NAG Library, Mark 27.1
NLW6I271EL - Licence Managed
Microsoft Windows x64, 64-bit, Intel C/C++ or Microsoft C/C++ or Intel Fortran,
32-bit integers

ユーザーノート

内容

1. イントロダクション

本ユーザーノートは，NAG Library，Mark 27.1 - NLW6I271EL（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 27.1（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
(b) Chapter Introduction
(c) Routine 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.
https://www.nag.com/doc/inun/nl27/w6i1el/supplementary.html

3. 一般情報

本ライブラリは, Intel ® Math Kernel Library for Windows (MKL) が提供する BLAS／LAPACK ルーチンを利用するライブラリ（スタティック版と DLL 版）と, NAG が提供する BLAS／LAPACK ルーチンを利用するライブラリ(スタティック版と DLL 版)を提供します.

本ライブラリは, MKL version 2020.0.2 を用いてテストされています. MKL version 2020.0.2 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/en-us/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 nag_mkl_MT.lib, nag_mkl_MD.lib, NLW6I271E_mkl.lib／NLW6I271E_mkl.dll のご利用を推奨します. これらのライブラリは NAG が提供する BLAS／LAPACK ルーチンを含みません.

また, MKL を利用しないバージョンの NAG ライブラリ nag_nag_MT.lib, nag_nag_MD.lib, NLW6I271E_nag.lib／NLW6I271E_nag.dll が提供されます. これらのライブラリは NAG が提供する BLAS／LAPACK ルーチンを含んでいます.

NAG ライブラリのスタティック版をご利用の場合は, 共にリンクされる Microsoft ランタイムライブラリに従って, NAG ライブラリを選択する必要があります. マルチスレッドスタティックランタイムライブラリと共にリンクする場合は, nag_mkl_MT.lib または nag_nag_MT.lib をご利用ください. または, マルチスレッド DLL ランタイムライブラリと共にリンクする場合は, nag_mkl_MD.lib または nag_nag_MD.lib をご利用ください.

NAG ライブラリの DLL 版をご利用の場合は, インポートライブラリ NLW6I271E_mkl.lib または NLW6I271E_nag.lib をリンクしてください. 実行時には, 対応する DLL ファイル NLW6I271E_mkl.dll または NLW6I271E_nag.dll の格納フォルダーのパスが環境変数 PATH に設定されている必要があります. 詳細は「3.1.1. コマンドウィンドウ」をご参照ください.

また,NAG AD ライブラリ nag_nag_ad_MT.lib および nag_nag_ad_MD.lib が提供されます.

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，C ルーチンに対しては，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．

NAG ライブラリをマルチスレッドアプリケーションで利用する場合の詳細は，ライブラリマニュアルの "CL Interface Multithreading" または "FL Interface Multithreading" ドキュメントをご参照ください．本製品で提供される Intel MKL ライブラリをマルチスレッドアプリケーションで利用する場合の詳細は，以下の Intel 社のウェブサイトをご参照ください．
https://software.intel.com/en-us/articles/intel-math-kernel-library-intel-mkl-using-intel-mkl-with-threaded-applications

NAG AD ライブラリはスレッドセーフではありませんので，NAG AD ライブラリのルーチンの呼び出しを並列で行うことはできません．

本製品で提供されているライブラリは並列化されていません．ただし，MKL ライブラリは OpenMP で並列化されています．スレッド数の設定につきましては「3.1.0 使用するスレッド数の設定」をご参照ください．

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/mkl-windows-developer-guide-reproducibility-conditions 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．ただし，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，ライブラリマニュアルの "How to Use the NAG Library" ドキュメントの「8.1 Bit-wise Reproducibility (BWR)」をご参照ください．

本製品は，MKL 10.3 よりも古いバージョンの MKL 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，以下のデフォルトのインストールフォルダーに本ライブラリがインストールされていることを前提とします.

