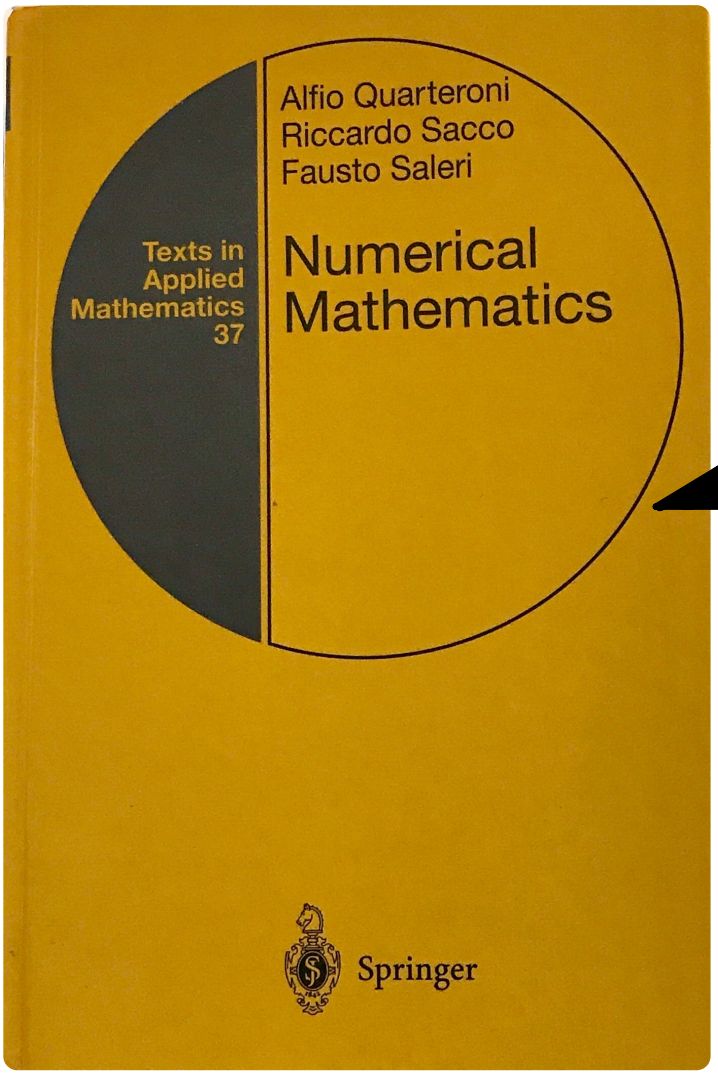
ME 5311

Solution of $A\vec{x} = \vec{b}$

Plan for today:

- Solution methods for linear systems
- Our first "grown up" discussion about CFD
  - Multi-dimensional problems
  - Method of solution is important



Chapters 3 and 4

on HuskyCT

- Great numerical analysis textbook
- Plenty of examples/problems
- Not really a CFD textbook

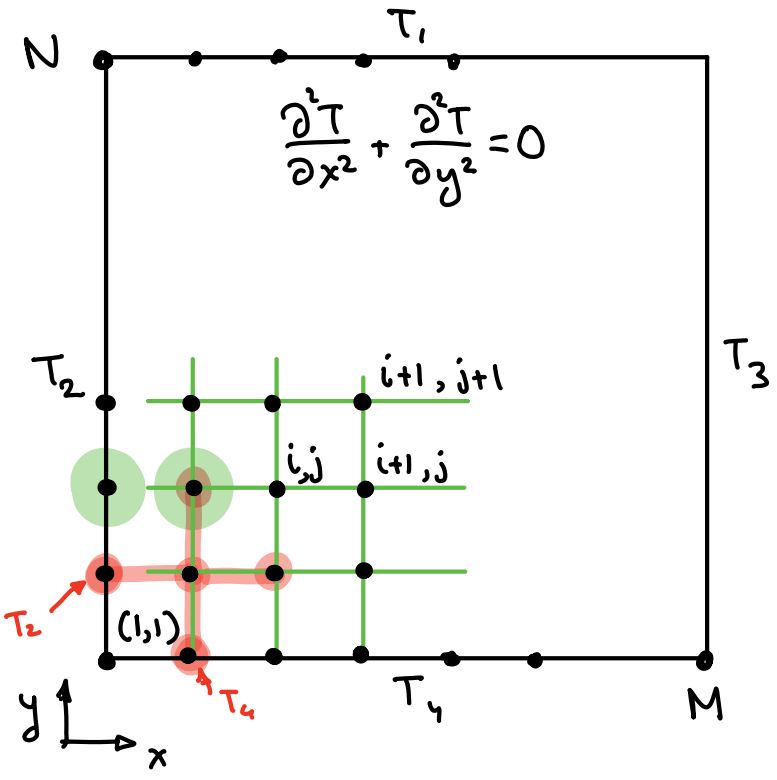
We want to solve $A\vec{x} = \vec{b}$

$n \times n$ $\quad$ $n \times 1$ $\quad$ $n \times 1$

Where do we find $A\vec{x} = \vec{b}$?

- Implicit method → space $\frac{\partial}{\partial x}$, Padé
  ↘ time $\frac{\partial}{\partial t}$
- Steady $A\vec{x} = \vec{b}$
  State problems
- Poisson equation for Pressure

Example A: Steady-state heat equation



Find : $T(x,y)$ steady-state

Steps: 1. Governing eq. (+BC)

2. Discretization

3. Numerical approximation

4. Solution method

→ How many unknowns T_{ij} ?

