Programové vybavení pro frekvenční analýzu zvukového signálu

Aleš Doležal

Bakalářská práce
2006

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

nascannované zadání s. 1

nascannované zadání s. 2

ABSTRAKT

Cílem této bakalářské práce je vytvoření programového vybavení pro frekvenční analýzu zvukového signálu. V první části je rozebráno teoretické pozadí problému, který se bude řešit v praktické části. V ní je předmětem popisu způsob získání vstupních dat, jejich analýza a následné vyhodnocení. Posledním bodem je zhodnocení výsledné aplikace.

Klíčová slova: frekvenční analýza, Fourierova transformace, váhovací okna, Advantech, FFTW

ABSTRACT

The aim of this Bachelor thesis is to create software equipment for frequency analysis of sound signal. The first part is focused on theoretical background of the problem, which will be solved in second part. In this part input data obtaining, their analysis and evaluation of the analysis results are described. Last point deals with estimation of the resulting application.

Keywords: frequency analysis, Fourier transform, measuring window, Advantech, FFTW

Poděkování

V první řadě moje díky patří mému odbornému vedoucímu Ing. Petru Dostálkovi za rady, připomínky, nekonečnou trpělivost a optimismus při těch mnoha zkouškách testovacích verzí programu. Dále děkuji Ing. Janu Dolinayovi za množství rad týkajících se programování v OS Windows a v neposlední řadě i za opravy mé neodborné terminologie.

Motto

„

Nikde po celém vesmíru, na tisících planet, nenajdou se lidé, aby se s námi podělili o naši osamělost. Zůstane možná inteligence, možná přetrvá moc, plující oblak naší zkázy budou možná prázdně sledovat odkudsi z hlubin vesmíru ohromná zařízení, jejichž tvůrci po nás toužili stejně, jako po nich toužíme my. Jenže odpověď na svou otázku dostali jsme už v samé podstatě evoluce. Ze všech tvorů, ať už kamkoli dospěli, nikdo tu nezůstane věčně...

“

Loren Eiseley

z knihy *The Immense Journey*

Prohlašuji, že jsem na celé bakalářské práci pracoval samostatně a použitou literaturu jsem citoval.

V Malém Beranově dne 10. června 2006 .....

Aleš Doležal

OBSAH

ÚVOD

Digitální zpracování signálů je jedním z moderních a prudce se rozvíjejících vědních oborů. Je rozprostřeno přes téměř všechny aspekty lidského života. Jen namátkou jmenujme oblasti jejího výskytu: komunikace, medicína, kontrola výrobků, komprese zvuků, obrázků a videa, či pátrání po ložiscích nerostů.

Tématem této práce je analýza zvukového signálu, což je spolu s výše zmíněnou kompresí zvuku, hlasovou telekomunikací či syntézou řeči, téma, kterým se zabývá obor zpracování zvuku.

Zvuk nás všechny obklopuje po celý život, sluch je jedním z pěti základních smyslů člověka. Proto je jeho analýza natolik důležitou. Mějme kupříkladu kuličkové ložisko bez zjevné viditelné vnější vady. Pomocí spektrální analýzy zvuku točícího se ložiska můžeme díky přítomnosti tónů o určitých frekvencích říci, že ložisko je vadné. Další oblastí, kde se analýza zvuku uplatňuje je rozpoznání hlasu, kde díky frekvenčnímu „otisku“ můžeme porovnáním se známými vzorky písmen určit co bylo řečeno. Komprese zvuku je pak založena na odstraňování frekvencí, které díky tomu, že jsou například umístěny v těsné blízkosti jiné, dominantní složky, nejsou lidským uchem postřehnutelné.

Frekvenční analýza zvukového signálu je vystavěna na možnostech Fourierovy transformace, či její speciální verze, rychlé Fourierovy transformace. Dalším stavebním kamenem je použití speciálních okenních funkcí, sloužících k určité žádané změně analyzovaného signálu. V ten okamžik mluvíme o krátkodobé Fourierově transformaci. Výše zmíněná témata jsou v této práci zpracována v její teoretické části.

Praktická část má jednoznačného jmenovatele, jedná se o aplikaci *f_analyzer*, konečný produkt a cíl této bakalářské práce. Je rozebrána vstupní brána dat programu, technologická karta Advantech. Dále jeho analytické srdce, knihovna FFTW. Následuje podrobný popis samotné aplikace.

I. TEORETICKÁ ČÁST

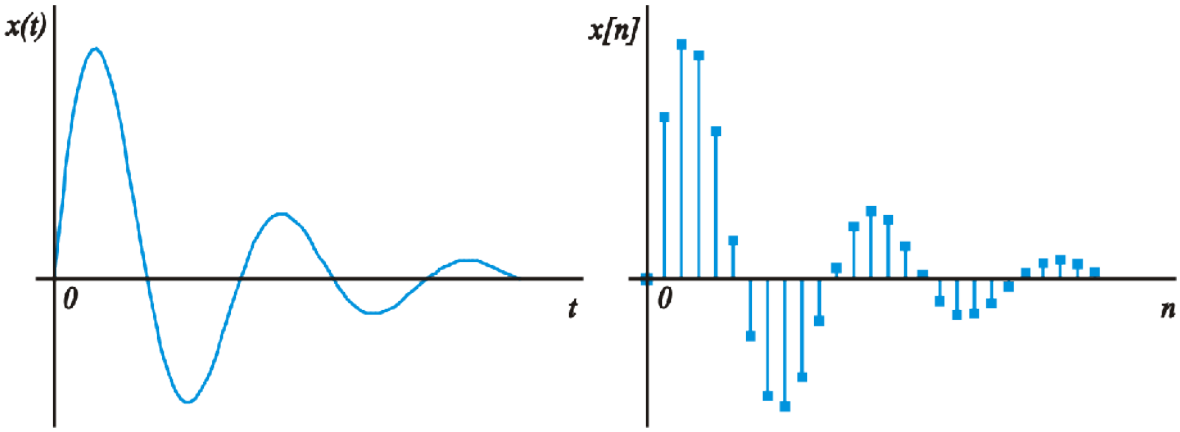
1 ZVUKOVÉ SIGNÁLY

1.1 Rozdělení zvukových signálů

Jevy, které probíhají v čase nazýváme v přírodních a technických vědách děje, popř. procesy. Signály pak nazýváme tyto děje převedené na změny fyzikálních veličin vhodných pro jejich přenos, analýzu a další zpracování. Tímto způsobem získáváme signály elektrické, zvukové, optické atd. Dále se budeme zabývat zvukovými signály.[7]

Signály se dají rozdělovat na základě dvou hlavních hledisek. Prvním hlediskem je spojitost. Signály se podle spojitosti dělí na spojité (analogové) a diskrétní (digitální).

Spojité signály jsou takové signály, které se vyskytují v analogových obvodech a jsou popsány spojitými funkcemi nezávisle proměnné, což je v drtivé většině případů čas. Naproti tomu diskrétní signály jsou popsány posloupnostmi, kde nezávisle proměnná nabývá pouze celočíselných hodnot. Příklady signálu ilustruje obrázek (Obr. 1).



Obr. 1. Spojitý a diskrétní signál

Jak spojité tak diskrétní signály můžeme dělit na základě jejich charakteru, matematického popisu.

První skupinou jsou deterministické signály. Tyto jsou popsány funkcemi či posloupnostmi a jejich hodnotu můžeme pro daný okamžik přesně vypočítat. Jedná se například o harmonický výstupní signál z generátoru.

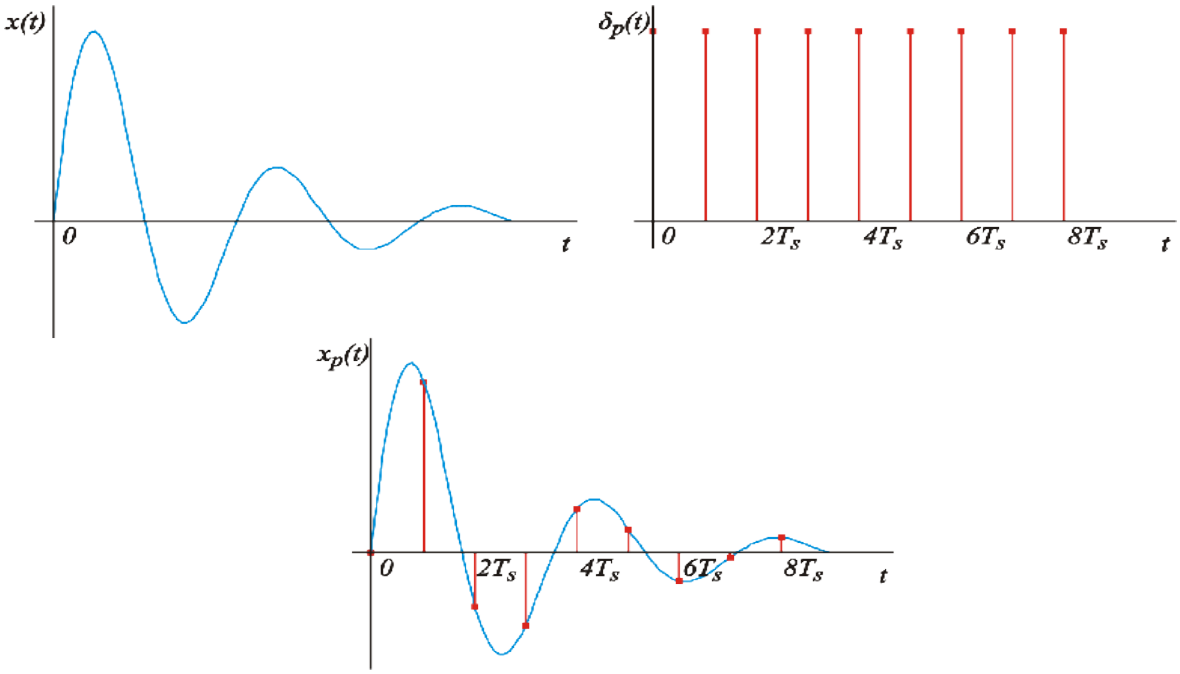
Druhým typem jsou signály náhodné (stochastické) které je možno popisovat pouze pravděpodobnostními či statistickými metodami.

1.2 Diskrétní zpracování zvukových signálů

Analogový signál pro zpracování počítači, což je i případ této práce, je nutné mít převeden do digitálního tvaru. Spojitý signál je při své cestě za digitalizací vzorkovám a kvantován. Tyto dvě operace jsou lineární, tudíž je možno je provést v libovolném pořadí.

1.2.1 Vzorkování

Vzorkování neboli diskretizace je proces při němž ve stanovených časových okamžicích přiřazujeme signálu $x_p(t)$ určité funkční hodnoty signálu $x(t)$. Vzorkování je znázorněno na obrázku (Obr. 2). Na obrázku můžeme pozorovat signál $x(t)$, dále pak posloupnost Dirackových impulsů $\delta_p(t)$, a výsledný diskrétní signál $x_p(t)$.



Obr. 2. Vzorkování spojitého signálu

Vzorkování je zde vyjádřeno jako násobení spojitého signálu posoupností Dirackových impulsů. Pro výstupní signál tedy platí

$$x_p(t) = x(t) \cdot \delta_p(t)\,, \tag{1}$$

kde

$$\delta_p(t) = \sum_{n=-\infty}^{\infty} \delta \cdot (t - nT_s)\,. \tag{2}$$

Platí tedy

$$x_p(t) = \sum_{n=-\infty}^{\infty} x(nT_s) \cdot \delta \cdot (t - nT_s) \tag{3}$$

1.2.2 Kvantování

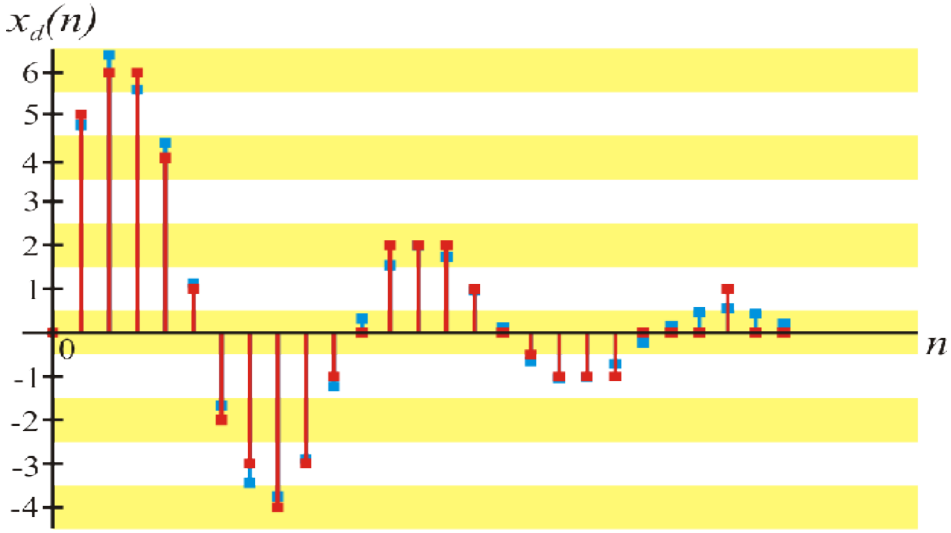
Jedná se o proces při němž se převádí v amplitudě spojitý signál na diskrétní. Při převodu diskretizovaného signálu $x_p(t)$ na jeho digitální reprezentaci x[n] je nutné všechny funkční hodnoty převést na menší, konečný, počet úrovní. Pro zobrazení digitálního signálu se téměř vždy využívá dvojková soustava, proto je počet úrovní, rozlišovací schopnost převodníku, dán jako N-tá mocnina čísla 2. Velikost kvantizačního intervalu (úrovně) q je pak

$$q = 2^{(-N)} \cdot U, \tag{4}$$

kde U je vstupní rozsah a N počet bitů převodníku. Výsledkem kvantování je plně digitální signál, lišící se od původního chybou, její velikost je v případě zaokrouhlování

$$\varepsilon = \left\langle \frac{-q}{2}; \frac{q}{2} \right) \tag{5}$$

Signál sestavený z těchto chyb je nazýván kvantizační šum.



Obr. 3. Kvantování signálu

Na obrázku (Obr. 3) je uveden příklad kvantování signálu. Modře je původní signál. Červeně pak signál kvantovaný. Kvantizační intervaly jsou znázorněny bílými a žlutými pásy.

2 FOURIEROVY TRANSFORMACE

Jean Baptista Joseph Fourier (*1768 - †1830) byl Francouzský matematik, fyzik, spisovatel, ale i politik. V roce 1807 dokončil svou práci *O šíření tepla v pevných látkách* v níž navrhnul harmonické sinusové signály jako slibný prostředek pro popis šíření tepla tělesem. Představa harmonické analýzy však nebyla zpočátku vědeckou obcí přijímána. Mezi kritiky byli takoví matematici jako Joseph Lagrange či Pierre Laplace. Práce mohla být zveřejněna až 15 let po dokončení, až poté co její největší odpůrce Lagrange zemřel. Posléze, kdy se ukázala jako platná, ji zpřesňovali další matematici. I přes to že sám Fourier mnoho k teorii Fourierových řad nepřinesl, byla významným skokem v celé teorii signálů.

Obr. 4. Jean Fourier

2.1 Podstata Fourierovy transformace

Každý signál je možno reprezentovat v jeho časové oblasti, ale také v duální oblasti frekvenční. Podstatou Fourierovy transformace je, že funkci v časové oblasti přetransformujeme do oblasti frekvenční a získáme tak jeho frekvenční spektrum. Fourierova transformace vychází z předpokladu, že každý signál lze vyjádřit jako superpozici nekonečně mnoha sinusových signálů. Výsledkem Fourierovy transformace jsou Fourierovy koeficienty, které nám říkají, jak daný sinusový průběh přispívá k celkovému signálu, což je vlastně její amplituda. Hlavním cílem Fourierovy transformace je tedy najít složky a jejich amplitudy.[14]

2.2 Fourierova transformace ve spojité oblasti

Fourierova transformace je vyjádřena tzv. Fourierovým integrálem

$$X(j\omega)=\int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt\,, \tag{6}$$

který existuje, je-li funkce *x(t)* absolutně integrovatelná

$$\int_{-\infty}^{\infty} |x(t)|\,dt<\infty\,. \tag{7}$$

Zpětná transformace je pak definovaná vztahem

$$x(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty} X(j\omega)e^{j\omega t}d\,\omega\,. \tag{8}$$

2.3 Diskrétní Fourierova transformace

Tato transformace se využívá v případě číslicového zpracování signálu. Pak pracujeme s konečným počtem hodnot a to jak v časové tak frekvenční oblasti. Signál má v obou oblastech stejný počet hodnot *N*. V anglicky psané literatuře je transformace označována jako Discrete Fourier Transform (DFT). Uvažujme tedy komplexní řadu *x[n]* o *N* prvích

$$x_{0,}x_{1,}x_{2,}x_{3,}\ldots,x_{N-1}\,, \tag{9}$$

kde každé *x* je komplexní číslo

$$x_i=x_{Re}+j\cdot x_{Im}\,. \tag{10}$$

Za předpokladu, že řada vně rozsahu *0* až *N-1* je periodické prodloužení základního intervalu, pak můžeme DFT definovat jako

$$X[k]=\sum_{n=0}^{N-1} x[n]e^{-jn\Theta_k}=\sum_{n=0}^{N-1} x[n]e^{\frac{-j2\pi nk}{N}}\,,k=0,1,\ldots,N-1\,, \tag{11}$$

pak zpětná transformace je

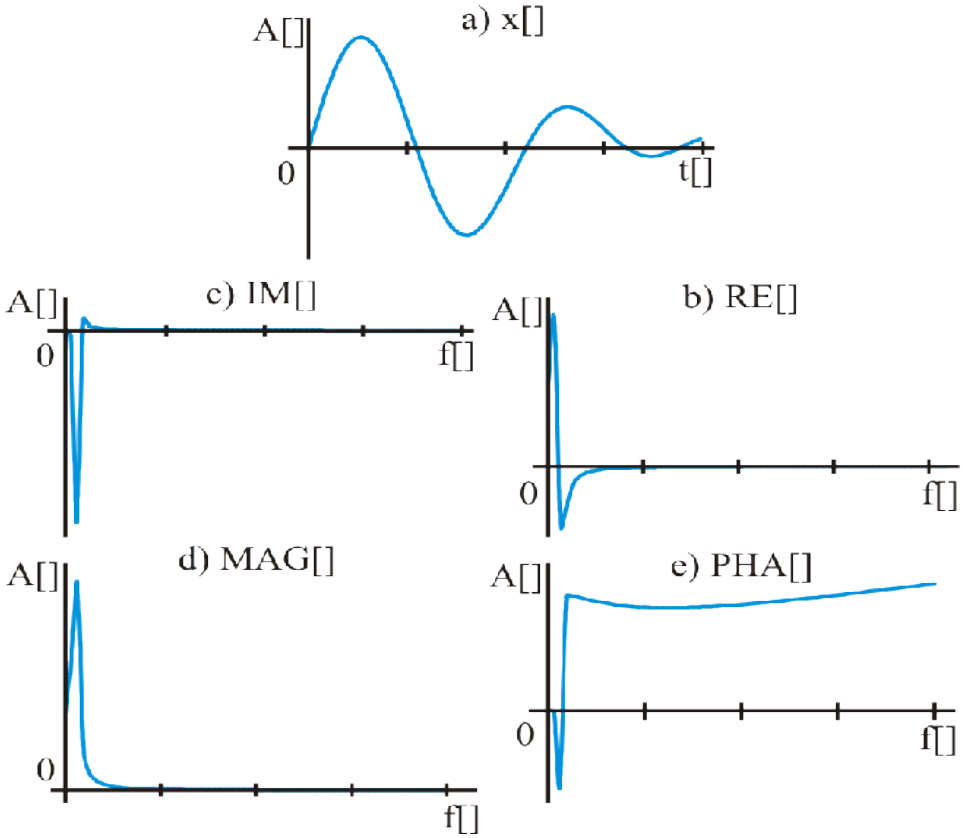
$$x[n]=\frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{jn\Theta_k}=\frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{\frac{j2\pi k}{N}}\,,n=0,1,\ldots,N-1\,. \tag{12}$$

Zde popsaná transformace pracuje se vstupní komplexní řadou. Není však proti pravidlům vynulovat imaginární část, naplnit reálnou část a tudíš pracovat jen s reálným signálem. Výsledný signál je opět komplexní řadou a dělí tudíš informaci na dvě části, které však bez převedení do polárních souřadnic nedávají přílišný smysl. Díky převedení do polárních souřadnic můžeme vypočítat amplitudu (magnitude - MAG) a fázový posun (phase - PHA) dle známých vztahů

$$MAG[k]=\sqrt{Re\ X[k]^2 + Im\ X[k]^2}\,, \tag{13}$$

$$PHA[k]=\arctan\left(\frac{Im\ X[k]}{Re\ X[k]}\right). \tag{14}$$

Na obrázku (Obr. 5) je zobrazen příklad DFT. Transformace převádí signál *x[]* (Obr. 5a) z časové oblasti do oblasti frekvenční, z které jsou zobrazeny průběhy reálné části *Re[]* (Obr. 5b) a imaginární *Im[]* (Obr. 5c). Na obrázku (Obr. 5d) je vidět amplitudový průběh a průběh fázového posunu (Obr 5e).



Obr. 5. příklad DFT

2.3.1 Symetrie a redundance výstupního signálu

Pokud *x[n]* bude reálný signál (např. po výše zmíněné úpravě komplexního signálu), pak výstup *X[k]* bude konjungovaně sudý (konjungovaně symetrický), čili

$$X[k]=X*[-k]=X*[N-k]\,. \tag{15}$$

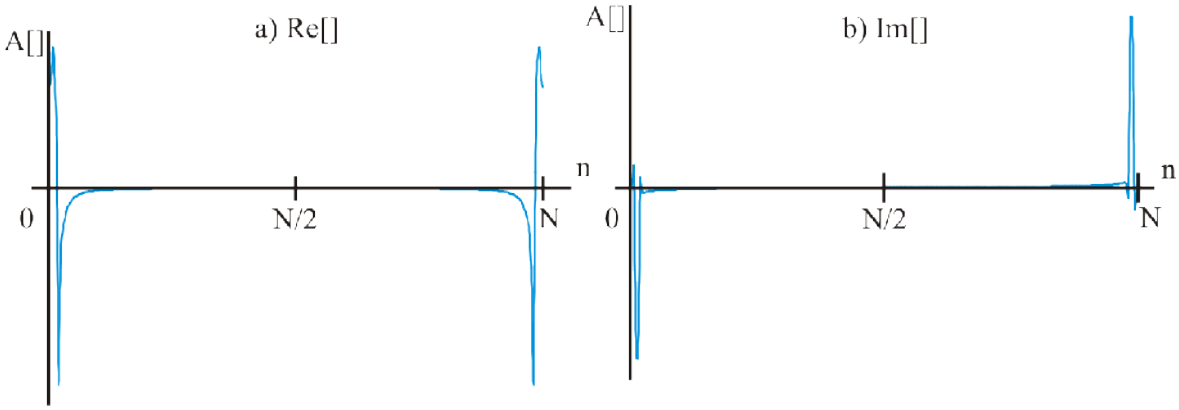
Z (15) vyplývají následujicí vlastnosti spektra

$$Re[X[k]] = Re[X[N-k]]\,, \tag{16}$$

$$Im[X[k]] = -Im[X[N-k]]\,, \tag{17}$$

$$|X[k]| = |X[N-k]|\,. \tag{18}$$

Z toho vyplývá, že stačí pro *X[k]* znát hodnoty $0 \le k \le N/2$. Ostatní hodnoty jsou symetrické a také tím pádem nadbytečné. Jak je vidět z (16) a (17) *Re[X[k]]* je funkce sudá a *Im[X[k]]* je funkcí lichou. Na obrázku (Obr. 6) to je vidět názorně.



