MECH 570C

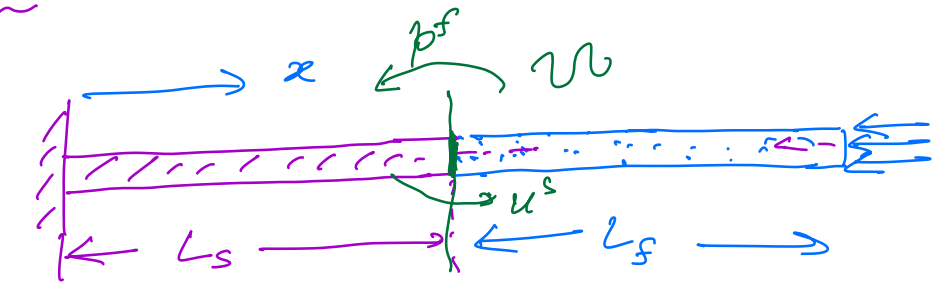
Fluid-Structure Interaction

Module 4: Finite Element Method for Continuum Mechanics (Part 1)

Rajeev K. Jaiman

Winter Term 2, 2020/21

1-D Fluid-Solid Interaction:



ρ_s, E

$P = \hat{p}\cos(\Omega t)$

Solid Part:

(linear) (*) $\dfrac{\partial \sigma^s}{\partial x} = \rho^s \dfrac{\partial^2 u^s}{\partial t^2}$

$\sigma^s = E\epsilon^s$

$\epsilon^s = \dfrac{\partial u^s}{\partial x}$

$0 < x < L_s$

Fluid Part:

Linearized pressure Eqn (**) $\dfrac{\partial^2 p^f}{\partial x^2} = \dfrac{1}{c^2}\dfrac{\partial^2 p^f}{\partial t^2}$

$L_s < x < L_s + L_f$

FSI interface: Traction/Stress Continuity usually

(a) $x = L_s$

(1) Diff w.r.t x $\sigma^s = -p^f + (\text{viscous} \to 0)$

(2) $\Rightarrow \dfrac{\partial p^f}{\partial x} = -\dfrac{\partial \sigma^s}{\partial x} = -\rho^s \dfrac{\partial^2 u^s}{\partial t^2}$

$$u^s(x,t) = \bar{u}\cos(\Omega t)$$
$$p^f(x,t) = \bar{p}\cos(\Omega t)$$

(3) $$\frac{\partial^2 \bar{u}}{\partial x^2} = -\frac{\Omega^2}{\omega_s^2}\bar{u}$$

(4) $$\frac{\partial^2 \bar{p}}{\partial x^2} = -\frac{\Omega^2}{c^2}\bar{p}$$

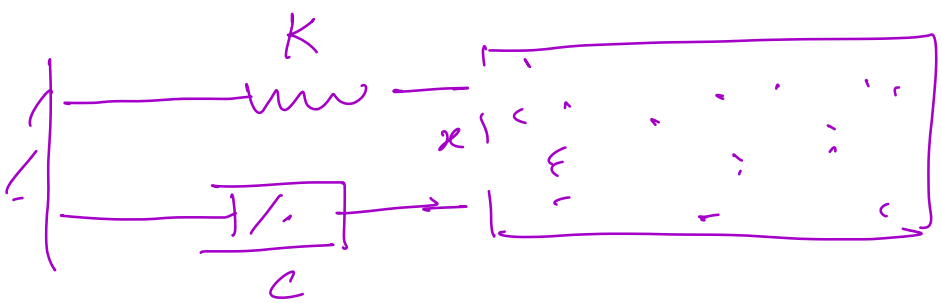
$$\bar{u}(x) = A \sin\frac{\Omega}{\omega_s}x$$

$$\bar{p}(x) = \hat{p}\cos\frac{\Omega}{c}(L-x) + B\sin\frac{\Omega}{c}(L-x)$$

$$\begin{bmatrix} E\Omega\cos\frac{\Omega L_s}{\omega_s} & \sin\Omega L_f \\ \rho^f \Omega c \sin\frac{\Omega L_s}{\omega_s} & \cos\frac{\Omega L_f}{c} \end{bmatrix} \begin{Bmatrix} A \\ B \end{Bmatrix} = \begin{bmatrix} -\hat{p}\cos\Omega L_f/c \\ \hat{p}\sin\frac{\Omega L_f}{c} \end{bmatrix}$$

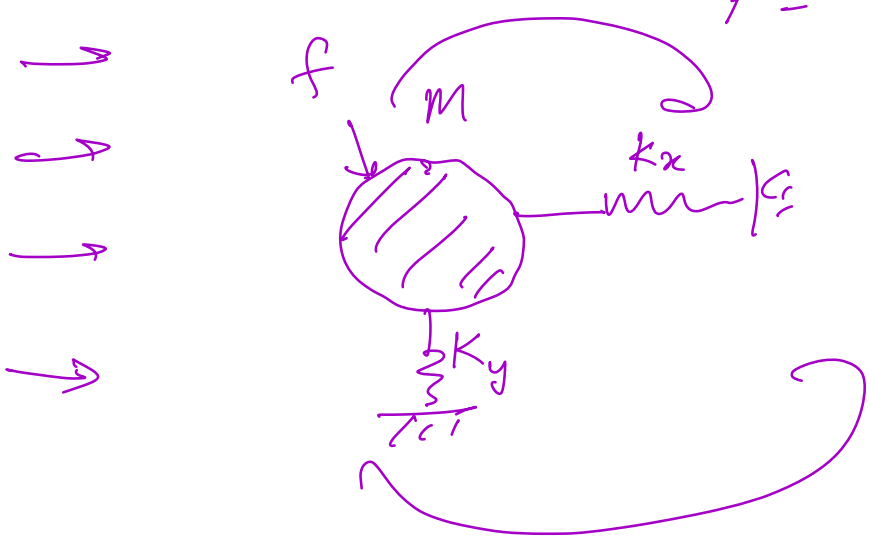
Solve for A & B to

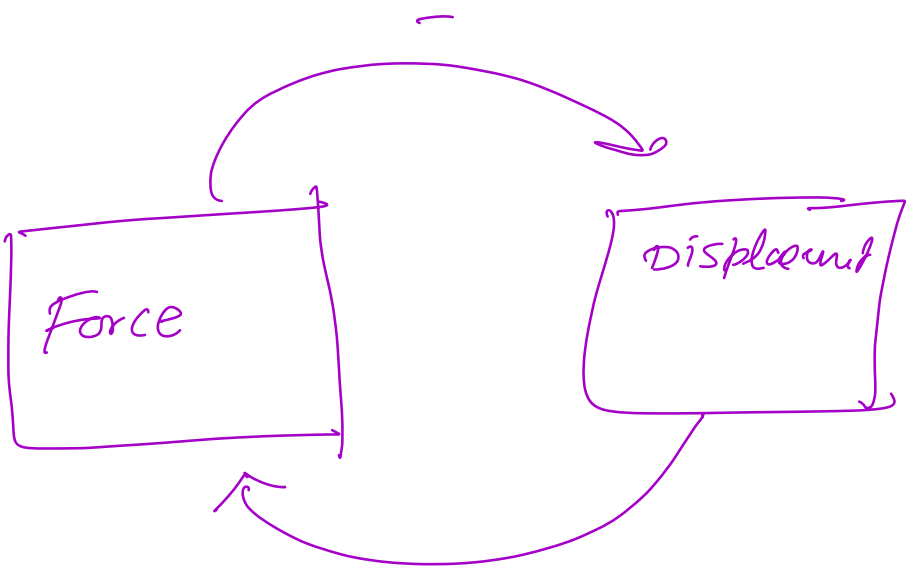
Comput x^s & F^f !

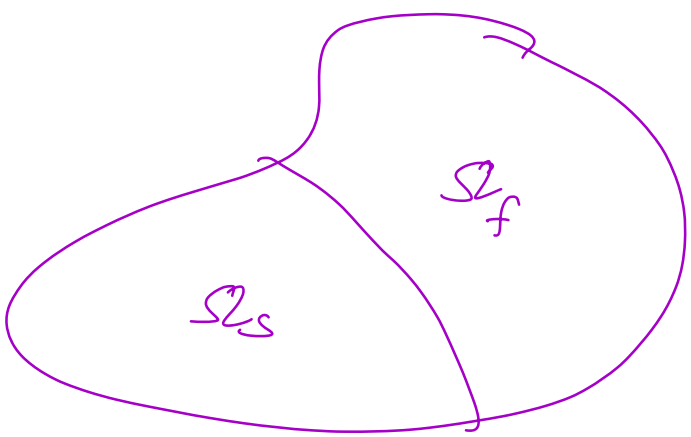


$$m\ddot{x} + cx + kx = F_f$$

$$F = P_0 \sin \omega t$$







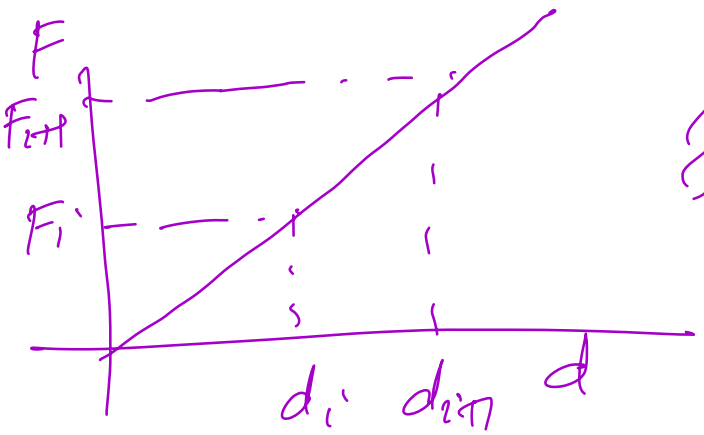
Linear: $[K]\{d\} = \{F\}$

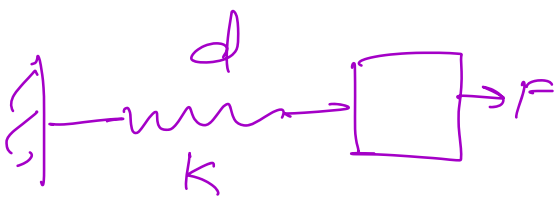
$$P(\underset{\sim}{d}) = \underset{\sim}{F}$$

$$\Rightarrow P(\underset{\sim}{d}_1 + \underset{\sim}{d}_2) = P(\underset{\sim}{d}_1) + P(\underset{\sim}{d}_2)$$

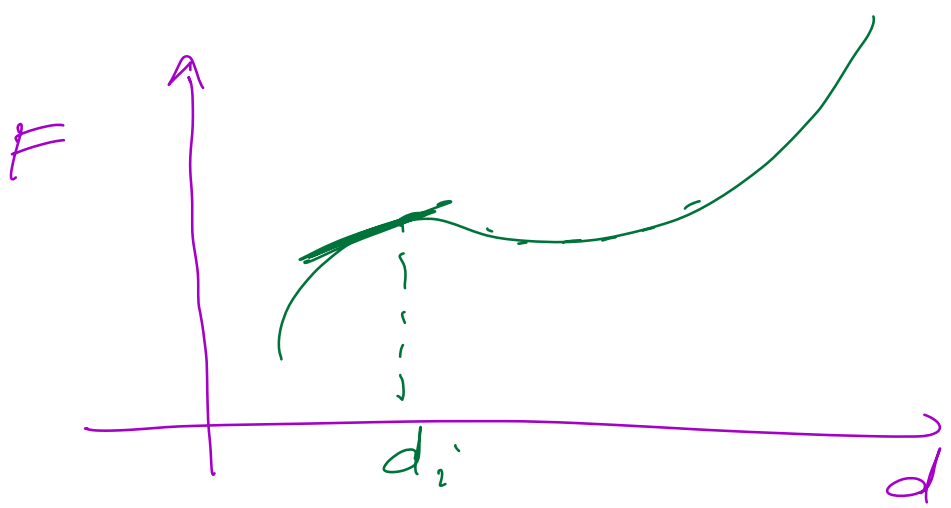
$$P(\alpha \underset{\sim}{d}) = \alpha P(\underset{\sim}{d})$$

$$= \alpha \underset{\sim}{F}$$





Nonlinear Force - displacement



$$P(d) = F$$

$$P(2d) \neq 2F$$

How to find $\underset{\sim}{d}$ for given $\underset{\sim}{F}$?

$\Rightarrow$ Incremental solution procedure for nonlinear force-displacement coupling

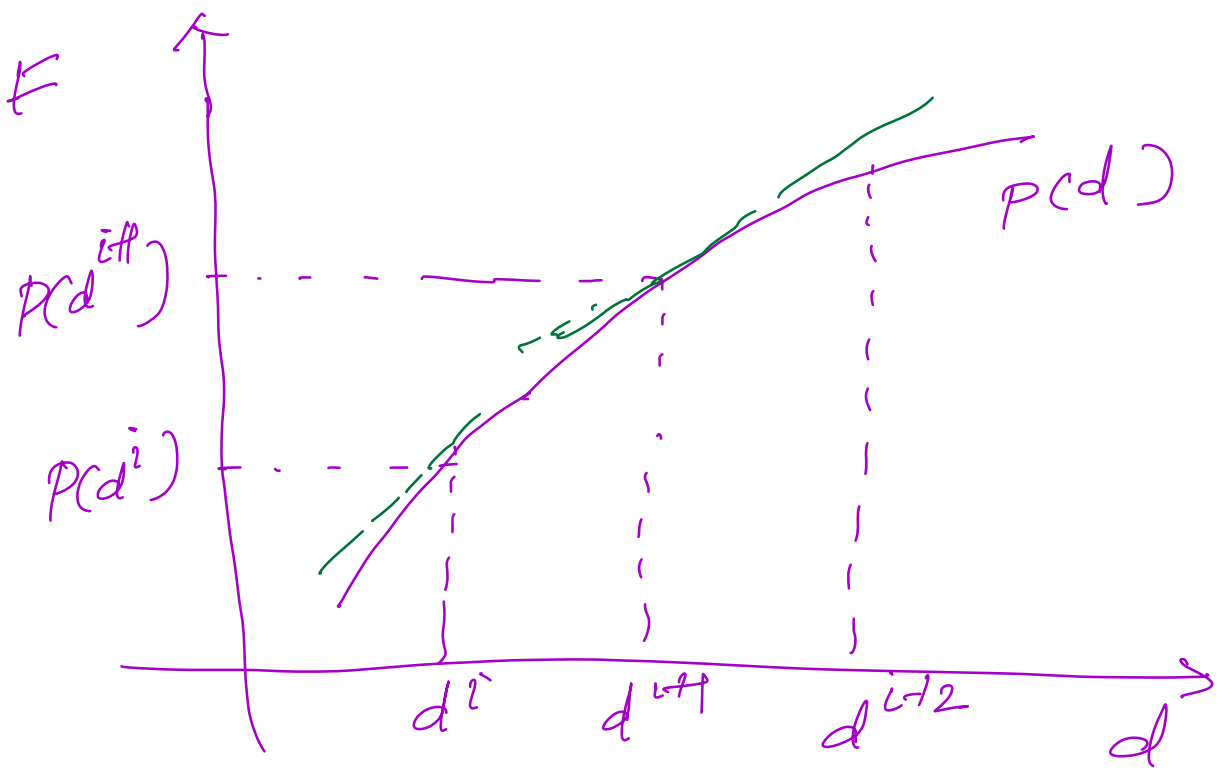
Linear system (Poisson / Laplace eq^n

$$\nabla^2 \phi = 0$$

$$\Downarrow$$

$$[A]\{\phi\} = \{b\}$$

$$\{\phi\} = [A]^{-1}\{b\}$$



Find d^{i+1}, first-order Taylor expansion

$$P(\underset{\sim}{d}^{i+1}) \simeq P(d^i) + \left(\frac{\partial \underset{\sim}{P}}{\partial \underset{\sim}{d}}\right) \cdot \Delta d^i$$

Jacobian or tangent matrix

$\leftarrow K_T^i(d^i)$

Solve for incremental solution:

$$K_T^i \Delta d^i = F - P(d^i)$$

$$F - P(d^i) = 0$$

Update:

$$d^{i+1} = d^i + \Delta d^i$$

$$| d^{i+1} - d^i | < \epsilon \quad \epsilon = 1 \times 10^{-4}$$

Some observations:

* $K_T^i(d^i)$ is not constant
* Δd^i incremental update
* RHS is residual force

$$R^i = F - P(d^i)$$

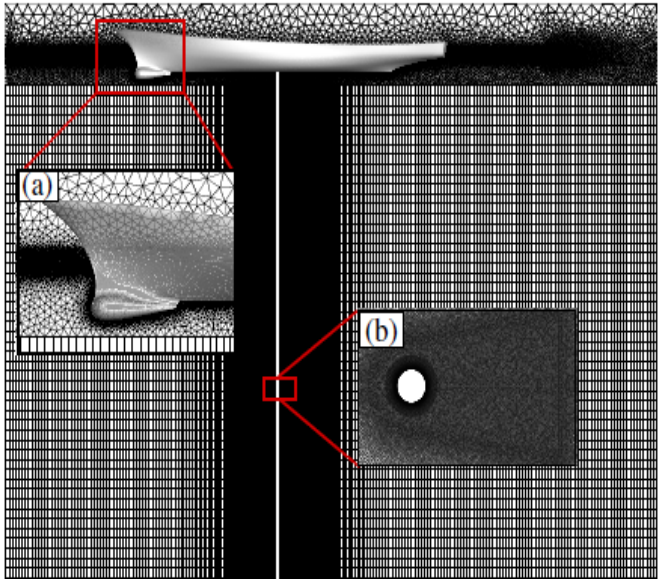
* Iteration will keep going

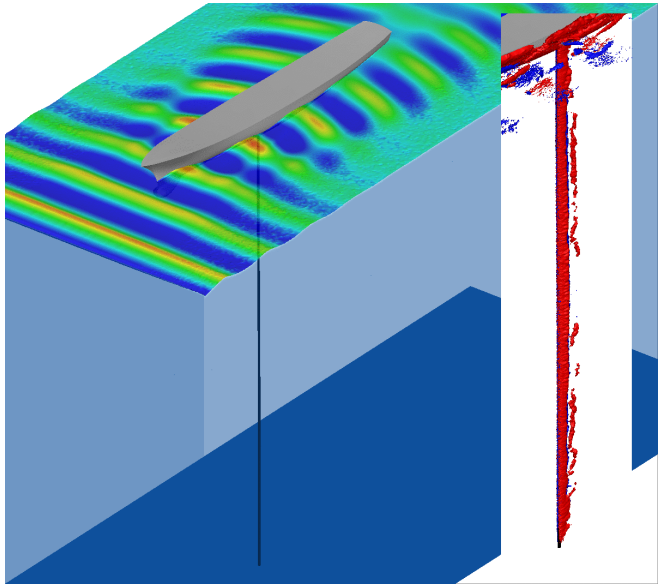
$$Conv < \text{tolerance}$$

$$Conv = \frac{\sum_{j=1}^{n} (R_j^{iter})^2}{1 + \sum_{j=1}^{n} (F_j)^2}$$

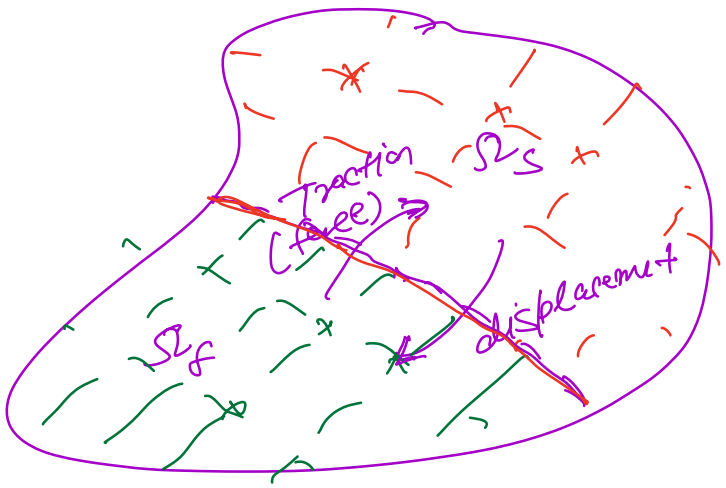
Why Variational/Finite Element Methods in CFD?

