Polynomisk interpolasjon

Hans Munthe-Kaas

1. jaunar 2002

Abstract

Dette notatet tar for seg interpolasjon med polynomer. Notatet er ment som et tillegg til læreboken i I162, og forsøker å framstille dette stoffet klarere og mer utfyllende enn i læreboken.

1 Vandermonde likningssystemer

Vi har gitt $n+1$ x-verdier $x_0, x_1, \ldots, x_n$ og $n+1$ y-verdier $f_0, \ldots, f_n$. Disse punktene kan enten være målepunkter fra et eksperiment, eller punkter som kommer fra beregning av en gitt funksjon, $f_i = f(x_i)$. *Interpolasjonsproblemet* består i at vi ønsker å finne et n-te grads $p(x)$ som går gjennom alle punktene. Vi skal se at dersom alle x_i er forskjellige så finnes nøyaktig ett polynom av grad n som løser dette problemet. De ulike måtene å løse interpolasjonsproblemet gir altså det samme svaret, men i ulik form.

En enkel måte å finne interpolasjonsproblemet er å sette opp et såkalt *Vandermonde likningssystem*. Hvis vi skriver

$$p(x) = p_0 + p_1 x + \ldots + p_n x^n$$

vil $p(x_i) = f_i$ for $i = 0, 1, \ldots, n$ gi likningene:

$$\begin{aligned} p_0 + x_0 p_1 + x_0^2 p_1 + \ldots + x_0^n p_n &= f_0 \\ p_0 + x_1 p_1 + x_1^2 p_1 + \ldots + x_1^n p_n &= f_1 \\ \vdots &= \vdots \\ p_0 + x_n p_1 + x_n^2 p_1 + \ldots + x_n^n p_n &= f_n \end{aligned}$$

som i matriseform kan skrives som:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & & & & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix} \begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_n \end{pmatrix} = \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \end{pmatrix}. \tag{1}$$

Koeffisient matrisen i dette systemet kalles Vandermonde matrisen for interpolasjonspunktene $x_0, \ldots, x_n$. I Matlab har vi en rutine `vander` som lager en slik matrise, bortsett fra at kolonnene kommer i motsatt rekkefølge. For å få matrisen nøyaktig slik som dette må vi speile den med kommandoen `fliplr`. Vi kan altså finne koeffisientene til $p(x)$ med kommandoen:

`p = fliplr(vander(x))\f`.

Her er `x`, `f` og `p` alle $n \times 1$ vektorer. Siden alle vektorer i Matlab starter med indeksen 1, har vi: `x(i)`$= x_{i-1}$, `f(i)`$= f_{i-1}$ og `p(i)`$= p_{i-1}$.

Vi bør vel her også bemerke at dersom vi hadde utelatt `fliplr`, ville vi fått de samme `p` verdiene, men i motsatt rekkefølge, `p = [`$p_n, \ldots, p_1, p_0$`]'`. Når Matlab i ulike sammenhenger representerer et n'te grads polynom med en $n+1$ vektor, så legges koeffisienten foran x^n i første posisjon, mens konstant leddet legges i siste posisjon. Dette er grunnen til at Vandermonde matrisen lages på en litt uvanlig måte.

2 Interpolasjon i Lagrange form

Lagrange løsningen på interpolasjonsproblemet er en form som ikke krever løsning av et likningssystem. La oss for enkelhets skyld se på tilfellet der $n = 3$. La polynomene $l_i(x)$ for $i = 0, 1, 2, 3$ være gitt som:

$$\begin{aligned} l_0(x) &= (x-x_1)(x-x_2)(x-x_3)/(x_0-x_1)(x_0-x_2)(x_0-x_3) \\ l_1(x) &= (x-x_0)(x-x_2)(x-x_3)/(x_1-x_0)(x_1-x_2)(x_0-x_3) \\ l_2(x) &= (x-x_0)(x-x_1)(x-x_3)/(x_2-x_0)(x_2-x_1)(x_2-x_3) \\ l_3(x) &= (x-x_0)(x-x_1)(x-x_2)/(x_3-x_0)(x_3-x_1)(x_3-x_2) \end{aligned}$$

