# Polynominterpolation mit Matlab.

Die Matlab-Funktion polyfit

```
a = polyfit(x,f,n-1);
```

berechnet die Koeffizienten

```
a = (a(1),a(2),...,a(n));
```

des Interpolationspolynoms

```
p(x) = a(1)*x^(n-1) + a(2)*x^(n-2) + ... + a(n-1)*x + a(n);
```

zu den Daten

```
x = (x(1),x(2),...,x(n));
f = (f(1),f(2),...,f(n));
```

Polynome kann man mit der Matlab-Funktion polyval auswerten.

## 8.3 Spline-Interpolation

Sei $\Delta_n$ eine Unterteilung des Intervalls $[a, b]$:

$$\Delta_n \quad : \quad a = x_0 < x_1 < \ldots < x_{n-1} < x_n = b$$

mit Teilintervallen $[x_{j-1}, x_j]$, $j = 1, \ldots, n$.

**Definition:** *Eine Funktion* $S : [a, b] \to \mathbb{R}$ *heißt* kubischer Spline, *falls*

- $S \in C^2([a, b])$, *d.h.* $S$ *ist zweimal stetig differenzierbar auf* $[a, b]$;
- $S$ *ist auf jedem Teilintervall* $[x_{j-1}, x_j]$, $1 \leq j \leq n$, *ein kubisches Polynom:*

$$S(x)\big|_{[x_{j-1}, x_j]} \equiv s_j(x) = a_j + b_j(x - x_{j-1}) + c_j(x - x_{j-1})^2 + d_j(x - x_{j-1})^3.$$

□

**Ziel:** Interpolation der Daten $(x_j, f_j)$, $0 \leq j \leq n$, mit einem kubischen Spline $S$, so dass

$$S(x_j) = f(x_j), \qquad \text{für } 0 \leq j \leq n.$$

# Interpolation mit kubischen Splines.

**Beobachtungen:**

- Ein kubischer Spline besitzt $4n$ Parameter, die wie folgt bestimmt werden.
  - Interpolationseigenschaft:

$$s_j(x_{j-1}) = f_{j-1} \quad \text{und} \quad s_j(x_j) = f_j \qquad \text{für alle } 1 \le j \le n;$$

  - Stetigkeit der Ableitung:

$$s_j'(x_j) = s_{j+1}'(x_j) \qquad \text{für alle } 1 \le j \le n-1;$$

  - Stetigkeit der zweiten Ableitung:

$$s_j''(x_j) = s_{j+1}''(x_j) \qquad \text{für alle } 1 \le j \le n-1;$$

- Dies sind insgesamt $4n-2$ Gleichungen für $4n$ Parameter.
- **OBS!** Es fehlen noch zwei Bedingungen.

# Zwei weitere Nebenbedingungen.

**Definition:** *Ein kubischer Spline heißt*

- `natürlicher Spline`, *falls* $S''(a) = S''(b) = 0$;
- `periodischer Spline`, *falls* $S^{(i)}(a) = S^{(i)}(b)$, $i = 0, 1, 2$;
- `vollständiger Spline`, *falls* $S'(a) = f'(a)$ *und* $S'(b) = f'(b)$.

**Beachte:** Jede drei obigen Bedingungen liefert zwei weitere Gleichungen.

**Satz:** *Unter allen interpolierenden* $C^2$*-Funktionen minimiert der* `natürliche kubische Spline` *das Funktional*

$$I[y] := \int_a^b (y''(x))^2 \, dx$$

□

**Bemerkung:** Das Funktional $I$ mißt die Krümmung von $y$ *approximativ*. □

# Berechnung des natürlichen kubischen Splines.

Sei $S$ auf dem Teilintervall $[x_{j-1}, x_j]$ gegeben durch

$$S(x)\big|_{[x_{j-1},x_j]} \equiv s_j(x) = a_j + b_j(x - x_{j-1}) + c_j(x - x_{j-1})^2 + d_j(x - x_{j-1})^3,$$

so gilt

$$\begin{aligned} a_j &= f_{j-1} \\ b_j &= \frac{f_j - f_{j-1}}{h_j} - \frac{2M_{j-1} + M_j}{6} h_j \\ c_j &= \frac{M_{j-1}}{2} \\ d_j &= \frac{M_j - M_{j-1}}{6h_j} \end{aligned}$$

wobei $h_j = x_j - x_{j-1}$ für $1 \le j \le n$.

Die Momente $M_j = S''(x_j)$ lösen ein lineares System mit *Tridiagonalmatrix*.

