INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Solution of non-linear elasticity problems using the *continu* software

Marina Vidrascu

N 4128

March 1, 2001

THEME 4

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--4128--FR+ENG

Solution of non-linear elasticity problems using the *continu* software

Marina Vidrascu *

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet Macs

Rapport de recherche n° 4128 — March 1, 2001 — 57 pages

Abstract: A general Newton method with arc-length continuation was implemented in the software *continu*. This paper recalls the main principle of the method and describes how to use and modify the software to make it possible to solve a large variety of non-linear problems for hyper-elastic materials. Several three-dimensional finite elements are available

Key-words: finite elasticity, large elastic deformations, Newton method, arc-length continuation, finite elements, incompressible materials

* Macs, email:Marina.Vidrascu@inria.fr

Solution de problèmes d'élasticité non-linéaire avec le logiciel *continu*

Résumé : Le logiciel *continu* utilise une méthode de Newton avec continuation. Ce document présente la méthode et indique comment utiliser et modifier ce logiciel afin de résoudre une large classe de problèmes d'élasticité non linéaire pour des matériaux hyper-élastiques. Plusieurs éléments finis tri-dimensionels sont disponibles

Mots-clés : élasticité non linéaire, grandes déformations, Méthode de Newton, continuation, élements finis, matériaux incompressibles

Contents

1 Introduction

In structural mechanics, the finite element based numerical solution of large non-linear problems involving materials in large deformations requires powerful algorithms and is very expensive. Newton's type algorithms with arc-length continuation appear as the only robust technique to solve three-dimensional non-linear elasticity problems with very strong material heterogeneities in large deformation.

This paper successively describes, on a model finite elasticity problem, a general Newton algorithm using an arc-length continuation procedure and how to use the *continu* software which implements this algorithm. This software is part of the Modulef library. Implementation details will allow the user to modify the actual software in order to be able to solve a large class of problems.

2 Problem description

2.1 Formulation

We consider the numerical solution of nonlinear three-dimensional elasticity problems with hyper-elastic constitutive laws. The Piola-Kirchoff stress tensor is given by :

$$T(x) = \frac{\partial \mathcal{W}}{\partial F}(x, F), \quad F = Id + \nabla u,$$

The variational formulation is obtained by writing the weak form of the equilibrium equations in a fixed reference configuration Ω with boundary $\partial\Omega = \partial\Omega_1 \cup \partial\Omega_2$, and by eliminating the Piola-Kirchoff stress tensor T

$$\int_\Omega \frac{\partial \mathcal{W}}{\partial F}(x, Id + \nabla u) : \nabla v dx = \int_\Omega f^\Omega \cdot v dx + \int_{\partial\Omega_2} f^\Gamma \cdot v da, \forall v \in \mathbf{H}(\Omega), \tag{1}$$

set on the (finite element) space of kinematically admissible displacement fields

$$\mathbf{H}(\Omega) = \left\{ v \in H^1(\Omega, \mathbf{R}^3),\ v = 0 \text{ on } \partial\Omega_1,\ v_{|T_e} \in P^k(T_e) \right\}$$

with respect to the unknown displacement field $u \in \mathbf{H}(\Omega)$. Above, $v_{|T_e}$ denotes the restriction of v on the finite element T_e of a given triangulation of Ω with $P^k(T_e)$ as the local space of interpolation. This total Lagrangian formulation takes the abstract form

$$\mathcal{F}(U, f^\Omega, f^\Gamma) = 0$$

(and is highly nonlinear due to the specific form of the energy density $\mathcal{W}$ as a function of ∇u).

For isotropic hyper-elastic materials the constitutive energy only depends on the invariants $I_1, I_2, J = \det F$ of the right Cauchy Green tensor $F^t F = (Id + \nabla u)^t (Id + \nabla u)$. A typical example is given by the Ciarlet-Geymonat law for compressible materials (which is a particular case of the Ogden law).

$$\mathcal{W}(F) = C_1(I_1 - 3) + C_2(I_2 - 3) + a(J^2 - 1) - (2C_1 + 4C_2 + 2a) log J, \tag{2}$$

and the Money-Rivlin law for incompressible materials :

$$\mathcal{W}(F) = C_1(I_1 - 3) + C_2(I_2 - 3) \tag{3}$$

where a, C_1, C_2 are material dependent constants.

Another example is the Saint-Venant-Kirchoff law, which is the direct generalization of the linear Hookian law, namely :

$$\begin{aligned} \mathcal{W}(F) &= \frac{\lambda}{2}\left(\mathrm{Tr}(E)\right)^2 + \mu \mathrm{Tr}\left(E^2\right) \\ \text{with } E(u) &= \frac{1}{2}\left(F^T F - Id\right) \end{aligned} \tag{4}$$

where λ and μ correspond to the usual Lamé constants in linear elasticity, and E is the Green-Lagrange tensor.

The numerical solution of this nonlinear variational problem is classically obtained by a Newton algorithm. In this case a linearized problem is solved at each step.

$$\frac{\partial \mathcal{F}(U^n, f^\Omega, f^\Gamma)}{\partial U}.\Delta U + \mathcal{F}(U^n, f^\Omega, f^\Gamma) = 0, \tag{5}$$

with

$$\frac{\partial \mathcal{F}(U^n, f^\Omega, f^\Gamma)}{\partial U} \cdot W = \int_\Omega \nabla w^T : \frac{\partial^2 \mathcal{W}}{\partial F^2}(x, Id + \nabla u) : \nabla v dx. \tag{6}$$

and

$$\mathcal{F}(U^n, f^\Omega, f^\Gamma) = \int_\Omega \frac{\partial W}{\partial F} : \nabla v - \int_\Omega f^\Omega v - \int_{\partial\Omega_2} f^\Gamma v \tag{7}$$

This algorithm converges fast if properly initialized. For this purpose, one may use an arc-length continuation algorithm, which computes the whole solution curve $\mathcal{F}(U(\lambda), \lambda f^\Omega, \lambda f^\Gamma) = 0$ for $0 \leq \lambda \leq 1$ by a Newton-Euler algorithm with automatic time-stepping. For a detailed presentation of the mechanical formulation and numerical approach see [11].

2.2 Non-linear solver : Newton with arc-length continuation

The basic iteration of the Newton algorithm with arc-length continuation used is detailed hereafter.

Initialization

- U^0 and Δs given (possibly $U^0 = 0$), choose an initial increment $\Delta\lambda^0$.
- $\lambda_1 = \Delta\lambda^0$

- compute by a **regular Newton** method U^1 solution of

$$\mathcal{F}(U^1, \lambda_1 f^\Omega, \lambda_1 f^\Gamma) = 0$$

- set

$$\begin{aligned} \Delta U_1 &= U^1 - U^0 && (8)\\ \Delta\lambda_1 &= \Delta\lambda_0 &&\\ \Delta s_1 &= \sqrt{2}||\Delta U_1|| && (9)\\ \omega &= ||\Delta U_1||^2/||\Delta\lambda_0||^2 && (10) \end{aligned}$$

Continuation $n \to n+1$

- the present displacement field U^n , the load increment λ_n, the arc length Δs_n being known,
- compute $\dot{U}^n$, the tangent to the solution curve, and $\dot{\lambda}^n$, the rate of load increment, by solving the linearized elasticity problem (5)

$$\frac{\partial \mathcal{F}(U^n, \lambda^n f^\Omega, \lambda^n f^\Gamma)}{\partial U}.\Delta U = -\frac{\partial \mathcal{F}(U^n, \lambda^n f^\Omega, \lambda^n f^\Gamma)}{\partial \lambda} \tag{11}$$

and by setting

$$\begin{aligned} \dot{\lambda}^n &= (||\Delta U||^2 + \omega)^{-\frac{1}{2}}\\ \dot{U}^n &= \dot{\lambda}^n \Delta U. \end{aligned}$$

- set the prediction to

$$\begin{aligned} U_0^{n+1} &= U^n + \varepsilon \Delta s_n \dot{U}^n\\ \lambda_0^{n+1} &= \lambda^n + \varepsilon \Delta s_n \dot{\lambda}^n. \end{aligned} \tag{12}$$

with

$$\varepsilon = \begin{cases} 1 & \text{if } \dot{U}^n \cdot [U^n - U^{n-1}] + \omega \dot{\lambda}^n [\lambda^n - \lambda^{n-1}] > 0\\ -1 & \text{if not} \end{cases}$$

- compute U^{n+1} and λ^{n+1} on the solution curve by solving the equation

$$\mathcal{F}(U^{n+1}, \lambda^{n+1} f^\Omega, \lambda^{n+1} f^\Gamma) = 0 \tag{13}$$

by an **extended Newton** algorithm (for k=0 until convergence) where the increments $(U_{k+1}^{n+1} - U_k^{n+1}, \lambda_{k+1}^{n+1} - \lambda_k^{n+1})$ are imposed to be perpendicular to the local tangent

$$\dot{U}_k^{n+1}(U_{k+1}^{n+1} - U_k^{n+1}) + \omega \dot{\lambda}_k^{n+1}(\lambda_{k+1}^{n+1} - \lambda_k^{n+1}) = 0.$$

- set
$$\Delta s_{n+1} = \nu_n \Delta s_n \tag{14}$$
with ν_n a user-defined parameter.

The algorithm stops when $\lambda^n \geq 1$.

Comments Both the choice of the initial increment $\Delta\lambda_0$ and of the parameter ν_n are problem dependent. If the problem is easy to solve a good choice should be $\Delta\lambda_0 = 1$ which means solving the problem by a regular Newton. As for ν_n, its value will depend on how fast the convergence of the step n of the extended Newton algorithm is. We can choose $\nu_n > 1$ if the convergence is fast, $\nu_n < 1$ for slow convergence and $\nu_n = 1$ for regular convergence. The choice of this parameter is done in the program.

The **extended Newton** algorithm (13) also reduces to a sequence of linearized elasticity problems since its solution amounts to

- solve the linear elasticity systems (5)

$$\begin{aligned}
\frac{\partial \mathcal{F}(U_k^{n+1}, \lambda_k^{n+1} f^\Omega, \lambda_k^{n+1} f^\Gamma)}{\partial U} \cdot P &= -\mathcal{F}(U_k^{n+1}, \lambda_k^{n+1} f^\Omega, \lambda_k^{n+1} f^\Gamma), \\
\frac{\partial \mathcal{F}(U_k^{n+1}, \lambda_k^{n+1} f^\Omega, \lambda_k^{n+1} f^\Gamma)}{\partial U} \cdot Q &= -\frac{\partial \mathcal{F}(U_k^{n+1}, \lambda_k^{n+1} f^\Omega, \lambda_k^{n+1} f^\Gamma)}{\partial \lambda}.
\end{aligned}$$

- update $\dot\lambda_{k+1}^{n+1}, \dot U_{k+1}^{n+1}$ by

$$\begin{aligned}
\dot\lambda_{k+1}^{n+1} &= \frac{\mathrm{sign}[\dot{\mathrm{U}}_{\mathrm{k}}^{\mathrm{n+1}} \cdot \mathrm{Q} + \omega \dot\lambda_{\mathrm{k}}^{\mathrm{n+1}}]}{(\|\mathrm{Q}\|^2 + \omega)^{\frac{1}{2}}} \\
\dot U_k^{n+1} &= \dot\lambda_{k+1}^{n+1} . Q \\
\Delta\lambda &= \frac{\dot U_{k+1}^{n+1} \cdot P}{\dot U_{k+1}^{n+1} Q + \omega \dot\lambda_{k+1}^{n+1}}
\end{aligned}$$

- and finally obtain

$$\begin{aligned}
\lambda_{k+1}^{n+1} &= \lambda_k^{n+1} + \Delta\lambda \\
U_{k+1}^{n+1} &= U_k^{n+1} + P + \Delta\lambda Q
\end{aligned}$$

3 Use of the *continu* software

3.1 General philosophy

The *continu* software is designed to be incorporated in the Modulef library and uses existing modules to perform the Newton iterations. In addition, standard Modulef **data structures** [3], [4] are used.

More precisely the *continu* software can be considered as a super-module, the **inputs** of this super-module are :

- the mesh of the domain stored in a **NOPO** data structure
- the description of material properties and loads use **MILI, FORC** data structures and, if necessary, user-defined subroutines (**MILIEU, FORCE or SPUVEL**) [13]
- the initial solution (if any) stored in a **B** data structure
- a **data file** containing general information on the actual problem (to be detailed in the next sections)
- a user program **VALCLD** to describe boundary conditions [13]
- a subroutine **ENERGY** to describe the internal energy
- a pair of subroutines **NLRAID, NLSECM** containing, for the selected finite elements, the computation of elementary stiffness matrix and right hand side.

and the **output** is

- the solution stored in a **B** data structure

The user can be faced to two different situations.

i) The actual software can solve the problem without modifications. In this case the user has to fill in the **data file**, to select an **internal energy** and **a finite element** among those available and to write the **user subroutines**. This is very easy to do for Modulef users and is explained in §3.2. For new users a more detailed presentation is given in §3.3 which provides the pointers to specific Modulef user manuals. When both the English and French version of the manual are available there are two references.

ii) The existing software has to be slightly modified because the problem is too stiff or because new elements or a new internal energy are to be added. To make this possible a description of the software is given in §5 and one of the internal energy in §4.1.

