Aspekte der Signalverarbeitung in der Spracherkennung

Dieses Manuskript stellt einige Auszüge aus der Vorlesung Spracherkennung und Dialogsysteme im SS2005 an der LMU München, Inst. f. Informatik, Lehrstuhl Medieninformatik (Prof. H. Hussmann) zusammen. Es ist nur für HörerInnen dieser Vorlesung als Begleitmaterial geeignet.

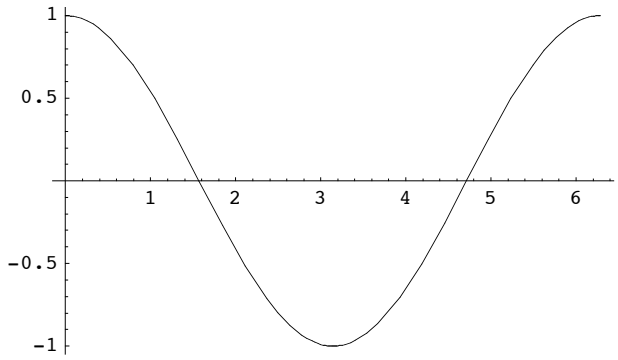
Autor: *Prof. M. Spies, LMU.*

Diskrete Fouriertransformation

Hier ist eine einfache Veranschaulichung einiger Eigenschaften der diskreten Fourier-Transformation (DFT). Für die mathematischen Grundlagen und die Eigenschaften der Fourier-Transformation s. Oppenheim, Schafer und Buck, Discrete-Time Signal Processing, Prentice Hall, 1999.

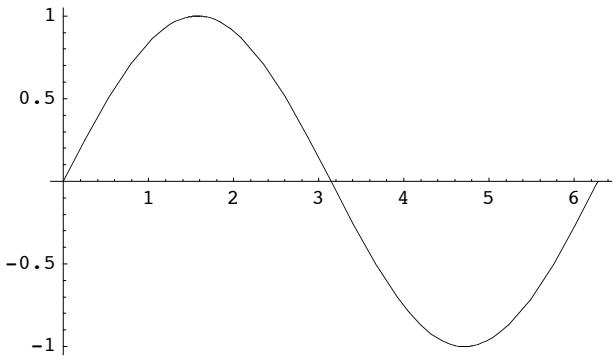
Die DFT einer periodischen Zeitreihe der Periode N beschreibt diese durch eine gewichtete Summe periodischer Referenzfunktionen. Dazu dienen die komplexen Exponentialfunktionen Exp[i φ]. Wie man mithilfe der Reihenentwicklung nachweist, haben die Exp[i φ] i.B. folgende Eigenschaften:
Ihr Realteil ist eine Cosinus-, ihr Imaginärteil eine Sinusfunktion, beide mit der Periode 2π. Daher die Darstellung
Exp[i φ] = Cos[φ] + i Sin[φ]

```
In[79]:= Plot[Re[Exp[ⅈ φ]], {φ, 0, 2 π}]
```




```
Out[79]= - Graphics -
```

In[80]:=
```
Plot[Im[Exp[ⅈ φ]], {φ, 0, 2 π}]
```

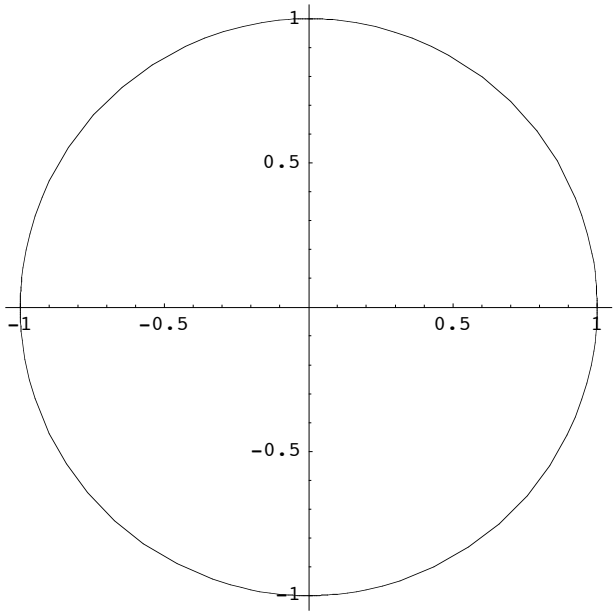


Out[80]= - Graphics -

Alle Punkte von Exp[i φ] liegen auf dem Einheitskreis der komplexen Zahlenebene. Dies ergibt sich sofort aus der Definition des Betrags für komplexe Zahlen.

In[81]:=
```
Cis[φ_] := Exp[ⅈ φ]
```

In[82]:=
```
ParametricPlot[{Re[Cis[x]], Im[Cis[x]]}, {x, 0, 2 π}, AspectRatio → 1]
```



Out[82]= - Graphics -

Anschaulich wird dies durch einen gegen den UZS mit der Frequenz 1 sich drehenden Ortsvektor erzeugt
Beachte: Bei $\varphi = \pi$ "springt" die Arg-Funktion:

In[83]:=
```
Arg[-1 - 0.1 ⅈ]
```

Out[83]= -3.04192

```
In[84]:= Arg[-1 + 0.1 i]

Out[84]= 3.04192
```

-- so macht die Arg-Funktion bei der komplexen Zahl (-1, 0) einen Sprung von π auf $-\pi$ (in Richtung fallender Werte der imaginären Achse, d.h. bei Bewegung des Ortsvektors in der komplexen Ebene gegen UZS)

Für die Herleitung der DFT einer periodischen Zeitreihe der Periode N bilden wir nun N Referenzfunktionen mit Periodizitäten von k = 0 bis N-1. Dazu samplen wir nun Exp[iφ] an jeweils k Punkten mit gleich langen Sehnen auf dem Einheitskreis.

Intuitiv für reelle eindimensionale Zeitreihen: Wir bilden Räder des Umfangs 2 π N mit jeweils k in gleichen Winkeln angebrachten Speichen und ohne Felgen -- oder auch entspr. Zahnräder. Diese Räder lassen wir über die reelle Zeitreihe drehen -- eine Umdrehung des Rads entspricht genau einer Periode der betrachteten Zeitreihe. Nehmen wir an, wir hätten ein Rad mit 3 Speichen. Wenn in der Zeitreihe jeder 3. Wert hoch ist, würde jede Speiche auf eine hohe Stützstelle treffen und das Rad insgesamt oberhalb der Werte der Zeitreihe drehen. Das entspricht aber einer Präsenz der Frequenz 3/N in der Zeitreihe, d.h. der Periode N/3.

Der Fall k=0 entspricht der Periode (unendlich) und einer konstanten Referenzfunktion. Sie ist bei der Analyse nützlich, um ein von 0 verschiedenes Base Level der Zeitreihe zu modellieren.

Um verschiedene Phasen derselben Periode zu modellieren, stehen uns je Periode ein Cosinus-Rad und ein Sinus-Rad zur Verfügung. Das Cosinus-Rad setzt beim 0.Sample der Zeitreihe mit einer Speiche an; das Sinus-Rad "um eine Vierteldrehung" versetzt. Eine gewichtete Mischung aus Cosinus- und Sinus-Analyse geht dann als komplexer Wert in die Fourieranalyse ein.