- The most general purpose and powerful method
  - Very flexible and well integrated with geometries (*CAD*)
  - For general differential equations arising from multiphysics and multiscale systems (*Physics*)
    - Variational/least action principle
    - Boundary conditions & interface treatment
    - Moving boundary problems
  - Mathematical proofs and strong foundation (*Rigorous math*)
    - Functional analysis and Hilbert spaces
    - Leray's proof on the existence of weak form of the Navier-Stokes equations.
    - Natural link with numerical linear algebra
  - Natural links with projection-based reduced order model and machine learning (*AI/Machine Learning*)
  - Business Trends in Modeling and Simulation (*Computer Aided Engineering*)
    - CAD and FEM dominate CFD business





Simflow, Computational Multiphysics Lab@UBC (cml.mech.ubc.ca)



Nonlinear Force - displacement Coupling

Fluid system:

$$\mathcal{L}\,\underset{\sim}{u}^f = \underset{\sim}{b}^f \quad \text{on} \quad \Omega^f$$

$\underset{\sim}{\sigma}(\underset{\sim}{v}^f, p^f)$

Residual or equilibrium

$$\underset{\sim}{R}^f = \mathcal{L}u^f - b^f = 0$$

(strong form)

Solid system: $\nabla \cdot \sigma + b = \rho \ddot{u}$

$$\mathcal{L}\,\underset{\sim}{u}^s = \underset{\sim}{b}^s \quad \text{on} \quad \Omega^s$$

$$\underset{\sim}{R}^s = \mathcal{L}\underset{\sim}{u}^s - \underset{\sim}{b}^s \quad \text{on} \quad \Omega^s$$

Goal: $\underset{\sim}{u}^f$, $\underset{\sim}{u}^s$ for $\Omega^f \cup \Omega^s$

FSI system challenges:

→ Reference frame

→ Interface Conditions

→ Nonlinear Coupled PDE's!

Ideas:

→ Linear vs. Nonlinear

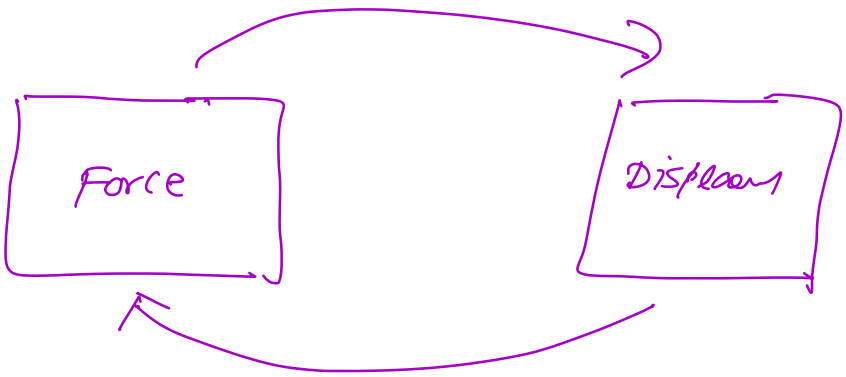
→ Strong vs. Weak/variational form

differential

→ Weak/variational ⇒ FEA (weighted residual form)

→ Galerkin vs. Petrov–Galerkin

(Standard FE)



Nonlinear system $\rightarrow P(\underset{\sim}{d}) = \underset{\sim}{F}$

$P(2\underset{\sim}{d}) \neq 2\underset{\sim}{F}$

Algorithm:

1. Tolerance : $tol = 10^{-3}$, $K = 0$, maxiter $= 10$

   initial guess $\underset{\sim}{d} = \underset{\sim}{d_0}$

2. Residual $R = F - P(d)$

3. Calculate "conv", if conv < tol stop

4. If $K >$ maxIter, error message

5. Calculate Jacobian matrix K_T

6. If the $\det(K_T) = 0$,
Error message

7. Calculate Δd

8. Update $\underset{\sim}{d} = \underset{\sim}{d} + \Delta d$

9. Set $K = K+1$

10. Go to Step 2

$\nabla \cdot \underset{\sim}{\sigma} + \underset{\sim}{b} = 0$

Example $P(\underset{\sim}{d}) = \underset{\sim}{F}$

$$P = \begin{Bmatrix} P_1 \\ P_2 \end{Bmatrix} = \begin{Bmatrix} d_1 + d_2 \\ d_1^2 + d_2^2 \end{Bmatrix} = F = \begin{Bmatrix} 3 \\ 9 \end{Bmatrix}$$

Two unknowns: d_1, d_2

Initial

$$d^0 = \begin{Bmatrix} 1 \\ 5 \end{Bmatrix}, \quad P(d)^0 = \begin{Bmatrix} 6 \\ 26 \end{Bmatrix}$$

$$K_T = \frac{\partial P}{\partial d} = \begin{bmatrix} \frac{\partial P_1}{\partial d_1} & \frac{\partial P_1}{\partial d_2} \\ \frac{\partial P_2}{\partial d_1} & \frac{\partial P_2}{\partial d_2} \end{bmatrix}$$

→ Linearization

$$= \begin{bmatrix} \frac{\partial}{\partial d_1}(d_1+d_2) & \frac{\partial}{\partial d_2}(d_1+d_2) \\ \frac{\partial}{\partial d_1}(d_1^2+d_2^2) & \frac{\partial}{\partial d_2}(d_1^2+d_2^2) \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 \\ 2d_1 & 2d_2 \end{bmatrix}$$

$$R^0 = F - P(d^0)$$

$$= \begin{Bmatrix} 3 \\ 9 \end{Bmatrix} - \begin{Bmatrix} 6 \\ 26 \end{Bmatrix} = \begin{Bmatrix} -3 \\ -17 \end{Bmatrix}$$

* Iteration 1 :

$$\begin{bmatrix} 1 & 1 \\ 2 & 10 \end{bmatrix} \begin{Bmatrix} \Delta d_1^0 \\ \Delta d_2^0 \end{Bmatrix} = \begin{Bmatrix} -3 \\ -17 \end{Bmatrix}$$

Increment: $\begin{Bmatrix} \Delta d_1^0 \\ \Delta d_2^0 \end{Bmatrix} = \begin{Bmatrix} -1.625 \\ -1.375 \end{Bmatrix}$

New $\underset{\sim}{d}^1 = \underset{\sim}{d}^0 + \Delta \underset{\sim}{d}$

$$= \begin{Bmatrix} -0.625 \\ 3.625 \end{Bmatrix}$$

Imbalance residue $\underset{\sim}{R}^1 = \underset{\sim}{F} - \underset{\sim}{P}(d^1)$

$$= \begin{Bmatrix} 0 \\ -4.531 \end{Bmatrix}$$

Out of balance error

Iteration 2:

$$\begin{bmatrix} 1 & 1 \\ -1.25 & 7.25 \end{bmatrix} \begin{Bmatrix} \Delta d_1^1 \\ \Delta d_2^2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ -4.533 \end{Bmatrix}$$

$$\begin{Bmatrix} \Delta d_1^1 \\ \Delta d_2^2 \end{Bmatrix} = \begin{Bmatrix} 0.533 \\ -0.533 \end{Bmatrix}$$

$$\underset{\sim}{d}^2 = \underset{\sim}{d}^1 + \underset{\sim}{\Delta d}^1 = \begin{Bmatrix} -0.092 \\ 3.092 \end{Bmatrix}$$

$$\underset{\sim}{R}^2 = \underset{\sim}{F} - P(\underset{\sim}{d}^2) = \begin{Bmatrix} 0 \\ -0.56 \end{Bmatrix}$$

Iteration 3: $\underset{\sim}{d}^3 = \begin{Bmatrix} -0.003 \\ 3.003 \end{Bmatrix}$

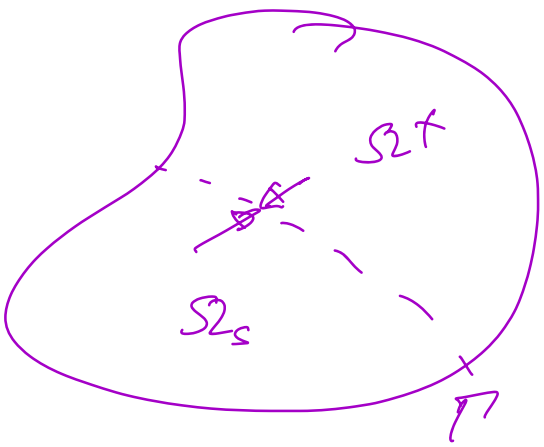
$$\underset{\sim}{R}^3 = \begin{Bmatrix} 0 \\ -0.016 \end{Bmatrix}$$

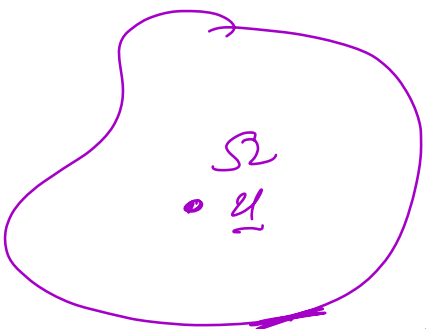
Iteration 4: $\underset{\sim}{d}^4 = \begin{Bmatrix} 0 \\ 3 \end{Bmatrix}$, $\underset{\sim}{R}^4 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$

Balance of Linear momentum

$$\nabla \cdot \underset{\sim}{\sigma} + \underset{\sim}{b} = 0$$

(Equilibrium)





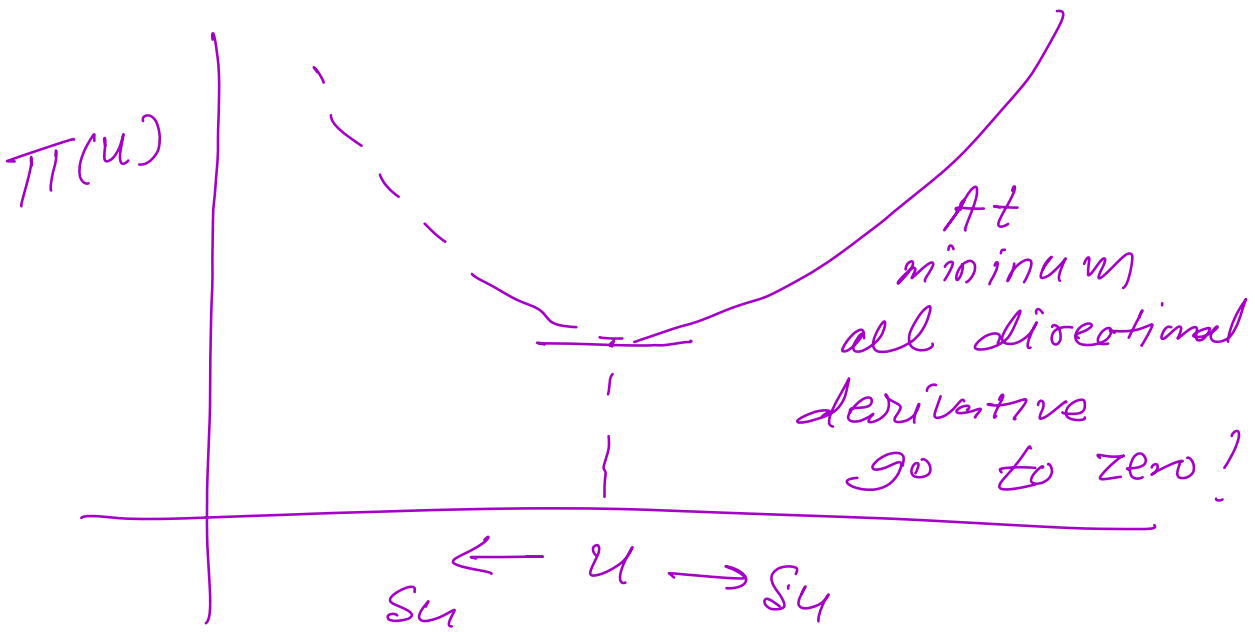
∂

Test

$$\bar{W} = \int_{\Omega} (\nabla \cdot \underset{\sim}{\sigma} + \underset{\sim}{b}) \cdot \underset{\sim}{\delta u} \, d\Omega$$

($\underset{\sim}{u}$)

$$\Pi(\underset{\sim}{u}, \psi) = \frac{d}{d\epsilon} \Pi(\underset{\sim}{u} + \epsilon \psi) \Big|_{\epsilon \to 0}$$



$\underset{\sim}{\delta u} \rightarrow$

Variational Form:

$$\bar{W} = \int_{\Omega} \underbrace{\left(\nabla \cdot \underset{\sim}{\sigma} + \underset{\sim}{b}\right)}_{\text{PDE system}} \cdot \underbrace{\underset{\sim}{\delta u}}_{= \underset{\sim}{\psi}} \qquad \underset{\sim}{\psi} = 0$$

$$\nabla \cdot \underset{\sim}{\sigma} + \underset{\sim}{b} = 0 \qquad \underset{\sim}{\sigma} = f(u)$$

$\underset{\sim}{u}$ ← known functions

$$\bar{W} = \int_{\Omega} (\nabla \cdot \underset{\sim}{\sigma}) \cdot \underset{\sim}{\psi} \, d\Omega + \int \underset{\sim}{b} \cdot \underset{\sim}{\psi} \, d\Omega$$

$\underset{\sim}{\psi}$ ↑ virtual velocity field

Integration by parts:

$$\int_{\Omega} \left[\nabla \cdot (\underset{\sim}{\sigma} \underset{\sim}{\psi}) - \underset{\sim}{\sigma} : \nabla \underset{\sim}{\psi} \right] d\Omega + \int \underset{\sim}{b} \cdot \underset{\sim}{\psi} \, d\Omega$$

Using Diversence Thm:

$$0 = \int_{\Gamma} \underset{\sim}{n} \cdot \underset{\sim}{\sigma} \cdot \underset{\sim}{\psi} \, d\Gamma - \int_{\Omega} \underset{\sim}{\sigma} : \nabla \underset{\sim}{\psi} \, d\Omega + \int_{\Omega} \underset{\sim}{b} \cdot \underset{\sim}{\psi} \, d\Omega$$

$\underset{\sim}{t}$ surface integral

body force

$$\Rightarrow \underbrace{\int_{\Omega} \underset{\sim}{\sigma} : \nabla \psi \, d\Omega}_{\text{Internal virtual work}} = \int_{\Gamma} \underset{\sim}{t} \psi \, d\Gamma + \underbrace{\int_{\Omega} \underset{\sim}{b} \psi \, d\Omega}_{\text{External virtual work}}$$

$$\int_{\Omega} \sigma_{ij} \, \psi_{ij} \, d\Omega = \int_{\Gamma} t_j \, \psi_j \, d\Gamma + \int_{\Omega} b_j \, \psi_j \, d\Omega$$

unknown

$$a\left(\underset{\sim}{u}, \psi\right) = l(\psi)$$

Energy — L.H.S = Right side vector — Load vector

$$\Rightarrow \text{stiffness} \rightarrow [K]\{u\} = \{F\}$$

$$S \approx W \sim M \sim$$

$$S \to W \to M$$

Overview

- What is FEM?
- Introduction to weak form
  - ODE example
- 1D Finite Element Examples
  - Elliptic PDE
  - Advection problem
- FEM for Steady Heat Transfer
- Examples

References and Reading on Finite Element Methods

- Chapter 5 on "Weighted Residuals Methods" of Fletcher, Computational Techniques for Fluid Dynamics. Springer, 2003.

- Löhner, Rainald. Applied Computational Fluid Dynamics Techniques: An Introduction Based on Finite Element Methods. Wiley, 2008. ISBN: 9780470519073.

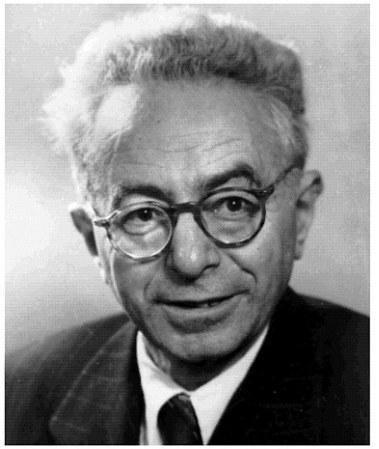
- Donea and Huerta, Finite Element Methods for Flow Problems, Wiley 2003

Background

- The finite element method constitutes a general tool for the numerical solution of partial differential equations in engineering and applied science
- All major practical advances of the method have taken place since the early 1950s in conjunction with the development of digital computers

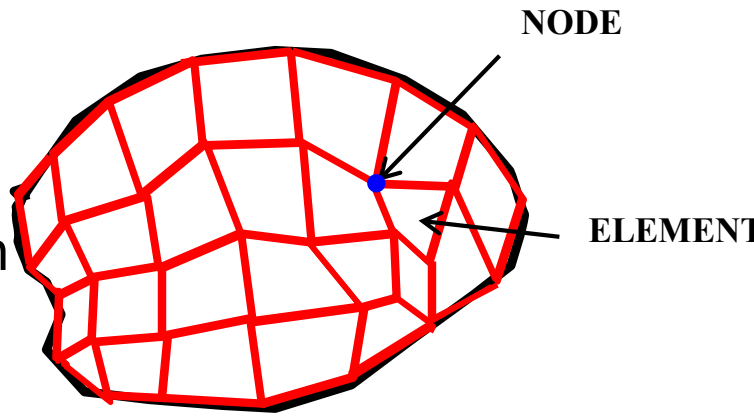
Boris Galerkin

- History:
  - Variational methods: Lord Rayleigh (1870) and W. Ritz (1909)
  - Weighted-residual approach: B. G. Galerkin (1915)
  - Lecture presented in 1941 by R. Courant to the American Association for the Advancement of Science.



General Ideas

- FEM is a numerical method to solve boundary value problems. The power of the FEM is the flexibility regarding the geometries (CAD and element topologies)
- Discretization: divide domain into "finite elements"



- General procedure
  - Find the weak formulation of the problem
  - Divide the domain into elements
  - Choose basic functions for the elements
  - Formulate a system of linear equations and solve it.
    - The linear system is typically sparse.

Basic Understanding (1)

- Approximation theory of functions:

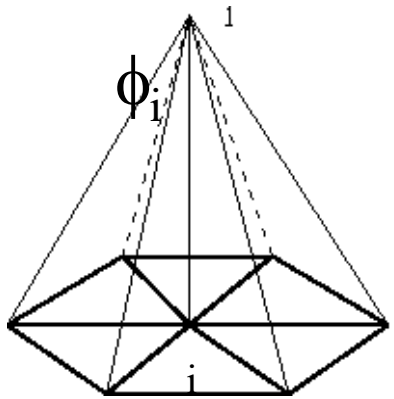
Basic Understanding (2)

- Approximation theory of functions:

Basic Understanding (3)

- Approximation theory of functions:

Galerkin Projection



- What is Projection?

Projection is a linear transformation P from a linear space $\mathbf{V}$ to itself such that

$$P^2 = P$$

Equivalently: let $\mathbf{V}$ is direct sum of subspaces $\mathbf{V}_1$ and $\mathbf{V}_2$

$$\mathbf{V} = \mathbf{V}_1 \oplus \mathbf{V}_2$$

i.e. for $\forall \mathbf{u} \in \mathbf{V}$ there are unique $\mathbf{u}_1 \in \mathbf{V}_1$ and $\mathbf{u}_2 \in \mathbf{V}_2$ s.t.

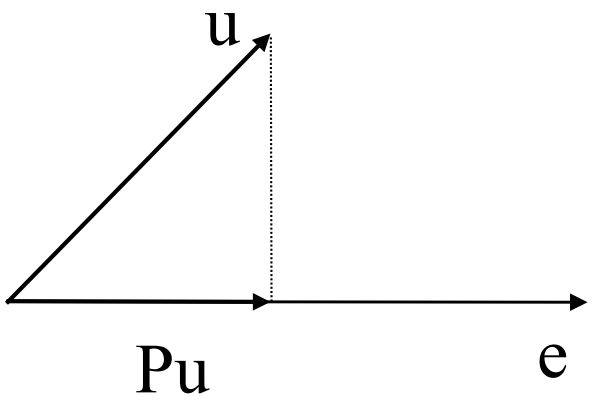
$$\mathbf{u} = \mathbf{u}_1 + \mathbf{u}_2$$

Then $P : V \to V_1$, is defined for $\forall u \in V$ as $Pu \equiv u_1$

To compute approximations $\mathbf{Pu} \approx \mathbf{u}$ where $\dim \mathbf{V}_1 << \dim \mathbf{V}$

$$\mathbf{V} = \mathbf{V}_1 \oplus \mathbf{V}_2$$

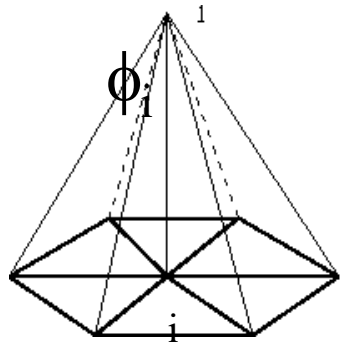
Projection in R^2



P : orthogonal projection of vector u on e

Projection in R^n / C^n

Projection in Functional Spaces

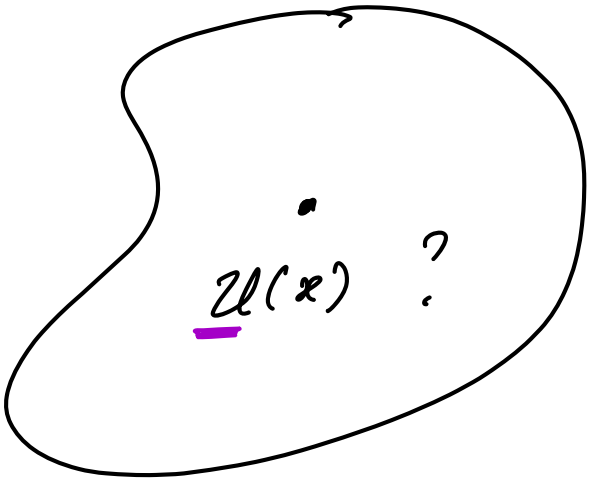


NS Eq^{n}s

Nonlinear convective

$$\rho^f \frac{\partial \underline{u}^f}{\partial t} + \rho^f (\underline{u}^f \cdot \nabla)\underline{u}^f = -\nabla p + \mu \nabla^2 \underline{u}^f + \rho \underline{g}$$

$$\nabla \cdot \underline{u} = 0$$



Approximate theory of functions.