# Herleitung des Splines mit Momentenmethode.

Der gewählte Ansatz

$$M_j := S''(x_j), \qquad \text{für } 0 \le j \le n,$$

heißt Momentenmethode: $s_j''(x)$ ist eine Gerade mit

$$s_j''(x) = M_{j-1} + \frac{M_j - M_{j-1}}{h_j}(x - x_{j-1}) \qquad \text{mit } h_j = x_j - x_{j-1}$$

Zweifache Integration über Intervall $[x_{j-1}, x]$ liefert

$$\begin{aligned} s_j'(x) &= B_j + M_{j-1}(x - x_{j-1}) + \frac{M_j - M_{j-1}}{2h_j}(x - x_{j-1})^2 \\ s_j(x) &= A_j + B_j(x - x_{j-1}) + \frac{M_{j-1}}{2}(x - x_{j-1})^2 + \frac{M_j - M_{j-1}}{6h_j}(x - x_{j-1})^3 \end{aligned}$$

mit Integrationskonstanten $A_j, B_j$.

# Lösung der Bedingungsgleichungen.

Aus den Interpolationsbedingungen $s_j(x_{j-1}) = f_{j-1}$ und $s_j(x_j) = f_j$ folgt direkt

$$A_j = f_{j-1} \quad \text{und} \quad B_j = \frac{f_j - f_{j-1}}{h_j} - \frac{h_j}{6}(M_j + 2M_{j-1}), \tag{1}$$

mit der Stetigkeit von $S'$ bei $x_j$, $1 \le j < n$, d.h. $s'_j(x_j) = s'_{j+1}(x_j)$ weiterhin

$$B_j + \frac{M_j + M_{j-1}}{2} h_j = B_{j+1} \qquad \text{für } 1 \le j \le n-1. \tag{2}$$

Einsetzen von (1) in (2) ergibt schließlich $n-1$ lineare Gleichungen

$$h_j M_{j-1} + 2(h_j + h_{j+1})M_j + h_{j+1} M_{j+1} = 6\left(\frac{f_{j+1} - f_j}{h_{j+1}} - \frac{f_j - f_{j-1}}{h_j}\right),$$

$1 \le j \le n-1$, für die $n-1$ *unbekannten* Momente $M_1, \ldots, M_{n-1}$.

**Beachte:** Die Momente $M_0 = 0$ und $M_n = 0$ sind bereits *bekannt*.

# Tridiagonalsystem für die Momente.

Das hergeleitete $(n-1) \times (n-1)$ lineare System hat die Form

$$\begin{bmatrix} 2k_1 & h_2 & & & \\ h_2 & 2k_2 & h_3 & & \\ & \ddots & \ddots & \ddots & \\ & & h_{n-2} & 2k_{n-2} & h_{n-1} \\ & & & h_{n-1} & 2k_{n-1} \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-2} \\ M_{n-1} \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_{n-2} \\ d_{n-1} \end{bmatrix}$$

mit $h_j = x_j - x_{j-1}$, $1 \le j \le n$, $k_j = h_j + h_{j+1}$, $1 \le j \le n-1$, und

$$d_j = 6\left(\frac{f_{j+1} - f_j}{h_{j+1}} - \frac{f_j - f_{j-1}}{h_j}\right) \qquad \text{für } 1 \le j \le n-1,$$

sowie den Randwerten $M_0 = M_n = 0$.

# Abschließende Bemerkungen zu Splines.

- Der natürliche kubische Spline kann *effizient* berechnet werden, nämlich durch Lösen des Tridiagonalsystems in nur $\mathcal{O}(n)$ Schritten.
- Eine Splineinterpolante vermeidet (unerwünschte) Oszillationen.
- Für $f \in C^4$ gilt die asymptotische Fehlerabschätzung

$$|f(x) - S(x)| = \mathcal{O}(h^4), \qquad h \to 0$$

  wobei $h = \max_{1 \le j \le n} h_j$.
- Verwendet man einen *vollständigen* Spline mit Randbedingungen

$$S'(a) = f'(a) \quad \text{und} \quad S'(b) = f'(b)$$

  so erhält man ein Tridiagonalsystem, das effizient gelöst werden kann.
- Verwendet man *periodische* Splines, so erhält man kein Tridiagonalsystem. Die Lösung kann dennoch effizient in $\mathcal{O}(n)$ Schritten berechnet werden.

# Spline-Interpolation mit Matlab.

Die Matlab-Funktion spline

```
s = spline(x,f);
```

berechnet die Darstellung s einer kubischen Splinefunktion zu den Daten

```
x = (x(1),x(2),...,x(n));
f = (f(1),f(2),...,f(n));
```

Splines kann man mit der Matlab-Funktion ppval wie folgt auswerten.

```
t = linspace(-1,1,1000);  % 1000 uniforme Knoten in [-1,1]
y = ppval(s,t);           % Auswertung des Splines s
```