx and qy
%=====
%---- parameters
global nnt nelt ndlt ndln nnel ndle
global ntypel nprop nprel
%---- mesh and allocation
global vcor kcone c ktypel
%---- properties
global vprelg vprel kprop
%---- solution
global vsol vdle
%----%
ndle=4;
vcore=vcor(kcone(ie,[1:4]),:); % elementary coordinates
%---- coordinates of the 4 Gaussian points
c=1/sqrt(3);vpg=[1,1,1,1] ;npg=4;
ksig=[-c,-c ;-c, c ;c,-c ;c, c];
%---- gradients on the PG
aire=0;vsig=zeros(4,2);
for ipg=1:npg

```

```

ksi=ksig(ipg,1); eta=ksig(ipg,2);poid=vgp(ipg);
%---- functions N, vn1=N,ksi N,eta
vn=.25★[(1-ksi)★(1-eta) (1+ksi)★(1-eta)... 
    (1+ksi)★(1+eta) (1-ksi)★(1+eta)]; %--- vec 1x4
vn1=.25★[-(1-eta) (1-eta) (1+eta) -(1+eta)
    -(1-ksi) -(1+ksi) (1+ksi) (1-ksi)];%--- vec 2x4
%---- Jacobian matrix
vj=vn1★vcore;
detj=vj(1,1)★vj(2,2)-vj(1,2)★vj(2,1);
pdet=detj★poid;
aire=aire+pdet;
vji=[vj(2,2)-vj(1,2); -vj(2,1) vj(1,1)]/detj ;
%---- matrix B (2,4) N,x N,y
vb=vji★vn1;
% matrix vh of properties
vh=[vprel(2)★vprel(4) 0;0 vprel(3)★vprel(4)];
%--- flux qx qy
vsig(ipg,:)=-(vh★vb★vdle)';
if(ipg ==1) fprintf(' %5i %15.5e %15.5e\n',[ie,vsig(1,:)]);
else fprintf(' %15.5e %15.5e\n',[ vsig(ipg,:)]);
end
%---- end of integration loop
end

```

6.5.2.3 L2 element : 'th_l2b_ke.m', 'th_l2b_gr.m'

```

function [vke,vfe]=th_l2b_ke(ie)
%=====
% L2 boundary element for T3/Q4 Laplace
% see section 4.3.4.4a
% Input: vcold
%         vprel(1=c,2=alfa,3=T_ext,4=thickness,5=fs)
% Output: vke(2,2) and vfe(2,1)
%=====

%---- parameters
global nnt nelt ndlt ndln nnel ndle
global ntypel nprop nprel
%---- mesh and allocation
global vcor kconec ktypel
%---- properties
global vprelg vprel kprop

```

```
%----- solution
global vsol vdle
%-----
ndle=2;
vcore=vcor(kconec(ie,[1 2]),:); % elementary coordinates
%----- length
xL=( vcose(2,1)-vcose(1,1))^2+ (vcose(2,2)-vcose(1,2))^2 )^.5;
%----- convection matrix vprel(2 3 4):alfa,T_ext,thickness
vke=(vprel(2)*vprel(4)*xL/6)*[ 2 1 ; 1 2 ];
%----- stress vprel(5):fs
vfe=( (vprel(5)+vprel(2)*vprel(3))*vprel(4)*xL/2)*[1 1 ]';

function vsig=th_l2b_gr(ie)
%=====
% L2 boundary element for T3/Q4 Laplace
% Calculation of qn= alfa*(T-T0)-fs
%
% Input:    vcose
%           vprel(1=c , 2=alfa, 3=T_ext , 4=thickness, 5=fs)
%
% Output:   qn
%=====

%----- parameters
global nnt nelt ndlt ndln nnel ndle
global ntypel nprop nprel
%----- mesh and allocation
global vcose kconec ktypel
%----- properties
global vprelg vprel kprop
%----- solution
global vsol vdle
%-----
ndle=2;
vcose=vcor(kconec(ie,[1 2]),:); % elementary coordinates
%----- length
xL=( vcose(2,1)-vcose(1,1))^2+ (vcose(2,2)-vcose(1,2))^2 )^.5;
%----- convection matrix vprel(2 3 4):alfa,T_ext,thickness
vke=(vprel(2)*vprel(4)*xL/6)*[ 2 1 ; 1 2 ];
%----- stress vprel(5):fs
vfe=( (vprel(5)+vprel(2)*vprel(3))*vprel(4)*xL/2)*[1 1 ];
vsig= vke*vdle-vfe;
fprintf(' %5i %15.5e %15.5e\n',[ie,vsig'])
```

6.5.3 LIST OF ELASTIC ELEMENTS

6.5.3.1 T3 element: 'el_t3_ke.m', 'el_t3_gr.m'

```

function [vke,vfe]=el_t3_ke(ie)
%=====
%T3 element (6x6) with isotropic elasticity: u v at the nodes
% Plane Constraints (CP) or Plane Deformations (DP)
% see section 4.3.4.5a
% Calculation of vke and of vfe
%
% Input:    vcore,
%           vprel( 1=ro ,2=E, 3=nu, 4 : Indicator CP=0 or DP=1,
%           5= thickness 6=fvx 7=fvy)
% Output:   vke(6,6) and vfe(6 1)
%=====

%---- parameters
global nnt nelt ndlt ndln nnel ndle
global ntypel nprop nprel
%---- mesh and allocation
global vcor kconec ktypel
%---- properties
global vprelg vprel kprop
%---- solution
global vsol vdle
%---- ndle=6;
vcore=vcor(kconec(ie,[1:3]),:); % elementary coordinates
%---- calculation of the determinant A=1/2★detj
detj=(vcore(2,1)-vcore(1,1))★(vcore(3,2)-vcore(1,2))...
      -(vcore(3,1)-vcore(1,1))★(vcore(2,2)-vcore(1,2));
%---- matrix B (3,6)
vb=zeros(3,6);
vb(1,[1 3 5])=(vcore([2 3 1],2)-vcore([3 1 2],2))' / detj;
vb(2,[2 4 6])=(vcore([3 1 2],1)-vcore([2 3 1],1))' / detj ;
vb(3,[1 3 5 2 4 6]) =[ vb(2,[2 4 6]), vb(1,[1 3 5]) ];
%---- matrix of the elastic properties vh(3x3)
cnu=vprel(3);eh=vprel(2)★vprel(5); % (5) =thickness
if(vprel(4)==0) % planar constraints
    vh=eh/(1-cnu★cnu)★[1 cnu 0;cnu 1 0 ;0 0 (1-cnu)/2];
else

```

```

c1=eh/((1+cnu)*(1-2*cnu));
vh=[c1*(1-cnu) c1*cnu 0;
    c1*cnu c1*(1-cnu) 0;
    0 0 .5*eh/(1+cnu)];
end
%---- matrix vke(6,6)
vke=vb'* (0.5*detj*vh)*vb;
%---- vector vfe(6,1); vprel(5 6 7):thickness fvx fvy
vfe([1 3 5],1)=(vprel(5)*vprel(6)*detj/6)*[1 1 1]';
vfe([2 4 6],1)=(vprel(5)*vprel(7)*detj/6)*[1 1 1];

function vsig=el_t3_gr(ie)
%=====
% Calculation of the constraints sigx sigy sigxy
% Elastic T3 element (Plane Constraints (CP) or Plane Deformations (DP))
%
% see section 4.3.4.5a
%
% Input: vcore, vdle
%         vprel(1=ro ,2=E, 3=nu, 4=ind-cp=0 or dp=1
%                 5= thickness 6=fvx 7=fvy)
%
% Output: sigx sigy sigxy
%=====

%---- parameters
global nnt nelt ndlt ndln nnel ndle
global ntypel nprop nprel
%---- mesh and allocation
global vcor kconecktypel
%---- properties
global vprelg vprel kprop
%---- solution
global vsol vdle
%---- 

ndle=6;
vcore=vcor(kconeck,[ie,1:3],:); % elementary coordinates
%---- calculation of the determinant A=1/2*detj
detj=(vcore(2,1)-vcore(1,1))*(vcore(3,2)-vcore(1,2))...
-(vcore(3,1)-vcore(1,1))*(vcore(2,2)-vcore(1,2));
%---- matrix B (3X6)
vb=zeros(3,6);
vb(1,[1 3 5])=(vcore([2 3 1],2)-vcore([3 1 2],2))'/detj;
vb(2,[2 4 6])=(vcore([3 1 2],1)-vcore([2 3 1],1))'/detj ;
vb(3,[1 3 5 2 4 6])=[ vb(2,[2 4 6]), vb(1,[1 3 5]) ];

```

```
%---- matrix of elastic properties vh(3x3)
cnu=vprel(3);eh=vprel(2)*vprel(5); % (5) =thickness
if(vprel(4)==0) % planar constraint
    vh=eh/(1-cnu*cnu)*[1 cnu 0;cnu 1 0 ;0 0 (1-cnu)/2];
else
    c1=eh/((1+cnu)*(1-2*cnu));
    vh=[c1*(1-cnu) c1*cnu 0;
         c1*cnu c1*(1-cnu) 0;
         0 0 .5*eh/(1+cnu)];
end
%---- calculation of sigma
vsig=(vh*vb*vdl)';
fprintf(' %5i %15.5e%15.5e %15.5e\n',[ie,vsig]);
```

6.5.3.2 Q4 element : 'el_q4_ke.m', 'el_q4_gr.m'

```
function [vke,vfe]=el_q4_ke(ie)
%=====
% Q4(8 x 8) element with isotropic elasticity: u v at the nodes
% Plane Constraints (CP) or Plane Deformations (DP)
% see section 4.3.4.5b
% Calculation of vke and of vfe
% 2x2 numerical integration
%
% Input: vcore,
%        vprel(1=ro ,2=E, 3=nu, 4=ind-cp=0 or dp=1,
%        5= thickness 6=fvx 7=fvy)
%
% Output: vke(8, 8) and vfe(8 1)
%=====

%---- parameters
global nnt nelt ndlt ndln nnel ndle
global ntypel nprop nprel
%---- mesh and allocation
global vcor kconec ktypel
%---- properties
global vprelg vprel kprop
%---- solution
global vsol vdle
%----  

ndle=8;
vcore=vcor(kconec(ie,[1:4],:); % elementary coordinates
%---- coordinates of the 4 Gaussian points (PG)
```

```

c=1/sqrt(3);vpg=[1,1,1,1] ;npg=4;
ksig=[-c,-c ;-c, c ;c,-c ;c, c];
%----- matrix of elastic properties vh(3x3)
cnu=vprel(3);eh=vprel(2)*vprel(5); % (5) =thickness
if(vprel(4)==0) % planar constraints
    vh=eh/(1-cnu*cnu)*[1 cnu 0;cnu 1 0 ;0 0 (1-cnu)/2];
else
    c1=eh/((1+cnu)*(1-2*cnu));
    vh=[ c1*(1-cnu) c1*cnu 0;
          c1*cnu c1*(1-cnu) 0;
          0 0 .5*eh/(1+cnu)];
end
%---- loop on the PG
aire=0;vke=zeros(ndle);vfe=zeros(ndle,1);
for ipg=1:npg
    ksi=ksig(ipg,1);eta=ksig(ipg,2);poid=vpg(ipg);
    %---- functions N, vn1=N,ksi N,eta
    vn=.25*[(1-ksi)*(1-eta) (1+ksi)*(1-eta)*...
               (1+ksi)*(1+eta) (1-ksi)*(1+eta)]; %--- vec 1x4
    vn1=.25*[-(1-eta) (1-eta) (1+eta) -(1+eta)
               -(1-ksi) -(1+ksi) (1+ksi) (1-ksi)];%--- vec 2x4
    %---- Jacobian matrix
    vj=vn1*vcore;
    detj=vj(1,1)*vj(2,2)-vj(1,2)*vj(2,1);
    pdet=detj*poid;
    aire=aire+pdet;
    vji=[vj(2,2) -vj(1,2); -vj(2,1) vj(1,1)]/detj ;
    %---- N,x N,y
    vnx=vji*vn1;
    %---- matrix B(3x8)
    vb=zeros(3,8);
    vb(1,[1 3 5 7])=vn1(1,:);
    vb(2,[2 4 6 8])=vn1(2,:);
    vb(3,[1 3 5 7,2 4 6 8])=[vn1(2,:);vn1(1,:)];
    %---- matrix vke(8x8)
    vke=vke+vb*(vh*pdet)*vb;
    %---- vector vfe(8,1) ;vprel(5 6 7):thickness fvx fvy
    vfe([1 3 5 7],1)=vfe([1 3 5 7],1)+(vprel(5)*vprel(6)*pdet)*vn';
    vfe([2 4 6 8],1)=vfe([2 4 6 8],1)+(vprel(5)*vprel(7)*pdet)*vn';
%---- end of integration loop

```

```

    end

function vsig=el_q4_gr(ie)
%=====
% Calculation of the constraints sigx sigy sigxy
% Elastic Q4 element (Plane Constraints (CP) or Plane Deformations (DP))
%      see section 4.3.4.5b
% 2x2 numerical integration
%
% Input:    vcore, vdle
%           vprel(1=ro ,2=E, 3=nu, 4: CP=0 or DP=1,
%           5=thickness 6=fvx 7=fvy)
%
% Output:   sigx sigy sigxy
%=====

%---- parameters
global nnt nelt ndlt ndln nnel ndle
global ntypel nprop nprel
%---- mesh and allocation
global vcor kconec ktypel
%---- properties
global vprelg vprel kprop
%---- solution
global vsol vdle
%---- ndle=8;
vcore=vcor(kconec(ie,[1:4],:)); % elementary coordinates
%---- coordinates of the 4 Gaussian points (PG)
c=1/sqrt(3);vpg=[1,1,1,1] ;npg=4;
ksig=[-c,-c ;-c, c ;c,-c ;c, c];
%---- matrix of elastic properties vh(3x3)
cnu=vprel(3);eh=vprel(2)*vprel(5); % (5) =thickness
if(vprel(4)==0) % planar constraints
    vh=eh/(1-cnu*cnu)*[1 cnu 0;cnu 1 0 ;0 0 (1-cnu)/2];
else
    c1=eh/((1+cnu)*(1-2*cnu));
    vh=[c1*(1-cnu) c1*cnu 0;
         c1*cnu c1*(1-cnu) 0;
         0 0 .5*eh/(1+cnu)];
end
%---- loop on the PG
aire=0;vsig=zeros(4,3);
for ipg=1:npg

```