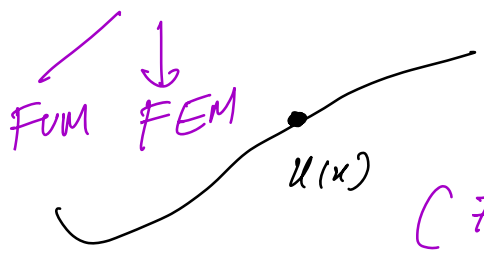
$u(x)$ in domain Ω

$$\Rightarrow u(x) \simeq u^h(x) = \sum_i N_i(x)\, q_i$$

$N_i(x)$: $(1, x, x^2, \cdots)$

Truncated Taylor series:

$$u(x) \simeq u^h(x) = a_0 + a_1 x + a_2 x^2 + \dots a_m x^m$$



$$a_j = \frac{1}{j!} \frac{d^j u}{dx^j}\Big|_{x=0}$$

(Finite difference methods)

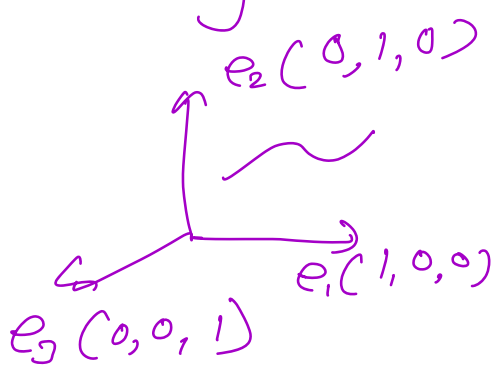
Fourier expansion: (Spectral Methods)

$$u(x) \approx u^h(x) = a_j \sin\left(\frac{j\pi x}{L}\right)$$

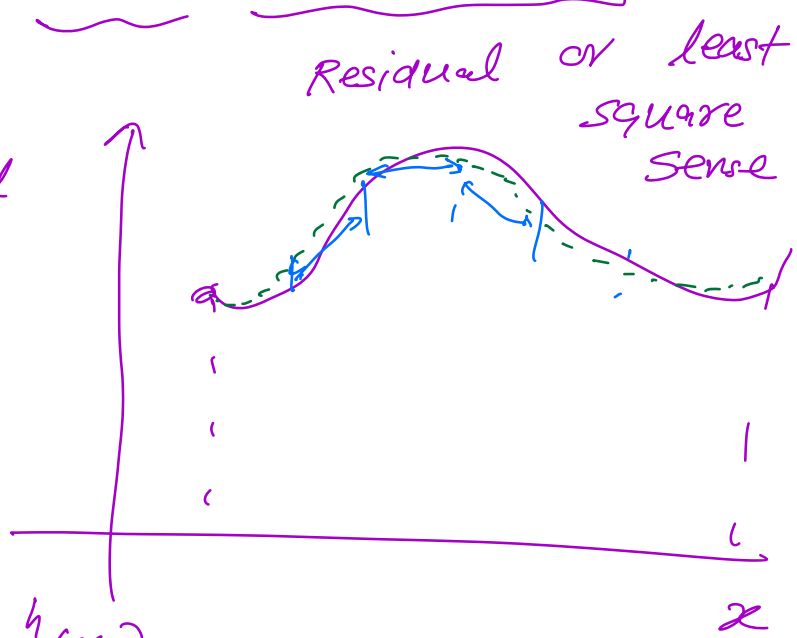
$$a_j = \frac{2}{L} \int u(x) \sin\left(\frac{j\pi x}{L}\right)$$

$$\left[1, \sin(x), \cos(x)\right]$$

(Spectral FE)



Weighted Residual Method



$$u(x) \approx u^h(x)$$

$$\Rightarrow \quad u(x) - u^h(x) = \epsilon$$

$$\Downarrow$$

$$\int_\Omega \epsilon \, W^i \, d\Omega$$

ϵ: Residue

$$u^h = \underline{N}^j a_j$$

W^i: weight function

$$= \int_\Omega (u - \underline{N}^j a_j) W^i \, d\Omega$$

Galerkin: $W^i = \underline{N}^i$

$$\int_{\Omega} N^i \left(u - N^j a_j \right) d\Omega \qquad i = 1, \ldots$$

$$\Rightarrow \int_{\Omega} N^i N^j a_j \, d\Omega = \int_{\Omega} N^i u \, d\Omega$$

$$\Rightarrow \left[\int_{\Omega} N^i N^j \, d\Omega \right] a_j = \int_{\Omega} N^i u \, d\Omega$$

Mass matrix — LHS matrix

Unknown Coff. vector

Right Side vector

$$[M]\{a\} = \{R\}$$

$$\Rightarrow \{a\} = [M^{-1}]\{R\}$$

$$u = a_i [N_i]$$

Overview of the Finite Element Method

$$(S) \Leftrightarrow (W) \approx (G) \Leftrightarrow (M)$$

Strong form	Weak form	Galerkin approx.	Matrix form

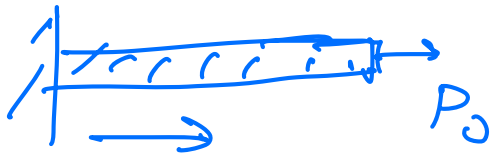
$$\rho^f \frac{\partial u^f}{\partial t} + \dots = -\nabla p + \mu \nabla^2 u + \rho g$$

$$\frac{d^2 u}{dx^2} = p_0$$

Example: Weak/Variational Form

- Consider simple 2nd order differential system (e.g., 1D steady heat transfer)

$$\frac{d^2u}{dx^2} = p_0$$ Strong Form



$$\frac{d^2u}{dx^2} - p_0 = 0$$ Residual $R=0$

$$\int_0^L \left(\frac{d^2u}{dx^2} - p_0 \right) v\,dx = 0$$ Weak Form

v is our test function

We will choose the test function later.

Example: Weak Form

Why is it "weak"?

It is a weaker statement of the problem.

A solution of the strong form will also satisfy the weak form, but not vice versa.

Analogous to "weak" and "strong" convergence:

$$\text{strong:} \quad \lim_{n\to\infty} x_n = x$$

$$\text{weak:} \quad \lim_{n\to\infty} \langle f | x_n \rangle = \langle f | x \rangle \quad \forall f$$

Example: Weak Form

Returning to the weak form:

$$\int_0^L \left(\frac{d^2u}{dx^2} - p_0 \right) v \, dx$$

test / weight functions

$$\int_0^L \frac{d^2u}{dx^2} v \, dx = \int_0^L p_0 v \, dx$$

Integrate LHS by parts:

$$= -\int_0^L \frac{du}{dx}\frac{dv}{dx} dx + \left[v(x) \frac{du}{dx} \right]_{x=0}^{x=L}$$

$$= -\int_0^L \frac{du}{dx}\frac{dv}{dx} dx + v(L)\frac{du}{dx}\Big|_{x=L} - v(0)\frac{du}{dx}\Big|_{x=0}$$

Integration by parts

$u(x)$

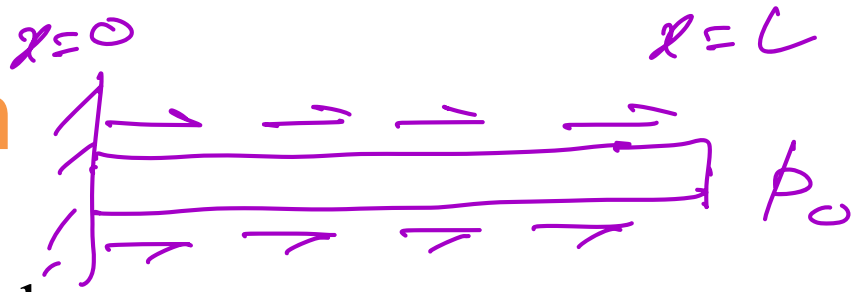
$du = u'(x)\,dx$

$v = v(x)$

$dv = v'(x)\,dx$

$$\int_a^b u' v \, dx = -\int u v' dx + [uv]_a^b$$

Example: Weak Form



Recall the boundary conditions on u and v:

$$u(0) = 0$$

$$\left(\frac{du}{dx}\right)_L = 0$$

$$v(0) = 0, \quad v(L) = 0$$

Hence,

$$-\int_0^L \frac{du}{dx}\frac{dv}{dx}\,dx$$

$$\Rightarrow \int_0^L \frac{du}{dx}\left(\frac{dv}{dx}\right)dx = \int_0^L p_0 v\,dx$$

The weak form satisfies Neumann conditions automatically!

Example: Weak Form

Why is it "variational"?

$$\int_0^L \frac{du}{dx}\frac{dv}{dx}\,dx = \int_0^L \phi_0\, v\, dx$$

Find $u \in H^1$

$$B(u, v) = F(v) \quad \forall v \in H_0^1$$

$B(u, v)$: unknown, known test function

$F(v)$: known

H_0^1: Sobolev space

Galerkin's Method

We still haven't done the "finite element method" yet, we have just restated the problem in the weak formulation.

So what makes it "finite elements"?

Solving the problem locally on elements

- Finite-dimensional approximation to an infinite-dimensional space → **Galerkin's Method**
- Basis functions formally required to be complete set of functions
- Can be seen as "residual forced to zero by being orthogonal to all basis functions"

$$\int_t \int_\Omega R(\mathbf{x}) w_i(\mathbf{x}) \, dx dt = 0, \quad i = 1, 2, \ldots, n$$

Weighted Residual Methods

- A variety of schemes arise from the definition of the weighting functions and of the choice of the shape functions
  - Galerkin method: The weighting functions are chosen to be the shape functions (the two functions are then often called basis functions or test functions)
  - Subdomain method: the weighting function is chosen to be unity in the sub-region over which it is applied
    - Non-overlapping domains often set to elements
    - Easy integration but not as accurate

$$\int_t \int_{\Omega_i} R(\mathbf{x})\, dxdt = 0, \quad i = 1, 2, ..., n$$

  - Collocation method: the weighting function is chosen to be a Dirac-delta
    - Requites no integration (like finite difference)

$$\int_t \int_{\Omega} R(\mathbf{x})\, \delta_{x_i}(\mathbf{x}) dxdt = 0, \quad i = 1, 2, ..., n$$

Petrov-Galerkin

Example: Galerkin's Method

(FEM)

Choose finite basis functions $\{\phi_i\}$

$$u(x) = \sum_{j=1}^{N} c_j \phi_j(x)$$

c_j unknowns

$$v(x) = \sum_{j=1}^{N} b_j \phi_j(x)$$

b_j some arbitrary value

Substitute into weak form:

$$\int_0^L \frac{du}{dx} \frac{dv}{dx} dx = \int_0^L \phi_0 \, v \, dx$$

Galerkin's Method

$$\int_0^L \left(\sum_{j=1}^{N+1} c_j \frac{d\phi_j}{dx} \right) \left(\sum_{i=1}^{N} b_i \frac{d\phi_i}{dx} \right) dx$$

$$= \int_0^L \phi_0 \sum_{j=1}^{N} b_j \phi_j \, dx$$

$$\Rightarrow \sum_{j=1}^{N} c_j \int_0^L \frac{d\phi_j}{dx} \frac{d\phi_i}{dx} dx = \int_0^L \phi_0 \phi_i \, dx$$

Galerkin's Method

$$\sum_{j=1}^{N} c_j \int_0^L \frac{d\phi_j}{dx}\frac{d\phi_i}{dx}\,dx = \int_0^L f_0\,\phi_i\,dx$$

Stiffness matrix (L.H.S) — R.H.S

$$K_{ij} = \int_0^L \frac{d\phi_i}{dx}\frac{d\phi_j}{dx}\,dx \quad \text{(Stiffness matrix)}$$

$$R_i = \int_0^L f_0\,\phi_i\,dx \quad \text{(Load vector)}$$

Galerkin's Method

To summarize, what have we done so far?

1) Reformulated the problem in the weak form.

2) Chosen a finite-dimensional approximation to the solution.

Recall weak form written in terms of residual:

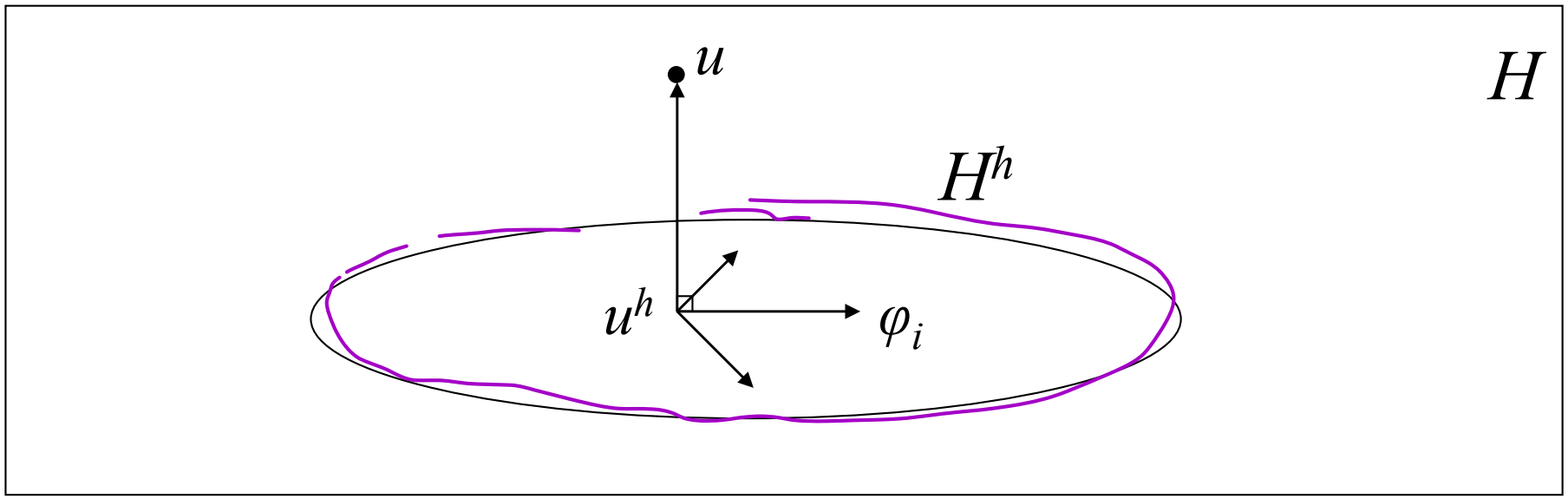
$$\int_0^L \left(\frac{d^2u}{dx^2} - p_0 \right) v dx = \int_0^L R v dx = \sum_i b_i \int_0^L R \phi_i dx = 0$$

This is an L_2 inner-product. Therefore, the residual is orthogonal to the space of basis functions. "*Orthogonality Condition*"

Orthogonality Condition

$$\int_0^L \left(\frac{d^2u}{dx^2} - p_0 \right) v dx = \int_0^L R v dx = \sum_i b_i \int_0^L R \phi_i dx = 0$$

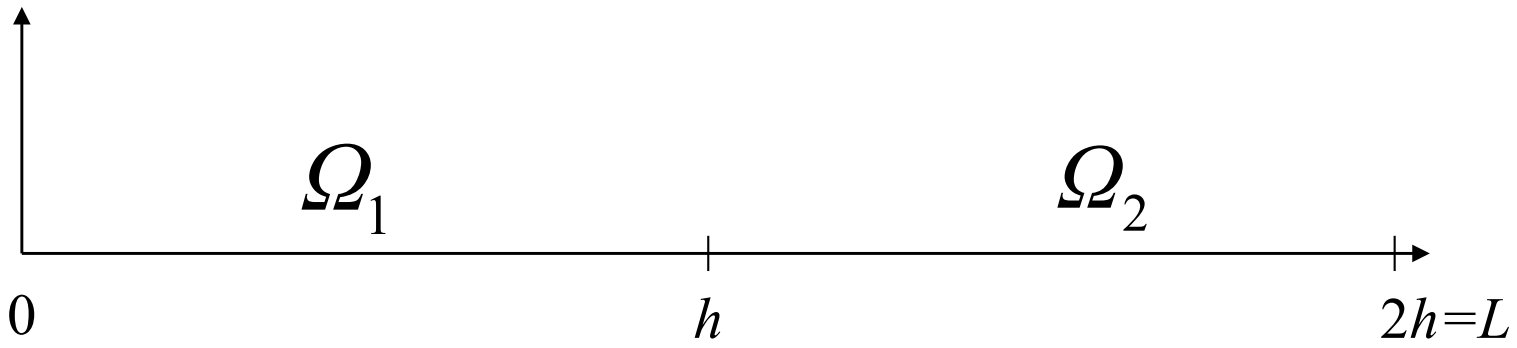
The residual is orthogonal to our space of basis functions:



Therefore, given some space of approximate functions H^h, we are finding u^h that is closest (as measured by the L_2 inner product) to the actual solution u.

Discretization and Basis Functions

Let's continue with our sample problem. Now we discretize our domain. For this example, we will discretize $x=[0, L]$ into 2 "elements".

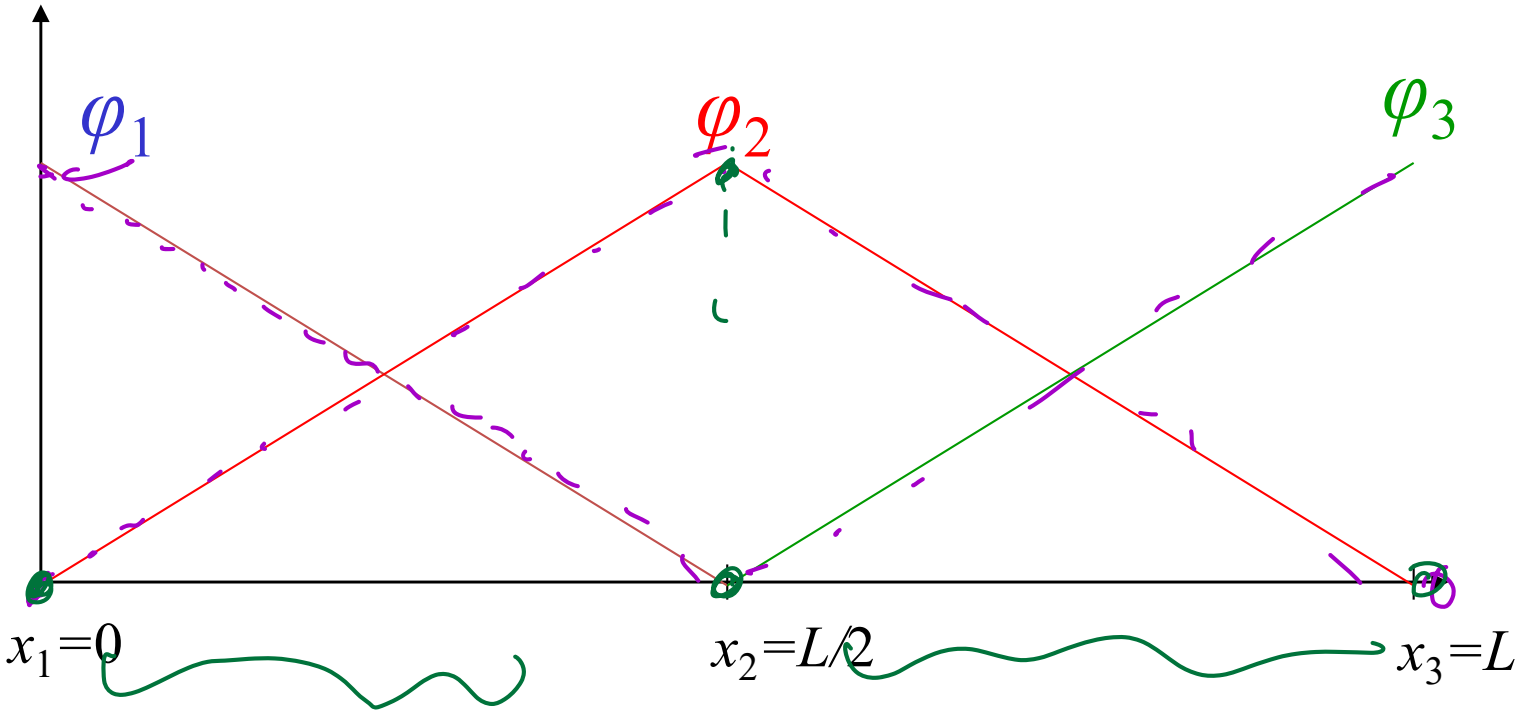


In 1-D, elements are segments. In 2-D, they are triangles, tetrads, etc. In 3-D, they are solids, such as tetrahedra. We will solve the Galerkin problem on each element.

