R. Brigola, TH Nürnberg Georg Simon Ohm,
Fakultät Angewandte Mathematik, Physik und Allgemeinwissenschaften

Juli 2013

Mathematica - Notebooks als Bonusmaterial zum Lehrbuch

[1] Rolf Brigola Fourier-Analysis und Distributionen,
Eine Einführung mit Anwendungen,
edition swk, Hamburg 2013

Teil 3 Grundwissen über die diskrete Fouriertransformation Beispiele zur diskreten Fouriertransformation (DFT, FFT) und zur diskreten Cosinustransformation (DCT I und DCT II)

Das Notebook ist mit Mathematica 9 unter Windows 7 erstellt. Dieses und weitere Demo - Notebooks findet man unter folgender URL des Autors:
www.stiftung-swk.de/mathematica

---

1. Grundwissen über die diskrete Fouriertransformation

Wir verwenden nachfolgend die DFT und die DCT in der Form wie in [1], Abschnitt 5.7, d.h. eine N-Punkte-DFT hat den Vorfaktor 1/N. Die DFT ist in *Mathematica* bereits mit einem FFT-Algorithmus implementiert. Wir betrachten ein erstes Beispiel:

1.1 DFT und Frequenz-Zuordnung, Umgang mit Alias-Effekten

Einer der wesentlichen Effekte bei einer DFT ist der *Alias-Effekt*, d.h. dass bei einer N-Punkte-DFT Kreisfrequenzen der Form $(k + mN)\,\omega_0$ im beobachteten Signal (ω_0=2 π/T, T die Beobachtungsdauer, m $\epsilon\,\mathbb{Z}$) nicht unterschieden werden können.

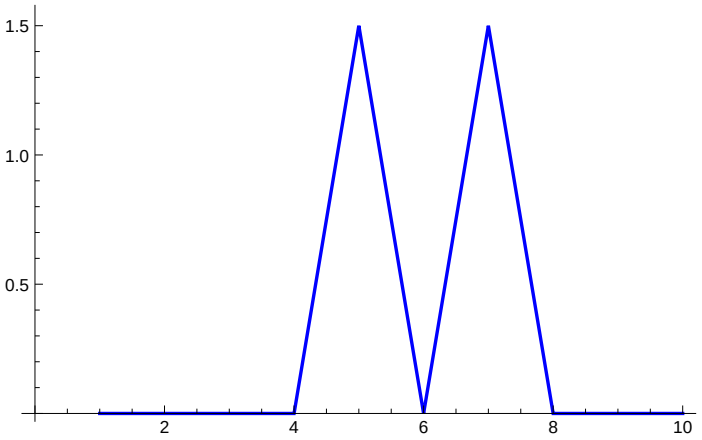
Beispiel 1: Wir betrachten eine DFT für f[t]=Sin[8 π t] +2 Sin[28 π t] mit T=1 s und N=10 und plotten einen interpolierenden Polygonzug zwischen den Absolutbeträgen der erhaltenen DFT-Koeffizienten. In der *Mathematica*-Numerierung gehört der Koeffizient mit der Nummer n zum DFT-Koeffizienten mit der Nummer n-1, wenn wir auf die Notation in [1] Bezug nehmen. Hier sind die Koeffizienten mit den Nummern 4 und 8 nahe Null. Sie kommen durch numerische Rundungsfehler zustande und müssten exakt Null sein. Die Werte 1.5 für die Nummern 5 und 7 täuschen wegen des Alias-Effekts eine einzige Schwingung von 4 Hz mit der Amplitude 3 vor, die beiden Schwingungen mit 4 Hz und mit 14 Hz sind in dieser DFT nicht zu unterscheiden, ihre Amplituden summieren sich dort wegen des “undersampling”. Die Symmetrie des DFT-Spektrums lässt sich ebenfalls mit dem Alias-Effekt erklären (vgl. [1], S. 67-68).

Im Beispiel wird eine Liste von Abtastwerten erzeugt, die mit dem *Mathematica*-Befehl Fourier einer FFT unterworfen wird. Danach plotten wir eine Darstellung des DFT-Betragsspektrums. Dargestellt wird mit ListLinePlot ein Polygonzug, der die Beträge der Spektralwerte verbindet.

In[1]:=
```
bsp1 = Table[Sin[8 π n / 10] + 2 Sin[28 π n / 10], {n, 0, 9}];
Abs[Fourier[bsp1, FourierParameters → {-1, -1}]]
ListLinePlot[%, PlotStyle → Directive[Blue, Thickness[0.005]]]
```

Out[2]=
$$\{0.,\ 0.,\ 1.37011 \times 10^{-16},\ 0.,\ 1.5,\ 0.,\ 1.5,\ 0.,\ 1.37011 \times 10^{-16},\ 0.\}$$

Out[3]=



Um eine eindeutige Frequenz-Zuordnung in Frequenzbändern der Form [mN/(2T), (m+1)N/(2T)] zu erreichen, verwendet man in der Signalverarbeitung Bandpassfilter, die für gewählte $m \in \mathbb{N}_0$ nur Signalanteile im gewünschten Frequenzband passieren lassen.

Beispiel 2: Bei EMV-Abstrahlungsmessungen im GHz-Bereich vermeidet man Abtastraten von mehreren Gigasamples pro Sekunde einfach durch solche Bandpassfilter, etwa mit einer Bandbreite von 1 MHz in Subbändern, und erreicht mit einer 512-Punkte-DFT mit Beobachtungsdauer 0.2 ms pro Frequenzband, also ca. 2.5 MHz Abtastfrequenz, eine Frequenzauflösung von etwa 5 kHz. Die Analysen der Subbänder können dann zu einem Gesamtbild zusammengesetzt werden (in English könnten wir etwa sagen: "*Undersampling Solution for High Frequency FFT Analysis*").

Wir betrachten das DFT-Betragsspektrum eines solchen Beispiels in einem einzelnen Subband der Breite 1 MHz, welches zeigt, ***dass der Alias-Effekt genau beachtet werden muss, wenn man Aussagen mit richtigen Frequenz-Zuordnungen machen will***. Vorausgesetzt wird, dass das Signal im Frequenzband [1GHz, 1GHz + 1MHz] liegt, etwa entstanden am Ausgang eines entsprechenden Bandpass-Filters.

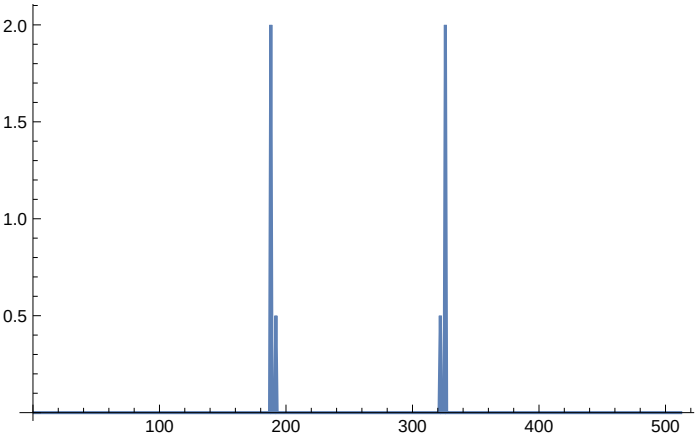
In[4]:=
```
T = 0.2 × 10^(-3);
NN = 512; (* T Abtastdauer, NN Anzahl der Abtastwerte
  (statt N wie oben, da N von Mathematica geschützt ist) *)
bsp2 = Table[Cos[2 Pi (10^9 + 5000) n T / NN] +
     4 Cos[2 Pi (10^9 + 25 000) n T / NN], {n, 0, 511}];
```

Wir betrachten also die *Superposition zweier hochfrequenter Schwingungen im GHz - Bereich*. Nun unsere DFT mit FourierParameters in *Mathematica* so, dass der gleiche Vorfaktor 1/NN wie

in [1] verwendet wird. Danach plotten wir das gesamte DFT-Betragsspektrum als Polygonzug sowie die relevanten Teile davon und sehen, wie die Frequenz-Zuordnung im Beispiel zu geschehen hat. *Wir haben 512 Abtastwerte des Signals in der Zeit T=0.2 ms genommen*.

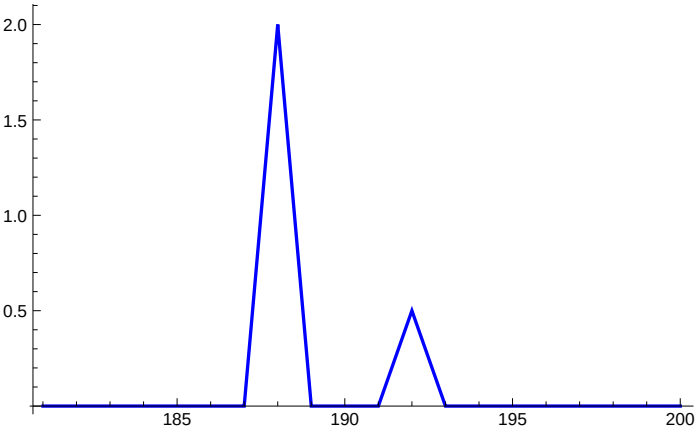
In[6]:=
```
absdft = Abs[Fourier[bsp2, FourierParameters → {-1, -1}]];
ListLinePlot[%, PlotRange → All]
```

Out[7]=



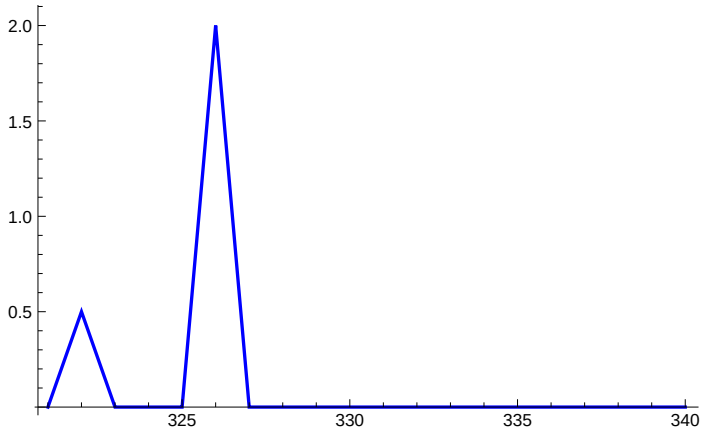
In[8]:=
```
list1 = Table[absdft[[n]], {n, 181, 200}];
(* Extraction von Teilen der Liste mit DFT-Werten*)
list2 = Table[absdft[[n]], {n, 321, 340}];
ListLinePlot[list1, PlotStyle → Directive[Blue, Thickness[0.005]],
 PlotRange → All, DataRange → {181, 200}]
```

Out[10]=



In[11]:=
```
ListLinePlot[list2, PlotStyle → Directive[Blue, Thickness[0.005]],
 PlotRange → All, DataRange → {321, 340}]
```

Out[11]=



Frequenz - Zuordnung: Die beiden “Peaks” der Höhe 2 gehören zur Schwingung mit Frequenz 10^9+25000 Hz und Amplitude 4. **Sie haben in *Mathematica* im Vergleich zur Notation in [1] eine um 1 erhöhte Nummer.** Die Peaks mit Nummern 188 und 326 gehören daher - hier kommt der Alias-Effekt zum Tragen - zu 4 Cos[2 π(10^9 + 25 000)t] = 4 Cos[(**325**+390*NN)ω_0t] mit ω_0=2π/T=2π*5*10^3 1/s, T=0.2*10^{-3} s Beobachtungsdauer wie oben, denn 187=-325+512 und (**325**+390*512) $*5*10^3$ = 1000025000. ***Der Peak mit der Nummer 188 in Mathematica gehört dann als Alias zum Schwingungsanteil***

2 Exp[-ί (325+390 NN)ω_0t]=2 Exp[+ί (187-391 NN)ω_0t]

in der Fourierreihe des T-periodisch fortgesetzten Signals.