```

ksi=ksig(ipg,1); eta=ksig(ipg,2);poid=vpg(ipg);
%----- functions N, vn1=N,ksi N,eta
vn=.25*[ (1-ksi)*(1-eta) (1+ksi)*(1-eta)...
(1+ksi)*(1+eta) (1-ksi)*(1+eta)]; %--- vec 1x4
vn1=.25*[-(1-eta) (1-eta) (1+eta) -(1+eta)
-(1-ksi) -(1+ksi) (1+ksi) (1-ksi)]; %--- vec 2x4
%----- Jacobian matrix
vj=vn1*vcore;
detj=vj(1,1)*vj(2,2)-vj(1,2)*vj(2,1);
pdet=detj*poid;
aire=aire+pdet;
vji=[vj(2,2) -vj(1,2); -vj(2,1) vj(1,1)]/detj ;
%---- N,x N,y
vnx=vji*vn1;
%---- matrix B(3x8)
vb=zeros(3,8);
vb(1,[1 3 5 7])=vnx(1,:);
vb(2,[2 4 6 8])=vnx(2,:);
vb(3,[1 3 5 7,2 4 6 8])=[vnx(2,:);vnx(1,:)];
%---- calculation of sigma
vsig(ipg,:)=(vh*vb*vdle)';
if(ipg ==1) fprintf(' %5i %15.5e%15.5e %15.5e\n',[ie,vsig(1,:)]);
else fprintf(' %15.5e %15.5e%15.5e\n',[ vsig(ipg,:)]);
end
%---- end of integration loop
end

```

6.5.3.3 L2 element: 'el_l2b_ke.m', 'el_l2b_gr.m'

```

function [vke,vfe]=el_l2b_gr(ie)
%=====
% Boundary element for elastic T3/Q4
% 2-node element
% Input: vcore,
%        vprel(1=ro ,2,3,4=vide ,5= thickness 6=fsx 7=fsy)
% Output: vke(4, 4) and vfe(4,1)
%=====
%---- parameters
global nnt nelt ndlt ndln nnel ndle
global ntypel nprop nprel
%---- mesh and allocation

```

```

global vcor kconec ktypel
%---- properties
global vprelg vprel kprop
%---- solution
global vsol vdle
%-----
ndle=4;vke=zeros(ndle);vfe=zeros(ndle,1);
vcore=vcor(kconec(ie,[1 2]),:); % elementary coordinates
%---- length
xL=( (vcore(2,1)-vcore(1,1))^2+ (vcore(2,2)-vcore(1,2))^2 )^.5;
%---- load vector fx,fy on the boundary
vfe([1 3],1)=(vprel(5)*vprel(6)*xL/2)*[1 1]';
vfe([2 4],1)=(vprel(5)*vprel(7)*xL/2)*[1 1]';

function [vke,vfe]=el_l2b_gr(ie)
%=====
% Calculation of the normal and tangential constraints, vfe: sign sigt
% Boundary element for elastic T3/Q4
% 2-node element
% Input: vcore,
%        vprel(1=ro ,2,3,4=void,5= thickness 6=fsx 7=fsy)
% Output: vke(4, 4) and vfe(4,1)
%=====

%---- parameters
global nnt nelt ndlt ndln nnel ndle
global ntypel nprop nprel
%---- mesh and allocation
global vcor kconec ktypel
%---- properties
global vprelg vprel kprop
%---- solution
global vsol vdle
%-----
ndle=4;vke=zeros(ndle);vfe=zeros(ndle,1);
vcore=vcor(kconec(ie,[1 2]),:); % elementary coordinates
%---- length
xL=( (vcore(2,1)-vcore(1,1))^2+ (vcore(2,2)-vcore(1,2))^2 )^.5;
%---- load vector fx,fy on the boundary
vfe([1 3],1)=(vprel(5)*vprel(6)*xL/2)*[1 1]';
vfe([2 4],1)=(vprel(5)*vprel(7)*xL/2)*[1 1]';
fprintf(' %5i%15.5e %15.5e%15.5e%15.5e\n',[ie, vfe'])

```

6.5.4 LIST OF ELEMENTS FOR FLUID MECHANICS

6.5.4.1 Q4 element: 'ns_q4_ke.m', 'ns_q4_me.m'

```

function [vke,vfe]=ns_q4_kt(ie)
%=====
% Calculation of the incompressible Navier-Stokes tangential matrix:
% Calculation of residual
% Formulation with penalty: u,v see Chapters 3 and 4
% 4-node element
% Linear terms: Stokes
%   [2*nu*(u,x*u,x+v,y*v,y) + nu*(u,y+v,x)*(u,y+v,x)] [(u,x+v,y)*p]
%   [-p*(u,x + v,y)]           [- p*p /lamda]
% Nonlinear terms:
%   u*( (2u du),x +(v du),y )   u*(u dv),y
%   v*(v du ),x      v*( (u dv),x +(2v dv),y)
% Nonlinear residual:
%   u*( (u u),x      u*(u v),y
%   v*(u v ),x      v*(v v),y
% Approximation:
%   u=Ni*ui; u^2=Ni*(ui^2); u*v=Ni*(ui*vi)
%   Input:    vcore,vprel(ro, nu lamda)
%   Output:   vke(8, 8) vfe(,1)
%=====

%---- parameters
global nnt nelt ndlt ndln nnel ndle
global ntypel nprop nprel
%---- mesh and allocation
global vcor kcnecl ktypel nomtype
%---- properties
global vprelg vprel kprop
%---- solution
global vsol vdle kloce
%---- time
global vsol0 dt npas alfa npilot aval
%-----
%-----

nonlin=1 ;      %--- indicator of nonlinear calculation
ndle=8;
vke = zeros(ndle);vfe=zeros(ndle,1);

```

```

vkup=zeros(ndle,1);
lamda=vprel(3);
if npilot==2           %---piloting by viscosity
    vprel(2)=vprel(2)/aval;
    re=1/vprel(2)
end
vh=vprel(2)*[2 0 0;0 2 0;0 0 1]; %--- matrix of viscous effects
%----- coordinates of the 4 Gaussian points (PG)
c=1/sqrt(3);
ksig=[-c,-c ; -c, c ; c,-c ; c, c];
vpg=[1,1,1,1] ; npg=4;
%----- local vectors
vcore=vcor(kconec(ie,:));      % local coordinates
vdle=vsol(kloce) ;             %--- elementary solution
locu = [ 1 3 5 7 ];            %--- localization of the DoF on 'u'
locv = [ 2 4 6 8 ];            %--- localization of the DoF on 'v'
solu = vdle( locu )';         %--- velocity u
solv = vdle( locv )';         %--- velocity v
%----- loop on the steps -----
aire=0;
for ipg=1:npg
    ksi=ksig(ipg,1); eta=ksig(ipg,2); poid=vpg(ipg);
    %---- functions N,  vn1=N,ksi N,eta
    vn=.25★[(1-ksi)★(1-eta) (1+ksi)★(1-eta)...
               (1+ksi)★(1+eta) (1-ksi)★(1+eta)];  %--- vector 1x4
    vn1=.25★[-(1-eta) (1-eta) (1+eta) -(1+eta)
               -(1-ksi) -(1+ksi) (1+ksi) (1-ksi)];  %--- vector 2x4
    %---- Jacobian matrix
    vj=vn1★vcore;
    detj=vj(1,1)★vj(2,2)-vj(1,2)★vj(2,1);
    pdet=detj★poid;
    aire=aire+pdet;
    vji=[vj(2,2) -vj(1,2); -vj(2,1) vj(1,1)]/detj ;
    %---- N,x N,y
    vnx=vji★vn1;
    %---- Linear terms: Stokes
    vb=zeros(3,8);
    vb(1,locu)=vnx(1,:);
    vb(2,locv)=vnx(2,:);
    vb(3,[locu,locv])=[vnx(2,:),vnx(1,:)];
    %---- calculation of the tangential k: Stokes

```

```

vke=vke+vb'*(vh*pdet)*vb;
vkup=vkup+vnx(:)*pdet; %--- pressure velocity term
%----- calculation of the residual: Stokes
grad=vb★vdle; %--- velocity gradient
sigma=vh★grad; %--- viscous constraints
vfe=vfe+vb'*(pdet★sigma);
%----- nonlinear terms
if nonlin>0
    v0=pdet★vn;
    %---- calculation of nonlinear tangential k
    vke(locu,locu)=vke(locu,locu)+...
        v0'*( vn1(1,:).*solu* 2)+vn1(2,:).*solv );
    vke(locu,locv)=vke(locu,locv)+...
        v0'* ( vn1(2,:).* solu );
    vke(locv,locu)=vke(locv,locu)+...
        v0'* ( vn1(1,:).*solv );
    vke(locv,locv)=vke(locv,locv)+...
        v0'* ( vn1(1,:).*solv + vn1(2,:).* (solv*2) );
    %---- calculation of the nonlinear residual
    vfe(locu,1)=vfe(locu,1)+...
        v0' *(vn1(1,:)*(solu.*solu)' +vn1(2,:)*(solu.*solv)' );
    vfe(locv,1)=vfe(locv,1)+...
        v0' *(vn1(1,:)*(solu.*solv)' +vn1(2,:)*(solv.*solv)' );
    end
end
%----- end of integration loop
%----- condensation on pressure
vke=vke+vkup★(vkup'*lamda/aire);
%----- term of pressure on the residual
p=-(lamda/aire)★vkup★vdle ;
vfe= -( vfe - vkup★p ) ;
fprintf(' pressure =%12.6f\n',p)

function [vke]=th_q4_me(ie)
%=====
% Q4 mass matrix element for Navier-Stokes
% 2x2 numerical integration
% Input: vcore
%         vprel(1=c )
% Output: vke(8,8)
%=====

```

```
%---- parameters
    global nnt nelt ndlt ndln nnel ndle
    global ntypel nprop nprel
%---- mesh and allocation
    global vcor kcone c ktypel nomtype
%---- properties
    global vprelg vprel kprop
%---- solution
    global vsol vdle
%---- time
    global vsol0 dt npas alfa npilot aval
%-----
%-----
    ndle=8;vke=zeros(ndle);
    vcore=vcor(kcone(ie,[1:4],:)); % elementary coordinates
%---- coordinates of the 4 Gaussian points (PG)
    c=1/sqrt(3);vpg=[1,1,1,1];npg=4;
    ksig=[-c,-c;c,c;-c,-c;c,c];
    vke=zeros(ndle);vfe=zeros(ndle,1);
    aire=0;
    iu=[1 3 5 7];iv=[2 4 6 8];
%---- loop on the PG
    for ipg=1:npg
        ksi=ksig(ipg,1); eta=ksig(ipg,2);poid=vpg(ipg);
        %---- functions N, vn1=N,ksi N,eta
        vn=.25★[(1-ksi)★(1-eta) (1+ksi)★(1-eta)...
            (1+ksi)★(1+eta) (1-ksi)★(1+eta)]; %--- vec 1x4
        %---- Jacobian matrix
        detj=vj(1,1)★vj(2,2)-vj(1,2)★vj(2,1);
        pdet=detj★poid;
        %---- mass matrix vke(4,4): vprel(1 4) :ro thickness
        vke(iu,iu)=vke(iu,iu)+vn'★vprel(1)★pdet★vn;
%---- end of integration loop
    end
    vke(iv,iv)=vke(iu,iu);
```

6.5.4.2 L2 element: 'ns_L2b_ke.m'

```
function [vke,vfe]=ns_L2b_kt(ie)
%=====
% Calculation of the stresses by imposed pressure
```

```
% Possibly, calculation of the friction matrix u*nu*u
% npilot=0 : piloting of final pressure
% p= vprel(4)
%=====
%----- parameters
    global nnt nelt ndlt ndln nnel ndle
    global ntypel nprop nprel
%----- mesh and allocation
    global vcor kcone c ktypel nomtype
%----- properties
    global vprelg vprel kprop
%----- solution
    global vsol vdle
%----- time
    global vsol0 dt npas alfa npilot aval
%-----
ndle=4;vke=zeros(ndlt);
vfe = zeros(ndle,1);vke=zeros(ndle);
%---- local vectors
vcore=vcor(kcone(ie,[1 2]),:); % local coordinates
x = vcore(2,:) - vcore(1,:);
xL = sqrt(x * x');
%-- calculation of the normals
nx = (vcore(2,2)- vcore(1,2))/xL;
ny = -(vcore(2,1)- vcore(1,1))/xL;
pe=vprel(4);
if npilot==0           %--- piloting by pressure
    pe=aval*pe;
end
vfe=-pe*xL*nx*[1/2;0;1/2;0];
```

6.6 Examples of application

In this section, we present three groups of examples based on the use of the finite element method:

Thermics:

- Neumann patch tests with T3 and Q4;

- heat transfer in a sector of a disk;
- heat transfer in a square plate (Poisson).

Planar elasticity:

- mechanical patch test with T3 and Q4;
- deformation of a slanting plate;
- thin bending beam.

Fluid mechanics:

- patch test with Q4;
- Poiseuille flow;
- lid-driven cavity flow;
- potential flow.

6.6.1 Heat transfer problems

a) Neumann patch tests with T3 and Q4 elements

The domain is a square whose side measures 1 m in length. Here, we perform a Neumann patch test (see section 4.1.3) for two meshes, respectively, comprising T3 elements ('th_t3') and Q4 elements Q4 ('th_q4') (see Figure 6.2).

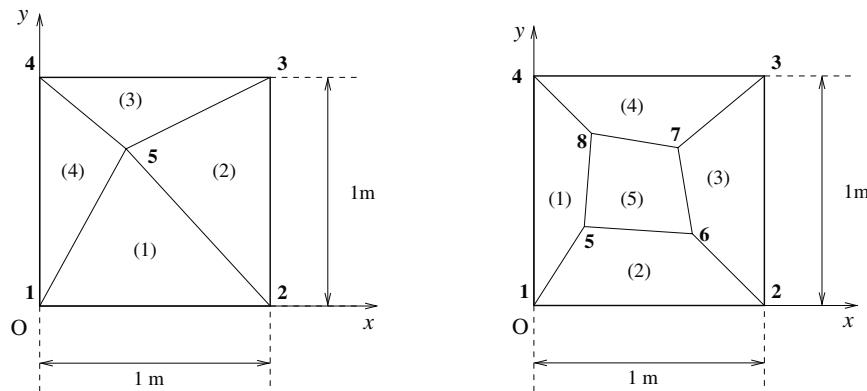


Figure 6.2. Meshes for patch tests with T3 and Q4 elements

The solution imposed is:

$$T(x, y) = 1 + 2.1x + 3.2y,$$

with flux boundary conditions:

$$\begin{aligned} f_s &= -6.4 \text{ on the segment } 1-2; \quad f_s = +6.4 \text{ on the segment } 3-4; \\ f_s &= -2.1 \text{ on the segment } 4-1; \quad f_s = +2.1 \text{ on the segment } 2-3; \end{aligned}$$

$$\text{where: } f_s = -q_n \text{ with } \vec{q} = -\begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix} \vec{\nabla} T.$$

The boundary conditions are imposed with four truss elements ('th_l2b'). A Dirichlet condition is imposed on the first node to eliminate the rigid body mode. The conductivities in relation to x and y are $k_1 = 1 \text{ W / m} - {}^\circ\text{C}$ and $k_2 = 2 \text{ W / m} - {}^\circ\text{C}$. The thickness is taken to be 2 m.

Data file with T3 : 'dthpatcht3.m'

