Blin Sébastien
Collin Pierre-Henri

ECOLE SUPERIEURE
D'INGENIEURS DE RENNES

École supérieure d'ingénieurs de Rennes

1ère Année
Parcours Informatique

Algorithmie et complexité

Compte-Rendu TP2

Sous l'encadrement de :

Ridoux Olivier
Maurel Pierre

1 Objectif

Ce second TP a pour objectif de nous faire implémenter la transformée de Fourier rapide et de nous faire découvrir 2 de ses applications. Dans la première partie de ce TP, nous allons donc utiliser la transformée de Fourier pour pouvoir réaliser des multiplications de polynômes. Dans la seconde partie, nous allons utiliser la transformée de Fourier pour pouvoir compresser des images, comme le fait en partie la compression jpeg par exemple.

2 Multiplication de polynômes

2.1 Moyens mis en œuvre

Pour la partie implémentation, nous nous sommes basés sur l'algorithme donné pour implémenter la transformée de Fourier. Voici notre méthode combine

```
//"combine" c1 et c2 selon la formule vue en TD
// c1 et c2 sont de même taille
// la taille du résultat est le double de la taille de c1
public static CpxTab combine(CpxTab c1, CpxTab c2) {
  assert (c1.taille()==c2.taille()) : "combine: c1 et c2 ne sont pas de même taille, taille c1=
  int taille = c1.taille()+c2.taille();
  CpxTab res = new CpxTab(taille);
  for(int i=0; i<c1.taille(); i++)
  {
    //première partie du tableau
    double termeReel =c2.get_p_reel(i)*Math.cos(2*Math.PI*i/taille)-c2.get_p_imag(i)*Math.sin(2
    double termeImag = c2.get_p_imag(i)*Math.cos(2*Math.PI*i/taille)+c2.get_p_reel(i)*Math.sin(

    res.set_p_reel(i,c1.get_p_reel(i)+termeReel);
    res.set_p_imag(i,c1.get_p_imag(i)+termeImag);
    //deuxième partie du tableau
    res.set_p_reel(c1.taille()+i,c1.get_p_reel(i)-termeReel);
    res.set_p_imag(c1.taille()+i,c1.get_p_imag(i)-termeImag);

  }
  return res;
}
```

Et la méthode pour calculer la FFT :

```
//renvoie la TFD d'un tableau de complexes
//la taille de x doit être une puissance de 2
public static CpxTab FFT(CpxTab x) {
  //A FAIRE : Test d'arrêt
  if(x.taille()==1)
    return x;
  assert (x.taille()%2==0) : "FFT: la taille de x doit être une puissance de 2";

  CpxTab pair = new CpxTab(x.taille()/2);
  CpxTab impair = new CpxTab(x.taille()/2);
```

```
    for(int i=0; i<x.taille(); i++)
    {
      if(i%2==0)
      {
        pair.set_p_reel(i/2,x.get_p_reel(i));
        pair.set_p_imag(i/2,x.get_p_imag(i));
      }
      else
      {
        impair.set_p_reel((i-1)/2,x.get_p_reel(i));
        impair.set_p_imag((i-1)/2,x.get_p_imag(i));
      }
    }

    return combine(FFT(pair), FFT(impair));
  }
```

Enfin la méthode inverse

```
  //renvoie la tr\tansformée de Fourier inverse de y
  public static CpxTab FFT_inverse(CpxTab y) {
    //A FAIRE
    CpxTab t = new CpxTab(y.taille());
    for(int i = 0; i < y.taille(); ++i)
      t.set_p_reel(i, 1./(double)y.taille());
    return CpxTab.multiplie(t, (FFT(y.conjuge())).conjuge());
  }
```

Nous l'avons ensuite testé sur des polynômes pour comparer les méthodes avec transformée de Fourier rapide et la méthode classique.

