FreeDV Treffen Anfang April 2022

1. **Terminplanung**

2. Fourier-Transformation
   - OFDM Signal-Erzeugung
   - bekannte Anwendungen der Fourier-Transformation
   - analytische, **diskrete** und *schnelle* Fourier-Transformation

3. offene Diskussion

1. Terminplanung

- FreeDV Runde jeden Sonntag
  ab 11:00 Uhr, 40m & 80m
  ab 13:30 Uhr, 40m & 80m
- nächstes Gruppentreffen am 28**. April**, ab 19:30 Uhr

April 2022							
KW	Mo	Di	Mi	Do	Fr	Sa	So
13					1	2	3
14	4	5	6	7	8	9	10
15	11	12	13	14	15	16	17
16	18	19	20	21	22	23	24
17	25	26	27	28	29	30	

nächstes Treffen in **3 Wochen**

FreeDV Treffen Anfang April 2022

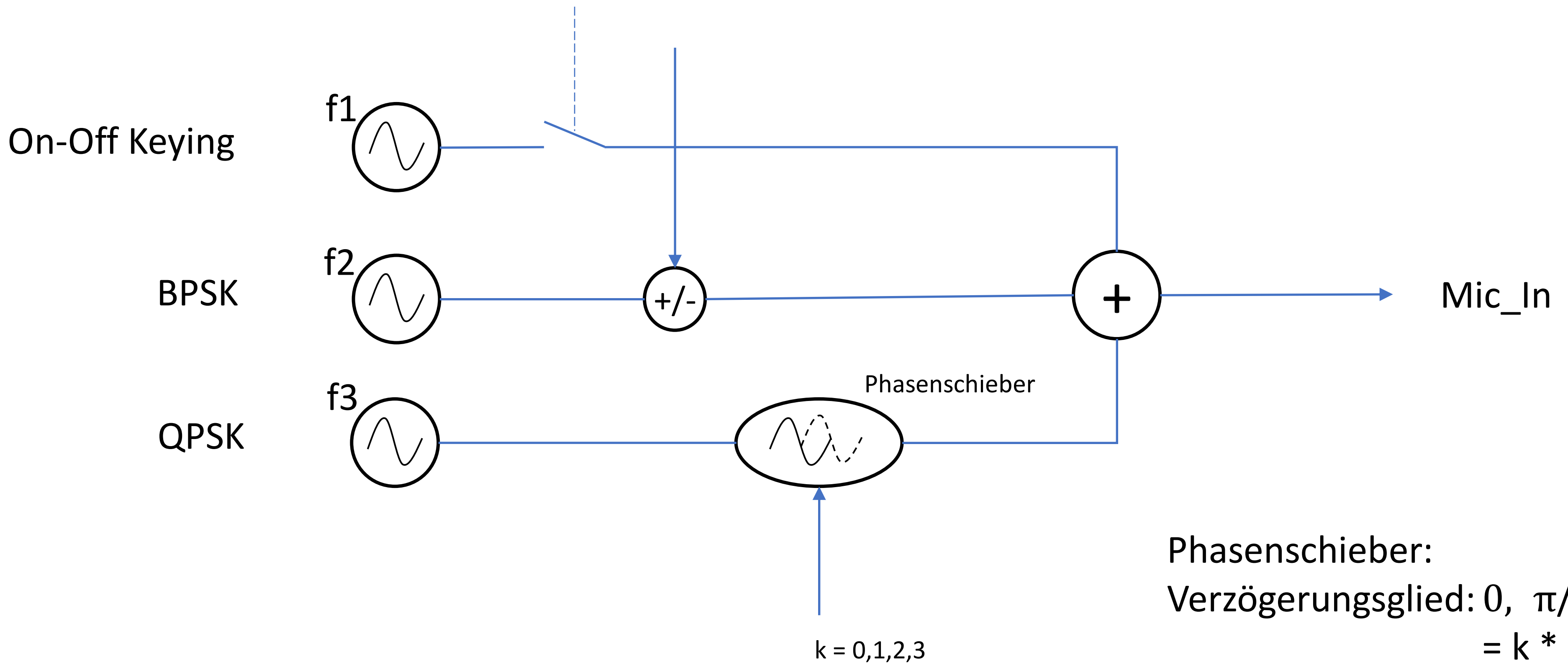
1. Terminplanung

2. **Fourier-Transformation**
    - OFDM Signal-Erzeugung
    - bekannte Anwendungen der Fourier-Transformation
    - analytische, **diskrete** und *schnelle* Fourier-Transformation

3. offene Diskussion

2. Fourier-Transformation

OFDM Signal erzeugen:

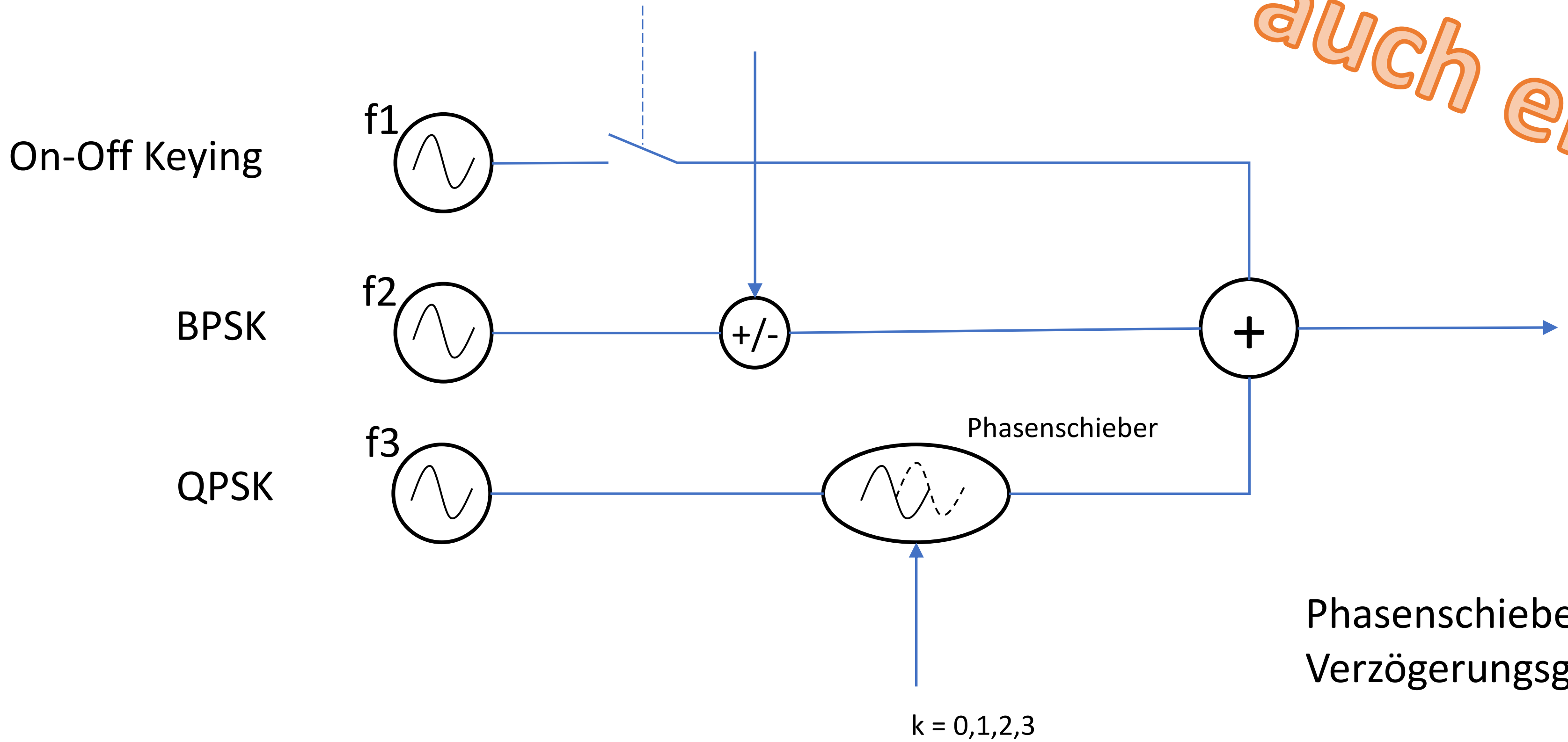


Phasenschieber:
Verzögerungsglied: $0, \ \pi/2, \ \pi, \ 3\pi/2$
$= k * \pi/2$

2. Fourier-Transformation

OFDM Signal erzeugen:

geht das nicht auch einfacher?



Phasenschieber:
Verzögerungsglied: $0,\ \pi/2,\ \pi,\ 3\pi/2$
$= k * \pi/2$

2. Fourier-Transformation

einfacher mit…

bekannte Anwendungen:

- Spektrum eines analogen Signals anzeigen
- Wasserfall Diagramm anzeigen
  (→ zeitliche Entwicklung eines Spektrums)

2. Fourier-Transformation

einfacher mit...

<u>**bekannte Anwendungen:**</u>

- Spektrum eines analogen Signals anzeigen
- Wasserfall Diagramm anzeigen
  (→ zeitliche Entwicklung eines Spektrums)

Hmmm?
Ist das ein ***echtes*** Spektrum?

Was macht ein „**Spectrum Analyzer**“ anders?

2. Fourier-Transformation

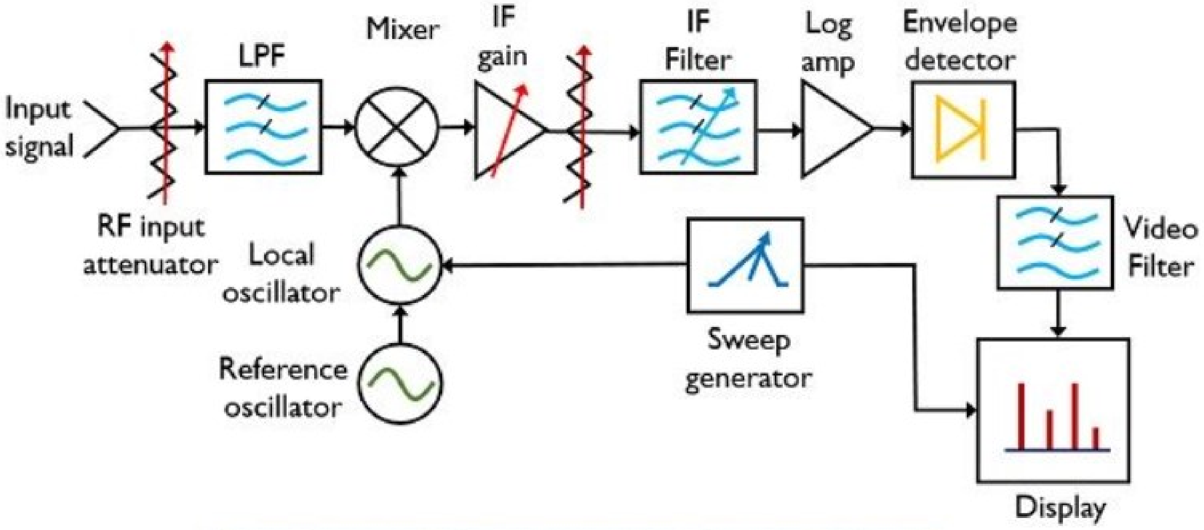
einfacher mit...

bekannte Anwendungen:

- Spektrum eines analogen Signals anzeigen
- Wasserfall Diagramm anzeigen
  (→ zeitliche Entwicklung eines Spektrums)

Hmmm?
Ist das ein ***echtes*** Spektrum?

Was macht ein „**Spectrum Analyzer**“ anders?



Block Diagram of Spectrum Analyzer

Quelle: https://electronicsdesk.com/spectrum-analyzer.html

Electronics Desk

2. Fourier-Transformation

<u>bekannte Anwendungen:</u>

- Spektrum eines analogen Signals anzeigen
- Wasserfall Diagramm anzeigen
  (→ zeitliche Entwicklung eines Spektrums)

Tatsächlich handelt es sich dabei um abgetastete Analogwerte,
die in eine spektrale Anzeige „übersetzt“ werden (Transformation).

Mathematisch unterscheidet man zwischen einer

- Fourier-Transformierten (mathematische Funktion)
- diskreten Fourier-Transformation (das verwenden wir typischerweise!)

Es gibt die

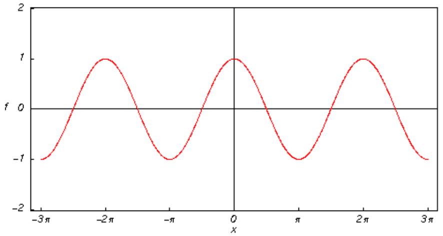
- Fourier-Transformation
- **Inverse** Fourier Transformation

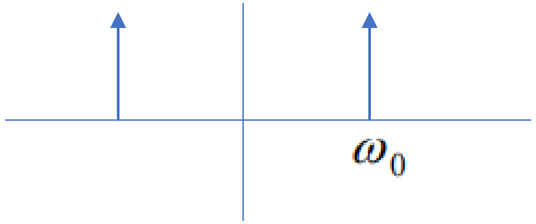
2. Fourier-Transformation

Mathematisch unterscheidet man zwischen einer

- Fourier-Transformierten
  (mathematische Funktion, Formeldarstellung)

Function, f(t)	**Fourier Transform, F(ω)**
Definition of Inverse Fourier Transform $$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{j\omega t}\,d\omega$$	***Definition of Fourier Transform*** $$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}\,dt$$
$\cos(\omega_0 t)$	$\pi[\delta(\omega-\omega_0)+\delta(\omega+\omega_0)]$



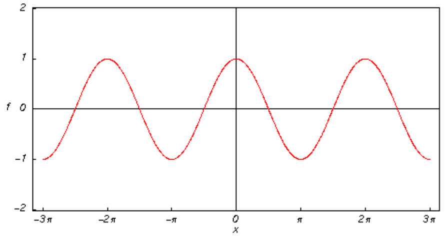


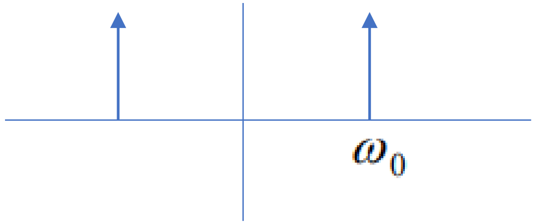
2. Fourier-Transformation

Mathematisch unterscheidet man zwischen einer

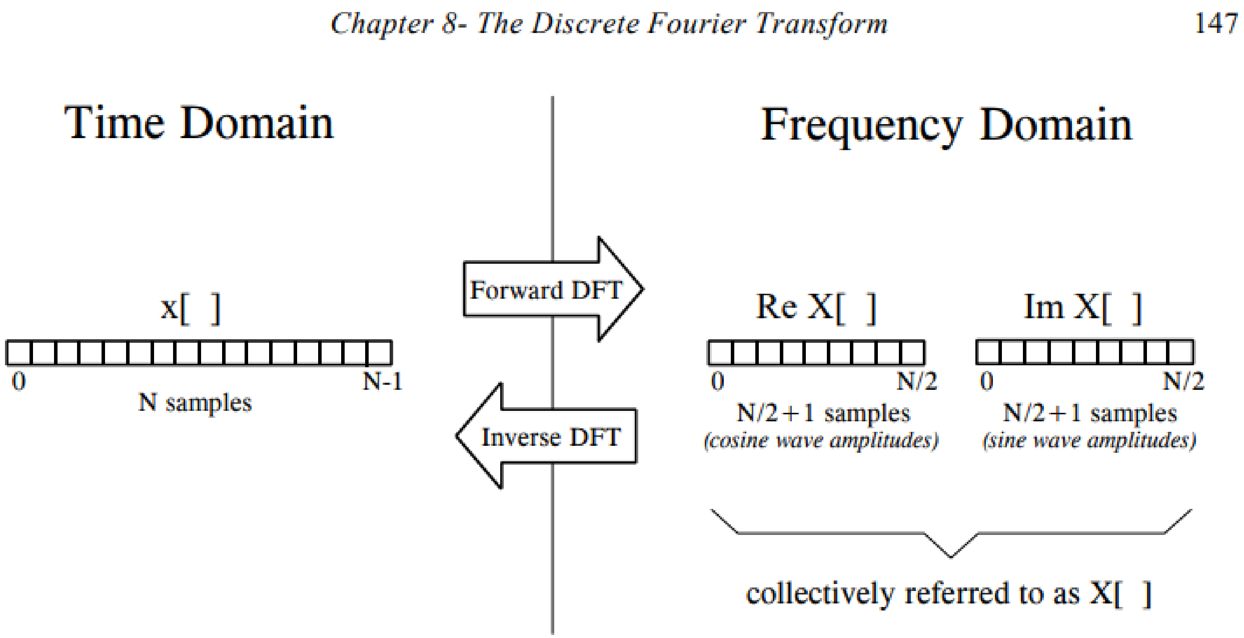
- Fourier-Transformierten
  (mathematische Funktion, Formeldarstellung)

