Die Fourier-Transformation

INHALTSVERZEICHNIS

Einleitung

- Periodische Vorgänge spielen schon seit jeher in Natur und Technik eine zentrale Rolle
  z.B. Pulsschlag, Planetenbewegung, Takt- und Tonfrequenz in Musik

- Die Rechentechnik zur Analyse solcher Vorgänge lieferte JEAN BAPTISTE JOSEPH FOURIER (1768-1830)

- Die Hauptaussage FOURIER's war:
  „Jede periodische Funktion kann durch Summen von Sinus- und Kosinusschwingungen dargestellt werden."

Der rechentechnische Durchbruch gelang jedoch erst COOLEY und TUKEY in den 60er Jahren mit der computergestützten FOURIERanalyse.

Dadurch konnten digitale Signale verarbeitet werden. Sie konnten analysiert, gefiltert und geglättet werden.

Die FOURIERtransformation ist ein fundamentales Verfahren in der Signalverarbeitung, es ermöglicht die Überführung von Signalen der Darstellung (Zeitpunkt, Abtastwert) in Signale der Darstellung (Frequenzanteil, Amplitude und/oder Phase).

Viele Operationen (z.B. Filter) sind im Frequenzraum wesentlich leichter durchführbar.
Anschließend wird das Signal mit Hilfe der inversen FOURIERtransformation zurücktransformiert.
Meist ist sogar die Transformation, das Anwenden der Operation und die Rücktransformation schneller als das direkte Anwenden der Operation auf das entsprechende Signal.

Anwendung findet die FOURIERtransformation u.a. in der Untersuchung natürlicher Töne und Signale.
Andersherum ist es möglich, mit Hilfe der inversen Eigenschaft Signale zu erzeugen.

Grundidee einer Transformation

Eine Transformation ist ein nützliches Hilfsmittel um vielseitige Probleme zu lösen. Die Idee beruht auf der Überführung einer mathematischen Menge (Zahl, Vektor, Funktion,...) in eine andere Form, die vielleicht unvertraut ist, aber dafür sehr nützliche Eigenschaften hat.

Beispiel 1:

Multiplikation zweier römischer Zahlen

$\mathrm{XCVI} \cdot \mathrm{XII}$	-	Transformation in die arabische Schreibweise (Hintransformation)
$96 \cdot 12 = 1152$	-	Berechnung
$1152 \rightarrow \mathrm{MCLII}$	-	Rücktransformation wieder in die römische Schreibweise (Rücktransformation)

Beispiel 2:

Multiplikation zweier reeller Zahlen

Früher, als noch kein Taschenrechner bekannt war, wurde die Multiplikation mittels Umwandlung in den Logarithmus berechnet, denn so braucht man die Logarithmen nur noch zu addieren (Logarithmentafeln).

$$Z = 1{,}2345 \cdot 1{,}5432$$

Weg I (Schulmethode):

$$\begin{array}{r} \underline{12345 \cdot 15432} \\ 12345 \\ 61725 \\ 49380 \\ 37035 \\ \underline{24690} \\ \underline{190508040} \end{array}$$

$$\underline{\underline{Z = 1.90508040}}$$

Weg II (Logarithmentafel):

$$\log(Z) = \log(1.2345 \cdot 1.5432)$$

$$\log(Z) = \log(1.2345) + \log(1.5432) \quad \textit{/Nachschlagen (Hintransformation)}$$

$$\log(\mathrm{Z}) \approx 0.0915 + 0.1884$$

$$\log(\mathrm{Z}) \approx 0.2799 \quad \textit{/Nachschlagen (Rücktransformation)}$$

$$\underline{\underline{Z \approx 1.9050}}$$

Hier erkennt man auch gut, dass das Nachschlagen in einer Tabelle wesentlich schneller erfolgt, als die Berechnung des Produkts mit Hilfe der Schulmethode.

Das Samplingtheorem

- Das Abtasten einer Funktion wird Sampling genannt
- Nyquist und Shannon haben bereits in den 40er Jahren des 20. Jahrhunderts das Sampling-Theorem gezeigt:
  „Die Abtastfrequenz sollte größer dem doppelten der maximal auftretenden Frequenz sein“

$$f \geq 2 \cdot f_{\max} \qquad \begin{array}{ll} f & - Nyquistrate \\ f_{\max} & - Nyquistfrequenz \end{array}$$

- Die Nyquist-Rate beschreibt die theoretisch minimalste Sampling-rate, bei der das Eingangssignal noch vollständig dargestellt werden kann. Es ist also jene Samplingrate, bei der das ursprüngliche Signal verlässlich wiederhergestellt werden kann.

