TP n°2:
Calcul numérique avec Scilab

Nous allons utiliser le logiciel SciLab, un clône open-source de MATLAB. Il peut être téléchargé ici:
http://www.scilab.org/fr/content/download/849/7897/file/Scilab_debutants.pdf [fr]
http://www.scilab.org/en/download/6.0.1 [en]
Pour ceux qui ne connaissent ni MATLAB ni SciLab il vaut la peine de regarder cette introduction:
https://wiki.scilab.org/Tutorials?action=AttachFile&do=view&target=Scilab_beginners.pdf
Une liste extensive de documentation et tutoriels pour SciLab:
https://wiki.scilab.org/Tutorials

Exercice 1: Transformée de Fourier Discrète avec FFT

Le but de cet Exercie est de prendre en main les fonctions `fft` et `ifft` de Scilab.

- Choisir le nombre d'échantillons `N`. Une puissance entière de 2 est approprié.
- Créer `n`, un vecteur contenant les numéros d'échantillon.
- Créer `k`, un vecteur contenant les numéros d'échantillon dans le spectre.
- Créer `fdelta`, un vecteur dont les valeurs suivent une fonction delta discret.
- Utilisant `fft`, calculer `fdelta_fft`, la transformé de Fourier discrète de `fdelta`.
- Utilisant plot, visualiser `fdelta` et `fdelta_fft`; vérifier que l'allure de `fdelta_fft` est celle attendue.
- Utilisant `ifft`, vérifier qu'on retrouve `fdelta` en prenant la transforme inverse de `fdelta_fft`.

```
N = 1024;
n = -N/2:N/2-1;
k = -N/2:N/2-1;
n0 = 0;
fdelta = zeros(1,N); fdelta(find(n==n0)) = N;
fdelta_fft = fftshift(fft(fftshift(fdelta)));
figure(1);
subplot(2,1,1), plot(k, abs(fdelta_fft), 'k.-');
subplot(2,1,2), plot(k, phasemag(fdelta_fft,'m'), 'k.-');
fdelta_test = ifftshift(ifft(ifftshift(fdelta_fft)));
sum(abs(fdelta_test - fdelta))
```

Alternatif:

```
N = 1024;
n = [0:N/2-1 -N/2:-1];
k = [0:N/2-1 -N/2:-1];
n0 = 0;
fdelta = zeros(1,N); fdelta(find(n==n0)) = N;
fdelta_fft = fft(fdelta);
figure(2);
subplot(2,1,1), plot(k, abs(fdelta_fft), 'k.-');
subplot(2,1,2), plot(k, phasemag(fdelta_fft,'m'), 'k.-');
fdelta_test = ifft(fdelta_fft);
sum(abs(fdelta_test - fdelta))
```

Exercice 2: Propriété de translation de la FFT

Le but de cet Exercice est de montrer comment utiliser la propriété de translation de la TF avec la `fft`.

- Créer `N`, `n`, `k`.
- Créer `fdelta`, un vecteur dont les valeurs suivent une fonction delta discret.
- Réaliser une translation de `nshift` = 100 par deux approches différentes.
- Tracer les deux résultats et vérifier qu'ils concordent.

```
N = 1024;
n = -N/2:N/2-1;
k = -N/2:N/2-1;
n0 = 0;
fdelta = zeros(1,N); fdelta(find(n==n0)) = N;
nshift = 100;
//nshift = 0.5;

fdelta_shift_1 = [fdelta(N-nshift:N) fdelta(1:N-nshift-1)];
fdelta_fft = fftshift(fft(fftshift(fdelta)));
fdelta_shift_2_fft = fdelta_fft.*exp(-%i*2*%pi*k*nshift/N);
fdelta_shift_2 = ifftshift(ifft(ifftshift(fdelta_shift_2_fft)));
figure(3);
subplot(2,1,1), plot(k, abs(fdelta_shift_1), 'k.-');
subplot(2,1,2), plot(k, abs(fdelta_shift_2), 'k.-');
```

Exercice 3: Rélation entre FFT et TF continu

Le but de cet Exercice est de montrer la rélation entre FFT, TFD, et TF.

