TANA23 Föreläsning 5

Interpolation med Splinefunktioner

- Linjära och Kubiska splinefunktioner. Feluppskattning.
- Tillämpning: Parametriska kurvor

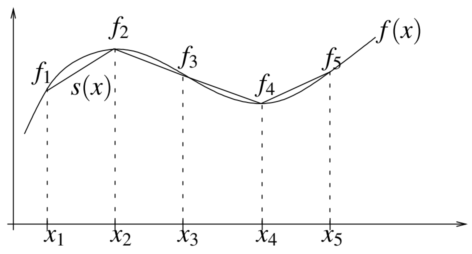
Approximation med Splines

- Beziér kurvor och ytor.
- Tillämpning - Matematisk beskrivning av fonter.

Splineinterpolation

Problem En funktion $f(x)$ är känd endast i vissa *noder* $x_1, x_2, \ldots, x_n$. Hur kan vi hitta en bra approximation $s(x) \approx f(x)$ på intervallet $[x_1, x_n]$?

Lösning Använd linjär interpolation på *varje* delintervall $[x_i, x_{i+1}]$.



Teorin för linjär interpolation gäller!

Linjära splines

Definition En funktion $s(x)$ är en *interpolerande linjär spline*, med *noder* $x_1, \ldots, x_n$ om

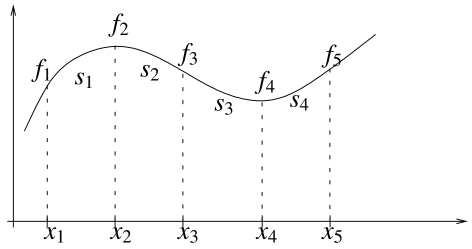
1. $s(x)$ är *kontinuerlig* på $[x_1, x_n]$.
2. $s(x)$ är en rät linje på varje delintervall $[x_i, x_{i+1}]$.
3. $s(x)$ interpolerar $f(x)$ vid noderna, i.e. $s(x_i) = f(x_i)$.

Sats För en *interpolerande linjär spline* gäller att

$$|f(x) - s(x)| \leq \frac{M}{8} h^2,$$

där $|f''(x)| \leq M$ och $h = \max |x_{i+1} - x_i|$.

Kubiskasplines



Definition En funktion $s(x)$ är en *interpolerande kubiskspline*, med *noder* $x_1, \ldots, x_n$ om

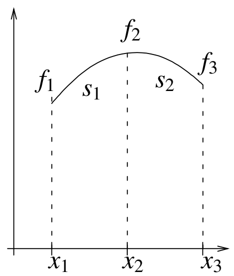
1. $s(x)$, $s'(x)$, och $s''(x)$ är *kontinuerliga* på $[x_1, x_n]$.
2. $s(x)$ är ett tredjegradspolynom på varje delintervall $[x_i, x_{i+1}]$.
3. $s(x)$ interpolerar $f(x)$ vid noderna, i.e. $s(x_i) = f(x_i)$.

Exampel Hitta den *kubiska spline* $s(x)$ som interpolerar tabellen

x	0	1	2
$f(x)$	1	2.5	2

med *randvillkor* $f'(0) = 1$ och $f'(2) = -1$.

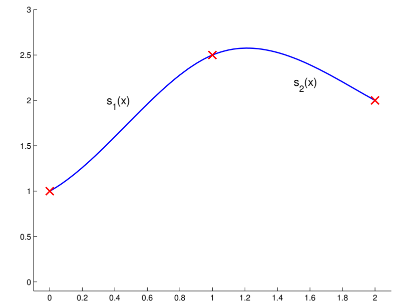
Lösning Hitta två polynom $s_1(x)$ och $s_2(x)$.



Formulera problemet att hitta s_1 och s_2 som ett linjärt ekvationssystem.

