TP ANALYSE APPLIQUEE
Utilisation de Transformées de Fourier sur Matlab :
`compte rendu`

Xavier Milhaud, Basile Voisin

24 avril 2006

Exercice 1 :
Séries de Fourier, étude du phénomène de Gibbs

Question 1

Cherchons la série de Fourier de f sous la forme $a_0 + \sum_{n=1}^{\infty} a_n \cos(n\omega x) + \sum_{n=1}^{\infty} b_n \sin(n\omega x)$:
Ici, f est impaire et 1-périodique, donc :

$\forall n \in \mathbb{N}$, $a_n = 0$;

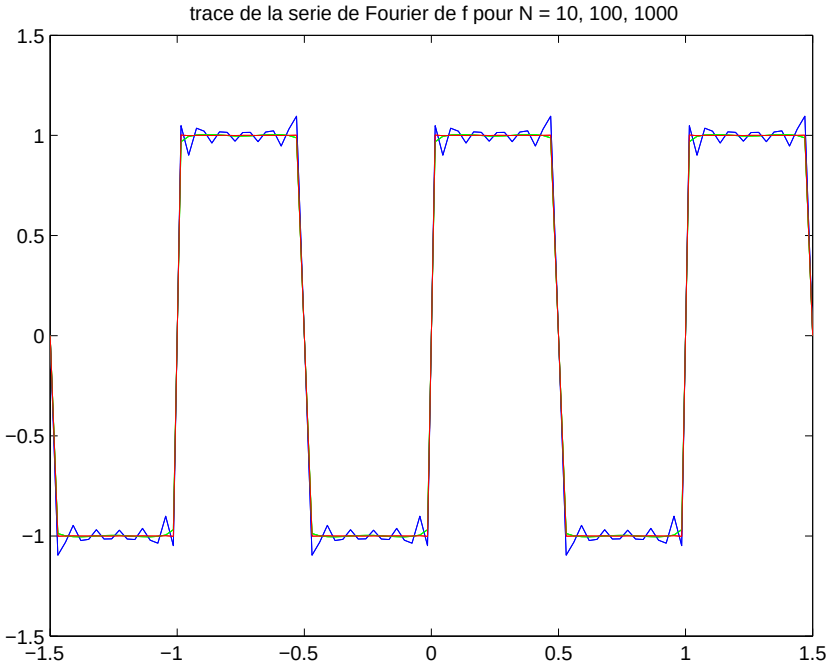
$$\begin{aligned} \text{et } b_n &= \frac{2}{T}\int_{-\frac{T}{2}}^{\frac{T}{2}} f(x)\sin(n\omega x)dx \\ &= 2\int_{-\frac{1}{2}}^{0} -\sin(2\pi nx)dx + 2\int_{0}^{\frac{1}{2}} \sin(2\pi nx)dx \\ &= \frac{2}{n\pi}\left(1-\cos(n\pi)\right) \\ &= \begin{cases} 0 & \text{si } n \text{ est pair} \\ \dfrac{4}{n\pi} & \text{si } n \text{ est impair} \end{cases} \end{aligned}$$

On a donc l'expression de la série de Fourier de f : $\frac{4}{\pi}\sum_{n=0}^{+\infty} \frac{\sin\left(2(2n+1)\pi x\right)}{2n+1}$

Question 2

Le programme traçant une approximation de f par sa série de Fourier tronquée s'écrit facilement à l'aide d'une boucle sur n (*cf. annexe 1*).

Ce programme nous a premis de tracer les courbes suivantes (en utilisant un echantillonage de 100 points) :



On constate ici des oscillations autour de la fonction initiale (dues au sinus) qui disparaissent au fur et à mesure que le nombre de termes de la somme augmente. On peut ici considere qu'a partir de 1000 termes, l'approximation de la fonction par sa série tronquée est excellente.

Exercice 2 :
Calcul des coefficients de Fourier en utilisant la FFT

Question 1

Pour calculer les coefficients de Fourier on utilise la fonction `fft` de Matlab en passant en argument le vecteur des $s(x)$.
Pour des raisons d'efficacité il sera préférable de prendre un vecteur d'ordonnées dont la taille sera une puissance de 2.

Question 2

Calcul des coefficients de Fourier théoriques $c_k(f)$ de la fonction f de l'exercice 1 :

$$\begin{aligned} c_k(f) &= \int_{-\frac{1}{2}}^{\frac{1}{2}} f(x)e^{-2i\pi kt}dt \\ &= \int_{-\frac{1}{2}}^{0} -e^{-2i\pi kt}dt + \int_{0}^{\frac{1}{2}} e^{-2i\pi kt}dt \\ &= i\frac{\cos(kt)-1}{\pi k} \end{aligned}$$

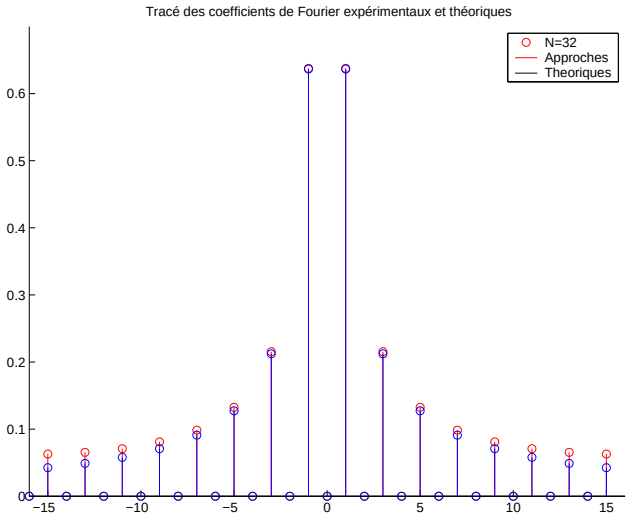
Les coefficients de Fouriers théoriques sont donc : $c_k(f) = \begin{cases} 0 & \text{si } k \text{ est pair} \\ \dfrac{2}{i\pi k} & \text{si } k \text{ est impair} \end{cases}$