Diesen Rädern (=periodischen Funktionen) in der Zeitdomäne entsprechen Sequenzen von Fourierkoeffizienten in der Frequenzdomäne. Da wir hier im Bereich der komplexen Exponentialfunktionen alles in der Periode 2π unterbringen müssen, entsprechen den verschiedenen Speichenanzahlen der periodischen Funktionen in der Zeitdomäne hier verschiedene Umlaufgeschwindigkeiten (eben Frequenzen!, oder genauer: Kreisfrequenzen) des Einheitsvektors, der bei der DFT-Analyse im UZS beginnend bei (1,0) umläuft (bei der DFT-Synthese gegen den UZS).
Bsp. für k=1, N=7:

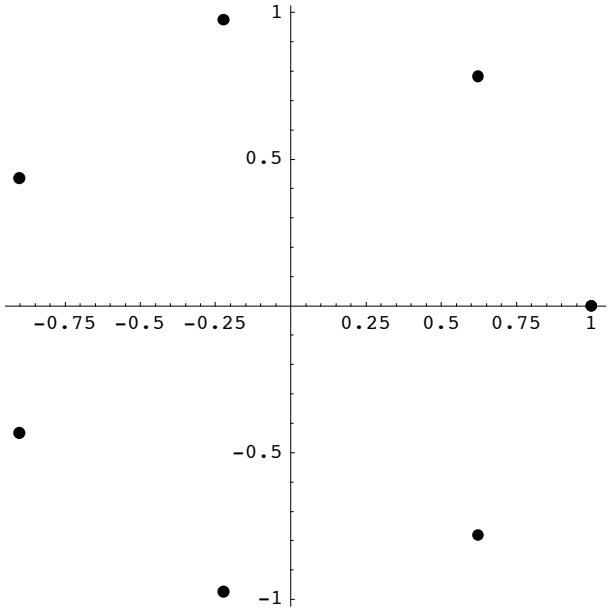
```
In[85]:= compcycle7 = Table[Exp[i 2 π n / 7], {n, 0, 7}]

Out[85]= {1, e^(2 i π/7), e^(4 i π/7), e^(6 i π/7), e^(-6 i π/7), e^(-4 i π/7), e^(-2 i π/7), 1}

In[86]:= re7 = Map[Re, compcycle7]

Out[86]= {1, Cos[2 π/7], Cos[4 π/7], Cos[6 π/7], Cos[6 π/7], Cos[4 π/7], Cos[2 π/7], 1}

In[87]:= im7 = Map[Im, compcycle7]

Out[87]= {0, Sin[2 π/7], Sin[4 π/7], Sin[6 π/7], -Sin[6 π/7], -Sin[4 π/7], -Sin[2 π/7], 0}
```

Man beachte die Symmetrien in diesen Fourierkoeffizienten. Diese werden in den FFT-Algorithmen ausgenutzt.

```
In[88]:= cycvals7 = Thread[List[ re7, im7]]
```

$$\text{Out[88]= } \left\{\{1, 0\}, \left\{\text{Cos}\left[\frac{2\pi}{7}\right], \text{Sin}\left[\frac{2\pi}{7}\right]\right\}, \left\{\text{Cos}\left[\frac{4\pi}{7}\right], \text{Sin}\left[\frac{4\pi}{7}\right]\right\}, \left\{\text{Cos}\left[\frac{6\pi}{7}\right], \text{Sin}\left[\frac{6\pi}{7}\right]\right\}, \left\{\text{Cos}\left[\frac{6\pi}{7}\right], -\text{Sin}\left[\frac{6\pi}{7}\right]\right\}, \left\{\text{Cos}\left[\frac{4\pi}{7}\right], -\text{Sin}\left[\frac{4\pi}{7}\right]\right\}, \left\{\text{Cos}\left[\frac{2\pi}{7}\right], -\text{Sin}\left[\frac{2\pi}{7}\right]\right\}, \{1, 0\}\right\}$$

```
In[89]:= ListPlot[cycvals7, AspectRatio → 1, PlotStyle → PointSize[0.02]]
```




```
Out[89]= - Graphics -
```

für k>1 erhält man dieselben Punkte auf dem Einheitskreis, allerdings wandert der Ortsvektor mit wachsendem n jeweils auf den k-nächsten Punkt modulo N; Bsp:

```
In[90]:= compcycle27 = Table[Exp[i 2 π n 2 / 7], {n, 0, 7}]

General::spell1 :
 Possible spelling error: new symbol name "compcycle27" is similar to existing symbol "compcycle7". Mehr…
```

$$\text{Out[90]= } \left\{1, e^{\frac{4 i \pi}{7}}, e^{-\frac{6 i \pi}{7}}, e^{-\frac{2 i \pi}{7}}, e^{\frac{2 i \pi}{7}}, e^{\frac{6 i \pi}{7}}, e^{-\frac{4 i \pi}{7}}, 1\right\}$$

Nun Beispiele für DFT. In *Mathematica* ist es zunächst erforderlich, die entsprechenden Built-In Funktionen mit geeigneten Parametern für DSP einzustellen ...

```
In[91]:= SetOptions[Fourier, FourierParameters → {1, -1}];
         SetOptions[InverseFourier, FourierParameters → {1, -1}]

Out[91]= {FourierParameters → {1, -1}}
```

Wir beginnen mit einer einfachen periodischen Reihe in Form eines Walzertaktes:

```
In[92]:= peri3 = {1, 0, 0}

Out[92]= {1, 0, 0}
```

Ihre DFT ist reellwertig und enthält alle 3 möglichen Cosinusfunktionen mit dem Gewicht 1:

```
In[93]:= four3 = Fourier[peri3]

Out[93]= {1. + 0. i, 1. + 0. i, 1. + 0. i}
```

Um das Ergebnis zu überprüfen, visualisieren wir mithilfe der diskreten Fouriersynthese. Zunächst definieren wir eine Liste mit den entspr. Koeffizenten:

```
In[94]:= wsyncoffs[N_] := Table[Exp[ i 2 π n k / N], {k, 0, N - 1}]

In[95]:= wsyncoffs[3]

Out[95]= {1, e^(2 i n π / 3), e^(4 i n π / 3)}
```

durch komponentenweise Multiplikation des Koeffizientenvektors mit den Fourierkoeffizienten erhalten wir die Komponenten der Fouriersynthese:

```
In[148]:=
        walcomps = four3 wsyncoffs[3]

Out[148]=
        {1. + 0. i, (1. + 0. i) e^(2 i n π / 3), (1. + 0. i) e^(4 i n π / 3)}
```

(Produkt von Vektoren ohne Dot-Operator wird in *Mathematica* komponentenweise ausgewertet.)
Zur Visualisierung formen wir in die entsprechenden Sinus- und Cosinusteile um.

