ISS Projekt 2020 / 21

Honza Brukner, Honza Švec, Katka Žmolíková a Honza Černocký, ÚPGM FIT VUT
November 27, 2020

1 Úvod

"Přes tu roušku ti není rozumět!" Věta, kterou jsme za posledních několik měsíců slyšeli každý několikrát. Pojďme se tedy podívat na roušky trochu zblízka. Cílem projektu je analyzovat vliv roušky na řeč.

Přesněji, Vaším úkolem bude nahrát jeden tón a jednu větu bez a s rouškou. Provést analýzu nahraných signálu a následně odhad parametrů Vaší roušky a použít je pro simulaci jejího vlivu na nahrávce bez roušky.

Projekt 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 být možné spustit na školním Linuxu nebo Windows a 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 knihoven `numpy` a `matplotlib` (přesněji `import numpy as np` a `from matplotlib import pyplot as plt`).

2 Odevzdání projektu

bude probíhat do informačního systému WIS ve dvou souborech:

1. `xlogin00.pdf` nebo `xlogin00_e.pdf` (kde "xlogin00" je Váš login) 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.
   - 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. Složka bude obsahovat soubory: `maskoff_tone.wav`, `maskon_tone.wav`, `maskoff_sentence.wav`, `maskon_sentence.wav` a `sim_maskon_{tone,sentence}.wav`, případně nahrávky z doplňujících úkolů.
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 je v pořádku, nemusíte tedy měnit názvy proměnných, přepisovat komentáře, atd.

3 Prvotní odhad

1. **[1 bod]** Nahrajte dvě **testovací nahrávky** — jeden a ten samý tón (například samohláska "á") bez roušky a s rouškou. Nahrávku bez roušky pojmenujte `maskoff_tone.wav` a s nasazenou rouškou `maskon_tone.wav`

   Udržet jeden specifický tón Vám může pomoct jakákoliv ladička (třeba aplikace "DaTuner" na Androidu nebo "Fine Tuner" na iOS).

   Nahrávat můžete na čemkoliv, stačí běžný smartphone nebo notebook, není nutné hledat studiový mikrofon a HiFi zvukovou kartu. Doporučujeme smartphony, mívají lepší mikrofon než notebooky. Volte běžné klidné prostředí (byt, kancelář), není nutné shánět odhlučněnou místnost. Požadovaný formát pro data je:

   - standardní WAV bez komprese,
   - vzorkovací frekvence F_s =16 kHz,
   - mono (1 kanál),
   - 16 bitů na 1 vzorek.

   Pokud Váš nahrávací software takový formát nepodporuje, můžete nahrávat na vyšší vzorkovací frekvenci a pak provést, např. pomocí běžného programu `ffmpeg`, pro běžné audio-nahrávky z Androidu např. takto: `ffmpeg -i Sa1.m4a  -ar 16000 -ac 1 -acodec pcm_s16le sa1.wav`. Nahrávku vždy zkontrolujte poslechem, a pokud v ní objevíte vážný problém (přeřeknutí, bouchnutí, atd.), prostě ji opakujte.

   Do protokolu uveďte tabulku s názvy souborů a délkou nahrávek ve vzorcích a v sekundách. Délku získáte např. pomocí programu `soxi` takto: `soxi maskoff_tone.wav`.

   Tyto nahrávky budou sloužit pro určení charakteristiky rouškového filtru.

   Kontrola: v adresáři `/audio` jsou dva WAV soubory: `maskoff_tone.wav` a `maskon_tone.wav`, s vzorkovací frekvencí F_s =16 kHz a bitovou šířkou 16 bitů, které obsahují to, co mají. Protokol obsahuje tabulku.

---

2. **[1 bod]** Nahrajte jednu a tu samou větu bez roušky a s rouškou. Nahrávku bez roušky pojmenujte `maskoff_sentence.wav` a s nasazenou rouškou `maskon_sentence.wav`.