Obr. 6. Symetrie a redundance výstupního signálu

2.4 Rychlá Fourierova transformace

V roce 1965 zveřejnili J. W. Cooley a J. W. Tuckey ve své práci *Algoritmus pro strojní výpočet komplexní Fourierovy řady* popis výpočtu rychlé Fourierovy transformace (Fast Fourier Transform - FFT). Pravda je však taková, ža tato technika byla objevena několik desetiletí před rokem 1965. Například Německý matematik Karl Friedrich Gauss ji popsal o více jak sto let dříve. Jeho práce však upadla v zapomění, protože v jeho době neexistovala hlavní věc, pro kterou byla FFT navržena - digitální počítač.

Hlavní výhodou FFT je její značná rychlost oproti DFT. Pro výpočet N hodnot pomocí DFT (11) je potřeba provést N^2 komplexních násobení a $N(N-1)$ komplexních sčítání, takže doba pro výpočet je dána přibližně časem pro výpočet N^2 operací. Naproti tomu doba výpočtu FFT je pouze $N/2 \cdot \log_2(N)$.

2.4.1 FFT algoritmus s decimací v čase

Algoritmus s decimací v čase (angl. *radix 2* nebo *decitation-in-time FFT algorithm*) je nejčastěji používaným FFT algoritmem. Je navržen pro délku transformace $N = 2^m$, kde m je přirozené číslo. Tyto algoritmy využívají periodičnosti a symetrií komplexní exponenciály v (11). Tato exponenciála se pak označuje *otáčecí činitel* a označuje se W_N:

$$W_N = e^{-j2\pi/N}. \tag{19}$$

Pak tedy s užitím (19) přepsat vztah (11) pro DFT jako

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}, \quad k = 0,1,\ldots,N-1. \tag{20}$$

Posloupnost $x[n]$ rozdělíme na dvě posloupnosti, jednu obsahující pouze sudé členy $x_1[n] = x[2n]$ a druhou obsahující pouze liché členy $x_2 = x[2n+1]$, kde $n \in \langle 0, N/2-1 \rangle$. Pak platí[6]

$$X[k] = \sum_{n=0}^{N/2-1} x[2n] W_N^{2nk} + \sum_{n=0}^{N/2-1} x[2n+1] W_N^{2n+1} = \sum_{n=0}^{N/2-1} x_1[n] W_N^{2nk} + W_N^k \sum_{n=0}^{N/2-1} x_2[n] W_N^{2n+1}. \tag{21}$$

Po úpravě pomocí vzhahu $W_N^{2kn} = W_{N/2}^{kn}$ (viz. [6] str. 35) dostáváme

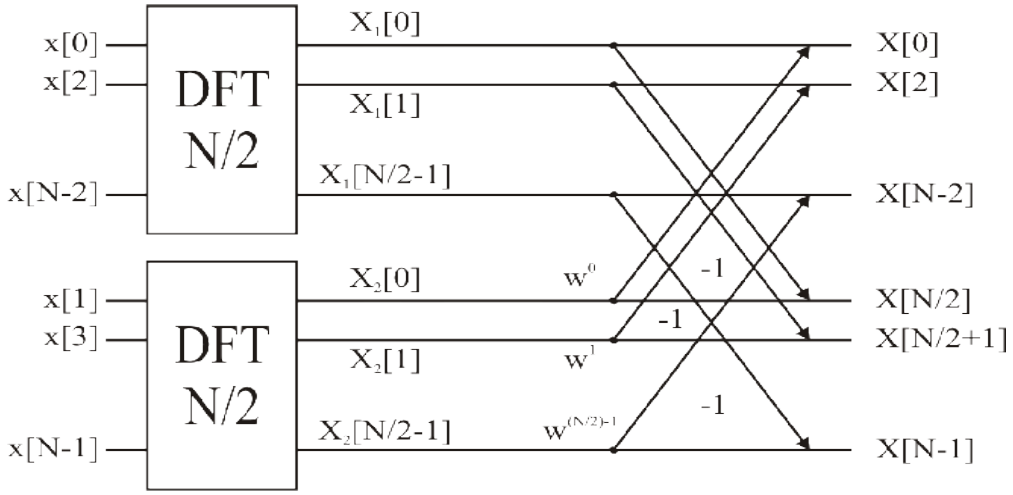
$$X[k] = X_1[k] + W_N^k X_2[k], \quad k = 0,1,\ldots,N-1. \tag{22}$$

Vztah (22) je součet dvou $N/2$ bodových DFT, aplikovaných na vybrané posloupnosti sudých a lichých členů signálů, přičemž $X_2[k]$ je násobena mocninami W_N. Protože jsou $X_1[k]$ a $X_2[k]$ periodické s periodou $N/2$ a W_N^k je periodická s periodou N, lze(22) rozepsat

jako[6]

$$X[k]=X_1[k]+W_N^k X_2[k]$$
$$X[k+N/2]=X_1[k]-W_N^k X_2[k],\quad k=0,1,\ldots,(N/2)-1\,. \tag{23}$$

Vztahy (23) lze graficky znázornit graficky (Obr. 7).



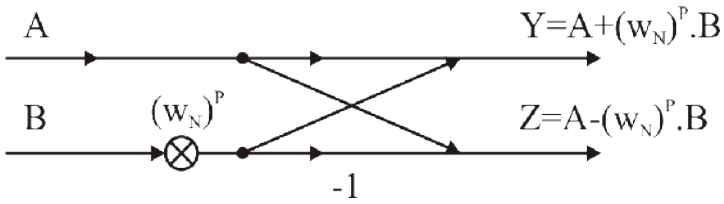
Obr. 7. První krok algoritmu DIT FFT

Rozdělením původní *N* bodové posloupnosti na dvě *N/2* bodové se ušetří 50 % operací. Další úspory lze dosáhnout například dalším dělením obou *N/2* posloupností stejným způsobem jako byl signál rozdělen poprvé. Výsledkem jsou čtyři *N/4* dobové DFT:[6]

$$X_1[k]=U[k]+W_N^{2k}V[k],$$
$$X_1[k+N/4]=U[k]-W_N^{2k}V[k], \tag{24}$$

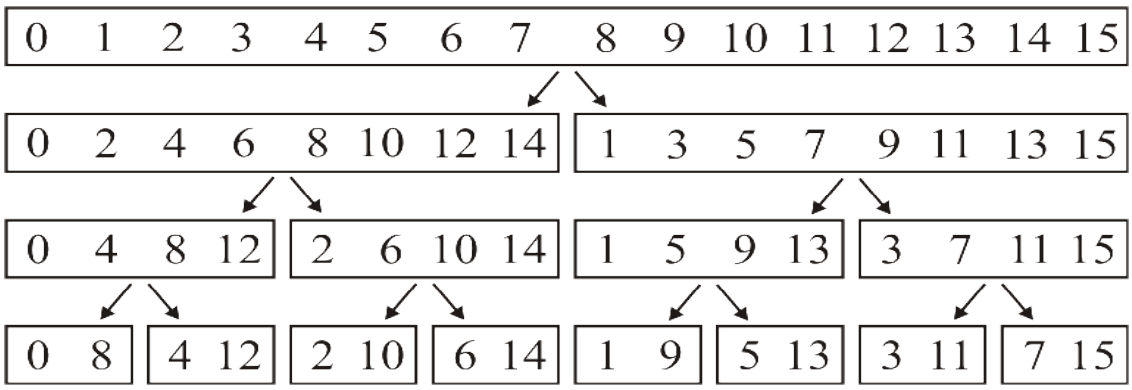
$$X_2[k]=Y[k]+W_N^{2k}Z[k],$$
$$X_2[k+N/4]=Y[k]-W_N^{2k}Z[k],\quad k=0,1,\ldots,(N/4)+1. \tag{25}$$

Vstupní posloupnosti *U[k]* a *V[k]* získáme pomocí DFT sudých a lichých členů $x_1[n]$, posloupnosti *Y[k]* a *Z[k]* pak pomocí sudých a lichých členů $x_2[n]$. Opakováním tohoto postupu se dostaneme až k základní dvojici rovnic, jež vyjadřuje dvoubodovou DFT. Znázornění těchto rovnic v grafu signálových toků (Obr. 8) se podle tvaru nazývá *motýlek DFT* (*DFT butterfly*). [6]



Obr. 8. Motýlek DIT FFT

Na obrázku (Obr. 9) můžeme vidět příklad rozdělování 16ti bodové posloupnosti až na osm dvoubodových posloupností.



Obr. 9. Rozklad signálu pro zpracování FFT

2.5 Krátkodobá Fourierova transformace

Potřeba zavést krátkodobou Fourierovu transformaci (Short Time Fourier Transform - STFT) vzniká v okamžiku, kdy máme potřebu analyzovat signál, u kterého nás zajímá nejen tvar jeho spektra, ale také jeho umístění v čase (tzv. časová lokalizace). Typickým příkladem je analýza hudby či řeči. Právě STFT je nejklasičtějším nástrojem časově-frekvenční analýzy, jejímž výsledkem je krátkodobé spektrum (STFT spectrum).

Klasická Fourierova analýza neumožňuje časovou lokalizaci spektra. Pokud ale v definici Fourierovy transformace použijeme signál vynásobený oknem, jehož poloha vůči signálu se mění s parametrem τ, dostaneme spektrum, jež je funkcí dvou proměnných - ω a τ. Pro Fourierovu transformaci spojitých signálů tedy jde o vztah[6]

$$X(\omega,\tau)=\int_{-\infty}^{\infty} x(t)\cdot w(t-\tau)e^{-j\omega t}dt\,. \tag{26}$$

Váhovací okno *w(t - τ)* potlačuje hodnoty signálu vně své délky, takže získáváme lokální spektra pro jednotlivé hodnoty τ. Posuv signálu o t_0 čili $x(t) \rightarrow x(t - t_0)$ vede k posuvu spektra také o t_0.[6]

Pro diskrétní čas a diskrétní frekvence je STFT spektrum dáno vztahem

$$X[k,m] = \sum_{n=0}^{N-1} x[n] \cdot w[n - mN] e^{-j\frac{2\pi nk}{M}} = \sum_{n=0}^{N-1} x[n] \cdot w[n - mN] W_M^{nk}, \tag{27}$$

kde $k = 0,1,\ldots,M-1$, $n = 0,1,\ldots,N-1$. Funkce (27) je periodická v k s periodou M.

2.5.1 Váhovací okna

V praxi se používá několik základních druhů oken, ve skutečnosti jich existuje velmi mnoho. Každé okno má různé vlastnosti a je vhodné pro jinou aplikaci. Všechny vztahy pro okna zde uvedené jsou v tzv. kauzální formě (tj. pro nezáporná n).

Obdélníkové (jinak taky pravoúhlé či Dirichletovo) okno (Obr. 10a) je nejjednoduším oknem, je popsáno vztahem

$$w[n] = 1, \quad n = 0,1,\ldots,N-1. \tag{28}$$

Jak je vidět ze vztahu (28) toto okno pouze „ořezává“ signál na N vzorků, ale jinak plně zachovává jeho tvar.

Hammingovo okno (Obr. 10b) je v praxi asi nejpoužívanějším oknem. Jeho předpis je

$$w[n] = 0{,}54 - 0{,}46 \cdot \cos\left(\frac{2\pi}{N} \cdot n\right), \quad n = 0,1,\ldots,N-1. \tag{29}$$

Hanningovo (Hannovo nebo raised cosine) okno (Obr. 10c) je speciální případ sinového okna. V praxi je opět velmi používané. Vztah (31) je přepis vztahu (30) pomocí funkce cos.

$$w[n] = \sin^2\left(\frac{\pi}{N} \cdot n\right), \quad n = 0,1,\ldots,N-1. \tag{30}$$

$$w[n] = 0{,}5 - 0{,}5 \cdot \cos\left(\frac{2\pi}{N} \cdot n\right), \quad n = 0,1,\ldots,N-1 \tag{31}$$

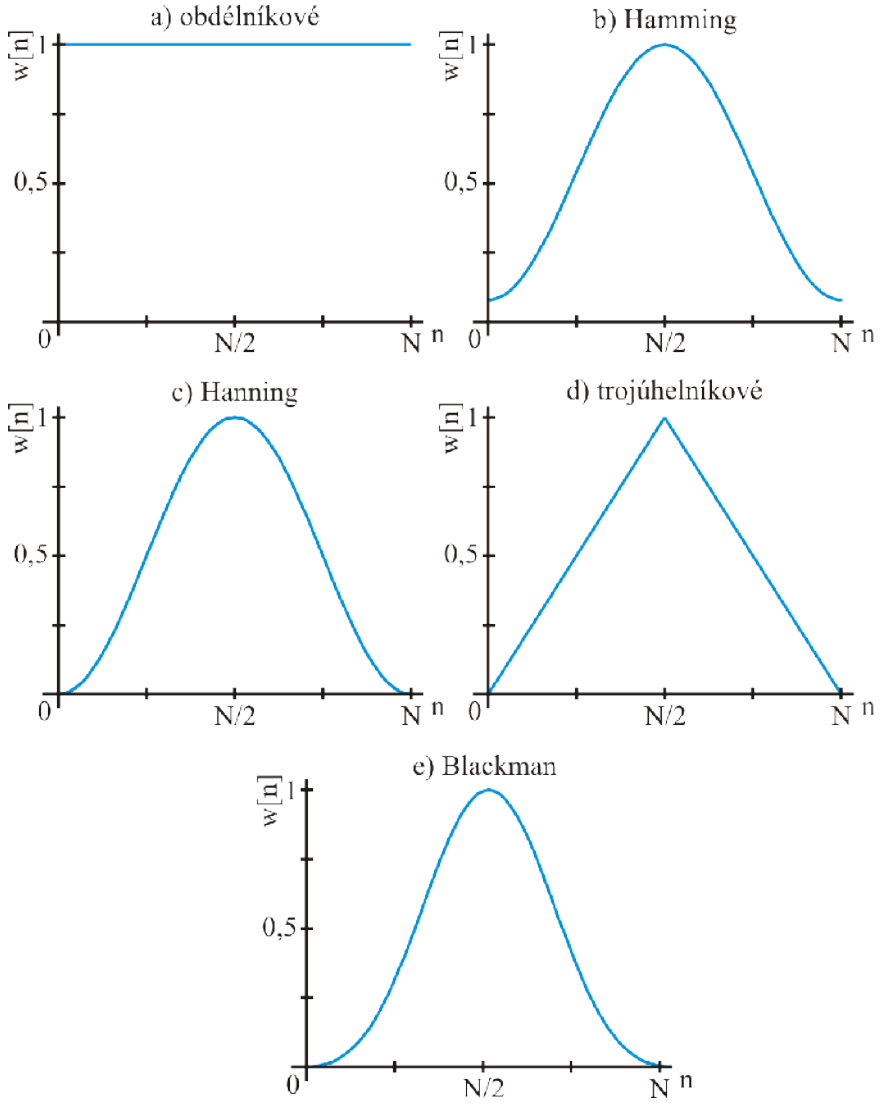
Dalším oknem je trojúhelníkové (Barletovo) okno (Obr. 10d) s předpisem

$$\begin{aligned} w[n] &= \frac{2n}{N}, \quad n=0,1,\ldots,\frac{N}{2}-1 \\ w[n] &= 2-\frac{2n}{N}, \quad n=\frac{N}{2},\ldots,N-1. \end{aligned} \tag{32}$$

Posledním zde zmíněným oknem je Blackmanovo okno (Obr. 10e) jehož tvar je

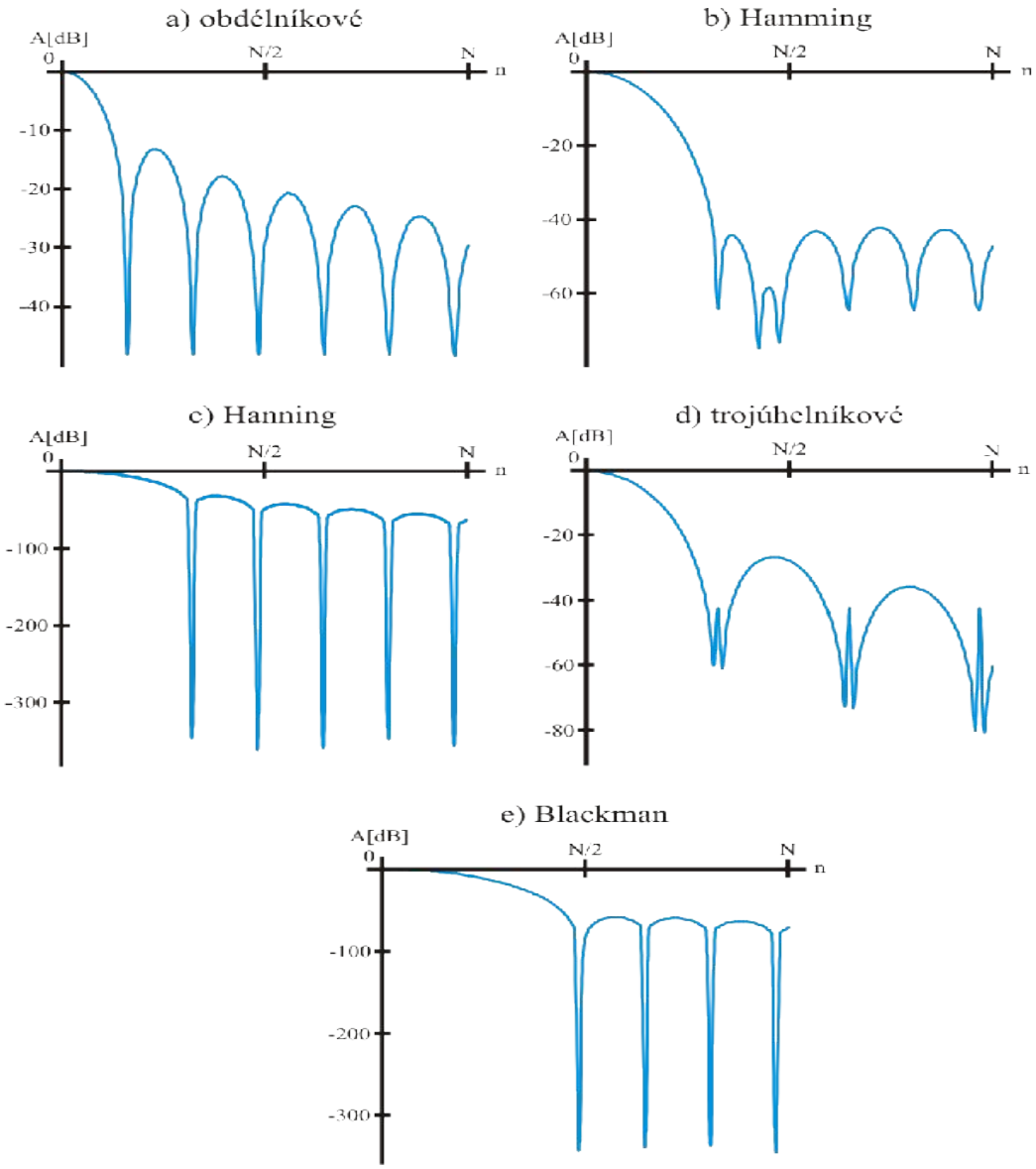
$$w[n] = 0{,}42-0{,}5\cdot\cos\left(\frac{2\pi}{N}\cdot n\right)+0{,}08\cdot\cos\left(\frac{2\pi}{N}\cdot 2\pi\right), \quad n=0,1,\ldots,N-1\,. \tag{33}$$

Jak již bylo zmíněno, existuje několik desítek dalších typů oken, namátkově jen jmenně se jedná o Kaiserovo, Rieszovo, Blackman-Harrisovo, Tukeyovo či sinové okno.



Obr. 10. Průběhy váhovacích oken

Na obrázku (Obr. 10) jsou zobrazena všechna výše zmíněná základní okna. Jejich spektra jsou zobrazena na obrázku (Obr. 11). Vlastnosti těchto základních oken jsou v tabulce (Tab. 1). V ní je *šířka hlavního laloku* a *ekvivalentní šumová šířka* je vztažena k $\Delta f = f_s / N$, kde f_s je vzorkovací frekvence a N je počet vzorků okna. Význam jednotlivých parametrů je objasněn v kapitole 2.5.2.

