Fra M3 husker du at dersom x_i er $n+1$ forskjellige punkter på x-aksen med korresponderende y-verdier y_i, finnes det et entydig polynom av maksimal grad n som interpolerer punktene (x_i, y_i). I dette kapitlet skal vi sette opp to forskjellige formler for dette polynomet, og ta en litt grundigere analyse. Til slutt skal vi også ta en titt på hvordan dette kan gjøres med trigonometriske funksjoner.

Lagranges interpolasjon

La x_i være $n+1$ forskjellige punkter på intervallet $[a,b]$, med $x_0 = a$ og $x_n = b$. For hvert punkt x_i, definerer vi et polynom:

$$l_i(x) = \prod_{\substack{k=0 \\ k \neq i}}^{n} \frac{(x - x_k)}{(x_i - x_k)}.$$

Polynomet $l_i(x)$ har orden n, og tilfredsstiller

$$l_i(x_k) = \begin{cases} 1 & \text{for } i = k \\ 0 & \text{for } i \neq k \end{cases}.$$

La f være en funksjon, med funksjonsverdier $f(x_i) = f_i$. Det er lett å se at

$$p_n(x) = \sum_{i=0}^{n} f_i l_i(x).$$

tilfredsstiller $p_n(x_i) = f(x_i)$ for alle i.

Teorem 8.1. *La x_i være $n+1$ forskjellige punkter på intervallet $[a,b]$, og f en funksjon med funksjonsverdier $f(x_i) = f_i$. Det finnes et entydig polynom som tilfredsstiller $p_n(x_i) = f(x_i)$ for alle i.*

Bevis. Konstruksjonen av Lagranges interpolasjon viser at det for en tabell med $n+1$ punkter eksisterer et interpolasjonspolynom av maksimal grad n; vi har jo nettopp konstruert det. Hvis vi antar at det finnes to forskjellige polynomer p_n og q_n av grad n som interpolerer den samme tabellen, og evaluerer differansen $p_n - q_n$ i punktene x_i, ser vi at

$$p_n(x_i) - q_n(x_i) = 0 \qquad 0 \leq i \leq n.$$

Men polynomet $p - q$ har maksimal grad n, og kan følgelig ha maksimalt n nullpunkter, så den eneste muligheten her er $p = q$, som betyr at interpolasjonspolynomet er entydig. □

Eksempel 8.2. En funksjon har følgende verdier:

i	0	1	2
x_i	1	2	3
f_i	4	5	6

Vi setter opp

$$l_0(x) = \frac{(x-2)(x-3)}{(1-2)(1-3)} = \frac{1}{2}(x-2)(x-3)$$
$$l_1(x) = \frac{(x-1)(x-3)}{(2-1)(2-3)} = -(x-1)(x-3)$$
og
$$l_2(x) = \frac{(x-1)(x-2)}{(3-1)(3-2)} = \frac{1}{2}(x-1)(x-2).$$

Det andre ordens polynomet som interpolerer denne tabellen er:

$$\begin{aligned} p(x) &= 4l_0(x) + 5l_1(x) + 6l_2(x) \\ &= 2(x-2)(x-3) - 5(x-1)(x-3) \\ &\quad + 3(x-1)(x-2). \end{aligned}$$ △

I gamle dager sto det i numerikkbøkene at lagrangepolynomene ikke måtte brukes i numeriske beregninger, fordi det koster for mange flyttalsoperasjoner å evaluere dem. Dette er bare tull dersom man bruker de rette interpolasjonspunktene og evaluerer polynomene på rett måte, men dette er dessverre utenfor vårt pensum.

Newtons interpolasjon

For å konstruere Newtons interpolasjon trenger vi å beregne de *dividerte differansene*. De defineres rekursivt:

$$f[x_i] = f(x_i)$$

$$f[x_i, x_{i-1}] = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

$$f[x_i, ..., x_{i-k}] = \frac{f[x_i, ..., x_{i-k+1}] - f[x_{i-1}, ..., x_{i-k}]}{x_i - x_{i-k}}$$

Newtons interpolasjonspolynom er:

$$p_n(x) = f_0 + \sum_{i=1}^{n} f[x_i, ..., x_0] \prod_{k=0}^{i-1} (x - x_k).$$

Merk også at Lagranges og Newtons interpolasjon er bare to forskjellige formler for å sette opp det samme polynomet, siden interpolasjonspolynomet er entydig.