Splinefunktionen är

$$s(x) = \begin{cases} s_1(x) = 1.0 + 1.00(x-0) + 1.75(x-0)^2 - 1.25(x-0)^3, & 0 \le x \le 1, \\ s_2(x) = 2.5 + 0.75(x-1) - 2.00(x-1)^2 + 0.75(x-1)^3, & 1 \le x \le 2, \end{cases}$$



Två kontinuerliga derivator och rätt lutning i ändpunkterna.

Randvillkor

Sats En kubiskspline $s(x)$, som interpolerar $f(x)$ vid noderna $x_1, \ldots, x_n$, blir entydigt bestämd om vi tillför två *ändpunktsvillkor* eller *randvillkor*.

Visas genom att fämföra antalet obekanta med antalet villkor.

Vi har följande alternativ

Naturliga villkor $s''(x_1) = s''(x_n) = 0$.
Rätta villkor $s'(x_1) = f'(x_1)$ och $s'(x_n) = f'(x_n)$.
Periodiska villkor $s'(x_1) = s'(x_n)$.

Feluppskattning

Sats Om *rätta ändpunktsvillkor* används gäller att

$$|s(x) - f(x)| \le \frac{5}{384} M h^4,$$

där $h = \max |x_{i+1} - x_i|$ och $M = \max |f^{(4)}(x)|$.

Exempel Approximera $f(x) = 1/(1+x^2)$, på $[-5, 5]$, med en kubisk spline och rätta ändpunktsvillkor. Hur beror felet på antalet noder?

N	5	9	17
h	1/4	1/8	1/16
Felet	0.2714	0.0561	0.0037

Felet beter sig som Ch^4.

I Python paketet `scipy` finns `CubicSpline` som används enligt

```
>>> f = CubicSpline(x, y, bc_type='natural')
```

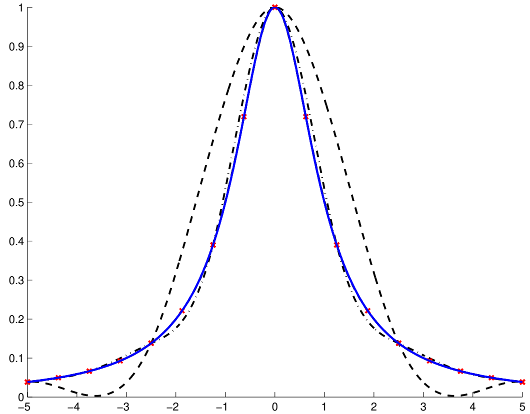
där naturliga randvillkor används. Kan även sätta första derivatan med

```
>>> f = CubicSpline(x, y, bc_type=( (1, 1.7 ), (1, -2.0) )
```

där $s'(x_1) = 1.7$ och $s'(x_n) = -2.0$.

Utparameter är en Python funktion som kan evalueras

```
>>> xx=np.linspace( a , b, 100 )
>>> yy=f( xx )
```

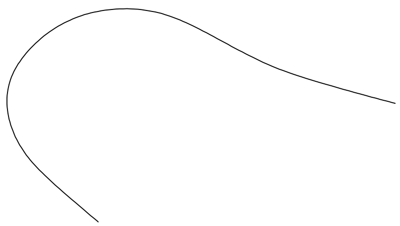



Plot av funktionen $f(x) = 1/(1+x^2)$ approximerad med kubiska splines, där $N = 5$, 9, och 17 noder utnyttjats.

Undviker stora fel mellan interpolationspunkterna!

Ett enkelt ritprogram

De flesta ritprogram, e.g. `xfig`, har en funktion där man ritar en kurva genom att klicka ut ett antal punkter.



Detta är inte en funktionsgraf $y = f(x)$. Vad gör vi istället?

Parametriska kurvor

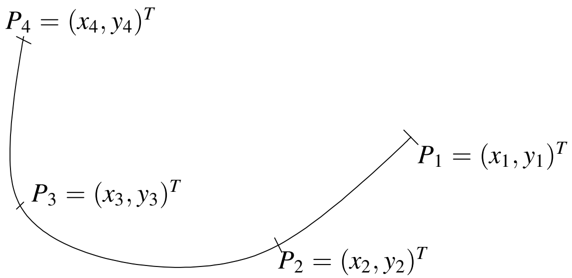
Definition En *parametrisk kurva* i planet skrivs som

$$s(t) = (x(t)\,,\, y(t))^T, \qquad a < t < b.$$

Varje kurva i planet beskrivs av *två* funktioner $x(t)$ och $y(t)$.

