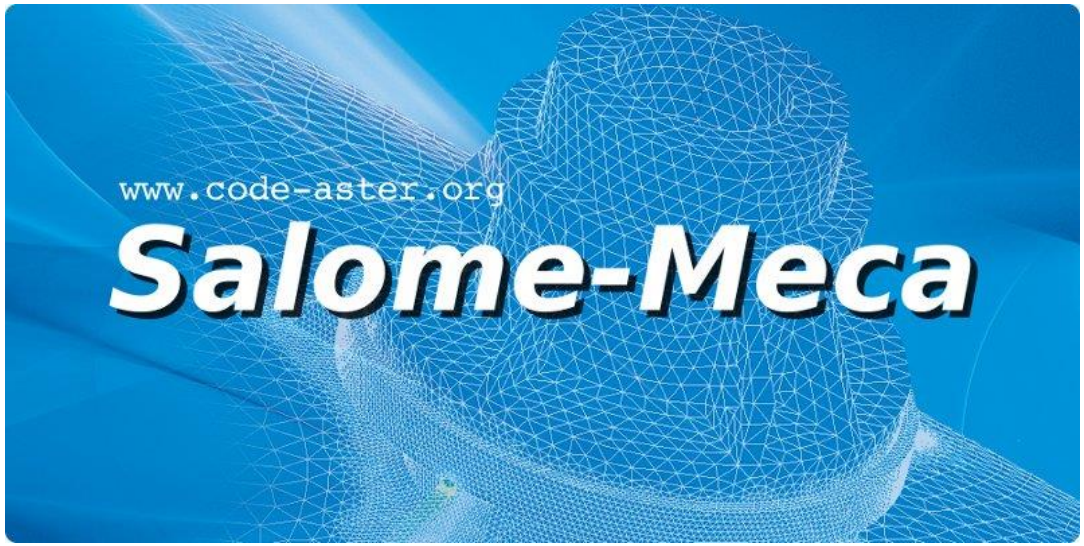
Linear dynamics



Code_Aster, Salome-Meca course material

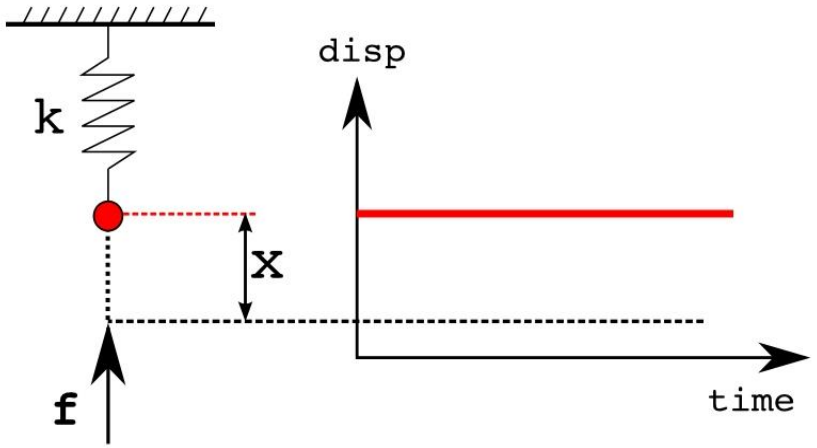
EDF

Outline

- What is dynamics about ? Inertia !
- Transient analysis
- Harmonic analysis
- Eigen vectors: what are they about ?
- Model reduction in dynamics
- Tips
- Wider scopes
- Bibliography