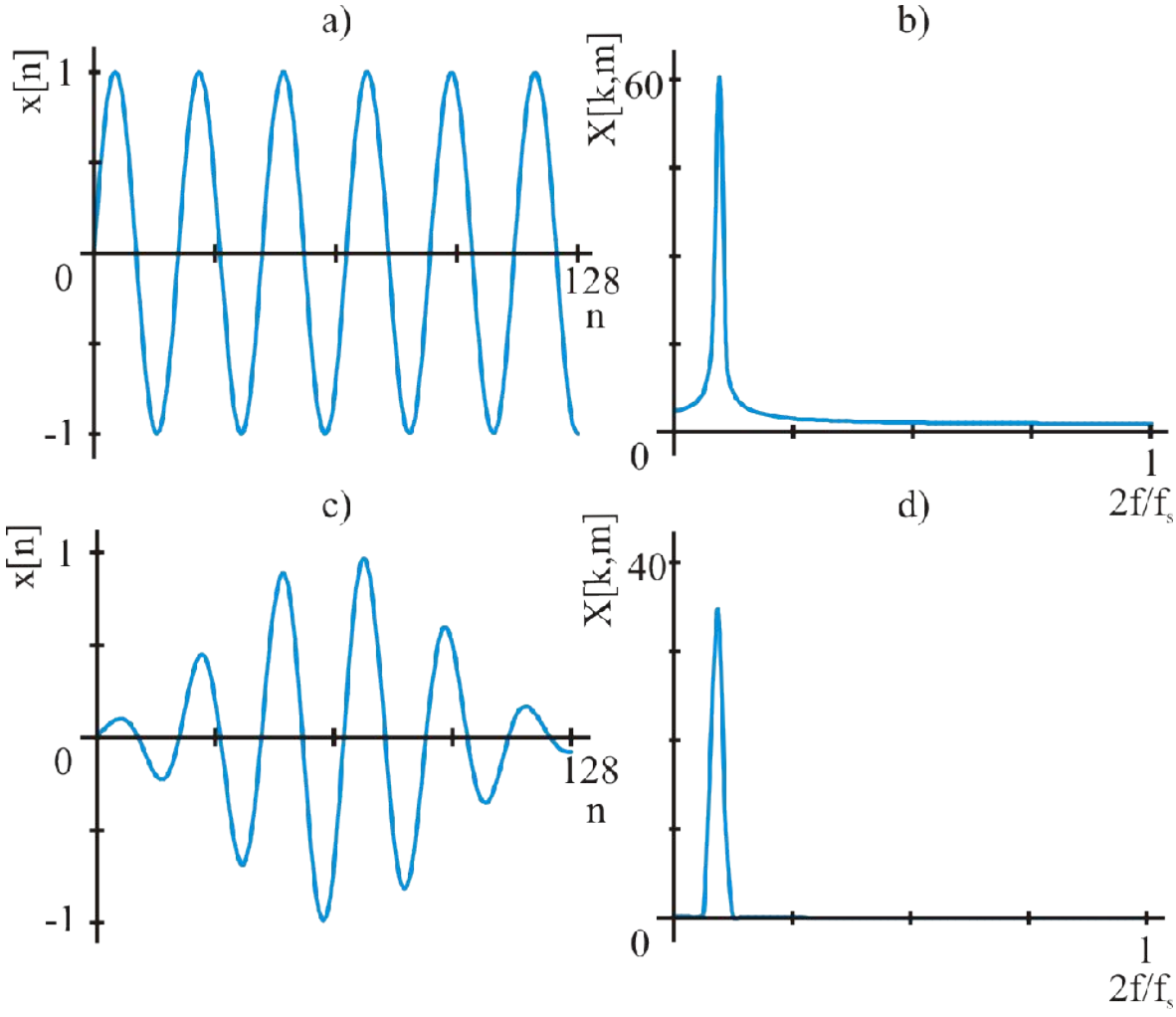


Obr. 11. Spektra váhovacích oken

Tab. 1. Vlastnosti základních typů oken

Okno	***Šířka hlavního laloku***	***Nejvyšší postranní lalok [dB]***	***Ekvivalentní šumová šířka***
obdélníkové	$1 \cdot \Delta f$	-13	1·Δf
Hammingovo	$2 \cdot \Delta f$	-41	1,5·Δf
Hanningovo	$2 \cdot \Delta f$	-31	1,36·Δf
trojúhelníkové	$2 \cdot \Delta f$	-25	1,33·Δf
Blackmanovo	$3{,}5 \cdot \Delta f$	-57	1,73·Δf

2.5.2 Vliv okna na frekvenční spektrum signálu



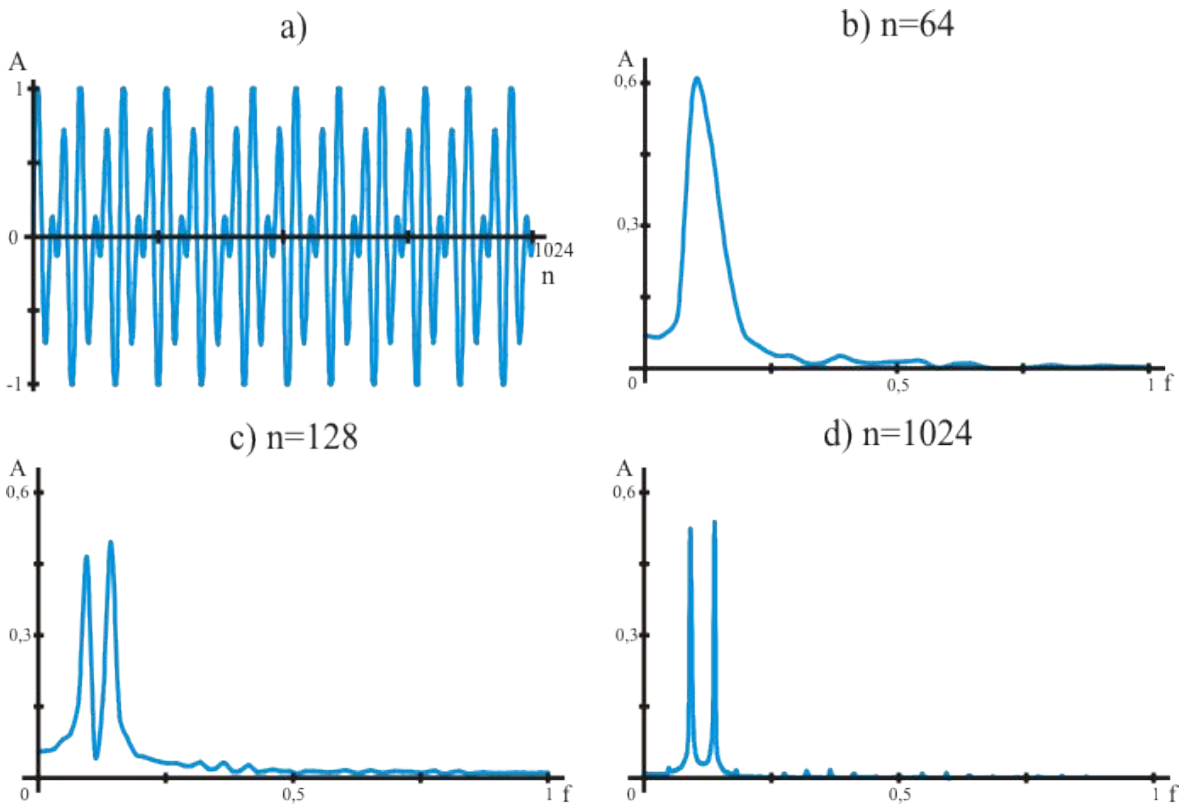
Obr. 12. Vliv okna na spektrum

Na obrázku (Obr. 12a) můžeme vidět signál (jedná se o sinový signál s frekvencí 2000 Hz a amplitudou 1) vynásobený obdélníkovým oknem (tj. vlasně bez úpravy), na obrázku (Obr. 12c) pak tentýž signál upravený vynásobením Hammingovým oknem. Na obrázcích

(Obr 12b) a (Obr 12d) jsou pak jejich spektra.

Jde vidět že obrázky (Obr. 12b) a (Obr. 12d) se liší ve dvou bodech. První z nich jsou nežádoucí frekvenční složky u spektra násobeného obdélníkovým oknem. Tento jev se nazývá *prosakováním spektra* (*spectral leakage*). To je způsobeno nechtěnými postranními laloky ve spektrech oken. Tento jev působí největší potíže při měření slabých signálů v blízkosti silných. Pro porovnání oken z tohoto hlediska slouží *velikost největšího laloku* (v tabulce (Tab. 1) je to hodnota *Nejvyšší postranní lalok*). Platí, že čím menší postranní lalok tím dochází k nižšímu prosakování spektra. Pak je zřejmé proč u signálu upraveného obdélníkovým oknem je v okolí hlavní frekvence „vzedmutí“ prosakujicích frekvencí. Naproti tomu Hammingovo okno má první postranní lalok velice nízký a tudíž k prosakování do jiných frekvencí nedochází, nebo dochází jen v minimální míře.

Druhý bod kde se průběhy liší je šířka hlavní frekvence. I když z tohoto obrázku to není příliš patrné tak frekvenční charakteristika upravená Hammingovým oknem má mnohem širší základnu než průběh upravený obdélníkovým oknem. Tím pádem dochází ke zmenšení rozlišovací schopnosti spektrární analýzy. Tato vlastnost vyplývá z šířky hlavního laloku spektra okna. Čím širší, tím větší ztráta rozlišovací schopnosti.



Obr. 13. Spektra při použití okna různé délky

Posledním parametrem vlastností oken, který uvádí tabulka (Tab. 1) je *ekvivalentní šumová šířka*. Každé spektrum okna obsahuje i svůj vlastní širokopásmový šum. Tento šum se pak díky úpravě oknem dostává i do analyzovaného signálu. Tento šum je samozřejmě nechtěnný a snažíme se jej minimalizovat. Platí, že čím nižší *ekvivalentní šumová sířka*, tím nižší šum.

Dalším důležitým aspektem při použití okna je jeho délka ve vzorcích *N*. Je zřejmé, že čím kratší okno, tím získáme lepší informace o časovém umístění signálu. Pokud však použijeme delší okno, získáme tím lepší popis frekvencí v signálu (ostré špičky). Vše je vidět na obrázku (Obr. 13). Na (Obr. 13a) je vidět prvních 1024 vzorků analyzovaného signálu. Tento signál byl upraven obdélníkovým oknem (tj. pouze oříznut na požadovaný počet vzorků *N*). Můžeme pozorovat jak se zvětšujicím se oknem zvyšuje přesnost analýzy a schopnost rozpoznat frekvenční špičky. Co se týče časového umístění spekter, pak v případě vzorkovací frekvence 44100 Hz (která je používána při záznamu hudby na CD nosič) je rozlišovací schopnost v případě *N*=64 vzorků 1,45 ms. U 128 vzorků je to přibližně dvounásobek, tj. 2,90 ms. A konečně 1024 vzorků poskytuje rozlišení 23,2 ms.

2.5.3 Překryv okna

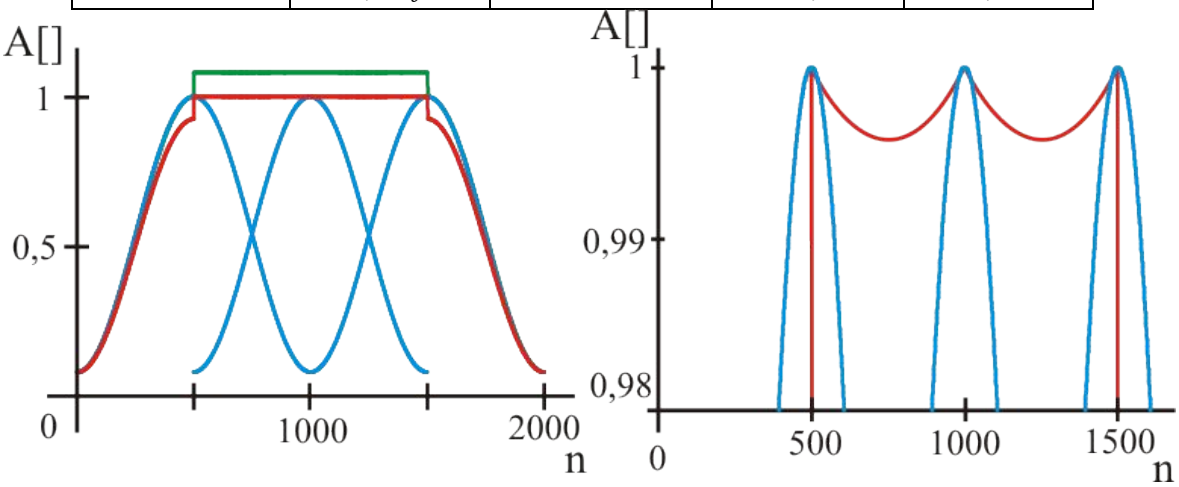
Poté co je vybráno okno a počet vzorků na okno *N*, musíme ještě stanovit poslední důleřitý parametr, kterým je překryv okna. Teoretický překryv se spočítá podle následujicího vztahu

$$\frac{M-m}{M}\cdot 100 = \left(1-\frac{1}{2\cdot k_c}\right)\cdot 100 \quad [\%] \tag{34}$$

Kde *m* je krok segmentace ,*(M - m) / M* je vlastní překryv a k_c reprezentuje šířku hlavního laloku. V tabulce (Tab. 2) jsou shrnuty vypočítané teoretické překryvy a reálná zvlnění obálky jejíž určení je patrno z obrázku (Obr. 14). V tabulce si můžeme všimnout, že v případě Hammingova a Hanningova okna a překrytí 50% je zvlnění menší než 1%, což je také důvodem, proč se praxi používá u těchto oken právě tento překryv, který navíc proti teoretickému 75% znamená dvounásobné snížení výpočetních nároků.

Tab. 2. Překryvy a zvlnění oken ve složeném signálu

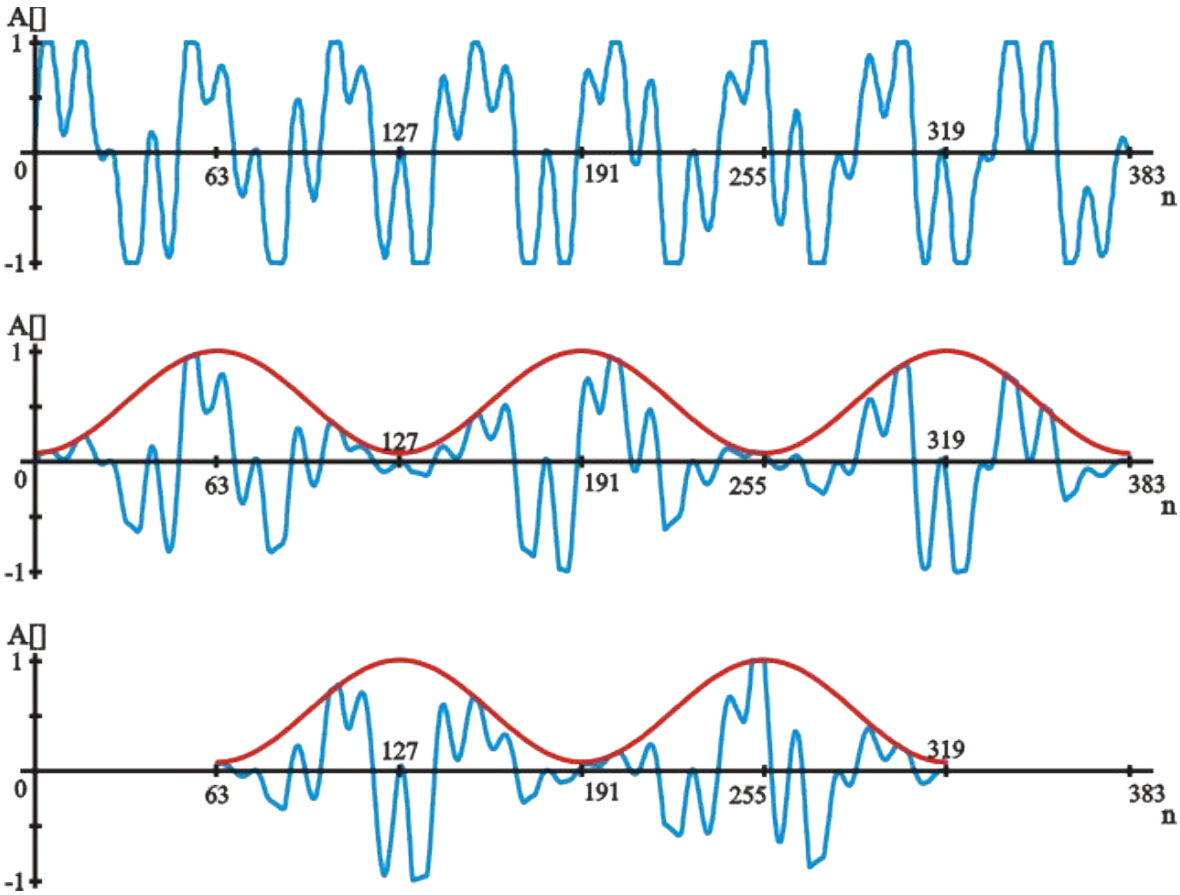
Okno	***Šířka laloku $k_c\cdot\Delta f$***	***Teoretický překryv [%]***	***Zvlnění pro 50 % [%]***	***Zvlnění pro 75 % [%]***
obdélníkové	$1\cdot\Delta f$	50	0	0
Hammingovo	$2\cdot\Delta f$	75	0,67	0,14
Hanningovo	$2\cdot\Delta f$	75	0,76	0,15
trojúhelníkové	$2\cdot\Delta f$	75	0	0
Blackmanovo	$3,5\cdot\Delta f$	85	32,0	0,04



Obr. 14. Obálka výstupního signálu při použití Hammingova okna

Na obrázku (Obr. 15) je zobrazen příklad rozkladu signálu (modrá čára) a váhování pomocí Hammingova okna (červená čára) s 50% překryvem, což je v praxi nejčastější použití. Jde vidět, že okno signál na krajích tlumí, ale při překryvu jsou tlumené signály zpracovány

vícekrát (v tomto případě dvakrát) a neztrácí se tak jejich informační hodnota.



Obr. 15. Signál a jeho rozklad na segmenty

II. PRAKTICKÁ ČÁST

3 TECHNOLOGICKÁ KARTA ADVANTECH

Cílem této bakalářské práce je vytvoření programového vybavení pro analýzu zvukového signálu. Tento signál je získáván jako vstup technologické karty Advantech PCI-1716.

3.1 Popis karty

Karta Advantech PCI-1716 je 16ti bitová karta určená k připojení přes PCI sběrnici osobního počítače. Plně podporuje technologii Plug-and-Play. Karta nabízí 16 jednopólových nebo 8 dvoupólových analogově digitálních (A/D) vstupů. Na kartě je umístěn jak 16ti bitový A/D převodník se vzorkovací frekvencí až 250 kHz, tak i dvoukanálový D/A převodník. Nabízí programovatelnou velikost zesílení pro každý vstupní kanál. Na kartě je umístěna FIFO paměť o velikosti 1 KB. Uživateli karty je k dispozici 16 digitálních vstupů a stejný počet digitálních výstupů, hardwarový časovač ve formě čipu 82C54, zapouzdřující tři 16ti bitové časovače, každý pracující s frekvencí 10 MHz.

3.2 Popis funkcí a struktur používaných programem

V případě programu *f_*analyzer, který provádí realtime zpracování dat je důležitou, ne-li tou nejdůležitější, vlastností rychlost. Driver karty Advantech naštěstí nabízí sadu vysokorychlostních funkcí využívající DMA přenosy pro získávání dat. Mimo tyto vysokorychlostní funkce je použita řada funkcí sloužících hlavně k nastavení karty a jejích parametrů.

3.2.1 Funkce DRV_DeviceOpen

Používá se k inicializaci karty a alokování potřebné paměti, musí být použita jako první před jakoukoliv jinou funkcí pro kartu Advantech. Její deklarace vypadá následovně:

```
1 LRESULT DRV_DeviceOpen(ULONG DeviceNum, LONG *DriverHandle);
```

Parametr `DeviceNum` je označením zařízení, které vrací funkce `DRV_SelectDevice()`. `DriverHandle` je popisovačem ovladače a vrací se v něm konfigurační data zařízení. Tento paramer se poté opakuje u všech dalších funkcí. Funkce vrací konstantu `SUCCESS` při úspěchu, při neúspěchu je to buď `MemoryAllocateFailed` (při chybě alokování paměti), `ConfigDataLost` (když selže vrácení konfigurace karty) nebo `CreateFileFailed` (pokud nízkoúrovňový ovladač vykáže I/O chybu).

3.2.2 Funkce DRV_DeviceClose

Tato slouží resetu karty do původního nastavení a uvolnění paměti alokované funkcí `DRV_DeviceOpen()`.

```
1 LRESULT DRV_DeviceClose(LONG *DriverHandle);
```

kde `DriverHandle` je ukazatel na popisovač ovladače. Pokud funkce uspěje vrací `SUCCESS`, jinak `InvalidDriverHandle`, pokud `DriverHandle` je roven hodnotě `NULL`, tzn. neexistuje.

3.2.3 Funkce DRV_SelectDevice

Slouží k určení používaného zařízení (karty) z list-boxu, který tato funkce naplní. Je deklarována jako

```
1 LRESULT DRV_SelectDevice(HWND hCaller, BOOL GetModule, ULONG
  *DeviceNum, UCHAR *Descriptions);
```

Kde parametr `hCaller` je handle na okno či list-box volající funkci, `GetModule` určuje funkci zda zvolit modul z registrů nebo ne. V parametru `DeviceNum` je vráceno číslo zařízení a v `Descriptions` je vrácen popis zařízení ve formě ASCIIZ řetězce. Funkce vrací `0` pokud uspěla a `1` pokud selhala.

3.2.4 Funkce DRV_GetFeatures

Vrací vlastnosti specifické pro používané zařízení.

```
1 LRESULT DRV_GetFeatures(LONG DriverHandle, LPT_DeviceGetFeatures
  lpDevFeatures);
```

Kde `lpDevFeatures` je ukazatel na strukturu ve které jsou vráceny informace o kartě. Návratové hodnoty jsou buď `SUCCESS` při úspěchu, `InvalidDriverHandle` při špatném popisovači a nebo `BoardIDNotSupported` když ID desky není podporováno.

3.2.5 Funkce DRV_EnableEvent

Povoluje či zakazuje události karty.

```
1 LRESULT DRV_EnableEvent(LONG DriverHandle, LPT_EnableEvent
  lpEnableEvent);
```

Kde `LPT_EnableEvent` je ukazatel na strukturu obsahující potřebné údaje. Funkce vrací `SUCCESS` při úspěchu, `InvalidDriverHandle`, `InvalidInputParam` při chybných vstupních parametrech a `InvalidEventType` při špatném určení události.

3.2.6 Funkce DRV_CheckEvent

Vyzvedne událost, je-li k dispozici a přečte současný stav.

```
1 LRESULT DRV_CheckEvent(LONG DriverHandle, LPT_ChectEvent
  lpCheckEvent);
```

Kde `LPT_CheckEvent` je ukazatel na strukturu obsahující potřebné údaje. Návratové hodnoty jsou známé `SUCCESS` a `InvalidDriverHandle` a navíc ještě `EventTimeOut` při vypršení časového intervalu uvedeného ve struktuře `LPT_CheckEvent`.

3.2.7 Funkce DRV_FAIDmaExStart

Jedná se o první z vysokorychlostních funkcí nabízených ovladačem karty. Tato funkce inicializuje a zahajuje přenos dat z paměti na kartě do paměti alokované ovladačem karty v RAM paměti počítače.

```
1 LRESULT DRV_FAIDmaExStart(LONG DriverHandle, LPT_FAIDmaExStart
  lpFAIDmaExStart);
```

Kde `lpFAIDmaExStart` je ukazatelem na strukturu obsahující potřebná nastavení. Funkce vrací v případě úspěchu `SUCCESS`, dalších 12 návratových kódů viz. [10].

3.2.8 Funkce DRV_FAIStop

Tato funkce slouží k zastavení datový přenosů. Také prování reset hardwaru i softwarového ovladače.

```
1 LRESULT DRV_FAIStop(LONG DriverHandle);
```

Ve funkci je `DriverHandle` nám již známým popisovačem ovladače. Funkce vrací buď `SUCCESS` při úspěchu nebo `InvalidDriverHandle` při neúspěchu.

3.2.9 Funkce DRV_FAICheck

Funkce vrací informaci o tom, zda je již aktuální přenos dat kompletní či nikoliv.

```
1 LRESULT DRV_FAICheck(LONG DriverHandle, LPT_FAICheck lpFAICheck);
```

Parametr `lpFAICheck` je ukazatelem na strukturu s daty. Návratové hodnoty funkce jsou opět `SUCCESS` při úspěchu a `InvalidDriverHandle` při neúspěchu.

3.2.10 Funkce DRV_FAITransfer

Tato funkce provádí přesun dat z bufferu ovladače do uživatelského bufferu.

```
1 LRESULT DRV_FAITransfer(LONG DriverHandle, LPT_FAITransfer
  lpFAITransfer);
```

Druhý parametr `lpFAITransfer` je ukazatel na strukturu dat, jež funkce potřebuje. Návratové hodnoty funkce mohou nabývat hodnot `SUCCESS` při úspěchu, `InvalidDriverHandle` při neexistujícím popisovači a `InvalidCountNumber` při špatném určení zdrojového bufferu.

3.2.11 Struktura PT_DeviceGetFeatures

Tato struktura je deklarována takto:

```
1 typedef struct PT_DeviceGetFeatures {
2     LPDEVFEATURES buffer;
3     USHORT size;
4 } PT_DeviceGetFeatures, *LPT_DeviceGetFeatures;
```

Kde proměnná `buffer` je strukturou obsahující např. maximální počet analogových a digitálních vstupů a výstupů, určení počtu bitů pro A/D převodník a jiné (podrobně viz. [10], položka `DEVFEATURES`) a proměnná `size` je velikost bufferu.

3.2.12 Struktura PT_EnableEvent

Deklarace vypadá následovně:

```
1 typedef struct PT_EnableEvent {
2     USHORT EventType;
3     USHORT Enabled;
4     USHORT Count;
5 } PT_EnableEvent, *LPT_EnableEvent;
```

Proměnná `EventType` určuje na kterou z událostí se má reagovat (možnosti viz. [10]). `Enabled` určuje zda povolujeme či zakazujeme danou událost a `Count` je počet přerušení, které událost bude generovat.

