Zeitdiskrete Signale

Grundlegende Verarbeitung zeitdiskreter Signale mit MATLAB

Zeitdiskrete Signale

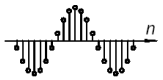
- Signal = mathematische Funktion von min. einer unabhängigen Variablen ⇒ x(t)
- t = Zeitindex, t = diskrete Werte ⇒ zeitdiskretes Signal, Folge x[n], n normierte Zeitvariable
- rechtsseitige Folge, wenn x[n] = 0, n < 0
- Abtastfolge mit Abtastintervall Ts ⇒ x[n] = x(t=nTs)

Elementare Signale I

- Sinoidalschwingung
  - Sinussignal
  - Kosinussignal
    - Ω => normierte Kreisfrequenz (2πf)
    - n = normierte Zeitvariable
    - φ = Phase
- Einheitssprung

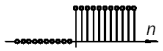
Sinusförmiges Signal

$$x(n) = \cos(\Omega_0 n + \varphi_0)$$



Einheitssprung

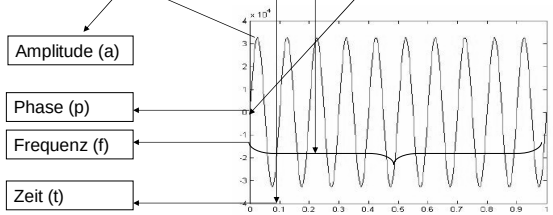
$$s(n) = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases}$$



Grafiken aus:
Hess, Wolfgang. Vorlesungsskripte zur Signalverarbeitung.
IKP, Universität Bonn, http://www.ikp.uni-bonn.de

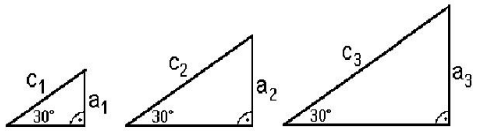
Sinus - Cosinus

- mathematische Herleitung der Sinuswelle
- Ziel: $y = a * \sin(2\pi * t * f + p)$



Sinus - Cosinus

- Wie kommt man zu dieser Formel?
- Erinnerung Mathe - Dreiecke !
- Ähnliche Dreiecke: Sind die Winkel in verschiedenen Dreiecken gleich = ähnliche Dreiecke



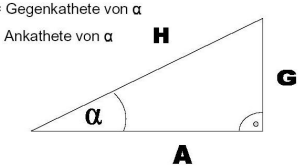
Sinus - Cosinus

- Winkelsummensatz: Summe der Winkel im Dreieck = 180°
- Ähnliche Dreiecke haben gleiches Seitenverhältnis =>
  a1/c1 = a2/c2 = a3/c3 (siehe vorh. Abb.)

H = Hypotenuse

G = Gegenkathete von α

A = Ankathete von α



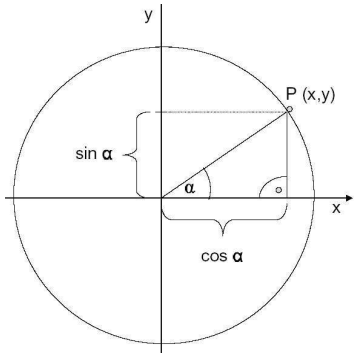
Berechnen: Sinus von α ?
Sinus α = Verhältnis von G/H
Bsp: G=2; H=4
= 2/4 = 0.5 => (inv) sin 0.5 = 30°
bzw.: sin 30°= 0.5

Analog Cosinus: cos α = A/H

Sinus - Cosinus

- Einheitskreis:
  Radius r = 1
- Hypothenuse H = r = 1
- da r = 1 =>
  sin α = G/1 = G
- cos α = A/1 = A

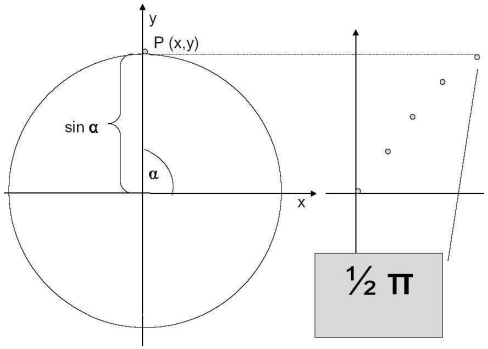
- x-Wert von P => sin α
- y-Wert von P => cos α



