JHPCN-DF Library

User's Guide

Ver. 1.4

Advanced Visualization Research Team
Advanced Institute for Computational Science
RIKEN

7-1-26, Minatojima-minami-machi, Chuo-ku, Kobe, Hyogo, 650-0047, Japan

http://aics.riken.jp/

February 2015

Revision history

rev1.0	2015/2/10	初版
rev1.1	2015/3/20	動作確認済環境の追加
rev1.2	2015/6/19	ファイル出力バッファサイズの変更機能など追加
rev1.3	2015/6/19	API 変更、LZ4 対応など追記
rev1.4	2015/8/19	ビルドオプションの変更など対応

目次

第 1 章

JHPCN-DF ライブラリの概要

本章では JHPCN-DF(Jointed Hierarchical Precision Compression Number Data Format) [1] ライブラリの概要と本文書について記述します。

1.1 JHPCN-DF とは

JHPCN-DF は、浮動小数点数に特化した不可逆圧縮アルゴリズムです。

本手法では、浮動小数点数の仮数部のうち、下位 bit 側のデータを切り捨て (0 埋め) することで同一 byte パターンの出現頻度を高め、deflate などの圧縮アルゴリズムが有効に作用するようにします。

また、切り捨てられる下位 bit 側のデータを別途保存することで、後から合成して元のデータを復元することも可能です。下位 bit 側のデータも符号部、指数部や仮数部の上位側が 0 で埋められているため、元のデータと比べると、より deflate による圧縮が効きやすくなっています。

本書および本ライブラリ内では、下位 bit 部分の 0 埋め処理および下位 bit データの抽出処理をエンコードと称し、逆に上位 bit のみのデータと下位 bit のみのデータから元のデータを復元する処理をデコードと称します。

1.2 JHPCN-DF ライブラリ

本ライブラリは JHPCN-DF にもとづいた、浮動小数点数データの圧縮/伸張機能を提供する C++ 用ライブラリです。C および Fortran から各ルーチンを呼び出すためのインタフェースルーチンも含まれており、C/C++/Fortran いずれの言語からでもデータのエンコード/デコードおよび圧縮済データの読み書きを行うことができます。

エンコード後のデータ圧縮および圧縮済データの伸張処理は、デフォルトでは zlib を用いて gzip 形式にて行なっています。したがって、圧縮後のファイルは gunzip コマンドなどの gzip 形式に対応した一般的なアーカイバソフトにて解凍することができます。また、ビルド時に LZ4 対応を有効にしていれば、引数を切り替えるだけで LZ4 形式による圧縮/伸長を用いることもできます。

1.3 動作確認済環境

本ライブラリは表 1.1 に記載の環境で動作を確認しています。

表 1.1　動作確認済環境

アーキテクチャ	OS	コンパイラ	ビルドツール
intel 64	RedHat Enterprise Linux 6.4 64bit	GNU/C++ ver 4.8.2 and 4.4.7	cmake 2.8.12 autoconf 2.63
		Intel Compiler 15.0.1	cmake 2.8.12 autoconf 2.63
	Windows 7 SP1(64bit) + cygwin	GNU/C++ ver 4.9.3	cmake 3.2.3 autoconf 2.69
	Windows 7 SP1(64bit)	Visual Studio 2015 Community edition	cmake 3.1.2
	MacOS 10.10(Yosemite)	GNU/C++ ver 4.9.2	cmake 3.2.3 autoconf 2.69
		clang 6.1.0	cmake 3.2.3 autoconf 2.69
SPARC64 IXfx		富士通コンパイラ	cmake 2.6

1.4　この文書について

1.4.1　文書中で使われる書式

以下の書式はシェルからのコマンド入力を表わします。前者は一般ユーザ権限でのコマンド入力、後者は root 権限でのコマンド入力を表します。

```
$ コマンド（コマンド引数）
# コマンド（コマンド引数）
```

第 2 章

パッケージのビルド

本章では、JHPCN-DF ライブラリのビルド方法について記述します。

2.1 JHPCN-DF ライブラリが依存するライブラリ

JHPCN-DF ライブラリは、dflate による圧縮に zlib を使用しているため、ライブラリをビルドする前に zlib のインストールを行なう必要があります。また、LZ4 による圧縮を用いる場合は、LZ4 ライブラリをインストールする必要があります。