  Beispiel: menschliches Gehör: 20 Hz bis 22 kHz
  Audiosignale in CD-Qualität: 44 kHz

- Dabei muss man beachten, dass die verwendete Samplerate höher sein sollte, als die Nyquist-Rate, da Quantisierungsfehler beim Samplingprozess das Ergebnis zusätzlich beeinflussen.
- Als Quantisieren bezeichnet man den Vorgang, bei dem der Wertebereich in Intervalle unterteilt wird (Wertdiskretisierung).

Die Diskrete Fouriertransformation (DFT)

- In der digitalen Signalverarbeitung wird das Ausgangssignal in den Raum der komplexen Zahlen überführt und als Frequenzspektrum mit realem und imaginären Anteil betrachtet und analysiert.

- Die FOURIERanalyse bildet die Grundlage bekannter Verfahren zur Kompression, Merkmalsextraktion und Mustererkennung.

gegeben: Vektor v $\in \mathbb{R}^n$(Messdaten-Abtastwerte)
$v = [v_0, v_1, \ldots, v_{n-1}]^T$

n = Anzahl Messpunkte in einer Periode

gesucht: Transformation in den Frequenzbereich (alle vorkommenden Frequenzen werden dargestellt - Frequenzspektrum)

- für die FOURIERtransformation gilt folgende Formel:

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix} = DFT \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ \vdots \\ v_{n-1} \end{pmatrix}$$

$$c_k = \frac{1}{n} \cdot \sum_{j=0}^{n-1} v_j \cdot \left[\cos\left(\frac{2p \cdot k \cdot j}{n} \right) - i \sin\left(\frac{2p \cdot k \cdot j}{n} \right) \right] \qquad \mathrm{k} = 0,1,...,\mathrm{n}-1$$

- oder umgeformt in Exponentialschreibweise:

$$c_k = \frac{1}{n} \cdot \sum_{j=0}^{n-1} v_j \cdot e^{-i \frac{2p \cdot k \cdot j}{n}} \qquad \mathrm{k} = 0,1,...,\mathrm{n}-1$$

- bzw. alternativ in Matrixschreibweise:

$$\overline{w} = e^{-i \frac{2p}{n}}$$

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix} = \frac{1}{n} \cdot \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \overline{w}^{\,1} & \overline{w}^{\,2} & \cdots & \overline{w}^{\,(n-1)} \\ 1 & \overline{w}^{\,2} & \overline{w}^{\,4} & \cdots & \overline{w}^{\,2\cdot(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \overline{w}^{\,(n-1)} & \overline{w}^{\,(n-1)\cdot 2} & \cdots & \overline{w}^{\,(n-1)(n-1)} \end{pmatrix} \cdot \begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ \vdots \\ v_{n-1} \end{pmatrix}$$

- ein Beispiel:

gegeben: eine Funktion $f(t) = 0.7 \cdot \cos(2p \cdot t) + 0.3 \cdot \cos(9 \cdot 2p \cdot t)$ (meist unbekannt) wird abgetastet.

T=1 , n=20
v=[1.0000 ; 0.3804 ; 0.8090 ; 0.2351 ; 0.3090 ;
-0.0000 ; -0.3090 ; -0.2351 ; -0.8090 ; -0.3804 ;
-1.0000 ; -0.3804 ; -0.8090 ; -0.2351 ; -0.3090 ;
0.0000 ; 0.3090 ; 0.2351 ; 0.8090 ; 0.3804]^T

gesucht: c $\in \mathbb{C}^n$ c:=DFT(v)

c wird einfach nach obiger Formel berechnet, es ergibt sich folgender Vektor (FOURIERkoeffizienten) :

