# Vzorkování, aliasing a antialiasing

Oldřich Plchot, FIT VUT Brno

V tomto cvičení si ukážeme základní postupy pro zpracování analogového signálu, jeho převedení do diskrétní podoby a následnou rekonstrukci. Budeme se také zabývat problémy, které při tomto postupu nastávají a předvedeme si způsoby, jak jim předcházet nebo je úplně odstranit.

## 1 Vytvoření analogového signálu

Matlab ze své podstaty nemůže pracovat se spojitými signály. Vygenerujeme jej tedy proto jako diskrétní signál s velmi malou vzorkovací periodou a abychom si předvedli vzorkování, tak jej budeme **podvzorkovávat**.

Mějme zadaný spojitý signál pomocí funkce:

$$x(t) = \left[\frac{3}{2} + \frac{3}{10}\sin 2\pi t + \sin\frac{2\pi}{3}t - \sin\frac{2\pi}{10}t\right] \cdot \frac{\sin \pi t}{\pi t}, \tag{1}$$

Vytvořme si tedy funkci v Matlabu, která nám bude generovat hodnoty této funkce. Následující definici funkce zkopírujte do souboru nazvaného funx.m a uložte jej někde, kde ho matlab uvidí a nebo správně nastavte cestu.

```
function outp = funx(t)
outp = ((3/2)+((3/10)*sin(2*pi*t))+sin(((2*pi)/3)*t)-sin(((2*pi)/10)*t)).*sinc(t);
```

Teď si můžete zkusit funkci zobrazit:

```
t = [-5:0.001:5];
plot(t,funx(t),'k-')
xlabel('Time')
ylabel('Amplitude')
```

## 2 Analýza našeho spojitého signálu

Před tím, než vůbec začneme s jakýmkoliv vzorkováním bychom si měli položit otázku ohledně toho, jaký signál vzorkujeme a jestli jsme schopni zjistit z jakých frekvencí je seskládán. Pokud dokážeme signál dobře zanalyzovat, tak dokážeme navrhnout takovou vzorkovací periodu nebo frekvenci, že při zpětné rekonstrukci námi navzorkovaného signálu dostaneme přesně původní analogový signál. Konkrétně nás samozřejmě bude zajímat splnění vzorkovacího teorému, který říká, že vzorkovací frekvence musí být alespoň dvakrát větší než největší frekvence zastoupená ve vzorkovaném signálu. Neboli:

$$\Omega_s > 2\omega_{max},$$

kde $\omega_{max}$ je nejvyšší frekvence, kde má ještě signál nějakou energii.

Tedy ještě jednou — mějme definovanou funkci:

$$x(t) = \left[\frac{3}{2} + \frac{3}{10}\sin 2\pi t + \sin\frac{2\pi}{3}t - \sin\frac{2\pi}{10}t\right] \cdot \frac{\sin \pi t}{\pi t}, \tag{2}$$

kterou si můžeme rozepsat jako

$$x(t) = x_1(t) \cdot x_2(t), \tag{3}$$

kde

$$x_1(t) = \frac{3}{2} + \frac{3}{10}\sin 2\pi t + \sin\frac{2\pi}{3}t - \sin\frac{2\pi}{10}t, \tag{4}$$

$$x_2(t) = \frac{\sin \pi t}{\pi t} = \operatorname{sinc} t. \tag{5}$$

Fourierovou transformací funkce $x(t)$ získáme její Fourierův obraz

$$F(j\omega) = \int_{-\infty}^{\infty} x(t)\, e^{-j\omega t}\, \mathrm{d}t, \tag{6}$$

který se dá analogicky podle konvolučního teorému rozepsat

$$F(j\omega) = \frac{1}{2\pi} F_1(j\omega) * F_2(j\omega), \tag{7}$$

kde operace $*$ značí konvoluci.