2.1.1 zlib のインストール

Linux, windows(cygwin 環境), MacOS でのインストール

これらの環境では、各パッケージマネージャにてコンパイル済のライブラリが配布されていますので、そちらを用いてインストールしてください。

FX10(クロスコンパイル) 環境でのインストール

FX10 用のクロスコンパイル環境では、zlib は”/opt/FJSVXosDevkit/sparc64fx/target/lib64”以下にインストールされています。見つからない場合は、管理者までお問い合わせください。

Windows(native) 環境でのインストール

zlib の web ページ (http://www.zlib.net/) から ”zlib compiled DLL”をダウンロードして展開してください。zip アーカイブに同梱されているファイルのうち、”zlib1.dll”, ”lib/zdll.lib”, ”include/zlib.h”, ”include/zconf.h”を適切なフォルダにコピーしてください。なお、同ページにて配布されている zlib ライブラリは 32bit 用にビルドされているので、64bit アプリケーションから JHPCN-DF を使う場合は、ソースコードから zlib をビルドする必要があります。

ソースからのインストール

zlib の web ページで配布されているソースコードを展開し、以下のコマンドを入力してください。

Unix 系 OS および sygwin の場合

```
$ mkdir BUILD
$ cd BUILD
$ cmake ..
# make install
```

Windows の場合

```
$ mkdir BUILD
$ cd BUILD
$ cmake .. -G "Visual Studio 14 2015 Win64"
$ msbuild /p:Configuration=Release zlib.sln
```

-G オプションで指定する文字列は、コンパイルに使う Visual Studio のバージョンに合わせて変更してください。

2.1.2　LZ4 のインストール

バイナリパッケージによるインストール

MacOS および Ubuntu

ソースからのインストール

lz4 の github ページ (https://github.com/Cyan4973/lz4) からソースコード一式を取得して、以下のコマンドを入力してください。

Unix 系 OS の場合

```
$ cd cmake_unofficial
$ cmake .
# make install
```

Windows の場合

```
$ cd cmake_unofficial
$ cmake . -G "Visual Studio 14 2015 Win64"
$ msbuild /p:Configuration=Release LZ4.sln
```

-G オプションで指定する文字列は、コンパイルに使う Visual Studio のバージョンに合わせて変更してください。

2.2　JHPCN-DF ライブラリのビルド

JHPCN-DF ライブラリ本体のビルドには Cmake 又は autotools を利用します。

2.2.1　cmake によるビルド

以下のコマンドを順に実行することで、ヘッダファイル、およびライブラリ本体がインストールされます。実行例中の PATH_TO_JHPCN-DF_DIR には CMakeLists.txt ファイルがあるディレクトリを指定してください。なお、ビルド用

に使われる一時ファイルなどは、全て最初に作成した”BUILD”ディレクトリ内に生成されます。ビルドオプションを変えて再ビルドを行なう場合は、別のディレクトリを作成して、そちらで cmake 以降のコマンドを実行することで複数のバージョンを互いに干渉させずにビルドすることができます。

```
$ mkdir BUILD
$ cd BUILD
$ cmake PATH_TO_JHPCN-DF_DIR
# make install
```

FX10 のクロス環境で cmake を実行する場合は以下のオプションを追加して cmake を実行してください。

```
cmake -DCMAKE_TOOLCHAIN_FILE=cmake/Toolchain_K.cmake
```

cmake コマンドの実行時に以下のように-D オプションを使用して、JHPCN-DF のコンパイル設定を変更することができます。

```
$ cmake -DCMAKE_CXX_COMPILER=icpc -Dbuild_examples=on
```

