Finite Difference Discretization of Elliptic Equations: FD Formulas and Multidimensional Problems

Lecture 4

1 Finite Difference Formulas

1.1 Problem Definition

SLIDE 1

We have seen that one of the necessary ingredients in devising finite difference approximations is the ability to accurately approximate derivatives in terms of differences. Here we will consider two approaches to developing such approximations.

Given $l + r + 1$ distinct points $(x_{-l}, x_{-l+1}, \ldots, x_0, \ldots, x_r)$, find the weights δ_j^m such that

$$\left.\frac{d^m v}{dx^m}\right|_{x=x_0} \approx \sum_{j=-l}^{r} \delta_j^m v_j$$

is of optimal order of accuracy. $\boxed{\text{N1}}$

Two approaches:

- **Lagrange interpolation**
- **Undetermined coefficients**

Note 1 ***Accuracy of a Finite Difference Approximation***

If,

$$\left(\left.\frac{d^m v}{dx^m}\right|_{x=x_i} - \sum_{j=-l}^{r} \delta_j^m v_j\right) = \mathcal{O}(\Delta x^p),$$

for all sufficiently smooth functions v, we say that the difference scheme is p-th order accurate.

1.2 Lagrange interpolation

SLIDE 2

Lagrange polynomials

$$L_j(x) = \frac{(x - x_{-l}) \cdots (x - x_{j-1})(x - x_{j+1}) \cdots (x - x_r)}{(x_j - x_{-l}) \cdots (x_j - x_{j-1})(x_j - x_{j+1}) \cdots (x_j - x_r)}$$

We note that, by construction, $L_j(x)$ takes the value of one at x_j and zero at all the other $r + l$ points.

Lagrange interpolant

$$\hat{v}(x) = \sum_{j=-l}^{r} L_j(x) v_j$$

The Lagrange interpolant is the polynomial of lowest degree that passes through all the points $(x_j, v_j), j = -l, \dots r$.

SLIDE 3

Approximate

$$\left.\frac{d^m v}{dx^m}\right|_{x=x_0} \approx \left.\frac{d^m \hat{v}}{dx^m}\right|_{x=x_0} = \sum_{j=-l}^{r} \left.\frac{d^m L_j}{dx^m}\right|_{x=x_0} v_j$$

Therefore,

$$\boxed{\delta_j^m = \left.\frac{d^m L_j}{dx^m}\right|_{x=x_0}.}$$

1.2.1 Example

SLIDE 4

Set $l = r = 1$, (x_{j-1}, x_j, x_{j+1})

Second order Lagrange interpolant

$$\begin{aligned}\hat{v}(x) \;=\; & \tfrac{(x-x_j)(x-x_{j+1})}{(x_{j-1}-x_j)(x_{j-1}-x_{j+1})}\, v_{j-1} + \tfrac{(x-x_{j-1})(x-x_{j+1})}{(x_j-x_{j-1})(x_j-x_{j+1})}\, v_j + \\ & \tfrac{(x-x_{j-1})(x-x_j)}{(x_{j+1}-x_{j-1})(x_{j+1}-x_j)}\, v_{j+1}\end{aligned}$$

SLIDE 5

Assuming a uniform grid

$m = 1$ (First derivative)

	δ_{j-1}^1	δ_j^1	δ_{j+1}^1	
$i = j - 1$	$-\frac{3}{2\Delta x}$	$\frac{2}{\Delta x}$	$-\frac{1}{2\Delta x}$	Forward
$i = j$	$-\frac{1}{2\Delta x}$	0	$\frac{1}{2\Delta x}$	Centered
$i = j + 1$	$\frac{1}{2\Delta x}$	$-\frac{2}{\Delta x}$	$\frac{3}{2\Delta x}$	Backward

For example a second order backward difference approximation can be written as $v_j' = (3v_j - 4v_{j-1} + v_{j-2})/(2\Delta x)$.

SLIDE 6

$m = 2$ (Second derivative)

δ_{j-1}^2	δ_j^2	δ_{j+1}^2	
$\frac{1}{\Delta x^2}$	$-\frac{2}{\Delta x^2}$	$\frac{1}{\Delta x^2}$	Centered

We note that since we are starting with three points, our interpolating polynomial is second order, and therefore, the second derivative is constant everywhere. In general, to approximate a derivative of order m we will require at least $m+1$ points.

$\boxed{\text{N2}}$

Note 2 — ***Fornberg's algorithm***

The δ_j^m can be computed very efficiently using a recursive algorithm developed by Fornberg (*Generation of Finite Difference Formulas on Arbitrarily Spaced Grids*, Mathematics of Computation, 51,184).

1.3 Undetermined coefficients

Slide 7

Start from

$$\left.\frac{d^m v}{dx^m}\right|_{x=x_i} \approx \sum_{j=-l}^{r} \delta_j^m v_j \ .$$

Insert Taylor expansions for v_j about $x = x_i$

$$v_j = v_0 + v_0'(x_j - x_i) + \frac{1}{2} v_0''(x_j - x_i)^2 + \ldots,$$

determine coefficients δ_j^m to maximize accuracy.

