

# KAPITTEL 9

## Approksimasjon av funksjoner

En grunnleggende teknikk som ofte brukes i ulike deler av matematikk og anvendelser er å tilnærme eller approksimere et objekt med et annet. Som regel er objektet som skal tilnærmes mer komplisert enn objektet som det tilnærmes med. På denne måten får vi et enklere objekt å håndtere på bekostning av at vi må akseptere en, vanligvis liten, feil. I dette kapitlet skal vi se litt på approksimasjon av funksjoner, for det meste med polynomer. Polynomer er populære som approksimasjoner siden verdien av et polynom i et punkt kan bestemmes eksakt (hvis vi ser bort fra avrundingsfeil) ved hjelp av de elementære operasjonene addisjon og multiplikasjon av tall. Dette gjør at polynomer er godt tilpasset beregninger på datamaskin. Mot slutten av kapitlet skal vi også se litt på approksimasjon med stykkevise polynomer. Dette er funksjoner som består av polynombiter som er lenket sammen til mer kompliserte funksjoner. Stykkevise polynomer egner seg også godt til beregninger på datamaskin siden den eneste operasjonen vi trenger i tillegg til polynomoperasjonene er å avgjøre hvilket polynombit vi er på. Dette gjør vi med en test som datamaskiner også håndterer effektivt og greit.

Vi skal først se litt raskt på Taylor polynomer og noen aspekter ved dem som ikke vanligvis dekkes i tradisjonelle matematikkbøker. Deretter skal vi så vidt ta for oss en annen tilnæringsmetode, nemlig interpolasjon. Et viktig poeng i dette kapitlet er at det er viktig hvordan vi skriver polynomene våre, og vi skal se at det kan være spesielt hendig å skrive polynomer ved hjelp av de såkalte Bernstein-polynomene. Dette leder oss over i såkalte Bezier-kurver som er mye brukt i grafikk. Mot slutten skal vi se mer på stykkevise polynomer som egner seg svært godt til å spalte opp en funksjon i komponenter med forskjellig oppløsning eller skala, såkalt *flerskalaoppløsning*. Som vi skal se egner dette seg godt for representasjon og kompresjon av lyd.

### 9.1 Taylor polynomer

Taylor<sup>1</sup>-polynomer er polynomer som tilnærmer en gitt funksjon godt i nærheten av et punkt  $x = a$ . Vi oppnår dette ved å kreve at Taylor-polynomet skal reprodusere så mange som mulig av funksjonens deriverte i  $a$ .

---

<sup>1</sup>Etter Brooke Taylor (1685–1731), engelsk matematiker.

### 9.1.1 Konstruksjon av Taylor-polynomer

Funksjonen vi skal tilnærme kaller vi  $f$ , og vi antar at  $f$  kan deriveres så mange ganger vi ønsker; en kort og vanlig måte å beskrive denne antagelsen på er å si at  $f$  skal være glatt. For å kunne bestemme Taylor-polynomer trenger vi et tall  $a$  i definisjonsområdet til  $f$ . Det enkleste Taylor-polynomet er av grad 0, det er altså konstant, og framkommer ved at vi konstruerer det enklest mulige polynomet som reproduserer verdien til  $f$  i  $a$ . Hvis vi betegner dette polynomet med  $p_0$  så har vi altså

$$p_0(x) = f(a).$$

Dette er en svært god tilnærming til  $f$  i  $x = a$ , men for de fleste funksjoner blir kvaliteten fort dårlig når vi fjerner oss fra dette punktet.

Neste steg er å forsøke å finne en enkel tilnærming til  $f$  som tar med seg både verdien til  $f$  og dens førstederiverte  $f'$  i  $a$ . Siden vi har to betingelser, er det rimelig å tro at vi trenger et polynom med to frie koeffisienter for å få til dette. Vi prøver derfor med et førstegradspolynom  $p_1(x) = b_0 + b_1x$ . De to betingelsene gir oss to ligninger i de to ukjente  $b_0$  og  $b_1$ ,

$$\begin{aligned} f(a) &= p_1(a) = b_0 + b_1a, \\ f'(a) &= p_1'(a) = b_1. \end{aligned}$$

Dette gir  $b_1 = f'(a)$  og  $b_0 = f(a) - af'(a)$  så  $p_1$  kan skrives som

$$p_1(x) = b_0 + b_1x = f(a) - af'(a) + xf'(a) = f(a) + (x - a)f'(a).$$

Siden  $p_1$  er en rett linje som har samme verdi og derivert som  $f$  i  $a$  er  $p_1$  tangenten til  $f$  i dette punktet.

La oss også ta bryderiet med å konstruere  $p_2 = b_0 + b_1x + b_2x^2$ , et andregradspolynom som tilnærmer  $f$  best mulig i  $a$ . Siden vi har tre frie koeffisienter er det rimelig å tvinge  $p_2$  til å ha samme verdi, førstederivert og andrederivert som  $f$  i  $a$ . Dette gir betingelsene

$$f(a) = p_2(a) = b_0 + b_1a + b_2a^2, \quad (9.1)$$

$$f'(a) = p_2'(a) = b_1 + 2b_2a, \quad (9.2)$$

$$f''(a) = p_2''(a) = 2b_2. \quad (9.3)$$

Vi legger merke til at dette ligningssystemet har en spesiell struktur som gjør det lett å løse. Fra (9.3) kan vi finne  $b_2$ , setter vi resultatet inn i (9.2) kan vi finne  $b_1$  og setter vi begge disse resultatene inn i (9.1) kan vi finne  $b_0$ . Løsningen vi finner er

$$\begin{aligned} b_2 &= \frac{f''(a)}{2}, \\ b_1 &= f'(a) - af''(a), \\ b_0 &= f(a) - af'(a) + \frac{a^2 f''(a)}{2}. \end{aligned}$$

Setter vi dette inn i uttrykket for  $p_2$  og forenkler så får vi

$$p_2(x) = f(a) + (x - a)f'(a) + \frac{1}{2}(x - a)^2 f''(a). \quad (9.4)$$

Fra dette er det ikke så vanskelig å gjette den generelle formen på Taylor-polynomet av grad  $n$ . Men før vi skriver opp dette, la oss reflektere litt over utledningene våre. Hvis vi ser på andregradstilfellet så begynte vi med å skrive opp det generelle polynomet av grad 2

$$p_2(x) = b_0 + b_1x + b_2x^2. \quad (9.5)$$

Vi skrev deretter opp betingelsene våre som ga oss et ligningssystem som var lett å løse. Etter noen forenklinger kom vi så fram til (9.15). Legg merke til at om vi til å begynne med hadde antatt at  $p_2$  var på formen

$$p_2(x) = c_0 + c_1(x - a) + c_2(x - a)^2 \quad (9.6)$$

så ville ligningssystemet vårt blitt

$$f(a) = p(a) = c_0, \quad f'(a) = p'(a) = c_1, \quad f''(a) = p''(a) = 2c_2$$

som har løsningen  $c_0 = f(a)$ ,  $c_1 = f'(a)$  og  $c_2 = f''(a)/2$ . Dette gir oss kjapt og greit (9.15) uten behov for noen forenklinger.

Dette illustrerer et viktig poeng: Et polynom kan skrives på mange ekvivalente måter, og ved å velge en form tilpasset problemet vi skal løse kan vi ofte forenkle beregningene våre. Et annet aspekt av dette er at ulike polynomformer kan være mer eller mindre følsomme for avrundingsfeil. Dessverre er det slik at (9.5), som er den vanligste måten å skrive polynomer på, kan være svært følsom for avrundingsfeil om ikke  $x$  ligger nær 0, særlig når graden er høy. Vi skal se nærmere på dette i seksjon 9.3 under.

Når vi nå har funnet en måte å skrive polynomer på som er godt tilpasset problemet vi skal løse, er det på tide å angripe det generelle tilfellet. Vi ønsker å danne et Taylor-polynom  $p_n$  av grad  $n$  som er en best mulig tilnærming til  $f$  i punktet  $x = a$ . Vi har lært av erfaringene over og skriver  $p_n$  på formen

$$p_n(x) = c_0 + c_1(x - a) + \cdots + c_n(x - a)^n.$$

Siden  $p_n$  har  $n + 1$  frie koeffisienter så forventer vi å kunne legge  $n + 1$  betingelser på  $f$ , nemlig betingelsene  $f(a) = p(a)$ ,  $f'(a) = p'_n(a)$ ,  $\dots$ ,  $f^{(n)}(a) = p_n^{(n)}(a)$ . Setter vi inn uttrykket for  $p_n$  får vi ligningssystemet

$$f(a) = c_0, \quad f'(a) = c_1, \quad f''(a) = 2c_2, \quad f'''(a) = 6c_3, \quad \dots, \quad f^{(n)}(a) = n!c_n$$

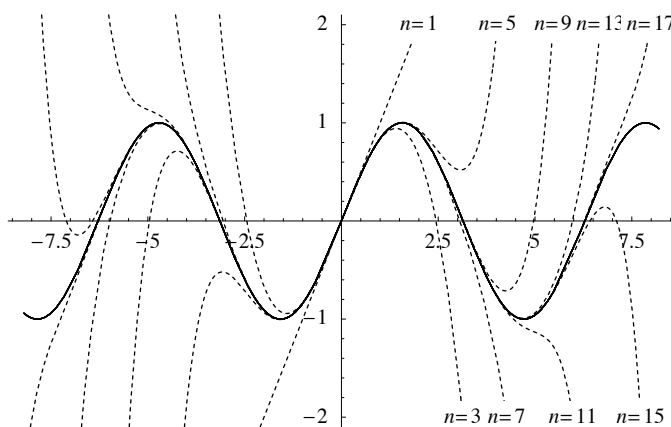
som har løsningen

$$c_0 = f(a), \quad c_1 = f'(a), \quad c_2 = \frac{f''(a)}{2}, \quad c_3 = \frac{f'''(a)}{6}, \quad \dots, \quad c_n = \frac{f^{(n)}(a)}{n!}.$$

Taylor-polynomet av grad  $n$  er dermed

$$p_n(x) = f(a) + (x - a)f'(a) + \frac{(x - a)^2}{2}f''(a) + \cdots + \frac{(x - a)^n}{n!}f^{(n)}(a),$$

og for å understreke avhengigheten av  $f$  skriver vi ofte  $p_n(x) = T_n f(x)$  eventuelt  $p_n(x) = T_n f(x; a)$  hvis vi også trenger å ta med avhengigheten av  $a$ .



Figur 9.1. Funksjonen  $\sin x$  og dens 9 første Taylor-polynomer.

### 9.1.2 Feilledet

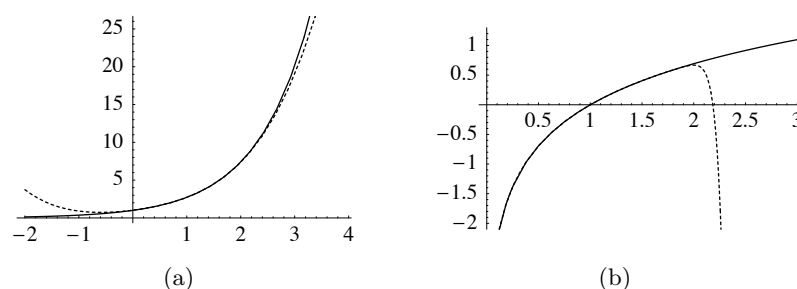
Det er viktig å huske på at Taylor-polynomet  $T_n f(x)$  er en tilnærming til  $f$ , og skal vi bruke denne tilnærmingen er det som regel viktig å ha en formening om hvor stor feilen er. Dette kan vi lese ut fra restleddet  $R_n f(x) = f(x) - T_n f(x)$  som kan skrives på formen

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

der  $c$  er et tall i intervallet  $(a, x)$  (intervallet  $(x, a)$  hvis  $x < a$ ).

Vi legger merke til at når  $x$  nærmer seg  $a$  så vil feilen  $R_n f(x)$  gå mot null. Dette er ikke så overaskende siden vi vet at feilen er 0 i  $x = a$ . Det er også interessant å se hva som skjer når  $n$  går mot uendelig og  $x$  holdes fast. Uttrykket  $1/(n+1)!$  vil ganske raskt bli svært lite mens  $(x-a)^{n+1}$  vil gå mot null, en eller uendelig, avhengig av om  $|x-a|$  er mindre enn en, lik en eller større enn en. Den siste faktoren  $f^{(n+1)}(c)$  kan essensielt oppføre seg på to måter. De peneste funksjonene er de som er slik at alle de deriverte kan begrenses av en konstant som er uavhengig av derivasjonsordenen. For eksempel vet vi at tallverdien til alle de deriverte til både  $\sin x$  og  $\cos x$  kan begrenses av konstanten 1. Siden uttrykket  $(x-a)^{n+1}/(n+1)!$  går mot null når  $n$  går mot uendelig, uansett hva  $x$  og  $a$  er, så ser vi at for slike funksjoner vil verdien av Taylor-polynomet i  $x$  konvergere pent mot  $f(x)$ . De ikke så pene funksjonene er de som har deriverte som ikke kan begrenses av en konstant uavhengig av derivasjonsordenen. Slike funksjoner skal vi ikke beskjeftige oss med her.

Figurene 9.1 og 9.2 viser noen kjente funksjoner med noen tilhørende Taylor-polynomer. I figur 9.1 ser vi hvordan Taylor-polynomene får med seg flere og flere av svingningene til  $\sin x$  når graden øker. Det viser seg at når  $n$  går mot uendelig så vil Taylor-polynomene konvergere mot  $\sin x$  for alle verdier av  $x$ . Det samme gjelder for eksponentialfunksjonen  $e^x$  som er vist i figur 9.2 (a). For logaritmefunksjonen  $\log x$  er situasjonen imidlertid annerledes. Som vi ser i figur 9.2 (b) så tilnærmes  $\log x$  godt av sitt Taylor-polynom av grad 20 på intervallet  $(0, 2]$ , men når  $x$  blir større enn 2 bli avviket fort stort. Det viser



**Figur 9.2.** Funksjonen  $e^x$  og dens Taylor-polynom av grad 4 (a) og funksjonen  $\log x$  og dens Taylor-polynom av grad 20.

seg at dette gjelder uansett hvor høy grad Taylor-polynomet har. Studiet av når følger av polynomer som disse konvergerer mot en fast funksjon er en viktig del av matematikken som du vil møte i senere matematikkurs.

### 9.1.3 Beregning av verdier på polynomer

Til slutt i denne delen skal vi ta for oss noe som ikke har spesielt med Taylor-polynomer å gjøre, men som vi trenger om vi skal bruke Taylor-polynomer på en datamaskin. Vi skal utlede en effektiv algoritme for å beregne verdien i  $x$  av polynomet

$$p(x) = c_0 + c_1x + \cdots + c_nx. \quad (9.7)$$

Legg merke til at om vi har en slik algoritme kan vi også beregne verdier på polynomer på formen  $q(x) = c_0 + c_1(x - a) + \cdots + c_n(x - a)^n$ . For hvis vi setter  $u = x - a$  kan vi skrive  $q$  som  $c_0 + c_1u + \cdots + c_nu^n$ , altså formen i (9.7), men med  $u$  som variabel.

Den opplagte måten å beregne  $p(x)$  på er å regne ut de forskjellige produktene på formen  $x \cdot x \cdots x$ , multiplisere med koeffisientene og addere sammen. En enklere måte er lettest å vise ved et eksempel. Anta at  $n = 3$ ; da kan vi skrive  $p$  som

$$p(x) = c_0 + c_1x + c_2x^2 + c_3x^3 = c_0 + x(c_1 + x(c_2 + xc_3)).$$

Dette kan vi gjøre for generell grad,

$$p(x) = \sum_{i=0}^n c_i x^i = c_0 + x(c_1 + x(c_2 + \cdots + x(c_{n-1} + xc_n) \cdots)).$$

For å regne ut dette begynner vi innerst og arbeider oss utover og bruker  $(r_i)_{i=0}^n$  for å ta vare på delresultatene. Vi setter først  $r_n = c_n$  og regner deretter ut

$$r_i = c_i + x r_{i+1}, \quad \text{for } i = n - 1, n - 2, \dots, 0.$$

Etter dette har vi multiplisert ut alle parentesene slik at  $r_0$  vil ha verdien  $p(x)$ . Siden vi ikke trenger å ta vare på verdien av alle  $r_i$ 'ene kan vi klare oss med en  $r$ . Dette gir oss følgende algoritme.

**Algoritme 9.1** (Nestet multiplikasjon). La polynomet  $p(x) = c_0 + c_1x + \dots + c_nx^n$  være gitt ved at koeffisientene er lagret i arrayen  $c$  og  $x$  i variabelen  $x$ . Etter beregningene

```
r = c[n];
for (i=n-1; i>=0; i--)
{
    r = c[i] + x*r;
}
```

vil variabelen  $r$  inneholde verdien  $p(x)$ .

## 9.2 Interpolasjon og andre approksimasjonsmetoder

Når vi nå er kjent med Taylor-polynomer er det naturlig å se seg om etter andre metoder for å tilnærme funksjoner. Før vi tar for oss interpolasjon kan det være på sin plass med noen overordnede betraktninger omkring approksimasjon av funksjoner.

