

Formulation and calculation of isoparametric finite element matrixes

1. Numerical integration
2. Implementation of a finite element computer code

Gilles Richner

20.12.06

Numerical integration

Goal: Integration of the matrix $F(r)$

Numerical integration

$$\int_a^b F(r) dr = \sum_i \alpha_i F(r_i)$$

With $a=-1$ and $b=1$ (isoparametric elements)

Approximation of $F(r)$ with $\psi(r)$

Polynomial:

$$\psi(r) = a_0 + a_1 r + a_2 r^2 + \dots + a_n r^n$$

Lagragian interp. :

$$\psi(r) = F_0 l_0(r) + F_1 l_1(r) + F_2 l_2(r) + \dots + F_n l_n(r)$$

With

$$l_j(r) = \frac{(r - r_0)(r - r_1) \cdots (r - r_{j-1})(r - r_{j+1}) \cdots (r - r_n)}{(r_j - r_0)(r_j - r_1) \cdots (r_j - r_{j-1})(r_j - r_{j+1}) \cdots (r_j - r_n)}$$

Numerical integration II

Newton: equally spaced sampling points

$$\int_a^b F(r) dr = (b - a) \sum_{i=0}^n C_i^n F(r_i) \quad \text{With } C_i^n = \text{Newton-Cotes constant}$$

Gauss: variation of sampling interval

$$\int_a^b F(r) dr = \sum_i \alpha_i F(r_i) \quad \text{With } \alpha_i = \int_{-1}^1 l_j(r) dr$$

Multidimensional integration

$$\int_{-1}^1 \int_{-1}^1 \int_{-1}^1 F(r, s, t) dr ds dt = \sum_{i,j,k} \alpha_i \alpha_j \alpha_k F(r_i, s_j, t_k)$$

Numerical integration III

Kind and order of integration

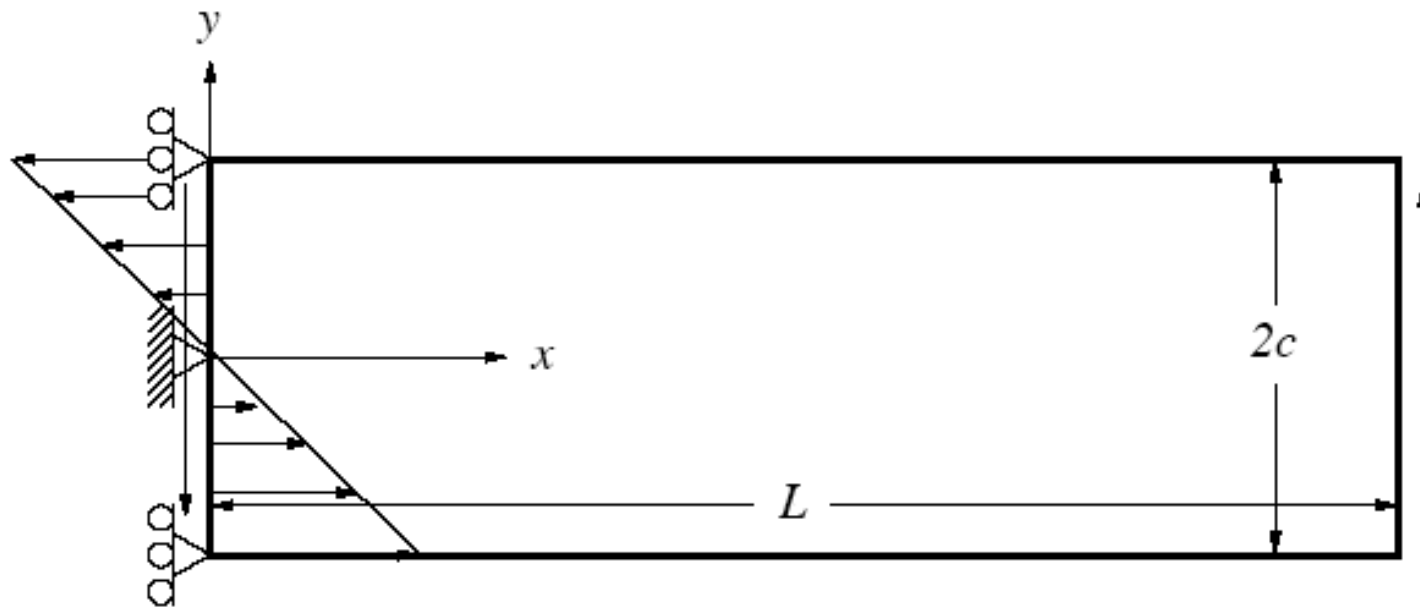
- => cost of analysis, errors, degree of local variables, linearity, ...
- => For displacement matrix => full integration

