Numerisk lineær algebra

Arne Morten Kvarving

Department of Mathematical Sciences
Norwegian University of Science and Technology

29. Oktober 2007

Problem og framgangsmåte

- Vi vil løse

$$\underline{A}\,\underline{x} = \underline{b}, \quad \underline{b}, \underline{x} \in \mathbb{R}^N, \underline{A} \in \mathbb{R}^{N \times N}.$$

- Vi vet at løsningen er gitt ved

$$\underline{x} = \underline{A}^{-1}\underline{b}$$

- Oppstår *veldig* ofte i numerikk, enten som eget problem eller oftere som en underbit av andre algoritmer.
- Navngivning:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \ddots & & \\ a_{N1} & \cdots & & a_{NN} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \ddots \\ x_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \ddots \\ b_N \end{bmatrix}$$

Løsningsstrategi

- Løsningsstrategi avhengig av struktur og egenskaper til $\underline{A}$.
- Banalt eksempel: $\underline{A}$ er en ortogonalmatrise. Da vet vi at

$$\underline{A}^{-1} = \underline{A}^{T}$$

så vi løser simpelthen systemet ved

$$\underline{x} = \underline{A}^{T}\underline{b}.$$

- SPD
- $\underline{A} = \mathrm{diag}\left(a_{11}, a_{22}, \cdots, a_{NN}\right)$ Løser ved

$$x_i = \frac{b_i}{a_{ii}}, \quad \forall\ i \in [1, N].$$

- Glisne

Generelle matriser

- For generelle matriser har dere lært løsningsstrategien i Matte 3.
- Cramer's regel - baserer seg på å finne N'te ordens determinanter. Dette har en beregningskompleksitet som skalerer med $N!$. Ubrukelig i praksis!
- Gausseliminering.

Gausseliminering

- Gausseliminering er en systematisk eliminasjonsprosess som lar oss skrive matrisen på triangulær form

$$\begin{bmatrix} x & x & x & x & x \\ 0 & x & x & x & x \\ 0 & 0 & x & x & x \\ 0 & 0 & 0 & x & x \\ 0 & 0 & 0 & 0 & x \end{bmatrix}$$

- Når vi har matrisen på triangulær form kan vi enkelt løse systemet ved bakoversubstitusjon.

$$\begin{aligned} 3x_1 + 5x_2 + \quad 2x_3 &= 8 \\ 8x_2 + \quad 2x_3 &= -7 \\ 6x_3 &= 3 \end{aligned}$$

Bakoversubstitusjon

- Nederste variabel er ikke koblet til noen andre:

$$x_3 = \frac{3}{6} = \frac{1}{2}$$

- Når vi har en verdi for x_3 kan vi plugge denne inn i ligning to og vi har igjen en ligning som ikke kobler til noen ukjente:

$$x_2 = \frac{-7 - 2x_3}{8} = \frac{-7 - 1}{8} = -1.$$

- Samme for siste ligning:

$$x_1 = \frac{8 - 2x_3 - 5x_2}{3} = \frac{8 - 1 + 5}{3} = 4.$$

- Fullstendig systematisk framgangsmåte $\Rightarrow$ egnet for implementasjon på datamaskin.

Hvordan finne triangulær form

- Som før er systemet vårt på formen

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &\quad + \cdots + a_{1N}x_N &&= b_1 \\ a_{21}x_1 + a_{22}x_2 &\quad + \cdots + a_{2N}x_N &&= b_2 \\ \vdots & &&= \vdots \end{aligned}$$

- Vil bli kvitt $a_{21}x_1$. Gjør dette ved å utføre operasjonen (rad 2)-$\frac{a_{21}}{a_{11}}$(rad 1).
- Dette gir systemet

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &\quad + \cdots + a_{1N}x_N &&= b_1 \\ 0 + \left(a_{22} - \frac{a_{21}}{a_{11}}a_{12}\right)x_2 &\quad + \cdots + a_{2N}x_N &&= b_2 - \frac{a_{21}}{a_{11}}b_1 \\ \vdots & &&= \vdots \end{aligned}$$

Hvordan finne triangulær form

- Gjentar dette
  - For all rader
  - For all kolonner under diagonalen
- Krav: Vi trenger $a_{kk} \neq 0$. Hvis ikke bytter vi om rekkefølgen på linjene - dette kalles *pivotering*.
- I tillegg bør $a_{kk} >> 0$ for å unngå kanselleringsfeil. Vi minimerer dette problemet ved å alltid plukke raden med størst verdi for a_{kk} - kalles for *delvis pivotering*.

