FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

2. PŘEDNÁŠKA – Tabulková syntéza

- **Generování číslicových signálů**
  - Aperiodické signály
  - Periodické signály
  - Zvuky telefonu
  - Tónová volba
- **Hudební stupnice**
  - Temperované ladění
  - Příklad v MATLABu
- **Tabulková (wavetable) syntéza**
  - Tabulkový oscilátor
  - Interpolace
  - Změna (posunutí) základní periody
  - Příklady tabulkové syntézy

Generování číslicových signálů

- ***Signály*** – funkce jedné nebo více nezávisle proměnných, které nesou *informaci* o podstatě a vlastnostech svého zdroje (nebo informaci záměrně do signálu zakódovanou).

  - Příklady signálů
  - Co není signál?

Generování číslicových signálů

- **Neperiodické**
- **Periodické, harmonické**

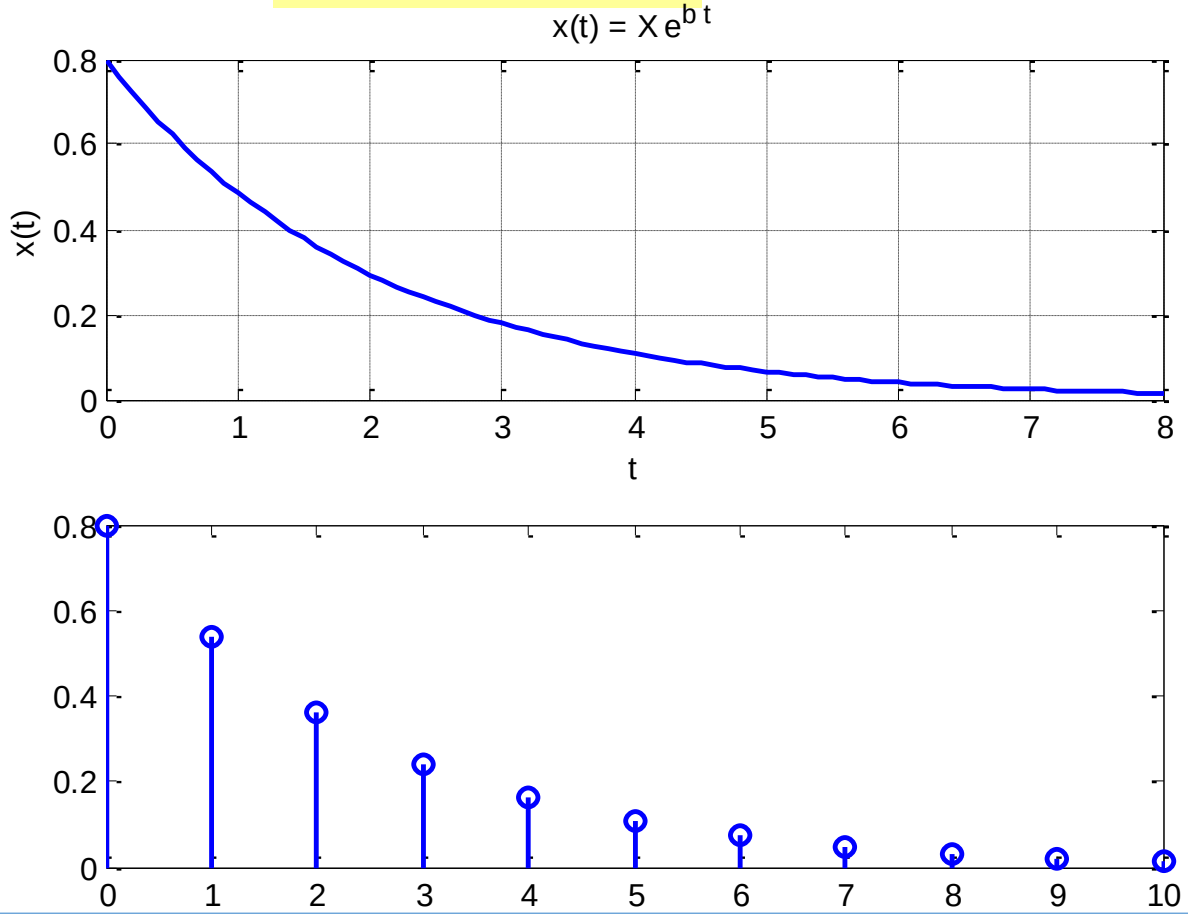
Generování číslicových signálů

- **Exponenciální signál**

$$x(t) = a \cdot \mathrm{e}^{bt}$$

$$x[n] = a \cdot c^{n}$$

```
t=0:0.1:8;
a=0.8; b=-0.5;
x_t=a*exp(b*t);
```



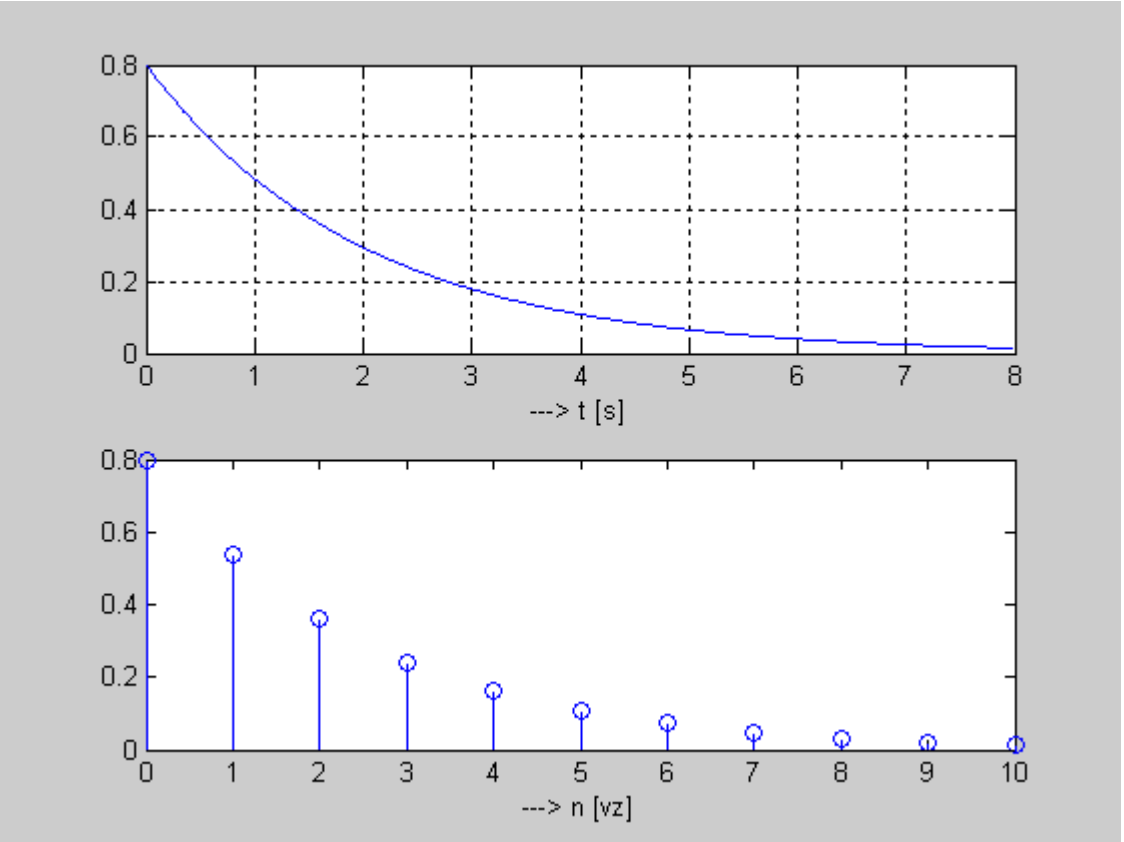
Generování číslicových signálů

- **Exponenciální signál**

$$x(t) = a \cdot \mathrm{e}^{bt}$$

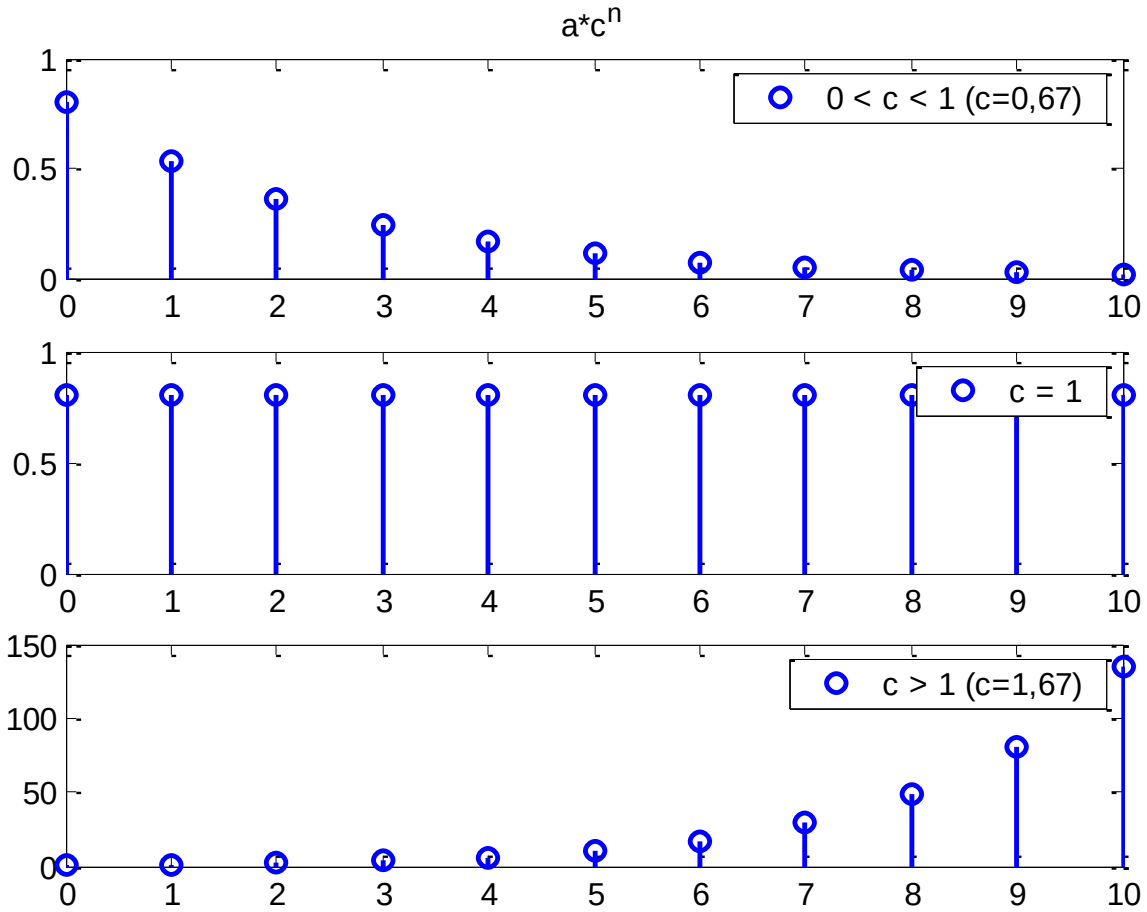
$$x[n] = a \cdot c^{n}$$

```
doba=8; fs=10;
a=0.8; tau=2;

t=0:1/fs:doba-1/fs;
x_t=a*exp(-t/tau);

pb=10;
n=0:pb;
c=exp(-doba/(pb*tau));
x_n=a*c.^n;
% c=0.6703
```



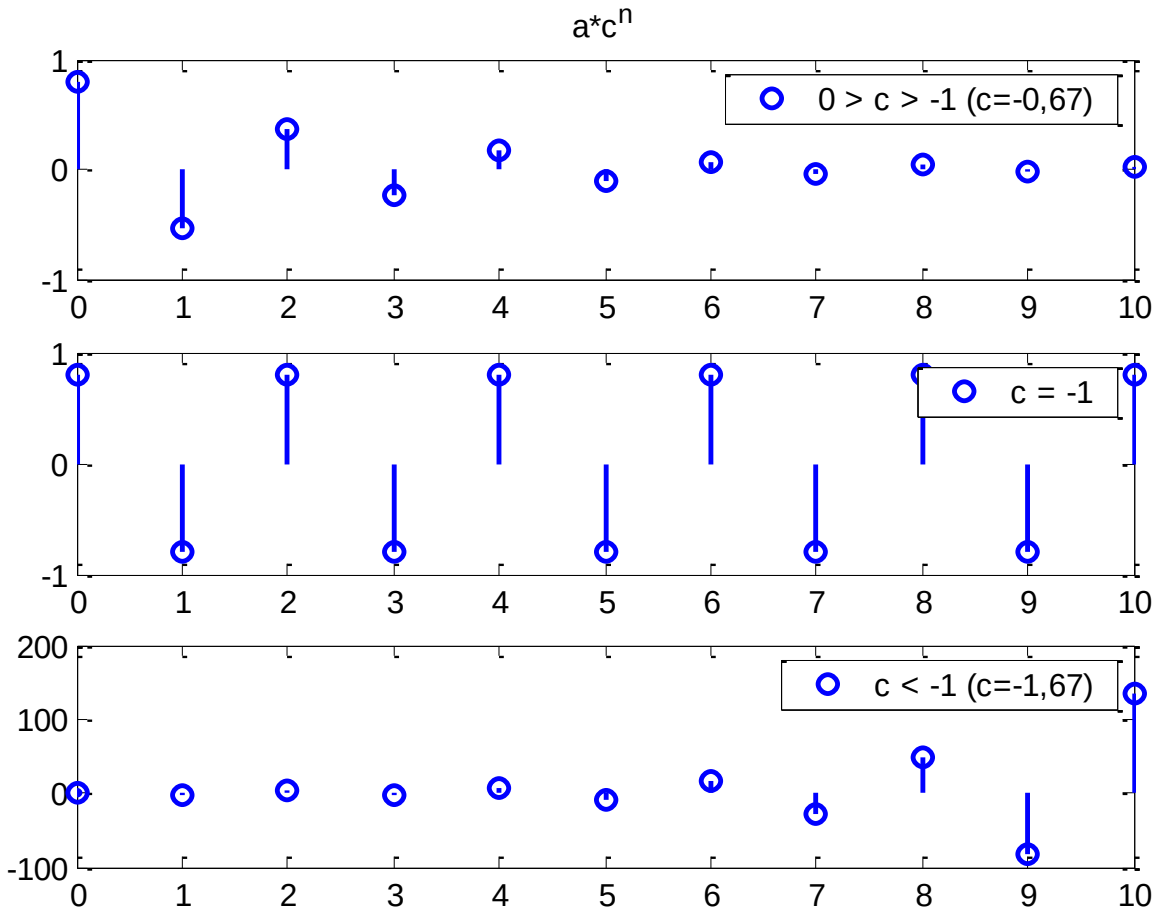
Generování číslicových signálů

- **Exponenciální signál**



Generování číslicových signálů

- **Exponenciální signál**



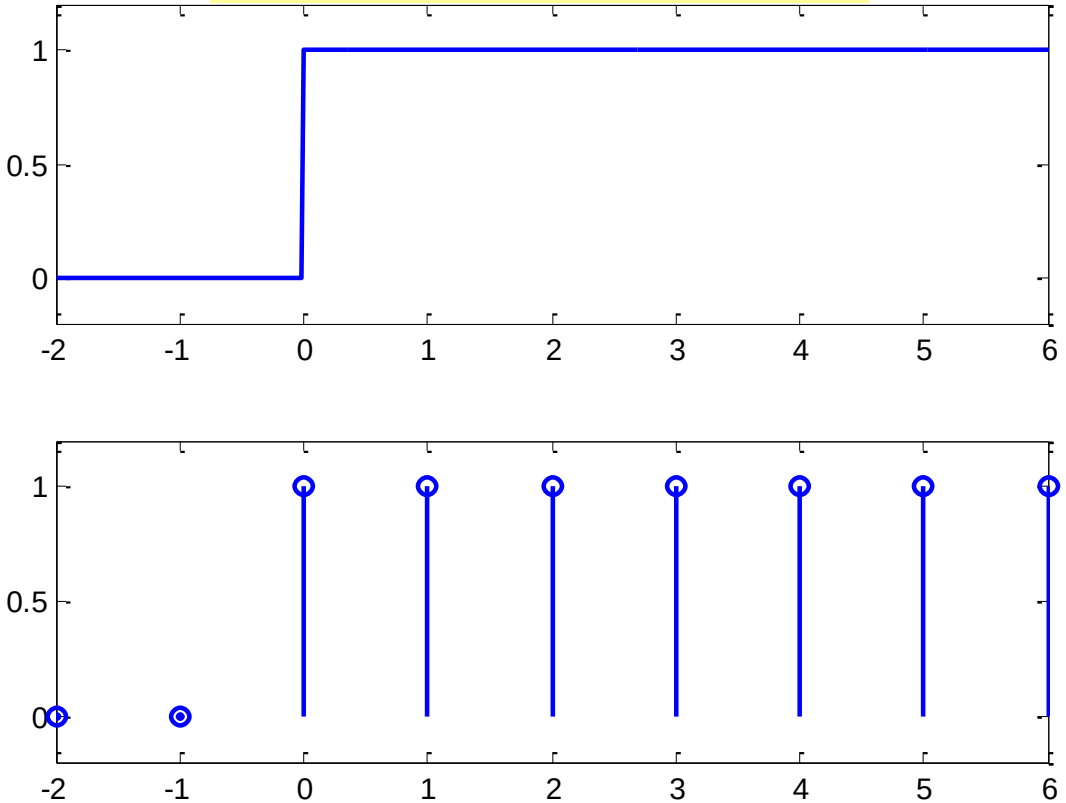
Generování číslicových signálů

- **Jednotkový skok**

$$u(t) = \begin{cases} 1, & t > 0 \\ 0, & t \leq 0 \end{cases}$$

$$u[n] = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

```
t=-2:0.01:6;
x_t = [t >= 0];
```



