# 5.3. Anwendungen

## 5.3.1. Fourierentwicklung:

Stückweise stetige, $2\pi$-periodische (in $[-\pi,\pi]$) Funktion f(x) lässt sich als Fourier-Reihe darstellen (vgl. Taylor/Potenz-Reihe):

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \left( a_k \cos(kx) + b_k \sin(kx) \right)$$

$$\left\{ \equiv \sum_{k=-\infty}^{k=\infty} c_k e^{ikx} = \sum_{k=-\infty}^{k=\infty} c_k \left( e^{ix} \right)^k \right\}$$

$a_k$ und $b_k$ heißen Fourier-Koeffizienten von f(x) und berechnen sich (wegen der Orhogonalität der Funktionen cos(kx) und sin(kx)) aus

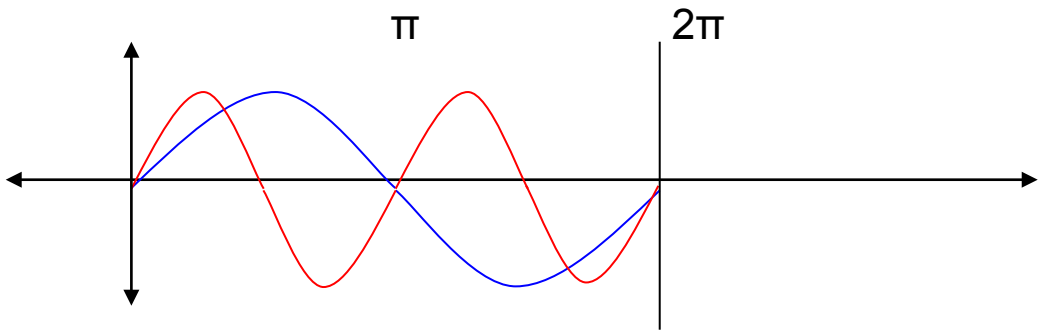
$$a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(kx) dx, \quad b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(kx) dx$$

Sie geben die Größe der Anteile von Vielfachen der Grundfrequenz (1/ $2\pi$) an, aus denen sich f zusammensetzt

**Wellenzahl: k**
**Periode: $2\pi/k$**
**Frequenz: $k/(2\pi)$**

**Grundfrequenz für k=1, Periode $2\pi$, Frequenz $1/(2\pi)$**

**$(a_k,b_k)$ messen Anteil von f zur Frequenz $k / (2\pi)$,**
(bzw. Periode $2\pi/k$, oder Wellenzahl k).




**Schwingung mit Wellenzahl k=1 und k=2**

**Wellenzahl k=2 entspricht Periode π, bzw. Frequenz 1/π**

*cos(kx), sin(kx) bilden Orthonormalsystem (Basis)!*
Fourierreihe entspricht Taylorreihe
Für periodische/auf Intervall definierte Funktion

## Das trigonometrische Polynom

$$f_n(x) = \frac{a_0}{2} + \sum_{k=1}^{n} \left( a_k \cos(kx) + b_k \sin(kx) \right)$$

stellt die optimale Approximation an die Funktion f dar aus dem Vektorraum der trigonometrischen Polynome vom Grad n:

$$\| f_n(x) - f(x) \|_2^2 = \int_{-\pi}^{\pi} \left( f_n(x) - f(x) \right)^2 dx$$ ist minimal.

# Näherungsweise Berechnung der Fourierkoeffizienten aus dem Integral mittels Trapezregel und äquidistanten Stützstellen

$$x_0 = -\pi, \qquad x_j = -\pi + j\frac{2\pi}{n}, \qquad x_n = \pi.$$

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(kx)dx \approx$$

$$\approx \frac{1}{\pi}\cdot\frac{2\pi}{n}\cdot\left(\frac{1}{2}f(-\pi)\cos(-k\pi) + f(x_1)\cos(kx_1) + \ldots\right.$$

$$\left.\ldots + f(x_{n-1})\cos(kx_{n-1}) + \frac{1}{2}f(\pi)\cos(k\pi)\right) =$$

$$= \frac{2}{n}\sum_{j=0}^{n-1} f(x_j)\cos\left(k(-\pi + j\frac{2\pi}{n})\right) =$$

$$= \frac{2}{n}\cos(k\pi)\sum_{j=0}^{n-1} f(x_j)\cos\left(\frac{2\pi kj}{n}\right) + \frac{2}{n}\sin(k\pi)\sum_{j=0}^{n-1} f(x_j)\sin\left(\frac{2\pi kj}{n}\right)$$

Daher ergeben sich die $a_k$ näherungsweise aus den Real- und Imginärteilen von

$$\sum_{j=0}^{n-1} f(x_j)\exp\left(\frac{2i\pi kj}{n}\right) = \sum_{j=0}^{n-1} f(x_j)\cos\left(\frac{2\pi kj}{n}\right) + i\cdot\sum_{j=0}^{n-1} f(x_j)\sin\left(\frac{2\pi kj}{n}\right)$$

Diese Werte erhält man aus der IDFT angewendet auf die Funktionswerte an den Stützstellen.

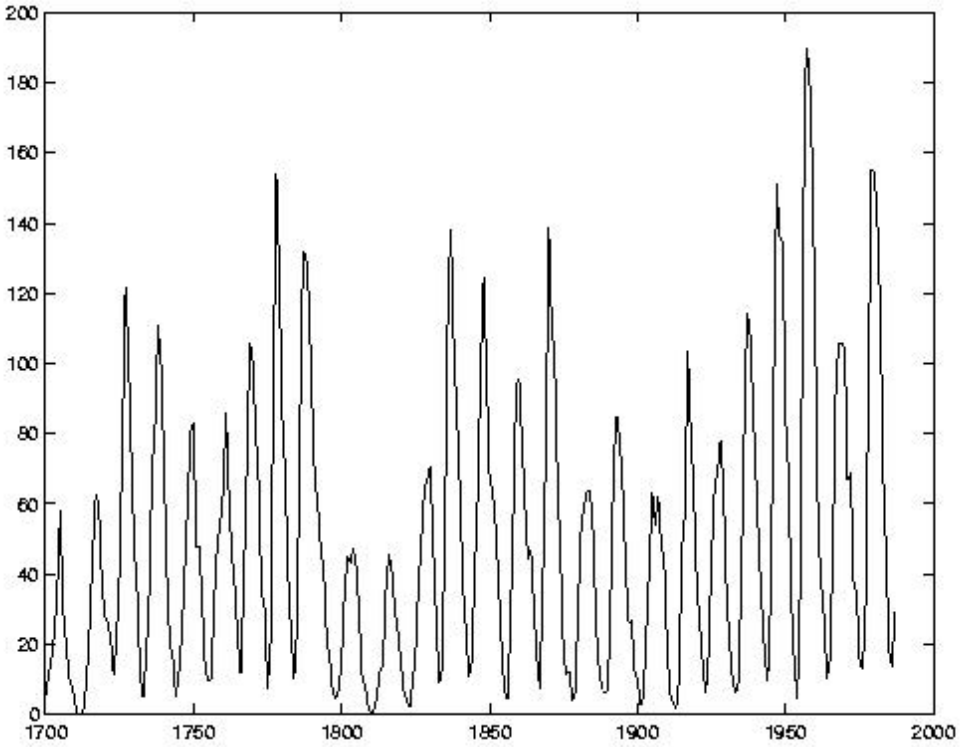
Genauso lassen sich die $b_k$ annähern.