```

nomtype=['th_t3 ';'th_l2b']
ndln=1;
%----- Reading of the coordinates x y
vcor=[0.0 0.0; 1.0 0.0; 1.0 1.0; 0.0 1.0; 0.55 0.66];
%----- Reading of the connectivities / elementary type
% 4 × T3 elements and 4 × L2b elements
kconec=[1 2 5; 2 3 5; 3 4 5; 4 1 5; 1 2 0; 2 3 0; 3 4 0; 4 1 0];
nnt=size(vcor,1);nnel=size(kconec,2);
nelt=size(kconec,1);
ndlt=ndln*nnt;
ktypel=[1 1 1 1 2 2 2 2];
%----- Properties
%T3:      c d1 d2 thickness=2 fv=0
% l2b :    c=0 alfa=0 T0=0 thickness=2 fs=...
vprelg=[0 1 2 2 0;0 0 0 2 -6.4;0 0 0 2 2.1;0 0 0 2 6.4;0 0 0 2 -2.1];
nprel=5;
kprop=[1 1 1 1 2 3 4 5]; % 5 types of properties
%----- Boundary condition
% Dirichlet: node 1: u=1
% kcond=[1 1 1 1 0]; %vcond=[1 3.1 6.3 4.2 0]; % for Dirichlet patch test
kcond=[1 0 0 0 0];vcond=zeros(1,ndlt);vcond(1)=1;
%----- load vector
vfcg=zeros(ndlt,1);

```

Data file with Q4: 'dthpatchq4.m'

```

nomtype=['th_q4 ';'th_l2b']
ndln=1;
%----- Reading of the coordinates x y
vcor=[0.0 0.0; 1.0 0.0; 1.0 1.0; 0.0 1.0; 0.3 0.3; 0.6 0.4; 0.7 0.66; 0.35 0.8];
%----- Reading of the connectivities/ elementary type
% 5 × Q4 elements and 4 × L2b elements
kcone=[1 5 8 4;1 2 6 5;2 3 7 6;3 4 8 7;5 6 7 8;1 2 0 0;2 3 0 0;3 4 0 0;4 1 0 0];
nnt=size(vcor,1);nnel=size(kcone,2);
nelt=size(kcone,1);
ndlt=ndln*nnt;
ktypel=[1 1 1 1 2 2 2 2];
%----- Properties
%T3 :      c d1 d2 thickness=2 fv=0
%l2b :      c=0 alfa=0 T0=0 thickness=2 fs=...
vprelg=[0 1 2 2 0;      0 0 0 2 -6.4;      0 0 0 2  2.1;      0 0 0 2  6.4;      0 0 0 2 -2.1];
nprel=5;
kprop=[1 1 1 1 2 3 4 5]; % 5 types of properties
%----- Boundary conditions
% Dirichlet: node 1: u=1
% kcond=[1 1 1 1 0]; %vcond=[1 3.1 6.3 4.2 0]; % for Dirichlet patch test
kcond=[1 0 0 0 0 0 0];vcond=zeros(1,ndlt);vcond(1)=1;
%----- load vector
vfcg=zeros(ndlt,1);

```

- Finite element method results:

The patch test comprising T3 elements is executed by successively typing, below the Matlab[®] invite ('>>'):

```

>> fdata='dthpatcht3'
>> blin

```

The solution obtained at the nodes is:

$$\langle T \rangle = \langle 1 \ 3.1 \ 6.3 \ 4.2 \ 4.267 \rangle.$$

The patch test comprising Q4 elements, for its part, is executed in the same way while taking care to change the name of the data file:

```

>> fdata='dthpatchq4'
>> blin

```

The solution obtained at the nodes is:

$$\langle T \rangle = \langle 1 \ 3.1 \ 6.3 \ 4.2 \ 2.59 \ 3.54 \ 4.582 \ 5.295 \rangle.$$

The solutions obtained for the two types of mesh satisfy the patch test (numerical solution identical to the analytical solution).

b) Heat transfer in a sector of a disk

The domain is a sector of a disk, shown in Figure 6.3:

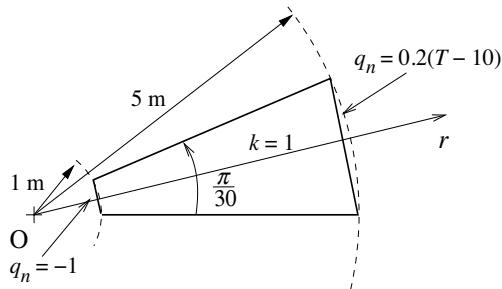


Figure 6.3. Boundary of the sector of a disk

For a unit heat conductivity, the axisymmetrical stationary solution of the problem is given by:

$$T_{ex}(r) = 11 + \ln\left(\frac{5}{r}\right), \quad r: \text{radius.}$$

The boundary conditions are given in Figure 6.3. Here, the problem is dealt with for two meshes, respectively, comprising eight T3 elements ('th_t3') and four Q4 elements ('th_q4') for 10 nodes. The integration of the boundary conditions requires two truss elements ('th_l2b') for both meshes.

Data file with T3: 'dthdiskt3.m'

```
nomtype=['th_t3 ';'th_l2b']
ndln=1;ndle=3;
%---- Reading of the coordinates x y
rad=pi/30; % angle of the sector of the disk
vcor=[1 0.0; cos(rad) sin(rad); 2 0.0; 2*cos(rad) 2*sin(rad); 3 0; 3*cos(rad) 3*sin(rad); 4 0;
      4*cos(rad) 4*sin(rad); 5 0; 5*cos(rad) 5*sin(rad)];
```

```
%----- Reading of the connectivities/ elementary type
kcone=[1 4 2;1 3 4;3 6 4;3 5 6;5 8 6; 5 7 8;7 10 8;7 9 10;1 2 0;9 10 0];
nnt=size(vcor,1);nnel=size(kcone,2);
nelt=size(kcone,1);
ndlt=ndln*nnt;
ktypel=ones(1,nelt);ktypel([9 10])=2;
%----- Properties
%T3:      c d1  d2 thickness fv
%l2b :      c alfa T0 thickness fs
% type 2: qn=-1;type 3: alfa(T-T0=10)
vprelg=[0 1 1 1 0;0 0 0 1 1;0 0.2 10 1 0];
nprel=5;
kprop=[ones(1,8),2 3]
%----- Boundary conditions
% Cauchy Neumann conditions
kcond=zeros(1,ndlt);vcond=zeros(1,ndlt);
%----- load vector
vfcg=zeros(ndlt,1);
%----- Exact solution
disp('---exact solution : vcor, vsol');disp(vcor(1:2:nnt,1)');
disp(11+log(5./vcor(1:2:nnt,1)'));
disp('--- flux qx---at the nodes=1/r');
disp(1./vcor(1:2:nnt,1));
```

Data file with Q4: 'dthdiskq4.m'

```
nomtype=['th_q4 ';'th_l2b']
ndln=1;ndle=4;
%----- Reading of the coordinates x y
rad=pi/30;    % angle of the sector of the disk
vcor=[1 0.0;  cos(rad) sin(rad);  2 0.0;          2*cos(rad) 2*sin(rad);      3 0;
      3*cos(rad) 3*sin(rad); 4 0;  4*cos(rad) 4*sin(rad); 5 0;  5*cos(rad) 5*sin(rad)];
%----- Reading of the connectivities / elementary type
kcone=[1 3 4 2; 3 5 6 4; 5 7 8 6; 7 9 10 8; 1 2 0 0; 9 10 0 0];
nnt=size(vcor,1);nnel=size(kcone,2);
nelt=size(kcone,1);
ndlt=ndln*nnt;
ktypel=[1 1 1 1 2 2 ];
%----- Properties
```

```
% T3:      c d1 d2 thickness fv
% L2b :    c alfa T0 thickness fs
% type 2: qn=-1;type 3: alfa(T-10)
vprelg=[0 1 1 1 0; 0 0 0 1 1; 0 0.2 10 1 0];
nprel=5;
kprop=[1 1 1 2 3]
%----- Boundary conditions
% Cauchy Neumann conditions
kcond=zeros(1,ndlt);vcond=zeros(1,ndlt);
%----- load vector
vfcg=zeros(ndlt,1);
%----- Exact solution
disp('---exact solution: vcor, vsol');disp(vcor(1:2:nnt,1)');
disp(11+log(5./vcor(1:2:nnt,1)'));
disp('--- flux qx---at the nodes =1/r');
disp(1./vcor(1:2:nnt,1)');
```

- Finite element results:

The calculation with the T3 mesh is executed by successively typing, below the Matlab[®] invite ('>>') :

```
>> fdata='dthdiskt3'
```

or, for the Q4 mesh:

```
>> fdata='dthdiskq4'
```

followed by:

```
>> blin
```

The solutions are identical for both meshes:

$r(m)$	1	2	3	4	5
$T_{ex}(r) ({}^{\circ}C)$	12.6094	11.9162	11.5108	11.2231	11
$T_{FEM}(r) ({}^{\circ}C)$	12.5724	11.9067	11.5072	11.2219	11

The numerical solution is slightly less than the exact solution on the internal nodes and identical on the external nodes. This difference arises from the fact that the incoming flux (to which the temperature increase is due) calculated by the model is overestimated in relation to the exact flux. It is integrated on a rectilinear truss element and not on an arc of a disk whose length is slightly greater. Thus, we have the following values for incoming and outgoing fluxes:

$$|\phi_{\text{incoming}}^{\text{exact}}| = |\phi_{\text{outgoing}}^{\text{exact}}| = 0.1047W, \quad |\phi_{\text{incoming}}^{\text{numerical}}| = |\phi_{\text{outgoing}}^{\text{numerical}}| = 0.1041W.$$

c) Heat transfer in a square plate (Poisson)

Here, we wish to determine the stationary and transient solutions of a problem of heat exchange in a square plate whose sides measure 2 m in length, subjected to a heat source $f_v = 1 W / m^2$. (see Figure 6.4). The conductivities in relation to x and y are $k_1 = k_2 = 1 W / m - {}^\circ C$.

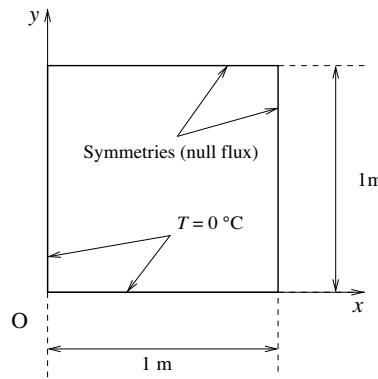


Figure 6.4. Poisson problem over a quarter of the domain

The boundary conditions are:

$$T(x, y, t) = 0, \text{ on the contour of the domain.}$$

For reasons of symmetry, the calculation is only performed on one quarter of the domain.

To deal with the non-stationary problem, we consider the following initial condition:

$$T(x, y, t = 0) = \sin\left(\pi \frac{x}{2}\right) \times \sin\left(\pi \frac{y}{2}\right).$$

The mesh is identical for both cases, and comprises eight T3 elements and nine nodes.

Data file with: dpoissont3.m

```
nomtype=['th_t3'];
ndln=1;
nx=10; ny=5; % numbers of nodes in relation to x and y
```

```

dx=1./(nx-1); dy=1./(ny-1);
vcor=zeros(nx*ny,2);
ni=0;
for j=1:ny
    for i=1:nx
        ni=ni+1; vcor(ni,1)=(i-1)*dx; vcor(ni,2)=(j-1)*dy;
    end
end
%----- Reading of the connectivities/elementary type
kconec=zeros((nx-1)*2*(ny-1),3);
ne=0;
for j=1:ny-1
    for i=1:nx-1
        ne=ne+1; kconec(ne,:)=[(j-1)*nx+i j*nx+i+1 j*nx+i];
        ne=ne+1; kconec(ne,:)=[(j-1)*nx+i (j-1)*nx+i+1 j*nx+i+1];
    end
end
nnt=size(vcor,1);nnel=size(kconec,2);
nelt=size(kconec,1);
ndlt=ndln*nnt;
ktypel=ones(1,nelt); % Single type of element
%----- Properties
% ro=1, k1=k2=1, thickness=1, fv=1
vprelg=[1 1 1 1];nprel=5;
kprop=ones(1,nelt); % Single type of properties
%----- Boundary conditions
% Dirichlet conditions on the sides (x=0) and (y=0)
kcond=zeros(1,ndlt);
vcond=zeros(1,ndlt);
for i=1:nx
    kcond(1,i)=1;
end
for j=1:ny
    kcond(1,(1+(j-1)*nx))=1;
end
%----- Initial solution and load vector
vfcg=zeros(ndlt,1);
vsol0=sin(pi*vcor(:,1)/2).*sin(pi*vcor(:,2)/2);

```

```
%----- Parameters for temporal scheme
dt=1/100 ; npas=10; alfa=1;
```

- MEF results:

The data file is initialized by:

```
>> fdata='dpoisont3'
```

The stationary calculation is executed by typing the command:

```
>> blin
```

The stationary solution calculated at the nodes is:

$$\langle T \rangle = \langle 0 \ 0 \ 0 \ 0 \ 0.1771 \ 0.2292 \ 0 \ 0.2292 \ 0.3125 \rangle.$$

The temporary calculation is executed by typing the command:

```
>> btemp
```

The temporary solution calculated using an implicit scheme, after 10 time-steps of 0.01 second, is:

$$\langle T \rangle = \langle 0 \ 0 \ 0 \ 0 \ 0.3729 \ 0.5 \ 0 \ 0.5 \ 0.73 \rangle.$$

- Comment: the data file does not need to be changed, whether or not the calculation is stationary.

6.6.2 Planar elastic problems

a) Mechanical patch test with T3 and Q4 elements

The domain is a square with 1 m long sides. Here, we wish to carry out a mechanical patch test (see section 4.1.3) for two meshes, respectively, comprising T3 elements ('el_t3') and Q4 elements ('el_q4'). The solution imposed on both components of the field of displacement is:

$$u(x, y) = 1 + \frac{x}{3} + \frac{y}{5}, \quad v(x, y) = 1 + \frac{4x}{5} + \frac{2y}{3}.$$

The stress boundary conditions are:

$f_{sx} = -3, f_{sy} = -1.5$ on the segment 1–2;
 $f_{sx} = +3, f_{sy} = +1.5$ on the segment 3–4;
 $f_{sx} = -2, f_{sy} = -1.5$ on the segment 4–1;
 $f_{sx} = +2, f_{sy} = +1.5$ on the segment 2–3;

The problem is supposed to be of plane stress type. The boundary conditions are imposed by truss elements ('el_l2b'). Three Dirichlet conditions are imposed to eliminate the three rigid body modes:

$$u_1 = 1, v_1 = 1 \text{ and } v_2 = \frac{9}{5}.$$

Young's modulus and Poisson's coefficient are:

$$E = \frac{15}{4} N/m^2, \nu = 0.25.$$

Here, the problem employs two meshes, respectively, comprising four T3 elements ('el_t3') for five nodes, and five Q4 elements ('el_q4') for eight nodes (see Figure 6.2). The integration of the boundary conditions requires four truss elements ('el_l2b') for both meshes.

Data file with T3: 'delpatcht3.m'

```
nomtype=['el_t3 ';'el_l2b']
ndln=2;
%----- Reading of the coordinates x y
vcor=[0.0 0.0; 1.0 0.0; 1.0 1.0; 0.0 1.0; 0.35 0.8];
%----- Reading of the connectivities/elementary type
% Five Q4 elements and four L2b elements
kconec=[1 2 5; 2 3 5; 3 4 5; 4 1 5; 1 2 0; 2 3 0; 3 4 0; 4 1 0];
nnt=size(vcor,1);nnel=size(kconec,2);
nelt=size(kconec,1);
ndlt=ndln*nnt;
ktypel=[1 1 1 2 2 2];
%----- Properties
% Q4:    ro,E,nu,ind cp/dp,thickness,fvx,fvy
% l2b:    ro,0,0,0,thickness,fsx,fsy
vprelg=[ 1 15/4 0.25 0 1 0 0; 0 0 0 0 1 -1.5 -3.0; 0 0 0 0 1 2.0 1.5;
          0 0 0 0 1 1.5 3.0; 0 0 0 0 1 -2.0 -1.5];
```

```

nprel=7;
kprop=[1 1 1 2 3 4 5];% 5 types of properties
%----- Boundary conditions
% three kinematic conditions: node 1: u=1 v=1; node 2 v=9/5
kcond=zeros(1,ndlt);kcond([1 2 4])=1;
vcond=zeros(1,ndlt);vcond([1 2 4])=[ 1 1 9/5];
%----- load vector
vfcg=zeros(ndlt,1);

```

Data file with Q4: 'delpatchq4.m'

```

nomtype=['el_q4 ';'el_l2b']
ndltn=2;
%----- Reading of the coordinates x y
vcor=[0.0 0.0;1.0 0.0;1.0 1.0;0.0 1.0;0.3 0.3;0.6 0.4;0.7 0.66;0.35 0.8];
%----- Reading of the connectivities / elementary type
% Five Q4 elements and four L2b elements
kccone=[1 5 8 4;1 2 6 5;2 3 7 6;3 4 8 7;5 6 7 8;1 2 0 0;2 3 0 0;3 4 0 0;4 1 0 0];
nnt=size(vcor,1);nnel=size(kccone,2);
nelt=size(kccone,1);
ndltn=ndltn*nnt;
ktypel=[1 1 1 1 1 2 2 2];
%----- Properties
% Q4:      ro, E, nu, ind CP/DP, thickness,fvx,fvy
% l2b:      ro, 0, 0, 0, thickness,fsx,fsy
vprelg=[1 15/4 0.25 0 1 0 0;0 0 0 0 1 -1.5 -3.0;0 0 0 0 1 2.0 1.5;
        0 0 0 0 1 1.5 3.0;0 0 0 0 1 -2.0 -1.5];
nprel=7;
kprop=[1 1 1 1 2 3 4 5];% 5 types of properties
%-----Boundary conditions
% three kinematic conditions: node 1: u=1 v=1; node 2 v=9/5
kcond=zeros(1,ndlt);kcond([1 2 4])=1;
vcond=zeros(1,ndlt);vcond([1 2 4])=[ 1 1 9/5];
%----- load vector
vfcg=zeros(ndlt,1);

```

- Finite element method results:

The calculation with the T3 mesh is executed by successively typing:

```
>> fdata='delpatcht3'
```

or for the Q4 mesh:

```
>> fdata='delpatchq4'
```

followed by:

```
>> blin
```

The solution at the nodes for the T3 mesh is:

$$\langle u \rangle = \langle 1 \ 1.333 \ 1.533 \ 1.2 \ 1.276 \rangle,$$

$$\langle v \rangle = \langle 1 \ 1.8 \ 2.467 \ 1.667 \ 1.813 \rangle.$$

The solution at the nodes for the Q4 mesh is:

$$\langle u \rangle = \langle 1 \ 1.333 \ 1.533 \ 1.2 \ 1.16 \ 1.28 \ 1.365 \ 1.276 \rangle,$$

$$\langle v \rangle = \langle 1 \ 1.8 \ 2.467 \ 1.667 \ 1.44 \ 1.746 \ 2 \ 1.813 \rangle.$$

The obtained values confirm that the patch tests have been successful.

b) Slanting plaque

In this case, we wish to study the deformation of a slanting plaque, fixed at one of its extremities and subject to an identical linear force at the opposite extremity. The dimensions are given in Figure 6.5:

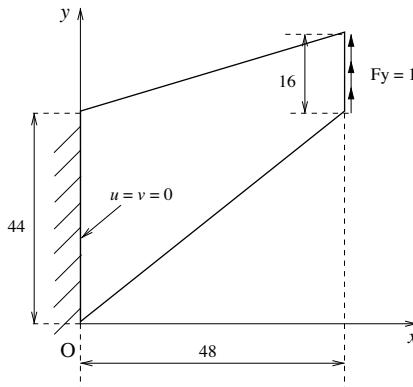


Figure 6.5. Slanting plaque

For the properties:

$$E = 1 \text{ N/m}^2, \nu = 0.3,$$

and a uniform thickness, the horizontal and vertical components of the displacement at its free end are given by:

$$u(x = 48) = 0.967, \quad v(x = 48) = 380.$$

Here, the mesh is made up of (4×4) Q4 elements. The integration of the stress requires four truss elements ('el_l2b').

Data file with Q4: 'd4biaisq4.m'

```

nomtype=['el_q4 ';'el_l2b']
ndln=2;ndle=8;
%----- Reading of the coordinates x y
vcor=[    0   0;     0 11;      0 22;      0 33;      0 44;
          12 11;    12 20.25;   12 29.5;   12 38.75;  12 48;
          24 22;    24 29.5;   24 37;    24 44.5;   24 52;
          36 33;    36 38.75;  36 44.5;  36 50.25; 36 56;
          48 44;    48 48;    48 52;    48 56;    48 60 ];
%----- Reading of the connectivities/elementary type
kcone=[1   6   7   2;   2   7   8   3;   3   8   9   4;   4   9   10   5
        6   11  12  7;   7   12  13  8;   8   13  14  9;   9   14  15  10
        11  16  17  12;  12  17  18  13;  13  18  19  14;  14  19  20  15
        16  21  22  17;  17  22  23  18;  18  23  24  19;  19  24  25  20
        21  22  0   0;   22  23  0   0;   23  24  0   0;   24  25  0   0];
nnt=size(vcor,1);nnel=size(kcone,2);
nelt=size(kcone,1);
ndlt=ndln*nnt;
ktypel=[ones(1,16), 2 2 2 2];
%----- Properties
% Q4:    ro, E, nu, ind cp/dp,thickness,fvx,fvy
% l2b:    ro, 0, 0, 0, thickness,fsx,fsy
vprelg=[1 1 0.3 0 1 0 0;  0 0 0 0 1 0 1];
nprel=7;
kprop=[ones(1,16),2 2 2 2];
%----- Boundary conditions
% kinematic conditions on nodes 1, 4 and 7
kcond=zeros(1,ndlt); kcond([1:10])=1;
vcond=zeros(1,ndlt);
%----- load vector
vfcg=zeros(ndlt,1);

```

- Finite element method results:

The calculation is executed by successively typing:

```
>> fdata='d4biaisq4'
>> blin.
```

For meshes of sizes (2×2) , (4×4) and (10×10) , respectively, the obtained solutions are:

$$\nu_{2 \times 2}(48,52) = 190, \nu_{4 \times 4}(48,52) = 293 \text{ and } \nu_{10 \times 10}(48,52) = 365.$$

c) Bending beam

Here, we deal with the case of a thin beam anchored at one of its extremities and subject to a uniform lineic force at the other extremity (see Figure 6.6).

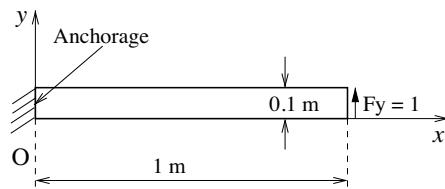


Figure 6.6. Beam anchored at $x = 0$ m subject to stress at $x = 1$ m

The exact arrow at its extremity is given by:

$$\nu_{ex}(x = 1, y) = \frac{P L^3}{3 E I}.$$

For the options $P = 0.1$ N, $L = 1$ m, $E = 10^3$ N/m 2 , we have $\nu_{ex}(x = 1) = 0.4$ m.

Here, the mesh is made up of $(nx \times ny)$ nodes and T3 elements ('el_t3') or Q4 elements ('el_q4'). The integration of the line load requires $(ny - 1)$ truss elements ('el_l2b').

Data file with T3: 'dpoutret3.m'

```
nomtype=['el_t3 ';'el_l2b']
%----- Reading of the coordinates x y
```

```

nx=10; ny=3;% numbers of the nodes in relation to x and y
dx=1./(nx-1); dy=0.1./(ny-1);
vcor=zeros(nx*ny,2);
ni=0;
for j=1:ny
    for i=1:nx
        ni=ni+1; vcor(ni,1)=(i-1)*dx; vcor(ni,2)=(j-1)*dy;
    end
end
%----- Reading of the connectivities/elementary type
kconec=zeros((nx-1)*2*(ny-1)+(ny-1),3);% for T3
ne=0;
for j=1:ny-1
    for i=1:nx-1
        ne=ne+1; kconec(ne,:)=[(j-1)*nx+i j*nx+i+1 j*nx+i];
        ne=ne+1; kconec(ne,:)=[(j-1)*nx+i (j-1)*nx+i+1 j*nx+i+1];
    end
end
%Addition of (ny-1) truss elements
for j=1:ny-1
    ne=ne+1; kconec(ne,[1 2])=[j*nx (j+1)*nx];
end
trace_mail(vcor,kconec)
ndln=2;ndle=8;
nnt=size(vcor,1);nnel=size(kconec,2);nelt=size(kconec,1);ndlt=ndln*nnt;
ktypel=ones(1,nelt);ktypel(nelt-(ny-2):nelt)=2;
%----- Properties
%T3 : ro, E, nu, ind CP/DP,thickness,fvx,fvy
% l2b: ro, 0, 0,0,thickness,fsx,fsy
vprelg=[ 1 1e3 0.3 0 1 0 0;
          0 0 0 0 1 0 1 ];
nprel=7;
kprop=ones(1,nelt);kprop(nelt-(ny-2):nelt)=2;
%----- Boundary conditions
% nodes blocked on the anchored face
kcond=zeros(1,ndlt);kcond([1:2*nx:ndlt])=1;kcond([2:2*nx:ndlt])=1;
vcond=zeros(1,ndlt);
%----- load vector
vfcg=zeros(ndlt,1);

```

Data file with Q4: 'dpoutreq4.m'

```

nomtype=['el_q4 ';'el_l2b']
%----- Reading of the coordinates x y
nx=40; ny=7;% numbers of the nodes in relation to x and y
dx=1./(nx-1);dy=0.1./(ny-1);
vcor=zeros(nx*ny,2);
ni=0;
for j=1:ny
    for i=1:nx
        ni=ni+1;      vcor(ni,1)=(i-1)*dx;      vcor(ni,2)=(j-1)*dy;
    end
end
%----- Reading of the connectivities/ elementary type
kconec=zeros((nx-1)*(ny-1)+ny-1,4);% for Q4
ne=0;
for j=1:ny-1
    for i=1:nx-1
        ne=ne+1;      kconec(ne,:)=[(j-1)*nx+i (j-1)*nx+i+1 j*nx+i+1 j*nx+i];
    end
end
% Addition of (ny-1) truss elements
for j=1:ny-1
    ne=ne+1;  kconec(ne,[1 2])=[j*nx (j+1)*nx];
end
trace_mail(vcor,kconec)
ndln=2;ndle=8;nnt=size(vcor,1);nnel=size(kconec,2);nelt=size(kconec,1);ndlt=ndln*nnt;
ktypel=ones(1,nelt);ktypel(nelt-(ny-2):nelt)=2;
% Q4 :    ro, E, nu, ind CP/DP,thickness,fvx,fvy
% l2b :    ro, 0, 0,0,thickness,fsx,fsy
vprelg=[ 1 1e3 0.3 0 1 0 0;
          0 0 0 0 1 0 1 ];
nprel=7;
kprop=ones(1,nelt);kprop(nelt-(ny-2):nelt)=2;
%----- Boundary conditions
% nodes blocked on the anchored face

```

```

kcond=zeros(1,ndlt);kcond([1:2★nx:ndlt])=1;kcond([2:2★nx:ndlt])=1;
vcond=zeros(1,ndlt);
%----- load vector
vfcg=zeros(ndlt,1);

```

- Finite element results:

Depending on the type of mesh, the calculation is executed by successively typing:

```
>> fdata='dpoutret3'
```

or:

```
>> fdata='dpoutreq4'
```

followed by:

```
>> blin
```

The solution obtained for a mesh of size $nx=ny=3$ is:

$$\nu^{T3}(48, 0.025) = 0.0149 \text{ m}, \quad \nu^{Q4}(48, 0.025) = 0.0377 \text{ m}.$$

The use of meshes with a greater number of elements enables us to reduce the difference between the numerical solution and the exact solution. The convergence curves for the T3 and Q4 elements are given in Figure 6.7.

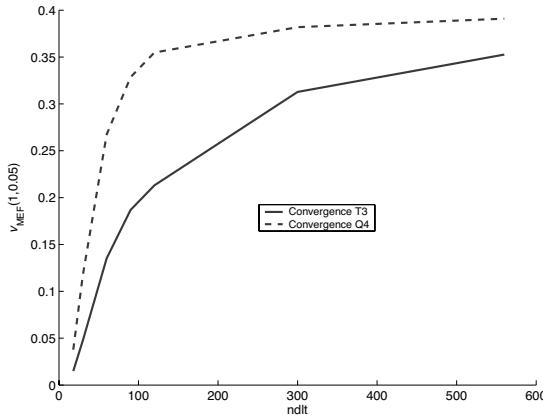


Figure 6.7. Convergence curves: bending beam

6.6.3 Fluid flow problems

a) Patch test with Q4 elements

The domain is a square with sides measuring 1 m in length. We carry out a mechanical patch test (see section 4.1.3) for a mesh composed of Q4 elements ('ns_q4') and of size (4×4) .

The solution imposed on the two components of the velocity field is:

$$u(x, y) = y, \quad v(x, y) = x.$$

This solution is imposed at all the nodes of the contour. With a time-step of 1 s and three iterations per round of calculation, two rounds are sufficient to obtain a converged solution. The solution is piloted by the boundary conditions. The patch test is successful if the solutions at the internal nodes satisfy the solution field shown above.

Data file with Q4: 'dpatchns.m'

```

nomtype=['ns_q4']
%----- Reading of the coordinates x y
nx=3; ny=3;% numbers of the nodes in relation to x and y
xl=1; yl=1;
dx=xl./(nx-1);dy=yl./(ny-1);
vcor=zeros(nx*ny,2);
ni=0;
for j=1:ny
    for i=1:nx
        ni=ni+1;      vcor(ni,1)=(i-1)*dx;      vcor(ni,2)=(j-1)*dy;
    end
end
%----- Reading of the connectivities/elementary type
kconec=zeros((nx-1)*(ny-1),4);% for Q4
ne=0;
for j=1:ny-1
    for i=1:nx-1
        ne=ne+1;      kconec(ne,:)=[(j-1)*nx+i (j-1)*nx+i+1 j*nx+i+1 j*nx+i];
    end
end
trace_mail(vcor,kconec);
nnt=size(vcor,1);nnel=size(kconec,2);ndln=2; nelt=size(kconec,1); ndlt=ndln*nnt;
ktypel=[ones(1,nelt)]; % single type of elements
% Remark: add the boundary elements in case of stress from pressure
%----- Properties

```

```

vprelg=[0 2 1e9 0]; % ro,nu lamda pressure
kprop=[ones(1,nelt)];
%----- Boundary conditions: Dirichlet
vcond=zeros(1,ndlt); kcond=zeros(1,ndlt);
% Faces y=0;y=1
for i=1:nx
    kcond((i-1)*ndl+1)=1;
    vcond((i-1)*ndl+1)=[0 (i-1)*dx]; % u=0,v=x
    kcond((i-1)*ndl+(ny-1)*nx*ndl+1)=1;
    vcond((i-1)*ndl+(ny-1)*nx*ndl+1)=[yl (i-1)*dx]; % u=vl v=x
end
% Faces x=0 x=1
for i=1:ny
    kcond((i-1)*nx*ndl+1)=1;
    vcond((i-1)*nx*ndl+1)=(i-1)*dy; % u=y v=0;
    kcond((i*nx-1)*ndl+1)=1;
    vcond((i*nx-1)*ndl+1)=[(i-1)*dy xl];
end
%----- Piloting by boundary conditions
vsol0=zeros(ndlt,1); vfcg=zeros(ndlt,1);
dt=1;npas=2;alfa=1;npilot=1;niter=3;

```

- Finite element method results

The calculation is executed by successively typing:

```

>> fdata='dpatchns'
>> bnlm

```

The solutions calculated for the two components (u, v) of the velocity field, for the nodes located at (x, y) , are displayed in (m/s) in the table below:

	$x = 0.00 \text{ m}$	$x = 0.25 \text{ m}$	$x = 0.50 \text{ m}$	$x = 0.75 \text{ m}$	$x = 1.00 \text{ m}$
$y = 1.00 \text{ m}$	(1.00, 0.00)	(1.00, 0.25)	(1.00, 0.50)	(1.00, 0.75)	(1.00, 1.00)
$y = 0.75 \text{ m}$	(0.75, 0.00)	(0.75, 0.25)	(0.75, 0.50)	(0.75, 0.75)	(0.75, 1.00)
$y = 0.50 \text{ m}$	(0.50, 0.00)	(0.50, 0.25)	(0.50, 0.50)	(0.50, 0.75)	(0.50, 1.00)
$y = 0.25 \text{ m}$	(0.25, 0.00)	(0.25, 0.25)	(0.25, 0.50)	(0.25, 0.75)	(0.25, 1.00)
$y = 0.00 \text{ m}$	(0.00, 0.00)	(0.00, 0.25)	(0.00, 0.50)	(0.00, 0.75)	(0.00, 1.00)

The obtained values confirm that the patch test has been successful.

b) Poiseuille flow

Here, we wish to deal with the case of a Poiseuille flow generated by a pressure difference. The domain is a square, with sides measuring 4 m in length, associated with the boundary conditions illustrated in Figure 6.8:

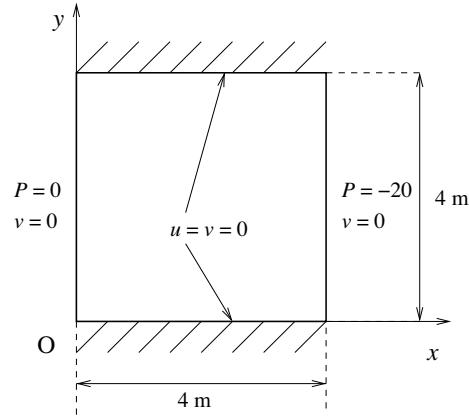


Figure 6.8. Poiseuille flow

The top and bottom walls no-slip boundary condition, and the pressure difference between entering the domain ($x=0$) and exiting it ($x=4\text{m}$) is 20 Pa. This difference is imposed with truss elements ('ns_12b'). The viscosity is $\mu = 2 \text{ m/s}^2$. The exact solution, which depends only on y , is given by:

$$u_{ex}(y) = \frac{3}{4}(4y - y^2).$$

Two rounds of calculation with a time-step of 1s and three iterations per round of calculation are sufficient to obtain a converged solution. Here, the mesh comprises (4×4) Q4 elements. The integration of the pressure stress requires (2×4) truss elements ('el_12b').

Data file with Q4: 'dpoiseuille.m'

```
nomtype=['ns_q4 ';'ns_12b'] % two types of elements
%----- Reading of the coordinates x y
xl=4;yl=4;% width and height of the domain
nx=9;ny=9; % numbers of the nodes in relation to x and y
dx=xl./(nx-1);dy=yl./(ny-1);
vcor=zeros(nx*ny,2);
```

```

ni=0;
for j=1:ny
    for i=1:nx
        ni=ni+1;      vcor(ni,1)=(i-1)*dx;      vcor(ni,2)=(j-1)*dy;
    end
end
%----- Reading of the connectivities/elementary type
kconec=zeros((nx-1)*(ny-1)+2*(ny-1),4);% for Q4
ne=0;
for j=1:ny-1
    for i=1:nx-1
        ne=ne+1;      kconec(ne,:)=[(j-1)*nx+i (j-1)*nx+i+1 j*nx+i+1 j*nx+i];
    end
end
% 2*(ny-1) boundary elements
for j=1:ny-1
    ne=ne+1;  kconec(ne,[1 2])=[(j-1)*nx+1 j*nx+1];
end
for j=1:ny-1
    ne=ne+1;  kconec(ne,[1 2])=[j*nx (j+1)*nx];
end
ktypel=[ones(1,(nx-1)*(ny-1)),2*ones(1,2*(ny-1))];% 2 types of elements
trace_mail(vcor,kconec);
nnt=size(vcor,1);nnel=size(kconec,2);ndln=2; nelt=size(kconec,1); ndlt=ndln*nnt;
% Remark: add the boundary elements in the case of stress from pressure
%----- Properties
vprelg=[0 2 1e9 0; 0 2 1e6 -20;0 2 1e6 0] ;% ro,nu, lamda, pressure
kprop=[ones(1,(nx-1)*(ny-1)),2*ones(1,(ny-1)),3*ones(1,(ny-1))];
%---- Boundary conditions: Dirichlet
vcond=zeros(1,ndlt);kcond=zeros(1,ndlt);
% u=v=0 on the top and bottom walls
for i=1:nx
    kcond((i-1)*ndln+[1 2])=1;
    kcond((i-1)*ndln+(ny-1)*nx*ndln+[1 2])=1;
end
kcond( 2:ndln:ndlt)=1; % component v=0 on all the nodes
%---- Initial solution and load vector
vfcg=zeros(ndlt,1);vsol0=zeros(ndlt,1);

```

```
%  
disp(' exact solution in relation to y')  
dp=(vprelg(2,4)-vprelg(3,4))/vprelg(1,2);  
u=-.5*(dp/xl)*(yl*vcor([1:ny],2)-vcor([1:ny],2).^2)  
%----- Time parameters: piloting by pressure  
dt=1;npas=2;alfa=1;npilot=0;niter=3;
```

- Finite element method results:

The calculation is executed by successively typing:

```
>> fdata='dpoiseuille'  
>> bnlm
```

The obtained solution is identical to the exact solution:

$$u(x, y = 1) = 3.75 \text{ m / s}, \quad u(x, y = 2) = 5 \text{ m / s}, \quad u(x, y = 3) = 3.75 \text{ m / s}.$$

c) Lid-driven cavity flow

The problem in question is a lid-driven cavity with uniform sides. This is delimited by three no-slip walls (the bottom and the two vertical walls). A velocity condition is imposed on the upper boundary (see Figure 6.9).

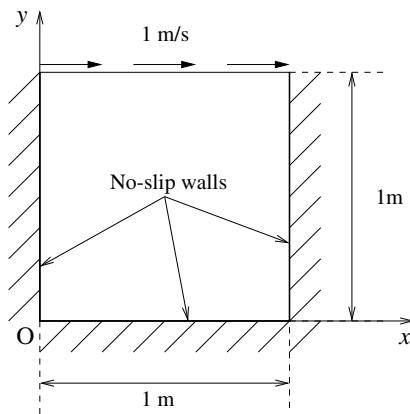


Figure 6.9. Lid-driven cavity

Two rounds of calculation are required, with a time-step of 1s and three iterations per round of calculation. The solution is piloted by the boundary conditions. Here, the mesh comprises (7×7) Q4 elements.

Data file with Q4: 'dcavite.m'

```

nomtype=['ns_q4'];
%----- Reading of the coordinates x y
xl=1;yl=1; % width and height
nx=8;ny=8; % numbers of the nodes in relation to x and y
dx=xl./(nx-1);dy=yl./(ny-1);
vcor=zeros(nx*ny,2);
ni=0;
for j=1:ny
    for i=1:nx
        ni=ni+1; vcor(ni,1)=(i-1)*dx; vcor(ni,2)=(j-1)*dy;
    end
end
%----- Reading of the connectivities / elementary type
kcone=zeros((nx-1)*(ny-1),4); % for Q4
ne=0;
for j=1:ny-1
    for i=1:nx-1
        ne=ne+1; kcone(ne,:)=[(j-1)*nx+i (j-1)*nx+i+1 j*nx+i+1 j*nx+i];
    end
end
trace_mail(vcor,kcone);
nnt=size(vcor,1);nnel=size(kcone,2);ndln=2; nelt=size(kcone,1); ndlt=ndln*nnt;
ktypel=ones(1,nelt); % single type of element
% Remark: add boundary elements in the case of stress from pressure
%----- Properties
vprelg=[0 .001 1e6 0]; % ro,nu lamda pressure
kprop=[ones(1,nelt)];
%----- Boundary conditions: Dirichlet
vcond=zeros(1,ndlt);
kcond=zeros(1,ndlt);
% faces (x, y=0) and (x, y=1)
for i=1:nx
    kcond((i-1)*ndln+[1 2])=1;
    kcond((i-1)*ndln+(ny-1)*nx*ndln+[1 2])=1;
    vcond((i-1)*ndln+(ny-1)*nx*ndln+[1 ])=1; % velocity =1 on y=1
end
% faces (x=0, y) and (x=1, y)
for i=1:ny
    kcond((i-1)*nx*ndln+[1 2])=1;
    kcond((i*nx-1)*ndln+[1 2])=1;
end
%----- Initial solution and load vector
vsol0=zeros(ndlt,1);
vfcg=zeros(ndlt,1);
%----- Time parameters: piloting by the boundary conditions
dt=1; npas=2; alfa=1; npilot=1; niter=3;

```

- Finite element method results:

The calculation is executed by successively typing, below the Matlab[©] invite (“>>”):

