Laboratorul 07.

SNR, decibeli și DFT

În acest laborator vom rezolva câteva exerciții legate de calcularea raportului semnal-zgomot(eng. signal-to-noise ratio - SNR), vom exprima acest raport în decibeli și vom căpăta experiență cu transformata Fourier discretă (DFT). Deși nu am discutat prea mult despre ea, în toate exercițiile care folosesc DFT vom utiliza funcția fft (Fast Fourier Transform) și inversa acesteia ifft.

Materiale utile:

- D. Johnson's book [http://www.ece.rice.edu/~dhj/courses/elec241/col10040.pdf], secțiunile 5.4-5.7

În exercițiile din acest laborator vom considera că DFT are același număr de componente ca și semnalul de intrare. Din curs știm formulele pentru DFT:

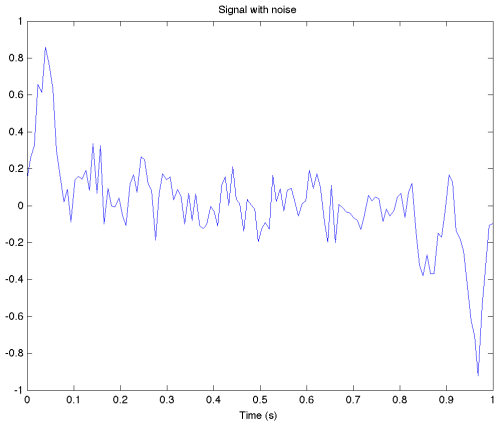
$$DFT: S(k) = \sum_{n=0}^{N-1} s(n) e^{\frac{-j2\pi nk}{K}}, k \in \{0, \ldots, K-1\}$$

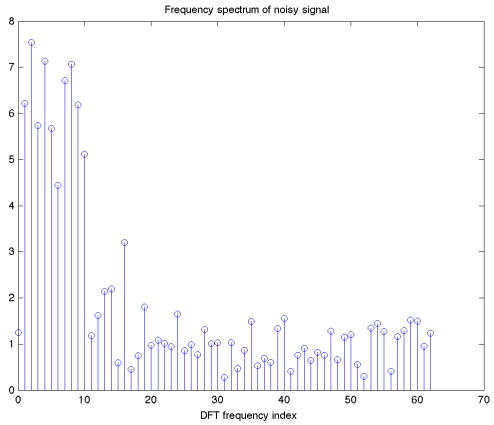
$$IDFT\ (inversa\ DFT): s(n) = \frac{1}{K} \sum_{k=0}^{K-1} S(k) e^{\frac{j2\pi nk}{K}}, n \in \{0, \ldots, N-1\}$$

Exercițiul 1 -- unde cu zgomot [6p]

În acest exercițiu aveți dat un semnal cu zgomot, (click aici), pentru care știți că informația de interes se află în primele 10 componente DFT, corespunzătoare, de fapt cu $k \in \{-9, \ldots, 9\}$, adică componenta continuă / directă (DC: $k = 0$) și componentele frecvențelor pozitive $k = 1, \ldots, 9$ și negative $k = -1, \ldots, -9$.

Mai jos puteți vedea semnalul cu zgomot precum și spectrul pozitiv(folosind doar $k = \{0, 1, \ldots, N/2 - 1\}$).





Vrem să calculam SNR în domeniul frecvență și apoi să eliminăm zgomotul din semnal folosind DFT și păstrând doar frecvențele de interes. Pentru aceasta va trebui să urmați pașii următori:

1. Încărcați semnalul zgomotos și plotați-l. Folosiți acest cod ca să încărcați semnalul:

```
load('noisy_signal.mat')
```

Acest semnal are $N = 128$ eșantioane și puteți considera că frecvența de eșantionare este $f_s = N = 128$ Hz.

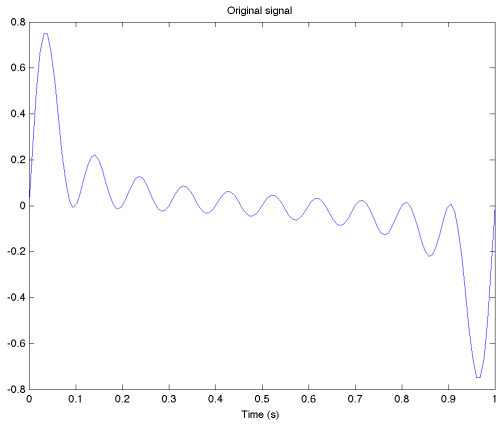
2. Calculați DFT pentru semnalul zgomotos folosind formula de mai sus, apoi comparați rezultatul cu cel al funcției fft din MATLAB. Plotați spectrul, atât în forma dată de fft, cât și în forma centrată în zero, formată aplicând funcția *fftshift* pe rezultatul de la fft. Ne amintim că funcția fft conține coeficienții pentru frecvențele pozitive între 0 și $N/2 - 1$, iar următorii coeficienți aparțin de fapt spectrului negativ, de la $-N/2$ la -1.