För en kurva i $\mathbb{R}^3$ behövs även en funktion $z(t)$.

Exempel Vi vill att en kurva $s(t)$ skall interpolera givna punkter.



Kurvan $s(t)$ skall passera P_1, P_2, P_3, och P_4.

Välj ett parameter intervall $a = t_1 \leq t \leq t_4 = b$ och hitta *två* kubiska splines som interpolerar tabellerna

t	t_1	t_2	t_3	t_4
$x(t)$	x_1	x_2	x_3	x_4

och

t	t_1	t_2	t_3	t_4
$y(t)$	y_1	y_2	y_3	y_4

Then
$$s(t_i) = (x(t_i), y(t_i))^T = (x_i, y_i)^T = P_i.$$

Kurvan $s(t) = (x(t), t(t)^T$, $a \leq t \leq b$, interpolerar de givna punkterna.

Exempel Hitta en kurva som interpolerar punkterna

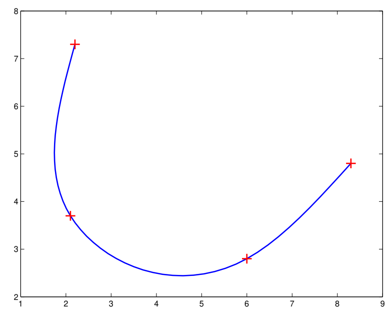
t	0	1	3	5
$x(t)$	8.3	6.0	2.1	2.2
$y(t)$	4.8	2.8	3.7	7.3

med naturliga ändpunktsvillkor.

Givet vektorer t, x aoch y skriver vi

```
>>> px=CubicSpline(t,x, bc_type='natural')
>>> py=CubicSpline(t,y, bc_type='natural')
>>> tt=np.linspace(1,4,100)
>>> pp.plot(px(tt),py(tt))
>>> pp.plot(x,y,'rx')
>>> pp.show()
```

Resultatet är en kurva som interpolerar tabellen



Inverkan av ändpunktsvillkor

Definition Kurvans *tangent* vid punkten $s(t)$ ges av

$$s'(t) = (x'(t), y'(t))^T.$$

Tangenten visar kurvans *lutning* i en viss punkt.

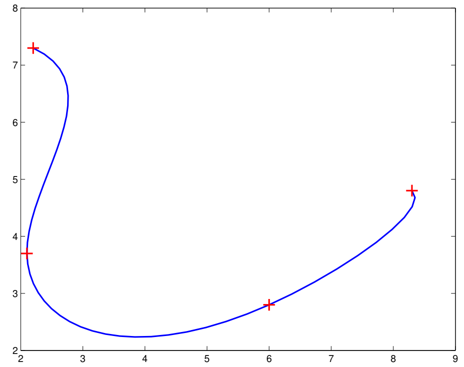
Exempel Använd samma tabell men välj

$$s'(0) = (1, -1)^T, \quad \text{och} \quad s'(5) = (-2, 1)^T.$$

Vad händer?

Givet vektorer t, x och y skriver vi istället

```
>>> px=CubicSpline(t,x,bc_type=((1,1.0 ),(1,-2.0)))
>>> py=CubicSpline(t,y,bc_type=((1,-1.0 ),(1,1.0)))
```

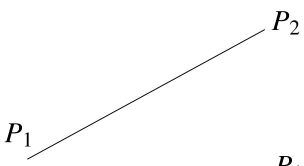


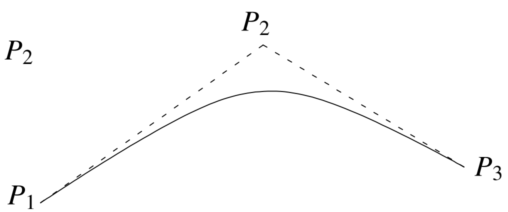
Kurvans tangent har ändrat riktning vid ändpunkterna.

