FUJITSU

FUJITSU Software
Technical Computing Suite V4.0L20

Development Studio
高速４倍精度基本演算ライブラリ
使用手引書

J2UL-2576-01Z0(00)
2020年2月

まえがき

本書は高速 4 倍精度基本演算ライブラリ(以降 fast_dd と呼びます)の使い方について記述しています.

本書は以下のような構成になっています.

1. 概説

fast_dd の概要について説明します.

2. Fortran 版の利用方法

fast_dd を Fortran プログラムから利用する場合の利用方法について説明します.

3. C++版の利用方法

fast_dd を C++プログラムから利用する場合の利用方法について説明します.

4. エラーメッセージ

fast_dd のルーチンが出力するエラーメッセージについて説明します.

輸出管理規制について

本ドキュメントを輸出または第三者へ提供する場合は, お客様が居住する国および米国輸出管理関連法規等の規制をご確認のうえ, 必要な手続きをおとりください.

出版年月および版数

版数	マニュアルコード
2020 年　2 月　初版	J2UL-2576-01Z0(00)

著作権表示

お願い

- 本書を無断で他に転載しないようお願いします.
- 本書は予告なしに変更されることがあります.

目次

図目次

表目次

1. 概説

本章では高速 4 倍精度演算ライブラリの概要について説明します.

1.1 高速 4 倍精度基本演算ライブラリについて

コンピュータで数値計算を行う場合に, より高精度の計算を行いたい場面があります. 例えば以下のような場合です.

- 倍精度で数値計算を行うプログラムにおいて, 精度が重要な部分があり, その部分で 4 倍精度を使った計算を行いたい
- 倍精度で計算した結果の精度を検証するため, 4 倍精度で計算を行いたい

言語処理系で提供されている 4 倍精度を利用することもできますが, ソフトウェアエミュレーションしているため, 性能に課題がありました.

このような場合に, 高速 4 倍精度基本演算ライブラリ(fast_dd)を使うことで, 高い精度の計算を高速に計算することができます.

fast_dd は, 4 倍精度の値を double-double 形式で表現し, 演算を行うライブラリです. double-double 形式は倍精度実数型の変数 2 つで 4 倍精度変数を表現する形式です. 演算はプロセッサの持つ倍精度演算命令を使って行うため, 言語処理系が持つ 4 倍精度実数型に比べて高速に演算できます.

精度については, 言語処理系の 4 倍精度は 10 進で約 33 桁であるのに対し, double-double 形式では約 31 桁となります. アプリケーションがこの精度で十分に満足できる場合, fast_dd の利用により 4 倍精度演算を高速に行うことができます.

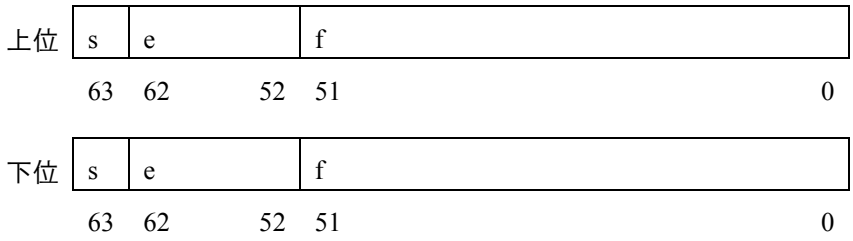
fast_dd は Fortran または C++で記述されたプログラムから利用することができます. Fortran 版と C++版では言語の仕様の違いから, 提供機能や使い方が異なっていますのでご注意ください.

fast_dd では代入, 四則演算, 比較, 数値関数, 数学関数などの機能をサポートしています. 代入, 四則演算, 比較などは Fortran や C++の演算子として定義しているため, 自然なスタイルでプログラムを書くことができます. また, 演算を高速に実行するためのマルチ・ベクトル演算ルーチンを提供しています. ルーチンはスレッドセーフになっており, OpenMP や自動並列化機能で並列化した処理の中で利用することができます.

1.2 double-double 形式

double-double 形式では次のように倍精度実数型の変数 2 つで 1 つの値を表現します.

図 1.2.1 double-double 形式



s : 符号 1 ビット

e : 指数部 11 ビット

f : 仮数部 52 ビット

fast_dd では double-double 形式の変数を次のような構造体で表しています.

Fortran 版の double-double 形式の実数

```
type dd_real
  real(8)::re(2)
end type dd_real
```

Fortran 版の double-double 形式の複素数

cmp(1:2)に実部, cmp(3:4)に虚部を格納します.

```
type dd_complex
  real(8)::cmp(4)
end type dd_complex
```

C++版の double-double 形式の実数

```
struct dd_real {
       double x[2];
};
```

2. Fortran 版の利用方法

fast_dd を Fortran プログラムから利用する方法について説明します. 最初に簡単なプログラム例を示し, その後, 使用できる各機能について説明します.

2.1 プログラム例

fast_dd を使用した Fortran プログラムの例を示します.

以下のプログラムは, double-double 形式を使ってテーラー展開で e (自然対数の底)の値を計算します.

```
!
!   MAIN PROGRAM
!
      program fast_dd_sample
      use fast_dd                          ! (1)
      type(dd_real)::x,y                   ! (2)
      type(dd_real)::dd_exp_sample
      real(16)::xx,yy,wy
!
!   call dd_exp_sample
!
      x=1.0d0
      y=dd_exp_sample(x)
!
!   call Fortran qexp
!
      xx=x
      yy=qexp(xx)
!
      wy=y
      write(6,100) wy                      ! (4)
      write(6,200) yy
 100  format('dd_exp_sample(1.0) :',F30.25 )
 200  format('qexp(1.0)         :',F30.25 )
      end
!
!   FUNCTION dd_exp_sample
!
      function dd_exp_sample(x)            ! (3)
      use fast_dd                          ! (1)
      type(dd_real)::x,dd_exp_sample       ! (2)
      type(dd_real)::wx,wy,wy0,wc
      real(8)::c
!
!    exp(x)=1 + x + x**2/2! + x**3/3! + x**4/4! + ...
!
      wx=x
      wy0=1.0d0
      c=2.0d0
      wc=1.0d0
```

```
wy =1.0d0+wx*wc    ! wy=1+x
do while(abs(wy0-wy)>=1d-25)
  wy0=wy
  wc=wc/c
  wx=wx*x
  wy=wy+wx*wc    ! wy=wy+x**i/i!
  c=c+1.0
end do
dd_exp_sample=wy
end
```

解説

(1) fast_dd を使用するプログラムではモジュール fast_dd を use します. use fast_dd は主プログラム, サブルーチン, 関数ごとに記述してください.

