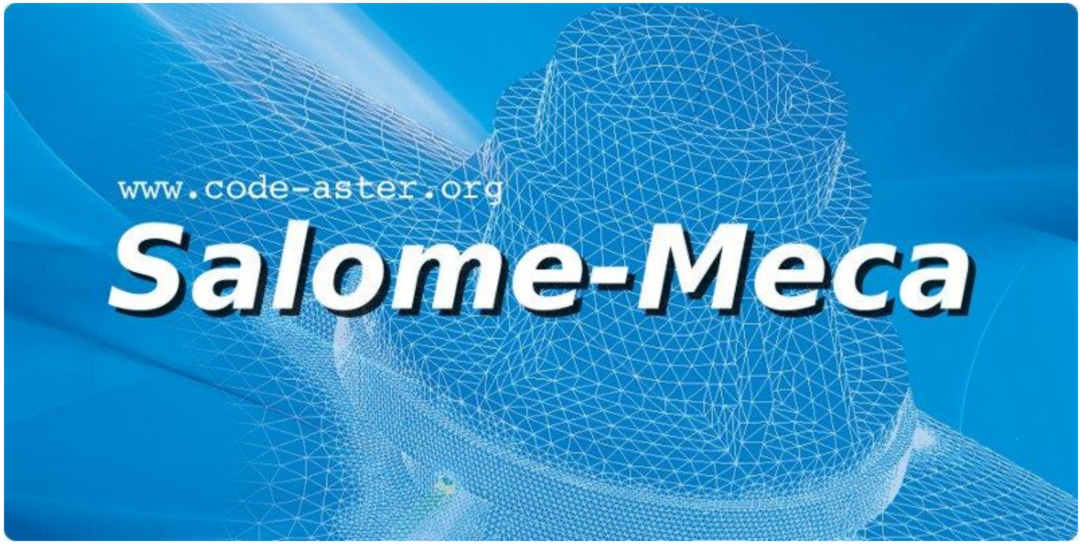
Linear transient & harmonic analysis



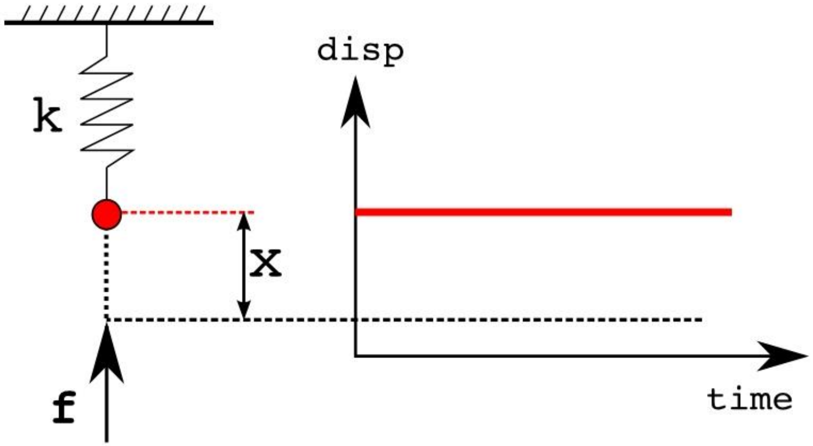
Code_Aster, Salome-Meca course material

Outline

- What is dynamics about ? Inertia !
- Transient analysis
- Harmonic analysis
- Eigen vectors and model reduction
- Syntax examples
- Some advice
- Bibliography

