Ciencia y Tecnología **17** No.1(1993) 1-16

Un Programa para obtener la transformada discreta de Fourier de secuencias cronológicas muy largas

F. J. Soley
Centro de Investigaciones Geofísicas y
Escuela de Física, Universidad de Costa Rica
Academia Nacional de Ciencias
(recibido octubre 1992, aceptado marzo 1993)

Abstract: At the Center for Geophysical Research of the University of Costa Rica, we recently implemented a program that calculates the discrete Fourier Transform of very long time series and which runs in IBM personal computers and compatibles. By very long we mean series that occupy in RAM more than two 64 Kb segments but will fit within the 640 Kb of conventional memory. By eliminating resident programs it is possible to process series of about 70 000 real data points in double precision arithmetic. The program uses an algorithm originally proposed by R. C. Singleton and written in Fortran for mainframe computers. We ported his algorithm to C and modified it accordingly to run within the memory constraints of personal computers. The algorithm is known as a mixed radix algorithm because it allows the order of transform to have even and odd factors.

Subject headings: helium, phonons, Bose-Einstein condensation.

Resumen: Recientemente habilitamos en el Centro de Investigaciones Geofísicas de la Universidad de Costa Rica un programa que permite obtener la transformada discreta de Fourier de secuencias cronológicas muy largas en microcomputadoras personales IBM y compatibles. Por muy largas queremos decir que ocupan en memoria RAM más de dos segmentos de 64 Kb y que no exceden la memoria RAM disponible dentro de los 640 Kb convencionales. Eliminando los programas residentes es posible procesar una secuencia de 70.000 datos reales en precisión doble. El programa utiliza un algoritmo propuesto por R. C. Singleton escrito en Fortran para computadoras "mainframe". Este programa se reescribió en C y se modificó para que se pudiere accesar todo la memoria RAM disponible. El algoritmo se conoce como de factores mixtos, ya que permite que la longitud de la serie se pueda descomponer en factores pares e impares.

Encabezados de materia: helio, fonones, condensación de Bose-Einstein.

1. Introducción

En el estudio de posibles periodicidades en series cronológicas equiespaciadas, la longitud de la serie está determinada en primera instancia por el periodo de interés, ya que generalmente se recomienda que la secuencia abarque varios periodos para obtener resultados más confiables. Por otro lado, no es inusual que un "pico" espectral observado sea compuesto, es decir, que se deba a varias frecuencias muy cercanas entre sí. En este caso, es posible llegar a resolver esas frecuencias aumentando la longitud de la serie. Un ejemplo reciente de esta aplicación se puede encontrar en Soley y Gutiérrez (1992). Esos autores muestran el espectro de amplitud que comprende los componentes diurnos P1, S1 y K1 y los componentes semidiurnos T2, S2 y K2 del nivel del mar para 1, 5, 11, 17 y 21 años de datos horarios en Puerto Quepos, Costa Rica. Con un año de observaciones no es posible resolver los tres componentes, pero en las secuencias de mayor longitud éstos están completamente separados. El programa descrito en este artículo puede procesar secuencias de unos 70000 datos en precisión doble y por lo tanto no podría utilizarse en secuencias de una longitud como las del ejemplo anterior. Sí se podría utilizar para el estudio de periodicidades tales como las observadas entre 5 y 8 años en el nivel del mar en el Pacífico (Sturges, 1987) y en estaciones mareográficas en la India (Das y Radhakrishna, 1991). Como ejemplo fuera del area de la geofísica, podemos citar que en el campo de la acústica se obtienen secuencias de estas longitudes con sólo 6 segundos de grabación discretizados a 11 KHz.

En computadoras "mainframe" hay poca dificultad en obtener la transformada discreta de Fourier de secuencias de la longitud que estamos considerando. Sin embargo, en los países en desarrollo no es frecuente encontrar esas computadoras en Servicios Meteorológicos, Centros de Investigación, Observatorios Geofísicos, etc. En el caso de que esos lugares cuentan con computadoras, lo más probable es que sean microcomputadoras IBM o compatibles. En este tipo de microcomputadora sí es problemático procesar secuencias muy largas debido a la estructura segmentada en el direccionamiento de memoria del sistema operativo DOS. El lenguage TURBO C provee un mecanismo para accesar toda la memoria RAM disponible que a pesar de ser muy sencilla, no es utilizada usualmente. En este artículo explicamos como se puede accesar toda la memoria utilizando los punteros enormes (huge pointers).