Reduced and Selective

- => Improving overall results by reducing the order of certain matrix evaluation
- => Different strains terms integrated with different orders

=> Stability and convergence

Example for FEM code



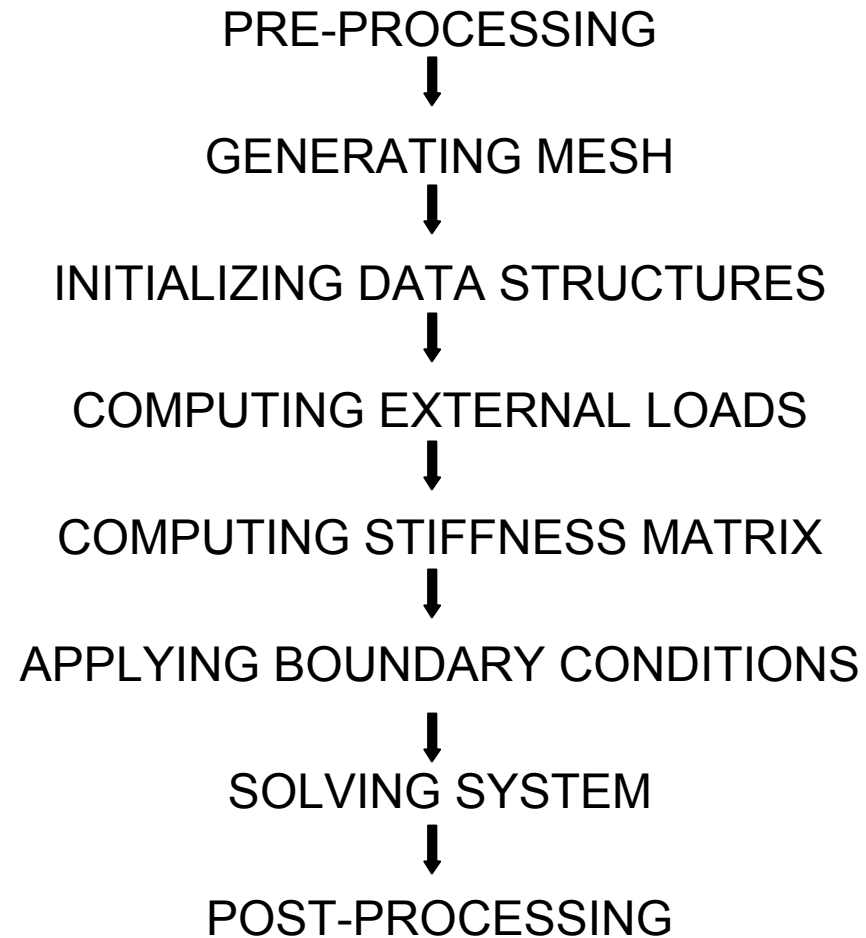
Displacement boundary conditions:

$$u_x = 0 \text{ @ } (0; c), (0,0) \text{ and } (0; -c), u_y = 0 \text{ @ } (0; 0).$$

Traction boundary conditions:

$$t_x = y \text{ on } x = 0 \text{ and } t_y = P(x^2 - c^2) \text{ on } x = L.$$

Implementation of computer code



Implementation of computer code II

PRE-PROCESSING

GENERATING MESH

INITIALIZING
DATA STRUCTURES

COMPUTING
EXTERNAL LOADS

COMPUTING
STIFFNESS MATRIX

APPLYING BOUNDARY
CONDITIONS

SOLVING SYSTEM

POST-PROCESSING

- **Material properties**
- **Beam properties**
- **Mesh properties**
- **Model assumption**