What is dynamics about: an illustrative example

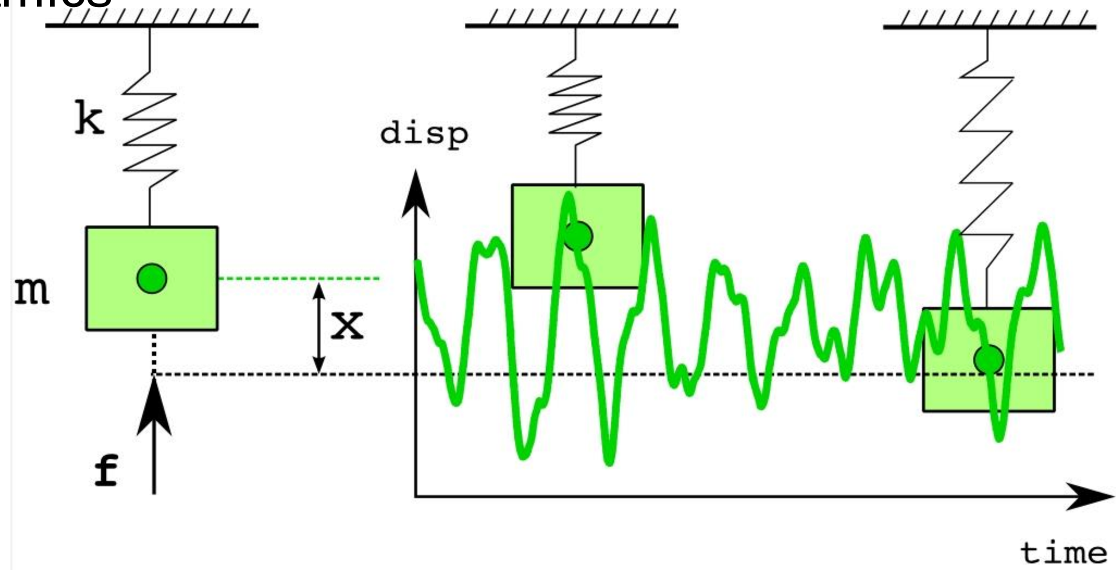
- Statics



We seek the system's stationary position

$$k\ x = f$$

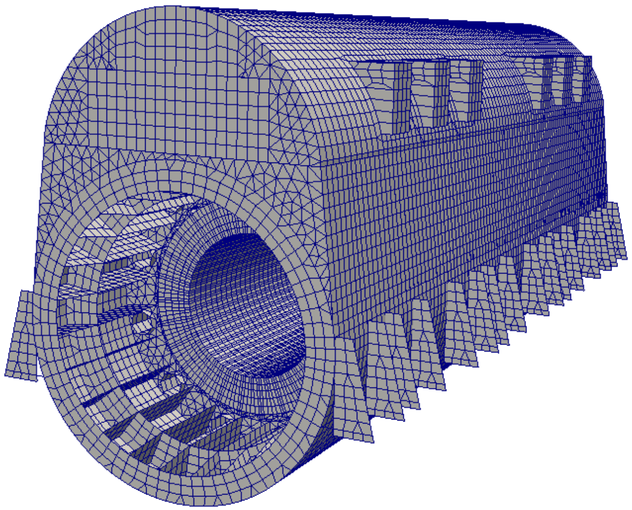
- Dynamics



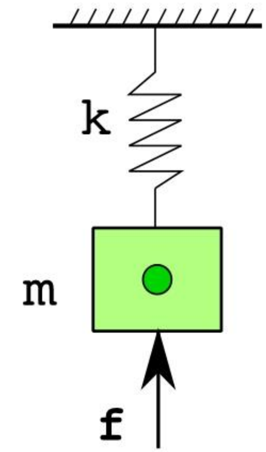
We seek the system's time history

$$m\ddot{x} + kx = f(t)$$

From continuous to discrete : Finite Elements



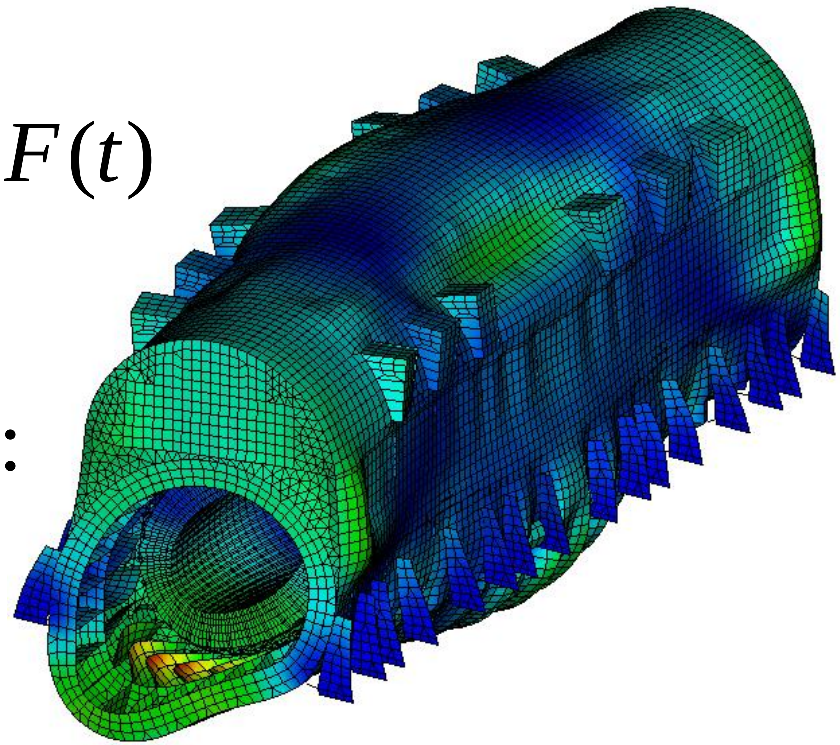
- Discretization → Matrix equation



$$m\ddot{x} + kx = f(t)$$

$$M\ \ddot{X} + K\ X = F(t)$$

$$X(t_i):$$

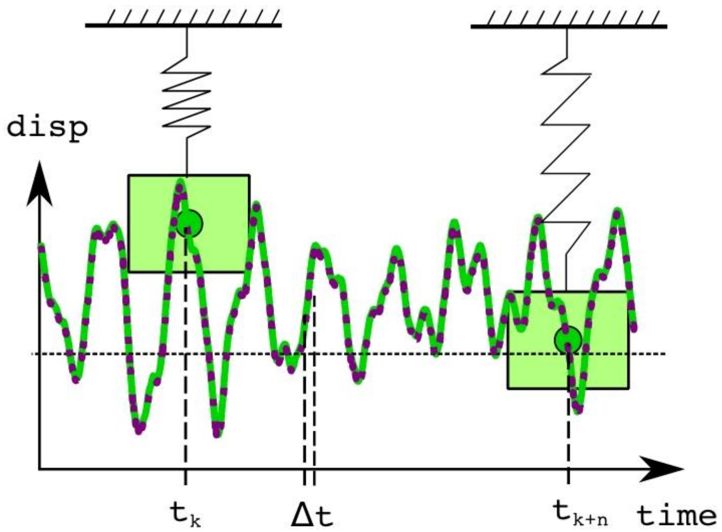


Transient analysis: principles

- Equation of motion
  - Separate time and space variables

$$M\ \ddot{X} + C\ \dot{X} + K\ X = F(t) = E\ u(t)$$

- We seek <u>time history</u> $\ddot{X}(t)\,;\,\dot{X}(t)\,;\,X(t)$



- Numerical time integration :
  - Force & Inertia balance → $\ddot{X}(t_k)$
  - Example : central differencing scheme →

