14.02: Gauss-eliminasjon og LU-faktorisering

torsdag 13. februar 2020 12:53

Læringsmål:

2.1 Gauss-eliminasjon: Repetisjon og implementering
2.2 LU-faktorisering : Gjenbruk av Gauss-eliminasjon
2.3 Feilkilder
2.4 PA = LU faktorisering: Fjerner en type feilkilde

2.1 Gauss-eliminasjon

Reduserer likningssystemet til et som er enklere

21.02: Iterative løsningsmetoder

torsdag 20. februar 2020 12:41

Læringsmål

Litt om normer

2.5 Iterative metoder: Jacobi og Gauss-Seidel

2.6 Newtons multivariabel metode for løsnings av ikke-lineære systemer, og

Broydens metode

Litt om normer for vektorer og matriser

Normer for vektorer og matriser er en måte å måle størrelse. Det finnes mange forskjellige måter å gjøre dette på. En som er mye brukt er de såkalte p-normene:

$$\|\boldsymbol{x}\|_p = \sqrt[p]{|x_1|^p + \cdots + |x_n|^p}$$

Vi har spesialtilfellene

$$\|\boldsymbol{x}\|_1 = |x_1| + \cdots + |x_n|$$

og

$$\|\boldsymbol{x}\|_\infty = \max_{0 \le i \le n} \{|x_i|\} = \lim_{p \to \infty} \|\boldsymbol{x}\|_p$$

Fra en norm på vektorer, kan vi definere den *induserte* normen på matriser ved å se på dem som lineære operatorer/transformasjoner:

$$\|A\| = \sup_{\boldsymbol{x} \neq 0} \frac{\|A\boldsymbol{x}\|}{\|\boldsymbol{x}\|} = \sup_{\|\boldsymbol{x}\| = 1} \|A\boldsymbol{x}\|$$

hvor $\|\boldsymbol{x}\|$ er en vektor-norm. Operatornormen måler det maksimale faktoren som operatoren «strekker» (eller krymper) lengden til en vektor.

For spesialtilfellene ovenfor får følgende induserte normer:

$$\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{n} |a_{ij}| \quad \|A\|_\infty = \max_{1 \le i \le n} \sum_{j=1}^{n} |a_{ij}|$$

Som er henholdsvis maks kolonnesum og maks radsum.

Eksempel:

$$A = \begin{bmatrix} -3 & 5 & 7 \\ 2 & 6 & 4 \\ 0 & 2 & 8 \end{bmatrix} \quad \|A\|_1 = 19 \quad \|A\|_\infty = 15$$

Hvorfor?

$$Ae_1 = \begin{bmatrix} -3 & 5 & 7 \\ 2 & 6 & 4 \\ 0 & 2 & 8 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 5 \\ 6 \\ 2 \end{bmatrix} \quad Ae_3 = \begin{bmatrix} 7 \\ 4 \\ 8 \end{bmatrix} \quad \|Ae_3\|_1 = 7+4+8 = 19$$

$$\|e_1\|_1 = 1 \quad \|Ae_1\| = |3+5+7|$$

$$\left\|\overset{v}{\begin{bmatrix}1\\1\\1\end{bmatrix}}\right\|_\infty = 1 \qquad \|Av\|_\infty = \left\|\begin{bmatrix}3+5+7\\2+6+4\\0+2+8\end{bmatrix}\right\|_\infty = 15$$

Iterative metoder:

Gjett og regn ut, gjenta med resultatet fra forrige runde. Konvergerer det mot et svar?

Jacobimetoden

Eksempel:

Vi skal løse 3x3 likningssystem. Løs hver likning for hver sin variabel:

$$\begin{aligned} 5x_1 + x_2 + 2x_3 &= 1 \\ x_1 + 4x_2 + x_3 &= 2 \\ 2x_1 + 2x_2 + 5x_3 &= 3 \end{aligned}$$

$$A = \begin{bmatrix} 5 & 1 & 2 \\ 1 & 4 & 1 \\ 2 & 2 & 5 \end{bmatrix}$$ Strengt diagonal-dominant

Løs likning i for variabel x_i:

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

Gjett på løsning, f.eks. $(x_1^{(0)}, x_2^{(0)}, x_3^{(0)}) = \underline{(1,1,1)}$. Sett inn og i likningene ovenfor og regn ut neste iterasjon:

$$x_1^{(1)} = \frac{1 - 1 - 2 \cdot 1}{5} = -\frac{2}{5}$$
$$x_2^{(1)} = \frac{2 - 1 - 1}{4} = 0$$
$$x_3^{(1)} = \frac{3 - 2 \cdot 1 - 2 \cdot 1}{5} = -\frac{1}{5}$$

Så bruker vi disse verdiene og setter inn i samme likninger. Dette gjentas inntil vi enten har konvergert tilstrekkelig mot en verdi (endringene er mindre enn en satt grense), eller at iterasjonen divergerer.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
x_1	1	-0,400	0,280	-0,234	0,061	-0,139	-0,017	-0,097	-0,047	-0,079	-0,059	-0,072	-0,063	-0,069
x_2	1	0,000	0,650	0,240	0,502	0,335	0,441	0,374	0,417	0,389	0,407	0,396	0,403	0,398
x_3	1	-0,200	0,760	0,228	0,598	0,375	0,522	0,430	0,489	0,452	0,476	0,461	0,470	0,464

Her konvergerte metoden mot en løsning, men her er et eksempel hvor det ikke fungerer:

Her fungerte det, men konvergensen er ikke så rask. Eksakte svar er

$$x_1 = -\frac{1}{15} \approx -0{,}067$$
$$x_2 = \frac{2}{5} = 0{,}4$$
$$x_3 = \frac{7}{15} \approx 0{,}467$$

Vi prøver med samme likninger, men endrer på rekkefølgen til følgende:

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

Her går det riktig galt:

0	1	2	3	4	5	6	7	8
1	-2	15	-85,5	334	-2108	9621	-50040,6	258718,3
1	-6	17	-130	591	-2884	15928	-75393	396863,1
1	-3	28	-81	608	-2694	13646	-73329,5	351614,6

Gauss-Seidel-metoden

Som Jacobi, men vi bruker resultatet x_{i+1} med en gang i neste likning, i samme iterasjon:

$$x_1^{(1)} = \frac{1 - 1 - 2 \cdot 1}{5} = -\frac{2}{5}$$
$$x_2^{(1)} = \frac{2 - (-\frac{2}{5}) - 1}{4} = \frac{7}{20}$$
$$x_3^{(1)} = \frac{3 - 2 \cdot (-\frac{2}{5}) - 2 \cdot \frac{7}{20}}{5} = \frac{62}{100}$$

Dette gir raskere konvergens:

0	1	2	3	4	5	6			
1	-0,4	-0,12	-0,074	-0,068	-0,067	-0,067	-0,0667	-0,06667	-0,07
1	0,35	0,375	0,3941	0,399	0,3998	0,4	0,4	0,4	0,4
1	0,62	0,497	0,4719	0,467	0,4668	0,4667	0,4667	0,466667	0,467

Når fungerer Jacobi- og Gauss-Seidel-metodene?

En $n \times n$ matrise $A = (a_{ij})$ kalles ***strengt diagonal-dominant*** hvis absoluttverdien til hvert element på hoved-diagonalen er større enn summen av absoluttverdiene av de øvrige elementene i samme rad.

Teorem: Hvis en kvadratisk matrise A er strengt diagonal-dominant, så er

- A ikke-singulær (dvs. invertibel)
- Jacobi- og Gauss-Seidel konvergerer (alltid!) mot eneste løsning.

NB! Å være strengt diagonal-dominant er ikke en *nødvendig* betingelse, men tilstrekkelig, for konvergens.

Eksempel

I eksemplet ovenfor hadde vi først koeffisientmatrisen på strengt diagonal form, men den var ikke det når vi løste den andre gang.

"Over-relaxation" ikke eksamensrelevant. Det er en metode for «raske på» konvergensen av Gauss-Seidel-metoden.

Glisne matriser («Sparse matrices»)

Matriser hvor de «fleste» elementene er lik null. Det er ofte en fordel å bruke en iterativ løsningsmetode på glisne matriser, fordi det trengs langt færre utregninger. Gauss-eliminasjon på en glissen matrise kan gjøre matrisen full, dvs at den har få elementer som er null. Dette kommer av radoperasjonene.

Forskjellige typer tynne matriser oppstår i forskjellige sammenhenger. For å finne effektive løsninger på store likningssystemer, så er det viktig å finne ut av strukturen på koeffisientmatrisen, og bruke riktig lagring og løsningsmetode.

Matlab har flere måter å definere og lagre forskjellige typer tynne matriser på.

~~Spdiags~~, sparse, spones, spalloc, speye

Newtons metode - for flere variable

Newton i en variabel: Starter med et punkt på grafen («gjetning»). Vi bruker lineariseringen til funksjonen i punktet som en tilnærming til funksjonen, og finner nullpunktet til lineariseringen. Dette gir nytt punkt på grafen, og vi itererer prosessen til vi er fornøyd med tilnærmingen, eller at vi ser at metoden feiler (med gitt startverdi).

Flervariabel linearisering

Her er alle fete bokstaver vektorer. For tre variable $x = (x, y, z)$, og en funksjon

$$f(\boldsymbol{x}) = f(\boldsymbol{x_0}) + \frac{\partial f}{\partial x}(\boldsymbol{x_0})(\cancel{y} - \cancel{y_0}) + \frac{\partial f}{\partial y}(\boldsymbol{x_0})(y - y_0) + \frac{\partial f}{\partial z}(\boldsymbol{x_0})(z - z_0)$$

Dette kan skrives mer kortfattet slik:

$f(\boldsymbol{x}) = f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x_0}) \cdot (\boldsymbol{x} - \boldsymbol{x_0})$

Hvor gradienten $\nabla f(\boldsymbol{x_0}) = \left(\frac{\partial f}{\partial x}(\boldsymbol{x_0}), \frac{\partial f}{\partial y}(\boldsymbol{x_0}), \frac{\partial f}{\partial z}(\boldsymbol{x_0})\right)$

En likning i tre variable vil jo vanligvis ha uendelig mange løsninger (eksempel).

Likningssystemer

$f_1(x, y, z) = 0$
$f_2(x, y, z) = 0$
$f_3(x, y, z) = 0$

Merk: Vi kan alltid flytte over så vi får 0 på høyresiden. Dette er ikke nødvendigvis lineære likninger.

Vi kan skrive dette mer kompakt som $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}$, hvor $\boldsymbol{F} = (f_1, f_2, f_3)$

I et punkt $\boldsymbol{x_0}$ med funksjonsverdi $\boldsymbol{y_0} = \boldsymbol{F}(\boldsymbol{x_0})$ har vi lineariseringer L_1, L_2 og L_3 av f'ene. Det felles nullpunktet for disse er gitt av det lineære likningssystemet

$L_1(x, y, z) = 0$
$L_2(x, y, z) = 0$
$L_3(x, y, z) = 0$

På matriseform blir dette

$$\begin{bmatrix} f_1(\mathbf{x}_0) \\ f_2(\mathbf{x}_0) \\ f_3(\mathbf{x}_0) \end{bmatrix} + \begin{bmatrix} \frac{\partial f_1}{\partial x}(\mathbf{x}_0) & \frac{\partial f_1}{\partial y}(\mathbf{x}_0) & \frac{\partial f_1}{\partial z}(\mathbf{x}_0) \\ \frac{\partial f_2}{\partial x}(\mathbf{x}_0) & \frac{\partial f_2}{\partial y}(\mathbf{x}_0) & \frac{\partial f_2}{\partial z}(\mathbf{x}_0) \\ \frac{\partial f_3}{\partial x}(\mathbf{x}_0) & \frac{\partial f_3}{\partial y}(\mathbf{x}_0) & \frac{\partial f_3}{\partial z}(\mathbf{x}_0) \end{bmatrix} \begin{bmatrix} (x - x_0) \\ (y - y_0) \\ (z - z_0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Som kan skrives på kompakt form

$D\boldsymbol{F}(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0) = -\boldsymbol{F}(\boldsymbol{x_0})$

$D\boldsymbol{F}$ kalles Jakobi-matrisen til $\boldsymbol{F}$:

$$D\boldsymbol{F} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial y} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial y} \end{bmatrix}$$

Eksempel:

La $f_1(x,y,z) = z + x\ln y$, $f_2(x,y,z) = zy + x - 1$, $f_3(x,y,z) = x^2z - y$

$\boldsymbol{F}(x,y,z) = (f_1, f_2, f_3)$ har Jakobi-matrise lik

$$D\boldsymbol{F} = \begin{bmatrix} \ln y & \frac{x}{y} & 1 \\ 1 & z & y \\ 2xz & -1 & x^2 \end{bmatrix}$$

Multivariabel Newtons metode for å løse $\boldsymbol{F} = \boldsymbol{0}$:

$$DF(x_i)\overbrace{(x - x_i)}^{\delta_{i+1}} = \overset{=0}{F(x)} - F(x_i)$$

1. Velg/finn/gjett på startverdi $\boldsymbol{x}_i = \boldsymbol{x}_0$
2. Finn $\boldsymbol{\delta}_{i+1}$ ved å løse det lineære likningssystemet $D\boldsymbol{F}(\boldsymbol{x}_i)\boldsymbol{\delta}_{i+1} = -\boldsymbol{F}(\boldsymbol{x}_i)$.
3. Regn ut $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{\delta}_{i+1}$
4. Når $\|\boldsymbol{\delta}_{i+1}\|_\infty > TOL$, gjenta 2-4 for å regne ut $\boldsymbol{x}_2, \boldsymbol{x}_3$, etc. Om $\|\boldsymbol{\delta}_i\|_\infty$ ikke går mot 0, så feiler metoden.

Eksempel (fortsettelse av forrige eksempel)

$$\boldsymbol{F}(x,y,z) = (z + x\ln y, \quad zy + x - 1,\ x^2z - y), D\boldsymbol{F} = \begin{bmatrix} \ln y & x/y & 1 \\ 1 & z & y \\ 2xz & -1 & x^2 \end{bmatrix}$$

Vi gjetter på $\boldsymbol{x}_0 = (1,1,1)$. Da er $\boldsymbol{F}(1,1,1) = (1,2,0)$ og

$$D\boldsymbol{F} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & y \\ 2 & -1 & 1 \end{bmatrix}$$

Vi løser $D\boldsymbol{F}(1,1,1)\boldsymbol{\delta_1} = -\boldsymbol{F}(1,1,1)$ som gir $\boldsymbol{\delta_1} = (0, -\frac{1}{2}, -\frac{1}{2})$, så $\boldsymbol{x}_1 = (1, \frac{1}{2}, \frac{1}{2})$

I neste trinn er $\boldsymbol{F}\left(1, \frac{1}{2}, \frac{1}{2}\right) = (\frac{1}{2} - \ln 2, \frac{1}{4}, 0) \approx (-0{,}1931, 0{,}5000, 0{,}0000)$, og

$$D\boldsymbol{F}(1, \frac{1}{2}, \frac{1}{2}) = \begin{bmatrix} -\ln 2 & 2 & 1 \\ 1 & 1/2 & 1/2 \\ 1 & -1 & 1 \end{bmatrix} \approx \begin{bmatrix} -0{,}6931 & 2{,}0000 & 1{,}0000 \\ 1{,}0000 & 0{,}5000 & 0{,}5000 \\ 1{,}0000 & -1{,}0000 & 1{,}0000 \end{bmatrix}$$

Løsningen til $D\boldsymbol{F}\boldsymbol{\delta}_2 = -\boldsymbol{F}$ er $\boldsymbol{\delta}_2 = (-0{,}2954, -0{,}1023, 0{,}1930)$, og $x_2 = (0{,}7046, 0{,}3977, 0{,}6930)$

Fortsetter vi iterasjonene så får vi

$\boldsymbol{x}_0 = (1.0000, 1.0000, 1.0000)$
$\boldsymbol{x}_1 = (1.0000, 0.5000, 0.5000)$
$\boldsymbol{x}_2 = (0.7046, 0.3977, 0.6930)$
$\boldsymbol{x}_3 = (0.7315, 0.3783, 0.7089)$
$\boldsymbol{x}_4 = (0.7312, 0.3791, 0.7092)$
$\boldsymbol{x}_5 = (0.7311, 0.3791, 0.7092)$

Konvergerer!

Broydens metoder

Hvis vi ikke har $D\boldsymbol{F}$, eller at den er beregningskrevende å finne, så kan vi bruke Broydens metoder. Disse følger samme iterative prosess som Newtons metode, men vi bruker tilnærminger til DF. Vi starter med en gjetning A_0 for DF, og gjør iterative oppdateringer A_i sammen med x_i.

Hvordan skal A_{i+1} oppdateres fra A_i?

Den lineære tilnærmingen kan skrives

$$DF(x_0)(x - x_0) \approx F(x) - F(x_0)$$

Vi vi ønsker at A_{i+1} skal respektere dette:

$$A_{i+1}(x_{i+1} - x_i) = F(x_{i+1}) - F(x_i)$$

Vi skriver $\delta_{i+1} = x_{i+1} - x_i$ og $\Delta_{i+1} = F(x_{i+1}) - F(x_i)$.

Dette sier bare noe om hva A_{i+1} skal være i en retning/dimensjon. På det ortogonale komplementet så lar vi A_{i+1} være lik A_i, dvs. $A_{i+1}w = A_i w$ for alle w hvor $w \perp \delta_{i+1}$

Da kan det vises at vi får iterasjonsformelen

$$\boxed{A_{i+1} = A_i + \frac{(\Delta_{i+1} - A_i\delta_i)\delta_{i+1}^T}{\delta_{i+1}^T\delta_{i+1}}}$$

Broydens metode 1

1. Velg $x_i = x_0$ og $A_i = A_0$
2. Finn $x_{i+1} = x_i + \delta_{i+1}$ ved å løse
   $A_i\delta_{i+1} = -F(x_i)$
   Og
   $$A_{i+1} = A_i + \frac{(\Delta_{i+1} - A_i\delta_i)\delta_{i+1}^T}{\delta_{i+1}^T\delta_{i+1}}$$
3. Gjenta trinn 2 inntil $\|\delta_i\|$ er tilstrekkelig liten, eller at vi ser at $\|\delta_i\|$ ikke går mot null.

Broydens metode 2

Det tar tid å løse $A_i\delta_{i+1} = -F(x_i)$. Vi kan heller starte med å gjette på DF^{-1}, kall den for B_0, og heller oppdatere denne. Det vi ønsker av B_{i+1} er at

$$\delta_{i+1} = B_{i+1}\Delta_{i+1}.$$

og at den skal være invers til A_{i+1}, så vil vi fortsatt ha $A_{i+1}w = A_i w$ for w i komplementet. Det gir

$$B_{i+1}A_i w = w$$

1. Velg $x_i = x_0$ og $B_i = B_0$
2. Finn
   A_i^{-1}
   $x_{i+1} = x_i - B_i F(x_i)$
   Og

$$B_{i+1} = B_i + \frac{(\delta_{i+1} - B_i \Delta_{i+1})\delta_{i+1}^T B_i}{\delta_{i+1}^T B_i \Delta_{i+1}}$$

3. Gjenta trinn 2 inntil $||\delta_i||$ er tilstrekkelig liten, eller at vi ser at $||\delta_i||$ ikke går mot null.

6.03: Minste kvadraters metode

fredag 6. mars 2020 01:05

forelesning - kap 4...

forelesning - kap 4...

Kurvetilpasning:
Minste kvadraters metode
Lineære og ikke-lineære likninger
QR-faktorisering

Forelesning 6.03.2020 TDAT 2002 – Dag Olav Kjellemo

Stoffet dekkes i læreboka i avsnittene 4.1 – 4.3

Læringsmål

- Kunne anvende minste kvadraters metode til kurvetilpasning av data
- Kjenne og kunne regne på forskjellige modeller
- Kunne QR-faktorisering og kjenne til dens fordeler

Inkonsistente likninger

Det er ikke alltid et system av likninger har løsninger. Det skjer typisk hvis det er flere likninger enn variable:

$$\begin{aligned} x + \ \ y &= 1 \\ x - \ \ y &= 3 \\ x + 2y &= 2 \end{aligned}$$

På matriseform blir dette

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$$

som vi skriver $A\boldsymbol{x} = \boldsymbol{b}$.

Det finnes ingen $\boldsymbol{x} = [x\ y]^T$ som tilfredsstiller likningen. En annen måte å si det på er at $\boldsymbol{b}$ ikke er en lineærkombinasjon av søylevektorene til A:

$$x \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + y \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$$

I stedet kan vi finne $\boldsymbol{x}$ slik at $A\boldsymbol{x}$ er nærmest mulig $\boldsymbol{b}$.

Vi gjør som vi ofte gjør, og skriver likningen som $A\boldsymbol{x} - \boldsymbol{b} = \boldsymbol{0}$. Vi vil finne $\boldsymbol{x} = \overline{\boldsymbol{x}}$ slik at $\|A\overline{\boldsymbol{x}} - \boldsymbol{b}\|_2$ er minst mulig. Dette kalles minste kvadraters metode.

I den alternative formuleringen prøver vi å finne det punktet i planet utspent av søylevektorene i A, som ligger nærmest $\boldsymbol{b}$.

Geometrisk ser vi at $A\overline{\boldsymbol{x}} - \boldsymbol{b}$ må stå normal på alle vektorer $A\boldsymbol{x}$:

$\boldsymbol{x}^T A^T (A\overline{\boldsymbol{x}} - \boldsymbol{b}) = 0$ for alle $\boldsymbol{x}$

Da må $A^T(A\overline{\boldsymbol{x}} - \boldsymbol{b}) = 0$, som vi kan skrive $A^T A\boldsymbol{x} = A^T\boldsymbol{b}$

$A^T A$ er en kvadratisk matrise.

