

---

# Echtzeit–Videoverarbeitung

## Filteralgorithmen

Marcus Barkowsky

---

# Überblick

## Zahlendarstellungen

## Filter zur Qualitätssteigerung

- Rauschunterdrückung
- Kantenverstärkung

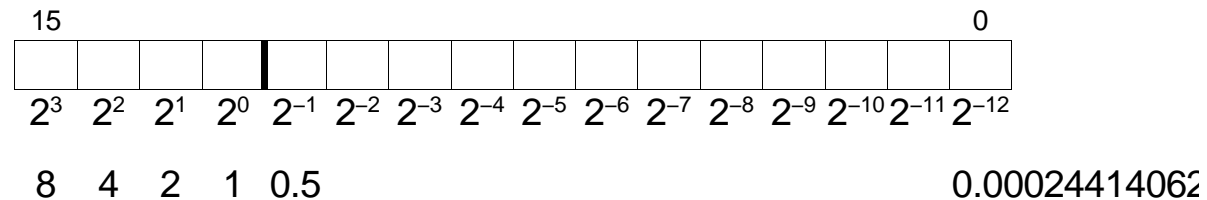
## Interpolationsfilter

- FIR–Filter
- Kubische Filter
- Splines

# Zahlendarstellungen

Unterscheidung nach

- Art der Zahlendarstellung
  - Festkommazahlen: Fixed Point, der „Punkt“ ist an einer Stelle festgelegt.  
insbesondere: Integer (Punkt ganz rechts) und Fractional (Punkt ganz links)  
Probleme: Eingeschränkter Wertebereich, Überlaufproblematik



- Gleitkommazahlen: Floating Point, Darstellung mit Exponent und Mantisse.  
Probleme: Auflösung ändert sich mit Exponent, dadurch Verschleierung von ungenügender Präzision, Fehlerausbreitung bei Mehrfachoperationen

---

# Zahlendarstellungen

Unterscheidung nach

- Zahlenbasis
  - Basis 2
  - Basis 10: BCD, besonders im kaufmännischen Bereich üblich, wegen exakter Repräsentation der benötigten Zahlen. Beispiel: 0.1 in binär

$$0 \cdot 2^0 + 0 \cdot 2^{-1} + 0 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} + 1 \cdot 2^{-5} + 0 \cdot 2^{-6} + 0 \cdot 2^{-7} + 1 \cdot 2^{-8} + 1 \cdot 2^{-9} = 0.099609375$$

- Vorzeichenbehandlung
  - Getrennte Behandlung des Vorzeichens: Sign–Magnitude  
Vorteil: einfache Multiplikation
  - 2er Komplement  
Vorteil: einfache Addition

---

# Ganzzahldarstellung (Integer)

Benötigte Datenbreite in Bits bei arithmetischen Operationen mit 2 Operanden:

- Addition:

$$\text{Datenbreite}_{\max} + 1$$

- Subtraktion:

$$\text{Datenbreite}_{\max} + 1$$

- Multiplikation:

$$\text{Datenbreite}_1 + \text{Datenbreite}_2$$

- Ganzzahldivision:

$$\text{Datenbreite}_1$$

- Restbildung (Modulo):

$$\text{Datenbreite}_1$$

---

# Filter zur Qualitätssteigerung

Eigenschaften:

- Rauschunterdrückung
- Kantenverstärkung

Verfahren mit geringer Komplexität:

- Median
- Sobel
- Schärfungsfilter
- 2 Helfer Signal

# Median-Filter

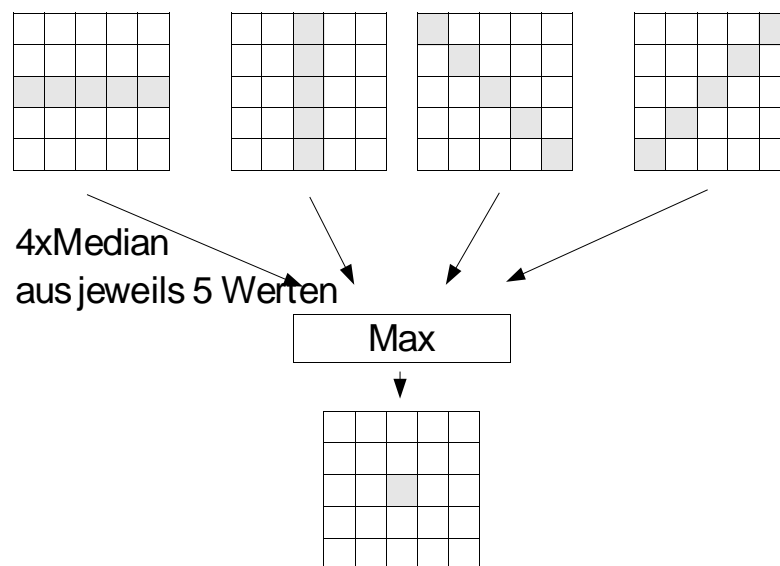
Grundprinzip:

- Ersetzung des aktuellen Pixelwertes durch mittleren Wert der Umgebung, z.B.:

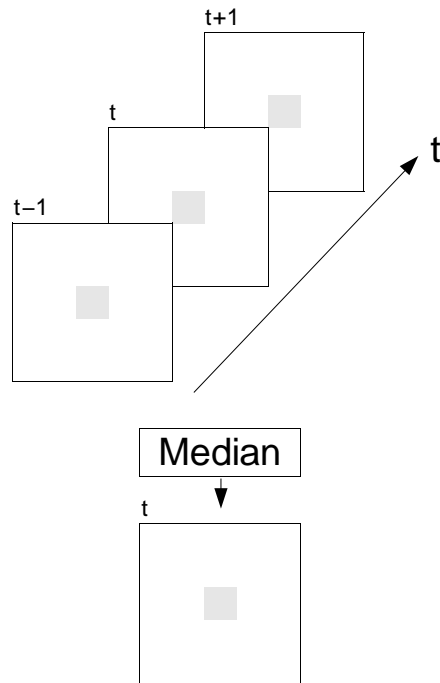
$\text{Median}\{20, 25, 100, 26, 20\} = \text{Median}\{20, 20, \underline{25}, 26, 100\} = 25$

→ Der Pixelwert 100 wird durch 25 ersetzt  $\{20, 25, 25, 26, 20\}$

- Praktische Realisierung:



# Zeitliche Median-Filterung



Problem:

Objektbewegung  $\rightarrow$  große Differenz  $\rightarrow$  falsche Substitution

Lösungsansatz:

Schwellwertentscheider:

Substitution nur bei kleinen Pixeldifferenzen (z.B.  $<30$ )

# Zeitliche Median-Filterung



---

# Sobelfilter

Kantendetektionsfilter:

Faltung mit den folgenden Matrizen:

1	2	1
0	0	0
-1	-2	-1

horizontal

1	0	-1
2	0	-2
1	0	-1

vertikal

0	1	2
-1	0	1
-2	-1	0

diagonal

2	1	0
1	0	-1
0	-1	-2

diagonal

Resultat:

