Algorithms of Scientific Computing

Finite Element Methods

Michael Bader
Summer Term 2015
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 $k$, ...
- observation from physical experiments: $q \approx k \cdot \delta T$  
  (heat flow proportional to temperature differences)
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:
  \[
  q_{ij}^{(\text{left})} = k_x (T_{ij} - T_{i-1,j}) \, h_y
  \]
  \( k_x \) depends on material
- heat flow across all edges determines change of heat energy:
  \[
  q_{ij} = k_x (T_{ij} - T_{i-1,j}) \, h_y + k_x (T_{ij} - T_{i+1,j}) \, h_y
  + k_y (T_{ij} - T_{i,j-1}) \, h_x + k_y (T_{ij} - T_{i,j+1}) \, h_x
  \]
- equilibrium with source term \( F_{ij} = f_{ij} h_x h_y \) (\( f_{ij} \) heat flow per area) requires \( q_{ij} + F_{ij} = 0 \):
  \[
  f_{ij} h_x h_y = -k_x h_y (2T_{ij} - T_{i-1,j} - T_{i+1,j})
  - k_y h_x (2T_{ij} - T_{i,j-1} - T_{i,j+1})
  \]
Discrete and Continuous Model

• system of equations derived from the discrete model:

\[ f_{ij} = -\frac{k_x}{h_x} \left( 2T_{ij} - T_{i-1,j} - T_{i+1,j} \right) \]
\[ -\frac{k_y}{h_y} \left( 2T_{ij} - T_{i,j-1} - T_{i,j+1} \right) \]

• result: average temperature in each cell

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

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

• *wanted:* approximate \( T(x, y) \) as a function!

→ solution possible using “coefficients and basis functions”?
Part II

Outlook: 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:
  \[
  u''(x) = f(x) \quad \text{for } x \in (0, 1)
  \]

  with \( u(0) = u(1) = 0 \).
Finite Elements – Main Idea

• we consider the residual of the (1D) PDE:

\[ u''(x) = f(x) \quad \Rightarrow \quad u''(x) - f(x) = 0 \]

• represent the functions \( u \) and \( f \) in our “favorite” form:

\[
\left( \sum u_j \phi_j(x) \right)'' - \sum f_j \phi_j(x) = 0
\]

• however: we will usually not find \( u_j \) that solve this equation exactly (as solution \( u \) can not be represented as \( \sum u_j \phi_j(x) \))

• remedy?

\( \Rightarrow \) find “best approximation”, given by orthogonality:

\[
\left\langle w(x), \left( \sum u_j \phi_j(x) \right)'' - \sum f_j \phi_j(x) \right\rangle = 0 \quad \text{“for all } w(x) \text{”}
\]

• remember that \( < g, f > = \int g \cdot f \ dx \)
Finite Elements – Main Ingredients

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

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

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

$$u'' = f \quad \rightarrow \quad - \int v' u' \, dx = \int vf \, dx$$

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

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

(such as the hat functions)
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 ("shape", "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*:

\[
\langle v, Lu \rangle = \int vLu \, dx = \int vf \, dx = \langle f, v \rangle \quad \forall v \in V
\]

\( V \) a certain class of functions
- “real solution” \( u \) also solves the weak form
  (but additional, approximate 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
  - approximate solution \( \hat{u} \) might not solve PDE: \( \hat{L} \hat{u} \neq f \)
    thus: additional functions need to be “acceptable” as solution
    \( \rightarrow \) “orthogonality” idea
Weak Form of the Poisson Equation – 1D

- Poisson equation with Dirichlet conditions:
  \[-u''(x) = f(x) \quad \text{in } \Omega = (0, 1), \quad u(0) = u(1) = 0\]

- weak form:
  \[-\int_{\Omega} v(x) u''(x) \, dx = \int_{\Omega} v(x) f(x) \, dx \quad \forall v\]

- integration by parts:
  \[-\int_{\Omega} v(x) u''(x) \, dx = -v(x) \cdot u'(x) \bigg|_0^1 + \int_{\Omega} v'(x) \cdot u'(x) \, dx\]

- choose functions \( v \) such that \( v(0) = v(1) = 0 \):
  \[\int_{\Omega} v'(x) \cdot u'(x) \, dx = \int_{\Omega} v(x) f(x) \, dx \quad \forall v\]
Weak Form of the Poisson Equation – 2D/3D