Å finne nærmeste punkt

Vi bruker ligningssystemet ovenfor som eksempel. Då får vi

$$A^T A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}$$

$$A^T b = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix} = \begin{bmatrix} 6 \\ 2 \end{bmatrix}$$

Løser normallikningen $A^T A\bar{x} = A^T b$:

$$\begin{bmatrix} 3 & 2 & 6 \\ 2 & 6 & 2 \end{bmatrix} \quad \begin{bmatrix} 2 & 6 & 2 \\ 3 & 2 & 6 \end{bmatrix} \div 2 \quad \begin{bmatrix} 1 & 3 & 1 \\ 0 & -7 & 3 \end{bmatrix}$$

$\bar{x} = 1 - 3y = 1 + \frac{9}{7} = \frac{16}{7}$

$\bar{y} = -\frac{3}{7}$

Tilbakesetting gir $\bar{y} = -\frac{3}{7}$, $\bar{x} = \frac{16}{7}$, og nærmeste punkt blir

$$A\overline{\boldsymbol{x}} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 16/7 \\ -3/7 \end{bmatrix} = \begin{bmatrix} 13/7 \\ 19/7 \end{bmatrix}$$

$$A\bar{\boldsymbol{x}} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 16/7 \\ -3/7 \end{bmatrix} = \begin{bmatrix} 13/7 \\ 19/7 \\ 10/7 \end{bmatrix}$$

og $\|A\bar{\boldsymbol{x}} - \boldsymbol{b}\|_2 = \sqrt{\left(\frac{6}{7}\right)^2 + \left(-\frac{2}{7}\right)^2 + \left(\frac{-4}{7}\right)^2} = \frac{\sqrt{56}}{7} \approx 1{,}07$

Eksempel: Kurvetilpasning i planet

Vi skal se på forskjellige *modeller*, dvs. typer kurver vi kan ha:

0. Konstant funksjon: $y = c_0$
1. Lineær funksjon (rett linje): $y = c_0 + c_1 x$
2. Kvadratiske modell (andregradspolynom): $y = c_0 + c_1 x + c_2 x^2$
3. Tredjegradspolynom: $y = c_0 + c_1 x + c_2 x^2 + c_3 x^3$

0. Konstant funksjon

Det enkleste tilfellet for minste kvadraters metode er når vi tar gjennomsnittet av en endelig mengde verdier. Vi trenger ikke x-verdier her, de påvirker ikke gjennomsnittet.

Gitt verdier $y_1, y_2, \dots, y_n$, så er gjennomsnittet gitt ved

$$\bar{y} = \frac{y_1 + y_2 + \cdots + y_n}{n}$$

Gjennomsnittet $\bar{y}$ er den verdien som minimerer summen av kvadratavvikene

$$\sigma^2 = (y - y_1)^2 + (y - y_2)^2 + \cdots + (y - y_n)^2$$

Dvs. at jeg kan ikke gjøre uttrykket ovenfor mindre enn det jeg får om jeg setter inn $\bar{y}$ for y.

Begrunnelse: Summen av kvadratavvikene er en funksjon i y (y_i er konstanter). For å finne minste verdi, kan vi derivere med hensyn på y og sette lik null:

$$\begin{aligned} g(y) &= (y - y_1)^2 + (y - y_2)^2 + \cdots + (y - y_n)^2 \\ g'(y) &= 2(y - y_1) + 2(y - y_2) + \cdots + 2(y - y_n) \\ g'(y) = 0 &\Leftrightarrow n \cdot y = y_1 + y_2 + \cdots + y_n \Leftrightarrow y = \frac{y_1 + y_2 + \cdots + y_n}{n} \end{aligned}$$

1. Lineær funksjon

1. Lineær funksjon

Vi vil finne den rette linjen som best tilnærmer et sett data $\{(x_i, y_i)\}_{i=1}^{n}$.

Eksempel

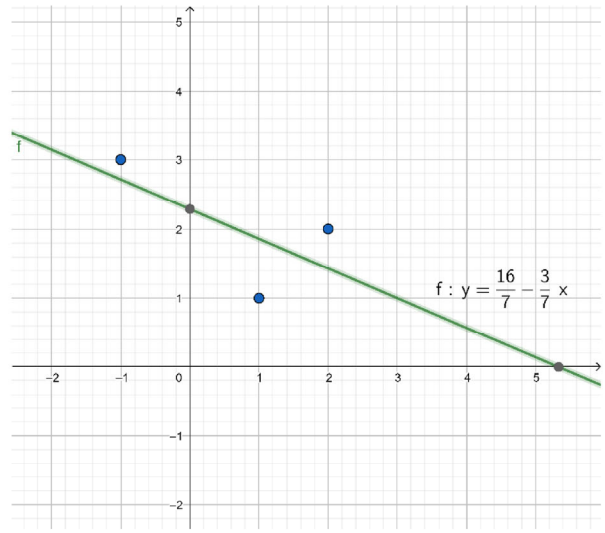
Finn lineær funksjon som best passer med data $(1,1)$, $(-1,3)$ og $(2,2)$.

Setter vi inn i lineær likning $c_0 + c_1 x = y$, så får vi

$$\begin{aligned} c_0 + \ \ c_1 &= 1 \\ c_0 - \ \ c_1 &= 3 \\ c_0 + 2c_1 &= 2 \end{aligned}$$

som på matriseform er

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$$



Dette er det samme systemet som vi hadde i starten. Det har ingen løsninger, men det som minimerer kvadratavviket er

$$\begin{bmatrix} c_0 \\ c_1 \end{bmatrix} = \begin{bmatrix} 16/7 \\ -3/7 \end{bmatrix}$$

Og «beste linje» blir

$$y = \frac{16}{7} - \frac{3}{7}x$$

Feilmål for minste kvadraters metode

$r = b - A\bar{x}$ kalles feilvektoren. Kvadratfeilen er $SE = \|r\|^2$, men et bedre mål å angi feilen på er å bruke gjennomsnittlig kvadratfeil:

$$\text{RMSE} = \sqrt{\frac{SE}{n}}$$

Trinn for kurvetilpasning ved hjelp av minste kvadraters metode

Gitt datapunkter.

Trinn 1: Velg modell med parametre

Trinn 2: Sett inn for datapunktene i modellen. Hvert datapunkt gir en likning, hvor de ukjente er parametrene.

Trinn 2: Sett inn for datapunktene i modellen. Hvert datapunkt gir en likning, hvor de ukjente er parametrene.

Trinn 3: Løs normallikningene $A^T Ax = A^T b$.

Kondisjon for minste kvadrater

Det kan vises at kondisjonstallet til $A^T A$ er omtrent kvadratet av kondisjonstallet til A. Det betyr at det er stor risiko for at normallikningene er dårlig kondisjonert. (Skal snakke mer om kondisjonstall senere.)

Flere modeller

4. Modell for periodiske data: $y = a \sin\left(\frac{2\pi x}{p}\right) + b \cos\left(\frac{2\pi x}{p}\right) + c$

her er a og b, c parametre i modellen.

5. Eksponensiell modell: $y = ae^{bx}$

6. Potensmodell: $y = ax^b$

ikke lineære i parametrene.

NB! For at vi skal kunne finne parametrene (a, b, etc), så må vi få lineære likninger i disse variablene når vi setter inn for x og y. Modellene trenger ikke være lineære i x og y.

Dette får vi for modellen for periodiske data, men ikke for den eksponensielle og potensmodellene.

Eksponensiell modell:

$y = ae^{bx}$ er ikke lineær i a og b. Tar vi logaritmen på begge sider, får vi

$$\ln y = \ln a + bx$$

Når vi setter inn for datapunkter (x, y) så gir dette gir lineære likninger for $\ln a$ og b. Vi finner $\ln a$ og b som før, og regner ut $a = e^{\ln a}$.

Potensmodell

Her bruker vi samme triks som for eksponensiell modell

$$\ln y = \ln a + b \ln x$$

6.03: Del 2

fredag 6. mars 2020 07:48

Modellering

Ulike data passer til ulike modeller. Vår kunskap om dataene og det som er målt kan vi velge ulike modeller.

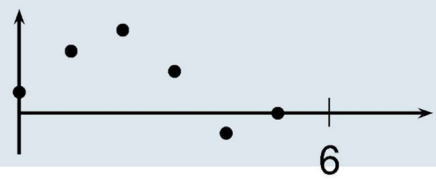
- Lineær modell: $y = ax + b$.
- Kvadratisk modell: $y = ax^2 + bx + c$.
- Modell for periodiske data: $y = a\sin(2\pi x/p) + b\,cos(2\pi x/p) + c$.
- Eksponensiell modell: $y = a\,e^{b\,x}$.
- Potensmodell: $y = a\,x^b$.

Modellen trenger ikke være lineær i x.
Vi ønsker at den skal være lineær i a, b, etc.

Periodiske data

Eksempel

Lag en modell som passer "best mulig" til dataene $(0,1)$, $(1,3)$, $(2,4)$, $(3,2)$, $(4,-1)$, $(5,0)$

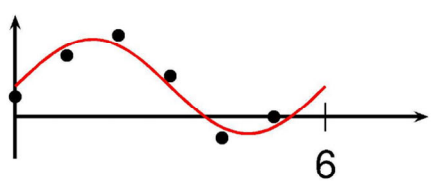


En mulighet er $y = a + b\sin(2\pi x/6)$.

$$
\begin{aligned}
1 &= a + b\sin(2\pi 0/6) = a + b\cdot 0 \\
3 &= a + b\sin(2\pi 1/6) = a + b\cdot(\sqrt{3}/2) \\
4 &= a + b\sin(2\pi 2/6) = a + b\cdot(\sqrt{3}/2) \\
2 &= a + b\sin(2\pi 3/6) = a - b\cdot 0 \\
-1 &= a + b\sin(2\pi 4/6) = a + b\cdot(-\sqrt{3}/2) \\
0 &= a + b\sin(2\pi 5/6) = a + b\cdot(-\sqrt{3}/2)
\end{aligned}
$$

Løsning

Vi setter opp system $A\mathbf{x} = \mathbf{b}$, $A = \begin{bmatrix} 1 & 0 \\ 1 & \sqrt{3}/2 \\ 1 & \sqrt{3}/2 \\ 1 & 0 \\ 1 & -\sqrt{3}/2 \\ 1 & -\sqrt{3}/2 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 1 \\ 3 \\ 4 \\ 2 \\ -1 \\ 0 \end{bmatrix}$. Setter opp normallikningen $A^T A\mathbf{x} = A^T\mathbf{b}$: $A^T A = \begin{bmatrix} 6 & 0 \\ 0 & 3 \end{bmatrix}$, $A^T\mathbf{b} = \begin{bmatrix} 9 \\ 4\sqrt{3} \end{bmatrix}$, løser og får: $a = 3/2$, $b = \frac{4}{3}\sqrt{3}$.



Vi kunne også ha hatt en bedre modell

En annen mulighet er $y = a + b\sin(2\pi x/6) + c\sin(4\pi x/6)$.

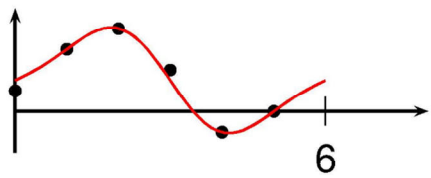
$$
\begin{aligned}
1 &= a + b\cdot 0 + c\cdot 0 \\
3 &= a + b\cdot(\sqrt{3}/2) + c\cdot(\sqrt{3}/2) \\
4 &= a + b\cdot(\sqrt{3}/2) - c\cdot(\sqrt{3}/2) \\
2 &= a - b\cdot 0 + c\cdot \\
-1 &= a + b\cdot(-\sqrt{3}/2) + c\cdot(\sqrt{3}/2) \\
0 &= a + b\cdot(-\sqrt{3}/2) - c\cdot(\sqrt{3}/2)
\end{aligned}
$$

Vi setter opp system $A\mathbf{x} = \mathbf{b}$, $A = \begin{bmatrix} 1 & 0 & 0 \\ 1 & \sqrt{3}/2 & \sqrt{3}/2 \\ 1 & \sqrt{3}/2 & -\sqrt{3}/2 \\ 1 & 0 & 0 \\ 1 & -\sqrt{3}/2 & \sqrt{3}/2 \\ 1 & -\sqrt{3}/2 & -\sqrt{3}/2 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} 1 \\ 3 \\ 4 \\ 2 \\ -1 \\ 0 \end{bmatrix}$.

Fortsettelse

Setter opp normallikningen $A^T A\mathbf{x} = A^T\mathbf{b}$:

$A^T A = \begin{bmatrix} 6 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 3 \end{bmatrix}$, $A^T\mathbf{b} = \begin{bmatrix} 9 \\ 4\sqrt{3} \\ \sqrt{3} \end{bmatrix}$, løser og får: $a = 3/2$, $b = \frac{4}{3}\sqrt{3}$, og $c = \frac{\sqrt{3}}{3}$.



Eksponensiell modell

- $y = a\,e^{bx}$ er ikke lineær i a og b
- Skriver om:

$$\ln y = \ln a + bx.$$

- Denne er lineær i $\ln a$ og b.
- Finner $\ln a$ og b som gir minste kvadratfeil for omformet modell.

$$\ln y = \ln(a\,e^{bx}) = \ln a + \ln(e^{bx})$$
$$= \ln a + bx$$

data $t = 0 \quad 1 \quad 2 \quad 3$

$y = 2 \quad 4 \quad 4 \quad 8$

$$\boxed{a = 2^{\log_2 a}}$$

modell $y = a\,2^{bx}$

$$\log_2 y = \log_2 (a\,2^{bt}) = \log_2 a + b\,t$$

$$\begin{aligned} 1 &= \log_2 a + 0 \\ 2 &= \log_2 a + b \\ 2 &= \log_2 a + 2b \\ 3 &= \log_2 a + 3b \end{aligned}$$

$$\underbrace{\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}}_{A} \begin{bmatrix} \log_2 a \\ b \end{bmatrix} = \underbrace{\begin{bmatrix} 1 \\ 2 \\ 2 \\ 3 \end{bmatrix}}_{b}$$

$$A^T A = \begin{bmatrix} 4 & 6 \\ 6 & 14 \end{bmatrix} \qquad A^T b = \begin{bmatrix} 8 \\ 15 \end{bmatrix} \qquad (A^T A)^{-1} = \frac{1}{20} \begin{bmatrix} 14 & -6 \\ -6 & 4 \end{bmatrix}$$

$$\begin{bmatrix} \log_2 a \\ b \end{bmatrix} = (A^T A)^{-1} \begin{bmatrix} 8 \\ 15 \end{bmatrix} = \cdots$$

Oversikt

Læringsmål

4.1 Minste kvadraters metode

4.2 Ulike modeller

4.3 QR-faktorisering

Gram-Schmidt ortogonalisering

En $m \times n$-matrise A med n L.U. søyler $\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n$.

$$A = \left[\mathbf{a}_1 \,\middle|\, \mathbf{a}_2 \,\middle|\, \cdots \,\middle|\, \mathbf{a}_n\right]$$

Gram-Schmidt prosessen lager en ortonormal basis for $\mathrm{Col}\ A$.

$$\begin{aligned}
\mathbf{q}_1 &= \mathbf{a}_1/|\mathbf{a}_1| \\
\mathbf{w}_2 &= \mathbf{a}_2 - (\mathbf{a}_2^T\mathbf{q}_1)\mathbf{q}_1 \\
\mathbf{q}_2 &= \mathbf{w}_2/|\mathbf{w}_2| \\
\mathbf{w}_3 &= \mathbf{a}_3 - (\mathbf{a}_3^T\mathbf{q}_1)\mathbf{q}_1 - (\mathbf{a}_3^T\mathbf{q}_2)\mathbf{q}_2 \\
\mathbf{q}_3 &= \mathbf{w}_3/|\mathbf{w}_3| \\
\mathbf{w}_4 &= \mathbf{a}_4 - (\mathbf{a}_4^T\mathbf{q}_1)\mathbf{q}_1 - (\mathbf{a}_4^T\mathbf{q}_2)\mathbf{q}_2 - (\mathbf{a}_4^T\mathbf{q}_3)\mathbf{q}_3 \\
\mathbf{q}_4 &= \mathbf{w}_4/|\mathbf{w}_4| \\
&\cdots \\
\mathbf{w}_n &= \mathbf{a}_n - (\mathbf{a}_n^T\mathbf{q}_1)\mathbf{q}_1 - \cdots - (\mathbf{a}_n^T\mathbf{q}_{n-1})\mathbf{q}_{n-1} \\
\mathbf{q}_n &= \mathbf{w}_n/|\mathbf{w}_n|
\end{aligned}$$

$\boxed{q_1^T q_1 = a_1^T a_1 / |a_1|^2 = 1}$ $\quad |x| = \sqrt{x_1^2 + x_2^2 + \cdots + x_m^2}$

$w_2 \cdot q_1 = (a_2^T - (a_2^T q_1) q_1^T) q_1$

$= a_2^T q_1 - a_2^T q_1$

Eksempel Gram Schmidt

Utfør Gram Schmidt på søylene til

$$A = \begin{bmatrix} 1 & 2 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & 2 \\ -1 & -2 & -4 \end{bmatrix}$$

$a_1 = [1\ -1\ 1\ -1]^T$, $\quad q_1 = ?$ $\quad |a_1| = \sqrt{1^2 + (-1)^2 + 1^2 + (-1)^2} = 2$

$q_1 = [\frac{1}{2}\ -\frac{1}{2}\ \frac{1}{2}\ -\frac{1}{2}]^T$

$w_2 = a_2 - (a_2 \cdot q_1) q_1 = ?$ $\quad a_2 = [2\ 1\ -1\ -2]$

$w_2 = a_2 - (a_2 \cdot q_1) q_1 = ?$ $\quad a_2 = [2\ 1\ -1\ -2]$

$a_2 \cdot q_1 = 1.$ $\quad w_2 = [2\ 1\ -1\ -2] - [1/2\ -1/2\ 1/2\ -1/2]$

$= [3/2\ \ 1/2\ \ -1/2\ \ -3/2]$, $\quad |w_2| = \frac{1}{2}\sqrt{3^2 + 1^2 + (-1)^2 + (-3)^2} = \underline{\underline{\sqrt{5}}}$

$q_2 = \frac{1}{\sqrt{5}} [2\ 1\ -1\ -2]$ ....

$a_3 = [1\ 1\ 2\ -4]^T$ $\quad w_3 = a_3 - (a_3 \cdot q_2) q_2 - (a_3 \cdot q_1) q_1$

Stick at $w_3 \cdot q_2 = 0$: $\quad a_3 \cdot q_2 - (a_3 \cdot q_2) \overbrace{q_2 \cdot q_2}^{=1} - (a_3 \cdot q_1) \overbrace{q_1 \cdot q_2}^{=0} = 0$

$q_3 = \frac{w_3}{\|w_3\|}$

Ortogonal matrise

En **ortogonal matrise** er en kvadratisk matrise der **alle søyler** (og rader) er **ortonormale**.
Konsekvenser:

$$Q^T Q = I.$$

$$Q^{-1} = Q^{\boldsymbol{T}}.$$

Eksemepel

$$Q = \begin{bmatrix} 1/2 & 1/2 & -1/2 & 1/2 \\ -1/2 & 1/2 & 1/2 & 1/2 \\ 1/2 & -1/2 & 1/2 & 1/2 \\ -1/2 & -1/2 & -1/2 & 1/2 \end{bmatrix}$$

QR-faktorisering definisjon

QR-faktoriseringen av en $m \times n$-matrise A:

$$A = QR.$$

- Q er ortogonal $m \times m$. Dvs $Q^T Q = I$, $Q^{-1} = Q^T$
- R er øvre triangulær $m \times n$-matrise.

QR-faktorisering

Vi skriver om hele Gram-Schmidt:

$$
\begin{aligned}
\mathbf{a}_1 &= |\mathbf{a}_1|\mathbf{q}_1 \\
\mathbf{a}_2 &= (\mathbf{a}_2^T\mathbf{q}_1)\mathbf{q}_1 + |\mathbf{w}_2|\mathbf{q}_2 \\
\mathbf{a}_3 &= (\mathbf{a}_3^T\mathbf{q}_1)\mathbf{q}_1 + (\mathbf{a}_3^T\mathbf{q}_2)\mathbf{q}_2 + |\mathbf{w}_3|\mathbf{q}_3 \\
&\cdots \\
\mathbf{a}_n &= (\mathbf{a}_n^T\mathbf{q}_1)\mathbf{q}_1 + (\mathbf{a}_n^T\mathbf{q}_2)\mathbf{q}_2 + \cdots + (\mathbf{a}_{n-1}^T\mathbf{q}_{n-1})\mathbf{q}_{n-1} + |\mathbf{w}_n|\mathbf{q}_n
\end{aligned}
$$

Finn $m-n$ nye vektorer $\mathbf{q}_{n+1}, \ldots, \mathbf{q}_m$ slik at $\mathbf{q}_1, \ldots, \mathbf{q}_m$ er ortonormale

QR-faktorisering

Da er

$$A = \left[\mathbf{q}_1 \,\middle|\, \mathbf{q}_2 \,\middle|\, \cdots \,\middle|\, \mathbf{q}_m \right] \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

$$r_{ii} = |\mathbf{w}_i|$$

$$r_{ij} = |\mathbf{q}_i^T \mathbf{a}_j|$$

Eksempel Gram Schmidt

QR-faktoriser

$$A = \begin{bmatrix} 1 & 2 & 1 \\ -1 & 1 & 1 \\ 1 & -1 & 2 \\ -1 & -2 & -4 \end{bmatrix}$$

Matlab =

```
> [Q, R] = qr(A)
```


QR-faktorisering og minste kvadrater

Vi skal bruke QR-faktorisering til minste kvadraters metode.

