

# Numerisk lineær algebra for Poissons ligning

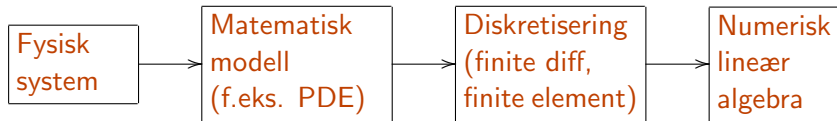
## NTNU

Brynjulf Owren

Institutt for matematiske fag

November 24, 2008

- 1 Motivasjon, generelt om ligningsløsning
- 2 Poisson's ligning i 2 dimensjoner
- 3 Litt om Gauss/LU/Cholesky
- 4 Litt om diagonalisering
- 5 Litt om iterative teknikker



- Ofte er "Numerisk lineær algebra" flaskehalsen
- Grensesprengende simuleringer MÅ utnytte struktur i de lineære ligningene
- Ofte innebærer det å studere alle de tre boksene til venstre.

Generell form

$$A \cdot \mathbf{u} = \mathbf{b}$$

Matrise:  $A$ ,  $m \times m$  (kjent)

Ukjent:  $\mathbf{u}$ ,  $m$ -vektor

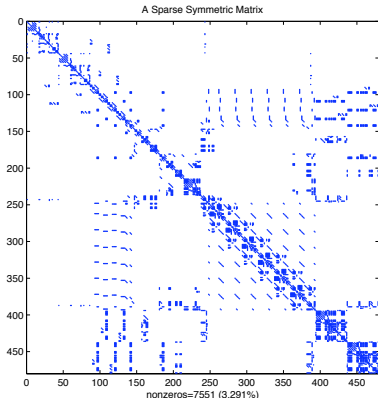
Høyreside:  $\mathbf{b}$ ,  $m$ -vektor (kjent)

- For "små" systemer ("liten  $m$ ") bruk **direkte-metode**, som for eksempel Gauss-eliminasjon.
- **Matlab** har hendig funksjon `\` (backslash) for dette

# Fulle kontra glisne matriser

En  $m \times m$ -matrise har  $m^2$  elementer.

- **Fulle matriser** er matriser der essensielt alle elementer er ulike 0.  $\text{nnz}(A) \sim m^2$  (number-of-non-zeros)
- **Glisne matriser** har mange 0-elementer, typisk  $\text{nnz}(A) \sim m$ , (konstant  $\times m$ ).



Eksempel fra Matlab-demo.  
Diffraksjonskolonne i kjemisk reaktor.

# Om fulle matriser

Gauss-eliminasjon (og flere andre direktemetoder). Anta  $A$   $m \times m$ .

- Beregningskost:  $\mathcal{O}(m^3)$  aritmetiske operasjoner
- Minnebruk:  $\mathcal{O}(m^2)$  bytes

## Example

- $m = 1000 \Rightarrow \sim 10^9$  ops.
- Klokkefrekvens: 1GHz. Ideell cpu: 1 op pr klokkesyklus.  
→ ca 1 sekund. (10 sek mer realistisk?)
- Minnebruk:  $m^2$  flyttall =  $8 m^2$  bytes =  $8M$ bytes.

Minnebegrensninger:

- 32-bits adressering: 4 Gbytes
- 8 bytes pr flyttall → 500 M flyttall
- $m \leq 20\,000$  for lagring av full  $A$ .

# Historien om fulle matriser

1950	$m = 20$	(Wilkinson)
1965	$m = 200$	(Forsythe & Moler)
1980	$m = 2000$	(Linpac)
1995	$m = 20000$	(LAPACK)

Over disse 45 år har

- Størrelsen økt med en faktor  $10^3$
- Datamaskinenes regnehastighet økt med  $10^9$

Dette er konsistent med Gauss-eliminering som har beregningskompleksitet  $\mathcal{O}(m^3)$ .

Det virkelige potensialet ligger i løsning av store **glisne** matriser.

Verktøykassen er fylt av: **Iterative metoder**

Kilde: Trefethen & Bau

## Hva om $m \gg 10^4$ ?

Mange viktige simuleringmodeller opererer i dette regimet.

