ZPRACOVÁNÍ ŘEČI V MATLABU – ÚVOD

Jan Černocký, FIT VUT Brno

MATLAB (MATrix LABoratory) – software pro vědecké výpočty a zobrazování.

Několik praktických poznámek

- budeme pracovat s Matlabem (později i s C) v Linuxu.
- Matlab spustíte z terminálu: `matlab`. Pokud chcete konzolovou verzi, která nevolá Java-mašinku (doporučuji): `matlab -nojvm`.
- v Matlabu můžete zadávat příkazy pouze, je-li kursor na příkazové řádce `>>`
- vřele doporučuji otevřít si ve vedlejším okně **textový soubor** (Emacs, vi, kate, ...), vše zapisovat do něj, a příkazy do Matlabu kopírovat pomocí copy and paste. Po ukončení cvičení Vám pak zbyde přesný záznam toho, co jste dělali, a co (snad) chodilo.
- Obsahuje-li řádek znak `%`, bere se jeho část za tímto znakem jako *poznámka* a není Matlabem prováděna.
- v Matlabu je možné se k předchozím příkazům vracet pomocí [↑]. Návrat může být i "selektivní", např. k již použitým příkazům `plot` se můžete vrátit pomocí:

  `pl` [↑] [↑] [↑] ...
- Matlab obsahuje velmi kvalitní on-line nápovědu: `help funkce`

Stručný přehled základních dovedností Matlabu

Pro toho, kdo Matlab v životě neviděl, viz 1. laboratoř ze signálů a systémů: `http://www.fit.vutbr.cz/~cernocky/sig` sekce 1–3.

1 ... než začneme s Matlabem — nahrávání zvuku

K nahrání zvuku v Linuxu použijte WaveSurfer: `ws`. Ve vedlejším terminálu je vhodné si otevřít mixér: `alsamixer`, který se ovládá šipkami a klávesou 'M' (mute). Vstup z mikrofonu bývá někdy označen jako 'Mic', někdy jako 'Capture' – "vytáhněte" raději oba dva tyto vstupy. Bude pravděpodobně nutné zapnout i zesílení mikrofonu (MicBoost) — nesmí být 'M'.

Ve `ws` označte v 'preferences', že hodláte nahrávat se vzorkovací frekvencí 8000 Hz a s kvantováním na 16 bitů. Pak otevřete nový zvukový soubor a něco nahrejte. Zkontrolujte přehrátím. Uložte ve formátu WAV a RAW (bez hlavičky).

Pokud nemáte mikrofon, můžete použít soubory `test.l16` (RAW) a `test.wav`.

2 Načtení řečového signálu do Matlabu

...je velmi jednoduché. Vzorky budou v Matlabu uloženy jako řádkový vektor. Soubory RAW (1 vzorek $\sim$ 1 short o 16 bitech) čteme podobně jako v C (všimněte si "vektorového chování" funkce `fread`).

```
ff=fopen('test.l16','r');
sig=fread(ff, [1 inf],'short')/32768;
fclose(ff)
plot(sig);
```

Všimněte si, že vzorky normalizujeme tak, aby ležely v intervalu ± 1. U souboru RAW musíme vždy předem vědět, jaký je formát vzorků a jaká je vzorkovací frekvence.

Formát WAV tyto informace obsahuje v hlavičce a čte se ještě jednodušeji:

```
sw=wavread('test.wav'); sw=sw';
[sw,Fs,Nbits] = wavread('test.wav'); sw=sw';
```

Druhé volání nám poskytne i informace z hlavičky. Všimněte si, že výsledek je nutné transponnovat, protože funkce wavread má na výstupu sloupcový vektor. Tato funkce sama normalizuje vzorky do intervalu ± 1.

Úkoly

1. Podívejte se na načtené signály: `plot(sig); plot(sw);`, vypadají stejně, že ?
2. Zkontrolujte, zda hlavička WAV obsahovala správné hodnoty vzorkovací frkvence a počtu bitů.
3. Srovnejte, zda jsou signály opravdu stejné: `ws - sig` nebo `plot(ws - wsig);`
4. Přehrejte si signál:
   `sound(ws)`
   `sound(50*ws)`
   `soundsc(50*ws)`
   Proč druhé přehrátí škrčí ? Proč třetí už zase ne ? Kdy budeme používat funkci `sound` a kdy `soundsc` ?

