wdv-notes 144
Stand: 15.OKT.1991 (2.)
Wiss.Datenverarbeitung
FREIE UNIVERSITÄT BERLIN

Mathematica: Sammlung von WDV-Prozeduren – Teil 5.

© 1990–1993 *Karl-Heinz Dittberner*

Pro

Das wissenschaftliche Programmier-System ***Mathematica*** der Firma Wolfram Research, USA (wdv-notes Nr. 71) hat sich inzwischen auf allen Computer-Plattformen – auch auf dem PC – als ein anspruchsvolles Werkzeug zur mathematischen Behandlung theoretischer und numerischer Probleme durchgesetzt.

In diesem Merkblatt werden vollständige Prozeduren und Algorithmen in ***Mathematica*** V 2.0 dargestellt, die von der Abt. Wiss. Datenverarbeitung seit 1990 im Rahmen anderer Fragestellungen entwickelt worden sind. Sämtliche Prozeduren sind sorgfältig getestet und mit Beispielen versehen.

22. Das obskure Window (Vorfilter) „Cosinus-Taper“:

Die Verwendung des Windows (Vorfilter) „Cosinus-Taper“ – auch als Tukey-Window bekannt – ist bei der diskreten („windowed“) Fourier-Transformation von Signalen (wFFT) sehr beliebt, da es (vermeintlich) einen „sanften“ und „glatten“, das Signal wenig beeinflußenden, Übergang an den Rändern (edges) des Signals bewirkt und außerdem leicht berechenbar ist. Das meinen viele Wissenschaftler, beachten dabei aber nicht, daß dieses Window-Filter erhebliche Artefakte in das zu analysierende Signal „einschleppen“ kann.

Mit der *Mathematica*-Package ***SignalProcessing*** (wdv-notes Nr. 139) ist heute in sehr kurzer Zeit eine genaue und aussagekräftige Analyse eines solchen Filters möglich. Dazu wird zunächst mit der Prozedur **CosinusTaper** das Filter im Zeitbereich definiert. Die Analyse zeigt dann sehr deutlich, warum das Window „Cosinus-Taper“ in der seriösen Signaldatenverarbeitung keine Verwendung finden sollte.

In[4]:=

```
<<SignalProcessing.m          (* Laden des Pakets. *)
Off  [General::spell1];
$DefaultFont = {"Helvetica", 10 (*pt*)};
```

. . .

In[8]:=

```
Begin["`Private`"];
Off  [General::spell1];

CosinusTaper [ x_, alpha_, width_:1 ] :=
  Module [ { edge, start, center, shift },
      edge   =  alpha  width /2;
      start  =  width /2 – edge;
    center   =  width – 2 edge;
      shift  =  center /2;
   window  :=  (*  Aus 3 Abschnitten:  *)
     CPulse [  edge,  x + width /2 ] *
               (1 + Cos [ Pi (x + shift) /edge ] ) /2
    + CPulse [  center,  x + start ]
    + CPulse [  edge,    x – start ] *
               (1 + Cos [ –Pi (x – shift) /edge ] ) /2;
  Return [ window ]   ]
End[ ];  (* Private *)
```

```
Support module has been loaded.
The knowledge base of signal processing operators
has been loaded.
```

Beispiel:

Mit dem Programm **ASPAnalyze** der Package ***SignalProcessing*** wird das Vorfilter „Cosinus-Taper“ für verschiedene Parameter α untersucht (α = 25 %, 50 %, 75 %, 100 %), wobei α den Cosinus-Anteil der gesamten Filterbreite *width* angibt (α = 2 *edge/width)*. Die Analyse des Falls α = *25 %* wird im folgenden ausführlich dargestellt.

