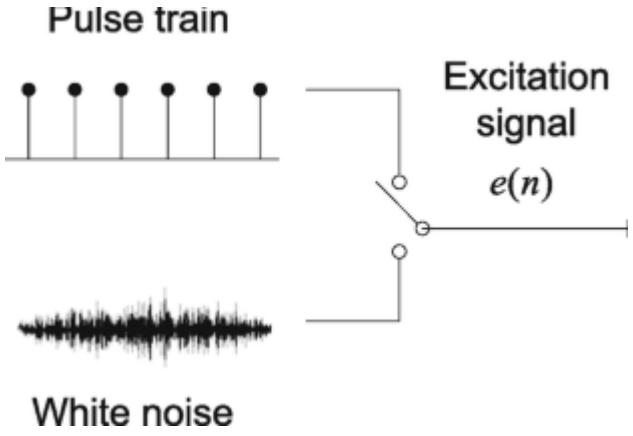
FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

5. PŘEDNÁŠKA – Filtrační syntéza

- **Filtrační syntéza**
  - Pulsní buzení
  - Buzení šumem
- **Jednoduché číslicové filtry**
  - Filtry s jedním pólem a jednou nulou
  - Číslicová filtrace v MATLABu
  - Filtry s časově proměnnými koeficienty
- **MIDI v MATLABu**
  - Úvod do MIDI
  - Příklad použití MIDI v MATLABu

Filtrační syntéza

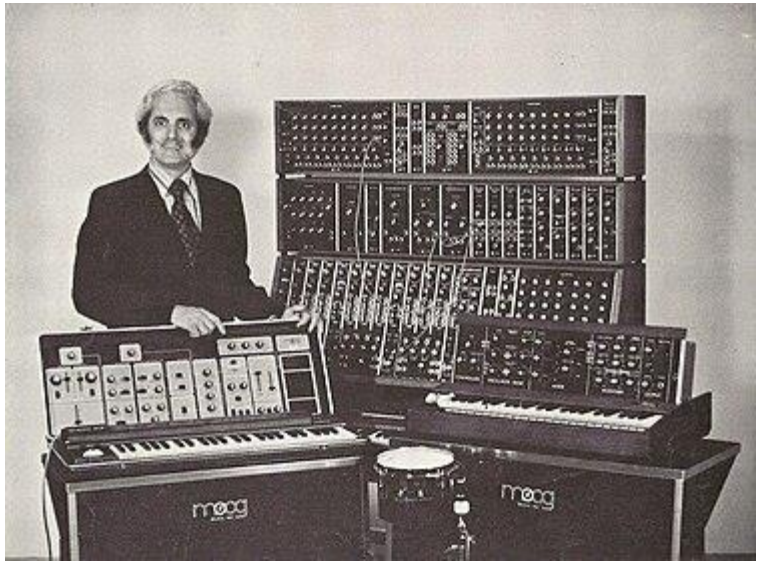
- **Základní princip filtrační syntézy:**
  - Syntetický signál se vytváří odečtením specifických spektrálních složek od zdrojového signálu.
  - To umožňuje tvarování zvuku filtrováním nežádoucích frekvencí.



- **Typy buzení:**
  - **Pulzní buzení:**
    - Vytváří ostré, periodické zvuky generováním průběhů, jako jsou pilovité, trojúhelníkové, obdélníkové a úzké obdélníkové pulzy.
    - Tento typ buzení je ideální pro simulaci strunných nástrojů (např. houslí, violoncella) a žesťových nástrojů (např. trumpet, pozounů).
  - **Šumové buzení:**
    - Generuje bohatší a složitější textury využitím náhodných šumových signálů.
    - Šumové buzení je obzvláště účinné pro vytváření perkusních zvuků, protože poskytuje náhodnost potřebnou pro nástroje jako malé bubínky, činely a bonga.
    - Ve filtrační syntéze se k tvarování a zjemňování zvuku používá barevný šum (bílý, růžový, červený, modrý, fialový).
    - Běžně se používá ve zvukovém designu k simulaci přírodních zvuků, jako je vítr, déšť, vlny a šustění listí.

Filtrační syntéza

- **Syntezátory:**
  - Historické modely:
    - Vocoder (1930), Moog (1960 – 1970) (průkopník v technologii syntezátorů)
  - Moderní modely:
    - Korg MS2000, Access Virus (poskytují pokročilé možnosti filtrování a modulace)

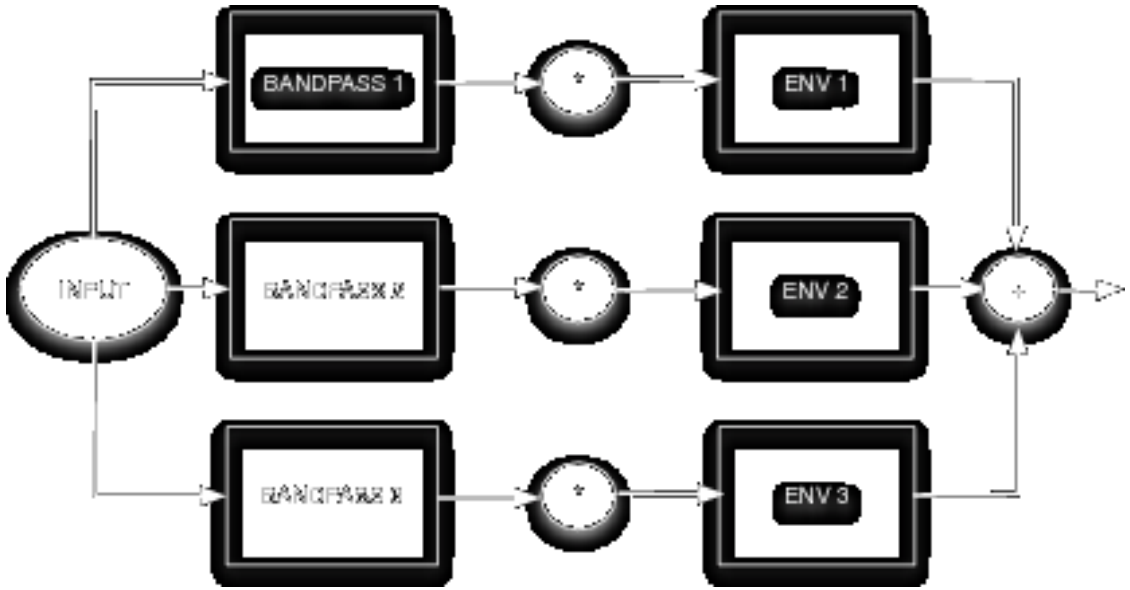


Robert Moog in the 1970s


Korg MS2000B

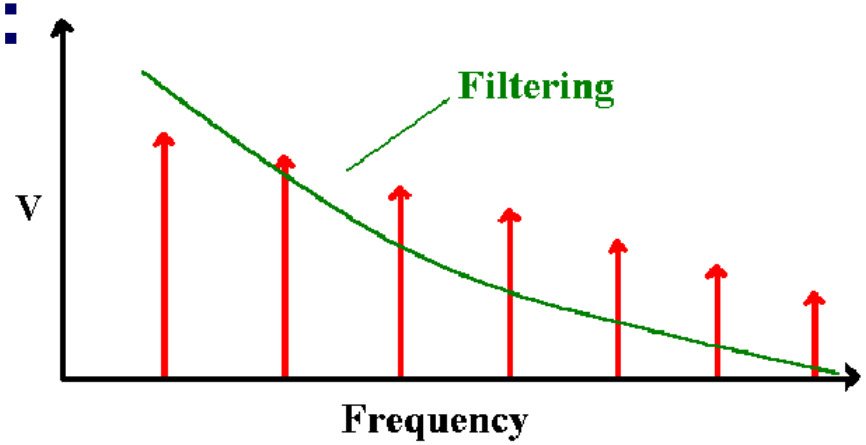
Virus Indigo

Filtrační syntéza

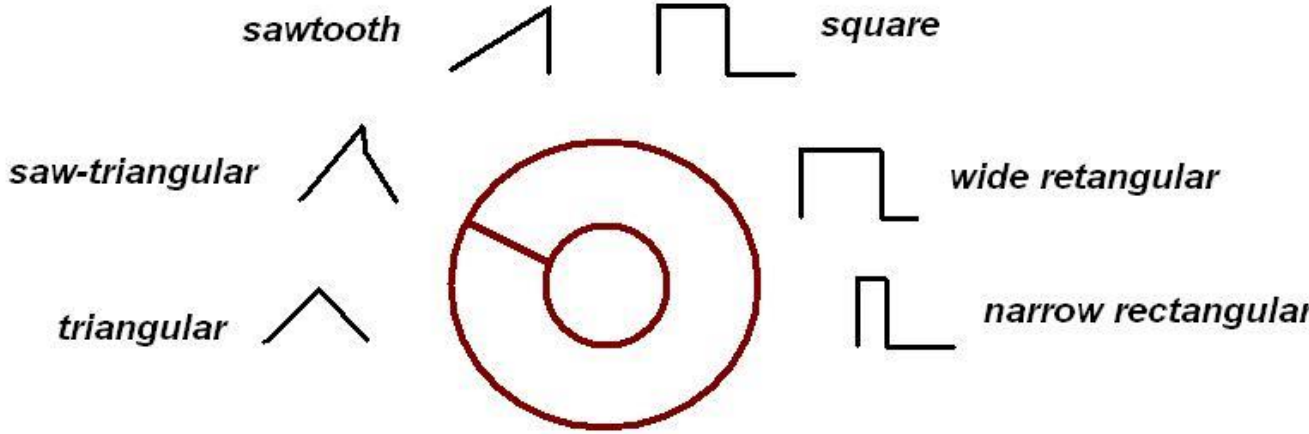


Filtrační syntéza

- **Periodické tvary vln používané v syntéze:**
  - Pila
    Všechny harmonické; „svěží“ zvuky.
  - Trojúhelník
    Měkčí zvuky; „hladší“ tóny.
  - Obdélník
    „Dutý“ zvuk.
  - Úzké obdélníkové pulzy
    „Ostré“ zvuky.

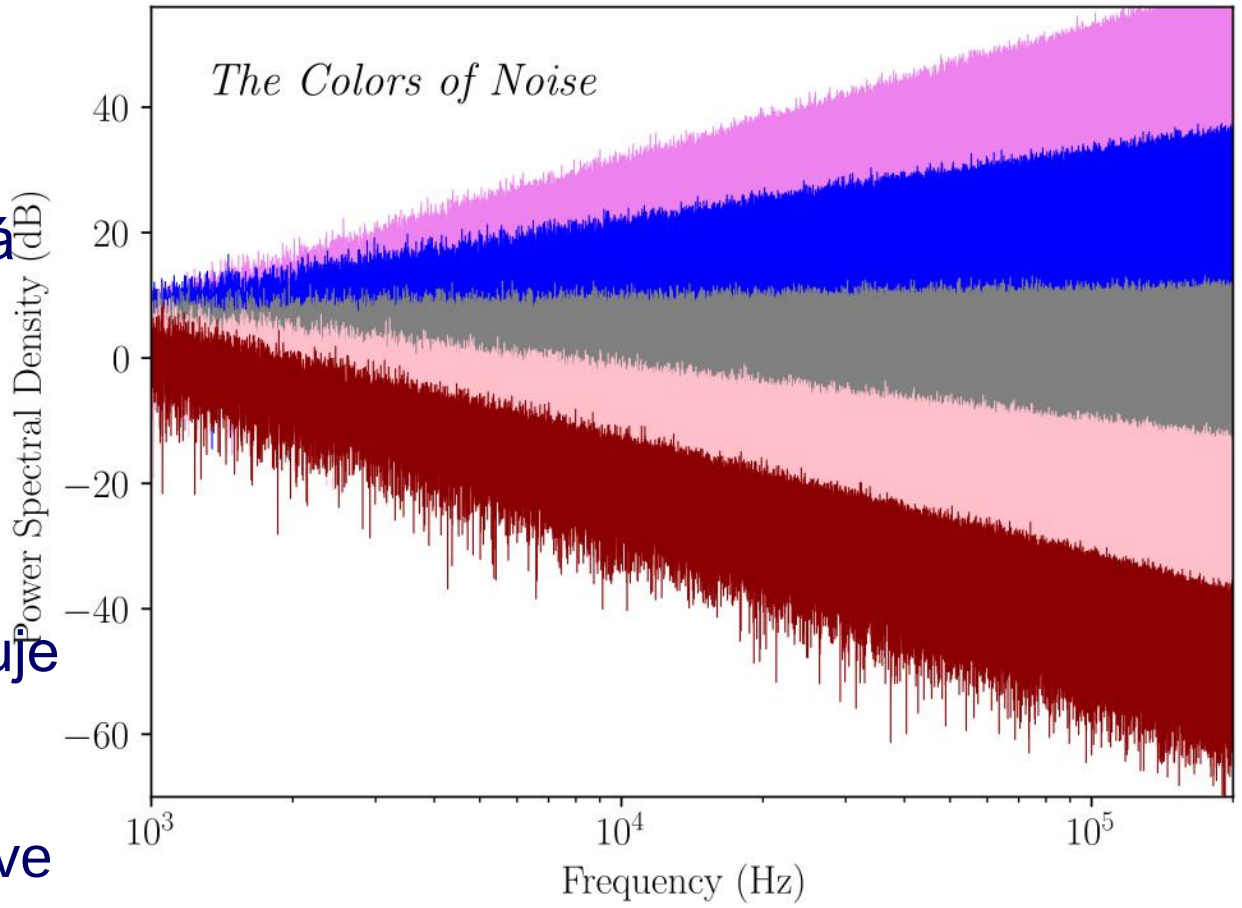


- **Použití u hudebních nástrojů:**
  - Smyčce: housle, violoncella
    pily, trojúhelníky.
  - Dechy: flétny, klarinety
    pulsní a šumové buzení.
  - Žestě: Trumpety, pozouny
    obdélníkové průběhy.



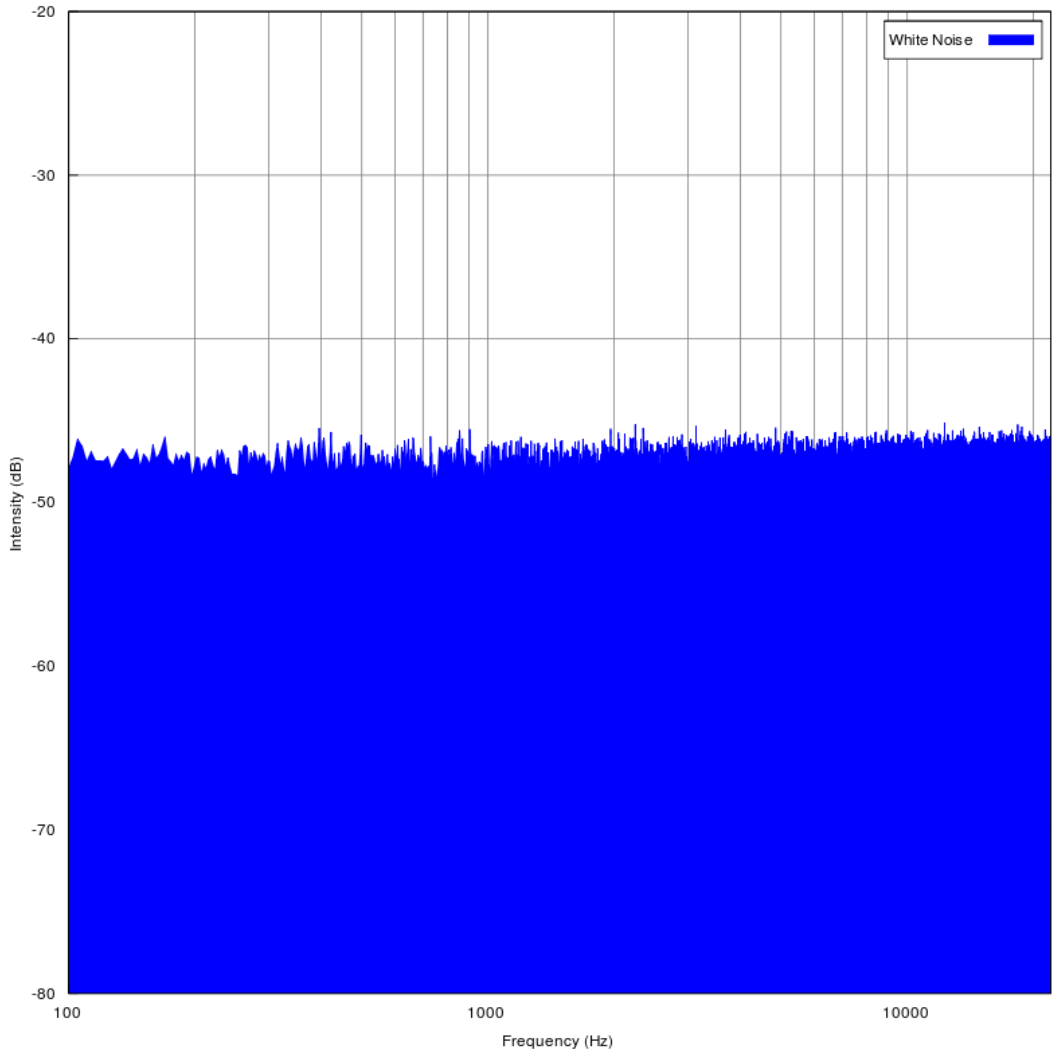
Filtrační syntéza

- **Typy barevného šumu:**
  - Bílý šum
    - obsahuje všechny frekvence se stejnou intenzitou
  - Růžový šum
    - snižuje energii o -3 dB na oktávu, stejná energie v každé oktávě
  - Červený šum
    - dále snižuje o -6 dB na oktávu, zdůrazňuje nižší frekvence
  - Modrý šum
    - zvyšuje se o +3 dB na oktávu, zdůrazňuje vyšší frekvence
  - Fialový šum
    - zvyšuje se o +6 dB na oktávu, výrazný ve velmi vysokých frekvencích



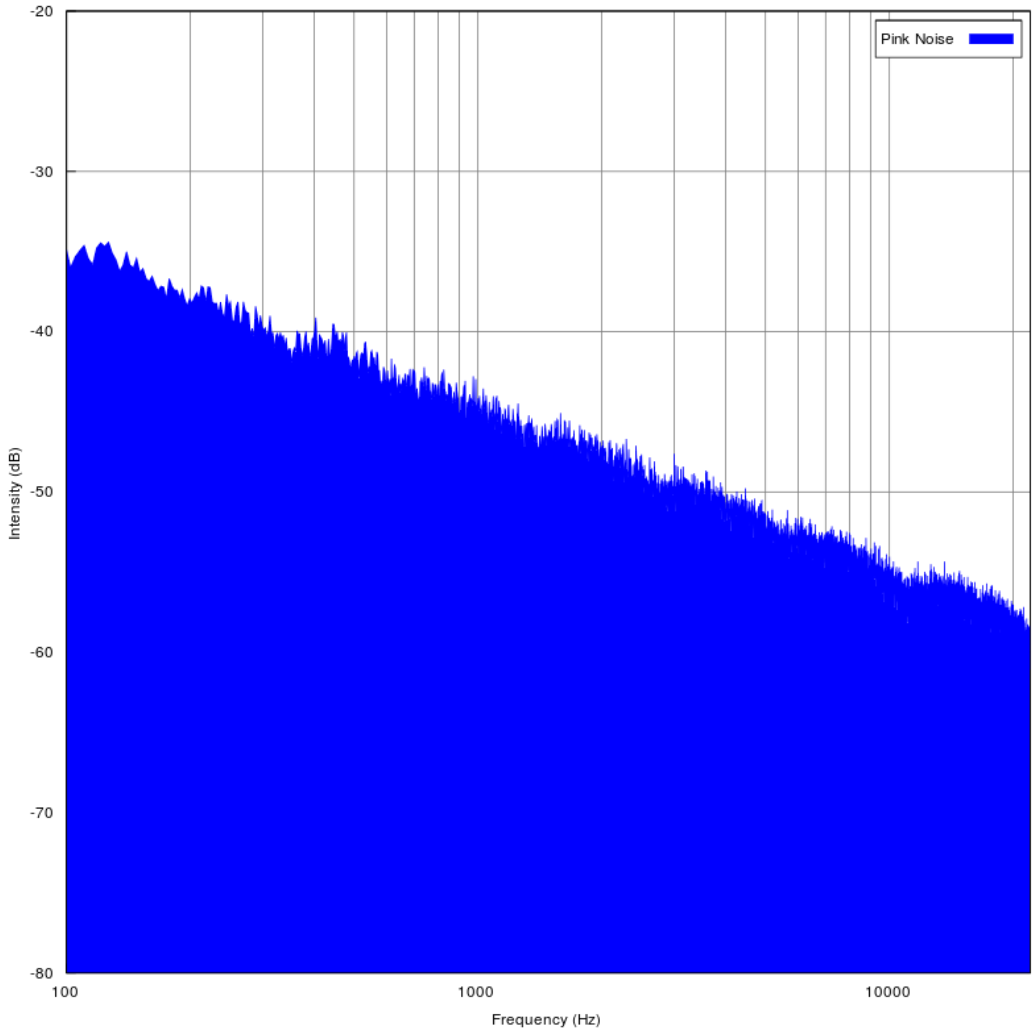
Filtrační syntéza

- **Bílý šum**
  - **obsahuje všechny frekvence se stejnou energií**
  - **aplikace**:
    - perkusní zvuky,
    - činely,
    - bubny,
    - bonga,
    - déšť,
    - exploze,
    - vysavač