LU-faktorisering

- Dette er Gausseliminasjon (nesten) slik som Matlab gjør det - og generelt slik det er implementert i det meste av programvare som brukes i dag.
- Vi vil finne to matriser $\underline{L}$ og $\underline{U}$ slik at

$$\underline{A} = \underline{L}\,\underline{U}\,.$$

  hvor $\underline{L}$ er nedretriangulær og $\underline{U}$ er øvretriangulær.
- Hvorfor? Jo fordi

$$\begin{aligned} \underline{A}\,\underline{x} &= \underline{b} \\ \underline{L}\,\underline{U}\,\underline{x} &= \underline{b} \Rightarrow \\ \underline{L}\,\underline{v} &= \underline{b} \\ \underline{U}\,\underline{x} &= \underline{v} \end{aligned}$$

  hvor vi kan løse de to siste ligningene først med foroversubstitusjon, og så ved bakoversubstitusjon.

Doolittles metode

Doolittle: Velg 1'ere på diagonalen til $\underline{L}$:

- For et eksempelsystem av dimensjon 2;

$$\underline{A} = \begin{bmatrix} 2 & 3 \\ 8 & 5 \end{bmatrix} = \begin{bmatrix} 1 & \\ l_{21} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} \\ & u_{22} \end{bmatrix} = \underline{L}\,\underline{U}$$

Dette gir

$$\begin{aligned} u_{11} = 2, u_{12} = 3 \\ 2l_{21} = 8 \Rightarrow l_{21} = 4 \\ 4 \cdot 3 + u_{22} = 5 \Rightarrow u_{22} = -7 \end{aligned}$$

altså

$$\underline{L} = \begin{bmatrix} 1 & \\ 4 & 1 \end{bmatrix}, \underline{U} = \begin{bmatrix} 2 & 3 \\ & -7 \end{bmatrix}$$

Doolittles metode - algoritme

- Vi gjør følgende steg

$$u_{1k} = a_{1k} \qquad k = 1, \cdots, N$$

$$l_{j1} = \frac{a_{j1}}{u_{11}} \qquad j = 2, \cdots, N$$

$$u_{jk} = a_{jk} - \sum_{s=1}^{j-1} l_{js} u_{sk} \qquad k = j, \cdots, N, j \geq 2$$

$$l_{jk} = \frac{1}{u_{kk}} \left(a_{jk} - \sum_{s=1}^{k-1} l_{js} u_{sk} \right) \qquad j = k+1, \cdots, N, k \geq 2$$

Crouts metode

Crout: Velg 1'ere på diagonalen til $\underline{U}$:

- For et eksempelsystem av dimensjon 2;

$$\underline{A} = \begin{bmatrix} 2 & 3 \\ 8 & 5 \end{bmatrix} = \begin{bmatrix} l_{11} & \\ l_{21} & l_{22} \end{bmatrix} \begin{bmatrix} 1 & u_{12} \\ & 1 \end{bmatrix} = \underline{L}\,\underline{U}$$

Dette gir

$$l_{11} = 2, l_{21} = 8$$

$$2u_{12} = 3 \Rightarrow u_{12} = \frac{3}{2}$$

$$8 \cdot \frac{3}{2} + l_{22} = 5 \Rightarrow l_{22} = -7$$

altså

$$\underline{L} = \begin{bmatrix} 2 & \\ 8 & -7 \end{bmatrix}, \underline{U} = \begin{bmatrix} 1 & \frac{3}{2} \\ & 1 \end{bmatrix}$$

LU-faktorisering

- Begge disse faktoriseringene er *unike*.
- Noen matriser kan ikke LU-faktoriseres. Da må vi bytte rader i matrisene.

