

Rechten voorbehouden

Van interne verslagen zijn nadruk of aanhalingen
slechts toegestaan met uitdrukkelijke toestemming
van het NIOZ.

Beschrijving van een computer-programma voor
Fourier-analyse

door

A. Sannes jr.

Stage-verslag

over

werkzaamheden verricht als onderwerp in het tijdvak

1 mei 1974 - tot 1 augustus

aan het

NIOZ te Texel

onder supervisie van

W.G. v. Arkel

NEDERLANDS INSTITUUT VOOR ONDERZOEK DER ZEE

PUBLICATIES EN VERSLAGEN:

nummer 1975 - 2

Beschrijving van een computer-programma voor
Fourier-analyse

door

A. Sannes jr.

Stage-verslag

Inhoud

1. Summary - Samenvatting	2
2. Fourier transformatie	2
2.1. Een Fourierreeks door een aantal meetpunten	4
2.2. Numerieke oplosmethoden	6
2.3. Fast Fourier Transform	7
2.4. Rekeningtijd	9
3. Programma - tekst	10
4. Programma - beschrijving	15
4.1. Opbouw	15
4.1.2. Schematische opbouw	16
4.2. Invoer	17
4.3. Uitvoer	18
4.4. Invoer-voorbeeld en Uitvoervoorbeeld	19
5. Gebruik van het programma	22
6. Aanbevelingen voor de metingen	22
7. IMSL - Routines	23
8. Literatuur	24
9. Bijlage	25

1. SUMMARY

During my practical work at the Netherlands Institute for Sea Research, Texel, from May until August 1974, I have been engaged with the Fourier-transformation. The direct motive was the problem of a guest-investigator who studied the regularity in the frequency of pulsations of the hearts of guillemots. A computerprogram that can handle general Fourier-problems is written.

This program is tested with some exact ritmic functions and used for pulsation problem. Some recommendations are given about the formatting of the input-data.

SAMENVATTING

Tijdens mijn stage van 1 mei tot 1 augustus 1974 bij het NIOZ te Texel heb ik me enige tijd bezig gehouden met de Fourier transformatie. De direkte aanleiding was het probleem van een gastonderzoeker die naar een bepaalde regelmatigheid zocht in de hartslagfrequentie van een zeekoet.

Ik heb een computer-programma geschreven dat de algemene Fourier-problemen verwerken kan. Dit programma is getest met de eerder genoemde hartslagfrequentie en enige exact rithmische functies.

Verder worden enige aanbevelingen gedaan over het formaat van de data die getransformeerd moet worden.

2. FOURIER TRANSFORMATIE

Een reeks van de gedaante :

$$\frac{1}{2} a_0 + \sum_{n=1}^{\infty} (a_n \cdot \cos(nx) + b_n \cdot \sin(nx)) \quad (1)$$

Waarin a_n en b_n onafhankelijk zijn van x heet een goniometrische reeks en staat in de wiskunde bekend als de reeks van Fourier.

Veronderstel dat we een reeks waarnemings-uitkomsten hebben die een zekere periodiciteit vertonen. Een dergelijke reeks kunnen we als een goniometrische functie beschrijven. De methode is dan via de functie:

$$f(x) = \frac{1}{2} a_0 + \sum_{n=1}^N (a_n \cdot \cos(nx) + b_n \cdot \sin(nx)) \quad (2)$$

(N is het aantal meetpunten)

Van deze reeks moeten de termen a_n en b_n berekend worden. Door integratie ontstaat een eenvoudige betrekking tussen $f(x)$ en de te berekenen coëfficiënten a_n en b_n .

Voor de berekening van a_n en b_n maken we gebruik van de volgende integralen:

$$\begin{aligned} \int_{-\pi}^{\pi} \cos(nx) dx &= 0 \\ \int_{-\pi}^{\pi} \sin(nx) dx &= 0 \\ \int_{-\pi}^{\pi} \sin(mx) \cdot \cos(nx) dx &= 0 \\ \int_{-\pi}^{\pi} \cos(mx) \cdot \cos(nx) dx &= \begin{cases} 0 & \text{als } m \neq n \\ \pi & \text{als } m = n \neq 0 \end{cases} \\ \int_{-\pi}^{\pi} \sin(mx) \cdot \sin(nx) dx &= \begin{cases} 0 & \text{als } m \neq n \\ \pi & \text{als } m = n \neq 0 \end{cases} \end{aligned} \quad (3)$$

We integreren (2) tussen de grenzen $-\pi$ en π ; volgens (3) wordt de hele reeks 0, behalve de eerste term.

$$\begin{aligned} \int_{-\pi}^{\pi} f(x) dx &= \frac{1}{2} a_0 \cdot \int_{-\pi}^{\pi} dx = \pi a_0 \\ a_0 &= \frac{1}{\pi} \cdot \int_{-\pi}^{\pi} f(x) dx \end{aligned}$$

We vermenigvuldigen nu de beide leden van (2) met $\cos(mx)$ en integreren op de zelfde wijze.

$$\begin{aligned} \int_{-\pi}^{\pi} f(x) \cdot \cos(mx) dx &= \frac{1}{2} a_0 \cdot \int_{-\pi}^{\pi} \cos(mx) dx + \\ &+ \sum_{n=1}^N a_n \cdot \int_{-\pi}^{\pi} \cos(mx) \cdot \cos(nx) dx + b_n \cdot \int_{-\pi}^{\pi} \cos(mx) \cdot \sin(nx) dx \end{aligned}$$

Volgens (3) zijn alle leden in het rechter lid 0, behalve de integrand $\cos(mx)\cos(mx)$. Dus :

$$\int_{-\pi}^{\pi} f(x) \cdot \cos^2(mx) dx = a_m \cdot \int_{-\pi}^{\pi} \cos^2(mx) dx = \pi \cdot a_m$$

$$a_m = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cdot \cos(mx) \cdot dx \quad (4)$$

vermenigvuldiging met $\sin(mx)$ levert verder :

$$b_m = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cdot \sin(mx) \cdot dx \quad (5)$$

2.1. EEN FOURIERREEKS DOOR EEN AANTAL MEETPUNTEN

Heeft men een rij waarnemingen dan kan daar een bepaalde periodiciteit in worden gevonden door de coëfficiënten van de Fourier-polynoom, die men door deze meetpunten kan aanpassen te gaan bepalen.

Zoals in de vorige paragraaf is gebleken, zijn de coëfficiënten a_n en b_n te berekenen met de formules (4) en (5), de gedaante van deze formules moet dan wel aangepast worden, omdat we $f(x)$ niet kennen.

De integraal :

$$\int_{-\pi}^{\pi} f(x) \cdot dx$$

wordt vervangen door een eindige som $\sum_{n=0}^{N-1} x_n \cdot \Delta x$

x_n is een meetpunt uit de meetvector, x

N is het aantal meetpunten.

Neemt men voor het stukje Δx een constante waarde, dan kan ook worden geschreven:

$$\Delta x = \sum_{n=0}^{N-1} x_n$$

dit vereenvoudigt de berekening van de coëfficiënten a_n en b_n . De onderlinge afstand tussen de x-waarden van de metingen moet daarom ook constant zijn.

Bij de berekening van a_n en b_n zijn we uitgegaan van een lengte van 2π van het interval waarover we de periodiciteit wilden opsporen. In het algemeen zal deze lengte echter 1 zijn.