3.2.13 Struktura PT_CheckEvent

Struktura je deklarována:

```
1 typedef struct PT_CheckEvent {
2     USHORT far *EventType;
3     DWORD Miliseconds;
```

```
4 } PT_CheckEvent, *LPT_CheckEvent;
```

`EventType` je ukazatelem na typ události a `Miliseconds` je doba čekání na událost v milisekundách.

3.2.14 Struktura PT_FAIDmaExStart

```
1 typedef struct PT_FAIDmaExStart {
2     USHORT TrigSrc;
3     USHORT TrigMode;
4     USHORT ClockSrc;
5     USHORT TrigEdge;
6     USHORT SRCType;
7     FLOAT TrigVol;
8     USHORT CyclicMode;
9     USHORT NumChans;
10    USHORT StartChan;
11    ULONG ulDelayCnt;
12    ULONG count;
13    ULONG SampleRate;
14    USHORT *GainList;
15    USHORT *CondList;
16    TRIGLEVEL *LevelList;
17    USHORT *buffer0;
18    USHORT *buffer1;
19    USHORT *Pt1;
20    USHORT *Pt2;
21    USHORT *Pt3;
22} PT_FAIDmaExStart, *LPT_FAIDmaExStart;
```

Jak je vidět jedná se o rozsáhlou strukturu obsahující 20 proměnných. Pomocí této struktury nastavujeme takové údaje jako např. vzorkovací frekvenci (proměnná `SampleRate`), počet kanálů (`NumChans`) či buffer (`buffer0`) do kterého se budou zapisovat data převzatá z karty. Pro bližší popis viz. [10].

3.2.15 Struktura PT_FAICheck

Její deklarace vypadá následovně:

```
1 typedef struct PT_FAICheck {
2     USHORT far *ActiveBuf;
3     USHORT far *stopped;
4     USHORT far *retrieved;
```

```
5     USHORT far *overrun;
6     USHORT far *HalfReady;
7 } PT_FAICheck, *LPT_FAICheck;
```

Nejdůležitější proměnnou této struktury je ukazatel `HalfReady`, ve kterém funkce `DRV_FAICheck` vrací která část bufferu je naplněna, jestli první, druhá nabo žádná. V našem případě má driver k dispozici pouze jeden buffer rozdělený na dvě poloviny, které neustále dokola plní a jejich připravenost dává najevo nastavením výše zmíněné proměnné.

3.2.16 Struktura PT_FAITransfer

Deklarace je následujicí:

```
1 typedef struct PT_FAITransfer {
2     USHORT ActiveBuf;
3     LPVOID DataBuffer;
4     USHORT DataType;
5     ULONG start;
6     ULONG count;
7     USHORT far *overrun;
8 } PT_FAITransfer, *LPT_FAITransfer;
```

Proměnná `ActiveBuffer` určuje zdrojový buffer, v našem případě se jedná o stále tentýž. Ukazatel na typ `void DataBuffer` je cílový buffer. Proměnnou `DataType` určujeme zda chceme data dostat jako `unsigned short` či `float`. Parametrem `start` určujeme počáteční místo ve zdrojovém bufferu (jedná se buď o nulu či polovinu délky bufferu) a parametrem `count` délku přenášených dat (vždy je rovná polovině délky bufferu). Do proměnné `overrun` funkce `PT_FAITransfer()` ukládá zda došlo k přetečení či nikoli.

4 KNIHOVNA FFTW

Pro potřeby programu bylo třeba použít existující či vytvořit novou knihovnu počítající rychlou Fourierovu transformaci. Algoritmus rychlé Fourierovy transformace s decimací v čase jak byl popsán v kapitole 2.4.1 je poměrně jednoduchý program na přibližně 100 řádků zdrojového textu. I přes to byla dána přednost použití již existujicí knihovny FFTW (Fastest Fourier Transfort in the West) jejímiž autory jsou pánové Matteo Frigo a Steven G. Johnson. Důvodem je hlavně stabilita, rychlost, množství dokumentace a zdrojových kódů a v neposlední řadě také licence, jelikož knihovna FFTW je distribuována pod GPL licencí.

4.1 Popis vlastností knihovny FFTW

Knihovna FFTW je tedy knihovna pro výpočty dobředné i zpětné rychlé Fourierovy transformace. Je psána v jazyku C pro operační systém Linux, ale existují i porty pro vetšinu ostatních operačních systémů, pro MS Windows, což je i náš případ, nevyjímaje. Knihovna může mít jako vstup posloupnost reálných i imaginárních dat libovolné délky (omezující je pouze hardware počítače). Nabízí možnost transformace *n*-rozměrných dat. V našem případě, kdy je předmětem analýzy zvukový signál, si vystačíme pouze s jedním rozměrem. FFTW podporuje rozšíření instrukčních souborů, jako jsou *SSE*, *SSE2* u procesorů Intel, *3DNow!* u procesorů AMD a u procesorů PowerPC je to *AltiVec*. Knihovna bez problémů umožňuje paralerní zpracování dat ve více vláknech.

4.2 Popis funkcí a typů knihovny FFTW použitých v programu

4.2.1 Funkce fftw_malloc

```
1 void * fftw_malloc(size_t n);
```

Tato funkce je obdobou klasické funkce jazyka ISO C (dříve označovaném jako ANSI C) `malloc()`. Rozdílem proti ní je ten, že tato funkce podporuje zarovnávání, aby bylo možno využít SIMD instrukcí moderních procesorů. Pamět přidělená touto funkcí by měla být pokud možno zrušena funkcí `fftw_free()`, použití klasického `free()` nemusí být korektní, použití operátoru `delete` je v tomto případě naprosto zakázané.

4.2.2 Funkce fftw_free

```
1 void fftw_free(void *p);
```

Tato funkce slouží k uvolnění paměti alokované pomocí `fftw_malloc()`.

4.2.3 Funkce fftw_plan_dft_1d

```
1 fftw_plan fftw_plan_dft_r2c_1d(int n, double *in, fftw_complex
  *out, unsigned int flags);
```

Tato funkce inicializacializuje proměnnou `fftw_plan`, která posléze slouží k samotnému výpočtu. Provádí přípravu pro jednodimenzionání dopřednou transformaci reálných vstupních dat s komplexním výstupem. Parametr funkce `n` je počet vzorků, jež se budou analyzovat. Proměnná `in` typu `double` je vstupní posloupnost reálných hodnot. Výstupní posloupnost je typu `fftw_complex` a jmenuje se `out`. Parametrem `flags` předáváme funkci parametry analýzy. Parametry mohou nabývat hodnot `FFTW_ESTIMATE` (pro co nejrychlejší výpočet výstupu, kdy se algoritmus výpočtu určí pomocí jednoduché heuristiky, jež však bohužel nemusí být optimální), `FFTW_MEASURE` (v tomto případě funkce několikrát přepočítá transformaci pro různé algoritmy a výsledný je ten s nekratším časem), `FFTW_PATIENT` (počítá ještě více algoritmů, má největší šanci ke zjištění toho opravdu nejúčinějšího) a `FFTW_EXHAUSTIVE` (který proti předchozímu navíc propočítává i algoritmy, které nebývají rychlé). Kromě této funkce, která jako vstup přebírá reálná data existují i funkce pro imaginární vstupní data, dále pro dvou a vícerozměrové transformace. Pro jejich popis viz. [4].

4.2.4 Funkce fftw_execute

```
1 void fftw_execute(const fftw_plan p);
```

Tato funkce provádí samotný výpočet způsobem, který určila funkce předchozí.

4.2.5 Funkce fftw_destroy_plan

```
1 void fftw_destroy_plan(const fftw_plan p);
```

Uvolňuje proměnnou `p` typu `fftw_plan` po výpočtu.

4.2.6 Typ fftw_complex

Tento typ je definovaný jako:

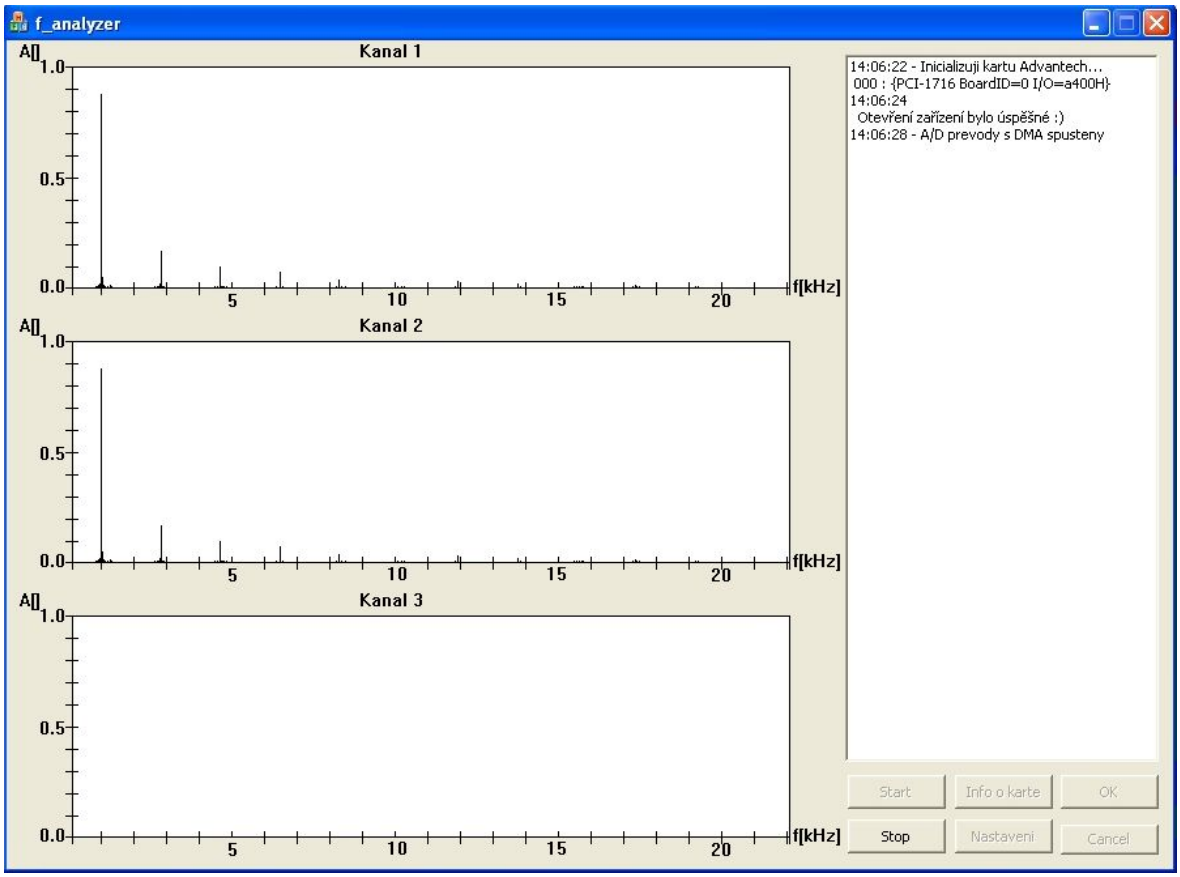
```
1 typedef double fftw_complex[2];
```

Jde tedy vidět, že se jedná o pole dvou reálných proměnných. Element `[0]` je reálnou částí a element `[1]` je imaginární částí čísla.

5 PROGRAM F_ANALYZER

Výsledným produktem této práce je program f_analyzer sloužící ke spektrární analýze signálu. Program je napsán v programovacích jazycích C a C++ v integrovaném vývojovém prostředí Microsoft Visual Studio 2005. K jeho realizaci byla využita knihovna MFC.

5.1 Hlavní dialogové okno aplikace



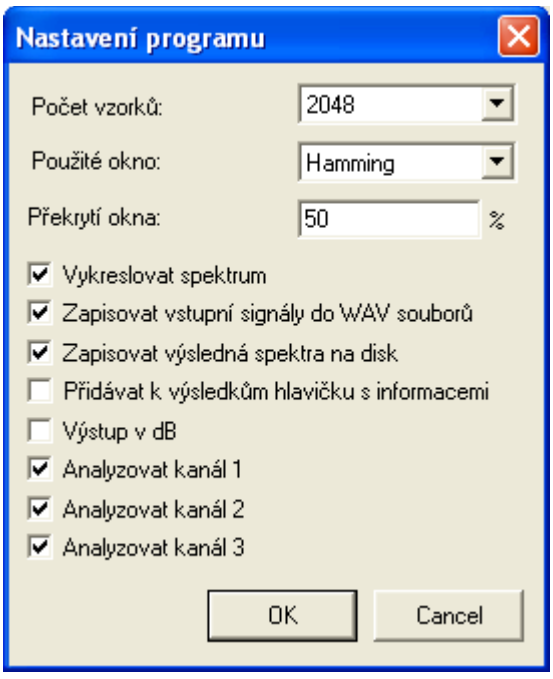
Obr. 16. Hlavní dialogové okno aplikace f_analyzer

Na hlavním dialogovém okně aplikace jsou umístěny všechny potřebné prvky programu. Jedná se především o tři souřadné systémy pro vykreslování grafů. Dále prvek list-box, ve kterém jsou zobrazovány informace o průběhu analýzy a nastavení technologické karty. Dále je na dialogu šestice tlačítek jejichž význam je jasný z jejich názvu. Screenshot hlavního dialogu aplikace je na obrázku (Obr. 16). Obrázek byl pořízen v okamžiku probíhajicí analýzy dvou obdélníkových signálů o frekvenci 1000 Hz.

5.2 Dialogové okno Nastavení programu

Druhým důležitým dialogem obsaženým v aplikace je dialog *Nastavení programu*. Jeho screenshot je na obrázku (Obr. 17). Tímto dialogem se nastavují parametry programu.

Jedná se o nastavení počtu vzorků k analýze (možnosti jsou: 256, 512, 1024, 2048, 4096 a 8192 vzorků). Program nabízí tato okna: obdélníkové, trojúhelníkové, Hammingovo, Hanningovo a Blackmanovo. Parametr *Překrytí okna* je vyjádřen v procentech počtu vzorků vstupujicích do analýzy a může nabývat hodnot od 0 do 99 procent. Přepínač *Vykreslovat spektrum* umožňuje povolit či zakázat vykreslování spekra (spekter) v průběhu analýzy. Parametr *Zapisovat vstupní signály do WAV souborů* ovlivňuje to, zda signál získaný z technologické karty bude zapsán na disk jako audio soubor. Zda budou výsledky analýzy zapisovány na disk můžeme rozhodnout přepínačem *Zapisovat výsledná spektra na disk*. Program má možnost v případě zápisu výsledků na disk přidávat do každého souboru hlavičku s informacemi o analýze, to je dosaženo přepínačem *Přídávat k výsledkům hlavičku s informacemi*. Tyto doplňující data jsou: datum, čas, rozsah analyzovaných vzorků, typ, velikost a překrytí okna. Přepínač *Výstup v dB* umožní přepočítat výsledky do decibellů. Pomocí posledních tří přepínačů *Analyzovat kanál 1, 2, 3* je možno nastavit, které vstupní kanály budou podrobeny analýze.



Obr. 17. Dialogové okno Nastavení programu

6 PRAKTICKÉ OVĚŘENÍ FUNKCE PROGRAMU

Pro praktické ověření funkčnosti programu jsem v aplikaci GoldWave v5.12 vytvořil sadu testovacích signálů. Jednalo se o signály sinus a obdélník o frekvencích 100, 1000 a 5000 Hz. Všechny signály byly vzorkovány s frekvencí 44100 Hz. Tyto signály byly poté přehrány přes integrovanou zvukovou kartu zkušebního počítače. Z výstupu zvukové karty byly kabely poté přivedeny na svorkovnici ADAM 3968 a z ní prostřednictvím stíněného 68 žilového SCSI kabelu na technologickou kartu. Na svorkovnici byly přivedeny na svorku 68 (první vstup), 34 (druhý vstup) a 67 (třetí vstup). Společná signálová zem byla připojena na svorku 60. Je nutné poznamenat, že měření bylo ovlivněno nižší kvalitou integrované zvukové karty, jež nebyla schopna dostatečně kvalitně vytvořit výstupní signály. Toto jde vidět hlavně u obdélníkového průběhu, jež není kvalitní ani při vzorkovací frekvenci 100 Hz, natož při 5000 Hz.

V příloze (Příloha P 1) jsou uvedeny signály jak byly zaznamenány programem f_analyzer. Jejich spektra jsou uvedena v přílohách (Příloha P 2, P 3). V první z nich je při úpravě signálu použito Hammingova okna s překrytím 50 % a 8192 vzorků na výpočetní cykl. V příloze (Příloha P 3) jsou pak spektra těchto signálů upravených obdélníkovým oknem, překrytí 0 % a opět s 8192 vzorky. Následují (Příloha P 4) ideální signály, které byly vytvořeny ve výše zmíněném programu GoldWave. V dalších přílohách (Příloha P 5, P 6) jsou pak spektra přímo těchto ideálních signálů, tedy takových signálů, které neprošly zvukovou a technologickou kartou počítače.

Ideální a reálné signály se od sebe liší hlavně v hodnotách amplitud. To je dáno ztrátami v průběhu přenosu signálu. Tento stav byl ovlivněn například nastavením hlasitosti v dialogu *Ovládání hlasitosti* operačního systému Windows XP, a dále nastavením zisku na technologické kartě.

ZÁVĚR

Cílem práce bylo vytvoření programového vybavení pro frekvenční analýzu zvukového signálu. Tento hlavní cíl se podařilo beze zbytku splnit. Výsledkem je aplikace *f_analyzer*, program provádějící amplitudovou spektrální analýzu signálu přivedeného jako vstup na technologickou kartu Advantech.

V teoretické části práce je rozebrána základní teorie signálů a jejich zpracování pomocí různých druhů Fourierovy transformace. Důležitým teoretickým tématem je užití okenních funkcí a tím pádem přechod do časově-frekvenční analýzy. Nyní již není problém určit nejen jaké frekvence se v signálu vyskytují, ale také, a to je důležité, v jakém okamžiku. Díky tomu by program mohl být odrazovým můstkem kupříkladu do světa zpracování mluveného slova či hudby. Teoretická část tedy přináší vysvětlení témat důležitých pro další úspěšné řešení problému.

Praktická část se jako prvním tématem zabývá technologickou kartou Advantech PCI-1716, jejím popisem a funkčním rozhraním pro komunikaci program - karta.

Druhou probíranou látkou je knihovna FFTW. Jedná se o zdarma šířenou sadu funkcí sloužících k výpočtu rychlé Fourierovy transformace. Pro potřeby programu se výborně hodí a tudíš bylo rozhodnuto jejích možností využít. I přes to, že autoři FFTW připouští možné neshody mezi knihovnou, původně psanou pro Linux, a operačním systémem MS Windows XP či překladačem jazyka C/C++ MS Visual Studio, nebyly tyto ani v jednom případě pozorovány. Běh knihovny je hladký, spolehlivý a neobyčejně rychlý.

V praktické části dále následuje popis vytvořené aplikace. Předmětem popisu je její uživatelské rozhraní, které bylo navrženo hlavně pro funkčnost. Program je možno v široké míře ovlivňovat nastavením všech potřebných parametrů, jejich charakteristika je opět uvedena. Zdrojové kódy k programu i samotná aplikace jsou přiloženy na disku CD.

Posledním tématem je ověření funkce programu v praxi. Jako zkušební signály byly použity sinus a obdélník o různých frekvencích. V přílohách jsou umístěny grafy těchto signálů, ideálních i těch, jak je nasnímala karta Advantech. Spektrální průběhy jsou přiloženy také. Z nich lze pozorovat, že spektra ideálních i reálných signálů se téměř neliší. Po zevrubnějším zkoumání naměřených dat bychom samozřejmě došli k mnoha rozdílům, ale ty nejsou natolik závažné, abychom mohli označit analýzu za nekvalitní, či dokonce nesprávnou.

SEZNAM POUŽITÉ LITERATURY

[1] CHALUPA, Radek. *1001 tipů a triků pro Visual C++*. 1. vyd.. Náměstí 28. dubna 48, Brno: Computer Press, a. s., 2003. ISBN: 80-7226-842-2

[2] CHAPMAN, Davis. *Teach Yourself Visual C++ 6*. 1. vyd.. 201 West 103rd St., Indianapolis, Indiana, 46290: Sams, A Division of Macmillan Computer Publishing, 1998. ISBN: 0-672-31240-9

[3] ECKEL, Bruce. *Myslíme v jazyku C++*. 1. vyd.. U Průhonu 22, Praha 7: Grada Publishing, spol. s r. o., 2000. ISBN: 80-247-9009-2

[4] FRIGO, Matteo, JOHNSON Steven G., *FFTW manual for version 3.1*. Massachusetts Institute of Technology, 2003.

[5] HEROUT, Pavel. *Učebnice jazyka C*. 3. vyd.. Šumavská 3, České Budějovice: Nakladatelství KOPP, 1999. ISBN: 80-85828-21-9

[6] HLAVÁČ, Václav, SEDLÁČEK, Miloš. *Zpracování signálů a obrazů*. 2. vyd. Thárukova 1, 160 36 Praha 6: Vydavatelství ČVUT, 2002. ISBN 80-01-03110-1

[7] KADLEC, František. *Zpracování akustických signálů*. 1. vyd. Thárukova 1, 160 41 Praha 6: Vydavatelství ČVUT, 2002. ISBN 80-01-02588-8.

[8] SMITH, Steven W. *The Scintist and Engineer's Guide to Digital Signal Processing*. 2. vyd.. P.O. Box 502407, San Diego, CA 92150-2407: California Technical Publishing, 1999. ISBN: 0-9660176-6-8

[9] VIRIUS, Miroslav*Jazyky C a C++*. 1. vyd.. U Průhonu 22, Praha 7: Grada Publishing, spol. s r. o., 2006. ISBN: 80-247-1494-9

[10] *PCI-1710 Series 12/16 bit Multifunction Card User's Manual*. . : Advantech Co., Ltd., 2001. ISBN 20-031-7160-0

[11] CHAPLAIS, F. *A Wavelet Tour of Signal Processing* [online]. [cit. 2006-03-28]. Dostupné z www: <http://cas.ensmp.fr/~chaplais/wavetour_presentation/wavetour_presentation_us.html>

[12] HLAVÁČ, Václav. *Počítačové vidění vs. digitální zpracování obrazu* [online]. [cit. 2006-04-13]. Dostupné z www: <http://cmp.felk.cvut.cz/~hlavac/public/teachinglectures/>

[13] MARŠÁLEK, Leoš, SKAPA, Jan. *Diskrétní transformace* [online]. [cit.