設定可能なオプションは、表 2.1 に示すとおりです。

表 2.1　JHPCN-DF のビルドオプション

オプション	内容 (デフォルト値)
CMAKE_CXX_COMPILER	使用する C++ コンパイラ
CMAKE_CXX_FLAGS	C++ コンパイラに渡すオプション
CMAKE_C_COMPILER	使用する C コンパイラ (サンプルコードのコンパイルのみに使用)
CMAKE_C_FLAGS	C コンパイラに渡すオプション
CMAKE_Fortran_COMPILER	使用する Fortran コンパイラ
CMAKE_Fortran_FLAGS	Fortran コンパイラに渡すオプション
CMAKE_INSTALL_PREFIX	ライブラリのインストール先 (/usr/local)
with_sse	SSE 命令を生成するオプションを指定する (on)
with_OpenMP	OpenMP による共有メモリ並列化を行う (on)
with_Fortran_interface	Fortran 用のインターフェースルーチンをビルドする (off)
use_lz4	LZ4 ライブラリを用いた圧縮機能を有効にする (off)
build_unit_tests	単体テストプログラムをコンパイルするかどうか指定するフラグ (off)
build_performance_test	性能テストプログラムをコンパイルするかどうか指定するフラグ (off)
REAL_8_BYTE	サンプルコードを double(8byte 実数) 用にビルドするかどうか指定するフラグ (off)

with_sse オプションが有効な場合、インテルコンパイラ用には-xHost、GNU コンパイラ用には-march=native といった、ビルドするマシンと同一アーキテクチャ向けの命令を生成するオプションが指定されます。ただし、MacOS 環境で GNU コンパイラを用いている場合、OS 標準の as コマンドが古く-march=native で生成されるアセンブラコードが

アセンブルできないため-msse3 を指定します。

他の環境で、SIMD 命令のレベルを明示的に指定する場合は、with_sse オプションを無効にした上で”cmake/CompileOptionSelector.cmake”を参考にコンパイルオプションを指定してください。

また、京コンピュータおよび FX 環境では、マシンがサポートする命令のレベルによる問題が生じる可能性は薄いと思われますので with_sse オプションの有効、無効に関らず-Kfast オプションが常に指定されるようになっています。

2.2.2　autotools によるビルド

GNU autotools を用いたビルドは、以下のコマンドを順次実行することで行ないます。configure 実行時に automake などの GNU autotools に起因するエラーが発生する場合は一度”autoreconf”を実行してから、”./configure”以降のコマンドを実行してください。

```
$ mkdir BUILD
$ cd BUILD
$ ../configure [options]
$ make
$ make install
```

2.2.3　サンプルコードのビルド

examples ディレクトリ以下には、C++, C, Fortran の各言語で書かれたサンプルコードが格納されています。これらのコードをビルドするには、cmake を用いて以下のコマンドを実行してください。

```
$ cmake PATH_TO_EXAMPLES_DIR -DCMAKE_PREFIX_PATH=PATH_TO_JHPCN-DF \
      -DCMAKE_MODULE_PATH=PATH_TO_JHPCN-DF/share
$ make all
```

コマンド実行例中の PATH_TO_EXAMPLES_DIR は examplss ディレクトリのパスを、PATH_TO_JHPCN-DF は JHPCNDF ライブラリをインストールしたディレクトリのパスをそれぞれ指定してください。

第 3 章

API

本章では JHPCN-DF ライブラリの API について記述します

3.1 API 構成

JHPCN-DF ライブラリのインタフェース関数は、FileIO 関数群と In-Memory 関数群の 2 つに分けることができます。

3.1.1 FileIO 関数群

FileIO 関数群は、C 言語の stdio や Fortran の Read/write 文と同じように、Open/Close というファイル操作を行う関数と実際にデータの読み書きを行う Read/Write 関数に分かれます。Read/Write 関数は、内部で JHPCN-DF によるエンコード/デコードおよび deflate による圧縮/伸張を行なうことで圧縮ファイルの IO を実現しています。

3.1.2 In-memory 関数群

In-memory 関数群は、メモリ上のデータに対してエンコードまたはデコードを行なって、引数で渡された別の領域にデータを書き込みます。どちらの処理においても、処理後のデータを格納するための領域は関数を呼び出す前にユーザプログラム側で確保する必要があります。

3.2 API 利用方法

3.2.1 ヘッダファイル

JHPCN-DF ライブラリを使用するために必要なヘッダファイルは”jhpcndf.h”のみです。C/C++ 言語のプログラムから利用する場合は、このファイルを include してください。