4 Werte, welche die Stärke einer Kante in der jeweiligen Richtung repräsentieren

---

## Kombiniertes Median – Sobel Verfahren:

- Bestimme Maximum der Sobel Filterwerte
- Falls Maximalwert  $< 30 \rightarrow$  Rauschunterdrückung mittels Median
- Sonst: Kantenverstärkung
  - Sortiere Pixelwerte der 3x3 Umgebung des aktuellen Pixels
  - Mittlere die 3 kleinsten und die 3 größten Pixelwerte
  - Berechne die Differenz des aktuellen Pixels zu diesen beiden Mittelwerten
  - Suche geringere Differenz und verschiebe entsprechend den Amplitudenwert des aktuellen Pixels in Richtung Maximum oder Minimum

---

# Schärfungsfilter

Filter 1:

Faltung mit folgendem Kern:

-1	-1	-1
-1	9	-1
-1	-1	-1

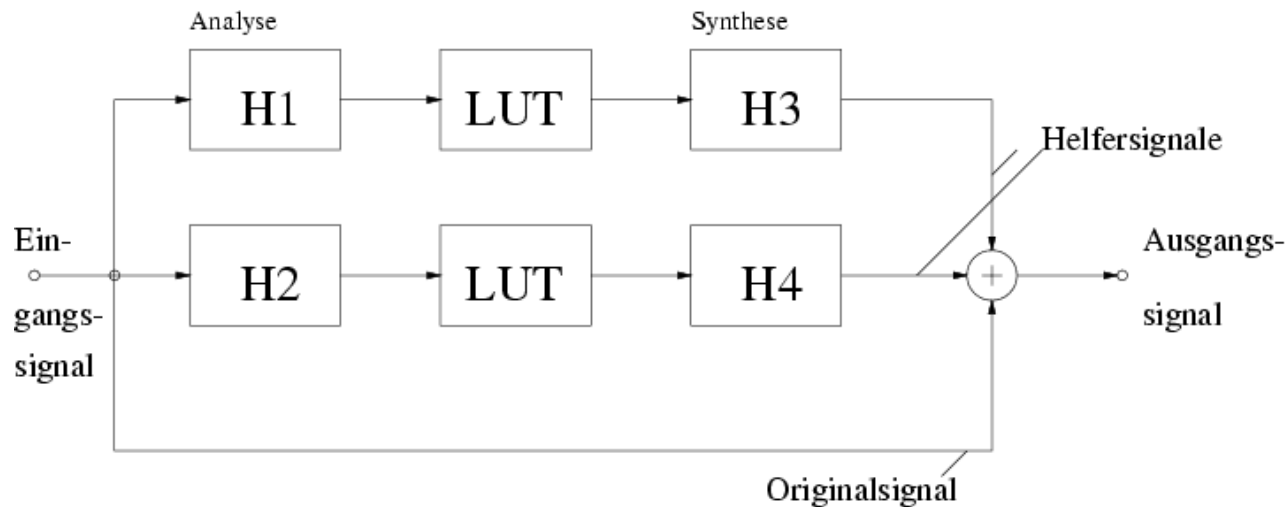
Intensität: Lineare Gewichtung des Ergebnisses

Filter 2: Schneller separierbarer Schärfungsfilter

Filterung mit FIR-Koeffizienten

$$\frac{1}{4} \cdot (-1, 6, -1)$$

## 2 Helfer-Signal Methode



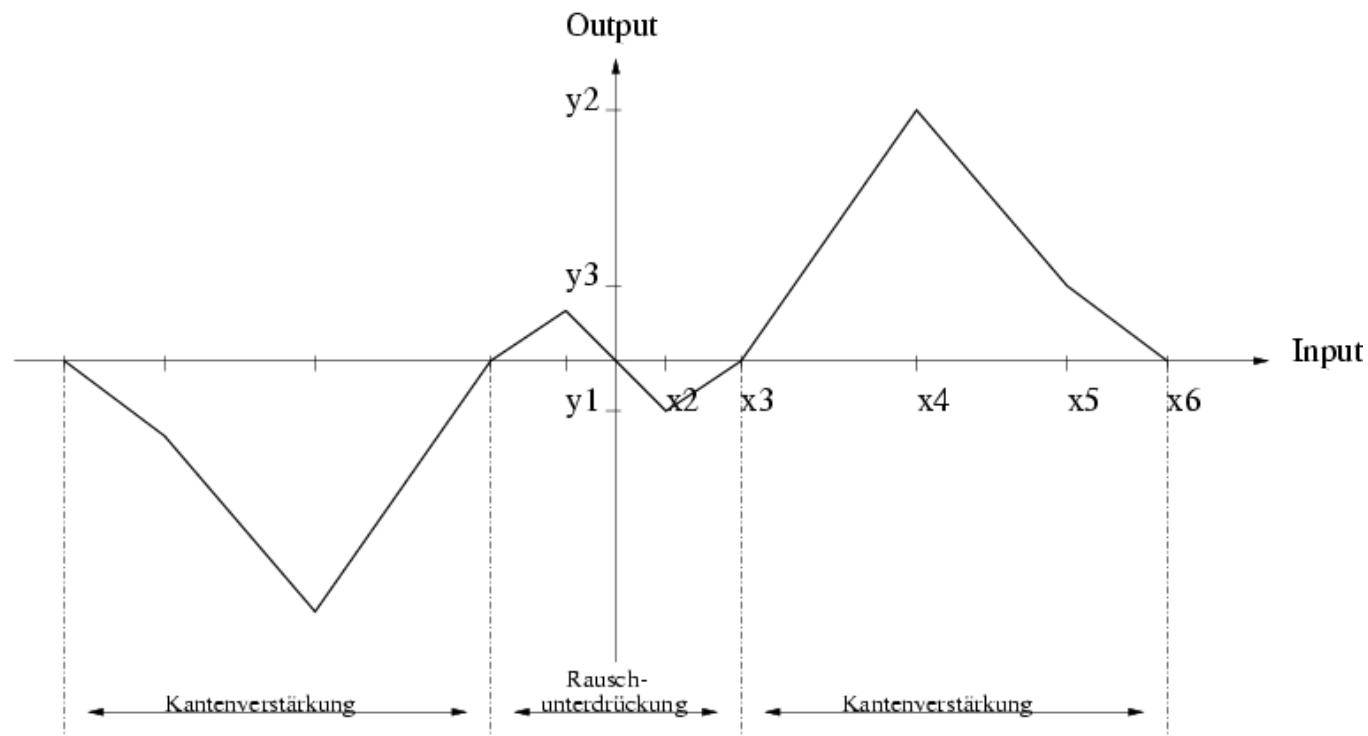
mit:

$$H1 = H3 = \begin{pmatrix} +1 & 0 \\ 0 & -1 \end{pmatrix} \quad H2 = H4 = \begin{pmatrix} 0 & +1 \\ -1 & 0 \end{pmatrix}$$

Beispiel:  $\begin{bmatrix} 50 & 70 \\ 70 & 20 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = 30; \quad \text{LUT}(30)=10; \quad 10 \bullet \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} + \begin{bmatrix} 50 & 70 \\ 70 & 20 \end{bmatrix} = \begin{bmatrix} 60 & 70 \\ 70 & 10 \end{bmatrix}$

- Die Faltung kann entweder für jedes Pixel oder nur für jedes zweite Pixel erfolgen.

