IMPLEMENTACION DE UN ALGORITMO FFT EN APL

F.L. Valer.- Centro de Investigación UAM-IBM

INTRODUCCION

En el presente trabajo se expone brevemente un algoritmo FFT así como su posterior implementación en APL, realizada en el año 1972 en este Centro de Investigación.

Se comienza por dar una definición de la transformada de Fourier discreta, pasándose a describir el algoritmo FFT utilizado y su programa asociado. La última sección se ocupa de los tiempos de CPU, referentes al programa presentado, comparándose con el de uno que utiliza directamente la definición del producto complejo, y con otro que es simple aplicación de la definición de la transformada Fourier discreta.

1 - TRANSFORMADA FOURIER DISCRETA

Consideremos una colección de N números complejos definidos por un parámetro k (k = 0,1, ... N-1); designémoslos genéricamente por X_k; por definición la componente n-sima de su transformada Fourier discreta es :

$$A_n = \sum_{k=0}^{N-1} X_k \exp(-2\pi\imath nk/N) \qquad n = 0, 1, ... N-1 \qquad (1)$$

Puede demostrarse, ver por ejemplo (7), que la inversa de esta relación es la :

$$X_k = \frac{1}{N} \sum_{n=0}^{N-1} A_n \exp(2\pi\imath nk/N) \qquad k = 0, 1, ... N-1 \qquad (2)$$

2 - EL ALGORITMO FAST FOURIER TRANSFORM.

Notemos que si aplicamos directamente la definición (1) deberíamos hacer N^2 multiplicaciones complejas para calcular la transformada Fourier discreta (D.F.T.).

Con el nombre de Fast Fourier Transform, se agrupa una familia de algoritmos para el cálculo de la D.F.T. de $N = 2^P$, P entero no nulo, números complejos, todos ellos basados en propiedades de simetría de la exponencial compleja y diferentes agrupaciones de la señal de entrada, reduciéndose el número de operaciones, con respecto a las correspondientes a la aplicación directa de la definición, a un número proporcional a $N \log_2 N$.

Como veremos (Figura 1) el sistema pasa por un número de niveles $P=\log_2 N$, en cada uno de los cuales realiza N multiplicaciones complejas; resulta evidente que el número de sumas tampoco ha aumentado.

Algunos autores sugieren que con el método de Golub, consistente en disminuir en la operación producto complejo el número de multiplicaciones en favor de un aumento del número de sumas reales [(a + ıb) (c + dı) = ((a + b) (c - d) + ad - bc) + ı(ad + bc)], se gana tiempo en la ejecución.

Los diferentes algoritmos tienen asociados diferentes diagramas, denominados mariposa; describiremos el utilizado en la implementación.

Este método, que es el usado en el programa TFCP, divide la secuencia X_k en dos subsecuencias :

$$X_k \qquad k = 0, \ldots \frac{N}{2} - 1$$

$$X_{k+\frac{N}{2}} \qquad k = 0, \ldots \frac{N}{2} - 1$$

Por la (1)

$$Ar = \sum_{k=0}^{\frac{N}{2}-1} [X_k \exp\left(\frac{-2\pi i r k}{N}\right) + X_{k+\frac{N}{2}} \exp\left(\frac{-2\pi i r (k+\frac{N}{2})}{N}\right)] =$$

$$= \sum_{k=0}^{\frac{N}{2}-1} X_k \exp\left(\frac{-2\pi i r k}{N}\right) + \sum_{k=0}^{\frac{N}{2}-1} X_{k+\frac{N}{2}} \exp\left(\frac{-2\pi i r k}{N}\right) \cdot$$

$$\cdot \exp(-\pi i r) = \sum_{k=0}^{\frac{N}{2}-1} X_k \exp\left(\frac{-2\pi i r k}{N}\right) + (-)^r \sum_{k=0}^{\frac{N}{2}-1} X_{k+\frac{N}{2}} \cdot$$

$$\cdot \exp\left(\frac{-2\pi i r k}{N}\right)$$

Consideremos las transformadas de índice par : por la fórmula precedente se tiene :

$$A_{2r} = \sum_{k=0}^{\frac{N}{2}-1} X_k \exp\left(\frac{-2\pi i r k}{N/2}\right) + \sum_{k=0}^{\frac{N}{2}-1} X_{k+\frac{N}{2}} \exp\left(\frac{-2\pi i r k}{N/2}\right) =$$

$$= \sum_{k=0}^{\frac{N}{2}-1} [X_k + X_{k+\frac{N}{2}}] \exp\left(\frac{-2\pi i r k}{N/2}\right)$$