Eksempel 8.3. Polynomet

$$\begin{aligned} p_2(x) = \\ f_0 + \frac{f_1 - f_0}{x_1 - x_0}(x - x_0) \\ + \frac{\frac{f_2 - f_1}{x_2 - x_1} - \frac{f_1 - f_0}{x_1 - x_0}}{x_2 - x_0}(x - x_0)(x - x_1) \end{aligned}$$

interpolerer tabellen

x_0	x_1	x_2
f_0	f_1	f_2

så lenge $x_0 \neq x_1 \neq x_2$. △

Det er vanlig å sette opp følgende tableau for å illustrere de dividerte differansene:

$$\begin{array}{llll} x_0 & f[x_0] & & \\ & & f[x_0,x_1] & \\ x_1 & f[x_1] & & f[x_0,x_1,x_2] \\ & & f[x_1,x_2] & & f[x_0,x_1,x_2,x_3] \\ x_2 & f[x_2] & & f[x_1,x_2,x_3] \\ & & f[x_2,x_3] & \\ x_3 & f[x_3] & & \end{array}$$

Eksempel 8.4. La igjen

i	0	1	2
x_i	1	2	3
f_i	4	5	6

Tableauet blir

$$\begin{array}{llll} 1 & 4 & & \\ & & 1 & \\ 2 & 5 & & 0 \\ & & 1 & \\ 3 & 6 & & \end{array}$$

og interpolasjonspolynomet blir

$$p(x) = 4 + x - 1 = 3 + x.$$

Dette er selvfølgelig det samme polynomet som i forrige eksempel. Merk at polynomet er av første grad, siden punktene tilfeldigvis ligger på en rett linje. △

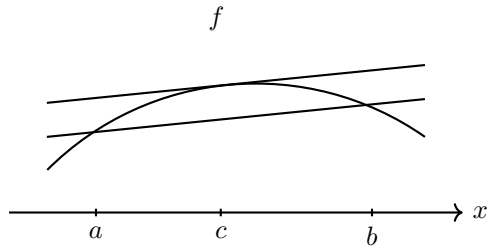
Interpolasjonsfeilen

Det sier seg selv at et interpolasjonspolynom ikke kan være lik funksjonen som interpoleres med mindre denne funksjonen er et polynom av ikke høyere grad enn interpolanten. Nå er vi kommet til steg to i den generelle oppskriften for numeriske metoder, nemlig analyse av feilen. Da må vi begynne med en generalisering av middelverdisatsen.

Middelverdisatsen sier at

$$f'(c) = \frac{f(b) - f(a)}{b - a} = f[a, b]$$

for en funksjon som er deriverbar på $[a, b]$.



En generalisert variant for dividerte differenser går som følger.

Teorem 8.5. *Dersom f er $n+1$ ganger deriverbar på $[a, b]$, og alle punktene x_i er forskjellige, er*

$$f[x_n, ..., x_0] = \frac{f^n(s)}{n!}$$

for en eller annen s i intervallet $[a, b]$.

Bevis. Siden p_n interpolerer f, må funksjonen $g = f - p_n$ ha minst $n+1$ nullpunkt på intervallet $[a, b]$. Gjentatt anvendelse av middelverdisatsen forteller oss at g' har minst n nullpunkt, at g'' har minst $n-1$ nullpunkt, og videre at g^{n+1} har minst ett nullpunkt på $[a, b]$. Vi kaller dette s. Siden

$$\frac{d^n}{dx^n} p_n(x) = n! f[x_n, ..., x_0],$$

må

$$f^n(s) = n! f[x_n, ..., x_0]. \qquad \square$$

Vi kan nå utlede et uttrykk for interpolasjonsfeilen.