   Na jazyku, jakým věty namluvíte nezáleží, jen se pokuste držet stejnou intonaci ve stejných částech věty, jednoduše se pokuste z akustického hlediska namluvit vše stejně.

   Do protokolu uveďte tabulku s názvy souborů a délkou nahrávek ve vzorcích a v sekundách. Délku získáte např. pomocí programu `soxi` takto: `soxi maskoff_sentence.wav` .

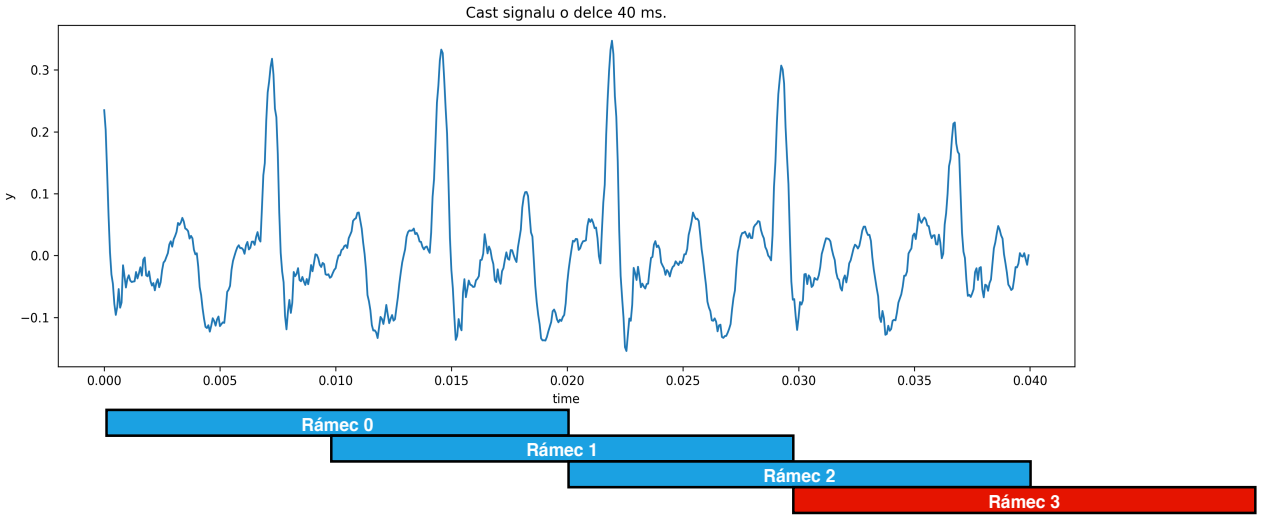
   Kontrola: v adresáři `/audio` jsou dva WAV soubory: `maskoff_sentence.wav` a `maskon_sentence.wav`, vzorkovací frekvencí F_s, bitovou šířkou, které obsahují to, co mají. Protokol obsahuje tabulku.

---

3. **[2 body]** Z každé testovací nahrávky (tónů) extrahujte část o délce 1 sekunda, ve které se budou signály co nejméně lišit. Následně signály:

   - ustředněte,
   - normalizujte do dynamického rozsahu $[-1; 1]$.

   Nyní oba extrahované úseky rozdělte na rámce. Rámec je část řečového signálu o délce typicky 20 – 25 ms. Použijte 20 ms. Jednotlivé rámce se překrývají o 10 ms. Na úseku dlouhém 1 sekundu byste tedy měli získat 100 rámců.



Z obrázku vidíme, že poslední rámec (Rámec 3) bychom získali s poloviční velikostí a to se nám nelíbí. Použijte tedy buď větší kus signálu (1.01 s) a 100 rámců, nebo původní 1 s a poslední rámec zahoďte.

Kontrola: V protokolu bude vzorec pro výpočet velikosti rámce ve vzorcích (1 bod) a graf dvou zvolených rámců, jeden z úseku bez roušky a jeden s rouškou (1 bod).

