Finite-Element Methods and Numerical Linear Algebra

Knut–Andreas Lie

Dept. of Informatics, University of Oslo

Weighted Residual Methods

Idea:

- Assume that the solution can be represented in terms of analytic functions
- Express the approximate solution as a sum of such functions (rather than point-values)

Many subclasses of methods:

- finite-element methods (today)
- finite-volume methods (previous lecture)
- spectral methods (not covered here)
- boundary element methods (not covered here)
- …

Weighted Residual Methods, cont'd

So far: finite-differences

- unknown function computed as a set of discrete nodal values
- differential formulation for each node
- Taylor series expansions on (structured) grids
- Increase accuracy by reducing local truncation error

Today: weighted-residual methods

- unknown function computed as a sum of continuous shape functions
- integral formulation of the equations
- minimize weighted residual for arbitrary control volume
- interpolation errors
- Increase accuracy by higher-order interpolation and optimized coefficients for minimum residuals

General Class of Problems

Assume the PDE: $\mathcal{L}(u(\mathbf{x})) = 0, \quad \mathbf{x} \in \Omega$

Example: Let us revisit the steady heat equation

$$-\nabla\big[K(x)\nabla u\big] = f(x), \qquad x \in \Omega$$
$$u(x) = g(x), \qquad x \in \partial\Omega$$

$$\longrightarrow \mathcal{L}(u(x)) = f(x) + \nabla\big[K(x)\nabla u\big]$$

Here

- $K(x)$, $f(x)$, and $g(x)$ are known functions
- $u(x)$ is the unknown function
- Ω is a domain with complete boundary $\partial\Omega$

Weighted Residual Methods, cont'd

Seek approximations of the form

$$\hat{u} = \sum_{j=1}^{M} u_j N_j(\mathbf{x})$$

where $N_j(\mathbf{x})$ are prescribed functions and u_j are unknown coefficients. The N_j's are called *basis functions* or *trial functions*

An approximate solution $\hat{u}$ should minimize the error $u - \hat{u}$:

- For special problems, minimize $||u - \hat{u}||$
- Not possible in general, since u is unknown

How to Compute the Coefficients u_j?

In general, to determine u_j we must minimize the residual

$$R(u_1, \ldots, u_M; \mathbf{x}) = \mathcal{L}(\hat{u})$$

i.e., minimize how far $\hat{u}$ is from satisfying the equation $\mathcal{L}(u) = 0$

- *Galerkin-type methods*: the weighted residual should disappear over Ω for linearly independent weights W_i or weighting functions $W_i(x)$

$$\int_\Omega RW_i \, d\Omega = 0, \quad i = 1, \ldots, M.$$

Notice that $R(\hat{u}) = 0$ in the *weak sense* and that $R(\hat{u}) \neq 0$ pointwise

How to Compute the Coefficients u_j, cont'd

- *The least-squares method*: minimize the average square residual $\int_\Omega R^2\,d\Omega$

$$\int_\Omega R\frac{\partial R}{\partial u_i}\,d\Omega = 0, \quad i = 1,\ldots,M$$

  i.e., $W_i = \partial R/\partial u_i$.

- *The collocation method*: choose $W_i = \delta(\mathbf{x} - \mathbf{x}^{[i]})$, where δ is the Dirac delta function,

$$R(u_1,\ldots,u_M;\mathbf{x}^{[i]}) = 0, \quad i = 1,\ldots,M.$$

- *The subdomain collocation method*: decompose Ω into M subdomains, $\Omega = \cup_{\ell=1}^{M}\Omega_\ell$ (equivalent to a finite-volume method)

$$\int_{\Omega_i} L(\hat{u})d\Omega = 0, \quad i = 1,\ldots,M.$$

Example: 1D Poisson equation

$$\mathcal{L}(u) = u''(x) + f(x) = 0, \qquad x \in [0,1]$$

Discretization:

$$u(x) \approx \hat{u}(x) = \sum_{j=1}^{M} u_j N_j(x)$$

The residual:

$$R\big(\hat{u}(x)\big) = f(x) + \sum_{j=1}^{M} u_j N_j''(x)$$

Using the least-squares approach:

$$\frac{\partial R}{\partial u_i} = \sum_{j=1}^{M} \frac{\partial u_j}{\partial u_i} N_j''(x) = N_i''(x)$$