## 2 Helfer-Signal Methode



Hier:  $y_1 = -5$ ,  $y_2 = 20$ ,  $y_3 = 8$ ,  $x_2 = 10$ ,  $x_3 = 20$ ,  $x_4 = 60$ ,  $x_5 = 170$ ,  $x_6 = 250$

---

# Interpolationsfilter

## FIR-Filter

- Andreas Hutters MPEG-4 Filter
- Iterativ optimierter Si-Filter
- Kubischer Filter
- Kubischer Warping Filter

## Splines

- Grundlagen
- Gleichungssystem
- Matrix-Version
- IIR-FIR Implementierung
- Anwendungen

---

# Digitale FIR–Filter: Anwendungsgebiete

## Interpolation

- Skalierung
- Rotation
- Verzerrung

## Qualitätssteigerung

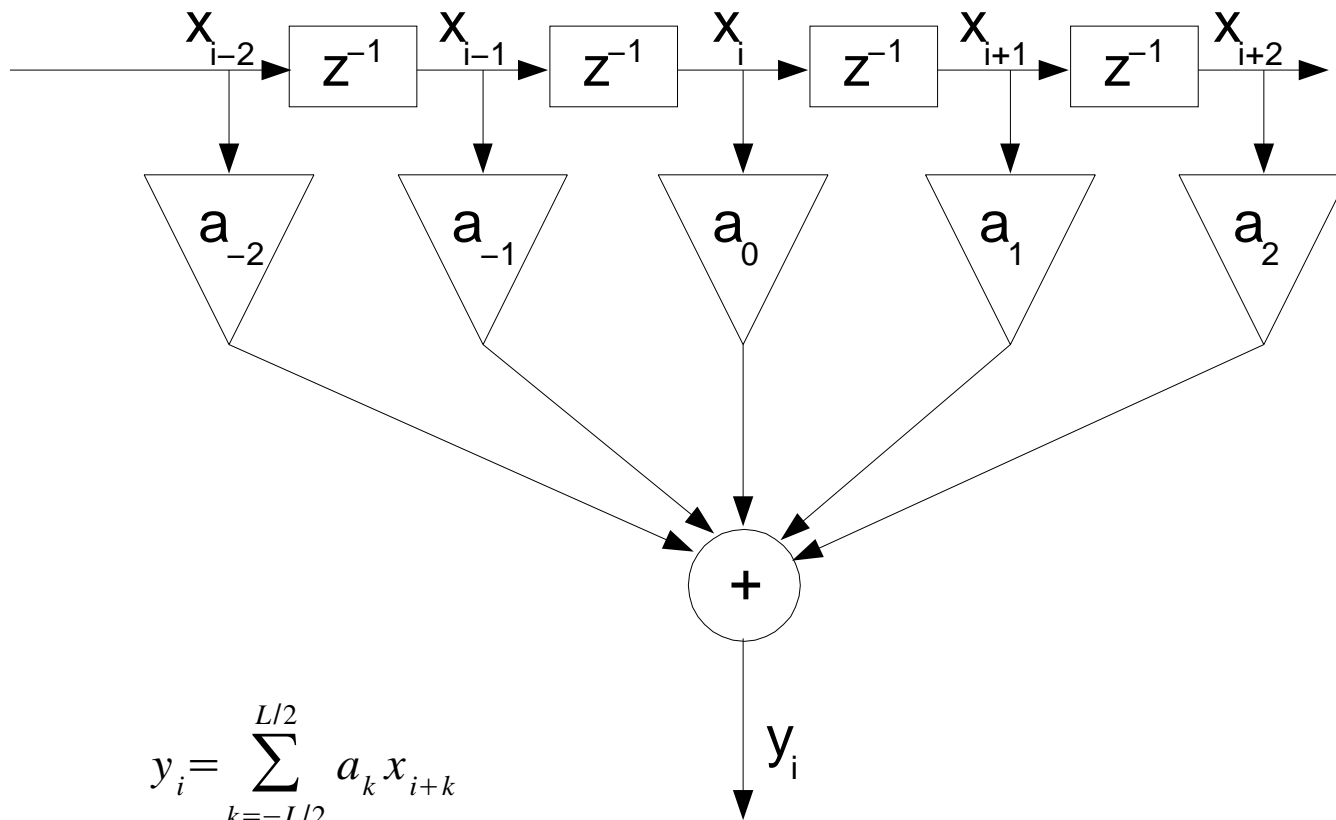
- Rauschunterdrückung
- Verwaschung (Blurring)
- Schärfung (Sharpening)

## Detektion

- Kantendetektion

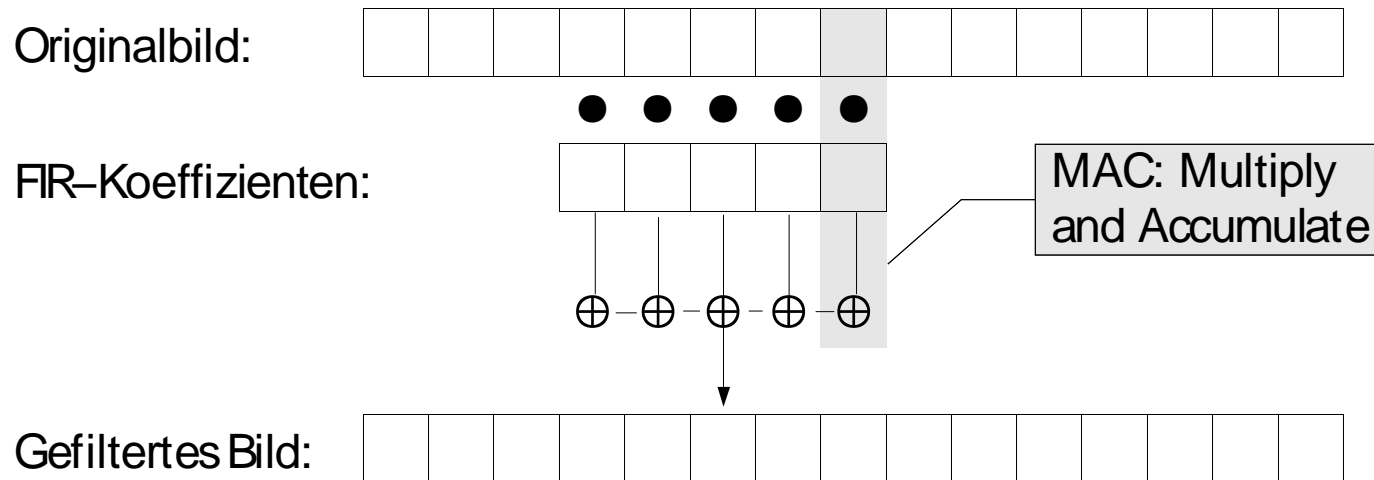
# Digitale FIR-Filter: Aufbau

Allgemein:



# Digitale FIR-Filter: Aufbau

Bildverarbeitung:



# Digitale FIR–Filter: Ganzzahl Implementierung

- Filterkoeffizienten müssen quantisiert werden
- Ergebnis ist stark quantisiert → Mehrfachoperationen vermeiden
- Wertebereich sollte ausgenutzt werden, aber Überlauf muß vermieden werden