- I minste kvadrater skal vi minimere

$$\|A\mathbf{x} - \mathbf{b}\|_2$$

- Vi skriver om til

$$\|QR\mathbf{x} - \mathbf{b}\|_2$$

- Det er det samme som å minimere

Fordi

$|Q^T r|^2 =$

$(Q^T r)^T Q^T r =$

$r^T Q Q^T r =$

$r^T I r =$

- Det er det samme som å minimere

$$\|Q^TQRx - Q^Tb\| = \left\|R\mathbf{x} - Q^T\mathbf{b}\right\|_2$$

$r^TQQ^Tr = r^TIr = r^Tr = |r|^2$

For ortogonal matrise Q, så er $\|Qx\|_2 = \|x\|_2$

Feilanalyse

La $\mathbf{d} = Q^T\mathbf{b}$. Lengden til vektoren

$$\begin{bmatrix} e_1 \\ \vdots \\ e_n \\ e_{n+1} \\ \vdots \\ e_m \end{bmatrix} = \overbrace{\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}}^{R} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} - \begin{bmatrix} d_1 \\ \vdots \\ d_n \\ d_{n+1} \\ \vdots \\ d_m \end{bmatrix}$$

er minst når $e_1 = e_2 = \cdots = e_n = 0$. Løser derfor

$$\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ d_n \end{bmatrix}$$

for å finne minste kvadraters løsning av $A\mathbf{x} = \mathbf{b}$.

Hvorfor QR-faktorisering?

- Hvis A er dårlig kondisjonert er $A^T A$ muligens enda dårligere kondisjonert.
- $R\mathbf{x} = Q^T\mathbf{b}$ vil være mye bedre kondisjonert enn $A^T A\mathbf{x} = A^T\mathbf{b}$

Eksempel

Finn minste kvadraters løsning til

$$\begin{array}{rcrcrcr} x & + & 2y & + & z & = & 6 \\ -x & + & y & + & z & = & 4 \\ x & - & y & + & 2z & = & 4 \\ -x & - & 2y & - & 4z & = & -5 \end{array}$$

13.03: QR-faktorisering, Newtons flervariabel metode

torsdag 12. mars 2020 10:40

Mer om QR-faktorisering

For en $m \times n$ matrise A, hvor $m \geq n$, så definerte vi QR-faktoriseringen som en faktorisering $A = QR$

$$\overbrace{[\boldsymbol{a}_1 \quad \boldsymbol{a}_2 \quad \dots \quad \boldsymbol{a}_n]}^{A} = \overbrace{[\boldsymbol{q}_1 \quad \boldsymbol{q}_2 \quad \dots \quad \boldsymbol{q}_n]}^{Q} \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ 0 & r_{22} & \dots & r_{2n} \\ 0 & 0 & \dots & r_{3n} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & r_{nn} \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

hvor Q var ortogonal, dvs. $Q^T Q = I_{nn}$, som også betyr at kolonnevektorene $\boldsymbol{q}_i$ danner en ortonormal basis: $\boldsymbol{q}_i^T \boldsymbol{q}_i = 1$ og $\boldsymbol{q}_i^T \boldsymbol{q}_j = 0$ for $i \neq j$.

Søylevektorene $\boldsymbol{a}_i$ trenger ikke være lineært uavhengige.

Ganger vi ut matrisen på høyresiden, ser vi at

$$\begin{aligned} \boldsymbol{a}_1 &= r_{11}\boldsymbol{q}_1 \\ \boldsymbol{a}_2 &= r_{12}\boldsymbol{q}_1 + r_{22}\boldsymbol{q}_2 \\ &\vdots \\ \boldsymbol{a}_n &= r_{1n}\boldsymbol{q}_1 + r_{2n}\boldsymbol{q}_2 + \cdots r_{nn}\boldsymbol{q}_n \end{aligned}$$

Vi ser at R er øvretriangulær, er det samme som at for hver $k = 1, \dots, n$, så må rommet utspent av $\boldsymbol{a}_1, \dots, \boldsymbol{a}_k$ være inneholdt i rommet utspent av $\boldsymbol{q}_1, \dots \boldsymbol{q}_k$.

Faktoriseringen er ikke entydig generelt, men hvis alle søylene til A er lineært uavhengige, og vi velger diagonalelementene r_{kk} til R å være positive, f.eks., så er faktoriseringen entydig.

Sist brukte vi Gram-Schmidt-metoden til å konstruere QR-faktoriseringen. Den er ganske lett å forstå geometrisk, lett å implementere, men er ikke den mest numerisk stabile metoden.

Householder reflektorer

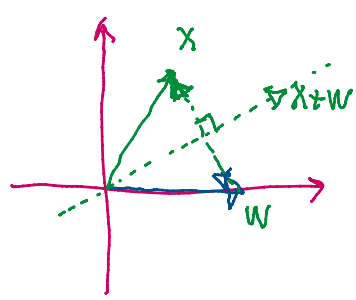
- Mer numerisk stabil enn Gram-Schmidt for å konstruere QR-faktorisering.
- Baserer seg på refleksjoner om et hyperplan

Lemma: Hvis vektorene u og v har samme lengde, dvs. $\|x\|_2 = \|w\|_2$ så er $x - w$ og $x + w$ ortogonale vektorer.

Bevis:

$$\begin{aligned} &(w - x)^T (w + x) = w^T x + w^T w - x^T x - x^T w \\ &= (w^T w - x^T x) + w^T x - x^T w \\ &= \|w\|_2 - \|x\|_2 + 0 \\ &= 0 \end{aligned}$$

$(x^T w)^T = w^T x$



For vektoren $\boldsymbol{v} = \boldsymbol{w} - \boldsymbol{x}$ definerer vi projeksjonsmatrisen

$$P = \frac{\boldsymbol{v}\boldsymbol{v}^T}{\boldsymbol{v}^T\boldsymbol{v}}$$

P som transformasjon definerer projeksjonen på v, og har følgende egenskaper:

$P^T = P, P^2 = P$ og $P\boldsymbol{v} = \boldsymbol{v}$, $P\boldsymbol{u} = 0$ dersom $\boldsymbol{v}^T\boldsymbol{u} = 0$, dvs. u står normalt på $\boldsymbol{v}$.

Vi definerer nå *Householder reflektoren* som

P som transformasjon definerer projeksjonen på v, og har følgende egenskaper:

$P^T = P, P^2 = P$ og $P\boldsymbol{v} = \boldsymbol{v}$, $P\boldsymbol{u} = 0$ dersom $\boldsymbol{v}^T\boldsymbol{u} = 0$, dvs. u står normalt på $\boldsymbol{v}$.

Vi definerer nå *Householder reflektoren* som

$$H = I - 2P$$

~~og ser at H re~~flekterer $\boldsymbol{x}$ til $\boldsymbol{w}$. (Vis ved regning!)

$Hx = \ldots$

H er symmetrisk og ortogonal:

$$H^T H = (I-2P)^T(I-2P) = I^2 - 2P - 2P + 4\overset{P}{P^2} = I$$

Eksempel 4.17. La $\boldsymbol{x} = [3\ 4]^T$, $\boldsymbol{w} = [5, 0]^T$. Finn $\boldsymbol{H}$ slik at $\boldsymbol{Hx} = \boldsymbol{w}$.

$$v = \begin{bmatrix} 2 \\ -4 \end{bmatrix} \qquad P = \frac{vv^T}{v^Tv} = \frac{1}{20}\begin{bmatrix} 4 & -8 \\ -8 & 16 \end{bmatrix} = \begin{bmatrix} 0{,}2 & -0{,}4 \\ -0{,}4 & 0{,}8 \end{bmatrix}$$

$$H = I - 2P = \begin{bmatrix} 1-0{,}4 & 0{,}8 \\ 0{,}8 & -0{,}6 \end{bmatrix} = \begin{bmatrix} 0{,}6 & 0{,}8 \\ 0{,}8 & -0{,}6 \end{bmatrix} \qquad Hx = W$$

Vi skal bruke Householder-refleksjonene til å konstruere en QR-faktorisering. Vi gjøres ved å transformere søylevektorene til koordinatvektorer, slik at vi få laget en øvretriangulær matrise.

Eksempel: Bruk Householder-reflektorer til å finne QR-faktorisering til matrisen

$$A = \begin{bmatrix} 1 & -1 & 4 \\ 1 & 4 & -2 \\ 1 & 4 & 2 \\ 1 & -1 & 0 \end{bmatrix}.$$

$A \quad 4\times 3$

Løsning: Vi skal finne H_1, H_2, H_3 slik at $H_3H_2H_1A = R$. Da blir $A = H_1H_2H_3R = QR$.

$H_3H_2H_1A = R$

$A = \underbrace{H_1H_2H_3}_{Q}R$

$$x_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \text{ og } \|x_1\| = 2 \quad w_1 = \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad v_1 = w_1 - x_1 = \begin{bmatrix} -3 \\ -1 \\ -1 \\ -1 \end{bmatrix},$$

$v = w - x$

$$H_1 = I - \frac{2}{v^Tv}vv^T = -\frac{1}{6}\begin{bmatrix} 3 & 3 & 3 & 3 \\ 3 & -5 & 1 & 1 \\ 3 & 1 & -5 & 1 \\ 3 & 1 & 1 & -5 \end{bmatrix}, \quad A_1 = H_1A = \begin{bmatrix} -2 & -3 & -2 \\ 0 & \frac{10}{3} & -4 \\ 0 & \frac{10}{3} & 0 \\ 0 & -\frac{5}{3} & -2 \end{bmatrix}$$

$H_1 = I - 2P$

I neste trinn så jobber vi med 3×2 matrise.

$$x_2 = \frac{1}{3}\begin{bmatrix} 10 \\ 10 \\ -5 \end{bmatrix} \text{ og } \|x_2\| = 5, w_2 = \begin{bmatrix} -5 \\ 0 \\ 0 \end{bmatrix}, v_2 = w_2 - x_2 = \begin{bmatrix} -\frac{25}{3} \\ -\frac{10}{3} \\ \frac{5}{3} \end{bmatrix}$$

$$H_2 = \begin{bmatrix} 1 & 0 \\ 0 & \widehat{H_2} \end{bmatrix}, \widehat{H_2} = I - \frac{2}{v^Tv}vv^T = \begin{bmatrix} -\frac{2}{3} & -\frac{2}{3} & \frac{1}{3} \\ -\frac{2}{3} & \frac{11}{15} & \frac{2}{15} \\ \frac{1}{3} & \frac{2}{15} & \frac{14}{15} \end{bmatrix}, A_2 = \begin{bmatrix} -2 & -3 & -2 \\ 0 & -5 & 2 \\ 0 & 0 & -\frac{12}{5} \\ 0 & 0 & -\frac{16}{5} \end{bmatrix}$$

$H_2 = \left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & \hat{H}_2 \end{array}\right]$ Hvis $\hat{H}_2$ ortogonal, så er H_2 ortogonal

$A_2 = H_2A_1 = H_2H_1A$

$$x_3 = \frac{1}{5}\begin{bmatrix} 12 \\ -16 \end{bmatrix} \text{ og } \|x_3\| = 4, w_3 = \begin{bmatrix} -4 \\ 0 \end{bmatrix}, v_3 = w_3 - x_3 = \frac{1}{5}\begin{bmatrix} -32 \\ 16 \end{bmatrix}.$$

$$\widehat{H_3} = I - \frac{2}{v^Tv}vv^T = \begin{bmatrix} -\frac{3}{5} & \frac{4}{5} \\ \frac{4}{5} & \frac{3}{5} \end{bmatrix}, A_3 = \begin{bmatrix} -2 & -3 & -2 \\ 0 & -5 & 2 \\ 0 & 0 & -4 \\ 0 & 0 & 0 \end{bmatrix}$$

$A_3 = H_3A_2 \qquad H_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \boxed{\hat{H}_3} \end{bmatrix}$

$$Q = H_1H_2H_3 = \frac{1}{2}\begin{bmatrix} -1 & 1 & -1 & -1 \\ -1 & -1 & 1 & -1 \\ -1 & -1 & -1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix}$$

Matlab-kode:

```
function [Q,R] = houseHolder(A)
    Ai = A;
    [m,n] = size(Ai);
    Q = eye(m);
    for i = 1:n
        w = Ai(:,1);
        x = zeros(m-i+1,1);
        x(1) = -sign(w(1))*sqrt(w'*w);
        v = w - x;
        P = (v*v') ./ (v'*v);
        H = eye(m-i+1) - 2*P;
        Q(:, i:end) = Q(:, i:end) * H;
        Ai = H * Ai;
        Ai = Ai(2:end,2:end);
    end
    R = Q'*A;
end
```

Ikke-lineære minste kvadraters metode: Gauss-Newton

Vi ser nå på ikke-lineære likningssystem

$$
\begin{aligned}
&r_1(x_1, x_2, \dots, x_n) = 0\\
&r_2(x_1, x_2, \dots, x_n) = 0\\
&\vdots\\
&r_m(x_1, x_2, \dots, x_n) = 0
\end{aligned}
$$

m likninger
n ukjente
$m \geq n$

Vi skriver $\boldsymbol{r} = [r_1, \dots, r_m]^T$ og $\boldsymbol{x} = [x_1, \dots, x_n]^T$.

Minste kvadraters løsning er den hvor følgende feil er minst

$$E(\boldsymbol{x}) = \frac{1}{2}(r_1(\boldsymbol{x})^2 + \cdots r(\boldsymbol{x})^2) = \frac{1}{2}\boldsymbol{F}^T\boldsymbol{F}$$

Minimere

For å finne minste feil, finner vi hvor gradienten er lik null, tilsvarende det vi gjør for funksjoner av en variabel.

Linearisering i flere variable

Illustrerer dette for enkelhets skyld med to variable:

$$
\begin{aligned}
&r_1(x, y) = r_1(x_0, y_0) + \frac{\partial r_1}{\partial x}(x_0, y_0)(x - x_0) + \frac{\partial r_1}{\partial y}(x_0, y_0)(y - y_0) + O(\square)\\
&r_2(x, y) = r_2(x_0, y_0) + \frac{\partial r_2}{\partial x}(x_0, y_0)(x - x_0) + \frac{\partial r_2}{\partial y}(x_0, y_0)(y - y_0) + O(\square)\\
&\vdots
\end{aligned}
$$

2

2

linearisering i flere variable
grafene er plan i $\mathbb{R}^3$ for hver r_i
Partiellderiverte i alle variablene.

Vi kan skrive dette på matriseform som

$$\boldsymbol{r}(\boldsymbol{x}) = \boldsymbol{r}(x_0) + D\boldsymbol{r}(\boldsymbol{x_0})[\boldsymbol{x} - \boldsymbol{x_0}]^{\square} + O(2)$$

Her er $D\boldsymbol{r}(x_0)$ Jacobi-matrisen, som generelt har følgende form:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad r = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_m \end{bmatrix} \qquad Dr.$$

$$D\boldsymbol{r} = \begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \frac{\partial r_1}{\partial x_2} & \cdots & \frac{\partial r_1}{\partial x_n} \\ \frac{\partial r_2}{\partial x_1} & \frac{\partial r_2}{\partial x_2} & \cdots & \frac{\partial r_2}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial r_m}{\partial x_1} & \frac{\partial r_m}{\partial x_2} & \cdots & \frac{\partial r_m}{\partial x_n} \end{bmatrix}$$

← svarer til den deriverte, r' i en variabel.
"stigningstall"

Denne skrives også på komprimert form som

$$\left[\frac{\partial r_i}{\partial x_j}\right] \text{ og } [Dr_{ij}]$$

Tilbake til minste kvadraters feil til likningssystemet $\boldsymbol{r}(x) = 0$

Den minste kvadratiske feilen er følgende størrelse er minst:

$$E = \frac{1}{2}\|r(x)\|^2 = \frac{1}{2} r(x)^T r(x)$$

For en slik minimumsverdi så må gradienten være null:

$$\nabla E = (Dr)^T r = 0$$

Minimere eller maksimere
Finner hvor gradienten er lik 0 som vektor.

produktregel

Vi skal bruke Newtons multivariable metode for å løse dette systemet.

deriverte i alle koordinatretninger er 0.

Vi skriver $F = (Dr)^T r$, og finner lineære tilnærmingen

$$F(x) \approx F(x_0) + DF(x_0)[x - x_0]$$

Analogt til $f(x) \approx f(x_0) + f'(x_0)(x - x_0)$

Produktregel: $\frac{1}{2}\nabla(r^T r)$
$\frac{1}{2}(Dr^T \cdot r + r^T Dr)$
$\frac{1}{2}(2(Dr)^T \cdot r)$
$= (Dr)^T \cdot r$

Jakobi-matrisen DF til F har koeffisienter

$$\frac{\partial f_i}{\partial x_j} = \sum_{k=1}^{m}\left(\frac{\partial r_k}{\partial x_j}\cdot\frac{\partial r_k}{\partial x_i} - r_k\frac{\partial r_k^2}{\partial x_I \partial x_j}\right)$$

Vi forenkler ved å noen av disse leddene, ved å anta at enten r_k eller $\frac{\partial r_k^2}{\partial x_I \partial x_j}$ er små.

Da er $DF \approx (Dr)^T Dr$. Vi setter $A = Dr(x_0)$ og får

$$F(x_0) = A^T r(x_0), DF(x_0) \approx A^T A$$

Å bruke denne tilnærmingen gir en variant av Newtons metode

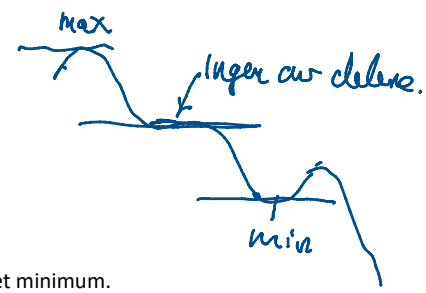
$$F(x) \approx A^T r(x_0) + A^T A[x - x_0]$$

Gitt en første gjetning x_0, så finner vi x_1 ved å løse for $s = x - x_0$ i likningen

$$A^T r(x_0) + A^T A s = 0$$

og sette $x_1 = x_0 + s$

Som før, så itereres dette, og vi observerer om det går mot en vektor eller ikke.



NB!! Selv om metoden konvergerer, så trenger ikke dette være en løsning: Det trenger ikke være et minimum.

Eksempel: Punkt i planet nærmest tre gitte sirkler

Gitt sirkler med radii 3,2 og 2, og sentre i (0,0), (4,0) og (3,4).

Finn punktet der summen av kvadratet av avstanden til sirklene er minst mulig.

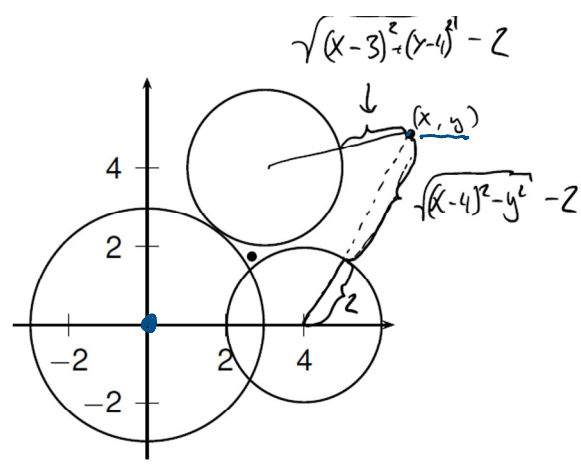
$$r_1(x, y) = \sqrt{x^2 + y^2} - 3$$

avstand til origo minus radius

$$r_2(x,y) = \sqrt{(x-4)^2 + y^2} - 2$$

$$r_3(x,y) = \sqrt{(x-3)^2 + (y-4)^2} - 2$$

$$D\vec{r}(x,y) = \begin{bmatrix} \frac{2x}{2\sqrt{x^2+y^2}} & \frac{y}{\sqrt{x^2+y^2}} \\ \dots & \dots \\ \dots & \dots \end{bmatrix}$$



Matlab-kode:

```
function [P] = GaussNewton(r, Dr, P, TOL)
    while(1)
        A = Dr(P);
        s = - (A'*A)\(A'*r(P));
        P = P + s';
        if (max(abs(s)) < TOL)
            break
        end
```

```
    end
end

C = [[0 0]; [4 0]; [3 4]] %Sirkelsentre
R = [3; 2; 2] %Sirkelradi

S = @(P) sqrt(sum((C-P).^2,2)) % Summer langs andre dim
r = @(P) S(P)-R
Dr = @(P) (P-C)./S(P)

P = [0 1]
GaussNewton(r, Dr, P, .5e-6)
```

Bruk av Gauss-Newton til å finne parametre i ikke-lineære modeller

Gitt datapunkter $(x_i, y_i), i = 1,2,3$. Vi ønsker å bestemme parametre c_1 og c_2 slik at modellen

$$y = c_1 e^{c_2 x}$$

minimerer (minste sum av kvadratene av) feilene (residuene)

$$r_i = c_1 e^{c_2 x_i} - y_i$$

Her er ikke uttrykkene lineære i parametrene c_1 og c_2.

Gitt f.eks. tre datapunkter, så har vi

$$r = \begin{bmatrix} c_1 e^{c_2 x_1} - y_1 \\ c_1 e^{c_2 x_2} - y_2 \\ c_1 e^{c_2 x_3} - y_3 \end{bmatrix}, \qquad Dr = \begin{bmatrix} e^{c_2 x_1} & c_1 x_1 e^{c_2 x_1} \\ e^{c_2 x_2} & c_1 x_2 e^{c_2 x_2} \\ e^{c_2 x_3} & c_1 x_3 e^{c_2 x_3} \end{bmatrix}$$

Matlab-kode for å finne parametrene ovenfor:

```
x = [1 2 3]';
y = [2 3 5]';

r = @(c) c(1)*exp(c(2)*x)-y;
Dr = @(c) [exp(c(2)*x)  c(1)*exp(c(2)*x).*x];

c = GaussNewton(r, Dr, [1 0], .5e-6)

f = @(t) c(1)*exp(c(2)*t)
t = -1:.01:4;

plot(t,f(t))

hold on
plot(x,y,'o')
hold off
```

20.03: Litt teoribakgrunn for prosjektet

fredag 20. mars 2020 08:02

Litt bakgrunn for prosjektet

Formler for numerisk derivert, med og uten randbetingelser

Taylors formel

Hvis f er tilstrekkelig mange gagner deriverbar på et intervall, så har vi

$$f(x+h) = f(x) + f'(x)h + \cdots + \frac{f^{(n)}(x)}{n!} + R_n(x)$$

For små h er h^{n+1} veldig liten

Restleddet er gitt ved

$$R_n(x) = \frac{f^{(n+1)}(c)}{(n+1)!}h^{n+1} = O(h^{n+1})$$

Hvor c ligger mellom x og $x+h$.

