TD 3 : représentation spectrale du son

1 Bibliothèque mathématique

La bibliothèque mathématique fournit une implantation de nombreuses fonctions dont vous trouverez une description à l'adresse suivante :

`http://www.gnu.org/software/libc/manual/html_node/Mathematics.html`

Le programme suivant utilise la fonction cosinus (`cos`) de la bibliothèque mathématique :

```
#include <stdlib.h>
#include <stdio.h>

/* bibliothèque mathématique */
#include <math.h>

int main (void)
{
  double phi = 0.5;

  printf ("le cosinus de %f radians vaut %f\n", phi, cos(phi));

  return EXIT_SUCCESS;
}
```

Lors de l'édition de liens, on doit spécifier quelles bibliothèques sont utilisées ainsi que leurs emplacements. Les chemins d'accès aux bibliothèques non standard peuvent être introduits comme suit : `-L<chemin>`. Ensuite, l'utilisation d'une bibliothèque est spécifiée ainsi : `-l<nom>`.
Pour la bibliothèque mathématique (`lib`**`m`**) standard, cela donne :

```
gcc exemple.c -o exemple -lm
```

Exercice 1 : Quels sont les avantages et inconvénients potentiels de l'utilisation de fonctions disponibles dans une bibliothèque par rapport à une implantation directe ?

2 Transformée de Fourier discrète

La bibliothèque FFTW fournit les types réel (`fftw_real`, équivalent au type standard `double`) et surtout complexe `fftw_complex`, en représentation cartésienne (parties réelle et imaginaire) :

```
#include <fftw.h>
...
fftw_complex i;

c_re (i) = 0.0;
c_im (i) = 1.0;
```

Utilisation de la bibliothèque FFTW :

– en-tête :
```
#include <fftw.h>
```
– compilation :
```
gcc -I/usr/local/include -c exemple.c -o exemple.o
gcc exemple.o -o exemple -L/usr/local/lib -lfftw -lm
```

La transformée de Fourier discrète (DFT, *Discrete Fourier Transform*) est une opération mathématique définie par :

$$S[m] = \sum_{n=0}^{N-1} s[n]\, e^{-i\frac{2\pi}{N}nm} \quad (0 \leq m < N)$$

Exercice 2 : Écrire la fonction

```
void dft (double s[N], fftw_complex S[N])
```

qui calcule la transformée de Fourier discrète d'une trame `s` de N échantillons et range le résultat dans le spectre `S` composé de N nombres complexes.

Exercice 3 : On souhaite afficher le module du spectre. Quelle formule mathématique permet de calculer le module d'un nombre complexe ?

Exercice 4 : La bibliothèque mathématique fournit une fonction pour cette opération. Quelle est-elle ?

Pour pouvoir retrouver un nombre complexe à partir de son module (son amplitude), il faut connaître aussi son argument (sa phase). C'est la fonction `atan2` de la bibliothèque mathématique qui sera alors utilisée.

Exercice 5 : Écrire une fonction

```
void cartesian_to_polar (fftw_complex S[N], fftw_real amp[N], fftw_real phs[N])
```

qui transforme les complexes du spectre `S` de la représentation cartésienne (parties réelles et imaginaires) par défaut en représentation polaire (amplitudes et phases).

Exercice 6 : Écrire un programme qui permet la visualisation du spectre de Fourier à court terme d'un signal. Ce programme doit afficher l'amplitude de spectres de trames successives. Le pas d'avancement entre deux trames de N échantillons doit être de $N/2$ échantillons.

Exercice 7 : L'affichage du module du spectre complet (indices de 0 à $N-1$) est-il pertinent ? Pourquoi ?

3 Transformée de Fourier rapide

La transformée de Fourier rapide (FFT, *Fast Fourier Transform*) réduit le nombre d'opérations nécessaires grâce à une structure algorithmique particulière mais aussi parfois en pré-calculant notamment les valeurs des exponentielles complexes. C'est le rôle du plan qui doit donc être mis en place une seule fois :

```
static fftw_plan plan;
 ...
plan = fftw_create_plan (N, FFTW_FORWARD, FFTW_ESTIMATE | FFTW_IN_PLACE);
```

Ce plan est ainsi disponible pour un nombre quelconque de transformées (autant de fois que nécessaire) :

```
#define N 1024
 ...
fftw_real s[N];  /* domaine temporel */
 ...
fftw_complex data[N];
```

```
 ...
for (i=0; i<N; i++)
  {
    c_re (data[i]) = s[i];
    c_im (data[i]) = 0;
  }
 ...
fftw_one (plan, data, NULL);
```

Le plan doit ensuite être détruit :

```
fftw_destroy_plan (plan);
```

Exercice 8 : Utiliser la bibliothèque FFTW (`www.fftw.org`) pour remplacer la transformée de Fourier discrète – et lente – précédente par la transformée de Fourier rapide (FFT) : écrire la fonction `fft` correspondante.

Exercice 9 : Du fait du fonctionnement par l'intermédiaire du plan, vous aurez besoin d'une fonction d'initialisation `fft_init` (à appeler avant tout appel à la fonction `fft`) et d'une fonction `fft_exit` (à appeler avant de quitter votre programme). Écrire les fonctions `fft_init` et `fft_exit`.

Conseil : En cas de problème avec la transformée de Fourier, il peut être utile de la tester sur un signal simple, le plus simple étant une sinusoïde d'amplitude a et de fréquence f, du type :

$$s[n] = a\cos(2\pi f n/F_e)$$

Remarque : la constante π est disponible dans la bibliothèque mathématique...

4 Performances

On souhaite estimer le gain de performance apporté par l'utilisation de la FFT par rapport à l'utilisation de la DFT. Pour cela, on doit disposer d'une estimation du temps de calcul nécessaire à chacune des deux solutions grâce à la fonction standard `clock` :

```
#include <stdio.h>

#include <time.h>

...

clock_t t1, t2;
double delta_t;

t1 = clock ();

/* le code à chronométrer */
...

t2 = clock ();

delta_t = t2 - t1;

printf ("le temps écoulé est de %f secondes\n", delta_t / CLOCK_PER_SEC);
```

Exercice 10 : Calculer les temps de calcul nécessaires à la DFT et la FFT lors de l'exécution du programme pour une trame du fichier `toms.aiff`. Les résultats sont-ils exploitables ? Pourquoi ?

Exercice 11 : Quelle est la complexité théorique des deux algorithmes testés ?

Exercice 12 : Mesurer le temps moyen nécessaire à la DFT lors de l'analyse du fichier `toms.aiff` complet.

Exercice 13 : Quel est alors le temps théorique nécessaire à la FFT pour ce même fichier ?

Exercice 14 : Mesurer le temps moyen nécessaire à la FFT pour analyser ce fichier.
Cette mesure correspond-elle au temps théorique attendu ? Pourquoi ?