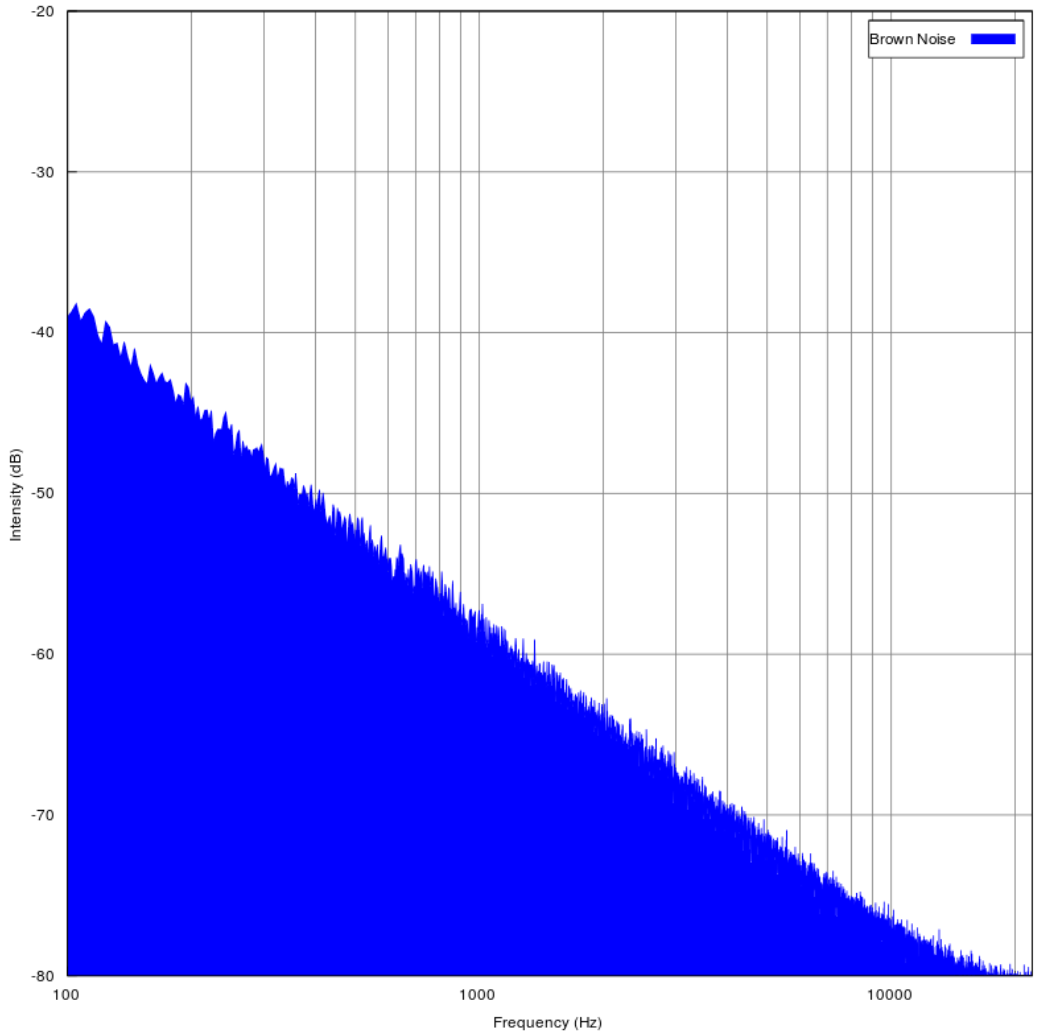
Filtrační syntéza

- **Růžový šum**
  - **+10dB/dek**
    (stejné energie v každé oktávě)
  - **aplikace**:
    - vlny oceánu,
    - šustění rostlin nebo stromů,
    - řeka,
    - turbíny,
    - dozvuk bílého šumu v efektech



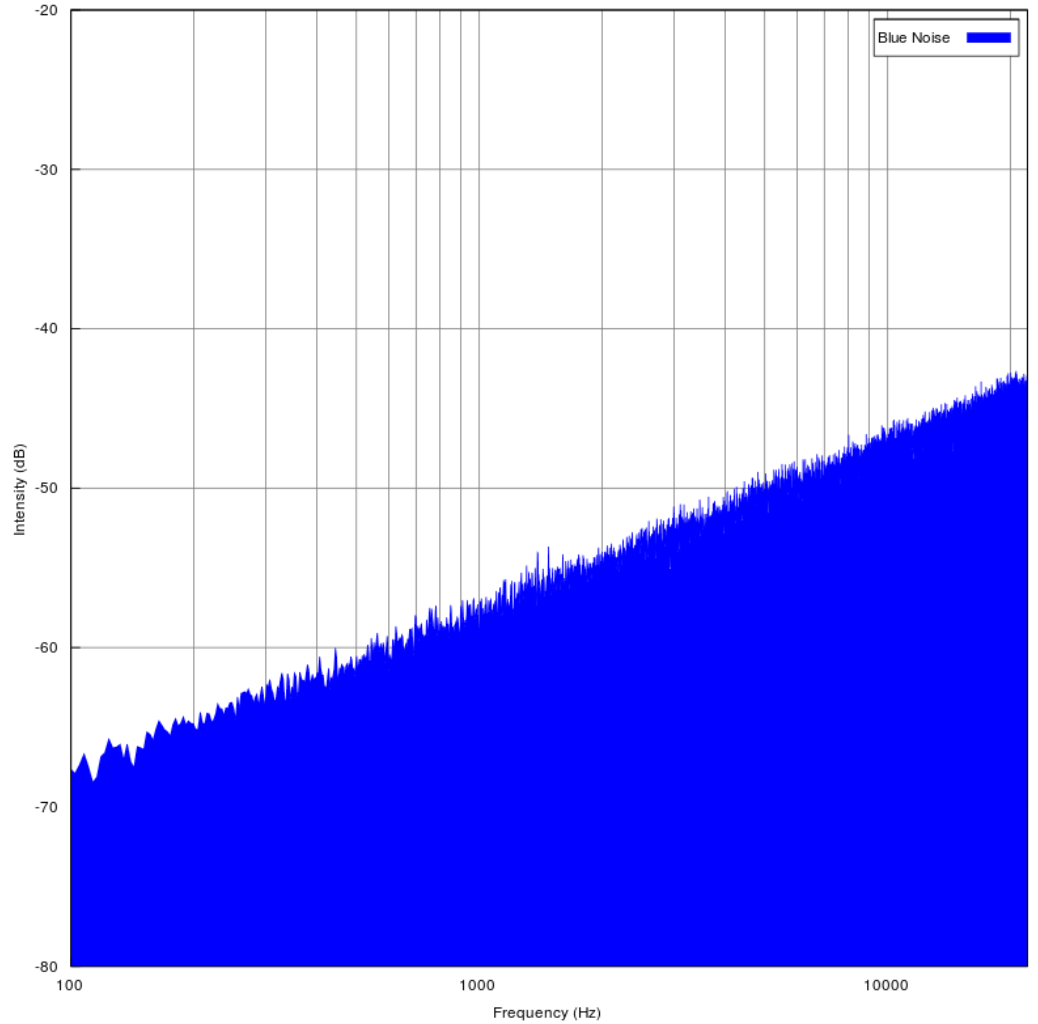
Filtrační syntéza

- **Červený šum**
  - **-20dB/dek**
  - **aplikace**:
    - bouřky,
    - hromy,
    - výbuchy,
    - motory,
    - hučení,
    - burácení,
    - sprcha,
    - chůze v písku,
    - výrazné nízkofrekvenční efekty



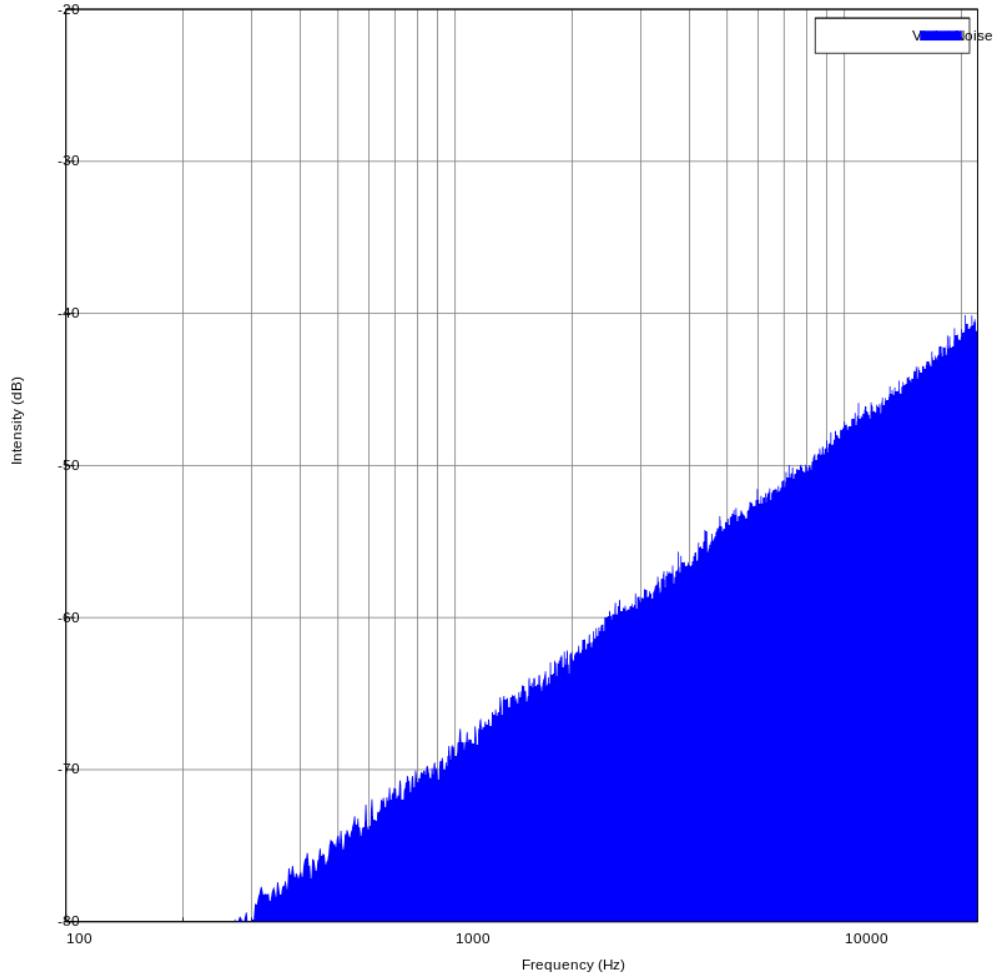
Filtrační syntéza

- **Modrý šum**
  - **+10dB/dek**
  - **aplikace**:
    - lasery,
    - elektronické efekty,
    - kosmické prvky,
    - generování vysokofrekvenčních efektů



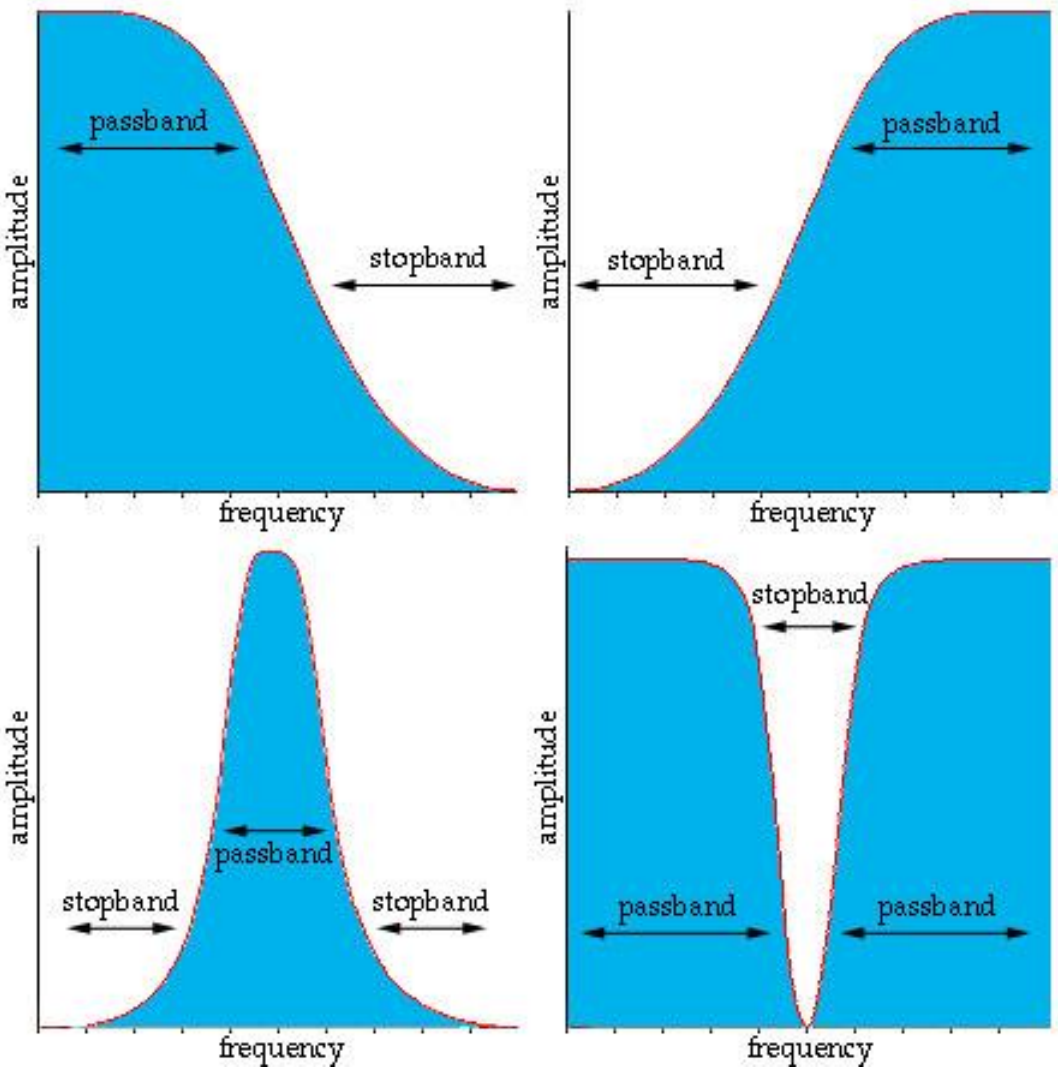
Filtrační syntéza

- **Fialový šum**
  - **+20dB/dek**
  - **aplikace**:
    - perkusní zvuky,
    - činely,
    - bubny,
    - bonga,
    - déšť,
    - exploze,
    - vysavač



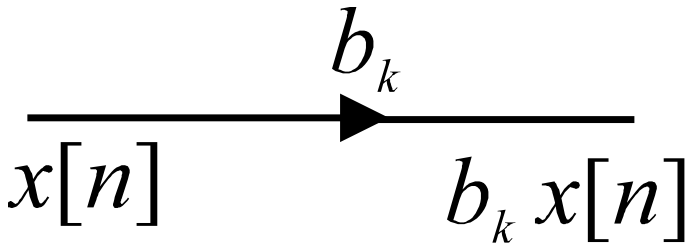
Jednoduché číslicové filtry

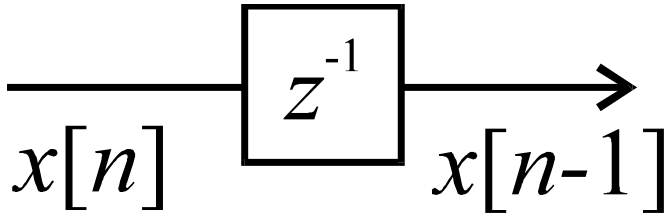
- **Číslicové filtry**
  - algoritmy provádějící lineární kombinaci vzorků vstupního a výstupního signálu tak, aby došlo ke zvýraznění, nebo naopak potlačení vybraných složek signálu.
- **Návrh filtrů**
  - Návrh koeficientů lineární kombinace (≡ koeficientů filtrů)

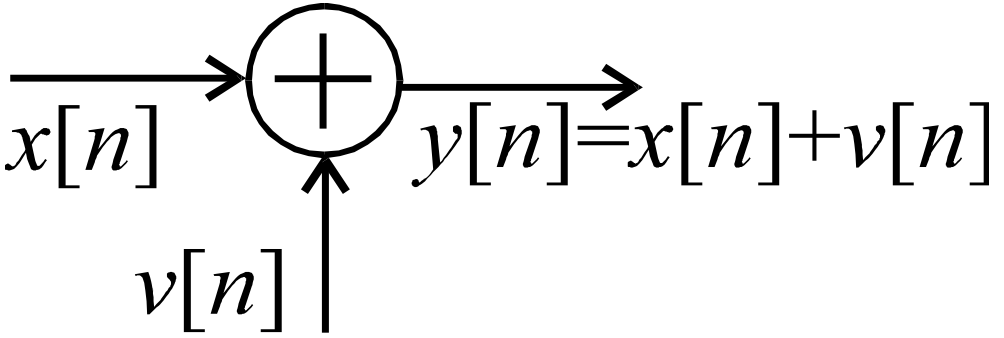


Jednoduché číslicové filtry

- **Základní operace číslicové filtrace**
  - Násobení konstantou
  - Zpoždění o jeden vzorek
  - Sčítání dvou posloupností

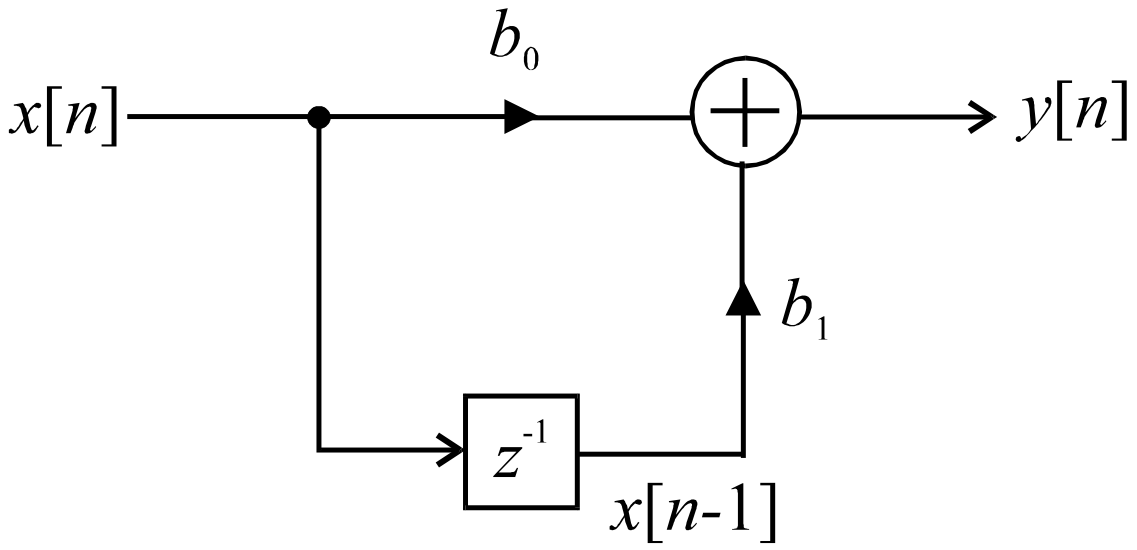






Jednoduché číslicové filtry – FIR(1)

- **FIR filtry prvního řádu**



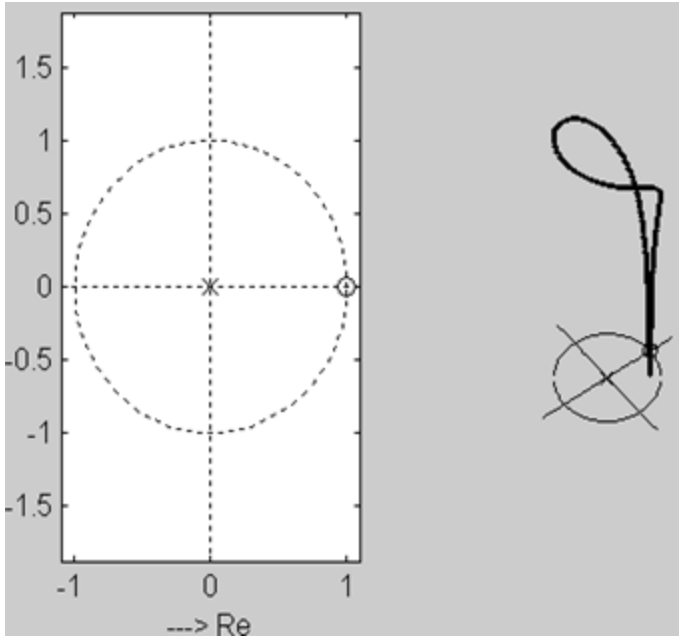
$x[n]$ … vstupní vzorek

$y[n]$ … výstupní vzorek

b_0, b_1.. Koeficienty filtru

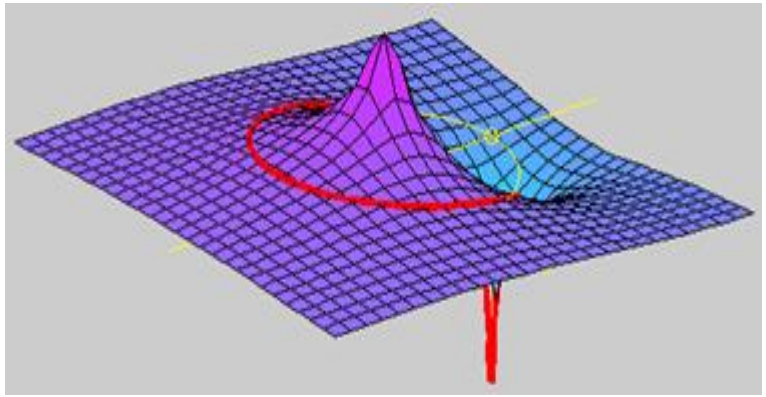
Jednoduché číslicové filtry – FIR(1)

- **FIR filtry s jednou nulou**
  - Základní tvar FIR filtru prvního řádu je v kombinaci aktuálního vzorku s předchozím.
  - Matematický model (diferenční rovnice):



$$y[n] = b_0 x[n] + b_1 x[n\text{-}1]$$

  - Tento filtr se nazývá „s jednou nulou“, protože jeho přenosová funkce má nulu v rovině z.