Example, cont'd

The system of equations becomes

$$\int_0^1 \Big(f(x) + \sum_{j=1}^{M} u_j N_j(x) \Big) N_i''(x)\, dx = 0$$

$$-\sum_{j=1}^{M} \Big(\int_0^1 N_i''(x) N_j''(x)\, dx \Big) u_j = \int_0^1 f(x) N_i''(x)\, dx$$

$\longrightarrow$ linear system of equations $\mathbf{Au} = \mathbf{b}$, where

- $A_{i,j} = \int_0^1 N_i''(x) N_j''(x)\, dx$
- $b_i = \int_0^1 f(x) N_i''(x)\, dx$

Example, cont'd

Using the Galerkin approach:

$$-\sum_{j=1}^{M}\Big(\int_0^1 W_i(x)N_j''(x)\,dx\Big)u_j = \int_0^1 f(x)W_i(x)\,dx$$

Again a system of equations $\mathbf{Au} = \mathbf{b}$.

Two type of methods:

- $W_i = N_i$, *Galerkin method*
- $W_i \neq N_i$, *Petrov–Galerkin* method

How to Choose Test Functions?

For simplicity, assume that $u = 0$ on $\partial\Omega$.

The functions $N_k(x)$ can in principle be choosen almost arbitrarily:

- power series: $N_k(x) = x^k$
- fourier series: $N_k(x) = \{\sin(kx), \cos(kx)\}$
- Lagrange, Hermite, Chebychev polynomials
- ...

In general, to get a well-behaved method we require that:

- $N_k = 0$ on the boundary
- N_k almost orthogonal, to avoid numerical instabilities

The Finite-Element Method (FEM)

Finite elements is a way of fulfilling these two requirements:

- divide the domain into non-overlapping *elements*
- let N_k be a simple polynomial over each element
- the global N_k is a picewise polynomial that vanishes except on a local patch of elements

Features:

- A very flexible approach
- Straightforward handling of complicated geometries
- Easy to construct higher-order approximations
- A broad spectrum of applications
- An engineering method
- Has a strong mathematical foundation

Piecewise Polynomial Basis Functions

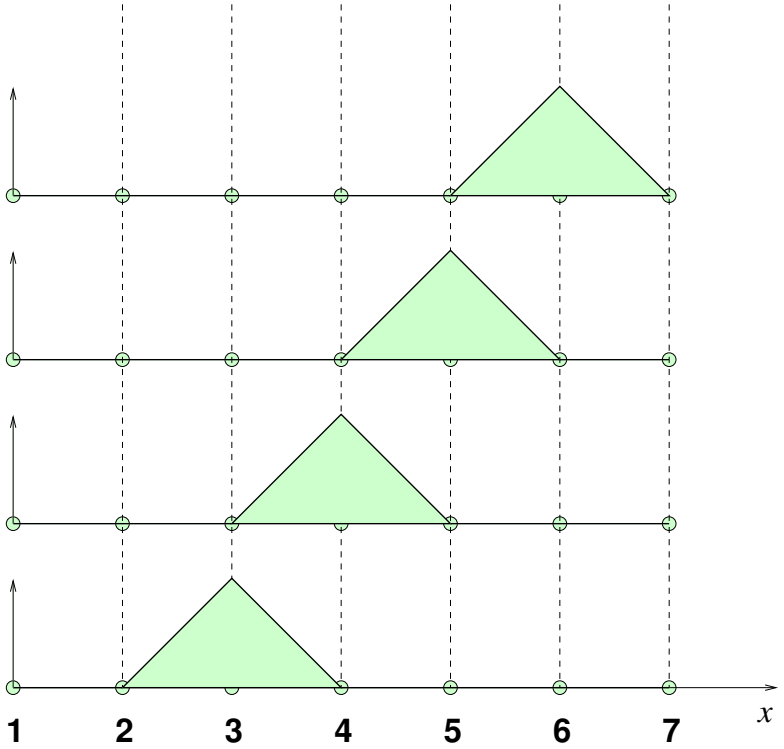
Define elements Ω_e and nodes $x^{[i]}$

The N_i's have the properties:

- N_i is a polynomial over each element, determined uniquely by the *nodal values*
- $N_i(x^{[j]}) = \delta_{i,j}$, i.e., 1 if $i = j$ and zero otherwise
- Hence, $\hat{u}(x^{[i]}) = \sum_j u_j N_j(x^{[i]}) = u_i$

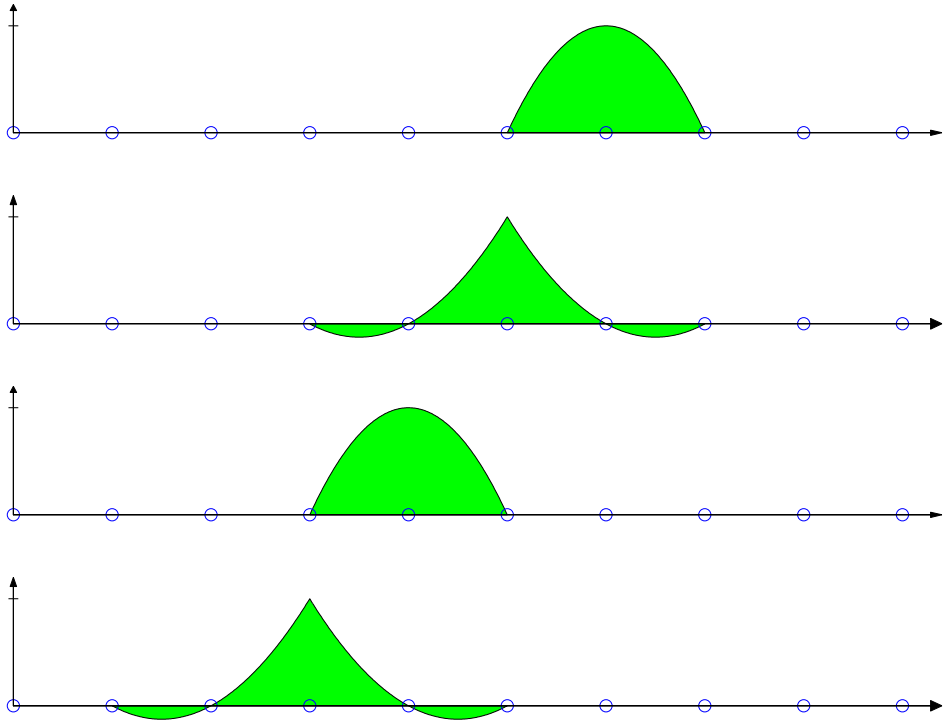
Examples of Basis Functions

Linear basis functions



Each element has two nodes

Quadratic basis functions



Each element has three nodes

Essential Boundary Conditions

Boundary-value problem

$$-u'' = f,\ x \in (0,1), \quad u(0) = u_L,\ u(1) = u_R$$

Expansion with $u_i = \hat{u}(x^{[i]})$:

$$\hat{u}(x) = \psi(x) + \sum_{j=2}^{n-1} \hat{u}_j N_j(x), \quad \psi(x) = u_L N_1(x) + u_R N_n(x)$$

Alternative: skip ψ and enforce $a_1 = u_L$ and $a_n = u_R$ directly in the linear system

This is a general procedure

A Worked Example with Linear Elements

Boundary-value problem

$$-u'' = f,\ x \in (0,1), \quad u(0) = u_L,\ u(1) = u_R$$

Galerkin's method (using integration by parts):