3. Calculați raportul semnal-zgomot în domeniul frecvență, ca raportul dintre puterea semnalului și puterea zgomotului. Pentru aceasta vom considera că semnalul util se află doar în componentele de frecvență $k \in \{-9, \ldots, 9\}$ pe când zgomotul se află în toate componentele de frecvență. Formula pentru calcularea puterii unui semnal este: $power = \frac{1}{N} \sum_{k=0}^{N-1} |S(k)|^2$, unde N este numărul total de componente considerate în semnal.

4. Care este echivalentul acestui SNR în decibeli(dB)?

5. Din ieșirea funcției fft eliminați frecvențele din afara zonei de interes (faceți-le zero). Aveți grijă la indicii frecvențelor pozitive și negative și luați în considerare că primul element corespunde componentei directe, care nu are corespondent negativ.

6. Pentru a reconstrui semnalul corespunzător acestui spectru, care are componente diferite de zero doar pentru frecvențele de interes, calculați transformata Fourier discretă inversă (IDFT) folosind formula de mai sus, iar apoi comparați rezultatul cu cel al funcției *ifft* din MATLAB. Plotați acest semnal și comparați-l cu originalul zgomotos. Astfel, practic ați implementat un tip foarte simplu de filtru trece-jos.



Aici aveți o imagine cu semnalul fără zgomot.

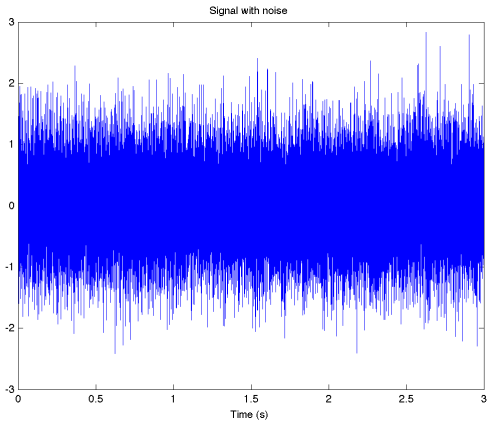
Arată precum semnalul vostru reconstruit?

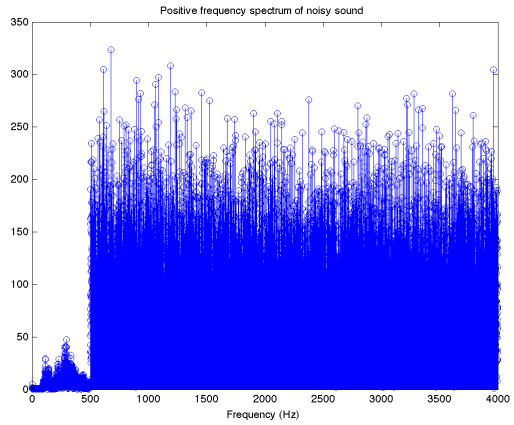
Pentru a putea reface perfect semnalul discret original avem nevoie să calculam un spectru de lungime egală cu semnalul, adică K trebuie să fie egal cu N

Exercițiul 2 -- sunet cu zgomot [4p]

Acest exercițiu este similar cu cel precedent. Acum aveți dat o înregistrare a unei secvențe vorbite (click aici) care a fost amestecată cu zgomot de frecvență înaltă. Trebuie sa aplicați pașii precedenți pentru a elimina zgomotul și pentru a determina secvența vorbită.

Aici aveți ploturile sunetului cu zgomot precum și spectrul său pozitiv:





Pentru acest sunet știm că semnalul de interes se află în intervalul 0-500 Hz și că sunetul a fost înregistrat la o frecvență de eșantionare de $f_s = 8000$ Hz. Observați că trebuie să convertiți indecșii DFT (adică $k = 0, 1, \ldots, N-1$) în frecvențe știind că fiecare element DFT corespunde unei frecvențe $\frac{k \cdot f_s}{N}$. Folosind această informație ar trebui să determinați ce frecvențe să păstrați, astfel încât să filtrați zgomotul, precum în exercițiul anterior.

Calculați SNR și filtrați zgomotul, asemănător ca la exercițiul anterior. Pentru calculul puterii zgomotului trebuie să luați în considerare că zgomotul nu se mai află în toate componentele de frecvență ca la exercițiul anterior.

Pentru a încărca sunetul și frecvența de eșantionare folosiți următorul cod:

```
S = load('noisy_sound.mat');
s = S.noisy_sound;
fs = S.fs;
```

Pentru a asculta sunetul stocat în vectorul *s*, folosiți codul următor:

```
sound(s)
```

Ascultați sunetul înainte și după eliminarea zgomotului. Dacă ați eliminat zgomotul într-un mod corect, atunci veți putea determina secvența vorbită.