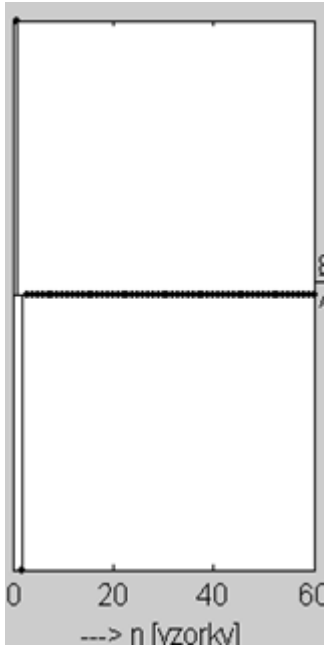
Jednoduché číslicové filtry – FIR(1)

- **Diferenční rovnice**:
  - $y[n] = b_0 x[n] + b_1 x[n\text{-}1]$
  - **Řešení:** rekurzivní výpočet výstupu filtru krok za krokem
    - počáteční podmínka x[-1] = 0
    - $n = 0$ $y[0] = b_0 x[0]$
    - $n = 1$ $y[1] = b_0 x[1] + b_1 x[0]$
    - $n = 2$ $y[2] = b_0 x[2] + b_1 x[1]$
    - $n = 3$ $y[3] = b_0 x[3] + b_1 x[2]$

Jednoduché číslicové filtry – FIR(1)

- **Impulsní charakteristika**
  - $x[-1] = 0, x[0] = 1, x[1] = 0, x[2] = 0 \ldots$
  - Řešení: $y[n] = b_0x[n] + b_1x[n-1]$

• $n = 0$	$y[0] = b_0x[0]$	$y[0] = b_0$
• $n = 1$	$y[1] = b_0x[1] + b_1x[0]$	$y[1] = b_1$
• $n > 1$	$y[2] = b_0x[2] + b_1x[1]$	$y[n] = 0$



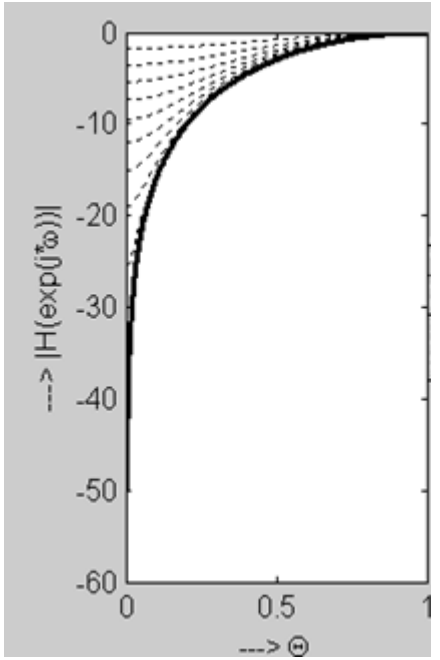
Jednoduché číslicové filtry – FIR(1)

- **Z-transformace a přenosová funkce**
  - $y[n] = b_0 x[n] + b_1 x[n-1]$
    $Y(z) = b_0 X(z) + b_1 z^{-1} X(z)$
  - $H(z) = Y(z)/X(z) = b_0 + b_1 z^{-1}$
    $= b_0 + b_1 / z$
    $= (b_0 z + b_1) / z$

Jednoduché číslicové filtry – FIR(1)

- **Frekvenční charakteristika**
  - Frekvenční charakteristika ukazuje, jak se filtr chová na různých frekvencích.

$$H(e^{j\Theta}) = b_0 + b_1 e^{-j\theta} = (b_0 e^{j\theta} + b_1) / e^{j\theta}, \quad \theta = 2\pi f / f_s$$

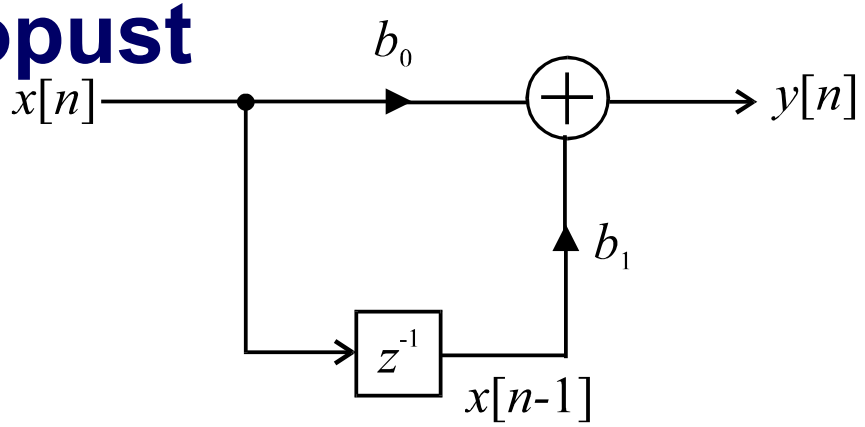


  - Důležité pro pochopení toho, jak filtr ovlivní signály na různých frekvencích (např. horní propust …).

frekv. charakteristiku získáme dosazením z = e^j2piF do H(z)

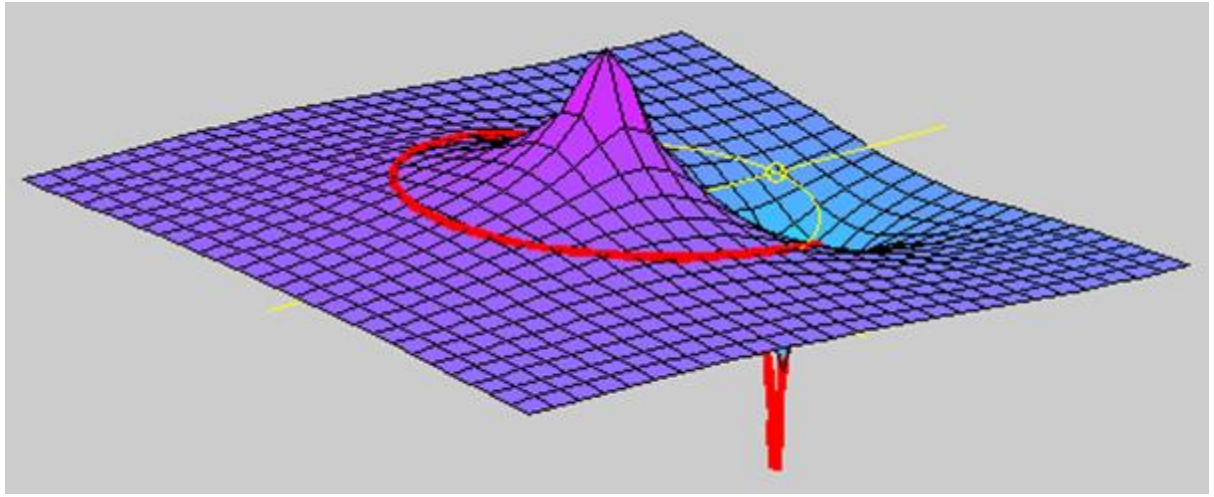
Jednoduché číslicové filtry – FIR(1)

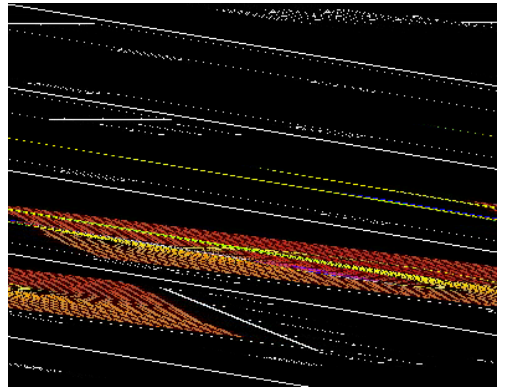
- Příklad: **horní propust**
  - $b_0 = 1$
  - $b_1 = -1...0$
  - nula $z = 1$
  - pól $z = 0$



$y[n] = x[n] - x[n-1]$

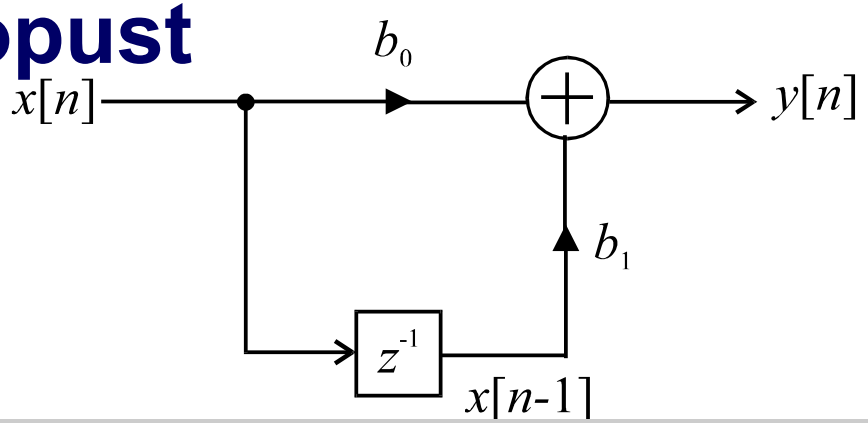
$H(z) = Y(z)/X(z) = 1 - z^{-1}$
$= 1 - 1/z = (z - 1)/z$





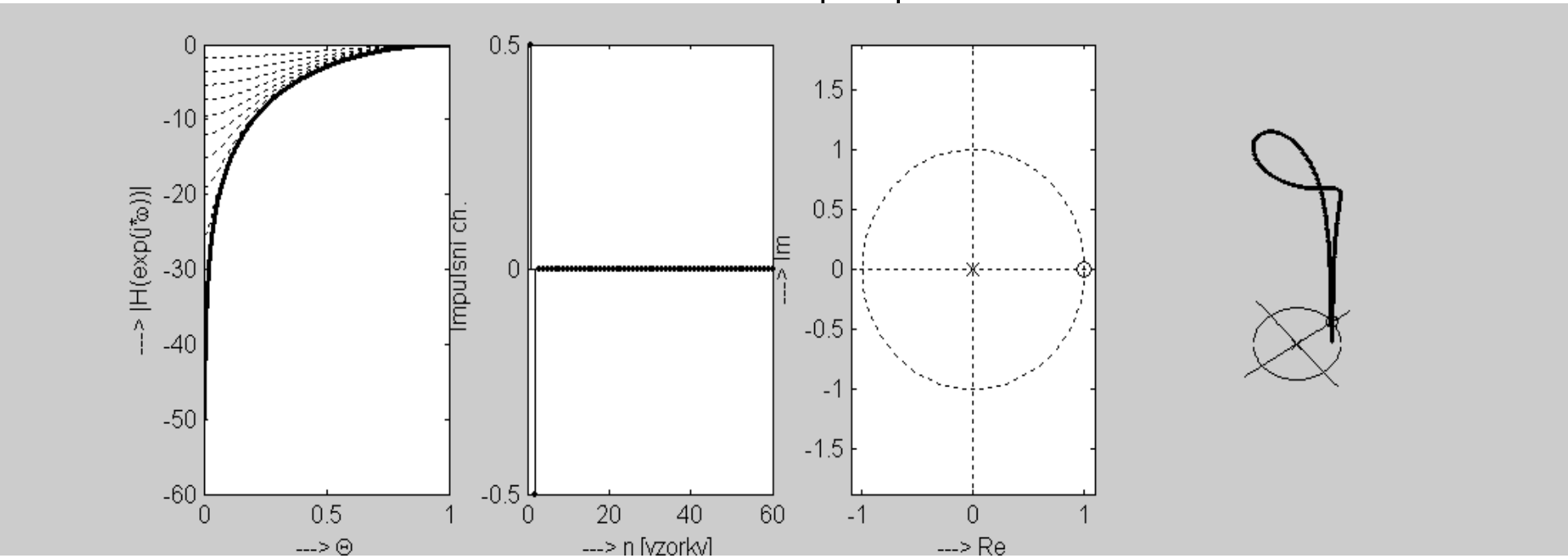
Jednoduché číslicové filtry – FIR(1)

- Příklad: **horní propust**
  - $b_0 = 1$
  - $b_1 = -1...0$
  - nula $z = 1$
  - pól $z = 0$



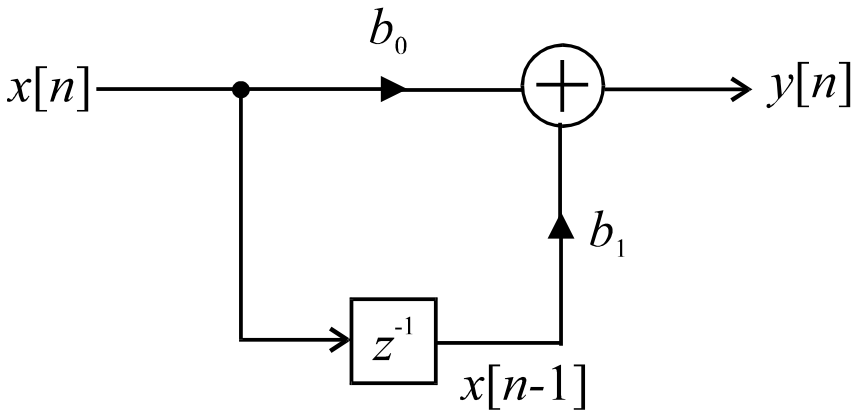
$y[n] = x[n] - x[n-1]$

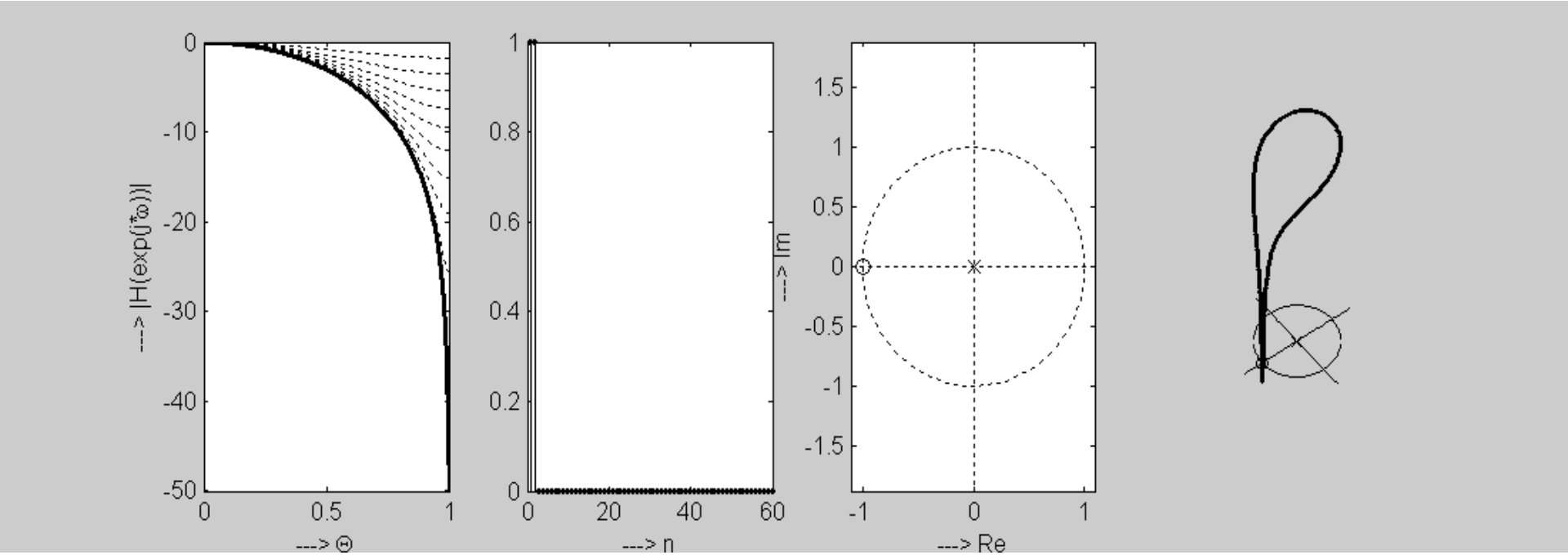
$H(z) = Y(z)/X(z) = 1 - z^{-1}$
$= 1 - 1/z = (z - 1)/z$



Jednoduché číslicové filtry – FIR(1)

- Příklad: **dolní propust**
  - $b_0 = 1$
  - $b_1 = 1...0$
  - nula $z = -1$
  - pól $z = 0$



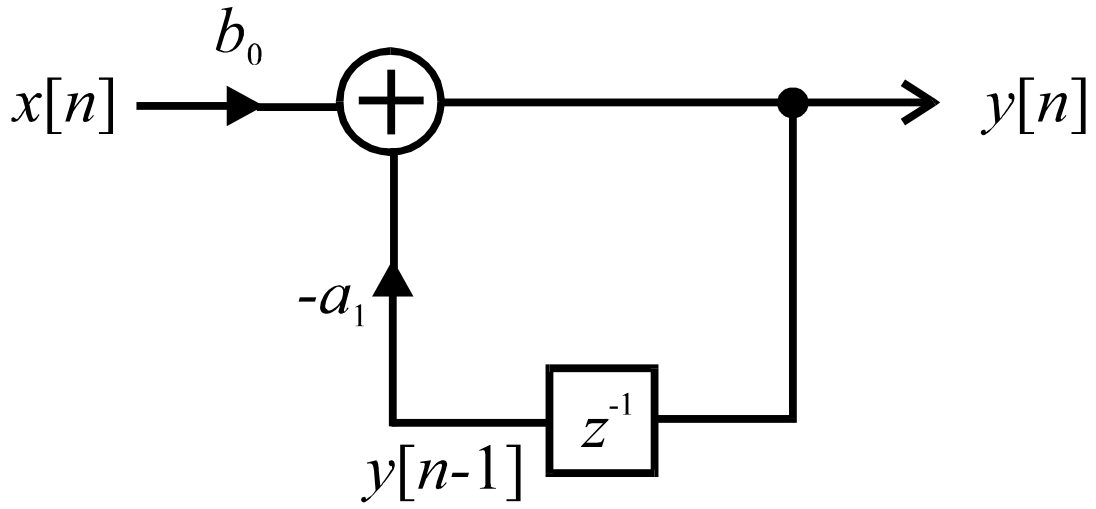


Jednoduché číslicové filtry – FIR

- **Praktické poznámky k FIR filtrům**
  - FIR filtry jsou vždy stabilní, protože nemají póly mimo jednotkovou kružnici.
  - Běžně se používají při zpracování zvuku pro úkoly, jako je ekvalizace a zpoždění
  - Konstrukce je flexibilní, ale FIR filtry vyžadují více koeficientů pro ostré frekvenční charakteristiky ve srovnání s IIR filtry.

Jednoduché číslicové filtry – IIR(1)

- **IIR filtry prvního řádu**



$x[n]$ … vstupní vzorek

$y[n]$ … výstupní vzorek

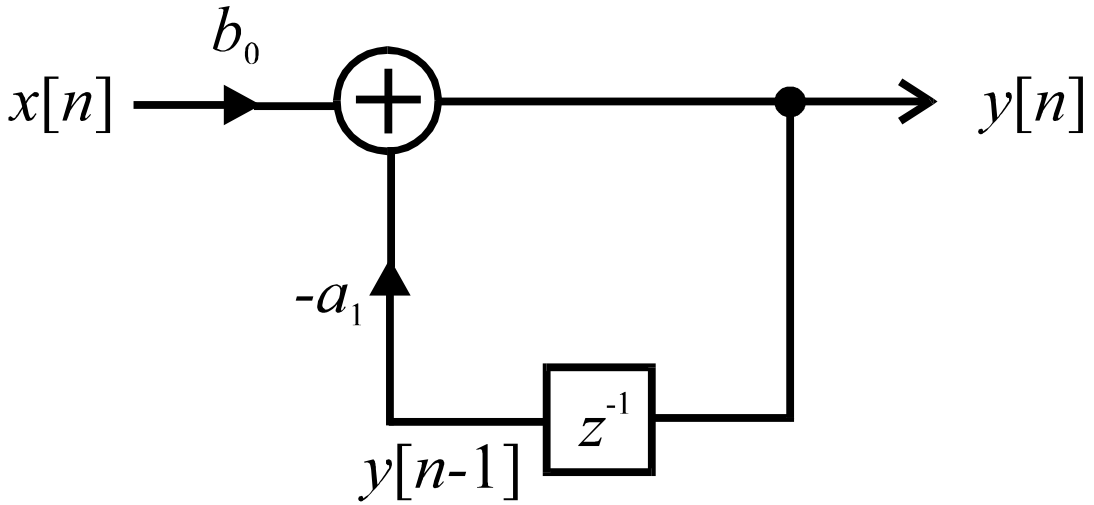
b_0, a_1.. koeficienty filtru

Jednoduché číslicové filtry – IIR(1)

- **Diferenční rovnice IIR filtru prvního řádu**

$$y[n] = b_0\, x[n] - a_1\, y[n\text{-}1]$$

  - Rovnice kombinuje aktuální a předchozí vstupní hodnoty a také předchozí výstupní hodnoty.



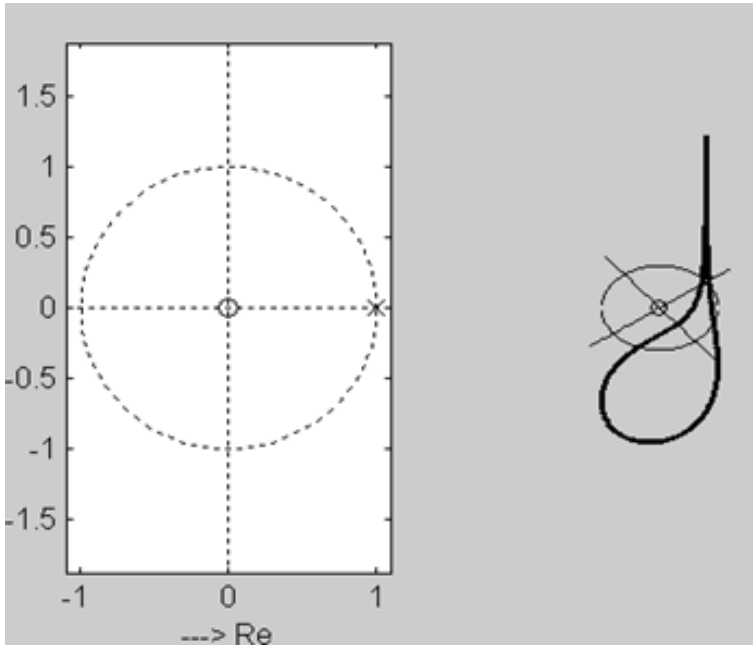
Jednoduché číslicové filtry – IIR(1)

- **Charakteristiky filtru v Z-oblasti**
  - Z-transformace

$$y[n] = b_0\, x[n] - a_1\, y[n-1]$$
$$Y(z) = b_0\, X(z) - a_1 z^{-1} Y(z)$$

  - Přenosová funkce

$$H(z) = \frac{b_0}{1 + a_1 z^{-1}} = \frac{b_0 z}{z + a_1}$$

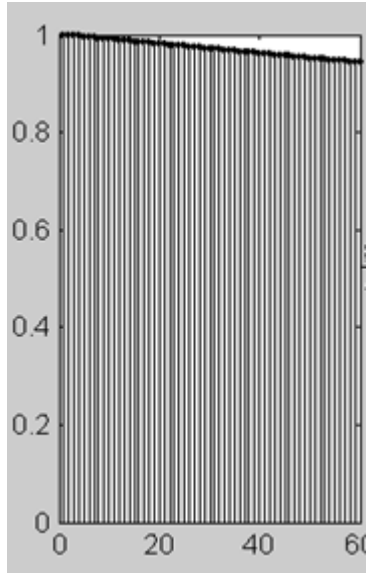


Jednoduché číslicové filtry – IIR(1)

- **Impulzní odezva**
  - Impulzní odezva IIR filtru teoreticky trvá neomezeně dlouho (kvůli zpětnovazební složce).
  - U IIR filtru prvního řádu odezva exponenciálně klesá:

$$y[n] = b_0\, x[n] - a_1\, y[n\text{-}1]$$

• $n = 0$	$y[0] = b_0 x[1]$	$y[0] = b_0$
• $n = 1$	$y[1] = b_0 x[1] - a_1\, y[0]$	$y[1] = - a_1\, b_0$
• $n = 2$	$y[2] = b_0 x[2] - a_1\, y[1]$	$y[2] = \; a_1\, a_1\, b_0$
• $n = 3$	$y[3] = b_0 x[3] - a_1\, y[2]$	$y[3] = - a_1\, a_1\, a_1\, b_0$

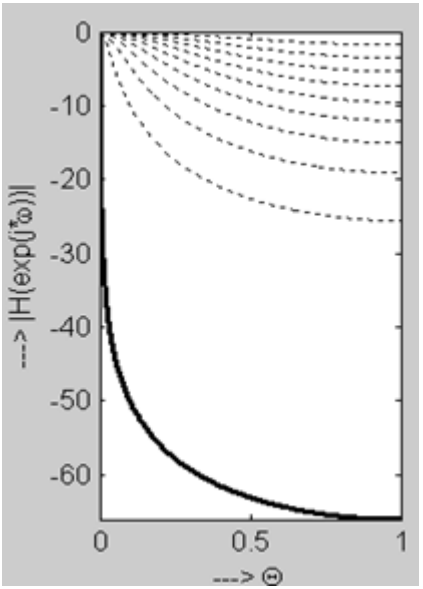


- Rychlost útlumu závisí na koeficientu zpětné vazby a_1.