1.3.1 Example

Slide 8

$m = 2$, $l = r = 1$, $i = 0$, (uniform spacing Δx)

Slide 9

$$\begin{array}{rcl} v_0'' & = & \delta_{-1}^2 (v_0 - \Delta x v_0' + \frac{\Delta x^2}{2} v_0'' - \frac{\Delta x^3}{6} v_0''' + \frac{\Delta x^4}{24} v_0^{(4)} + \ldots) \\ & + & \delta_0^2 \quad v_0 \\ & + & \delta_1^2 (v_0 + \Delta x v_0' + \frac{\Delta x^2}{2} v_0'' + \frac{\Delta x^3}{6} v_0''' + \frac{\Delta x^4}{24} v_0^{(4)} + \ldots) \end{array}$$

Equating coefficients of $v_0^{(k)}$

$$\begin{array}{lcl} k = 0 & \Rightarrow & \delta_{-1}^2 + \delta_0^2 + \delta_1^2 = 0 \\ k = 1 & \Rightarrow & \Delta x(\delta_1^2 - \delta_{-1}^2) = 0 \\ k = 2 & \Rightarrow & \frac{\Delta x^2}{2}(\delta_1^2 + \delta_{-1}^2) = 1 \end{array}$$

In general we will start with $k = 0$, and increase k until a sufficient number of independent equations is generated so that all the coefficients are uniquely determined.

Solve,

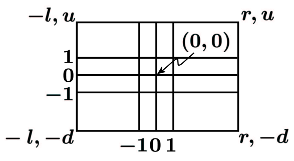
$$\delta^2_{-1} = \frac{1}{\Delta x^2}, \quad \delta^2_0 = -\frac{2}{\Delta x^2}, \quad \delta^2_1 = \frac{1}{\Delta x^2}$$

to recover the same second order central difference approximation to a second derivative previously derived.

$\boxed{\text{N3}}$ $\boxed{\text{N4}}$

Note 3 **Multidimensional finite difference formulas**

The Lagrange interpolation and undetermined coefficients approach we have seen can be easily extended to multiple dimensions.
Lagrange interpolants can be constructed in multiple dimensions by combining one dimensional Lagrange polynomials. Given a lattice of $(l+r+1)\times(d+u+1)$ points



we can construct the following one-dimensional Lagrange polynomials for a typical point j, k

$$L^x_j(x) = \frac{(x - x_{-l})\cdots(x - x_{j-1})(x - x_{j+1})\cdots(x - x_r)}{(x_j - x_{-1})\cdots(x_j - x_{j-1})(x_j - x_{j+1})\cdots(x_j - x_r)}$$

$$L^y_k(y) = \frac{(y - y_{-d})\cdots(y - y_{k-1})(y - y_{k+1})\cdots(y - y_u)}{(y_k - y_{-d})\cdots(y_k - y_{k-1})(y_k - y_{k+1})\cdots(y_k - y_u)}$$

The Lagrange interpolant is thus obtained as

$$\hat{v}(x, y) = \sum_{j=-l}^{r} \sum_{k=-d}^{u} L^x_j(x)\, L^y_k(y)\, v_{jk} \ .$$

We note that by construction $L^x_j(x)\, L^y_k(y)$ takes a value of one at (x_j, y_k) and zero at all other points in the lattice.

A general partial derivative can be approximated as

$$\frac{\partial^{m+n} v}{\partial x^m\, \partial y^n} \approx \frac{\partial^{m+n} \hat{v}}{\partial x^m\, \partial x^n} = \sum_{j=-l}^{r} \sum_{k=-d}^{u} \frac{\partial^{m+n}}{\partial x^m\, \partial y^n} L_j(x)\, L_k(y)\, v_{jk} \ .$$

Therefore, defining the weights

$$\delta_{jk}^{mn} = \frac{\partial^{m+n}}{\partial x^m\, \partial y^n} L_j(x)\, L_k(y) \bigg|_{(x_0,y_0)}$$

we have

$$\frac{\partial^{m+n} \hat{v}}{\partial x^m\, \partial y^n} \bigg|_{(x_0,y_0)} \simeq \sum_{j=-l}^{r} \sum_{k=-d}^{u} \delta_{jk}^{mn}\, v_{jk}$$

The method of undetermined coefficient extends to multiple dimensions in a straightforward manner if we consider multidimensional Taylor series expansions. Thus, assuming a uniform Δx and Δy, we have

$$\begin{aligned} v_{jk} \quad = \quad & v_{(x_0,y_0)} + j\,\Delta x \frac{\partial v}{\partial x}\bigg|_{(x_0,y_0)} + k\,\Delta y \frac{\partial v}{\partial y}\bigg|_{(x_0,y_0)} + \frac{(j\Delta x)^2}{2} \frac{\partial^2 v}{\partial x^2}\bigg|_{(x_0,y_0)} \\ & + (j\Delta x\, k\Delta y) \frac{\partial^2 v}{\partial x \partial y}\bigg|_{(x_0,y_0)} + \frac{(k\Delta y)^2}{2} \frac{\partial^2 v}{\partial y^2}\bigg|_{(x_0,y_0)} + \cdots \ . \end{aligned}$$

