Modal Analysis

Code_Aster, Salome-Meca course material
GNU FDL licence (http://www.gnu.org/copyleft/fdl.html)
Outline

- What are natural frequencies & normal modes?
- Presentation of the eigenvalue problem
- Resolution methods
- Implementation in Code_Aster
- Post-processing & verification
- Tips
What are natural frequencies & normal modes?

- They represent motions where the interchange between the 2 forms of energy (kinetic & potential) can easily occur.

- Mathematically: \((\phi_k, \omega_k) \mid [-\omega_k^2 M + K] \phi_k = 0 \; ; \; \phi_k \neq 0\)

- Single DOF: \([-\omega_o^2 m + k] x_0 = 0 \; ; \; x_0 \neq 0 \Rightarrow \omega_o = \sqrt{\frac{k}{m}}\)

- 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
The eigenvalue problem

Two types of problems

- Generalised eigenvalue problem – for undamped systems

  \[ M\ddot{u} + Ku = 0 \quad \Rightarrow \quad (K - \lambda M) \Phi = 0 \]

- Quadratic eigenvalue problem – for damped systems

  \[ M\ddot{u} + Cu + Ku = 0 \quad \Rightarrow \quad (K + \lambda C + \lambda^2 M) \Phi = 0 \]

The unknowns

- Eigen values (pulsations) \[ 0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n \quad (\lambda = \omega^2) \]
- Eigen vectors (modes) \[ \Phi_1, \Phi_2, \Phi_3, \ldots, \Phi_n \]
Resolution methods – command CALC_MODES

Modal calculation methods can be classified in 3 categories:

■ Power iteration methods
  ■ Scope: calculation of extreme values of the spectrum
  ■ Pros: simplicity, good estimation of the eigenvector in a few iterations
  ■ Cons: convergence problems, poor identification of multiple modes
  ■ Code_Aster: OPTION = 'PROCHE'/'SEPARE'/'AJUSTE'

■ Subspace iteration methods
  ■ Scope: calculation of a part of the spectrum
  ■ Pros: reduced size of the problem, less memory required
  ■ Cons: convergence problems, costly pre- and post-processing steps
  ■ Code_Aster: OPTION = 'PLUS_PETITE'/'BANDE'/'CENTER'/'TOUT'
    SOLVEUR_MODAL = _F( METHODE = 'SORENSEN'/'TRI_DIAG'/'JACOBI' ... )

■ QR/QZ type methods
  ■ Scope: calculation of the whole spectrum (+ optional filtering of the results)
  ■ Pros: robustness
  ■ Cons: slow convergence, memory and CPU costs (maximum 10^3 DOFs)
  ■ Code_Aster: OPTION = 'PLUS_PETITE'/'BANDE'/'CENTER'/'TOUT'
    SOLVEUR_MODAL = _F( METHODE = 'QZ' ... )
Power iteration methods: algorithm illustration

- Standard eigenvalue problem $[A] \cdot x = \lambda \cdot x$
- Determine the largest eigenvalue $\lambda_1$

Algorithm:
- arbitrary initial vector $v_0$
- for $i = 1, \ldots$
  - $w = [A] \cdot v_{i-1}$
  - $\lambda^i = \frac{\|[A] \cdot w\|}{\|w\|}$
  - $v_i = \frac{w}{\lambda^i}$ if $\lambda^i \neq 0$
- \(\lambda^i, v_i \rightarrow (\lambda_1, u_1)\)

Proof:
- \((\lambda_j, u_j)_{j=1,n}\) eigenmodes and eigenvalue $s$
- $|\lambda_1| > |\lambda_2| > |\lambda_3| > \ldots$
- $v_0 = \sum_j (v_0, u_j) u_j$
- $\frac{1}{\lambda_1^k} A^k v_0 = (v_0, u_1) u_1 + \sum_{j=2,n} \left( \frac{\lambda_j}{\lambda_1} \right)^k (v_0, u_j) u_j$