Discretization and Basis Functions

For a set of basis functions, we can choose anything. For simplicity here, we choose piecewise linear "hat functions".

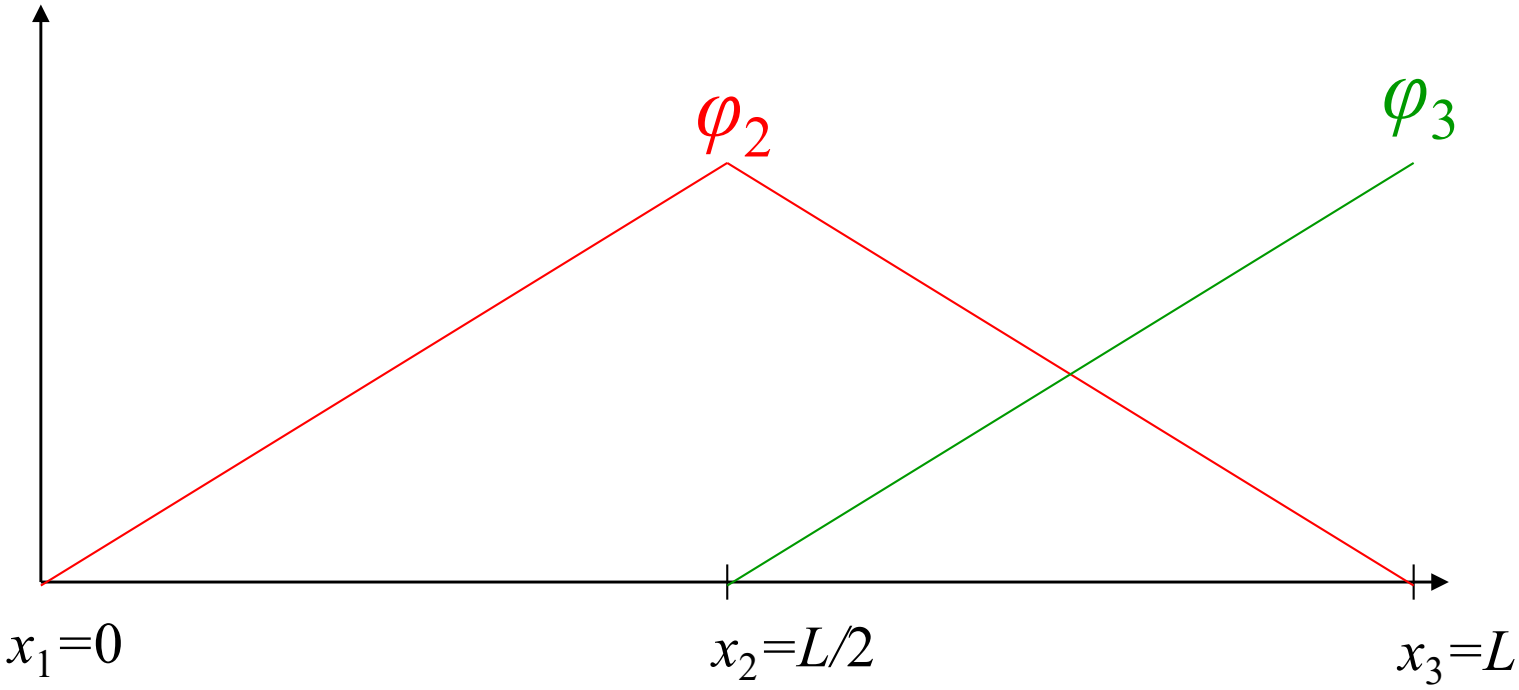
Our solution will be a linear combination of these functions.



Basis functions satisfy : $\varphi_i(x_j) = \delta_j^i \Rightarrow$ Our solution will be interpolatory. Also, they satisfy the partition of unity.

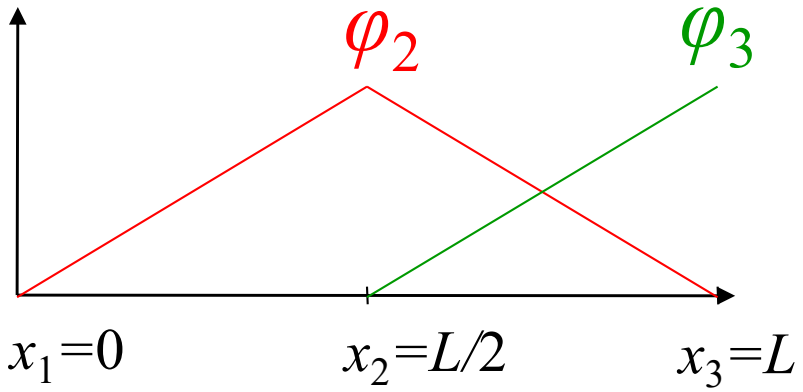
Discretization and Basis Functions

To save time, we can throw out φ_1 a priori because, since in this example $u(0)=0$, we know that the coefficent c_1 must be 0.

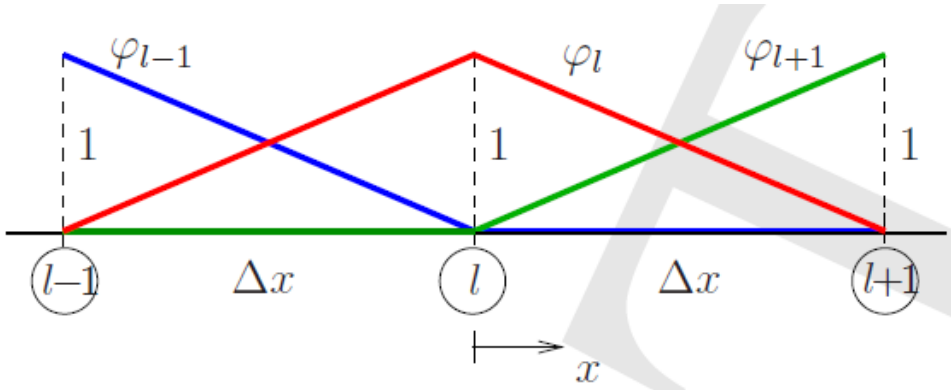


Basis Functions

$$\phi_2 = \begin{cases} \dfrac{2x}{L} & \text{if } x \in \left[0, \frac{L}{2}\right] \\ 2 - \dfrac{2x}{L} & \text{if } x \in \left[\frac{L}{2}, L\right] \\ 0 & \text{otherwise} \end{cases}$$



$$\phi_3 = \begin{cases} \dfrac{2x}{L} - 1 & \text{if } x \in \left[\frac{L}{2}, L\right] \\ 0 & \text{otherwise} \end{cases}$$



Review: Galerkin FEM

As indicated earlier, we start from the governing differential eq. (GDE) itself :

$$u''(x) = p_o \qquad (0 \le x \le L)$$

The basic idea of the Galerkin WRM is as follows : instead of satisfying the GDE at every point in the domain ($0 \le x \le L$), we will only satisfy it in a "**weighted average sense**" (weak form) :

$$\int_0^L \left(u''(x) - p_o \right) \; v(x) \; dx$$

where ***v(x)*** is **any weight function** defined on the domain **vanishing where essential boundary conditions are applied on *u*** (here, the essential BC is $u(0) = 0$ and thus we must have $v(0) = 0$. But even if we had $u(0)=5$, we would still impose that the weight function satisfy $v(0)=0$).

$$\int_0^L u''(x)\, v(x)\; dx \;=\; \int_0^L p_o \; v(x)\; dx \qquad (*)$$

To balance these requirements, let us integrate the left-hand side by parts :

$$\int_0^L u'' v \, dx \;=\; \left[u' v\right]_0^L \;-\; \int_0^L u' v' \, dx = u'(L)v(L) \;-\; u'(0)v(0) \;-\; \int_0^L u' v' \, dx$$

At this point, we introduce into *(**)* the interpolation-based approximation for *u(x)* :

$$\int_0^L \left\langle \frac{dN(x)}{dx} \right\rangle v'(x)\, dx \quad \{U\} \;=\; \int_0^L p_o \, v(x)\, dx$$

Review: Galerkin FEM

Choose our weight functions to be the shape functions: $v(x) = N_i(x)$ $(i=1, \ldots, N+1)$

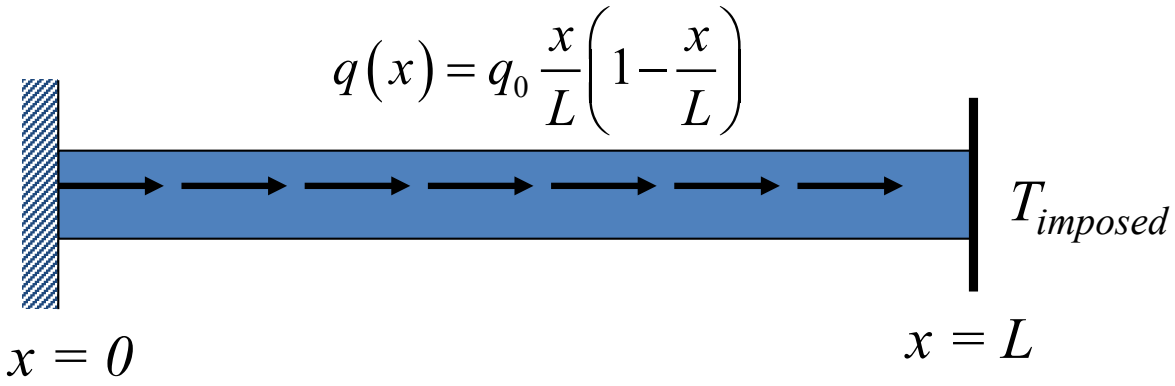
$$\Rightarrow \underbrace{\int_0^L \left\{\frac{dN(x)}{dx}\right\}\left\langle\frac{dN(x)}{dx}\right\rangle dx}_{[K]} \{U\} = \underbrace{\int_0^L p_o \{N(x)\}\, dx}_{[R]}$$

$$K_{ij} = \int_0^L \frac{dN_i(x)}{dx}\,\frac{dN_j(x)}{dx}\, dx$$

$$R_i = \int_0^L p_o\, N_i(x)\, dx$$

Implementation: FEM for 1D Steady Heat Transfer

Let us consider the following problem:



The corresponding BVP is

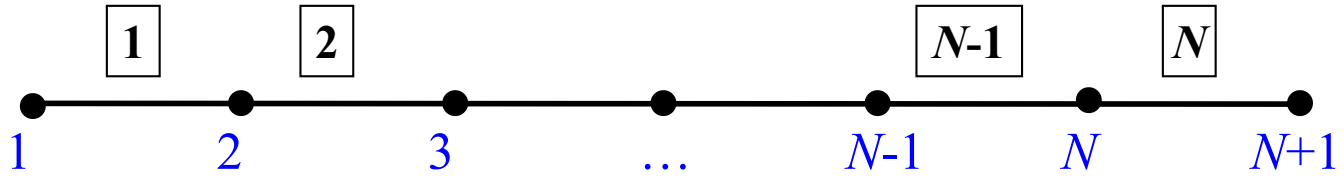
$$\text{GDE:} \quad \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) = q(x) = -q_o \frac{x}{L}\left(1 - \frac{x}{L}\right) \qquad \text{for } 0 \le x \le L$$

$$\text{BC:} \quad \begin{cases} T(0) = 0 \\ T(L) = T_{imposed} \end{cases}$$

The exact solution is given by

$$\frac{T_{ex}(x)}{q_o L^2} = \frac{1}{12}\left(\frac{x}{L}\right)^4 - \frac{1}{6}\left(\frac{x}{L}\right)^3 + \left(\eta + \frac{1}{12}\right)\frac{x}{L} \qquad \text{with} \quad \eta = \frac{T_{imposed}}{q_o L^2}$$

Let us solve this problem using N equal size 2-node axially loaded bar elements.



There are three ways to handle the imposed degrees of freedom

$$\begin{cases} T_1 = 0 \\ T_{N+1} = T_{imposed} \end{cases}$$

in the finite element code.

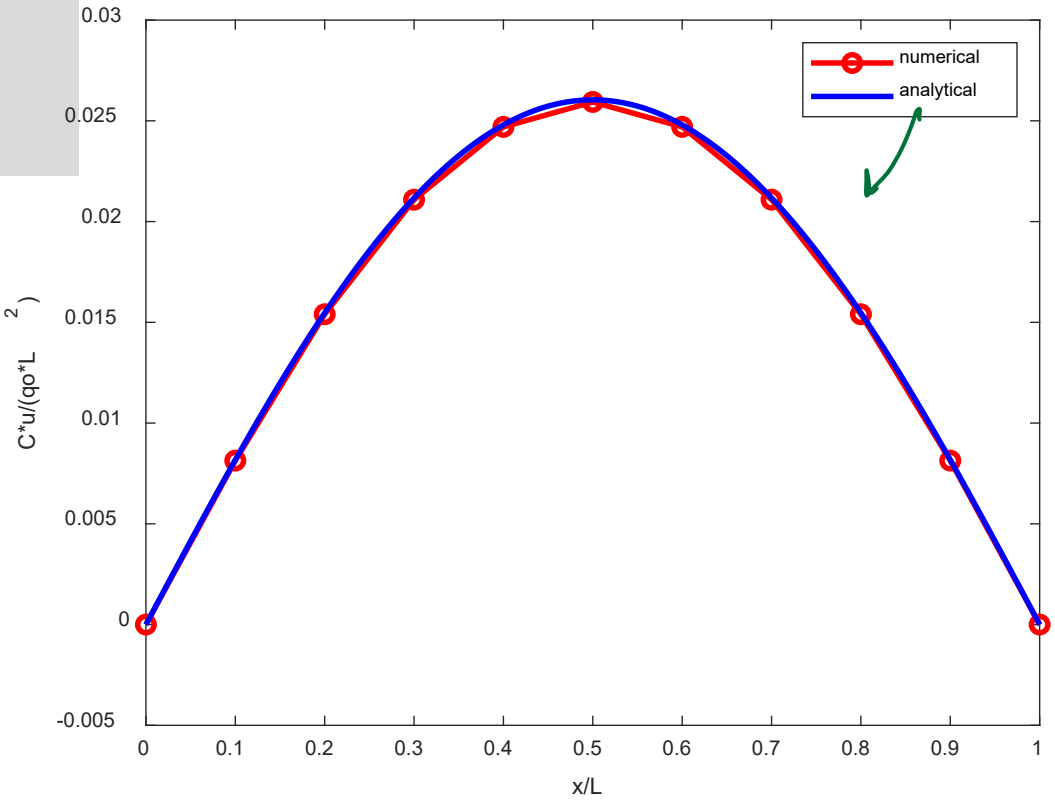
Method #1: modify corresponding rows of [*K*] and { *R* }

$$\begin{bmatrix} X & X & 0 & 0 & 0 & \dots & 0 & 0 \\ X & X & X & 0 & 0 & \dots & 0 & 0 \\ 0 & X & X & X & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & X & X & X \\ 0 & 0 & 0 & 0 & \dots & 0 & X & X \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ \dots \\ T_N \\ T_{N+1} \end{Bmatrix} = \begin{Bmatrix} X \\ X \\ X \\ \dots \\ X \\ X \\ X \end{Bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ X & X & X & 0 & 0 & \dots & 0 & 0 \\ 0 & X & X & X & 0 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & X & X & X \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ \dots \\ T_N \\ T_{N+1} \end{Bmatrix} = \begin{Bmatrix} 0 \\ X \\ X \\ X \\ X \\ T_{imposed} \end{Bmatrix}$$

```
% MECH 479: FEM for 1D Heat Transfer
%format long  % number of significant digits in screen
output
N=input(' Enter the number of elements N ...    ');
%N= 10;
L =1       % length (in m)
C = 1;     % Conductivity
qo=500;     % Distributed heat load (in N/m)
uimposed=100;  % imposed temperature at x=L (in m)
% INITIALIZATIONS
numnp=N+1;  % number of nodes
numel=N;    % number of elements
% initialize the global stiffness matrix K and global
load vector R
K=sparse(N+1,N+1);
R=zeros(N+1,1);
% create coordinate vector x(N+1) and connectivity table
lm(2,N)
x=zeros(N+1,1);
x(1)=0;
for i=1:N
  x(i+1)=x(i)+L/N;
end
lm=zeros(2,N);
for i=1:N
  lm(1,i)=i;
  lm(2,i)=i+1;
end
```

```
% FINITE ELEMENT SOLUTION
% loop over the N elements
for i=1:N
%   initialize local stiffness matrix k(2,2) and local
load vector r(2)
  k=zeros(2,2);
  r=zeros(2,1);
%   get element length
  l=abs(x(i+1)-x(i));
%   get load on the element (= value of p(x) at the
center of the element)
  xm=(x(i)+x(i+1))/2;     % location of element center
  qm=qo*xm*(L-xm)/L^2;
%   compute local km and local lv
  k(1,1)=C/l;
  k(1,2)=-k(1,1);
  k(2,1)=k(1,2);
  k(2,2)=k(1,1);
  r(1)=qm*l/2;
  r(2)=qm*l/2;
% assemble into K and R
  n1=lm(1,i);
  n2=lm(2,i);
  K(n1,n1)=K(n1,n1)+k(1,1);
  K(n1,n2)=K(n1,n2)+k(1,2);
  K(n2,n1)=K(n2,n1)+k(2,1);
  K(n2,n2)=K(n2,n2)+k(2,2);
  R(n1)=R(n1)+r(1);
  R(n2)=R(n2)+r(2);
end   % for loop on elements
% account for boundary conditions
K(1,1)=1;
K(N+1,N+1)=1;
for i=2:N+1
  K(1,i)=0;
  K(N+1,i-1)=0;
end
R(1)=0;
R(N+1)=uimposed;
% solve linear system
u=zeros(N+1,1);
u=K\R;
```

```
% POSTPROCESSING
% compute exact solution
uex1=zeros(N+1,1);      % used for pointwise comparison
xx=zeros(501,1);        % used for plotting exact solution
xx=0:L/500:L;
uex2=zeros(501,1);      % used for plotting exact solution
eta=C*uimposed/(qo*L^2);
uex1=qo*L^2/(C)*(((x/L).^4)/12-
((x/L).^3)/6+(1/12+eta)*(x/L));
uex2=qo*L^2/(C)*(((xx/L).^4)/12-
((xx/L).^3)/6+(1/12+eta)*(xx/L));
% print and plot temperature solution
sprintf('%s','nodal displacements')
iii=1:1:numnp;
aux=[iii;x';u';uex1'];
sprintf(' node %d    x = %8.3f    u = %15.5f    uex =
%15.5f\n',aux)
plot(x/L,C*u/(qo*L^2),'ro-',xx/L,C*uex2/(qo*L^2),'b-
','linewidth',2)
xlabel 'x/L'
ylabel 'C*u/(qo*L^2)
```



Method #2: partitioning of [K] and { R }

The global degree-of-freedom vector $< U >$ is composed of free (to be determined) and prescribed (known) degrees of freedom. Let us organize $< U >$ as

$$\{ U \} = \begin{Bmatrix} U^f \\ U^p \end{Bmatrix}$$

where $\{ U^f \}$ is the (column) vector regrouping all free dofs, and $\{ U^p \}$ is the (column) vector regrouping all prescribed dofs.