Atenție la redarea semnalului înaintea procesării pentru că zgomotul înalt poate fi foarte neplăcut. Așa că setați volumul foarte jos pentru această etapă. După filtrare puteți mări volumul pentru a auzi secvența vorbită.

În acest exercițiu folosiți pentru calculul spectrului funcția fft, respectiv ifft pentru reconstruirea semnalului.

Exercițiul 3 -- transformata fourier discretă pe imagini [Bonus 1p]

Până acum am descompus semnale 1D în serii de sinusoide cu transformata Fourier. Însă putem aplica același procedeu și pentru semnale 2D, descompunându-le în sinusoide 2D. Putem să ne gândim la o sinusoidă 2D ca la un val. Cel mai comun semnal 2D este o imagine cu nivele de gri.

Putem aplica transformata Fourier discretă pe imagini cu funcția fft2 din MATLAB. în acest exercițiu veți face următoarele:

1. încărcați o imagine folosind funcția imread din MATLAB: ex: img = imread('peppers.png')
2. convertiți imaginea RGB în imagine gri și normalizați: img = double(rgb2gray(img))/255
3. aplicați fft2 pe imagine și obțineți un spectru 2D (o matrice) S(k)
4. tăiați frecvențele joase din spectru obținând un nou spectru S1(k). Frecvențele joase 2D sunt cele de frecvență mică pe ambele axe, adică, pe rezultatul dat de fft sunt în colturile matricii.

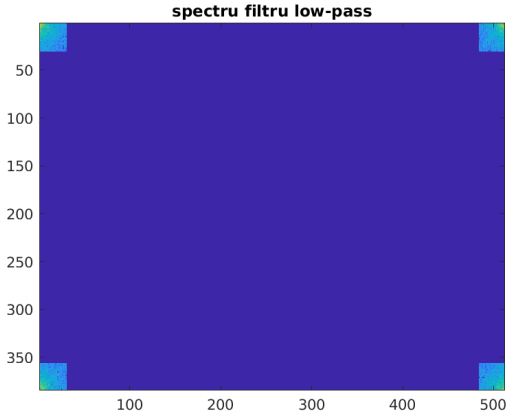
5. tăiați frecvențele înalte din spectru obținând un nou spectru S2(k). Frecvențele înalte sunt formate din tot spectrul (toată matricea) mai puțin colțurile.
6. obțineți din fiecare spectru o imagine: reconstructed_img = ifft2(S1)
7. afișați cele 2 imagini cu funcția imshow()

Pentru a utiliza funcția **rgb2gray** în Octave trebuie să instalați pachetul Octave **image** din Octave command prompt.

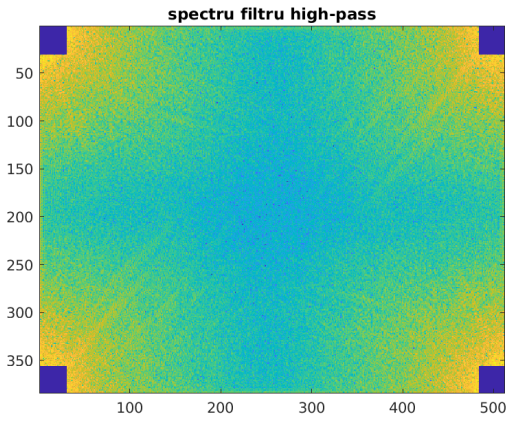
```
pkg install -forge image
pkg load image
```

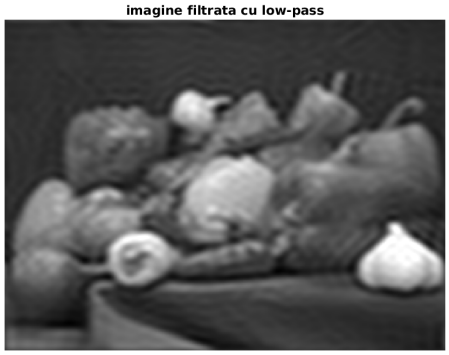
Folosiți această imagine:

Rezultatele ar trebui să arate în felul următor:

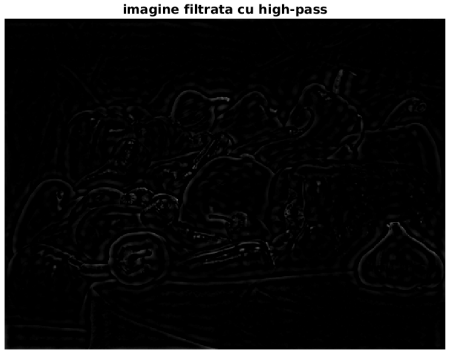


Pixel info: (X, Y) Pixel Value





imagine filtrata cu high-pass



ps/labs/07.txt · Last modified: 2020/11/25 11:45 by ionut.gorgos