Teorem 8.6. *La f være en $n+1$ ganger deriverbar funksjon på $[a, b]$, interpolert i punktene x_i.* Interpolasjonsfeilen *er*

$$f(x) - p_n(x) = \frac{f^{n+1}(s)}{(n+1)!} \prod_{k=0}^{n} (x - x_k).$$

Bevis. Vi skriver opp Newtons interpolasjonspolynom

$$\begin{aligned} p_n(x) = & \\ & f_0 + \sum_{i=1}^{n-1} f[x_i, ..., x_0] \prod_{k=0}^{i-1} (x - x_k) \\ & + f[x_n, ..., x_0] \prod_{k=0}^{n-1} (x - x_k), \end{aligned}$$

det siste leddet er skrevet ut kun av pedagogiske hensyn. Nå bytter vi ut x_n med x i uttrykket over (tenk på x som et nytt interpolasjonspunkt), og får

$$\begin{aligned} f(x) = & \\ & f_0 + \sum_{i=1}^{n-1} f[x_i, ..., x_0] \prod_{k=0}^{i-1} (x - x_k) \\ & + f[x, ..., x_0] \prod_{k=0}^{n-1} (x - x_k). \end{aligned}$$

Merk den snedige måten å skrive om f på. Trekker vi de to foregående uttrykkene fra hverandre, får vi

$$\begin{aligned} f(x) - p_n(x) = & \\ (f[x, x_{n-1}..., x_0] - f[x_n, x_{n-1}, ..., x_0]) & \prod_{k=0}^{n-1} (x - x_k) \\ = f[x, x_n, ..., x_0](x - x_n) & \prod_{k=0}^{n-1} (x - x_k) \\ = f[x, x_n, ..., x_0] & \prod_{k=0}^{n} (x - x_k), \end{aligned}$$

og bruker vi den generaliserte middelverdisatsen over, får vi

$$f(x) - p_n(x) = \frac{f^{n+1}(s)}{(n+1)!} \prod_{k=0}^{n} (x - x_k),$$

for en eller annen $s \in [a, b]$. Merk at s avhenger av x, akkurat som i Taylors teorem fra M1. $\square$

Punktfordeling

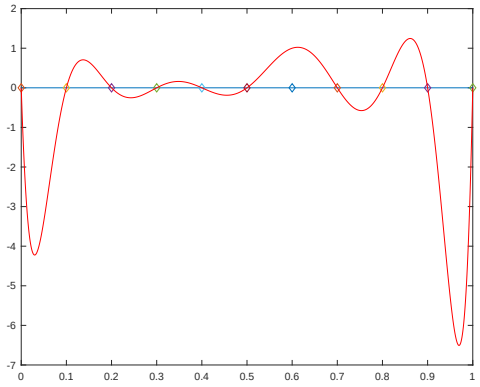
Steg tre i oppskriften på numeriskemetoder, er å finne ut om metoden kan ha noen gode egenskaper utover høy presisjon. I interpolasjonsfaget er det en relativt kjapp måte å avgjøre om en interpolasjonsmetode er god eller ikke: Det er avgjørende for kvaliteten på interpolasjonen at punktene står riktig fordelt på intervallet $[a, b]$.

Men hvordan skal vi finne gode punkter for polynominterpolasjon, og hva skiller de gode fra de dårlige punktene? Dette spørsmålet kan besvares på mange måter, og vi skal vende tilbake til spørsmålet i kapitlet om numerisk integrasjon. La oss begynne med å ta for oss en god og en dårlig punktmengde.

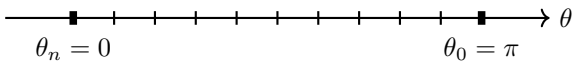
Eksempel 8.7. Den dårlige punktmengden er den kjente og kjære *ekvidistante* punktmengden. Et ekvidistant gitter med n punkter på intervallet $[a, b]$ er gitt ved

$$x_i = a + hi$$

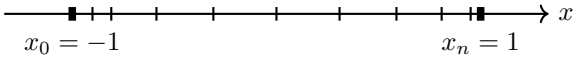
der $0 \leq i \leq n$ og $h = (b - a)/n$. I figuren under er et plot av en lagrangefunksjon på et ekvidistant gitter med elleve punkt. Merk oscillasjonene polynomet gjør mellom interpolasjonspunktene. △