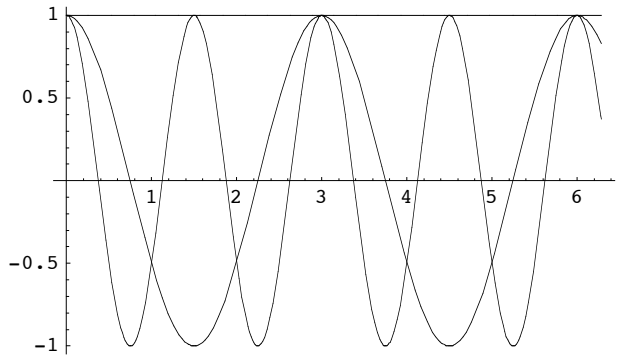
```
In[149]:=
        walsyn = ExpToTrig[walcomps]

Out[149]=
        {1. + 0. i, (1. + 0. i) Cos[2 n π / 3] + (0. + 1. i) Sin[2 n π / 3],
          (1. + 0. i) Cos[4 n π / 3] + (0. + 1. i) Sin[4 n π / 3]}
```

Dies ist wirklich -- bis auf Normalisierung -- die korrekte Fouriersynthese unseres Walzertaktes, *wobei hier zur Veranschaulichung die vollen Sinus- und Cosinuskurven dargestellt werden und nicht nur die Stützpunkte für Integer-Werte.*

Die Realteile ergeben (man führe zur Überprüfung die Summierung "optisch" durch):

In[150]:=

```
Plot[Evaluate[Re[walsyn]], {n, 0, 2 π}]
```



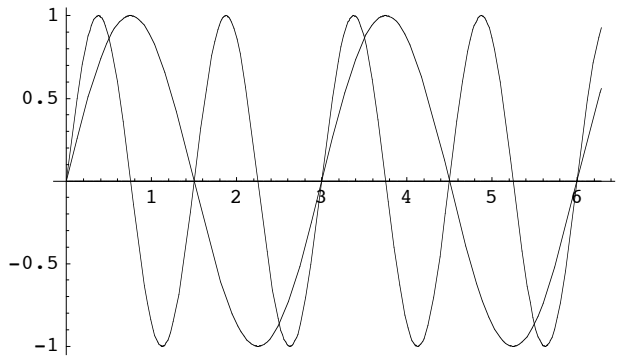
Out[150]=

```
- Graphics -
```

Die Imaginärteile der Synthese heben sich an allen Stützstellen unserer periodischen Funktion (hier: den Integern) auf:

In[152]:=

```
Plot[Evaluate[Im[walsyn]], {n, 0, 2 π}]
```



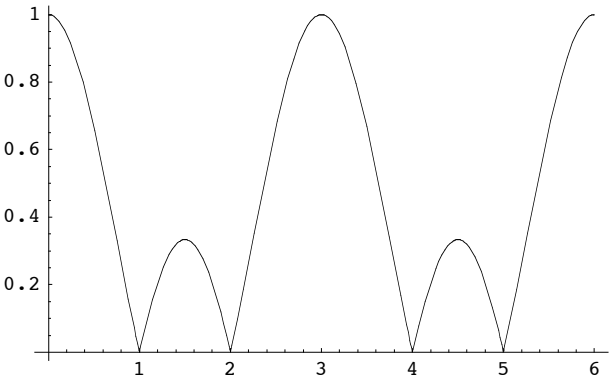
Out[152]=

```
- Graphics -
```

Eine übersichtliche Darstellung liefert der Betrag der Summe der Komponenten der Fouriersynthese; diese Darstellung wird in der Literatur meist angegeben:

In[155]:=

```
Plot[Evaluate[Abs[Apply[Plus, walsyn]]] / 3, {n, 0, 6}]
```



Out[155]=

```
- Graphics -
```

Nun verschieben wir den Walzertakt um einen Tick nach links. Damit erhalten wir eine ungerade Funktion -- es geht die Symmetrie bzgl. der y-Achse verloren.

In[100]:=

```
peri3sh1 = RotateLeft[peri3]
```

Out[100]=

```
{0, 0, 1}
```

Entsprechend hat nun die DFT auch einen imaginären Teil. (Evtl. Hermitische Eigenschaft ...)

In[101]:=

```
four3sh1 = Fourier[peri3sh1]
```

Out[101]=

```
{1. + 0. i, -0.5 + 0.866025 i, -0.5 - 0.866025 i}
```

In[102]:=

```
syn3sh1 = four3sh1 wsyncoffs[3]
```

Out[102]=

$$\left\{1. + 0.\, i,\ (-0.5 + 0.866025\, i)\, e^{\frac{2\, i\, n\, \pi}{3}},\ (-0.5 - 0.866025\, i)\, e^{\frac{4\, i\, n\, \pi}{3}}\right\}$$

In[103]:=

```
ExpToTrig[syn3sh1]
```

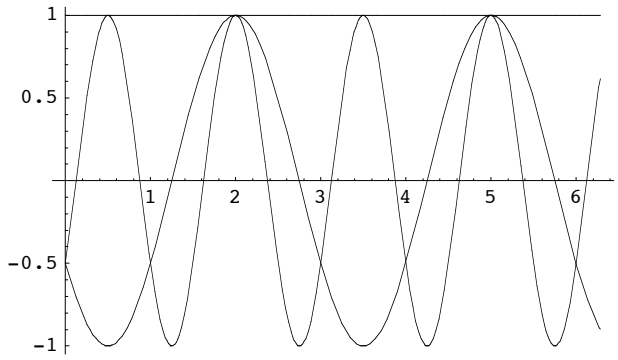
Out[103]=

$$\left\{1. + 0.\, i,\ (-0.5 + 0.866025\, i)\, \text{Cos}\left[\frac{2\, n\, \pi}{3}\right] - (0.866025 + 0.5\, i)\, \text{Sin}\left[\frac{2\, n\, \pi}{3}\right],\right.$$
$$\left.(-0.5 - 0.866025\, i)\, \text{Cos}\left[\frac{4\, n\, \pi}{3}\right] + (0.866025 - 0.5\, i)\, \text{Sin}\left[\frac{4\, n\, \pi}{3}\right]\right\}$$

Wieder ergibt sich bis auf Normalisierung das korrekte Ergebnis.

In[104]:=

```
Plot[Evaluate[Re[syn3sh1]], {n, 0, 2 π}]
```



Out[104]=

```
- Graphics -
```

Zum Abschluß ein etwas komplexeres Beispiel:
Wir überlagern unseren Walzertakt mit einem 4-er-Takt (was übrigens in der Jazz-Musik bisweilen vorkommt). D.h., immer wenn der Index der Zeitreihe durch 3 oder durch 4 teilbar ist, kommt ein Impuls:

In[105]:=

```
peri4x3f = Map[KroneckerDelta[Mod[ #, 3] Mod[#, 4]] &, Range[12] - 1]
```

Out[105]=

```
{1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0}
```

(Nebenbei: *Mathematica* ist sehr ähnlich zu LISP: Man mappt hier eine Funktion auf Elemente einer Liste. Die Funktion wird dabei als anonyme Funktion ohne ausdrückliche Definition angegeben. In LISP entspricht dies (lambda ...) Ausdrücken, hier nutzt man # für ein erstes Argument und & als syntaktische Begrenzung der anonymen Funktion.)
Die weiteren Schritte wie gehabt:

In[106]:=

```
vis43 = coff4x3 wsyncoffs[12]
```

Out[106]=

$$\left\{6.,\ -1.\, e^{\frac{i\,n\,\pi}{6}},\ -1.\, e^{\frac{i\,n\,\pi}{3}},\ 2.\, e^{\frac{i\,n\,\pi}{2}},\ 3.\, e^{\frac{2\,i\,n\,\pi}{3}},\ -1.\, e^{\frac{5\,i\,n\,\pi}{6}},\right.$$
$$\left.2.\, e^{i\,n\,\pi},\ -1.\, e^{\frac{7\,i\,n\,\pi}{6}},\ 3.\, e^{\frac{4\,i\,n\,\pi}{3}},\ 2.\, e^{\frac{3\,i\,n\,\pi}{2}},\ -1.\, e^{\frac{5\,i\,n\,\pi}{3}},\ -1.\, e^{\frac{11\,i\,n\,\pi}{6}}\right\}$$

