NAG Fortran Library, Mark 26, Multithreaded

FSL6I26DCL - License Managed

Linux 64 (Intel 64 / AMD64), Intel Fortran, Double Precision, 32-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

3.1.1 使用するスレッド数の設定 7

3.1.2. C または C++ からのライブラリの呼び出し 9

3.2. インターフェースブロック 10

3.3. Example プログラム 12

3.4. Fortran 型と強調斜体文字の解釈 14

3.5. メンテナンスレベル 15

4. ルーチン固有の情報 16

5. ドキュメント 21

6. サポート 22

7. コンタクト情報 22

1. イントロダクション

本ユーザーノートは，NAG Fortran Library，Mark 26，Multithreaded - FSL6I26DCL（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/fs26/l6idcl/supplementary.html

3. 一般情報

本製品では，Intel® Math Kernel Library for Linux（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 libnag_mkl.a および共有ライブラリ libnag_mkl.so 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 libnag_nag.a および共有ライブラリ libnag_nag.so を提供します．本ライブラリは，MKL version 11.3.3 を用いてテストされています．MKL version 11.3.3 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 libnag_mkl.a または libnag_mkl.so のご利用を推奨します．

NAG ライブラリをマルチスレッドアプリケーションから使用する場合の注意点については，"How to Use the NAG Library and its Documentation" ドキュメントの「3.12.1 スレッドセーフ」をご参照ください．また，本製品に付属の MKL をマルチスレッドアプリケーションから使用する場合の詳細については，以下の Intel 社のウェブサイトをご参照ください．

http://software.intel.com/en-us/articles/intel-math-kernel-library-intel-mkl-using-intel-mkl-with-threaded-applications

本ライブラリは OpenMP を用いてコンパイルされています．また，異なるコンパイラ間では OpenMP ランタイムライブラリの互換性は保証されません．従って，自身のコードが OpenMP を利用している場合は，インストールノートに記載されているコンパイラを使用することを推奨します．なお，システムのデフォルトのスレッドスタックサイズは，マルチスレッドアプリケーション内の全ての NAG ライブラリルーチンを実行するには十分ではないことに注意してください．スレッドスタックサイズは OpenMP 環境変数 OMP_STACKSIZE で増やすことができます．

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．一定の条件（https://software.intel.com/en-us/node/528579 参照）をユーザーコードが満たしていれば，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「3.11.1 Bit-wise Reproducibility (BWR)」をご参照ください．

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．

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/fsl6i26dcl 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.f90 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：

```
ifort -qopenmp -I[INSTALL_DIR]/nag_interface_blocks driver.f90 \
    [INSTALL_DIR]/lib/libnag_mkl.a \
    -Wl,--start-group \
    [INSTALL_DIR]/mkl_intel64_11.3.3/lib/libmkl_intel_lp64.a \
    [INSTALL_DIR]/mkl_intel64_11.3.3/lib/libmkl_intel_thread.a \
    [INSTALL_DIR]/mkl_intel64_11.3.3/lib/libmkl_core.a \
    -Wl,--end-group \
    [INSTALL_DIR]/rtl/intel64/libiomp5.a -lpthread -lm -ldl -lstdc++
```

共有ライブラリを利用する場合：

```
ifort -qopenmp -I[INSTALL_DIR]/nag_interface_blocks driver.f90 \
    [INSTALL_DIR]/lib/libnag_mkl.so \
    -L[INSTALL_DIR]/mkl_intel64_11.3.3/lib \
    -lmkl_intel_lp64 -lmkl_intel_thread -lmkl_core \
    -L[INSTALL_DIR]/rtl/intel64 -liomp5 -lpthread -lm -ldl
```

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：

```
ifort -qopenmp -I[INSTALL_DIR]/nag_interface_blocks driver.f90 \
    [INSTALL_DIR]/lib/libnag_nag.a -lstdc++
```