Hint:

(a) Ustřednění: $x[n] - \bar{\mathbf{x}}$, například s pomocí knihovny `numpy`: `x -= np.mean(x)`.

(b) Normalizace: $\frac{x[n]}{max|\mathbf{x}|}$, v knihovně `numpy`: `x /= np.abs(x).max()`.

Note: Úseky můžete vybrat oba ručně, nebo ručně zvolit jenom jeden, ten následně pomocí cross-korelace porovnat s celou druhou nahrávkou a vybrat úsek odpovídající největšímu koeficientu. Pokud použijete jinou než ruční metodu, krátce ji popište do protokolu.

---

4. **[4 body]** Pokud jste nahrávali testovací tóny ve stejném prostředí a se stejným zařízením, jediný rozdíl, který by mohl být mezi nahrávkami (kromě roušky), je výška tónu. Zkontrolujeme tedy, jestli se neliší, to by nám totiž mohlo dělat potíže při odhadu filtru.

   Existuje mnoho sofistikovaných metod pro získání základního tónu f_0 z řeči, nám by, ale měla stačit pouze ta základní (máme čistý signál obsahující pouze jednu samohlásku):

   (a) Nejprve aplikujeme metodu zvanou centrální klipování (Center clipping), kterou vzorky, které jsou nad 70 % maxima absolutní hodnoty převedeme na 1, pod 70 % záporného maxima absolutní hodnoty převedeme na -1 a ostatní na 0.

   (b) Provedeme autokorelaci rámce (implementujte svoji vlastní). Očekáváme velkou podobnost signálu pro:

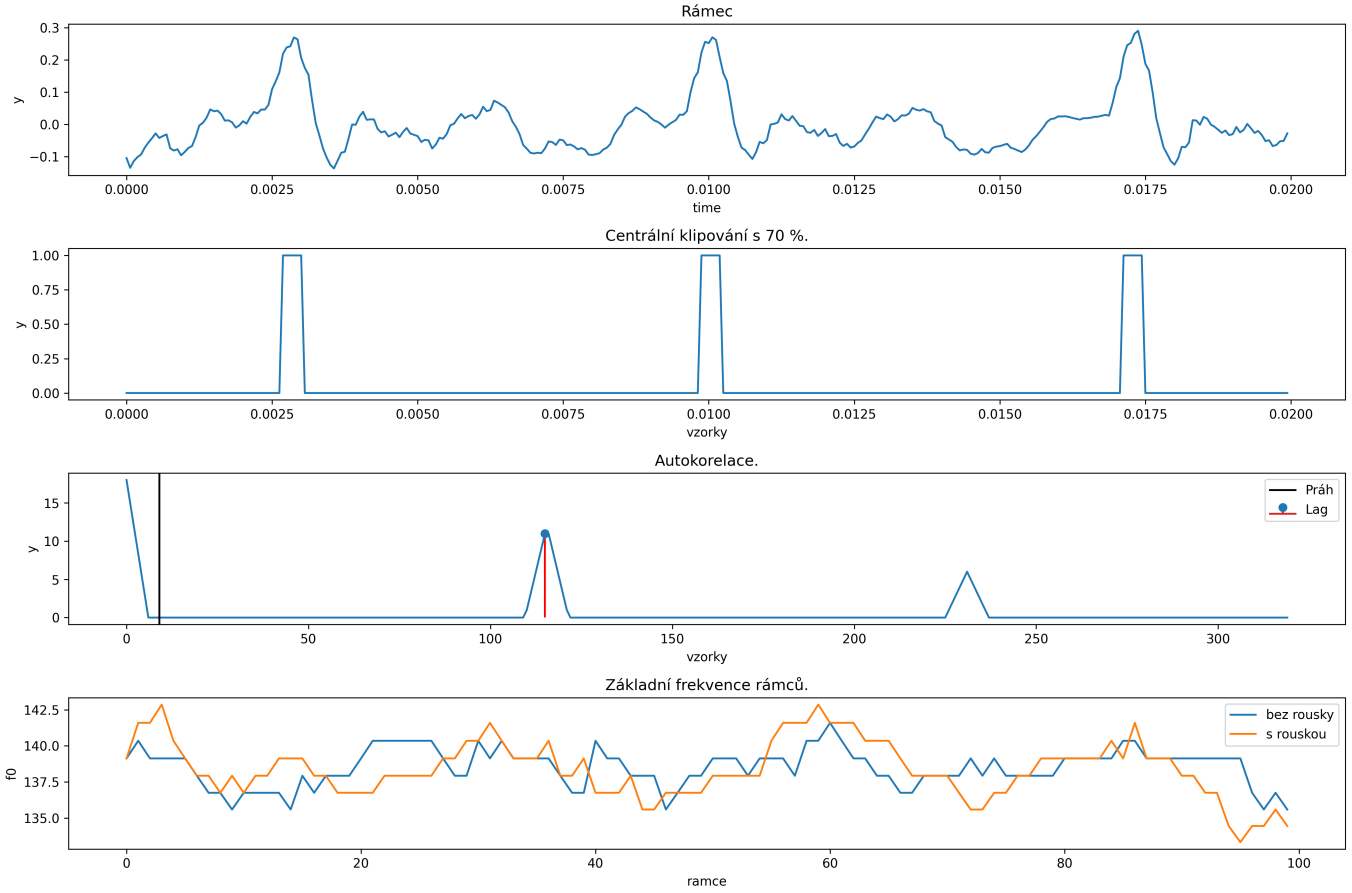
      i. koeficienty blízko nule (signál je podobný sám se sebou).