El programa está basado en el algoritmo de factores múltiples para el cálculo de la transformada rápida de Fourier (Singleton, 1969). Este algoritmo fue realizado en el idioma Fortran para una computadora "mainframe". A pesar de que en la bibliografía especializada se encuentran múltiples versiones del algoritmo de la transformada rápida de Fourier el autor ha encontrado que la versión de Singleton reune características que todavía no han sido superadas por las versiones más modernas. Dos de sus características tienen especial relevancia para este artículo: permite que el orden de transformada pueda descomponerse en factores impares además de pares y calcula las funciones seno y coseno recursivamente . Resulta curioso que este programa sea poco conocido a pesar de su gran calidad. En este artículo se explica la transcripción al Turbo C y como se maneja toda la memoria RAM con los punteros enormes. Además se explica en detalle algunas características

del programa que en el artículo de Singleton apenas se mencionan. En particular, los algoritmos para el cálculo de la transformada rápida de Fourier para las raíces 2, 4, 3, 5 y número impar arbitrario se dan con todo detalle.

2. Direccionamiento de la memoria

Los microprocesadores de la serie 8086 funcionando en el modo real dentro del sistema operativo DOS direccionan la memoria en forma segmentada. En este método, se utiliza un segmento de dos bytes que fijan la dirección base de una partición de la memoria, y un desplazamiento, también de dos bytes, respecto al inicio del segmento. Con el desplazamiento se pueden accesar únicamente 64 Kb (65536 posiciones de memoria). Por lo tanto, las instrucciones para el manejo dinámico de la memoria usualmente restringen la longitud máxima de una variable al tamaño de un segmento o un poco menor a un segmento. Por ejemplo, las intrucciones GetMem y New del Turbo Pascal asignan memoria hasta un máximo de 65521 bytes (64 Kb - 15). La situación es la misma para las instrucciones malloc() y calloc() del Turbo C, pero no para farmalloc() y farcalloc() que permiten asignar porciones de memoria mayores a 64 Kb. Las dos funciones últimas devuelven un puntero largo (far pointer) consistente de segmento y desplazamiento. El apuntador largo se envuelve sobre sí mismo cuando al direccionar alguna posición en memoria el desplazamiento excede 65535 (es decir, la dirección apuntada por el desplazamiento es efectivamente dirección módulo 64 Kb). Para remediar esta situación, Turbo C define las punteros enormes que al igual que los largos consisten de segmento y desplazamiento, cada uno de dos bytes. Se diferencian en que incorporan una normalización del segmento. Esta normalización consiste en que el segmento direcciona los "párrafos" de memoria (divisiones de 16 bytes) y el desplazamiento los 16 bytes dentro de un párrafo. El segmento es actualizado automáticamente conforme se direccionan las posiciones sucesivas en memoria. De esta manera se evita el envolvimiento en sí mismo, al precio de una disminución en la velocidad de direccionamiento de la memoria. El fragmento de código en el Apéndice 1 ilustra el manejo de la memoria.

3. Evaluación de la propagación de errores

Cuando se procesan ecuencias largas, existe la preocupación de determinar hasta que punto se están propagando los errores de truncamiento al utilizar las funciones seno y coseno de la biblioteca de funciones matemáticas. Singleton evita este problema calculando las funciones seno y coseno recursivamente a partir de valores iniciales que se obtienen de la biblioteca, e incluyendo una corrección a los valores calculados. El cálculo recursivo y diferentes esquemas de corrección están debidamente documentados en el artículo de Singleton (1969). En el apéndice 2 se da el pseudocódigo del esquema utilizado en el programa.