Ganz analog kann man den beiden Peaks mit den Nummern 192 und 322 mit der Höhe 1/2 die andere im Signal enthaltene Schwingungsfrequenz zuordnen. Führen Sie bitte selbst die analoge kleine Rechnung durch.

Insbesondere bemerken wir, dass die den positiven Signalfrequenzen zugehörigen Werte mit den Nummern 322 und 326 in der oberen Hälfte des DFT-Spektrums liegen, während die mit den Nummern 188 und 192 als “Aliaswerte” von Anteilen der (komplexen) Fourierreihe mit negativen Frequenzen zu verstehen sind (s.o.).
Der Aliaseffekt erfordert also mitunter etwas Überlegung, um die DFT-Linienspektren den richtigen Frequenzen zuzuordnen. Man kann sich bei Bedarf aber ein kleines Programm dafür schreiben (Aufgabe unten).

Übung: *Überlegen Sie (ggf. mit Bleistift und Papier), welche Nummern die Peaks einer 512-Punkte-DFT mit Mathematica bei Beobachtungsdauer T = 0.2$*10^{-3}$ s für eine* Schwingung mit der Frequenz 1001060000 *Hz haben?*
Aufgabe: *Schreiben Sie ein kleines Programm, welches eine Abbildung eines wie oben berechneten DFT-Linienspektrums auf das zugehörige Linienspektrum mit den korrekten Frequenz-Zuordnungen realisiert und grafisch entsprechend über einer Frequenzachse darstellt.*

Dass es noch weitere beachtenswerte Aspekte gibt, zeigen die folgenden Beispiele.

1.2 Anwendung: Schätzung von Signalspektren, Aliasing, Leakage, Zeitfenster

Wie schon aus den obigen Beispielen ersichtlich kann die DFT verwendet werden, um Spektren unbekannter Signale zu schätzen, wenn Kenntnisse über deren Bandbreite vorliegen. Dabei sind i.A. “Abschneide-Effekte” durch das bei der DFT verwendete Zeitfenster zu beachten (vgl. [1], Abschnitt 11.4). Sie sind u.a. bedingt durch die Unschärferelation (**Heisenbergsche Unschärferelation**, vgl. [1], Abschnitt 11.2) für das Zeitdauer-Bandbreite-Produkt.

Wir betrachten Signale f, die auf [0,T[stetig sind und mit der Vereinbarung f(0)=f(T) eine stückweise stetig differenzierbare T-periodische Fortsetzung auf ganz $\mathbb{R}$ besitzen. Bezeichnet w_T das Rechteck-Fenster UnitStep[t]-UnitStep[t-T] der Dauer T, dann ist der DFT-Koeffizient $C_k(\mathrm{fw}_T)$ einer N-Punkte-DFT für Signale f wie oben wegen des Aliaseffekts im Vergleich mit den Fourierkoeffizienten $c_k(\mathrm{fw}_T)$ der Fourierreihendarstellung von fw_T gegeben durch

$$C_k(\mathrm{fw}_T) = \sum_{m=-\infty}^{+\infty} c_{k+mN}(\mathrm{fw}_T) + \frac{1}{2N}(f(0)-f(T-)). \qquad (***)$$

Fazit: Hat die T-periodische Fortsetzung von fw_T bei T eine Sprungstelle oder besitzt f Schwingungsanteile mit einer Kreisfrequenz $\omega \neq 2\pi k/T$, $k \in \mathbb{Z}$, dann treten Verfälschungen in der DFT im Vergleich zum wahren Signalspektrum auf (siehe [1], S. 337ff).
Die Verfälschungen werden in der engl. Literatur als **Aliasing** und als **Leakage** bezeichnet.

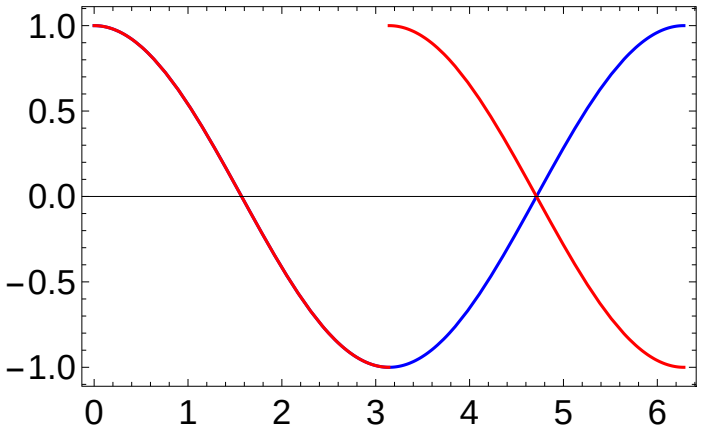
Wir betrachten ein einfaches Beispiel:

Beispiel 3: f1(t)=Cos[t] hat **für $T=\pi$** eine unstetige T-periodische Fortsetzung; für die Funktion f2(t)=-Cos(t/2)+Cos(t)/2 ist die T-periodische Fortsetzung von $f2 \cdot w_T$ stetig, f2 selbst ist aber nicht T-periodisch. Wir betrachten zunächst Ausschnitte der Graphen von f1 und $f1 \cdot w_T$, T-periodisch fortgesetzt, und Ausschnitte der Graphen von f2 und der T-periodischen Fortsetzung von $f2 \cdot w_T$, jeweils mit dem Rechteck-Fenster w_T.

In[12]:=
```
T := Pi; f1[t_] := Cos[t]; wi[t_] := UnitStep[t] - UnitStep[t - T];
f2[t_] := -Cos[t / 2] + Cos[t] / 2; f1w[t_] := f1[t] wi[t];
f2w[t_] := f2[t] wi[t];
```

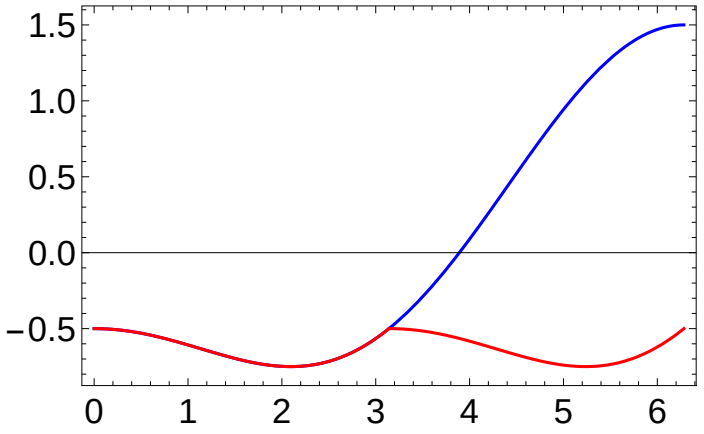
In[14]:=
```
p1 := Plot[f1[t], {t, 0, 2 T}, PlotRange → All, Frame → True,
  FrameStyle → Directive[Black, FontSize → 18, FontWeight → Plain],
  PlotStyle → {Blue, Thickness[0.005]}]
p2 := Plot[f1w[t] + f1w[t - T], {t, 0, 2 T}, PlotRange → All, Frame → True,
  FrameStyle → Directive[Black, FontSize → 18, FontWeight → Plain],
  PlotStyle → {Red, Thickness[0.005]}]
Show[{p1, p2}] (* Hier der Unterschied zwischen f1 (blau) und der T-
 periodischen Fortsetzung von f1·wT (rot) *)
```

Out[16]=



In[17]:=
```
p3 := Plot[f2[t], {t, 0, 2 T}, PlotRange → All, Frame → True,
  FrameStyle → Directive[Black, FontSize → 18, FontWeight → Plain],
  PlotStyle → {Blue, Thickness[0.005]}]
p4 := Plot[f2w[t] + f2w[t - T], {t, 0, 2 T}, PlotRange → All, Frame → True,
  FrameStyle → Directive[Black, FontSize → 18, FontWeight → Plain],
  PlotStyle → {Red, Thickness[0.005]}]
Show[{p3, p4}] (* Hier der Unterschied zwischen f2 (blau) und der T-
 periodischen Fortsetzung von f2·wT (rot) *)
```

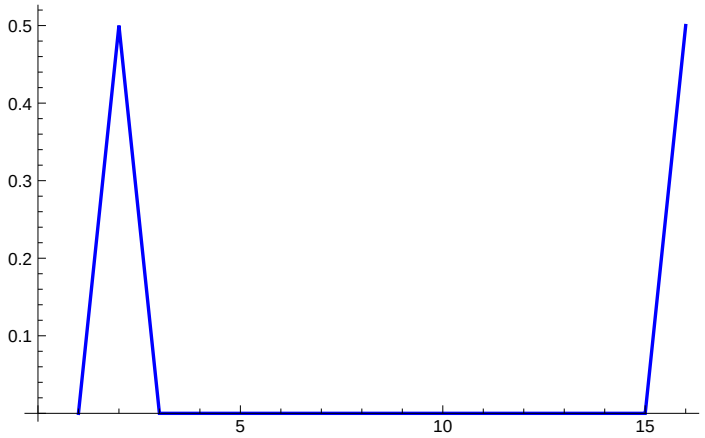
Out[19]=



Selbstverständlich sind daher nun große Unterschiede der DFT-Spektren und der wahren Spektren der so mit einem Rechteck-Fenster betrachteten periodischen Signale zu erwarten:
Zunächst das mit dem Rechteck-Fenster der Dauer T=2π mit N=16 Punkten berechnete DFT-Betragsspektrum von f1. Damit sehen wir die **Signalkreisfrequenz** von 1 [1/s] deutlich. Sie entspricht in der *Mathematica*-Nummerierung den Peaks bei Nummer 2 und 16.