A finite difference approximation of the form

$$\frac{\partial^{m+n} v}{\partial x^m\, \partial y^n} \bigg|_{(x_0,y_0)} \approx \sum_{j=-l}^{r} \sum_{k=-d}^{u} \delta_{jk}^{mn}\, v_{jk}$$

can be obtained by inserting the Taylor series expansions for v_{jk} and determining the δ_{jk}^{mn} coefficients by equating the coefficients of the different derivative terms.

Note 4 — ***Compact difference approximations***

The finite difference approximations considered here are called divided differences approximations. More sophisticated, and accurate, difference approximations are also possible. These approximations are called compact approximations and take the form

$$\sum_{j=-l_1}^{r_1} \frac{d^m v}{dx^m}\bigg|_{x=x_j} \approx \sum_{j=-l}^{r} \delta_j^m v_j \ ,$$

and a particular well known compact approximation to the first derivative is

$$\frac{1}{6}(v'_{j+1} + 4v'_j + v'_{j-1}) = \frac{v_{j+1} - v_{j-1}}{2\Delta x}.$$

It is easy to verify, using Taylor expansions or otherwise, that this approximation if fourth order accurate for a sufficiently smooth function, i.e. $p = 4$.

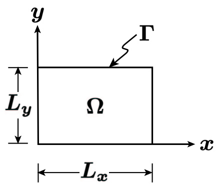
2 Poisson Equation in 2D

2.1 Definition

SLIDE 10

$$
\begin{aligned}
-\nabla^2 u(x,y) &= f(x,y) \quad &\text{in} \quad \Omega \\
u &= 0 \quad &\text{on} \quad \Gamma
\end{aligned}
$$

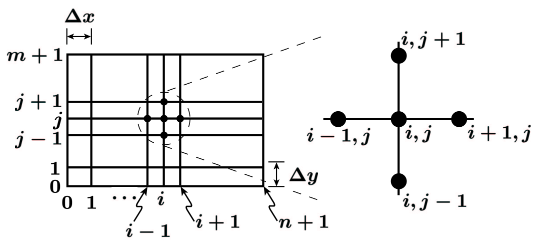
$$\nabla^2 \equiv \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}, \qquad f \in \mathcal{C}^0$$



We have seen that one of the critical requirements to obtain optimal a-priori error estimates is that the solution be sufficiently regular so that the derivatives appearing in the leading terms of the truncation error exist. In 1D the regularity of the solution depends exclusively on the regularity of f (see Fourier analysis in Lecture 1). In multidimensions however, the regularity of the boundary plays also a very important role and, for general domains involving corners or non-regular boundary data, the solution may not be very regular. This topic will be dealt with in greater detail in the finite element lectures.

2.2 Discretization

SLIDE 11



$$\Delta x = \tfrac{L_x}{n+1}, \quad \Delta y = \tfrac{L_y}{m+1}, \quad x_i = i\Delta x, \quad y_j = j\Delta y$$

2.3 Approximation

SLIDE 12

For example ...

$$\left.\frac{\partial^2 v}{\partial x^2}\right|_{i,j} \approx \frac{v_{i+1,j} - 2v_{i,j} + v_{i-1,j}}{\Delta x^2}$$

$$\left.\frac{\partial^2 v}{\partial y^2}\right|_{i,j} \approx \frac{v_{i,j+1} - 2v_{i,j} + v_{i,j-1}}{\Delta y^2}$$

for Δx, Δy small

2.4 Equations

SLIDE 13

$-u_{xx} - u_{yy} = f$ suggests ...

$$-\frac{\hat{u}_{i+1,j} - 2\hat{u}_{i,j} + \hat{u}_{i-1,j}}{\Delta x^2} - \frac{\hat{u}_{i,j+1} - 2\hat{u}_{i,j} + \hat{u}_{i,j-1}}{\Delta y^2} = \hat{f}_{i,j}$$

$$\begin{array}{lll} \hat{u}_{0,j} = \hat{u}_{n,j} = 0 & 1 & \leq j \leq m \\ \hat{u}_{i,0} = \hat{u}_{i,m} = 0 & 1 & \leq i \leq n \end{array}$$

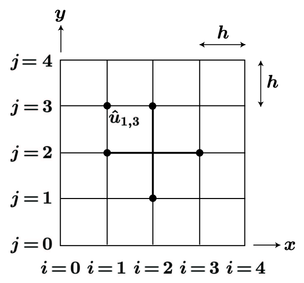
$$\Rightarrow \quad \boxed{A\underline{\hat{u}} = \underline{\hat{f}}}$$

2.4.1 Example

SLIDE 14

$n = m = 3$

$\Delta x = \Delta y = h$



The 9 unknowns are collected into a column vector unknown $\underline{\hat{u}}$, by, somehow arbitrarily, stacking the unknowns row by row. Thus, $\underline{\hat{u}} = \{\hat{u}_{1,1}, \hat{u}_{2,1}, \hat{u}_{3,1}, \hat{u}_{1,2}, \hat{u}_{2,2}, \ldots, \hat{u}_{3,3}\}$.