(2) double-double 形式の変数は type(dd_real)を指定して宣言します.

(3) dd_real 型の変数を, サブルーチンや関数の引数や復帰値に指定できます.

(4) fast_dd では入出力をサポートしていないため, 値の出力は Fortran の real(16)等を利用してください.

2.2 機能一覧

fast_dd Fortran 版で提供する機能を表 2.2.1～表 2.2.4に示します.

表 2.2.1 代入・演算

演算子	機能
=	代入
+, -, *, /	四則演算
-	単項マイナス
.EQ. , .NE. , .LT. , .LE. , .GT. , .GE. , == , /= , < , > , <= , >=	関係演算

表 2.2.2 数値関数

関数名	機能
ddreal	dd_real 型への型変換
ddcomplex	dd_complex 型への型変換
int	dd_real 型を integer 型へ変換
real	dd_complex 型の実部
dble	real(8)型への型変換
cmplx	complex(8)型への型変換
abs	絶対値
sign	符号の付け替え
max	最大値

関数名	機能
min	最小値
aint	整数値へ切り捨て
anint	小数部分を四捨五入
aimag	複素数の虚部
conjg	共役複素数

表 2.2.3 数学関数

関数名	機能
sqrt	平方根
exp	指数関数
log	対数関数
sin	正弦関数
cos	余弦関数
sincos	正弦関数と余弦関数

表 2.2.4 マルチ演算，ベクトル演算ルーチン

ルーチン名	機能
m2_add_dd	2 つの加算
m2_sum_dd	合計
m2_sub_dd	2 つの減算
m2_mul_dd	2 つの乗算
v_add_dd	ベクトルの加算
v_sub_dd	ベクトルの減算
v_mul_dd	ベクトルの乗算
vm_add_dd	ベクトルの加算(スレッド並列)
vm_sub_dd	ベクトルの減算(スレッド並列)
vm_mul_dd	ベクトルの乗算(スレッド並列)

2.3 fast_dd モジュール

fast_dd を Fortran プログラムから使用する場合，モジュール fast_dd を use します．use fast_dd は，fast_dd を使用する主プログラム，サブルーチン，関数ごとに記述してください．

例：主プログラムとサブルーチンの中で fast_dd を使用します．

```
program main
use fast_dd
```

```
   :
end
subroutine sub()
use fast_dd
   :
end subroutine
```

2.4 変数の宣言

fast_dd の Fortran 版では, double-double 形式の変数の型として, dd_real 型と dd_complex 型が提供されています. dd_real 型は実数, dd_complex 型は複素数です.

例 1：dd_real 型のスカラ変数 x と配列 a を宣言します.

```
type(dd_real):: x,a(100)
```

例 2：dd_complex 型のスカラ変数 zx と配列 za を宣言します.

```
type(dd_complex):: zx,za(100)
```

2.5 代入

dd_real 型または dd_complex 型を使った代入を行うことができます. さらに, real(8), real(16), integer(4), complex(8), complex(16)型との間で代入を行うことができます. サポートしている機能を表 2.5.1に示します.

表 2.5.1 代入

	左辺の型	右辺の型
代入(=)	dd_real	dd_real
	dd_complex	dd_complex
	real(8)	dd_real
	real(16)	dd_real
	integer(4)	dd_real
	dd_real	real(8)
	dd_real	real(16)
	dd_real	integer(4)
	dd_real	dd_complex
	complex(8)	dd_complex
	complex(16)	dd_complex
	real(8)	dd_complex
	dd_complex	dd_real
	dd_complex	complex(8)
	dd_complex	complex(16)
	dd_complex	real(8)
	dd_complex	integer(4)

例 1：代入

```
type(dd_real)::a,b
type(dd_complex)::za,zb
a=b
za=zb
```

例 2：integer(4)や real(8)の変数や値を dd_real 型変数に代入.

```
type(dd_real)::a
integer::n
real(8)::d
a=n
a=10
a=d
a=1.0d-5    ! real(8)の代入は精度に注意.
```

例 3：integer(4), real(8), complex(8)の変数や値を dd_complex 型に代入

```
type(dd_complex)::a
integer::n
real(8)::d
complex(8)::z
a=n
a=10
a=d
a=z
a=(1.0d0,2.0d0)
```

例 4：dd_real 型を integer(4)型や real(8)型の変数へ代入

```
type(dd_real)::a
integer::n
real(8)::d
d=a
n=a
```

例 5：dd_complex 型を real(8)型または complex(8)型の変数へ代入

```
type(dd_complex)::a
real(8)::d
complex(8)::z
d=a
z=a
```

例 6：real(16)型, complex(16)型の変数や値を dd_real 型, dd_complex 型の変数へ代入

```
type(dd_real)::a
type(dd_complex)::c
real(16)::q
complex(16)::z
a=q
a=0.1q-5
c=z
```

例 7：real(16)型, complex(16)型の変数への代入

```
type(dd_real)::x
type(dd_complex)::c
real(16)::q
complex(16)::z
q=x
z=c
```

2.6 演算

演算として四則演算 +, –, *, / が提供されています. また, 単項マイナス – を提供します.

四則演算では dd_real 型同士, dd_complex 型同士の演算のほかに, real(8)型や integer 型との演算をサポートしています. サポートしている機能は表 2.6.1～表 2.6.3のとおりです.

表 2.6.1 加減算

演算	左辺の型	右辺の型	結果の型
加算, 減算	dd_real	dd_real	dd_real
	real(8)	dd_real	dd_real
	dd_real	real(8)	dd_real
	integer(4)	dd_real	dd_real
	dd_real	integer(4)	dd_real
	dd_complex	dd_complex	dd_complex
	real(8)	dd_complex	dd_complex
	dd_complex	real(8)	dd_complex
	dd_complex	dd_real	dd_complex
	dd_real	dd_complex	dd_complex

表 2.6.2 乗算

演算	左辺の型	右辺の型	結果の型
乗算	dd_real	dd_real	dd_real
	real(8)	dd_real	dd_real
	dd_real	real(8)	dd_real
	integer(4)	dd_real	dd_real
	dd_real	integer(4)	dd_real
	dd_complex	dd_complex	dd_complex
	real(8)	dd_complex	dd_complex
	dd_complex	real(8)	dd_complex
	integer(4)	dd_complex	dd_complex
	dd_complex	integer(4)	dd_complex
	dd_complex	dd_real	dd_complex
	dd_real	dd_complex	dd_complex