De coëfficiënten a_n en b_n zijn dan :

$$a_n = \frac{2}{1} \int_0^1 f(x) \cos \left(\frac{2n\pi x}{1} \right) \cdot dx \quad (6)$$

$$b_n = \frac{2}{1} \int_0^1 f(x) \sin \left(\frac{2n\pi x}{1} \right) \cdot dx \quad (7)$$

Voor een gemeten vektor is dit dan te schrijven als :

$$a_n = \frac{2}{1} \sum_{k=0}^{N-1} y_k \cdot \cos \left(\frac{2\pi \cdot n \cdot x_k}{1} \right) \cdot \Delta x$$

met $1 = N \cdot \Delta x$ en $x_k = k \cdot \Delta x$

$$\begin{aligned} a_n &= \frac{2 \Delta x}{1} \cdot \sum_{k=0}^{N-1} y_k \cdot \cos \left(\frac{2\pi n \cdot k \cdot \Delta x}{N \cdot \Delta x} \right) \\ &= \frac{2}{N} \sum_{k=0}^{N-1} y_k \cdot \cos \left(\frac{2\pi nk}{N} \right) \end{aligned} \quad (8)$$

evenzo voor b_n

$$b_n = \frac{2}{N} \sum_{k=0}^{N-1} y_k \cdot \cos \left(\frac{2\pi nk}{N} \right) \quad (9)$$

De gedaante van de Fourierreeks wordt ook veranderd, deze wordt als volgt

$$a_0 + \sum_{n=1}^{N/2} \left(a_n \cdot \cos \left(\frac{2\pi n}{L} x \right) + b_n \cdot \sin \left(\frac{2\pi n}{L} x \right) \right) \quad (10)$$

2.2. NUMERIEKE OPLOSMETHODEN

Oplossen van a_n en b_n is numeriek eenvoudig. Men moet echter voor de oplossing van de coëfficiënten $N \times N$ berekeningen uitvoeren. Dit is voor grote meetreeksen een zeer tijdrovende en dus kostbare methode.

In de laatste jaren is er echter een veel snellere methode ontwikkeld die bekend is onder de naam "Fast Fourier Transform" (FFT). Deze FFT is een efficiënte methode om de "Discrete Fourier Transform" (DFT) te berekenen.

De DFT is een wiskundige voorstelling van een goniometrische reeks en is gedefinieerd door :

$$\begin{aligned} A_r &= \sum_{k=0}^{N-1} x_k \cdot \exp(-2\pi i rk/N) \quad r=0, 1, \dots, N-1 \\ &= \sum_{k=0}^{N-1} x_k \cdot W^{rk} \end{aligned} \quad (11)$$

$$\text{met } W = \exp(-2\pi i/N)$$

A_r is de r^e coëfficiënt van de DFT, x_k is het k^e meetpunt
 i is $\sqrt{-1}$

Voor het terugtransformeren van de data moet de inverse van de DFT bekend zijn, deze inverse is :

$$x_l = (1/N) \sum_{r=0}^{N-1} A_r \cdot W^{-r \cdot l} \quad l = 0, 1, \dots, N-1 \quad (12)$$

Deze inverse vertoont veel overeenkomst met de DFT. Wordt voor numerieke oplossing een subroutine gebruikt, dan kan men deze subroutine met gewijzigde invoer ook gebruiken voor het terugtransformeren.

Uit de wiskunde is bekend :

$$\exp(ix) = \cos(x) + i \cdot \sin(x) \quad \text{en}$$

$$\exp(-ix) = \cos(x) - i \cdot \sin(x)$$

De DFT kan daarom ook geschreven worden als :

$$A_r = \sum_{k=0}^{N-1} x_k \cdot (\cos(2\pi rk/N) - i \cdot \sin(2\pi rk/N))$$

terugtransformeren levert dan :

$$x_l = (1/N) \sum_{r=0}^{N-1} A_r \cdot (\cos(2\pi rl/N) + i \cdot \sin(2\pi rl/N))$$

A_r is een complexe grootte, onafhankelijk van x_k . Het reële deel van A_r wordt steeds vermenigvuldigd met het stuk $\cos(2\pi rl/N)$, het imaginaire deel met het stuk $i \cdot \sin(2\pi rl/N)$. Dit laatste is dan ook weer reële. Het reële deel van A_r zou de coëfficiënt a_n en het imaginaire deel de coëfficiënt b_n zijn, als de sommatie niet van 0 tot $N-1$ liep.

2.3. FAST FOURIER TRANSFORM

Bij de FFT wordt de data reeks x_k verdeeld in 2 reeksen y_k en z_k . De rij y_k bevat de even genummerde data en de rij z_k de oneven. Van deze rijen kan men m.b.v. de DFT de coëfficiënten B_r en C_r bepalen.

$$B_r = \sum_{k=0}^{N/2-1} y_k \cdot \exp(-4\pi i rk/N) \quad r=0, 1, \dots, N/2-1 \quad (13)$$

$$C_r = \sum_{k=0}^{N/2-1} z_k \cdot \exp(-4 \pi i r k / N) \quad (14)$$

We zijn echter geïnteresseerd in de coëfficiënten A_r : deze zijn te berekenen uit B_r en C_r m.b.v. :

$$A_r = B_r + C_r \cdot \exp(-2 \pi i r / N) \quad (15)$$

$r = 0, 1, \dots, N/2-1$

$$\begin{aligned} A_{r + \frac{N}{2}} &= B_r + \exp(-2 \pi i (r + \frac{N}{2}) / N) \cdot C_r \\ &= B_r - \exp(-2 \pi i r / N) \cdot C_r \end{aligned}$$

met $\exp(-2 \pi i / N) = W$ geldt:

$$\begin{aligned} A_r &= B_r + W^r \cdot C_r \\ A_{r+N/2} &= B_r - W^r \cdot C_r \end{aligned} \quad (16)$$

De coëfficiënten B_r en C_r kunnen we nu weer op de zelfde wijze berekenen. Is het aantal meetpunten N nu gelijk aan 2^m , dan kan men doorgaan tot een serie DFT's van steeds 2 punten overgebleven is. Deze kan met de formule (11) worden berekend en met de formules (16) kunnen hieruit de coëfficiënten A_r worden bepaald.

Omdat er voor de periodiciteit geen verschil is tussen een sinus en een cosinus ($\sin(x) = \cos(x + \frac{\pi}{2})$) worden het reële en imaginaire van de termen A_r kwadratisch opgeteld. Deze rij met sommen moet men de optredende frequentie. De absolute grootte van deze frequentie geeft weinig informatie, de relative frequentie-grootte geeft echter een goede verhouding weer tussen de verschillende rithmiken die optreden.

2.4. REKENTIJD

Het aantal berekeningen voor de FFT is $N \cdot {}^2\log(N)$, die van de DFT is N^2 . Nemen we als voorbeeld een meet rij met $1024 = 2^{10}$ meetpunten dan is

$$N \cdot {}^2\log(N) = 10 \cdot 1024 = 10240$$

$$N^2 = 1024 \cdot 1024 = 1048576$$

dit is ruim 100 * groter.

De rekentijd zal van de DFT daarom ook bij een rij van 1024 meetpunten ongeveer 100 * zo lang zijn als die bij de FFT.