2.2 Résultats

On remarque que la méthode avec transformée de Fourier est plus rapide pour un polynôme a 4000 coefficients. En dessous, les méthodes sont à peu prêt équivalents niveau temps.

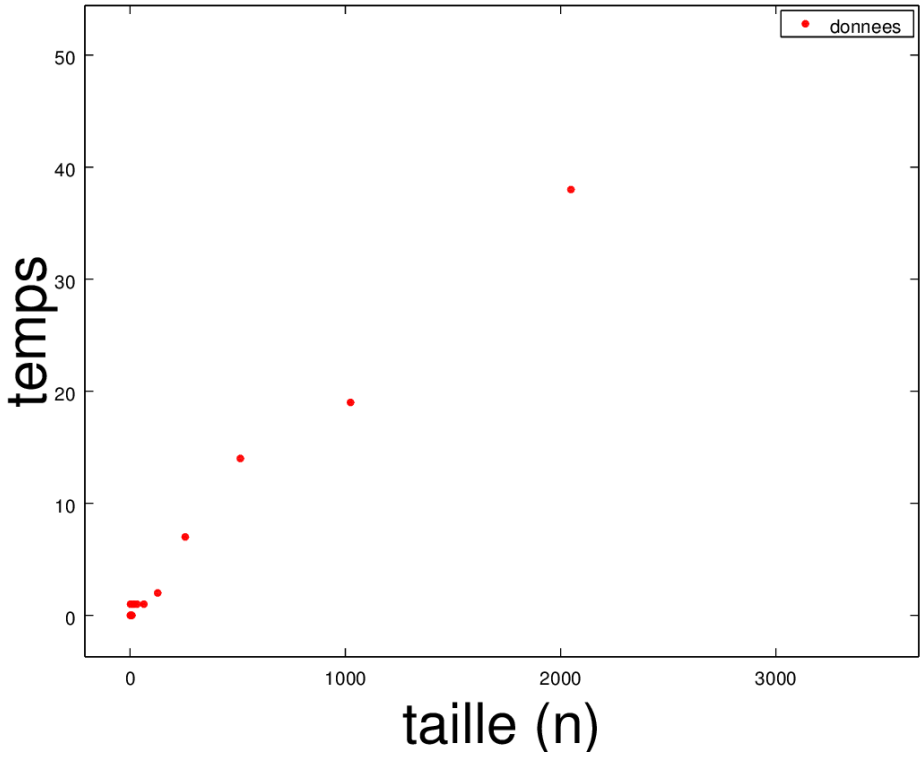
3 Compression d'images

3.1 Moyens mis en œuvre

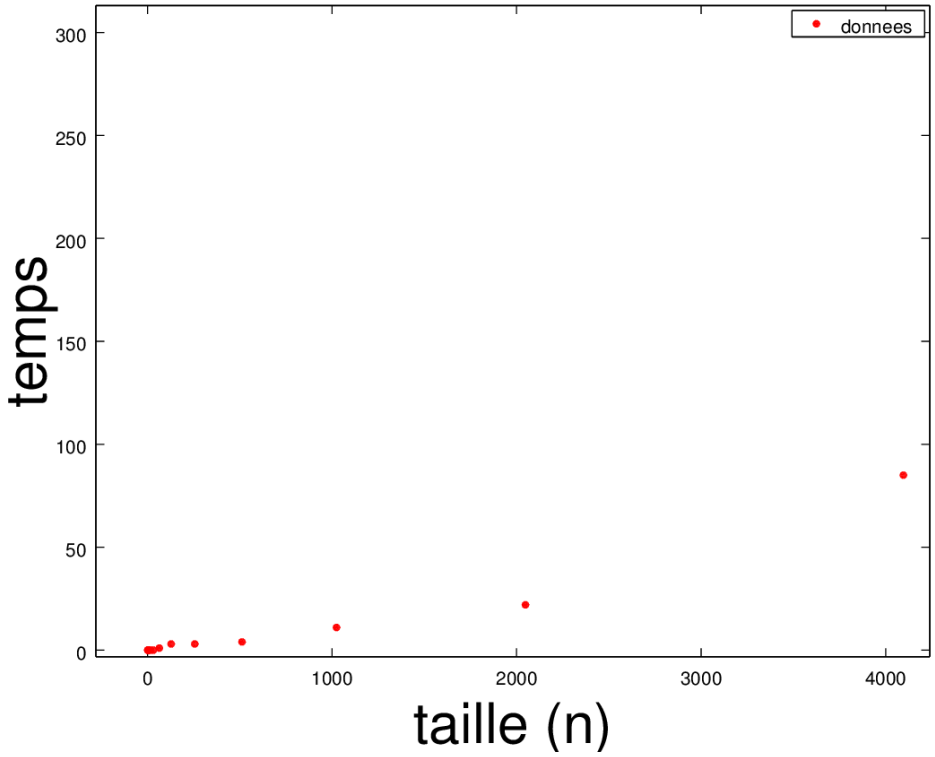
Dans la seconde partie de ce TP nous avons tout d'abord calculé la TFD de différents signaux (constant, sinusoides, etc.). Nous avons alors remarqué que la TFD nous donne la fréquence. Par exemple pour un signal constant, le tableau de sortie nous donne seulement la première valeur. Pour une sinusoide, on obtient 2 valeurs par symétrique, etc. Nous avons ensuite appliqué la transformée en 2 dimensions sur des images.

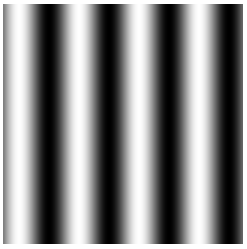
3.2 Résultats