- Donner l'expression analytique **f(t)** d'une Gaussienne avec largeur à mi-hauteur `N/8`.
- Donner l'expression analytique **F(t)** de sa TF.
- Créer l'axe temporelle `t`, avec un pas d'échantiollonnage `Delta` approprié.
- Créer `fgauss`, un vecteur dont les valeurs suivent **f(t)**.
- Créer `fgauss_TF`, un vecteur dont les valeurs suivent **F(ν)**.
- Utilisant `fft`, calculer `fgauss_fft`, la transformé de Fourier discrète de `fgauss`.
- Utilisant `plot`, visualiser `fgauss` et `fgauss_fft`; vérifier que la rélation entre leur largeurs à mi-hauteur est celle attendue.
- Utilisant `plot`, vérifier la corréspondance entre `fgauss_fft` et `fgauss_TF`.
- Vérifier que Plancheval-Parserel est satisfait pour `fgauss_fft` ainsi que `fgauss_TF`.
- Réaliser une convolution `fconv_1` de `fgauss` avec elle-même par une approche directe.
- Réaliser une convolution `fconv_2` de `fgauss` avec elle-même utilisant la propriété de convolution de la `fft`.
- Tracer `fconv_1` et `fconv_2` afin de montrer la corréspondance.

```
//f(t) = exp(-4*log(2)*(t).^2/FWHM^2); F(nu) = sqrt(%pi)*(FWHM/
(2*sqrt(log(2))))*exp(-%pi^2*(FWHM/(2*sqrt(log(2))))^2*nu.^2);
N = 1024;
n = -N/2:N/2-1;
Delta = 1e-3;
t = Delta*n;
k = -N/2:N/2-1;
nu = 1/(N*Delta)*k;
FWHM = 0.01;
a = 2*log(2)/FWHM^2;
fgauss = exp(-a*t.^2);
fgauss_fft = Delta*fftshift(fft(fftshift(fgauss)));
fgauss_TF_analytique = sqrt(%pi/a)*exp(-%pi^2*nu.^2/a);
figure(4);
plot(nu,fgauss_TF_analytique, 'k');
plot(nu,abs(fgauss_fft), 'kx');
[sum(abs(fgauss).^2)*Delta sum(abs(fgauss_fft).^2)*1/Delta/N

tic;
fconv_1 = zeros(1,N);
for n = -N/2:N/2-1
    n
    for l = -N/2:N/2-1
        fconv_1(n+N/2+1) = fconv_1(n+N/2+1) +
Delta*fgauss(l+N/2+1).*fgauss(modulo(N+(n-l+N/2),N)+1);
    end
end
toc
tic;
fconv_2_fft = fgauss_fft.*fgauss_fft;
fconv_2 = 1/Delta*ifftshift(ifft(ifftshift(fconv_2_fft)));
toc
figure(5); plot(t,fconv_1); plot(t,fconv_2, 'kx')
```

Exercice 4: Comprendre l'oscilloscope numérique

Le but de cet Exercice est de comprendre le foncionnement de l'oscilloscope numérique, tel celui vu dans le TP n° 1.