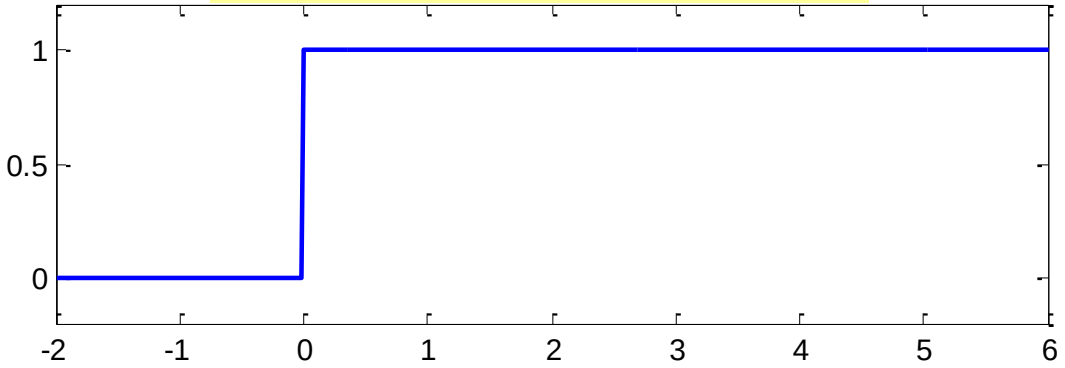
Generování číslicových signálů

- **Jednotkový skok**

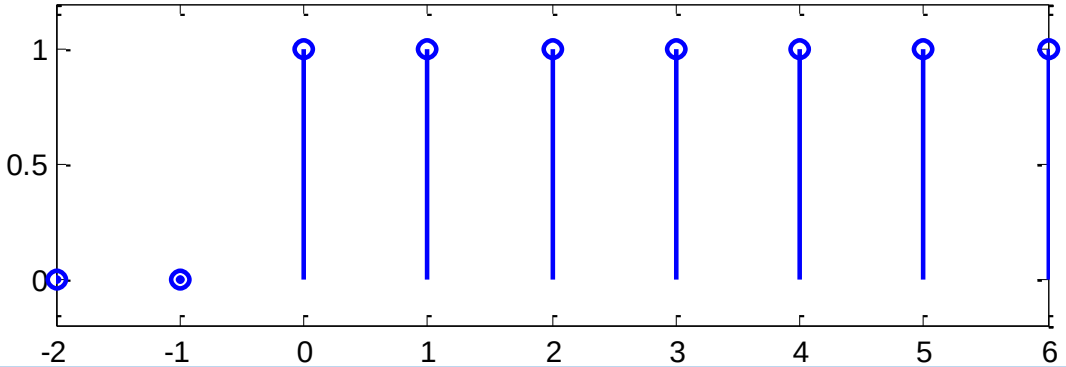
$$u(t) = \begin{cases} 1, & t > 0 \\ 0, & t \le 0 \end{cases}$$

$$u[n] = \begin{cases} 1, & n \ge 0 \\ 0, & n < 0 \end{cases}$$

```
t=-2:0.01:6;
x_t = [t >= 0];
```




```
n=-2:6;
x_n = [n >= 0];
```

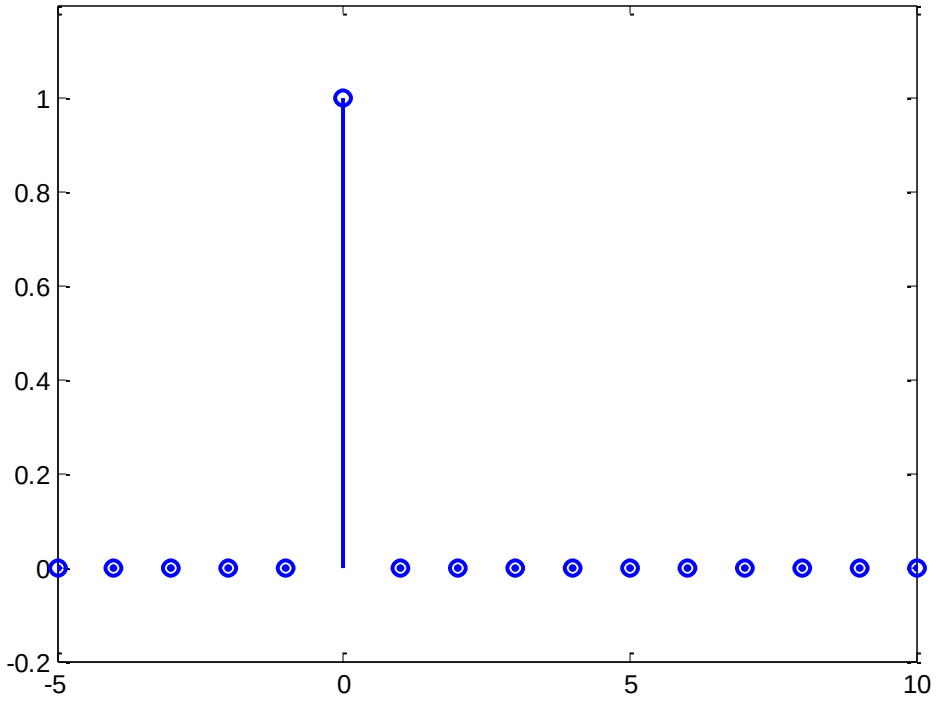


Generování číslicových signálů

- **Jednotkový impulz (Dirac)**

$$\delta[n] = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}$$

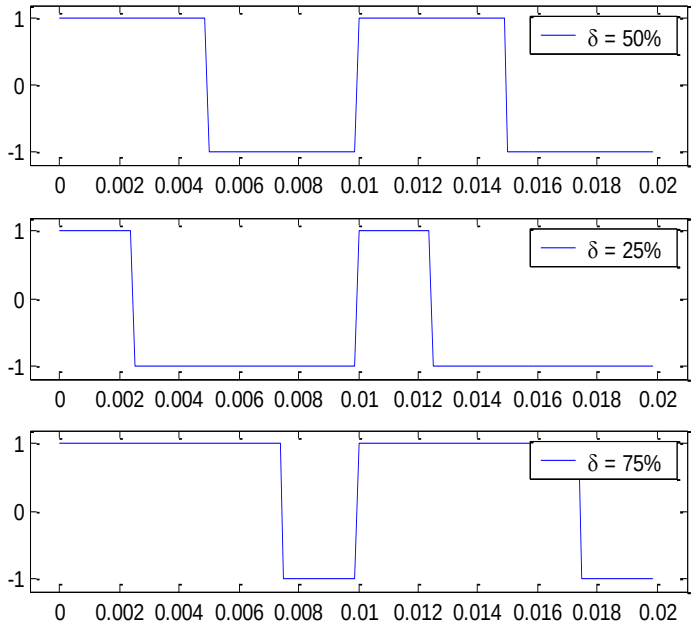
```
n=-5:10;
x_n = [n == 0];
```



Generování číslicových signálů

- **Obdélník**
  - **`square(om*t, delta)`**
    - stejné jako generování sin()
    - delta … % z periody, které má být kladné

```
>> o1_t = square(2*pi*f*t1);
>> o2_t = square(2*pi*f*t1,25);
>> o3_t = square(2*pi*f*t1,75);
```

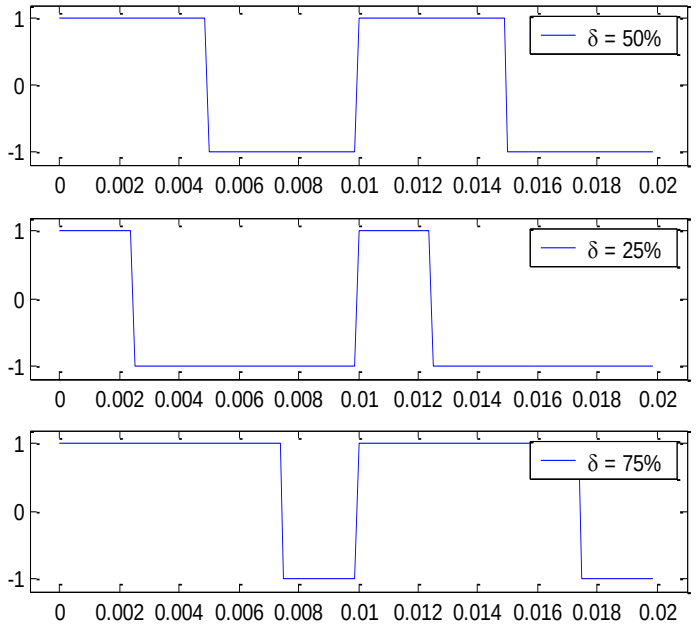


Generování číslicových signálů

- **Obdélník**
  - **`square(om*t, delta)`**
    - stejné jako generování sin()
    - delta … % z periody, které má být kladné

```
>> o1_t = square(2*pi*f*t1);
>> o2_t = square(2*pi*f*t1,25);
>> o3_t = square(2*pi*f*t1,75);
```

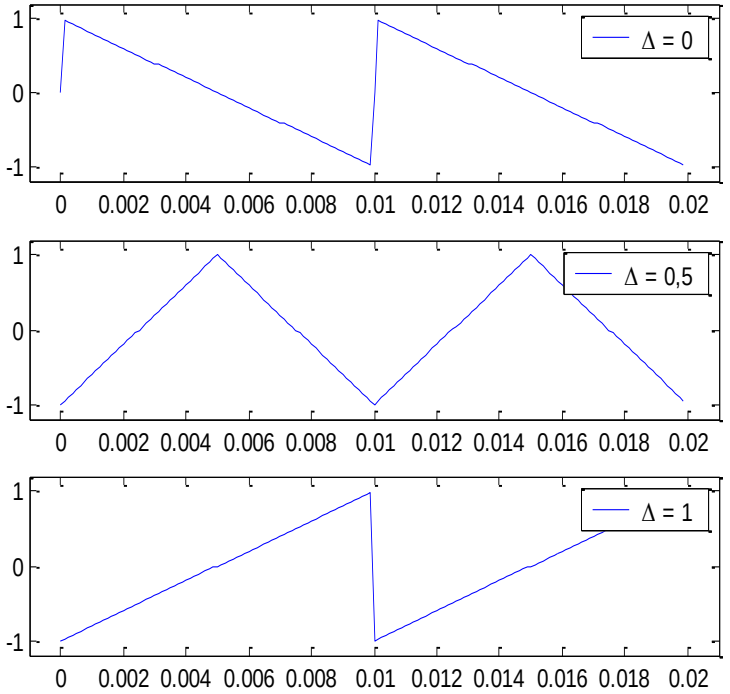
```
>> x_min =-.001; x_max =0.021;
>> y_min =-1.2;  y_max =1.2;
>> axis([x_min x_max y_min y_max])
```



Generování číslicových signálů

- **Pila**
  - **`sawtooth(om*t, DELTA)`**
    - stejné jako generování sin()
    - DELTA … maximum na intervalu 0..1

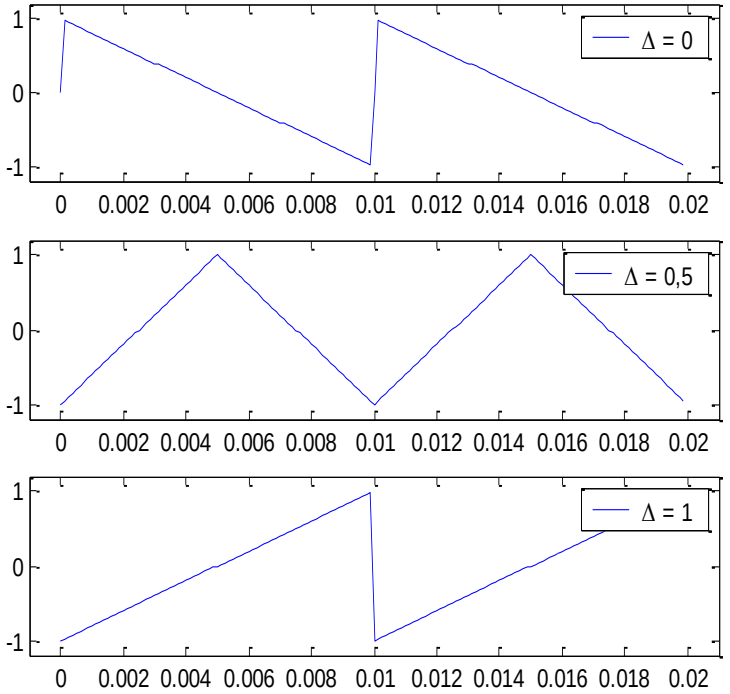
```
>> p1_t = sawtooth(2*pi*f*t1,0);
```



Generování číslicových signálů

- **Pila**
  - **`sawtooth(om*t, DELTA)`**
    - stejné jako generování sin()
    - DELTA … maximum na intervalu 0..1

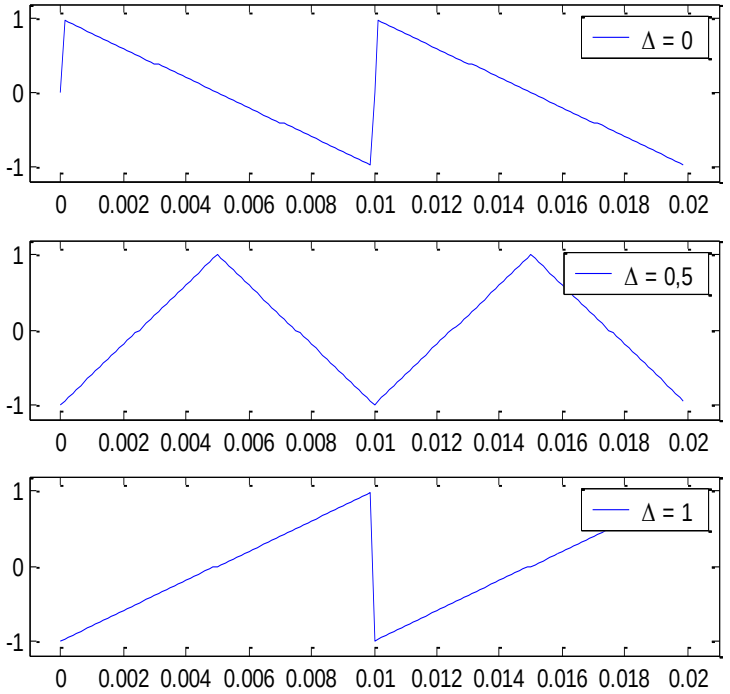
```
>> p1_t = sawtooth(2*pi*f*t1,0);

>> p2_t = sawtooth(2*pi*f*t1,0.5);
```