3.2 Expert Modulef users

3.2.1 Input data structures

The following data structures are needed as input files for the *continu* software :

- NOPO the data structure containing the mesh
- MILI the data structure describing the material properties
- FORC the data structure describing the loads
- B if an initial guess of the solution is known

All names of files containing data structures are generated by the software. The name is composed by a regular name NOMG and the type of data structure. The only exception is the name of the file containing the initial guess. This is a data structure of type B. It's name is <NOMG>.BINI . It is the user's responsibility to give the appropriate names to input data structures.

3.2.2 Output data structure

The software generates an output data structure of type B containing the solution.

It is possible to use the output of a first computation as input for further computations. In this case the user just needs to change the file name from <NOMG>.B to <NOMG>.BINI.

3.2.3 Description of the data file

The input data file is an ASCII file. It will be read using the Modulef free format, that means comments and variables are allowed (see [3] for details). For all items their type (integer I, real single R or double D precision, character C) is specified bellow. When several data are used for a specific module the name of this module is given with the number of the Modulef Guide describing it.

The data file contains :

NOMG	(C)	regular name of files. The file names will be <NOMG>.<D.S. name> (e.g. if NOMG is 3d the name of the mesh file is 3d.NOPO)
INITIN	(I)	0 if the algorithm is initialized at zero 1 if the algorithm starts from a previous solution stored in the file <NOMG>.BINI
NDSD	(I)	number of subdomains (materials)
IEPOBA	(I)	0 if the direct access POBA file is not used by the selected finite element

		1 if POBA is used
NMPOBA	(C)	name of the file if IEPOBA is equal to 1

data for COMACO ([13])

NDIM	(I)	space dimension
NBSC	(I)	number of curved surfaces (if curve finite elements are used)
NBLC	(I)	number of curved lines

Loop 1 to NBSC
NOSC(J) (I) number of the j-th curved surface
end of the loop

Loop 1 to NBLC
NOLC(J) (I) number of the j-th curved line
end of the loop

NOMBIB	(C)	name of the library 4 characters (in general ELAS)
NTYED	(I)	number of straight finite element types

Loop 1 to NTYED
NAMED (C) name of the element twice 4 characters
end of the loop

NTYEC (I) number of curved finite element types

Loop 1 to NTYEC
NAMED (C) name of the element twice 4 characters
end of the loop

data for COBDC1 ([13])

NBFR (I) number of reference numbers used for boundary conditions (the same number may appear several times for different unknowns or mnemonics)

Loop 1 to NBFR

NOFR(i,1)	(I)	reference number
end of the loop		
Loop 1 to NBFR		
NOFR(i,2)	(I)	number of the variational unknown
end of the loop		
Loop 1 to NBFR		
NOFR(i,3)	(I)	corresponding mnemonic of the variational unknown
end of the loop		
NTY	(I)	type of the unknowns (5 for double precision or 2 for single)

data for THENEW ([13])

NPROV	(I)	type of problem (in general 2 for regular elasticity)
IOPT(1:4)	(I)	elementary arrays to be computed
		IOPT(1) = 0 no mass matrix (general case), 1 otherwise
		IOPT(2) = 1 compute stiffness matrix (general case), 0 otherwise
		IOPT(3) = 1 compute right hand side (general case), 0 otherwise
		IOPT(4) = 0 no elementary stress matrix (general case), 1 otherwise
ND	(I)	number of degrees of freedom per node (if constant), 0 otherwise
COUL	(C)	name of the array defining the coloring properties
		(used only with vectorized elements and LVECT > 1)
LVECT	(I)	length of vectors (in general 1)
		(except on vector computers for some elements)

specific data for the continuation process

ITEMAX	(I)	maximum number of iterations for standard Newton
Loop 1 to NDSD		
LOI(i)	(I)	describe the internal stored energy.
		New values can be added
		1 Ogden Ciarlet-Geymonat (hyper-elastic compressible)
		2 Mooney-Rivlin (hyper-elastic incompressible)
		3 Saint Venant Kirchoff
end of the loop		
DLAM	(D)	coefficient to apply to the load

The structure of this input file is, on purpose very close to the input data of standard modules used. This is why sometimes part of the data required may appear as unnecessary (for example for the array iopt). Notice that the list of finite elements used is given for the whole domain and not subdomain by subdomain.

3.2.4 User subroutines

All user subroutines (except SPUVEL) are parameters of the modules used and declared as **EXTERNAL**. This means that the names may change from one application to another. The names used here are those generally encountered in the Modulef documentation. Once the **user subroutines** are written or selected it is necessary to link them with the main program to obtain the software.

Description of boundary conditions

It is mandatory to give the function describing prescribed boundary conditions. The function **VALCLD** is the same as in Module COBDC1 (see [13])

```
      DOUBLE PRECISION FUNCTION VALCLD(I,X,Y,Z)
C  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  AIM  :  DESCRIBE PRESCRIBED BOUNDARY CONDITIONS
C  ...................................................................
C   Input Parameters
C   -------------------
C    I       : LINE NUMBER IN ARRAY  NOFR DESCRIBING THE TRIPLE
C              (REFERENCE NUMBER, NUMBER OF VARIATIONAL UNKNOWN, MNEMONIC)
C    X, Y, Z : NODES COORDINATES
C
C   Output parameter
C   --------------------
C    VALCLD  :  VALUE PRESCRIBED TO D.O.F
C .................................................................
```

Description of material characteristics

The following subroutines are optional. The user has various options to describe material characteristics and loads. (This is the regular use of Modulef finite elements, for details see [13]) :

i) using arrays

ii) using subroutines FORCE and/or MILIEU

iii) using the user defined subroutine SPUVEL

If options ii) or iii) are selected the corresponding subroutine must be supplied by the user.

Here the parameters of these routines are recalled, see §3.3 or [13] for a more detailed presentation.

The routine MILIEU is used for matrices and FORCE for right-hand sides. Both have the same input and output parameters and are used element by element.

```
      SUBROUTINE FORCE(M,LOPT,X,NDIM,NPO,TAR,LTAR,LVECT,IADR,I1,NARE,IA)
      SUBROUTINE MILIEU(M,LOPT,X,NDIM,NPO,TAR,LTAR,LVECT,IADR,I1,NARE,IA)
C ..................................................................
C AIM : GIVE MATERIAL CHARACTERISTICS OR LOADS IN ORDER TO COMPUTE
C ---   ELEMENTARY MASS, STIFFNESS OR RIGHT HAND SIDES.
C
C ..................................................................
C   INPUT PARAMETERS
C   -------------------
C
C    M     : WORKING ARRAY
C    LOPT  : NUMBER OF THE OPTION AS INDICATED IN COMILI(COFORC)
C    X     : REAL ARRAY X(LVECT,NPO,NDIM ) CONTAINS THE POINT COORDINATES
C            OF THE ACTUAL ELEMENT
C    NDIM  : SPACE DIMENSION
C    NPO   : NUMBER OF POINTS OF THE ACTUAL ELEMENT
C    LTAR  : LENGTH (IN WORDS) OF THE ARRAY TO FILL-IN
C    LVECT : LENGTH OF VECTORS
C    IADR  : ADDRESS OF VARIABLES (GIVEN IN COMILI/COFORC)
C            (IN GENERAL 1, NOT USED)
C    I1    : STANDARD USE I1 IS  0  (NOT USED)
C    NARE  : NUMBER IN THE LOCAL NUMBERING OF THE ELEMENT OF THE
C            FACE (IN 3D), THE EDGE  OR THE POINT
C    IA    : ADDRESS -1  IN ARRAY MAIL OF THE DATA STRUCTURE MAIL
C            OF THE ACTUAL ELEMENT
C
C   OUTPUT PARAMETER
C   --------------------
C    TAR   : ARRAY (LTAR,LVECT) CONTAINS DATA USED TO COMPUTE
C            ELEMENTARY ARRAYS
C

C ------------------------ DOCUMENTATION ----------------------------
C --        GUIDE 4
C ..................................................................
```