- Glisne ligningssystemer
- Iterativ løsning
- Parallell prosessering

### Ideell målsetning

Anta

$$A : m \times m, \quad \mathbf{u}, \mathbf{b} : m\text{-vektorer}$$

- Løs  $A\mathbf{u} = \mathbf{b}$  i  $\mathcal{O}(m)$  ops
- Minnebruk:  $\mathcal{O}(m)$  flyttall

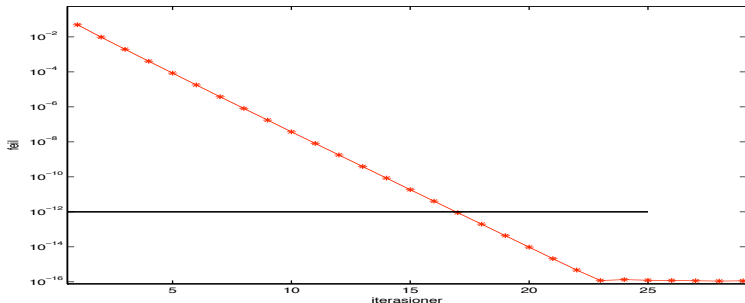


# Direktemetoder og Iterative metoder

- Direktemetoder finner (gitt eksakt aritmetikk) den eksakte løsningen av  $A\mathbf{u} = \mathbf{b}$  i et **endelig** antall operasjoner
- Iterative metoder bruker en **iterasjonsprosess**
  - 1 "Gjett" initiell løsning  $\mathbf{u}^{(0)}$ ,
  - 2 og beregn

$$\mathbf{u}^{(1)}, \mathbf{u}^{(2)}, \dots$$

ved iterasjon,  $\mathbf{u}^{(k+1)} = F(\mathbf{u}^{(k)})$ .



Konseptuelt er iterative metoder ofte enkle å beskrive, enkle algoritmer.

Analysen krever gode matematikk-kunnskaper

- Matriseteori
- Egenverdier, egenvektorer
- Vektorrom, indreprodukt og normer
- Ortogonalitet, projeksjon
- Faktoriseringsprinsipper

## Kan direktemetoder brukes for store $m$ ?

Svaret er **JA**, forbedret teknikk er under utvikling "as we speak".

Ulemper:

- Problemer for veldig store  $m$  (blant annet minnebruk)
- Lite egnet for parallell prosessering.

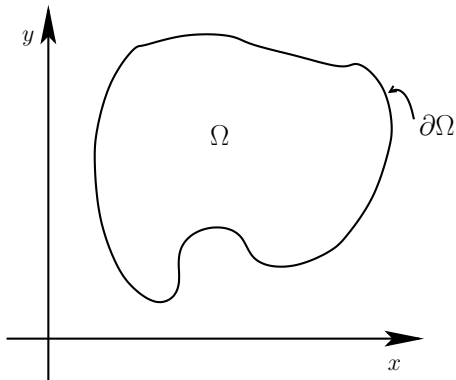
**Spesialtilfelle.** Raske tensorproduktløsere for spesielle problem. Inkluderer Poissons ligning på rektangulært område.

## Poissons ligning i 2D

Finn funksjon  $u = u(x, y)$ , slik at

$$-(u_{xx} + u_{yy}) = f(x, y), \quad (x, y) \in \Omega$$

$$u = g(x, y), \quad (x, y) \in \partial\Omega$$



## Rektangulær $\Omega$ , diskretisering

La  $\Omega = (0, 1) \times (0, 1)$ . Innfør gitter

$$x_i = ih, \quad y_j = jh, \quad h = \frac{1}{N+1}$$

og la  $U_{i,j} \approx u(x_i, y_j)$  for hver  $(x_i, y_j)$ . Approksimer  $u_{xx}, u_{yy}$  i punktet  $(x_i, y_j)$