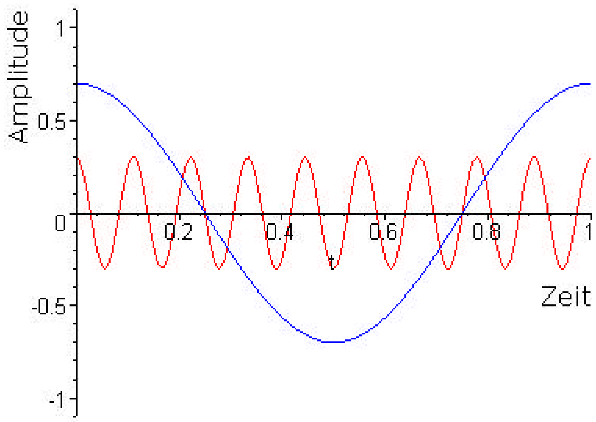
c= [0.0000 ; 0.3500 ; 0.0000 ; 0.0000 ; 0.0000 ;
0.0000 ; 0.0000 ; 0.0000 ; 0.0000 ; 0.1500 ;
0.0000 ; 0.1500 ; 0.0000 ; 0.0000 ; 0.0000 ;
0.0000 ; 0.0000 ; 0.0000 ; 0.0000 ; 0.3500]^T

Bemerkung: alle Imaginäranteile sind verschwunden(da, nur Kosinusanteile in der gegebenen Funktion vorhanden sind)

- Die FOURIERkoeffizienten repräsentieren ein diskretes Spektrum. Für reelle Funktionen sind sie symmetrisch. Zur Darstellung wird deshalb nur die Hälfte der FOURIERkoeffizienten genutzt. Die Amplitude ergibt sich dann zu $y_k = 2 \cdot |c_k|$ (Amplitudenspektrum).

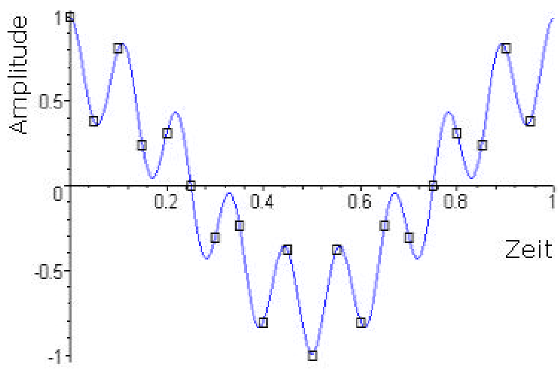
- Es ist auch möglich die Phase über der Frequenz abzutragen. Für die Phase ergibt sich: $j_k = \arctan\left(\frac{\mathrm{Im}(c_k)}{\mathrm{Re}(c_k)}\right)$ (Phasenspektrum).

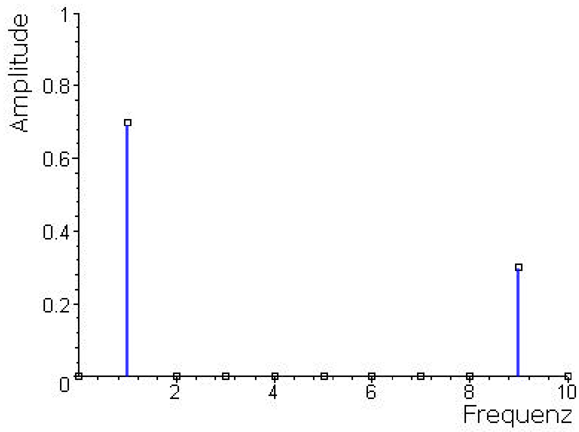
- grafische Darstellung:



$$f_1(t) = 0.7 \cdot \cos(2p \cdot t)$$
$$+$$
$$f_2(t) = 0.3 \cdot \cos(9 \cdot 2p \cdot t)$$

Das Bild zeigt die jeweilige Funktion im Intervall [0,1]. Sie werden addiert und es entsteht die resultierende Funktion f(t).