''Vi ser lett at" (som det heter i matematisk sjargong) disse såkalte Lagrange basis polynomene tilfredstiller

$$l_i(x_i) = 1, \;\; l_i(x_j) = 0 \quad \text{for } j \neq i.$$

Dermed ser vi ved innsetting at polynomet

$$p(x) = l_0(x)f_0 + l_1(x)f_1 + l_2(x)f_2 + l_3(x)f_3$$

løser interpolasjonsproblemet. Vi har f.eks.

$$p(x_2) = l_0(x_2)f_0 + l_1(x_2)f_1 + l_2(x_2)f_2 + l_3(x_2)f_3 = 0 \cdot f_0 + 0 \cdot f_1 + 1 \cdot f_2 + 0 \cdot f_3 = f_2.$$

Denne metoden kan selvsagt benyttes uansett n. Hvis vi lar tegnet $\prod$ betegne produktet av mange ledd, og $\sum$ betegne summasjon, får vi den generelle Lagrange formen skrevet som:

$$l_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

og dermed

$$p(x) = \sum_{i=0}^{n} l_i(x) f_i = \sum_{i=0}^{n} \left(\prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \right) f_i. \tag{2}$$

Dette er en eksplisitt formel som løser interpolasjonsproblemet. Dermed har vi vist halvparten av følgende resultat:

Teorem 1 *Gitt $n+1$ punkt $(x_0, f_0), (x_1, f_1), \ldots, (x_n, f_n)$. Dersom alle $\{x_i\}_{i=0}^n$ er forskjellige, finnes det ett og bare ett n-te grads polynom som interpolerer i alle punktene.*

For å vise at det kun er ett interpolerende polynom, kan vi tenke oss at vi har *to* interpolerende polynom, $p(x)$ og $q(x)$, begge av grad n. Da må $r(x) = p(x) - q(x)$ være et polynom av grad høyst n som har $n+1$ nullpunkt, i alle x_i. Med unntak av $r(x) = 0$, finnes det ingen slike polynom. Dermed må $p(x) = q(x)$, og vi har vist at det er kun ett interpolerende polynom.

3 Dividerte differanser

Dividerte differanser er en rekursiv måte å bygge opp det interpolerende polynom der en kan ta med fler og fler punkt, og stoppe når en er fornøyd med resultatet. Dette er også ofte den billigste måten å lage interpolasjonspolynomet. Vi skal også se at denne metoden kan brukes til mange andre ting, slik som å regne ut deriverte til interpolasjonspolynomet, eller finne polynom som interpolerer verdien til en funksjon i noen punkter og den deriverte til funksjonen i andre punkt.

Prinsippet for dividete differanser er at vi bygger opp en tabell som følger (igjen med $n = 3$ som eksempel):

$$
\begin{array}{lcl}
x_0 & - & f[x_0] \\
 & & \quad\searrow \\
 & & \quad\nearrow f[x_0,x_1] \\
x_1 & - & f[x_1] \qquad\qquad\searrow \\
 & & \quad\searrow \qquad\qquad\nearrow f[x_0,x_1,x_2] \\
 & & \quad\nearrow f[x_1,x_2] \qquad\qquad\searrow \\
x_2 & - & f[x_2] \qquad\qquad\nearrow f[x_1,x_2,x_3] \\
 & & \quad\searrow \\
 & & \quad\nearrow f[x_2,x_3] \\
x_3 & - & f[x_3]
\end{array}
$$

