

NAG C Library, Mark 26.1
CLMI6261DL - License Managed
Intel-based Mac, Mac OS X 64, Intel C/C++ or Apple clang, 64-bit integers

ユーザーノート

内容

1. イントロダクション	1
2. 追加情報	1
3. 一般情報	2
3.1. ライブラリのリンク方法	4
3.2. Example プログラム	8
3.3. データ型	9
3.4. メンテナンスレベル	9
4. ルーチン固有の情報	10
5. ドキュメント	14
6. サポート	15
7. コンタクト情報	15

1. イントロダクション

本ユーザーノートは、NAG C Library, Mark 26.1 – CLMI6261DL（ライブラリ）のご利用方法（リンク方法）を説明します。

本ユーザーノートには、NAG Library Manual, Mark 26（ライブラリマニュアル）には含まれない製品毎の情報が含まれています。ライブラリマニュアルに「ユーザーノート参照」などと書かれている場合は、本ユーザーノートをご参照ください。

ライブラリルーチンのご利用に際しては、ライブラリマニュアル（「5. ドキュメント」参照）の以下のドキュメントをお読みください。

- (a) How to Use the NAG Library and its Documentation
- (b) Chapter Introduction
- (c) Function Document

2. 追加情報

本ライブラリの動作環境やご利用方法についての最新の情報は、以下のウェブページをご確認ください。

<http://www.nag.co.uk/doc/inun/cl26/mi61dl/supplementary.html>

3. 一般情報

本製品では、Intel® Math Kernel Library (MKL) が提供する BLAS/LAPACK ルーチンを利用するスタティックライブラリ `libnagc_mkl.a` および共有ライブラリ `libnagc_mkl.dylib` と、NAG が提供する BLAS/LAPACK ルーチンを利用するスタティックライブラリ `libnagc_nag.a` および共有ライブラリ `libnagc_nag.dylib` を提供します。本ライブラリは、MKL version 2017.0.3 を用いてテストされています。MKL version 2017.0.3 は本製品の一部として提供されます。MKL の詳細については Intel 社のウェブサイト <https://software.intel.com/intel-mkl> をご参照ください。パフォーマンスの面からは、MKL を利用するバージョンの NAG ライブラリ `libnagc_mkl.a` または `libnagc_mkl.dylib` のご利用を推奨します。

NAG ライブラリはメモリリークが起きないように設計されています。メモリの解放は NAG ライブラリ自身によってか、もしくはユーザーが `NAG_FREE()` を呼び出すことによって行われます。しかしながら、NAG ライブラリが依存している他のライブラリ（コンパイラのランタイムライブラリなど）がメモリリークを起こすかもしれません。このため、NAG ライブラリをリンクしているプログラムに対して何らかのメモリトレースツールを使った際に、場合によってはメモリリークが検出されるかもしれません。リークするメモリの量はアプリケーションによって異なると思われますが、NAG ライブラリの呼び出し回数に比例して際限なく増加するものではありません。

本製品で提供される MKL version 2017.0.3 はマルチスレッド化されており、環境変数 `OMP_NUM_THREADS` が設定されていない場合、複数のプロセッサまたはマルチコアチップを持つシステムでは、計算速度の向上のためにマルチスレッドで計算を行います。もし、MKL に複数のプロセッサまたはコアを使わせたくない場合は、環境変数 `OMP_NUM_THREADS` に “1”（もしくは、必要なスレッド数）を設定してください。なお、X06 ルーチンは MKL のスレッドの振る舞いに影響を与えません。例えば、

C シェルの場合：

```
setenv OMP_NUM_THREADS 1
```

Bourne シェルの場合：

```
OMP_NUM_THREADS=1  
export OMP_NUM_THREADS
```

本ライブラリは MKL 10.3 よりも古いバージョンとは互換性がありません。

また、MKL には、条件付きビット単位の再現性 (Bit-wise Reproducibility (BWR)) オプションがあります。

ユーザーコードが一定の条件 (<https://software.intel.com/en-us/node/528579> 参照)