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

- Den åpenbare fordelen med å gjøre Gausseliminasjon på denne formen er at vi kan kalkulere $\underline{L}$ og $\underline{U}$ en gang, for så å løse systemet for mange forskjellige $\underline{b}$. På vanlig form må vi modifisere vektoren $\underline{b}$ før vi kan løse...

Choleskys metode

- Vi har nå en symmetrisk og positiv definitt (SPD) matrise $\underline{A}$, dvs
  - $\underline{x}^T \underline{A} \underline{x} > 0 \quad \forall\, \underline{x} \neq \underline{0}$.
  - $\underline{A}$ er symmetrisk, dvs $\underline{A} = \underline{A}^T$.
  - Dette er det samme som at alle egenverdiene til $\underline{A}$ er reelle og positive.
  - Vi kan da velge $\underline{U} = \underline{L}^T$! Med generelle verdier på diagonalen.

Choleskys metode - eksempel

- Vi bruker matrisen

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}.$$

  Matrisen er åpenbart symmetrisk og den har egenverdier $\lambda_1 = 1, \lambda_2 = 3$ ergo den er SPD.

- Vi vil altså finne

$$\underline{A} = \underline{L}\,\underline{L}^T = \begin{bmatrix} a & \\ b & c \end{bmatrix} \begin{bmatrix} a & b \\ & c \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

Choleskys metode - eksempel

- Vi bestemmer verdiene for a, b, c ved

$$a^2 = 2 \Rightarrow a = \sqrt{2}$$

$$ab = 1 \Rightarrow b = \frac{1}{\sqrt{2}} = \frac{\sqrt{2}}{2}$$

$$b^2 + c^2 = 2 \Rightarrow c = \sqrt{2 - \frac{1}{2}} = \frac{\sqrt{6}}{2}$$

Choleskys metode - algoritme

- Vi gjør følgende steg

$$l_{11} = \sqrt{a_{11}} \tag{1}$$

$$l_{j1} = \frac{a_{j1}}{l_{11}} \tag{2}$$

$$l_{jj} = \sqrt{a_{jj} - \sum_{s=1}^{j-1} l_{js}^2} \quad j = 2, \cdots, N \tag{3}$$

$$l_{pj} = \frac{1}{l_{jj}} \left(a_{pj} - \sum_{s=1}^{j-1} l_{js} l_{ps} \right) \quad p = j+1, \cdots, N, j \geq 2 \tag{4}$$

Choleskys metode - nok et eksempel

For å vise hvordan dette fungerer i praksis skal vi se på problemet

$$\underline{A} = \begin{bmatrix} 4 & 2 & 14 \\ 2 & 17 & -5 \\ 14 & -5 & 83 \end{bmatrix} = \begin{bmatrix} l_{11} & & \\ l_{21} & l_{22} & \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ & l_{22} & l_{32} \\ & & l_{33} \end{bmatrix}.$$

(1)

$$l_{11} = \sqrt{a_{11}} = 2$$

(2)

$$l_{21} = \frac{a_{21}}{l_{11}} = 1, l_{31} = \frac{a_{31}}{l_{11}} = 7$$

(3)

$$l_{22} = \sqrt{a_{22} - l_{21}^2} = \sqrt{17 - 1} = 4$$

Choleskys metode - nok et eksempel

$$\underline{A} = \begin{bmatrix} 4 & 2 & 14 \\ 2 & 17 & -5 \\ 14 & -5 & 83 \end{bmatrix} = \begin{bmatrix} l_{11} & & \\ l_{21} & l_{22} & \\ l_{31} & l_{32} & l_{33} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} \\ & l_{22} & l_{32} \\ & & l_{33} \end{bmatrix}.$$

