ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Method of Finite Elements I:
Demo 2: Numerical Integration

Adrian Egger

Contents

- **Part 1: Euler Bernoulli Beam Element**
  - Galerkin formulation
  - Example
- **Part 2: Numerical Integration**
  - Gaussian Quadrature
  - Computer Implementation
- **Part 3: Basics of Stress Recovery and Error Estimators**
  - Inter-element continuity
  - Back-calculation of stresses and strains
  - Super-Convergent Patch Recovery

From strong to weak form

- Principle of Virtual Work

$$W_{int} = \int_\Omega \bar{\epsilon}^T \tau d\Omega = W_{ext} = \int_\Omega \bar{\mathbf{u}}^T \mathbf{b} d\Omega + \int_\Gamma \bar{\mathbf{u}}^{ST} \mathbf{T_S} d\Gamma + \sum_i \bar{\mathbf{u}}^{iT} \mathbf{R_C}^i$$

- Principle of Minimum Potential Energy

$$\mathbf{\Pi} = \mathbf{U} - \mathbf{W}$$

$$\mathbf{U} = \frac{1}{2} \int_\Omega \epsilon^T C \epsilon d\Omega$$

$$\mathbf{W} = \int_\Omega \bar{\mathbf{u}}^T \mathbf{b} d\Omega + \int_{\Gamma_T} \bar{\mathbf{u}}^{ST} \mathbf{T_s} d\Gamma_T + \sum_i \bar{\mathbf{u}}_i^T \mathbf{R_C}^i$$

$\mathbf{T_S}$: surface traction (along boundary Γ)

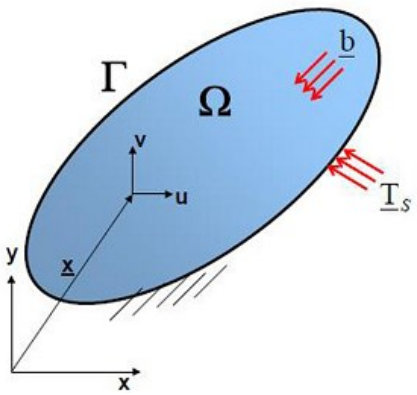
$\mathbf{b}$: body force per unit area

$\mathbf{R_C}$: nodal loads

$\bar{\mathbf{u}}$: virtual displacement

$\bar{\epsilon}$: virtual strain

τ: stresses



- Methods of weighted residuals
  (Galerkin, Collocation, Least Squares methods, etc)

Beam Theory: Strong form

(S)

(1) $$\frac{d^2}{dx^2}\left(EI\frac{d^2v}{dx^2}\right) - p(x) = 0 \quad \text{in } \Omega$$

(2) $$v = \bar{v} \quad \text{displacement} \quad \text{on } \Gamma_u$$

(3) $$\frac{dv}{dx} = -\bar{\theta} \quad \text{angle} \quad \text{on } \Gamma_\theta$$

(4) $$EI\frac{d^2v}{dx^2} = \bar{M} \quad \text{moment} \quad \text{on } \Gamma_M$$

(5) $$-EI\frac{d^3v}{dx^3} = \bar{S} \quad \text{shear force} \quad \text{on } \Gamma_S$$

Boundary conditions

$$\Gamma_u \cap \Gamma_S = \emptyset \qquad \Gamma_u \cup \Gamma_S = \Gamma$$

$$\Gamma_\theta \cap \Gamma_M = \emptyset \qquad \Gamma_\theta \cup \Gamma_M = \Gamma$$

Free end with applied load

$$EI\frac{d^2v}{dx^2} = \bar{M} \text{ on } \Gamma_M$$

$$-EI\frac{d^3v}{dx^3} = \bar{S} \text{ on } \Gamma_S$$

Simple support

$$EI\frac{d^2v}{dx^2} = 0 \text{ on } \Gamma_M$$

$$v = 0 \text{ on } \Gamma_u$$

Clamped support

$$\frac{dv}{dx} = 0 \text{ on } \Gamma_\theta$$

$$v = 0 \text{ on } \Gamma_u$$

From Strong to Weak form I

- Galerkin approach for equations (1), (4), (5):
  1. Multiply by weighting function w
  2. Integrate over the domain
  3. Discretize and sum the contributions of each element in domain

$$\int_0^L w\left((EIv'')'' - p(x)\right) dx = 0 \qquad \text{in } \Omega$$

$$\left[w'\left(EIv'' - \bar{M}\right)\right]_{\Gamma_M} = 0$$

$$\left[w\left(-EIv''' - \bar{S}\right)\right]_{\Gamma_S} = 0$$

From Strong to Weak form II

- Apply the divergence theorem:
  - Equivalent to integration by parts in 1D

$$\int_0^L w\left((EIv'')'' - p(x)\right) dx = 0$$

First integration by parts

$$\left[w\left(EIv''\right)'\right]_\Gamma - \int_0^L w'\left(EIv''\right)' dx - \int_0^L wp(x)dx = 0$$

Second integration by parts gives

$$\left[w\left(EIv''\right)'\right]_\Gamma - \left[w'\left(EIv''\right)\right]_\Gamma - \int_0^L w''EIv''dx - \int_0^L wp(x)dx = 0$$

Beam Theory: Weak form

(W)

Find $v \in \mathcal{S}$ such that

$$\int_0^L w'' EI v'' dx = \int_0^L wp(x)dx + \left[w'\bar{M}\right]_{\Gamma_M} + \left[w\bar{S}\right]_{\Gamma_S} \qquad \forall w \in \mathcal{S}^0$$

$$\mathcal{S} = \{v | v \in \mathcal{C}^1, v = \bar{v} \text{ on } \Gamma_u, v' = \bar{\theta} \text{ on } \Gamma_\theta\}$$

$$\mathcal{S}^0 = \{w | w \in \mathcal{C}^1, w = 0 \text{ on } \Gamma_u, w' = 0 \text{ on } \Gamma_\theta\}$$

Discretization of weight and trial functions:

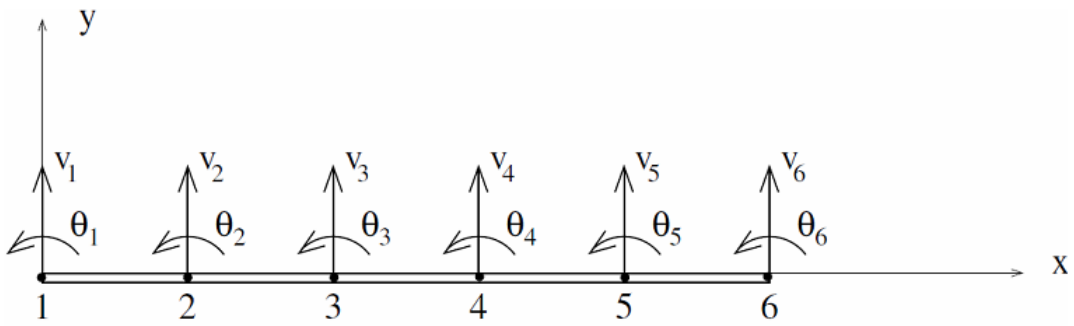
$$w = \sum H_i w_i$$

$$v = \sum H_i v_i$$

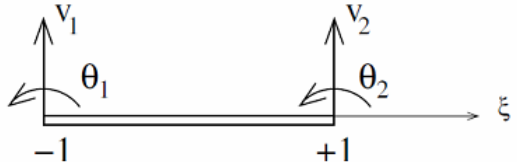
Due to the second order derivatives on the weak form,
Shape functions must be at least twice differentiable.
This mandates $C^{m-1} = C^{2-1} = C^1$-continuity,
i.e. continuity also of the derivatives.

