Einführung in *Mathematica*
-- Teil III .-

zur Vorlesung *Mathematische Methoden der Physik* im WiSe 2014/15

: : Prof. Dr. Norbert Dragon und PD Dr. Michael Flohr : : 05. 12. 2014 : :

Interaktion

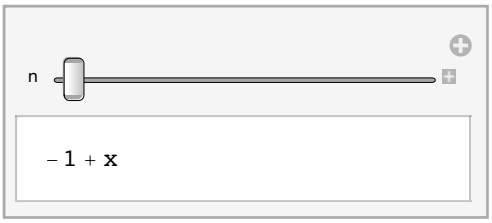
Eine der großen Unterschied von *Mathematica* zu gewöhnlichen Programmiersprachen ist, dass man sehr einfach interaktive Sandkästen bauen kann, in denen man spielen und ausprobieren kann. Das ist ein unschätzbar wertvolles Hilfsmittel, um eigenständig bei mathematischen Problemen Lösungsstrategien zu finden, oder Ergebnisse in Abhängigkeit von den Parametern zu untersuchen.

Manipulate stellt eine Art "Schachtelfunktion" dar, mit welcher sich ein oder mehrere "innere" Parameter eingebetteter Funktionen (beispielsweise Plot) interaktiv in gegebenen Grenzen variieren lassen.

Achtung: verwendet man in Manipulate extern definierte Funktionen, dann müssen die Parameter in der externen Definition als Variablen deklariert werden. Sonst behandelt *Mathematica* diese als Konstanten, und Manipulate kann diese dann nicht mehr variieren.

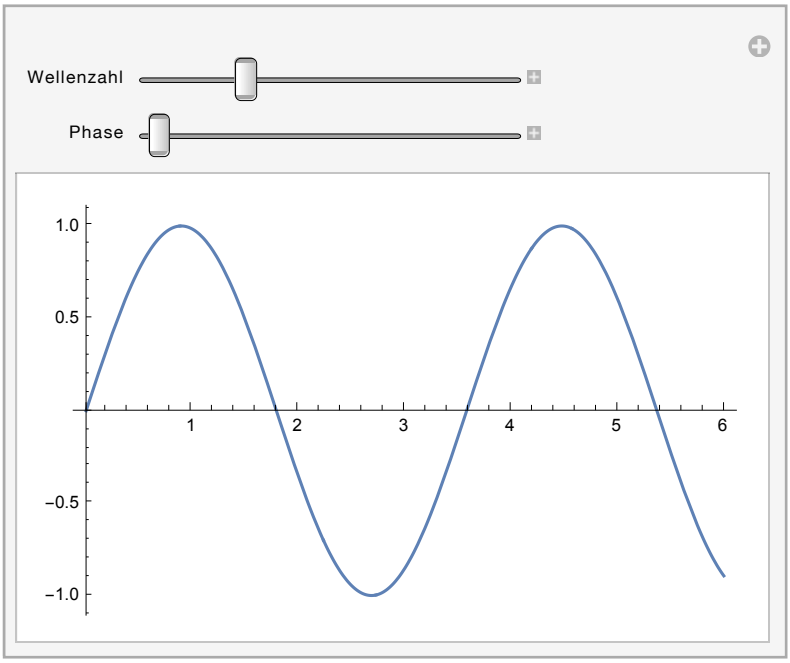
Manipulation eines Parameters in diskreten Stufen

```
Manipulate[Factor[x^n - 1], {n, 1, 100, 1}]
```



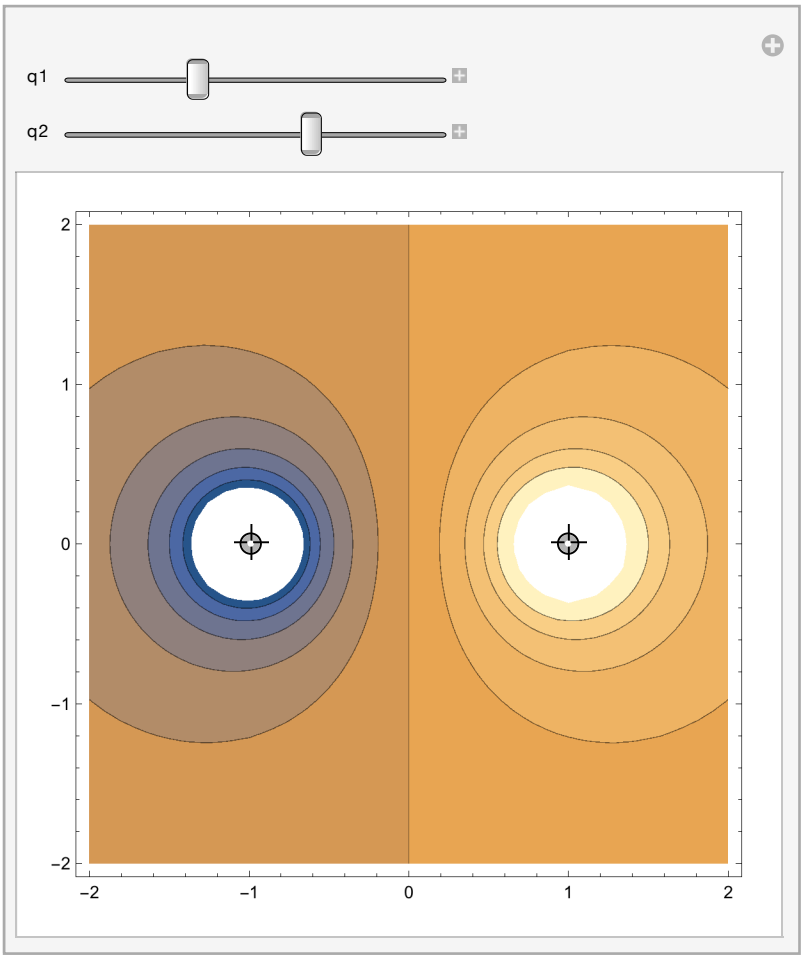
Manipulation zweier kontinuierlicher Parameter

```
Manipulate[Plot[Sin[a x + b], {x, 0, 6}],
 {{a, 2, "Wellenzahl"}, 1, 4},
 {{b, 0, "Phase"}, 0, 10}]
```