Function, f(t)	**Fourier Transform, F(ω)**
Definition of Inverse Fourier Transform $f(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty}F(\omega)e^{j\omega t}\,d\omega$	***Definition of Fourier Transform*** $F(\omega)=\int_{-\infty}^{\infty}f(t)e^{-j\omega t}\,dt$
$\cos(\omega_0 t)$	$\pi\left[\delta(\omega-\omega_0)+\delta(\omega+\omega_0)\right]$





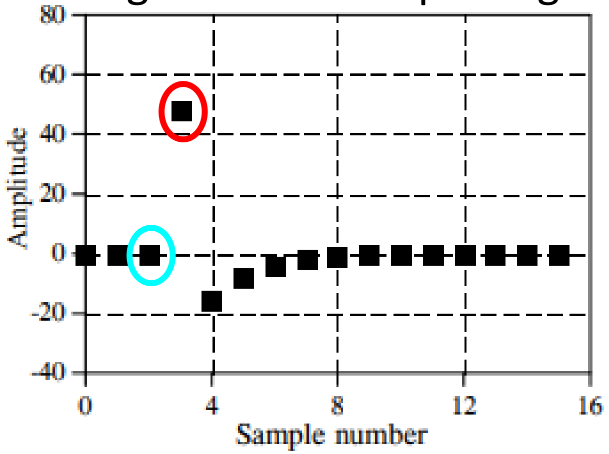
- **diskrete Fourier-Transformation**
  (das verwenden wir typischerweise, d.h. wir arbeiten mit Mengen von **Abtastwerten**)



Quellen:
https://ethz.ch/content/dam/ethz/special-interest/baug/ibk/structural-mechanics-dam/education/identmeth/fourier.pdf, https://www.dspguide.com/

2. Fourier-Transformation

abgetastetes Beispiel-Signal



Abtastwerte 0..15 (N=16)

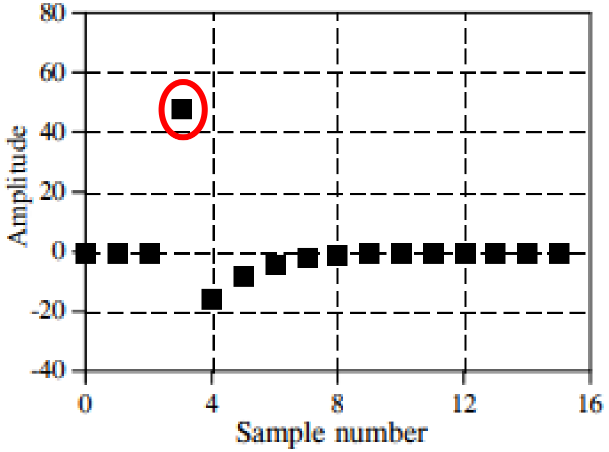
Das gezeigte „Signal“ ist nicht werte-kontinuierlich, es ist bereits abgetastet.

(zwischen Wert 3 und 4 gibt es keine Information über den Kurven-Verlauf!)

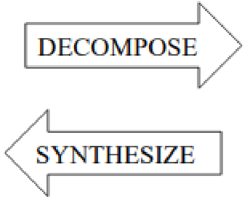
Die Abtastrate ist so gewählt, dass genau N=16 Abtastwerte
vom Kurven-Verlauf erzeugt werden.

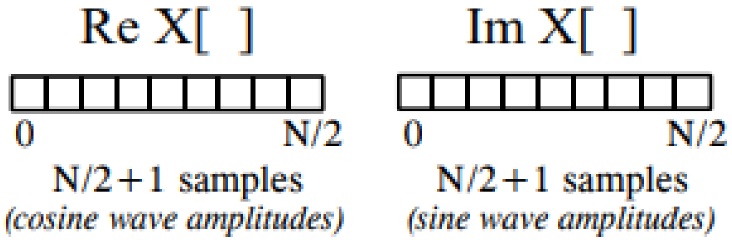
Quelle: https://www.dspguide.com/

2. Fourier-Transformation

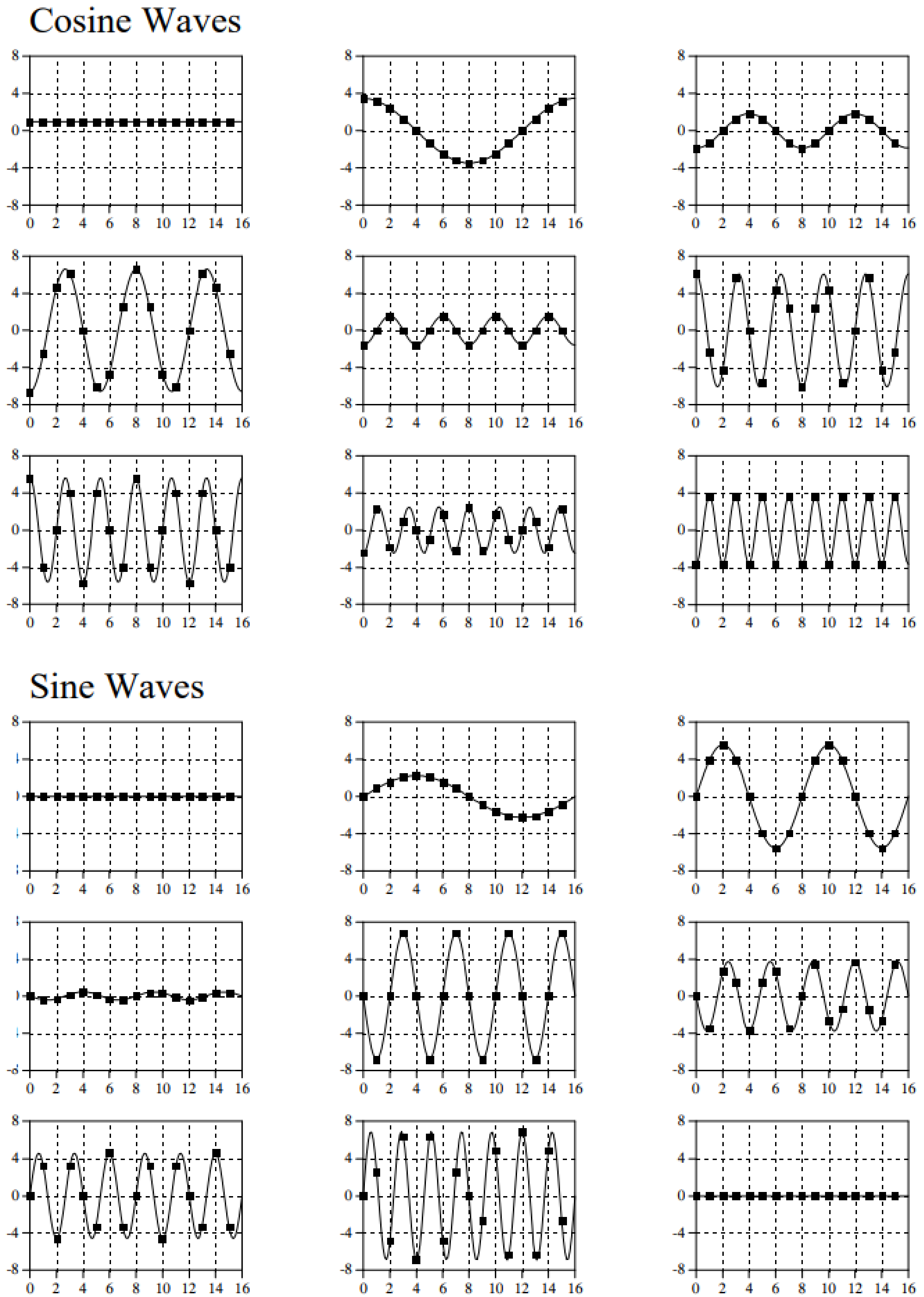


Abtastwerte 0..15 (N=16)



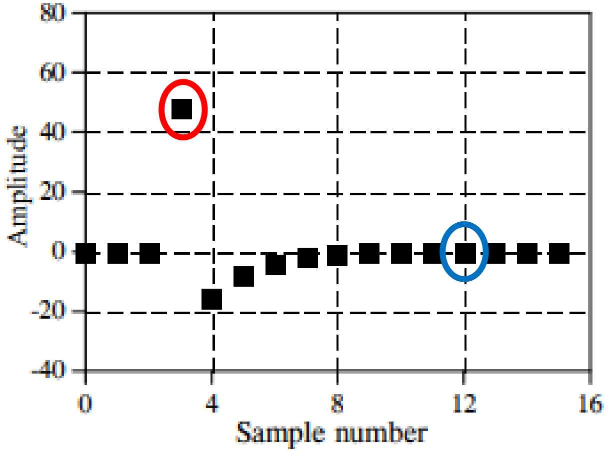


N/2 + 1 = **9** **Amplitudenwerte** von **Cosinus-** und **Sinus-** Schwingungen

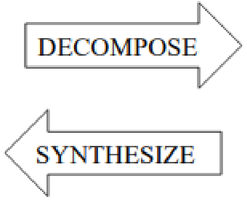


Quelle: https://www.dspguide.com/

2. Fourier-Transformation

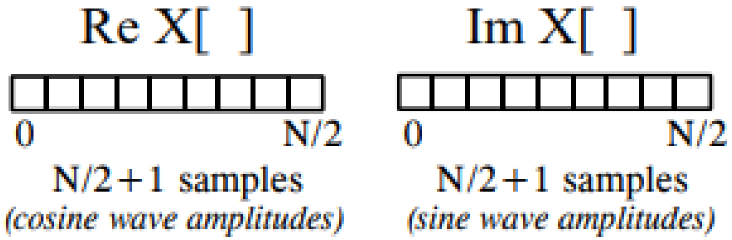


Abtastwerte 0..15 (N=16)