The linear system $[K] \{ U \} = \{ R \}$ can then be rewritten as

$$\begin{bmatrix} K^{ff} & K^{fp} \\ K^{pf} & K^{pp} \end{bmatrix} \begin{Bmatrix} U^f \\ U^p \end{Bmatrix} = \begin{Bmatrix} R^f \\ R^p \end{Bmatrix}$$

We thus solve for the unknown quantities $< D^f >$ and $< R^p >$ as follows:

1) solve for $< U^f >$ through $\left[K^{ff} \right] \left\{ U^f \right\} = \left\{ R^f \right\} - \left[K^{fp} \right] \left\{ U^p \right\}$

2) solve for $< R^p >$ through $\left\{ R^p \right\} = \left[K^{pf} \right] \left\{ U^f \right\} + \left[K^{pp} \right] \left\{ U^p \right\}$

```
% INITIALIZATIONS
numnp=N+1;  % number of nodes
numel=N;    % number of elements
numimp=2;             % number of imposed dof
numeq=numnp-numimp;   % number of equations
idof=zeros(numnp,1);  % vector with equation numbers
idof(1)=-1;           % negative for first and last nodes
idof(numnp)=-2;
for i=2:numnp-1
  idof(i)=i-1;        % positive for other nodes
end
%initialize the global stiffness matrices Kff, Kpf and Kpp
%    and global load vectors Rf and Rp
Kff=sparse(numeq,numeq);
Kpf=sparse(numimp,numeq);
Kpp=sparse(numimp,numimp);
Rf=zeros(numeq,1);
Rp=zeros(numimp,1);
%initialize the global dof vectors Uf(numeq) and Up(numimp)
Uf=zeros(numeq,1);
Up=zeros(numimp,1);
Up(1)=0.;
Up(2)=uimposed;
```

```
%
% start of FEA
%
% loop over the N elements
for i=1:N
%   assemble into Kff, Kpf, Kpp and Rf, Rp
    iaux=zeros(2,1);
    iaux(1)=idof(n1);
    iaux(2)=idof(n2);
    for jj=1:2
        nj=iaux(jj);
        for kk=1:2
            nk=iaux(kk);
            if nj>0 & nk>0
                    Kff(nj,nk)=Kff(nj,nk)+k(jj,kk);
            elseif nj<0 & nk>0
                    Kpf(-nj,nk)=Kpf(-nj,nk)+k(jj,kk);
            elseif nj<0 & nk<0
                Kpp(-nj,-nk)=Kpp(-nj,-nk)+k(jj,kk);
            end
        end
        if nj>0
            Rf(nj)=Rf(nj)+r(jj);
        else
            Rp(-nj)=Rp(-nj)+r(jj);
        end
    end
end         % end of loop over elements
% account for displacement boundary conditions (stored in Up)
Rf=Rf-(Kpf)'*Up;
% solve linear system
Uf=Kff\Rf;
% POSTPROCESSING
% 1) rebuild, display and plot the global displacement vector
U(numnp)
U=zeros(numnp,1);
for i=1:numnp
    if idof(i) > 0
            U(i)=Uf(idof(i));
    else
            U(i)=Up(-idof(i));
    end
end
```

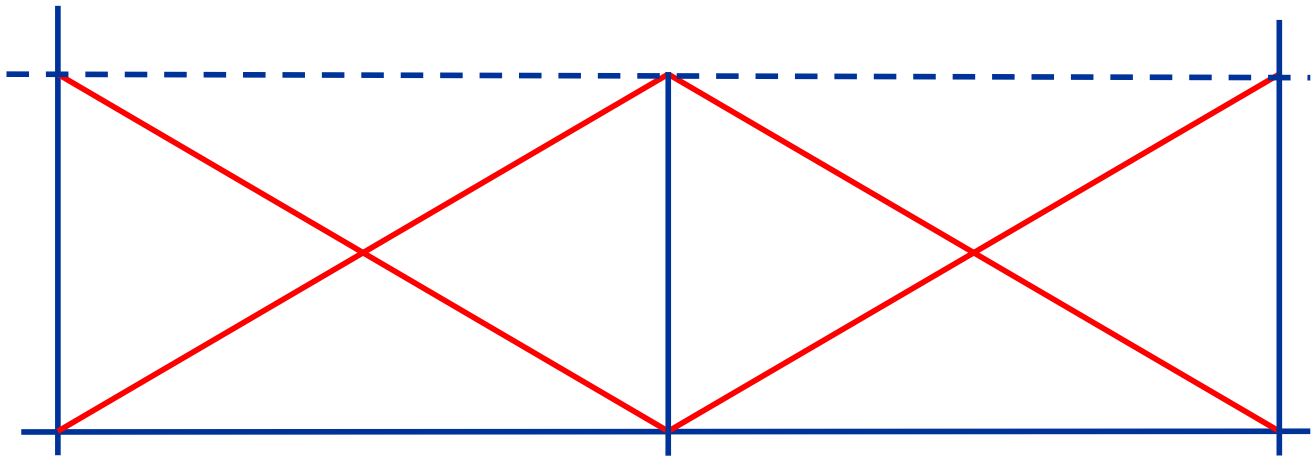
Finite Element vs. Finite Difference

- For finite element methods, we expand the solution in terms of a known basis function defined locally over each element

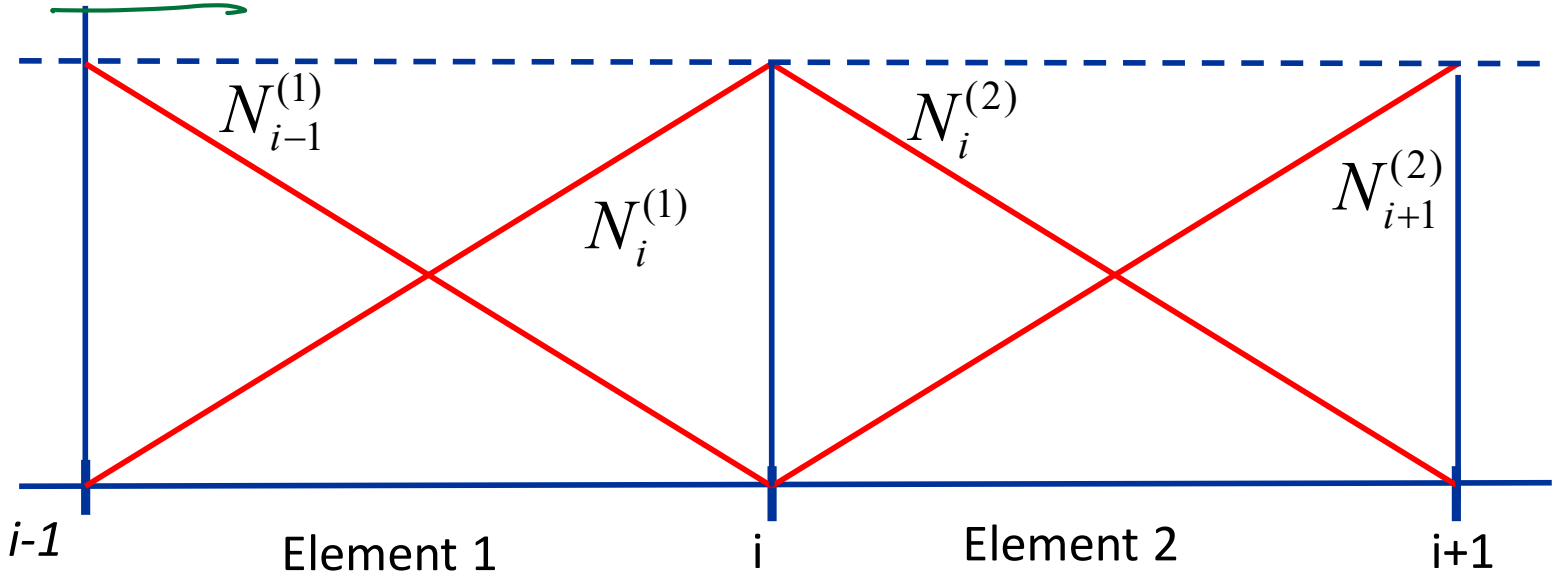
$$u(x) = \sum u_a N_a(x),$$

$$\frac{\partial u}{\partial x} = \sum u_a \frac{\partial N_a}{\partial x}$$

- Consider simple linear one-dimensional element and shape functions.



FEM vs. FDM



For element 1:

$$N_i^{(1)} = \frac{x - x_{i-1}}{\Delta x_i}; \quad N_{i-1}^{(1)} = \frac{x_i - x}{\Delta x_i}; \quad \frac{\partial N_i^{(1)}}{\partial x} = \frac{1}{\Delta x_i}; \quad \frac{\partial N_{i-1}^{(1)}}{\partial x} = \frac{-1}{\Delta x_i}$$

For element 2:

$$N_i^{(2)} = \frac{x_{i+1} - x}{\Delta x_{i+1}}; \quad N_{i+1}^{(2)} = \frac{x - x_i}{\Delta x_{i+1}}; \quad \frac{\partial N_i^{(2)}}{\partial x} = -\frac{1}{\Delta x_{i+1}}; \quad \frac{\partial N_{i+1}^{(2)}}{\partial x} = \frac{1}{\Delta x_{i+1}}$$

Example: Elliptic PDE Problem

Example: Elliptic PDE Problem (Weak Form)

Example: Elliptic PDE Problem (Galerkin Method)

Example: Elliptic PDE Problem (Galerkin Method)

Example 2: Unsteady Problem

- Finite Element Approximation of an unsteady problem

Advection problem:

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0$$

Weak formulation

$$\int_{\Omega} W\frac{\partial u}{\partial t}dx + \int_{\Omega} c\frac{\partial u}{\partial x}Wdx = 0$$

Introduce the finite element representation

$$u(x) = \sum_a u_a N_a(x)$$

Example: Unsteady PDE Problem

Some Points

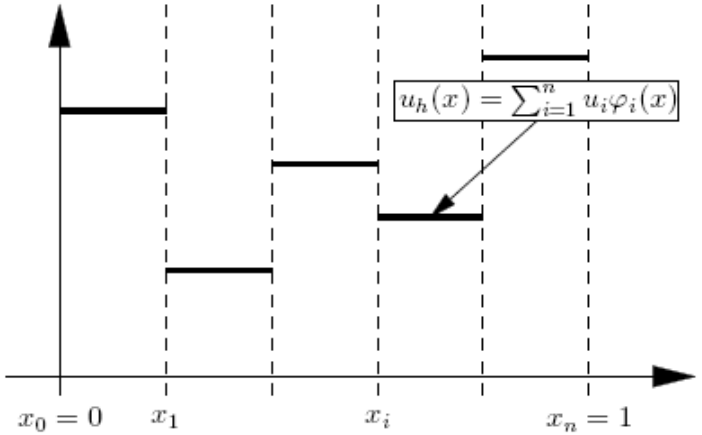
- A few additional points:
  - The mass matrix makes it complex to use simple explicit time integrators. This can be overcome by “lumping” the matrix
  - The integration over the elements is often done numerically using Gaussian integration where the function is evaluated at optimum points
  - For incompressible flows, the pressure and the velocity are often represented by elements of different order

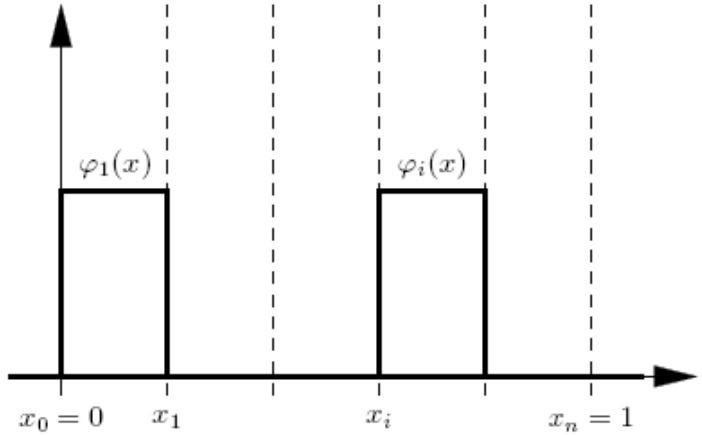
The Finite Volume Method is Galerkin FE

- Consider 1-D conservation law with following finite volume representation

$$\frac{\partial u}{\partial t}+\frac{\partial f(u)}{\partial x}=0$$

$$u_h(x)=\sum_{i=1}^{n}u_i\varphi_i(x) \qquad \varphi_i(x)=\begin{cases}1 & x_{i-1}<x<x_i\\ 0 & otherwise\end{cases}$$





Galerkin formulation: *Find* $u_h \in V_h$ *such that*

$$\int_0^1 \frac{\partial u_h}{\partial t} v dx + \int_0^1 \frac{\partial f(u_h)}{\partial x} v dx = 0 \quad \forall v \in V_h$$

$$\text{Set } v = \varphi_i = \begin{cases} 1 & x \in [x_{i-1}, x_i] \\ 0 & \text{otherwise} \end{cases}$$

Integral / weak form

$$\int_{x_{i-1}}^{x_i} \frac{\partial u_h}{\partial t} dx + \int_{x_{i-1}}^{x_i} \frac{\partial f(u_h)}{\partial x} dx = 0 \Leftrightarrow \int_{x_{i-1}}^{x_i} \frac{\partial u_h}{\partial t} dx + \left[f\left(u_h(x)\right) \right]_{x_{i-1}}^{x_i}$$

Since u *is discontinous at* x_i *and* x_{i-1}, *employ a numerical flux function* $F(u_R, u_L)$ *to obtain*:

$$h \frac{\partial u_i}{\partial t} + F(u_{i+1}, u_i) - F(u_i, u_{i-1}) = 0$$

The above formula is standard FVM on a unform grid

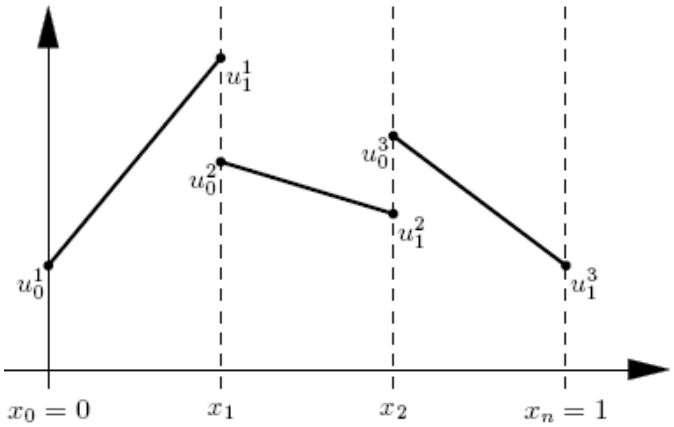
Discontinuous Galerkin = Finite Volume + Finite Element

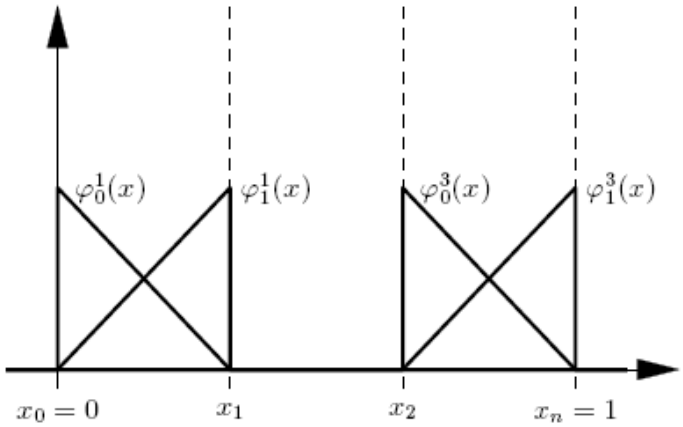
- Let us extend Galerkin FE to higher order degree
- Nodal representation with values

$$u_h(x) = \sum_{i=1}^{n}\sum_{k=0}^{p} u_i^k \varphi_i^k(x)$$

$$Set\ v = \varphi_i^k$$

$$\int_{x_{k-1}}^{x_k} \frac{\partial u_h}{\partial t}\varphi_i^k dx + \left[\frac{\partial f(u_h)}{\partial x}\varphi_i^k\right]_{x_{k-1}}^{x_k} - \int_{x_{k-1}}^{x_k} f(u_h(x))\frac{d\varphi_i^k}{dx}dx = 0$$





Finite Element in a Nutshell

- The finite element method
  - is a special form of the Weighted Residual Method
  - is based on the interpolation (using shape functions) of nodal degrees of freedom
  - contains 6 basic steps
    1) discretization : mesh, nodes, element
    2) choose appropriate basis functions
    3) form properties of a generic element : local stiffness matrix [k] and load vector {r}
    4) assembling : global load matrix [K] and load vector {R}
    5) apply the natural (Neumann) and essential (Dirichlet) boundary conditions
    6) solve the linear system [K] {U} = {R}
    7) post-processing : deformed shape, element stresses and support reactions

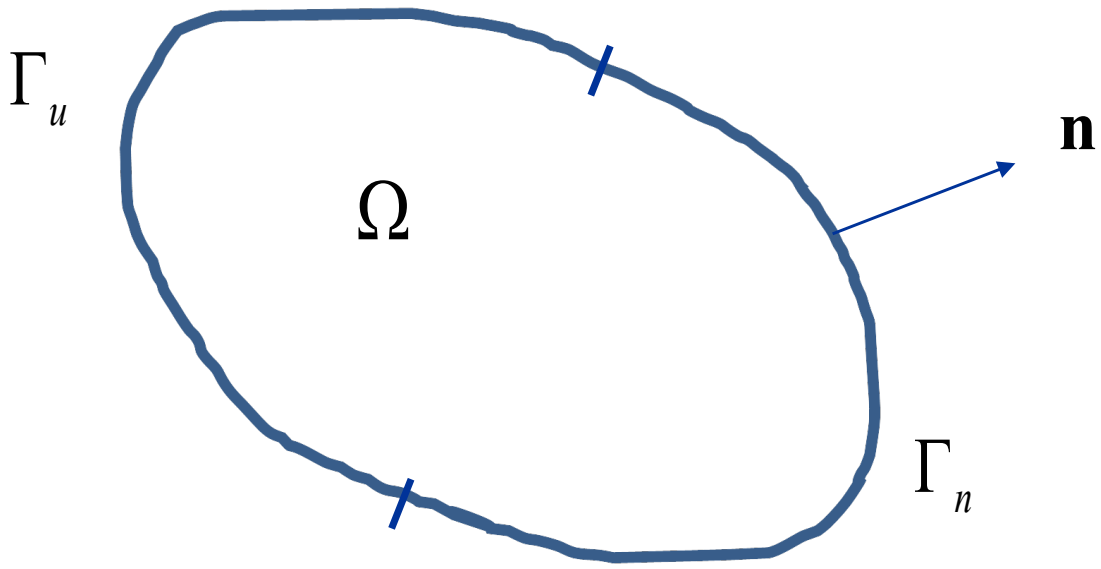
Summary

- Finite Difference Methods: based on a discretization of the differential form of the conservation equations
  - Solution domain divided in a grid of discrete points or nodes
  - PDE replaced by finite-divided differences = "point-wise" approximation
  - Harder to apply to complex geometries

- Finite Volume Methods:
  - Based on a discretization of the integral forms of the conservation (integral) equations:
  - Grid generation: divide domain into set of discrete control volumes (CVs) or finite volume "cells" in ad-hoc manner
  - Discretize integral equation
  - Solve the resultant discrete volume/flux equations

- Finite Element Methods:
  - Based on reformulation of PDEs into minimization problem, pre-assuming piecewise shape of solution over finite element
  - Grid generation: divide the domain into simply shaped regions or "elements"
  - Develop approximate solution of the PDE for each of these elements
  - Link together or assemble these individual element solutions, ensuring some continuity at inter-element boundaries
    - ◇ PDE is satisfied in piecewise fashion

STEADY HEAT TRANSFER PROBLEM



2D Steady Heat Transfer Problem

Our objective is to derive the FE formulation for the Poisson problem:

$$\kappa_{xx}\frac{\partial^2 T}{\partial x^2}+\kappa_{yy}\frac{\partial^2 T}{\partial y^2}+Q(x,y)=0 \qquad on \quad \Omega \tag{1}$$

where T(x,y) is the (unknown) temperature field
Q(x,y) is the (known) distributed heat source
κ_{xx} and κ_{yy} are the thermal conductivities in the x- and y-directions, respectively
Ω is a 2-D domain of arbitrary shape

We can rewrite the governing differential equation (GDE) in a matrix form as

$$\langle\nabla\rangle\,[\Lambda]\,\{\nabla T(x,y)\}+Q(x,y)=0 \qquad \text{on } \Omega$$

with

$$[\Lambda]=\text{conductivity matrix}=\begin{bmatrix}\kappa_{xx} & 0\\ 0 & \kappa_{yy}\end{bmatrix}$$

$$\langle\nabla\rangle=\text{gradient operator vector}=\left\langle\frac{\partial}{\partial x}\quad\frac{\partial}{\partial y}\right\rangle$$

Heat Transfer Problem

We will consider two types of boundary conditions :

$$-(\text{essential}) \quad : \quad T = T^* \quad \text{along} \quad \partial\Omega_T \tag{2}$$

$$-(\text{natural}) \quad : \quad n_x \kappa_{xx} \frac{\partial T}{\partial x} + n_y \kappa_{yy} \frac{\partial T}{\partial y} + q_n^* = 0 \quad \text{along} \quad \partial\Omega_q = \partial\Omega \setminus \partial\Omega_T \tag{3}$$

where T_n^* is the **imposed** temperature
q_n^* is the **imposed** outward heat flux
n = (n_x, n_y) is the unit outward normal

Note #1 : we can rewrite the natural boundary condition in a matrix form as

$$\langle n \rangle [\Lambda] \{\nabla T\} + q_n^* = 0 \qquad \text{with} \quad \langle n \rangle = \begin{pmatrix} n_x & n_y \end{pmatrix}$$

Note #2 : Although the problem above is written in terms of temperature, the Poisson prob
described by (1)-(3) is very general and is applicable to many other problems.
For examples:

- **inviscid, incompressible fluid flow** : T = stream function ψ and Q = 0

Galerkin Weighted Residual

For this approach, we start from the differential equation and BC:

$$\begin{cases} \langle \nabla \rangle [\Lambda] \{\nabla T\} + Q = 0 & \text{on} \quad \Omega \\ T = T^* & \text{along} \quad \partial\Omega_T \\ \langle n \rangle [\Lambda] \{\nabla T\} + q_n^* = 0 & \text{along} \quad \partial\Omega_q \end{cases}$$

Satisfy the DEQ on an "average sense" :

$$\int_{\Omega} w(x,y) \left(\langle \nabla \rangle [\Lambda] \{\nabla T\} + Q \right) \, d\Omega = 0$$

where *w(x,y)* is the weighting function. We then "balance" the continuity requirements on *w* and *T* by applying Green's theorem (the 2-D equivalent of integration by parts)

$$\int_S f \, \nabla . \nabla g \, dS = -\int_S \nabla f . \nabla g \, dS + \int_{\partial S} f \, \boldsymbol{n} . \nabla g \, d\partial S$$

or, in a component form, (with α= 1, 2 summed when repeated)

$$\int_S f \frac{\partial}{\partial x_\alpha} \left(\frac{\partial g}{\partial x_\alpha} \right) dS = -\int_S \frac{\partial f}{\partial x_\alpha} \frac{\partial g}{\partial x_\alpha} dS + \int_{\partial S} f \, n_\alpha \frac{\partial g}{x_\alpha} d\partial S$$

Here $f \equiv w$ and $\nabla g \equiv [\Lambda]\{\nabla T\}$

$$\Rightarrow \int_S \left(\langle \nabla w \rangle [\Lambda] \{\nabla T\} - w Q \right) dS - \int_{\partial\Omega_q} w \underbrace{\langle n \rangle [\Lambda] \{\nabla T\}}_{-q_n^*} d\partial\Omega_q - \cancel{\int_{\partial\Omega_T} w \langle n \rangle [\Lambda] \{\nabla T\} d\partial\Omega_T} = 0$$

$w = 0$

Then substitute

$$T(x,y) \approx \tilde{T}(x,y) = \langle N(x,y) \rangle \{D\} = \langle D \rangle \{N(x,y)\}$$

$$w(x,y) = N_i(x,y) \qquad (i = 1, \ldots, N)$$

to find

$$[K]\{D\} = \{R\}$$

with

$$[K_{ij}] = \int_\Omega \left\langle \frac{\partial N_i}{\partial x} \; \frac{\partial N_i}{\partial y} \right\rangle \begin{bmatrix} \kappa_{xx} & 0 \\ 0 & \kappa_{yy} \end{bmatrix} \begin{Bmatrix} \frac{\partial N_j}{\partial x} \\ \frac{\partial N_j}{\partial y} \end{Bmatrix} d\Omega$$

$$\{R_i\} = \int_\Omega Q \, N_i \, d\Omega - \int_{\partial\Omega_q} q_n^* \, N_i \, d\partial\Omega_q$$

Interpolation

Our objective in this section is to derive the expression of the local stiffness matrix [k] and t
local load vector {r} for a generic 3-node triangular element

Now we substitute the interpolation-based approximation of the temperature field

$$T(x,y) \approx \tilde{T}(x,y) = \langle N(x,y) \rangle \{d\}$$

where

$\langle d \rangle = \langle T_a \quad T_b \quad T_c \rangle$ contains the three **nodal temperatures** (to be solved for)

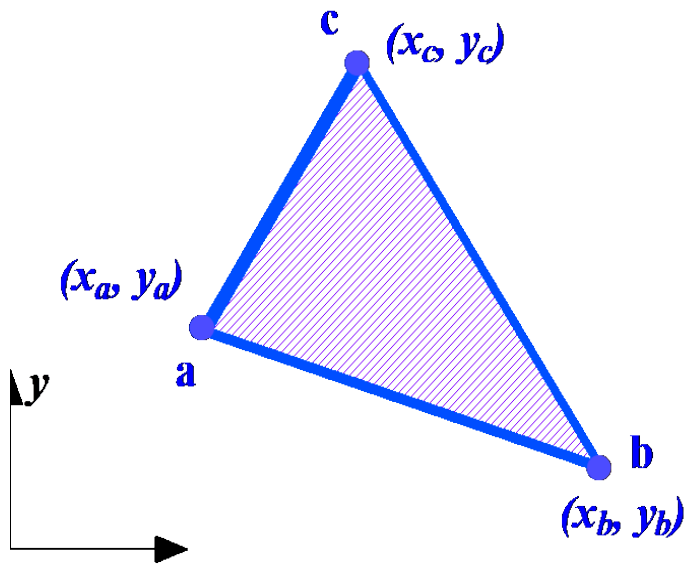
$\langle N(x,y) \rangle = \langle N_a(x,y) \quad N_b(x,y) \quad N_c(x,y) \rangle$ contains the **interpolation** (or **shape**) **functions** (one for each node)

How to find the shape functions?

Obviously, since we have 3 nodes, the interpolation functions will be linear in x and y :

$$\Rightarrow \quad N(x,y) = \alpha + \beta x + \gamma y$$

We also know that a shape function must be
= 1 at the corresponding node
= 0 at the other two nodes



Shape functions

For example,

$$N_a(x,y) = \alpha_1 + \beta_1 x + \gamma_1 y$$

We find the constants α_1, β_1 and γ_1 through the three equations :

$$\begin{cases} \alpha_1 + \beta_1 x_a + \gamma_1 y_a = 1 \\ \alpha_1 + \beta_1 x_b + \gamma_1 y_b = 0 \\ \alpha_1 + \beta_1 x_c + \gamma_1 y_c = 0 \end{cases}$$

We find

$$\begin{cases} \alpha_1 = (x_b y_c - x_c y_b)/2A^e \\ \beta_1 = (y_b - y_c)/2A^e \\ \gamma_1 = (x_c - x_b)/2A^e \end{cases} \quad \text{with} \quad A^e = \frac{1}{2}\begin{vmatrix} 1 & x_a & y_a \\ 1 & x_b & y_b \\ 1 & x_c & y_c \end{vmatrix} = \text{area of triangle}$$

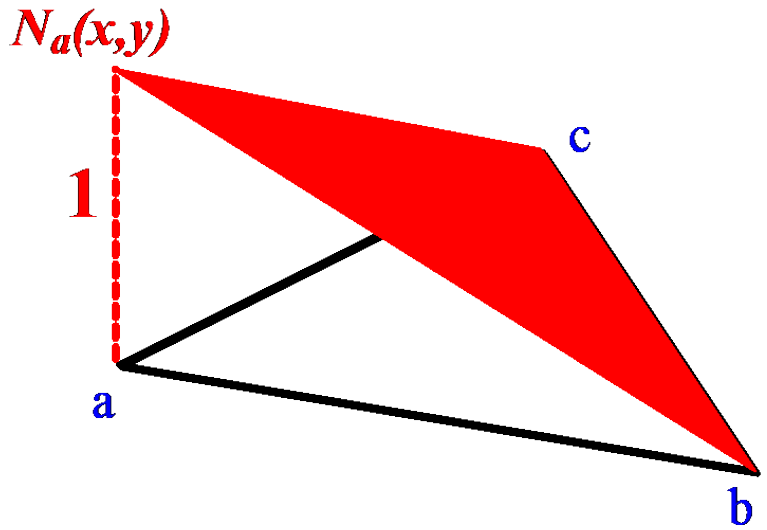
Repeat for the other two shape functions $N_b(x,y)$ and $N_c(x,y)$ to find

$$\langle N(x,y) \rangle = \left\langle \frac{a_1 + b_1 x + c_1 y}{2A^e} \quad \frac{a_2 + b_2 x + c_2 y}{2A^e} \quad \frac{a_3 + b_3 x + c_3 y}{2A^e} \right\rangle$$

$$\text{with} \quad \begin{cases} a_1 = x_b y_c - x_c y_b & a_2 = x_c y_a - x_a y_c & a_3 = x_a y_b - x_b y_a \\ b_1 = y_b - y_c & b_2 = y_c - y_a & b_3 = y_a - y_b \\ c_1 = x_c - x_b & c_2 = x_a - x_c & c_3 = x_b - x_a \end{cases}$$

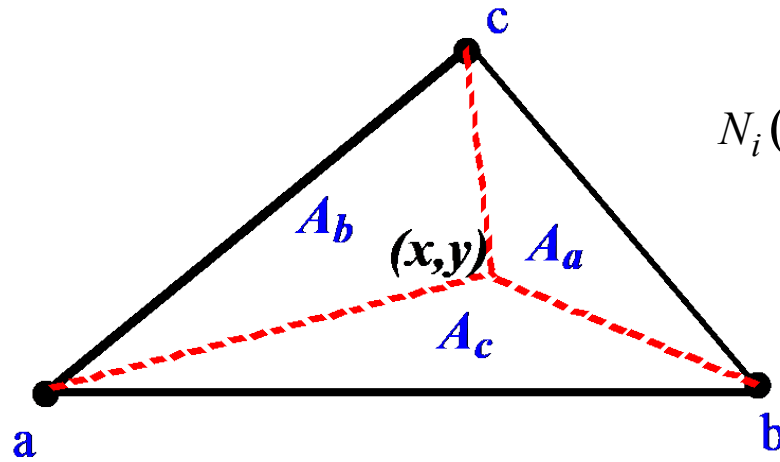
Shape functions (Cont'd)

What do the shape functions look like?



Note #1 :$\tilde{T}(x,y)$ is C^0 continuous inside and between elements. **Why?**

Note #2 : Geometrical significance of shape functions



$$N_i(x,y) = \frac{A_i}{A^e}$$

Local stiffness matrix and load vector

$$[k] = \int_{\Omega^e} [B]^T \, [\Lambda][B] \, d\Omega^e$$

$$\{r\} = \{r\}_Q - \{r\}_q = \int_{\Omega^e} Q \, \{N\} \, d\Omega^e - \int_{\partial\Omega_q^e} q_n^* \, \{N\} \, d\partial\Omega_q^e$$

Note : for the 3-node element, the shape functions $N_i(x,y)$ are linear, thus [B] is a constant matrix

$$[B] = \begin{bmatrix} \frac{\partial N_a}{\partial x} & \frac{\partial N_b}{\partial x} & \frac{\partial N_c}{\partial x} \\ \frac{\partial N_a}{\partial y} & \frac{\partial N_b}{\partial y} & \frac{\partial N_c}{\partial y} \end{bmatrix} = \frac{1}{2A^e} \begin{bmatrix} b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix}$$

Substitute into $[k] = \int_{\Omega^e} [B]^T [\Lambda] [B] \, d\Omega^e$ to find

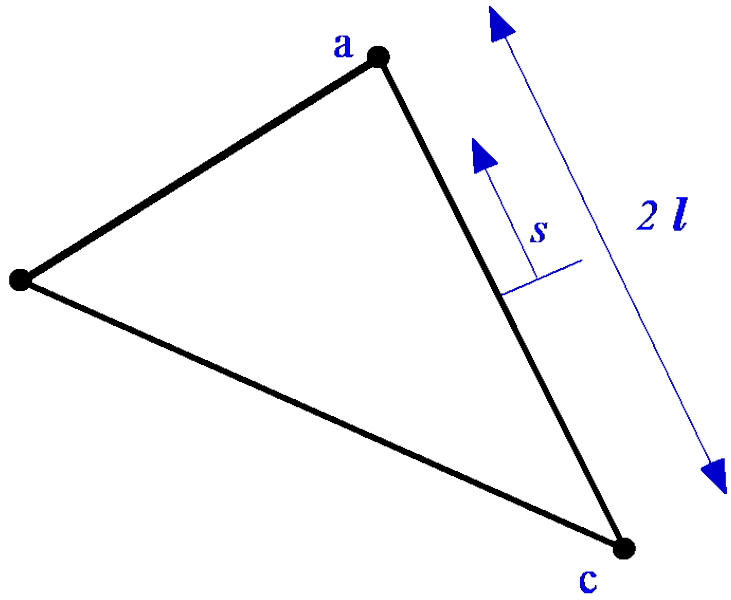
$$[k] = \frac{l}{4 A^e} \begin{bmatrix} \kappa_{xx} b_1^2 + \kappa_{yy} c_1^2 & \kappa_{xx} b_1 b_2 + \kappa_{yy} c_1 c_2 & \kappa_{xx} b_1 b_3 + \kappa_{yy} c_1 c_3 \\ & \kappa_{xx} b_2^2 + \kappa_{yy} c_2^2 & \kappa_{xx} b_2 b_3 + \kappa_{yy} c_2 c_3 \\ \textbf{symmetric} & & \kappa_{xx} b_3^2 + \kappa_{yy} c_3^2 \end{bmatrix}$$

Similarly, we find, for a constant Q

$$\{r\}_Q = \int_{\Omega^e} Q \, \{N\} \, d\Omega^e = \frac{Q A^e}{3} \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix}$$

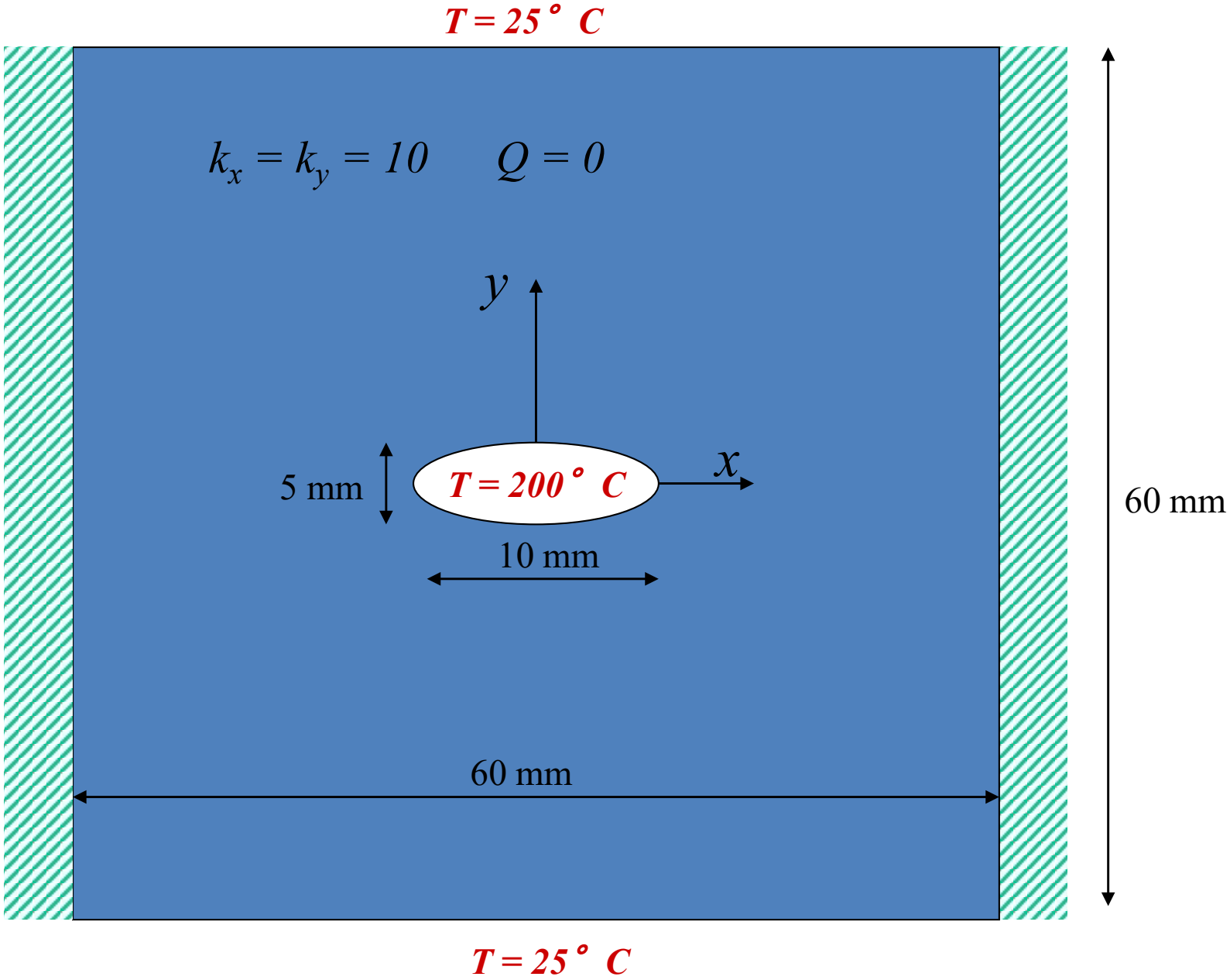
Finally, assuming that the imposed heat flux is applied along the side a-c (of length 2 l), we get

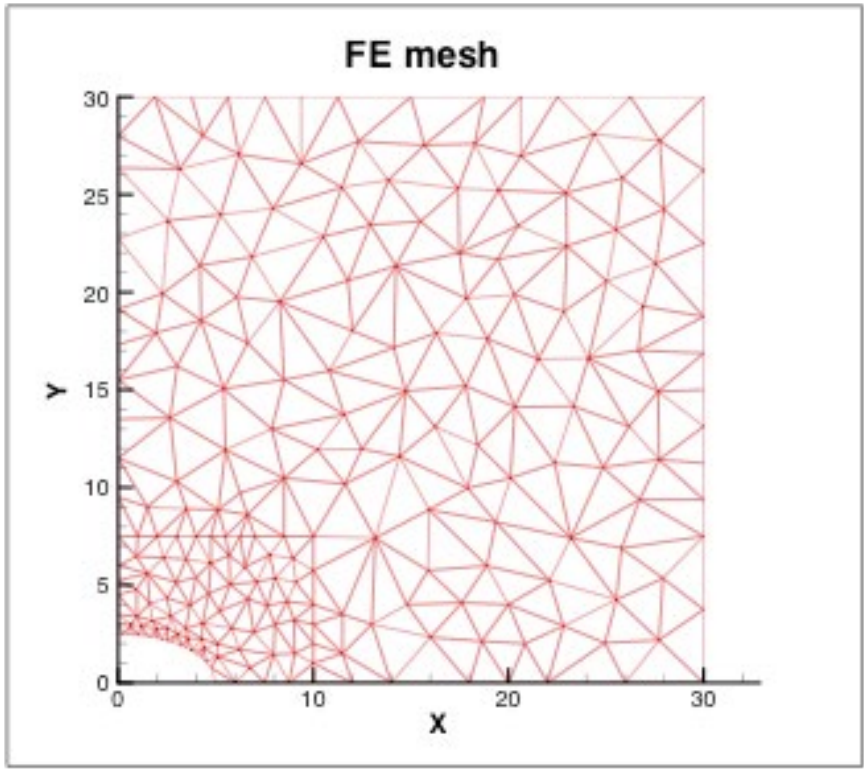
$$\{r\}_q = \int_{-l}^{l} q_n^* \, \{N\}_{\text{along } a-c} \, ds = \int_{-l}^{l} q_n^* \begin{Bmatrix} \frac{l+s}{2l} \\ 0 \\ \frac{l-s}{2l} \end{Bmatrix} ds = q_n^* \, l \begin{Bmatrix} 1 \\ 0 \\ 1 \end{Bmatrix}$$

