**TD/TP 1**: séries de Fourier et transformée de Fourier discrète

**Exercice 1.** Le seuillage dur (resp. doux) d'un vecteur $v \in \mathbb{R}^N$ consiste à appliquer aux coefficients de $v$ dans une base $\{e_k\}_{k=1,\dots,N}$, $v = \sum_{k=1}^{N} v_k e_k$ la fonction

$$\mathcal{S}_D(x,\eta) = \begin{cases} x & \text{si } |x| > \eta \\ 0 & \text{sinon} \end{cases} \quad \text{resp.} \quad \mathcal{S}_d(x,\eta) = \begin{cases} x-\eta & \text{si } x > \eta \\ x+\eta & \text{si } x < -\eta \\ 0 & \text{sinon} \end{cases}$$

où $\eta > 0$ est un paramètre.

En d'autres termes, on remplace le vecteur $v$ par le vecteur

$$\mathcal{S}_D(v,\eta) = \sum_{k=1}^{N} \mathcal{S}_D(v_k,\eta) e_k = \sum_{k\,:|v_k|>\eta} v_k e_k \quad \text{resp.} \quad \mathcal{S}_d(v,\eta) = \sum_{k=1}^{N} \mathcal{S}_d(v_k,\eta) e_k.$$

Écrire deux fonctions `matlab` qui prennent en entrée le paramètre $\eta$ et la suite $\{v_k\}$ et dont la sortie est le seuillage dur/doux des $v_k$, c'est-à-dire le seuillage dur/doux dans la base canonique.

Dans la suite, on considère deux fonctions 1-périodiques $f, g$ données par

$$f(t) = \begin{cases} 1 & \text{si } t \in [0,1/2) \\ 0 & \text{si } t \in [1/2,1) \end{cases} \quad \text{et} \quad g(t) = |t| \text{ pour } t \in [-1/2, 1/2).$$

(1) Déterminer les séries de Fourier de $f$ et $g$ et préciser la nature de leur convergence.

(2) Écrire deux fonctions `matlab` qui à un entier $n$ associent les suites $F_n = \{f(k/n), k = 0,\dots,n-1\}$ et $G_n = \{g(k/n), k = 0,\dots,n-1\}$. Utiliser ces deux fonctions pour tracer les graphes de $f$ et $g$ (`plot`).

*Indication:* Pour la première, on pourra utiliser les fonctions `matlab zeros` et `ones`.

(3) On prend maintenant $n = 2^k$ avec $k = 4, 5, 6, \dots, 10$.

Représenter sur un même graphe les coefficients de Fourier de $f$ $\{c_j(f), j = -2^{k-1}+1, \dots, 2^{k-1}\}$ et ceux de la transformée de Fourier discrète de $F_n$ $\mathcal{F}_n[F_n](j)$, $j = 0,\dots,n-1$ (utiliser `fft`). Comment faut-il faire pour que les 2 courbes se superposent (presque)? (*Indication:* dans l'aide de matlab, regarder à `fftshift`). Calculer la norme de l'erreur avec `norm`.

Faire de même avec $g$ et $G_n$.

(4) On prend maintenant $n = 2^{10}$.

(a) Créer une matrice $(n,n)$ $D$ dont les lignes contiennent les échantillons en $t = k/1024$, $k = 0,\dots,n-1$ de $e^{2i\pi jt}$, $j = -n/2+1,\dots,n/2$.

À l'aide de la fonction `sum`, écrire une fonction `matlab` qui à un entier $N$ (pair, $\leq n$) associe les valeurs de

$$S_N(f)(t) = \sum_{j=-N/2+1}^{N/2} c_j(f) e^{2i\pi jt}$$

en $t = k/1024$, $k = 0,\dots,n-1$.

Tracer le résultat pour différentes de valeurs de $N$. Que constatez vous?

(b) Recommancer avec $g$.

(c) Noter que $S_N$ est obtenue à partir de la série de Fourier de $f$ en remplaçant les coefficients $c_k(f)$ pour $|k| > N$ par 0.
On a par ailleurs vu que la transformée de Fourier discrète approche les coefficients de Fourier et son inverse (`ifft`) approche la somme de la série de Fourier. Pour approcher $S_N$, il suffit donc remplacer certains coefficients de la transformée de Fourier par 0. Lesquels ?
Effectuer l'opération correspondante avec `matlab` et tracer le résultat.