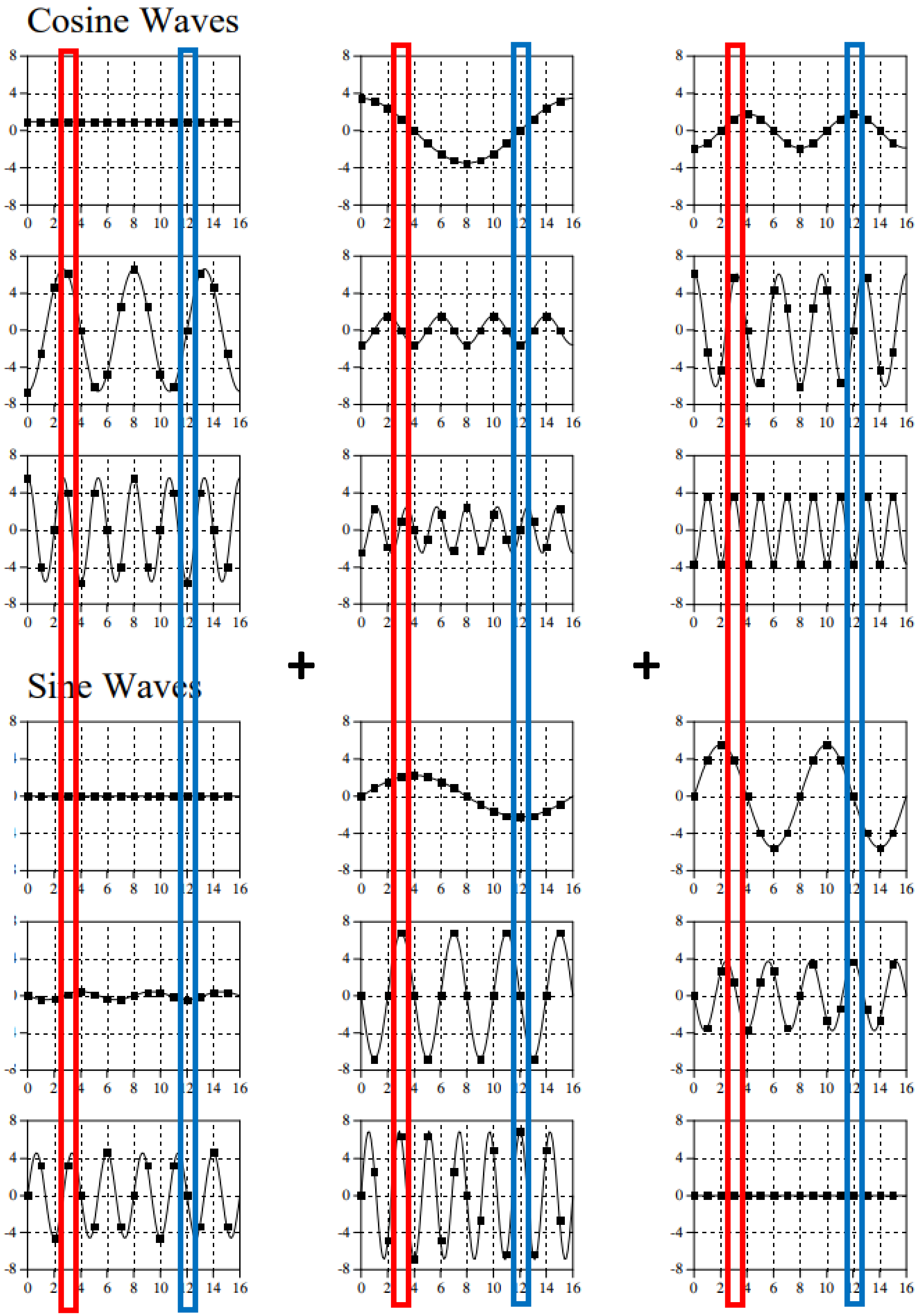


Transformation

inverse Transformation

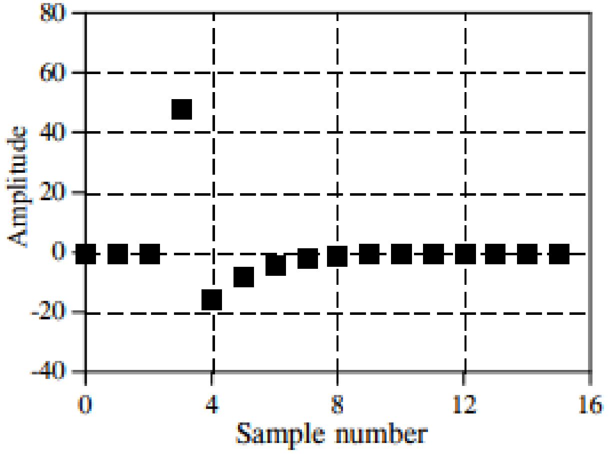


N/2 + 1 = **9** **Amplitudenwerte** von **Cosinus-** und **Sinus-** Schwingungen

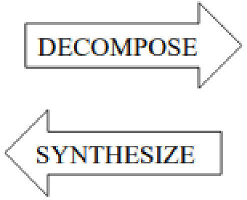


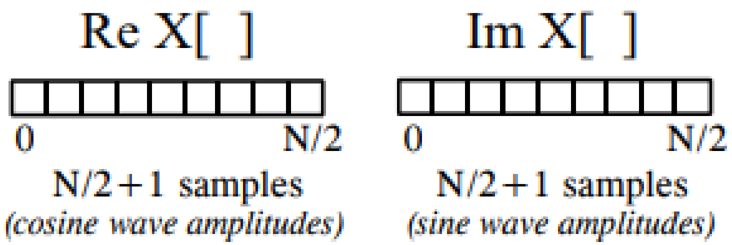
Quelle: https://www.dspguide.com/

2. Fourier-Transformation

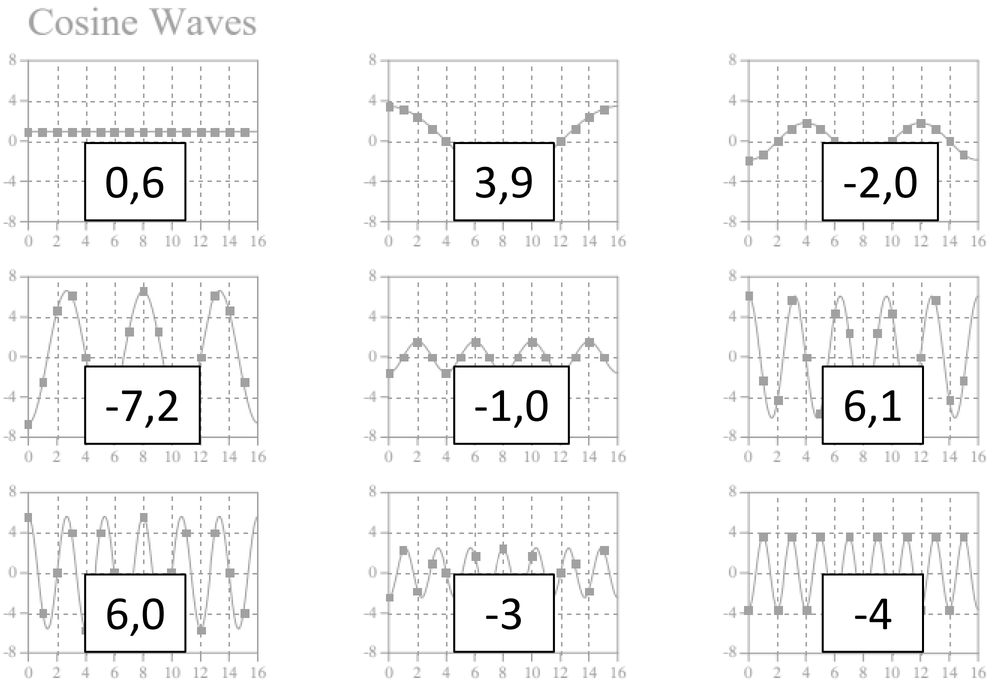


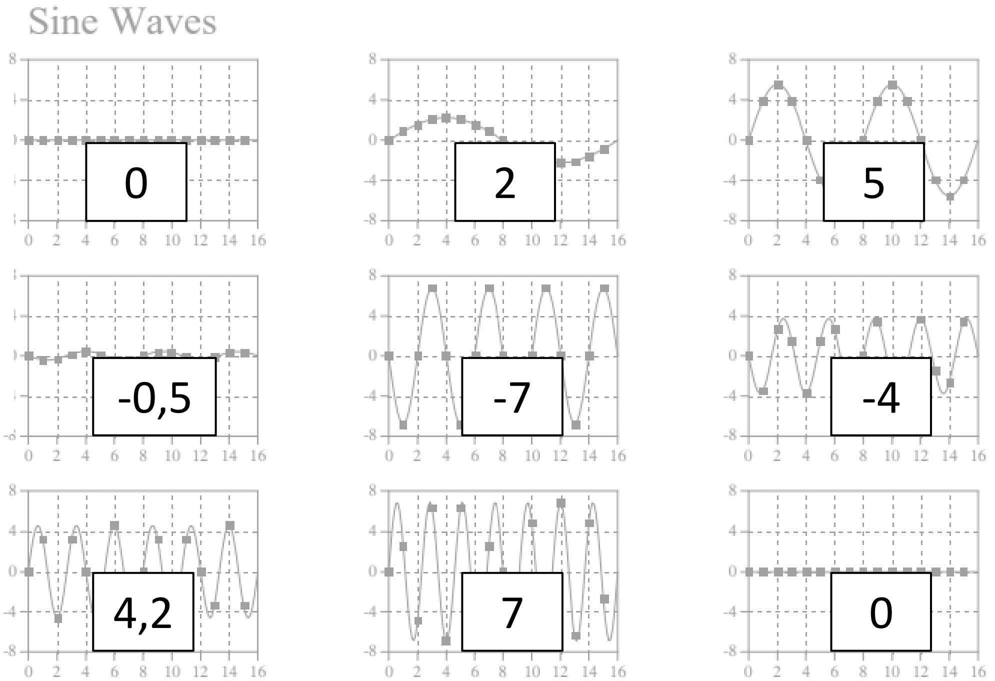
Abtastwerte 0..15 (N=16)





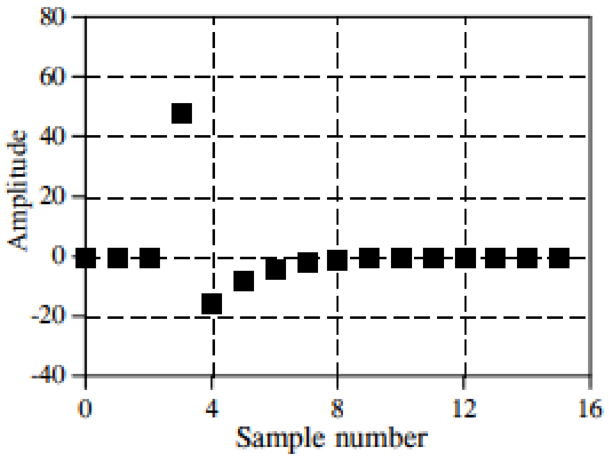
N/2 + 1 = **9** **Amplitudenwerte** von **Cosinus-** und **Sinus-**Schwingungen





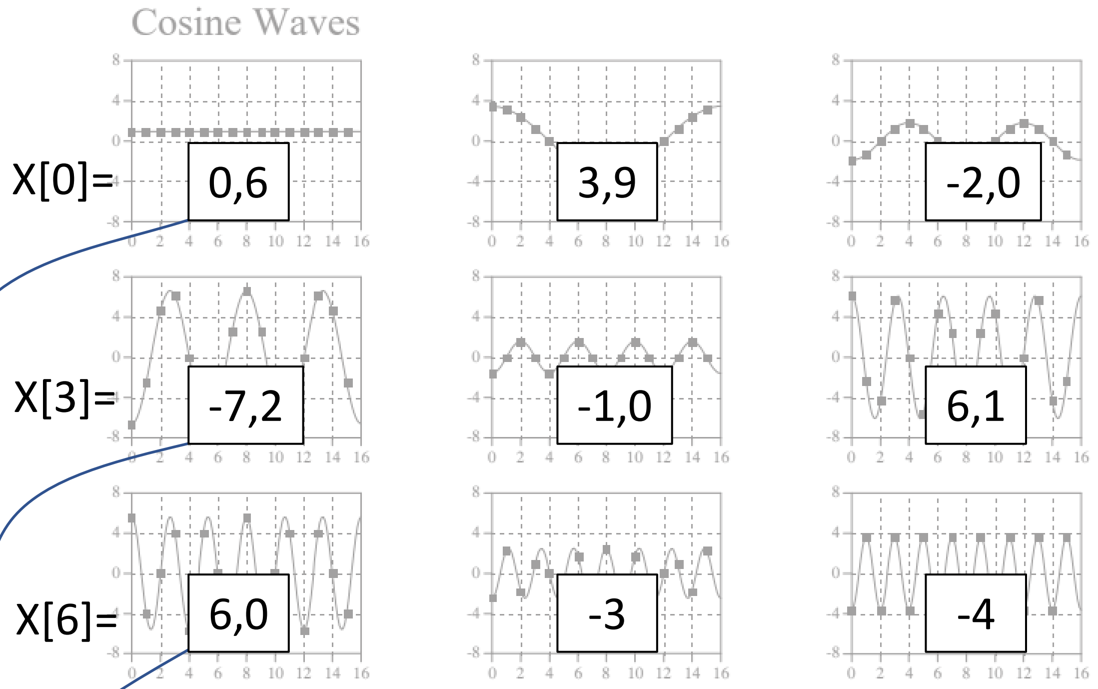
Quelle: https://www.dspguide.com/

2. Fourier-Transformation



DECOMPOSE

SYNTHESIZE



Abtastwerte 0..15 (N=16)

***x* [*i*]**

cos() Amplituden

sin() Amplituden

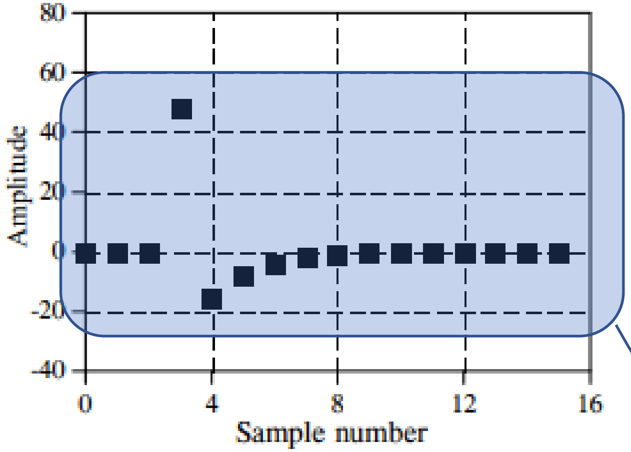
$$x[i] = \sum_{k=0}^{N/2} Re\bar{X}[k] \cos(2\pi ki/N) + \sum_{k=0}^{N/2} Im\bar{X}[k] \sin(2\pi ki/N)$$

„Rückwärtsrechnung“
(**inverse** diskrete
Fourier-Transformation)

EQUATION 8-2
The synthesis equation. In this relation, $x[i]$ is the signal being synthesized, with the index, i, running from 0 to $N-1$. $Re\bar{X}[k]$ and $Im\bar{X}[k]$ hold the amplitudes of the cosine and sine waves, respectively, with k running from 0 to $N/2$. Equation 8-3 provides the normalization to change this equation into the inverse DFT.

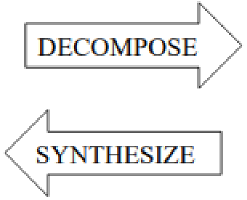
Quelle: https://www.dspguide.com/

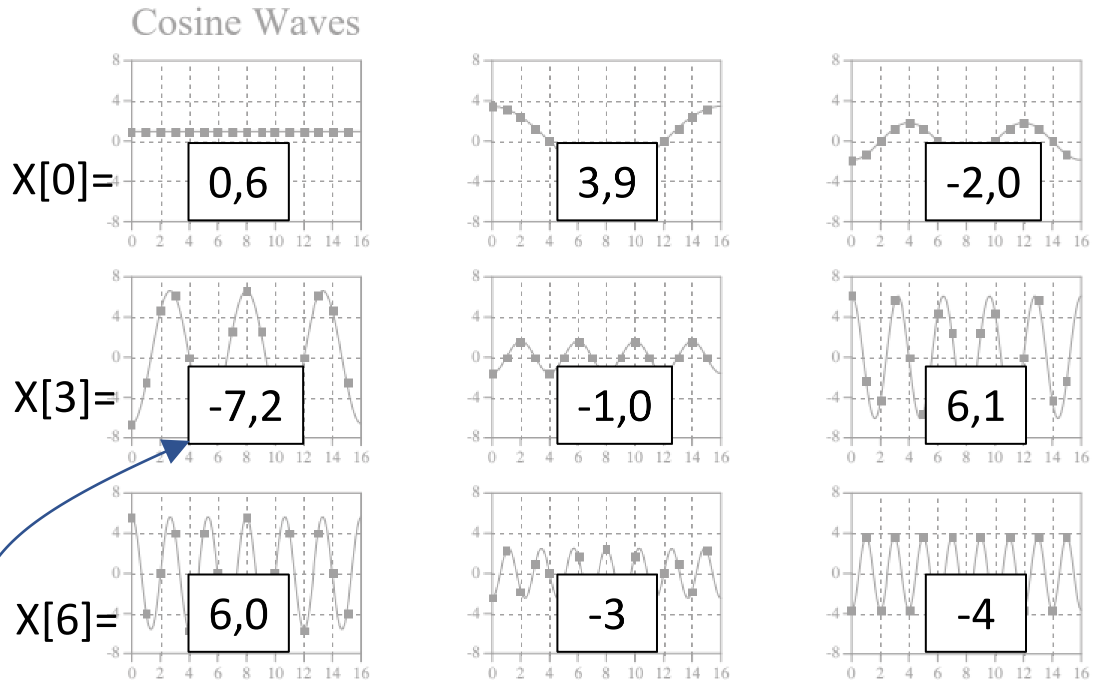
2. Fourier-Transformation



Abtastwerte 0..15 (N=16)

***x* [*i*]**





cos() Amplituden

$$ReX[k] = \sum_{i=0}^{N-1} x[i]\cos(2\pi k i / N)$$

„**Vorwärtsrechnung**“
(diskrete Fourier-Transformation)

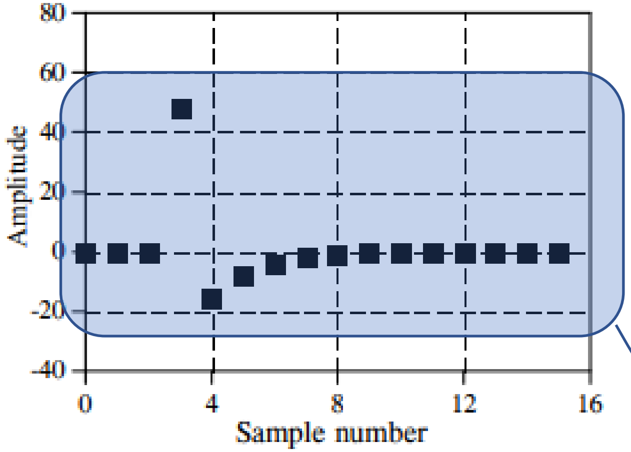
EQUATION 8-4
The analysis equations for calculating the DFT. In these equations, $x[i]$ is the time domain signal being analyzed, and $Re\,X[k]$ & $Im\,X[k]$ are the frequency domain signals being calculated. The index i runs from 0 to $N-1$, while the index k runs from 0 to $N/2$.

