wdv-notes 141
Stand: 3.DEZ.1992 (2.)
Wiss.Datenverarbeitung
FREIE UNIVERSITÄT BERLIN

Mathematica: Sammlung von WDV-Prozeduren – Teil 2.

© 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.

12. Die diskrete Fourier-Transformation:

Die folgende Prozedur **FFT** bildet die IDL-Funktion FFT(array, direction) der Interactive Data Language (wdv-notes Nr. 85) für (zur Zeit nur) eindimensionale Felder nach.

Dabei ist die unterschiedliche Handhabung der Normfaktoren A und B (wdv-notes Nr. 34) durch den Korrekturfaktor *nf* berücksichtigt worden, so daß die Ergebnisse zahlenmäßig den Ergebnissen entsprechen, wie sie mit IDL ermittelt werden. Gegenüber der IDL-Funktion FFT ist hier *kein* Shift des transformierten Signals erforderlich!

In[2]:=

```
FFT [ array_List, direction_ ] :=
   Module [ {  res,  n = Length [ array ], nf },
                     nf = N [ Sqrt [ n ] ];
      res := Chop  [ Fourier  [ { array } /nf ] ] /;
                                direction == 1;
      res := Chop  [ InverseFourier [ { array } nf ] ] /;
                                direction == –1;
      Flatten [ res ]   ]
```

Beispiel:

Mit den folgenden Aufrufen von **FFT** wird zunächst das komplexe Spektrum einer Rechteckfunktion berechnet (N = 16) und daraus durch inverse **FFT** wieder das Original-Signal ermittelt. Die grafische Darstellung erfolgt unter Punkt 13.

In[4]:=

```
rechteck   = { 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1 };
spectrum   = FFT [ rechteck, 1]
```

Out[4]=

```
{0.5, -0.063 - 0.314 I, 0, -0.063 - 0.094 I, 0,
      -0.063 - 0.042 I, 0, -0.063 - 0.012 I, 0,
      -0.063 + 0.012 I, 0, -0.063 + 0.042 I, 0,
      -0.063 + 0.094 I, 0, -0.063 + 0.314 I}
```

In[5]:=

```
signal = FFT [ spectrum, –1]
```

Out[5]=

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

13. Plot eines abgetasteten Signals:

Mit der im folgenden angegebenen neuen Prozedur **SPlot** kann ein diskretes Signal in einer grafischen Darstellung ausgegeben werden, in der die Abtastung besonders verdeutlicht wird. Für die Prozedur **SPlot** sind alle Grafik-Optionen der *Mathematica*-Prozedur **Show** zugelassen.

In[6]:=

```
SPlot [ s_List, {x_Symbol, xMin_, xMax_ },
                  options___ ] :=

  Module [ {factor, fun, lineThick, nend, n=Length[s],
            nstart, t, optlist, plot, pointThick },


  GetTicks [ nstart_, nend_ ] :=
    Module [ { fact, i, ninc, ns, nterm,
            baseticks= { 1, 2, 2, 5, 5, 5, 5, 5, 10, 10 } },
            ninc = Round [ (nend – nstart) / 5];
            If [ ninc == 0, ninc = 1 ];
            If [ ninc > 2,
            fact  = 10;
            While [ ninc > fact, fact *= 10 ];
            fact /= 10;
            nterm = Quotient [ ninc, fact ];
            ninc  = fact * baseticks [[ nterm ]]  ];
            ns = Sign[ nstart ]  ninc *
                  Quotient [ Abs [ nstart ], ninc ];
            Table [  { i+1, i, { 0, 0.02 } },
                     { i, ns, nend, ninc } ] ];


GetValue [ f_, lab_Symbol, n0_ ] :=
  Module [ {value},
            value = N [ f /. lab –> n0 ];
            If [ NumberQ [ value ],
               value,
            N [ Limit [ f, lab –> n0 ] ] ] ];


ToCollection [ a__ ] := a;


OneLine [ xy_ ] :=
     ToCollection [Line [ { { xy [[ 1 ]], 0 }, xy } ],
                    Point [ xy ]  ];


ToList [ ] := { };
ToList [ arg_List ] := arg;
ToList [ arg_ ] := List [ arg ] /; !
                   SameQ [ Head [ arg ], List ];
ToList [ arg1_, args__ ] := List [ arg1, args ];


Element [ z__, i_?IntegerQ ] := ToList [ z ]  [[ i ]];


  nstart   =   Floor [ N [ xMin ] ];
  nend     =   Ceiling [ N [xMax ] ];
  nticks   =   GetTicks [ nstart, nend ];


  optlist  =   Join [ Flatten [ ToList [ options ] ],
               { Axes –> True,
                 DefaultFont –> {"Helvetica",10},
                 PlotLabel –> None,
                 AxesLabel –> { "   "x, " " },
                 PlotRange –> { { 0, n+1 }, All     },
                 Ticks –> { nticks, Automatic }   }  ];


  fun        =   Element [ s, x ];
  factor     =   N [ Log [ 10, nend – nstart + 3 ] ];
  lineThick  =   0.0075 / factor;
  pointThick =   0.0225 / factor;


  Off [ Power::infy,  Infinity::indt ];
  plot = Show [
     Graphics [
           Join [ {  Thickness [ lineThick ],
                     PointSize   [ pointThick ] },
              N [  Map [ OneLine,
                   Table [  { t, GetValue [ fun, x,
                                 Round [ t ]  ] },
                            { t, nstart, nend } ]  ]  ]  ],
                   optlist ];
  On [ Power::infy,  Infinity::indt ];
  Return [ plot ]   ]        (* Ende von SPlot. *)
```