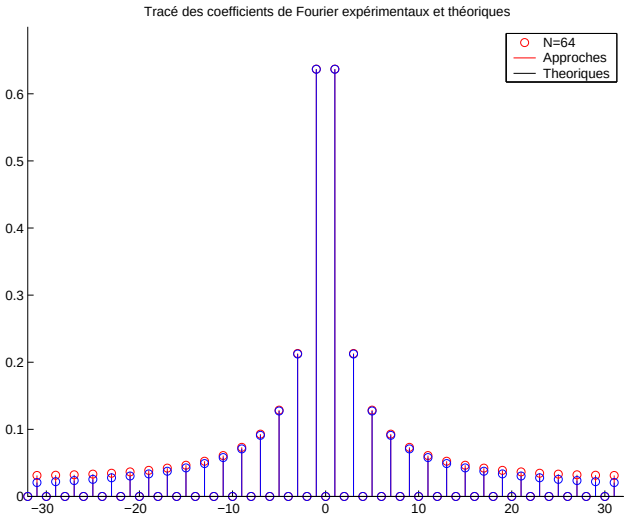
Question 3

Pour calculer les coefficients de Fourier approchés de la fonction de l'exercice 1, on applique le méthode de la question 1 à celle-ci (*cf. annexe 2*)
(Nous avions utilisé également `fft(y,N)` mais il fallait que la taille de N et des ordonnées correspondent pour avoir un résultat précis.)

Question 4

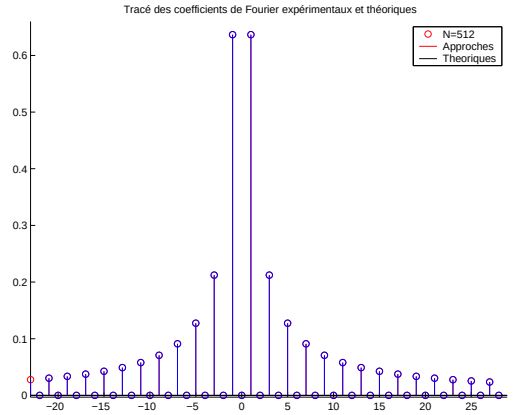
Graphes obtenus en traçant les coefficients de Fourier théoriques et calculés par la fonction `fft` de matlab pour différentes valeurs de N :

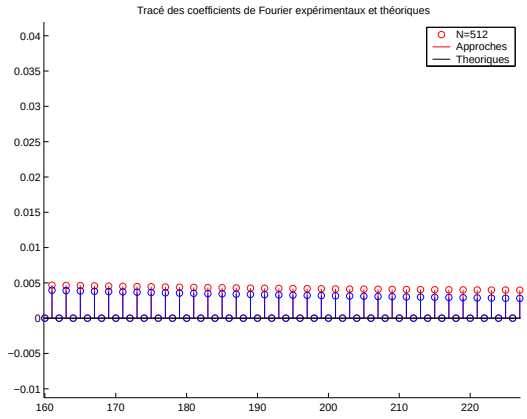




Alors que pour N différent d'une puissance de 2 nous obtenions d'assez mauvais résultats (notamment l'approximation des coefficients d'indice pair pour N impair n'était pas toujours nulle !), nous obtenons de bons résultats pour $N = 32$ ou $N = 64$.

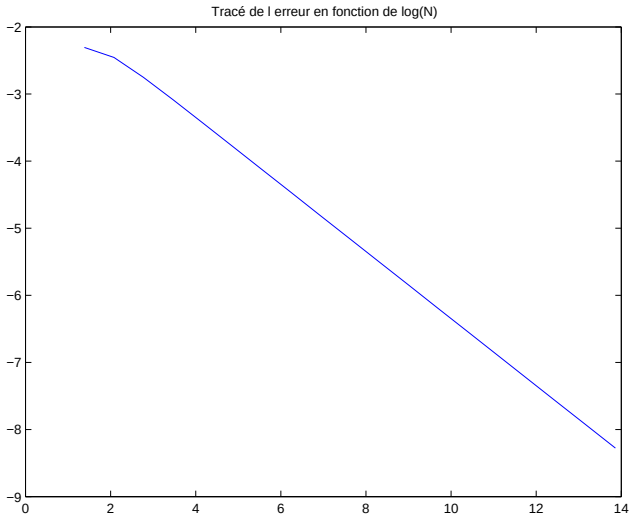
Pour $N = 512$, les résultats sont excellents. Des zooms centrés en 0 et aux extrémités nous permettent de constater que la précision du calcul est bien meilleure que pour $N = 64$ et que les écarts entre la théorie et la pratique ne subsistent que pour indices grands.





Question 5

Nous avons calculé et tracé la norme l^2 de la différence en utilisant la somme de la formule de Parseval (*cf. annexe 3*) :



Ayant traçé cette courbe avec un vecteur de puissances de 2 en entrée (abscisses), nous remarquons que le logarithme de l'erreur est proportionnel au logarithme de N pour une pente de $-0,4$. L'erreur est donc de l'ordre de $\frac{1}{N^{0,4}}$.

Exercice 3 :
Résolution d'une équation différentielle en utilisant la FFT

Question 1

Si les d_k sont les coefficients de Fourier de la fonction φ, on peut approcher φ par les fonctions φ_n.