Ein komplexeres Beispiel anhand des elektrostatischen Potentiales zweier Ladungen

```
Manipulate[ContourPlot[q1 / Norm[{x, y} - p[[1]]] + q2 / Norm[{x, y} - p[[2]]],
  {x, -2, 2}, {y, -2, 2}, Contours → 10],
  {{q1, -1}, -3, 3},
  {{q2, 1}, -3, 3},
  {{p, {{-1, 0}, {1, 0}}}, {-1, -1}, {1, 1}, Locator},
  Deployed → True]
```



Import und Export von Daten

Wir können in *Mathematica* leicht Daten im- und exportieren. Das kann vor allem dann nützlich sein, wenn man Messdaten mit *Mathematica* weiterverarbeiten möchte, oder Ergebnisse von *Mathematica* in spezieller Software für numerische Aufgaben einspeisen möchte (weil *Mathematica* die symbolischen Ausdrücke nicht mehr weiter lösen kann). Hierbei kann man beim Im- und Export kontrollieren, in welchem Format dies geschehen soll. In unserem ersten Beispiel handelt es sich um Listen, die in *Mathematica* mit dem Table-Konstrukt erzeugt werden (siehe Teil II).

```
Export["hurz.txt", Table[{n, N[(-1)^n / n!]}, {n, 0, 20}], "Table"]
```

```
hurz.txt
```

```
FilePrint["hurz.txt"]
```

```
0   1.
1   -1.
2   0.5
3   -0.16666666666666666
4   0.041666666666666664
5   -0.008333333333333333
6   0.001388888888888889
7   -0.0001984126984126984
8   0.0000248015873015873
9   -2.7557319223985893e-6
10  2.755731922398589e-7
11  -2.505210838544172e-8
12  2.08767569878681e-9
13  -1.6059043836821613e-10
14  1.1470745597729725e-11
15  -7.647163731819816e-13
16  4.779477332387385e-14
17  -2.8114572543455206e-15
18  1.5619206968586225e-16
19  -8.22063524662433e-18
20  4.110317623312165e-19
```

```
Import["hurz.txt", "Table"]
```

$$\{\{0, 1.\}, \{1, -1.\}, \{2, 0.5\}, \{3, -0.166667\}, \{4, 0.0416667\},$$
$$\{5, -0.00833333\}, \{6, 0.00138889\}, \{7, -0.000198413\}, \{8, 0.0000248016\},$$
$$\{9, -2.75573 \times 10^{-6}\}, \{10, 2.75573 \times 10^{-7}\}, \{11, -2.50521 \times 10^{-8}\},$$
$$\{12, 2.08768 \times 10^{-9}\}, \{13, -1.6059 \times 10^{-10}\}, \{14, 1.14707 \times 10^{-11}\},$$
$$\{15, -7.64716 \times 10^{-13}\}, \{16, 4.77948 \times 10^{-14}\}, \{17, -2.81146 \times 10^{-15}\},$$
$$\{18, 1.56192 \times 10^{-16}\}, \{19, -8.22064 \times 10^{-18}\}, \{20, 4.11032 \times 10^{-19}\}\}$$

```
Export["zufall.dat",
  Table[RandomInteger[{-1000, 1000}], {i, 1, 30}, {j, 1, 30}], "List"];
```