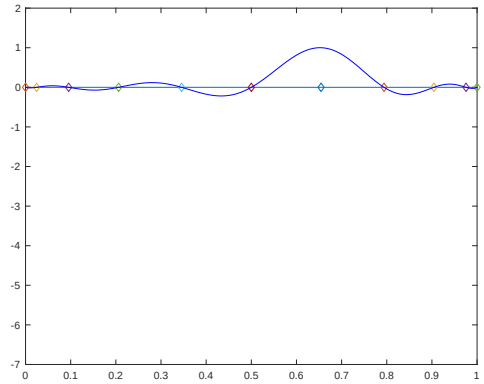
Eksempel 8.8. La θ_i være et ekvidistant gitter på $[0, \pi]$, med $\theta_n = 0$ og $\theta_0 = \pi$.



Nå definerer vi $x_i = \cos\theta_i$. Dette gitteret ligger på $[-1, 1]$.



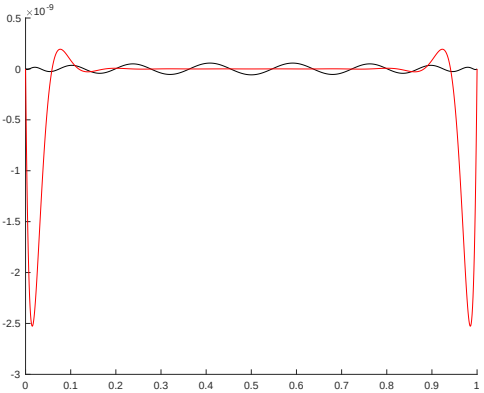
Det gitteret egner seg skikkelig godt for polynominterpolasjon. Under er et tilsvarende plot av en lagrangefunksjon på dette gitteret. Merk hvordan interpolasjonspolynomet ikke oscillerer særlig mellom interpolasjonspunktene. △



Man finner gode punkter for interpolasjon ved å plassere dem slik at de minimerer utslaget til polynomet

$$\prod_{k=0}^{n}(x - x_k)$$

i interpolasjonsfeilen, og alle gode punktmengder for polynominterpolasjon klumper seg i endene av intervallet. For ekvidistante gitre har feilpolynomet stort utslag i endepunktene, og det forklarer hvorfor lagrangefunksjonen i figuren over har størst utslag fra funksjonsverdiene der. Vi skal nå ta for oss noen forskjellige gode punktmengder.



Vi skal gjøre alt på intervallet $[-1, 1]$. Dersom man har en punktfordeling x_i på dette intervallet, kan man flytte fordelingen til intervallet $[a, b]$ med formelen gitt i eksempel 8.11 under.

Chebyshevpunkter

For å prøve å forklare hva som skjedde i figuren må vi introdusere noen polynomer.

Teorem 8.9. *Funksjonen*

$$T_n(x) = \cos(n \arccos x)$$

er et polynom når n er et naturlig tall.

Bevis. La $T_n(\theta) = \cos(n\theta)$. Da har vi

$$\begin{aligned} T_0 &= 1 \\ T_1 &= \cos\theta \\ T_2 &= \cos 2\theta = 2\cos^2\theta - 1 \end{aligned}$$

Merk at alle disse er polynomer i $\cos\theta$. Dette er en sentral observasjon, for når man gjør variabelskiftet $x = \cos\theta$ vil man få polynomer i x.

Vi fortsetter med et induksjonsbevis for at $\cos n\theta$ er et polynom i $\cos\theta$ for alle naturlige tall n. Legg sammen

$$\cos(n+1)\theta = \cos n\theta\cos\theta - \sin n\theta\sin\theta$$

og

$$\cos(n-1)\theta = \cos n\theta\cos\theta + \sin n\theta\sin\theta$$

slik at

$$\cos(n+1)\theta = 2\cos n\theta\cos\theta - \cos(n-1)\theta.$$

Dersom vi antar at $\cos n\theta$ og $\cos(n-1)\theta$ er polynomer i $\cos\theta$, må

$$2\cos n\theta\cos\theta - \cos(n-1)\theta$$

være et polynom i $\cos\theta$, og følgelig må også

$$\cos(n+1)\theta$$

være et polynom i $\cos\theta$. Siden $T_0 = 1$ og $T_1 = \cos\theta$ (og $T_2 = 2\cos^2\theta - 1$) er polynomer i $\cos\theta$, må $\cos n\theta$ være det for alle naturlige n. Dersom $\cos n\theta$ er et polynom i $\cos\theta$, må $\cos(n\arccos x)$ være et polynom i x. □