In[20]:=
```
data1 = Table[N[f1[ 2 π n / 16]], {n, 16}];
absdft1 = Abs[Fourier[data1, FourierParameters → {-1, -1}]];
ListLinePlot[absdft1, PlotRange → All, PlotStyle → {Blue, Thickness[0.005]}]
```

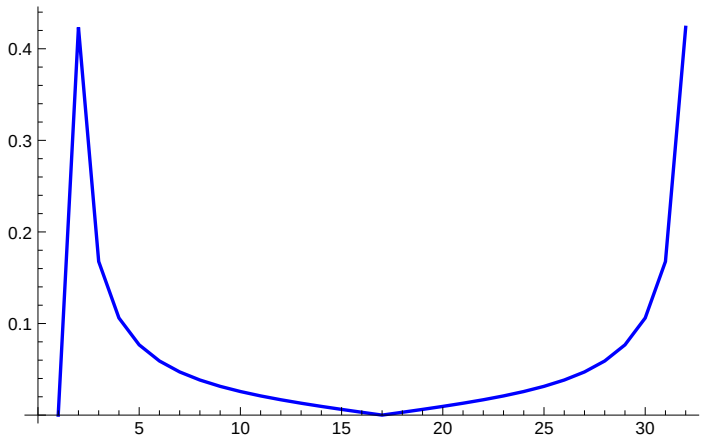
Out[22]=



Nun eine DFT von f1 mit T=π, N=32 Abtastpunkten, und damit das Betragsspektrum von f1$\cdot w_T$ mit dem Rechteckfenster. Man erkennt deutlich die Alias- und Leakage-Effekte, die Schwingungsanteile mit Kreisfrequenzen ≠ 1 [1/s] vermuten lassen, wenn man naiv mit der DFT umgeht und das Ergebnis zur Signal-interpretation für bare Münze nimmt.

In[23]:=
```
data2 = Table[N[f1w[ T n / 32]], {n, 32}];
absdft2 = Abs[Fourier[data2, FourierParameters → {-1, -1}]];
ListLinePlot[absdft2, PlotRange → All, PlotStyle → {Blue, Thickness[0.005]}]
```

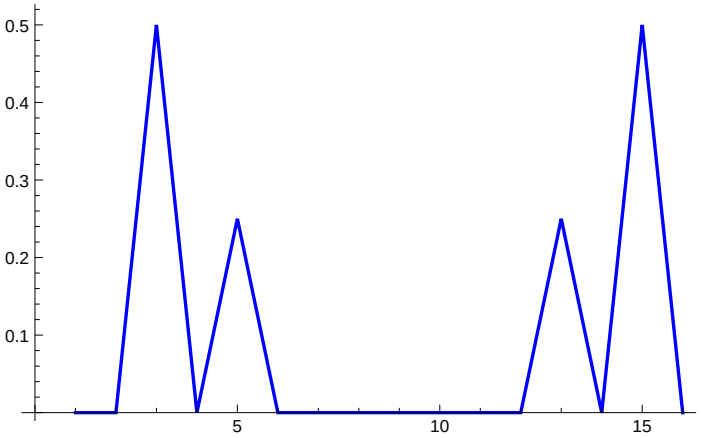
Out[25]=



Nun die entsprechenden Darstellungen für f2 und f2w, **im ersten Fall mit T=8π**, also einer exakten Periode von f2, **im zweiten Fall mit T=π**, jeweils mit dem Rechteck-Fenster und N=16 Punkten für die DFT. Im 2. Fall ist das eigentliche Signalspektrum schwerlich noch richtig zu schätzen (Man beachte etwa den hohen “Gleichanteil” mit der Nummer 1 entsprechend der Grafik von f2w oben).

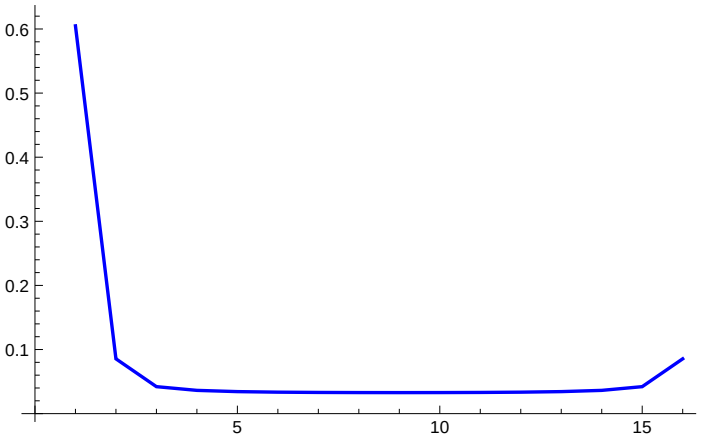
In[26]:=
```
data3 = Table[N[f2[ 8 π n / 16]], {n, 16}];
absdft3 = Abs[Fourier[data3, FourierParameters → {-1, -1}]];
ListLinePlot[absdft3, PlotRange → All, PlotStyle → {Blue, Thickness[0.005]}]
```

Out[28]=



In[29]:=
```
data4 = Table[N[f2w[T n / 16]], {n, 16}];
absdft4 = Abs[Fourier[data4, FourierParameters → {-1, -1}]];
ListLinePlot[absdft4, PlotRange → All, PlotStyle → {Blue, Thickness[0.005]}]
```

Out[31]=



Ebenso problematisch ist das Aufdecken von Schwingungsanteilen eng benachbarter Frequenzen (genügend hohe Auflösung), insbesondere wenn solche Anteile sehr unterschiedliche Amplituden aufweisen und kleine Amplitudenwerte neben größeren durch Verfälschungseffekte der DFT verdeckt werden, oder wenn das beobachtete Signal von kurzzeitigen Störungen überlagert ist. Man vgl. hierzu die Beispiele in [1], S. 339-340 und den Zusammenhang zwischen Glattheitseigenschaften des Signals einerseits und dem Abfallen des Betragsspektrums periodischer Funktionen andererseits (siehe erstes Notebook auf der Homepage dieser Serie und [1], Abschnitt 4.5).
In der Praxis versucht man, möglichst gute Spektralschätzungen durch eine **möglichst lange Beobachtungsdauer**, durch **hohe Abtastfrequenzen** und eine entsprechend **hohe Anzahl von Abtastwerten** und durch **Verwendung von Gewichtsfunktionen, sog. Zeitfenstern**, zu erhalten. Dazu im nächsten Abschnitt.

Wir merken uns: Eine Erhöhung der Beobachtungsdauer (und damit verbunden Erhöhung der Anzahl von Abtastwerten) verbessert die Frequenzauflösung, eine Erhöhung der Abtastfrequenz (wiederum mit entsprechender Erhöhung der Anzahl von Abtastwerten)

vergrößert die erfasste Bandbreite. Beide Effekte wirken Verfälschungen durch Aliasing und Leakage entgegen. Ein weiteres Hilfsmittel für eine solche Verbesserung ist der Gebrauch von Gewichtsfunktionen im Zeitbereich, sog. Zeitfenstern:

Milderung von Alias- und Leakage-Effekten durch Verwendung von Zeitfenstern um den Preis von Amplitudenverfälschungen

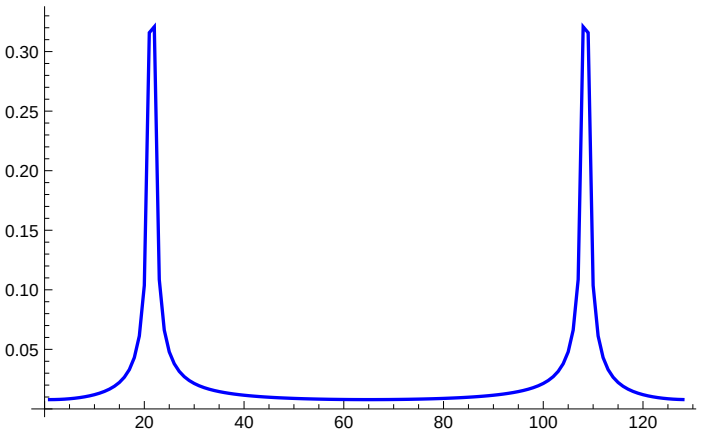
Zur Veranschaulichung betrachten wir eine Superposition f3 zweier Schwingungen mit eng benachbarten Frequenzen und sehr unterschiedlichen Amplituden. Das betrachtete Signal liegt im Frequenzband bis 30 Hz. Wir verwenden 64 Punkte pro s für die DFT.

Wir berechnen und plotten die DFT-Amplitudenspektren für 3 Varianten, dargestellt wie vorher durch einen Polygonzug, der die Spektralwertbeträge verbindet:

1) Eine DFT mit Dauer T=2 s und 64 Abtastwerten pro s: **Die unterschiedlichen Signalfrequenzen sind kaum wahrzunehmen.** Wir plotten wieder das Betragsspektrum.

In[32]:=
```
f3[t_] := Cos[2 Pi 10.25 t] + 0.03 Cos[2 Pi 12 t]
data5 = Table[N[f3[n / 64]], {n, 128}];
absdft5 = Abs[Fourier[data5, FourierParameters → {-1, -1}]];
ListLinePlot[absdft5, PlotRange → All, PlotStyle → {Blue, Thickness[0.005]}]
```

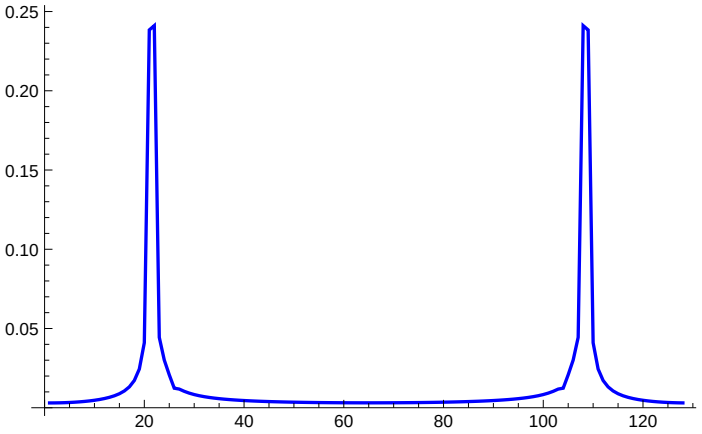
Out[35]=



2) Eine analoge DFT, wobei das Signal mit dem Hann-Fenster w(t)=0.5-0.5 cos(2πt/T) für 0≤t≤T, hier mit T=2, gewichtet wird. Dies beseitigt wegen w(0)=w(4π)=0 die Sprungstellen in der T-periodischen Fortsetzung von f3·w und mildert damit die Alias- und Leakage-Effekte in der DFT. **Die 12 Hz-Signalfrequenz (Peak 25) lässt sich in der DFT nun schon eher vermuten**.