表 2.6.3 除算

演算	左辺の型	右辺の型	結果の型
除算	dd_real	dd_real	dd_real
	real(8)	dd_real	dd_real
	dd_real	real(8)	dd_real

演算	左辺の型	右辺の型	結果の型
	integer(4)	dd_real	dd_real
	dd_real	integer(4)	dd_real
	dd_complex	dd_complex	dd_complex
	dd_complex	dd_real	dd_complex
	dd_real	dd_complex	dd_complex
	dd_complex	real(8)	dd_complex

例 1：dd_real 型同士の四則演算

```
type(dd_real)::a,b,c,d,e
c=a+b
c=a-b
c=a*b
c=a/b
c=a*b+d/e
c=-a            ! 単項マイナス
```

例 2：dd_complex 型同士の四則演算

```
type(dd_complex)::za,zb,zc,zd,ze
zc=za+zb
zc=za-zb
zc=za*zb
zc=za/zb
zc=za*zb+zd/ze;
```

例 3：dd_real 型の演算では，左辺または右辺に real(8)型の変数や定数を指定することができます.

```
type(dd_real)::a,b,c
real(8)::d
c=d+a
c=a-123.0d0
c=2.0d0*b
c=d/b
```

例 4：dd_real 型の演算では，左辺または右辺に integer(4)型の変数や定数を指定することができます.

```
type(dd_real)::a,b,c
integer(4)::n
c=a+123
c=a-n
c=a*n
c=a/2
```

2.7 関係演算

dd_real 型および dd_complex 型を使った比較を行うことができます.

表 2.7.1にサポートする関係演算について示します.

表 2.7.1 関係演算

関係演算子	左辺の型	右辺の型	結果の型
.EQ. , .NE. , == , /=	dd_real	dd_real	logical
	real(8)	dd_real	logical
	dd_real	real(8)	logical
	integer(4)	dd_real	logical
	dd_real	integer(4)	logical
	dd_complex	dd_complex	logical
	dd_real	dd_complex	logical
	dd_complex	dd_real	logical
.GT. , .GE. , .LT. , .LE. , > , >= , < , <=	dd_real	dd_real	logical
	real(8)	dd_real	logical
	dd_real	real(8)	logical
	integer(4)	dd_real	logical
	dd_real	integer(4)	logical

例 1：関係演算

```
type(dd_real)::a,b,c
type(dd_complex)::x,y
if(a>b) then
   c=0.0d0
end if
if(a>=b.and.(c/=0.0d0.or.c/=1.0d0)) then
   c=0.0d0
end if
if(x==y) then
   y=(0.0d0,0.0d0)
end if
```

2.8 数値関数

2.8.1 ddreal

呼び出し形式

y=ddreal(x)

x　　dd_real 型, integer 型, real(8)型または dd_complex 型

y　　dd_real 型

機能

x が dd_real 型, integer 型または real(8)型のとき, dd_real 型に変換します.

x が dd_complex 型のとき, x の実部を返します.

使用例

```
type(dd_real)::y
type(dd_complex)::z
```

```
y=ddreal(1.0d0)
y=ddreal(123)
y=ddreal(z)
```

2.8.2 ddcomplex

呼び出し形式

y=ddcomplex(xr,xi)

xr	dd_real 型
xi	dd_real 型
y	dd_complex 型

または,

y=ddcomplex(xr)

xr	dd_real 型, real(8)型または complex(8)型
y	dd_complex 型

機能

xr を実部, xi を虚部に持つ複素数を返します.

xi を省略したときは, xr を実部, 0.0 を虚部に持つ複素数を返します.

使用例

```
type(dd_complex)::y
type(dd_real)::r,c
y=ddcomplex(r,c)
y=ddcomplex(1.0d0)
```

2.8.3 int

呼び出し形式

n=int(x)

x	dd_real 型
n	integer 型

機能

dd_real 型の値を整数に変換します.

|x|<1.0 の場合, 結果の値は 0 となります.それ以外の場合は, x の絶対値以下で最大の整数値に x の符号を付けた値となります.

使用例

```
type(dd_real)::x
integer::n
n=int(x)+50
```

2.8.4 real

呼び出し形式

y=real(x)

x	dd_complex 型
y	dd_real 型

機能

x の実部を返します.

使用例

```
type(dd_real)::y
type(dd_complex)::x
y=real(x)
```

2.8.5 dble

呼び出し形式

y=dble(x)

x	dd_real 型
y	real(8)型

または,

y=dble(x)

x	dd_complex 型
y	real(8)型

機能

x が dd_real 型のとき, real(8)型に変換します.

x が dd_complex 型の場合, x の実部を real(8)型に変換します.

使用例

```
type(dd_real)::x
type(dd_complex)::z
real(8)::d,e
d=dble(x)+1.5
e=dble(z)
```

2.8.6 cmplx

呼び出し形式

y=cmplx(x)

x	dd_complex 型
y	complex(8)型

機能

dd_complex 型の値を, complex(8)型に変換します.

使用例

```
type(dd_complex)::x
complex(8)::z
z=cmplx(x)
```

2.8.7 abs

呼び出し形式

y=abs(x)

x	dd_real 型または dd_complex 型
y	dd_real 型

機能

絶対値を返します.

使用例

```
type(dd_real)::x,y
type(dd_complex)::z
y=abs(x)
y=abs(z)
```

2.8.8 sign

呼び出し形式

y=sign(x,s)

x	dd_real 型
s	dd_real 型または real(8)型
y	dd_real 型

機能

x の絶対値に s の符号をつけた値を返します.

使用例

```
type(dd_real)::x,y
x=10.0d0
y=sign(x,-1.0d0)      ! y は-10.0 になります.
```

2.8.9 max

呼び出し形式

y=max(x1,x2[,x3,...])

x1, x2,...	dd_real 型
y	dd_real 型

機能

全ての引数の中の最大値を求めます. 引数は 2～9 個指定することができます.

使用例

```
type(dd_real)::x1,x2,x3,x4,x5,x6,x7,x8,x9,y
y=max(x1,x2)
y=max(x1,x2,x3,x4,x5,x6,x7,x8,x9)
```

2.8.10 min

呼び出し形式

y=min(x1,x2[,x3,...])

x1, x2,...	dd_real 型
y	dd_real 型

機能

全ての引数の中の最小値を求めます. 引数は 2～9 個指定することができます.