Einfacher Ansatz:

$$\text{Bits}(y) = 1 + [\text{Bits}(x) + \text{Bits}(a)] + \log_2(L)$$

Vorzeichenbit      Multiplikation      Addition

Nachteil:  $a_i \ll a_{\max}$ , daher:

$$\text{Bits}(y) = 1 + \text{Bits}(x) + \text{Bits}\left(\sum_{i=-L/2}^{L/2} \begin{bmatrix} a_i & : & a_i > 0 \\ 0 & : & \text{sonst} \end{bmatrix}\right)$$

Vorzeichenbit

---

# Beispiel: Andreas Hutter MPEG-4 Filter

Verwendete Filter:

A: { 5, 11, 11, 5 }

B: { 2, 0, -4, -3, 5, 19, 26, 19, 5, -3, -4, 0, 2 }

C: { -5, -4, 0, 5, 12, 19, 24, 26, 24, 19, 12, 5, 0, -4, -5 }

Erzeugung von CIF

Luminanz: 704x288 → B → 352x288

Chrominanz: 352x288 → B → 176x288 → A → 176x144

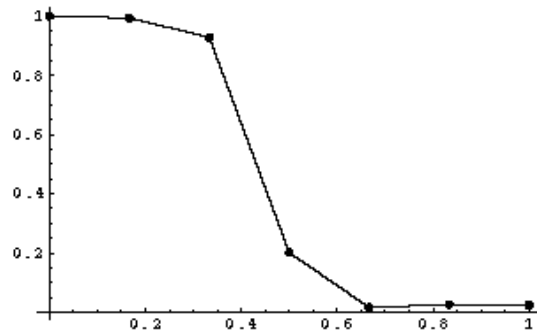
Erzeugung von QCIF

Luminanz: 704x288 → C → 176x288 → B → 176x144

Chrominanz: 352x288 → C → 88x288 → B → 88x144 →  
A → 88x72

# Analyse: Andreas Hutter MPEG-4 Filter

Frequenzraumanalyse für B-Filter:

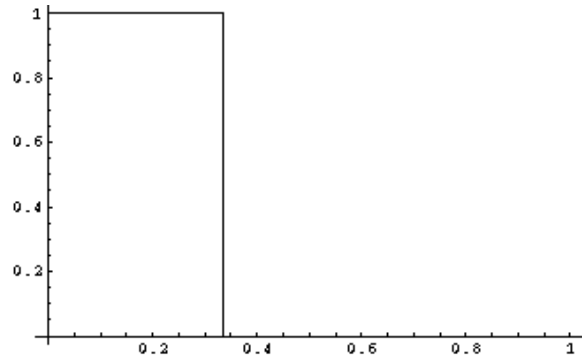


- Kein Aliasing
- Hohe Frequenzen bedämpft
- Glatter Frequenzgang im Durchlaßbereich
- Großer Sperrabstand

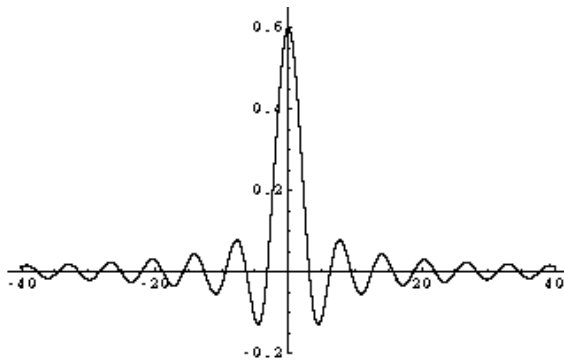
# Aufgabe: Filter für 1/3 entwerfen

Ansatz: Tiefpaßfilter

Theorie:



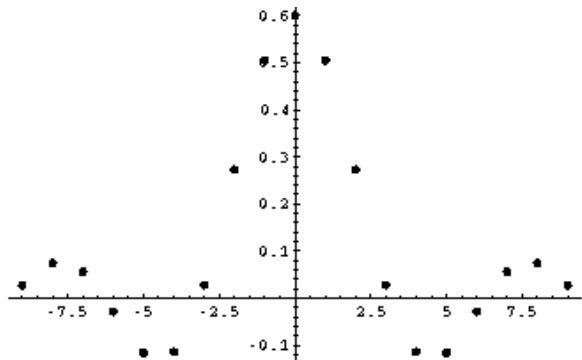
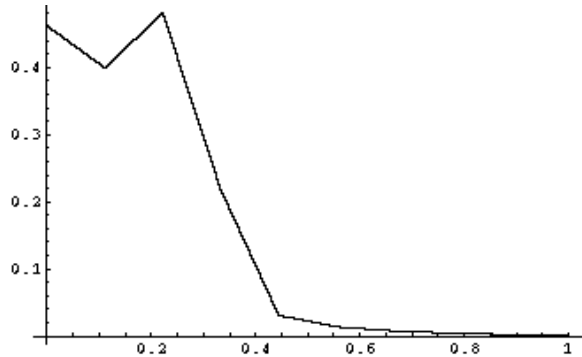
$$H(j\omega) = \begin{cases} 1 & \text{für } -F_G < j\omega < F_G \\ 0 & \text{sonst} \end{cases}$$



$$h(x) = 2 \cdot F_G \frac{\sin(x) \cdot \pi F_G}{x \cdot \pi F_G}$$

# Aufgabe: Filter für 1/3 entwerfen

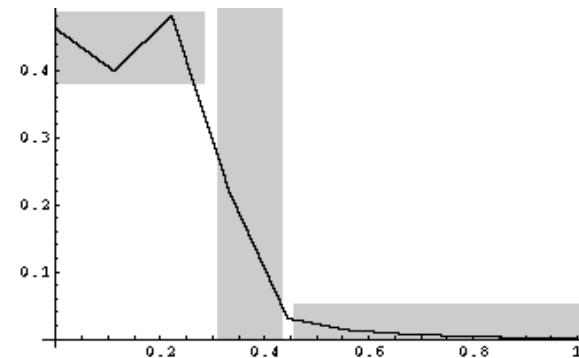
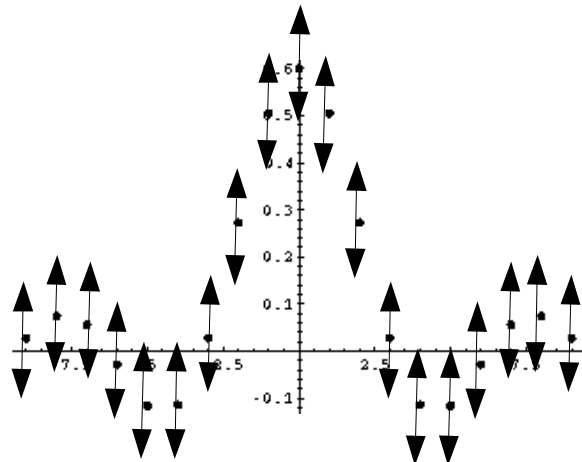
Praxis: 19 Filterkoeffizienten