sin() Amplituden

$$ImX[k] = -\sum_{i=0}^{N-1} x[i]\sin(2\pi k i / N)$$

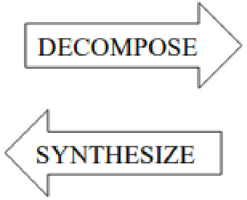
Quelle: https://www.dspguide.com/

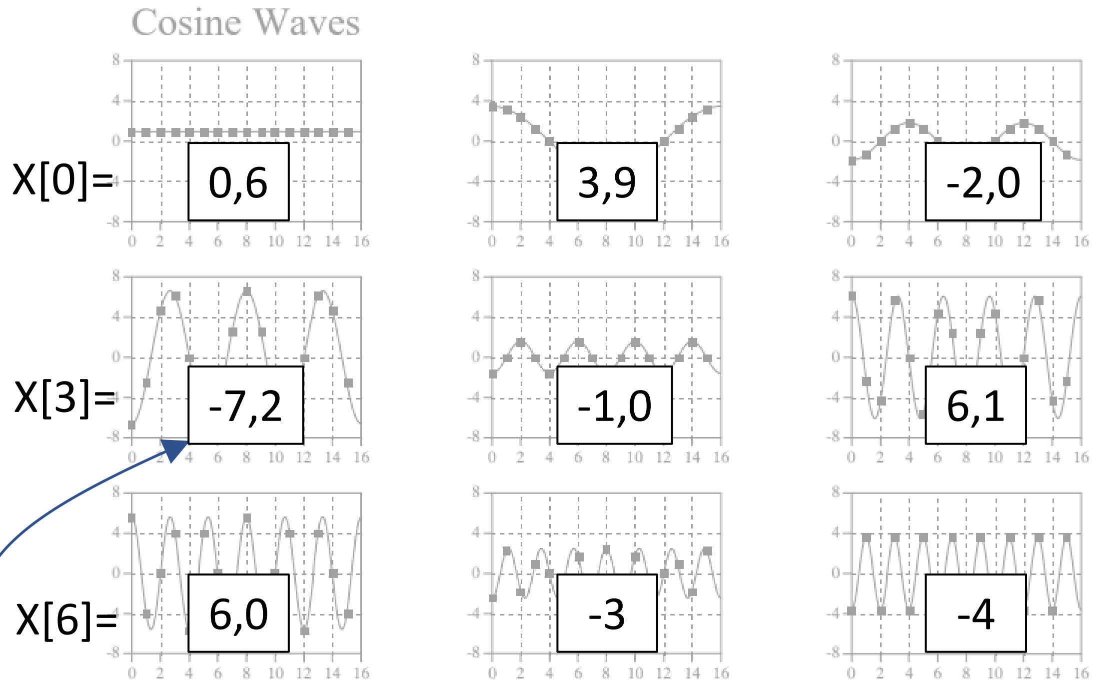
2. Fourier-Transformation



Abtastwerte 0..15 (N=16)

***x* [*i*]**





cos() Amplituden

$$Re X[k] = \sum_{i=0}^{N-1} x[i] \cos(2\pi k i / N)$$

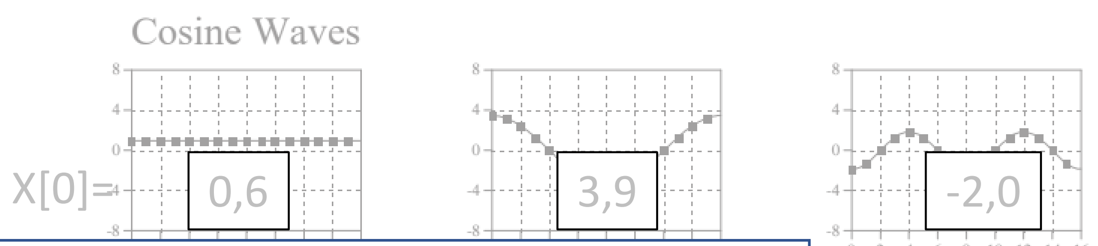
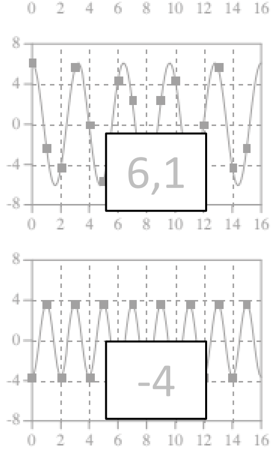
Re X[3] = x(0) cos (2π*3*0/16)
+ x(1) cos (2π*3*1/16)
+ x(2) cos (2π*3*2/16)
+ x(3) cos (2π*3*3/16)
+ x(4) cos (2π*3*4/16)
+ …

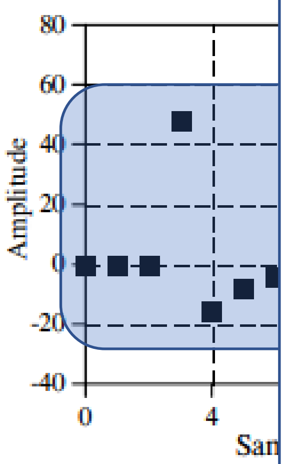
„Vorwärtsrechnung“
(diskrete Fourier-Transformation)

EQUATION 8-4
The analysis equations for calculating the DFT. In these equations, $x[i]$ is the time domain signal being analyzed, and $Re\,X[k]$ & $Im\,X[k]$ are the frequency domain signals being calculated. The index i runs from 0 to $N-1$, while the index k runs from 0 to $N/2$.

Quelle: https://www.dspguide.com/

2. Fourier-Transformation

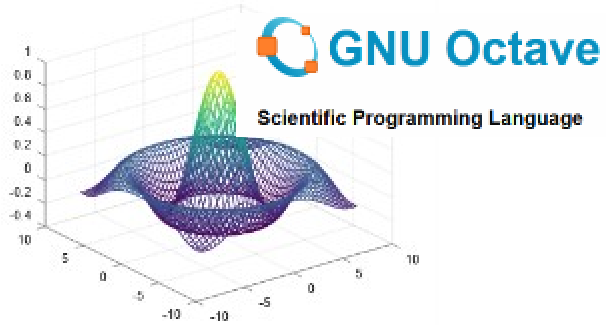





Abtastwerte
x [*i*]

viele Zahlen, viel berechnen, d.h.

→ wir brauchen einen programmierbaren, wissenschaftlichen Taschenrechner, z.B. Matlab (kostet Geld), oder die freie Variante **GNU Octave**



Besonderheit in beiden Programmen:

- komplex programmierbar, viele fertige Bibliotheken
- viele Rechnungen können mit **Einzelwerten** erfolgen oder mit **großen Datenmengen**, in einem Schritt:

Einfach-Berechnung

```
>> sin(1.1)
ans = 0.8912
```

Daten-Vektor

```
>> a=[1.1 1.5 1.8 2.1 2.2]
a =

   1.1000    1.5000    1.8000    2.1000    2.2000
```

Rechnung 1

Rechnung 2

```
>> 3+a
ans =

   4.1000    4.5000    4.8000    5.1000    5.2000
```

$\pi k i / N)$

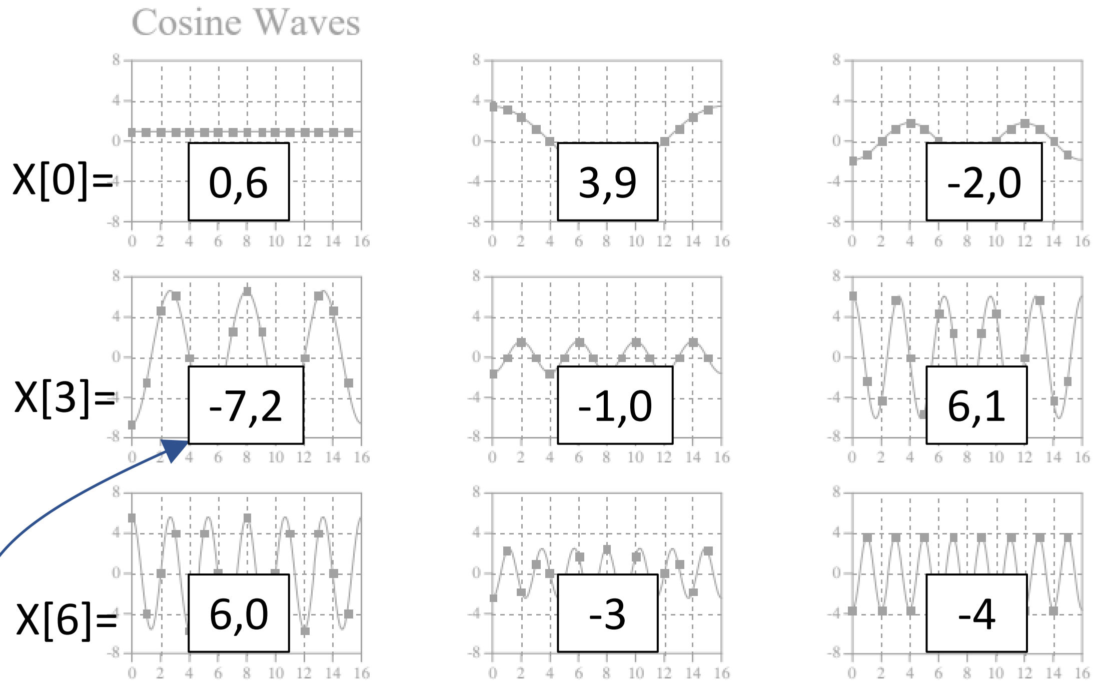
3*0/16)
3*1/16)
3*2/16)
3*3/16)
3*4/16)

Quelle: https:/

Datei Bearbeiten Ansicht Debuggen Ausführen Hilfe

v2.m | v1_beispiel.m

```
disp ("einfache diskrete Transformation");

# 16 Beispielwerte
werte = [ 0, 0, 0, 50, -18, -9, -4.5, -2, -1, ...
          -0.5, -0.2, -0.05, 0, 0, 0, 0];

function ret = fourier_cos(samples)
  N    = length(samples);
  for k = ( 1 : N/2 +1)
    a(k) = 0;
    for n = ( 1 : N )
      a (k) = a(k) + samples(n) * cos( 2*pi * (k-1) * (n-1) /N);
    endfor
  endfor
  ret = 2/N * a;
end

function ret = fourier_sin(samples)
  N    = length(samples);
  for k = ( 1 : N/2 +1)
    a(k) = 0;
    for n = ( 1 : N )
      a (k) = a(k) + samples(n) * sin( 2*pi * (k-1) * (n-1) /N);
    endfor
  endfor
  ret = 2/N * a;
end

fourier_cos(werte)
fourier_sin(werte)
sqrt ( fourier_cos(werte) .^2 + fourier_sin(werte) .^2 )
```



cos() Amplituden

$$ReX[k] = \sum_{i=0}^{N-1} x[i] \cos(2\pi k\, i / N)$$

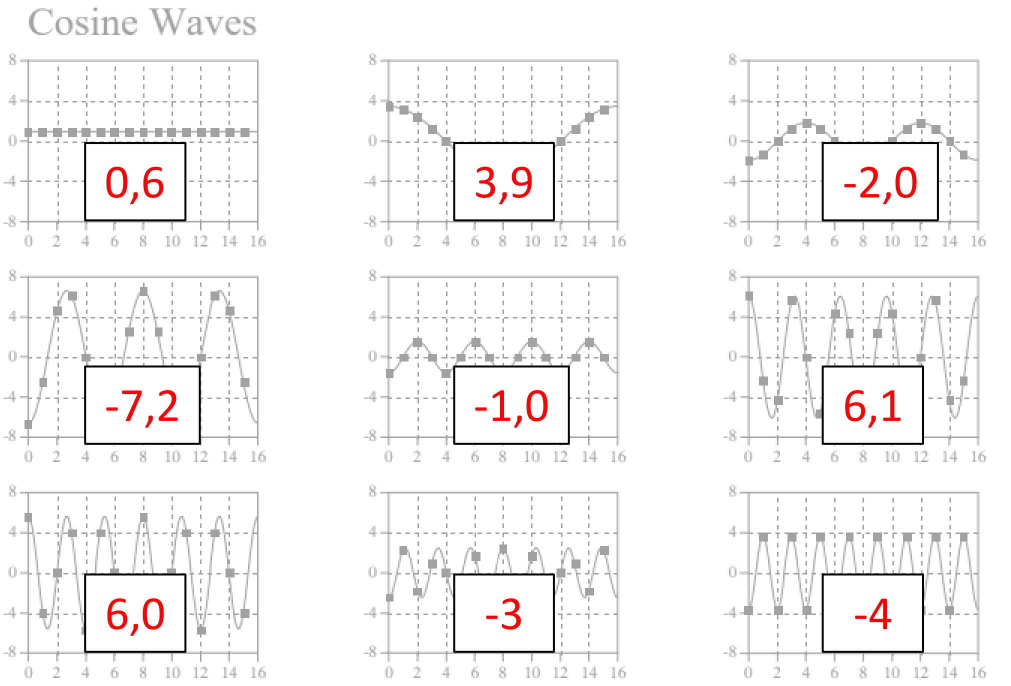
$$Re\,X[3] = x(0)\cos(2\pi \cdot 3 \cdot 0/16) + x(1)\cos(2\pi \cdot 3 \cdot 1/16) + x(2)\cos(2\pi \cdot 3 \cdot 2/16) + x(3)\cos(2\pi \cdot 3 \cdot 3/16) + x(4)\cos(2\pi \cdot 3 \cdot 4/16) + \ldots$$

