wdv-notes 142
Stand: 15.DEZ.1992 (2.)
Wiss.Datenverarbeitung
FREIE UNIVERSITÄT BERLIN

Mathematica: Sammlung von WDV-Prozeduren – Teil 3.

© 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.

17. Plots in der „Dot-Dash"–Technik:

Zur Visualisierung von bestimmten Daten – zum Beispiel Daten von „chaotischen" Systemen – benötigt man eine *xy*-Darstellung, aus der deutlicher die Dichte sowie eine Cluster-Bildung der Datenpunkte abzulesen sind. Hierfür eignet sich die sogenannte „Dot-Dash"-Darstellung.

Die folgende Prozedur **DotDashPlot** im Rahmen der neuen WDV-Package **SonderPlots** (P 508/3) erlaubt unter *Mathematica* eine solche Darstellung. Das Programm beruht auf dem Vorschlag von E. Gossett (E-Mail: gossett@bethel.edu).

In[13]:=

```
BeginPackage [ "SonderPlots`" ];

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

  xDashes [ xvals_, ymin_, yinc_ ] :=
    Module [ { dashes = { }, i },
      For  [ i=1, i<= Length [ xvals ], i++,
             dashes = Append [ dashes,
               Line [ { { xvals [[ i ]], ymin – 4 yinc },
                        { xvals [[ i ]], ymin – 2 yinc } } ] ]
           ];  Return [ dashes ]  ];

  yDashes [ yvals_, xmin_, xinc_ ] :=
    Module [ { dashes = { }, i },
      For  [ i=1, i<= Length [ yvals ], i++,
             dashes = Append [ dashes,
               Line [ { { xmin – 4 xinc, yvals [[ i ]] },
                        { xmin – 2 xinc, yvals [[ i ]] } } ] ]
           ];  Return [ dashes ]  ];

DotDashPlot [ data_List ] :=
  Module [ { xvals, yvals, xt, yt, xrange, yrange,
             xmin, xmax, ymin, ymax, xmargin,
             ymargin, labels, n = Length [ data ] },

    xvals    = Transpose [ data ] [[ 1 ]];
    yvals    = Transpose [ data ] [[ 2 ]];
    xmin     = Min  [ xvals ];
    xmax     = Max [ xvals ];
    ymin     = Min  [ yvals ];
    ymax     = Max [ yvals ];
    xrange   = xmax – xmin;
    yrange   = ymax – ymin;
          xt = .025 / GoldenRatio  xrange;
          yt = .025  yrange;
    xmargin  = xDashes  [ xvals, ymin, yt ];
    ymargin  = yDashes  [ yvals, xmin, xt ];

    labels = Graphics [
     Text [ToString [n], { xmin–2xt, ymin–8yt },{ 1,0} ],
     Text ["n = ",          { xmin–6xt, ymin–8yt },{ 1,0} ],
     Text [ToString [xmin], { xmin, ymin–6yt },{–1,0} ],
     Text ["xmin",          { xmin, ymin–9yt },{–1,0} ],
     Text [ToString [xmax], { xmax, ymin–6yt },{ 1,0} ],
     Text ["xmax",          { xmax, ymin–9yt },{ 1,0} ],
     Text [ToString [ymin], { xmin–6xt, ymin },{ 1,0} ],
     Text ["ymin",          { xmin–6xt, ymin–3yt },{ 1,0} ],
     Text [ToString [ymax], { xmin–6xt, ymax },{ 1,0} ],
     Text ["ymax",          { xmin–6xt, ymax–3yt },{ 1,0} ]
           ];    (* Ende von Graphics. *)

    Show [ ListPlot [ data, Axes –> False,
                      DefaultFont –> {"Helvetica", 8},
                      PlotStyle –> PointSize [.01],
                      PlotRange –> { { xmin–xt, xmax+xt },
                                     { ymin–yt, ymax+yt } },
                      DisplayFunction –> Identity ],
           Graphics [ {Thickness [.001], xmargin, ymargin} ],
           Graphics [ Line [ {  { xmin–2 xt, ymin },
                                { xmin–2 xt, ymax } } ]  ],
           Graphics [ Line [ {  { xmin,  ymin–2 yt },
                                { xmax, ymin–2 yt } } ]  ],
           labels,    PlotRange –> All,
                      DisplayFunction –> $DisplayFunction ]
         ] /;       (* Ende von Show. *)
                    ( MatrixQ [ data ] &&
                    ( Dimensions [ data ] [[ 2 ]]  ==  2 ) )

On [General::spell1];
End [ ];        (* Private *)
Protect [ DotDashPlot ];
EndPackage [ ];
```

Beispiel:

Die Anwendung von **DotDashPlot** wird im folgenden mit Zufallszahlen demonstriert. Die *x*-Koordinaten der Datenpunkte liegen dabei im Bereich [1,10] und die *y*-Koordinaten liegen im Bereich [1, 5]. Bei 167 gezogenen Datenpunkten zeigt sich mit dieser Art der Darstellung, daß nicht alle Bereiche gleich dicht besetzt sind. Es gibt Bereiche in denen sich Datenpunkte häufen (Cluster), was sich sehr deutlich aus den Strichen (dashes) an den Achsen ablesen läßt.

