Quick review on elasticity
Finite element method (FE)
Finite difference method (FD)
Extension to dynamics

Ferdinando Auricchio

1 DICAR – Dipartimento di Ingegneria Civile e Architettura, Università di Pavia, Italy
2 IMATI – Istituto di Matematica Applicata e Tecnologie Informatiche, CNR, Italy

April 29, 2014
3D elasticity: compact notation

- **Three-dimensional body**: $\Omega \subset \mathcal{R}^3$
- **Deformable body**: variable distance between points
- **Field equations**:

\[
\begin{align*}
\text{equilibrium eq.} & \quad \text{div } \sigma + b = 0 \quad \text{in } \Omega \\
\text{constitutive eq.} & \quad \sigma = D\varepsilon \quad \text{in } \Omega \\
\text{compatibility eq.} & \quad \varepsilon = \nabla^s u \quad \text{in } \Omega
\end{align*}
\]

with:

\[
\begin{align*}
\sigma : & \quad \text{stress (second-order) tensor} \\
u : & \quad \text{displacement vector} \\
\text{div} : & \quad \text{divergence operator} \\
D : & \quad \text{elasticity (fourth-order) tensor}
\end{align*}
\]

\[
\begin{align*}
\varepsilon : & \quad \text{strain (second-order) tensor} \\
b : & \quad \text{body force vector} \\
\nabla^s : & \quad \text{symmetric gradient operator}
\end{align*}
\]

- **Boundary conditions**:

\[
\begin{align*}
\text{imposed displ.} & \quad u = \bar{u} \quad \text{on } \partial \Omega_u \\
\text{imposed forces} & \quad \sigma n = \bar{t} \quad \text{on } \partial \Omega_t
\end{align*}
\]

with $\bar{u}$ & $\bar{t}$: assigned data; $n$: outward normal to $\partial \Omega$; $\partial \Omega_u \cup \partial \Omega_t = \partial \Omega$

- **Linear elasticity assumption. Easy to generalize** $\sigma = \sigma(\varepsilon)$!
3D elasticity: indicial notation I

- **Recall:** summation convention, i.e. repeated indices imply summation
- **Recall:** comma indicates derivation

**Field equations:**

<table>
<thead>
<tr>
<th>Type</th>
<th>Equation</th>
<th>Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equilibrium eq.</td>
<td>$\sigma_{ij,j} + b_i = 0$</td>
<td>$\Omega$</td>
</tr>
<tr>
<td>Constitutive eq.</td>
<td>$\sigma_{ij} = D_{ijkl} \varepsilon_{kl}$</td>
<td>$\Omega$</td>
</tr>
<tr>
<td>Compatibility eq.</td>
<td>$\varepsilon_{ij} = \frac{1}{2} (u_{i,j} + u_{j,i})$</td>
<td>$\Omega$</td>
</tr>
</tbody>
</table>

with:

\[
\begin{aligned}
\sigma_{ij} &: \text{ stress (second-order) tensor} \\
u_i &: \text{ displacement vector} \\
D_{ijkl} &: \text{ elasticity (fourth-order) tensor} \\
\varepsilon_{ij} &: \text{ strain (second-order) tensor} \\
b_i &: \text{ body force vector}
\end{aligned}
\]

**Boundary conditions:**

\[
\begin{aligned}
imposed \text{ displ.} &: u_i = \bar{u}_i \quad \text{on} \quad \partial\Omega_u \\
imposed \text{ forces} &: \sigma_{ij} n_j = \bar{t}_i \quad \text{on} \quad \partial\Omega_t
\end{aligned}
\]

with $\bar{u}_i \& \bar{t}_i$: assigned data; $n_i$: outward normal to $\partial\Omega_t$; $\partial\Omega_u \cup \partial\Omega_t = \partial\Omega$. 
3D elasticity: engineering notation

- We take advantage of symmetry, i.e.:

\[
\begin{align*}
\sigma &= \sigma^T, & \varepsilon &= \varepsilon^T, & \mathcal{D} &= \mathcal{D}^T, & \mathcal{D} &= \mathcal{D}^t \\
\sigma_{ij} &= \sigma_{ji}, & \varepsilon_{ij} &= \varepsilon_{ji}, & \mathcal{D}_{ijkl} &= \mathcal{D}_{klij}, & \mathcal{D}_{ijkl} &= \mathcal{D}_{ijlk}
\end{align*}
\]

- \(\{\sigma\}\), stress (second-order) tensor, in vector notation:

\[
\{\sigma\} = \{\sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{12}, \sigma_{23}, \sigma_{31}\}^T
\]

- \(\{b\}\), body force, in vector notation:

\[
\{b\} = \{b_1, b_2, b_3\}^T
\]

- \(\{\varepsilon\}\), strain (second-order) tensor, in vector notation:

\[
\{\varepsilon\} = \{\varepsilon_{11}, \varepsilon_{22}, \varepsilon_{33}, \varepsilon_{12}, \varepsilon_{23}, \varepsilon_{31}\}^T \quad \text{or} \quad \{\varepsilon^M\} = \{\varepsilon_{11}, \varepsilon_{22}, \varepsilon_{33}, 2\varepsilon_{12}, 2\varepsilon_{23}, 2\varepsilon_{31}\}^T
\]

- \([\mathcal{D}]\) elasticity (fourth-order) tensor, in matrix notation:

\[
[\mathcal{D}] = \begin{bmatrix}
\mathcal{D}_{1111} & \mathcal{D}_{1122} & \mathcal{D}_{1133} & \mathcal{D}_{1112} & \mathcal{D}_{1123} & \mathcal{D}_{1113} \\
\mathcal{D}_{1122} & \mathcal{D}_{2222} & \mathcal{D}_{2233} & \mathcal{D}_{2212} & \mathcal{D}_{2223} & \mathcal{D}_{2213} \\
\mathcal{D}_{1133} & \mathcal{D}_{2233} & \mathcal{D}_{3333} & \mathcal{D}_{3312} & \mathcal{D}_{3323} & \mathcal{D}_{3313} \\
\mathcal{D}_{1112} & \mathcal{D}_{2212} & \mathcal{D}_{3312} & \mathcal{D}_{1212} & \mathcal{D}_{1223} & \mathcal{D}_{1213} \\
\mathcal{D}_{1123} & \mathcal{D}_{2223} & \mathcal{D}_{3323} & \mathcal{D}_{1223} & \mathcal{D}_{2323} & \mathcal{D}_{2313} \\
\mathcal{D}_{1113} & \mathcal{D}_{2213} & \mathcal{D}_{3313} & \mathcal{D}_{1212} & \mathcal{D}_{2313} & \mathcal{D}_{1313}
\end{bmatrix}
\]
3D elasticity: engineering notation I

* Take advantage of vector Voigt notation for second-order tensors (and matrix notation for fourth-order tensors)

** Field equations: **

<table>
<thead>
<tr>
<th>Equation</th>
<th>Description</th>
<th>Domain ( \Omega )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equilibrium eq.</td>
<td>([L]^T {\sigma} + {b} = {0})</td>
<td>(\Omega)</td>
</tr>
<tr>
<td>Constitutive eq.</td>
<td>({\sigma} = [D][M]{\varepsilon})</td>
<td>(\Omega)</td>
</tr>
<tr>
<td>Compatibility eq.</td>
<td>({\varepsilon} = [L]{u})</td>
<td>(\Omega)</td>
</tr>
</tbody>
</table>

with:

- \(\{\sigma\}\) : stress (second-order) tensor
- \(\{u\}\) : displacement vector
- \([D]\) : elasticity (fourth-order) tensor
- \([M]\) : double-contraction
- \(\{\varepsilon\}\) : strain (second-order) tensor
- \(\{b\}\) : body force vector
- \([L]\) : diff. operator (to be defined)

** Boundary conditions: **

\[
\begin{cases}
\text{imposed displ.} & \{u\} = \{\bar{u}\} \\
\text{imposed forces} & \{\sigma\} \otimes \{n\} = \{\bar{t}\}
\end{cases}
\quad \text{on} \quad \partial\Omega_u \cup \partial\Omega_t = \partial\Omega
\]

with \(\{\bar{u}\}\) & \(\{\bar{t}\}\): assigned data; \(\{n\}\) : outward normal to \(\partial\Omega_t\); \(\partial\Omega_u \cup \partial\Omega_t = \partial\Omega\)
Voigt representation requires the introduction of some special operators (!)

- Differential operator $[\mathbf{L}]^T$ correspondent to $\text{div}$-operator

\[
[L]^T = \begin{bmatrix}
\frac{\partial}{\partial x} & 0 & 0 & \frac{\partial}{\partial y} & 0 & \frac{\partial}{\partial z} \\
0 & \frac{\partial}{\partial y} & 0 & \frac{\partial}{\partial x} & \frac{\partial}{\partial z} & 0 \\
0 & 0 & \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial y} & \frac{\partial}{\partial x}
\end{bmatrix}
\]

- Algebraic operator $[\mathcal{M}]$ corresponding to double-contraction

\[
[M] = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 2 & 0 \\
0 & 0 & 0 & 0 & 0 & 2
\end{bmatrix}
\]
3D elasticity I

- For a **linear isotropic material**, constitutive equations specializes as follows:

\[
\sigma = D \varepsilon = \lambda \operatorname{tr}(\varepsilon) \mathbf{1} + 2\mu \varepsilon
\]

where

\[
\operatorname{tr}(\varepsilon) = \varepsilon : \mathbf{1} = \varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33}
\]

\[
D = [\lambda (\mathbf{1} \otimes \mathbf{1}) + 2\mu \mathbb{I}]
\]

- In engineering notation

\[
\{\sigma\} = [D^M]\{\varepsilon\} = \lambda \operatorname{tr}(\varepsilon) \{\mathbf{1}\} + 2\mu \{\varepsilon\}
\]

where

\[
\{\mathbf{1}\} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad [D^M] = [D][M] = \\
\begin{bmatrix}
\lambda + 2\mu & \lambda & \lambda & 0 & 0 & 0 \\
\lambda & \lambda + 2\mu & \lambda & 0 & 0 & 0 \\
\lambda & \lambda & \lambda + 2\mu & 0 & 0 & 0 \\
0 & 0 & 0 & 2\mu & 0 & 0 \\
0 & 0 & 0 & 0 & 2\mu & 0 \\
0 & 0 & 0 & 0 & 0 & 2\mu
\end{bmatrix}
\]
For **linear isotropic material**, constitutive equations written also in different forms

- Split stress and strain into volumetric and deviatoric components

\[
\begin{align*}
\sigma &= \rho \mathbf{1} + \mathbf{s} \\
\epsilon &= \frac{\theta}{3} \mathbf{1} + \mathbf{e}
\end{align*}
\]

with

\[
\begin{align*}
\rho &= \frac{1}{3} \text{tr}(\sigma) = \frac{1}{3} \sigma : \mathbf{1} = \frac{1}{3} (\sigma_{11} + \sigma_{22} + \sigma_{33}) \\
\theta &= \text{tr}(\epsilon) = \epsilon : \mathbf{1} = \epsilon_{11} + \epsilon_{22} + \epsilon_{33}
\end{align*}
\]

- Accordingly

\[
\begin{align*}
\mathbf{s} &= \sigma - \rho \mathbf{1} \\
\mathbf{e} &= \epsilon - \frac{\theta}{3} \mathbf{1}
\end{align*}
\]

- Constitutive equation simplifies as follows:

\[
\begin{align*}
\rho &= K \theta \\
\mathbf{s} &= 2 \mu \mathbf{e}
\end{align*}
\]
3D elasticity I

- Corresponding elasticity matrix has simple form in compact notation:

$$\mathbb{D} = \left[ K (\mathbf{1} \otimes \mathbf{1}) + 2\mu \left( \mathbb{I} - \frac{1}{3} \mathbf{1} \otimes \mathbf{1} \right) \right]$$

- In engineering notation

$$\begin{bmatrix}
K + \frac{4}{3} \mu & K - \frac{2}{3} \mu & K - \frac{2}{3} \mu & 0 & 0 & 0 \\
K - \frac{2}{3} \mu & K + \frac{4}{3} \mu & K - \frac{2}{3} \mu & 0 & 0 & 0 \\
K - \frac{2}{3} \mu & K + \frac{4}{3} \mu & K - \frac{2}{3} \mu & 0 & 0 & 0 \\
0 & 0 & 0 & 2\mu & 0 & 0 \\
0 & 0 & 0 & 0 & 2\mu & 0 \\
0 & 0 & 0 & 0 & 0 & 2\mu
\end{bmatrix}$$
Principle of virtual work I

Given a system \( F (\sigma, b, \bar{t}) \) and a system \( D (\delta u, \delta \varepsilon) \), if for any system \( D \) which satisfies compatibility the following equality holds:

\[
\delta L^{ext} = \delta L^{int}
\]

then system \( F \) satisfies equilibrium.