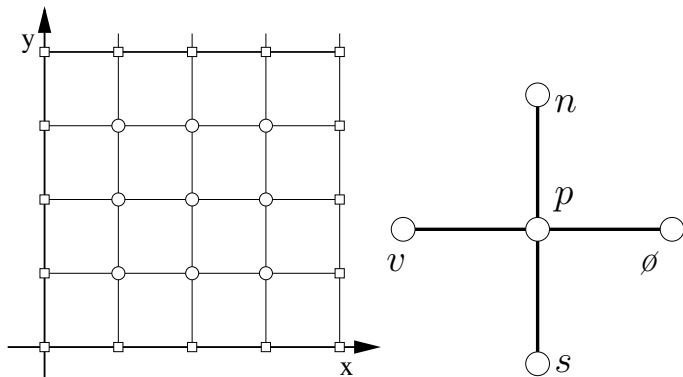
$$(u_{xx})_{ij} \approx \frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{h^2}$$

$$(u_{yy})_{ij} \approx \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{h^2}$$

### Gauss 5-pkts formel

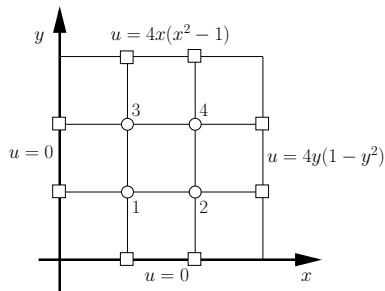
$$4U_{i,j} - U_{i+1,j} - U_{i-1,j} - U_{i,j+1} - U_{i,j-1} = h^2 f_{i,j}$$

# Gauss 5-punkt formel



Gitter og beregningsmolekyl for Poissons ligning.

# Eksempel



La her  $f(x, y) \equiv 0$ . Lag enkelindeks

$$(1, 1) \mapsto 1$$

$$(2, 1) \mapsto 2$$

$$(1, 2) \mapsto 3$$

$$(2, 2) \mapsto 4$$

$$\begin{bmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{bmatrix} \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{32}{27} \\ -\frac{32}{27} \\ 0 \end{bmatrix}$$





# Tre aktuelle løsningsmetoder for $2D$ Poisson

I praksis fins det tre valg

- 1 Gauss-eliminering ( $LU$ -faktorisering)
- 2 Fast Poisson-solver (Diagonalisering). Fungerer for rektangulær  $\Omega$
- 3 Konjugerte gradienters metode. Prekondisjonering er nødvendig. Dette er en iterativ metode.

## Noen poenger rundt Gauss-eliminasjon for 2D-Poisson

- Matrisen  $A$  fra Poisson er **symmetrisk, positiv definitt** (SPD)
- For SPD matriser kan vi bruke **Cholesky**-faktorisering, dvs, vi finner **nedretriangulær**  $L$  slik at  $A = LL^T$ .

$$A = LL^T = \begin{bmatrix} x & 0 & 0 & 0 \\ x & x & 0 & 0 \\ x & x & x & 0 \\ x & x & x & x \end{bmatrix} \cdot \begin{bmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x & x \\ 0 & 0 & 0 & x \end{bmatrix}$$

- **Pivoting** (ombytting av rader og kolonner) er ikke nødvendig for SPD-matriser.
- **Båndform**. Alle elementer i  $A$  nedenfor (ovenfor)  $N$ 'te subdiagonal (superdiagonal) er null. Da vil  $L$  ha null-elementer nedenfor  $N$ 'te subdiagonal

# Beregningskompleksitet i bånd-Cholesky

- Båndmatriser generelt:  $A$  er  $m \times m$  med  $a_{ij} = 0$  når  $i - j > p$ .
- Beregningskostnad for faktorisering er

$$C_g(m, p) \approx mp^2 + 7mp + 2m \text{ når } m \gg p.$$

- I  $2D$  Poisson er  $p = N = \sqrt{m}$  så vi får

$$C(m) = m^2 + \mathcal{O}(m^{3/2}) \text{ ops}$$

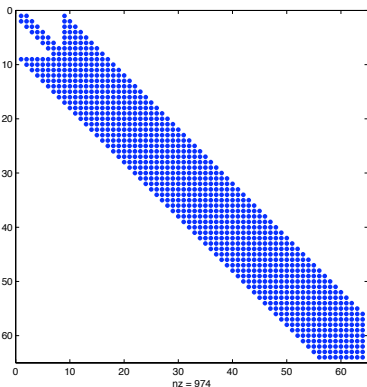
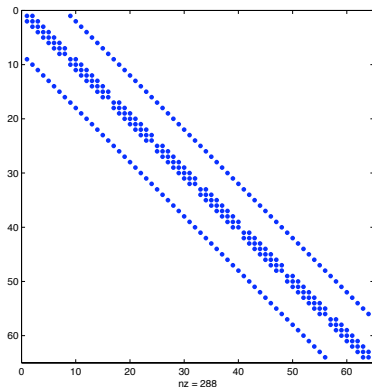
- **Minnekrav.** Vi må lagre elementene i  $L$  som er ulik null.