Interpolasjonspunktene føres opp i de to første kolonnene, og de øvrige kolonnene regnes ut etter rekursjonsformelen:

$$
\begin{aligned}
f[x_i] &= f_i = f(x_i) \\
f[x_i,x_{i+1}] &= \frac{f[x_{i+1}]-f[x_i]}{x_{i+1}-x_i} \\
&\vdots \qquad\qquad \vdots \\
f[x_i,x_{i+1},\ldots,x_{i+k}] &= \frac{f[x_{i+1},x_{i+2},\ldots,x_{i+k}]-f[x_i,x_{i+1},\ldots,x_{i+k-1}]}{x_{i+k}-x_i}
\end{aligned}
\tag{3}
$$

Newtons interpolasjonsformel uttrykkes ved tallene i den øverste NV-SØ diagonalen i differansetabellen som

$$
\begin{aligned}
p(x) &= f[x_0]+(x-x_0)f[x_0,x_1]+(x-x_0)(x-x_1)f[x_0,x_1,x_2]+\cdots \\
&\quad +(x-x_0)(x-x_1)\cdots(x-x_{n-1})f[x_0,x_1,\ldots,x_n].
\end{aligned}
\tag{4}
$$

Vi vil nå vise at denne formelen er korrekt[1]. Den holder opplagt for $n = 1$, og for å vise at den holder generelt bruker vi *induksjon*, dvs. vi antar at den er korrekt opp til $n = k-1$ og vil fra dette vise at den er korrekt for $n = k$. Hvis formelen er korrekt opp til $n = k-1$, kan vi danne to polynomer $p_{k-1}(x)$ som interpolerer i punktene $x_0, x_1, \ldots, x_{k-1}$ og $q_{k-1}(x)$ som interpolerer i $x_1, x_2, \ldots, x_k$. Disse finner vi ved å lage de $k-1$ første leddene i de to øverste diagonalene i differansetabellen. Nå skriver vi polynomet som interpolerer i punktene $x_0, x_1, \ldots, x_k$ som:

$$p_k(x) = p_{k-1}(x) + c_k(x-x_0)(x-x_1)\cdots(x-x_k),$$

der c_k er en konstant som vi ennå ikke kjenner. Dette må være ok, siden det siste leddet er et polynom av grad k som har nullpunkt i $x_0, x_1, \ldots, x_{k-1}$. Dermed ødelegger ikke dette leddet noen av interpolasjonspunktene til $p_{k-1}(x)$. Dessuten kan vi få $p_k(x)$ til å oppnå en hvilket som helst verdi i punktet x_k ved å justere c_k. For å finne den riktige c_k, merker vi oss at c_k er koeffisienten foran leddet av høyest grad,

$$p_k(x) = c_k x^k + \text{ledd av orden } x^{k-1} \text{ og lavere.}$$

Det litt vanskelige tricket i dette beviset består i at vi nå også skriver $p_k(x)$ som

$$p_k(x) = \frac{x-x_0}{x_k-x_0}q_{k-1}(x) - \frac{x-x_k}{x_k-x_0}p_{k-1}(x). \tag{5}$$

Vi kan verifisere at dette også er korrekt ved å sette inn alle punktene $x_0, \ldots, x_k$ og se at dette polynomet faktisk interpolerer overalt. Dette kan dere sjekke selv. (Husk at $p_{k-1}(x_i) = f(i)$ for $i = 0, 1, \ldots, k-1$

[1]Beviset som følger kan hoppes over ved første gangs lesning, men å forstå slike bevis øker forståelsen og gir god trening i bevisførsel.

og $q_{k-1}(x_i) = f(i)$ for $i = 1, 2, \ldots, k$). Nå finner vi tilslutt c_k fra (5) ved å regne ut leddet av høyest grad. Vi vet at leddene av høyest grad i $p_{k-1}(x)$ og $q_{k-1}(x)$ er henholdsvis $f[x_0, x_1, \ldots, x_{k-1}]x^{k-1}$ og $f[x_1, x_2, \ldots, x_k]x^{k-1}$. Dermed gir (5) at høyeste ledd i $p_k(x)$ blir:

$$c_k x^k = \frac{f[x_1, x_2, \ldots, x_k]x^k}{x_k - x_0} - \frac{f[x_0, x_1, \ldots, x_{k-1}]x^k}{x_k - x_0}$$

hvilket viser at

$$c_k = f[x_0, x_1, \ldots, x_k] = \frac{f[x_1, x_{i+2}, \ldots, x_k] - f[x_0, x_1, \ldots, x_{k-1}]}{x_k - x_0}.$$

