ISS/VSG Projekt 2021/22

Honza Černocký, Honza Brukner a Honza Švec, ÚPGM FIT VUT

November 17, 2021

1 Úvod

Často se stane, že dostaneme signál zarušený nějakými artefakty a je potřeba jej vyfiltrovat. Pro tento projekt jsme pro Vás připravili signály ze známé databáze TIMIT, kam se ale "zatoulaly" čtyři harmonicky vztažené cosinusovky. Vaším úkolem je signál analyzovat, najít, na kterých frekvencích cosinusovky jsou, navrhnout filtr nebo filtry pro čištění signálu a pak signál vyčistit.

Projekt je individuální a je možno řešit v Python-u, Matlab-u, Octave, jazyce C nebo v libovolném jiném programovacím jazyce. Je možné použít libovolné knihovny. Projekt se nezaměřuje na "krásu programování", není tedy nutné mít vše úhledně zabalené do okomentovaných funkcí, ošetřené všechny chybové stavy, atd. Důležitý je výsledek. **Kód musí prokazatelně produkovat výsledky obsažené ve Vašem protokolu.**

V zadání se občas vyskytují funkce, či jiné části kódu, jako nápovědy. Tyto ukázky jsou v Python-u s použitím knihovny `numpy` (přesněji `import numpy as np`).

2 Vstup

Váš osobní signál máte v souboru `https://www.fit.vutbr.cz/study/courses/ISS/public/proj2021-22/signals/xlogin00.wav`, kde "xlogin00" je login pro studenty FIT, případně šestimístné číslo studenta pro studenty FSI. Váš login nebo číslo studenta musíte vypsat, adresář není možné vylistovat (nerad bych dával světu spam-list všech studentů ISS/VSG).

3 Odevzdání projektu

bude probíhat do informačního systému WIS (studenty FSI prosím o zaslání řešení emailem) ve dvou souborech:

1. `xlogin00.pdf` nebo `xlogin00_e.pdf` (kde "xlogin00" je Váš login či číslo studenta) je protokol s řešením.
   - V záhlaví prosím uveďte své jméno, příjmení a login.
   - Pak budou následovat odpovědi na jednotlivé otázky — obrázky, numerické hodnoty, komentáře.
   - U každé otázky uveďte stručný postup - může se jednat o kousek okomentovaného kódu, komentovanou rovnici nebo text. Není nutné kopírovat do protokolu celý zdrojový kód. Není nutné opisovat zadání či teorii, soustřeďte se přímo na řešení. Je-li v zadání "zobrazte", znamená to, že výsledek chceme vidět v protokolu.
   - Pokud využijete zdroje mimo standardních materiálů (přednášky, cvičení a studijní etapa projektu ISS), prosím uveďte, odkud jste čerpali (např. dokumentace numpy, Matlab, scipy, ...).
   - Protokol je možné psát v libovolném systému (Latex, MS-Word, Libre Office, ...), můžete jej psát i čitelně rukou, dolepit do něj obrázky a pak oskenovat.
   - Protokol může být česky, slovensky nebo anglicky. Budete-i psát anglicky, prosíme, abyste nazvali výsledný soubor `xlogin00_e.pdf`. Za angličtinu v protokolu není žádné zvýhodnění ani penalizace, slouží nám jen pro výběr opravujících.
2. `xlogin00.tar.gz` je komprimovaný archiv obsahující následující adresáře:
   - `/src` - Vaše zdrojové kódy – může se jednat o jeden soubor (např. `moje_reseni.py`), o více souborů či skriptů nebo o celou adresářovou strukturu.
   - `/audio` - audio soubory – ve formátu WAV, na vzorkovací frekvenci 16 kHz, bitová šířka 16 bitů, bez komprese.
3. Projekt je **samostatná práce**, proto budou Vaše zdrojové kódy křížově korelovány a v případě silné podobnosti budou vyvozeny příslušné závěry.

