

---

Stelsel vergelijkingen

$$\sum_{k=-3}^{n-1} c_k B_k^3(t_i) = f_i := f(t_i)$$

$$\begin{pmatrix} c_{-3}^3 B_{-3}^3(t_0) + c_{-2}^3 B_{-2}^3(t_0) + \dots + c_{n-1} B_{n-1}^3(t_0) \\ \vdots \\ c_{-3}^3 B_{-3}^3(t_n) + c_{-2}^3 B_{-2}^3(t_n) + \dots + c_{n-1} B_{n-1}^3(t_n) \end{pmatrix} = \begin{pmatrix} f(t_0) \\ \vdots \\ f(t_n) \end{pmatrix}$$

$$\begin{pmatrix} B_{-3}^3(t_0) & B_{-2}^3(t_0) & \dots & B_{n-1}^3(t_0) \\ \vdots & \vdots & \ddots & \vdots \\ B_{-3}^3(t_n) & B_{-2}^3(t_n) & \dots & B_{n-1}^3(t_n) \end{pmatrix} \begin{pmatrix} c_{-3} \\ \vdots \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} f(t_0) \\ \vdots \\ f(t_n) \end{pmatrix}$$

Deze heeft  $n + 1$  vergelijkingen en  $n + 3$  onbekenden. Er moeten dus nog twee vergelijkingen toegevoegd worden, namelijk

$$\left( \sum_{k=-3}^{n-1} c_k B_k^3(t_0) \right)'' = 0$$

en

$$\left( \sum_{k=-3}^{n-1} c_k B_k^3(t_n) \right)'' = 0$$

---

De tweede afgeleiden in  $t_0$  en  $t_n$

Ik bereken de  $2^e$  afgeleide zonder gebruik te maken van de equidistantie omdat deze later misschien nog nuttig is.

$$\begin{aligned} \left( \sum_{k=-3}^{n-1} c_k B_k^3(x) \right)'' &= \sum_{k=-3}^{n-1} c_k B_k^3(x)'' \\ &= \sum_{k=-3}^{n-1} \frac{3c_k B_k^2(x)'}{t_{k+3} - t_k} - \frac{3c_k B_{k+1}^2(x)'}{t_{k+4} - t_{k+1}} \\ &= \sum_{k=-3}^{n-1} \frac{6c_k}{t_{k+3} - t_k} \left( \frac{B_k^1(x)}{t_{k+2} - t_k} - \frac{B_{k+1}^1(x)}{t_{k+3} - t_{k+1}} \right) \\ &\quad - \frac{6c_k}{t_{k+4} - t_{k+1}} \left( \frac{B_{k+1}^1(x)}{t_{k+3} - t_{k+1}} - \frac{B_{k+2}^1(x)}{t_{k+4} - t_{k+2}} \right) \end{aligned}$$

Als we weten dat  $t_k = ih$  komen we tot volgende constatactie

$$\begin{aligned}
\left( \sum_{k=-3}^{n-1} c_k B_k^3(x) \right)'' &= \sum_{k=-3}^{n-1} \frac{6c_k}{3h} \left( \frac{B_k^1(x)}{2h} - \frac{B_{k+1}^1(x)}{2h} \right) - \frac{6c_k}{3h} \left( \frac{B_{k+1}^1(x)}{2h} - \frac{B_{k+2}^1(x)}{2h} \right) \\
&= \sum_{k=-3}^{n-1} \frac{c_k (B_k^1(x) - B_{k+1}^1(x))}{h^2} - \frac{c_k (B_{k+1}^1(x) - B_{k+2}^1(x))}{h^2} \\
&= \sum_{k=-3}^{n-1} \frac{c_k (B_k^1(x) - 2B_{k+1}^1(x) + B_{k+2}^1(x))}{h^2}
\end{aligned}$$

De term  $\frac{B_k^1(x) - 2B_{k+1}^1(x) + B_{k+2}^1(x)}{h^2}$  zal voortaan  $D_k(x)$  genoemd worden.

### Aangepast stelsel

Het stelsel met de twee vergelijkingen toegevoegd wordt dan

$$\begin{pmatrix} D_{-3}(t_0) & D_{-2}(t_0) & \dots & D_{n-1}(t_0) \\ B_{-3}^3(t_0) & B_{-2}^3(t_0) & \dots & B_{n-1}^3(t_0) \\ B_{-3}^3(t_1) & B_{-2}^3(t_1) & \dots & B_{n-1}^3(t_1) \\ \vdots & \vdots & \ddots & \vdots \\ B_{-3}^3(t_n) & B_{-2}^3(t_n) & \dots & B_{n-1}^3(t_n) \\ D_{-3}(t_n) & D_{-2}(t_n) & \dots & D_{n-1}(t_n) \end{pmatrix} \begin{pmatrix} c_{-3} \\ c_{-2} \\ c_{-1} \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix} = \begin{pmatrix} 0 \\ f(t_0) \\ f(t_1) \\ \vdots \\ f(t_n) \\ 0 \end{pmatrix}$$

Nu moet bewezen worden dat dit stelsel tridiagonaal is. Daartoe wordt eerst onderzocht wanneer  $B_k^3 = 0$

### $B_k^3 = 0$ ?

Hier wordt recursief gewerkt.

$$\begin{aligned}
B_k^0(t_i) = 0 &\Leftrightarrow t_i \geq t_{k+1} \vee t_i < t_k \\
&\Leftrightarrow i \neq k
\end{aligned}$$

$$\begin{aligned}
B_k^1(t_i) = 0 &\Leftrightarrow B_k^0(t_i) = 0 \wedge B_{k+1}^0(t_i) = 0 \\
&\Leftrightarrow i \neq k \wedge i \neq k + 1
\end{aligned}$$