- Créer `Wrect` représentant une fénêtre rectangulaire; calculer sa FFT `Wrect_fft`.
- Créer `WHann` représentant une fénêtre Hanning; calculer sa FFT `WHann_fft`.
- Créer `WFlat` représentant une fénêtre Flat-top; calculer sa FFT `WFlat_fft`.
- Créer `Wexp` représentant une fénêtre exponentielle; calculer sa FFT `Wexp_fft`.
- Tracer afin de comparer les allures temporelles et spectrales.

```
N = 1024;
n = -N/2:N/2-1;
k = -N/2:N/2-1;
Wrect = ones(1,N); //Rectangulaire (aucune fenetrage)
Wrect_fft =fftshift(fft(fftshift(Wrect)));
WHann = 1/2 - 1/2*cos(2*%pi*n/(N-1)); //Hanning (raised cosine)
WHann_fft = fftshift(fft(fftshift(WHann)));
a0 = 1; a1 = 1.93; a2 = 1.29; a3 = 0.388; a4 = 0.028; //Flattop
WFlat = a0 - a1*cos(2*%pi*n/(N-1)) + a2*cos(4*%pi*n/(N-1)) - a3*cos(6*%pi*n/(N-1))
+ a4*cos(8*%pi*n/(N-1));
WFlat_fft = fftshift(fft(fftshift(WFlat)));
tau = N/4; //Exponentielle
Wexp = exp(-abs(n)/tau);
Wexp_fft = fftshift(fft(fftshift(Wexp)));
[sum(abs(Wexp).^2) sum(abs(Wexp_fft).^2)/N]
figure(6);
subplot(4,2,1), plot(n,Wrect, 'k.-');
subplot(4,2,2), plot(k,abs(Wrect_fft), 'k.-');
a = gca(); a.data_bounds = [-10,0;10,1024];
subplot(4,2,3), plot(n,WHann, 'k.-');
subplot(4,2,4), plot(k,abs(WHann_fft), 'k.-');
a = gca(); a.data_bounds = [-10,0;10,1024];
subplot(4,2,5), plot(n,WFlat, 'k.-');
subplot(4,2,6), plot(k,abs(WFlat_fft), 'k.-');
a = gca(); a.data_bounds = [-10,0;10,1024];
subplot(4,2,7), plot(n,Wexp, 'k.-');
subplot(4,2,8), plot(k,abs(Wexp_fft), 'k.-');
a = gca(); a.data_bounds = [-10,0;10,1024];
```