Finite Element formulation

Physical domain



Natural domain



Element displacement vector

$$\boldsymbol{d}^e = \begin{bmatrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \end{bmatrix}$$

Element force vector

$$\boldsymbol{f}^e = \begin{bmatrix} s_1 \\ m_1 \\ s_2 \\ m_2 \end{bmatrix}$$

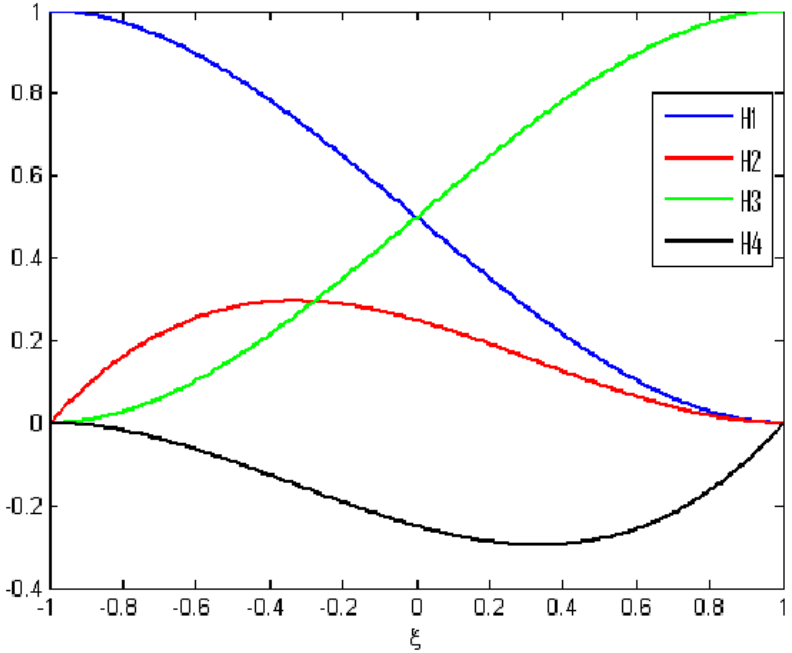
Choice of Shape Functions

Hermite Polynomials

$$H_I = a_I + b_I\xi + c_I\xi^2 + d_I\xi^3$$

$$H_{vI}(x_J) = \delta_{IJ} \qquad H_{\theta I,x}(x_J) = \delta_{IJ}$$

$$\begin{aligned}
v_1 \;&\rightarrow\; H_1 = \frac{1}{4}(1-\xi)^2(2+\xi)\\
\theta_1 \;&\rightarrow\; H_2 = \frac{1}{4}(1-\xi)^2(\xi+1)\\
v_2 \;&\rightarrow\; H_3 = \frac{1}{4}(1+\xi)^2(2-\xi)\\
\theta_2 \;&\rightarrow\; H_4 = \frac{1}{4}(1+\xi)^2(\xi-1)
\end{aligned}$$



Note: The choice of a cubic polynomial is related to the homogeneous strong form of the problem $EIv'''' = 0$.

Jacobian

Finally, the weight functions and trial solutions are:

$$v(\xi) = H_1 v_1 + H_2 \left(\frac{dv}{d\xi}\right)_1 + H_3 v_2 + H_4 \left(\frac{dv}{d\xi}_1\right)_2$$

However note, that the rotation is actually the derivative of the (vertical) deflection: $\theta = \dfrac{dv}{dx}$

The connection between $\dfrac{dv}{dx}$ and $\dfrac{dv}{d\xi}$ is delivered via the **Jacobian**. This is calculated from the coordinate transformation relationship:

$$x = \frac{1-\xi}{2} x_1^e + \frac{1+\xi}{2} x_2^e \Rightarrow \frac{dv}{d\xi} = \theta = \frac{l^e}{2}\frac{dv}{dx}$$

$$J = \frac{dx}{d\xi} = \frac{l^e}{2}$$

where l^e is the length of the element.

Curvature-Displacement Matrix B

From the weak form, we had

$$\int_0^L w'' EI v'' dx = \int_0^L wp(x)dx + \left[w'\bar{M}\right]_{\Gamma_M} + \left[w\bar{S}\right]_{\Gamma_S}$$

The second derivative of the shape functions of the element, $\mathbf{H}^e$, therefore needs to be calculated:

$$\frac{d\mathbf{H}^e}{dx^2} = \mathbf{B}^e = \frac{d\mathbf{H}^e}{d\xi^2} J^{-2} = \frac{1}{l^e}\left[\frac{6\xi}{l^e} \quad 3\xi - 1 \quad -\frac{6\xi}{l^e} \quad 3\xi + 1\right]$$

where $J = \frac{l^e}{2}$

Matrix $\mathbf{B}^e$ now connects the curvature of the element, $\frac{d^2v}{dx^2}$ to the nodal displacement vector $\mathbf{d}^e$:

$$\frac{d^2v}{dx^2} = \mathbf{B}^e\mathbf{d}^e$$

$$\kappa(x,y,z) = B(r,s,t)_{,xx} * u(x,y,z)$$

This is why the Jacobian is necessary!

Derivation by Computer Algebra System (CAS)

```
1 A :=  [[1,r,r^2, r^3]];
  //coefficients of the assumed polynomial shape function
```

$$\begin{bmatrix} 1, & r, & r^2, & r^3 \end{bmatrix}$$

```
2 E := [[1,-1,1,-1],[0,2/L, -4/L, 6/L],[1,1,1,1],[0,2/L, 4/L, 6/L]];
  //polynomial evaluated at each DOF fo each node (r=-1 and r=1)
  //this already includes the jacobian for the rotational DOF!
```

$$\begin{bmatrix} 1, & -1, & 1, & -1 \\ 0, & \frac{2}{L}, & -\frac{4}{L}, & \frac{6}{L} \\ 1, & 1, & 1, & 1 \\ 0, & \frac{2}{L}, & \frac{4}{L}, & \frac{6}{L} \end{bmatrix}$$

```
3 H := factor(A * inv(E));
  //obtaining the shape functions
```

$$\begin{bmatrix} \frac{(r-1)^2*(r+2)}{4}, & \frac{L*(r+1)*(r-1)^2}{8}, & -\frac{(r+1)^2*(r-2)}{4}, & \frac{L*(r+1)^2*(r-1)}{8} \end{bmatrix}$$

```
4 B := simplify(factor(diff(H,r,2))*(L/2)^-2);
  //deriving the shape functions twice to obtain the curvature-displacement matrix B
```

$$\begin{bmatrix} \frac{6*r}{L^2}, & \frac{3*r-1}{L}, & -\frac{6*r}{L^2}, & \frac{3*r+1}{L} \end{bmatrix}$$

```
5 k_analytical := int((transpose(B) * C * I * B * L/2),r,-1,1);
  //analytical integration of the stiffness matrix
```

(L^3)],[(3*C*I*L*(r^3+(r^2)/2))/(L^3),(C*I*L*(3*r^3-r))/(2*L^2),-(3*C*I*L*(r^3+(r^2)/2))/(L^3),(C*I*L*(3*r+1)^3)/(18*L^2)]] has singular points for definite integration in [-1,1]

$$\begin{bmatrix} \frac{12*C*I}{L^3}, & \frac{6*C*I}{L^2}, & -\frac{12*C*I}{L^3}, & \frac{6*C*I}{L^2} \\ \frac{6*C*I}{L^2}, & \frac{4*C*I}{L}, & -\frac{6*C*I}{L^2}, & \frac{2*C*I}{L} \\ -\frac{12*C*I}{L^3}, & -\frac{6*C*I}{L^2}, & \frac{12*C*I}{L^3}, & -\frac{6*C*I}{L^2} \\ \frac{6*C*I}{L^2}, & \frac{2*C*I}{L}, & -\frac{6*C*I}{L^2}, & \frac{4*C*I}{L} \end{bmatrix}$$