$$\begin{cases} \dot{X}(t_{k+\frac{1}{2}}) = \dot{X}(t_{k-\frac{1}{2}}) + \Delta t\ \ddot{X}(t_k) \\ X(t_{k+1}) = X(t_k) + \Delta t\ \dot{X}(t_{k+\frac{1}{2}}) \\ \dot{X}(t_{k+1}) = \dot{X}(t_{k+\frac{1}{2}}) + \dfrac{\Delta t}{2}\ \ddot{X}(t_{k+1}) \end{cases}$$

Transient analysis: implementation

- We need
  - Structural matrices M (mass) , C (damping), K (stiffness)
  - External loading F_0 and its time evolution u(t)

- Requirements
  - The model → **AFFE_MODELE**
  - Materials → **AFFE_MATERIAU**
  - Boundary conditions → **AFFE_CHAR_MECA**
  - Characteristics of structural elements (if needed) → **AFFE_CARA_ELEM**

- Model assembly
  - Matrices M, C , K ; Loading F → **ASSEMBLAGE**
  - **Time evolution → FORMULE / DEFI_FONCTION**

- Solve using **DYNA_VIBRA(TYPE_CALCUL='TRAN',BASE_CALCUL='PHYS')**

- Post-processing (same tools as in statics)
  - **CALC_CHAMP, POST_CHAMP** …
  - Output : **IMPR_RESU**

Harmonic Analysis: principles

- We seek the <u>steady-state</u> response
- Transient analysis for steady-state oscillatory excitation & response

