Fourier Transformation und der Phase Vocoder

Thomas Neuhaus, 4/2004

Geschichtlicher Hintergrund

- Jean Baptiste Joseph de Fourier (1768-1830)
- Beginn der Entwicklung der Theorie der trigonometrischen Reihen in "Théorie analytique de la chaleur" (1822)
- Mit Auftreten der Elektronenrechner in den 50er und 60er Jahren erste Implementationen der Diskreten Fourier transformation
- 1965 Veröffentlichung der Aufsatzes *An Algorithm for the Machine Calculation of Complex Fourier Series* von James W. Cooley und John W. Tukey. Verschiedene Forscher verweisen auf verschiedene Vorgänger, so z.B. Runge 1942
- J.L. Flanagan and R.M. Golden, "Phase vocoder," Bell Syst. Tech. J., vol. 45, pp. 1493–1509, Nov 1966.

Mathematische Grundlagen der FT

- Original Fourier Integrale

$$\mathcal{F}\{f(t)\} = F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-iwt}dt$$

$$\mathcal{F}^{-1}\{F(\omega)\} = f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{iwt}d\omega$$

Interpretation: *Jede **periodische** Funktion f(x) ist darstellbar als eine Summe aus Sinus- und Cosinusgliedern, deren Frequenzen **ganzzahlige Vielfache** der Basisfrequenz der Ausgangsfunktion sind. (vgl. Obertöne)*

Mathematische Grundlagen

- Komplexe Zahlen bestehen aus einem Realteil und einem Imaginärteil. Der Imaginärteil wird als Faktor von *i* verstanden, wobei *i* (und -*i*) Lösungen der Gleichung

$$i^2 + 1 = 0$$

sind. Notiert werden komplexe Zahlen als

$(a + bi)$ oder einfach (a, b)

Bei Komplexen Zahlen gelten die normalen Rechenregeln. Z.B.:

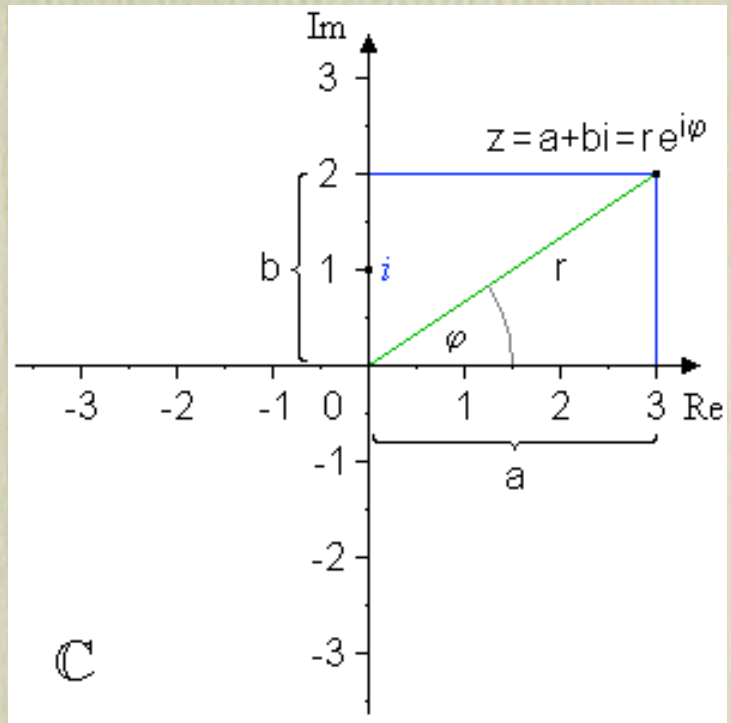
$$(a + bi)(c + di) = ac + adi + bci + bdi^2 = ac - bd + (ad + bc)i$$

$$\frac{a + bi}{c + di} = \frac{(a + bi)(c - di)}{(c + di)(c - di)} = \frac{ac + bd}{c^2 + d^2} + \frac{bc - ad}{c^2 + d^2}i$$