In[36]:=
```
whann[t_] = 0.5 - 0.5 Cos[2 Pi t / T];
T = 2; f4[t_] := f3[t] whann[t];
data6 = Table[N[f4[n / 64]], {n, 128}];
absdft6 = Abs[Fourier[data6, FourierParameters → {-1, -1}]];
ListLinePlot[absdft6, PlotRange → All, PlotStyle → {Blue, Thickness[0.005]}]
```

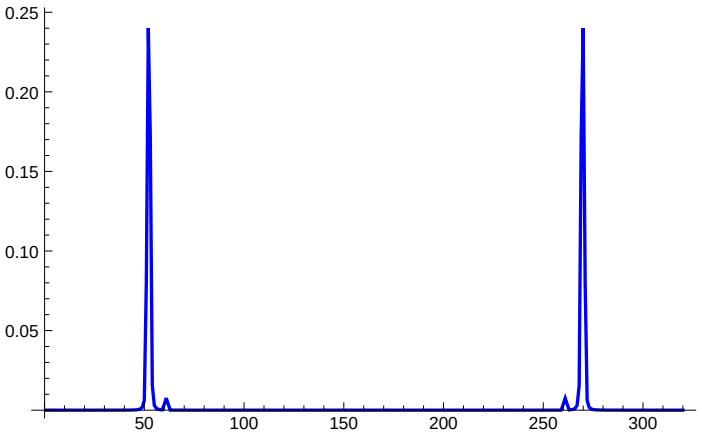
Out[40]=



3) Eine DFT mit dem Hann-Fenster, längerer Beobachtungsdauer T=5 s und höherer Anzahl von Abtastwerten bei gleicher Abtastfrequenz. Wieder das Betragsspektrum, das die DFT liefert, dargestellt:

In[41]:=
```
T = 5;
f4[t_] = f3[t] (0.5 - 0.5 Cos[2 π t / T]);
data7 = Table[N[f4[n / 64]], {n, 320}];
absdft7 = Abs[Fourier[data7, FourierParameters → {-1, -1}]];
ListLinePlot[absdft7, PlotRange → All, PlotStyle → {Blue, Thickness[0.005]}]
```

Out[43]=



Fazit: Erhöhung der Beobachtungsdauer, ggf. der Abtastfrequenz und Gebrauch von Zeitfenstern kann die Aufdeckung von Signalfrequenzen verbessern. ***Als Preis dafür zahlt man, dass das Signal durch die Gewichtsfunktion im Zeitverlauf unterschiedliche Dämpfungen erfährt, so dass die in der DFT angezeigten Amplitudenwerte dann in der Regel an Aussagekraft verlieren***. Je länger die Beobachtungsdauer und je höher die Anzahl der Abtastwerte pro s (und damit die Frequenzauflösung der DFT), desto eher kann man auf Gewichtsfunktionen auch verzichten (siehe Gleichung (11.2) in [1] bzw. Gleichung (***) weiter oben) und einfach

mit dem Rechteck-Fenster arbeiten.
Es sind viele unterschiedliche Fensterfunktionen je nach Zweck in der Praxis der diskreten Signalverarbeitung in Gebrauch. Referenzen zu Arbeiten hierüber findet man in [1].

1.3 Näherung der Fouriertransformierten von zeitbegrenzten Signalen mit Hilfe der DFT durch sogenanntes "Zeropadding" und "Upsampling" im Zeitbereich mit Hilfe von Zeropadding

Vielfach wird DFT in der Praxis verwendet, um Näherungen für die Fouriertransformierten von i.A. auch nicht-periodischen Signalen f zu verwenden. Es ist sinnvoll, sich dabei auf zeitbegrenzte Signale (d.h. solche mit einem beschränkten Träger) zu beschränken (vgl. weiter unten). Für solche, hier als stückweise stetig differenzierbar vorausgesetzte Funktionen f mit Träger etwa in [0,T], Abtastwerte F_k ihrer Fouriertransformierten an den Stellen $2\pi k/T$, $k \epsilon \mathbb{Z}$, und die Fourierkoeffizienten c_k der T-periodischen Fortsetzung von f gilt ([1], 10.5, S. 290)

$$F_k = T\, c_k \quad (****)$$

Man kann daher die DFT verwenden, um Abtastwerte der Fouriertransformierten von f näherungsweise zu berechnen, und kann dann die DFT-Werte durch Interpolation zur Veranschaulichung der Fouriertransformierten von f benutzen. Durch eine Erhöhung der Abtastdauer über den Träger von f hinaus, d.h. durch Anfügen vieler Nullen bei den Abtastwerten (bei gleichbleibender Abtastfrequenz), verbessert man die Approximation von f (gesehen als Funktion auf ganz $\mathbb{R}$) durch den Signalausschnitt und dadurch mit der Frequenzauflösung der DFT die Näherung durch Interpolation. Die Interpolationspunkte lauten also $(2\pi k/T, T\, c_k)$. (Man vgl. hierzu auch [1], Abschnitt 9.7, Bemerkung auf S. 259.)

Wir betrachten als Beispiel die gerade reelle Dreieckfunktion als Funktion der Zeit t

f5[t] = (1-|t|) (UnitStep[t+1]-UnitStep[t-1]).

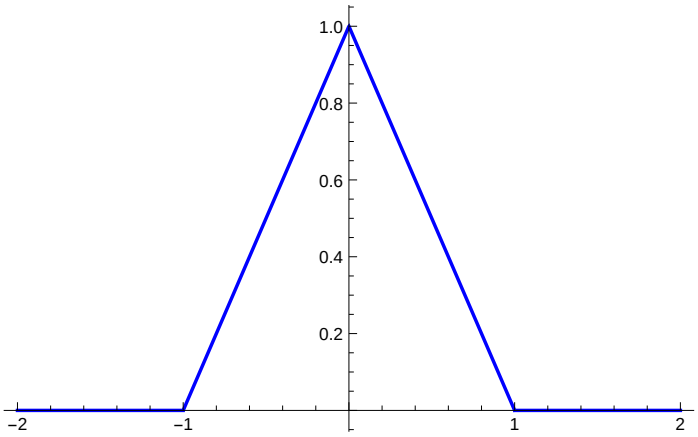
Wir plotten zunächst diese Funktion f5 und ihre (rellwertige, gerade) Fouriertransformierte als Funktion der Kreisfrequenz ω für $\omega \epsilon [0, 4\pi]$.

Aus Symmetriegründen genügte es, nur die halbseitige Fouriertransformierte für $\omega \geq 0$ zu zeichnen. (Die Fouriertransformierte ist reell, gerade und nicht-negativ, siehe unten.)

Anmerkung: *Nachfolgend wird die Option* ***FourierParameters→{1,-1}*** *für die Fouriertransformation verwendet, damit diese wie in [1] definiert wird.*
Mathematica-Default ist sonst FourierParameters→{0,1}.

In[44]:=
```
f5[t_] = (1 - Abs[t]) (UnitStep[t + 1] - UnitStep[t - 1]) ;
Plot[f5[t], {t, -2, 2}, PlotRange → All, PlotStyle → {Blue, Thickness[0.005]}]
```

Out[45]=



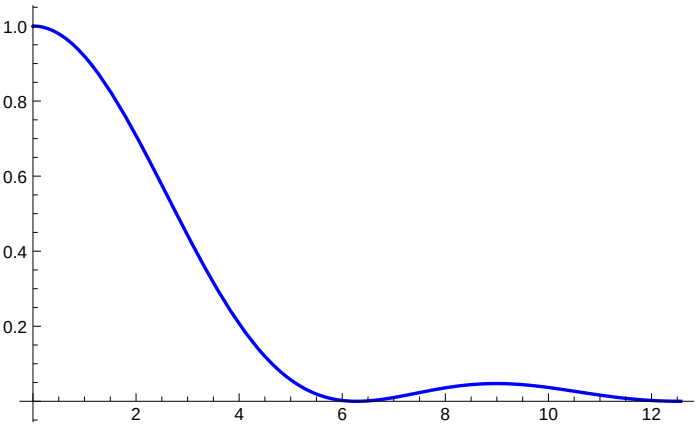
In[46]:=
```
f5Fou = FullSimplify[FourierTransform[f5[t], t, ω, FourierParameters → {1, -1}]]
```

Out[46]=

$$\frac{2 - 2\,\text{Cos}[\omega]}{\omega^2}$$

In[47]:=
```
plot14 =
 Plot[f5Fou, {ω, 0, 4 Pi}, PlotRange → All, PlotStyle → {Blue, Thickness[0.005]}]
```

Out[47]=



Nun eine erste Näherung der Fouriertransformierten mit einer 8-Punkte-DFT **mit Werten aus [-1,1]**, Schrittweite 1/4 und Interpolation der Werte durch einen Polygonzug (wie oben schon zur Darstellung der DFT-Amplitudenspektren verwendet).

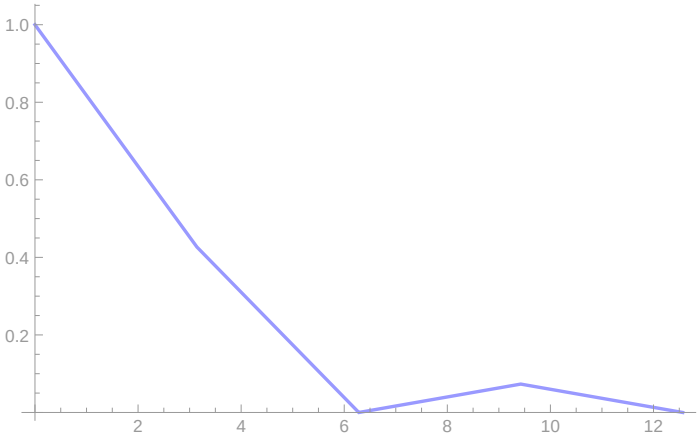
Da die DFT standardmäßig mit Abtastwerten ab t=0 definiert wurde, unser Beispiel aber nicht-verschwindende Werte auch für t<0 hat, ***müssen die entsprechenden Phasenänderungen berücksichtigt werden, wenn man hier Abtastwerte in [-1,1] nimmt****. Man kann dies im Beispiel durch Multiplikation der DFT-Koeffizienten c_k mit jeweils $Exp[\imath\, k\, \pi] = (-1)^k$ erreichen oder alternativ durch Interpolation mit den Beträgen der DFT-Koeffizienten statt mit diesen selbst. Wir verwenden nachfolgend beide Alternativen zur Veranschaulichung (vgl. auch [1], Abschnitt 4.2).*

In[48]:=
```
data8 = Table[f5[-1 + n / 4] , {n, 0, 7}];
dataphase1 = Table[(-1)^k, {k, 0, 7}];
dft8 = dataphase1 * Chop[Fourier[data8, FourierParameters → {-1, -1}]]
```

Out[49]=
```
{0.5, 0.213388, 0, 0.0366117, 0, 0.0366117, 0, 0.213388}
```

```
ListLinePlot[2 dft8[[1 ;; 5]],
 DataRange → {0, 4 Pi}, PlotStyle → {Blue, Thickness[0.005]}]
(* Im ListLinePlot-Befehl werden hier nur die
  ersten 5 Werte der Liste dft8 extrahiert *)
```

Out[50]=



Das ist eine noch recht grobe Näherung für die Fouriertransformierte, die uns interessiert. Die durch die Abtastrate erfasste Bandbreite der Fouriertransformierten als Funktion der Kreisfrequenz ω ist wie oben 4π.

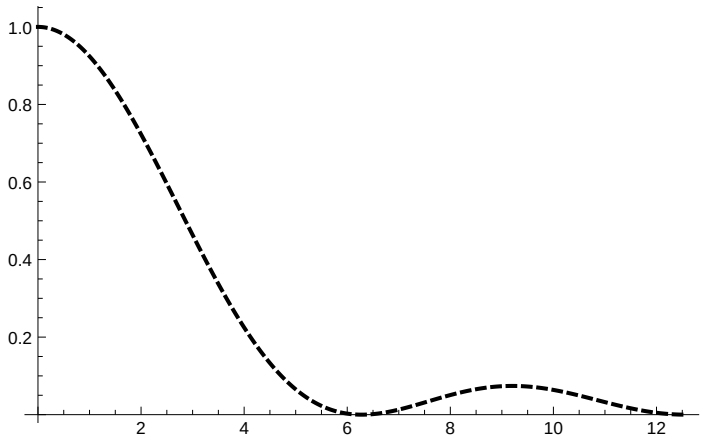
Zeropadding zur Verbesserung der Frequenzauflösung

Nun eine Näherung mit 2048 Abtastwerten, die wir durch eine DFT mit “zeropadding”, d.h. durch Anhängen von 2040 Nullen an unsere schon vorhandenen Werte, gewinnen. Wir betrachten wieder nur Werte für $\omega \in [0, 4\pi]$.