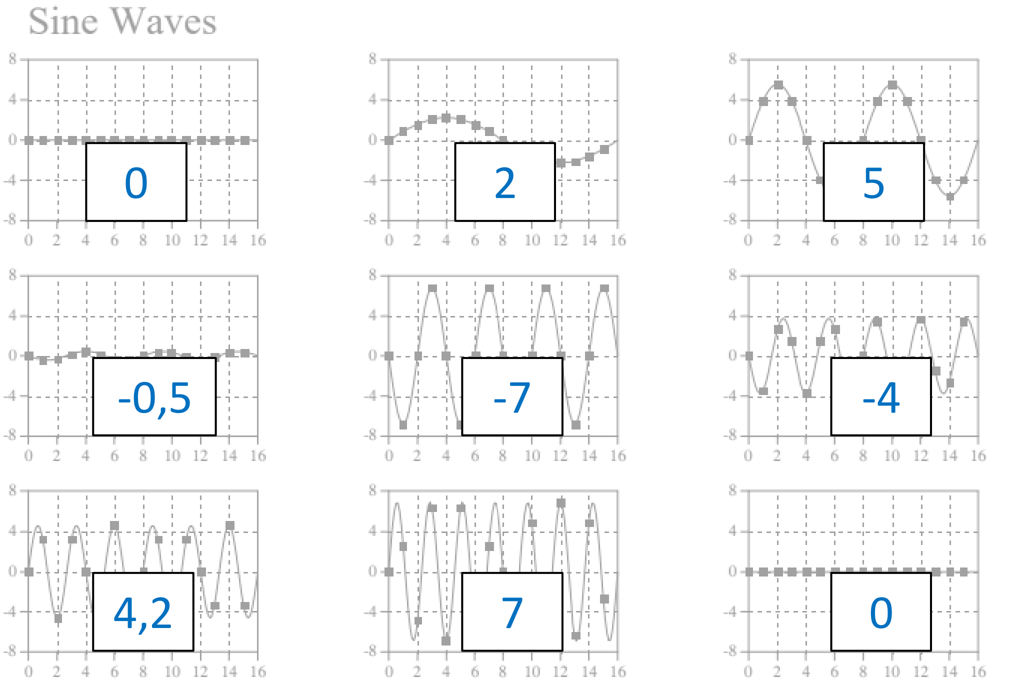
Datei Bearbeiten Ansicht Debuggen Ausführen Hilfe

v2.m | v1_beispiel.m

```
disp ("einfache diskrete Transformation");

# 16 Beispielwerte
werte = [ 0, 0, 0, 50, -18, -9
          -0.5, -0.2, -0.05, 0

function ret = fourier_cos(sam
  N   = length(samples);
  for k = ( 1 : N/2 +1)
    a(k) = 0;
    for n = ( 1 : N )
      a (k) = a(k) + samples(n
    endfor
  endfor
  ret = 2/N * a;
end

function ret = fourier_sin(sam
  N   = length(samples);
  for k = ( 1 : N/2 +1)
    a(k) = 0;
    for n = ( 1 : N )
      a (k) = a(k) + samples(n
    endfor
  endfor
  ret = 2/N * a;
end

fourier_cos(werte)
fourier_sin(werte)
sqrt ( fourier_cos(werte) .^2 + fourier_sin(werte) .^2 )
```

```
einfache diskrete Transformation
ans =

   1.8438    3.6538   -1.7155
  -6.9902   -1.7875    6.4094
   5.9655   -2.5730   -7.7688

ans =

   0         2.0388    5.8806
  -0.2669   -7.1812   -4.0068
   4.8056    7.2990    0.0000

ans =
   1.8438    4.1842    6.1257
   6.9953    7.4004    7.5587
   7.6603    7.7392    7.7688
```





2. Fourier-Transformation

<u>Was haben wir gemacht?</u>

- wir haben ein Rechenprogramm benutzt (GNU Octave)

- Information aus dem Zeitbereich (Abtastwerte, 16 Stück)
- transformiert in Sinus- und Cosinus-Amplituden Information (je 9 Stück)
- anschaulich Frequenz-Spektrum-Anteile

→ andere Darstellung der gleichen Information (transformierte Information, ohne Verlust!)

diskrete Fourier Transformation

cos() Amplituden

$$ReX[k] = \sum_{i=0}^{N-1} x[i] \cos(2\pi k i / N)$$

sin() Amplituden

$$ImX[k] = -\sum_{i=0}^{N-1} x[i] \sin(2\pi k i / N)$$

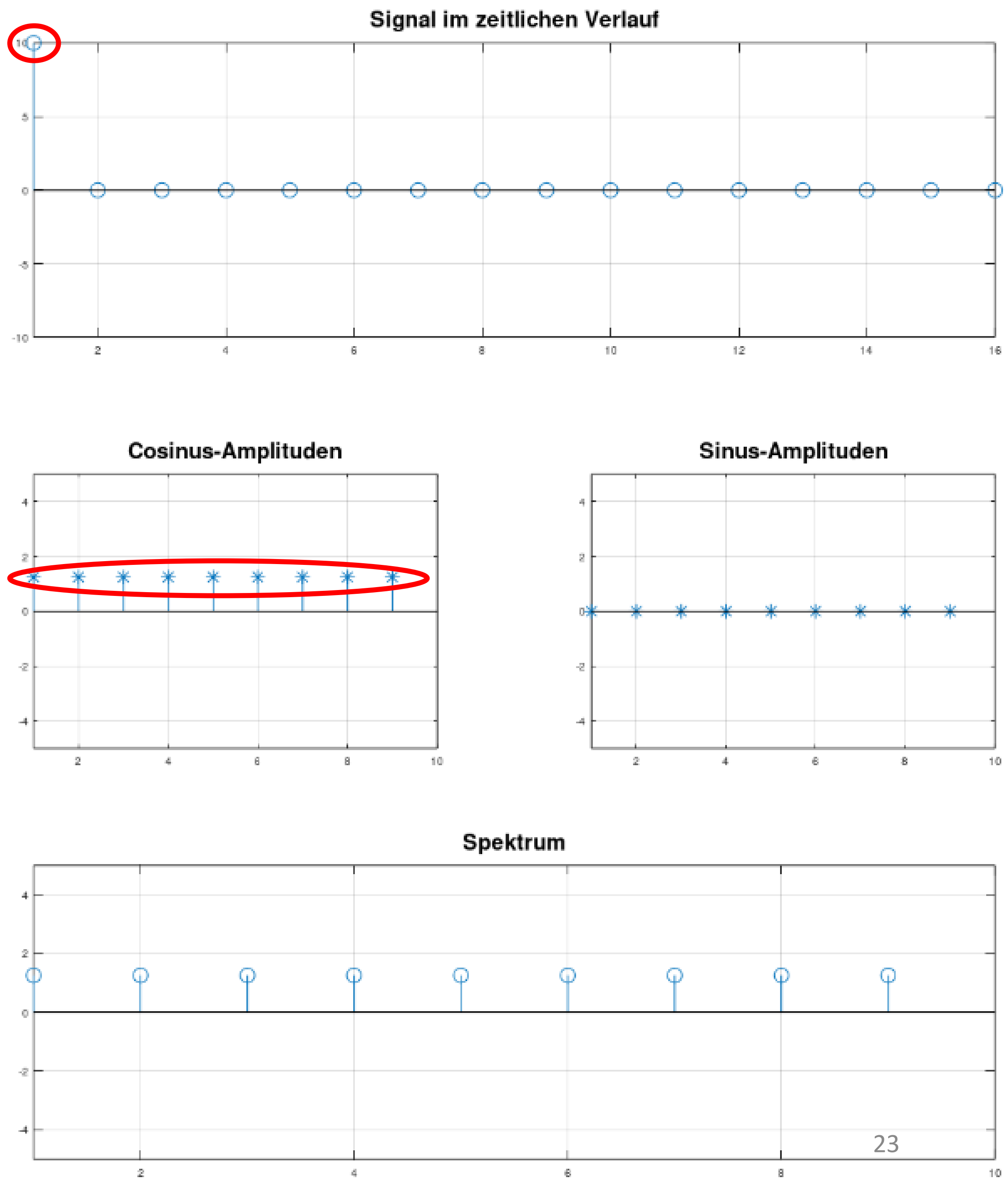
Quelle: https://www.dspguide.com/

2. Fourier-Transformation

Spielereien…

```
disp ("einfache diskrete Transformation");

# 16 Beispielwerte
werte = [ 10, 0, 0, 0, 0, 0, 0, 0, ...
          0, 0, 0, 0, 0, 0, 0, 0 ];

function ret = fourier_cos(samples)
function ret = fourier_sin(samples)

subplot(3,2,1:2);
stem (werte,"o"); grid on; axis([1 16 -10 10]);
title ("Signal im zeitlichen Verlauf", "fontsize",20);

subplot(3,2,3);
stem ( fourier_cos(werte), "*"); grid on; axis([1 10 -5 5]);
title ("Cosinus-Amplituden", "fontsize",20);

subplot(3,2,4);
stem ( fourier_sin(werte), "*"); grid on; axis([1 10 -5 5]);
title ("Sinus-Amplituden", "fontsize",20);

subplot(3,2,5:6);
stem (sqrt ( fourier_cos(werte) .^2 + fourier_sin(werte) .^2 ),"o")
grid on; axis([1 10 -5 5]);
title ("Spektrum", "fontsize",20);
```

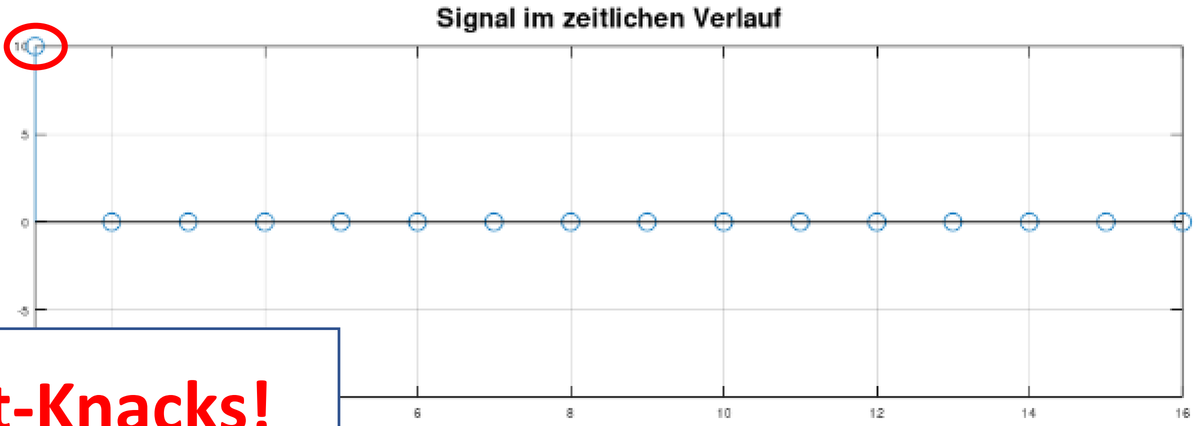

```
# 16 Beispielwerte
werte = [ 10, 0, 0, 0, 0, 0, 0, 0, ...
          0, 0, 0, 0, 0, 0, 0, 0 ];
```

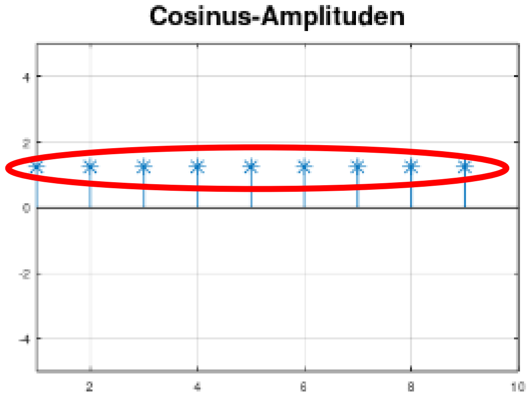
2. Fourier-Transformation

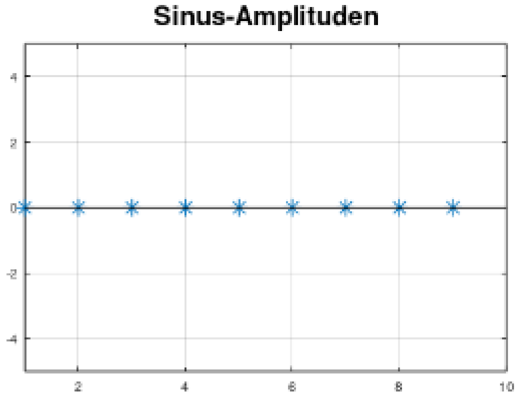
Spielereien…

```
disp ("einfache diskrete Transformation");

# 16 Beispielwerte
werte = [ 10, 0, 0, 0, 0, 0, 0, 0, ...
          0, 0, 0, 0, 0, 0, 0, 0 ];

function ret = fourier_cos(samples)
function ret = fourier_sin(samples)

subplot(3,2,1:2);
stem (werte,"o"); grid on; axis([1 16 -10 10]);
title ("Signal im zeitlichen Verlauf", "fontsize",20);

subplot(3,2,3);
stem ( fourier_cos(werte), "*"); grid on; axis([1 10 -5 5]);
title ("Cosinus-Amplituden", "fontsize",20);

subplot(3,2,4);
stem ( fourier_sin(werte), "*"); grid on; axis([1 10 -5 5]);
title ("Sinus-Amplituden", "fontsize",20);

subplot(3,2,5:6);
stem (sqrt ( fourier_cos(werte) .^2 + fourier_sin(werte) .^2 ),"o")
grid on; axis([1 10 -5 5]);
title ("Spektrum", "fontsize",20);
```

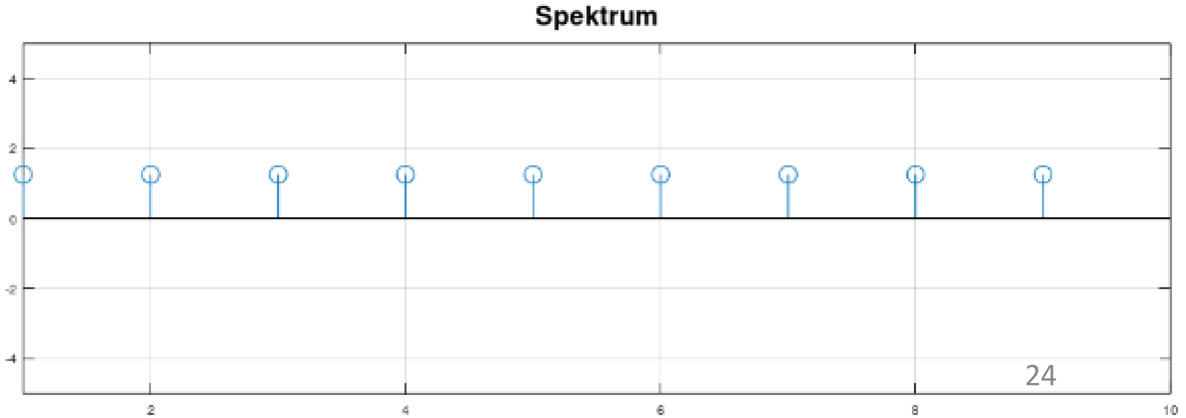


ein Einschalt-Knacks!






```
# 16 Beispielwerte
werte = [ 10, 0, 0, 0, 0, 0, 0, 0, ...
          0, 0, 0, 0, 0, 0, 0, 0 ];
```



