HANDLEDNING TILL LABORATION I GEOMETRI

NIELS CHR. OVERGAARD

1. Inledning

Denna laboration består av två delar, en om interpolationstekniker och en annan om bézierritning (som man kan kalla en designteknik). Labhandledningen innehåller lite teori, vissa obevisade fakta och enstaka övningar. För att bli godkänd på laborationen skall du redovisa lösningar till alla numrerade övningar (d.v.s. när det står **Övning 37:** ..., t.ex.).

2. Starta upp

Laborationen och denna laborationshandledning finns utlagd på kursens hemsida på nätet:

`www.maths.lth.se/matematiklth/personal/ufn/geometri/`

Gå till denna hemsida och ladda ner `laborationen`. Du skall spara filen i din egen matlab–katalog. Har du ingen sådan katalog skapar du lätt en med kommandot

```
mkdir matlab
```

Ställ dig sen i matlab–katalogen. Filen som innehåller de relevanta matlabfilerna är båda packad och komprimerad. Först skall vi dekomprimera filen, vilket sker med kommandot

```
gzip -d geolabb.tar.gz
```

Därnäst packar du upp filen genom att skriva

```
tar xvf geolabb.tar
```

Skriv sedan `ls`, då skulle du gärna se en ny katalog `laboration/`. Gå till denna katalogen med `cd laboration`. Nu är du klar att påbörja laborationen. Starta först Matlab:

```
matlab
```

När du ser prompten

```
>>
```

är Matlab redo att ta emot dina kommandon. Eftersom att jag antar att du redan har läst igenom handledningen en gång, kan du nu gå till avsnitt 4, där laborationens första del finns (denna mening utelämnas vid första genomläsningen!).

3. Lagrange interpolation

Antag att man samplar en kontinuerlig funktion $f \in C^0([a,b])$ vid ett antal tillfällen $a = t_0 < t_1 < \ldots < t_n = b$. Efter samplingen inskränks vår kännedom om f till **noderna** $\{(t_i, f(t_i)\}_{i=0}^n$. Hur skall man gå till väga för att återskapa f? Svaret är enkelt: Det går inte! Det inses lätt eftersom man kan tänka sig många kontinuerliga funktioner vars grafer passerer genom punkterna $\{(t_i, f(t_i)\}_{i=0}^n$. Man får alltså formulera om problemet för att det skall bli vettigt, till exempel: kan man enkelt skapa en kontinuerlig funktion som går genom noderna $\{(t_i, f(t_i)\}_{i=0}^n$? Eftersom polynom är speciellt enkla kan man till och med fråga sig, om det går att hitta ett polynom (gärna entydigt bestämt) som passerar noderna $\{(t_i, f(t_i)\}_{i=0}^n$? Svaret på denna fråga är ja!, vilket vi strax skall se. Men först behöver vi lite terminologi.

Låt i fortsättningen $\mathcal{P}_n$ beteckna mängden av alla polynom av grad $\leq n$, dvs

$$\mathcal{P}_n = \Big\{ \sum_{k=0}^{n} a_k t^k : a_k \in \mathbf{C} \Big\}.$$

Observera att $\mathcal{P}_n$ bildar ett vektorrum, eftersom $P, Q \in \mathcal{P}_n$ och $\lambda \in \mathbf{C}$ medför att $P + Q \in \mathcal{P}_n$ och $\lambda P \in \mathcal{P}_n$. Dimensionen av vektorrummet $\mathcal{P}_n$ är $n+1$ (**Övning:** Hur ser man detta?).

Sats 1. *Givet noder* $\{(t_i, f(t_i)\}_{i=0}^n$, *där* $a = t_0 < t_1 < \ldots < t_n = b$ *och* $f \in C^0([a,b])$. *Då finns det ett entydigt bestämt polynom* $P \in \mathcal{P}_n$ *sådant att* $P(t_i) = f(t_i), i = 0, 1, \ldots, n$.