Grafisch können komplexe Zahlen als Punkte in einer Ebene angesehen werden. Auf der X-Achse wird dann der Real- auf der Y-Achse der Imaginärteil angetragen.
Komplexe Zahlen haben dann die Eigenschaften von Vektoren mit Länge (Betrag) und Winkel. Es gilt für alle komplexen Zahlen *z=(a+bi)*:

$$r = |z| = \sqrt{a^2 + b^2}$$

$$\varphi = \arctan(\frac{a}{b})$$



Exponentialfunktionen mit komplexen Exponenten sind periodisch, wie man aus den jeweiligen Reihenentwicklungen herleiten kann. Es gilt speziell für die E-funktion zur Basis *e*:

$$e^{\omega i} = \cos\omega + i\sin\omega$$

Daher kann man Sinusoide auch als E-Funktion mit komplexem Exponenten darstellen, und hat gleichzeitig Information über Betrag (Amplitude) und Phase (Winkel)

Entwicklung der FT

- Diskretisierung DFT: Summenform statt Integral

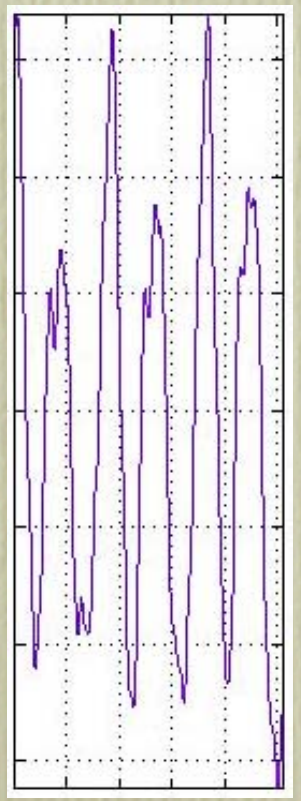
$$S(f_k) = \sum_{j=0}^{j=N} s(t_j) e^{i2\pi f_k t_j} (t_{j+1} - t_j) \quad k = 0, 1, \ldots, N-1$$

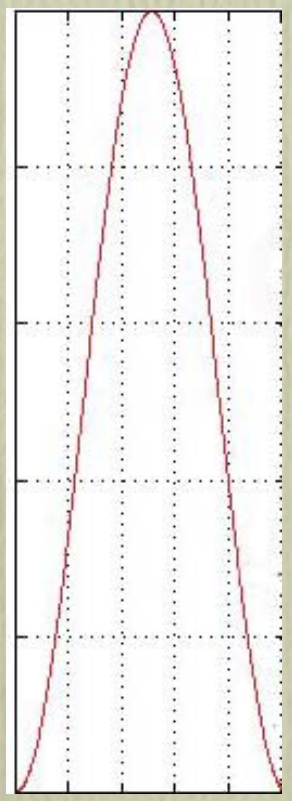
Direkte Implementation führt zu Algorithmus mit quadratischer Komplexität.

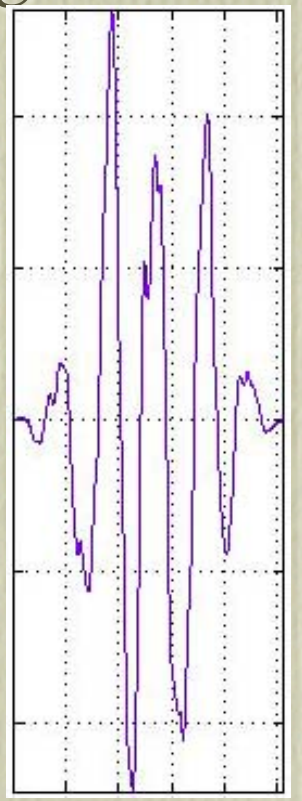
Die N Gleichungen lassen sich als Matrix darstellen. Für $N=2^n$ lassen sich bestimmte Optimierungen nutzen, die dann zu den FFT-Algorithmen führen.