La oss aller først minne om at matematiske metoder og teknikker generelt kan brukes på minst to måter. Taylor-polynomer er et eksempel på en teknikk som ofte brukes for å få dypere innsikt i en funksjon, og ofte trenger vi bare papir og blyant for å gjøre beregningene. Det er derfor viktig å øve seg i bruk av Taylor-polynomer ved å regne oppgaver (med papir og blyant). Taylor-polynomer er også et nyttig verktøy for å utlede andre metoder som Newtons metode for å finne nullpunkter og Eulers metode for å løse differensialligninger. Taylor-polynomer brukes sjeldnere på datamaskin som en praktisk approksimasjonsmetode fordi vi som regel trenger en approksimasjon som ikke bare gir liten feil i nærheten av et punkt, men over et intervall.

De andre approksimasjonsmetodene vi skal se på her er for det meste praktiske metoder som brukes på datamaskiner for å tilpasse måledata med glatte kurver. Dette betyr at det ikke er så viktig å kunne bruke metodene på papiret. Derimot er det viktig å vite om det er noen begrensninger på når metoden kan brukes, hvor komplisert er algoritmen som implementerer metoden, hvilke egenskaper vil tilnærmingen ha, hvor nøyaktig er tilnærmingen, hvor følsom er metoden for avrundingsfeil også videre. Dette betyr at det er viktigere med en god forståelse av metodene mer enn fingerferdighet med å bruke dem på mindre problemer med papir og blyant (selv om det også kan være nyttig i blant). En slik grundig forståelse får du neppe av å lese det som står her, men forhåpentligvis får du en liten smakebit på hvordan metodene framkommer og hva de kan brukes til.

Når vi skal velge oss en approksimasjonsmetode er det to hovedvalg vi må gjøre. Det første er å bestemme oss for hvilken form approksimasjonen skal ha. Skal den være et polynom, en trigonometrisk funksjon, en rasjonal funksjon, en eksponensialfunksjon eller noe annet? Det fins gode metoder basert på alle disse funksjonsklassene, men det vanligste er å bruke approksimasjonsmetoder basert på polynomer eller trigonometriske funksjoner. Og av disse to klassene er nok polynomer det vanligste. Dette har sammenheng med det vi før har vært inne på, at polynomer egner seg særdeles godt til beregninger på datamaskin.

Det andre valget dreier seg om hvilken metode vi velger for å bestemme approksimasjonen når klassen av funksjoner vi skal bruke er valgt. Vi skal se litt nærmere på dette når vi først har tatt for oss et alternativ til Taylor-polynomer.

### 9.2.1 Interpolasjon med polynomer

Når vi tilnærmer en funksjon  $f$  med Taylor-polynomer må vi kunne beregne deriverte av  $f$ . Det er mange situasjoner der dette ikke er mulig, den vanligste er kanskje når  $f$  bare er kjent ved måleverdier, slik som når vi måler temperaturen ved ulike tidspunkter eller når vi samler et lydsignal. I kapittel 6 så vi litt på hvordan den deriverte kan tilnærmes ut fra kjente funksjonsverdier, men det er som regel vel så bra å beregne approksimasjonen direkte fra funksjonsverdiene uten å gå veien om deriverte.

Naturlig nok er tilnærmingene våre polynomer når vi bruker polynominterpolasjon, og tilnærmingen bestemmes ved at vi krever at feilen skal være null i noen isolerte punkter. Et velkjent eksempel er sekanten til en funksjon  $f$ . Da velger vi to forskjellige punkter  $x_0$  og  $x_1$  og lar tilnærmingen  $p_1$  være den rette linja (det polynomet av grad 1) som har samme verdi som  $f$  i disse punktene. Med andre ord så krever vi at

$$f(x_0) = p_1(x_0), \quad f(x_1) = p_1(x_1).$$

Hvis vi skriver  $p_1$  på standardformen  $p_1(x) = b_0 + b_1x$  får vi et ligningssystem som består av to ligninger med de to ukjente  $b_0$  og  $b_1$  som har løsningen

$$b_0 = \frac{f(x_0)x_1 - f(x_1)x_0}{x_1 - x_0}, \quad b_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}.$$

Siden vi har antatt at  $x_0 \neq x_1$  gir disse formlene alltid mening. Dette betyr at  $p_1$  er gitt ved

$$p_1(x) = \frac{f(x_1)x_0 - f(x_0)x_1}{x_1 - x_0} + \frac{f(x_1) - f(x_0)}{x_1 - x_0}x. \quad (9.8)$$

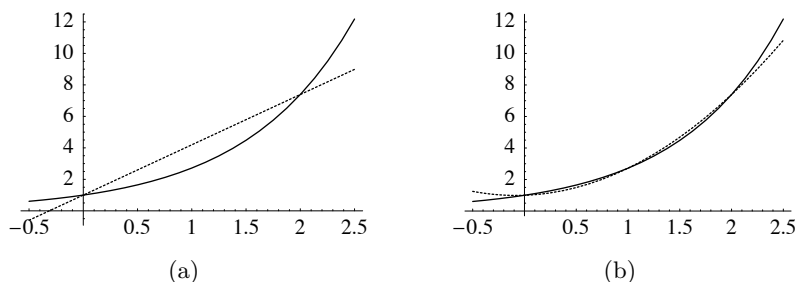
Dette polynomet kan også skrives som

$$p_1(x) = f(x_0)\frac{x_1 - x}{x_1 - x_0} + f(x_1)\frac{x - x_0}{x_1 - x_0}. \quad (9.9)$$

At dette er riktig ser vi ved at høyresiden er et førstegradspolynom som har verdien  $f(x_0)$  når  $x = x_0$  og verdien  $f(x_1)$  når  $x = x_1$ . Dette illustrerer igjen fenomenet med at et polynom kan skrives på flere måter. Formen (9.9) er spesielt hendig siden vi der kan bruke de to funksjonsverdiene  $f(x_0)$  og  $f(x_1)$  som koeffisienter. Dette kommer av at polynomet  $\ell_0(x) = (x_1 - x)/(x_1 - x_0)$  tilfredstiller betingelsene  $\ell_0(x_0) = 1$  og  $\ell_0(x_1) = 0$  mens  $\ell_1(x) = (x - x_0)/(x_1 - x_0)$  tilfredstiller betingelsene  $\ell_1(x_0) = 0$  og  $\ell_1(x_1) = 1$ . Dette skriver vi ofte som  $\ell_i(x_j) = \delta_{i,j}$  for  $i, j = 0, 1$  der symbolet  $\delta_{i,j}$  er definert som

$$\delta_{i,j} = \begin{cases} 1, & \text{hvis } i = j; \\ 0, & \text{ellers.} \end{cases}$$

La oss nå forsøke å øke graden. Siden et andregradspolynom har tre frie koeffisienter er det rimelig å håpe på at vi kan tvinge et slikt polynom gjennom tre gitte punkter  $x_0$ ,  $x_1$  og  $x_2$  med tilhørende funksjonsverdier  $f(x_0)$ ,  $f(x_1)$  og  $f(x_2)$ . Hvis vi skriver polynomet som  $p_2(x) = b_0 + b_1x + b_2x^2$  så vil de tre interpolasjonsbetingelsene gi tre ligninger for de tre ukjente  $b_0$ ,  $b_1$  og  $b_2$ . Dette ligningssystemet har ingen spesiell struktur som vi



**Figur 9.3.** Funksjonen  $e^x$  og med sekanten som interpolerer i  $x = 0$  og  $x = 2$  i (a), og parabelen som interpolerer i  $x = 0$ ,  $x = 1$  og  $x = 2$  i (b).

kan utnytte for å forenkle løsningsprosedyren. La oss derfor se om det lar seg gjøre å generalisere den spesielle formen for sekanten som vi fant i (9.9).

Trikset består i finne andregradspolynom som er 0 i alle punkter unntatt ett, der de har verdien 1. Det er lett å finne et polynom som er 0 i  $x_1$  og  $x_2$ . Polynomet  $q(x) = c(x - x_1)(x - x_2)$  har åpenbart denne egenskapen uansett hva konstanten  $c$  er. Det vil vanligvis ikke ha verdien 1 i  $x_0$ , men det kan vi fort ordne ved å velge  $c$  på riktig måte. Betingelsen

$$q(x_0) = c(x_0 - x_1)(x_0 - x_2) = 1$$

gir  $c = 1/((x_0 - x_1)(x_0 - x_2))$ . Vi ser derfor at polynomet

$$\ell_{0,2}(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}$$

tar verdien 1 i  $x_0$  og verdien 0 i  $x_1$  og  $x_2$ . Denne konstruksjonen kan vi gjenta med de andre punktene. Dette gir oss de to polynomene  $\ell_{1,2}$  og  $\ell_{2,2}$ ,

$$\ell_{1,2}(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)},$$

$$\ell_{2,2}(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}.$$

Vi ser at  $\ell_{1,2}$  har verdien 1 i  $x_1$  og 0 i  $x_0$  og  $x_2$ , mens  $\ell_{2,2}$  har verdien 1 i  $x_2$  og verdien 0 i  $x_0$  og  $x_1$ . Men dette betyr at polynomet

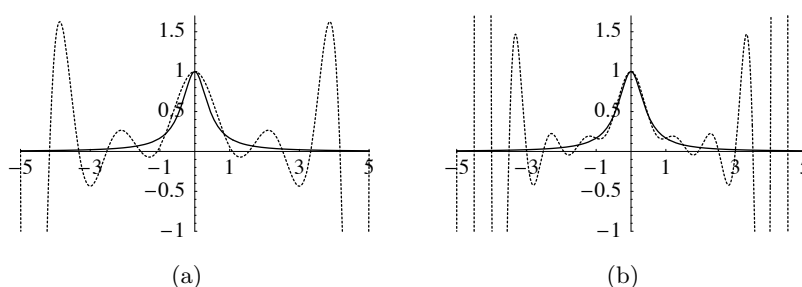
$$p_2(x) = f(x_0)\ell_{0,0}(x) + f(x_1)\ell_{1,2}(x) + f(x_2)\ell_{2,2}(x)$$

er av grad 2 og har samme verdi som  $f$  i punktene  $x_0$ ,  $x_1$  og  $x_2$  siden hvert av de tre leddene bare er ulik 0 i ett punkt, og i dette punktet har de riktig funksjonsverdi. Når vi adderer de tre leddene sammen "ødelegger de derfor ikke for hverandre".

Figur 9.3 viser eksempler på interpolasjon av eksponensialfunksjonen med en rett linje og med en parabel (polynomene er stiplet). Vi ser at parabelen er en så god tilnærming at det er vanskelig å skille den fra funksjonen den tilnærmer. Hvis vi øker graden og interpolerer  $e^x$  i flere uniformt spredte punkter kan vi få feilen så liten vi måtte ønske.

Konstruksjonen av interpolerende polynom kan generaliseres til vilkårlig grad.





**Figur 9.4.** Funksjonen  $1/(1+5x^2)$  med polynomet av grad 11 som interpolerer 12 uniformt spredte punkter i intervallet  $[-5, 5]$  (a) og interpolanten av grad 19 som interpolerer 20 uniformt spredte punkter i det samme intervallet (b) (polynomene er stiple).

**Setning 9.2.** La funksjonen  $f$  være gitt sammen med de  $n+1$  punktene  $x_0, x_1, \dots, x_n$ , og definer de  $n+1$  polynomene  $\{\ell_{i,n}\}_{i=0}^n$  ved

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

Da er polynomet  $p_n(x) = \sum_{i=0}^n f(x_i)\ell_{i,n}(x)$  det eneste polynomet av grad  $n$  som tilfredstiller interpolasjonsbetingelsene  $p_n(x_j) = f(x_j)$  for  $j = 0, 1, \dots, n$ .

Som i det kvadratiske tilfellet er det ikke så vanskelig å sjekke at  $\ell_{i,n}$  er 1 i  $x_i$  og 0 i alle de andre gitte punktene. Ved å multiplisere med  $f(x_i)$  får vi derfor et polynom som har verdien  $f(x_i)$  i  $x_i$  og verdien 0 i alle de andre punktene. Når vi så adderer opp alle leddene får vi polynomet av grad  $d$  som tilfredstiller interpolasjonsbetingelsene.

Det som gjenstår er å vise at det bare fins ett polynom som tilfredstiller interpolasjonsbetingelsene. Vi skal anta at det fins et polynom  $r$  som også har denne egenskapen og vise at da må  $r$  være lik  $p_n$ . La oss se på differansen  $e = p_n - r$  mellom de to polynomene. Siden både  $p_n$  og  $r$  har verdien  $f(x_i)$  i  $x_i$  for  $i = 0, \dots, n$  ser vi at  $e$  er 0 i alle de  $n+1$  punktene. Dette polynomet av grad  $n$  har altså  $n+1$  nullpunkter. Men fra algebraens fundamentalteorem vet vi at dette er umulig for et polynom av grad  $n$  så da er eneste mulighet at  $e$  er identisk null (alle koeffisientene er null). Men hvis  $e = p_n - r = 0$  må vi ha  $r = p_n$ . Altså har vi bare ett polynom som tilfredstiller interpolasjonsbetingelsene.

Figur 9.4 viser polynomer av forholdsvis høy grad (11 og 19) som interpolerer funksjonen  $1/(1+5x^2)$ . Noe overaskende så ser vi at feilen er forholdsvis stor, særlig nær endene av intervallet, og den blir større ettersom graden øker (plottene er klippet i  $y$ -retningen). I midten av intervallet er imidlertid tilnærmingen bra og ser ut til å bedres med økende grad. Forøvrig er kanskje det mest iøynefallende de voldsomme svingningene i de interpolerende polynomene. Selv når feilen er liten vil et interpolerende polynom vanligvis svinge rundt funksjonen det interpolerer, det ligger i interpolasjonens natur. Du vil se det samme om du plasserer en bøyeelig linjal på et bord og forsøker å tvinge den til å gå gjennom noen faste punkter.

Selv om figur 9.4 kan virke litt deprimerende med tanke på om interpolasjon er en god approksimasjonsmetode så fins det en del positive resultater. Resultatene forutsetter

at funksjonen  $f$  som vi interpolerer er glatt (en funksjon som kan deriveres mange ganger uten problemer). Dersom vi lar interpolasjonspunktene bevege seg mot hverandre og bli konsentrert i et stadig mindre intervall vil feilen på dette intervallet gå mot null når intervallbredden går mot null. I grensen, når alle punktene blir like, vil det interpolerende polynomet falle sammen med Taylor-polynomet av samme grad. Det går også an å vise at for en glatt funksjon definert på et intervall  $[a, b]$  så går det for hvert heltall  $n \geq 0$  an å finne en samling av  $n + 1$  interpolasjonspunkter slik at hvis vi interpolerer i disse punktene med et polynom av grad  $n$  vil feilen gå mot null når graden  $n$  går mot uendelig.

### 9.2.2 Andre metoder

De to approksimasjonsmetodene vi har sett på så langt er begge basert på å konstruere polynomer som reproduserer informasjon om  $f$  i diskrete punkter. Taylor-polynomer reproduserer verdien og de første deriverte til  $f$  i ett punkt, mens vi ved interpolasjon reproduserer funksjonsverdiene til  $f$  i flere punkter. Det er imidlertid andre måter å finne tilnærminger på.

En vanlig metode er å finne det polynomet som gjør at feilen i approksimasjonen blir minst mulig. Dette kan høres greit ut, men innebærer noen uventede utfordringer. Spørsmålet er nemlig hvordan vi kan måle feilen når vi tilnærmer  $f$  med et polynom  $p$ . Feilfunksjonen  $e = f - p$  er jo en funksjon, og det er dermed ikke uten videre så lett å se hva det vil si å gjøre denne feilen minst mulig. Nøkkelen ligger i hvordan vi kan måle størrelsen på en funksjon.

La oss anta at funksjonen  $f$  vi skal tilnærme er definert på et interval  $[a, b]$ . Et mål for størrelsen til feilen har vi ved uttrykket

$$\max_{x \in [a, b]} |f(x) - p(x)|. \quad (9.10)$$

Dette svarer til at vi benytter verdien av feilfunksjonen i det punktet der den er størst som et mål på feilen. Det å minimere feilen betyr da å finne et polynom  $p$  slik at dette maksimale avviket blir minst mulig. Hvis vi bruker polynomer av grad  $n$  og for enkelhets skyld skriver dem på formen  $p(x) = c_0 + c_1x + \dots + c_nx^n$  så ser vi at feilen i (9.20) kan skrives som

$$\epsilon_1(c_0, \dots, c_n) = \max_{x \in [a, b]} |f(x) - (c_0 + c_1x + \dots + c_nx^n)|.$$

Å finne polynomet av grad  $n$  som gjør feilen minst mulig innebærer derfor å minimere funksjonen  $\epsilon_1$  med hensyn på variablene  $c_0, c_1, \dots, c_n$ .

En annen måte å måle feilen på er ved integralet

$$\int_a^b |f(x) - p(x)| dx,$$

altså arealet av forskjellen mellom de to funksjonene. I dette tilfellet blir funksjonen vi skal minimere

$$\epsilon_2(c_0, \dots, c_n) = \int_a^b |f(x) - (c_0 + c_1x + \dots + c_nx^n)| dx.$$

Igjen kan vi bestemme en tilnærming til  $f$  ved å gjøre dette uttrykket minst mulig. Vi ser da at det er viktig å ha med tallverditegnet, for hvis ikke kan vi få feilen til å bli veldig liten ved å velge  $p$  slik at arealet mellom funksjonene er stort, men med negativt fortegn, noe som ikke svarer til at  $p$  er nær  $f$ .