SLIDE 15

$$A = \frac{1}{h^2}\left[\begin{array}{ccc|ccc|ccc} 4 & -1 & & -1 & & & & & \\ -1 & 4 & -1 & & -1 & & & & \\ & -1 & 4 & & & -1 & & & \\ \hline -1 & & & 4 & -1 & & -1 & & \\ & -1 & & -1 & 4 & -1 & & -1 & \\ & & -1 & & -1 & 4 & & & -1 \\ \hline & & & -1 & & & 4 & -1 & \\ & & & & -1 & & -1 & 4 & -1 \\ & & & & & -1 & & -1 & 4 \end{array}\right]$$

2.4.2 Numbering

SLIDE 16

$$\underline{\hat{u}} = \begin{pmatrix} \hat{u}_{11} \\ \vdots \\ \hat{u}_{n1} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \hat{u}_{nm} \end{pmatrix}, \quad \underline{\hat{f}} = \begin{pmatrix} \hat{f}_{11} \\ \vdots \\ \hat{f}_{n1} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \hat{f}_{nm} \end{pmatrix}$$

(i, j) becomes component $(jm + i)$

2.4.3 Block Matrix

SLIDE 17

$$A = \begin{pmatrix} A_x + 2I_y & -I_y & 0 & \cdots & 0 \\ -I_y & A_x + 2I_y & -I_y & \ddots & \vdots \\ 0 & \ddots & \ddots & \vdots & 0 \\ \vdots & \ddots & -I_y & A_x + 2I_y & -I_y \\ 0 & \cdots & 0 & -I_y & A_x + 2I_y \end{pmatrix}$$

Block $(m \times m)$ tridiagonal matrix

$A_x, I_y : (n \times n)$, $\boxed{A : (nm \times nm)}$

SLIDE 18

Block Definitions

$$A_x = \frac{1}{\Delta x^2} \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \vdots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix}, \quad I_y = \frac{1}{\Delta y^2} \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \ddots & \vdots \\ 0 & \ddots & \ddots & \vdots & 0 \\ \vdots & \ddots & 0 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 \end{pmatrix}$$

A has a banded structure

Bandwidth : $2n + 1$

We note that when $m < n$ the unkowns can be assembled into $\underline{\hat{u}}$ by columns rather than rows in which case A will be $(n \times n)$ block tridiagonal with blocks of size $(m \times m)$, and the bandwidth in this case will be $2m + 1$.

2.4.4 SPD Property

The system of equations $A\underline{\hat{u}} = \underline{\hat{f}}$ will have a unique solution provided the matrix A is non-singular. It can be easily verified that for any vector $\underline{v}$,

SLIDE 19

$$\underline{v}^T A \underline{v} = \sum_{i=1}^{n+1} \sum_{j=1}^{m+1} [\frac{1}{\Delta x^2}(v_{ij} - v_{i-1,j})^2 + \frac{1}{\Delta y^2}(v_{ij} - v_{i,j-1})^2]$$

Hence $\boxed{\underline{v}^T A \underline{v} \geq 0, \text{ for any } \underline{v} \not\equiv 0}$ (A is **SPD**)

$$A\underline{\hat{u}} = \underline{\hat{f}} \ : \ \underline{\underline{\hat{u}} \textbf{ exists} \text{ and is } \textbf{unique}}$$

If $m \approx n$ are large, Gaussian elimination (or LU) is expensive $\Rightarrow \mathcal{O}(n^4)$

For a matrix of size $N \times N$ the cost of performing Gaussian elimination is proprotional to N^3. If the matrix is banded with bandwidth M, the matrix structure can be exploited to reduce the cost to NM^2. Gaussian elimination will be discussed in much greater detail in future lectures. For our matrix A, $N = n^2$ and $M = 2n+1$ hence a total cost of $\mathcal{O}(n^4)$. Iterative methods can also be pursued – these will depend on the condition number, discussed below.

2.5 Error Analysis

2.5.1 Truncation Error

Recall from Lecture 1 that the trucation error is obtained by inserting the exact solution into the difference scheme.