A distinct way to give data is to use the user subroutine SPUVEL. In this case the data are given by integration point.

```
      SUBROUTINE SPUVEL(CHAINE,NUMERO,X,Y,Z,LVAL1,NCOMP,LV,VAL,NPI)
C  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  AIM :   DESCRIBE DATA FOR THE COMPUTATION OF ELEMENTARY ARRAYS
C  ---     IN ELASTICITY PROBLEMS
C
C  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C  INPUT PARAMETERS
C  ------------------
C    CHAINE    : KEY WORD 4 CHARACTERS INDICATE TYPE OF DATA TO PROVIDE
C    NUMERO    : SUBDOMAIN OR REFERENCE NUMBER
C    X,Y,Z     : COORDINATES  OF THE NPI INTEGRATION POINTS
C                FOR LV ELEMENTS X(LV,NPI)
C    LVAL1     : NUMBER OF VALUES TO COMPUTE
C    NCOMP     : NUMBER OF COMPONENTS OF THE ARRAY DESCRIBING DATA
C    LV        : LENGTH OF VECTORS (OR 1)
C    NPI       : NUMBER OF INTEGRATION POINTS
C
C  OUTPUT PARAMETERS
C  --------------------
C    VAL       : ARRAY (LVAL, LVECT) CONTAINING THE DATA
C  +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

The parameter **CHAINE** in the subroutine SPUVEL will specify the type of data and can take the following values :

'ELAS'	give the elasticity tensor
	(Young coefficient and Poisson ratio for isotropic materials)
'EFOO'	values of forces on the domain Ω
'EFOF'	values of forces on the boundary Γ

Description of finite elements

All the finite elements are described in the subroutines **NLRAID, NLSECM**. If new elements are developed they can be added to these routines or use different names. The construction is the same as the one adopted in the linear case and described in [13]. The accessible elements are listed hereafter. A description of these elements is given in §4.2

TETR 3P20	10 nodes tetrahedra	(curved element)
HEXA 3Q2D	20 nodes hexaedra	(straight element)
HEXA 3Q2C	20 nodes hexaedra	(curved element)
HEXA 3QCC	27 nodes hexaedra	(curved element)
PENT 3R2D	15 nodes pentaedra	(straight element)

Description of internal energy

Several internal energies are available (see parameter LOI in §3.2.3 and §4.1 for a detailed description).

They correspond to the following routines :

energy	Saint Venant Kirchoff	(loi=3)
enerog	hyper-elastic (Ogden or Mooney-Rivlin)	(loi=1 or 2)

It is possible to add new types of energies. In this case a new program describing the energy has to be written (see §4.1) and a new value for loi associated to the new energy be defined.

Comments on enerog

The same routine is used for compressible and incompressible materials. We recall here the specific form of the corresponding constitutive energy (2), (3); the Ogden law for compressible materials :

$$\mathcal{W}(F) = C_1(I_1 - 3) + C_2(I_2 - 3) + a(J^2 - 1) - (2C_1 + 4C_2 + 2a)logJ,$$

and the Mooney-Rivlin law for incompressible materials :

$$\mathcal{W}(F) = C_1(I_1 - 3) + C_2(I_2 - 3)$$

where a, C_1, C_2 are material dependent constants to be provided by the user.

As described in §2.1 for compressible materials the problem to be solved is (1) :

$$\int_\Omega \frac{\partial \mathcal{W}}{\partial F}(x, Id + \nabla u_h) : \nabla v_h dx = \int_\Omega f^\Omega \cdot v_h dx + \int_{\partial\Omega_2} f^\Gamma \cdot v_h da, \forall v_h \in \mathbf{H}(\Omega),$$

In the incompressible case the incompressibility constraint is penalized and the problem to be solved reads :

$$\int_\Omega \frac{\partial \mathcal{W}}{\partial F}(x, Id + \nabla u_h) : \nabla v_h dx + \int_\Omega \frac{1}{\varepsilon}[\mathrm{det}(\mathrm{Id} + \nabla_\mathrm{h} \mathrm{u_h}) - 1]\frac{\partial \mathrm{det}}{\partial \mathrm{F}}(\mathrm{Id} + \nabla_\mathrm{h} \mathrm{u_h}) : \nabla_\mathrm{h} \mathrm{v_h} \mathrm{dx}$$
$$= \int_\Omega f^\Omega v_h dx + \int_{\partial\Omega_2} f^\Gamma v_h da, \forall v_h \in \mathbf{H}(\Omega),$$

where

- $\nabla_h v$: is the reduced gradient used in the penalization of the incompressibility constraint.
- ε : is the penalization parameter.

In this case the linearized form ((5) for the compressible case) reads

$$\frac{\partial \mathcal{F}(U^n, f^\Omega, f^\Gamma)}{\partial U}.\Delta U + \mathcal{F}(U^n, f^\Omega, f^\Gamma) = 0,$$

with

$$\frac{\partial \mathcal{F}(U^n, f^\Omega, f^\Gamma)}{\partial U} \cdot W = \int_\Omega (\frac{\partial^2 W}{\partial F^2}(x, Id + \nabla u_h) : \nabla w) : \nabla v dx$$
$$+ \int_\Omega \frac{1}{\varepsilon}[\det(\mathrm{Id} + \nabla_{\mathrm{h}}\mathrm{u_h}) - 1][\frac{\partial^2 \det}{\partial \mathrm{F}^2}(\mathrm{Id} + \nabla_{\mathrm{h}}\mathrm{u_h}) : \nabla_{\mathrm{h}}\mathrm{v}] : \nabla_{\mathrm{h}}\mathrm{wdx}$$
$$+ \int_\Omega \frac{1}{\varepsilon}[\frac{\partial \det}{\partial \mathrm{F}}(Id + \nabla_h u_h) : \nabla_h v][\frac{\partial \det}{\partial \mathrm{F}}(Id + \nabla_h u_h) : \nabla_h w]dx$$

Combine finite elements and internal energy

In the general Modulef specification a finite element refers to both an interpolation and a set of equations to be solved. In the present context several stored internal energies can be used with the same finite element. The table hereafter will give, for each stored energy the list of elements and the constants to be provided.

energy	E, ν	TETR 3P20, HEXA 3Q2D, HEXA 3Q2C, HEXA 3QCC, PENT 3R2D
enerog	compressible C_1, C_2, a	TETR 3P20, HEXA 3Q2D, HEXA 3Q2C, HEXA 3QCC, PENT 3R2D
	incompressible C_1, C_2, ε	TETR 3P20, HEXA 3QCC

3.3 Detailed presentation

The Modulef guides (in French or English) are part of the Modulef distribution and are also accessible on the web at

http://www-rocq.inria.fr/modulef/Doc/FR/welcome.html
http://www-rocq.inria.fr/modulef/Doc/GB/welcome.html

3.3.1 Input data structures

All input data structures are obtained by calling the relevant preprocessors in the MODULEF library. The aim here is to guide the user through the process of selecting the appropriate modules and documentation guides. Before starting with specific modules it may be helpful to get used with the Modulef terminology which is described, through examples in the introduction of Part II (How do I use Modulef?) in Modulef Guide 1 ([1],[2]). An important concept is that of **subdomain and reference number** used not only for geometrical reasons but also to impose boundary conditions and provide data.

The input data structures are recalled hereafter and the corresponding module and preprocessor are indicated. It is convenient and easy to create these structures in an interactive fashion by choosing the appropriate option from the main menu which will be displayed on the screen once a preprocessor is activated. The introductory examples of Guide 1 and of the specific documentations may help.

- NOPO : the data structure containing the mesh. A description of the mesh generation capabilities in Modulef is given in Guide 3 ([6], [7]). The preprocessor **apnoxx** is used to obtain a two dimensional mesh if such a mesh is needed to create a three-dimensional one. The preprocessor **apn3xx** deals with 3d meshes. A particular attention should be paid to key word MA23 (for module MA2D3E).
- MILI, FORC the data structures describing the material properties and loads. These D.S. are build using the preprocessor **fomixx**, which uses modules COMILI and COFORC, described in [13]. In these data structures the user gives, for each type of data, the code indicating if it is provided by :
  - 1: array,
  - 2: subroutine **FORCE, MILIEU**,
  - 3: user-defined subroutine **SPUVEL**.

  It is possible to mix options depending on the type of data. There are always several alternatives to provide the data. A simple choice is to select the array option for data which is constant on a subdomain (for example the constants used to define the energy). If the values to provide are point-dependent, (for example f^Ω in (1) takes the form $f^\Omega = F(x, y, z)$) option 2 or 3 will be selected. A discussion on relevant choices is given hereafter in §3.3.2.
- B if an initial guess of the solution is known. There are two options to obtain such a data structure. The first one, already mentioned before, is to use the result of a previous computation. In this case just change the name of the file containing the D.S. B. The second possibility is to build a D.S. B with the preprocessor **cosdxx**. This processor uses the modules COSDB , COSNOB or COSMCB and makes it possible to produce constant vectors or to use functions defined at the nodes (with NOPO D.S. or with MAIL and COOR D.S. if the number of degrees of freedom per node is not constant).

3.3.2 User subroutines

Description of boundary conditions : function VALCLD

Give the values of imposed boundary conditions using the function **VALCLD** is the only option for the *continu* software, unlike in standard Modulef preprocessor **cobdxx**. The only difficulty in giving this function is to understand the correlation between **I** line number of the triple (reference number, number of variational unknown, mnemonic) (see §3.2.4) and the content of the **data file** (see §3.2.3).

As shown in §3.2.3 the **data file** provides, column by column information to fill-in an array $NOFR(NTRIPL, 3)$ with

NOFR(i,1)	reference number
NOFR(i,2)	number of the variational unknown
NOFR(i,3)	corresponding mnemonic of the variational unknown

That means that $NTRIPL$ different boundary conditions are defined. Let us consider a simple example : the solution of a 2d elasticity model problem. The mesh with reference numbers is represented in Figure 1. The object is clamped on reference 1 and a displacement of $.5x + .1y$ in the x direction is imposed on the edge referenced 2.

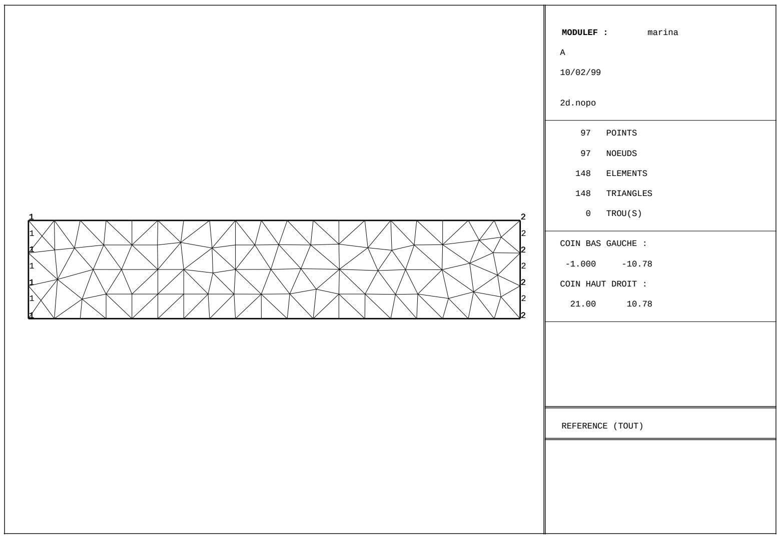

Figure 1: *Model elasticity problem*

Suppose standard Lagrange finite elements are used. The following boundary conditions will be imposed.

$$\begin{aligned} u_x(x,y) &= 0 \text{ on reference } 1 \\ u_y(x,y) &= 0 \text{ on reference } 1 \\ u_x(x,y) &= .5x + .1y \text{ on reference } 2 \\ u_y(x,y) &= 0 \text{ on reference } 2 \end{aligned}$$

In this case $NTRIPL = 4$. The four different functions of x and y needed are coded in a single function **VALCLD**, the **I** parameter will allow to define several functions. For this particular example, if the functions are ordered as above the **data file** will contain the following information :

```
$ data for cobdc1 $
 4 $ number of triples
 1  1  2   2   $ reference number
 1  2  1   2   $ variational unknown
VN VN VN  VN   $ mnemonic
```

and the user function should be :

```
      double precision function valcld(i,x,y,z)
c
c     ux=0 on ref 1 (i=1)
c     uy=0 on ref 1 (i=2) and   uy=0 on ref 2 (i=4)
c
      if (i.eq.1 .or.i.eq.2 .or. i.eq.4) then
         valcld=0
c
c     ux=.5x+.1y on ref 2 (i=3)
c
      else if (i .eq. 3) then
         valcld=.5*x+.1*y
      endif
      end
```

Description of subroutines FORCE, SPUVEL

The main difference between these two options is that subroutine **FORCE** operates on an element and the user-defined subroutine **SPUVEL** on an integration point. In most cases the use of **SPUVEL** is more convenient. Suppose for instance that the data of interest is a right-hand side of the form $f^{\Omega} = F(x, y, z)$ and the element TETR 3P20 was selected. The description of this element (§4.2) indicates that, for the right-hand side, the data for one element the array $\mathcal{V}$ will contain 45 values which are the 3 components of the forces at the 15 integration points. To compute these values the user needs to know the coordinates of the integration points. If the user-defined subroutine **SPUVEL** is utilized these coordinates are input data for the routine. On the other hand, if the subroutine **FORCE** is used it is the user's task to compute these values using the coordinates of the points of the actual element and also information contained in the file **POBA** which is obviously more difficult.

In other situations the use of **SPUVEL** or **FORCE** may present the same degree of difficulty, for example if an homogeneous force is imposed (see §6 for an example of using **FORCE**).

In fact, the option **FORCE** is more general as it gives access to all the information concerning the actual element. For example, if f^{Ω} is not defined as an analytical function but as a nodal function, to compute f^{Ω} one needs the number of nodes of the element under consideration. This information is not available in **SPUVEL**.

4 Internal energies and finite elements available

4.1 Internal energy

For all materials used by the *continu* software, the constitutive energy only depends on the invariants I_1, I_2, J (see 2.1) $\mathcal{W} = \mathcal{W}(I_1, I_2, J)$. To add a new internal energy the user has to provide **a new subroutine** with **same list of parameters** as the existing routines (energy, enerog) and to modify the existing subroutine **inener**. The aim of the new program is to compute first and second derivatives of $\mathcal{W}$ with respect to the invariants I_1, I_2, J.

The routine describing the Saint-Venant-Kirchoff law (4) **energy** is listed below to give an example of such a program.

```
C-PARALLEL-CALL *****************************************************
      subroutine energy( lvect, gi1, gi2, gj,
     &           wi1, wi2, wi3, wi11, wi12, wi13,
     &           wi22, wi23, wi33,ndsde )
      double precision e1,e2,a,rien, gi1, gi2, gj,
     &           wi1, wi2, wi3, wi11, wi12, wi13,
     &           wi22, wi23, wi33
C*******************************************************************
C Gives the 1st and 2nd partial derivatives of W versus the Cauchy
C   invariants I1, I2 and J.
C
C Law: St-Venant - Kirchhoff
C
C      W = lambda(Tr(E))^2 + mu Tr(E^2)
C
```

```
C               W=e1/8(gi1-3)^2+e2/4(gi1^2-2gi2-2gi1+3)
C
C
C Data:
C #  - lvect: number of elements.
C #  - gi1[lvect]: 1st Cauchy invariant.
C #  - gi2[lvect]: 2nd Cauchy invariant.
C #  - gj[lvect]: 3rd Cauchy invariant.
C    - e1 (lambda), e2 (nu) : in common cenerg.
C
C Output:
C #  - wi1: d( W ) / d( I1 )
C #  - wi2: d( W ) / d( I2 )
C #  - wi3: d( W ) / d( J )
C #  - wi11: d^2( W ) / d( I1 )^2
C #  - wi12: d^2( W ) / ( d( I1 ) d( I2 ) )
C #  - wi13: d^2( W ) / ( d( I1 ) d( J ) )
C #  - wi22: d^2( W ) / d( I2 )^2
C #  - wi23: d^2( W ) / ( d( I2 ) d( J ) )
C #  - wi33: d^2( W ) / d( J )^2
C
C********************************************************************
      parameter ( zero = 0.0, deux = 2.0, quatre = 4.0,ndsd=20 )

      common /cenerg/e1(ndsd),e2(ndsd), rien(ndsd,4)

      dimension gi1(lvect), gi2(lvect), gj(lvect),
     &          wi1(lvect), wi2(lvect), wi3(lvect), wi11(lvect),
     &          wi12(lvect), wi13(lvect), wi22(lvect), wi23(lvect),
     &          wi33(lvect)

C     print *, gi1, gi2, gj

      do 10, k = 1, lvect

      wi1(k) = e1(ndsde)*(gi1(k)-3)/quatre + e2(ndsde)*(gi1(k)-1)/deux
      wi2(k) = - e2(ndsde) / deux
      wi3(k) = zero
      wi11(k) = e1(ndsde) / quatre + e2(ndsde) / deux
      wi12(k) = zero
      wi13(k) = zero
      wi22(k) = zero
      wi23(k) = zero
      wi33(k) = zero

10    continue
```

```
      return
      end
```

Comments on the subroutine

- The package allows to compute several elements at the same time (LVECT). This option is particularly valuable on vector computers.
- The domain under consideration can be made with different materials which have the same internal energy but different constants or even different energies. It is common to mix compressible and incompressible materials. The parameter NDSDE gives the subdomain number of the element under consideration.
- As already noticed, several material dependent constants are used to define the energy. They are transmitted to the **energy** subroutine by the **common** /**cenerg**/
- There is a limitation on the number of parameters allowed to define a new energy ($NBRCT = 6$) and for the number of materials used ($NDSD = 20$). It is possible to increase these values for specific applications. (change the data of the subroutine **inener** , line 19)