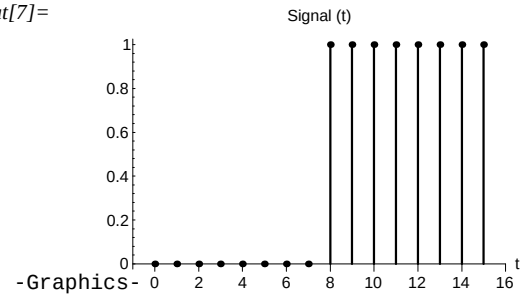
Beispiel:

Im folgenden sind mit **SPlot** die bereits unter Punkt 12 mittels der Prozedur **FFT** berechneten Signale dargestellt.

In[7]:=

```
sigPlot = SPlot [  signal, { t, 1, 16 },
                   PlotLabel –> "Signal (t)" ]
```

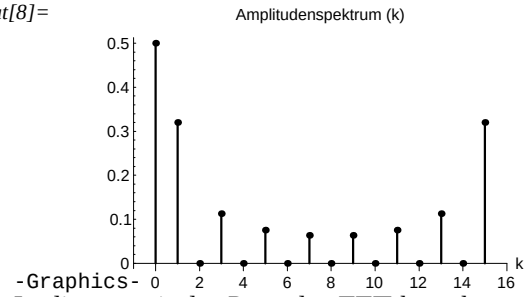
Out[7]=



In[8]:=

```
specPlot = SPlot [ Abs [ spectrum ], { k, 1, 16 },
                   PlotLabel –> "Amplitudenspektrum (k)" ]
```

Out[8]=



In diesem mit der Prozedur **FFT** berechneten diskreten Amplitudenspektrum der Rechteckfunktion (Punkt 12) ist deutlich die Symmetrie zur Nyquist-Frequenz $k = N/2 = 8$ zu erkennen. Der „Gleichanteil“ des Signals bildet sich im Spektrum bei der Frequenz $k = 0$ ab.

14. Erzeugen eines Test-Potentials:

Mit der Prozedur **TPot** kann ein künstliches Test-Potentials erzeugt werden, dem mit der *Mathematica*-Prozedur **Random** ein bestimmter Rauschanteil zugefügt wird.

In[9]:=

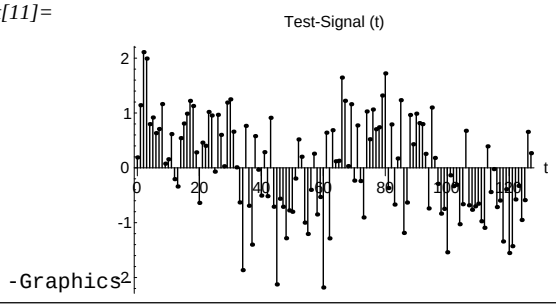
```
TPot [ n_Integer ] :=
   Table [ N[  0.6 Sin [ 2 Pi   2 i /n ]
             +  0.5 Sin [ 2 Pi 10 i /n ]
             +  0.8 Sin [ 2 Pi 10 i /n Cos [ 2 Pi 2.5 i/n ] ]
             +  (Random [ Real, {0,1.5},4] – 3/4)], {i,n} ]
```

Beispiel:

In[11]:=

```
testSignal = TPot [ 128 ];
SPlot [ testSignal, { t, 1, 128 },
                    PlotLabel –> "Test-Signal (t)" ]
```

Out[11]=



15. Powerspektrum eines Signals:

Mit der Prozedur **PowerSp** kann mittels der unter Punkt 12 beschriebenen Prozedur **FFT** ein normiertes Leistungsspektrum bzw. die Autokorrelation eines diskreten Signals (wdv-notes Nr. 8) berechnet werden.