Polynomene T_n kalles *chebyshevpolynomer*. Disse kan beregnes ved rekursjonen

$$T_{n+1} = 2xT_n - T_{n-1},$$

som er likningen

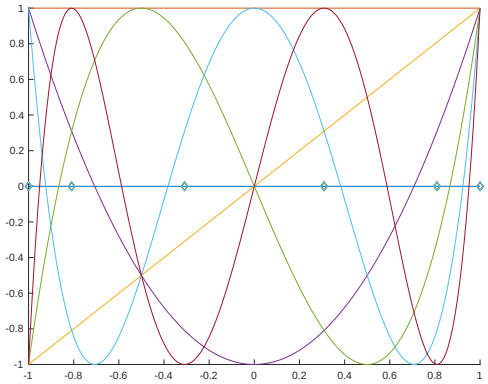
$$\cos(n+1)\theta + \cos(n-1)\theta = 2\cos n\theta\cos\theta.$$

fra forrige bevis, skrevet ut i form av T_n. De første er

$$\begin{aligned} T_0 &= 1 \\ T_1 &= x \\ T_2 &= 2x^2 - 1 \\ T_3 &= 4x^3 - 3x \\ T_4 &= 8x^4 - 8x^2 + 1 \\ &\vdots \end{aligned}$$

Gitteret i eksempel 8.8 kalles Chebyshevs ekstremalgitter, for punktene er ekstremalpunktene til et chebyshevpolynom. Det n-te ordens polynomet T_n gir opphav til et gitter med $n+1$ punkter, der $n-1$ av dem er T_ns stasjonære punkter, og to er endepunktene i intervallet. De første seks polynomene er plottet under, sammen med 6-punktsgitteret som er ekstremalpunktene til T_5. En formel for gitterpunktene er:

$$x_i = \cos\frac{\pi i}{n} \quad 0 \le i \le n.$$



Eksempel 8.10. Gitteret i figuren over er gitt ved

x_0	-1.000000000000000
x_1	-0.809016994374947
x_2	-0.309016994374947
x_3	0.309016994374947
x_4	0.809016994374947
x_5	1.000000000000000

Merk at endepunktene er en annen type ekstremalpunkter enn de andre; det er der T_5 er klippet av intervallgrensene. △

Eksempel 8.11. Hvis vi ønsker å sette opp gitteret fra T_5 på et intervall $[a, b]$, bruker vi bare formelen

$$y_i = a + (b-a)\frac{x_i + 1}{2},$$

der x_i er punktene på $[-1, 1]$ og y_i er punktene på $[a, b]$. På intervallet $[1, 4]$ blir punktene

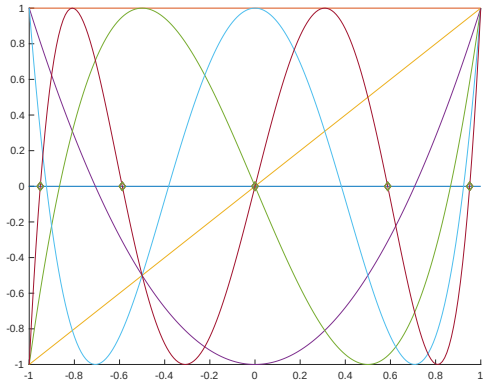
x_0	1.000000000000000
x_1	1.286474508437579
x_2	2.036474508437579
x_3	2.963525491562421
x_4	3.713525491562421
x_5	4.000000000000000

Det er viktig å forstå at det hvordan punktene er fordelt på intervallet som har noe å si for kvaliteten på interpolasjonen. △

Chebyshevpolynomenes nullpunkter gir opphav til en annen punktmengde som kalles *Chebyshevs nullpunktgitter*. Dette gitteret er på papiret enda bedre enn Chebyshevs ekstremalgitter, men noe mindre praktisk, siden det ikke inneholder endepunktene. Polynomet T_{n+1} gir opphav til et gitter med $n+1$ punkter, gitt ved

$$x_i = \cos\frac{\pi(2i+1)}{2n+2} \quad 0 \le i \le n.$$

Under er et nok et plot av de første par chebyshevpolynomene, med nullpunktgitteret til T_5. Vi tar med et teorem om interpolasjonsfeilen til Chebyshevs nullpunktgitter, men lar det stå ubevist.