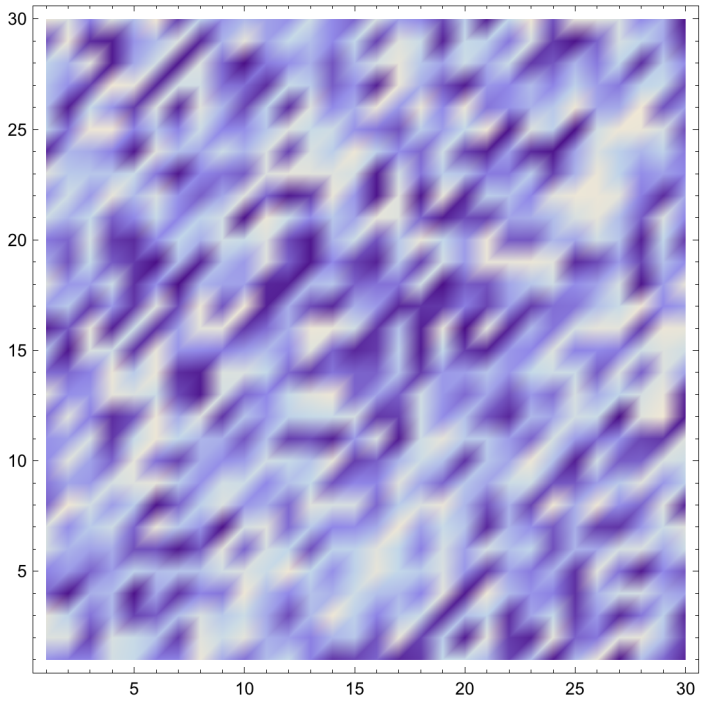
```
FilePrint["zufall.dat"]
```

```
{-251, -376, -355, 565, 379, -377, -358, 907, -44, -454, 102, 379, -366, -783, -692, -526, -85
{716, 689, -697, 629, 419, 167, -728, 777, 476, 804, 16, -118, -369, 57, 195, -611, 871, -698,
{828, 287, -437, 48, -707, -558, 110, 648, 208, 463, -50, -645, 93, 971, 271, -349, 186, 446,
{-525, -959, -13, 97, -973, -73, -790, -354, -424, 962, 83, -321, 150, 19, 480, 148, 83, -336,
{-353, -157, -520, -144, 277, 112, 168, 160, 543, 687, 441, 992, 337, -245, 278, 858, 345, 224
{605, 39, -108, -16, -633, -624, -931, -588, 790, -511, 598, 313, -625, 277, 282, 776, 327, -3
{665, -258, 238, 483, -515, 863, 193, -197, -974, 569, 773, -474, 209, -327, -283, -36, 950, -
{-535, 988, 561, -133, 227, -957, -524, 263, 674, -204, -20, -495, 745, 624, 614, -504, 241, 6
{-419, -75, -186, 596, 736, 854, 228, -243, 546, 474, -314, 139, 361, -596, -345, -350, 751, -
{-288, 758, -102, -518, 841, -451, -698, -30, -266, -157, 546, 402, 173, 854, -723, -553, 54,
{-238, -134, 606, -767, 393, 941, 329, 45, -10, -590, 130, -727, -658, -899, -168, 615, -925,
{-449, 732, -373, -824, -618, 938, -28, 700, -774, 147, -213, 275, 387, 446, 595, -766, -724,
{787, -260, -199, 860, 661, 461, -675, -941, 593, -497, -326, 384, 786, 958, -524, -105, -548,
{-488, -501, -360, 910, 224, 895, -827, -909, 683, 673, 357, -491, -744, -426, -532, -538, 156
{167, -946, 665, 521, -788, 867, -227, -307, -385, -167, 92, -271, 757, -523, -851, -751, 262,
{-884, -587, 820, -884, 459, -797, -120, 788, 825, -753, -484, -12, 925, 887, 520, -789, 145,
{-79, 201, -734, -182, -771, 707, -653, 917, -593, 960, -897, -24, -648, -850, -12, -187, -863
{702, -199, -145, 449, -806, -265, -998, 56, -104, -124, -872, -897, 90, 167, -529, -452, -440
{-31, -605, 361, -582, -755, -985, 118, -876, 143, -164, 739, -544, -952, -323, -704, -880, 19
{-411, -516, 581, -746, -839, -296, 329, 2, 543, 420, 706, -653, -955, 39, 452, -610, 204, 886
{495, 50, 374, -539, -311, 395, -341, 273, 548, -972, 909, 414, -286, 199, -92, 840, 729, -526
{-216, 499, -164, 848, 105, 652, -249, -540, -65, 743, -514, -927, -763, 884, -188, -961, 748,
{748, 222, 196, 193, 445, -412, -943, -320, -432, -561, 702, 822, 784, 733, 591, -932, 530, -4
{-851, 54, -46, -361, -935, 769, 123, -70, 464, 49, 667, -702, 627, 250, 941, 256, -275, -507,
{305, -450, 963, 967, -358, 171, -348, 473, -336, -275, 184, 722, 418, 484, -237, -420, -621,
{138, -914, 388, -209, -743, 667, -871, 745, 57, -682, 280, -816, 701, -300, 9, 882, -171, 889
{188, -268, -712, 990, 837, -859, 540, 525, -447, -473, -815, 401, 216, -559, -244, -895, 895,
{309, 26, 770, -929, -51, 529, -939, 679, -501, -990, -395, -265, -1, -603, 60, 574, 870, 200,
{-608, 917, -838, -923, -225, 101, 672, -863, 27, 818, -108, 367, 129, -313, 345, -700, 742, 4
{912, 676, -348, 680, 136, 997, -708, 272, -205, 74, -106, -243, 490, -302, -505, 621, -132, -
```

Statt Import muß nun ReadList benutzt werden, da *Mathematica* sonst jede Zeile fälschlich als Zeichenkette einliest und dann nicht wie erwartet eine Liste aus dreißig Listen mit jeweils dreißig

Elementen erzeugt! Hier wollen wir mal sehen, wie zufällig der Zufallsgenerator von *Mathematia* würfelt. Die Funktion ListDensityPlot interpoliert die diskreten Werte über einem zwei-dimensionalen Gitter, wobei hellere Farböne größere Werte anzeigen. Sie ist besonders geeignet, die Einträge von Listen von Listen (also einer Liste von Zeilen, so dass in unserem Beispiel ein 30x30 Gitter entsteht) auf einen Schlag zu visualisieren. Je ungeordneter dieses Muster aussieht, desto zufälliger sind die Einträge gewesen.

```
ListDensityPlot[ReadList["zufall.dat"]]
```