*Also können mittels DFT auf den Vektor der Funktionswerte die Frequenzanteile von f näherungsweise bestimmen werden*.

DFT transformiert Funktionswerte in Koeffizienten

# 5.3.2. MATLAB-Analyse der Sonnenaktivität:

Sonnenflecken treten alle 11 Jahre verstärkt auf.
Seit 1700 wird die jährliche Sonnenfleckenaktivität beschrieben durch die sog. Wolfer-Zahl (Größe und Anzahl der Flecken) (Rudolf Wolf ca. 1850).




**Wolferzahl für die Jahre 1700 bis 2000**

Wolferzahlen in Vektor v als Funktionswerte einer unbekannten periodischen Funktion g mit den Jahreszahlen als Stützstellen.

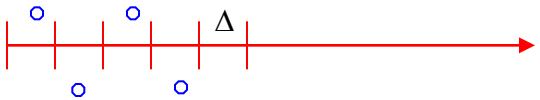
Gesamtbeobachtungszeitraum T besteht aus 300 Jahren.

Ersetze daher Intervall $[0,2\pi]$ durch das Intervall $[0,T]$ durch den Übergang von

**$\exp(ikx) \rightarrow \exp(ikx*2\pi/300)$**

Zeitintervall $\Delta = 1$ Jahr, in dem eine Wolferzahl bestimmt wurde;
$N = T / \Delta$ ist Anzahl der Beobachtungsintervalle =
= Länge des Vektors v
= Anzahl der Stützstellen;

Wellenzahl wieder k , Periode T/k = 300J./k, Frequenz k/T;
Grundfrequenz also 1/T.




Es können nur periodische Vorgänge erfasst werden mit einer Periode > 2Δ = 2 Jahre, da bei kleineren Perioden die ‚Schwingung' feiner als die feinste Unterteilung ( 1 Jahr) wäre, und daher als solche nicht erkennbar ist.

Daher ist die größte beobachtbare Wellenzahl k gleich N/2=150, entspricht der Frequenz (N/2)(1/T)=1/(2Δ), der sog. Nyquist-Frequenz

Berechne y = FFT(v)
y(0) ist die Gesamtsumme aller Wolferzahlen und wird gleich 0 gesetzt (enthält keine Angabe über Periodizität).

Die anderen Koeffizienten von y enthalten die Größe der verschiedenen Frequenzanteile von g (aber als komplexe Zahlen).

Berechne $p(k) = |y(k)|^2$ *für* $k = 1,..., N/2$

also untersuche nur Frequenzanteile bis zur Nyquist-Wellenzahl k = (N/2).

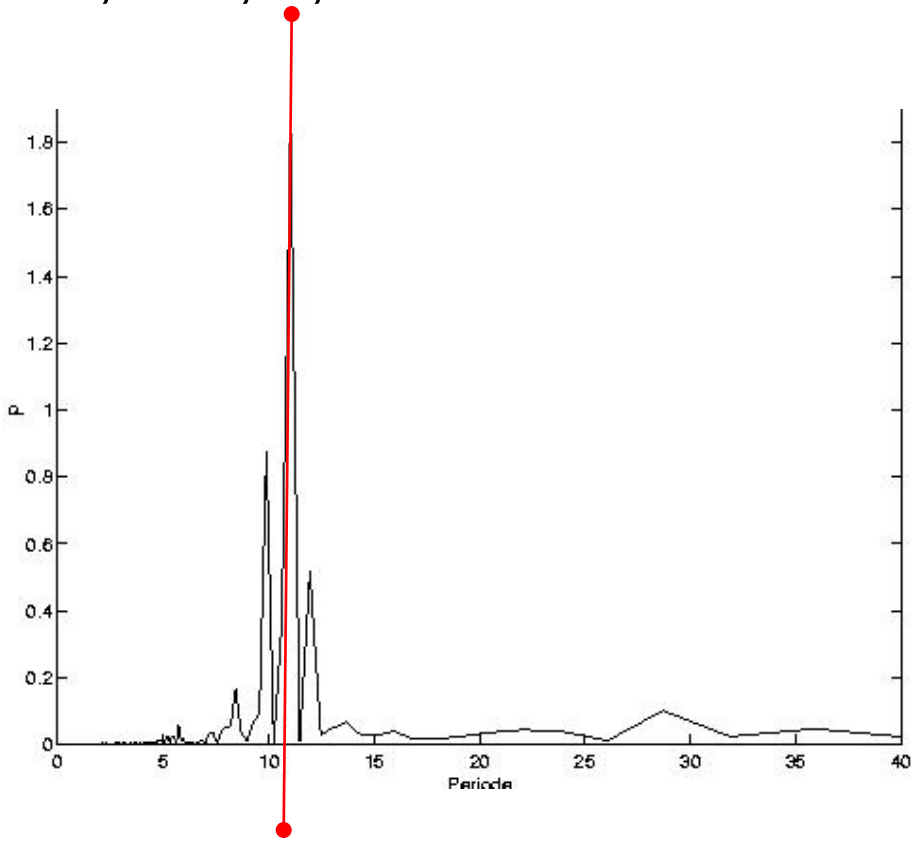
p(k) misst die Größe der k-ten Vielfachen der Grundfrequenz, also der Frequenz

$$f(k) = k / T = k / (\Delta N) \quad \text{für } k=1,2,...,N/2$$

Anders ausgedrückt bestimmt p(k) das Gewicht der Periode

$$1 / f(k) = T / k$$ in der Funktion f.

Wir tragen nun den Vektor p auf gegen die dazugehörigen Perioden T/k, k=1,...,N/2




Betragsquadrate der Fourierkoeffizienten, aufgetragen über die dazugehörigen Perioden.
Wir können den Zyklus von 11 Jahren direkt ablesen.