Question 2

En remplaçant φ par φ_N dans (4) et en en cherchant une solution approchée u_N, on a la relation : $u_N(x) - u_N''(x) - \varphi_N(x) = 0$, $\forall x \in [0,1]$.
Or $u_N''(x) = \sum_{-\frac{N}{2}+1}^{\frac{N}{2}} -4\pi^2 k^2 c_k e^{2i\pi kt}$, on a donc :

$\sum_{-\frac{N}{2}+1}^{\frac{N}{2}} \left(c_k + 4\pi^2 k^2 c_k - d_k\right) e^{2i\pi kt} = 0$

Et comme les $e^{2i\pi kt}$ forment une base hilbertienne de $L^2([0,1])$, on a :

$\forall k = -\frac{N}{2}+1, \cdots, \frac{N}{2}$, $c_k\left(1 + 4\pi^2 k^2\right) = d_k$

d'où la relation permettant de calculer les c_k :

$$c_k = \frac{d_k}{1 + 4\pi^2 k^2}$$

Question 3

Le calcul suivant des $u_N\left(\frac{n}{N}\right)$

$$\begin{aligned} u_N\left(\tfrac{n}{N}\right) &= \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} c_k e^{\frac{2i\pi kn}{N}} \\ &= \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} \frac{d_k}{1 + 4\pi^2 k^2} e^{\frac{2i\pi kn}{N}} \end{aligned}$$

(en utilisant le résultat précédent)

$$u_N\left(\tfrac{n}{N}\right) = \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} \left(\frac{1}{N}\sum_{j=0}^{N-1} \varphi\left(\tfrac{j}{N}\right) e^{-\frac{2i\pi kj}{N}}\right) \frac{1}{1 + 4\pi^2 k^2} e^{\frac{2i\pi kn}{N}}$$

(avec le calcul approché des coefficients de Fourier de φ_N)

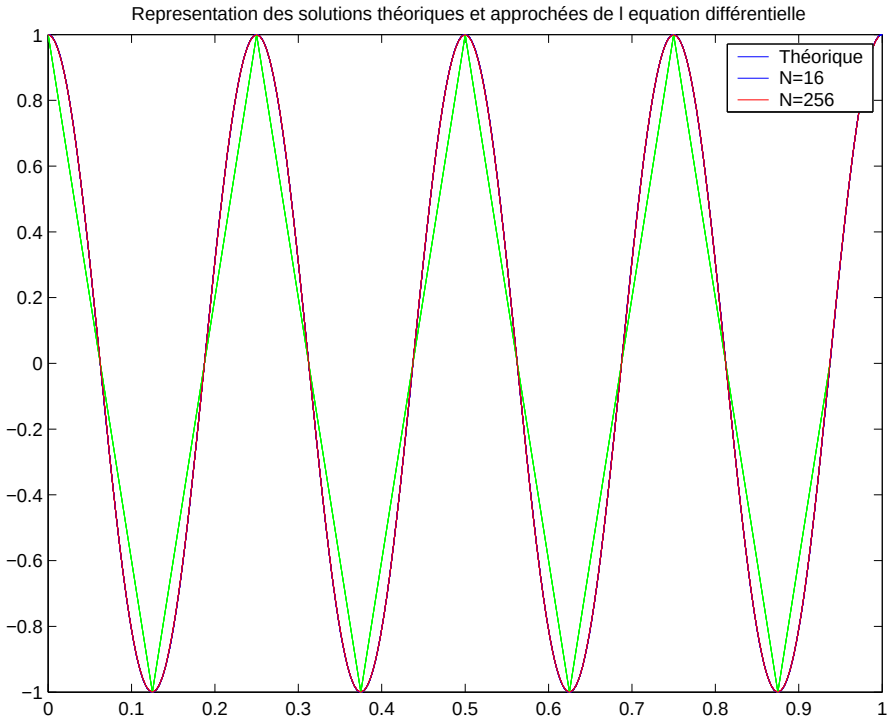
$$u_N\left(\tfrac{n}{N}\right) = \sum_{k=-\frac{N}{2}+1}^{\frac{N}{2}} \sum_{j=0}^{N-1} \frac{\varphi\left(\frac{j}{N}\right)}{N\left(1 + 4\pi^2 k^2\right)} e^{\frac{2i\pi k}{N}(n-j)}$$

nous permet d'écrire un algorithme utilisant `fft` et `ifft`. Il faut néanmoins, lors du calcul des c_k, prendre garde à l'ordre des indices k donné par les fonctions de matlab et construire artificiellement un vecteur d'indices pour ce calcul (*cf. annexe 4*).

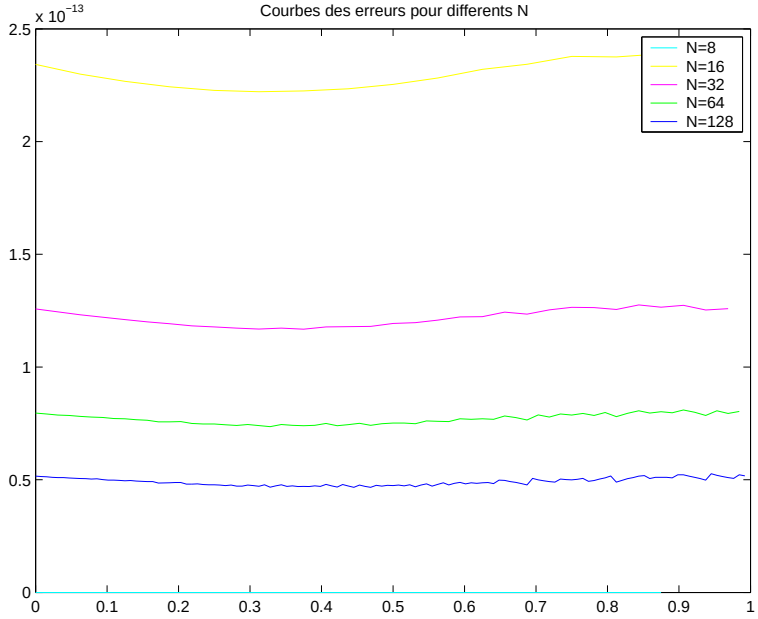
Application :
Pour pouvoir par la suite utiliser les algorithmles deja ecrits nous feront les calculs comme si la fonction φ était 1-périodique (elle est en fait $\frac{1}{4}$-périodique donc cela ne pose aucun problème).

Expression de la solution exacte du problème :
La forme des solutions de l'équation homogène est : $\alpha e^x + \beta e^{-x}$
De plus, la fonction $x \mapsto \cos(8\pi x)$ est une solution évidente de l'equation différnentielle.
Les solutions de cette équation différentielle sont donc les fonctions $x \mapsto \alpha e^x + \beta e^{-x} + \cos(8\pi x)$.
Etant donné que nous cherchons une solution $1-$périodique on prendra $\alpha = \beta = 0$ ce qui donne comme solution théorique : $\cos(8\pi x)$.