Some definitions:

\[
\delta L^{int} = \int_{\Omega} (\sigma : \delta \varepsilon) \, dV
\]

\[
\delta L^{ext} = \int_{\Omega} (b \cdot \delta u) \, dV + \int_{\partial \Omega_t} (\bar{t} \cdot \delta u) \, dA
\]

- Principle of virtual work constitutes an integral (weak) formulation of equilibrium, i.e. it is proper for the development of FE discretization

Exercise. Prove the equivalence between principle of virtual work and equilibrium equation. Discuss also the assumptions under such a statement.
Total potential energy I

- Consider **total potential energy functional**:

\[
\Pi(u) = \frac{1}{2} \int_{\Omega} \left[ \mathbb{D} \varepsilon(u) : \varepsilon(u) \right] dV - \int_{\Omega} [b \cdot u] dV
\]

with

\[
\varepsilon(u) = \nabla^s u = \frac{1}{2} \left[ \nabla u + (\nabla u)^T \right]
\]

where the term relative to external surface forces is neglected.

- Stationarity of \( \Pi \) gives

\[
d\Pi(u, \delta u) = \int_{\Omega} \left[ \mathbb{D} \varepsilon(u) : \varepsilon(\delta u) \right] dV - \int_{\Omega} [b \cdot \delta u] dV = 0 \tag{1}
\]

- Equation 1 implies equilibrium for system \( F(\sigma, b, \bar{t}) \) with \( \sigma \) computed from constitutive and compatibility equation, i.e. \( \sigma = \mathbb{D} \nabla^s u \)

- Stationarity of Total Potential energy constitutes an integral (weak) formulation of equilibrium, i.e. it is proper for development of FE discretization

**Exercise.** Prove the equivalence between stationarity of \( \Pi \) and equilibrium equation under the position that \( \sigma \) is computed from the constitutive equation and the compatibility equation. Discuss also the assumptions introduced to prove such a statement.
Observe that in potential energy approach we enforce in weak form only equilibrium equation and not constitutive nor compatibility.

It is possible to consider more general weak formulations.

Consider **Hu-Washizu functional**

\[
\Pi(u, \varepsilon, \sigma) = \frac{1}{2} \int_{\Omega} [\varepsilon : D\varepsilon] \, dV - \int_{\Omega} [\sigma : (\varepsilon - \nabla^{s} u)] \, dV - \Pi^{\text{ext}}
\]

Term \( \Pi^{\text{ext}} \) represents the contribution due to external loads.

Taking the variation of the functional wrt \( u, \varepsilon \) and \( \sigma \) it is possible to recover equilibrium equation, constitutive equation and compatibility equation.

**Exercise.** Prove the equivalence between stationarity of \( \Pi \) and the full set of elasticity equations. Discuss also the assumptions introduced to prove such a statement.
It is possible to consider other weak formulations

Consider **Hellinger-Reissner functional**

\[
\Pi(u,\sigma) = -\frac{1}{2} \int_{\Omega} \left[ \sigma : D^{-1} \sigma \right] dV + \int_{\Omega} \left[ \sigma : \nabla s u \right] dV - \Pi^{\text{ext}}
\]

Term \( \Pi^{\text{ext}} \) represents the contribution due to external loads

Taking the variation of the functional wrt \( u \) and \( \sigma \) it is possible to recover equilibrium equation and a combination of constitutive and compatibility equation

**Exercise.** Prove the equivalence between stationarity of \( \Pi \) and a partial set of elasticity equations. Discuss also the assumptions introduced to prove such a statement.

**Exercise.** Derive Hellinger-Reissner functional from Hu-Washizu functional.
Potential energy: reduction to 2D case I

- For a 2D elastic problem adopting an engineering notation we may assume:
  \[
  \{\sigma\} = \{\sigma_{11}, \sigma_{22}, \sigma_{12}\}^T
  \]
  \[
  \{\varepsilon\} = \{\varepsilon_{11}, \varepsilon_{22}, \varepsilon_{12}\}^T
  \]
  with \(D\) now a 3 \(\times\) 3 matrix

- In particular for a **plane strain** problem we have:
  \[
  [D^M] = \begin{bmatrix}
  \lambda + 2\mu & \lambda & 0 \\
  \lambda & \lambda + 2\mu & 0 \\
  0 & 0 & 2\mu
  \end{bmatrix}
  \]

- In particular for a **plane stress** problem we have:
  \[
  [D^M] = \begin{bmatrix}
  \bar{\lambda} + 2\mu & \bar{\lambda} & 0 \\
  \bar{\lambda} & \bar{\lambda} + 2\mu & 0 \\
  0 & 0 & 2\mu
  \end{bmatrix}
  \]
  with
  \[
  \bar{\lambda} = \frac{2\lambda\mu}{\lambda + 2\mu}
  \]
**GOAL:** solve elastic boundary value problem in general situation, i.e. for general domains, loading conditions, etc.

- Too difficult to obtain general closed-form solutions
- Closed-form solutions possible only for simple cases, mainly linear!
- Need to resort to numerical approximation
  - Finite element method
  - Finite difference method
Finite-element method (FEM): general approach to compute approximate solution for any general differential equation

Three main steps to construct a finite-element method from any ODE

1. Convert differential to integral form (strong → weak form)
2. Introduce approximation fields (integral → algebraic form)
3. Solution of algebraic problem

- Differential form: natural format for BVP
- Weak form: often associated to variational principles
- Approximation: imply problem discretization
- Algebraic problem (linear or non-linear) easy to solve ⇒ matrix inversion

Apply concepts to a “general” 1D ODE
FEM basic steps II

- J.Fish and T.Belytschko, *A first course in Finite Elements*, Wiley (June 2007)


- T. Belytschko, W.K. Liu, B. Moran *Nonlinear Finite Elements for Continua and Structures*, Wiley; First edition (September 2000)
1D problem: string under tension

**Strong formulation**

Given \( f(x) \), \( \bar{g} \) and \( \bar{h} \), find \( u(x) \) such that:

\[
\begin{align*}
    u_{xx} + f &= 0 \quad \text{in} \quad \Omega \\
    u(0) &= \bar{g} \\
    u_x(l) &= \bar{h}
\end{align*}
\]

**Toward a weak formulation**

- Multiply differential equation by a generic “weight” function \( w(x) \)
  \[
  w : \bar{\Omega} \rightarrow \mathcal{R}
  \]
- Integrate product over problem domain
- Observe that integral is equal to zero
- Integrate by parts
1D problem: weak form I

- Multiply differential equation by a weight function \( w(x) \)
  \[ w(u,_{xx}+f) = 0 \]

- Integrate over domain and use integral additive property
  \[ \int_0^l [w(u,_{xx}+f)] \, dx = 0 \]
  \[ \int_0^l [wu,_{xx}] \, dx + \int_0^l [wf] \, dx = 0 \]

- Use integration by parts
  \[ [wu,_{x}]|_{x=0}^{l} - \int_0^l [w,_{x} u,_{x}] \, dx + \int_0^l [wf] \, dx = 0 \]

- Change sign and impose to choose \( w(x) \) s.t. \( w(0) = 0 \)
  \[ \int_0^l [w,_{x} u,_{x}] \, dx - \int_0^l [wf] \, dx - w(l)\bar{h} = 0 \]
1D problem: weak form II

**Weak formulation**

Given \( f(x) \), \( \bar{g} \), and \( \bar{h} \), find \( u(x) \) such that \( \forall w(x) \):

\[
\int_0^l [w, u_x] \, dx - \int_0^l [wf] \, dx - w(l)\bar{h} = 0 \quad \text{in} \quad \Omega
\]

with

\[
u(0) = \bar{g} \quad , \quad w(0) = 0
\]

- Boundary condition \( u, x(l) = \bar{h} \) has disappeared, i.e. no more required !!
- Easy to obtain weak form from any differential problem

**Mechanical interpretation:** principle of virtual work

- Define internal virtual work \( \mathcal{L}^{\text{int}} \) and external virtual work \( \mathcal{L}^{\text{ext}} \) as follows:

\[
\mathcal{L}^{\text{int}} = \int_0^l [w, u_x] \, dx \quad , \quad \mathcal{L}^{\text{ext}} = \int_0^l [wf] \, dx + w(l)\bar{h}
\]

with \( w(x) \) compatible virtual field

- Principle of virtual work states that:

\[
\text{equilibrium} \quad \Leftrightarrow \quad \mathcal{L}^{\text{int}} - \mathcal{L}^{\text{ext}} = 0 \quad \text{for all} \quad w(x)
\]
**Static case: toward a weak/integral form**

**Strong form**

[differential form]

Given \( b, \bar{u} \) and \( \bar{t} \), find \( u \) such that:

\[
\begin{cases}
\text{div } \sigma + b = 0 & \text{in } \Omega \\
u = \bar{u} & \text{on } \partial\Omega_u \\
\sigma n = \bar{t} & \text{on } \partial\Omega_t
\end{cases}
\]

**Toward a weak formulation**

[integral form]

- Multiply differential equation by a generic “weight” function \( w \)
- Integrate product over problem domain
- Observe that integral is equal to zero
- Integrate by parts
Static case: weak form I

- Multiply differential equation by a weight function $w$
  \[ w \cdot (\text{div} \, \sigma + b) = 0 \]

- Integrate over domain and use integral additive property
  \[ \int_{\Omega} [w \cdot (\text{div} \, \sigma + b)] \, d\Omega = 0 \]
  \[ \int_{\Omega} (w \cdot \text{div} \, \sigma) \, d\Omega + \int_{\Omega} (w \cdot b) \, d\Omega = 0 \]

- Use integration by parts and changing sign
  \[ \int_{\Omega} (\nabla w : \sigma) \, d\Omega - \int_{\partial\Omega} (w \cdot \sigma n) \, d\Omega - \int_{\Omega} (w \cdot b) \, d\Omega = 0 \]

- Introducing boundary conditions and requiring $w = 0$ on $\partial\Omega_u$
  \[ \int_{\Omega} (\nabla w : \sigma) \, d\Omega - \int_{\partial\Omega_t} (w \cdot \bar{t}) \, d\Omega - \int_{\Omega} (w \cdot b) \, d\Omega = 0 \]
Static case: weak form II

- **Weak formulation**  

[**integral form**]

Given \( \mathbf{b}, \bar{\mathbf{u}}, \bar{\mathbf{t}} \), find \( \mathbf{u} \) such that \( \forall \mathbf{w} \):

\[
\int_{\Omega} [\nabla \mathbf{w} : \sigma(\mathbf{u})] \, d\Omega - \int_{\partial\Omega_t} \mathbf{w} \cdot \bar{\mathbf{t}} \, d\Omega - \int_{\Omega} \mathbf{w} \cdot \mathbf{b} \, d\Omega = 0
\]

with

\( \mathbf{u} = \bar{\mathbf{u}} \) and \( \mathbf{w} = 0 \) on \( \partial\Omega_u \)

- Boundary condition \( \sigma \mathbf{n} = \bar{\mathbf{t}} \) no more required !!
- Easy to obtain weak form from any differential problem

**Mechanical interpretation:** principle of virtual work

\[
\mathcal{L}^{\text{int}} - \mathcal{L}^{\text{ext}} = 0 \quad \text{with} \quad \begin{cases} 
\mathcal{L}^{\text{int}} = \int_{\Omega} [\nabla \mathbf{w} : \sigma(\mathbf{u})] \, d\Omega \\
\mathcal{L}^{\text{ext}} = \int_{\partial\Omega_t} \mathbf{w} \cdot \bar{\mathbf{t}} \, d\Omega - \int_{\Omega} \mathbf{w} \cdot \mathbf{b} \, d\Omega
\end{cases}
\]

with \( \mathbf{w} \) compatible virtual field
Static case: weak form I

- On $\partial \Omega_u$ we have $u = \bar{u} \implies w = 0$
  that is: since on $\partial \Omega_u$ the unknown function $u(x)$ has a known value, there is no need to consider a non null variation
  hence: on $\partial \Omega_u$ the weight function can be zero $\implies w(x) = 0$

- On $\partial \Omega_t$ we have $\sigma n = \bar{t} \implies w \neq 0$
  that is: since on $\partial \Omega_t$ the unknown function $u(x)$ has a unknown value, there is the need to consider a non null variation
  hence: on $\partial \Omega_t$ the weight function cannot be zero $\implies w(x) \neq 0$

- Similarly: for any $x$ the unknown function $u(x)$ has an unknown value, there is the need to consider a non null variation
  hence: for any $x$ the variation should be non zero $\implies w(x) \neq 0$
Elasticity: weak form I

**STEP 1:** construct weak form, or start from “Principle of virtual work”

\[
\int_{\Omega} \left\{ \varepsilon(w)^T \sigma(u) \right\} dV - \int_{\Omega} [w^T b] dV - \int_{\partial \Omega_t} w^T \bar{t} dA = 0
\]

which should be verified for any displacement variation \( w \)