使用例

```
type(dd_real)::x1,x2,x3,x4,x5,x6,x7,x8,x9,y
y=min(x1,x2)
```

```
y=min(x1,x2,x3,x4,x5,x6,x7,x8,x9)
```

2.8.11 aint

呼び出し形式

y=aint(x)

x	dd_real 型
y	dd_real 型

機能

小数点部分の切捨てを行います.

|x|<1.0 の場合, 結果の値は 0.0 となります. それ以外の場合は, x の絶対値以下で最大の整数値に x の符号を付けた値となります.

使用例

```
type(dd_real)::x,y
x=5.678d0
y=aint(x)      ! y の値は 5.0 になります.
x=-5.678d0
y=aint(x)      ! y の値は-5.0 になります.
```

2.8.12 anint

呼び出し形式

y=anint(x)

x	dd_real 型
y	dd_real 型

機能

小数点部分の四捨五入を行います.

x が 0.0 または正の場合, 結果の値は aint(x+0.5)となります. x が負の場合は, aint(x-0.5)となります.

使用例

```
type(dd_real)::x,y
x=5.678d0
y=anint(x)      ! y の値は 6.0 になります.
```

2.8.13 aimag

呼び出し形式

y=aimag(x)

x	dd_complex 型
y	dd_real 型

機能

複素数の虚部を求めます.

使用例

```
type(dd_complex)::z
type(dd_real)::y
y=aimag(z)
```

2.8.14 conjg

呼び出し形式

y=conjg(x)

x	dd_complex 型
y	dd_complex 型

機能

共役な複素数を求めます.

使用例

```
type(dd_complex)::y,z
complex(8)::c
z=conjg(y)
```

2.9 数学関数

2.9.1 sqrt

呼び出し形式

y=sqrt(x)

x	dd_real 型
y	dd_real 型

機能

平方根を計算します.

x>=0 でなければなりません.

使用例

```
type(dd_real)::x,y
y=sqrt(x)
```

2.9.2 exp

呼び出し形式

y=exp(x)

x	dd_real 型
y	dd_real 型

機能

指数関数を計算します.

x<709.0 でなければなりません.

使用例

```
type(dd_real)::x,y
y=exp(x)
```

2.9.3 log

呼び出し形式

y=log(x)

x		dd_real 型
y		dd_real 型

機能

対数関数を計算します.

x>0 でなければなりません.

使用例

```
type(dd_real)::x,y
y=log(x)
```

2.9.4 sin

呼び出し形式

y=sin(x)

x		dd_real 型
y		dd_real 型

機能

正弦関数を計算します.

|x|<1.4e19 でなければなりません.

使用例

```
type(dd_real)::x,y
y=sin(x)
```

2.9.5 cos

呼び出し形式

y=cos(x)

x		dd_real 型
y		dd_real 型

機能

余弦関数を計算します.

|x|<1.4e19 でなければなりません.

使用例

```
type(dd_real)::x,y
y=cos(x)
```

2.9.6 sincos

呼び出し形式

call sincos(x,s,c)

x	入力	dd_real 型
s	出力	dd_real 型
c	出力	dd_real 型

機能

正弦関数および余弦関数を計算します. s に sin(x), c に cos(x)の値を返します.

同じ x について正弦および余弦の値が必要な場合は, sin, cos を別々に呼び出すより, sincos を使用したほうが高速に計算できます.

|x|<1.4e19 でなければなりません.

使用例

```
type(dd_real)::x,s,c
call sincos(x,s,c)
```

2.10 マルチ・ベクトル演算ルーチン

マルチ演算ルーチンは, 1 回の呼び出しで 2 つの演算を行うことで, 演算器を効率よく使用して高速に計算を行います.

ベクトル演算ルーチンは, dd_real 型の 1 次元配列を渡すことにより, ルーチン内部でより効率的に計算を行います.

ベクトル演算ルーチンには, スレッド並列版も用意されています. スレッド並列版は, ルーチン内部で OpenMP による並列化を行い, 複数コアで並列に計算します. スレッド並列版ルーチンは, 他の fast_dd ルーチンと同様, スレッドセーフな作りになっており, OpenMP で並列化された内および外から呼び出すことができます.

2.10.1 m2_add_dd

呼び出し形式

call m2_add_dd(a,b,c,x,y,z)

a	入力	dd_real 型
b	入力	dd_real 型
c	出力	dd_real 型
x	入力	dd_real 型
y	入力	dd_real 型
z	出力	dd_real 型

機能

c=a+b, z=x+y を計算します.

使用例

```
type(dd_real)::a,b,c,x,y,z
call m2_add_dd(a,b,c,x,y,z)
```

2.10.2 m2_sum_dd

呼び出し形式

call m2_sum_dd(a,b,c)

a	入力	dd_real 型
b	入力	dd_real 型
c	入出力	dd_real 型

機能

c=c+a+b を計算します.

使用例

```
type(dd_real)::a,b,c
call m2_sum_dd(a,b,c)
```

2.10.3 m2_sub_dd

呼び出し形式

call m2_sub_dd(a,b,c,x,y,z)

a	入力	dd_real 型
b	入力	dd_real 型
c	出力	dd_real 型
x	入力	dd_real 型
y	入力	dd_real 型
z	出力	dd_real 型

機能

c=a-b, z=x-y を計算します.

使用例

```
type(dd_real)::a,b,c,x,y,z
call m2_sub_dd(a,b,c,x,y,z)
```

2.10.4 m2_mul_dd

呼び出し形式

call m2_mul_dd(a,b,c,x,y,z)

a	入力	dd_real 型または real(8)型
b	入力	dd_real 型
c	出力	dd_real 型
x	入力	dd_real 型または real(8)型
y	入力	dd_real 型
z	出力	dd_real 型

機能

c=a*b, z=x*y を計算します.

a と x は同じ型にします.

使用例

```
type(dd_real)::a,b,c,x,y,z
real(8)::p,q
call m2_mul_dd(a,b,c,x,y,z)
call m2_mul_dd(p,b,c,q,y,z)
```

2.10.5 v_add_dd

呼び出し形式

call v_add_dd(a,b,n,c)

a	入力	dd_real 型で大きさ n の 1 次元配列.
b	入力	dd_real 型で大きさ n の 1 次元配列.
n	入力	integer(4)型または integer(8)型. 配列 a,b,c の要素数.
c	出力	dd_real 型で大きさ n の 1 次元配列.

機能

1 次元配列 a, b, c について, c=a+b を計算します.