Een uitzondering hier is dat als  $i = k$ , het resultaat ook 0 wordt.

$$\begin{aligned}
B_k^1(t_k) &= \frac{t_k - t_k}{h} B_k^0(t_k) + \frac{t_{k+2} - t_k}{h} B_{k+1}^0(t_k) \\
&= 0 + \frac{t_{k+2} - t_k}{h} B_{k+1}^0(t_k)
\end{aligned}$$

$$\begin{aligned}
&\text{Omdat } k + 1 \neq k \text{ is ook de tweede term } 0 \\
&= 0
\end{aligned}$$

Dus, even samenvatten

$$\begin{aligned}
B_k^0(t_i) &= 0 \quad \text{als } i \neq k \\
B_k^1(t_i) &= 0 \quad \text{als } i \neq k + 1
\end{aligned}$$



$$\begin{aligned}
B_k^3(t_{k+1}) &= \frac{B_k^2(t_{k+1}) + B_{k+1}^2(t_{k+1})}{3} \\
&= \frac{B_k^2(t_{k+1})}{3} \\
&= \frac{B_k^1(t_{k+1}) + 2B_{k+1}^1(t_{k+1})}{6} \\
&= \frac{B_k^1(t_{k+1})}{6} \\
&= \frac{1}{6}
\end{aligned}$$

$$\begin{aligned}
B_k^3(t_{k+2}) &= \frac{2}{3} (B_k^2(t_{k+2}) + B_{k+1}^2(t_{k+2})) \\
&= \frac{2}{2 \cdot 3} (B_k^1(t_{k+2}) + 2B_{k+1}^1(t_{k+2}) + B_{k+2}^1(t_{k+2})) \\
&= \frac{4}{6} B_{k+1}^1(t_{k+2}) \\
&= \frac{4}{6}
\end{aligned}$$

$$\begin{aligned}
D_k(t_{k+3}) &= \frac{B_k^1(t_{k+3}) - 2B_{k+1}^1(t_{k+3}) + B_{k+2}^1(t_{k+3})}{h^2} \\
&= \frac{1}{h^2}
\end{aligned}$$

$$\begin{aligned}
D_k(t_{k+2}) &= \frac{B_k^1(t_{k+2}) - 2B_{k+1}^1(t_{k+2}) + B_{k+2}^1(t_{k+2})}{h^2} \\
&= \frac{-2}{h^2}
\end{aligned}$$

$$\begin{aligned}
D_k(t_{k+1}) &= \frac{B_k^1(t_{k+1}) - 2B_{k+1}^1(t_{k+1}) + B_{k+2}^1(t_{k+1})}{h^2} \\
&= \frac{1}{h^2}
\end{aligned}$$

$$A := \begin{pmatrix} \frac{1}{h^2} & \frac{-2}{h^2} & \frac{1}{h^2} & 0 & & & \\ \frac{1}{6} & \frac{4}{6} & \frac{1}{6} & 0 & & & \\ 0 & \frac{1}{6} & \frac{4}{6} & \frac{1}{6} & & & \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \frac{1}{6} & \frac{4}{6} & \frac{1}{6} & 0 & \\ & & 0 & \frac{1}{6} & \frac{4}{6} & \frac{1}{6} & \\ & & 0 & \frac{1}{h^2} & \frac{-2}{h^2} & \frac{1}{h^2} & \end{pmatrix}$$

Bewijs dat deze matrix inverseerbaar is

In onderstaand bewijs worden enkel elementaire rijoperaties gebruikt. Als dan de uiteindelijke matrix een determinant heeft  $\neq 0$  dan is ook de eerste matrix ( $A$ ) inverseerbaar.

Eerst wordt de factoren  $h^2$  en 6 weggewerkt door de eerste rij en de laatste rij te vermenigvuldigen met  $h^2$  en al de andere te vermenigvuldigen met 6. Dit is toegestaan omdat  $h \neq 0$ .

$$A^0 := \begin{pmatrix} 1 & -2 & 1 & 0 & & & \\ 1 & 4 & 1 & 0 & & & \\ 0 & 1 & 4 & 1 & & & \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 & 0 & \\ & & 0 & 1 & 4 & 1 & \\ & & 0 & 1 & -2 & 1 & \end{pmatrix}$$

Definieer nu  $\forall i \in \{1 \dots n + 1\}$

$$A_j^i := \begin{cases} A_j^{i-1} & \text{Als } j \neq i \\ \frac{A_i^{i-1} - A_{i-1}^{i-1}}{4 - A_{i-1,i}^{i-1}} & \text{Als } j = i \end{cases}$$

Nu wordt een inductie gepleegd over n met volgende eigenschap die wordt meegeseleurd.

$$\forall i \in \{1 \dots n + 1\} \quad \begin{cases} 0 \leq A_{i,i+1}^i \leq 1 & (1) \\ A_{i,i}^i = 1 & (2) \\ A_{i,j}^i = 0 \quad \forall j < i & (3) \\ A_{i,i+2+j}^i = 0 \quad \forall j \geq 0 & (4) \end{cases}$$