また、fortran プログラムから利用する場合は以下のように use 文を使用してください。

```
use jhpcndf
```

3.2.2 C++ 版インタフェース関数

C++ 版のインタフェース関数は全て namespace JHPCNDF 内で定義されているため、呼び出し時には JHPCNDF::を付けるか、using 指令を使って名前解決をしてください。

ファイルオープン関数

ファイルオープン関数のシグネチャは以下のとおりです。

```
int fopen(const std::string& filename_upper,
          const std::string& filename_lower = "",
          const char*        mode = "rb",
          const std::string& comp = "gzip",
          const size_t&      buff_size=32768);
```

本関数を呼び出すと、filename_upper および filename_lower の二つファイルを指定された mode で open し、これらの FILE 構造体へアクセスするための ID 番号を返します。

filename_lower として空文字列 (””) が指定された場合は、filename_lower に相当するファイルは open されず、その時に返された key を使って、read/write した時は上位 bit 側のデータのみを入出力します。

第 4 引数の comp では圧縮方法を指定することができます。使用可能な値は表 3.1 に記載の 3 種類です。

表 3.1　圧縮方法の一覧

gzip(デフォルト)	zlib を用いて gzip 形式で圧縮します。
lz4	lz4 ライブラリを用いて圧縮します。
none	圧縮しません

また、gzip/lz4 ともに gzip_3 のような形でアンダースコアで区切って数字を追加することで、ライブラリに渡すオプションを指定することができます。詳細は jhpcndf.h のコメントをご参照ください。

第 5 引数の buff_size は zlib などの圧縮ライブラリが用いるバッファのサイズで、単位は Byte です。

ファイルクローズ関数

ファイルクローズ関数のシグネチャは以下のとおりです。

```
void fclose(const int& key);
```

本関数を呼び出すと、引数で渡された key に対応するファイルポインタを close します。

ファイル出力関数

ファイル出力関数のシグネチャは以下のとおりです。

```
template <typename T>
size_t fwrite(const T*           ptr,
              size_t             size,
              size_t             nmemb,
              const int&         key,
              const float&       tolerance,
              const bool&        is_relative=true,
              const std::string& enc="binary_search",
              const bool&        time_measuring = false
              const bool&        byte_swap = false);
```

本関数を呼び出すと、ptr で指す領域に格納された nmemb 個のデータを圧縮して、key が指すファイルポインタに対して出力します。

第 2 引数の size にはデータ 1 要素のサイズをバイト単位で渡してください。引数で渡された size が sizeof(T) と違っていた場合、予期せぬエラーが発生する可能性があります。

第 5 引数の tolerance では許容誤差を相対値として指定します。圧縮の際にはここで指定した誤差を満たすように下位 bit 側を 0 埋めして切り捨てます。ただし、第 6 引数の is_relative が false の時はこの値は絶対値として扱われます。

第 7 引数 enc ではエンコーダのアルゴリズムを指定することができます。使用可能な値は表 3.2 に記載してあります。

表 3.2　エンコーダの一覧

normal	文献 [1] に記載の方法でエンコードを行なう
byte_aligned	上位 bit と下位 bit の分割位置を、仮数部と指数部の境界もしくは、下位側から n*8bit の位置に限定して許容誤差を満たす分割位置を探す
linear_search	仮数部の範囲内で上位 bit 側から順に線形探索によって許容誤差を満たす分割位置を探す
binary_search	仮数部の範囲内で、二分探索によって許容誤差を満たす分割位置を探す (デフォルト)
nbit_filter	指定された bit 位置で分割する
dummy	分割を行なわず、全てのデータを上位 bit 側に格納する

第 8 引数の time_measuring を true に設定すると、実行時に標準エラー出力に対して経過時間を出力します。

第 9 引数の byte_swap を true に設定するとファイル出力時にエンディアン変換を行ってから出力します。

本関数の戻り値は出力したファイルサイズです。圧縮してファイル出力を行なった場合でも、圧縮前のデータサイズを返しますので、帰ってきた値と実際のファイルサイズは異なる可能性があります。