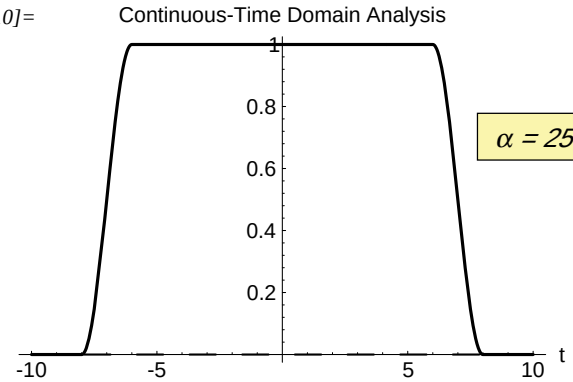
Es gilt folgende Notation: I = Sqrt (–1) = j, E = Eulersche Zahl *e*, *w* = Kreisfrequenz $\omega = 2\,\pi\,f$, *s* = komplexe Frequenz = τ + j ω, CPulse ist die Einheitsimpulsfunktion und *t* = Zeit.

In[10]:=

```
Begin["`Private`"];
ASPAnalyze [ CosinusTaper [ t, 1/4, 16], t, –10, 10 ]
```

. . .

Out[10]=




```
((1 + Cos[(Pi (-6 + t))/2]) CPulse   [-6 + t])/2
                                 2
 + ((1 + Cos[(Pi (6 + t))/2]) CPulse   [8 + t])/2
                                    2
 + CPulse   [6 + t]
         12
has the following Laplace transform:

        2 s    14 s    16 s    2
((-1 - E    + E     + E    ) Pi )/

  8 s     3         2
(E    (8 s  + 2 s Pi ))

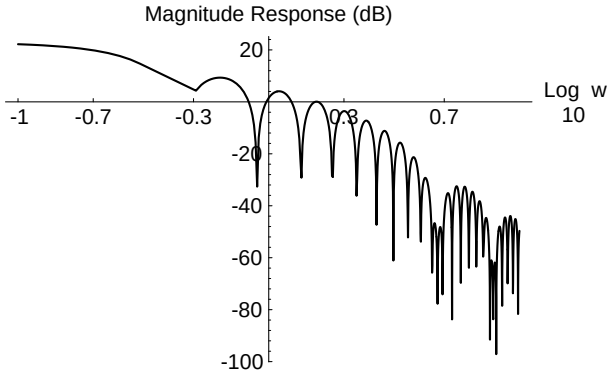
The region of convergence is: 0 < Re(s) < Infinity

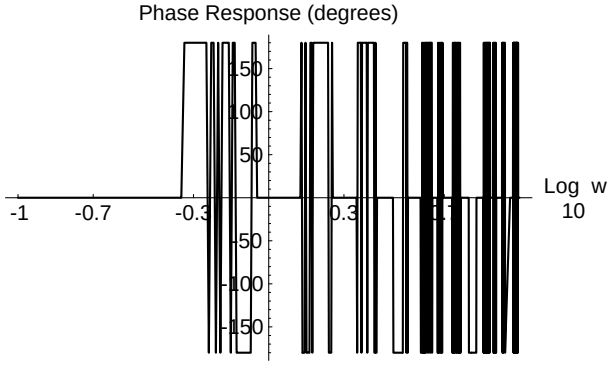
The system is not stable.

The signal has the following frequency response:

  -8 I w        2 I w    14 I w    16 I w    2
(E       (-1 - E      + E       + E       ) Pi )/

       3           2
(-8 I w  + 2 I w Pi )
```






```
PoleZeroPlot::notrational: Transform is not a ratio-
   nal polynomial. A pole-zero plot cannot be
   generated.
```

```
                    2 s    14 s    16 s    2
              (-1 - E    + E     + E    ) Pi
   LTransData[-------------------------------,
                    8 s       2        3
                   E    (2 Pi  s + 8 s )

   Rminus[0], Rplus[Infinity], LVariables[s]]
```

Ergebnis:

Im Amplitudenspektrum (magnitude response) erkennt man, daß das Vorfilter im Frequenzbereich sehr viele Seitenbänder (sidelobes) aufweist, wovon sich das erste *nur* –12,0 dB (Amplitudenverhältnis 1:4) unterhalb des Durchlaßbereichs des Tiefpaß befindet. Bis –40 dB (Amplitudenverhältnis 1:100) weist das Window-Filter sogar insgesamt 7 störende Seitenbänder auf. Damit verbunden ist ein sehr komplexer („unschöner") Verlauf des Phasenspektrums (phase response).

