wdv-notes 143
Stand: 14.OKT.1992 (2.)
Wiss.Datenverarbeitung
FREIE UNIVERSITÄT BERLIN

Mathematica: Sammlung von WDV-Prozeduren – Teil 4.

© 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.

20. Signalanalyse eines „Spikes“:

Mit dem Softwarepaket ***SignalProcessing*** des Georgia Institute of Technology (wdv-notes Nr. 139) lassen sich auch Signale (Zeitfunktionen) bezüglich ihrer Basiseigenschaften analysieren. Das ***SignalProcessing*** stellt für die Analyse kontinuierlicher (mehrdimensionaler) Signale die Prozedur **ASPAnalyze** zur Verfügung. Diese untersucht in mehreren Schritten das (formelmäßig!) gegebene Signal und ermittelt u. a. das Amplitudenspektrum (magnitude response), das Phasenspektrum (phase response) und gegebenenfalls das Pol-Nullstellendiagramm in der komplexen *s*-Ebene sowie die (formelmäßige) Fourier-Transformation (frequency response).

Im folgenden Beispiel wird an Hand eines biologischen Signals (Aktionspotential) eine vollständige Sitzung einer solchen Signalanalyse mit **ASPAnalyze** dokumentiert. Man beachte dabei die Notationen: I = Sqrt (–1) = j, E = Eulersche Zahl *e*, *w* = Kreisfrequenz $\omega = 2\pi f$, *s* = komplexe Frequenz = $\tau + j\omega$, CStep ist die Einheitssprungfunktion und *t* = Zeit.

In[4]:=

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

```
Defining stubs for objects in version 2.6 of
the signal processing support packages ...
Defining stubs for the objects in version 2.6 of
the digital signal processing packages ...
Digital signal processing extensions have been
loaded.
      ...
Defining stubs for objects in version 2.6 of
the analog signal processing packages ...

Analog signal processing extensions have been
loaded.
      ...
support::loaded:
   Supporting routines, objects, and rules
   for the signal processing package have been
   loaded successfully.
digital::loaded:
   Routines, objects, and rules for the digital
   signal processing packages have been loaded
   successfully.
support::loaded:
   Supporting routines, objects, and rules for the
   signal processing package have been loaded
   successfully.
analog::loaded:
   Routines, objects, and rules for the analog
   signal processing packages have been loaded
   successfully.
```

Beispiel:

Bekanntlich läßt sich ein Aktionspotential (Nervenimpuls, „spike“) näherungsweise durch folgende Zeitfunktion beschreiben:

$$f(t) = \text{step}(t) \cdot e^{-7t/4} \cdot \cos(4\pi/5)$$

In der *Mathematica*-Notation ergibt sich daraus folgende Darstellung für das biologische Signal:

In[5]:=

```
spike = CStep[ t ] Exp[ –7 t /4 ] Cos[ 4 Pi t /5 ]
```

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

Out[5]=

```
      4 Pi t
   Cos[------] CStep[t]
        5
   --------------------
          (7 t)/4
         E
```

Mit dieser Beschreibung des Aktionspotentials kann jetzt die Analyse vorgenommen werden.

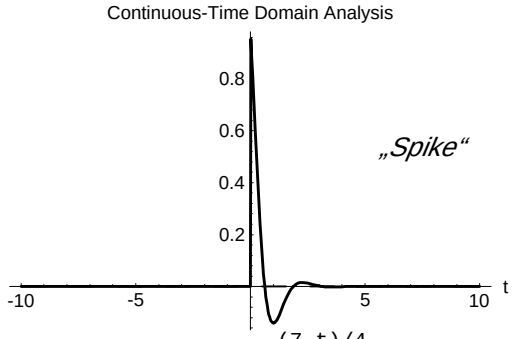
In[6]:=

```
ASPAnalyze [ spike, t, –10, 10 (*ms*) ]
```

```
Region of convergence routines for the transform
rule bases are loaded.
Supporting routines for transform rule bases are
loaded.
Supporting routines and objects for the Laplace
transform are loaded.
Supporting routines for filter design are loaded.
The forward Laplace transform rule base LaPlace
has been loaded.
The inverse Laplace transform rule base
InvLaPlace has been loaded.
The supporting functions are loaded for symbolic
Fourier transforms.
The continuous Fourier transform (CTFT)
functions are loaded.
The analog signal analyzer is now loaded.
```