Resulting Matrices

Stiffness matrix

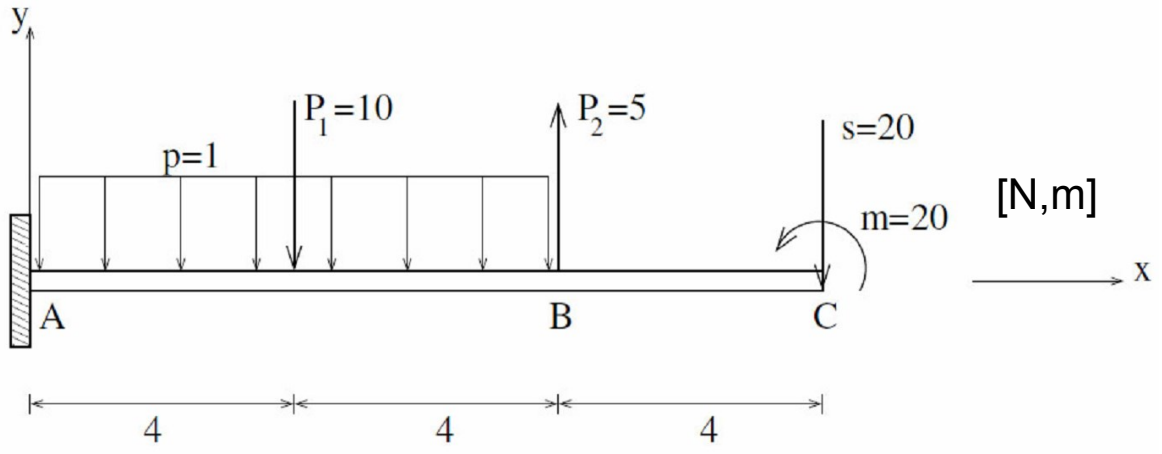
$$\boldsymbol{K}^e = \int_{\Omega^e} (\boldsymbol{B}^e)^T \, EI \, \boldsymbol{B}^e d\Omega^e = \frac{EI}{(l^e)^3} \begin{bmatrix} 12 & 6l^e & -12 & 6l^e \\ 6l^e & 4\,(l^e)^2 & -6l^e & 2\,(l^e)^2 \\ -12 & -6l^e & 12 & -6l^e \\ 6l^e & 2\,(l^e)^2 & -6l^e & 4\,(l^e)^2 \end{bmatrix}$$

Force vector

$$\boldsymbol{f}^e = \underbrace{\int_{\Omega} (\boldsymbol{H}^e)^T \, p(x) dx}_{\boldsymbol{f}^e_\Omega} + \underbrace{\left[\frac{d\,(\boldsymbol{H}^e)^T}{dx} \bar{M} \right]_{\Gamma_M} + \left[(\boldsymbol{H}^e)^T \, \bar{S} \right]_{\Gamma_S}}_{\boldsymbol{f}^e_\Gamma}$$

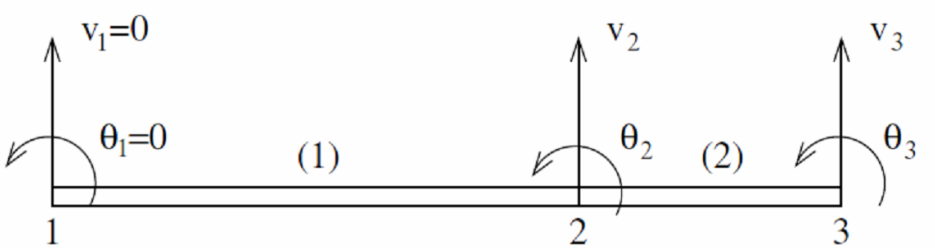
Example: Pre-processing

Analytical model:



Definition of:

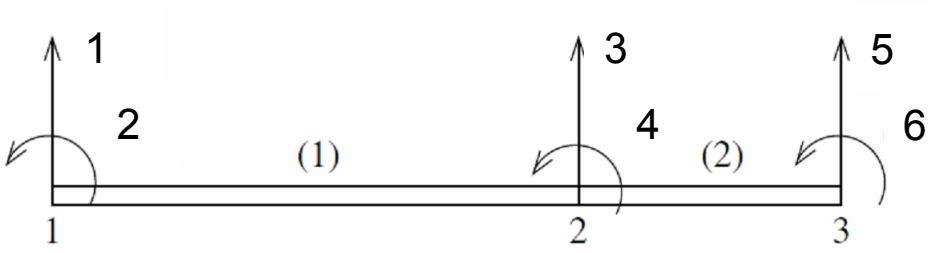
Nodes, Elements, Boundary Conditions and Material properties, etc.



$EI = 10^4 Nm^2$

Assign:

DOF identifiers



Example: Solver → Stiffness Matrix

For element (1)

$$\boldsymbol{K}^1 = 10^3 \begin{bmatrix} 0.23 & 0.94 & -0.23 & 0.94 \\ 0.94 & 5.00 & -0.94 & 2.50 \\ -0.23 & -0.94 & 0.23 & -0.94 \\ 0.94 & 2.50 & -0.94 & 5.00 \end{bmatrix}$$

[1] [2] [3] [4]

For element (2)

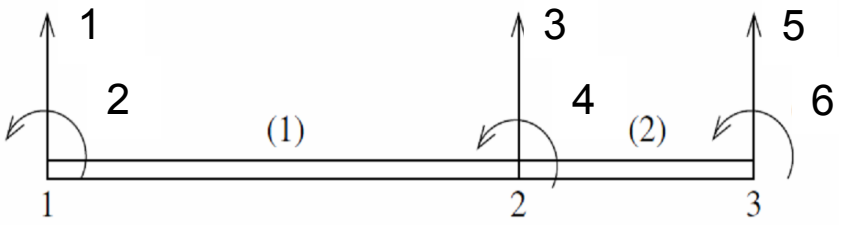
$$\boldsymbol{K}^2 = 10^3 \begin{bmatrix} 1.88 & 3.75 & -1.88 & 3.75 \\ 3.75 & 10.00 & -3.75 & 5.00 \\ -1.88 & -3.75 & 1.88 & -3.75 \\ 3.75 & 5.00 & -3.75 & 10.00 \end{bmatrix}$$

[3] [4] [5] [6]

Assembly into a *Global Stiffness Matrix*

$$\boldsymbol{K} = 10^3 \begin{bmatrix} 0.23 & 0.94 & -0.23 & 0.94 & 0 & 0 \\ & 5.00 & -0.94 & 2.5 & 0 & 0 \\ & & 2.11 & 2.81 & -1.88 & 3.75 \\ & & & 15.00 & -3.75 & 5.00 \\ & SYM & & & 1.88 & -3.75 \\ & & & & & 10.00 \end{bmatrix}$$

[1] [2] [3] [4] [5] [6]



Example: Solver → Force vectors of body loads

Distributed load:

$$\boldsymbol{f}_{\Omega}^{e} = \int_{\Omega} (\boldsymbol{H}^{e})^{T} p dx$$

Point load:

$$\boldsymbol{f}_{\Omega}^{e} = \boldsymbol{H}(\xi_A) P_A$$

For element (1) Given: $P_1 = -10 \text{ and } p(x) = -1$

$$\boldsymbol{f}_{\Omega}^{1} = \int_{\Omega} (\boldsymbol{H}^{e})^{T} p dx + \boldsymbol{H}(\xi_A) P_A = p \int_{\Omega} \begin{bmatrix} H_{v1} \\ H_{\theta 1} \\ H_{v2} \\ H_{\theta 2} \end{bmatrix} dx + P_1 \begin{bmatrix} H_{v1} \\ H_{\theta 1} \\ H_{v2} \\ H_{\theta 2} \end{bmatrix}_{\xi=0} = \begin{bmatrix} -9 \\ -15.3 \\ -9 \\ 15.3 \end{bmatrix}$$

For element (2) Given: $P_2 = 5$

$$\boldsymbol{f}_{\Omega}^{2} = P_2 \begin{bmatrix} H_{v1} \\ H_{\theta 1} \\ H_{v2} \\ H_{\theta 2} \end{bmatrix}_{\xi=-1} = \begin{bmatrix} 5 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Example: Solver → Force vectors of boundary loads

$$\boldsymbol{f}_{\Gamma}^{e} = \left[\frac{d\left(\boldsymbol{H}^{e}\right)^{T}}{dx}\bar{M}\right]_{\Gamma_M} + \left[\left(\boldsymbol{H}^{e}\right)^{T}\bar{S}\right]_{\Gamma_S}$$