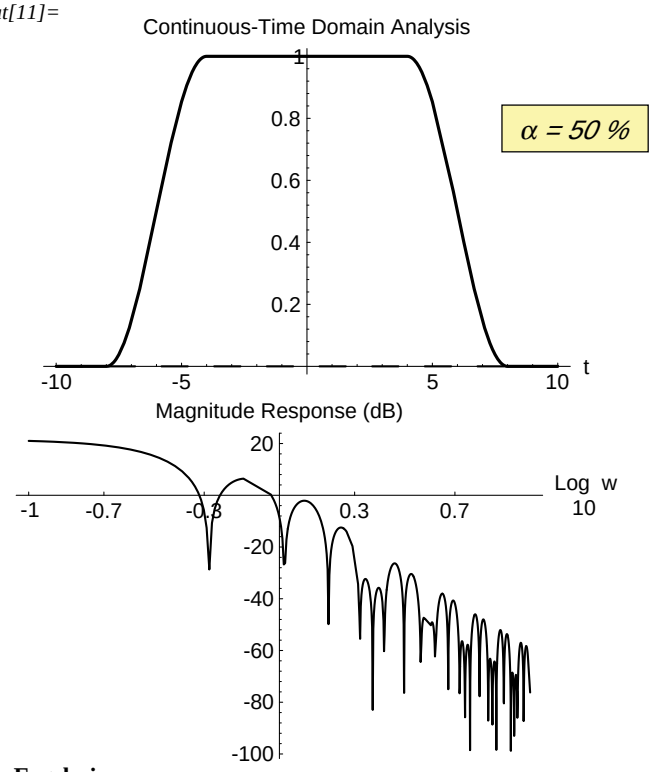
Verwendet man z. B. ein solches „Cosinus-Taper"-Filter im Zusammenhang mit der Berechnung des Powerspektrums eines Signals, dann sind die Seitenbänder des Vorfilters aufgrund der Produktbildung vollständig im berechneten Powerspektrum als verfälschende Komponenten enthalten.

Beispiel:

In[11]:=

ASPAnalyze [CosinusTaper [t, 1/2, 16], t, –10, 10]

Out[11]=



Ergebnis:

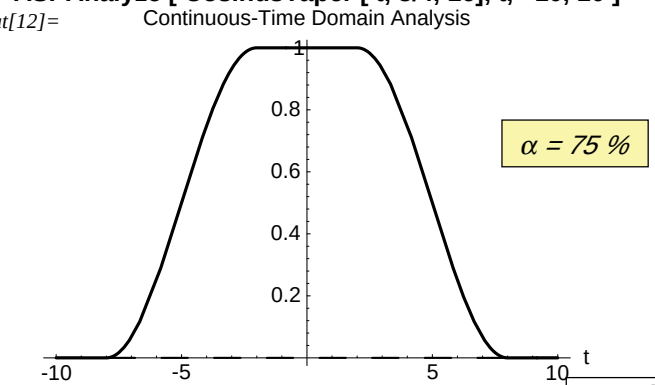
Mit steigendem Cosinus-Anteil α verbessert sich der Störabstand zu den Seitenbändern und beträgt –14,2 dB zum ersten Seitenband bei α = 50 %.