      ii. koeficienty odpovídající základní frekvenci.

      Protože ty první budou mnohem vyšší, musíme si zvolit nějaký práh (například 500 Hz), nad kterým již nebudeme tóny uvažovat.

   (c) Index maximálního koeficientu je zvaný "lag", ten musíme ještě převést na frekvenci.

   Algoritmus aplikujte na každý rámec z obou testovacích nahrávek a porovnejte, zda jsou vaše f_0 podobné. Pokud nejsou, zkuste zvolit jiné úseky nahrávek, případně je nahrajte znovu. Kontrolu můžete provést, kromě vizuálního pozorování, například pomocí průměru a rozptylu.



Kontrola: Máte vlastní implementaci korelace a v protokolu je:

(a) Graf podobný tomu v zadání (libovolný rámec, na něm aplikované centrální klipování a autokorelace) a graf srovnávající základní frekvence obou nahrávek. (2 body)

(b) Střední hodnota a rozptyl základní frekvence obou testovacích nahrávek. (1 bod)

(c) Odpověď na otázku "Jistě jste si všimli, že pokud se hodnota "lag-u" liší o 1, poměrně dost to zamává s frekvencí. Jak by se dala zmenšit velikost změny f_0 při chybě $\pm$ 1?" a zdůvodnění. (1 bod)

Hint:

(a) Hodnota prahu může být individuální.

(b) Je rozdíl mezi `np.max` a `np.argmax`.

---

5. **[3 body]** Nyní už máme dvě sady rámců s řečí, u kterých předpokládáme, že jediným rozdílem je "rouškový filtr". Podívejme se na ně ve spektrální oblasti.

(a) Spočítejte DFT spektrum z každého rámce s $N = 1024$.

(b) Implementujte vlastní funkci počítající DFT a porovnejte ji s knihovní implementací FFT (např. `np.fft.fft`).