2006-04-03]. Dostupné z www: <http://www.sendme.cz/sklad>

[14] MARŠÁLEK, Tomáš, MERTA, Michal. *Úvod do analýzy signálů* [online]. [cit. 2006-04-03]. Dostupný z WWW: <http://home.zcu.cz/~tmarsal/kro/index.html>

[15] NOUZA, Jan. *Počítačové zpracování signálů* [online]. [cit. 2006-04-03]. Dostupné z www: <http://www.fm.vslib.cz/~kes/pages/nove_predmety/pzs/ramce_main.html>

[16] ČERNOCKÝ, Jan. *Zpracování řeči* [online]. [cit. 2006-06-01]. Dostupné z www: <http://www.fit.vutbr.cz/~cernocky/oldspeech>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

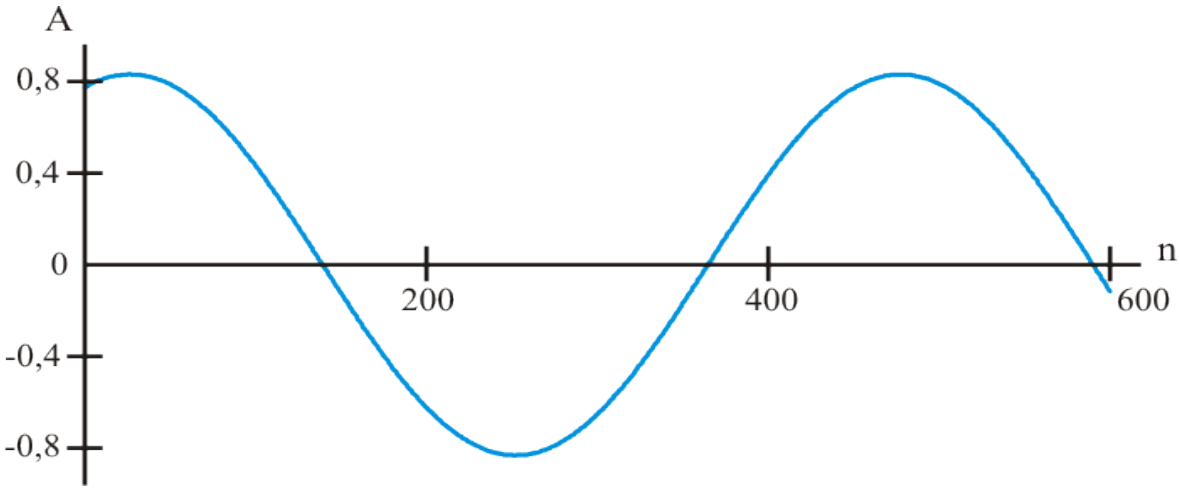
A/D	Analogově-digitální
AMD	Advanced Micro Devices
ANSI	American National Standard Institute
ASCII	American Standard Code for Information Interchange
ASCIIZ	ASCII Zero terminated
D/A	Digitálně-analogový
dB	decibell
DFT	Discrete Fourier Transform - diskrétní Fourierova transformace.
DIT	Decimation-In-Time - decimace v čase
DMA	Direct Memory Acsess
FFT	Fast Fourrier Transform - rychlá Fourierova transformace
FFTW	Fast Fourier Transform in the West
FIFO	First In First Out
GNU	GNU's Not Unix
GPL	GNU Public Licence
Hz	Hertz
I/O	Input - Output
IM	Imaginární část
Intel	INTegrated ELectronics
ISO	International Standard Organization
KB	Kilo Byte
MAG	Magnitude - amplituda.
MFC	Microsoft Foundation Classes
MHz	Mega Hertz
MS	Microsoft
ms	milisekunda
PCI	Peripheral Component Interconnect
PHA	Phase - fázový posun.
RE	Reálná část
SCSI	Small Computer System Interface
SIMD	Single Instruction Multiple Data
SSE	Streaming SIMD Extension
STFT	Short Time Fourier Transform - krátkodobá Fourierova transformace

SEZNAM OBRÁZKŮ

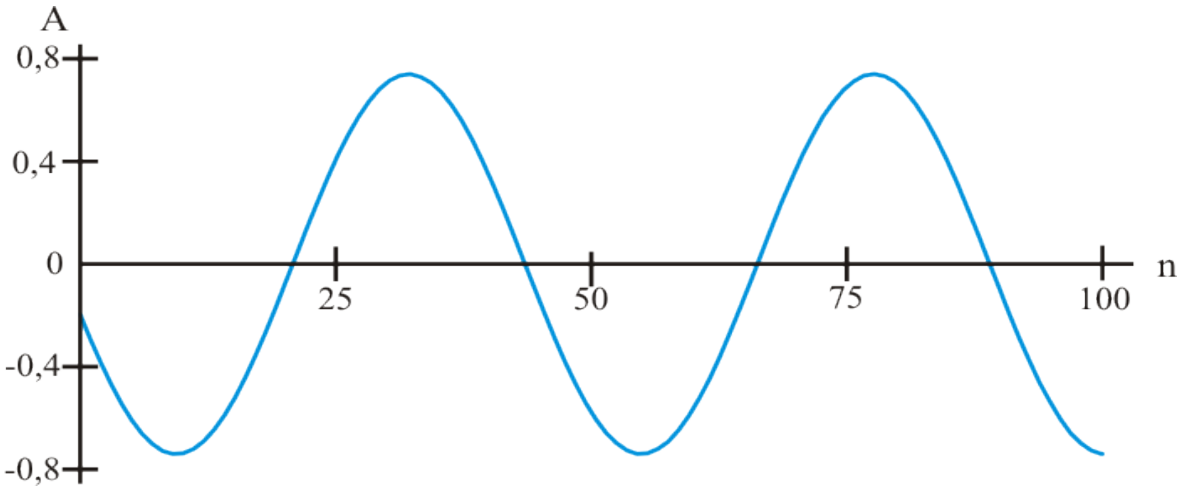
SEZNAM TABULEK

SEZNAM PŘÍLOH

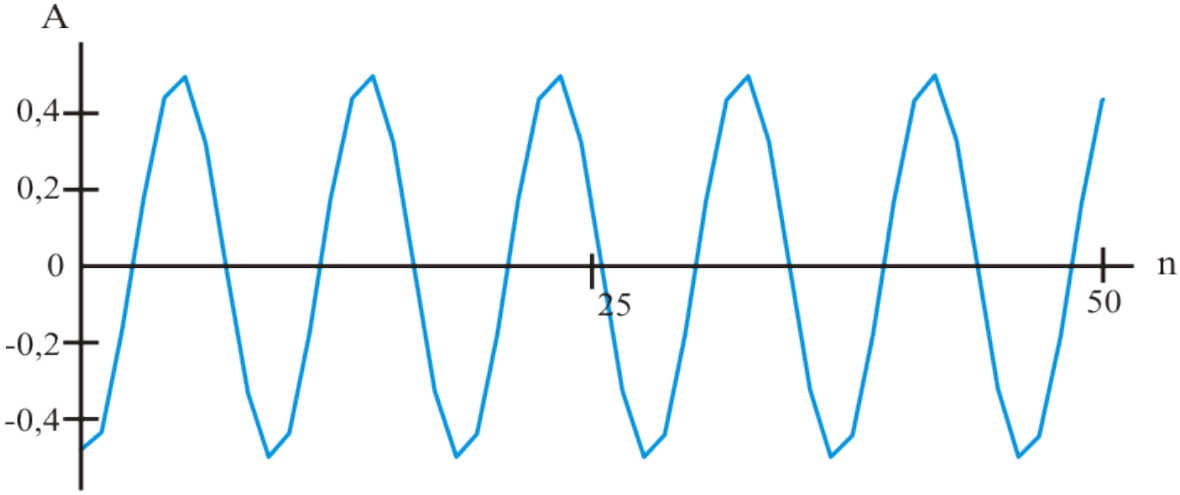
PŘÍLOHA P 1: ZKUŠEBNÍ SIGNÁLY



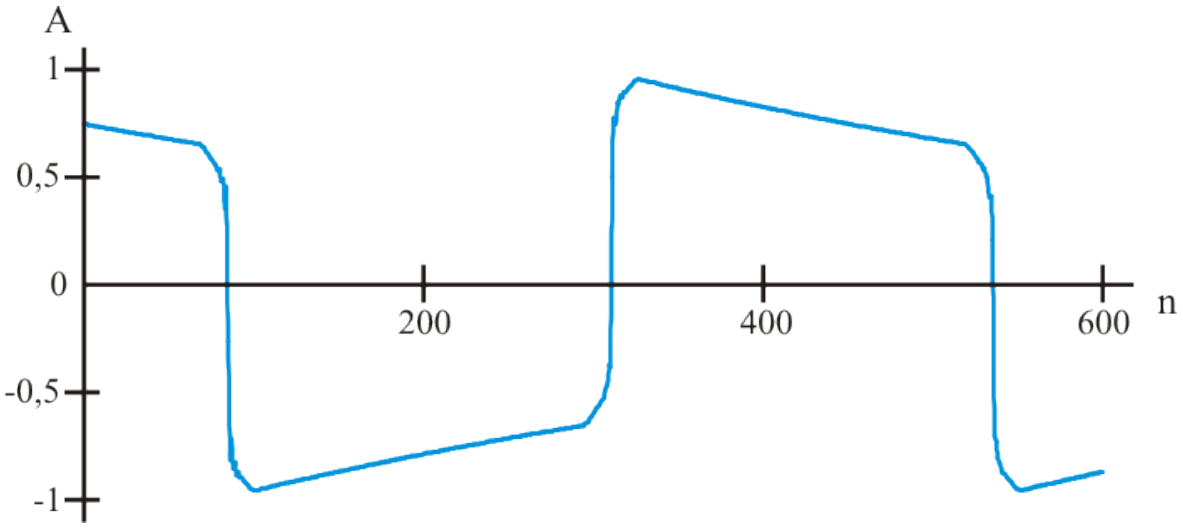
Obr. 18. Signál sinus o frekvenci f = 100 Hz



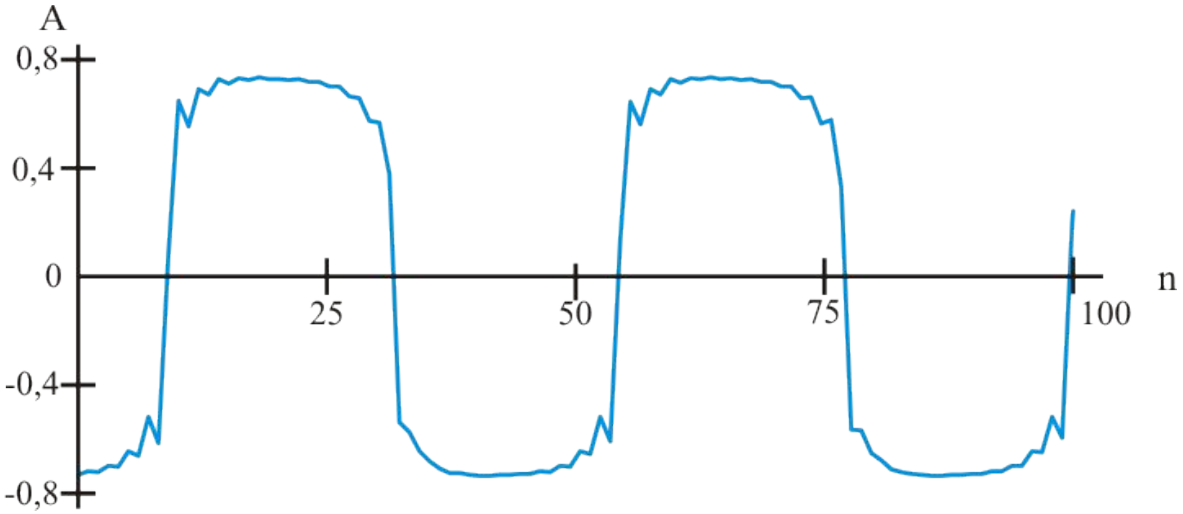
Obr. 19. Signál sinus o frekvenci f = 1000 Hz



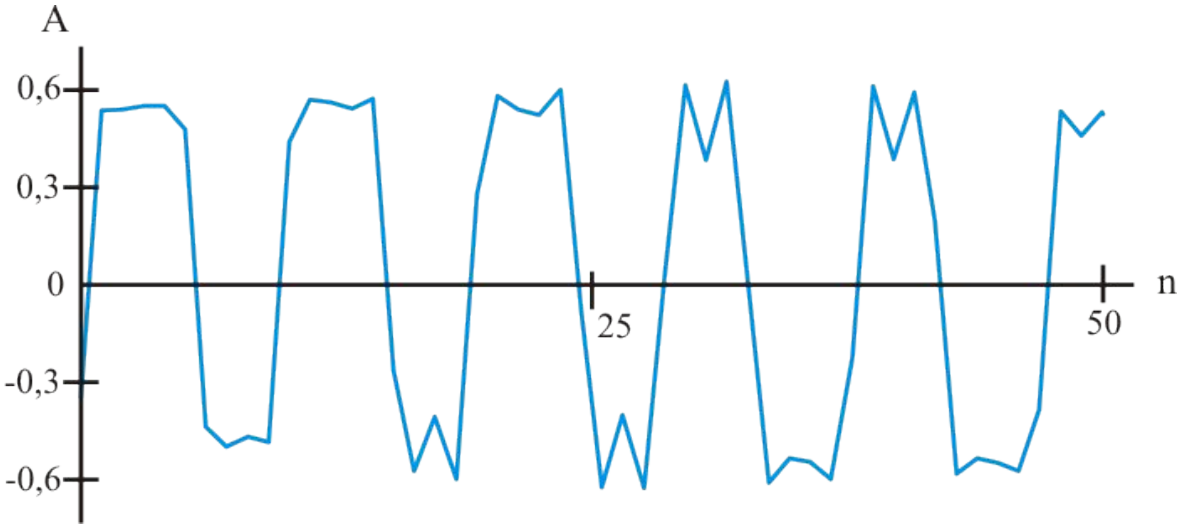
Obr. 20. Signál sinus o frekvenci f = 5000 Hz



Obr. 21. Signál obdélník o frekvenci f = 100 Hz

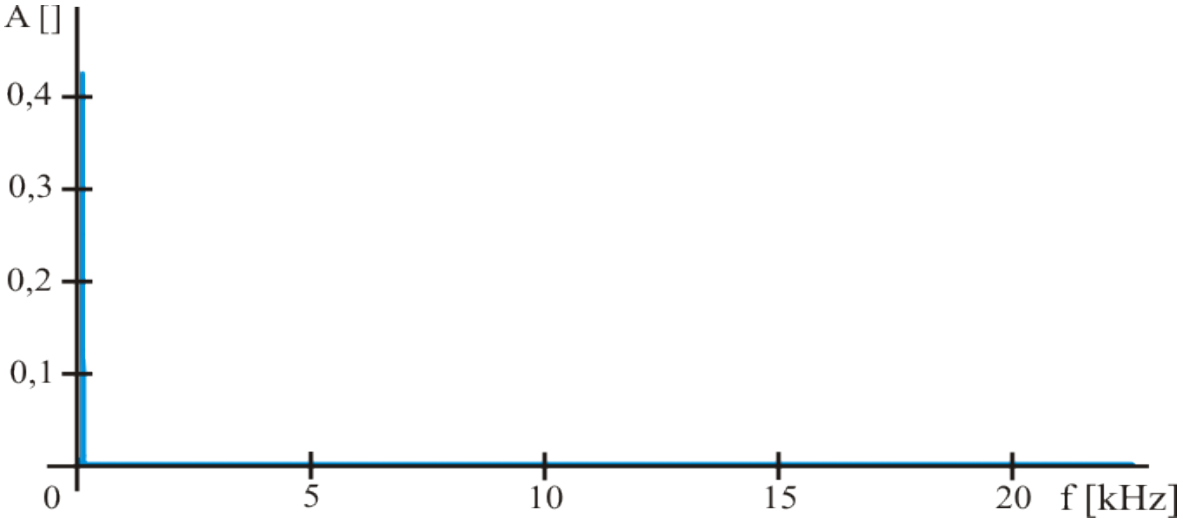


Obr. 22. Signál obdélník o frekvenci f = 1000 Hz

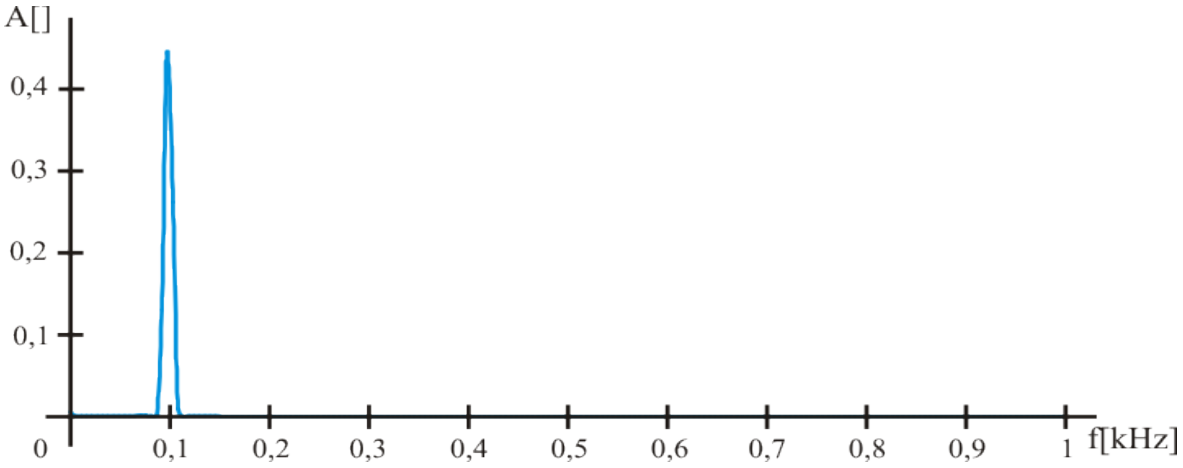


Obr. 23. Signál obdélník o frekvenci f = 5000 Hz

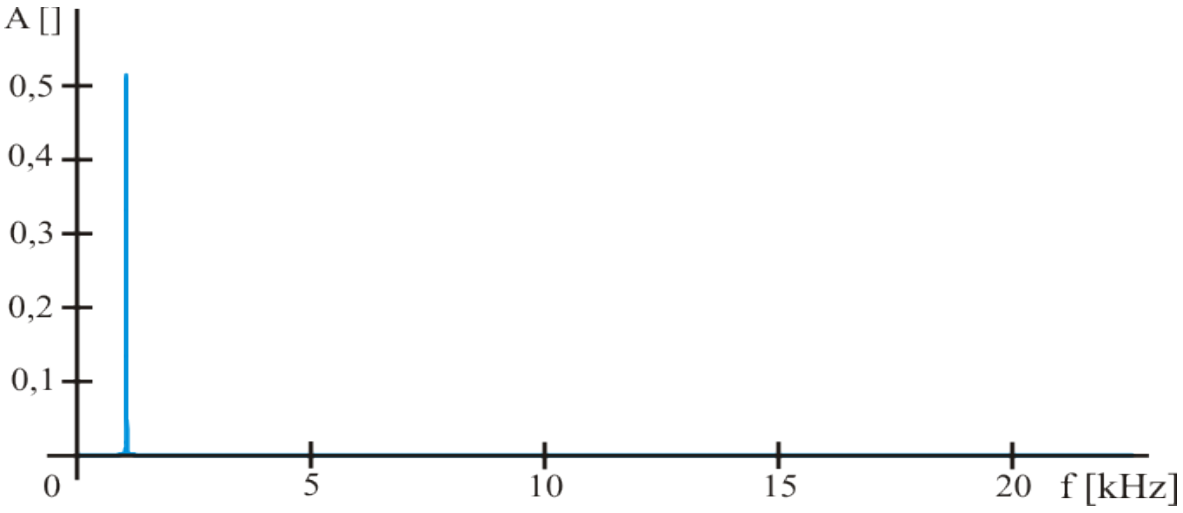
PŘÍLOHA P 2: SPEKTRA SIGNÁLŮ UPRAVENÝCH HAMMINGOVÝM OKNEM



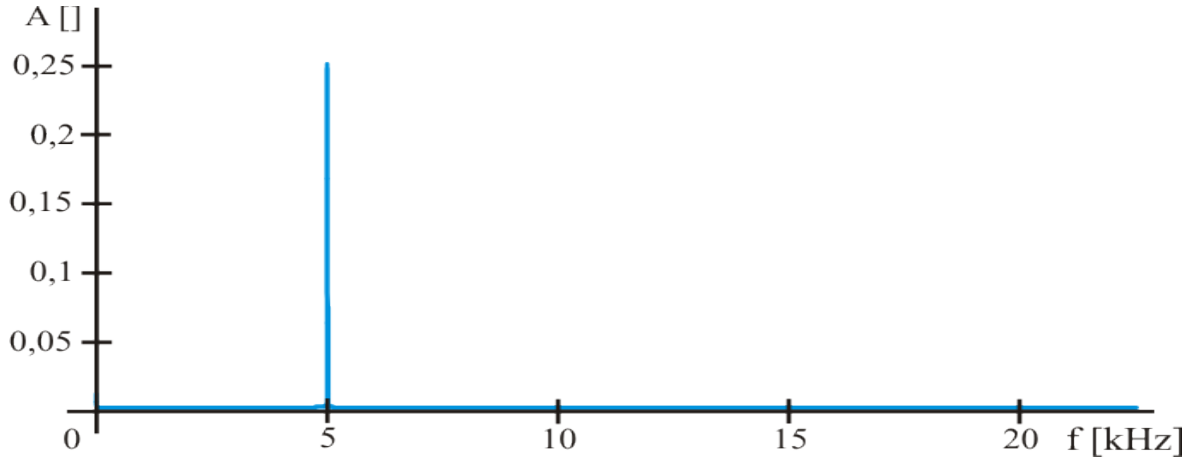
Obr. 24. Spektrum signálu sinus o f = 100 Hz upraveného Hammingovým oknem



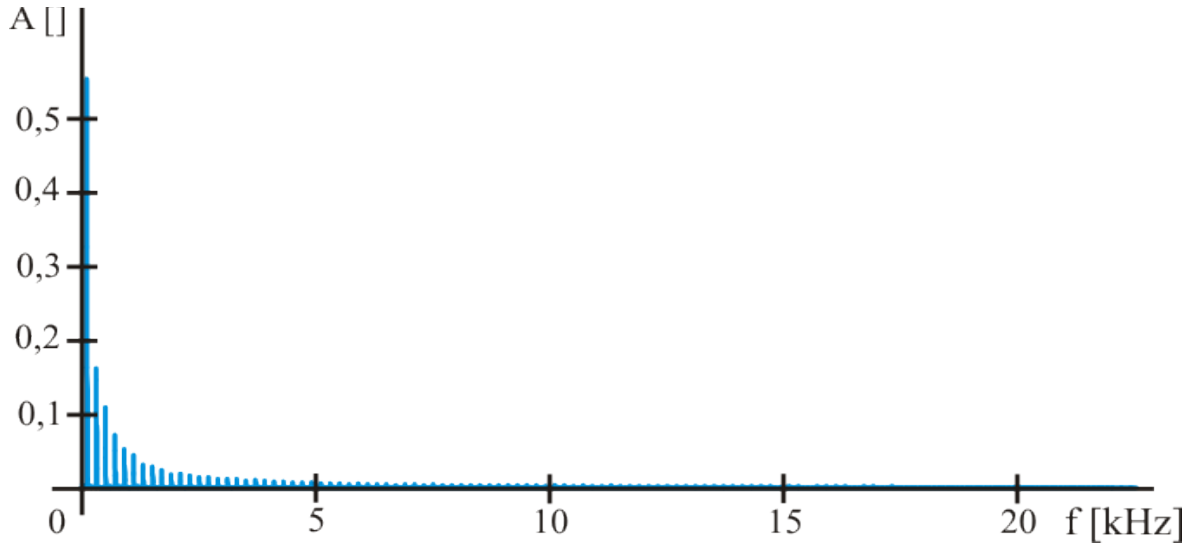
Obr. 25. Detail spektra signálu sinus o f = 100 Hz upraveného Hammingovým oknem