共有ライブラリを利用する場合：

```
ifort -qopenmp -I[INSTALL_DIR]/nag_interface_blocks driver.f90 \
    [INSTALL_DIR]/lib/libnag_nag.so
```

共有ライブラリを利用する場合には，環境変数 LD_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 LD_LIBRARY_PATH [INSTALL_DIR]/lib:[INSTALL_DIR]/mkl_intel64_11.3.3/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 LD_LIBRARY_PATH \
  [INSTALL_DIR]/lib:[INSTALL_DIR]/mkl_intel64_11.3.3/lib:${LD_LIBRARY_PATH}
```

Bourne シェルの場合：

```
LD_LIBRARY_PATH=[INSTALL_DIR]/lib:[INSTALL_DIR]/mkl_intel64_11.3.3/lib
export LD_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.

```
LD_LIBRARY_PATH=[INSTALL_DIR]/lib:\
  [INSTALL_DIR]/mkl_intel64_11.3.3/lib:${LD_LIBRARY_PATH}
export LD_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 LD_LIBRARY_PATH に設定する必要があるかもしれません.

また，Intel コンパイラの異なるバージョンを使用する場合は，[INSTALL_DIR]/rtl/intel64 ディレクトリに提供される Intel コンパイラのランタイムライブラリをリンクする必要があるかもしれません.

3.1.1. スレッド数の設定

本ライブラリと MKL は，OpenMP を用いてマルチスレッドを実装しています.
実行時に使用されるスレッド数を環境変数 OMP_NUM_THREADS に設定してください.

C シェルの場合：
setenv OMP_NUM_THREADS N

Bourne シェルの場合：
OMP_NUM_THREADS=N
export OMP_NUM_THREADS

N はご利用のスレッド数です. 環境変数 OMP_NUM_THREADS はプログラムの実行毎に再設定することができます. プログラムの異なる部分で，使用するスレッド数を変更したい場合は，NAG ライブラリのチャプター X06 のルーチンがご利用いただけます.

NAG ライブラリと MKL のいくつかのルーチンは，複数レベルの OpenMP 並列処理を持ちます. これらのルーチンは，ユーザーアプリケーションの OpenMP 並列領域内から呼び出すこともできます. デフォルトでは，OpenMP ネスト並列処理は無効になっており，最も外側の並列領域だけが N スレッドで実行されます. 内部レベルはアクティブにならず，1 スレッドで実行されます. OpenMP 環境変数 OMP_NESTED の値を確認・設定するか，もしくはチャプター X06 のルーチンを使用して，OpenMP ネスト並列処理の有効／無効の確認・設定を行うことができます. OpenMP ネスト並列処理が有効になっている場合，上位レベルの各スレッドの各並列領域に N 個のスレッドが作成されるため，例えば，2 つのレベルの OpenMP 並列処理がある場合，合計 N * N スレッドになります. ネスト並列処理では，各レベルで必要なスレッド数を，環境変数 OMP_NUM_THREADS にカンマ区切りで指定することができます.

C シェルの場合：
setenv OMP_NUM_THREADS N, P

Bourne シェルの場合：
OMP_NUM_THREADS=N, P
export OMP_NUM_THREADS

この設定例では，第 1 レベルの並列処理に対して N 個のスレッドが生成され，内部レベルの並列処理に対して P 個のスレッドが生成されます.

注意：環境変数 OMP_NUM_THREADS が設定されていない場合，デフォルト値はコンパイラ毎，またベンダーライブラリ毎に異なります．通常は，１もしくはシステムで使用可能な最大コア数に等しくなります．特に後者では，システムを他のユーザーと共有している場合や，自分のアプリケーション内で複数レベルの並列処理を実行している場合に問題となる可能性があります．従って，OMP_NUM_THREADS は明示的に設定することをお勧めします.

一般的に，推奨されるスレッドの最大数は，ご利用の共有メモリシステムの物理コア数です．ただし，殆どの Intel プロセッサはハイパースレッディングと呼ばれる機能をサポートしています．この機能は１つの物理コアが同時に２つのスレッドをサポートすることを可能にします（従って，オペレーティングシステムには２つの論理コアとして認識されます）．この機能が有益かどうかは，使用するアルゴリズムや問題のサイズに依存します．従って，自身のアプリケーションにとってこの機能が有益かどうかは，追加の論理コアを使用する場合と使用しない場合でベンチマークを取り判断することをお勧めします．これは，使用するスレッド数を OMP_NUM_THREADS に設定するだけで簡単に実現できます．ハイパースレッディングの完全な無効化は，通常，起動時にシステムの BIOS 設定で行うことができます.