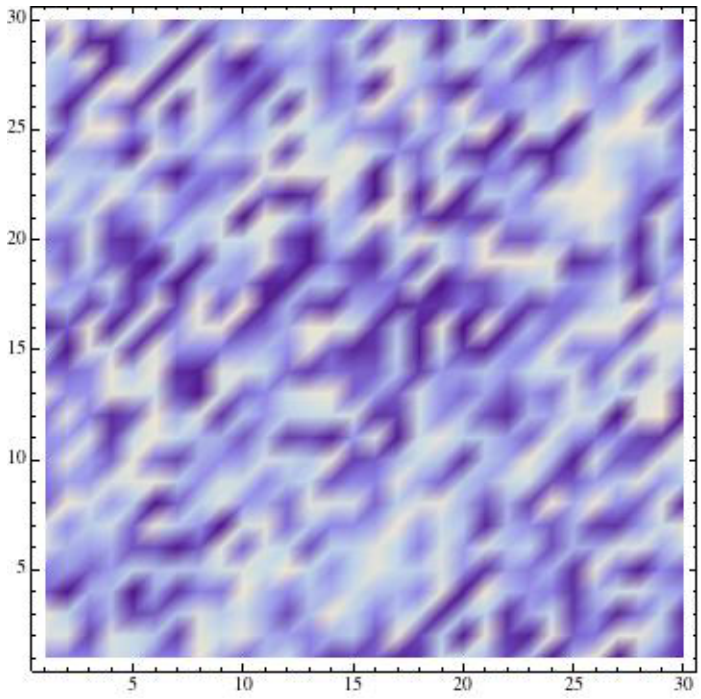



```
Export["zufall.jpg", %, "JPEG"]
```

```
zufall.jpg
```

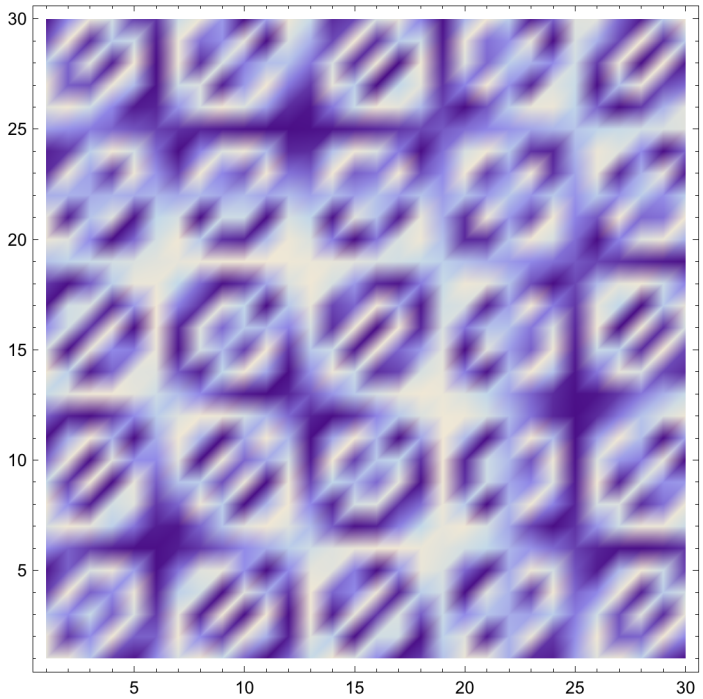
Anzeigen der zuvor gespeicherten Datei

```
Import["zufall.jpg"]
```



Nur zum Vergleich: das folgende Beispiel zeigt ganz klar, dass die Werte in dieser Matrix nicht rein zufällig sein können.

```
daten = Table[N[Sin[i j]], {i, 1, 30}, {j, 1, 30}];
ListDensityPlot[daten]
```



Mit *Mathematica* sind auch Bildverarbeitungen und -analysen möglich - und zwar mit beliebigen "mathematisch exakten" Filtern. Damit kann man zum Beispiel komplexe Daten sehr gut durch geschickte FIlterung und Visualisierung so darstellen, dass man Zusammenhänge, also mögliche Korrelationen, sichtbar machen kann.

```
(* Das folgende Kommando müssen Sie auf Ihre Situation anpaasen.
   Je nach dem, wo Sie die Datei Mainhattan.bmp abspeichern,
   müssen Sie anschließend den Pfad zu Laden des Bildes entsprechend
   setzen. *)
SetDirectory["mathematica/mmdp/WiSe14-15"]
```

Out[4]= /Users/phytzel/mathematica/mmdp/WiSe14-15

In[5]:=
```
bild = Import["Mainhattan.bmp"(* by Martin Paech, 2009/02/18 18:46:33 CET *)]
```

Out[5]=

Zunächst müssen die Bilddaten in eine mehrdimensionale Liste umgewandelt werden, ...

```
bilddaten = ImageData[bild, "Byte"];
```

```
bildgroesse = Dimensions[bilddaten]
```

{912, 1280, 3}

```
bildgroesse[[1]] * bildgroesse[[2]]
```

1 167 360

... welche aus 1167360 Bildpunkten (912 Zeilen zu jeweils 1280 Spalten) besteht, mit einem "Tripel" (dreikomponentiger Vektor) aus ganzen Zahlen für die drei Grundfarben (in 256 Stufen: 0 ist schwarz, 255 entspricht reinem Rot, Grün oder Blau) im RGB-Farbraum.