を満たしていれば、環境変数 MKL_CBWR を設定することにより BWR が有効になります。

詳細は MKL のドキュメントをご参照ください。しかしながら、多くの NAG ルーチンはこれらの条件を満たしていません。従って、MKL を利用するバージョンの NAG ライブラリの全ルーチンに対して、異なる CPU アーキテクチャに渡り MKL_CBWR による BWR を保証することはできません。BWR に関するより一般的な情報は、“How to Use the NAG Library and its Documentation” ドキュメントの「3.9.1 Bit-wise Reproducibility (BWR)」をご参照ください。

3.1. ライブラリのリンク方法

本セクションでは [INSTALL_DIR] に本ライブラリがインストールされていることが前提となります。

デフォルトの [INSTALL_DIR] は \$HOME/NAG/clmi6261dl となります。また、インストール時に [INSTALL_DIR] を指定することもできます。

MKL を利用するバージョンの NAG ライブラリを利用する場合は、以下のようにコンパイル・リンクを行ってください。（ここで driver.c がユーザープログラムです。）

スタティックライブラリを利用する場合：

```
icc driver.c -I[INSTALL_DIR]/include [INSTALL_DIR]/lib/libnagc_mkl.a \  
[INSTALL_DIR]/mkl_intel64_2017.0.3/lib/libmkl_intel_ilp64.a \  
[INSTALL_DIR]/mkl_intel64_2017.0.3/lib/libmkl_intel_thread.a \  
[INSTALL_DIR]/mkl_intel64_2017.0.3/lib/libmkl_core.a \  
[INSTALL_DIR]/rtl/libiomp5.a [INSTALL_DIR]/rtl/libifcoremt.a \  
-framework IOKit -framework CoreFoundation \  
-lpthread -lc++ -lm
```

共有ライブラリを利用する場合：

```
icc driver.c -I[INSTALL_DIR]/include [INSTALL_DIR]/lib/libnagc_mkl.dylib \  
-L[INSTALL_DIR]/mkl_intel64_2017.0.3/lib -lmkl_intel_ilp64 \  
-lmkl_intel_thread -lmkl_core \  
-L[INSTALL_DIR]/rtl \  
-liomp5 -lifcoremt \  
-framework IOKit -framework CoreFoundation \  
-lpthread -lc++ -lm
```

MKL を利用しないバージョンの NAG ライブラリを利用する場合は、以下のようにコンパイル・リンクを行ってください。（ここで driver.c がユーザープログラムです。）

スタティックライブラリを利用する場合：

```
icc driver.c -I[INSTALL_DIR]/include [INSTALL_DIR]/lib/libnagc_nag.a \  
[INSTALL_DIR]/rtl/libifcoremt.a \  
-framework IOKit -framework CoreFoundation \  
-lpthread -lc++ -lm
```

共有ライブラリを利用する場合：

```
icc driver.c -I[INSTALL_DIR]/include [INSTALL_DIR]/lib/libnagc_nag.dylib \  
[INSTALL_DIR]/rtl/libifcoremt.dylib \  
-framework IOKit -framework CoreFoundation \  
-lpthread -lc++ -lm
```

異なるコンパイラまたは Intel コンパイラ icc の異なるバージョンを使用する場合は、[INSTALL_DIR]/rtl ディレクトリに提供されるライブラリをリンクする必要があります。

例えば、clang を使用する場合は、以下のようにコンパイル・リンクを行ってください。

MKL を利用するバージョンの NAG スタティックライブラリを利用する場合：

```
clang driver.c -I[INSTALL_DIR]/include [INSTALL_DIR]/lib/libnagc_mkl.a \  
[INSTALL_DIR]/mkl_intel64_2017.0.3/lib/libmkl_intel_ilp64.a \  
[INSTALL_DIR]/mkl_intel64_2017.0.3/lib/libmkl_intel_thread.a \  
[INSTALL_DIR]/mkl_intel64_2017.0.3/lib/libmkl_core.a \  
[INSTALL_DIR]/rtl/libiomp5.a \  
[INSTALL_DIR]/rtl/libifcoremt.a \  
[INSTALL_DIR]/rtl/libimf.a \  
[INSTALL_DIR]/rtl/libirc.a \  
[INSTALL_DIR]/rtl/libsvml.a \  
-framework IOKit -framework CoreFoundation \  
-lstdc++ -lpthread -lm
```