What is dynamics about: an illustrative example

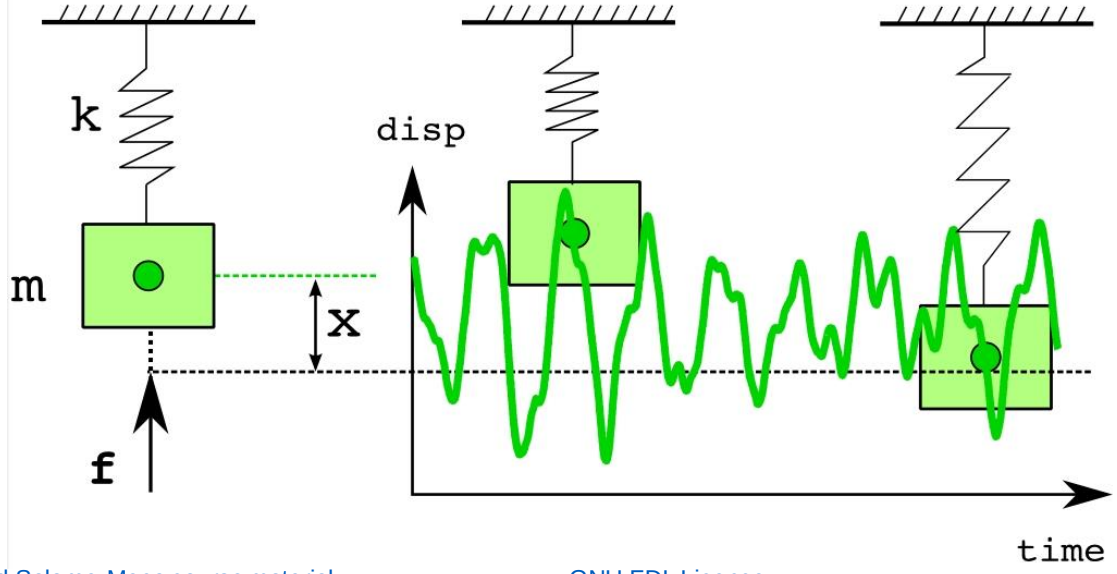
- Statics



We seek the system's stationary position

$$k\,x = f$$

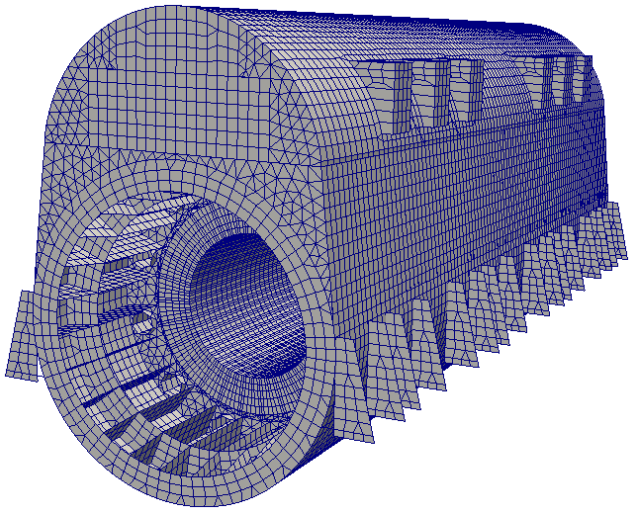
- Dynamics



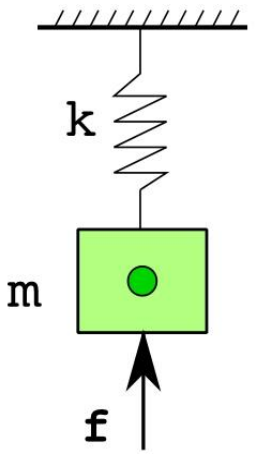
We seek the system's time history

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

From continuous to discrete : Finite Elements



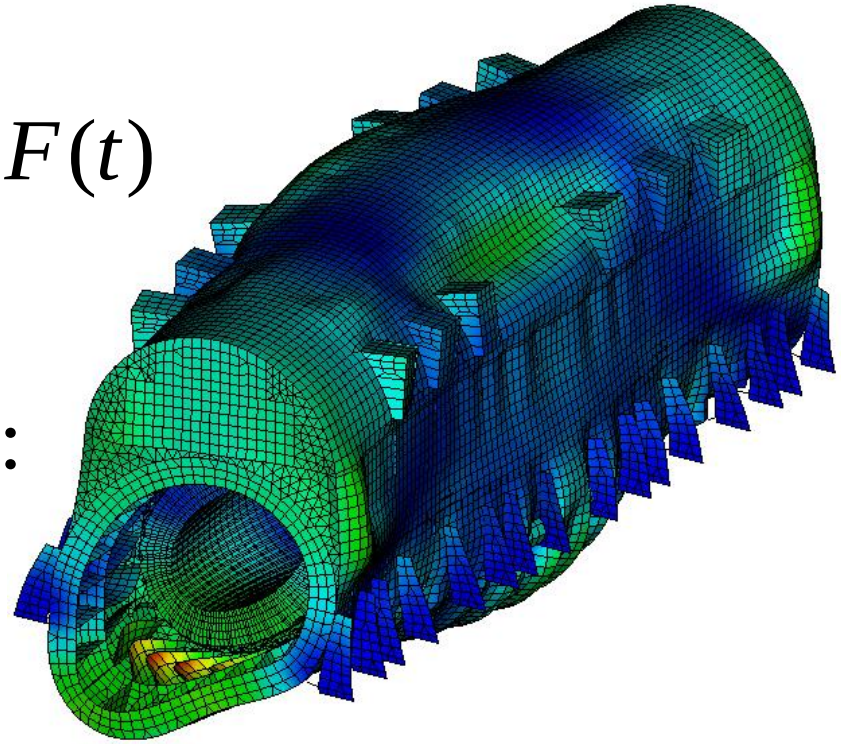
- Discretization → Matrix equation



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

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

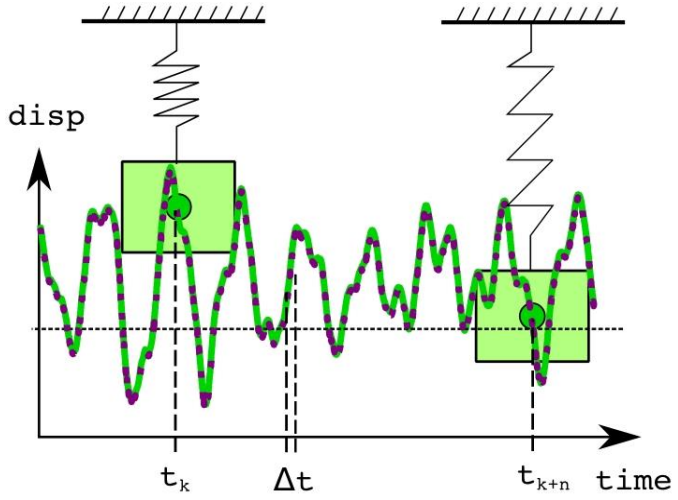
$X(t_i)$:



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 the time history $\ddot{X}(t)\,;\ \dot{X}(t)\,;\ X(t)$

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

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

Transient analysis: implementation

- We need
  - Structural matrices M (mass) , C (damping), K (stiffness)
  - Excitation field E 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**
  - Loads → **AFFE_CHAR_MECA**

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

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

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

Harmonic Analysis: principles

- We seek the steady-state response
  - Transient analysis for steady-state oscillatory excitation & response