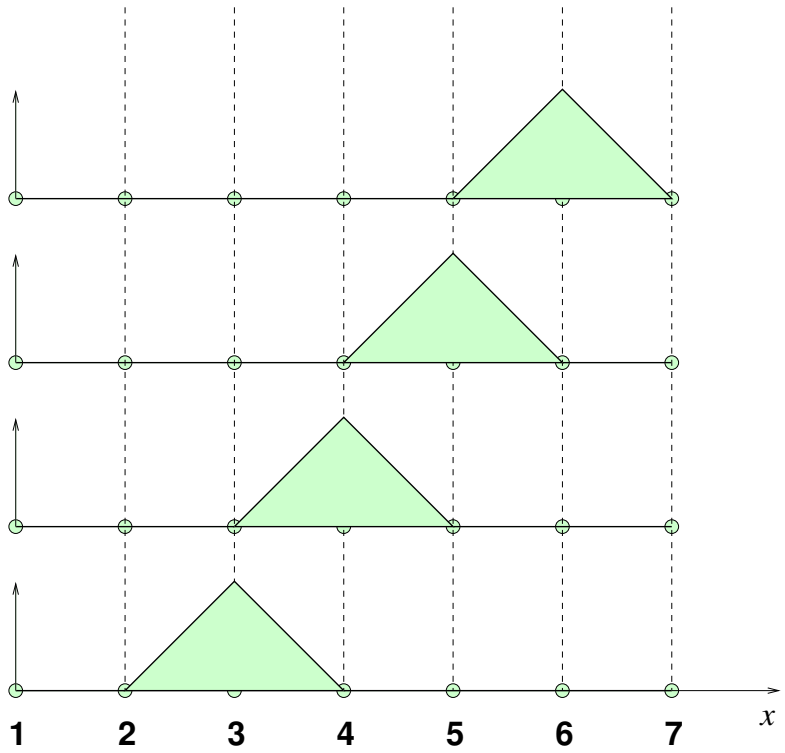
$$\sum_{j=1}^{n} A_{i,j} u_j = b_i, \quad i = 1, \ldots n$$

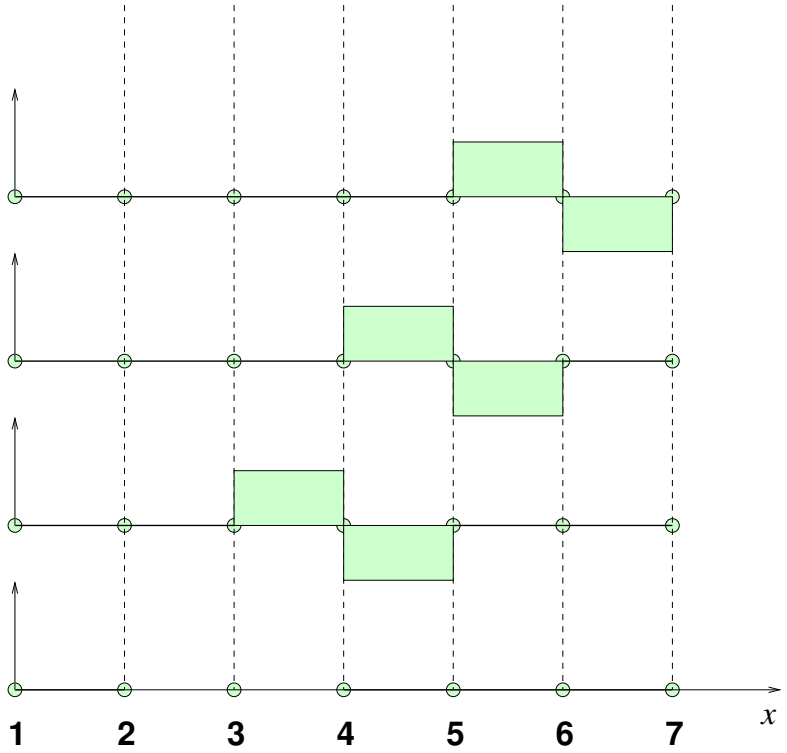
where

$$A_{i,j} = \int_0^1 N_i'(x) N_j'(x) dx, \quad b_i = \int_0^1 f(x) N_i(x) dx$$

Worked Example, cont'd

Observation: $N_i(x)$ and $N_i'(x)$ vanish over large parts of the domain ("nearly" orthogonal functions)





$A_{i,j} = \int_0^1 N_i'(x) N_j'(x) dx \neq 0$ only for $j = i-1, i, i+1$

Example: Evaluation of the Coefficients

Direct computations:

$$A_{i,i-1} = \int_0^1 N'_{i-1} N'_i dx = -\frac{1}{h} \qquad A_{1,1} = A_{n,n} = \frac{1}{h}$$

$$A_{i,i} = \int_0^1 N'_i N'_i dx = \frac{2}{h}, \qquad A_{1,2} = A_{n,n-1} = -\frac{1}{h}$$

$$A_{i,i+1} = \int_0^1 N'_i N'_{i+1} dx = -\frac{1}{h} \qquad b_i = \int_0^1 f(x) N_i(x)\, dx$$