Zu beachten ist dabei: Die Abtastdauer T, die für die richtigen Frequenzzuordnungen der DFT-Koeffizienten gebraucht wird, hatten wir gerade oben T=2 s gewählt. Wir hatten 4 Werte pro s abgetastet. Die Abtastdauer, die sich dann bei nun 2048 DFT-Punkten mit der gleichen Abtastfrequenz ergibt, ist T=2048/4=512 s. Mit diesem Wert sind alle DFT-Koeffizienten für die Näherung zu multiplizieren (siehe (****)).

In[51]:=
```
data9 = Table[f5[-1 + n / 4], {n, 0, 2047}];
dft9 = Abs[Fourier[data9, FourierParameters → {-1, -1}]];
plot15 = ListLinePlot[512 * dft9[[1 ;; 1025]], DataRange → {0, 4 Pi},
  PlotStyle → {Dashed, GrayLevel[0], Thickness[0.006]}]
```

Out[53]=

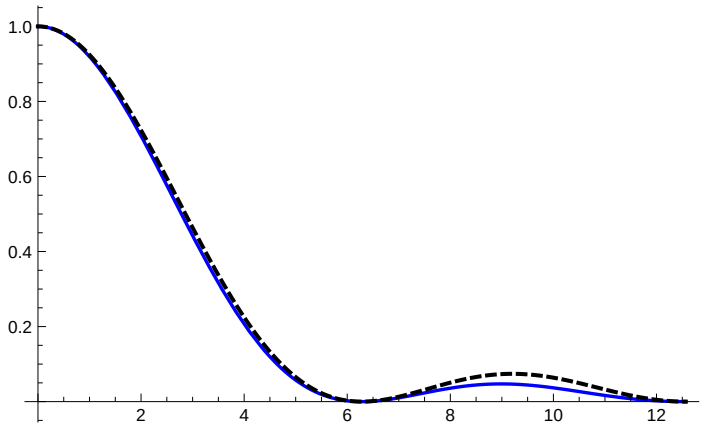


Dies ist eine schon ganz passable Näherung für die gewünschte Fouriertransformierte im Bereich $[0, 4\pi]$.

Wir plotten nochmal die echte Fouriertransformierte von oben (blau) und diese Näherung (gestrichelt) in einer gemeinsamen Grafik. *Die Abweichung der Näherung kommt durch Aliaseffekte zustande, welche durch die numerische Näherung der echten Fourierkoeffizienten mit Hilfe der DFT bedingt sind. Da die Werte der betrachteten Fouriertransformierten sämtlich nicht-negativ sind, entsteht durch Aliasing eine Näherung, die etwas oberhalb der wahren Fouriertransformierten verläuft, weil dadurch ja Spektralwerte höherer Frequenzen auf solche von niedrigeren aufaddiert werden.* Während durch $(2\pi k/T, Tc_k)$ mit den Fourierkoeffizienten der T-periodischen Fortsetzung der Dreiecksfunktion deren Fouriertransformierte exakt interpoliert wird, berechnet man mit der DFT nur Näherungen solcher Fourierkoeffizienten c_k durch numerische Integration mittels Trapezregel, was dann wegen des Aliaseffekts i.A. den Verlust der exakten Interpolationseigenschaft zur Folge hat. Dies erklärt ebenso die Abweichung der Näherungsfunktion von der Fouriertransformierten.

In[54]:=
```
Show[{plot14, plot15}]
```

Out[54]=



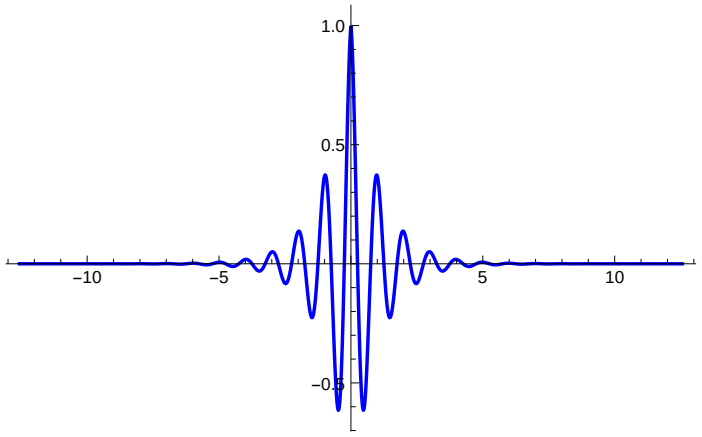
Anmerkung: Bei Funktionen mit unbeschränktem Träger bringt zeropadding i.A. keine Verbesserung der Näherung für die Fouriertransformierte mit Hilfe einer DFT mit Abtastwerten aus einem endlichen Signalausschnitt.

Wir betrachten dazu einen Signalausschnitt einer mit Exp[-|t|] amplitudenmodulierten Cos-Funktion: Die Fouriertransformierte ist eine gerade, reellwertige, positive Funktion (vgl. [1], S. 233 oder weiter unten).
Wir lassen *Mathematica* die Fouriertransformierte berechnen und auch Näherungen für die positive Maximalstelle und den Maximalwert dort. Dann tasten wir 1024 Werte über T=2 s von t=-1 bis t=+1 ab und plotten eine "DFT-Näherung" der Fouriertransformierten als gestrichelte Liniengrafik und zum Vergleich die echte Fouriertransformierte im Bereich [0, 12π] in blau.
Die Näherung ist insbesondere um die Maximalstelle der Fouriertransformation herum ziemlich erbärmlich.

(Man überlege sich hier insbesondere nochmals die Option DataRange im folgenden Plot-Befehl für plot16, um die richtige Frequenzzuordnung zu erhalten.)
Zuerst die betrachtete Funktion:

In[55]:=
```
f7[t_] = Exp[-Abs[t]] * Cos[2 Pi t];
Plot[f7[t], {t, -4 Pi, 4 Pi},
 PlotRange → All, PlotStyle → {Blue, Thickness[0.005]}]
```

Out[56]=

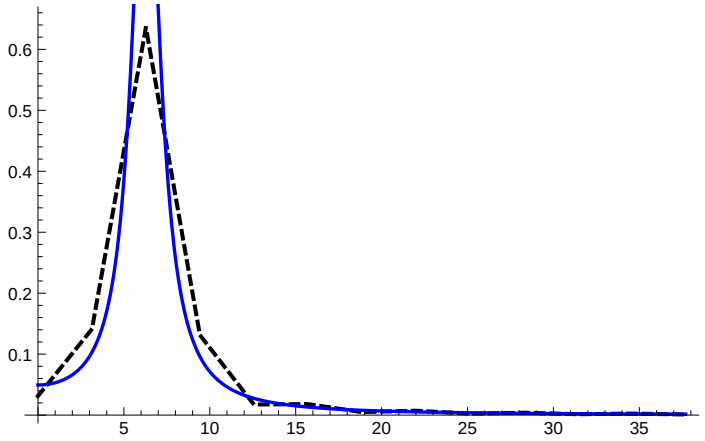


In[57]:=
```
f7Fou = FullSimplify[FourierTransform[f7[t], t, ω, FourierParameters → {1, -1}]]
(*wir lassen die FT berechnen*)
FindMaximum [f7Fou, {ω, 6}] (* und wir lassen Näherungen
 für die Maximalstelle und den Maximalwert berechnen*)
data10 = Table[f7[-1 + n / 512], {n, 0, 1023}];
dft10 = Abs[Fourier[data10, FourierParameters → {-1, -1}]];
plot16 = ListLinePlot[2 * dft10[[1 ;; 13]], DataRange → {0, 12 Pi},
   PlotRange → All, PlotStyle → {Dashed, GrayLevel[0], Thickness[0.006]}];
plot17 = Plot[f7Fou, {ω, 0, 12 Pi}, PlotRange → All,
   PlotStyle → {Blue, Thickness[0.005]}];
Show[{plot16, plot17}]
```

Out[57]=
$$\frac{1}{1+(-2\pi+\omega)^2}+\frac{1}{1+(2\pi+\omega)^2}$$

Out[58]=
$$\{1.00629,\ \{\omega \to 6.28269\}\}$$

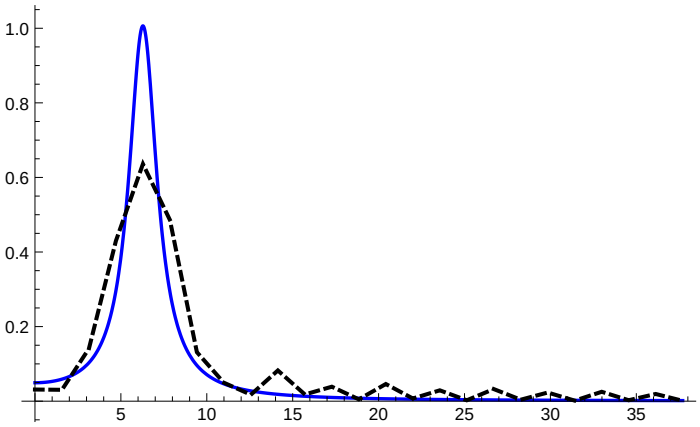
Out[63]=



Diese Näherung wird auch durch Zeropadding nicht besser, sondern eher noch schlechter. Als Beispiel werden nachfolgend 1024 Nullen für die DFT angefügt und das Ergebnis nochmal im Plot dargestellt.