Tout d'abord, pour l'image mire1, nous remarquons qu'elle ressemble fortement à une onde de fréquence fixe qui se propage horizontalement. Du coup, sa transformée contiendra 3 points

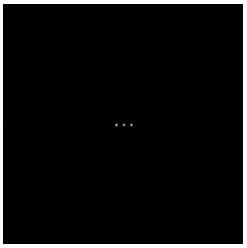


Avec FFT


donnees
temps
taille (n)
0
50
100
150
200
250
300
0
1000
2000
3000
4000
Sans FFT



mire1



FFT mire1

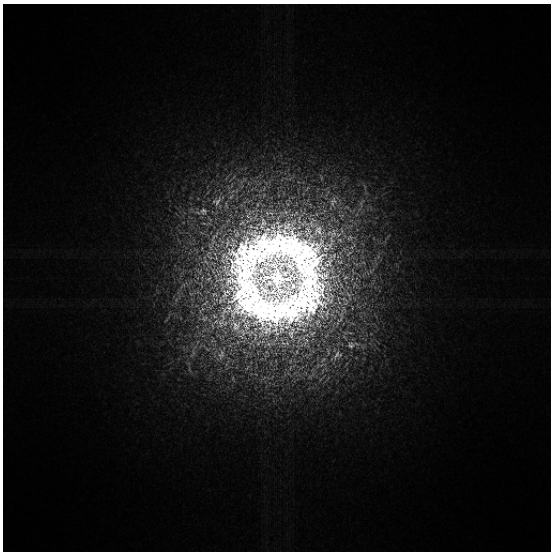
(le centre qui est la somme de toutes les valeurs, une qui représente la fréquence sur l'horizontale et son symétrique). Fingerprint a quand à elle, beaucoup de basses fréquences dans tous les sens, d'où sa transformée en forme de cercle.
Enfin pour finir sur les exemples, barbara contient énormément de rayures dans certaines directions, d'où les sortes d'halo sur l'extérieure de l'image. La barre horizontale représentant ses vêtements.

