L'algorisme FFT

Lluís Alsedà

Departament de Matemàtiques
Universitat Autònoma de Barcelona

http://www.mat.uab.cat/∼alseda

30 de gener de 2009

UAB
Universitat Autònoma de Barcelona
DEPARTAMENT DE MATEMÀTIQUES

Els coeficients de Fourier

Volem calcular els *coeficients de Fourier* de la transformada discreta:

$$c_j = \frac{1}{N}\sum_{k=0}^{N-1} a_k w^{jk} \quad j = 0, 1, \ldots, N-1 \tag{1}$$

on $a_k = f\left(\frac{2k\pi}{N}\right)$ i $w = \exp\left(\frac{-2\pi i}{N}\right)$.

La DFT

El càlcul directe dels coeficients (1) pel mètode de Horner es coneix com a `DFT`. A la següent transparència es mostra un programa que ho implementa.

Les dades (d'entrada) es guarden a dos vectors de mida N:

`Ref[0],...,Ref[N-1]` i `Imf[0],...,Imf[N-1]`.

`Ref[k]` conté la part real de $f\left(\frac{2k\pi}{N}\right)$ i `Imf[k]` la part imaginaria.

Els coeficients calculats (de sortida) es guarden a dos vectors de mida N:

`ReC[0],...,ReC[N-1]` i `ImC[0],...,ImC[N-1]`.

Com abans, `ReC[k]` conté la part real de c_k i `ImC[k]` la part imaginaria.

Una implementació de la DFT

```c
#include <math.h>
#define DOSPI (double) 6.28318530717959

void DFT(double *Ref, double *Imf, unsigned long N,  double *ReC, double *ImC){
  int k, j, NM1=N-1, NM2=N-2;
  double ReWj=1.0, ImWj=0.0, ReW, ImW, aux = - DOSPI/N;
/*  w = ReW + i ImW */
  ReW = cos(aux); ImW = sin(aux);

/* Calcul dels coeficients per Horner */
  for(j=0 ; j < N ; j++){ ReC[j]=Ref[NM1]; ImC[j]=Imf[NM1];
      for(k=NM2 ; k >= 0 ; k--){
           aux=ReC[j]*ReWj - ImC[j]*ImWj + Ref[k];
           ImC[j]=ReC[j]*ImWj + ImC[j]*ReWj + Imf[k];
           ReC[j]=aux;
      } ReC[j]=ReC[j]/N; ImC[j]=ImC[j]/N; // Normalitzacio
/*     Actualitzem w^j = ReWj + i ImWj */
      aux=ReWj*ReW - ImWj*ImW; ImWj=ReWj*ImW + ImWj*ReW; ReWj=aux;
  }
}
```

L'algorisme FFT

Es basa en fer el càlcul dels coeficients de Fourier de la forma següent:

$$c_j = \frac{1}{N}\left(\varphi(j_1) + w^j \psi(j_1)\right), \quad j = 0, 1, \ldots, N-1$$

on $j_1 \equiv j \pmod{N/2}$ i

$$\varphi(j) = \sum_{k=0}^{N/2-1} a_{2k}(w^2)^{jk} \qquad \psi(j) = \sum_{k=0}^{N/2-1} a_{2k+1}(w^2)^{jk}$$

amb $j = 0, 1, \ldots, N/2-1$.

Interpretació

Els coeficients de Fourier d'N punts es calculen a partir dels $N/2$ coeficients de Fourier *sense normalització* dels dos senyals que consisteixen en els punts parells del senyal original: $\{a_{2k}\}_{k=0}^{n/2-1}$ i els punts senars del senyal original: $\{a_{2k+1}\}_{k=0}^{n/2-1}$.

L'algorisme FFT (continuació)

Nota

Quan $N = 1$ tenim

$$c_0 = a_0.$$

Per tant, si apliquem recursivament L'estratègia descrita, quan arribem als problemes d'un punt (n'hi ha N), el càlcul de l'únic coeficient de Fourier de cada problema és trivial.

En particular, per a poder portar a terme aquesta recursivitat amb facilitat,

suposarem que N és potència de 2.

Cal separar adequadament els punts parells i senars *recursivament* a totes les etapes. Això ho veurem més clarament amb un exemple per una N concreta.

Exemple de separació recursiva de punts ($N = 16$)

senyals × punts	Distribució de punts i senyals
1×16	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
2×8	0 2 4 6 8 10 12 14 | 1 3 5 7 9 11 13 15
4×4	0 4 8 12 | 2 6 10 14 | 1 5 9 13 | 3 7 11 15
8×2	0 8 | 4 12 | 2 10 | 6 14 | 1 9 | 5 13 | 3 11 | 7 15
16×1	0 | 8 | 4 | 12 | 2 | 10 | 6 | 14 | 1 | 9 | 5 | 13 | 3 | 11 | 7 | 15

La primera fase divideix el senyal de $N = 16$ punts en dues senyals de $N/2 = 8$ punts cadascuna. La segona fase descompon les dades a quatre senyals de $N/4 = 4$ punts. Aquest patró continua fins que hi hagi N senyals de d'un únic punt cadascuna.

Ordenació del vector final: lexicogràficament segons l'expressió de l'index del vector de dades en binari

Índexs del vector de dades en l'ordre usual.

Decimal	Binari
0	0000
1	1000
2	0100
3	1100
4	0010
5	1010
6	0110
7	1110
8	0001
9	1001
10	0101
11	1101
12	0011
13	1011
14	0111
15	1111

Índexs del vector de dades ordenats lexicogràficament en binari.

Decimal	Binari
0	0000
8	0001
4	0010
12	0011
2	0100
10	0101
6	0110
14	0111
1	1000
9	1001
5	1010
13	1011
3	1100
11	1101
7	1110
15	1111

Implementació de la FFT

Com abans, les dades d'entrada es guarden a dos vectors de dimensió N:

`Ref[0],...,Ref[N-1]` i `Imf[0],...,Imf[N-1]`

on `Ref[k]` conté la part real de $f\left(\frac{2k\pi}{N}\right)$ i `Imf[k]` la part imaginaria.

Algorisme d'ordenació

```
unsigned long N2=N/2, N1=N-1;
register unsigned long i, j = N2, b;

for (i=1; i<N2; i++){
    if (j > i){ SWAP(Ref[j],Ref[i]); SWAP(Imf[j],Imf[i]);
       if((j/2)<(N/4)){ SWAP(Ref[N1-j],Ref[N1-i]); SWAP(Imf[N1-j],Imf[N1-i]); }
    } ; b=N2;
    while (b >= 2 && j >= b){ j -= b; b /= 2; }
    j += b;
}
```

Fases de la ordenació per $N = 16$

i	1	2	3	4	5	6	7
j	8	4	12	2	10	6	14
Swap i $\leftrightarrow$ j?	S	S	S	N	S	N	S
Swap N-i-1 $\leftrightarrow$ N-j-1?	N	S	N	—	N	N	N

i=0	i=1	i=2	i=3	i=5	i=7
0: 0000	0: 0000	0: 0000	0: 0000	0: 0000	0: 0000
1: 1000	8: 0001	8: 0001	8: 0001	8: 0001	8: 0001
2: 0100	2: 0100	4: 0010	4: 0010	4: 0010	4: 0010
3: 1100	3: 1100	3: 1100	12: 0011	12: 0011	12: 0011
4: 0010	4: 0010	2: 0100	2: 0100	2: 0100	2: 0100
5: 1010	5: 1010	5: 1010	5: 1010	10: 0101	10: 0101
6: 0110	6: 0110	6: 0110	6: 0110	6: 0110	6: 0110
7: 1110	7: 1110	7: 1110	7: 1110	7: 1110	14: 0111
8: 0001	1: 1000	1: 1000	1: 1000	1: 1000	1: 1000
9: 1001	9: 1001	9: 1001	9: 1001	9: 1001	9: 1001
10: 0101	10: 0101	10: 0101	10: 0101	5: 1010	5: 1010
11: 1101	11: 1101	13: 1011	13: 1011	13: 1011	13: 1011
12: 0011	12: 0011	12: 0011	3: 1100	3: 1100	3: 1100
13: 1011	13: 1011	11: 1101	11: 1101	11: 1101	11: 1101
14: 0111	14: 0111	14: 0111	14: 0111	14: 0111	7: 1110
15: 1111	15: 1111	15: 1111	15: 1111	15: 1111	15: 1111

Implementació de la FFT: Fase de càlcul dels coeficients de Fourier

Etapes

Una etapa consisteix a passar dels coeficients de Fourier de $\frac{N}{\mathtt{np}}$ problemes de `np` punts als de $\frac{N}{2*\mathtt{np}}$ problemes de $2 * \mathtt{np}$ punts. Per tant `np` prendrà els valors $1, 2, \ldots, \frac{N}{2}$.

`np` és el número d'*etapa*.

Nota

La FFT és una rutina que treballa *in place*. Això vol dir que fa servir la mateixa memòria de l'entrada per a emmagatzemar les dades de *totes* les etapes de càlcul. Es a dir, a cada etapa es sobreescriuen les dades inicials amb les dades finals. En particular, a la sortida de la funció, `Ref[k]` conté la part real de c_k i `Imf[k]` la part imaginaria.

A cada etapa pensem que el vector de dades esta dividit en $\frac{N}{2*\texttt{np}}$ blocs de punts:

$$f[b], f[b+1], \ldots, f[b+\texttt{np}-1],$$
$$f[b+\texttt{np}], f[b+\texttt{np}+1], \ldots, f[b+2*\texttt{np}-1],$$

on $b = \overline{b}*2*\texttt{np}$ i $\overline{b}$ és el número de bloc. A l'inici de l'etapa cada punt és un dels np coeficients de Fourier d'un problema de Fourier *conegut* de np punts (el bloc conté dos d'aquests conjunts de coeficients).

Objectiu de l'etapa

Al finalitzar l'etapa cada punt ha de ser un coeficient de Fourier d'un problema *nou* de 2*np punts.

El nombre w de l'etapa

El nombre w corresponent a l'etapa $\texttt{np}$ es defineix com

$$w = \exp\left(-\frac{2 * \pi}{2 * \texttt{np}}\right) = \exp\left(-\frac{\pi}{\texttt{np}}\right).$$

Observem que $w^{\texttt{np}} = \exp(-\pi) = -1$ i, llavors, $w^{\texttt{np}+j} = -w^j$ per $j = 0, 1, \ldots, \texttt{np} - 1$.

Degut a l'ordenació inicial del vector, podem pensar que els primers np punts del bloc són els coeficients φ mentres que el segon conjunt de np punts són els coeficients ψ (ordenats tots dos grups en ordre creixent d'índexs). Es a dir, per a cada b, podem reinterpretar el bloc com:

$$f[b] = \varphi(0), f[b+1] = \varphi(1), \ldots, f[b+\mathtt{np}-1] = \varphi(\mathtt{np}-1)$$
$$f[b+\mathtt{np}] = \psi(0), f[b+\mathtt{np}+1] = \psi(1), \ldots,$$
$$f[b+2*\mathtt{np}-1] = \psi(\mathtt{np}-1).$$

Observem que per a cada $b = 0, 2*\mathtt{np}, \ldots, N-2*\mathtt{np}$ i per a cada $j = 0, 1, \ldots, \mathtt{np}-1$, el càlcul de c_j i $c_{\mathtt{np}+j}$ solament depèn de

$$f[b+j],\ f[b+\mathtt{np}+j],\ w^j \text{ i } w^{\mathtt{np}+j} = -w^j.$$

Per tant, per a evitar repeticions en el càlcul de w^j, podem realitzar els càlculs de tots els blocs al mateix temps.

Per a això s'introdueix la notació

$$\texttt{jj} = b + j, \text{i}$$
$$\texttt{jjrm} = \texttt{jj} + \texttt{np} = b + \texttt{np} + j.$$

Així, la realització de l'etapa $\texttt{np}$ consisteix a recórrer totes les $j = 0, 1, \ldots, \texttt{np} - 1$ i, per a cada una d'aquestes j, calcular (iterativament) w^j i

$$f[\texttt{jjrm}] = \varphi(j) + w^{\texttt{np}+j}\psi(j) = f[\texttt{jj}] - w^j * f[\texttt{jjrm}]$$
$$f[\texttt{jj}] = \varphi(j) + w^j\psi(j) = f[\texttt{jj}] + w^j * f[\texttt{jjrm}]$$

per $b = 0, 2 * \texttt{np}, \ldots, N - 2 * \texttt{np}$.

Observació sobre el càlcul recursiu de w^j

El càlcul de

$$w^j = \cos(-j * \pi/\mathtt{np}) + i * \sin(-j * \pi/\mathtt{np}) = \mathtt{ReWj} + i * \mathtt{ImWj}$$

es realitza recursivament usant les fórmules de Stoer-Burlisch pp. 24 i 25:

Inicialització

```
ReWj = 1.0 ; ImWj = 0.0 ;
theta=isign*((double) 3.14159265358979323 85)/np;
ds=sin(theta);
```

(recordem que $w^0 = 1$). A més inicialitzem,

$$\mathtt{dc} = \cos(\mathtt{theta}) - 1 \quad \text{i} \quad \mathtt{t} = -4 * \sin^2(\mathtt{theta}/2) :$$

```
aux=sin(theta/2.0); dc=-2.0*aux*aux; t=2.0*dc;
```

Algorisme recurrent per al calcul de ReWj i ImWj

```
ReWj = ReWj + dc; dc = t*ReWj + dc;
ImWj = ImWj + ds; ds = t*ImWj + ds;
```

Codi complet de la FFT

A les transparències següents donarem el codi complet de la FFT. Recordem que N és potència de dos. A més, usarem el paràmetre `isign` que val -1 si es vol fer la transformada directa i 1 si es vol fer la inversa.

Nota

En el codi, per seguretat es comprova que `N` sigui potència de 2 i que `isign` només pugui tenir els valors legals de 1 o -1. El test `is_power_of_two` usa fortament l'aritmètica de bits.

Codi de la FFT: Entrada i ordenació

```
#include <math.h>
#include <stdio>
#define SWAP(a,b) aux=(a);(a)=(b);(b)=aux

unsigned is_power_of_two (unsigned long n) {
    if (!n || (n & (n - 1))) return 0; else return 1;
}

void FFT(double *Ref, double *Imf, unsigned long N, char isign){
  unsigned long N2=N/2, N1=N-1;
  register unsigned long i, j = N2, b, np;
  double ReWj, ImWj, aux;

  if(N < 2 || !is_power_of_two(N)){
     printf("\nError: N no es una potencia de 2 legal.\nAvortant...\n\n");
     return ;
  }
  if(isign != 1 && isign != -1){
     printf("\nError: 'isign' ha de ser 1 o -1.\nAvortant...\n\n");
     return ;
  }

  for (i=1; i<N2; i++){
      if (j > i){ SWAP(Ref[j],Ref[i]); SWAP(Imf[j],Imf[i]);
         if((j/2)<(N/4)){ SWAP(Ref[N1-j],Ref[N1-i]); SWAP(Imf[N1-j],Imf[N1-i]); }
      } ; b=N2;
      while (b >= 2 && j >= b){ j -= b; b /= 2; }
      j += b;
  }
```

Codi de la FFT: Càlcul dels coeficients de Fourier

```
  for(np=1 ; np <= N2; np*=2) { double dc, ds, t; unsigned long np2 = 2*np;
     aux=isign*((double) 3.14159265358979323851)/np;
     /* NOTA: dc = cos(aux) -1 */
     ds=sin(aux); aux=sin(aux/2.0); dc=-2.0*aux*aux; t=2.0*dc;
     ReWj = 1.0 ; ImWj = 0.0 ; // W^0 = 1 + i*0
     for(j=0 ; j < np ; j++){ unsigned long jj, jjnp;
         for(jj=j ; jj < N ; jj+=np2){ double ReT, ImT;
             jjnp = jj + np;

             ReT=Ref[jjnp]*ReWj - Imf[jjnp]*ImWj;
             ImT=Ref[jjnp]*ImWj + Imf[jjnp]*ReWj;
             Ref[jjnp]=Ref[jj] - ReT;
             Imf[jjnp]=Imf[jj] - ImT;
             Ref[jj]=Ref[jj] + ReT;
             Imf[jj]=Imf[jj] + ImT;
         } // Fi del calcul de c_j, c_{np+j}, c_{2*np+j}, c_{3*np+j} ...
/* Calcul de W^{j+1} = cos(isign*(j+1)*Pi/np) + i sin(isign*(j+1)*Pi/np)
                     = ReWj + i ImWj
   usant la recurrencia de Stoer-Burlisch pp. 24 i 25. */
         ReWj = ReWj + dc; dc = t*ReWj + dc;
         ImWj = ImWj + ds; ds = t*ImWj + ds;
     } // Fi del calculs dels coeficients agrupats en problemes de 2*np punts
  } // Fi del calcul dels coeficients de fourier
  for(j=0 ; j < N ; j++) { Ref[j] /= N ; Imf[j] /= N ; } // Normalitzacio
}
#undef SWAP
```