De inductiebasis:

$$\begin{aligned} A_{1,0}^1 &= \frac{A_{1,0}^0 - A_{0,0}^0}{4 - A_{0,1}^0} \\ &= \frac{1 - 1}{4 - (-2)} \\ &= 0 \end{aligned} \tag{3}$$

$$\begin{aligned} A_{1,2}^1 &= \frac{A_{1,2}^0 - A_{0,2}^0}{4 - A_{0,2}^0} \\ &= \frac{1 - 1}{4 - (-2)} \\ &= 0 \end{aligned} \tag{1}$$

$$\begin{aligned} A_{1,1}^1 &= \frac{A_{1,1}^0 - A_{0,1}^0}{4 - A_{0,1}^0} \\ &= \frac{4 - (-2)}{4 - (-2)} \\ &= 1 \end{aligned} \tag{2}$$

$$\begin{aligned} A_{1,3+j}^1 &= \frac{A_{1,3+j}^0 - A_{0,3+j}^0}{4 - A_{0,3+j}^0} \\ &= \frac{0 - 0}{4 - 0} \\ &= 0 \end{aligned} \tag{4}$$

De inductiestap:

Bewijs van (1):

$$\begin{aligned}
 A_{i,i+1}^i &= \frac{A_{i,i+1}^{i-1} - A_{i-1,i+1}^{i-1}}{4 - A_{i-1,i}^{i-1}} \\
 &\geq \frac{A_{i,i+1}^{i-1} - A_{i-1,i+1}^{i-1}}{4} \\
 &\geq \frac{1}{4}
 \end{aligned}$$

$$\begin{aligned}
 A_{i,i+1}^i &= \frac{A_{i,i+1}^{i-1} - A_{i-1,i+1}^{i-1}}{4 - A_{i-1,i}^{i-1}} \\
 &\leq \frac{A_{i,i+1}^{i-1} - A_{i-1,i+1}^{i-1}}{4 - 1} \\
 &\leq \frac{1}{3}
 \end{aligned}$$

De teller in bovenstaande mag gelijkgesteld worden aan 1 omdat  $A_{i,i+2}^{i-1} = 0$  (inductiehypothese) en  $A_{i,i+1}^{i-1} = A_{i,i+1}^{i-2} = \dots = A_{i,i+1}^0 = 1$

Bewijs van (2):

$$\begin{aligned}
 A_{i,i}^i &= \frac{A_{i,i}^{i-1} - A_{i-1,i}^{i-1}}{4 - A_{i-1,i}^{i-1}} \\
 &= \frac{4 - A_{i-1,i}^{i-1}}{4 - A_{i-1,i}^{i-1}} \\
 &= 1
 \end{aligned}$$

De noemer in bovenstaande is  $\neq 0$  omdat  $0 \leq A_{i-1,i}^{i-1} \leq 1$ . Verder is  $A_{i,i}^{i-1} = A_{i,i}^{i-2} = \dots = A_{i,i}^0 = 4$ .

Bewijs van (3):

Als  $j < i - 1$  dan hebt u dat

$$\begin{aligned}
 A_{i,j}^i &= \frac{A_{i,j}^{i-1} - A_{i-1,j}^{i-1}}{4 - A_{i-1,i}^{i-1}} \\
 &\quad \text{inductiehypothese voor } A_{i-1,j}^{i-1} \\
 &= \frac{A_{i,j}^{i-1}}{4 - A_{i-1,i}^{i-1}} \\
 &\quad \text{omdat } A_{i,j}^{i-1} = A_{i,j}^{i-2} = \dots = A_{i,j}^0 = 0 \\
 &= 0
 \end{aligned}$$

Als  $j = i - 1$

$$\begin{aligned}
A_{i,i-1}^i &= \frac{A_{i,i-1}^{i-1} - A_{i-1,i-1}^{i-1}}{4 - A_{i-1,i}^{i-1}} \\
&\text{inductiehypothese voor } A_{i-1,i-1}^{i-1} \\
&= \frac{A_{i,i-1}^{i-1} - 1}{4 - A_{i-1,i}^{i-1}} \\
&\text{omdat } A_{i,i-1}^{i-1} = A_{i,i-1}^{i-2} = \dots = A_{i,i-1}^0 = 1 \\
&= 0
\end{aligned}$$

Dan, als laatste het bewijs van (4)

$$\begin{aligned}
A_{i,i+2+j}^i &= \frac{A_{i,i+2+j}^{i-1} - A_{i-1,i+2+j}^{i-1}}{4 - A_{i-1,i}^{i-1}} \\
&\text{omdat } A_{i,i+2+j}^{i-1} = A_{i,i+2+j}^{i-2} = \dots = A_{i,i+2+j}^0 = 0 \\
&= \frac{-A_{i-1,i+2+j}^{i-1}}{4 - A_{i-1,i}^{i-1}} \\
&\text{Inductiehypothese} \\
&= 0
\end{aligned}$$

Uiteindelijk moet de laatste rij nog bewerkt worden zodat  $A_n$  en  $A_{n+1}$  0 worden.