$(M-2) \times (N-2)$

all the interior points

(x,y)

$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \longrightarrow (i,j)$

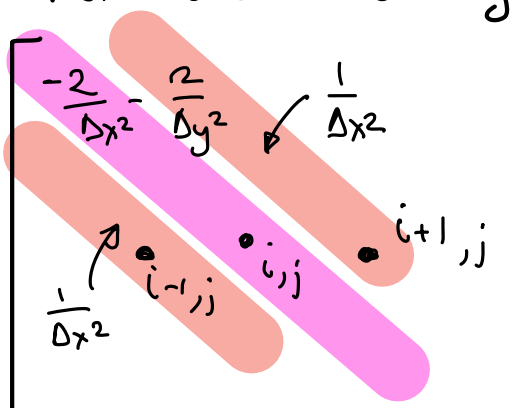
$$\frac{T_{i+1,j} - 2T_{i,j} + T_{i-1,j}}{\Delta x^2} + \frac{T_{i,j+1} - 2T_{i,j} + T_{i,j-1}}{\Delta y^2} = 0$$

$(M-2) \times (N-2)$

$i = 2 \quad , \quad j = 2$

$$\frac{T_{3,2} - 2T_{2,2} + T_2}{\Delta x^2} + \frac{T_{2,3} - 2T_{2,2} + T_4}{\Delta y^2} = 0$$

Form the linear system



$$
\begin{bmatrix}
T_{2,2} & & \\
T_{3,2} & & \\
T_{4,2} & & \\
\vdots & & \\
T_{M-1,2} & & \\
T_{2,3} & & \\
T_{3,3} & & \\
T_{4,3} & & \\
\vdots & & \\
T_{M-1,3} & &
\end{bmatrix}
=
\begin{bmatrix}
+\frac{T_2}{\Delta x^2} & +\frac{T_4}{\Delta y^2} \\
 & \\
 & \\
 & \\
 & \\
+\frac{T_2}{\Delta x^2} & \\
 & \\
 & \\
 & \\
 & \\
 & -
\end{bmatrix}
$$

(2,2)

$\vec{x}$ $\vec{b}$

number of elements

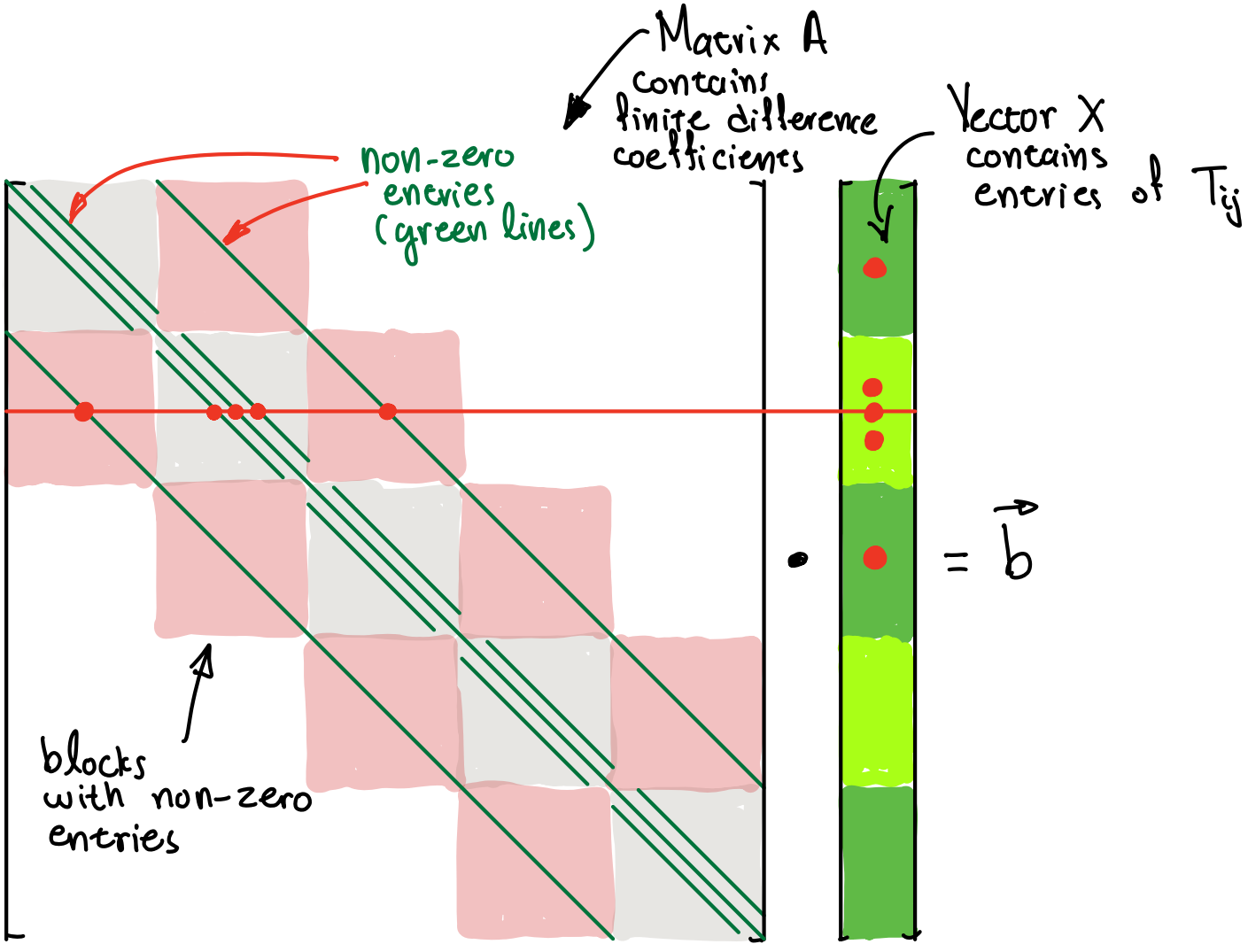
$(M-2)(N-2) \times (M-2)(N-2)$

N_x N_y

number of elements : $(M-2)(N-2)$

Structure of matrix A:

- Block diagonal



What is the computational cost?

Memory / storage

elements

1D $T_i : N_x \rightarrow A = N_x^2$

2D $T_{i,j} : N_x \cdot N_y \rightarrow A = (N_x \cdot N_y)^2$

3D $T_{i,j,k} : N_x N_y N_z \rightarrow = (N_x N_y N_z)^2$

			Memory
$N = 256$	$A :$	65×10^3 $\times$ 8 bytes $\rightarrow$	0.5 MB
		4.3×10^9	34 GB
		2.8×10^{14}	2252 TB

OMG!!!

CPU:

Cramer's $x_j = \frac{D_j}{\det(A)}$ ← det A but j-column with b

Cost is $(n+1)!$ operations

$n = 50 \rightarrow 51! = 1.5 \times 10^{66}$ operations

Intel Xeon "Skylake" 1000 Gflops

1 flop : floating point operation per sec.