Jednoduché číslicové filtry – IIR(1)

- **Frekvenční charakteristika IIR filtrů prvního řádu**

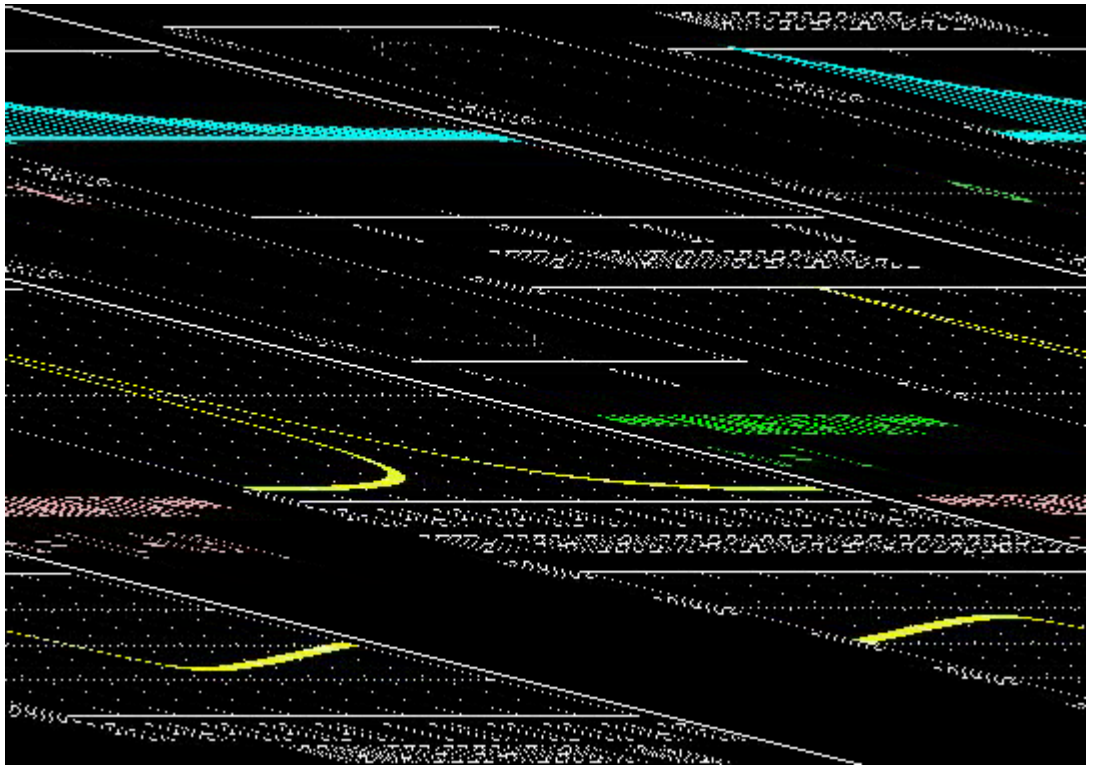
$$H(e^{j\theta}) = \frac{b_0}{1 + a_1 e^{-j\theta}}$$



- Charakteristika ukazuje chování filtru ve frekvenčním pásmu (horní propust. …)
  - Klíčové aspekty:
    - **Stabilita:** Filtr je stabilní, pokud je pól uvnitř jednotkové kružnice.
    - **Fázová charakteristika:** IIR filtry zavádí fázové posuny, které je třeba v některých aplikacích zohlednit.

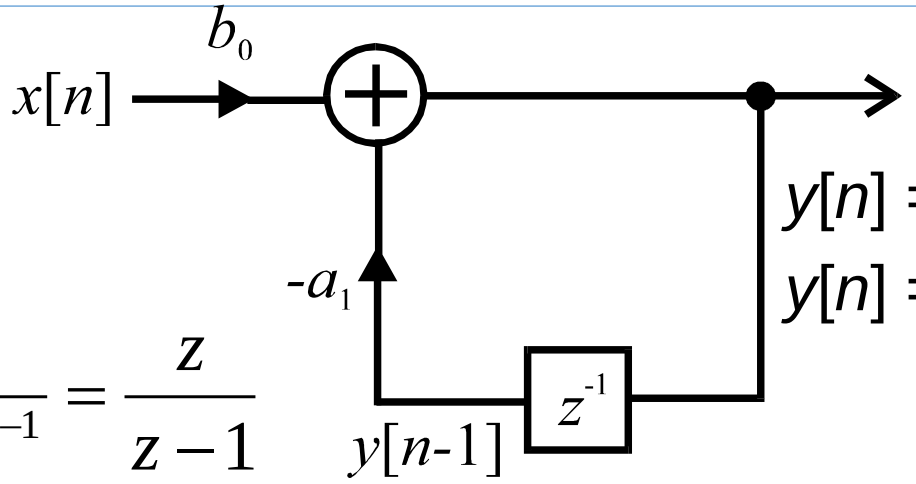
Jednoduché číslicové filtry – IIR(1)

- **Pól pohybující se podél reálné osy** – animace



Jednoduché číslicové filtry – IIR(1)

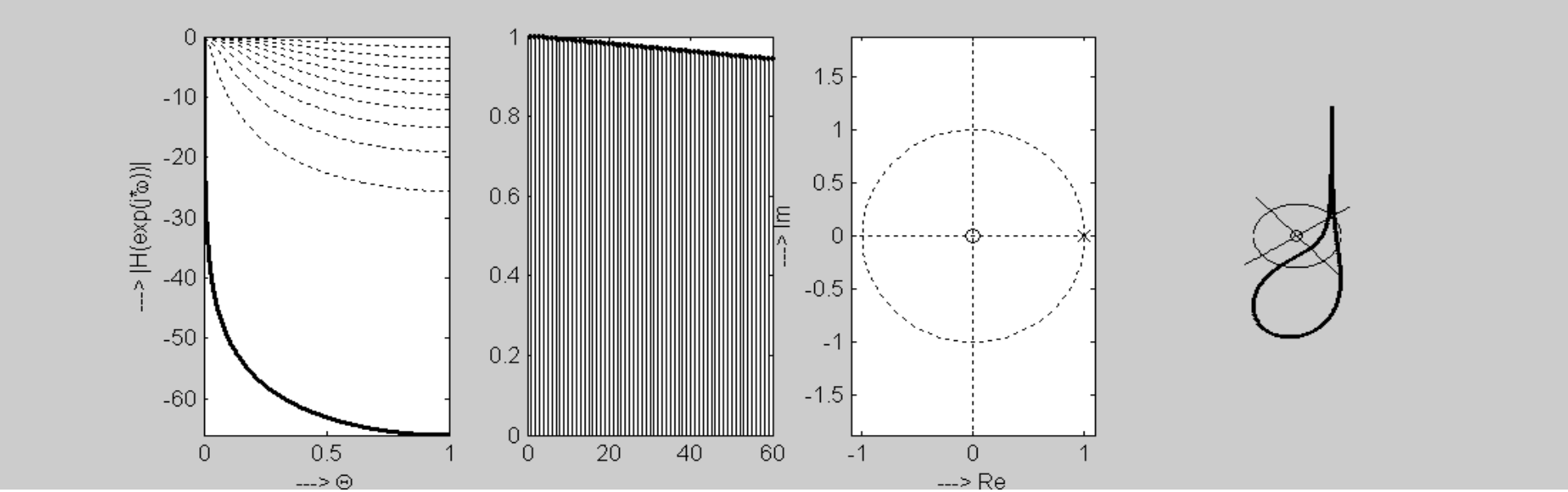
- Příklad: **dolní propust**
  - $b_0 = 1$
  - $a_1 = -0.999...0$
  - zero $z = 0$
  - pole $z = 1$



$$y[n] = b_0 x[n] - a_1 y[n-1]$$

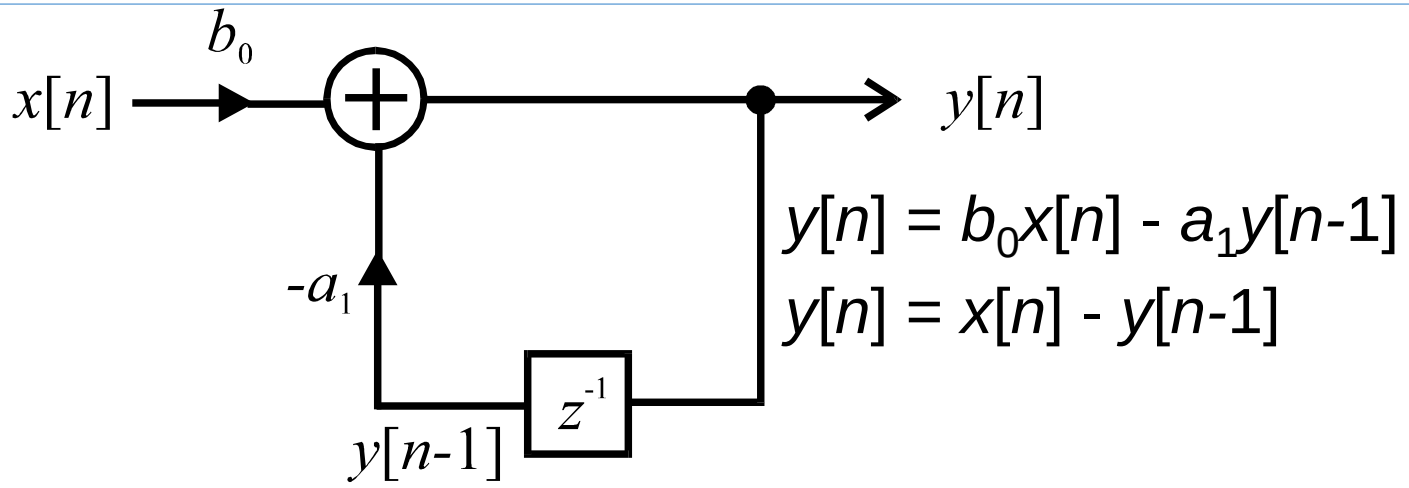
$$y[n] = x[n] + y[n-1]$$

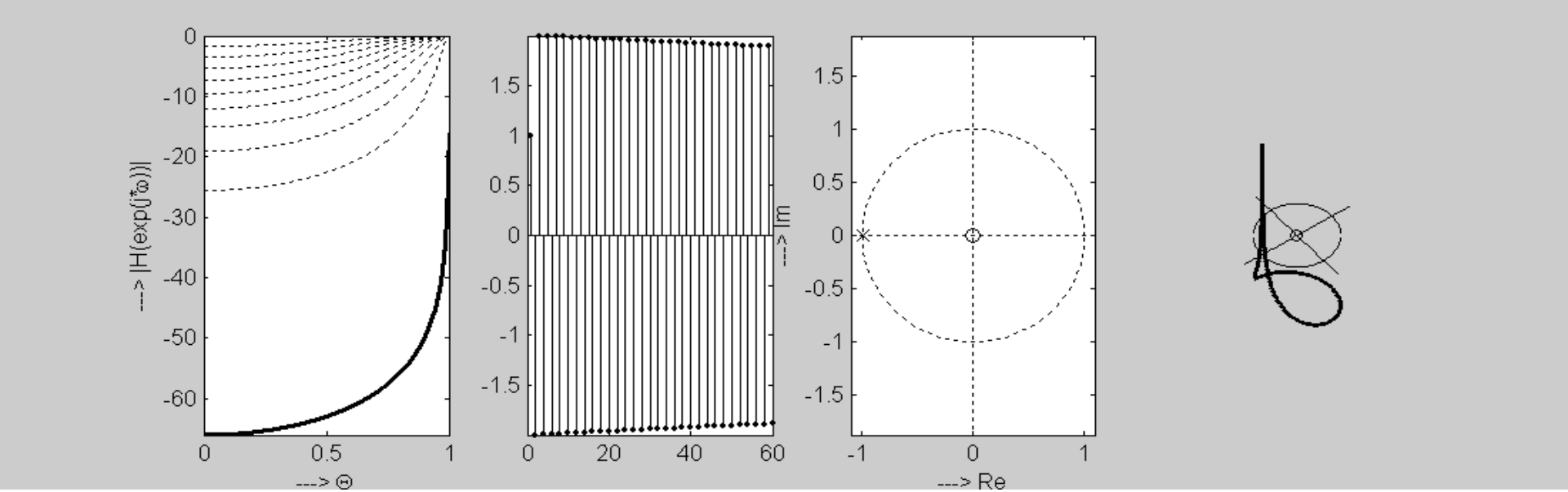
$$H(z) = \frac{1}{1 - z^{-1}} = \frac{z}{z - 1}$$



Jednoduché číslicové filtry – IIR(1)

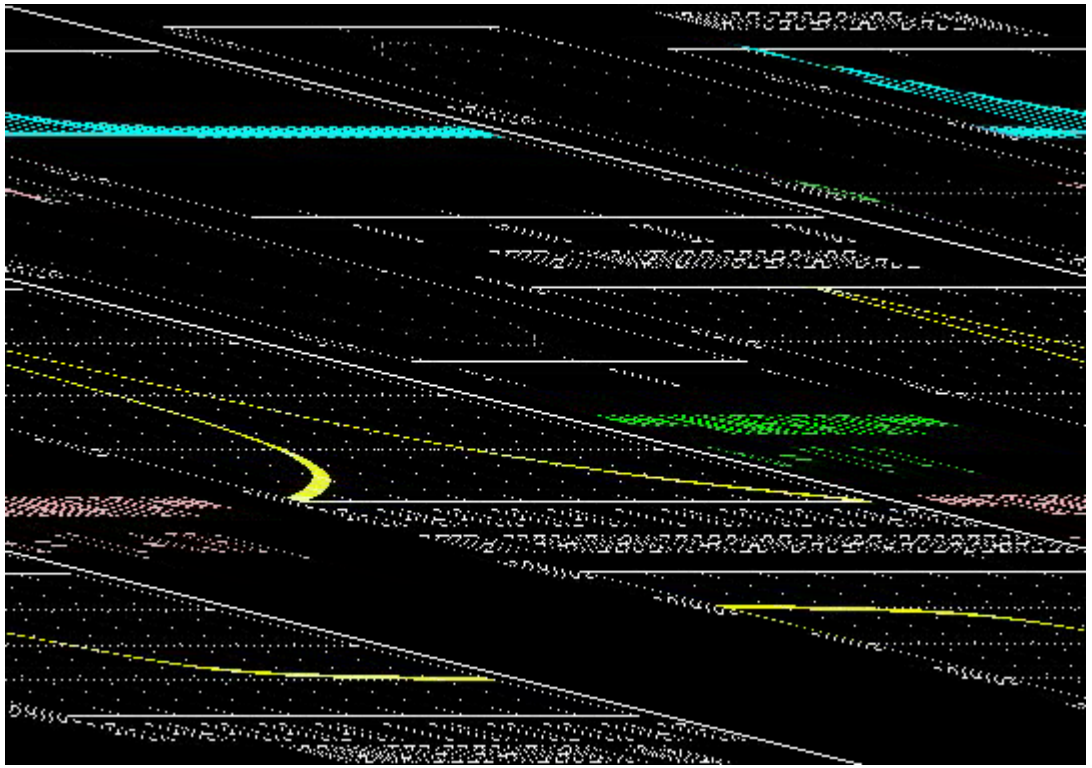
- Příklad: **horní propust**
  - $b_0 = 1$
  - $a_1 = 0.999...0$
  - zero $z = 0$
  - pole $z = -1$





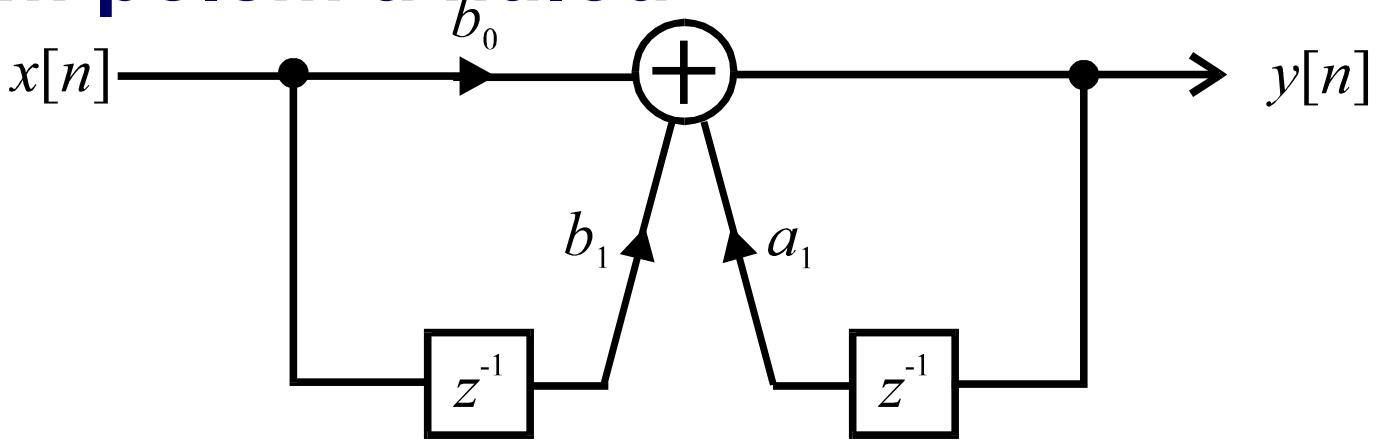
Jednoduché číslicové filtry – IIR(1)

- **Nula a pól pohybující se podél reálné osy** – animace



Jednoduché číslicové filtry – IIR(1)

- **IIR s jedním pólem a nulou**



Diferenční rovnice: $$y[n] = b_0 x[n] + b_1 x[n-1] + a_1 y[n-1]$$

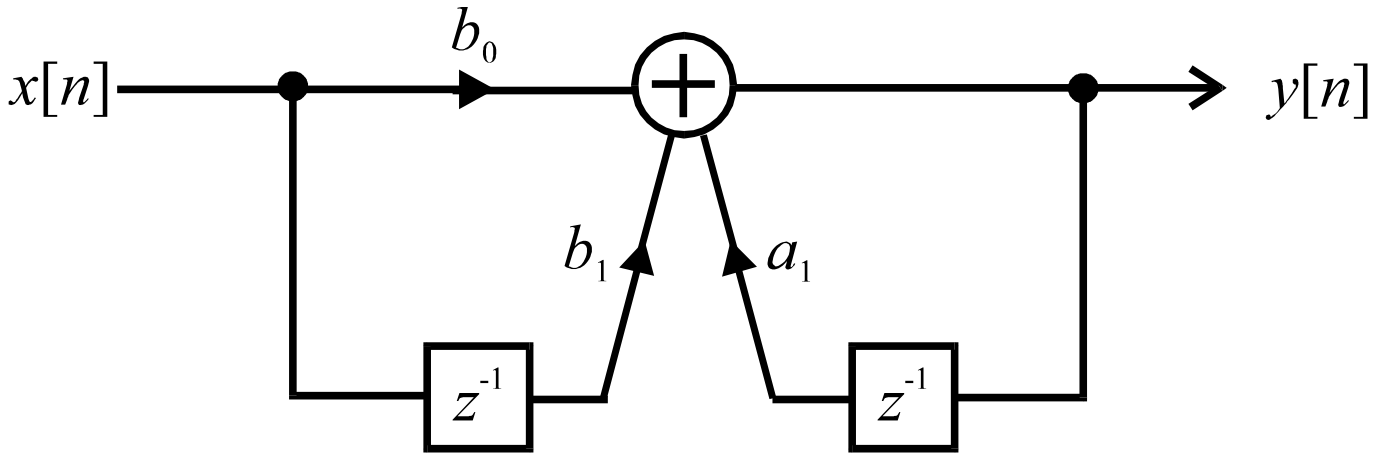
Z-transformace: $$Y(z) = b_0 X(z) + b_1 z^{-1} X(z) + a_1 z^{-1} Y[z]$$

