02 Geneze signálů, diskrétní Fourierova transformace (DFT)

Geneze signálů

Za signál v dnešním cvičení budeme považovat jednorozměrnou časovou sérii měnící se v čase. Všechny generované signály začínají a končí, jedná se tedy o deterministické signály, finitní. Ačkoliv pro jednoduchost budeme ze začátku využívat harmonických periodických funkcí (sinus), během semestru začneme pracovat i s biologickými signály, tedy signály náhodnými neperiodickými.

Dodržování **vzorkovacího teorému** při vzorkování, časové diskretizaci, je klíčové pro jakékoliv operace se signály. Pokud tuto podmínku nesplníme, původní informaci v signálu nelze korektně interpretovat.

$$f_s > 2f_{\max},$$

kde f_s je vzorkovací kmitočet a $f_{\max}$ je nejvyšší (nejrychlejší) frekvenční složka signálu.

Signály poté lze zobrazovat ve vzorcích $y[n]$, kde $n \in 0, 1, 2, \ldots, (N-1)$ a N je počet vzorků signálu, nebo v diskrétním čase $y[t_n]$. Přepočet mezi vzorky na diskrétní čas se provádí intuitivně přes vzorkovací kmitočet f_s, neboli počet vzorků *k* za sekundu. Z trojčlenkou snadno odvodíme vzájemný přepočet:

$$f_s \text{ vzorků } \ldots 1\ s$$
$$n \text{ vzorků } \ldots t_n$$

$$t_n = \frac{n}{f_s}$$

Obdobně spočítáme i celkovou délku signálu:

$$T = \frac{N}{f_s}$$

Povšimněte si, že pro jednoduchost je první vzorek signálu často značen jako $n = 0$ a poslední jako $N - 1$. MATLAB ale indexuje od jedničky, tedy $\acute{n} \in 1, 2, \ldots, N$, proto výpočet času t_n musí být o jeden vzorek posunut zpět

$$t_n = \frac{\acute{n} - 1}{f_s}$$

Pozn.: Vzorkovací teorém a přepočet vzorků na čas si **dobře zapamatujte**.

Diskrétní Fourierovy řady

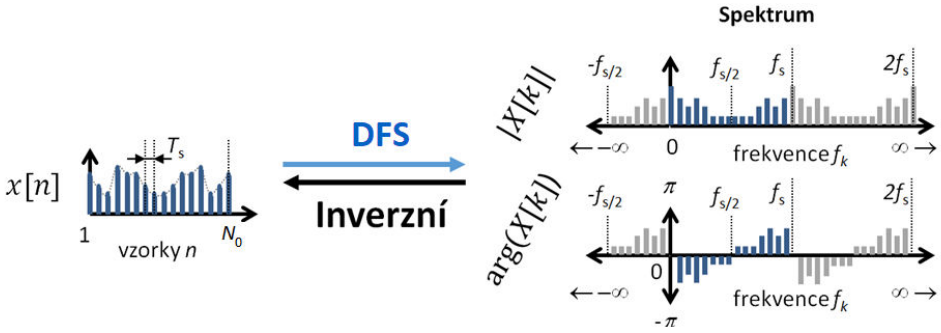
Diskrétní Fourierovy řady (DFS) je lineární transformací, která rozkládá diskrétní periodický signál $x[t_n]$ (časovou řadu vzorků n odpovídající diskrétnímu času $t_n = \frac{n}{f_s}$, délky/periodou N_0 s vzorkovacím kmitočtem $f_s = \frac{1}{T_s}$) na harmonické složky $X[f_k]$ pomocí ortogonální funkce komplexní exponenciály $e^{-i\Omega_k n}$, kde $\Omega_k = 2\pi \frac{f_k}{f_s} = 2\pi \frac{k}{N_0}$. Celou operaci si můžeme představit jako výpočet vzájemné energie signálu a komplexní

harmonické funkce s frekvencí celého násobku $f_k = \frac{k f_s}{N_0}$. Modul komplexního čísla $|X[f_k]|$ pak reprezentuje amplitudu harmonické složky, která je i rovna vzájemné energii signálu a ortogonální báze s jednotkovou energií. Argument $\varphi = \arg(X[f_k])$ je fáze harmonické složky.

Komplexní spektrum reálného signálu je diskrétní, symetrické kolem 0 na intervalu $\left(-\frac{f_s}{2}; \frac{f_s}{2}\right)$ a periodicky se opakující s celým násobkem f_s. Proto pro výpočet spektra postačí pouze koeficienty s celým násobkem $k \in \langle 0, N_0 - 1 \rangle$. Stejnosměrné složce odpovídá $k = 0$, tj. $\Omega_k = 0$, tj. $X[0] = \frac{1}{N_0} \sum_{n=0}^{N_0-1} x[n] e^0$, tj. aritmetický průměr vzorků $X[0] = \frac{1}{N_0} \sum_n x[n]$.

$$\text{DFS} : X[f_k] = \frac{1}{N_0} \sum_{n=0}^{N_0-1} x[n] e^{-i\Omega_k n}$$

$$\text{IDFS} : x[t_n] = \sum_{k=0}^{N_0-1} X[f_k] e^{i\Omega_k n}$$



Původní signál $x[t_n]$ můžeme opět získat inverzní transformací (IDFS) prostým součtem všech harmonických komponent $x_k[t_n]$ s odpovídajícími amplitudami a fázovým zpožděním.

$$x_k[t_n] = X[f_k] e^{i\Omega_k n}$$

Rychlá Fourierova transformace (FFT)

Výpočetní náročnost DFS je ve složitosti N^2. Celé řešení však lze nahradit rekurzí a postupným rozdělováním signálu na části o dvou vzorcích dle známého Cooley-Tukey algoritmu. Celková výpočetní složitost se tak redukuje na $N \log(N)$. Pro optimální výpočet by tedy délka analyzovaného signálu měla mít délku druhé mocniny $N = 2^b$. Implementace je v MATLAB provedena pomocí funkce `fft()`a pro inverzní transformaci `ifft()`. Ta však oproti DFS vychází z předpisu diskrétní Fourierovy transformace (DFT) pro neperiodické signály délky *N*, tedy normování "periodou" $1/N$ nemůže být na straně dopředné transformace, ale je na straně inverzní. Detailně bude DFT probírána v příštím týdnu.

$$\text{FFT}: X[f_k] = \sum_{n=0}^{N-1} x[n] e^{-i\Omega_k n}$$

$$\text{IFFT}: x[t_n] = \frac{1}{N} \sum_{k=0}^{N-1} X[f_k] e^{i\Omega_k n}$$

Pokud má tedy výpočet pomocí `fft()` odpovídat DFS, tj. modul spektra odpovídat amplitudě frekvenčních komponent, musí být komplexní spektrum v kódu dodatečně normováno $\frac{1}{N}$ a při inverzní transformaci vynásobeno N:

```
N=length(x); % počet vzorků signálu (jedna perioda)
X=(1/N)*fft(x); % DFS/DFT – komplexní spektrum X[k]
x=N*ifft(X); % IDFS – rekonstrukce signálu x[n]

magnitude=abs(X); % |X[k]|
phase=angle(X); % Phi[k]
```

To ovšem mírně komplikuje korektní vyjádření magnitudy spektra $|X(f_k)|_{\text{dB}}$ v dB, což by při prostém logaritmování vektoru $X[f_k]$ neodpovídalo definici v **Parsevalově rovnosti** energií v časové a frekvenční oblasti: $\sum_{n=1}^{N} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{K-1} |X[f_k]|^2$.

$$20 \cdot \log_{10}\left(\frac{1}{N} |X[f_k]|\right) = 10 \cdot \log_{10}\left(\frac{1}{N^2} |X[f_k]|^2\right) \text{ ... nikoli požadovaná } 10 \cdot \log_{10}\left(\frac{1}{N} |X[f_k]|^2\right)$$