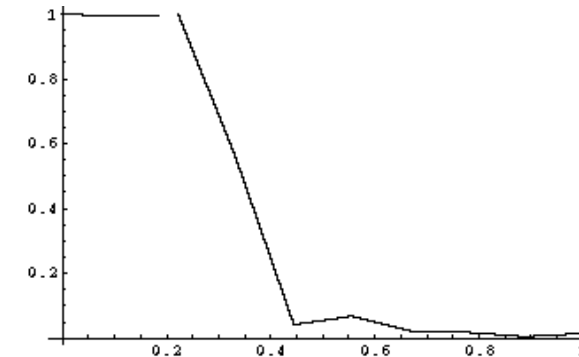
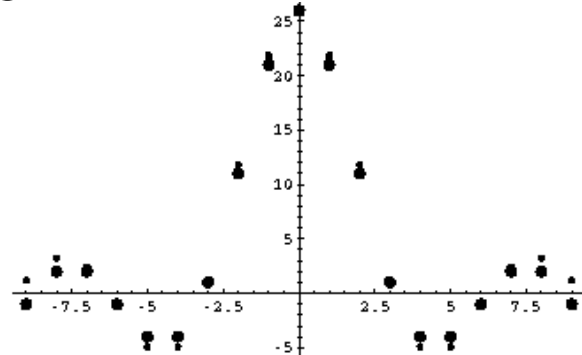
# Aufgabe: Filter für 1/3 entwerfen

Idee: Iteratives Vorgehen zur Verbesserung des Frequenzverlaufs

Bewertungskriterien: Durchlaßbereich, Steigung, Sperrbereich

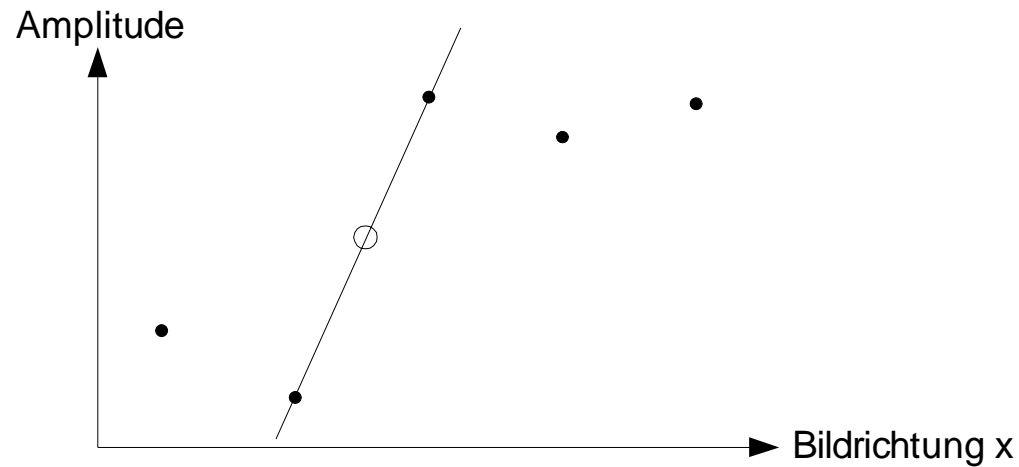


Ergebnis:



# Vergrößerung:

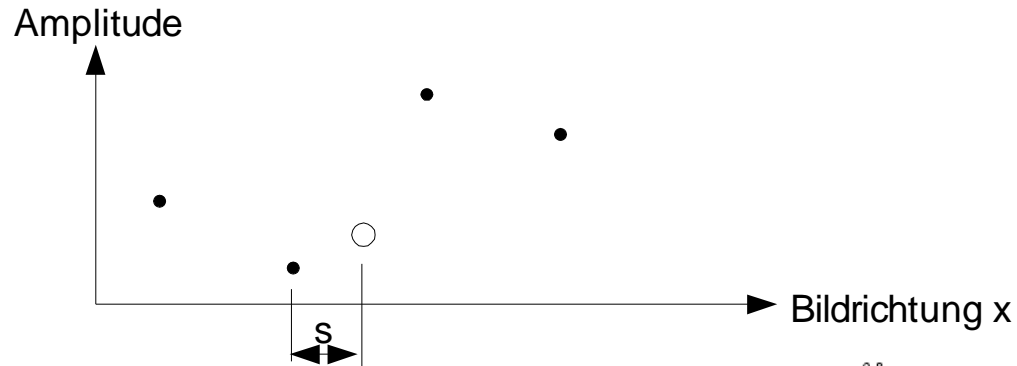
Lineare Interpolation:



- + Sehr schnell realisierbar
- Geringe Ausdehnung
- Bei nicht geradzahligen Vergrößerungsfaktoren starker Detailverlust
- Verwaschung von Kanten

# Kubischer Interpolationsfilter:

Berechnung von 4 FIR–Koeffizienten für die 4 benachbarten Pixel:  
 Mit  $s$ :  $x$ –Offset zum linken Interpolationskoeffizient  $0 < s < 1$

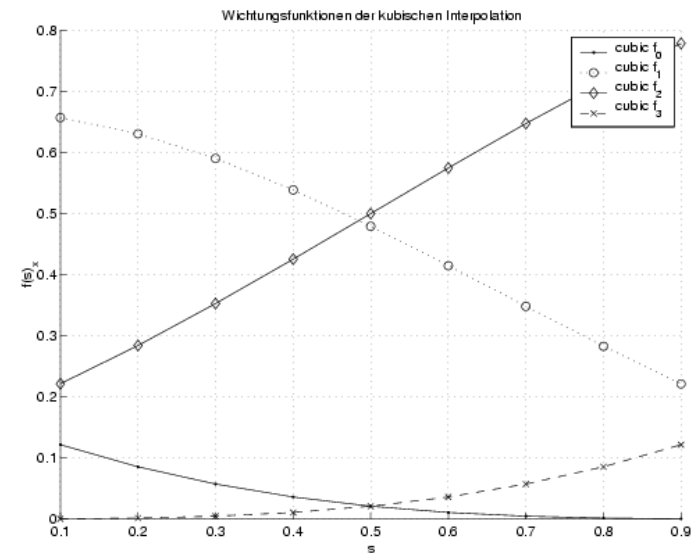


$$a_{-1}(s) = \frac{(3+s)^3 - 4(2+s)^3 + 6(1+s)^3 - 4s^3}{6}$$

$$a_0(s) = \frac{(2+s)^3 - 4(1+s)^3 + 6s^3}{6}$$

$$a_1(s) = \frac{(1+s)^3 - 3s^3}{6}$$

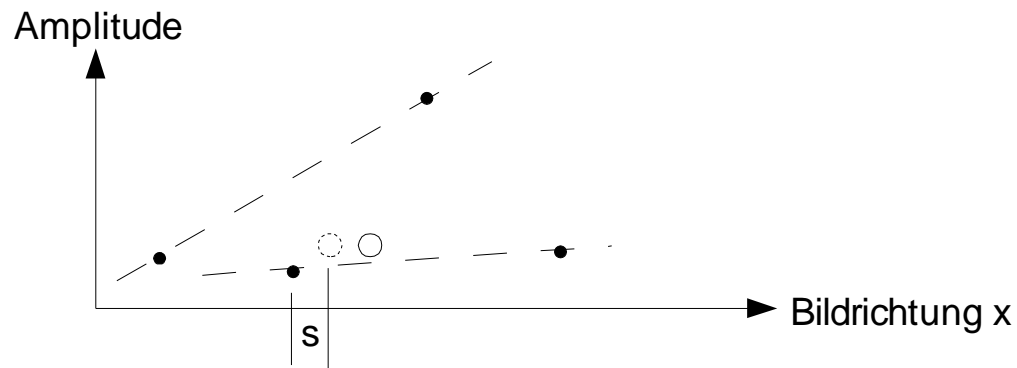
$$a_2(s) = \frac{s^3}{6}$$



# Kubischer Warping Interpolationsfilter:

Virtuelle Annäherung des Amplitudenwertes an diejenige Seite, die den höheren Gradientenwert besitzt.