(4)

$$l_{32} = \frac{1}{l_{22}} \left(a_{32} - l_{21} l_{31} \right) = \frac{1}{4} \left(-5 - 7 \cdot (-1) \right) = -3$$

(3)

$$l_{33} = \sqrt{a_{33} - l_{31}^2 - l_{32}^2} = \sqrt{83 - 7^2 - (-3)^2} = 5$$

Choleskys metode

Theorem

Choleskyfaktorisering er numerisk stabil.

$$a_{jj} = l_{j1}^2 + \cdots + l_{jj}^2 \quad \text{(kvadrerer (3))}$$

Så alle element inni her tilfredsstiller

$$l_{jk}^2 \leq l_{j1}^2 + \cdots + l_{jj}^2 = a_{jj}$$

Betyr at vi ikke får tall som vokser seg store $\Rightarrow$ liten risiko for kanselleringsfeil for "snille" $\underline{A}$.

Hvordan lage invers

- Generelt lager vi *aldri* inverser med mindre det virkelig trengs.
- Hvis vi virkelig må er dette en framgangsmåte: Løs

$$\underline{A}\,\underline{x} = \underline{b} \text{ for } \underline{b} \in \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \cdots$$

- Sett så sammen de oppnådde $\underline{x}$-vektorene til en matrise $\Rightarrow \underline{A}^{-1}$.
- Dette er det samme som å løse matrise-matrise-ligningen

$$\underline{A}\,\underline{X} = \underline{I}$$

Evaluering av numerisk metode

- Når vi skal evaluere effektiviteten av en numerisk algoritme må vi ta flere aspekter i betraktning. Disse inkluderer
  - minnekrav
  - tidskrav
  - stabilitet
  - for parallelle NUMA-maskiner; datalokalitet.
- Som tidligere nevnt; tidskravet kan måles på to måter
  - faktisk brukeropplevd tid / veggklokke tid. Denne varierer sterkt med maskinvare.
  - Hvordan mengden operasjoner som kreves skalerer med problemstørrelsen - mye mer interessant.

  Dette betyr at en bedre datamaskin ikke løser problem fortere, den kan derimot løse større problem.

Evaluering av numerisk metode

- Veldig fristende å tenke at dagens datamaskiner er så kraftige at det gjør ikke noe om vi ikke gjør ting på den mest effektive måten. Jeg vil påstå at det er motsatt! Jo kraftigere datamaskiner vi har, jo viktigere er det av vi løser problem den på "riktig" måten. Hvorfor? Jo, fordi algoritmer sjeldent skalerer lineært. Med bedre maskin kan vi løse større problem, men siden algoritmene ikke er lineære blir vi straffet enda hardere enn før hvis vi gjør ting på feil måte.
- Eksempel: Gausseliminasjon krever $\mathcal{O}\left(N^3\right)$ operasjoner - AU!

N	T
10	$1000\text{s} = 17\text{min}$
100	$1 \cdot 10^6\text{s} = 11$ dager
1000	$1 \cdot 10^9\text{s} = 31$ år

Her er det verdt å merke seg at $N = 1000$ slettes ikke er et stort problem - vi løser ofte problemer med millioner av ukjente.

Hvordan lage invers på en lurere måte

- Dette var en veldig arbeidssom måte å finne inversen på - vi må løse N system som hver krever $\mathcal{O}\left(N^3\right)$ operasjoner $\Rightarrow$ metoden er $\mathcal{O}\left(N^4\right)$ (kan reduseres til $\mathcal{O}\left(N^3\right)$ vha $\underline{L}\,\underline{U}$-faktorisering.)
- Fins bedre metoder - Gauss-Jordan-eliminering.
- I praksis nesten som $\underline{L}\,\underline{U}$-faktorisering, men $\underline{U} \longrightarrow \underline{D}$ - en diagonal matrise.
- Utfør en Gausseliminering for å redusere matrisen til en øvretriangulær matrise. Utfør de samme operasjonene på $\underline{I}$.
- Utfør de ekstra operasjonene som trengs for å gjøre $\underline{A}$ diagonal (identiteten). Utfør de samme operasjonene på $\underline{I}$. Når dette er oppnådd vil matrisen som resulterer fra operasjonen på $\underline{I}$ være inversen.