3.1.2. C または C++ からのライブラリの呼び出し

本ライブラリは C または C++ 環境からもご利用いただけます.
ご利用の支援として Fortran と C の間の型マッピング情報を持った C/C++ ヘッダーファイル [INSTALL_DIR]/c_headers/nagmk26.h が提供されます. ヘッダーファイルから必要な部分だけを（ファイルの先頭にある #defines なども忘れずに）自身のプログラムにコピー＆ペーストするか, もしくはヘッダーファイルを単純にインクルードしてご利用ください.

C または C++ から NAG Fortran Library を呼び出す際のアドバイスは, ドキュメント [INSTALL_DIR]/c_headers/techdoc.html をご参照ください.

3.2. インターフェースブロック

NAG Fortran Library インターフェースブロック（引用仕様宣言）はライブラリルーチンの型と引数を定義します. Fortran プログラムからライブラリルーチンを呼び出す際に必ず必要という性質のものではありませんが（ただし本製品で提供される Example を利用する際には必要となります), これを用いることでライブラリルーチンが正しく呼び出されているかどうかのチェックを Fortran コンパイラに任せる事ができます. 具体的にはコンパイラが以下のチェックを行うことを可能にします.

(a) サブルーチン呼び出しの整合性
(b) 関数宣言の型
(c) 引数の数
(d) 引数の型

NAG Fortran Library インターフェースブロックファイルはチャプター毎のモジュールとして提供されますが, これらをまとめて一つにしたモジュールが提供されます.

nag_library

これらのモジュールは, Intel Fortran コンパイラ用にプリコンパイルされた形式（*.mod ファイル）で提供されます.
コンパイル時に, -I"pathname" オプションを用いて, モジュールファイルが置かれているディレクトリのパス（[INSTALL_DIR]/nag_interface_blocks）を指定してください.

提供されるモジュールファイル（.mod ファイル）はインストールノートの「2.2. 開発環境」にあるコンパイラを用いて生成されています. モジュールファイルはコンパイラ依存のファイルであるため, ご利用のコンパイラとの間に互換性がない場合は, ご利用のコンパイラでモジュールファイルを生成する必要があります.（自身のプログラムでインターフェースブロックをご利用にならないのであれば, この限りではありません. ただし, Example プログラムはインターフェースブロックを利用しますので, Example プログラムをご利用になる場合は必要です.）

提供されるスクリプト nag_recompile_mods を用いて，モジュールファイルのセットを指定のディレクトリ（例えば nag_interface_blocks_alt）に生成することができます.
例）
[INSTALL_DIR]/scripts/nag_recompile_mods nag_interface_blocks_alt

このスクリプトは PATH 環境変数に設定された Intel Fortran コンパイラを使用します.
[INSTALL_DIR]/scripts/nag_recompile_mods を実行する前に，ご利用の Intel Fortran コンパイラの環境設定スクリプトを実行しておくと安心です.
新しいモジュールファイルのセットをデフォルトのセットとして使用するには，元のモジュールファイルのセットを含むディレクトリ [INSTALL_DIR]/nag_interface_block の名前を [INSTALL_DIR]/nag_interface_blocks_original に変更し，新しいモジュールファイルのセットを含むディレクトリ [INSTALL_DIR]/nag_interface_blocks_alt の名前を [INSTALL_DIR]/nag_interface_blocks に変更してください.

3.3. 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 Fortran 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 libnag_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ことができます．
（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．）

- nag_example_mkl