Las secuencias menores de 32 K no muestran el efecto de la propagación de los errores de truncamiento en las primeras 10 cifras significativas cuando los cálculos se hacen en precisión doble (8 bytes) y se usan las funciones seno y coseno de la biblioteca. Una secuencia de 32 K comienza a mostrar esporádicamente diferencias en la 10 cifra significativa como se muestra en el Cuadro 1. En este cuadro se muestra la transformada discreta de un pulso rectangular de longitud 32768 y ancho 8192 calculada teóricamente (Soley, 1978) y con un programa del dominio público. El programa utilizado se llama FFT.ZIP, fue escrito por Steve Sampson, y se obtuvo de la Computadora Simtel20. Utiliza las funciones seno y coseno de la biblioteca matemática y fue modificado para que el direccionamiento de memoria se hiciera con punteros enormes. De los valores exhibidos, sólo uno muestra una diferencia en la 10 cifra significativa. Como el ancho del pulso es la cuarta parte de la longitud total, todas los valores del espectro correspondientes a índices de frecuencia múltiplos de cuatro deben ser cero. Nótese que los valores calculados son cero a 10 cifras significativas y que los teóricos dan valores de 10 o menores. Otros programas utilizados por el autor muestran diferencias en la octava cifra significativa para longitudes cerca de 70 K. Como se verá más adelante, nuestro programa coincide en 10 cifras significativas al menos con los valores teóricos para longitudes de 70000 cuando se utiliza el esquema de corrección del Apéndice 2.

4. Algoritmos de raiz impar

El cálculo de la transformada discreta de Fourier de una secuencia compleja $x(k)$, donde $k = 0, 1, 2, \ldots, N-1$, dada por la relación

$$X(n) = \frac{1}{N} \sum_{k=0}^{N-1} x(k) W_N^{-nk}, \quad \textit{donde} \quad W_N = \exp i2\pi/N,$$

utiliza un número de sumas y multiplicaciones complejas del orden de N^2. El algoritmo de la transformada rápida (Cooley y Tukey, 1965) logra una disminución importante en el número de operaciones porque la transformada se puede expresar en términos de transformadas de orden menor. Cooley y Tukey ponen como ejemplo de su algoritmo el caso en que $N = 2^M$, en donde la repetida aplicación de la

transformada de raíz 2 permite una disminución en el número de operaciones a $N \times M$ aproximádamente. Como veremos en detalle más adelante, se pueden obtener reducciones en el número de operaciones también cuando el orden de la transformada tiene factores diferentes a 2. Es más, Singleton (1969) encuentra que la reducción máxima se obtiene cuando $N = 4^P$. Los algoritmos para factores de 2 y 4 se encuentran fácilmente en diversas fuentes (por ejemplo, Brigham 1974 y las referencias en ese texto). Sin embargo, el autor no ha encontrado en la literatura especializada los detalles de los algoritmos para factores impares. En este sección explicamos el cálculo de la transformada discreta cuando el orden N tiene como factor a 3. En los apéndices se dan los pseudocódigos para factores de 2, 4, 5 y factor impar en general.

Supongamos que $N = 3N_2$. Podemos entonces representar los índices n y k como sigue

$$n = 3n_1 + n_0, \quad n_0 = 0,1,2, \quad n_1 = 0,1,2,\ldots,N_2 - 1,$$

$$k = N_2k_1 + k_0, \quad k_0 = 0,1,2,\ldots,N_2 - 1, \quad k_1 = 0,1,2.$$

Aplicando estas representaciones al factor exponencial complejo se obtiene

$$W_N^{nk} = W_N^{3N_2n_1k_1} \quad W_N^{3n_1k_0} \quad W_N^{N_2n_0k_1} \quad W_N^{n_0k_0}.$$

El primer término de la derecha es la unidad ya que corresponde a $[W_N^N]^{n_1k_1}$. Los otros tres términos se pueden reagrupar de maneras distintas dando origen a las varias versiones del algoritmo de la transformada rápida. El agrupamiento en que se basa el programa es el conocido como la versión Sande- Tukey con factores de rotación (twiddle factors)

$$X(n_1,n_0) = \sum_{k_0=0}^{N_2-1} W_N^{-3n_1k_0} \quad \Big[\sum_{k_1=0}^{2} x(k_1,k_0)\; W_N^{-N_2n_0k_1}\Big] \quad W_N^{-n_0k_0}.$$

La clave del algoritmo radica en que $W_N^3 = W_{N_2}$, $W_N^{N_2} = W_3$, y de esta manera el término entre paréntesis cuadrados corresponde a una transformada de orden 3 mientras que la primera sumatoria a una transformada de orden N_2.

$$X(n_1,n_0) = \sum_{k_0=0}^{N_2-1} W_{N_2}^{-n_1k_0} \quad \Big[\sum_{k_1=0}^{2} x(k_1,k_0)\; W_3^{-n_0k_1}\Big] \quad W_N^{-n_0k_0}.$$