C:¥Program Files¥NAG¥NL27¥nlw6i271el

もし，このフォルダーが存在しない場合は，システム管理者（本ライブラリをインストールされた方）にお尋ねください. 以降の説明ではこのフォルダーを install_dir として参照します. また，スタートメニューの NAG Library (NLW6I271EL) に以下のライブラリコマンドプロンプトのショートカットが置かれていることを前提とします.

NAG NLW6I271EL Command Prompt

もし，このショートカットが存在しない場合は，システム管理者（本ライブラリをインストールされた方）にお尋ねください. また，本ライブラリのインストール時に作成される他のショートカットも同じ場所に置かれていることを前提とします.

NAG DLL（NLW6I271E_mkl.dll／NLW6I271E_nag.dll）をご利用の場合は，実行時に NAG DLL にアクセスできるように install_dir¥bin フォルダーにパスを通してください. また，適切な Intel ランタイムライブラリにパスが通っていない場合は，install_dir¥rtl¥bin フォルダーにパスを通してください. また，MKL を利用する NAG DLL（NLW6I271E_mkl.dll）をご利用の場合は，install_dir¥mkl¥bin フォルダーにパスを通してください. この時，install_dir¥mkl¥bin は install_dir¥bin の後ろに設定してください. これは BLAS／LAPACK ルーチンのいくつかは，ベンダーバージョンとの問題を避けるために，NAG バージョン（NLW6I271E_mkl.dll に含まれる）を使用する必要があるからです.（「4. ルーチン固有の情報」参照）

NAG DLL へのアクセスをチェックするために，スタートメニューの
NAG Library (NLW6I271EL) にある以下のショートカットから診断プログラム NAG_Library_DLL_info.exe を実行してください.

Check NAG NLW6I271EL DLL Accessibility

この診断プログラムの詳細については，インストールノートの「4.2.2. アクセスチェック」をご参照ください.

3.1.0. 使用するスレッド数の設定

MKL は OpenMP を用いて並列化されています. 実行時に使用するスレッド数を環境変数 OMP_NUM_THREADS に設定してください. 例えば, コマンドウィンドウでは以下のように行います. （なお, 環境変数は Windows のコントロールパネルから通常の方法で設定することもできます. ）

例）

```
set OMP_NUM_THREADS=N
```

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.

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・設定することにより, 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.

例）

```
set OMP_NUM_THREADS=N,P
```

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．

本製品で提供される Intel MKL ライブラリには（OMP_NUM_THREADS 以外にも）MKL 内のスレッドをより細かく制御するための幾つもの環境変数があります．これらの環境変数の詳細につきましては，以下の Intel 社のウェブサイトをご参照ください．
https://software.intel.com/en-us/articles/intel-math-kernel-library-intel-mkl-intel-mkl-100-threading
多くの NAG ルーチンは MKL ルーチンを利用しています．従って，MKL 環境変数は間接的に NAG ライブラリの動作にも影響を与えます．基本的には，MKL 環境変数のデフォルト設定が NAG ライブラリには適しています．従って，これらの MKL 環境変数を明示的に設定しないことをお勧めします．

3.1.1. コマンドウィンドウ

本ライブラリをコマンドウィンドウからご利用になる場合は，環境変数の設定が必要です.（なお，インストール時に環境変数の自動設定を選択された場合は，必要な環境変数はシステム環境変数に既に設定されています.）スタートメニューの NAG Library (NLW6I271EL) にある以下のショートカットがご利用いただけます.

NAG NLW6I271EL Command Prompt

このショートカットは，本ライブラリと本製品で提供される MKL に対して必要な環境変数 INCLUDE, LIB, PATH を正しく設定した上でコマンドプロンプトを開きます. また，バッチファイル nag_example_*.bat が必要とする環境変数 NAG_NLW6I271EL も設定します. このショートカットを利用しない場合は，環境変数の設定を手動で行う必要があります. 環境変数の設定はバッチファイル envvars.bat を用いて行うことができます. このバッチファイルのデフォルトの格納位置を以下に示します.