Numerical integration by trapezoidal rule:

$$\int_0^1 f(x) N_i(x)\, dx \approx \frac{1}{2} f(x^{[1]}) N_i(x^{[1]}) h + \sum_{j=1}^{n} f(x^{[j]}) N_i(x^{[j]}) h + \frac{1}{2} f(x^{[n]}) N_i(x^{[n]}) h$$

$$= 0 + \cdots + 0 + f(x^{[i]}) h + 0 + \cdots + 0$$

Example: The Linear Equations

Replace eq. no. 1 and n by boundary conditions

The linear system:

$$
\begin{aligned}
u_1 &= u_L, \\
-\frac{1}{h}u_{i-1} + \frac{2}{h}u_i - \frac{1}{h}u_{i+1} &= f(x^{[i]})h, \quad i = 2, \ldots, n-1, \\
u_n &= u_R
\end{aligned}
$$

Same result as from the finite difference method!

Exact or more accurate numerical integration: different right-hand side term

Element by Element Computations

Split integral into a sum over each element:

$$A_{i,j} = \int_0^1 N_i' N_j' \, dx = \sum_{e=1}^{m} A_{i,j}^{(e)}, \qquad A_{i,j}^{(e)} = \int_{\Omega_e} N_i' N_j' \, dx$$

$$b_i = \int_0^1 f N_i \, dx = \sum_{e=1}^{m} b_i^{(e)}, \qquad b_i^{(e)} = \int_{\Omega_e} f N_i \, dx$$

$A_{i,j}^{(e)} \neq 0$ iff i and j are nodes in element e

$b_i^{(e)} \neq 0$ iff i is node in element e

Element by Element Computations, cont'd

Collect nonzero $A_{i,j}^{(e)}$ in a 2×2 elemental matrix (for piecewise linear elements):

$$\tilde{A}_{r,s}^{(e)}, \quad r, s = 1, 2$$

r, s: local node numbers

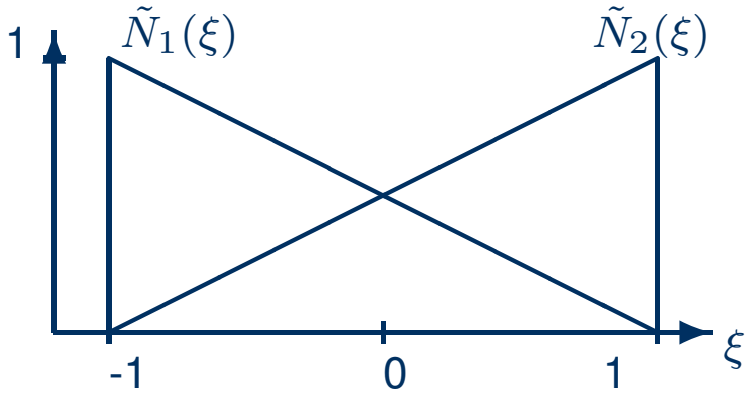
Similar strategy for $b_i^{(e)}$ give the elemental vector $\tilde{b}_r^{(e)}$

Algorithm:

- compute all $\tilde{A}_{r,s}^{(e)}$ and $\tilde{b}_r^{(e)}$,
- combine them to a linear system

Local Coordinates

- Map element $\Omega_e = [x^{[e]}, x^{[e+1]}]$ to $[-1, 1]$
- Define N_i in local ξ coordinates
- Perform all computations in local coordinates



  - Local node r (=1,2) in element e corresponds to global node $i = q(e, r)$
  - Local linear basis functions:

$$\tilde{N}_1(\xi) = \frac{1}{2}(1 - \xi), \quad \tilde{N}_2(\xi) = \frac{1}{2}(1 + \xi)$$

Local Coordinates, cont'd

- Jacobian matrix of mapping: J
- Change integration variable from x to ξ:

$$\int_{x^{[e]}}^{x^{[e+1]}} N_i'(x) N_j'(x)\, dx = \int_{-1}^{1} J^{-1} \frac{d\tilde{N}_r(\xi)}{d\xi} J^{-1} \frac{d\tilde{N}_s(\xi)}{d\xi} \det J\, d\xi$$