Implementation of computer code III

PRE-PROCESSING

GENERATING MESH

INITIALIZING
DATA STRUCTURES

COMPUTING
EXTERNAL LOADS

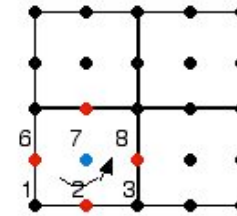
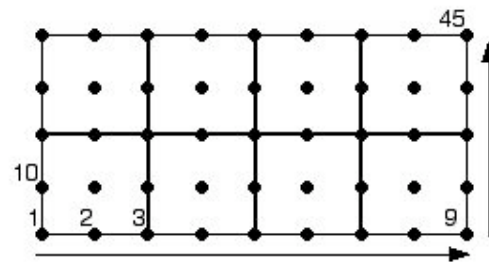
COMPUTING
STIFFNESS MATRIX

APPLYING BOUNDARY
CONDITIONS

SOLVING SYSTEM

POST-PROCESSING

➤ Element connectivity matrix: Node pattern



➤ Boundary node pattern

➤ Displacement boundary

Implementation of computer code IV

PRE-PROCESSING

GENERATING MESH

INITIALIZING
DATA STRUCTURES

COMPUTING
EXTERNAL LOADS

COMPUTING
STIFFNESS MATRIX

APPLYING BOUNDARY
CONDITIONS

SOLVING SYSTEM

POST-PROCESSING

➤ Nodal displacement vector

```
U=zeros(2*numnode,1);
```

➤ External load vector

```
f=zeros(2*numnode,1);
```

➤ Stiffness matrix

```
K=sparse(2*numnode,2*numnode);
```

With numnode = total # nodes

Implementation of computer code V

PRE-PROCESSING

GENERATING MESH

INITIALIZING
DATA STRUCTURES

COMPUTING
EXTERNAL LOADS

COMPUTING
STIFFNESS MATRIX

APPLYING BOUNDARY
CONDITIONS

SOLVING SYSTEM

POST-PROCESSING

➤ Gaussian quadrature

```
[W,Q]=quadrature( 4, 'GAUSS', 1 );
```

Where W and Q are the sampling points and weights

➤ Shape function

```
[N,dNdx]=lagrange_basis(edgeElemType,pt);
```

➤ Integration of the tractions on the left and right edges

```
f(sctrx)=N*fxPt*detJ0*wt
```

With $fxPt$ = x traction at quadrature point

Implementation of computer code VI

PRE-PROCESSING

GENERATING MESH

INITIALIZING
DATA STRUCTURES

COMPUTING
EXTERNAL LOADS

COMPUTING
STIFFNESS MATRIX

APPLYING BOUNDARY
CONDITIONS

SOLVING SYSTEM

POST-PROCESSING

➤ Shape function

```
[W,Q]=quadrature( 4, 'GAUSS', 2 );
```

```
[N,dNdx]=lagrange_basis(edgeElemType,pt);
```

```
J0=node(sctr,:)'*dNdx;
```

```
dNdx=dNdx*invJ0;
```

➤ Compute Element Stiffness at quadrature point

```
K(sctrB,sctrB)=B'*C*B*W(q)*det(J0)
```

Implementation of computer code VII

PRE-PROCESSING

GENERATING MESH

INITIALIZING
DATA STRUCTURES

COMPUTING
EXTERNAL LOADS

COMPUTING
STIFFNESS MATRIX

APPLYING BOUNDARY
CONDITIONS

SOLVING SYSTEM

POST-PROCESSING

Enforcement of the essential boundary conditions

Modifying the system but keeping it symmetric so that the boundary condition are satisfied according to the applied mapping.

Implementation of computer code VIII

PRE-PROCESSING

GENERATING MESH

INITIALIZING
DATA STRUCTURES

COMPUTING
EXTERNAL LOADS

COMPUTING
STIFFNESS MATRIX

APPLYING BOUNDARY
CONDITIONS

SOLVING SYSTEM

POST-PROCESSING

➤ Solving system

$$U=K \setminus f$$

➤ Compute Element strain and stress at the stress point

$$\text{strain}=B*U(\text{sctr}B);$$
$$\text{stress}(e,q,:)=C*\text{strain};$$

Literature

Code:

<http://www.tam.northwestern.edu/jfc795/Matlab/>

The finite element method, O.C. Zienkiewicz, R.L. Taylor - Butterworth
Heinemann (2000) Vol.1-3

Additional:

Matlab guide to finite elements: an interactive approach, (2005) P.I. Kattan

Introduction to finite and spectral element methods using Matlab,(2003)
C. Pozrikidis

The finite element method using Matlab (2000), Y.W. Kwon, H. Bang