2. Fourier-Transformation

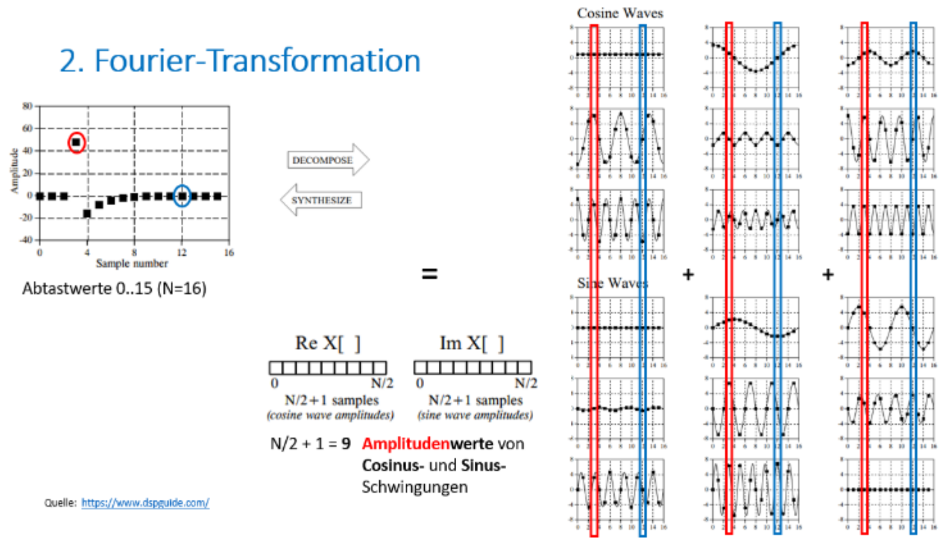
```
function ret = ift_cs(Fc, Fs)
  N   = length(Fc)*2-1;
  for i = ( 1 : N )
    x(i) = 0;
    for k = ( 1 : N/2+1 )
      x (i) = x(i) + Fc(k) * cos ( 2*pi * (k-1) * (i-1) /N) ...
                  + Fs(k) * sin ( 2*pi * (k-1) * (i-1) /N);
    endfor
  endfor
  ret = x;
end
```

und rückwärts: <u>**inverse Fourier Transformation**</u>

cos() Amplituden sin() Amplituden

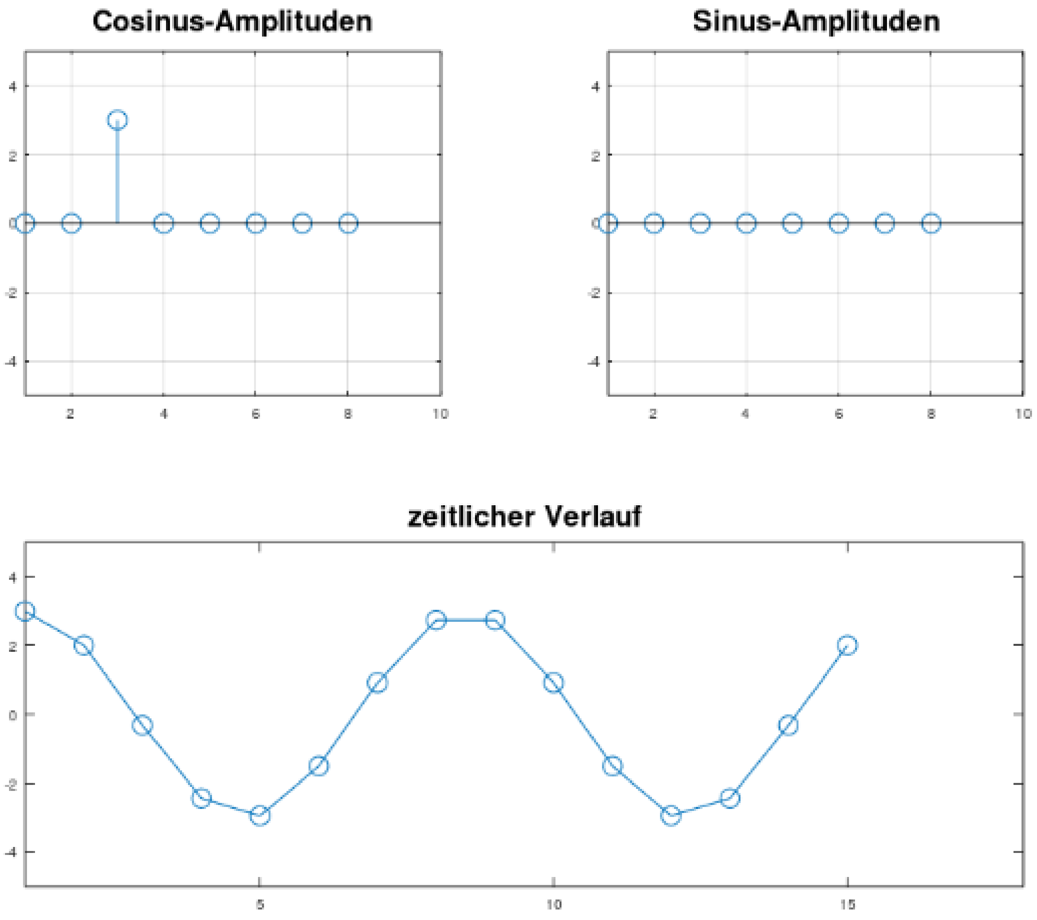
$$x[i] = \sum_{k=0}^{N/2} Re\bar{X}[k]\cos(2\pi ki/N) + \sum_{k=0}^{N/2} Im\bar{X}[k]\sin(2\pi ki/N)$$

und das hatten wir vorhin schon anschaulich „im Kopf“ und grafisch berechnet…



2. Fourier-Transformation

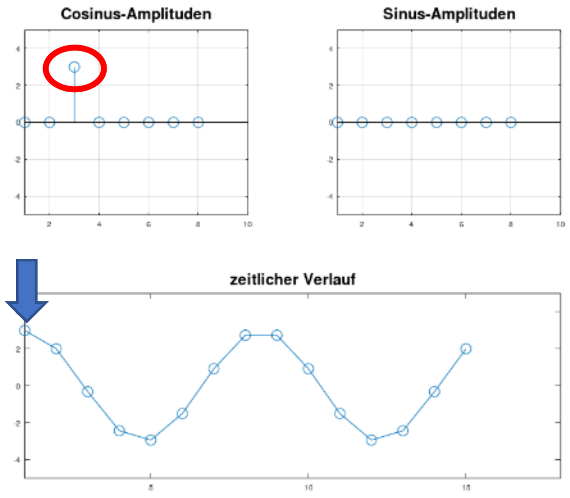
Spielereien…

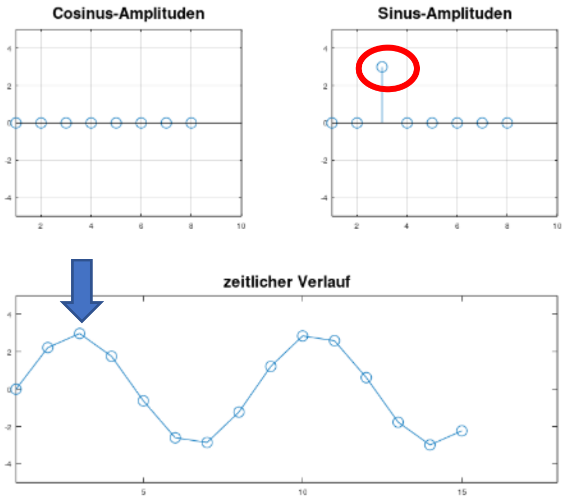


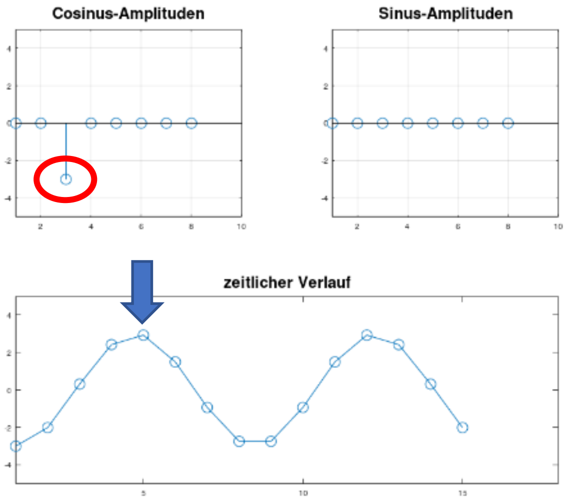

```
pkg load signal;

F_Cosinus = [0, 0, 3, 0, 0, 0, 0, 0 ];
F_Sinus   = [0, 0, 0, 0, 0, 0, 0, 0 ];

function ret = ift_cs(Fc, Fs)
  N   = length(Fc)*2-1;
  for i = ( 1 : N )
    x(i) = 0;
    for k = ( 1 : N/2+1 )
      x (i) = x(i) + Fc(k) * cos ( 2*pi * (k-1) * (i-1) /N) ...
                   + Fs(k) * sin ( 2*pi * (k-1) * (i-1) /N);
    endfor
  endfor
  ret = x;
end


f = ift_cs(F_Cosinus, F_Sinus);


# subplot (3,2,1:2);  plot(t,f,"o-"); # axis ([0 t_tot -1.5 1.5]);
subplot (3,2,1);
stem(F_Cosinus); axis([1 10 -5 5]); grid on;
title ("Cosinus-Amplituden", "fontsize",20);

subplot (3,2,2);
stem(F_Sinus); axis([1 10 -5 5]); grid on;
title ("Sinus-Amplituden", "fontsize",20);

subplot (3,2,3:4);
plot(f, "o-"); axis([1 18 -5 5]);
title ("zeitlicher Verlauf", "fontsize",20);
```

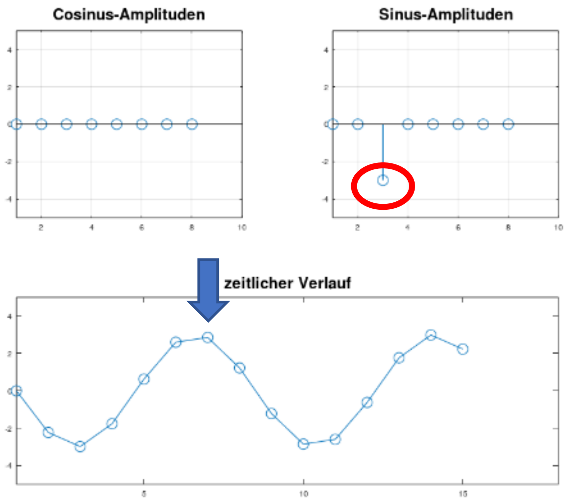
2. Fourier-Transformation

Spielereien…





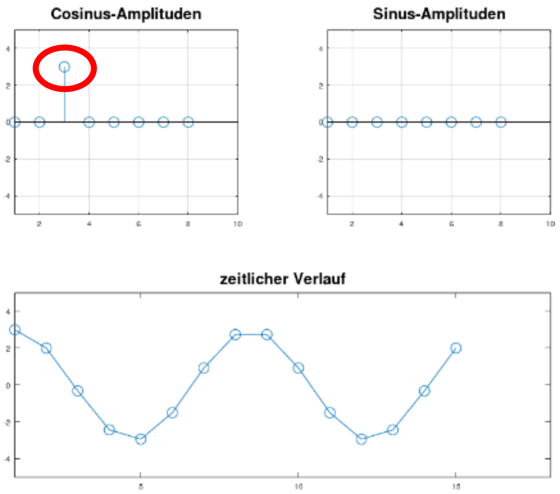


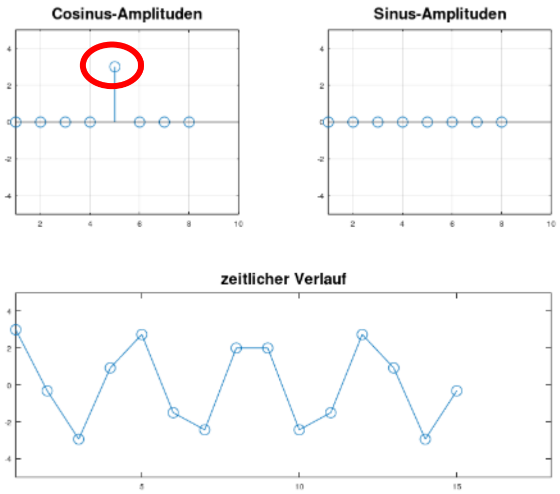


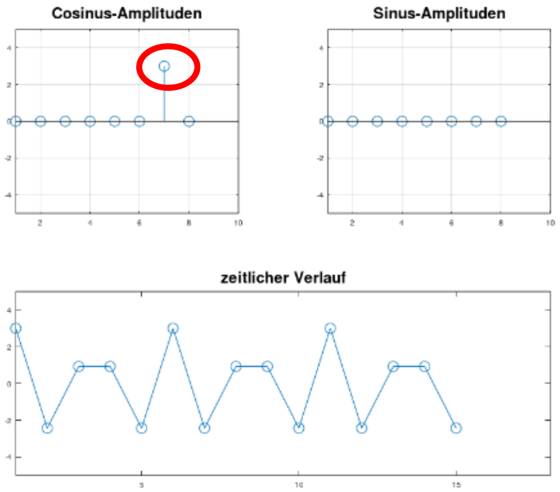
Wir haben ein QPSK mit f=2 erzeugt!