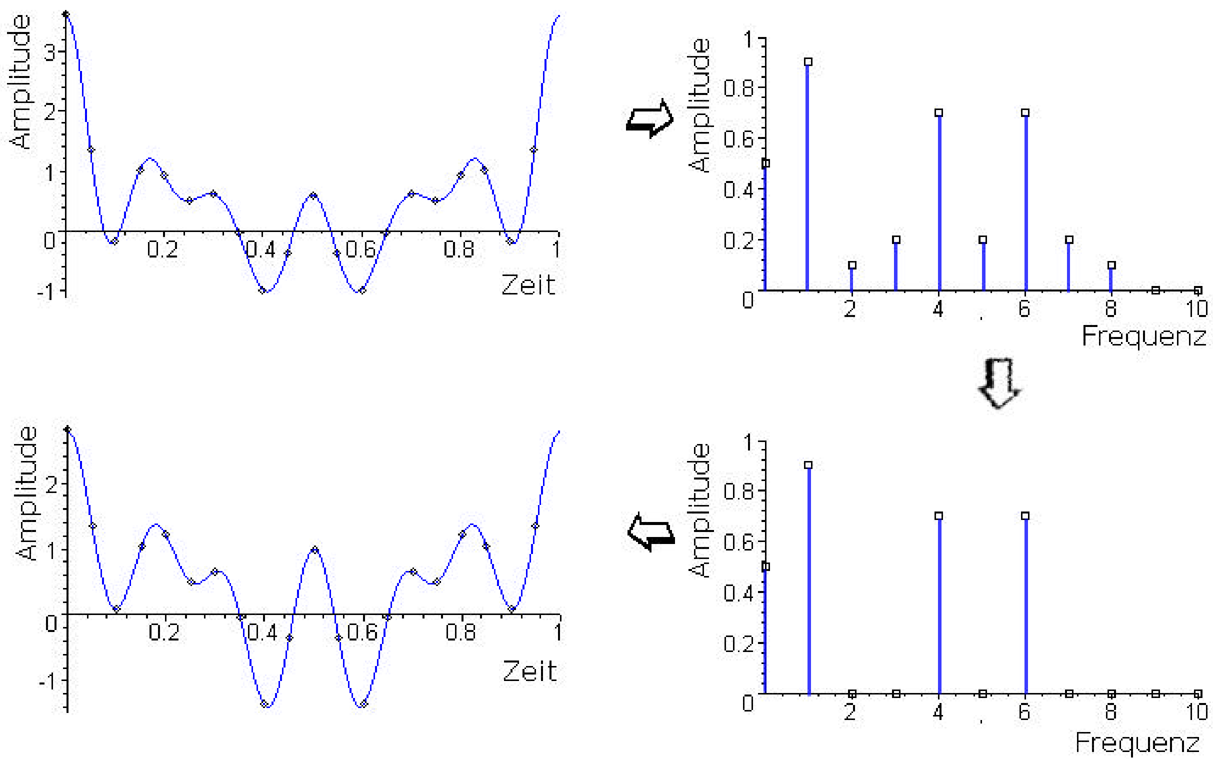




$$f(t) = 0.7 \cdot \cos(2p \cdot t)$$
$$+ 0.3 \cdot \cos(9 \cdot 2p \cdot t)$$

Es werden genau die Frequenzanteile angezeigt, die auch in der Ausgangsfunktion enthalten waren.

- bei der digitalen Filterung wird das Signal analysiert, die Frequenzanteile herausgefiltert (d.h. entsprechende FOURIERkoeffizienten werden auf 0 gesetzt) und anschließend rücktransformiert.



Im Bild sind kaum Veränderungen zwischen dem Ausgangssignal und dem erzeugten Signal sichtbar, d.h. die Schwingungen mit geringer Amplitude tragen kaum zur Form der Grundschwingung bei.

Dies kann zur (verlustbehafteten) Kompression genutzt werden in dem man die FOURIERkoeffizienten mit geringem Betrag auf Null setzt.

Es lassen sich auch sehr einfach Hoch-, Tief- oder Bandpassfilter implementieren. (Die entsprechenden FOURIERkoeffizienten der „ungewollten" Frequenzen werden auf Null gesetzt).

Die Inverse Diskrete Fouriertransformation (IDFT)

Die inverse diskrete FOURIERtransformation stellt das Gegenstück zur DFT dar. Sie wandelt die Daten vom Frequenzbereich wieder zurück in den Zeitbereich.

gegeben: Vektor c $\in \mathbb{C}^n$ (FOURIERkoeffizienten)
$c = [c_0, c_1, ..., c_{n-1}]^T$
n = Anzahl Messpunkte

gesucht: Ausgangssignal im Zeitbereich

- für die inverse FOURIERtransformation gilt folgende Formel:

$$\begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ \vdots \\ v_{n-1} \end{pmatrix} = IDFT \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix}$$

$$v_j = \sum_{k=0}^{n-1} c_k \cdot e^{i\frac{2p \cdot k \cdot j}{n}} \qquad \mathrm{j = 0,1,..., n-1}$$