Generování číslicových signálů

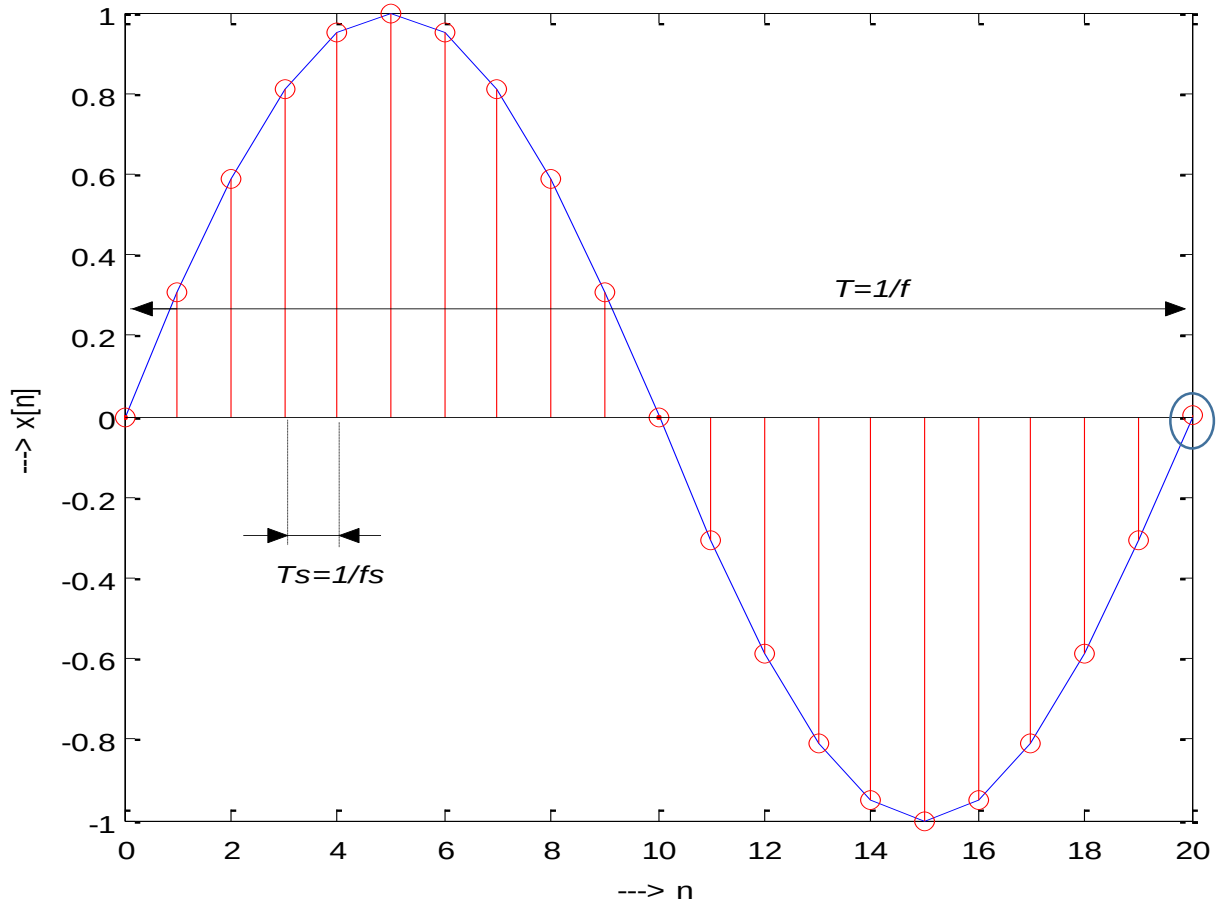
- **Pila**
  - **`sawtooth(om*t, DELTA)`**
    - stejné jako generování sin()
    - DELTA … maximum na intervalu 0..1

```
>> p1_t = sawtooth(2*pi*f*t1,0);

>> p2_t = sawtooth(2*pi*f*t1,0.5);

>> p3_t = sawtooth(2*pi*f*t1,1);
```



Generování číslicových signálů

- **Analogově číslicový převod**



Generování číslicových signálů

- **Analogově číslicový převod**

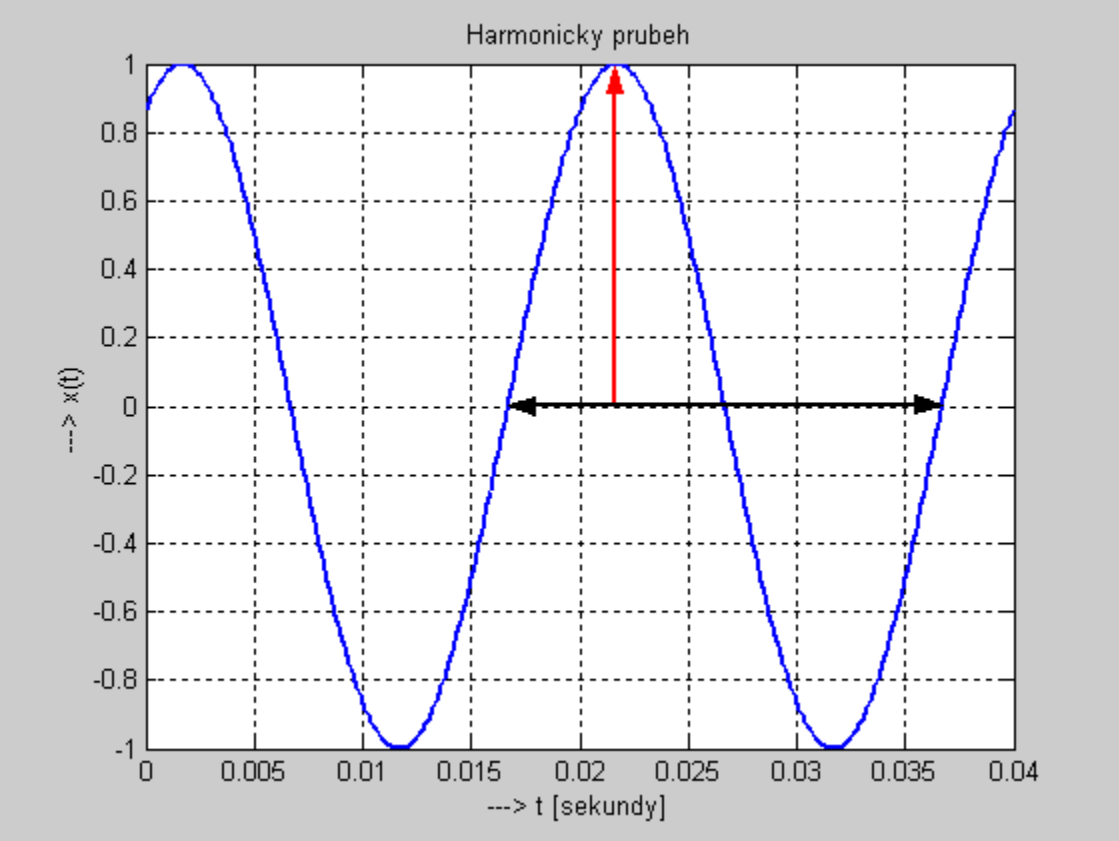
$$x(t) = X_m \sin(2\pi f_0 t + \phi)$$

Amplituda

Fáze [rad]

Čas [s]

Frekvence [Hz]



Generování číslicových signálů

- **Analogově číslicový převod**

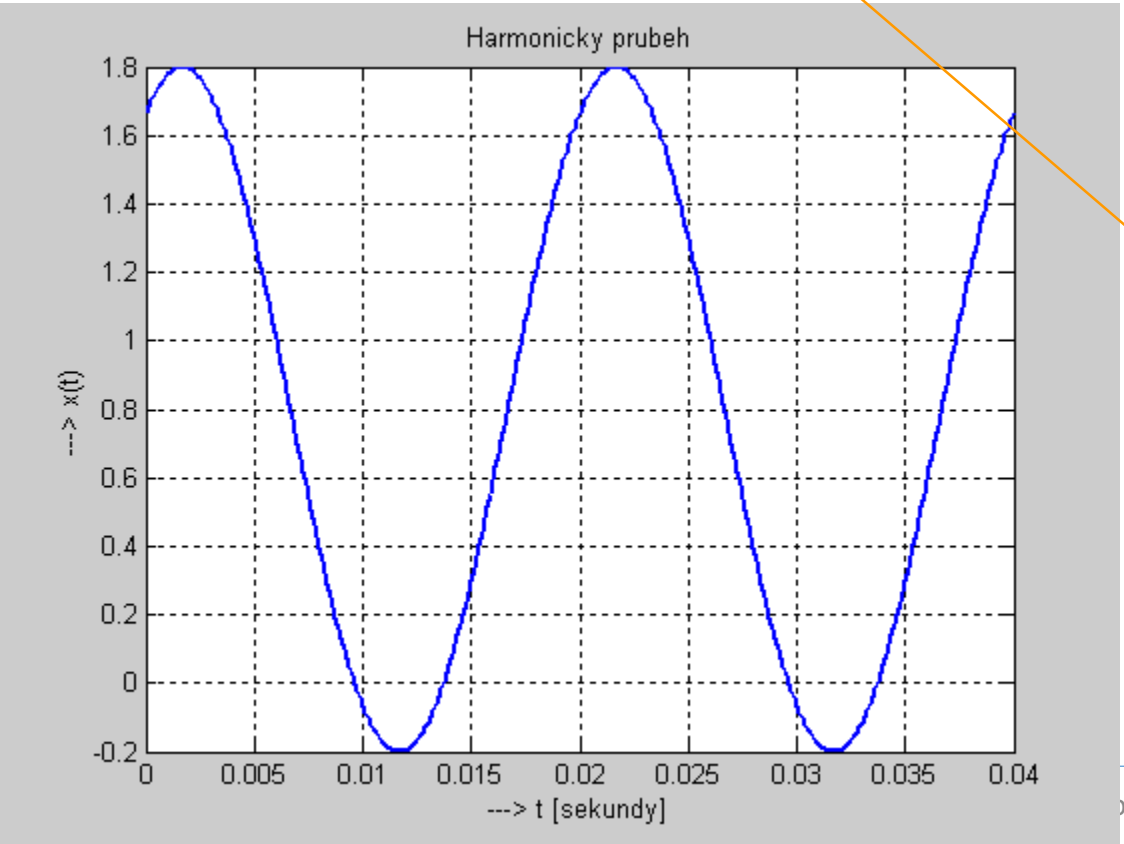
$$x(t) = X_0 + X_m \sin(2\pi f_0 t + \phi)$$

?

?

?

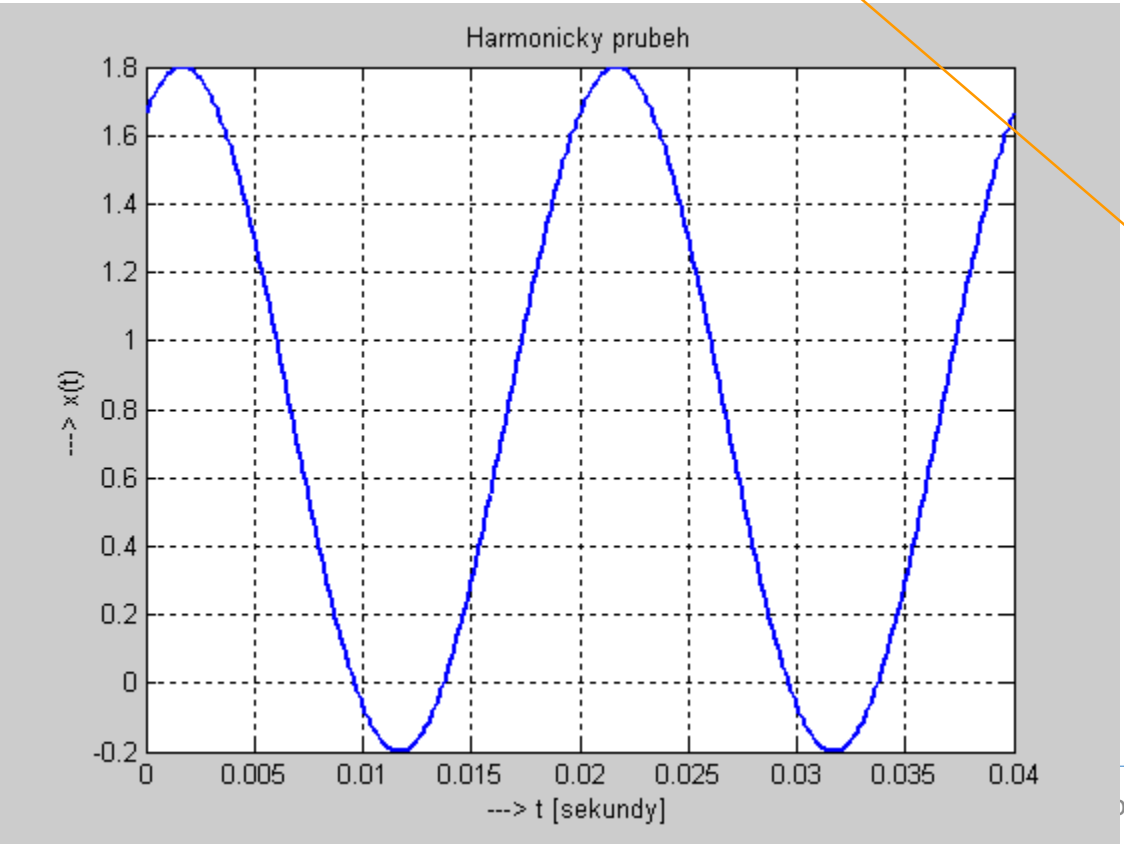
?



Generování číslicových signálů

- **Analogově číslicový převod**

$$x(t) = X_0 + X_m \sin(2\pi f_0 t + \phi)$$



0,95

50

0,8

1

Generování číslicových signálů

- **Generování harmonických signálů v MATLABu**

  - Generujme 20 ms harmonického signálu o frekvenci 440 Hz a vzorkovací frekvenci 8 kHz

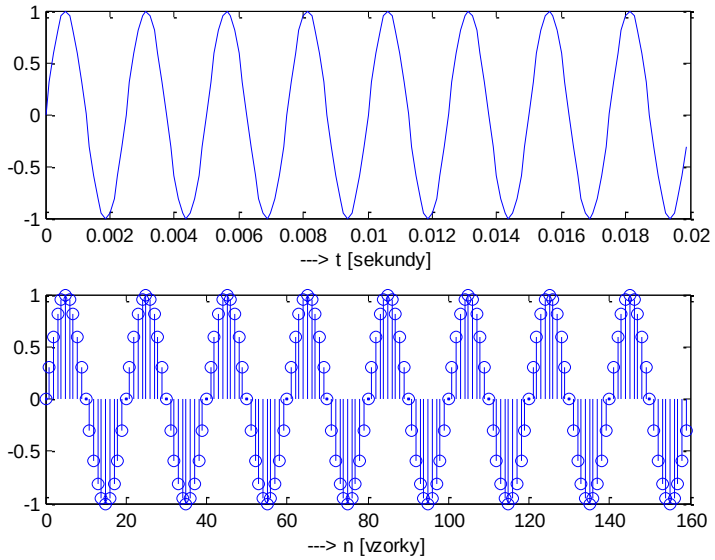
  - Generujme 4 periody signálu o frekvenci 440 Hz a vzorkovací frekvenci 8 kHz

  - Generujme 25 vzorků signálu o frekvenci 440 Hz a vzorkovací frekvenci 8 kHz

Generování číslicových signálů

- **Generování harmonických signálů v MATLABu**
  - Generujme 20 ms harmonického signálu o frekvenci 440 Hz a vzorkovací frekvenci 8 kHz

```
>> f=440; fs=8000; duration =.02;

>> t1   = 0:1/fs: duration -1/fs;
>> x1_t = sin(2*pi*f*t1);

>> n1   = 0:1:fs* duration -1;
>> x1_n = sin(2*pi*f*n1/fs);
```