$$\begin{aligned}
A_{n+2}^{n+2} &:= \frac{A_{n+2}^{n+1} - A_n^{n+1} + (2 + A_{n,n+1}^{n+1})A_{n+1}^{n+1}}{1 + (2 + A_{n,n+1}^{n+1})A_{n+1,n+2}^{n+1}} \\
&= \frac{A_{n+2}^0 - A_n^{n+1} + (2 + A_{n,n+1}^{n+1})A_{n+1}^{n+1}}{1 + (2 + A_{n,n+1}^{n+1})A_{n+1,n+2}^{n+1}} \\
A_{n+2,n+2}^{n+2} &= \frac{A_{n+2,n+2}^0 - A_{n,n+2}^{n+1} + (2 + A_{n,n+1}^{n+1})A_{n+1,n+2}^{n+1}}{1 + (2 + A_{n,n+1}^{n+1})A_{n+1,n+2}^{n+1}} \\
&= \frac{1 - 0 + (2 + A_{n,n+1}^{n+1})A_{n+1,n+2}^{n+1}}{1 + (2 + A_{n,n+1}^{n+1})A_{n+1,n+2}^{n+1}} \\
&= 1 \\
A_{n+2,n+1}^{n+2} &= \frac{A_{n+2,n+1}^0 - A_{n,n+1}^{n+1} + (2 + A_{n,n+1}^{n+1})A_{n+1,n+1}^{n+1}}{1 + (2 + A_{n,n+1}^{n+1})A_{n+1,n+2}^{n+1}} \\
&= \frac{-2 - A_{n,n+1}^{n+1} + 2 + A_{n,n+1}^{n+1}}{1 + (2 + A_{n,n+1}^{n+1})A_{n+1,n+2}^{n+1}} \\
&= 0 \\
A_{n+2,n}^{n+2} &= \frac{A_{n+2,n}^0 - A_{n,n}^{n+1} + (2 + A_{n,n+1}^{n+1})A_{n+1,n}^{n+1}}{1 + (2 + A_{n,n+1}^{n+1})A_{n+1,n+2}^{n+1}} \\
&= \frac{1 - 1 + (2 + A_{n,n+1}^{n+1})0}{1 + (2 + A_{n,n+1}^{n+1})A_{n+1,n+2}^{n+1}} \\
&= 0
\end{aligned}$$

Voila, zo is bewezen dat het stelsel een eenduidige oplossing heeft. De matrix  $A^{n+2}$  is een bovendriehoeksmatrix met op de diagonaal allemaal 1-tjes. Dit wil zeggen dat de determinant niet 0 is, en dat de matrix dus inverseerbaar is.

---

Afgeleiden van de gegeven functies

$$\begin{aligned} f_a(x) &= \cos(2\pi x) \\ f_a(x)' &= -2\pi \sin(2\pi x) \\ f_a(x)'' &= -4\pi^2 \cos(2\pi x) \\ f_a(x)''' &= 8\pi^3 \sin(2\pi x) \end{aligned}$$

$$\begin{aligned} f_b(x) &= \left| x - \frac{1}{2} \right| \sin(2\pi x) \\ f_b(x)' &= \begin{cases} \pi(2x-1)\cos(2\pi x) + \sin(2\pi x) & \text{if } x \geq \frac{1}{2} \\ \pi(1-2x)\cos(2\pi x) - \sin(2\pi x) & \text{otherwise} \end{cases} \\ f_b(x)'' &= \begin{cases} 2\pi^2(1-2x)\sin(2\pi x) + 4\pi\cos(2\pi x) & \text{if } x \geq \frac{1}{2} \\ 2\pi^2(2x-1)\sin(2\pi x) - 4\pi\cos(2\pi x) & \text{otherwise} \end{cases} \\ f_b(x)''' &= \begin{cases} 4\pi^3(1-2x)\cos(2\pi x) - 12\pi^2\sin(2\pi x) & \text{if } x \geq \frac{1}{2} \\ 4\pi^3(2x-1)\cos(2\pi x) + 12\pi^2\sin(2\pi x) & \text{otherwise} \end{cases} \end{aligned}$$

$$\begin{aligned} f_c(x) &= \left(x - \frac{1}{2}\right) \sin(\pi x) \\ f_c(x)' &= \pi\left(x - \frac{1}{2}\right) \cos(\pi x) + \sin(\pi x) \\ f_c(x)'' &= \pi^2\left(\frac{1}{2} - x\right) \sin(\pi x) + 2\pi \cos(\pi x) \\ f_c(x)''' &= \pi^3\left(\frac{1}{2} - x\right) \cos(\pi x) - 3\pi^2 \sin(\pi x) \end{aligned}$$

---

Bespreking van de meetresultaten

Op volgende bladzijden vindt u de meetresultaten. Helemaal op het einde zijn de grafieken ondergebracht.

*Part 1* is de berekening waarbij als sluitingsvergelijkingen gesteld werden dat de tweede afgeleiden in de randpunten 0 moeten zijn. Dat dit inderdaad zo is kan afgelezen worden in de meetresultaten. Zo is de fout op de afgeleide van functie a 39.5, wat normaal is aangezien de B-spline 0 moet verkondigen en de juiste afgeleide van de functie 39.5 is.

*Part 2* is de berekening waarbij aan de randpunten op het halve interval de B-spline ook overeen moet komen met de functie. Het ligt dan natuurlijk voor de hand dat de fouten voor 'Part 2' kleiner zijn dan voor 'Part 1' als men met weinig punten werkt. Dit kan zeer duidelijk afgelezen worden op grafieken van functie a.

Voor alle grafieken geldt verder dat men grote fouten heeft als men met weinig steunpunten werkt.

De twee eerste punten in de grafieken worden dus buiten beschouwing gelaten om een grootte-orde te schatten.

Naar het midden toe benaderen de B-splines met vrije rand de functie in  $O(h^4)$  (dit zie je omdat de punten evenwijdig lopen met de  $h^4$  rechte). Aan de rand daarentegen geraakt men niet beter dan  $O(h^2)$ . In het midden van het interval is het effect van de randvoorwaarde eigenlijk niet meer voelbaar. Vandaar ook dat voor functies a en c de maximumfout in  $[\frac{1}{4}, \frac{3}{4}]$  voor de Part1-benadering ongeveer gelijk is aan de Part2-benadering.