```
1           subroutine inener(vol,lvol,loi,nocal,ndsde)
2     c     -----------------------------------------
3     c     initialise les tableaux vol (resp surf en 2d)
4     c     in  : nocal = 1 premier appel
5     c     --            2 apres
6     c           loi   = 1 ogden
7     c                   2 mooney-rivlin
8     c                   3 Saint Venant Kirchoff
9     c                   4 melange
10    c           ndsde = nr du sous-domaine de l'element courant si nocal=2
11    c     out : lvol  = si nocal = 1 nombre de constantes de la loi choisie
12    c     ---
13    c           si nocal= 2 le common /cenerg est initialise
14    c                       Ce common figure dans toutes les enrgies
15    c                       on limite a 20 sousdomaines,6 constantes
16    c     -----------------------------------------------------------------
17    c     programmeur : marina Vidrascu  INRIA 1991
18    c     -----------------------------------------------------------------
19          parameter (ndsd=20,nbrct=6)
20          common /cenerg/ctes(ndsd,nbrct)
21          double precision ctes,vol
22          dimension vol(*)
```

```
23        nty = iinfo('REEL2')
24        call typmot(nty,nbremo)
25        if (nocal .eq. 1) then
26           if (loi .eq. 1 .or. loi .eq. 2) then
27              lvol = 3*nbremo
28           else if (loi .eq. 3) then
29              lvol = 2*nbremo
30           else if (loi .eq. 4 ) then
31  c           melange de lois lvol majore toutes les precedentes
32              lvol = 3*nbremo
33           end if
34        else
35           do 1 i = 1,lvol/nbremo
36              ctes(ndsde,i) = vol(i)
37   1       continue
38            if (loi .eq. 2) ctes(ndsde,3)=0.
39        end if
40        end
```

Comments on the inener subroutine

- This routine sets the number of constants used by the new law. This is settled when $NOCAL = 1$. (lines 25-34). For example for the Saint Venant Kirchoff energy (4) we have $loi = 3$ and line 29 indicates that 2 constants are needed (E, ν)
- For $NOCAL = 2$ the constants provided by the user (via D.S. MILI and FORC) are used to initialize **common** /**cenerg**/. This part does not change with new elements.

4.2 Finite elements

4.2.1 The direct access file POBA

Some of the finite elements may use the POBA file. This file contains, for different finite elements, data which make it possible to calculate the elementary arrays and which can be computed once for all. Characteristic information contained in this file concerns numerical integration formulae (coordinates, weights) and values of the shape functions and their derivatives at integration points on the reference element.

The content of a specific array is the responsibility of the programmer of a given finite element. The description of each specific element include the name of arrays used in POBA and their content listed in a systematic manner.

A typical array in POBA contains

NDIM	space dimension
NBPOLY	number of shape functions
NPI	number of integration points
IPOIDS	pointer on presence of weights : 1 if they are present, 0 otherwise
IPOL	pointer on presence of shape functions : 1 if they are present, 0 otherwise
IDPOL	pointer on presence of derivatives of shape functions : 1 if they are present, 0 otherwise
ω_k	if IPOIDS = 1, the weights in the integration formulae $\omega_k(NPI)$
$[P]$	if IPOL = 1, the values of shape functions at integration points $P(NBPOLY, NPI)$
$[DP]$	if IDPOL = 1, the values of the derivatives of the shape functions at integration points $DP(NDIM, NBPOLY, NPI)$
β_k	coordinates of the integration points

4.2.2 Description of the finite elements

4.2.3 Tetraedra $P2 - P0$

Finite element: **TETR 3P20**

DESCRIPTION OF THE ELEMENT

- Space dimension : $3d$
- Unknowns : u_1, u_2 et u_3, the three displacements
- Variational formulation : displacement
- Identification number : 200013
- Geometry : 5 tetrahedra
- Interpolation : isoparametric $P2$-Lagrange for displacements, discontinous $P0$ for pressure (if incompressible)
- Number of points : 10
- Number of point types : 1
- The points are defined only by their coordinates: $NCACAR = 0$
- Number of nodes : 10
- Number of node types : 1
- Type of nodes : vertices and mid edges, 3 d.o.fs per node 'VN' for u_1, u_2 and u_3
- Points and nodes are identical : $NCOPNP = 1$
- Direct access file POBA : used
- Natural boundary conditions (D.S. BDCL) : u_i(node) = value
- User-defined subroutine: available (SPUVEL)

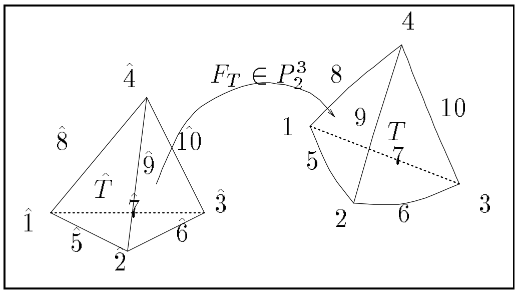

Figure 2: *Element TETR 3P20*

Coordinates of the reference element :

$\hat{S_1}$: $(0., 0., 0.)$ $\hat{S_2}$: $(1., 0., 0.)$ $\hat{S_3}$: $(0., 1., 0.)$ $\hat{S_4}$: $(0., 0., 1.)$ $\hat{S_5}$: $(0.5, 0., 0.)$ $\hat{S_6}$: $(0.5, 0.5, 0.)$ $\hat{S_7}$: $(0., 0.5, 0.)$ $\hat{S_8}$: $(0., 0., 0.5)$ $\hat{S_9}$: $(0.5, 0., 0.5)$ $\hat{S_{10}}$: $(0., 0.5, 0.5)$

NUMERICAL INTEGRATION

- Volume integration : Numerical integration 15 points.

  – $npiv = 15$

Points b_k	λ_1	λ_2	λ_3	λ_4	weighs ω_k
1	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{8}{405}$
2	b	a	a	a	α
3	a	b	a	a	α
4	a	a	b	a	α
5	a	a	a	b	α
6	d	c	c	c	β
7	c	d	c	c	β
8	c	c	d	c	β
9	c	c	c	d	β
10	e	e	f	f	γ
11	f	e	e	f	γ
12	f	f	e	e	γ
13	e	f	f	e	γ
14	e	f	e	f	γ
15	f	e	f	e	γ

Table 3P20 V : *15 points formula.*

  – with :
  $a = \frac{7-\sqrt{15}}{34} = 0.0919711$, $b = \frac{13+3\sqrt{15}}{34} = 0.7240868$, $c = \frac{7+\sqrt{15}}{34} = 0.3197936$,
  $d = \frac{13-3\sqrt{15}}{34} = 0.0406191$, $e = \frac{10-2\sqrt{15}}{40} = 0.0563508$, $f = \frac{10+2\sqrt{15}}{40} = 0.4436492$

  – $\alpha = \frac{2665+14\sqrt{15}}{226800}$, $\beta = \frac{2665-14\sqrt{15}}{226800}$ and $\gamma = \frac{5}{567}$

- Surface integration : Numerical integration 7 points.

  – $npis = 7$

Points b_k	λ_1	λ_2	λ_3	weights ω_k
1	c	d	c	α
2	d	c	c	α
3	c	c	d	α
4	b	b	a	β
5	a	b	b	β
6	b	a	b	β
7	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	γ

Tableau 3P20 S : *7 points formula.*

- with :
  $\gamma = \frac{9}{80} = 0.11250$, $\alpha = \frac{155-\sqrt{15}}{2400} = 0.06296959$
  and $\beta = \frac{155+\sqrt{15}}{2400} = 0.066197075$
- $a = \frac{9-2\sqrt{15}}{21} = 0.05961587$, $b = \frac{6+\sqrt{15}}{21} = 0.47014206$, $c = \frac{6-\sqrt{15}}{21} = 0.10128651$,
  $d = \frac{9+2\sqrt{15}}{21} = 0.797427$

DATA TO BE PROVIDED (in double precision)

- D.S MILI:
  - mass matrix $[M_T]$, not available
  - stiffness matrix $[K_T]$, array $\mathcal{V}$
    * E, ν if used with Saint-Venant Kirchoff energy
    * C_1, C_2, a if used with Ogden energy
    * C_1, C_2, ε if used with Mooney Rivlin energy
- D.S FORC: $NDSM$ right hand sides (in general in nonlinear problems NDSM=1).
  - Volume contribution $[B_T]$, array $\mathcal{V}$
    The parameter $NOTEL$ specify if a force is deformation-dependent
    NOTEL = 1 if the force is non-deformation dependent
    NOTEL = 7 if the force is deformation-dependent (such as pressure)

    * $f^{\Omega}(i,j,k)$ for $i = 1, NDSM$; $j = 1,3$; $k = 1, npiv$; the j component of the i right hand side at integration point k.
  - Surface contribution $[B_T]$, array $\mathcal{S}$
    * $f^{\Gamma}(i,j,k)$ for $i = 1, NDSM$; $j = 1,3$; $k = 1, npis$; the j component of the i right hand side at integration point k.

Remarks

- to use the user-defined subroutine SPUVEL you must add 20 to the value of $NOTEL$ indicated in the previous paragraph. The user provides this value when building D.S MILI and FORC (see modules COMILI, COFORC in [13])
- to deal with the subroutines MILIEU or FORCE you may need the values contained in POBA (see & 4.2.1). For this element the following arrays may be necessary :
  - array $3P25$: $NDIM = 3, NBPOLY = 10, NPIV = 15, 1, 1, 1, \omega_k, [P], [DP], b_k$
  - array $2P25$: same information as before for the faces $NDIM = 2, NBPOLY = 6, NPIS = 7, 1, 1, 1 \ldots$

4.2.4 Straight $Q2$ 20 nodes hexaedra

Finite element: **HEXA 3Q2D**

DESCRIPTION OF THE ELEMENT

- Space dimension : $3d$
- Unknowns : u_1, u_2 et u_3, the three displacements
- Variational formulation : displacement
- Identification number : 200017
- Geometry : 7 hexaedra
- Interpolation : $Q2$-Lagrange
- Number of points : 8
- Number of point types : 1
- The points are defined only by their coordinates: $NCACAR = 0$
- Number of nodes : 20
- Number of node types : 1
- Type of nodes : vertices and mid edges, 3 d.o.fs per node 'VN' for u_1, u_2 and u_3
- Points and nodes are distinct : $NCOPNP = 0$
- Direct access file POBA : used
- Natural boundary conditions (D.S. BDCL) : u_i(node) = value
- User-defined subroutine: available (SPUVEL)

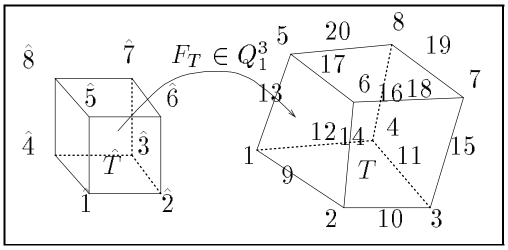

Figure 3: *Element HEXA 3Q2D*

Coordinates of the reference element :

$\hat{S}_1$: (0., 0., 0.) $\hat{S}_2$: (1., 0., 0.) $\hat{S}_3$: (1., 1., 0.) $\hat{S}_4$: (0., 1., 0.) $\hat{S}_5$: (0., 0., 1.) $\hat{S}_6$: (1., 0., 1.) $\hat{S}_7$: (1., 1., 1.) $\hat{S}_8$: (0., 1., 1.)

NUMERICAL INTEGRATION

- Volume integration : Numerical integration 27 Gauss points.
  - $npiv = 27$
  - $\omega_l = w_i w_j w_k$ for $i, j, k = 1, 3$
    with $w_1 = w_3 = \frac{5}{18}$ and $w_2 = \frac{8}{18}$
  - $b_l = (\alpha_i, \alpha_j, \alpha_k)$ for $i, j, k = 1, 3$
    with $\alpha_1 = \frac{1-\sqrt{\frac{3}{5}}}{2}$, $\alpha_2 = 0.5$ and $\alpha_3 = \frac{1+\sqrt{\frac{3}{5}}}{2}$
- Surface integration : Numerical integration 9 Gauss points.
  - $npis = 9$
  - $\omega_k = w_i w_j$ pour $i, j = 1, 3$
    with w_i as above
  - $b_k = (\alpha_i, \alpha_j)$ pour $i, j = 1, 3$
    with $\alpha_1 = \frac{1-\sqrt{\frac{3}{5}}}{2}$, $\alpha_2 = 0.5$ and $\alpha_3 = \frac{1+\sqrt{\frac{3}{5}}}{2}$

DATA TO BE PROVIDED (in double precision)

- D.S MILI:
  - mass matrix $[M_T]$, array $\mathcal{V}$
    * ρ the density, supposed constant by element.
  - stiffness matrix $[K_T]$, array $\mathcal{V}$
    * E, ν if used with Saint-Venant Kirchoff energy
    * C_1, C_2, a if used with Ogden energy
- D.S FORC: $NDSM$ right hand sides (in general in nonlinear problems NDSM=1).