(5) On recommance maintenant avec une fonction moins régulière. Utiliser la fonction `matlab` Makesignal que vous téléchargerez d'abord à l'adresse suivante:

http://www.cmap.polytechnique.fr/~peyre/cours/tp_ondelettes/MakeSignal.m

et l'utiliser pour tracer un signal régulier par morceaux de longueur $n$ de la façon suivante: `f = MakeSignal('Piece-Regular', n);`

Nous allons également utiliser un signal aléatoire de longueur $n$ dont les valeurs sont (uniformément réparties) dans $[-1/2, 1/2]$ et qu'on génèrées par `g=rand(n)-1/2`.

(a) Avec `matlab` comparer $f$ (resp. $g$) avec les sommes partielles de sa série de Fourier

$$S_N(f)(t) = \sum_{k=-N/2+1}^{N/2} c_k(f) e^{2i\pi kt}$$

(resp. $S_N(g)$) pour différentes valeurs de $N$.

(b) Avec `matlab` comparer $f$ (resp. $g$) avec les seuillages doux et durs de sa série de Fourier (ou plutôt de $S_n$)

$$\mathcal{S}_{d \text{ ou } D}(f)(t) = \sum_{k=-n/2+1}^{n/2} \mathcal{S}_{d \text{ ou } D}\big(c_k(f), \eta\big) e^{2i\pi kt}$$

où $\mathcal{S}$ est la fonction de seuillage dur ou doux (cf exo précédent) et $\eta$ le seuil qu'on fera varier.

(c) Faire de même avec $f + g/100$.

**Seuillage.**

La façon naive de coder cette fonction est d'utiliser une boucle et un test:

```
function [S]=sdur(E,s)
  [J,K]=size(E);
  S=zeros(E);
  for j=1:J
    for k=1:K
    if (E(j,k)>s) S(j,k)=E(j,k)-s;
    elseif (E(j,k)<-s) S(j,k)=E(j,k)-s;
    end;
  end;
end;
```

Ceci sera très lent car `matlab` n'est pas optimisé pour parcourir des boucles et les tests sont des opérations couteuses en temps de calcul.

Pour faire mieux, il faut remarquer que le seuillage dur consiste à remplacer la suite $(v_k)$ par la suite $(v_k \times 1_{|v_k|>\eta})$. Ceci se code très aisément en `matlab` puisque la suite $1_{|v_k|>\eta}$ est exactement ce que donne la commande `(abs(v)>eta)`. Ainsi, la fonction de seuillage dur s'obtient par

```
function [S]=sdur(E,s)
  s=E.*(abs(E)>s);
end
```

La fonction de seuillage doux est un peux plus subtile puisqu'au résultat du seuillage dur on ajoutte ou on retranche $\eta$ selon que la suite soit négative ou positive. Notons que $v$ est soit positive, soit négative mais, pas les deux. On voit donc que $S_d(x,\eta) = (x-\eta)\mathbf{1}_{x>\eta} + (x+\eta)\mathbf{1}_{x<-\eta}$. On peut donc modifier le code ci-dessus

```
function [S]=sdur(E,s)
  S=(E-s).*(E>s)+(E+s).*(E<-s);
end
```

Une autre façon de voir le suillage doux: on diminue la valeur absolue de $v_k$ de $\eta$, si on obtient quelque chose de négative, on met le résultat à 0, si on obtient quelque chose de positif, on rétabli le signe de $v_k$. Ceci donne le code alternatif

```
function [S]=sdur(E,s)
  S=sign(E).*max(0,abs(E)-s);
end
```

**Les deux séries de Fourier.**

Notons d'abord qu'une primitive de $e^{\alpha t}$ est $\frac{1}{\alpha}e^{\alpha t}$, même si $\alpha \in \mathbb{C}$ (c'est le b-a-ba d'intégration si $\alpha$ est réel). En effet, notons $\alpha = a + ib$ de sorte que $e^{\alpha t} = e^{(a+ib)t} = e^{at}(\cos bt + i \sin bt)$. Mais alors, en intégrant deux fois par parties

$$\begin{aligned} I &:= \int e^{at}(\cos bt + i\sin bt)\,\mathrm{d}t = \frac{1}{a}e^{at}(\cos bt + i\sin bt) - \frac{1}{a}\int e^{at}(-b\sin bt + ib\cos bt)\,\mathrm{d}t \\ &= \frac{1}{a}e^{at}(\cos bt + i\sin bt) - \frac{1}{a^2}e^{at}(-b\sin bt + ib\cos bt) + \frac{1}{a^2}\int e^{at}(-b^2\cos bt - ib^2\sin bt)\,\mathrm{d}t \\ &= \frac{1}{a^2}e^{at}\big((a-ib)\cos bt + (ia+b)\sin bt\big) - \frac{b^2}{a^2}I. \end{aligned}$$