Hvordan lage invers på en lurere måte

- Som et eksempel skal vi se på hvordan dette virker på matrisen

$$\left[\begin{array}{ccc|ccc} 3 & 5 & 2 & 1 & & \\ & 8 & 2 & & 1 & \\ & & 6 & & & 1 \end{array}\right]$$

- Vi skalerer hver rad slik at vi får 1 på diagonalelementene;

$$\left[\begin{array}{ccc|ccc} 1 & \frac{5}{3} & \frac{2}{3} & \frac{1}{3} & & \\ & 1 & \frac{1}{4} & & \frac{1}{8} & \\ & & 1 & & & \frac{1}{6} \end{array}\right]$$

Hvordan lage invers på en lurere måte

- Utfører (rad 1) = (rad 1)-$\frac{5}{3}$(rad 2)

$$\left[\begin{array}{ccc|ccc} 1 & & \frac{1}{4} & \frac{1}{3} & \frac{-5}{24} & \\ & 1 & \frac{1}{4} & & \frac{1}{8} & \\ & & 1 & & & \frac{1}{6} \end{array}\right]$$

- Utfører (rad 1) = (rad 1)-$\frac{1}{4}$(rad 3)

$$\left[\begin{array}{ccc|ccc} 1 & & & \frac{1}{3} & \frac{-5}{24} & \frac{-1}{24} \\ & 1 & \frac{1}{4} & & \frac{1}{8} & \\ & & 1 & & & \frac{1}{6} \end{array}\right]$$

Hvordan lage invers på en lurere måte

- Utfører (rad 2) = (rad 2)-$\frac{1}{4}$(rad 3)

$$\left[\begin{array}{ccc|ccc} 1 & & & \frac{1}{3} & \frac{-5}{24} & \frac{-1}{24} \\ & 1 & \frac{1}{4} & & \frac{1}{8} & \\ & & 1 & & & \frac{1}{6} \end{array}\right]$$

- Matrisen til høyre er nå $\underline{A}^{-1}$!
- Åpenbart systematisk
- Kjappere enn den foreslåtte metoden.

Iterative ligningsløsere

- Gausseliminasjon er en direkte metode - finner "rett" svar etter et fikset antall operasjoner. Et antall operasjoner som i mange tilfeller er alt for mange. I tillegg utnytter ikke (generell) Gausseliminasjon strukturen i matrisene.
- Noen ganger trenger vi ikke å løse systemer perfekt, vi kan ha andre feilkilder som dominerer el.l.
- Vi tyr da til iterative ligningsløsere - indirekte metoder.

Fikspunktiterasjoner

- Fristende å teste de iterative skjemaene vi allerede har sett på.
- Fikspunktiterasjon:

$$\underline{f}(\underline{x}) = \underline{0} = \underline{g}(\underline{x}) - \underline{x}$$
$$\underline{A}\,\underline{x} - \underline{b} = \underline{0} = \underline{g}(\underline{x}) - \underline{x} \Rightarrow$$
$$(\underline{A} + \underline{I})\underline{x} - \underline{b} = \underline{x} \Rightarrow \underline{x}^{k+1} = (\underline{A} + \underline{I})\underline{x}^{k} - \underline{b}$$

Dette vil *sjelden* fungere da kravet om at $\underline{g}$ er en kontraksjon er det samme som at

$$\max_i |\lambda_i(\underline{I} + \underline{A})| < 1$$

noe som er et fryktelig strengt krav.

Newtoniterasjoner

- Newtoniterasjoner:

$$\underline{x}^{k+1} = \underline{x}^{k} - \underline{A}^{-1}\left(\underline{A}\,\underline{x}^{k} - \underline{b}\right) = \underline{A}^{-1}\underline{b}\,.$$

  Men dette er jo problemet vi prøver å løse i utgangspunktet!