$$\sum_{k=0}^p (m - k) = mp + m - \frac{p(p+1)}{2}$$

- For  $2D$  Poisson fås

$$\text{Minnekrav: } m^{3/2} + \mathcal{O}(m) \text{ flyttall}$$

# $LU$ -faktorisering og fill-in (Minnebehov)



Matlab SPY-plot av  $A$  og  $L + U$ , der  $A = LU$ .

Minner om 5-punktsformelen

$$4U_{i,j} - U_{i+1,j} - U_{i-1,j} - U_{i,j+1} - U_{i,j-1} = h^2 f_{i,j}$$

La oss lagre ukjent "vektor"  $\mathbf{U}$  i en matrise, slik formelen antyder.  
Definer  $N \times N$  matrisen ( $m = N^2$ )

$$T = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}$$

$$(TU)_{1,j} = 2U_{1,j} - U_{2,j}$$

$$(TU)_{i,j} = 2U_{i,j} - U_{i-1,j} - U_{i+1,j}, \quad 2 \leq i \leq n-1$$

$$(TU)_{n,j} = 2U_{n,j} - U_{n-1,j}$$

Videre er

$$(UT)_{i,1} = 2U_{i,1} - U_{i,2}$$

$$(UT)_{i,j} = 2U_{i,j} - U_{i,j-1} - U_{i,j+1}, \quad 2 \leq j \leq n-1$$

$$(UT)_{i,n} = 2U_{i,n} - U_{i,n-1}$$

Vi konkluderer med at hele 5-punktsformelen kan skrives som en matriseligning

$$TU + UT = h^2 \mathbf{F} =: \mathbf{G}$$

# Diagonalisering

- Anta at vi har kjennskap til **egenverdiene** og **egenvektorene** til matrisen  $T$

$$T = Q\Lambda Q^T, \quad \Lambda = \text{diag}(\lambda_1, \dots, \lambda_N), \quad Q^T Q = I$$

- $\Lambda$  er en diagonalmatrise med egenverdiene til  $T$  på diagonalen. Fordi  $T$  er symmetrisk (og PD) er  $\lambda_i$  reell (og positiv).
- $Q$  er en **ortogonal** matrise hvis kolonner er egenvektorene til  $T$ . For slike matriser er  $Q^{-1} = Q^T$ .
- Sett inn faktoriseringen for  $T$ .

$$Q\Lambda Q^T \mathbf{U} + \mathbf{U}Q\Lambda Q^T = \mathbf{G}$$

$$\Lambda \underbrace{(Q^T \mathbf{U} Q)}_{\tilde{\mathbf{U}}} + \underbrace{(Q^T \mathbf{U} Q)}_{\tilde{\mathbf{U}}} \Lambda = \underbrace{Q^T \mathbf{G} Q}_{\tilde{\mathbf{G}}}$$

Vi har da

$$\Lambda \tilde{\mathbf{U}} + \tilde{\mathbf{U}} \Lambda = \tilde{\mathbf{G}}$$

Men fordi  $\Lambda$  er diagonal blir dette på komponent form

$$\lambda_i \tilde{U}_{i,j} + \tilde{U}_{i,j} \lambda_j = \tilde{G}_{ij}$$

som løses uten videre som

$$\tilde{U}_{i,j} = \frac{\tilde{G}_{i,j}}{\lambda_i + \lambda_j}$$



# Diagonalisering – algoritme

Vi kan løse problemet i 3 steg (gitt  $\lambda_i$  og  $Q$ )

1 Beregn

$$\tilde{\mathbf{G}} = h^2 \mathbf{Q}^T \mathbf{F} \mathbf{Q}$$

(matrise-matrise-produkt)

2 Finn elementene i  $\tilde{\mathbf{U}}$

$$\tilde{U}_{i,j} = \frac{\tilde{G}_{i,j}}{\lambda_i + \lambda_j}, \quad i, j = 1, \dots, N$$

3 Sett

$$\mathbf{U} = \mathbf{Q} \tilde{\mathbf{U}} \mathbf{Q}^T$$

(matrise-matrise-produkt)

Merk: Alle involverte matriser er  $N \times N$  der  $N = \sqrt{m}$

- 1 Generell kostnad for produkt av to  $N \times N$ -matriser er  $\sim 2N^3$  ops. Dermed  $4N^3$  for to produkter.
- 2 Dominerende kost  $N^2$  divisjoner.
- 3 Nye 2 matrise-matrise produkter  $4N^3$  ops.
  - Total kostnad  $\approx 8N^3 = 8m^{3/2}$  ops.
  - Kan reduseres til  $\mathcal{O}(N^2 \log N) = \mathcal{O}(m \log m)$  hvis FFT brukes ved matrisemultiplikasjon.
  - Lagringskrav  $\mathcal{O}(N^2)$  flyttall, dvs  $\mathcal{O}(m)$  flyttall.

Konklusjon: For tensor-gitter (Poisson på rektangulært område) er diagonalisering bra sammenlignet med Cholesky (Gauss).

# Konjugerte gradienters metode (Iterativ)

- Algoritmen koster  $\mathcal{O}(m)$  ops pr iterasjon.
- Utdfordring: For  $2D$  Poisson er antall påkrevde iterasjoner  $\mathcal{O}(m)$ , så total kostnad blir  $\mathcal{O}(m^2)$
- Løsning: Bruk prekondisjonering.

Generelt

$$\begin{aligned} \mathbf{A}\mathbf{u} &= \mathbf{b} \\ \Updownarrow \\ \mathbf{S}^{-1}\mathbf{A}\mathbf{T}^{-1}\bar{\mathbf{u}} &= \mathbf{S}^{-1}\mathbf{b}, \quad \bar{\mathbf{u}} = \mathbf{T}\mathbf{u} \end{aligned}$$

Formål: Bestem  $\mathbf{S}$  og  $\mathbf{T}$  slik at  $\mathbf{S}^{-1}\mathbf{A}\mathbf{T}^{-1}$  er en mer "godartet" matrise enn  $\mathbf{A}$  selv.

Anta for enkelhets skyld at  $T = I$ , så vi har systemet

$$S^{-1}A\mathbf{u} = S^{-1}\mathbf{b}$$

- På en eller annen måte approksimerer  $S$  matrisen  $A$  slik at  $\tilde{A} := S^{-1}A \approx I$  (men ikke for god approksimasjon)
- Konjugerte gradienters metode har som en hovedingrediens å multiplisere  $A$  med vektorer

$$\mathbf{w}_j = A\mathbf{p}_j$$

- Det er ikke slik at  $\tilde{A}$  dannes eksplisitt i programmet.
- Det viser seg at vi kun trenger ( i hver iterasjon) å multiplisere  $S^{-1}$  med en vektor  $\mathbf{r}_j$ .
- heller ikke  $S^{-1}$  foreligger vanligvis som en eksplisitt matrise, men er kun resultatet av en prosess som approksimativt løser  $A\mathbf{z}_j = \mathbf{r}_j$ .

Det fins to svært interessante tolkninger av  $S^{-1}$  som en approksimativ løser for Poisson  $2D$ -problemet. Disse er

- Domene-dekomposisjon
- Multigridalgoritmen

Her er noen referanser

- 1 **Y. Saad**: Iterative Methods for Sparse Linear Systems.
- 2 **L.N. Trefethen & D. Bau**, Numerical Linear Algebra.
- 3 **J.W. Demmel**, Applied Numerical Linear Algebra .
- 4 **G. Golub & C. Van Loan**: Matrix Computations.
- 5 **W. L. Briggs, V.E. Henson, S.F. Mc Cormick**: A multigrid tutorial

**Takk for oppmerksomheten!**