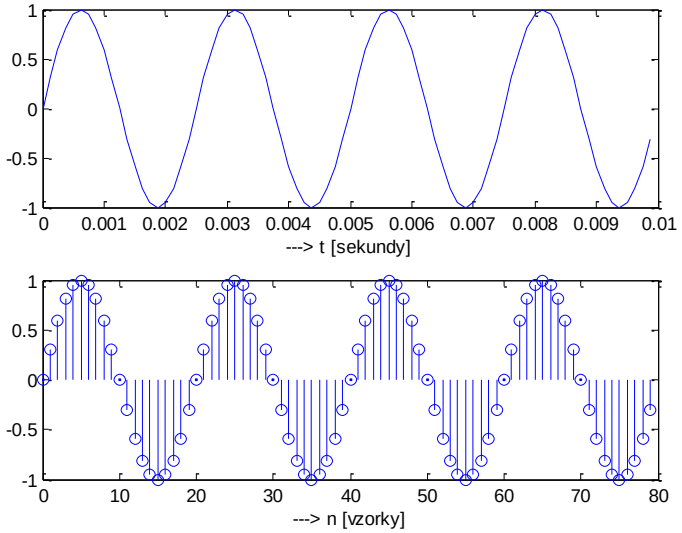


Generování číslicových signálů

- **Generování harmonických signálů v MATLABu**
  - Generujme 4 periody signálu o frekvenci 440 Hz a vzorkovací frekvenci 8 kHz

```
>> f=440; fs=8000; np=4;
```

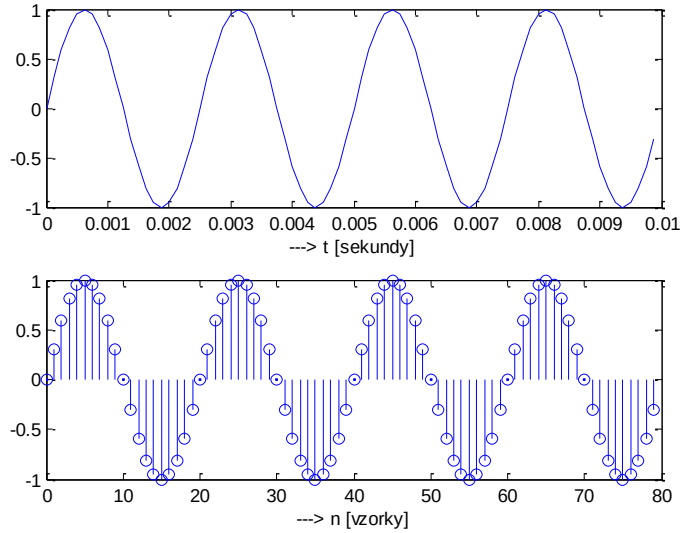
```
>> t2   = 0:1/fs:np/f-1/fs;
>> x2_t = sin(2*pi*f*t2);

>> n2   = 0:1:np*fs/f-1;
>> x2_n = sin(2*pi*f*n2/fs);
```



Generování číslicových signálů

- **Generování harmonických signálů v MATLABu**
  - Generujme 25 vzorků signálu o frekvenci 440 Hz a vzorkovací frekvenci 8 kHz

```
>> f=440; fs=8000; N=25;


>> t3   = 0:1/fs:N/fs-1/fs;
>> x3_t = sin(2*pi*f*t3);

>> n3   = 0:1:N-1;
>> x3_n = sin(2*pi*f*n3/fs);;
```

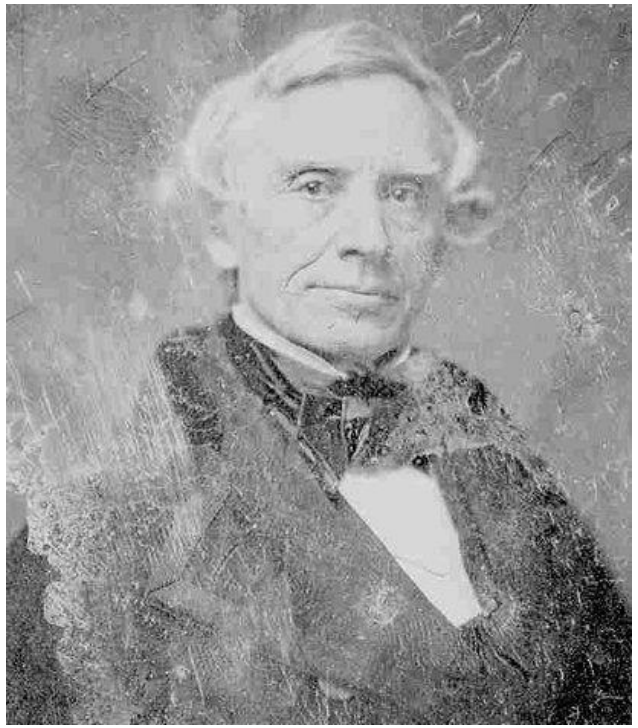


Generování číslicových signálů

- Morse Code Generator

```
>> morse('sos')
```

. . ./- - -/. . .//. - - ./- - -/- -/- - -/- . - ./



A	.-	M	--	Y	-.--	6	-....
B	-...	N	-.	Z	--..	7	--...
C	-.-.	O	---	Ä	.-.-	8	---..
D	-..	P	.--.	Ö	---.	9	----.
E	.	Q	--.-	Ü	..--	.	.-.-.-
F	..-.	R	.-.	Ch	----	,	--..--
G	--.	S	...	0	-----	?	..--..
H	....	T	-	1	.----	!	··--·
I	..	U	..-	2	..---	:	---...
J	.---	V	...-	3	...--	“	.-..-.
K	-.-	W	.--	4	....-	‘	.----.
L	.-..	X	-..-	5	.....	=	-...-

http://www.shaman.cz/sifrovani/morseova-abeceda.htm

Generování číslicových signálů

a) Oznamovací tón:
F = [425 0 425 0]; [Hz]
T = [0.33 0.33 0.66 0.66]; [s]

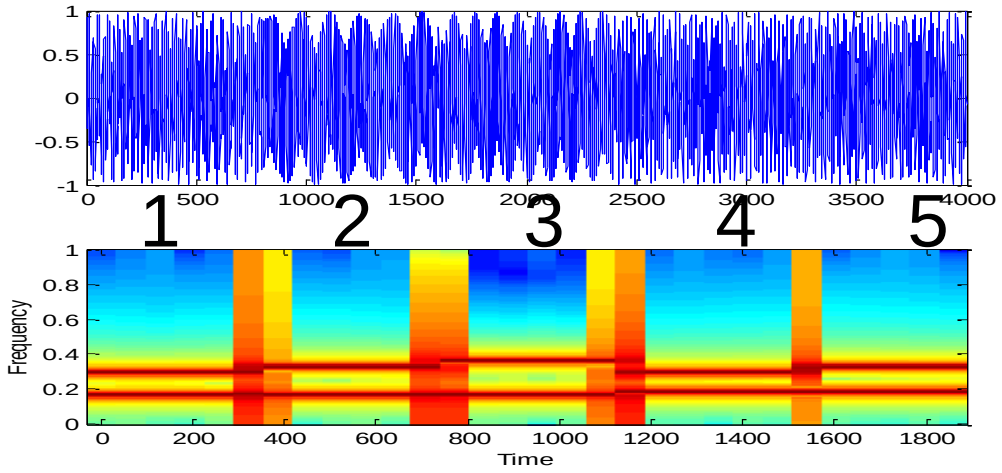
b) Vyzváněcí tón:
F = [425 0]; [Hz]
T = [1 3]; [s]

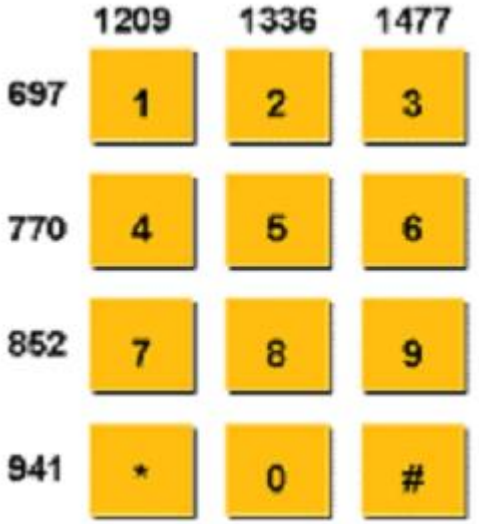
c) Odkazovací tón:
F = [950 0 1400 0 1800 0]; [Hz]
T = [0.33 0.03 0.33 0.03 0.33 1.25]; [s]

d) Obsazovací tón:
F = [425 0]; [Hz]
T = [0.33 0.42]; [s]

Generování číslicových signálů

- **Tónová volba (DTMF - Dual Tone Multi-Frequency)**
  - Frekvence nejsou:
    - násobkem jiné frekvence
    - rozdílem či součtem frekvencí





Generování číslicových signálů

- **DTMF volba**

```
>> signal = DTMFdial([6 0 3]);
```

```
function tones = DTMFdial(numbers)
Fs = 8000;
t = (0:799)/Fs;     % generating time axis
f = [ 697   697   697   770   770   770   852   852   852   941   941   941;
      1209  1336  1477  1209  1336  1477  1209  1336  1477  1336  1209  1477];
tones=[];
for i=1:length(numbers),
      if(numbers(i)==0), numbers(i)=10; end;
      tone1 = 0.5*sin(2*pi*f(1,numbers(i))*t);
      tone2 = 0.5*sin(2*pi*f(2,numbers(i))*t);
      tones =[tones; tone1 + tone2];
end;
tones = tones'; tones = tones(:);
```

Hudební stupnice

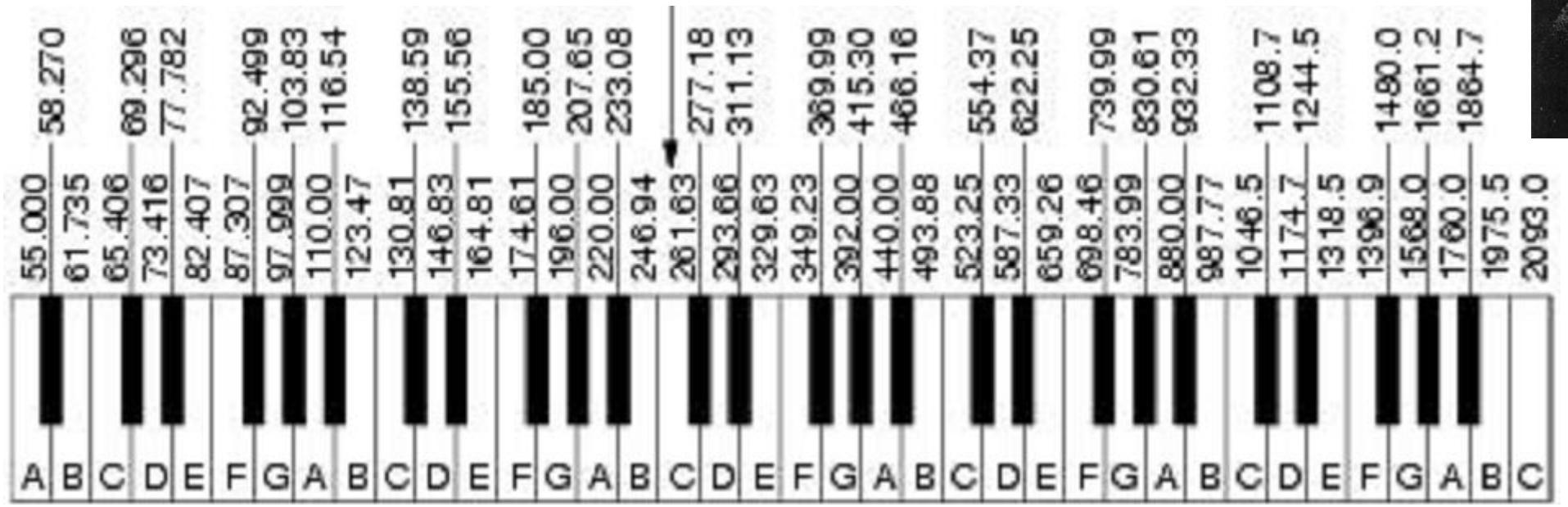
- **Ladění**
  - **Čisté nebo přirozené**
    - Jako **čistá** nebo také **přirozená** ladění se označují ladění využívající pouze tóny, jejichž frekvence jsou ve vzájemných **poměrech** vyjádřitelných **celými čísly**.
  - **Temperované**
    - **Temperované** ladění je v hudbě způsob ladění, při kterém jsou některé čisté intervaly záměrně rozladěny, aby se docílilo přesnějšího naladění intervalů jiných.

Hudební stupnice

- **Temperované ladění**

>> logspace(log10(261.63),log10(2*261.63),13)

$$f = 440 \cdot 2^{\frac{k-49}{12}} \ [\mathrm{Hz}]$$



>>261.63; 277.19; 293.67; 311.13; 329.63; 349.23; 370.00; 392.00; 415.31; 440.01; 466.17; 493.89; 523.26

Hudební stupnice

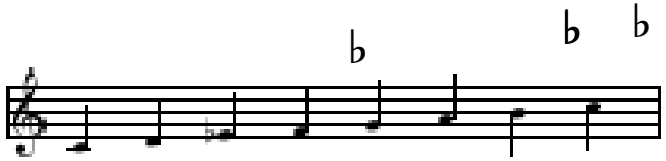
- **Stupnice**

Chromatická [0 1 2 3 4 5 6 7 8 9 10 11 12]
1 1 1 1 1 1 1 1 1 1 1 1
oktáva představuje 12 tónů chromatické stupnice

diatonická (sedm tónů - pět tónů a dva půltóny)

- **durová** [0 2 4 5 7 9 11 12]
2 2 1 2 2 2 1

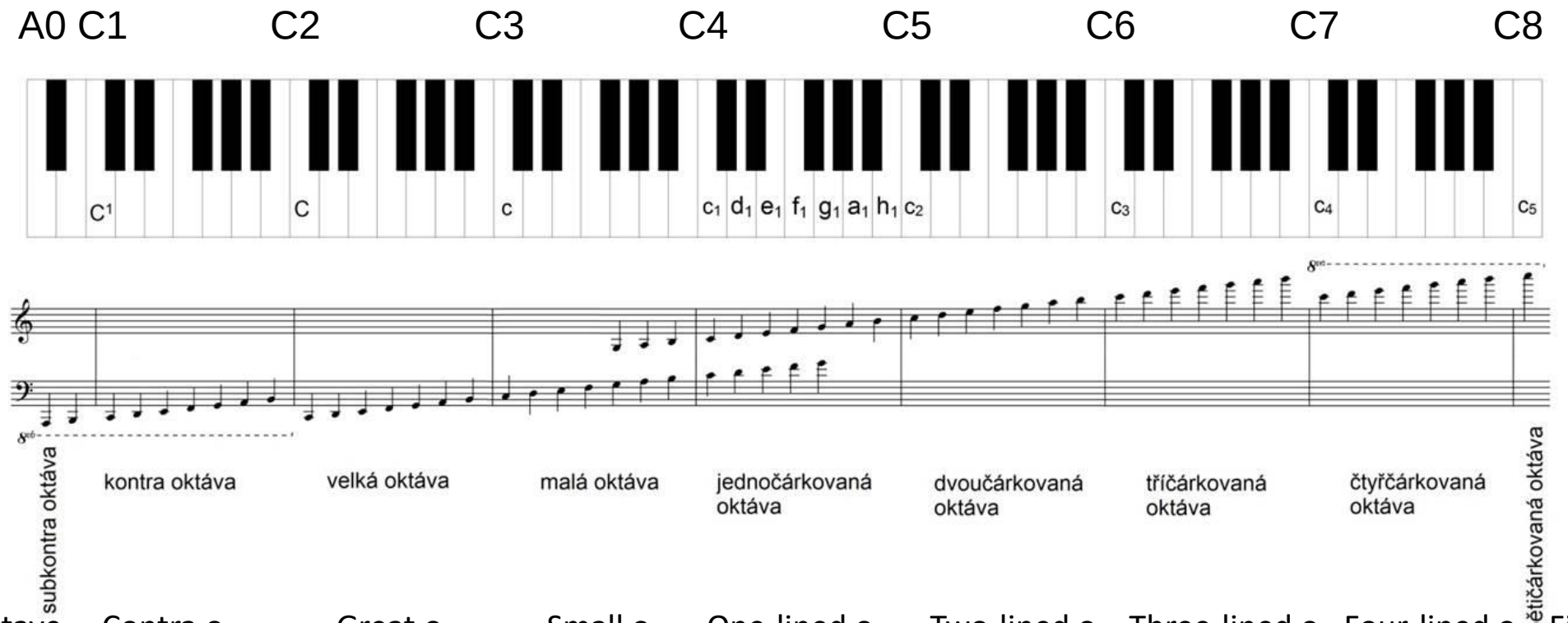
- **molová** [0 2 3 5 7 8 10 12]
2 1 2 2 1 2 2



c1 (C4) … c2 (C5)