使用例

```
type(dd_real)::a(100),b(100),c(100)
call v_add_dd(a,b,100,c)
```

2.10.6 v_sub_dd

呼び出し形式

call v_sub_dd(a,b,n,c)

a	入力	dd_real 型で大きさ n の 1 次元配列.
b	入力	dd_real 型で大きさ n の 1 次元配列.
n	入力	integer(4)型または integer(8)型. 配列 a,b,c の要素数.
c	出力	dd_real 型で大きさ n の 1 次元配列.

機能

1 次元配列 a, b, c について, c=a-b を計算します.

使用例

```
type(dd_real)::a(100),b(100),c(100)
call v_sub_dd(a,b,100,c)
```

2.10.7 v_mul_dd

呼び出し形式

call v_mul_dd(a,b,n,c)

a	入力	dd_real 型で大きさ n の 1 次元配列.
b	入力	dd_real 型で大きさ n の 1 次元配列.
n	入力	integer(4)型または integer(8)型. 配列 a,b,c の要素数.
c	出力	dd_real 型で大きさ n の 1 次元配列.

機能

1 次元配列 a, b, c について, c=a*b を計算します.

使用例

```
type(dd_real)::a(100),b(100),c(100)
call v_mul_dd(a,b,100,c)
```

2.10.8 vm_add_dd

呼び出し形式

call vm_add_dd(a,b,n,c)

a	入力	dd_real 型で大きさ n の 1 次元配列.

b	入力	dd_real 型で大きさ n の 1 次元配列.
n	入力	integer(4)型または integer(8)型. 配列 a,b,c の要素数.
c	出力	dd_real 型で大きさ n の 1 次元配列.

機能

1 次元配列 a, b, c について, c=a+b を計算します. 本ルーチンの中でスレッドを生成し, スレッド並列に計算します.

使用例

```
type(dd_real)::a(100),b(100),c(100)
call vm_add_dd(a,b,100,c)
```

2.10.9 vm_sub_dd

呼び出し形式

call vm_sub_dd(a,b,n,c)

a	入力	dd_real 型で大きさ n の 1 次元配列.
b	入力	dd_real 型で大きさ n の 1 次元配列.
n	入力	integer(4)型または integer(8)型. 配列 a,b,c の要素数.
c	出力	dd_real 型で大きさ n の 1 次元配列.

機能

1 次元配列 a, b, c について, c=a-b を計算します. 本ルーチンの中でスレッドを生成し, スレッド並列に計算します.

使用例

```
type(dd_real)::a(100),b(100),c(100)
call vm_sub_dd(a,b,100,c)
```

2.10.10 vm_mul_dd

呼び出し形式

call vm_mul_dd(a,b,n,c)

a	入力	dd_real 型で大きさ n の 1 次元配列.
b	入力	dd_real 型で大きさ n の 1 次元配列.
n	入力	integer(4)型または integer(8)型. 配列 a,b,c の要素数.
c	出力	dd_real 型で大きさ n の 1 次元配列.

機能

1 次元配列 a, b, c について, c=a*b を計算します. 本ルーチンの中でスレッドを生成し, スレッド並列に計算します.

使用例

```
type(dd_real)::a(100),b(100),c(100)
call vm_mul_dd(a,b,100,c)
```

2.11 エラー発生時の動作について

fast_dd の一部のルーチンでは実行中に引数の値の妥当性をチェックしています. ルーチンとチェック内容については“4. **エラーメッセージ**”を参照してください.

これらのルーチンでエラーが発生した場合, 計算結果に NaN を設定して計算を続行します. この動作は以下のルーチンで変更することができます.

呼び出し形式

call fast_dd_errlvl(n)

n	入力	integer 型. このルーチンを呼び出した以降に fast_dd のルーチンでエラーが発生した場合の振る舞いを指定します. 0 : エラーメッセージを出力せず, 処理を続行する. 10 : エラーメッセージを出力し, 処理を続行する. 90 : エラーメッセージを出力し, プログラムを終了する. これら以外の値を指定したときは, 変更しません.

機能

エラーが発生した場合の動作を変更します.

3. C++版の利用方法

fast_dd を C++プログラムから利用する方法について説明します. 最初に簡単なプログラム例を示し, その後, 使用できる各機能について説明します.

3.1 プログラム例

fast_dd を使用した C++プログラムの例を示します.

以下のプログラムは, double-double 形式を使ってテーラー展開で e (自然対数の底)の値を計算します.

```
//
//   Include header files
//
#include <iostream>
#include <iomanip>
#include <math.h>
#include <fast_dd.h>                          // (1)

dd_real dd_exp_sample(const dd_real &x);

//
//   MAIN PROGRAM
//
int main()
{
   dd_real x,y;                               // (2)
   long double xx,yy,wy;
//
//   call dd_exp_sample
//
   x=1.0;
   y=dd_exp_sample(x);
//
//   call expl
//
   xx=to_long_double(x);
   yy=expl(xx);
   wy=to_long_double(y);
   std::cout <<"dd_exp_sample(1.0) :"
           <<std::setprecision(20)
           <<std::setw(25)<<wy<<std::endl;   // (4)
   std::cout <<"expl(1.0)          :"
           <<std::setprecision(20)
           <<std::setw(25)<<yy<<std::endl;
   return 0;
}

//
//   Function dd_exp_sample
//
```

```
dd_real dd_exp_sample(const dd_real &x)      // (3)
{
    dd_real wx,wy,wy0,wc;                    // (2)
    double c;
//
// exp(x)=1 + x + x**2/2! + x**3/3! + x**4/4! + ...
//
    wx=x;
    wy0=1.0;
    c=2.0;
    wc=1.0;
    wy=1.0+wx*wc;       //  wy=1+x
    while(abs(wy0-wy)>=1e-20) {
        wy0=wy;
        wc=wc/c;
        wx=wx*x;
        wy=wy+wx*wc;    //  wy=wy+x**i/i!
        c=c+1.0;
    }
    return wy;
}
```

解説

(1) fast_dd を使用するプログラムではヘッダファイル fast_dd.h を include します.

(2) double-double 形式の変数は dd_real を指定して宣言します.

(3) dd_real 型の変数を, サブルーチンや関数の引数や復帰値に指定できます.

(4) fast_dd では入出力をサポートしていないため, 値の出力はC++の long double型等を利用してください.

3.2 機能一覧

fast_dd C++版で提供する機能を表 3.2.1～表 3.2.4に示します.