Vi kan verifisere hvor godt (med hvor høy orden) en differensformel tilnærmer en derivert:

Foroverformel/høyretilnærmingen for den deriverte er av første orden:

$$\frac{f(x+h)-f(x)}{h} = \frac{\cancel{f(x)} + f'(x)h + O(h^2) - \cancel{f(x)}}{h} = f'(x) + \frac{O(h^2)}{h} = f'(x) + O(h)$$

Oppgave: Vis at sentraltilnærmingen er av andre orden:

$$f'(x) = \frac{f(x+h)-f(x-h)}{2h} + O(h^2)$$

Oppgave: Vis at

$$f''(x) = \frac{f(x+h)-2f(x)+f(x-h)}{h^2} + O(h^2)$$

På randen (endepunkt) kan vi bare bruke funksjonsverdier i en retning.

Hvis $f'(0)=0$ $f''(2)=0$.

Taylors formel for deriverte av f

Hvis f er tilstrekkelig mange ganger deriverbar, så er f', f'' så har vi Taylorformler for disse også:

$$f'(x+h) = f'(x) + f''(x)h + \cdots + \frac{f^{(n+1)}(x)}{n!}h^n + O(h^{n+1})$$

$$f^{(k)}(x+h) = f^{(k)}(x) + f^{(k+1)}(x)h + \cdots + \frac{f^{(n+k)}(x)}{n!}h^n + O(h^{n+1})$$

Foroverfeil, bakoverfeil, forstørringsfaktor og kondisjonstall

Hvordan disse begrepene er definert, avhenger av hvilket problem vi løser.

Det er flere feiltyper i beregninger

- Avrundingsfeil
- Usikkerhet i input (feil i tidligere beregninger, målusikkerhet, …)
- Trunkering (f.eks. ved bruk av Taylorpolynom av en gitt grad)

Presisjon og nøyaktighet

Nøyaktighet handler om feilen, ~~nøyaktighet~~ presisjon om antall sifre vi regner med.

Foroverfeil og bakoverfeil

(Se også side 45 i læreboka)

Vi skal regne

ut input

$$y = f(x)$$

$$\frac{\|y-\hat{y}\|}{\|y\|}$$

Hvis implementasjonen $\hat{f}$ regner $\hat{y}$ så er foroverfeilen $|y - \hat{y}|$, og relativ feil…

Men i stedet for å fokusere på foroverfeilen, så kan vi se på hvilket x-verdi vi har løst problemet. Den minste $|\Delta x|$ som er slik at

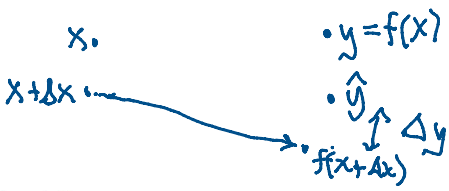
$$\hat{y} = f(x + \Delta x)$$

kalles bakoverfeilen. Hvis bakoverfeilen er mindre enn feil/usikkerhet i input data, så kan vi ikke egentlig ønske noe mer.

En variant av dette er en blandet forover-bakoverfeil er å se på

$$\hat{y} + \Delta y = f(x + \Delta x)$$

Vi vet ikke y nøyaktig



Kondisjonstall

Kondisjonstallet til en beregning måler hvor sensitivt svaret er for endringer i input og for avrundinger. Vi definerte det som største feilforstørringsfaktor, relativ foroverfeil delt på relativ bakoverfeil.

For å måle disse, trenger vi også å måle feilene. Da kan vi bruke forskjellige normer. (Vi antar at vi jobber med vektorer og matriser.)

Normer: $\| \ \|_2 = \sqrt{x_1^2 + \dots + x_n^2}$

$\| \ \|_\infty = \max_{i=1,n} |x_i|$

$\| \ \|_1 = |x_1| + \dots + |x_n|$

Vi definerte

matrise

$$\|A\| = \max \frac{\|Ax\|}{\|x\|}$$

$$= \max_{\|x\|=1} \|Ax\|$$

som største «strekkfaktor». Vi trenger også å se på minste strekkfaktor,

$y = Ax \quad x = A^{-1}y$

$$\min \frac{\|Ax\|}{\|x\|} = \min \frac{\|y\|}{\|A^{-1}y\|} = 1/\max \frac{\|A^{-1}y\|}{\|y\|} = 1/\|A^{-1}\|$$

$\frac{1}{\frac{a}{b}} = \frac{b}{a}$

Vi har definert kondisjonstallet til en matrise

$$\kappa(A) = \|A\| \cdot \|A^{-1}\|$$

men en matrise har egentlig flere kondisjonstall: Det avhenger hvilken utregning vi gjør med matrisen. $\kappa(A)$ er relevant når vi skal løse likningssystemet

$$Ax = b$$

hvor A er en $n \times n$ matrise. For å se dette kan vi se på hvordan endringer i x påvirker endringer i b:

$$A(x + \Delta x) = b + \Delta b$$

$\cancel{Ax} + A\Delta x = \cancel{b} + \Delta b$

Dette gir $A\Delta x = \Delta b$. Fra operatornormen får vi at

$$\|b\| \leq \|A\| \cdot \|x\| \text{ og } \|\Delta b\| \geq \|A^{-1}\|\|\Delta x\|$$

Dette gir

forover bakover

$$\frac{\|\Delta x\|}{\|x\|} \leq \kappa(A) \frac{\|\Delta b\|}{\|b\|}$$

$K(A) = \|A\| \cdot \|A^{-1}\|$

$\frac{\text{forover}}{\text{bakover}} \leq K(A)$

Spørsmål: Hvilke deler i uttrykket ovenfor er relative forover- og bakoverfeil her?

Hva med kondisjonstallet for å regne ut $Ax = y$? Her tenker vi oss at matrisen A er fiksert, og vi vil se på hvordan verdien y endrer seg med endring (feil) i x. Relativ foroverfeil delt på relativ bakoverfeil blir her

x input y output

Δx bakover

Δy foroverfeil

$A(x+\Delta x) - A(x) = A \cdot \Delta x$

$$\sup_{\Delta x} \frac{\frac{\|A(\Delta x)\|}{\|Ax\|}}{\frac{\|\Delta x\|}{\|x\|}} = \sup_{\Delta x} \frac{\|A(\Delta x)\|}{\|\Delta x\|} \cdot \frac{\|x\|}{\|Ax\|} = \|A\| \cdot \frac{\|x\|}{\|Ax\|} \leq \|A\| \cdot \|A^{-1}\| = \kappa(A)$$

max

Kondisjonstallet gir en grense også her for hvor stor feilforstørringsfaktoren er.

$\frac{f}{\sum f}[1 \ldots 1]$ ← eksakt

$\cdot \frac{1}{h^4} A \ldots$ regne ut i Matlab

For

En matrise kan være dårlig kondisjonert (stort kondisjonstall) for noen problemer, men velkondisjonert for andre. Og vice-versa.

20.03: Egenverdier

torsdag 19. mars 2020 22:07

NTNU
Kunnskap for en bedre verden

Egenverdier og egenvektorere
Iterative metoder

Plan

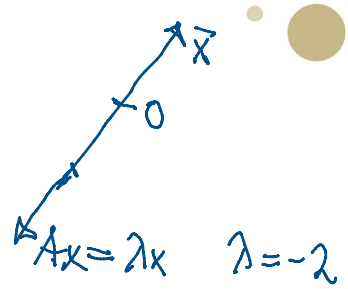
Læremål

12.1 Egenvektorer

12.2 QR-algoritmen

Egenverdier og egen vektorer

- Matrise A Kvadratisk
- $A\mathbf{x} = \lambda\mathbf{x}$ λ skalar
- $\mathbf{x}$ kalles **egenvektor** til A
- λ kalles **egenverdi** til A



Eksempel:

$$A = \begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix}$$

har egenverdier 5 og 1 med tilhørende egen vektorer $[1\ 1]^T$ og $[-3\ 1]^T$.

Svarer til $\begin{bmatrix} 5 & 0 \\ 0 & 1 \end{bmatrix}$

$$\begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \end{bmatrix} = 5\begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \lambda = 5$$

$$\begin{bmatrix} \quad & \quad \end{bmatrix}\begin{bmatrix} -3 \\ 1 \end{bmatrix} = \begin{bmatrix} -3 \\ 1 \end{bmatrix} \qquad \lambda = 1$$

$$\begin{bmatrix} & \\ & \end{bmatrix}\begin{bmatrix} -3 \\ 1 \end{bmatrix} = \begin{bmatrix} -3 \\ 1 \end{bmatrix} \qquad \lambda = 1$$

Tilbakeblikk på karakteristisk polynom

1. Egenverdiene
$$\lambda_1, \lambda_2, \ldots, \lambda_k$$
er løsningene til karakteristisk likning
$$\det(A - \lambda I) = 0$$
2. Finner tilhørende egenvektorer ved å løse
$$\begin{aligned} (A - \lambda_1 I)\mathbf{x} &= \mathbf{0}, \\ (A - \lambda_2 I)\mathbf{x} &= \mathbf{0}, \\ &\vdots \\ (A - \lambda_k I)\mathbf{x} &= \mathbf{0}. \end{aligned}$$

$$Ax = \lambda x$$
$$Ax - \lambda x = 0$$
$$Ax - \lambda I x = 0$$
$$\underbrace{(A - \lambda I)}_{n \times n} x = 0$$

Matrise som byr på problemer

$$A = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 2 & 0 & \cdots & 0 \\ 0 & 0 & 3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 20 \end{bmatrix}$$

$$\det(A - \lambda I) = \det \begin{bmatrix} 1-\lambda & 0 & & \\ 0 & 2-\lambda & & \\ & & \ddots & \\ & & & 20-\lambda \end{bmatrix} = 0$$

$$(1-\lambda)(2-\lambda)\cdots(20-\lambda) = 0$$

$$\lambda = 1 \quad \text{or} \quad \lambda = 2 \quad \text{or} \quad \cdots \quad \text{or} \quad \lambda = 20$$

$$(1-\lambda)(2-\lambda)\cdots(20-\lambda)=0$$
$$\lambda=1 \vee \lambda=2 \vee \cdots \vee \lambda=20$$

Problem med karakteristisk polynom, (Wilkinson)

Eksempel: Koden

```
function [ maxfeil ] = charpolSolveError(n)
    d=(1:n)'; % Diagonal
    A=diag(d); % Diagonalmatrise
    P=charpoly(A); % Finner kar. polynom
    lambda=roots(P); % Finner egenverdiene
    maxfeil = max(abs(sort(lambda)-sort(d))); % feilen
end
```

Gir svaret:

```
>> charpolSolveError(20)
ans =
    0.1196
```

Potens-iterasjonsmetoder

La $A = \begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix}$. Velg vilkårlig vektor $[-2\ 2]^T$

x_0

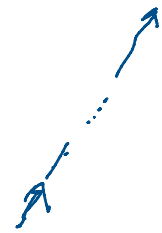
$$
\begin{aligned}
A[-2\ 2]^T &= [2\ 6]^T \\
A[2\ 6]^T &= [22\ 26]^T \\
A[22\ 26]^T &= [122\ 126]^T \\
A[122\ 126]^T &= [622\ 626]^T \\
A[622\ 626]^T &= [3122\ 3126]^T
\end{aligned}
$$

$A^2 x_0$

$A^3 x_0$

$\vdots$

$A^6 x_0$



Ser litt nærmere på forige eksempel

Egenvektorene danner basis

La $A = \begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix}$. Velg vilkårlig vektor $[-2\ 2]^T = [-3\ 1]^T + [1\ 1]^T$

$$
\begin{aligned}
A[-2\ 2]^T &= [-3\ 1]^T + 5[1\ 1]^T \\
A[2\ 6]^T &= [-3\ 1]^T + 5^2[1\ 1]^T \\
A[22\ 26]^T &= [-3\ 1]^T + 5^3[1\ 1]^T \\
A[122\ 126]^T &= [-3\ 1]^T + 5^4[1\ 1]^T \\
A[622\ 626]^T &= [-3\ 1]^T + 5^5[1\ 1]^T
\end{aligned}
$$

Vi ser at $[1\ 1]^T$ dominerer over $[-3\ 1]^T$
Egenverdien med størst absoluttverdi kalles for **dominant**.

Algoritme, Potensiterasjon. (Finn dominant egenverdi)

Gitt startvektor x_0. Vi vil finne egenverdien med høyest absolutt verdi
For $i = 0, 1, 2, 3, \ldots$
Konvergerer lineært.

1. Normaliser: $u_i = x_i / \|x_i\|_2$
2. Løser: $x_{i+1} = Au_i$.

itterer

Egenverdi: $\lambda = u_i^T x_{i+1}$ Egenvektor: $u_i = x_{i+1} / \|x_{i+1}\|_2$

$\|Au_i\| = u_i^T Au_i$

Lineær konvergens.

Algoritme, Invers potensiterasjon.

Hvis A har egenverdier $\lambda_1, \ldots, \lambda_k$
Så har A^{-1} egenverdier $\lambda_1^{-1}, \ldots, \lambda_k^{-1} = \frac{1}{\lambda_k}$

$Ax = \lambda x$
$A^{-1}Ax = A^{-1}\lambda x$
$\frac{1}{\lambda}x = A^{-1}x$

$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_k$
$\lambda_k^{-1} \geq \lambda_{k-1}^{-1} \geq \ldots \geq \lambda_1^{-1}$
(ta absoluttverdier)

Gitt startvektor x_0. Vi vil finne egenverdien nærmest s
For $i = 0, 1, 2, 3, \ldots$
Konvergerer lineært.

1. Normaliser: $u_i = x_i / \|x_i\|_2$
2. Løser: $(A - sI)x_{i+1} = u_i$.

må løse likningssystem

Egenverdi: $\lambda = 1/u_i^T x_{i+1} + s$ Egenvektor: $u_i = x_i / \|x_i\|_2$

$(A - sI)x = \lambda x$
$Ax - sx = \lambda x$
$Ax = (s + \lambda)x$

Algoritme, Rayleighbrøk iterasjon.

Gitt startvektor x_0.
For $i = 0, 1, 2, 3, \ldots$

1. Normaliser: $u_i = x_i / \|x_i\|_2$
2. $\lambda_i = u_i^T A u_i$
3. Løser: $(A - \lambda_i I) x_{i+1} = u_i$.

Egenvektor: $u_i = x_i / \|x_i\|_2$
Konvergerer svært fort!
Kubisk konvergens når A er symmetrisk.
Kvadratisk konvergens ellers.

QR-faktorisering

Repetisjon fra forelesning 6.
Vi skriver om A til et produkt $A = QR$ der

1. Q er ortogonal ($Q^T Q = I$.)
2. R er øvre triangulær

Hvordan: Gram Schmidt eller ved hjelp av matlab;
`[Q,R]=qr(A);` Householder reflektorer

Gram-Schmidt

Vi vil finne en ortogonal basis for $col(A)$.

$$
\begin{aligned}
v_1 &= a_1 \\
u_1 &= v_1/\|v_1\|_2 \\
v_2 &= a_2 - (u_1^T a_2)u_1 \\
u_2 &= v_2/\|v_2\|_2 \\
v_3 &= a_3 - (u_1^T a_3)u_1 - (u_2^T a_3)u_2 \\
u_3 &= v_3/\|v_3\|_2 \\
v_4 &= a_4 - (u_1^T a_4)u_1 - (u_2^T a_4)u_2 - (u_3^T a_4)u_3 \\
u_4 &= v_4/\|v_4\|_2 \\
&\vdots \\
Q &= \begin{bmatrix} u_1 & u_2 & \dots & u_n \end{bmatrix} \\
R &= Q^T A
\end{aligned}
$$

Normalisert Simultan iterasjon

Vi begynner med $Q_0 = I$ og A
Vi regner så ut

$$
\begin{aligned}
Q_1 R_1 &= AQ_0 \\
Q_2 R_2 &= AQ_1 \\
Q_3 R_3 &= AQ_2 \\
Q_4 R_4 &= AQ_3
\end{aligned}
$$

Anta at Q_n konvergerer. Dvs:

$$Q_n \approx Q_{n-1} \quad .$$

for stor nok n. Da er

$$R_n \approx Q_n^{-1} A Q_n \quad =$$

skifte av koordinater

R_n og A har identiske egenverdier.

Hva er egenverdiene til triangulær matrise?

→ diagonalelementene.

QR-algoritme

Metode for å finne alle egenverdier og egenvektorer.
Begynn med $A_0 = A$ og fortsett som følgende for k=0,1,2,3,...

- QR-faktoriser $A_k = Q_k R_k$
- Regn ut $A_{k+1} = R_k Q_k = Q_k^T Q_k R_k Q_k = Q_k^T A_k Q_k$

Alle A_k-ene har identiske egenverdier.
Mest sannsylig konvergerer A_k mot en øvre triangulær matrise.

27.03: Fouriertransform og interpolasjon, 1

torsdag 26. mars 2020 16:19

Forelesning-fourier1.pdf

En seeverdig youtube-kanal er 3Blue1Brown, som blant annet har en visuell forklaring på hva en Fouriertransformasjon er.

But what is the Fourier Transform? A visual introduction.

NTNU

Det skapende universitet

Matematiske metoder 2 Data

Diskret fouriertransformasjon

Dag Olav Kjellemo

IDI, AIT

24. mars 2020

Plan for forelesningen

Læremål

10.1 Fourier Transform

10.2 Trigonometrisk interpolasjon

Læremål

- Kjenne til og kunne regne ut **diskret Forurier-transform** manuelt og på datamaskin.
- Kjenne til **Fast Fourier Transform** (FFT) og dens fortrinn til DFT
- Kjenne til begrepene sampling og undersampling.
- Kjenne til det trigonometriske interpolasjonsteoremet.
- Kunne interpolere ved hjelp av FFT.

Oversikt

Læremål

10.1 Fourier Transform

10.2 Trigonometrisk interpolasjon

NTNU
Det skapende universitet

Interpolasjon og approksimasjon

Interpolasjon er å finne funksjon som går stemmer med gitte verdier,

$$y_i = f(x_i)$$

Approksimasjon er å finne funksjoner som ikke stemmer med gitte verdier, men er en godtilnærming. En måte å måle dette på, som vi har sett på før, er å minimere summen av kvadratavvikene

minimere $$\sum (y_i - f(x_i))^2$$

I begge tilfellene kan vi ønske at funksjonen f skal ha andre ønskede egenskaper.
Her skal vi se hvordan vi kan lage både interpolerende og approksimerende funksjoner ved å legge sammen trigonometriske funksjoner.

Fourier transformasjon

Hva: Presentere data ved hjelp av sinus og cosinus. sum av funksjoner

Data: Tabell, Signal eller funksjoner.

$f(x) = \sum a_k \cos kx + \sum b_k \sin kx$

k bestemmer frekvens/periode.

$a_k, b_k \in \mathbb{R}$, koeffisienter

Hvordan: Ved hjelp av *komplekse tall*

Komplekse tall

Et **kompleks tall** z består av 2 reelle tall (a, b)

- Den **reelle** delen a
- Den **imaginære** delen b
- Vi skriver $z = a + ib$, der i har egenskapen $i^2 = -1$.

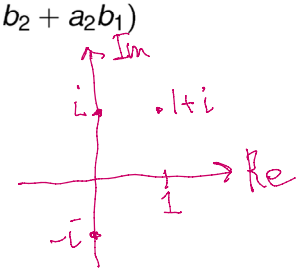
$i = \sqrt{-1}$

$(a_1 + ib_1) \cdot (a_2 + ib_2)$

$a_1 a_2 - b_1 b_2 + i(b_1 a_2 + a_1 b_2)$

Addisjon:	$z_1 + z_2$	$=$	$(a_1 + a_2) + i(b_1 + b_2)$
Multiplikasjon:	$z_1 z_2$	$=$	$(a_1 a_2 - b_1 b_2) + i(a_1 b_2 + a_2 b_1)$
Konjugert:	$\bar{z}$	$=$	$a - ib$
Lengde:	$\lvert z \rvert$	$=$	$\sqrt{a^2 + b^2}$

$\mathbb{C} \ni a + ib \longleftrightarrow \begin{bmatrix} a \\ b \end{bmatrix} \in \mathbb{R}^2$

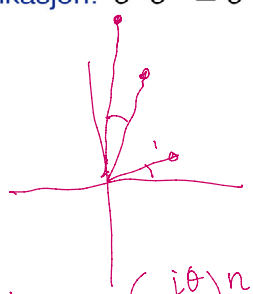


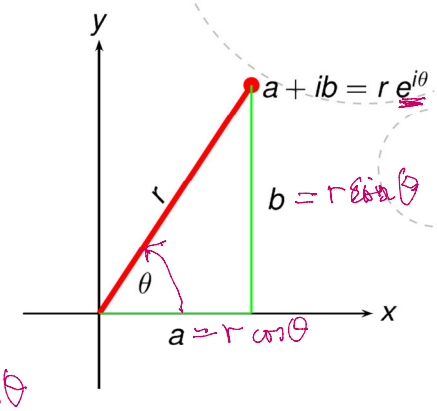
Komplekse tall på polar form

Polar form: $z = r(\cos\theta + i\sin\theta)$

Eulers formel: $e^{i\theta} = \cos\theta + i\sin\theta$

Multiplikasjon: $e^{i\theta}e^{i\gamma} = e^{i(\theta+\gamma)}$





Potenser $(e^{i\theta})^n = e^{in\theta}$

$(e^{x+iy})^n = e^{nx+iny} = e^{nx} \cdot e^{iny}$

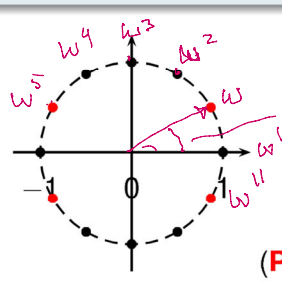
Røttene til enheten

Definisjon ($n$te-rot av enheten)

Hvis $z^n = 1$ kalles z for en **n-te-rot av enheten**.