Dette viser at Newton formen er korrekt også for $n = k$, og induksjonsbeviset er fullført.

3.1 Interpolasjonsfeilen

Vi vil nå se hva feilen i polynomisk interpolasjon blir i et gitt punkt x dersom dataene f_i kommer fra en funksjon $f(x)$ som kan deriveres $n+1$ ganger. Det letteste er å benytte Newtons formel, og se på $(x, f(x))$ som et siste interpolasjonspunkt! (Et trick som kanskje synes rart første gang). Newtons formel gir da:

$$\begin{aligned} f(x) = p_{n+1}(x) &= f[x_0] + \cdots + (x - x_0) \cdots (x - x_{n-1}) f[x_0, \ldots, x_n] + \\ &= (x - x_0) \cdots (x - x_n) f[x_0, \ldots, x_n, x] \end{aligned}$$

Dette gir feilen som siste ledd:

$$e(x) = f(x) - p_n(x) = (x - x_0)(x - x_1) \cdots (x - x_n) f[x_0, x_1, \ldots, x_n, x]. \tag{6}$$

Det kan vises (men vi utelater beviset her) at dersom f er $n + 1$ ganger deriverbar så finnes det alltid et punkt $\xi \in (x_0, x_1, \ldots, x_n, x)$ slik at

$$f[x_0, x_1, \ldots, x_n, x] = \frac{1}{(n+1)!} \frac{d^{n+1} f(\xi)}{dx^{n+1}}. \tag{7}$$

Dette resultatet viser at høyere ordens differanser vanligvis vil går mot null når vi tar med mange punkt.

Problemet med feil i polynomisk interpolasjon er ikke knyttet til denne faktoren, men heller til polynomet $(x - x_0)(x - x_1) \cdots (x - x_n)$. La oss plotte dette for tilfellet $n = 10$, $x_i = 0, 1, \ldots, 10$. Kommandoene under gir Figur 1.

```
xi = 0:10;
p = poly(xi);
xval = 0:0.1:10;
pval = polyval(p,xval);
plot(xval,pval)
hold on
plot(xi,zeros(size(xi)),'o')
title('Feilpolynom ved n=11, ekvidistante xi.');
```

Vi bør vel her også nevne at dersom vi har mulighet til å velge interpolasjonspunkter, kan vi faktisk klare å interpolere nøyaktig med høyereordens polynomer. Tricket er å legge interpolasjonspunktene tettere mot kantene. Dette henger sammen med noe som heter Chebychev polynomer som er pensum i kursens I161 og I264. Dette er viktige ting i mange deler av numerisk analyse. Figur 2 viser feilen ved interpolasjon med 11. grads polynom når vi har valgt disse interpolasjonspunktene. Maksimalfeilen er blitt ca. 100 ganger mindre, redusert fra ca. $4 \cdot 10^6 f^{(11)}(\xi)/11! \approx 0.1 \cdot f^{(11)}(\xi)$ til omlag $4 \cdot 10^4 f^{(11)}(\xi)/11! \approx 0.001 \cdot f^{(11)}(\xi)$.