- Uniform partition in 1D: $J = \{h/2\}$
- We often write

$$\int_{\Omega_e} \frac{d\tilde{N}_r}{dx} \frac{d\tilde{N}_s}{dx} \det J\, d\xi$$

as the expression in local coordinates, knowing that

$$\frac{d\tilde{N}_r}{dx} = J^{-1} \frac{d\tilde{N}}{d\xi} = \frac{2}{h} \frac{d\tilde{N}}{d\xi}$$

1D Poisson Equation, revisited

- $-u''(x) = f(x)$
- Elemental matrix and vector:

$$\tilde{A}^{(e)}_{r,s} = \int_{-1}^{1} \frac{2}{h} N'_r(\xi) \frac{2}{h} N'_s(\xi) \frac{h}{2}\, d\xi$$

$$\tilde{b}^{(e)}_{r} = \int_{-1}^{1} f(x^{(e)}(\xi)) \tilde{N}_r(\xi) \frac{h}{2}\, d\xi$$

- Example: $r = s = 1$,

$$\tilde{A}^{(e)}_{1,1} = \frac{2}{h} \int_{-1}^{1} (-\frac{1}{2})(-\frac{1}{2})\, d\xi = \frac{1}{h}$$

1D Poisson Equation, cont'd

- Elemental matrix and vector:

$$\left\{\tilde{A}^{(e)}_{r,s}\right\} = \frac{1}{h}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

$$\left\{\tilde{b}^{(e)}_{r}\right\} = \frac{h}{2}\begin{pmatrix} f(x^{(e)}(-1)) \\ f(x^{(e)}(1)) \end{pmatrix}$$

where numerical integration is used:

$$\int_{-1}^{1} g(\xi)\, d\xi \approx g(-1) + g(1)$$

Numerical Integration

Integration rules are normally tabulated for integrals on $[-1, 1]$:

$$\int_{-1}^{1} g(\xi) d\xi \approx \sum_{k=1}^{n_I} g(\xi_k) w_k$$

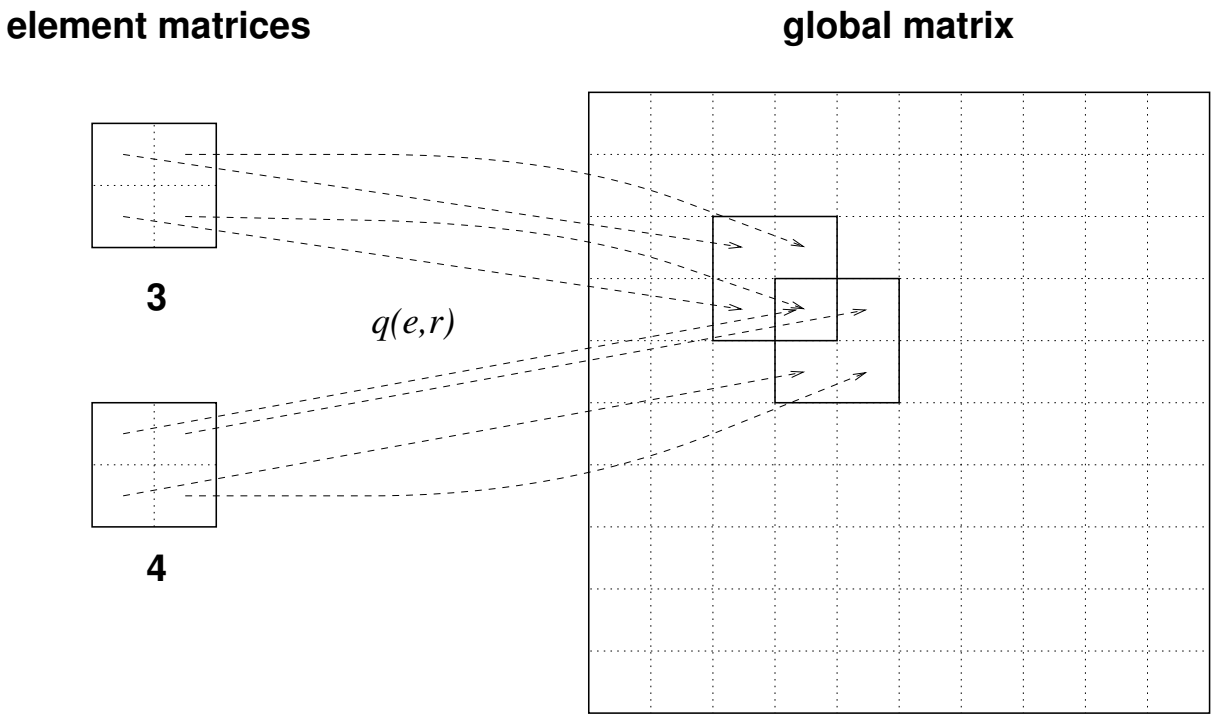
ξ_k: integration points, w_k: integration weights

Some rules:

name	n_I	p	weights	points
Gauss-Legendre	1	1	(1)	(0)
Gauss-Legendre	2	3	$(1, 1)$	$(-1/\sqrt{3}, 1/\sqrt{3})$
Gauss-Legendre	3	5	$(5/9, 8/9, 5/9)$	$(-\sqrt{3/5}, 0, \sqrt{3/5})$
Gauss-Lobatto	2	2	$(1, 1)$	$(-1, 1)$
Gauss-Lobatto	3	3	$(1/3, 4/3, 1/3)$	$(-1, 0, 1)$

The rules integrate polynomials of degree p exactly

Assembly



Elemental matrices and vectors must be assembled in the global system. Essential: local $\rightarrow$ global mapping, $q(e,r)$

Algorithm:

$$A_{q(e,r),q(e,s)} := A_{q(e,r),q(e,s)} + \tilde{A}^{(e)}_{r,s}, \quad r,s = 1,2$$

$$b_{q(e,r)} := b_{q(e,r)} + \tilde{b}^{(e)}_{r}, \quad r = 1,2$$

Summing up the Procedures

- Weighted residual formulation, often Galerkin's choice with $W_i = N_i$
- Integration by parts
- Derivative boundary conditions in boundary terms
- Compute elemental matrices and vectors
  - Local coordinates with local numbering
  - Numerical integration
  - Enforce essential boundary conditions
  - Assemble local contributions
- Solve linear system

The Elementwise Algorithm:

initialize global linear system:

set $A_{i,j} = 0$ for $i, j = 1, \ldots, n$, $\quad b_i = 0$ for $i = 1, \ldots, n$

loop over all elements:

for $e = 1, \ldots, m$

set $\tilde{A}^{(e)}_{r,s} = 0$, $r, s = 1, \ldots, n_e$, $\quad$ set $\tilde{b}^{(e)}_r = 0$, $r = 1, \ldots, n_e$

loop over numerical integration points:

for $k = 1, \ldots, n_I$

evaluate $\tilde{N}_r(\xi_k)$, derivatives of $\tilde{N}_r$ wrt. ξ and x, J

contribution to elemental matrix and vector from the current integration point

for $r = 1, \ldots, n_e$

for $s = 1, \ldots, n_e$

$$\tilde{A}^{(e)}_{r,s} := \tilde{A}^{(e)}_{r,s} + \frac{d\tilde{N}_r}{dx}\frac{\tilde{N}_s}{dx} \det J w_k$$

$$\tilde{b}^{(e)}_r := \tilde{b}^{(e)} + f(x^{(e)}(\xi_k)) N_r \det J w_k$$

incorporate essential boundary conditions:

for $r = 1, \ldots, n_e$

if node r has an essential boundary condition then

modify $\tilde{A}^{(e)}_{r,s}$ and $\tilde{b}^{(e)}_r$ due to this condition

assemble:

for $r = 1, \ldots, n_e$

for $s = 1, \ldots, n_e$

$$A_{q(e,r),q(e,s)} := A_{q(e,r),q(e,s)} + \tilde{A}^{(e)}_{r,s}$$

$$b_{q(e,r)} := b_{q(e,r)} + \tilde{b}^{(e)}_r$$

2D Domains

Strength of the finite element method:
easy to work with geometrically complicated domains

Lake Superior with 6 islands, 2330 triangles