Teorem 8.12. *La f være en $n+1$ ganger deriverbar funksjon. Interpolasjonsfeil for interpolanten på chebyshevs nullpunktgitter på $[a,b]$ er:*

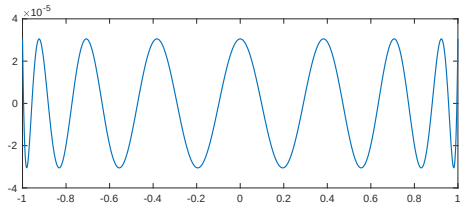
$$\max_{x\in[a,b]} |f(x) - p_n(x)| \leq \frac{(b-a)^{n+1}}{2^{2n+1}} \max_{x\in[a,b]} |f^{n+1}(x)|$$

Under er et plot av feilpolynomet

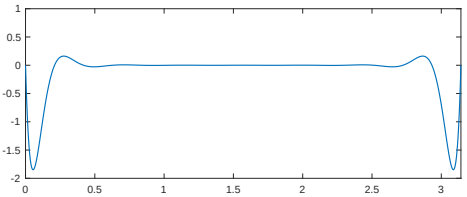
$$\prod_{k=0}^{n}(x - x_k)$$

for chebyshevs nullpunktgitter og $n = 16$. Dette polynomet har maksimalt utslag

$$\frac{(b-a)^{n+1}}{2^{2n+1}} \approx 1.525878906250000e-05.$$



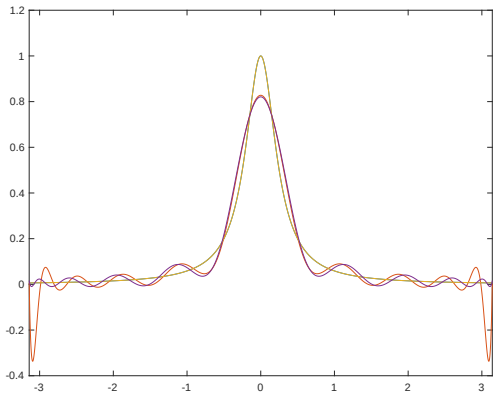
Vi tar med et plot av tilsvarende feilpolynom for ekvidistant gitter med $n = 16$.



Eksempel 8.13. Nedenfor er en figur av 20. ordens interpolanter av Runges funksjon

$$f(x) = \frac{1}{1+16x^2}$$

på ekvidistant og chebyshevgitter. Denne funksjonen er kjent for sine patologisk store n-te deriverte. Dette er en størrelse vi ikke har kontroll på, merk nok en gang hvordan feilpolynomet illustrerer hvorfor ekvidistant tinærmer vesentlig dårligere enn chebyshev i endene. △



Gauss-punkter

En punktmengde som likner på Chebyshev, og klumper seg i endene av intervallet, kalles Gauss-punkter. Akkurat som for Chebyshev, kommer disse i to varianter, som er henholdsvis null- og ekstremalpunkter til en følge av polynomer, som kalles Gauss-Legendre-polynomene. Legendre-polynomene er gitt ved rekursjonen

$$\begin{aligned} P_0 &= 1 \\ P_1 &= x \\ (n+1)P_{n+1}(x) &= (2n+1)xP_n(x) - nP_{n-1}(x). \end{aligned}$$

Vi skal ikke utlede disse, men i kapitlet om numerisk integrasjon skal vi vise en metode for å produsere nullpuntene.

