Tips for optimizing your computations with Code_Aster

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

- Where does the code spend time?
- Some tips to optimize a (sequential) study
- Parallel version of Code_Aster
- Parametric studies
Where does Code_Aster spend time?
A simple simulation with Code_Aster

- Read mesh
- Define the model
- Define boundary conditions
- Define material properties

- Compute elementary matrices and build the problem matrix $K$
- Solve $KU = F$

- Compute fields
- Write results

Nonlinear solver
Time loop
Dominating steps: Assembly/Solve

- Elementary matrices computations to build $K$
  - may take time if the material constitutive equations are complex

- Solving the linear system $KU = F$
  - Usually (and always for linear problems) the most expensive step

- These steps should be efficient since they are called by:
  - the iterative loop of the nonlinear solver
  - the time loop
Tips for the study definition
Definition of the problem

- Optimize the mesh
  - to keep a reasonable number of unknowns
- Prefer `AFFE_CHAR_CINE` (to `AFFE_CHAR_MECA`) to define boundary conditions:
  - `AFFE_CHAR_MECA` uses dualization:
    - Pros:
      - general and flexible approach
    - Cons:
      - it increases the number of unknowns and results to a saddle-point matrix.
  - `AFFE_CHAR_CINE` uses elimination:
    - Pros:
      - no additional unknowns
      - better numerical properties for the resulting matrix: allows to use more efficient (iterative) solvers.
    - Cons:
      - less general: it allows local boundary conditions (\(u=u_0\)) but no global linear relations (ex `AFFE_CHAR_MECA/LIAISON_SOLIDE`)
Choose an efficient linear solver

- **Direct solvers** are classically preferred in computational mechanics.
- Based on a LU factorization of K followed by triangular solves.
- Robust method (though costly for large systems).
- In Code_Aster
  - MUMPS external sparse direct solver is the default option.
  - MUMPS means MUltifrontal Massively Parallel sparse direct Solver, developed by CERFACS, ENS Lyon, INPT(ENSEEIHT)-IRIT, Inria, Mumps Technologies and University of Bordeaux.
  - EDF is a member of MUMPS consortium.

```plaintext
RES = STAT_NON_LINE( ... 
    SOLVEUR=_F(METHODE='MUMPS',) 
    ...) 
```
Choose an efficient linear solver

- **Iterative solvers** may be very efficient
- They are faster than direct solvers but sometimes fail to converge.
- A good preconditioner is mandatory
  - Matrices may have a very poor conditionning number when structural elements (beams, ...) are present

- In code_aster
  - external iterative solver (ONLY parallel ...)

```
RES = STAT_NON_LINE(...
    SOLVEUR=_F(METHODE='PETSC',)
  ...
)
```

```
RES = STAT_NON_LINE(...
    SOLVEUR=_F(METHODE='PETSC',
                PRE_COND='BOOMER',)
    ...)
```

9 - Code_Aster and Salome-Meca course material  GNU FDL Licence
Use Code_Aster parallel version (_mpi)
Is it worth using parallelism?

- Find out where Code_Aster spends time
  - In elementary computations?
  - In the linear solver process? Is the number of degrees of freedom large?

**USE AN MPI VERSION!**

- Need a lot of time steps to model your problem?
- Very non-linear problems (slow Newton convergence)? Post-processing? …..

**YOU MAY USE AN MPI VERSION**
**BUT GAIN IS NOT STRAIGHTFORWARD**
HPC resources – EOLE cluster

- **eochn[1-1164]**
  - 1164 standard Computing Nodes
  - 1 node=2CPUx14nodes/CPU
  - 128 Go RAM/node

- **eobm[1-0096]**
  - 96 Big Memory Nodes
  - 1 node=2CPUx14nodes/CPU
  - 256 Go RAM/node

- **eobm[0097-0144]**
  - 48 Big Memory Nodes
  - 1 node=2CPUx14nodes/CPU
  - 512 Go RAM/node

- **eobm[0145-0160]**
  - 16 Big Memory Nodes
  - 1 node=2CPUx4nodes/CPU
  - 256 Go RAM/node

- **eobm[0161-0162]**
  - 2 Big Memory Nodes
  - 1 node=4CPUx16nodes/CPU
  - 2 To RAM/node

- **eofront[1-5]**
  - 5 login nodes

**/scratch/NNI 1To**
HPC resources – GAIA cluster (R&D only)

- **qacn[1-1112]**
  - 1112 standard Computing Nodes
  - 1 node=2CPUx18nodes/CPU
  - 192 Go RAM/node

- **qabm[1-0048]**
  - 48 Big Memory Nodes
  - 1 node=2CPUx14nodes/CPU
  - 384 Go RAM/node

- **qahm[1-0032]**
  - 48 Big Memory Nodes
  - 1 node=2CPUx4nodes/CPU
  - 384 Go RAM/node

- **gafront[1-5]**
  - 5 login nodes

/scratch/NNI  2To

Attached storage

Users' computers

Fast Interconnect

Login nodes

Computing nodes
How to use a parallel version of code_aster?

In History View, go to Run Parameters panel, then Basic tab.

Choose a cluster (here EOLE)

Select a parallel version (_mpi) of code_aster

Choose the total number of MPI processes $n_{cpu_{total}}$

Choose the amount of memory PER CPU PROCESS $mem_{per\ CPU}$

Set the ELAPSED TIME needed for the computation
How to use a parallel version of code_aster?

In **History View**, go to **Run Parameters** panel, then **Advanced** tab.

- Run the computation in **Batch** mode.
- Choose $n_{nodes}$ the number of computing nodes.

Optimized version to run fast!

Leave the number of threads equal to 0: the optimal value will be set for you.

$$\frac{ncpu_{total}}{n_{nodes}} = ncpu_{per\ node}$$

$$ncpu_{per\ node} \times mem_{per\ CPU} < mem_{of\ node}$$
Parametric studies
What for?

- A parametric study is a series of simulations where one or more parameters of the problem are varied.

- The purpose is to investigate the impact of a variation of the parameter(s)
  - a geometrical size, physical material property, mechanical load ….

- AsterStudy npw has a direct interface to OpenTURNS which allows to manage parametric studies.
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.