Výsledná spektra vykreslete jako "logaritmický výkonový spektrogram" tedy obrázek s **časem** v x-ové ose a **frekvencí** v y-ové ose. Hodnoty jednotlivých koeficientů upravte pomocí následujícího vzorce.

$$P[k] = 10 \log_{10} |X[k]|^2$$

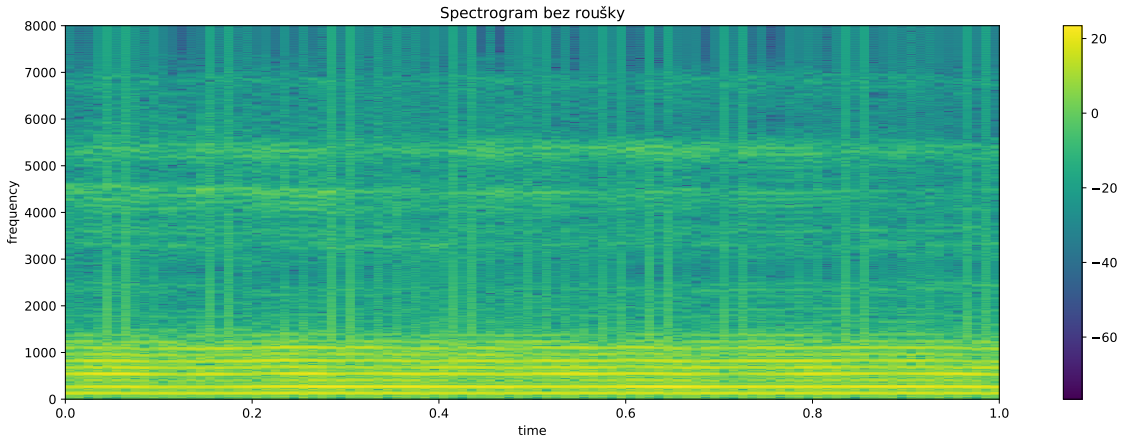
Spektrogramy vykreslete pouze pro koeficienty $k = [0..512]$

Kontrola: V protokolu je:

(a) funkce implementující DFT. (1 bod)

(b) spektrogramy pro tón s rouškou a bez roušky. (2 body)

Hint: Spektrum je možné vykreslit například pomocí `plt.imshow`, pozor ale na to, že obrázky mají (0,0) vlevo nahoře a my ho chceme mít vlevo dole.



---

6. **[3 body]** Když máme k disposici spektrogramy signálů s rouškou a bez, můžeme spočítat frekvenční charakteristiku roušky. Tu získáme (výpočtem, který záměrně není uveden) pro každý rámec, zprůměrujeme ji přes všechny rámce, abychom získali jednu frekvenční charakteristiku. Velmi pečlivě uvažte, jak budete provádět podíl a průměrování komplexních čísel. Velmi doporučujeme průměrovat pouze absolutní hodnoty.

   Vykreslete frekvenční charakteristiku filtru, opět jako výkonové spektrum.

   Kontrola: Protokol obsahuje

   (a) vztah pro výpočet $H(e^{j\omega})$. (1 bod)

   (b) frekvenční charakteristiku roušky. (1 bod)

   (c) krátký komentář k filtru. (1 bod)

---

7. **[2 body]** Samotná filtrace bude jednodušší v časové oblasti, frekvenční charakteristiku převedeme na impulsní odezvu pomocí inverzní DFT. Opět naimplementujte vlastní verzi a porovnejte s knihovní implementací (např. `np.fft.ifft`).

   Kontrola: Protokol obsahuje

   (a) vlastní implementaci IDFT. (1 bod)

   (b) graf impulsní odezvy roušky. (1 bod)

---

8. **[2 body]** Když už máte koeficienty masky, můžete jednoduše provést simulaci roušky, tedy filtraci na tónu a na namluvené větě bez roušky. Jak ji provedete, je čiště na Vás, ale můžete zkusit použít funkci `scipy.signal.lfilter`.

   Porovnejte vámi nahranou větu bez roušky, s rouškou a simulovanou rouškou. Popište jak se od sebe liší signál z rouškou a ten na kterém roušku jenom simulujeme. Jsou signály podobné? Kde se nejvíce podobají a kde liší?