- oder wieder in Matrixschreibweise:

$$w = e^{i\frac{2p}{n}}$$

$$\begin{pmatrix} v_0 \\ v_1 \\ v_2 \\ \vdots \\ v_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w^{1} & w^{2} & \cdots & w^{(n-1)} \\ 1 & w^{2} & w^{4} & \cdots & w^{2\cdot(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & w^{(n-1)} & w^{(n-1)\cdot 2} & \cdots & w^{(n-1)(n-1)} \end{pmatrix} \cdot \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix}$$

- ein Beispiel:

gegeben: n=20

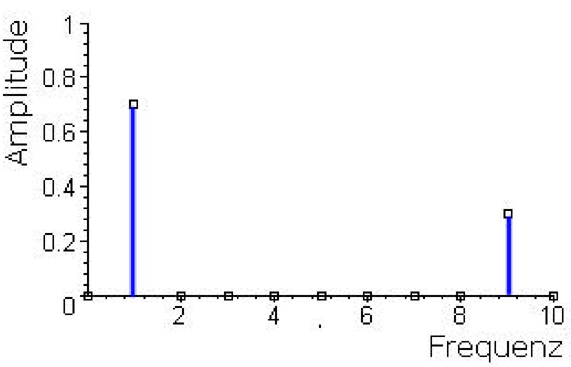
c= [0.0000 ; 0.3500 ; 0.0000 ; 0.0000 ; 0.0000 ;
0.0000 ; 0.0000 ; 0.0000 ; 0.0000 ; 0.1500 ;
0.0000 ; 0.1500 ; 0.0000 ; 0.0000 ; 0.0000 ;
0.0000 ; 0.0000 ; 0.0000 ; 0.0000 ; 0.3500 $]^T$

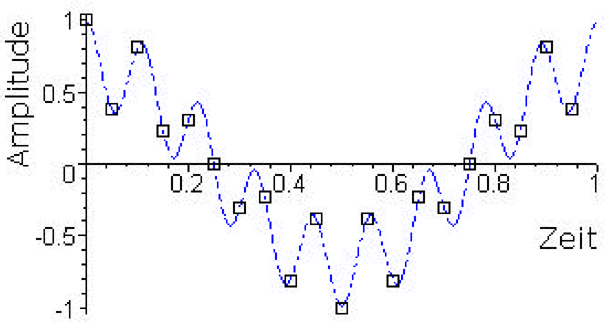
gesucht: $v \in \mathbb{R}^n$

v=[1.0000 ; 0.3804 ; 0.8090 ; 0.2351 ; 0.3090 ;
-0.0000 ; -0.3090 ; -0.2351 ; -0.8090 ; -0.3804 ;
-1.0000 ; -0.3804 ; -0.8090 ; -0.2351 ; -0.3090 ;
0.0000 ; 0.3090 ; 0.2351 ; 0.8090 ; 0.3804 $]^T$

Die Werte entsprechen exakt den Ausgangswerten.

- grafische Darstellung:





Die Schnelle Fouriertransformation (FFT)

- bei DFT ist der Aufwand $O(n^2)$, für praktische Anwendungen mit einer großen Anzahl an Eingangsdaten ist dies zu lange
- deshalb suchte man einen schnelleren Algorithmus zur Berechnung der FOURIERtransformation: der Aufwand war jetzt nur noch $O(n \cdot \log n)$

- dies hat erst Anwendungen auch auf kleineren Prozessrechnern ermöglicht z.B. Bildverarbeitung, Glättung von Daten
- weitere Anwendungen sind die schnelle Polynommultiplikation bzw. Multiplikation zweier großer nichtnegativer Zahlen

- klassisches divide & conquer Prinzip
- der FFT Algorithmus beruht auf einer geschickten Zusammenfassung von Summanden, um gewisse Symmetrieeigenschaften auszunutzen.

Voraussetzung: $n = 2^p$

1. Teilung des Ausgangsvektors in 2 Vektoren (gerade und ungerade Indizes) mit jeweils nur noch der halben Länge

$$c_k = \frac{1}{n} \cdot \sum_{j=0}^{n-1} v_j \cdot e^{-i\frac{2p \cdot k \cdot j}{n}}$$

$$c_k = \frac{1}{n} \cdot \left(\sum_{j=0}^{\frac{n}{2}-1} v_{2j} \cdot e^{-i\frac{2p \cdot k \cdot 2j}{n}} + \sum_{j=0}^{\frac{n}{2}-1} v_{2j+1} \cdot e^{-i\frac{2p \cdot k \cdot (2j+1)}{n}} \right) \qquad n = 2 \cdot m$$