In[64]:=
```
data11 = Table[f7[-2 + n / 512]
    (UnitStep[(n - 1024) / 512 + 1] - UnitStep[(n - 1024) / 512 - 1]), {n, 0, 2047}];
dft11 = Abs[4 * Fourier[data11, FourierParameters → {-1, -1}]];
plot18 = ListLinePlot[dft11[[1 ;; 25]], DataRange → {0, 12 Pi},
   PlotRange → All, PlotStyle → {Dashed, GrayLevel[0], Thickness[0.006]}];
Show[{plot17, plot18}]
```

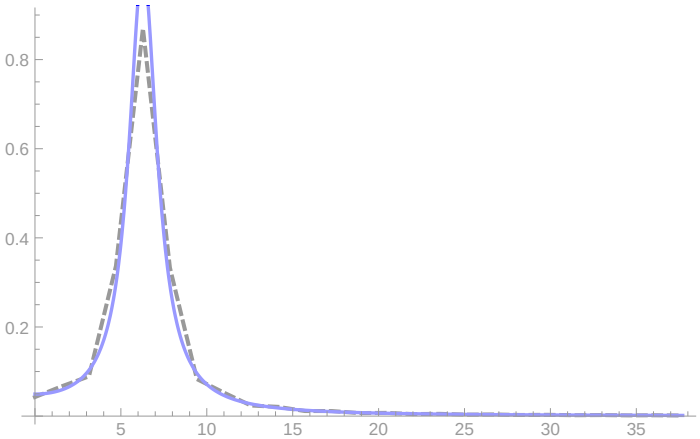
Out[67]=



Natürlich können wir auch hier wieder durch Erhöhung der Abtastrate, d.h. durch eine Vergrößerung der Bandbreite unserer DFT die Aliaseffekte abmildern und dadurch eine Verbesserung der Näherung erreichen, weil dann geringere Anteile, die da nicht hingehören, in unseren DFT-Koeffizienten aufaddiert werden. Auch dazu, weil es mit *Mathematica* schnell zu machen ist, noch eine Veranschaulichung. Wir nehmen doppelt soviele “Samples” wie zuletzt im gleichen Zeitraum von 4 s. Das Ergebnis sieht dann etwas besser aus.

```
data12 = Table[f7[-2 + n / 1024]
    (UnitStep[(n - 2048) / 2048 + 1] - UnitStep[(n - 2048) / 2048 - 1]), {n, 0, 4095}];
dft12 = Abs[Fourier[data12, FourierParameters → {-1, -1}]];
plot18 = ListLinePlot[4 * dft12[[1 ;; 25]], DataRange → {0, 12 Pi},
   PlotRange → All, PlotStyle → {Dashed, GrayLevel[0], Thickness[0.006]}];
Show[{plot18, plot17}]
```

Out[71]=



Zeropadding zur Verbesserung der Zeitauflösung:

Beispiel zum “Upsampling” mittels Zeropadding im DFT-Spektrum eines abgetasteten trigonometrischen Polynoms.

Eine solches “Upsampling” wird z.B. bei OFDM-Übertragungsverfahren verwendet, um aus diskreten Abtastwerten mittels Tiefpassfilterung eine bessere Näherung für das zugrundeliegende Analogsignal - das man eigentlich übertragen möchte - zu erhalten. Gegeben sind dabei gewisse Abtastwerte des Analogsignals und man möchte daraus weitere Zwischenwerte gewinnen. Vorausgesetzt wird dabei die Kenntnis der Bandbreite des zugrunde liegenden Analogsignals, das - wie beim OFDM - als trigonometrisches Polynom mit bekannter Grundfrequenz angenommen sei, und eine DFT der vorhandenen Abtastwerte, welche die Bandbreite dieses Signals abdeckt. (Man vgl. mein Notebook Nr.15 zu OFDM unter der gleichen URL wie hier.)

Ich wähle als einfaches Beispiel einen Zeitausschnitt einer Superposition von 2π-periodischen Cos- und Sinus-Funktionen als Analogsignal, das ich durch ein Rechteckfenster in der Zeit von Null bis T=2π begrenze.
Das Signal gehört dann - auf ganz $\mathbb{R}$ betrachtet - zu $L^2(\mathbb{R})$, auf [0,T] betrachtet zu $L^2([0, T])$.
Es ist aber wegen der Multiplikation mit der Rechteckfunktion **nicht bandbegrenzt**.
Weitere Beispiel finden Sie in meinem Notebook zu den OFDM-Übertragungsverfahren.
Ich nehme 9 äquidistante Abtastwerte in [0, T[im Abstand T/M wie bei einer DFT üblich.

In[72]:=
```
M = 9                (* Anzahl Abtastwerte *)
T = 2 Pi             (* Zeitdauer Abtastung *)
f[t_] = (Cos[t] + 0.5 Cos[2 t] - 1.5 Sin[3 t]) (UnitStep[t] - UnitStep[t - T] )
(* Analogsignal *)
tn[n_] = n T / M     (* Abtastzeitpunkte *)
Omega = Pi M / T     (* Bandbreite, die ein Tiefpass braucht,
um eine Näherung aus den diskreten Werten zu generieren;
siehe Shannontheorem *)
plot1 = Plot[f[t], {t, 0, T}, PlotStyle → Directive[Blue, Thickness[0.007]]]
 (* Plot f(t)  *)
```

Out[72]= 9

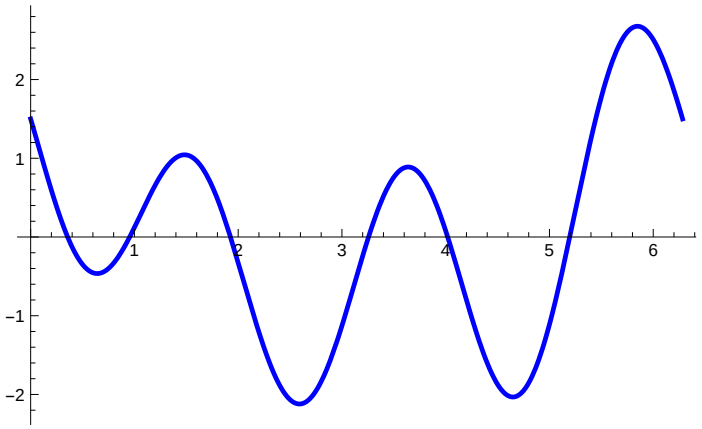
Out[73]= $2\,\pi$

Out[74]= $(\text{Cos}[t] + 0.5\,\text{Cos}[2\,t] - 1.5\,\text{Sin}[3\,t])\,(\text{UnitStep}[t] - \text{UnitStep}[-2\,\pi + t])$

Out[75]= $\frac{2\,n\,\pi}{9}$

Out[76]= $\frac{9}{2}$

Out[77]=



In[78]:=
```
idftwerte = N[Table[f[tn[n]], {n, 0, M - 1}]]
  (* IDFT = Abstastwerte der Funktion f *)
```

Out[78]= {1.5, −0.44617, 1.00284, −0.75, −1.85571, 0.742368, −0.75, −1.59524, 2.15191}

In[79]:=
```
dftwerte = Chop[Fourier[idftwerte, FourierParameters → {-1, -1}]]
  (* DFT dieser Werte *)
```

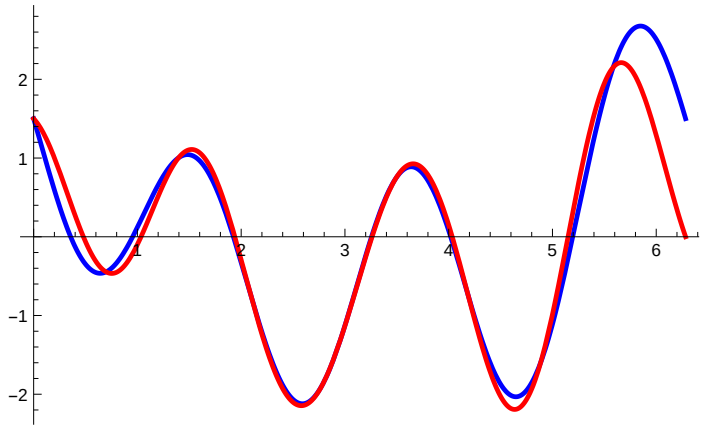
Out[79]= {0, 0.5, 0.25, 0. + 0.75 i, 0, 0, 0. − 0.75 i, 0.25, 0.5}

Wir sehen uns einmal an, wie eine aus den Abtastwerten erzeugte **Näherung gemäß Shannon-Theorem** aussehen würde, wenn wir eine zugehörige Impulsfolge durch ein ideales Tiefpassfilter schicken würden. Dabei sind die *Impulsstärken diese Abtastwerte, die Impulszeiten entsprechend dazu die Abtastzeitpunkte*.
Die Näherung mit der Shannonschen Abtastreihe ergibt die Orthogonalprojektion des Signals - **das nicht bandbegrenzt ist** - in den Raum der durch Omega bandbegrenzten Funktionen in $L^2(\mathbb{R})$.

Wir erkennen eine für t->T ziemlich schlechte Näherung bei so wenigen Abtastwerten, allerdings - und das ist in Anwendungen wichtig - **an den Abtaststellen interpolierend**.
Bei Verwendung eines realisierbaren Tiefpassfilters als Näherung für ein ideales Tiefpassfilters wie im Shannon-Theorem kämen weitere Fehler hinzu.

In[80]:=
```
plot2 =
  Plot[Sum[idftwerte[[k + 1]] Sin[Omega t - k Pi] / (Omega t - k Pi), {k, 0, M - 1}],
   {t, 0, T}, PlotRange -> All, PlotStyle -> Directive[Red, Thickness[0.007]]] ;
(* Näherung mittels Shannontheorem *)
Show[{plot1, plot2}]
```

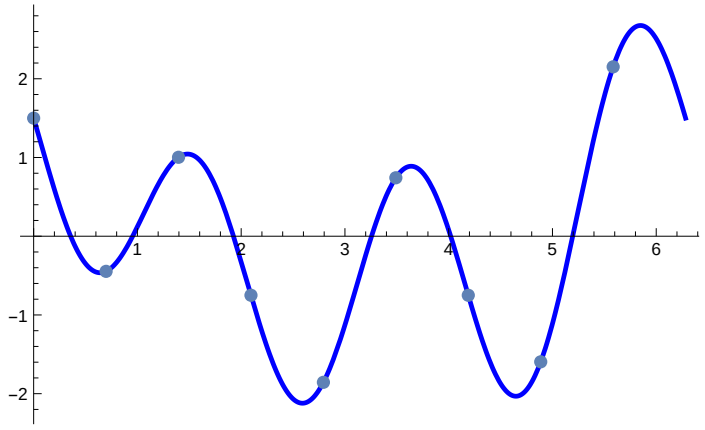
Out[81]=



Hier die Lage der 9 Interpolationsstellen als Punkte im Funktionsgraph des Signals f (t)

In[82]:=
```
plot3 = ListPlot[idftwerte, DataRange → {0, 2 Pi - T / M}];
Show[{plot1, plot3}]
```

Out[83]=



Nun Zeropadding: Wir erweitern die DFT-Werte durch **Einschieben einer Reihe von Nullen**, hier im Beispiel von M=9 Nullen. **Durch Einschieben in der Mitte der DFT-Koeffizientenliste bleiben die Frequenzzuordnungen der DFT-Koeffizienten erhalten** (vgl. [1], S. 68-69).
Eine IDFT ergibt dann weitere Interpolationsstellen von f(t), d.h. wir haben mit der IDFT ein “Upsampling” von f(t) erreicht (vgl. [1], S.74-75).