Hudební stupnice

kontinentální: c1 d1 e1 f1 g1 a1 h1 c2
anglosaské: C4 D4 E4 F4 G4 A4 B4 C5



Sub-contra octave | Contra o. | Great o. | Small o. | One-lined o. | Two-lined o. | Three-lined o. | Four-lined o. | Five-lined octave

Hudební stupnice

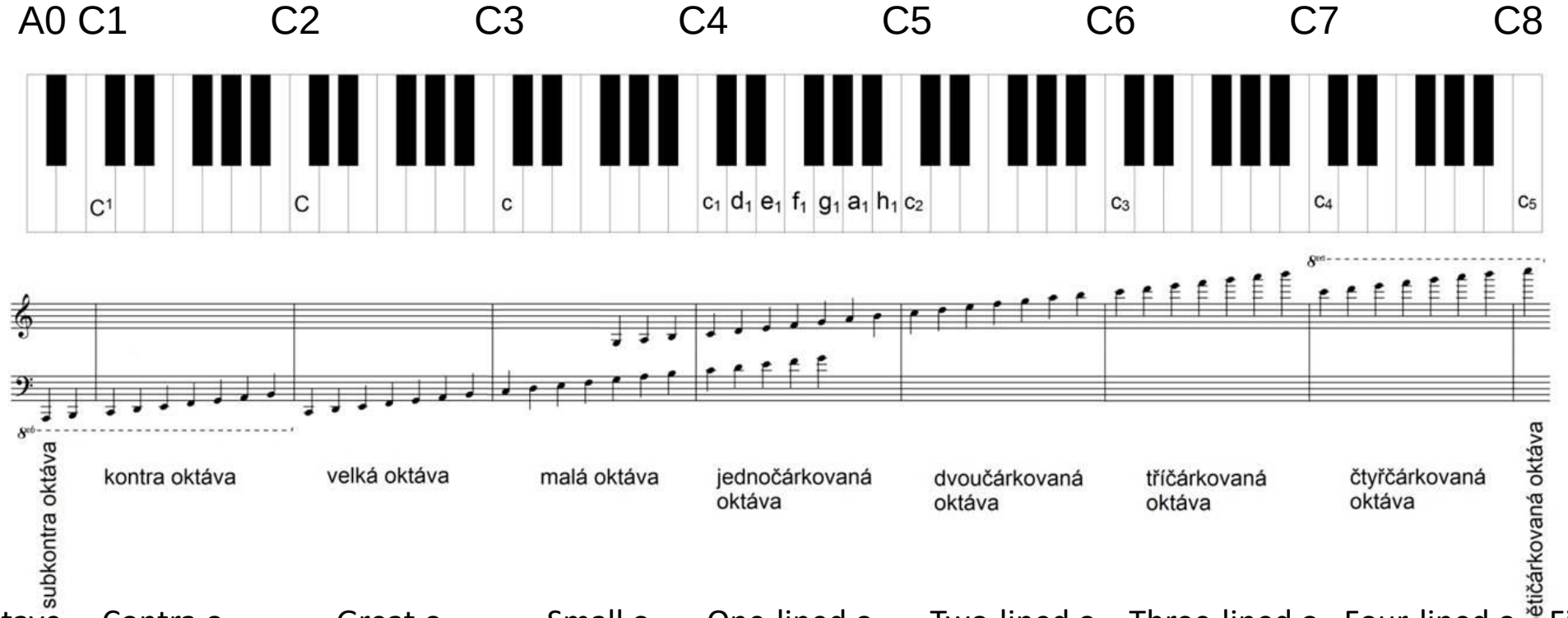
- **MIDI**

Převod mezi pořadím MIDI not (m=0 až 127) na frekvenci f [Hz] a naopak

m = 69 + 12*log2(f/440); [MIDI, Hz]

f = (440/32)*2.^((m-9)/12); [Hz, MIDI]

- MIDI umožňuje číslovat výšku tónu pořadím 0,1,...,127.
- přenos v rozsahu 8.18 Hz až 12544 Hz
- rozsah tónů na klavíru: od A0 = **27.5 Hz**, MIDI=**21** až do c5 = **4186 Hz**, MIDI=**108**



Sub-contra octave Contra o. Great o. Small o. One-lined o. Two-lined o. Three-lined o. Four-lined o. Five-lined octave

Hudební stupnice

- **A4 (a1) Pitch Standard** (Concert Pitch)

A4 = 440Hz jediný oficiální standard

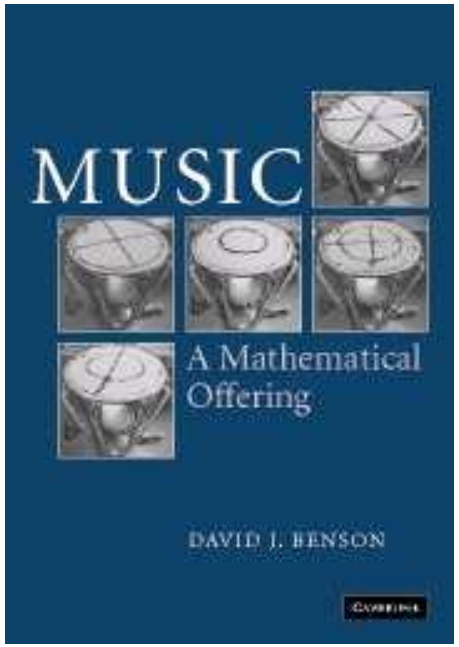
významné orchestry však ladí i na 441, 442, 443, …

Tuner - gStrings Free

https://cs.wikipedia.org/wiki/Komorní_tón
https://en.wikipedia.org/wiki/Concert_pitch

https://logosfoundation.org/kursus/music_math.pdf

Music: a Mathematical Offering



Hudební stupnice

```
function tone = note(key,duration)
% input: key  ... key number of the desired tone
% c1=40; d1=42; e1=44; f1=45;
% g1=47; a1=49; h1=51; c2=52
%        duration  ... signal duration [s]
% output:tone      ... generated signal [Hz]

fs  = 8000;
f   = 440 * 2 ^ ((key-49)/12);
t   = 0:1/fs:duration-1/fs;
tone= sin(2*pi*f*t);
```

Hudební stupnice

```
close all,clear
x        = [];      % initialize variable
fs       = 8000;    % also change in the function
keys     = [40 42 44 45 47 49 51 52];
durations= 0.5*ones(1,length(keys));

for k = 1:length(keys)
    tone = note(keys(k),durations(k));
    x    = [x tone];
end
soundsc(x,fs)       % listen to the musical scale
```

Tabulková (wavetable) syntéza

pravděpodobně nejstarší technika pro vytváření zvuků pomocí počítačů

PPG Wave Series: Implementace syntézy tabulek využívala pole obsahující 64 ukazatelů na jednotlivé jednocyklové vlny.

Waldorf Microwave: Nová generace PPG.

Roland D-50 and Roland MT-32/variants: ”*Linear Arithmetic*” syntezátor (v podstatě wavetable syntezátor se 2 tabulkami).

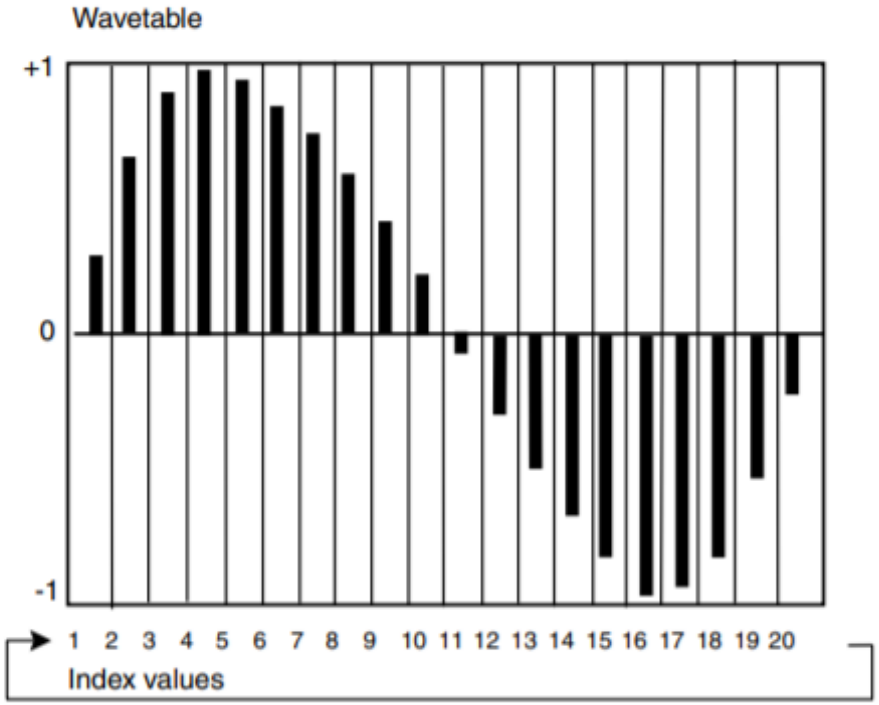
KorgWavestation: ”*Vector synthesis*” přechod mezi tabulkami pomocí 2d mřížky.

Tabulková (wavetable) syntéza

- Tabulková syntéza simuluje nástroj pomocí vzorků vyjmutých ze skutečného nástroje.

- **Lookup table**

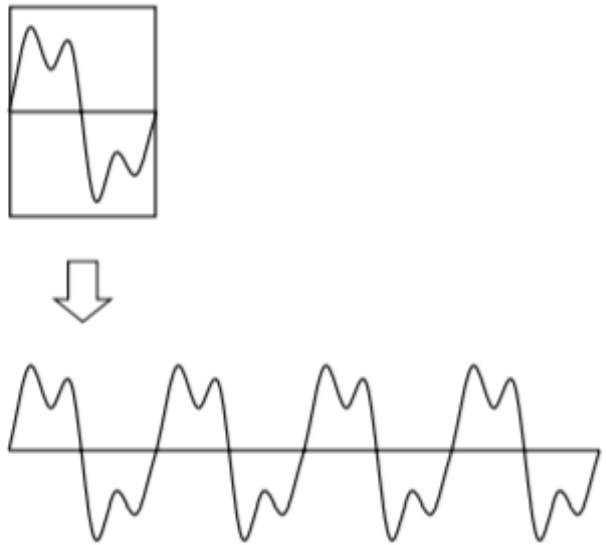
- Různé mechanismy
  - Single wavecycle
  - Looping
  - Multiple wavecycle
  - Crossfading
  - ADSR envelope
  - Pitch Shifting



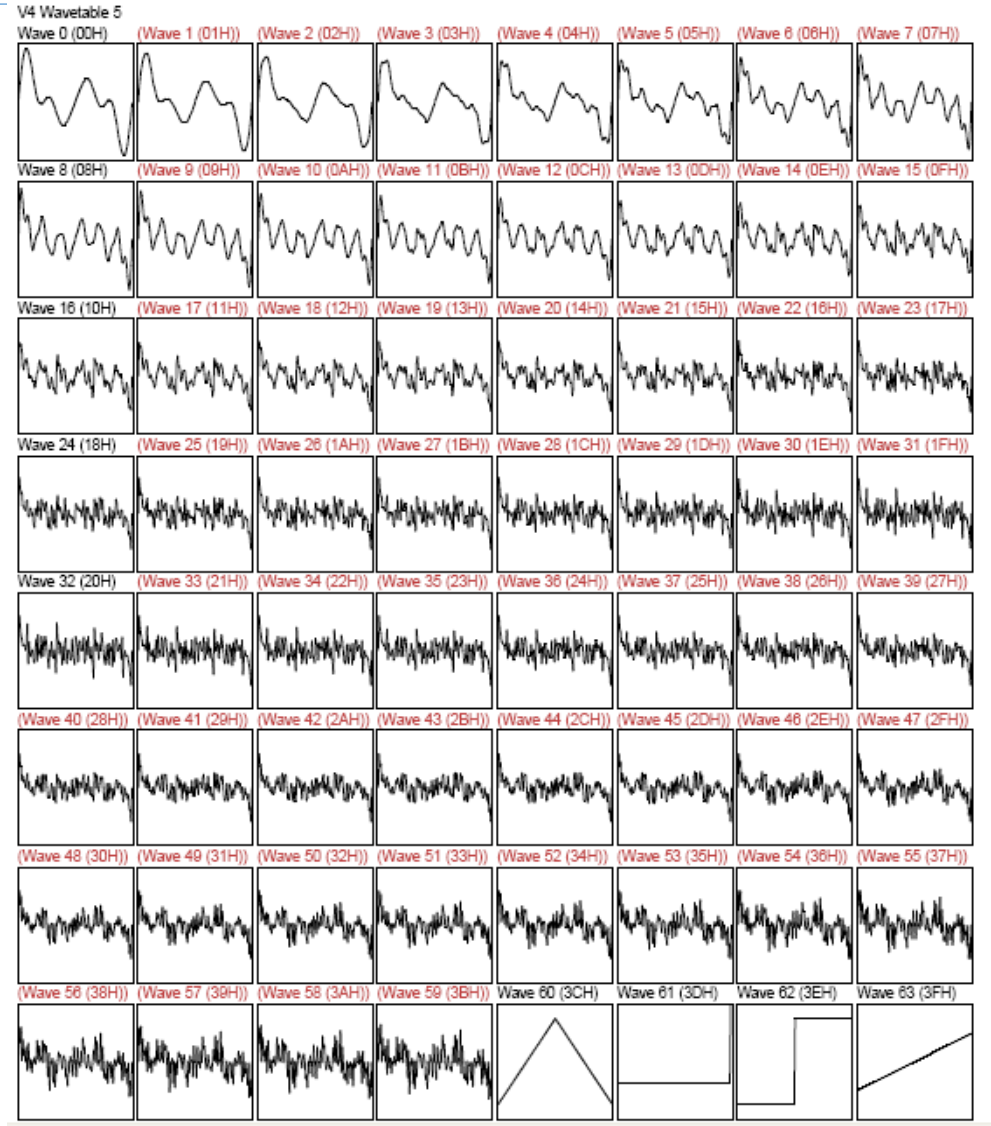
Vzorky pro daný tvar vlny jsou uloženy v tabulce, zvuk je produkován opakovaným přehráváním vzorků.