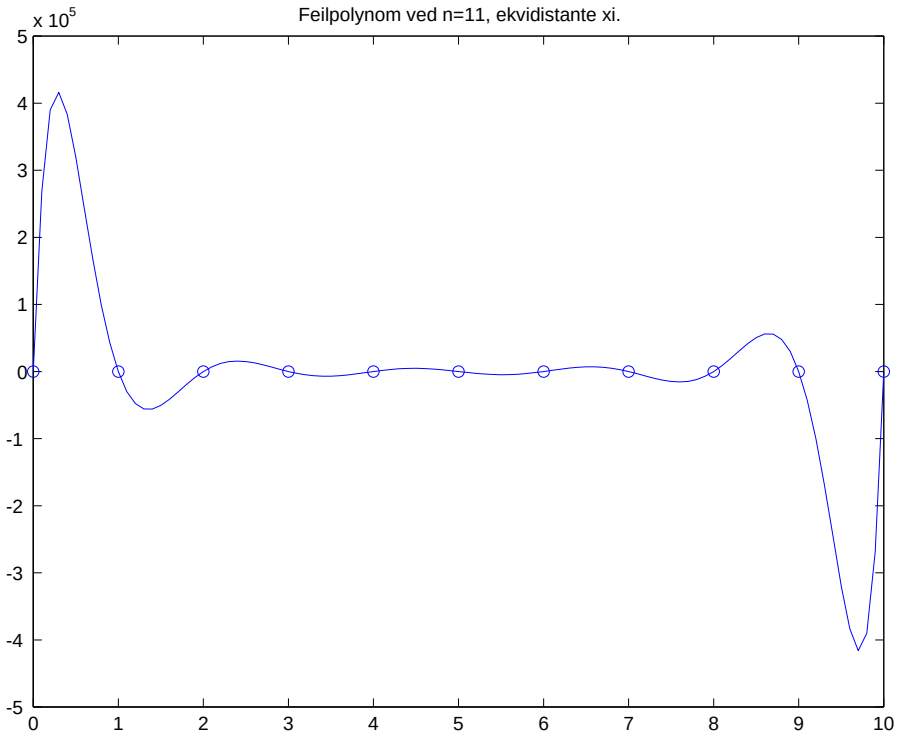

Figure 1: Feilpolynom, $n = 11$, ekvidistante punkt.

3.2 Hermite interpolasjon

Målet i dette delkapittelet er å regne ut et interpolerende polynom i tilfellet der en vet hvilke punkter polynomet skal gå gjennom noen steder, og en vet hva den deriverte skal være andre steder. Vi skal ikke diskutere de eksakte betingelsene for eksistens og entydighet av løsningen, men det er opplagt at hvis alle betingelsene vi pålegger dreier seg om deriverte og evt. høyere deriverte, kan ikke problemet ha entydig lœsning, da konstantleddet til polynomet i så fall kan velges fritt. Så vi antar at minst en av punktene gir verdien til polynomet.

La oss ta utgangspunkt i likning (7), skrevet som

$$f[x_0, x_1, \ldots, x_k] = \frac{1}{k!}\frac{d^k f(\xi)}{dx^k} \quad \text{for } \xi \in (x_0, x_k).$$

La nå alle punktene x_i nærme seg et gitt punkt $\tilde{x}$. Da må $f[x_0, \ldots, x_n] \to d^k f(\tilde{x})/k! dx^k$. I tilfellet der samme tall $\tilde{x}$ inngår $k+1$ ganger i differansen velger vi derfor å *definere* dividerte differanser som en derivasjon:

$$f[\tilde{x}, \tilde{x}, \ldots, \tilde{x}] \stackrel{\text{def}}{=} \frac{1}{k!}\frac{d^k f(\tilde{x})}{dx^k} \tag{8}$$

Ideen er nå at dersom vi i et punkt $\tilde{x}$ kjenner den k'te deriverte av f, så får vi dette inn i differansetabellen ved å gjenta samme punkt $k+1$ ganger. Som eksempel på denne metoden kan vi finne et polynom $p_4(x)$ som har skal interpolere $f(x)$ med følgende betingelser:

$$f(0) = 0, f'(0) = 1, f(1) = 2, f'(1) = 3, f''(1)/2! = 4.$$

Vi setter opp differansetabellen. (Jeg har satt en strek under alle tall vi puttet inn i starten. De andre er