Avec cette methode, nous obtenons des resultats très satisfaisants. En effet, dès $N = 16$, les écarts entre la courbe théorique et la courbe approchée (bien que cette dernière garde ses coins anguleux dûs au faible nombre de points utilisés) sont minimes :



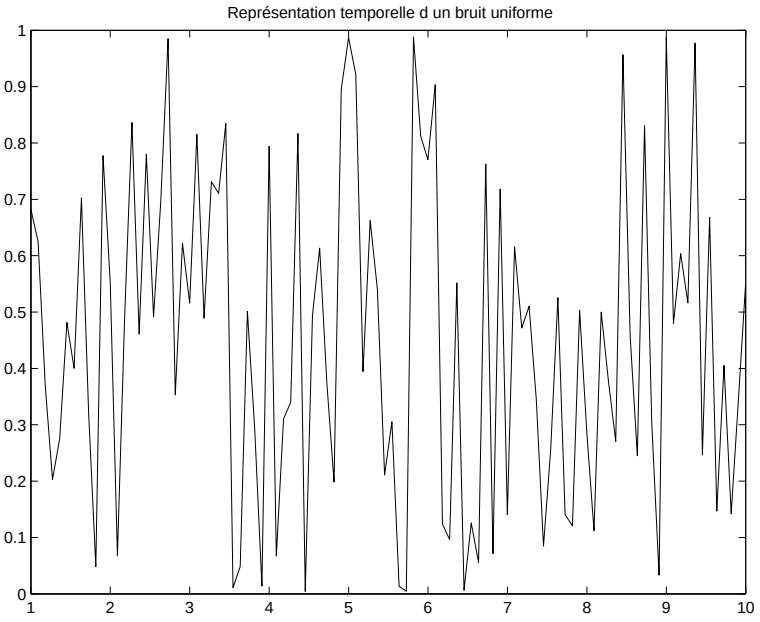
Nous avons traçé les erreurs $\|u(x) - u_N(x)\|$ sur le graphe suivant (*cf. annexe 5*) : on remarque que mis à part quand $N = 8$ où l'erreur est étonnement faible, celle-ci (de l'ordre de 10^{-13}) décroit rapidement quand N augmente.

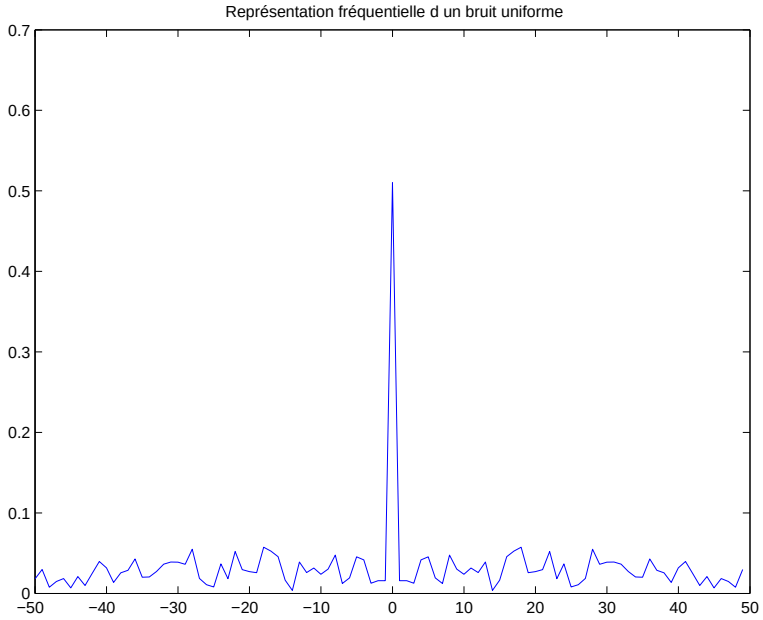


Exercice 4 :
Filtrage d'un signal bruité (deuxième séance)