表 3.2.1 代入・演算

演算子	機能
=	代入
+, -, *, /	四則演算
-	単項マイナス
+=, -=, *=, /=	代入演算
== , != , < , > , <= , >=	等値演算, 関係演算

表 3.2.2 数値関数

関数名	機能
dd_real	dd_real 型への型変換
to_int	dd_real 型を int 型へ変換

関数名	機能
to_double	dd_real 型を double 型へ変換
to_long_double	dd_real 型を long double 型へ変換
fabs, abs	絶対値
max	最大値
min	最小値
aint	整数値へ切り捨て
nint	小数部分を四捨五入

表 3.2.3 数学関数

関数名	機能
sqrt	平方根
exp	指数関数
log	対数関数
sin	正弦関数
cos	余弦関数
sincos	正弦関数と余弦関数

表 3.2.4 マルチ演算，ベクトル演算ルーチン

ルーチン名	機能
m2_add_dd	2 つの加算
m2_sum_dd	合計
m2_sub_dd	2 つの減算
m2_mul_dd	2 つの乗算
v_add_dd	ベクトルの加算
v_sub_dd	ベクトルの減算
v_mul_dd	ベクトルの乗算
vm_add_dd	ベクトルの加算(スレッド並列)
vm_sub_dd	ベクトルの減算(スレッド並列)
vm_mul_dd	ベクトルの乗算(スレッド並列)

3.3 ヘッダファイル

fast_dd を C++プログラムから利用する場合, ソースプログラムの中でヘッダファイル fast_dd.h をインクルードしてください.

例：
```
#include <fast_dd.h>
```

3.4 変数の宣言

fast_dd の C++版では, double-double 形式の変数の型として, dd_real 型が提供されています. dd_real 型は C++の構造体であり, 変数は次の例のように宣言します.

例 1：dd_real 型の変数 a と配列 x を宣言する.
```
dd_real a;
dd_real x[100];
```

次のように初期値を設定することができます.

例 2：dd_real 型の変数や配列を宣言し, 初期値を設定する.
```
dd_real a(1.0,1e-18);    // 変数の上位, 下位を指定する
dd_real b=1;             // 整数値を設定する
dd_real c=1.0;           // double 型の値を設定する
dd_real d=0.1L;          // long double 型の値を設定する
dd_real x[2]={dd_real(1.0), dd_real(2.0)};  // 配列の初期値の設定
```

3.5 代入

dd_real 型を使った代入を行うことができます. さらに, double 型との間で代入を行うことができます. サポートしている機能を表 3.5.1に示します.

double 型以外の変数や値を dd_real 型の変数に代入する場合は, いったん double 型に変換してから代入しています. このため long double 型などを代入する場合は, 後の例に示すように, 明示的な型変換を行ってください.

表 3.5.1 代入

	左辺の型	右辺の型
代入(=)	dd_real	dd_real
	dd_real	double

例 1：代入
```
dd_real a,b;
a=b;
```

例 2：int 型や double 型の変数や値を dd_real 型の変数に代入
```
dd_real a;
int n;
double d;
a=n;              // a=(double)n;と同じになります
a=10;
a=d;
```

```
a=1.0e-5;          // double 型の代入は精度に注意.
```

例 3：long double 型の変数や値を dd_real 型に代入する場合は, 精度が落ちないように明示的な型変換を行ってください.

```
dd_real a;
long double q;
a=dd_real(q);
```

例 4：左辺が dd_real 型でない場合は明示的な型変換が必要です

```
dd_real a;
int n;
double d;
long double q;
d=to_double(a);
n=to_int(a);
q=to_long_double(a);
```

3.6 演算

演算として四則演算 +, -, *, / が提供されています. 単項マイナス - および代入演算子 +=, -=, *=, /=を提供します.

サポートしている演算を表 3.6.1, 表 3.6.2に示します.

表 3.6.1 演算

演算	左辺の型	右辺の型	結果の型
加算, 減算, 乗算, 除算	dd_real	dd_real	dd_real
	double	dd_real	dd_real
	dd_real	double	dd_real

表 3.6.2 代入演算

演算	右辺の型	結果の型
加算(+=), 減算(-=), 乗算(*=), 除算(/=)	dd_real	dd_real
	double	dd_real

dd_real 型以外の型を演算に指定した場合, その型の変数を double 型に変換して計算します.

例 1：dd_real 型の演算

```
dd_real a,b,c,d,e;
c=a+b;
c=a-b;
c=a*b;
c=a/b;
c=a*b+d/e;
c+=a;
c-=a;
c*=a;
```

```
c/=a;
c=-a;      // 単項マイナス
```

例 2：double 型や int 型を含む演算の例

```
dd_real a,b,c;
double d;
c=d+a;
c=a-123.0;
c=2.0*b;
c=d/b;
```

例 3：long double 型を含む演算の例

```
dd_real x,y;
long double q;
y=x+dd_real(q);  // q を dd_real 型に変換してから計算してください
```

3.7 関係演算, 等値演算

dd_real 型を使った比較を行うことができます.

表 3.7.1にサポートする関係演算子, 等値演算子について示します.

表 3.7.1 関係演算, 等値演算

演算子	左辺	右辺	結果
関係演算子 (> , >= , < , <=)	dd_real 型	dd_real 型	bool 型
等値演算子 (==, !=)	double 型	dd_real 型	bool 型
	dd_real 型	double 型	bool 型

例 1：関係演算

```
dd_real a,b,c;
if(a>b) {
    c=0.0;
}
if(a>=b && (c!=0.0 || c!=1.0)) {
    c=0.0;
}
```

3.8 数値関数

3.8.1 dd_real

呼び出し形式

y=dd_real(x)

x　　dd_real 型, int 型, doulbe 型, long double 型

y　　dd_real 型

機能

x の値を持つ dd_real 型の変数を返します.

dd_real はコンストラクタとしてサポートしている機能です. ここではこれを明示的な型変換として使います.

使用例

```
dd_real y;
dd_real z;
y=dd_real(1.0);
y=dd_real(123);
y=dd_real(1.23L);
y=dd_real(z);
```

3.8.2 to_int

呼び出し形式

n=to_int(x)

x　　dd_real 型

n　　int 型

機能

dd_real 型の値を整数に変換します.

|x|<1.0 の場合, 結果の値は 0 となります.それ以外の場合は, x の絶対値以下で最大の整数値に x の符号を付けた値となります.

使用例

```
dd_real x;
int n;
n=to_int(x)+10;
```

3.8.3 to_double

呼び出し形式

y=to_double(x)

x　　dd_real 型

y　　double 型

機能

dd_real 型の x を, doulbe 型に変換します.