Para las transformadas de índice impar

$$A_{2r+1} = \sum_{k=0}^{\frac{N}{2}-1} X_k \exp\left(\frac{-2\pi i r k}{N/2}\right) \exp\left(\frac{-2\pi i k}{N}\right) -$$

$$- \sum_{k=0}^{\frac{N}{2}-1} X_{k+\frac{N}{2}} \exp\left(\frac{-2\pi i r k}{N/2}\right) \exp\left(\frac{-2\pi i k}{N}\right) =$$

$$= \sum_{k=0}^{\frac{N}{2}-1} [(X_k - X_{k+\frac{N}{2}}) \exp\left(\frac{-2\pi i k}{N}\right)] \exp\left(\frac{-2\pi i r k}{N/2}\right)$$

Nuevamente reiterando el procedimiento P veces ($N = 2^P$) se llega al cálculo de transformadas Fourier de secuencias de dos puntos .

El diagrama mariposa asociado a esta idea es :

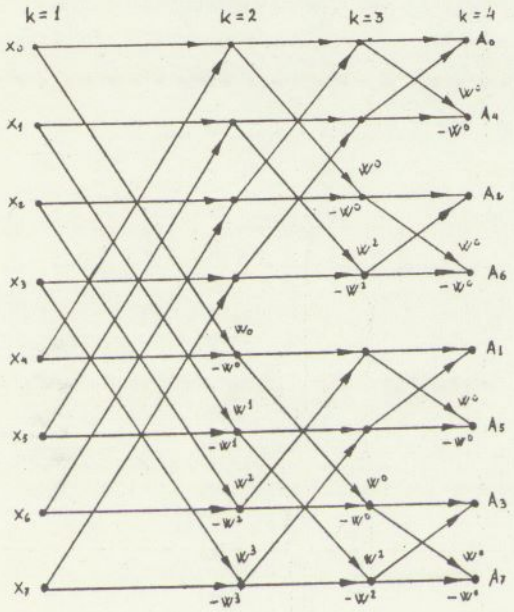


Fig. 1

Hay varios detalles interesantes en este diagrama, con relación a una computación posterior.

i) Juega un papel sustancial el peso W.

ii) Podemos recorrerlo con dos contadores, uno K =1, 2,....p para indicar el nivel en que estamos, otro

J = 1 .....N para indicar el término del nivel considerado.

iii) Toda flecha con dirección ascendente no tiene peso, así en el nivel K sólo aparecen pesos en las flechas que acaban en los 2^{P-K+1} últimos términos; en este sentido podemos decir que cada nivel está gobernado por el número $M = 2^{P-K+1}$.

3.- IMPLEMENTACION

Con la idea mencionada en el apartado anterior , y aprovechando que el algoritmo descrito para la directa será análogo al de la inversa, se hicieron dos programas TFCP, con el fin de comprobar cuál era más rápido, uno aplicando el método de Golub para realizar productos complejos (PC) y otro aplicando la definición del producto de números complejos (COMPL). Se vió que la rapidez estaba en relación directa con el ordenador utilizado, así con un IBM 360/40 era más rápido el TFCP con COMP, mientras con un IBM 360/50 lo era el TFCP con PC.

En este programa las sentencias (1), (2), (3), (4) y (5) son de tipo ilustrativo.

La (6) se ocupa de ver si la entrada es admisible para aplicar el algoritmo F.F.T., es decir, si el número de puntos es de la forma $N=2^P$ con p = entero no nulo; si esto no se verifica nos manda a la (21) y nos introduce el contador K que recorre todos los niveles del diagrama mariposa.

Las (7), (14) y (20) son indicadores para el tiempo de ejecución. Así TI es el origen de tiempos, TC el tiempo de cálculos y TT el de cálculos más el de ordenación de los coeficientes, en minutos, segundos y milisegundos.

Las (8) y (9) nos definen magnitudes de importancia sustancial en los cálculos, es decir, el vector M, y la matriz de pesos W.

La (10) nos define a Y como matriz de dimensión (2, N, 2), el primer 2 indica el número de hojas de la matriz (la primera es la parte real, la segunda la imaginaria). N el número de filas, y el segundo 2 el de columnas. Colocando la entrada X en el sitio conveniente, es decir, la parte real en la primera columna de la primera hoja, y la imaginaria en la primera columna de la segunda hoja.

Es de notar que al no ser necesario para el conocimiento de un nivel más que el procedente, renovamos continuamente la matriz Y, con la consiguiente ganancia en la capacidad de cálculos.