Při běžné analýze normalizaci N vzorky často úplně ignorujeme, protože hledáme např. nejvyšší spektrální čáry a vynechání normalizace mění pouze měřítko magnitudy, v dB pouze stejnosměrný posun všech čar. Energie signálu počítáme pouze z jedné veličiny intenzity (typicky U nebo I) bez známé resistance za předpokladu ($R = 1\Omega$), tedy nelze mluvit doslova o energii ve smyslu fyzikálním, hovoříme pak o **výkonovém spktru** či **výkonové spektrální hustotě**. Pro formální úplnost nebo měření energie se všemi známými veličinami je ovšem Parsevalovu rovnost nutno dodržet a umocnění spektra na energetické/ výkonové spektrum provézt ještě **před normalizací**.

```
N=length(x); % počet vzorků signálu
X=(1/N)*fft(x); % DFS/DFT – komplexní spektrum X[k]
XdB=10*log10( (1/N)*abs(fft(x)).^2 ); % PSD - výkonové spektrum
```

Pro vykreslování grafů musíme definovat časovou osu signálu v sekundách a frekvenční osu pro spektrum. Uvědomte si, že spektrum DFT má stejný počet spektrálních čar, jako je počet vzorků v signálu $K = N$:

```
f=linspace(0,fs-fs/N,N); % frekvence <0; fs) Hz (kde, fs již odpovídá periodické DC složce)
t=linspace(0,(N-1)/fs,N); % časová osa, též t=0:1/fs:(N-1)/fs
% Pozn.: N=T*fs;
```

Inverzní rychlá Fourierova transformace (IFFT) a filtrace ve spektru

Při IFFT dochází k rekonstrukci signálu součtem harmonických signálů o frekvenci f_k, amplitudě $|X[f_k]|$ a fázovém posunu $\varphi[f_k]$. V případě, že chceme ze signálu některé frekvenční složky odstranit - **filtrovat**,

můžeme tak učinit ve spektru. Ve spektru nalezneme odpovídající spektrální čáry a vynulujeme je $|X[f_k]| = 0;\ \text{pro}\, f_k \in \text{filtrace}.$

```
% Př.: odstraňte síťové rušení 50 Hz
N=length(x);
X=(1/N)*fft(x); % X[f]
f=linspace(0,fs-fs/N,N); % frekvenční osa
idx=f>49.5 & f<50.5; % hledáme indexy spektrálních čár v rozmezí 49.5-50.5 Hz
X(idx)=0;
x=N*ifft(X); % filtrovaný signál x[t]
```

Pozn.: Ačkoli MATLAB pracuje s 64bit přesností (double), i zde dochází k numerickým nepřesnostem, které způsobí po IFFT vznik komplexního signálu (s minimální imaginární složkou) namísto původního čistě reálného. Je tedy nutné používat pouze reálnou složku výsledku.

```
x=real(x);
```

Cíle úlohy

Geneze signálů

1) Vygenerujte si harmonický signál

- $y[t_k] = Y_0 + Y_{\max}\sin(2\pi f_0 t_n + \varphi)$
- vykreslete si signál ve vzorcích $y[n]$
- vykreslete signál v diskrétním čase $y[t_n]$

2) Vygenerujte si harmonický signál o kmitočtu f_0 pro různé vzorkovací kmitočty a sledujte, co se stane po nesplnění vzorkovacího teorému

- $f_s > 10 f_0$
- $f_s > 3 f_0$
- $f_s > 1,2 f_0$

Diskrétní Fourierovy řady

1) Spočtěte spektrum signálu

- Generujte umělý signál součtem dvou různých harmonických funkcí
- Spočtěte komplexní spektrum, zobrazte amplitudovou i fázovou složku
- Nalezněte všechny nenulové spektrální čáry $|X[f_k]| > 0$ a pomocí magnitudy a fáze zobrazte jejich časový průběh $x_{f_k}[t_n]$

Bonus

- Vygenerujte obdélníkové průběhy a zobrazte jejich spektra,
- vygenerujte trojúhelníkové průběhy a zobrazte jejich spektra.

Užitečné funkce:

```
fft,ifft,abs,real,angle,exp,log10,find,mod,load,for,if,end
```

Obsah

Nápověda

Geneze signálu

1) Generujte harmonický signál

Harmonický signál (sinusovka) se řídí předpisem $y[t_n] = Y_0 + Y_{\max}\sin(2\pi f_0 t_n + \varphi)$, kde se hodnota $y[t_n]$ mění v závislosti na čase t_n, ostatní proměnné jsou konstantami. Jedná se tedy zjednodušeně o funkci $y = f(t)$. Vygenerujeme si tedy časový vektor, ke kterému dopočítáme hodnoty *y*.

```
T=      ; % délka signálu (např. 5 s)
fs=     ; % vzorkovací kmitočet, (např. 1000 Hz)
f0=     ; % frekvence harmonické složky (např. 5 Hz)
fi=     ; % fáze (např. pi/4)
Ymax=   ; % amplituda harmonické složky (např. 2)
Y0=     ; % stejnosměrný posun (např. 1)
```

Zadefinujte si proměnné:

```
T=5;

f0=5;
fs=100;

fi=pi/4;
Ymax=2;
Y0=1;
```

Vytvoříme si časový vektor *t*. Můžeme postupovat několika způsoby. Buď přes vektor vzorků signálu nebo rovnou v čase.

Přes vzorky signálu:

```
n=1:(T*fs); % vektor od 1. po poslední vzorek N, N=T*fs;
t=n/fs; % vzorky na čas
```

Postup výše by platil, kdyby signál začínal od nultého vzorku a končil v *N*-1 vzorku. MATLAB ale indexuje od 1, signál proto začíná prvním vzorkem a končí v *N* vzorku a časový vektor t nezačíná nulou. Proto, pokud nám signál má začínat v čase t=0, je vhodné časový vektor o jeden vzorek posunout.

```
t=(n-1)/fs;
```

Lze generovat vektor `t` přímo pomocí `linspace`:

```
% linspace( počátek, poslední prvek, počet vzorků )
t = linspace(0, T-1/fs, T*fs);
```

Další variantou je generování vektoru s krokem. Počáteční čas v našem případě bude 0, konečný *T,* ale opět posunutý o čas jednoho vzorku *T*-1/*fs.* Krok (časový rozestup mezi sousedními vzorky) odpovídá 1/*fs.*

```
t=0:1/fs:T-1/fs; % od času 0 po T-1/fs s krokem 1/fs
```

Pro výpočet funkčních hodnot y jednoduše použijeme skalární násobení:

```
y=Y0+Ymax*sin(2*pi*f0*t+fi);
```

Vygenerujte si časový vektor *t* a signál *y*(*t*):

```
t = 0:1/fs:T-1/fs
```

```
t = 1×500
         0    0.0100    0.0200    0.0300    0.0400    0.0500    0.0600    0.0700 ···
```

```
y= Y0 + Ymax*sin(2*pi*f0*t+fi)
```

```
y = 1×500
    2.4142    2.7820    2.9754    2.9754    2.7820    2.4142    1.9080    1.3129 ···
```

Zobrazte signál:

v subplot 3x1:

- signál s časovou osou (propojené vzorky)

```
plot(t,y)
```

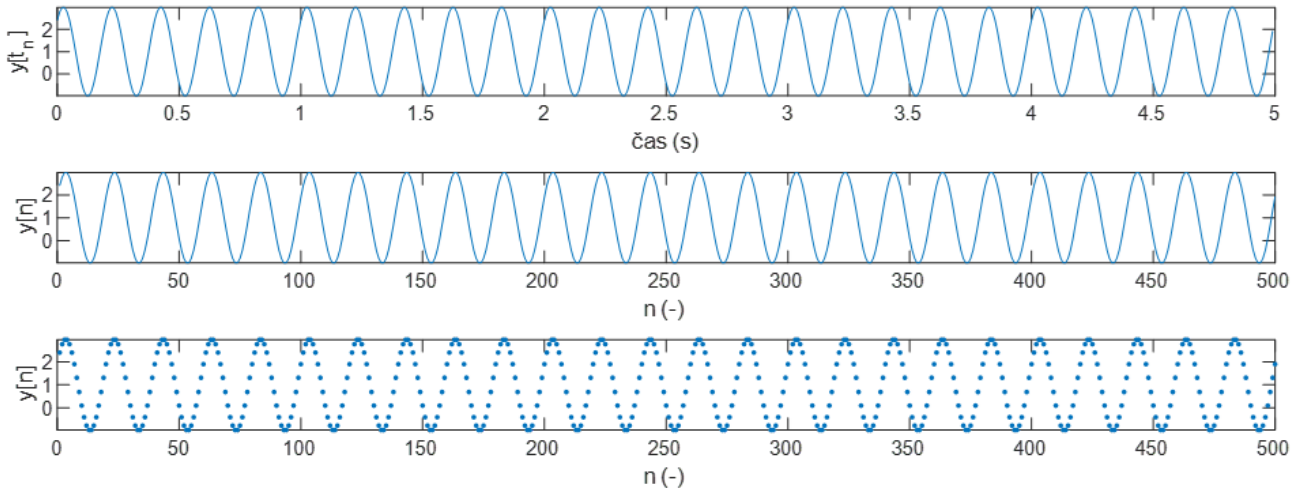
- signál s osou ve vzorcích (propojené vzorky)

```
% plot(y) bude mít osu od 1, tj. musíme upravit na:
plot(n,y)
```