SLIDE 20

$$\begin{aligned} &-\frac{u(x_{i+1}, y_j) - 2u(x_i, y_j) + u(x_{i-1}, y_j)}{\Delta x^2} \\ &-\frac{u(x_i, y_{j+1}) - 2u(x_i, y_j) + u(x_i, y_{j-1})}{\Delta y^2} = f(x_i, y_j) \\ &\underbrace{-\frac{\Delta x^2}{12}\frac{\partial^4 u}{\partial x^4}(x_i + \theta_i^x \Delta x, y_j) - \frac{\Delta y^2}{12}\frac{\partial^4 u}{\partial y^4}(x_i, y_j + \theta_j^y \Delta y)}_{\tau_{i,j}} \end{aligned}$$

For $u \in \mathcal{C}^4$ $\boxed{\tau_{i,j} \sim \mathcal{O}(\Delta x^2, \Delta y^2)}$ for all i, j

2.5.2 $\|\cdot\|_\infty$ Stability

SLIDE 21

It can be shown that

$$\boxed{\|A^{-1}\|_\infty \le \frac{1}{8}}$$

The above estimate can be proved following essentially the same steps as we followed in Lecture 1 for the one dimensional case (see [TW]).

Ingredients:

- Positivity of the coefficients of A^{-1}
- Bound on the maximum row sum

2.5.3 $\|\cdot\|_\infty$ Convergence

SLIDE 22

Error equation $A\underline{e} = \underline{\tau} \quad \Rightarrow \boxed{\underline{e} = A^{-1}\underline{\tau}}$

$$\begin{aligned}
\|\underline{e}\|_\infty &= \|A^{-1}\underline{\tau}\|_\infty \le \|A^{-1}\|_\infty \; \|\tau\|_\infty \le \frac{1}{8}\|\tau\|_\infty \\
&\le \frac{1}{96}(\Delta x^2 \max_{(x,y)\in\Omega} |u_x^{(4)}| + \Delta y^2 \max_{(x,y)\in\Omega} |u_y^{(4)}|)
\end{aligned}$$

$$\boxed{\textbf{If} \;\; u \in \mathcal{C}^4 \quad \|\underline{e}\|_\infty \sim \mathcal{O}(\Delta x^2, \Delta y^2)}$$

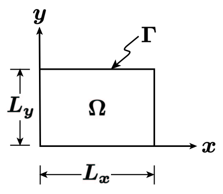
Here, the issue of regularity of the solution u must be stressed, since in most practical situations (e.g. involving general domains), the solution may not have sufficient regularity for the above estimate to hold.

3 Eigenvalue Problem in 2D

3.1 Statement

SLIDE 23

$$\begin{aligned}
-\nabla^2 u = \lambda u & \quad \text{in} \quad \Omega \\
u = 0 & \quad \text{on} \quad \Gamma
\end{aligned}$$



Assume (for simplicity)
$L_x = L_y = 1$

Solutions $(u(x,y), \; \lambda)$

3.2 Exact Solution

SLIDE 24

Eigenvalues

$$u^{k,l}(x,y) = \sin(k\pi x)\sin(l\pi y)$$

In the two dimensional case, it can be verified by direct substitution, that the eigenfunctions are simply the product of the eigenfunctions in the x and y directions. Note that by construction, the eigenfunctions satisfy the boundary conditions.

$$-\nabla^2 u^{k,l} = (k^2\pi^2 + l^2\pi^2)\ u^{k,l}$$

Eigenvectors

$$\lambda^{k,l} = k^2\pi^2 + l^2\pi^2, \qquad k,l = 1,\dots$$

3.3 Discrete Problem

3.3.1 Eigenvectors

SLIDE 25

$$\boxed{A\underline{\hat{u}} = \hat{\lambda}\underline{\hat{u}}} \quad \Rightarrow \quad (\underline{u}^{k,l},\ \hat{\lambda}^{k,l})$$

It can be verified that in the two dimensional case, the eigenvectors of the discrete operator A, conicide with the eigenfunctions of the continuous operator $-\nabla^2$, evaluated at the grid points.

$$\begin{aligned} u^{k,l}_{i,j} &= \sin(k\pi x_i)\ \sin(l\pi y_j) \\ &= \sin(k\pi i\Delta x)\ \sin(l\pi j\Delta y) \\ &= \sin(\frac{k\pi i}{n+1})\ \sin(\frac{l\pi j}{m+1}) \\ & \quad\ k,i = 1,\dots,n \quad l,j = 1,\dots,m \end{aligned}$$

3.3.2 Eigenvalues

SLIDE 26

By direct substitution into the $A\underline{\hat{u}} = \hat{\lambda}\underline{\hat{u}}$ we find that

$$\hat{\lambda}^{k,l} = \frac{2}{\Delta x}\{1-\cos(k\pi\Delta x)\} + \frac{2}{\Delta y}\{1-\cos(l\pi\Delta y)\}$$

Low Modes	High Modes
$\Delta x, \Delta y \to 0$ (k, l fixed)	$k \approx n$, $l \approx m$
$\hat{\lambda}^{k,l} = k^2\pi^2 + l^2\pi^2 + \mathcal{O}(\Delta x^2, \Delta y^2)$	$\hat{\lambda}^{k,l} = 4(n+1)^2 + 4(m+1)^2$ as $\Delta x, \Delta y \to 0$

3.4 Condition Number of A

SLIDE 27

$$\kappa_A \to \frac{4n^2+4m^2}{2\pi^2} \quad \text{as } \Delta x, \Delta y \to 0$$

If $m \approx n$

$$\kappa_A \to \frac{4n^2}{\pi^2}$$

grows (in $\mathbb{R}^2$) as number of grid points.
(better than in 1D, relatively speaking !!)