Ulempen med disse to metodene er at det for begge er forholdsvis vanskelig å bestemme polynomene som gjør feilen minst mulig. Fra teorien for funksjoner av flere variable vet vi at vi kan finne ekstremalpunktene til en deriverbar funksjon ved å løse ligningene vi får ved å sette gradienten (de partielle deriverte) til null. Dessverre inneholder uttrykkene for  $\epsilon_1$  og  $\epsilon_2$  tallverdifunksjonen som ikke er deriverbar overalt ( $|x|$  har en knekk i  $x = 0$ ). Det er fremdeles mulig å bestemme minimumspunktene til  $\epsilon_1$  og  $\epsilon_2$ , men den eneste muligheten er å gjøre det ved iterative algoritmer som først gjetter på en løsning og så stadig forsøker å forbedre denne i håp om at prosessen vil konvergere mot polynomet som gjør feilen minst mulig.

Et tredje feilmål er populært fordi en med det kan finne polynomet som gjør feilen minst mulig uten å gjøre slike iterasjoner. Vi måler da feilen ved

$$\int_a^b (f(x) - p(x))^2 dx.$$

Siden vi kvadrerer feilfunksjonen vil uttrykket under integraltegnet aldri bli negativt så vi slipper unna tallverditegnet. Setter vi inn for  $p$  får vi

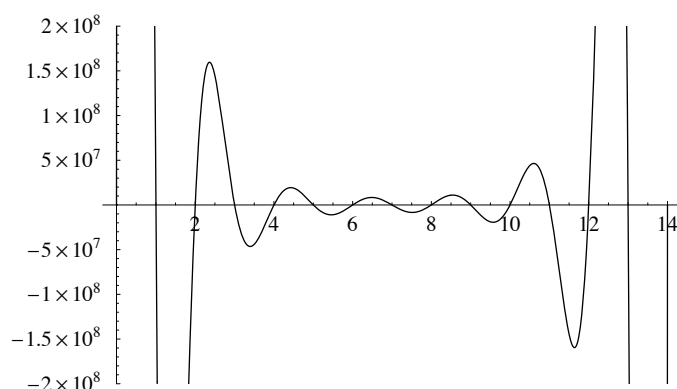
$$\epsilon_3(c_0, \dots, c_n) = \int_a^b (f(x) - (c_0 + c_1x + \dots + c_nx^n))^2 dx.$$

Dette uttrykket kan vi derivere med hensyn på hver av de  $n + 1$  koeffisientene  $c_0, \dots, c_n$ . Setter vi disse deriverte lik null får vi et lineært ligningssystem med  $n + 1$  lineære ligninger i de  $n + 1$  ukjente koeffisientene  $c_0, c_1, \dots, c_n$ . Det er ikke så vanskelig å vise at dette ligningssystemet har nøyaktig en løsning og at denne løsningen gir det eneste minimumspunktet til  $\epsilon_3$ .

### 9.3 Valg av basis er viktig, Bernstein-basisen

Så langt i dette kapitlet har vi sett flere eksempler på at det kan være fordelaktig å ikke alltid skrive et polynom av grad  $n$  på standardformen  $c_0 + c_1x + \dots + c_nx^n$ . Ved å bruke andre former så vi at det ble mye enklere å finne både Taylor-polynomer og interpolerende polynomer. Ikke overaskende har de ulike måtene å skrive polynomer på ulik følsomhet for avrundingsfeil også. I denne seksjonen skal vi presentere en spesielt gunstig måte å skrive polynomer på, ikke minst med tanke på å gjøre avrundingsfeilen liten.

Men hvorfor trenger vi å velge oss en måte å skrive polynomer på, kan vi ikke bruke den formen som til en hver tid er mest hendig? Hvis vi arbeider med papir og blyant er det selvsagt riktig, da er det ingen som legger noen begrensninger på hvordan vi representerer polynomer. Arbeider vi på en datamaskin derimot blir det fort tungvint å bruke flere forskjellige polynomformer. Skal vi lagre et polynom er det mest naturlig å lagre koeffisientene i forhold til en eller annen polynomform. Hvis vi da skal bruke ulike polynomformer må vi si fra hvilken polynomform som er brukt hver gang vi lagrer et



**Figur 9.5.** Wilkinson-polynomet  $w(x)$  (noe klipping i  $y$ -retningen).

polynom og vi må ha programvare for å oversette mellom de forskjellige formene. En ting er at det tar tid å skrive slik programvare, en annen at hver gang vi gjør en slik konvertering gjør vi en avrundingsfeil som dermed ødelegger noe av nøyaktigheten i vår representasjon. Det er derfor flere fordeler med å velge seg en fast representasjon når vi skal lage et programsystem.

### 9.3.1 Et problematisk polynom

Aller først trenger vi litt terminologi omkring ulike skrivemåter for polynomer. Når vi skriver polynomer på formen  $c_0 + c_1x + \dots + c_nx^n$  så er polynomet dannet ved *lineære kombinasjoner* av polynomene  $1, x, \dots, x^n$ . Disse enkle polynomene som vi bruker som byggeklosser kalles *basis-polynomer* og samlingen av alle disse basis-polynomene kalles *potens-basisen*. Som vi har sett fins det andre måter å skrive polynomer på, og dette svarer til å bruke andre basiser. Skriver vi polynomer som  $c_0 + c_1(x-a) + \dots + c_n(x-a)^n$  bruker vi den skiftede potensbasisen  $1, x-a, \dots, (x-a)^n$ , mens basisen  $\{\ell_{i,n}\}_{i=0}^n$  som vi brukte i forbindelse med interpolasjon kalles *Lagrange-basisen*<sup>2</sup>.

Med tanke på avrundingsfeil er potensbasisen spesielt problematisk, særlig når vi beveger oss langt bort fra  $x = 0$ . Som et eksempel skal vi se på polynomet

$$w(x) = \prod_{i=0}^{14} (x - i)$$

som ofte kalles Wilkinson-polynomet<sup>3</sup>, se figur 9.5. Hvis vi multipliserer ut parentesene

<sup>2</sup>Etter den franske matematikeren Joseph-Louis Lagrange (1736–1813).

<sup>3</sup>Etter James Wilkinson (1919–1986), engelsk matematiker. Wilkinson var involvert i arbeidet med den første engelske datamaskinen ACE og var en av pionerene innen analyse av avrundingsfeil.

og skriver polynomet på potensform finner vi at polynomet også kan skrives

$$\begin{aligned} w(x) = & 87178291200x - 283465647360x^2 + 392156797824x^3 - 310989260400x^4 \\ & + 159721605680x^5 - 56663366760x^6 + 14409322928x^7 \\ & - 2681453775x^8 + 368411615x^9 - 37312275x^{10} + 2749747x^{11} \\ & - 143325x^{12} + 5005x^{13} - 105x^{14} + x^{15}. \end{aligned}$$

Hvis vi representerer koeffisientene som 64-bits flyttall forteller Mathematica oss at  $w(13) = 264$ . Dette er selvsagt helt feil siden vi har konstruert polynomet slik at det er 0 i alle heltallene mellom 0 og 14. Spørsmålet er hva som er årsaken til problemet. Er det bare rutinene i Mathematica for å beregne verdien av et polynom på potensform som er dårlige, er det potensformen som skaper problemene eller er polynomet i seg selv så problematisk at vi ikke kan forvente et mer nøyaktig svar? Polynomet er helt klart problematisk i seg selv, men det viser seg at potensformen forsterker problemet. For å vise dette skal vi representere polynomet i en annen basis og se at om vi da regner ut  $w(13)$  så får vi et bedre resultat.

Vår nye basis er spesielt tilpasset et intervall  $[a, b]$  og kalles *Bernstein-basisen*<sup>4</sup>. Den er gitt ved

$$b_{i,n}(x) = \binom{n}{i} \left(\frac{x-a}{b-a}\right)^i \left(\frac{b-x}{b-a}\right)^{n-i}, \quad i = 0, 1, \dots, n.$$

Hvis vi setter  $[a, b] = [0, 1]$  forenkler høyresiden seg og basisen blir

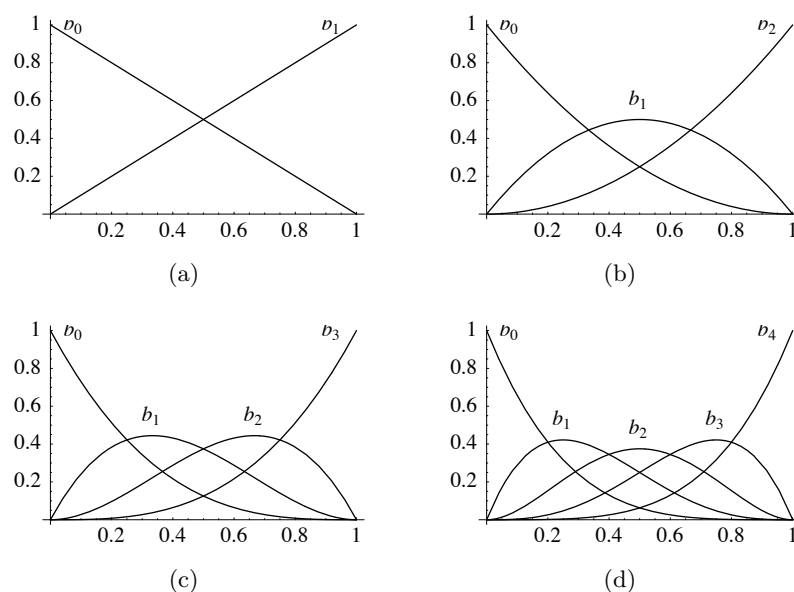
$$b_{i,n}(x) = \binom{n}{i} x^i (1-x)^{n-i}, \quad i = 0, 1, \dots, n.$$

De ulike polynomene i Bernstein-basisen av grad 1–4 på intervallet  $[0, 1]$  er vist i figur 9.6. Når det er klart fra sammenhengen hvilken grad vi bruker vil vi ofte utelate den andre indeksen  $n$  i  $b_{i,n}$ .

Bernstein-basisen på et generelt intervall  $[a, b]$  framkommer ved å flytte venstre ende til  $a$  og så strekke eller krympe funksjonene til høyre ende faller i  $b$ . Plott av basisen for et vilkårlig intervall ser derfor akkurat ut som plottene i figur 9.6 bortsett fra at tallene på  $x$ -aksen vil være annerledes.

Wilkinson-polynomet  $w(x)$  kan også skrives som  $w(x) = \sum_{i=0}^{15} c_i b_{i,15}(x)$  for passende koeffisienter ( $c_i$ ). Gjør vi det og representerer koeffisientene som 64-bits flyttall kan vi regne ut  $w(13)$  i denne representasjonen. Resultatet vi da får er  $w(13) = -0.0000244541$ . Selv om dette også er galt er det langt fra så galt som svaret vi fikk med potensbasisen. Det viser seg at Bernstein-basisen generelt oppfører seg bedre numerisk enn potensbasisen. Bernstein-basisen har også mange andre gode egenskaper som gjør at den ofte brukes ved representasjon av polynomer på datamaskiner. Ikke minst er Bernstein-basisen utgangspunktet for de såkalte Bezier-kurvener.

<sup>4</sup>Sergi Natanovich Bernstein (1880–1968) var født i Ukraina, men arbeidet i Russland (Sovjetunionen)



Figur 9.6. Polynomene i Bernsteinbasen av grad 1 (a), 2 (b), 3 (c) og 4 (d).

### 9.3.2 Noen egenskaper ved Bernstein-basisen

Når polynombasisen er bestemt er all informasjon om et polynom gitt ved koeffisientene. Det burde derfor være mulig å lese polynomets oppførsel ut fra koeffisientene. For de fleste polynombasiser er dette ikke så enkelt, men den viktigste egenskapen til Bernstein-basisen er at koeffisientene modellerer polynomet på en intuitiv og naturlig måte. Før vi ser hva dette innebærer må vi ta for oss noen viktige egenskaper ved Bernstein-basisen. Disse egenskapene er stort sett ganske enkle å utlede, så vi overlater bevisene til oppgaver.

**Lemma 9.3.** *Bernstein-basisen har blant annet følgende egenskaper:*

1. Alle polynomene i Bernstein-basisen er alltid større enn eller lik null på intervallet  $[a, b]$ ,

$$b_{i,n}(x) \geq 0 \quad \text{for alle } x \in [a, b].$$

2. Polynomene i Bernsteinbasen summerer seg opp til 1,

$$\sum_{i=0}^n b_{i,n}(x) = 1, \quad \text{for alle } x \in \mathbb{R}.$$

3. Basispolynomene  $b_{i,n}(x)$  har sitt maksimum på intervallet  $[a, b]$  i punktet  $x_i^* = a + i(b - a)/n$ .

4. I  $x = a$  er  $b_{0,n}(a) = 1$  mens alle de andre basispolynomene er 0. I  $x = b$  er  $b_{n,n}(b) = 1$  mens alle de andre basispolynomene er 0.

5. Den deriverte av  $b_{i,n}$  er gitt ved

$$b'_{i,n}(x) = \begin{cases} -n b_{0,n-1}(x), & \text{når } i = 0; \\ n (b_{i-1,n-1}(x) - b_{i,n-1}(x)), & \text{når } 1 \leq i \leq n-1; \\ n b_{n-1,n-1}(x), & \text{når } i = n. \end{cases}$$

6. Med  $u = (x-a)/(b-a)$  så er  $1-u = (b-x)/(b-a)$  slik at

$$b_{i,n}(x) = \binom{n}{i} \left(\frac{x-a}{b-a}\right)^i \left(\frac{b-x}{b-a}\right)^{n-i} = \binom{n}{i} u^i (1-u)^{n-i}.$$

Egenskap 6 er svært nyttig ved programmering av polynomer på Bernstein-form i og med at polynomer på et generelt intervall kan omskrives til polynomer på intervallet  $[0, 1]$ . Ved hjelp av denne transformasjonen kan vi også oversette egenskaper som gjelder på intervallet  $[0, 1]$  til det mer generelle intervallet  $[a, b]$ .

### 9.3.3 Egenskaper ved polynomer på Bernstein-form

Fra egenskapene over kan vi nå utlede egenskaper for polynomer uttrykt i Bernstein-basisen. For enkelhets skyld gjør vi dette i det kubiske tilfellet ( $n = 3$ ).

For å forenkle terminologien skal vi heretter kalle et polynom som er skrevet i Bernstein-basisen et *Bernstein-polynom*. La

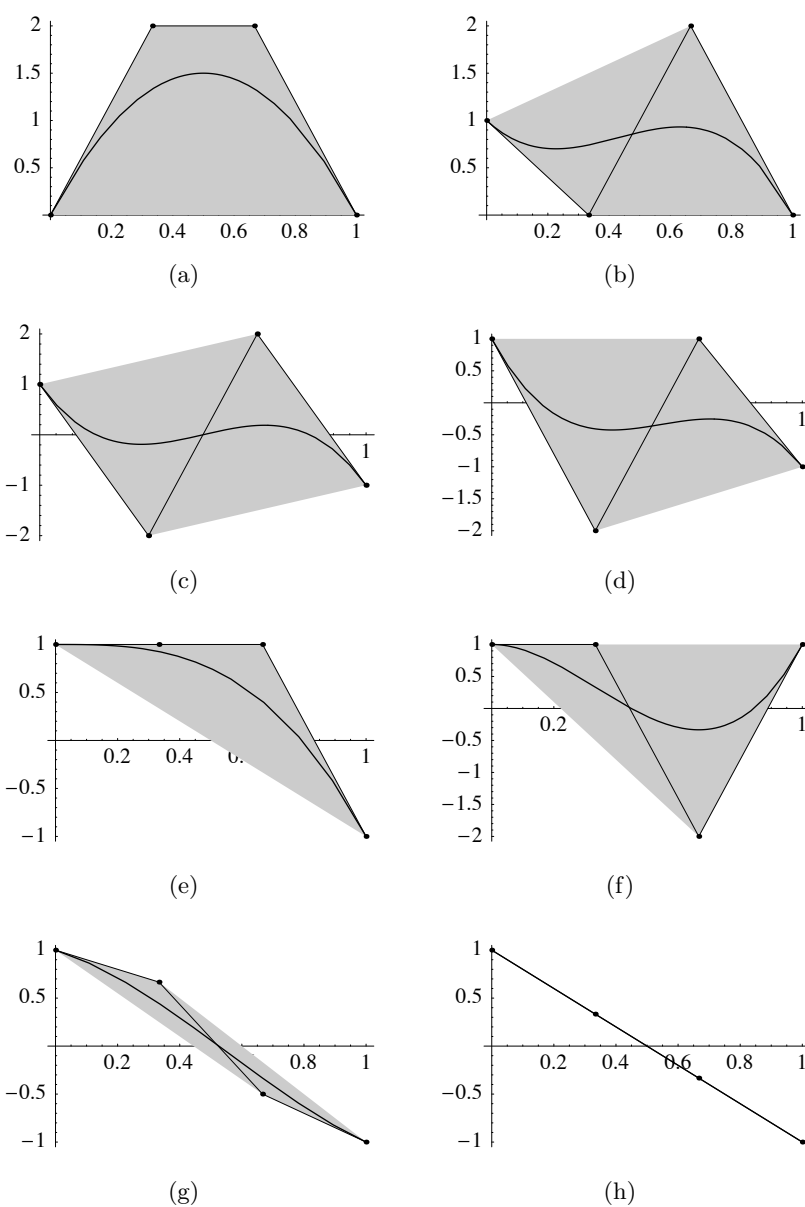
$$p(x) = c_0 b_0(x) + c_1 b_1(x) + c_2 b_2(x) + c_3 b_3(x)$$

være et kubisk Bernstein-polynom på intervallet  $[a, b] = [0, 1]$ . Hvis  $x$  ligger i intervallet  $[a, b]$  så vet vi fra egenskapene 1 og 2 at de fire basispolynomene er større enn eller lik null og summerer seg til 1. Dette betyr at  $p(x)$  er et veiet gjennomsnitt av de fire koeffisientene  $c_0, \dots, c_3$ . Siden et gjennomsnitt av en samling med tall alltid må være større enn det minste tallet og mindre enn det største tallet, gjelder det samme her — tallet  $p(x)$  må ligge mellom  $\min_i c_i$  og  $\max_i c_i$  (hvis alle fire  $c$ 'ene er like vil  $p(x)$  være lik denne felles verdien).