- signál bez spojnic (pouze diskrétní hodnoty)

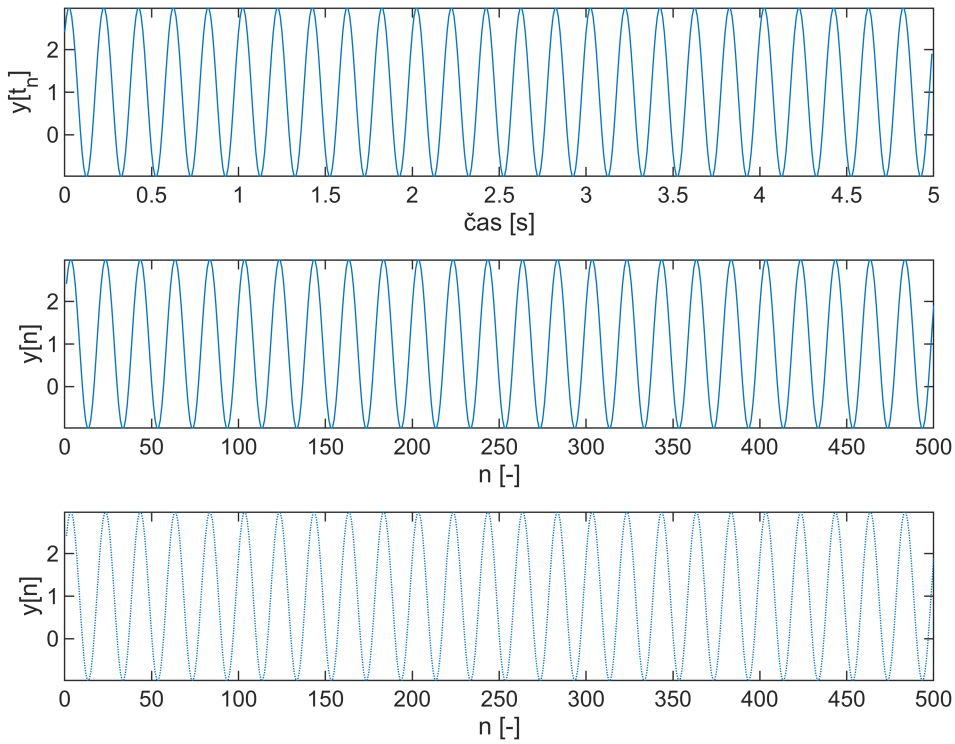
```
plot(n,y,':')
```

Popište všechny osy a uveďte jednotky. Připomeňte si funkce a jejich parametry: `figure, subplot, plot, xlabel, ylabel`



Uvědomte si, že ačkoliv budeme vykreslovat většinu signálů spojitě, vždy se bude jednat pouze o diskrétní signál jako je na subplot(3,1,3).

```
figure
subplot(3,1,1);
plot(t,y)
xlabel("čas [s]")
ylabel("y[t_n]")
n=1:(T*fs);
subplot(3,1,2);
plot(n,y)
xlabel("n [-]");
ylabel("y[n]")
subplot(3,1,3)
plot(n,y,":")
xlabel("n [-]");
ylabel("y[n]")
```



DFS

1) Výpočet spektra signálu

Vytvořte si signál součtem dvojice harmonických signálů. Doba trvání signálů může být např. $T = 2\,\mathrm{s}$ (celý násobek periody harmonických složek).

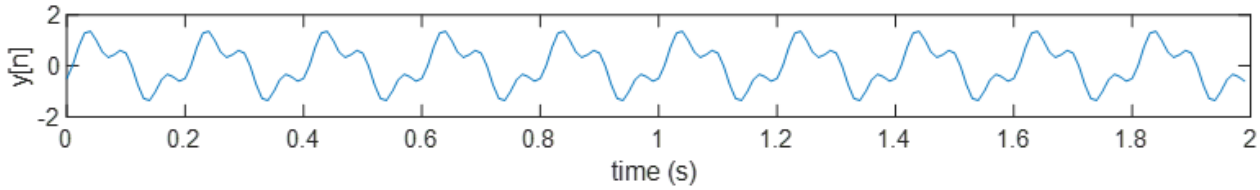
```
% 1. harmonická složka y1[t]
f1=5; % Hz
A1=1; % V
phi1=0; % rad

% 2. harmonická složka y2[t]
f2=15; % Hz
A2=0.5; % V
phi2=-pi/2; % rad
```

Generujte časovou osu `t` platnou pro vzorkovací kmitočet `fs`. **Doplňte kód:**

```
fs=100; % Hz
T=2; % s
N = fs*T;
n = 1:N;
t=0:1/fs:T-1/fs;
            % časová osa od 0 po T-1/fs (s) pro T*fs vzorků
```

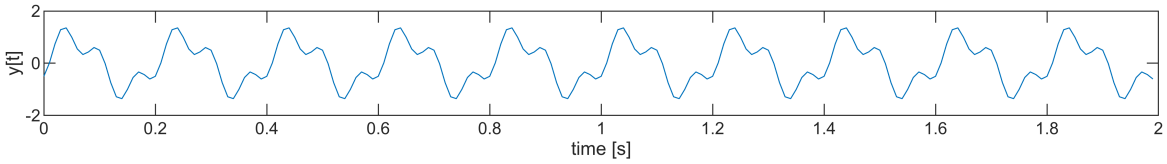
Generujte signál jako součet dvojice harmonických signálů a zobrazte výsledek:

$$y[t_n] = A_1 \cdot \sin(2\pi f_1 t + \varphi_1) + A_2 \cdot \sin(2\pi f_2 t + \varphi_2)$$



Pozn.: Nezapomeňte popsat osy, platí i pro budoucí grafy

```
yt = A1*sin(2*pi*f1*t+phi1)+A2*sin(2*pi*f2*t+phi2);        % y[t]= y1[t]+y2[t]
figure
%subplot(2,1,1)
plot(t,yt)
set(gcf, 'Position', [0 0 900 100]);
xlabel("time [s]")
ylabel("y[t]")
```



Spočtěte komplexní Fourierovo spektrum $Y[k]$ signálu $y[n]$ a vygenerujte frekvenční osu f_k.

$$Y[k] = \mathrm{FFT}\{y[n]\}$$

$$f_k \in < 0; f_s)$$

$$f_k = \frac{k f_s}{N} \ldots Y[f_k]$$

$$K = N;$$

Dle definice výše bude počet spektrálních čar `K` odpovídat počtu vzorků v signálu `N`. Hodnoty frekvenční osy tedy budou od stejnosměrné složky $f_k = 0$(pro $k = 0$) po $f_k = f_s - \frac{f_s}{N}$ (pro $k = K - 1$). Pro generování frekvenční osy využijte funkce `linspace` nebo generování vektoru s krokem.

```
Y=(1/N)*fft(yt)% Y[k] komplexní spektrum
```

```
Y = 1×200 complex
   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i ···
```

```
f = linspace(0,fs-fs/N,N)
```