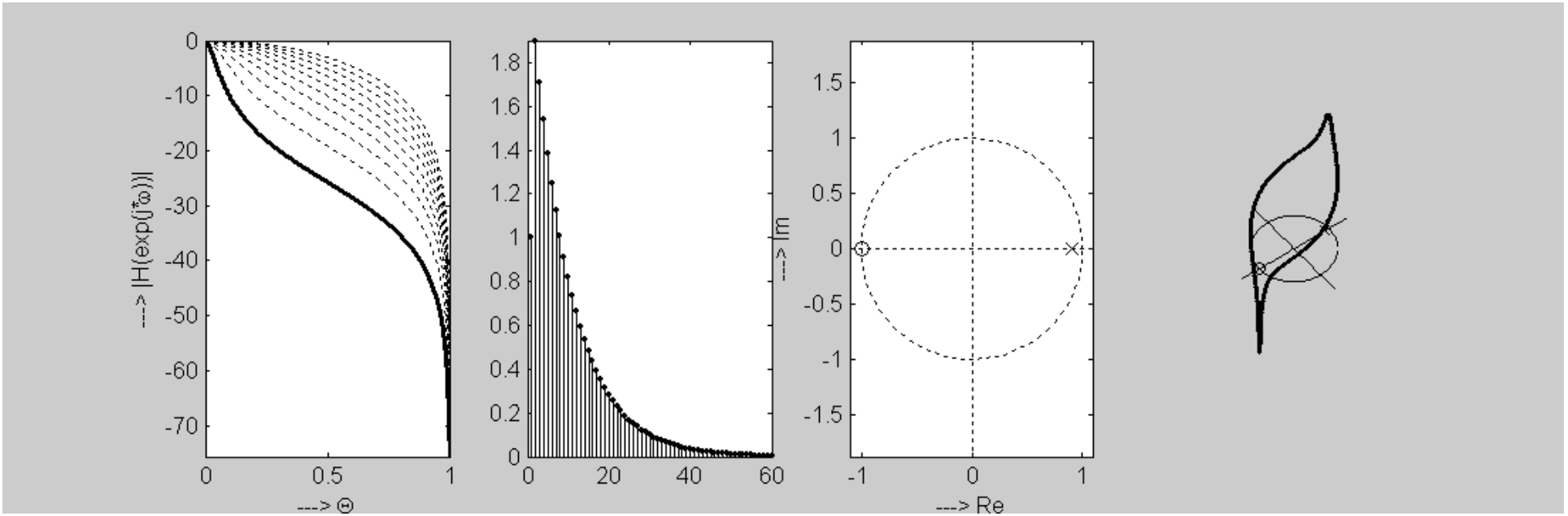
Přenosová funkce: $$H(z) = \frac{b_0 + b_1 z^{-1}}{1 - a_1 z^{-1}} = \frac{b_0 z + b_1}{z - a_1}$$

Frekvenční charakteristika: $$H(e^{j\theta}) = \frac{b_0 + b_1 e^{-j\theta}}{1 - a_1 e^{-j\theta}}$$

Jednoduché číslicové filtry – IIR(1)

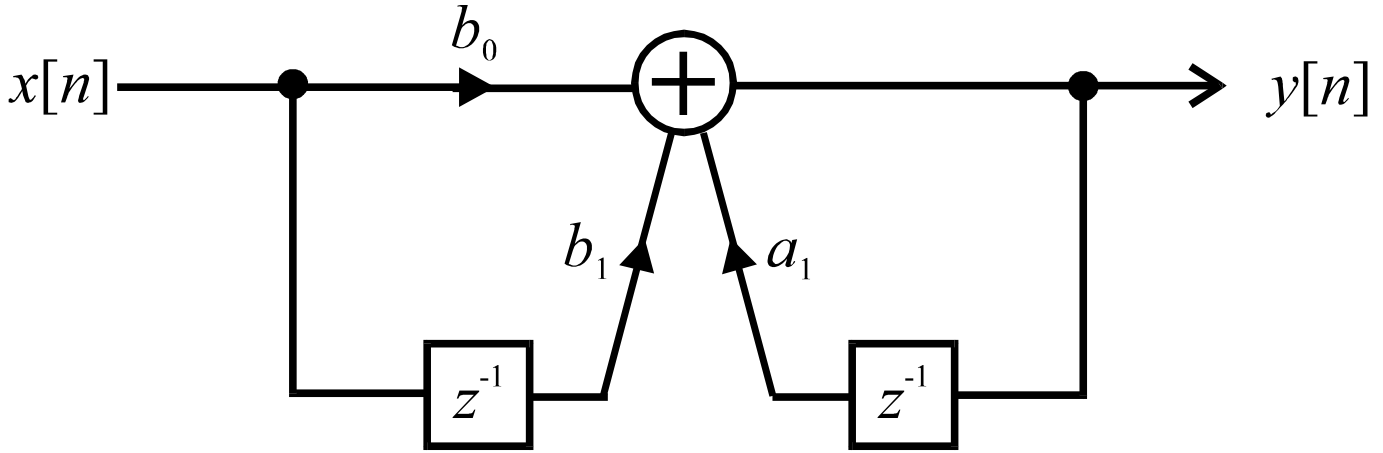
- Příklad: **Dolní propust**
  - $b_0 = 1$
  - $b_1 = 1$
  - $a_1 = -0.9...0$

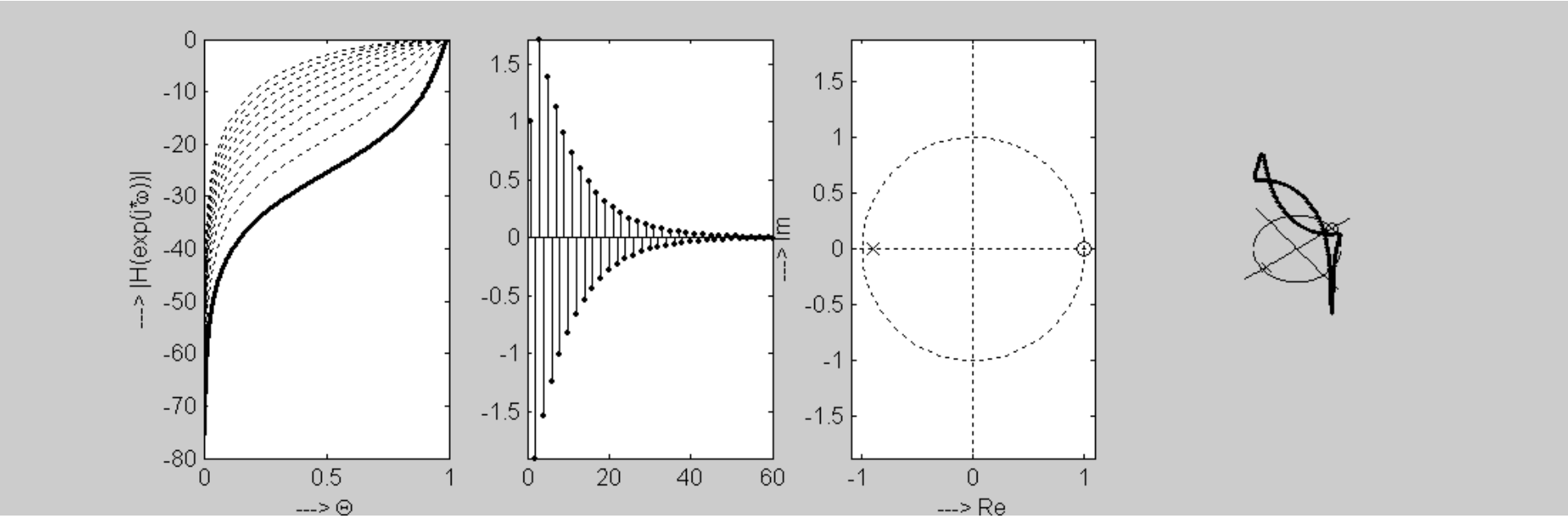




Jednoduché číslicové filtry – IIR(1)

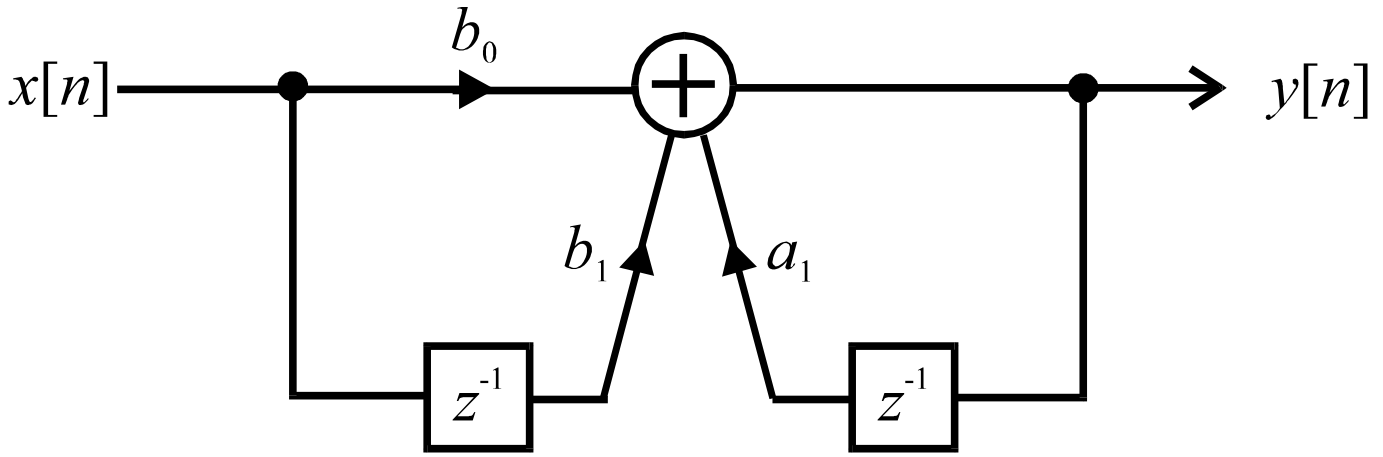
- Příklad: **Horní propust**
  - $b_0 = 1$
  - $b_1 = -1$
  - $a_1 = 0.9...0$

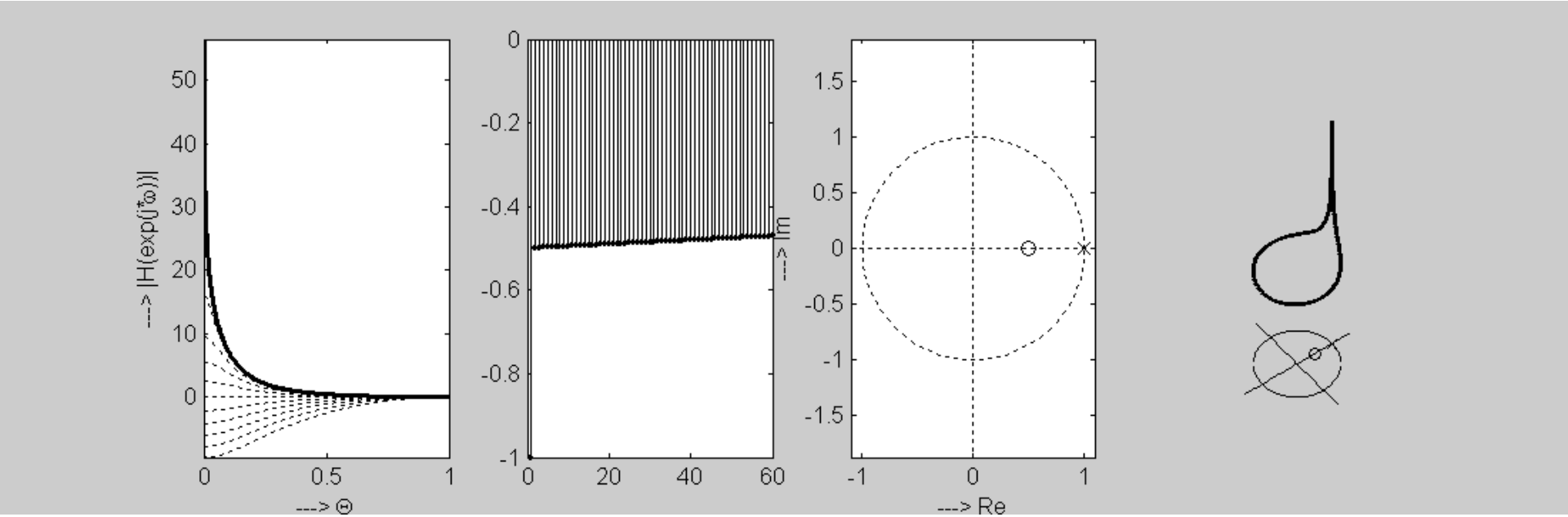




Jednoduché číslicové filtry – IIR(1)

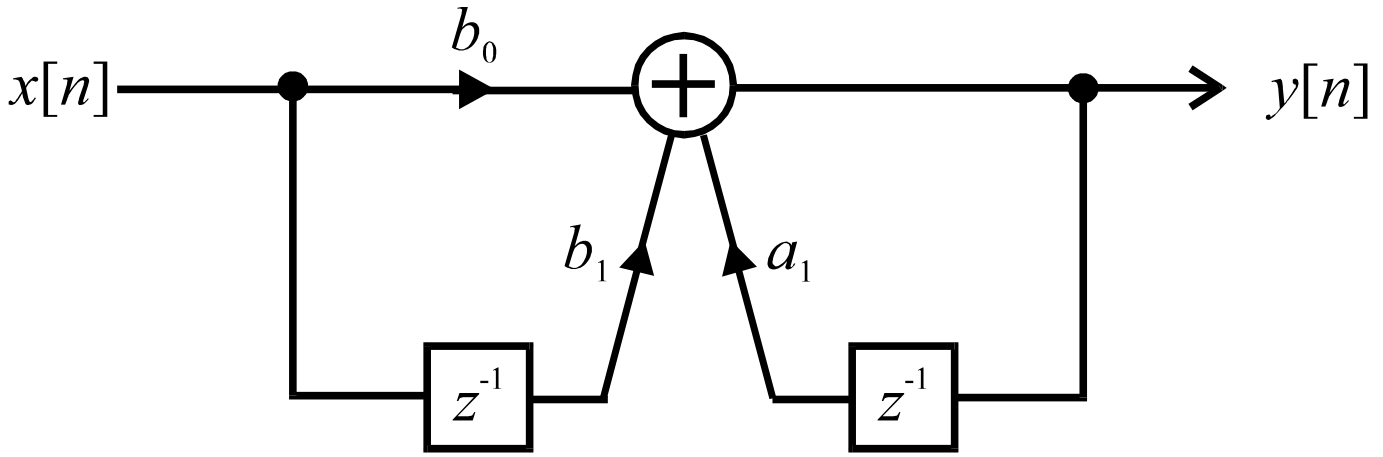
- Příklad: **ovládání basů**
  - b_0 = -1
  - b_1 = 0.5
  - a_1 = -0.999...0

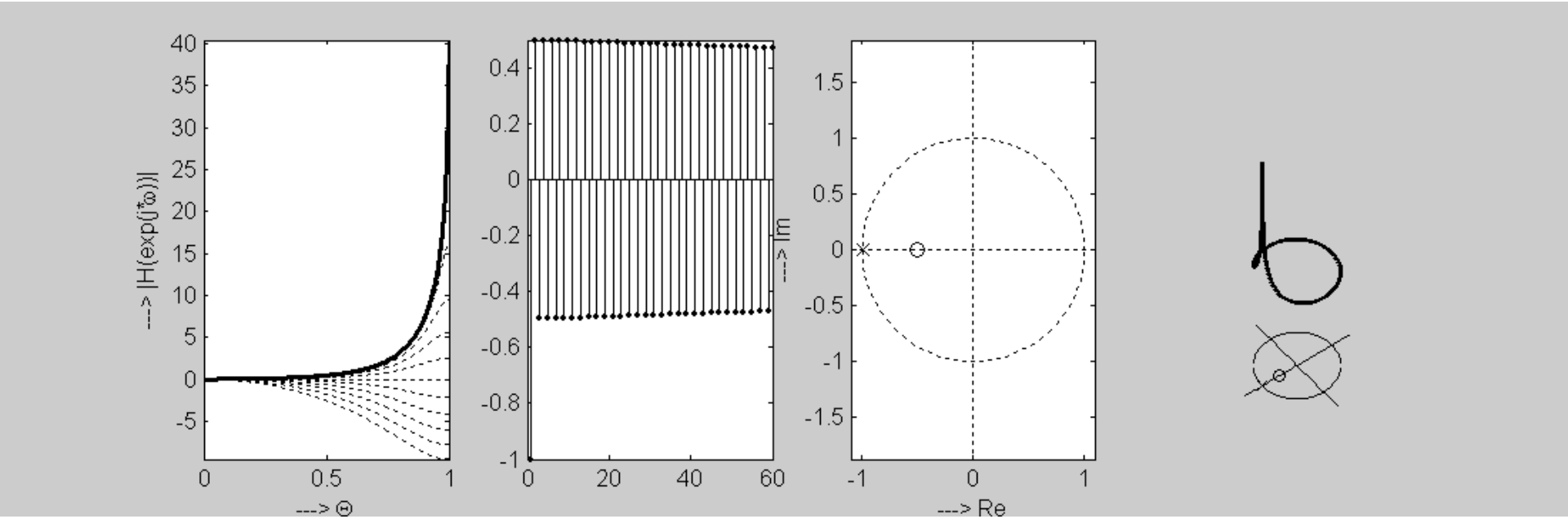




Jednoduché číslicové filtry – IIR(1)

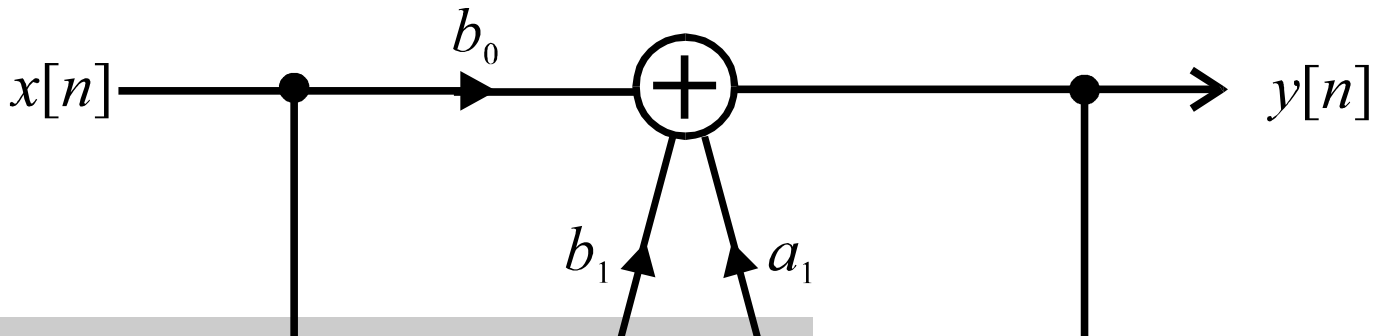
- Příklad: **ovládání výšek**
  - b_0 = -1
  - b_1 = 0.5
  - a_1 = 0.999...0

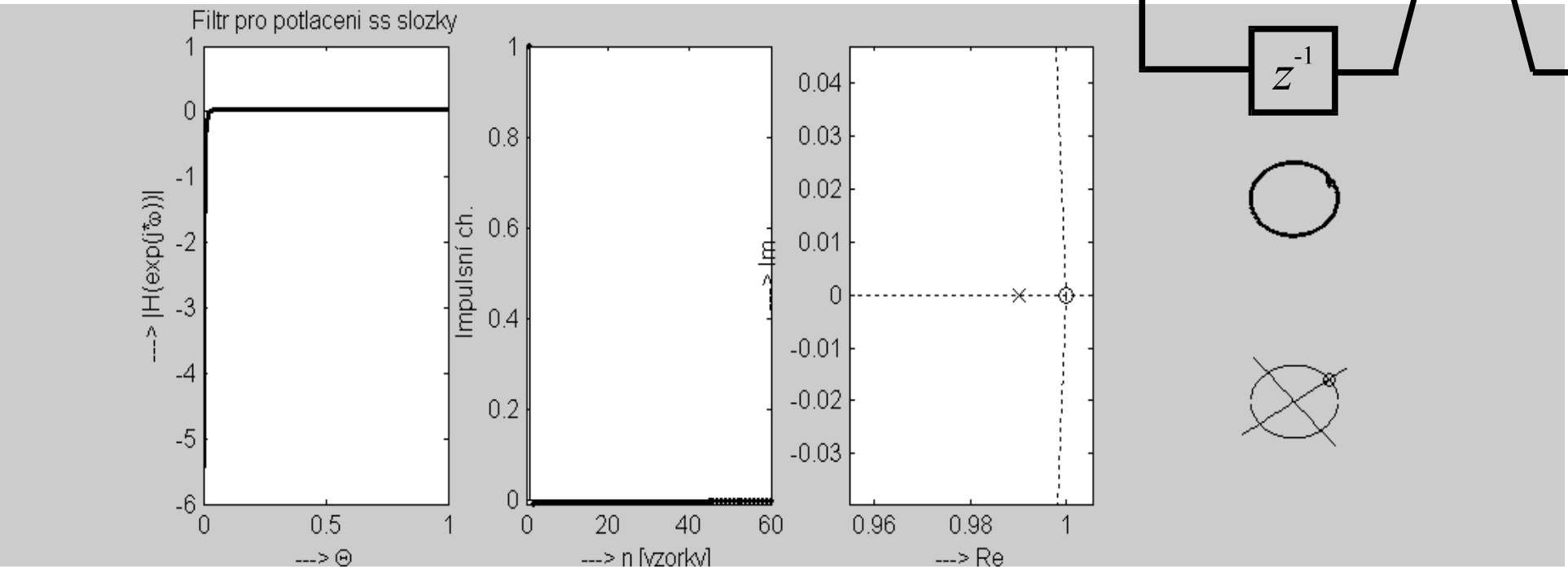




Jednoduché číslicové filtry – IIR(1)

- Příklad: **Potlačení stejnosměrné složky**
  - $b_0 = 1$
  - $b_1 = -1$
  - $a_1 = -0.99$





FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Jednoduché číslicové filtry – IIR(1)

- **Praktické aspekty IIR filtrů prvního řádu**
  - **Účinnost**: IIR filtry se často používají v aplikacích zpracování v reálném čase (vyžadují méně výpočtů ve srovnání s FIR).
  - **Fázové zkreslení**: Jednou z nevýhod IIR filtrů je fázové zkreslení, zejména pokud filtr není pečlivě navržen, což může ovlivnit kvalitu signálu.
  - **Aplikace**: IIR filtry se široce používají ve zpracování zvuku, včetně ekvalizérů, systémů potlačení šumu a řídicích systémů založených na zpětné vazbě (jako jsou adaptivní filtry).
  - **Výhody**: IIR filtry dosahují ostřejšího přechodu s menším počtem koeficientů, což je činí efektivnějšími pro úkoly, jako je dolní propust nebo pásmová propust.