Vorteil: Kantenverstärkung



$$A = \frac{|f(x_{k+1}) - f(x_{k-1})| - |f(x_{k+2}) - f(x_k)|}{255}$$

---

# Interpolationsfunktionen

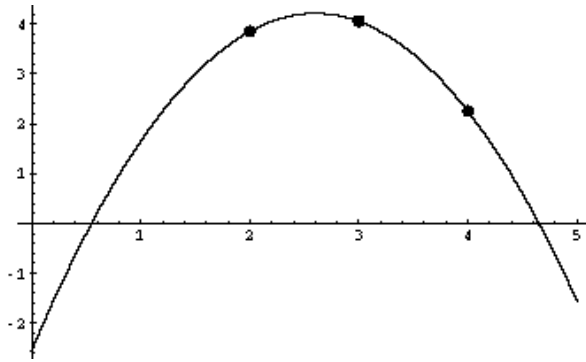
Kurze FIR–Filter (<20 Filtertaps)

- + Einfach und schnell zu implementieren
- Schlechte Signalregenerierungseigenschaften
- Empfindlich gegen Rauschen

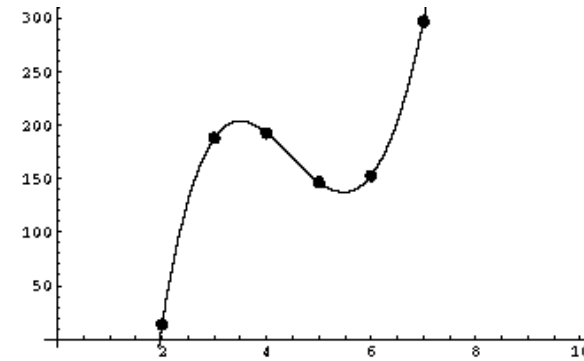
Polynomialansatz: Erzeugung kontinuierlicher Funktionen

- + Berücksichtigung des gesamten Signals
- + Möglichkeit zur Signalapproximation
- hoher Rechenaufwand

# Interpolation



3 Punkte: Parabel



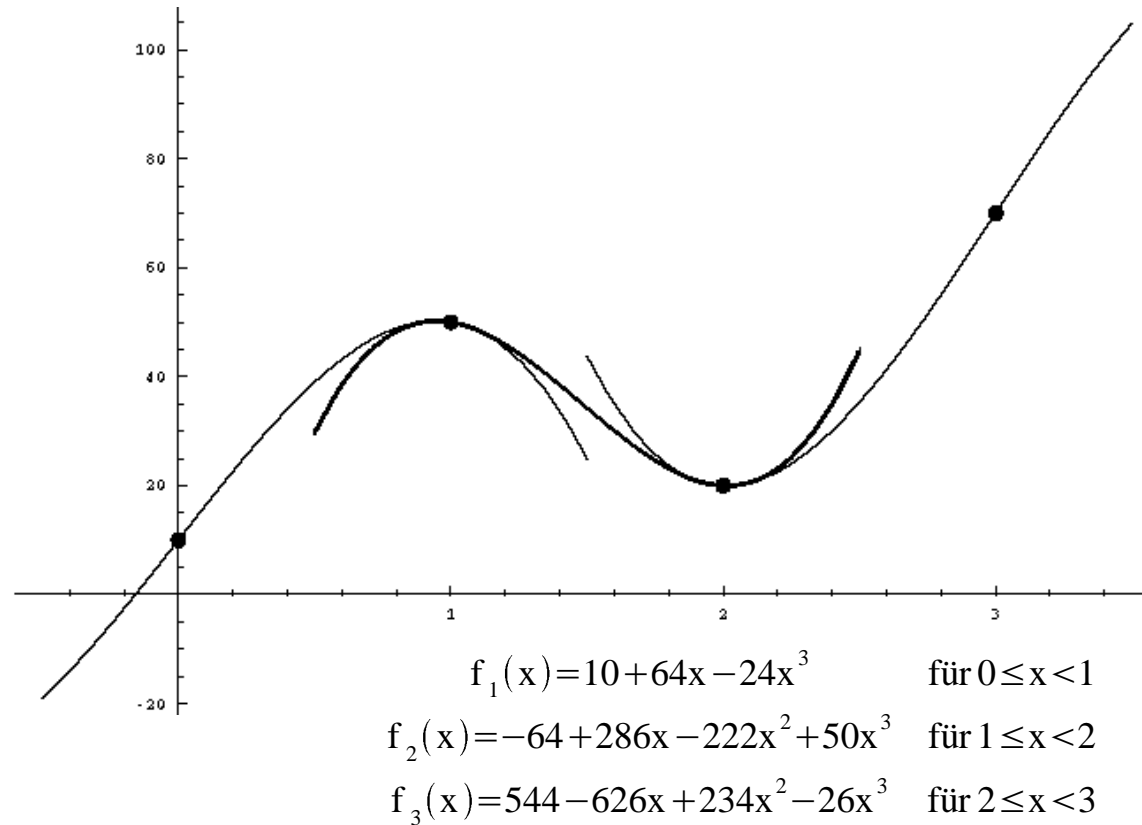
6 Punkte: Funktion 5. Grades

Für  $x$  Punkte wird ein Polynom vom Grad  $x-1$  benötigt

- Rechenzeitproblematik für  $>20$  Punkte
- Es kommt zu starken Überschwingern, der Kurvenverlauf wird „unnatürlich“

# Splines: Stückweise Polynome

Ansatz: Interpolation oder Approximation durch Kurvenstücke von Polynomen niedrigen Grades (z.B. Grad 3)



---

# Splines: Berechnung

3 kubische Funktionen:

$$X_1(x) = a_1 x^3 + b_1 x^2 + c_1 x + d_1$$

$$X_2(x) = a_2 x^3 + b_2 x^2 + c_2 x + d_2$$

$$X_3(x) = a_3 x^3 + b_3 x^2 + c_3 x + d_3$$

4x3 Unbekannte → 12 linear unabhängige Bedingungen:

6 durch die 4 vorgegebenen Punkte (auch Stetigkeit)

$$P_0 = X_1(0)$$

$$P_1 = X_1(1) = X_2(1)$$

$$P_2 = X_2(2) = X_3(2)$$

$$P_3 = X_3(3)$$

2 durch Differenzierbarkeit (Glattheitsbedingung)

$$X_1'(1) = X_2'(1) \quad X_2'(2) = X_3'(2)$$

2 durch Differenzierbarkeit 2. Ordnung (Steigung)

$$X_1''(1) = X_2''(1) \quad X_2''(2) = X_3''(2)$$

2 durch Randwertbedingung (am Rand wie Gerade)

$$X_1''(0) = 0 \quad X_3''(3) = 0$$

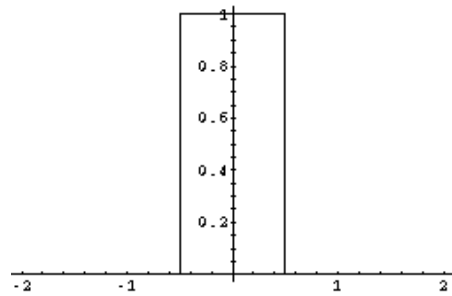
# Splines: Mathematica Anweisungen

```
X[i_]:=Function[t,a[i]+b[i] t+d[i]t^2+d[i]t^3]
es0={X[0][0]==10,X[0][1]==50}
es1={X[1][1]==50,X[1][2]==20}
es2={X[2][2]==20,X[2][3]==70}
ec11={(D[X[0][t],t] /. t->1)==(D[X[1][t],t]/. t->1)}
ec12={(D[X[1][t],t] /. t->2)==(D[X[2][t],t]/. t->2)}
ec21={(D[X[0][t],t,t]/.t->1)==(D[X[1][t],t,t]/.t->1)}
ec22={(D[X[1][t],t,t]/.t->2)==(D[X[2][t],t,t]/.t->2)}
ec20={(D[X[0][t],t,t]/.t->0)==0}
ec23={(D[X[2][t],t,t]/.t->3)==0}
eq=Join[es0,es1,es2,ec11,ec12,ec21,ec22,ec20,ec23]
sol=Solve[eq][[1]];
X[0][t] /. sol
X[1][t] /. sol
X[2][t] /. sol
Cs[t_]:=X[0][t] /. sol) /; 0<=t && t<1
Cs[t_]:=X[1][t] /. sol) /; 1<=t && t<2
Cs[t_]:=X[2][t] /. sol) /; 1<=2 && t<3
Plot[Cs[t],{t,0,3},Compiled->False]
```

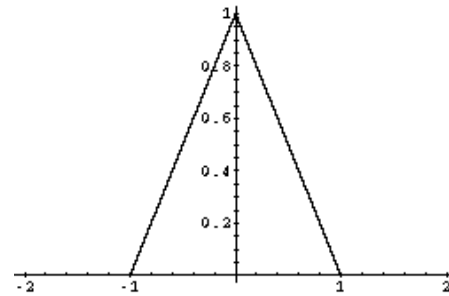
# Splines: Andere Darstellungsform: Basis-Funktionen

Einschränkung: Punkte müssen gleichen Abstand haben

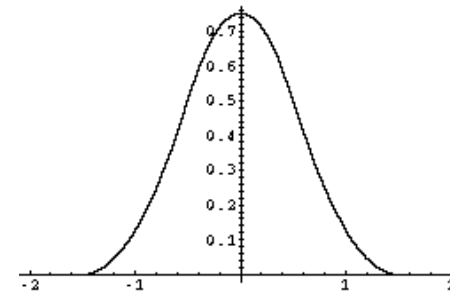
Rekursionsvorschrift:  $\beta^n = \beta^0 * \beta^{n-1}$