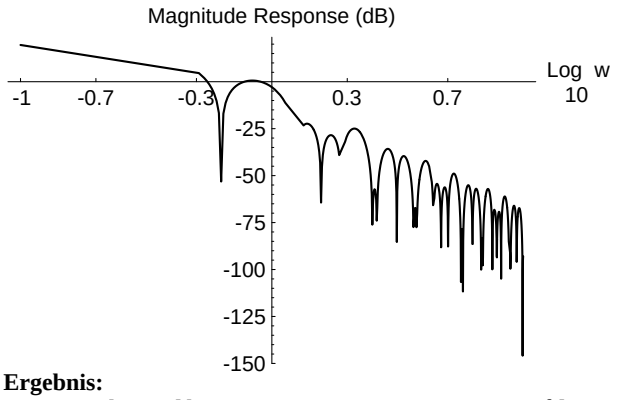
Beispiel:

In[12]:=

ASPAnalyze [CosinusTaper [t, 3/4, 16], t, –10, 10]

Out[12]=





Ergebnis:

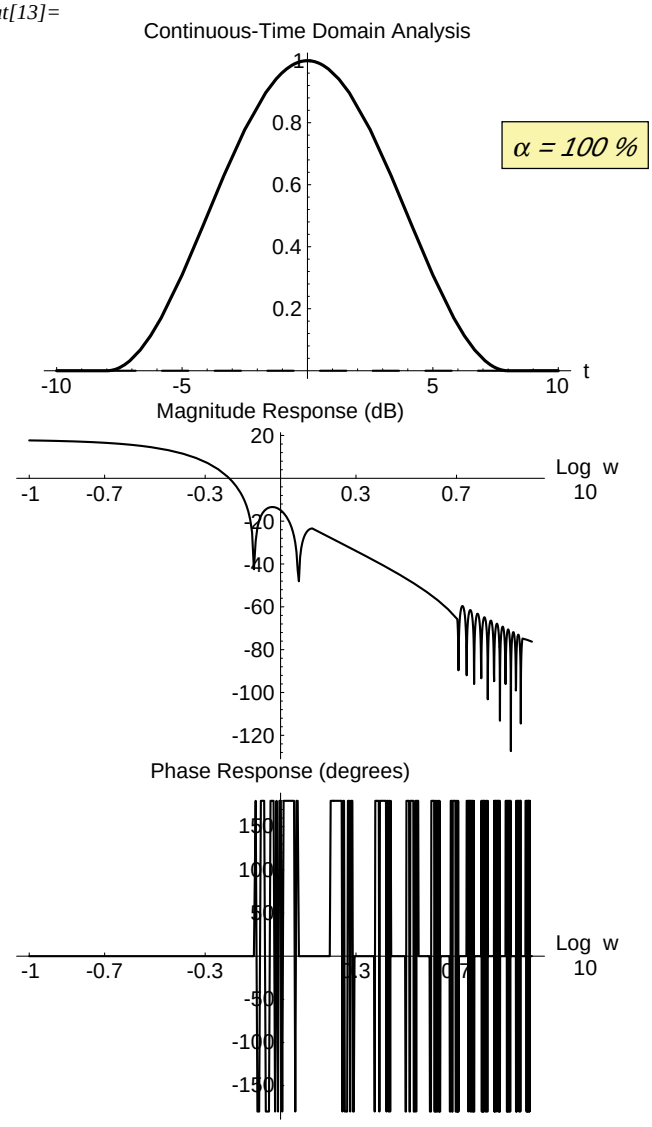
Der Störabstand bei einem „Cosinus-Taper"-Vorfilter mit α = 75 % beträgt bereits –18,7 dB, was auch noch nicht für eine solide Verwendungsmöglichkeit spricht.

Beispiel:

In[13]:=

ASPAnalyze [CosinusTaper [t, 1, 16], t, –10, 10]

Out[13]=



Ergebnis:

Erst bei einem Cosinus-Anteil von α = 100 %, also bereits bei einem reinen Cosinus-Window vom Hann-Typ, wird ein Störabstand von –30,9 dB (Amplitudenverhältnis 1:35) erreicht, was für manche Anwendungen ausreichen kann.

Zusammenfassung:

Das **ASPAnalyze** hat es bereits mitgeteilt: Beim Window „Cosinus-Taper" handelt es sich (systemtheoretisch) um kein stabiles Signal. *Fazit*: „**Never use it!**" – Es könnte nämlich sogar eine Periodizität in einem Signal vorgetäuscht werden.