\(\neq 0\) if possible
Convergence rate
Power iteration methods: different options

- Smallest eigenvalue: $[A]^{-1}$
- Eigenvalue closest to $\sigma$: $[A - \sigma \cdot I]^{-1}$
- Rayleigh quotient: $r(x) = \frac{x^T A x}{x^T x}$: if $x$ is an eigen vect or $\Rightarrow r(x) = \lambda$

Algorithm

\[
\text{for } i = 1, \ldots, \\
\quad \sigma_i = r(v_{i-1}) \\
\quad \text{solve for } w: [A - \sigma_i I] \cdot w = v_{i-1} \\
\quad v_i = \frac{w}{\|w\|}
\]

\text{OPTION_INV = 'RAYLEIGH'}
Subspace iteration methods

- Extrapolation of the iteration method on one eigenvalue to a subspace => methods able to compute many eigenvalues at once
  - More efficient than iterations on one value

- Many “flavours”
  - Lanczos
  - Bathe and Wilson
  - IRAM – Sorensen
  - …

- Always use `METHODE = 'SORENSEN'` with default options in the command `CALC_MODES` as your first attempt
  - Robust
  - Efficient
Verifications

- Convergence of each method
- *A posteriori* error norm on the eigen-vectors/values

Eigenvalue problem

\[ [K - \omega^2 \cdot M] \cdot x = 0 \]

\[ error_i = \left\| [K] \cdot v_i - \omega_i^2 [M] \cdot v_i \right\| \]

if \( error_i \leq \text{threshold} \)

OK

else

< F > atal error

- Sturm verification: eigenvalue count in a given range

- Frequencies requested by the user
- Frequencies calculated but not requested by the user
Implementation in **Code_Aster**

- Main steps for carrying out a modal analysis

---

**Model construction**
- Mesh
- Materials
- Element properties
- Boundary conditions

**Elementary matrices**
- Mass \( m_i \)
- Stiffness \( k_i \)
- Damping \( c_i \)

**Macro ASSEMBLAGE**

**Numbering the DOFs**

**Matrices Assembly**
- \( M \)
- \( K \)
- \( C \)

**Modal Calculation**

**Post-processing**
- Normalization
- Extraction
- Printing

---

**Code_Aster solving commands for modal calculation**

- **CALC_MODES** uses subspace and inverse methods defined according to the `OPTION =` and `SOLVEUR_MODAL=_F( METHODE = )` keywords

- **MODE_ITER_CYCL** modal calculation of a structure with cyclic symmetry
Subspace methods in *Code_Aster*: **CALC_MODES**

Arnoldi IRAM – Sorensen
Bathe and Wilson
Lanczos

Three subspace iterative methods are available
- Same principle as power iteration methods but formulated in subspaces (several frequencies are calculated at a time)
- Best efficiency is obtained with the Arnoldi (IRAM – Sorensen) method

Recommended method
- **SORENSEN**
  - Fast and low CPU cost
  - Robust
  - Calculates multiple modes (same frequency but different/orthogonal modes)
  - Can obtain rigid body modes (unconstraint movements)
    - *(OPTION = 'BANDE' with lower bound 0. or slightly negative)*
CALC_MODES command options

» Calculate the nbf lowest frequencies
  - OPTION = 'PLUS_PETITE'
  - NB_FREQ = nbf

» Calculate all frequencies in a given range \([f_{\text{min}}, f_{\text{max}}]\)
  - OPTION = 'BANDE'
  - FREQ = \([f_{\text{min}}, f_{\text{max}}]\)

» Calculate all frequencies around a given reference \(f_{\text{ref}}\)
  - OPTION = 'CENTER'
  - FREQ = \(f_{\text{ref}}\)

» 3 methods
  - 'JACOBI' Bathe and Wilson method
  - 'TRI_DIAG' Lanczos method
  - 'SORENSEN' evolution of the Arnoldi method
    preferred method (by default)
**CALC_MODES — recommended method SORENSEN**