Die grafische Darstellung des Powerspektrums erfolgt, wenn im Aufruf von **PowerSp** die Option *Plot –> True* angegeben wird. Wird als Option *Plot –> False* angegeben, dann wird die Datenliste des Powerspektrums (zum Beispiel für eine weitere Verarbeitung) übergeben. In den Plots des Powerspektrums wird nur der Frequenzbereich von $f = 0$ bis zur Nyquist-Frequenz $f = N/2$ (wdv-notes Nr. 34) dargestellt.

In[13]:=

```
Options [ PowerSp ] = { Plot –> True };    (* Default *)

PowerSp [ s_List, options___Rule ] :=
  Module [ { spectr, total, power, doPlot, defOpt,
             optList, n = Length [s], ny },
        ny = Floor [ n/2 ] + 1;
        doPlot = Plot /. {options} /. Options[PowerSp];
        defOpt  = { Plot –> doPlot };
        optList =  Join [ Flatten [
                   Complement [{options}, defOpt ] ],
                   { PlotLabel –> "Power Spectrum (f)"} ];
        spectr  := Abs [ FFT [ s, 1] ]^2;
        total  := Sum [ spectr [[ k ]], {k, 1, ny}](*/(2 n)*);
        Print ["Total Power = ", total ]; (*Print[ny];*)
        power   := Take [ spectr, ny ] /total;
        If [ doPlot === True,
           SPlot [ power, { f, 1, ny }, optList ],
           Release [ power ] ]  ]
```

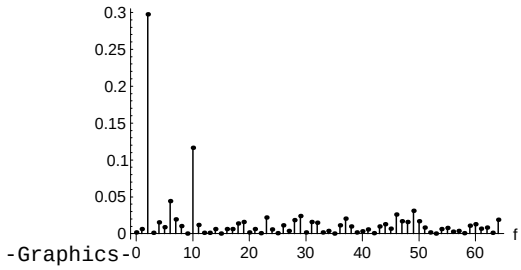
Beispiele:

Mit dieser neuen Prozedur **PowerSp** wird das Powerspektrum des unter Punkt 14 erzeugten Test-Signals berechnet und unter Verwendung von **SPlo**t dargestellt.

In[14]:=

```
power1 = PowerSp [ testSignal, Plot –> True ]
Total Power = 0.394487
```

Out[14]=



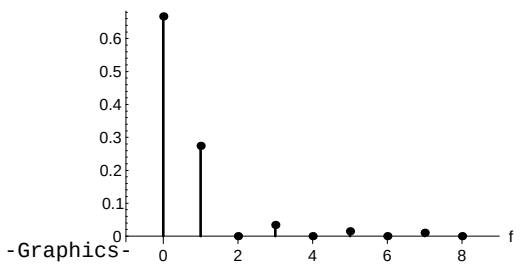
Im Powerspektrum sind deutlich die Spektrallinien bei den Frequenzen $f = 2$ und 10 sowie das bei der Frequenz $f = 10 – 2{,}5 = 7{,}5$ durch die Modulation entstandene kleine „Seitenband“ zu erkennen.

Im folgenden Beispiel wird das Powerspektrum des Rechtecksignals von Punkt 12 berechnet und dargestellt.

In[15]:=

```
power2 =   PowerSp [ rechteck,
   PlotLabel –> "Powerspektrum des Rechtecksignals"]
Total Power = 0.375
```

Out[15]=



16. Berechnen von Pi auf 100 Stellen:

Mit *Mathematica* sind numerische Berechnungen mit beliebiger Genauigkeit möglich. Die bei anderen Programmiersprachen übliche Beschränkung auf Zahlen mit endlicher Genauigkeit (z. B. auf 9 oder 16 Stellen) existiert nicht!

Beispiel:

In[16]:=

```
N [ Pi, 100 ]
```

Out[16]=

```
3.14159265358979323846264338327950288419716939937510
  58209749445923078164062862089986280348253421170 68
```