**Allgemeines Problem:**
In welcher Form stellt man numerische Daten dar?

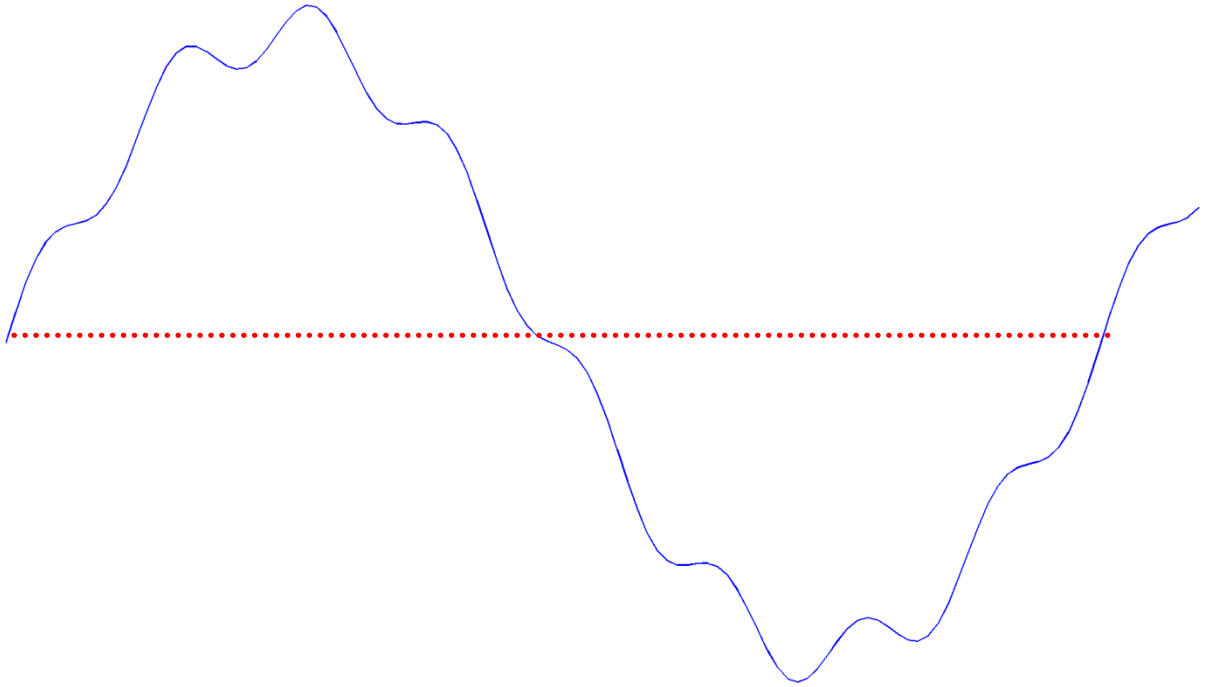
Normalerweise aus Messwerten: Samples $f(x_j)$
Dies entspricht einer Darstellung im Ortsraum.

Nachteil: Viele Daten, schlecht komprimierbar, keine Analyse

Besser: Übergang zu einer Darstellung bzgl. geeigneter Basisfunktionen (z.B. $x^j$, exp(jx), cos(jx),...)

Dann reichen ev. wenige Koeffizienten aus, um das Signal (fast) vollständig zu beschreiben. Außerdem kann man aus den Koeffizienten Schlüsse über das Verhalten der Funktion ziehen (Frequenzanteile)

Beispiel: f(x) = 0.5*sin(x) + 0.1 * sin(10x)



Zwei Koeffizienten reichen aus zur Beschreibung

Die Koeffizienten bezeichnet man auch als Beschreibung der Funktion im Phasenraum (Frequenzraum)

# 5.3.3. JPEG:

Bilder werden üblicherweise als Matrix gespeichert, deren Einträge pixelweise den Grauwert des entsprechend nummerierten Pixels enthalten, bzw. bei Farbbildern den Anteil einer der drei Farbkomponenten Rot, Grün oder Blau (= RGB).

*RGB wird meist in YUV-Form umgewandelt, bei der die Matrix Y die Helligkeit beschreibt, und U und V den Farbton (U,V lassen sich stärker komprimieren).*
JPEG zerlegt das Bild in 8 x 8 – Pixel große Blöcke, die zunächst getrennt bearbeitet werden. Auf jede dieser Teilmatrizen wird eine zweidimensionale Cosinus-Transformation angewandt: FCT

Grund, warum nicht DFT: *Bilddaten sind reell, und das Bild ist nicht periodisch*!

Aus Anwendung der FCT erhält man für die Teilmatrizen wieder die Frequenzanteil-koeffizienten $a_k$ und $b_k$ in Teilmatrizen.

*Die Teilmatrix mit den Frequenzanteilen wird nun quantisiert,d.h. die darin stehenden reellen Zahlen werden bestimmten Zahlenintervallen zugeordnet und Intervallweise jeweils durch eine Zahl angenähert, die für ein ganzes Intervall gilt.*

Dadurch werden auch automatisch alle kleinen Komponenten, die in dem ersten Intervall mit den kleinsten Werten liegen, durch Null ersetzt.
Dies erzeugt einen Qualitätsverlust!

*Die entstandene Gesamtmatrix aus den diskreten Zahlenwerten wird codiert* – ***Huffman-Codierung***

Nachteile:

Fourier-Methoden haben Schwierigkeiten mit Kanten in Bildern

Kanten ←→ Unstetigkeiten

Durch stetige Funktionen cos(kx), sin(kx) sind Kanten schlecht darstellbar!

Kosten der DCT: O(n log(n)) bei n Pixel. Zu teuer.

Daher DCT auf 8x8-Blöcke.

**JPEG2000:**

Anstatt Cosinus-Transformation auf 8 x 8 wendet man eine Wavelet-Transformation auf größere Teilmatrizen an.

Wavelet-Ansatzfunktionen haben keine Problem mit Kanten, vgl. B-Splines. Genauere Untersuchung folgt später.

Kosten für Wavelet-Transformation O(n)

**Beispiel: MATLAB Gibbs-Phänomen** (recht.m)

Fourierkoeffizienten für Rechtecksignal;
Überschwingen an Kante unvermeidbar bei Darstellung durch endliches trigonometrisches Polynom!

# 5.3.4. Lineare Filter:

Betrachte Vektor v, dessen Komponenten wieder diskrete Werte einer Funktion oder eines Bildes darstellen (Sampling).

Zunächst 1-dimensional.
Wir *filtern* diesen Vektor, indem wir jede Komponente ersetzen durch eine gewichtete Kombination der Nachbarkomponenten.

So entspricht die *Maske*:

$\frac{1}{4}[1 \quad 2 \quad 1]$ der Operation $v_j = \frac{v_{j-1} + 2v_j + v_{j+1}}{4}$,

bzw. der Matrixmultiplikation

$$v \rightarrow \frac{1}{4}\begin{pmatrix} 2 & 1 & & & \\ 1 & 2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 2 & 1 \\ & & & 1 & 2 \end{pmatrix} \cdot v$$

(Vorsicht am Rand!)