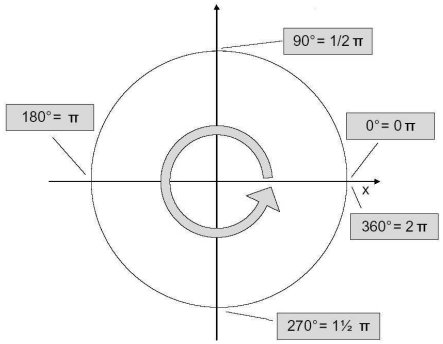
Sinus - Cosinus

- Degree – Radian
- Degree (deg): Winkelmaß
- Radian (rad): Bogenmaß
- Berechnung: Umfang Kreis = r * 2π
  => bei Einheitskreis: 1 * 2π = 2π
  => Winkel α lässt sich auch mit 2π messen

Sinus - Cosinus



Sinus - Cosinus



Sinussignal

- Formel zur Erinnerung:

$$y = a * \sin(2\pi * t * f + p)$$

Sinussignal

- Sinussignal mit MATLAB
- Zeit Vektor (t) bestimmen
- t = 0:0.1:1;
- Für jedes Element des Zeitvektors Sinus berechnen und mit 2π multiplizieren
- y = sin(2π * t);
- plot (t,y);
- Sinus-Wave sieht nicht so aus wie gewollt – warum?
- Lösung: Abtastrate erhöhen =>
- t = 0:0.01:1; plot(t,y);

Sinussignal

- Periode erhöhen – wie?
- Mit Faktor multiplizieren y = sin(2π * 2 * t);
- Abtastrate noch zu niedrig => gewollt CD-Qualität (44100 Samples/Sek) – wie?
- Abtastrate erhöhen => t = 0:1/44100:1
- Amplitude verkleinern – wie?
- mit Faktor a multiplizieren
  y = ½ sin(2π * 2 * t);

Sinussignal

- Erstellen eines Sinussignals mit 440 Hz und anschließend anhören
- t = 0:1/44100:1;
- y = ½ * sin(2 * pi * t * 440);
- plot(t,y);
- sound(y,44100,16);

- Erstelltes Sinussignal als Wav-File speichern
- wavwrite() + Argumente = wavwrite(y,44100,‘sinus‘);
-

Sinussignal

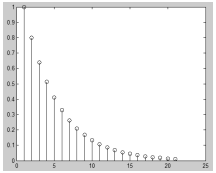
- Gespeichertes Sinussignal ist nicht genau 1 Sekunde, sondern 1.000023 Sekunden Länge – Wieso?
- Lösung:
  Bei Zeitvektor (t) 0 bis 1 haben wir einen Wert – „die 0“ – zuviel
  => 1Sample wieder subtrahieren
  => t = 0:1/44100:1-1/44100;

Grundlegende Operationen

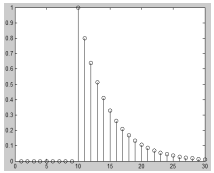
- Transformation auf der Zeitachse
  - Verzögerung
  - Inversion
  - Dezimation

Grundlegende Operationen

- Originalsignal
  - x = 0.8 .^ [0:20]
  - stem(x)

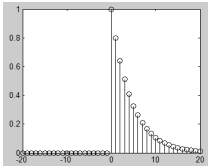


- Verzögerung
  - take vector x
  - create matrix of zeros and x
  - stem matrix

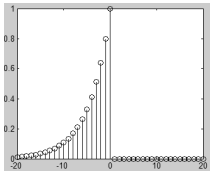


Grundlegende Operationen

- Originalsignal
  - n = -20:20;
  - x = 0.8 .^ [0:20];
  - x = [zeros(1,20), x];
  - stem(n,x)