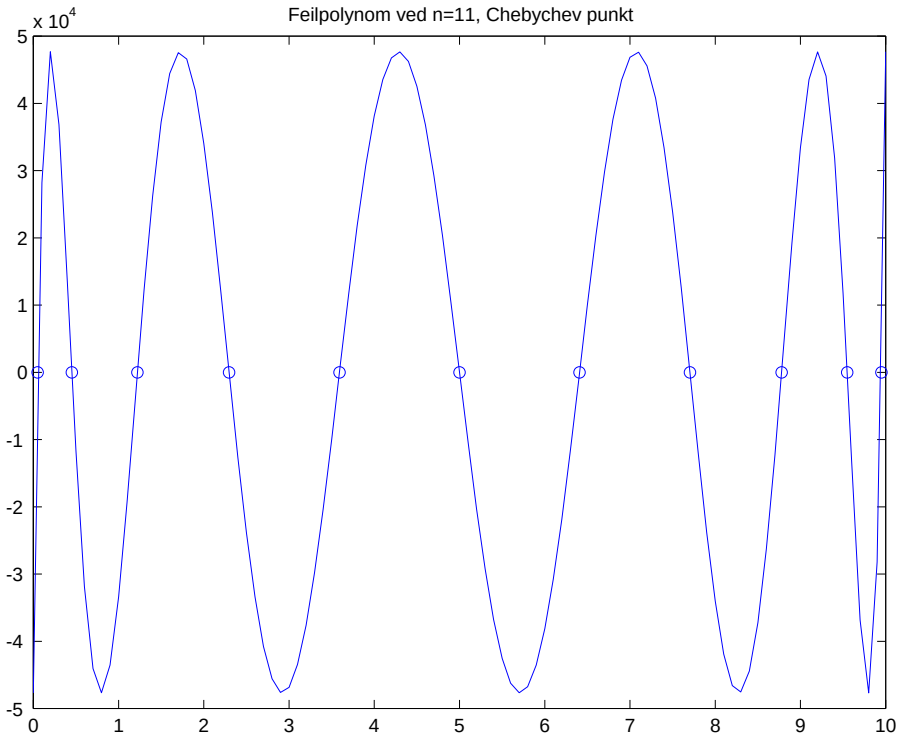

Figure 2: Feilpolynom, $n = 11$, Chebychev punkt.

regnet ut.)

$$
\begin{array}{ccccccc}
0 & - & \underline{0} & & & & \\
 & & & \underline{1} & & & \\
0 & - & \underline{0} & & 1 & & \\
 & & & 2 & & 0 & \\
1 & - & \underline{2} & & 1 & & 3 \\
 & & & \underline{3} & & 3 & \\
1 & - & \underline{2} & & \underline{4} & & \\
 & & & \underline{3} & & & \\
1 & - & \underline{2} & & & &
\end{array}
$$

Nå kan vi på samme måte som før lese av interpolasjonspolynomet:

$$
\begin{aligned}
p_4(x) &= 0 + (x-0)\cdot 1 + (x-0)^2 \cdot 1 + (x-0)^2(x-1)\cdot 0 + (x-0)^2(x-1)^2 \cdot 3 = \\
&= x + x^2 + 3x^2(x-1)^2.
\end{aligned}
$$

3.3 Baklengsregning i differansetabellen

Dette at vi kan få fram deriverte ved å gjenta samme punkt mange ganger i differansetabellen er kjekt også for å *regne ut* hva de (høyereordens) deriverte til et interpolasjonspolynom er i et gitt punkt. Dette er en utregning som ofte er vanskelig å få korrekt dersom vi gjør det rett fram ved å derivere polynomet gitt i Newton form.