$$c_k = \frac{1}{n} \cdot \left(\sum_{j=0}^{m-1} v_{2j} \cdot e^{-i\frac{2p \cdot k \cdot j}{m}} + \overline{w}^k \sum_{j=0}^{m-1} v_{2j+1} \cdot e^{-i\frac{2p \cdot k \cdot j}{m}} \right) \qquad \overline{w} = e^{-i\frac{2p}{n}}$$

$$k = 0,1,\ldots,n-1$$

$$m = \frac{n}{2}$$

für $e^{-i\frac{2p \cdot k \cdot j}{m}}$ gilt: $e^{-i\frac{2p \cdot (k+m) \cdot j}{m}} = e^{-i\left(\frac{2p \cdot k \cdot j}{m} + 2p \cdot j\right)} = e^{-i\frac{2p \cdot k \cdot j}{m}}$

für $\overline{w}^k$ gilt: $\overline{w}^{k+m} = e^{-i\frac{2p}{n}\cdot\left(k + \frac{n}{2}\right)} = e^{-i\left(\frac{2p \cdot k}{n} + p\right)} = -\overline{w}^k$

dadurch kann die obige Formel umgeschrieben werden:

$$c_k = \frac{1}{n}\cdot\left(\sum_{j=0}^{m-1} v_{2j}\cdot e^{-i\frac{2p\cdot k\cdot j}{m}} + \overline{w}^k \sum_{j=0}^{m-1} v_{2j+1}\cdot e^{-i\frac{2p\cdot k\cdot j}{m}}\right)$$

$$c_{k+m} = \frac{1}{n}\cdot\left(\sum_{j=0}^{m-1} v_{2j}\cdot e^{-i\frac{2p\cdot k\cdot j}{m}} - \overline{w}^k \sum_{j=0}^{m-1} v_{2j+1}\cdot e^{-i\frac{2p\cdot k\cdot j}{m}}\right) \qquad k = 0,1,\ldots,\frac{n}{2}\text{-}1$$

2. Es wird für jeden Vektor erneut das Unterprogramm aufgerufen
3. Aufteilung wieder in zwei Vektoren mit geraden und ungeraden Indizes
   .
   .
   .
4. Abbruch: Vektor der Länge 1, Anwendung von DFT auf Vektor trivial
   $c_0 := v_0$;
5. Ergebnis wird der nächsthöheren Stufe mitgeteilt
6. Vektor wird zu einem doppelt so langen Vektor zusammengefügt (Formel siehe oben).

- ein Beispiel:
  gegeben: $v \in \mathbb{R}^n$, $n=4$, $v=[1,2,3,4]^T$
  gesucht: $c \in \mathbb{C}^n$, $c := FFT(v)$

$[1,2,3,4]^T$ — $n=4$

$[1,3]^T$ $[2,4]^T$ — $n=2$

$[1]$ $[3]$ $[2]$ $[4]$ — $n=1$

$c_0 = 1$ $c_0 = 3$ $c_0 = 2$ $c_0 = 4$

$c_0 = (1+3) = 4$ $c_0 = (2+4) = 6$

$c_1 = (1-3) = -2$ $c_1 = (2-4) = -2$

$$c_0 = \frac{1}{4}(4+6) = 2.5$$

$$c_2 = \frac{1}{4}(4-6) = -0.5$$

$$c_1 = \frac{1}{4}(-2 + e^{-i\frac{2\pi}{4}}\cdot(-2)) = -0.5 + 0.5\cdot i \qquad e^{-i\frac{2\pi}{4}} = -i$$

$$c_3 = \frac{1}{4}(-2 - e^{-i\frac{2\pi}{4}}\cdot(-2)) = -0.5 - 0.5\cdot i$$

es ergibt sich also $c = \begin{pmatrix} 2.5 \\ -0.5+0.5\cdot i \\ -0.5 \\ -0.5-0.5\cdot i \end{pmatrix}$

- für die inverse schnelle FOURIERtransformation (IFFT) wird ein analoges Verfahren angewandt (Faktor $\frac{1}{n}$ wird weggelassen und Übergang zum einfachen Komplexen).