- Créer `S1` représentant un signal sinusoïdal échantillonné dont la fréquence égale un multiple entier de `1/(N * Delta)`.
- Créer `S2` représentant un signal sinusoïdal échantillonné dont la fréquence n'est pas égale un multiple entier de `1/(N * Delta)`.
- Tracer `S1` fénêtré par les quatre différentes fénêtres, et leur FFT.
- Tracer `S2` fénêtré par les quatre différentes fénêtres, et leur FFT.
- Investiguer comment le choix de fénêtre influe sur le spectre dans les cas de `S1` et `S2`.

```
k1 = 100;
S1 = cos(2*%pi*k1*n/N);
k2 = 100.5;
S2 = cos(2*%pi*k2*n/N);
figure(7);
subplot(4,4,1), plot(n,S1, 'k.-');
subplot(4,4,2), plot(n,S1.*Wrect, 'k.-');
subplot(4,4,3), plot(k, abs(fftshift(fft(fftshift(S1.*Wrect)))), 'k.-');
a = gca(); a.data_bounds = [k1-10,0;k1+10,1024];
subplot(4,4,4), plot(k, phasemag(fftshift(fft(fftshift(S1.*Wrect))),'m'), 'k.-');
a = gca(); a.data_bounds = [k1-10,-360;k1+10,360];
subplot(4,4,5), plot(n,S1, 'k.-');
subplot(4,4,6), plot(n,S1.*WHann, 'k.-');
subplot(4,4,7), plot(k, abs(fftshift(fft(fftshift(S1.*WHann)))), 'k.-');
a = gca(); a.data_bounds = [k1-10,0;k1+10,1024];
subplot(4,4,8), plot(k, phasemag(fftshift(fft(fftshift(S1.*WHann))),'m'), 'k.-');
a = gca(); a.data_bounds = [k1-10,-360;k1+10,360];
subplot(4,4,9), plot(n,S1, 'k.-');
subplot(4,4,10), plot(n,S1.*WFlat, 'k.-');
subplot(4,4,11), plot(k, abs(fftshift(fft(fftshift(S1.*WFlat)))), 'k.-');
a = gca(); a.data_bounds = [k1-10,0;k1+10,1024];
subplot(4,4,12), plot(k, phasemag(fftshift(fft(fftshift(S1.*WFlat))),'m'), 'k.-');
a = gca(); a.data_bounds = [k1-10,-360;k1+10,360];
subplot(4,4,13), plot(n,S1, 'k.-');
subplot(4,4,14), plot(n,S1.*Wexp, 'k.-');
subplot(4,4,15), plot(k, abs(fftshift(fft(fftshift(S1.*Wexp)))), 'k.-');
a = gca(); a.data_bounds = [k1-10,0;k1+10,1024];
subplot(4,4,16), plot(k, phasemag(fftshift(fft(fftshift(S1.*Wexp))),'m'), 'k.-');
a = gca(); a.data_bounds = [k1-10,-360;k1+10,360];
figure(8);
subplot(4,4,1), plot(n,S2, 'k.-');
subplot(4,4,2), plot(n,S2.*Wrect, 'k.-');
subplot(4,4,3), plot(k, abs(fftshift(fft(fftshift(S2.*Wrect)))), 'k.-');
a = gca(); a.data_bounds = [k2-10,0;k2+10,1024];
subplot(4,4,4), plot(k, phasemag(fftshift(fft(fftshift(S2.*Wrect))),'m'), 'k.-');
a = gca(); a.data_bounds = [k2-10,-360;k2+10,360];
subplot(4,4,5), plot(n,S2, 'k.-');
subplot(4,4,6), plot(n,S2.*WHann, 'k.-');
subplot(4,4,7), plot(k, abs(fftshift(fft(fftshift(S2.*WHann)))), 'k.-');
a = gca(); a.data_bounds = [k2-10,0;k2+10,1024];
subplot(4,4,8), plot(k, phasemag(fftshift(fft(fftshift(S2.*WHann))),'m'), 'k.-');
a = gca(); a.data_bounds = [k2-10,-360;k2+10,360];
subplot(4,4,9), plot(n,S2, 'k.-');
```

```
subplot(4,4,10), plot(n,S2.*WFlat, 'k.-');
subplot(4,4,11), plot(k, abs(fftshift(fft(fftshift(S2.*WFlat)))), 'k.-');
a = gca(); a.data_bounds = [k2-10,0;k2+10,1024];
subplot(4,4,12), plot(k, phasemag(fftshift(fft(fftshift(S2.*WFlat))),'m'), 'k.-');
a = gca(); a.data_bounds = [k2-10,-360;k2+10,360];
subplot(4,4,13), plot(n,S2, 'k.-');
subplot(4,4,14), plot(n,S2.*Wexp, 'k.-');
subplot(4,4,15), plot(k, abs(fftshift(fft(fftshift(S2.*Wexp)))), 'k.-');
a = gca(); a.data_bounds = [k2-10,0;k2+10,1024];
subplot(4,4,16), plot(k, phasemag(fftshift(fft(fftshift(S2.*Wexp))),'m'), 'k.-');
a = gca(); a.data_bounds = [k2-10,-360;k2+10,360];
```