Als de fout over het ganse interval bekeken wordt is duidelijk dat de fout nog maar  $O(h^2)$  is. [Zie grafieken]

Aangezien de maximum-afgeleide een coëfficiënt is bij de  $h^4$  in de afschatting van de fout is het ook vanzelfsprekend dat functie c beter benadert wordt dan functie a.

Functie b is een speciaal geval omdat de afgeleide hier een discontinuïteit vertoont in  $\frac{1}{2}$ . Veel van de maximumfouten waren dan ook in deze buurt te vinden.

Indien de randpunten gespeend waren van een multipliciteit die de 4 zou zijn (geen 3, geen 2, en 5 is volledig uit den boze) dan zou het stelsel niet oplosbaar zijn: Door de multipliciteit te verhogen verlaagt de afleidbaarheid. In dit geval zelfs zodanig dat de  $2^e$  afgeleide in  $t_n$  altijd 0 zou zijn. Dit wil zeggen dat de rang van de matrix  $n + 2$  zou zijn, i.p.v.  $n + 3$  en dat er dus geen unieke oplossing meer is. Hieronder staat het bewijs dat  $B_{n-3}^{3(2)}(t_n) = 0$ .  $B_{n-2}^{3(2)}(t_n)$  en  $B_{n-1}^{3(2)}(t_n)$  zijn analoog.

$$\begin{aligned}
 B_{n-3}^{3(2)}(t_n) &= B_{n-3}^{2(1)}(t_n) + B_{n-2}^{2(1)}(t_n) \\
 &= B_{n-3}^{1(0)}(t_n) + B_{n-2}^{1(0)}(t_n) + B_{n-2}^{1(0)}(t_n) + B_{n-1}^{1(0)}(t_n) \\
 &= B_{n-1}^{1(0)}(t_n) \\
 &= B_{n-1}^0(t_n) + B_n^0(t_n) \\
 &= \begin{cases} 1 & \text{als } t_n \leq t_n < t_n + 1 \\ 0 & \text{anders} \end{cases} \\
 &= 0
 \end{aligned}$$

De spline-interpolant door de gegeven punten vind u op de allerlaatste bladzijde. In vergelijking met de  $12^e$  graadsinterpolatiepolynoom swingt deze functie veel minder op en neer, vooral tussen de randpunten. Dit kan verklaart worden door te zeggen dat de graad van de stukgewijze veelterm tussen twee punten ten hoogste 3 is.

### Het CalcB algoritme

CalcB(int p, xfloat x, xfloat rooster[], int d)

berekent  $B^{p(d)}(rooster[0], \dots, rooster[p + 1]; x)$ . Waarbij de  $(d)$  staat voor de ' $d^e$  afgeleide naar  $x$ '. Het algoritme zelf:

```

xfloat CalcB(int p, xfloat x, xfloat rooster[], int d)
{
  xfloat answer;
  if (!p)
  {
    if (d) return 0;

```

Als er nog een afgeleide genomen moet worden is het duidelijk dat het resultaat 0 moet

```

        zijn.
    if (xgeq(x,rooster[0]) && x1(x,rooster[1])) return 1;
        De controle dat  $x \geq rooster[0]$  is een beetje
        een vreemd geval. Er wordt gecontroleerd of
         $x \geq rooster[0] - \eta$ . † Ook zo om te con-
        troleren of  $x < rooster[1]$ , deze wordt dan
         $x < rooster[1] - \eta$ .

    return 0;
}
if (d)
{
    answer=
    (p*CalcB(p-1,x,rooster,d-1))/(rooster[p]-rooster[0])
    -(p*CalcB(p-1,x,rooster+1,d-1))/(rooster[p+1]-rooster[1]);
        Het berekenen van de afgeleide met behulp
        van de formule op pagina 40 (4.4)
    if (xz(answer)) return 0.;
        xz controleert of answer voldoende klein is
        om gelijk gesteld te kunnen worden aan 0.
        Dit komt er op neer dat gekeken wordt of
         $|answer| < \eta$ . Als deze regel niet toegevoegd
        is krijgt men geen tridiagonale matrix.

    return answer;
}
answer=
((x-rooster[0])*CalcB(p-1,x,rooster,d))/(rooster[p]-rooster[0])
+ ((rooster[p+1]-x)*CalcB(p-1,x,rooster+1,d))/(rooster[p+1]-rooster[1]);
        Het berekenen van de B-spline met behulp
        van de formule op pagina 39, (4.2)
if (xz(answer)) return 0.;
return answer;
}

```

---

#### Source

```

/* #define GRAPHICS_PLEASE*/
#include <vga.h>
#include <math.h>
#include <stdlib.h> /* random */
#include <assert.h>
typedef double xfloat;
xfloat pi,pi2,pi3,tweepi;
const xfloat xzero= 1.0e-17; /* machineprecisie bij 53 bits */
/* we gaan wijlen hier ne keer een grafieksken neerpoten */
xfloat lx, rx, ty, by;
void Graphics(xfloat x1,xfloat x2, xfloat y1,xfloat y2)
{
    lx=x1;
    rx=x2;
    ty=y1;
    by=y2;
}