Číslicová filtrace v MATLABu

- **Standardní tvar diferenční rovnice:**

$$a_0 y[n] + a_1 y[n-1] + \ldots = b_0 x[n] + b_1 x[n-1] \ldots$$

- Tato rovnice představuje lineární časově invariantní (LTI) systém, kde $x[n]$ je vstupní signál a $y[n]$ je výstupní signál.

```
>> y = filter(b, a, x);
```

Číslicová filtrace v MATLABu

- **Základní syntaxe v MATLABu**
  - Pro návrh a analýzu digitálních filtrů použijte v MATLABu následující příkazy:

```
>> y = filter(b, a, x);              % Filtrace
>>
>>
>>
>>
>>
>>
>>
>>
>>
```

Číslicová filtrace v MATLABu

- **Základní syntaxe v MATLABu**
  - Pro návrh a analýzu digitálních filtrů použijte v MATLABu následující příkazy:

```
>> y = filter(b, a, x);          % Filtrace
>> impz(b, a);                   % Impulzní odezva
>>
>>
>>
>>
>>
>>
>>
```

Číslicová filtrace v MATLABu

- **Základní syntaxe v MATLABu**
  - Pro návrh a analýzu digitálních filtrů použijte v MATLABu následující příkazy:

```
>> y = filter(b, a, x);          % Filtrace
>> impz(b, a);                   % Impulzní odezva
>> impz(b, a, N);                % Impulzní odezva o délce N
>>
>>
>>
>>
>>
>>
```

Číslicová filtrace v MATLABu

- **Základní syntaxe v MATLABu**
  - Pro návrh a analýzu digitálních filtrů použijte v MATLABu následující příkazy:

```
>> y = filter(b, a, x);          % Filtrace
>> impz(b, a);                   % Impulzní odezva
>> impz(b, a, N);                % Impulzní odezva o délce N
>> freqz(b, a);                  % Frekvenční charakteristika
>>
>>
>>
>>
>>
```

Číslicová filtrace v MATLABu

- **Základní syntaxe v MATLABu**
  - Pro návrh a analýzu digitálních filtrů použijte v MATLABu následující příkazy:

```
>> y = filter(b, a, x);          % Filtrace
>> impz(b, a);                   % Impulzní odezva
>> impz(b, a, N);                % Impulzní odezva o délce N
>> freqz(b, a);                  % Frekvenční charakteristika
>> freqz(b, a, N);               % Frekvenční charakteristika s N body
>>
>>
>>
>>
```

Číslicová filtrace v MATLABu

- **Základní syntaxe v MATLABu**
  - Pro návrh a analýzu digitálních filtrů použijte v MATLABu následující příkazy:

```
>> y = filter(b, a, x);          % Filtrace
>> impz(b, a);                   % Impulzní odezva
>> impz(b, a, N);                % Impulzní odezva o délce N
>> freqz(b, a);                  % Frekvenční charakteristika
>> freqz(b, a, N);               % Frekvenční charakteristika s N body
>> freqz(b, a, N, fs);           % Frekvenční charakteristika s fs
>>
>>
>>
```

Číslicová filtrace v MATLABu

- **Základní syntaxe v MATLABu**
  - Pro návrh a analýzu digitálních filtrů použijte v MATLABu následující příkazy:

```
>> y = filter(b, a, x);          % Filtrace
>> impz(b, a);                   % Impulzní odezva
>> impz(b, a, N);                % Impulzní odezva o délce N
>> freqz(b, a);                  % Frekvenční charakteristika
>> freqz(b, a, N);               % Frekvenční charakteristika s N body
>> freqz(b, a, N, fs);           % Frekvenční charakteristika s fs
>> zplane(b, a);                 % Zero-pole graf
>>
>>
```

Číslicová filtrace v MATLABu

- **Základní syntaxe v MATLABu**
  - Pro návrh a analýzu digitálních filtrů použijte v MATLABu následující příkazy:

```
>> y = filter(b, a, x);          % Filtrace
>> impz(b, a);                   % Impulzní odezva
>> impz(b, a, N);                % Impulzní odezva o délce N
>> freqz(b, a);                  % Frekvenční charakteristika
>> freqz(b, a, N);               % Frekvenční charakteristika s N body
>> freqz(b, a, N, fs);           % Frekvenční charakteristika s fs
>> zplane(b, a);                 % Zero-pole graf
>> [z, p, k] = tf2zp(b, a);      % Převod přenosové funkce na tvar zero-pole
>>
```

Číslicová filtrace v MATLABu

- **Základní syntaxe v MATLABu**
  - Pro návrh a analýzu digitálních filtrů použijte v MATLABu následující příkazy:

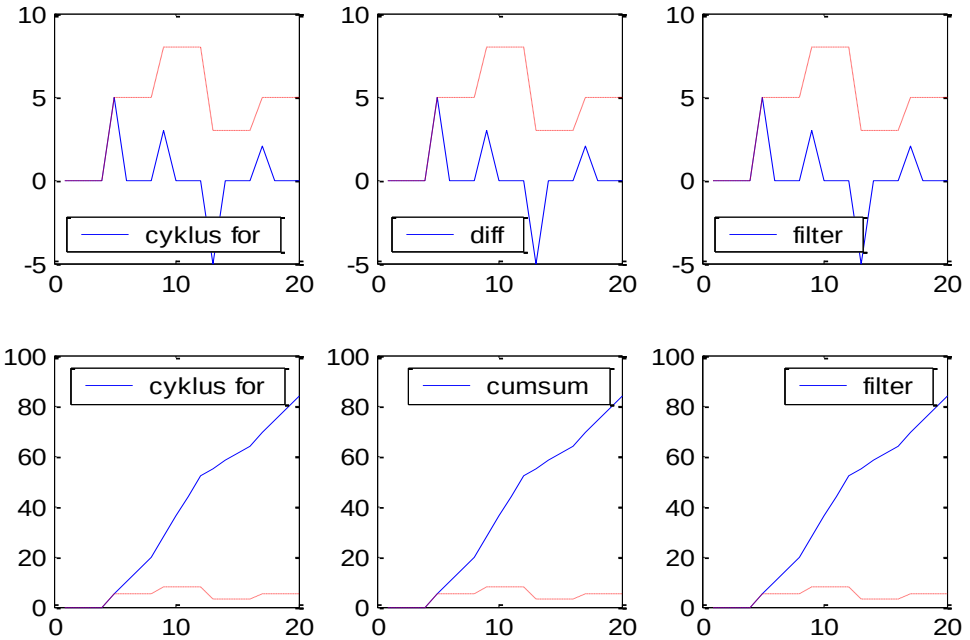
```
>> y = filter(b, a, x);          % Filtrace
>> impz(b, a);                   % Impulzní odezva
>> impz(b, a, N);                % Impulzní odezva o délce N
>> freqz(b, a);                  % Frekvenční charakteristika
>> freqz(b, a, N);               % Frekvenční charakteristika s N body
>> freqz(b, a, N, fs);           % Frekvenční charakteristika s fs
>> zplane(b, a);                 % Zero-pole graf
>> [z, p, k] = tf2zp(b, a);      % Převod přenosové funkce na tvar zero-pole
>> [b, a] = zp2tf(z, p, k);      % Převod zero-pole tvaru na přenosovou funkci
```

Číslicová filtrace v MATLABu

- **Příklad 5.1: Simulace diferenciátorů a integrátorů**
  - Implementujte diferenciátor a integrátor různými způsoby.
  - MATLAB:

```
% Example code to calculate differentiation and integration
x = [0 0 0 0 5 5 5 5 8 8 8 8 3 3 3 3 5 5 5 5];
for n = 2:length(x)
      y_dif(n) = x(n) - x(n-1);
      y_int(n) = x(n) + y_int(n-1);
end
% MATLAB built-in methods: diff and cumsum
```

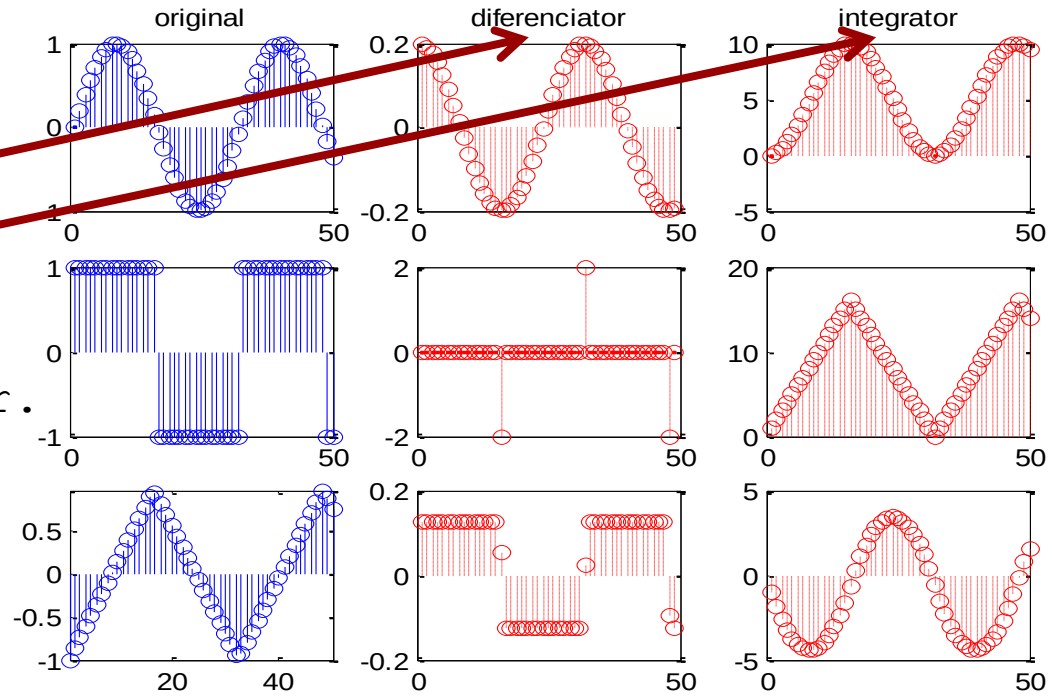
$y[n] = x[n] - x[n-1]$ $\quad$ $y[n] = x[n] + y[n-1]$



Číslicová filtrace v MATLABu

- **Příklad 5.2: Chování diferenciátorů a integrátorů**
  - Aplikujte diferenciátor a integrátor na sinusový, obdélníkový a pilovitý průběh.
  - MATLAB:

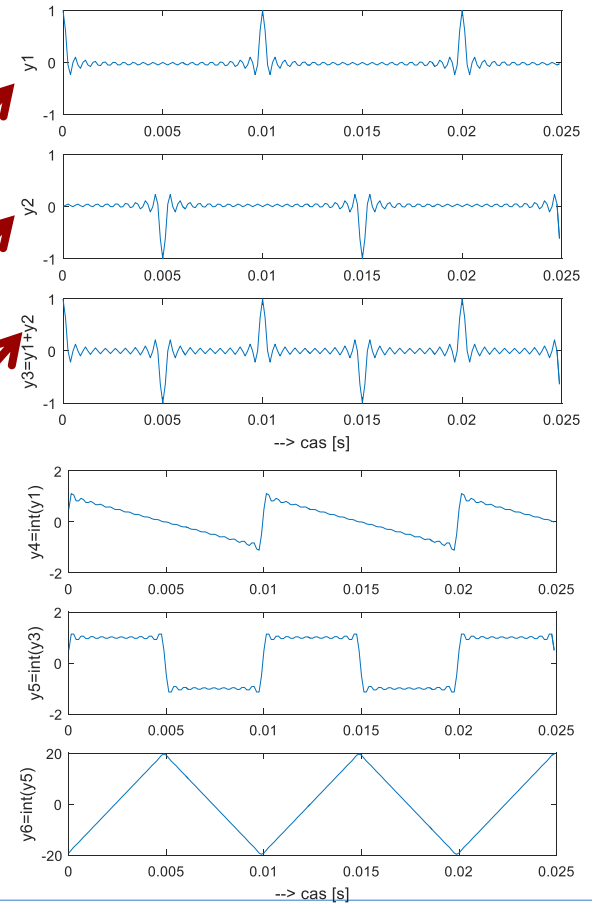
```
x1 = sin(0:0.2:10-0.2);
x2 = square(0:0.2:10-0.2);
x3 = sawtooth(0:0.2:10-0.2, 0.5);
% Use diff and cumsum to analyze the behavior.
```



Číslicová filtrace v MATLABu

- **Příklad 5.3: Generování pásmově omezených periodických signálů**
  - Generujte průběhy s využitím integrátoru.
  - MATLAB:

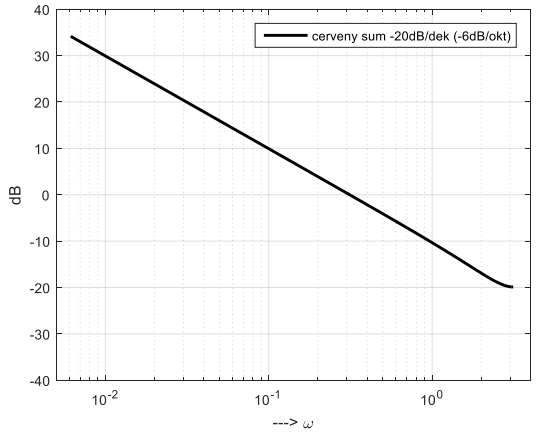
```
% Create band-limited periodic signals using integration.
N = floor(fs/(2*f0));
fs = 8000;
t = 0:1/fs:2-1/fs;
f0 = 100;
y1 = ([1.^k] * cos(k' * 2 * pi * f0 * t)) / N;
y2 = (-[(-1).^k] * cos(k' * 2 * pi * f0 * t)) / N;
y3 = y1 + y2;
% Further integration for waveform creation.
```

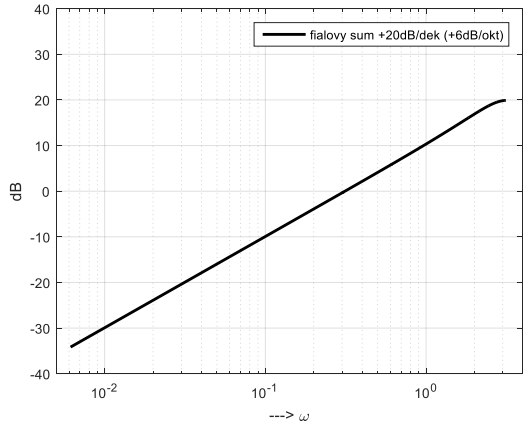


Číslicová filtrace v MATLABu

- **Příklad 5.4: Generování barevného šumu**
  - Generujte červený a fialový šum pomocí filtrace bílého šumu.
  - MATLAB:

```
white_noise = randn(1, fs * cas);
red_noise = filter([1 0], [1 -1], white_noise);
purple_noise = filter([1 -1], [1 0], white_noise);
```

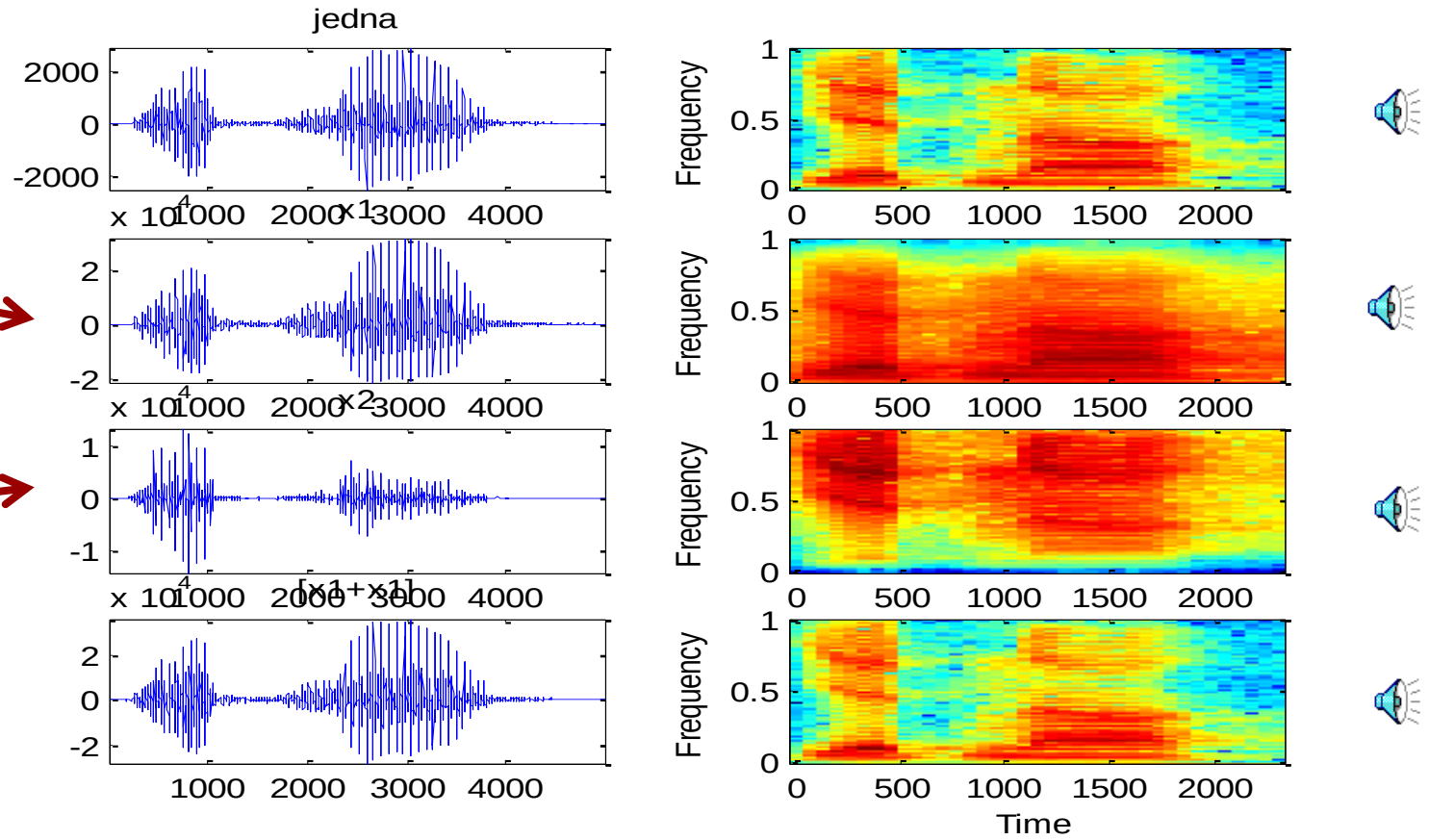




Číslicová filtrace v MATLABu

- **Příklad 5.5: Filtrace řečového signálu**
  - Aplikujte dolní a horní propust na řečový signál
  - MATLAB:

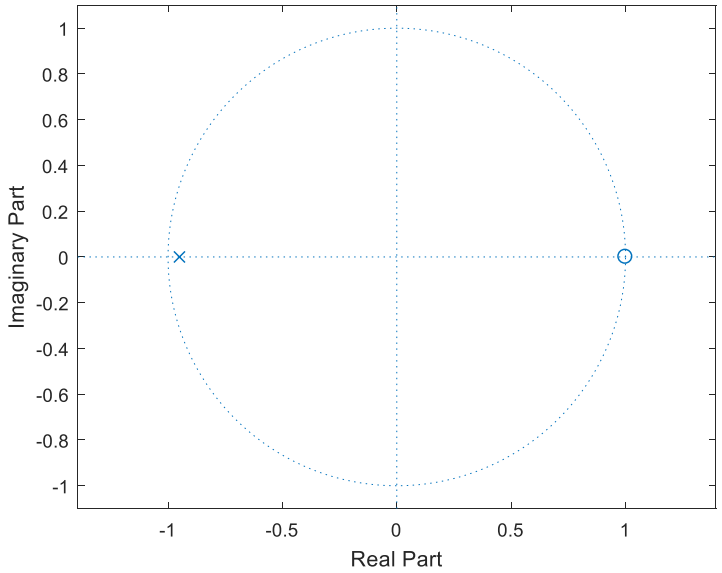
```
% Low-pass filter
b_lp = [1 1];
a_lp = [1 -0.9];

% High-pass filter
b_hp = [1 -1];
a_hp = [1 0.9];
```

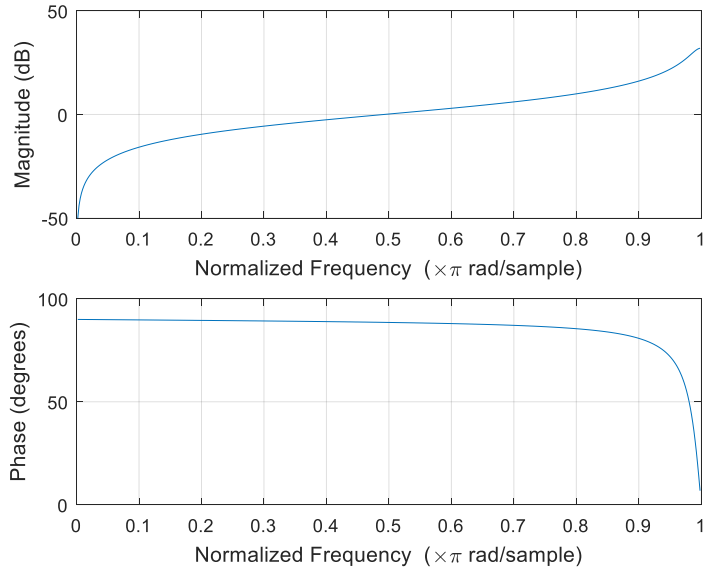


Číslicová filtrace v MATLABu

- **Příklad 5.6: Filtrace hlasu pomocí jednopólového filtru**
  - Analyzujte vliv různých koeficientů na filtraci řeči.
  - MATLAB:

```
b_filt = [1 -1]; a_filt = [1 0.95];
% Vary the filter coefficient and examine the frequency response.
```