- Pour les quatre fénêtres, mettre en evidence (i) l'allure du spectre ; (ii) l'amplitude maximal du spectre; (iii) l'énergie dans le spectre lorsque la fréquence du signal échantillonné change de `k1/(N * Delta)` à `(k1+1)/(N * Delta)`.

```
counter = 0
for k1 = 100:0.1:101;
    figure(8);
    counter = counter + 1;
    S = cos(2*%pi*k1*n/N);
    ES(counter) = sum(abs(S).^2);
    Srect_fft = fftshift(fft(fftshift(S.*Wrect)));
    clf; plot(k,abs(Srect_fft), 'k.-'); a = gca(); a.data_bounds = [k1(1)-
5,0;k1(1)+5,512];sleep(500);
    maxamprect(counter) = max(abs(Srect_fft));
    Erect(counter) = 1/N*sum(abs(Srect_fft).^2);
    SHann_fft = fftshift(fft(fftshift(S.*WHann)));
//    clf; plot(k,abs(SHann_fft), 'k.-'); a = gca(); a.data_bounds = [k1(1)-
5,0;k1(1)+5,512];sleep(500);
    maxampHann(counter) = max(abs(SHann_fft));
    EHann(counter) = 1/N*sum(abs(SHann_fft).^2);
    SFlat_fft = fftshift(fft(fftshift(S.*WFlat)));
//    clf; plot(k,abs(SFlat_fft), 'k.-'); a = gca(); a.data_bounds = [k1(1)-
5,0;k1(1)+5,512];sleep(500);
    maxampFlat(counter) = max(abs(SFlat_fft));
    EFlat(counter) = 1/N*sum(abs(SFlat_fft).^2);
    Sexp_fft = fftshift(fft(fftshift(S.*Wexp)));
//    clf; plot(k,abs(Sexp_fft), 'k.-'); a = gca(); a.data_bounds = [k1(1)-
5,0;k1(1)+5,512];sleep(500);
    maxampexp(counter) = max(abs(Sexp_fft));
    Eexp(counter) = 1/N*sum(abs(Sexp_fft).^2);
end
figure(9);
subplot(4,2,1), plot(maxamprect);
subplot(4,2,2), plot(Erect./ES);
subplot(4,2,3), plot(maxampHann);
subplot(4,2,4), plot(EHann./ES);
subplot(4,2,5), plot(maxampFlat);
subplot(4,2,6), plot(EFlat./ES);
subplot(4,2,7), plot(maxampexp);
subplot(4,2,8), plot(Eexp./ES);
```

Exercice 5: Transformée de Fourier Discrète bi-dimensionelle avec FFT

- Choisir le nombre d'échantillons selon le premier axe `N`. Une puissance entière de 2 est approprié.
- Choisir le nombre d'échantillons selon le second axe `M`. Une puissance entière de 2 est approprié.
- Créer `n`, un vecteur contenant les numéros d'échantillon selon la première direction.
- Créer `m`, un vecteur contenant les numéros d'échantillon selon la seconde direction.
- Créer `k`, un vecteur contenant les numéros d'échantillon dans le spectre selon la première direction.
- Créer `l`, un vecteur contenant les numéros d'échantillon dans le spectre selon la seconde direction.
- Créer `fbox`, une matrice dont les valeurs suivent une fonction créneau.
- Utilisant `fft`, calculer `fbox_fft`, la transformé de Fourier 2D discrète de `fbox`.
- Utilisant `Matplot`, visualiser `fbox` et `fbox_fft`; vérifier que l'allure de `fbox_fft` est celle attendue.

```
N = 64;
M = 64;
n = -N/2:N/2-1;
m = -M/2:M/2-1;
k = -N/2:N/2-1;
l = -M/2:M/2-1;
n0 = 0;
m0 = 0;
fbox = zeros(N,M); fbox(abs(n)<=N/8,abs(m)<=N/8) = 1;
fbox_fft = fftshift(fft(fftshift(fbox)));
f = scf(10);
Matplot(round(63*fbox/max(fbox)) + 1);
f.color_map = graycolormap(64);
f = scf(11);
subplot(2,1,1),
Matplot(round(63*abs(fbox_fft)/max(abs(fbox_fft))) + 1);
subplot(2,1,2),
Matplot(63/360*phasemag(fbox_fft,'m')+1);
f.color_map = graycolormap(64);
```

Exercice 6: Simple illusion optique – superposition d'image à basses fréquences spatiales avec une image à hautes fréquences spatiales

Le but de cet Exercice est de donner une simple exemple du traitement d'image qui peut être fait grâce à la `fft`.