$$M\ \ddot{X} + C\ \dot{X} + K\ X = F_0\ u(t)$$

- Alternative approach for steady-state motion

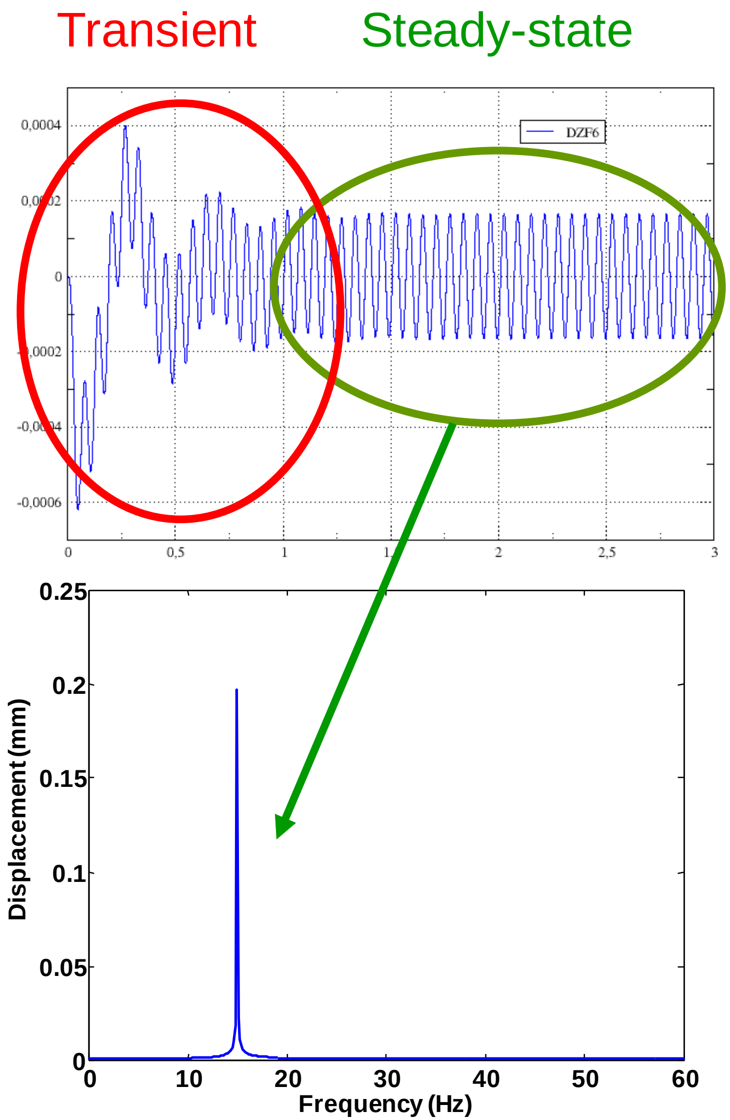
$$u(t) = u_0 e^{j\omega t} \Rightarrow\ X(t) = X(\omega) e^{j\omega t}$$

$\Downarrow$ (Fourier transform)

$$[-\omega^2 M + j\omega\, C + K]\ X(\omega)\ e^{j\omega t} = F_0\ u_0\ e^{j\omega t}$$

- Frequency-by-frequency computing

Responses to 2 different frequencies are completely independent

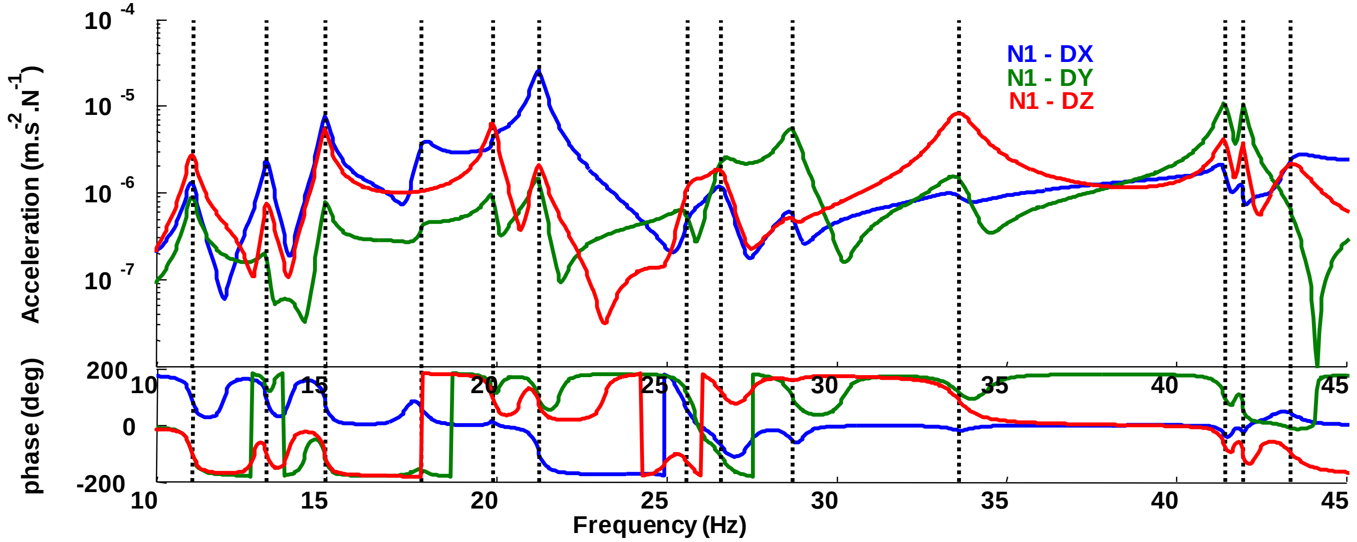


Harmonic analysis: implementation

- We need
  - Structural matrices M (mass) , C (damping), K (stiffness)
  - External loading F_0 and its frequency evolution u(ω)