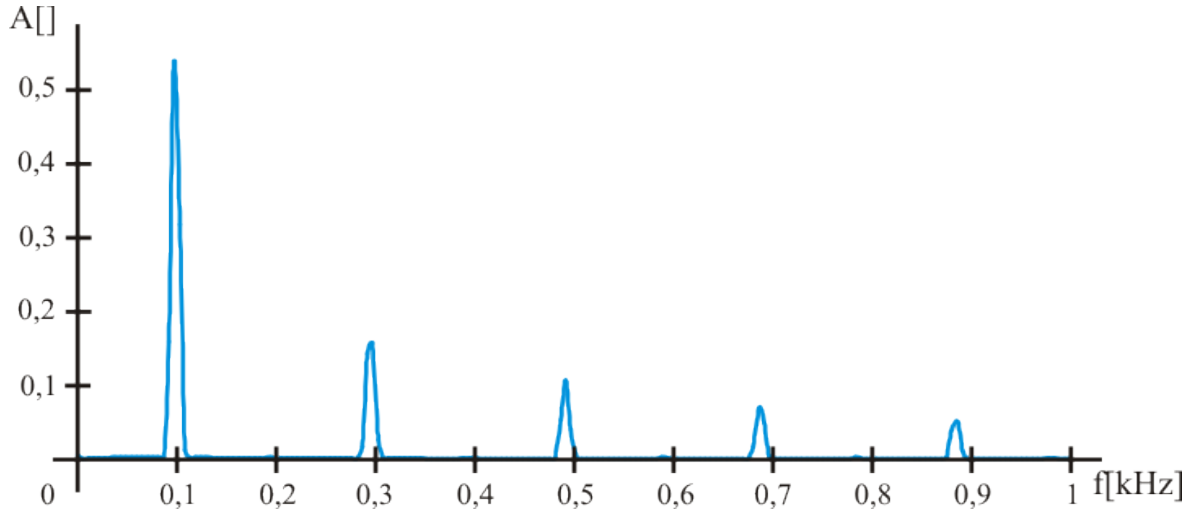
Obr. 26. Spektrum signálu sinus o f = 1000 Hz upraveného Hammingovým oknem



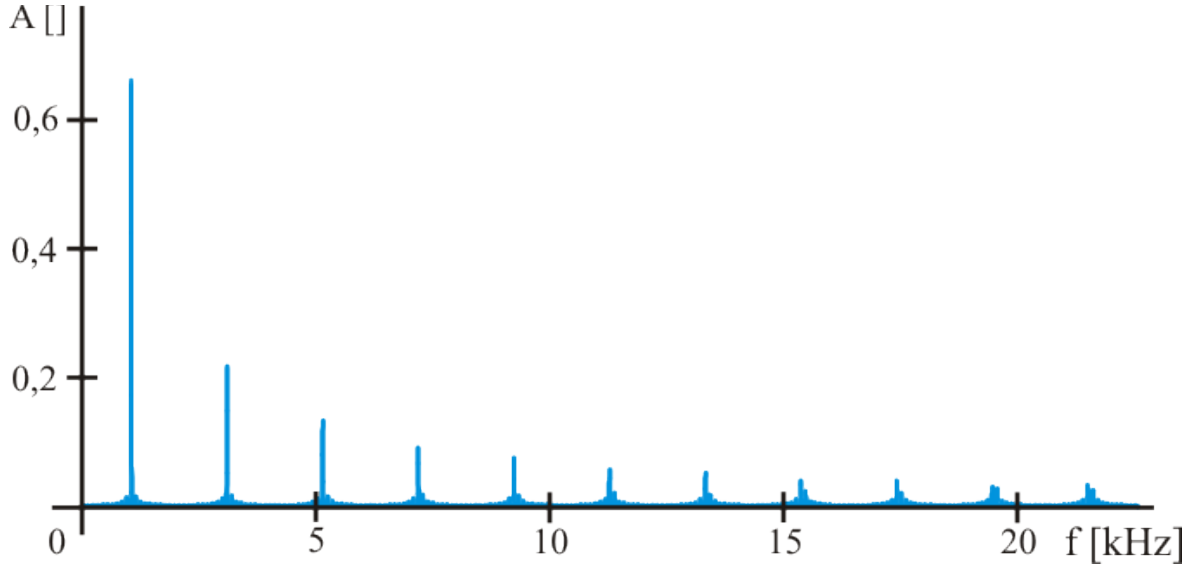
Obr. 27. Spektrum signálu sinus o f = 5000 Hz upraveného Hammingovým oknem



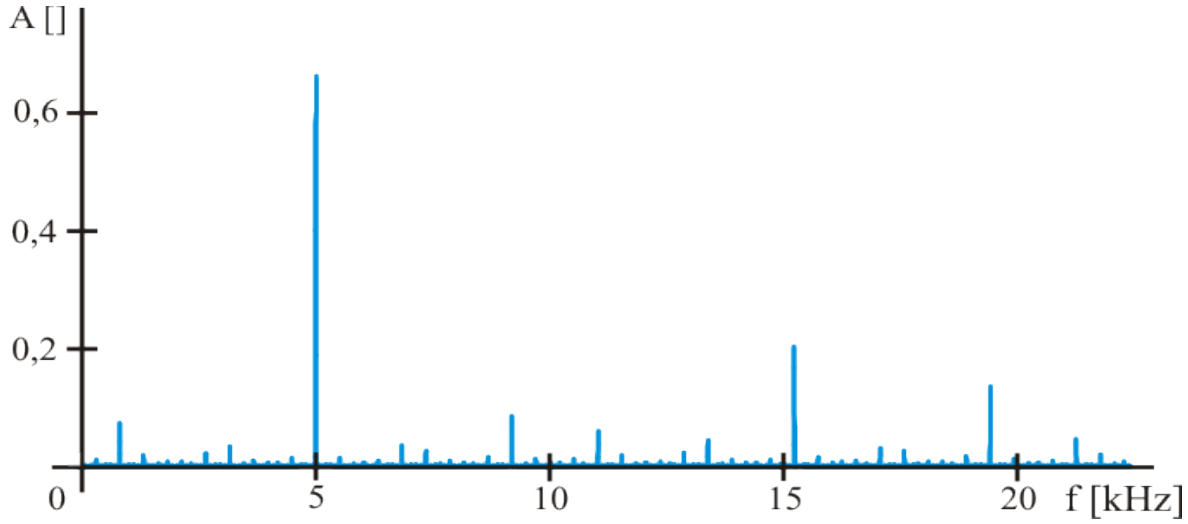
Obr. 28. Spektrum signálu obdélník o f = 100 Hz upraveného Hammingovým oknem



Obr. 29. Detail spektra signálu obdélník o f = 100 Hz upraveného Hammingovým oknem

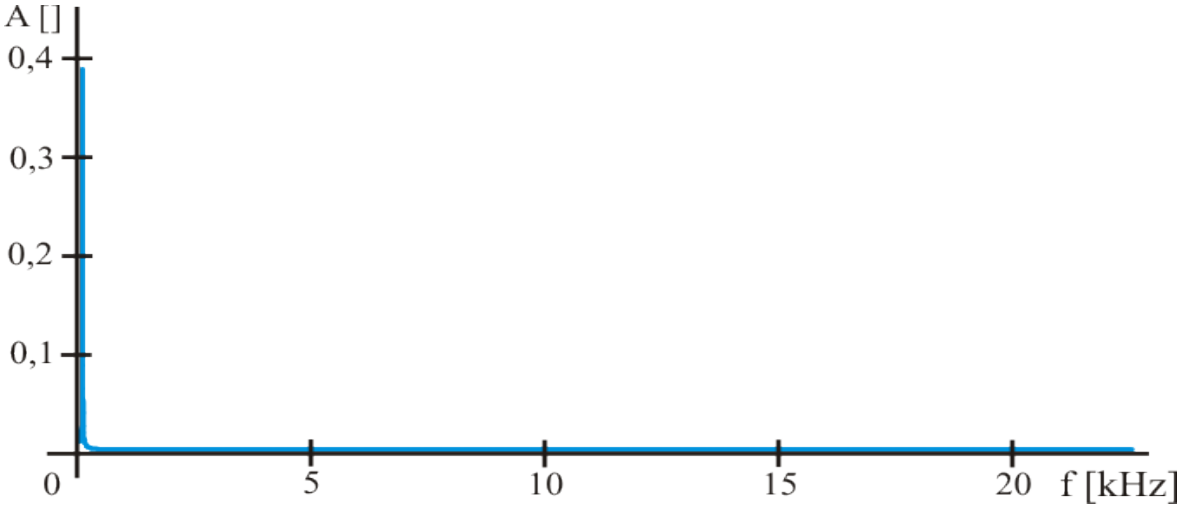


Obr. 30. Spektrum signálu obdélník o f = 1000 Hz upraveného Hammingovým oknem

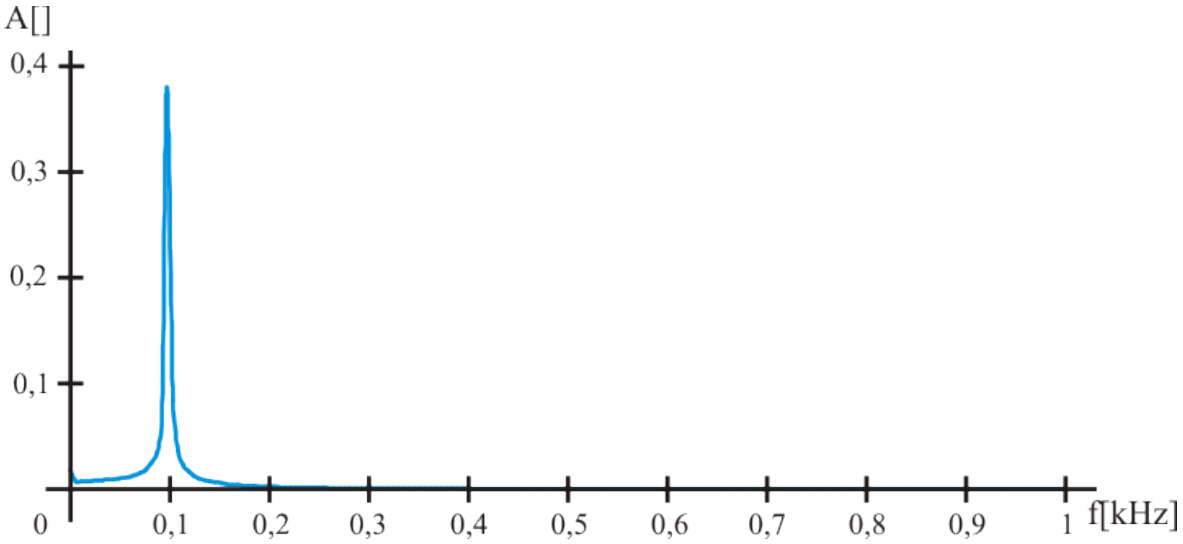


Obr. 31. Spektrum signálu obdélník o f = 5000 Hz upraveného Hammingovým oknem

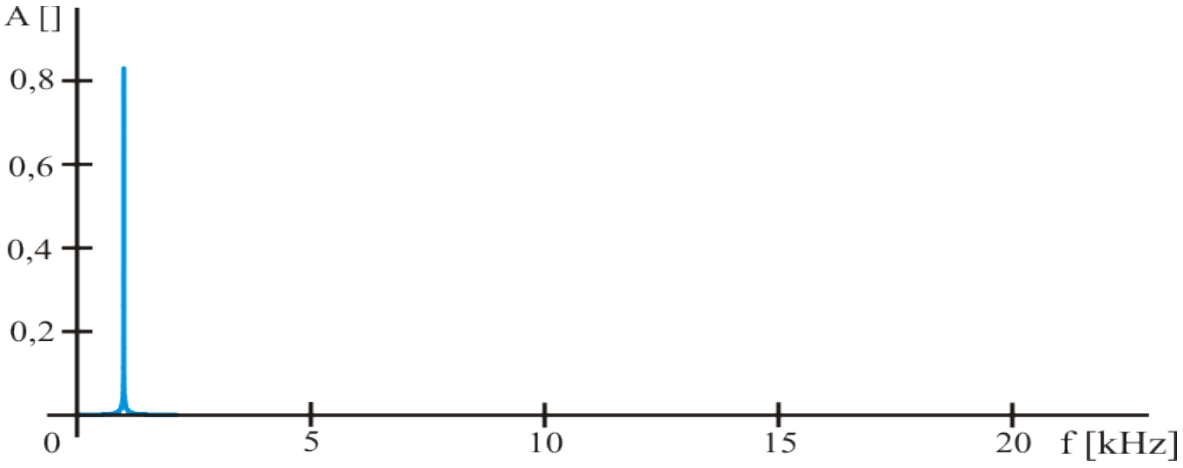
PŘÍLOHA P 3: SPEKTRA SIGNÁLŮ UPRAVENÝCH OBDÉLNÍKOVÝM OKNEM



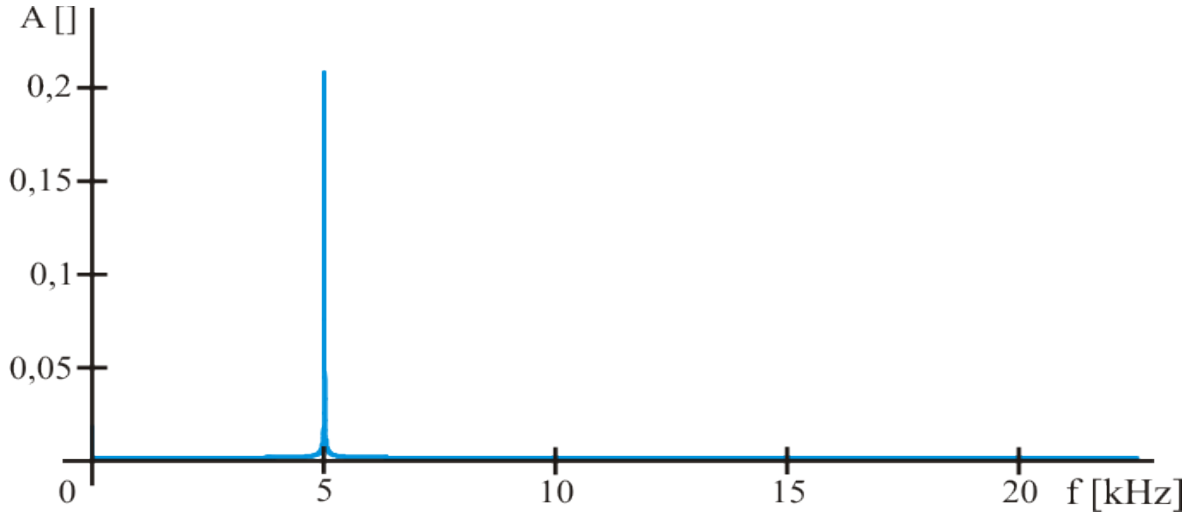
Obr. 32. Spektrum signálu sinus o f = 100 Hz upraveného obdélníkovým oknem



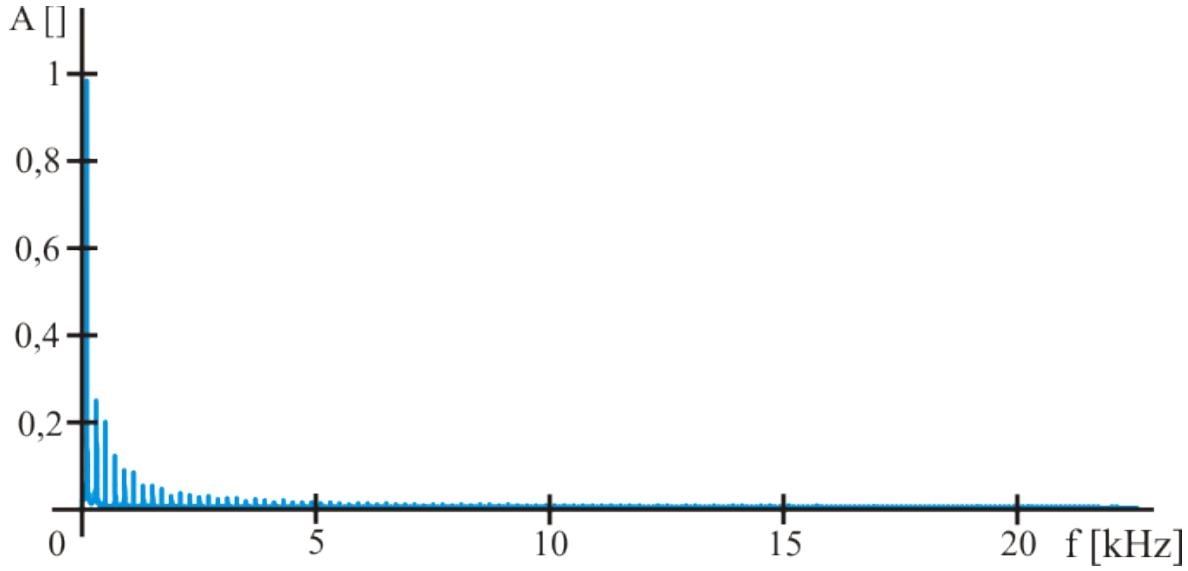
Obr. 33. Detail spektra signálu sinus o f = 100 Hz upraveného obdélníkovým oknem



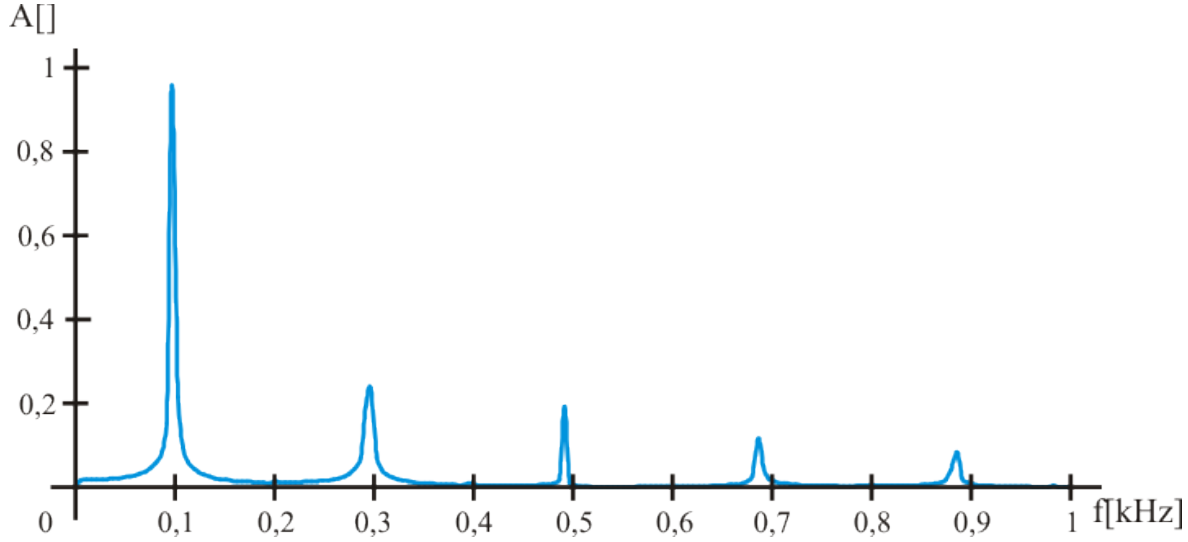
Obr. 34. Spektrum signálu sinus o f = 1000 Hz upraveného obdélníkovým oknem



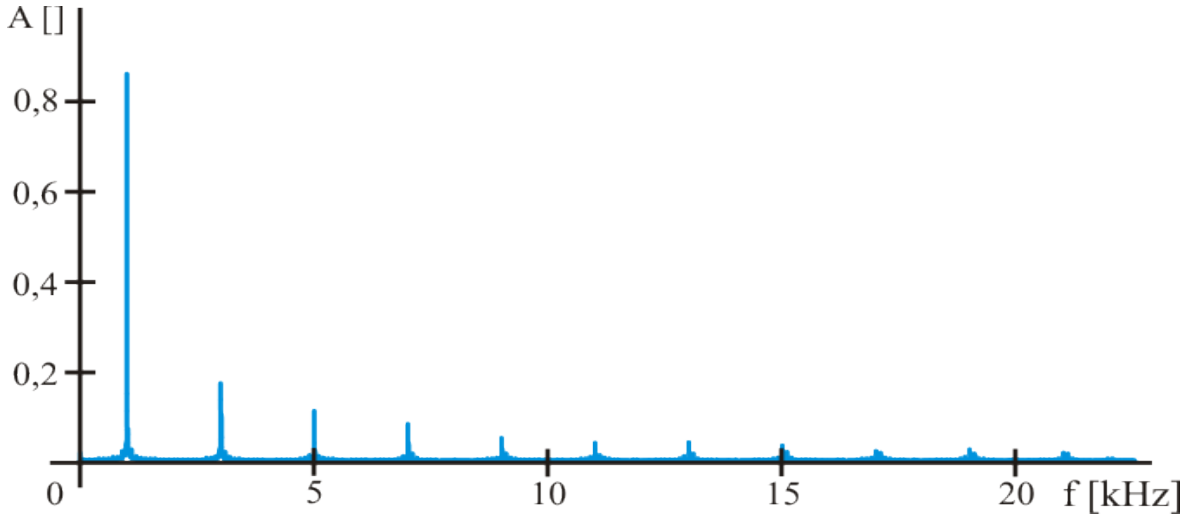
Obr. 35. Spektrum signálu sinus o f = 5000 Hz upraveného obdélníkovým oknem



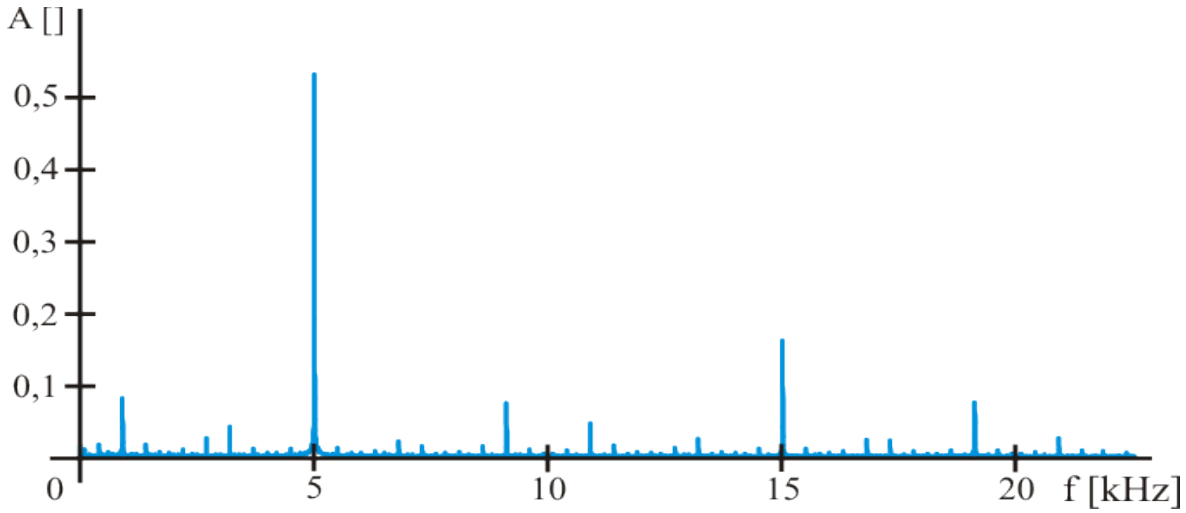
Obr. 36. Spektrum signálu obdélník o f = 100 Hz upraveného obdélníkovým oknem