**Remarks:**
- Matrix (engineering) notation more appropriate for computer implementation
- \( w \) plays role of weight function!!
- \( \varepsilon(u) = \nabla u \), \( \varepsilon(w) = \nabla w \)
- \( \sigma = \sigma(u) \) in general, \( \sigma = D \varepsilon(u) \) for linear elasticity
- \( \Omega \) can be either a 3D or a 2D domain
  - \( \Omega \subset \mathbb{R}^3 \rightarrow u: 3 \text{ component vector} \)
  - \( \varepsilon: 6 \text{ component vector} \)
  - \( D: 6 \times 6 \text{ matrix} \)
  - \( \Omega \subset \mathbb{R}^2 \rightarrow u: 2 \text{ component vector} \)
  - \( \varepsilon: 3 \text{ component vector} \)
  - \( D: 3 \times 3 \text{ matrix} \)
Elasticity: weak form II

- Weak form 2 can be rewritten using **formal integro-differential operators**

\[
a(u, w) - (b, w) = 0
\]

with

\[
\begin{align*}
a(u, w) &= \int_{\Omega} \left[ \varepsilon(w)^T \sigma(u) \right] dV \\
(b, w) &= \int_{\Omega} \left[ w^T b \right] dV + \int_{\partial\Omega_t} w^T \bar{t} dA
\end{align*}
\]

- Weak form 2 can be also extended to a linear **thermo-elastic material**

\[
\sigma = D \left[ \varepsilon(u) - \alpha (T - T_0) 1 \right]
\]

with \(\alpha\) thermo-elastic coefficient, \(T_0\) reference stress-free temperature, \(1\) the isotropic identity matrix (second-order tensor)

- Weak form 2 can be also extended to **non-linear (elastic or inelastic) materials**
FEM for elasticity 1

**STEP 2:** introduce field approximation based on domain discretization (mesh) with elements of finite dimensions (finite elements)

- $ndm$: problem space dimension
- $nnp$: number of mesh nodes (nodal points) $[nnp = numnp]$  
- $nel$: number of mesh elements $[nel = numel]$  
- $ndf$: maximum number of unknowns per node  
- $nen$: maximum number of nodes per element  
- $neq$: number of active equations

**Standard situations:**

- → For a three-dimensional elastic problem: $ndm = ndf = 3$
- → For a two-dimensional elastic problem: $ndm = ndf = 2$
- → For a two-dimensional thermo-elastic problem: $ndm = 2$, $ndf = 3$
FEM for elasticity: discretization I

- At continuous level deal with a space of infinite dimension

\[ u(x) \in S \]

- \( S \) space of infinite dimension

- Simpler to search solution in a space of finite dimension

\[ u(x) \approx N_1(x)\hat{u}_1 + \ldots + N_4(x)\hat{u}_4 \in S^h \]

- \( S^h \) space of finite dimension

- \( \mathbb{R}^3 \) is an example of space of finite dimension
FEM for elasticity I

- Introduce vector field discretization, i.e. displacements and displacement variations

\[ u(x) = \sum_{j=1}^{nnp} N_j(x) \hat{u}_j = N(x) \hat{u} \]

\[ w(x) = \sum_{i=1}^{nnp} N_i(x) \hat{w}_i = N(x) \hat{w} \]

where:
- \( \hat{u}_j \): nodal **unknown** parameter vectors \((ndf \times 1)\)
- \( \hat{w}_i \): nodal **arbitrary** parameter vectors \((ndf \times 1)\)
- \( N_j(x) \): shape function matrices \((ndf \times ndf)\)
- \( \hat{u} \) and \( \hat{w} \): collection of nodal parameters \((neq \times 1)\)
- \( N(x) \): collection of shape functions \((ndf \times neq)\)

- Accordingly, associated strain fields

\[
\begin{align*}
\varepsilon(u) &= \nabla u = \sum_{j=1}^{nnp} \left( \nabla N_j \right) \hat{u}_j = \sum_{j=1}^{nnp} B_j \hat{u}_j = B \hat{u} \\
\varepsilon(w) &= \nabla w = \sum_{i=1}^{nnp} \left( \nabla N_i \right) \hat{w}_i = \sum_{i=1}^{nnp} B_i \hat{w}_i = B \hat{w}
\end{align*}
\]

where:
- \( B_j \): shape function derivative matrices \((6 \times ndf \text{ in 3D, } 3 \times ndf \text{ in 2D})\)
- \( B \): collection of shape function derivative \((6 \times neq \text{ in 3D, } 3 \times neq \text{ in 2D})\)
For 3D problems ⇒ displacement vector has 3 components:

\[ u = \{u, v, w\}^T \]

Since displacement components \( u, v, w \) are “in general” seen as independent fields, general interpolation position looks like

\[
\begin{bmatrix}
  u \\
  v \\
  w
\end{bmatrix} = \begin{bmatrix}
  \sum_j N_j \hat{u}_j \\
  \sum_j N_j \hat{v}_j \\
  \sum_j N_j \hat{w}_j
\end{bmatrix} = \sum_j \begin{bmatrix}
  N_j & 0 & 0 \\
  0 & N_j & 0 \\
  0 & 0 & N_j
\end{bmatrix} \begin{bmatrix}
  \hat{u}_j \\
  \hat{v}_j \\
  \hat{w}_j
\end{bmatrix}
\]

Recalling

\[ u = \sum_{j=1}^{nnp} N_j \hat{u}_j \]

we conclude

\[ N_j = \begin{bmatrix}
  N_j & 0 & 0 \\
  0 & N_j & 0 \\
  0 & 0 & N_j
\end{bmatrix} \]
FEM for elasticity: a possible shape function choice

- For 3D problems ⇒ strain vector has 6 components:

\[ \boldsymbol{\epsilon} = \{\epsilon_{xx}, \epsilon_{yy}, \epsilon_{zz}, \gamma_{xy}, \gamma_{yz}, \gamma_{xz}\}^T \]

Accordingly:

\[ \epsilon_{xx} = \frac{\partial u}{\partial x} = \sum_j \frac{\partial N_j}{\partial x} \hat{u}_j \quad , \quad \epsilon_{yy} = \frac{\partial v}{\partial y} = \sum_j \frac{\partial N_j}{\partial x} \hat{v}_j \quad , \quad .... \]

\[ \gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = \sum_j \frac{\partial N_j}{\partial y} \hat{u}_j + \sum_j \frac{\partial N_j}{\partial x} \hat{v}_j \quad , \quad .... \]

Hence:

\[ \boldsymbol{\epsilon} = \sum_j \boldsymbol{B}_j \hat{u}_j \quad \text{with} \]

\[ \boldsymbol{B}_j = \begin{bmatrix} N_{j,x} & 0 & 0 \\ 0 & N_{j,y} & 0 \\ 0 & 0 & N_{j,z} \end{bmatrix} \]

- Exercise. Compute \( \boldsymbol{N}_i \) and \( \boldsymbol{B}_i \) for 2D elasticity.
- Exercise. Compute \( \mathbb{D} \) for 3D elasticity, 2D plane strain elasticity, 2D plane stress elasticity.
Substitute compact approximation of weight functions in weak form

\[
\int_{\Omega} \left[ (B\hat{w})^T \sigma(u) \right] dV - \int_{\Omega} \left[ (N\hat{w})^T b \right] dV - \int_{\partial\Omega_t} (N\hat{w})^T \bar{t} dA = 0
\]

to be verified for any displacement variation \( w \) i.e. for any set of parameters \( \hat{w} \)

Accordingly, we have:

\[
\hat{w}^T \left\{ \int_{\Omega} \left[ B^T \sigma(u) \right] dV - \int_{\Omega} \left[ N^T b \right] dV - \int_{\partial\Omega_t} N^T \bar{t} dA \right\} = 0
\]

Due to arbitrariness of \( \hat{w} \), go from scalar to vector equation

\[
R = \int_{\Omega} \left[ B^T \sigma(u) \right] dV - \int_{\Omega} \left[ N^T b \right] dV - \int_{\partial\Omega_t} N^T \bar{t} dA = 0
\]

Equation in residual form!!

For a linear elastic material \( \sigma = D\varepsilon = DB\hat{u} \)

\[
R = \int_{\Omega} \left[ B^T D B\hat{u} \right] dV - \int_{\Omega} \left[ N^T b \right] dV - \int_{\partial\Omega_t} N^T \bar{t} dA = 0
\]
FEM for linear elasticity I

- For a linear elastic material (no surface forces !!) write previous equation in matrix form

\[ R = K \hat{u} - f = 0 \quad \text{or} \quad K \hat{u} = f \]

where

\[ K = \int_{\Omega} \left[ B^T D B \right] dV \quad \text{global stiffness matrix} \]
\[ f = \int_{\Omega} \left[ N^T b \right] dV \quad \text{global load vector} \]

- Using integral additive property, switch to an element point of view:

\[
\begin{cases}
K = \int_{\Omega} \left[ B^T D B \right] dV = \sum_{e=1}^{nel} \int_{\Omega^e} \left[ B^T D B \right] dV = A_{e=1}^{nel} K^e \\
f = \int_{\Omega} \left[ N^T b \right] dV = \sum_{e=1}^{nel} \int_{\Omega^e} \left[ N^T b \right] dV = A_{e=1}^{nel} f^e
\end{cases}
\]

where

\[
\begin{cases}
K^e = \int_{\Omega^e} \left[ B^T D B \right] dV \quad \text{element stiffness matrix} \\
f^e = \int_{\Omega^e} \left[ N^T b \right] dV \quad \text{element load vector}
\end{cases}
\]

★ Assembly operator \( A \)
Stiffness construction I

- **Example:** 2D elastic problem

  - $ndm = 2$ space dimension of mesh
  - $nel$ number of mesh elements
  - $ndf = 2$ maximum number of unknowns per node
  - $nen$ maximum number of nodes per element
  - $neq$ number of active equations

- $K^e$ has dimension $nst \times nst$  
  [ normally $nst = ndf \times nen$ ]

- $K$ has dimension $neq \times neq$

  ```
  for $e = 1$ to $nel$
  construct $K^e$
  assemble $K^e$ into $K$
  end if
  ```

- **TRI**
  - $nen = 3$
  - $nst = 6$

- **QUAD**
  - $nen = 4$
  - $nst = 8$
FEM for elasticity: second approach I

- We may also use other interpolation expression (through nodal quantities)
  - Substitute approximation in weak form

\[
\int_{\Omega} \left[ \left( \sum_{i}^{nnp} B_i \hat{w}_i \right)^T \sigma(u) \right] dV - \int_{\Omega} \left[ \left( \sum_{i}^{nnp} N_i \hat{w}_i \right)^T b \right] dV - \int_{\partial\Omega_t} \left( \sum_{i}^{nnp} N_i \hat{w}_i \right)^T \tilde{t} dA = 0
\]

to be verified for any displacement variation \( w \) i.e. for any set of parameters \( \hat{w}_i \)

- Due to arbitrariness of \( \hat{w}_i \) we conclude that:

\[
R_i = \int_{\Omega} \left[ B_i^T \sigma(u) \right] dV - \int_{\Omega} \left[ N_i^T b \right] dV - \int_{\partial\Omega_t} N_i^T \tilde{t} dA = 0
\]

- Obtain \( nnp \) vector equations, each one composed by \( ndf \) scalar equations
  - For a linear elastic material \( \sigma = D \varepsilon(u) = D \left( \sum_{i}^{nnp} B_j \hat{u}_j \right) = D (B_j \hat{u}_j) \) (i.e., adopting summation convention)

\[
R_i = \int_{\Omega} \left[ B_i^T D B_j \hat{u}_j \right] dV - \int_{\Omega} \left[ N_i^T b \right] dV - \int_{\partial\Omega_t} N_i^T \tilde{t} dA = 0
\]
For a linear elastic material – neglecting also surface force term – write previous equation in **matrix form**

\[ R_i = K_{ij} \hat{u}_j - f_i = 0 \quad \text{or} \quad K_{ij} \hat{u}_j = f_i \]

where

\[ K_{ij} = \int_{\Omega} \left[ B_i^T \mathbb{D} B_j \right] dV \quad \text{node-to-node stiffness matrix} \]

\[ f_i = \int_{\Omega} \left[ N_i^T b \right] dV \quad \text{node-to-node load vector} \]

Using integral additive property, switch to an **element point of view**:

\[
\begin{cases}
K_{ij} = \int_{\Omega} \left[ B_i^T \mathbb{D} B_j \right] dV = \sum_{e=1}^{\text{nel}} \int_{\Omega_e} \left[ B_i^T \mathbb{D} B_j \right] dV = \sum_{e=1}^{\text{nel}} K_{ij}^e \\
\end{cases}
\]

\[
\begin{cases}
f_i = \int_{\Omega} \left[ N_i^T b \right] dV = \sum_{e=1}^{\text{nel}} \int_{\Omega_e} \left[ N_i^T b \right] dV = \sum_{e=1}^{\text{nel}} f_i^e \\
\end{cases}
\]

where

\[
\begin{cases}
K_{ij}^e = \int_{\Omega_e} \left[ B_i^T \mathbb{D} B_j \right] dV \quad \text{node-to-node element stiffness matrix} \\
f_i^e = \int_{\Omega_e} \left[ N_i^T b \right] dV \quad \text{node-to-node element load vector} \\
\end{cases}
\]
Stiffness construction: second approach