  - Volume contribution $[B_T]$, array $\mathcal{V}$
    * $f^{\Omega}(i, j, k)$ for $i = 1, NDSM$; $j = 1, 3$; $k = 1, npiv$; the j component of the i right hand side at integration point k.
  - Surface contribution $[B_T]$, array $\mathcal{S}$
    * $f^{\Gamma}(i, j, k)$ for $i = 1, NDSM$; $j = 1, 3$; $k = 1, npis$; the j component of the i right hand side at integration point k.

Remarks

- to use the user-defined subroutine SPUVEL you must add 20 to the value of $NOTEL$ indicated in the previous paragraph. The user provides this value when building D.S MILI and FORC (see modules COMILI, COFORC in [13])
- to deal with the subroutines MILIEU or FORCE you may need the values contained in POBA (see & 4.2.1). For this element the following arrays may be necessary :
  - array $3Q25$ (2274 values) : $NDIM = 3, NBPOLY = 20, NPIV = 27, 1, 1, 1,$ $\omega_k, [P], [DP], b_k$
  - array $3Q23$ (978 values) : same information for straight elements i.e. $NBPOLY = 8$
  - array $2Q25$ (249 values) : same information as before for the faces $NDIM = 2, NBPOLY = 8, NPIS = 9$
  - array $2Q23$ (141 values) : same information as before for straight faces $NBPOLY = 4$

4.2.5 Curved $Q2$ 20 nodes hexaedra

Finite element : **HEXA 3Q2C**

DESCRIPTION OF THE ELEMENT

- Space dimension : $3d$
- Unknowns : u_1, u_2 et u_3, the three displacements
- Variational formulation : displacement
- Identification number : 200018
- Geometry : 7 hexaedra
- Interpolation : isoparametric $Q2$-Lagrange for displacements,
- Number of points : 20
- Number of point types : 1
- The points are defined only by their coordinates: $NCACAR = 0$
- Number of nodes : 20
- Number of node types : 1
- Type of nodes : vertices and mid edges, 3 d.o.fs per node 'VN' for u_1, u_2 and u_3
- Points and nodes are identical : $NCOPNP = 1$
- Direct access file POBA : used
- Natural boundary conditions (D.S. BDCL) : u_i(node) = value
- User-defined subroutine: available (SPUVEL)

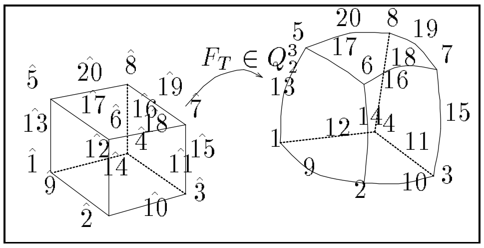

Figure 4: *Elément HEXA 3Q2C*

Coordinates of the reference element :

$\hat{S_1}$: $(0., 0., 0.)$ $\hat{S_2}$: $(1., 0., 0.)$ $\hat{S_3}$: $(1., 1., 0.)$ $\hat{S_4}$: $(0., 1., 0.)$ $\hat{S_5}$: $(0., 0., 1.)$ $\hat{S_6}$: $(1., 0., 1.)$ $\hat{S_7}$: $(1., 1., 1.)$ $\hat{S_8}$: $(0., 1., 1.)$ $\hat{S_9}$: $(0.5, 0., 0.)$ $\hat{S_{10}}$: $(1., 0.5, 0.)$ $\hat{S_{11}}$: $(0.5, 1., 0.)$ $\hat{S_{12}}$: $(0., 0.5, 0.)$ $\hat{S_{13}}$: $(1., 0., 0.5)$ $\hat{S_{14}}$: $(1., 1., 0.5)$ $\hat{S_{15}}$: $(0., 1., 0.5)$ $\hat{S_{16}}$: $(0., 0., 0.5)$ $\hat{S_{17}}$: $(0.5, 0., 1.)$ $\hat{S_{18}}$: $(1., 0.5, 1.)$ $\hat{S_{19}}$: $(0.5, 1., 1.)$ $\hat{S_{20}}$: $(0., 0.5, 1.)$

NUMERICAL INTEGRATION

- Volume integration : Numerical integration 27 Gauss points.
  - $npiv = 27$
  - $\omega_l = w_i w_j w_k$ for $i, j, k = 1, 3$
    with $w_1 = w_3 = \frac{5}{18}$ and $w_2 = \frac{8}{18}$
  - $b_l = (\alpha_i, \alpha_j, \alpha_k)$ for $i, j, k = 1, 3$
    with $\alpha_1 = \frac{1-\sqrt{\frac{3}{5}}}{2}$, $\alpha_2 = 0.5$ and $\alpha_3 = \frac{1+\sqrt{\frac{3}{5}}}{2}$
- Surface integration : Numerical integration 9 Gauss points.
  - $npis = 9$
  - $\omega_k = w_i w_j$ pour $i, j = 1, 3$
    with w_i as above
  - $b_k = (\alpha_i, \alpha_j)$ pour $i, j = 1, 3$
    with $\alpha_1 = \frac{1-\sqrt{\frac{3}{5}}}{2}$, $\alpha_2 = 0.5$ and $\alpha_3 = \frac{1+\sqrt{\frac{3}{5}}}{2}$

- D.S MILI:
  - mass matrix $[M_T]$, array $\mathcal{V}$
    - ρ the density, supposed constant by element.
  - stiffness matrix $[K_T]$, array $\mathcal{V}$
    - E, ν if used with Saint-Venant Kirchoff energy
    - C_1, C_2, a if used with Ogden energy
- D.S FORC: $NDSM$ right hand sides (in general in nonlinear problems NDSM=1).
  - Volume contribution $[B_T]$, array $\mathcal{V}$
    - $f^{\Omega}(i, j, k)$ for $i = 1, NDSM$; $j = 1, 3$; $k = 1, npiv$; the j component of the i right hand side at integration point k.
  - Surface contribution $[B_T]$, array $\mathcal{S}$
    - $f^{\Gamma}(i, j, k)$ for $i = 1, NDSM$; $j = 1, 3$; $k = 1, npis$; the j component of the i right hand side at integration point k.

Remarks

- the isoparametric element uses the same data as the straight one (**HEXA 3Q2D**), and also same arrays of POBA (see below)

- to use the user-defined subroutine SPUVEL you must add 20 to the value of $NOTEL$ indicated in the previous paragraph. The user provides this value when building D.S MILI and FORC (see modules COMILI, COFORC in [13])
- to deal with the subroutines MILIEU or FORCE you may need the values contained in POBA (see & 4.2.1). For this element the following arrays may be necessary :
  - array $3Q25$ (2274 values) : $NDIM = 3, NBPOLY = 20, NPIV = 27, 1, 1, 1$, $\omega_k, [P], [DP], b_k$
  - array $2Q25$ (249 values) : same information as before for the faces $NDIM = 2, NBPOLY = 8, NPIS = 9$

4.2.6 Curved $Q2 - P1$ 27 nodes hexaedra

Finite element : **HEXA 3QCC**

DESCRIPTION OF THE ELEMENT

- Space dimension : $3d$
- Unknowns : u_1, u_2 et u_3, the three displacements
- Variational formulation : displacement
- Identification number : 200019
- Geometry : 7 hexaedra
- Interpolation : isoparametric $Q2$-Lagrange for displacements, discontinous $P1$ for pressure (if incompressible)
- Number of points : 27
- Number of point types : 1
- The points are defined only by their coordinates: $NCACAR = 0$
- Number of nodes : 27
- Number of node types : 1
- Type of nodes : vertices and mid edges, 3 d.o.fs per node 'VN' for u_1, u_2 and u_3
- Points and nodes are identical : $NCOPNP = 1$
- Direct access file POBA : used
- Natural boundary conditions (D.S. BDCL) : u_i(node) = value
- User-defined subroutine: available (SPUVEL)

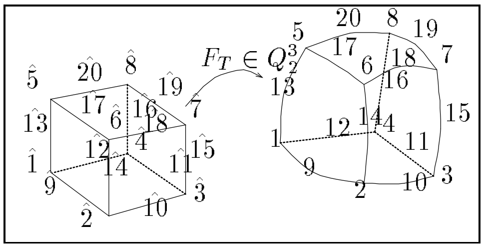

Figure 5: *Elément HEXA 3Q2C*

Coordinates of the reference element :

$\hat{S_1}$: $(0., 0., 0.)$ $\hat{S_2}$: $(1., 0., 0.)$ $\hat{S_3}$: $(1., 1., 0.)$ $\hat{S_4}$: $(0., 1., 0.)$ $\hat{S_5}$: $(0., 0., 1.)$ $\hat{S_6}$: $(1., 0., 1.)$ $\hat{S_7}$: $(1., 1., 1.)$ $\hat{S_8}$: $(0., 1., 1.)$ $\hat{S_9}$: $(0.5, 0., 0.)$ $\hat{S_{10}}$: $(1., 0.5, 0.)$ $\hat{S_{11}}$: $(0.5, 1., 0.)$ $\hat{S_{12}}$: $(0., 0.5, 0.)$ $\hat{S_{13}}$: $(1., 0., 0.5)$ $\hat{S_{14}}$: $(1., 1., 0.5)$ $\hat{S_{15}}$: $(0., 1., 0.5)$ $\hat{S_{16}}$: $(0., 0., 0.5)$ $\hat{S_{17}}$: $(0.5, 0., 1.)$ $\hat{S_{18}}$: $(1., 0.5, 1.)$ $\hat{S_{19}}$: $(0.5, 1., 1.)$ $\hat{S_{20}}$: $(0., 0.5, 1.)$ $\hat{S_{21}}$: $(0.5, 0.5, 0.)$ $\hat{S_{22}}$: $(0., 0.5, 0.5)$ $\hat{S_{23}}$: $(0.5, 0., 0.5)$ $\hat{S_{24}}$: $(0.5, 0.5, 1.)$ $\hat{S_{25}}$: $(1., 0.5, 0.5)$ $\hat{S_{26}}$: $(0.5, 1., 0.5,)$ $\hat{S_{27}}$: $(0.5, 0.5, 0.5)$

NUMERICAL INTEGRATION

- Volume integration : Numerical integration 27 Gauss points.
  - $npiv = 27$
  - $\omega_l = w_i w_j w_k$ for $i, j, k = 1, 3$
    with $w_1 = w_3 = \frac{5}{18}$ and $w_2 = \frac{8}{18}$
  - $b_l = (\alpha_i, \alpha_j, \alpha_k)$ for $i, j, k = 1, 3$
    with $\alpha_1 = \frac{1-\sqrt{\frac{3}{5}}}{2}$, $\alpha_2 = 0.5$ and $\alpha_3 = \frac{1+\sqrt{\frac{3}{5}}}{2}$
- Surface integration : Numerical integration 9 Gauss points.
  - $npis = 9$
  - $\omega_k = w_i w_j$ pour $i, j = 1, 3$
    with w_i as above
  - $b_k = (\alpha_i, \alpha_j)$ pour $i, j = 1, 3$
    with $\alpha_1 = \frac{1-\sqrt{\frac{3}{5}}}{2}$, $\alpha_2 = 0.5$ and $\alpha_3 = \frac{1+\sqrt{\frac{3}{5}}}{2}$

- D.S MILI:
  - mass matrix $[M_T]$, array $\mathcal{V}$
    * ρ the density, supposed constant by element.
  - stiffness matrix $[K_T]$, array $\mathcal{V}$
    * E, ν if used with Saint-Venant Kirchoff energy
    * C_1, C_2, a if used with Ogden energy
    * C_1, C_2, ε if used with Mooney Rivlin energy
- D.S FORC: $NDSM$ right hand sides (in general in nonlinear problems NDSM=1).
  - Volume contribution $[B_T]$, array $\mathcal{V}$
    * $f^{\Omega}(i, j, k)$ for $i = 1, NDSM$; $j = 1, 3$; $k = 1, npiv$; the j component of the i right hand side at integration point k.
  - Surface contribution $[B_T]$, array $\mathcal{S}$
    * $f^{\Gamma}(i, j, k)$ for $i = 1, NDSM$; $j = 1, 3$; $k = 1, npis$; the j component of the i right hand side at integration point k.

Remarks

- the isoparametric element uses the same data as the straight one (**HEXA 3Q2D**), and also same arrays of POBA (see below)

- to use the user-defined subroutine SPUVEL you must add 20 to the value of $NOTEL$ indicated in the previous paragraph. The user provides this value when building D.S MILI and FORC (see modules COMILI, COFORC in [13])
- to deal with the subroutines MILIEU or FORCE you may need the values contained in POBA (see & 4.2.1). For this element the following arrays may be necessary :
  - array $3Q25$ (2274 values) : $NDIM = 3, NBPOLY = 20, NPIV = 27, 1, 1, 1,$ $\omega_k, [P], [DP], b_k$
  - array $2Q25$ (249 values) : same information as before for the faces $NDIM = 2, NBPOLY = 8, NPIS = 9$

4.2.7 Straight $R2$ pentaedra

Finite element: **PENT 3R2D**

DESCRIPTION OF THE ELEMENT

- Space dimension : $3d$
- Unknowns : u_1, u_2 et u_3, the three displacements
- Variational formulation : displacement
- Identification number : 200022
- Geometry : 6 pentahedra
- Interpolation : $R2$-Lagrange
- Number of points : 6
- Number of point types : 1
- The points are defined only by their coordinates: $NCACAR = 0$
- Number of nodes : 15
- Number of node types : 1
- Type of nodes : vertices and mid edges, 3 d.o.fs per node 'VN' for u_1, u_2 and u_3
- Points and nodes are distinct : $NCOPNP = 0$
- Direct access file POBA : used
- Natural boundary conditions (D.S. BDCL) : u_i(node) = value
- User-defined subroutine: available (SPUVEL)

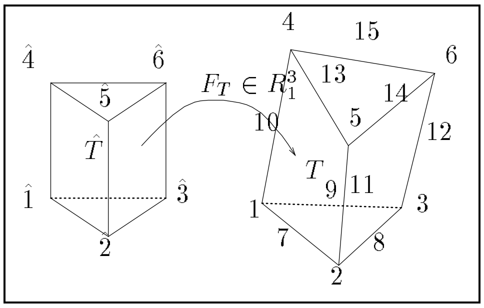

Figure 6: *Element PENT 3R2D*

Coordinates of the reference element :

$\hat{S}_1 : (0., 0., 0.)$ $\hat{S}_2 : (1., 0., 0.)$ $\hat{S}_3 : (0., 1., 0.)$ $\hat{S}_4 : (0., 0., 1.)$ $\hat{S}_5 : (1., 0., 1.)$ $\hat{S}_6 : (0., 1., 1.)$

NUMERICAL INTEGRATION

- Volume integration : Numerical integration 21 Gauss points.

  – $npiv = 21$

Points b_k	x	y	z	poids ω_k
1	d	d	e	$\alpha.\beta$