1.5×10^{66} / 1000×10^9 $= 1.5 \times 10^{54}$ sec

(# operation / flops)

$= 5 \times 10^{46}$ years

Conclusions:

1. In CFD the boundary between "possible" computation and "not possible" is very sharp

   Δx becomes $\frac{\Delta x}{2}$ in 3D $\Rightarrow$ Δt becomes $\frac{\Delta t}{2}$

   N grid points $\rightarrow$ 2N points in each direction

   N_t time steps $\rightarrow$ $2N_t$ time steps

   Old cost $N^3 N_t$ $\rightarrow$ new cost $2^3 N\, 2 N_t = 16$ (old cost)

2. Good algorithms are important

   Classic exampe is FFT : cost is $N \log N$ operations instead of N^2

   for $N = 1000$: $N^2 = 1{,}000{,}000$

   $N \log N \simeq 7{,}000$

   factor of about 150 difference

Very important:

- The matrix is sparce

  Sparse: most elements are zero
  Dense: (not sparse) most elements are not zero

- Do not form/store the entire matrix A
- Data structure stores only the non-zero elements

  e.g.: row, column, value

  i , j , $a_{i,j}$

- Typically we only need to find $\vec{y} = A\vec{x}$
  - we can do this without explicitly forming A
  - e.g.

```
function y = matrix_multiply(x)
```

- Do not form the inverse
  - The inverse is dense!
  - $x = A^{-1}b$ is just notation

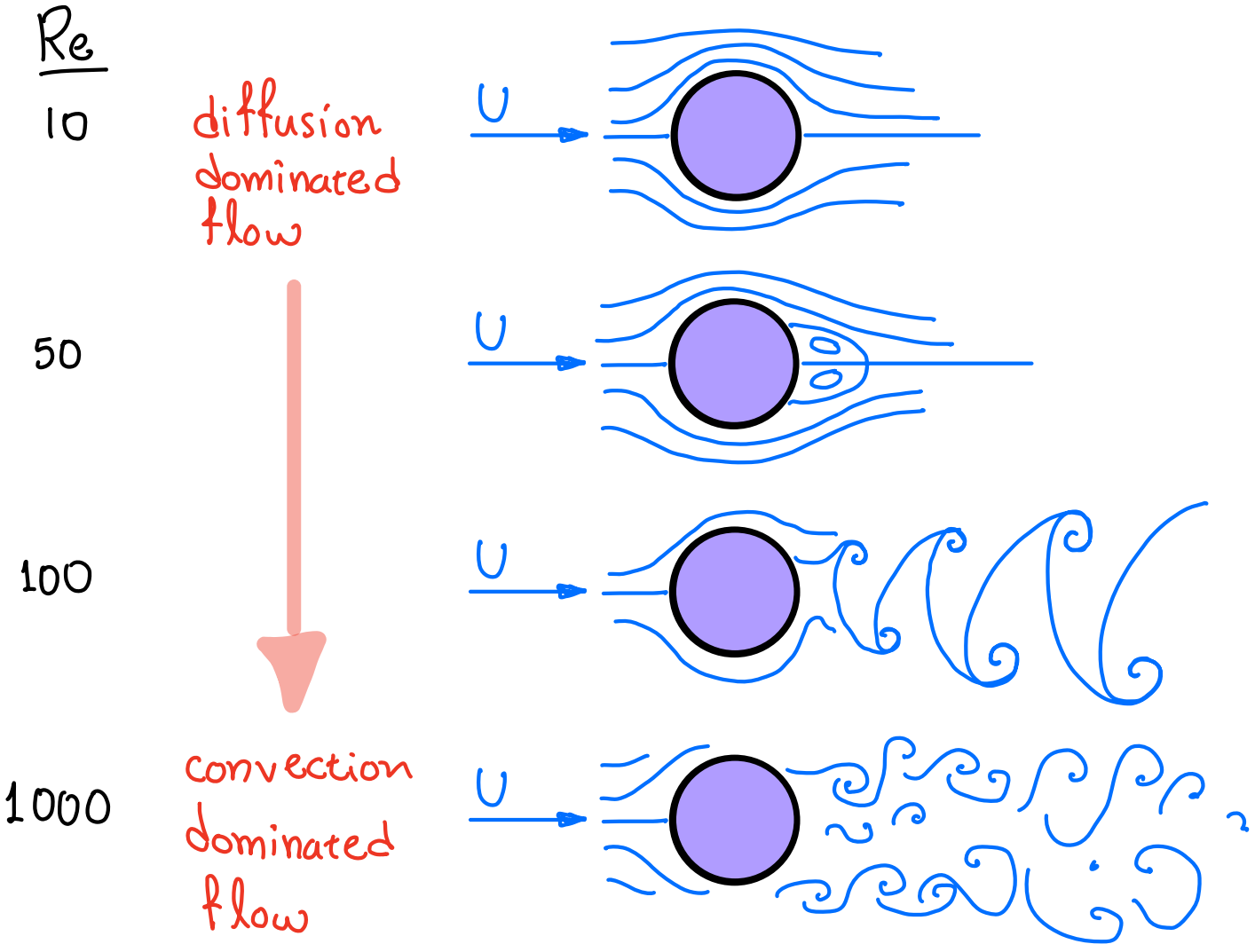
If you need to construct the matrix use a sparce matrix format!

Example B: Convection-Diffusion Problem

Fluid Dynamics Problem: flow around cylinder

Parameters (dimensional): fluid velocity U
fluid kinematic viscosity $\nu = \frac{\mu}{\rho}$
cylinder diameter D