- Same requirements and model assembly as transient analysis

- Resolution with **`DYNA_VIBRA(TYPE_CALCUL='HARM',BASE_CALCUL='PHYS')`**



- Post-processing (same tools as in statics)
  - **`CALC_CHAMP, POST_CHAMP`** …
  - Output : **`IMPR_RESU`**

Why are normal modes useful ?

- Modal coordinates give a “natural” description of the motion

$$X(t) = \phi_1 \eta_1(t) + \cdots + \phi_N \eta_N(t)$$

- Modal projection reduces the analysis cost

the number of unknowns is now equal to the number of modes (p) !

$$X(t) \approx \phi_1 \eta_1(t) + \cdots + \phi_p \eta_p(t) \,;\; p << N$$

- Simple *rule-of-thumb*: use eigen frequencies up to 2 x maximal loading frequency
- Warning!!! Always check the validity of the modal basis
  increase the number of modes, apply static corrections

Modal projection : implementation

- Same requirements and model assembly as transient or harmonic analysis
- Compute normal modes → **CALC_MODES**
- Reduced (projected) model and load assembly → **PROJ_BASE**
- Integration of the differential (dynamic) equations of motion :
  - transient → **DYNA_VIBRA(TYPE_CALCUL='TRAN',BASE_CALCUL='GENE')**
  - harmonic → **DYNA_VIBRA(TYPE_CALCUL='HARM',BASE_CALCUL='GENE')**
- Back to physical coordinates :
  - whole model → **REST_GENE_PHYS**
  - few points → **POST_GENE_PHYS(RESU_GENE=…) => table**

Syntax example : matrix assembly (K,M,…)

Example : 3D model

```
DEBUT()
#-------- model description -------------------------
ma    = LIRE_MAILLAGE  ( )
mo    = AFFE_MODELE    ( MAILLAGE= ma,
                         AFFE = _F(TOUT = 'OUI', PHENOMENE='MECANIQUE',
                         MODELISATION='3D'))
steel = DEFI_MATERIAU  ( ELAS = _F( E = 2.1E+11,  NU = 0.3, RHO = 7800.)
cmat  = AFFE_MATERIAU  ( MAILLAGE=ma, AFFE=_F(TOUT = 'OUI', MATER=steel ))

#-------- boundary conditions -------------------------
block = AFFE_CHAR_MECA( MODELE=mo, DDL_IMPO=_F(GROUP_MA='BOUND',LIAISON='ENCASTRE')

#-------- matrix assembly -----------------------------
ASSEMBLAGE(  MODELE    = mo, CHARGE= block, CHAM_MATER= cmat,
             NUME_DDL  = CO('nddl'),
             MATR_ASSE= _F(
                    ( MATRICE= CO('matrigi'), OPTION= 'RIGI_MECA' ),
                    ( MATRICE= CO('matmass'), OPTION= 'MASS_MECA' )))
```

- *N.B. : **nddl** is a numbering concept which insures consistency between matrixes and vectors*

Syntax example : normal modes computation

- **Computation with `CALC_MODES`**
  - **10 first frequencies**

```
modes = CALC_MODES ( MATR_A= matrigi, MATR_B= matmass,
                     OPTION='PLUS_PETITE',
                     CALC_FREQ=_F(NMAX_FREQ= 10) )
```

  - **Frequencies between f1=0.0 Hz and f2=100.0 Hz**

```
modes = MODE_ITER_SIMULT (   MATR_A= matrigi, MATR_B= matmass,
                             OPTION='BANDE',
                             CALC_FREQ=_F(FREQ= (0.,100.) )
```

- Printing to SALOME visual interface ('.med' format)

```
IMPR_RESU(FORMAT='MED',UNITE=80, RESU=_F(RESULTAT=modes,))
```