In[107]:=

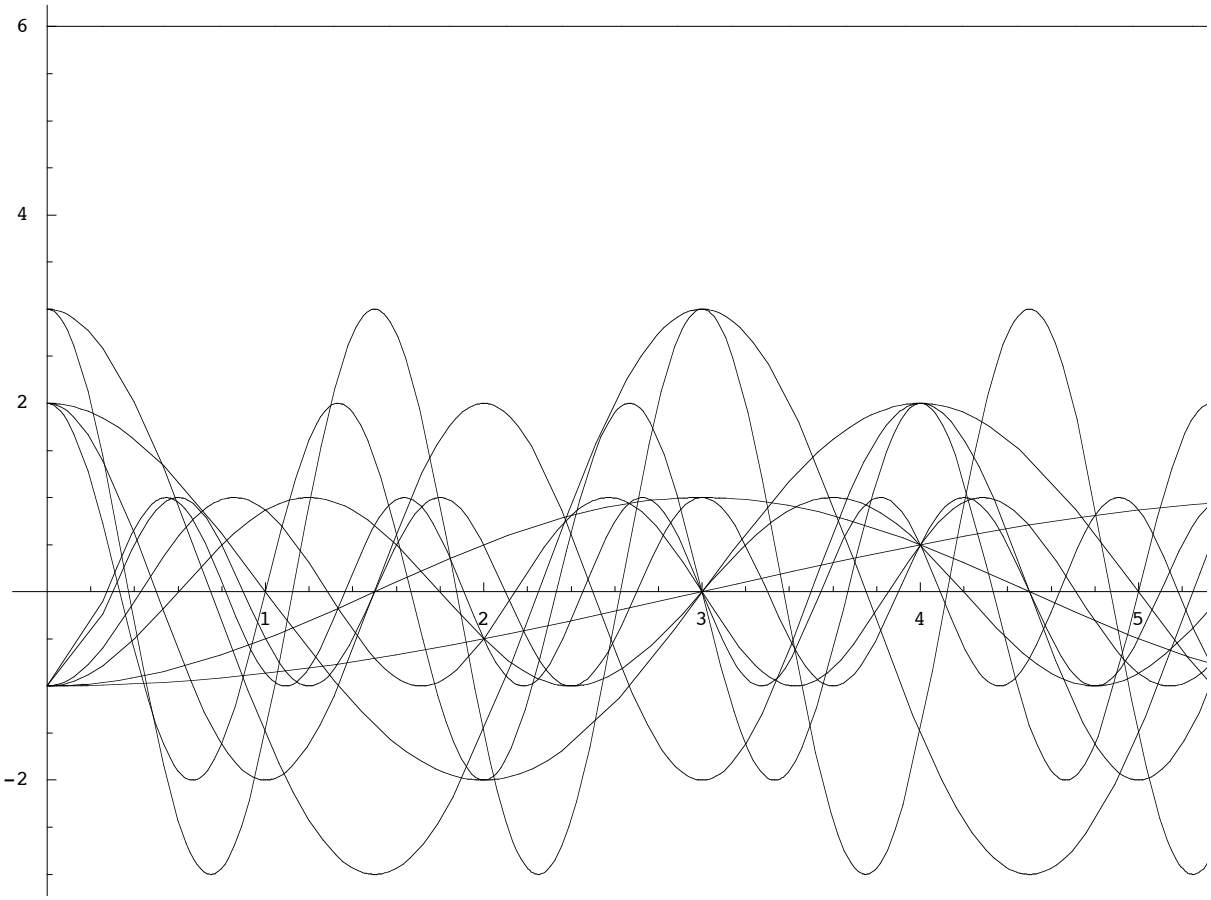
```
vist43 = ExpToTrig[vis43]
```

Out[107]=

$$\left\{6.,\ -1.\,\text{Cos}\left[\frac{n\,\pi}{6}\right] - 1.\,i\,\text{Sin}\left[\frac{n\,\pi}{6}\right],\right.$$
$$-1.\,\text{Cos}\left[\frac{n\,\pi}{3}\right] - 1.\,i\,\text{Sin}\left[\frac{n\,\pi}{3}\right],\ 2.\,\text{Cos}\left[\frac{n\,\pi}{2}\right] + 2.\,i\,\text{Sin}\left[\frac{n\,\pi}{2}\right],$$
$$3.\,\text{Cos}\left[\frac{2\,n\,\pi}{3}\right] + 3.\,i\,\text{Sin}\left[\frac{2\,n\,\pi}{3}\right],\ -1.\,\text{Cos}\left[\frac{5\,n\,\pi}{6}\right] - 1.\,i\,\text{Sin}\left[\frac{5\,n\,\pi}{6}\right],$$
$$2.\,\text{Cos}[n\,\pi] + 2.\,i\,\text{Sin}[n\,\pi],\ -1.\,\text{Cos}\left[\frac{7\,n\,\pi}{6}\right] - 1.\,i\,\text{Sin}\left[\frac{7\,n\,\pi}{6}\right],$$
$$3.\,\text{Cos}\left[\frac{4\,n\,\pi}{3}\right] + 3.\,i\,\text{Sin}\left[\frac{4\,n\,\pi}{3}\right],\ 2.\,\text{Cos}\left[\frac{3\,n\,\pi}{2}\right] + 2.\,i\,\text{Sin}\left[\frac{3\,n\,\pi}{2}\right],$$
$$\left.-1.\,\text{Cos}\left[\frac{5\,n\,\pi}{3}\right] - 1.\,i\,\text{Sin}\left[\frac{5\,n\,\pi}{3}\right],\ -1.\,\text{Cos}\left[\frac{11\,n\,\pi}{6}\right] - 1.\,i\,\text{Sin}\left[\frac{11\,n\,\pi}{6}\right]\right\}$$

In[108]:=

```
Plot[Evaluate[Re[vist43]], {n, 0, 2 π}]
```



Out[108]=

```
- Graphics -
```

Zur Überprüfung des Wellensalats bilden wir die Hüllkurve (envelope):