3.5 Link to $-\nabla^2 u = f$

3.5.1 $||\cdot||$ Error Estimate

The norm $||\cdot||$ should approximate the continuous $p = 2$ norm. In two dimensions this requires multiplication of the vector $p = 2$ norm by the area factor $\sqrt{\Delta x \Delta y}$.

SLIDE 28

Error equation $\quad A\underline{e} = \underline{\tau} \quad \Rightarrow \boxed{\underline{e} = A^{-1}\underline{\tau}}$

$$||\underline{e}||_2 = ||A^{-1}\underline{\tau}||_2 \le ||A^{-1}||_2 \, ||\tau||_2 \le \frac{1}{\hat{\lambda}^{1,1}}||\tau||_2$$

$$(\Delta x \Delta y)^{1/2}||\underline{e}||_2 \le \frac{1}{\hat{\lambda}^{1,1}}(\Delta x \Delta y)^{1/2}||\underline{\tau}||_2$$

$$\boxed{\Rightarrow \qquad ||\underline{e}|| \le \frac{1}{\hat{\lambda}^{1,1}}||\underline{\tau}|| \sim \mathcal{O}(\Delta x^2, \Delta y^2)}$$

4 Discrete Fourier Solution

4.1 Poisson Problem 2D

In most practical situations, the solution to the Poisson equation will be required on general domains, and with more complicated boundary conditions than those considered here. In this case, some of the solution techniques that will be considered later on in this course may be more appropriate. However, in those specific situations where the Poisson equation has to be solved on a rectangle with sufficiently simple boundary conditions, dicrete Fourier techniques may be employed to solve the system of equations $A\underline{\hat{u}} = \underline{\hat{f}}$ very efficiently.

SLIDE 29

A is SPD $\quad \Rightarrow \quad \boxed{A = Z\Lambda Z^T}$
Λ diagonal matrix of eigenvalues $(nm \times nm)$
Z is matrix of eigenvectors $(nm \times nm)$

$$Z = 2\sqrt{\Delta x \Delta y}\begin{pmatrix} \hat{u}^{1,1} & \hat{u}^{2,1} & & \hat{u}^{n,m} \\ \vdots & \vdots & & \vdots \\ \downarrow & \downarrow & & \downarrow \\ \vdots & \vdots & & \vdots \end{pmatrix}$$

SLIDE 30

$$Z\Lambda Z^T\underline{\hat{u}} = \underline{\hat{f}} \quad \Rightarrow \quad Z^T\underline{\hat{u}} = \Lambda^{-1}Z^T\underline{\hat{f}} \quad \boxed{\underline{\hat{u}} = Z\Lambda^{-1}Z^T\underline{\hat{f}}}$$

ALGORITHM

```
1.    $\underline{\hat{f}}^* = Z^T\underline{\hat{f}}$
2.    $\underline{\hat{u}}^* = \Lambda^{-1}\underline{\hat{f}}^*$
3.    $\underline{\hat{u}} = Z\underline{\hat{u}}^*$
```

Still cost is $\mathcal{O}(n^4)$ $(n \approx m)$ ... **BUT** ...

SLIDE 31

- Matrix multiplications can be reorganized (tensor product evaluation) $\boxed{\text{N5}}$ $\Rightarrow \mathcal{O}(n^3)$

- $\underline{\hat{f}}^* = Z^T\underline{\hat{f}}$ $(\underline{\hat{u}} = Z\underline{\hat{u}}^*)$ is a (Inverse) Discrete Fourier Transform

  Using FFT $\Rightarrow \mathcal{O}(n^2 \log n)$

Note 5 ***Tensor Product Evaluation***

An alternative representation of the system $A\underline{\hat{u}} = \underline{\hat{f}}$ is the matrix equation

$$A_x\,\hat{U} + \hat{U}A_y = \hat{F}$$

where the unknowns $\hat{u}_{i,j}$ and the right hand side $\hat{f}_{i,j}$, have been rearranged into an $n \times m$ matrices

$$\hat{U} = \begin{pmatrix} \hat{u}_{11} & \hat{u}_{12} & \hat{u}_{13} & \cdots & \hat{u}_{1m} \\ \hat{u}_{21} & \hat{u}_{22} & \hat{u}_{23} & \ddots & \vdots \\ \hat{u}_{31} & \ddots & \ddots & \vdots & \cdot \\ \vdots & \ddots & \cdot & \cdot & \hat{u}_{n-1m} \\ \hat{u}_{n1} & \cdots & \cdot & \hat{u}_{nm-1} & \hat{u}_{nm} \end{pmatrix}, \quad \hat{F} = \begin{pmatrix} \hat{f}_{11} & \hat{f}_{12} & \hat{f}_{13} & \cdots & \hat{f}_{1m} \\ \hat{f}_{21} & \hat{f}_{22} & \hat{f}_{23} & \ddots & \vdots \\ \hat{f}_{31} & \ddots & \ddots & \vdots & \cdot \\ \vdots & \ddots & \cdot & \cdot & \hat{f}_{n-1m} \\ \hat{f}_{n1} & \cdots & \cdot & \hat{f}_{nm-1} & \hat{f}_{nm} \end{pmatrix}$$