MKL を利用するバージョンの NAG 共有ライブラリを利用する場合：

```
clang driver.c -I[INSTALL_DIR]/include [INSTALL_DIR]/lib/libnagc_mkl.dylib \  
-L[INSTALL_DIR]/mkl_intel64_2017.0.3/lib \  
-lmkl_intel_ilp64 -lmkl_intel_thread -lmkl_core \  
-L[INSTALL_DIR]/rtl \  
-liomp5 -lifcoremt -limf -lsvml -lintlc -lirng \  
-framework IOKit -framework CoreFoundation \  
-lstdc++ -lpthread -lm
```

MKL を利用しないバージョンの NAG スタティックライブラリを利用する場合：

```
clang driver.c -I[INSTALL_DIR]/include [INSTALL_DIR]/lib/libnagc_nag.a \  
[INSTALL_DIR]/rtl/libifcoremt.a [INSTALL_DIR]/rtl/libimf.a \  
[INSTALL_DIR]/rtl/libirc.a [INSTALL_DIR]/rtl/libsvml.a \  
-framework IOKit -framework CoreFoundation \  
-lstdc++ -lpthread -lm
```

MKL を利用しないバージョンの NAG 共有ライブラリを利用する場合：

```
clang driver.c -I[INSTALL_DIR]/include [INSTALL_DIR]/lib/libnagc_nag.dylib \  
[INSTALL_DIR]/rtl/libifcoremt.dylib \  
-framework IOKit -framework CoreFoundation \  
-lstdc++ -lpthread -lm
```

共有ライブラリを利用する場合には、環境変数 DYLD_LIBRARY_PATH を正しく設定し、実行時のリンクが行えるようにしてください。

C シェルの場合：

```
setenv DYLD_LIBRARY_PATH \  
[INSTALL_DIR]/lib:[INSTALL_DIR]/mkl_intel64_2017.0.3/lib
```

または、既存の設定がある場合には次のように拡張します。

```
setenv DYLD_LIBRARY_PATH \  
[INSTALL_DIR]/lib:[INSTALL_DIR]/mkl_intel64_2017.0.3/lib:${DYLD_LIBRARY_PATH}
```

Bourne シェルの場合：

```
DYLD_LIBRARY_PATH=[INSTALL_DIR]/lib:[INSTALL_DIR]/mkl_intel64_2017.0.3/lib  
export DYLD_LIBRARY_PATH
```

または、既存の設定がある場合には次のように拡張します。

```
DYLD_LIBRARY_PATH=[INSTALL_DIR]/lib:[INSTALL_DIR]/mkl_intel64_2017.0.3/lib:\  
${DYLD_LIBRARY_PATH}  
export DYLD_LIBRARY_PATH
```

場合に依っては（例えば、より新しいバージョンのコンパイラを使用する場合など）、その他のパス（例えば、コンパイラのランタイムライブラリなど）を DYLD_LIBRARY_PATH に設定する必要があるかもしれません。

3.2. Example プログラム

提供される Example 結果は、インストールノートの「2.2. 開発環境」に記載されている環境で生成されています。Example プログラムの実行結果は、異なる環境下（例えば、異なる C コンパイラ、異なるコンパイラライブラリ、異なる BLAS または LAPACK ルーチンなど）で若干異なる場合があります。そのような違いが顕著な計算結果としては、固有ベクトル（スカラー（多くの場合 -1）倍の違い）、反復回数や関数評価、残差（その他マシン精度と同じくらい小さい量）などがあげられます。

提供される Example 結果は NAG スタティックライブラリ `libnagc_mkl.a`（MKL 提供の BLAS / LAPACK ルーチンを使用）を用いて算出されています。NAG 提供の BLAS / LAPACK ルーチンを使用した場合、結果が僅かに異なるかもしれません。

Example プログラムは本ライブラリが想定する動作環境に適した状態で提供されます。そのため、ライブラリマニュアルに記載されている Example プログラムに比べて、その内容が若干異なる場合があります。