- Créer `filter1`, un filtre numérique 2D passe-bas avec fréquence de coupure ca. 10 pix^{-1}. Sa taille doit être 128x128.
- Créer `filter2`, un filtre numérique 2D passe-haut avec fréquence de coupure ca. 10 pix^{-1}. Sa taille doit être 128x128.
- Importer l'image contenue dans le fichier `'im11.txt'` dans la matrice `im11`.
- Calculer `im11filt`, l'image obtenue en appliquant le filtre `filter1` à `im11`.
- Importer l'image contenue dans le fichier `'im22.txt'` dans la matrice `im22`.
- Calculer `im22filt`, l'image obtenue en appliquant le filtre `filter2` à `im2`.
- Calculer `imcomb` en faisant la combinaison linéaire `im11filt + 0.33*im22filt`.
- Utiliser `Matplot` pour visualiser le résultat. Dans l'image résultante on discernera Albert Einstein qui tire la langue si on la regarde de près, et le prof qui ne tire pas la langue si on la regarde de loin.
- Expliquer pourquoi.

```
N = 128;
n = -N/2:N/2-1;
M = 128;
m = -M/2:M/2-1;
[X Y] = meshgrid(n,m);
filter1 = exp(-4*log(2)*(X.^2+Y.^2)/15^2);
filter2 = 1-exp(-4*log(2)*(X.^2+Y.^2).^3/10^6);
im11 = read('im11.txt',128,128);
im11_fft = fftshift(fft2(fftshift(im11)));
im11filt = ifftshift(fft(ifftshift(((im11_fft).*filter1)), +1));
im11filt = abs(im11filt);
im11filt = im11filt/max(max(im11filt));
im22 = read('im22.txt',128,128);
im22_fft = fftshift(fft2(fftshift(im22)));
im22filt = ifftshift(fft(ifftshift((im22_fft.*filter2)),+1));
im22filt = abs(im22filt);
im22filt = max(max(abs(im22filt)))-abs(im22filt);
im22filt = im22filt/max(max(im22filt));
imcomb = im11filt +0.33*im22filt;

f = scf(12);
subplot(1,3,1), Matplot(round(63*abs(im11filt)/max(abs(im11filt)))+1);
f.color_map = graycolormap(64);
subplot(1,3,2), Matplot(round(63*abs(im22filt)/max(abs(im22filt)))+1);
f.color_map = graycolormap(64);
subplot(1,3,3), Matplot(round(63*abs(imcomb)/max(abs(imcomb)))+1);
f.color_map = graycolormap(64);
```

Exercice 7: Optique de Fourier avec la FFT

Le but de cet Exercice est de montrer comment utiliser la FFT pour calculer la transformation d'un champ électrique entre le plan focal objet $E_{NF}(x,y)$ (champ proche, "near-field") et le plan focal image $E_{FF}(x,y)$ (champs lointain, "far-field") d'une lentille.

Posons $\widetilde{E}_{NF}(f_x,f_y)=TF[E_{NF}(x,y)]$, alors $E_{FF}(x,y)=\widetilde{E}_{NF}(f_x\cdot\lambda\cdot f,f_y\cdot\lambda\cdot f)$ où **f** est la longueur focal de la lentille; **λ** est la longueur d'onde de la lumière; et $\mathbf{f_x}$, $\mathbf{f_y}$ sont les "fréquences spatiales".