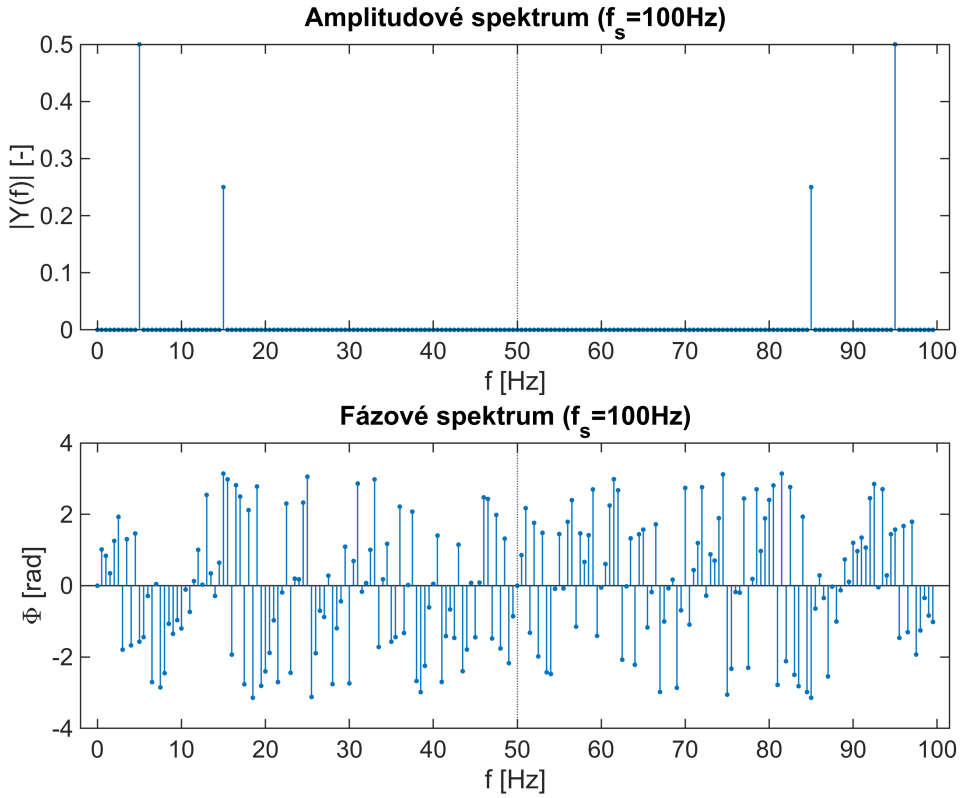
```
f = 1×200
         0    0.5000    1.0000    1.5000    2.0000    2.5000    3.0000    3.5000 ···
```

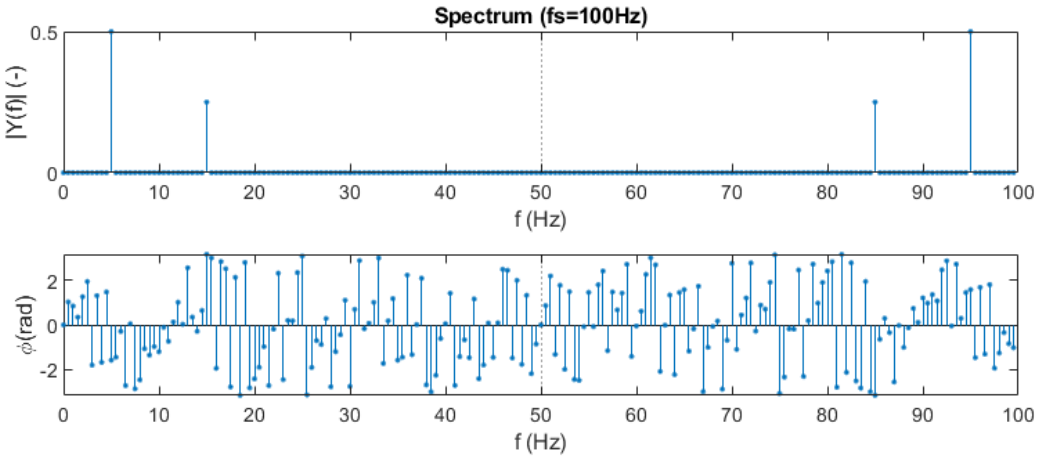
Pomocí funkce stem zobrazte amplitudové i fázové spektrum:

$|Y[f_k]|$

$\varphi[f_k]$

```
figure
subplot(2,1,1)
stem(f,abs(Y),'.')% |Y[f]| absolutní spektrum (magnitude)
xline(fs/2,":")
ylabel("|Y(f)| [-]");
xlabel("f [Hz]");
title("Amplitudové spektrum (f_s=100Hz)")

subplot(2,1,2)
stem(f,angle(Y),".")% Phi(Y[f]) fázové spektrum (phi)
xline(fs/2,":")
title("Fázové spektrum (f_s=100Hz)")
xlabel("f [Hz]")
ylabel("\Phi [rad]")
```





V absolutním spektru nalezněte nenulové frekvenční komponenty (kvůli kvantizačnímu šumu např. $Y[k] > 10^{-6}$).

```
k=find(abs(Y)>1e-6); % množina nenulových spektrálních čar
disp(f(k)) % vypíše hodnotu frekvence k-té spektrální čáry
```

```
     5    15    85    95
```

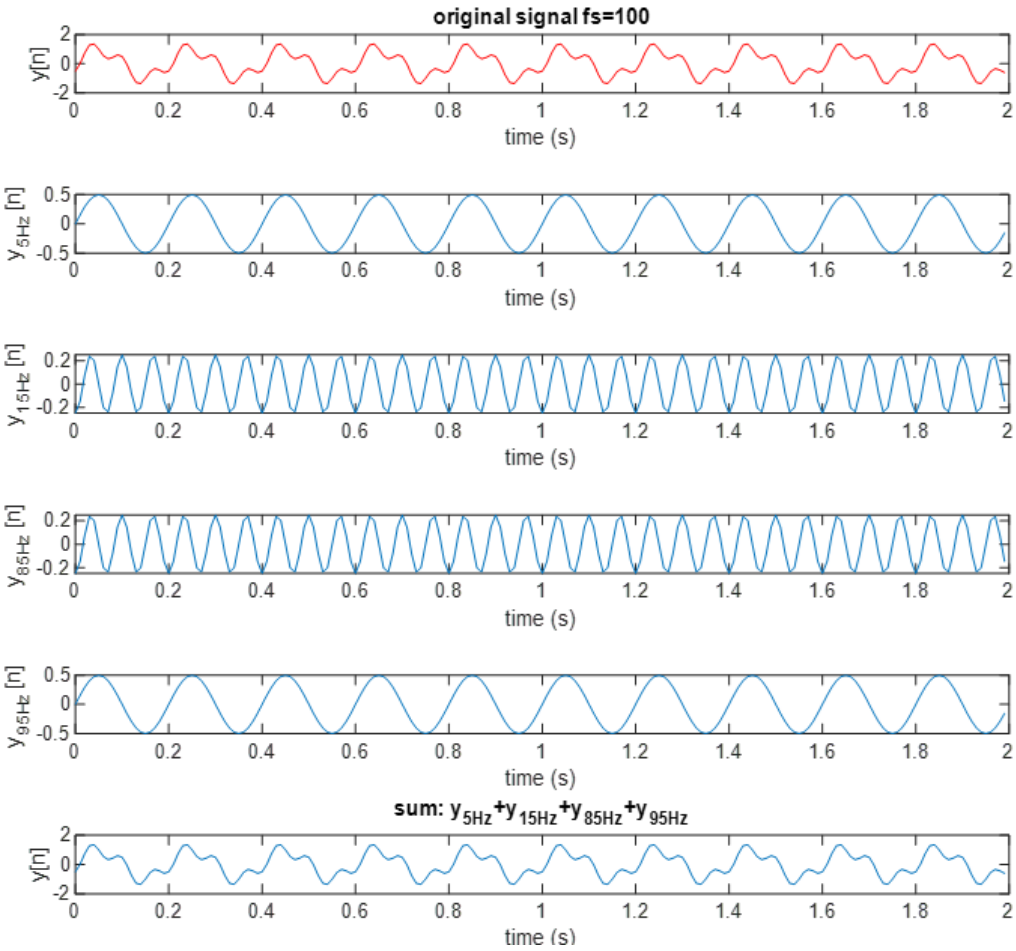
Z nalezených spektrálních čar vygenerujte časové průběhy:

$$y_k[t_n] = Y[k]e^{i\Omega_k n} = Y[k]e^{\mathrm{i}2\pi f_k \cdot \frac{n}{f_s}} = Y[k]e^{\mathrm{i}2\pi f_k \cdot t_n}$$

Dokončete kód:

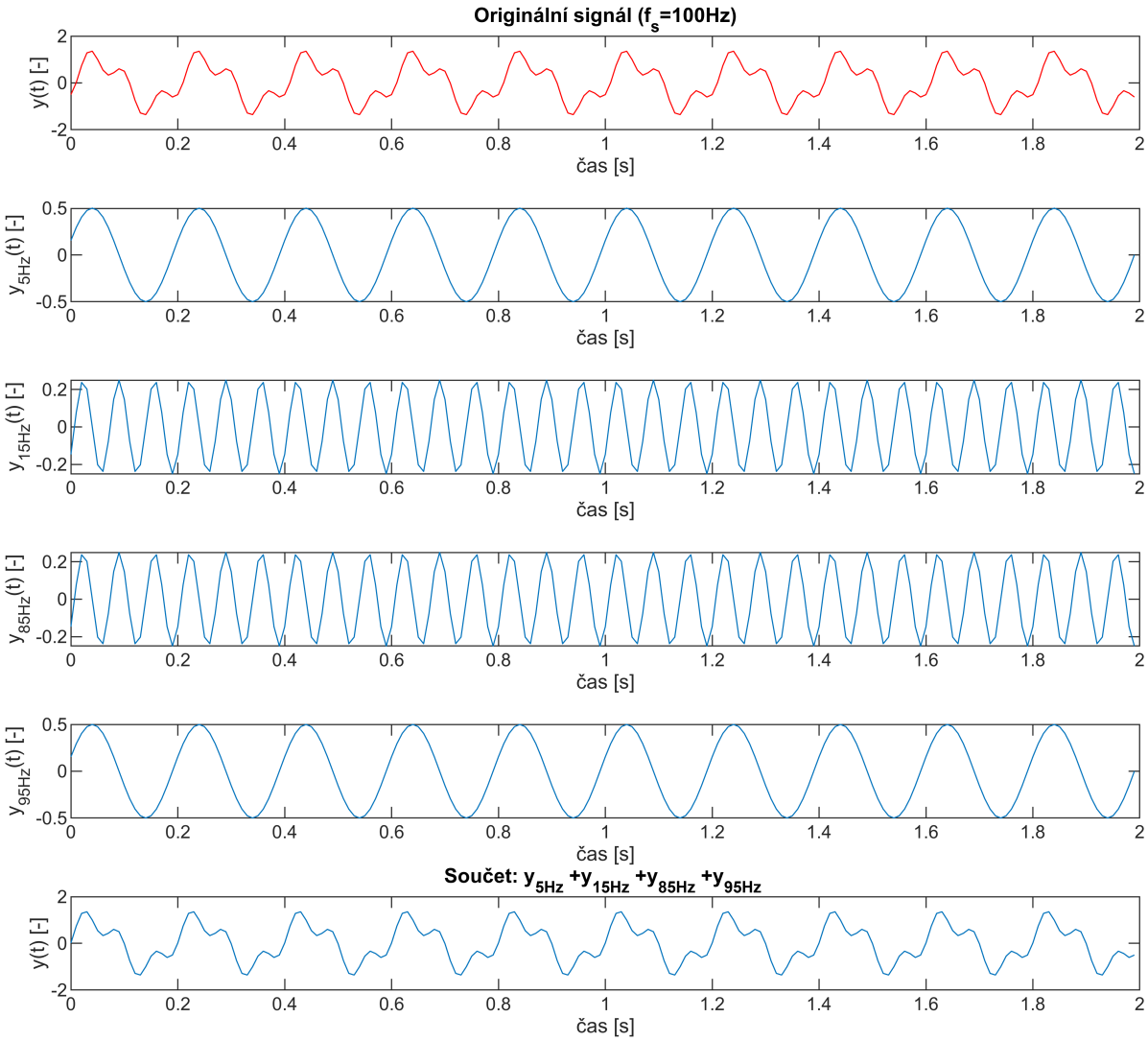
```
y1=Y(k(1))*exp(1i*2*pi*f(k(1)).*(n/fs)); %pro Y(k(1)) a f(k(1))
y2=Y(k(2))*exp(1i*2*pi*f(k(2)).*(n/fs)); %pro Y(k(2)) a f(k(2))
y3=Y(k(3))*exp(1i*2*pi*f(k(3)).*(n/fs)); %pro Y(k(3)) a f(k(3))
y4=Y(k(4))*exp(1i*2*pi*f(k(4)).*(n/fs)); %pro Y(k(4)) a f(k(4))

y1=real(y1); % pouze reálná část kvůli numerické nepřesnosti
y2=real(y2); % pouze reálná část kvůli numerické nepřesnosti
y3=real(y3); % pouze reálná část kvůli numerické nepřesnosti
y4=real(y4); % pouze reálná část kvůli numerické nepřesnosti
```

Sečtěte všechny komponenty $y'[t_n] = y_1[t_n] + y_2[t_n] + y_3[t_n] + y_4[t_n]$**, všechny průběhy zobrazte v subplot 6 x 1 a součet porovnejte s originálním signálem** $y[t_n] \Leftrightarrow y'[t_n]$:




```
y_sum = y1 + y2 + y3 + y4;
figure
set(gcf, 'Position', [0 0 800 700]);
subplot(6,1,1);
plot(t,yt,Color="red")
title("Originální signál (f_s=100Hz)")
ylabel("y(t) [-]")
xlabel("čas [s]")

subplot(6,1,2);
plot(t,y1)
ylabel("y_{5Hz}(t) [-]")
xlabel("čas [s]")

subplot(6,1,3);
plot(t,y2)
ylabel("y_{15Hz}(t) [-]")
xlabel("čas [s]")

subplot(6,1,4);
plot(t,y3)
ylabel("y_{85Hz}(t) [-]")
xlabel("čas [s]")

subplot(6,1,5);
plot(t,y4)
ylabel("y_{95Hz}(t) [-]")
```

```
xlabel("čas [s]")

subplot(6,1,6)
plot(t,y_sum)
title("Součet: y_{5Hz} +y_{15Hz} +y_{85Hz} +y_{95Hz} ")
ylabel("y(t) [-]")
xlabel("čas [s]")
```



Odpovězte na otázky:

1) vysvětlete, proč harmonická složka $y_{5\mathrm{Hz}}[t_n]$ vypadá jako $y_{95\mathrm{Hz}}[t_n]$, respektive $y_{15\mathrm{Hz}}[t_n]$ jako $y_{85\mathrm{Hz}}[t_n]$:

Frekvenční spektrum DFT je periodické s periodou odpovídající vzorkovací frekvenci (fs =100Hz) a symetrické kolem fs/2 (50Hz).

Při vzorkovací frekvenci 100Hz nejde rozeznat zda se jedná o harmonickou složku s frekvencí 5Hz (15Hz) nebo 95Hz (85Hz).

2) Jaká je perioda zobrazeného signálu $y_{95\mathrm{Hz}}[t_n]$ a tedy jeho frekvence? Jak a proč se liší frekvence zobrazená a frekvence odpovídající spektrální čáře 95 Hz?

Perioda zobrazeného signálu y95Hz je 0.2s, což odpovídá frekvenci 5Hz. Zobrazený signál byl vzorkován frekvencí 100Hz, což porušuje Nyquistovo kritérium pro frekvenci odpovídající spektrální čáře 95Hz. Došlo tedy k aliasingu a signál se zobrazil s frekvencí 5Hz.

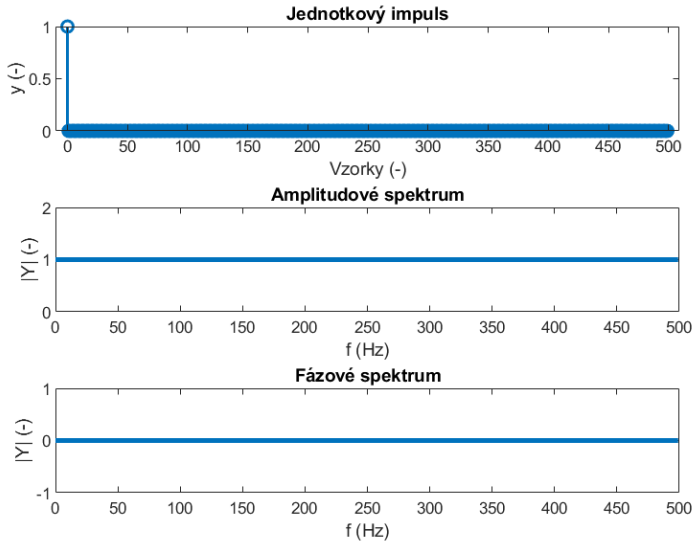
Užitečné signály a jejich spektra

Jednotkový impuls

Jednotkový impuls je první pohled jednoduchý a nenápadný signál, kde první prvek je roven jedné a zbytek je nulových, nabývá ve spektru konstanty (tzn. má všechny kmitočty). Můžeme samozřejmě signál časově zpozdit a tedy jednička nebude na první pozici, ale stále platí, že všechny ostatní vzorky jsou nulové. Jednotkový impuls budeme využívat pro popis digitálních systémů v následujících cvičení. Také se používá v syntéze periodických signálů (např. u řeči na znělé hlásky).