Im zwei-dim. entspricht dies z.B. der Maske

$$\frac{1}{8}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \rightarrow \quad v_{i,j} = \frac{v_{i,j-1} + v_{i-1,j} + 4v_{i,j} + v_{i,j+1} + v_{i+1,j}}{8},$$

Genauso Gaussfilter: $\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$

Dies entspricht glättenden Filtern, die zu einem weicheren Bild führen:
Mittelwertfilter, Glätter, Weichzeichner, oder Tiefpassfilter zur Abschwächung von Rauschen, bzw. hochfrequenten Anteilen.
*Hochfrequente Störungen, die einzelne Komponente verändern, werden durch die Mittelwertbildung verringert!*

Entsprechend kann man Hochpassfilter definieren, die die Unterschiede hervorheben, das Bild härter machen und niederfrequente (glatte) Anteile abschwächen (Differenzfilter, Scharfzeichner).

z.B. Laplacefilter:
(Sobelfilter)

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

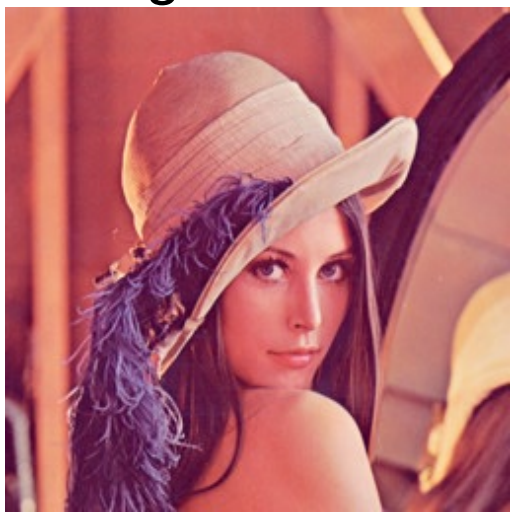
Bisher entsprechen die Methoden einer *Filterung im Ortsraum*. Im Gegensatz dazu kann man auch *Filter im Phasenraum* (Frequenz~) zum Entfernen von Rauschen (Noise) verwenden:

v → DFT(v) → Filter → IDFT(v)

Also Transformation in (Fourier-)Koeffizientenraum, dann Filtern der Koeffizienten, dann Rücktransformation (vgl. MP3)

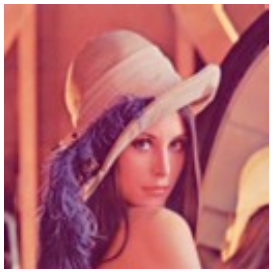
# Filtern von Lena:

Original



Mittelwertfilter

Reduktion



Differenzfilter

Reduktion

# 5.3.5. Wavelets:

Betrachte **Kombination** von Tiefpassfilter und Hochpassfilter im 1-Dimensionalen zu Masken

$$\begin{bmatrix}1 & 1\end{bmatrix}/2 \qquad \text{und} \qquad \begin{bmatrix}1 & -1\end{bmatrix}/2$$

Ersetze den ursprünglichen Vektor verlustfrei durch tief-, bzw. hochpass-gefilterte Vektoren halber Länge (down sampling):

$$\begin{pmatrix} v_t \\ \hline v_h \end{pmatrix} = \frac{1}{2} \left( \begin{array}{cccccc} 1 & 1 & & & & \\ & & 1 & 1 & & \\ & & & & \ddots & \ddots \\ \hline 1 & -1 & & & & \\ & & 1 & -1 & & \\ & & & & \ddots & \ddots \end{array} \right) \cdot v$$

oder

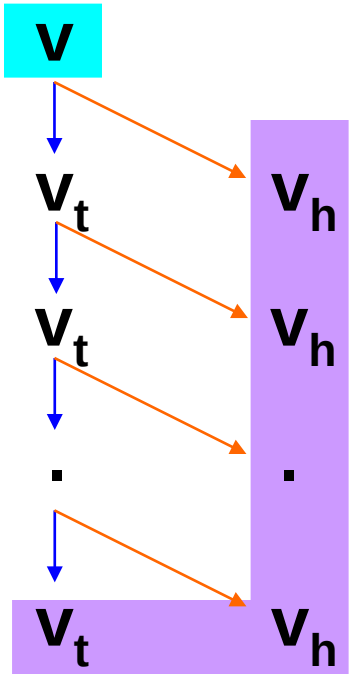
$$\frac{1}{2}\begin{pmatrix} \boxed{\begin{matrix}1 & 1\\ 1 & -1\end{matrix}} & & & & \\ & \boxed{\begin{matrix}1 & 1\\ 1 & -1\end{matrix}} & \ddots & & \\ & & \ddots & \ddots & \\ & & & \ddots & \\ & & & & \boxed{\begin{matrix}1 & 1\\ 1 & -1\end{matrix}} \end{pmatrix} \cdot v$$

(nur anders sortiert)

Der Differenzanteil $v_h$ ist in der Regel klein und wird nicht weiterbearbeitet!

Der Mittelwertanteil (tiefpassgefiltert) $v_t$ wird nach demselben Schema weiteraufgespalten wiederum in Tief/Hochpassanteil durch dieselben Masken.

**Insgesamt:**




**Level 0 , Länge n**

**Level 1 , n/2**

**Level 2 , n/4**

**.**

**Level p , 1**

Ersetze nun den Ausgangsvektor $v \in \mathbf{R}^n$ durch den letzten Mittelwertanteil auf Level p und sämtliche Differenzanteile $v_h$ , das sind auch genau n Zahlen.
( $v_h$ und $v_t$ entsprechen quasi den Fourierkoeffizienten)

# Vorteile:

Gesamtkosten der Transformation O(n)
Differenz-Anteile meist klein → gut komprimierbar

Differenzanteil auf Level k enthält Information über das Verhalten von v auf diesem Level, bzw. in dieser Auflösung
→

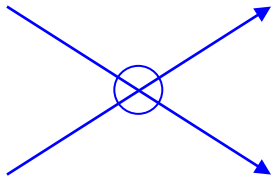
Multiskalenanalyse, Zerlegung des Vektors in verschiedene Frequenzbereiche = Skalen
Entspricht in etwa der Fourier-Analyse

Filter muss leicht umkehrbar (invertierbar) sein!
Rekonstruktionsbedingung! (vgl. DFT und IDFT )