  NAG スタティックライブラリ libnag_mkl.a および本製品で提供される MKL をリンクします．
- nag_example_shar_mkl
  NAG 共有ライブラリ libnag_mkl.so および本製品で提供される MKL をリンクします．

- nag_example
  NAG スタティックライブラリ libnag_nag.a をリンクします．

- nag_example_shar
  NAG 共有ライブラリ libnag_nag.so 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と OpenMP スレッド数をスクリプトの引数に指定してください.
例）
nag_example_mkl e04nrf 4

この例では, e04nrfe.c（ソースファイル), e04nrf e.d（データファイル), e04nrfe.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, および 4 OpenMP スレッドで実行を行い, e04nrfe.r（結果ファイル）を生成します.

3.4. Fortran 型と強調斜体文字の解釈

本ライブラリは 32-bit 整数を使用します.

ライブラリとライブラリマニュアルでは浮動小数点変数を以下のようにパラメーター化された型を用いて記述しています.

REAL(KIND=nag_wp)

ここで nag_wp は Fortran の種別パラメーターを表しています.
nag_wp の値は製品毎に異なり, その値は nag_library モジュールに定義されています.

これに加え, いくつかのルーチンで以下の型が使用されます.

REAL(KIND=nag_rp)

これらの型の使用例については, 各種 Example プログラムをご参照ください.

本ライブラリでは, これらの型は次のような意味を持っています.

REAL (kind=nag_rp)　　- REAL (単精度実数)
REAL (kind=nag_wp)　　- DOUBLE PRECISION (倍精度実数)
COMPLEX (kind=nag_rp) - COMPLEX (単精度複素数)
COMPLEX (kind=nag_wp) - 倍精度複素数 (e.g. COMPLEX*16)

上記に加え, ライブラリマニュアルでは強調斜体文字を用いていくつかの用語を表現しています. 詳細は "How to Use the NAG Library and its Documentation" の「4.4 実装依存情報」をご参照ください.

3.5. 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 A00AAF 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 nag_example* を引数 a00aaf 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（「3.3. Example プログラム」参照）．ライブラリルーチン A00AAF 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. OpenMP 並列領域からユーザー関数を呼び出すルーチン

以下の NAG ルーチンはルーチン内の OpenMP 並列領域からユーザー関数を呼び出します.

D03RAF D03RBF E05SAF E05SBF E05UCF E05USF F01ELF F01EMF F01FLF F01FMF F01JBF F01JCF F01KBF F01KCF

従って，本ライブラリの製造に用いられたコンパイラと同じコンパイラを使用する限り，ユーザー関数で orphaned OpenMP 指示文を使うことができます. また，ユーザー用のワークスペース配列 IUSER と RUSER もスレッドセーフである必要があります. これらの配列は読み取り専用のデータをユーザー関数に与えるためにだけ使用するのがベストです.

b. C06

以下の NAG ルーチンは可能な限り本製品で提供される MKL の
Intel Discrete Fourier Transforms Interface（DFTI）ルーチンを利用します.

C06PAF C06PCF C06PFF C06PJF C06PKF C06PPF C06PQF C06PRF C06PSF C06PUF C06PVF C06PWF C06PXF C06PYF C06PZF C06RAF C06RBF C06RCF C06RDF

Intel DFTI ルーチンは必要なワークスペースを自身で内部的に割り当てます. 従って，上記のルーチンの引数 WORK（ワークスペース配列）のサイズは，それぞれの
Routine Document に指示されている値で十分です（変更の必要はありません）.

c. F06, F07, F08, F16

チャプター F06, F07, F08, F16 においては BLAS／LAPACK 由来のルーチンに対して別個のルーチン名が用意されています. これらのルーチン名については，関係する Chapter Introduction をご参照ください. パフォーマンス面からは，NAG スタイルの名前よりも BLAS／LAPACK スタイルの名前でルーチンをご利用ください.