Definisjon (Primitiv)

Hvis $z^n = 1$ men $z^k \neq 1$ for $k = 1, 2, \ldots n-1$ kalles z for en **primitiv** n-te-rot av enheten.



$\frac{2\pi}{12} = \frac{\pi}{6} = 30°$

$\omega = e^{i\cdot\frac{\pi}{6}}$

$\omega^2 = e^{i\frac{\pi}{3}}$

$\omega^3 = e^{i\frac{\pi}{2}} = i$

(**Primitive**) 12te-røtter av enheten.

For reelle tall

$x^4 = 1$

$x = \pm 1$

$x^{12} = 1$

$\omega^{12} = 1$

$\Rightarrow \omega$ er en 12-te rot av 1.

$\omega = \sqrt[12]{1}$.

$\omega^3 = i$ er 12-rot, men ikke primitiv.

ω^k er primitiv dersom $\gcd(k, 12) = 1$.

dvs. relativt primiske.

Egenskap til enhetsrøtter

$k = 1$

Egenskap til enhetsrøtter

Teorem (Summen av enhetsrøtter)

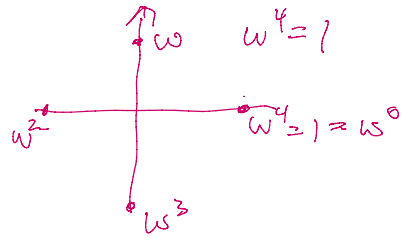
Hvis ω er en primitiv nte enhetsrot og k er et heltall, så er

heltall

$$\sum_{j=0}^{n-1} \omega^{jk} = \begin{cases} n & \text{hvis } k/n \text{ er et heltall} \\ 0 & \text{ellers} \end{cases}$$

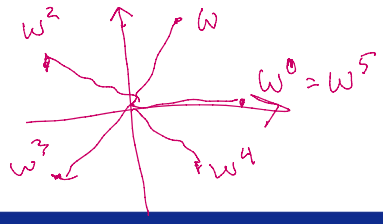
$k = t \cdot n$

$k=1$

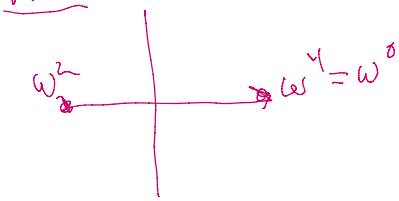


$$\sum_{j=0}^{3} \omega^j = 0$$

$n=5$



$k=2$



$$(1+\omega+\omega^2+\cdots+\omega^{n-1})(1-\omega)$$
$$= 1 + \cancel{\omega+\omega^2+\cdots+\omega^{n-1}} - \cancel{\omega-\omega^2-\cdots}-\omega^n$$
$$= 1-\omega^n = 1-1 = 0$$

Diskret Fourier Transform (DFT)

Anvendelser er ofte med tid, t_i som "x-verdi".

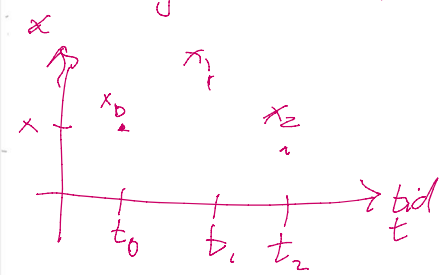
Definisjon

DFT av vektoren $\mathbf{x} = [x_0, x_1, \ldots, x_{n-1}]^T$ er vektoren $\mathbf{y} = [y_0, y_1, \ldots, y_{n-1}]^T$ definert ved

$$y_k = \frac{1}{\sqrt{n}} \sum_{j=0}^{n-1} x_j \omega^{jk},$$

der $\omega = e^{-i2\pi/n} = \cos\frac{-2\pi}{n} + i\sin\frac{-2\pi}{n}$. ω er prim. n'te rot.

"x er y-verdiene".



Diskret Fourier Transform (DFT)

På matriseform har vi

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_{n-1} \end{bmatrix} = \frac{1}{\sqrt{n}} \begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \cdots & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \cdots & \omega^{n-1} \\ \omega^0 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ \omega^0 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \end{bmatrix}$$

Matriseform

Vi kan skrive DFT på formen

$$\mathbf{y} = F_n \mathbf{x}$$

der $F_n = \frac{1}{\sqrt{n}} \left[\omega^{(i-1)(j-1)}\right] =$

$$\frac{1}{\sqrt{n}} \begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \omega^0 & \cdots & \omega^0 \\ \omega^0 & \omega^1 & \omega^2 & \omega^3 & \cdots & \omega^{n-1} \\ \omega^0 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(n-1)} \\ \omega^0 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ \omega^0 & \omega^{n-1} & \omega^{2(n-1)} & \omega^3(n-1) & \cdots & \omega^{(n-1)(n-1)} \end{bmatrix}$$

Eksempel

$\omega = -1$

Eksempel

$\omega = -1$

$$F_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. = \begin{bmatrix} \omega^0 & \omega^0 \\ \omega^0 & \omega^1 \end{bmatrix}$$

$$F_3 = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & -\frac{1}{2} - i\frac{\sqrt{3}}{2} & -\frac{1}{2} + i\frac{\sqrt{3}}{2} \\ 1 & -\frac{1}{2} + i\frac{\sqrt{3}}{2} & -\frac{1}{2} - i\frac{\sqrt{3}}{2} \end{bmatrix}.$$

$$F_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}.$$

$\omega = -i$

Eksempel

Eksempel

Vi vil finne DFT av $\mathbf{x} = \begin{bmatrix} 1 & 0 & 2 & 1 \end{bmatrix}^T \in \mathbb{R}^4$.

Svar $\omega = e^{-i\pi/2} = -i$. $DFT(\mathbf{x}) =$

$$F_4\mathbf{x} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 2 \\ 1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 4 \\ -1+i \\ 2 \\ -1-i \end{bmatrix} = \begin{bmatrix} 2 \\ -1/2 + i/2 \\ 1 \\ -1/2 - i/2 \end{bmatrix}$$

Eksempel

I MATLAB bruker vi funksjonen `fft()`

```
>> fft([1 0 2 1])/2

ans =

   2.0000+0.0000i  -0.5000+0.5000i  1.0000+0.0000i  -0.5000
```


Invers DFT, (IDFT)

Den inverse til F_n er $F_n^{-1} = \bar{F}_n = \frac{1}{\sqrt{n}}\left[\omega^{-(i-1)(j-1)}\right] =$

$$\frac{1}{\sqrt{n}}\begin{bmatrix} \omega^0 & \omega^0 & \omega^0 & \omega^0 & \cdots & \omega^0 \\ \omega^0 & \omega^{-1} & \omega^{-2} & \omega^{-3} & \cdots & \omega^{-(n-1)} \\ \omega^0 & \omega^{-2} & \omega^{-4} & \omega^{-6} & \cdots & \omega^{-2(n-1)} \\ \omega^0 & \omega^{-3} & \omega^{-6} & \omega^{-9} & \cdots & \omega^{-3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ \omega^0 & \omega^{-(n-1)} & \omega^{2(n-1)} & \omega^3(n-1) & \cdots & \omega^{-(n-1)(n-1)} \end{bmatrix}$$

F_n symmetrisk

$F_n^{-1} = \bar{F}_n^T = \bar{F}_n$

$\rightarrow F_n$ unitær.

$F_n \bar{F}_n^T =$ Identitetsmatrise

svarer til ortogonale matriser

Bevis:

$$(F_n \bar{F}_n)_{ij} = \left(\frac{1}{\sqrt{n}}\right)^2 \sum_{k=0}^{n-1} \overset{F_n}{\omega^{(i-1)k}} \overset{\bar{F}_n}{\omega^{-k(j-1)}} = \frac{1}{n}\sum_{k=0}^{n-1} \omega^{(i-1)k}\omega^{-k(j-1)}$$

$$= \frac{1}{n}\sum_{k=0}^{n-1} \omega^{(i-j)k} = \begin{cases} 1 & i = j \\ 0 & i \neq j. \end{cases} \quad (1)$$

Eksempel

$$F_2 = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} = F_2^{-1}.$$

$$F_3 = \frac{1}{\sqrt{3}}\begin{bmatrix}1 & 1 & 1\\ 1 & -\frac{1}{2} - i\frac{\sqrt{3}}{2} & -\frac{1}{2} + i\frac{\sqrt{3}}{2}\\ 1 & -\frac{1}{2} + i\frac{\sqrt{3}}{2} & -\frac{1}{2} - i\frac{\sqrt{3}}{2}\end{bmatrix}.$$

$$F_3^{-1} = \frac{1}{\sqrt{3}}\begin{bmatrix}1 & 1 & 1\\ 1 & -\frac{1}{2} + i\frac{\sqrt{3}}{2} & -\frac{1}{2} - i\frac{\sqrt{3}}{2}\\ 1 & -\frac{1}{2} - i\frac{\sqrt{3}}{2} & -\frac{1}{2} + i\frac{\sqrt{3}}{2}\end{bmatrix}.$$

$$F_4 = \frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & -i & -1 & i\\ 1 & -1 & 1 & -1\\ 1 & i & -1 & -i\end{bmatrix}, \quad F_4^{-1} = \frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & i & -1 & -i\\ 1 & -1 & 1 & -1\\ 1 & -i & -1 & i\end{bmatrix}.$$

Eksempel IDFT

Eksempel

Vi vil finne IDFT av
$\mathbf{y} = \begin{bmatrix}2 & (-1/2 + i/2) & 1 & (-1/2 - i/2)\end{bmatrix}^T \in \mathbb{R}^4$.

Svar $\omega = e^{-i\pi/2} = -i$. $IDFT(\mathbf{y}) =$

$$F_4^{-1}\mathbf{y} = \frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & i & -1 & -i\\ 1 & -1 & 1 & -1\\ 1 & -i & -1 & i\end{bmatrix}\frac{1}{2}\begin{bmatrix}4\\ -1+i\\ 2\\ -1-i\end{bmatrix} = \begin{bmatrix}1\\ 0\\ 2\\ 1\end{bmatrix} =$$

Eksempel IDFT

I MATLAB bruker vi funksjonen `ifft()`

```
>> ifft([2 -0.5+0.5i 1 -0.5-0.5i])*2

ans =

    1 0 2 1
```


Kompleksitet av DFT

DFT bruker

- $2n^2$ multiplikasjoner
- $2n^2 - 2n$ addisjoner

Dvs ca $4n^2$ aritmetiske operasjoner.

FFT

FFT

Legg merke til følgende $n = 8 = 2^3$

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega & -i & -i\omega & -1 & -1\omega & i & i\omega \\ 1 & \omega^2 & -1 & -1\omega^2 & 1 & \omega^2 & -1 & -1\omega^2 \\ 1 & \omega^3 & i & i\omega^3 & -1 & -1\omega^3 & -i & -i\omega^3 \\ 1 & \omega^4 & 1 & 1\omega^4 & 1 & 1\omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & -i & -i\omega^5 & -1 & -1\omega^5 & i & i\omega^5 \\ 1 & \omega^6 & -1 & -1\omega^6 & 1 & \omega^6 & -1 & -1\omega^6 \\ 1 & \omega^7 & i & i\omega^7 & -1 & -1\omega^7 & -i & -i\omega^7 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$$

FFT

Vi kan skrive

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \overbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}}^{F_4} \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \omega & 0 & 0 \\ 0 & 0 & \omega^2 & 0 \\ 0 & 0 & 0 & \omega^3 \end{bmatrix} \overbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}}^{F_4} \begin{bmatrix} x_1 \\ x_3 \\ x_5 \\ x_7 \end{bmatrix}$$

og

$$\begin{bmatrix} y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \overbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}}^{F_4} \begin{bmatrix} x_0 \\ x_2 \\ x_4 \\ x_6 \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \omega & 0 & 0 \\ 0 & 0 & \omega^2 & 0 \\ 0 & 0 & 0 & \omega^3 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \begin{bmatrix} x_1 \\ x_3 \\ x_5 \\ x_7 \end{bmatrix}$$

FFT algoritme,

Vi skriver $F_n = \frac{1}{\sqrt{n}} M_n$. For $n = 2^r$ regner vi ut $M_n \mathbf{x}$ rekursivt:

$$\mathbf{u} = M_k \begin{bmatrix} x_0 & x_2 & \cdots x_{2k-2} \end{bmatrix}^T$$

$$\mathbf{v} = M_k \begin{bmatrix} x_1 & x_3 & x_5 & \cdots x_{2k-1} \end{bmatrix}^T$$

$$\mathbf{u} = M_k \begin{bmatrix} x_0 & x_2 & \cdots x_{2k-2} \end{bmatrix}^T$$

$$\mathbf{v} = M_k \begin{bmatrix} x_1 & x_3 & x_5 & \cdots x_{2k-1} \end{bmatrix}^T$$

$$\mathbf{y} = M_{2k}\mathbf{x} = \begin{bmatrix} u_0 + v_0 \\ u_1 + \omega v_1 \\ u_2 + \omega^2 v_2 \\ \vdots \\ u_{k-1} + \omega^{k-1} v_2 \\ u_0 - v_0 \\ u_1 - \omega v_1 \\ u_2 - \omega^2 v_2 \\ \vdots \\ u_{k-1} - \omega^{k-1} v_2 \end{bmatrix}$$

$O(n \log n)$ vs. $O(n^2)$ ~~for~~

Oversikt

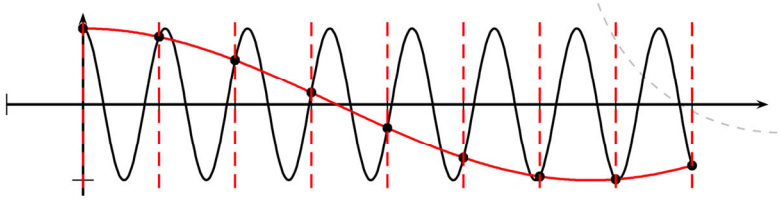
Læremål

10.1 Fourier Transform

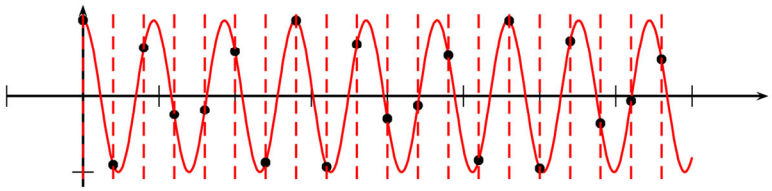
10.2 Trigonometrisk interpolasjon

NTNU
Det skapende universitet

Undersampling



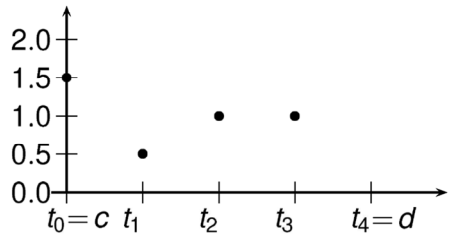
Vi trenger samplingsfrevens som er minst 2 ganger største signalfrekvens



DFT - Interpolering (naiv)

Vi har data som vi ønsker å beskrive som en sum av trigonometriske funksjoner. DFT og FFT viser seg å være godt egnet. Under er en tabell og et diagram som viser dataene grafisk.

t_n	x_n
0.0	1.5
0.4	0.5
0.8	1.0
1.2	1.0



DFT på dataene gir:

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix} \begin{bmatrix} 1.5 \\ 0.5 \\ 1.0 \\ 1.0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0.25 + 0.25i \\ 0.5 \\ 0.25 - 0.25i \end{bmatrix}$$

DFT - Interpolering (naiv)

DFT - Interpolering (naiv)

Vi vil bruke DFT til å interpolere med cosinus og sinus-funksjoner
Tiden går fra $t_0 = c$ til $t_n = d$. Vi har n tidsintervaller á

$$\Delta t = (d - c)/n.$$

Det gir

$$t_j = c + j\Delta t.$$

Vi løser for j og får $j = \frac{t_j - c}{\Delta t}$.

$$j = n\frac{t_j - c}{d - c}.$$

$e^{i\theta} = \cos\theta + i\sin\theta$
$= (a+ib)(\cos\theta + i\sin\theta)$
$= \boxed{(a\cos\theta - b\sin\theta)} + i(b\cos\theta + a\sin\theta)$

DFT - Interpolering (naiv)

$y_k = a_k + ib_k$

Realdelen av invers DFT er

$$x_j = \frac{1}{\sqrt{n}}\sum_{k=0}^{n-1} e^{2kj\pi i/n} y_k = \frac{1}{\sqrt{n}}\sum_{k=0}^{n-1} a_k \cos\frac{2kj\pi}{n} - \frac{1}{\sqrt{n}}\sum_{k=0}^{n-1} b_k \sin\frac{2kj\pi}{n}.$$

Vi setter inn

$$j = \frac{n(t_j - c)}{d - c}. \tag{2}$$

t

$$x(t) = \frac{1}{\sqrt{n}}\sum_{k=0}^{n-1} a_k \cos\frac{2k\pi(t-c)}{d-c} - \frac{1}{\sqrt{n}}\sum_{k=0}^{n-1} b_k \sin\frac{2k\pi(t-c)}{d-c}.$$

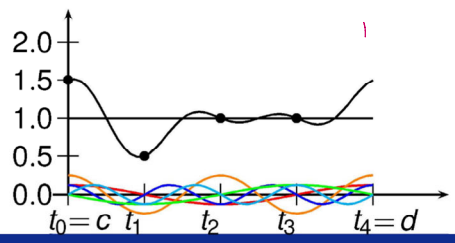
Vi har da en formel for kontinuerlige verdier t.

DFT - Interpolering (naiv)

I vårt tilfelle blir dette

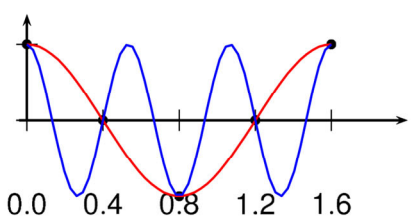
$$x(t) = 1 + 0.125\cos\frac{2\pi t}{1.6} + 0.25\cos\frac{4\pi t}{1.6} + 0.125\cos\frac{6\pi t}{1.6} - 0.125\sin\frac{2\pi t}{1.6} + 0.125\sin\frac{6\pi t}{1.6}.$$

Vi plotter inn i diagrammet og ser at funksjonen interpolerer dataene. Vi har også plottet hver komponent



DFT - Interpolering og undersampling

$\cos\frac{2\pi t}{1.6}$ og $\cos\frac{6\pi t}{1.6}$ samples likt når $n = 4$

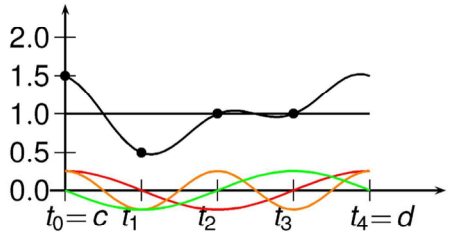


Vi kan derfor erstatte $\cos\frac{6\pi t}{1.6}$ med $\cos\frac{2\pi t}{1.6}$ og på samme måte $\sin\frac{6\pi t}{1.6}$ med $-\sin\frac{2\pi t}{1.6}$. Vi har da

$$x(t) = 1 + 0.25\cos\frac{2\pi t}{1.6} + 0.25\cos\frac{4\pi t}{1.6} - 0.25\sin\frac{2\pi t}{1.6}$$

DFT - Interpolasjonsteorem

Vi plotter inn i diagrammet og ser at funksjonen interpolerer dataene. Vi har også plottet hver komponent



Fordelen er at vi trenger færre ledd og at vi ungår svigninger med kortere periode enn to samplinger.

Trigonometrisk interpolasjonsteorem

Gitt samplingsdata $x_0, x_1, x_2, \ldots, x_{n-1}$ ved tidene $t_0, t_1, \ldots, t_{n-1}$ der $t_0 = c$ og $t_n = d$. Når n er jevn vil funksjonen

$$P_n(t) = \frac{a_0}{\sqrt{n}} + \frac{2}{\sqrt{n}} \sum_{k=1}^{n/2-1} \left(a_k \cos \frac{2k\pi(t-c)}{d-c} - b_k \sin \frac{2k\pi(t-c)}{d-c} \right) + \frac{a_{n/2}}{\sqrt{n}} \cos \frac{n\pi(t-c)}{d-c}$$

tilfredstille $P_n(t_k) = x_k$. Når n er odde vil funksjonen

$$P_n(t) = \frac{a_0}{\sqrt{n}} + \frac{2}{\sqrt{n}} \sum_{k=1}^{\frac{n-1}{2}} \left(a_k \cos \frac{2k\pi(t-c)}{d-c} - b_k \sin \frac{2k\pi(t-c)}{d-c} \right)$$

tilfredstille $P_n(t_k) = x_k$.

