Janne Koljonen
Vaasa yliopisto
1.3.2012
AUTO1030 Signaalien käsittely
Harjoitus 4, tehtävä 2–3

Diskreetti Fourier-muunnos

Diskreetti Fourier-muunnos saadaan kaavalla:

$$X_k = \sum_{n=0}^{N-1} x_n \mathrm{e}^{-i2\pi\frac{k}{N}n},$$

jossa x_n on diskreetti signaali (aikasarja) ja X_k on aikasarjan diskreetti Fourier-muunnos ja kokonaisluku $k = \{0, 1, \ldots, N-1\}$. on harmoninen taajuus.

Tietyllä taajuudella k diskreetti Fourier-muunnos on siis pistetulo aikasarjan x_n ja kantafunktion (vektorin) $\mathrm{e}^{-i2\pi\frac{k}{N}n}$ välillä. Jos esimerkiksi $N = 3$ ja $x_n = \{1, -1, 1\}$, on kantafunktio taajuudella $k = 0$: $\left\{\mathrm{e}^{-i2\pi\frac{0}{3}0}, \mathrm{e}^{-i2\pi\frac{0}{3}1}, \mathrm{e}^{-i2\pi\frac{0}{3}2}\right\} = \{1,1,1\}$. Vastaava pistetulo on: $\begin{pmatrix}1 & -1 & 1\end{pmatrix}\begin{pmatrix}1\\1\\1\end{pmatrix} = 1\cdot 1 + (-1)\cdot 1 + 1\cdot 1 = 1.$

Kun taajuus $k = 1$, kantafunktio on: $\left\{\mathrm{e}^{-i2\pi\frac{1}{3}0}, \mathrm{e}^{-i2\pi\frac{1}{3}1}, \mathrm{e}^{-i2\pi\frac{1}{3}2}\right\} = \left\{1, \mathrm{e}^{-i\frac{2}{3}\pi}, \mathrm{e}^{-i\frac{4}{3}\pi}\right\}$. Jne.

Eulerin kaavalla saadaan Fourier-muunnoksen kantafunktio muunnettua karteesiseen muotoon:

$$X_k = \sum_{n=0}^{N-1}\left[x_n \cos\left(-2\pi\frac{k}{N}n\right) + i\sin\left(-2\pi\frac{k}{N}n\right)\right] =$$
$$\sum_{n=0}^{N-1} x_n \cos\left(-2\pi\frac{k}{N}n\right) + i\sum_{n=0}^{N-1} x_n \sin\left(-2\pi\frac{k}{N}n\right)$$

Havaitaan, että diskreetissä Fourier-muunnoksessa lasketaan pistetuloa aikasarjan x_n sekä kosini- ja sinifunktioiden välillä. Pistetulo on kertoo (oleellisesti) vektorin pituuden projisoituna annetulla kantavektorille. Diskreetissä Fourier-muunnoksessa projisoidaan siis aikasarjaa eritaajuisille kosini- ja sinifunktioille.

Kun projektio molemmille on laskettu, voidaan laskea aikasarjan x_n eri taajuuskomponenttien amplitudi ja vaihekulma suhteessa nollakulmaan, joka on kosifunktion vaihekulma. Amplitudi saadaan Pythagoraan lauseella, ja se on yhtä kuin kompleksiluku X_k:n itseisarvo. Vaihekulma saadaan arcustangentilla, ja se on yhtä kuin kompleksiluku X_k:n vaihekulma.

Esimerkiksi jos taajuuskomponentti on 90 astetta kosinifunktiota edellä, sen projektio vastaavantaajuiselle kosinifunktiolle on 0 ja projektio vastaavantaajuiselle sinifunktiolle on X_k.. Jos taajuuskomponentti on 90 astetta kosinifunktiota jäljessä, sen projektio vastaavantaajuiselle kosinifunktiolle on myös 0 ja projektio vastaavantaajuiselle sinifunktiolle on $-X_k$. Koska kosini- ja sinifunktiot ovat tällä tavalla ortogonaalisia ja kompleksitason reaali- ja imaginääriakselit ovat vastaavasti ortogonaalisia (kohtisuoria), voidaan kosini- ja sinifunktioita ja edelleen Eulerin kaavan avulla vastaavaa eksponenttiesitystä kätevästi käyttää aikasarjan taajuusanalyysissä.