Esta afirmación se puede ver más claro haciendo las siguientes definiciones. Sean $x_1(.)$ los datos originales transformadas por la transformada de orden 3 y $\tilde{x}_1(.)$ estos últimos multiplicados por los factores de rotación

$$x_1(n_0,k_0) = \sum_{k_1=0}^{2} x(k_1,k_0)\; W_3^{-n_0k_1},$$

$$\tilde{x}_1(n_0,k_0) = x_1(n_0,k_0)\; W_N^{-n_0k_0}.$$

Así se obtiene

$$X(n_0, n_1) = \sum_{k_0=0}^{N_2-1} W_{N_2}^{-n_1 k_0}\, \tilde{x}_1(n_0, k_0).$$

La última expresión es la transformada de orden N_2 de la salida de la transformada de orden 3 multiplicada por los factores de rotación. Nótese que los índices de la transformada está invertidos, indicando que los resultados se guardan en memoria en una posición que no les corresponde y que al final de los cálculos es necesario reordenarlos. Otro punto importante es que el mismo procedimiento se puede seguir para un factor dado de N_2 y expresar esa transformada en términos de transformadas de orden menor. Por ejemplo, si $N = 2520 = 2^3 \times 3^2 \times 5 \times 7$, la transformada se obtiene aplicando la transformada de orden 2 tres veces, la transformada de orden 3 dos veces, y las transformadas de 5 y 7 una vez cada una. A primera vista pareciera que la reducción en el número de operaciones se logra a costa de utilizar espacio adicional en memoria para guardar las salidas de las transformadas de orden menor. Dichosamente los cálculos se pueden realizar sobre el arreglo de los datos originales si se ordenan correctamente y utilizando unas pocas variables para guardar resultados intermedios. Otra ventaja del algoritmo es que se reduce el número de veces que se calcula las funciones seno y coseno. Por ejemplo, $W_3^{k_1}$, para $k = 1, 2$ da factores del tipo $(-1 \pm i\sqrt{3})/2$, que se pueden codificar directamente, evitando el cálculo de los senos y cosenos respectivos. En resumen, para calcular la transformada de orden 3 es necesario entonces hacer lo siguiente:

Calcular para $k_0 = 0, 1, 2, \ldots, N-1$

Comienzo:

$$\begin{aligned} x_1(k_0) &= x(k_0) + x(N_2 + k_0) + x(2N_2 + k_0) \\ x_1(N_2 + k_0) &= x(k_0) + a \times x(N_2 + k_0) + b \times x(2N_2 + k_0) \\ x_1(2N_2 + k_0) &= x(k_0) + b \times x(N_2 + k_0) + a \times x(2N_2 + k_0) \end{aligned}$$

donde a y b son los números complejos

$$a = (-1 - i\sqrt{3})/2, \qquad b = (-1 + i\sqrt{3})/2.$$

Fin:

Calcular para $k_0 = 0, 1, 2, \ldots N-1$

Comienzo:

$$\begin{aligned} \tilde{x}_1(k_0) &= x(k_0) \qquad (No\ se\ hace\ nada) \\ \tilde{x}_1(N_2 + k_0) &= x(N_2 + k_0) \exp(-i2\pi k_0/N) \\ \tilde{x}_1(2N_2 + k_0) &= x(2N_2 + k_0) \exp(-i4\pi k_0/N) \end{aligned}$$

Fin:

Del pseudocódigo anterior se puede deducir que solo es necesario guardar provisionalmente en variables adicionales los valores de $x(k_0)$, $x(N_2+k_0)$ y $x(2N_2+k_0)$ y que los valores calculados se pueden almacenar en el mismo arreglo de los valores de entrada.

5. Cálculo de la transformada y su inversa

La transformada inversa se obtiene mediante la relación

$$x(n) = \sum_{k=0}^{N-1} X(k)\, W_N^{nk}$$

que se diferencia formalmente de la transformada directa por el signo de la exponencial compleja y por la ausencia del factor 1/N. Por la tanto es conveniente escribir una sola función que calcula la expresión

$$salida(n) = \sum_{k=0}^{N-1} entrada(k)\, W_N^{(signo)nk}$$

donde la entrada pueden ser los datos originales (y $signo = -1$) ó los valores transformados (y $signo = 1$). En el caso de la transformada, la división por N se debe hacer externa a la función. En detalle, esta función debe realizar lo siguiente:

(1) descomponer el orden N de la transformada en factores de 2, 4, 3 ,5 y factores impares diferentes a 3 y 5.