Díky tomu můžeme vypočítat Fourierovou transformaci odděleně pro každou funkci $x_1(t)$ a $x_2(t)$

$$\begin{aligned} F_1(j\omega) &= 3\pi\delta(\omega) + j\frac{3\pi}{10}\cdot(\delta(\omega+2\pi) - \delta(\omega-2\pi)) + \\ &\quad + j\pi\cdot\left(\delta\left(\omega+\frac{2\pi}{3}\right) - \delta\left(\omega-\frac{2\pi}{3}\right)\right) - j\pi\cdot\left(\delta\left(\omega+\frac{2\pi}{10}\right) - \delta\left(\omega-\frac{2\pi}{10}\right)\right), \\ F_2(j\omega) &= \sqcap\left(\frac{\omega}{2\pi}\right) = u(\omega+\pi) - u(\omega-\pi), \end{aligned} \tag{8}$$

kde obdelníková funkce

$$\sqcap\left(\frac{\omega}{\tau}\right) = \operatorname{rect}\left(\frac{\omega}{\tau}\right) = u\left(\omega+\frac{\tau}{2}\right) - u\left(\omega-\frac{\tau}{2}\right), \qquad \tau = 2\pi, \tag{9}$$

nabývá hodnot

$$\operatorname{rect}(k) = \sqcap(k) = \begin{cases} 0 & \text{pro } |k| > \frac{1}{2} \\ 1,0 & \text{pro } |k| = \frac{1}{2} \\ 1 & \text{pro } |k| < \frac{1}{2}. \end{cases} \tag{10}$$

Konvoluce s Diracovým impulsem má specifický výsledek

$$(\delta * f)(\omega) = \int_{-\infty}^{\infty} \delta(\omega - a) f(a)\, \mathrm{d}a = f(\omega). \tag{11}$$

Výsledný Fourierův obraz (7) bude součtem obdélníkových funkcí s různou amplitudou a šířkou

$$\begin{aligned} F(j\omega) &= 3\pi\sqcap(\omega) + j\frac{3\pi}{10}\sqcap\left(\frac{\omega}{2\pi}+1\right) - j\frac{3\pi}{10}\sqcap\left(\frac{\omega}{2\pi}-1\right) + \\ &\quad + j\pi\sqcap\left(\frac{\omega}{2\pi}+\frac{1}{3}\right) - j\pi\sqcap\left(\frac{\omega}{2\pi}-\frac{1}{3}\right) + j\pi\sqcap\left(\frac{\omega}{2\pi}+\frac{1}{10}\right) - j\pi\sqcap\left(\frac{\omega}{2\pi}-\frac{1}{10}\right). \end{aligned} \tag{12}$$

Podle hraniční podmínky (10) vychází nejvyšší maximální frekvence $\omega_{\max}$

$$|k| = \frac{1}{2}, \qquad k = \frac{\omega_{\max}}{2\pi} - 1, \tag{13}$$

$$\omega_{\max} = 3\pi. \tag{14}$$

Vzorkovací frekvence $\Omega_s$ by pak měla splňovat vzorkovací teorém

$$\Omega_s > 6\pi = 2\,\omega_{\max}, \tag{15}$$

$$f_s > 3, \tag{16}$$

$$T_s < \frac{1}{3}. \tag{17}$$

# 3 Vzorkování

Jakmile jsme se dozvěděli, jak velká musí být naše vzorkovací perioda, aby byl splněn vzorkovací teorém, můžeme přikročit k samotnému vzorkování. Vzorkovací periodu zvolme třeba $T_s = 1/4$ a navzorkujme "spojitou" funkci $x(t)$ jako $x[n] = x(nT_s)$.

```
Ts = 1/4;
n = [-5/Ts:1:5/Ts];
stem(n,funx(n*Ts),'k-')
xlabel('n');
ylabel('Amplitude')
```

## 3.1 Rekonstrukce navzorkovaného signálu

Frekvenčně omezený signál může být přesně rekonstruován pomocí vztahu 18. Musí být samozřejmě splněn vzorkovací teorém, což jsme si poněkud složitou teorií v tomto případě zajistili.

$$x(t) = \sum_{n=-\infty}^{\infty} x[n]\, sinc\left(\frac{t - nT_s}{T_s}\right) \tag{18}$$

Rekonstruovaný spojitý signál $x(t)$ se tedy skládá z nekonečného množství škálovaných a posunutých impulzních odezev. Vytvoříme si tedy funkci, pomocí které budeme rekonstruovat náš signál. Opět uložte tuto funkci do souboru `recon.m`, a to do adresáře tam, kde ji matlab uvidí.

```
function xr = recon(t,range,xn,Ts)
% Suma pres interval, cas t a vzorky xn.
tmp = 0;
for n = -range:range;
  tmp = tmp + (xn(round(length(xn)/2)+n).*sinc((t-n*Ts)/Ts));
end
xr = tmp;
```