Non-dimensional parameter: Reynolds number $Re = \frac{UD}{\nu}$



Model Problem: One-dimensional convection-diffusion

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \qquad \text{on periodic domain}$$

$$\frac{d\vec{u}}{dt} = \left[-\frac{a}{2\Delta x}B_p(-1,0,1) + \frac{\nu}{\Delta x^2}B_p(1,-2,1)\right]\vec{u}$$

$$\frac{d\vec{u}}{dt} = (A_c + A_d)\vec{u}$$

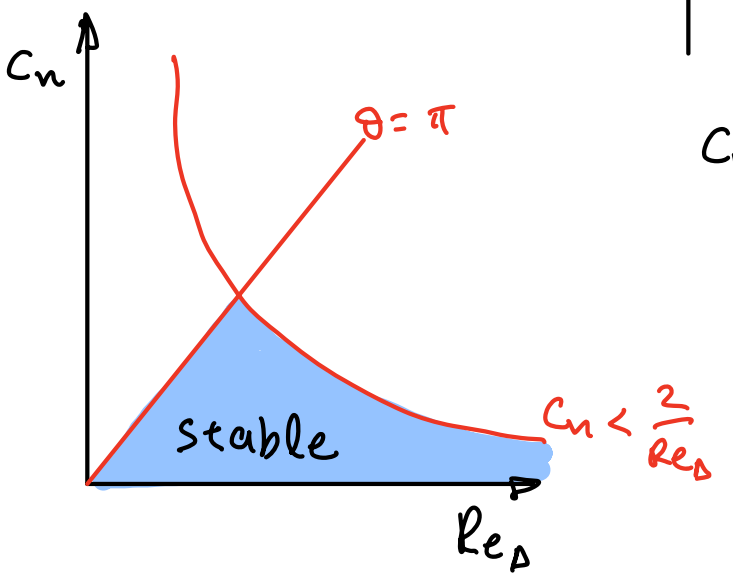
CFL number $C_n = \frac{ah}{\Delta x}$ ← Δt

grid Reynolds $Re_\Delta = \frac{a\Delta x}{\nu}$

Explicit - Explicit

$$\vec{u}^{n+1} = \vec{u}^n + h(A_c + A_d)\vec{u}^n$$

$$|\sigma| = \sqrt{1 + C_n^2 \sin\theta\left[1 - 4\frac{C_n}{Re_\Delta}\sin^2\frac{\theta}{2}\right]}$$



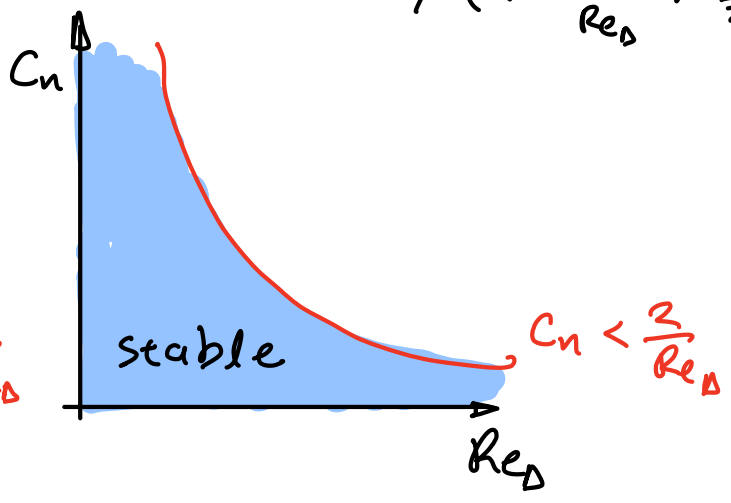
Explicit - Implici-

$$\frac{u^{n+1} - u^n}{h} = A_c u^n + A_d u^{n+1}$$

$$\underbrace{[I - hA_d]}_{A}\vec{u}^{n+1} = [I - hA_c]u^n$$

$$A\vec{x} = \vec{b}$$

$$|\sigma| = \sqrt{1 + C_n^2 \sin\theta} \Big/ \left(1 + 4\frac{C_n}{Re_\Delta}\sin^2\frac{\theta}{2}\right)$$



Solution of $A\vec{x} = \vec{b}$

Direct

given $A, \vec{b}$ $\xrightarrow{\text{seq.}}$ $\vec{x}$

Pro: "exact" $\vec{x}$

Con: computationally expensive

Iterative

start $\vec{x}^{(0)}$ $\xrightarrow{\text{seq}}$ $\vec{x}^{(n)} \simeq \vec{x}$

Pro: "cheaper"

Con: infinite number of iterations

Solution Methods for $A\vec{x} = \vec{b}$:

- **Direct**
- **Iterative**

Direct: given $A, \vec{b}$ perform a sequence of operations to get $\vec{x}$, e.g. Cramer's rule

positive: can get an "exact" $\vec{x}$

negative: is expensive

Iterative: start with an initial guess for $\vec{x}_0$ and produce a sequence $\vec{x}_n \longrightarrow \vec{x}$

positive: much "cheaper" than direct

negative: Infinite number of iterations to get to "exact" $\vec{x}$, must accept an error tolerance

When solution $A\vec{x} = \vec{b}$ exists? $\det(A) \neq 0$

Condition number: $k(A) = \|A\| \, \|A^{-1}\|$

$k = \infty$ if A is singular

Matlab cond(A)

Solution of Triangular systems

$$A = \begin{bmatrix} a_{11} & & & \\ a_{21} & a_{22} & 0 & \\ & a_{32} & \ddots & \\ & & & a_{nn} \end{bmatrix} \qquad \vec{x} = \vec{b}$$

$$\begin{bmatrix} \blacksquare & \\ 0 & \end{bmatrix} \vec{x} = \vec{b} \qquad L\vec{x} = \vec{b}$$

$$\begin{bmatrix} \blacksquare & 0 \\ & \end{bmatrix} \vec{x} = \vec{b} \qquad U\vec{x} = \vec{b}$$

$$x_1 = \frac{b_1}{a_{11}}$$

$$x_2 = \frac{b_2 - a_{21}x_1}{a_{22}}$$

$$\vdots$$

multiplications and division : $\frac{n(n+1)}{2}$

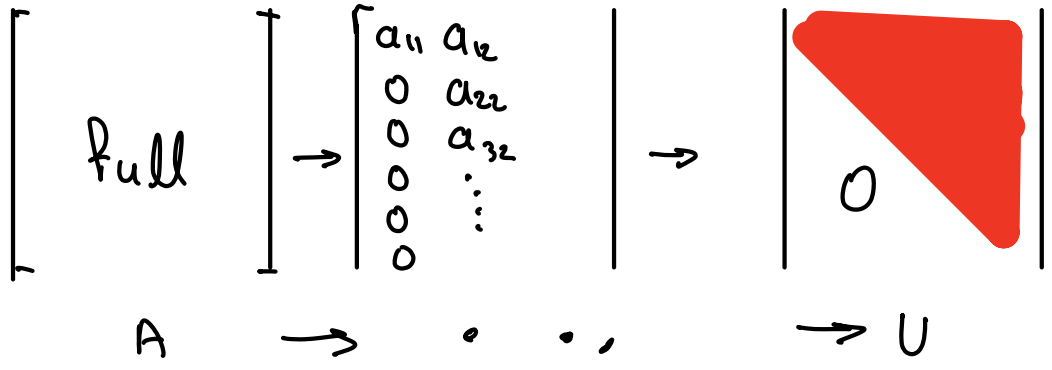
sum and subtractions : $\frac{n(n-1)}{2}$