FFT og interpolering

La $A_k(t) = \frac{2k\pi(t-c)}{d-c}$. Vi kan for n et partall skrive om $P_n(t)$ til

$$P_n(t) = \frac{a_0}{\sqrt{n}} + \frac{2}{\sqrt{n}} \sum_{k=1}^{n/2-1} (a_k \cos A_k(t) - b_k \sin A_k(t)) + \frac{a_{n/2}}{\sqrt{n}} \cos A_n(t)$$

For n et oddetall blir dette:

Effektiv utregning av interpolerende funksjon

Funksjonen $P_n(t)$ gir oss de opprinnelige verdiene $P_n(t_j) = x_j$. I stedet for å regne ut den trigonometriske summen, så vet vi at vi får tilbake x_j ved den inverse DFT,

$$x = F_n^{-1}(a + ib)$$

Trikset er å se på $P_n(t)$ som en interpolerende funksjon for p jevnt fordelte dataverdier, $p > n$, hvor vi setter koeffisientene $a_k = b_k = 0$ for $n/2 < k \leq p/2$:

$$P_p(t) = \frac{\sqrt{\frac{p}{n}} a_0}{\sqrt{p}} + \frac{2}{\sqrt{p}} \sum_{k=1}^{n/2-1} \left(\sqrt{\frac{p}{n}} a_k \cos A_k(t) - \sqrt{\frac{p}{n}} b_k \sin A_k(t) \right) + \frac{\sqrt{\frac{p}{n}} a_{n/2}}{\sqrt{p}} \cos A_n(t)$$

Vi regner ut fourierkoeffisientene $y = F_n x$, ganger disse med $\sqrt{\frac{p}{n}}$ og setter inn i en p-dimensjonal vektor

$$\begin{bmatrix} a_0 \\ a_1 + ib_1 \end{bmatrix}$$

og setter inn i en p-dimensjonal vektor

$$yp = \sqrt{\frac{p}{n}} \begin{bmatrix} a_0 \\ a_1 + ib_1 \\ \vdots \\ a_{n/2-1} + ib_{n/2-1} \\ a_{n/2} \\ 0 \\ \vdots \\ 0 \\ a_{n/2-1} - ib_{n/2-1} \\ \vdots \\ a_1 - ib_1 \end{bmatrix}$$

$a_{\frac{n}{2}+1}$
$\vdots$
a

Det siste steget er å finne de interpolerte vektoren $xp = F_p^{-1} yp$

Nå vet vi at

$$F_n = \frac{1}{\sqrt{n}} \operatorname{fft}$$

og

$$F_p^{-1} = \sqrt{p} \operatorname{ifft}$$

Setter vi dette sammen, får vi interpolasjonsvektoren xp ved

$$\frac{p}{n} \cdot \operatorname{ifft}(p) \cdot \text{konverter vektor} \cdot \operatorname{fft}(n)$$

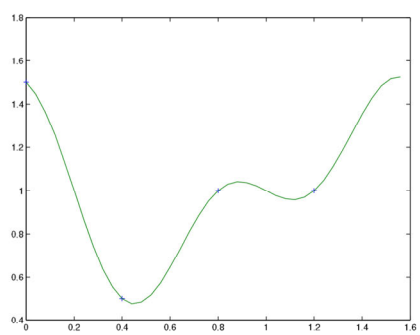
Implementasjon

```
function xp = dftinterpolasjon( intervall , x , n , p)
    close all;
    c = intervall(1);
    d = intervall(2);
    dtp = (d-c)/p;
    dtx = (d-c)/n;
    tp = c+(0:p-1)*dtp;
    tx = c+(0:n-1)*dtx;
    yn = fft(x);
    yp = zeros(p,1);
    yp(1:n/2+1)=yn(1:n/2+1);
    yp(p-n/2+2:p)=yn(n/2+2:n);
    xp = real(ifft(yp*p/n));
    plot(tx,x,'+',tp,xp);
end
```


TEST

```
>> x=[1.5 .5 1 1];
>> dftinterpolasjon( [0 1.6] , x , 4 , 40 );
```

gir:



3.04: Fourier 2 & Polynominterpolasjon 1

fredag 3. april 2020 07:47

forelesningTDAT2002-3-04-2020.pdf

forelesning...

Fouriertransform 2
FFT og signal-behandling+
Mer om interpolasjon

Dag Olav Kjellemo

Institutt for datateknologi og informatikk

3. april 2020

Plan

Læremål

10.3.1 Ortogonalitet og interpolering

10.3.2 Minste kvadraters metode med trigonometriske funksjoner.

3.1 Data og interpolerende funksjoner

Læremål

- Interpolering med trigonometriske funksjoner
- Filtrering
- Ideer om komprimering

Tema

Læremål

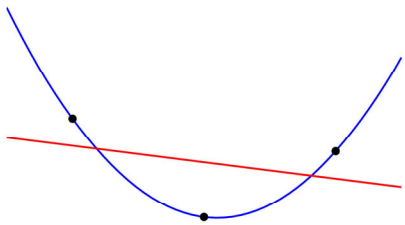
10.3.1 Ortogonalitet og interpolering

10.3.2 Minste kvadraters metode med trigonometriske funksjoner.

3.1 Data og interpolerende funksjoner

Modelering av data: Interpolasjon og tilpasning/approksimasjon

Den blå kurven interpolerer dataene. Den røde er en lineær tilpassing.



Modellering: Interpolasjon og tilpasning/approksimasjon

- Gitt kjente datapunkter $(t_0, x_0), \ldots (t_{n-1}, x_{n-1})$
- Interpolasjon er å estimere x-verdier for funksjonen for andre t-verdier
- dvs finne en funksjon $f(t)$ med $f(t_i) = x_i$
- For en *tilpansing/approksimasjon* trenger vi ikke likhet.
- Det finnes (selvfølgelig) uendelig mange interpolasjoner.
- Ønsker å ha systematisk måte å lage interpolasjon eller tilpasninger, med
- gode egenskaper (egnet modell, effektiv utregning og stabil numerikk)

Strategi: Velge en *basis* av funksjoner $f_0, \ldots f_{m-1}$ og lage interpolerende funksjon som lineær interpolasjon av disse:

$$F(t) = \sum_{i=0}^{m} c_i f_i(t), \;\; f(t_i) = x_i$$

c_i er parametrene vi må bestemme.

Eksempler

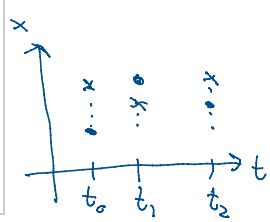
- $1, t, t^2, \ldots, t^m$; polynom-interopoasjon $F(t) = c_0 \cdot 1 + c_1 \cdot t + c_2 \cdot t^2 + \ldots + c_m t^m$
- Trigonometriske funskjoner, som vi straks skal se mer på.
- Flere varianter av polynomer

Egenskaper til basisfunksjonene

- La $t = [t_0 \ t_1 \ldots \ t_{n-1}]^T$, $x = [x_0 \ x_1 \ \ldots \ x_{n-1}]^T$
- Gitt *basisfunksjoner* $f_0, f_1, \ldots, f_{m-1}$
- Hvert punkt t_i gir en rad, og hver funksjon f_j en søyle i

$$B = \begin{bmatrix} f_0(t_0) & f_1(t_0) & \cdots & f_{m-1}(t_0) \\ f_0(t_1) & f_1(t_1) & \cdots & f_{m-1}(t_1) \\ \vdots & \vdots & \ddots & \vdots \\ f_0(t_{n-1}) & f_1(t_{n-1}) & \cdots & f_{m-1}(t_{n-1}) \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{m-1} \end{bmatrix} = c_0 \begin{bmatrix} f_0(t_0) \\ \vdots \\ f_0(t_{n-1}) \end{bmatrix} + c_1 \begin{bmatrix} f_1(t_0) \\ \vdots \\ f_1(t_{n-1}) \end{bmatrix} + \ldots$$

- For at lineærkombinasjonen $F(t) = \sum c_i f_i(t)$ skal være interpolerende, må $Bc = x$
- For at vi skal kunne interpolere alle valg av x, så må søylevektorene utspenne $\mathbb{R}^n$, $(m \geq n)$.
- Hvis de er søylene er lineært uavhengige, så er c entydige bestemt $(m \leq n)$.
- Hvis $m = n$, så er begge disse egenskapene ekvivalente med at B er ikke-singulær, dvs. invertibel.



Bestemme koeffisientene

For at

$$F(t_i) = \sum_{k=0}^{m-1} c_k f_k(t_i) = x_i$$

skal interpolerer punktene (t_i, x_i), må de være en løsning til likningssystemet

$$Bc = x$$

Hvis B er invertibel får vi

$$c = B^{-1}x$$

Det er spesielt enkelt å løse hvis $B^{-1} = B^T$, dvs. hvis B er ortogonal, eller hvis vi bruker komplekse tall, hvis B er unitær, $B^{-1} = \bar{B}^T$

Ortogonalitet og interpolasjon

- La $f_0(t), f_2(t), \ldots, f_{n-1}(t)$ være funksjoner av t.
- og $t_0, t_2, \ldots, t_{n-1}$ være reelle tall
- slik at

$$B^T = A = \begin{bmatrix} f_0(t_0) & f_0(t_1) & \cdots & f_0(t_{n-1}) \\ f_1(t_0) & f_1(t_1) & \cdots & f_1(t_{n-1}) \\ \vdots & \vdots & \ddots & \vdots \\ f_{n-1}(t_0) & f_{n-1}(t_1) & \cdots & f_{n-1}(t_{n-1}) \end{bmatrix}$$

  er ortogonal. $(A^T A = I)$
- og $\mathbf{y} = c = A\mathbf{x}$ $\quad c = B^{-1}x$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-1} \end{bmatrix}$$

Da interpolerer funksjonen

$$F(t) = \sum_{i=1}^{n} y_i f_i(t)$$

dataene $(t_1, x_1), (t_2, x_2), (t_3, x_3), \ldots, (t_n, x_n)$.
Dvs $F(t_1) = x_1, F(t_2) = x_2, \ldots$

Trigonometriske funksjoner.

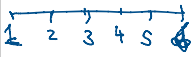
La n være partall.

- Del intervallet $[c, d]$ i n deler med lik lengde $h = (d - c)/n$.
- Delene starter i
  $t_0 = c, t_1 = c + h, t_2 = c + 2h, \ldots, t_{n-1} = c + (n-1)h$, $t_n = d$
- La

$$
\begin{aligned}
f_0(t) &= \sqrt{1/n} \quad \text{konstant funksjon} \quad , \\
f_1(t) &= \sqrt{2/n}\cos\tfrac{2\pi(t-c)}{d-c} \quad , \quad & f_2(t) &= \sqrt{2/n}\sin\tfrac{2\pi(t-c)}{d-c} \\
f_3(t) &= \sqrt{2/n}\cos\tfrac{4\pi(t-c)}{d-c} \quad , \quad & f_4(t) &= \sqrt{2/n}\sin\tfrac{4\pi(t-c)}{d-c} \\
&\vdots \\
f_{n-3}(t) &= \sqrt{2/n}\cos\tfrac{(n-2)\pi(t-c)}{d-c}, \quad & f_{n-2}(t) &= \sqrt{2/n}\sin\tfrac{(n-2)\pi(t-c)}{d-c} \\
f_{n-1}(t) &= \sqrt{1/n}\cos\tfrac{n\pi(t-c)}{d-c} \quad ,
\end{aligned}
$$

Da er $A = \left[f_i(t_j)\right]$ ortogonal.
Dette kan vises ved å bruke at Fourier-matrisen F_n er ortogonal, og at $e^{i\theta} = \cos\theta + i\sin\theta$.

$n = 5$



$c = 1, d = 6$

$h = \frac{6-1}{5} = 1$

$$\left.\begin{aligned} f_0(t_0) &= \tfrac{1}{\sqrt{n}} \\ f_1(t_1) &= \tfrac{1}{\sqrt{n}} \\ &\vdots \\ f_0(t_{n-1}) &= \tfrac{1}{\sqrt{2}} \end{aligned}\right\} \text{vektor med lengde } 1.$$

$\omega = e^{i\frac{2\pi}{n}}$ primitiv n'te rot.

$= \cos\left(\frac{2\pi}{n}\right) + i \cdot \sin\left(\frac{2\pi}{n}\right)$

DFT og utregning av koeffisientene y

Fourier-matrisen

Vi finner $\mathbf{z} = F_n\mathbf{x}$, $z_j = a_j + ib_j$.
Da er $\mathbf{y} =$
$[a_0, \sqrt{2}a_1, -\sqrt{2}b_1, \sqrt{2}a_2, -\sqrt{2}b_2, \ldots, \sqrt{2}a_{n/2-1}, -\sqrt{2}b_{n/2-1}, a_{n/2}]$.
Ved bruk av FFT så husk at

Matlab-funksjon

$$dft(x) = \frac{1}{\sqrt{n}} fft(x).$$

normalisering, dvs A blir ortogonal.

Tema

Læremål

10.3.1 Ortogonalitet og interpolering

10.3.2 Minste kvadraters metode med trigonometriske funksjoner.

3.1 Data og interpolerende funksjoner

FFT og signalbehandling

Vi har at $P_n(t)$ interpolerer dataene n partall
$(x_0, t_0), (x_1, t_1), \ldots, (x_{n-1}, t_{n-1})$

$$P_n(t) = \frac{a_0}{\sqrt{n}} + \frac{2}{\sqrt{n}} \sum_{k=1}^{n/2-1} \left(a_k \cos \frac{2k\pi(t-c)}{d-c} - b_k \sin \frac{2k\pi(t-c)}{d-c} \right) + \frac{a_{n/2}}{\sqrt{n}} \cos \frac{n\pi(t-c)}{d-c}$$

Interpolering er greit når signalet inneholder få punkter. Om punktene er mange så vil vi heller tilpasse en funksjon som er "best" tilpasser funksjonen og som fjerner de høyeste frekvensene.

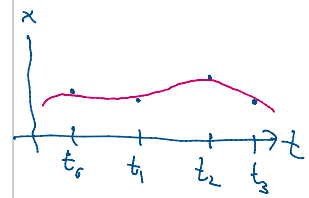
Beste minste kvadraters trigonometrisk tilpassing av orden m

La m være partall og $a_0, a_1, b_1, \ldots, a_{n-1}, b_{n-1}, a_n$ være koefisientene til interpolerende polynom. Ved å fjerne sette alle koefisientene fra b_m til a_n til null vil $m \leq n$

$$P_m(t) = \frac{a_0}{\sqrt{n}} + \frac{2}{\sqrt{n}} \sum_{k=1}^{m/2-1} \left(a_k \cos \frac{2k\pi(t-c)}{d-c} - b_k \sin \frac{2k\pi(t-c)}{d-c} \right) + \frac{a_{m/2}}{\sqrt{n}} \cos \frac{m\pi(t-c)}{d-c}$$

gi minst mulig kvadratisk feil $\sum (x_k - P_m(t_k))^2$, blant alle funksjoner

$$c_0 + \sum_{k=1}^{m/2-1} \left(c_k \cos \frac{2k\pi(t-c)}{d-c} - d_k \sin \frac{2k\pi(t-c)}{d-c} \right) + c_m \cos \frac{m\pi(t-c)}{d-c}$$



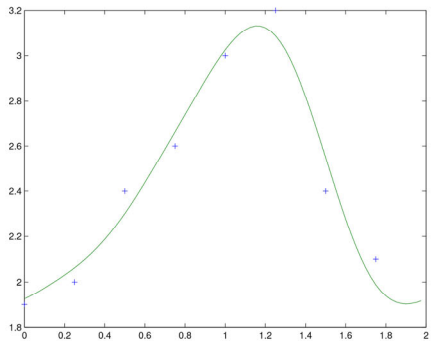
FFT og komprimering

Ved hjelp av minste kvadraters trigonometrisk tilpassing av orden m kastes informasjon om signalet. På denne måten vil dataene ta mindre plass på en slik måte at feilen blir minst mulig.

TEST

```
>> x = [ 1.9 2 2.4 2.6 3 3.2 2.4 2.1 ];
>> dftfilt( [ 0 2.0 ] , 8 , 6 , x , 80 );
```

gir:



Frame Title

Læremål

10.3.1 Ortogonalitet og interpolering

10.3.2 Minste kvadraters metode med trigonometriske funksjoner.

3.1 Data og interpolerende funksjoner

Polynom-interpolasjon: Lagranges interpolasjon

Gitt n punkter $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$.
Målet er å interpolerer disse punktene med et polynom $L_{n-1}(x)$ av grad $n-1$ (eller mindre). Idéen er som følgende:

- Finn n polynomer $L_i(x)$ der hvert polynom har verdiene

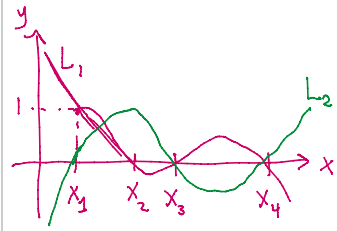
$$L_i(x) = \begin{cases} 1, & x = x_i \\ 0, & x = x_j, j \neq i \\ ?, & \text{ellers} \end{cases}$$

- Dette polynomet kan regnes ut ved

$$L_i = \frac{\overset{x_i - x_1}{(x - x_1)} \cdots \overset{x_i}{(x - x_{i-1})} \overset{x_i}{(x - x_{i+1})} \cdots \overset{x_i}{(x - x_n)}}{(x_i - x_1) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)}$$

- Lagrangepolynomet blir derfor

$$P_{n-1}(x) = \sum_{i=1}^{n} y_i L_i(x).$$



$L_i(x_i) = 1$
$L_i(x_j) = 0, \; i \neq j.$

$P_{n-1}(x_j) = y_j \underbrace{L_j(x_j)}_{=1} = y_j$

$$A = \begin{bmatrix} L_1(x_1) & L_1(x_2) & \cdots & L_1(x_n) \\ L_2(x_1) & L_2(x_2) & \cdots & \\ \vdots & & & \\ L_n(x_1) & L_n(x_2) & \cdots & \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & 0 \\ 0 & 0 & & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & & 1 \end{bmatrix}$$

Lagrange for tre punkter

Formel for lagrange for tre punkter er

$$P_2(x) = y_1 \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} + y_2 \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} + y_3 \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)}.$$

Vi har:

$$P_2(x_1) = y_1 \frac{(x_1-x_2)(x_1-x_3)}{(x_1-x_2)(x_1-x_3)} + y_2 \frac{(x_1-x_1)(x_1-x_3)}{(x_2-x_1)(x_2-x_3)} + y_3 \frac{(x_1-x_1)(x_1-x_2)}{(x_3-x_1)(x_3-x_2)} = y_1.$$
$$P_2(x_2) = y_1 \frac{(x_2-x_2)(x_2-x_3)}{(x_1-x_2)(x_1-x_3)} + y_2 \frac{(x_2-x_1)(x_2-x_3)}{(x_2-x_1)(x_2-x_3)} + y_3 \frac{(x_2-x_1)(x_2-x_2)}{(x_3-x_1)(x_3-x_2)} = y_2.$$
$$P_2(x_3) = y_1 \frac{(x_3-x_2)(x_3-x_3)}{(x_1-x_2)(x_1-x_3)} + y_2 \frac{(x_3-x_1)(x_3-x_3)}{(x_2-x_1)(x_2-x_3)} + y_3 \frac{(x_3-x_1)(x_3-x_2)}{(x_3-x_1)(x_3-x_2)} = y_3.$$

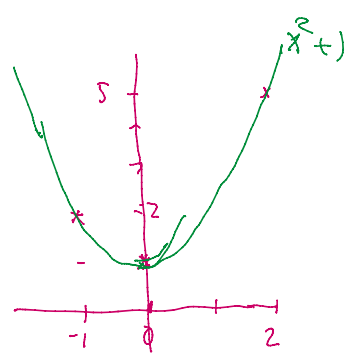
Hvis vi legger til (x_4, y_4) så må vi regne alt fra starten. Ingen gjenbruk.

Eksempel

Eksempel

Finn et polynom som går igjennom punktene $(-1, 2)$, $(0, 1)$, $(2, 5)$

$$2 \cdot \frac{(x-0)(x-2)}{(-1-0)(-1-2)} + 1 \frac{(x-(-1))(x-2)}{(0-(-1))(0-2)} + 5 \cdot \frac{(x-(-1))(x-0)}{(2-(-1))(2-0)}$$
$$= \frac{2}{3}(x)(x-2) - \frac{1}{2}(x+1)(x-2) + 1 \cdot (x+1)(x)$$
$$= x^2 + 0 \cdot x + 1 = x^2 + 1$$



Entydighet

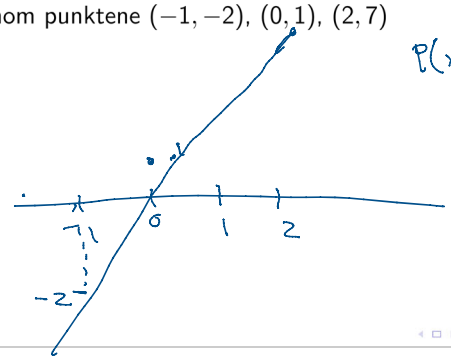
Teorem

Gjennom n punkter med med forskjellige x-verdier går det ett og bare ett polynom av grad $n-1$ eller lavere.

Eksempel

Finn det entydige polynomet av grad 2 eller lavere som går igjennom punktene $(-1,-2)$, $(0,1)$, $(2,7)$

$P(x) = 3x + 1$



Newtons dividerte differanser

Mål: interpolere en funksjon $f(x)$ i punktene

$$x_1, x_2, x_3, \ldots, x_n.$$

$y_1, y_2, \ldots y_n$

Definisjon

$f[x_1, x_2, \ldots, x_n]$ er koeffisienten til x^{n-1} i det entydige polynomet av grad $n-1$ eller mindre som interpolerer punktene

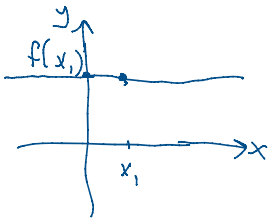
$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n).$$

Da er

$$\begin{aligned} P_n(x) = f[x_1] \quad &+ \quad f[x_1, x_2](x - x_1) \\ &+ \quad f[x_1, x_2, x_3](x - x_1)(x - x_2) \\ &+ \quad f[x_1, x_2, x_3, x_4](x - x_1)(x - x_2)(x - x_3) \\ &+ \quad \cdots \\ &+ \quad \underbrace{f[x_1, x_2, \ldots, x_n]}\underbrace{(x - x_1)(x - x_2)\cdots(x - x_{n-1})}_{\text{polynom av grad } n-1} \end{aligned}$$

lavere grad enn $n-1$.