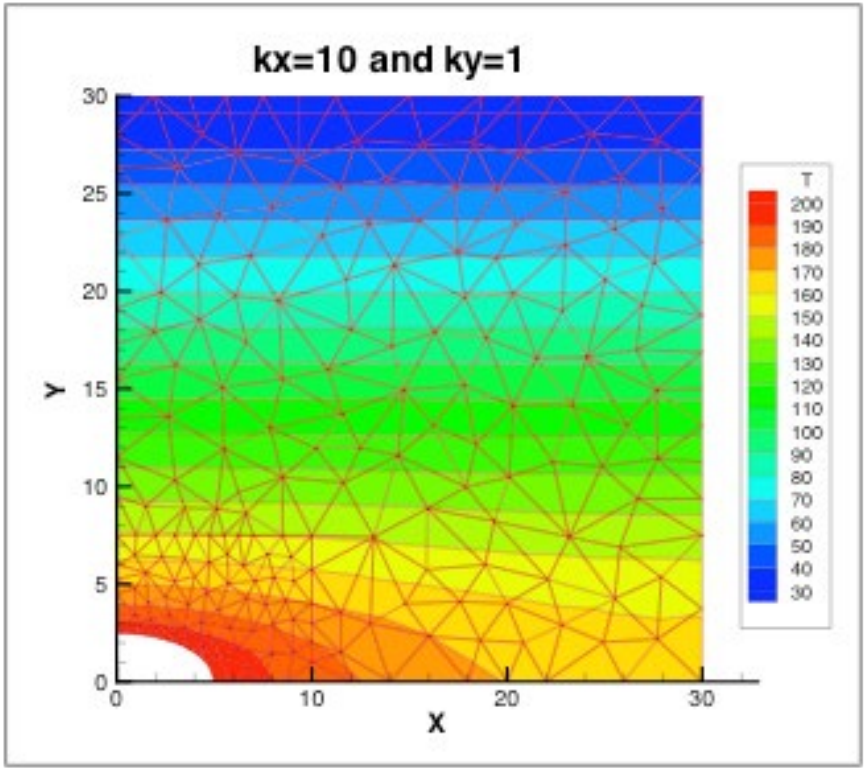


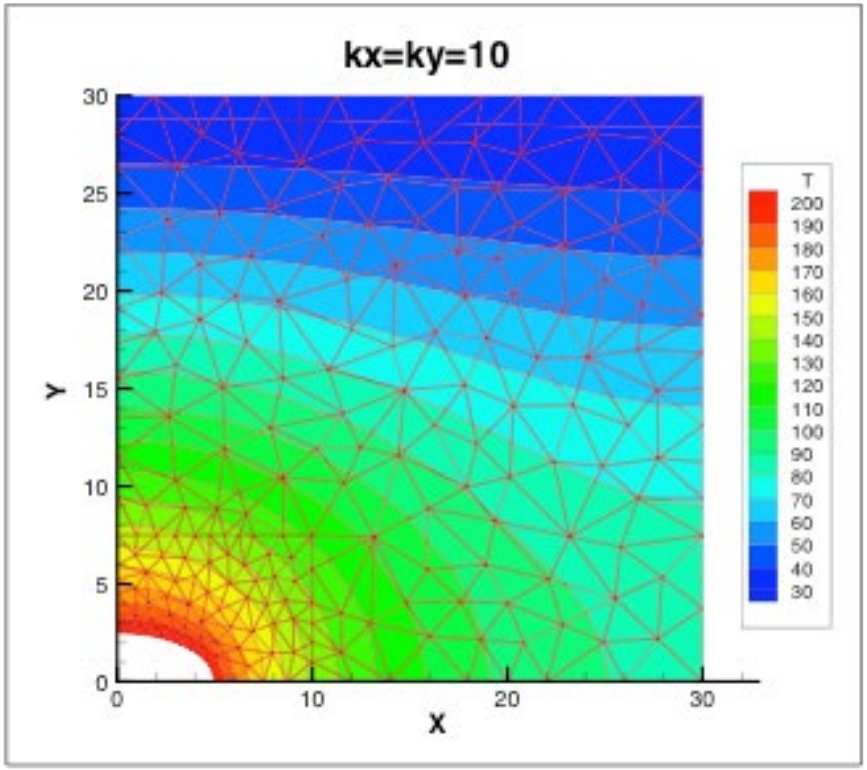
The rest of the FE formulation (assembling, minimization, …) is identical, leading once again to

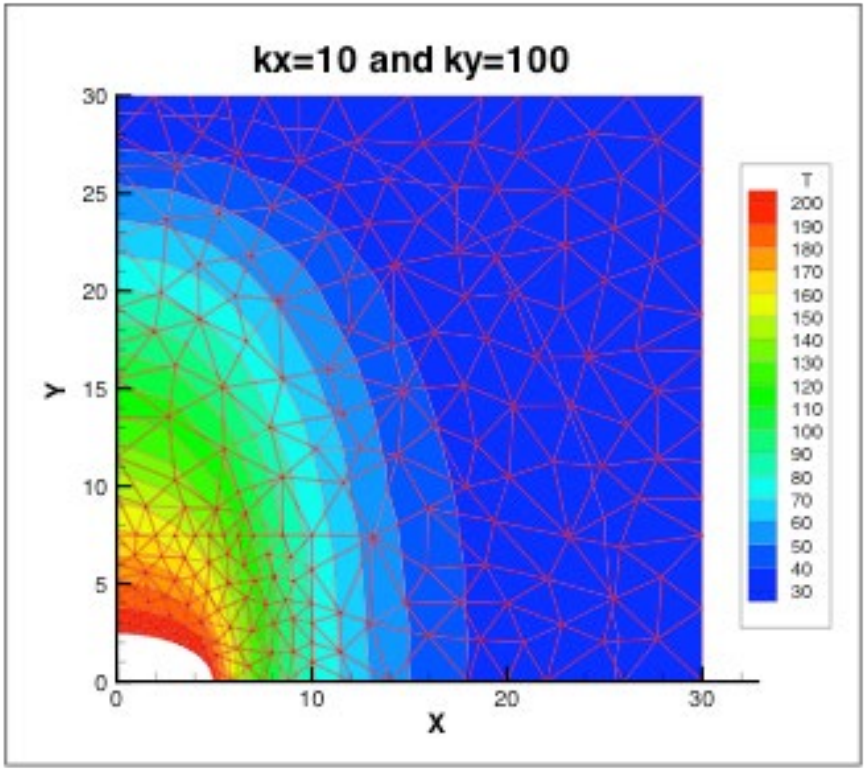
$$[K] \{D\} = \{R\}$$

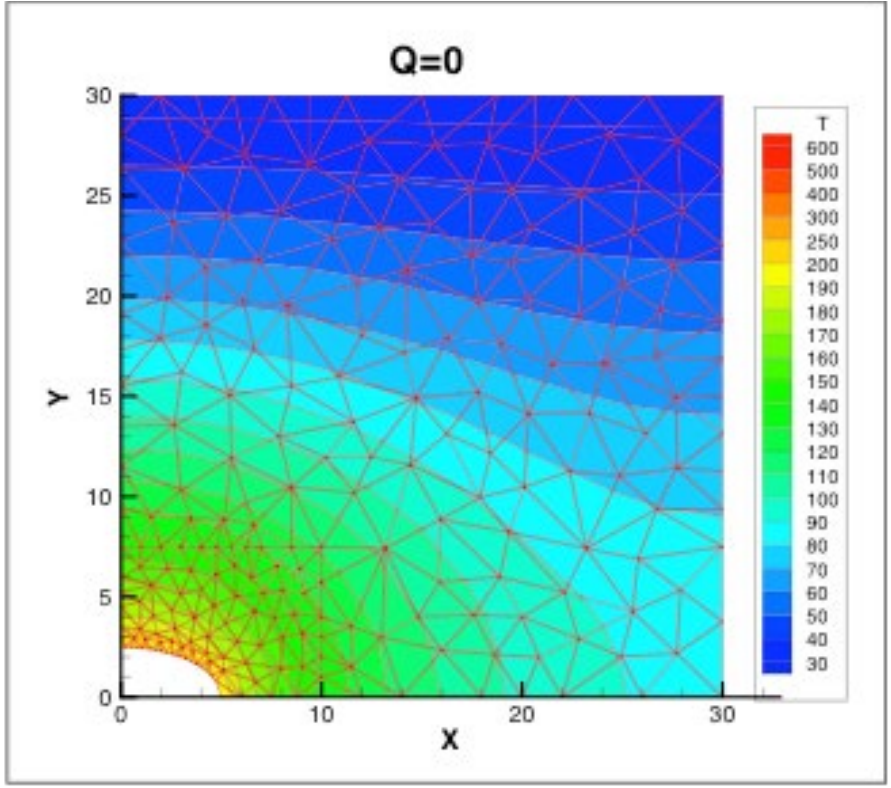

T = 25° C
$k_x = k_y = 10$
$Q = 0$
y
x
5 mm
T = 200° C
10 mm
60 mm
60 mm
T = 25° C

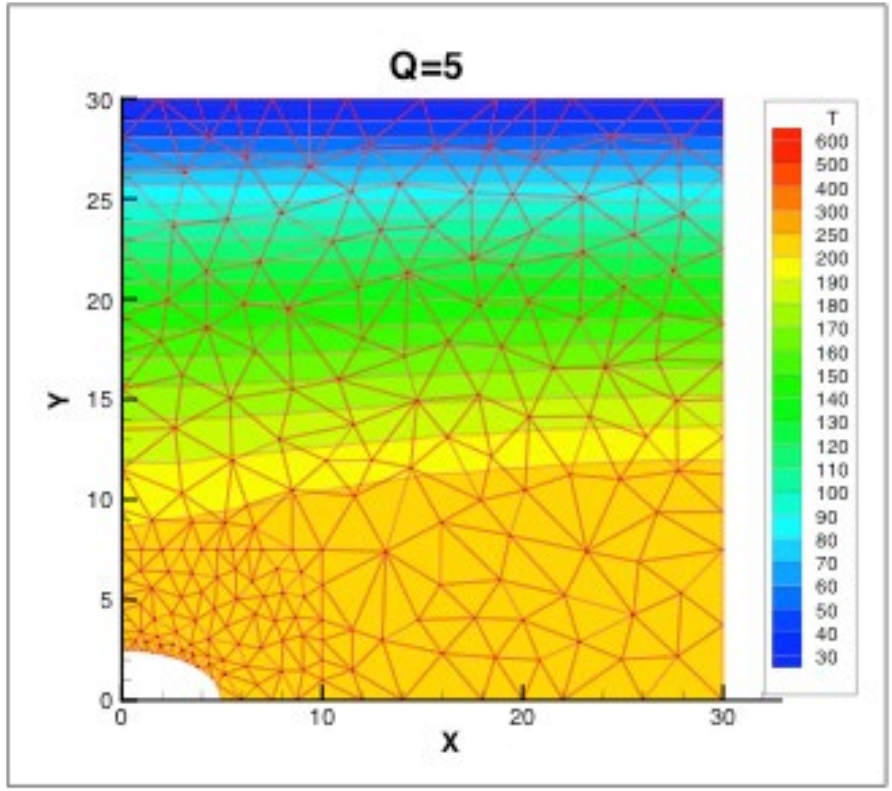

FE mesh
Y
X
0
5
10
15
20
25
30

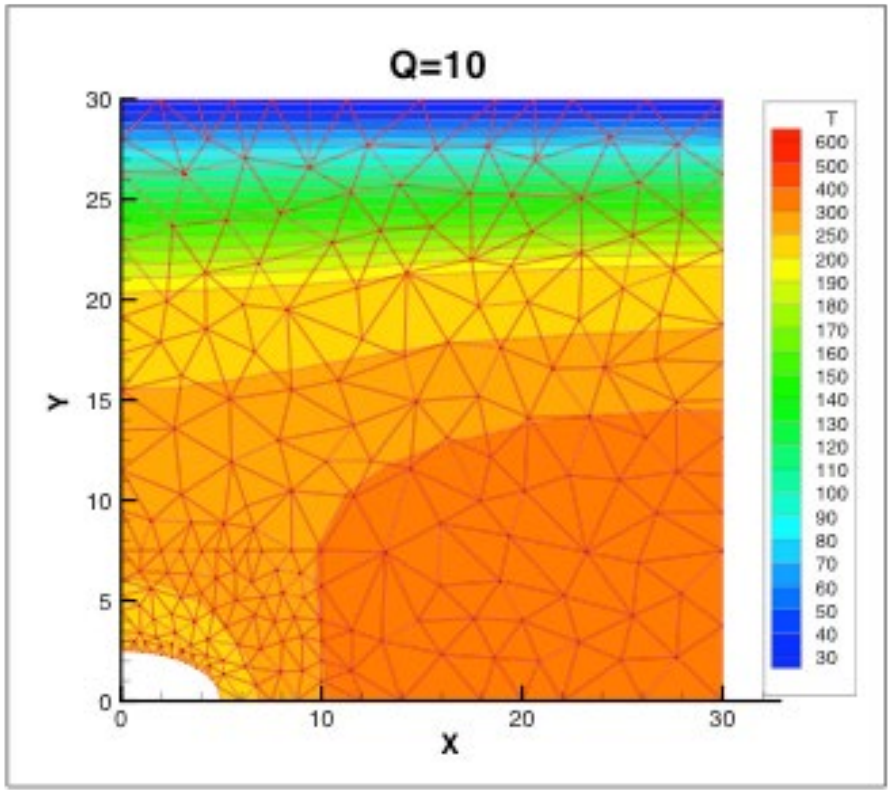

kx=10 and ky=1
Y
X
T
200
190
180
170
160
150
140
130
120
110
100
90
80
70
60
50
40
30

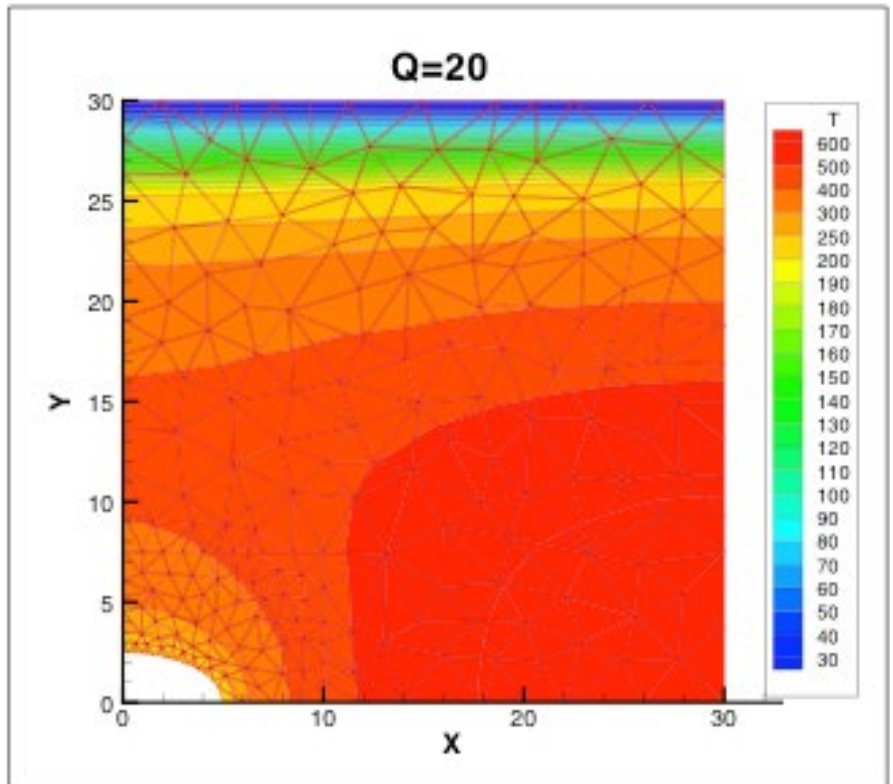

kx=ky=10
Y
X
T
200
190
180
170
160
150
140
130
120
110
100
90
80
70
60
50
40
30


kx=10 and ky=100
Y
X
T
200
190
180
170
160
150
140
130
120
110
100
90
80
70
60
50
40
30


Q=0
T
600
500
400
300
250
200
190
180
170
160
150
140
130
120
110
100
90
80
70
60
50
40
30
Y
X
0
5
10
15
20
25
30

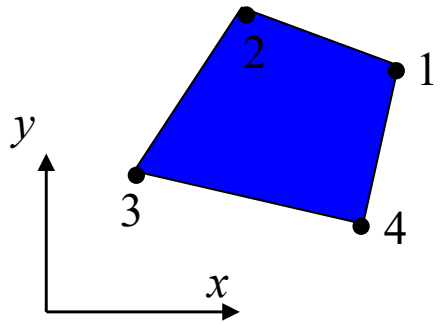

Q=5
T
600
500
400
300
250
200
190
180
170
160
150
140
130
120
110
100
90
80
70
60
50
40
30
Y
X
0
5
10
15
20
25
30


Q=10
T
600
500
400
300
250
200
190
180
170
160
150
140
130
120
110
100
90
80
70
60
50
40
30
Y
X
0
5
10
15
20
25
30


Q=20
T
600
500
400
300
250
200
190
180
170
160
150
140
130
120
110
100
90
80
70
60
50
40
30
Y
X
0
5
10
15
20
25
30

Examples of 2-D global elements

4-node quadrilateral element.



$$T(x,y) \approx \tilde{T}(x,y) = \langle N(x,y) \rangle \{d\}$$

with

$$\langle d \rangle = \langle T_1 \;\; T_2 \;\; T_3 \;\; T_4 \rangle$$

$$\langle N \rangle = \langle N_1(x,y) \; N_2(x,y) \; N_3(x,y) \; N_4(x,y) \rangle$$

and

$$N_i(x,y) \;=\; \alpha_i + \beta_i\, x + \gamma_i\, y + \delta_i\, x\, y \qquad (i = 1, 2, 3, 4)$$

To find the coefficients α_i, β_i, γ_i, δ_i, use the basic property of the shape functions.

E.g., $N_1(x_1, y_1) = 1$; $N_1(x_2, y_2) = N_1(x_3, y_3) = N_1(x_4, y_4) = 0$.

But, even for this relatively simple 4-node element, the expression of the shape functions is quite messy. Furthermore, integrating these shape functions (or their derivatives) on a general quadrilateral shape is quite complex. This is why mapped (or isoparametric) elements are preferred (see below).

2-D triangular elements

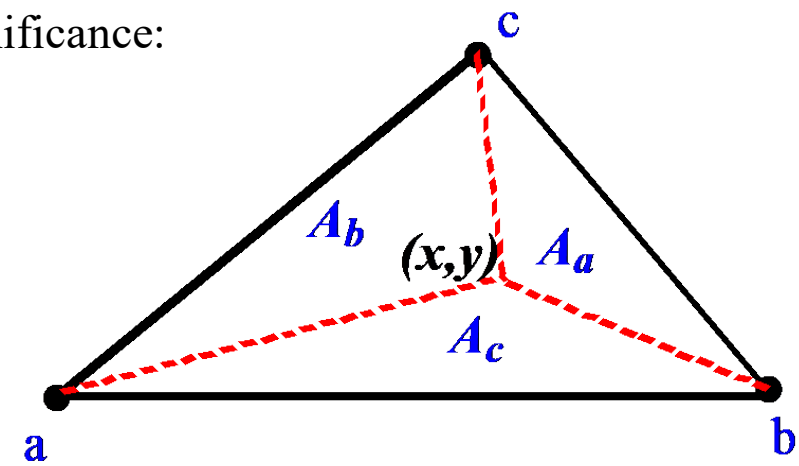
Global triangular elements are easier to handle than global quadrilateral elements, especially if we work in terms of the triangular coordinates L_1, L_2 and L_3 (instead of the x and y coordinates) defined as:

$$L_1 = (a_1 + b_1 x + c_1 y)/2A$$
$$L_2 = (a_2 + b_2 x + c_2 y)/2A$$
$$L_3 = (a_3 + b_3 x + c_3 y)/2A = 1 - L_1 - L_2$$

with, as seen before,

$$\begin{cases} a_1 = x_b y_c - x_c y_b & a_2 = x_c y_a - x_a y_c & a_3 = x_a y_b - x_b y_a \\ b_1 = y_b - y_c & b_2 = y_c - y_a & b_3 = y_a - y_b \\ c_1 = x_c - x_b & c_2 = x_a - x_c & c_3 = x_b - x_a \end{cases}$$

Recall: geometrical significance:

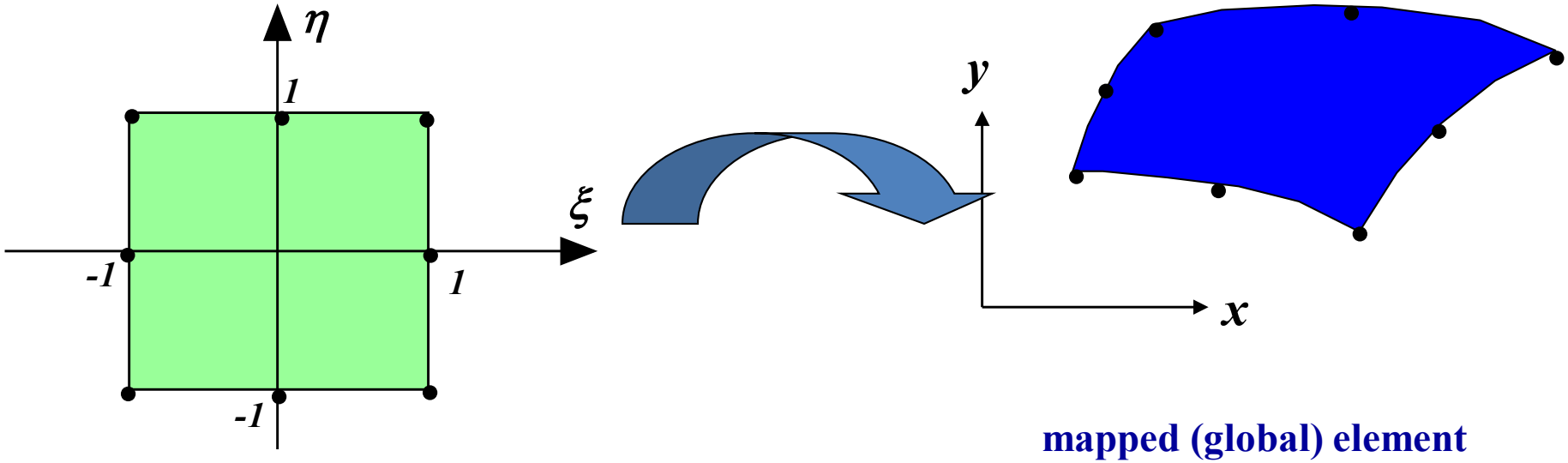


$$L_i(x,y) = \frac{A_i}{A}$$

Iso-parametric Elements

Objective: to generate “more complicated elements” (even with curved sides).