   Kontrola: Protokol obsahuje

   (a) graf nahrané věty bez roušky, s rouškou a se simulovanou rouškou. (1 bod)

   (b) odpovědi na otázky. (1 bod)

   Adresář `/audio` obsahuje soubory `sim_maskon_{sentence,tone}.wav`.

   Hint: Filtr, který představuje masku je typu FIR.

---

9. **[2 body]** Závěr: zpracování signálu není jednoduché a je možné, že Vaše výsledky nebudou "nic moc" ani když budete mít vše dobře naprogramované. Krátce zhodnoťte zda Vaše řešení funguje a pokud ne, kde funguje, kde selhává.

   Kontrola: V protokolu je závěr, přečtěte si ho po sobě. (2 body)

4 Doplňující úkoly

Asi jste si všimli, že bodové hodnocení předchozích úkolů nedává součet 30 bodů, ale pouze 20. Pro získání plného počtu bodů budete muset nějak vylepšit současný postup odhadu parametrů rousky, případně zvolit odlišný přístup nebo jinak zdokonalit použité metody.

Následující metody můžete nakombinovat jak budete chtít, abyste dosáhli na požadovaný počet bodů (30 je ale pořád maximum). Student/ka s nejvyšším počtem bodů a s nejzajímavějším řešením navíc dostane láhev dobrého francouzského červeného vína.

Pokud vás napadne nějaká další úprava či vylepšení, neváhejte nás kontaktovat a bude také bodově odměněna.

10. **[3 body]** Chtěli jsme se vyhnout filtrování ve spektrální oblasti, důvodem bylo, že nemáme pouze jedno spektrum, ale celý spektrogram, ve kterém se rámce překrývají. Implementujte metodu "overlap–add", která tento problém řeší.

    Kontrola: V protokolu je implementace této metody.

    Adresář `/audio` obsahuje soubory `sim_maskon_{sentence,tone}_overlap_add.wav`.

    Hint: `https://en.wikipedia.org/wiki/Overlap%E2%80%93add_method`

---

11. **[3 body]** Při výpočtu spektra jsme nepoužili žádnou okénkovou funkci... Použijte ji.

    Kontrola: Protokol obsahuje:

    (a) záznam o vybrané okénkové funkci.

    (b) její graf (v časové i spektrální oblasti).

    (c) porovnání spektra rámce bez a s aplikací okénkové funkce (graf a krátký komentář (proč je užitečná?)).

    Adresář `/audio` obsahuje soubory `sim_maskon_{sentence,tone}_window.wav` (použijte filtrování v časové oblasti).

    Hint: Okénkovou funkci aplikujte až těsně před spektrální analýzou, při analýze f_0 by nám naopak škodila.

    `https://numpy.org/doc/stable/reference/routines.window.html`

    `https://en.wikipedia.org/wiki/Window_function`

---

12. **[3 body]** Typickou chybou při získávání základní frekvence je detekce n-násobného lagu (frekvence se tím sníží na $\frac{1}{n} f_0$). Opravte tyto chyby.

    Kontrola: Protokol obsahuje:

    (a) graf korelačních koeficientů pro rámec, ve kterém byl chybně nalezený lag.

    (b) popis, jak byla chyba odstraněna.

    Hint: `https://www.fit.vutbr.cz/study/courses/ZRE/public/opora/zre_opora.pdf` Sekce 7.6.1

    Pokud jste tuto chybu nezaznamenali, zkuste si pohrát s úrovní klipování, případně s prahem detekce, aby se projevila.

---

13. **[2 body]** Výpočet základního tónu se zdá být jako úkrok stranou, který nám pouze přidal práci... Proveďte výpočet frekvenční charakteristiky pouze z rámců, které mají opravdu stejnou základní frekvenci.

    Kontrola: V protokolu je porovnání frekvenční charakteristiky bez a s touto metodou s krátkým komentářem.