$f[x_1] = f(x_1) = y_1$



Newtons dividerte differanser

Formler for å regne ut koefisientene

1. Verdiene til f:
$$f[x_k] = f(x_k).$$
2. Newtons dividerte differenser:
$$f[x_k, x_{k+1}, \ldots, x_{k+r}] = \frac{f[x_{k+1}, x_{k+2}, \ldots, x_{k+r}] - f[x_k, x_{k+1}, \ldots, x_{k+r-1}]}{x_{k+r} - x_k}$$

Skjema for å regne ut koeffisientene

$$\begin{array}{c|ccccc}
x_1 & f[x_1] & & & & \\
 & & f[x_1, x_2] & & & \\
x_2 & f[x_2] & & f[x_1, x_2, x_3] & & \\
 & & f[x_2, x_3] & & f[x_1, x_2, x_3, x_4] & \\
x_3 & f[x_3] & & f[x_2, x_3, x_4] & & \\
 & & f[x_3, x_4] & & & \\
x_4 & f[x_4] & & & &
\end{array}$$

$$= \frac{f[x_2, x_3, x_4] - f[x_1, x_2, x_3]}{x_4 - x_1}$$

$$= \frac{f[x_3, x_4] - f[x_2, x_3]}{x_4 - x_2}$$

$$\begin{aligned}
P(x) &= f[x_1] \\
&+ f[x_1, x_2](x - x_1) \\
&+ f[x_1, x_2, x_3](x - x_1)(x - x_2) \\
&+ f[x_1, x_2, x_3, x_4](x - x_1)(x - x_2)(x - x_3)
\end{aligned}$$

eksempler

Eksempel

a) Finn polynomet minimal grad igjennom punktene $(-1,-2)$, $(0,1)$, $(2,7)$, $(3,10)$.

-1	-2	3		
0	1		0	
		3		0
2	7		0	
		3		
3	10			

$$P(x) = -2 + 3(x-(-1)) + 0 \cdots + 0 \cdots$$
$$= -2 + 3x + 3$$
$$= 3x + 1$$

eksempler

Eksempel

b) Finn polynomet minimal grad igjennom punktene $(0,1)$,$(1,0)$,$(2,1)$,$(3,4)$

c) Som b) men med et ekstra punkt $(-1,3)$

0	1				
1	0	-1			
2	1	1	1		
3	4	3	1	0	
-1	3	$\frac{1}{4}$	$\frac{11}{36}$	$\frac{25}{72}$	$-\frac{25}{72}$

$$P(x) = 1 - 1(x-0) + 1(x-0)(x-1) + \underbrace{0}_{\text{3dje grads leddet}} - \frac{25}{72}(x-0)(x-1)(x-2)(x-(-1))$$

$$f[x_1\, x_2] = f[x_2\, x_1]$$

$$\frac{\frac{1}{4} - 3}{-1-2} = \frac{\frac{-11}{12}}{-3} = \frac{11}{36} \qquad \frac{\frac{11}{36} - 1}{-1-1} = \frac{\frac{11-36}{36}}{-2} = \frac{25}{72}$$

Høyere ordens polynomer

Hvis $P_3(x)$ interpolerer $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$ så interpolerer også uendelig mange polynomer av orden 4 eller større de samme punktene:

$$P_4(x) = P_3(x) + c \cdot \underbrace{(x - x_1)(x - x_2)(x - x_3)(x - x_4)}_{\text{null for alle } x_i\text{'ene.}}.$$

velges fritt.

Gjenvisitt med Horners metode

Newtons dividerte differenser

$$\begin{aligned} P(x) &= f[x_1] \\ &+ f[x_1, x_2](x - x_1) \\ &+ f[x_1, x_2, x_3](x - x_1)(x - x_2) \\ &+ f[x_1, x_2, x_3, x_4](x - x_1)(x - x_2)(x - x_3) \end{aligned}$$

Newtons dividerte differenser omskrevet med Horners metode.

$$P(x) = \Big(\big(f[x_1, x_2, x_3, x_4](x - x_3) + f[x_1, x_2, x_3] \big)(x - x_2) + f[x_1, x_2] \Big)(x - x_1) + f[x_1]$$

Repeterer funksjonen Nest

MATLAB-funksjonen nest hadde et valgfritt argument b:

```
function y = nest(d,c,x,b)
    % Detailed explanation goes here
    if nargin < 4, b=zeros(d,1); end
    y = c(d+1);
    for i=d:-1:1
        y = y.*(x-b(i))+c(i);
    end
end
```

Nå buker vi $b = (x_1, x_2, \ldots, x_n)$ som fjerde argument.

Implementasjon av Newtons dividerte differenser

```
function c=mynewtdd(xin,yin,n)
    x(:,1)=xin;
    v=zeros(n,n);
    v(:,1)=yin;
    for i=2:n
        delta=x(i:end)-x(1:end-i+1);
        Delta=v(i:end,i-1)-v(i-1:end-1,i-1);
        v(i:end,i)=Delta./delta;
    end
    c=diag(v);
end
```

Representasjon av funksjoner

Viktig anvendelse av interpolasjon: tilnærming av funksjoner.
Regn ut $y_i = f(x_i)$. Finn interpolerende polynom $P_n(x)$.

17.04: Interpolasjonsfei, splines og Bezierkurver

torsdag 16. april 2020 09:09

forelesning...

Forelesning
Interpolasjonsfeil
Splines og Bezierkurver

Dag Olav Kjellemo

Institutt for datateknologi og informatikk

17. april 2020

Plan

Læremål

3.2 Interpolasjonsfeil og Runges fenomen

3.4 Splines

3.5 Bezierkurver

Læremål

- Kjenne til feilformelen for interpolasjon
- Kjenne til Runges feonomen
- Kjenne til kubiske splines og deres bruksområder.
- Kunne beregne kubiske splines.
- Kjenne til Bezier-kurver og deres anvendelser
- Kunne algoritme for Bezierkurver.

Oversikt

Læremål

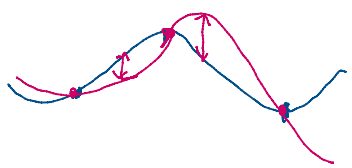
3.2 Interpolasjonsfeil og Runges fenomen

3.4 Splines

3.5 Bezierkurver

Interpolasjonsfeil

Den interpolerende funksjonen er korrekt i punktene vi interpolerer i, selvfølgelig. Hva kan vi si om andre punkter x?



Feilformel

Teorem polynom

Hvis $P(x)$ interpolerer $f(x)$ i punktene $x_1, x_2, \ldots, x_n$ så er feilen i x

feilen: $$f(x) - P(x) = \frac{(x-x_1)(x-x_2)\cdots(x-x_n)}{n!} f^{(n)}(c)$$

for en c som ligger mellom minste og største verdi av $x, x_1, x_2, \ldots, x_n$

Finnes ett polynom av grad n-1 eller lavere som interpolerer

c ligger her et sted

x_1 x_2 x x_3

Eksempel $\cos x$ og $P_3(x)$. $0, \frac{\pi}{6}, \frac{\pi}{3}, \frac{\pi}{2}$

$$|\cos x - P_3(x)| \leq \frac{(x-0)(x-\pi/6)(x-2\pi/6)(x-3\pi/6)}{24} \cdot 1$$

$(\cos x)^{(4)} = \cos x$

$|\cos x| \leq 1$

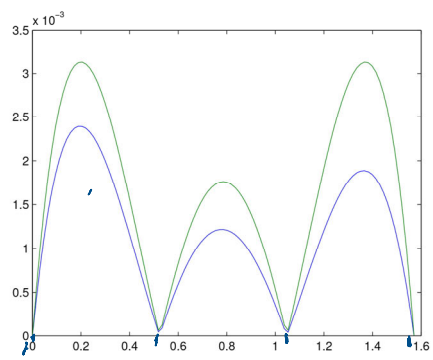
Feil-estimat

Setning

La $f(x)$ være en funksjon definert på $[a, b]$, la $P(x)$ være et polynom av lavest mulig grad som interpolerer $f(x)$ i punktene $x_1 < x_2 < \ldots < x_n$. La også $|f^{(n)}(x)| < M$ for alle $x \in [a, b]$. Da er absoluttverdien av feilen i x

$$|f(x) - P(x)| < \left| \frac{(x - x_1)(x - x_2) \cdots (x - x_n)}{n!} \right| M$$

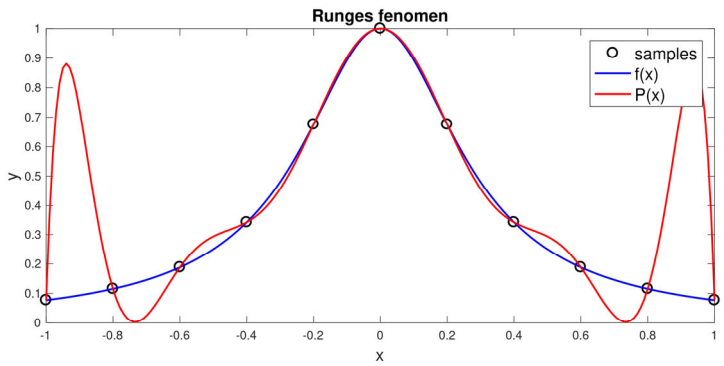
Feil og feil-estimat



Figur: Feil og feil-estimat for $P_3(x) \approx \cos x$.

Runge-fenomenet

Interpolering av $f(x) = 1/(1 + 12x^2)$ gir merkelige resultater:



Figur: Runge fenomenet

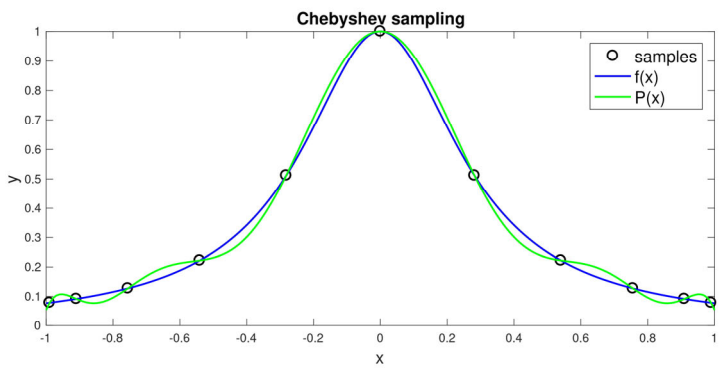
Chebyshev interpolasjon

Om vi sampler med forskjellig lengde mellom punktene, nærmere bestemt

$$x = cos\left(\frac{2k-1}{2n}\pi\right), \quad k = 1, 2, \ldots, n,$$

så blir estimert feil minst mulig på intervallet.

[a, b]
↑skalerer ved
[-1, 1] lineær omforming.



Figur: Chebyshev interpolasjon

Oversikt

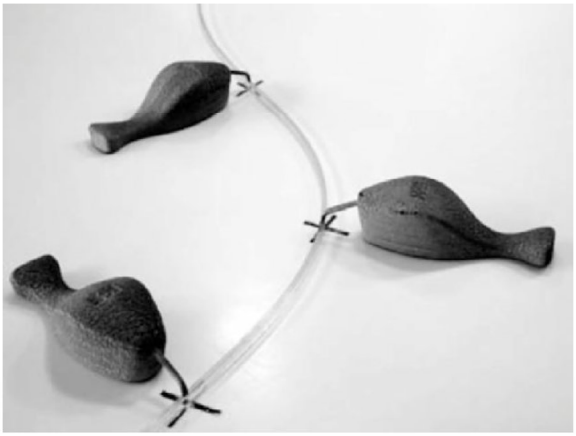
Læremål

3.2 Interpolasjonsfeil og Runges fenomen

3.4 Splines

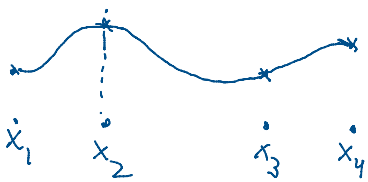
3.5 Bezierkurver

Kubiske Splines

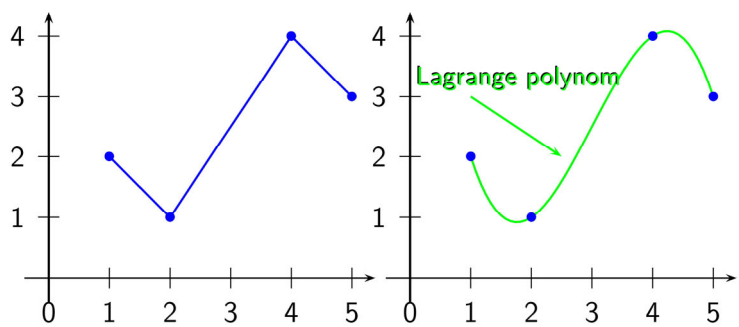


Kubiske Splines

- Hva: Interpoler (forbinder) punkter.
- Skiller seg fra Lagrange:
- Gitt data punkter $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$
- Finner 3-grads polynomer
- — som skjøter disse sammen.
- Tilbyr kontroll på endene.



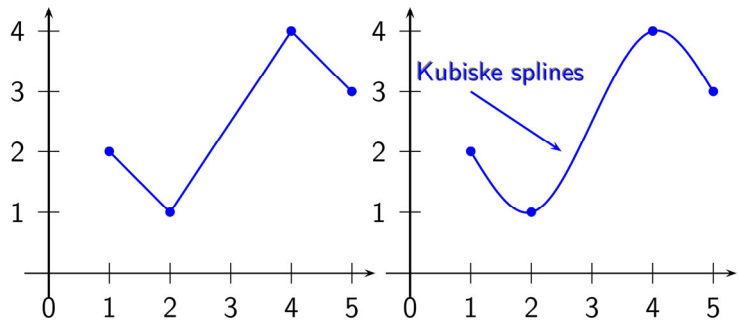
Kubiske splines v.s. Lagrange



Figur: Lineære og kubiske splines

$$
\begin{aligned}
\text{På } [1,2]\text{: } f_1(x) &= 2 - \tfrac{13}{8}(x-1) + \tfrac{5}{8}(x-1)^3 \\
\text{På } [2,4]\text{: } f_2(x) &= 1 + \tfrac{1}{4}(x-2) + \tfrac{15}{8}(x-2)^2 - \tfrac{5}{8}(x-2)^3 \\
\text{På } [4,5]\text{: } f_3(x) &= 4 + \tfrac{1}{4}(x-4) - \tfrac{15}{8}(x-4)^2 + \tfrac{5}{8}(x-4)^3
\end{aligned}
$$

Kubiske splines v.s. Lagrange



Figur: Lineære og kubiske splines

$$
\begin{aligned}
\text{På } [1,2]\text{: } f_1(x) &= 2 - \tfrac{13}{8}(x-1) + \tfrac{5}{8}(x-1)^3 \\
\text{På } [2,4]\text{: } f_2(x) &= 1 + \tfrac{1}{4}(x-2) + \tfrac{15}{8}(x-2)^2 - \tfrac{5}{8}(x-2)^3 \\
\text{På } [4,5]\text{: } f_3(x) &= 4 + \tfrac{1}{4}(x-4) - \tfrac{15}{8}(x-4)^2 + \tfrac{5}{8}(x-4)^3
\end{aligned}
$$

Om kubiske splines

Definisjon

Kubisk spline igjennom n punkter. En kubisk spline igjennom punktene $(x_1, y_1), \ldots, (x_n, y_n)$ består av $n-1$ kubiske polynomer:

$$
\begin{aligned}
S_1(x) &= \overset{a_1}{y_1} + b_1(x-x_1) + c_1(x-x_1)^2 + d_1(x-x_1)^3 \\
S_2(x) &= y_2 + b_2(x-x_2) + c_2(x-x_2)^2 + d_2(x-x_2)^3 \\
&\vdots \\
S_{n-1}(x) &= y_{n-1} + b_{n-1}(x-x_{n-1}) + c_{n-1}(x-x_{n-1})^2 + d_{n-1}(x-x_{n-1})^3
\end{aligned}
$$

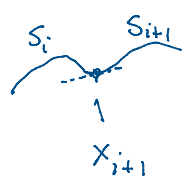
$x = x_1 \rightarrow S(x_1) = y_1$

Definert på henholdsvis $[x_1, x_2], [x_2, x_3], \ldots, [x_{n-1}, x_n]$ og med egenskapene

S1: $S_i(x_i) = y_i$ og $S_i(x_{i+1}) = y_{i+1}$.

S2: $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$ $\quad n-2$

S3: $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$ $\quad n-2$



Naturlig kubisk spline. (4a)

- $n-1$ splines gir $3n-3$ ukjente
- men bare $3n-5$ likninger.
- Systemet er ubestemt. Velger to nye likninger:

$$S_1''(x_1) = 0 \text{ og } S_{n-1}''(x_n) = 0.$$

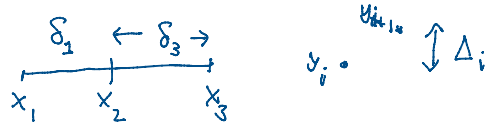
- En slik spline kalles for **naturlig**.

NB! Det finnes flere måter å tilpasse endene men det må alltid være to ekstra likninger.

Triks: Innfører ekstra ukjent

$$2c_n = S_{n-1}''(x_n). \qquad c_n = \frac{1}{2} S_{n-1}''(x_n)$$

Løsning av likningene



Skriver $\Delta y_i = \Delta_i = y_{i+1} - y_i$ og $\Delta x_i = \delta_i = x_{i+1} - x_i$.
Behandler tredje egenskap $S_i''(x_{i+1}) = S_{i+1}''(x_{i+1})$:

$$S_i''(x_{i+1}) = 2c_i + 6d_i\overbrace{(x_{i+1} - x_i)}^{\delta_i} = 2c_i + 6\delta_i d_i$$

$$S_{i+1}''(x_{i+1}) = 2c_{i+1} + 6d_{i+1}(\overbrace{x_{i+1} - x_{i+1}}^{0}) = 2c_{i+1}$$

$$d_i = \frac{c_{i+1} - c_i}{3\delta_i}, \quad i = 1, 2, \ldots, n-1.$$

$S_i(x) = y_i + b_i(x-x_i) + c_i(x-x_i)^2 + d_i(x-x_i)^3$
$S_i'(x) = b_i + 2c_i(x-x_i) + 3d_i(x-x_i)^2$
$S_i''(x) = 2c_i + 6d_i(x-x_i)$

Løsning av likningene

$S_i(x_i) = y_i \rightarrow a_i = y_i$

Vi behandler første egenskap $S_i(x_{i+1}) = y_{i+1}$:

$$S_i(x_{i+1}) = y_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2 + d_i(x_{i+1} - x_i)^3$$

$$y_{i+1} = y_i + b_i\delta_i + c_i\delta_i^2 + d_i\delta_i^3$$

Erstatter d_i med $\dfrac{c_{i+1} - c_i}{3\delta_i}$.

$$y_{i+1} = y_i + b_i\delta_i + c_i\delta_i^2 + (c_{i+1} - c_i)\delta_i^2/3$$

Løser for b_i:

$$b_i = \frac{\Delta_i}{\delta_i} - \frac{\delta_i}{3}(2c_i + c_{i+1}), \quad i = 1, 2, \ldots, n-1.$$

Det betyr at vi finner b-ene og d-ene så snart vi har funnet c-ene.

Likningene for $c_1, c_2, \ldots, c_n$

- Andre egenskap, $S_i'(x_{i+1}) = S_{i+1}'(x_{i+1})$, gir

$$b_i + 2c_i\underbrace{\delta_i}_{(x_{i+1}-x_i)} + 3d_i\delta_i^2 = b_{i+1} + \cancel{2c_{i+1}\overbrace{(x_{i+1} - x_{i+1})}^{0}}$$

- Ved å sette inn for b_i, b_{i+1}, d_i, d_{i+1} som vi fant over, og forenkler, får vi

$$\delta_i c_i + 2(\delta_i + \delta_{i+1})c_{i+1} + \delta_{i+1}c_{i+2} = 3\left(\frac{\Delta_{i+1}}{\delta_{i+1}} - \frac{\Delta_i}{\delta_i}\right),$$

$$i = 1, 2, \ldots, n-2$$

I tillegg kommer $S_1''(x_1) = 2c_1 = 0$ og $S_{n-1}''(x_n) = 2c_n = 0$.

Matriseform

$$A = \begin{bmatrix} 1 & 0 & 0 & & & & \\ \delta_1 & 2(\delta_1+\delta_2) & \delta_2 & \ddots & & & \\ 0 & \delta_2 & 2(\delta_2+\delta_3) & \delta_3 & & & \\ & \ddots & \ddots & \ddots & \ddots & & \\ & & & & & & 0 \\ & & 0 & \delta_{n-2} & 2(\delta_{n-2}+\delta_{n-1}) & \delta_{n-1} & \\ & & & 0 & 0 & 1 & \end{bmatrix}.$$

$$\mathbf{c} = \begin{bmatrix} c_1 & c_2 & c_3 & \cdots & c_n \end{bmatrix}^T$$

$$\mathbf{b} = \begin{bmatrix} 0 & 3\left(\frac{\Delta_2}{\delta_2} - \frac{\Delta_1}{\delta_1}\right) & \cdots & 3\left(\frac{\Delta_{n-1}}{\delta_{n-1}} - \frac{\Delta_{n-2}}{\delta_{n-2}}\right) & 0 \end{bmatrix}^T$$

Likningen blir $A\mathbf{c} = \mathbf{b}$. Den er strikt diagonal-dominant, og har alltid en løsning, siden $\delta_i > 0$

Krumingsjusterte kubiske splines (4b)

Vi kan kontrollere krumingen i endepunktene

$$S_1''(x_1) = 2c_1 = k_1, \quad S_{n-1}''(x_n) = 2c_n = k_n$$

Vi bytter ut første og siste rad med:

$$\left[\begin{array}{ccccccc|c} 2 & 0 & 0 & 0 & \cdots & 0 & 0 & k_1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 2 & k_n \end{array}\right]$$

```
...
    A(1,1:2)=[2 0]; A(n,n-1:n)=[0 2]; %
    B(1)=k1;        B(n)=kn;          %
...
```

Låste kubiske splines (4c)

Vi kan kontrollere tangenten i endepunktene

$$S_1'(x_1) = \delta_1(2c_1 + c_2) = v_1, \quad S_{n-1}'(x_n) = \delta_{n-1}(c_{n-1} + 2c_n) = v_n$$

Vi bytter ut første og siste rad med:

$$\left[\begin{array}{ccccccc|c} 2\delta_1 & \delta_1 & 0 & \cdots & 0 & 0 & 0 & v_1 \\ 0 & 0 & 0 & \cdots & 0 & \delta_{n-1} & 2\delta_{n-1} & v_n \end{array}\right]$$

Parabler i endene (4d)

Vi kan kontrollere den 3.-deriverte i endepunktene

$$d_1 = 0, \quad d_{n-1} = 0$$

Dette er ekivalent med at

$$c_1 = c_2, \quad c_{n-1} = c_n.$$

Vi bytter ut første og siste rad med:

$$\left[\begin{array}{ccccccc|c} 1 & -1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -1 & 0 \end{array}\right]$$

Not-a-knot (4e)

x_2 $\qquad$ x_{n-1}

Vi kan sørge for at $S_1(x) = S_2(x)$ og $S_{n-2}(x) = S_{n-1}$ ved at

$$d_1 = d_2, \quad d_{n-2} = d_{n-1}$$

Vi bytter ut første og siste rad med:

Not-a-knot (4e)

x_2 $\qquad$ x_{n-1}

Vi kan sørge for at $S_1(x) = S_2(x)$ og $S_{n-2}(x) = S_{n-1}$ ved at

$$d_1 = d_2, \quad d_{n-2} = d_{n-1}$$

Vi bytter ut første og siste rad med:

$$\left[\begin{array}{cccccccc|c} \delta_2 & -(\delta_1+\delta_2) & \delta_1 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \delta_{n-1} & -(\delta_{n-2}+\delta_{n-1}) & \delta_{n-2} & 0 \end{array}\right]$$

Eksempel

Vi vil finne en kubisk spline igjennom punktene

$$(0,0), (1/3, 1/4), (2/3, 3/4), (1,1)$$

Vi ønsker at den ~~tredje~~ andre-derivertе er null i endepunktene (Normal naturlig spline).