多くの LAPACK ルーチンは "workspace query" メカニズムを利用します. ルーチン呼び出し側にどれだけのワークスペースが必要であるかを問い合わせるメカニズムですが, NAG 提供の LAPACK と MKL 提供の LAPACK ではこのワークスペースのサイズが異なる場合があるので注意してください.

libnag_mkl.a, libnag_mkl.so では, BLAS／LAPACK ルーチンは MKL 提供のものが使われます. ただし, 以下のルーチンは NAG 提供のものが使われます.

BLAS_DAMAX_VAL	BLAS_DAMIN_VAL	BLAS_DAXPBY	BLAS_DDOT	BLAS_DMAX_VAL
BLAS_DMIN_VAL	BLAS_DSUM	BLAS_DWAXPBY	BLAS_ZAMAX_VAL	BLAS_ZAMIN_VAL
BLAS_ZAXPBY	BLAS_ZSUM	BLAS_ZWAXPBY		

libnag_mkl.a, libnag_mkl.so では, 以下の NAG ルーチンはベンダーライブラリから LAPACK ルーチンを呼び出すためのラッパーです.

F07ADF/DGETRF	F07AEF/DGETRS	F07ARF/ZGETRF	F07ASF/ZGETRS	F07BEF/DGBTRS
F07BSF/ZGBTRS	F07FDF/DPOTRF	F07FEF/DPOTRS	F07FRF/ZPOTRF	F07FSF/ZPOTRS
F07GEF/DPPTRS	F07GSF/ZPPTRS	F07HEF/DPBTRS	F07HSF/ZPBTRS	F08AEF/DGEQRF
F08AGF/DORMQR	F08ASF/ZGEQRF	F08AUF/ZUNMQR	F08FEF/DSYTRD	F08FSF/ZHETRD
F08GEF/DSPTRD	F08GFF/DOPGTR	F08GSF/ZHPTRD	F08GTF/ZUPGTR	F08JEF/DSTEQR
F08JSF/ZSTEQR	F08KEF/DGEBRD	F08KSF/ZGEBRD	F08MEF/DBDSQR	F08MSF/ZBDSQR

d. S07 - 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.

S07AAF F_1 = 1.0E+13
F_2 = 1.0E-14

S10AAF E_1 = 1.8715E+1
S10ABF E_1 = 7.080E+2
S10ACF E_1 = 7.080E+2

S13AAF x_hi = 7.083E+2
S13ACF x_hi = 1.0E+16
S13ADF x_hi = 1.0E+17

S14AAF IFAIL = 1 if X > 1.70E+2
        IFAIL = 2 if X < -1.70E+2
        IFAIL = 3 if abs(X) < 2.23E-308
S14ABF IFAIL = 2 if X > x_big = 2.55E+305

S15ADF x_hi = 2.65E+1
S15AEF x_hi = 2.65E+1
S15AGF IFAIL = 1 if X >= 2.53E+307
        IFAIL = 2 if 4.74E+7 <= X < 2.53E+307
        IFAIL = 3 if X < -2.66E+1

S17ACF IFAIL = 1 if X > 1.0E+16
S17ADF IFAIL = 1 if X > 1.0E+16
        IFAIL = 3 if 0 < X <= 2.23E-308
S17AEF IFAIL = 1 if abs(X) > 1.0E+16
S17AFF IFAIL = 1 if abs(X) > 1.0E+16
S17AGF IFAIL = 1 if X > 1.038E+2
        IFAIL = 2 if X < -5.7E+10
S17AHF IFAIL = 1 if X > 1.041E+2
        IFAIL = 2 if X < -5.7E+10
S17AJF IFAIL = 1 if X > 1.041E+2
        IFAIL = 2 if X < -1.9E+9
S17AKF IFAIL = 1 if X > 1.041E+2
        IFAIL = 2 if X < -1.9E+9
S17DCF IFAIL = 2 if abs(Z) < 3.92223E-305
        IFAIL = 4 if abs(Z) or FNU+N-1 > 3.27679E+4
        IFAIL = 5 if abs(Z) or FNU+N-1 > 1.07374E+9
S17DEF IFAIL = 2 if AIMAG(Z) > 7.00921E+2
        IFAIL = 3 if abs(Z) or FNU+N-1 > 3.27679E+4
        IFAIL = 4 if abs(Z) or FNU+N-1 > 1.07374E+9