- Créer `X` et `Y`, des matrices NxM définissant la grille spatiale. Utiliser un pas `Delta` de 1 mm.
- Créer `Egauss`, une matrice NxM représentant le champs transvers d'un faisceau gaussien incident sur une masque dans le plan focal object.
- Créer `mask`, une matrice NxM représentant une masque d'amplitude et/out de phase qui se trouve dans le plan focal objet. Dans un premier temps mettre tous les éléments égals à 1.
- Calculer `E_NF`, une matrice NxM représentant le champs proche, i.e. le champs en sortie de la masque, en multipliant `Egauss` avec `mask`.
- Calculer `E_FF`, une matrice NxM représentant le champs lointain en utilisant la rélation de Fourier entre `E_NF` et `E_FF`.
- Calculer `X_FF` et `Y_FF`, des matrices NxM définissant la grille spatial de `E_FF` en observant la rélation $\mathbf{x} = \mathbf{f_x} \cdot \mathbf{l} \cdot \mathbf{f}$ (cf ci-dessus).
- Utiliser `grayplot` pour visualiser le résultat.
- Refaire pour d'autres `mask`. Par exemple une masque double-fente ou une masque de phase aléatoire.

```
N = 128;
M = 128;
Delta = 1; //[par ex mm]
n = -N/2:N/2-1;
m = -N/2:N/2-1;
x = Delta*n;
y = Delta*m;
[X,Y] = meshgrid(x,y);
k = -N/2:N/2-1;
l = -M/2:M/2-1;
fx = 1/(N*Delta)*k; //[mm-1]
fy = 1/(M*Delta)*l; //[mm-1]
[FX,FY] = meshgrid(fx,fy);
//Definir un object (faisceau gaussien), champ proche
FWHMx = 10; //[mm]
FWHMy = 10; //[mm]
E_gauss = exp(-2*log(2)*X.^2/FWHMx^2 - 2*log(2)*Y.^2/FWHMy^2);
//no mask
// = ones(N,M);
//double-slit mask
//mask = zeros(N,M);
//mask(:,14*N/32+1:15*N/32) = 1;
//mask(:,17*N/32+1:18*N/32) = 1;
//random phase mask
//mask = exp(%i*2*%pi*rand(N,M));
E_NF = mask.*E_gauss;
```

```
figure(13);
grayplot(x,y,abs(E_NF).^2);
a = gcf(); a.color_map = graycolormap(64);
//Une lentille
f = 500; //[mm]
lambda = 633e-6; //[mm]
//Coordonnees en champs lointain
x_FF = lambda*f*fx;
y_FF = lambda*f*fy;
[X_FF,Y_FF] = meshgrid(x_FF,y_FF);
//Champs lointain
E_FF = fftshift(fft2(fftshift(E_NF)));
figure(11)
grayplot(x_FF,y_FF,abs(E_FF).^2);
a = gcf(); a.color_map = graycolormap(64);
```

Exercice 8: Filtre analogique de 1e ordre

Le but de cet Exercice est de mettre en évidence certaines propriétés des filtres analogiques.
cf cours.

Voir aussi Monin: “Systèmes linéaires du premier et deuxième ordre”.

- Créer `G1`, une vecteur 1xN représentant le gain complexe d'un filtre analogique de premier ordre.
- Calculer `h1`, la fonction de réponse associée.
- Tracer la fonction de réponse; la réponse indicielle; la diagramme de Bode; et la diagramme de Nyquist.

```
N = 1024;
n = -N/2:N/2-1;
Delta = 1e-3;
t = Delta*n;
k = -N/2:N/2-1;
nu = 1/(N*Delta)*k;
//Filtre de 1e ordre
A0 = 1; nu0 = 10;
G1 = A0./(1 + %i*(2*%pi*nu)/(2*%pi*nu0));
h1 = ifftshift(ifft(ifftshift(G1)));
//Tracer fonction de reponse - reponse indicielle - Diagramme de Bode - Diagramme
de Nyquist
//to come...
//Fonction Heaviside
theta = 0*t; theta(find(t<0)) = 0; theta(find(t>=0)) = 1;
//TF de la fonction Heaviside
theta_fft = fftshift(fft(fftshift(theta)));
//Réponse indicielle
repindi1 = ifftshift(ifft(ifftshift(theta_fft.*G1)));
figure(12);
//Tracer fonction de reponse
subplot(5,1,1), plot(t(find(t>=0)),h1(find(t>=0)), 'k')
//Tracer reponse indicielle
subplot(5,1,2), plot(t(find(t>=0)), repindi1(find(t>=0)), 'k')
//Diagramme de Bode - Gain
subplot(5,1,3), plot(log10(nu(find(nu>=0))/nu0), 20*log10(abs(G1(find(nu>=0)))),
'k')
//Diagramme de Bode - Argument
subplot(5,1,4), plot(log10(nu(find(nu>=0))/nu0), phasemag(G1(find(nu>=0))), 'k')
//Diagramme de Nyquist
subplot(5,1,5), plot(real(G1(find(nu>=0))),imag(G1(find(nu>=0))), 'k')
```

Exercice 9: Filtre analogique de 2e ordre
Le but de cet Exercice est de mettre en évidence certaines propriétés des filtres analogiques.
cf cours.