```
>> fdata='dcavite'  
>> bnlin
```

The calculated solutions are represented by the velocity field illustrated in Figure 6.10. The table below gives some values (in m/s) of the two velocity components at nodes of the mesh.

	$x = 0.285 \text{ m}$	$x = 0.571 \text{ m}$	$x = 0.857 \text{ m}$
$y = 0.857 \text{ m}$	$(2.64 \cdot 10^{-1}, 5.17 \cdot 10^{-2})$	$(3.91 \cdot 10^{-1}, 2.27 \cdot 10^{-3})$	$(1.52 \cdot 10^{-1}, -1.52 \cdot 10^{-1})$
$y = 0.571 \text{ m}$	$(-6.26 \cdot 10^{-2}, 1.54 \cdot 10^{-1})$	$(-1.16 \cdot 10^{-1}, -1.32 \cdot 10^{-2})$	$(-4.50 \cdot 10^{-2}, -2.09 \cdot 10^{-1})$
$y = 0.285 \text{ m}$	$(-6.38 \cdot 10^{-2}, 5.36 \cdot 10^{-2})$	$(-1.06 \cdot 10^{-1}, -1.53 \cdot 10^{-2})$	$(-2.62 \cdot 10^{-2}, -5.19 \cdot 10^{-2})$

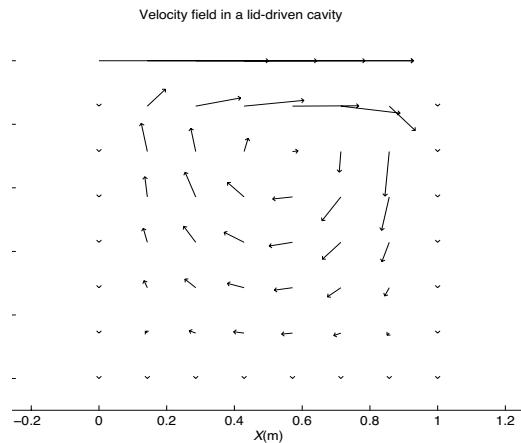


Figure 6.10. Velocity field in a cavity

d) Flow by stream function

For supposedly incompressible, non-viscous (i.e. irrotational) flows, it is possible to simplify the calculation of the velocity field for a confined flow, using the stream function $\psi(x, y)$ defined by:

$$u(x, y) = \frac{\partial \psi}{\partial y} \quad \text{and} \quad v(x, y) = -\frac{\partial \psi}{\partial x}.$$

$u(x, y)$ and $v(x, y)$ are the two components in the plane (x, y) of the velocity field.

The irrotational flow hypothesis enables us to obtain the equilibrium equation for the problem:

$$\Delta\psi(x, y) = 0,$$

This equation is similar to the stationary thermal problem. The streamlines (identical to the trajectories in a stationary regime) are defined by the curves $\psi = \text{constant}$. The domain considered for the study is shown in Figure 6.11. Because the walls are impermeable, each of them is associated with Dirichlet boundary conditions of values denoted by ψ_1 and ψ_2 . The condition of flow at the velocity U_{entry} upon entering the domain is written:

$$U_{\text{entry}} \approx \frac{\psi_2 - \psi_1}{H}.$$

The values of the boundary conditions being defined to a near constant, we choose $\psi_2 = 0$. For $U_{\text{entry}} = 8 \text{ m/s}$ and an aperture $H = 0.125 \text{ m}$, we have $\psi_2 = 1$.

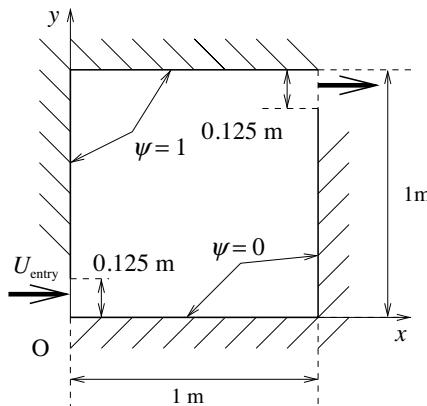


Figure 6.11. Stream function flow in a cavity

The data file below is used to discuss solution obtained with a mesh comprising $(nx \times ny)$ nodes with T3 elements ('th_t3') or Q4 elements ('th_q4') by simply modifying the first line.

Data file with T3 or Q4: 'dcourant.m'

```
%nomtype=['th_t3'] % calculation with T3
nomtype=['th_q4'] % calculation with Q4
%---- Reading of the coordinates x y
xl=1;yl=1; % width and height of the domain
nx=9;ny=9; % numbers of nodes in relation to x and y
```

```

vcor=zeros(nx*ny,2);kconec2=zeros((nx-1)*(ny-1),4);
for i=1:nx
    for j=1:ny
        vcor((i-1)*ny+j,:)=[(i-1)*xl/(nx-1),(j-1)*yl/(ny-1)];
    end
end
%----- Reading of the connectivities/elementary type
for i=1:nx-1
    for j=1:ny-1
        ie=(nx-1)*(i-1)+j;in=nx*(i-1)+j;
        kconec2(ie,:)=[in,in+ny,in+ny+1,in+1];
    end
end
% for a mesh with T3, the quadrilaterals are divided into two triangles
if(nomtype=='th_t3')
    kconec=zeros(2*(nx-1)*(ny-1),3);
    kconec([1:(nx-1)*(ny-1)],:)=kconec2(:,[1 2 3]);
    kconec([(nx-1)*(ny-1)+1:2*(nx-1)*(ny-1)],:)=kconec2(:,[1 3 4]);
else
    kconec=kconec2;
end
nnt=size(vcor,1);nnel=size(kconec,2);ndln=1; nelt=size(kconec,1); ndlt=ndln*nnt;
ktypel=[ones(1,nelt)]; % single type of element
trace_mail(vcor,kconec);
%---- Properties
vprelg=[0 1 1 1 0]; % c k1 k2 thickness fv
kprop=[ones(1,nelt)];
%---- Boundary conditions: Dirichlet
vcond=zeros(1,ndlt);kcond=zeros(1,ndlt);
% we impose phi=0 on the nodes of the segments (y=0) and (x=1)
kcond(1,find(vcor(:,2)==0))=1;
kcond(1,find(vcor(:,1)==xl))=1;
% we impose phi=1 on the nodes of the segments (y=1) and (x=0)
kcond(1,find(vcor(:,2)==yl))=1; vcond(1,find(vcor(:,2)==yl))=1;
kcond(1,find(vcor(:,1)==0))=1; vcond(1,find(vcor(:,1)==0))=1;
vcond(1,1)=0; % correction for the node (x=0,y=0)
%---- load vector
vfcg=zeros(ndlt,1);

```

- Finite element method results:

The calculation is executed by successively typing:

```

>> fdata='dcourant'
>> blin

```

For a mesh with $nx=ny=9$, the isovalues of $\psi(x, y)$ representing the streamlines of the flow are illustrated in Figure 6.12. The results are identical both with T3 and Q4 elements. The table below shows some values extracted from the nodes of the mesh.

	$x = 0.25 \text{ m}$	$x = 0.50 \text{ m}$	$x = 0.75 \text{ m}$
$y = 0.75 \text{ m}$	0.862	0.719	0.5
$y = 0.50 \text{ m}$	0.719	0.5	0.28
$y = 0.25 \text{ m}$	0.5	0.28	0.137

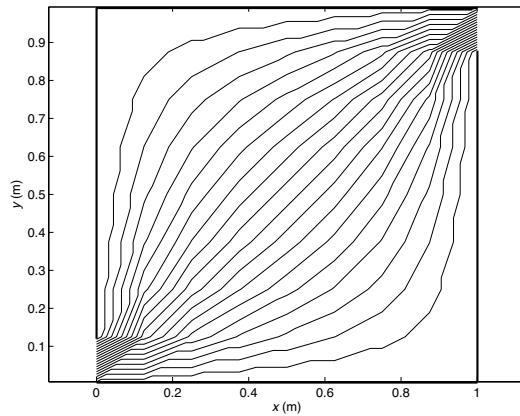


Figure 6.12. Streamlines of the flow

- APPENDIX -

SPARSE SOLVER

- Sofiane Hadji -

This section offers an introduction to the principles of a solver based on sparse coding. The goal here is to provide the interested reader with a general approach and an understanding of the strategy inherent in that approach, rather than a complete and directly exploitable tool.

7.0 Introduction

The finite element method solves linear systems in the form $[K]\{U\} = \{F\}$, where $[K]$ is the stiffness matrix, $\{U\}$ the (discretized) displacement and $\{F\}$ the generalized forces. An implicit nonlinear analysis and the search for a modal basis often rely on a “basic operation”: the solving of linear systems. In all these cases, the matrices $[K]$ very often have a particular topological characteristic that it would be useful to exploit: the matrices are **sparse**, meaning that they contain a great many null terms. In order to take advantage of this property, we must:

- Use appropriate storage techniques, because the number of operations needed to solve the system $[K]\{U\} = \{F\}$ is directly linked to the number of terms stored from the matrix. To store the full matrix $[K]$ (i.e. all the terms), 8 MB of memory is required when the size of the matrix is $n = 10^3$, and when $n = 10^5$, 80 GB is needed.
- Precede the phase of numerical factorization with a renumbering of the degrees of freedom, intended to permute the rows and columns of the matrix $[K]$, so that numerical factorization of the permuted matrix creates far fewer non-null terms and gives rise to far fewer calculations than if the same operation were performed on the original matrix.

There are many techniques for storage and renumbering. The most commonly used are the techniques of storage in *band* matrices and *skyline, sparse storage* (see Chapter 5 and [DUF 90] for skyline storage) and the renumbering techniques

of *Cuthill–McKee*, *minimum degree* and *nested dissection* [CUT 69; TIN 67; GEO 89].

Today, because of their efficiency and robustness, the tendency is to lean toward *sparse* storage and *minimum degree* renumbering.

7.1 Methodology of the sparse solver

Whether we wish to use iterative methods (conjugate gradient, GMRES, etc.) [SAA 86] or direct methods (such as the Gaussian elimination) to solve large linear systems, it is very useful to store the matrix $[K]$ using sparse coding so as to avoid storing null terms. Iterative methods only use matrix-by-vector products, for which the null terms can easily be ignored. These methods are used when there is only one linear system to be solved, so as to avoid a costly factorization of the matrix, or when the systems needing to be solved are so large that a direct method becomes too greedy in terms of memory space.

A direct resolution technique (a solver) based on sparse storage performs the following steps:

1. Assembly and storage of the global matrix in sparse form;
2. Search for the optimal permutation matrix $[P]$ using the *minimum degree* algorithm;
3. Reorganization of the matrix and the pointers in accordance with the permutation $[P]$;
4. Symbolic factorization of the matrix;
5. Numerical factorization of the matrix;
6. Solution by descent and ascent.

For each of the above steps, there are many algorithms that can be used. Each stage has its own numerical characteristics:

- Stages 2, 3 and 4 use the graphs associated with the matrices and consequently involve only integers. The other stages use real numbers.
- Stage 5 takes up the most time in terms of calculation, while stage 6 is the quickest.
- The algorithm used in stage 2 is completely independent of that employed in stage 5. On the other hand, the algorithms used for stages 4 and 5 are similar.

- For simple systems of equations – particularly when the matrix is symmetrical and positive definite – all the above stages can be dealt with independently of one another. For the general case of non-symmetrical systems, the solver can combine stages 4 and 5 or stages 2, 3, 4 and 5, so that the values of the terms in the matrix play a part in determining the optimal permutation of the rows and columns.

In what follows, we introduce and detail the various stages in the form of algorithms.

7.1.1 ASSEMBLY OF MATRICES IN SPARSE FORM: ROW-BY-ROW FORMAT

The non-null terms of the global matrix are stored successively row-by-row in a single vector vkg . The information necessary to reconstruct the global matrix from vkg is stored in two auxiliary vectors: $jvkg$ and $ivkg$, whose components are numbered, respectively, from (1 to nnz) and (1 to $neq + 1$). nnz is the number of non-null terms, while neq defines the size of the matrix. $jvkg(i)$ indicates the number of the column of the i th non-null term in the matrix, and $ivkg(i)$ indicates the position in vkg of the first non-null term on row i . A term K_{ij} in the matrix is non-null and occupies the position $ipos$, if and only if:

$$ivkg(i) < ipos < ivkg(i+1) \text{ and } jvkg(ipos) = j.$$

By convention: $ivkg(neq + 1) = nnz + 1$.

The gain in terms of storage is appreciable in comparison to full storage of the matrix. Indeed, instead of storing neq^2 terms (real numbers), only nnz real numbers and $(neq + 1)$ integers are stored. This gain is all the more significant when the matrix $[K]$ is very sparse. This format is called row-by-row format.

EXAMPLE 7.1. Sparse storage of the terms of a symmetrical matrix

Consider the following symmetrical matrix $[K]$:

$$[K] = \begin{bmatrix} 1.6 & 2.1 & 6 & 0 \\ & 3.4 & 0 & 7 \\ & & 7 & 2 \\ sym & & & 7 \end{bmatrix}$$

with $neq = 4$, $nnz = 8$.

The corresponding pointers are:

$$\begin{aligned}\langle vkg \rangle &= \langle 1.6 \quad 2.1 \quad 6 \quad 3.4 \quad 7 \quad 7 \quad 2 \quad 7 \rangle \\ \langle jvkg \rangle &= \langle 1 \quad 2 \quad 3 \quad 2 \quad 4 \quad 3 \quad 4 \quad 4 \rangle \\ \langle ivkg \rangle &= \langle 1 \quad 4 \quad 6 \quad 8 \quad 9 \rangle\end{aligned}$$

EXAMPLE 7.2. Sparse storage of the terms of a non-symmetrical matrix

Consider the following non-symmetrical matrix $[K]$:

$$[K] = \begin{bmatrix} 1.6 & 2.1 & 6 & 0 \\ 3.1 & 3.4 & 0 & 7 \\ -1 & 0 & 7 & 2 \\ 0 & -5 & 1 & 7 \end{bmatrix}$$

with $neq = 4$, $nnz = 12$.

The corresponding pointers are:

$$\begin{aligned}\langle vkg \rangle &= \langle 1.6 \quad 2.1 \quad 6 \quad 3.1 \quad 3.4 \quad 7 \quad -1 \quad 7 \quad 2 \quad 5 \quad 1 \quad 7 \rangle \\ \langle jvkg \rangle &= \langle 1 \quad 2 \quad 3 \quad 1 \quad 2 \quad 4 \quad 1 \quad 3 \quad 4 \quad 2 \quad 3 \quad 4 \rangle \\ \langle ivkg \rangle &= \langle 1 \quad 4 \quad 7 \quad 10 \quad 13 \rangle\end{aligned}$$

The organization of the subprograms of the assembly phase is as follows:

```

PointeursInitialiser
    Loop on the elements
        PointeursMorse
        PointeursAssembler
    End of loop on the elements
    PointeursReordonner
    PointeursTrier

```

Figure 7.1. Assembly algorithm

The function of each of these pointers is described below:

PointeursInitialiser

- This subprogram initializes the pointers and the vector containing the values from the global matrix. The pointer of rows $ivkg$ is initialized at $(-i)$ for each row i , whereas the other vectors are initialized at 0. Note that:

$$ivkg(neq+1) = maxnnz,$$

where $maxnnz$ is the maximum memory space allocated for the storage of the matrix $[K]$. This enables us to verify whether the memory space is sufficient. The vector $itmp$ is a work vector of size $maxnnz$. It will be destroyed at the end of the subprogram.

The algorithm is detailed in Figure 7.2:

```

Loop: i = 1, ..., neq
      ivkg(i) = -i
End of loop
      ivkg(neq+1) = maxnnz
Loop: i = 1, ..., maxnnz
      jvkg(i) =      0,      vkg(i) =      0,  itmp (i) = 0
End of loop

```

Figure 7.2. PointeursInitialiser algorithm

PointeursMorse

- This subprogram is within a loop on the elements and calls the subprogram **PointeursAssembler**. It owes its name to the French term for “sparse”: “Morse”. To begin with, it is used to locate the connectivities of the elementary stiffness matrix and their associated values corresponding to each element. It then feeds this to the subprogram **PointeursAssembler**.

The algorithm is detailed in Figure 7.3:

```

Loop: i = 1, idle
    ii = kcle(i)

    Loop: j = 1, idle
        jj = kcle(j)
        valeur = vke(i,j)

        PointeursAssembler (ii, jj, valeur)

    End of loop

End of loop

```

Figure 7.3. PointeursMorse algorithm

where:

- *idle* is the number of degrees of freedom for each element;
- *kcle(i)* is the position of the degree of freedom *i* in the global matrix;
- *vke* is the elementary stiffness matrix.

PointeursAssembler

- This subprogram positions the elementary degrees of freedom and their respective values in the global pointers and vectors *ivkg*, *jvkg* and *vkg*. Each time a new value transmitted by **PointeursMorse**, this subprogram compares and positions that value in relation to the values already assembled. If the number of the degree of freedom has not already been transmitted, a new case is created in the global pointers and vectors. The vectors *ivkg*, *jvkg* and *vkg* are then updated. If, on the other hand, the number of the degree of freedom exists, only the global vector *vkg* is updated by adding the new term to the old value.

PointeursReordonner

- This subprogram reorganizes the global vectors and pointers in their pseudo-final forms by moving the terms from one position to another in accordance with the linking vector *itmp* calculated above. The rows and columns are not sorted in ascending order.

PointeursTrier

- This subprogram sorts the numbers of the rows and columns in ascending order and transforms the vector $ivkg$, which contains the number of non-null terms in each row, into a cumulative pointer such that:

$$ivkg(i+1) - ivkg(i) = nnzligne_i = nzi$$

where nzi ($=nnzligne_i$) is the number of non-null terms in row i .

To do so, we first calculate the permutation vector for each row. We then change the position of the terms in each row in accordance with the permutation found. Note that the permutations calculated in the subprogram **PointeursTrier** do not correspond to an optimal renumbering of the unknowns. These permutations serve only to reorder the different pointers.

On return from the three assembly subprograms above – **PointeursAssembler**, **PointeursReordonner** and **PointeursTrier** – all the pointers and localization vectors for the matrix $[K]$ are constructed. By way of comparison, the two algorithms detailed in Figures 7.4 and 7.5 can be used to localize any term from the global matrix for two types of storage: full storage (the matrix is stored in a two-dimensional table, all the terms are stored) and *sparse* storage (only the non-null terms are stored):

Loop on the rows: $i = 1, neq$

Loop on the columns: $j = 1, neq$

valeur = $vk\mathbf{g}(\mathbf{i}, \mathbf{j})$

End of loop

End of loop

Figure 7.4. Algorithm for locating a term with full storage

Loop on the rows: $i = 1, neq$

ligne_i = ivkg(i) (pointer at start of row i)

ligne_i1 = ivkg(i+1) (pointer at start of row i+1)

```

Loop on the columns: j = ligne_i, iligne_i1
    colonne = jvk(j)      (number of the column)
    valeur = vkg(j)
End of loop
End of loop

```

Figure 7.5. Algorithm for locating a term with sparse storage

The permutation subprograms **dvperm** and **ivperm** called in the subprogram **PointeursTrier** are almost identical. They are used to permute a real (or integer, respectively) vector x in accordance with the permutation vector $perm$ such that:

$$x(perm(j)) = x(j), \quad j = 1, 2, \dots, n.$$

7.1.2 PERMUTATION USING THE “MINIMUM DEGREE” ALGORITHM

The *minimum degree* algorithm is used to construct a permutation matrix $[P]$ that minimizes the number of non-null terms that appear when the matrix is decomposed in the form:

$$[K] = [L][S].$$

These terms are called “fill-in terms”. The solution of the initial system $[K]\{U\} = \{F\}$ is then replaced with that of the equivalent system:

$$([P][K][P]^T)[P]\{U\} = [P]\{F\}.$$

i.e. $[K']\{U'\} = \{F'\}$, in which the number of non-null terms in the decomposition $[K'] = [L][S']$ is less than the number of non-null terms that there would be in the decomposition of the initial matrix.

To illustrate the phenomenon of fill-in, let us consider a linear system of equations $[K]\{U\} = \{F\}$, where $[K]$ is a sparse symmetrical matrix. A direct method of solving this problem consists of decomposing the matrix:

$$[K] = [L][D][L]^T$$

where $[L]$ is a lower triangular matrix and $[D]$ a diagonal matrix. The matrix $[L]$ is obtained by successive calculation of the intermediary matrices $[L^k]$, where

k varies from 1 to n , n being the dimension of the matrix $[K]$. The factorization involves starting with $[L^0]=[K]$ and then calculating the terms of the matrix $[L^{k+1}]$ by adding the products of the rows and columns of $[L^k]$ with other rows and columns of $[L^k]$. Thus, the matrix $[L]$ has exactly the same topology as $[K]$ plus additional non-null terms that were equal to zero in the initial matrix $[K]$. These additional terms are called fill terms or “*fill elements*”. It is the presence of these fill terms that increases the cost of calculation and storage of the factorization.

Figure 7.7 illustrates the different stages of factorization of an initial matrix $[K]$. In the example, the non-null terms from the initial matrix are marked with an “**X**”, whereas the fill elements are marked with black circles. A graph G^0 is associated with this matrix initially (stage 0). The peaks $1, 2, \dots, n$ on the graph G^0 correspond to columns $1, 2, \dots, n$ from the matrix $[K]$. A line (i, j) on the graph G linking the peaks i and j in G^0 exists if and only if $[K(i,j)]$ is non-null. We use the term “degree of a peak” to denote the number of lines emanating from that peak. Because the assembly process is symmetrical, $[K(j,i)]$ is also non-null, even for non-symmetrical matrices $[K]$. The graph is a representation of the non-null positions; the value of the terms is of no importance here. Factorization of the first variable (move from stage 0 to stage 1) eliminates node 1 of the graph and introduces a new line between nodes 3 and 4. This is a so called “*fill edge*”. It corresponds to the appearance of a non-null term in place of a term that was initially null in the matrix. The continued factorization process alters the graph in accordance with this scheme (see illustrations of stages 2–5): this is the elimination graph. The graph G^{k+1} is obtained by adding edges so as to render mutually adjacent all the peaks v^k which were not previously adjacent; then the peak v^k is eliminated, along with all the edges which are linked to it.

The succession of graphs G^k , represents the fill created for each stage of the factorization. The initial sequence of elimination of the graph is the graph of the matrix, $G^0=G([K])$. At each stage k , we consider v^k to be the peak corresponding to the k th column of $[K]$ to be eliminated. Here, we have considered the initial permutation $< 1 \ 2 \ 3 \ 4 \ 5 \ 6 >$. During factorization, two fill elements appear in stages 1 and 3. A *minimum degree* algorithm will construct a new permutation, aimed at minimizing filling.

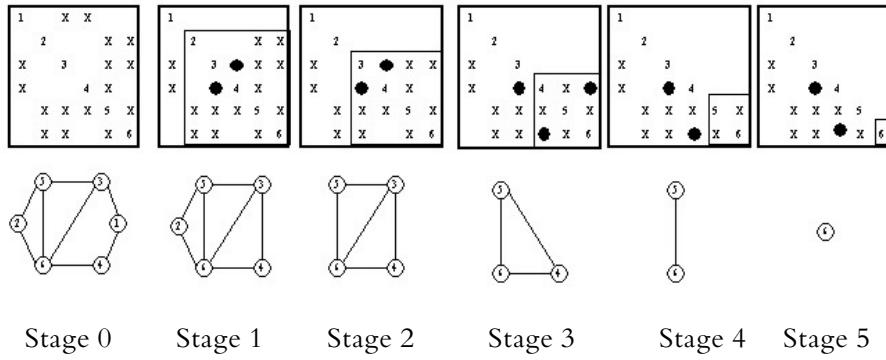


Figure 7.6. Factorization of a symmetrical matrix with fill elements

Minimum degree methods are based on graph theory and use the adjacency graph G^0 of the sparse matrix to be factorized as their initial data. They construct a new permutation of the peaks. They are heuristics that sequentially define the order of the peaks by choosing to number the peak with smallest current degree that has not yet been numbered next. Thus, the graph is updated as and when the peaks (i.e. the rows and columns of $[K]$) are renumbered. The algorithms that result from these methods are very quick and yield the best known results for particular types of graphs.

The principle of a minimum degree algorithm is detailed in Figure 7.7:

Beginning: construct G^0 representing the graph of the matrix $[K]$

Iterate: for: for $k = 1, 2, \dots$, up to $G^k = \{\}$

Select a peak $v^k \in G^k$ which has a minimum degree

Eliminate v^k from G^k to form G^{k+1}

End of loop

Figure 7.7. Minimum degree algorithm

The final permutation is the list of peaks obtained $\langle v^0, v^1, \dots \rangle$.

The minimum degree algorithm minimizes fill-in by choosing the peak with the current minimum degree each time, i.e. the peak that has fewest current edges in the sequence of elimination from the graph. This process is repeated

until all the peaks from the graph have been eliminated: the numbering is then complete and the new permutation is constructed. The application of the algorithm to the example given in Figure 7.7 constructs the permutation:

$< 2 \ 6 \ 1 \ 3 \ 4 \ 5 >$

which produces only a single fill element, and this is the minimum number of fill elements for this example. From this, we deduce the permutation matrix:

$$[P] = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

7.1.3 MODIFIED COLUMN–COLUMN STORAGE FORMAT

The format proposed here, which we will call the *modified column–column* format, is a variant of the *row-by-row* format presented above. The reasons for which we first stored the matrix in a *row-by-row* format are that this offers a time-gain for the assembly and permutation phases. The computation time needed to transform *row-by-row* storage into modified *column–column* storage is paltry in comparison to the time required to carry out the decomposition phase, for instance.

In the *modified column–column* storage format, the values of the terms on the diagonal are stored first in the vector vkg , followed by the values of the terms of the upper part of the matrix, taken column-by-column, followed finally, if the matrix is asymmetrical, by the values of the terms of the lower part of the matrix, taken row-by-row. Only two vectors are needed to represent $[K]$: the vectors vkg and $jvkg$.

Consider the following:

- neq : the dimension of the matrix $[K]$;
- nnz_i : the number of non-null terms in the matrix, strictly above the column i ;
- $nnz = \sum_{i=1}^{neq} nnz_i$.

The dimension of the vector $jvkg$ is $(neq + 1 + nnz)$. The size of the vector vkg is $(neq + 1 + nnz)$ if the matrix $[K]$ is symmetrical and $(neq + 1 + 2*nnz)$ if $[K]$ is non-symmetrical. The values of $jvkg(i)$ for $i = 1, neq + 1$, are defined by:

$$\begin{aligned} jvkg(1) &= neq + 2 \\ jvkg(i+1) &= jvkg(i) + nnz_i, \quad i = 1, \dots, neq. \end{aligned}$$

Similarly, the terms of vkg are defined as:

$$\begin{aligned} vkg(1) &= K(1,1), \quad i = 1, \dots, neq, \\ vkg(neq+1) &\text{ is arbitrary,} \\ vkg(k) &= K(i,j), \quad j = jvkg(k), \quad jvkg(i) \leq k \leq jvkg(i-1)-1. \end{aligned}$$

If $[K]$ is non-symmetrical:

$$vkg(k+nnz) = K(i,j), \quad j = jvkg(k), \quad jvkg(i) \leq k \leq jvkg(i-1)-1.$$

EXAMPLE 7.3. Sparse storage of the terms of a non-symmetrical matrix

Consider the following non-symmetrical matrix $[K]$:

$$[K] = \begin{bmatrix} K_{11} & K_{12} & K_{13} & 0 & 0 \\ K_{21} & K_{22} & 0 & K_{24} & 0 \\ K_{31} & 0 & K_{33} & K_{34} & K_{35} \\ 0 & K_{42} & K_{43} & K_{44} & 0 \\ 0 & 0 & K_{53} & 0 & K_{55} \end{bmatrix}$$

The corresponding pointers are:

$$\begin{aligned} \langle vkg \rangle &= \langle K_{11} \ K_{22} \ K_{33} \ K_{44} \ K_{55} \ X \ K_{12} \ K_{13} \ K_{24} \ K_{34} \ K_{35} \ K_{21} \ K_{31} \ K_{42} \ K_{43} \ K_{53} \rangle \\ \langle jvkg \rangle &= \langle 7 \ 7 \ 8 \ 9 \ 11 \ 12 \ 1 \ 1 \ 2 \ 3 \ 3 \rangle \end{aligned}$$

The advantages of such a storage format over the previous ones are:

- the vector $ivkg$ vanishes;
- given that the matrix is topologically symmetrical, we do not store the numbers of the columns of the lower triangular matrix;
- a single pointer $jvkg$ exists to locate and manipulate the terms of the matrix. This technique offers an appreciable gain in computation time – especially on a parallel computer.

By way of example, the algorithm detailed in Figure 7.8 enables us to calculate the product of a matrix $[K]$ by a vector $\{U\}$. The result is stored in a vector $\{F\}$.

```

Loop: i = 1, neq
    y(i) = K(i) * x(i)
    lpos = jvkg(neq+1) - jvkg(1)
    upos = 0
    Loop: i = 1, neq
        Loop: k = jvkg(i), jvkg(i+1) - 1
            j = jvkg(k)
            y(i) = y(i) + K(k + lpos) * x(j)
            y(j) = y(j) + K(k + upos) * x(i)
        End of loop
    End of loop
End of loop

```

Figure 7.8. Algorithm for the product of a matrix by a vector (Sparse)

For a symmetrical matrix, we write $lpos = 0$, and for a matrix product $\{Y\} = [K]^T \{X\}$ we write $lpos = 0$ and $upose = jvkg(neq + 1) - jvkg(1)$.

- To solve a system of equations with an iterative method, we have to use independent access to the terms of the diagonal and the upper and lower parts of the matrix. It is easier to obtain these terms with this format than with the *row-by-row* format.

7.1.4 SYMBOLIC FACTORIZATION

The symbolic decomposition of the matrix is an important phase in the process of resolution. It not only enables us to construct the pointers of the decomposed matrix $[L][S]$, but also precisely determines the number of non-null terms in the decomposed matrix. Thus, the memory space needing to be allocated to the matrix $[L][S]$ is determined even before the actual decomposition has taken place. Hence, this phase offers more effective memory management.

The format for storage of the decomposed matrix $[L][S]$ is similar to that of $[K]$. The diagonal elements are stored first, followed by the terms of the upper matrix column-by-column, and then those of the lower matrix row-by-row.

Thus, with this type of format, the column pointer jl , similar to $jvkg$, is of the same size as it, rather than the size of the decomposed matrix.

7.1.5 NUMERICAL FACTORIZATION

The algorithm presented below genuinely factorizes the matrix $[K]$. To do so, we use the pointer calculated above, namely the vector jl . After this routine, the matrix $[K]$ is factorized and stored in a vector vlu . The subprogram can deconstruct a symmetrical or a non-symmetrical matrix, thanks to an indicator of symmetry $isym$.