S17DGF IFAIL = 3 if abs(Z) > 1.02399E+3
        IFAIL = 4 if abs(Z) > 1.04857E+6

S17DHF IFAIL = 3 if abs(Z) > 1.02399E+3
        IFAIL = 4 if abs(Z) > 1.04857E+6
S17DLF IFAIL = 2 if abs(Z) < 3.92223E-305
        IFAIL = 4 if abs(Z) or FNU+N-1 > 3.27679E+4
        IFAIL = 5 if abs(Z) or FNU+N-1 > 1.07374E+9

S18ADF IFAIL = 2 if 0 < X <= 2.23E-308
S18AEF IFAIL = 1 if abs(X) > 7.116E+2
S18AFF IFAIL = 1 if abs(X) > 7.116E+2
S18DCF IFAIL = 2 if abs(Z) < 3.92223E-305
        IFAIL = 4 if abs(Z) or FNU+N-1 > 3.27679E+4
        IFAIL = 5 if abs(Z) or FNU+N-1 > 1.07374E+9
S18DEF IFAIL = 2 if REAL(Z) > 7.00921E+2
        IFAIL = 3 if abs(Z) or FNU+N-1 > 3.27679E+4
        IFAIL = 4 if abs(Z) or FNU+N-1 > 1.07374E+9

S19AAF IFAIL = 1 if abs(X) >= 5.04818E+1
S19ABF IFAIL = 1 if abs(X) >= 5.04818E+1
S19ACF IFAIL = 1 if X > 9.9726E+2
S19ADF IFAIL = 1 if X > 9.9726E+2

S21BCF IFAIL = 3 if an argument < 1.583E-205
        IFAIL = 4 if an argument >= 3.765E+202
S21BDF IFAIL = 3 if an argument < 2.813E-103
        IFAIL = 4 if an argument >= 1.407E+102

e. X01

数学定数は以下のとおりです.

X01AAF (pi)   = 3.1415926535897932
X01ABF (gamma) = 0.5772156649015328

f. X02

マシン定数は以下のとおりです.

浮動小数点演算の基本的なパラメーター:

X02BHF = 2
X02BJF = 53
X02BKF = -1021
X02BLF = 1024

浮動小数点演算の派生的なパラメーター:

X02AJF = 1.11022302462516E-16
X02AKF = 2.22507385850721E-308
X02ALF = 1.79769313486231E+308
X02AMF = 2.22507385850721E-308
X02ANF = 2.22507385850721E-308

コンピューター環境のその他のパラメーター:

X02AHF = 1.42724769270596E+45
X02BBF = 2147483647
X02BEF = 15

g. X04

エラーメッセージおよびアドバイスメッセージのデフォルトの出力先装置番号は 6 番となります.

h. X06

本チャプターのルーチンは, 本ライブラリの MKL スレッドの動作も変更します.

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-fortran-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_fl26/html/frontmatter/manconts.html
nagdoc_fl26/pdf/frontmatter/manconts.pdf
nagdoc_fl26/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_fl26/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/fl/nagdoc_fl26/html/genint/essint.html

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）

MKL の詳細については, 以下の Intel 社のウェブサイトをご参照ください.

https://software.intel.com/intel-mkl

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（FSL6I26DCL）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 9：30～12:00，13:00～17:30 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）