$$\text{Total} = n^2$$

$A: n \times n$

Gaussian Elimination

$$A\vec{x} = \vec{b} \rightarrow U\vec{x} = \vec{f}$$



A → ∘ ∘ ∘ → U

Gauss elimination : $2(n-1)n(n+1)/3 + n(n-1) \sim \frac{2}{3}n^3$

$+ n^2$ to solve $U\vec{x} = b$

Total : $\frac{2n^3}{3} + 2n^2 \sim O(n^3)$

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 7 & 8 & 9 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 \\ 0 & 0 & -1 \\ 0 & -6 & -12 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 \\ 0 & -6 & -12 \\ 0 & 0 & -1 \end{bmatrix}$$

Pivoting

Partial n^2 searches

complete $2n^3/3$

LU factorization

$$A \to A^{(1)} \to A^{(2)} \to \dots A^{(n)} = U$$

matrix $n \times n$

$$A^{(1)} = M A$$

$$A^{(k+1)} = M_k A^{(k)}$$ ← iteration #

$$M_{n-1} M_{n-2} \dots M_1 A = U$$

$$M_k = \begin{vmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & -m_{k+1,k} & 1 & \\ & & \vdots & & \\ & & & & 1 \end{vmatrix}$$

k

↪ Gaussian transformation matrix

$$A = \underbrace{M_1^{-1} M_2^{-1} \dots M_{n-1}^{-1}}\ U$$

$$A = L\ U$$

↳ lower triangular, diagonal entries = 1

$$A\vec{x} = \vec{b}$$

$$LU\vec{x} = b$$

$$\underbrace{U\vec{x}}_{\vec{y}} = b$$

$$L\vec{y} = \vec{b} \Rightarrow \text{find } \vec{y} \Rightarrow U\vec{x} = \vec{y} \Rightarrow \text{get } \vec{x}$$

cost : $O(n^3)$

" $A^{-1}b$ "

ξ notation

if we have to compute A^{-1} ...

$$AA^{-1} = I$$

$$A\Sigma = I$$

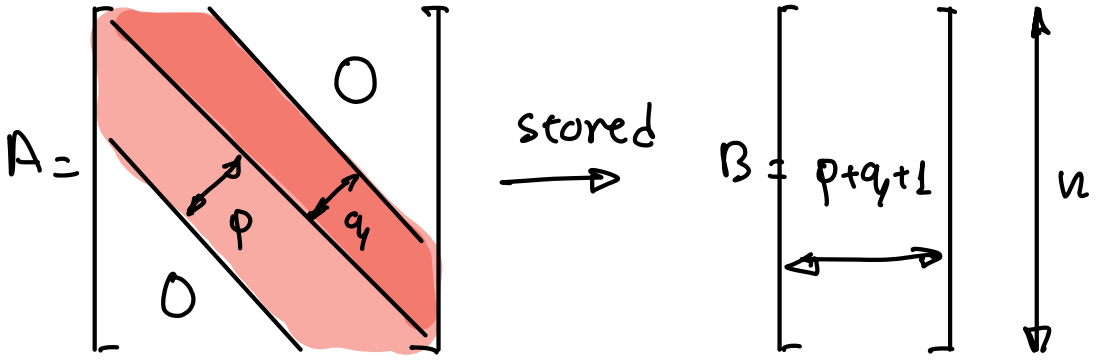
$$A\vec{x}_i = \vec{e}_i \qquad e_i = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \leftarrow i$$

$\xi_{\vec{x}_i}$

$$A = LU$$

$$\left.\begin{aligned} LU\vec{x}_i &= \vec{e}_i \\ L\vec{y} &= \vec{e}_i \\ U\vec{x}_i &= \vec{y}_i \end{aligned}\right\} \text{Solve } 2n \text{ triangular systems}$$

Banded systems



sparce matrix: $\begin{bmatrix} i, j, \text{value} \end{bmatrix}$

Tridiagonal

$$A = \begin{bmatrix} a_1 & c_1 & & & \\ b_2 & a_2 & c_2 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & b_{n-1} & a_{n-1} & c_{n-1} \\ & 0 & & b_n & a_n \end{bmatrix}$$

$$L = \begin{bmatrix} 1 & & & & \\ b_2 & 1 & & & 0 \\ & b_3 & 1 & & \\ & & \ddots & \ddots & \\ 0 & & & \ddots & 1 \end{bmatrix} \qquad U = \begin{bmatrix} \alpha_1 & c_1 & & & \\ & \alpha_2 & c_2 & & 0 \\ & & \ddots & \ddots & \\ & & & \ddots & \\ 0 & & & & \alpha_n \end{bmatrix}$$

$$\alpha_1 = a_1 \quad b_i = \frac{b_i}{\alpha_{i-1}} \quad \alpha_i = a_i - b_i c_{i-1}, \; i = 2, \ldots n$$

Thomas Algorithm:

$L\vec{y} = \vec{f}$

$$y_2 = f_1$$

$$y_i = f_i - b_i y_{i-1}$$

$U\vec{x} = \vec{y}$

$$x_n = \frac{y_n}{\alpha_n}$$

$$x_i = (y_i - c_i x_{i+1}) / \alpha_i$$

Cost: $8n - 7$

Thomas Algorithm

without divisions : $A = L D M^T$

$$\begin{bmatrix} \gamma_1^{-1} & & & \\ b_2 & \gamma_2^{-1} & & \\ & \ddots & \ddots & \\ & & & \end{bmatrix} \begin{bmatrix} \gamma_1 & & & \\ & \gamma_1 & & \\ & & \ddots & \\ & & & \gamma_n \end{bmatrix} \begin{bmatrix} \gamma_1^{-1} & c_1 & & \\ & \gamma_2^{-1} & & \\ & & & \end{bmatrix}$$

$$\gamma_i = (a_i - b_i \gamma_{i-1} c_{i-1})^{-1} \qquad i = 1, \ldots n$$

$$\gamma_0 = 0 \;, \; b_1 = 0 \quad c_n = 0$$

$$L\vec{y} = \vec{f} : \quad y_1 = \gamma_1 b_1 \qquad y_i = \gamma_i (f_i - b_i y_{i-1})$$