本関数は、char, short, int, long およびこれらの unsignd 型と float, double 型に対してのみ明示的インスタンス化されています。float, double 用の関数は引数での指定にしたがって、JHPCN-DF によるエンコードと fopen 時に指定した形式での圧縮を行った上でファイル出力を行います。

他の型用の関数は、JHPCN-DF によるエンコードは行なわず、圧縮とファイル出力のみを行います。したがって第 7 引数で指定されたエンコーダはこの場合無視されます。また、fopen 時に下位 bit 用のファイルを開いていた場合は、上位 bit 用のファイルと下位 bit 用のファイルの両方に同じ内容を出力します。なお、これ以外の型 (ユーザ定義クラスなど) のデータ出力に本関数を用いる場合は char 型へキャストして呼び出してください。

ファイル入力関数

ファイル入力関数のシグネチャは以下のとおりです。

```
template <typename T>
size_t fread(T* ptr, size_t size, size_t nmemb,
     const int& key, const bool& byte_swap=false);
```

本関数を呼び出すと、key で指定されたファイルから nmemb 個のデータを読み込み、ptr が指す領域に格納します。

第 2 引数の size にはデータ 1 要素のサイズをバイト単位で渡してください。引数で渡された size が sizeof(T) と違っていても引数で渡された値を使用します。

第 5 引数の byte_swap を true に設定すると、ファイルを読み込んだ後でエンディアン変換を行います。

JHPCNDF::fopen で開いたファイルが gzip 形式で圧縮されていた場合は、自動的に展開してデータを読み取ります。なお、圧縮ファイルを読み込む場合、第三引数の nmemb には圧縮前のデータサイズを指定する必要があることにご留意ください。

本関数は、char, short, int, long およびこれらの unsignd 型と float, double 型に対してのみ明示的インスタンス化されています。JHPCNDF::fopen の実行時に下位 bit 用のファイルも開いていた時は、float, double 用の関数では 2 つのファイルを読み取ってビット毎の論理和をとり、元の精度のデータを復元した上で ptr が指す領域に格納します。それ以外の型用の関数や、float, double 用の関数で下位 bit 用のファイルを fopen 時に指定していなかった場合は、上位 bit 用のファイルのデータのみを読み取ります。

エンコード関数

エンコード関数のシグネチャは以下のとおりです。

```
template<typename T>
void encode(const size_t&       length,
            const T* const      src,
            T* const            dst,
            T* const            dst_lower,
            const float&        tolerance,
            const bool&         is_relative=true,
            const std::string&  enc = "binary_search",
            const bool&         time_measuring = false);
```

本関数を呼び出すと、src で指定された領域に保存されている length 個のデータを上位 bit 側のデータと下位 bit 側のデータに分け、それぞれ dst および dst_lower に格納します。

dst および dst_lower が指す領域はユーザプログラム側で領域を確保してから、関数を呼び出してください。なお、dst_lower として NULL が渡された場合は下位ビット側のデータは生成しません。

本関数は float および double に対してのみ明示的にインスタンス化されているため、その他の型用の関数を呼び出そうとすると、リンク時に undefined reference のエラーが発生します。

第 5 引数の tolerance 以降の引数は fwrite() と同じです。

デコード関数

デコード関数のシグネチャは以下のとおりです。

```
template<typename T>
void decode(const size_t&       length,
                  const T* const src_upper,
                  const T* const src_lower,
                        T* const dst);
```

本関数を呼び出すと、src_upper と src_lower に保存されている length 個のデータの bit 毎の論理和をとって、dst に格納します。

dst が指す領域はユーザプログラム側で領域を確保してから、関数を呼び出してください。

本関数は float および double に対してのみ明示的にインスタンス化されているため、その他の型用の関数を呼び出そうとすると、リンク時に undefined reference のエラーが発生します。

3.2.3　C 言語版インタフェース関数

ファイルオープン関数

ファイルオープン関数のシグニチャは以下のとおりです。

```
int JHPCNDF_fopen(const char* filename_upper, const char* filename_lower,
                  const char* mode, const char* comp, const size_t buff_size);
```