(2) llamar sucesivamente las transformadas correspondientes a los factores que componen el orden N,

(3) reordenar la salida. (Si $N = factor \times N_2$, la salida en $n_1 \times factor + n_0$ se encuentra en la posición $n_0 \times N_2 + n_1$).

6. Transformada de datos reales

La derivación de los algoritmos y su realización suponen que los datos son complejos. En la mayoría de los casos se trabaja con datos reales. Las funciones descritas se pueden usar poniendo la parte imaginaria a cero, aunque esto representa un uso poco eficaz de la memoria y el cálculo de una serie de operaciones innecesarias. El algoritmo de duplicación (Brigham, 1974) permite calcular la transformada de $N = 2 \times N\_MITAD$ datos reales utilizando una transformada de orden N_MITAD. El procedimiento es el siguiente:

(1) leer los datos reales alternativamente en la parte real y la parte imaginaria del arreglo complejo *entrada*[ ], es decir, la parte real contiene *dato*[0], *dato*[2], *dato*[4] hasta $dato[N-2]$ y la parte imaginaria, *dato*[1], *dato*[3], hasta $dato[N-1]$. Poner cero en $entrada[N]$.

(2) calcular la transformada $A[n]$ de orden N_MITAD de *entrada*[ ].

(3) calcular

$$A_1(n) = A^*[N\_MITAD - n] + A[n]$$
$$A_2(n) = i(A^*[N\_MITAD - n] - A[n])$$

para $n = 0, 1, 2, \ldots N\_MITAD$.

(4) calcular

$$C[n] = A_1[n] + A_2(n) \times W_N^{-n}$$
$$C[N\_MITAD - n] = A_1^*[n] - A_2^*[n] \times W_N^n$$

para $n = 0, 1, 2$ hasta $N\_MITAD/2$ ó $(N\_MITAD - 1)/2$ dependiendo si N_MITAD es par o impar.

(5) dividir $C[n]$ por $4 \times N\_MITAD$ para obtener la transformada de los datos de entrada reales.

7. Realización de los programas

Las funciones descritas en este artículo están disponibles al público en un paquete llamado FFTSING.ZIP que puede ser solicitado al autor o puede ser obtenido de la computadora WSMR-SIMTEL20.ARMY.MIL (26.2.0.74) a través de BITNET o INTERNET en el subdirectorio PD1:<MSDOS.TURBO-C> . El paquete contiene:

(1) SING.C: procedimientos que realizan la transformada o antitransformada de un arreglo complejo y la transformada de un arreglo de datos reales.

(2) VERSING.C: un programa ejecutable que verifica el funcionamiento de los procedimientos.

(3) README: un archivo con información de como se utilizan las funciones de SING.C.

8. Tiempos de ejecución y verificación de los resultados

Un método para examinar el funcionamiento de las subrutinas descritas es generar una señal cuya transformada sea conocida, obtener la transformada numéricamente y compararla con la expresión matemática, y luego invertir la transformada y contrastar el resultado con la señal original. De varias señales posibles (Soley, 1978), se escogió el pulso rectangular (M unos seguidos de ceros hasta completar una longitud L_TOT dada), por tener una transformada sencilla y porque al estar compuesta de dos números se facilita la comparación de la señal original con la obtenida invirtiendo la transformada numérica. Además, esta señal permite establecer cuál es el cero efectivo de máquina. Para el efecto se escribió un programa que permite desplegar los resultados calculados y exactos de la transformada y transformada inversa. En el Cuadro 2 se muestra el tiempo que tarda el cálculo de la transformada cuando se utiliza el algoritmo de duplicación y el programa se corre en una IBM PS2/80 con coprocesador matemático 80387.

Nótese que para 65536, que es una potencia de cuatro, el tiempo es menor que para una longitud de 61440 = (5) (3) (4096), en la cual la presencia de los factores impares 3 y 5 reducen la velocidad de cálculo.