Adresář `/audio` obsahuje soubory `sim_maskon_{sentence,tone}_only_match.wav` (použijte filtrování v časové oblasti).

Hint:

Pozor na to, aby vám předchozí úloha (lagy) opravdu detekovala pouze chybu typu double lag a nedělala přílišné vyhlazení průběhu f_0.

---

14. **[5 bodů]** Úplně jinou metodou, jak odhadnout charakteristiku roušky může být odhad z energie v signálu při použití pásmové propusti. Pokud porovnáme energii v daném pásmu v signálu bez roušky a s rouškou pro každé pásmo, které dává smysl, měli bychom získat přibližně množství energie, které se ztratí použitím roušky. Poměrem těchto energií získáme odhad modulů frekvenční charakteristiky.

    Kontrola: V protokolu je porovnání modulů frekvenční charakteristiky bez a s touto metodou s krátkým komentářem.

    Hint:

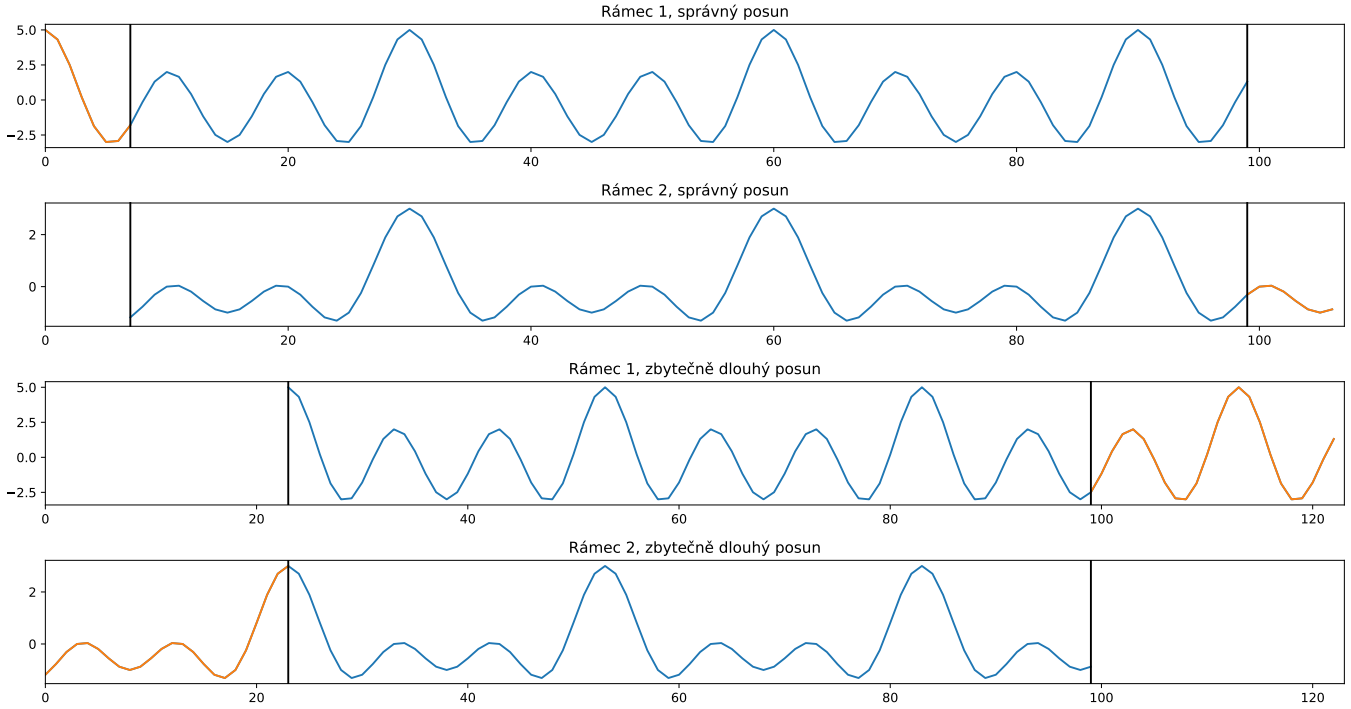
    - Studijní etapa etapa projektu sekce Práce se zvukem, filtrace a spektrální analýza, `https://www.fit.vutbr.cz/study/courses/ISS/public/#proj`, můžete použít hotovou banku filtrů.
    - `https://scipy-cookbook.readthedocs.io/items/ButterworthBandpass.html`