and the A_x and A_y are simply the "one dimensional" matrices, in the x and y directions respectively, of size $n \times n$ and $m \times m$ respectively,

$$A_x = \frac{1}{\Delta x^2}\begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \vdots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix}, \quad A_y = \frac{1}{\Delta y^2}\begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \vdots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix}.$$

Let Z_x and Z_y be the matrices that contain the normalized eigenvectors of A_x and A_y respectively (see slide 8). Thus, we can write

$$A_x = Z_x \Lambda_x Z_x^T, \qquad A_y = Z_y \Lambda_y Z_y^T,$$

where Λ_x and Λ_y are the diagonal matrices of eigenvalues.
By inserting the above expressions for A_x and A_y into our matrix equation, premultiply by Z_x^T and postmultiply by Z_y, we obtain

$$\Lambda_x Z_x^T \hat{U} Z_y + Z_x^T \hat{U} Z_y \Lambda_y = Z_x^T \hat{F} Z_y.$$

Defining $\hat{U}^* = Z_x^T \hat{U} Z_y$ and $\hat{F}^* = Z_x^T \hat{F} Z_y$ we propose the followoing algorithm

1. $\hat{F}^* = Z_x^T \hat{F} Z_y$
2. $\{\hat{U}^*\}_{k,l} = \frac{1}{\lambda_x^k + \lambda_y^l}\{\hat{F}^*\}_{k,l}$ for $k = 1, \dots, n$, $l = 1, \dots, m$.
3. $\hat{U} = Z_x \hat{U}^* Z_y^T$

We observe that the most expensive steps 1 and 3 only involve multiplication of matrices of size n or m. Thus, if $n \approx m$, the cost of the above algorithm scales like $\mathcal{O}(n^3)$.
We have already indicated above that the very special structure of the matrices Z_x and Z_y allow for an even more efficient implementation of the matrix products, using Fast Fourier Transforms (or Fast Sine Transforms in this case) to obtain a nearly optimal solution cost $\mathcal{O}(n^2 \log n)$.

---

5 Non-Rectangular Domains

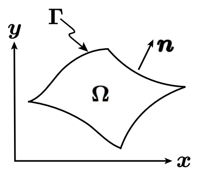
5.1 Poisson Problem 2D

SLIDE 32

Here we consider the extension of the finite difference method to non-rectangular domains.

We are interested in solving

$$\begin{aligned} -\nabla^2 u &= f && \text{in } \Omega \\ u &= g && \text{on } \Gamma_D \\ \frac{\partial u}{\partial n} &= h && \text{on } \Gamma_N = \Gamma \backslash \Gamma_D \end{aligned}$$



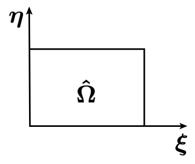
where f, g, and h are given.

We assume that the boundary of Ω, Γ, can be divided into Γ_D and Γ_N, such that $\Gamma_D \cap \Gamma_N = 0$. On Γ_D Dirichlet ($u = g$) boundary conditions are applied whereas Neumann conditions ($\frac{\partial u}{\partial n} = h$) are applied on Γ_N. Mixed, or Robin, type boundary conditions involving the function and the normal derivative could also be easily incorporated.

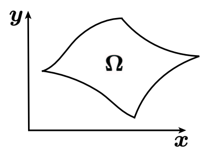
5.1.1 Mapping

SLIDE 33

We require a non-singular mapping between a rectangular region $\hat{\Omega}$, and the domain of interest Ω.



$$\begin{aligned} x &= x(\xi, \eta) \\ y &= y(\xi, \eta) \\ &\longrightarrow \end{aligned}$$



? ? $\qquad$ $-\nabla^2 u = f$

Can we determine an equivalent problem to be solved on $\hat{\Omega}$?

In order to solve our problem on $\hat{\Omega}$, we need our differential equation to be expressed in such a way that all derivatives are taken with respect to the independent variables ξ and η.

Depending on the complexity of Ω, devising this mapping may be very difficult, or simply not possible. There are a number of grid generation techniques specifically devoted to this type (see [TSW] for more details).

5.1.2 Transformed equations

SLIDE 34

By simple chain rule differentiation we have that ...

$$u(x,y) \equiv \underbrace{u\left(x(\xi,\eta), y(\xi,\eta)\right)}_{u(\xi,\eta)} \quad \Rightarrow \quad \begin{array}{rcl} u_x &=& \xi_x u_\xi + \eta_x u_\eta \\ u_y &=& \xi_y u_\xi + \eta_y u_\eta \end{array}$$

How do we evaluate terms ξ_x, η_x, ξ_y, and η_y?