In[109]:=

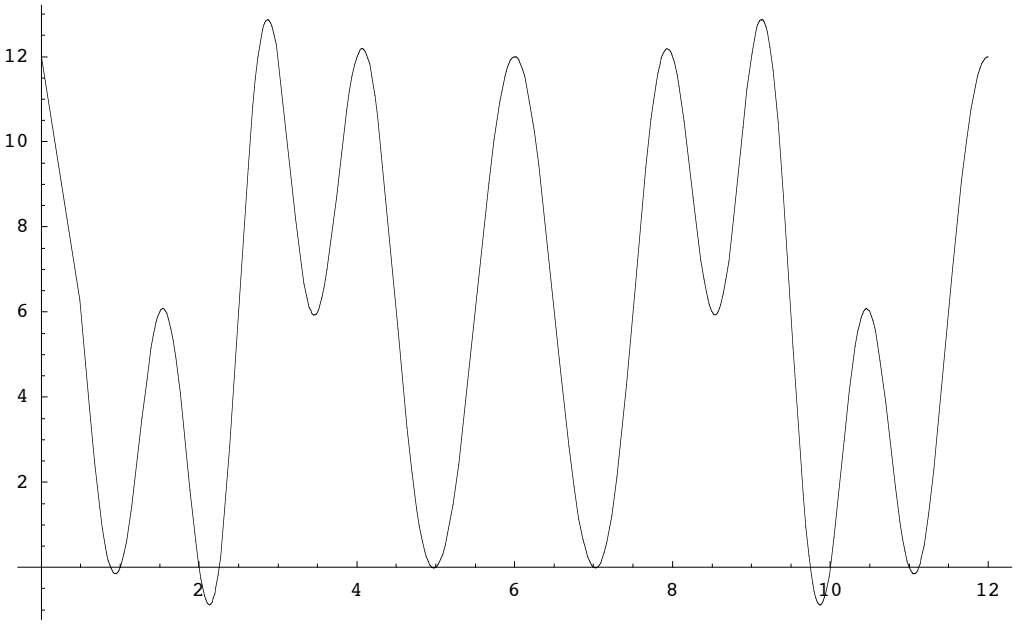
```
env43 = Apply[Plus, Re[vist43]]
```

Out[109]=

$$6. + (1. + 0.\,i)\,\mathrm{Im}\left[\mathrm{Sin}\left[\frac{n\pi}{6}\right]\right] + (1. + 0.\,i)\,\mathrm{Im}\left[\mathrm{Sin}\left[\frac{n\pi}{3}\right]\right] -$$
$$(2. + 0.\,i)\,\mathrm{Im}\left[\mathrm{Sin}\left[\frac{n\pi}{2}\right]\right] - (3. + 0.\,i)\,\mathrm{Im}\left[\mathrm{Sin}\left[\frac{2n\pi}{3}\right]\right] + (1. + 0.\,i)\,\mathrm{Im}\left[\mathrm{Sin}\left[\frac{5n\pi}{6}\right]\right] -$$
$$(2. + 0.\,i)\,\mathrm{Im}[\mathrm{Sin}[n\pi]] + (1. + 0.\,i)\,\mathrm{Im}\left[\mathrm{Sin}\left[\frac{7n\pi}{6}\right]\right] - (3. + 0.\,i)\,\mathrm{Im}\left[\mathrm{Sin}\left[\frac{4n\pi}{3}\right]\right] -$$
$$(2. + 0.\,i)\,\mathrm{Im}\left[\mathrm{Sin}\left[\frac{3n\pi}{2}\right]\right] + (1. + 0.\,i)\,\mathrm{Im}\left[\mathrm{Sin}\left[\frac{5n\pi}{3}\right]\right] + (1. + 0.\,i)\,\mathrm{Im}\left[\mathrm{Sin}\left[\frac{11n\pi}{6}\right]\right] -$$
$$1.\,\mathrm{Re}\left[\mathrm{Cos}\left[\frac{n\pi}{6}\right]\right] - 1.\,\mathrm{Re}\left[\mathrm{Cos}\left[\frac{n\pi}{3}\right]\right] + 2.\,\mathrm{Re}\left[\mathrm{Cos}\left[\frac{n\pi}{2}\right]\right] + 3.\,\mathrm{Re}\left[\mathrm{Cos}\left[\frac{2n\pi}{3}\right]\right] -$$
$$1.\,\mathrm{Re}\left[\mathrm{Cos}\left[\frac{5n\pi}{6}\right]\right] + 2.\,\mathrm{Re}[\mathrm{Cos}[n\pi]] - 1.\,\mathrm{Re}\left[\mathrm{Cos}\left[\frac{7n\pi}{6}\right]\right] + 3.\,\mathrm{Re}\left[\mathrm{Cos}\left[\frac{4n\pi}{3}\right]\right] +$$
$$2.\,\mathrm{Re}\left[\mathrm{Cos}\left[\frac{3n\pi}{2}\right]\right] - 1.\,\mathrm{Re}\left[\mathrm{Cos}\left[\frac{5n\pi}{3}\right]\right] - 1.\,\mathrm{Re}\left[\mathrm{Cos}\left[\frac{11n\pi}{6}\right]\right]$$

In[110]:=

```
Plot[Evaluate[Re[env43]], {n, 0, 12}]
```



Out[110]=

```
- Graphics -
```

Man sieht, wie die Synthesefunktion die zu realisierenden Impulswerte gleichsam ansteuert.

Berechnung der DFT: Fast Fourier Transform

Wir notieren noch einmal zusammenfassend die Gleichungen der DFT:

In[112]:=

$$\text{Xana}[k\_] := \sum_{n=0}^{N-1} x[n]\, e^{-2\pi i n k/N}$$

Die Berechnung für alle k erfordert O(N^2) komplexe Multiplikationen und komplexe Additionen.

In[113]:=

$$\text{xsyn}[n\_] := \frac{1}{N} \sum_{k=0}^{N-1} X[k]\, e^{2\pi i n k/N}$$

Für die verwendeten Fourierkoeffizienten hat sich folgende Notation eingebürgert:

In[114]:=

$$W[N\_, k\_, n\_] := e^{-2\pi i n k/N}$$

In[115]:=

```
W[7, 1, 3]
```

Out[115]=

$$e^{-\frac{6 i \pi}{7}}$$

In[116]:=
```
N[%]
```

Out[116]=
```
-0.900969 - 0.433884 i
```

Wie bereits der graphischen Darstellung der F-koeff. auf dem Einheitskreis zu entnehmen, gibt es hier mehrere Redundanzen.
Zunächst die offenkundige Periodizität über N:

In[117]:=
```
Simplify[W[N, k, n + N] / W[N, k, n], N ∈ Integers && k ∈ Integers && n ∈ Integers]
```

Out[117]=
```
1
```

Dann die Eigenschaft, daß Quadrate im Koeffizientenvektor von N/2 vorkommen:

In[118]:=
```
W[N, k, n]^2 == W[N / 2, k, n]
```

Out[118]=
```
True
```

Diese Eigenschaft wird im Fast Fourier Transform genutzt:

$X[k] = \sum_{r=0}^{N/2-1} x[2\,r]\, W[N,\, k,\, 2\,r] + \sum_{r=0}^{N/2-1} x[2\,r+1]\, W[N,\, k,\, 2\,r+1]$

bzw.
$X[k] = \sum_{r=0}^{N/2-1} x[2\,r]\, W[N/2,\, k,\, r] + W[N,\, k,\, 1] \sum_{r=0}^{N/2-1} x[2\,r+1]\, W[N/2,\, k,\, r]$

denn

In[119]:=
```
W[N, k, 2 r + 1] == W[N, k, 1] W[N / 2, k, r]
```

Out[119]=

$$e^{-\frac{2\,i\,k\,\pi\,(1+2\,r)}{N}} == e^{-\frac{2\,i\,k\,\pi}{N}-\frac{4\,i\,k\,\pi\,r}{N}}$$

```
In[120]:=
        Simplify[%]

Out[120]=
        True
```

Folglich läßt sich die DFT einer Sequenz geradzahliger Länge N als Summe von zwei DFT von Sequenzen der Länge N/2 darstellen.
Berechnungsaufwand N+2(N/2)^2 (Summand N ergibt sich aus Multiplikation mit Koeffizient W[N,k,1] für ungerade Sequenzglieder und N-maliger Addition zur Summe für gerade Sequenzglieder

Ist N eine Potenz von 2, so läßt sich dies $\text{Log}_2\, N$-mal durchführen, bis man nur noch DFT über Sequenzen mit jeweils 2 Positionen hat.
Wir erhalten N + 2(N/2 +2(N/4)^2) usf, das macht insgesamt O(N log_2 N).

Lineare Systeme.