```

---

†  $\eta$  is de machineprecisie

```

    }

void Assekruis(int color)
{
    int x,y;
    vga_setcolor(color);
    x=Recalcx(0.);
    y=Recalcy(0.);
    vga_drawline(x,0,x,479);
    vga_drawline(0,y,639,y);
}

int Recalcx(xfloat x)
{
    return floor(((x-lx)*640.)/(rx-lx));
}

int Recalcy(xfloat y)
{
    return floor((((y-by)*480.)/(ty-by)));
}

void DrawDot(xfloat x, xfloat y, int color)
{
    int b;
    vga_setcolor(color);
    b=Recalcy(y);
    if (b>479 || b<0) return;
    vga_drawpixel(Recalcx(x),b);
}

void StartGraphics(void)
{
    vga_init();
    vga_setmode(G640x480x16);
}

int xgez(xfloat a)
{return (a>-xzero);}

int xlz(xfloat a)
{return (a<-xzero);}

int xz(xfloat a)
{return ((a<=0 && a>-xzero) || (a>=0 && a<xzero));}

int xeq(xfloat a, xfloat b)
{return xz(a-b);}

int xgeq(xfloat a, xfloat b)
{return xgez(a-b);}

int xl(xfloat a, xfloat b)

```

```

    {return xlz(a-b);}

void printxfloat(xfloat watte)
    {printf("%8.3g",watte);}

xfloat CalcB(int p, xfloat x, xfloat rooster[], int d)
/* p is de graad van de B-spline */
/* d is hoeveel maal de functie afgeleid dient te worden */
/* de nodige roosterpunten worden in rooster meegegeven */
    {
        xfloat answer;
        xfloat temp;
        assert(p>=0);
        assert(d>=0);
#ifdef DEBUG_CALC_B
        printf("CalcB(%d, %g)",p,x);
        {
            int t;
            for(t=0;t<=p+1;t++)
                printf("%8.3g ",rooster[t]);
            printf("\n");
        }
        /* eventuele controle op dubbele punten */
#endif
        if (!p)
            {if (d) return 0;
             if (xgeq(x,rooster[0]) && xl(x,rooster[1])) return 1;
             return 0;}
        if (d)
            {
                answer=
                (p*CalcB(p-1,x,rooster,d-1))/(rooster[p]-rooster[0])
                -(p*CalcB(p-1,x,rooster+1,d-1))/(rooster[p+1]-rooster[1]);
                if (xz(answer)) return 0.;
                return answer;
            }
        answer=
        ((x-rooster[0])*CalcB(p-1,x,rooster,d))/(rooster[p]-rooster[0])
        + ((rooster[p+1]-x)*CalcB(p-1,x,rooster+1,d))/(rooster[p+1]-rooster[1]);
        if (xz(answer)) return 0.;
        return answer;
    }

void PrintMatrix(int nx, int ny,xfloat **m)
    {
        int x,y;
        for(y=0;y<ny;y++)
            {
                for(x=0;x<nx;x++)
                    printf("%8.3g ",m[x][y]);
                printf("\n");
            }
    }

```

```

        printf("-----\n");
    }

void SolveStelsel(xfloat **m,xfloat *c, int n)
/* de matrix is een array van kolommen */
/* n is de laatste rij, dus */
/* n+1 is het aantal rijen in de matrix */
/* n+2 is het aantal kolommen in de matrix (de laatste kolom zijn de functie-
 * resultaten) */
/* BEWARE !!!!, m wordt veranderd door deze operatie, net zoals c*/
{
    int x,a,y;
    xfloat tozero;
    assert(m);
    assert(c);
    for(x=0;x<=n;x++)
    {
        xfloat*k=m[x];
        assert(k);
        /* Probeer een 1 op de diagonaal te krijgen, in deze kolom */
        for(y=0;y<=n;y++)
        {
            /* hier moet inderdaad exact op 0 getest kunnen worden */
            if (x==y) /* het diagonale element */
            {
                assert(m[x][y]);
                continue; /* wordt op de moment nog overgeslagen */
            }
            /*het niet-diagonaal element is reeds 0 */
            if ((tozero=m[x][y])==0.) continue;
            /* rij=rij-tozero*diagrij/diagelement*/
            assert(m[x][x]);
            for(a=0;a<=n+1;a++)
                m[a][y]-=tozero*m[a][x]/m[x][x];
            /* normaal zou het element nu op 0 moeten staan*/
            /* maar ik zet het nog eens expliciet op 0 om afrondfouten
             * de kop in te dreunen */
            m[x][y]=0;
        }
        /* PrintMatrix(n+2,n+1,m);*/
    }
    /* dan kan ik nu de diagonaal op 1 brengen door een delingsken*/
    for(y=0;y<=n;y++)
        c[y]=m[n+1][y]/m[y][y];
}

xfloat xrandom()
{
    return (float)random()/(float)RAND_MAX;
}

xfloat *CreateRooster(int n,int p, xfloat *h)
{

```

```

xfloat old=0;
xfloat *rooster;
int t=0;
rooster=(xfloat*)malloc(sizeof(xfloat)*(n+1+p*2));
assert(rooster);
rooster[0]=0;
for(t=1;t<=n;t++) rooster[t+p]=old+=xrandom()+0.5;
for(t=1;t<=n;t++) rooster[t+p]/=old;
for(t=-p;t<0;t++) rooster[t+p]=(float)t/(float)n;
for(t=n+1;t<=n+p;t++) rooster[t+p]=((float)(t-n)/(float)n)+1.;
for(t=0;t<n;t++)
    if (rooster[t+p+1]-rooster[t+p]>*h)
        *h=rooster[t+p+1]-rooster[t+p];
/* for(t=-p;t<=n+p;t++) printf("%8.3g\n",rooster[t+p]); */
return rooster;
}

xfloat**CreateMatrix(int n, int p, xfloat*rooster)
{
    xfloat **m;
    int x,y;
    /* alloceren van de matrix */
    m=(xfloat**)malloc((n+p+1)*sizeof(xfloat*));
    assert(m);
    for(x=0;x<=n+p;x++)
    {
        m[x]=(xfloat*)malloc((n+p)*sizeof(xfloat));
        assert(m[x]);
        for(y=0;y<n+p;y++) m[x][y]=0;
    }
    /* de juiste data er in steken */
    for(y=1;y<=n+1;y++)
        for (x=0;x<n+p;x++)
            m[x][y]=CalcB(p,rooster[p+y-1],rooster+x,0);
    /* merk op dat de eerste rij, de laatste rij en de
     * laatste kolom nog niet ingevuld zijn */
    return(m);
}

typedef xfloat eenfunctie(xfloat a, int d);

xfloat **CreatePart1Matrix(int n, int p, xfloat *rooster, eenfunctie f)
{
    xfloat**m;
    int x;
    m=CreateMatrix(n,p,rooster);
    for(x=0;x<n+p;x++) m[x][0]=CalcB(p,rooster[p],rooster+x,2);
    for(x=0;x<n+p;x++) m[x][n+p-1]=CalcB(p,rooster[n+p],rooster+x,2);
    return m;
}

xfloat **CreatePart2Matrix(int n, int p, xfloat *rooster, eenfunctie f)
{