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

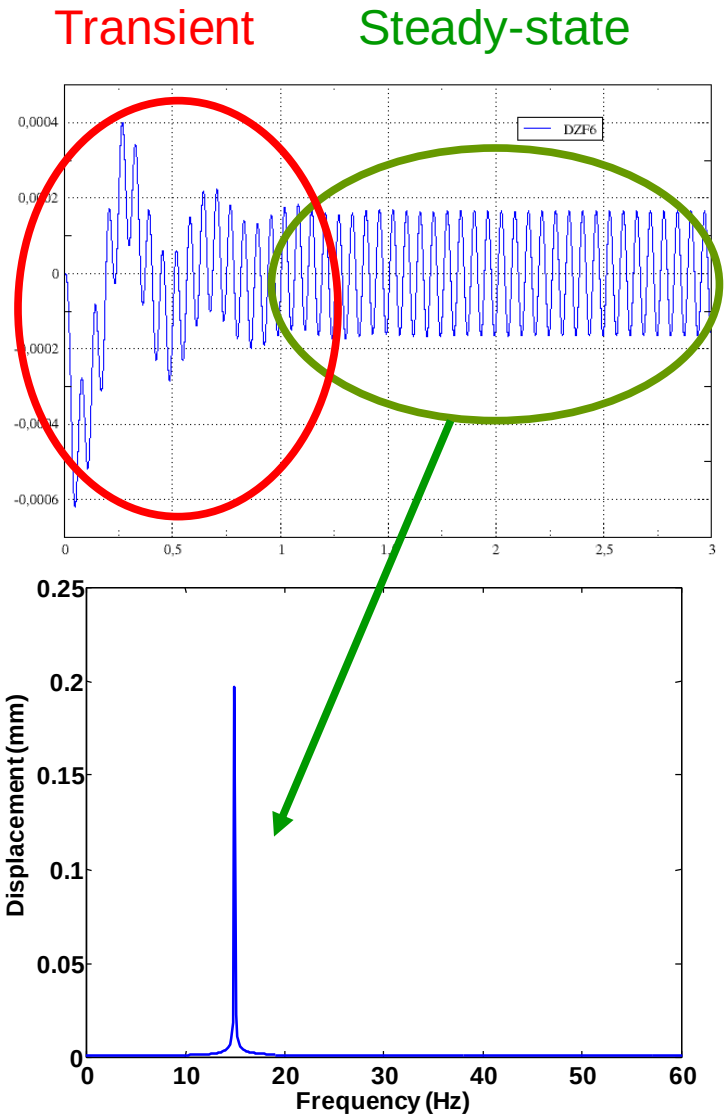
  - Alternative approach for steady-state motion

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

⬇ (Fourier transform)

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

  - Frequency-per-frequency computation

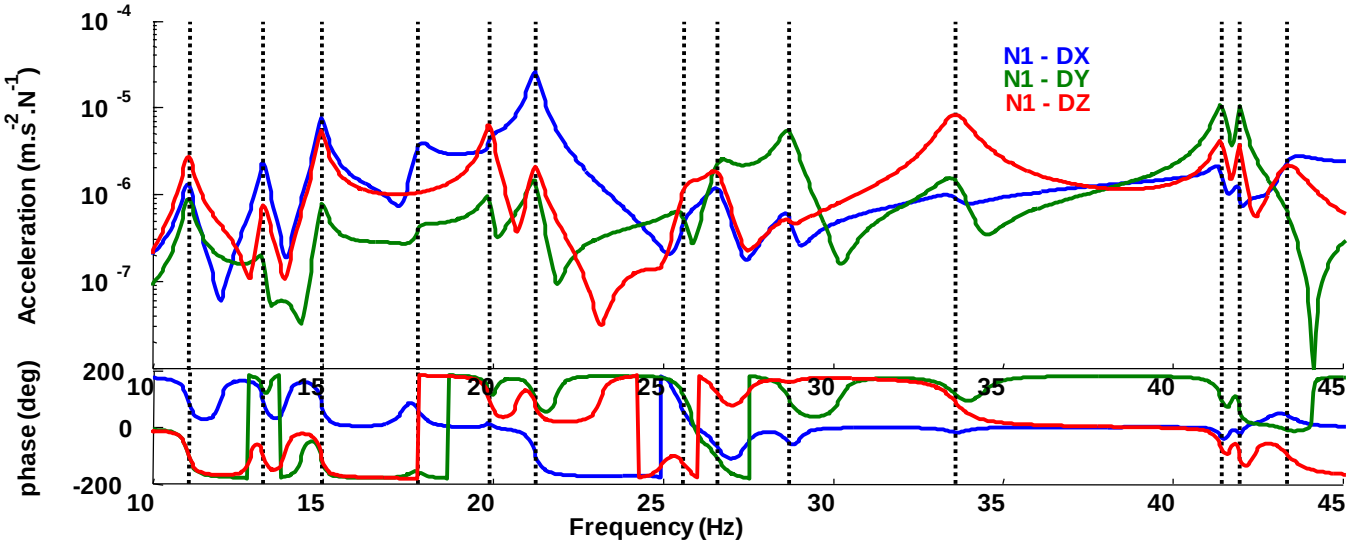


Harmonic analysis: implementation

- We need
  - Structural matrices M (mass) , C (damping), K (stiffness)
  - Excitation field E and its frequency evolution u(ω)