- Printing frequencies in the .resu file

```
IMPR_RESU( RESU=_F(RESULTAT=modes, TOUT_CHAM='NON', NOM_PARA=('FREQ',)))
```

Syntax example: direct time-history analysis

How to use the command

- External loading
  - `FXELEM = AFFE_CHAR_MECA(MODELE=MODELE, FORCE_NODALE=_F(GROUP_NO='BOUT', FX=1.0))`
- Assembly

```
ASSEMBLAGE(MODELE= mo, CHARGE= block, CHAM_MATER= cmat,
           NUME_DDL=CO('nddl'),
           MATR_ASSE= (   _F( MATRICE= CO('matrigi') , OPTION= 'RIGI_MECA' ),
                          _F( MATRICE= CO('matmass'), OPTION= 'MASS_MECA' ),),
           VECT_ASSE=  _F( VECTEUR= CO('matrigi'), CHARGE=FXELEM, OPTION= 'CHAR_MECA'),)
```

- Function of time
  - **Either `FORMULE` : mathematical expression of time**
    - **NB : time in *Code_Aster* is always noted `'INST'`**
  - Or `DEFI_FONCTION` : tabulated magnitude
  - `impuls=DEFI_FONCTION(NOM_PARA='INST',PROL_DROITE='CONSTANT',PROL_GAUCHE='CONSTANT',`
    `VALE=(.0,.0, 0.9,.0, 1.0,g, 2.0,g, 2.1,.0,))`
- List of time steps
  - `LINST=DEFI_LIST_REEL(DEBUT=0., INTERVALLE=_F(JUSQU_A=tfin, PAS=pa))`
  - `CALC_FONC_INTERP` : tabulation on the time steps to optimize the computing time
    `rimpuls=CALC_FONC_INTERP(FONCTION=IMPULS, LIST_PARA=LINST,)`
- Transient analysis

```
DLT   = DYNA_VIBRA (TYPE_CALCUL='TRAN', BASE_CALCUL='PHYS',
        SCHEMA_TEMPS=_F(SCHEMA='NEWMARK'),
        MATR_MASS=matmass, MATR_RIGI=matrigi,
        EXCIT=_F(VECT_ASSE=fx, FONC_MULT=rimpuls,),
        INCREMENT=_F(LIST_INST=LINST))
```

– How to chose the time step :
  - Frequency content of the system
  - Frequency content of the input

Syntax example: modal transient analysis

- Projection

```
PROJ_BASE(BASE=modes,
          MATR_ASSE_GENE=(  _F(MATRICE=CO('maspro'), MATR_ASSE=matmass,),
                            _F(MATRICE=CO('ripro'), MATR_ASSE=matrigi,),
          VECT_ASSE_GENE=(  _F(VECTEUR=CO('fxpro'), VECT_ASSE=fx)))
```

- Transient Analysis

```
DTM = DYNA_VIBRA(TYPE_CALCUL='TRAN', BASE_CALCUL='GENE',
          SCHEMA_TEMPS=_F(SCHEMA='NEWMARK'),
          MATR_MASS=maspro, MATR_RIGI=ripro,
          INCREMENT=_F(INST_FIN=tfin, PAS=pa,),
          EXCIT=_F(VECT_ASSE_GENE=fxpro, FONC_MULT=rimpuls))
```

- The route backwards to physical coordinates
  - Natural way :

```
REPHYS=REST_GENE_PHYS(RESU_GENE=DTM, NOM_CHAM=('ACCE','DEPL'))
```

    - May be costly !

  - More efficiently for observing the trajectories of some nodes or elements

```
DXOBS=POST_GENE_PHYS(RESU_GENE=DTM,
          OBSERVATION=_F(NOM_CHAM='DEPL', NOM_CMP='DX',GROUP_NO='OBS'))
```

=> table

Some advice

- EFICAS can help
  - Right syntax (but no guaranty on the rightness of the model !)
  - Translation from one version to another (changes in syntax)

- Read U2 & U4 documents
  - (and to go further : R for References)

- Validation tests are (often) good examples

- A modal analysis is always the starting point
  - Eigenfrequencies
  - Check of the FE model
  - Indication in the choice of time step

A brief bibliography

- *http://www.code-aster.org*

- *Mechanical Vibrations - Theory and Application to Structural Dynamics*

M. Géradin, D. Rixen - Wiley

- *Vibration Problems in Engineering*

S. Timoshenko - Wiley

- *Finite Element Analysis with Error Estimators*

J.E. Akin – Elsevier

- *Dynamics of structure*

R.W. Clough, J. Penzien – McGraw-Hill

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.