# Algorithmische Ähnlichkeit zur Fourier-Transformation am Beispiel des Haar-Filters mit den Masken

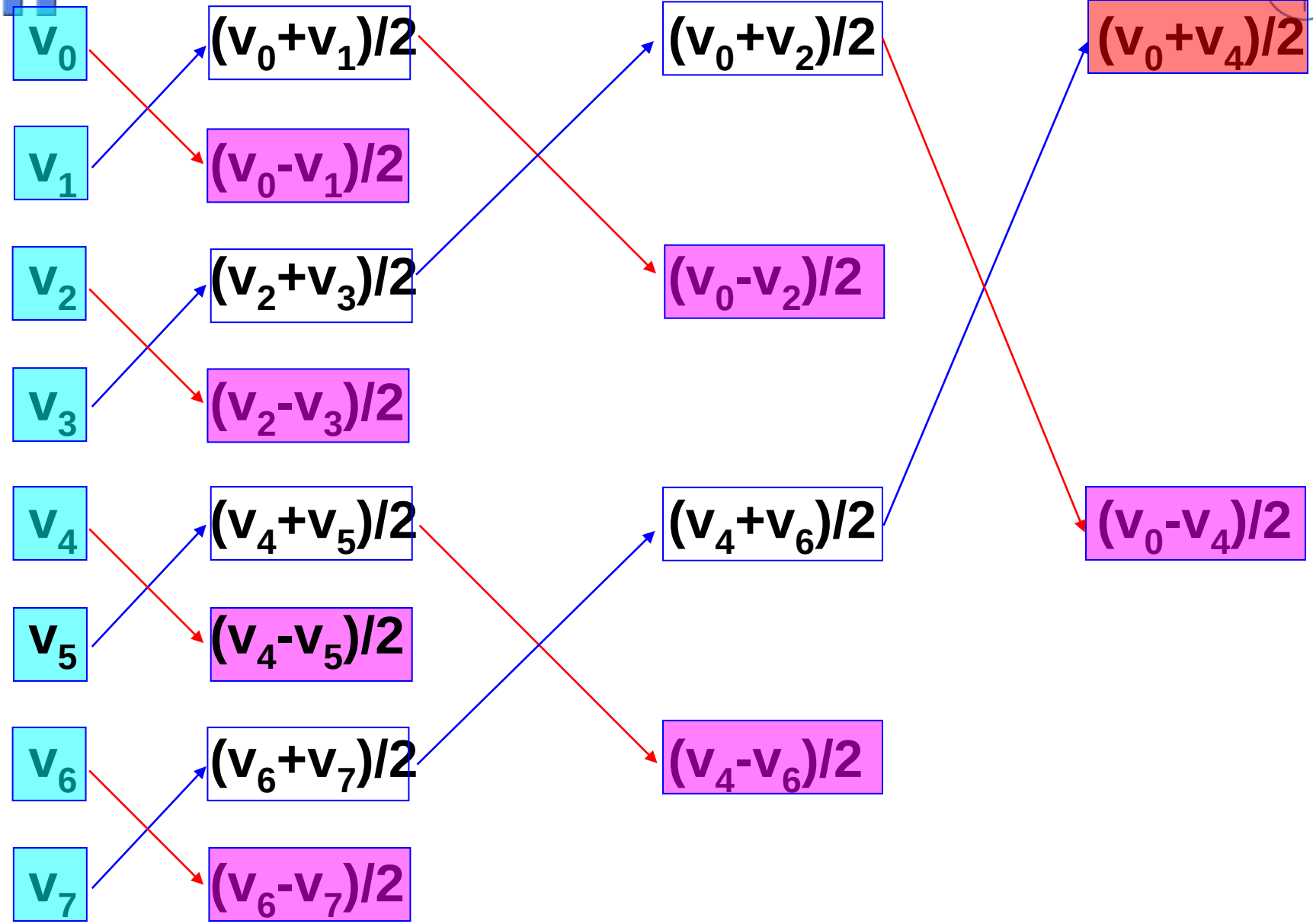
$$\frac{1}{2}[1 \quad 1] \qquad \text{und} \qquad \frac{1}{2}[1 \quad -1] \;:$$

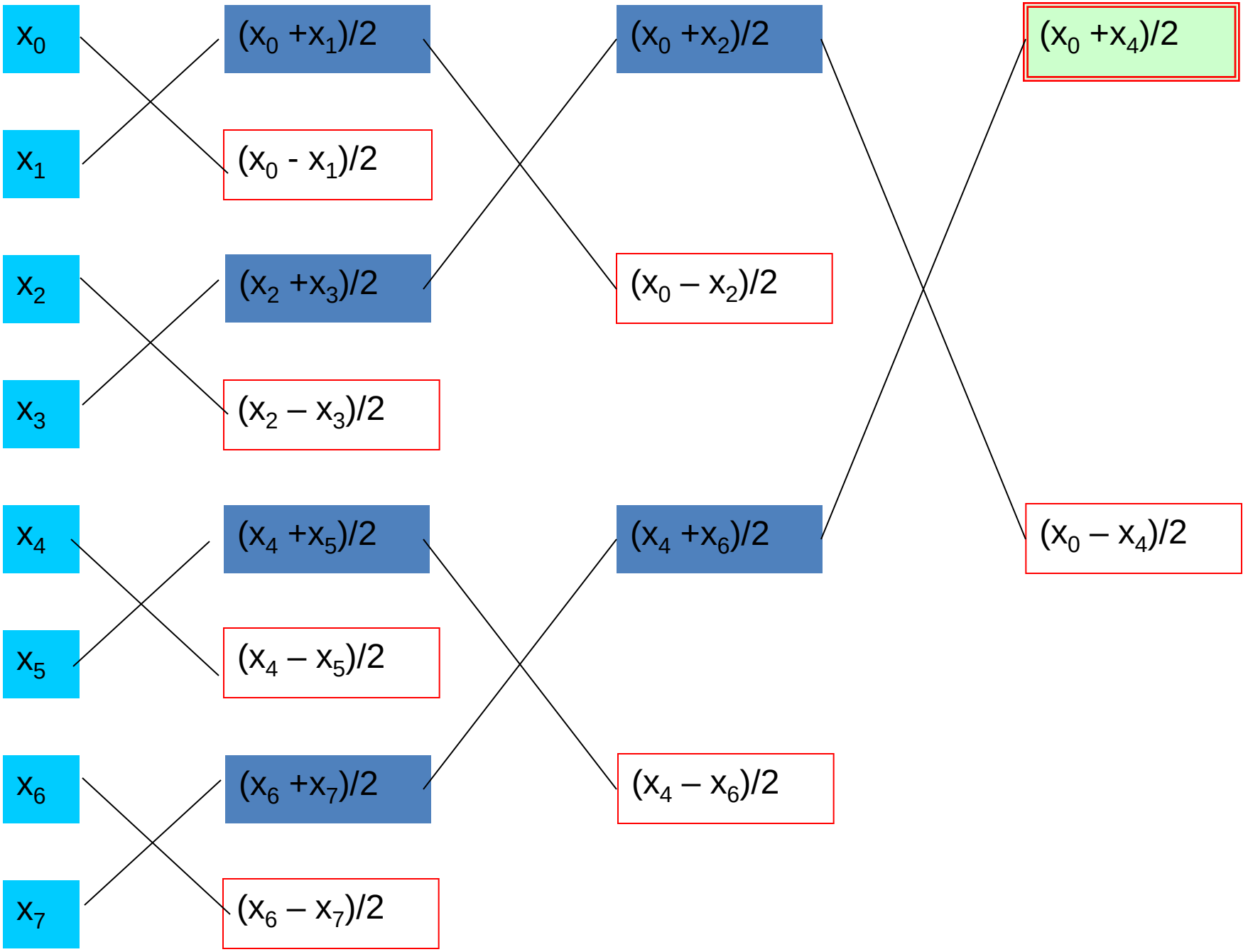
**Butterfly**



**u** → **(u+v)/2**

**v** → **(u-v)/2**


$v_0$
$v_1$
$v_2$
$v_3$
$v_4$
$v_5$
$v_6$
$v_7$
$(v_0+v_1)/2$
$(v_0-v_1)/2$
$(v_2+v_3)/2$
$(v_2-v_3)/2$
$(v_4+v_5)/2$
$(v_4-v_5)/2$
$(v_6+v_7)/2$
$(v_6-v_7)/2$
$(v_0+v_2)/2$
$(v_0-v_2)/2$
$(v_4+v_6)/2$
$(v_4-v_6)/2$
$(v_0+v_4)/2$
$(v_0-v_4)/2$