A nyní, jakmile máme funkci vytvořenou, můžeme zkusit náš navzorkovaný signál rekonstruovat:

```
clc;

t = [-5:0.001:5];
n=[-5/Ts:1:5/Ts];
xn = funx(n*Ts);

% postupne tisknout funkci pres vetsi interval abyste videli, jak se vysledek
% zpresnuje (zvetsujte druhy parametr funkce recon)
 hold off
 plot(t,funx(t),'k--')            %vytiskneme originalni signal
 hold on
 plot(t,recon(t,2,xn,Ts),'k-')
```

# 4 Aliasing a antialiasing

Aliasing je jev, kdy je vlivem vzorkování původní frekvence spojitého signálu nahrazena úplně jinou frekvencí. K Aliasingu dochází při nedodržení vzorkovacího teorému. Nejprve se podíváme na několik ukázek aliasingu. První bude demo s jednoduchou kosinusovkou. Rozbalte si archiv condis-v200.zip a v matlabu se přepněte do adresáře s demem. Potom spusťte demo pomocí `con2dis`. Experimentujte s aplikací a sledujte kdy dochází k aliasingu.

Druhá ukázka aliasingu bude na 2D signálu - obrázku. Zde bude také docházet k aliasingu vlivem převzorkování a to konkrétně při zmenšování obrazu. Stáhněte si ze stránek projektu demo `image_aliasing.m` a obrázek `barbara.gif`. Demo spustíte zavoláním skriptu `image_aliasing`. Můžete demo upravit tak, aby místo zobrazování obrázku vytvořilo krátká videa. V tomto případě snižte krok, po kterém jsou generovány jednotlivé obrázky ze 100 na například 5 a zakomentujte zobrazování obrázku. Podařilo se vám vytvořit video? :-)

Nyní budeme pokračovat s ruční prací, podíváme na příklady z přednášky a ukážeme si případy kdy aliasing nenastane, kdy nastane a potom jej odstraníme filtrem typu dolní propusť.

## 4.1 Případ kdy je všechno OK

Nejprve tedy příklad kdy nám to vyjde. Nadefinujeme frekvence, osy, indexy:

```
Fs = 8000; fmax = 3000;
Os = 16000 * pi; omax = 6000 * pi;
o = (-400:400)/100 * Os; % presne 100 bodu na jednu periodu.
gj0 = find (o==0);

zind = (-300:300) + gj0;  %indexy na zobr.
zob = o(zind);   % omegy na zobr.
```

```
%vytvorime Hammingovo okno
X = zeros(size(o));
kam = (-37:37) + gj0; X(kam) = hamming(2*37+1);
subplot(411); plot (zob, X(zind)); axis tight
% ted periodizovane - okolo -3,-2,-1,0,1,2,3
T = 1/Fs;
Xs1 = zeros(size(o)); Xs2=Xs1; Xs3=Xs1;Xs4=Xs1;Xs5=Xs1;Xs6=Xs1; Xs7=Xs1;
kam = (-37:37) + gj0 - 300; Xs1(kam) = 1/T * hamming(2*37+1);
kam = (-37:37) + gj0 - 200; Xs2(kam) = 1/T * hamming(2*37+1);
kam = (-37:37) + gj0 - 100; Xs3(kam) = 1/T * hamming(2*37+1);
kam = (-37:37) + gj0 - 0; Xs4(kam) = 1/T * hamming(2*37+1);
kam = (-37:37) + gj0 + 100; Xs5(kam) = 1/T * hamming(2*37+1);
kam = (-37:37) + gj0 + 200; Xs6(kam) = 1/T * hamming(2*37+1);
kam = (-37:37) + gj0 + 300; Xs7(kam) = 1/T * hamming(2*37+1);
Xs  = Xs1 + Xs2 + Xs3 + Xs4 + Xs5 + Xs6 + Xs7;
subplot(412);
plot(zob,Xs1(zind),'b',zob,Xs2(zind),'b',zob,Xs3(zind),'b',zob,Xs4(zind),...
'b',zob,Xs5(zind),'b',zob,Xs6(zind),'b',zob,Xs7(zind),'b',zob,Xs(zind),'r'); axis tight
% vytvorime si rekonstrukcni filtr
H = zeros(size(o));
H((-50:50) + gj0) = T;
subplot(413); plot(zob,H(zind)); axis tight
% a vyfiltrujeme zpatky nas originalni signal
Xr = H .* Xs;
subplot(414); plot(zob,Xr(zind)); axis tight
```