2. Fourier-Transformation

Spielereien…

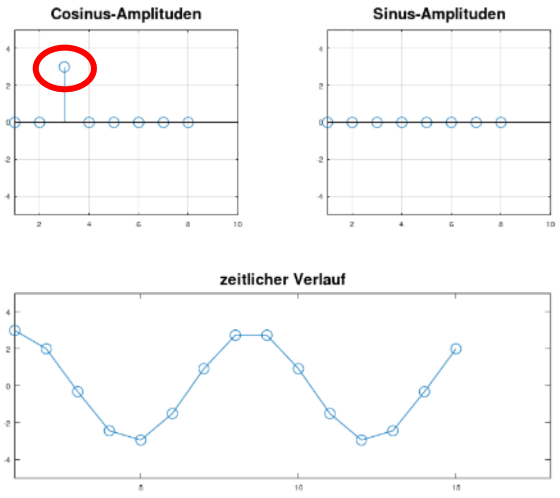


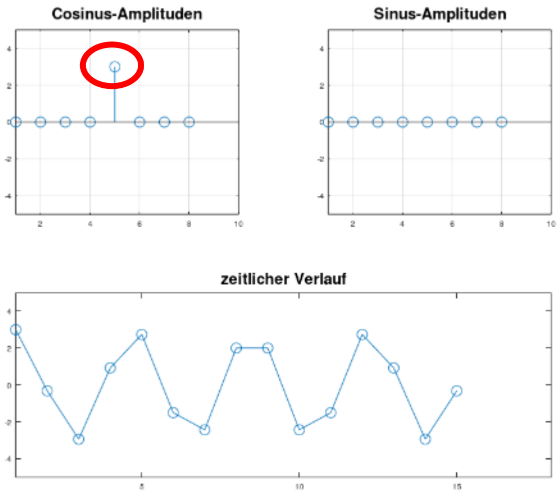


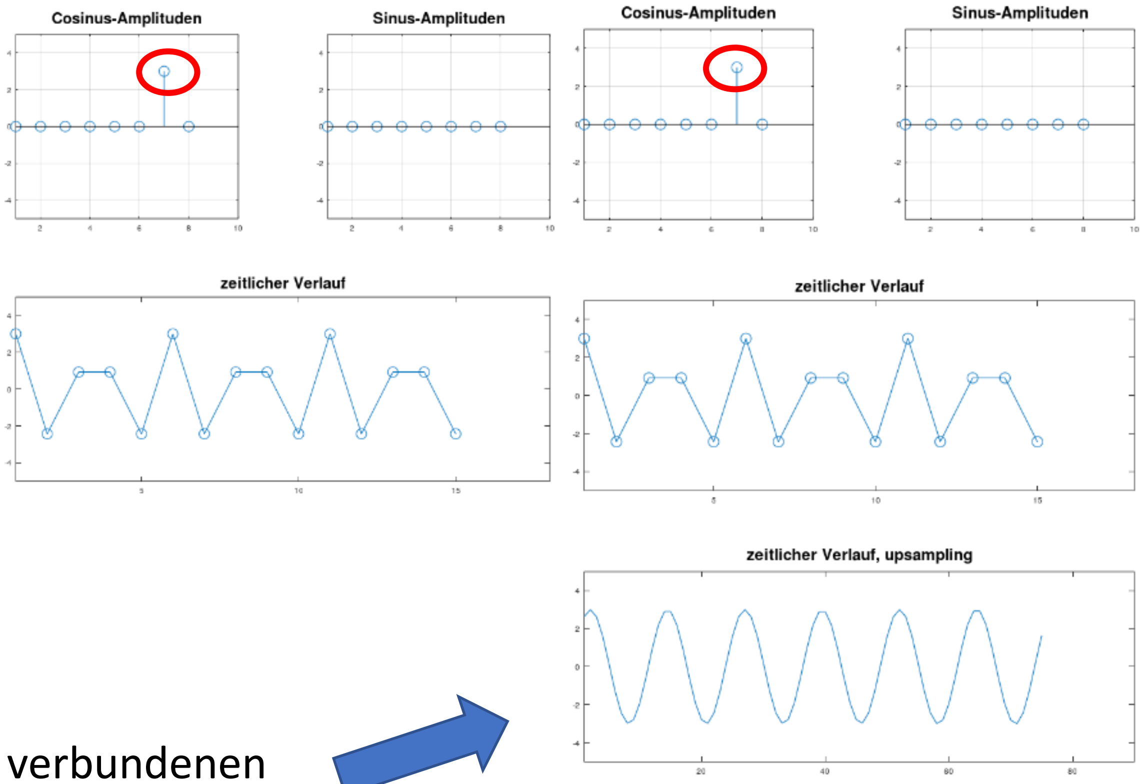


2. Fourier-Transformation

Spielereien…



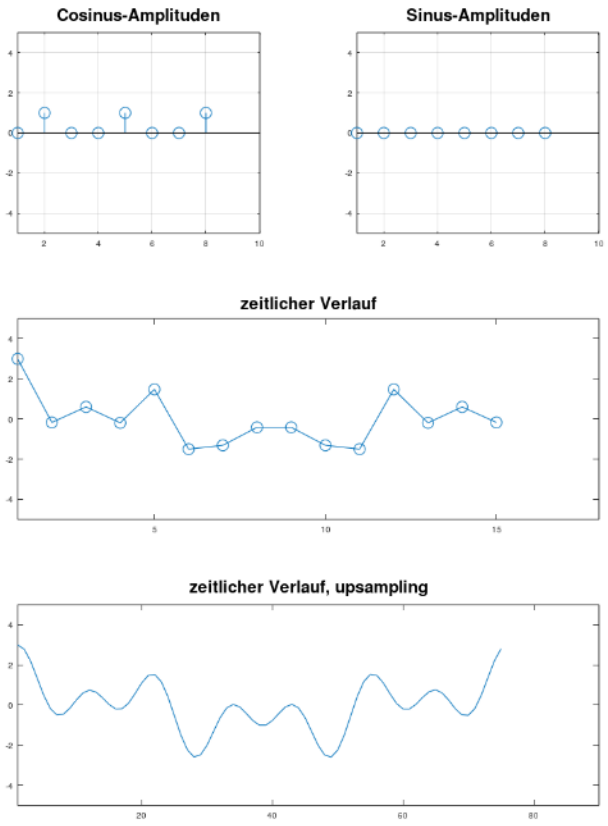


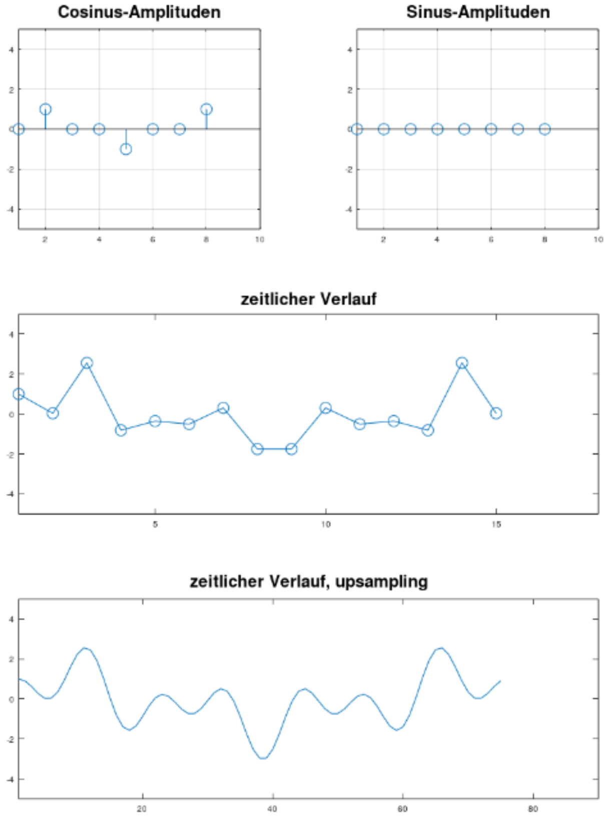


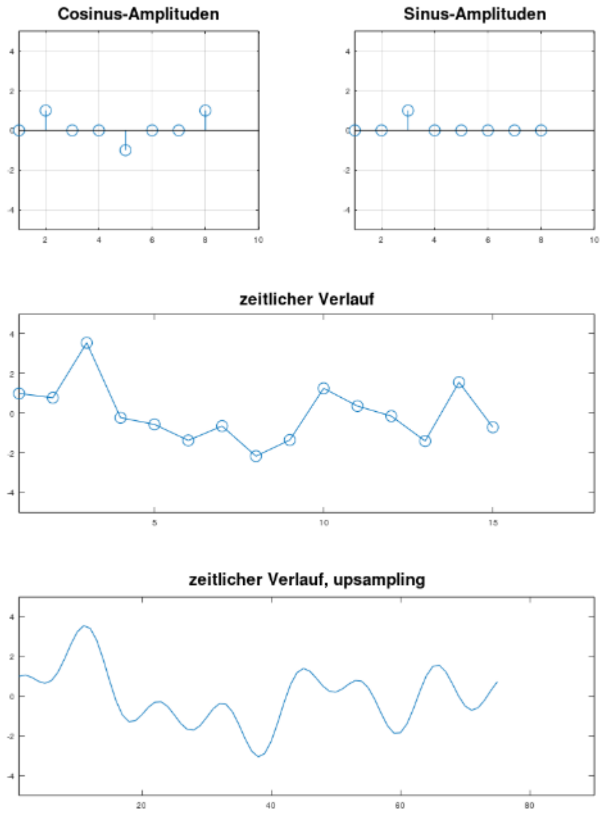
Die Darstellung mit verbundenen Punkten im Zeitbereich ist schwer erkennbar; mit etwas Tricksen…

2. Fourier-Transformation

OFDM!...



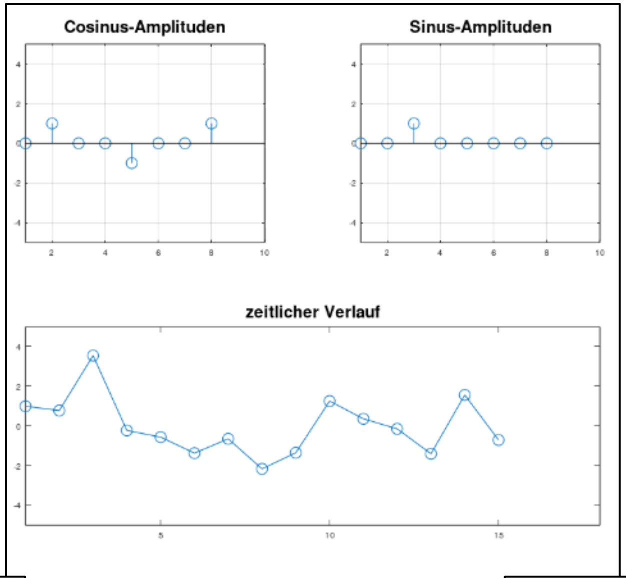




<u>Beobachtung 1:</u>
Die Amplituden im Zeitbereich „ergeben“ sich einfach durch die **inverse Fourier-Transformation** der Informations-Symbole aus dem Frequenzbereich.

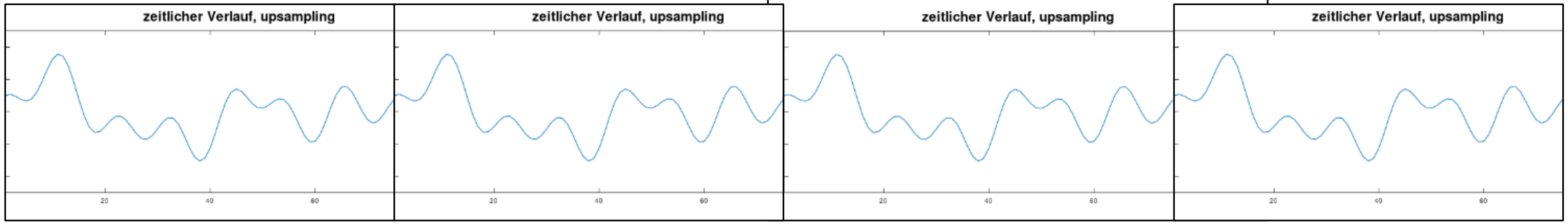
2. Fourier-Transformation

OFDM!…



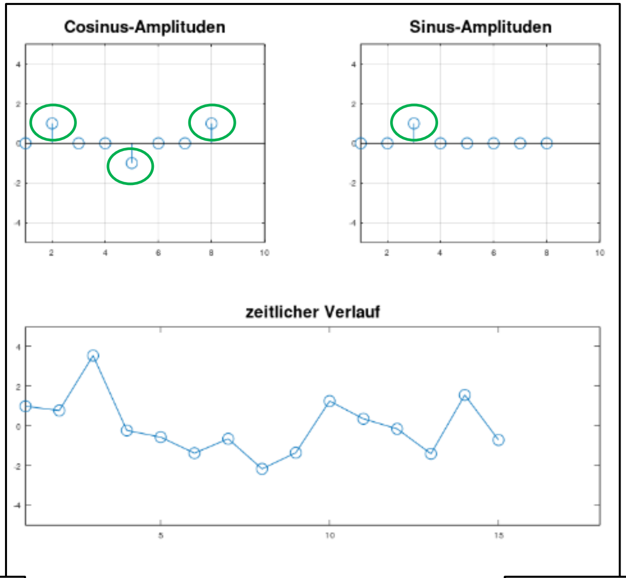
Beobachtung 1:
Die Amplituden im Zeitbereich „ergeben“ sich einfach durch die **inverse Fourier-Transformation** der Informations-Symbole aus dem Frequenzbereich.

Beobachtung 2:
der zeitliche Verlauf eines OFDM Signals ergibt sich durch einfache **Wiederholung** der inversen Fourier-Transformierten….!



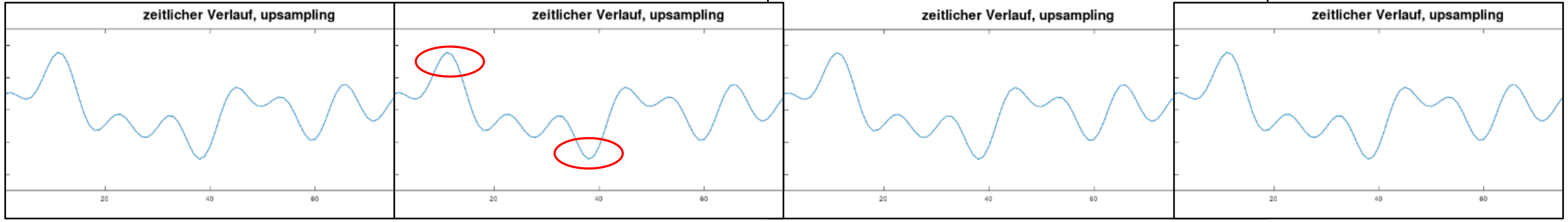
2. Fourier-Transformation

OFDM!...



<u>Beobachtung 1:</u>
Die Amplituden im Zeitbereich „ergeben“ sich einfach durch die **inverse Fourier-Transformation** der Informations-Symbole aus dem Frequenzbereich.

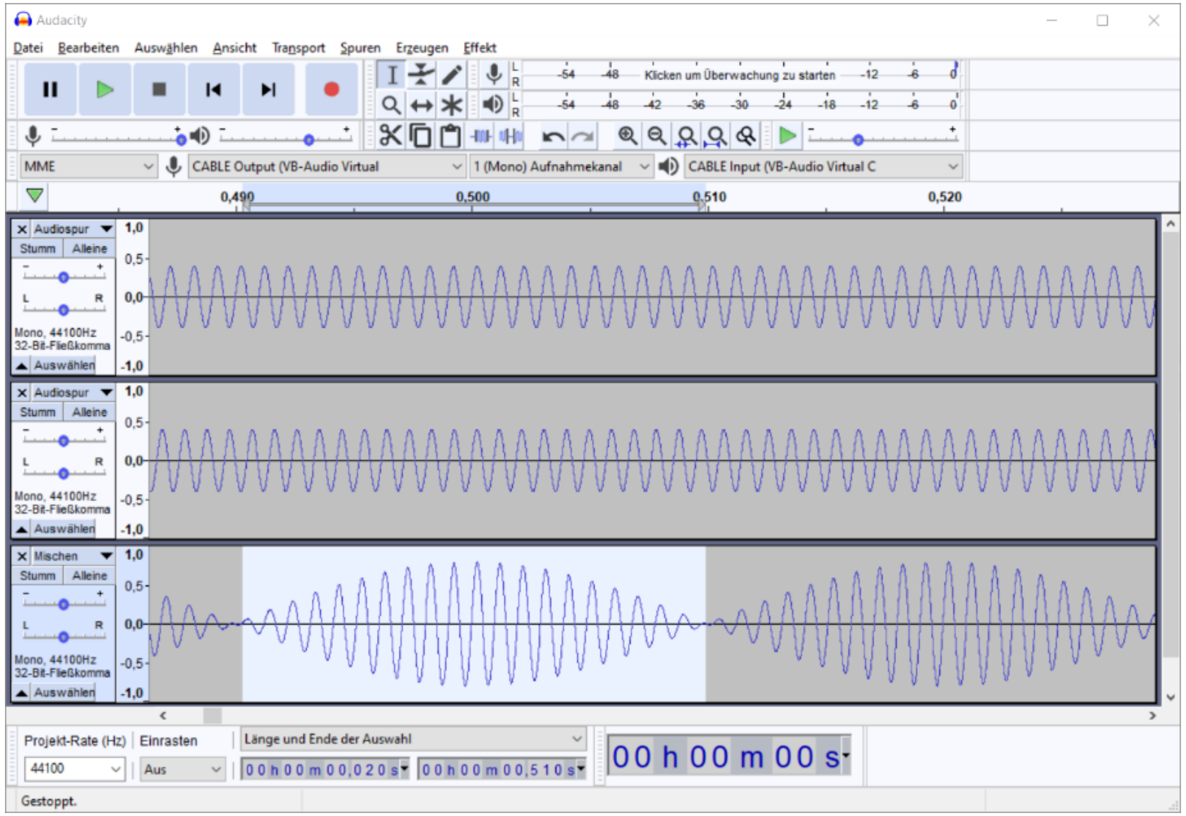
<u>Beobachtung 2:</u>
der zeitliche Verlauf eines OFDM Signals ergibt sich durch einfache **Wiederholung** der inversen Fourier-Transformierten….!