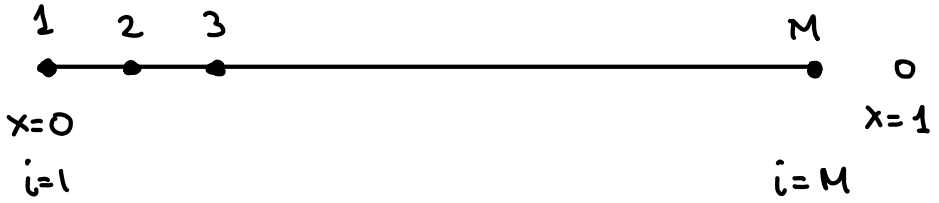
$$U\vec{x} = \vec{y} \qquad x_n = y_n \qquad x_i = y_i - \gamma_i c_i x_{i+1}$$

Direct methods: Using Fourier Transform

Solve 1D Poisson equation $\frac{d^2p}{dx^2} = d(x)$ on $x \in [0, L]$

with periodic BCs.

Discretization: $\frac{1}{\Delta x^2}\left(p_{i-1} - 2p_i + p_{i+1} \right) = d_i$ (1)



$\Delta x = \frac{L}{M}$ uniform grid spacing

Note that $d_i = d(x_i) = d(i\Delta x)$

if $d(x) = \cos(x) \rightarrow d_i = \cos(i\Delta x)$

Solution: need to find $\vec{p}$: $A\vec{p} = \vec{d}$

$$\frac{1}{\Delta x^2}\begin{bmatrix} -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ 1 & & & 1 & -2 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_M \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_M \end{bmatrix}$$

Solution method:

We will take advantage of the periodicity in x and utilize a Fourier expansion

Remember: $p_i = \sum_{k=0}^{M-1} \hat{p}_k e^{\frac{i}{}k\Delta x i}$ (2)

↑ Fourier coefficients from x-direction

Discrete Fourier Transform

Use ② in ①:

$$\sum_{k=0}^{M-1}\left[\frac{1}{\Delta x^2}\hat{p}_k\left(e^{\frac{i}{-}k\Delta x(i-1)} - 2e^{\frac{i}{-}k\Delta x i} + e^{\frac{i}{-}k\Delta x(i+1)}\right)\right] = \sum_{K=0}^{M-1}\hat{d}_k\, e^{\frac{i}{-}k\Delta x i}$$

$$\sum_{k=0}^{M-1}\left[\frac{1}{\Delta x^2}\hat{p}_k\left(e^{\frac{-i}{-}k\Delta x} - 2 + e^{\frac{i}{-}k\Delta x}\right)e^{\frac{i}{-}k\Delta x i}\right] = \sum_{K=0}^{M-1}\hat{d}_k\, e^{\frac{i}{-}k\Delta x i}$$

Because of the orthogonality property of the $e^{\frac{i}{-}k\Delta x i}$ we can write separate equations for each k

Thus...

$$\frac{1}{\Delta x^2}\hat{p}_k\left(e^{\frac{-i}{-}k\Delta x} - 2 + e^{ik\Delta x}\right) = \hat{d}_k \quad \text{for each } k = 0, \ldots, M-1$$

$$\frac{1}{\Delta x^2}\hat{p}_k\left(\cos(-k\Delta x) + \underline{i}\sin(k\Delta x) - 2 + \cos(k\Delta x) + \underline{i}\sin(k\Delta x)\right) = \hat{d}_k$$

$$\frac{1}{\Delta x^2}\hat{p}_k\left(2\cos(k\Delta x) - 2\right) = \hat{d}_k$$

$$\frac{1}{\Delta x^2}\hat{p}_k\left(2\cos\left(\frac{2\pi k}{M}\right) - 2\right) = \hat{d}_k$$

$$\hat{p}_k = \Delta x^2 \hat{d}_k\left(2\cos\left(\frac{2\pi k}{M}\right) - 2\right)^{-1} \qquad k = 0, \ldots, M-1$$

$$p_i = \text{inverse FT}\left(\hat{p}_k\right)$$

Algorithm: Input: Vector of d_i, Δx, M

1. Compute FT of $d_i \rightarrow \hat{d}_k$
2. Compute $\hat{\rho}_k = \Delta x^2 \hat{d}_k \left(2\cos\left(\frac{2\pi k}{M}\right) - 2\right)^{-1}$
3. Compute ρ_i = inverse FT $\hat{\rho}_k$

Note that $\hat{\rho}_{k=0}$ is undefined

Fundamentals:

- Spectrum $\sigma(A)$: the set of eigenvalues λ of A
- Spectral radius $\rho(A) = \max|\lambda| \quad , \lambda \in \sigma(A)$
- $\rho(A) \leq \|A\|$ ← any $\|\cdot\|$
- Positive definite $A\vec{x} \cdot \vec{x} > 0 \quad \forall \vec{x} \neq 0$
  ↑ inner product
- Diagonally dominant
  - by rows $|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \quad i = 1, \dots, n$
  - by columns $|a_{ii}| \geq \sum_{\substack{i=1 \\ i \neq j}} |a_{ij}| \quad j = 1, \dots, n$

- Convergence

$$\vec{x} = \lim_{k \to \infty} \vec{x}^{(k)}$$

↑ sequence of x vectors

$$\vec{e}^{(k)} = \vec{x}^{(k)} - \vec{x}$$

↑ exact

$$\lim_{k \to \infty} e^{(k)} = 0$$

- Sequence $\vec{x}^{(k+1)} = B\vec{x}^{(k)} + \vec{f} \qquad x^{(0)}$ initial guess given

↑ iteration matrix $n \times n$ ↑ $f_n(\vec{b})$ n

- Cost $B\vec{x}$: $O(n^2)$

  LU : $O(\frac{2}{3}n^3)$

  hope that $n^{0.xyz}$

- Consistency : $\vec{f} = (I - B)A^{-1}\vec{b}$ ⟷ equivalent
  $\vec{x} = B\vec{x} + \vec{f}$