2	b	d	e	$\alpha.\beta$
3	d	b	e	$\alpha.\beta$
4	c	a	e	$\gamma.\beta$
5	c	c	e	$\gamma.\beta$
6	a	c	e	$\gamma.\beta$
7	$\frac{1}{3}$	$\frac{1}{3}$	e	$\delta.\beta$
8	d	d	f	$\alpha.\epsilon$
9	b	d	f	$\alpha.\epsilon$
10	d	b	f	$\alpha.\epsilon$
11	c	a	f	$\gamma.\epsilon$
12	c	c	f	$\gamma.\epsilon$
13	a	c	f	$\gamma.\epsilon$
14	$\frac{1}{3}$	$\frac{1}{3}$	f	$\delta.\epsilon$
15	d	d	g	$\alpha.\beta$
16	b	d	g	$\alpha.\beta$
17	d	b	g	$\alpha.\beta$
18	c	a	g	$\gamma.\beta$
19	c	c	g	$\gamma.\beta$
20	a	c	g	$\gamma.\beta$
21	$\frac{1}{3}$	$\frac{1}{3}$	g	$\delta.\beta$

Table 3R2D : *21 points formula.*

  – with : $a = \frac{9-2\sqrt{15}}{21}$, $b = \frac{9+2\sqrt{15}}{21}$, $c = \frac{6+\sqrt{15}}{21}$, $d = \frac{6-\sqrt{15}}{21}$, $e = 0.5(1-\sqrt{\frac{3}{5}})$, $f = 0.5$ and $g = 0.5(1+\sqrt{\frac{3}{5}})$

  – $\alpha = \frac{155-\sqrt{15}}{2400}$, $\beta = \frac{5}{18}$, $\gamma = \frac{155+\sqrt{15}}{2400}$, $\delta = \frac{9}{80}$ and $\epsilon = \frac{8}{18}$

- Surface integration : Numerical integration 7 or 9 points.

  – $npis = 7$ for a triangular face (see $TETR3P2$)) and $npis = 9$ for a quadrangular face (see $HEXA3Q2D$).

DATA TO BE PROVIDED (in double precision)

- D.S MILI:

- mass matrix $[M_T]$, array $\mathcal{V}$
  * ρ the density, supposed constant by element.
- stiffness matrix $[K_T]$, array $\mathcal{V}$
  * E, ν if used with Saint-Venant Kirchoff energy
  * C_1, C_2, a if used with Ogden energy

- D.S FORC: $NDSM$ right hand sides (in general in nonlinear problems NDSM=1).
  - Volume contribution $[B_T]$, array $\mathcal{V}$
    * $f^{\Omega}(i,j,k)$ for $i = 1, NDSM$; $j = 1, 3$; $k = 1, npiv$; the j component of the i right hand side at integration point k.
  - Surface contribution $[B_T]$, array $\mathcal{S}$
    * $f^{\Gamma}(i,j,k)$ for $i = 1, NDSM$; $j = 1, 3$; $k = 1, npis$; the j component of the i right hand side at integration point k.

Remarks

- to use the user-defined subroutine SPUVEL you must add 20 to the value of $NOTEL$ indicated in the previous paragraph. The user provides this value when building D.S MILI and FORC (see modules COMILI, COFORC in [13])
- to deal with the subroutines MILIEU or FORCE you may need the values contained in POBA (see & 4.2.1). For this element the following arrays may be necessary :
  - array $3R25$ (1350 values) : $NDIM = 3, NBPOLY = 15, NPIV = 21, 1, 1, 1, \omega_k, [P], [DP], b_k$
  - array $2Q25$ (249 values) : same information for quadrangular faces with $NDIM = 2, NBPOLY = 8, NPIS = 9$
  - array $2P25$ (153 values) : same information for triangular faces with $NDIM = 2, NBPOLY = 6, NPIS = 7$
  - array $3R23$ (594 values) : as in array $3R25$ for straight elements $NDIM = 3, NBPOLY = 6, NPIV = 21$
  - array $2P23$ (90 values) : same information for triangular faces $NDIM = 2, NBPOLY = 3, NPIS = 7$
  - array $2Q23$ (141 values) : same information for quadrangular faces $NDIM = 2, NBPOLY = 4, NPIS = 9$

5 Description of the *continu* software

The *continu* package is an implementation of the iterative algorithm described in §2.2. The aim here is to roughly explain how the software is organized to make it possible to easily modify it, in particular to change the choice of ν_n in (14) or to be able to solve the problem using quasi-Newton or related methods. The scope of the main subroutines of the software is presented hereafter :

The main program *continu* performs the following tasks :

- read data
- perform all operations that can be done once for all in **subroutine avnewt**. The subroutine **avnewt** does the following :
  - the interpolation **module COMACN**. Module COMACN is very closed to **COMACO** [13]. The difference is one additional parameter, the name of the subroutine containing the definition of the elements considered. (in the actual version of the software **subroutine dobnew**).
  - compute matrix pointers **module PREPAC** ([9], [10])
- perform the algorithm described in §2.2 with subroutine **contnw**
  - initialize data structures in **subroutine newt0**
  - initialize arrays
  - implement the algorithm using **subroutine regnew** to solve regular Newton problems and **subroutine extnew** for extended Newton problems. Both **regnew** and **extnew** routines will :
    - compute and assemble the tangent matrix **subroutine thens1**
    - impose boundary conditions on the matrix **subroutine clmuad**
    - factorize the matrix in Crout form ([9], [10]) **subroutine crmc1d**
    - perform forward backward substitutions **subroutine drcr1d** ([9], [10])
    - verify the **stopping criteria**.

This is an attempt to indicate the **classical** changes to adapt the algorithm to a particular problem :

- in subroutine *continu*
  - to change the size of the problem modify value of **LM** in the *PARAMETER* instruction (second line).
  - to change the internal energy or the user subroutines modify the name of the function in the *EXTERNAL* sentence at the beginning of the program.

  - to modify the value of ε used to stop iterations. In the program the variable ε is called **epsil**.
  - the **common /typeloi/ loii(0:25)** is there only to make available the information concerning the entire set of stored energies for the whole domain.

- in subroutine **contnw**, to modify parameter ν_n of (14)
  - define number of iterations for **easy convergence : npetit** and number of iterations for **regular convergence : nreg** in the *PARAMETER* instruction
  - new values of ν_n just after the solution of the extended Newton problem. It is clear that the values of ν_n will have an influence on the computation of the new increment. If it is too small the convergence will be slow, on the other hand if it is too big the problem may not converge.

- in subroutines **regnew**, **extnew** modify the **stopping criteria**. It is well known that it is difficult to have a very robust stopping criteria. The one used here is a function of the relative error on the increment **enors** and the error in the right hand side **ernorr**. Both values are printed during the execution of the software and, in case of a good behavior of the algorithm both are small at convergence.

6 Examples

The problem to be solved is the Rivlin cube. The interest of this problem is double, on the one hand the geometry is very simple, and on the other hand for small forces an analytical solution is known [8]. This make it possible to easy evaluate new elements. The cube is made of a compressible hyper-elastic material (the internal energy is given by (2)) and is subjected to a normal traction on each face. The intensity of these forces is the same for a pair of parallel faces. For symmetry reasons only one eighth of the object is considered.

The aim is thus to solve problem (1) with

- Ω is the cube $[0,1]^3$
- $f^\Omega = 0$
- f^Γ is given by

$$\begin{aligned} f^\Gamma &= \{g_1, 0, 0\} \text{ on the face } x = 1 \\ &= \{0, g_2, 0\} \text{ on the face } y = 1 \\ &= \{0, 0, g_3\} \text{ on the face } z = 1 \end{aligned}$$

with

$$\begin{aligned} g_1 &= \lambda_1(E_1 + E_2 * (\lambda_1^2 + \lambda_3^2) + E_3\lambda_1^2\lambda_3^2) \\ g_2 &= \lambda_2(E_1 + E_2 * (\lambda_1^2 + \lambda_3^2) + E_3\lambda_1^2\lambda_3^2 \\ g_3 &= \lambda_3(E_1 + E_2 * (\lambda_2^2 + \lambda_1^2) + E_3\lambda_2^2\lambda_1^2 \end{aligned}$$

and finally if C_1, C_2, a are the constants in (2) :

$$C_1 = 0.5 \quad C_2 = 0.0056 \quad a = 0.3736$$

$$\lambda_1 = 1.1 \quad \lambda_2 = 1.2 \quad \lambda_3 = 1.3$$

$$E_1 = 2C_1 \quad E_2 = 2C_2 \quad E_3 = 2a - \frac{(2C_1 + 4C_2 + 2a)}{\lambda_1^2\lambda_2^2\lambda_3^2}$$

- and symmetry boundary conditions given by :

$$\begin{aligned} u_x &= 0 \text{ on the face } x = 0 \\ u_y &= 0 \text{ on the face } y = 0 \\ u_z &= 0 \text{ on the face } z = 0 \end{aligned}$$

The exact solution of this problem is

$$u_x = .1x \quad u_y = .2y \quad u_z = .3z$$

The values g_1, g_2, g_3 are constant. These rather complicated expressions allow to change the problem under consideration. If λ_i is given, the displacement in the i direction will be $u_i = (\lambda_i - 1)x_i$. In addition if $\lambda_1\lambda_2\lambda_3 = 1$ it is also possible to derive and incompressible example by replacing E_3 with $E_3 = -p$ where p denotes the hydrostatic pressure.

To solve this problem one has to

- analyze the problem in order to give reference numbers to the mesh and decide how to provide data (arrays, subroutines MILIEU and/or FORCE ...)
- create Data Structures NOPO, MILI and FORC
- create the input data file
- run the software *continu*

The first step is commented hereafter. First consider the reference numbers; the aim here is to minimize the total number of references.

- Face reference numbers :
  - Three reference numbers are used on faces $x = 1$, $y = 1$ and $z = 1$ to give the loads. The reference number of edges or vertices are not used for tractions.
  - Three different reference numbers are used on faces $x = 0$, $y = 0$ and $z = 0$ to impose symmetry conditions.
- Edge reference number :
  - if the edge is the intersection of two faces submitted to a traction the reference number is irrelevant
  - if the edge is the intersection of two faces, one submitted to a traction and the other one with an imposed displacement (for the symmetry conditions) the edge has the reference number of the face where essential boundary conditions are imposed.
  - if the edge is the intersection of two faces where symmetry conditions are imposed the reference number of the edge is different of the one of the faces as both boundary conditions are needed for the edge.
- Vertex reference number are obtained using the same analysis as for edges considering that one vertex is the intersection of three edges

As the whole domain is a hexaedra, in the table hereafter the local numbering of edges, vertices and faces are employed to indicate the reference numbers used. The Modulef Guide 2 [3], [4] gives this local numbering with the description of the data structure NOPO.