Element (1) has no boundary on Γ_S or Γ_M

$$\boldsymbol{f}_{\Gamma}^{1} = \begin{bmatrix}0 & 0 & 0 & 0\end{bmatrix}^{T}$$

For element (2) we have

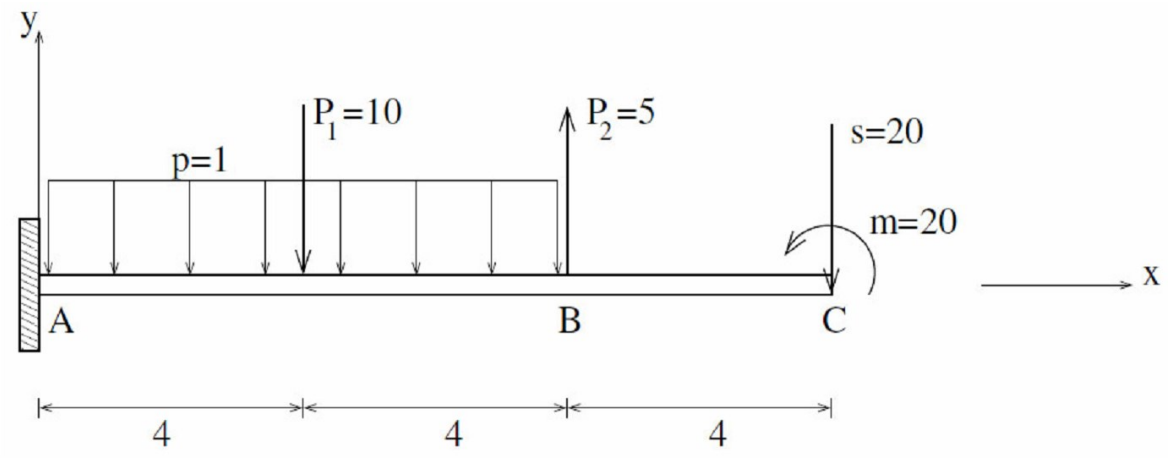
$$\boldsymbol{f}_{\Gamma}^{2} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}_{\Gamma_M} + \begin{bmatrix}0\\0\\1\\0\end{bmatrix}_{\Gamma_S} = \begin{bmatrix}0\\0\\-20\\20\end{bmatrix} \begin{matrix}[3]\\[4]\\[5]\\[6]\end{matrix}$$

Example: Solver → computing displacements

$$\boldsymbol{K}\boldsymbol{d} = \boldsymbol{f} \rightarrow \boldsymbol{d} = \boldsymbol{K}^{-1}\boldsymbol{f}$$

- "Inversion" of stiffness matrix usually dominates required computational time for the finite element solution
  - Direct Methods
    - Efficiency highly dependent on bandwidth of matrix and symmetry
      - Gauss Elimination
      - LU-Decomposition
      - Cholesky-Decomposition
      - Frontal Solvers
      - …
  - Iterative Methods (+Preconditioner)
    - Efficiency highly dependent on condition number of stiffness matrix
      - Conjugate Gradient Method (CG)
      - Generalized Minimal Residual Method (GMRES)
      - Biconjugate Gradient Method (BiCG)
      - …

Example: Post-processing



Element 1:

$$m^1 = EI\frac{d^2v^1}{dx^2} = EI\frac{d^2\boldsymbol{H}}{dx^2}\boldsymbol{d}^1$$

$$s^1 = -EI\frac{d^3v^1}{dx^3} = -EI\frac{d^3\boldsymbol{H}}{dx^3}\boldsymbol{d}^1$$

Element 2:

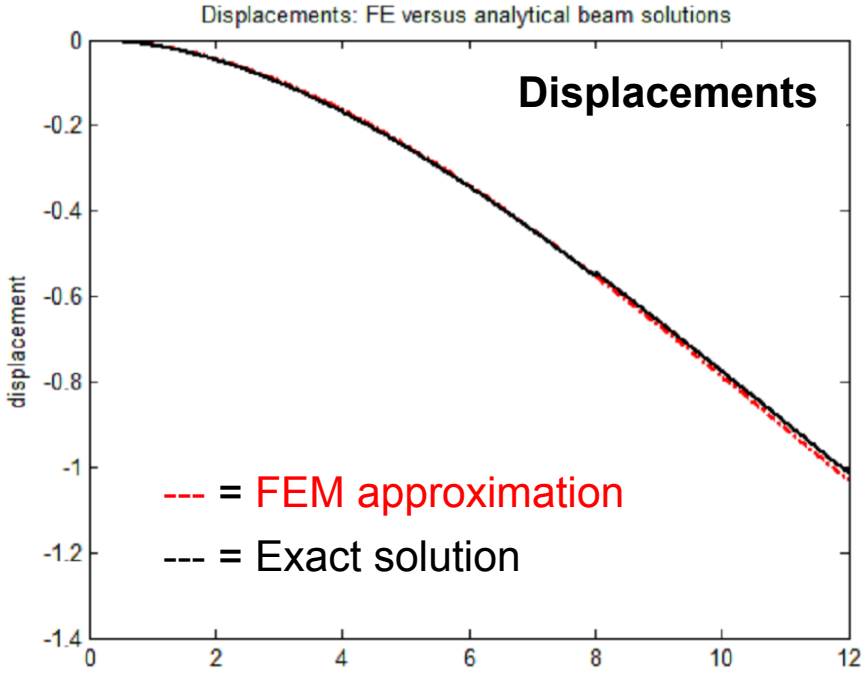
$$m^2 = EI\frac{d^2v^2}{dx^2} = EI\frac{d^2\boldsymbol{H}}{dx^2}\boldsymbol{d}^2$$

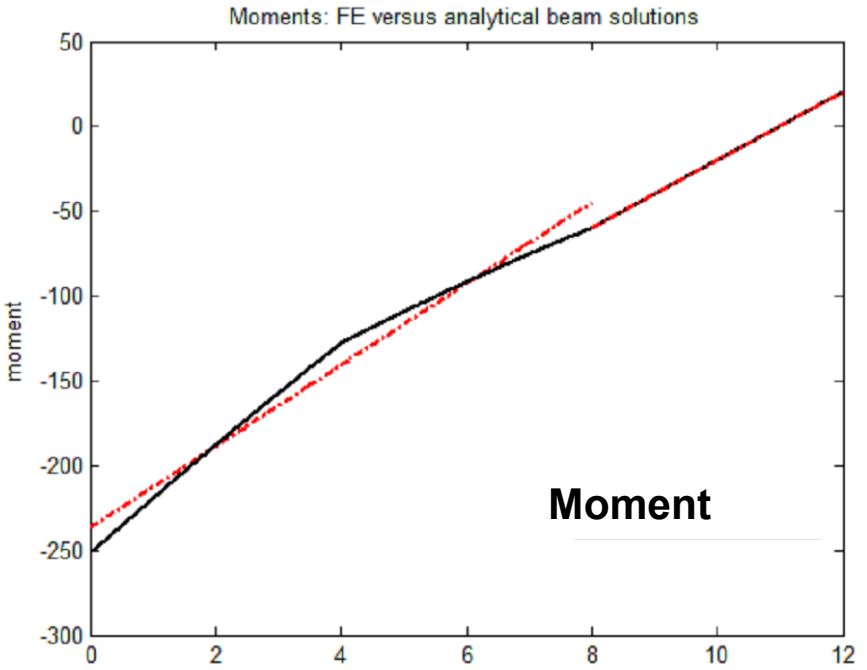
$$s^2 = -EI\frac{d^3v^2}{dx^3} = -EI\frac{d^3\boldsymbol{H}}{dx^3}\boldsymbol{d}^2$$

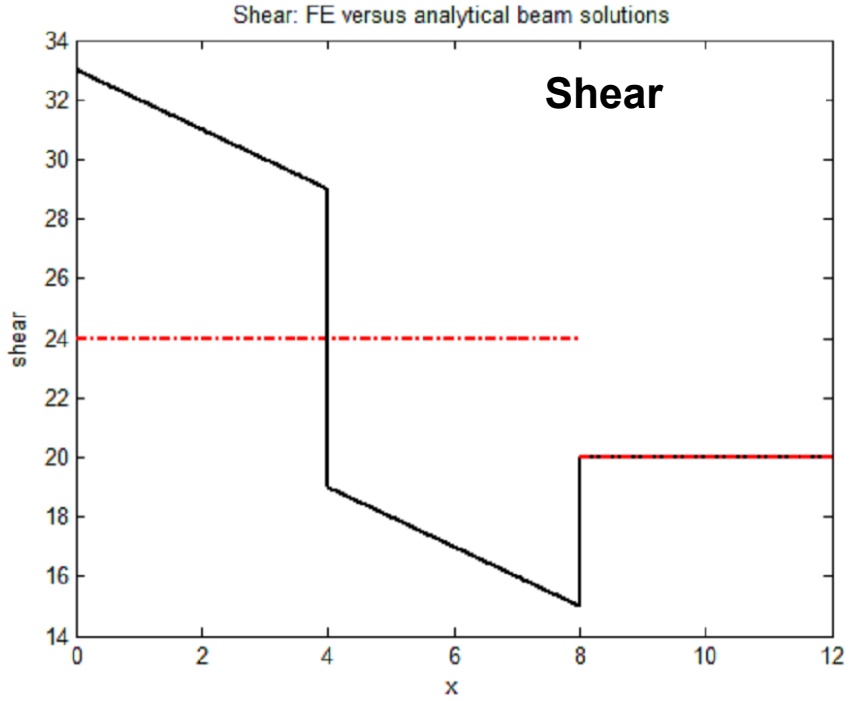
Will this lead to the exact solution for the internal forces?