Out[6]=

```
Real part of the function is shown as solid lines.
Imaginary part of the function is shown as dashed
lines.
```

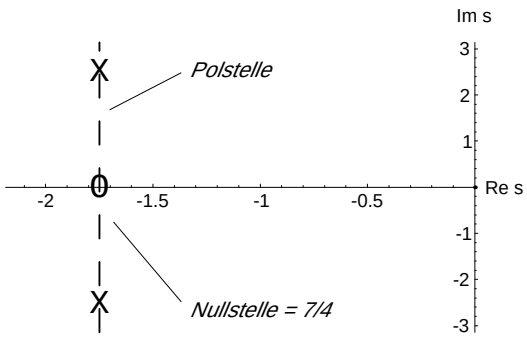



```
                    (7 t)/4
(Cos[(4 Pi t)/5] CStep[t])/E
has the following Laplace transform:
                          2          2
(7/4 + s)/((7/4 + s)  + (16 Pi )/25)

The region of convergence is:
-(7/4) < Re(s) < Infinity

The system is stable.

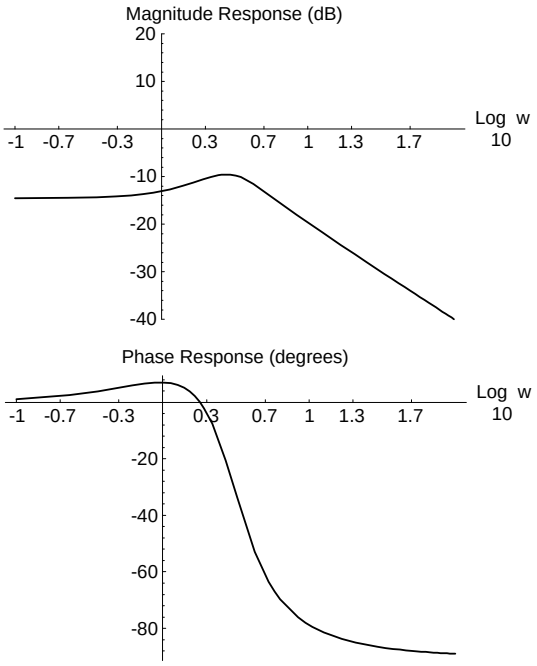
The zeroes are: {-1.75}
The poles are:  {-1.75 + 2.513 I, -1.75 - 2.513 I}
```



```
Since the signal is stable, the
frequency response can be computed
directly from the Laplace transform.

                                (7 t)/4
(Cos[(4 Pi t)/5] CStep[t])/E
has the following frequency response:
                          2         2
(7/4 + I w)/((7/4 + I w)  + (16 Pi )/25)
```

$$F(\omega) = \frac{\frac{7}{4} + j\,\omega}{\left(\frac{7}{4} + j\,\omega\right)^2 + \left(\frac{4\pi}{5}\right)^2}$$




```
                  7
                  - + s
                  4                          7
LTransData[-----------------, Rminus[-(-)],
                2                            4
           16 Pi     7     2
           ------ + (- + s)
             25      4

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

Unter dem Datatag **LTransData** wird abschließend die vollständige Laplace-Transformation des Signals mitgeteilt. Dabei begrenzen **Rminus** bzw. **Rplus** den Konvergenzbereich. Mit dem Datatag **LVariables** werden die Namen der Laplace-Variablen (hier *s*) angegeben (wdv-notes Nr. 139).

Die gesamte Laufzeit dieser Analyse betrug auf einem Macintosh-Computer vom Typ Quadra-700 im 32 Bit-Mode mit eingeschaltetem Cache 72 Sekunden, wobei 6,5 MByte Speicher (memory) benutzt wurden.

21. Zufügen eines 3D-Achsensystems:

Mathematica gibt 3D-Darstellungen mit einer einrahmenden „Box“ aus. Die Ausgabe eines 3D-Achsensystems läßt sich jedoch leicht „nachrüsten“.

Mit der im folgenden beschriebenen neuen Prozedur **Add3DAxes** wird in einer bereits produzierten 3D-Grafik nachträglich (!) das Achsensystem zugefügt, wobei mit der Option *AxesOrigin –> { x0, y0, z0 }* die Koordinaten des Ursprungs angegeben werden können. Die Prozedur **Add3DAxes** kann um weitere Optionen erweitert werden.

In[11]:=