## 4.2 Případ, kdy dojde k aliasingu

Tentokrát náš signál bude mít ne 3000, ale rovnou 7000Hz. Protože ponecháme vzorkovací frekvenci na 8000Hz, tak dojde k aliasingu.

```
X = zeros(size(o));
kam = (-87:87) + gj0; X(kam) = hamming(2*87+1);
subplot(411); plot (zob, X(zind)); axis tight
% ted periodizovane - okolo -3,-2,-1,0,1,2,3
T = 1/Fs;
Xs1 = zeros(size(o)); Xs2=Xs1; Xs3=Xs1;Xs4=Xs1;Xs5=Xs1;Xs6=Xs1; Xs7=Xs1;
kam = (-87:87) + gj0 - 300; Xs1(kam) = 1/T * hamming(2*87+1);
kam = (-87:87) + gj0 - 200; Xs2(kam) = 1/T * hamming(2*87+1);
kam = (-87:87) + gj0 - 100; Xs3(kam) = 1/T * hamming(2*87+1);
kam = (-87:87) + gj0 - 0; Xs4(kam) = 1/T * hamming(2*87+1);
kam = (-87:87) + gj0 + 100; Xs5(kam) = 1/T * hamming(2*87+1);
kam = (-87:87) + gj0 + 200; Xs6(kam) = 1/T * hamming(2*87+1);
kam = (-87:87) + gj0 + 300; Xs7(kam) = 1/T * hamming(2*87+1);
Xs  = Xs1 + Xs2 + Xs3 + Xs4 + Xs5 + Xs6 + Xs7;
subplot(412);
plot(zob,Xs1(zind),'b',zob,Xs2(zind),'b',zob,Xs3(zind),'b',zob,Xs4(zind),...
'b',zob,Xs5(zind),'b',zob,Xs6(zind),'b',zob,Xs7(zind),'b',zob,Xs(zind),'r'); axis tight
% filtr
H = zeros(size(o));
H((-50:50) + gj0) = T;
subplot(413); plot(zob,H(zind)); axis tight
% rekonstrukce
Xr = H .* Xs;
subplot(414); plot(zob,Xr(zind)); axis tight
```

Vysvětlete, jak vznikl výsledný aliasovaný signál. Zamyslete se nad tím, kde byste Hammingovo okno uřezali, abyste dostali při rekonstrukci alespoň něco rozumného.

## 4.3 Antialiasingový filtr

Abychom se zbavili nežadoucího efektu, kdy se nám periodizované signály překrývají, tak je musíme ořezat. Ořez provedeme filtrováním pomocí dolní propusti. Musíme dosáhnout toho, že když nakopírujeme po vzorkovací frekvenci ořezané signály vedle sebe, tak by se nepřekrývaly a tím pádem se mezi sebou nemohly násobit a zbůsobovat tak vznik úplně jiného signálu, než byl náš původní. Tímto postupem sice ztratíme možnost stoprocentně zrekonstruovat původní signál, ale výsledek po rekonstrukci bude narozdíl od výsledku v přechozím příkladě náš originální signál stále připomínat.

```
% puvodni signal
X = zeros(size(o));
kam = (-87:87) + gj0; X(kam) = hamming(2*87+1);
subplot(511); plot (zob, X(zind)); axis tight
%% orezane Hammingovo okno
aux = hamming(2*87+1); aux = aux (88-49:88+49);
Xaa = zeros(size(o));
kam = (-49:49) + gj0; Xaa(kam) = aux;
subplot(512); plot (zob, Xaa(zind)); axis tight
% ted periodizovane - okolo -3,-2,-1,0,1,2,3
T = 1/Fs;
Xs1 = zeros(size(o)); Xs2=Xs1; Xs3=Xs1;Xs4=Xs1;Xs5=Xs1;Xs6=Xs1; Xs7=Xs1;
kam = (-49:49) + gj0 - 300; Xs1(kam) = 1/T .* aux;
kam = (-49:49) + gj0 - 200; Xs2(kam) = 1/T .* aux;
kam = (-49:49) + gj0 - 100; Xs3(kam) = 1/T .* aux;
kam = (-49:49) + gj0 - 0; Xs4(kam) = 1/T .* aux;
kam = (-49:49) + gj0 + 100; Xs5(kam) = 1/T .* aux;
kam = (-49:49) + gj0 + 200; Xs6(kam) = 1/T .* aux;
kam = (-49:49) + gj0 + 300; Xs7(kam) = 1/T .* aux;
Xs  = Xs1 + Xs2 + Xs3 + Xs4 + Xs5 + Xs6 + Xs7;
subplot(513);
plot(zob,Xs1(zind),'b',zob,Xs2(zind),'b',zob,Xs3(zind),'b',zob,Xs4(zind),'b',...
zob,Xs5(zind),'b',zob,Xs6(zind),'b',zob,Xs7(zind),'b',zob,Xs(zind),'r'); axis tight
% filtr
H = zeros(size(o));
H((-50:50) + gj0) = T;
subplot(514); plot(zob,H(zind)); axis tight
% rekonstrukce
Xr = H .* Xs;
subplot(515); plot(zob,Xr(zind)); axis tight
```