**Example:** 2D elastic problem

- \( ndm = 2 \) \quad \text{space dimension of mesh}
- \( nel \) \quad \text{number of mesh elements}
- \( ndf = 2 \) \quad \text{maximum number of unknowns per node}
- \( nen \) \quad \text{maximum number of nodes per element}
- \( neq \) \quad \text{number of active equations}

\[
\text{for } e = 1 \text{ to } nel \\
\quad \text{construct } K^e \\
\quad \text{for } i = 1 \text{ to } nen \\
\quad \quad \text{for } j = 1 \text{ to } nen \\
\quad \quad \quad \text{construct } K_{ij}^e \\
\quad \quad \text{end if} \\
\quad \text{end if} \\
\quad \text{assemble } K_{ij}^e \text{ into } K^e \\
\quad \text{assemble } K^e \text{ into } K \\
\text{end if}
\]

- \( K \) \quad \text{has dimension } neq \times neq
- \( K^e \) \quad \text{has dimension } nst \times nst \quad \text{(normally } nst = ndf \times nen)\)
- \( K_{ij}^e \) \quad \text{has dimension } ndf \times ndf
Choose shape functions such that $K^e$ and $f^e$ are easy to compute

- Local shape functions
- Hat shape functions

**Piecewise linear shape functions**

\[ N_i(x_i) = 1 \quad , \quad N_i(x_j) = 0 \quad \text{with} \quad j \neq i \]
TRI-3: linear triangular element

- Using three nodes, very simple to construct linear (complete) interpolating functions
- Construct interpolating functions element-by-element!!
- Need to introduce area coordinates $L_1, L_2, L_3$
2D TRI-3 element II

- **Area coordinates** constructed such that:

\[
\begin{align*}
  x &= L_1 x_1 + L_2 x_2 + L_3 x_3 \\
  y &= L_1 y_1 + L_2 y_2 + L_3 y_3 \\
  1 &= L_1 + L_2 + L_3
\end{align*}
\]  

(3)

- Functions $L_1, L_2, L_3$ are s.t.

\[L_i(x_i) = 1, \quad L_i(x_j) = 0 \text{ with } i \neq j\]

- Very simple to construct $L_1$:

\[
L_1(x) = \frac{\text{Area triangle (1-2-3)}}{\text{Area triangle (1-2-3)}}
\]

which is also the reason for calling them "area coordinates"

- Other functions obtain with cyclic rotation of indices !!
Alternatively, we can compute $L_1, L_2, L_3$ solving equation 3, obtaining

$$L_1 = \frac{a_1 + b_1 x + c_1 y}{2\Delta}$$

where

$$\Delta = \frac{1}{2} \det \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix} = \text{Area triangle (1-2-3)}$$

and

$$\begin{cases} a_1 = x_2 y_3 - x_3 y_2 \\ b_1 = y_2 - y_3 \\ c_1 = x_3 - x_2 \end{cases}$$

Other functions obtain with cyclic rotation of indices !!

Interesting aspect: easy to compute close form integrals

$$\int_\Delta L_1^a L_2^b L_3^c dxdy = 2\Delta \frac{a! b! c!}{(a + b + c + 2)!}$$
2D TRI-3 element IV

- For a 3-node triangle:

  \[ L_i = \frac{a_i + b_i x + c_i y}{2\Delta} \]

  hence

  \[ \frac{\partial}{\partial x} L_i = \frac{b_i}{2\Delta}, \quad \frac{\partial}{\partial y} L_i = \frac{c_i}{2\Delta} \]

- Therefore:

  \[ B_i = \frac{1}{2\Delta} \begin{bmatrix} b_i & 0 \\ 0 & c_i \\ c_i & b_i \end{bmatrix} \]

- For a homogeneous body

  \[ K_{ij}^e = \int_{\Delta} \left[ B_i^T \mathcal{D} B_j \right] d\Delta = \frac{1}{2\Delta} \left[ B_i^T \mathcal{D} B_j \right] \]

**Exercise.** Study Matlab code and modify to run problems with surface load. Run a simple patch test.

**Exercise.** Solve a convergence problem plotting error curves

**Exercise.** Study a problem with \( \nu = 0.49, 0.4999, 0.499999 \)
2D elastic problems I

Convergence for a known problem

- Look at a problem for which exact solution is known
- Compare numerical solution vs exact solution

$$error = \frac{||u^{num} - u^{ex}||}{||u^{ex}||}$$

- Importance of right norm from a mathematical point of view
- First indication from engineering considerations (displacement in a point!)
- Example of patch tests
Important the form in which you plot the error
Inability to produce some specific situations

Study a problem with $\nu = 0.49, 0.4999, 0.499999$

In the limit obtain a problem with no volumetric changes

Problem comes out from inability of discrete displacement to satisfy a constraint (beam/plate shear locking, shell membrane/bending locking)

Need for other formulations besides displacement-based ones
Different formulations are possible

- **Displacement formulations** (Principle of virtual work)
  \[ R(u) = 0 \]

- **Incompatible formulations**
  \[ R(u) = 0 \quad \text{with} \quad u = u^{\text{comp}} + u^{\text{incomp}} \]

- **Enhanced formulations**
  \[ R(u, \alpha) = 0 \quad \text{with} \quad \varepsilon = Lu + \alpha \]

- **Mixed formulations**
  \[ R(u, \sigma) = 0 \quad \text{Hellinger-Reissner} \]
  \[ R(u, \varepsilon, \sigma) = 0 \quad \text{Hu-Washizu} \]

  with \( u, \varepsilon \) and \( \sigma \) independent approximations (!)

- **Possible many other approaches**: under-integrated, stabilized, etc. etc.
We may think of two different set of interpolations

- Geometric interpolation
- Unknown field interpolation

\[ x = x(\bar{x}_i) \quad , \quad u = u(\hat{u}_i) \]
To construct shape functions, convenient to introduce a **parent element**
- Parent element: fixed known geometry
- Current element: variable geometry (nodal coordinates depend on specific element)

- $\xi$: coordinates for parent element (often called **natural coordinates**)
- $x$: coordinates for current element

\[
\begin{align*}
\xi &= \begin{bmatrix} \xi \\ \eta \end{bmatrix}, & x &= \begin{bmatrix} x \\ y \end{bmatrix}
\end{align*}
\]

- Clearly, we can introduce a map from current to parent element and viceversa

\[
\xi = \xi(x) \quad , \quad x = x(\xi)
\]
2D QUAD isoparametric element I

- Restrict attention to a single element
- Using same interpolating functions for unknown interpolation as well as for \( x = x(\xi) \) geometrical interpolation \( \Rightarrow \textbf{isoparametric element} \)

\[
\begin{align*}
\mathbf{u} &= \left\{ \begin{array}{c} u \\ v \end{array} \right\}, \\
\mathbf{v} &= \left\{ \begin{array}{c} \mathbf{u} \\ \mathbf{v} \end{array} \right\}, \\
\mathbf{x} &= \left\{ \begin{array}{c} x \\ y \end{array} \right\}, \\
\mathbf{y} &= \left\{ \begin{array}{c} \bar{x}_i \\ \bar{y}_i \end{array} \right\}, \\
\end{align*}
\]

where \( \bar{x}_i \) and \( \bar{y}_i \) are the \( x \)- and \( y \)-coordinates of the element nodes
2D QUAD isoparametric element I

- At this stage $N_i$ are functions defined on parent element

$$N_i = N_i(\xi, \eta) = N_i(\xi)$$

- Once defined on parent element, $N_i$ functions can be mapped to current element if or whenever necessary:

$$\xi = \xi(x) \Rightarrow N_i = N_i(\xi) = N_i(\xi(x)) = N_i(x)$$

- Easier to define $N_i$ on parent element since parent element has a fixed geometry

- Possible choice:

$$N_i = \frac{1}{4} \left(1 + \bar{\xi}_i \xi \right) \left(1 + \bar{\eta}_i \eta \right)$$

with $\bar{\xi}_i$ and $\bar{\eta}_i$ nodal parent coordinates, i.e.:

$$\begin{cases} 
\bar{\xi}_1 = -1, & \bar{\xi}_2 = 1, & \bar{\xi}_3 = 1, & \bar{\xi}_4 = -1 \\
\bar{\eta}_1 = -1, & \bar{\eta}_2 = -1, & \bar{\eta}_3 = 1, & \bar{\eta}_4 = 1
\end{cases}$$
Explicit expression for shape functions

\[
\begin{align*}
N_1 &= \frac{1}{4} (1 - \xi)(1 - \eta) \\
N_2 &= \frac{1}{4} (1 + \xi)(1 - \eta) \\
N_3 &= \frac{1}{4} (1 + \xi)(1 + \eta) \\
N_4 &= \frac{1}{4} (1 - \xi)(1 + \eta)
\end{align*}
\]

Note that:

\[N_i(\xi_i) = 1 \quad , \quad N_i(\xi_j) = 0\]

Recall: a shape function is relative to a node in global setting and it is not defined only on a single element.

Consider different element patches.
2D QUAD isoparametric element 1

- Shape functions depend on the natural coordinates $\xi$
  \[ N_i(\xi) \]

- Shape function derivatives computed wrt current coordinates $x$
  \[ \frac{\partial N_i}{\partial x} \]

- Required a change of coordinates (chain rule!!)
  \[ u = u(\xi) = u(\xi(x)) \]

\[
\begin{align*}
  u_x &= \frac{\partial u}{\partial x} = \frac{\partial u}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial u}{\partial \eta} \frac{\partial \eta}{\partial x} \\
  u_y &= \frac{\partial u}{\partial y} = \frac{\partial u}{\partial \xi} \frac{\partial \xi}{\partial y} + \frac{\partial u}{\partial \eta} \frac{\partial \eta}{\partial y}
\end{align*}
\]

\[
\begin{bmatrix}
  u_x \\
  u_y
\end{bmatrix} =
\begin{bmatrix}
  \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\
  \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y}
\end{bmatrix}
\begin{bmatrix}
  \frac{\partial u}{\partial \xi} \\
  \frac{\partial u}{\partial \eta}
\end{bmatrix}
\]
2D QUAD isoparametric element I

- Derivatives wrt natural coordinates are easy to compute

\[
\begin{align*}
\frac{\partial u}{\partial \xi} &= \frac{\partial}{\partial \xi} \left( \sum_{i=1}^{4} N_i \hat{u}_i \right) = \sum_{i=1}^{4} \left( \frac{\partial}{\partial \xi} N_i \right) \hat{u}_i = \sum_{i=1}^{4} N_{i,\xi} \hat{u}_i \\
\frac{\partial u}{\partial \eta} &= \frac{\partial}{\partial \eta} \left( \sum_{i=1}^{4} N_i \hat{u}_i \right) = \sum_{i=1}^{4} \left( \frac{\partial}{\partial \eta} N_i \right) \hat{u}_i = \sum_{i=1}^{4} N_{i,\eta} \hat{u}_i
\end{align*}
\]

- Accordingly:

\[
\begin{align*}
N_i &= \frac{1}{4} (1 + \bar{\xi}_i \xi) (1 + \bar{\eta}_i \eta) \\
N_{i,\xi} &= \frac{1}{4} \bar{\xi}_i (1 + \bar{\eta}_i \eta) \\
N_{i,\eta} &= \frac{1}{4} \bar{\eta}_i (1 + \bar{\xi}_i \xi)
\end{align*}
\]
2D QUAD isoparametric element I

- Slightly more complex to compute:

\[
F = \begin{bmatrix}
\frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\
\frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y}
\end{bmatrix}
\]

This is the transpose of the jacobian relative to the coordinate change

\[
\xi = \xi(x)
\]

- Since we know the inverse relation

\[
x = x(\xi)
\]

we can easily compute:

\[
G = \begin{bmatrix}
\frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\
\frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta}
\end{bmatrix}
\]

- Then:

\[
F = G^{-T}
\]
2D isoparametric element I

Feap already provides you with all the subroutine for implementing a 2D isoparametric element. In particular:

- **subroutine shp2D**
  - it computes shape functions and derivatives for quadrilateral elements
  - up to 9 node elements
  - possible to compute derivatives wrt to natural or to current coordinates
  - compute also jacobian determinant
  - \( shp2d(ss, xl, shp, xsj, ndm, nel, ix, flg) \)

- **subroutine elmt01**
  - dummy subroutine for the implementation of a user element
  - \( elmt01 (d, ul, xl, ix, tl, s, p, ndf, ndm, nst, isw) \)
subroutine shp2d(ss, xl, shp, xsj, ndm, nel, ix, flg)

% Purpose: Computes shape function and derivatives for quadrilateral elements

% Inputs:
% ss(2) - Natural coordinates for point
% xl(ndm,*) - Nodal coordinates for element
% ndm - Spatial dimension of mesh
% nel - Number of nodes on element
% ix(*) - Nodes attached to element
% flg - Flag, compute global x/y derivatives if false, else derivatives are w/r natural coords.