- Consistency + $\rho(B) < 1$ => Convergence

$$\vec{x}^{(k+1)} = B\vec{x}^{(k)} + \vec{f}$$

$$\vec{x}^{(k)} = e^{(k)} + \vec{x}$$

$$\underbrace{e^{(k+1)} + \vec{x}}_{x^{(k+1)}} = B\underbrace{(e^{(k)} + \vec{x})}_{x^{(k)}} + \vec{f}$$

$$e^{(k+1)} + \vec{x} = B\vec{x} + \vec{f} + B e^{(k)}$$

$$e^{(k+1)} = B e^{(k)}$$

$$e^{(k+1)} = B^k e^{(0)}$$

$$\lim_{k \to \infty} (B^k e^{(0)}) = 0 \quad \text{iff} \quad \rho(B) < 1$$

or... $1 > \|B\| \geq \rho(B)$

- $\|B^m\|$ convergence factor

  $\|B^m\|^{1/m}$ average convergence factor after m-steps

$$R_m(B) = \frac{1}{m} \log \|B^m\|$$ average convergence rate after m steps

$$R(B) = \lim_{k \to \infty} R_k(B) = -\log \rho(B)$$

↑ asymtotic convergence rate

When to stop ???

$$\|e^{(k)}\| \leq \varepsilon \|e^{(0)}\|$$

↑ tolerance

iterations: $k \geq \frac{\log(\varepsilon)}{\frac{1}{2}\log\|B^k\|} = -\frac{\log(\varepsilon)}{R_k(B)}$ ↑

$$k_{min} \simeq -\frac{\log(\varepsilon)}{R(B)}$$

1. Stop based on increment

$$\|\vec{x}^{(k+1)} - \vec{x}^{(k)}\| < \varepsilon$$

$$\frac{\|\vec{x}^{(k+1)} - \vec{x}^{(k)}\|}{\|\vec{b}\|} < \varepsilon$$

2. Stop based on the residual

$$\frac{\|\vec{r}^{(k)}\|}{\|\vec{b}\|} < \varepsilon \qquad r^{(k)} = A\vec{x}^{(k)} - \vec{b}$$

Linear Iterative methods

$A = P - N$

↑ Preconditioner

$$P\,x^{(k+1)} = N\,x^{(k)} + \vec{b} \qquad k \geq 0$$

$$x^{(k+1)} = \underbrace{P^{-1}N}_{B}\,x^{(k)} + \underbrace{P^{-1}\vec{b}}_{\vec{f}}$$

↑ Iteration matrix

$$x^{(k+1)} = P^{-1}(P-A)\,x^{(k)} + P^{-1}\vec{b}$$

$$= (I - P^{-1}A)\,x^{(k)} + P^{-1}\vec{b}$$

$$= x^{(k)} + P^{-1}\underbrace{(-A\,x^{(k)} + b)}_{r^{(k)}}$$

$$x^{(k+1)} = x^{(k)} + P^{-1}\,r^{(k)}$$

Jacoby:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left[\, b_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}\,x_j^{(k)} \right]$$

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

⇓

$$\frac{1}{\Delta y^2}\left(T_{i,j-1} - 2T_{i,j} + T_{i,j+1}\right) + \frac{1}{\Delta x^2}\left(T_{i-1,j} - 2T_{i,j} + T_{i+1,j}\right) = S_{i,j}$$

$$T_{i,j}^{(k+1)} = \left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}\right)^{-1}\left[\frac{1}{\Delta y^2}\left(T_{i,j-1}^{(k)} + T_{i,j+1}^{(k)}\right) + \frac{1}{\Delta x^2}\left(T_{i-1,j}^{(k)} + T_{i+1,j}^{(k)}\right)\right] + S_{i,j}$$

$P = D$ ← diagonal of A

$N = D - A$ $\Big\}$ → $B_J = D^{-1}(D-A) = I - D^{-1}A$

Over-relaxation method

$$x^{(k+1)} = \frac{\omega}{a_{ii}}\left[b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)} \right] + (1-\omega)\, x_i^{(k)}$$

↑ relaxation parameter ω

$$B_{J_\omega} = \omega B_j + (1-\omega) I$$

↑ Jacobi iteration matrix

$$T_{i,j}^{(k+1)} = \omega \left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2} \right)^{-1} \left[\frac{T_{i+1,j}^{(k)} + T_{i-1,j}^{(k)}}{\Delta x^2} + \frac{T_{i,j+1}^{(k)} + T_{i,j-1}^{(k)}}{\Delta y^2} \right] + (1-\omega)\, T_{i,j}^{(k)}$$

under-relaxation $0 < \omega < 1$

over relaxation $\omega > 1$

Gauss-Siedel:

use available values at $(k+1)$

$$T_{i,j}^{(k+1)} = \omega \left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2} \right)^{-1} \left[\frac{T_{i+1,j}^{(k)} + T_{i-1,j}^{(k+1)}}{\Delta x^2} + \frac{T_{i,j+1}^{(k)} + T_{i,j-1}^{(k+1)}}{\Delta y^2} \right] + (1-\omega)\, T_{i,j}^{(k)}$$

Residual: $r_{i,j} = \dfrac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{\Delta x^2} + \dfrac{T_{i,j-1} - 2T_{i,j} + T_{i,j+1}}{\Delta y^2}$

then,
- $\max |r_{i,j}|$

or
- $\frac{1}{N_x} \frac{1}{N_y} \left(\sum_{i,j} |r_{i,j}|^2 \right)^{1/2}$

$$\| r_{i,j} \| = \max_{i,j} |r|$$

Optimal value of relaxation parameter ω:

$$\omega_{opt} = 1 + \left(\rho / \left(1 + \sqrt{1-\rho^2}\right)\right)^2$$

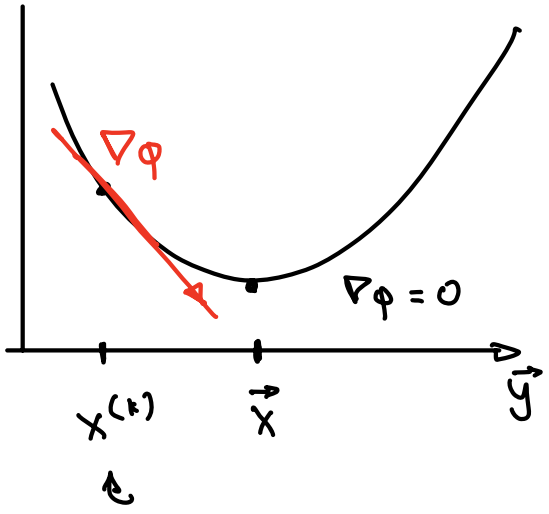
where ρ is the spectral radius of the Jacobi iteration matrix $B_J = I - P^{-1}A$