Um aus dem Negativbild ein Positiv zu machen, ist nur folgendes einfaches Umrechnungsprogramm nötig:

```
For[y = 1, y ≤ bildgroesse[[1]], y++, (* Achtung:
  Mathematica vertauscht die "natürlichen" x- und y-Koordinaten! *)
 For[x = 1, x ≤ bildgroesse[[2]], x++,
  bilddaten[[y, x]][[1]] = 255 - bilddaten[[y, x]][[1]] (* roter Farbkanal *);
  bilddaten[[y, x]][[2]] = 255 - bilddaten[[y, x]][[2]] (* grüner Farbkanal *);
  bilddaten[[y, x]][[3]] = 255 - bilddaten[[y, x]][[3]] (* blauer Farbkanal *)
 ]
]
```

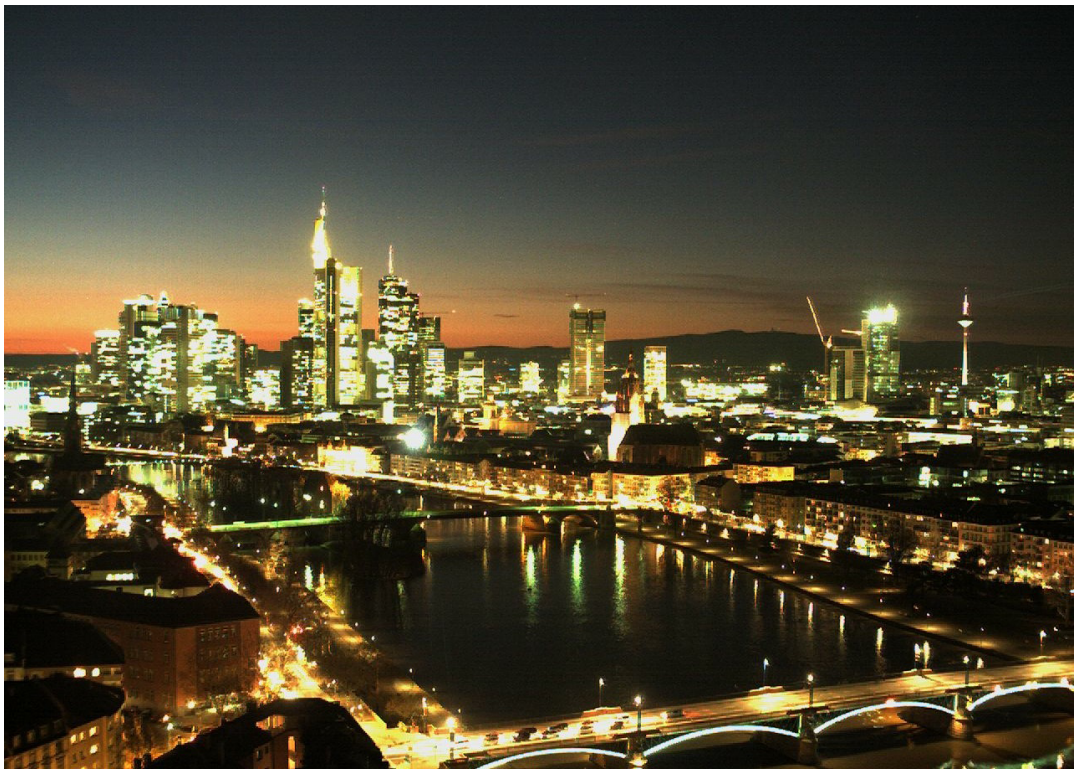
Man beachte, dass wir diese Manipulation “in palce” vorgenommen haben. Wir erkennen nunmehr, daß Frankfurt am Main im Abendlicht (mindestens ;-) ebenso reizvoll wie Hannover sein kann:

```
Image[bilddaten, "Byte"]
```

Doch halt! Dom, Fernsehturm, Großer Feldberg ... befinden sich nördlich des Mains. Auch das ist schnell repariert:

```
For[y = 1, y ≤ bildgroesse[[1]], y++,
 bilddaten[[y]] = Reverse[bilddaten[[y]]]
]
```

```
Image[bilddaten, "Byte"]
```



Animationen

Nachdem wir gelernt haben, wie wir selbst Daten interaktiv manipulieren können, und dass man sozusagen fast alles zu Daten machen kann, die mit mathematischen und symbolischen Methoden bearbeitet werden können, und nachdem wir vielleicht lange an der Lösung eines Problems gebastelt haben, wollen wir das ganze dann am Ende möglichst cool und effektvoll in Szene setzen - also am besten gleich als eine Art Movie. In der Tat bietet *Mathematica* auch erstaunlich einfach zu handhabende Möglichkeiten an, Formeln und Daten in animierte, also bewegte, Grafiken umzusetzen.

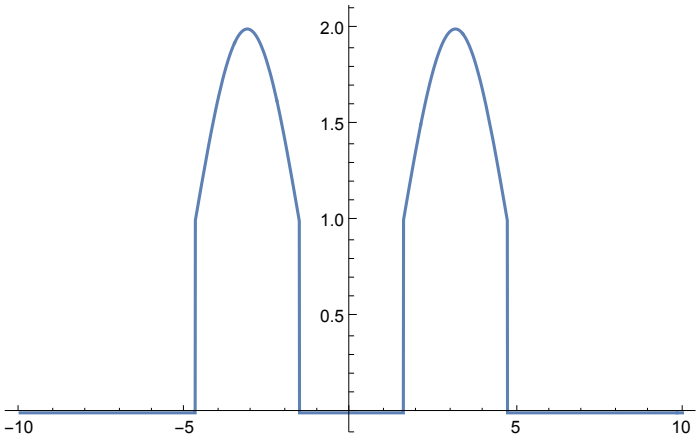
```
Clear[x, y]
```

Das folgende stellt einen brauchbare Approximation für “digitale” Schaltsignale (Pulse) dar. Diese sind selten echte Rechteckfunktionen, sondern sehen durchaus eher so aus (oder noch “schlimmer”) aus.