In[84]:=
```
zeros = Table[0, {n, 1, M}];
tab1 = Flatten[Append[Table[dftwerte[[k]], {k, 1, (M + 1) / 2}], zeros]];
tab2 = Table[dftwerte[[k]], {k, (M + 3) / 2, M}];
dftupsampling = Flatten[Append[tab1, tab2]] (* "zeropadded dft list" *)
```

Out[87]=
```
{0, 0.5, 0.25, 0. + 0.75 i, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0. - 0.75 i, 0.25, 0.5}
```

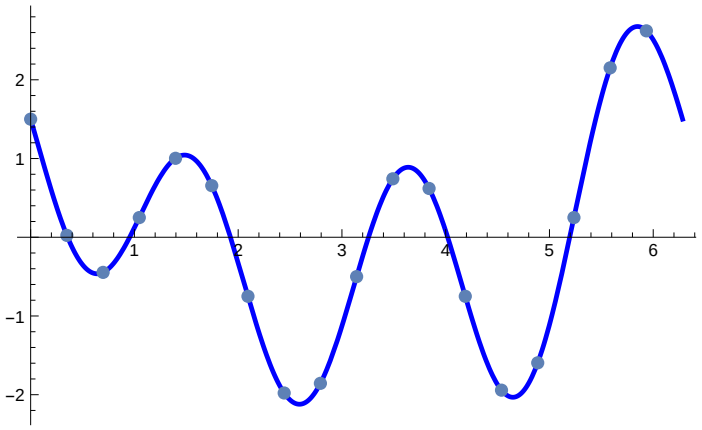
In[88]:=
```
idftneu = Chop[InverseFourier[dftupsampling, FourierParameters → {-1, -1}]]
 (* neue, nun doppelt soviele Samples von f(t) *)
```

Out[88]=
```
{1.5, 0.0236767, -0.44617, 0.25, 1.00284, 0.655544, -0.75, -1.97826, -1.85571,
 -0.5, 0.742368, 0.619818, -0.75, -1.94253, -1.59524, 0.25, 2.15191, 2.62175}
```

Zum Ansehen: Die neuen 18 Interpolationsstellen und ihre Lage auf dem Graphen von f(t):

In[89]:=
```
plot4 = ListPlot[idftneu, DataRange → {0, 2 Pi - T / (2 M)}];
Show[{plot1, plot4}]
```

Out[90]=



Hier noch einmal die Näherung, die man mittels Shannontheorem mit diesen Abtaststellen erhalten würde:

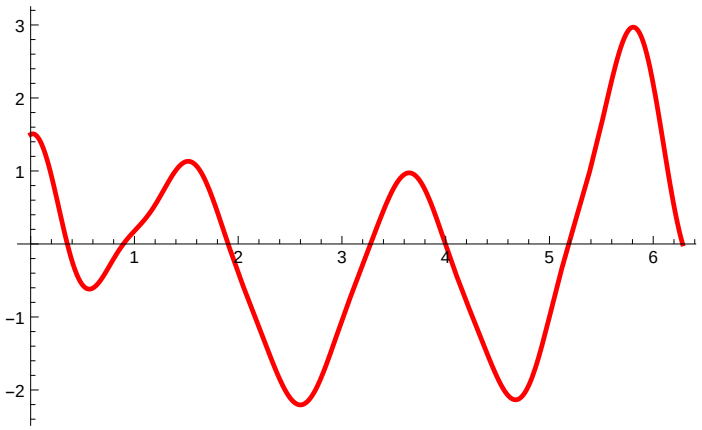
D.h. die Orthogonal-Projektion des Signals f ϵ $L^2(\mathbb{R})$ in den (hier dann endlich-dimensionalen) **Hilbertraum** der durch 2 Omega bandbegrenzten Funktionen, welcher von den Funktionen

e_k(t)=sin(2 Omega t - k π)/(2 Omega - k π) für k=0,..., Length(idftneu) -1

erzeugt wird. **Wir haben dabei nun 18 Interpolationsstellen bekommen.**

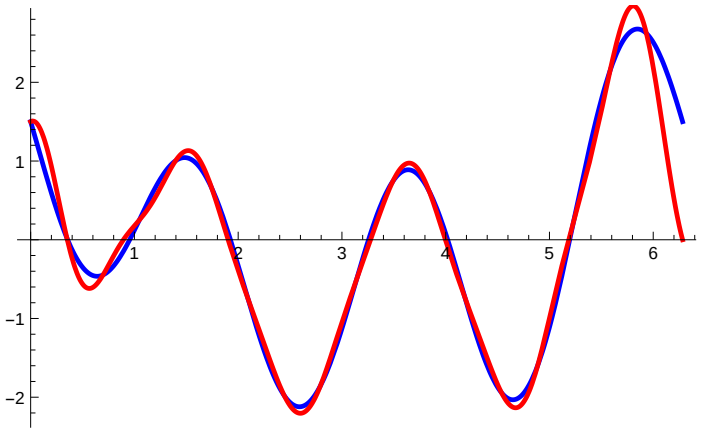
In[91]:=
```
plot5 = Plot[Sum[idftneu[[k + 1]] Sin[2 Omega t - k Pi] / (2 Omega t - k Pi),
    {k, 0, Length[idftneu] - 1}], {t, 0, T}, PlotRange -> All,
  PlotStyle -> Directive[Red, Thickness[0.007]]]
 (* Näherung mittels Shannontheorem *)
```

Out[91]=



In[92]:=
```
Show[{plot1, plot5}] (* Ausgangssignal in blau, Näherung in rot nach Shannon-
 Theorem/Tiefpassfilterung mit idealem Tiefpass mit den Abtaststellen
  Die Näherung hat nun doppelt soviele
  Interpolationsstellen im Vergleich zu oben *)
```

Out[92]=



1.4 Trigonometrische Interpolation mit Hilfe von DFT, DCT I, DCT II

Per Definition erhält man mit der DFT oder ihren reellen Varianten DCT I oder auch der vielgebrauchten DCT II - Näheres dazu im Anschluss - trigonometrische Interpolationen von Funktionen, von denen eine gewisse Anzahl äquidistanter Abtastwerte vorliegt.

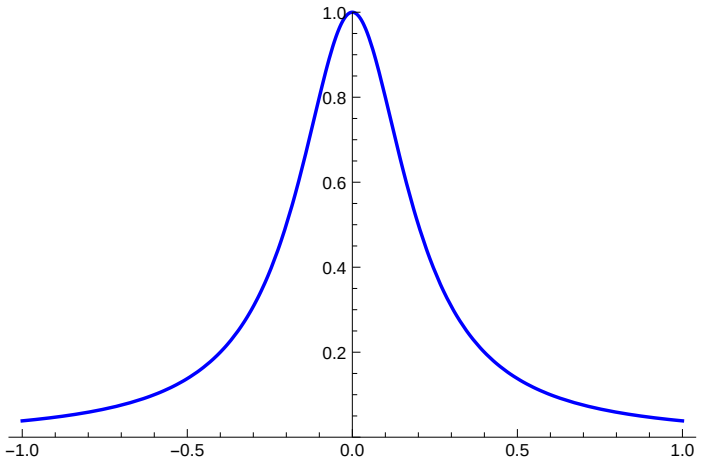
Wir betrachten folgende Situation: Von einer Funktion f:[-a,a] ->ℝ mögen NN äquidistante Abtastwerte $y_n = f(t_n)$, beginnend bei -a bis a-2a/NN, vorliegen. Gesucht wird eine trigonometrische Interpolation mit Maximalfrequenz NN/(4a) Hz. Als Beispiel wählen wir die interessante “Runge-Funktion”

runge[t]=1/(1+25 t^2) im Intervall [-1,1].

Diese Funktion ist beliebig oft differenzierbar auf ganz ℝ, ihre Potenzreihenentwicklung hat aber wegen der Polstellen bei $\pm 1/\sqrt{5}\ i$ nur den Konvergenzradius $1/\sqrt{5}$. Wir betrachten trigonometrische Interpolationen, einmal mit Hife der DFT resp. der DCT vom Typ I, dann eine solche mit verschobenen Knoten und der DCT vom Typ II.

In[93]:=
```
runge[t_] = 1 / (1 + 25 t^2);
plot20 = Plot[runge[t], {t, -1, 1},
  PlotRange → {0, 1}, PlotStyle → {Blue, Thickness[0.005]}]
```

Out[94]=



1) Trigonometrische Interpolation mit einer geraden Anzahl NN von äquidistanten Abtastwerten mit Hilfe der DFT resp. mit der DCT I

Nun eine DFT mit einer geraden Anzahl NN von Abtastpunkten. Wie schon oben müssen die Phasen im DFT-Spektrum modifiziert werden, weil wir Abtastwerte in [-1, 1] nehmen. Das resultierende Spektrum muss reell und gerade sein (vgl. [1], Abschnitt 4.1).

In[95]:=
```
T = 2;(*Intervallbreite von [-a,a]=[-1,1], T=2a *)
NN = 16;
(* gerade Anzahl von äquidistanten Abtastwerten an den Stellen -1 + 2k/NN,
k=0..NN-1*)
data20 = Table[runge[-1 + T k / NN] , {k, 0, NN - 1}];
dataphase20 = Table[(-1)^k, {k, 0, NN - 1}];
dft20 = dataphase20 * Chop[Fourier[data20, FourierParameters → {-1, -1}]]
```

Out[98]=
```
{0.274611, 0.172182, 0.0878271, 0.0486474, 0.025066,
 0.0142951, 0.00749787, 0.00525968, 0.00383778, 0.00525968,
 0.00749787, 0.0142951, 0.025066, 0.0486474, 0.0878271, 0.172182}
```

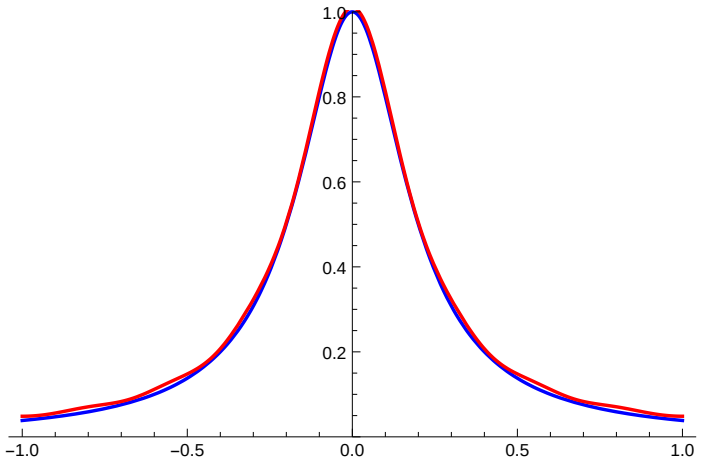
Hier nun das eindeutig bestimmte reelle trigonometrische Näherungspolynom mit der gewünschten Maximalfrequenz, das im Vektorraum $V_{NN/2}$ liegt, der von den Funktionen 1, cos(2πk/T t) für k=1,..,NN/2 und den Funktionen sin(2πk/T t) für k=1,..,NN/2-1 erzeugt wird (vgl. [1], 5.7, Satz auf S. 75). Man beachte die Zuordnung des DFT-Koeffizienten mit der Nummer NN/2 +1 (in der *Mathematica*-Nummerierung) zur Cos-Schwingung der Frequenz NN/(2T) Hz, damit wir eine reellwertige Näherungsfunktion erhalten. In [1] auf S. 75 wurde die Näherung mit $P_2(t)$ bezeichnet und dort NN/2=m gesetzt. Im Beispiel hat das trigonometrische Näherungspolynom die Maximalfrequenz 4 Hz.
Wir plotten die Runge-Funktion und diese Näherung, letztere in Rot mit einem kleinen Offset zur besseren Sichtbarkeit, sonst könnte man die beiden Grafiken am Bildschirm nur schlecht unterscheiden.

