Janne Koljonen
Vaasa yliopisto
16.3.2012
AUTO1030 Signaalien käsittely
Harjoitus 7, tehtävät 6–7

Fourier-muunnoksen laskeminen Javalla

Diskreetin Fourier-muunnoksen (DFT) kaava voidaan muuntaa Eulerin kaavalla muotoon, jossa reaaliosa ja imaginääriosa voidaan laskea erikseen ilman kompleksilukulaskentaa:

$$X(k) = \sum_{n=0}^{N-1} x(n)\mathrm{e}^{-\mathrm{j}k\Omega n} = \sum_{n=0}^{N-1} x(n)\cos(k\Omega n) - \mathrm{j}\sum_{n=0}^{N-1} x(n)\sin(k\Omega n),$$

jossa $k = \{0, 1, N-1\}$, $\Omega = 2\pi/N$ ja N on aikasarjan pituus.

Vastaavasti saadaan käänteismuunnos (IDFT):

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\mathrm{e}^{\mathrm{i}k\Omega n} = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\cos(k\Omega n) + \mathrm{j}\frac{1}{N}\sum_{k=0}^{N-1} X(k)\sin(k\Omega n)$$

Oletetaan, että sekä aikasarjat $x(n)$ että Fourier-muunnokset $X(k)$ voivat olla kompleksisia. Tämäkään ei juuri vaikeuta laskentaa, koska aikasarjat ja laskenta voidaan jakaa reaali- ja imaginääriosiin:

$$X(k) = \sum_{n=0}^{N-1} x(n)\cos(k\Omega n) - \mathrm{j}\sum_{n=0}^{N-1} x(n)\sin(k\Omega n) =$$

$$\sum_{n=0}^{N-1}\left(\mathrm{Re}\{x(n)\} + \mathrm{j}\,\mathrm{Im}\{x(n)\}\right)\cos(k\Omega n) - \mathrm{j}\sum_{n=0}^{N-1}\left(\mathrm{Re}\{x(n)\} + \mathrm{j}\,\mathrm{Im}\{x(n)\}\right)\sin(k\Omega n) =$$

$$\sum_{n=0}^{N-1}\mathrm{Re}\{x(n)\}\cos(k\Omega n) + \mathrm{j}\sum_{n=0}^{N-1}\mathrm{Im}\{x(n)\}\cos(k\Omega n) - \mathrm{j}\sum_{n=0}^{N-1}\mathrm{Re}\{x(n)\}\sin(k\Omega n) - \mathrm{j}\sum_{n=0}^{N-1}\mathrm{j}\,\mathrm{Im}\{x(n)\}\sin(k\Omega n) =$$

$$\sum_{n=0}^{N-1}\mathrm{Re}\{x(n)\}\cos(k\Omega n) + \mathrm{j}\sum_{n=0}^{N-1}\mathrm{Im}\{x(n)\}\cos(k\Omega n) - \mathrm{j}\sum_{n=0}^{N-1}\mathrm{Re}\{x(n)\}\sin(k\Omega n) + \sum_{n=0}^{N-1}\mathrm{Im}\{x(n)\}\sin(k\Omega n) =$$

$$\left(\sum_{n=0}^{N-1}\mathrm{Re}\{x(n)\}\cos(k\Omega n) + \sum_{n=0}^{N-1}\mathrm{Im}\{x(n)\}\sin(k\Omega n)\right) + \mathrm{j}\left(\sum_{n=0}^{N-1}\mathrm{Im}\{x(n)\}\cos(k\Omega n) - \sum_{n=0}^{N-1}\mathrm{Re}\{x(n)\}\sin(k\Omega n)\right)$$

Vastaavasti IDFT reaali- ja imaginääriosiin jaettuna:

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)\mathrm{e}^{\mathrm{j}k\Omega n} =$$

$$\frac{1}{N}\left(\sum_{k=0}^{N-1}\mathrm{Re}\{X(k)\}\cos(k\Omega n) - \sum_{k=0}^{N-1}\mathrm{Im}\{X(k)\}\sin(k\Omega n)\right) + \mathrm{j}\frac{1}{N}\left(\sum_{k=0}^{N-1}\mathrm{Im}\{X(k)\}\cos(k\Omega n) + \sum_{k=0}^{N-1}\mathrm{Re}\{X(k)\}\sin(k\Omega n)\right)$$

Toteutetaan Javalla metodit, jotka laskevat DTF:n ja IDFT:n:

```
/** Janne Koljonen
Vaasan yliopisto
AUTO1030 */

import java.text.*;
```

