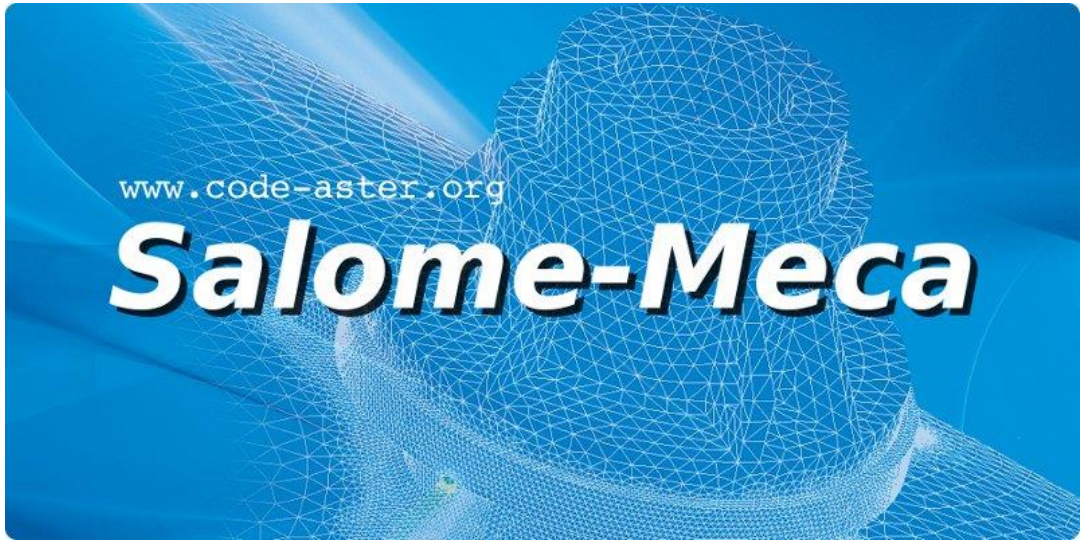
A simple mechanical study

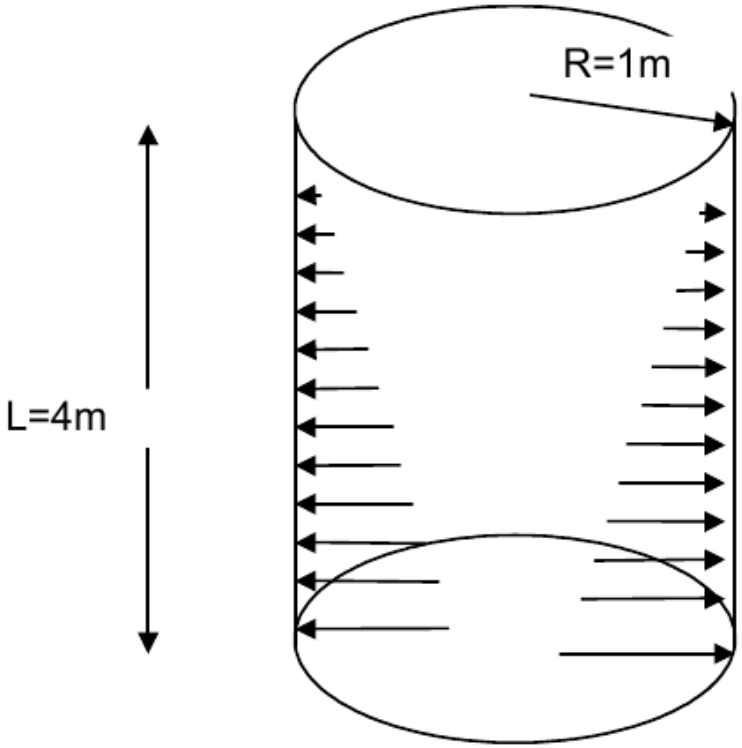


Code_Aster, Salome-Meca course material

code_aster step by step: outline

- A simple example
- Starting the study and acquiring the mesh
- Selecting, defining and assigning the modelling data
  - Models and finite elements
  - The materials properties
  - The characteristics of structural elements
  - The boundary conditions and loadings
- Solving the problem
  - The setting of solving operators
  - The linear solvers, parallel solvers and distributed computations
  - The results outputs
- Post-processing the results
  - Calculation of options and operations on results
  - Extraction of values and outputs

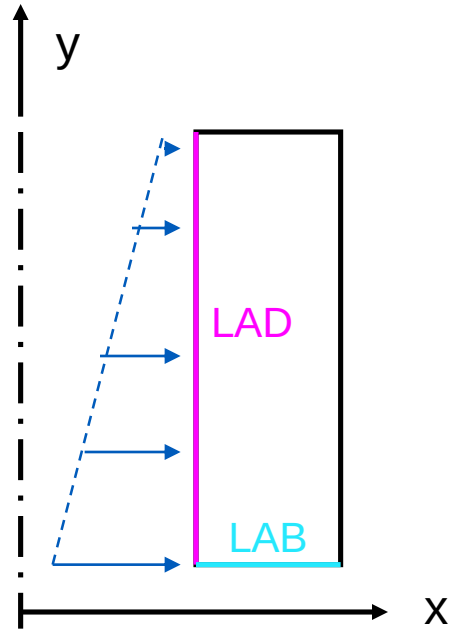
A simple example



Test case FORMA00A : [U1.05.00] "A simple example of use"

```
DEBUT()

# Reading of the mesh
mesh=LIRE_MAILLAGE(FORMAT='MED')


# Model definition
model=AFFE_MODELE(MAILLAGE=mesh,
                  AFFE=_F(TOUT='OUI',
                          PHENOMENE='MECANIQUE',
                          MODELISATION='AXIS',),)

# Definition of material properties
steel=DEFI_MATERIAU(ELAS=_F(E=2.1E11,
                            NU=0.3,),)

# Affectation of the material on the mesh
mater=AFFE_MATERIAU(MAILLAGE=mesh,
                    AFFE=_F(TOUT='OUI',
                            MATER=steel,),)

# Definition of boundary conditions
bc=AFFE_CHAR_MECA(MODELE=model,
                  FACE_IMPO=_F(GROUP_MA='LAB',
                               DY=0,),)

# Definition of loadings
f_y=DEFI_FONCTION(NOM_PARA='Y',
                  VALE=(0.,200000.,
                        4.,0.,),)

load=AFFE_CHAR_MECA_F(MODELE=model,
                      PRES_REP=_F(GROUP_MA='LDA',
                                  PRES=f_y,),)
```

```
# Resolve
result=MECA_STATIQUE(MODELE=model,
                     CHAM_MATER=mater,
                     EXCIT=(_F(CHARGE=load,),
                            _F(CHARGE=bc,),),)

# Stress Calculation at nodes
result=CALC_CHAMP(reuse=result,
                  RESULTAT=result,
                  CONTRAINTE='SIGM_ELNO')


# Print results for display in Salome
IMPR_RESU(FORMAT='MED',
          RESU=_F(RESULTAT=result))

FIN()
```



Starting the study and acquiring the mesh

Special commands: DEBUT, FIN, POURSUITE

- The **DEBUT** command
  - Begins execution, previous lines are ignored
  - Basic usage:

```
DEBUT()
```

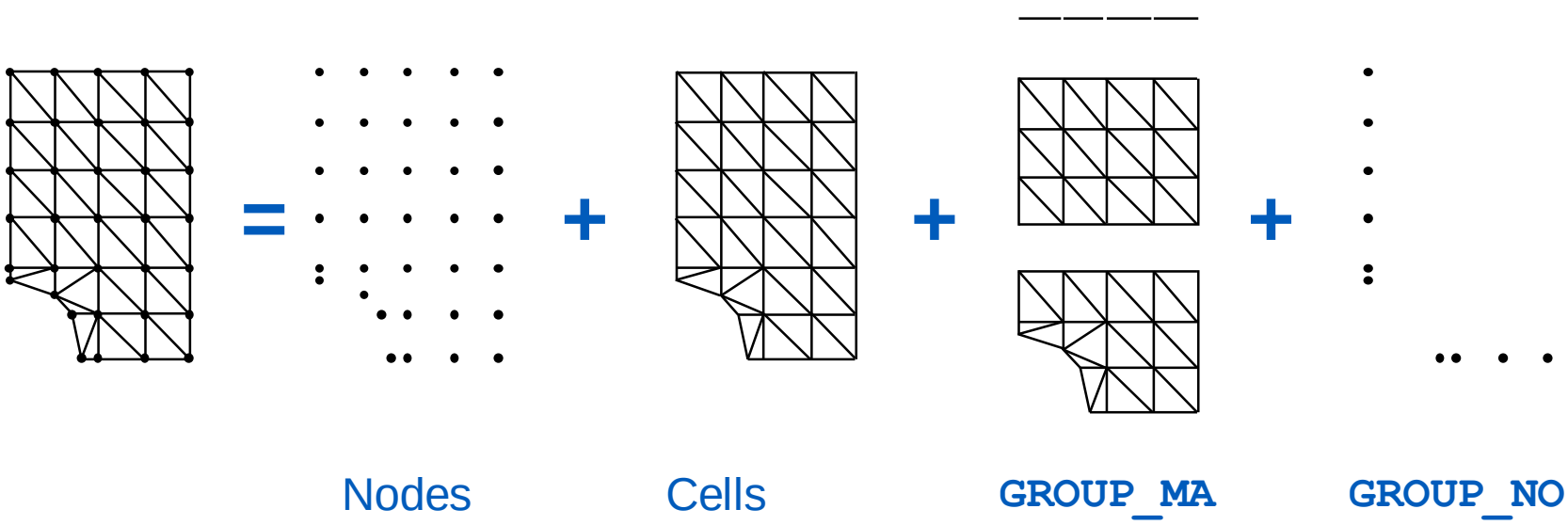
- The **POURSUITE** command
  - Restarts execution from a database provided as an input
  - Useful to continue a calculation initiated with the same version of code_aster
  - Recommended to separate the calculation from the post-processing
  - Basic usage:

```
Calculation:      DEBUT()  ...  FIN()
Post-processing:  POURSUITE()  ...  FIN()
```

- The **FIN** command
  - Ends the command file and ends the run, following lines are ignored
  - Closes the database at the end of execution: folder containing all the concepts generated during the calculation (mesh data, intermediate structures, results ...)
  - Specifies the format used for the produced base: HDF or Aster

What is a mesh?

- Coordinates of nodes
- Cells defined by their connectivity
- Groups of cells (`GROUP_MA`) and groups of nodes (`GROUP_NO`)



Reading the mesh

Meshes in Aster format or MED format

- code_aster format

```
mymesh = LIRE_MAILLAGE(FORMAT='ASTER')
```

- MED format (default input format)

```
mymesh = LIRE_MAILLAGE()
```

Meshes in other formats

- Commands **PRE_*** ** to convert to Aster format: **PRE_GIBI**, **PRE_IDEAS**, **PRE_GMSH**
- Example:

```
PRE_***()
mymesh = LIRE_MAILLAGE(FORMAT='ASTER')
```

Outputing the mesh (if modified for example)

- Command **IMPR_RESU**
- For instance in MED format (default output format):

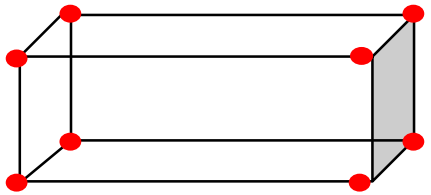
```
IMPR_RESU(  RESU=_F(MAILLAGE=mymesh,),)
```

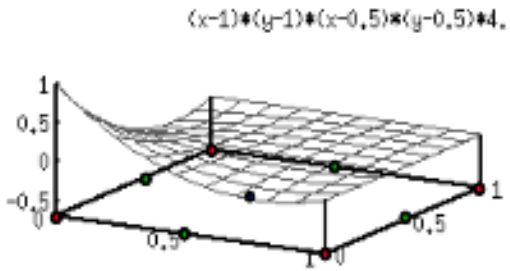
Selecting, defining and assigning the modelling data

The choice of finite elements

A finite element is:

- A geometric description, provided by the mesh
- Shape functions
- Degrees of freedom





Its choice determines:

- The equations that are solved
- Discretization and integration hypothesis
- The fields unknowns

```
mymodel = AFFE_MODELE(  MAILLAGE=mymesh,
                      AFFE=_F(    GROUP_MA          ='ZONE_1',
                                  PHENOMENE         ='MECANIQUE'
                         MODELISATION  ='C_PLAN'),)
```

The choice of finite elements

- **Example for mechanical phenomenon, 3D elements:**

```
♦ /   ♦ PHENOMENE =      'MECANIQUE'
  ♦  MODELISATION =
        /  '3D'
        /  '3D_SI'
        /  '3D_INCO'
        /  '3D_INCO_UP'
        /  '3D_INCO_UPG'
        /  '3D_FLUIDE'
        /  '3D_FAISCEAU'
        /  '3D_ABSO'
        /  '3D_FLUI_ABSO'
        /  '3D_GRAD_VARI'
        /  '3D_THM'...
```

3D isoparametric mechanical element

But also many other mechanical 3D elements:
- sub-integrated,
- incompressible,
- non-local,
- fluid-structure,
- thermo-hydro-mechanics,
- …

The choice of finite elements

- **Example for mechanical phenomenon, 2D, 1D, 0D elements:**

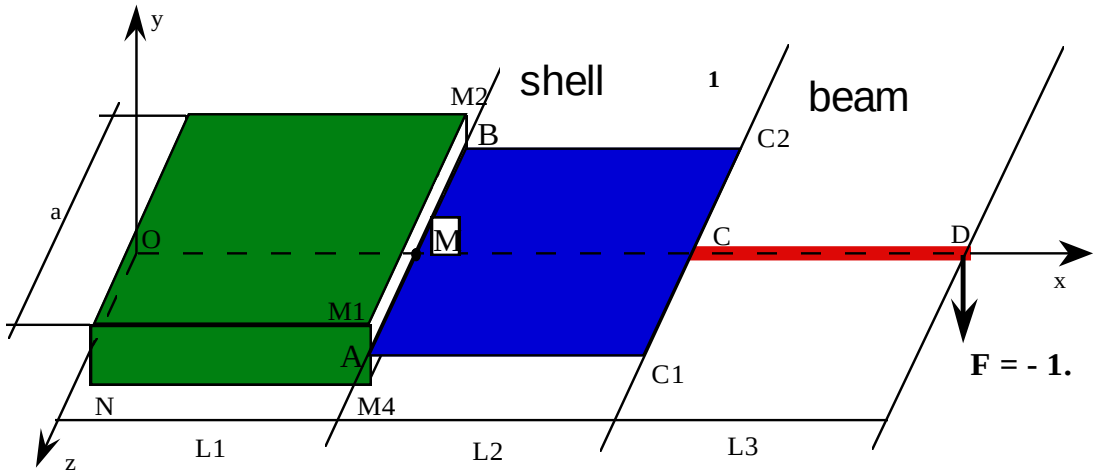
```
♦ /   ♦ PHENOMENE =      'MECANIQUE'
  ♦  MODELISATION =
        /  'D_PLAN'
        /  'C_PLAN'
        /  'AXIS'
                                    :
        /  'DKT'
        /  'DST'
        /  'COQUE_3D'
                                    :
        /  'POU_D_T'
        /  'TUYAU'
                                    :
```

2D mechanical elements
- plane strain
- plane stress
- Axi-symmetry

Surfacic mechanical elements
- plate elements
- plate elements with shear modelling
- shell elements

1D mechanical elements
- Beams
- Pipes
- Cables …

Mixture of finite elements: beware of connections




```
mo=AFFE_MODELE(
    MAILLAGE= ma,
    AFFE=(_F(MAILLE= he1,     PHENOMENE='MECANIQUE',MODELISATION='3D'),
          _F(GROUP_MA= grma1,PHENOMENE='MECANIQUE',MODELISATION='DKT'),
          _F(GROUP_MA= grma2,PHENOMENE='MECANIQUE',MODELISATION='POU_D_E')))
```

- One will have to write kinematic connections between degrees of freedom
  - See following chapter for loadings and boundary conditions

Other points to pay attention to:

- **Axi-symmetrical model `AXIS`:**
  - Node coordinates: ***y*** must be the axis of symmetry and the values of ***x*** must be positive
- **Orientation of the normals:**
  - Command **`MODI_MAILLAGE`** [U4.23.04]
  - Checks the orientation of the normal vector for plate/shell elements as well as for the borders where a pressure is applied (2D/3D)
  - Reorients properly the cells where contact is defined

◊ `ORIE_NORM_COQUE = _F(`

This keyword is for testing whether in a list of surface mesh cells (shells), the normals are mutually consistent. Otherwise, some cells are redirected.

◊ `ORIE_PEAU_2D =`

◊ `ORIE_PEAU_3D =`

These keywords are used to redirect the mesh edges so that their normals are consistent (pointing towards the outside of the material). This is a prerequisite if, for example, one wants to apply a pressure load on this "skin".

The definition and assignment of materials

A material is the definition of numerical parameters

- Units are not managed by **code_aster**: the user must use a consistent system of units throughout the study, that is between materials, unit length of the mesh, loading data, etc
- Example:

Coordinates of the mesh nodes	mm	m
Young's modulus	MPa	Pa
Applied force	N	N
Stress obtained as a result	MPa	Pa

The material properties must be assigned to the mesh

- Assignment to a geometrical area: groups of finite elements designated by the names of groups of cells (`GROUP_MA`)

The definition and assignment of materials

- **The material must be consistent with the assumptions of the resolution (constitutive equation)**

- **Example:**
  - **VMIS_ISOT_TRAC**: constitutive equation for von Mises elasto-plasticity with isotropic nonlinear hardening
  - One must have defined a stress-strain curve in the material, in addition to the elasticity parameters

```
steel=DEFI_MATERIAU(ELAS      =_F(E=2.1.E11, NU=0.3,),)
                    TRACTION  =_F( SIGM = CTRACB),)

mater=AFFE_MATERIAU(MAILLAGE=mesh,
                      AFFE=_F(TOUT='OUI', MATER=steel,),)

result=STAT_NON_LINE( ...
                    CHAM_MATER=mater,
                    COMPORTEMENT=_F(
                          RELATION = 'VMIS_ISOT_TRAC'),)
```

The definition and assignment of materials

- Be careful to assign what has been defined, then use what has been assigned:

```
STEEL1 = DEFI_MATERIAU( ELAS = _F( E = 205000.0E6,
                                   NU = 0.3, ), )
STEEL2 = DEFI_MATERIAU( ELAS = _F( E = 305000.0E6,
                                   NU = 0.3, ), )
STEEL3 = DEFI_MATERIAU( ELAS = _F( E = 405000.0E6,
                                   NU = 0.3, ), )
```

```
CHMAT1 = AFFE_MATERIAU(MAILLAGE=MESH,
                       AFFE =_F(TOUT='OUI',
                                MATER= STEEL2,),)
CHMAT2 = AFFE_MATERIAU(MAILLAGE=MESH,
                       AFFE =_F(TOUT='OUI',
                                MATER= STEEL3,),)
```

```
RESU   = MECA_STATIQUE(...
                       CHAM_MATER= CHMAT1,
                        ...)
```

One defines ***three*** different materials: **STEEL1** **STEEL2** **STEEL3**
But ***two*** of them are assigned: **CHMAT1** **CHMAT2**
Finally, only ***one*** material **CHMAT1** is used in the calculation!

The characteristics of structural elements

- **Shells, plates, beams, pipes, discrete elements …**
- **Geometrical information not provided by the mesh**
  - **Shells**: thickness ; reference frame in the tangent plane
  - **Beams**: cross-section characteristics ; orientation of the principal axes of inertia around the neutral axis ; curvature of the curved elements
  - **Discrete elements** (spring, mass, dashpot): values of the stiffness, mass or damping matrices
  - **Bars or cables**: area of the cross-section
  - **3D and 2D continuous media elements:** local reference axes defined with respect to the anisotropy directions

```
charac=AFFE_CARA_ELEM(MODELE=MODELE,
                      POUTRE=_F( GROUP_MA='rotor',
                                 SECTION='CERCLE',
                                 CARA='R',
                                 VALE=.05,),
```

Loadings

- Commands **AFFE_CHAR_*****

- Loadings inside the continuous media

```
| PESANTEUR
| ROTATION
| FORCE_INTERNE
| FORCE_NODALE
. . .
```

- Border loadings

```
| FORCE_FACE
| FORCE_ARETE
| FORCE_CONTOUR
| PRES_REP
. . .
```

- Loadings specific to structural elements

```
| FORCE_POUTRE
| FORCE_COQUE
| FORCE_TUYAU
. . .
```

Loadings

- **Commands `AFFE_CHAR_***`**

- **Imposed relationships on the degrees of freedom**

`DDL_IMPO`	Unknowns imposed on a node or group
`FACE_IMPO`	Unknowns imposed on the nodes of a cell or a group of cells
`LIAISON_SOLIDE`	Rigid body element on a set of nodes or cells
`LIAISON_ELEM`	3D-beam, beam-shell, 3D-pipe connections …
`LIAISON_COQUE`	connection between shells

•••

- **Be careful of the consistency with the degrees of freedom allowed by the chosen finite element**

```
char=AFFE_CHAR_MECA(MODELE=MO,
                    DDL_IMPO=_F( GROUP_NO = 'A',
                                 DX  = 0.,
                                 DY  = 0.,
                                 DZ  = 0.,
                                 DRX = 0.,
                                 DRY = 0.,
                                 DRZ = 0.),
```

Rotational displacements only for structural elements

The overloading rule for assignments

```
STEEL1 = DEFI_MATERIAU( ELAS = _F( E = 305000.0E6,
                                   NU = 0.3, ), )
STEEL2 = DEFI_MATERIAU( ELAS = _F( E = 405000.0E6,
                                   NU = 0.3, ), )
```

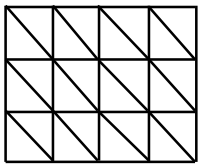
```
CHMAT1 = AFFE_MATERIAU(MAILLAGE=MESH,
                       AFFE =(_F(GROUP_MA=('GR1','GR2',),
                                 MATER=STEEL1,),
                              _F(GROUP_MA=('GR2',),
                                 MATER=STEEL2,),),)
```

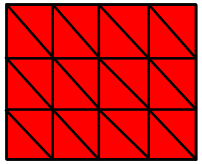
1

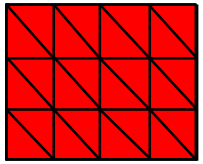
2

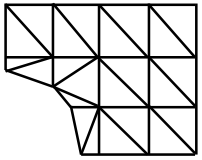
GR1

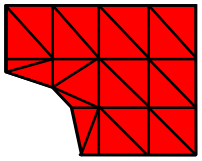
GR2

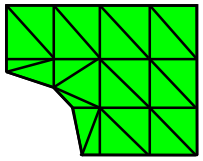












1

2

Solving the problem

The setting of solving operators

- **There are about fifteen resolution operators to solve the physical problems**
  - **Thermics:** `THER_LINEAIRE, THER_NON_LINE`
  - **Mechanics:** `MECA_STATIQUE, STAT_NON_LINE`
  - **Dynamics:** `DYNA_VIBRA, DYNA_NON_LINE`
  - **Modal calculation:** `CALC_MODES`

- **One must input the description of the problem prior to solving**
  - The model: `MODELE`
  - Materials: `CHAM_MATER`
  - Geometrical characteristics (for structural elements): `CARA_ELEM`
  - Loadings: `EXCIT`
  - A time stepping if necessary: `LIST_INST`

- **One can also change settings on the resolution algorithm → advanced usage**
  - In most cases, the default values are suitable

Choosing the linear solver

- **The linear solver for the system of equations can be chosen via the keyword `SOLVEUR / METHODE`**
- **Relevant depending on the problem to be solved**

- **Direct solvers**
  - `MULT_FRONT` (multi-frontal): Default method. Universal solver, very robust. Not recommended for mixed models of X-FEM, incompressible ...
  - `MUMPS` (MUltifrontal Massively Parallel sparse direct Solver): external solver. Slightly broader scope than `MULT_FRONT`. Provides access to parallelism.

- **Iterative solvers**
  - `GCPC` (Preconditioned conjugate gradient): recommended method for thermics. Useful for well conditioned, large problems.
  - `PETSC`: external multi-method solver. Very fast and robust when associated with pre-conditioner `LDLT_SP`. Provides access to parallelism.

Parallelism

- Use of parallelism
  - Choose `SOLVEUR=_F(METHODE='MUMPS')` or `SOLVEUR=_F(METHODE='PETSC')` in the command file
  - Choose a MPI version of **code_aster**
  - Choose the number of processors

- Up to 8 processors, the gains are virtually guaranteed for large enough problems (50000 unknowns)

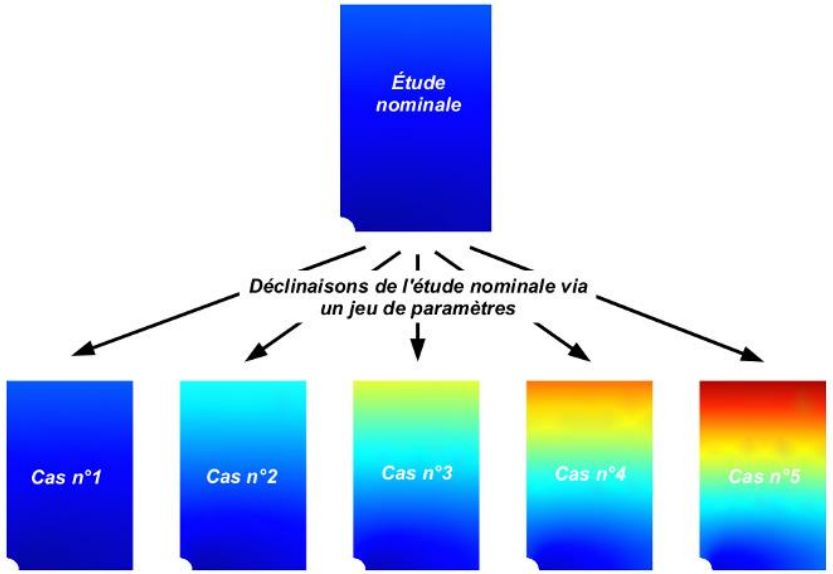
Parametric resolution

- Solve the same problem by varying a set of parameters (Young's modulus, thickness, etc. ...)
- Automatic run of distributed calculations only with ASTK GUI
  - Nominal command file + ranges of variation of a parameter (or more)
    - Integer, real, text
      - [U2.08.07] “Distribution of parametric calculations”

Building submitted to an earthquake

`300 calculations of 2h each → total execution time in 6h on 100 procs`

`Speed-up = 100`



Post-processing the results

The results in code_aster

- **Single field, single physical quantity: `cham_gd`**
  - **Type;**
  - **Several components;**
  - **A single access number (no time step for example).**

- **Result data structure: `resultat`**
  - **Gathering several fields of quantities in a given physics;**
  - **Several access numbers;**
  - **Parameters (depending on the model).**

The different types of fields

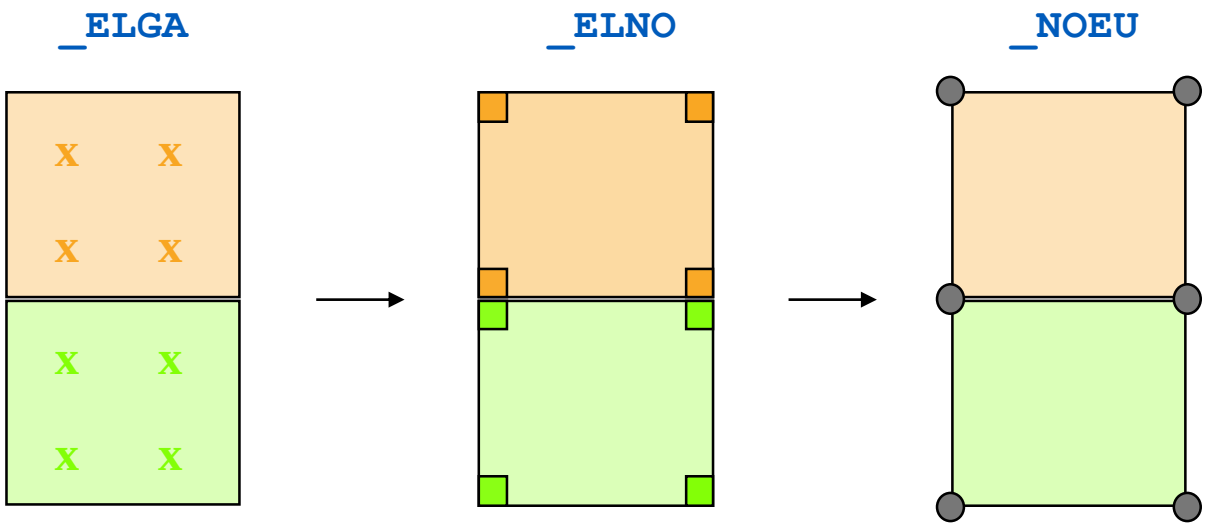
- **Location of the values:**
  - **Fields on nodes (`NOEU`) ;**
  - **On the elements:**
    - **Fields by element on Gauss points (`ELGA`);**
    - **Fields by element on nodes (`ELNO`);**
    - **Constant field on element (`ELEM`).**

- **Naming rule `XXXX_YYYY` :**
  - **Four first characters: name of the quantity (`SIGM, EPSI, ERRE, etc`);**
  - **Four last characters : location (`NOEU, ELGA, ELNO` or `ELEM`).**

- **Examples:**
  - **Exception! `DEPL, VITE, ACCE`**
  - **`SIEQ_ELNO, SIGM_ELNO`**

Location of the values

- **Three main locations**



CHAM_ELEM
given at Gauss points
per cell
SIGM_ELGA

CHAM_ELEM
given at nodes
per cell
SIGM_ELNO

CHAM_NO
given at nodes
SIGM_NOEU

The data structure `resultat`

- **Resolution operators produce typed data structures: `resultat`**
- **The type depends on the operator**
  - `EVOL_ELAS (linear mechanics), EVOL_NOLI (non-linear mechanics), EVOL_THER (linear thermics), MODE_MECA (modal analysis), ...`
- **At each computation step, one or more fields are stored in the data structure `resultat`**
- **Fields are identified by access variables**
  - `INST, NUME_ORDRE, FREQ, NUME_MODE, ...`
- **Examples of stored fields**
  - Temperatures for a list of time steps
  - Displacements for the first n modes
  - Displacements and stresses for a list of time steps

Calculating physical fields or performing mathematical operations: CALC_CHAMP / POST_ELEM

- **Produces fields and tables.**

- **CALC_CHAMP operator [U4.81.04]:**
  - **Creates or completes a result data structure;**
  - **Calculation of stresses, strains, energies, criteria, error indicators,…**

- **POST_ELEM operator [U4.81.22]:**
  - **Creates a table.**
  - **Calculation of energies (potential, elastic, kinetic, dissipated,…) ;**
  - **Calculation of integrals or average quantities.**

Handling fields, tables and data structures: CREA_CHAMP / CREA_RESU / CREA_TABLE / CALC_TABLE

- **Produces fields, results and tables.**

- **CREA_CHAMP operator [U4.72.04]:**
  - **Creates single fields (cham_gd),**

- **CREA_RESU operator [U4.44.12] :**
  - **Creates a result data structure from input fields.**

- **CREA_TABLE operator [U4.33.02]:**
  - **Creates a table from a function or a list of scalars.**

- **CALC_TABLE operator [U4.33.03]:**
  - **Manipulates data from tables like a spreadsheet.**

Extracting values :

- **MACR_LIGN_COUPE: Extract component values from fields on:**
  - Nodes;
  - Group of nodes;
  - Cut lines;
  - Example: components DX, DY, DZ from DEPL on node N125.

- **POST_RELEVE_T: Performs:**
  - Averaging;
  - Resultants and moments of vector fields;
  - Invariants of tensor fields;
  - Directional trace fields.

- **Create a table**

Outputing a result: IMPR_RESU

- **Outputs:**
  - **Meshes;**
  - **Fields;**
  - **Data contained in a result data structure.**

- **Different formats:**
  - **Listing** `FORMAT='RESULTAT'`
  - **Aster (for meshes only)** `FORMAT='ASTER'`
  - **MED** `FORMAT='MED'`
  - **GMSH** `FORMAT='GMSH'`
  - **IDEAS** `FORMAT='IDEAS'`

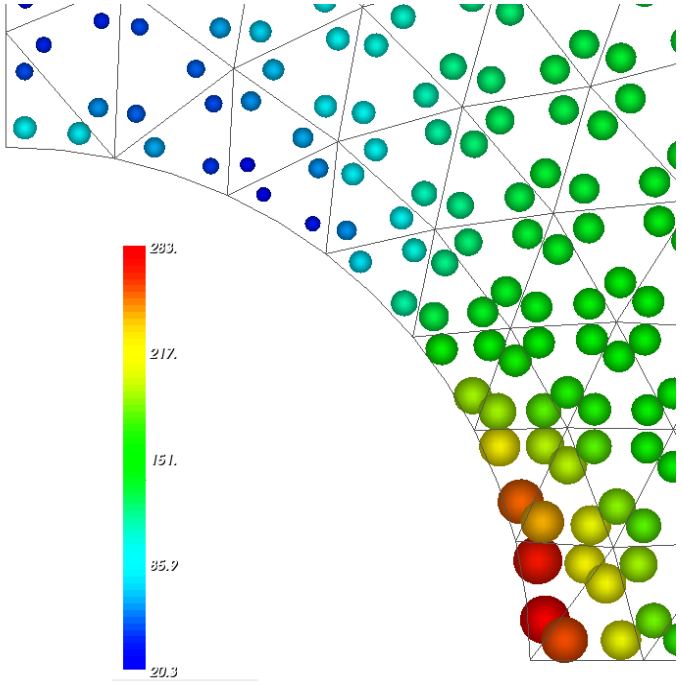
Outputing results: IMPR_RESU / FORMAT='RESULTAT'

- This procedure allows you to write either:
  - Fields on nodes (displacement, temperature, eigen modes, static modes, …);
  - Fields by elements on nodes;
  - Fields by elements on Gauss points (stress, generalized efforts, internal variables,…).
  - Parameters form a result data structure;
  - Min and max values;
  - Restriction between two bounds;
  - Etc.

Outputing results: IMPR_RESU / FORMAT='MED'

- **Neutral binary format developed by EDF for exchange between sofware:**
  - **Allows any other software interfaced with MED to read the results produced by code_aster with IMPR_RESU.**
  - **This is the default output format and the one to use for post-processing in Salome-Meca.**

- **This procedure allows you to write:**
  - **A mesh;**
  - **Fields on nodes ;**
  - **Fields by elements on nodes ;**
  - **Fields by elements on Gauss points;**
  - **… and restrict to a part of the model,**



Vizualisation on Gauss points

Outputing results: IMPR_RESU / FORMAT='GMSH'

- **This procedure allows you to write (*.msh file):**
  - **A mesh;**
  - **Fields on nodes;**
  - **Fields by elements on nodes.**

- **But we cannot write:**
  - **Fields by elements on Gauss points.**
  - **Use the MED format !**

- **Must specify the type of the quantity:**

```
TYPE_CHAM  = / 'SCALAIRE',    [DEFAUT]
             / 'VECT_3D',
             / 'TENS_2D',
             / 'VECT_2D',
             / 'TENS_3D',
NOM_CMP = lnomcmp,
```

Outputing results: IMPR_RESU / FORMAT='IDEAS'

- This procedure allows you to write (*.unv file):
  - A mesh;
  - Fields on nodes;
  - Fields by elements on nodes.

- Datasets used for the universal file:
  - dataset 55 for fields on nodes (displacement, temperatures,…)
  - dataset 57 for fields by elements on nodes (strain, stress,…)
  - dataset 56 for fields by elements on Gauss points (actually, constant filed on element are written by average on the Gauss points)

Outputing a table: IMPR_TABLE

- Print the contents of a table in a listing or an Excel file (document [U4.91.03]).
- Also plot curves !
- Selection :
  - NOM_PARA :
    - Choice of columns to print;
  - FILTRE :
    - Choice of lines to print (some lines verifying a criterion);
  - TRI :
    - Choice of the order of rows to print (ascending or descending);
  - FORMAT :
    - Choice of output format.

Building a function: RECU_FONCTION

- **Extract the evolution of a quantity as a function of another ([U4.32.04]);**
  - From a result data structure, the generated function corresponds to the temporal evolution of a component of a node or Gauss point of the model.

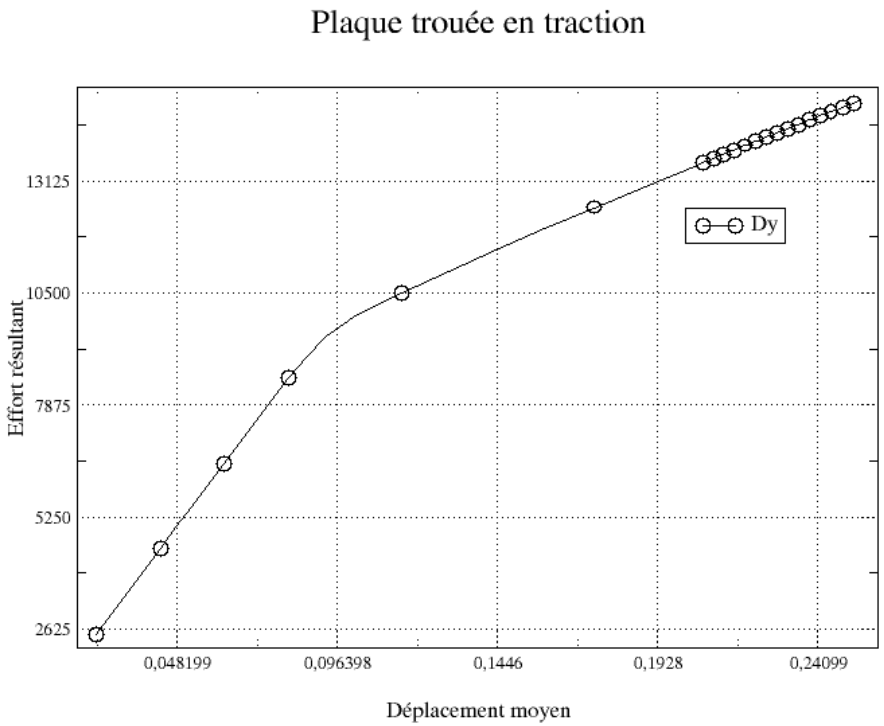
```
% COURBE SIGMA YY AU POINT G EN FONCTION DU TEMPS
   SYYG = RECU_FONCTION ( RESULTAT = RESUNL,
                          NOM_CMP = 'SIYY',
                          NOM_CHAM = 'SIEF_NOEU',
                          GROUP_NO = G,
                          TOUT_ORDRE = 'OUI') ;
```

  - From a table data structure, this operator can extract the evolution of two parameters.

```
% COURBE SIGMA XX A T DONNE EN FONCTION DE L'ABSCISSE
   fct = RECU_FONCTION  ( TABLE = tab,
                          PARA_X = 'ABSC_CURV',
                          PARA_Y = 'SIXX') ;
```

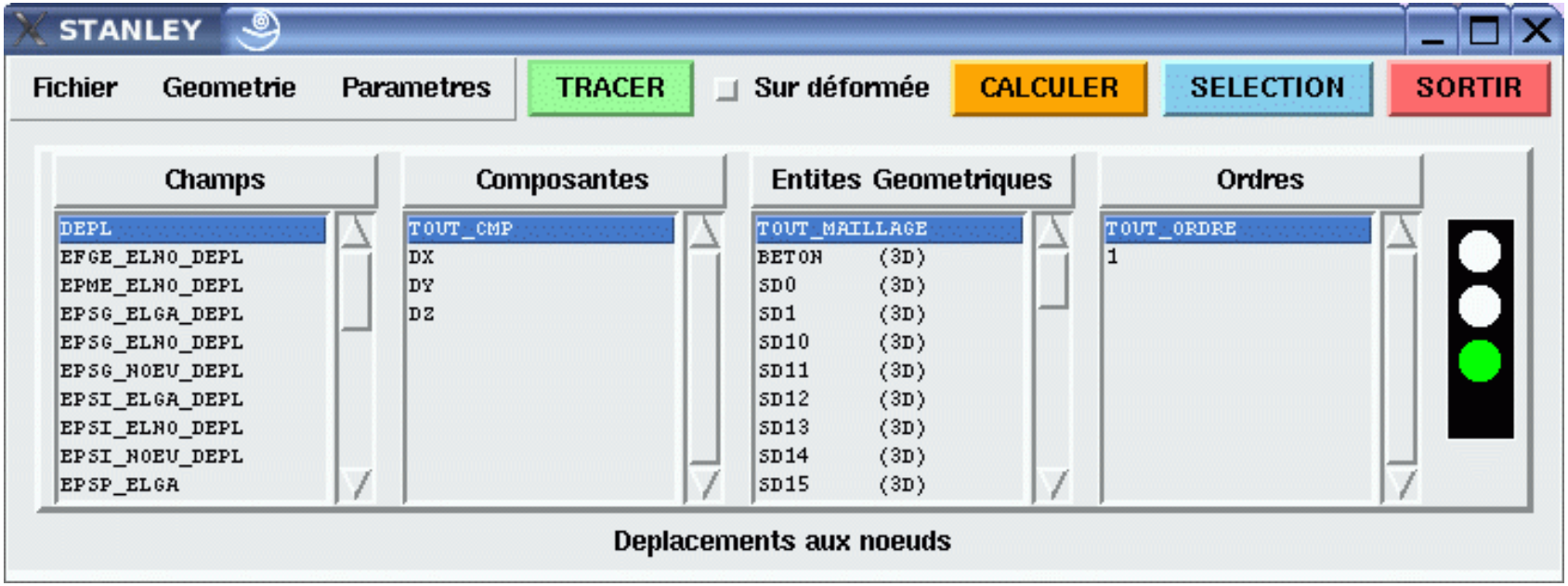
Outputing a function: IMPR_FONCTION

- **Outputing a function from a code_aster function data structure:**
  - **Formats `'TABLEAU'` or `'XMGRACE'`,**
  - **[U4.33.01];**
  - **Example:**



STANLEY : interactive post-processing tool

- Interactive post-processing operations;
- Visualization in Salome-Meca or GMSH;
- Fields or plots;
- Call with a single command: `STANLEY()`



End of presentation

Is something missing or unclear in this document?

Or feeling happy to have read such a clear tutorial?

Please, we welcome any feedbacks about Code_Aster training materials.

Do not hesitate to share with us your comments on the Code_Aster forum dedicated thread.