In[7]:=
$$\texttt{tp[x\_] := If}\left[\frac{\pi}{2} \leq \texttt{Abs[x]} \leq \frac{3\,\pi}{2}, \texttt{1 - Cos[x]}, \texttt{0}\right]$$

In[8]:=
```
Plot[tp[x], {x, -10, 10}]
```

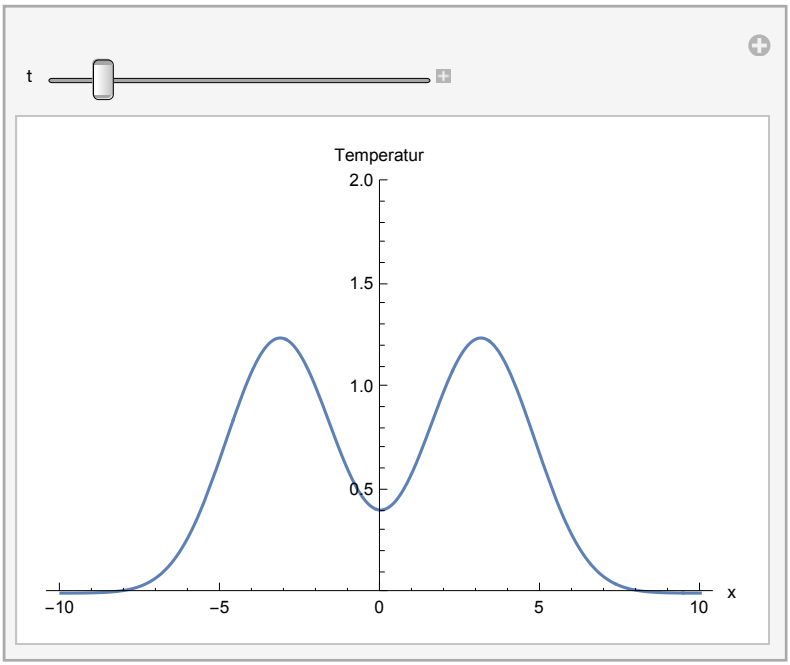
Out[8]=



Das folgende Integral nennt man *Faltungsintegral*. Unser Doppelpuls wird mit einer Gaußkurve gefaltet. Dies simuliert zum Beispiel, wie ein Puls durch Dämpfung in Leitungen seine Form verlieren kann, oder wie zum Beispiel ein eng fokussierter Lichtpuls durch Streuung auseinanderläuft. Ein weiteres typisches Beispiel ist der Verlust von Signalkohärenz durch temperaturbedingte Effekte (thermische Störungen bzw. thermisches Rauschen).
In unserem Beispiel wollen wir uns vorstellen, dass wir ein Material an zwei Stellen sehr heiß gemacht haben, und wollen sehen, wie es sich mit der Zeit abkühlt. Die Gaußkurve ist nämlich auch hierfür die richtige Funktion, sie spielt bei der Lösung der Wärmeleitgleichung eine wichtige Rolle. Wir haben hier die Breite der Gaußkurve als Funktionsparameter definiert, so dass wir die Auswirkung unterschiedlich starker Streuung bzw. unterschiedlich hoher Temperatur jetzt untersuchen können.

In[9]:=
```
tptd[x_, t_] = ∫_-∞^∞ e^-((-x+y)^2/(4 t)) / (2 √(π t)) tp[y] dy;
```

```
Manipulate[Plot[Re[tptd[x, t]], {x, -10, 10},
   AxesLabel → {"x", "Temperatur"},
   AxesOrigin → {0, 0},
   PlotRange → {0, 2}],
 {{t, 1}, 10^-6, 10}]
```

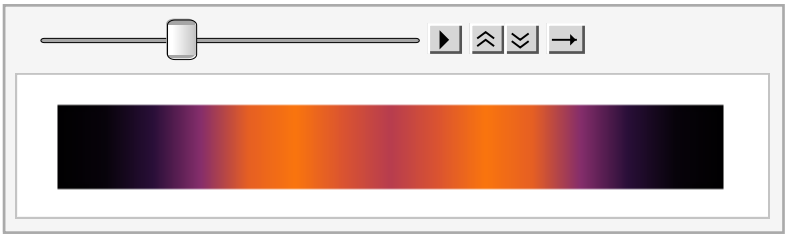


Und jetzt wollen wir das ganze als einen Film sehen, wie unsere beiden weißglüheden Stellen im Eisen abkühlen und den restlichen Barren dabei erwärmen (zumindest ist dies eine Interpretation des Vorgangs, die mit der gewählten Visualisierung gut zusammen passt).

```
Export["Temperaturprofil.avi",
 Table[DensityPlot[Re[tptd[x, t] / 2], {x, -10, 10}, {y, 0, 1},
   AspectRatio → 1 / 8, ColorFunction → "SunsetColors",
   ColorFunctionScaling → False, Frame → None], {t, 10^-6, 5, 0.1}], "AVI"]
```

```
Temperaturprofil.avi
```

```
Import["Temperaturprofil.avi", "Animation"]
```