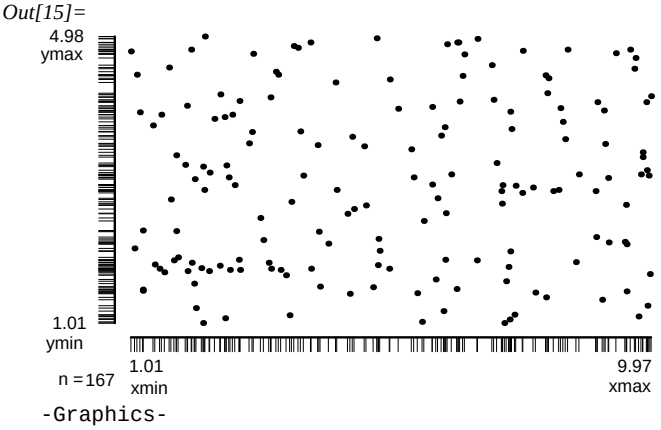
An den Achsen sind jeweils die Maximal- und die Minimalwerte notiert. In der unteren linken Ecke des Plots ist die Anzahl der dargestellten Punkte angegeben.

In[15]:=

```
Begin[ "SonderPlots`Private`" ];

DotDashPlot [ Table [ { Random [ Real, {1,10}, 3 ],
                        Random [ Real, {1,  5}, 3 ] },
                        { i, 167 } ]  ]
```

Out[15]=



18. Ermitteln einer Filterfunktion:

Im Rahmen von Wavelet-Analysen (wdv-notes Nr. 57) wird nach Mallat (1989) als Filterfunktion die 6. Ableitung einer speziellen Sinusfunktion $f(w)$ in einer bestimmten Form benötigt, wobei w die Kreisfrequenz ist. Mit *Mathematica* läßt sich ein solches analytisches Problem sehr einfach behandeln.

In[16]:=

```
filter [ x_ ] := Module [ { f, sechsteAbltg },
                 f := 1/ (4 Sin [ x/2 ] ^2 );
  sechsteAbltg  =  Simplify [ D [ f, { x, 6} ] ]  ]
```

In[17]:=

```
z = filter [ w ]
```

Out[17]=

```
                                                          w  8
(1208 + 1191 Cos[w] + 120 Cos[2 w] + Cos[3 w]) Csc[-]
                                                          2
-------------------------------------------------------------
                            128
```

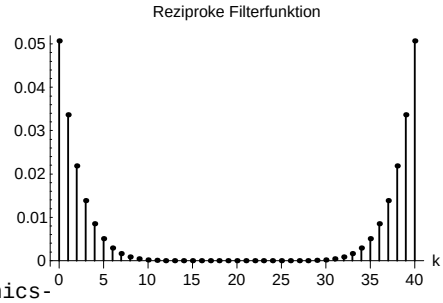
Beispiel:

Im folgenden wird diese Filterfunktion – das analytische Ergebnis (!) – im Bereich w = [–2, 2] mit der *Mathematica*-Prozedur **Table** abgetastet und die reziproke Funktion mit **SPlot** (wdv-notes Nr. 141) dargestellt.

In[18]:=

```
SPlot [ N [ 1/Table [ z, { w, –2,  2,  0.1 } ] ], { k, 1, 41 },
        PlotLabel –> „Reziproke Filterfunktion“ ]
```

Out[18]=




```
-Graphics-
```

19. Das Wavelet „Mexican Hat“:

Ein sehr häufig bei Wavelet-Transformationen (wdv-notes Nr. 57) verwendetes analysierendes Wavelet (Wellchen) ist die „Mexican Hat“ genannte Funktion. Darunter versteht man die negative 2. Ableitung der normierten Gauß-Funktion. Im folgenden wird diese Funktion symbolisch und grafisch mit den Mitteln von *Mathematica* dargestellt, wobei der Mittelwert m = 0 und s die Streuung (Varianz) ist.