Ny strategi

- Vi trenger nye iterative metoder.
- Generell framgangsmåte: Splitt matrisen $\underline{A}$.

$$\underline{A} = \underline{D} + \underline{L} + \underline{U}$$

$$\begin{bmatrix} \ddots & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \ddots \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ \vdots & 0 & 0 \\ \vdots & \cdots & 0 \end{bmatrix} + \begin{bmatrix} 0 & \cdots & \vdots \\ 0 & 0 & \vdots \\ 0 & 0 & 0 \end{bmatrix}$$

Viktig: $\underline{L}$ og $\underline{U}$ er *IKKE* de samme matrisene som du har i LU-faktorisering!

Gauss-Jakobi og Gauss-Seidel iterasjoner

- Ser nå på

$$\begin{aligned} \underline{A}\,\underline{x} &= \underline{b} \\ (\underline{D} + \underline{L} + \underline{U})\underline{x} &= \underline{b} \\ \underline{D}\,\underline{x} &= \underline{b} - \underline{L}\,\underline{x} - \underline{U}\,\underline{x} \Rightarrow \\ \underline{x}^{k+1} &= \underline{D}^{-1}\left(\underline{b} - \underline{L}\,\underline{x}^{k} - \underline{U}\,\underline{x}^{k}\right) \end{aligned}$$

  Dette er Gauss-Jakobi-iterasjoner.
- Vi kan velge en annen fremgangsmåte;

$$\begin{aligned} \underline{A}\,\underline{x} &= \underline{b} \\ (\underline{D} + \underline{L} + \underline{U})\underline{x} &= \underline{b} \\ (\underline{D} + \underline{L})\underline{x} &= \underline{b} - \underline{U}\,\underline{x} \Rightarrow \\ \underline{x}^{k+1} &= (\underline{D} + \underline{L})^{-1}\left(\underline{b} - \underline{U}\,\underline{x}^{k}\right) \end{aligned}$$

  Dette er Gauss-Seidel-iterasjoner.

Iterasjonene på komponentform

- Gauss-Jakobi:

$$x_j^{k+1} = \frac{1}{a_{jj}} \left(b_j - \sum_{i=1, i \neq j}^{n} a_{ji} x_i^k \right)$$

- Gauss-Seidel:

$$x_j^{k+1} = \frac{1}{a_{jj}} \left(b_j - \sum_{j=i}^{j-1} a_{ji} x_i^{k+1} - \sum_{i=j+1}^{n} a_{ji} x_i^k \right)$$

- Spørsmål: Hva er den faktiske forskjellen mellom disse to metodene?

Gauss-Jakobi og Gauss-Seidel - eksempel

- Ser på problemet

$$\begin{bmatrix} 7 & -6 \\ -8 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ -4 \end{bmatrix}$$

- Gauss-Jakobi:

$$x_1^{k+1} = \frac{1}{7}\left(3 - a_{12}x_2^k\right) = \frac{6}{7}x_2^k + \frac{3}{7}$$
$$x_2^{k+1} = \frac{1}{9}\left(-4 - a_{21}x_1^k\right) = \frac{8}{9}x_1^k - \frac{4}{9}$$

- Gauss-Seidel:

$$x_1^{k+1} = \frac{1}{7}\left(3 - a_{12}x_2^k\right) = \frac{6}{7}x_2^k + \frac{3}{7}$$
$$x_2^{k+1} = \frac{1}{9}\left(-4 - a_{21}x_1^{k+1}\right) = \frac{8}{9}x_1^{k+1} - \frac{4}{9}$$

Noen observasjoner

- Den faktisk forskjellen ligger i at Gauss-Seidel bruker nye iteratverdier så fort de er tilgjengelig.
- Hvorfor i det hele tatt se på Gauss-Jakobi? - Jo, i parallelle situasjoner vil vi unngå datautveksling under iterasjonene.
- Begge disse iterative metodene (og en haug med andre) kan skrives på formen

$$\begin{aligned} \underline{M}\,\underline{x}^{k+1} &= \underline{N}\,\underline{x}^{k} + b \Rightarrow \\ \underline{x}^{k+1} &= \underline{M}^{-1}\underline{N}\,\underline{x}^{k} + \underline{M}^{-1}\underline{b} \end{aligned}$$

- Gauss-Jakobi har $\underline{M} = \underline{D}\,, \underline{N} = -(\underline{U} + \underline{L}\,)$.
- Gauss-Seidel har $\underline{M} = \underline{D} + \underline{L}\,, \underline{N} = -\underline{U}\,$.

