FEM12 - Lecture 1

Johan Jansson

jjan@kth.se

CSC

KTH

Course overview

- Science - differential equations
- Function approximation using polynomials
- Galerkin’s method (finite element method)
- Assembly of discrete systems
- Error estimation
- Mesh operations
- Stability
- Existence and uniqueness of solutions

Course structure

- Course divided into self-contained modules (from preceding slide)
- Module:
  - Theory
  - Software
  - Examination: write report (theory + software)

Science - modeling

Science: modeling (formulating equations) + computation (solving equations)

- Model natural laws (primarily) in terms of differential equations
- Partial differential equation (PDE):

$$A(u(x)) = f, \quad x \in \Omega$$

with A differential operator.

Initial value problem $u(x_0) = g$ (x is “time”, $\Omega = [0, T]$)

Boundary value problem $u(x) = g, \quad x \in \Gamma$ or
$(\nabla u(x)) \cdot n = g, \quad x \in \Gamma$ (x is “space”)

Boundary value problem $u(x) = g, \quad x \in \Gamma$ (x is “space”)

Initial boundary value problem Both are also possible

Science - computation

Finite Element Method (FEM): approximate solution function u as (piecewise) polynomial.

Compute coefficients by enforcing orthogonality (Galerkin's method).

Implement general algorithms for arbitrary differential equations

In this course we will use and understand a general implementation for discretizing PDE with FEM: FEniCS using the Python programming language.

Free software / Open source implementations

Science/FEM - examples

Newton's 2nd law: $F = ma, u = (u_1, u_2)$:

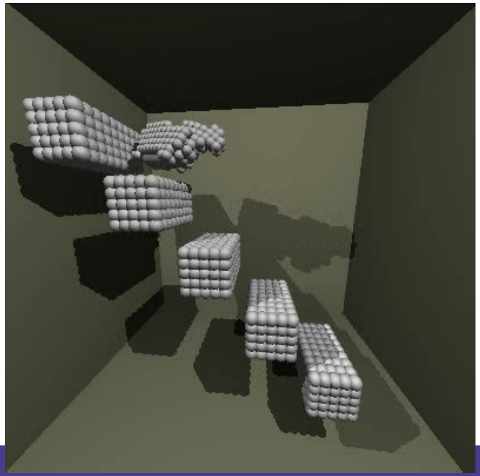
$$
\begin{aligned}
\dot{u}_1(t) &= u_2(t) \\
\dot{u}_2(t) &= F(u(t)) \\
u(0) &= u_0, \quad t \in [0, T]
\end{aligned}
$$

Science/FEM - examples

Incompressible Navier-Stokes

$$
\begin{aligned}
\dot{u} + u \cdot \nabla u - \nu \Delta u + \nabla p &= f \\
\nabla \cdot u &= 0
\end{aligned}
$$

Elasticity - solid mechanics



$$
\dot{u} + \nabla \cdot \sigma = f
$$

Polynomial approximation

Systematic method for computing approximate solutions:

We seek polynomial approximations U to u.

A vector space can be constructed with set of polynomials on domain (a, b) as basis vectors, where function addition and scalar multiplication satisfy the requirements for a vector space.

We can also define an inner product space with the L_2 inner product defined as:

$$(f, g)_{L_2} = \int_\Omega f(x)g(x)dx$$

Polynomial approximation

The L_2 inner product generates the L_2 norm:

$$\|f\|_{L_2} = \sqrt{(f,f)}_{L_2}$$

Just like in R^d we define orthogonality between two vectors as:

$$(f,g)_{L_2} = 0$$

We also have Cauchy-Schwartz inequality:

$$|(f,g)_{L_2}| \leq \|f\|_{L_2} \|g\|_{L_2}$$

Basis

We call our polynomial vector space $V^q = P^q(a,b)$ consisting of polynomials:

$$p(x) = \sum_{i=0}^{q} c_i x^i$$

One basis is the monomials: $\{1, x, ..., x^q\}$

Piecewise linear polynomials

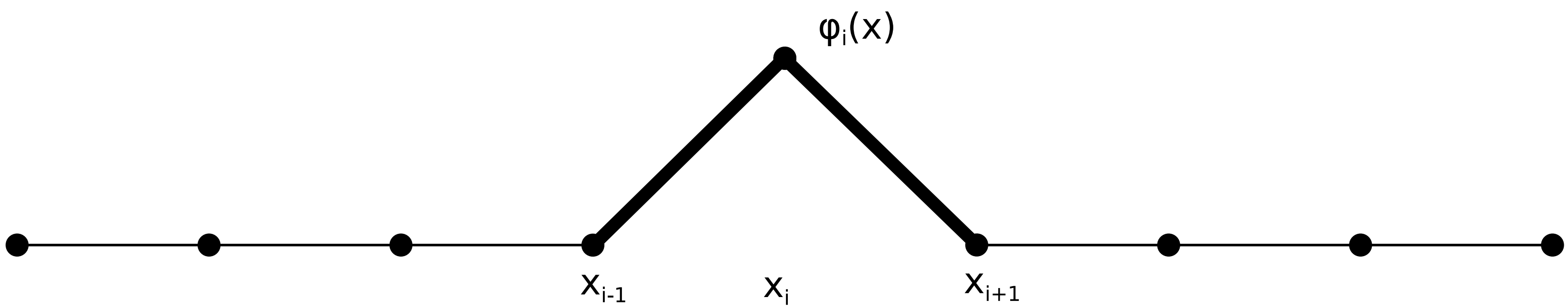
Global polynomials on the whole domain (a, b) led to vector space V^q (monomial basis: $\{1, x, ..., x^q\}$). Only way of refining approximate solution U is by increasing q.

We instead look at *piecewise* polynomials.

Partition domain $I = (a, b)$ into mesh:
$a = x_0 < x_1 < x_2 < \cdots < x_{m+1} = b$ by placing *nodes* x_i.

Define polynomial function on each subinterval $I_i = (x_{i-1}, x_i)$ with length $h_i = x_i - x_{i-1}$.

Piecewise linear polynomials



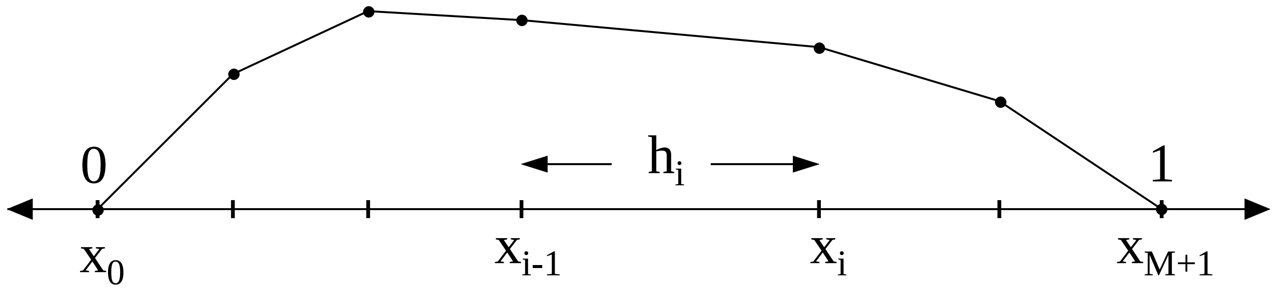
Nodal basis: $\phi_i(x_i) = 1$, $\phi_i(x_j) = 0$, $i \neq j$

Basis function $\phi_i(x)$:

$$\phi_i(x) = \begin{cases} 0, & x \notin [x_{i-1}, x_{i+1}], \\ \frac{x - x_{i-1}}{x_i - x_{i-1}}, & x \in [x_{i-1}, x_i], \\ \frac{x - x_{i+1}}{x_i - x_{i+1}}, & x \in [x_i, x_{i+1}]. \end{cases}$$

Vector space of continuous piecewise linear polynomials: V_h with basis $\{\phi_i\}_1^M$, M number of nodes in mesh.

Piecewise linear polynomials



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

Equation

What do we mean by equation?

We define the *residual* function $R(U)$ as:
$R(U) = A(U) - f$

We can thus define an *equation* with exact solution u as:
$R(u) = 0$

Galerkin's method

We seek a solution U in finite element vector space V_h of the form:

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

We require the residual to be orthogonal to V_h:

$$(R(U), v) = 0, \forall v \in V_h \quad (2)$$

For terms in $R(u)$ with two derivatives we perform integration by parts to move one derivative to the test function (piecewise linear functions only have one derivative).