Die Diskrete Kosinustransformation (DCT)

- Spezialfall der FOURIERtransformation
- um 1970 entwickelt
- besonders für Bilddaten (JPEG) gut geeignet

Die Diskrete Kosinustransformation wandelt wie die FOURIERtransformation Daten in eine Beschreibung um, bei der die Werte durch Frequenzen und Amplituden repräsentiert werden

Für Bilddaten geben die Frequenzen an, wie schnell sich Farben innerhalb eines Bildes verändern, die Amplituden beschreiben die Stärke der Veränderung.
Es findet hauptsächlich die zweidimensionale Variante Verwendung. Sie arbeitet üblicherweise auf 8x8 Blöcken des zu kodierenden Bildes. Wenn in einem Pixelblock keine plötzlichen Farbsprünge, sondern nur ähnliche Farben sind, haben auch die DCT-Koeffizienten außer dem Ersten ähnliche, kleine Werte.

weitere Anwendung: MPEG - Videokompression
MP3 - Kompression

Formel: ähnlich wie DFT, aber ohne Sinusanteil

1D-Transformation

Hintransformation (DCT) :

$$F_k = \sum_{j=0}^{n-1} f_j \cdot \cos\left(\frac{p \cdot k \cdot \left(j + \frac{1}{2}\right)}{n}\right) \qquad k = 0,...,n\text{-}1$$

Rücktransformation (IDCT) :

$$f_j = \frac{2}{n} \cdot \sum_{k=0}^{n-1} F_k \cdot \cos\left(\frac{p \cdot k \cdot \left(j + \frac{1}{2}\right)}{n}\right) \qquad j = 0,...,n\text{-}1$$

2D-Transformation:

Hintransformation:

$$F_{k_1 k_2} = \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} f_{j_1 j_2} \cdot \cos\left(\frac{p \cdot k_1 \cdot \left(j_1 + \frac{1}{2}\right)}{n_1}\right) \cdot \cos\left(\frac{p \cdot k_2 \cdot \left(j_2 + \frac{1}{2}\right)}{n_2}\right)$$

Rücktransformation:

$$f_{j_1 j_2} = \frac{4}{n_1 n_2} \sum_{k_1=0}^{n_1-1} \sum_{k_2=0}^{n_2-1} f_{k_1 k_2} \cdot \cos\left(\frac{p \cdot k_1 \cdot \left(j_1 + \frac{1}{2}\right)}{n_1}\right) \cdot \cos\left(\frac{p \cdot k_2 \cdot \left(j_2 + \frac{1}{2}\right)}{n_2}\right)$$

Unterschiede zur DFT:

- DFT erzeugt komplexe Zahlen; DCT im Gegensatz nur reelle Zahlen
- DFT nimmt an, dass die Funktion periodisch ist

Es gibt auch für die DCT eine schnelle Variante, auf die hier aber nicht näher eingegangen werden soll.

Grenzen & Probleme

- fehlende Lokalisation auf Zeitachse

- alle Daten wurden gleich bewertet (Rechteckfenster), dadurch entstehen am Anfang und Ende des Datensatzes evtl. Sprünge, die bei der Transformation Frequenzbänder vortäuschen, die im Signal überhaupt nicht enthalten waren.
  Ausweg:
  Multiplikation des zu transformierenden Signals mit einer Bewertungsfunktion (Dreiecks-, Hamming-, oder Hanningfenster)
  Die Daten am Anfang und Ende eines Fensters werden dadurch schwächer bewertet.

- FOURIERkoeffizienten enthalten Informationen über die Funktion aus dem gesamten Definitionsbereich, man kann ihnen nicht ansehen wo Sprungstellen oder ausgeprägte Spitzen der Funktion liegen

- Problem wird beseitigt durch die WAVELET-THEORIE (Weiterentwicklung der FOURIERanalyse)

Literaturverzeichnis

[1] Salomon, David: Data Compression.
Springer Verlag, New York 1997

[2] Locher, Franz: Numerische Mathematik für Informatiker.
Springer Verlag, Berlin 1993

[3] Huckle, Thomas: Numerik für Informatiker.
Springer Verlag, Berlin 2002

[4] Sedgewick, R.: Algorithmen.
Addison-Wesley, Bonn 1991

[5] Paul, Reinhold: Elektrotechnik und Elektronik für Informatiker.
Teubner, Stuttgart 1995

[6] Schulz, Dieter: PC-gestützte Meß- und Regeltechnik
Grundlagen und praktische Anwendungen.
Franzis, Feldkirchen 1994