以下のスクリプトを用いて Example プログラムを簡単に利用する事ができます。
（これらのスクリプトは、`[INSTALL_DIR]/scripts` ディレクトリに提供されます。）

- `nagc_example_mkl`
NAG スタティックライブラリ `libnagc_mkl.a` および本製品で提供される MKL ライブラリをリンクします。
- `nagc_example_shar_mkl`
NAG 共有ライブラリ `libnagc_mkl.dylib` および本製品で提供される MKL ライブラリをリンクします。
- `nagc_example`
NAG スタティックライブラリ `libnagc_nag.a` をリンクします。
- `nagc_example_shar`
NAG 共有ライブラリ `libnagc_nag.dylib` をリンクします。

これらのスクリプトは、Example プログラムのソースファイル（必要に応じて、データファイル、オプションファイルその他）をカレントディレクトリにコピーして、コンパイル・リンク・実行を行います。

ご利用の NAG ライブラリルーチンの名前をスクリプトの引数に指定してください。

例)

```
nagc_example_mkl e04ucc
```

この例では、e04ucce.f (ソースファイル)、e04ucce.d (データファイル)、e04ucce.opt (オプションファイル) をカレントディレクトリにコピーして、コンパイル・リンク・実行を行い e04ucce.r (結果ファイル) を生成します。

3.3. データ型

NAG データ型 Integer と Pointer は、本ライブラリでは以下のように定義されています。

NAG 型	C 型	サイズ (バイト)
Integer	long	8
Pointer	void *	8

sizeof(Integer) と sizeof(Pointer) の値は a00aac の Example プログラムから得ることもできます。その他の NAG データ型の情報はライブラリマニュアル (「5. ドキュメント」参照) の “How to Use the NAG Library and its Documentation” ドキュメントをご参照ください。

3.4. メンテナンスレベル

ライブラリのメンテナンスレベルは、ライブラリルーチン a00aac の Example プログラムをコンパイル・リンク・実行することにより確認することができます。この時、スクリプト nagc_example* を引数 a00aac と共に用いれば、Example プログラムのコンパイル・リンク・実行を容易に行うことができます (「3.2. Example プログラム」参照)。ライブラリルーチン a00aac はライブラリの詳細 (タイトル, 製品コード, 使用されるコンパイラおよび精度, バージョン (Mark) など) を出力します。

4. ルーチン固有の情報

本ライブラリルーチン固有の情報を（チャプター毎に）以下に示します.

a. f06, f07, f08, f16

libnagc_mkl.a, libnagc_mkl.dylib は, MKL の BLAS/LAPACK ルーチンを使用します.
ただし, 以下の BLAS/LAPACK ルーチンは, MKL バージョンの使用に問題があるため,
NAG バージョンが使用されます（呼び出されます）.

None

b. s10 – s21

これらのチャプターの関数の動作は, ライブラリ実装毎に異なります.

一般的な詳細はライブラリマニュアルをご参照ください.

本ライブラリ固有の値を以下に示します.