La (11) y (12) nos calculan los Y correspondientes al nivel en cuestión, teniendo en cuenta el diferente modo de cálculo de los N/2 primeros elementos y de los N/2 posteriores; debido a esto jugará un papel primordial la idea de existencia o no de pesos que podemos materializar por la condición:

No hay pesos si

$(M\,[(K-1)]\,|\,J) < M[K]$

donde $M = N(0, 1/2, 1/4, \ldots, 1/2^{p-1})$

y en este caso

$Y\,[;J;K] = Y\,[;J;(K-1)] + Y\,[;(J+M\,[K])\;;\;(K-1)]$

verificándose en el caso de que haya pesos que éste es para el elemento J simo del nivel K,

$W\,[;\,1 + (N \div M\,[(K-1)]) \times (M\,[K]\,|\,J-1)]$

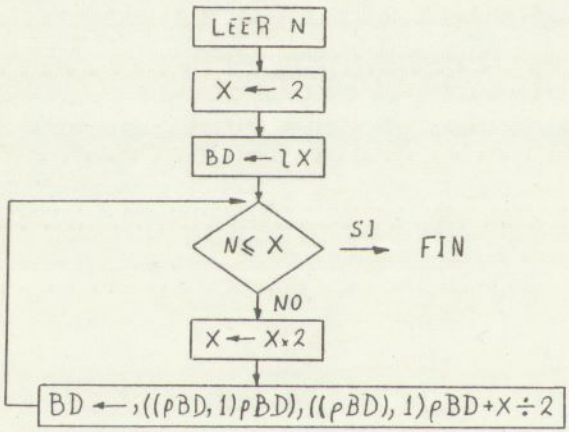
donde

$$W[\ ;\mu\ ] \equiv \cos\frac{2\pi(\mu-1)}{N} \underset{+}{-} \operatorname{sen}\frac{2\pi(\mu-1)}{N}$$

tomándose el signo - para la directa, y el + para la inversa.

La (13) nos hace volver el programa a (11), en el caso en que no se hayan recorrido todos los niveles.

Las (15), (16), (17) y (18) son sentencias de ordenación de los coeficientes que podemos describir con el organigrama siguiente :



Observemos que las variables X,BD al llegar a estas sentencias han cumplido su objetivo precedente, por lo que les asignamos nuevos valores, que ahora necesitamos, ganando posiciones de memoria.

La (19) nos da A que es una matriz de tres columnas y N filas, estando formada cada fila por el número de orden de la componente Fourier considerada,

su parte real y su parte imaginaria.

Y la (22) nos asigna a la salida el valor vacío,en el caso en que el programa no se haya ejecutado, por ser $N \neq 2P$, p entero.

Los tiempos involucrados son, para un ordenador 360/50

N	TIEMPO DE CALCULO (EN MIN.SEG.÷60SEG)		
2	0	0	24
4	0	0	46
8	0	1	9
16	0	1	45
32	0	2	45
64	0	4	36
128	0	8	33
256	0	16	53
512	0	35	0

TIEMPO TOTAL (ID)		
0	0	27
0	0	52
0	1	17
0	1	58
0	3	0
0	4	56
0	8	59
0	17	26
0	35	47

Mientras que con el FFT y la aplicación discreta de la definición producto de números complejos son :

N	TIEMPO DE CALCULO (EN MIN.SEG.÷60SEG)		TIEMPO TOTAL (ID)
2	0	0	24
4	0	0	47
8	0	1	13
16	0	1	46
32	0	2	46
64	0	4	40
128	0	8	37
256	0	17	11
512	0	35	31
	0	0	27
	0	0	53
	0	1	23
	0	1	58
	0	3	2
	0	5	2
	0	9	5
	0	17	47
	0	36	18

Por aplicación discreta de la definición son :