4. Silná korelace s kódy ze studijní etapy projektu, z Python notebooků studijní podpory a z příkladů Katky Žmolíkové[1] je v pořádku, nemusíte tedy měnit názvy proměnných, přepisovat komentáře, atd.

4 Standardní zadání

Úspěšné řešení těchto bodů zadání vede k plnému počtu bodů za projekt, tedy 18ti.

4.1 Základy – 1 bod

Načtěte vstupní signál, určete a napište jeho délku ve vzorcích a v sekundách, určete a napište jeho minimální a maximální hodnotu a zobrazte jej se slušnou časovou osou v sekundách.

4.2 Předzpracování a rámce 1 bod

Načtený signál ustředněte (odečtěte střední hodnotu) a normalizujte do dynamického rozsahu -1 až 1 dělením maximem absolutní hodnoty. Signál rozdělte na úseky (rámce) o délce 1024 vzorků s překrytím 512 vzorků, rámce uložte jako sloupce matice. Vyberte "pěkný" rámec s periodickým charakterem (znělý) a zobrazte jej se slušnou časovou osou v sekundách.

4.3 DFT – 2 body

Implementujte vlastní funkci pro výpočet diskrétní Fourierovy transformace pro N=1024 vzorků. Snažte se pracovat "vektorově", tedy s minimálním počtem cyklů. Transformace by měla být realizována jako násobení matice bází s vektorem signálu. Spusťte Vaši funkci na vybraném rámci, zobrazte modul DFT pro frekvence od 0 do $\frac{F_s}{2}$ se slušnou frekvenční osou v Hz. Porovnejte Váš výsledek s knihovní implementací FFT (např. `np.fft.fft`) - graficky a budete-li chtít, pomocí funkce na přibližné porovnání, např. `np.allclose`.

4.4 Spektrogram – 1 bod

Pro celý signál vypočtěte a zobrazte "logaritmický výkonový spektrogram" tedy obrázek s **časem** v sekundách na x-ové ose a s **frekvencí** v Hz na y-ové ose (opět do poloviny vzorkovací frekvence). Použijte opět délku okna 1024 vzorků a překrytí 512 vzorků. Hodnoty jednotlivých koeficientů DFT upravte pomocí $P[k] = 10 \log_{10} |X[k]|^2$. Můžete využít knihovní funkci, ale rádi bychom, aby časová a frekvenční osa měly správné hodnoty. Pro hodnotu koeficientu můžete dle libosti použít stupeň šedi nebo barvu.

4.5 Určení rušivých frekvencí – 2 body

Na spektrogramu budou jasně viditelné rušivé komponenty. Určete jejich frekvence f_1, f_2, f_3, f_4 v Hz. Ověřte, že jsou 4 rušivé cosinusovky harmonicky vztažené, tedy že f_2, f_3 a f_4 jsou násobky té nejnižší frekvence. Na určení frekvencí si můžete napsat funkci nebo je odečíst "ručně" ze spektrogramu či jednoho spektra.

Hint: při odečítání z jednoho spektra si dejte pozor na to, abyste rušivou frekvenci nezaměnili za součást spektra řeči.

4.6 Generování signálu – 3 body

Vygenerujte signál se směsí 4 cosinusovek na frekvencích f_1, f_2, f_3, f_4, o stejné délce jako původní signál. Uložte jej do souboru `audio/4cos.wav`. Zobrazte jeho spektrogram. Poslechem a srovnáním spektrogramů ověřte, že jste frekvence určili a signál vygenerovali správně.