Teknikken vi skal bruke er basert på følgende observasjon:

Lemma 1 *Dersom $f(x) = p_n(x)$, et n'te grads polynom, så er $f[x_0, x_1, \ldots, x_n]$ konstant, uavhengig av $\{x_i\}$.*

Bevis: Dette følger direkte fra likning (7). ♡

For et interpolerende polynom framkommet ved en dividert differansetabell kan en derfor regne ut deriverte ved å utvide tabellen oppover med kopier av punktet der en vil derivere, og *regne baklengs* ved å starte med å kopiere høyre punkt i tabellen oppover. La oss illustrere med et eksempel der vi først har funnet polynomet $p(x)$ av grad 3 som interpolerer i $(1,1)$, $(2,5)$, $(3,14)$, $(4,30)$. I differansetabellen nedenfor er alle tallene uten nedre indeks de vi har regnet ut på dette stadiet. Vi har altså funnet polynomet

$$p(x) = 1 + 4(x-1) + 5(x-1)(x-2) + 2(x-1)(x-2)(x-3).$$

Nå ønsker vi å finne $p(0)$, $p'(0)$, $p''(0)$ og $p'''(0)$. Vi gjør dette ved å kopiere opp høyre element i tabellen og regne baklengs. De nedre indeksene på tallene indikerer hvilken rekkefølge vi har funnet dem.

$$\begin{array}{lllllll}
0 & - & -5_3 & & & & \\
 & & & 11_5 & & & \\
0 & - & -5_3 & & -7_6 & & \\
 & & & 11_5 & & 2 & \\
0 & - & -5_3 & & -5_4 & & \\
 & & & 6_2 & & 2 & \\
1 & - & 1 & & -1_1 & & \\
 & & & 4 & & 2 & \\
2 & - & 5 & & 5 & & \\
 & & & 14 & & 2 & \\
3 & - & 19 & & 11 & & \\
 & & & 36 & & & \\
4 & - & 55 & & & &
\end{array}$$

Herfra leser vi nå av: $p(0) = -5$, $p'(0) = 11$, $p''(0)/2 = -7$ og $p'''(0)/3! = 2$. Legg også merke til at vi fra den øverste diagonalen kan lese av polynomet i den vanlige potensformen:

$$p(x) = -5 + 11x - 7x^2 + 2x^3.$$

Denne teknikken er faktisk den billigste måten å regne om et polynom fra Newton form til potens form.

3.3.1 Taylors formel med feilledd

Hvis vi velger $n+1$ x-verdier i differansetabellen *alle* være samme tall x_0, så blir interpolasjonspolynomet med feilledd gitt som:

$$\begin{aligned}
f(x) &= f[x_0] + (x-x_0)f[x_0,x_0] + (x-x_0)^2 f[x_0,x_0,x_0] + \cdots \\
&\quad + (x-x_0)^n f[x_0,\ldots,x_0] + (x-x_0)^{n+1} f[x_0,\ldots,x_0,x] = \\
&= f(x_0) + \frac{(x-x_0)}{1!}f'(x_0) + \frac{(x-x_0)^2}{2!}f''(x_0) + \\
&\quad + \frac{(x-x_0)^n}{n!}f^{(n)}(x_0) + \frac{(x-x_0)^{n+1}}{(n+1)!}f^{(n+1)}(\xi), \quad \xi \in (x_0,x).
\end{aligned} \tag{9}$$

Vi ser altså at Taylors formel med feilledd faktisk er et spesialtilfelle av den generelle Newton formen for interpolasjonspolynomet!