- No, it will not!
- The assumptions on which the shape functions are based require no element loading!
  - The choice of a cubic polynomial is related to the homogenous form of the problem EIv'' = 0
- The exact solution therefore exists only on elements without element loads (i.e. element 2)

Example: Post-processing







Observations:

- Notice the decay in accuracy for displacements < moments < shear
- Since the **shape functions used, $H_i(x)$, are cubic,** the moment is linear as a 2nd derivative and the shear is constant as a 3rd derivative.

Numerical Integration

- The computation of the stiffness matrix and load vectors requires the evaluation of one or more integrals depending on the dimension of the requested analysis.

$$\mathbf{K}^{\mathbf{e}} = \int_{\Omega^e} \mathbf{B}^T \mathbf{D} \mathbf{B} d\Omega$$

- Why not analytical evaluation of the integral?
  - Analytical solution not always feasible
  - Analytical solution takes too much time to compute
  - No guarantee that numerical issues are removed
    - Division by zero
    - Machine precision induced errors
    - Etc.
  - Post-processing requires numerical evaluation of quantities
    - **B**-matrix

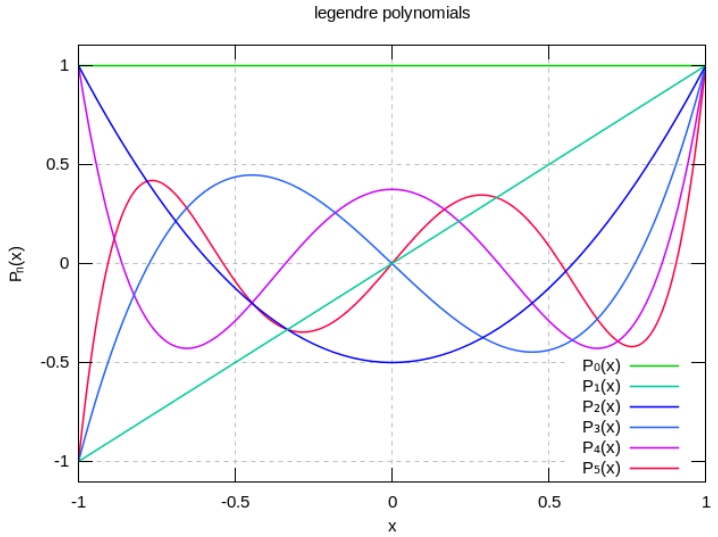
Numerical Integration: Overview

- **Trapezoidal rule**

$$\int_a^b f(x)\,dx \approx (b-a)\left[\frac{f(a)+f(b)}{2}\right].$$

- **Simpsons rule**

$$\int_a^b f(x)\,dx \approx \tfrac{b-a}{6}\left[f(a)+4f\left(\tfrac{a+b}{2}\right)+f(b)\right].$$



- **Newton-Cotes quadrature rules**
  - Assumes equally spaced evaluation points and **optimized weights**
  - Requires n evaluations for exact integration of n -1 degree polynomial

$$\int_a^b f(x)\,dx \approx \sum_{i=0}^{n} w_i\, f(x_i)$$

$$\int_a^b f(x)\,dx \approx \int_a^b L(x)\,dx = \int_a^b \Big(\sum_{i=0}^{n} f(x_i)\, l_i(x)\Big)\,dx = \sum_{i=0}^{n} f(x_i) \underbrace{\int_a^b l_i(x)\,dx}_{w_i}.$$

$$\ell_j(x) := \prod_{\substack{0\le m\le k \\ m\neq j}} \frac{x-x_m}{x_j-x_m} = \frac{(x-x_0)}{(x_j-x_0)}\cdots\frac{(x-x_{j-1})}{(x_j-x_{j-1})}\frac{(x-x_{j+1})}{(x_j-x_{j+1})}\cdots\frac{(x-x_k)}{(x_j-x_k)},$$

Numerical Integration: Gaussian Quadrature

- Especially efficient for the evaluation of polynomials

$$\int_a^b F(r)\,dr = \alpha_1 F(r_1) + \alpha_2 F(r_2) + \cdots + \alpha_n F(r_n) + R_n$$

where both the weights $\alpha_1, \ldots, \alpha_n$ and the sampling points $r_1, \ldots, r_n$ are variables.

  - Position of sampling points **and** value of weights are both optimized
    - The sampling points can be obtained by solving:

$$\int_a^b P(r) r^k\,dr = 0 \qquad k = 0, 1, 2, \ldots, n-1$$

$$P(r) = (r - r_1)(r - r_2)\cdots(r - r_n)$$

    - The weights are computed the same way as with Newton-Cotes:

$$\alpha_j = \int_{-1}^{+1} l_j(r)\,dr; \qquad j = 1, 2, \ldots, n$$

  - Yields exact results for polynomials of degree 2n-1 or lower, where n is equal to the amount of weights / sampling points.

Gaussian Quadrature: Example

EXAMPLE 5.38: Derive the sampling points and weights for two-point Gauss quadrature. In this case $P(r) = (r - r_1)(r - r_2)$ and (5.149) gives the two equations

$$\int_{-1}^{+1} (r - r_1)(r - r_2)\, dr = 0$$

$$\int_{-1}^{+1} (r - r_1)(r - r_2) r\, dr = 0$$

Solving, we obtain $r_1 r_2 = -\frac{1}{3}$

and $r_1 + r_2 = 0$

Hence $r_1 = -\frac{1}{\sqrt{3}}; \qquad r_2 = +\frac{1}{\sqrt{3}}$

The corresponding weights are obtained using (5.150), which in this case gives

$$\alpha_1 = \int_{-1}^{+1} \frac{r - r_2}{r_1 - r_2}\, dr$$

$$\alpha_2 = \int_{-1}^{+1} \frac{r - r_1}{r_2 - r_1}\, dr$$

Since $r_2 = -r_1$, we obtain $\alpha_1 = \alpha_2 = 1.0$.

Gaussian Quadrature: Tables

Depending on the order of the polynomial to be approximated, the corresponding order (number of point) for the quadrature are selected along with the corresponding weights from the following table:

TABLE 5.6 *Sampling points and weights in Gauss-Legendre numerical integration (interval −1 to +1)*

n	r_i	α_i
1	0. (15 zeros)	2. (15 zeros)
2	±0.57735 02691 89626	1.00000 00000 00000
3	±0.77459 66692 41483	0.55555 55555 55556
	0.00000 00000 00000	0.88888 88888 88889
4	±0.86113 63115 94053	0.34785 48451 37454
	±0.33998 10435 84856	0.65214 51548 62546
5	±0.90617 98459 38664	0.23692 68850 56189
	±0.53846 93101 05683	0.47862 86704 99366
	0.00000 00000 00000	0.56888 88888 88889
6	±0.93246 95142 03152	0.17132 44923 79170
	±0.66120 93864 66265	0.36076 15730 48139
	±0.23861 91860 83197	0.46791 39345 72691