Pozn.: Pro popis analogových systémů by se použil Dirakův impuls, který v čase 0 má nekonečnou amplitudu a ve zbytku času je nulový. Spektrálně potom také odpovídá konstantě.

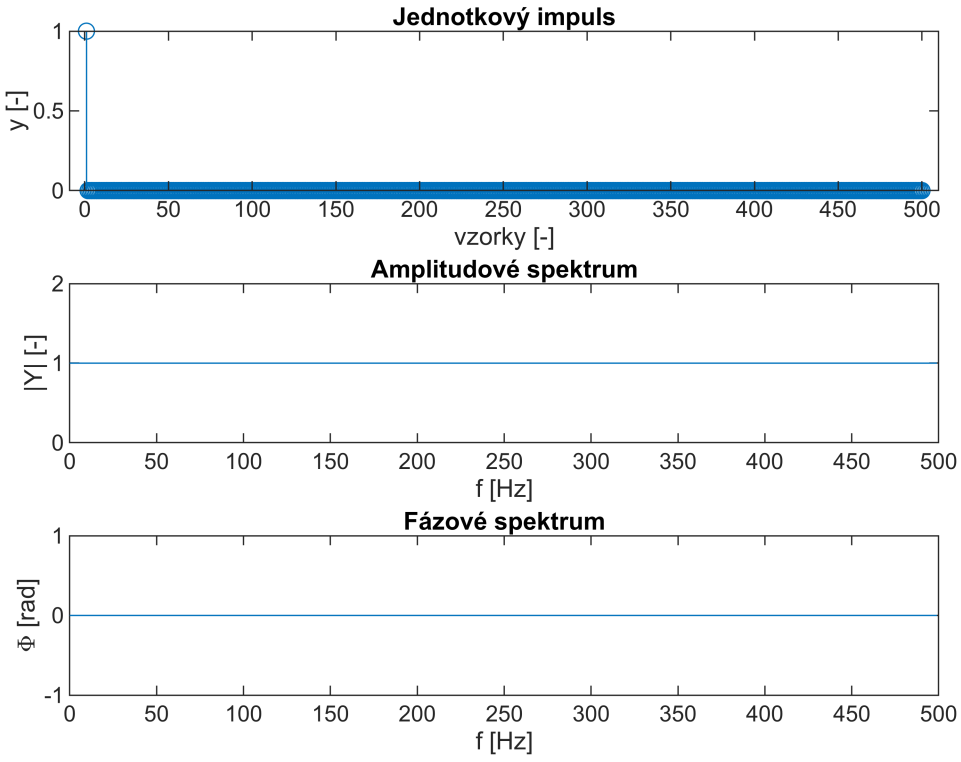
Vygenerujte jednotkový impuls pro *N* = 500 a $f_s = 500$ Hz, jeho NEnormované amplitudové spektrum, fázové spektrum a zobrazte do jednoho obrázku




```
N = 100; % počet vzorků
jednotkovy_impuls = zeros(N, 1); % vygenerování nulového vektoru
jednotkovy_impuls(1) = 1; % náhrada prvního prvku jedničkou

% alternativní zápis
jednotkovy_impuls = [1; zeros(N-1, 1)];
```

```
N = 500; % počet vzorků
fs = 500; % Hz, vzorkovací frekvence
t = 1:N;
jednotkovy_impuls = [1 zeros(N-1,1)'];
```

```
Y = fft(jednotkovy_impuls);
f = linspace(0,fs-fs/N,N);
figure
subplot(3,1,1)
stem(t, jednotkovy_impuls)
title("Jednotkový impuls")
ylabel("y [-]")
xlabel("vzorky [-]")

subplot(3,1,2)
plot(f,abs(Y))
title("Amplitudové spektrum")
ylabel("|Y| [-]")
xlabel("f [Hz]")

subplot(3,1,3)
plot(f, angle(Y))
title("Fázové spektrum")
ylabel("\Phi [rad]")
xlabel("f [Hz]")
```



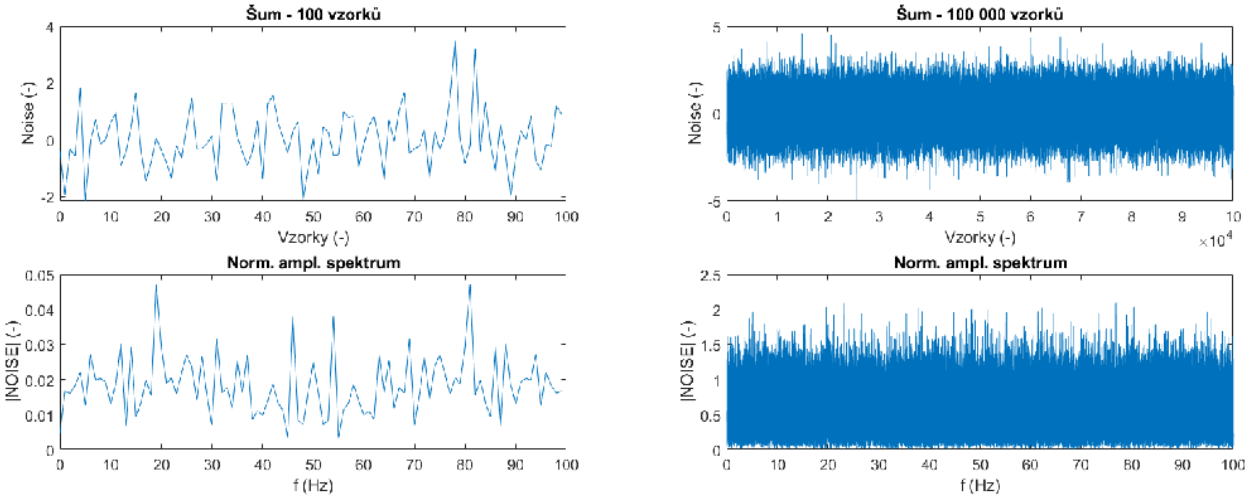
Šum s normálním rozdělením

Šum s normálním rozdělením (nebo také šum s Gaussovským rozdělením, nebo bílý šum) je nejrozšířenější šum. Spektrálně, podobně jako jednotkový impuls, obsahuje všechny frekvence, a proto se také široce využívá (např. muzice jako podklad, u syntézy řeči na neznělé hlásky, testování systému na odolnost proti šumu, atp.).

Vygenerujte bílý šum:

1. **o *N* = 100 vzorků a** $f_s = 100$ **Hz,**
2. **o *N* = 100 000 vzorků a** $f_s = 100$ **Hz.**