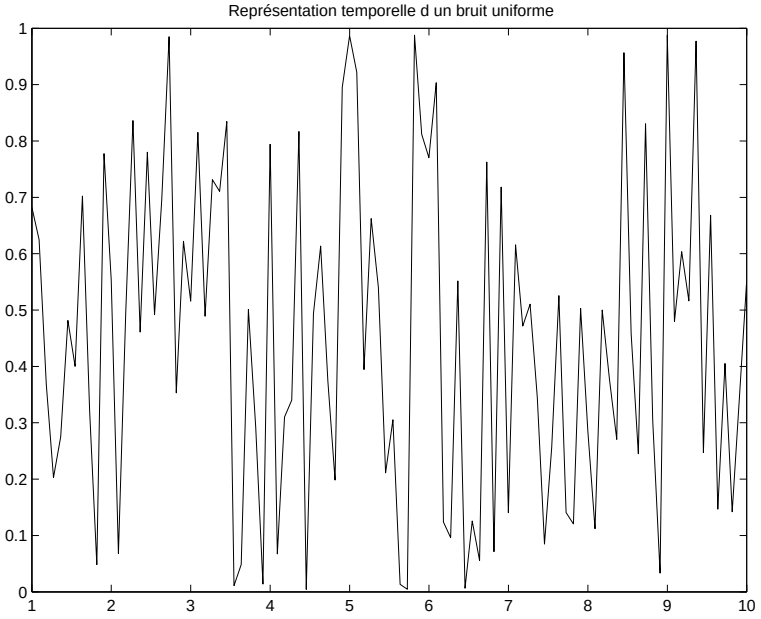
Question 1

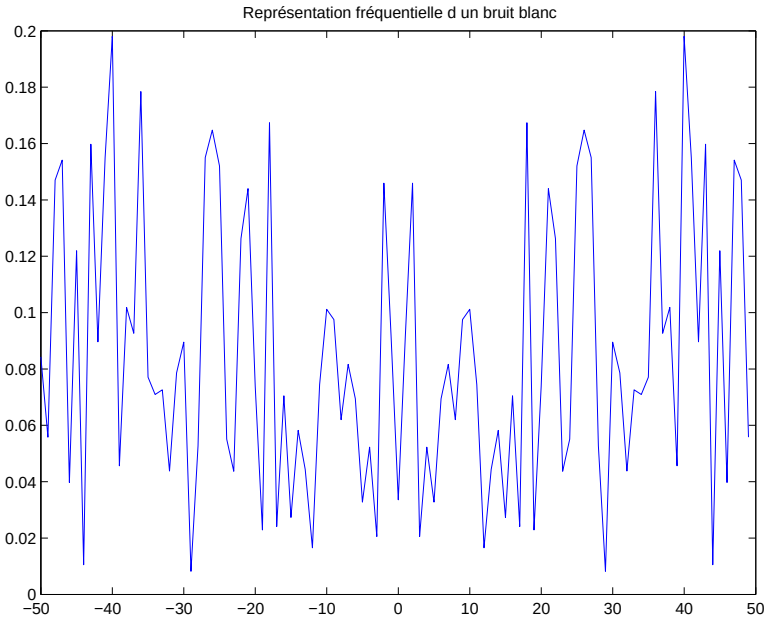
Representations temporelles et fréquentielles des différents bruits :
Bruit uniforme :





Bruit blanc :

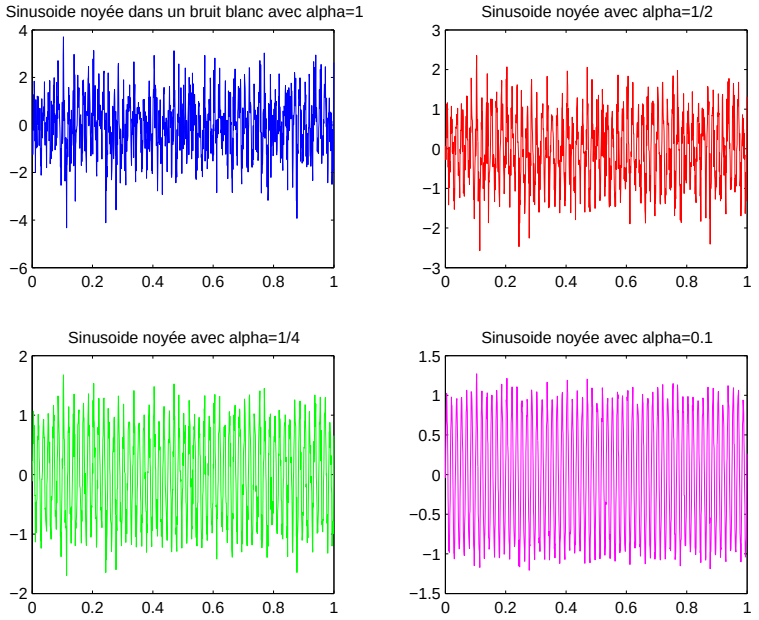




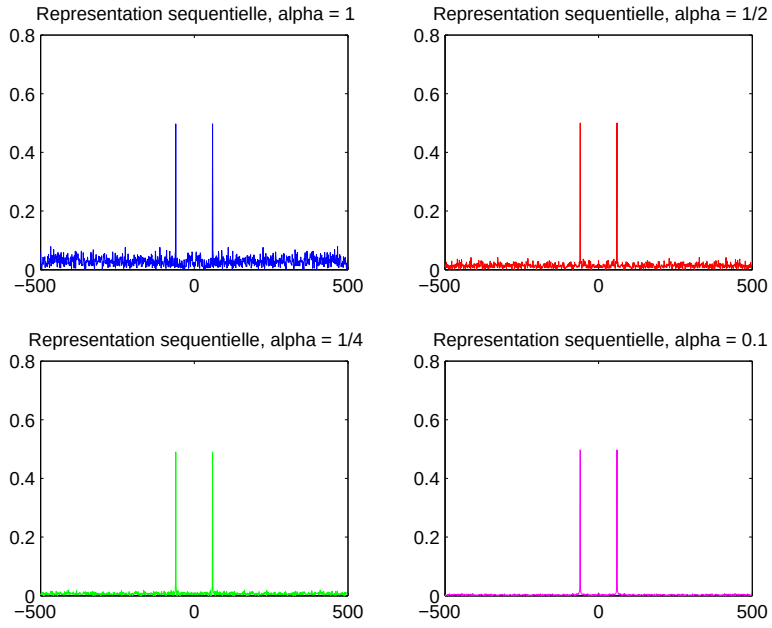
On peut remarquer que, contrairement au bruit uniforme dont la representation fréquentielle etait concentrée en un pic à la fréquence 0, la representation fréquentielle du bruit blanc s'etale sur toutes les fréquences.

Question 2.1

Representation temporelle d'une sinusoide de $60Hz$ echantillonée à $100Hz$ noyée dans un bruit blanc sur $[0, 1]$ pour differents rapports signal/bruit :



Representation fréquentielle :

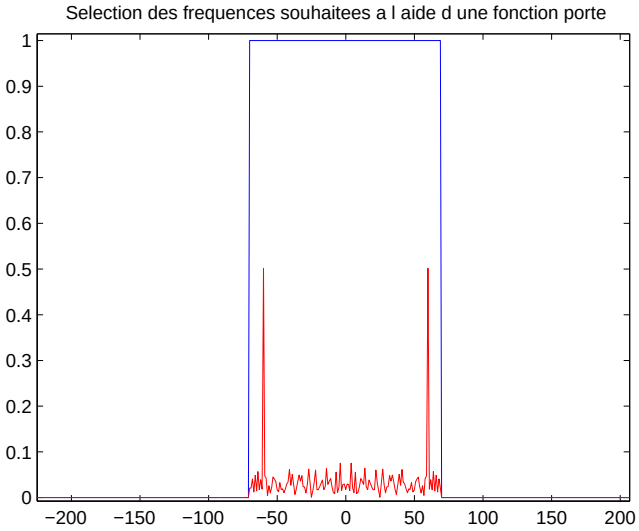


Cette derniere représentation nous permet de voir que le bruit perturbe également la transformée de Fourier mais que l'on conserve des pics aux fréquences 60 et -60, correspondant à la fréquence de la sinusoide, comme le montre ce zoom ($\alpha = 1$) :