- Same requirements and model assembly as in transient analysis

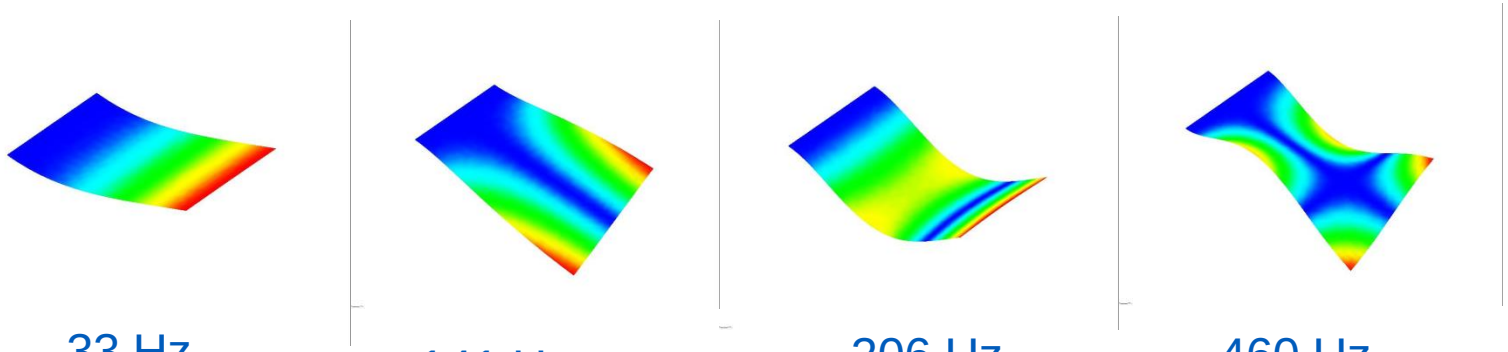
- Resolution with `DYNA_VIBRA(TYPE_CALCUL='HARM',BASE_CALCUL='PHYS')`



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

What are natural frequencies & normal modes ?

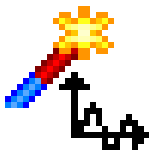
- They represent motions where the interchange between the 2 forms of energy (kinetic & potential) can easily occur
- Mathematically : $(\phi_k, \omega_k) \mid [-\omega_k{}^2 M + K]\,\phi_k = 0 \quad ; \phi_k \neq 0$
- Single DOF : $[-\omega_o{}^2 m + k]\,x_0 = 0 \quad ; x_0 \neq 0 \quad \Rightarrow \quad \omega_o = \sqrt{\frac{k}{m}}$
- They depend on the boundary conditions but not on the external loading
  - Natural frequency (or Eigen frequency) : number of oscillations per second
  - Normal mode (or Eigen vector) : corresponding deformation shape



33 Hz | 141 Hz | 206 Hz | 460 Hz

How to get normal modes with *Code_Aster* ?

- The easiest way : the « wizard »
  - Graphical user-interface
  - Available within SalomeMeca
  - Good starting point for transient / harmonic analysis



- The usual way : **CALC_MODES**
  - M, K assembled matrices are required (+ C matrix if needed)
  - Several options are available to refine the computed modes (their number, range, etc.) as well as the algorithm parameters.
  - In most cases the default configuration works well
    - **First 10 frequencies**

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

    - **Frequencies between f1=0.0 Hz and f2=60.0 Hz**

```
modes = CALC_MODES ( MATR_RIGI= matrigi, MATR_MASS= matmass,
                     OPTION='BANDE', CALC_FREQ=_F(FREQ=(0.,60.))
```

Why are normal modes useful ?