```
FACE REFERENCES    :      2      6      9      3      8     7

EDGE REFERENCES    :      1      2      2      4     10      9
                          7      6      9      3      3      6

VERTEX REFERENCES  :      5      1      2      4     10     9
                          3      6
```

It is necessary to choose a finite element before the mesh is generated as the nodes which are not vertices are defined at this level. The choice here is the hexaedra HEXA 3Q2D, that means that once the topological mesh of the object is completed the key word ADPOIN will be used in the preprocessor apn3xx to generate nodes on the edges.

Several choices are available, for this test problem, to give material characteristics and loads. One of these possibilities is to give :

- C_1, C_2, a from (1) as **an array associated to D.S. MILI**. (named `/car` in the example below)

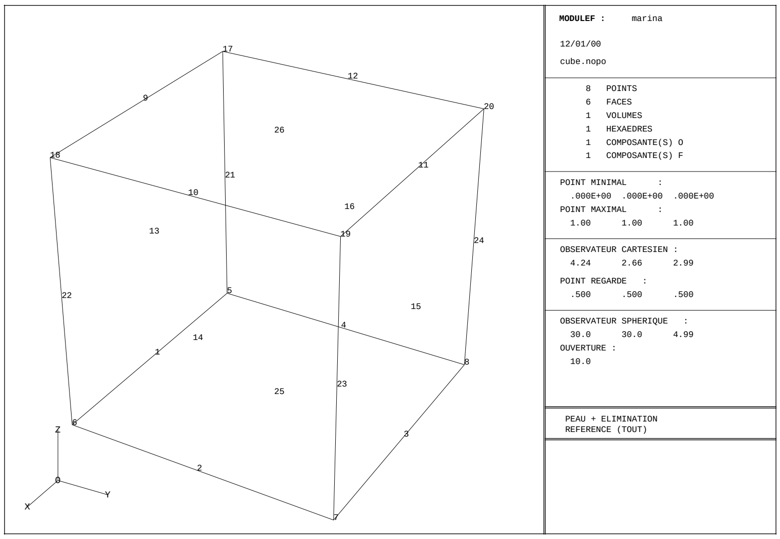

Figure 7: *Reference numbers on the Rivlin cube*

- f^Γ by using the **user subroutine FORCE**. In this case the subroutine has to define three functions, this will be done using three different options **LOPT** :

$$
\begin{aligned}
f^\Gamma &= \{g_1,0,0\} \text{ on the face } x=1, \text{ reference number 8, } \mathbf{LOPT{=}1} \\
&= \{0,g_2,0\} \text{ on the face } y=1, \text{ reference number 7, } \mathbf{LOPT{=}2} \\
&= \{0,0,g_3\} \text{ on the face } z=1, \text{ reference number 3, } \mathbf{LOPT{=}3}
\end{aligned}
$$

For this particular choice an interactive use of preprocessor **fomixx** with option **C** to create a data file, will produce the following input file :

```
'                                                              '
 q2.FORC                                 $ NOM DU FICHIER DE LA S.D FORC
```

```
    1     0                          $ SON NIVEAU ET NB DE SES TAB. ASSOCIES
 q2.MILI                             $ NOM DU FICHIER DE LA S.D MILI
    1     1                          $ SON NIVEAU ET NB DE SES TAB. ASSOCIES
 $  DONNEES RELATIVES A LA S.D. FORC  $
    1                                 $ NDSM
    0     3     0     0     0     0     0     0
     8    0     1     2               $ NOSD NFRO NOPT ITRAIT
    1     1                           $ LOPT IADR
     7     0     1     2               $ NOSD NFRO NOPT ITRAIT
    2     1                           $ LOPT IADR
     3     0     1     2               $ NOSD NFRO NOPT ITRAIT
    3     1                           $ LOPT IADR
 $  DONNEES RELATIVES A LA S.D. MILI  $
  /car       5       6                $ NOM TYPE NBREMOT
       0.500000000E+00                $ /car(  1)
       0.560000000E-02                $ /car(  2)
       0.373600000E+00                $ /car(  3)
  ' c1 c2 a                                                      '  $ CONTENU
    1     0     0     0     0     0     0     0
    1     0     1     1               $ NOSD NFRO NOPT ITRAIT
/car     1                            $ NTABL IADR
```

This file is then used by preprocessor **fomixx** with option **E** for execution and will produce the two data structures MILI and FORC. (names q2.MILI resp q2.FORC).

The user subroutine **FORCE** used is listed bellow.

```
      SUBROUTINE FORCE(M,LOPT,COOR,NDIM,NPO,TAR,LTAR,LV,IADR,I1,NARE,IA)
      INTEGER M(*)
      REAL COOR(NPO,NDIM)
      double precision  TAR(3,9)

c
c     9 integration poins on each face
c
      do 1 i = 1,9
       IF (LOPT .eq. 1) THEN
C
C        SURFACES 8
c
         TAR(1,i) = 1.53
         TAR(2,i) = 0.
         TAR(3,i) = 0.
C
       else if (lopt .eq. 2) then
C
C        SURFACES 7
c
```

```
          TAR(1,i) = 0.
          TAR(2,i) = 1.597848
          TAR(3,i) = 0.
C
        else if (lopt .eq. 3) then
C
C         SURFACES 3
c
          TAR(1,i) = 0.
          TAR(2,i) = 0.
          TAR(3,i) = 1.66985

        END IF
 1     continue
       END
```

It is also possible to write a more general subroutine **FORCE** suitable for a class of Rivlin problems on the cube, for example one used both for for tetrahedra and hexaedra, for compressible and incompressible materials.

```
      SUBROUTINE FORCE(M,LOPT,COOR,NDIM,NPO,TAR,LTAR,LV,IADR,I1,NARE,IA)
      common /typeloi/ loii(0:25)
      INTEGER M(*)
      REAL COOR(NPO,NDIM)
      double precision  TAR(3,*) ,e1,e2,e3,l1,l2,l3,lam,mu,E,nu

      l1 = 1.1
      l2 = 1.2
      l3 = 1.3
c
c     choice of elements
c     ------------------
c     hexaedra  : HEXA 2P2D (8 points)  HEXA 2P2C (20 points)
c                 HEXA 3QCC (27 points)
c     tetraedra : TETR 3P20 (10 points)
c
      if (npo .eq. 27 .or. npo .eq. 20 .or. npo. eq. 8) then
         npis = 9
      else
         npis = 7
      endif
      ndsde = m(ia+4)
      loi   = loii(ndsde)
      if (loi .eq. 1) then
c
c
c       COMPRESSIBLE
```

```
c       --------------
c
c
        E1=1.
        E2=0.0112
        E3=2*0.3736-1.7696/(l1*l1*l2*l2*l3*l3)

        do 1 i = 1,npis

           IF (LOPT .eq. 1) THEN
C
C             SURFACES 8
c
              TAR(1,i) = l1*(E1+E2*(l2*l2+l3*l3)+E3*l2*l2*l3*l3)
              TAR(2,i) = 0.
              TAR(3,i) = 0.
C
           else if (lopt .eq. 2) then
C
C             SURFACES 7
c
              TAR(1,i) = 0.
              TAR(2,i) = l2*(E1+E2*(l1*l1+l3*l3)+E3*l1*l1*l3*l3)
              TAR(3,i) = 0.
C
           else if (lopt .eq. 3) then
C
C             SURFACES 3
c
              TAR(1,i) = 0.
              TAR(2,i) = 0.
              TAR(3,i) = l3 *(E1+E2*(l1*l1+l2*l2)+E3*(l1*l1 *l2*l2))

           END IF
 1       continue


      else if (loi .eq. 2) then
c
c
c       INCOMPRESSIBLE
c       --------------
c
        E1=1.
        E2=0.0112
        E3=-1.
```

```
        do 10 i = 1,npis

           IF (LOPT .eq. 1) THEN
C
C             SURFACES 8
c
              TAR(1,i) = l1*(E1+E2*(l2*l2+0.757576*0.757576))+
     &                   E3*l2*0.757576
              TAR(2,i) = 0.
              TAR(3,i) = 0.
C
           else if (lopt .eq. 2) then
C
C             SURFACES 7
c
              TAR(1,i) = 0.
              TAR(2,i) = l2*(E1+E2*(l1*l1+0.757576*0.757576))+
     &                   E3*l1*0.757576
              TAR(3,i) = 0.
C
           else if (lopt .eq. 3) then
C
C             SURFACES 3
c
              TAR(1,i) = 0.
              TAR(2,i) = 0.
              TAR(3,i) = 0.757576*(E1+E2*(l1*l1+l2*l2))+E3*l1*l2
           END IF
 10     continue
      else if (loi .eq. 3) then
c
c    st venant
c
c
      E= 10000
      nu = .4
c
      lam = (e*nu)/((1+nu)*(1-2*nu))
      mu  = E/(2*(1+nu))
c
        do 3 i = 1,npis

           IF (LOPT .eq. 1) THEN
C
C             SURFACES 8
```

```
120  c
121               TAR(1,i) = .5*(lam+2*mu)*(l1*l1-1) +.5*lam*(l2*l2+l3*l3-2)
122               TAR(2,i) = 0.
123               TAR(3,i) = 0.
124  C
125            else if (lopt .eq. 2) then
126  C
127  C            SURFACES 7
128  c
129               TAR(1,i) = 0.
130               TAR(2,i) = .5*(lam+2*mu)*(l2*l2-1) +.5*lam*(l1*l1+l3*l3-2)
131               TAR(3,i) = 0.
132  C
133            else if (lopt .eq. 3) then
134  C
135  C            SURFACES 3
136  c
137               TAR(1,i) = 0.
138               TAR(2,i) = 0.
139               TAR(3,i) = .5*(lam+2*mu)*(l3*l3-1) +.5*lam*(l1*l1+l2*l2-2)
140
141            END IF
142   3     continue
143
144       end if
145       END
```

Comments on subroutine FORCE

- In this particular case both **FORCE** subroutines will produce the same result.
- Lines 7 - 9 define the problem, lines 31-33 (and 71-73) the constants
- The number of values to give is 3 per integration point. If the subroutine is used with several elements it is necessary to know the number of integration points used on a face for the actual element. In our case that is easy to know by checking the number of points.
- In lines 22-23 the information about the energy used (compressible or incompressible) is obtained

The user also has to give the function **valcld**. In this case this routine is trivial.

```
      DOUBLE PRECISION FUNCTION VALCLD(I,X,Y,Z)
C     ----------------------------------------
```

```
      VALCLD=0.D0
      RETURN
      END
```

The last step before running the software *continu* is to create the **data file**. For this particular problem the data file can take the following form

```
q2   $  regular name of files
0    $  algorithm initialized at  0 , 1 init by d.s. bini
1    $  ndsd (number of materials)
1    $  nfpoba is used
/usr/local/modulef/hp700/exp/etc/poba.direct
$ data for  comacn
3     $ndim
0 0   $nbsc nblc
ELAS
1
HEXA 3Q2D
0
$ data for cobdc1
 12                $ NBFR
1  2   4   5       $ NOFR
1  5  10   9       $ NOFR
5  4  10   6       $ NOFR

3  3   3   3       $ INC
2  2   2   2       $ INC
1  1   1   1       $ INC

VN VN VN VN
VN VN VN VN
VN VN VN VN
5   $ntype
$  data for thelas
2         $  nprov
0 1 1 0
3         $ nd
COUL      $ nom
1         $ LVECT
30        $ itemax
1         $ loi 1 compressible
1.
```

Comments on the input data file

- line 1 : the names of the D.S. files will be q2.NOPO, q2.MILI .... Notice this was the name chosen to construct data structures MILI and FORC in the example above.
- line 4 : as indicated in the file describing the finite element selected, the direct access file POBA is used.
- line 5 : name of the direct acces POBA file (depends on the implementation of the library on the specific computer used).
- lines 15-18 contain data used to impose boundary conditions. Notice that, as already mentioned in §3.3.2 the same reference number may appear several times. Here, for example, the node at the intersection of the three symmetry plans is referenced 5 and this reference number appears three times.
- line 36 : the problem will be solved by a standard Newton algorithm as the initial load increment is set to 1.
- the last line in the data is not necessary, this parameter is used only if the initialization is not zero and is used in the initialization of ω (see §2.2)

Hereafter a listing of the execution :
stiffness

```
echo resolv.data | continu
 ESPION: VERSION MODULEF  DU 17 Juin 1998

 M   M    OOO    DDDD    U   U   L       EEEEE   FFFFF
 MM MM   O   O   D   D   U   U   L       E       F
 M M M   O   O   D   D   U   U   L       EEEE    FFFF
 M   M   O   O   D   D   U   U   L       E       F
 M   M    OOO    DDDD     UUU    LLLLL   EEEEE   F      VERSION 96

 DATE   : 16/02/99
 AUTEUR : marina
 -- NOM DU FICHIER CONTENANT LES DONNEES  ?
 &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
 MODULE COMACO :
 FIN DU MODULE COMACO
 &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

 &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
 MODULE CONDL1 :
 NOMBRE TOTAL DE NOEUDS DES TYPES D ELEMENT   (LNOET) :    20
 NOMBRE MAXIMUM DE NOEUDS D UN ELEMENT       (NNOMAX) :    20
 NOMBRE CONSTANT DE D.L. EN TOUS LES NOEUDS OU 0 (ND) :     3
 NOMBRE MAXIMUM DE D.L. EN UN NOEUD          (NDLMAX) :     3
 NOMBRE DE NOEUDS PAR PAGE DE NDL1           (NNPSEG) :   425
```

```
OTABLEAU DU NOMBRE DE DEGRES DE LIBERTE EN CHAQUE NOEUD

 NDLN(  1)=  3   NDLN(  2)=  3   NDLN(  3)=  3   NDLN(  4)=  3   NDLN(  5)=  3
 NDLN(  6)=  3   NDLN(  7)=  3   NDLN(  8)=  3   NDLN(  9)=  3   NDLN( 10)=  3
 NDLN( 11)=  3   NDLN( 12)=  3   NDLN( 13)=  3   NDLN( 14)=  3   NDLN( 15)=  3
 NDLN( 16)=  3   NDLN( 17)=  3   NDLN( 18)=  3   NDLN( 19)=  3   NDLN( 20)=  3
OTOUS LES NOEUDS SUPPORTENT    3 DEGRES DE LIBERTE