- Inversion
  - create time vector
  - create exp signal
  - create matrix of zeros and x
  - create vector and store values from end to 1
  - stem time vector, rev-vector

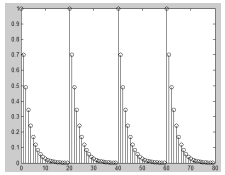


Grundlegende Operationen

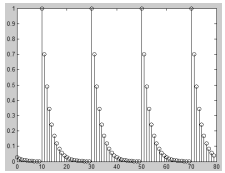
- Zeitverschiebung periodischer Signale = zyklische Zeitverschiebung
- Beachte:
  Vektorlänge = Vielfaches der Periodendauer
- Berechnung in MATLAB durch Einsatz des Modulo Operators rem() => help rem
- Bei negativem Zeitindex => Fehler in MATLAB. Abhilfe durch Addition der Periodendauer zur Signalverzögerung

Grundlegende Operationen

- periodisches Signal
  - N = 20;
  - n = 0:N-1;
  - x = 0.7 .^ n;
  - n = 0:4*N-1;
  - xp = x(rem(n,N)+1);
  - stem(n,xp);



- Zeitverschobenes Signal
  - add vector to delay

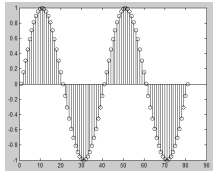


Grundlegende Operationen

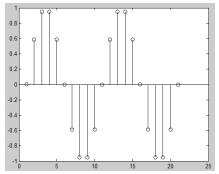
- Dezimation (Unterabtastung)
- Anwendung bei Abtastrate, Speicherung und Darstellung von Signalen
  - Dezimation mittels ganzzahligem Faktor M
  - x(n) -> x(nM)
    =>
  - Signallänge wird um Faktor M reduziert Echtzeitverarbeitung, Abtastintervall wird um Faktor M vergrößert

Grundlegende Operationen

- Originalsignal
  - n = -40:40;
  - theta = pi/20;
  - x = sin(theta*n);
  - stem(x);



- dezimiertes Signal
  - define decimation factor
  - create vector with interval size M from 1 to end
  - stem vector



Grundlegende Operationen

- Praktische Anwendung der Dezimation
  - Darstellung eines Sprachsignals am Bildschirm
  - Es werden nicht alle Amplitudenwerte dargestellt, sondern nur jeder M-te Amplitudenwert => Verfälschung der Signalstruktur
  - Vorgehen:
  - M Abtastwerte werden zu einem Block zusammengefasst und für jeden Block das Min und Max der Amplituden bestimmt
- Grafische Abbildung der Min-, Max-Werte =>
  zeigt etwa Einhüllende des Sprachsignals

Grundlegende Operationen

- MATLAB Funktionen für Dezimation
  - Hilfe in MATLAB mit help funktion

- wavread() % liest MS wav-files ein
- length() % bestimmt die Länge des Arguments
- floor(), ceil() % ab-, aufrunden in MATLAB
- max(), min() % größtes, kleinstes Element bestimmen
- hold() on/off % Ausgabe wird zwischengespeichert
- colormap() % zeichnet farbige Ausgabe
  (0 0 0) = schwarz
  (1 1 1) = weiß
  Werte zw. 0 und 255

Grundlegende Operationen

- Beispielanwendung in MATLAB[1] für Dezimation

```
x = wavread('test.wav');        %read in wav-file
M = 20;                         %define decimation factor
L = length(x);                  %length of wav-file
B = floor(L/M);                 %num blocks of size m
L = B*M                         %set length to product B and M
X = zeros(M,B);                 %create zero matrix of M x B
X(:) = x(1:L);                  %transform X in columns and store M samples
Xmax = max(X);                  %vector of maxima of x
Xmin = min(X);                  %vector of minima of x
t = 0 : B-1;                    %time vector

hold on, area(t,Xmax),
area(t,Xmin),hold off,
colormap([0 0.6 1]), grid on    %graphical output
```

1) Beispiel aus Doblinger, G.: MATLAB Programmierung in der digitalen Signalverarbeitung, Schlembach, 2001