C:¥Program Files¥NAG¥NL27¥nlw6i271el¥batch¥envvars.bat

その後，以下に示すコマンドの一つでコンパイル／リンクを行ってください.
（ここで，driver.c または driver.f90 がユーザープログラムです.）

```
cl /MD driver.c NLW6I271E_mkl.lib
ifort  /MD driver.f90 NLW6I271E_mkl.lib

cl /MD driver.c NLW6I271E_nag.lib
ifort  /MD driver.f90 NLW6I271E_nag.lib

cl /MT driver.c nag_mkl_MT.lib mkl_intel_lp64.lib mkl_intel_thread.lib
   mkl_core.lib libiomp5md.lib user32.lib
   /link /nodefaultlib:ifconsol.lib /nodefaultlib:ifmodintr.lib
         /nodefaultlib:libifcoremt.lib /nodefaultlib:libifport.lib
         /nodefaultlib:ifwin.lib
ifort  /MT driver.f90 nag_mkl_MT.lib mkl_intel_lp64.lib mkl_intel_thread.lib
   mkl_core.lib libiomp5md.lib user32.lib
```

```
cl /MT driver.c nag_nag_MT.lib user32.lib
   /link /nodefaultlib:ifconsol.lib /nodefaultlib:ifmodintr.lib
         /nodefaultlib:libifcoremt.lib /nodefaultlib:libifport.lib
         /nodefaultlib:ifwin.lib
ifort  /MT driver.f90 nag_nag_MT.lib user32.lib

cl /MD driver.c nag_mkl_MD.lib mkl_intel_lp64.lib mkl_intel_thread.lib
   mkl_core.lib libiomp5md.lib user32.lib
   /link /nodefaultlib:ifconsol.lib /nodefaultlib:ifmodintr.lib
         /nodefaultlib:libifcoremd.lib /nodefaultlib:libifportmd.lib
         /nodefaultlib:ifwin.lib
ifort  /MD driver.f90 nag_mkl_MD.lib mkl_intel_lp64.lib mkl_intel_thread.lib
   mkl_core.lib libiomp5md.lib user32.lib

cl /MD driver.c nag_nag_MD.lib user32.lib
   /link /nodefaultlib:ifconsol.lib /nodefaultlib:ifmodintr.lib
         /nodefaultlib:libifcoremd.lib /nodefaultlib:libifportmd.lib
         /nodefaultlib:ifwin.lib
ifort  /MD driver.f90 nag_nag_MD.lib user32.lib
```

注意：いくつかのコマンドは紙面の都合により二行以上で書かれていますが，実際は一行で打ち込んでください.

注意：ここではMicrosoft Cコンパイラclを用いていますが，Intel Cコンパイラiclをご利用の場合は，上記コマンドのclをiclに置き換えてください．どちらのコンパイラでもコンパイラオプションは同じです.