引数や戻り値などは、C++ 版の fopen()(3.2.2) と同様です。

ファイルクローズ関数

ファイルクローズ関数のシグニチャは以下のとおりです。

```
void JHPCNDF_fclose(const int key);
```

引数や戻り値などは、C++ 版の fclose()(3.2.2) と同様です。

ファイル出力関数

ファイル出力関数は、float 用、double 用、その他 (void) 用として 3 種類が用意されています。関数のシグニチャは以下のとおりです。

```
size_t JHPCNDF_fwrite_float(const float* ptr, size_t size, size_t nmemb, const int key,
                            const float tolerance, const int is_relative, const char* enc);

size_t JHPCNDF_fwrite_double(const double * ptr, size_t size, size_t nmemb, const int key,
                             const float tolerance, const int is_relative, const char* enc);

size_t JHPCNDF_fwrite(const void* ptr, size_t size, size_t nmemb, const int key,
                      const float tolerance, const char* enc);
```

引数や戻り値などは、C++ 版の fwrite()(3.2.2) と同様です。

ファイル入力関数

ファイル入力関数は、float 用、double 用、その他 (void) 用として 3 種類が用意されています。関数のシグニチャは以下のとおりです。

```
size_t JHPCNDF_fread_float(float* ptr, size_t size, size_t nmemb, const int key);

size_t JHPCNDF_fread_double(double* ptr, size_t size, size_t nmemb, const int key);

size_t JHPCNDF_fread(void* ptr, size_t size, size_t nmemb, const int key);
```

引数や戻り値などは、C++ 版の fread(3.2.2) と同様です。

エンコード関数

エンコード関数は、float 用、double 用の 2 種類が用意されています。関数のシグニチャは以下のとおりです。

```
void JHPCNDF_encode_float(const size_t length, const float* const src, float* const dst,
                          float* const dst_lower, const float tolerance,
```

```
                          const int is_relative, const char* enc);

void JHPCNDF_encode_double(const size_t length, const double* const src, double* const dst,
                           double* const dst_lower, const float tolerance,
                           const int is_relative, const char* enc);
```

引数や戻り値などは、C++ 版の encode()(3.2.2) と同様です。

デコード関数

デコード関数は float 用、double 用の 2 種類が用意されています。関数のシグニチャは以下のとおりです。

```
void JHPCNDF_decode_float(const size_t length, const float* const src_upper,
                          const float* const src_lower, float* const dst);

void JHPCNDF_decode_double(const size_t length, const double* const src_upper,
                           const double* const src_lower, double* const dst);
```

引数や戻り値などは、C++ 版の encode()(3.2.2) と同様です。

3.2.4　Fortran 版インタフェースサブルーチン

ファイルオープンサブルーチン

ファイルオープンサブルーチンは、C++ 版の fopen() に対するラッパールーチンです。

```
 subroutine jhpcndf_open(unit, ufile, lfile, action, comp, buff_size)
   integer(4)       :: unit
   character(len=*) :: ufile, lfile, action, comp
   integer(8)       :: buff_size
```

第 1 引数の unit には通常の open 文における装置番号と同様に出力先を指定するための ID 番号が入ります。ただし、open 文と違って番号は JHPCN-DF ライブラリが自動的に選択してこの引数に格納して返します。したがって、他の場所で open 文で開いた装置番号や、事前接続された装置番号と同じ番号が返ってくることもありますが、実際には別のファイルに対して読み書きされるため問題ありません。逆に、jhpcndf_open で取得した ID 番号に対して write 文や read 文を実行しても、ここで指定したファイルの読み書きはできませんので、ご留意ください。

第 2 引数の ufile および第 3 引数の lfile には、それぞれ上位 bit、下位 bit のデータを出力するファイル名を指定してください。lfile に空文字列が指定されていると、下位 bit 側のファイルは生成されません。

第 4 引数の action には出力時は”write”を入力時は”read”を指定してください。指定する文字列は大文字小文字どちらでも構いません。なお、実際には、”write”以外の文字列が指定されていた場合は”read”が指定されたものとして扱われます。

ファイルクローズサブルーチン

ファイルクローズサブルーチンは C++ 版の fclose() に対するラッパールーチンです。

```
 subroutine jhpcndf_close(unit)
       integer(4)       :: unit
```