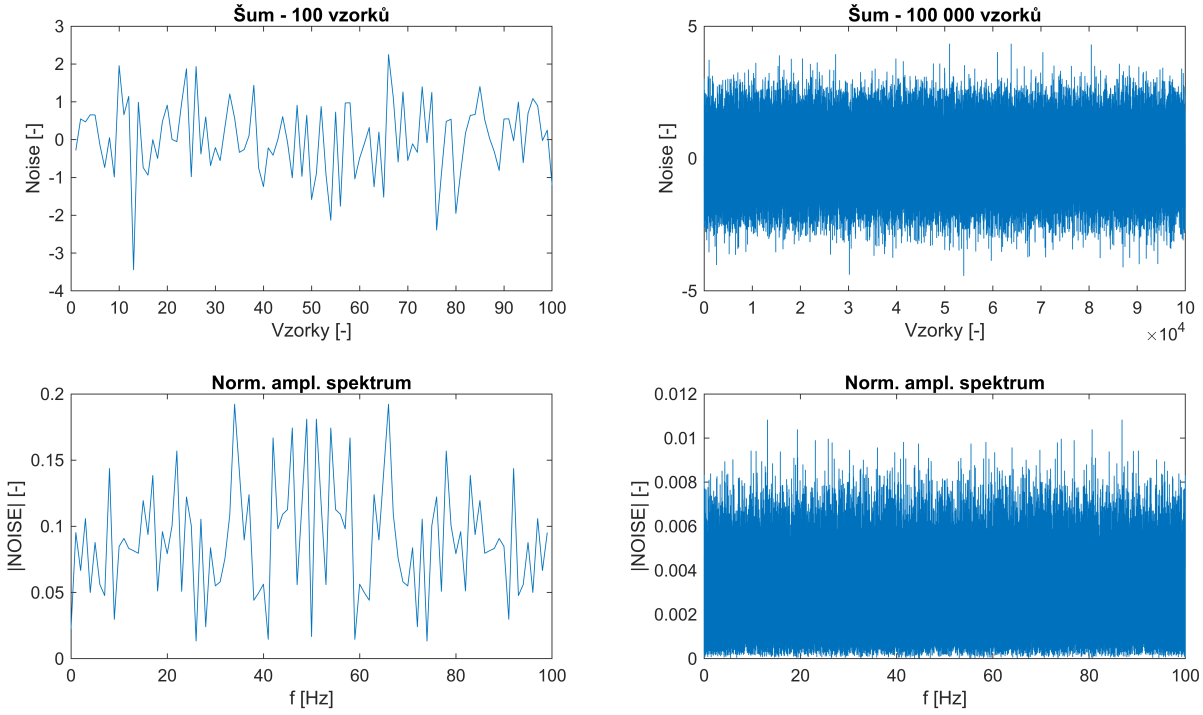
Vykreslete signály a normované amplitudové spektrum.




```
% bílý šum
% NEPOJMENOVÁVAT sum !!! předefinujete si tím funkci sčítání
N = 100;
bily_sum = randn(N,1);
```

```
N = 100;
noise = randn(N,1);
N2 = 100000;
NOISE = randn(N2,1);
fs = 100;

Y1 = (1/N)*fft(noise);
Y2 = (1/N2)*fft(NOISE);

f = linspace(0,fs-fs/N,N);
f2 = linspace(0,fs-fs/N2,N2);
figure
set(gcf, 'Position', [0 0 1300 700]);
subplot(2,2,1);
plot(1:N,noise)
xticks(0:N/10:N)
title("Šum - 100 vzorků")
ylabel("Noise [-]")
xlabel("Vzorky [-]")

subplot(2,2,2)
plot(1:N2, NOISE)
xticks(0:N2/10:N2)
title("Šum - 100 000 vzorků")
ylabel("Noise [-]")
xlabel("Vzorky [-]")
```

```
subplot(2,2,3)
plot(f,abs(Y1))
title("Norm. ampl. spektrum")
ylabel("|NOISE| [-]")
xlabel("f [Hz]")

subplot(2,2,4)
plot(f2,abs(Y2))
title("Norm. ampl. spektrum") %amplituda spektra velikostně neodpovídá
ukázkovému grafu, ale dle mého by měla být velikost amplitudy po normalizaci
v mém grafu správně
ylabel("|NOISE| [-]")
xlabel("f [Hz]")
```



Bonus (1b)

Obdélníkový signál je velmi jednoduchý technický signál, jež je v digitální a měřicí technice hojně využíván.

Obdélníkový signál

Obdélníkový signál můžeme vytvořit několika způsoby. Např. maticově spojením jedničkového a nuluvého vektoru (`zeros`, `ones`), pomocí funkcí sinus a signum (s dodatkem pro nulu), ale můžeme využít i funkce z Signal processing toolboxu - `square`.

Vytvořte obdélníkové průběhy s základním kmitočtem $f_0 = 2$ Hz, délkou signálu $T = 3$ s a vzorkovacím kmitočtem $f_s = 50$ Hz:

1. **bez posunu fáze, s minimem -1 V a maximem +1 V a střídou 0,5** (tzn. nulová stejnosměrná složka, střída je poměr časů obdélníka v hodnotě maxima a periody),
2. **bez posunu fáze, s minimem 0 V a maximem +2 V a střídou 0,5** (tzn. stejnosměrná složka rovna jedné),
3. **s posunem fáze o $\frac{\pi}{2}$, s minimem -1 V a maximem +1 V a střídou 0,5.**

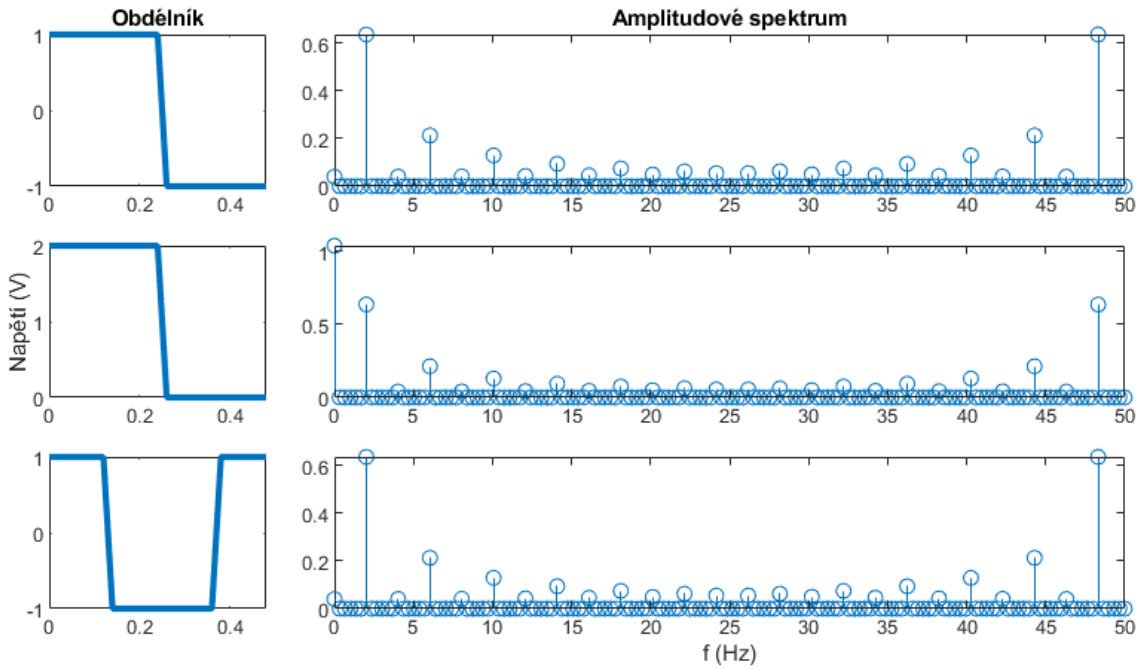
Pozn.: Zbylé dva signály snadno vytvoříte z prvního signálu. Buď přičtením konstanty nebo pomocí indexace.

```
f = 2;
fs = 50;
T = 3;
N = T/(1/fs)
```