Jede Zeitreihe (d.h. immer mit diskretem Index) kann als Summe gewichteter Unit Samples dargestellt werden:

x[n]=$\sum_{k=-\infty}^{\infty} x[k]$ KroneckerDelta[$n-k$]

Diese -- zunächst nicht besonders vielsagende -- Darstellung wird interessant, wenn wir eine Zeitreihe durch ein lineares System T linear transformieren, d.h. T(ax[n])=aT(x[n]) und T(x_1[n]+x_2[n])=T(x_1[n])+T(x_2[n]). Aus der Linearität von T folgt dann

y[n] = T(x[n])=$\sum_{k=-\infty}^{\infty} x[k]\, T$(KroneckerDelta[$n-k$]) = $\sum_{k=-\infty}^{\infty} x[k] h_k$[n-k],

wobei h_k[n] die Antwort des Systems auf einen Impuls (Unit Sample) zum Zeitpunkt k darstellt -- die sog. Impulsantwort (Bsp: Vibration einer Lautsprechermembran nach kurzem Baßton).
Wenn nun die Impulsantwort von der Zeit unabhängig ist, sprechen wir von einem verschiebungsinvarianten linearen System (auch LTI-System). Dieses ist gekennzeichnet durch die Gleichung

y[n]=$\sum_{k=-\infty}^{\infty} x[k] h$[n-k],

dies wird als Konvolution oder Faltung des Eingangssignals mit der Impulsantwort bezeichnet; dieser Operator ist kommutativ, daher ist die übliche Notation y[n]=x[n]*h[n] gerechtfertigt.

Es ist interessant, die Eigenschaften der Faltung anzusehen, wenn sich die Impulsantworten auf mehrere Eingangsimpulse überlagern (im Bsp.: wenn die Nachschwingung der Lautsprechermembran noch andauert, wenn der nächste Baßton kommt).

Das betrachten wir mit einfachen Listen von algebraischen Variablen. Summation und Multiplikation ersetzen wir zur Veranschaulichung durch Listenbildung. Wir erhalten für jeden Zeitpunkt (äußerste Liste) eine Liste (für die Summation) von Listen (jeweils Produkt betreffendes Eingabesignal mit relevantem Punkt der Impulsantwort).

Möge {d,e,f} für die x(n), {a,b,c} für die Impulsantwort h(k) stehen (wobei wir die o.a. Faltungsformel in der kommutierten Variante benutzen).

In[121]:=

```
ListConvolve[{a, b, c}, {d, e, f}, {1, -1}, , List, List]
```

Out[121]=

```
{{{c, Null}, {b, Null}, {a, d}}, {{c, Null}, {b, d}, {a, e}},
 {{c, d}, {b, e}, {a, f}}, {{c, e}, {b, f}, {a, Null}}, {{c, f}, {b, Null}, {a, Null}}}
```

etwas lesbarer ...

```
In[122]:=
        % // MatrixForm
```

```
Out[122]//MatrixForm=
        ⎛ ⎛ c    ⎞  ⎛ b    ⎞  ⎛ a    ⎞ ⎞
        ⎜ ⎝ Null ⎠  ⎝ Null ⎠  ⎝ d    ⎠ ⎟
        ⎜ ⎛ c    ⎞  ⎛ b    ⎞  ⎛ a    ⎞ ⎟
        ⎜ ⎝ Null ⎠  ⎝ d    ⎠  ⎝ e    ⎠ ⎟
        ⎜ ⎛ c    ⎞  ⎛ b    ⎞  ⎛ a    ⎞ ⎟
        ⎜ ⎝ d    ⎠  ⎝ e    ⎠  ⎝ f    ⎠ ⎟
        ⎜ ⎛ c    ⎞  ⎛ b    ⎞  ⎛ a    ⎞ ⎟
        ⎜ ⎝ e    ⎠  ⎝ f    ⎠  ⎝ Null ⎠ ⎟
        ⎜ ⎛ c    ⎞  ⎛ b    ⎞  ⎛ a    ⎞ ⎟
        ⎝ ⎝ f    ⎠  ⎝ Null ⎠  ⎝ Null ⎠ ⎠
```

In dieser Matrix stehen die Zeilen für n, die Spalten rückwärtig für k. Man sieht, wie die Impulsantwort an den verschiedenen Punkten von x ansetzt (3. Spalte) und im nächsten Zeitpunkt (in der Spalte nach links vorrückend) weiter wirkt. Liest man das Ergebnis zeilenweise, wird deutlich, warum die Impulsantwort in der Faltungsformel zeitlich gespiegelt auftritt -- es handelt sich sozusagen um aufgesammelte Nachwirkungen der vorausgehenden Impulsantworten.

Konvolutionstheorem

Das Konvolutionstheorem begründet, warum die Darstellung in der Frequenzdomäne für LTI-Systeme wichtige Eigenschaften dieser Systeme effizient zu berechnen erlaubt. Verbal ausgedrückt:

Der Konvolution (Faltung) von Eingabezeitreihe und Impulsantwort in der Zeitdomäne entspricht die Multiplikation von Fourier-Transformierter Eingabezeitreihe und Frequenzantwort in der Frequenzdomäne.

Formal gilt für ein LTI $y[n]=\sum_{k=-\infty}^{\infty} x[k]h[n-k]$,

Der Beweis des Konvolutionstheorems läuft wird mithilfe einer Indexsubstitution über die bei der Faltung und bei der Fouriertransformation beteiligten unendlichen Summen geführt (s. z.B. Huang, Kapitel 4)

Cepstrum

Die Theorie linearer Systeme wird in der Spracherkennung i.B. für die Definition von Parametern benutzt, die das Sprachsignal geeignet beschreiben können. Ausgangspunkt ist die Überlegung, daß das Sprachsignal für eine phonetische Einheit jeweils durch Konvolution eines Anregungssignals mit einer Resonanz im Vokaltrakt beschrieben werden kann. Bei Vokalen ist das Anregungssignal durch die periodische Schwingung der Stimmbänder gegeben, bei stimmlosen Konsonanten durch einen aperiodischen, geräuschhaften Vorgang (z.B. Pressen der Schneidezähne gegen die Lippe etc.). Bei stimmhaften Konsonanten sind sowohl ein periodisches als auch ein aperiodisches Anregungssignal beteiligt. Die Resonanzen im Vokaltrakt lassen sich für kurze Segmente des Sprachsignals (ohne Veränderung der Zungenstellung oder Mundöffnung) als verschiebungsinvariante Impulsantworten modellieren (da hier nur Resonanzen und keine neue Impulserzeugung beteiligt ist, entspricht dies den sog. all-pole Filtern in der Theorie linearer Filter, s. Huang und Oppenheim & Schafer).

Aus dieser Überlegung resultiert, daß eine Darstellung kurzer Ausschnitte des Sprachsignals durch eine Trennung von Anregungsvorgang und Impulsantwort für die Spracherkennung vorteilhaft ist. Dazu wird

eine Dekonvolution von Anregungsvorgang und Impulsantwort benötigt. Diese läßt sich mithilfe der sog. homomorphen Signalverarbeitung erreichen: Nach dem Konvolutionstheorem entspricht der Konvolution in der Zeitdomäne eine Multiplikation in der Frequenzdomäne. Bildet man nun in der Frequenzdomäne den Logarithmus, so erhält man eine Trennung von Anregung und Resonanz als Summanden. Dies läßt sich sowohl durch Anwendung des komplexen Logarithmus als auch durch Anwendung des reellen Logarithmus auf den Betrag der Fouriertransformierten erreichen. (Anm.: Bei Nutzung des komplexen Logarithmus sind wegen der Nichteindeutigkeit des Arg-Anteils zusätzliche Glättungen zu berechnen.) Allerdings sind die erhaltenen Summanden im Frequenzbereich nicht eindeutig der Anregung oder einer Vokaltraktresonanz zuzuordnen. Um dies zu erreichen, wendet man nun die inverse Fouriertransformation an, um in die Zeitdomäne zurückzukommen. Dort muß ja das Anregungssignal der Resonanz zeitlich vorausgehen. Das Ergebnis dieser Rücktransformation heißt -- je nach verwendetem Logarithmus reales oder komplexes -- Cepstrum (s. Huang et al.).