Ainsi

$$\begin{aligned}\frac{a^2+b^2}{a^2}I &= \frac{1}{a^2}e^{at}\big((a-ib)\cos bt+(ia+b)\sin bt\big)\\ &= \frac{1}{a^2(a+ib)}e^{at}\big((a-ib)(a+ib)\cos bt+i(a-ib)(a+ib)\sin bt\big)\\ &= \frac{1}{a^2(a+ib)}e^{at}\big((a^2+b^2)\cos bt+i(a^2+b^2)\sin bt\big)\end{aligned}$$

d'où le résultat.

On peut alors calculer

$$c_k(f)=\int_0^1 f(t)e^{-2i\pi kt}\,\mathrm{d}t=\int_0^{1/2} e^{-2i\pi kt}\,\mathrm{d}t=\begin{cases}1/2 & \text{si } k=0\\ \frac{1}{-2i\pi k}\big[e^{-2i\pi kt}\big]_0^{1/2} & \text{si } k\neq 0\end{cases}.$$

Mais

$$\frac{1}{-2i\pi k}\big[e^{-2i\pi kt}\big]_0^{1/2}=\frac{1-e^{-i\pi k}}{2i\pi k}=\frac{1-(-1)^k}{2i\pi k}=\begin{cases}0 & \text{si } k\neq 0 \text{ est pair}\\ \frac{1}{i\pi k} & \text{si } k \text{ est impair}\end{cases}.$$

Ainsi, la série de Fourier de $f$ est donnée par

$$S(f)(t)=\frac{1}{2}+\sum_{p\in\mathbb{Z}}\frac{e^{2i(2p+1)\pi t}}{i\pi(2p+1)}.$$

Comme $f$ est continue à droite et à gauche en tout point avec une dérivée à droite et à gauche (et même $\mathcal{C}^\infty$ ailleurs qu'en $k/2$, $k\in\mathbb{Z}$ *i.e.* ailleurs qu'en $\frac{1}{2}\mathbb{Z}$), $f$ vérifie les conditions du théorème de Dirichlet. Ainsi la série de Fourier converge simplement et

$$S(f)(t)=\frac{f(t^+)+f(t^-)}{2}=\begin{cases}f(t) & \text{si } t\notin\frac{1}{2}\mathbb{Z}\\ \frac{1}{2} & \text{si } t\in\frac{1}{2}\mathbb{Z}\end{cases}.$$

Notez qu'on trouve ici un des grands inconvénients des séries de Fourier: la présence d'une seule singularité empêche la bonne convergence de toute la série de Fourier (en fait elle converge un peu mieux que ce que donne le théorème de Dirichlet: la série converge uniformément sur tout interval fermé qui ne rencontre pas $\frac{1}{2}\mathbb{Z}$.

Pour $c_k(g)$, on fait une intégration par parties (pour $k\neq 0$)

$$\begin{aligned}c_k(g) &= \int_{-1/2}^0 -te^{-2i\pi kt}\,\mathrm{d}t+\int_0^{1/2} te^{-2i\pi kt}\,\mathrm{d}t\\ &= \left[-t\frac{1}{-2i\pi k}e^{-2i\pi kt}\right]_{-1/2}^0+\int_{-1/2}^0\frac{1}{-2i\pi k}e^{-2i\pi kt}\,\mathrm{d}t+\left[t\frac{1}{-2i\pi k}e^{-2i\pi kt}\right]_0^{1/2}-\int_0^{1/2}\frac{1}{-2i\pi k}e^{-2i\pi kt}\,\mathrm{d}t\\ &= \frac{1}{4i\pi k}e^{i\pi k}+\left[\frac{1}{(-2i\pi k)^2}e^{-2i\pi kt}\right]_{-1/2}^0+\frac{1}{-4i\pi k}e^{-i\pi k}-\left[\frac{1}{(-2i\pi k)^2}e^{-2i\pi kt}\right]_0^{1/2}\\ &= \frac{1-e^{i\pi k}}{(-2i\pi k)^2}-\frac{e^{-i\pi k}-1}{(-2i\pi k)^2}=\begin{cases}-\frac{1}{\pi^2k^2} & \text{si } k \text{ impair}\\ 0 & \text{si } k\neq 0 \text{ pair}\end{cases}.\end{aligned}$$