$x_0$
$x_1$
$x_2$
$x_3$
$x_4$
$x_5$
$x_6$
$x_7$
$(x_0 + x_1)/2$
$(x_0 - x_1)/2$
$(x_2 + x_3)/2$
$(x_2 - x_3)/2$
$(x_4 + x_5)/2$
$(x_4 - x_5)/2$
$(x_6 + x_7)/2$
$(x_6 - x_7)/2$
$(x_0 + x_2)/2$
$(x_0 - x_2)/2$
$(x_4 + x_6)/2$
$(x_4 - x_6)/2$
$(x_0 + x_4)/2$
$(x_0 - x_4)/2$

v wird dabei transformiert in
alle Differenzanteile und
den letzten Mittelwert.

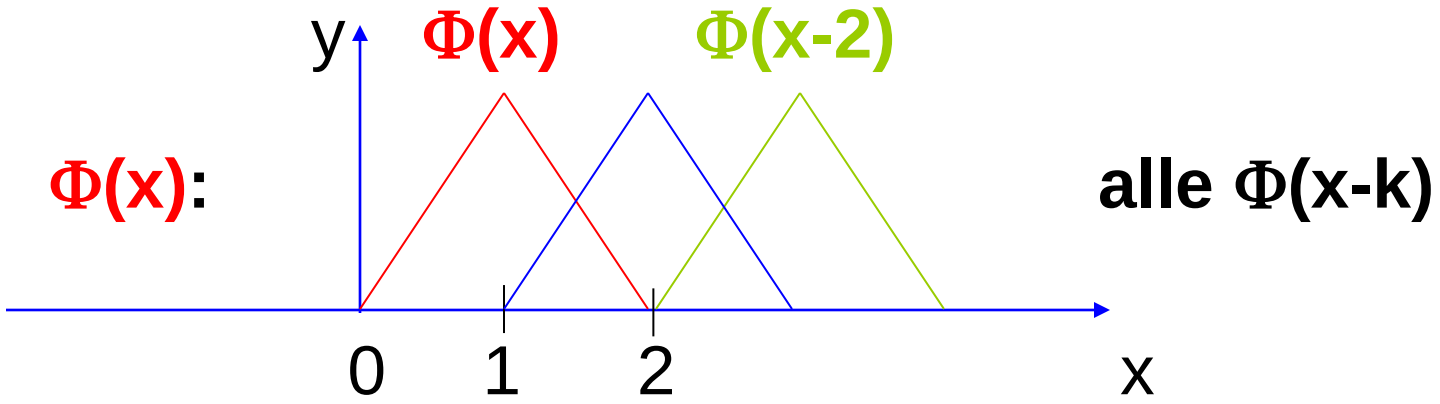
*Unterschied zur DFT:*

*keine Sortierphase,*
*einfacher Butterfly,*
*nur die Mittelwertanteile werden weiter bearbeitet*

→ *O(n)*

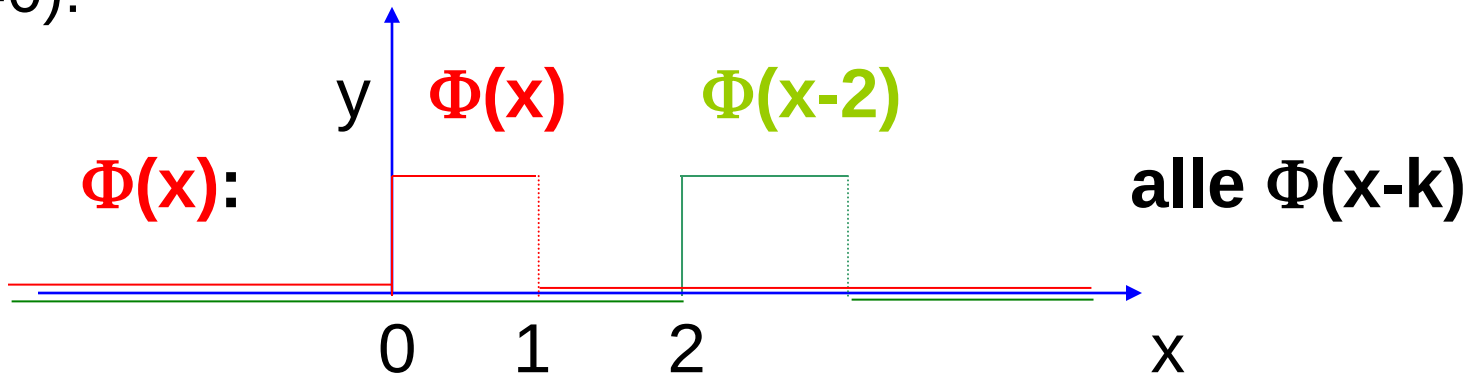
# Funktionsansatz:

## Erinnerung: lineare B-Splines (Hut-Funktion)



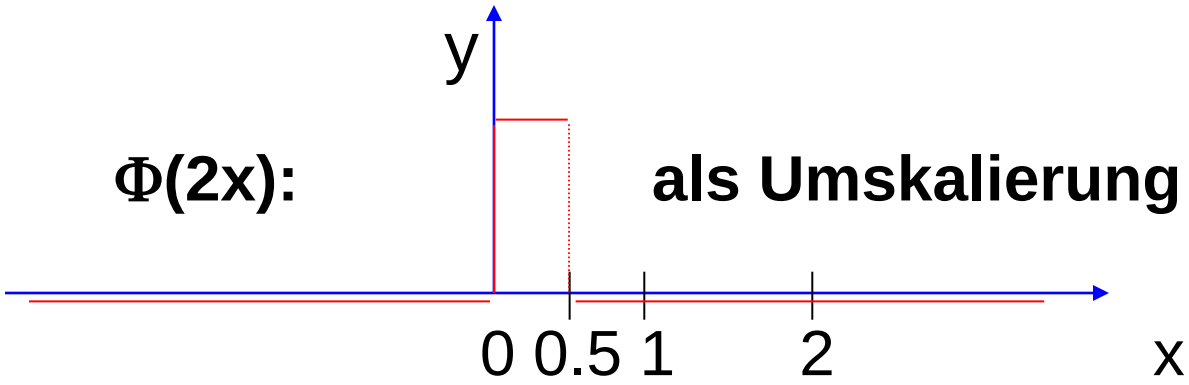

Äquidistante Stützstellen …,-1,0,1,2,…

Zur Vereinfachung betrachten wir den konstanten B-Spline (k=0):