Fouriertransformation

Nach diesen Spielereien wollen wir jetzt zum Abschluss noch ein weiteres wichtiges Anwendungsgebiet von *Mathematica* vorstellen, nämlich Transformationen von Funktionen. Die wichtigste solche Transformation, die Physiker sehr viel gebrauchen, ist die Fouriertransformation. Hinter ihr steckt eigentlich nicht viel anderes, als die vielleicht intuitiv nachvollziehbare Tatsache, dass ich Musik als Amplitude in der Zeit aufnehmen kann (wie auf der Schallplatte), oder als Frequnzspektrum (wie,

etwas vereinfacht betrachtet, im mp3-Spieler). Die Fouriertransformation rechnet die eine Forn in die (völlig äquivalente) andere um.
Auch zwei-dimensional kennen Sie das alle. Um digitale Bilder abzuspeichern, kann man entweder die Helligkeitswerte (Amplituden) eines Farbkanals für die Bildpunkte (zwei-dimensional statt ein-dimensionale Zeit) abspeichern, oder äquivalent die Fouriertransformation davon, was die Frequenzen sind, mit denen sich die Helligekitswerte in verschiedenen Richtungen "wiederholen". Das ist das, was in einer jpg-Datei passiert, die aus den meisten Digitalkameras rauskommt. Das erstaunliche ist, dass Bilder der Realität nach Fouriertransformation oft erstaunlich gut kompirimiert werden können, so wie auch Musik im mp3-Format bei tolerablen Einbußen in der Klangtreue viel weniger Speicherplatz einnimmt. *Übungsaufgabe*: Überlegen Sie, woran das liegen könnte ...

Elementare Funktionen

Zunächst ein-dimensional:

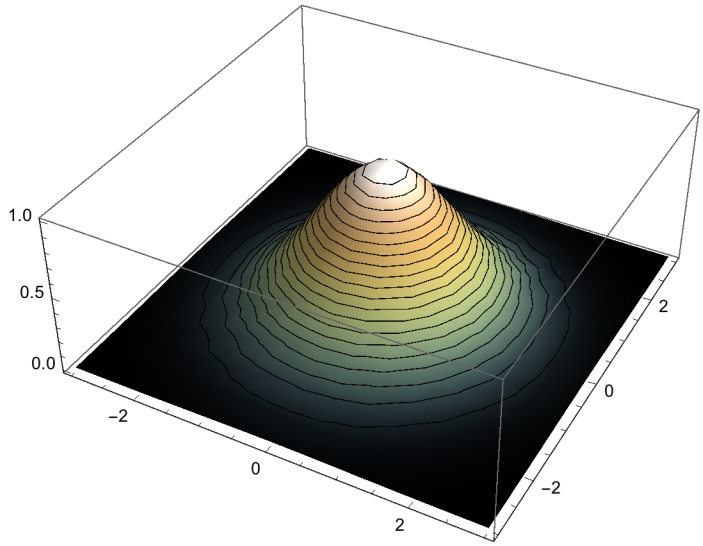
```
l = {1, x, x^2, 1/x, 1/x^2, e^(-x^2/2)};

TableForm[{l, FourierTransform[l, x, k]},
 TableDirections → Row, TableSpacing → {2, 6},
 TableAlignments → {Center, Left}, TableHeadings → {{"f(x)", "F(k)"}}]
```

f(x)	F(k)
1	$\sqrt{2\pi}$ DiracDelta[k]
x	$-i\sqrt{2\pi}$ DiracDelta′[k]
x^2	$-\sqrt{2\pi}$ DiracDelta″[k]
$\frac{1}{x}$	$i\sqrt{\frac{\pi}{2}}$ Sign[k]
$\frac{1}{x^2}$	$-k\sqrt{\frac{\pi}{2}}$ Sign[k]
$e^{-\frac{x^2}{2}}$	$e^{-\frac{k^2}{2}}$

Und nun zwei-dimensional:

```
g[x_, y_] := Exp[-(x^2 + y^2)/2]
```

```
Plot3D[g[x, y], {x, -3, 3}, {y, -3, 3}, PlotRange → All,
 MeshFunctions → {#3 &}, Mesh → 20, ColorFunction → "GreenBrownTerrain"]
```




```
FourierTransform[g[x, y], {x, y}, {u, v}]
```

$$e^{-\frac{u^2}{2}-\frac{v^2}{2}}$$

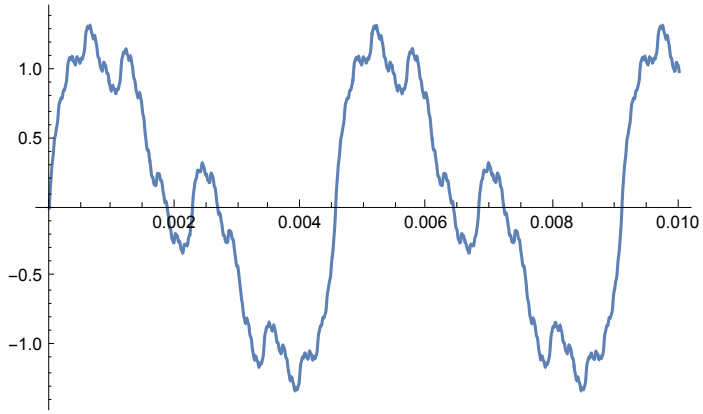
Periodische Funktionen

Strikt periodische Funktionen (also zum Beispiel kontinuierliche Töne) lassen sich durch Fourierreihen darstellen, denn in ihrem Frequenzspektrum treten nur die echten vielfachen der Grundfrequenz auf (Obertöne ...). Wir betrachten nun folgende reihenentwickelte Funktion:

In[10]:=
$$f[t\_] := \sum_{n=0}^{7} \frac{1}{2^n} \text{Sin}[2^n\, 220 \times 2\, \pi\, t]$$

In[11]:=
```
Plot[f[t], {t, 0, 0.01}]
```

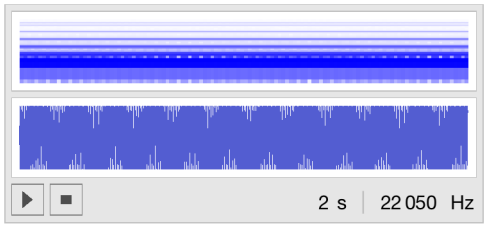
Out[11]=



Das kann *Mathematica* auch hörbar machen kann (220 Hz Grundton mit sieben Obertönen, was einen leicht orgelpfeienähnlichen Klang ergibt).