Hvis vi tar en titt på figur 9.6 (c) så ser vi at når vi beveger oss fra  $x = 0$  til  $x = 1$  så varierer størrelsen på basispolynomene. Spesielt så har alle basispolynomene et entydig maksimum i  $[a, b]$ , og fra egenskap 3 over vet vi at  $b_i$  har sitt maksimum i  $x = i/3$ . Dette betyr at koeffisienten  $c_i$  bidrar mest til  $p(x)$  i  $x = i/3$ . Hvis vi skal tegne ut koeffisienten  $c_i$  bør vi derfor tegne den med  $x$ -koordinat  $i/3$ , mens  $y$ -koordinaten er  $c_i$ . Punktet i planet gitt ved  $(i/3, c_i)$  kalles det  $i$ 'te *kontrollpunktet* til  $p$  slik at  $p$  til sammen har de fire kontrollpunktene  $\{(i/3, c_i)\}_{i=0}^3$ . Hvis vi forbinder kontrollpunktene med rette linjer får vi *kontrollpolygonet* til  $p$ .

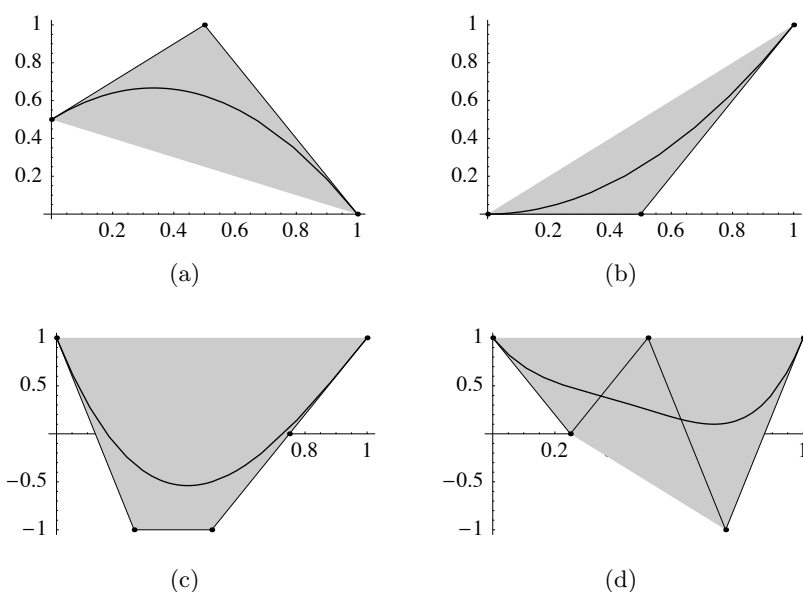
Dette er motivasjonen bak den generelle definisjonen av kontrollpunkter og kontrollpolygon.

**Definisjon 9.4.** La Bernstein-polynomet  $p(x) = \sum_{i=0}^n c_i b_{i,n}(x)$ , definert på intervallet  $[a, b]$ , være gitt. De  $n+1$  punktene i planet gitt ved  $c_i^p = (a + i(b-a)/n, c_i)$  for  $i = 0, 1, \dots, n$ , kalles kontrollpunktene til  $p$  mens den brudne linja som framkommer ved å forbinde kontrollpunktene med rette linjer kalles kontrollpolygonet til  $p$ .



**Figur 9.7.** Åtte eksempler på kubiske Bernstein-polynomer med tilhørende kontrollpolygoner. Koeffisientene er  $(0, 2, 2, 0)$  i (a),  $(1, 0, 2, 0)$  i (b),  $(1, -2, 2, -1)$  i (c),  $(1, -2, 1, -1)$  i (d),  $(1, 1, 1, -1)$  i (e),  $(1, 1, -2, 1)$  i (f),  $(1, 2/3, -1/2, -1)$  i (g) og  $(1, 1/3, -1/3, -1)$  i (h).





**Figur 9.8.** Eksempler på kvadratiske Bernstein-polynomer med tilhørende kontrollpolygoner ((a) og (b)) og Bernstein-polynomer av grad 4 ((c) og (d)).

Tolv eksempler på kubiske Bernstein-polynomer med tilhørende kontrollpunkter og kontrollpolygon er vist i figur 9.7 og 9.8. I alle tilfellene ser vi hvordan Bernstein-polynomet er en utglattet versjon av sitt kontrollpolygon. Det samlede visuelle inntrykket av sammenhengen mellom funksjon og kontrollpolygon har flere ingredienser.

**Lemma 9.5.** *Et Bernstein-polynom  $p(x)$  definert på intervallet  $[a, b]$  har blant annet følgende egenskaper:*

1. Verdien av polynomet i  $x = a$  faller sammen med det første kontrollpunktet, og verdien av polynomet i  $x = b$  faller sammen med det siste kontrollpunktet.
2. Tangenten til polynomet i  $x = a$  har samme retning som linja fra det første til det andre kontrollpunktet og tangenten i  $x = b$  har samme retning som linja fra det nest siste til det siste kontrollpunktet.
3. Verdien av polynomet ligger alltid mellom verdien til minste og største koeffisient.

Disse egenskapene følger direkte fra egenskaper ved basispolynomene  $b_{i,n}$ . For eksempel så følger egenskap 1 fra egenskap 4 i lemma 9.3 og egenskap 3 fra egenskap 1 og 2 i lemma 9.3, se oppgavene. Egenskap 2 følger fra egenskap 5 i lemma 9.3, men dette krever litt regning så vi tar med denne utledningen her.

**Bevis for egenskap 2.** La oss igjen holde oss til kubiske Bernstein-polynomer på in-

tervallet  $[0, 1]$ . Den deriverte ser vi da er gitt ved

$$\begin{aligned} p'(x) &= \sum_{i=0}^3 c_i b'_{i,3}(x) \\ &= 3 \left( -c_0 b_{0,2}(x) + c_1 (b_{0,2}(x) - b_{1,2}(x)) + c_2 (b_{1,2}(x) - b_{2,2}(x)) + c_3 b_{2,2}(x) \right) \\ &= 3(c_1 - c_0)b_{0,2}(x) + 3(c_2 - c_1)b_{1,2}(x) + 3(c_3 - c_2)b_{2,2}(x). \end{aligned}$$

Med andre ord ser vi at den deriverte er et kvadratisk Bernstein-polynom med koeffisienter som er gitt ved differansen mellom to nabokoeffisienter til det opprinnelige polynomet, multiplisert med 3. Spesielt så får vi at

$$p'(0) = 3(c_1 - c_0), \quad p'(1) = 3(c_3 - c_2),$$

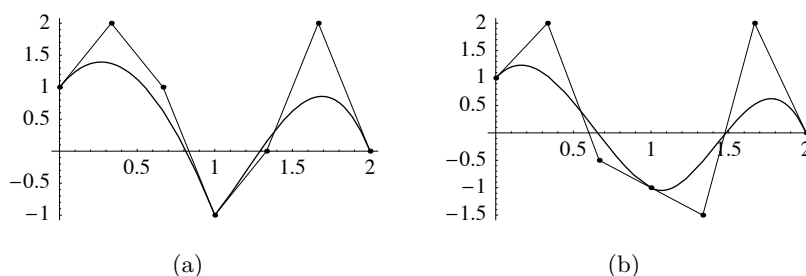
siden  $b_{0,2}(x) = 1$  mens  $b_{1,2}(0) = b_{2,2}(0) = 0$ , og tilsvarende så er  $b_{2,2}(1) = 1$  mens  $b_{0,2}(1) = b_{1,2}(1) = 0$ . Siden vi også har  $p(0) = c_0$  fra egenskap 1 så vet vi dermed at tangenten til  $p$  i  $x = 0$  har ligningen  $h(x) = c_0 + 3(c_1 - c_0)x$ . Vi har dessuten  $h(1/3) = c_1$  så tangenten går gjennom de to første kontrollpunktene til  $p$  og faller dermed sammen med det første segmentet av kontrollpolygonet. ■

Vi har skissert enda en egenskap ved Bernstein-polynomer i figurene: vi ser at Bernstein-polynomet i hvert tilfelle holder seg innenfor det grå området definert av kontrollpunktene. Før vi kan formulere dette mer presist må vi vite hvordan dette området framkommer. Det grå området  $\mathbb{G}$  er gitt ved den *konvekse innhyllningen* til kontrollpunktene. Det at området er *konvekst* betyr at om to punkter ligger i  $\mathbb{G}$  så vil også alle punktene på linjesegmentet som forbinder punktene ligge i  $\mathbb{G}$ . Nå går det an å vise at den konvekse innhyllningen til en samling punkter er den minste konvekse mengden som inneholder alle punktene i samlingen. Det grå området i figurene er dermed den minste mengden i planet som inneholder alle kontrollpunktene til det aktuelle Bernstein-polynomet. Med denne bakgrunnen kan vi formulere den siste egenskapen.

**Lemma 9.6.** *La  $p = \sum_{i=0}^n c_i b_{i,n}(x)$  være et Bernstein-polynom. For enhver  $x$  i intervallet  $[a, b]$  så ligger punktet  $(x, p(x))$  i den konvekse innhyllningen til kontrollpunktene til  $p$ .*

Beviset for dette vil det føre for langt å ta med her, men det hele bunner i at den konvekse innhyllningen av en samling punkter kan tolkes som samlingen av alle veide gjennomsnitt av punkter i samlingen. Vi ser derved sammenhengen med Bernstein-polynomer som jo er et veiet gjennomsnitt av sine koeffisienter.

Bernstein-polynomer har en mengde andre elegante egenskaper som vi ikke kan gå innpå her, men kjernen i de aller fleste av disse egenskapene er den nære sammenhengen mellom polynomet selv og dets kontrollpolygon. Det typiske er at en egenskap ved et Bernstein-polynom som regel kan karakteriseres ved en tilsvarende egenskap ved polynomet sitt kontrollpolygon. Dette er særlig hendig ved beregninger siden vi istedenfor å gjøre beregninger direkte med polynomet da kan gjøre tilsvarende beregninger med kontrollpolygonet. Fordelen med dette er at kontrollpolygonet matematisk sett er enkelt siden det bare består av  $n$  rette linjesegmenter.



**Figur 9.9.** Sammenlenking av to kubiske Bernstein-polynomer i  $x = 1$ . I (a) er resultatet kontinuerlig i skjøten, men den deriverte er diskontinuerlig, mens i (b) er både funksjonsverdi og derivert kontinuerlige i skjøten.

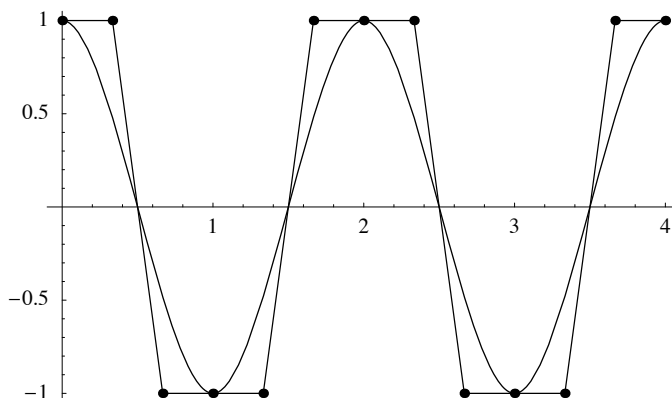
### 9.3.4 Sammenlenking av Bernstein-polynomer

De mange fine egenskapene ved Bernstein-basisen gjør at den ofte er å foretrekke når vi skal arbeide med polynomer på en datamaskin. Men hvor anvendelige er egentlige polynomer som approksimasjonsredskap? For å lage en tilnærming som skal være nøyaktig i nærheten av ett punkt fungerer Taylor-polynomer svært godt, men hva om vi skal tilnærme en komplisert funksjon over et langt intervall? Vi så jo faktisk i seksjon 9.2.1 at det ikke nødvendigvis er så lurt å interpolere i mange punkter med et polynom av høy grad. Og selv om polynomer av lav grad egner seg godt for datamaskinberegninger, øker både kompleksiteten og avrundingsfeilen raskt når graden øker, selv om vi bruker Bernstein-basisen.

For å kunne approksimere kompliserte funksjoner trenger vi mange frie parametre som kan justeres slik at tilnærmingen blir god. For polynomer er de frie parametrene koeffisientene, og polynomer med mange koeffisienter må derfor nødvendigvis ha høy grad, noe som altså ikke er så attraktivt fra et beregningsteknisk synspunkt. Heldigvis fins det en annen mulighet. Det er ikke bare ved å øke graden vi kan få tilgang på mange koeffisienter, vi kan også få det til med lav grad om vi bruker mange polynomer. Med en slik framgangsmåte kan vi dele opp det totale definisjonsintervallet til funksjonen vi skal approksimere i mindre delintervaller og så bruke et polynom på hvert delintervall. Det eneste vi må passe på er at vi hekter polynombitene pent sammen i skjøtene.

Figur 9.9 viser to eksempler på funksjoner som begge er sammensatt av to Bernstein-polynomer. I begge tilfellene er polynomene lenket sammen i  $x = 1$ . Siden Bernstein-polynomer interpolerer første og siste koeffisient er det lett å sikre at to sammenlenkede polynomer er kontinuerlige i skjøten. Dette vil være tilfelle om den siste koeffisienten i polynomet til venstre er lik den første koeffisienten i polynomet til høyre, slik som i figur 9.9.

Når det sammenlenkede polynomet bare er kontinuerlig i en skjøt vil det som regel ha en iøynefallende knekk, slik som i figuren. For å få en glattere overgang trenger vi at også den deriverte er kontinuerlig i skjøten. Vi vet at tangenten i det venstre endepunktet til et Bernstein-polynom faller sammen med det første linjesegmentet i kontrollpolygonet mens tangenten i det høyre endepunktet faller sammen med kontrollpolygonets siste linjesegment. Fra dette ser vi at både funksjonsverdi og derivert blir kontinuerlig i skjøten om det siste linjesegmentet i det første kontrollpolygonet ligger på samme rette linje som



**Figur 9.10.** En funksjon med kontinuerlig derivert satt sammen av 4 Bernstein-polynomer.

det første linjesegmentet i det andre kontrollpolyognet, slik som i figur 9.9 (b).

La oss oppsummere dette i en setning.

**Setning 9.7.** La  $p(x) = \sum_{i=0}^n c_i b_i(x)$  være et Bernstein-polynom av grad  $n$  på intervallet  $[a, b]$ , la  $q(x) = \sum_{i=0}^n d_i b_i(x)$  være et Bernstein-polynom av samme grad på intervallet  $[b, c]$  og la  $f$  betegne den sammensatt funksjonen gitt ved

$$f(x) = \begin{cases} p(x), & \text{for } x \in [a, b]; \\ q(x), & \text{for } x \in [b, c]. \end{cases}$$

Da er  $f$  kontinuerlig i  $x = b$  hvis og bare hvis  $c_n = d_0$ , mens  $f$  er kontinuerlig med kontinuerlig derivert i  $x = b$  hvis og bare hvis  $c_n = d_0$  og de tre kontrollpunktene  $(b - h_1, c_{n-1})$ ,  $(b, c_n)$  og  $(b + h_2, d_1)$  ligger på en rett linje. Her er  $h_1 = (b - a)/n$  og  $h_2 = (c - b)/n$ .

Ved å lenke sammen Bernstein-polynomer får vi en svært fleksibel klasse av funksjoner med gode approksimasjonsegenskaper. Et enkelt eksempel er vist i figur 9.10. For å beregne tilnærmingene kan vi bruke interpolasjon eller minimering av et passende feilmål, akkurat som for polynomer.

I en del sammenhenger kan det være ønskelig at både den andrederiverte og enda høyere deriverte skal være kontinuerlige over skjøtene mellom polynombitene. Dette kan vi få til ved å legge passende betingelser på koeffisientene, slik som i setning 9.7, men etterhvert blir disse betingelsene kompliserte å holde styr på. Det viser seg at det er mulig å konstruere enkle basisfunksjoner bestående av ulike polynomsegmenter som har denne kontinuiteten innebygd. Ved å multiplisere disse basisfunksjonene med koeffisienter og så addere sammen kan vi danne generelle stykkevise polynomer. Bygger vi stykkevise polynomer på denne måten kalles de ofte *splinefunksjoner*<sup>5</sup>.