Obr. 37. Detail spektra signálu obdélník o f = 100 Hz upraveného obdélníkovým oknem

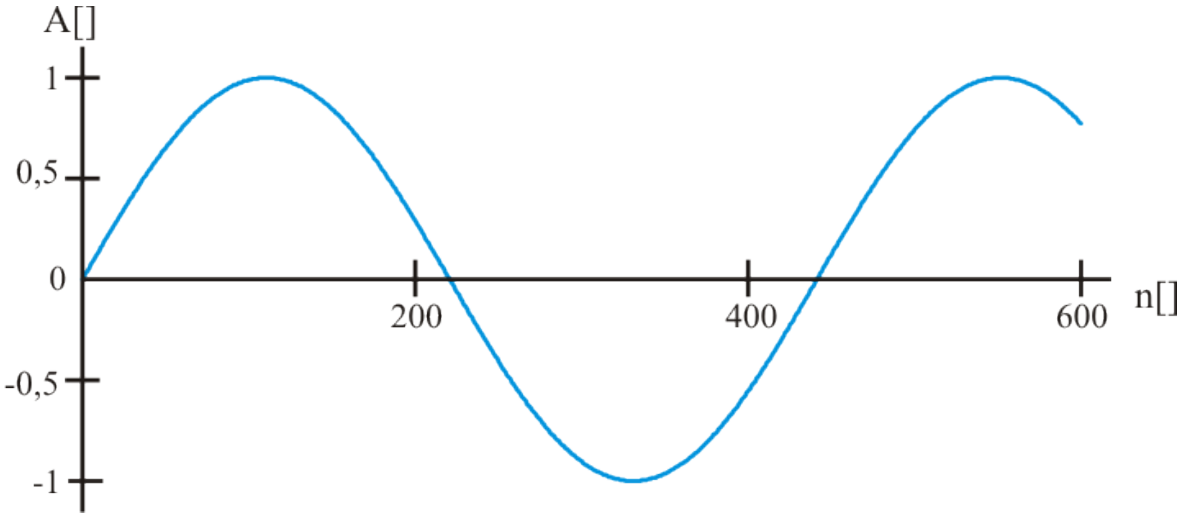


Obr. 38. Spektrum signálu obdélník o f = 1000 Hz upraveného obdélníkovým oknem

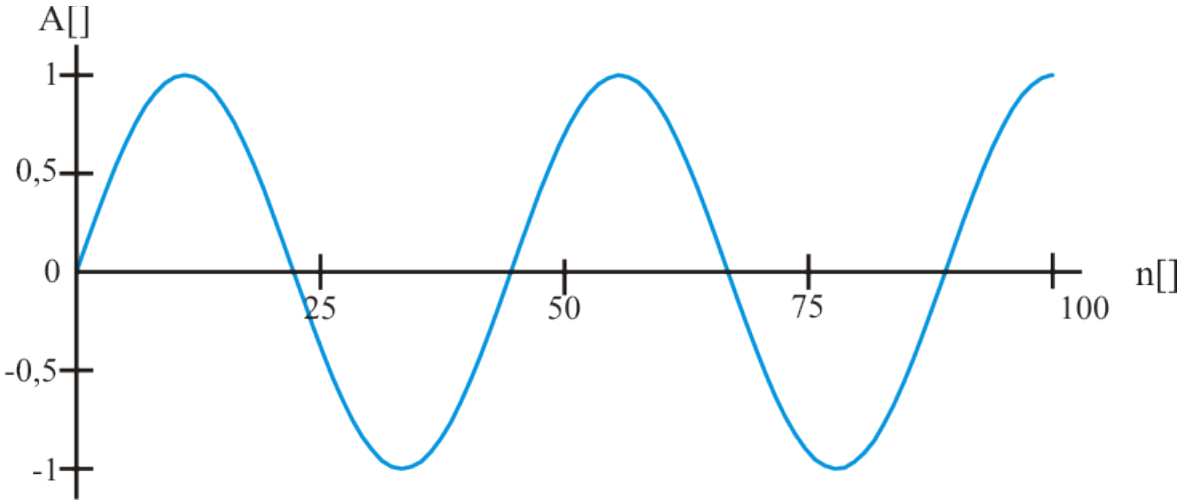


Obr. 39. Spektrum signálu obdélník o f = 5000 Hz upraveného obdélníkovým oknem

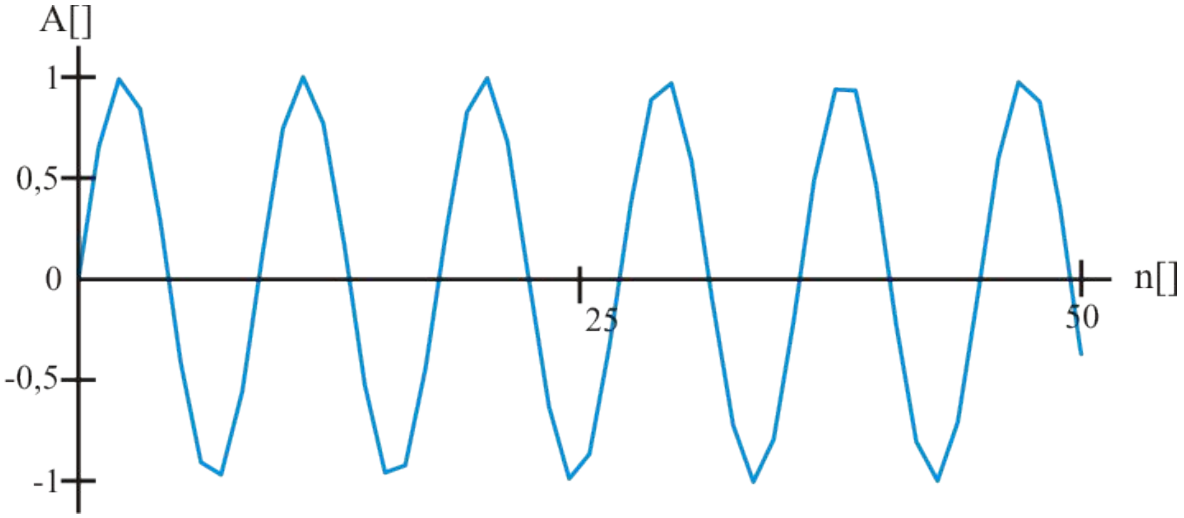
PŘÍLOHA P 4: IDEÁLNÍ SIGNÁLY



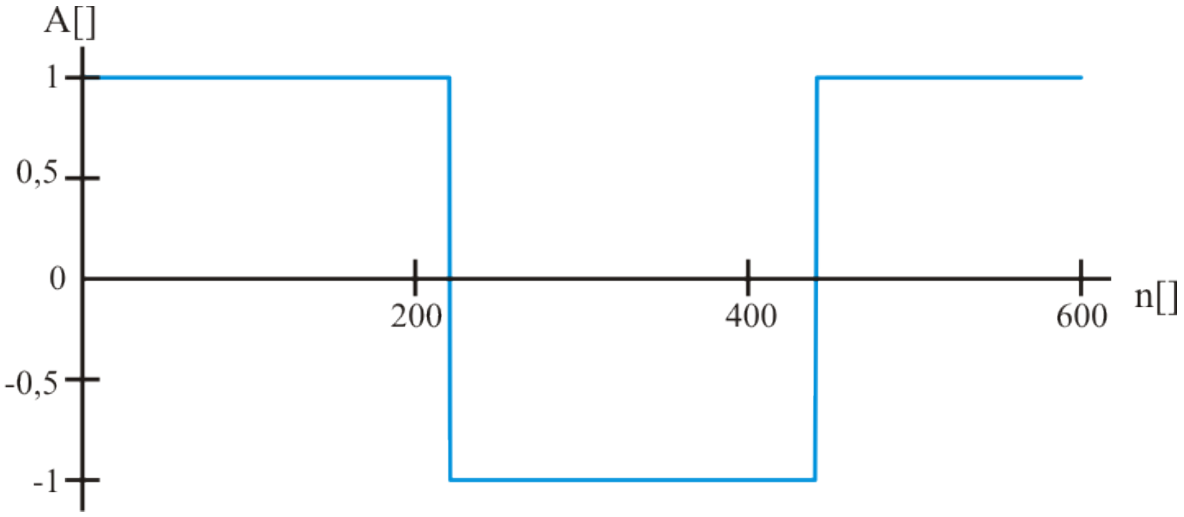
Obr. 40. Ideální signál sinus o f = 100 Hz



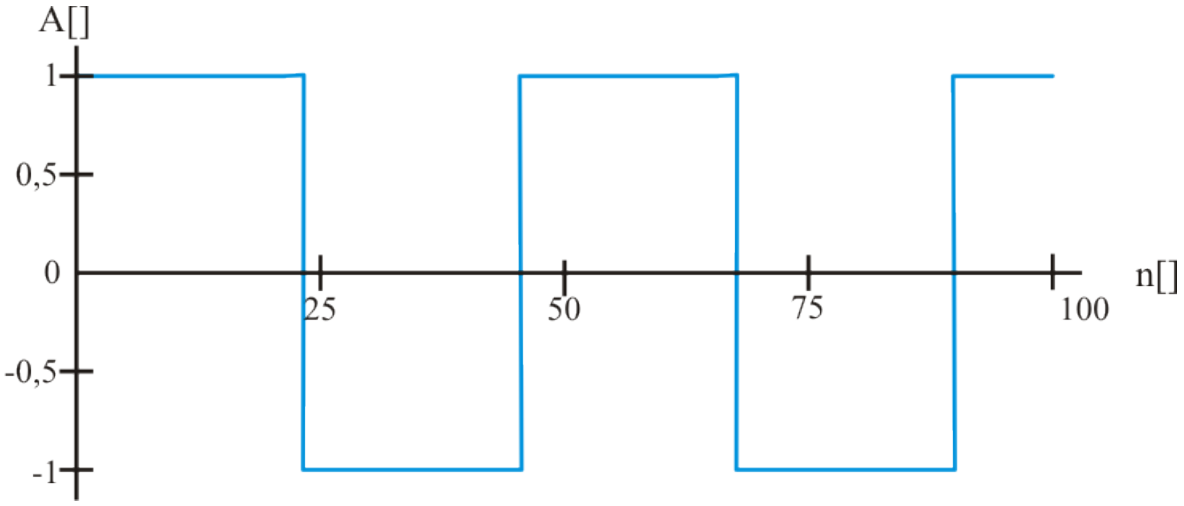
Obr. 41. Ideální signál sinus o f = 1000 Hz



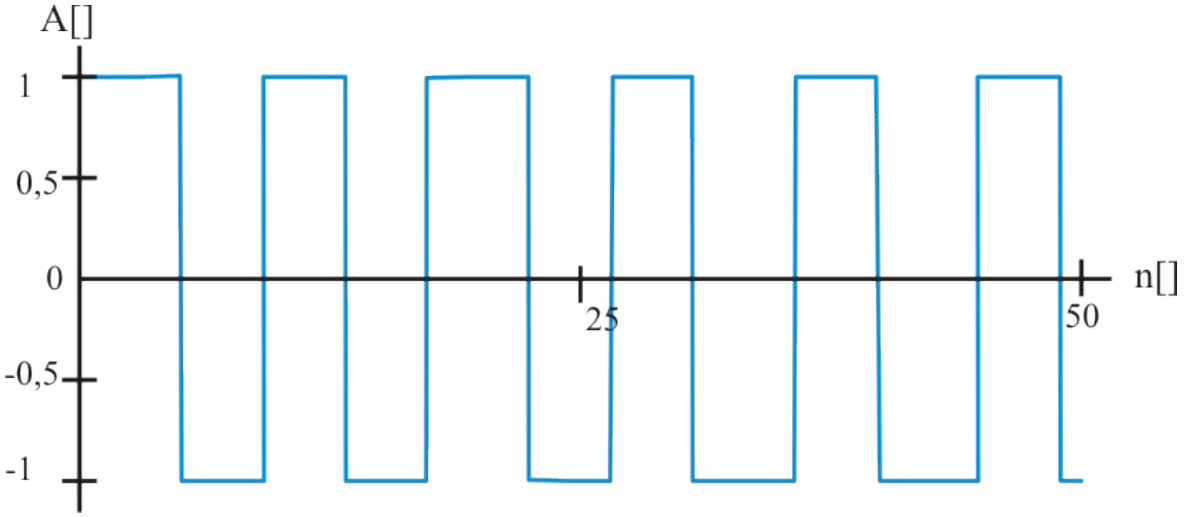
Obr. 42. Ideální signál sinus o f = 5000 Hz



Obr. 43. Ideální signál obdélník o f = 100 Hz

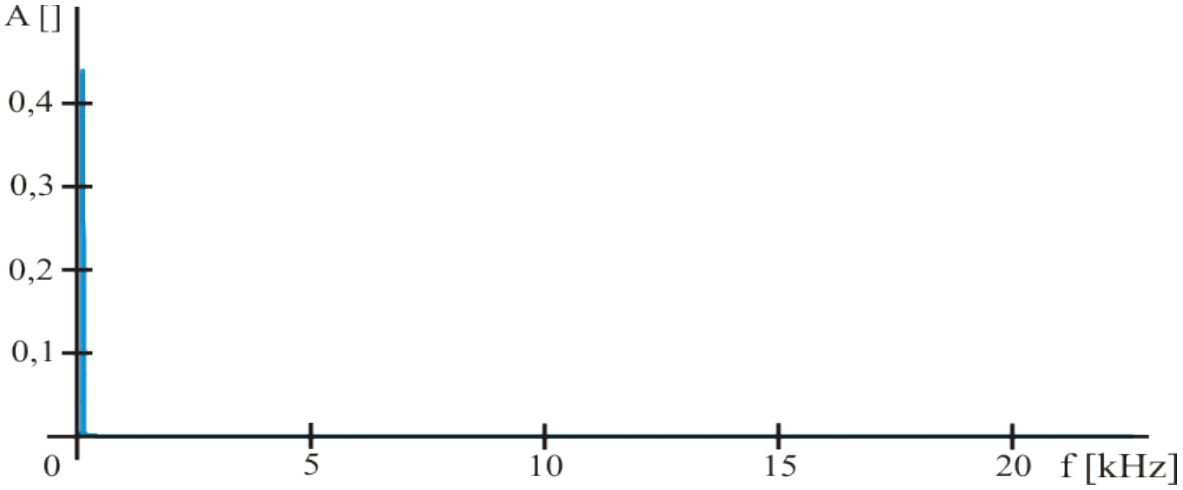


Obr. 44. Ideální signál obdélník o f = 1000 Hz

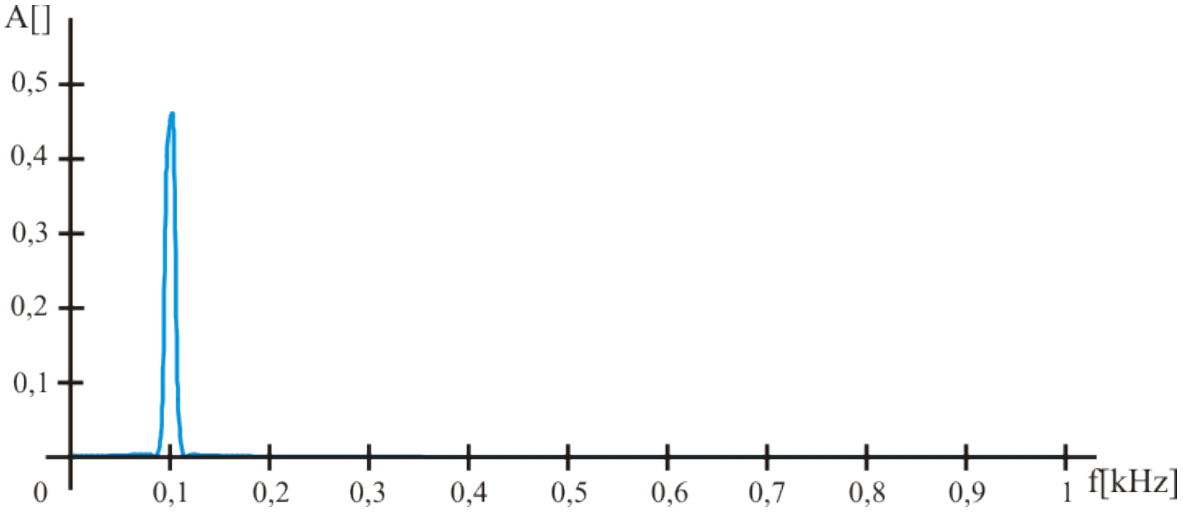


Obr. 45. Ideální signál obdélník o f = 5000 Hz

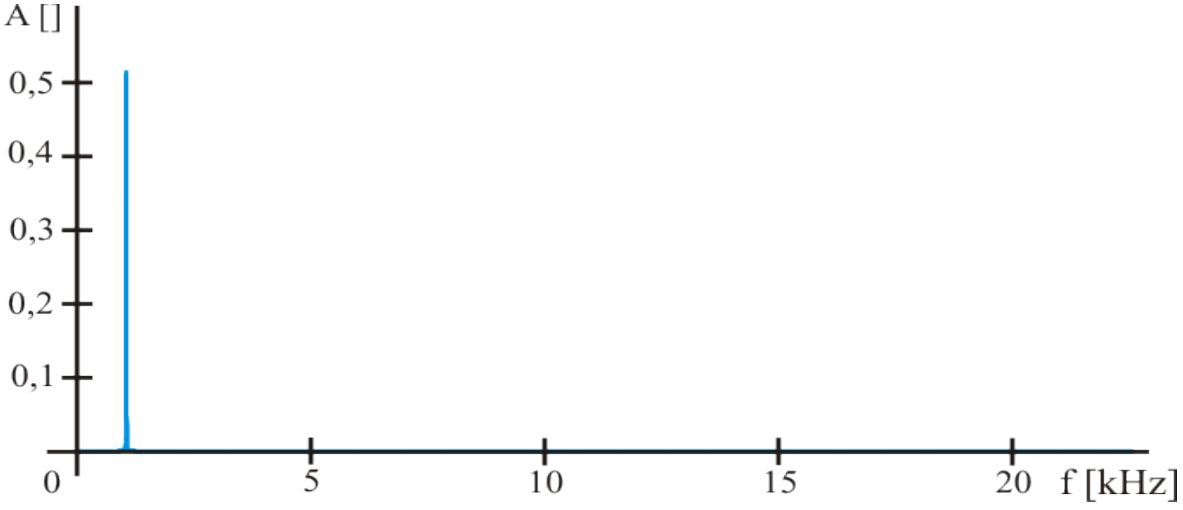
PŘÍLOHA P 5: SPEKTRA IDEÁLNÍCH SIGNÁLŮ UPRAVENÝCH HAMMINGOVÝM OKNEM



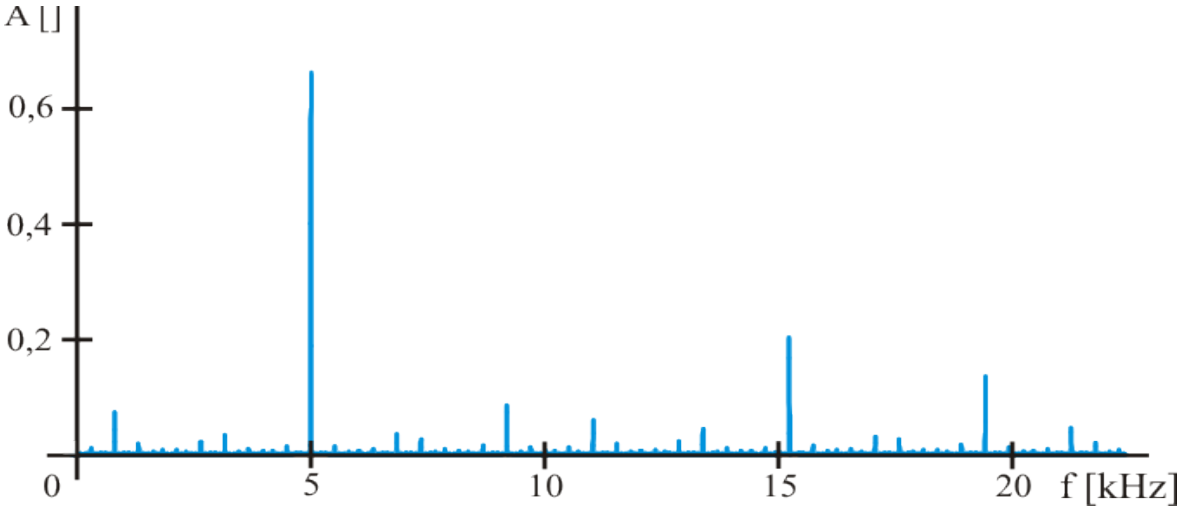
Obr. 46. Spektrum ideálního signálu sinus o f = 100 Hz upraveného Hammingovým oknem



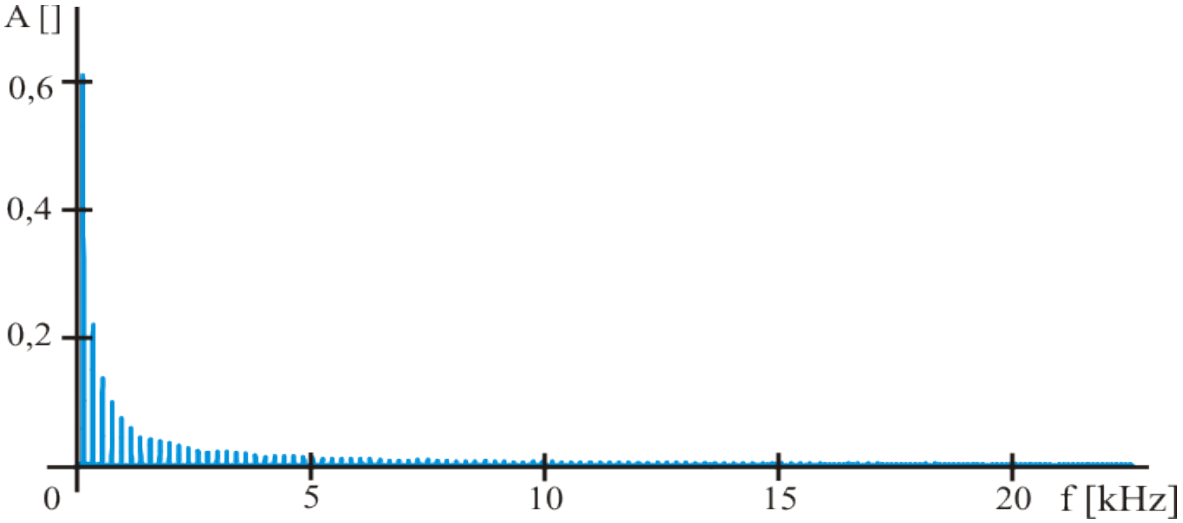
Obr. 47. Detail spektra ideálního signálu sinus o f = 100 Hz upraveného Hammingovým oknem