**Syntax**

```plaintext
mode = CALC_MODES (MATR_RIGI = K_MATR,
                     MATR_MASS = M_MATR,
                     OPTION    = 'PLUS_PETITE' / 'BANDE',
                     CALC_FREQ = _F( NMAX_FREQ = n / FREQ = (f1,f2)),
                     SOLVEUR_MODAL = _F( METHODE = 'SORENSEN' ) )
```

**Many options and parameters**

*Normally reserved for advanced users*
Les fréquences calculées sont comprises entre :
Fréquence inférieure : 7.02615E+01
Fréquence supérieure : 4.20512E+02

Calcul modal : Méthode d’itération simultanée
Méthode de Sorensen

numéro  fréquence (HZ)  norme d'erreur
1       7.02615E+01  9.87164E-13
2       1.28256E+02  3.73093E-13
3       1.40530E+02  3.24591E-13
4       1.89140E+02  2.11480E-13
5       2.11116E+02  1.53475E-13

Norme d'erreur moyenne : 1.72031E-13 => error norm check (by default error < 1.E-6)

Vérification à posteriori des modes
Dans l’intervalle ( 7.00856E+01 , 4.21562E+02 ) il y a bien 20 fréquence(s). => Mode number check
Pre/post-processing command: INFO_MODE

Count the number of frequencies in (a) given interval(s)

\[ f_{\text{table}} = \text{INFO\_MODE}(\text{MATR\_RIGI} = K_{\text{MATR}}, \text{MATR\_MASS} = M_{\text{MATR}}, \]
\[ \text{FREQ} = (f_1, f_2, f_3, \ldots) \]

\[ \text{TABLE\_FREQ} = f_{\text{table}} \] : can be used as input to CALC_MODES

Special interest for processor mapping to frequency intervals in parallel computing

Recommended interval size for efficient CALC_MODES

- 10 to 30 frequencies per interval
High Performance Computing: \texttt{CALC MODES/INFO MODE}

Frequency intervals & parallelisation

- Determine the frequency distribution for the desired range by partitioning the range into smaller intervals
  - Saves CPU time and memory and improves algorithm convergence
  - Possible parallel computing of the partitioned range

Tip: choose intervals of uniform distribution of modes
- Count the modes in each of the intervals (\texttt{INFO MODE})
- Change the range partitioning as to acquire an \textit{almost} constant number of modes in each interval (balance the loads for multi processing instances)

Parallelisation technique
- First level: many frequency intervals
  - \texttt{CALC_FREQ = \_F ( FREQ = (f_1,f_2,f_3,...) )}
  - \textbf{Gain in time}/Gain in memory
- Second level: direct parallel linear solver \texttt{`MUMPS’}
  - Gain in time / \textbf{Gain in memory}
Resulting data structure: storage indices

**NUME_ORDRE**

- In the `mode_meca` data structure, the modes are sorted by ascending frequency. **NUME_ORDRE** is the mode position in the calculated data structure.

**NUME_MODE**

- Mode position in the full spectrum

<table>
<thead>
<tr>
<th>NUME_ORDRE</th>
<th>NUME_MODE</th>
<th>FREQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23</td>
<td>2.51972E+01</td>
</tr>
<tr>
<td>2</td>
<td>24</td>
<td>2.63652E+01</td>
</tr>
<tr>
<td>3</td>
<td>25</td>
<td>2.78854E+01</td>
</tr>
<tr>
<td>4</td>
<td>26</td>
<td>2.79978E+01</td>
</tr>
<tr>
<td>5</td>
<td>27</td>
<td>2.89328E+01</td>
</tr>
</tbody>
</table>
Additional remarks: power iteration methods