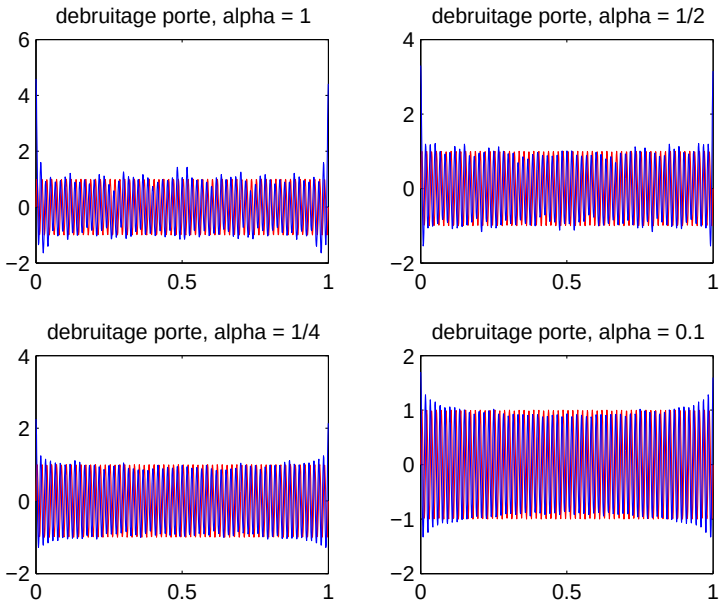
Question 2.2

Débruitage de la sinusoide bruitée à l'aide d'une fonction porte centrée en 0 et couvrant les fréquences fondamentales (de -70 à $+70$) :

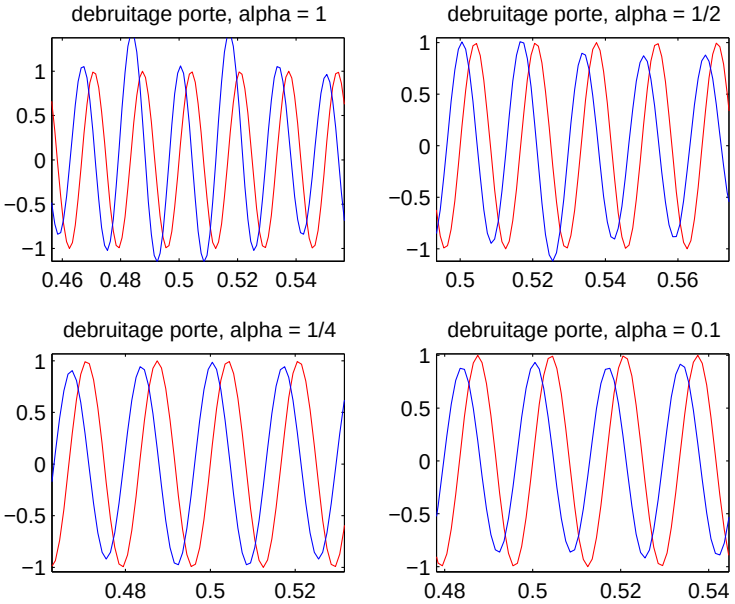
Notre fonction porte nous permet de ne selectionner qu'une partie des frequences du signal bruité, comme le montre la représentation séquentielle suivante :



Le débruitage de la sinusoide pour differents rapports signal/bruit donnent les résultats suivants :



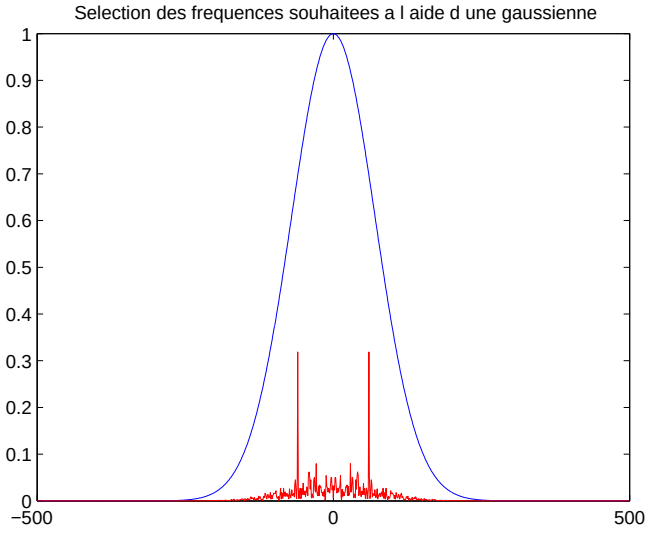
Outre des resultats surprenants aux bornes de l'intervalle, nous remarquons que le signal débruite a bien une forme sinusoidale, mais reste éloigné du signal pur, comme le montre le zoom suivant :



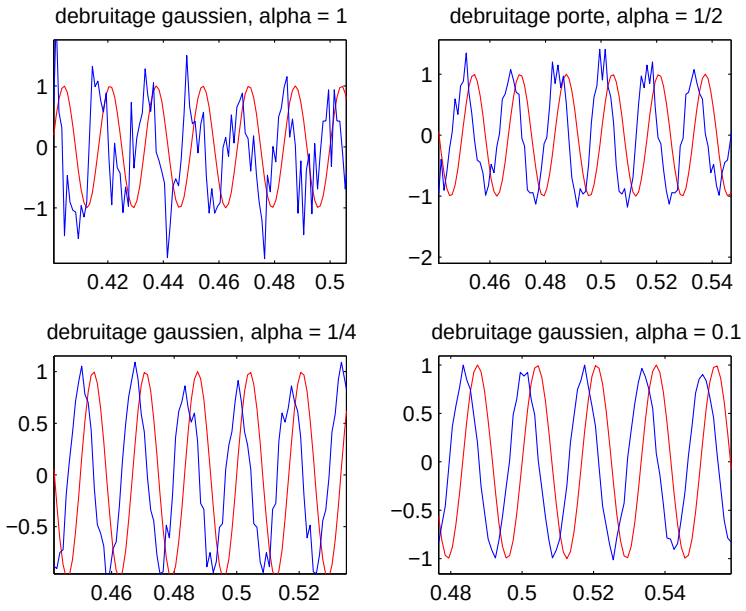
Cet agrandissement nous montre également que le signal débruité est légèrement décalé par rapport au signal original.