% Outputs:
% shp(3,*) - Shape functions and derivatives at point
% shp(1,i) = dN_i/dx or dN_i/dxi_1
% shp(2,i) = dN_i/dy or dN_i/dxi_2
% shp(3,i) = N_i
% xsj - Jacobian determinant at point
ELASTIC 2D QUADRILATERAL ELEMENT: DISPLACEMENT FORMULATION

Parameter passed:

- \( d(*) \) = array storing material parameter
- \( u_l(n_{dof}, n_{en}) \) = displacement, \( i: \) dof (u,v), \( j: \) node # (1->4)
- \( x_l(n_{dim}, n_{en}) \) = nodal coordinates, \( i: (x,y)-\) coordinate, \( j: \) node # (1->4)
- \( i_x(*) \) = global/local node number (0 free - 1 fixed)
- \( t_l(*) \) = nodal temperature array
- \( s(n_{st}, n_{st}) \) = stiffness matrix
- \( p(n_{st}) \) = residual
- \( n_{dof} \) = max. no. of dof at any nodes = 2
- \( n_{dim} \) = spatial dimension of the problem = 2
- \( n_{st} \) = \( n_{dof} \times n_{en} \) = 2 x 4 = 8
- \( i_{sw} \) = action to perform

- \( n_{el} \) = max. no. of nodes on current element = 4
- \( n_{en} \) = max. no. of nodes connected to any element = 4
2D numerical integration I

- General term in the element stiffness:
  \[ \int_{\Omega^e} f(x) dA \]

- Transform integration over \( \Omega^e \) to integration over \( \Box \), with \( \Box \) the bi-unit parent element
  \[ \int_{\Omega^e} f(x) dA = \int_{\Box} f[x(\xi)] J d\Box \]

  where:
  - the function expression does not change, i.e.:
    \[ f(x) = f[x(\xi)] \]
  - \( J \) is the determinant of the jacobian matrix corresponding to the map \( x = x(\xi) \), i.e.:
    \[ J = \det \left( \frac{\partial x}{\partial \xi} \right) = \det(G) \]

- For a 2D problem
  \[ J = \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{vmatrix} \]
2D numerical integration I

- General term in the element stiffness

\[ \int g(\xi) d\xi \]

- Function \( g \) is usually quite complex and non-linear (i.e., it includes also jacobian determinant)
- Possible and convenient to compute integrals in close form only for specific cases (linear triangles)
- Non practical or not possible to compute it in close form in the general case

**Adopt numerical integration (quadrature formulas)**

\[ \int g(\xi) d\xi \approx \sum_{l=1}^{ng} g(\tilde{\xi}_l) w_l \]

where:
- \( \tilde{\xi}_l \): well-defined points where function should be evaluated
- \( w_l \): weights

- Integration over parent element transformed into a combination – through the weights \( w_l \) – of the function evaluated at specific point
2D numerical integration. Gauss-Legendre I

- Many quadrature formulas exist
- Gauss-Legendre quadrature are the most accurate for polynomials
- Gauss quadrature are generally tabulated over the range of coordinates $-1 < \xi < 1$ (hence our main reason for also choosing many shape functions on this interval)
- In 1D Gaussian quadrature integrates a function with the following approximation:

$$\int_{-1}^{1} f(\xi) d\xi = \sum_{j=1}^{n} f(\xi_j) w_j + O\left(\frac{d^{2n}f}{d\xi^{2n}}\right)$$

with $j$ the points where the function is evaluated and $w_j$ the weights
- A $n$-point Gauss formula integrates exactly a polynomial of order $2n - 1$
- Integrations over multi-dimensional problems may be performed by products of the one-dimensional formula
- As an example in 2D

$$\int_{-1}^{1} \int_{-1}^{1} f(\xi, \eta) d\xi d\eta \approx \sum_{j=1}^{n} \sum_{k=1}^{n} f(\xi_j, \eta_k) w_j w_k$$
A simple MATLAB FE code: construction of local stiffness matrix

**Shape functions**

\[
\{N\}_i = N_i, \quad \{N\}_{1\times 4} = \begin{bmatrix} N_1 & N_2 & N_3 & N_4 \end{bmatrix}
\]

**Shape function derivatives**

\[
\begin{bmatrix} \frac{\partial N}{\partial \xi} \end{bmatrix}_{ij} = \frac{\partial N_j}{\partial \xi_i}, \quad \begin{bmatrix} \frac{\partial N}{\partial \xi} \end{bmatrix}_{2\times 4} = \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} & \frac{\partial N_3}{\partial \xi} & \frac{\partial N_4}{\partial \xi} \\ \frac{\partial N_1}{\partial \eta} & \frac{\partial N_2}{\partial \eta} & \frac{\partial N_3}{\partial \eta} & \frac{\partial N_4}{\partial \eta} \end{bmatrix}
\]

\[
\begin{bmatrix} \frac{\partial N}{\partial x} \end{bmatrix}_{ij} = \frac{\partial N_j}{\partial x_i}, \quad \begin{bmatrix} \frac{\partial N}{\partial x} \end{bmatrix}_{2\times 4} = \begin{bmatrix} \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial x} & \frac{\partial N_4}{\partial x} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_2}{\partial y} & \frac{\partial N_3}{\partial y} & \frac{\partial N_4}{\partial y} \end{bmatrix}
\]

**Shape-function and shape-function derivatives wrt natural coordinates** (easy !!)

\[
\begin{cases}
N_i = \frac{1}{4} \left(1 + \bar{\xi}_i \xi \right) \left(1 + \bar{\eta}_i \eta \right) \\
N_i,\xi = \frac{1}{4} \bar{\xi}_i \left(1 + \bar{\eta}_i \eta \right) \\
N_i,\eta = \frac{1}{4} \bar{\eta}_i \left(1 + \bar{\xi}_i \xi \right)
\end{cases}
\]

with

\[
\bar{\xi} = \{-1, 1, 1, -1\}, \quad \bar{\eta} = \{-1, -1, 1, 1\}
\]
A simple MATLAB FE code: construction of local stiffness matrix II

Shape-function chain-rule

\[
\begin{align*}
\frac{\partial N_i}{\partial x} &= \frac{\partial N_i}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial N_i}{\partial \eta} \frac{\partial \eta}{\partial x} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \end{bmatrix} \begin{bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{bmatrix} \\
\left\{ \begin{array}{c} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial \eta} \end{array} \right\} &= \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ \frac{\partial \xi}{\partial \eta} & \frac{\partial \eta}{\partial \eta} \end{bmatrix} \begin{bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{bmatrix} \\
\begin{bmatrix} \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial x} & \frac{\partial N_4}{\partial x} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_2}{\partial y} & \frac{\partial N_3}{\partial y} & \frac{\partial N_4}{\partial y} \end{bmatrix} &= \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ \frac{\partial \xi}{\partial \eta} & \frac{\partial \eta}{\partial \eta} \end{bmatrix} \begin{bmatrix} \frac{\partial N_1}{\partial \xi} & \frac{\partial N_2}{\partial \xi} & \frac{\partial N_3}{\partial \xi} & \frac{\partial N_4}{\partial \xi} \\ \frac{\partial N_1}{\partial \eta} & \frac{\partial N_2}{\partial \eta} & \frac{\partial N_3}{\partial \eta} & \frac{\partial N_4}{\partial \eta} \end{bmatrix} \\
\begin{bmatrix} \frac{\partial \mathbf{N}}{\partial x} \end{bmatrix}_{2 \times 4} &= \begin{bmatrix} \frac{\partial \xi}{\partial x} \end{bmatrix}_{2 \times 2}^T \begin{bmatrix} \frac{\partial \mathbf{N}}{\partial \xi} \end{bmatrix}_{2 \times 4}
\end{align*}
\]
A simple MATLAB FE code: construction of local stiffness matrix III

- **Jacobian transformation**
  Need to compute

\[
\begin{bmatrix}
\frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\
\frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y}
\end{bmatrix} = \left[ \begin{array}{c}
\frac{\partial \xi}{\partial x} \\
\frac{\partial \eta}{\partial x}
\end{array} \right]^T = \left( \left[ \begin{array}{c}
\frac{\partial x}{\partial \xi} \\
\frac{\partial y}{\partial \xi}
\end{array} \right]^T \right)^{-1}
\]

First compute

\[
\begin{bmatrix}
\frac{\partial x}{\partial \xi} \\
\frac{\partial y}{\partial \xi}
\end{bmatrix}^T = \begin{bmatrix}
\frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\
\frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta}
\end{bmatrix}
\]

- **Geometric map**

\[
x = x(\xi) \quad \Rightarrow \quad x = \sum_{i=1}^{4} N_i \hat{x}_i, \quad y = \sum_{i=1}^{4} N_i \hat{y}_i
\]
A simple MATLAB FE code: construction of local stiffness matrix IV

\[
\begin{bmatrix}
\frac{\partial x}{\partial \xi} \\
\frac{\partial y}{\partial \eta}
\end{bmatrix}
= \sum_{i=1}^{4} \frac{\partial N_i}{\partial \xi} \hat{x}_i
\]

\[
\begin{bmatrix}
\frac{\partial x}{\partial \xi} \\
\frac{\partial y}{\partial \eta}
\end{bmatrix}
= \sum_{i=1}^{4} \frac{\partial N_i}{\partial \eta} \hat{x}_i
\]

\[
\text{jac} = \det \left( \frac{\partial x}{\partial \xi} \right) = \det (G)
\]
A simple MATLAB FE code: construction of local stiffness matrix V

- **Final matrices** for load and stiffness construction

\[ N = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 \end{bmatrix} \]

\[ B = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_3}{\partial x} & 0 & \frac{\partial N_4}{\partial x} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & 0 & \frac{\partial N_3}{\partial y} & 0 & \frac{\partial N_4}{\partial y} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial y} & \frac{\partial N_3}{\partial x} & \frac{\partial N_4}{\partial y} & \frac{\partial N_4}{\partial x} \end{bmatrix} \]
A simple MATLAB FE code: the code!

Finite element driver for 4-node plane strain elastic problems

%==========================================================================
%
% CODE: elast
%
% Solve quite-general 2D elastic problems reading input data from files
%
%==========================================================================

%% xxx_nodes.dat
% node1,ngen,x1,y1
% node2,ngen,x2,y2

%% xxx_elements.dat
% nelm1,ngen,matn,node1,node2,node3,node4
% nelm2,ngen,matn,node1,node2,node3,node4

%% xxx_loads.dat
% node1,ngen,loadx1,loady1
% node2,ngen,loadx2,loady2

%% xxx_bc.dat
% node1,ngen,bcx1,bcy1
% node2,ngen,bcx2,bcy2

%==========================================================================
A simple MATLAB FE code: the code! II

```matlab
clc
clear all

ngdl = 2;  % number of gdl / node
nen = 4;   % Maximum number of nodes on any element
nmat = 1;  % Maximum number of material properties
ngauss = 3

ee = 1000; % Input material parameters
nu = 0.3;
mat(1,1) = ee;
mat(1,2) = nu;

fx = 0;    % Input volume loads [ feature not tested yet !! ]
fy = 0;

% load mesh data and material properties

load square_nodes.dat
load square_elements.dat
load square_loads.dat
load square_bc.dat

nodes = square_nodes;
elements = square_elements;
loads = square_loads;
bc = square_bc;
```
% Compute problem characteristic dimensions

nnod = size(nodes,1);
nel = size(elements,1);
ndof = nnod * ngdl;

% Number dofs for all nodes

dofs = zeros(nnod,2);
number = [1:nnod*2];
dofs = reshape(number,2,nnod)';  \% dofs = [ u1, u2, ... un, v1, v2, ... vn ]

% Construct connectivity matrix

connect = zeros(nel,nen);
connect(:,[1:4]) = elements(:,[4:4+nen-1]);

% Inizialize solution quantities

%k_gl = sparse(ndof,ndof);
k_gl = zeros(ndof,ndof);
f_gl = zeros(ndof,1);
displ = zeros(nnod,3);

% Compute gauss point and weights on parent element
[gp,gw] = HP_gauleg(-1,1,ngauss);

% Construct stiffness matrix
for i = 1:nel                 % loop over elements
    nodes_el = connect(i,1:4);  % extract local nodes
    coor_el  = nodes(nodes_el,3:4);  % extract coord element nodes
    dofs_el  = dofs(nodes_el,1:2);  % extract local dofs
    mat_el   = mat(elements(i,3),:);  % extract material number