Nullpunktgitrene til Legendre-polynomene kalles Gauss-Legendre-punkter. Her er en tabell med et par lavere ordens punktfordelinger på intervallet $[-1,1]$

n	x_i
2	$\pm\sqrt{\frac{1}{3}}$
3	$0, \pm\sqrt{\frac{3}{5}}$
4	$\pm\sqrt{\frac{3}{7} \pm \frac{2}{7}\sqrt{\frac{6}{5}}}$
5	$0, \pm\frac{1}{3}\sqrt{5 \pm 2\sqrt{\frac{10}{7}}}$

Gauss-Legendre-punktene er på papiret enda bedre enn chebyshevpunktene, men mindre praktisk i bruk.

Ekstremalpunktene til Legendre-polynomene kalles Gauss-Lobatto-punkter. Vi hoster opp nok en tabell med et par lavere ordens punktfordelinger på intervallet $[-1,1]$

n	x_i
3	$0,\ \pm 1$
4	$\pm\sqrt{\frac{1}{5}},\ \pm 1$
5	$0,\ \pm\sqrt{\frac{3}{7}},\ \pm 1$
6	$\pm\sqrt{\frac{1}{3} \pm \sqrt{\frac{2}{3\sqrt{7}}}},\ \pm 1$
7	$0,\ \pm\sqrt{\frac{5}{11} \pm \frac{2}{11}\sqrt{\frac{5}{3}}},\ \pm 1$

Gauss-Lobatto er på papiret noe dårligere enn Gauss-Legendre, men er noe mer praktisk i bruk, for de inneholder intervallets endepunkter. Fremdeles mindre praktisk enn Chebyshev.

Andre typer polynominterpolasjon

Til slutt kan nevnes at man trenger ikke nøye seg med å kreve at interpolanten p skal ta f sine verdier i interpolasjonspunktene. Man kan også skru opp graden på polynomet, og i tillegg kreve at p skal ha samme stigningstall som f i punktene. I dette tilfellet kalles det *Hermite-interpolasjon*.

Dersom man krever at p skal ha samme verdi som f sine $n+1$ første deriverte i et enkelt punkt, er vi tilkake i taylorpolynomene du kjenner fra M1.

Man kan også, istedet for å lage et interpolasjonspolynom som interpolerer f i alle punktene, for eksempel sortere interpolasjonspunktene i grupper på fire og fire etterfølgende punkter, interpolere hver gruppe med et tredjeordens polynom, og så sette sammen en stykkvis kontinuerlig deriverbar polynominterpolant. Dette kalles *spline-interpolasjon*.

DFT - trigonometrisk interpolasjon

DENNE DELEN ER IKKE PENSUM I TMA4125 VÅREN 2019. Vi skal ta en kjapp vending innom fourieranalysen igjen. Fourieranalyse og chebyshevinterpolasjon henger nemlig sammen, og for å se hvordan, må vi lære å interpolere med trigonometriske funksjoner.

I dette avsnittet skal vi jobbe med det ekvidistante gitteret

$$x_k = \frac{\pi k}{N} \quad \text{der} \quad -N \leq k \leq N-1$$

på intervallet $[-\pi, \pi]$. Vi ønsker å finne et trigonometrisk polynom

$$\sum_{n=-N}^{N} c_n e^{inx}$$

som interpolerer funksjonen f i gitterpunktene, altså at

$$f(x_k) = \sum_{n=-N}^{N} c_n e^{inx_k} = \sum_{n=-N}^{N} c_n e^{in\frac{\pi k}{N}}.$$

Merk. Du tenker kanskje at det hadde vært naturlig å ta med gitterpunktet $x = \pi$. Men det trigonometriske polynomet vi skal interpolere med, har fundamentalperiode 2π, så hvis vi bestemmer hva polynomet skal være i $x = -\pi$, bestemmer vi samtidig hva det skal være i $x = \pi$. Derfor er ikke $x = \pi$ med i listen over interpolasjonspunkter.