4.7 Čisticí filtr – 3 body

Navrhněte filtr nebo sadu filtrů typu pásmová zádrž pro čištění signálu — musí potlačovat frekvence f_1, f_2, f_3, f_4. Můžete postupovat jednou ze tří alternativ:

1. **výroba filtru v z-rovině:** udělejte filtr tak, aby měl 4 nulové body "sedící" na jednotkové kružnici, vypočítáte je pomocí konverze frekvence v Hz na normované kruhové frekvence $\omega_k = 2\pi \frac{f_k}{F_s}$ a pak takto: $n_k = e^{j\omega_k}$. Doplňte k nim ještě 4 další komplexně sdružené nulové body (např. `np.conj`) a pak nulové body převeďte na koeficienty filtru (např. `np.poly`). Měli byste dostat FIR filtr s 9-ti koeficienty. Pokud postup

[1] `https://www.fit.vutbr.cz/~izmolikova/ISS/project/`

nespletete, bude vyrobený filtr dobře potlačovat rušení, ale také zkreslovat signál. To není důvod ke stržení bodů, ale zkuste vysvětlit, proč tomu tak je.

2. **návrh filtru ze spektra:** nejprve si "vymodelujte" požadovanou frekvenční charakteristiku filtru $H[k]$ (např. na 1024 bodech): doporučujeme pracovat pouze na 513 bodech od $H[0]$ do $H[512]$, naplnit je jedničkami a pak najít indexy koeficientů, které je třeba vynulovat. Indexy zjistíte pomocí převodu z f_1, f_2, f_3, f_4, případně se dá spektrum nebo spektrogram zobrazit s frekvenční osou s indexy koeficientů a odečíst je přímo. Pozor, pak musíte doplnit druhou polovinu spektra, tedy vzít body $H[1 \dots 511]$, otočit jejich pořadí (např. `np.fliplr`) a přilepit za body $H[0 \dots 512]$. Teoreticky byste je měli ještě i komplexně sdružit, ale jsou to jen nuly nebo jedničky, není to potřeba. Pak vyrobíte impulsní odezvu filtru pomoci inverzní FFT. Pozor, její výstup je ještě nutné přerovnat tak, aby bylo maximum impulsní odezvy uprostřed (např. `np.fftshift`), jinak to nefunguje.

3. **návrh 4 pásmových zádrží:** vyhledejte si a použijte funkce pro návrh filtrů a vyrobte 4 pásmové zádrže (band-stop filters) se závěrnými pásmy (stop-bands) okolo frekvencí f_1, f_2, f_3, f_4. Doporučujeme např. `scipy.signal.buttord` a `scipy.signal.butter` nebo `scipy.signal.ellipord` a `scipy.signal.ellip`. Při návrhu filtrů je dobré nedefinovat je úplně "ostré" (dostanete je pak šíleně dlouhé), ale s rozumnou šíří závěrného pásma, třeba 30 Hz, a s rozumnou šíří přechodů do propustého pásma, třeba 50 Hz na každé straně. Zvlnění (ripple) v propustném pásmu nastavte třeba na 3 dB a potlačení v závěrném pásmu (stop-band attenuation) třeba na -40 dB. Pozor na normování frekvencí, návrhové funkce v Pythonu a Matlabu normují frekvence pomocí Nyquistovy frekvence $\frac{F_s}{2}$ a ne pomocí vzorkovací frekvence F_s !

4. jakýkoliv jiný způsob návrhu filtru vedoucí ke kýženému výsledku je vítán.

Uveďte koeficienty filtru nebo filtrů a zobrazte jeho/jejich impulsní odezvy. Pokud jsou nekonečné, omezte je na délku vhodnou pro zobrazení.

4.8 Nulové body a póly – 2 body

Vypočtěte nulové body a póly navrženého filtru nebo filtrů a zobrazte je v komplexní rovině. Zde budou ve výhodě uživatelé Matlabu či Octave, kteří využijí funkce `zplane`. Pythonisté si ji budou muset naprogramovat (asi 5 řádků, využijte `np.roots`) nebo vygooglit již hotovou.

4.9 Frekvenční charakteristika – 2 body

Vypočtěte frekvenční charakteristiku filtru/filtrů a zobrazte ji/je se slušnou frekvenční osou v Hz. Ověřte, že filtr potlačuje rušivý signál na správných frekvencích.