# 5 Kvantování

Ve zpracování signálů je kvantování proces, kdy se snažíme reprezentovat spojitý interval hodnot, nebo velké množství diskrétních hodnot relativně malou množinou diskrétních symbolů nebo celých čísel. Typické použití kvantování je převod diskrétního signálu (což je navzorkovaný spojitý signál) do digitální podoby, kterou můžeme například uložit na pevný disk počítače. Příkladem může být třeba hudební CD, na kterém je uložen navzorkovaný analogový signál (vzorkovací frekvence je 44100 Hz) a každý je uložen (nakvantován) na 16 bitech. Každý vzorek tedy může nabývat jednu ze 65536 hodnot.

Jednoduchým způsobem kvantizace je uniformní kvantizace. Předpokládejme, že hodnoty našeho signálu se mohou měnit v intervalu $[a, b]$, tak kvantizační funkce bude mít tvar

$$x_Q = \left\lfloor \frac{x-a}{b-a} \cdot \left(2^N - 1\right) \right\rfloor \cdot \frac{b-a}{2^N - 1} + a, \tag{19}$$

kde N reprezentuje počet bitů, na kterých bude uložena navzorkovaná hodnota. Signál je rozdělen do N intervalů, které jsou rovnoměrně rozloženy. Tato kvantizační funkce pracuje následovně:

- Signál je převeden do intervalu $[0, 1]$
- Naškálován do rozsahu $[0, 2^N - 1]$
- Ořezem desetinné čárky (funkce floor) vyjádřen celočíselně
- Kvantovaný signál je znovu převeden zpět do intervalu $[a, b]$

Kvantizační funkci zapíšeme v matlabu:

```
function outp = my_quantization(x, a, b, N)
outp = floor(((2^N)-1)*(x-a)/(b-a))*(b-a)/((2^N)-1) + a;
```

## 5.1 Kvantování zvuku

Načtěte si zvukový signál a postupně jej kvantujte na 4 a 2 bity. Vždy si poslechněte výsledek. Zkuste si taky poslechnout rozdíl původního a nakvantovaného signálu. Můžete si také zpočítat chybu mezi původním a nakvantovaným signálem.

```
% kvantizace audia
load handel;
soundsc(y, Fs);
x = size(y);
N = 4;
yq = zeros(x(1),1);
for k = 1:x(1)
  yq(k) = my_quantization(y(k), -1, 1, N);
end
soundsc(yq, Fs);
soundsc(y-yq, Fs);
```

## 5.2 Kvantování obrazu

Podobně jako u zvuku si načtěte libovolný obrázek a uložte jej například na 4 nebo 2 bitech a zobrazte výsledek. Zobrazte také rozdíl mezi nakvantovaným a původním obrázkem. Opět můžete spočítat chybu mezi původním a nakvantovaným obrázkem na různých počtech bitů.

```
% kvantizace obrazku
I=imread('lena.gif');
imshow(I);
x = size(I);
N = 2;
I=double(I);
IQ = zeros(x(1),x(2));
for k=1:x(1)
  for l=1:x(2)
    IQ(k,l) =   my_quantization(I(k,l), 0, 255, N);
  end
end
I=uint8(I);
IQ=uint8(IQ);
imshow(IQ);
```

# 6 Reference


http://www.jiisuki.net/reports/nadlun-5_lab3_report.pdf
http://webfea-lb.fea.aub.edu.lb/dsaf/labs/EECE691C_Prelab3.pdf
http://www.fit.vutbr.cz/study/courses/ISS/public/pred/vzork/vzork.pdf