Da unter *Mathematica* die normierte Gauß-Funktion selbst nicht als Prozedur zur Verfügung steht, muß diese erst über die (vorhandene) Normalverteilung als Prozedur **normGauss** sowie **Gauss** definiert werden. Die Differentiation wird mit dem *Mathematica*-Operator **D** vorgenommen. Außerdem wird für die grafische Ausgabe eine besondere Prozedur **LPlot** definiert, bei der sich im Aufruf bequem die unterschiedlichen Beschriftungen (Labels) angeben lassen.

In[26]:=

```
Needs["Statistics`NormalDistribution`"];

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

      Gauss [ x_, s_ ] :=
                PDF [ NormalDistribution [ 0, s ], x ]
normGauss  [ x_,s_ ] :=          (* Normierung *)
                Gauss [ x, s ] / Gauss [ 0, s ]
    Gauss2 [ x_, s_ ] :=          (* Neg. 2. Ableitung *)
                –D [ normGauss [ x, s ], { x, 2 } ]

LPlot [ function_, range_, Title_, xTitle_, yTitle_ ] :=
  Plot [ function, range,
      Frame –> True,
      PlotStyle     –> { Thickness [ 0.008 ] },
      RotateLabel –> False,
      DefaultFont –> {"Helvetica", 8 },
      PlotLabel     –> FontForm [ Title,
                       {"Helvetica-Bold", 12} ],
      FrameLabel –> { FontForm [ xTitle,
                       {"Times-Italic", 10} ],
                      FontForm [ yTitle,
                       {"Times-Italic", 10} ] } ]
  End[ ];  (* Private *)
```

Beispiel:

Es wird nun zunächst die normierte Gauß-Funktion $g(x)$ symbolisch berechnet (in Kästen als Formelsatz notiert) und dann mit der Prozedur **LPlot** für den Fall s = 1 grafisch im Bereich x = [–6, 6] dargestellt.

In[28]:=

```
Begin["`Private`"];
g = normGauss [ x, s ]
```

Out[28]=

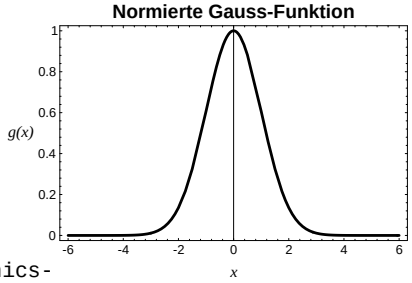
```
     2     2
  -x /(2 s )
 E
```

$$g(x) = e^{-x^2/2s^2}$$

In[29]:=

```
LPlot  [ normGauss [  x, 1 ], { x, –6, 6 },
       "Normierte Gauss-Funktion", "x", "g(x)" ]
```

Out[29]=




```
-Graphics-
```

Durch zweimaliges Differenzieren von $g(x)$ nach x erhält man mit der Prozedur **Gauss2** die allgemeine Form dieses Wavelets als $g''(x)$.

In[30]:=

```
g2 [ x_, s_:1 ]  = Simplify [ Gauss2 [ x, s ] ]
```

Out[30]=

```
     2   2
    s - x
-------------
     2     2
  x /(2 s )    4
 E            s
```

$$g''(x) = \frac{1}{s^2} \cdot \left(1 - \frac{x^2}{s^2}\right) \cdot e^{-x^2/2s^2}$$

Daraus ergibt sich schließlich mit s = 1 das analysierende Basis-Wavelet $\psi(x)$ (wdv-notes Nr. 91), dessen Plot mit Hilfe der oben definierten Prozedur **LPlot** im Bereich x = [–6, 6] ausgegeben wird.

In[31]:=

```
psi [ x_, s_ ] = Simplify [ g2 [ x ]  ]
```

Out[31]=

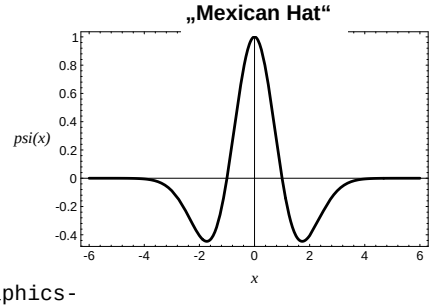
```
       2
1 - x
------
  2
 x /2
E
```

$$\psi(x) = \left(1 - x^2\right) \cdot e^{-x^2/2}$$

In[32]:=

```
LPlot [ psi  [ x, 1 ], {  x, –6, 6 },
        "„Mexican Hat“", "x", "psi(x)" ]
```

Out[32]=



m = 0
s = 1

```
-Graphics-
```