Tabulková (wavetable) syntéza

- **Single Wavecycle**
  - Simple
  - Repetitive sound
  - Sine, Triangle, Sawtooth



Single wavecycle přístup k tabulkové syntéze



Tabulková (wavetable) syntéza

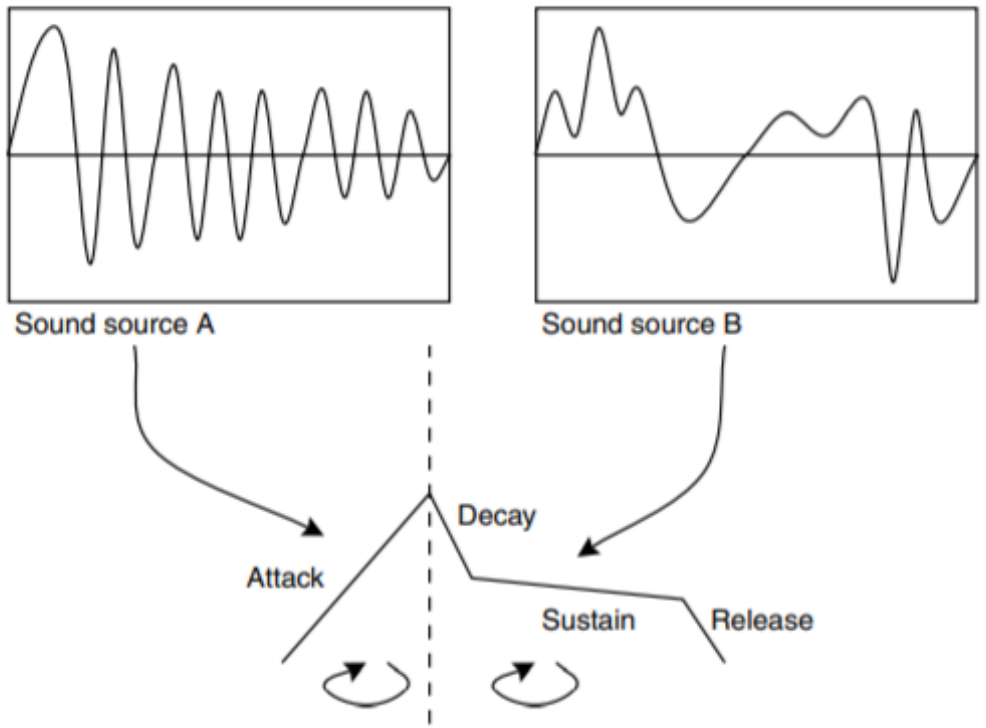
- **Looping**
  - Pečlivý výběr period:
    - velké změny amplitudy (vytvářejí nežádoucí zvukové efekty)
    - fázové změny (slyšitelné kliknutí)

E.g. 1

E.g. 2

Tabulková (wavetable) syntéza

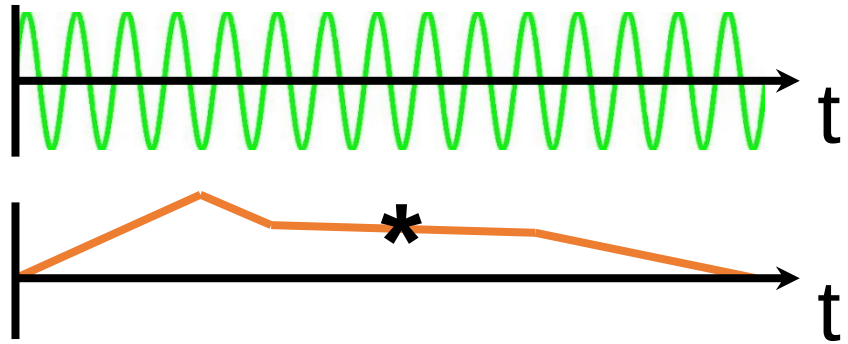
- **Multiple Wavecycle**
  - Více zdrojů

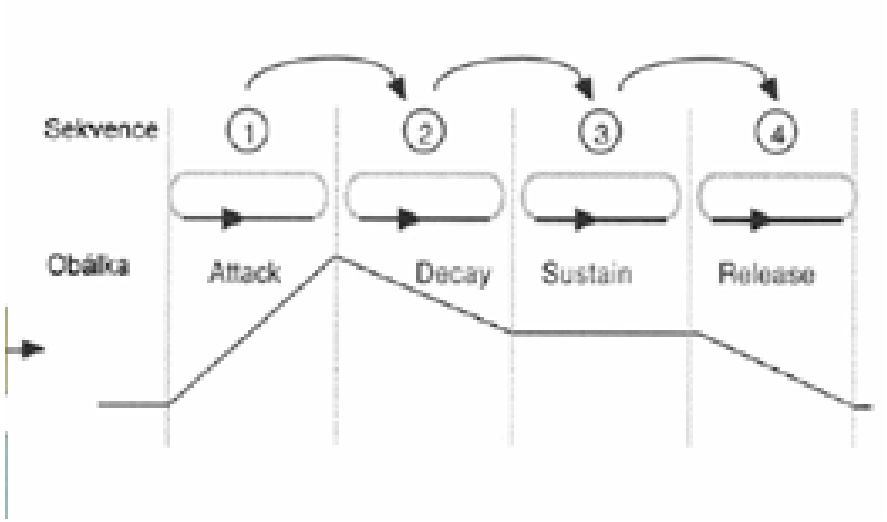


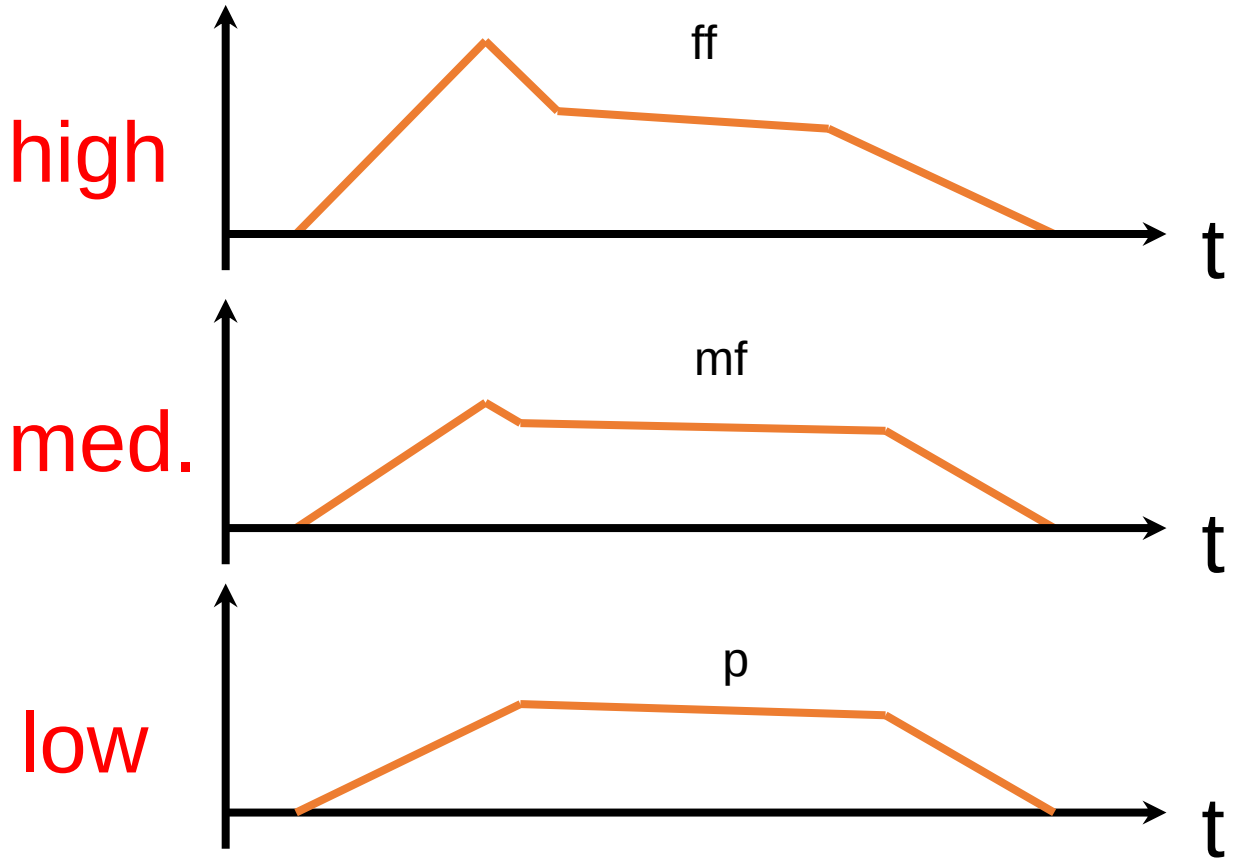
Multiple wavecycle používá různé zdroje pro různé části zvuku

Tabulková (wavetable) syntéza

- **ADSR obálka**

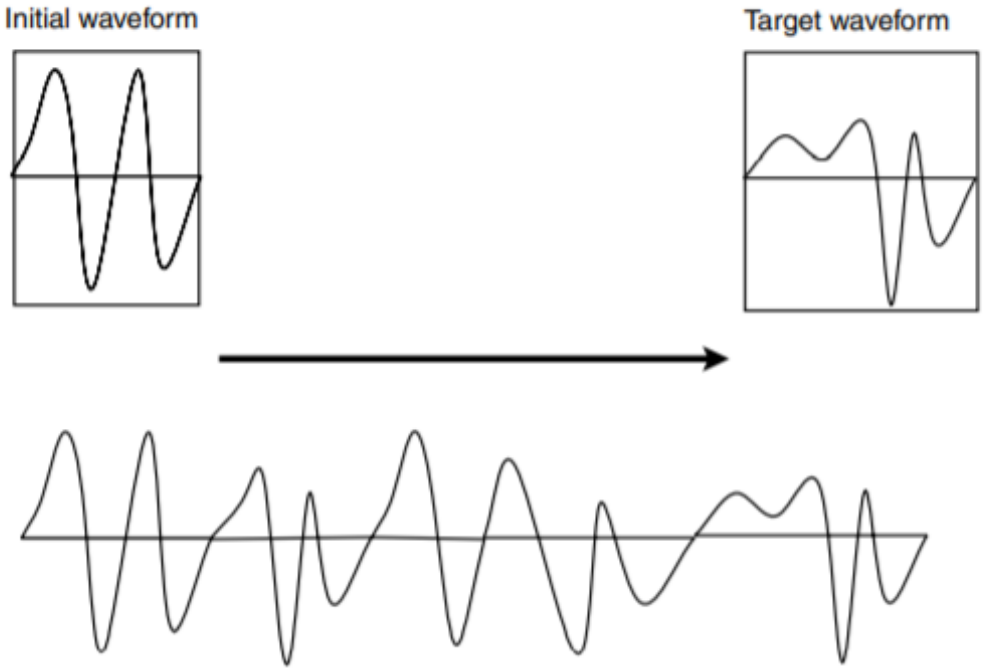




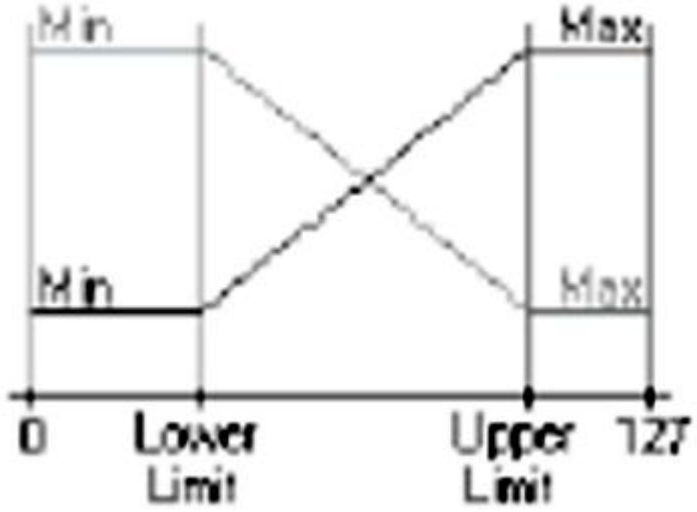


Tabulková (wavetable) syntéza

- **Crossfading**
  - prolínání



Crossfading pracuje tak, že postupně mění výstupy vzorků z jedné tabulky na výstupy vzorků další tabulky.

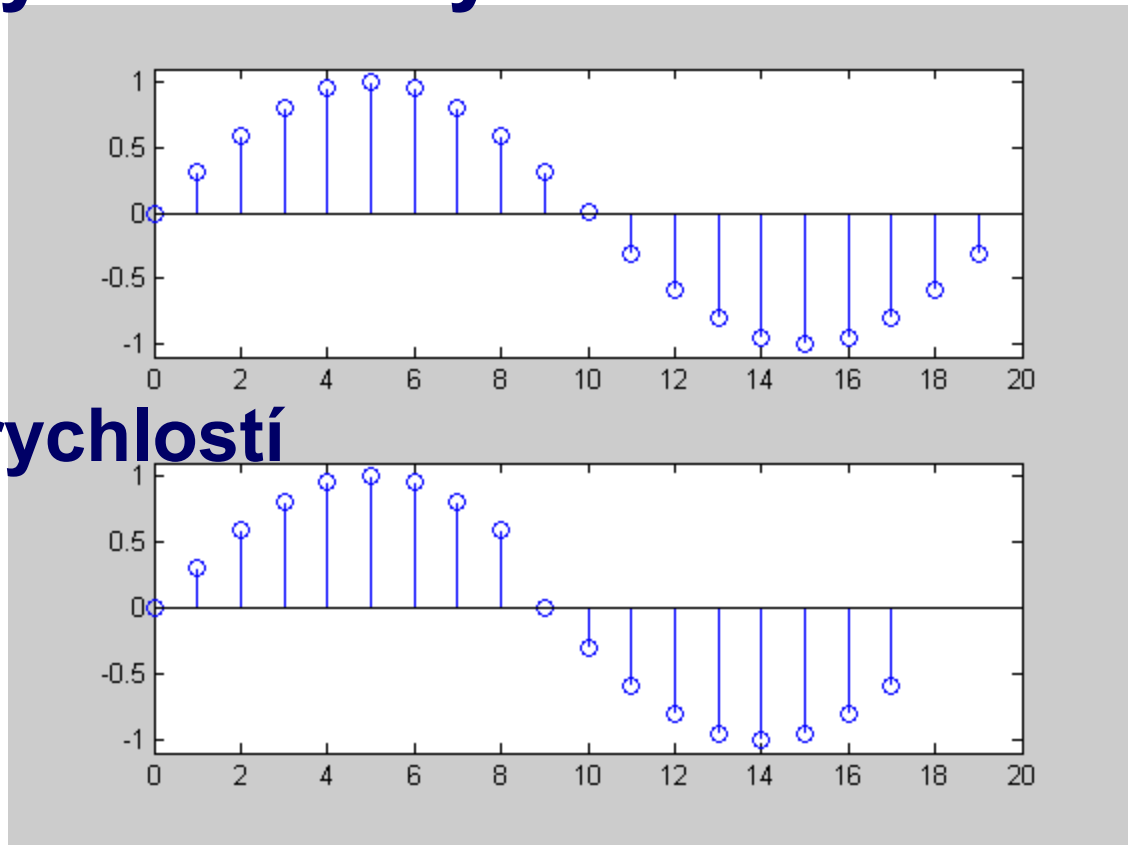


- Linear crossfading

Tabulková syntéza v Matlabu

- **Uložení jedné periody do tabulky**

```
fs=8000;
f0=400;
N0=fs/f0;
n0=0:N0-1;
P=sin(2*pi*f0/fs*n0);
```



- **a čtení tabulky různou rychlostí**

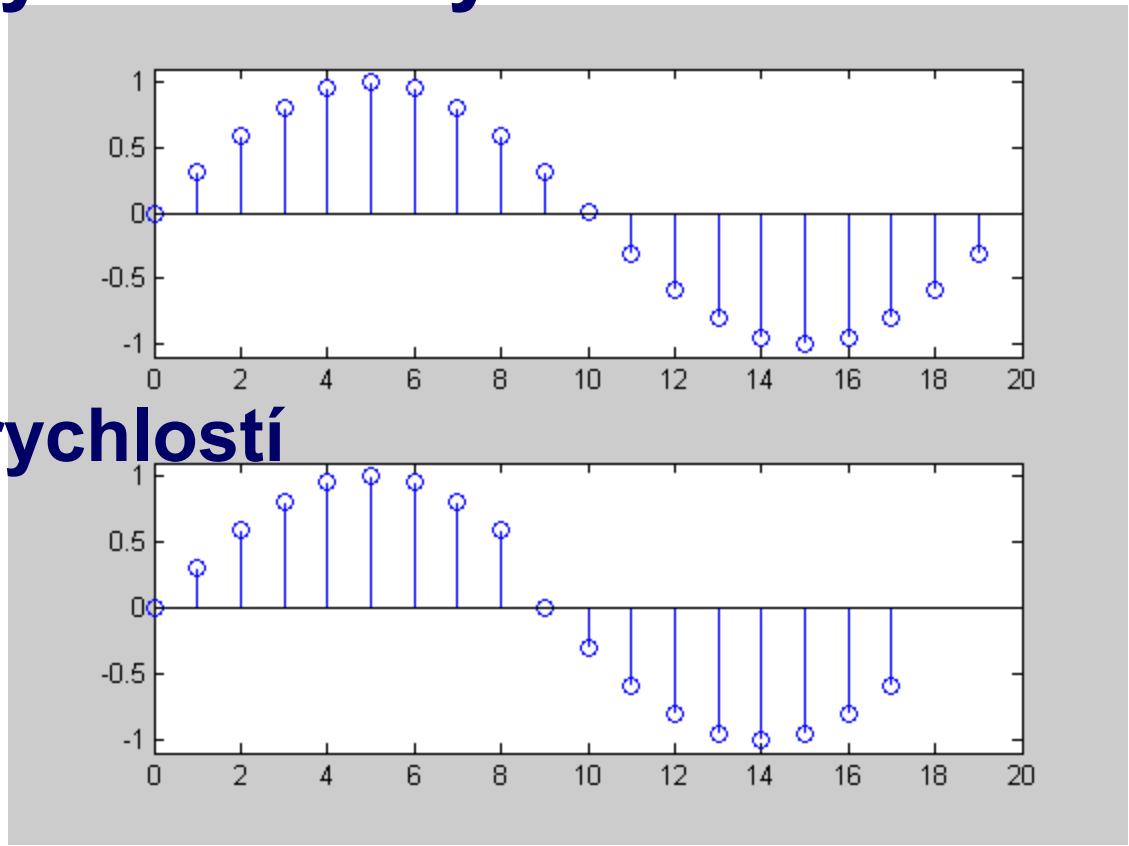
Tabulková syntéza v Matlabu

- **Uložení jedné periody do tabulky**

```
fs=8000;
f0=400;
N0=fs/f0;
n0=0:N0-1;
P=sin(2*pi*f0/fs*n0);
```