Gaussian Quadrature: Example in 1D

```
1 poly1 := 1+2*r-2*r^2 + 4*r^3
```

$$4*r^3-2*r^2+2*r+1$$

2*2_integration_points - 1 ≥ 3

```
2 int(poly1,r,-1,1)
```

$$\frac{2}{3}$$

```
3 Int1 := simplify(subs(poly1,r=-1/sqrt(3)) * 1 +  subs(poly1,r=1/sqrt(3)) * 1)
```

$$\frac{2}{3}$$

```
4 poly2 := 1+2*r-2*r^2 + 4*r^3 - 5* r^4
```

$$-5*r^4+4*r^3-2*r^2+2*r+1$$

2*3_integration_points - 1 ≥ 4

```
5 int(poly2,r,-1,1)
```

$$-\frac{4}{3}$$

```
6 Int2 := simplify(subs(poly2,r=-sqrt(3/5)) * (5/9) +  subs(poly2,r=0) * (8/9) + subs(poly2,r=sqrt(3/5)) * (5/9)
```

$$-\frac{4}{3}$$

Gaussian Quadrature: Example in 1D

```
6 integral := expand((transpose(B) * C * I * B * L/2))
  //highest degreee of r --> 2
  //requires Gaussian quadrature with two integration points
```

$$\begin{bmatrix} \frac{18*C*I*L*r^2}{L^4}, & \frac{9*C*I*L*r^2}{L^3} - \frac{3*C*I*r}{L^2}, & -\frac{18*C*I*L*r^2}{L^4}, & \frac{9*C*I*L*r^2}{L^3} + \frac{3*C*I*r}{L^2} \\ \frac{9*C*I*L*r^2}{L^3} - \frac{3*C*I*r}{L^2}, & \frac{9*C*I*r^2}{2*L} - \frac{3*C*I*r}{L} + \frac{C*I}{2*L}, & -\frac{9*C*I*L*r^2}{L^3} + \frac{3*C*I*r}{L^2}, & \frac{9*C*I*L*r^2}{2*L^2} - \frac{3*C*I*r}{2*L} + \frac{3*C*I*L*r}{2*L^2} - \frac{C*I}{2*L} \\ -\frac{18*C*I*L*r^2}{L^4}, & -\frac{9*C*I*L*r^2}{L^3} + \frac{3*C*I*r}{L^2}, & \frac{18*C*I*L*r^2}{L^4}, & -\frac{9*C*I*L*r^2}{L^3} - \frac{3*C*I*r}{L^2} \\ \frac{9*C*I*L*r^2}{L^3} + \frac{3*C*I*r}{L^2}, & \frac{9*C*I*L*r^2}{2*L^2} - \frac{3*C*I*r}{2*L} + \frac{3*C*I*L*r}{2*L^2} - \frac{C*I}{2*L}, & -\frac{9*C*I*L*r^2}{L^3} - \frac{3*C*I*r}{L^2}, & \frac{9*C*I*r^2}{2*L} + \frac{3*C*I*r}{L} + \frac{C*I}{2*L} \end{bmatrix}$$

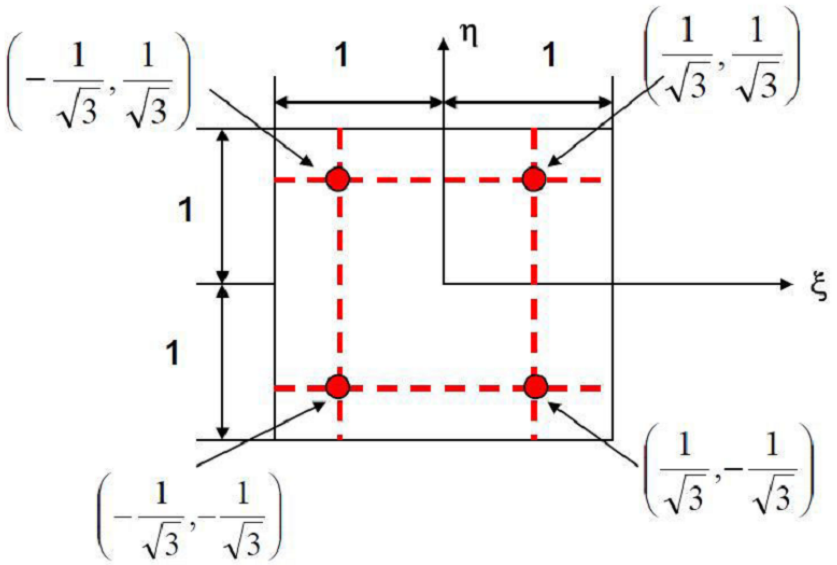
```
7 k_numerical := factor(subs(integral,r = -1/sqrt(3)) * 1 + subs(integral,r = 1/sqrt(3)) * 1)
```

$$\begin{bmatrix} \frac{12*C*I}{L^3}, & \frac{6*C*I}{L^2}, & -\frac{12*C*I}{L^3}, & \frac{6*C*I}{L^2} \\ \frac{6*C*I}{L^2}, & \frac{4*C*I}{L}, & -\frac{6*C*I}{L^2}, & \frac{4*C*I}{2*L} \\ -\frac{12*C*I}{L^3}, & -\frac{6*C*I}{L^2}, & \frac{12*C*I}{L^3}, & -\frac{6*C*I}{L^2} \\ \frac{6*C*I}{L^2}, & \frac{4*C*I}{2*L}, & -\frac{6*C*I}{L^2}, & \frac{4*C*I}{L} \end{bmatrix}$$

Gaussian Quadrature: Reduced Integration

- Reduced integration entails using fewer integration points than required by (full) conventional Gaussian quadrature.
- This has the effect that only a lower degree of polynomial effect can be captured in the integration process.
- This can be beneficial when encountering shear locking as in for example the Timoshenko beam
  - Since we assume the same order of polynomial for displacements and rotations, even though we know they are related by derivative, using reduced integration numerically simulates the use of a lower polynomial. Thus the inherent relations between displacements and rotations can be better accounted for.

Gaussian Quadrature: 2D



$$I = \int_{-1}^{1}\int_{-1}^{1} f(\xi,\eta)d\xi d\eta$$

$$= \sum_{i=1}^{Ngp}\sum_{i=1}^{Ngp} W_i W_j f(\xi_i,\eta_j)$$

where W_i, W_j are the weights and (ξ_i, η_j) are the integration points.

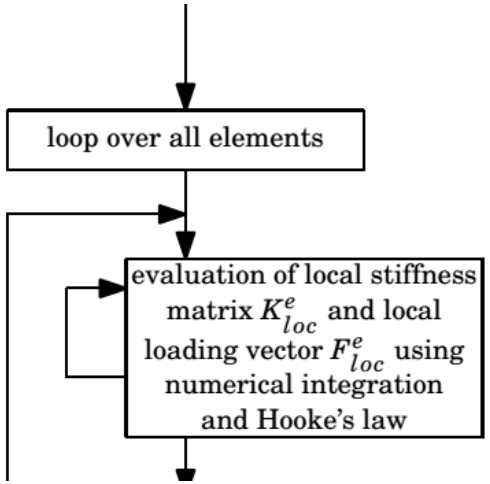
EXAMPLE 5.40: Given that the (i, j)th element of a stiffness matrix $\mathbf{K}$ is $\int_{-1}^{+1}\int_{-1}^{+1} r^2 s^2\, dr\, ds$.

2. Using two-point Gauss quadrature, we have

$$\int_{-1}^{+1}\int_{-1}^{+1} r^2 s^2\, dr\, ds = \int_{-1}^{+1}\left[(1)\left(\frac{1}{\sqrt{3}}\right)^2 + (1)\left(\frac{1}{\sqrt{3}}\right)^2\right] s^2\, ds$$

$$= \int_{-1}^{+1}\frac{2}{3}s^2\, ds = \frac{2}{3}\left[(1)\left(\frac{1}{\sqrt{3}}\right)^2 + (1)\left(\frac{1}{\sqrt{3}}\right)^2\right] = \frac{4}{9}$$

Gaussian Quadrature: Computer Implementation



Processing:
for *each element* **do**
  recall element specific properties (length, cross-section, material)
  recall numerical integration points ξ_i and their weights w_i
  evaluate numerical integration of local stiffness matrix and load vector:
  for *each numerical integration point* **do**
    recall evaluated shape functions H_i and derivatives B_i
    evaluate Jacobian J and its determinant $|J|$
    evaluate local stiffness matrix $\mathbf{K^e_{loc}} = \mathbf{K^e_{loc}} + \mathbf{B^T D B |J|}\Big|_{\xi_i} \cdot w_i$
    evaluate local loading vector $\mathbf{F^e_{loc}} = \mathbf{F^e_{loc}} + \mathbf{H^T f}\Big|_{\xi_i} \cdot w_i$
  end