N	TIEMPO DE CALCULO (EN MIN.SEG.÷60SEG)		
2	0	0	11
4	0	0	17
8	0	0	35
16	0	1	54
32	0	7	22

Observamos que hay una clara ventaja de los programas basados en el algoritmo F.F.T., frente al basado en la aplicación directa de la definición, en cuanto a capacidad en el número de puntos de la entrada, para el área de 60 K utilizada, y en los tiempos de cálculo, a partir de 16 puntos.

```
      ∇TFCP[⎕]∇

    ∇ Y←S TFCP X;K;N;P;M;W;BD;TI
[1]   ⍝ PROGRAMA PARA EL CALCULO DE LA TRANSFORMADA DE FOURIER DE N=2*P (P ENTERO) PUNTOS
[2]   ⍝ EL PRIMER ARGUMENTO DE LA FUNCION PUEDE TOMAR LOS VALORES ¯1 Y 1, ¯1 PARA LA DIRECTA
[3]   ⍝ 1 PARA LA INVERSA.EL SEGUNDO ARGUMENTO ES LA FUNCION A TRANSFORMAR COMO MATRIZ 2×N,LA PRIMERA
[4]   ⍝ FILA ESTA FORMADA POR LAS COMPONENTES REALES,LA SEGUNDA POR LAS IMAGINARIAS.
[5]   ⍝ CENTRO DE INVESTIGACION UAM-IBM.
[6]    →(P≠⌊P←2⍟N←(⍴X)[K←2])/MU
[7]    TI←⎕AI[2]
[8]    M←N×2*1-⍳1+P
[9]    W← 2 1 ○.×3○(2÷N)×(⍳N÷2)-1
[10]   Y←X,[2.5] 0
[11] VA:Y[;X;1+2|K+1]←Y[;X;1+2|K]+Y[;(M[K]+X←(BD←(M[(K-1)]|¯1+⍳N)<M[K])/⍳N);1+2|⍳⍴]
[12]   Y[;X;1+2|K+1]←W[;1+(N÷M[(K-1)])×(M[K]|X-1)] PC Y[;(X-M[K]);(BD←1)+2|K]-Y[;X←((~BD)/⍳N);1+2|⍳⍴]
[13]   →((P+1)≥K←K+1)/VA
[14]   TC← 60 60 1000 ⊤(⎕AI[2])-TI
[15]   BD←⍳X←2
[16] LAB:→(X=N)/FIN
[17]   BD←,((L,1)⍴BD),((L←⍴BD),1)⍴BD+(X←X×2)÷2
[18]   →LAB
[19] FIN:A←((N,1)⍴¯1+⍳N),QY←Y[;BD;1+2|K]÷3+(N,1)
[20]   →(TT=TT← 60 60 1000 ⊤(⎕AI[2])-TI)/0
[21] MU:'TE RECUERDO QUE ESTE PROGRAMA SOLO CALCULA TRANSFORMADAS FOURIER PARA UN NUMERO DE PUNTOS N=2*P'
[22]   Y←⍳0
    ∇


      ∇PC[⎕]∇


    ∇ NO←PR PC SE
[1]   ⍝ PROGRAMA PARA EL CALCULO DEL PRODUCTO DE DOS SECUENCIAS DE NUMEROS COMPLEJOS
[2]   ⍝ POR EL METODO DE GOLUB,LAS ENTRADAS DEBEN DARSE COMO MATRICES 2×N
[3]   ⍝ LA PRIMERA FILA ESTA FORMADA POR LAS COMPONENTES REALES,LA SEGUNDA POR LAS IMAGINARIAS
[4]   ⍝ CENTRO DE INVESTIGACION UAM-IBM.
[5]    NO←(⍴SE)⍴(((+/[1] PR)×-/[1] SE)+AD-BC),(AD←PR[1;]×.×SE[2;])+BC←PR[2;]×SE[1;]
    ∇
```

BIBLIOGRAFIA

(1) COOLEY J.W., TURKEY J.W.
"An algorithm for the machine calculation of complex Fourier series"
MATH. COMPUT., vol. 19.
1965, pp. 297-301

(2) COCHRAN W.T., COOLEY J.W.
"What is the fast Fourier transform?"
IEEE TRANS. ON AUD. AND ELECT., Vol. AU-15, NO. 2.
JUNE 1967, pp. 45-55.

(3) COOLEY J.W., LEWIS P.A.
"The finite Fourier transform"
IEEE TRANS. ON AUD. AND ELECT., VOL. AU - 17, NO. 2.
JUNE 1969, pp. 77-85

(4) THEILHEIMER F.
"A matrix version of the fast Fourier transform"
IEEE TRANS. ON AUD. AND ELECT., Vol. AU - 17, NO. 2.
JUNE 1969, pp. 158-161.

(5) COOLEY,J.W., LEWIS P.A.W., WELCH P.D.
"The fast Fourier transform algorithm: Programming considerations in the calculation of sine, cosine and Laplace transforms".
J. SOUND VIB., Vol. 12, NO. 3.
1970, pp. 315-337

(6) HARTWELL J.W.
"A procedure for implementing the fast Fourier transform on small computers".
IBM J. RES. DEVELOP.
SEPTEMBER 1971, pp. 355-363

(7) VALER F.L.
"El algoritmo Fast Fourier Transform y algunas de sus aplicaciones"
CENTRO DE INVESTIGACION UAM-IBM
P1-02.73, AGOSTO 1973.