The Gradient Method

$$\Phi(\vec{y}) = \frac{1}{2}\vec{y}^T A \vec{y} - \vec{y}^T \vec{b}$$

$$\nabla\Phi(\vec{y}) = \frac{1}{2}(A^T + A)\vec{y} - \vec{b} = A\vec{y} - \vec{b}$$

$\nabla\Phi(\vec{x}) = 0$ because $A\vec{x} = \vec{b}$



$\Phi(\vec{y}) > \Phi(\vec{x})$

$$x^{(k+1)} = x^{(k)} + a_k d^{(k)}$$

a_k: scalar; $d^{(k)}$: direction

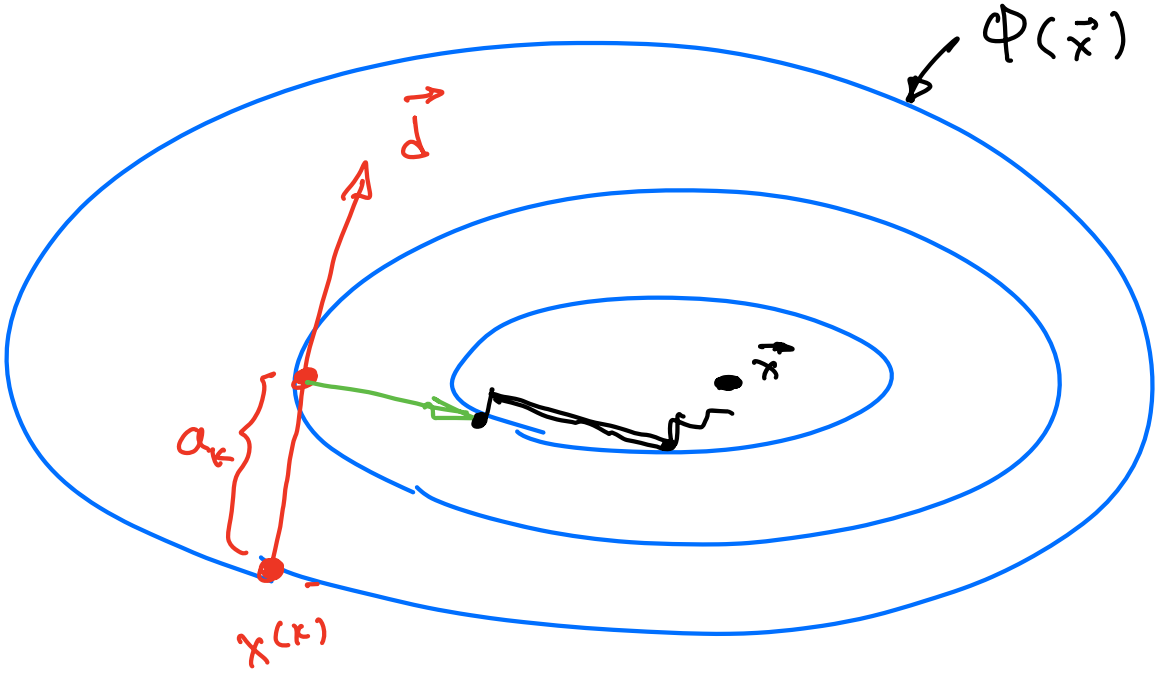
To find a: $\nabla_a \Phi(x^{(k+1)}) \rightarrow a_k = \dfrac{r^{(k)T} r^{(k)}}{r^{(k)T} A r^{(k)}}$

Method: given $x^{(0)}$

$$r^{(k)} = \vec{b} - A x^{(k)}$$

$$a_k = \frac{r^{(k)T} r^{(k)}}{r^{(k)T} A r^{(k)}}$$

$$x^{(k+1)} = x^{(k)} + a_k d^{(k)}$$



The Conjugate Gradient

choose direction $\vec{p}$

$$a_k = \frac{p^{(k)T} r^{(k)}}{p^{(k)T} A p^{(k)}}$$

$$\Phi(\vec{x}^{(k)}) \leq \Phi(\vec{x}^{(k)} + \alpha \vec{p})$$

↑ optimal ↑ w.r.t.

⋮ math

② $p \perp r$ $\quad$ $\vec{p}$ and $\vec{q}$ are A - conjugate

② $x^{(k+1)} = x^{(k)} + \vec{q} \Rightarrow \underbrace{p^T A q = 0}$

$$p^{(k+1)} = r^{(k+1)} + b_k p^{(k)}$$

b_k scalar

Relaxation Method

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

Discretization: $\frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{\Delta x^2} + \frac{T_{i,j-1} - 2T_{i,j} + T_{i,j+1}}{\Delta y^2} = f_{i,j}$

Iteratively solve for $T_{i,j}$

Iteration

use available values at (k+1)

$$T_{i,j}^{(k+1)} = \omega \left(\frac{2}{\Delta x^2} + \frac{2}{\Delta y^2}\right)^{-1} \left[\frac{T_{i+1,j}^{[k]} + T_{i-1,j}^{(k+1)}}{\Delta x^2} + \frac{T_{i,j+1}^{(k)} + T_{i,j-1}^{(k+1)}}{\Delta y^2} - f_{i,j} \right] + (1-\omega) T_{i,j}^{(k)}$$

relaxation parameter $0 < \omega < 2$

Residual

$$r_{i,j} = \frac{T_{i-1,j} - 2T_{i,j} + T_{i+1,j}}{\Delta x^2} + \frac{T_{i,j-1} - 2T_{i,j} + T_{i,j+1}}{\Delta y^2} - f_{i,j}$$

↑ this is a 2D array

we can make a scalar by taking $\max |r_{i,j}|$

Optimal value of relaxation parameter ω:

$$\omega_{opt} = 1 + \left(\rho / \left(1 + \sqrt{1-\rho^2}\right)\right)^2$$

where ρ is the spectral radius of the Jacobi iteration matrix $B_J = I - P^{-1}A$

↑ P = diagonal of A