SLIDE 35

$$\begin{array}{c} \xi = \xi(x,y) \\ \eta = \eta(x,y) \end{array} \qquad\qquad \begin{array}{c} x = x(\xi,\eta) \\ y = y(\xi,\eta) \end{array}$$

$$\begin{pmatrix} d\xi \\ d\eta \end{pmatrix} = \begin{pmatrix} \xi_x & \xi_y \\ \eta_x & \eta_y \end{pmatrix} \begin{pmatrix} dx \\ dy \end{pmatrix} \qquad \begin{pmatrix} dx \\ dy \end{pmatrix} = \begin{pmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{pmatrix} \begin{pmatrix} d\xi \\ d\eta \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} \xi_x & \xi_y \\ \eta_x & \xi_y \end{pmatrix} = \begin{pmatrix} x_\xi & x_\eta \\ y_\xi & y_\eta \end{pmatrix}^{-1} = \frac{1}{J} \begin{pmatrix} y_\eta & -x_\eta \\ -y_\xi & x_\xi \end{pmatrix}$$

$$J = x_\xi y_\eta - x_\eta y_\xi$$

SLIDE 36

$$\begin{array}{rcl} u_x &=& \frac{1}{J}\left(y_\eta u_\xi - y_\xi u_\eta\right) \\ u_y &=& \frac{1}{J}\left(-x_\eta u_\xi + x_\xi u_\eta\right) \end{array}$$

and

$$\begin{array}{rcl} u_{xx} &=& \dfrac{\partial}{\partial x}\left(u_x\right) = \left(\xi_x \dfrac{\partial}{\partial \xi} + \eta_x \dfrac{\partial}{\partial \eta}\right) u_x \\ &=& \dfrac{1}{J}\left(y_\eta \dfrac{\partial}{\partial \xi} - y_\xi \dfrac{\partial}{\partial \eta}\right) u_x \\ &=& \ldots \\ u_{yy} &=& \ldots \end{array}$$

SLIDE 37

Finally, $-(u_{xx} + u_{yy}) = f$, becomes

$$\boxed{\frac{-1}{J^2}\left(a\, u_{\xi\xi} - 2b\, u_{\xi\eta} + c\, u_{\eta\eta} + d\, u_\eta + e\, u_\xi\right) = f}$$

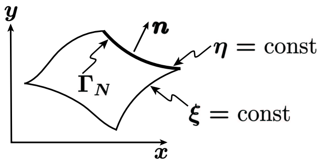
a, b, c, d, and e depend on the mapping.

$$
\begin{array}{lll}
a = x_\eta^2 + y_\eta^2 & e = \dfrac{x_\eta \beta - y_\eta \alpha}{J} & \alpha = a x_{\xi\xi} - 2b x_{\xi\eta} + c x_{\eta\eta} \\
b = x_\xi x_\eta + y_\xi y_\eta & d = \dfrac{y_\xi \alpha - x_\xi \beta}{J} & \beta = a y_{\xi\xi} - 2b y_{\xi\eta} + c y_{\eta\eta} \\
c = x_\xi^2 + y_\xi^2 & &
\end{array}
$$

We note that in this equation all partial derivatives, including the mapping coefficients, a, b, c, d, and e, are with respect to ξ and η.

5.1.3 Normal Derivatives

SLIDE 38



$\boldsymbol{n} = (n^x, n^y)$ is parallel to $\nabla\eta$ (or $\nabla\xi$); e.g., on Γ_N

$$
\boldsymbol{n} = \frac{1}{\sqrt{\eta_x^2 + \eta_y^2}} (\eta_x, \eta_y) = \frac{1}{\sqrt{x_\xi^2 + y_\xi^2}} (-y_\xi, x_\xi)
$$

Similarly, expressions for $\boldsymbol{n}$ can be obtained for the other boundaries.

SLIDE 39

Thus,

$$
\begin{aligned}
\frac{\partial u}{\partial n} = u_x n^x + u_y n^y \; &= \; \frac{1}{J} \; [(y_\eta n^x - x_\eta n^y) \, u_\xi \\
&\qquad + (-y_\xi n^x + x_\xi n^5 y) \, u_\eta]
\end{aligned}
$$

with $(n^x, n^y) = \dfrac{1}{\sqrt{x_\xi^2 + y_\xi^2}} (-y_\xi, x_\xi)$.

REFERENCES

[D] J.W. Demmel, "Applied Numerical Linear Algebra", SIAM, 1997.

[TW] A. Tweito and R. Winther, "Introduction to Partial Differential Equations: A Computational Approach", Texts in Applied Mathematics 29, Springer, 1998.

[TSW] J. Thompson, B. Soni, and N.P. Wheatherill, “Handbook of Grid Generation”, CRC Press, 1999.