Φ(x) heißt Skalierungsfunktion und liefert Basis Φ(x-k), k
Weiterhin erfüllt Φ(x) die Gleichung

$$\Phi(x) = ( 1 \cdot \Phi(2x) + 1 \cdot \Phi(2x-1) )$$




Genauso liefern Φ(2x-k) (feinere) Ansatzfunktionen

**z.B. 1= Φ(0.25) =**

**= ( Φ(2*0.25) + Φ(2*0.25-1) ) = Φ(0.5) = 1**

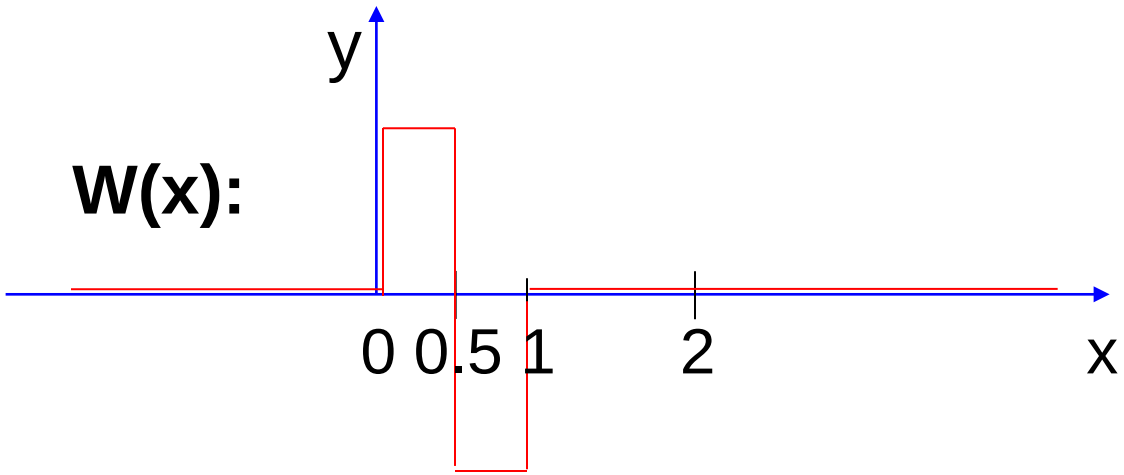
Filter-Maske [ 1 1 ] , denn

**1*Φ(2x) + 1*Φ(2x-1)**

zu Tiefpass-Filter (Mittelwertfilter)

Notwendig ist zweiter Hochpass-Filter (Differenz-Filter).
Definiere dazu zur Skalierungsfunktion $\Phi(x)$ die eigentliche Waveletfunktion

**W(x) = 1*Φ(2x) - 1*Φ(2x-1) ,  Maske [ 1  -1 ]**




Wichtig: Orthogonalität der Ansatzfunktionen!

$$\int \Phi(x-k)\cdot W(x-j)dx = 0$$

$\Phi(2x-k)$ sind Basis zu feinerer Diskretisierung und höherer Auflösung (Genauigkeit).

Mit den obigen Beziehungen kann $\Phi(2x)$ eindeutig durch $\Phi(x)$ und $W(x)$ dargestellt werden:

$$\Phi(x) = \Phi(2x) + \Phi(2x-1)$$
$$W(x) = \Phi(2x) - \Phi(2x-1)$$

oder umgekehrt

$$\Phi(2x) = \left(\Phi(x) + W(x)\right)/2$$
$$\Phi(2x-1) = \left(\Phi(x) - W(x)\right)/2$$

**Gegeben:** $f(x) = \sum a_k \cdot \Phi(2x-k)$

Nun können wir den Übergang von Koeffizienten zur Basis $\Phi(2x-k)$ zu Koeffizienten bezgl. $\Phi(x-k)$, $W(x-k)$ beschreiben

$$a_{2k} \cdot \Phi(2x-2k) \rightarrow a_{2k} \cdot (\Phi(x-k) + W(x-k))/2$$
$$a_{2k+1} \cdot \Phi(2x-2k-1) \rightarrow a_{2k+1} \cdot (\Phi(x-k) - W(x-k))/2$$

$$f(x) = \sum a_k^{(1)} \Phi(2x-k) =$$

$$= \sum a_{2k}^{(1)} \Phi(2x-2k) + \sum a_{2k+1}^{(1)} \Phi(2x-2k-1) =$$

$$= \sum \frac{a_{2k}^{(1)}}{2} \left(\Phi(x-k) + W(x-k)\right) +$$

$$+ \sum \frac{a_{2k+1}^{(1)}}{2} \left(\Phi(x-k) - W(x-k)\right) =$$

$$= \sum \frac{a_{2k}^{(1)} + a_{2k+1}^{(1)}}{2} \cdot \Phi(x-k) + \sum \frac{a_{2k}^{(1)} - a_{2k+1}^{(1)}}{2} \cdot W(x-k) =$$

$$= \sum a_k^{(0)} \cdot \Phi(x-k) + \sum b_k^{(0)} \cdot W(x-k) =$$

$$= f^{\,tiefpass}(x) + f^{\,hochpass}(x)$$

Anwendung der beiden Filtermasken **[ 1  1 ] /2** und **[ 1  -1 ] /2** auf die Koeffizienten $a_k$ liefert neue Koeffizienten zu hoch/tiefpass-gefilterten Teil-Funktionen.

Umgekehrt kann aus den beiden gefilterten Teilfunktionen die Gesamtfunktion f(x) wieder rekonstruiert werden mittels

$$f^{tiefpass}(x) + f^{hochpass}(x) =$$

$$\sum a_k^{(0)} \Phi(x-k) + b_k^{(0)} W(x-k) =$$

$$= \sum a_k^{(0)} (\Phi(2x-2k) + \Phi(2x-2k-1)) +$$

$$+ \sum b_k^{(0)} (\Phi(2x-2k) - \Phi(2x-2k-1)) =$$

$$= \sum (a_k^{(0)} + b_k^{(0)}) \cdot \Phi(2x-2k) +$$

$$+ \sum (a_k^{(0)} - b_k^{(0)}) \Phi(2x-2k-1) =$$

$$= \sum a_{2k}^{(1)} \Phi(2x-2k) + \sum a_{2k+1}^{(1)} \Phi(2x-2k-1) =$$

$$= \sum a_k^{(1)} \Phi(2x-k) = f(x)$$

Also Anwendung der Filter

**[ 1 1 ] und [ 1 -1 ]**

auf die Koeffizienten der hoch/tiefpassgefilterten Koeffizienten liefert die Koeffizienten zur feineren Diskretisierung.