3.4 Ekvidistante interpolasjonspunkt

Vi avslutter dette notatet med å diskutere tilfellet der avstanden mellom x_i og x_{i+1} er konstant h for alle i. Vi har altså $x_i = x_0 + ih$. I dette tilfellet blir mange av uttrykkene spesielt enkle, men teorien er i prinsippet den samme som beskrevet over. La oss introdusere *forover differanse operatoren* Δ som virker på en sekvens $\{f_i\}$ ved

$$\Delta f_i = f_{i+1} - fi. \tag{10}$$

Høyere differanser får vi ved å anvende Δ flere ganger:

$$\Delta^2 f_i = \Delta(\Delta f_i) = \Delta(f_{i+1} - fi) = f_{i+2} - f_{i+1} - f_{i+1} + f_i = f_{i+2} - 2f_{i+1} + f_i.$$

Vi ser at forskjellen mellom en differanse og en dividert differanse er kun at i k'te trinn av dividerte differanser deler vi med kh. Dermed har vi

$$\Delta^k f_i = h^k k! f[x_i, x_{i+1}, \ldots, x_{i+k}]. \tag{11}$$

Det er nyttig å introdusere s ved

$$s = \frac{x - x_0}{h} \tag{12}$$

som gir $(x - x_k) = h(s - k)$. Vi kan nå sette inn i likning (4) og få den såkalte Newton–Gregory interpolasjonsformelen

$$\begin{aligned} p(x) \quad = \quad & p(x_0 + sh) = f(x_0) + s\Delta f(x_0) + \frac{s(s-1)}{2!}\Delta^2 f(x_0) + \frac{s(s-1)(s-2)}{3!}\Delta^3 f(x_0) + \ldots \\ & + \frac{s(s-1)\cdots(s-n+1)}{n!}\Delta^n f(x_0). \end{aligned} \tag{13}$$

3.4.1 Lek med symboler

Tilslutt vil vi leke oss litt med symboler, og se at denne formelen faktisk kan utledes på en ganske annen måte. De som ikke liker denne typen lek kan uten problemer la være å lese dette avsnittet. Vi vet at $e^x = 1 + x + x^2/2 + \ldots + x^i/i! + \ldots$. Vi kan da kanskje si at:

$$e^{h\frac{d}{dx}} = I + h\frac{d}{dx} + \frac{h^2}{2!}\frac{d^2}{dx^2} + \ldots + \frac{h^i}{i!}\frac{d^i}{dx^i} + \ldots$$

Fra Taylors formel får vi

$$e^{h\frac{d}{dx}} f(x_0) = f(x_0) + h\frac{d}{dx}f(x_0) + \frac{h^2}{2!}\frac{d^2}{dx^2}f(x_0) + \ldots + \frac{h^i}{i!}\frac{d^i}{dx^i}f(x_0) + \ldots = f(x_0 + h).$$

Hva er så Δ?

$$\Delta f(x_0) = f(x_0 + h) - f(x_0) = \left(e^{h\frac{d}{dx}} - I\right) f(x_0).$$

Vi har altså $e^{h\frac{d}{dx}} = I + \Delta$. Men da må

$$e^{sh\frac{d}{dx}} = \left(e^{h\frac{d}{dx}}\right)^s = (I + \Delta)^s.$$

Dermed har vi

$$f(x_0 + sh) = e^{sh\frac{d}{dx}} f(x_0) = (I + \Delta)^s f(x_0).$$

For å gi dette en form som kan brukes i beregninger benytter vi den såkalte *binomial formelen*

$$(a+b)^s = a^s + sa^{s-1}b + \frac{s(s-1)}{2!}a^{s-2}b^2 + \ldots + \frac{s(s-1)\cdots(s-i+1}{i!}a^{s-i}b^i + \ldots,$$

som faktisk gjelder også når s ikke er heltall, dersom vi fortsetter i det uendelige. Dette gir oss Newton–Gregory's formel dersom vi lar $a = I$ og $b = \Delta$.

Koeffisientene

$$\frac{s(s-1)\cdots(s-i+1)}{i!} \equiv \binom{s}{i}$$

kalles *binomial koeffisienter*.

♣ ♢ ♡ ♠ ♣ ♢ ♡ ♠ ♣ ♢ ♡ ♠ ♣ ♢ ♡ ♠ ♣ ♢ ♡ ♠