<sup>5</sup>Ordet spline brukes på engelsk om en bøyelig linjal som kan tvinges til å gå gjennom noen faste punkter.

### 9.3.5 Beregning av verdier på Bernstein-polynomer

Vi har så langt ikke nevnt noen metode for å beregne verdien av et Bernstein-polynom. En mulighet er selvsagt å beregne verdien av hvert basispolynom, multiplisere med koeffisientene og så summere. Det fins flere andre (og bedre) metoder. Den mest brukte er basert på å beregne en sekvens av veide gjennomsnitt mellom to og to tall og er svært motstandsdyktig overfor avrundingsfeil. En enklere metode er å omskrive Bernstein-polynomet til en slags potensform, noe som i det kubiske tilfellet kan gjøres ved

$$\begin{aligned} p(x) &= c_0(1-x)^3 + 3c_1x(1-x)^2 + 3c_2x^2(1-x) + c_3x^3 \\ &= (1-x)^3 \left( c_0 + 3c_1 \frac{x}{1-x} + 3c_2 \left( \frac{x}{1-x} \right)^2 + c_3 \left( \frac{x}{1-x} \right)^3 \right) \\ &= (1-x)^3 (c_0 + 3c_1v + 3c_2v^2 + v^3), \end{aligned}$$

der vi har satt  $v = x/(1-x)$  og antar at  $x \neq 1$ . I stedet for å sette  $(1-x)^3$  utenfor som faktor kan vi sette  $x^3$  utenfor og få et polynom på potensform i  $w = (1-x)/x$ ,

$$p(x) = x^3(c_0w^3 + 3c_1w^2 + 3c_2w + c_3).$$

Potensbasisen gir større avrundingsfeil jo lenger bort fra  $x = 0$  vi kommer. Fra et beregningssynspunkt er det derfor best å bruke det første alternativet når  $x \in [0, 1/2]$  og det andre når  $x \in [1/2, 1]$ . Ordner vi oss på denne måten vil vi alltid ha  $v \leq 1$  eller  $w \leq 1$ , og på intervallet  $[0, 1]$  oppfører potensbasisen seg rimelig pent med tanke på avrundingsfeil.

For generell grad får vi at  $p(x)$  kan skrives på de to ekvivalente måtene

$$\begin{aligned} p(x) &= (1-x)^n \sum_{i=0}^n \binom{n}{i} c_i v^i \\ &= x^n \sum_{i=0}^n \binom{n}{i} c_{n-i} w^i \end{aligned} \tag{9.11}$$

der  $v = x/(1-x)$  og  $w = (1-x)/x$  som før (den første formelen gjelder ikke når  $x = 1$  og den siste ikke når  $x = 0$ ). På denne måten har vi dermed redusert det å beregne en funksjonsverdi for et Bernstein-polynom til det å beregne en verdi for et polynom skrevet på potensform.

Hvis vi arbeider på et generelt intervall  $[a, b]$  husker vi fra egenskap 6 i lemma 9.3 at det er lurt å sette  $u = (x-a)/(b-a)$ . Da er også  $1-u = (b-x)/(b-a)$  slik at

$$b_{i,n}(x) = \binom{n}{i} \left( \frac{x-a}{b-a} \right)^i \left( \frac{b-x}{b-a} \right)^{n-i} = \binom{n}{i} u^i (1-u)^{n-i}.$$

Dermed kan vi beregne verdier på Bernstein-polynomer på et vilkårlig intervall  $[a, b]$  dersom vi kan gjøre slike beregninger på intervallet  $[0, 1]$ .

Vi er nå nesten klar til å gi en algoritme for å beregne  $p(x)$ , det eneste som mangler er beregning av binomialkoeffisientene. Ved å sette inn definisjonen av  $\binom{n}{i-1}$  er det ikke så vanskelig å se at relasjonen

$$\binom{n}{i} = \frac{n-i+1}{i} \binom{n}{i-1}$$

holder for  $i = 1, \dots, n$ . Når vi dessuten vet at  $\binom{n}{0} = 1$  gir dette oss en grei algoritme for å beregne binomialkoeffisientene. Dermed har vi alle ingrediensene for å beregne verdien av et Bernstein-polynom.

**Algoritme 9.8.** La Bernstein-polynomet  $p(x) = \sum_{i=0}^n c_i b_{i,n}(x)$  på intervallet  $[a, b]$  være gitt. Etter at koden under er utført vil variabelen  $r$  inneholde verdien  $p(x)$ .

```

binom = 1; d[0] = c[0];
for (i=1; i<n+1; i++)
  {
    binom = binom*(n-i+1)/i;
    d[i] = c[i]*binom;
  }

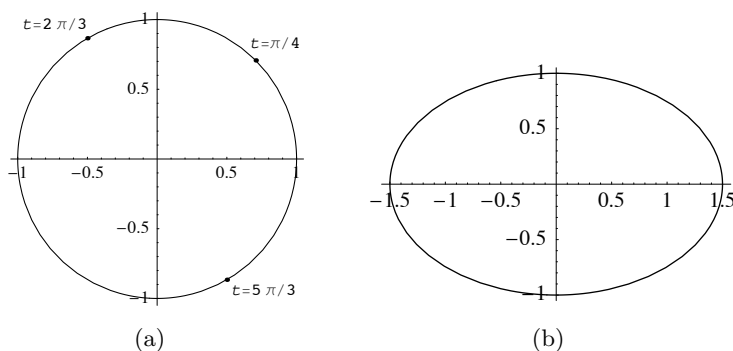
u = (x-a)/(b-a);

if (u < 0.5)
  {
    v = u/(1-u);
    r = d[n];
    for (i=n-1; i>=0; i--)
      {
        r = d[i] + v*r;
      }
    r = Math.pow(1-u,n)*r;
  }
else
  {
    w = (1-u)/u;
    r = d[0];
    for (i=1; i<=n; i++)
      {
        r = d[i] + w*r;
      }
    r = Math.pow(u,n)*r;
  }

```

I den første delen av algoritmen multipliserer vi koeffisientene med binomialkoeffisienter og lagrer resultatet i en ny array  $d$ . Matematisk svarer dette til å utføre  $d_i = \binom{n}{i} c_i$  for  $i = 0, \dots, n$ . Legg merke til at dette kan gjøres på samme måte uansett hvilken av de to formlene i (9.11) vi bruker siden en linje i Pascals trekant er symmetrisk om midten (mer presist så har vi  $\binom{n}{i} = \binom{n}{n-i}$ ).

Med tilordningen  $u = (b - x)/(b - a)$  transformerer vi oss fra intervallet  $[a, b]$  til intervallet  $[0, 1]$ . Den nye variabelen er dermed  $u$ , og avhengig av hvilken halvdel av  $[0, 1]$  denne ligger i velger vi den varianten i (9.11) som gjør at den endelige variabelen ( $v$  eller



**Figur 9.11.** De to parametriske kurvene  $\mathbf{r}(t) = (\cos t, \sin t)$  (i (a)) og  $\mathbf{r}(t) = (2 \cos t, \sin t)$  (i (b)).

$w$ ) ligger i intervallet  $[0, 1]$ . Hvis  $u < 0.5$  setter vi derfor  $v = u/(1 - u)$  og beregner  $p(x)$  som  $p(x) = (1 - u)^n \sum_{i=0}^n d_i u^i$ , hvis  $u \geq 0.5$  bruker vi  $p(x) = u^n \sum_{i=0}^n d_{n-i} w^n$ .

## 9.4 Parametriske kurver

En grunnleggende egenskap ved funksjonsbegrepet er at til hver verdi av argumentet så svarer det nøyaktig en funksjonsverdi. Geometrisk betyr dette at en vertikal linje ikke kan skjære grafen til funksjonen mer enn en gang. Ved hjelp av parametriske kurver kan vi representere mer generelle, en-dimensjonale, geometriske former.

### 9.4.1 Definisjon av parametriske kurver

Fra skolematematikken vet vi at ligningen for en sirkel med radius 1 er gitt ved  $x^2 + y^2 = 1$ . Hvis vi løser denne med hensyn på  $y$  er vi vant til å skrive  $y = \pm\sqrt{1 - x^2}$ . Dette siste uttrykket gir ikke  $y$  som én funksjon av  $x$ , men som to funksjoner av  $x$ , en verdi på den øvre halvsirkelen og en på den nedre. Det er altså umulig å representere hele sirkelen ved hjelp av en funksjon.

Ved hjelp av *parametriske representasjoner* kommer vi unna denne begrensningen. En parametriske representasjon av sirkelen er gitt ved uttrykket

$$\mathbf{r}(t) = (\cos t, \sin t), \quad t \in [0, 2\pi].$$

Vi ser at som funksjoner av en variabel så trenger  $\mathbf{r}$  et argument som er et reelt tall, men forskjellen er at  $\mathbf{r}$  for hver slik  $t$  i definisjonsområdet  $[0, 2\pi]$  produserer *to* reelle tall. Disse kan vi tenke på som et punkt i planet (eller ekvivalent, som en vektor i planet), og vi angir dette ved å skrive  $\mathbf{r} : [0, 2\pi] \mapsto \mathbb{R}^2$ .

Legg merke til at om vi setter  $x = \cos t$  og  $y = \sin t$  så kommer vi tilbake til den vanlige sirkelligningen via en velkjent identitet for trigonometriske funksjoner,

$$x^2 + y^2 = \cos^2 t + \sin^2 t = 1.$$

Et plott av  $\mathbf{r}$  er vist i figur 9.11 (a) med verdien av  $t$  angitt for noen typiske punkter, og figur 9.11 (b) viser den relaterte parametriske representasjonen  $\mathbf{r}(t) = (2 \cos t, \sin t)$ .

Hvis vi nå setter  $x = 2 \cos t$  og  $y = \sin t$  så ser vi at

$$\frac{x^2}{4} + y^2 = \cos^2 t + \sin^2 t = 1$$

som vi kjenner igjen som en typisk ellipseligning.

Noen flere eksempler på parametriske representasjoner er gitt i figur 9.12. Legg spesielt merke til at den parametriske representasjonen  $\mathbf{r}(t) = \mathbf{a} + t(\mathbf{b} - \mathbf{a})$  gir den rette linja gjennom de to punktene i planet gitt ved  $\mathbf{a}$  og  $\mathbf{b}$ . Med funksjoner har vi problemer med å representere vertikale linjer, men på parametriske form skiller slike linjer seg ikke ut. Hvis  $\mathbf{a}$  og  $\mathbf{b}$  har samme  $x$ -koordinat får vi en vertikal linje uten problemer, se (b). I (c) ser vi at funksjoner kan representeres som parametriske kurver. I (d) har vi rotert kurven i (c) slik at den havner langs  $y$ -aksen i steden, noe som selvsagt er umulig med funksjoner. Funksjonene i (e)–(h) har det til felles at de alle er på formen  $\mathbf{r}(t) = h(t)(\cos t, \sin t)$  for ulike valg av funksjonen  $h$ . Siden  $(\cos t, \sin t)$  gir enhets sirkelen så ser vi at alle kurvene oppstår ved å trykke sammen eller trekke ut sirkelen på ulike måter.

Ut fra disse innledende eksemplene kan vi sette opp en generell definisjon av parametriske representasjoner.

**Definisjon 9.9.** *En plan, parametriske representasjon eller parametriske kurve definert på intervallet  $[a, b]$ , er et uttrykk på formen  $\mathbf{r}(t) = (f(t), g(t))$  der  $f$  og  $g$  er to funksjoner definert på intervallet  $[a, b]$ .*

En viktig observasjon er at ulike parametriske representasjoner kan gi opphav til samme geometriske bilde. For eksempel kan vi også få fram sirkelen ved representasjonen  $\mathbf{r}(t) = (\cos 2\pi t, \sin 2\pi t)$ , men nå holder det om  $t$  varierer i intervallet  $[0, 1]$ . En litt mer overaskende sirkelrepresentasjon er gitt ved

$$\mathbf{r}(t) = \left( \frac{t}{\sqrt{1+t^2}}, \frac{1}{\sqrt{1+t^2}} \right), \quad t \in \mathbb{R},$$

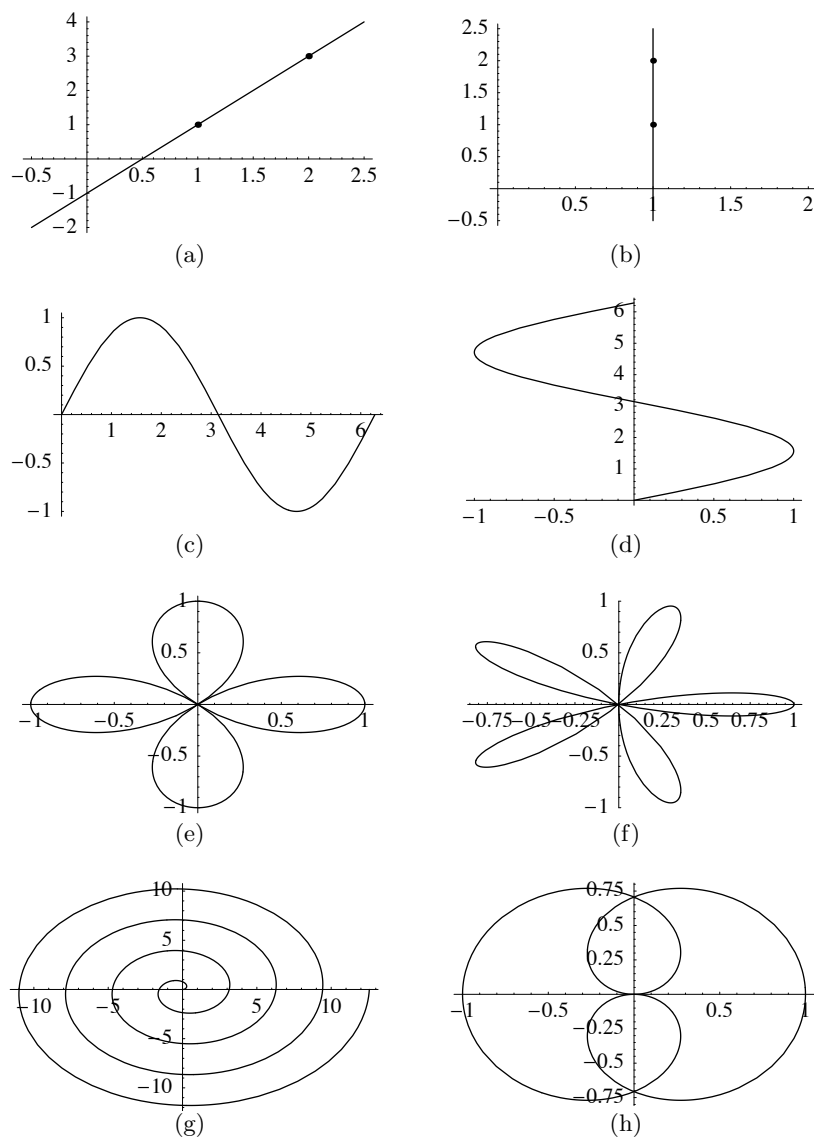
men merk at denne bare gir halve sirkelen.

Generelt er det alltid mange parametriske representasjoner som gir det samme, geometriske bildet, og i mer formelle framstillinger er en parametriske kurve definert som samlingen av alle parametriske representasjoner som gir opphav til det samme geometriske bildet. Vi skal ikke være så formelle og vil bruke begrepene om hverandre.

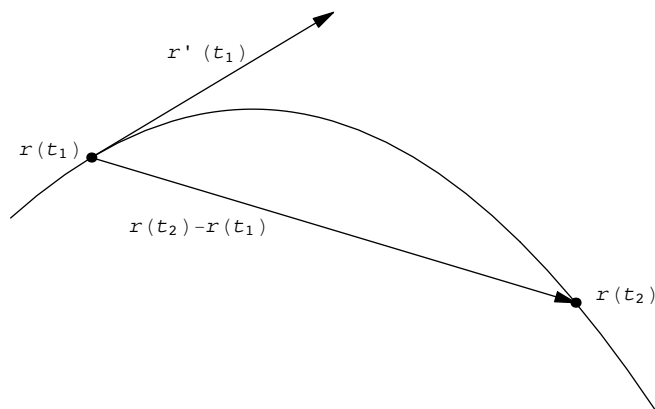
Vi skal holde oss til parametriske kurver i planet, men de kan enkelt generaliseres til høyere dimensjoner ved å legge til flere komponenter. For eksempel representerer den parametriske representasjonen  $\mathbf{r}(t) = (\cos t, \sin t, t)$  en spiral i rommet som hever seg over  $xy$ -planet. Dette ser vi siden de to første koordinatene gir en sirkel i  $xy$ -planet, mens  $z$ -komponenten vokser med  $t$  slik at vi beveger oss oppover i rommet når  $t$  øker. Ved å legge på flere komponenter får vi parametriske representasjoner som kan representere kurver med så mange romdimensjoner som vi måtte trenge.