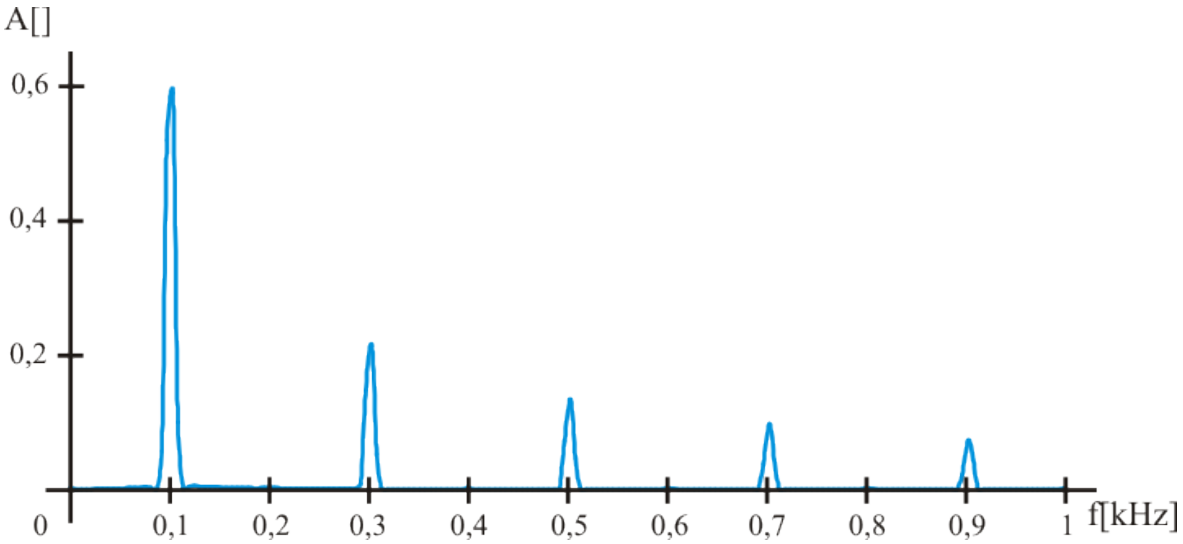
Obr. 48. Spektrum ideálního signálu sinus o f = 1000 Hz upraveného Hammingovým oknem



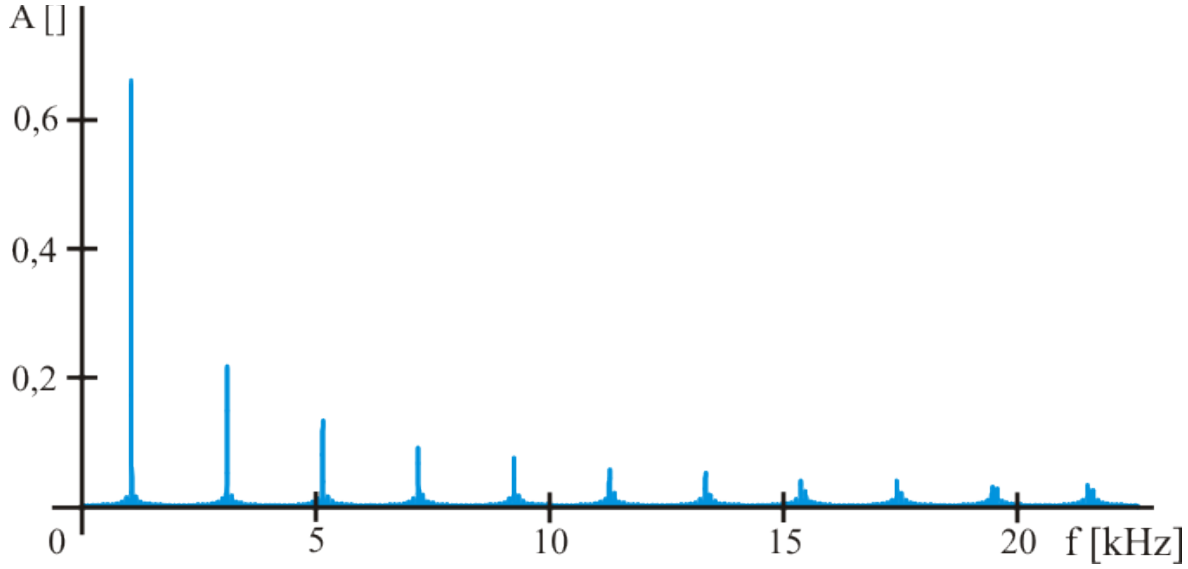
Obr. 49. Spektrum ideálního signálu sinus o f = 5000 Hz upraveného Hammingovým oknem



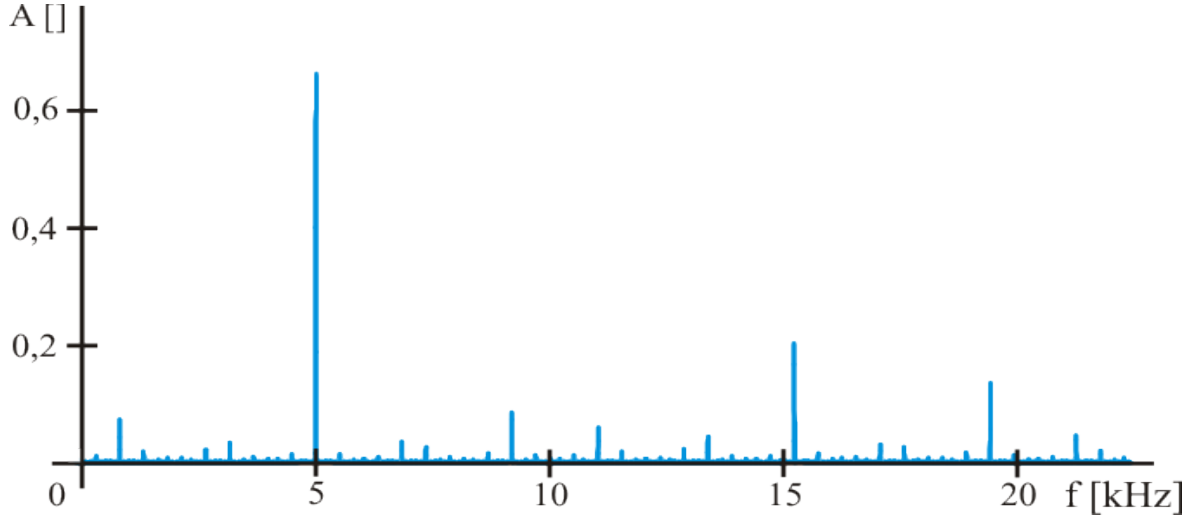
Obr. 50. Spektrum ideálního signálu obdélník o f = 100 Hz upraveného Hammingovým oknem



Obr. 51. Detail spektra ideálního signálu obdélník o f = 100 Hz upraveného Hammingovým oknem

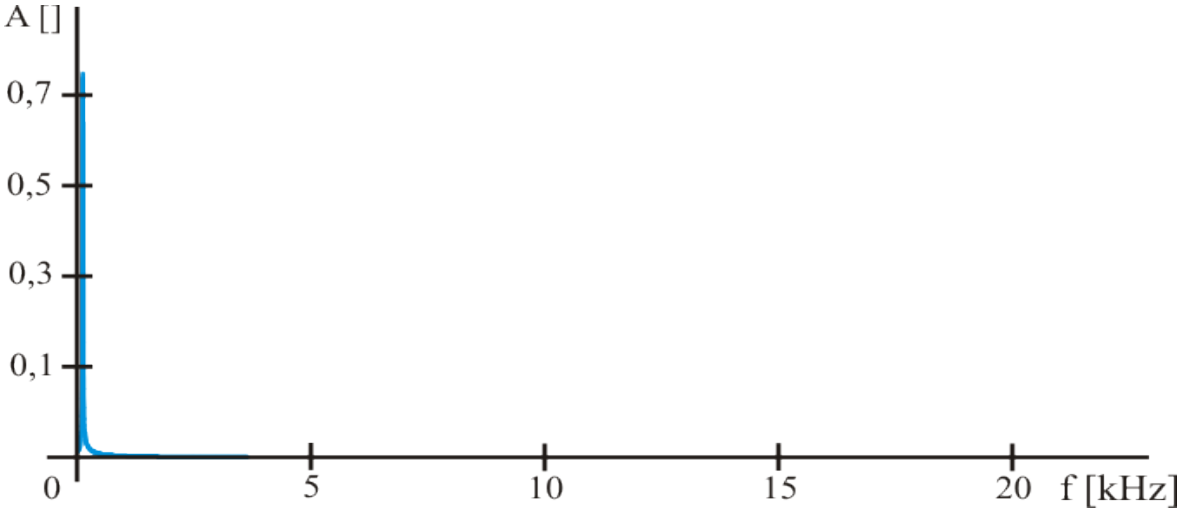


Obr. 52. Spektrum ideálního signálu obdélník o f = 1000 Hz upraveného Hammingovým oknem

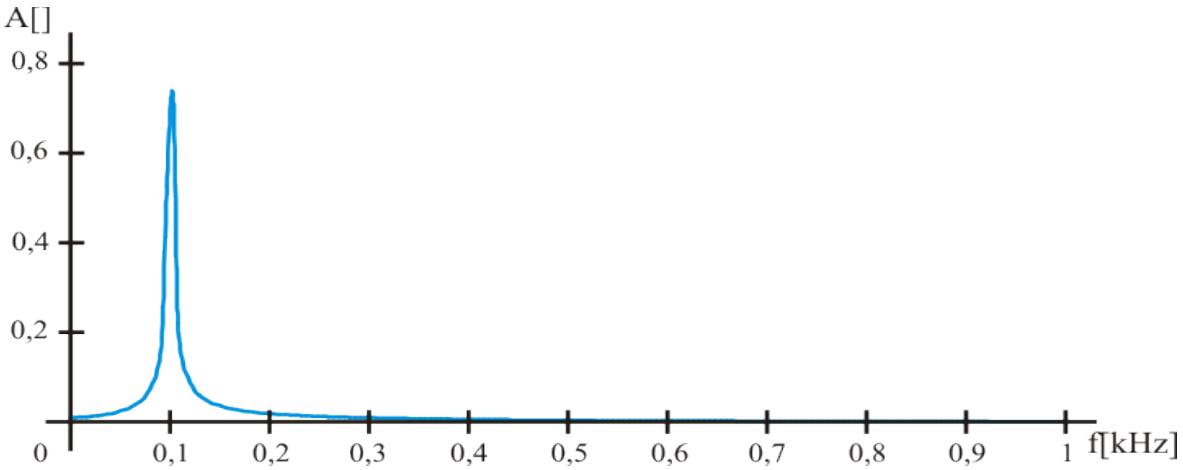


Obr. 53. Spektrum ideálního signálu obdélník o f = 5000 Hz upraveného Hammingovým oknem

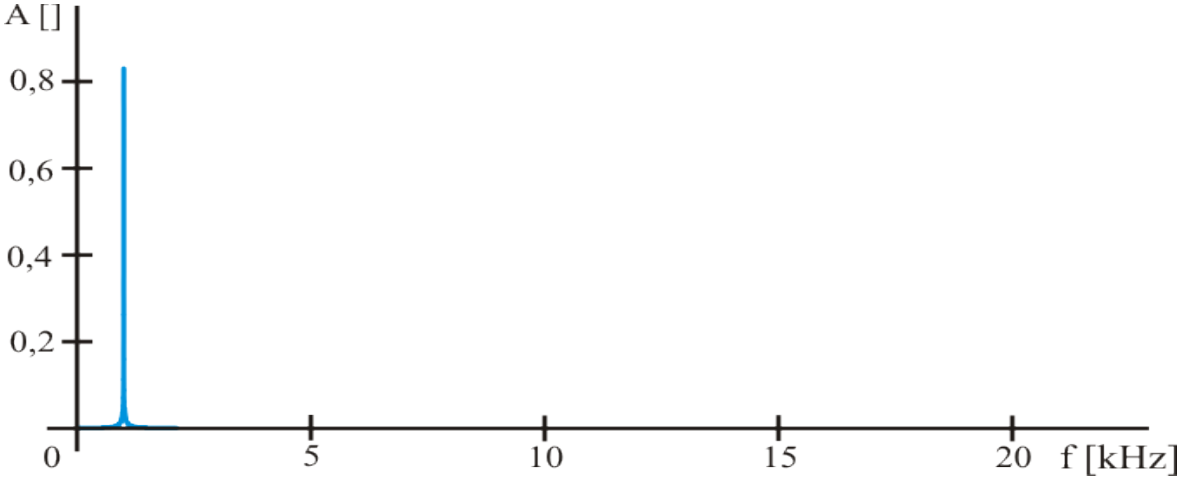
PŘÍLOHA P 6: SPEKTRA IDEÁLNÍCH SIGNÁLŮ UPRAVENÝCH OBDÉLNÍKOVÝM OKNEM



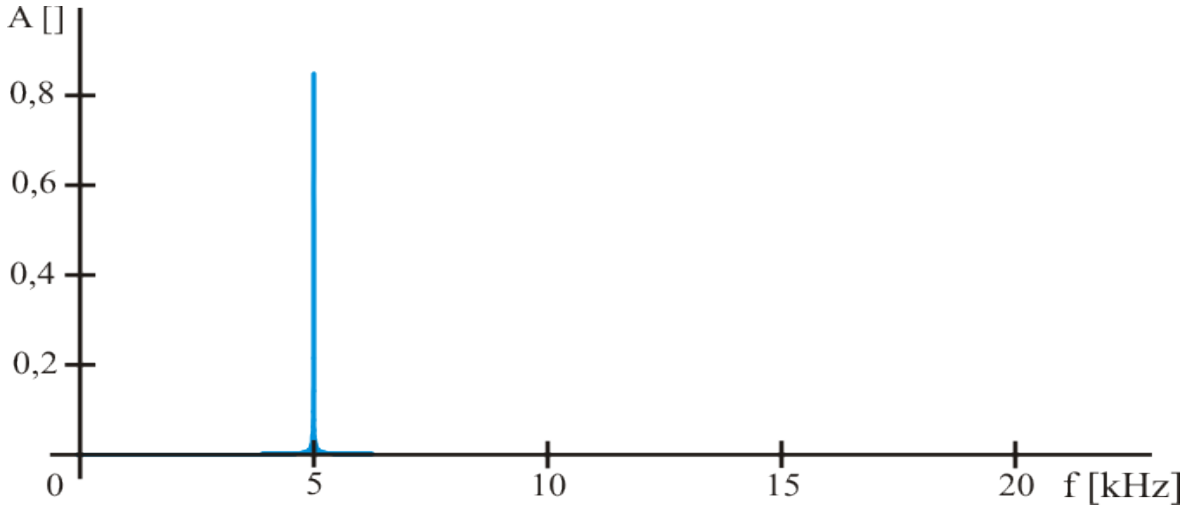
Obr. 54. Spektrum ideálního signálu sinus o f = 100 Hz upraveného obdélníkovým oknem



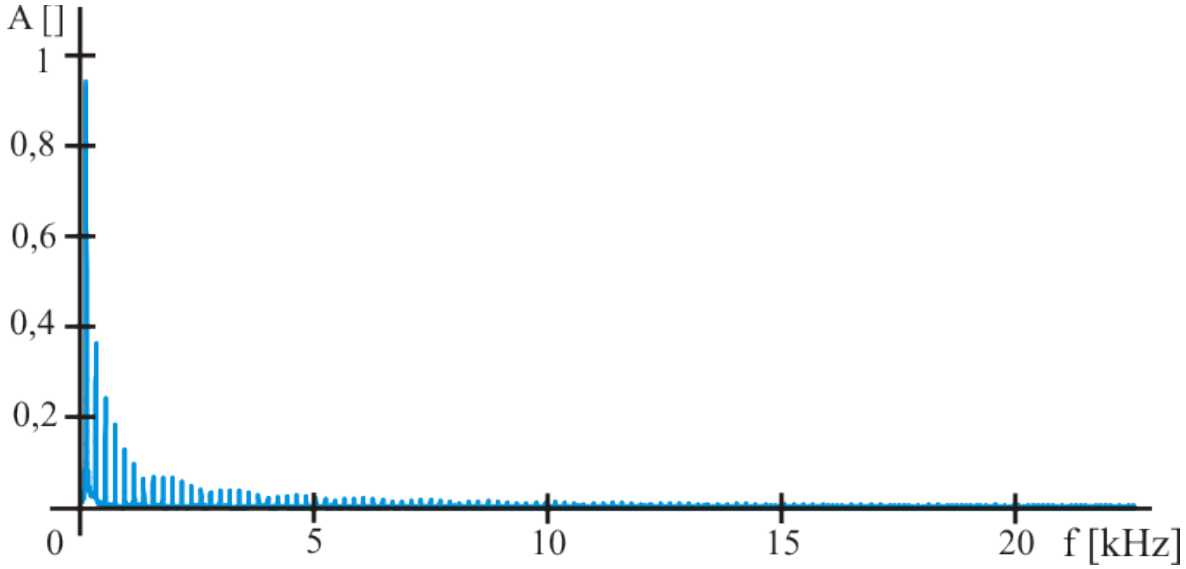
Obr. 55. Detail spektra ideálního signálu sinus o f = 100 Hz upraveného obdélníkovým oknem



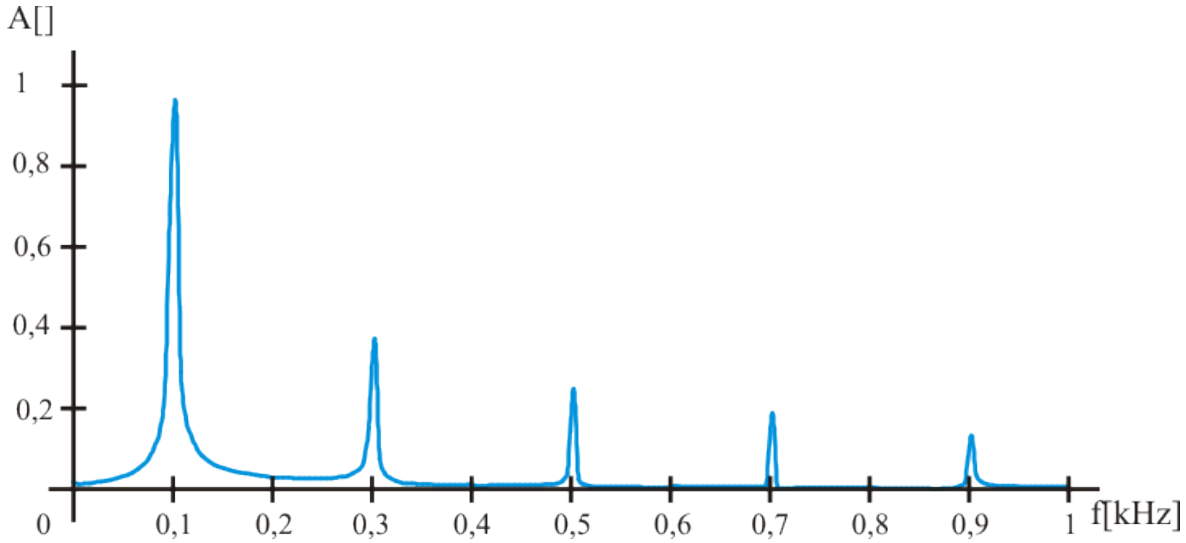
Obr. 56. Spektrum ideálního signálu sinus o f = 1000 Hz upraveného obdélníkovým oknem



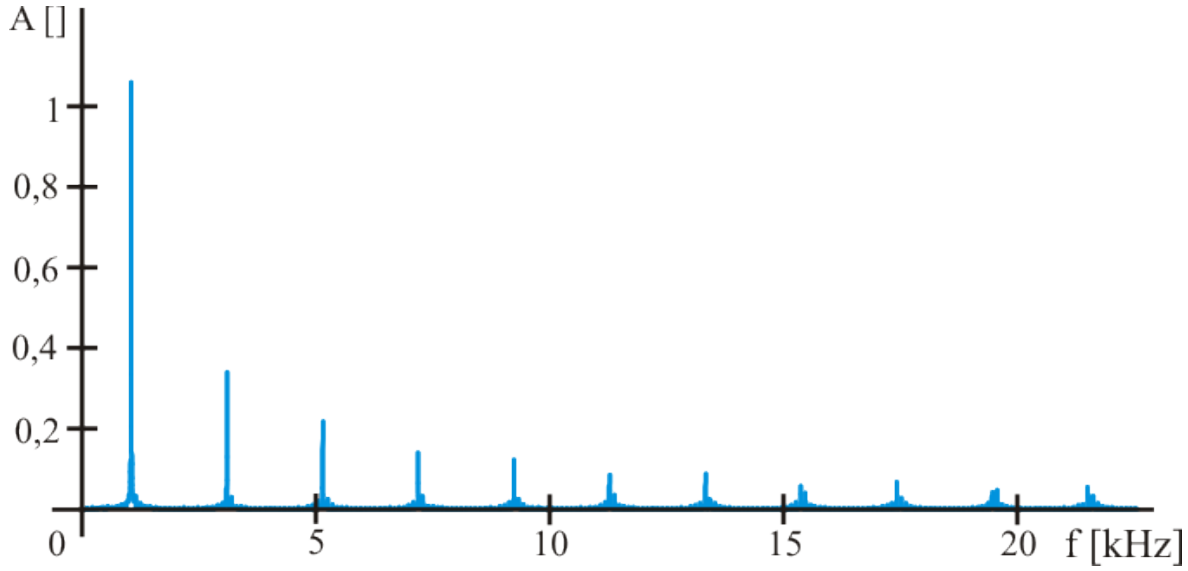
Obr. 57. Spektrum ideálního signálu sinus o f = 5000 Hz upraveného obdélníkovým oknem



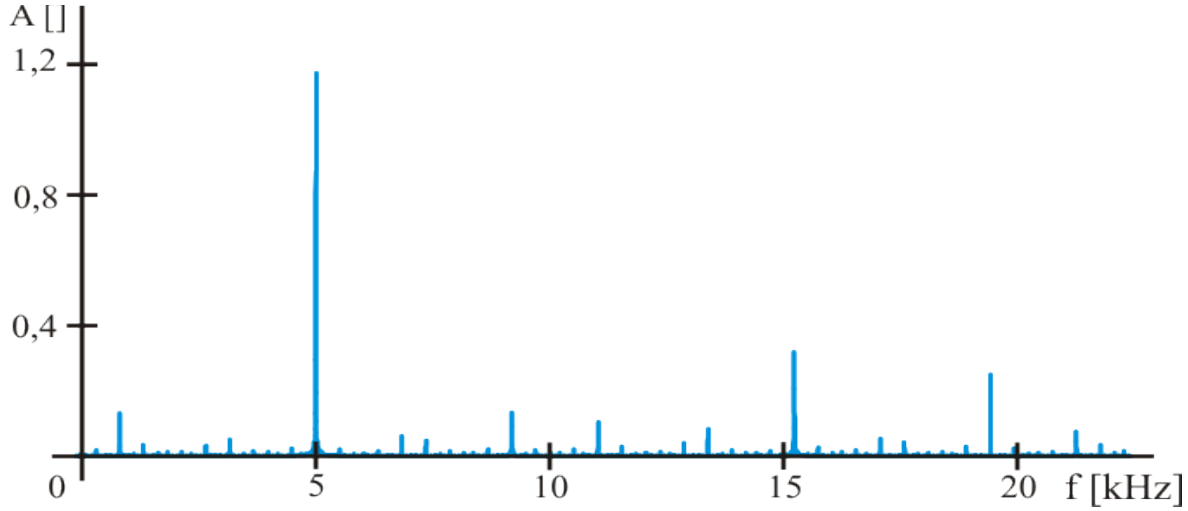
Obr. 58. Spektrum ideálního signálu obdélník o f = 100 Hz upraveného obdélníkovým oknem



Obr. 59. Detail spektra ideálního signálu obdélník o f = 100 Hz upraveného obdélníkovým oknem



Obr. 60. Spektrum ideálního signálu obdélník o f = 1000 Hz upraveného obdélníkovým oknem



Obr. 61. Spektrum ideálního signálu obdélník o f = 5000 Hz upraveného obdélníkovým oknem