    [k_el,f_el] = k_el_q1(mat_el,coor_el,gp,gw,fx,fy); % compute element stiffness
    [k_gl,f_gl] = assembl_el(k_el,k_gl,f_el,f_gl,dofs_el); % assemble into global stiffness
end

f_gl = assembl_load(f_gl,loads,dofs);
[k_gl,f_gl] = reduce_bc(k_gl,f_gl,bc,dofs);

u_gl = k_gl_gl;           % solve the linear system by gaussian elimination
plot_sol(nodes,u_gl)

format long
u_gl(end-1)
exit
A simple MATLAB FE code: the code! V

4-node plane strain elastic element. Stiffness and load construction

%---------------------------------------------------------------------
% 4-node plane strain elastic element. Stiffness and load construction
%
% Input:
% mat_el = material data
% coor_el = element nodal coordinates
% gp = integration pts for Gauss-integration
% gw = weight vector for Gauss-integration
% fx = body force in the x-direction
% fy = body force in the y-direction
%
% Output:
%
% (converted from q1 element by Alessandro Reali)
%---------------------------------------------------------------------

function [k_el,f_el]=k_el_q1(mat_el,coor_el,gp,gw,fx,fy)

E = mat_el(1);
nu = mat_el(2);
mu = E/(2*(1 + nu));

lambda = E*nu/((1 + nu)*(1 - 2*nu));
CC = mu*[2 0 0; 0 2 0; 0 0 1] + lambda*[1 1 0; 1 1 0; 0 0 0];

ngauss = size(gw,2);

E = mat_el(1);
nu = mat_el(2);
mu = E/(2*(1 + nu));

lambda = E*nu/((1 + nu)*(1 - 2*nu));
CC = mu*[2 0 0; 0 2 0; 0 0 1] + lambda*[1 1 0; 1 1 0; 0 0 0];

ngauss = size(gw,2);
A simple MATLAB FE code: the code! VI

AA = zeros(2,8);
BB = zeros(3,8);
k_el = zeros(8,8);
f_el = zeros(8,1);

N = zeros(1,4);
dN_dxi = zeros(2,4);
dN_dx = zeros(2,4);

for ig = 1:ngauss
    for jg = 1:ngauss
        N(1) = (1 - gp(ig))*(1 - gp(jg))/4;  % bilinear shape functions
        N(2) = (1 + gp(ig))*(1 - gp(jg))/4;
        N(3) = (1 + gp(ig))*(1 + gp(jg))/4;
        N(4) = (1 - gp(ig))*(1 + gp(jg))/4;

        dN_dxi(1,1) = -(1 - gp(jg))/4;  % shape-function derivatives
        dN_dxi(1,2) = (1 - gp(jg))/4;  % wrt csi
        dN_dxi(1,3) = (1 + gp(jg))/4;
        dN_dxi(1,4) = -(1 + gp(jg))/4;

        dN_dxi(2,1) = -(1 - gp(ig))/4;  % shape-function derivatives
        dN_dxi(2,2) = -(1 + gp(ig))/4;  % wrt eta
        dN_dxi(2,3) = (1 + gp(ig))/4;
        dN_dxi(2,4) = (1 - gp(ig))/4;
    end
end
% Jacobian matrix inverse [ matrix "G^-T" in the notes ]
    dx_dxiT = zeros(2,2);
    for i = 1:4
        dx_dxiT(1,1) = dx_dxiT(1,1) + coor_el(i,1)*dN_dxi(1,i);
        dx_dxiT(1,2) = dx_dxiT(1,2) + coor_el(i,2)*dN_dxi(1,i);
        dx_dxiT(2,1) = dx_dxiT(2,1) + coor_el(i,1)*dN_dxi(2,i);
        dx_dxiT(2,2) = dx_dxiT(2,2) + coor_el(i,2)*dN_dxi(2,i);
    end
    Jac = det(dx_dxiT);

% Jacobian matrix [ matrix "F" in the notes ]
    dxi_dxT = inv(dx_dxiT);
    dN_dx = dxi_dxT * dN_dxi;
    gwt = gw(ig)*gw(jg)*Jac;
    AA(1,1:2:7) = N;
    AA(2,2:2:8) = N;
    BB([1,3],1:2:7) = dN_dx;
    BB([3,2],2:2:8) = dN_dx;
    k_el = k_el + BB'*CC*BB * gwt;
    f_el = f_el + AA'*[fx;fy] * gwt;
end
end
A simple MATLAB FE code: the code! VIII

Assembling of element stiffness matrix and load vector

function \([k_{gl},f_{gl}] = \text{assembl\_el}(k_{el},k_{gl},f_{el},f_{gl},\text{dofs\_el})\)

% set global and local indices
index_{gl} = \text{reshape}(\text{dofs\_el}',1,8);
index_{loc} = [1:8];

% assemble \(k_{el}\) into \(k_{gl}\)
\(k_{gl}(\text{index\_gl},\text{index\_gl}) = k_{gl}(\text{index\_gl},\text{index\_gl}) + ...\)
\(k_{el}(\text{index\_loc},\text{index\_loc});\)

return

Construction of global load contribution

function \(f_{gl} = \text{assembl\_load}(f_{gl},\text{loads},\text{dofs})\)

nload = \text{size}(\text{loads},1);

for \(i=1:nload\)
    \(nn = \text{loads}(i,1);\)
    \(\text{index\_gl} = \text{dofs}(nn,:);\)
    \(f_{gl}(\text{index\_gl}) = f_{gl}(\text{index\_gl}) + \text{loads}(i,3:4)';\)
end

return
A simple MATLAB FE code: the code! IX

Imposition of boundary conditions on global stiffness matrix and load vector

function [k_gl,f_gl] = reduce_bc(k_gl,f_gl,bc,dofs)

nbc = size(bc,1);

pt = 2; % pointer to position in "bc" array

for i=1:nbc
    nn = bc(i,1); % node with boundary condition
    for j=1:2 % explore all dofs / node
        if (bc(i,pt+j)==1) % if active boundary condition
            index_gl = dofs(nn,j); % global dof number
            f_gl(index_gl) = 0; % modify global load vector
            k_gl(index_gl,:) = 0; % modify global tangent vector
            k_gl(:,index_gl) = 0;
            k_gl(index_gl,index_gl) = 1;
        end
    end
end
return
A simple MATLAB FE code: some simple considerations I

- **2D elastic matrix**

\[
C = \mu \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \lambda \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \lambda + 2\mu & \lambda & 0 \\ \lambda & \lambda + 2\mu & 0 \\ 0 & 0 & \mu \end{bmatrix}
\]

- **First consideration:** third strain component is \( \gamma \)
- **Second consideration:** 3D constitutive

\[
\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{13} \end{bmatrix} = \begin{bmatrix} \lambda + 2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda + 2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda + 2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \varepsilon_{12} \\ \varepsilon_{23} \\ \varepsilon_{13} \end{bmatrix}
\]

Therefore \( \varepsilon_{33} = \varepsilon_{13} = \varepsilon_{23} = 0 \) ⇒ **plain strain problem** ⇒ \( \sigma_{33} \neq 0 \)
Third consideration: solution for a uniaxial test $\sigma_{22} = 0$  

Consider inverse relation

$$
\begin{align*}
\begin{bmatrix}
\varepsilon_{11} \\
\varepsilon_{22} \\
\varepsilon_{33} \\
\varepsilon_{12}
\end{bmatrix}
&= \frac{1}{E} \begin{bmatrix}
1 & -\nu & -\nu & 0 \\
-\nu & 1 & -\nu & 0 \\
-\nu & -\nu & 1 & 0 \\
0 & 0 & 0 & 1 - \nu
\end{bmatrix}
\begin{bmatrix}
\sigma_{11} \\
\sigma_{22} \\
\sigma_{33} \\
\sigma_{12}
\end{bmatrix}
\end{align*}
$$

Eliminate third equation, imposing $\varepsilon_{33} = 0$, therefore

$$
\frac{1}{E} \left[ -\nu (\sigma_{11} + \sigma_{22}) + \sigma_{33} \right] = 0 \Rightarrow \sigma_{33} = \nu (\sigma_{11} + \sigma_{22})
$$

Assume uniaxial traction condition $\sigma_{22} = \sigma_{12} = 0$, i.e.

$$
\begin{align*}
\varepsilon_{11} &= \frac{\sigma_{11}}{E} - \frac{\nu}{E} \sigma_{33} \\
\varepsilon_{22} &= -\frac{\nu}{E} (\sigma_{11} + \sigma_{33})
\end{align*}
\Rightarrow
\begin{align*}
\varepsilon_{11} &= \frac{1 - \nu^2}{E} \sigma_{11} \\
\varepsilon_{22} &= -\frac{\nu}{E} (1 + \nu) \sigma_{11}
\end{align*}
$$
A simple MATLAB FE code: some simple tests I

Uniaxial patch test

- Pure traction problem

The problem admits a simple analytical solution. Solve the problem for example with

\[ L = 10, \ E = 1000, \ \nu = 0.3 \]

Solution \( @x = y = 10 \) \( \Rightarrow \)
\[
\begin{align*}
\nu &= 9.1 \times 10^{-3} \\
\nu &= -3.9 \times 10^{-3}
\end{align*}
\]
A simple MATLAB FE code: some simple tests II

- Solve now numerically using a $2 \times 2$ element regular mesh
- Solve numerically using a $2 \times 2$ element non-regular mesh
- Solve numerically using a $10 \times 10$ element regular mesh

- Verify that for all the problems it is possible to obtain the correct solution independently of the number of the elements and of the mesh regularity
- Improve the code computing and plotting the stresses and the strains.
A simple MATLAB FE code: some simple tests III

- Element stiffness matrix

\[
\begin{bmatrix}
5.7692 \times 10^2 & 2.4038 \times 10^2 & -3.8462 \times 10^2 & 4.8077 \times 10^1 & -2.8846 \times 10^2 \\
2.4038 \times 10^2 & 5.7692 \times 10^2 & -4.8077 \times 10^1 & 9.6154 \times 10^1 & -2.4038 \times 10^2 \\
-3.8462 \times 10^2 & -4.8077 \times 10^1 & 5.7692 \times 10^2 & -2.4038 \times 10^2 & 9.6154 \times 10^1 \\
4.8077 \times 10^1 & 9.6154 \times 10^1 & -2.4038 \times 10^2 & 5.7692 \times 10^2 & -4.8077 \times 10^1 \\
-2.8846 \times 10^2 & -2.4038 \times 10^2 & 9.6154 \times 10^1 & -4.8077 \times 10^1 & 5.7692 \times 10^2 \\
-2.4038 \times 10^2 & -2.8846 \times 10^2 & 4.8077 \times 10^1 & -3.8462 \times 10^2 & 2.4038 \times 10^2 \\
9.6154 \times 10^1 & 4.8077 \times 10^1 & -2.8846 \times 10^2 & 2.4038 \times 10^2 & -3.8462 \times 10^2 \\
-4.8077 \times 10^1 & -3.8462 \times 10^2 & 2.4038 \times 10^2 & -2.8846 \times 10^2 & 4.8077 \times 10^1 \\
\end{bmatrix}
\]

- Prove the correctness of stiffness matrix, for example summing up all the rows corresponding to horizontal dofs lying on the right of the element. What is the physical meaning of the results？

- Prove that homothetically scaling the element dimension (from a square of side 1 to a square of side 5) the element stiffness does not change.