- **a čtení tabulky různou rychlostí**

```
f1=f0*2^(1/12);
```



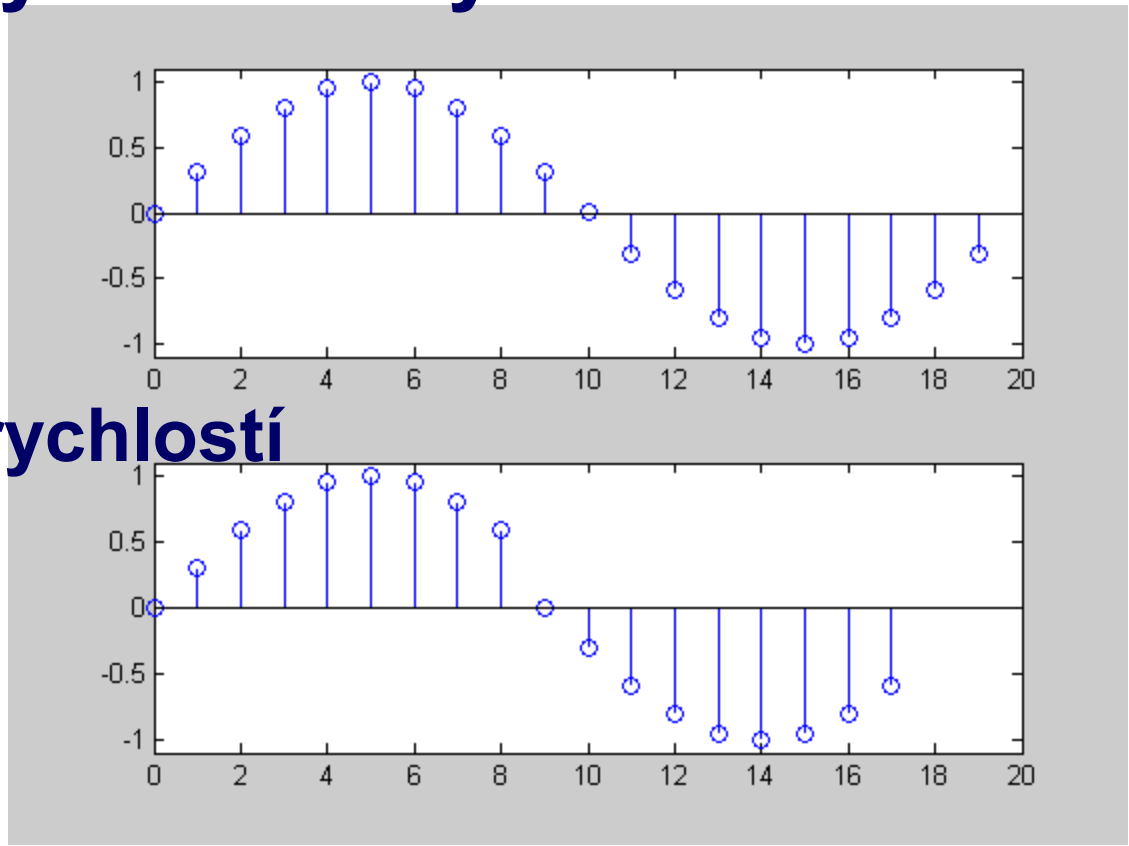
Tabulková syntéza v Matlabu

- **Uložení jedné periody do tabulky**

```
fs=8000;
f0=400;
N0=fs/f0;
n0=0:N0-1;
P=sin(2*pi*f0/fs*n0);
```

- **a čtení tabulky různou rychlostí**

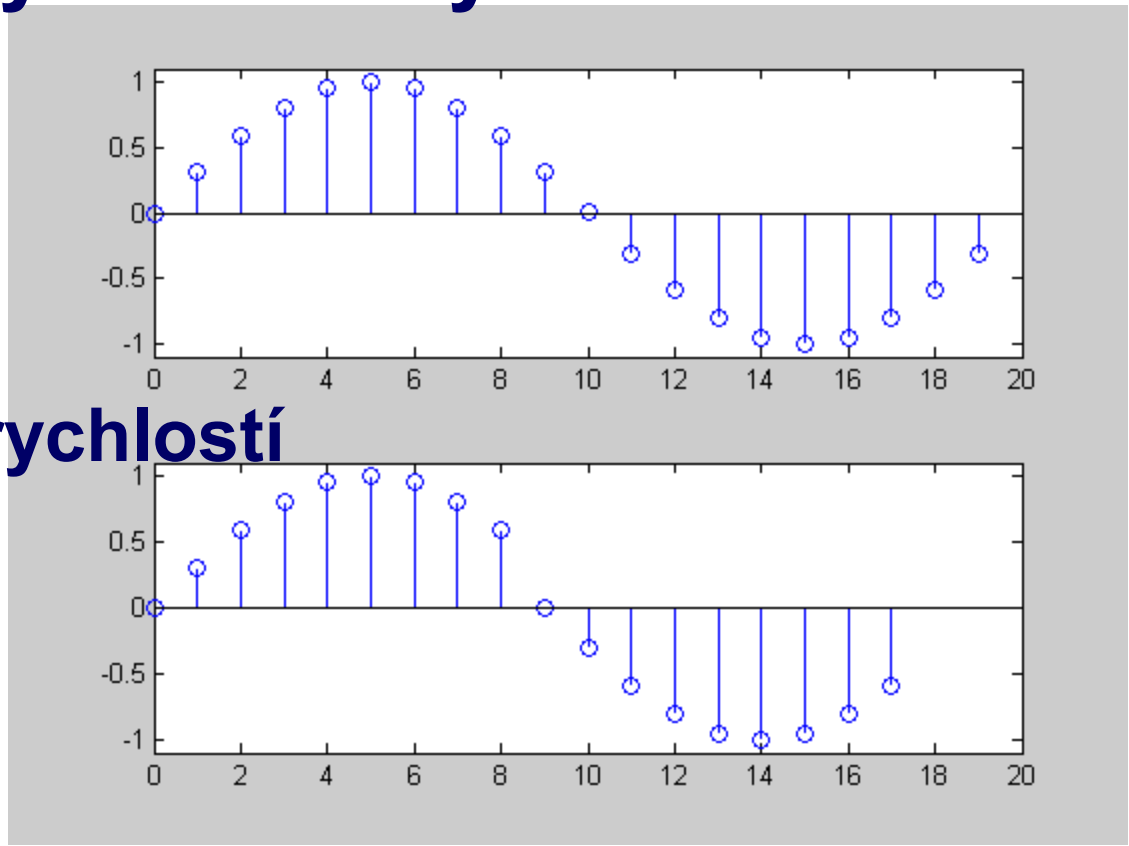
```
f1=f0*2^(1/12);
N1=fs/f1;
n1=0:N1-1;
```



Tabulková syntéza v Matlabu

- **Uložení jedné periody do tabulky**

```
fs=8000;
f0=400;
N0=fs/f0;
n0=0:N0-1;
P=sin(2*pi*f0/fs*n0);
```



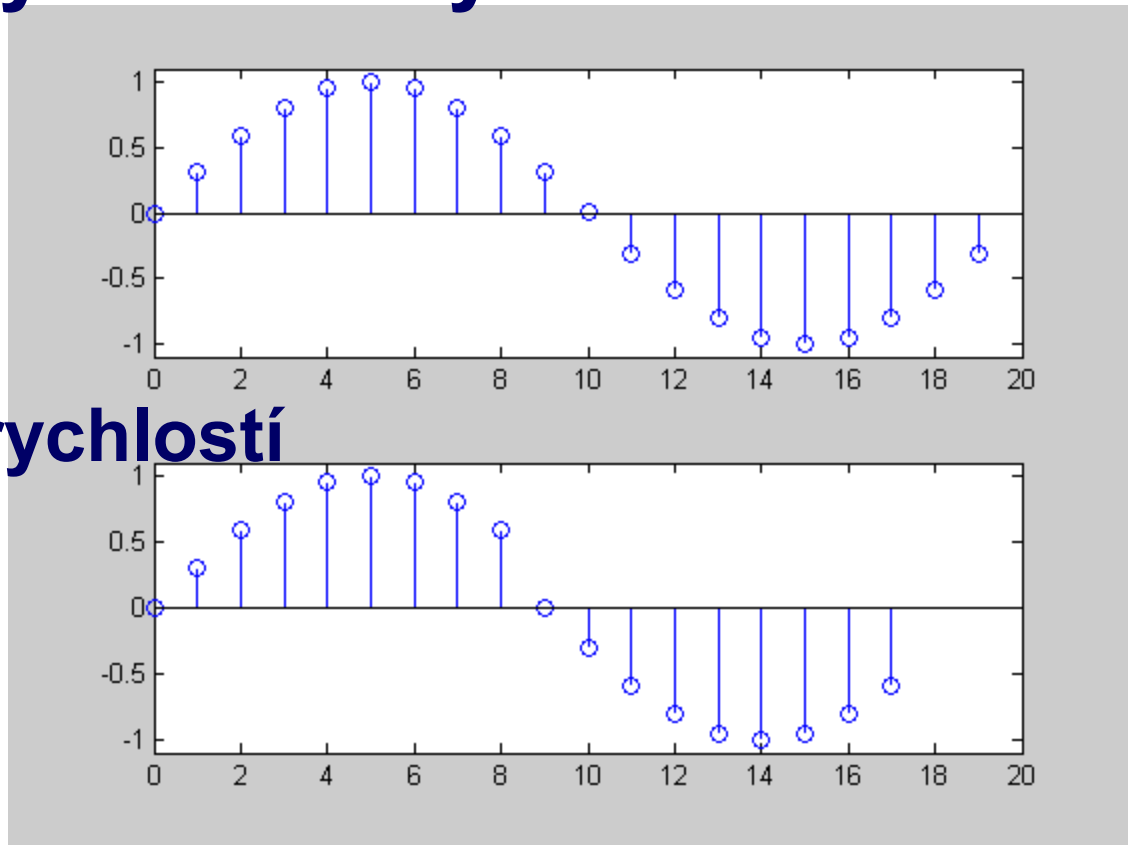
- **a čtení tabulky různou rychlostí**

```
delta=f1*length(P)/fs;
% delta=f1/f0; % delta=T0/T1;
```

Tabulková syntéza v Matlabu

- **Uložení jedné periody do tabulky**

```
fs=8000;
f0=400;
N0=fs/f0;
n0=0:N0-1;
P=sin(2*pi*f0/fs*n0);
```



- **a čtení tabulky různou rychlostí**

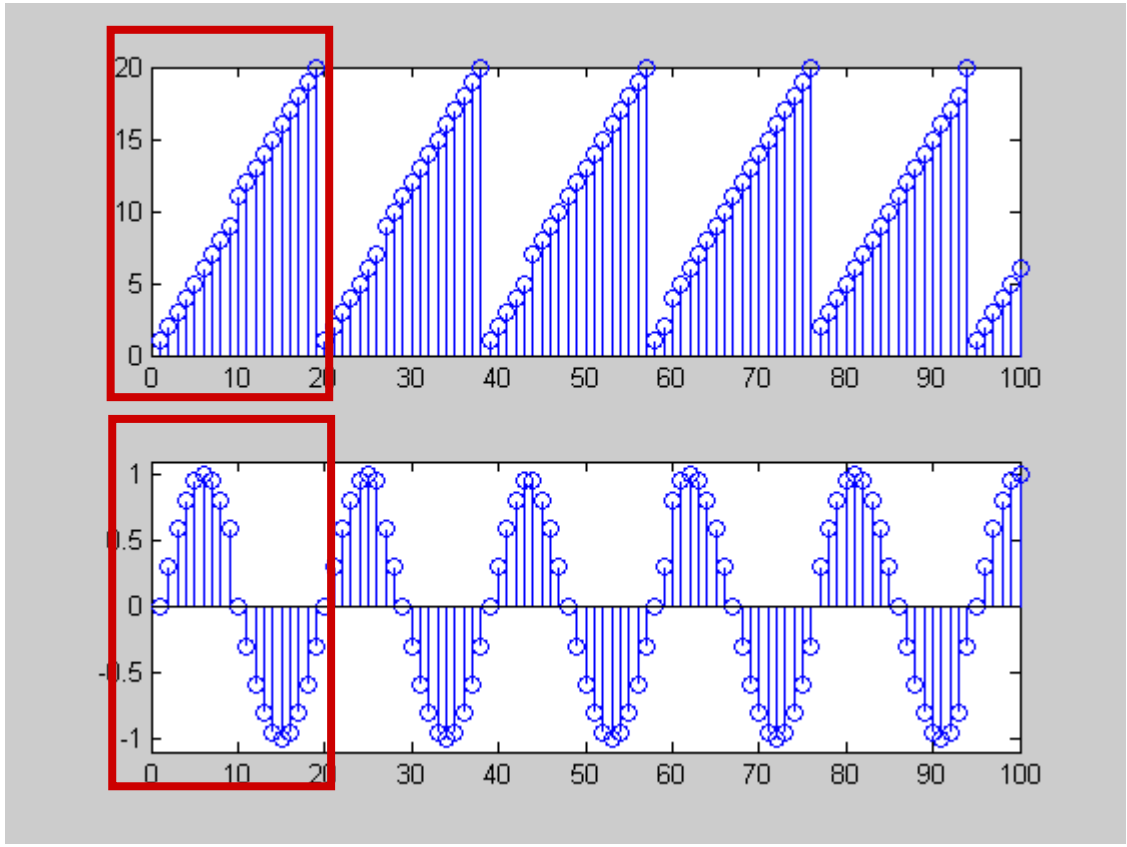
```
f1=f0*2^(1/12);
N1=fs/f1;
n1=0:N1-1;
delta=f1*length(P)/fs;
% delta=f1/f0; % delta=T0/T1;
ind = round(0:delta:delta*(N1-1))+1;
stem(n1,P(ind))
```

Tabulková syntéza v Matlabu

- **Wavetable oscilátor** (s pilou)

```
% transformace tabulky s jednou periodou f0
% na signál libovolné délky ("duration„) a frefrekvence f1

duration=1;
ind = mod(round(0:delta:(delta*duration*fs-1)),N0)+1;
stem(ind),
stem(P(ind))
```

(.wav)



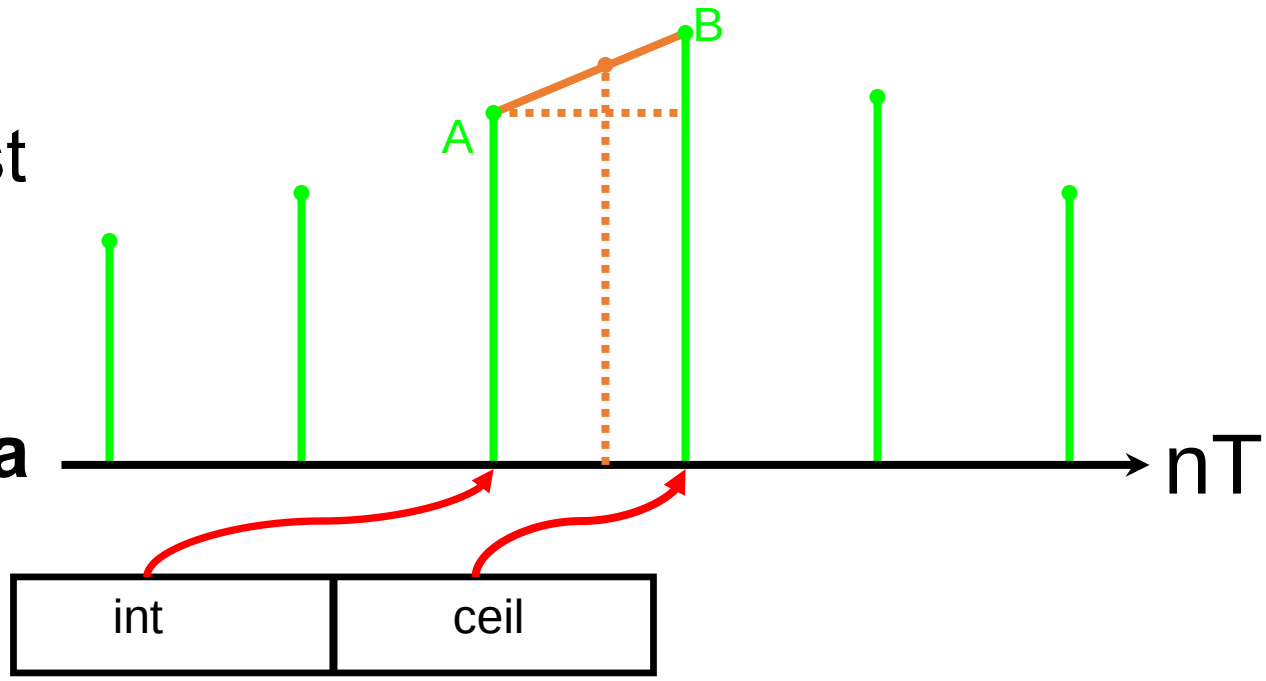
- **Problém:**
  adresy nejsou celá čísla!

