FEM08 - lecture 2

Johan Jansson

jjan@csc.kth.se

CSC

KTH

Galerkin's method

$$(R(u), v)_{L_2} = (-(au')' - f, v) = 0, \quad x \in [a, b], \quad \forall v \in V_h$$

Technical step:
Integrate by parts (move derivative to test function)

- Linear approximation only has one derivative
- Simplifies enforcement of boundary conditions

Galerkin's method

Recall integration by parts:

$$\int_a^b w'v dx = -\int_a^b wv' dx + w(b)v(b) - w(a)v(a)$$

$$R(u) = -(au')' - f$$
$$(R(u), v) = \int_a^b -(au')'v - fv dx =$$
$$\int_a^b (au')v' - fv dx + u'(b)v(b) - u'(a)v(a)$$

For homogenous Dirichlet BC we can use $v(a) = v(b) = 0$

Galerkin's method

Insert piecewise linear approximation:

$$U(x) = \sum_{j=1}^{M} \xi_j \phi_j(x)$$

We are left to solve:

$$\int_a^b (aU')v' - fv dx = 0, \quad x \in [a, b], \quad \forall v \in V_h$$

Or equivalently:

$$\int_a^b (aU')\phi_i' - f\phi_i dx = 0,$$
$$x \in [a, b], \quad i = 1, ..., M$$

Discrete system

Substituting U:

$$\int_a^b (a(\sum_{j=1}^{M} \xi_j \phi_j)')\phi_i' - f\phi_i dx = 0,$$

$$x \in [a,b], i = 1, ..., M$$

Clean up:

$$\sum_{j=1}^{M} \int_a^b a\xi_j \phi_j' \phi_i' - f\phi_i dx = 0,$$

$$x \in [a,b], i = 1, ..., M$$

Discrete system

Left with algebraic system in $\xi = (\xi_1, ..., \xi_M)^\top$:

$$F(\xi) = 0$$

In this case F is a linear system $F(\xi) = A\xi - b$ with:

$$A_{ij} = \sum_{j=1}^{M} \int_a^b a\phi_j'\phi_i' dx,$$

$$b_i = \int_a^b -f\phi_i dx$$

Solve for ξ, costruct solution function $U(x) = \sum_{j=1}^M \xi_j\phi_j(x)$
If F is not linear, can use Newton's method.

Piecewise polynomials in 2D

Construct triangulation T of domain Ω

Define size of triangle $K \in T$ is h_K as diameter of triangle

Define N as node (in this case vertex of triangle)

Want to define basis functions for vector space V_h: space of piecewise linear functions on T

Requirement for nodal basis:

$$\phi_j(N_i) = \begin{cases} 1, & i = j, \\ 0, & i \neq j, \end{cases} \quad i, j = 1, ..., M \tag{1}$$

Piecewise polynomials in 2D

Define local basis functions v^i on triangle K with vertices $a^i = (a^i_1, a^i_2)$, $i = 1, 2, 3$

v is linear $\Rightarrow v(x) = c_0 + c_1 x_1 + c_2 x_2$

Values of v in vertices: $v_i = v(a^i)$ (1 or 0)

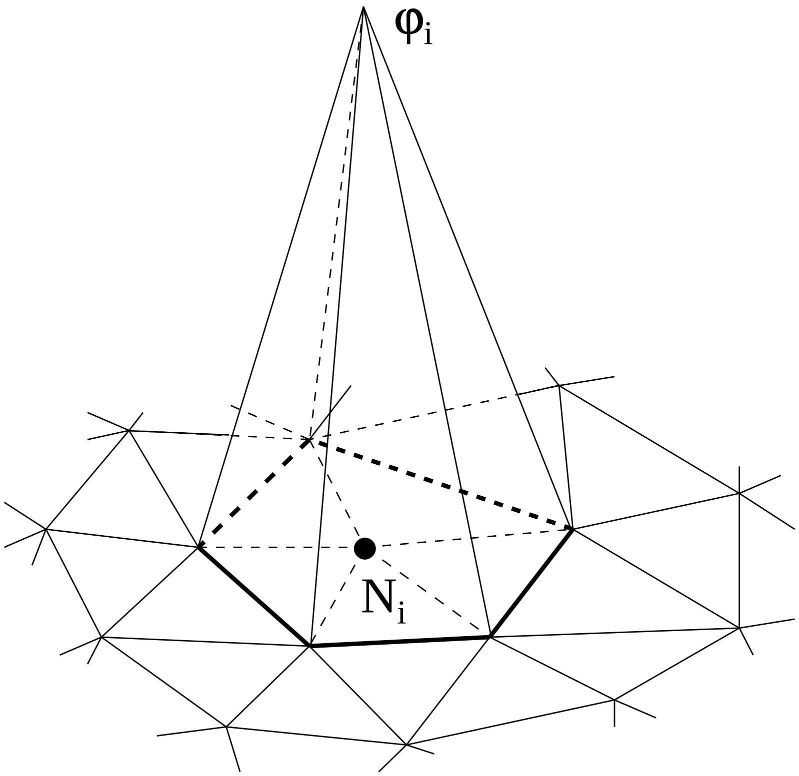
Linear system for coefficients c:

$$\begin{pmatrix} 1 & a^1_1 & a^1_2 \\ 1 & a^2_1 & a^2_2 \\ 1 & a^3_1 & a^3_1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}.$$

Piecewise polynomials in 2D

Sum local basis functions:

$$\phi_i = \sum_j v^j, \quad N_i = a_j \tag{2}$$



Poisson in 2D

Automated discretization in FEniCS

General bilinear form $a(\cdot,\cdot)$

In general the matrix A_h, representing a bilinear form

$$a(u,v) = (A(u),v),$$

is given by

$$(A_h)_{ij} = a(\varphi_j, \hat{\varphi}_i).$$

and the vector b_h representing a linear form

$$L(v) = (f,v),$$

is given by

$$(b_h)_i = L(\hat{\varphi}_i).$$

Assembling the matrices

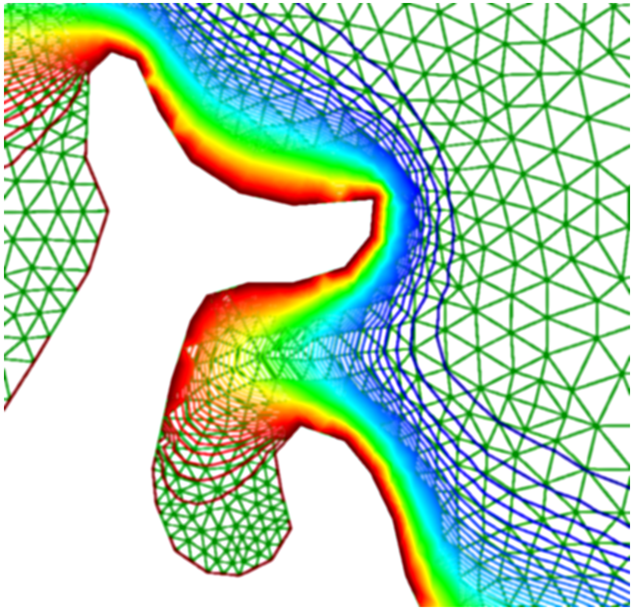
Computing $(A_h)_{ij}$

Note that

$$(A_h)_{ij} \quad = \quad a(\varphi_j, \hat{\varphi}_i) = \sum_{K \in \mathcal{T}} a(\varphi_j, \hat{\varphi}_i)_K.$$

Iterate over all elements K and for each element K compute the contributions to all $(A_h)_{ij}$, for which φ_j and $\hat{\varphi}_i$ are supported within K.

Assembly of discrete system



Noting that $a(v,u) = \sum_{K \in \mathcal{T}} a_K(v,u)$, the matrix A can be assembled by