    Jestli je filtr stabilní ověříte například takto: `np.all(np.abs(np.roots(a))<1)`

---

15. **[5+2 bodů]** Při počítání $H(e^{j\omega})$ jsme úplně zapomněli na vliv fázového posunutí signálů! Tento problém bude trošku komplexnější, a tak si zaslouží širší rozepsání.

    Mohli bychom zkusit zarovnat signály na úrovni našeho vteřinového výřezu, ale nikdo z nás asi není tak dobrý, aby udržel dokonale periodický tón i po tak krátkou dobu. Budeme tedy muset řešit zarovnání na úrovni rámců. Když zarovnáme korespondující rámce tak, aby měly oba signály stejnou fázi, ztratíme tím nějaké vzorky, jak je ilustrováno na obrázku.



    Budeme s tím počítat a zvolíme si větší velikost rámce – 25 ms (proč právě tuto se dozvíte v **note**) a zvolenou část signálu narámcujeme jako v úkolu 3. Teď už se můžeme pustit do zarovnání, které provedeme nad korespondujícími rámci:

    (a) Nejdříve spočítáme fázový posun, výpočet bude velmi podobný získání základní frekvence. Na oba rámce použijeme klipování. Poté provedeme korelaci, tentokrát mezi rámcem bez roušky a rámcem s rouškou (`correlate(nerouška, rouška)`). Fázový posun získáme zase najitím maxima (argmax). Protože hledáme ten "správný posun" provedeme korelaci i obráceně (`correlate(rouška, nerouška)`) a získáme fázový posun na druhou stranu.

(b) Když máme získaný správný fázový posun (ten menší), můžeme se vrhnout na oříznutí rámce. Pokud vyšla fáze `correlate(nerouška, rouška)` menší ořízneme z neroušk ového rámce začátek a z rouškového konec a obráceně.

Všechny rámce jsou už pěkně zarovnané, zbývá nám vyřešit poslední problém, každý je jinak dlouhý (tím by každé spektrum mělo jinak velký dynamický rozsah). Ten vyřešíme jednoduše tak, že všechny zkrátíme na původně použitou délku 20 ms. Dál můžeme pokračovat bodem 5 zadání.

Kontrola:

- Protokol obsahuje graf znázorňující zarovnání dvou rámců (podobný jako je níže). (2 body)
- Protokol obsahuje graf fázového průběhu posunu mezi rámci. (1 bod)
- Adresář `/audio` obsahuje soubory `sim_maskon_{sentence,tone}_phase.wav` (použijte filtrování v časové oblasti). (2 body)
- (bonus) Když sečteme fázový posun na obě strany (ve vzorcích), získáme tím hodnotu, která je podobná lag-u. Vysvětlete, čím je to způsobené. (+2 body)

Note: Chceme, abychom měli i ty rámce s největším "ořezem" dlouhé alespoň 20 ms a největší "ořez" bude odpovídat fázovému posunu π (při použití "správného" posunu z obrázku). Jak spočítáme bezpečnou velikost rámce? Větší problém nám budou dělat tóny s nižší frekvencí – velkou periodou. Předpokládejme, že i nejhlubší tón nebude mít frekvenci nižší než 100 Hz tedy periodu 10 ms. S fázovým posunem π se dostaneme na maximální posunutí (a ořez) polovinu z této periody 5 ms, takže by nám měl v každém případě stačit rámec dlouhý 25 ms.