Tabulková syntéza v Matlabu

- **Vstupní hodnota není celé číslo**

- Můžeme se rozhodnout:

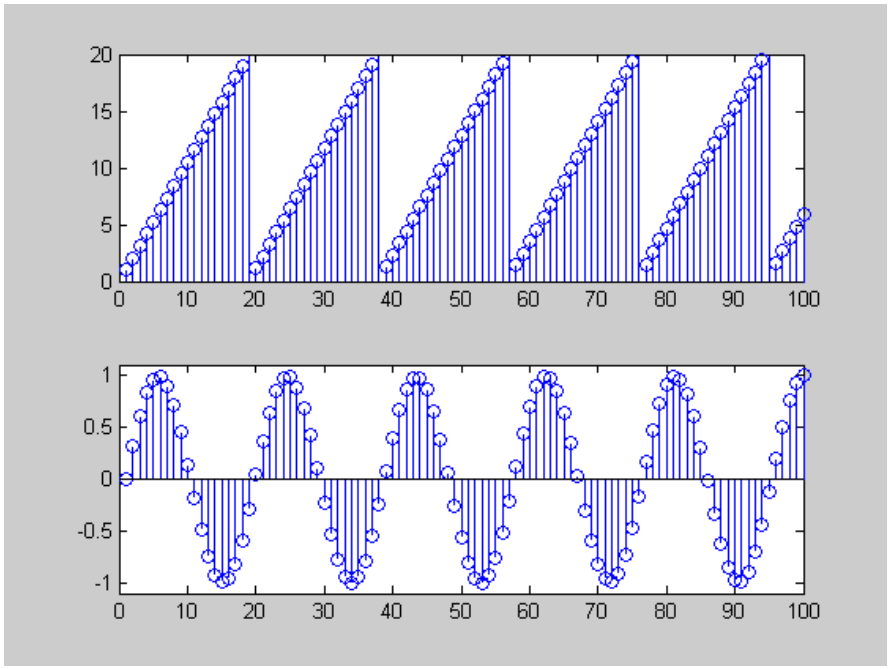
1. vzít celé číslo a oříznout zlomkovou část
2. zaokrouhlit na nejbližší celé číslo
3. **interpolovat mezi dvěma body vlnové tabulky**

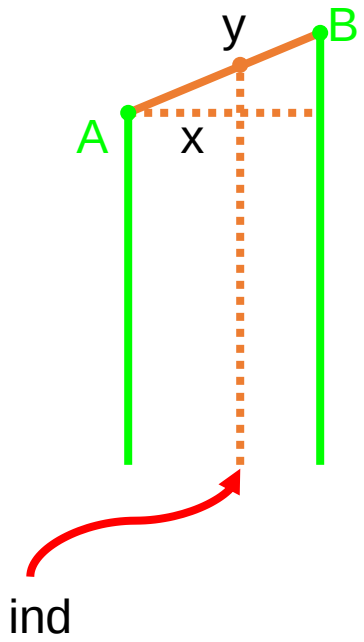


Tabulková syntéza v Matlabu

- **Lineármí interpolace**

```
% transformace tabulky s jednou periodou f0
% na libovolne dlouhy signal o frekvenci f1
% pomoci linearni interpolace

duration=1;
ind = mod(0:delta:(delta*duration*fs-1),N0)+1;
x=ind-floor(ind);
P=[P P(1)];
A=P(floor(ind));
B=P(ceil(ind));
y=(B-A).*x+A;
stem(ind)
stem(y)
```



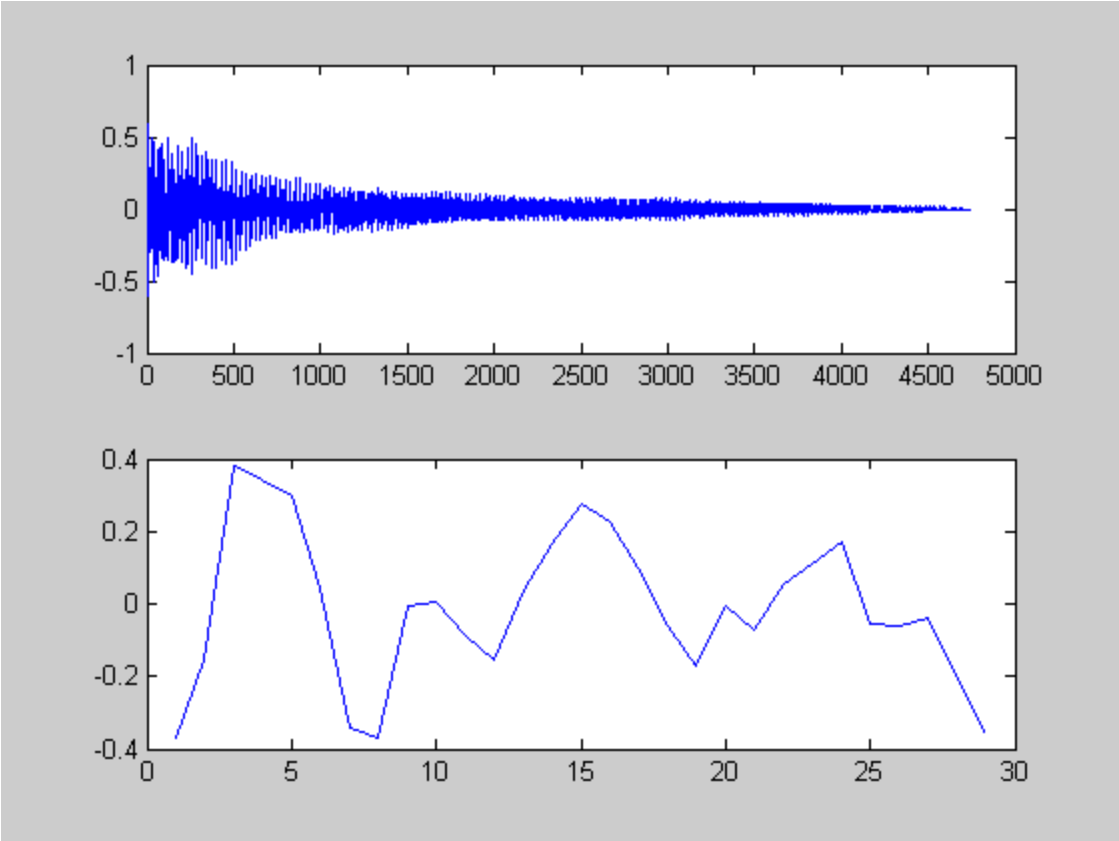


Tabulková syntéza v Matlabu

```
function y = tabsynth(P, f, duration, fs)
% Wavetable synthesis with linear interpolation
% P        = one period of the sampled signal (the table)
% f        = desired frequency of the output signal
% duration = duration of the output signal
% fs       = sampling frequency of the output signal
% y        = output signal
% Usage:   y = tabsynth(P, f, duration, fs)
% by Roman Cmejla

P = P(:)';   % Ensure P is a row vector
delta = f * length(P) / fs;   % Step size for reading the table
ind = mod(0:delta:(delta * duration * fs - 1), length(P)) + 1;  % Index in the table
x   = ind - floor(ind);       % Fractional part for interpolation
P   = [P P(1)];               % Wrap the table for interpolation
A   = P(floor(ind));          % First interpolation point
B   = P(ceil(ind));           % Second interpolation point
y   = (B - A) .* x + A;       % Linear interpolation
```

Tabulková syntéza v Matlabu

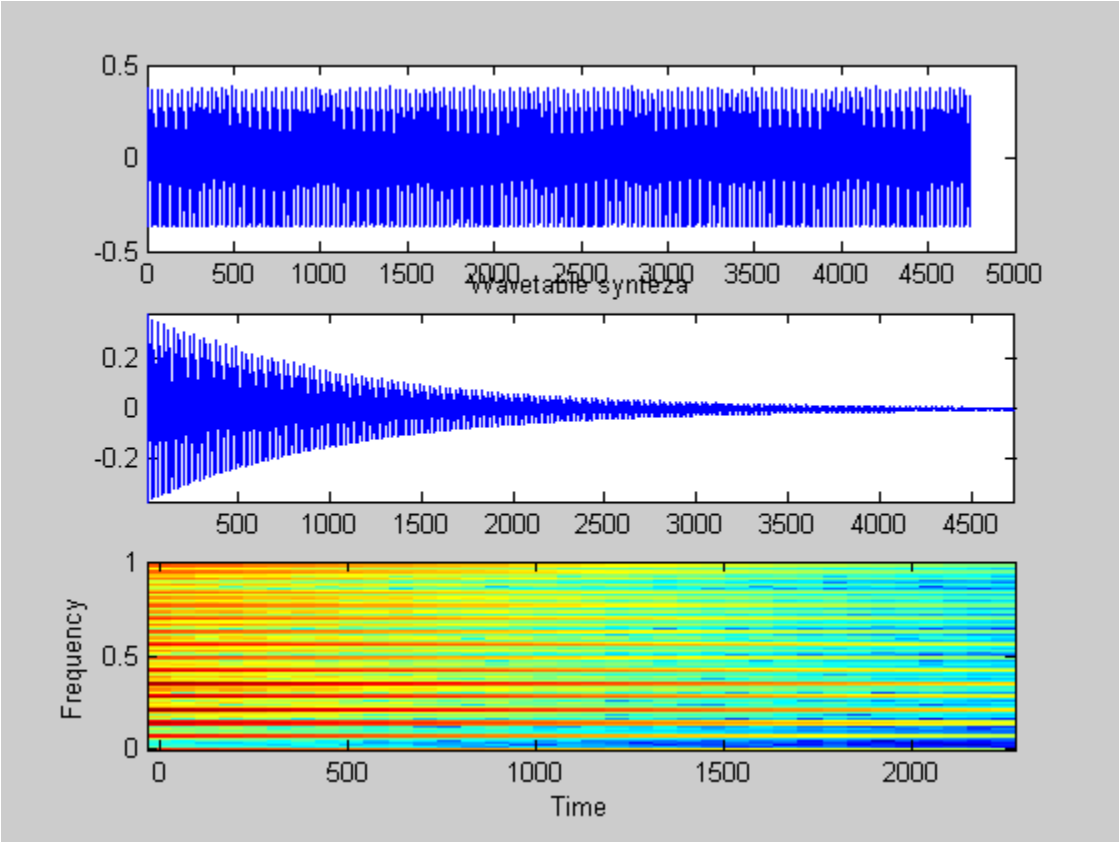
- **banjo**




```
% example of wavetable synthesis of a banjo
[x,fs]=wavread('banjo.wav');
P=x(144:172);
```

Tabulková syntéza v Matlabu

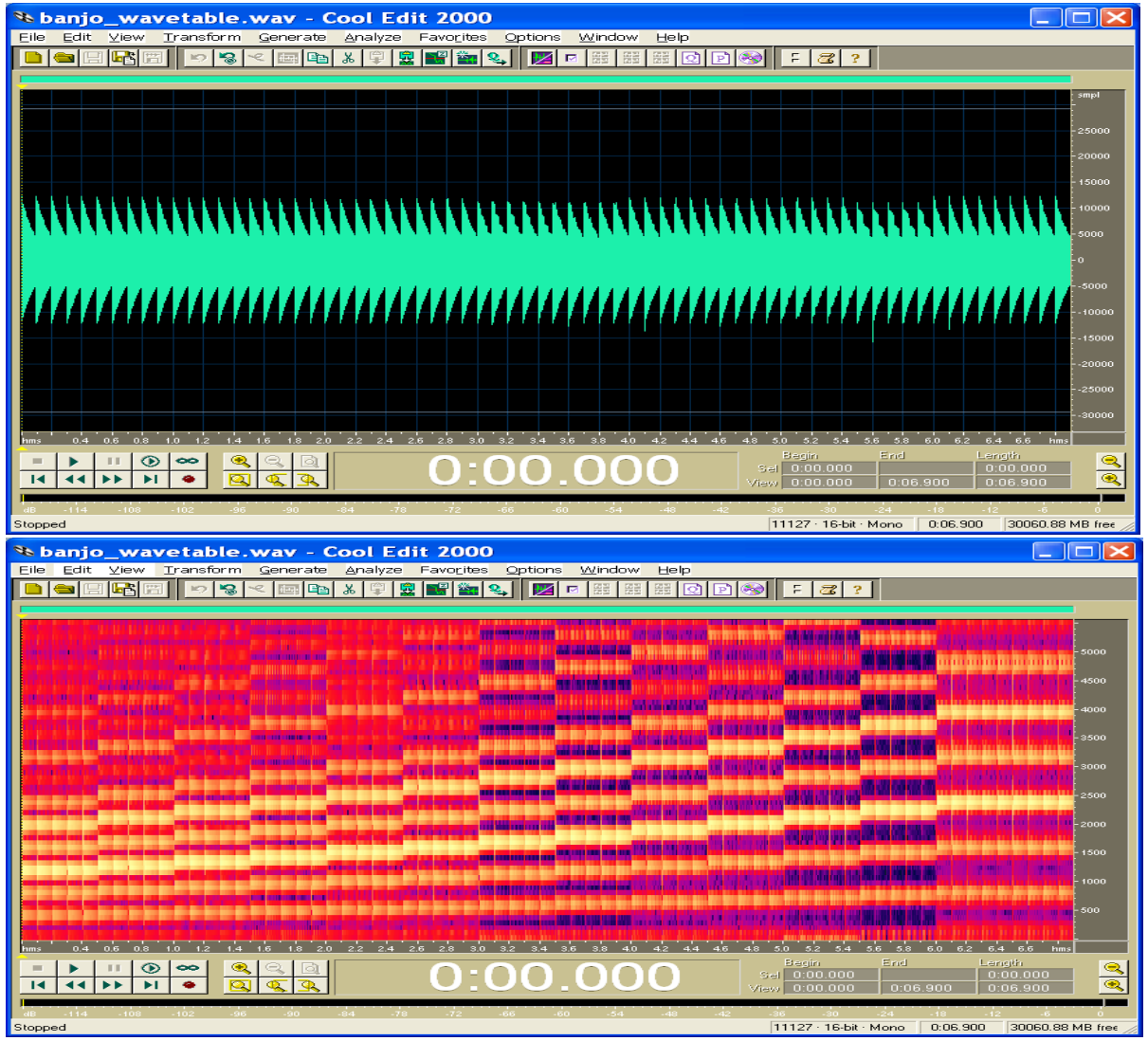
- **banjo**




```
y=tabsynth(P,f0,duration,fs);
y=exp(-[0:length(y)-1]./fs./.1).*y;
```

Tabulková syntéza v Matlabu

- **banjo**



Aplikace tabulkové syntézy

- **Hudební syntéza** – klávesy, simulace hudebních nástrojů
- **Automotive** – blinkry, výstražné signály, volnoběh motoru
- **Výuka DSP** – demonstrace principů digitální syntézy
- **Zvukové efekty** – hry, simulace prostředí, multimédia

Ukázky zvuků:
Blinkr – mechanický (blinkr_mech.wav)
Blinkr – elektronický (blinkr_el.wav)
Couvání – základní (reverse_beep_basic.wav)
Couvání – urgentní (reverse_beep_urgent.wav)
Motor – 4 válce (engine_idle_4cyl.wav)
Motor – 6 válců (engine_idle_6cyl.wav)