引数の unit には、jhpcndf_open の第 1 引数として返ってきた ID 番号を指定してください。

ファイル出力サブルーチン

ファイル出力サブルーチンは C++ 版の fwrite() に対するラッパールーチンです。本サブルーチンは総称名で宣言されており、実際には real(4)、real(8)、integer(4)、integer(8)、character の出力に対応しています。

```
subroutine jhpcndf_write(unit, recl, data, tol, is_rel, enc)
  integer(4)        :: unit
  integer(8)        :: recl
  real(4)           :: tol
  logical           :: is_rel
  character(len=*)  :: enc
  real(4)           :: data(:)
```

第 1 引数の unit には、jhpcndf_open が返してきた ID 番号を指定してください。

第 2 引数の recl は出力するデータ長を指定してください。

第 3 引数の data には出力するデータが格納されている配列を指定してください。data の型は、実際には real(4)、real(8)、integer(4)、integer(8) および character が使えます。

第 4 引数の tol には許容誤差を指定してください。第 5 引数の is_rel が true の時はこの値は相対値として、false の時は絶対値として扱われます。

第 6 引数の enc には、エンコード方法を表 3.2 に記載されている中からひとつ指定してください。

ファイル入力サブルーチン

ファイル入力サブルーチンは C++ 版の fread() に対するラッパールーチンです。本サブルーチンは総称名で宣言されており、実際には real(4)、real(8)、integer(4)、integer(8)、character の入力に対応しています。

```
subroutine jhpcndf_read(unit, recl, data)
  integer(4)        :: unit
  integer(8)        :: recl
  real(4)           :: data(:)
```

第 1 引数の unit には、jhpcndf_open が返してきた ID 番号を指定してください。

第 2 引数の recl は読み込むデータ長を指定してください。

第 3 引数の data には読み取ったデータを格納する配列を指定してください。data の型は、実際には real(8)、integer(4)、integer(8) および character が使えます。

エンコードサブルーチン

エンコードサブルーチンは C++ 版の encode() に対するラッパールーチンです。本サブルーチンは総称名で宣言されており、実際には real(4)、real(8)、integer(4)、integer(8) のエンコードに対応しています。

```
subroutine jhpcndf_encode(length, src, dst, dst_lower, tol, is_rel, enc)
  integer(8)        :: length
  real(4)           :: src(:)
  real(4)           :: dst(:)
  real(4)           :: dst_lower(:)
  real(4)           :: tol
  logical           :: is_rel
  character(len=*)  :: enc
```

第 1 引数の length はエンコードするデータ長を指定してください。

第 2 引数の src にはエンコードするデータが格納されている配列を指定してください。

第 3 引数の dst には上位 bit 側のデータを格納する配列を指定してください。

第 4 引数の dst_lower には下位 bit 側のデータを格納する配列を指定してください。

src, dst, dst_lower には real(4) または real(8) が使えます。

第 5 引数の tol には許容誤差を指定してください。第 6 引数の is_rel が true の時はこの値は相対値として、false の時は絶対値として扱われます。

第 7 引数の enc には、エンコード方法を表 3.2 に記載されている中からひとつ指定してください。

デコードサブルーチン

デコードサブルーチンは C++ 版の decode() に対するラッパールーチンです。本サブルーチンは総称名で宣言されており、実際には real(4)、real(8)、integer(4)、integer(8) のデコードに対応しています。

```
subroutine jhpcndf_decode(length, src_upper, src_lower, dst)
  integer(8)        :: length
  real(4)           :: src_upper(:)
  real(4)           :: src_lower(:)
  real(4)           :: dst(:)
```

第 1 引数の length にはデコードするデータ長を指定してください。

第 2 引数の src_upper には上位 bit 側のデータが格納されている配列を指定してください。

第 3 引数の src_lower には下位 bit 側のデータが格納されている配列を指定してください。

第 4 引数の dst にはデコード後のデータを格納する配列を指定してください。

src_upper, src_lower, dst には real(4) または real(8) が使えます。

付録 A　メモリ使用量

本ライブラリでは、ライブラリ外とのデータのやりとりに必要なメモリ領域は、呼び出し側の責任で確保および解放をする設計としています。しかし、fwrite および fread 関数内部での処理のために一時領域が必要なため、それぞれ 3 および 4 に記載のサイズの領域を関数の内部で確保および解放しています。