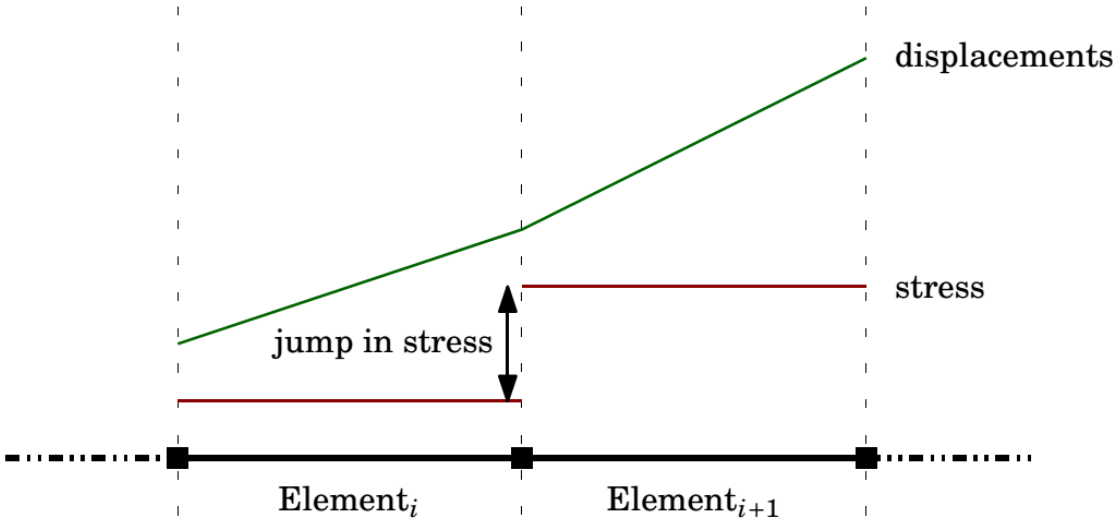
```
9   [ samplingPoints , weights ] = integrationParameters1D ( nodesPerElement , quadratureType);
12  [shapeFunction , shapeFunctionDerivative] = shapeFunction1D (nodesPerElement , samplingPoint);

        x   = shapeFunction * xCoord;
        y   = shapeFunction * yCoord;
      x_n = shapeFunctionDerivative * xCoord;
      y_n = shapeFunctionDerivative * yCoord;
13    detJ = x * y_n - y * x_n;
      B = 1/detJ * [y_n ,   0 ;...
                      0 , -x_n;...
                   -x_n ,  y_n] * shapeFunction2D;

14  k_temp = k_temp + transpose(B) * D * B * detJ * weight;
```

Extremely important remarks:

- All calculations and quantities obtained only at integrations points!
- Thus, strain-displacement matrix **B** evaluated exactly only at the integration points!
- Thus stresses and strains most accurate at Gauss integration points!

Inter-element Continuity (at nodal points)

- In finite element analysis, we often encounter C^0-continuity
  - C^0-continuity: continuity of displacements only
  - C^1-continuity: continuity of once derived quantities (stress & strain)
  - C^2-continuity: continuity of twice derived quantities
  - …



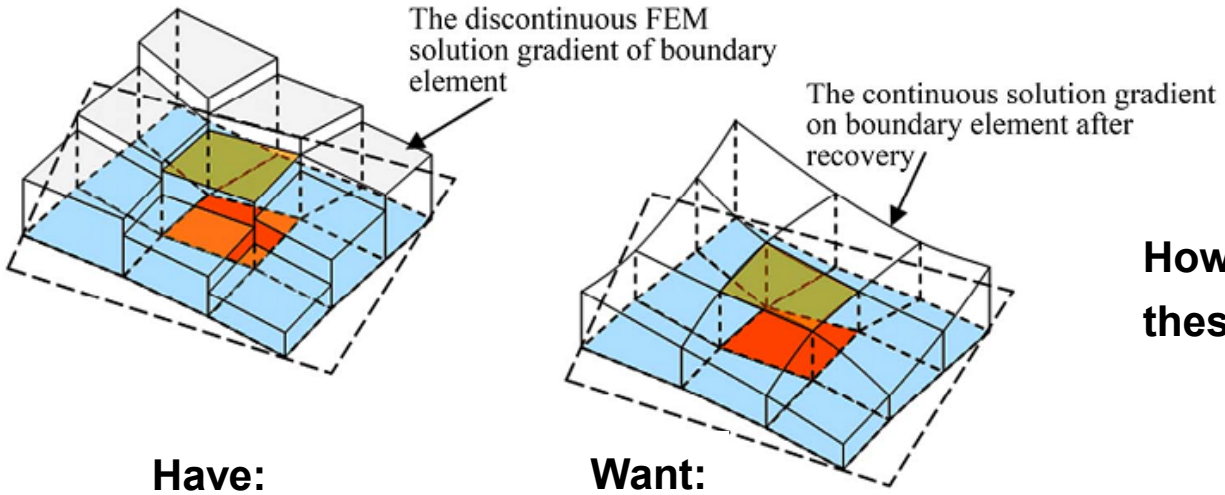
- Which value for the stress due we choose?

Post-processing of Stress and Strain

- Given the displacement solution:

$$\boldsymbol{K} * \boldsymbol{d} = \boldsymbol{F} \rightarrow \boldsymbol{d} = \boldsymbol{K}^{-1} * \boldsymbol{F}$$

- Strains: $\boldsymbol{\varepsilon} = \boldsymbol{B} * \boldsymbol{d}$
- Stress: $\boldsymbol{\sigma} = \boldsymbol{C} * \boldsymbol{B} * \boldsymbol{d}$
  - Stress and strain evaluated using B-matrix anywhere on domain
  - Continuous on each element
  - Jumps across elements



Have: **Want:**

How does one remove these unwanted jumps?

Superconvergent Patch Recovery Theory

- Assumption:
  - There exist points within the element, namely the gauss integration points, at which the stresses are superconvergent.
- Aim:
  - Use these stresses at the gauss integration points to recover nodal stresses that are also superconvergent.
  - Use least squares fit to find a better solution
- New possibilities:
  - Based on the difference between raw (“computed”) stress and recovered (“averaged”) some options arise:
    - A posteriori error estimator
    - Adaptive mesh refinement
    - C^0 continuous stress and strain distribution

Exact Error Estimator

A solution method will yield a displacement field $\{u_h(x,y)\}$ which is an approximation to the true displacement field

$$\{u_h(x,y)\} \approx \{u(x,y)\} \tag{8}$$

Although the error in the displacement field varies over the solution domain, a scalar norm of the error may be used to characterize the accuracy of the solution over a finite volume. A commonly used norm is the so-called energy norm, which can be interpreted as a weighted root-mean-square of the stresses.

$$\|\{\sigma(x,y)\}\|_V = \left(\int_V \{\sigma(x,y)\}^{\mathrm{T}} \{\varepsilon(x,y)\} \, \mathrm{d}V \right)^{\frac{1}{2}} = \left(\int_V \{\sigma(x,y)\}^{\mathrm{T}} [D]^{-1} \{\sigma(x,y)\} \, \mathrm{d}V \right)^{\frac{1}{2}} \tag{9}$$

If the local stress error is defined as

$$\{e_\sigma(x,y)\} = \{\sigma(x,y)\} - \{\sigma_h(x,y)\} \tag{10}$$

the energy norm of the error in stress is

$$\|\{e_\sigma(x,y)\}\|_V = \left(\int_V \{e_\sigma(x,y)\}^{\mathrm{T}} [D]^{-1} \{e_\sigma(x,y)\} \, \mathrm{d}V \right)^{\frac{1}{2}} \tag{11}$$