Janne Koljonen
Vaasa yliopisto
16.3.2012
AUTO1030 Signaalien käsittely
Harjoitus 7, tehtävät 6–7

```
public class Fourier
{
   // Laskee diskreetin Fourier-muunnoksen annetulle kompleksiselle aikasarjalle.
   // Lisäksi parameterina annetaan viittaukset taulukoihin, joihin tulos (Re{X(k)} ja Im{X(k)})
kirjoitetaan
   public static void transform(double[] Rexn, double[] Imxn, double[] ReXk, double[] ImXk)
   {
      double omega = 2*Math.PI/Rexn.length;

      // k:n silmukka eli eri Fourier-komponenttien laskenta
      for (int k=0; k<ReXk.length; k++)
      {
         // Kaikki summaukset tietyllä k-arvolla voidaan nyt tehdä samassa for-silmukassa
         for (int n=0; n<Rexn.length; n++)
         {
            // Oletetaan, että aluksi ReXk[k] = 0. Lisätään siihen reaaliosien summaukset
            ReXk[k] += Rexn[n]*Math.cos(k*omega*n) + Imxn[n]*Math.sin(k*omega*n);

            // Oletetaan, että aluksi ImXk[k] = 0. Lisätään siihen imaginääriosien summaukset
            ImXk[k] += Imxn[n]*Math.cos(k*omega*n) - Rexn[n]*Math.sin(k*omega*n);
         }
      }
   }

   // Laskee diskreetin käänteis-Fourier-muunnoksen annetulle kompleksiselle Fourier-
muunnossarjalle.
   // Lisäksi parameterina annetaan viittaukset taulukoihin, joihin tulos kirjoitetaan
   public static void inverse(double[] ReXk, double[] ImXk, double[] Rexn, double[] Imxn)
   {
      double omega = 2*Math.PI/ReXk.length;

      // n:n silmukka eli eri aikasarjaelementtien laskenta
      for (int n=0; n<Rexn.length; n++)
      {
         // Kaikki summaukset tietyllä n-arvolla
         for (int k=0; k<ReXk.length; k++)
         {
            // Oletetaan, että aluksi Rexn[n] = 0. Lisätään siihen reaaliosien summaukset
            Rexn[n] += ReXk[k]*Math.cos(k*omega*n) - ImXk[k]*Math.sin(k*omega*n);

            // Oletetaan, että aluksi Imxn[n] = 0. Lisätään siihen imaginääriosien summaukset
            Imxn[n] += ImXk[k]*Math.cos(k*omega*n) + ReXk[k]*Math.sin(k*omega*n);
         }
         // Muistetaan vielä jakaa N:llä
         Rexn[n] /= ReXk.length;
         Imxn[n] /= ImXk.length;
      }
   }

   public static void print(double[] Rex, double[] Imx)
   {
      // Pyöristetään tulokset tasan kahteen desimaaliin
      NumberFormat nf=NumberFormat.getInstance();
      nf.setMinimumFractionDigits(2);
      nf.setMaximumFractionDigits(2);

      for (int i=0; i<Rex.length; i++)
      {
         System.out.print(nf.format(Rex[i])+" + "+nf.format(Imx[i])+"j");
         System.out.println();
      }
   }

   // Pääohjelmametodi: testaa DFT-metodia
   public static void main(String args[]) throws Exception
   {
      // Kirjoittaa
      System.out.println("DFT");


      // Testiaikasarjat
      double[] Rexn1={-1, 0, 0, 1};
      double[] Imxn1={0, 0, 0, 0};
      double[] ReXk1=new double[Rexn1.length];
      double[] ImXk1=new double[Imxn1.length];
```

```
      // DFT
      transform(Rexn1, Imxn1, ReXk1, ImXk1);

      // Tulostetaan
      System.out.println("X(k)");
      print(ReXk1, ImXk1);

      // Lasketaan käänteismuunnos edellisen tulokselle
      // Pitäisi tulla alkuperäinen aikasarja (liukulukulaskennan takia voi tulla pieniä eroja!)
      double[] Rexn2=new double[Rexn1.length];
      double[] Imxn2=new double[Imxn1.length];
      inverse(ReXk1, ImXk1, Rexn2, Imxn2);

      // Tulostetaan
      System.out.println("x(n)");
      print(Rexn2, Imxn2);



      // Testataan vielä kompleksisen aikasarjan tapauksessa (tehtävän 5 luvuilla)
      double[] ReXk3={1, 0, 0, 1};
      double[] ImXk3={0, 1, 1, 0};
      double[] Rexn3=new double[ReXk3.length];
      double[] Imxn3=new double[ImXk3.length];

      // Nyt ensin IDFT
      inverse(ReXk3, ImXk3, Rexn3, Imxn3);

      // Tulostetaan
      System.out.println("");
      System.out.println("x3(n)");
      print(Rexn3, Imxn3);

      // Tarkistetaan laskemalla DFT
      double[] ReXk4=new double[Rexn3.length];
      double[] ImXk4=new double[Imxn3.length];
      transform(Rexn3, Imxn3, ReXk4, ImXk4);

      // Tulostetaan
      System.out.println("Xk4(k)");
      print(ReXk4, ImXk4);


   }
}
```

Tulos:

```
DFT
X(k)
0,00 + 0,00j
-1,00 + 1,00j
-2,00 + -0,00j
-1,00 + -1,00j
x(n)
-1,00 + -0,00j
0,00 + -0,00j
-0,00 + -0,00j
1,00 + -0,00j

x3(n)
0,50 + 0,50j
-0,00 + -0,50j
0,00 + 0,00j
0,50 + -0,00j
Xk4(k)
1,00 + 0,00j
-0,00 + 1,00j
0,00 + 1,00j
1,00 + -0,00j
```