Debruitage de la sinusoide bruitée à l'aide d'une gaussienne centrée en 0 et couvrant les fréquences fondamentales (de -70 à $+70$) :

Notre fonction porte nous permet de ne selectionner qu'une partie des frequences du signal bruité, comme le montre la représentation séquentielle suivante :

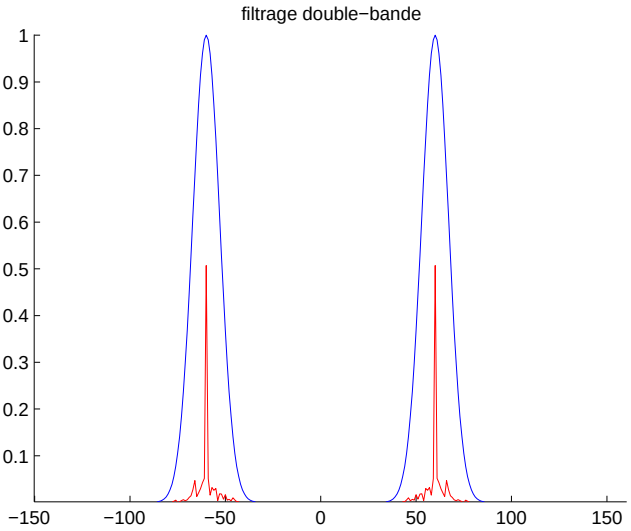


Comme pour le débruitage à l'aide de la fonction porte, nous remarquons des explosions du signal aux bornes de l'intervalle. et un décalage temporel des deux signaux. De plus, le zoom suivant nous montre que pour des bruits importants ($\alpha \geq 0,2$) le signal reste perturbé. Ce phénomène peut être explique par le fait que la gaussien filtre les fréquences extrèmes, mais n'atténue pas les fréquences proches de 0 alors qu'on perd de l'amplitude pour les fréquences qui nous interessent ($+60$ et -60Hz).

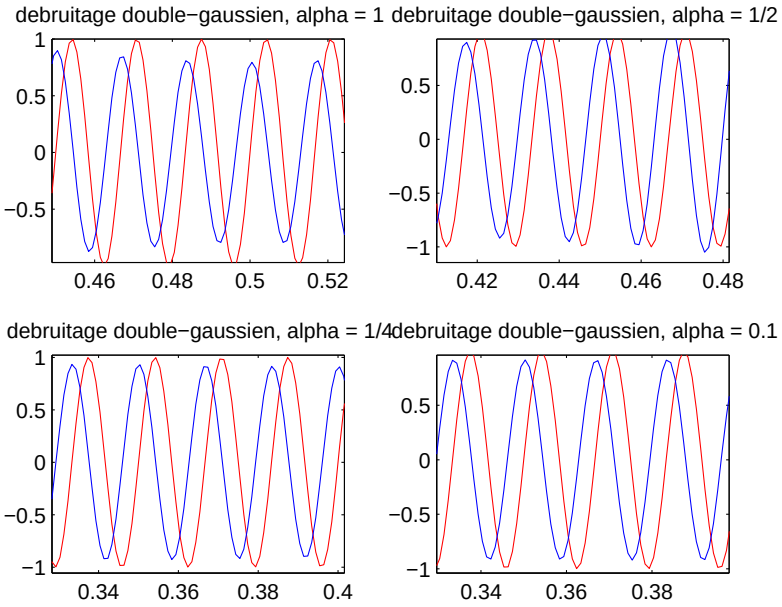


Ces deux débruitages nous montrent donc que pour débruiter une sinusoide de 60Hz, filtrer une plage de fréquences allant de -70 a 70Hz n'est pas toujours suffisant. Pour être efficace sans dégrader le signal il faut en effet que le filtrage supprime les bruits avant, entre et apres les fréquences fondamentales sans atténuer celles-ci. L'idéal serait donc d'utiliser un filtre passe"double-bande" capable de ne laisser passer parfaitement que de très petites plages centrées sur 60 et -60Hz :

Débruitage de la sinusoide à l'aide d'une fonction "double-gaussienne" :



Comme précédemment, on constate un léger décalage des deux signaux, mais il faut sourtout remarquer que, même pour des signaux fortement bruités, la fonction débruitée est parfaitement sinusoidale :



Question 2.2

La méthode de débruitage utilisée ci-dessus est bien équivalente à trouver une fonction h telle que $y(x) = (s * h)(x)$.
En effet, en appliquant la transformmée de Fourier à cette égalité, on transforme le produit de convolution en un produit classique et on obtient $H = \hat{h}$.
De même, en appliquant la transformée de Fourier inverse au produit de $\hat{y}(\nu) = \hat{s}(\nu)H(\nu)$ et en notant h la transformée inverse de H, on retrouve le produit de convolution $y(x) = (s * h)(x)$.

Remarque : Afin de ne pas surcharger le compte-rendu, nous avons décidé de ne pas vous fournir en Annexes les commandes qui nous ont servi lors de la deuxième partie du TP (les commandes sont simples).