Bevis: Ni har säkert redan sett ett bevis för denna satsen på föreläsningen. Här skall jag ge ett annat och lite mera abstrakt bevis, som illustrerar styrkan av linjär algebra. Betrakta den linjära avbildningen $L : \mathcal{P}_n \to \mathbf{R}^{n+1}$ som ges av

$$L(P) = (P(t_0), P(t_1), \ldots, P(t_n)).$$

Avbildningen L samplar alltså polynomet P i punkterna $t_i, i = 0, 1, \ldots, n$ (**Övning:** kontrollera att avbildningen faktiskt är linjär). Påståendet i satsen är att det i vektorrummet $\mathcal{P}_n$ finns precis ett polynom P så $L(P) = (f(t_0), f(t_1), \ldots, f(t_n)) \equiv \mathbf{f}$.

Antag först att vi redan vet att det existerar ett P sådant att $L(P) = \mathbf{f}$. Då kommer P att vara entydigt bestämt, för antag att det finns ett annat polynom $Q \in \mathcal{P}_n$ med $L(Q) = \mathbf{f}$. Då har vi $P(t_i) = Q(t_i), i = 0, 1, \ldots, n$ eller ekvivalent

$$P(t_i) - Q(t_i) = 0, \quad i = 0, 1, \ldots, n.$$

Från endimensionella analysen vet vi att ett polynom av grad n, som inte är identiskt noll, högst kan ha n nollställen (Persson/Böiers: Analys i en variabel, faktorsatsen, sidan 25). Men som vi såg ovan har $P - Q$ åtminstone $n+1$ nollställen. Som graden av $P - Q$ är $\leq n$ följer det härav att $P - Q \equiv 0$ eller $P = Q$.

Att lösningen till interpolationsproblemet är entydigt bestämt är det samma som att säga att den linjära avbildningen L är *injektiv* (dvs en-entydig). Som $\dim \mathcal{P}_n = \dim \mathbf{R}^{n+1} = n+1$ följer det av huvudsatsen för linjära avbildningar (Sparr: Linjär algebra, sidan 197) att L automatiskt är *surjektiv*, dvs att alla punkter i $\mathbf{R}^{n+1}$ träffas under avbildningen L, speciellt finns det ett P så $L(P) = \mathbf{f}$. Därmed är beviset klart. Observera att vi fick existensen (nästan) gratis genom att visa entydigheten (snyggt!). □

I fortsättningen skall vi beteckna P i satsen med $P(f|t_0, t_1, \ldots, t_n)$ för att visa att P interpolerar funktionsvärderna för f i punkterna $t_i, i = 0, 1, \ldots, n$.

Att interpolationspolynomet är entydigt bestämt av noderna är viktigt. Vi skall ge två tillämpningar av detta. Först skall vi bevisa att interpolation är en linjär avbildning från $C^0([a,b])$ till $\mathcal{P}_n$: Om f, g är kontinuerliga funktioner på $[a,b]$, så gäller att $P(f|t_0, t_1, \ldots, t_n)(t_i) = f(t_i), \forall i$ och $P(g|t_0, t_1, \ldots, t_n)(t_i) = g(t_i), \forall i$ och därför

$$P(f|t_0, t_1, \ldots, t_n)(t_i) + P(g|t_0, t_1, \ldots, t_n)(t_i) = f(t_i) + g(t_i), \quad i = 0, 1, \ldots, n.$$

Som vänsterledet i denna ekvation tillhör $\mathcal{P}_n$ och interpolerar $\{(t_i, f(t_i) + g(t_i))\}_{i=0}^n$, så följer det av entydigheten att

$$P(f+g|t_0, t_1, \ldots, t_n) = P(f|t_0, t_1, \ldots, t_n) + P(g|t_0, t_1, \ldots, t_n). \tag{1}$$

Entydigheten kan också användas till att bevisa en rekursions formel for interpolerande polynom:

Sats 2 (Aitkens lemma). *Om $f \in C^0([a,b])$ och $a = t_0 < t_1 < \ldots < t_n = b$ så gäller:*

(2)

$$P(f|t_0, \ldots, t_n)(t) = \frac{(t_n - t)P(f|t_0, \ldots, t_{n-1})(t) + (t - t_0)P(f|t_1, \ldots, t_n)(t)}{t_n - t_0}.$$

Bevis: Se övning 1 nedan. □

Vi noterar att $P(f|t_0) = f(t_0), \ldots, P(f|t_n) = f(t_n)$, varför man kan ta fram $P(f|t_0, \ldots, t_n)$ mha Aitkens lemma genom att räkna igenom följande schema från vänster mot höger (här är $n = 3$ för överskådlighetens skull):

$$\begin{array}{lllllll}
P(f|t_0) & & & & & & \\
 & \searrow & & & & & \\
P(f|t_1) & \to & P(f|t_0,t_1) & & & & \\
 & \searrow & & \searrow & & & \\
P(f|t_2) & \to & P(f|t_1,t_2) & \to & P(f|t_0,t_1,t_2) & & \\
 & \searrow & & \searrow & & \searrow & \\
P(f|t_3) & \to & P(f|t_2,t_3) & \to & P(f|t_1,t_2,t_3) & \to & P(f|t_0,t_1,t_2,t_3)
\end{array}$$

Denna tablå kallas **Nevilles schema**. I denna laboration har Neville schemat implementerats i Matlab (`neville.m`) och används till att beräkna interpolerande polynom.

Övningar:
1. Bevisa Aitkens lemma. (Ledning: Använd tekniken från beviset av sats 1 till att bevisa att polynomen i höger– och vänsterleden är identiska.)
2. Låt $t_0 < t_1 < t_2 < t_3$ vara givna. Beräkna det interpolerande polynomet för noderna $\{(t_0, 0), (t_1, 1), (t_2, 0), (t_3, 0)\}$ mha Nevilles schema. Vad blir svaret? Har du sett liknande förut, t.ex. på en utav föreläsningerna?
3. Givet punkter $p_0, p_1, \ldots, p_n$ i planet $\mathbf{R}^2$. Hur skulle man kunna gå till väga för att konstruera en polynomiell kurva som passerar genom alla dessa punkter i den angivna ordningen (se figur 1).

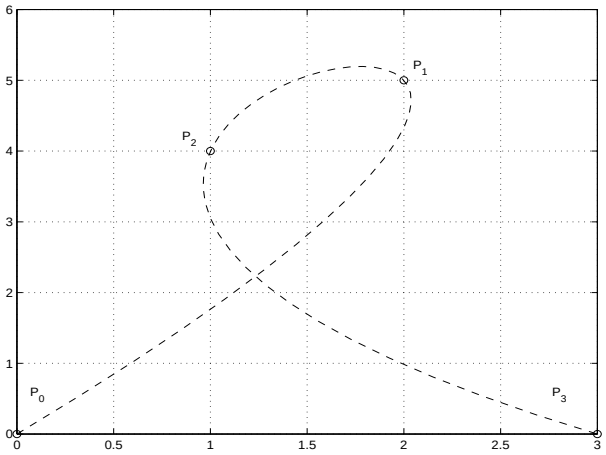

FIGURE 1. Exempel på en polynomiell tredjegradskurva $P(t) = (P_1(t), P_2(t))$ genom fyra punkter p_0, p_1, p_2, p_3 i planet.

4. Instabilitet av Lagrange interpolation

Här börjar första delen av laborationen. Vi skall först sampla funktionen $f(t) = \sin(2\pi t), 0 \leq t \leq 1$. Kommandot

```
>> T=0:0.05:1
```

ger oss en ekvidistant indelning $0 = t_0 < t_1 < \ldots < t_{20} = 1$ av intervallet $[0, 1]$. (Som alternativ kan man också använda kommandot `T=linspace(0,1,21)` till att skapa en ekvidistant indelning av $0 \leq t \leq 1$ som har 21 punkter).

```
>> F=sin(2*pi*T);
```

samplar $\sin(2\pi t)$ i punkterna t_i. Semikolon (;) efter kommandot ovan gör att svaret ej skrivs ut på skärmen. Om du ändå önskar att se `F` skall du skriva

```
>> F
```

och slå `enter`. För att beräkna och rita $P(f|t_0,\ldots,t_n)$ använder vi programmet `interpol.m` (**Övning:** Vilken grad kommer P att få?). Skriv

```
>> interpol(T,F)
```

Detta ger en figur där interpolationskurvan ritas i rött och den samplade kurvan i blått. Som man ser är polynomet snyggt anpassat till $\sin(2\pi t)$. Vi skall ändra `F` lite grand, och på så sätt härma att man har samplat en annan kontinuerlig funktion som ligger nära $\sin(2\pi t)$. Skriv först

```
>> F(11)
```

för att kolla värdet på $\sin(2\pi t_{10})$. Ändra sen detta värde genom att skriva

```
>> F(11)=0.01;
```

och kör `interpol.m` precis som förut. Vad hände? Författaren känner sig övertygad om att detta är en slående illustration av vad som kan inträffa om man interpolerar med polynom av hög grad.

Forsök också gärna att rita Lagranges basfunktioner $L_i(t) = \prod_{j\neq i}(t-t_j)/(t_i-t_j)$, genom att ta `F=[0 0 ... 1 ... 0]`, där ettan är på den i'te platsen. Om du har gott om tid kan också studera **Runges exempel** $f(t) = 1/(1+t^2); -5 \leq t \leq 5$. T.ex. kan man prova att interpolerar med olika indelningar och olika gradtal (Observera att för gradtal > 50 kan beräkningarna ta ganska lång tid).

Lite teori: Antag att man samplar en funktion $f \in C^0([a,b])$ i flera och flera punkter, $a = t_0 < t_1 < \ldots < t_n = b$ där $t_{i+1} - t_i = (b-a)/n, 0 \leq i < n$, och beräknar det motsvarande interpolationspolynom $P(f|t_0,\ldots,t_n)$. Kommer då

$$P(f|t_0,\ldots,t_n) \longrightarrow f, \quad \text{likformigt}^1\text{då } n \to \infty?$$

Om man gör detta för $\sin(2\pi t)$ verkar allting att fungera bra (i själva verket går det bra när f har en maclaurinutveckling som konvergerar mot $f(t)$ för alla t). Men exemplet ovan antyder att läget är mindre gynnsamt om man betraktar någon annan kontinuerlig funktion. Svaret på frågan ovan visar sig också vara nekande: Det finns kontinuerliga funktioner som inte kan återskapas genom att man interpolerar med polynom i flera och flera punkter. Detta hänger ihop med att maximumvärdet av Lagranges basfunktioner $L_i(t) = \prod_{j\neq i}(t-t_j)/(t_i-t_j)$ växer ut över alla gränser när antalet punkter i indelningen $\to \infty$. Vi

[1]Likformigt vill säga att $\max_{a\leq t\leq b}|P(f|t_0,\ldots,t_n)(t) - f(t)| \to 0$ då $n \to \infty$, alltså att den största avvikelsen mellan de två funktionerna går mot noll.

kan inte gå in på beviset här eftersom att det kräver kunskap om grunderna i funktionalanalys (Helt exakt: Teorin för linjära avbildningar på normerade rum och Banach–Steinhaus' sats, se t.ex. Kreyszig: Introductory Functional Analysis with Applications, section 4.7, eller anlita institutionens kurs i funktionalanalys).

Ett exempel på en kontinuerlig funktion, som inte kan approximeras m.h.a. interpolationspolynom, på ovanstående sättet, är just Runges exempel. Man kan bevisa att

$$P((1+t^2)^{-1}|t_0,\ldots,t_n)(s) \longrightarrow (1+s^2)^{-1} \quad \text{då } n\to\infty,$$

om $|s| < 3.63\ldots$ (ok så långt!), men att interpolationspolynomen divergerar för alla s så att $|s| > 3.63\ldots$. Ett annat slående exempel är **Bernsteins exempel**, där man interpolerar funktionen $f(t) = |t|; -1 \le t \le 1$ med ekvidistanta samplingspunkter. I detta fallet kommer man att ha divergens i alla punkter förutom $t = -1, 0, 1$. Prova gärna att rita upp några interpolationer av Bernsteins funktion.

5. Interpolation med kubiska splines

Betrakta återigen en indelning $\Delta : a = t_0 < t_1 < \ldots < t_n = b$ av intervallet från a till b. Med en spline av ordning k förstås en funktion s på $[a,b]$ sådan att:
(i) s är $k-2$ gångar kontinuerligt deriverbar (d.v.s. $s \in C^{k-2}(a,b)$).
(ii) På varje delintervall $[t_i, t_{i+1}], i = 0,\ldots,n-1$ är s ett polynom av grad $\le k-1$ (m.a.o. $s|_{[t_i,t_{i+1}]} \in \mathcal{P}_{k-1}$).

Lägg märke till att man behöver k parametrar för att specifiera ett polynom av grad $k-1$. Det är därför som man säger att s är en spline av ordning k.

Mängden av splines av ordning k på $[a,b]$ betecknas med $\mathcal{S}_{k,\Delta}(a,b)$. Man ser lätt att $\mathcal{S}_{k,\Delta}(a,b)$ är ett vektorrum (**Övning:** Vad är dimensionen av detta vektorrummet?). Observera att $\mathcal{P}_{k-1} \subset \mathcal{S}_{k,\Delta}(a,b)$. Om $k = 4$ är polynombiterna av grad 3 och man kallar $s \in \mathcal{S}_{4,\Delta}(a,b)$ för en **kubisk spline**. I det följande skall vi endast arbete med kubiska splines.

Dimensionen av $\mathcal{S}_{4,\Delta}(a,b)$ är $n+3$, där $n+1$ är antalet av punkter i indelningen Δ. De såkallade **B–splines** är $n+3$ stycken kubiska splines som bildar en bas för vektorrummet $\mathcal{S}_{4,\Delta}(a,b)$. I figur 2 nedan visas B–splinebasen då $[a,b] = [0,1]$ och $\Delta = \{0, 0.1, 0.2, \ldots, 1\}$.

B–spline basfunktionerna betecknas med $N_{4,i}, i = 1,\ldots,n+3$ (fyran eftersom att $k = 4$). Varje kubisk spline s kan skrivas

$$s(t) = \sum_{i=1}^{n+3} d_{i-2} N_{4,i}(t),$$

där koordinaterna $d_{-1}, d_0, \ldots, d_n, d_{n+1}$ är de såkallade **de Boor punkterna**.

För att illustrera detta interpolerar vi återigen sinus:

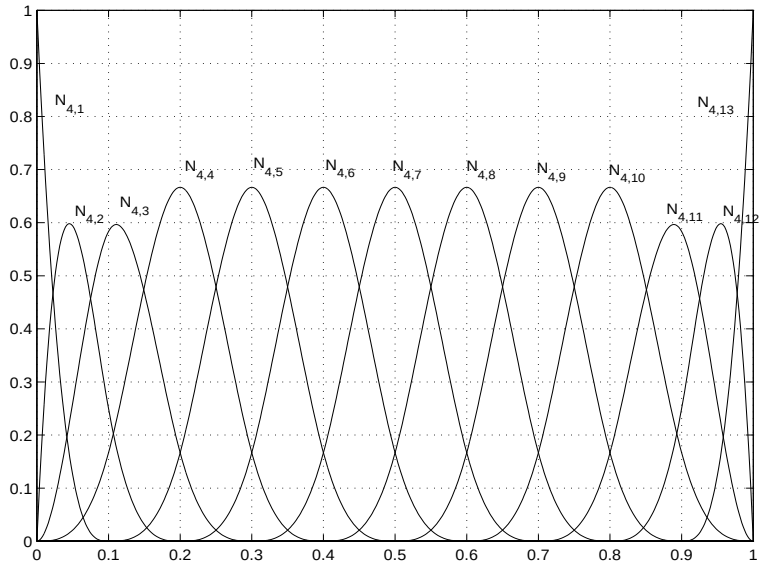

FIGURE 2. B–spline basfunktionerna $N_{4,i}, i = 1, \ldots, 10 + 3$ associerad med indelningen $\Delta = \{0, 0.1, 0.2, \ldots, 1\}$.

```
>> T=0:0.1:1;
>> F=sin(2*pi*T);
>> splineinterpol(T,F);
```

Detta ger en bra överensstämmelse med sinus. För att få ett intryck av detta kan du rita grafen för sinus–kurvan:

```
>> hold on %gör att den befindtliga grafen inte suddas ut när
           %den nya ritas. Hold on slås från med 'hold off'.
>> W=0:0.01:1; G=sin(2*pi*W); plot(W,G,'r--');  %ritar sinus.
```

Du kan granska de två graferna mera noga genom att skriva `zoom` och slå `enter`. Sen kan du mha vänster musknapp dra fram en ram kring det området som du vill förstora. Du kan sudda figuren genom att skriva

```
>> clf
```

Precis som för skall vi ändra `F`:

```
>> F(6)=0.1;
```

Observera att den nya ändringen är tio gångar större än den vi gjorde i förra exemplet.

```
>> splineinterpol(T,F);
```

Vad hände? Ser du att vår störning endast gav utslag lokalt och nära den punkten där störningen skedde? Detta hänger ihop med att varje B–spline basfunktion endast är nollskiljd på fyra av delintervallen i Δ (därför att $k = 4$). Exemplet skall illustrera att spline interpolation är stabil i motsats till Lagrange interpolation.

6. Bézierkurvor

Detta är den tredje delen av laborationen. Tanken är att illustrera kurvritning med hjälp av bézierteknik. För att rita en bézierkurva med kontrollpunkterna $b_0, b_1, \ldots, b_n$ använder man **de Casteljaus algoritm**:

$$b_i^k(t) = (1-t)b_i^{k-1}(t) + t\, b_{i+1}^{k-1}(t), \quad i = 0, \ldots, n-k,$$

där $b_i^0(t) \equiv b_i, i = 0, \ldots, n$. Algoritmen kan skrivas på en schemaform som påminner om Nevilles schema (för överskådlighetens skull tar vi $n = 3$).

$$\begin{array}{ccccccc}
b_0 & & & & & & \\
 & \searrow & & & & & \\
b_1 & \to & b_0^1 & & & & \\
 & \searrow & & \searrow & & & \\
b_2 & \to & b_1^1 & \to & b_0^2 & & \\
 & \searrow & & \searrow & & \searrow & \\
b_3 & \to & b_2^1 & \to & b_1^2 & \to & b_0^3
\end{array}$$

Låt oss börja med att rita något enkelt. Ta till exempel fyra bézierpunkter:

```
>>   b=[ 0 1 2 3 ; 0 2 2 0 ]

b =

     0     1     2     3
     0     2     2     0
```

Här skrivas bézierpunkterna som kolonner i matrisen `b`.

```
>> bezier(b);
```

ger oss nu en figur med bézierkurvan i blått och kontrollpolygonen i rött. Prova också med punkterna

```
>> b=[ 0 3 0 3 ; 0 3 3 0 ];
```

vilket ger en spets (eng.: cusp), eller med

```
>> b=[ 0 3.5 -0.5 3 ; 0 3 3 0 ];
```

som genererar en ögla. Prova även att konstruera bézierkurvor av högre ordning, alltså med flera kontrollpunkter, tex genom först att skissa någon (inte alldeles för komplicerad) kurva på papper, gissa vilka Bézierpunkter som behöves för att alstra denna, och använda sedan `bezier.m` för att se om det stämmer.

Övningar:

4. Betrakta den kurvan P med parameterformen

$$P : t \mapsto \begin{pmatrix} t \\ t^2 \end{pmatrix}.$$

Beräkna bézierpunkterna för P motsvarande parameterintervallen (a) $0 \leq t \leq 1$ och (b) $-1 \leq t \leq 2$ (troligen inte så svårt!).

5. Betrakta den kurvan P med parameterformen

$$P : t \mapsto \begin{pmatrix} t \\ t^3 - t^2 \end{pmatrix}.$$

Beräkna bézierpunkterna för P motsvarande parameterintervallen (a) $0 \leq t \leq 1$ och (b) $-1 \leq t \leq 2$ (kanske inte fullt lika enkelt!).

7. Underindelning av bézierkurvor

En bézierkurva i planet med kontrollpolygonen $b_0, b_1, \ldots, b_n$ är ett (vektorvärt) polynom $P(t) = (P_1(t), P_2(t))$ av grad n så att $P(0) = b_0$ och $P(1) = b_n$. Med andra ord, så gäller det att funktionen $P : [0, 1] \to \mathbf{R}^2$ interpolerar punkterna b_0 och b_n. Låt nu q_1 och q_2 vara två andra punkter på kurvan $P(t)$. Då finns det nya bézierpunkter $q_1 = c_0, c_1, \ldots, c_n = q_2$ och ett tillhörande polynom $Q(t)$ av grad n, så att (a) $P(t)$ och $Q(t)$ beskriver samma kurva i planet och (b) $Q(0) = c_0, Q(1) = c_n$.

Det är ovanstående egenskap som man utnyttjar vid underindelning av Bézierkurvor. Givet en kurva med kontrollpolygonen $b_0, b_1, \ldots, b_n$, samt ett reellt tal $0 < \lambda < 1$ (ofta $\lambda = 0.5$), beräknar man de två kontrollpolygonerna motsvarande punktparen $(p(0), p(\lambda))$ och $(p(\lambda), p(1))$. Unionen av dessa två polygoner bildar en ny polygon som ligger närmare bézierkurvan än den ursprungliga kontrollpolygonen.

Hur ser dessa nya bézierpunkter ut? Svaret är enkelt: Om man räknar igenom de Casteljaus algoritm med $t = \lambda$, så kommer bézierpunkterna motsvarande kurvan med ändpunkterna $(p(0), p(\lambda))$ att vara $(b_0, b_0^1, b_0^2, b_0^3)$, med andra ord, den översta diagonalen i schemat ovan, läst från vänster mot höger. Bézierpunkterna motsvarande ändpunkterna $(p(\lambda), p(1))$ blir $(b_0^3, b_1^2, b_2^1, b_3)$, d.v.s. den nedersta raden läst från höger mot vänster (**Övning:** Verifiera detta påståendet!).

Om man fortsätter induktivt och underindeler de nya kontrollpolygoner erhåller man fyra polygoner vars union ligger ännu närmare bézierkurvan. Man kan upprepa proceduren ett godtyckligt antal gånger, och får sålunda en svit av polygoner som konvergerar snabbt mot en gränskurva, som just är bézierkurvan. Vi skall illustrera detta med ett exempel:

```
>> subdivide(0)
```

visar ett kontrollpolygon motsvarande en bézierkurva.

```
>> subdivide(1)
```

utför en underindelning av denna kurva, och

```
>> subdivide(2)
```

utförar två underindelningar av den ursprungliga kurvan. Figurerna torde tala för sig själva! Kom ihåg att om du vill sudda ut en figur så ger du bara kommandot `clf`. Underindelning är ett effektivt sätt att rita bézierkurvor, vilket vi utnyttjade i programmet `bezier.m` (se avsnitt 5).

Övningar:

6. Återvända till övning 5(a). Genomföra för hand en underindelning av bézierkurvan P i två lika stora delar ($\lambda = 0.5$).

Lunds Tekniska Högskola, Sölvegatan 18, Box 118, 221 00 Lund.