 FIN DU MODULE CONDL1
 &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

 &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
 MODULE CORNOE :
 FIN DU MODULE CORNOE
 &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

 &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
 MODULE COBDC1 :
  %% ATTENTION COBDC1 :
    LES PARAMETRES IATA5 - IATA6 ONT CHANGES
    LE PARAMETRE  VALCLC  A ETE AJOUTE
 FIN DU MODULE COBDC1
 &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

 &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
 MODULE PREPAC :
 NOMBRE DE MOTS DE LA MATRICE       (LMUA5) :   588498
 DIFFERENCE MAX ENTRE 2 NOEUDS D'UN ELEMENT (LBDP) :      165
 DIFFERENCE MAX ENTRE 2 D.L. D'UN ELEMENT (LBDPDL) :      495
 NOMBRE MAX DE NOEUDS D'UN ELEMENT (NNOMAX) :       20
 NOMBRE DE SECONDS MEMBRES           (NDSM) :        1
 NOMBRE DE MOTS EN M.C. POUR LA S.D. NDL1 (MCNDL1) :        0
 NOMBRE DE MOTS EN M.C. POUR LA S.D. MUA   (MCMUA) :   589820
 NOMBRE DE MOTS EN M.C. POUR LA S.D. B       (MCB) :     2595

 FIN DU MODULE PREPAC
 &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
 avant newton 4.000000000000000E-02
 &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
 MODULE CONTNW :

 LE TABLEAU  3Q23 DE   1950 MOTS EST RESTAURE DANS M ( PAGES   40 A    47 DU FICHIER NFPOBA   -11 )

 LE TABLEAU  3Q25 DE   4542 MOTS EST RESTAURE DANS M ( PAGES   78 A    95 DU FICHIER NFPOBA   -11 )
```

```
LE TABLEAU  2Q25 DE    492 MOTS EST RESTAURE DANS M ( PAGES   10 A    11 DU FICHIER NFPOBA   -11 )

LE TABLEAU  2Q23 DE    276 MOTS EST RESTAURE DANS M ( PAGES    8 A     9 DU FICHIER NFPOBA   -11 )

TABLEAU  T A E 2
----------------

NOMBRE D'ELEMENTS                                 (NE) :    64
NOMBRE DE NOEUDS                                 (NOE) :   425
NOMBRE DE TABLEAUX ASSOCIES A CHAQUE ELEMENT   (NTACE) :     2
NOMBRE MAXIMUM DE NOEUDS D'UN ELEMENT         (NNOMAX) :    20
NOMBRE CONSTANT DE D.L. PAR NOEUD ( 0 SI NON )    (ND) :     3
NOMBRE MAXIMUM DE D.L. PAR NOEUD              (NDLMAX) :     3
NOMBRE DE TYPE D'ELEMENTS                     (NTYELM) :     1
SOMME DU NOMBRE DE NOEUDS DES TYPES D'ELEMENTS (LNOET) :    20

PROBLEME : 1 THERMIQUE, 2 ELASTIQUE, 3 AUTRE   (NPROV) :     2
LES OPTIONS CHOISIES                          (NOPTNT) :    -1


NUMERO DU TABLEAU ASSOCIE QUI REPRESENTE :
LA MASSE                                :     0
LA RAIDEUR                              :     1
LA PREMIERE COMBINAISON LINEAIRE        :     0
LA DERNIERE                             :     0
LE(S) SECOND(S) MEMBRE(S)               :     2
LES CONTRAINTES OU LE FLUX              :     0

 ntdl  1275
 fin de newt0
uini  .0
bfix avant c.l  .6436668327534161
bfix  .6436668327534161
      ernorr  .6436668327534161
      enors    5.27663586381763
 ITERATION :  1  erreur  2.23942844204955
      ernorr  .1942800683538641
      enors    2.41248464200825
 ITERATION :  2  erreur  .2530115166112537
      ernorr  1.646551142273279E-02
      enors    1.07127825210688
 ITERATION :  3  erreur  7.620125404312008E-02
      ernorr  3.742366803706144E-03
      enors    3.170851796477224E-02
 ITERATION :  4  erreur  3.941122119215049E-03
      ernorr  3.631617592004449E-06
```

```
       enors    2.539939021727116E-04
  ITERATION :  5  erreur  1.792588392600874E-05
 #####  prem newton CONVERGE APRES    5 ITERATIONS, RESIDU :     .1793D-04
 FIN DU MODULE CONTNW
 &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

 apres newton 31.47
31.51u 0.06s 0:31.98 98.7%
```

Finally line 36 of the input data file is modified. The initial load increment is fixed to .4 and the results obtained with the continuation algorithm are listed here after.

```
  ntdl  1275
  fin de newt0
 uini  .0
 bfix avant c.l  .2574667331013664
 bfix  .2574667331013664
       ernorr  .2574667331013662
       enors    2.11065434552705
  ITERATION :  1  erreur  1.04901704370856
       ernorr  2.549062498872416E-02
       enors    .1978360449097135
  ITERATION :  2  erreur  6.424586665881868E-02
       ernorr  2.078980365469744E-04
       enors    4.020470892078356E-03
  ITERATION :  3  erreur  9.041070176187770E-04
       ernorr  3.959372460006626E-08
       enors    1.024229040490265E-06
  ITERATION :  4  erreur  2.130160075175091E-07
 #####  prem newton CONVERGE APRES    4 ITERATIONS, RESIDU :     .2130D-06
 ndelu, omega,ds  3.8263054432781 23.9144090204881 2.76633528093689
       ++++ entree de extnew xlam  .4
            uini  1.95609443618607
            ernorr  3.073901162771188E-15
            enorq*2    20.86380816468056
       ++++ maj de xlam ds extnew  .8134008986933207
            bfix ds extnew  .5235591802207119
            ernorr  2.643188478849725E-02
            enors    .3115054757806357
            enorq*2    20.22657894752399
    ITERATION :  1  erreur  7.595562675465917E-02
       ++++ maj de xlam ds extnew  .8089743933499465
            ernorr  2.738861236387044E-04
            enors    1.590123512309121E-02
            enorq*2    22.04951619567505
    ITERATION :  2  erreur  2.980776892429193E-03
       ++++ maj de xlam ds extnew  .8099841893488203
```

```
             ernorr  5.264739364584434E-07
             enors    1.750851672028890E-05
             enorq*2    21.95045747962086
      ITERATION :  3  erreur  3.422845270481860E-06
        ++++ maj de xlam ds extnew  .8099850764229914
 #####  newton etendu CONVERGE EN  3 ITERATIONS, RESIDU :  .3423D-05 lamda  .8100D+00
        =====>  ds  3.59623573330881
        ++++ entree de extnew xlam  .8099850764229914
             uini  3.75807308202612
             ernorr  7.002433990710995E-13
             enorq*2    21.9503474983584
        ++++ maj de xlam ds extnew  1.34100229775661
             bfix ds extnew  .8631587017120519
             ernorr  3.849040015076172E-02
             enors    3.85043720940623
             enorq*2    323.6392750060239
      ITERATION :  1  erreur  .6127351881936446
        ++++ maj de xlam ds extnew  1.14621298127046
             ernorr  5.850080666125171E-03
             enors    .3387861017407943
             enorq*2    96.59095267406849
      ITERATION :  2  erreur  3.002169851595499E-02
        ++++ maj de xlam ds extnew  1.16978236294207
             ernorr  1.156678228750545E-04
             enors    7.875418101938887E-03
             enorq*2    96.00199213482094
      ITERATION :  3  erreur  5.613417630024396E-04
        ++++ maj de xlam ds extnew  1.1701199860583
             ernorr  1.433251938136533E-07
             enors    8.597569265192276E-06
             enorq*2    96.00579818282534
      ITERATION :  4  erreur  5.809590626025605E-07
        ++++ maj de xlam ds extnew  1.17012028033593
 #####  newton etendu CONVERGE EN  4 ITERATIONS, RESIDU :  .5810D-06 lamda  .1170D+01
  uini  4.75955770930083
  bfix avant c.l  .6436668327534161
  bfix  .6436668327534161
        ernorr  3.922836110918511E-03
        enors    .2564410426512324
   ITERATION :  1  erreur  3.516898309421063E-02
        ernorr  2.195269083137414E-04
        enors    1.695252781217216E-03
   ITERATION :  2  erreur  3.388825267972148E-04
        ernorr  1.222806640589436E-08
        enors    7.785508355314867E-07
   ITERATION :  3  erreur  3.332465694350095E-08
```

```
82    dern newton CONVERGE APRES    3 ITERATIONS, RESIDU :     .3332D-07
83    FIN DU MODULE CONTNW
84    &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
85
86    apres newton 98.26000000000001
87   98.31u 0.08s 1:40.98 97.4%
```

Note that the first part of the listing was removed since it is the same as for the standard Newton execution.

Comments on the output file

Find a good stoping criteria is not obvious in a Newton method. Let us discuss this topic before giving details concerning the output file. At each Newton iteration we mainly solve the problem

$$Ks = -r$$

with

- K the tangent matrix (given in equation (6)),
- r the residual (given in equation (7)),
- s the new displacement increment

At convergence, both r and s are small, which means that the convergence criterion may equally use increments for r or s. Nevertheless, in some particular situations the increment of one of these quantities is very small (satisfy a convergence criterion) before convergence is reached (the other quantity still vary significantly). To improve the robustness of the method the stopping criteria used by the *continu* software uses both r and q. More precisely, the algorithm stops when :

$$erreur = \frac{||r||\frac{||s_0||}{||r_0||} + ||s||}{(||b_{fix}|| + ||r_0||)\frac{||s_0||}{||r_0||} + (||u_{ini}|| + ||s_0||)} \leq \varepsilon \tag{15}$$

In equation (15) $||u_{ini}||$ is the L_2 norm of the initial solution and $b_{fix} = \int_\Omega f^\Omega \cdot vdx + \int_{\partial\Omega_2} f^\Gamma \cdot vda$.

- The specific outputs for the Newton method starts with line 99 for the first run and line 1 for the second one. On this line the total number of degrees of freedom (ntdl) is recalled.
- The standard input, for each iteration will be **ernorr** for $||r_0||$, **enors** for $||s||$ and **erreur** which is the stopping criterion (15)
- When the arc-length continuation algorithm is used some additional printings allow to follow the convergence process.

- line 19, after the initialization of the algorithm by a regular Newton, **ndelu** designates the L_2 norm of ΔU_1 (see (8), **ds** indicates Δs_1 in equation (9) and **omega** stands for ω (see equation (10))
- At the beginning of each extended-Newton algorithm (subroutine **extnew**, see §5) the actual value of λ, called **xlam** in the program is printed (see lines 20, 43 for example).
- During the extended-Newton algorithm this value is updated (see equation (12)) and printed (see lines 24, 30 ...)
- **enorq*2** denotes $(||Q||^2$.

References

[1] H du Toit. *Introduction à MODULEF*. INRIA, 1991. Guide Modulef 1.

[2] H du Toit. *An introduction to MODULEF*. INRIA, 1991. Modulef Guide 1.

[3] P.-L. George and P. Laug. *Normes d'utilisation et de programmation*. INRIA, 1992. Guide Modulef 2.

[4] P.-L. George and P. Laug. *Programming and Utilisation Standards*. INRIA, 1992. Modulef Guide 2.

[5] P.-L. George, P. Paté, and M. Vidrascu. *Fiches Techniques*. INRIA, 1993. Guide Modulef 7.

[6] P.L. George. *Construction et Modification de Maillages*. INRIA, 1991. Guide Modulef 3.

[7] P.L. George. *Mesh Construction and Modification*. INRIA, 1991. Modulef Guide 3.

[8] M. Gurtin. *Topics in finite elasticity*. SIAM Studies 35, 1981.

[9] P. Joly and M. Vidrascu. *Résolution de systèmes linéaires*. INRIA, 1993. Guide Modulef 5.

[10] P. Joly and M. Vidrascu. *Solution of linear systems*. INRIA, 1994. Modulef Guide 5.

[11] P. Le Tallec. *Handbook of numerical Analysis*, volume 3, chapter Numerical Methods for Nonlinear Three-Dimensional Elasticity, pages 465–622. North-Holland, P.G. Ciarlet and J.L. Lions edition, 1994.

[12] P. Paté and M. Vidrascu. *Post-traitements et graphiques*. INRIA, 1992. Guide Modulef 6.

[13] P. Paté and M. Vidrascu. *Interpolation - Création des tableaux élémentaires*. INRIA, 1993. Guide Modulef 2.

Unit de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unit de recherche INRIA Lorraine : LORIA, Technople de Nancy-Brabois - Campus scientiøque
615, rue du Jardin Botanique - BP 101 - 54602 Villers-ls-Nancy Cedex (France)
Unit de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unit de recherche INRIA Rhne-Alpes : 655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)
Unit de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

diteur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399