Pour la compression, il y a 2 méthodes. Soit on décide d'enlever toutes les hautes fréquences. On a juste à sélectionner la partie centrale de l'image et enlever tout le reste, notre oeil étant sensible aux basses fréquences. Ou d'enlever toutes les images en dessous d'un seuil fixé. Ainsi dans le cas des fréquences, la compression fini par montrer visuellement les différentes ondes (cf tigre compresse) alors que la compression par seuil va appliquer un effet de grain (cf tigre compresse seuil). Parfois, la compression peut donner des résultats assez marrants, en enlevant par exemple les rayures de l'image de barbara.

4 Conclusion

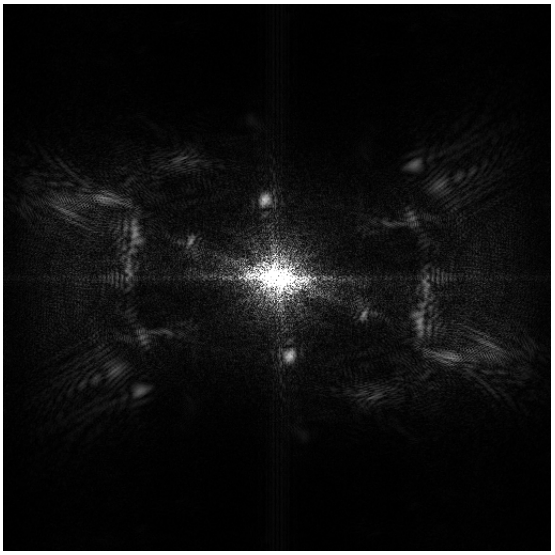
On remarque donc que la transformée de Fourier est très présente dès qu'il y a un traitement sur un signal à réaliser. Ici, nous n'avons vu qu'une toute petite partie de ses applications.

fingerprint



FFT fingerprint

barbara

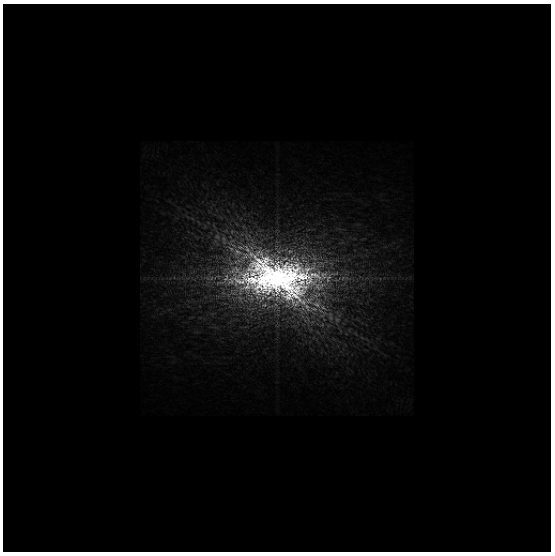


FFT barbara

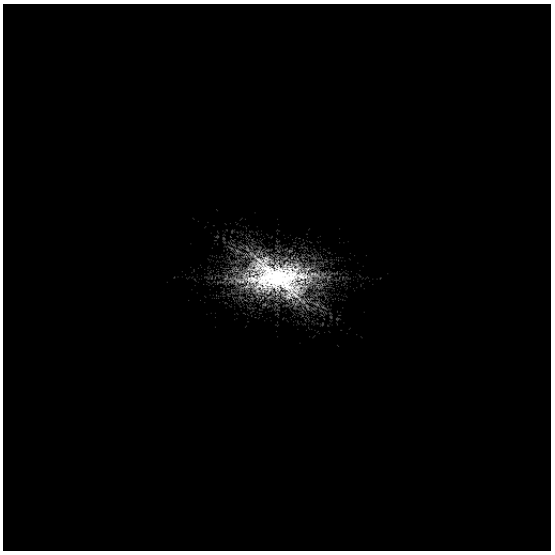
tigre

Compression par fréquences

Compression par seuil



Compression par fréquences



Compression par seuil

Barbara compressée