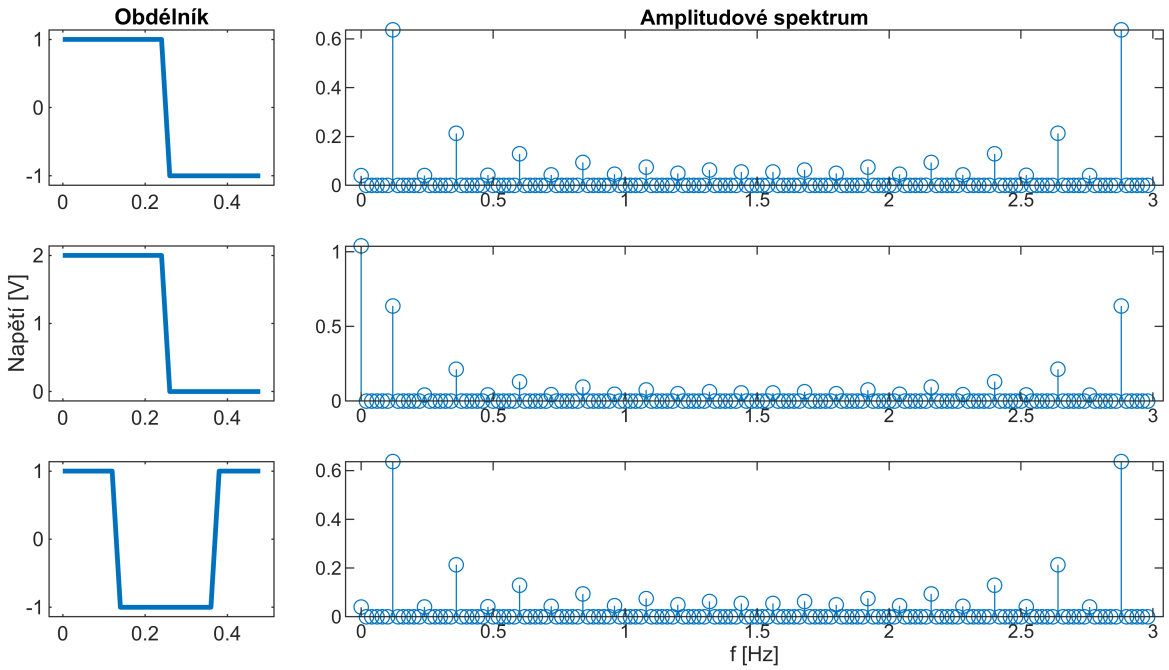
```
N =
150
```

```
t = 0:1/fs:T-1/fs;
signal1 = square(2*pi*f.*t);
signal2 = 1+signal1;
signal3 = square(2*pi*f.*t + pi/2);
y1 = (1/N).*fft(signal1);
y2 = (1/N).*fft(signal2);
y3 = (1/N).*fft(signal3);

T = 1/f;
idx = t<T ;
one_period1 = signal1(idx);
one_period2 = signal2(idx);
one_period3 = signal3(idx);
t_one_period= t(idx);
```

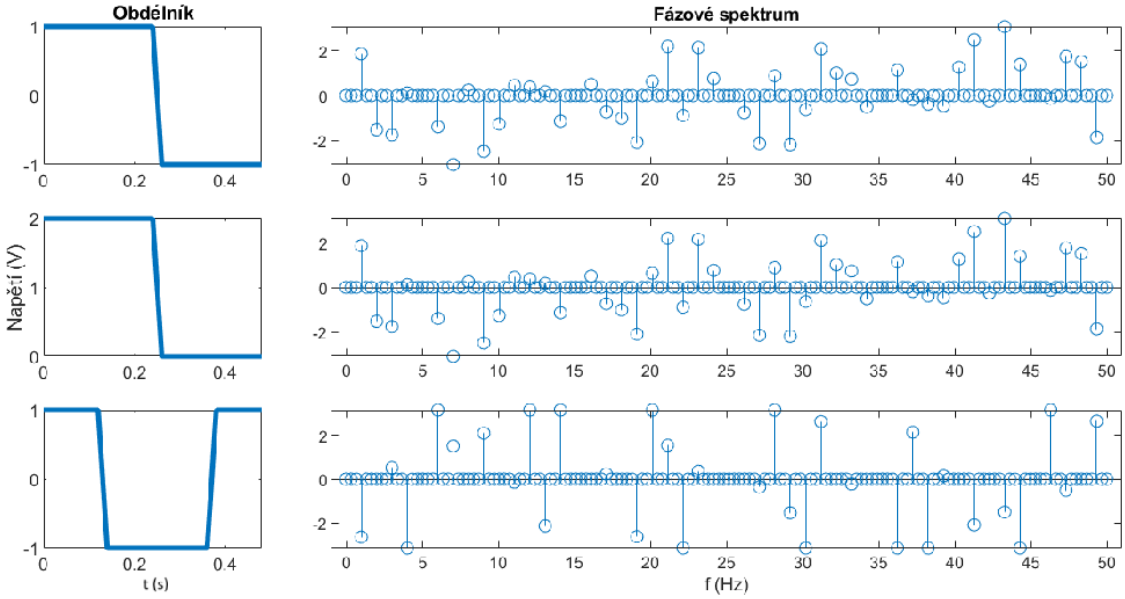
Do jednoho obrázku pod sebe vykreslete první periody signálů a amplitudové spektra (z celých signálů, mohou být nenormovaná).




```
figure
set(gcf, 'Position', [0 0 800 400]);
subplot(3,4,1)
plot(t_one_period, one_period1,LineWidth=2)
axis padded
title("Obdélník")

subplot(3,4,[2 3 4])
stem(t,abs(y1))
title("Amplitudové spektrum")

subplot(3,4,5)
plot(t_one_period, one_period2,LineWidth=2)
axis padded
ylabel("Napětí [V]")

subplot(3,4, [6 7 8])
stem(t,abs(y2))

subplot(3,4,9)
plot(t_one_period, one_period3,LineWidth=2)
axis padded

subplot(3,4, [10 11 12])
stem(t,abs(y3))
xlabel("f [Hz]")
```

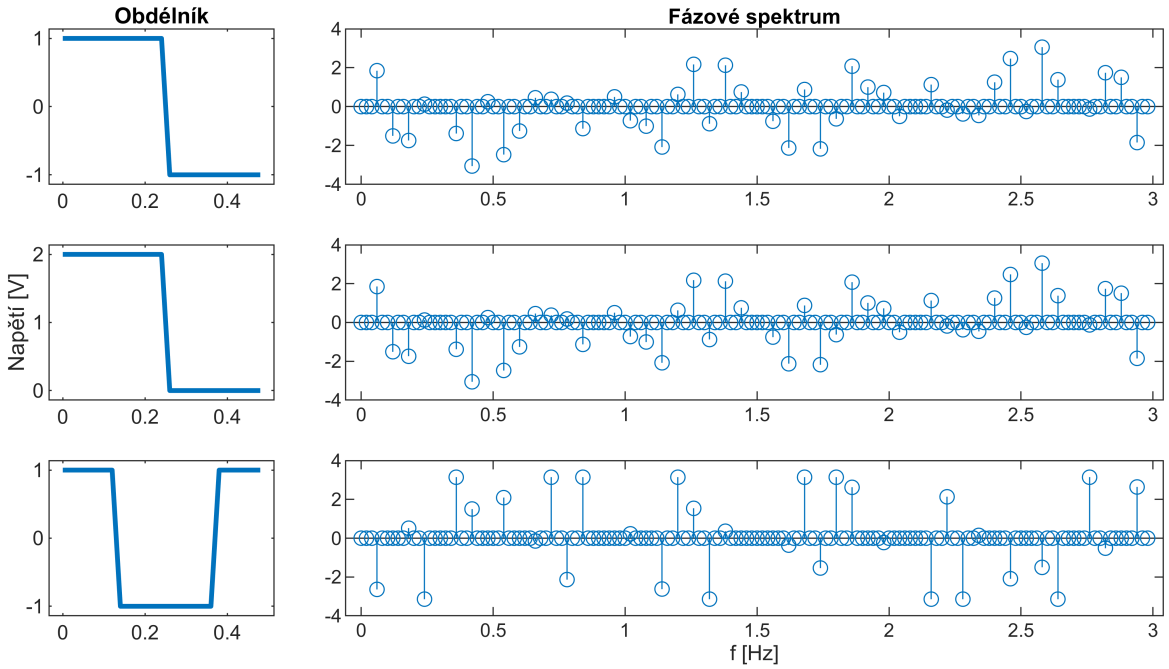


Stejně pro fázové spektrum




```
figure
set(gcf, 'Position', [0 0 800 400]);
subplot(3,4,1)
plot(t_one_period, one_period1,LineWidth=2)
axis padded
title("Obdélník")

subplot(3,4,[2 3 4])
stem(t,angle(y1))
title("Fázové spektrum")

subplot(3,4,5)
plot(t_one_period, one_period2,LineWidth=2)
```

```
axis padded
ylabel("Napětí [V]")

subplot(3,4, [6 7 8])
stem(t,angle(y2))

subplot(3,4,9)
plot(t_one_period, one_period3,LineWidth=2)
axis padded

subplot(3,4, [10 11 12])
stem(t,angle(y3))
xlabel("f [Hz]")
```



Pozn.: Ačkoli se jedná o stejný průběh pouze s rozdílem v amplitudě nebo v fázi, i patřičná spektra se budou lišit. Např.pro první dva signály je rozdíl ve stejnosměrné složce, což se projeví rozdílným magnitudovým spektrem, ale stejným fázovým. Obráceně, magnituvové spektrum prvního a třetího je stejné, liší se však fázové.