Algorithms of Scientific Computing

Finite Element Methods

Michael Bader

Summer Term 2012

Part I

Looking Back: Discrete Models for Heat Transfer and the Poisson Equation

Modelling of Heat Transfer

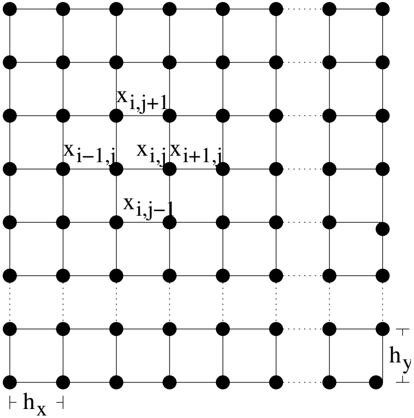
- objective: compute the temperature distribution of some object
- under certain prerequisites:
  - temperature T at object boundaries given
  - heat sources
  - material parameters $k, \ldots$
- observation from physical experiments:

$$q \approx k \cdot \delta T$$

heat flow proportional to temperature differences

A Wiremesh Model

- consider rectangular plate as fine mesh of wires
- compute temperature $T_{ij} \approx T(x_{ij})$ at nodes x_{ij} of the mesh



Wiremesh Model (2)

- model assumption:

  temperatures in equilibrium at every mesh node

- for all temperatures T_{ij}:

$$T_{ij} = \frac{1}{4}\left(T_{i-1,j} + T_{i+1,j} + T_{i,j-1} + T_{i,j+1}\right)$$

- temperature known at (part of) the boundary; for example:

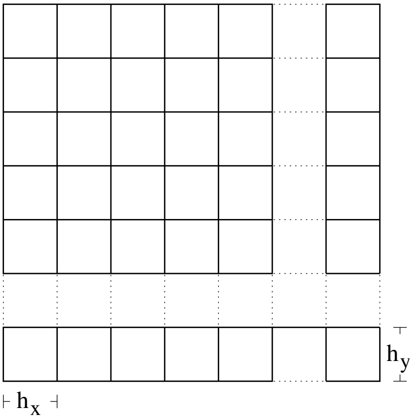
$$T_{0,j} = 0 \qquad T_{n,j} = 1$$

- task: solve system of linear equations
- in standard notation:

$$-T_{i-1,j} - T_{i+1,j} + 4T_{ij} - T_{i,j-1} - T_{i,j+1} = 0$$

A Finite Volume Model

- object: a rectangular metal plate (again)
- model as a collection of small connected rectangular cells



- examine the heat flow across the cell edges

Heat Flow Across the Cell Boundaries

- Heat flow across a given edge is proportional to
  - temperature difference $(T_1 - T_0)$ between the adjacent cells
  - length h of the edge
- e.g.: heat flow across the left edge:

$$q_{ij}^{(\mathrm{left})} = k_x \left(T_{ij} - T_{i-1,j}\right) h_y$$

  k_x depends on material
- heat flow across all edges determines change of heat energy:

$$\begin{aligned} q_{ij} &= k_x \left(T_{ij} - T_{i-1,j}\right) h_y + k_x \left(T_{ij} - T_{i+1,j}\right) h_y \\ &+ k_y \left(T_{ij} - T_{i,j-1}\right) h_x + k_y \left(T_{ij} - T_{i,j+1}\right) h_x \end{aligned}$$

Discrete and Continuous Model

- consider source term $F_{ij} = f_{ij} h_x h_y$ (f_{ij} heat flow per area)
- equilibrium with source term requires $q_{ij} + F_{ij} = 0$:

$$
\begin{aligned}
f_{ij} h_x h_y \quad = \quad & -k_x h_y \left(2T_{ij} - T_{i-1,j} - T_{i+1,j}\right) \\
& -k_y h_x \left(2T_{ij} - T_{i,j-1} - T_{i,j+1}\right)
\end{aligned}
$$

- system of equations derived from the discrete model:

$$
\begin{aligned}
f_{ij} \quad = \quad & -\frac{k_x}{h_x} \left(2T_{ij} - T_{i-1,j} - T_{i+1,j}\right) \\
& -\frac{k_y}{h_y} \left(2T_{ij} - T_{i,j-1} - T_{i,j+1}\right)
\end{aligned}
$$

- corresponds to *partial differential equation* (PDE):

$$
-k \left(\frac{\partial^2 T(x,y)}{\partial x^2} + \frac{\partial^2 T(x,y)}{\partial y^2} \right) = f(x,y)
$$

Part II

Finite Element Methods

For *Model Problem*:

- 2D Poisson equation:

$$\frac{\partial^2 T(x,y)}{\partial x^2} + \frac{\partial^2 T(x,y)}{\partial y^2} = f(x,y)$$

- first, however, we consider the 1D case:

$$u''(x) = f(x) \qquad \text{for } x \in (0,1)$$

with $u(0) = u(1) = 0$.

Finite Elements – Main Ingredients

1. compute a *function* as numerical solution;
   search in a function space W_h:

$$u_h = \sum_j u_j \varphi_j(x), \qquad \mathrm{span}\{\varphi_1, \ldots, \varphi_J\} = W_h$$

2. solve *weak form* of PDE to reduce regularity properties

$$u'' = f \quad \longrightarrow \quad \int v' u' \,\mathrm{dx} = \int vf \,\mathrm{dx}$$

3. choose basis functions with *local support*, e.g.:

$$\varphi_j(x_i) = \delta_{ij}$$

Choose Test and Ansatz Space

- search for solution functions u_h of the form

$$u_h = \sum_j u_j \varphi_j(x)$$

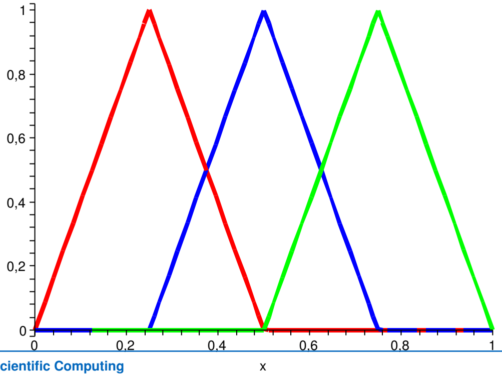
- the basis (ansatz) functions $\varphi_j(x)$ build a vector space (or function space) W_h

$$\text{span}\{\varphi_1, \ldots, \varphi_J\} = W_h$$

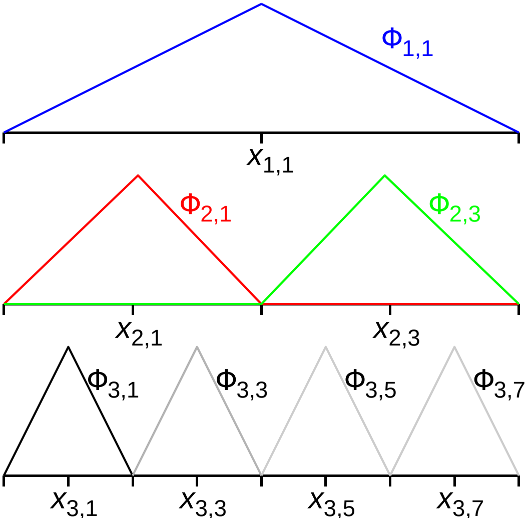
- the “best” solution u_h in this function space is wanted

Example: Nodal Basis

$$\varphi_i(x) := \begin{cases} \frac{1}{h}(x - x_{i-1}) & x_{i-1} < x < x_i \\ \frac{1}{h}(x_{i+1} - x) & x_i < x < x_{i+1} \\ 0 & \text{otherwise} \end{cases}$$



Or Better A Hierarchical Basis?



Weak Forms and Weak Solutions

- consider a PDE $Lu = f$ (e.g. $Lu = \Delta u$)
- transformation to the *weak form*:

$$\langle v, Lu \rangle = \int vLu \,\mathrm{dx} = \int vf \,\mathrm{dx} = \langle f, v \rangle \quad \forall v \in V$$

  V a certain class of functions
- "real solution" u also solves the weak form
  (but additional solutions accepted ...)
- motivation for weak form:
  - we cannot test $Lu(x) = f(x)$ for all $x \in (0, 1)$ on a computer (infinitely many x)
  - frequent choice $V = W_h$, so check whether Lu and f have the "same behaviour" w.r.t. scalar product
  - next slide: additional type of functions may "qualify" as solution

Weak Form of the Poisson Equation – 1D

- Poisson equation with Dirichlet conditions:

$$-u''(x) = f(x) \quad \text{in } \Omega = (0,1), \qquad u(0) = u(1) = 0$$

- weak form:

$$-\int_\Omega v(x)u''(x)\,\mathrm{dx} = \int_\Omega v(x)f(x)\,\mathrm{dx} \quad \forall v$$

- integration by parts:

$$-\int_\Omega v(x)u''(x)\,\mathrm{dx} = -v(x)\cdot u'(x)\Big|_0^1 + \int_\Omega v'(x)\cdot u'(x)\,\mathrm{dx}$$

- choose functions v such that $v(0) = v(1) = 0$:

$$\int_\Omega v'(x)\cdot u'(x)\,\mathrm{dx} = \int_\Omega v(x)f(x)\,\mathrm{dx} \quad \forall v$$

Weak Form of the Poisson Equation – 2D/3D

- Poisson equation with Dirichlet conditions:

$$-\Delta u = f \quad \text{in } \Omega, \qquad u = 0 \quad \text{on } \delta\Omega$$

- weak form:

$$-\int_\Omega v \Delta u \, d\Omega = \int_\Omega v f \, d\Omega \quad \forall v$$

- apply Green's formula:

$$-\int_\Omega v \Delta u \, d\Omega = \int_\Omega \nabla v \cdot \nabla u \, d\Omega - \int_{\partial\Omega} v \cdot \nabla u \, ds$$

- choose functions v such that $v = 0$ on $\partial\Omega$:

$$\int_\Omega \nabla v \cdot \nabla u \, d\Omega = \int_\Omega v f \, d\Omega \quad \forall v$$

Weak Form of the Poisson Equation – Summary

- Poisson equation with Dirichlet conditions:

$$-\Delta u = f \quad \text{in } \Omega, u = 0 \quad \text{on } \delta\Omega$$

- transformed into weak form:

$$\int_\Omega \nabla v \cdot \nabla u \, \mathrm{d}\Omega = \int_\Omega v f \, \mathrm{d}\Omega \quad \forall v$$

- weaker requirements for a solution u:
  twice differentiabale $\rightarrow$ *first derivative integrable*
- remember use of nodal basis: availability of first vs. second derivative!

Choose Test and Ansatz Space

- search for solutions u_h in a function space W_h:

$$u_h = \sum_j u_j \varphi_j(x)$$

  where $\text{span}\{\varphi_j\} = W_h$ (“ansatz space”)

- insert into weak solution

$$\int vL\left(\sum_j u_j \varphi_j(x)\right) \mathrm{dx} = \int vf \, \mathrm{dx} \quad \forall v \in V$$

Choose Test and Ansatz Space (2)

- choose a basis $\{\psi_i\}$ of the *test* space V
- then: if all basis functions ψ_i satisfy

$$\int \psi_i L\Big(\sum_j u_j \varphi_j(x)\Big)\,\mathrm{dx} = \int \psi_i f\,\mathrm{dx} \quad \forall \psi_i$$

  then all $v \in V$ satisfy the equation
- leads to system of equations for unknowns u_j
  (one equation per test basis function ψ_i)
- V is often chosen to be identical to W_h (Ritz-Galerkin method)

Discretisation – Finite Elements

- L linear $\Rightarrow$ system of linear equations

$$\sum_j \underbrace{\left(\int \psi_i L \varphi_j(x)\,\mathrm{d}x \right)}_{=:A_{ij}} u_j = \int \psi_i f \,\mathrm{d}x \quad \forall \psi_i$$

- aim: make matrix A *sparse* $\rightarrow$ most $A_{ij} = 0$
- approach: local basis functions on a discretisation grid
- ψ_j, φ_j zero everywhere except in grid cells adjacent to grid point x_j
- $A_{ij} = 0$, if ψ_i and φ_j don't overlap

Example Problem: Poisson 1D

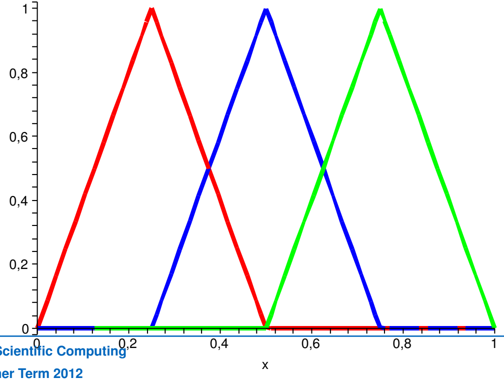
- in 1D: $u''(x) = f(x)$ on $\Omega = (0,1)$,
  hom. Dirichlet boundary cond.: $u(0) = u(1) = 0$
- weak form:

$$\int_0^1 v'(x) \cdot u'(x)\,\mathrm{dx} = \int_0^1 v(x)f(x)\,\mathrm{dx} \quad \forall v$$

- compuational grid:
  $x_i = ih$, (for $i = 1, \ldots, n-1$); mesh size $h = 1/n$
- $V = W$: piecewise linear functions
  (on intervals $[x_i, x_{i+1}]$)

Nodal Basis

$$\varphi_i(x) := \begin{cases} \frac{1}{h}(x - x_{i-1}) & x_{i-1} < x < x_i \\ \frac{1}{h}(x_{i+1} - x) & x_i < x < x_{i+1} \\ 0 & \text{otherwise} \end{cases}$$



Nodal Basis – System of Equations

- stiffness matrix:

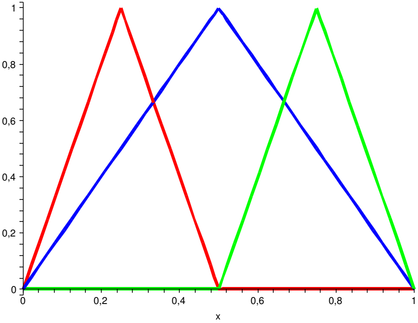
$$\frac{1}{h}\begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix}$$

- right hand sides (assume $f(x) = \alpha \in \mathbb{R}$):

$$\int_0^1 \varphi_i(x) f(x)\,\mathrm{d}x = \int_0^1 \varphi_i(x)\alpha\,\mathrm{d}x = \alpha h$$

- system of equations very similar to finite differences

Hierarchical Basis



- leads to diagonal stiffness matrix!
  (for 1D Poisson)
- solution function identical to that with nodal basis (same function space)

Part III

Finite Element Methods – Towards Implementation

Element Stiffness Matrices

- domain Ω splitted into finite elements $\Omega^{(k)}$:

$$\Omega = \Omega^{(1)} \cup \Omega^{(2)} \cup \cdots \cup \Omega^{(n)}$$

- observation: basis functions are defined element-wise
- use: $\int_a^b f(x)\mathrm{dx} = \int_a^c f(x)\mathrm{dx} + \int_c^b f(x)\mathrm{dx}$
- element-wise evaluation of the integrals:

$$\begin{aligned} \int_\Omega \nabla v \cdot \nabla u \,\mathrm{dx} &= \sum_k \int_{\Omega^{(k)}} \nabla v \cdot \nabla u \,\mathrm{dx} \\ \int_\Omega vf \,\mathrm{dx} &= \sum_i \int_{\Omega^{(i)}} vf \,\mathrm{dx} \end{aligned}$$

Element Stiffness Matrices (2)

- leads to local stiffness matrices for each element:

$$\underbrace{\int_{\Omega^{(k)}} \nabla\phi_i \cdot \nabla\phi_j \, \mathrm{d}x}_{=:A_{ij}^{(k)}}$$

- and respective element systems:

$$A^{(k)} x = b^{(k)}$$

- accumulate to obtain global system:

$$\underbrace{\sum_k A^{(k)}}_{=:A} \, x = \sum_k b^{(k)}$$

Element Stiffness Matrices (3)

Some comments on notation:

- assume: 1D problem, n elements (i.e. intervals)
- in each element only two basis functions are non-zero!
- hence, almost all $A_{ij}^{(k)}$ are zero:

$$A_{ij}^{(k)} = \int_{\Omega^{(k)}} \nabla \phi_i \cdot \nabla \phi_j \, \mathrm{dx}$$

- only 2×2 elements of $A^{(k)}$ are non-zero
- therefore convention to omit zero columns/rows
  $\Rightarrow$ leaves only unknowns that are in $\Omega^{(k)}$

Example: 1D Poisson

- $\Omega = [0, 1]$ splitted into $\Omega^{(k)} = [x_{k-1}, x_k]$
- nodal basis; leads to element stiffness matrix:

$$A^{(k)} = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

- consider only two elements:

$$A^{(1)} + A^{(2)} = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{pmatrix}$$

- in stencil notation:

$$[-1 \quad 1^*] + [1^* \quad -1] \rightarrow [-1 \quad 2 \quad -1]$$

(Note: on this slide, factor $\frac{1}{h}$ is left away for better readability!)

Project: Adaptive Hierarchical Basis

Consider:

- 1D Poisson problem
- FEM with hierarchical basis
- however: not all basis functions used on each grid
  $\rightarrow$ adaptive hierarchical basis

Discuss:

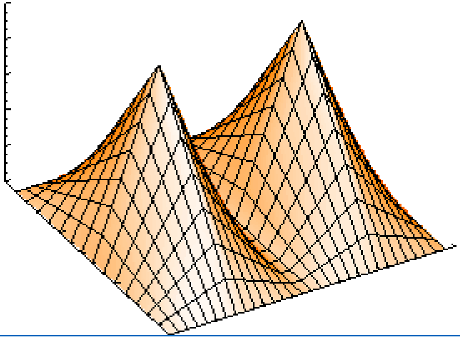
- how to compute the stiffness matrix?
- what do you need to compute, if you add a hierarchical basis function?
- how do you know when to add a basis function?

Example: 2D Poisson

- $-\Delta u = f$ on domain $\Omega = [0, 1]^2$
- splitted into $\Omega^{(i,j)} = [x_{i-1}, x_i] \times [x_{j-1}, x_j]$
- bilinear basis functions

$$\varphi_{ij}(x, y) = \varphi_i(x)\varphi_j(y)$$

- “pagoda” functions



Example: 2D Poisson (2)

- leads to element stiffness matrix:

$$A^{(k)} = \begin{pmatrix} 2 & -\frac{1}{2} & -\frac{1}{2} & -1 \\ -\frac{1}{2} & 2 & -1 & -\frac{1}{2} \\ -\frac{1}{2} & -1 & 2 & -\frac{1}{2} \\ -1 & -\frac{1}{2} & -\frac{1}{2} & 2 \end{pmatrix}$$

- accumulation leads to 9-point stencil

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Typical workflow

1. choose elements:
   - quadratic or cubic cells
   - triangles (structured, unstructured)
   - tetrahedra, etc.
2. set up basis functions for each element $\Omega^{(k)}$;
   for example, at all nodes $x_i \in \Omega^{(k)}$

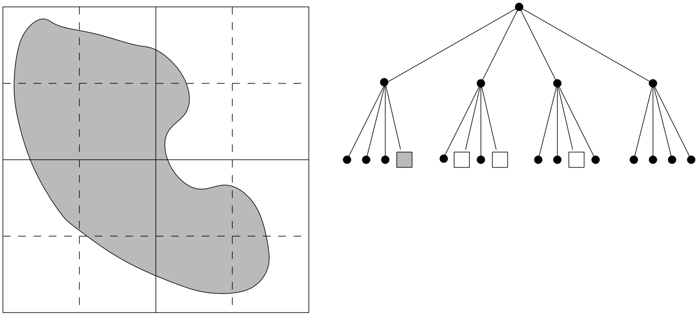
$$\begin{aligned} \varphi_i(x_i) &= 1 \\ \varphi_i(x_j) &= 0 \quad \text{for all } j \neq i \end{aligned}$$

3. for element stiffness matrix, compute all

$$A_{ij}^{(k)} = \int_{\Omega^{(k)}} \varphi_i L \varphi_j \, \mathrm{d}\Omega$$

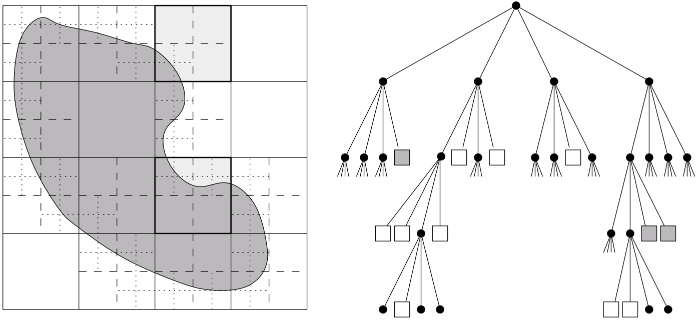
4. accumulate global stiffness matrix

Quadtrees to Represent Objects



- start with an initial square (covering the entire domain)
- recursive substructuring into four subsquares
- adaptive refinement?

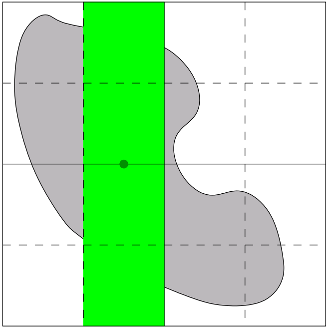
Quadtrees to Represent Objects (2)



- refine, unless squares entirely within or outside domain
- also: refine, if solution not exact enough!
- question: can we build a hierarchical basis on such a quadtree?

Hierarchical Basis vs. Quadtree

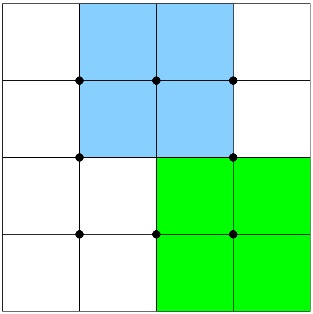
Use hierarchical basis as in 2D sparse grids?

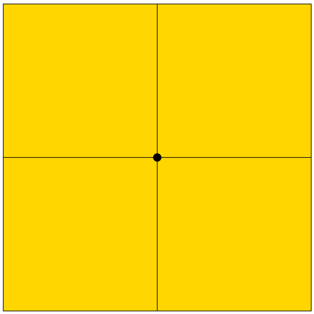


$\Rightarrow$ tensor basis functions do not match quadtree cells

Hierarchical Basis for Quadtrees

Use hierarchical multilevel basis:





hierarchical concept (again): skip basis functions that exist on previous level!

Project: 2D Adaptive Hierarchical Basis

Consider:

- 2D Poisson problem
- FEM with quadtree-compatible hierarchical basis
- adaptive quadtree-based hierarchical basis

Discuss (again):

- how to compute the stiffness matrix?
- what do you need to compute, if you add a hierarchical basis function?
- how do you know when to add a basis function?

Time-Dependent Problems

Example: 1D Heat Equation

- $u_t = u_{xx} + f$ on domain $\Omega = [0, 1]$ for $t \in [0, t_{\text{end}}]$
- spatial discretisation: weak form

$$\begin{aligned} \int v u_t \, \mathrm{dx} &= \int v u_{xx} \, \mathrm{dx} + \int v f \, \mathrm{dx} \\ \frac{\partial}{\partial t} \left(\int v u \, \mathrm{dx} \right) &= \int v u_{xx} \, \mathrm{dx} + \int v f \, \mathrm{dx} \end{aligned}$$

- spatial discretisation – finite elements:

$$\frac{\partial}{\partial t} (M_h u_h) = A_h u_h + f_h$$

M_h: mass matrix, A_h: stiffness matrix, $u_h = u_h(t)$

Time-Dependent Problems (2)

Solve a system of ordinary differential equations:

- after spatial discretisation (M_h constant):

$$\frac{\partial}{\partial t} M_h\left(u_h\right) = A_h u_h + f_h$$

- u_h a vector of time-dependent functions:

$$u_h = \left(u_{11}(t), \ldots, u_{ij}(t), \ldots, u_{nn}(t)\right)^T$$

- usually: approximate M_h by a simpler matrix (diagonal matrix, e.g.) $\rightarrow$ "mass lumping"