Teorem 8.14. *Det trigonometriske polynomet*

$$\sum_{n=-N}^{N} c_n e^{inx}$$

interpolerer f i punktene x_i dersom koeffisientene er gitt ved

$$c_n = \frac{1}{2N} \sum_{k=-N}^{N-1} f(x_k) e^{-inx}.$$

Bevis. Vi ganger ligningen over med e^{-imx} og summerer over alle gitterpunkter

$$\sum_{k=-N}^{N-1} f(x_k) e^{-imx} = \sum_{k=-N}^{N-1} \sum_{n=-N}^{N} c_n e^{i\frac{\pi k}{N}(n-m)}.$$

Nå bytter vi summasjonsrekkefølgen på høyre side, og får

$$\sum_{k=-N}^{N-1} f(x_k) e^{-imx} = \sum_{n=-N}^{N} c_n \sum_{k=-N}^{N-1} e^{i\frac{\pi k}{N}(n-m)}.$$

Merk at den innerste summen er en endelig geometrisk rekke med multiplikasjonsfaktor $e^{i\frac{\pi k}{N}}$. Dersom $n \neq m$ er

$$\begin{aligned}
\sum_{k=-N}^{N-1} e^{i\frac{\pi k}{N}(n-m)} &= e^{i\frac{\pi k}{N}(-N-m)} \sum_{k=0}^{2N-1} e^{i\frac{\pi k}{N}(n-m)} \\
&= e^{i\frac{\pi k}{N}(-N-m)} \sum_{k=0}^{2N-1} \left(e^{i\frac{\pi}{N}(n-m)}\right)^k \\
&= e^{i\frac{\pi k}{N}(-N-m)} \frac{1 - \left(e^{\frac{\pi i}{N}(n-m)}\right)^{2N}}{1 - e^{\frac{\pi i}{N}(n-m)}} \\
&= e^{i\frac{\pi k}{N}(-N-m)} \frac{1 - e^{2\pi i(n-m)}}{1 - e^{\frac{2\pi i}{N}(n-m)}} = 0,
\end{aligned}$$

og dersom $n = m$, er

$$\sum_{k=-N}^{N-1} e^{\frac{\pi i}{N}k(n-m)} = \sum_{k=-N}^{N-1} 1 = 2N,$$

slik at

$$\sum_{n=-N}^{N} c_n \sum_{k=-N}^{N-1} e^{\frac{2\pi i}{N}k(n-m)} = 2Nc_n.$$

Med andre ord er

$$\begin{aligned}
\sum_{k=-N}^{N-1} f(x_k) e^{-imx} &= \sum_{k=-N}^{N-1} \sum_{n=-N}^{N} c_n e^{i\frac{\pi k}{N}(n-m)} \\
&= \sum_{n=-N}^{N} c_n \sum_{k=-N}^{N-1} e^{\frac{2\pi i}{N}k(n-m)} \\
&= 2Nc_n.
\end{aligned}$$

slik at

$$c_n = \frac{1}{2N} \sum_{k=-N}^{N-1} f(x_k) e^{-inx}. \qquad \square$$

Det finnes imidlertid en teoretisk enda enklere måte å skrive opp det trigonometriske interpolasjonspolynomet på.

Teorem 8.15. *Dirichletkjernen*

$$\sum_{n=-N}^{N} e^{in(x-x_k)}$$

tar verdien 1 i x_k og 0 i de andre gitterpunkene. Interpolanten kan skrives

$$\sum_{n=-N}^{N-1} f(x_k) D_n(x - x_k).$$

Merk at interpolasjonspolynomet er skrevet som en diskret konvolusjon. I kapitlene om fourierrekker og fouriertransform har vi skrevet funksjoner som trigonometriske rekker og integraler. Dette er en diskret variant av akkurat det samme, og kalles derfor *diskret fouriertransform*. Koeffisientene c_n kan beregnes ekstremt kjapt med en berømt algoritme som går under navnet FFT - *fast fourier transform*. Både Python og Matlab har innebygde rutiner for dette.

I fouriertransformkapitlet var både transformasjonen og inverstransformasjonen gitt ved integraler. I fourierekkekapitlet var fourierkoeffisientene gitt ved integraler, og inverstransformer gitt ved en sum. I dette kapitlet var både transformasjon og inverstransformasjon gitt ved summer. Det finnes en fjerde variant der fourierkoeffisientene er gitt ved summer, mens inverstransformasjonen er et integral, men denne er utenfor vårt pensum.