**OPTION = 'PROCHE'/'SEPARE'/'AJUSTE'**

- Direct and Rayleigh inverse options
  - Costly algorithms: many factorizations are required
  - **OPTION = 'PROCHE'** does not calculate multiple modes (modal position false)
  - Useful to calculate some eigenvalues
  - Refine the calculation of an eigenvalue (and eigenvector) found by another algorithm

- Verification
  - Error norm on eigenmodes
Resulting data structure : saved parameters

- **Generalized mass or stiffness**

\[ m_x = x^T M x \quad k_x = x^T K x \quad \omega = \frac{k_x}{m_x} \]

\[ v(t) = \sum_i \alpha_i(t)x_i \]

\[ M\ddot{v} + K v = f \quad \Rightarrow \quad m_x \dddot{x}_i + k_x x_i = f_i \]

- **Effective modal mass (unitary)**

**MASS_EFFE_ (UN_) D ** *

- **Participation Factor**

**FACT_Partici_D ** *

\[ U_d \text{ normalized vector in } \mathbb{R}^3 \]

\[ m_{x,d} = \frac{(x^T MU_d)^2}{x^T M x} \quad \text{effective modal mass} \]

\[ \sum_i m_{x_i,d} = \text{mass}_{\text{structure}} \]

\[ \bar{m}_{x,d} = \frac{1}{\text{mass}_{\text{structure}}} m_{x,d} \quad \text{unit effective modal mass} \]

\[ P_{x,d} = \left(\frac{x^T MU_d}{x^T M x}\right)^2 \quad \text{participation factor} \]

Property: \[ \sum_i \left(P_{x_i,d}\right)^2 m_{x_i} = \text{mass}_{\text{structure}} \]
Post-processing command: \texttt{NORM\_MODE}

- Normalising
  - By default: largest physical DOF equal to 1

\begin{itemize}
  \item \texttt{NORME = ‘MASS\_GENE’} \quad x^T M x = 1
  \item \texttt{NORME = ‘RIGI\_GENE’} \quad x^T K x = 1
\end{itemize}

- Largest physical DOF equal to 1 taken within a group of DOF

\begin{itemize}
  \item \texttt{NORME = ‘TRAN’} \quad (DX, DY, DZ)
  \item \texttt{NORME = ‘TRAN\_ROTA’} \quad (DX, DY, DZ, DRX, DRY, DRZ)
  \item \texttt{NOEUD = no} and \texttt{NOM\_CMP=DOF} \quad DOF(no) = 1
  \item \texttt{SANS\_CMP = (CMP1, \ldots)} \quad group without these DOFs
  \item \texttt{AVEC\_CMP = (CMP1, \ldots)} \quad group composed of these DOFs
\end{itemize}
Post-processing command: `EXTR_MODE`

- Concatenate `mode_meca` data structures produced by `CALC_MODES`

- Extract and sort the modes by:
  - Elimination
    - `MASS_GENE` < threshold
    - `MASS_EFFE_UN` < threshold
  - Selection
    - `NUME_ORDRE` list
    - `NUME_MODE` list

- Print the total value of `MASS_*`

- Detection of duplicate modes from several `mode_meca` to be appended
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=\text{modes},
                      NOM_CHAM='DEPL',
                      TYPE_CHAM='VECT\_3D', NOM_CMP=('DX', 'DY', 'DZ',),),)

  IMPR_RESU( RESU=_F(RESULTAT=\text{modes}, TOUT_CHAM='NON', NOM_PARA=('FREQ',)))

- The normal modes are simple displacement fields
Tips

- Weigh the model (POST_ELEM)

- Estimate the number of frequencies (INFO_MODE)

- Use the 'BANDE' option

- Read doc U2.06.01 => practical handbook of modal computation

- Use parallelisation if the number of frequencies is significant

- Take into account the error or alarm messages <F> or <A>
  - Detecting null pivot, shift change
  - Important norm error on a mode
  - Calculated number of frequencies not consistent with the STURM count verification
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.