DISCRETIZATION OF PARTIAL DIFFERENTIAL EQUATIONS

Goal: to show how partial differential lead to sparse linear systems

- **See Chap. 2 of text**
- **Finite difference methods**
- **Finite elements**
- **Assembled and unassembled finite element matrices**

Why study discretized PDEs?

➤ Still the most important source of sparse linear systems

➤ Will help understand the structures of the problem and their connections with "meshes" in 2-D or 3-D space

➤ Also: iterative methods are often formulated for the PDE directly – instead of a discretized (sparse) system.

A typical numerical simulation

Physical Problem

↓

Nonlinear PDEs

↓

Discretization

↓

Linearization (Newton)

↓

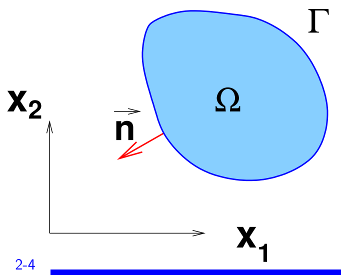
Sequence of Sparse Linear Systems $Ax = b$

Example: discretized Poisson equation

➤ Common Partial Differential Equation (PDE) :

$$\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} = f, \text{ for } x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \text{ in } \Omega$$

where Ω = bounded, open domain in $\mathbb{R}^2$



➤ + boundary conditions:

Dirichlet: $u(x) = \phi(x)$

Neumann: $\frac{\partial u}{\partial \vec{n}}(x) = 0$

Cauchy: $\frac{\partial u}{\partial \vec{n}} + \alpha(x)u = \gamma$

➤ $\Delta = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2}$ is the Laplace operator or Laplacean

➤ How to approximate the problem?

➤ Answer: discretize, i.e., replace continuum with discrete set.

➤ Then approximate Laplacean usinge this discretization

➤ Many types of discretizations.. wll briefly cover Finite differences and finite elements.

Finite Differences: Basic approximations

➤ Formulas derived from Taylor series expansion:

$$u(x+h) = u(x) + h\frac{du}{dx} + \frac{h^2}{2}\frac{d^2u}{dx^2} + \frac{h^3}{6}\frac{d^3u}{dx^3} + \frac{h^4}{24}\frac{d^4u}{dx^4}(\xi)$$

Discretization of PDEs - Basic approximations

➤ Simplest scheme: forward difference

$$\frac{du}{dx} = \frac{u(x+h)-u(x)}{h} - \frac{h}{2}\frac{d^2u(x)}{dx^2} + O(h^2)$$
$$\approx \frac{u(x+h)-u(x)}{h}$$

➤ Centered differences for second derivative:

$$\frac{d^2u(x)}{dx^2} = \frac{u(x+h)-2u(x)+u(x-h)}{h^2} - \frac{h^2}{12}\frac{d^4u(\xi)}{dx^4},$$

where $\xi_- \le \xi \le \xi_+$.

Notation:

$$\delta^+ u(x) = u(x+h) - u(x)$$
$$\delta^- u(x) = u(x) - u(x-h)$$

➤ Operations of the type: $\frac{d}{dx}\left[a(x)\frac{d}{dx}\right]$ are very common [in-homogeneous media].

➤ The following is a second order approximation:

$$\frac{d}{dx}\left[a(x)\frac{du}{dx}\right] = \frac{1}{h^2}\delta^+\left(a_{i-\frac{1}{2}}\,\delta^- u\right) + O(h^2)$$
$$\approx \frac{a_{i+\frac{1}{2}}(u_{i+1}-u_i) - a_{i-\frac{1}{2}}(u_i-u_{i-1})}{h^2}$$

✍ Show that $\delta^+\left(a_{i-\frac{1}{2}}\,\delta^- u\right) = \delta^-\left(a_{i+\frac{1}{2}}\,\delta^+ u\right)$

Finite Differences for 2-D Problems

Consider the simple problem,

$$-\left(\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2}\right) = f \quad \text{in } \Omega \tag{1}$$
$$u = 0 \quad \text{on } \Gamma \tag{2}$$

$\Omega =$ rectangle $(0, l_1) \times (0, l_2)$ and Γ its boundary.

Discretize uniformly :

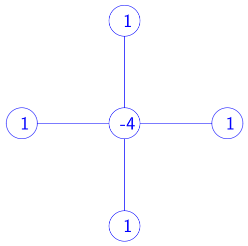
$$x_{1,i} = i \times h_1 \quad i = 0, \ldots, n_1+1 \quad h_1 = \frac{l_1}{n_1+1}$$
$$x_{2,j} = j \times h_2 \quad j = 0, \ldots, n_2+1 \quad h_2 = \frac{l_2}{n_2+1}$$

Finite Difference Scheme for the Laplacean

➤ Using centered differences for both the $\frac{\partial^2}{\partial x_1^2}$ and $\frac{\partial^2}{\partial x_2^2}$ terms - with mesh sizes $h_1 = h_2 = h$:

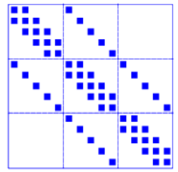
$$\Delta u(x) \approx \frac{1}{h^2}[u(x_1+h,x_2) + u(x_1-h,x_2) + \\ +u(x_1,x_2+h) + u(x_1,x_2-h) - 4u(x_1,x_2)]$$

The 5-point 'stencil:'



The resulting matrix has the following block structure:

$$A = \frac{1}{h^2}\begin{bmatrix} B & -I & \\ -I & B & -I \\ & -I & B \end{bmatrix}$$



Matrix for 7×5 finite difference mesh

With

$$B = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & -1 & 4 & -1 & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{bmatrix}.$$

Finite Elements: a quick overview

Background: Green's formula

$$\int_\Omega \nabla v . \nabla u \;\; dx = -\int_\Omega v \Delta u \;\; dx + \int_\Gamma v \frac{\partial u}{\partial \vec{n}} \, ds.$$

➤ ∇ = gradient operator. In 2-D:

$$\nabla u = \begin{pmatrix} \frac{\partial u}{\partial x_1} \\ \frac{\partial u}{\partial x_2} \end{pmatrix},$$

➤ The dot indicates a dot product of two vectors.

➤ Δu = Laplacean of u

➤ $\vec{n}$ is the unit vector that is normal to Γ and directed outwards.

➤ Frechet derivative:

$$\frac{\partial u}{\partial \vec{v}}(x) = \lim_{h \to 0} \frac{u(x + h\vec{v}) - u(x)}{h}$$

➤ Green's formula generalizes the usual formula for integration by parts

➤ Define

$$a(u,v) \equiv \int_\Omega \nabla u . \nabla v \, dx = \int_\Omega \left(\frac{\partial u}{\partial x_1}\frac{\partial v}{\partial x_1} + \frac{\partial u}{\partial x_2}\frac{\partial v}{\partial x_2} \right) dx$$

$$(f,v) \equiv \int_\Omega f v \, dx.$$

Denote:

$$(u,v) = \int_\Omega u(x)v(x)dx,$$

➤ With Dirichlet BC, the integral on the boundary in Green's formula vanishes $\to$

$$a(u, v) = -(\Delta u, v).$$

➤ Weak formulation of the original problem: select a subspace of reference V of L^2 and then solve

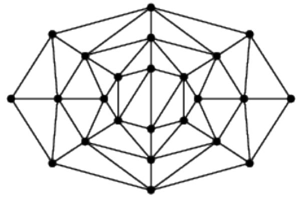
$$\text{Find} \quad u \ \in V \quad \text{such that} \quad a(u, v) = (f, v), \quad \forall\, v \in V$$

➤ Finite Element method solves this weak problem...

➤ ... by discretization

➤ The original domain is approximated by the union Ω_h of m triangles K_i,

Triangulation of Ω :



$$\Omega_h = \bigcup_{i=1}^{m} K_i.$$

➤ Some restrictions on angles, edges, etc..

$$V_h = \{\phi \mid \phi_{|\Omega_h} \ \in \ \mathcal{C}^0, \quad \phi_{|\Gamma_h} = 0, \ \phi_{|K_j} \text{ linear } \forall\, j\}$$

➤ $\mathcal{C}^0$ = set of *continuous* functions

➤ $\phi_{|X}$ == restriction of ϕ to the subset X

➤ Let $x_j, j = 1, \ldots, n$, be the nodes of the triangulation

➤ Can define a (unique) 'hat' function ϕ_j in V_h associated with each x_j s.t.:

$$\phi_j(x_i) = \delta_{ij} = \begin{cases} 1 \text{ if } x_i = x_j \\ 0 \text{ if } x_i \neq x_j \end{cases}.$$

➤ Each function u of V_h can be expressed as (!)

$$u(x) = \sum_{j=1}^{n} \xi_j \phi_j(x). \qquad (*)$$

➤ The finite element approximation consists of writing the Galerkin condition for functions in V_h:

$$\text{Find } u \ \in V_h \text{ such that } a(u, v) = (f, v), \ \ \forall \ \ v \in V_h$$

➤ Express u in the basis $\{\phi_j\}$ (see *), then substitute above

➤ Result: the linear system (!)

$$\sum_{j=1}^{n} \alpha_{ij} \xi_j = \beta_i$$

where (!)