```

```

        xfloat**m;
        int x;
        m=CreateMatrix(n,p,rooster);
        for(x=0;x<n+p;x++) m[x][0]=CalcB(p,rooster[p+1]/2,rooster+x,2);
        for(x=0;x<n+p;x++)
            m[x][n+p-1]=CalcB(p,1-(rooster[n+p]-rooster[n+p-1])/2,rooster+x,2);
        m[n+p][0]=f(rooster[p+1]/2,0);
        m[n+p][n+p-1]=f(1-(rooster[n+p]-rooster[n+p-1])/2,0);
        return m;
    }

xfloat * CreateCoefficienten(int n, int p)
{
    xfloat *c;
    c=(xfloat*)malloc(sizeof(xfloat)*(n+p));
    assert(c);
    return c;
}

void PrintCoefficienten(int aantal, xfloat *c)
{
    int x=0;
    for(;x<aantal;printf("%8.3g ",c[x++]));
    printf("\n-----\n");
}

void DeleteMatrix(xfloat **m, int nx)
{
    while(nx) free(m[--nx]);
    free(m);
}

xfloat LinComBsplines(int n, int p, xfloat value, xfloat *coef,
                      int d, xfloat *rooster)
{
    int x;
    xfloat a;
    a=0;
    for(x=0;x<n+p;x++) a+=coef[x]*CalcB(p,value,rooster+x,d);
    return a;
}

xfloat functiona(xfloat x, int d)
{
    switch(d)
    {
        case 0: return cos((double)x*tweepi);
        case 1: return -tweepi*sin((double)x*tweepi);
        case 2: return -4*pi2*cos((double)x*tweepi);
        case 3: return 8*pi3*sin((double)x*tweepi);
    }
    assert(0);
}

```

```

xfloat functionb(xfloat x, int d)
{
    if (x>.5)
        switch(d)
        {
            case 0: return (x-.5)*sin(x*tweepi);
            case 1: return tweepi*(x-.5)*cos(tweepi*x)+sin(tweepi*x);
            case 2: return 4.*pi*cos(tweepi*x)+pi2*(2.-4.*x)*sin(tweepi*x);
            case 3: return pi3*(4.-8.*x)*cos(tweepi*x)-pi2*12.*sin(pi*x);
        }
    else
        switch(d)
        {
            case 0: return (.5-x)*sin(x*tweepi);
            case 1: return tweepi*(.5-x)*cos(tweepi*x)-sin(tweepi*x);
            case 2: return pi2*(4.*x-2.)*sin(tweepi*x)-4.*pi*cos(tweepi*x);
            case 3: return pi3*(8.*x-4.)*cos(tweepi*x)+pi2*12.*sin(pi*x);
        }
    assert(0);
}

xfloat functionc(xfloat x, int d)
{
    switch(d)
    {
        case 0: return (x-.5)*sin(x*pi);
        case 1: return pi*(x-.5)*cos(x*pi)+sin(pi*x);
        case 2: return tweepi*cos(pi*x)+pi2*(.5-x)*sin(x*pi);
        case 3: return pi3*(.5-x)*cos(x*pi)-3*pi2*sin(pi*x);
    }
    assert(0);
}

void AddFunctionResults(xfloat**m,xfloat*rooster, int n, int p, eenfunctie f)
{
    int x=n+1;
    while(x--) m[n+p][x+1]=f(rooster[x+p],0);
}

typedef xfloat **matrixcreator(int n, int p, xfloat rooster[], eenfunctie f);
void Show(int n, eenfunctie f, matrixcreator CreatePartMatrix)
{
    int p=3;
    int i;
    xfloat *rooster;
    xfloat **m;
    xfloat *c;
    xfloat h=0;
    xfloat maxI[4]={0,0,0,0}; /* het grote interval */
    xfloat maxi[4]={0,0,0,0}; /* het kleinste interval */
    rooster=CreateRooster(n,p,&h);
    m=CreatePartMatrix(n,p,rooster,f);
}