Weighted per cent root-mean-square of the error in the stress field:

$$\eta = \frac{\|\{e_\sigma(x,y)\}\|_V}{\|\{\sigma(x,y)\}\|_V} \times 100\%$$

Reference: DOI: 10.1002/nme.439

Finite Element Error Estimator

Unfortunately, since in general the exact stress field is not known, the relative energy norm of the stress error cannot be computed directly. Instead a recovery procedure is used to obtain an approximation to the exact stress field which is significantly better than the raw stresses computed from the displacement field. This 'better' approximation (designated by a superscript asterisk) can be used in place of the exact stress field to form an approximation of the error [8]

$$\{\sigma(x,y)\} \approx \{\sigma^*(x,y)\} \tag{13}$$

The stress error can be approximated by

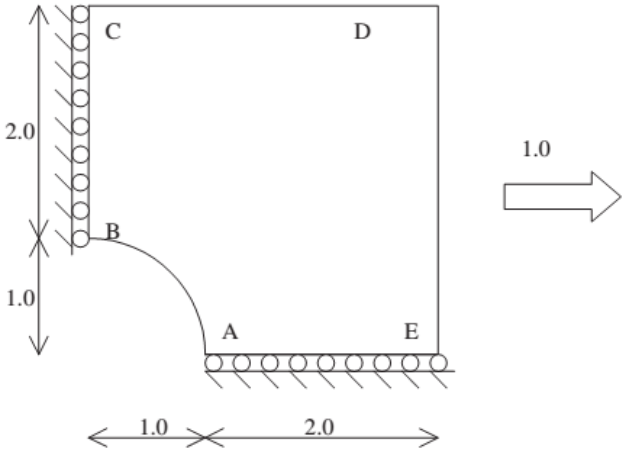
$$\{e_\sigma(x,y)\} \approx \{e_\sigma^*(x,y)\} = \{\sigma^*(x,y)\} - \{\sigma_h(x,y)\} \tag{14}$$

and the error estimator is then

$$\eta^* = \frac{\|\{e_\sigma^*(x,y)\}\|_V}{\|\{\sigma^*(x,y)\}\|_V} \times 100\% \tag{15}$$

Reference: DOI: 10.1002/nme.439

Raw vs. Recovered Stress Fields:



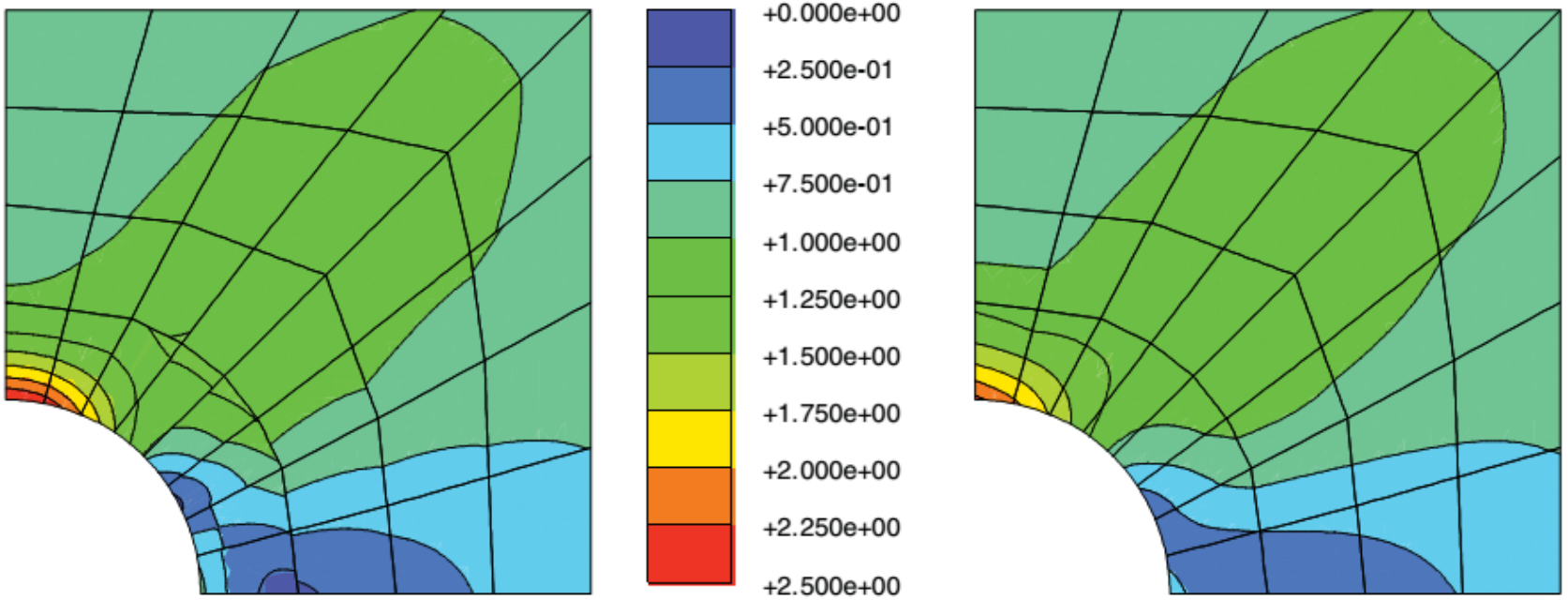


Plate 1. Raw (left) and recovered (right) stress fields computed by the finite element method (for the intermediate mesh of the example).

Reference: DOI: 10.1002/nme.439

Visualization of the Error-Estimator

Comparison with exact solution: Why should this result not surprise us?

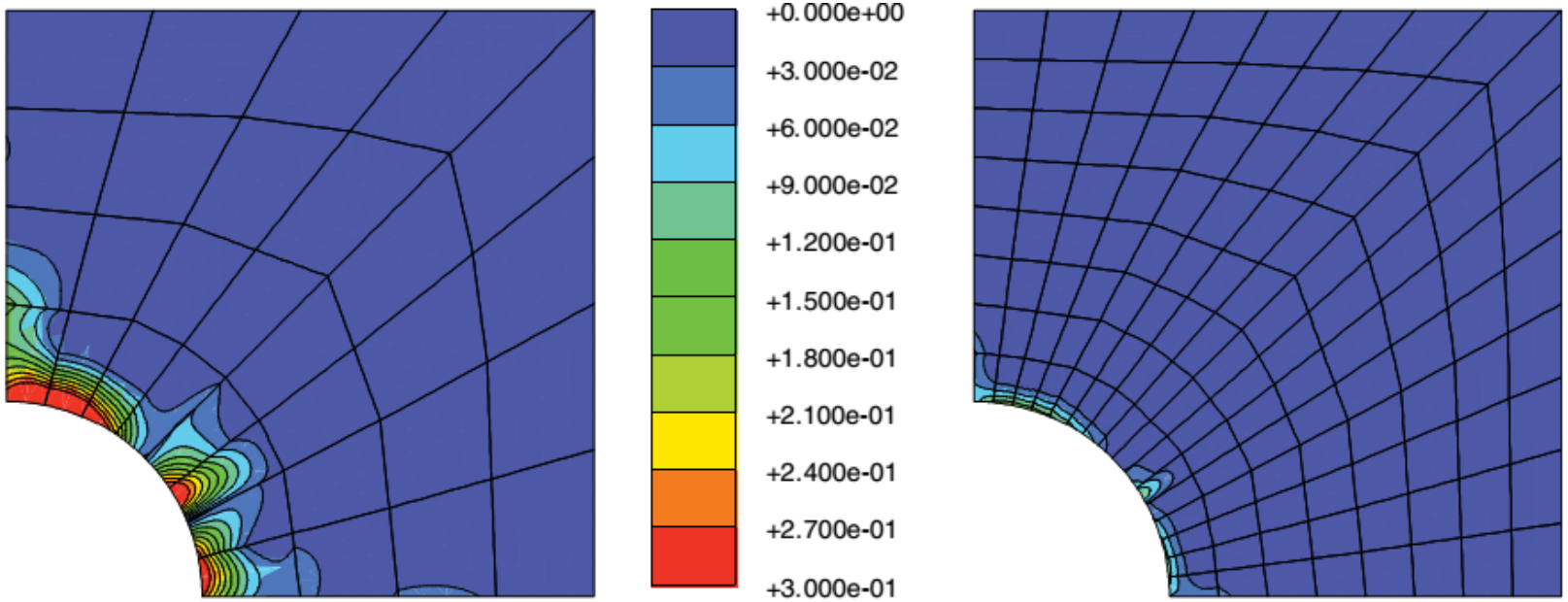


Plate 4. Point-wise variation of effective stress error for the intermediate and fine meshes analysed by the finite element method.

Table II. Performance of the standard finite element error estimator for Example 1.

Mesh	DOF	$\eta^*(\%)$	$\eta(\%)$	δ_x^*	σ_x^*	Time(s)
Coarse	64	14.0	6.6	2.974	1.597	0.1
Intermediate	224	4.6	2.7	2.999	2.333	0.2
Fine	832	1.1	1.0	3.000	2.601	0.8

Reference: DOI: 10.1002/nme.439

Questions