In[99]:=
```
rungeNaeh1[t_] =
 FullSimplify[Sum[dft20[[k + 1]] Exp[I * 2 Pi / T k t], {k, 0, NN / 2 - 1}] +
    Sum[dft20[[k + 1]] Exp[I * 2 Pi / T * (k - NN) t], {k, NN / 2 + 1, NN - 1}]] +
  dft20[[NN / 2 + 1]] Cos[NN / 2 * 2 Pi / T * t]
```

Out[99]=
$$0.274611 + 0.344365\cos[\pi t] + 0.175654\cos[2\pi t] + 0.0972947\cos[3\pi t] + 0.050132\cos[4\pi t] + 0.0285903\cos[5\pi t] + 0.0149957\cos[6\pi t] + 0.0105194\cos[7\pi t] + 0.00383778\cos[8\pi t]$$

In[100]:=
```
offset = 0.01;
plot21 = Plot[rungeNaeh1[t] + offset, {t, -1, 1},
   PlotRange → {0, 1.1}, PlotStyle → {Red, Thickness[0.005]}];
Show[{plot20, plot21}]
```

Out[102]=



Die erhaltene trigonometrische Näherung für das Beispiel ist die gleiche, die man mit der **DCT vom Typ I** erhält mit NN/2+1=9 Stützstellen im Intervall [0,1] von der Form t_k= k/m, für k=0,..,m, m=NN/2. Man vgl. dazu [1], Abschnitt 5.7, S. 75.

2) Trigonometrische Interpolation mit einer geraden Anzahl NNN von äquidistanten Abtastwerten mit Hilfe der DCT vom Typ II

Eine trigonometrische Interpolation mit Hilfe der **DCT vom Typ II**, d.h. hier mit den Knoten t_k=(2n+1)/NNN für 0⩽n⩽NNN/2-1, ist eine weitere gebräuchliche Option.
Wir verwenden dazu die **in *Mathematica* verfügbare DCT II** mit dem Befehl **FourierDCT[list,2]**. Wir wählen **NNN=NN+2**, **m=NNN/2** um möglichst nahe an den Notationen von [1] zu bleiben und um die gleiche Maximalfrequenz des trigonometrischen Interpolationspolynoms wie vorher mit der DFT/DCT I mit den nachfolgenden Befehlen zu erhalten.

Anmerkung zu Unterschieden in den Definitionen von DFT und DCT bei *Mathematica* und in meinem Lehrbuch [1]:

Die Unterschiede in den Definitionen von DFT und den vorgestellten DCT-Varianten sind folgende: Ich habe in [1] die Normierungsfaktoren so gewählt, dass die DFT- und DCT II-Koeffizienten mit dem Index Null jeweils den "Gleichanteil" - bei einer betrachteten Wechselspannung also etwa den Gleichspannungsanteil (engl. dc value) - angeben und die anderen Koeffizienten die richtigen Amplitudenwerte beteiligter Schwingungen im beobachteten Signal wiedergeben. In der in [1] verwendeten Normierung kann man die DCT I- und DCT II-Koeffizienten auch direkt zur Interpolation mit Tschebyscheff-Polynomen verwenden (vgl. [1], Abschnitt 5.7, S. 84).
Mit der DFT in Mathematica erzielt man eine Übereinstimmung mit [1] durch die Option FourierParameters→ {-1,-1}, bei der Fouriertransformation mit FourierParameters→ {1,-1} wie oben schon benutzt. .
Bei der DCT in Mathematica steht - soweit ich es gesehen habe - diese Option nicht zur Verfügung. Die DCT II von Mathematica liefert im Vergleich bis auf einen Normierungsfaktor die Koeffizienten, die in [1] auf S. 77 mit α_k bezeichnet sind. Dabei ist wie vorher schon der Mathematica-Index k um Eins im Vergleich zu [1] erhöht. Beachten Sie also bitte diese kleinen Unterschiede, die schnell auszugleichen sind, wenn Sie mein Lehrbuch [1] mit Mathematica verwenden.

Zunächst also die nötigen Abtastwerte an den im Vergleich zu vorher verschobenen Abtaststellen, dann die DCT II, dann das resultierende trigonometrische Interpolationspolynom (das ist $P_3(t)$ in der Notation von [1], S. 77) und wieder ein Plot mit der Näherung in Rot. Man beachte also wie gerade gesagt, den richtigen Normierungsfaktor und die richtige Frequenzzuordnung der DCT-Koeffizienten zu wählen (vgl. [1], S. 77).

Zum Vergleich in einem zweiten Plot der Verlauf der absoluten **Approximationsfehler** der beiden Interpolationen, der mit der DFT/DCT I in Rot, der mit der DCT II in Blau.
In der Umgebung des Nullpunkts liefert die DFT/DCT I-Näherung eine bessere Approximation, etwas weiter weg davon ist die Näherung mit der DCT II besser.
Sie könnten sich selbst auch noch die prozentualen absoluten Fehler einmal ansehen und vergleichen.

```
NNN = NN + 2;
m = NNN / 2; (* nun sind m Abtastwerte in [0,1] zu nehmen *)
list21 = Table[runge[(2 n + 1) / (2 m)], {n, 0, m - 1}]
dct21 = 1 / Sqrt[m] FourierDCT[list21, 2] (* Normierung wie in [1], s.o. *)
rungeNaeh2[t_] = dct21[[1]] + Sum[2 dct21[[k]] Cos[(k - 1) π t], {k, 2, m}]
(* Das trig. Näherungspolynom, das in [1],
auf Seite 77 mit P3 bezeichnet wird. *)
```

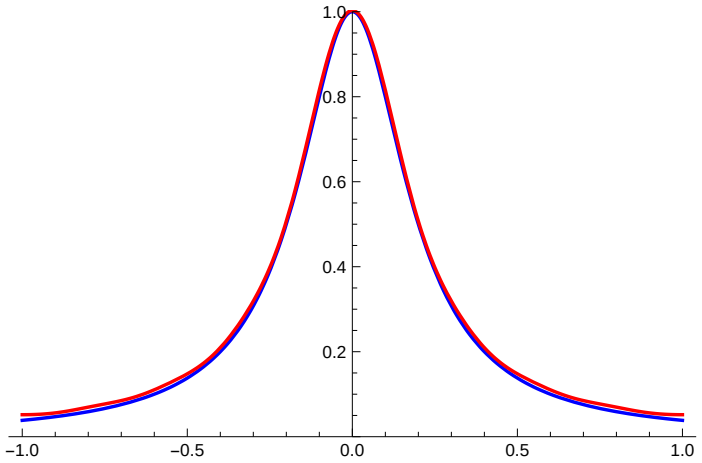
Out[104]= $\left\{\frac{324}{349}, \frac{36}{61}, \frac{324}{949}, \frac{324}{1549}, \frac{4}{29}, \frac{324}{3349}, \frac{324}{4549}, \frac{36}{661}, \frac{324}{7549}\right\}$

Out[105]= {0.27471, 0.172006, 0.0878997, 0.0483856,
0.025004, 0.0137254, 0.00692886, 0.00363661, 0.00143487}

Out[106]= 0.27471 + 0.344011 Cos[π t] + 0.175799 Cos[2 π t] +
0.0967712 Cos[3 π t] + 0.050008 Cos[4 π t] + 0.0274507 Cos[5 π t] +
0.0138577 Cos[6 π t] + 0.00727322 Cos[7 π t] + 0.00286974 Cos[8 π t]

In[107]:=
```
offset = 0.01;
plot22 = Plot[rungeNaeh2[t] + offset, {t, -1, 1},
   PlotRange → {0, 1.1}, PlotStyle → {Red, Thickness[0.005]}];
Show[{plot20, plot22}]
```

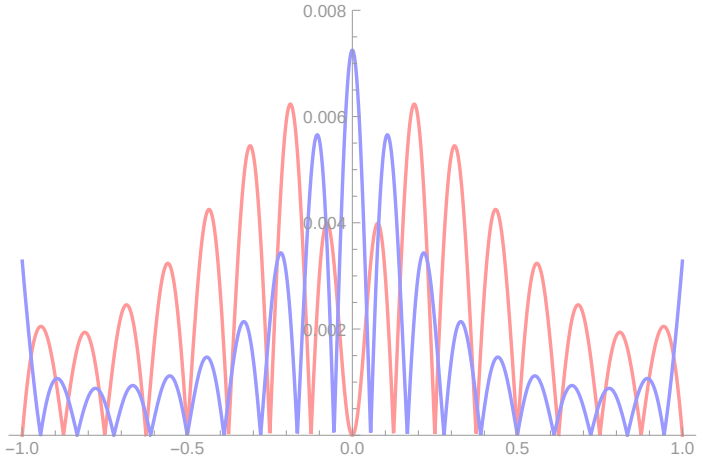
Out[109]=



Nun die absoluten Approximationsfehler der beiden betrachteten trigonometrischen Näherungspolynome. Die Nullstellen der jeweiligen Fehlerkurven sind die Stellen, an denen die Runge-Funktion interpoliert wird. In beiden Fällen haben wir 9 solcher Knoten im Intervall von [0,1].

```
fehlerdct1[t_] = Abs[runge[t] - rungeNaeh1[t]];
fehlerdct2[t_] = Abs[runge[t] - rungeNaeh2[t]];
plot23 = Plot[fehlerdct1[t], {t, -1, 1},
   PlotRange → {0, 0.008}, PlotStyle → {Red, Thickness[0.005]}];
plot24 = Plot[fehlerdct2[t], {t, -1, 1}, PlotRange → {0, 0.008},
   PlotStyle → {Blue, Thickness[0.005]}];
Show[{plot23, plot24}]
```

Out[113]=



Bemerkung: Das Beispiel mit der Runge-Funktion finden Sie auch auf den Help-Seiten von *Mathematica* als Beispiel zum Befehl FourierDCT. Dort mit etwas anderen Befehlen, die *Mathematica* zur Verfügung stellt, durchgeführt.
Man vgl. außerdem Aufgabe A20 in [1], Abschnitt 5.8, S. 101. Dort werden **polynomiale Interpolationen der Runge-Funktion** betrachtet, und zwar einmal mit äquidistanten Knoten und dann mit den sogenannten Tschebyscheff-Knoten. Ich gehe darauf in einem anderen Notebook dieser Serie näher ein (siehe URL zu Beginn).

Schluss: Damit das Notebook nicht zu umfangreich wird, sei es hiermit beendet.
Es gibt unzählige Möglichkeiten, interessante Aspekte der DFT/DCT zu betrachten und Anwendungsbeispiele der DFT/DCT zu zeigen. Ich werde eine kleine Auswahl aus weiteren Themen in diesem Zusammenhang in nachfolgenden Notebooks vorstellen.

Weitere empfehlenswerte Literatur :

[2] W.L. Briggs, Van Emden Henson
The DFT, An Owners Manual for the Discrete Fourier Transform, SIAM, Philadelphia, 1995

[3] M. Hanke-Bourgeois
Grundlagen der Numerischen Mathematik und des Wissenschaftlichen Rechnens, Teubner, Wiesbaden, 2008