また、gzip 形式での圧縮/伸張を指定した場合は、zlib が使用するための領域をさらにこれらの関数の内部で、確保および解放しています。zlib が使用する領域のサイズはデフォルトでは 32kByte ですが、JHPCNDF::fopen() の第４引数で Byte 単位で任意のサイズを指定することができます。

表 3　JHPCNDF::fwrite() 内部で使用する一時領域のサイズ

データ型	オプション指定	使用するメモリサイズ
float または double	下位 bit 出力指定無し	sizeof(T)*nmemb Byte
float または double	下位 bit 出力指定有り	2*sizeof(T)*nmemb Byte
float および double 以外	エンディアン変換オプション指定無し	0
float および double 以外	エンディアン変換オプション指定有り	sizeof(T)*nmemb Byte

表 4　JHPCNDF::fread() 内部で使用する一時領域のサイズ

データ型	オプション指定	使用するメモリサイズ
float または double	下位 bit 出力指定無し	sizeof(T)*nmemb Byte
float または double	下位 bit 出力指定有り	2*sizeof(T)*nmemb Byte
float および double 以外	エンディアン変換オプション指定無し	0
float および double 以外	エンディアン変換オプション指定有り	sizeof(T)*nmemb Byte

付録 B　許容誤差

JHPCNDF::encode および JHPCNDF::fwrite の呼び出し時に、許容誤差を 10 進数の浮動小数点数として指定しますが、本ライブラリのエンコード処理は、仮数部の数値を切り捨てているため、許容誤差の設定に対して切り捨てられる桁数は 1 に示すように対数に比例して増加し 0.5 に設定した時が最も低精度な値となります。

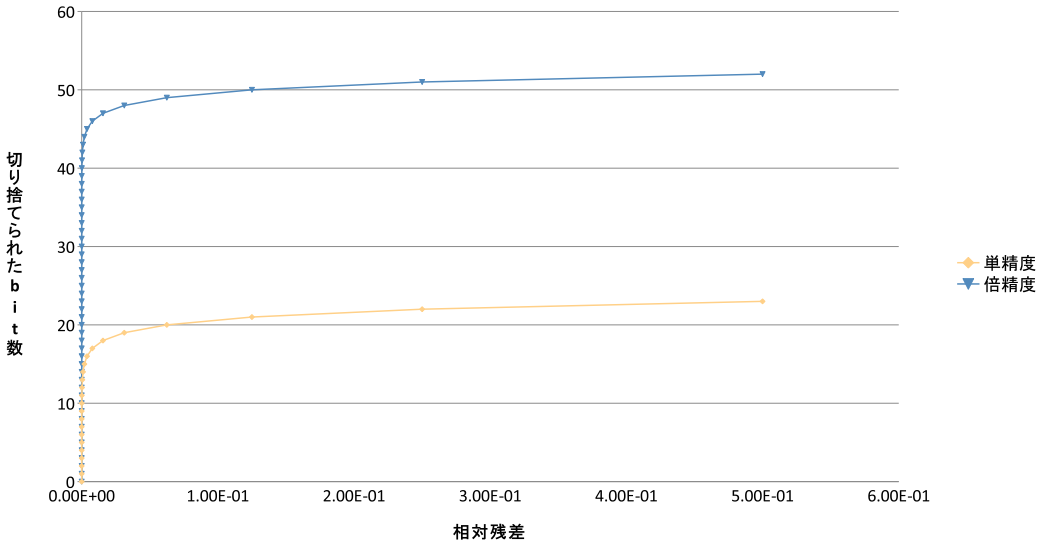

図 1　許容誤差 (相対値) の設定値と切り捨てられる bit 数の関係

参考文献

[1] K. Hagita, M. Omiya, T. Honda and M. Ogino. ”Efficient Data Compression by Efficient Use of HDF5 Format” SC14 poster session