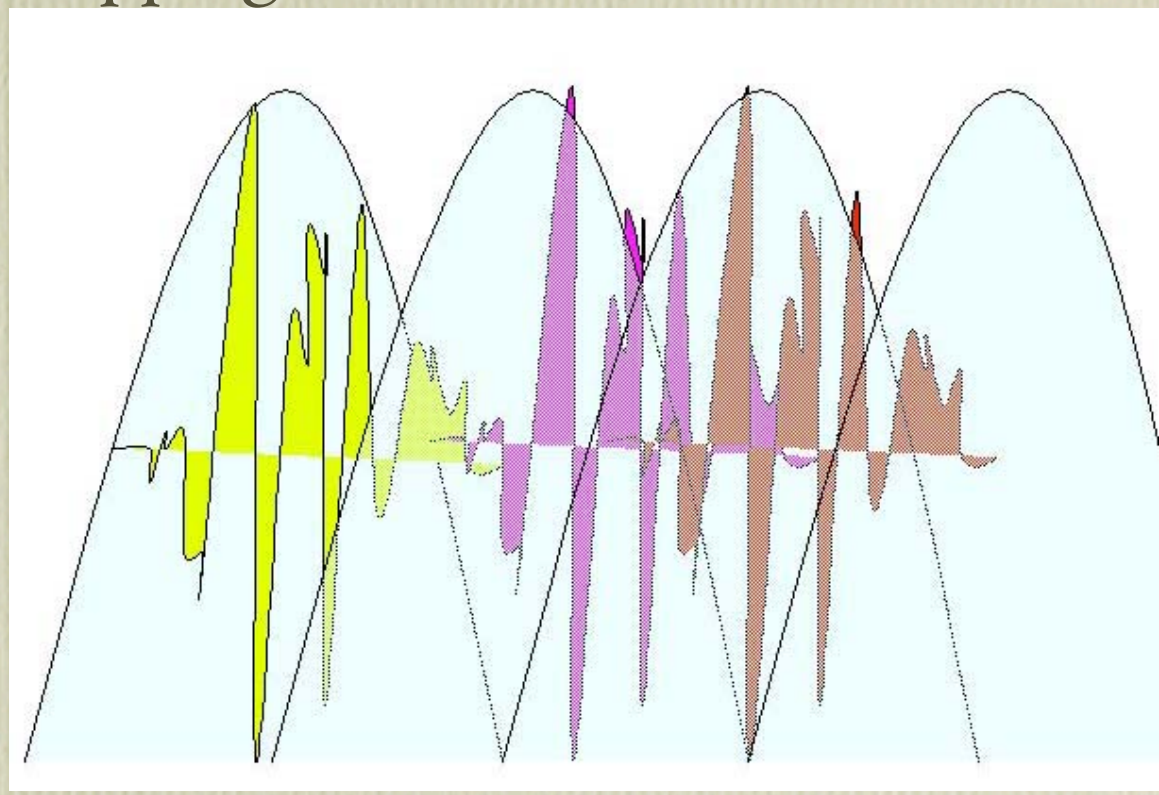
Nicht Periodische Signale (Alle wirklich interessanten)

- STFFT: Frames und Bins
- Windowing und Overlapping









Problemfelder der FFT

- Zeitauflösung vs. Frequenzauflösung (”Unschärferelation”)
- Linearität führt zu intervallisch niedriger Auflösung in niedrigen Frequenzen

Der Phasevocoder

- Resyntheseverfahren auf der Basis von STFFT Daten
- Timestretching/compression durch zeitliche Neuanordnung der Frames
- Transposition durch Neuinterpretation des Frequenzgehaltes (oder Zeitdehnung/Stauchung +Resampling)

CSound Implementation

- Mit dem Utility Programm *pvanal* werden die Analysedaten erzeugt.
- Wichtige Größen (Framesize, Overlap etc.) können per Kommandozeile eingegeben werden
- Opcodes, die dann STFT Resynthese implementieren sind *ktableseg, pvadd, pvbufread, pvcross, pvinterp, pvoc, pvread, tableseg, tablexseg,* und *vpvoc.*

Beispiele

- Simple Time-Stretching/Compression

```
sr=44100
kr=4410
nchnls=1
ksmps=10

instr 1

idur    =       p3
istart  =       p4
iend    =       p5


ktimpnt line    istart,idur,iend
asig    pvoc    ktimpnt,1,"mahoute.pv"
        out asig
endin
```

```
i 1 0 5 .1 1.1  ; Timestretch 1:5
i 1 6 5 1.1  .1  ; Timestretch 1:5 und rueckckwrts
i 1 12 1 0 5 ; Timecompression 1:5
i 1 14 10 .1 .2 ; Timestretch 1:100
e
```

- Cross-Synthese

```
sr=44100
kr=4410
nchnls=1
ksmps=10

instr 1

ktime1  line 0, p3, p3 ; used as index in the speech pv file
ktime2  line 0, p3, p3 ; used as index in the choral pv file
kinterp line 0,p3,1
        pvbufread ktime1, "mahoute.pv"
apv     pvinterp ktime2,1,"al-jorn.pv",1,1,1,1,kinterp,1-kinterp
        out 0.1*apv
endin
```

```
i 1 0 15; 15 seconds interpolation
```