Konvergens av metodene

- Først: Konvergens i denne sammenheng er at følgen

$$\underline{x}^0, \underline{x}^1, \cdots, \underline{x}^k$$

går mot den eksakte løsningen av $\underline{A}\,\underline{x} = \underline{b}$.

- Veldig vanlig måte å måle feilen på er med *residualfeilen*

$$\underline{r} = \underline{b} - \underline{A}\,\underline{x}^k$$

- Vi skal bruke en annen definisjon, nemlig *forskyvningsfeilen*

$$\underline{d}^k = \underline{x}^k - \underline{x}\,.$$

Problemet med konvergens kan nå uttrykkes som: Hva er $\underline{d}^{k+1}$ gitt $\underline{d}^k$?

Konvergens av metodene

- Vi setter inn vårt iterative skjema og får

$$\underline{d}^{k+1} = \underline{x}^{k+1} - \underline{x} = \underline{M}^{-1}\underline{N}\,\underline{x}^{k} + \underline{M}^{-1}\underline{b} - \left(\underline{M}^{-1}\underline{N}\,\underline{x} + \underline{M}^{-1}\underline{b}\right)$$
$$= \underline{M}^{-1}\underline{N}\left(\underline{x}^{k} - \underline{x}\right) = \underline{M}^{-1}\underline{N}\,\underline{d}^{k}$$

- Denne bruker vi rekursivt og finner at

$$\underline{d}^{k} = \left(\underline{M}^{-1}\underline{N}\right)^{k}\underline{d}^{0}.$$

- For konvergens krever vi at "$\underline{M}^{-1}\underline{N} < 1$" i en eller annen form. Vi trenger et mål for størrelsen av matrisen - dette målet er kjent som en *norm*.

Matrisenormer

- Det fins mange forskjellige måter å måle størrelsen av en matrise på.
- De har en rigorøs definisjon som vi hopper over her.
- Frobeniusnormen:

$$\|A\|_F = \sqrt{\sum_i \sum_j a_{ij}^2}$$

- Ener-normen - største kolonnesum

$$\|A\|_1 = \max_i \sum_j |a_{ij}|$$

- Toer-normen - største radsum

$$\|A\|_2 = \max_j \sum_i |a_{ij}|$$

Konvergens av metodene

- To resultater for normer som vi bare bruker her:
  - Over $\mathbb{R}^{N\times N}$ er alle normer ekvivalente. Dvs, for alle norm-par $\|\cdot\|_a, \|\cdot\|_b$ kan vi finne konstanter c_1 og c_2 slik at

$$c_1\|\cdot\|_a \leq \|\cdot\|_b \leq c_2\|\cdot\|_a.$$

  - Spektralradiusen til en matrise $\underline{A}$ er definert som

$$\rho(\underline{A}) = \max_i |\lambda_i|$$

    hvor λ_i er den i'te egenverdien til matrisen $\underline{A}$. Har da resultatet

$$\rho(\underline{A}) = \inf_{\|\cdot\|} \|A\|$$

    dvs spektralradiusen er det minste målet for en matrise vi kan komme opp med.

Konvergens av metodene

- Siden alle normer er ekvivalente vil konvergens i en norm implisere konvergens i alle andre.
- Vi kan se på normen som har minst verdi - altså kan vi se på spektralradiusen til $\underline{A}$.
- Kravet vårt blir at

$$\lim_{k \to \infty} \underline{d}^{\,k} = \underline{0} \Leftrightarrow \rho(\underline{M}^{-1}\underline{N}) < 1$$