```

If [K] is symmetrical:
    iadecale = 0, iudecale = 0, lenu = neq + 1 + jl(neq)
    else:
        iadecale = jvkg(neq+1) - jvkg(1),
        iudecale = jl(neq), lenu = neq + 1 + 2 * jl(neq)
        iexpk = neq
        Loop: i = neq, 1, -1
            indexe(i) = 0
            if: jl(i) = i+1, then: iexpress(i) = iexpk,
            else: iexpress(i) = 0, iexpk = i
        End of loop
        iexpress(neq) = 0
        Loop: i = 1, neq
            somme = 0, si: jvkg(i) >= jvkg(i+1) go to 4
            Loop: innul = jvkg(i+1)-1, jvkg(i), -1
                k = jvkg(innul)
                vlu(jl(innul-1))=vkg(innul),
                vlu(jl(innul-1)+iudecale)=vkg(innul+iadecale)
                Loop: indexk = jl(innul -1), jl(innul) - 1
                    indexe(k) = indexk, iexpk = iexpress(k)
                    if: iexpk >= 0 then:
                        if: iexpress(iexpk) = 0 iexpress(iexpk) = -k
                        if: jvkg(k) >= jvkg(k+1) go to 3
                        lsomme = vlu(indexk+iudecale), usomme = vlu(indexk)

```

```

Loop: knnul = jvkg(k), jvkg(k+1) - 1
      j = jvkg(knnul), jstart = jl(knnul-1)
1      if: jstart >= jl(knnul) go to 2
      if: indexe(j) = 0 then:
          if: iexpres(j) > 0 then:
              jold = j, j = iexpres(j)
              if: iexpres(j) <= 0 then: j = -iexpres(j)
                  jstart = jstart + j - jold
              else: j = jl(j), jstart = jstart + 1
                  go to 1
      Loop: indexj = jstart, jl(knnul) - 1
      lsomme = lsomme - vlu(indexj) * vlu(indexe(j)+iudecale)
      usomme = usomme - vlu(indexj+iudecale) * vlu(indexe(j))
      j = jl(j)
      End of loop
2      End of loop
      vlu(indexk+iudecale) = lsomme, vlu(indexk) = usomme
      k = jl(k)
3      End of loop
      End of loop
      Loop: innul = jvkg(i), jvkg(i+1) - 1
      k = jvkg(innul)
      Loop: indexk = jl(innul - 1), jl(innul)-1
          usaute = vlu(indexk+iudecale)
          vlu(indexk) = vlu(indexk)*vlu(k),
          vlu(indexk+iudecale) = usaute*vlu(k)
          somme = somme + vlu(indexk) * usaute
          indexe(k) = 0, iexpk = iexpres(k)
          if: iexpk > 0, then: iexpres(iexpk) = 0
          k = jl(k)
          End of loop
      End of loop
4      vlu(i) = vkg(i) - somme

```

```

if:   vlu (i) = 0 then: pivot null, stop
else: vlu (i) = 1 / vlu (i)
End of loop

```

Figure 7.9. Numerical factorization algorithm

7.1.6 SOLUTION OF THE SYSTEM BY DESCENT/ASCENT

Following decomposition of the matrix $[K]$ into a product of two matrices $[L]$ and $[S]$, this stage enables us to find the ultimate solution $\{U\}$. In order to do this, three algorithms are needed to successively solve the upper triangular system, the diagonal system and finally the lower triangular system.

$$\begin{aligned} [L]\{b\} &= \{F\} : \text{lower triangular system} \\ [S]\{U\} &= \{b\} : \text{upper triangular system} \end{aligned}$$

The three algorithms needed to solve the system are detailed in Figures 7.10–7.12 (the vector vlu contains the decomposed matrix $[L][S]$):

```

Loop: i = 1, neq
      somme = 0
      Loop: il = jvkg(i), jvkg(i+1) - 1
            k = jvkg(il)
            Loop: j = jl(il-1), jl(il) - 1
                  somme = somme + vlu(j) * u(k)
                  k = jl(k)
            End of loop
      End of loop
      u(i) = f(i) - somme
End of loop

```

Figure 7.10. Resolution of the lower triangular system

```

Loop: i = 1, neq
      u(i) = vlu(i) * u(i)
End of loop

```

Figure 7.11. Resolution of the diagonal system

```

Loop: i = neq, 1, -1
    Loop: il = jvkg(i), jvkg(i+1)-1
        k = jvkg(il)
        Loop: j = jl(il-1), jl(il)-1
            u(k) = u(k) - vlu(j) * u(i)
            k = jl(k)
        End of loop
    End of loop
End of loop

```

Figure 7.12. Resolution of the upper triangular system

7.2 Numerical examples

In this section, we compare the performances obtained in terms of storage and computation time by the skyline solver and the sparse solver. The hydrodynamics software REFLUX, developed by the LHN (Laboratoire d'Hydraulique Numérique – Numerical Hydraulics Laboratory) in Compiègne in France includes both solvers for an implicit resolution of the Navier-Stokes or Saint-Venant equations.

The different tests were conducted on a 2 GHz laptop running Windows XP operating system with 512 MB of RAM.

We examine three scenarios for this study. The first is an academic case, and the latter two are industrial situations. The computation time considered here represents the total time taken to solve a single system of equations.

Example 1

This example models a flow in a permanent fluvial regime. The domain is a rectangle, whose dimensions are 10 m x 2,000 m. The mesh is made up of six-node triangles for 16,641 nodes in total and 8,320 elements. The size of the global matrix is 37,000.

Example 2

This second example relates to the hydraulic impact of certain installations at the level of a community. The domain in question has a length and breadth of

2 km and 1.5 km, respectively. The mesh is composed of six-node triangular elements for 56,424 nodes in total and 28,022 elements. The size of the global matrix is 126,154.

Example 3

The third example relates to the hydraulic impact of certain installations at the level of a community. The domain in question has a length and breadth of 5 km and 1.5 km, respectively. The mesh is composed of six-node triangular elements for 99,458 nodes in total and 49,558 elements. The size of the global matrix is 222,100.

The results obtained are recapped in Figures 7.13–7.15.

Problem	Number of nodes	Number of elements	Number of equations
Example 1	16,641	8,320	37,000
Example 2	56,424	28,022	126,154
Example 3	99,460	49,560	222,100

Figure 7.13. Dimensions of the problems

Problem	Number of terms $(\times 10^6)$				Computation time (sec.)	
	SL [K]	SL [L][S]	Sparse [K]	Sparse [L][S]	SL	Sparse
Example 1	27.4	27.4	1.074543	5.999945	3.5	2.2
Example 2	305.895	305.895	3.712620	19.656784	★★	52
Example 3	520.9	520.9	6.520583	35.553335	★★	90

★★: The size of the matrix is too large.

SL: Skyline.

Figure 7.14. Dimensions of the matrices and computation time

Problem	Number of terms $[L][S]_{Sparse}$ / number of terms $[L][S]_{Skyline}$
Example 1	21.90 %
Example 2	6.42 %
Example 3	6.82 %

Figure 7.15. Comparison of the two means of storage

Bibliography

- [CUT 69] CUTHILL E., MCKEE J., “Reducing the bandwith of sparse symmetric matrices”, *Proceedings of the 24th National Conference of the ACM*, pp. 157–172, 1969.
- [DUF 90] DUFF I.S., ERISMAN A.M., REID J.K., *Direct Methods for Sparse Matrices*, Oxford University Press, 1990.
- [GEO 89] GEORGE A., LIU J., “The evolution of the minimum degree ordering algorithm”, *SIAM Review*, vol. 31, no. 1, pp. 1–19, 1989.
- [SAA 86] SAAD Y., SCHULTZ M.H., “GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems”, *SIAM Journal on Scientific and Statistical Computing*, vol. 7, pp. 856–869, 1986.
- [TIN 67] TINNEY W.F., WALKER J.W., “Direct solutions of sparse network equations by optimally ordered triangular factorization”, *Proceedings of the IEEE*, pp. 1801–1809, 1967.

INDEX

A

approximation
finite element, 28
generalized, 55
nodal, 26, 55
non-nodal, 25
properties, 59, 60
sub-domain, 28
of vectorial values, 153
assembly, 231, 297, 303
technique, 305–310

B

bandwidth, 311
boundary conditions, 173, 231, 318, 320
bubble (element with \sim function), 122

C

Cauchy, 169
changing of independent variables, 421
characteristic polynomials, 482
classification of systems, 163, 166
collocation, 162, 198, 201
compatible elements, 51, 122, 245, 257
complementary functional, 162, 194
complete polynomial, 52, 244
consistency, 244
continuity, 244
 inter-element, 50, 58
 over the element, 50
continuous systems, 163–172
convergence, 243
coordinates
 generalized, 55
 local, 41, 117, 127
 nodal, 28, 55

cubic elements, 103, 105, 119, 121, 123,
124, 131–133, 141, 147
curvilinear elements, 120, 134, 149

D

decomposition
 spectral, 483
 triangular, 345, 391, 394–399
Degrees of Freedom (DoF), 163
Dirichlet, 169–170
discrete systems, 163–164
discretization of integral forms,
 194, 264
distributed calculation, 402

E

eigenvalues, 480–502
eigenvector, 482
element
 infinite, 158
 notion of, 28
 real, 37
 reference, 37
 with a variable number of
 nodes, 156
equations
 boundary-integral, 162
 linear, 384–403
 nonlinear, 404–429
equilibrium problem, 164
error
 approximation, 26, 75–89
 geometric discretization, 34
Euler, 431–455
expansion (matrices and vectors), 299
explicit Eulerian algorithm, 433

F

FEM (Matlab program)
descriptions of data, 517
examples, 549–576
general description, 512
list of, 521–549

finite difference
central, 466
method, 10

finite element
approximation, 28
method, 231

fluid mechanics, 217–223

forces, 164

functional
mixed, 162, 192
notion of a, 162, 182–194

functional blocks, 516, 521

functions
basis, 25
geometrical transformation, 39
interpolation, 25, 46, 73, 97
weighting, 162, 173, 198

G

Galerkin, 162, 183, 202, 206

Gauss-Radau, 364

Gaussian
elimination, 345, 385
numerical integration, 348

Gaussian elimination, 345, 385

generalized functional, 190

H

Hammer formulae, 367

Helmholtz, 170

Hermite elements, 104, 110, 123, 134, 142, 150

hexahedric elements, 143

homogeneous system, 167

Houbolt, 467

I

implicit, 436

improvement of precision, 83

incremental method, 420

initial value problems, 165

integral form
discretized, 194, 264
element, 275, 276
generalized, 190
global, 231
of nonlinear problems, 161
weak, 177

integral formulation, 161

integration
numerical, 281, 345–383
by parts, 162, 174, 175

inverse iteration, 345, 481

isoparametric element, 53

J

Jacobian, 62, 490–493

L

Lagrange multipliers, 162, 188, 190

Lagrangian
elements, 101, 115, 129, 140, 144
multiplier, 162, 188, 190
polynomials, 104

Lax-Wendroff, 449–455

least squares, 162, 204

linear
elements, 99, 113, 128, 139, 143
functional, 185
system, 167

linear relations, 323

list of PDEs, 209–223

localization table, 308–310

locking, 334

M

matrix
elementary, 231, 233, 274, 277, 280
global, 231, 234

Jacobian, 63

mass, 280

nodal, 56

tangent, 412

mesh, 10, 87, 236

minimization of bandwidth, 313

mixed functional, 162, 192

modal superposition, 436, 476
 modification of elements, 155
 modularity of programs, 511

N

Navier-Stokes, 171, 220
 Neumann, 169, 170
 Newmark, 468, 469
 Newton-Cotes, 356
 Newton-Raphson, 345, 411-420
 nodes
 geometrical, 33
 interpolation, 28, 46

O

organization of programs, 513

P

parameters
 general, 25
 nodal, 25
 undetermined, 162
 partition of a domain, 33
 patch test, 231, 256-264
 physical system, 162
 pivot, 389
 polynomial basis, 54
 positive definite
 differential system, 167
 functional, 186
 positive differential system, 167
 positivity, 249-255
 prediction-correction, 444, 445
 prismatic elements, 150
 projectors of a matrix, 483
 propagation problems, 164
 properties
 of the global matrices, 310
 of the systems of equations, 208
 pyramidal element, 152

Q

quadratic
 elements, 101, 116, 134, 140, 144
 functional, 185
 quadrilateral elements, 127

R

Rayleigh quotient, 486
 reaction, 321
 rectangular elements, 136
 reduced integration, 375
 reference space, 37, 41-43
 residuals
 notion of, 172
 weighted, 162, 172
 rigid body modes, 246, 247
 Ritz method
 for eigenvalues, 494
 for functional, 162, 205, 206
 Runge-Kutta, 447, 475

S

scalar field problem, 210-213
 self-adjoint, 167
 separation of eigenvalues, 486
 shifting of eigenvalues, 487
 singularity of the Jacobian matrix, 68
 skyline, 231, 315-318
 solid mechanics, 213-216
 sparse solver, 577-595
 spatial shifting, 459
 stability, 245
 stationarity principle, 187
 steady-state systems, 164
 storage of global matrices, 314-318
 sub-parametric element, 53
 subspace, 345, 495
 substitution method, 345, 407-408
 super-parametric element, 53, 158

T

table
 connectivity, 44
 coordinates, 44
 localization, 308-309
 tetrahedral elements, 137
 transformation
 geometrical, 37
 of an integral, 64, 270
 of derivation operators, 61-71
 of integral forms, 174
 of nodal variables, 135, 137, 270, 321

triangular elements, 111
triangularization, 385, 391
types of elements, 97

V

variables
generalized, 54
modal, 28, 55

variation of a functional, 182, 184
vector
elementary load, 281, 346

W

weak integral form, 177
Wilson, 470