Voir aussi Monin: “Systèmes linéaires du premier et deuxième ordre”.

- Créer `G2`, une vecteur 1xN représentant le gain complexe d'un filtre analogique de deuxième ordre.
- Calculer `h2`, la fonction de réponse associée.
- Tracer la fonction de réponse; la réponse indicielle; la diagramme de Bode; et la diagramme de Nyquist.
- Refaire pour des différentes valeurs de `m`, par exemple 2; 1; 0.7; 0.5; 0.2.
- Il est possible d'observer comment les différentes propriétés du filtre évolue avec m: temps de réponse à 5%; fréquence 3 dB; surtension; dépassement.

```
N = 1024;
n = -N/2:N/2-1;
Delta = 1e-3;
t = Delta*n;
k = -N/2:N/2-1;
nu = 1/(N*Delta)*k;
//Filtre de 2e ordre
//A0 = 1; nu0 = 10; m = 2;
//A0 = 1; nu0 = 10; m = 1;
//A0 = 1; nu0 = 10; m = 0.7;
//A0 = 1; nu0 = 10; m = 0.5;
A0 = 1; nu0 = 10; m = 0.2;
G2 = A0./(1 - (2*%pi*nu).^2/(2*%pi*nu0)^2 + %i*(2*m*(2*%pi*nu)/(2*%pi*nu0)));
h2 = ifftshift(ifft(ifftshift(G2)));
//Fonction de Heaviside
theta = 0*t; theta(find(t<0)) = 0; theta(find(t>=0)) = 1;
//TF de la fonction de Heaviside
theta_fft = fftshift(fft(fftshift(theta)));
//Réponse indicielle
repindi2 = ifftshift(ifft(ifftshift(theta_fft.*G2)));
figure;
//Tracer fonction de réponse
subplot(5,1,1), plot(t(find(t>=0)),h2(find(t>=0)), 'k')
//Tracer réponse indicielle
subplot(5,1,2), plot(t(find(t>=0)), repindi2(find(t>=0)), 'k')
//Tracer diagramme de Bode - Gain
subplot(5,1,3), plot(log10(nu(find(nu>=0))/nu0), 20*log10(abs(G2(find(nu>=0)))),
'k')
//Tracer diagramme de Bode - Argument
subplot(5,1,4), plot(log10(nu(find(nu>=0))/nu0), phasemag(G2(find(nu>=0))), 'k')
//Tracer diagramme de Nyquist
subplot(5,1,5), plot(real(G2(find(nu>=0))),imag(G2(find(nu>=0))), 'k')
//Possibilite de regarder temps de reponse a 5% - frequence 3dB - surtension - ...
```

Projets indépendants / avancés

“Creative time”!

Suggestion 1: Split-step Fourier method
- https://www.researchgate.net/publication/281441538_An_introduction_to_the_Split_Step_Fourier_Method_using_MATLAB

Suggestion 2: Holographie digitale
- http://wavefrontshaping.net/index.php/63-community/tutorials/phase-measurement/94-off-axis-holography

Suggestion 3: Faconnage super-Nyquist
- https://en.wikipedia.org/wiki/Pulse_shaping

Suggestion 4: Orthogonal frequency-division multiplexing (OFDM)
- https://en.wikipedia.org/wiki/Orthogonal_frequency-division_multiplexing

Suggestion 5: Propagation de Fresnel
- https://en.wikipedia.org/wiki/Fresnel_diffraction

Suggestion 6: Équation de la chaleur en 2 et 3 dimensions
- https://en.wikipedia.org/wiki/Heat_equation

Suggestion 7: Sous-échantillonnage
- https://en.wikipedia.org/wiki/Undersampling

Suggestion 8: Algorithme de Gerchberg-Saxton
- https://en.wikipedia.org/wiki/Gerchberg%E2%80%93Saxton_algorithm

Suggestion 9: Coder une TFD et la comparer avec la FFT
- cf cours

Suggestion 10: Dynamic range compression d'une piste audio
- https://en.wikipedia.org/wiki/Dynamic_range_compression