s10aac $E_1 = 1.8715e+1$

s10abc $E_1 = 7.080e+2$

s10acc $E_1 = 7.080e+2$

s13aac $x_{hi} = 7.083e+2$

s13acc $x_{hi} = 1.0e+16$

s13adc $x_{hi} = 1.0e+17$

s14aac fail.code = NE_REAL_ARG_GT if $x > 1.70e+2$

fail.code = NE_REAL_ARG_LT if $x < -1.70e+2$

fail.code = NE_REAL_ARG_TOO_SMALL if $\text{abs}(x) < 2.23e-308$

s14abc fail.code = NE_REAL_ARG_GT if $x > x_{big} = 2.55e+305$

s15adc $x_{hi} = 2.65e+1$

s15aec $x_{hi} = 2.65e+1$

s15agc fail.code = NW_HI if $x \geq 2.53e+307$

fail.code = NW_REAL if $4.74e+7 \leq x < 2.53e+307$

```

fail.code = NW_NEG if  $x < -2.66e+1$ 

s17acc fail.code = NE_REAL_ARG_GT if  $x > 1.0e+16$ 
s17adc fail.code = NE_REAL_ARG_GT if  $x > 1.0e+16$ 
        fail.code = NE_REAL_ARG_TOO_SMALL if  $0 < x \leq 2.23e-308$ 
s17aec fail.code = NE_REAL_ARG_GT if  $\text{abs}(x) > 1.0e+16$ 
s17afc fail.code = NE_REAL_ARG_GT if  $\text{abs}(x) > 1.0e+16$ 
s17agc fail.code = NE_REAL_ARG_GT if  $x > 1.038e+2$ 
        fail.code = NE_REAL_ARG_LT if  $x < -5.7e+10$ 
s17ahc fail.code = NE_REAL_ARG_GT if  $x > 1.041e+2$ 
        fail.code = NE_REAL_ARG_LT if  $x < -5.7e+10$ 
s17ajc fail.code = NE_REAL_ARG_GT if  $x > 1.041e+2$ 
        fail.code = NE_REAL_ARG_LT if  $x < -1.9e+9$ 
s17akc fail.code = NE_REAL_ARG_GT if  $x > 1.041e+2$ 
        fail.code = NE_REAL_ARG_LT if  $x < -1.9e+9$ 
s17dcc fail.code = NE_OVERFLOW_LIKELY if  $\text{abs}(z) < 3.92223e-305$ 
        fail.code = NW_SOME_PRECISION_LOSS if  $\text{abs}(z)$  or  $\text{fnu}+n-1 > 3.27679e+4$ 
        fail.code = NE_TOTAL_PRECISION_LOSS if  $\text{abs}(z)$  or  $\text{fnu}+n-1 > 1.07374e+9$ 
s17dec fail.code = NE_OVERFLOW_LIKELY if  $\text{AIMAG}(z) > 7.00921e+2$ 
        fail.code = NW_SOME_PRECISION_LOSS if  $\text{abs}(z)$  or  $\text{fnu}+n-1 > 3.27679e+4$ 
        fail.code = NE_TOTAL_PRECISION_LOSS if  $\text{abs}(z)$  or  $\text{fnu}+n-1 > 1.07374e+9$ 
s17dgc fail.code = NW_SOME_PRECISION_LOSS if  $\text{abs}(z) > 1.02399e+3$ 
        fail.code = NE_TOTAL_PRECISION_LOSS if  $\text{abs}(z) > 1.04857e+6$ 
s17dhc fail.code = NW_SOME_PRECISION_LOSS if  $\text{abs}(z) > 1.02399e+3$ 
        fail.code = NE_TOTAL_PRECISION_LOSS if  $\text{abs}(z) > 1.04857e+6$ 
s17dlc fail.code = NE_OVERFLOW_LIKELY if  $\text{abs}(z) < 3.92223e-305$ 
        fail.code = NW_SOME_PRECISION_LOSS if  $\text{abs}(z)$  or  $\text{fnu}+n-1 > 3.27679e+4$ 
        fail.code = NE_TOTAL_PRECISION_LOSS if  $\text{abs}(z)$  or  $\text{fnu}+n-1 > 1.07374e+9$ 

s18adc fail.code = NE_REAL_ARG_TOO_SMALL if  $0 < x \leq 2.23e-308$ 
s18aec fail.code = NE_REAL_ARG_GT if  $\text{abs}(x) > 7.116e+2$ 
s18afc fail.code = NE_REAL_ARG_GT if  $\text{abs}(x) > 7.116e+2$ 
s18dcc fail.code = NE_OVERFLOW_LIKELY if  $\text{abs}(z) < 3.92223e-305$ 
        fail.code = NW_SOME_PRECISION_LOSS if  $\text{abs}(z)$  or  $\text{fnu}+n-1 > 3.27679e+4$ 
        fail.code = NE_TOTAL_PRECISION_LOSS if  $\text{abs}(z)$  or  $\text{fnu}+n-1 > 1.07374e+9$ 
s18dec fail.code = NE_OVERFLOW_LIKELY if  $\text{REAL}(z) > 7.00921e+2$ 