Funktionenfamilien auf verschiedenen Skalen:

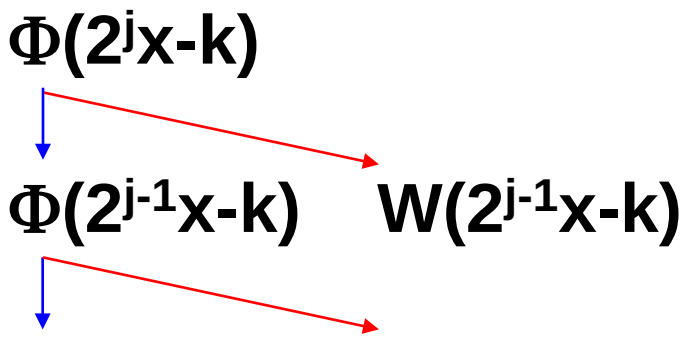
**$\Phi(x-k)$ , k = ..., -1,0,1,...**

**$\Phi(2x-k)$ , k = ..., -1,0,1,...**

**....**

**$\Phi(2^j x-k)$ , k = ..., -1,0,1,... , j = ...,-1,0,1,...**

# Basisumrechnung von f(x):




beschrieben durch die Transformation der Koeffizienten:

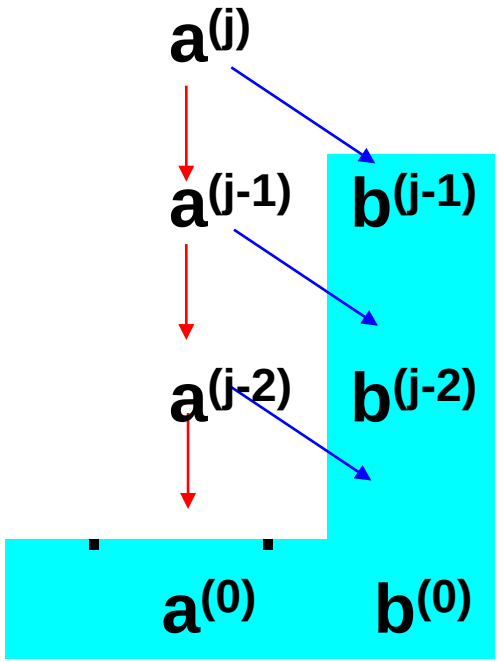
$$\begin{pmatrix} \vdots \\ \vdots \\ a_k^{(j-1)} \\ b_k^{(j-1)} \\ \vdots \\ \vdots \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & 1 & & & & \\ 1 & -1 & & & & \\ & & 1 & 1 & & \\ & & 1 & -1 & & \\ & & & & \ddots & \ddots \\ & & & & \ddots & \ddots \end{pmatrix} \cdot a^{(j)}$$

entsprechend Tief- und Hochpassfilter mit Haar-Filter.

Umkehrung mit

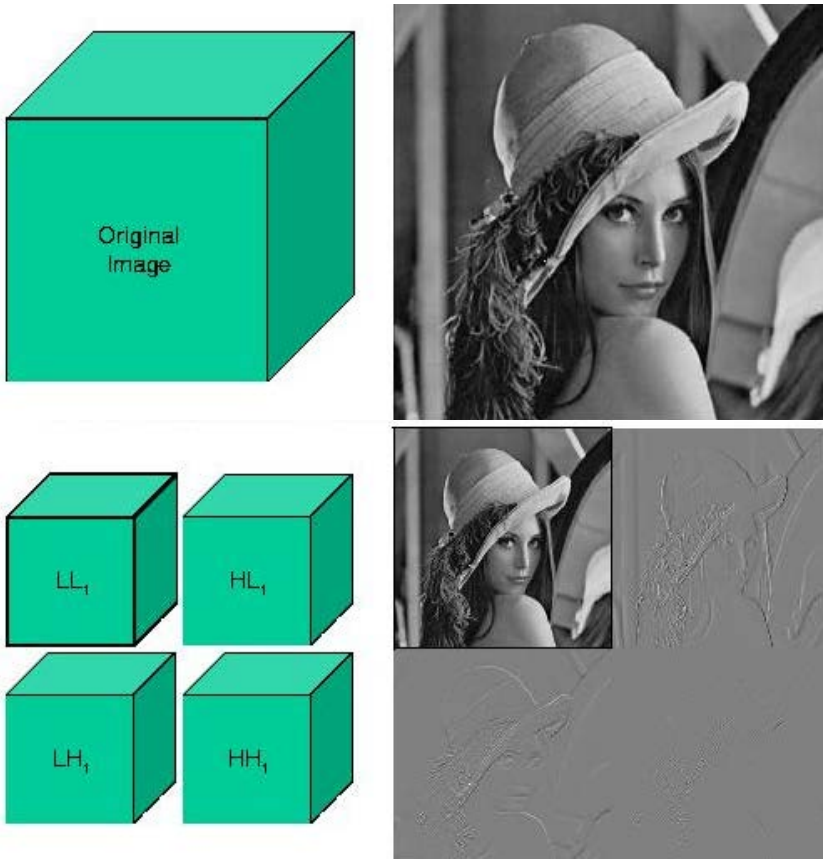
$$a^{(j)} = \begin{pmatrix} 1 & 1 & & & & \\ 1 & -1 & & & & \\ & & 1 & 1 & & \\ & & 1 & -1 & & \\ & & & & \ddots & \ddots \\ & & & & \ddots & \ddots \end{pmatrix} \cdot \begin{pmatrix} \vdots \\ \vdots \\ a_k^{(j-1)} \\ b_k^{(j-1)} \\ \vdots \\ \vdots \end{pmatrix}$$

**Rekursive Wiederholung:**



Wavelet als Verschmelzung der
klassischen Filter und
Fourieranalyse mit der Idee
lokaler Basisfunktionen wie bei den B-Splines.

Haar-Filter ←→ Haar-Wavelet


Original
Image
LL1
HL1
LH1
HH1
LL2
HL2
LH2
HH2
HL1
LH1
HH1

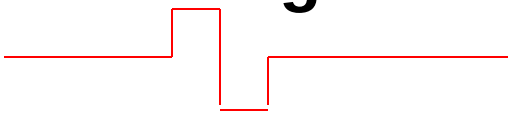
# Fourier-Analyse
## FFT

cos(kx), sin(kx)
Skalierung k
Alle gleichberechtigt
globaler Träger

keine Variations-möglichkeit
Kantenproblem
Volle Rekursion
Kosten O(n log(n))
Frequenz aus Fourierkoeffizienten

# Multiskalen-Analyse
## Wavelet

$W(2^j x-m)$, $\Phi(2^j x-m)$
Shift m, Skalierung $2^j$
Mittelwert/Differenz
lokaler Träger



Wahl von Approximations-güte und Diff'barkeit
Lokal, adaptiv
Rekursion nur im Mittelwert
Kosten O(n)
Frequenz aus Wavelekoeffizienten