使用例

```
dd_real x;
double d;
d=to_double(x)+1.5;
```

3.8.4 to_long_double

呼び出し形式

y=to_long_double(x)

x　　dd_real 型

y　　long double 型

機能

dd_real 型の x を, long duble 型に変換します.

使用例

```
dd_real x;
long double d;
d=to_long_double(x);
```

3.8.5 fabs, abs

呼び出し形式

y=fabs(x)
y=abs(x)

x	dd_real 型
y	dd_real 型

機能

絶対値を返します.

使用例

```
dd_real x,y;
y=fabs(x);
y=abs(x);
```

3.8.6 max

呼び出し形式

#include <algorithm>

z=std::max(x,y)

x	dd_real 型
y	dd_real 型
z	dd_real 型

機能

x と y の最大値を求めます.

この機能は標準ライブラリの補助関数を使っています.

使用例

```
#include <algorithm>
dd_real x,y,z;
z=std::max(x,y);
```

3.8.7 min

呼び出し形式

#include <algorithm>

z=std::min(x,y)

x	dd_real 型
y	dd_real 型
z	dd_real 型

機能

x と y の最小値を求めます.

この機能は標準ライブラリの補助関数を使っています.

使用例

```
#include <algorithm>
dd_real x,y,z;
z=std::min(x,y);
```

3.8.8 aint

呼び出し形式

y=aint(x)

x　　dd_real 型

y　　dd_real 型

機能

小数点部分の切捨てを行います.

|x|<1.0 の場合, 結果の値は 0.0 となります. それ以外の場合は, x の絶対値以下で最大の整数値に x の符号を付けた値となります.

使用例

```
dd_real x,y;
x=5.678;
y=aint(x);      // y の値は 5.0 になります.
x=-5.678;
y=aint(x);      // y の値は-5.0 になります.
```

3.8.9 nint

呼び出し形式

y=nint(x)

x　　dd_real 型

y　　dd_real 型

機能

小数点部分の四捨五入を行います.

x が 0.0 または正の場合, 結果の値は aint(x+0.5)となります. x が負の場合は, aint(x-0.5)となります.

使用例

```
dd_real x,y;
x=5.678;
y=nint(x);      // y の値は 6.0 になります.
```

3.9 数学関数

3.9.1 sqrt

呼び出し形式

y=sqrt(x)

x　　dd_real 型

y　　dd_real 型

機能

平方根を計算します.

x>=0 でなければなりません.

使用例

```
dd_real x,y;
y=sqrt(x);
```

3.9.2 exp

呼び出し形式

y=exp(x)

x	dd_real 型
y	dd_real 型

機能

指数関数を計算します.

x<709.0 でなければなりません.

使用例

```
dd_real x,y;
y=exp(x);
```

3.9.3 log

呼び出し形式

y=log(x)

x	dd_real 型
y	dd_real 型

機能

対数関数を計算します.

x>0 でなければなりません.

使用例

```
dd_real x,y;
y=log(x);
```

3.9.4 sin

呼び出し形式

y=sin(x)

x	dd_real 型
y	dd_real 型

機能

正弦関数を計算します.

|x|<1.4e19 でなければなりません.

使用例

```
dd_real x,y;
y=sin(x);
```

3.9.5 cos

呼び出し形式

y=cos(x)

x		dd_real 型
y		dd_real 型

機能

余弦関数を計算します.

|x|<1.4e19 でなければなりません.

使用例

```
dd_real x,y;
y=cos(x);
```

3.9.6 sincos

呼び出し形式

sincos(x,s,c)

x	入力	dd_real 型
s	出力	dd_real 型
c	出力	dd_real 型

機能

正弦関数および余弦関数を計算します. s に sin(x), c に cos(x)の値を返します.

同じ x について正弦および余弦の値が必要な場合は, sin, cos を別々に呼び出すより, sincos を使用したほうが高速に計算できます.

|x|<1.4e19 でなければなりません.

使用例

```
dd_real x,s,c;
sincos(x,s,c);
```

3.10 マルチ・ベクトル演算ルーチン

マルチ演算ルーチンは, 1 回の呼び出しで 2 つの演算を行うことで, 演算器を効率よく使用して高速に計算を行います.

ベクトル演算ルーチンは, dd_real 型の 1 次元配列を渡すことにより, ルーチン内部でより効率的に計算を行います.

ベクトル演算ルーチンには, スレッド並列版も用意されています. スレッド並列版は, ルーチン内部で OpenMP による並列化を行い, 複数コアで並列に計算します. スレッド並列版ルーチンは, 他の fast_dd ルーチンと同様, スレッドセーフな作りになっており, OpenMP で並列化された内および外から呼び出すことができます.

3.10.1 m2_add_dd

呼び出し形式

m2_add_dd(a,b,c,x,y,z)

a	入力	dd_real 型
b	入力	dd_real 型
c	出力	dd_real 型
x	入力	dd_real 型
y	入力	dd_real 型
z	出力	dd_real 型

機能

c=a+b, z=x+y を計算します.

使用例

```
dd_real a,b,c,x,y,z;
m2_add_dd(a,b,c,x,y,z);
```

3.10.2 m2_sum_dd

呼び出し形式

m2_sum_dd(a,b,c)

a	入力	dd_real 型
b	入力	dd_real 型
c	入出力	dd_real 型

機能

c=c+a+b を計算します.

使用例

```
dd_real a,b,c;
m2_sum_dd(a,b,c);
```

3.10.3 m2_sub_dd

呼び出し形式

m2_sub_dd(a,b,c,x,y,z)

a	入力	dd_real 型
b	入力	dd_real 型
c	出力	dd_real 型
x	入力	dd_real 型
y	入力	dd_real 型
z	出力	dd_real 型

機能

c=a-b, z=x-y を計算します.

使用例

```
dd_real a,b,c,x,y,z;
m2_sub_dd(a,b,c,x,y,z);
```

3.10.4 m2_mul_dd

呼び出し形式

m2_mul_dd(a,b,c,x,y,z)

a	入力	dd_real 型または double 型
b	入力	dd_real 型
c	出力	dd_real 型
x	入力	dd_real 型または double 型
y	入力	dd_real 型
z	出力	dd_real 型

機能

c=a*b, z=x*y を計算します.

a と x は同じ型にします.