Pour $k=0$, on a

$$c_0(g)=\int_{-1/2}^{1/2}|t|\,\mathrm{d}t=2\int_0^{1/2}t\,\mathrm{d}t=\frac{1}{4}.$$

Comme $g$ est continue et $\mathcal{C}^1$ par morceaux, sa série de Fourier converge uniformément vers $g$. Ainsi

$$S(g)(t)=\frac{1}{4}-\sum_{k\in\mathbb{Z}}\frac{e^{2i(2p+1)\pi t}}{\pi^2(2p+1)^2}.$$

**Programmation matlab**

(2) La suite $F_n$ est la suite $\underbrace{1,1,\ldots,1}_{n/2 \text{ fois } 1},\underbrace{0,0,\ldots,0}_{n/2 \text{ fois } 0}$. En matlab, cela se code simplement

```
function [S]=Fn(n)
  p=floor(n/2);
S=[ones(1,p) zeros(1,n-p)];
end
```

(la première ligne sert simplement à éviter tout problème lorsque $n$ n'est pas pair).

Pour $G_n$, la première chose est de voir que $g(x) = x$ si $x \leq 1/2$ et $g(x) = 1 - x$ pour $x \in (1/2, 1)$.

Si $n$ est pair, $n = 2p$, la suite $G_n$ est la suite $0, 1/n, \ldots, p/n = 1/2, 1 - (p+1)/n = 1/2 - 1/n, \ldots, 1 - (2p-1)/n = 1/n$. C'est donc la suite aritmétique de pas $1/n$ allant de 0 à $1/2$ puis celle de pas $-1/n$ allant de $1/2 - 1/n = (n-2)/2n$ à $1/n$. On obtient bien $n$ termes en tout ($n/2 + 1$ pour la première et $n/2 - 1$ pour la seconde). En `matlab` celle-ci se construit simplement `[1:1/n:1/2 (n-2)/(2*n):-1/n:1/n]`

Si $n$ est impair, $n = 2p+1$ ($p = (n-1)/2$), la suite $G_n$ est la suite $0, 1/n, \ldots, p/n, 1 - (p+1)/n = p/n, \ldots, 1 - 2p/n = 1/n$. En `matlab` celle-ci se construit simplement `[1:1/n:1/2 (n-1)/(2*n):-1/n:1/n]` En effet, la suite `1:1/n:1/2` s'arrête bien à $p/n$ puisque le terme suivant $(p+1)/n > 1/2$.

Pour éviter d'utiliser un test, notez que $n - 2 * [n/2] = \begin{cases} 1 & \text{si } n \text{ impair} \\ 0 & \text{si } n \text{ pair} \end{cases}$, on peut donc coder la fonction $G_n$ comme suit

```
function [S]=Gn(n)
  p=n-2*floor(n/2);
S=[1:1/n:1/2 (n-2+p)/(2*n):1/n:1/n];
end
```

Dans la suite, nous serons amenés à tracer des courbes *complexes*. Cela se fait en traçant la partie réelle et la partie imaginaire. Le module est souvent une information intéressante. Nous allons donc créer une fonction pour cela.

```
function [S]=plotc(f)
  subplot(311)
plot(real(f));
subplot(312)
plot(imag(f));
subplot(313)
plot(abs(f));
end
```

`subplot` permet de tracer plusieurs courbes sur une même figure. `subplot(mnp)` signifie qu'on découpe la figure en $m$ lignes $\times n$ colonnes de sous-figures (une "matrice de figures") et qu'on va tracer le graphe suivant la $p$-ième figure (on compte par ligne: la première ligne est donc $p = 1$ à $p = n$ puis la deuxième $p = n + 1$ à $p = n + n$... jusqu'à $p = mn$.)