Diskreetin Fourier-muunnoksen ohjelmointi matriisitulona

Jos haluamme käyttää reaalilukulaskentaa, käytämme muotoa, jossa eksponenttifunktio on hajotettu kosini- ja sinifunktioiksi. Laskemme siis diskreetin Fourier-muunnoksen reaali- ja imaginääriosat erikseen.

Aiemmin olemme ohjelmoineet matriisitulon. Nyt siis riittää muodostaa kantafunktiomatriisit, kun aikasarjan pituus N on annettu. Kantafunktiomatriisissa kukin rivi vastaa yhtä taajuuskomponenttia k.

Tehdään Matlabilla testifunktioita:

```
>> xn=[1; -1; 1]

xn =

     1
    -1
     1

>> fft(xn)

ans =

   1.0000
   1.0000 + 1.7321i
   1.0000 - 1.7321i
```

Java-koodi (lisää itse testitapauksia ja automatisoi testaus ja virheilmoitukset):

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

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


      // Testiaikasarjat
      double[] xn1={1.0, -1.0, 1.0};
      // Tee vielä neljä testiä ja niiden tulokset
      // Oletetetut tulokset {{Re}, {Im}}
      double[][] Xk1_check={{1.0, 1.0, 1.0}, {0.0, 1.7321, -1.7321}};

      // Lasketaan
      double[][] Xk1_computed=DFT(xn1);
      // Tarkistetaan
      tarkista(Xk1_computed, Xk1_check);
   }

   // Laskee diskreetin Fourier-muunnoksen annetulle aikasarjalle
   public static double[][] DFT(double[] xn)
   {
      double[][] temp=new double[2][xn.length];

      // Muunnetaan xn matriisiksi, jossa Vain yksi sarake
      double[][] temp2=new double[xn.length][1];
      for (int i=0; i<xn.length; i++)
      {
         temp2[i][0]=xn[i];
      }

      // Kantafunktiot
```

```
      double[][] W_cos=new double[xn.length][xn.length];
      double[][] W_sin=new double[xn.length][xn.length];

      // for loop riveille (= 1. taulukkoindeksi)
      for (int k=0; k<xn.length; k++)
      {
         // for loop sarakkeille
         for (int n=0; n<xn.length; n++)
         {
            W_cos[k][n]=Math.cos(-2*Math.PI*k*n/xn.length);
            W_sin[k][n]=Math.sin(-2*Math.PI*k*n/xn.length);
         }
      }

      double[][] temp3=tulo(W_cos, temp2);
      double[][] temp4=tulo(W_sin, temp2);

      // Yhdistetään cos ja sin yhteen taulukkoon
      for (int i=0; i<xn.length; i++)
      {
         temp[0][i]=temp3[i][0];
         temp[1][i]=temp4[i][0];
      }

      return temp;
   }

   // Matriisitulo
   public static double[][] tulo(double[][] X1, double[][] X2)
   {
      double[][] temp=new double[X1.length][X2[0].length];

      for (int rivi=0; rivi<temp.length; rivi++)
      {
         for (int sarake=0; sarake<temp[rivi].length; sarake++)
         {
            // Pistetulo
            for (int i=0; i<temp.length; i++)
            {
               temp[rivi][sarake]+=X1[rivi][i]*X2[i][sarake];
            }
         }
      }
      return temp;
   }

   // Verrataan kahta taulukkoa
   public static void tarkista(double[][] X1, double[][] X2)
   {
      for (int rivi=0; rivi<X1.length; rivi++)
      {
         for (int sarake=0; sarake<X1[rivi].length; sarake++)
         {
            // Sallitaan pieni pyöristysvirhe: 1 %
            if ( Math.abs(X1[rivi][sarake]-X2[rivi][sarake]) > Math.abs(0.01*X2[rivi][sarake]) )
            {
               System.out.println("Ero kohdassa: ("+rivi+", "+sarake+"): on "+X1[rivi][sarake]+" ,
p.o. "+X2[rivi][sarake]);
            }
         }
      }
   }


   public static void print(double[][] X)
   {
      for (int rivi=0; rivi<X.length; rivi++)
      {
         for (int sarake=0; sarake<X[rivi].length; sarake++)
         {
            System.out.print(X[rivi][sarake]+", ");
         }
         System.out.println();
      }
   }

}
```