使用例

```
dd_real a,b,c,x,y,z;
double p,q;
m2_mul_dd(a,b,c,x,y,z);
m2_mul_dd(p,b,c,q,y,z);
```

3.10.5 v_add_dd

呼び出し形式

v_add_dd(a,b,n,c)

a	入力	dd_real 型で大きさ n の 1 次元配列.
b	入力	dd_real 型で大きさ n の 1 次元配列.
n	入力	int 型または long int 型. 配列 a,b,c の要素数.
c	出力	dd_real 型で大きさ n の 1 次元配列.

機能

1 次元配列 a, b, c について, c=a+b を計算します.

使用例

```
dd_real a[100],b[100],c[100];
v_add_dd(a,b,100,c);
```

3.10.6 v_sub_dd

呼び出し形式

v_sub_dd(a,b,n,c)

a	入力	dd_real 型で大きさ n の 1 次元配列.
b	入力	dd_real 型で大きさ n の 1 次元配列.
n	入力	int 型または long int 型. 配列 a,b,c の要素数.
c	出力	dd_real 型で大きさ n の 1 次元配列.

機能

1 次元配列 a, b, c について, c=a-b を計算します.

使用例

```
dd_real a[100],b[100],c[100];
v_sub_dd(a,b,100,c);
```

3.10.7 v_mul_dd

呼び出し形式

v_mul_dd(a,b,n,c)

a	入力	dd_real 型で大きさ n の 1 次元配列.
b	入力	dd_real 型で大きさ n の 1 次元配列.
n	入力	int 型または long int 型. 配列 a,b,c の要素数.
c	出力	dd_real 型で大きさ n の 1 次元配列.

機能

1 次元配列 a, b, c について, c=a*b を計算します.

使用例

```
dd_real a[100],b[100],c[100];
v_mul_dd(a,b,100,c);
```

3.10.8 vm_add_dd

呼び出し形式

vm_add_dd(a,b,n,c)

a	入力	dd_real 型で大きさ n の 1 次元配列.
b	入力	dd_real 型で大きさ n の 1 次元配列.
n	入力	int 型または long int 型. 配列 a,b,c の要素数.
c	出力	dd_real 型で大きさ n の 1 次元配列.

機能

1 次元配列 a, b, c について, c=a+b を計算します. 本ルーチンの中でスレッドを生成し, スレッド並列に計算します.

使用例

```
dd_real a[100],b[100],c[100];
vm_add_dd(a,b,100,c);
```

3.10.9 vm_sub_dd

呼び出し形式

vm_sub_dd(a,b,n,c)

a	入力	dd_real 型で大きさ n の 1 次元配列.
b	入力	dd_real 型で大きさ n の 1 次元配列.
n	入力	int 型または long int 型. 配列 a,b,c の要素数.
c	出力	dd_real 型で大きさ n の 1 次元配列.

機能

1 次元配列 a, b, c について, c=a-b を計算します. 本ルーチンの中でスレッドを生成し, スレッド並列に計算します.

使用例

```
dd_real a[100],b[100],c[100];
vm_sub_dd(a,b,100,c);
```

3.10.10 vm_mul_dd

呼び出し形式

vm_mul_dd(a,b,n,c)

a	入力	dd_real 型で大きさ n の 1 次元配列.
b	入力	dd_real 型で大きさ n の 1 次元配列.
n	入力	int 型または long int 型. 配列 a,b,c の要素数.
c	出力	dd_real 型で大きさ n の 1 次元配列.

機能

1 次元配列 a, b, c について, c=a*b を計算します. 本ルーチンの中でスレッドを生成し, スレッド並列に計算します.

使用例

```
dd_real a[100],b[100],c[100];
vm_mul_dd(a,b,100,c);
```

3.11 エラー発生時の動作について

fast_dd の一部のルーチンでは実行中に引数の値の妥当性をチェックしています. ルーチンとチェック内容については“4. **エラーメッセージ**”を参照してください.

これらのルーチンでエラーが発生した場合, 計算結果に NaN を設定して計算を続行します. この動作は以下のルーチンで変更することができます.

呼び出し形式

fast_dd_errlvl(n)

n	入力	int 型. このルーチンを呼び出した以降に fast_dd のルーチンでエラーが発生した場合の動作を指定します. 0 : エラーメッセージを出力せず, 処理を続行する. 10 : エラーメッセージを出力し, 処理を続行する. 90 : エラーメッセージを出力し, プログラムを終了する. これら以外の値を指定したときは, 変更しません.

機能

エラーが発生した場合の動作を変更します.

4. エラーメッセージ

本章では, fast_dd のルーチンが出力するエラーメッセージについて説明します.

エラーメッセージを出力するかしないかは fast_dd_errlvl ルーチンで制御できます.

fast_dd-error : 4001 : In exp(x), x>=709.0 : x= *value*

- **メッセージの説明**
  exp で引数の x の値が x>=709.0 でした.
- **値の説明**
  value : 引数に指定した値を表示します
- **利用者の処置**
  引数 x の値が x<709.0 になるようにしてください.

fast_dd-error : 4101 : In log(x), x<=0.0 : x= *value*

- **メッセージの説明**
  log で引数の x の値が x<=0.0 でした.
- **値の説明**
  value : 引数に指定した値を表示します
- **利用者の処置**
  引数 x の値が x>=0.0 になるようにしてください.

fast_dd-error : 4201 : In sin(x), abs(x)>=1.4e19 : x= *value*

- **メッセージの説明**
  sin で引数の x の値が|x|>= 1.4e19 でした.
- **値の説明**
  value : 引数に指定した値を表示します
- **利用者の処置**
  引数 x の値が|x|<1.4e19 になるようにしてください.

fast_dd-error : 4301 : In cos(x), abs(x)>=1.4e19 : x= *value*

- **メッセージの説明**
  cos で引数の x の値が|x|>= 1.4e19 でした.
- **値の説明**
  value : 引数に指定した値を表示します
- **利用者の処置**
  引数 x の値が|x|<1.4e19 になるようにしてください.

fast_dd-error : 4401 : In sincos(x,s,c), abs(x)>=1.4e19 : x= *value*

- **メッセージの説明**
  sincos で引数の x の値が|x|>= 1.4e19 でした.
- **値の説明**
  value : 引数に指定した値を表示します
- **利用者の処置**
  引数 x の値が|x|<1.4e19 になるようにしてください.

fast_dd-error : 4501 : In sqrt(x), x<0.0 : x= *value*

- **メッセージの説明**
  sqrt で引数の x の値が x<0.0 でした.
- **値の説明**
  value : 引数に指定した値を表示します
- **利用者の処置**
  引数 x の値が x>=0.0 になるようにしてください.

索引