(3) Il faut représenter les coefficients de Fourier $c_j(f)$ pour $j = -2^{k-1} + 1$ à $j = 2^{k-1}$. La seule difficulté est que l'indexation matlab commence à 1, il faut donc un décallage: $c_{-2^{k-1}+1}(f)$ est le coefficient `cf(1)`, $c_{-2^{k-1}+2}(f)$ est le coefficient `cf(2)`,..., $c_{-2^{k-1}+k}(f)$ est le coefficient `cf(k)`. En particulier $c_0(f)$ est `cf(2^(k-1))`. On en déduit la formule

```
function [S]=cf(k)
  m=2^(k-1);
  cf=zeros(1,2*m);
  cf(m)=1/2;
  M=-m+1:2:m;
  cf(1:2:2*m)=1./(i*pi*M);
end
```

En effet, on commence par prendre pour `cf` un vecteur de bonne longueur ne contenant que des zéros: `cf=zeros(1,2*m);`. Ensuite, on définit $c_0(f)$, c'est-à-dire `cf(2^(k-1))`. Enfin, $c_n(f)$ est donné par $1/(i\pi n)$ si $n$ est impair. On prend donc les impairs de $-2^{k-1}+1$ à $2^{k-1}$ `M=-m+1:2:m;`. La suite des $1/(i\pi n)$ si $n$ est impair $-2^{k-1}+1$ à $2^{k-1}$ se calcule avec matlab avec `1./(i*pi*M);` et cette suite doit aussi être la suite `cf(1),cf(3),...` c'est-à-dire `cf(1:2:2*m)`. il suffit donc de faire `cf(1:2:2*m)=1./(i*pi*M);`.

On pouvait évidemment prendre la version plus lente

```
function [S]=cf(k)
  m=2^(k-1);
  cf=zeros(1,2*m);
  cf(m)=1/2;
  for j=-m+1:2:m cf(j)=1./(i*pi*j); end;
end
```

On trace alors les coefficients avec `plotc(cf(k))` (il faut donner une valeur à $k$: donc remplacer par $k = 4, 5, ...$).

Les coefficients de Fourier discrets de $F_n$ se calculent simplement avec (mettre une valeur à la place des ???)

```
k=???;
F=Fn(2^k);
FF=fft(F);
plotc(FF)
new figure
plotc(cf(k))
new figure
plotc(cf(k)-FF)
```

En faisant cela, les courbes ne se superposent pas et la troisième figure n'est pas nulle.

Il faut en fait ré-ordonner soit les coefficients de Fourier, soit les coefficients de Fourier discrets. Pour ces derniers `ifftshift` fait le travail, il suffit donc d'insérer une ligne

```
k=???;
F=Fn(2^k);
FF=fft(F);
FFs=ifftshift(FF);
plotc(FFs)
new figure
plotc(cf(k))
new figure
plotc(cf(k)-FFs)
```

N.B. Vérifier si `fftshift` ne marche pas mieux!

**Remarque.** On peut vérifier que la commande `ifft` permet de reconstruire $f$ (aux erreurs numériques près: attention, bien que le $f$ de départ soit réel, celui qu'on reconstruit aura une partie imaginaire non nulle, mais très petit).

```
k=???;
F=Fn(2^k);
FF=fft(F);
FR=ifft(FF);
plotc(F)
new figure
plotc(FR)
new figure
plotc(F-FR)
```

(4) Mathématiquement, $D$ est la matrice $D = [e^{2i\pi jk/1024}]_{j=-N/2+1..N/2,k=0..1023}$. En matlab, les indices commencent à 1, il suffit donc de faire un décallage en posant

– $j' = j + a$, et $j' = 1$ quand $j = -N/2 + 1$ soit $1 = -N/2 + 1 + a$, $a = N/2$ et ainsi $j' = 1..102N4$

– $k' = k + b$ et $k' = 1$ quand $k = 0$ soit $1 = 0 + b$, $b = 1$ et $k' = 1..1024$. La matrice $D$ est donc encore $D' = [e^{2i\pi(j'-N/2)(k'-1)/1024}]_{j'=1..N,k'=1..1024}$.

On peut alors définir $D'$ avec une boucle

```
function [D]=D(N)
  for j=1:N
   for k=1:1024
    D(j,k)=exp(2*i*pi*(j-N/2)*(k-1)/1024);
 end
end
end
```

En faisant plus attention, on se rend compte que $D$ est obtenu en appliquant la fonction $t \to \exp(2i\pi t/1024)$ à chaque entrée de la matrice $[(j' - N/2)(k' - 1)]_{j'=1..N,k'=1..1024}$. Mais, un peu d'algèbre linéaire montre que cette matrice est la matrice

$$\big(1 - N/2, 2 - N/2, \ldots, N/2\big) \begin{pmatrix} 1-1 \\ 2-1 \\ \vdots 1024-1 \end{pmatrix} = \big(1 - N/2, 2 - N/2, \ldots, N/2\big) \big(1-1, 2-1, \ldots 1024-1\big)^t$$

où $^t$ désigne la transposition (un "prime" en matlab).