$A = 0$
for all elements $K \in \mathcal{T}$
  A += A^K

The *element matrix* A^K is defined by

$$A_{ij}^K = a_K(\hat{\phi}_i, \phi_j)$$

for all local basis functions $\hat{\phi}_i$ and ϕ_j on K

Assembling A_h

for all elements $K \in \mathcal{T}$

for all test functions $\hat{\varphi}_i$ on K

for all trial functions φ_j on K

1. Compute $I = a(\varphi_j, \hat{\varphi}_i)_K$
2. Add I to $(A_h)_{ij}$

end

end

end

Assembling b

for all elements $K \in \mathcal{T}$

for all test functions $\hat{\varphi}_i$ on K

1. Compute $I = L(\hat{\varphi}_i)_K$
2. Add I to b_i

end

end

Mapping from a reference element

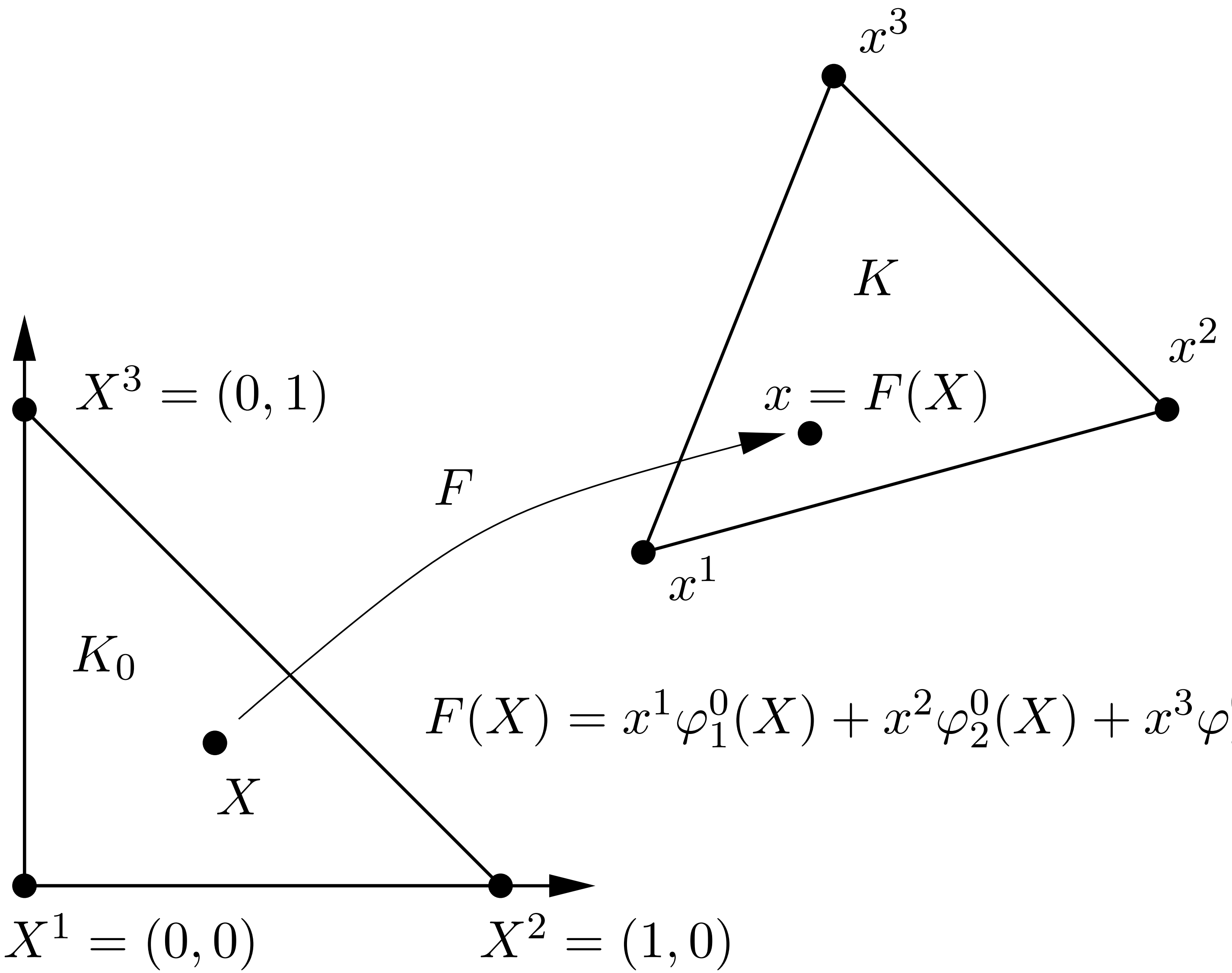
Isoparametric mapping

- We want to compute basis functions and integrals on a *r*eference element K_0
- Most common mapping is isoparametric mapping (use the basis functions also to define the geometry):

$$x(X) = F(X) = \sum_{i=1}^{n} \phi_i(X) x_i$$

- Linear basis functions $\Rightarrow$
Affine mapping: $x(X) = F(X) = BX + b$

The mapping $F : K_0 \to K$



$$F(X) = x^1\varphi_1^0(X) + x^2\varphi_2^0(X) + x^3\varphi_3^0(X)$$

Integration: coordinate transform

Let $v = v(x)$ be a function defined on a domain Ω and let

$$F : \Omega_0 \to \Omega$$

be a (differentiable) mapping from a domain Ω_0 to Ω. We then have $x = F(X)$ and

$$\begin{aligned} \int_\Omega v(x)\, dx &= \int_{\Omega_0} v(F(X))\, |\det \partial F_i / \partial X_j|\, dX \\ &= \int_{\Omega_0} v(F(X))\, |\det \partial x / \partial X|\, dX. \end{aligned}$$

Affine mapping

When the mapping is affine, the determinant is constant:

$$
\begin{aligned}
& \int_K \varphi_j(x)\hat{\varphi}_i(x)\, dx \\
= & \int_{K_0} \varphi_j(F(X))\hat{\varphi}_i(F(X))\, |\det \partial x/\partial X|\, dX \\
= & \, |\det \partial x/\partial X| \int_{K_0} \varphi_j^0(X)\hat{\varphi}_i^0(X)\, dX
\end{aligned}
$$

Transformation of derivatives

To compute derivatives, we use the transformation

$$\nabla_X = \left(\frac{\partial x}{\partial X}\right)^{\top} \nabla_x,$$

or

$$\nabla_x = \left(\frac{\partial x}{\partial X}\right)^{-\top} \nabla_X.$$

The stiffness matrix

For the computation of the stiffness matrix, this means that we have

$$\begin{aligned}
& \int_K \epsilon(x) \nabla \varphi_j(x) \cdot \nabla \hat{\varphi}_i(x) \, dx \\
= \quad & \int_{K_0} \epsilon_0(X) \left[(\partial x / \partial X)^{-\top} \nabla_X \varphi_j^0(X) \right] \cdot \left[(\partial x / \partial X)^{-\top} \nabla_X \hat{\varphi}_i^0(X) \right] \cdots \\
& \quad \cdots \left| \det \left(\partial x / \partial X \right) \right| \, dX.
\end{aligned}$$

Note that we have used the short notation $\nabla = \nabla_x$.

in the affine case the $\partial x / \partial X$ are simply elements of the matrix B in $x(X) = F(X) = BX + b$

Computing integrals on K_0

- The integrals on K_0 can be computed symbolically or by quadrature.
- In some cases quadrature is the only option.
- Note that basis functions and products of basis functions can be integrated exactly with quadrature (if polynomial)

Standard form:

$$\int_{K_0} v(X)\, dX \approx |K_0| \sum_{i=1}^{n} w_i v(X^i)$$

where $\{w_i\}_{i=1}^n$ are quadrature weights and $\{X^i\}_{i=1}^n$ are quadrature points in K_0.

FEniCS: Example syntax (Poisson)

```
# The bilinear form a(v, u) and linear form L(v) for
# Poisson's equation, 2D version

mesh = UnitSquare(32, 32)

element = FiniteElement("Lagrange", "triangle", 1)

v = TestFunction(element)
u = TrialFunction(element)
f = Source(element, mesh)
g = Flux(element, mesh)

a = dot(grad(v), grad(u))*dx
L = v*f*dx + v*g*ds
```

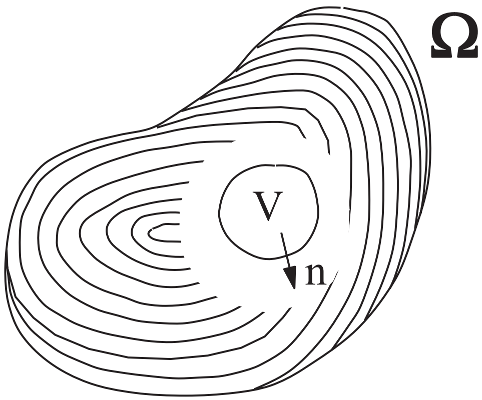
Poisson in 2D

We first model *heat conduction* in a heat-conducting material occupying the volume Ω in ^{3} with boundary Γ, over a time interval $I = [0, T]$. We let $u(x,t)$ denote the *temperature* and $q(x,t)$ the *heat flux* at the point x at time t. The heat flux is a vector $q = (q_1, q_2, q_3)$, where q_i is the heat flux, or rate of heat flowing in the direction x_i. We let $f(x,t)$ denote the rate of heat (per unit of volume) supplied at (x,t) by a *heat source*.

Poisson in 2D

We derive the model using a basic *conservation law* expressing *conservation of heat* in the following form: for any fixed domain V in Ω with boundary S, the rate of the total heat introduced in V by the external source is equal to the rate of the total heat accumulated in V plus the total heat flux through S.
This is based on the conviction that the heat introduced in V by the external source can choose between two options only: (i) flow out of V or (ii) be accumulated in V. With S denoting the boundary of V and n denoting the outward unit normal to S.

Poisson in 2D



Figur 1: An arbitrary subset V of a heat conducting body Ω.

The conservation law can be expressed as

$$\int_V f\,dx = \frac{\partial}{\partial t}\int_V \lambda u\,dx + \int_S q\cdot n\,ds, \tag{3}$$

where $\lambda(x,t)$ is the *heat capacity coefficient* and all functions are evaluated at a specific time $t \in I$.

Poisson in 2D

By the Divergence theorem,

$$\int_S q \cdot n \, ds = \int_V \nabla \cdot q \, dx,$$

and combined with heat balance, this implies that

$$\int_V \left(\frac{\partial}{\partial t}(\lambda u) + \nabla \cdot q \right) dx = \int_V f \, dx,$$

where the time derivative could be moved under the integral sign because V does not depend on time t.

Poisson in 2D

Since V is arbitrary, assuming the integrands are Lipschitz continuous, it follows that

$$\frac{\partial}{\partial t}(\lambda u)(x,t) + \nabla \cdot q(x,t) = f(x,t) \quad \text{for all } x \in \Omega,\ 0 < t \leq T, \tag{4}$$

which is a differential equation describing *conservation of heat* involving two unknowns: the temperature $u(x,t)$ and the heat flux $q(x,t)$. We thus have one equation and two unknowns and we need yet another equation.

Poisson in 2D

The second equation is a *constitutive equation* that couples the heat flux q to the temperature gradient ∇u. *Fourier's law* states that heat flows from warm to cold regions with the heat flux proportional to the temperature gradient:

$$q(x,t) = -a(x,t)\nabla u(x,t) \quad \text{for } x \in \Omega,\, 0 < t \leq T \tag{5}$$

where the factor of proportionality $a(x,t)$ is the coefficient of heat conductivity. Note the minus sign indicating that the heat flows from warm to cold regions, and that the heat conductivity $a(x,t)$ is positive.

Poisson in 2D

Combining diffeqheatbalance and fourierslaw, we obtain the basic differential equation describing heat conduction:

$$\frac{\partial}{\partial t}(\lambda u) - \nabla \cdot (a \nabla u) = f \quad \text{in } \Omega \times (0, T], \tag{6}$$

where $a(x,t)$ and $\lambda(x,t)$ are given positive coefficients depending on (x,t) and $f(x,t)$ is a given heat source, and the unknown $u(x,t)$ represents the temperature.

Poisson in 2D

To define the solution uniquely, the differential equation is complemented by initial and boundary conditions. The complete model with *Dirichlet boundary conditions* reads

$$\begin{cases} \frac{\partial}{\partial t}(\lambda u) - \nabla \cdot (a \nabla u) = f & \text{in } \Omega \times (0,T], \\ u = u_b & \text{on } \Gamma \times (0,T], \\ u(x,0) = u_0(x) & \text{for } x \in \Omega, \end{cases} \tag{7}$$

where u_0 is the initial temperature and u_b is the boundary temperature.

Poisson in 2D

The Dirichlet boundary condition corresponds to immersing the body Ω in a large reservoir with a specified temperature u_b and assuming that the boundary acts as a perfect thermal conductor so that the temperature of the body on the boundary is equal to the specified outside reservoir temperature u_b. Note that the given boundary temperature $u_b = u_b(x, t)$ may vary with (x, t).

Poisson in 2D

Other commonly encountered boundary conditions are *Neumann* and *Robin* boundary conditions. A Neumann boundary condition corresponds to prescribing the heat flux $q \cdot n$ across (out of) the boundary:

$$q \cdot n = -a\nabla u \cdot n = -a\frac{\partial u}{\partial n} = -a\partial_n u = g \quad \text{on } \Gamma,$$

with g given. A *homogeneous Neumann boundary condition* with $g = 0$ corresponds to a perfectly insulating boundary with the heat flux across the boundary being zero.

Poisson in 2D

A homogenous Robin boundary condition is intermediate with the boundary neither being perfectly conducting nor perfectly insulated, with the heat flux through the boundary being proportional to the difference of the temperature u inside and a given temperature u_b outside Ω:

$$-a\partial_n u = \kappa(u - u_b)$$

with κ a positive coefficient representing the heat conductivity of the boundary.

Poisson in 2D

Partitioning the boundary Γ into disjoint pieces Γ_1, Γ_2 and Γ_3 with different types of boundary conditions, the *general initial boundary value problem* IBVP for the heat equation has the form,

$$\begin{cases} \frac{\partial}{\partial t}(\lambda u) - \nabla \cdot (a \nabla u) = f & \text{in } \Omega \times (0,T], \\ u = u_b & \text{on } \Gamma_1 \times (0,T], \\ -a\partial_n u = g & \text{on } \Gamma_2 \times (0,T], \\ a\partial_n u + \kappa(u - u_b) = 0 & \text{on } \Gamma_3 \times (0,T], \\ u(x,0) = u_0(x) & \text{for } x \in \Omega, \end{cases} \tag{8}$$

where u_b represents a given “exterior” boundary temperature, and g represents a given outward normal heat flux on the boundary.