A simple mechanical study

code_aster, salome_meca course material
GNU FDL licence (http://www.gnu.org/copyleft/fdl.html)
code_aster step by step: outline

- Introducing 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 functions and time stepping
  - The boundary conditions and loadings
- Solving the problem
  - The setting of solving operators
  - The linear solvers, parallel solvers and distributed computations
- Post-processing the results
  - Result types
  - Calculation of options and operations on results
  - Extraction of values and outputs
A simple example

R=1m

L=4m
A simple example: Test case forma00a [U1.05.00]

```plaintext
# 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()
```

Axis-symmetry
Starting the study and acquiring the mesh
Special commands: **DEBUT, FIN, POURSUITE**

**The DEBUT command**
- Begins execution, previous lines are ignored
- Basic usage:
  ```plaintext
  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:
  ```plaintext
  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
Reading the mesh

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
  \texttt{mymesh = LIRE_MAILLAGE(FORMAT='ASTER')} 
- MED format (default input format)
  \texttt{mymesh = LIRE_MAILLAGE()} 

Meshes in other formats
- Commands \texttt{PRE_***} to convert to Aster format: \texttt{PRE_GIBI, PRE_IDEAS, PRE_GMSH} 
- Example:
  \begin{verbatim}
  PRE_***()
  mymesh = LIRE_MAILLAGE(FORMAT='ASTER')
  \end{verbatim}

Outputting the mesh (if modified for example)
- Command \texttt{IMPR_RESU} 
- For instance in MED format (default output format):
  \begin{verbatim}
  IMPR_RESU( RESU=_F(MAILLAGE=mymesh,),)
  \end{verbatim}
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".
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

```python
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:

\[
\begin{align*}
\text{PHENOMENE} & = \text{'MECANIQUE'} \\
\text{MODELISATION} & = \text{'}D\_PLAN' \\
& \text{'}C\_PLAN' \\
& \text{'}AXIS' \\
& \text{'}DKT' \\
& \text{'}DST' \\
& \text{'}COQUE\_3D' \\
& \text{'}POU\_D\_T' \\
& \text{'}TUYAU'
\end{align*}
\]

- 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

One will have to write kinematic connections between degrees of freedom

- See following chapter for loadings and boundary conditions
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:

<table>
<thead>
<tr>
<th>Coordinates of the mesh nodes</th>
<th>mm</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>Young's modulus</td>
<td>MPa</td>
<td>Pa</td>
</tr>
<tr>
<td>Applied force</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Stress obtained as a result</td>
<td>MPa</td>
<td>Pa</td>
</tr>
</tbody>
</table>

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

```
steelm=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:

\[
\begin{align*}
\text{STEEL1} &= \text{DEFI\_MATERIAU}(\text{ELAS} = _F( \text{E} = 205000.0\text{E}6, \\
                    \text{NU} = 0.3, ), ) \\
\text{STEEL2} &= \text{DEFI\_MATERIAU}(\text{ELAS} = _F( \text{E} = 305000.0\text{E}6, \\
                    \text{NU} = 0.3, ), ) \\
\text{STEEL3} &= \text{DEFI\_MATERIAU}(\text{ELAS} = _F( \text{E} = 405000.0\text{E}6, \\
                    \text{NU} = 0.3, ), ) \\
\text{CHMAT1} &= \text{AFFE\_MATERIAU}(\text{MAILLAGE=MESH,} \\
                    \text{AFFE} = _F(\text{TOUT='OUI',} \\
                    \text{MATER=} \text{STEEL2}, ), ) \\
\text{CHMAT2} &= \text{AFFE\_MATERIAU}(\text{MAILLAGE=MESH,} \\
                    \text{AFFE} = _F(\text{TOUT='OUI',} \\
                    \text{MATER=} \text{STEEL3}, ), ) \\
\text{RESU} &= \text{MECA\_STATIQUE}(\ldots \\
                    \text{CHAM\_MATER=} \text{CHMAT1}, \\
                    \ldots )
\end{align*}
\]

One defines \textbf{three} different materials: \texttt{STEEL1} \texttt{STEEL2} \texttt{STEEL3} 
But \textbf{two} of them are assigned: \texttt{CHMAT1} \texttt{CHMAT2} 
Finally, only \textbf{one} material \texttt{CHMAT1} can be used in the calculation!
The overloading rule for assignments

```plaintext
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,)),)
```

GR1

GR2
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

```plaintext
charac=AFFE_CARA_ELEM(MODELE=MODELE,
   POUTRE=_F( GROUP_MA='rotor',
      SECTION='CERCLE',
      CARA='R',
      VALE=.05),
```
The functions and time stepping

Functions

- Define a real or complex function of a real variable
- Usually for the loading or boundary conditions, which depends on a variable of space or time
- For example

```plaintext
FUNC = DEFI_FONCTION( NOM_PARA = 'Y',
                       VALE = ( 0.0, 200000.0, 4.0, 0.0 ) )
```

Time stepping

- Time list for the calculation
- Time management for the iterative algorithms

```plaintext
listr = DEFI_LIST_REEL( DEBUT = 0.0,
                        INTERVALLE = _F( JUSQU_A = 10.0,
                                        NOMBRE = 10 ) )

time = DEFI_LIST_INST( DEFI_LIST = _F( LIST_INST = listr, ) )
```
Loadings

- Commands `AFFE_CHAR_**** (_F)`

- Loadings inside the continuous media
  - PESANTEUR
  - ROTATION
  - FORCE_INTERNE
  - FORCE_NODALE
  - . . .

- Border loadings
  - FORCE_FACE
  - FORCE_ARTE
  - FORCE_CONTOUR
  - PRES_REP
  - . . .

- Loadings specific to structural elements
  - FORCE_POUTRE
  - FORCE_COQUE
  - FORCE_TUYAU
  - . . .
Loadings

**Commands** `AFFE_CHAR_****(_F)`

**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**

```plaintext
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
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)
Post-processing the results
The results in code_aster

- Single field, single physical quantity: \texttt{cham\_gd}
  - Type;
  - Several components;
  - A single access number (no time step for example).

- Result data structure: \texttt{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**
The different types of fields

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
Operations to produce fields, tables, results:

- **Fields:** `CALC_CHAMP` / `CREA_CHAMP`

- **Tables:** `POST_ELEM` / `CREA_TABLE` / `CALC_TABLE` / `POST_RELEVE_T` / `MACR_LIGN_COUPE`

- **Results:** `CREA_RESU`

- **Functions:** `RECU_FONCTION`
Operations to output files:

- **IMPR_RESU**
  - Meshes / Fields / Data contained in a result data structure.
  - Different formats: RESULTAT / ASTER/ MED / GMSH/ IDEAS
    *(Post-process in AsterStudy)*

- **IMPR_TABLE**
  - Different formats: Print the contents of a table in a listing or an Excel file, also plot curves

- **IMPR_FONCTION**
  - Extract the evolution of a quantity as a function of another
  - Formats ‘TABLEAU’ or ‘XMGRACE’,
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.