Eksempel

Utregning av Δ_i og δ_i.

$$\delta_1 = \delta_2 = \delta_3 = 1/3.$$

$$\Delta_1 = \Delta_3 = 1/4, \Delta_2 = 1/2$$

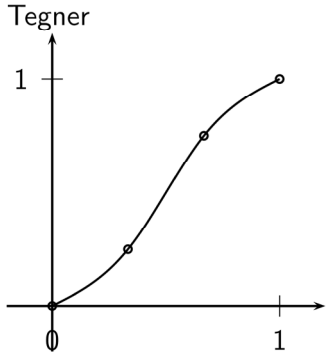
Matrisen

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1/3 & 4/3 & 1/3 & 0 \\ 0 & 1/3 & 4/3 & 1/3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Vektoren $\mathbf{b} = \begin{bmatrix} 0 & 9/4 & -9/4 & 0 \end{bmatrix}^T$ Vektoren $\mathbf{c} = A^{-1}\mathbf{b} = \begin{bmatrix} 0 & 9/4 & -9/4 & 0 \end{bmatrix}^T$.

Eksempel

Regner ut $d_1 = 9/4, d_2 = -9/2, d_3 = 9/4$.
Regner ut $b_1 = 1/2, b_2 = 5/4, b_3 = 5/4$.
Tegner



neste

Oversikt

Læremål

3.2 Interpolasjonsfeil og Runges fenomen

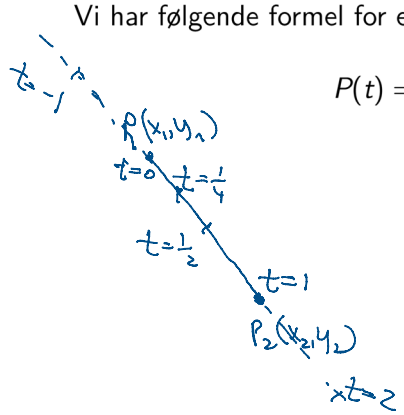
3.4 Splines

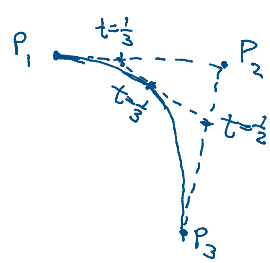
3.5 Bezierkurver

Rett linje mellom to punkter

Vi har følgende formel for en rett linje mellom to punkter P_1 og P_2

$$P(t) = (1-t)\,P_1 + t\,P_2.$$





Bezier kurve

Vi har følgende formel for en kurve mellom to punkter P_1 og P_4 som er "styrt" av punktene P_2 og P_3

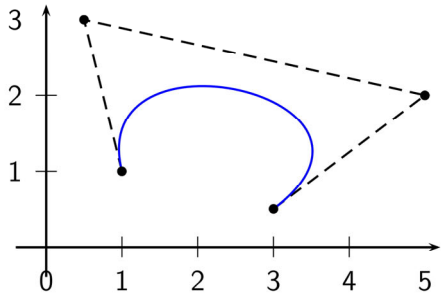
$$P(t) = (1-t)^3\, P_1 + 3\, t\,(1-t)^2\, P_2 + 3\, t^2(1-t)\, P_3 + t^3\, P_4.$$

Egenskaper

- $P(0) = P_1$
- $P(1) = P_4$
- $P'(0) = 3(P_2 - P_1)$
- $P'(1) = 3(P_4 - P_3)$

Eksempel

Linjene mellom punktene virker som styrelinjer.



Figur: Bezierkurve

Standardform

Gitt kontrollpunkter $P_1(x_1, y_1)$, $P_2(x_2, y_2)$, $P_3(x_3, y_3)$ og $P_4(x_4, y_4)$. Gitt

$$
\begin{aligned}
a_x &= x_1 & a_y &= y_1 \\
b_x &= 3(x_2 - x_1) & b_y &= 3(y_2 - y_1) \\
c_x &= 3(x_3 - 2x_2 + x_1) & c_y &= 3(y_3 - 2y_2 + y_1) \\
d_x &= x_4 - 3x_3 + 3x_2 - x_1 & d_y &= y_4 - 3y_3 + 3y_2 - y_1
\end{aligned}
$$

Da er

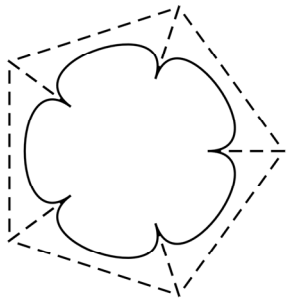
$$
\begin{aligned}
x(t) &= a_x + b_x t + c_x t^2 + d_x t^3 \\
y(t) &= a_y + b_y t + c_y t^2 + d_y t^3
\end{aligned}
$$

$(x(t), y(t))$

Anvendelser

- Postscript fonter — i PDF- og PS-filer
- 2D — Grafikk
- 3D — versjon for kurver i rommet.
- n-D — versjon for bevegelser av robot-armer med n frihetsgrader.

Eksempel: Blomst



20.05: Repetisjon, gjennomgang eksamensoppgaver

onsdag 20. mai 2020 12:41

TDAT2002-eksamen...

Noregs teknisk–naturvitskaplege universitet
Institutt for datateknikk og informasjonsvitenskap

Eksamen i matematikk2 – Numerikk
TDAT2002
Våren 2018

Løsningforslag

1 **a)** (i) Heltallsdelen er lik 0. Desimaldelen er 5/14. Algoritmen gir

$$\begin{aligned}
5/14 \cdot 2 &= 0 + 5/7 \\
5/7 \cdot 2 &= 1 + 3/7 \\
3/7 \cdot 2 &= 0 + 6/7 \\
6/7 \cdot 2 &= 1 + 5/7 \\
5/7 \cdot 2 &= 1 + 3/7 \\
3/7 \cdot 2 &= 0 + 6/7 \\
6/7 \cdot 2 &= 1 + 5/7 \\
&\vdots
\end{aligned}$$

Bitene gjentar seg med en periode på 3 fra 2 bit etter komma. Vi har derfor:

$$5/14 = (0.0\overline{101})_2$$

(ii) Enklste og raskeste måte er å bruke Horners metode

$7 \cdot 16^2 + 14 \cdot 16 + 2$

$$(7E2)_{16} = (7 \cdot 16 + 14) \cdot 16 + 2 = 2018.$$

Alternativt: Tabellen kan også brukes ved å skrive om $(7E2)_{16} = (111\,1110\,0010)_2$ og bruke omregning fra binær til desimal:

$$(7E2)_{16} = (111\,1110\,0010)_2 = 2^{11} + 2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^1 = 2018.$$

(iii) Heltallsdelen er $(101)_2 = 4 + 1 = 5$. Resten av tallet er $x = (0.\overline{1100})_2$. Vi har periode på fire bit i dette tallet. Vi ganger x med $2^4 = 16$ og får $16x = (1100.\overline{1100})_2$. Da er $16x - x = (1100.\overline{1100})_2 - (0.\overline{1100})_2 = 1100 = 12$. Da er $15x = 12$ og $x = 4/5$. Brøken blir $5 + 4/5 = 29/5$.

b) (i) Det største tallet som kan skrives i IEEE 754 halv-presisjon lagres som $(0)_2$ for fortegnet. Største eksponeneten er $(11110)_2 = 30$. Vi trekker fra eksponentbias og får $e = 30 - 15 = 15$. Mantissa bruker 10 pluss ett siffer.

$$m = 1 + \frac{2^{10} - 1}{2^{10}} = 1.9990234375$$

. Det gir

$$MAX_{halfprecition} = 2^{15}(1.9990234375) = 65504$$

$1.1111111111/00000$

$2^{16} = 65536 - 32$

$= 65504$

(ii) Det minste halvpresisjonstallet større enn 1 er

$$1 + \frac{1}{2^{10}} \approx 1.0009765625$$

. Verdien til maskin epsilon er $2^{-10} = 0.0009765625$.

(iii) Fortegnet er positivt. Eksponenten er lagret som $(10000)_2 = 16$. Siden bias er 15 er eksponenten lik $16 - 15 = 1$. Tallet er derfor $2^1\left(1 + \frac{(1001001000)_2}{2^{10}}\right) = 2 + \frac{(1001001)_2}{2^6} = 3 + \frac{9}{64} = 3.140625$.

c) Vi utvider brøken $\frac{\sqrt{1+\varepsilon}-1}{1}$ ved å gange med $\sqrt{1+\varepsilon}+1$ over og under brøkstreken

$$\frac{\sqrt{1+\varepsilon}-1}{1} = \frac{(\sqrt{1+\varepsilon}-1)(\sqrt{1+\varepsilon}+1)}{\sqrt{1+\varepsilon}+1} = \underline{\frac{1+\varepsilon-1}{\sqrt{1+\varepsilon}+1}} \approx \frac{\varepsilon}{2}.$$

$\boxed{2}$ **a)** Det jeg ikke har tatt høyde for er swamping. Det jeg burde ha gjort er å bytt om på radene. (Delvis pivotering.)

b) Vi gjennomfører Gauss-eliminasjon med delvis pivotering.

$$\begin{bmatrix} 10^{-20} & -1 \\ 1 & 2 \end{bmatrix} \begin{matrix} \leftarrow \\ \leftarrow \end{matrix} \sim \begin{bmatrix} 1 & 2 \\ 10^{-20} & -1 \end{bmatrix} \begin{matrix} \leftarrow (-10^{-20}) \\ \leftarrow + \end{matrix} \begin{bmatrix} 1 & 2 \\ 0 & -1 \end{bmatrix}$$

Det gir $L = \begin{bmatrix} 1 & 0 \\ 10^{-20} & 1 \end{bmatrix}$, $U = \begin{bmatrix} 1 & 2 \\ 0 & -1 \end{bmatrix}$, $P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$.

$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{matrix} \leftarrow \\ \leftarrow \end{matrix} \rightarrow \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

$\boxed{3}$ **a)** Jeg velger å bruke Newtons dividerte differenser.

$$\begin{array}{c|l}
-1 & 4 \\
 & \\
0 & -1 \\
 & \\
1 & -2 \\
 & \\
2 & 1
\end{array}
\qquad
\begin{aligned}
&f[-1,0] = \frac{f[0]-f[-1]}{0-(-1)} = -5 \\
&f[-1,0,1] = \frac{f[0,1]-f[-1,0]}{1-(-1)} = 2 \\
&f[0,1] = \frac{f[1]-f[0]}{1-0} = -1 \qquad f[-1,0,1,2] = \frac{f[0,1,2]-f[-1,0,1]}{2-(-1)} = 0 \\
&f[0,1,2] = \frac{f[1,2]-f[0,1]}{2-0} = 2 \\
&f[1,2] = \frac{f[2]-f[1]}{2-1} = 3
\end{aligned}$$

Det gir

$$P(x) = 4 - 5(x+1) + 2(x+1)x.$$

Vi kan omforme til følgende men det er ikke nødvendig for å få full uttelling.

$$P(x) = 2x^2 - 3x - 1.$$

b) (i) Nøyaktig 1 polynom av grad 2 interpolerer de gitte punktene i a). Begrunnelse. Det er nøyaktig et polynom av orden 3 eller lavere som interpolerer 4 punkter med forskjellige x verdier. Fra punkt a) ser vi at dette polynomet er av grad 2.

(ii) Ingen polynomer av grad 3 interpolerer de gitte punktene i a). Det eneste polynomet av grad 3 eller lavere som interpolerer punktene er av grad 2.

(iii) Det er uendelig mange polynomer av grad 4 som interpolerer de fire punktene. Alle polynomene

null i alle gitte x-verdier

$$P_3(x) + c\underline{(x+1)x(x-1)(x-2)}$$

der c er et hvilket som helst reelt tall interpolerer de 4 punktene.

c) Den beste lineær tilnærmingen av punktene er en lineær funksjon $y = c_0 + c_1 x$. Innsatt verdiene for de fire punktene får vi fire likninger

$$\begin{array}{rcrcr} c_0 & - & c_1 & = & 4 \\ c_0 & & & = & -1 \\ c_0 & + & c_1 & = & -2 \\ c_0 & + & 2c_1 & = & 1 \end{array}$$

eller $A\mathbf{c} = \mathbf{y}$ der

$$A = \begin{bmatrix} 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}, \ \mathbf{c} = \begin{bmatrix} c_0 \\ c_1 \end{bmatrix} \text{ og } \mathbf{y} = \begin{bmatrix} 4 \\ -1 \\ -2 \\ 1 \end{bmatrix}$$

Vi løser normallikningene $A^T A\bar{\mathbf{c}} = A^T\mathbf{y}$:
$A^T A = \begin{bmatrix} 4 & 2 \\ 2 & 6 \end{bmatrix}$ og $A^T\mathbf{y} = \begin{bmatrix} 2 \\ -4 \end{bmatrix}$. Vi får $\bar{\mathbf{c}} = (A^T A)^{-1} A^T\mathbf{y} = \frac{1}{20}\begin{bmatrix} 6 & -2 \\ -2 & 4 \end{bmatrix}\begin{bmatrix} 2 \\ -4 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$.
Beste lineære tilnærming er derfor

$$y = 1 - x.$$

d) DFT av vektoren

$$\begin{bmatrix} -1 \\ -2 \\ 1 \\ 4 \end{bmatrix}$$

er

$$\frac{1}{\sqrt{4}}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}\begin{bmatrix} -1 \\ -2 \\ 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 \\ -1+3i \\ -1 \\ -1-3i \end{bmatrix}$$

$= \frac{1}{\sqrt{4}} \, fft([-1 \ -2 \ 1 \ 4]')$

transpose-X

e) Først bestemmer vi verdien av c og d. Vi har $c = x_0 = 0$ og $d = x_4 = 4$. Videre er $a_0 = 1, a_1 = -1, a_2 = -1$ og $b_1 = 3$ Formelen for trigonometrisk er

$p = \frac{2\pi}{d-c}$

$$y(t) = \frac{a_0}{\sqrt{n}} + \frac{2}{\sqrt{n}}\sum_{k=1}^{\frac{n}{2}-1}\left(a_k \cos\frac{2k\pi(t-c)}{d-c} - b_k \sin\frac{2k\pi(t-c)}{d-c}\right) + \frac{a_{n/2}}{\sqrt{n}}\cos\frac{n\pi(t-c)}{d-c}$$

$\cos(p(t-c))$

I vårt tilfelle blir det.

$$y(t) = \frac{1}{2} - \cos\frac{\pi t}{2} - 3\sin\frac{\pi t}{2} - \frac{1}{2}\cos\pi t$$

f) Siden det trigonometriske polynomet har **periode 4** så er $y(-1) = y(-1+4) = y(3) = 4$. Direkte utregning gis også noe poeng: Siden $y(-1) = \frac{1}{2} - \cos\frac{-\pi}{2} - 3\sin\frac{-\pi}{2} - \frac{1}{2}\cos(-\pi t) = \frac{1}{2} - 0 - 3\cdot(-1) - \frac{1}{2}\cdot(-1) = 4$ så interpolerer $y(t)$ også punktene fra deloppgave a).

$\boxed{4}$ **a)** Vi skal finne $\|A\|_\infty$ til matrisen

$$A = \begin{bmatrix} 6 & 3 & -1 \\ 2 & -4 & 1 \\ -4 & 1 & -6 \end{bmatrix}.$$

$\|x\|_\infty = \max_{i=1}^{n} |x_i|$

$\|A\| = \max_{\|x\|=1} \|Ax\|$

Det er maksimal radsum er $\|A\|_\infty = |-4| + 1 + |-6| = 11$.
Vi skal finne $\|A\|_1$ til matrisen

$$A = \begin{bmatrix} 6 & 3 & -1 \\ 2 & -4 & 1 \\ -4 & 1 & -6 \end{bmatrix}.$$

Det er maksimal kolonnesum: $\|A\|_1 = 6 + 2 + |-4| = 12$.

b) En slik vektor er $v = \begin{bmatrix}-1 & 1 & -1\end{bmatrix}^T$ fordi

$$\|Av\|_\infty = \left\| \begin{bmatrix} 6 & 3 & -1 \\ 2 & -4 & 1 \\ -4 & 1 & -6 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix} \right\|_\infty = \left\| \begin{bmatrix} 2 \\ 7 \\ -11 \end{bmatrix} \right\|_\infty = \max\{|2|, |7|, |-11|\} = 11$$

er lik $\|A\|_\infty / \|v\|_\infty = 11/1 = 11$.

c) En slik vektor er $w = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$ fordi

$$\|Aw\|_1 = \left\| \begin{bmatrix} 6 & 3 & -1 \\ 2 & -4 & 1 \\ -4 & 1 & -6 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right\|_1 = \left\| \begin{bmatrix} 6 \\ 2 \\ -4 \end{bmatrix} \right\|_1 = |6| + |2| + |-4| = 12$$

er lik $\|A\|_1 / \|w\|_1 = 12/1 = 12$.

5 **a)** (i) Potensiterasjon gir altid den dominante egenverdien. Det ver egenverdien med størst absoluttverdi. I dette tilfellet er det 4.

(ii) Invers potensiterasjon uten skift gir altid egenverdien med lavest absoluttverdi. I dette tilfellet er det 1.

(iii) Invers potensiterasjon med skift $s = 3$ gir egen verdien nærmest 3. Det er 4.

(iv) Invers potensiterasjon med skift $s = -3$ gir egenverdien nærmest -3. Det er -2.

Egenv. til A^{-1} er $\frac{1}{\lambda}$ egen. til A.

b) Potensiterasjonen er som følgende.

$$u_0 = \frac{1}{\sqrt{3}} \begin{bmatrix}1 & -1 & 1\end{bmatrix}^T$$

norm.

$$x_1 = Au_0 = \frac{1}{\sqrt{3}} \begin{bmatrix}4 & -5 & 2\end{bmatrix}^T$$

$$\lambda_1 = u_0^T x_1 = \frac{11}{3}$$

c) For å finne ut hvilken verdi Rayleight Kotientiterasjon konvergerer mot er det tilstrekkelig å regne ut λ_0.

$$u_0 = \frac{1}{\sqrt{3}} \begin{bmatrix}1 & 1 & 1\end{bmatrix}^T$$

$$\lambda_0 = u_0^T A u_0 = \frac{-5}{3}$$

Siden λ_0 er nærmest -2 vil Rayleighs kvotientiterasjon konvergere mot -2.

6 **a)** Vi bruker formelen gitt på forelesning i stedet for den i boka.

$$B(t) = (1-t)^3 \begin{bmatrix}2\\2\end{bmatrix} + 3(1-t)^2 t \begin{bmatrix}4\\0\end{bmatrix} + 3(1-t)t^2 \begin{bmatrix}4\\4\end{bmatrix} + t^3 \begin{bmatrix}2\\2\end{bmatrix}$$

Ekspansjon gir

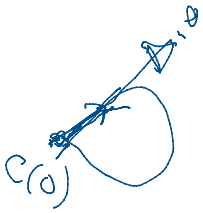
$$B(t) = \begin{bmatrix} 2 + 6t - 6t^2 \\ 2 - 6t + 18t^2 - 12t^3 \end{bmatrix}$$

b) Speilingen av kurven er gitt ved å speile x-vediene om linjen verdien 2. Det fås ved å translatere 2 til 0, skifte fortegn for så å translatere fra 0 til 2

$$x \mapsto x - 2 \mapsto -(x-2) \mapsto -(x-2) + 2 = 4 - x.$$

Formel for speilet $B(t)$ er

$$C(t) = \begin{bmatrix} 4 - (2 + 6t - 6t^2) \\ 2 - 6t + 18t^2 - 12t^3 \end{bmatrix}$$



Vi deriverer for å finne kontrollpunktene

$$C'(t) = \begin{bmatrix} -6 + 12t \\ -6 + 36t - 36t^2 \end{bmatrix}$$

Første kontrollpunkt er

$$C(0) + \frac{1}{3}C'(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Andre kontrollpunkt er

$$C(1) - \frac{1}{3}C'(1) = \begin{bmatrix} 0 \\ 4 \end{bmatrix}$$

Det gis også poeng for å si noe som for eksempel. Kontrollpunktene er også speilet om den vertikale linjen $x = 2$ slik som kurven. Det gir kontroll punktene

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

og

$$\begin{bmatrix} 0 \\ 4 \end{bmatrix}.$$