In[12]:=
```
Play[f[t], {t, 0, 2}, SampleRate → 22 050]
```

Out[12]=



Die (auch aus obiger Formel ersichtlichen) diskreten Frequenzen und Gewichte der Grundschwingung und der Oberschwingungen finden sich als δ-Funktionen in der Fouriertransformierten, der Lösung des Fourierintegrales

$$\sqrt{\frac{|b|}{(2\pi)^{1-a}}}\int_{-\infty}^{\infty} f(t)\, e^{i\,b\,\omega\,t}\, dt\,,$$

in welchem die Parameter *a* und *b* nachfolgend derart gewählt sind, daß sich unmittelbar das Frequenzspektrum und nicht die "Winkelgeschwindigkeiten" ergeben.

```
a = 0;
```

```
b = -2 π;
```

("Hintransformation")

```
ff[ω_] = FourierTransform[f[t], t, ω, FourierParameters → {a, b}]
```

$$-\frac{1}{256}\,i\,\text{DiracDelta}[-28\,160+\omega]-\frac{1}{128}\,i\,\text{DiracDelta}[-14\,080+\omega]-\frac{1}{64}\,i\,\text{DiracDelta}[-7040+\omega]-\frac{1}{32}\,i\,\text{DiracDelta}[-3520+\omega]-\frac{1}{16}\,i\,\text{DiracDelta}[-1760+\omega]-\frac{1}{8}\,i\,\text{DiracDelta}[-880+\omega]-\frac{1}{4}\,i\,\text{DiracDelta}[-440+\omega]-\frac{1}{2}\,i\,\text{DiracDelta}[-220+\omega]+\frac{1}{2}\,i\,\text{DiracDelta}[220+\omega]+\frac{1}{4}\,i\,\text{DiracDelta}[440+\omega]+\frac{1}{8}\,i\,\text{DiracDelta}[880+\omega]+\frac{1}{16}\,i\,\text{DiracDelta}[1760+\omega]+\frac{1}{32}\,i\,\text{DiracDelta}[3520+\omega]+\frac{1}{64}\,i\,\text{DiracDelta}[7040+\omega]+\frac{1}{128}\,i\,\text{DiracDelta}[14\,080+\omega]+\frac{1}{256}\,i\,\text{DiracDelta}[28\,160+\omega]$$

("Rücktransformation")

```
InverseFourierTransform[ff[ω], ω, t, FourierParameters → {a, b}]
```

$$-\frac{1}{256}\,i\,e^{-56\,320\,i\,\pi\,t}\left(-1-2\,e^{28\,160\,i\,\pi\,t}-4\,e^{42\,240\,i\,\pi\,t}-8\,e^{49\,280\,i\,\pi\,t}-16\,e^{52\,800\,i\,\pi\,t}-32\,e^{54\,560\,i\,\pi\,t}-64\,e^{55\,440\,i\,\pi\,t}-128\,e^{55\,880\,i\,\pi\,t}+128\,e^{56\,760\,i\,\pi\,t}+64\,e^{57\,200\,i\,\pi\,t}+32\,e^{58\,080\,i\,\pi\,t}+16\,e^{59\,840\,i\,\pi\,t}+8\,e^{63\,360\,i\,\pi\,t}+4\,e^{70\,400\,i\,\pi\,t}+2\,e^{84\,480\,i\,\pi\,t}+e^{112\,640\,i\,\pi\,t}\right)$$

Bei der Rücktransformation kann in diese Falle auch auf die Funktion InverseFourierSinTransform zurückgegriffen werden, welche sich auf ungerade, asymmetrische Komponenten beschränkt. Damit vermeiden wir, uns mit der komplexen e-Funktion auseinandersetzen zu müssen.

```
InverseFourierSinTransform[ff[ω], ω, t, FourierParameters → {a, b}]
```

$$\frac{1}{128}\,\mathrm{i}\,(128\,\mathrm{Sin}[440\,\pi\,t] + 64\,\mathrm{Sin}[880\,\pi\,t] + 32\,\mathrm{Sin}[1760\,\pi\,t] + 16\,\mathrm{Sin}[3520\,\pi\,t] + 8\,\mathrm{Sin}[7040\,\pi\,t] + 4\,\mathrm{Sin}[14\,080\,\pi\,t] + 2\,\mathrm{Sin}[28\,160\,\pi\,t] + \mathrm{Sin}[56\,320\,\pi\,t])$$

Die Nachbarbeitung der allgemeinen Lösung ...

```
fff[t_] = Simplify[Im[%], Assumptions → t ∈ Reals]
```

$$\frac{1}{128}\,(128\,\mathrm{Sin}[440\,\pi\,t] + 64\,\mathrm{Sin}[880\,\pi\,t] + 32\,\mathrm{Sin}[1760\,\pi\,t] + 16\,\mathrm{Sin}[3520\,\pi\,t] + 8\,\mathrm{Sin}[7040\,\pi\,t] + 4\,\mathrm{Sin}[14\,080\,\pi\,t] + 2\,\mathrm{Sin}[28\,160\,\pi\,t] + \mathrm{Sin}[56\,320\,\pi\,t])$$

... führt erwartungsgemäß auf die Ausgangsfunktion ("Beweis durch Hingucken").

```
Plot[{f[t], fff[t]}, {t, 0, 0.01},
 PlotStyle → {Directive[Thick, Red], Directive[Dashed, Thick, Black]}]
```