4.10 Filtrace – 1 bod

Proveďte filtraci signálu pomocí Vámi navrženého filtru / filtrů. Zkontrolujte, zda je výsledný signál ve slušném dynamickém rozsahu od -1 do +1, pokud není, upravte. Uložte výsledek jako `audio/clean_aaa.wav`, kde "aaa" je zkratka použité metody návrhu filtru ("z" pro z-rovinu, "spec" pro převod filtru ze spektrální oblasti a "bandstop" pro pásmové zádrže). Ověřte poslechem, zda došlo k vyčištění signálu. Výsledek okomentujte.

5 Bonusové úkoly

Za tyto úkoly nejsou body, ale bude Vás také hřát pocit skutečného porozumění signálům. Autor/ka nejlepšího řešení dostane láhev kvalitního francouzského červeného vína.

5.1 Přesné určení frekvence

Určete frekvencí rušivých signálů s přesností na 1 Hz. Běžná FFT to nedokáže, protože 16000/1024=15.6 Hz. Můžete

- zkusit nadvzorkování spektra s interpolací pomocí kardinálního sinu (pozor, Matlab i Python ho definují jako $\sin(\pi x)/\pi x$). Matlab navíc obsahuje šikovnou funkci `interp`, která vše udělá za Vás.
- hodilo by se ale zprůměrovat spektra přes všechny rámce. Pozor, rozhodně průměrujte absolutní hodnoty, ne původní komplexní koeficienty.
- nebo můžete frekvenci určit nahrubo (FFT) a pak generovat komplexní exponenciály s krokem 1 Hz okolo té nahrubo určené a dívat se, která je nejpodobnější. Podobnost určujte na celém signálu.

5.2 Přesné určení amplitudy

Určete přesně amplitudy všech 4 cosinusovek.

- Vezměte absolutní hodnotu příslušného koeficientu, dělte počtem vzorků FFT a násobte dvěma (viz teorie jak vypočítat z DFT parametry diskrétní cosinusovky).
- opět bude asi potřeba průměrovat přes více rámců, aby se výsledek zpřesnil. Opět absolutní hodnoty, ne komplexní koeficienty.
- Spektrum řeči může určení amplitud rušivého signálu nabourat - bude lepší amplitudy určovat na začátku a na konci nahrávky, kde je ticho (řečaři tomu říkají voice activity detection – VAD – tady ji můžete udělat od oka).

5.3 Určení počátečních fází cosinusovek

To jsme doposud ignorovali, tak to zkuste. Referenční pro určení fází je nultý rámec ležící od času 0.

- ještě lepší by ale bylo fáze počítat přes všechny rámce, jenže ony se nám posouvají: pro každý další tam bude změna fáze o $-\omega_k 512$, kde ω_k je normovaná kruhová frekvence dané cosinusovky. O tyto fáze by to tedy chtělo fáze spočítané FFTčkem korigovat (tedy vždy přičíst $r\omega_k 512$), kde r je číslo rámce.
- před průměrováním je také potřeba vypočtené hodnoty zarovnat do intervalu od $-\pi$ do $+\pi$. Stackoverflow radí např. toto: `phases = (phases + np.pi) % (2 * np.pi) - np.pi`
- bude-li vypočtená fáze blízko $-\pi$ nebo $+\pi$ a mezi těmito dvěma hodnotami bude "přeblikávat" kvůli numerickým nepřesnostem, může to při průměrování způsobit problém. Zkuste s tím něco udělat.
- Pro korekci rozhodně doporučujeme přesně vypočítané hodnoty frekvence s těmi "hrubými FFTčkovými" se Vám korekce asi brzy rozjedou....

5.4 Odečet rušivého signálu

Vygenerujte podle přesných frekvencí, amplitud a fází rušivý signál (můžete srovnat s tím přibližným ze cvičení 6) a odečtěte ho od vstupního. Výsledek zobrazte v jednom obrázku s původním. Výsledek dejte do `audio/bonus.wav`, poslechněte si jej, prohlédněte a okomentujte, jak to dopadlo.