```
Begin[„`Private`“];
Off  [General::spell1];

Options [ Add3DAxes ]
      = { AxesOrigin –> { 0, 0, 0 } };        (* Default *)

Add3DAxes [g:Graphics3D[ obj_,  opts___ ],
                                  opt1___ ] :=
  Module [ { down, up, origin, axes },
    { down, up} = Transpose [
                      FullOptions  [ g, PlotRange ] ];
      origin = AxesOrigin /. { opt1 } /.
                          Options [ Add3DAxes ];
      axes =  Table [ Line [
              { ReplacePart [ origin, Part [ down, i ], i],
                ReplacePart [ origin, Part [ up,  i], i] } ],
                              { i, 3 } ];
      Graphics3D [ { axes, obj }, opts ]  ]
  End[ ];  (* Private *)
```

Beispiel:

Die Verwendung von **Add3DAxes** wird an Hand der Grafikausgabe eines Schalenteiles verdeutlicht. Zunächst erfolgt mit der *Mathematica*-Bibliotheksprozedur **SphericalPlot3D** die Standardausgabe, wobei das Grafikobjekt unter der Variablen *gplot* zur Nachbearbeitung aufgehoben wird.

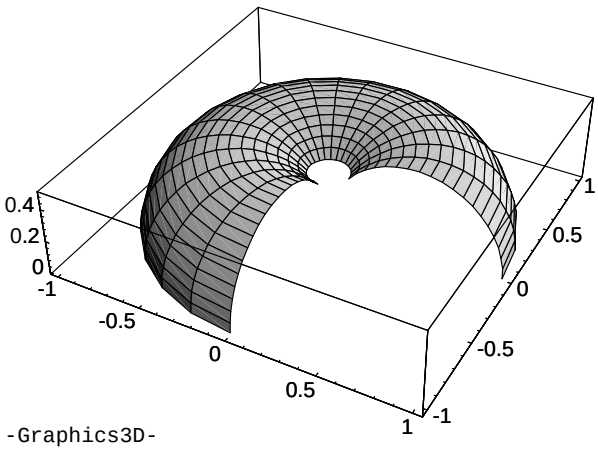
In[12]:=

```
<<ParametricPlot3D.m      (* Laden der Package. *)
```

In[13]:=

```
gplot = SphericalPlot3D [ Sin  [ theta ],
          { theta, Pi /9, Pi /2 }, { phi,  0,  3 Pi /2 },
            ColorOutput –> GrayLevel ]
```

Out[13]=



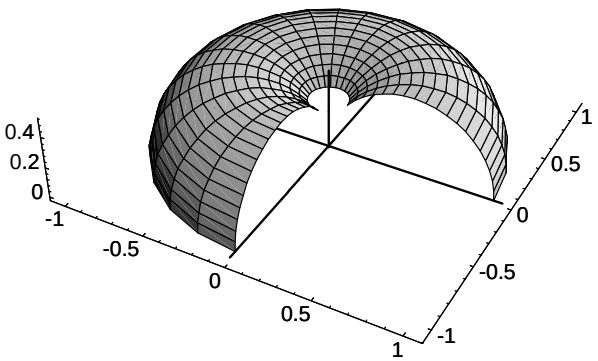

```
-Graphics3D-
```

Das Grafikobjekt wird nun nochmals unter Verwendung von **Add3DAxes** ausgegeben. Dabei wird mit der Option *Boxed –> False* außerdem die „volle Box“ weggenommen.

In[15]:=

```
Begin[„`Private`“];
Show [ Add3DAxes [ gplot ], Boxed –> False ]
```

Out[15]=




```
-Graphics3D-
```

Individuell lassen sich an die drei Achsen noch Beschriftungen anbringen, auf die in diesem Beispiel verzichtet wurde. Mit der Option *ViewPoint –> {vx, vy, vz}* besteht die Möglichkeit, das gesamte Objekt im 3D-Raum zu drehen.