Abschließend veranschaulichen wir die Cepstralanalyse an einem Beispiel. Hier wird ein Ausschnitt aus einem Sprachsignal (Hammingfenster) bearbeitet. Die ersten Cepstralkoeffizienten nach einem anfänglichen Peak geben den Anregungsvorgang wieder.

In[123]:=
```
UnitStepTo[x_, y_] := UnitStep[x] - UnitStep[x - y]
```

In[124]:=
```
GeneralHamming[α_, n_, N_] := (1 - α) UnitStepTo[n, N] - α UnitStepTo[n, N] Cos[2 π n / N]
```

Nun analysieren wir ein isoliertes gesprochenes "a".

In[125]:=
```
adata = Import["/Users/marcus/speech\ sounds/speech_by_protools/(l)a(nge)", "AIFF"]
```

Out[125]=
```
- Sound -
```

In[126]:=

```
Show[adata]
```

Out[126]=

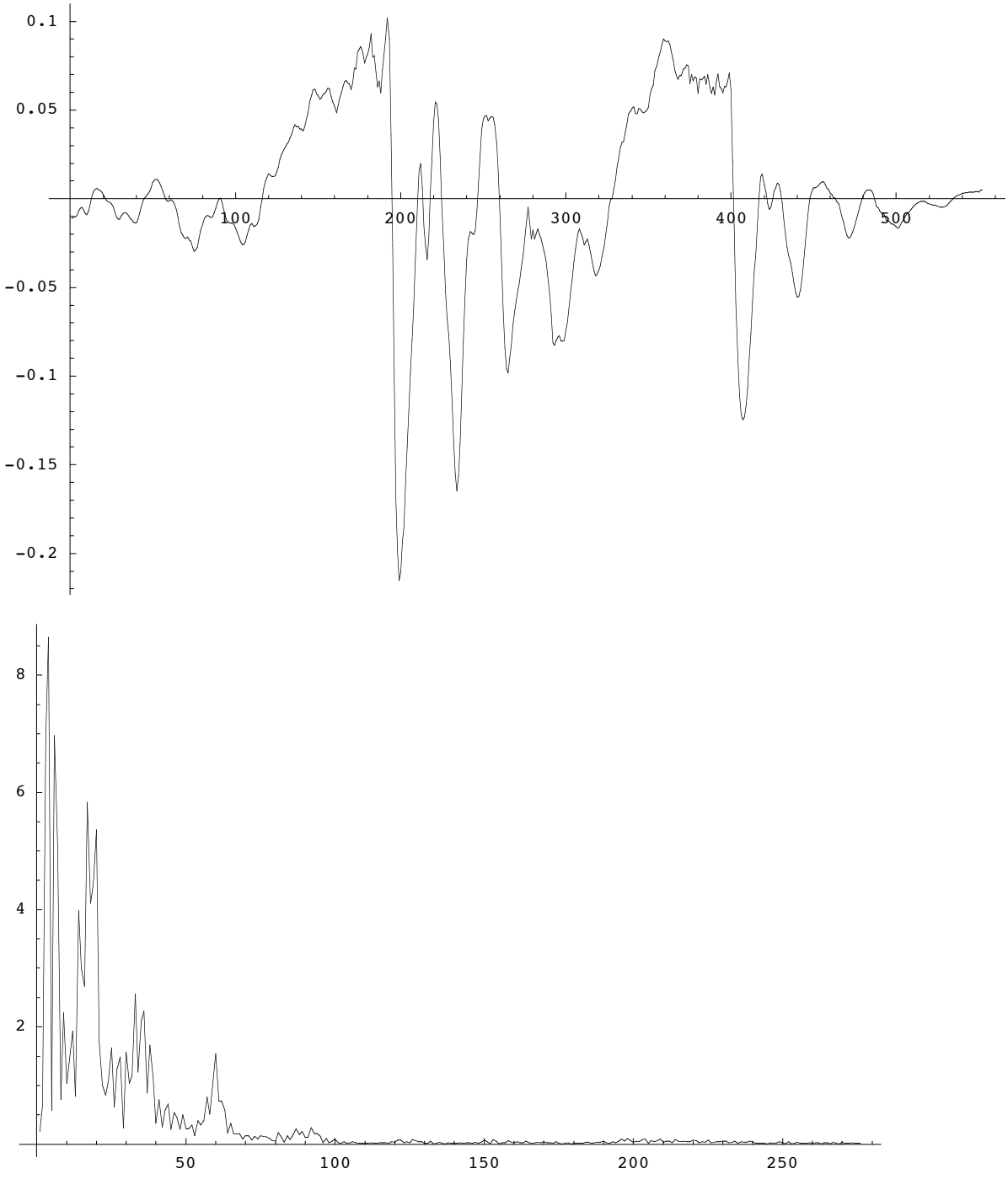
```
- Sound -
```

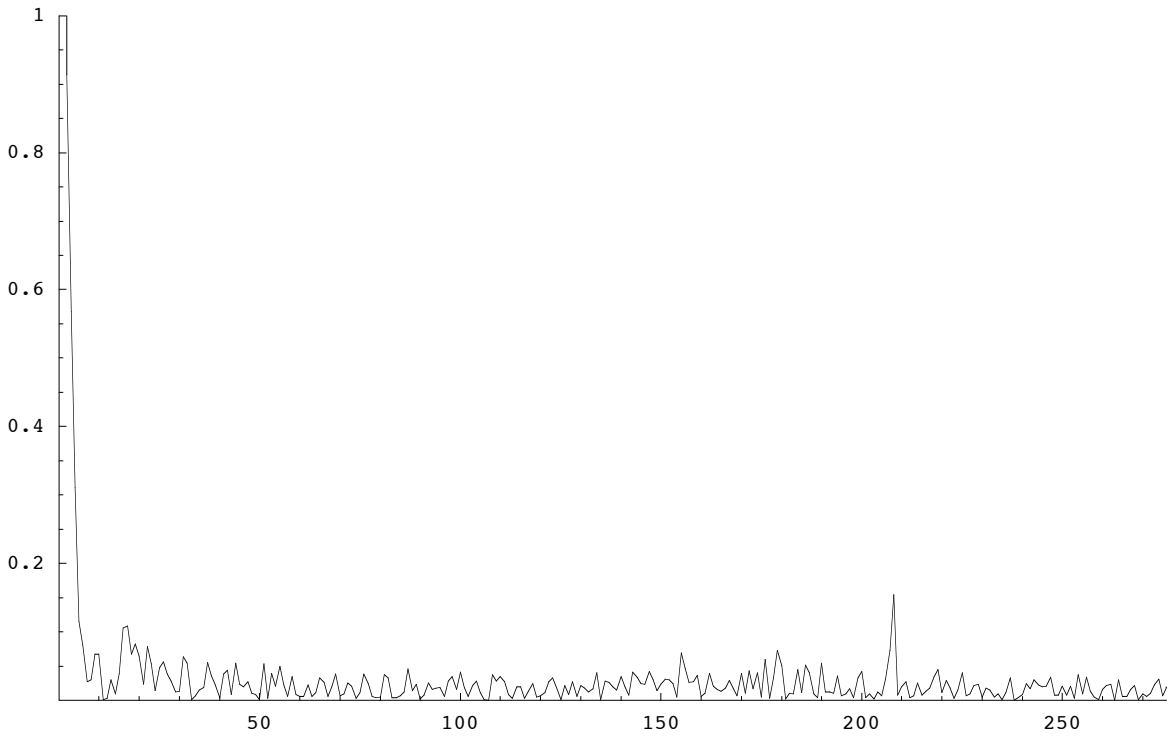
In[140]:=

```
speechAnalyze[sounddata_] :=
 With[{data = sounddata},
  inputLen = Length[data[[1, 1]]];
  sampleRate = data[[1, 2, 1]];
  wwidth = Ceiling[inputLen / 2];
  windowWidth = 0.025;
  windowPoints = Ceiling[sampleRate windowWidth];
  hammingwindowedData = Take[data[[1, 1]], {100, 100 + windowPoints - 1}]
    Table[GeneralHamming[.46, n, windowPoints], {n, 0, windowPoints - 1}];
  ListPlot[hammingwindowedData, PlotJoined → True, PlotRange → All];
  fou = Take[Fourier[hammingwindowedData], Ceiling[windowPoints / 2]];
  ListPlot[Abs[fou], PlotJoined → True, PlotRange → All];
  ceps = InverseFourier[Log[Abs[Fourier[hammingwindowedData]]]];
  ListPlot[Abs[ceps], PlotJoined → True,
   PlotRange → {{0, Ceiling[windowPoints / 2]}, {0, 1.}}]
 ]
```