En los Cuadros 3 y 4 se muestran los primeros valores de la transformada para $L\_TOT = 70000$ con $M = 17500$. Nótese que los valores calculados coinciden en la mayoría de los casos en más de 10 cifras significativas. El Cuadro 5 muestra la antitransformada en la región cercana al punto de transición del pulso rectangular. Los valores antitransformados difieren de la unidad en la 14 cifra significativa o en menos. Nótese que los valores que deben ser ceros son del orden de 10^{-15} , lo que fija el cero de máquina en esta aplicación.

9. Conclusiones

El programa descrito permite calcular la transformada discreta de Fourier de series muy largas en microcomputadoras PC y compatibles en tiempos aceptables. El uso de punteros enormes permite el direccionamiento lineal de la pila alta sin los problemas inherentes a los punteros largos. La longitud de la serie queda limitada por la memoria RAM disponible después de cargar el sistemo operativo (incluyendo manejadores, programas residentes TSR, etc.) y el programa que calcula las transformadas. La longitud de la serie no tiene que ser una potencia de dos lo que evita el tener que rellenar las series con ceros.

10. Referencias

[1] Brigham E O, 1974 *The Fast Fourier Transform* Prentice-Hall, Inc. Englewood Cliffs, New Jersey 252 página

[2] Cooley J W and Tukey J W, 1965 *An algorithm for machine calculations of complex Fourier series* **Math. Computation Vol 19** 297-301

[3] Das P K y Radhakrishna, 1991 *An analysis of Indian tide-gauge records* **Proc. Indian Acad. Sci. (Earth Planet Sci.) 100, No. 2** 177-194

[4] Singleton R C, 1969 *An algorithm for computing the mixed radix fast Fourier Transform* **IEEE Trans. Audio Electroacoust. AU-17** 93-103

[5] Sturges W, 1987 *Large-scale coherence of sea level at very low frequencies* **Journal of Physical Oceanography 17** 2084-2094

[6] Soley F J, 1978 *La transformada discreta de Fourier como aproximación a la transformada continua de Fourier* **Revista Ciencia y Tecnología, 2 (2)** 131-148

[7] Soley F J y Gutiérrez A, 1992 *Aplicación de un método de análisis de Fourier para secuencias temporales estremadamante largas: transformada discreta de Fourier de 21 años de registro horario del nivel del mar en Quepos, Costa Rica* Cienc.Tec.**17** No.1-2, 47-61

Table 3. Propagación de errores de truncamiento en el cálculo de la transformada discreta de Fourier utilizando el cálculo recursivo de las funciones seno y coseno. Se muestra la parte real de la transformada.

Indice	Real (teórico)	Real (calculado)
0	2.500000000000000e-01	2.500000000000000e-01
1	1.591620858421813e-01	1.591620858421800e-01
2	1.428571428573223e-05	1.428571428500598e-05
3	-5.304450451958492e-02	-5.304450451958477e-02
4	-9.749421526272799e-18	-1.076312667053257e-16
5	3.183813094123745e-02	3.183813094123742e-02
6	1.428571428572045e-05	1.428571428609446e-05
7	-2.272927683655823e-02	-2.272927683655820e-02
8	-9.749421212075457e-18	3.828152520799935e-17

Table 4. Propagación de errores de truncamiento en el cálculo de la transformada discreta de Fourier utilizando el cálculo recursivo de las funciones seno y coseno. Se muestra la parte imaginaria de la transformada.

Indice	Imag. (teórico)	Imag. (calculado)
0	-0.000000000000000e+00	0.000000000000000e+00
1	-1.591478001278956e-01	-1.591478001278968e-01
2	-1.591549426644678e-01	-1.591549426644677e-01
3	-5.305879023387063e-02	-5.305879023387076e-02
4	-1.750212078443084e-21	-8.075857443733972e-16
5	-3.182384522695174e-02	-3.182384522695178e-02
6	-5.305164641501572e-02	-5.305164641501584e-02
7	-2.274356255084394e-02	-2.274356255084400e-02
8	-3.500424156886169e-21	6.8107743339588350-17

Table 5. Transformada inversa para recobrar el pulso rectangular. Se muestra la zona de transición.