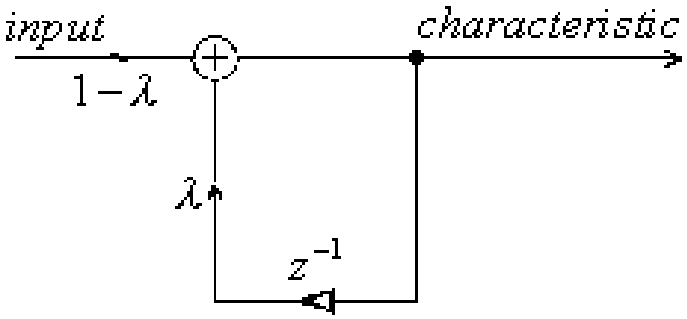


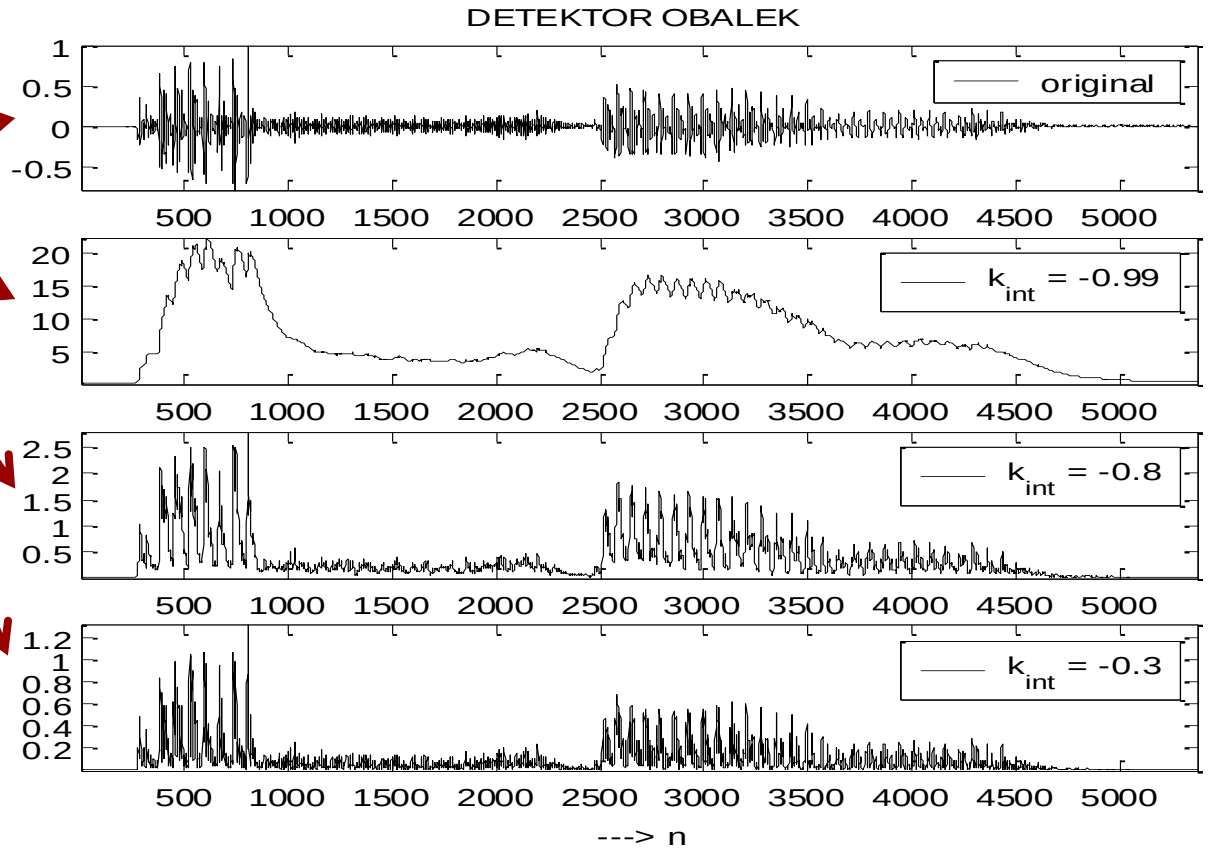


Číslicová filtrace v MATLABu

- **Příklad 5.7: Detektor obálek**
  - Navrhněte detektor obálky s integrátorem.
  - MATLAB:

```
xc = osum1 ./ max(abs(osum1));
xi = filter(1, [1 -0.99], abs(xc));
xi = filter(1, [1 -0.8], abs(xc));
xi = filter(1, [1 -0.3], abs(xc));
```

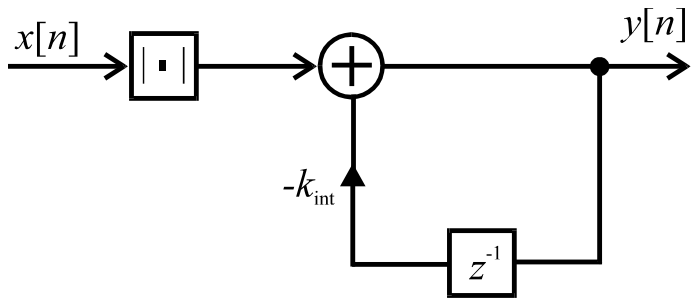


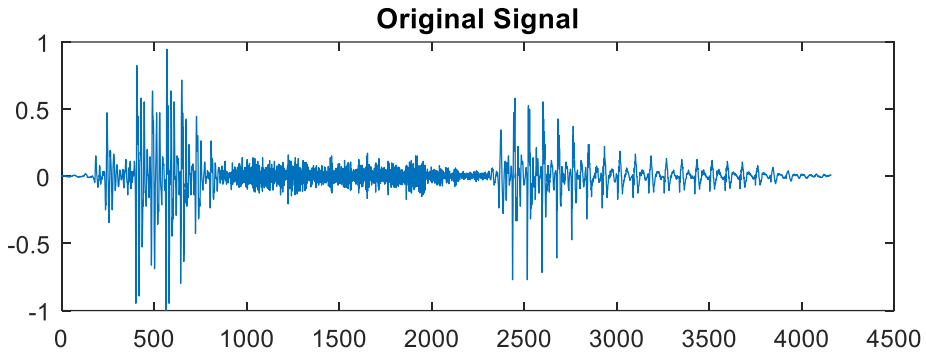


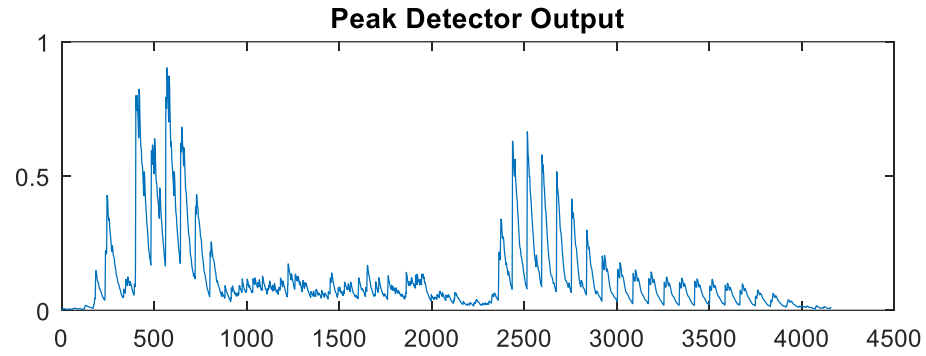
Číslicová filtrace v MATLABu

- **Příklad 5.8: Detektor špiček**
  - Implementujte detektor reagující na špičky v signálu.
  - MATLAB:

```
if abs(x(n)) > y(n-1)
        k_int = 0.3;
else
        k_int = 0.95;
end
k = 1 - k_int;
y(n) = k * x(n) + k_int * y(n-1);
```



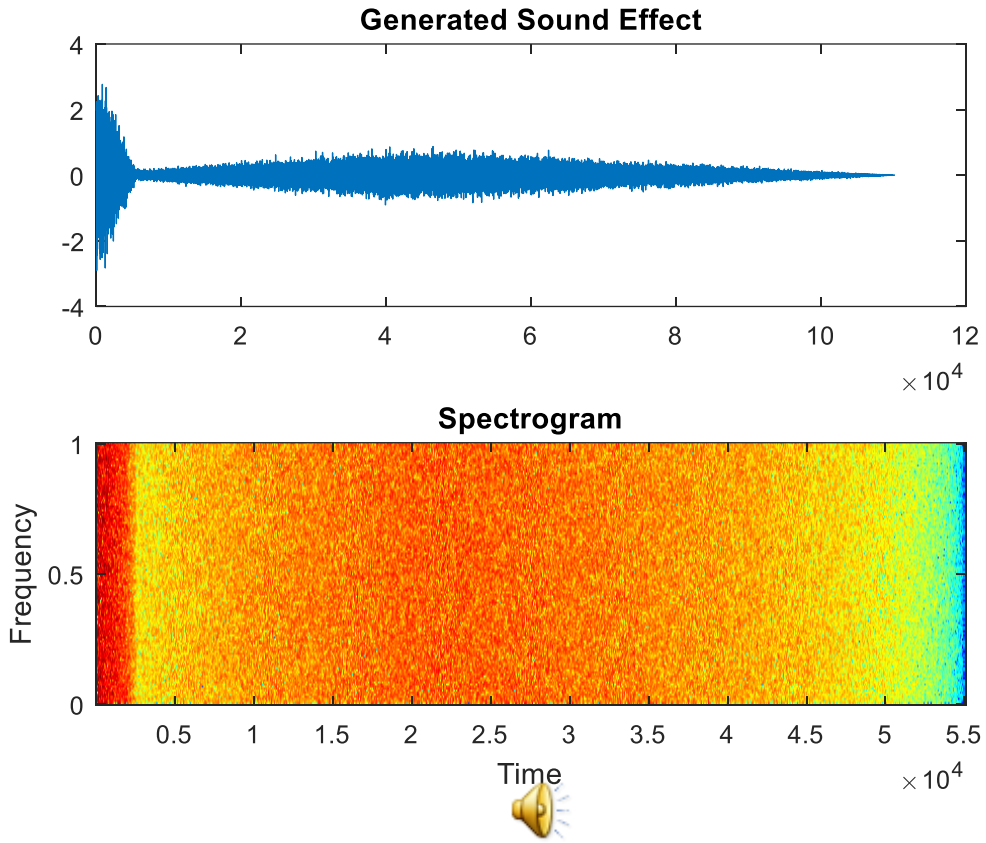




Číslicová filtrace v MATLABu

- **Příklad 5.9: Zvukový efekt založený na šumu s obálkou**
  - Navrhněte zvukový efekt pomocí šumu a tvarovací obálky.
  - MATLAB:

```
x = randn(1, duration * fs);

X = [0 0.05 0.4 1];
Y = [1 0.05 0.25 0];
O = interp1(X, Y, t ./ t(end));

sig = x .* O;
```

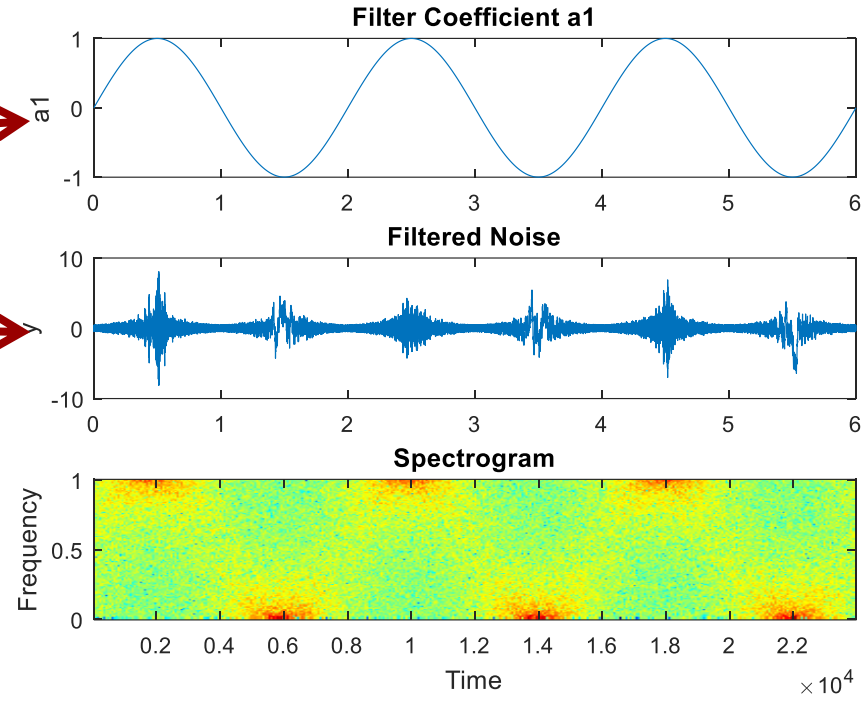


FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE

Číslicová filtrace v MATLABu

- **Příklad 5.10: Simulace ladění rádia**
  - Simulujte efekt ladění rádia.
  - MATLAB:

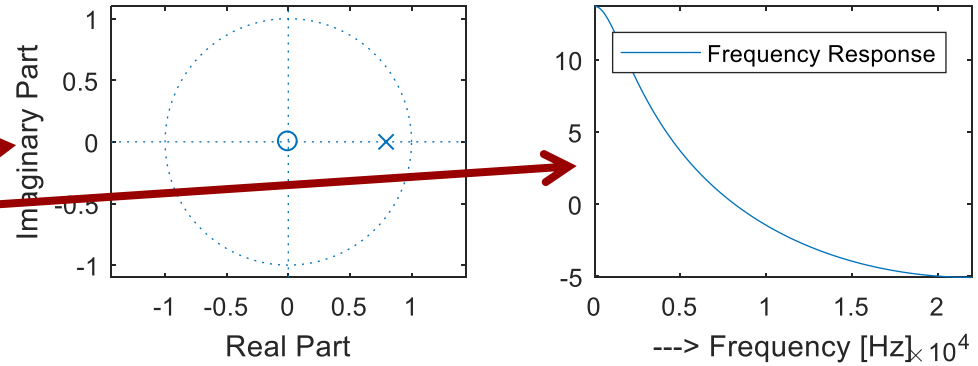
```
for n = 2:length(x)
    a1(n) = 0.999 * (sin(2 * pi * fm * nT(n)));
    y(n) = x(n) - a1(n) * y(n-1);
end
```

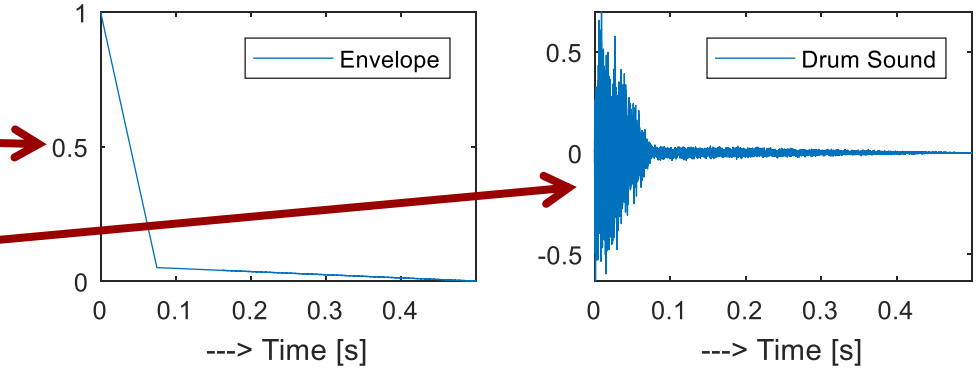


Číslicová filtrace v MATLABu

- **Příklad 5.11: Zvuk bicích s využitím šumu a obálky**
  - Generujte zvuk bicích kombinací šumu a obálky.
  - MATLAB:

```
b = 1;
a = [1 -0.7934];
y = filter(b, a, x);

X = [0 0.15 1];
Y = [1 0.05 0];
O = interp1(X, Y, nT/nT(end));

O .* y
```

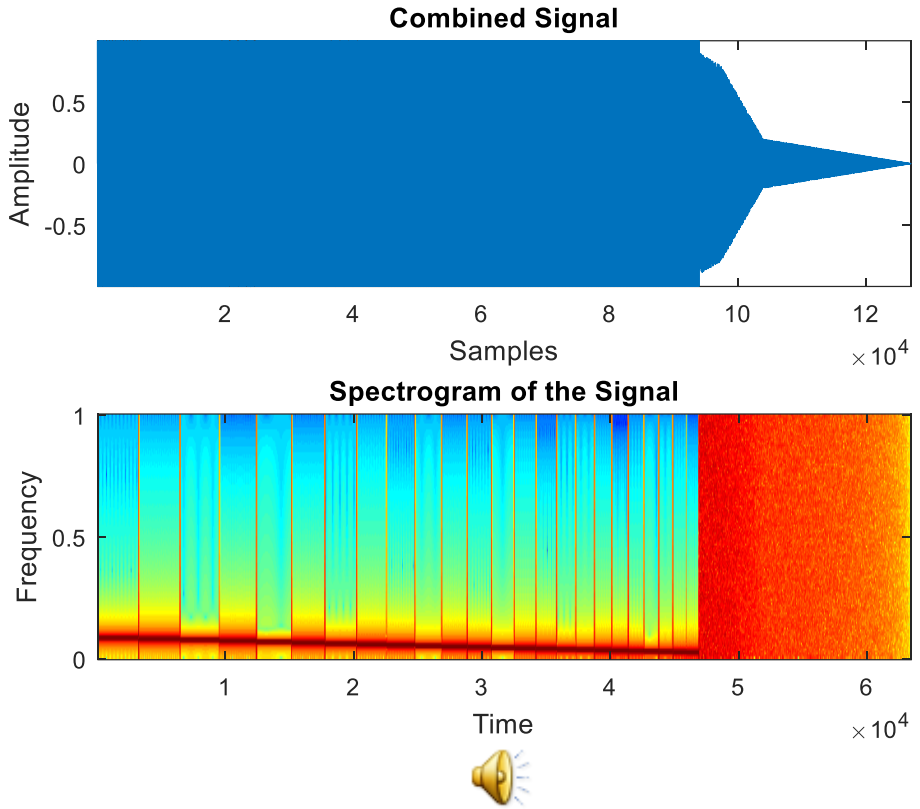




Číslicová filtrace v MATLABu

- **Příklad 5.12: Zvukový efekt padajícího tónu**
  - Vytvořte dojem klesajících tónů a následného dopadu.
  - MATLAB:

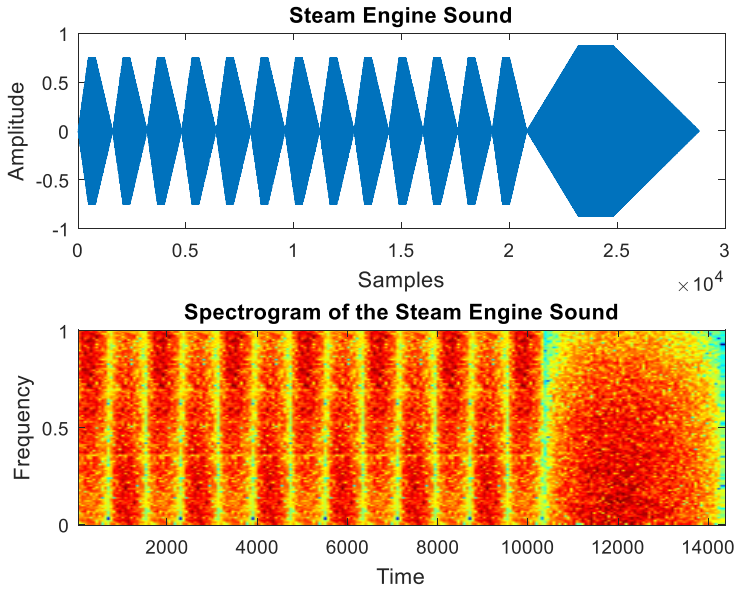
```
% Generating tones with decreasing frequency
K = 24; % Number of frequency changes

% Logarithmic frequency decrease
dF = logspace(log10(880), log10(220), K);
% Time intervals for tones
dT = logspace(log(0.6), log(0.35), K);
```



Číslicová filtrace v MATLABu

- **Příklad 5.13: Zvuk parního stroje**
  - Simulujte zvuk parního stroje pomocí filtrace a obálek.
  - MATLAB:

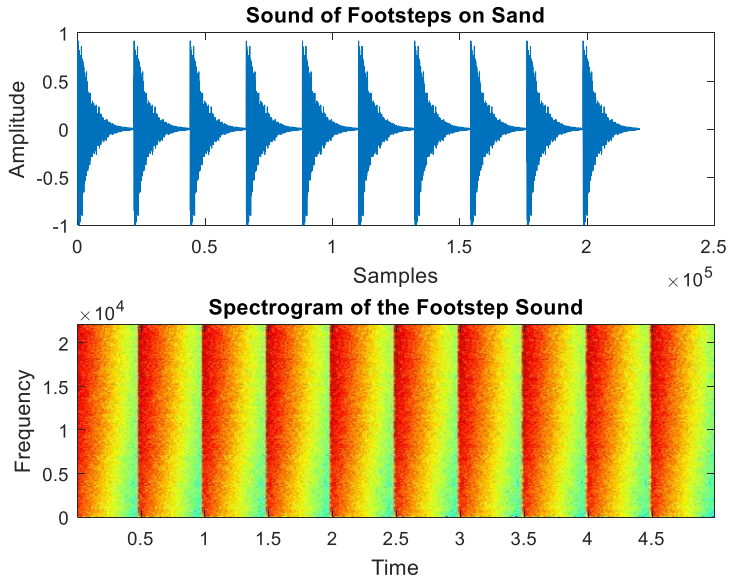
```
% First filter
f1 = filter([1 -0.5], 1, x);
% Second filter
f2 = filter([1 0.5], 1, x);

% Create envelope for the sound
o1 = interp1([0 0.3 0.5 1] * t(end), [0 1 1 0], t);
y1 = o1 .* f1; % Modulate the first filtered signal
y2 = o1 .* f2; % Modulate the second filtered signal
```



Číslicová filtrace v MATLABu

- **Příklad 5.14: Kroky v písku nebo štěrku**
  - Simulujte zvuk kroků pomocí filtračníní syntézy a tvarování šumu.
  - MATLAB:

```
% Generate white noise as basic step sound
white_noise = randn(1, fs * duration);

% Filter noise with FIR filter to simulate step sound
b = [1 -0.5]; % FIR filter
step_sound = filter(b, 1, white_noise); % Apply filter
```



MIDI v MATLABu

- **Úvod do MIDI**
  - **Co je MIDI?**
    - **MIDI** je zkratka pro **M**usical **I**nstrument **D**igital Interface
    - Je to standardizovaný protokol pro komunikaci mezi elektronickými hudebními nástroji, počítači a audio systémy.
    - MIDI umožňuje hudebním nástrojům a zařízením komunikovat odesíláním dat o hře, nikoli však samotný zvuk.
  - **Klíčové vlastnosti MIDI:**
    - **Kompaktní velikost souboru:** Soubory MIDI ukládají pouze instrukce, takže jsou velmi malé.
    - **Všestrannost:** Podporuje širokou škálu nástrojů a zvuků.
    - **Flexibilita:** Snadná úprava nebo přeskupení her.
  - **Jak MIDI funguje:**
    - MIDI data functions like a musical score: it tells the instrument how to play, but not what the sound should be.
    - A synthesizer or sound card interprets the data to produce sound.