```

```

        AddFunctionResults(m,rooster,n,p,f);
        c=CreateCoefficienten(n,p);
/*      PrintMatrix(n+p+1,n+p,m);*/
        SolveStelsel(m,c,n+p-1);
/*      PrintMatrix(n+p+1,n+p,m);*/
/*      PrintCoefficienten(n+p,c);*/
        DeleteMatrix(m,n+p+1);
#ifdef GRAPHICS_PLEASE
        StartGraphics();
        Graphics(0,1,-0.00001,0.00001);
        Assekruis(2);
#endif
        for(i=0;i<=10*n;i++)
        {
            int d;
            xfloat x=(double)i/(10.*n);
/*          xfloat x=(double)rooster[p+i];*/
            for(d=0;d<=3;d++)
            {
                xfloat B;
                xfloat A;
                xfloat F;
                xfloat Fa;
                A=LinComBsplines(n,p,x,c,d,rooster);
                B=f(x,d);
                F=A-B;
                Fa=fabs(F);
                if (d==1)
                {
#ifdef GRAPHICS_PLEASE
                    DrawDot(x,F,14);
/*          DrawDot(x,B,15);*/
#endif
                    /* controleer de maxima */
                }
                if (Fa>maxI[d]) maxI[d]=Fa;
                if (x>=.25 && x<=.75 && Fa>maxi[d]) maxi[d]=Fa;
            }
        }
        printf("%d\t%8.3g\t%9.6g\t%9.6g\t%9.6g\t%9.6g\t"
            "%9.6g\t%9.6g\t%9.6g\t%9.6g\n",
            n,h,
            maxI[0],maxi[0],
            maxI[1],maxi[1],
            maxI[2],maxi[2],
            maxI[3],maxi[3]);
        free(c);
        free(rooster);
    }

void SpecialShow()
{
    int i;

```

```

xfloat *rooster;
xfloat **m;
xfloat *c;
xfloat h=0;
rooster=malloc(19*sizeof(xfloat));
assert(rooster);
for(i=0;i<19;i++) rooster[i]=i*2-18;
m=CreatePart1Matrix(12,3,rooster,NULL);
/* redelijk gevaarlijk, maar de f(=NULL) wordt toch
 * nooit anageroepen */
printf("Functiewaarden toevoegen...\n");
m[15][1]=0;
m[15][2]=1;
m[15][3]=2;
m[15][4]=3;
m[15][5]=3;
m[15][6]=3;
m[15][7]=3;
m[15][8]=4;
m[15][9]=5;
m[15][10]=6;
m[15][11]=6;
m[15][12]=6;
m[15][13]=6;
c=CreateCoefficients(12,3);
PrintMatrix(16,15,m);
SolveStelsel(m,c,14);
PrintMatrix(16,15,m);
PrintCoefficients(15,c);
DeleteMatrix(m,16);
StartGraphics();
Graphics(-12,12,-2,8);
Assekruis(2);
for(i=0;i<640*10;i++)
{
    xfloat x=((double)i*24/6400.)-12.;
    xfloat A;
    A=LinComBsplines(12,3,x,c,0,rooster);
    DrawDot(x,A,14);
}
while (1) ;
free(c);
free(rooster);
}

```

```

void main(void)
{
    pi=M_PI;
    pi2=pi*pi;
    pi3=pi2*pi;
    tweepi=pi*2;
}

```

```

printf(" \t \t0 \t \t1 \t \t2 \t \t3 \t\n"
       "n\th\tMaxI\tmaxi\tMaxI\tmaxi\tMaxI\tmaxi\tMaxI\tmaxi\n");
printf("\nPart 1, function a\n");
Show(2,functiona,CreatePart1Matrix);
Show(4,functiona,CreatePart1Matrix);
Show(8,functiona,CreatePart1Matrix);
Show(16,functiona,CreatePart1Matrix);
Show(32,functiona,CreatePart1Matrix);
Show(64,functiona,CreatePart1Matrix);
Show(128,functiona,CreatePart1Matrix);
printf("\nPart 1, function b\n");
Show(2,functionb,CreatePart1Matrix);
Show(4,functionb,CreatePart1Matrix);
Show(8,functionb,CreatePart1Matrix);
Show(16,functionb,CreatePart1Matrix);
Show(32,functionb,CreatePart1Matrix);
Show(64,functionb,CreatePart1Matrix);
Show(128,functionb,CreatePart1Matrix);
printf("\nPart 1, function c\n");
Show(2,functionc,CreatePart1Matrix);
Show(4,functionc,CreatePart1Matrix);
Show(8,functionc,CreatePart1Matrix);
Show(16,functionc,CreatePart1Matrix);
Show(32,functionc,CreatePart1Matrix);
Show(64,functionc,CreatePart1Matrix);
Show(128,functionc,CreatePart1Matrix);

printf("\nPart 2, function a\n");
Show(2,functiona,CreatePart2Matrix);
Show(4,functiona,CreatePart2Matrix);
Show(8,functiona,CreatePart2Matrix);
Show(16,functiona,CreatePart2Matrix);
Show(32,functiona,CreatePart2Matrix);
Show(64,functiona,CreatePart2Matrix);
Show(128,functiona,CreatePart2Matrix);
printf("\nPart 2, function b\n");
Show(2,functionb,CreatePart2Matrix);
Show(4,functionb,CreatePart2Matrix);
Show(8,functionb,CreatePart2Matrix);
Show(16,functionb,CreatePart2Matrix);
Show(32,functionb,CreatePart2Matrix);
Show(64,functionb,CreatePart2Matrix);
Show(128,functionb,CreatePart2Matrix);
printf("\nPart 2, function c\n");
Show(2,functionc,CreatePart2Matrix);
Show(4,functionc,CreatePart2Matrix);
Show(8,functionc,CreatePart2Matrix);
Show(16,functionc,CreatePart2Matrix);
Show(32,functionc,CreatePart2Matrix);
Show(64,functionc,CreatePart2Matrix);
Show(128,functionc,CreatePart2Matrix);

SpecialShow();

```

}