Indice	Parte real	Parte imaginaria
8747	9.999999999999915e-01	9.999999999999846e-01
8748	9.999999999999883e-01	9.999999999999747e-01
8749	9.999999999999850e-01	9.999999999999640e-01
8750	2.887724173734948e-15	2.795805182302613e-15
8751	-4.883354051638422e-15	-1.830504824666351e-15
8752	-1.261711739258006e-14	-6.448061497106775e-15

12. Apendice: Manejo de la memoria con los punteros enormes.

```
                                /* Variables globales */
unsigned long memoria_disponible, tamanno_maximo ;
double huge *hcomienzo;
                                /* Sigue el programa */
void main(void) {
      double far *comienzo;
                    /* Lo utiliza la función que asigna la memoria */
                                /* Sigue el programa */
      memoria_disponible = farcoreleft();
                   /* Devuelve lo que está disponible en la pila alta */
      printf ("%lu bytes libres en la pila alta ", memoria_disponible);
      tamanno_maximo = (memoria_disponible) / sizeof(double);
      /* Se calcula el tamaño máximo de una secuencia en precisión doble */
      printf( "Longitud máxima de los datos es %lu \n ", tamanno_maximo);
                              /* Se asigna toda la pila alta */
     if (( comienzo = farcalloc(tamanno_maximo, sizeof(double))) == NULL) {
          printf("Error en la asignación de memoria \n");
          exit(1);
      }
      hcomienzo = comienzo;
/* Ahora el apuntador "huge" hcomienzo apunta al inicio de la memoria asignada */
      printf("La secuencia comienza en %Fp ", hcomienzo);
        /* Los datos pueden direccionarse hcomienzo [10] = (double) 0, etc */
                                /* sigue el programa */
}
```

13. Apendice: Cálculo recursivo de las funciones seno y coseno.

Las funciones trigonométricas que utiliza la transformada discreta de Fourier se pueden representar mediante potencias de exponente entero de la raíz enésima de la unidad $\exp(\mathrm{i}2\pi/n)$. La secuencia de valores calculados corresponde a la ecuación recursiva

$$\xi_{k+1} = \xi_k \exp(\mathrm{i}\theta)$$

donde $\theta = 2\pi/n$ y $\xi_0 = 1$. En esta forma la ecuación acumula errores de truncamiento rápidamente por lo que se debe introducir un factor de corrección

$$\eta = -2\,\mathrm{sen}^2(\theta/2) + \mathrm{i}\,\mathrm{sen}(\theta)$$

Procedimiento:

(1) se calcula un valor provisional $\gamma_k = \xi_k + \eta_k \xi_k$

(2) el valor provisional γ_k se corrige con el factor δ_k como sigue, $\xi_{k+1} = \delta_k \times \gamma_k$ donde

$$\delta_k = [\gamma_k \gamma_k^*]^{-1/2} \approx \frac{1}{2}\left[\frac{1}{\gamma_k \gamma_k^* + 1}\right]$$

14. Apendice: Transformada de orden 2

Calcular para $k_0 = 0, 1, 2, \ldots, N_2 - 1$

Comienzo:

$$x_1(k_0) = x(k_0) + x(N_2 + k_0)$$
$$x_1(N_2 + k_0) = x(k_0) - x(N_2 + k_0)$$

Fin:

Calcular para $k_0 = 0, 1, 2, \ldots, N_2 - 1$

Comienzo:

$$\tilde{x}_1(k_0) = x(k_0) \qquad (\text{ No se hace nada})$$
$$\tilde{x}_1(N_2 + k_0) = x(N_2 + k_0)\exp(-\mathrm{i}2\pi k_0/N)$$

Fin:

15. Apendice: Transformada de orden 4

Calcular para $k_0 = 0, 1, 2, \ldots, N_2 - 1$
Comienzo:

$$\begin{aligned}
x_1(k_0) &= x(k_0) + x(N_2 + k_0) + x(2N_2 + k_0) + x(3N_2 + k_0) \\
x_1(N_2 + k_0) &= x(k_0) - \mathrm{i}x(N_2 + k_0) - x(2N_2 + k_0) + \mathrm{i}x(3N_2 + k_0) \\
x_1(2N_2 + k_0) &= x(k_0) - x(N_2 + k_0) + x(2N_2 + k_0) - x(3N_2 + k_0) \\
x_1(3N_2 + k_0) &= x(k_0) + \mathrm{i}x(N_2 + k_0) - x(2N_2 + k_0) - \mathrm{i}x(3N_2 + k_0)
\end{aligned}$$