- Is the previous result general？ Why not？
A simple MATLAB FE code: some simple tests IV

- Pure bending problem

- Solve now numerically using a $1 \times 2$ element regular mesh
- Solve numerically using a $1 \times 2$ element **non-regular** mesh
A simple MATLAB FE code: some simple tests V
Finite difference method

- A general discussion
- A 1D problem
- Extension to 2D
- Finite-differences vs finite-elements
Finite difference method I

**Finite difference method:** a more traditional (but less flexible!) method to solve boundary value problems, directly in the differential form

- **Two main steps** to construct a finite-difference method from any ODE
  1. Approximate derivatives with finite differences (differential $\rightarrow$ algebraic form)
  2. Solve algebraic problem

- Approach the solution directly from the differential form
- Classical method to find a numerical solution of a differential equation
- Derivatives are replaced by **finite difference approximations**
- From differential directly to algebraic form
- No need for a weak (integral) form
1D problem: string under tension

- **Strong formulation**

Given \( f(x) \), \( \bar{g} \) and \( \bar{h} \), find \( u(x) \) such that:

\[
\begin{align*}
\begin{cases}
 u_{xx} + f &= 0 \quad \text{in} \quad \Omega \\
 u(0) &= \bar{g} \\
 u_x(l) &= \bar{h}
\end{cases}
\end{align*}
\]

- Same equation considered as example for FEM
Finite difference method: 1 D problem I

- Introduce a spatial discretization of the domain based on equal spacing $\Delta x$
- Consider $N$-discretization internal points and indicate with $u_i = u(x_i)$

Approximate first derivative as

$$u_{i+\frac{1}{2}}' = \frac{u_{i+1} - u_i}{\Delta x}, \quad u_{i-\frac{1}{2}}' = \frac{u_i - u_{i-1}}{\Delta x}$$

Approximate second derivative as

$$u_i'' = \frac{u_{i+\frac{1}{2}}' - u_{i-\frac{1}{2}}'}{\Delta x} = \frac{u_{i+1} - u_i}{\Delta x} - \frac{u_i - u_{i-1}}{h}$$

Obtain the central difference approximation for the second derivative

$$u_i'' = \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2}$$
Introduce approximation in the differential equation

\[
\frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2} + f_i = 0 \quad \text{for} \quad i = 1, \ldots, N - 1
\]

Previous equations represent a set of \(N\) equations

Need to consider also two boundary conditions

\[
\begin{align*}
  u_0 &= u(x = 0) = \bar{g} \\
  u_N' &= u'(x = l) = \frac{u_N - u_{N-1}}{\Delta x} = \bar{h}
\end{align*}
\]

Obtain a total of \(N+2\) equations for \(N+2\) unknowns
Finite difference method: 2 D problem I

- Possible to generalize to higher dimensional problems

\[ u_{xx} + u_{yy} + f = 0 \quad \text{with} \quad u(x, y) \]

- Central difference approximation

\[
\begin{align*}
  u_{xx}(x_i, y_i) &= \frac{1}{(\Delta x)^2} [u(x_{i+1}, y_i) - 2u(x_i, y_i) + u(x_{i-1}, y_i)] \\
  u_{yy}(x_i, y_i) &= \frac{1}{(\Delta y)^2} [u(x_i, y_{i+1}) - 2u(x_i, y_i) + u(x_i, y_{i-1})]
\end{align*}
\]

- Stencil for uniform grid
Finite elements vs Finite differences I

**Finite element method**
- Very flexible in terms of domain geometry
- Related to model derivation/approximation
  - (beam vs continuum, field order reduction)
- Contain finite-difference method (collocation schemes)
  - Naturally tied to less-natural variational format

**Finite difference method**
- Historically first method
- Naturally tied to differential format
- Computationally very fast
  - Difficult to generalize to non-uniform stencils
Dynamic problems

- Strong form
- FE discrete form
- Free-response / eigenvalue problem
- Frequency response procedures
- General time-histories
Dynamic problem: strong/differential form in compact notation

- All the fields are now time-dependent

\[ \sigma = \sigma(x, t), \quad \varepsilon = \varepsilon(x, t), \quad u = u(x, t) \]

- **Equilibrium:**

\[ \text{div} (\sigma) + b - \rho \ddot{u} = 0 \quad \text{in} \quad \Omega \quad + \quad \text{BC} \quad \sigma(x, t) n = \bar{t}(x, t) \quad \text{on} \quad \partial \Omega_t \]

- **Constitutive equation**

\[ \sigma = \sigma(\varepsilon) \quad \text{general nonlinear elastic material} \]
\[ \sigma = C\varepsilon \quad \text{general linear elastic material} \]

- **Compatibility**

\[ \varepsilon = \nabla^s u \quad + \quad \begin{cases} \text{BC} \quad u(x, t) = \bar{u}(x, t) \quad \text{on} \quad \partial \Omega_u \\ \text{IC} \quad u(x, 0) = \bar{u}_0(x) \end{cases} \]

- Some remarks:

\[ [\text{div} (\sigma)]_i = \sigma_{ij,j} \]
\[ \partial \Omega = \partial \Omega_u \cup \partial \Omega_t \]
\[ \sigma = \sigma(u) \]
Dynamic problem: finite element formulation

- From now on assume trivial displacement BC (i.e. $\tilde{u} = 0$)
- Follow steps very similar to static case
  - Introduce time-independent weight functions
    $$w(x) = N_i(x)w_i$$
  - Obtain
    $$R_i = -R_i^{int} + R_i^{ext} - R_i^{inert} = 0$$
    where
    $$R_i^{int} = \int_\Omega B_i^T \sigma, \quad R_i^{ext} = \int_\Omega N_i^T b + \int_{\partial\Omega_t} N_i^T \bar{t}, \quad R_i^{inert} = \int_\Omega N_i^T \rho \ddot{u}$$
  - Accordingly
    $$R_i(u, \ddot{u}) = 0$$
Dynamic problem: finite element formulation II

- Introduce time-dependence displacement interpolation

\[
\mathbf{u}(\mathbf{x}, t) = \mathbf{N}_I(\mathbf{x})\mathbf{u}_I(t)
\]

such that

\[
\ddot{\mathbf{u}}(\mathbf{x}, t) = \mathbf{N}_I(\mathbf{x})\ddot{\mathbf{u}}_I(t)
\]

- Accordingly

\[
\mathbf{R}_I^{inert} = \int_{\Omega} \mathbf{N}_I^T \rho \mathbf{N}_J \ddot{\mathbf{u}}_J = \mathbf{M}_{IJ} \ddot{\mathbf{u}}_J
\]

with

\[
\mathbf{M}_{IJ} = \int_{\Omega} \mathbf{N}_I^T \rho \mathbf{N}_J
\]

- Dynamical nodal discrete equilibrium equation

\[
\mathbf{R}_I(\mathbf{u}_J, \ddot{\mathbf{u}}_J) = -\mathbf{R}_I^{int}(\mathbf{u}_J) + \mathbf{R}_I^{ext} - \mathbf{M}_{IJ} \ddot{\mathbf{u}}_J = 0
\]

- Dynamical global discrete equilibrium equation

\[
\mathbf{R}(\mathbf{u}, \ddot{\mathbf{u}}) = -\mathbf{R}_I^{int}(\mathbf{u}) + \mathbf{R}_I^{ext} - \mathbf{M} \ddot{\mathbf{u}} = 0
\]
Dynamic problem: finite element formulation III

- For linear elastic problem
  \[ M\ddot{u} + Ku = f \]

- Including also a linear viscous resistance dynamical equilibrium is
  \[ \nabla (\sigma) + b - \rho\ddot{u} - \mu\dot{u} = 0 \]

- Linear elasticity and linear viscous resistance

\[ M\ddot{u} + Cu + Ku = f \] (4)

where

\[ C = \int_\Omega N^T \mu N \]

- In structural analysis often assumption on the form of \( C \)
  \[ C = \alpha M + \beta K \]
  with \( \alpha \) and \( \beta \) to be determined experimentally

★ Focus now on the solution of equation 4, considering this also as the form corresponding to a finite-difference approach (\( M \) and \( C \) diagonal!)
Case 1: no damping or forcing terms. Dynamical linear problem reduces as

\[ M\ddot{u} + Ku = 0 \]  

(5)

A general solution can be written in the form

\[ u = \bar{u}\exp(i\omega t) \]

with the corresponding real part representing an harmonic response since

\[ \exp(i\omega t) = \cos(\omega t) + i\sin(\omega t) \]

Back-substitution into equation 5 returns

\[ \left(-\omega^2 M + K\right)u = 0 \]

(6)

i.e. a general linear eigenvalue or characteristic value problem

For non-zero solutions determinant of coefficient matrix must be zero

\[ \det \left(-\omega^2 M + K\right) = 0 \]

(7)
Provided that $M$ and $K$ are $n \times n$ symmetric-definite matrices, condition 7 gives $n$ positive roots or eigenvalues $\omega_i$ with $i = 1..n$. For each eigenvalue using equation 6 possible to compute the eigenvectors or normal modes $\bar{u}_i$ of the system. Eigenvectors in general are normalized:

$$\bar{u}_i^T M \bar{u}_i = 1$$

s.t.

$$\bar{u}_i^T K \bar{u}_i = \omega_i^2$$

Recall classical orthogonality condition:

$$\bar{u}_i^T M \bar{u}_j = \bar{u}_i^T K \bar{u}_j = 0 \quad \text{for} \quad i \neq j$$
Dynamic problem: forced periodic response I

**Case 2: periodic forcing term** Assuming

\[ f = \bar{f} \exp(\alpha t) \]

with \(\alpha\) complex (i.e. \(\alpha = \alpha_1 + i\alpha_2\)), then general solution again written as

\[ u = \bar{u} \exp(\alpha t) \]

which plugged into linear dynamical equation gives

\[ \left( \alpha^2 M + \alpha C + K \right) \bar{u} = \bar{f} \]

No more an eigenvalue problem, hence it can simply solved as

\[ \bar{u} = \tilde{K}^{-1} \bar{f} \]

where

\[ \tilde{K} = \alpha^2 M + \alpha C + K \]
Case 3: general (non-periodic) forcing term

We just showed that response of a linear dynamical system to a general periodic force can be obtained solving a linear system.

In general an arbitrary forcing function can be represented approximately by a Fourier series.

Accordingly, response to an arbitrary forcing function can be obtained properly combining the response to a series of periodic forcing functions.

The technique of frequency response is readily adapted to problems where the matrix $C$ is of an arbitrary form.
For free response the general solution is

\[ u = \sum_{i}^{n} \bar{u}_i \exp(\alpha_i t) \]

with \( \alpha_i \) complex eigenvalues and \( u_i \) complex eigenvectors.

For forced response we assume that the problem is linear and that the solution can be expressed as a linear combination of the modes

\[ u = \sum_{i}^{n} \bar{u}_i y_i(t) \]  \hspace{1cm} (8)

with \( y_i \) scalar modal participation factors, assumed to be function of time.

Position 8 presents no restriction as all the modes are linearly independent.

Assume simplified version for damping matrix \( C = \alpha M + \beta K \)
Using position 8 in the dynamical equation and premultiplying by the complex conjugate eigenvectors $\bar{u}_i^T$, the result is simply a set of scalar, independent equations

$$m_i\ddot{y} + c_iz + k_iy_i = f_i$$

where

$$m_i = \bar{u}_i^T M \bar{u}_i, \quad c_i = \bar{u}_i^T C \bar{u}_i, \quad k_i = \bar{u}_i^T K \bar{u}_i, \quad f_i = \bar{u}_i^T f$$

since we have

$$\bar{u}_i^T M \bar{u}_j = \bar{u}_i^T C \bar{u}_j = \bar{u}_i^T K \bar{u}_j = 0 \quad \text{when} \quad i \neq j$$

Each scalar equation can be solved by elementary procedures independently.

Total motion is obtained by superposition using 8.
Described analytical solutions provide insights into behaviour of linear systems
- Not computationally economical for the solution of transient problems
- Cannot be extended to nonlinear cases

Investigate now discretization methods directly applicable to time-domain
Since time domain is infinite, we refer to a finite time increment $\Delta t$ considered between $t_n$ and $t_{n+1} = t_n + \Delta t$, obtaining the so-called recurrence relations
MDOF numerical solution II

- **GOAL:** solve equation of motion for a general load condition and possibly for general internal force response
  - Time interval of interest: \([0, T]\)
  - Discretize time interval in \(N\) time sub-intervals of size \(\Delta t\), such that
    \[
    t_0 = 0, \ t_1 = \Delta t, \ t_2 = 2\Delta t, \ldots \ t_N = N\Delta t = T
    \]
    or assume variable time sub-intervals
  - Resort to time-marching numerical scheme, i.e. assume to know solutions up to \(t_n\), want to compute solution at \(t_{n+1}\)
  - Look for numerical solutions with bounded error wrt exact solution
    \[
    u_{n+1} \approx u(t_{n+1}) \quad \text{such that} \quad u_{n+1} = u(t_{n+1}) + O(\Delta t^k)
    \]
    then time integration method is k-order accurate
- Second order accurate methods are desirable !!!!
Central difference method

- Equation of motion at time $t_n$

$$M\ddot{u}_n + C\dot{u}_n + Ku_n = f_n \quad (9)$$

- Central difference approximation of first derivative

$$\dot{u}_n = \frac{u_{n+1} - u_{n-1}}{2\Delta t} \quad (10)$$

- Central difference approximation of second derivative

$$\ddot{u}_n = \frac{\dot{u}_{n+\frac{1}{2}} - \dot{u}_{n-\frac{1}{2}}}{\Delta t} = \frac{u_{n+1} - u_n}{\Delta t} - \frac{u_n - u_{n-1}}{\Delta t} \quad (11)$$

$$= \frac{u_{n+1} - 2u_n + u_{n-1}}{\Delta t^2}$$
Substituting positions 10 and 11 into equation of motion 9, we obtain:

\[
\frac{1}{(\Delta t)^2}M + \frac{1}{2\Delta t}C \mathbf{u}_{n+1} = \mathbf{f}_n - \left[ \frac{1}{(\Delta t)^2}M - \frac{1}{2\Delta t}C \right] \mathbf{u}_{n-1} - \left[ K - \frac{2}{(\Delta t)^2}M \right] \mathbf{u}_n
\]

i.e.

\[
\bar{K}\mathbf{u}_{n+1} = \bar{f}_n
\]

where

\[
\begin{align*}
\bar{K} &= \left[ \frac{1}{(\Delta t)^2}M + \frac{1}{2\Delta t}C \right] \\
\bar{f}_n &= p_n - \left[ \frac{1}{(\Delta t)^2}M - \frac{1}{2\Delta t}C \right] \mathbf{u}_{n-1} - \left[ K - \frac{2}{(\Delta t)^2}M \right] \mathbf{u}_n
\end{align*}
\]

Note: explicit method since restoring force based only on quantities evaluated at \( t_n \)

Easy to extend to non-linear problems
Starting procedure

- Required $u_0, u_{-1}$ to solve for $u_1$ !!

  - Compute acceleration at $t = 0$

    $$\ddot{u}_0 = M^{-1} (f_0 - C\dot{u}_0 - Ku_0)$$

  - Assume that acceleration $\ddot{u}_0$ at $t = 0$ acts for whole time-interval $[t_{-1}, t_0]$.

  - Estimate $u_{-1}$ backward from $t_0$ to $t_{-1}$ and initial conditions at $t_0$

    $$u_{-1} = u_0 + (-\Delta t) \dot{u}_0 + \frac{1}{2} (-\Delta t)^2 \ddot{u}_0$$

    i.e.

    $$u_{-1} = u_0 - \Delta t \dot{u}_0 + \frac{1}{2} (\Delta t)^2 \ddot{u}_0$$
Properties of numerical scheme

- **Consistency** with order of accuracy $k$:
  - gives informations on the relation between numerical approximated scheme and original differential equation based on time-interval discretization $\Delta t$.
  - “Approximated” numerical scheme is said consistent if it reproduces “exact” differential equation when $\Delta t \to 0$.

- **Stability**:
  - Local truncation error in numerical approximation does not grow in time.

- **Convergence**:
  - Numerical solution approaches exact solution when $\Delta t$ goes to zero
Consider equation

\[ \ddot{u} + \omega^2 u = f(t) \]

Define local truncation error \( \tau \) as

\[ \tau = \text{approx. ODE} - \text{exact ODE} \]

In the particular case under investigation

\[ \tau = \left[ \frac{u_{n+1} - 2u_n + u_{n-1}}{\Delta t^2} + \omega u_n - f_n \right] - \left[ \ddot{u}_n + \omega^2 u_n - f_n \right] \]

and after simplyfication

\[ \tau = \frac{u_{n+1} - 2u_n + u_{n-1}}{\Delta t^2} - \ddot{u}_n \] (13)

Expand displacement in Taylor series

\[
\begin{align*}
\left\{ \begin{array}{l}
    u_{n+1} = u_n + \dot{u}_n \Delta t + \frac{1}{2} \ddot{u}_n (\Delta t)^2 + \frac{1}{6} \dddot{u} (\Delta t)^3 + O(\Delta t^4) \\
    u_{n-1} = u_n - \dot{u}_n \Delta t + \frac{1}{2} \ddot{u}_n (\Delta t)^2 - \frac{1}{6} \dddot{u} (\Delta t)^3 + O(\Delta t^4)
\end{array} \right.
\end{align*}
\]
Substituting Taylor expansion in 13 we obtain local truncation error as:

\[ \tau = \frac{u_{n+1} - 2u_n + u_{n-1}}{\Delta t^2} - \ddot{u}_n = \mathcal{O}\left(\Delta t^2\right) \]  

(14)

Central difference method is consistent with second order accuracy

Damping term does not affect second order accuracy
We obtain a condition on time step ⇒ numerical approximation of ODE is conditionally stable

\[ \frac{\Delta t}{T} < \frac{1}{\pi} \]

For accuracy however a good guideline is

\[ \frac{\Delta t}{T} < \frac{1}{10} \]
Equation of motion at time $t_n$ with nonlinear internal force

$$M\ddot{u}_n + C\dot{u}_n + f^{int}(u_n) = f_n$$ (15)

Substituting positions 10 and 11 into equation of motion 15, we obtain:

$$\left[ \frac{1}{(\Delta t)^2} M + \frac{1}{2\Delta t} C \right] u_{n+1} = f_n - \left[ \frac{1}{(\Delta t)^2} M - \frac{1}{2\Delta t} C \right] u_{n-1} + \left[ \frac{2}{(\Delta t)^2} Mu_n + f^{int}(u_n) \right]$$ (16)

i.e.

$$\bar{K}u_{n+1} = \bar{f}_n$$

where

$$\begin{cases}
\bar{K} = \left[ \frac{1}{(\Delta t)^2} M + \frac{1}{2\Delta t} C \right] \\
\bar{f}_n = f_n - \left[ \frac{1}{(\Delta t)^2} M - \frac{1}{2\Delta t} C \right] u_{n-1} + \left[ \frac{2}{(\Delta t)^2} Mu_n + f^{int}(u_n) \right]
\end{cases}$$

Nonlinear internal force can be produced for example by an elasto-plastic spring.
MDOF numerical solution: Newmark’s method

**Newmark’s method**

- Assume form of acceleration during time interval \([t_n, t_{n+1}]\)
  
  **constant acceleration, i.e.**  
  \[ \ddot{u}(\tau) = \frac{1}{2} (\ddot{u}_n + \ddot{u}_{n+1}) \]

  **linear acceleration, i.e.**  
  \[ \ddot{u}(\tau) = \ddot{u}_n + (\ddot{u}_{n+1} - \ddot{u}_n) \frac{\tau}{\Delta t} \quad (17) \]

  with \(\tau \in [0, \Delta t]\) and \(\Delta t = t_{n+1} - t_n\)

- Integrate twice over time interval to obtain velocity and displacement

\[
\begin{align*}
\dot{u}(\tau) &= \dot{u}_n + \frac{1}{2} (\ddot{u}_n + \ddot{u}_{n+1}) \tau \\
\dot{u}(\tau) &= \dot{u}_n + \ddot{u}_n \tau + \frac{1}{4} (\ddot{u}_n + \ddot{u}_{n+1}) \tau^2 \\
\dot{u}(\tau) &= \dot{u}_n + \ddot{u}_n \tau + \frac{1}{2} (\ddot{u}_{n+1} - \ddot{u}_n) \frac{\tau^2}{\Delta t} \\
\dot{u}(\tau) &= \dot{u}_n + \ddot{u}_n \tau + \frac{1}{2} \ddot{u_n} \tau^2 + \frac{1}{6} (\ddot{u}_{n+1} - \ddot{u}_n) \frac{\tau^3}{\Delta t}
\end{align*}
\]
MDOF numerical solution: Newmark’s method II

- Evaluate at the end of time interval

\[
\begin{align*}
\dot{u}_{n+1} &= \dot{u}_n + \frac{1}{2} (\ddot{u}_{n+1} + \ddot{u}_n) \Delta t \\
\ddot{u}_{n+1} &= \ddot{u}_n + \frac{1}{2} (\ddot{u}_{n+1} + \ddot{u}_n) \Delta t \\
u_{n+1} &= u_n + \dot{u}_n \Delta t + \frac{1}{4} (\ddot{u}_{n+1} + \ddot{u}_n) \Delta t^2 \\
u_{n+1} &= u_n + \dot{u}_n \Delta t + \left(\frac{1}{6} \ddot{u}_{n+1} + \frac{1}{3} \ddot{u}_n\right) \Delta t^2
\end{align*}
\]

- Both constant acceleration and linear acceleration can be put in the same form in terms of two parameters $\beta$ and $\gamma$ defining the integration method

\[
\begin{align*}
\dot{u}_{n+1} &= \dot{u}_n + [(1 - \gamma) \ddot{u}_n + \gamma \ddot{u}_{n+1}] \Delta t \\
u_{n+1} &= u_n + \dot{u}_n \Delta t + \left[\left(\frac{1}{2} - \beta\right) \ddot{u}_n + \beta \ddot{u}_{n+1}\right] \Delta t^2
\end{align*}
\]

such that

- constant acceleration $\gamma = \frac{1}{2}$ and $\beta = \frac{1}{4}$

- linear acceleration $\gamma = \frac{1}{2}$ and $\beta = \frac{1}{6}$
MDOF numerical solution: Newmark’s method III

It is interesting that both methods can be put in a predictor-corrector

$$\begin{align*}
\dot{u}_{n+1} &= \dot{u}_n + (1 - \gamma) \ddot{u}_n \Delta t + [\gamma \ddot{u}_{n+1} \Delta t] \\
\ddot{u}_{n+1} &= \ddot{u}_n \Delta t + \left(\frac{1}{2} - \beta\right) \dddot{u}_n \Delta t^2 + [\beta \dddot{u}_{n+1} \Delta t^2]
\end{align*}$$

which can be rewritten as

$$\begin{align*}
\dot{u}_{n+1} &= \ddot{u}_{n+1} + \gamma \Delta t \dddot{u}_{n+1} \\
\ddot{u}_{n+1} &= \ddot{u}_{n+1} + \beta \Delta t^2 \dddot{u}_{n+1}
\end{align*}$$

(18)

where

$$\begin{align*}
\ddot{u}_{n+1} &= \ddot{u}_n + (1 - \gamma) \dddot{u}_n \Delta t \\
\dddot{u}_{n+1} &= \dddot{u}_n \Delta t + \left(\frac{1}{2} - \beta\right) \dddot{u}_n \Delta t^2
\end{align*}$$
MDOF numerical solution: Newmark’s method IV

- Assume to know solution at $t_n$, want to solve at $t_{n+1}$

\[ M\ddot{u}_{n+1} + C\dot{u}_{n+1} + Ku_{n+1} = f_{n+1} \]

- Two options:
  - Express $u_{n+1}$ and $\dot{u}_{n+1}$ in terms of $\ddot{u}_{n+1}$ and solve for $\ddot{u}_{n+1}$
  - Express $\ddot{u}_{n+1}$ and $\dot{u}_{n+1}$ in terms of $u_{n+1}$ and solve for $u_{n+1}$

- Second approach is particularly interesting in case of nonlinear internal forces
MDOF Newmark’s method in terms of displacements I

- From equation 18 we express

\[
\ddot{u}_{n+1} = \frac{1}{\beta \Delta t^2} (u_{n+1} - \ddot{u}_{n+1}) \tag{19}
\]

and using 18

\[
\dot{u}_{n+1} = \ddot{u}_{n+1} + \gamma \Delta t \ddot{u}_{n+1} \\
= \ddot{u}_{n+1} + \frac{\gamma}{\beta \Delta t} (u_{n+1} - \ddot{u}_{n+1}) \tag{20}
\]

- Plugging back into equilibrium equation we obtain

\[
M \left[ \frac{1}{\beta \Delta t^2} (u_{n+1} - \ddot{u}_{n+1}) \right] + C \left[ \ddot{u}_{n+1} + \frac{\gamma}{\beta \Delta t} (u_{n+1} - \ddot{u}_{n+1}) \right] + Ku_{n+1} = f_{n+1}
\]

i.e.

\[
\left[ \frac{1}{\beta \Delta t^2} M + \frac{\gamma}{\beta \Delta t} C + K \right] u_{n+1} = \left[ f_{n+1} + M \frac{1}{\beta \Delta t^2} \ddot{u}_{n+1} + C \frac{\gamma}{\beta \Delta t} \ddot{u}_{n+1} \right]
\]
MDOF Newmark’s method in terms of displacements II

- Previous equation can be rewritten as:

\[ \tilde{K} u_{n+1} = \tilde{f}_{n+1} \]

where

\[
\tilde{K} = \left[ \frac{1}{\beta \Delta t^2} M + \frac{\gamma}{\beta \Delta t} C + K \right] \\
\tilde{f}_{n+1} = \left[ f_{n+1} + M \frac{1}{\beta \Delta t^2} \ddot{u}_{n+1} + C \frac{\gamma}{\beta \Delta t} \dddot{u}_{n+1} \right]
\]

and where we recall that

\[
\begin{aligned}
\dddot{u}_{n+1} &= \dot{u}_n + (1 - \gamma) \ddot{u}_n \Delta t \\
\dddot{u}_{n+1} &= u_n + \dot{u}_n \Delta t + \left( \frac{1}{2} - \beta \right) \dddot{u}_n \Delta t^2 
\end{aligned}
\]  (21)

- Once solved for \( u_{n+1} \), possible to compute new velocities and accelerations as

\[
\begin{aligned}
\dot{u}_{n+1} &= \frac{1}{\beta \Delta t^2} (u_{n+1} - \dddot{u}_{n+1}) \\
\dot{u}_{n+1} &= \dddot{u}_{n+1} + \gamma \Delta t \dddot{u}_{n+1}
\end{aligned}
\]
Simple starting procedure: only need is to compute acceleration $\ddot{u}_0 = \ddot{u}(0)$

$$\ddot{u}_0 = M^{-1} \left( p_0 - C\dot{u}_0 - Ku_0 \right)$$
Evaluation of local truncation error of velocity and acceleration shows that:

\[ \tau = O(\Delta t^k) \]

with:

\[
\begin{cases} 
  k = 2 \quad \text{for } \gamma = \frac{1}{2} & \text{i.e second order} \\
  k = 1 \quad \text{for } \gamma \neq \frac{1}{2} & \text{i.e first order}
\end{cases}
\]