$$\alpha_{ij} = a(\phi_j, \phi_i), \quad \beta_i = (f, \phi_i).$$

The above equations form a linear system of equations

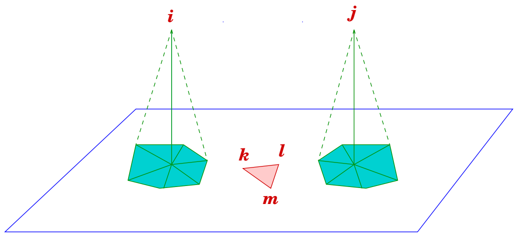
$$Ax = b$$

➤ A is *Symmetric Positive Definite*

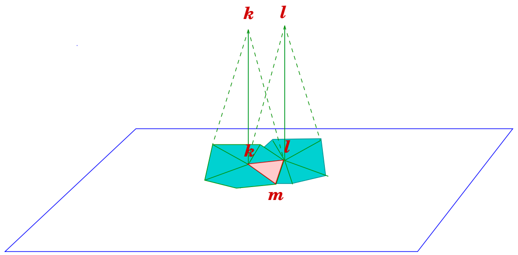
Prove it

The Assembly Process: Illustration

If triangle $K \notin$ support domains of both ϕ_i and ϕ_j then $a_K(\phi_i, \phi_j) = 0$

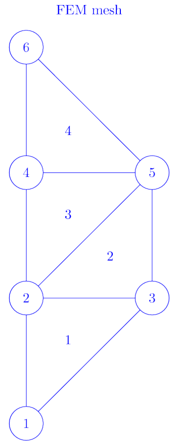


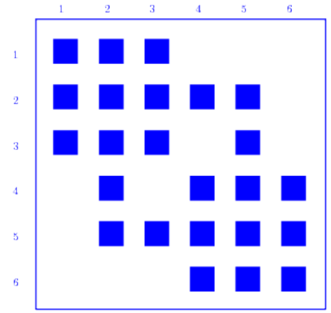
If triangle $K \in$ *both* nonzero domains of ϕ_i and ϕ_j then $a_K(\phi_i, \phi_j) \neq 0$



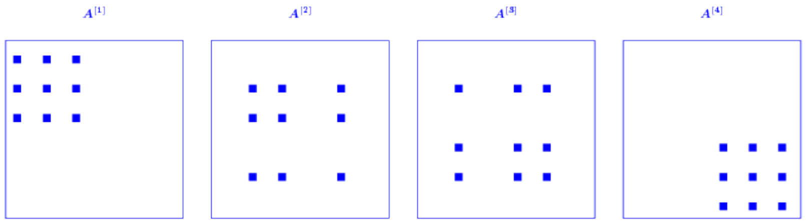
➤ So: $a_K(\phi_i, \phi_j) \neq 0$ iff $i \in \{k, l, m\}$ and $j \in \{k, l, m\}$.

The Assembly Process





A simple finite element mesh and the pattern of the corresponding assembled matrix.



Element matrices $A^{[e]}$, $e = 1, \ldots, 4$ for FEM mesh shown above

➤ Each element contributes a 3×3 submatrix $A^{[e]}$ (spread out)

➤ Can use the matrix in un-assembled form - To multiply a vector by A for example we can do

$$y = Ax = \sum_{e=1}^{nel} A^{[e]}x \; = \; \sum_{e=1}^{nel} P_e A_{K_e}(P_e^T x).$$

➤ Can be computed using the element matrices A_{K_e} - no need to assemble

➤ The product $P_e^T x$ gathers x data associated with the e-element into a 3-vector consistent with the ordering of the matrix A_{K_e}.

➤ Advantage: some simplification in process

➤ Disadvantage: cost (memory + computations).

Resources: A few matlab scripts

➤ These (and others) will be posted in the matlab folder of class web-site

```
>> help fd3d
  function A = fd3d(nx,ny,nz,alpx,alpy,alpz,dshift)
  NOTE nx and ny must be > 1 -- nz can be == 1.
  5- or 7-point block-Diffusion/conv. matrix. with
```

➤ A stripped-down version is `lap2D(nx,ny)`

```
>> help mark
 [A] = mark(m)
 generates a Markov chain matrix for a random walk
 on a triangular grid. A is sparse of size n=m*(m+1)/2
```


✍ Explore A few useful matlab functions

* `kron`

* `gplot` for ploting graphs

* `reshape` for going from say 1-D to 2-D or 3-D arrays

✍ Write a script to generate a 9-point discretization of the Laplacean.

The Matlab PDE toolbox

➤ The PDE toolbox provides functions for setting up and solving a PDE of the form

$$m\frac{\partial^2 u}{\partial t^2} + d\frac{\partial u}{\partial t} - \Delta(c\nabla u) + au = f$$

- model=`createmodel()`. Initiates the class 'model'
- `geometryFromEdges(model,...)` Creates the geometry.
- `pdegplot(model,...)` plots the geometry
- `applyBoundaryCondition(model,...)` Applies boundary conditions
- `specifyCoefficients(model,...)` Sets coeff.s m, d, c, a, f above

- `generateMesh(model,...)` Generates the mesh
- `results = pdesolve(model,...)` solves the PDE
- `pdeplot(model,...)` plots solution

➤ Also `assembleFEMatrices(model,...)` assembles the FEM problem, [returns K and M in a structure]

✍ Follow the example in the documentation and get an understanding of the functions that are called.