3 Střední hodnota a ustřednění

Důležitou operací je výpočet a zničení stejnosměrné složky v řeči (dc-offset). Jedná se o střední hodnotu signálu. Pokud máme k disposici celý signál, je výpočet a ustřednění jednoduché:

```
mean(sw)
% ustredneni
sc = sw - mean(sw);
% kontrola
mean(sc)
```

V případě, že signál zpracováváme on-line, nemůžeme čekat, až skončí. V tomto případě se počítá tzv. "klouzavý odhad" střední hodnoty, pro n-tý vzorek:

$$m[n] = km[n-1] + (1-k)s[n],$$

kde k je konstanta blízká 1.

```
k = 0.99;
N = length(sw);
sconline = zeros(1,N);
mntolook = zeros(1,N);
mn_1=0;
for ii=1:N,         % ... radeji nepouzivat 'i' jako ridici promnennou
                    % komplexni jednotka :-(
  mn = k * mn_1 + (1-k) * sw(ii);
  sconline(ii) = sw(ii) - mn;
  mntolook(ii) = mn;
  mn_1 = mn;
end
```

Cykly jsou v Matlabu velmi pomalé a nepraktické, ale jinak to v tomto případě nejde...

Úkoly

1. Srovnejte původní a ustředněný signál: `plot (1:N,sw,1:N,sconline)`
2. Podívejte se na průběh odhadnuté střední hodnoty: `plot (1:N,mntolook);`
3. ... a srovnejte ji s předchozím odhadem: `mean(sconline)`
4. Proč hodnoty on-line odhadu kmitají okolo správné hodnoty ? Jak byste tyto kmity utlumili ? Jakou by to mělo nevýhodu ?

4 Výběr a uložení signálu

Výběr signálu uskutečníme tak, že zadáme hraniční vzorky, např: `x = sw(1000:2000);`
Do formátu WAV ukládáme: `wavwrite(x,8000,16,'vyrez.wav');`
Do formátu RAW:

```
ff=fopen('vyrez.l16','w');
fwrite(ff, round(x * 32768), 'short');
fclose(ff)
```

Všimněte si nutnosti de-normalizace, ukládáme celá čísla — `short`'s.

Úkoly

1. Vyberte znělý úsek, poslechněte si jej pomocí `sound(x)`. Pak opakujte s neznělým a s přechodovým úsekem (plozíva). Poznáte z krátkého úseku, o kterou hlásku se jedná ?
2. Uložte vybraný úsek ve WAV a RAW a pokuste se načíst do WaveSurferu. Vše v pořádku ?

5 Dělení signálu na rámce

Dozvíme se, že díky tzv. kvazistacionaritě řeči nemůžeme téměř nic dělat se signálem jako celkem, ale musíme dělit na kratší úseky — tzv. **rámce** (frames). Typická délka je 20 ms a překrytí 10 ms. Výpočet délky rámců ve vzorcích:

```
l = 20e-3; r = 10e-3;
ls = round (l * Fs)
rs = round (r * Fs)
```

A dělení na rámce, které budou ve sloupcích matice `swram`:

```
% frame shift
fs = ls - rs;
%Pocet ramcu - bez odvozeni (ale jde na to prijit) - bude na prednesu
Nram = 1 + floor((length(sw) - ls) / fs)
% alokujeme pro ramce - budou ve sloupcich
swram = zeros(ls, Nram);
% a naplnime:
odtud=1; potud=ls;
for ii=1:Nram,
  ramec = sw(odtud:potud)';
  swram(:,ii) = ramec;
  odtud = odtud + fs;
  potud = potud + fs;
end
```

Úkoly

1. Podívejte se na znělý a neznělý rámec (např pro test.l16 13-tý a 100-tý): `plot (swram (:,13)); plot (swram (:,100))`
2. Poslechněte si tyto rámce — stačí Vám 20 ms na rozpoznání hlásky ?
3. Uzavřete dělení na rámce do funkce — budeme ji potřebovat. Umístíte-li tuto funkci do svého `$HOME/matlab`, "uvidí" na ni Matlab odkudkoliv.

6 Spektrogram

je oblíbenou pomůckou pro analýzu řeči. Zatím bez teorie, pouze srovnání se signálem:

```
subplot(211);
plot(sw); axis tight;
subplot(212);
specgram(sw,256,Fs);
```

Úkoly

1. Proč jsou použita nastavení os `axis tight` ?
2. Jakým zvukům odpovídá vysoká/nízká energie na vyšších/nižších kmitočtech ?