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

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

Mode #1 Modal coordinate #1

- Using modal reduction, the analysis cost is considerably reduced (the number of unknowns, degrees of freedom, is reduced to the number of modes in the considered basis)

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

  - Simple **rule of thumb** : Natural frequencies up to **2 x maximal input frequency**
  - Always check the validity of the modal basis (increase the number of modes, static correction)

Model reduction: implementation

- Same requirements and model assembly as transient or harmonic analysis (M, K, C)
- Compute normal modes → **CALC_MODES**
- Model projection (matrices and excitation field) → **PROJ_BASE**
- Resolution:
  - 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`

Damping

- Viscous (discrete) damping
  - Dashpot modeling using discrete elements

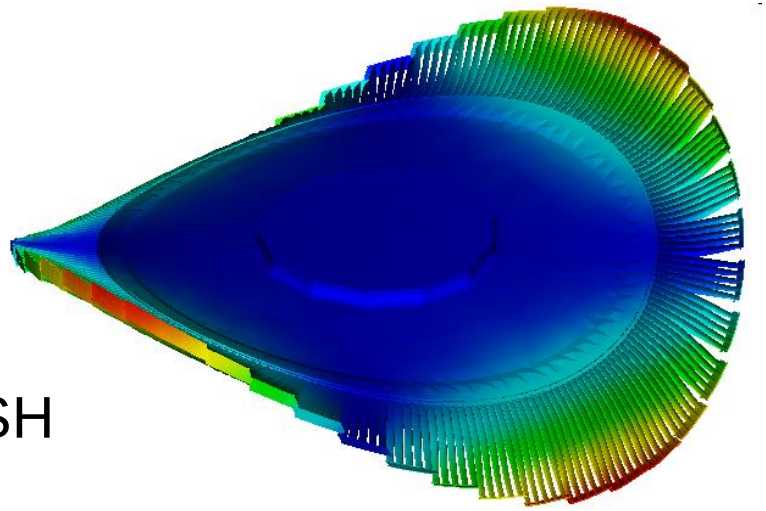
$$M\ \ddot{X} + C\ \dot{X} + K\ X = E\ u(t)$$

- Rayleigh damping $C = \alpha\ K + \beta\ M$
  - **DEFI_MATERIAU** : **AMOR_ALPHA** & **AMOR_BETA**
  - **ASSEMBLAGE (OPTION = 'AMOR_MECA')** → Matrix C
- Structural (hysteretic) damping
  - Reserved for Frequency Response Analysis $-\omega^2 M\ \ddot{X} + (j\kappa + 1) K\ X = E u(\omega)$
  - **ASSEMBLAGE (OPTION='RIGI_MECA_HYST')** → Matrix C
- Modal damping
  - **DYNA_VIBRA(TYPE_CALCUL = 'TRAN',BASE_CALCUL='GENE')**
    - **AMOR_REDUIT = 0.01**

$$\ddot{\eta}_i + \xi_i \dot{\eta}_i + \omega_i{}^2 \eta_i = \frac{\phi_i^T E}{m_i} u(t)$$

  - **DYNA_VIBRA(TYPE_CALCUL='TRAN',BASE_CALCUL='PHYS')**
    - Can also be used without model reduction but require a modal basis
    - **AMOR_MODAL=_F(MODES=modes, AMOR_REDUIT = 0.01)**

Printing & visualization of normal modes



- Within SALOME_MECA/PARAVIS
  - “Macro/modes”
  - Magnitudes are arbitrary

- Or with other tools : by example GMSH

```
IMPR_RESU(      FORMAT='GMSH', UNITE=37,
                RESU=_F(RESULTAT=modes,
                NOM_CHAM='DEPL',
                TYPE_CHAM='VECT_3D', NOM_CMP=('DX','DY','DZ',),),)
```

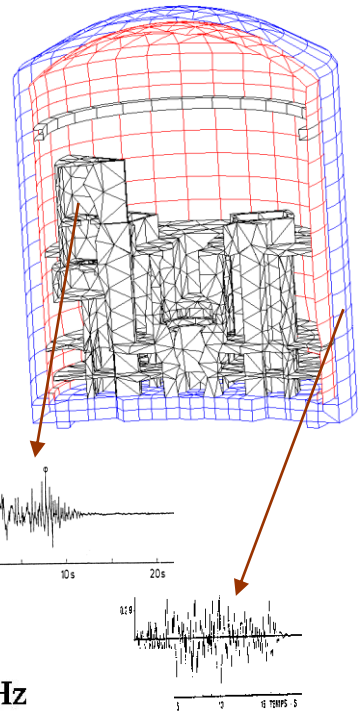
- Printing frequencies in the .resu file