Le programme ci-dessus peut donc être remplaçé par le suivant qui n'utilise pas de boucle et est donc bien plus rapide:

```
function [D]=D(N)
  J=1:N;
  K=1:1024;
  D=exp(2*i*pi*(J-N/2)*(K-1)'/1024);
end
```

On remarque alors que même le décallage ne sert à rien si on fait l'observation ci-dessu dès le début:

```
function [D]=D(N)
  J=-N/2+1:N/2;
  K=0:1023;
  D=exp(2*i*pi*J*K'/1024);
end
```

Un peun plus d'algèbre linéaire montre que la série de Fourier est alors obtenue en multpliant le vecteur (ligne) $[c_{-N/2}(f), \ldots, c_{N/2}(f)]$ (qu'on a programmé plus haut) par $D(N)$.

```
N=???;
c=cf(N);
D=D(N);
S=c*D;
plotc(S)
```

On constate que la somme de la série de Fourier $S$ a toujours un pic près des sauts de $f$ (sauf si $N = 1024$) et que ce pic a toujours la même hauteur (mais se déplace vers le saut quand on augmente $N$. Lorsque $N = 1024$, ce pic est toujours présent, simplement il se trouve trop près du saut pour qu'on le voit puisqu'on ne trace qu'un point tous les 1/1024-ièmes).

On a vu que la transformée de Fourier discrète/FFT approche les coefficients de Fourier, la somme de la série de Fourier est elle approchée par la transformée de Fourier inverse. La seule chose qui diffère est l'ordre des coefficients.

Ainsi, on dispose des coefficients de Fourier $c_{-n/2+1,\ldots,n/2}(f)$ et des coefficients de Fourier discrets de $F_n$ `FF(1),...,F(n)`, alors `FF(1)`$\simeq c_0(f)$,.., `FF(n/2+1)`$\simeq c_{n/2}(f)$, `F(n/2+2)`$\simeq c_{-n/2+1}(f)$,...`F(n-j+1)`$\simeq c_{-j}(f)$,...,`F(n)`$\simeq c_{-1}(f)$.

Pour approcher la somme partielle

$$\sum_{j=-p}^{q} c_j(f)e^{2i\pi jt} \tag{1}$$

de la série de Fourier de $f$, il suffit donc d'échantilloner $f$: `n=2k;`(par exemple $k = 10$ pour avoir suffisamment de valeurs), `F=Fn(n);` , de prendre la transformée de Fourier discrète `FF=fft(F);`, d'annuler les coefficients qui n'apparaissent pas dans (1) (on suppose ici que $p < 2^k/2 - 1$ et $q < 2^k/2$) `FF(q+2:n-p)=zeros(1,n-p-q-1);`

En effet $c_{q+1}(f), \ldots, c_{n/2}(f)$ sont remplacés par 0 et ils correspondent à `FF(q+2),...,FF(n/2+1)`, et $c_{-n/2+1}(f), \ldots, c_{-p-1}(f)$ sont remplacés par 0 et ils correspondent à `FF(n/2+2),...,FF(n-(p+1)+1)=FF(n-p)`. En tout, il y a $n - p - (q + 2) + 1 = n - p - q - 1$ zéros.

Enfin, on prend la transformée de Fourier inverse

```
k=???;
n=2^k;
F=Fn(n);
FF=fft(F);
q=???;
p=-q+1;
FF(q+2:n-p)=zeros(1,n-p-q-1);
FR=ifft(FF);
plotc(FR)
new figure
plotc(FR-F)
```

Notez enfin que le seuillage dur ou doux est plus simple à implémenter car il ne dépend pas de l'ordre dans lequel les coefficients sont codés. Faire varier $s$ (petit!) dans le programme ci-dessous:

```
k=???;
n=2^k;
F=Fn(n);
FF=fft(F);
s=???;
FSD=sdur(FF,s);
FSd=sdoux(FF,s);
FRD=ifft(FSD);
FRd=ifft(FSd);
plotc(FRD)
new figure
plotc(FRD-F)
new figure
plotc(FRd)
new figure
plotc(FRd-F)
```

Comme les coefficients d'indice impairs sont décroissants (et les pairs nuls), les deux méthodes reviennent au même.