Why **mapped** elements? Because these elements use a **mapping** between a **parent element** and the **global element**.



parent element

mapped (global) element

Associated with the transformation is a Jacobian (or transformation) matrix [J] that relate the x- and y-derivatives to the ξ- and η-derivatives (see later).

Iso-parametric Elements

- The term iso-parametric is derived from the use of the same shape functions (or interpolation functions) [*N*] to define the element's geometric shape as are used to define the solution within the element.

For example, when the interpolation function is $u = a_1 + a_2\xi$ for the temperature, we use x=$a_1 + a_2\xi$ for the description of the nodal coordinate of a point on the 1D element

Shape function (interpolation) for the solution field is the same as the for the geometry

Iso-parametric Mapping

• A mapped element has thus two different mappings:

- one for the unknown solution field (velocity, temperature, …):

$$T(x,y) \approx \tilde{T}(x,y) = \langle\ N\ \rangle \{\ d\ \}$$

with

$$\langle\ d\ \rangle = \langle T_1\ T_2\ T_3\ \ldots\ T_M \rangle = \text{vector with unknown nodal values}$$

- one for the coordinates

$$\begin{Bmatrix} x \\ y \end{Bmatrix} = \left[\ \tilde{N}\ \right] \{c\}$$

with

$$\langle\ c\ \rangle = \langle x_1 \quad y_1 \quad x_2 \quad y_2 \quad x_3 \quad y_3 \quad \ldots \quad x_M \quad y_M \rangle = \text{vector with nodal coordinates}$$

• For mapped elements, the shape functions N and $\tilde{N}$ are functions of ξ and η, because it is much easier to define them on the simply shaped parent element.

• We have different types of mapped elements based on the choice of shape functions:

- an **isoparametric** (or "same parameter") element is one for which $N \equiv \tilde{N}$
- a **subparametric** element is such that the degree of $N >$ degree of $\tilde{N}$
- a **superparametric** element is such that the degree of $N <$ degree of $\tilde{N}$

Since the mapping somewhat complicates the formulation of [k] and { r }, we will need numerical integration.

Iso-parametric Mapping

In the mapped coordinates, the shape functions need to satisfy

1. Kronecker delta property

Then $$N_i = \begin{cases} 1 & \text{at node } i \\ 0 & \text{at all other nodes} \end{cases}$$

2. Polynomial completeness

$$\sum_i N_i = 1$$
$$\sum_i N_i x_i = x$$
$$\sum_i N_i y_i = y$$

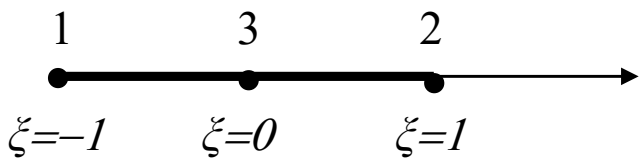
The relationship

$$x = \sum_i N_i(\xi,\eta) x_i$$
$$y = \sum_i N_i(\xi,\eta) y_i$$

(ξ,η): isoparametric coordinates
(x,y): global coordinates

Example: Isoparametric 1-D elements (3-noded)

parent element



mapped element

Shape functions:

$$\begin{cases} N_1(\xi) = \dfrac{-\xi(1-\xi)}{2} \\ N_2(\xi) = \dfrac{\xi(1+\xi)}{2} \\ N_3(\xi) = 1-\xi^2 \end{cases}$$

Geometric mapping:

$$x(\xi) = \sum_{i=1}^{3} x_i\, N_i(\xi) = x_1\, N_1(\xi) + x_2\, N_2(\xi) + x_3\, N_3(\xi)$$

i.e., $\xi = -1$ gets mapped into $x = x_1$

$\xi = 0$ gets mapped into $x = x_3$

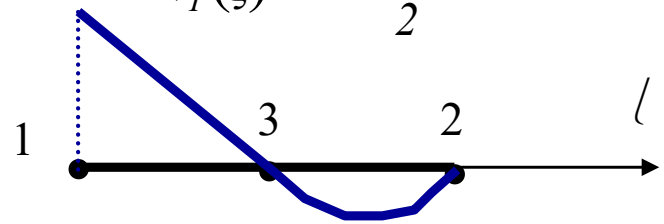
$\xi = 1$ gets mapped into $x = x_2$

$\xi = 1/2$ gets mapped into $x = -\dfrac{1}{8}x_1 + \dfrac{3}{8}x_2 + \dfrac{6}{8}x_3$

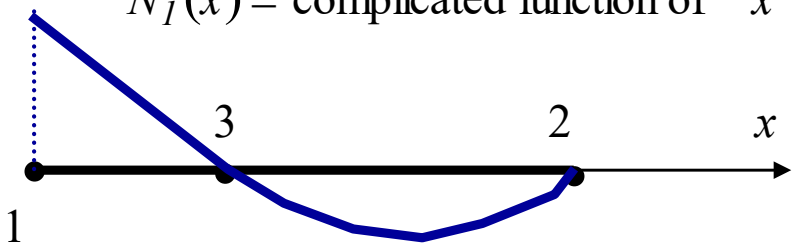
Note: although the shape functions are exclusively defined on the parent element, we can consider the shape functions to be mapped as well.

E.g., for N_1:

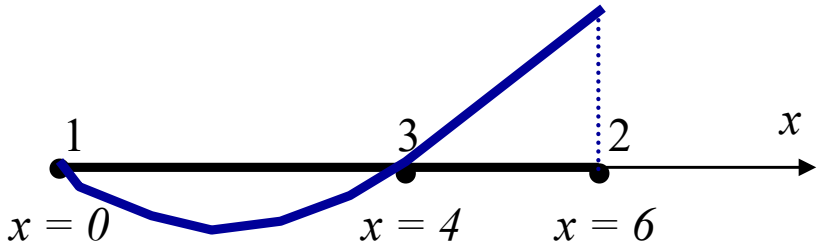
$$N_1(\xi) = \frac{-\xi(1-\xi)}{2}$$



$$N_1(x) = \text{complicated function of } x$$



Example: consider the second shape function for the following mapped element:



$$x(\xi) = x_1 N_1(\xi) + x_2 N_2(\xi) + x_3 N_3(\xi)$$

$$= 0.\frac{-\xi(1-\xi)}{2} + 6.\frac{\xi(1+\xi)}{2} + 4.(1-\xi^2)$$

$$= 4 + 3\xi - \xi^2$$

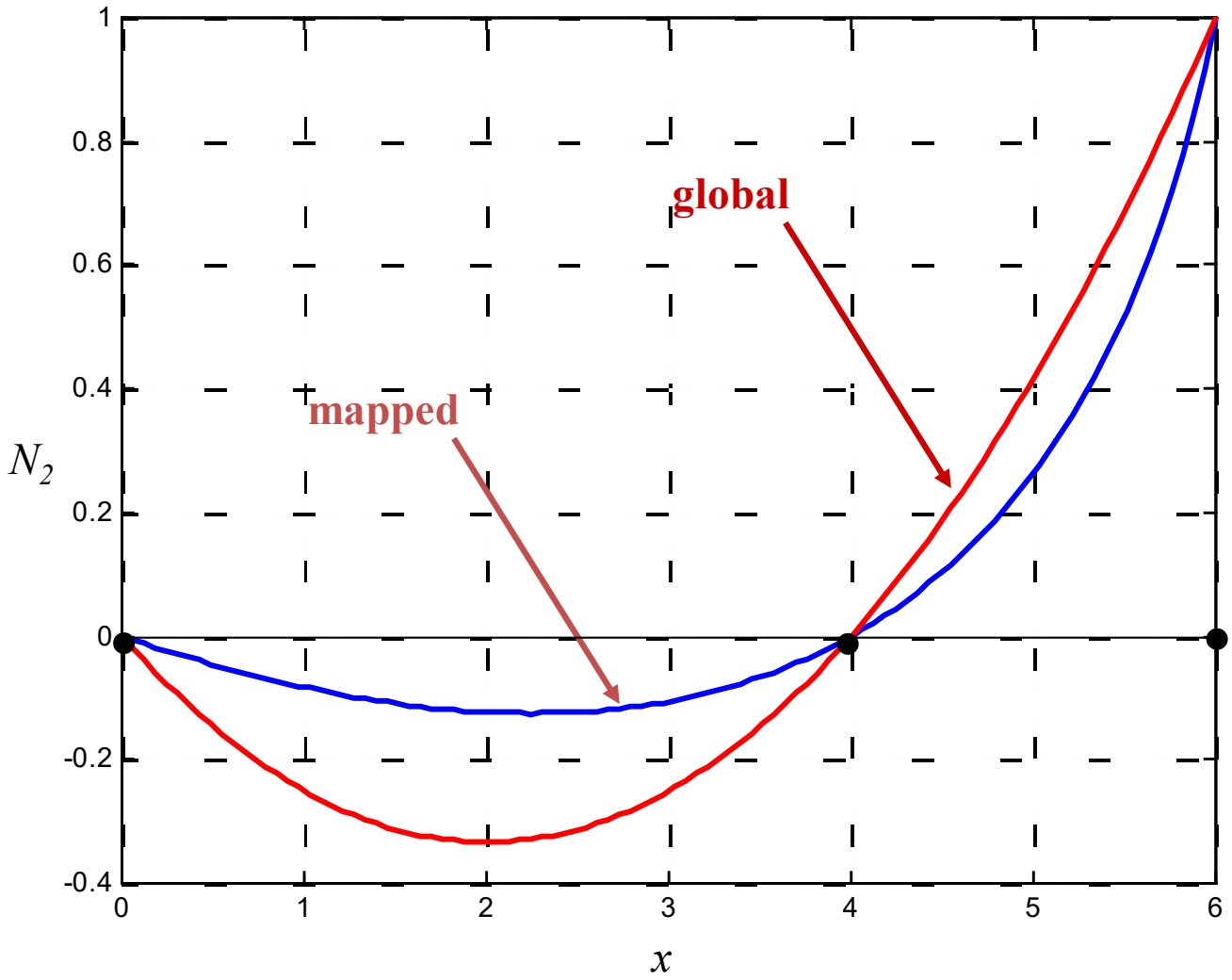
Invert $\Rightarrow$ $\boxed{\xi = \dfrac{3-\sqrt{25-4x}}{2}}$

We can thus rewrite the shape function as $N_2 = \dfrac{\xi(1+\xi)}{2} = \dfrac{1}{2}\left(\dfrac{3-\sqrt{25-4x}}{2}\right)\left(1+\dfrac{3-\sqrt{25-4x}}{2}\right)$

i.e., $\boxed{N_2 = \dfrac{1}{2}\left(10 - 2\sqrt{25-4x} - x\right)}$

Let us compare the " mapped shape function" $N_2 = \frac{1}{2}\left(10 - 2\sqrt{25 - 4x} - x\right)$

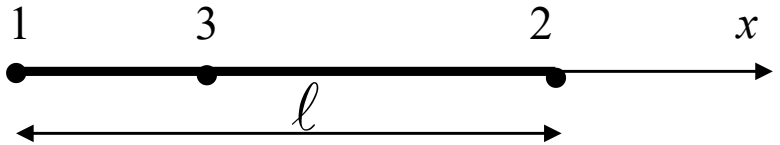
with the " global shape function" $N_2 = \frac{(x-0)(x-4)}{(6-0)(6-4)} = \frac{1}{12}x(x-4)$



Stiffness matrix of a 1-D mapped element

The initial steps of the derivation are identical. For, say, a 3-node element of length ℓ:

$$[\ k\ ] = \int_0^{\ell} \{\ B\ \}\ k \langle\ B\ \rangle\ dx$$



where the vector $\langle\ B\ \rangle$ is such that $\dfrac{d\tilde{u}}{dx} = \langle\ B\ \rangle \{d\} = \langle\ B\ \rangle \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \end{Bmatrix}$

But now, $\langle\ N\ \rangle = \langle\ N(\xi)\ \rangle = \langle\ N_1\ \ N_2\ \ N_3 \rangle = \left\langle \dfrac{-\xi(1-\xi)}{2} \quad \dfrac{\xi(1+\xi)}{2} \quad 1-\xi^2 \right\rangle$

and $\langle\ B\ \rangle = \dfrac{d}{dx} \langle\ N\ \rangle = \dfrac{d}{d\xi} \langle\ N\ \rangle \dfrac{d\xi}{dx}$

To find $\dfrac{d\xi}{dx}$, we use the geometrical mapping $x(\xi) = \sum_{i=1}^{3} x_i\ N_i(\xi)$

Thus the Jacobian of the transformation $J(\xi) = \dfrac{dx}{d\xi} = \sum_{i=1}^{3} x_i\ \dfrac{dN_i}{d\xi}$ with $\left\langle \dfrac{dN}{d\xi} \right\rangle = \left\langle \dfrac{-1+2\xi}{2} \quad \dfrac{1+2\xi}{2} \quad -2\xi \right\rangle$

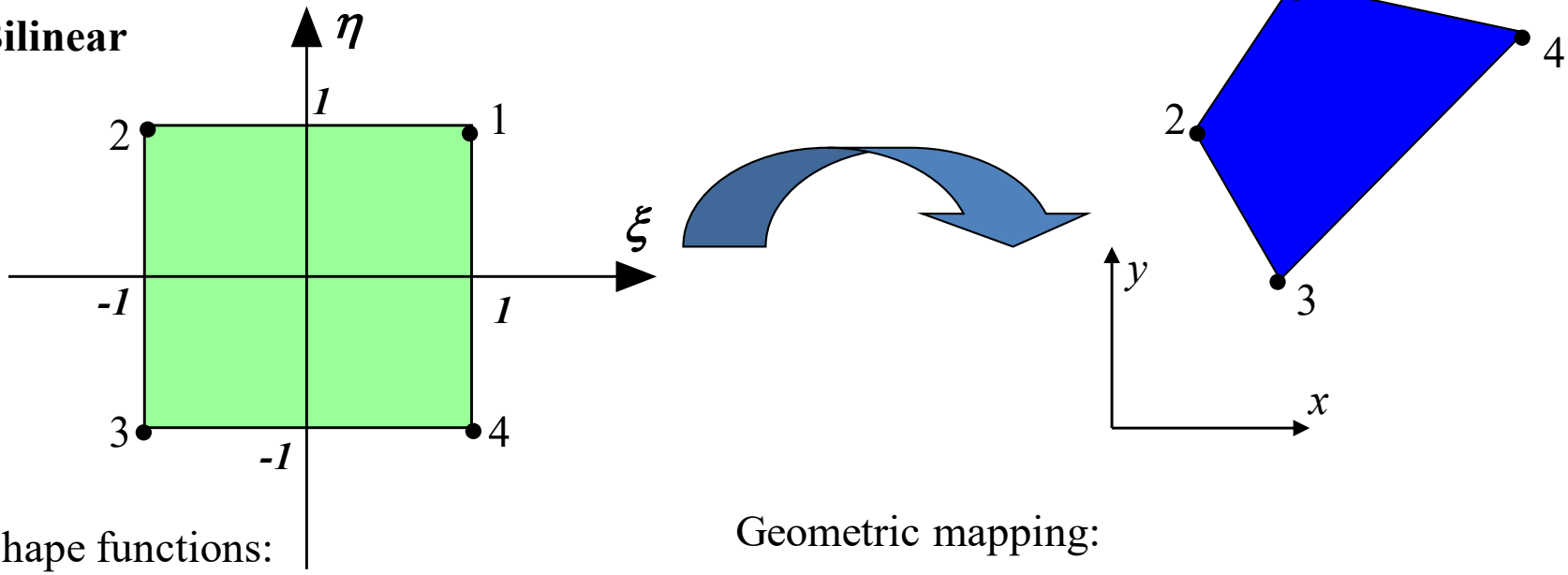
Thus $[\ k\ ] = \int_0^{\ell} \{\ B\ \}\ k \langle\ B\ \rangle\ dx = \int_{-1}^{1} \{\ B\ \}\ k \langle\ B\ \rangle\ J(\xi)\ d\xi$

with $\langle\ B\ \rangle = \dfrac{d}{dx} \langle\ N \rangle = \dfrac{1}{J(\xi)} \dfrac{d}{d\xi} \langle\ N \rangle$

Isoparametric 2-D elements

A. Square parent element

Bilinear



Shape functions:

$$\begin{cases} N_1(\xi,\eta) = \dfrac{(1+\xi)(1+\eta)}{4} \\ N_2(\xi,\eta) = \dfrac{(1-\xi)(1+\eta)}{4} \\ N_3(\xi,\eta) = \dfrac{(1-\xi)(1-\eta)}{4} \\ N_4(\xi,\eta) = \dfrac{(1+\xi)(1-\eta)}{4} \end{cases}$$

Geometric mapping:

$$\begin{cases} x(\xi,\eta) = \sum_{i=1}^{4} x_i \, N_i(\xi,\eta) \\ y(\xi,\eta) = \sum_{i=1}^{4} y_i \, N_i(\xi,\eta) \end{cases}$$

2-D Iso-parametric Elements

Recall :

$$[k] = \int_{S^e} [B]^T [\Lambda] [B] \, dS^e$$

$$\{r\} = \{r\}_Q - \{r\}_q$$

$$\{r\}_Q = \int_{S^e} Q \{N\} \, dS^e$$

$$\{r\}_q = \int_{B_q^e} q_n^* \{N\} \, dB_q^e$$

where, for a generic M - node element :

$$\langle N \rangle = \langle N_1(x,y) \;\; N_2(x,y) \;\; N_3(x,y) \;\; ..... \;\; N_M(x,y) \rangle$$

$$[B] = \begin{bmatrix} \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial x} & \cdots & \frac{\partial N_M}{\partial x} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_2}{\partial y} & \cdots & \frac{\partial N_M}{\partial y} \end{bmatrix}$$

1) The shape functions N_i are already written in terms of ξ and η.

2) What about $\dfrac{\partial N_i}{\partial x}$ and $\dfrac{\partial N_i}{\partial y}$?

For any function ϕ, we have (chain rule):

$$\begin{Bmatrix} \phi_{,\xi} \\ \phi_{,\eta} \end{Bmatrix} = \underbrace{\begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} \end{bmatrix}}_{[J(\xi,\eta)]} \begin{Bmatrix} \phi_{,x} \\ \phi_{,y} \end{Bmatrix}$$

Note: $x(\xi,\eta) = \sum_{i=1}^{M} x_i \, N_i(\xi,\eta)$

$$\Rightarrow \quad \frac{\partial x}{\partial \xi} = \sum_{i=1}^{M} x_i \frac{\partial N_i(\xi,\eta)}{\partial \xi}, \quad \frac{\partial x}{\partial \eta} = \ldots$$

Inverting: $\begin{Bmatrix} \phi_{,x} \\ \phi_{,y} \end{Bmatrix} = \underbrace{\begin{bmatrix} \dfrac{\partial \xi}{\partial x} & \dfrac{\partial \eta}{\partial x} \\ \dfrac{\partial \xi}{\partial y} & \dfrac{\partial \eta}{\partial y} \end{bmatrix}}_{[J(\xi,\eta)]^{-1}} \begin{Bmatrix} \phi_{,\xi} \\ \phi_{,\eta} \end{Bmatrix}$

This result is obviously valid for $\phi \equiv N_i$:

$$\begin{Bmatrix} \dfrac{\partial N_i}{\partial x} \\ \dfrac{\partial N_i}{\partial y} \end{Bmatrix} = \left[J\left(\xi,\eta\right) \right]^{-1} \begin{Bmatrix} \dfrac{\partial N_i}{\partial \xi} \\ \dfrac{\partial N_i}{\partial \eta} \end{Bmatrix}$$

3) $dS^e = dx\,dy = det[J]\; d\xi\, d\eta = \left(\dfrac{\partial x}{\partial \xi}\dfrac{\partial y}{\partial \eta} - \dfrac{\partial x}{\partial \eta}\dfrac{\partial y}{\partial \xi} \right) d\xi\, d\eta$

* For a triangular element :

$$[k] = \int_0^1 \int_0^{1-\eta} [B]^T [\Lambda][B] \det[J] d\xi d\eta$$

$$\{r\}_Q = \int_0^1 \int_0^{1-\eta} Q\{N\} \det[J] d\xi d\eta$$

$$\{r\}_q = \int_0^1 q_n^* \{N(\xi, \eta = 0)\} \sqrt{\left(\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial y}{\partial \xi}\right)^2} \, d\xi$$

assuming that the imposed heat flux is applied along the edge mapped

where $x = \sum_{i=1}^{M} N_i(\xi, \eta)\, x_i$

$$y = \sum_{i=1}^{M} N_i(\xi, \eta)\, y_i$$

$$[J] = \text{Jacobian} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix}$$

$$\cdot \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} = [J]^{-1} \begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{Bmatrix} \quad \Rightarrow \quad \text{build } [B] \text{ matrix}$$

Rajeev K. Jaiman
Email: rjaiman@mech.ubc.ca