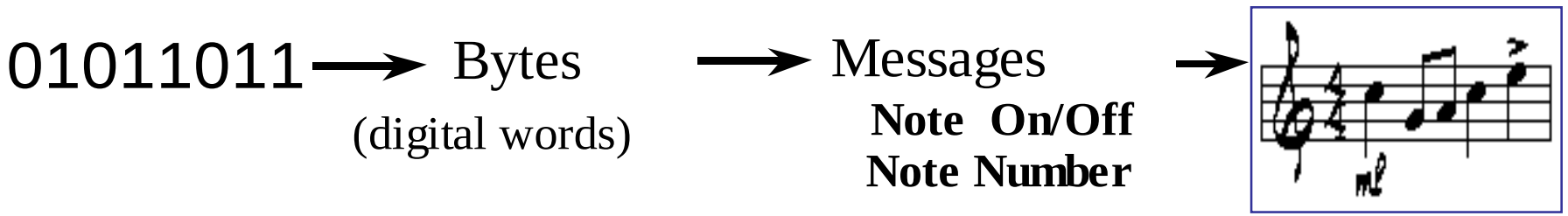
MIDI v MATLABu MIDI v MATLABu

- **Historie MIDI a jeho moderní aplikace**
  - **Před MIDI (před 80. lety 20. století):**
    - No standardized way to connect electronic musical instruments.
    - Manufacturers used incompatible systems, limiting integration.
  - **Potřeba standardizace:**
    - Musicians required a universal system to connect and control multiple devices seamlessly.
  - **Vytvoření MIDI:**
    - Roland proposed a universal standard.
    - 1983: The first public MIDI demo successfully linked synthesizers from different manufacturers.
  - **Aplikace MIDI dnes:**
    - Widely used in digital music production, live performances, film scoring, and video game soundtracks.
    - Enables easy integration between hardware and software instruments.

MIDI v MATLABu

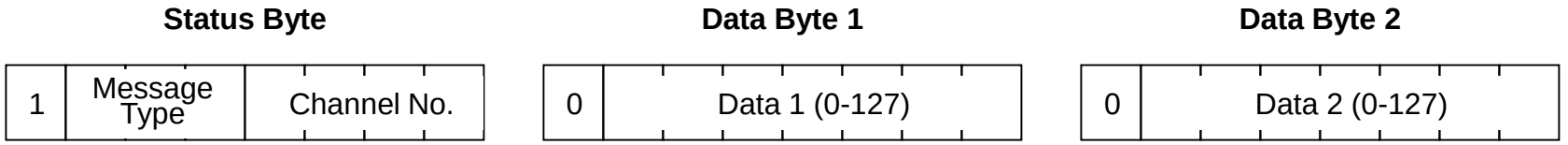
- **Komunikační protokol MIDI**

- Jak funguje MIDI:
  - Po stisknutí klávesy se do připojeného zařízení odešle zpráva *Note On*, která spustí generování zvuku.
  - Po uvolnění klávesy se odešle zpráva *Note Off*, která zastaví zvuk..
- Řídicí zprávy:
  - Zprávy *Control Change* upravují parametry, jako je typ nástroje, efekty (např. reverb) nebo jiné ovládací prvky výkonu (např. pedály).



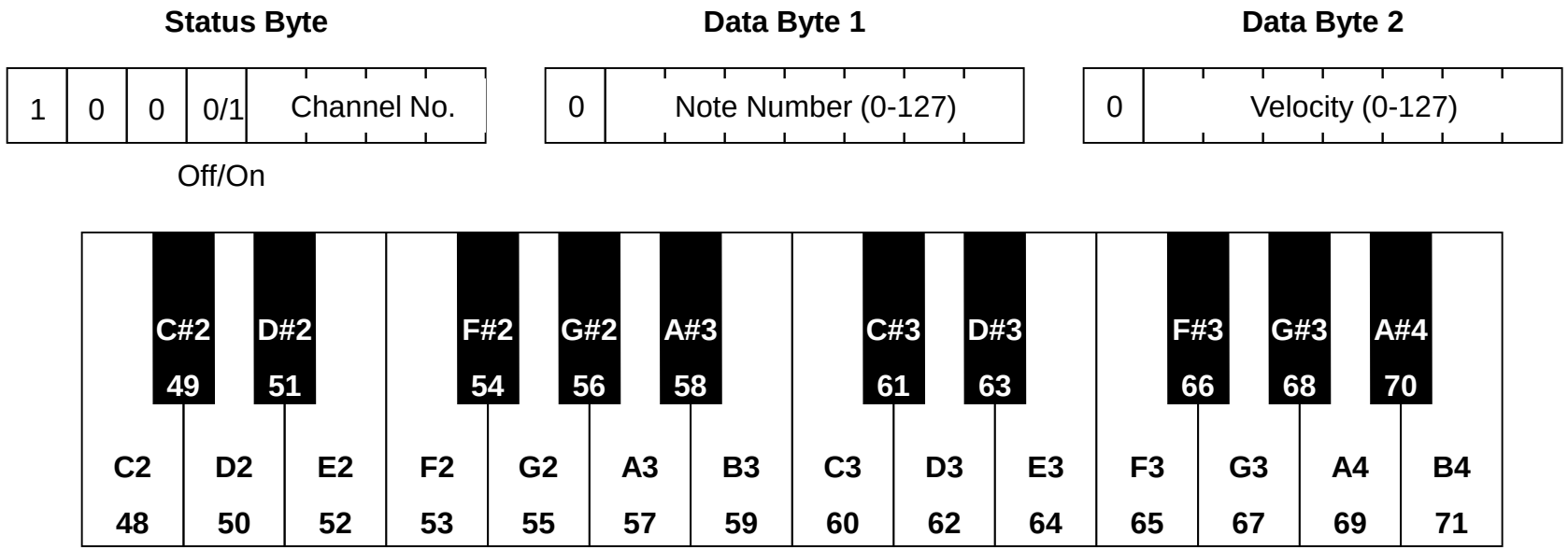
MIDI v MATLABu

- **Komunikační protokol MIDI – fyzická vrstva**
- Přenos dat MIDI:
  - MIDI je simplexní asynchronní sériové rozhraní.
    - **Simplexní:** Data tečou pouze jedním směrem.
    - **Sériové:** Datové bity se přenášejí jeden po druhém.
    - **Asynchronní:** Není odesílán žádný hodinový signál; obě zařízení se spoléhají na přesné načasování.
- Datové pakety:
  - Každá zpráva MIDI se skládá z 8 bitů, plus startovacího bitu a stop bitu.
  - Zprávy se přenášejí pevnou rychlostí 31 250 bitů za sekundu.



MIDI v MATLABu

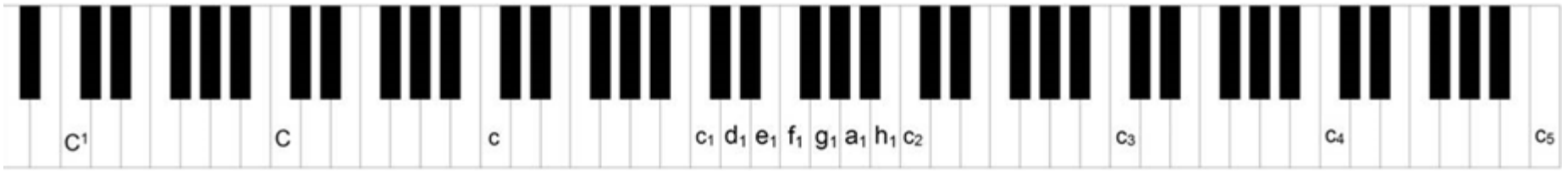
- **Běžné MIDI zprávy**
  - **Note On/Note Off:** Ovládá začátek a konec noty, přičemž rychlost ovlivňuje hlasitost.
  - **Polyphonic Aftertouch:** Odesílá informace o tlaku pro každou notu po jejím stisknutí.
  - **Control Change:** Používá se k úpravě efektů, ovládání pedálů nebo úpravě nastavení.
  - **Program Change:** Přepíná mezi přednastavenými zvuky (např. změna z piana na kytaru).
  - **Pitch Bend:** Mění výšku tónu noty.



MIDI v MATLABu

- **Konverze MIDI not**
  - Konverze mezi číslem MIDI not (*m* = 0 to 127) a frekvencí (*f* [Hz]):
    - ***m* = 69 + 12 x log2 (*f* / 440)** [MIDI na Hz]
    - ***f* = 440 x 2^ (m-69) / 12)** [Hz na MIDI].
  - Rozsah MIDI not od 8.18 Hz do 12,544 Hz.
    - Rozsah klavíru: **A0 = 27.5 Hz** (MIDI = **21**) to **C8 = 4186 Hz** (MIDI = **108**).

A0 C1 C2 C3 C4 C5 C6 C7 C8
21 60 69 108



MIDI v MATLABu

- **Standardní MIDI soubor (SMF)**
  - Co je standardní MIDI soubor?
    - MIDI sekvence lze ukládat a přehrávat na různých zařízeních pomocí formátu **standardního MIDI souboru** (SMF) format.
  - **Typy MIDI souborů:**
    - **Type 0:** Všechna data o hře jsou uložena na jedné stopě.
    - **Type 1:** Více stop je uloženo odděleně, ale přehráváno současně.
    - **Type 2:** Ukládá více aranžmá na samostatné stopy, přičemž každé se přehrává postupně (používá se zřídka).

MIDI v MATLABu

- **Specifikace General MIDI (GM)**
  1. **Mapa nastavení nástrojů (Instrument Patch Map):**
     - 128 přednastavených zvuků seskupených do 16 rodin nástrojů, každá s 8 variacemi.
       https://jazz-soft.net/demo/GeneralMidi.html
  2. **Mapa kláves perkusí (Percussion Key Map – Channel 10):**
     - Kanál 10 je vyhrazen pro perkusní nástroje, přičemž každá klávesa je namapována na specifický zvuk bicích..
       https://jazz-soft.net/demo/GeneralMidiPerc.html
  3. **Další specifikace:**
     - Pro komunikaci je k dispozici 16 MIDI kanálů..
     - MIDI nota číslo 60 představuje střední C (C4).

MIDI v MATLABu

- **Klasifikace rodiny mapy patchů nástrojů MIDI**
  - **Obecná mapa patchů nástrojů MIDI (GM):**
    - Nástroje jsou seskupeny do 16 rodin s 8 variantami v každé rodině.

Family	Range	Family	Range
Piano	1–8	Reed	65–72
Pitched Percussion	9–16	Pipe	73–80
Organ	17–24	Synth Lead	81–88
Guitar	25–32	Synth Pad	89–96
Bass	33–40	Synth Effects	97–104
Strings	41–48	Ethnic	105–112
Ensemble	49–56	Percussive	113–120
Brass	57–64	Sound Effects	121–128

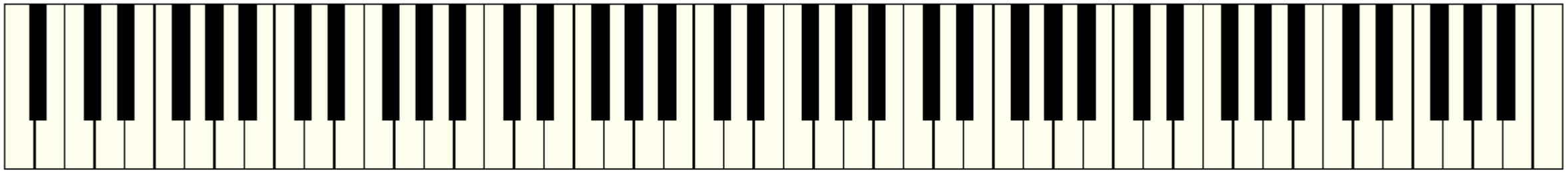
MIDI v MATLABu

- **MIDI Demo** https://jazz-soft.net/demo/GeneralMidi.html

General MIDI Program Chart

MIDI Out: Web Audio (*note: not all instruments support General MIDI*)

If you don't see any General MIDI instrument, please consider installing Jazz-Plugin!



Piano

0 Acoustic Grand Piano
1 Bright Acoustic Piano
2 Electric Grand Piano
3 Honky-tonk Piano
4 Electric Piano 1
5 Electric Piano 2
6 Harpsichord
7 Clavinet

Chromatic Percussion

8 Celesta
9 Glockenspiel
10 Music Box
11 Vibraphone
12 Marimba
13 Xylophone
14 Tubular Bells
15 Dulcimer

Organ

16 Drawbar Organ
17 Percussive Organ
18 Rock Organ
19 Church Organ
20 Reed Organ
21 Accordion
22 Harmonica
23 Tango Accordion

MIDI v MATLABu

- [Ken Schutte's MIDI Project] (http://kenschutte.com/midi)
- Přehled MIDI Toolbox
  - Root Directory:
    - **main.m:** Hlavní skript pro inicializaci a spuštění funkcí MIDI toolboxu.
    - **synth.m:** Syntetizuje zvuk na základě MIDI vstupu.
    - **synthchallenge.m:** Soubor, který uživatelům umožňuje upravovat parametry.

\midi Directory (obsahuje MIDI soubory, např.:)
Barcarolle.mid
Typewriter.mid
…

\private Directory (obsahuje funkce pro zpracování MIDI)
…

\result Directory (výstupní soubory vygenerované toolboxem)

MIDI v MATLABu

- Private Directory Functions:
  - **readmidifile.m:** Čte a analyzuje MIDI soubor a extrahuje MIDI události.
  - **midimeta.m:** Zpracovává metadata (např. tempo, …) z MIDI souboru.
  - **midi2nmat.m:** Převádí MIDI data do formátu *notematrix* pro snadnější manipulaci.
  - **miditool.m:** Základní nástroj pro zpracování a transformaci MIDI událostí.
  - **playmidi.m:** Přehrává MIDI sekvenci pomocí vestavěné zvukové syntézy.
  - **midichannel.m:** Extrahuje a zpracovává data ze specifických MIDI kanálů.
  - **midimsg.m:** Interpretuje jednotlivé MIDI zprávy (např. note on, note off).

MIDI v MATLABu

- **synth.m** function

```
function y=synth(freq, dur, amp, Fs, synthtype, channel)
  % Generate sound based on standard instrument tables
  if channel ~= 10
    if synthtype < 40
      tau = 0.1;  % Decay time
      y = amp .* exp(-t/tau) .* sin(2 * pi * freq * t);
    elseif synthtype < 60
      y = amp .* sin(2 * pi * freq * t);
    else
      y = amp .* square(2 * pi * freq * t);
    end
  end
end
```

MIDI v MATLABu

- **synth.m** for percussion instruments (Channel 10)

```
if channel == 10
  if note > 50
    y = randn(size(t));  % Generate random noise
  else
    tau = 0.01;  % Shorter decay for percussion
    y = amp .* exp(-t/tau) .* randn(size(t));
  end
end
```

In this case, the function generates noise-based sounds,
which are typical for percussion instruments on MIDI channel 10.

MIDI v MATLABu

- **Examples for MIDI files:**
  - Barcarolle.mid
    - String Ensemble 49 (129 notes).
    - String Ensemble 49 (162 notes).
    - Concert Grand Piano 01 (1238 notes).
  - Typewriter.mid
    - Concert Grand Piano 01 (718 notes).
    - Electric Grand Piano 03 (499 notes).
    - Glockenspiel 10 (50 notes).
    - Channel 10: High Bongo 62 (577 notes).

These are examples of MIDI file contents and the instruments they involve, showing how multiple tracks and channels are used.

MIDI v MATLABu

BOHEMIAN.MID

Acoust Grand Piano	01 (1920 notes)
Distort Guitar	31 (273 notes)
Fingered El Bass	34 (480 notes)
Timpani	48 (178 notes)
String Ensemble 1	49 (779 notes)
Choir "Aah"	53 (262 notes)
Orchestral Hit	56 (75 notes)
French Horn	61 (142 notes)
Synth Brass 1	63 (154 notes)
Alto Sax	66 (138 notes)
English Horn	70 (137 notes)

Channel 10:

Acou Grand Piano	1 (1114 notes)
Bass Drum 2	35 (334 notes),
Snare Drum 2	40 (132 notes),
Low Tom 2	41 (2 notes),
Closed Hi-hat	42 (114 notes),
Low Tom 1	43 (52 notes),
Mid Tom 2	45 (30 notes),
Open Hi-hat	46 (94 notes),
Mid Tom 1	47 (1 notes),
Crash Cymbal 1	49 (66 notes),
Ride Cymbal 1	51 (141 notes),
Chinese Cymbal	52 (7 notes),
Splash Cymbal	55 (2 notes),
Crash Cymbal 2	57 (13 notes),
Open Triangle	81 (126 notes)

MIDI v MATLABu

MISSION.MID

Hammond Organ	17 (4203 notes)
Cello	33 (32 notes)
Synth Bass 1	39 (446 notes)
Synth Bass 2	40 (183 notes)
String Ensemble 1	49 (193 notes)
Orchestral Hit	56 (61 notes)
French Horn	62 (44 notes)
Flute	74 (190 notes)
New Age Pad	89 (117 notes)
Sweep Pad	96 (22 notes)
Seashore	123 (11 notes)
Helicopter	126 (4 notes)

Channel 10:	
AcoustSteel Guitar	26 (368 notes)
Bass Drum 1	36 (357 notes),
Side Stick	37 (38 notes),
Snare Drum 1	38 (161 notes),
Hand Clap	39 (8 notes),
Low Tom 2	41 (25 notes),
Closed Hi-hat	42 (870 notes),
Low Tom 1	43 (8 notes),
Pedal Hi-hat	44 (125 notes),
Mid Tom 2	45 (19 notes),
Open Hi-hat	46 (9 notes),
Mid Tom 1	47 (1 notes),
High Tom 2	48 (3 notes),
Crash Cymbal 1	49 (37 notes),
Ride Cymbal 1	51 (10 notes),
Chinese Cymbal	52 (2 notes),
Tambourine	54 (959 notes),
Splash Cymbal	55 (2 notes),
Crash Cymbal 2	57 (18 notes),
Low Bongo	61 (219 notes),
Mute High Conga	62 (234 notes),
Open High Conga	63 (235 notes),
Cabasa	69 (1144 notes),
Long Guiro	74 (15 notes),
Claves	75 (8 notes),
Open Triangle	81 (64 notes)

MIDI v MATLABu

Antonín Dvořák „ Symfonie č. 9 - Novosvětská„

https://musescore.com/user/937936/scores/3081476

https://www.youtube.com/watch?v=xwE1nHtYjec

https://www.youtube.com/watch?v=ZuehaT7iHn0

MIDI nástroje	**(*nástroje v partituře*)**
Ch 01: instr. no. 74	příčná flétna
Ch 02: instr. no. 72	klarinet
Ch 03: instr. no. 01	piano
Ch 04: instr. no. 41	housle
Ch 07: instr. no. 41	housle
Ch 11: instr. no. 43	violoncello

MIDI v MATLABu

Skladba „Také On Me“ norské skupiny A-ha

https://musescore.com/user/937936/scores/3081476

MIDI nástroje	**(*nástroje v partituře*)**
Ch 01: instr. no. 69	hoboj
Ch 02: instr. no. 74	příčná flétna
Ch 03: instr. no. 72	klarinet
Ch 04: instr. no. 41	housle
Ch 07: instr. no. 25	akustická kytara (nylonové struny)
Ch 08: instr. no. 47	orchestrální harfa
Ch 12: instr. no. 35	elektrická baskytara (trsátkem)
Ch 13: instr. no. 01	piano

MIDI v MATLABu

B. Smetana "Vltava", symfonická báseň z cyklu Má vlast

MIDI nástroje	*(nástroje v partituře)*
46 smyčce pizzicato	(housle 1, housle 2, kontrabas)
47 orchestrální harfa	(harfa)
48 tympány	(tympány)
49 smyčcový soubor	(viola, violoncello 1, violoncello 2)
50 smyčcový soubor	(pomalé nasazení tónu - housle, violoncello)
59 tuba	(tuba)
61 lesní roh	(lesní roh v C1, lesní roh v C2)
62 žesťová sekce	(trubka)
69 hoboj	(hoboj)
71 fagot	(fagot)
72 klarinet	(klarinet)
74 příčná flétna	(příčná flétna)

kanál 10, nástroj 49 High Tom 2, nota C2 Hi Mid Tom

MIDI v MATLABu

Julius Fučík „Vjezd gladiátorů"

gladiators.mid https://cs.wikipedia.org/wiki/Vjezd_gladiátorů

MIDI nástroje **(*nástroje v partituře*)**

Ch 01: instr. no. 49 smyčcový soubor
Ch 02: instr. no. 49 smyčcový soubor

Ch 03: instr. no. 62 žesťová sekce
Ch 04: instr. no. 57 trubka
Ch 05: instr. no. 58 pozoun
Ch 06: instr. no. 59 tuba

Ch 07: instr. no. 72 klarinet
Ch 08: instr. no. 74 příčná flétna

Ch 09: instr. no. 48 tympány
Ch 11: instr. no. 10 zvonkohra
Ch 10: instr. no. 1 perkuse
no. 36, 38, 40, 43, 49

perkuse no. 36 basový buben (kopák), 69 not
perkuse no. 38 vířivý buben (virbl), 59 not
perkuse no. 40 vířivý buben (virbl), 95 not
perkuse no. 43 floor tom (kotel), 22 not
perkuse no. 49 crash činel (crash), 51 not
perkuse no. 52 china činel (čína), 1 nota
perkuse no. 57 crash činel (crash), 1 nota

MIDI v MATLABu

- **Freeware MIDI Tools:**
  - **Sekaiju** – MIDI editor and sequencer.
  - **Anvil Studio** – Tool for creating and recording MIDI music.
  - **MidiEditor** – Simple and easy to use MIDI editor.
  - **MuseScore** – Notation program with MIDI input support.
  - **Aria Maestosa** – MIDI sequencer and editor for easy composing.
  - **MidiPiano Virtual Piano** – Virtual piano for playing and recording MIDI files.
  - **Virtual MIDI Piano Keyboard** – Virtual piano keyboard with MIDI input/output.