Conclusion

Ce TP aura donc été l'occasion de mettre en pratique nos connaissances théoriques d'Analyse Appliqueées gràce au logiciel Matlab, et de voir quelques applications concrètes de la transformée de Fourier.
En effet, le filtrage est très utilisé dans le domaine de la physique et du traitement du signal et nous avons ainsi pu distinguer concrètement pourquoi l'utilisation de la transformée de Fourrier est très pratique.

annexe 1 : programme `seritronq`

```
% Prend en arguments le vecteur des points servant a tracer la fonction
function z = serietronq(x)

Ntermes = input('Entrer le nombre de termes : ') ;

z = 0 ;

for n = 0:Ntermes
     z = z+((sin(2*(2*n+1)*pi*x))/(2*n+1)) ;
end

n = (4/pi)*z ;

plot(x,z) ;
```

annexe 2 : programme `coeffourf`

```
% Programme de calcul des coefficients de Fourier
function z = coeffourf(N)

% calcul du vecteur des images et affichage de C(k) a l'abscisse k
if (mod(N,2) == 0)
         x = [0:1:(N/2)-1] ;
         t = [-N/2:1:-1] ;
         x = [x t] ;
         y = ones(1,N/2) ;
         y = [-y y] ;
else
         x = [0:1:(N-1)/2] ;
         t = [-(N-1)/2:1:-1] ;
         x = [x t] ;
         y = ones(1,(N-1)/2) ;
         y = [-y 0 y] ;
end

z = abs((1/N)*fft(y)) ;

% calcul des coefficients theoriques de Fourrier
c = x ;
for i = 1:N ,
        if (mod(c(i),2) == 0)
                 c(i) = 0;
        else
                 c(i) = 2/(pi*abs(c(i)));
        end
end

figure(1) ; hold on ;
ValN = sprintf('N=%d',N) ;
title('Trac\'e des coefficients de Fourier exp\'erimentaux et th\'eoriques') ;
axis([-N/2 N/2 0 0.7]);
```

```
Approches = stem(x,z,'r') ; Theoriques = stem(x,c,'b') ;
legend(ValN,'Approches','Theoriques');
```

annexe 3 : programme erreur

```
% Programme calculant l'erreur commise lors du calcul approch\'e des coefficients de Fourier
function e = erreur(N)

e = zeros(1,length(N)) ;

for i = 1:length(N)

% calcul du vecteur des images et affichage de C(k) a l'abscisse k
if (mod(N(i),2) == 0)
      x = [0:1:(N(i)/2)-1] ;
      t = [-N(i)/2:1:-1] ;
      x = [x t] ;
      y = ones(1,N(i)/2) ;
      y = [-y y] ;
else
      x = [0:1:(N(i)-1)/2] ;
      t = [-(N(i)-1)/2:1:-1] ;
      x = [x t] ;
      y = ones(1,(N(i)-1)/2) ;
      y = [-y 0 y] ;
end

z = abs((1/N(i))*fft(y)) ;

% calcul des coefficients theoriques de Fourrier
c = x ;
for j = 1:N(i) ,
      if (mod(c(j),2) == 0)
            c(j) = 0;
      else
            c(j) = 2/(pi*abs(c(j)));
      end
end

e(i) = 0 ;
for k = 1:N(i)
      e(i) = e(i) + (c(k)-z(k))^2 ;
end
e(i) = sqrt(e(i)) ;

end

plot(log(N),log(e)) ;
title('Trac de l erreur en fonction de log(N)') ;
```

annexe 4 : programme `eqdiff`

```
% Programme calculant et tracant des solutions approchees de l'equation differentielle
function u = eqdiff

plot(linspace(0,1,1000),cos(8*pi*linspace(0,1,1000)),'b')

Th = sprintf('Thorique') ;
Ap1 = sprintf('N=16') ;
Ap2 = sprintf('N=256') ;

t = linspace(0,1,2^4+1) ;
t = t(1:2^4) ;
x = (1+64*pi*pi)*cos(8*pi*t) ;
hold on ;
plot(t,calcul(x),'g') ;

t = linspace(0,1,2 \^{} 8+1) ;
t = t(1:2^8) ;
x = (1+64*pi*pi)*cos(8*pi*t) ;
hold on ;
plot(t,calcul(x),'r') ;

axis([0 1 -1 1]) ;
title('Representation des solutions thoriques et approches de l equation diffrentielle') ;
legend(Th, Ap1,Ap2) ;

% Fonction auxiliaire calculant une solution approchee pour des phiN donnes
function z = calcul(y)

N = length(y) ;

% Calcul des dk :
z = fft(y) ;

% Calcul des ck :
if (mod(N,2) == 0)
      x = [0:1:(N/2)-1] ;
      t = [-N/2:1:-1] ;
      x = [x t] ;
else
      x = [0:1:(N-1)/2] ;
      t = [-(N-1)/2:1:-1] ;
      x = [x t] ;
end

z = z./(ones(1,N)+4*pi*pi*x.^2) ;

% Calcul des uN(tn)
z = ifft(z) ;
```

annexe 5 : programme `erreureqdiff`

```
% Programme calculant et tracant l'erreur entre la solution approchee
% et la solution theorique de l'equation differentielle.

function u = erreureqdiff

colors=[ 'c' 'y' 'm' 'g' 'b' ];    % Differentes couleurs pr les courbes suivant N

for i = 3:7
      t = linspace(0,1,2^i+1) ;
      t = t(1:2^i) ;
      x = (1+64*pi*pi)*cos(8*pi*t) ;
      plot(t,abs(cos(8*pi*t)-calcul(x)),colors(i-2)) ;
      hold on ;
end

title('Courbes des erreurs pour differents N') ;
legend('N=8','N=16','N=32','N=64','N=128') ;

function z = calcul(y)

N = length(y) ;

% Calcul des dk :

z = fft(y) ;

% Calcul des ck :

if (mod(N,2) == 0)
      x = [0:1:(N/2)-1] ;
      t = [-N/2:1:-1] ;
      x = [x t] ;
else
      x = [0:1:(N-1)/2] ;
      t = [-(N-1)/2:1:-1] ;
      x = [x t] ;
end

z = z./(ones(1,N)+4*pi*pi*x^2) ;

% Calcul des uN(tn)

z = ifft(z) ;
```