```

fail.code = NW_SOME_PRECISION_LOSS if $\text{abs}(z)$ or $\text{fnu}+n-1 > 3.27679\text{e}+4$
fail.code = NE_TOTAL_PRECISION_LOSS if $\text{abs}(z)$ or $\text{fnu}+n-1 > 1.07374\text{e}+9$

s19aac fail.code = NE_REAL_ARG_GT if $\text{abs}(x) \geq 5.04818\text{e}+1$
s19abc fail.code = NE_REAL_ARG_GT if $\text{abs}(x) \geq 5.04818\text{e}+1$
s19acc fail.code = NE_REAL_ARG_GT if $x > 9.9726\text{e}+2$
s19adc fail.code = NE_REAL_ARG_GT if $x > 9.9726\text{e}+2$

s21bcc fail.code = NE_REAL_ARG_LT if an argument $< 1.583\text{e}-205$
fail.code = NE_REAL_ARG_GE if an argument $\geq 3.765\text{e}+202$
s21bdc fail.code = NE_REAL_ARG_LT if an argument $< 2.813\text{e}-103$
fail.code = NE_REAL_ARG_GT if an argument $\geq 1.407\text{e}+102$

c. x01

以下の数学定数がヘッダーファイル nagx01.h に提供されます。

X01AAC (π) = 3.1415926535897932
X01ABC (γ) = 0.5772156649015328

d. x02

以下のマシン定数がヘッダーファイル nagx02.h に提供されます。

浮動小数点演算の基本的なパラメーター：

X02BHC = 2
X02BJC = 53
X02BKC = -1021
X02BLC = 1024

浮動小数点演算の派生的なパラメーター：

X02AJC = 1.11022302462516e-16
X02AKC = 2.22507385850721e-308
X02ALC = 1.79769313486231e+308

X02AMC = 2. 22507385850721e-308

X02ANC = 2. 22507385850721e-308

コンピューター環境のその他のパラメーター :

X02AHC = 1. 42724769270596e+45

X02BBC = 9223372036854775807

X02BEC = 15

5. ドキュメント

ライブラリマニュアルは本製品の一部として提供されます。

また、NAG のウェブサイトからダウンロードすることもできます。

ライブラリマニュアルの最新版は以下のウェブサイトをご参照ください。

<http://www.nag.co.uk/content/nag-c-library-manual>

ライブラリマニュアルは以下の形式で提供されます。

- HTML5 - HTML/MathML マニュアル（各ドキュメントの PDF 版へのリンクを含む）
- PDF - PDF マニュアル（PDF のしおり、または HTML 目次ファイルから閲覧する）

これらの形式に対して、以下の目次ファイルが提供されます。

nagdoc_cl26/html/frontmatter/manconts.html

nagdoc_cl26/pdf/frontmatter/manconts.pdf

nagdoc_cl26/pdf/frontmatter/manconts.html

また、これらの目次ファイルへのリンクをまとめたマスター目次ファイルが提供されます。

nagdoc_cl26/index.html

各形式の閲覧方法および操作方法については、以下のドキュメントをご参照ください。

http://www.nag.co.uk/numeric/cl/nagdoc_cl26/html/genint/essint.html#onlinedoc

加えて、以下のドキュメントが提供されます。

- in.html - インストールノート（英語版）
- un.html - ユーザーノート（英語版）

6. サポート

製品のご利用に関してご質問等がございましたら、電子メールにて「日本 NAG ヘルプデスク」までお問い合わせください。その際、ご利用の製品の製品コード（CLMI6261DL）並びに、お客様の User ID をご明記いただきますようお願い致します。
ご返答は平日 9：30 ～ 17:30 に行わせていただきます。

日本 NAG ヘルプデスク

Email: naghelp@nag-j.co.jp

7. コンタクト情報

日本ニューメリカルアルゴリズムズグループ株式会社（日本 NAG）

〒104-0032

東京都中央区八丁堀 4-9-9 八丁堀フロンティアビル 2F

Email: sales@nag-j.co.jp

Tel: 03-5542-6311

Fax: 03-5542-6312

NAG のウェブサイトでは製品およびサービスに関する情報を定期的に更新しています。

<http://www.nag-j.co.jp/> （日本）

<http://www.nag.co.uk/> （英国本社）

<http://www.nag.com/> （米国）