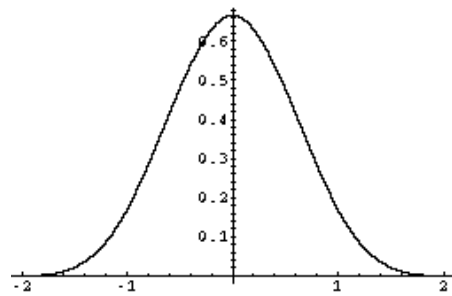
$\beta^0$



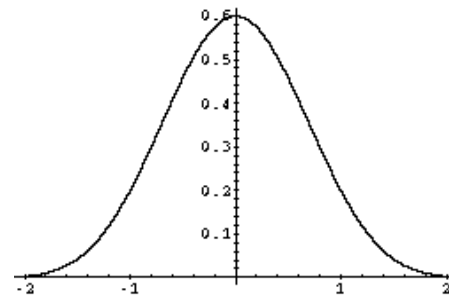
$\beta^1$



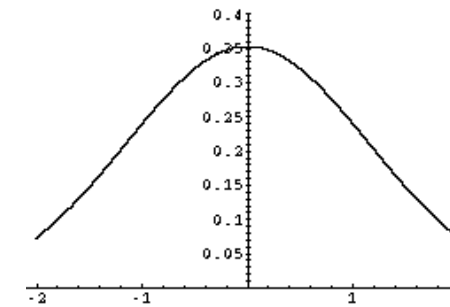
$\beta^2$



$\beta^3$

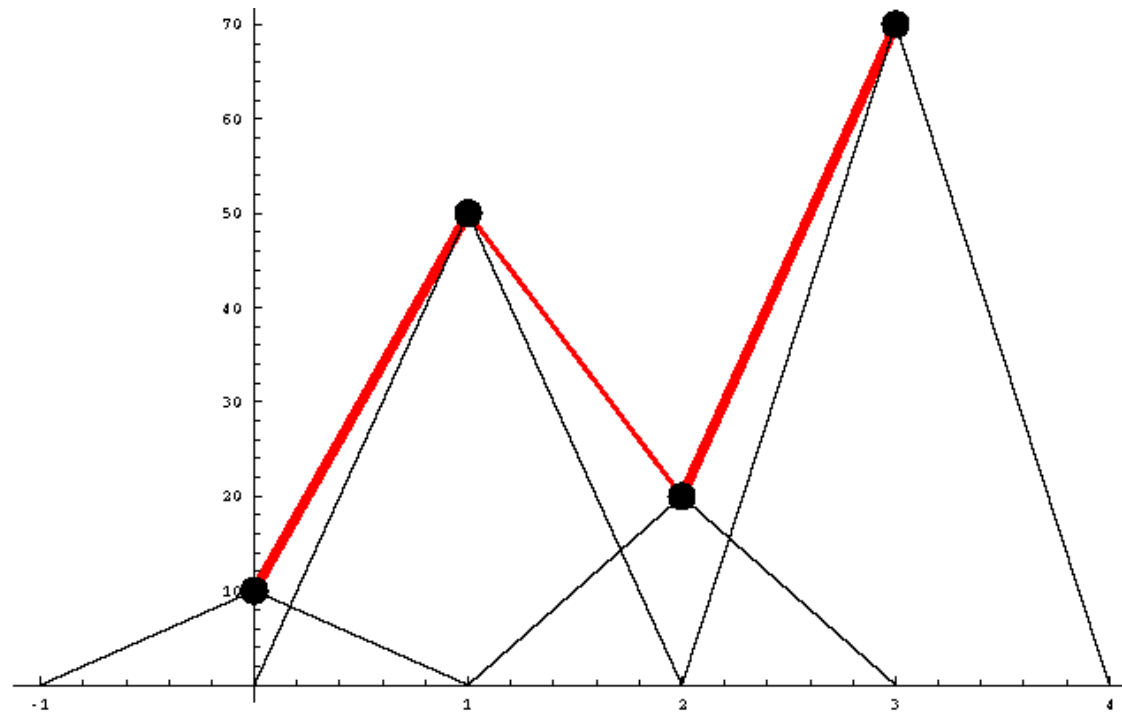


$\beta^4$



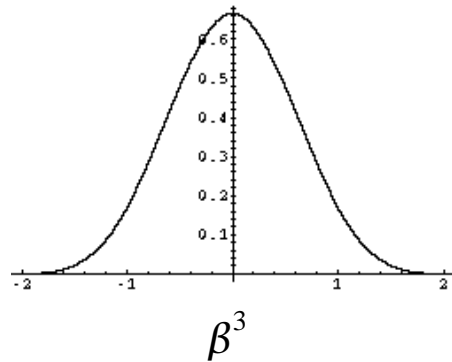
$\beta^\infty = \text{Gauss}$

# B-Splines: Überlagerung der Basisfunktionen 1. Grades



# B-Splines: Funktion im Zeitbereich

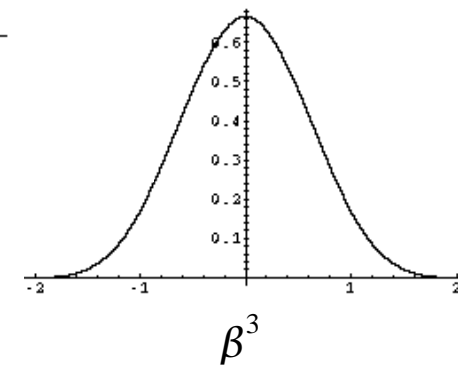
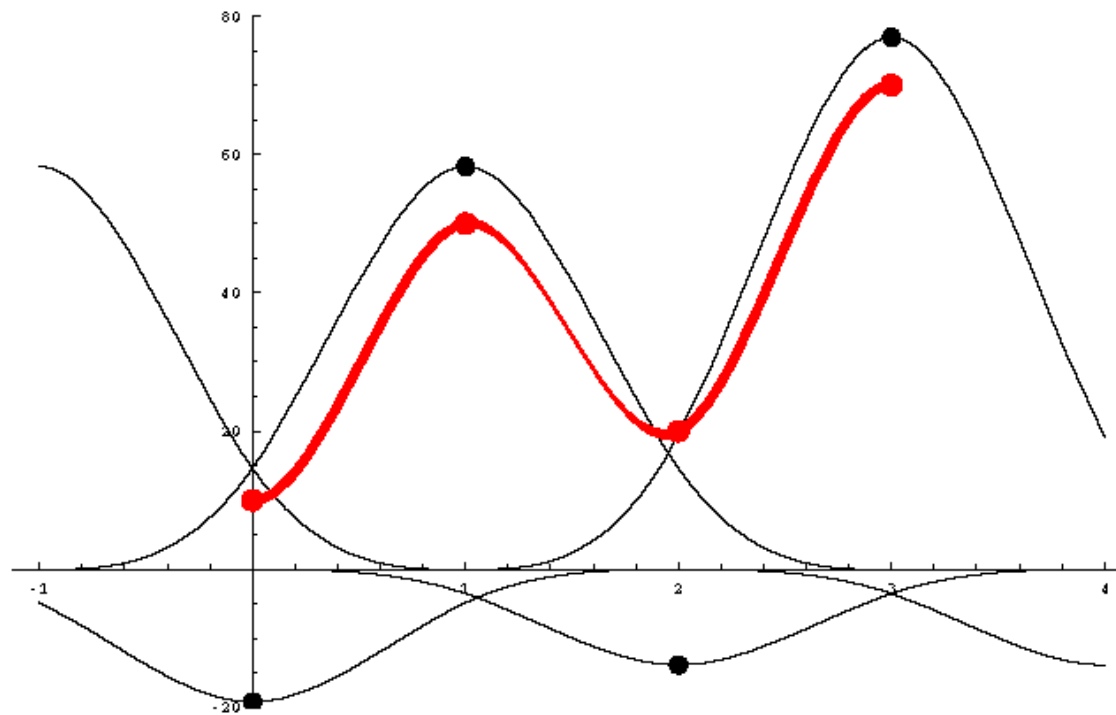
Die Faltung abschnittsweise definierter Funktionen ergibt wiederum abschnittsweise definierte Funktionen



$$\beta^3(x) = \begin{cases} \frac{2}{3} - x^2 + \frac{|x|^3}{2} & \text{für } 0 \leq |x| < 1 \\ \frac{(2-|x|)^3}{6} & \text{für } 1 \leq |x| < 2 \\ 0 & \text{sonst} \end{cases}$$
$$\beta^3(-1) = \frac{1}{6}$$
$$\beta^3(0) = \frac{4}{6}$$
$$\beta^3(1) = \frac{1}{6}$$

# Splines: Überlagerung der gewichteten Basisfunktionen

Im kubischen Fall setzt sich das Signal aus mindestens 3 Basisfunktionen additiv zusammen



---

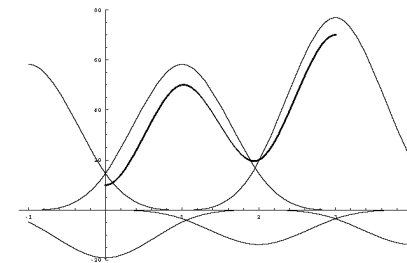
# Splines: Gewinnung der Spline-Koeffizienten

An den Interpolations-Stützstellen soll das Signal unverändert bleiben, d.h. alle drei beteiligten Spline-Basisfunktionen müssen sich zum ursprünglichen Wert addieren.

$$\beta^3(-1) = \frac{1}{6} \quad \beta^3(0) = \frac{4}{6} \quad \beta^3(1) = \frac{1}{6}$$

$$y(2) = g(1) \cdot \beta^3(-1) + g(2) \cdot \beta^3(0) + g(3) \cdot \beta^3(1)$$

Anmerkung: Da jeder Koeffizient mehrere Stützstellen überdeckt, muß auch jeder Koeffizient aus allen Signalwerten berechnet werden.



# Splines: Matrixform

Matrixansatz:

$$\begin{bmatrix} \frac{4}{6} & \frac{2}{6} & 0 & 0 \\ \frac{1}{6} & \frac{4}{6} & \frac{1}{6} & 0 \\ 0 & \frac{1}{6} & \frac{4}{6} & \frac{1}{6} \\ 0 & 0 & \frac{2}{6} & \frac{4}{6} \end{bmatrix} \cdot \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

Aufgabe: Gleichungssystem lösen

Da die Matrix Toeplitz-Form besitzt, kann ein schneller Algorithmus mit 5 Multiplikationen und 3 Additionen pro Pixel verwendet werden

---

# Splines: IIR-Filter

Andere Möglichkeit:

Darstellung der B-Spline Funktion im Frequenzraum (z-Transformierte)

$$B_1^3(z) = \frac{z + 4 + z^{-1}}{6}$$

$$B_1^3(z)^{-1} = \frac{6}{z + 4 + z^{-1}}$$

Die Transformation kann auch durch obige Funktion erfolgen.

Pole:

$$B_1^3(z)^{-1} = 0 \rightarrow z_{1,2} = -2 + \sqrt{3}$$

Das Ergebnis ist ein IIR-Filter.

Dieser benötigt 2 Additionen und 2 Multiplikationen pro Pixel.

---

# Besonderheiten der Spline-Transformation

Das Signal liegt in quasi-kontinuierlicher Darstellung vor:

- Analogfilter können nach entsprechender Transformation genutzt werden
  - Glättungsoperationen
  - Tiefpässe
  - Verwaschungsfilter
  - Schärfungsfilter
  - Differentiation und Integration
  - Bandpässe, z.B. für Subband-Coding
- Anwendung mehrerer Filter ohne zusätzlichen Transformationsaufwand
- Aber: Splinegrad kann sich durch Filteroperationen verändern (z.B. Tiefpaß)