In[141]:=

```
speechAnalyze[adata]
```


0.1
0.05
-0.05
-0.1
-0.15
-0.2
100
200
300
400
500
8
6
4
2
50
100
150
200
250



Out[141]=

- Graphics -

Der isolierte Peak rechts in diesem Cepstrum entspricht einer Periodenlänge der Grundfrequenz. Nun ein Beispiel eines stimmlosen Sprachabschnitts.

In[142]:=

```
chdata = Import["/Users/marcus/speech\ sounds/speech_by_protools/(i)ch", "AIFF"]
```

Out[142]=

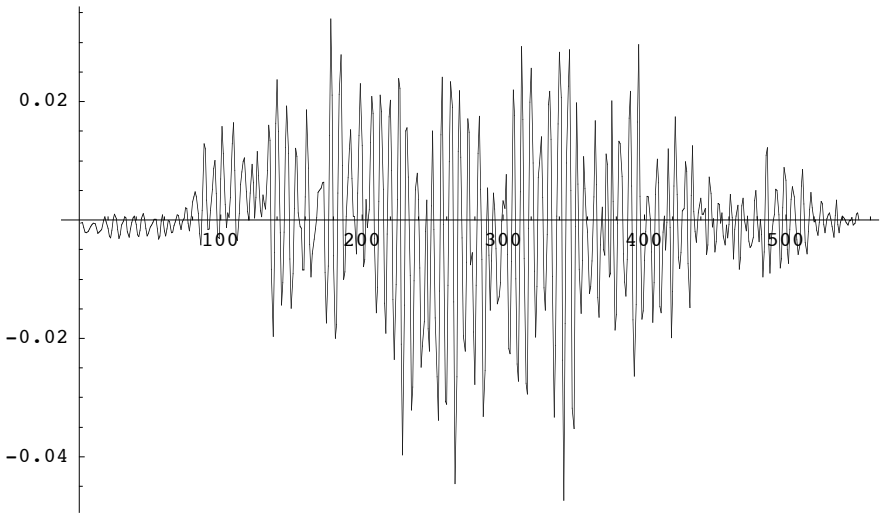
- Sound -

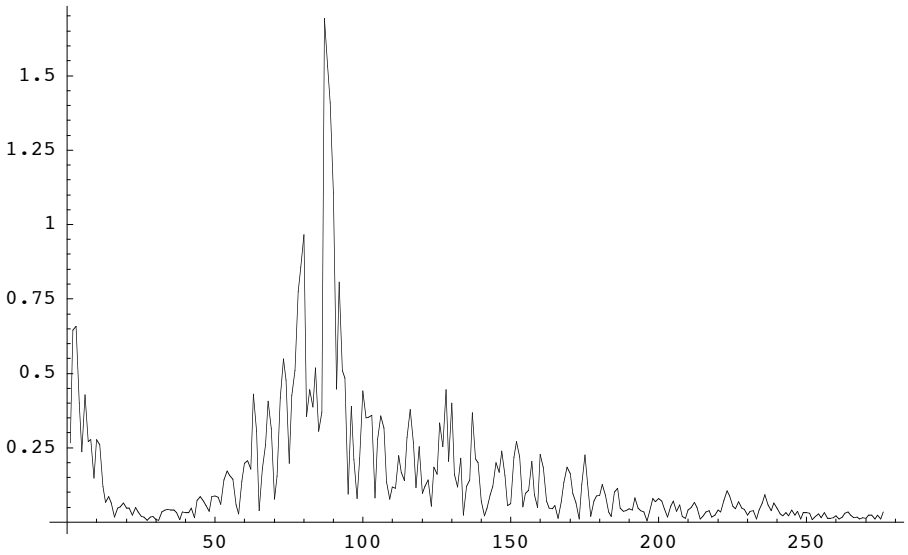
```
Show[chdata]
```

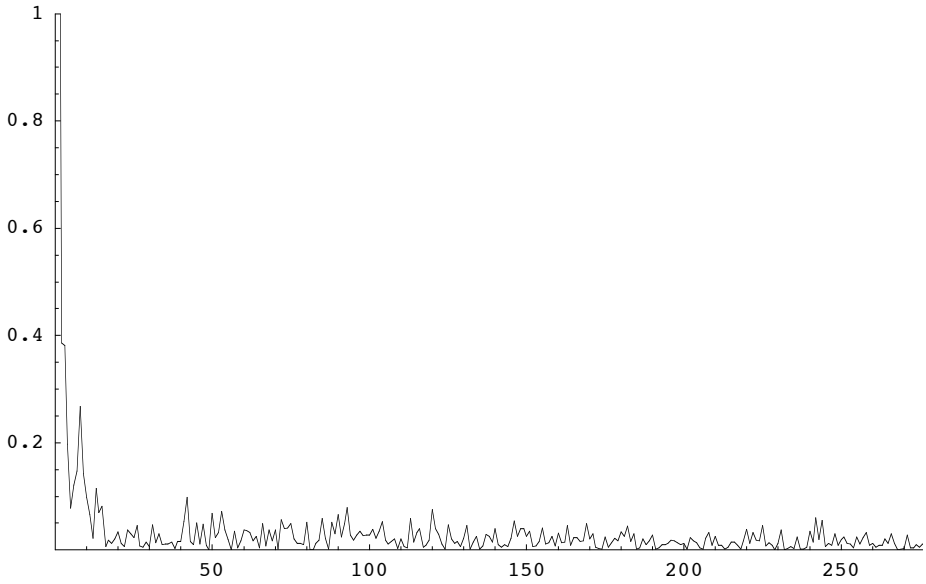
- Sound -

In[143]:=

```
speechAnalyze[chdata]
```


0.02
-0.02
-0.04
100
200
300
400
500


1.5
1.25
1
0.75
0.5
0.25
50
100
150
200
250


1
0.8
0.6
0.4
0.2
50
100
150
200
250

```
Out[143]=
        - Graphics -
```

Die weiteren Einzelheiten zur Nutzung von auditiven Filtern und zur Bildung von Merkmalsvektoren aus Cepstralkoeffizienten entnehme man der Dokumentation des Sphinx-Systems der Carnegie Mellon University (s. Referenzliste auf unserer Webseite).