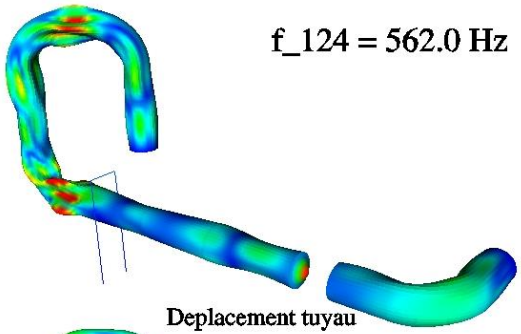
```
IMPR_RESU(      RESU=_F(RESULTAT=modes, TOUT_CHAM='NON', NOM_PARA=('FREQ',)))
```

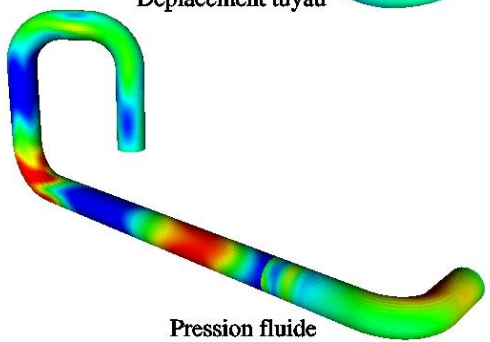
- The normal modes are simple displacement fields

Wider scopes

- Limits of modal analysis
  - Linear behaviour only
  - With the exception of point contact (in `DYNA_VIBRA(TYPE_CALCUL='TRAN',BASE_CALCUL='GENE')`)
- For non-linear dynamics
  - `DYNA_NON_LINE`
- Specific topics
  - Fluid-structure interactions
    - Potential flow
    - Acoustic waves
    - Surface waves
  - Stochastic
  - Seismic analysis
  - …







Tips

- EFICAS can help
  - Obtain the right syntax (this does not guarantee the rightness of the model !)
  - Translation from one version to another (syntax update)
- 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**
  - Good check of the Finite Elements model (materials, element characteristics, BC, etc.)
  - Eigen Frequencies and modes give a good insight on the vibration modes of the structure
  - Eigen frequencies can be essential for a suitable choice of integration time step / frequencies for a harmonic analysis

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.

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 to assure consistency between matrixes and between vectors

Syntax example : normal modes computation

- Computation with **CALC_MODES**
  - First 10 frequencies

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

  - All frequencies between f1=0.0 Hz and f2=100.0 Hz

```
modes = CALC_MODES ( MATR_RIGI= matrigi, MATR_MASS= matmass,
                     OPTION='BANDE', CALC_FREQ=_F(FREQ= (0.,100.) )
```

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

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

- Printing the Eigen 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
  - Excitation field
    - `CHFNO =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('fx'), CHARGE=CHFNO, OPTION= 'CHAR_MECA'),)
```

  - Function of time
    - Either **FORMULE** : mathematical expression of time (python format)
      - 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 of the force 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',
                                     MATR_MASS=matmass, MATR_RIGI=matrigi,
                                     SCHEMA_TEMPS=_F(SCHEMA='NEWMARK'),
                                     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='EULER'),
               MATR_MASS=maspro, MATR_RIGI=ripro,
               INCREMENT=_F(INST_FIN=tfin, PAS=pa,),
               EXCIT=_F(VECT_ASSE_GENE=fxpro,
                        FONC_MULT=rimpuls))
```

- Back to physical coordinates
  - Natural way (whole model)

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

    can be costly, if the model is huge or the number of steps is significant!

  - Efficient way to follow the trajectories or stresses at some points/elements :

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