3. Programma-tekst

```
PROGRAM FOURIER(INPUT,OUTPUT,TAPE1=INPUT,TAPE2=OUTPUT)
```

```
C ***
```

```
C ***
```

```
C ***
```

```
DIT PROGRAMMA LEVERT EEN FOURIER TRANSFORMATIE VAN EEN  
MEET VECTOR MET EEN WILLEKEURIG AAN TAL PUNTEN.
```

```
5 C ***
```

```
C ***
```

```
C ***
```

```
DE VOLGENDE DIMENSION STATEMENT MOET VOOR HET GEBRUIK VAN HET  
PROGRAMMA AAN GEPAST WORDEN.
```

```
C ***
```

```
C ***
```

```
DIMENSION COM(N),COSIN(N),IX(N),F(N/2+1)
```

```
10 C ***
```

```
C ***
```

```
C ***
```

```
N IS HET AANTAL MEETPUNTEN NAAR BOVEN AFGEROND TOT EEN  
MACHT VAN TWEE.
```

```
C ***
```

```
V.B. AANTAL MEETPUNTEN IS 10 DAN N=16 (8<10)
```

```
C ***
```

```
15
```

```
DIMENSION COM(300),COSIN(300),IX(300),F(151)
```

```
COMPLEX COM
```

```
CALL LEES(COM,N,M,NREAD,1)
```

```
CALL TABLE(COM,N,2,11HMEASURED ,.FALSE.)
```

```
CALL INIT(COSIN,IX,N,M)
```

```
20
```

```
CALL RFT(COM,M,COSIN,IX)
```

```
CALL TABLE(COM,N,2,11HTRANSFORMED,.TRUE.)
```

```
CALL FREQ(F,N,COM,1)
```

```
CALL TABLE(COM,N,2,11HTRANSFORMED,.TRUE.)
```

```
CALL HEAD(1,2)
```

```
25
```

```
CALL GRAFIEK(F,N,2)
```

```
DO 10 I=1,N
```

```
COM(I)=CONJG(COM(I))/FLOAT(N)
```

```
10 CONTINUE
```

```
CALL RFT (COM,M,COSIN,IX)
```

```
30
```

```
DO 20 I=1,N
```

```
COM(I)=CONJG(COM(I))
```

```
20 CONTINUE
```

```
CALL TABLE(COM,N,2,13HRETRANSFORMED,.FALSE.)
```

```
END
```



```

SUBROUTINE LEES(C,N,M,NREAD,IUNIT)
DIMENSION C(1)
COMPLEX C
NREAD=RN(IUNIT)
5 M=IFIX(ALOG(FLOAT(NREAD))/ALOG(2.))+1
N=2**M
IF(N.EQ.2*NREAD) M=M-1
IF(N.EQ.2*NREAD) N=N/2
10 DO 30 K=NREAD,N
C(K)=CMPLX(0.,0.)
30 CONTINUE
DO 40 I=1,NREAD
HULP=RN(IUNIT)
C(I)=CMPLX(HULP,0.)
15 40 CONTINUE
RETURN
END

```

```

SUBROUTINE FREQ(F,N,C,IUNIT)
DIMENSION F(1),C(1)
COMPLEX C
NF=N/2+1
5 DO 10 I=1,NF
A=REAL(C(I))
B=AIMAG(C(I))
F(I)=SQRT(A*A+B*B)
10 CONTINUE
10 CALL DESTROY(IUNIT,F,NF,C,N)
RETURN
END

```

```

SUBROUTINE GRAFIEK(F,N,IUNIT)
DIMENSION F(1),IBLANK(100)
DATA IBLANK,ISTER/100*1H ,1H*/
H00GST=0.
5 NF=N/2+1
DO 10 I=2,NF
IF(F(I).GT.H00GST) H00GST=F(I)
10 CONTINUE
EENHEID=H00GST/99.
10 DO 20 I=2,NF
M=F(I)/EENHEID+1.5
WRITE(IUNIT,2) F(I),(IBLANK(J),J=1,M),ISTER
20 CONTINUE
2 FORMAT(E12.5,5H . ,101A1)
15 RETURN
END

```

```
SUBROUTINE DESTROY(IUNIT,F,NF,C,N)
```

```
LOGICAL ERROR
```

```
DIMENSION F(1),C(1)
```

```
COMPLEX C
```

```
FAKTOR=RN(IUNIT)
```

```
ERROR=(FAKTOR.LT.0.).OR.(FAKTOR.GT.1.)
```

```
IF(ERROR) CALL DISPLA(42HINPUT ERROR DETECTED BY DESTROY,FAKTOR I
```

```
1: ,FAKTOR)
```

```
IF(ERROR) CALL DISPLA(13HPROGRAM STOPS ,0)
```

```
IF(ERROR) STOP
```

```
H00GST=0.
```

```
DO 20 I=2,NF
```

```
IF(H00GST.LT.F(I)) H00GST=F(I)
```

```
20 CONTINUE
```

```
H00GST=H00GST*FAKTOR
```

```
DO 30 I=2,NF
```

```
IF(F(I).LT.H00GST) C(I)=C(N-I+2)=CMPLX(0.,0.)
```

```
30 CONTINUE
```

```
RETURN
```

```
END
```

```
SUBROUTINE TABLE(C,N,IUNIT,IARRAY,COMPLX)
```

```
LOGICAL COMPLX
```

```
DIMENSION C(1),IARRAY(2)
```

```
N2=N+N
```

```
WRITE(IUNIT,1) (IARRAY(I),I=1,2)
```

```
IF(COMPLX) WRITE(IUNIT,2) (C(I),I=1,N2)
```

```
IF(.NOT.COMPLX) WRITE(IUNIT,3) (C(I),I=1,N2,2)
```

```
1 FORMAT(1H1,*THIS LISTING CONTAINS THE *,A10,A4,*DATA.*)
```

```
2 FORMAT(1H ,*EVERY SECOND NUMBER IS THE IMAGINARY PART OF THE TRAN
```

```
1FORMED DATA*/1H0/(10E13.6))
```

```
3 FORMAT(1H0/(10E13.6))
```

```
RETURN
```

```
END
```

```
SUBROUTINE HEAD(INP,OUTP)
```

```
INTEGER INP,OUTP
```

```
DIMENSION I(8)
```

```
K=RN(INP)
```

```
WRITE(OUTP,2)
```

```
IF(K.EQ.0) RETURN
```

```
DO 10 J=1,K
```

```
READ(INP,3) I
```

```
WRITE(OUTP,4) I
```

```
10 CONTINUE
```

```
2 FORMAT(1H1)
```

```
3 FORMAT(8A10)
```

```
4 FORMAT(1H ,8A10)
```

```
RETURN
```

```
END
```

SUBROUTINE RFT(Z,M,CS,IX)

```

C      FOURIERTRANSFORM OF Z OF 2**M POINTS
C      Z(K) = SUM(EXP(2*PI*I*.*K/N)*Z(.)).
5  C      ASSUMING : CS(K) = COS(2*PI*(K-1)/2**M), K = 1 TO N/4+1.
C      AND IX(I+1) = 1+BIT-REVERSE OF I FOR I = 0 TO N-1.

      COMPLEX Z(M),T,Y
      DIMENSION CS(M),IX(M)

10      N4 = 2**(M-2)
      N2 = 2*N4
      N = 2*N2
      M2MAX = 1
15      M1MAX = N2
      NDIF = 2

      C      M2MAX = 2**(L-1), WHERE L = 1,M
      C      M1MAX = 2**(M-L), WHERE L = 1,M
20  C      NDIF = 2**L , WHERE L = 1,M

      DO 10 I = 2,N2
      J = IX(I)
      IF (I.GE.J) GOTO 10
25      T = Z(I)
      Z(I) = Z(J)
      Z(J) = T
10  CONTINUE
      NF = N2+4
30      NL = N-2**((M+1)/2)
      DO 11 I = NF,NL,2
      J = IX(I)
      IF (I.GE.J) GOTO 11
35      T = Z(I)
      Z(I) = Z(J)
      Z(J) = T
11  CONTINUE
      DO 100 L = 1,M
      DO 90 M2 = 1,M2MAX
40      C      M2 = 1,2**(L-1)

      K = (M2-1)*M1MAX
      IF (K-N4) 20,30,30
45      20 W = CS(1+K)
      GOTO 40
      30 W = -CS(1+N2-K)
      40 CONTINUE
      T = CMPLX(W,CS(1+IABS(K-N4)))
50      C      T = EXP(2*PI*I*K/N), K=1 TO N/2-1

      DO 80 M1 = 1,M1MAX
55  C      M1 = 1,2**(M-L)

```

```

      NP = (M1-1)*NDIF+M2
      Y = T*Z(NP+M2MAX)
      Z(NP+M2MAX) = Z(NP)-Y
5      Z(NP) = Z(NP)+Y
      80 CONTINUE
      90 CONTINUE
      M2MAX = 2*M2MAX
      M1MAX = M1MAX/2
10     NDIF = 2*NDIF
      100 CONTINUE
      Z(N+1) = Z(1)

      RETURN
15     END

```

```

      SUBROUTINE INIT(CS,IX,N,M)

      DIMENSION CS(N),IX(N)

5      C      NON ANSI FORTRAN EXT. 4.0
      C      ASSUMED N = 2**M
      C      CS(I) = COS(2*PI*(I-1)/2**M) , K = 1 TO N/4 + 1
      C      OUTPUT : CS,IX

10     PI = 3.141592653590
      TWOM1 = 2.0/N
      IUP = N/4 + 1
      DO 1 K = 1,IUP
15     1 CS(K) = COS(PI*(K-1)*TWOM1)

      C      IX(I) = 1 + BIT-REVERSE OF I-1 FOR I = 1 TO N
      C      THE CASE M = 1 (N = 2) GOES WRONG

20     M1 = M-1
      DO 3 K = 1,N
      K1 = K-1
      L = K1 .AND. 1
      DO 2 J = 1,M1
25     2 L = SHIFT(L,1) + (SHIFT(K1,-J) .AND. 1)
      IX(K) = L+1
      3 CONTINUE

      RETURN
      END

```

4. PROGRAMMA BESCHRIJVING

Het programma Fourier is geschreven in Fortran -- Extended voor de CDC -- 6600 en heeft gedraaid met de FTN 4.1 + P370 compiler. Het is een algemeen programma met een flexibele invoer.

4.1. OPBOUW

Het programma is, zoals dit tegenwoordig steeds gebruikelijker is, opgebouwd uit modulen. De modulen zijn de subroutines die zoveel mogelijk een logische naam hebben, zodat de programma-voortgang voor iedereen eenvoudig te lezen is. In het kort zal de functie van de verschillende subroutines in het hoofd-programma beschreven worden.

-- LEES --- Met deze routine wordt het getal gelezen dat het aantal data-punten aangeeft; vervolgens worden de data-punten gelezen. Het aantal data (nread) wordt nu naar boven afgerond tot een macht van 2, dit is n . De gelezen waarden worden toegekend aan het reële deel van het complexe array COM. De overige elementen van het array COM worden met 0 geïnialiseerd, evenals het imaginaire deel.

(De lengte van COM is $n = 2^{**m}$)

-- TABLE --- Deze subroutine maakt een listing van de data. De laatste parameter is . FALSE. d.w.z. dat alleen het reële deel van COM wordt geprint.

-- INIT --- Dit is een routine die de arrays COSIN en IX initialiseert voor het gebruik in de routine RFT.

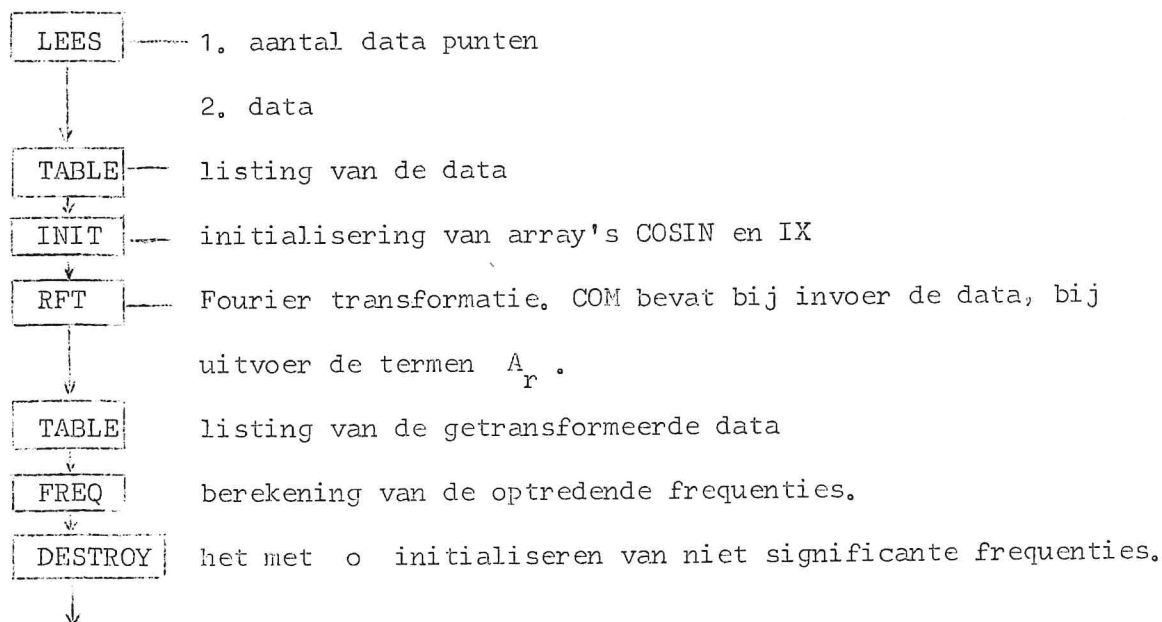
-- RFT --- Dit is de FFT routine. De array COM bevat bij de invoer de data, zoals hierboven beschreven, bij de uitvoer bevat het de termen A_r .

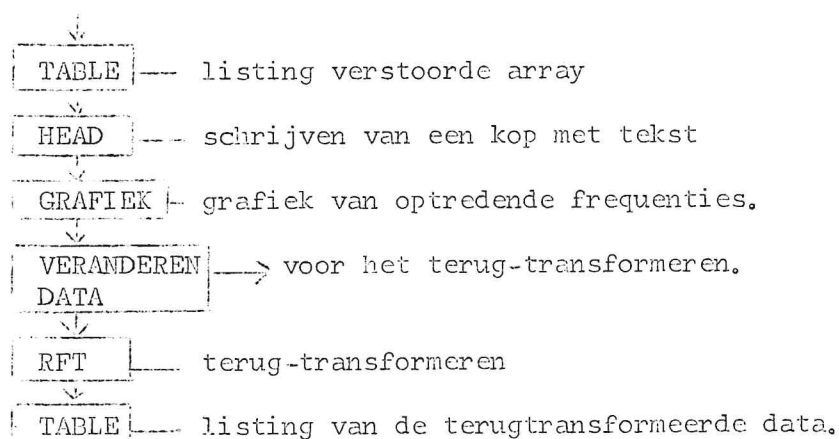
- TABLE -- (Zie boven) de laatste parameter is nu . TRUE. dan wordt ook het
imaginaire deel gelist.
- FREQ -- Deze routine berekent uit de termen A_r de frequentie-grootte
zoals deze in de meet-vector optreedt. Vervolgens wordt er een
faktor (tussen de 0 en 1) gelezen die aangeeft welke frequenties
niet significant zijn. Deze frequenties worden vervolgens bepaald,
en de elementen in de array COM, waaruit deze frequenties
berekend zijn worden met 0 geïnitieeliseerd.
- TABLE -- (Zie boven)
- HEAD -- Deze routine leest een aantal records met tekst van de input.
file en schrijft deze als kop boven de hierna beschreven grafiek.
- GRAFIEK -- Deze routine maakt een grafiek van de optredende frequenties
op de regeldrukker.

Voor het terug-transformeren moet de data veranderd worden (Zie theorie)
om de zelfde routine weer te kunnen gebruiken. Dit is de volgende stap
in het programma.

- RFT -- Hiermee wordt nu de data teruggetransformeerd.
- TABLE -- (Zie boven)

4.1.2. SCHEMATISCHE OPBOUW





4.2. INVOER

De invoer is verdeeld in 3 groepen t.w.

- de te transformeren data.
- de significantie faktor.
- de tekst boven de grafiek.

Het formaat van de groepen zal hier worden beschreven :

a) De te transformeren data :

Dit is een willekeurig aantal data punten (alleen y-waarde).

Als eerste getal moet het aantal data punten gegeven worden, dit moet een integer getal zijn. Daarna komen de data punten. De getallen moeten gescheiden worden door een komma. Ook tussen het eerste getal en de volgende data moet een komma. Het formaat van de data en het aantal per kaart is geheel vrij.

b) De significantie-faktor :

Dit is een getal tussen 0 en 1 en kan direkt na de data, gescheiden door een komma, ingevoerd worden. (Op een aparte kaart ponsen is echter wel overzichtelijker.)

De significantie is een getal waarmee de waarde van de sterkste frequentie wordt vermenigvuldigd. Iedere frequentie die een kleinere waarde heeft dan dit produkt is dan niet significant.

Voorbeeld :

$F1 = 10$; $F2 = 12$; $F3 = 2$, $F4 = 15$, $F5 = 6$ $F6 = 3$.

de significantie-faktor is 0.5

het produkt is $0.5 * F_u = 7.5$ dus de frequenties $F3$, $F5$ en $F7$ zijn niet significant,

Leest het programma een faktor die niet ligt tussen de 0 en 1

dan wordt het programma beëindigd. De waarde van de gelezen

variabele wordt in de dayfile geprint, na de mededeling :

"Input error detected by DESTROY factor is :"

c) Tekst boven de grafiek

Na de kaart met de significantie-faktor komt een kaart met het aantal tekst-kaarten. (mag ook 0 zijn). De daarna volgende kaarten worden als tekst gelezen en volgens gelijk formaat boven de frequentie-grafiek geprint. (Een blanke kaart is een regel overslaan).

4.3. UITVOER

De uitvoer komt normaal op de Regeldrukker en bestaat uit 5 groepen, t.w.:

- Een listing van de gelezen data
- Een listing van de getransformeerde data
- Een listing van de getransformeerde en gereduceerde data.
- Een frequentie-grafiek
- Een listing van de geretransformeerde data

(heeft men als significantie-faktor 0 genomen, dan moet de geretransformeerde data gelijk zijn aan de gelezen data).

4.4 Invoer-voorbeeld en Uitvoer-voorbeeld

JOB CARD.
 ACCOUNT CARD.
 FTN.
 REQUEST, LIB, *PF.
 5 EDIT LIB, USER.
 CATALOG, LIB, BIBLIOTHEEK, ID=NI0ZSAN, RP=0.
 7/8/9

PROGRAMMA TEKST
 MET SUBROUTINES

7/8/9
 LIBRARY (LIB, NEW)
 REWIND (LG0)
 5 ADD (*, LG0)
 SETAL (FOURIER, 1)
 FINISH.
 ENDRUN.
 6/7/8/9

JOB CARD.
 ACCOUNT CARD.
 ATTACH, LIB, BIBLIOTHEEK, ID=NI0ZSAN.
 LIBRARY, LIB.

5 FOURIER.
 FOURIER.
 7/8/9
 200

10	0.0000,	.0314,	.0628,	.0941,	.1253,	.1564,	.1873,	.2180,	.24
	.3089,	.3386,	.3679,	.3970,	.4256,	.4538,	.4815,	.5088,	.53
	.5875,	.6126,	.6372,	.6610,	.6843,	.7068,	.7287,	.7498,	.77
	.8087,	.8268,	.8441,	.8605,	.8760,	.8908,	.9046,	.9175,	.92
	.9509,	.9601,	.9684,	.9758,	.9822,	.9876,	.9920,	.9955,	.99
15	1.0000,	.9995,	.9981,	.9956,	.9922,	.9878,	.9825,	.9761,	.96
	.9514,	.9412,	.9301,	.9182,	.9053,	.8915,	.8768,	.8613,	.84
	.8097,	.7908,	.7712,	.7509,	.7298,	.7080,	.6854,	.6622,	.63
	.5888,	.5631,	.5369,	.5102,	.4829,	.4552,	.4270,	.3984,	.36
	.3104,	.2804,	.2501,	.2196,	.1889,	.1579,	.1268,	.0956,	.06
20	.0016,	-.0298,	-.0612,	-.0925,	-.1237,	-.1548,	-.1857,	-.2165,	-.24
	-.3074,	-.3371,	-.3665,	-.3955,	-.4241,	-.4524,	-.4801,	-.5074,	-.53
	-.5862,	-.6114,	-.6359,	-.6598,	-.6831,	-.7057,	-.7276,	-.7488,	-.76
	-.8078,	-.8259,	-.8432,	-.8597,	-.8753,	-.8900,	-.9039,	-.9169,	-.92
	-.9504,	-.9597,	-.9680,	-.9754,	-.9819,	-.9873,	-.9918,	-.9953,	-.99
25	-1.0000,	-.9996,	-.9982,	-.9958,	-.9924,	-.9881,	-.9827,	-.9765,	-.96
	-.9518,	-.9417,	-.9307,	-.9188,	-.9059,	-.8922,	-.8776,	-.8621,	-.84
	-.8106,	-.7918,	-.7723,	-.7519,	-.7309,	-.7091,	-.6866,	-.6634,	-.63
	-.5901,	-.5645,	-.5383,	-.5115,	-.4843,	-.4566,	-.4285,	-.3999,	-.37
	-.3119,	-.2819,	-.2517,	-.2211,	-.1904,	-.1595,	-.1284,	-.0972,	-.06

0.
 30 3
 TEST MET EEN SINUS

PROGRAMMEUR A.SANNES

7/8/9

35 DATA

7/8/9
 6/7/8/9

LISTING CONTAINS THE TRANSFORMED DATA.
Y SECOND NUMBER IS THE IMAGENAIR PART OF THE TRANSFORMED DATA

3687E-12	.141668E-11	.241989E+00	.966076E+00	.983741E+00	.184045E+01
0469E+01	.322193E+01	.321586E+02	.316734E+01	-.194139E+02	.287978E+01
2038E+01	.124333E+01	-.327052E+00	.691494E+00	-.493905E-01	.248303E+00
5162E-01	-.149349E+00	.303573E-12	.246730E-12	.376787E+00	.225838E+00
9325E+01	.834089E+00	-.185281E+01	.876314E+00	-.986729E+00	.809787E+00
6725E-01	.152846E+00	-.932032E-02	-.946308E-01	-.103323E+00	-.288768E+00
4467E+00	-.339130E+00	-.756414E+00	-.189472E+00	.120000E+02	-.364288E-11
9156E+00	.416155E+00	-.267635E+00	.400544E+00	-.103323E+00	.288768E+00
1573E+00	-.414214E+00	-.479333E+00	-.646305E+00	-.986729E+00	-.809787E+00
0627E+02	-.690856E+00	.156171E+01	-.473739E+00	.376787E+00	-.225838E+00
1183E-01	.175108E+00	-.224816E-02	.457622E-01	-.493905E-01	-.248303E+00
8434E+01	-.184251E+01	-.582843E+01	-.241421E+01	-.194139E+02	-.287978E+01
1028E+01	-.301367E+01	.230461E+01	-.254274E+01	.983741E+00	-.184045E+01

.230461E+01	.254274E+01	.451028E+01	.301367E+01
-.582843E+01	.241421E+01	-.248434E+01	.184251E+01
-.224816E-02	-.457622E-01	-.531183E-01	-.175108E+00
.156171E+01	.473739E+00	.140627E+02	.690856E+00
-.479333E+00	.646305E+00	-.171573E+00	.414214E+00
-.267635E+00	-.400544E+00	-.459156E+00	-.416155E+00
-.756414E+00	.189472E+00	-.634467E+00	.339130E+00
-.932032E-02	.946308E-01	-.226725E-01	-.152846E+00
-.185281E+01	-.876314E+00	-.419325E+01	-.834089E+00
-.303573E-12	.360417E-12	-.895162E-01	.149349E+00
-.327052E+00	-.691494E+00	-.102038E+01	-.124333E+01
.321586E+02	-.316734E+01	.900469E+01	-.322193E+01
.241989E+00	-.966076E+00		

TEST PROGRAMMA MET BLOKGOLF

PROGRAMMEUR A.SANNES

```

.99592E+00
.20869E+01
.34317E+01
.54245E+01
.95638E+01
.32314E+02
.19626E+02
.63086E+01
.30930E+01
.16084E+01
.76494E+00
.25317E+00
.45817E+01
.18299E+00
.17412E+00
.39119E+12
.43928E+00
.16320E+01
.14080E+02
.42754E+01
.20496E+01
.12765E+01
.80466E+00
.44834E+00
.15452E+00
.95089E+01
.30670E+00
.48173E+00
.61968E+00
.71941E+00
.77978E+00
.12000E+02

```

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

*

5. GEBRUIK VAN HET PROGRAMMA

Voor het gebruik van het programma moet men het hoofdstuk invoer goed lezen!

Verder moet het programma aangepast worden aan de grootte van het probleem. Hiervoor moet het 2^e statement gewijzigd worden. De lengten van de te gebruiken arrays zijn gegeven in de comment-kaarten. Voor een probleem met 200 meetpunten wordt het 2^e statement :

```
DIMENSION COM (256) , COSIN (256), IX (256), F (129)
```

Om onnodig dragen met kaarten tegen te gaan heb ik de job in tweeën verdeeld.

In het eerste deel wordt het programma vertaald en op permanent file opgeborgen, in het tweede deel wordt dit programma geladen en tot uitvoer gebracht. Gaat het programma nu door een invoer fout, dan hoeft men het hele carddeck niet meer in te lezen en te vertalen.

6. AANBEVELINGEN VOOR DE METINGEN

Uit de theorie volgt dat de afstand tussen de meetpunten (x-as) gelijk moet zijn. De x-waarden worden voor de transformatie niet gebruikt, en daarom ook niet ingelezen.

Heeft men nu bij bepaalde metingen grote hoeveelheden meet-gegevens dan doet zich vaak snel het probleem voor, "hoe krijg ik ze op ponskaarten". Het tekenen van grafieken is hiervoor waarschijnlijk wel de slechtste oplossing. Ervaring heeft geleerd dat het opmeten vanaf deze grafieken en vervolgens het ponsen en zeer tijdrovende en vervelende bezigheid is. Een betere oplossing zou al zijn om de data met een digitale meter de bepalen en vervolgens op papier de laten typen door een printer, maar hierbij blijft de ponstijd.

De beste oplossing is waarschijnlijk om een ponsband-ponscr te gebruiken. Het programma zal hiervoor wel moeten worden aangepast, maar deze methode

bespaart zeer veel ponswerk.

De programma wijziging komt doordat de in leesroutine RN(N), die gebruikt wordt voor het in-lezen, gemaakt is voor ponskaarten. Het is echter altijd mogelijk om de getallen m.b.v. een zeer eenvoudig programma in te lezen vanaf de ponsband en ze daarna op ponskaarten te schrijven. Het is misschien ook mogelijk dat de in leesroutine RN aangepast wordt. Heeft U Uw gegevens op ponsband staan, dan is het aan te bevelen om eens kontakt op te nemen met M. Helvensteyn op het Reken Centrum van het R.C.N. te Petten. Hij is namelijk de maker van de in leesroutine RN.

Het is verder aan te bevelen om bij metingen met een hoge gemiddelde waarde in vergelijking tot de maximale variantie, een constante van de meet waarden af te trekken, daar anders de belangrijk frequenties niet zichtbaar worden.

Als voorbeeld wil ik dit noemen:

een rij :

1 , - 1 , 1 , - 1 , 1 , - 1 zal betere resultaten geven als de rij:
1001, 999, 1001, 999, 1001, 999

Dit reduceren kan eventueel nog aan het programma toegevoegd worden.

7. DE IMSL -- ROUTINES

Ook zijn er in de systeem-bibliotheek een aantal Fourier-routines opgenomen. Van deze routines wordt in mindere mate gebruik gemaakt, omdat de RFT routine sneller en efficiënter is dan de vergelijkbare IMSL routines.

Ik heb de beschrijving van deze routines als bijlage aan dit verslag toegevoegd.

8. LITERATUUR

- Hogere Wiskunde voor het Technisch Onderwijs
uitgeverij H. Stam.
- Idee transactions on audio and electroacoustics,
vol. AU-15, no. 2, June 1967.

Bijlage : IMSL - Routines.

```

SUBROUTINE FFCSIN (X,N,A,B,IWK)
C
C-----S-----LIBRARY 3-----
C
C  FUNCTION          - TO COMPUTE THE SINE AND COSINE TRANSFORMS
C                    OF A SET OF REAL DATA.
C  USAGE            - CALL FFCSIN(X,N,A,B,IWK)
C  PARAMETERS      X  - ON INPUT, THE VECTOR X (LENGTH N) CONTAINS
C                    THE DATA TO BE TRANSFORMED. ON OUTPUT,
C                    X IS DESTROYED.
C                    N  - NUMBER OF DATA POINTS (LENGTH OF X).
C                    N MUST BE EVEN.
C                    A  - OUTPUT VECTOR OF LENGTH N/2 + 1, CONTAINING
C                    THE COSINE TRANSFORMS. (VECTORS X AND A MAY
C                    COINCIDE.
C                    B  - OUTPUT VECTOR OF LENGTH N/2 + 1, CONTAINING
C                    THE SINE TRANSFORMS.
C                    IWK - WORK AREA VECTOR. IF N=2**M, THEN THE LENGTH
C                    OF IWK IS M. OTHERWISE, IWK HAS LENGTH OF
C                    3(F+N)+26, WHERE F IS THE NUMBER OF PRIME
C                    FACTORS IN N/2.
C  PRECISION        - SINGLE
C  REGD. IMSL ROUTINES - FFRDR2,FFTR,FFTP,FFT2
C  LANGUAGE          - FORTRAN
C-----

```

FFCS0010
FFCS0020
FFCS0030
FFCS0040
FFCS0050
FFCS0060
FFCS0070
FFCS0080
FFCS0090
FFCS0100
FFCS0110
FFCS0120
FFCS0130
FFCS0140
FFCS0150
FFCS0160
FFCS0170
FFCS0180
FFCS0190
FFCS0200
FFCS0210
FFCS0220
FFCS0230
FFCS0240
FFCS0250

CALL FFCSIN(X,N,A,B,IWK)

Purpose

FFCSIN computes the sine and cosine transformations of a real data vector X. It is assumed that the length of the data set is even.

Algorithm

Given a set of N(even) data points $X=(x_1, x_2, \dots, x_N)^t$ where $x_1, x_2, \dots, x_N$ are real numbers, define the cosine transformation A and the sine transformation B by

$$A_{k+1} = \sum_{j=0}^{N-1} x_{j+1} \cos(2\pi jk/N) \quad B_{k+1} = \sum_{j=0}^{N-1} x_{j+1} \sin(2\pi jk/N) \text{ for } k=0, 1, \dots, N/2$$

FFCSIN computes the vectors A and B.

See reference: Singleton, Richard C., "On computing the fast Fourier transform", Comm. ACM, 10(10) 1967, 647-654.

Programming Notes

1. Coefficient B(1)=0 and A(1) is 2N times the average of the input data sequence.
2. In dimensioning the work area, IWK, where N is not a power of 2, the number of prime factors (F) in N/2 must be determined, where the factor "1" is disregarded. Thus, if N=36, the prime factors in 18(N/2) are 2, 3, and 3; F=3 and IWK has dimension 3(F+N)+26=143. For N=14, F=1 and IWK has dimension 71.

Example

DIMENSION X(4),A(3),B(3),IWK(2)

Input:

X=(1, -1, 1, -1)

N=4

CALL FFCSIN(X,N,A,B,IWK)

Output:

A=(0.0, 0.0, 8.0)

B=(0.0, 0.0, 0.0)

FFCSIN-1

SUBROUTINE FFRDR2 (A,M,IWK)

```

C                                     FFRD0010
C                                     FFRD0020
C-FFRDR2-----S-----LIBRARY 3-----FFRD0030
C                                     FFRD0040
C  FUNCTION                          - THIS SUBROUTINE PERMUTES A COMPLEX DATA VECTOR FFRD0050
C                                     IN REVERSE BINARY ORDER TO NORMAL ORDER. THE FFRD0060
C                                     ROUTINE CAN ALSO BE USED TO PERMUTE A COM- FFRD0070
C                                     PLEX DATA VECTOR IN NORMAL ORDER TO REVERSE FFRD0080
C                                     BINARY ORDER SINCE THE PERMUTATION IS SYM- FFRD0090
C                                     METRIC. FFRD0100
C  USAGE                             - CALL FFRDR2(A,M,IWK) FFRD0110
C  PARAMETERS   A                    - COMPLEX VECTOR OF LENGTH N=2**M WHICH CONTAINS FFRD0120
C                                     ON INPUT THE DATA TO BE PERMUTED. ON OUTPUT, FFRD0130
C                                     A CONTAINS THE PERMUTED DATA VECTOR. FFRD0140
C                                     M FFRD0150
C                                     - N=2**M IS THE NUMBER OF DATA POINTS, FFRD0160
C                                     IWK - WORK AREA VECTOR OF LENGTH M+1 FFRD0170
C  PRECISION                          - SINGLE FFRD0180
C  LANGUAGE                          - FORTRAN FFRD0190
C-----

```

CALL FFRDR2(A,M,IWK)

Purpose

FFRDR2 permutes a data vector in reverse binary order to normal order, or, it permutes a data vector in normal order to reverse binary order.

Algorithm

The reverse binary order function $r(k)$ is defined as follows:

Let the binary representation of k be $k = d_0 + d_1 2 + d_2 2^2 + \dots + d_{M-1} 2^{M-1}$.

Then $r(k) \equiv d_{M-1} + d_{M-2} 2 + d_{M-3} 2^2 + \dots + d_0 2^{M-1}$. FFRDR2 reorders $\{A_{r(k)+1}\}$, $k=0, \dots, 2^M-1$.

into the order $\{A_{k+1}\}$, $k=0, \dots, 2^M-1$.

Since $r(r(k))=k$, FFRDR2 can also be used to reorder $\{A_{k+1}\}$ $k=0, \dots, 2^M-1$ into the order $\{A_{r(k)+1}\}$, $k=0, \dots, 2^M-1$.

CALL FFTP (A,N,IWK,WK,LL)

FFTP computes the fast Fourier transform (FFT) of a complex vector of length N where N is any positive integer. Both input and output vectors are in normal order.

The output vector $\Lambda(\text{out})$ is defined mathematically as

$$A(\text{out})_{k+1} = \sum_{j=0}^{N-1} A(\text{in})_{j+1} e^{2\pi i j k / N} \quad \text{where } k=0, \dots, N-1 \text{ and } i=\text{SQRT}(-1)$$

See reference: Singleton, Richard C., "On computing the fast Fourier transform", Comm. ACM, 10(10) 1967, 647-654

1. Both input and output coefficients are given and computed in normal (natural) order.
2. In the calling sequence, the user may repeat the work area IWK three times. In the subroutine itself the calling sequence is coded as (A,N,IWK,WK,LL) since integer values and logical values are stored in the work area as well as real values.
3. In dimensioning the work area, IWK, the number of prime factors (F) in N must be determined, where the factor "1" is disregarded. Thus, if N=36, the prime factors are 2,2,3, and 3; F=4 and IWK has dimension 3F+6N+26=254. For N=7, F=1 and IWK has dimension 71.

IMSL has tested FFTP for $N=1,2,3,4,5,8,30,77$, and 128. For most of these values of N , multiple test cases were run. In each test case the coefficients of $A(\text{in})$ were on the unit circle. In all cases, the single precision version of FFTP produced absolute errors in the range 10^{-15} to 10^{-13} with some errors exactly equal to zero.

```

SUBROUTINE FFTR (A,GAMN,N,IWK)
C-----S-----LIBRARY 3-----
C
C FUNCTION - TO COMPUTE THE FAST FOURIER TRANSFORM OF A
C REAL DATA SEQUENCE.
C
C USAGE - CALL FFTR(A,GAMN,N,IWK)
C
C PARAMETERS A - REAL VECTOR OF LENGTH N WHICH, ON INPUT,
C CONTAINS THE DATA TO BE TRANSFORMED. ON
C OUTPUT, THE FIRST N/2 COMPLEX FOURIER COEFF-
C FICIENTS ARE STORED IN THE N LOCATIONS OF
C A. THE (N/2+1)TH FOURIER COEFFICIENT IS
C CONTAINED IN GAMN (DESCRIBED BELOW). THE
C REMAINING COEFFICIENTS MAY BE DETERMINED BY
C A(N+2-I)=COMPLEX CONJUGATE OF A(I)
C FOR I = 2,3,4,...,N/2.
C
C GAMN - COMPLEX VARIABLE WHICH, ON OUTPUT, CONTAINS
C THE (N/2+1)TH FOURIER COEFFICIENT.
C
C N - LENGTH OF DATA VECTOR A. N MUST BE A
C POSITIVE EVEN INTEGER.
C
C IWK - WORK STORAGE. IF N IS ANY INTEGRAL POWER OF 2,
C SAY N=2**M, THEN IWK MUST HAVE LENGTH M. IF
C N IS NOT A POWER OF 2, THEN IWK MUST HAVE
C LENGTH 3(F+N)+26, WHERE F IS THE NUMBER OF
C PRIME FACTORS IN N/2.
C
C PRECISION - SINGLE
C
C REQD. IMSL ROUTINES - FFRDR2,FFTP,FFT2
C
C LANGUAGE - FORTRAN
C-----

```

CALL FFTR(A,GAMN,N,IWK)

Purpose

FFTR computes the fast Fourier transform (FFT) of a real vector of length N where N is any positive even integer. Both input and output are in normal order.

Algorithm

The output vector A(out) is defined mathematically as

$$A(out)_{k+1} = \sum_{j=0}^{N-1} A(in)_{j+1} e^{2\pi i j k / N} \quad \text{where } k=0, \dots, N-1 \text{ and } i=\text{SQRT}(-1)$$

FFTR factors N into its prime factors and applies Cooley-Tukey techniques for each prime factor.

See reference: Singleton, Richard C., "On computing the fast Fourier transform", Comm. ACM, 10(10) 1967, 647-654.

Programming Notes

1. On input A is real. On output A is complex. Thus on output A(1) contains the real part of the first coefficient and A(2) contains the imaginary part. In order to avoid the differences in subscripting, the user might program, for instance

```

DIMENSION IWK(335),A(100),C(50)
INTEGER IWK
COMPLEX C,GAMN
EQUIVALENCE (A,C)
N=100

```

CALL FFTR(A,GAMN,N,IWK)

FFTR-1

2. In dimensioning the work area, IMK, where N is not a power of 2, the number of prime factors (F) in $N/2$ must be determined, where the factor "1" is disregarded. Thus, if $N=36$, the prime factors in $18(N/2)$ are 2, 3, and 3; $F=3$ and IMK has dimension $3(F+N)+26=143$. For $N=14$, $F=1$ and IMK has dimension 71.

Accuracy

INSL has tested FFTR for $N=2, 4, 8, 30, 128$, and 154. For most of these values of N , multiple tests were run. In each case the coefficients of $A(in)$ were in the unit interval. In all cases, FFTR produced absolute errors in the range 10^{-16} to 10^{-14} with some errors exactly equal to zero.

SUBROUTINE FFT2 (A,M,IWK)

FFT20010

FFT20020

FFT20030

FFT20040

FFT20050

FFT20060

FFT20070

FFT20080

FFT20090

FFT20100

FFT20110

FFT20120

FFT20130

FFT20140

FFT20150

FFT20160

FFT20170

FFT20180

C

C-FFT2-----3-----LIBRARY 3-----

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C-----

FUNCTION

- COMPUTE THE FAST FOURIER TRANSFORM, GIVEN A
COMPLEX VECTOR OF LENGTH EQUAL TO A POWER
OF TWO

USAGE

- CALL FFT2(A,M,IWK)

PARAMETERS A

- COMPLEX VECTOR OF LENGTH N=2**M WHICH CONTAINS
ON INPUT THE DATA TO BE TRANSFORMED (ASSUMED
TO BE IN NORMAL ORDER). ON OUTPUT, A CON-
TAINS THE FOURIER COEFFICIENTS STORED IN
REVERSE BINARY ORDER.

M

- N = 2**M IS THE NUMBER OF DATA POINTS.

IWK

- WORK AREA VECTOR OF LENGTH M+1.

PRECISION

- SINGLE

LANGUAGE

- FORTRAN

CALL FFT2(A,M,IWK)

Purpose

FFT2 computes the fast Fourier transform (FFT) of a complex vector of length equal to a power of two. The coefficients of the input vector A are in normal order. The coefficients of the output transformed vector overwrite the input coefficients and are stored in reverse binary order.

Algorithm

The output vector A(out) is defined mathematically as $A(out)_{k+1} = \sum_{j=0}^{n-1} A(in)_{j+1} e^{2\pi i j k / n}$

where $k=0,1,\dots,n-1$; $n=2^M$; $i=\text{SQRT}(-1)$; $l=r(k)$

The function $r(k)$ denotes reverse binary order. Let the binary representation of k be

$$k = d_0 + d_1 2 + d_2 2^2 + \dots + d_{M-1} 2^{M-1}, \text{ then } r(k) = d_{M-1} + d_{M-2} 2 + d_{M-3} 2^2 + \dots + d_0 2^{M-1}.$$

Computationally, the above summation which defines A(out) requires n^2 complex multiplications and additions. FFT2 utilizes a modification of the Singleton version of the Cooley-Tukey FFT algorithm. This algorithm requires only $n \log_2 n$ basic sets of operations.

See reference: Singleton, Richard C., "On computing the fast Fourier transform", Comm. ACM, 10(10)1967, 647-654.

Programming Notes

FFT2 can be used to compute the inverse Fourier transform $A(out)_{k+1} = \frac{1}{n} \sum_{j=0}^{n-1} A(in)_{j+1} e^{-2\pi i j k / n}$

by performing the following four steps:

- Conjugate A(in); i.e., $A^*(in) = \text{Re}(A(in)) - i \text{Im}(A(in))$
- Call FFT2 using $A^*(in)$ as the input vector
- Conjugate the output vector from FFT2
- Divide each element of the conjugated output vector by n , the length of the vector (i.e., the number of data points)

Accuracy

Let $A = (0.0, 1.0, \underbrace{0.0, \dots, 0.0}_{126 \text{ times}})$

FFT2 will produce results with absolute error in the range 10^{-15} to 10^{-13} uniformly.

Example

$A = (0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)$

$M=3$

Output:

$A = (1.0, -1.0, 1, -1, 0.70711 + 0.70711i, -0.70711 - 0.70711i, -0.70711 + 0.70711i, 0.70711 - 0.70711i)$

```

SUBROUTINE FFT2RV (A,M,IWK)
C
C-FFT2RV-----S-----LIBRARY 3-----
C
C  FUNCTION          - COMPUTE THE FAST FOURIER TRANSFORM, GIVEN A
C                     COMPLEX VECTOR OF LENGTH EQUAL TO A POWER
C                     OF TWO (DATA IN REVERSE BINARY ORDER)
C  USAGE            - CALL FFT2RV(A,M,IWK)
C  PARAMETERS      A  - COMPLEX VECTOR OF LENGTH N=2**M WHICH CONTAINS
C                     ON INPUT THE DATA TO BE TRANSFORMED (ASSUMED
C                     TO BE IN REVERSE BINARY ORDER). ON OUTPUT,
C                     A CONTAINS THE FOURIER COEFFICIENTS STORED
C                     IN NORMAL ORDER.
C                     M  - N = 2**M IS THE NUMBER OF DATA POINTS.
C                     IWK - WORK AREA VECTOR OF LENGTH M+1
C  PRECISION        - SINGLE
C  LANGUAGE         - FORTRAN
C-----

```

CALL FFT2RV(A,M,IWK)

Purpose

FFT2RV computes the Fast Fourier Transform (FFT) of a complex vector of length equal to a power of two. The coefficients of the input vector A are in reverse binary order. The coefficients of the output transformed vector overwrite the input coefficients and are stored in normal order.

Algorithm

The output vector A(out) is defined mathematically as

$$A(\text{out})_{k+1} = \sum_{j=0}^{n-1} A(\text{in})_{r(j)+1} e^{2\pi i j k / n} \quad \text{where } k=0,1,\dots,n-1; n=2^M \text{ and } i=\text{SQRT}(-1).$$

The function r(j) denotes reverse binary order. Let the binary representation of j be

$$j = d_0 + d_1 2^1 + \dots + d_{M-1} 2^{M-1}; \text{ then } r(j) = d_{M-1} + d_{M-2} 2^1 + d_{M-3} 2^2 + \dots + d_0 2^{M-1}$$

Computationally, the above summation which defines A(out) requires n^2 complex multiplications and additions. FFT2RV utilizes a modification of the Singleton version of the Cooley-Tukey FFT algorithm. This algorithm requires only $n \log_2 n$ basic sets of operations.

See reference: Singleton, Richard C., "On computing the fast Fourier transform", Comm. ACM, 10(10) 1967, 647-654.

Programming Notes

1. FFT2RV can be used to compute the inverse Fourier transform

$$A(\text{out})_{k+1} = \sum_{j=0}^{n-1} A(\text{in})_{r(j)+1} e^{-2\pi i j k / n}$$

by performing the following four steps:

- a. Conjugate A(in); i.e., $A^*(\text{in}) = \text{RE}(A(\text{in})) - i\text{IM}(A(\text{in}))$
- b. Call FFT2RV using $A^*(\text{in})$ as the input vector
- c. Conjugate the output vector from FFT2RV
- d. Divide by n, the length of the vector (i.e., the number of data points)

Accuracy

Let $A = (\underbrace{0, 0, \dots, 0}_{64 \text{ times}}, 1, \underbrace{0, 0, \dots, 0}_{63 \text{ times}})$

FFT2RV will produce results with absolute error in the range 10^{-15} to 10^{-13} uniformly.

Example

Input:

$A = (0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0)$
 $M = 3$

Output:

$A = (1.0, 0.70711 + 0.70711i, 1, -0.70711 + 0.70711i, -1.0, -0.70711 - 0.70711i, -i, 0.70711 - 0.70711i)$