Parametriske kurver har en fysisk tolkning som ofte kan være nyttig. Representasjonen  $\mathbf{r}$  definert på intervallet  $[a, b]$  gir et geometrisk bilde som vi kan tenke på som en måte å kjøre bil fra  $\mathbf{r}(a)$  til  $\mathbf{r}(b)$  langs veien definert av  $\mathbf{r}$ . Uttrykket  $\mathbf{r}(t)$  gir dermed posisjonen langs veien ved tidspunkt  $t$ . Ulike parametriske representasjoner svarer da til





**Figur 9.12.** Åtte parametriske kurver: (a)  $r(t) = a + t(b - a)$  med  $a = (1, 1)$  og  $b = (2, 3)$ ; (b) som (a), men med  $a = (1, 1)$  og  $b = (1, 2)$ ; (c)  $r(t) = (t, \sin t)$ ; (d)  $r(t) = (\sin t, t)$ , (e)  $r(t) = \cos(2t)(\cos t, \sin t)$ ; (f)  $r(t) = \cos(5t)(\cos t, \sin t)$ ; (g)  $r(t) = t(\cos t, \sin t)/2$ , (h)  $r(t) = \sin(t/2)(\cos t, \sin t)$ .



Figur 9.13. Grenseovergang fra sekant til tangent.

ulike kjøremønstre langs veien. Vi kan for eksempel kjøre fort eller bruke lang tid, vi kan til og med kjøre et stykke, rygge litt og så kjøre framover igjen, eller vi kan kjøre med konstant fart.

#### 9.4.2 Tangent, hastighet og derivert

Ved de to tidspunktene  $t_1$  og  $t_2$  har vi de to posisjonene  $\mathbf{r}(t_1)$  og  $\mathbf{r}(t_2)$  på kurven vår. Differansen  $\mathbf{r}(t_2) - \mathbf{r}(t_1)$  gir dermed vektoren fra  $\mathbf{r}(t_1)$  til  $\mathbf{r}(t_2)$ , altså sekanten mellom de to punktene, og lar vi  $t_2$  nærme seg  $t_1$  vil sekanten nærme seg tangenten til kurven i  $\mathbf{r}(t)$ . Det viser seg at det er lurt å skalere lengden med forskjellen i tid  $t_2 - t_1$ . Gjør vi dette får vi

$$\lim_{t_2 \rightarrow t_1} \frac{\mathbf{r}(t_2) - \mathbf{r}(t_1)}{t_2 - t_1} = \left( \lim_{t_2 \rightarrow t_1} \frac{f(t_2) - f(t_1)}{t_2 - t_1}, \lim_{t_2 \rightarrow t_1} \frac{g(t_2) - g(t_1)}{t_2 - t_1} \right) = (f'(t_1), g'(t_1)), \quad (9.12)$$

Dette er utgangspunktet for vår neste setning og er illustrert i figur 9.13.

**Setning 9.10.** Anta at  $f$  og  $g$  er deriverbare funksjoner. Den deriverte av den parametriserte kurven  $\mathbf{r}(t) = (f(t), g(t))$  er definert som  $\mathbf{r}'(t) = (f'(t), g'(t))$ . Retningen til den deriverte faller sammen med tangentlinja til kurven i  $\mathbf{r}(t)$  mens lengden av tangenten gir farten til bevegelsen ved dette tidspunktet.

**Bevis.** Utsagnet om retningen til den deriverte følger fra kommentarene foran og relasjonene i (9.12), mens utsagnet om farten trenger litt forklaring. Fra (9.12) har vi

$$|\mathbf{r}'(t_1)| = \lim_{t_2 \rightarrow t_1} \frac{|\mathbf{r}(t_2) - \mathbf{r}(t_1)|}{t_2 - t_1}, \quad (9.13)$$

der notasjonen  $|\mathbf{a}|$  generelt angir lengden til vektoren  $\mathbf{a}$ . Uttrykket  $|\mathbf{r}(t_2) - \mathbf{r}(t_1)|$  er avstanden mellom de to punktene  $\mathbf{r}(t_1)$  og  $\mathbf{r}(t_2)$ , målt i "luftlinje". Når  $t_2$  nærmer seg  $t_1$  vil avstanden etterhvert nærme seg avstanden langs veien gitt ved kurven. Dividerer vi med tiden  $t_2 - t_1$  som det tar å bevege seg fra  $\mathbf{r}(t_1)$  til  $\mathbf{r}(t_2)$  får vi gjennomsnittsfarten

over dette tidsintervallet. Når så lengden av tidsintervallet går mot 0 som i (9.13) får vi farten ved tidspunktet  $t_1$  som lengden av tangenten. ■

Det er vanlig å referere til tangenten  $\mathbf{r}'(t)$  som *hastigheten* til bevegelsen gitt ved parametriseringen  $\mathbf{r}(t)$  mens lengden av tangenten er farten, det vi måler på speedometeret på en bil. Hastigheten gir altså mer informasjon enn farten og viser også hvilken retning vi beveger oss i. Informasjonen vi nå har om tangentretninger er vesentlig når vi skal hekte sammen kurvebiter til større kurver i seksjon 9.5.

### 9.4.3 Anvendelser av parametriske kurver

Parametriske kurver har mange mange anvendelser. De klassiske anvendelsene er innen fysikk, som ett skreddersydd verktøy for å angi partikkelbaner. Med datateknologien har mange nye anvendelser kommet til. Innen datagrafikk er det alltid behov for å kunne representere geometrien som skal vises fram på skjermen, og kurver representeres da som regel på parametriske form som her. Etterhvert har det blitt vanlig også å lage film ved hjelp av datamaskin, der handlingen foregår i en virtuell verden som bare eksisterer inne i datamaskiner. Slike filmer filmes ikke på vanlig måte. I steden produseres hvert bilde digitalt og lagres på disk, basert på hvor det virtuelle kameraet er plassert i den virtuelle verdenen. I en slik sammenheng er det viktig å føre kameraet riktig, og kamerabaner representeres naturlig som parametriske kurver. Legg merke til at i en slik anvendelse er det ikke bare den geometriske kamerabanen som er viktig, det er også av avgjørende betydning hvor fort kameraet beveger seg, med andre ord er det viktig hvilken parametriske representasjon som er valgt blant de mange som alle representerer den samme geometriske banen.

En annen sentral anvendelse av parametriske kurver er innen font-teknologi (en font er en bestemt måte å tegne de ulike tegnene som er tilgjengelig ved trykking, i våre dager de tegnene som datamaskinen kan tegne). Omrisset av hver bokstav representeres ved en eller flere parametriske kurver, og når bokstaven skal tegnes på skjerm eller papir bestemmer maskinen hvilke punkter som skal skrues av eller på for å vise bokstaven på en pen måte. Vi skal se litt nærmere på dette i seksjonen om Bezier-kurver.

## 9.5 Bezier-kurver

Eksempelene i seksjon 9.4 illustrerte forhåpentligvis at vi kan representere et stort spekter av geometriske kurver ved hjelp av parametriske representasjoner. Ved å lagre formelen i en datamaskin med passende programvare kan vi alltid tegne den geometriske formen så pent som utskriftsmediet tillater. Spørsmålet er bare hvordan vi kan finne fram til en parametriske representasjon som gir akkurat den formen vi er ute etter. Vi kan selvsagt prøve oss fram med formler av typen i figur 9.12, men det blir som regel fomling mer eller mindre i blinde.

Det vi trenger er en samling av enkle, men fleksible, parametriske representasjoner som inneholder frie koeffisienter som kan justeres slik at vi kan få fram ulike geometriske former. Vi kan da bruke teknikker som interpolasjon og minimering av feil til å beregne gode tilnærminger til gitte geometriske former. Hvis de frie koeffisientene har intuitive

geometriske tolkninger, kan vi også gjøre bruk av disse basisrepresentasjonene til interaktivt å bygge opp en ønsket kurve ved å manipulere koeffisienter. Det viser seg at en generalisering av Bernstein-polynomene har de ønskede egenskapene.

**Definisjon 9.11.** En Bezier-kurve av grad  $n$  i planet er en parametrisk kurve på formen

$$\mathbf{p}(t) = \mathbf{c}_0 b_{0,n}(t) + \mathbf{c}_1 b_{1,n}(t) + \cdots + \mathbf{c}_n b_{n,n}(t), \quad t \in [a, b],$$

der koeffisientene  $(\mathbf{c}_i)_{i=0}^n$  er punkter i planet og  $\{b_{i,n}\}_{i=0}^n$  er Bernstein-basisen på intervallet  $[a, b]$  gitt ved

$$b_{i,n}(t) = \binom{n}{i} \left(\frac{t-a}{b-a}\right)^i \left(\frac{b-t}{b-a}\right)^{n-i}.$$

Koeffisientene kalles også kontrollpunktene til  $\mathbf{p}$  og den brudne linja som framkommer ved å forbinde kontrollpunktene med rette linjer kalles kontrolpolygonet til  $\mathbf{p}$ .

Legg merke til at polynomene som inngår er 'vanlige' funksjoner som til hver  $t$  vi putter inn gir ut et reelt tall. Grunnen til at dette blir kurver og ikke funksjoner er at koeffisientene vi multipliserer med ikke lenger er tall, men punkter i planet. Uttrykket  $\mathbf{c}_i b_{i,n}(t)$  betyr vanlig skalar multiplikasjon av en vektor med et tall; hvis  $\mathbf{c}_i = (x, y)$  er  $\mathbf{c}_i b_{i,n}(t) = (x b_{i,n}(t), y b_{i,n}(t))$ . Dette betyr at det er lett å regne ut verdier på en Bezier-kurve. Hvis vi har en kvadratisk Bezier-kurve med kontrollpunkter  $\mathbf{c}_i = (x_i, y_i)$  for  $i = 0, 1, 2$  så har vi

$$\begin{aligned} \mathbf{p}(t) &= \mathbf{c}_0 b_0(t) + \mathbf{c}_1 b_1(t) + \mathbf{c}_2 b_2(t) \\ &= (x_0 b_0(t) + x_1 b_1(t) + x_2 b_2(t), y_0 b_0(t) + y_1 b_1(t) + y_2 b_2(t)). \end{aligned}$$

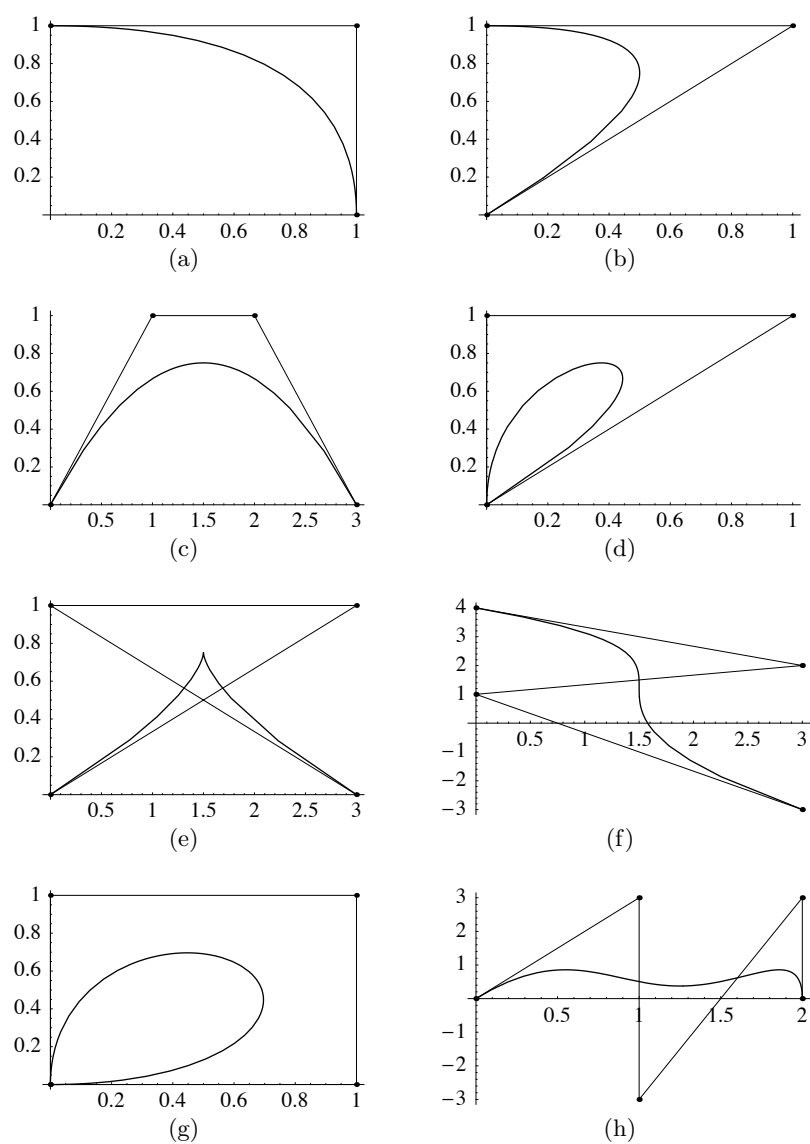
En slik oppspalting kan vi gjøre for generell grad, så det å beregne et punkt på en Bezier-kurve i planet er derfor ikke noe annet enn å beregne verdien av to Bernstein-polynomer, ett for  $x$ -koordinaten og ett for  $y$ -koordinaten.

Vi ser at definisjonen av kontrollpunkter er enklere for kurver enn for funksjoner. For funksjoner måtte vi finne en  $x$ -verdi å assosiere hver koeffisient med, det er ikke lenger nødvendig siden koeffisientene er punkter i planet.

Bezier-kurven av grad 1 gitt ved  $\mathbf{p}(t) = \mathbf{c}_0 b_{0,1}(t) + \mathbf{c}_1 b_{1,1}(t) = (1-t)\mathbf{c}_0 + t\mathbf{c}_1$  gir linjesegmentet som starter i  $\mathbf{c}_0$  og ender i  $\mathbf{c}_1$ . En del eksempler på Bezier-kurver av høyere grad er vist i figur 9.14, og ikke overaskende ser vi at disse også interpolerer sitt første og siste kontrollpunkt. Vi ser i tillegg at tangentretningene i endene er gitt ved retningen til kontrolpolygonet i endene. Dette følger av de grunnleggende egenskapene ved Bernstein-basisen i lemma 9.3.

**Setning 9.12.** En Bezier-kurve  $\mathbf{p}(t) = \sum_{i=0}^n \mathbf{c}_i b_i(t)$  har verdiene  $\mathbf{p}(a) = \mathbf{c}_0$  og  $\mathbf{p}(b) = \mathbf{c}_n$  i endepunktene. Tangenten til kurven i  $t = a$  er parallell med linja gjennom  $\mathbf{c}_0$  og  $\mathbf{c}_1$ , mens tangenten i  $t = b$  er parallell med linja gjennom  $\mathbf{c}_{n-1}$  og  $\mathbf{c}_n$ .

Plottene i figur 9.14 viser at Bezier-kurvene oppfører seg helt tilsvarende Bernstein-polynomene, men vi har i tillegg fått den ekstra friheten som kurver gir i forhold til funksjoner. Samtidig er det slik at for å få mange frihetsgrader å spille på som vi kan utnytte til å få mer kompliserte geometriske former, trenger vi Bezier-kurver av høy grad. Som for funksjoner er dette ikke så lurt med tanke på effektivitet og avrundingsfeil, og vi



**Figur 9.14.** Åtte Bezier-kurver med kontrollpunkter. Kvadratiske ((a)og (b)), kubiske ((c)–(f)), grad 4 ((g) og (h)).

ser fra figurene at sammenhengen mellom kontrollpolygonet og kurven blir mindre tydelig når graden øker.

En god løsning å danne mer kompliserte former ved å hekte sammen flere Bezier-kurver, helt parallelt med konstruksjonen vi gjorde for Bernstein-polynomer.

**Definisjon 9.13.** En sammensatt Bezier-kurve av grad  $n$  på intervallet  $[0, m]$  er en parametrisk kurve på formen

$$\mathbf{p}(t) = \begin{cases} \mathbf{p}_1(t), & \text{for } t \in [0, 1); \\ \mathbf{p}_2(t), & \text{for } t \in [1, 2); \\ \vdots \\ \mathbf{p}_m(t), & \text{for } t \in [m-1, m]; \end{cases}$$

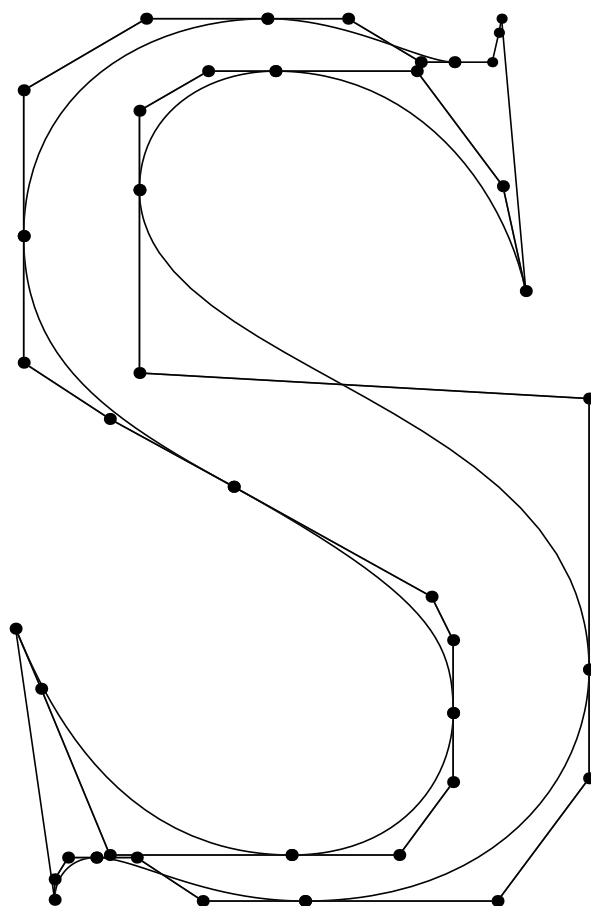
der  $\mathbf{p}_i$  er en Bezier-kurve på intervallet  $[i-1, i]$  for  $i = 1, 2, \dots, m$ . En sammensatt Bezier-kurve kalles ofte bare en Bezier-kurve.

Her har vi antatt at alle delkurvene er definert på intervaller av bredde 1. Det er ingenting å tape på dette hvis vi bare er interessert i den geometriske formen som kurven representerer. Hvis vi for eksempel skal representere kamerabaner må vi være mer forsiktede siden bredden på intervallene sier noe om hvor lang tid vi bruker på å gjennomløpe hver kurvebit og dermed noe om hastigheten vi beveger kameraet med.

På grunn av den enkle sammenhengen mellom kontrollpolygonet og kurven i endene er det enkelt å hekte sammen to Bezier-kurver slik at den sammensatte kurven blir både kontinuerlig og har kontinuerlig tangentretning.

**Setning 9.14.** En sammensatt Bezier-kurve bestående av de to segmentene  $\mathbf{p}(t) = \sum_{i=0}^n \mathbf{c}_i b_i(t)$  og  $\mathbf{q}(t) = \sum_{i=0}^n \mathbf{d}_i b_i(t)$  definert på henholdsvis  $[0, 1]$  og  $[1, 2]$ , vil være kontinuerlig i  $t = 1$  hvis og bare hvis  $\mathbf{c}_n = \mathbf{d}_0$ . Hvis den er kontinuerlig i  $t = 1$  vil tangentretningen være kontinuerlig i  $t = 1$  hvis og bare hvis de tre punktene  $\mathbf{c}_{n-1}$ ,  $\mathbf{c}_n = \mathbf{d}_0$  og  $\mathbf{d}_1$  ligger på en rett linje.

Den matematiske bakgrunnen vi nå har om Bezier-kurver bør være nok til å gi en grov ide om hvordan denne kurvetyper kan utnyttes til å representere geometriske objekter i planet. Det kommersielle programmet *Adobe Illustrator* som ble lansert i 1988 var det første programmet som gjorde Bezier-kurver tilgjengelig for 'massene'. Slike kurver hadde blitt brukt i bil- og flyindustri siden slutten av 1950-tallet til å representere geometrisk form, men da var brukerne ingeniører. Selskapet Adobe sin forretningside var å lage et dataspråk som kunne beskrive alt innhold på en papirside, inklusive tegninger og bokstaver, ved hjelp av matematikk, og til det utviklet de språket *Postscript*. Det grunnleggende primitivet i Postscript er kubiske Bezier-kurver, og Postscript-fonter er bygget opp av slike kurver. Ett eksempel er vist i figur 9.15. Ideen i Adobe Illustrator er å utnytte kubiske Bezier-kurver til frihåndstegning på datamaskin, og programmet er i dag et av de viktigste verktøyene for grafiske designere. Mange skrivere (blant annet på Universitet i Oslo) er basert på Postscript. Dette betyr at hver gang du skriver ut et dokument på en slik skriver må den regne ut en Bezier-kurve for hver eneste bokstav i dokumentet ditt!



**Figur 9.15.** Bezier-kurve med kontrollpolygon som representerer bokstaven 'S' i Postscript-fonten Times-Roman. De ulike Bezier-segmentene ser du mellom kontrollpunktene som ligger på konturen til bokstaven. Legg merke til at den totale kurven også består av noen rette linjestykker.

Dataselskapet Apple utviklet tidlig på 1990-tallet en teknologi for å bryte Adobes monopol på font-området. Denne teknologien kalles *TrueType* og er basert på kvadratiske Bezier-kurver istedenfor kubiske. TrueType er i dag svært utbredt og brukes også av Microsoft.

## 9.6 Tilnærminger til deriverte og integraler

Så langt har vi sett hvordan vi ved hjelp av polynomer og stykkevise polynomer kan finne tilnærminger til mer generelle funksjoner ved hjelp av data (funksjonsverdier og deriverte) hentet fra funksjonene. I denne seksjonen skal vi se på en annen form for approksimasjon, nemlig hvordan vi kan tilnærme én operasjon på funksjoner med en annen operasjon på funksjoner. Eksemplene vi skal se på er hvordan vi kan regne ut tilnærminger til den deriverte og til integralet av en funksjon ut fra funksjonsverdier, såkalt numerisk derivasjon og integrasjon. Vi har vært innom disse temaene tidligere,

men her skal vi også gi estimater for feilen som gjøres. Nøkkelen til det hele er Taylors formel med feilledd.

Hvis vi bare ser på de ulike tilnærmingene for seg, kan både numerisk integrasjon og derivasjon lett synes som en samling med mer eller mindre tilfeldige metoder, og det kan virke nokså uforståelig hvordan noen kunne finne på disse metodene. For å få oversikt i virvaret, er det som alltid viktig å prøve å få øye på noen overordnede prinsipper. Det overordnede prinsippet for både numerisk integrasjon og derivasjon er følgende:

*Erstatt funksjonen  $f$  som skal integreres eller deriveres med et interpolerende polynom  $p$  og integrer eller deriver  $p$  istedenfor  $f$ .*

Under skal vi se nærmere på hvordan dette gjøres i praksis. Stoff om interpolasjon med polynomer finner du i seksjon 9.2.

### 9.6.1 Numerisk integrasjon

I seksjonene 8.7 i Kalkulus og 7.2 i kompendiet har vi allerede sett prinsippet over gjennomført i praksis i forbindelse med integrasjon. Der delte vi integrasjonsområdet  $[a, b]$  inn i delintervaller, tilnærmet  $f$  med et polynom på hvert delintervall, integrerte hvert av disse polynomene og summerte til slutt opp bidragene fra hvert delintervall. Den største utfordringen er dermed hvordan vi skal finne polynomtilnærmingen på hvert delintervall.

I forbindelse med numerisk derivasjon så vi på tre hovedmuligheter: Tilnærming på et delintervall med en konstant funksjon, en lineær funksjon og en parabel. La oss kjapt repetere disse metodene.

**Trapesformelen.** La oss for enkelhets skyld se på det første delintervallet  $[a, a + h]$ . Vi finner først den rette linja  $p_1$  som interpolerer  $f$  i  $a$  og  $a + h$ . Denne kan skrives som

$$p_1(x) = f(a) + \frac{f(a+h) - f(a)}{h}(x - a). \quad (9.14)$$

Deretter bruker vi arealet under denne linja som tilnærming til integralet av  $f$ ,

$$\int_a^{a+h} p_1(x) dx = \frac{h}{2}(f(a) + f(a+h)) \approx \int_a^{a+h} f(x) dx.$$

**Simpsons metode.** Her interpolerer vi med en parabel på hvert delintervall. Siden vi trenger tre punkter for å interpolere med parabler, ser vi på intervallet  $[a, a + 2h]$  og integrerer parabelen  $p_2$  som har samme verdi som  $f$  i punktene  $a$ ,  $a + h$  og  $a + 2h$ . Vi kan skrive denne parabelen slik som i seksjon 9.2.1, men til vårt bruk her er en annen form mer hendig. Hvis punktene vi skal interpolere i er  $x_0$ ,  $x_1$  og  $x_2$ , skriver vi polynomet som  $p_2(x) = b_0 + b_1(x - x_0) + b_2(x - x_0)(x - x_1)$ . I vårt tilfelle er  $x_0 = a$ ,  $x_1 = a + h$  og  $x_2 = a + 2h$ . Interpolasjonsbetingelsene gir da

$$\begin{aligned} f(a) = p_2(a) &= b_0, & f(a+h) = p_2(a+h) &= b_0 + b_1h, \\ f(a+2h) = p_2(a+2h) &= b_0 + 2b_1h + 2b_2h^2. \end{aligned}$$



Fra den første ligningen finner vi  $b_0 = f(a)$ , fra den andre

$$b_1 = \frac{f(a+h) - b_0}{h} = \frac{f(a+h) - f(a)}{h}$$

og fra den tredje får vi

$$b_2 = \frac{f(a+2h) - b_0 - 2b_1h}{2h^2} = \frac{f(a+2h) - 2f(a+h) + f(a)}{2h^2}.$$

Parabelen  $p_2$  kan derfor skrives som

$$p_2(x) = f(a) + \frac{f(a+h) - f(a)}{h}(x-a) + \frac{f(a+2h) - 2f(a+h) + f(a)}{2h^2}(x-a)(x-a-h). \quad (9.15)$$

Integrerer vi  $p_2$  får vi etter litt opprydding

$$\int_a^{a+2h} p_2(x) dx = \frac{h}{3}(f(a) + 4f(a+h) + f(a+2h))$$

som tilnærming til  $\int_a^{a+2h} f(x) dx$ , og dette kjenner vi igjen som Simpsons formel.

**Tilnærming med konstanter.** Integralet av  $f$  er i Kalkulus definert ved hjelp av øvre og nedre trappesummer som jo nettopp består i å tilnærme  $f$  med en konstant på hvert delintervall. For den øvre trappesummen bruker vi, på intervallet  $[a, a+h]$ , verdien av  $f$  i det punktet  $C$  der  $f$  oppnår sitt maksimum på  $[a, a+h]$ . Integraltilnærmingen på dette intervallet blir da

$$hf(C).$$

For den nedre trappesummen bruker vi på tilsvarende måte verdien av  $f$  i punktet  $c$  der  $f$  oppnår sitt minimum på  $[a, a+h]$ , slik at tilnærmingen blir

$$hf(c).$$

Disse tilnærmingene er hendige når vi skal finne fram til egenskaper ved integralet og vise at de øvre og nedre trappesommene konvergerer mot hverandre. Men for praktiske, numeriske beregninger er disse trappesommene som regel til liten hjelp siden vi da må finne maksimum- og minimumverdiene til  $f$  på  $[a, a+h]$ , noe som kan være vanskelig eller til og med umulig om  $f$  er en komplisert funksjon. En mye enklere løsning er å bruke verdien av  $f$  i et fast punkt, for eksempel  $f(a)$ , siden vi da ikke trenger å kaste bort tid på å regne oss fram til det rette punktet. Tilnærmingen til integralet blir da

$$hf(a).$$

Andre muligheter ville være å bruke  $f(b)$  eller  $f((a+b)/2)$  som tilnærming til  $f$  på intervallet  $[a, a+h]$ .

### 9.6.2 Numerisk derivasjon

I seksjon 6.2 tilnærmet vi den deriverte til  $f$  ved hjelp av uttrykket  $f'(a) \approx (f(a+h) - f(a))/h$  som er naturlig ut fra definisjonen av den deriverte. Men med utgangspunkt i ideen vi har brukt for å finne fram til formler for numerisk integrasjon kan vi også finne fram til andre tilnærminger til den deriverte. Framgangsmåten er som før at vi interpolerer  $f$  i noen punkter med et polynom og bruker den deriverte til polynomet som en tilnærming til den deriverte av  $f$ .

Her vil det åpenbart ikke fungere om vi tilnærmer  $f$  med en konstant siden den deriverte av en konstant alltid er 0. La oss derfor forsøke å derivere den rette linja  $p_1$  som interpolerer  $f$  i  $a$  og  $a+h$ . Formelen for  $p_1$  er gitt ved (9.14) og deriverer vi denne får vi

$$f'(a) \approx p_1'(a) = \frac{f(a+h) - f(a)}{h}$$

som er akkurat den samme tilnærmingen vi brukte i seksjon 6.2. Legg merke til at den deriverte av  $p_1$  er konstant siden  $p_1$  er et polynom av grad 1. Vi kan derfor like gjerne bruke tilnærmingen i det andre punktet,

$$f'(a+h) \approx p_1'(a+h) = \frac{f(a+h) - f(a)}{h}.$$

En tredje mulighet er bruke denne tilnærmingen i midtpunktet  $a+h/2$ ,

$$f'(a+h/2) \approx p_1'(a+h/2) = \frac{f(a+h) - f(a)}{h}. \quad (9.16)$$

Denne tilnærmingen har den fordelen at den er mer symmetrisk — vi 'favoriserer' ikke noen av de to gitte punktene når vi bruker tilnærmingen i et punkt som ligger midt mellom dem. Med det vi nå vet om slike tilnærminger har vi ingen grunn til å tro at dette betyr noe, men som vi skal se under viser det seg faktisk å være svært viktig for tilnærmingens nøyaktighet.

Formelen (9.16) brukes vanligvis på formen

$$f'(a) \approx \frac{f(a+h) - f(a-h)}{2h}.$$

Vi ser at vi fremdeles tilnærmer den deriverte til  $f$  midt mellom to gitte punkter  $a-h$  og  $a+h$  ved å ta forholdet mellom forskjellen i funksjonsverdier og avstanden mellom funksjonsverdiene. Alternativt kan vi utlede denne formelen ved å interpolere  $f$  med en rett linje i punktene  $a-h$  og  $a+h$ , derivere denne og bruke tilnærmingen i  $x=a$ .

Ved å bruke flere funksjonsverdier kan vi interpolere  $f$  med polynomer av høyere grad og forhåpentligvis finne fram til andre tilnærminger både til  $f'$  og høyere deriverte. For eksempel ser vi at interpolasjonspolynomet  $p_2$  gitt ved (9.15) gir

$$f'(a) \approx p_2'(a) = \frac{-f(a+2h) + 4f(a+h) - 3f(a)}{2h}.$$

Dette er en mer komplisert formel enn vi fikk fra  $p_1$ , men forhåpentligvis er den også mer nøyaktig. Vi får en mer symmetrisk formel hvis vi bruker  $p_2$  til å finne en tilnærming til  $f'(a+h)$ . Vi får da

$$f'(a+h) \approx p_2'(a+h) = \frac{f(a+2h) - f(a)}{2h}.$$

Dette er bare en forkledd versjon av (9.16) og er altså samme formel som vi kom fram til ved hjelp av  $p_1$ .

Fra  $p_1$  kunne vi ikke finne noen tilnærming til  $f''$  siden  $p_1''$  er 0 overalt. Ved hjelp av  $p_2$  kan vi finne en slik formel,

$$f''(a+h) \approx p_2''(a+h) = \frac{f(a+2h) - 2f(a+h) + f(a)}{h^2}. \quad (9.17)$$

Siden  $p_2''$  er konstant, kunne dette like gjerne vært en tilnærming til  $f''(a)$  eller  $f''(a+h)$ , men som vi har nevnt tidligere er symmetri bra i denne sammenhengen og formelen fungerer best i midtpunktet. Tilnærmingen (9.17) brukes forøvrig som regel på formen

$$f''(a) \approx \frac{f(a+h) - 2f(a) + f(a-h)}{h^2}.$$

Ved å interpolere i flere punkter med polynomer av høyere grad kan vi finne mer kompliserte (og mer nøyaktige) formler og tilnærminger til høyere deriverte. Men det underliggende prinsippet for hvordan vi finner fram til formlene er illustrert med eksemplene over.

### 9.6.3 Feilestimater

Over har vi utledet formler som lar oss regne ut tilnærminger til integralet og den første og andre deriverte til en funksjon  $f$ . Spørsmålet er hvor nøyaktige disse formlene er. Ved hjelp av Taylors formel skal vi nå finne uttrykk for feilen slik at vi kan sammenligne formlene.

Utregningene under kan kanskje virke overveldende og kompliserte. Men hemmeligheten er som vanlig ikke å drukne i detaljene, men se de overordnede prinsippene. Disse skal vi oppsummere mot slutten av seksjonen.

Vi minner om Taylors formel som sier at

$$f(x) = f(a) + (x-a)f'(a) + \frac{(x-a)^2}{2}f''(a) + \cdots + \frac{(x-a)^n}{n!}f^{(n)}(a) + R_n f(x) \quad (9.18)$$

der  $R_n f(x)$  minner om det neste leddet,

$$R_n f(x) = \frac{(x-a)^{n+1}}{(n+1)!} f^{(n+1)}(c),$$

og  $c$  er et tall i intervallet  $[a, c]$  (intervallet  $[c, a]$  om  $x < a$ ) som avhenger av  $x$ . Dette gjelder så sant  $f^{(n+1)}$  og de lavere deriverte alle er kontinuerlige nær  $a$ .

La oss se hvordan vi kan finne et uttrykk for feilen i tilnærmingen

$$f'(a) \approx \frac{f(a+h) - f(a)}{h}. \quad (9.19)$$

Feilen er gitt ved

$$E_1(f, h) = f'(a) - \frac{f(a+h) - f(a)}{h}. \quad (9.20)$$

Ideen er å erstatte  $f(a+h)$  med et Taylorpolynom med restledd med et håp om at en del ledd da vil falle mot hverandre. Vi setter  $x = a+h$ , bruker (9.18) og skriver  $f(a+h)$  som

$$f(a+h) = f(a) + hf'(a) + \frac{h^2}{2}f''(c)$$

der  $c$  er et tall i intervallet  $[a, a+h]$ . Setter vi dette inn i (9.20) får vi

$$E_1(f, h) = f'(a) - \frac{f(a) + hf'(a) + h^2f''(c)/2 - f(a)}{h} = \frac{h}{2}f''(c).$$

Spesielt ser vi at feilen vil gå mot null når  $h$  går mot null.

For lettere å kunne finne fram til dette resultatet i teksten formulerer vi det som en setning.

**Setning 9.15.** Anta at  $f$ ,  $f'$  og  $f''$  er kontinuerlige funksjoner på intervallet  $[a, a+h]$ . Da kan  $f'(a)$  tilnærmes med

$$f'(a) \approx \frac{f(a+h) - f(a)}{h},$$

og feilen i tilnærmingen er gitt ved

$$f'(a) - \frac{f(a+h) - f(a)}{h} = \frac{h}{2}f''(c),$$

der  $c$  er et tall i intervallet  $[a, a+h]$ .

La oss også finne et uttrykk for feilen i den andre tilnærmingen til den deriverte,

$$f'(a) \approx \frac{f(a+h) - f(a-h)}{2h}.$$

Denne feilen er gitt ved

$$E_2(f, h) = f'(a) - \frac{f(a+h) - f(a-h)}{2h}. \quad (9.21)$$

Her trenger vi å uttrykke både  $f(a-h)$  og  $f(a+h)$  ved hjelp av Taylor-polynomer. Vi har

$$f(a+h) = f(a) + hf'(a) + \frac{h^2}{2}f''(a) + \frac{h^3}{6}f'''(c_1), \quad (9.22)$$

$$f(a-h) = f(a) - hf'(a) + \frac{h^2}{2}f''(a) - \frac{h^3}{6}f'''(c_2), \quad (9.23)$$

der  $c_1$  ligger i intervallet  $[a, a + h]$  og  $c_2$  i intervallet  $[a - h, a]$ . Vi legger merke til at leddene her alternerer i fortegn slik at

$$f(a + h) - f(a - h) = 2hf'(a) + \frac{h^3}{6}(f'''(c_1) + f'''(c_2)).$$

Setter vi dette inn i (9.21) faller leddene som involverer  $f'(a)$  mot hverandre og vi står igjen med

$$E_2(f, h) = -\frac{h^2}{12}(f'''(c_1) + f'''(c_2)).$$

Dette uttrykket kan forenkles litt hvis vi ser på tallverdier og godtar en øvre grense for feilen istedenfor et eksakt uttrykk. Vi har nemlig

$$\begin{aligned} |E_2(f, h)| &= \left| \frac{h^2}{12}(f'''(c_1) + f'''(c_2)) \right| \\ &\leq \frac{h^2}{12} (|f'''(c_1)| + |f'''(c_2)|) \\ &\leq \frac{h^2}{12} \left( \max_{c_1 \in [a-h, a+h]} |f'''(c_1)| + \max_{c_2 \in [a-h, a+h]} |f'''(c_2)| \right) \\ &= \frac{h^2}{6} \max_{x \in [a-h, a+h]} |f'''(x)|. \end{aligned}$$

I den første overgangen har vi brukt trekantulikheten mens den andre ulikheten følger når vi tar maksverdier av  $f$ . I den siste likheten har vi bare utnyttet at de to maksverdiene er like. Det hele er oppsummert i følgende setning.

**Setning 9.16.** *La funksjonen  $f$  være definert på intervallet  $[a - h, a + h]$  og anta at både  $f$ ,  $f'$ ,  $f''$  og  $f'''$  er kontinuerlige på dette intervallet. Da kan  $f'(a)$  tilnærmes med uttrykket*

$$f'(a) \approx \frac{f(a + h) - f(a - h)}{2h}$$

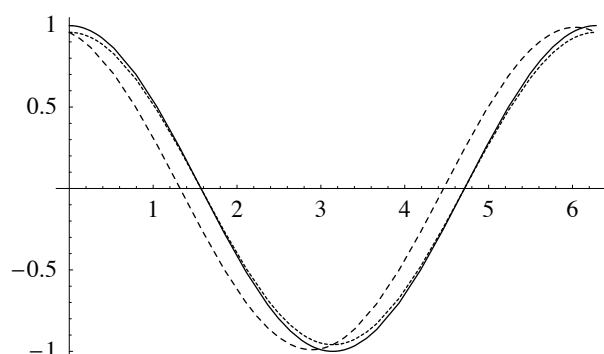
og feilen i denne tilnærmingen er begrenset av

$$\left| f'(a) - \frac{f(a + h) - f(a - h)}{2h} \right| \leq \frac{h^2}{6} \max_{x \in [a-h, a+h]} |f'''(x)|.$$

Legg merke til at feilen her avhenger av  $h^2$ , mens den for metoden i setning 9.15 avhenger av  $h$ . Dette er en vesentlig forskjell. Hvis for eksempel  $h = 10^{-4}$  vil feilen i den første metoden være omtrent  $10^{-4}$  om  $f''$  ikke er for stor, mens feilen i den andre metoden vil være omtrent  $10^{-8}$  så sant  $f'''$  ikke er for stor. En illustrasjon av dette er vist i figur 9.16 der det framgår tydelig at metoden i setning 9.16 gir minst feil. Generelt vil feilen bli mindre jo høyere potens av  $h$  som forekommer i feiluttrykket. Men samtidig vil vanligvis metodens kompleksitet, og dermed nødvendig regnetid, også øke når eksponenten på  $h$  øker.

La oss til slutt utlede et feilestimat for trapesregelen i det enkle tilfellet at vi bare bruker ett linjestykke i tilnærmingen til  $f$ . Formelen er gitt ved

$$\int_a^{a+h} f(x) dx \approx \frac{h}{2}(f(a) + f(a + h)).$$



**Figur 9.16.** Den deriverte av  $\sin x$  og de to tilnærmingene til den deriverte gitt ved setningene 9.15 (grovstiplet) og 9.16 (finstiplet) med  $h = 0.5$ .

Nok en gang benytter vi oss av Taylorutviklinger. I dette tilfellet bruker vi

$$f(x) = f(a) + (x - a)f'(a) + \frac{(x - a)^2}{2}f''(c_1),$$

$$f(a + h) = f(a) + hf'(a) + \frac{h^2}{2}f''(c_2),$$

der  $c_1$  ligger i intervallet  $[a, x]$  og  $c_2$  i intervallet  $[a, a + h]$ . Vi erstatter først  $f(x)$  med denne Taylorutviklingen og integrerer,

$$\begin{aligned} \int_a^{a+h} f(x) dx &= \int_a^{a+h} \left( f(a) + (x - a)f'(a) + \frac{(x - a)^2}{2}f''(c_1) \right) dx \\ &= \left[ (x - a)f(a) + \frac{(x - a)^2}{2}f'(a) \right]_a^{a+h} + \int_a^{a+h} \left( \frac{(x - a)^2}{2}f''(c_1) \right) dx \\ &= hf(a) + \frac{h^2}{2}f'(a) + \int_a^{a+h} \frac{(x - a)^2}{2}f''(c_1) dx. \end{aligned}$$

Igjen ser vi på feilen,

$$E_3(f, h) = \int_a^{a+h} f(x) dx - \frac{h}{2}(f(a) + f(a + h)).$$

Her setter vi inn verdien for integralet som vi fant over samt Taylorutviklingen av  $f(a + h)$ . Dette gir

$$\begin{aligned} E_3(f, h) &= hf(a) + \frac{h^2}{2}f'(a) + \frac{1}{2} \int_a^{a+h} (x - a)^2 f''(c_1) dx \\ &\quad - \frac{h}{2} \left( f(a) + f(a + h) + hf'(a) + \frac{h^2}{2}f''(c_2) \right) \\ &= \frac{1}{2} \int_a^{a+h} (x - a)^2 f''(c_1) dx - \frac{h^3}{4}f''(c_2). \end{aligned}$$

For å få dette på en litt penere form tar vi igjen tallverdier og godtar estimater istedenfor likheter,

$$|E_3(f, h)| \leq \frac{1}{2} \left| \int_a^{a+h} (x-a)^2 f''(c_1) dx \right| + \frac{h^3}{4} |f''(c_2)|.$$

I det følgende kan vi spare litt trykksverte ved å innføre notasjonen  $M = \max_{x \in [a, a+h]} |f''(x)|$ . Det første leddet i uttrykket over estimerer vi til

$$\begin{aligned} \frac{1}{2} \left| \int_a^{a+h} (x-a)^2 f''(c_1) dx \right| &\leq \frac{1}{2} \int_a^{a+h} |(x-a)^2 f''(c_1)| dx \\ &\leq \frac{1}{2} \int_a^{a+h} |(x-a)^2 M| dx \\ &= \frac{M}{2} \int_a^{a+h} (x-a)^2 dx = \frac{Mh^3}{6}. \end{aligned}$$

Den første ulikheten kommer av at integralet aldri kan bli mindre når vi integrerer absoluttverdien av et uttrykk istedenfor uttrykket selv. Den andre ulikheten følger når vi erstatter  $f(c_1)$  med maksimumsverdien av  $f$  på intervallet  $[a, a+h]$ . Dermed blir den totale feilen begrenset av

$$|E_3(f, h)| \leq \frac{Mh^3}{6} + \frac{Mh^3}{4} = \frac{5Mh^3}{12}$$

når vi også erstatter  $f(c_2)$  med maksverdien til  $f$  på  $[a, a+h]$ . Dette gir oss følgende setning.

**Setning 9.17.** *La funksjonen  $f$  være slik at både  $f$ ,  $f'$  og  $f''$  er kontinuerlige på intervallet  $[a, a+h]$ . Da kan integralet av  $f$  på dette intervallet tilnærmes med*

$$\int_a^{a+h} f(x) dx \approx \frac{h}{2} (f(a) + f(a+h))$$

og feilen i denne tilnærmingen er begrenset ved

$$\left| \int_a^{a+h} f(x) dx - \frac{h}{2} (f(a) + f(a+h)) \right| \leq \frac{5h^3}{12} \max_{x \in [a, a+h]} |f''(x)|.$$

Vi understreker at dette feilestimatet bare gjelder for ett delintervall. Bruker vi metoden over flere delintervaller viser det seg at vi mister en potens på  $h$ , men dette skal vi ikke gå inn på her.

Merk også at det i seksjon 8.7 i Kalkulus er oppgitt et feilestimat for trapesmetoden der faktoren  $5/12$  er erstattet med  $1/12$ . Det er mulig å få til dette ved å bruke en litt mer raffinert metode enn vi har benyttet her.

Over har vi utledet estimater for den 'matematiske' feilen i de forskjellige tilnærmingene, og i alle tilfellene går denne mot null når  $h$  går mot null. Men det er viktig å huske på at når disse metodene implementeres på en datamaskin er det en feilkilde til som spiller inn, nemlig avrundingsfeilen. Dette har vi allerede sett eksempel på i seksjon 6.2 når

det gjelder numerisk derivasjon. Dessverre er det generelt slik at for numerisk derivasjon øker avrundingsfeilen med avtagende  $h$ . For numerisk integrasjon er det derimot små problemer med avrundingsfeil. En presis matematisk analyse av avrundingsfeil er langt mer komplisert enn den feilanalysen vi har tatt for oss her.

#### 9.6.4 Generell strategi for å utlede feilestimer

Over utledet vi tre feilestimer med til dels lange utledninger. I slike sammenhenger er det viktig å se den overordnede ideen, ellers er det lett å drukne i detaljene. La oss forsøke å oppsummere strategien vi brukte.

Utgangspunktet er at vi har en tilnærming til den deriverte eller integralet av en funksjon  $f$  nær et punkt  $a$ . For å finne et uttrykk for feilen gjør vi følgende:

1. Skriv opp uttrykket for feilen.
2. Erstatt alle funksjonsverdier unntatt  $f(a)$  med Taylorpolynomer med feilledd, utviklet rundt  $a$ .
3. Trekk sammen alle uttrykkene (og beregn de integralene som kan beregnes direkte). Har du regnet riktig og tatt med passende antall ledd i Taylorutviklingene vil du ende opp med et uttrykk som bare involverer restleddene.
4. Står det igjen mer enn ett restledd, må du ta tallverdi av feilen og bruke trekantulikheten slik at du får en øvre grense som er summen av tallverdiene av restleddene.
5. Ta maksverdier i alle funksjonsuttrykkene over det største intervallet som er involvert i tilnærmingen. Slike maksverdier kan settes utenfor eventuelle integraler slik at de gjenstående integralene lett kan beregnes. Alle feilledd kan så slås sammen til ettuttrykk.

Det er lurt å gå tilbake å se over utledningene over for å gjenkjenne disse stegene.

Det mest mystiske er kanskje hvordan vi kan vite hvor mange ledd som skal tas med i Taylorpolynomene. Tar du med for få ledd vil du til slutt ende opp med et feiluttrykk som inneholder en sum av to eller flere restledd fra Taylorutviklingene der summen av koeffisientene er null. Dette betyr at du kan ta med minst ett ledd til i hvert Taylorpolynom siden disse leddene da vil falle mot hverandre (da får du jo samme argument i alle funksjonsuttrykkene). Tar du med for mange ledd vil du til slutt stå igjen med flere ledd enn restleddene fra Taylorutviklingene i det endelige feiluttrykket. Se oppgave 9 under for konkret øvelse i dette.



## Oppgaver

9.1 Taylor-polynomene til  $e^x$ ,  $\cos x$  og  $\sin x$  er

$$\begin{aligned} e^x &= 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \frac{x^4}{24} + \frac{x^5}{120} + \frac{x^6}{720} + \frac{x^7}{5040} + \dots \\ \cos x &= 1 - \frac{x^2}{2} + \frac{x^4}{24} - \frac{x^6}{720} + \dots \\ \sin x &= x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040} + \dots \end{aligned}$$

Regn ut Taylor-polynomet til  $e^{ix}$  og bruk dette til å forklare at Eulers formel  $e^{ix} = \cos x + i \sin x$  er en rimelig definisjon av  $e^{ix}$ .

9.2 Utled egenskapene 1 og 2 ved Bernsteinbasisen gitt i lemma 9.3. Et nyttig redskap for å bevise egenskap 2 er binomialteoremet.

9.3 Utled egenskap 3 i lemma 9.3.

9.4 Utled egenskap 4 og 5 i lemma 9.3.

9.5 Utled egenskapene 1–3 i lemma 9.5 for Bernstein-polynomer av generell grad  $n$  på intervallet  $[0, 1]$ .

9.6 Sjekk at relasjonen

$$\binom{n}{i} = \frac{n-i+1}{i} \binom{n}{i-1}$$

stemmer.

9.7 a) Vis at polynomet  $x$  kan uttrykkes i Bernsteinbasisen som

$$x = \sum_{i=0}^3 \frac{i}{3} b_{i,3}(x).$$

Hint: Sjekk at de deriverte på de to sidene av likhetstegnet er like og at de to sidene har samme verdi i ett punkt.

b) Bruk (a) til å vise relasjonen

$$(x, p(x)) = \sum_{i=0}^3 (i/3, c_i) b_{i,3}(x).$$

(Dette må tolkes som en vektorrelasjon der venstresiden  $(x, p(x))$  og kontrollpunktene  $(i/3, c_i)_{i=0}^3$  er punkter i planet, mens  $b_{i,3}(x)$  for hver  $x$  er en skalar.) Denne relasjonen viser at grafen til  $p(x)$  er et veiet gjennomsnitt av sine kontrollpunkter.

- 9.8 Programmer algoritme 9.8 og test programmet ved å plote ut noen av Bernstein-polynomene i figur 9.7 og 9.8.
- 9.9 I denne oppgaven skal vi gjenta utregningene som førte oss fram til setning 9.16, men med flere eller færre ledd i Taylorpolynomet.
- Ta med et ledd mindre i hver av de to Taylorutviklingene (9.22)–(9.23) (slik at restleddene involverer  $h^2$ ), men gjennomfør ellers utregningene som vist over. Hvordan kan du gjenkjenne at dette kan gjøres bedre?
  - Ta med et ledd mer i hver av de to Taylorutviklingene (slik at restleddene involverer  $h^4$ ), men gjennomfør igjen utregningene som over. Hvordan kan du se at du har tatt med for mange ledd?

9.10 Vis at feilen  $E(f, h)$  i tilnærmingen til den førstederiverte gitt ved

$$f'(a) \approx \frac{-f(a+2h) + 4f(a+h) - 3f(a)}{2h}$$

er begrenset av

$$|E(f, h)| \leq h^2 \max_{x \in [a, a+2h]} |f'''(x)|.$$

9.11 Utled et feilestimat for tilnærmingen til den andrederiverte gitt ved

$$f''(a) \approx \frac{f(a+h) - 2f(a) + f(a-h)}{h^2}.$$

9.12 Utled et feilestimat for tilnærmingen til integralet gitt ved

$$\int_a^{a+h} f(x) dx \approx hf(a+h/2).$$

Hint: Erstatt  $f(x)$  med en passende Taylorutvikling om  $x = a + h/2$ .