Fin:
Calcular para $k_0 = 0, 1, 2, \ldots, N_2 - 1$
Comienzo:

$$\begin{aligned}
\tilde{x}_1(k_0) &= x(k_0) \qquad \text{(No se hace nada)} \\
\tilde{x}_1(N_2 + k_0) &= x(N_2 + k_0) \exp(-\mathrm{i}2\pi k_0/N) \\
\tilde{x}_1(2N_2 + k_0) &= x(2N_2 + k_0) \exp(-\mathrm{i}4\pi k_0/N) \\
\tilde{x}_1(3N_2 + k_0) &= x(3N_2 + k_0) \exp(-\mathrm{i}6\pi k_0/N)
\end{aligned}$$

Fin:

16. Apendice: Transformada de orden 5

Calcular para $k_0 = 0, 1, 2, \ldots, N_2 - 1$
Comienzo:

$$\begin{aligned}
x_1(k_0) &= x(k_0) + x(N_2 + k_0) + x(2N_2 + k_0) \\
&\quad + x(3N_2 + k_0) + x(4N_2 + k_0) \\
x_1(N_2 + k_0) &= x(k_0) + W_5^1 x(N_2 + k_0) + W_5^2 x(2N_2 + k_0) \\
&\quad + W_5^3 x(3N_2 + k_0) + W_5^4 x(4N_2 + k_0) \\
x_1(2N_2 + k_0) &= x(k_0) + W_5^2 x(N_2 + k_0) + W_5^4 x(2N_2 + k_0) \\
&\quad + W_5^1 x(3N_2 + k_0) + W_5^3 x(4N_2 + k_0) \\
x_1(3N_2 + k_0) &= x(k_0) + W_5^3 x(N_2 + k_0) + W_5^1 x(2N_2 + k_0) \\
&\quad + W_5^4 x(3N_2 + k_0) + W_5^2 x(4N_2 + k_0) \\
x_1(4N_2 + k_0) &= x(k_0) + W_5^4 x(N_2 + k_0) + W_5^3 x(2N_2 + k_0) \\
&\quad + W_5^2 x(3N_2 + k_0) + W_5^1 x(4N_2 + k_0)
\end{aligned}$$

donde los W son los números complejos

$W_5^1 = \ \ 0.309016994 - \mathrm{i}0.951056516$ $\qquad$ $W_5^2 = -0.809016994 - \mathrm{i}0.587785252$
$W_5^3 = -0.809016994 - \mathrm{i}0.587785252$ $\qquad$ $W_5^4 = \ \ 0.309016994 + \mathrm{i}0.951056516$

Fin:

Calcular para $k_0 = 0, 1, 2, \ldots, N_2 - 1$
Comienzo:

$$\tilde{x}_1(k_0) = x(k_0) \qquad \text{(No se hace nada)}$$
$$\tilde{x}_1(N_2 + k_0) = x(N_2 + k_0) \exp(-\mathrm{i}2\pi k_0/N)$$
$$\tilde{x}_1(2N_2 + k_0) = x(2N_2 + k_0) \exp(-\mathrm{i}4\pi k_0/N)$$
$$\tilde{x}_1(3N_2 + k_0) = x(3N_2 + k_0) \exp(-\mathrm{i}6\pi k_0/N)$$
$$\tilde{x}_1(4N_2 + k_0) = x(4N_2 + k_0) \exp(-\mathrm{i}8\pi k_0/N)$$

Fin:

17. Apendice: Transformada de orden impar P

Calcular para $k_0 = 0, 1, 2, \ldots, N_2 - 1$
Comienzo:
Calcular para $n_0 = 0, 1, 2, \ldots, P - 1$
Comienzo:

$$x_1(n_0 N_2 + k_0) = \sum_{k_1=0}^{P-1} x(k_1 N_2 + k_0) W_P^{-n_0 K_1}$$

Fin:
Fin:
Calcular para $k_0 = 0, 1, 2, \ldots, N_2 - 1$
Comienzo:
Calcular para $n_0 = 1, 2, 3, \ldots, P - 1$
Comienzo:

$$\tilde{x}_1(n_0 N_2 + k_0) = x(n_0 N_2 + K_0) \exp(-\mathrm{i}2\pi k_0 n_0/N)$$

Fin:
Fin: