# FFTPACK

Paul N. Swarztrauber
勝手な訳 桂田 祐史

1996 年 10 月 15 日

この文書の誤りを見つけた方は、勝手な訳者まで連絡して頂けると幸いです。連絡先: mk@math.meiji.ac.jp.

## 目 次

# 1　勝手な訳注

この文書は、有名な FFT ライブラリィ・パッケージである

FFTPACK
by Paul N. Swarztrauber

の INDEX ファイルを翻訳したものに、訳者が C プログラマー向けの注を補ったものです。

元々のプログラムは、FORTRAN の伝統的なプログラミング・スタイルで書かれていて、例えば配列の添字の下限は 1 であると仮定してあります。FFT を学んだ人は、時には添字の下限を 0 にした方が、すっきりした表現になることを知っているでしょう。C 言語の場合は、FORTRAN とは異なり、配列の添字は 0 から始まるので、自然にプログラムを書けば、すっきりとした表現になるわけです。そうした場合、元の文書に書いてある説明とは食い違いが出て来ますので、そこのところは訳者が説明を補っています。なお、一部では、配列の添字の下限を 1 からにした方が自然な場合もあります。訳者の方針は、C 言語からの利用の仕方を説明するところでは、 1 要素程度のメモリーの無駄は無視して、なるべく自然な表現になるようにすることでした。なお、FORTRAN 77 では、配列の添字の下限を好きなところから始めるように宣言できますから、無駄がなく、しかも自然な書き方をすることが可能です。その場合にも、 C 言語向けの注は参考になると思います。

# 2　コピーライトと内容一覧

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                  version 4  april 1985

     a package of fortran subprograms for the fast fourier
      transform of periodic and other symmetric sequences

                         by

                  paul n swarztrauber

  national center for atmospheric research  boulder,colorado 80307

   which is sponsored by the national science foundation

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

このパッケージは、以下に列挙するような、実および複素周期数列やある種の対称数列に対する高速フーリエ変換のプログラムから構成されています。

**rffti** rfftf と rfftb を初期化する

**rfftf** 実周期数列の順変換

**rfftb** 実係数配列の逆変換

**ezffti** ezfftf と ezfftb を初期化する

**ezfftf** 単純化された実周期の順変換

**ezfftb** 単純化された実周期の逆変換

**sinti** sint を初期化する

**sint** 実奇関数列のサイン変換

**costi** cost を初期化する

**cost** 実偶関数列のコサイン変換

**sinqi** sinqf と sinqb を初期化する

**sinqf** 奇波数の順サイン変換

**sinqb** sinqf の非正規化逆変換

**cosqi** cosqf と cosqb の初期化

**cosqf** 奇波数の順コサイン変換

**cosqb** cosqf の非正規化逆変換

**cffti** cfftf と cfftb の初期化

**cfftf** 複素周期数列の順変換

**cfftb** 複素周期数列の非正規化逆変換

# 3 各関数の一覧

## 3.1 rffti

### 機能と引数並び

**FORTRAN の場合**

`subroutine` **rffti**($n$, $wsave$)
`integer` $n$
`real` $wsave$(*)

**C の場合**

`void` **rffti**(`int` $n$, `REAL` $wsave$[]);

サブルーチン **rffti** は **rfftf** と **rfftb** の両者で使われる配列 $wsave$ を初期化する。$n$ の素因数分解と三角関数の表が計算され、$wsave$ に格納される。

### 入力パラメーター

$n$ 変換する数列の長さ

#### 出力パラメーター

$wsave$ 少なくとも長さ $2n+15$ の作業用配列。 $n$ が変更されない限り、**rfftf** と **rfftb** の双方で同じ作業用配列を使うことが出来る。$n$ の異なる値に対しては異なる配列 $wsave$ が必要になる。$wsave$ の内容は **rfftf** もしくは **rfftb** の呼び出しまで変更してはならない。

## 3.2 rfftf

#### 機能と引数並び

**FORTRAN の場合**
subroutine **rfftf**($n$,$r$,$wsave$)
integer $n$
real $r$(*),$wsave$(*)

**C の場合**
void **rfftf**(int $n$, REAL $r$[], REAL $wsave$[]);

サブルーチン **rfftf** は実周期数列の Fourier 係数を計算する (Fourier 解析[1])。この変換は後述の出力パラメーター $r$ のところで定義されるものである。

#### 入力パラメーター

$n$ 変換される配列 $r$ の長さ。その方法は $n$ が小さな素数の積である時にとても効率的である。異なる作業用配列を与えるならば $n$ を変えてもよい。

$r$ 変換される数列を含む長さ $n$ の実数型配列。

$wsave$ **rfftf** を呼ぶプログラムにおける少なくとも長さ $2n+15$ の作業用配列。配列 $wsave$ はサブルーチン **rffti**($n$,$wsave$) を呼ぶことにより初期化されねばならない。異なる $n$ の値に対しては、異なる配列 $wsave$ が使われねばならない。この初期化は $n$ が変更されない限り、繰り返す必要はないので、引き続く変換は最初よりも速く得られる。**rfftf** と **rfftb** では同じ配列 $wsave$ が使える。

#### 出力パラメーター

$r$ Fourier 係数が入ります。

**FORTRAN の場合** まず

$$r(1) = \sum_{i=1}^{n} r(i).$$

[1] ここでは、周期関数の関数値の列から Fourier 係数を求めることを Fourier 解析 (Fourier Analysis)」と呼んでいる。

もしも $n$ が偶数の場合 $\ell = n/2$, $n$ が奇数の場合 $\ell = (n+1)/2$ とおく。
$k = 2, \ldots, l$ に対して

$$r(2k-2) = \sum_{i=1}^{n} r(i) \cos\left(\frac{2\pi(k-1)(i-1)}{n}\right),$$

$$r(2k-1) = \sum_{i=1}^{n} -r(i) \sin\left(\frac{2\pi(k-1)(i-1)}{n}\right).$$

もしも $n$ が偶数ならば

$$r(n) = \sum_{i=1}^{n} (-1)^{i-1} r(i).$$

**C の場合** まず

$$r[0] = \sum_{i=0}^{n-1} r[i].$$

もしも $n$ が偶数の場合 $\ell = n/2 - 1$, $n$ が奇数の場合 $\ell = (n-1)/2$ とおく。$k = 1, \ldots, \ell$ に対して

$$r[2k-1] = \sum_{i=0}^{n-1} r[i] \cos\frac{2\pi ki}{n}, \quad r[2k] = -\sum_{i=0}^{n-1} r[i] \sin\frac{2\pi ki}{n}.$$

もしも $n$ が偶数ならば

$$r[n-1] = \sum_{i=0}^{n-1} (-1)^{i} r[i].$$

**wsave** **rfftf** もしくは **rfftb** を呼び出すまで壊してはならない計算結果を含む。

**注意**　この変換は正規化されていないので、**rfftf** に続いて **rfftb** を呼び出すと、入力列を $n$ 倍することになる。

## 3.3 rfftb

### 機能と引数並び

**FORTRAN の場合**

subroutine **rfftb**($n$,$r$,$wsave$)
integer $n$
real $r$(*), $wsave$(*)

**C の場合**

void **rfftb**(int $n$, REAL $r$[], REAL $wsave$[])

サブルーチン **rfftb** は、実周期数列をその Fourier 係数から計算する (Fourier synthesis)。この変換は後述の出力パラメーター $r$ のところで定義される。

### 入力パラメーター

$n$ 変換される配列 $r$ の長さ。その方法は $n$ が小さな素数の積である時にとても効率的である。異なる作業用配列を与えるならば $n$ を変えてもよい。

$r$ 変換される数列を含む長さ $n$ の実数配列。

$wsave$ **rfftb** を呼ぶプログラムにおける少なくとも長さ $2n+15$ の作業用配列。配列 $wsave$ はサブルーチン **rffti**($n$,$wsave$) を呼ぶことにより初期化されねばならない。異なる $n$ の値に対しては、異なる配列 $wsave$ が使われねばならない。この初期化は $n$ が変更されない限り、繰り返す必要はないので、引き続く変換は最初よりも速く得られる。

### 出力パラメーター

$r$ 関数値の列が入る。

**FORTRAN の場合** 偶数 $n$ に対しては、$i = 1, \cdots, n$ に対して、

$$\begin{aligned} r(i) &= r(1) + (-1)^{i-1} r(n) \\ &\quad + \sum_{k=2}^{n/2} \left( 2r(2k-2) \cos \frac{2\pi(k-1)(i-1)}{n} - 2r(2k-1) \sin \frac{2\pi(k-1)(i-1)}{n} \right). \end{aligned}$$

奇数 $n$ に対しては、$i = 1, \cdots, n$ に対して、

$$r(i) = r(1) + \sum_{k=2}^{(n+1)/2} \left( 2r(2k-2) \cos \frac{2\pi(k-1)(i-1)}{n} - 2r(2k-1) \sin \frac{2\pi(k-1)(i-1)}{n} \right).$$

**C の場合** 偶数 $n$ に対しては、$i = 0, \cdots, n-1$ に対して、

$$r[i] = r[0] + (-1)^i r[n-1] + \sum_{k=1}^{n/2-1} \left( 2r[2k-1] \cos \frac{2\pi ki}{n} - 2r[2k] \sin \frac{2\pi ki}{n} \right).$$

奇数 $n$ に対しては、$i = 0, \cdots, n-1$ に対して、

$$r[i] = r[0] + \sum_{k=1}^{(n-1)/2} \left( 2r[2k-1] \cos \frac{2\pi ki}{n} - 2r[2k] \sin \frac{2\pi ki}{n} \right).$$

**wsave** **rfftf** もしくは **rfftb** を呼び出すまで壊してはならない計算結果を含む。

**注意** この変換は正規化されていないので、**rfftf** に続いて **rfftb** を呼び出すと、入力列を $n$ 倍することになる[2]。

[2] これは誤植ではないかな。多分 $n/2$ 倍ではないかと思う。

## 3.4 ezffti

**機能と引数並び**

**FORTRAN の場合**
`subroutine` **ezffti**($n$,$wsave$)
`integer` $n$
`real` $wsave$(*)

**C の場合**
`void` **ezffti**(`int` $n$, `REAL` $wsave$[])

サブルーチン **ezffti** は **ezfftf** と **ezfftb** の両方で使われる配列 $wsave$ を初期化する。$n$ の素因数分解と三角関数の表が計算されて $wsave$ に格納される。

**入力パラメーター**

$n$ 変換する数列の長さ

**出力パラメーター**

$wsave$ 少なくとも長さ $3n + 15$ の作業用配列。$n$ が変更されない限り **ezfftf** と **ezfftb** の双方で同じ作業用配列が使える。異なる $n$ の値に対しては異なる配列 $wsave$ が要求される。

## 3.5 ezfftf

**機能と引数並び**

**FORTRAN の場合**
`subroutine` **ezfftf**($n$,$r$,$azero$,$a$,$b$,$wsave$)
`integer` $n$
`real` $r$(*),$azero$,$a$(*),$b$(*),$wsave$(*)

**C の場合**
`void` **ezfftf**(`int` $n$, `REAL` $r$[], `REAL` *$azero$, `REAL` $a$[], `REAL` $b$[], `REAL` $wsave$[])

サブルーチン **ezfftf** は実周期数列の Fourier 変換を計算する (Fourier 解析)。この変換は後述の出力パラメーター $azero$, $a$, $b$ のところで定義してある。 **ezfftf** は **rfftf** の、簡単だが、よりのろまなバージョンである。

**C プログラマーへのコメント** 配列 $r$[], $a$[] については、0 から要素を詰めて格納するのが自然だが、sin の項の Fourier 係数を納める $b$[] だけは、添字は 1 から始めた方が自然だと思われる。そこで、以下では、

REAL $r$[MAXN], $a$[MAXN/2+1], $b$[MAXN/2+1], $wsave$[3*MAXN+15];

のように宣言しておいて、

**ezfftf**($n$, $r$, &$a$[0], $a$+1, $\underline{b\text{+1}}$, $wsave$);

あるいは

**ezfftf**($n$, $r$, $a$, $a$+1, $\underline{b\text{+1}}$, $wsave$);

と呼び出して使うことを念頭に説明してある。$b[0]$ は未使用で無駄になっているが、これが嫌ならば、

**ezfftf**($n$, $r$, &$a$[0], $a$+1, $b$, $wsave$);

のようにすればよい (宣言も $b$[MAXN/2] で良くなる)。実は

**ezfftf**($n$, $r$, $a$, $b$, $wsave$);

のように呼び出せるようにインターフェイスを作ることも可能だが、 FORTRAN の場合の呼び出し方との互換性を重視してやめにした。

## 入力パラメーター

$n$ 変換される配列 $r$ の長さ。その方法は $n$ が小さな素数の積である時にとても効率的である。

$r$ 変換される数列を含む長さ $n$ の実数型配列。$r$ は破壊されない。

$wsave$ **ezfftf** を呼ぶプログラムにおける少なくとも長さ $3n+15$ の作業用配列。配列 $wsave$ はサブルーチン **ezffti**($n$,$wsave$) を呼ぶことにより初期化されねばならず、異なる $n$ の値に対しては、異なる配列 $wsave$ が使われねばならない。この初期化は $n$ が変更されない限り、繰り返す必要はないので、引き続く変換は最初よりも速く得られる。 **ezfftf** と **ezfftb** で同じ配列 $wsave$ が使える。

## 出力パラメーター

$azero$ FORTRAN の場合は $\displaystyle\sum_{i=1}^{n} r(i)/n$. C の場合は $\displaystyle\sum_{i=0}^{n-1} r[i]/n$.

$a$,$b$ Fourier 係数が入る。

**FORTRAN の場合** 偶数 $n$ に対しては $b(n/2)=0$ で $a(n/2)=\displaystyle\sum_{i=1}^{n}(-1)^{i-1}r(i)/n$.
偶数 $n$ に対しては $kmax=n/2-1$, 奇数 $n$ に対しては $kmax=(n-1)/2$ と定義し、$k=1,\cdots,kmax$ に対して

$$a(k)=\sum_{i=1}^{n}\frac{2}{n}r(i)\cos\frac{2\pi k(i-1)}{n},\quad b(k)=\sum_{i=1}^{n}\frac{2}{n}r(i)\sin\frac{2\pi k(i-1)}{n}.$$

**C の場合** 偶数 $n$ に対しては $b[n/2]=0$ で $a[n/2]=\sum_{i=0}^{n-1}(-1)^i r[i]/n$. 偶数 $n$ に対しては $kmax=n/2-1$, 奇数 $n$ に対しては $kmax=(n-1)/2$ と定義し、$k=1,\cdots,kmax$ に対して

$$a[k]=\frac{2}{n}\sum_{i=0}^{n-1}r[i]\cos\frac{2\pi ki}{n},\quad b[k]=\frac{2}{n}\sum_{i=0}^{n-1}r[i]\sin\frac{2\pi ki}{n}.$$

## 3.6 ezfftb

### 機能と引数並び

**FORTRAN の場合**

```
subroutine ezfftb(n,r,azero,a,b,wsave)
integer n
real r(*),azero,a(*),b(*),wsave(*)
```

**C の場合**

```
void ezfftb(int n, REAL r[], REAL *azero, REAL a[], REAL b[], REAL wsave[])
```

サブルーチン **ezfftb** は実周期数列の Fourier 変換を計算する (Fourier 同調)。この変換は後述の出力パラメーター $r$ のところで定義してある。**ezfftb** は **rfftb** の、簡単だが、よりのろまなバージョンである。

**C プログラマーへのコメント**　まず **ezfftf** に対してのコメントはここでも有効である。そこで、以下では、

```
REAL r[MAXN], a[MAXN/2+1], b[MAXN/2+1], wsave[3*MAXN+15];
```

のように宣言しておいて、

```
ezfftb(n, r, &a[0], a+1, b+1, wsave);
```

あるいは

```
ezfftb(n, r, a, a+1, b+1, wsave);
```

と呼び出して使うことを念頭に説明してある。$b[0]$ は未使用で無駄になっているが、これが嫌ならば、

```
ezfftb(n, r, &a[0], a+1, b, wsave);
```

のようにすればよい (宣言も `b[MAXN/2]` で良くなる)。なお、**ezfftf** とは異なり、$a$, $b$ は入力であるから、`&a[0]` でなく、`a[0]` を引数として渡すようにすることも可能であったが (というか、C のプログラムとしては、その方が自然)、**ezfftf** との対称性を重視して、このままにしてある。

## 入力パラメーター

$n$ 変換される配列 $r$ の長さ。その方法は $n$ が小さな素数の積である時にとても効率的である。

$azero$ 定数 Fourier 係数

**$a$, $b$** 残りの Fourier 係数を含む配列。これらの配列は破壊されない。これらの配列の長さは $n$ が偶数か奇数かによる。 $n$ が偶数の場合、$n/2$ だけの場所が必要である。 $n$ が奇数の場合、$(n-1)/2$ だけの場所が必要である。

$wsave$ **ezfftb** を呼ぶプログラムにおける少なくとも長さ $3n+15$ の作業用配列。配列 $wsave$ はサブルーチン **ezffti**($n$,$wsave$) を呼ぶことにより初期化されねばならず、異なる $n$ の値に対しては、異なる配列 $wsave$ が使われねばならない。この初期化は $n$ が変更されない限り、繰り返す必要はないので、引き続く変換は最初よりも速く得られる。 **ezfftf** と **ezfftb** で同じ配列 $wsave$ が使える。

## 出力パラメーター

$r$ 関数値の列が入る。

**FORTRAN の場合** $n$ が偶数ならば $kmax = n/2$, $n$ が奇数ならば $kmax = (n-1)/2$ と定義し、$i = 1, \cdots, n$ に対して

$$r(i) = azero + \sum_{k=1}^{kmax} \left( a(k) \cos \frac{2\pi k(i-1)}{n} + b(k) \sin \frac{2\pi k(i-1)}{n} \right).$$

複素数を用いて書くと、$j = 1, \cdots, n$ に対して

$$r(j) = \sum_{k=-kmax}^{kmax} c(k) \exp \frac{2\pi i k(j-1)}{n},$$

ここで

$$\begin{aligned} c(k) &= \frac{a(k) - ib(k)}{2}, \quad c(-k) = \overline{c(k)} \quad (k = 1, \cdots, kmax), \\ c(0) &= azero, , \end{aligned}$$

ただし $i = \sqrt{-1}$.

*************** 振幅 – 位相による記述 **********************

**FORTRAN の場合** $i = 1, 2, \cdots, n$ に対して

$$r(i) = azero + \sum_{k=1}^{kmax} \alpha_k \cos \left( \frac{2\pi k(i-1)}{n} + \beta_k \right),$$

ここで

$$\alpha_k = \sqrt{a_k^2 + b_k^2}, \quad \cos\beta_k = a(k)/\alpha_k, \quad \sin\beta_k = -b(k)/\alpha_k.$$

**C の場合** $i = 0, 1, \cdots, n-1$ に対して

$$r[i] = azero + \sum_{k=1}^{kmax} \alpha_k \cos\left(\frac{2\pi ki}{n} + \beta_k\right),$$

ここで

$$\alpha_k = \sqrt{a_k^2 + b_k^2}, \quad \cos\beta_k = a(k)/\alpha_k, \quad \sin\beta_k = -b(k)/\alpha_k.$$

## 3.7 sinti

### 機能と引数並び

`FORTRAN` **の場合**

`subroutine` **sinti**($n$,$wsave$)

`integer` $n$

`real` $wsave$(*)

`C` **の場合**

`void` **sinti**(`int` $n$, `REAL` $wsave$[])

サブルーチン **sinti** は **sint** で使われる配列 $wsave$ を初期化する。 $n$ の素因数分解と三角関数の表が $wsave$ に格納される。

### 入力パラメーター

$n$ 変換される配列 $r$ の長さ。その方法は $n+1$ が小さな素数の積である時にとても効率的である。

### 出力パラメーター

$wsave$ 少なくとも $\mathrm{int}(2.5n + 15)$ を配置するだけの長さの作業用配列。異なる $n$ の値に対しては、異なる配列 $wsave$ が要求される。$wsave$ の内容は **sint** の呼び出しまでは変更してはならない。

## 3.8 sint

### 機能と引数並び

`FORTRAN` **の場合**

`subroutine` **sint**($n$,$x$,$wsave$)

`integer` $n$

`real` $x$(*),$wsave$(*)

`C` **の場合**

`void` **sint**(`int` $n$, `REAL` $x$[], `REAL` $wsave$[])

サブルーチン **sint** は奇関数列 $x(i)$ の離散 Fourier サイン変換を計算する。この変換は後述の出力パラメーター $x$ のところで定義される。

**sint** は自分自身の非正規化逆変換である。なぜならば、**sint** 呼び出しに続けてもう一度 **sint** を呼び出すと、入力数列 $x$ は $2(n+1)$ 倍されるからである。

サブルーチン **sint** で使われる配列 $wsave$ はサブルーチン **sinti**($n$,$wsave$) 呼び出しで初期化せねばならない。

**C プログラマーへのコメント**　配列 $x$[] には、sin に対応する Fourier 係数が格納されるので、添字は 1 から始まるとした方が自然である。そこで、以下では

REAL $x$[MAXN+1];

のように宣言しておいて、

**sint**(n，<u>$x$+1</u>，$wsave$);

のように呼び出すことを仮定して説明をした。こうした場合、FORTRAN と C で、式の定義に違いはなくなる (ただ $r(\cdot)$ を $r[\cdot]$ として読み代えるだけ)。

### 入力パラメーター

- $n$ 変換される配列 $r$ の長さ。その方法は $n+1$ が小さな素数の積である時にとても効率的である。
- $x$ 変換される数列を含む長さ $n$ の配列。
- $wsave$ **sint** を呼ぶプログラムにおける少なくとも長さ int$(2.5n+15)$ の作業用配列。配列 $wsave$ はサブルーチン **sinti**($n$,$wsave$) を呼ぶことにより初期化されねばならず、異なる $n$ の値に対しては、異なる配列 $wsave$ が使われねばならない。この初期化は $n$ が変更されない限り、繰り返す必要はないので、引き続く変換は最初よりも速く得られる。

### 出力パラメーター

- $x$ $i=1,\cdots,n$ に対して

$$x(i)=\sum_{k=1}^{n} 2x(k)\sin\frac{ki\pi}{n+1}.$$

  **sint** 呼び出しに引き続き、もう一度 **sint** を呼び出すと数列 $x$ は $2(n+1)$ 倍される。それゆえ **sint** は自分自身の非正規化逆変換であると言える。
- $wsave$ **sint** の呼び出しまで壊してはならない、初期化計算の結果を含む。

## 3.9 costi

### 機能と引数並び

FORTRAN **の場合**

subroutine **costi**($n$,$wsave$)

integer $n$

real $wsave$(*)

C **の場合**

void **costi**(int $n$, REAL $wsave$[])

サブルーチン **costi** は **cost** で使われる配列 $wsave$ を初期化する。 $n$ の素因数分解と三角関数の表が $wsave$ に格納される。

### 入力パラメーター

$n$ 変換される配列 $r$ の長さ。その方法は $n-1$ が小さな素数の積である時にとても効率的である。

### 出力パラメーター

$wsave$ 少なくとも長さ $3n+15$ の作業用配列。異なる $n$ の値に対しては、異なる配列 $wsave$ が要求される。$wsave$ の内容は **cost** の呼び出しまでは変更してはならない。

## 3.10 cost

### 機能と引数並び

FORTRAN **の場合**

subroutine **cost**($n$,$x$,$wsave$)

integer $n$

real $x$(*),$wsave$(*)

C **の場合**

void **cost**(int $n$, REAL $x$[], REAL $wsave$[])

サブルーチン **cost** は偶関数列 $x(i)$ の離散 Fourier コサイン変換を計算する。この変換は後述の出力パラメーター $x$ のところで定義される。

**cost** は自分自身の非正規化逆変換である。なぜならば、**cost** 呼び出しに続けてもう一度 **cost** を呼び出すと、入力数列 $x$ は $2(n-1)$ 倍されるからである。

サブルーチン **cost** で使われる配列 $wsave$ はサブルーチン **costi**($n$,$wsave$) 呼び出しで初期化せねばならない。

### 入力パラメーター

$n$ 変換される配列 $r$ の長さ。$n$ は 1 より大きくなければならない。その方法は $n-1$ が小さな素数の積である時にとても効率的である。

$x$ 変換される数列を含む長さ $n$ の配列。

$wsave$ **cost** を呼ぶプログラムにおける少なくとも長さ $3n+15$ の作業用配列。配列 $wsave$ はサブルーチン **costi**($n$,$wsave$) を呼ぶことにより初期化されねばならず、異なる $n$ の値に対しては、異なる配列 $wsave$ が使われねばならない。この初期化は $n$ が変更されない限り、繰り返す必要はないので、引き続く変換は最初よりも速く得られる。

### 出力パラメーター

$x$ 変換された数列。

**FORTRAN の場合** $i=1,\cdots,n$ に対して

$$x(i)=x(1)+(-1)^{i-1}x(n)+\sum_{k=2}^{n-1}2x(k)\cos\frac{(k-1)(i-1)\pi}{n-1}.$$

**cost** 呼び出しに引き続き、もう一度 **cost** を呼び出すと数列 $x$ は $2(n-1)$ 倍される。それゆえ **cost** は自分自身の非正規化逆変換であると言える。

**C の場合** $i=0,\cdots,n-1$ に対して

$$x[i]=x[0]+(-1)^{i}x[n-1]+\sum_{k=1}^{n-2}2x[k]\cos\frac{ki\pi}{n-1}.$$

**cost** 呼び出しに引き続き、もう一度 **cost** を呼び出すと数列 $x$ は $2(n-1)$ 倍される。それゆえ **cost** は自分自身の非正規化逆変換であると言える。

$wsave$ **cost** の呼び出しまで壊してはならない、初期化計算の結果を含む。

## 3.11 sinqi

### 機能と引数並び

**FORTRAN の場合**

```
subroutine sinqi(n,wsave)
integer n
real wsave(*)
```

**C の場合**

```
void sinqi(int n, REAL wsave[])
```

サブルーチン **sinqi** は **sinqf** と **sinqb** の双方で使われる配列 $wsave$ を初期化する。 $n$ の素因数分解と三角関数の表が $wsave$ に格納される。

**入力パラメーター**

$n$ 変換する数列の長さ。その方法は $n$ が小さな素数の積である時にとても効率的である。

**出力パラメーター**

*wsave* 少なくとも長さ $3n+15$ の作業用配列。$n$ が変更されない限り、**sinqf** と **sinqb** の双方で同じ作業用配列が使える。異なる $n$ の値に対しては、異なる配列 *wsave* が使われねばならない。*wsave* の内容は **sinqf** あるいは **sinqb** の呼び出しまでは変更してはならない。

## 3.12 sinqf

**機能と引数並び**

`FORTRAN` **の場合**

subroutine **sinqf**($n$,$x$,*wsave*)
integer $n$
real $x$(*),*wsave*(*)

`C` **の場合**

void **sinqf**(int $n$, REAL $x$[], REAL *wsave*[])

サブルーチン **sinqf** は quarter wave data の高速 Fourier 変換を計算する。すなわち、**sinqf** は奇波数を持ったサイン級数表現から数列を計算する。この変換は後述の出力パラメーター $x$ のところで定義される。

**sinqb** は **sinqf** の非正規化逆変換である。なぜならば、**sinqb** 呼び出しに続けて **sinqf** を呼び出すと、数列 $x$ は $4n$ 倍されるからである。

サブルーチン **sinqf** で使われる配列 *wsave* はサブルーチン **sinqi**($n$,*wsave*) 呼び出しで初期化せねばならない。

**C プログラマーへのコメント** 配列 $x$[] には、sin に対応する Fourier 係数が格納されるので、添字は 1 から始まるとした方が自然である。そこで、以下では

REAL $x$[MAXN+1];

のように宣言しておいて、

**sinqf**(n, $x$+1, *wsave*);

のように呼び出すことを仮定して説明をした。こうした場合、FORTRAN と C で、式の定義に違いはなくなる (ただ $r(\cdot)$ を $r[\cdot]$ として読み代えるだけ)。

### 入力パラメーター

$n$ 変換される配列 $x$ の長さ。その方法は $n$ が小さな素数の積である時にとても効率的である。

$x$ 変換される数列を含む長さ $n$ の配列。

$wsave$ **sinqf** を呼ぶプログラムにおける少なくとも長さ $3n+15$ の作業用配列。配列 $wsave$ はサブルーチン **sinqi**($n$,$wsave$) を呼ぶことにより初期化されねばならず、異なる $n$ の値に対しては、異なる配列 $wsave$ が使われねばならない。この初期化は $n$ が変更されない限り、繰り返す必要はないので、引き続く変換は最初よりも速く得られる。

### 出力パラメーター

$x$ $i = 1, \cdots, n$ に対して

$$x(i) = (-1)^{i-1}x(n) + \sum_{k=1}^{n-1} 2x(k)\sin\frac{(2i-1)k\pi}{2n}$$

**sinqf** 呼び出しに続いて **sinqb** を呼び出すと、数列 $x$ を $4n$ 倍することになる。それゆえ **sinqb** は **sinqf** の非正規化逆変換である。

$wsave$ **sinqf** や **sinqb** の呼び出しまで壊してはならない、初期化計算の結果を含む。

## 3.13 sinqb

### 機能と引数並び

`FORTRAN` **の場合**

```
subroutine sinqb(n,x,wsave)
integer n
real x(*),wsave(*)
```

`C` **の場合**

```
void sinqb(int n, REAL x[], REAL wsave[])
```

サブルーチン **sinqb** は quarter wave data の高速 Fourier 変換を計算する。すなわち、**sinqb** は奇波数を持ったサイン級数表現から数列を計算する。この変換は後述の出力パラメーター $x$ のところで定義される。

**sinqf** は **sinqb** の非正規化逆変換である。なぜならば、**sinqf** 呼び出しに続けて **sinqb** を呼び出すと、数列 $x$ は $4n$ 倍されるからである。

サブルーチン **sinqb** で使われる配列 $wsave$ はサブルーチン **sinqi**($n$,$wsave$) 呼び出しで初期化せねばならない。

**C プログラマーへのコメント**　配列 $x$[] には、sin に対応する Fourier 係数が格納されるので、添字は 1 から始まるとした方が自然である。そこで、以下では

REAL $x$[MAXN+1];

のように宣言しておいて、

**sinqb**(n，$\underline{x\texttt{+1}}$，$wsave$);

のように呼び出すことを仮定して説明をした。こうした場合、FORTRAN と C で、式の定義に違いはなくなる (ただ $r(\cdot)$ を $r[\cdot]$ として読み代えるだけ)。

### 入力パラメーター

$n$ 変換される配列 $x$ の長さ。その方法は $n$ が小さな素数の積である時にとても効率的である。

$x$ 変換される数列を含む長さ $n$ の配列。

$wsave$ **sinqb** を呼ぶプログラムにおける少なくとも長さ $3n + 15$ の作業用配列。配列 $wsave$ はサブルーチン **sinqi**($n$,$wsave$) を呼ぶことにより初期化されねばならず、異なる $n$ の値に対しては、異なる配列 $wsave$ が使われねばならない。この初期化は $n$ が変更されない限り、繰り返す必要はないので、引き続く変換は最初よりも速く得られる。

### 出力パラメーター

$x$ $i = 1, \cdots, n$ に対して

$$x(i) = \sum_{k=1}^{n} 4x(k) \sin \frac{(2k-1)i\pi}{2n}.$$

**sinqb** 呼び出しに続いて **sinqf** を呼び出すと、数列 $x$ を $4n$ 倍することになる。それゆえ **sinqf** は **sinqb** の非正規化逆変換である。

$wsave$ **sinqf** や **sinqb** の呼び出しまで壊してはならない、初期化計算の結果を含む。

## 3.14 cosqi

### 機能と引数並び

**FORTRAN の場合**

subroutine **cosqi**($n$,$wsave$)
integer $n$
real $wsave$(*)

**C の場合**

void **cosqi**(int $n$, REAL $wsave$[])

サブルーチン **cosqi** は **cosqf** と **cosqb** の双方で使われる配列 $wsave$ を初期化する。 $n$ の素因数分解と三角関数の表が $wsave$ に格納される。

### 入力パラメーター

$n$ 変換する配列の長さ。その方法は $n$ が小さな素数の積である時にとても効率的である。

### 出力パラメーター

$wsave$ 少なくとも長さ $3n+15$ の作業用配列。$n$ が変更されない限り、**cosqf** と **cosqb** の双方で同じ作業用配列が使える。異なる $n$ の値に対しては、異なる配列 $wsave$ が使われねばならない。$wsave$ の内容は **cosqf** あるいは **cosqb** の呼び出しまでは変更してはならない。

## 3.15 cosqf

### 機能と引数並び

FORTRAN **の場合**

subroutine **cosqf**($n$,$x$,$wsave$)
integer $n$
real $x$(*),$wsave$(*)

C **の場合**

void **cosqf**(int $n$, REAL $x$[], REAL $wsave$[])

サブルーチン **cosqf** は quarter wave data の高速 Fourier 変換を計算する。すなわち、**cosqb** は奇波数を持ったコサイン級数表現から係数を計算する。この変換は後述の出力パラメーター $x$ のところで定義される。

**cosqf** は **cosqb** の非正規化逆変換である。なぜならば、**cosqf** 呼び出しに続けて **cosqb** を呼び出すと、数列 $x$ は $4n$ 倍されるからである。

サブルーチン **cosqf** で使われる配列 $wsave$ はサブルーチン **cosqi**($n$,$wsave$) 呼び出しで初期化せねばならない。

### 入力パラメーター

$n$ 変換される配列 $x$ の長さ。その方法は $n$ が小さな素数の積である時にとても効率的である。

$x$ 変換される数列を含む長さ $n$ の配列。

$wsave$ **cosqf** を呼ぶプログラムにおける少なくとも長さ $3n+15$ の作業用配列。配列 $wsave$ はサブルーチン bf cosqi($n$,$wsave$) を呼ぶことにより初期化されねばならず、異なる $n$ の値に対しては、異なる配列 $wsave$ が使われねばならない。この初期化は $n$ が変更されない限り、繰り返す必要はないので、引き続く変換は最初よりも速く得られる。

**出力パラメーター**

$x$ Fourier 係数を格納する。

**FORTRAN の場合** $i = 1, 2, \cdots, n$ に対して

$$x(i) = x(1) + \sum_{k=2}^{n} 2x(k) \cos \frac{(2i-1)(k-1)\pi}{2n}.$$

**cosqf** 呼び出しに引き続き **cosqb** を呼び出すと数列 $x$ は $4n$ 倍される。それゆえ **cosqb** は **cosqf** の非正規化逆変換である。

**C の場合** $i = 0, 1, \cdots, n-1$ に対して

$$x[i] = x[0] + \sum_{k=1}^{n-1} 2x[k] \cos \frac{(2i+1)k\pi}{2n}.$$

**cosqf** 呼び出しに引き続き **cosqb** を呼び出すと数列 $x$ は $4n$ 倍される。それゆえ **cosqb** は **cosqf** の非正規化逆変換である。

$wsave$ **cosqf** や **cosqb** の呼び出しまで壊してはならない、初期化計算の結果を含む。

## 3.16 cosqb

**機能と引数並び**

**FORTRAN の場合**

```
subroutine cosqb(n,x,wsave)
integer n
real x(*),wsave(*)
```

**C の場合**

```
void cosqb(int n, REAL x[], REAL wsave[])
```

サブルーチン **cosqb** は quarter wave data の高速 Fourier 変換を計算する。すなわち、**cosqb** は奇波数を持ったコサイン級数表現から数列を計算する。この変換は後述の出力パラメーター $x$ のところで定義される。

**cosqb** は **cosqf** の非正規化逆変換である。なぜならば **cosqb** 呼び出しに続けて **cosqf** を呼び出すと数列 $x$ を $4n$ 倍することになるから。

サブルーチン **cosqb** で使われる配列 $wsave$ はサブルーチン **cosqi**($n$,$wsave$) の呼び出しで初期化せねばならない。

**入力パラメーター**

$n$ 変換される配列 $x$ の長さ。その方法は $n$ が小さな素数の積である時にとても効率的である。

$x$ 変換される数列を含む長さ $n$ の配列。

$wsave$ **cosqb** を呼ぶプログラムにおける少なくとも長さ $3n+15$ の作業用配列。配列 $wsave$ はサブルーチン **cosqi**($n$,$wsave$) を呼ぶことにより初期化されねばならず、異なる $n$ の値に対しては、異なる配列 $wsave$ が使われねばならない。この初期化は $n$ が変更されない限り、繰り返す必要はないので、引き続く変換は最初よりも速く得られる。

### 出力パラメーター

$x$ 関数値の列が格納される。

**FORTRAN の場合** $i = 1, \cdots, n$ に対して

$$x(i) = \sum_{k=1}^{n} 4x(k) \cos \frac{(2k-1)(i-1)\pi}{2n}.$$

**cosqb** 呼び出しに続けて **cosqf** を呼び出すと数列 $x$ を $4n$ 倍することになる。それゆえ **cosqf** は **cosqb** の非正規化逆変換である。

**C の場合** $i = 0, 1, \cdots, n-1$ に対して

$$x[i] = \sum_{k=0}^{n-1} 4x[k] \cos \frac{(2k+1)i\pi}{2n}.$$

**cosqb** 呼び出しに続けて **cosqf** を呼び出すと数列 $x$ を $4n$ 倍することになる。それゆえ **cosqf** は **cosqb** の非正規化逆変換である。

$wsave$ **cosqf** や **cosqb** の呼び出しまで壊してはならない、初期化計算の結果を含む。

## 3.17 cffti

### 機能と引数並び

**FORTRAN の場合**

```
subroutine cffti(n,wsave)
integer n
real wsave(*)
```

**C の場合**

```
void cffti(int n, REAL wsave[])
```

サブルーチン **cffti** は **cfftf** や **cfftb** の双方で使われる配列 $wsave$ を初期化する。$n$ の素因数分解と三角関数の表が $wsave$ に格納される。

### 入力パラメーター

$n$ 変換される数列の長さ。

### 出力パラメーター

$wsave$ 少なくとも長さ $4n + 15$ の作業用配列。$n$ が変更されない限り、**cfftf** と **cfftb** の双方で同じ作業用配列が使える。異なる $n$ の値に対しては、異なる配列 $wsave$ が使われねばならない。$wsave$ の内容は **cfftf** あるいは **cfftb** の呼び出しまでは変更してはならない。

## 3.18 cfftf

### 機能と引数並び

**FORTRAN の場合**

subroutine **cfftf**($n$,$c$,$wsave$)
integer $n$
real $wsave$(*)
complex $c$(*)

**C の場合**

void **cfftf**(int $n$, COMPLEX $c$[], REAL $wsave$[])

サブルーチン **cfftf** は複素離散 Fourier 変換を計算する (Fourier analysis)、すなわち **cfftf** 複素周期数列の Fourier 係数を計算する。この変換は後述の出力パラメーター $c$ のところで定義される。

この変換は正規化されていない。正規化した変換を得る為には出力を $n$ で割らなければならない。そうしないで、**cfftf** の呼び出しに続けて **cfftb** の呼び出しをすると、数列を $n$ 倍することになる。

サブルーチン **cfftf** で使われる配列 $wsave$ はサブルーチン **cffti**($n$,$wsave$) の呼び出しによって初期化せねばならない。

### 入力パラメーター

$n$ 複素数列 $c$ の長さ。その方法は $n$ が小さな素数の積である時にとても効率的である。

$c$ 数列を含む長さ $n$ の複素数配列。

$wsave$ **cfftf** を呼ぶプログラムにおける少なくとも長さ $4n + 15$ の作業用配列。配列 $wsave$ はサブルーチン **cffti**($n$,$wsave$) を呼ぶことにより初期化されねばならず、異なる $n$ の値に対しては、異なる配列 $wsave$ が使われねばならない。この初期化は $n$ が変更されない限り、繰り返す必要はないので、引き続く変換は最初よりも速く得られる。

### 出力パラメーター

$c$ Fourier 係数が格納される。

**FORTRAN の場合** $j = 1, \cdots, n$ に対して

$$c(j) = \sum_{k=1}^{n} c(k) \exp \frac{-2\pi i(j-1)(k-1)}{n},$$

ここで $i = \sqrt{-1}$.

**C の場合** $j = 0, 1, \cdots, n-1$ に対して

$$c[j] = \sum_{k=0}^{n-1} c[k] \exp \frac{-2\pi ijk}{n},$$

ここで $i = \sqrt{-1}$.

*wsave* サブルーチン **cfftf** もしくは **cfftb** の呼び出しまでの間に破壊してはならない初期化の計算結果を含む。

## 3.19 cfftb

### 機能と引数並び

**FORTRAN の場合**

```
subroutine cfftb(n,c,wsave)
integer n
real wsave(*)
complex c(*)
```

**C の場合**

```
void cfftb(int n, COMPLEX c[], REAL wsave[])
```

サブルーチン **cfftb** は複素離散 Fourier 逆変換を計算する (Fourier 同調)、すなわち **cfftb** 複素周期数列をその Fourier 係数から計算する。この変換は後述の出力パラメーター $c$ のところで定義される。

**cfftf** の呼び出しに続けて **cfftb** の呼び出しをすると、数列を $n$ 倍することになる。

サブルーチン **cfftb** で使われる配列 *wsave* はサブルーチン **cffti**($n$,*wsave*) の呼び出しによって初期化せねばならない。

### 入力パラメーター

$n$ 複素数列 $c$ の長さ。その方法は $n$ が小さな素数の積である時にとても効率的である。

$c$ 数列を含む長さ $n$ の複素数配列。

*wsave* **cfftb** を呼ぶプログラムにおける少なくとも長さ $4n+15$ の実作業用配列。配列 *wsave* はサブルーチン **cffti**($n$,*wsave*) を呼び出すことによって初期化しなければならず、異なる $n$ の値に対しては異なる配列 *wsave* を使わねばならない。**cfftf** と **cfftb** で同じ配列 *wsave* が使える。

**出力パラメーター**

$c$ 関数値の列が納められる。

**FORTRAN の場合** $j = 1, \cdots, n$ に対して

$$c(j) = \sum_{k=1}^{n} c(k) \exp \frac{i(j-1)(k-1)2\pi}{n},$$

ここで $i = \sqrt{-1}$.

**C の場合** $j = 0, 1, \cdots, n-1$ に対して

$$c[j] = \sum_{k=0}^{n-1} c[k] \exp \frac{2\pi ijk}{n},$$

ここで $i = \sqrt{-1}$.

*wsave* サブルーチン **cfftf** もしくは **cfftb** の呼び出しまでの間に破壊してはならない初期化の計算結果を含む。

[”send index for vfftpk” describes a vectorized version of fftpack]