- Poisson equation with Dirichlet conditions:
  \[-\Delta u = f \quad \text{in } \Omega, \quad u = 0 \quad \text{on } \delta \Omega\]

- weak form:
  \[-\int_{\Omega} v \Delta u \, d\Omega = \int_{\Omega} vf \, d\Omega \quad \forall v\]

- 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 $\partial \Omega$:
  \[\int_{\Omega} \nabla v \cdot \nabla u \, d\Omega = \int_{\Omega} vf \, d\Omega \quad \forall v\]
Weak Form of the Poisson Equation – Summary

- Poisson equation with Dirichlet conditions:
  
  \[-\Delta u = f \quad \text{in } \Omega, \quad u = 0 \quad \text{on } \delta \Omega\]

- transformed into weak form:
  
  \[\int_{\Omega} \nabla v \cdot \nabla u \, d\Omega = \int_{\Omega} vf \, d\Omega \quad \forall v\]

- 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
  
  $$ \int v L \left( \sum_j u_j \varphi_j(x) \right) \, dx = \int v f \, dx \quad \forall v \in V $$
Choose Test and Ansatz Space (2)

- choose a basis \( \{ \psi_i \} \) of the test space \( V \)
- then: if all basis functions \( \psi_i \) satisfy

\[
\int \psi_i L \left( \sum_j u_j \phi_j(x) \right) \, dx = \int \psi_i f \, dx \quad \forall \psi_i
\]

then all \( v \in V \) satisfy the equation

- leads to system of equations for unknowns \( u_j \)
  (one equation per test basis function \( \psi_i \))
- \( V \) is often chosen to be identical to \( W_h \) (Ritz-Galerkin method)
Discretisation – Finite Elements

- $L$ linear $\Rightarrow$ system of linear equations
  $$\sum_j \left( \int \psi_i L \varphi_j(x) \, dx \right) u_j = \int \psi_i f \, dx \quad \forall \psi_i$$

- aim: make system of equations easy to solve!
  $\Rightarrow$ thus, make matrix $A$ sparse $\Rightarrow$ most $A_{ij} = 0$

- approach: local basis functions on a discretisation grid
- consider hat functions, e.g.: $\psi_j, \varphi_j$ zero everywhere, except in grid cells adjacent to grid point $x_j$
- then $A_{ij} = 0$, if $\psi_i$ and $\varphi_j$ 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:
  \[
  \int_0^1 v'(x) \cdot u'(x) \, dx = \int_0^1 v(x) f(x) \, dx \quad \forall v
  \]

- computational 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:

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

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

$$\int_0^1 \varphi_i(x)f(x) \, dx = \int_0^1 \varphi_i(x)\alpha \, dx = \alpha h$$

- 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 – Basis Functions for 2D

Hierarchical Basis in 2D
Quadtrees and Hierarchical Bases
Quadtrees to Represent Objects
Hierarchical Basis vs. Quadtree
2D Hierarchical Basis – Tensor Product

- define 2D basis functions via tensor product:
  \[ \phi_{i,j}(x, y) := \phi_i(x) \cdot \phi_j(y) \]

- remember multi-index for 2D hierarchical basis:
  \[ \phi_{\vec{l}, \vec{k}}(x_1, x_2) := \phi_{l_1, l_2, k_1, k_2}(x_1, x_2) := \phi_{l_1, k_1}(x_1) \cdot \phi_{l_2, k_2}(x_2) \]

- illustrate via support of the basis functions:
Illustrate via Location of Hat Functions
Adding Adaptivity: Quadtrees

Quadtrees to Represent Objects:

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

Adaptively Refined Meshes for Finite Elements:
- 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?

⇒ 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!
Illustrate via Location of Hat Functions

\[ l_1 = 1 \quad l_1 = 2 \quad l_1 = 3 \]

\[ l_2 = 1 \quad l_2 = 2 \quad l_2 = 3 \]
Quadtree-Compatible Hierarchical Basis

Basis Functions

Similar to tensor-product basis:

- Level-wise hierarchical increments
  \[ W_{\vec{l}} := \text{span}\{\phi_{\vec{l}, \vec{i}}\}_{\vec{i} \in \hat{I}_{\vec{l}}} \]
- Only use “diagonal” levels:
  \[ \vec{l} := \{l, \ldots, l\} \]
- Omit grid points for which all indices are even:
  \[ \hat{I}_{\vec{l}} := \{\vec{i} : 1 \leq \vec{i} < 2^n, \text{ any } i_j \text{ odd}\} \]
Part IV

Finite Element Methods – Towards Implementation

FEM and Hierarchical Basis Transform

Hierarchical Basis Transformation
FEM and Hierarchical Basis Transform
Element Stiffness Matrices
Workflow
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?

Idea: move from node-oriented to element-oriented approach
Hierarchical Basis Transformation

- represent hat functions $\phi_{n-1,i}(x)$ via fine-level functions $\phi_{n,j}(x)$

$$
\phi_{n-1,i}(x) = \frac{1}{2}\phi_{n,2i-1}(x) + \phi_{n,2i}(x) + \frac{1}{2}\phi_{n,2i+1}(x)
$$

- hierarchical-basis transformation as matrix-vector product:

$$
\begin{pmatrix}
\psi_{n,i-1}(x) \\
\psi_{n,i}(x) \\
\psi_{n,i+1}(x)
\end{pmatrix}
:=
\begin{pmatrix}
\phi_{n,2i-1}(x) \\
\phi_{n-1,i}(x) \\
\phi_{n,2i+1}(x)
\end{pmatrix}
= \begin{pmatrix}
1 & 0 & 0 \\
\frac{1}{2} & 1 & \frac{1}{2} \\
0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
\phi_{n,2i-1}(x) \\
\phi_{n,2i}(x) \\
\phi_{n,2i+1}(x)
\end{pmatrix}
$$
Hierarchical Basis Transformation (2)

- hierarchical basis transformation: \( \psi_{n,i}(x) = \sum_j H_{i,j} \phi_{n,j}(x) \)
- written as matrix-vector product: \( \vec{\psi}_n = H_n \vec{\phi}_n \)
- \( H \) can be written as a sequence of level-wise transforms:
  \[
  H_n = H_n^{(1)} \ldots H_n^{(n-2)} H_n^{(n-1)}
  \]
- where each transform has a shape similar to
  \[
  H_3^{(2)} = \begin{pmatrix}
  1 & 0 & 0 & 0 & 0 & 0 & 0
  \\
  1/2 & 1 & 1/2 & 0 & 0 & 0 & 0
  \\
  0 & 0 & 1 & 0 & 0 & 0 & 0
  \\
  0 & 0 & 1/2 & 1 & 1/2 & 0 & 0
  \\
  0 & 0 & 0 & 0 & 1 & 0 & 0
  \\
  0 & 0 & 0 & 0 & 1/2 & 1 & 1/2
  \\
  0 & 0 & 0 & 0 & 0 & 0 & 1
  \end{pmatrix}
  \]
Hierarchical Coordinate Transformation

• consider function \( f(x) \approx \sum_i a_i \psi_{n,i}(x) \) represented via hier. basis

• wanted: corresponding representation in nodal basis

\[
\sum_k b_k \phi_{n,k}(x) = \sum_i a_i \psi_{n,i}(x) \approx f(x)
\]

• with \( \psi_{n,i}(x) = \sum_j H_{i,j} \phi_{n,j}(x) \) we obtain

\[
\sum_k b_k \phi_{n,k}(x) = \sum_i a_i \sum_j H_{i,j} \phi_{n,j}(x) = \sum_j \sum_i a_i H_{i,j} \phi_{n,j}(x)
\]

• compare coordinates (identify indices \( j \) and \( k \)) and get

\[
b_j = \sum_i H_{i,j} a_i = \sum_i (H^T)_{j,i} a_i
\]

• written in vector notation: \( b = H^T a \)
FEM and Hierarchical Basis Transform

- FEM discretisation with hierarchical test and shape functions:
  \[
  \int \psi_i(x)L\left(\sum_j v_j \psi_j(x)\right) \, dx = \int \psi_i(x)f(x) \, dx \quad \forall \psi_i
  \]

- stiffness matrix with hierarchical basis as shape functions:
  \[
  \int \psi_i(x)L\left(\sum_j u_j \psi_j(x)\right) \, dx = \sum_j u_j \int \psi_i(x)L\psi_j(x) \, dx = \sum_j u_j A_{i,j}^{\text{HB}}
  \]

- vs. stiffness matrix with nodal basis as shape functions:
  \[
  \int \psi_i(x)L\left(\sum_j v_j \phi_j(x)\right) \, dx = \sum_j v_j \int \psi_i(x)L\phi_j(x) \, dx = \sum_j v_j A_{i,j}^{\text{*}}
  \]

- note that \((A^{\text{HB}}u)_i = \sum_j u_j A_{i,j}^{\text{HB}} = \sum_j v_j A_{i,j}^{\text{*}} = (A^* v)_i\) and \(v = H^T u\)
FEM and Hierarchical Basis Transform (2)

- status: FEM with hierarchical test and nodal shape functions

\[ \int \psi_i(x)L \left( \sum_j v_j \phi_j(x) \right) \, dx = \int \psi_i(x)f(x) \, dx \]

- represent test functions via nodal basis:

\[ \int \sum_k H_{i,k} \phi_k(x)L \left( \sum_j v_j \phi_j(x) \right) \, dx = \int \sum_k H_{i,k} \phi_k(x)f(x) \, dx \]

\[ \sum_k H_{i,k} \int \phi_k(x)L \left( \sum_j v_j \phi_j(x) \right) \, dx = \sum_k H_{i,k} \int \phi_k(x)f(x) \, dx \]

- leads to new system of equations: \( HA^{NB} \nu = H b^{NB} \)

where \( A^{NB} \) and \( b^{NB} \) stem from nodal-basis FEM discretisation!

- with \( \nu = H^T u \) we obtain \( HA^{NB} H^T u = H b \) as system of equations, thus: \( A^{HB} = H A^{NB} H^T \) (\( \rightsquigarrow \) Galerkin coarsening)
Element Stiffness Matrices

- domain $\Omega$ splitted into finite elements $\Omega^{(k)}$:
  \[
  \Omega = \Omega^{(1)} \cup \Omega^{(2)} \cup \ldots \cup \Omega^{(n)}
  \]

- observation: basis functions are defined element-wise

- use: $\int_{a}^{b} f(x) \, dx = \int_{a}^{c} f(x) \, dx + \int_{c}^{b} f(x) \, dx$

- element-wise evaluation of the integrals:
  \[
  \int_{\Omega} \nabla v \cdot \nabla u \, dx = \sum_{k} \int_{\Omega^{(k)}} \nabla v \cdot \nabla u \, dx
  \]
  \[
  \int_{\Omega} vf \, dx = \sum_{i} \int_{\Omega^{(i)}} vf \, dx
  \]
Element Stiffness Matrices (2)

- leads to local stiffness matrices for each element:

\[
\int_{\Omega^{(k)}} \nabla \phi_i \cdot \nabla \phi_j \, dx =: A^{(k)}_{ij}
\]

- and respective element systems:

\[
A^{(k)} x = b^{(k)}
\]

- accumulate to obtain global system:

\[
\sum_k A^{(k)} x = \sum_k b^{(k)} =: A
\]
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:

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

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

   \[
   \varphi_i(x_i) = 1
   \]

   \[
   \varphi_i(x_j) = 0 \quad \text{for all } j \neq i
   \]

3. for element stiffness matrix, compute all

   \[
   A^{(k)}_{ij} = \int_{\Omega^{(k)}} \varphi_i \mathbf{L} \varphi_j \, d\Omega
   \]

4. accumulate global stiffness matrix
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:

$$\begin{bmatrix} -1 & 1^* \\ 1^* & -1 \end{bmatrix} + \begin{bmatrix} 1^* & -1 \end{bmatrix} \rightarrow \begin{bmatrix} -1 & 2 & -1 \end{bmatrix}$$
Project: Adaptive Hierarchical Basis

Consider:

- 1D Poisson problem
- FEM with hierarchical basis
- however: not all basis functions used on each grid
  → 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

\[
\begin{pmatrix}
-1 & -1 & -1 \\
-1 & 8 & -1 \\
-1 & -1 & -1
\end{pmatrix}
\]