Beziér kurvor

Definition En Beziér kurva (eller yta) bestäms av ett antal *styrpunkter*.

Exempel



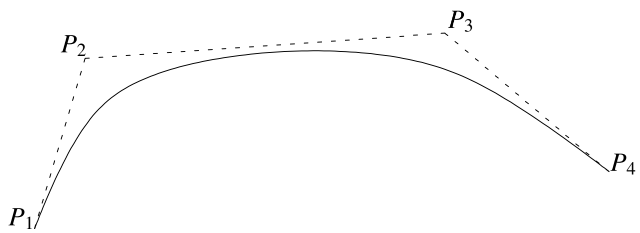


En *linjär* Beziér kurva ges av två punkter. En *kvadratisk* kurva ges av tre punkter.

Beziér kurvor är *approximerande* splines.

Kubiska Beziér kurvor

En *kubisk Beziér kurva* ges av fyra styrpunkter.



Kurvan ges av uttrycket

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

Lemma En *kubisk Beziér kurva* $s(x)$ interpolerar den första och den sista styrpunkten, i.e. $s(0) = P_1$ och $s(1) = P_4$.

Definition Det *konvexa höljet* som ges av punkterna $\{P_i\}_{i=1}^n$ består av alla punkter som kan skrivas som en konvex linjär kombination av samma punkter $\{P_i\}_{i=1}^n$.

Exempel Vad är det konvexa höljet av punkterna

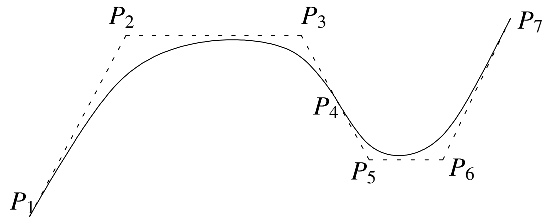
$$P_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad P_2 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \quad \text{and } P_3 = \begin{pmatrix} 1 \\ 3 \end{pmatrix}.$$

Sats En Beziér kurva ligger i det konvexa höljet tillhörande dess styrpunker.

Sats En Beziér kurva, med styrpunkter $\{P_i\}_{i=1}^n$, har en tangent, som ges av

$$s'(0) = \alpha(P_2 - P_1), \text{ och } s'(1) = \alpha(P_n - P_{n-1}), \quad \alpha > 0.$$

Vi kan bestämma kurvans lutning i ändpunkterna.



Två kubiska Beziér kurvor som ges av styrpunkter $\{P_1, P_2, P_3, P_4\}$ och $\{P_4, P_5, P_6, P_7\}$. Interpolations punkten P_4 är *gemensam*.

Villkoret $P_4 - P_3 = P_5 - P_4$ ger en kontinuerlig riktning för tangenten.

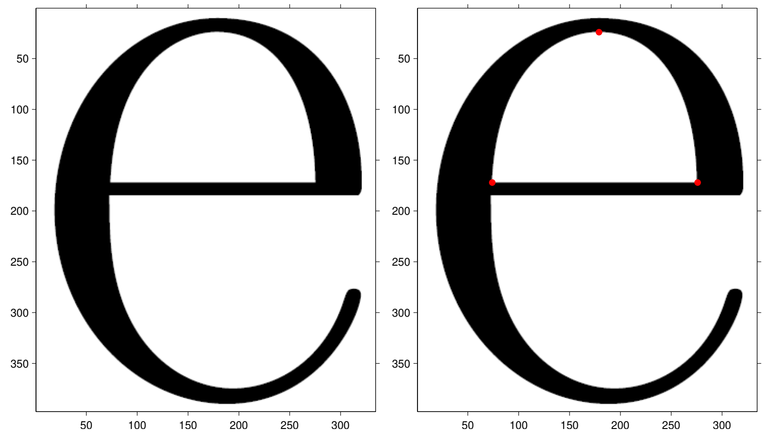
Vektoriserade fonter

Ett dokument kan skrivas ut på olika skrivare och med olika upplösning. Vi behöver kunna generera bitmap bilder som representerar bokstäver i olika storlekar.

Exempel Tre bokstäver från *Computer Modern*

Hur skall bokstävernas form beskrivas matematiskt?

Exempel För att beskriva bokstaven *e* introducerar vi ett antar styrpunkter.



Först väljer vi några *interpolationspunkter* och några extra *styrpunkter*.

Först väljer vi interpolationspunkter och ritar en linje.

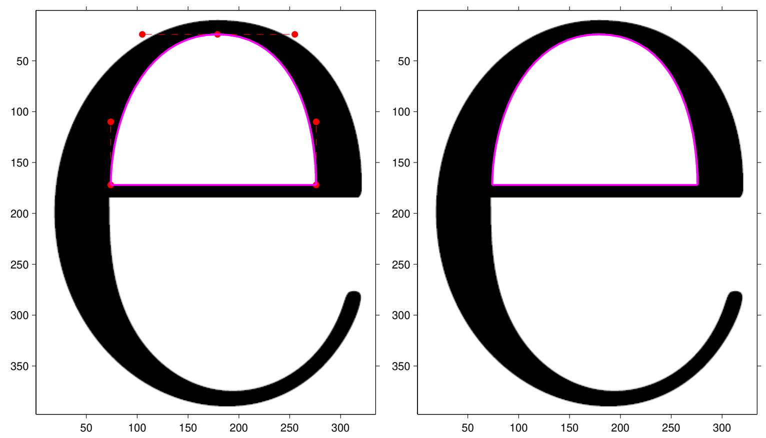
```
>>>P=np.array([ [ 74, 276, 179
                 [172, 172, 24] ])
>>>plot( P(1,1:2),P(1,1:2)
```

Pick a number of control points

```
>>>C = np.array([ [ 74, 105, 255, 276]
                  [110,  24,  24, 110] ])
```

and draw a dashed line from interpolation point P_1 and control point C_1 by

```
>>>pp.plot([P[1,1],C[1,1]],[P[2,1],C[2,1]],’r--’)
>>>pp.show()
```

Den färdiga kurvan med och utan styrpunkterna.

Totalt krävs 2 linjer och 11 Beziér kurvor för att beskriva bokstaven. Fontpaket är rätt små.

Beziérytor

Introducera interpolationspunkter P_1, P_2 och P_3, samt styrpunkter C_1, C_2 och C_3.

En kvadratisk yta ges av

$$s(x,y) = P_1x^2 + P_2y^2 + P_3(1-x-y)^2 + P_4 2xy +$$

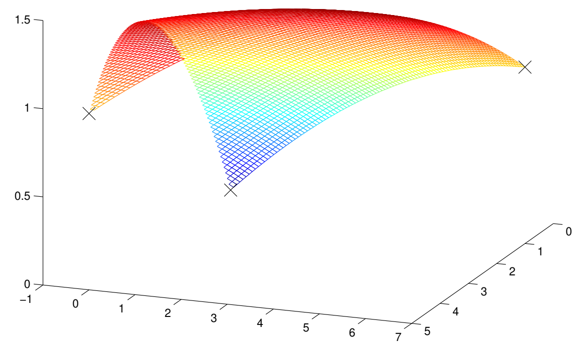
$$P_5 2x(1-x-y) + P_6 2y(1-x-y),$$

där $0 \leq x \leq 1$, $0 < y \leq 1$ och $x + y \leq 1$.

Välj interpolations- och styrpunkter

```
>>>P = np.array([ [ 0, 0, 0, 0, 3, 2]
                  [ 0, 5, 1, 4, 1, 1]
                  [ 7, 1, 1, 5, 4, 2]])
```

Rita ytan!



Styrpunkterna ligger högre i z-riktningen. Detta ger ytans krökning.