<u>Beobachtung 3:</u>
es ergeben sich **Extremwerte** …

2. Fourier-Transformation

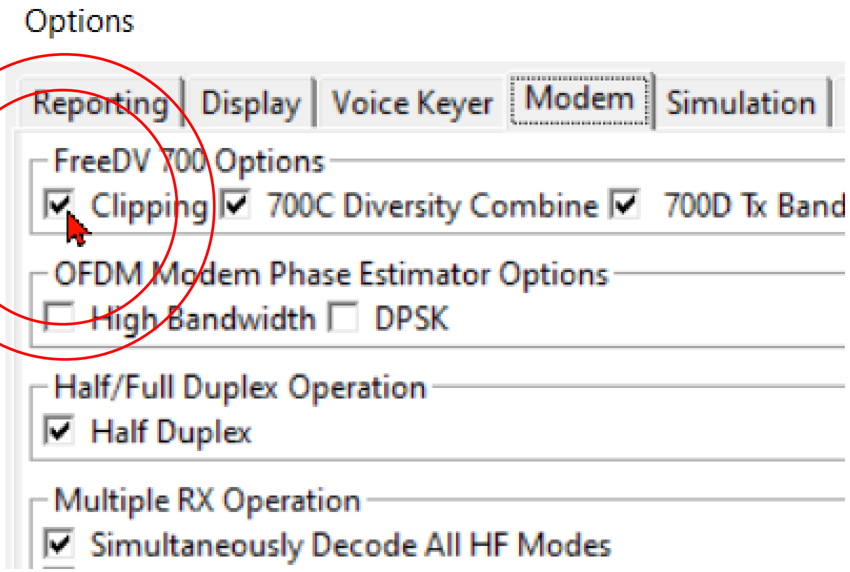
→ Folie vom 7. Januar !!



Darstellung von letzten Treffen:

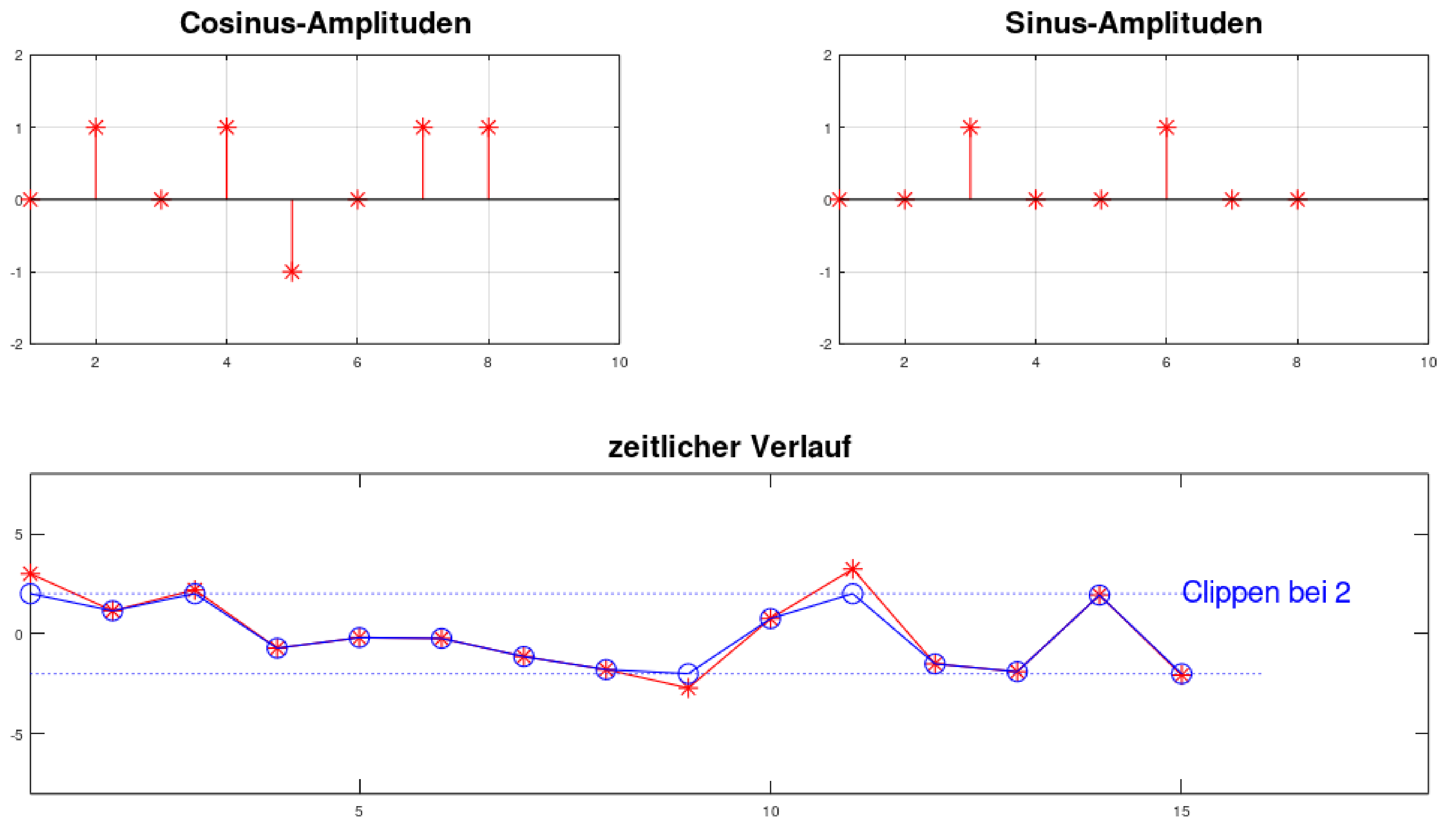
→ überlagerte einzelträger führen zu Maxima bzw. Auslöschung, abhängig von der Phasenlage

- da ist doch eine Einstellung unter (Tools → Options → Modem → Clipping)
- was macht das?



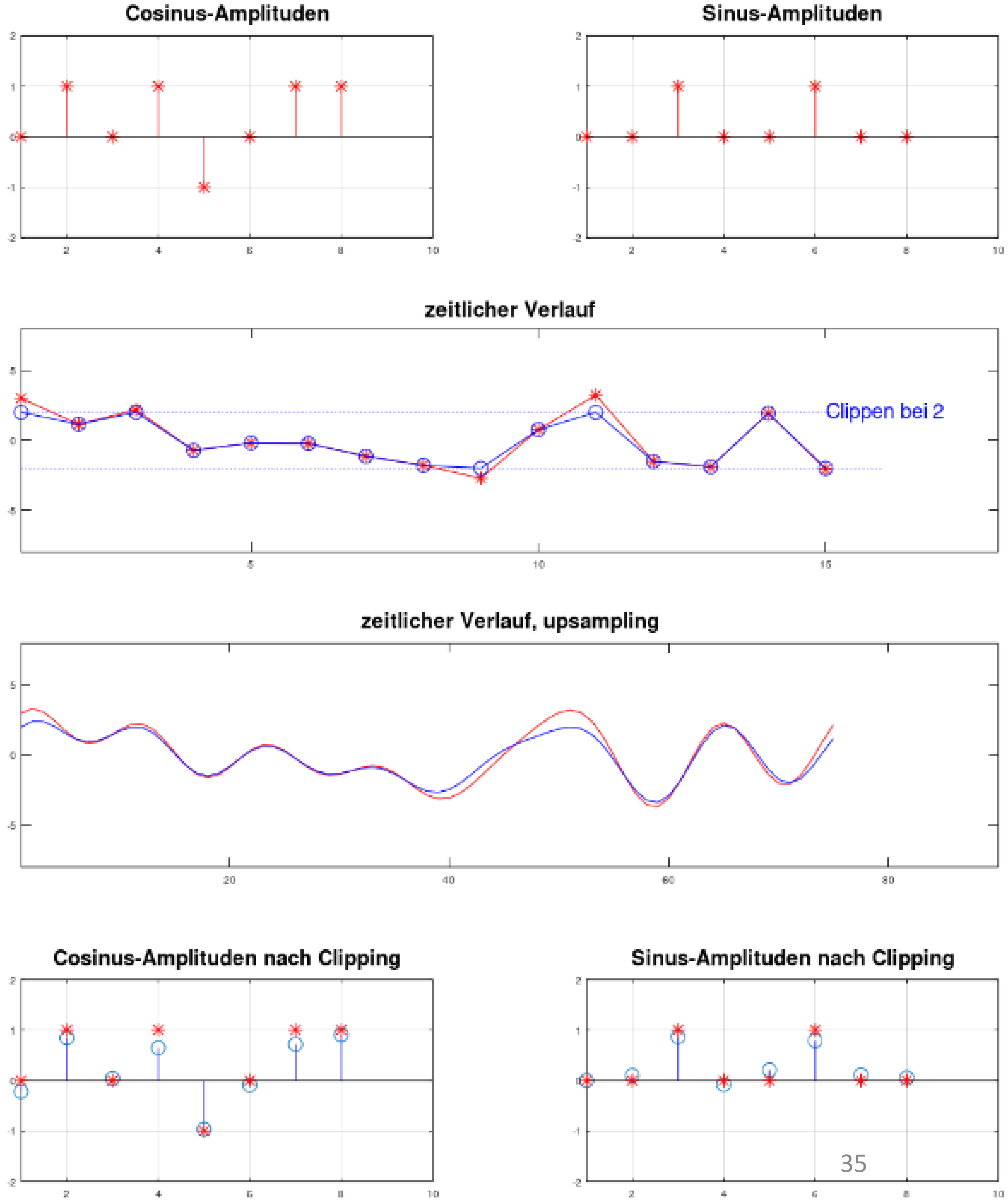
2. Fourier-Transformation

OFDM mit Clipping!



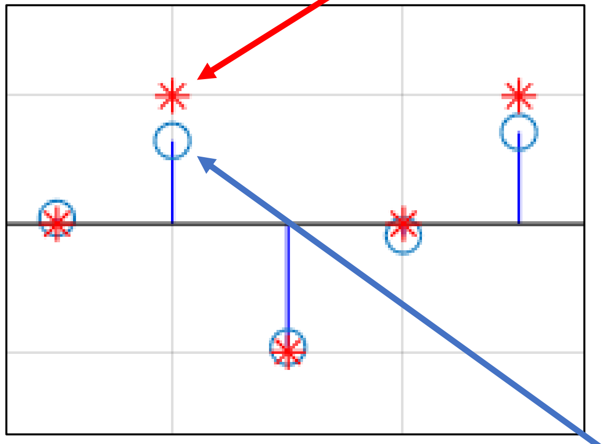
2. Fourier-Transformation

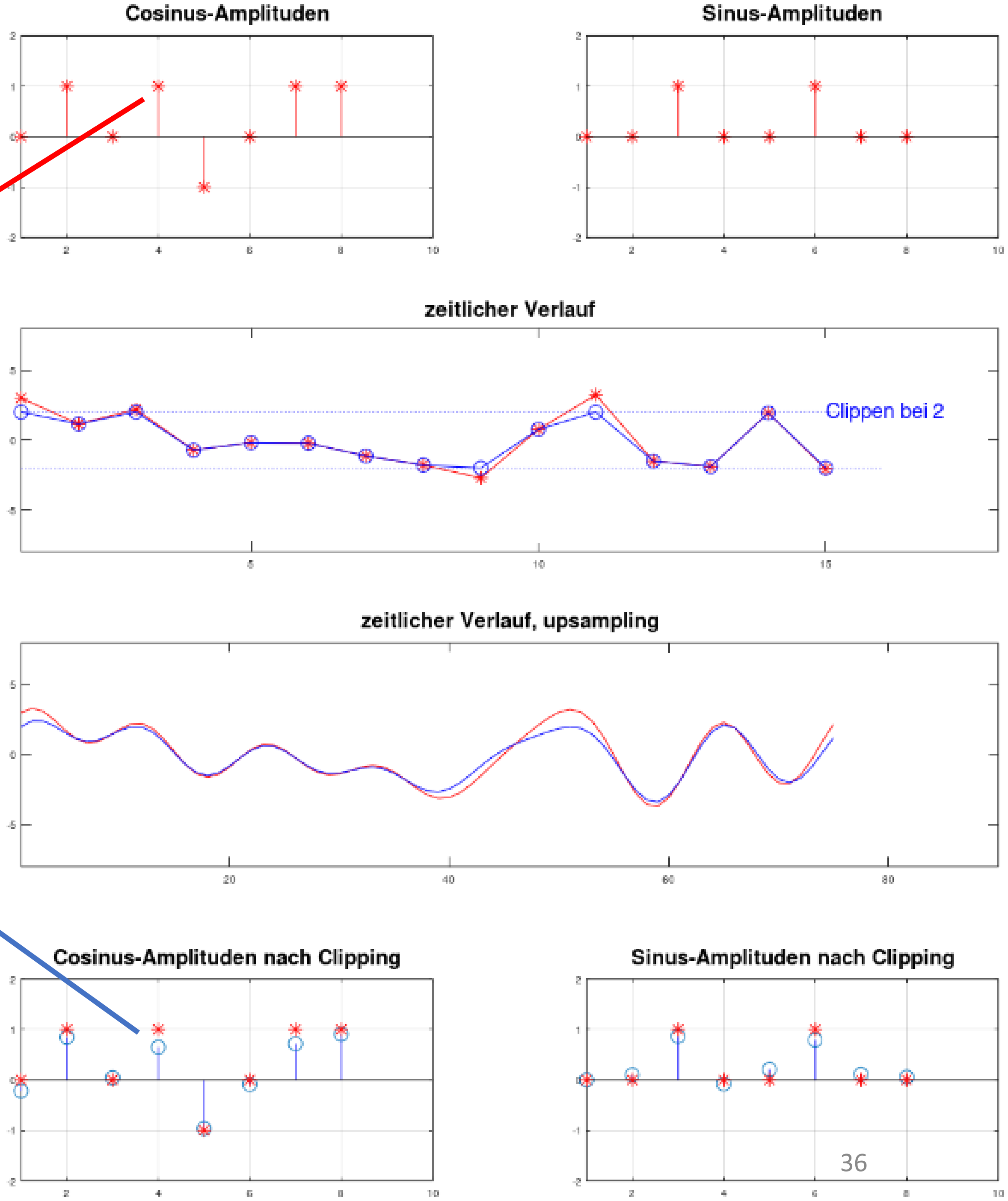
OFDM mit Clipping!



2. Fourier-Transformation

OFDM mit Clipping!





<u>Einsicht:</u>
alle Symbole tragen irgendwie zu Extremwerten bei;
Clippen der Extremwerte beeinträchtigt
alle **Quellsymbole**…

HAM group FreeDV

2. Fourier-Transformation

Die gezeigten Algorithmen sind unglaublich ineffizient (viele Rechenoperationen für wenig Ergebnis).

Von James Cooley und John Turkey wurde 1965 eine effiziente Berechnungsmethode für die diskrete Fourier-Transformation wieder entdeckt und für moderne Rechner-Systeme adaptiert:

> **Schnelle Fourier Transformation,**
> **Fast Fourier Transformation** (FFT)

Das ist einer der gängigen Algorithmen, die heute regelmäßig verwendet werden.

Quelle: https://de.wikipedia.org/wiki/Schnelle_Fourier-Transformation

FreeDV Treffen Anfang April 2022

1. Terminplanung

2. **Fourier-Transformation**
   - OFDM Signal-Erzeugung
   - bekannte Anwendungen der Fourier-Transformation
   - analytische, **diskrete** und *schnelle* Fourier-Transformation

3. offene Diskussion