NAG ADライブラリを利用する場合は，以下のようにNAGライブラリの前にNAG ADライブラリを追加してください.

- ```
  libnag_dcof_MT.lib nag_nag_ad_MT.lib nag_nag_MT.lib user32.lib
  /link /nodefaultlib:ifconsol.lib /nodefaultlib:ifmodintr.lib
  /nodefaultlib:libifcoremt.lib /nodefaultlib:libifport.lib
  /nodefaultlib:ifwin.lib
  ```
  ```

- libnag_dcof_MD.lib nag_nag_ad_MD.lib nag_nag_MD.lib user32.lib
  /link /nodefaultlib:ifconsol.lib /nodefaultlib:ifmodintr.lib
  /nodefaultlib:libifcoremd.lib /nodefaultlib:libifportmd.lib
  /nodefaultlib:ifwin.lib

dcofインターフェースレイヤーライブラリ(libnag_dcof_MT.libとlibnag_dcof_MD.lib)が２つのテープメモリモデル（blob と chunk）用に提供されます. メモリモデルの詳細については, NAG dco/c++ のユーザーガイドをご参照ください. envvars.batはblobバージョンへのアクセス用に環境変数LIBを構成します.chunkバージョンへのアクセスに切り替えるには, envvars.bat（または, これがLIBに設定した値）を編集し, lib¥blobをlib¥chunkに置き換えてください.

コンパイラオプションとリンカーオプション：

/MD

コンパイラランタイムライブラリのマルチスレッド DLL バージョンのインポートライブラリとのリンクを指定するオプションです.

/MT

コンパイラランタイムライブラリのスタティックマルチスレッドバージョンとのリンクを指定するオプションです.

/nodefaultlib:

（ここでは不要な）ランタイムライブラリを無視するようリンカーに指示します.

NLW6I271E_mkl.lib は MKL BLAS／LAPACK を利用する DLL インポートライブラリです. NLW6I271E_nag.libはNAG BLAS／LAPACKを含むDLLインポートライブラリです. これらのライブラリは /MD オプションを付けてコンパイルされています. これらのライブラリを利用する場合には /MDオプションが必要です.

nag_mkl_MT.libはBLAS／LAPACKを含まないスタティックライブラリで, MKLスタティックライブラリとリンクする必要があります. nag_nag_MT.libはNAG BLAS／LAPACKを含むスタティックライブラリです. これらのライブラリは /MT オプションを付けてコンパイルされています. これらのライブラリを利用する場合には /MTオプションが必要です.

nag_mkl_MD.libはBLAS／LAPACKを含まないスタティックライブラリで, MKLスタティックライブラリとリンクする必要があります. nag_nag_MD.libはNAG BLAS／LAPACKを含むスタティックライブラリです. これらのライブラリは /MD オプションを付けてコンパイルされています. これらのライブラリを利用する場合には /MDオプションが必要です.

3.1.2. Microsoft Visual Studio

本セクションの説明はMicrosoft Visual Studio 2019を想定しています.
他のバージョンでは詳細が若干異なるかもしれません.

Visual StudioからのNAGライブラリのご利用には, 適切なオプション設定が必要です.

Visual Studioを起動して, 通常通りにプロジェクトを作成してください.
以降の説明は, プロジェクトが開いていることを前提とします.

本ライブラリは最大最適化されています. そのため Debug モードだとランタイムライブラリについての警告メッセージが表示されますが, 通常これは無視して構いません. Release モードではこの警告メッセージは出力されません. Release モードへの変更は, ツールバーもしくはメニュー「ビルド > 構成マネージャー」から行うことができます.

本ライブラリは 64-bit ライブラリです.「構成マネージャー」の「プラットフォーム」が "x64" に設定されていることを確認してください.

プロジェクトにNAGライブラリを追加する手順を以下に示します.

1. プロジェクトのプロパティページを開いてください.
   プロパティページは次のいずれかの操作で開くことができます.

- ソリューションエクスプローラーでプロジェクトを選択して,
  メニュー「プロジェクト > プロパティ」を選択してください.
- ソリューションエクスプローラーでプロジェクトを右クリックして,
  「プロパティ」を選択してください.
- ソリューションエクスプローラーでプロジェクトを選択して,
  ツールバーの「プロパティ ウィンドウ」ボタンを選択してください.
  「プロパティ」ウィンドウの「プロパティ ページ」アイコンを選択してください.

2. 様々なフォルダーの場所を設定する必要があります.

Microsoft または Intel C または C++ プロジェクトの場合：

左パネルの「構成プロパティ > VC++ ディレクトリ」を選択してください.

- 「インクルードディレクトリ」を選択して,
  install_dir¥include フォルダーを追加してください.
- 「ライブラリディレクトリ」を選択して,
  install_dir¥lib フォルダーおよび install_dir¥rtl¥lib フォルダーを追加してください.（必要に応じて install_dir¥mkl¥lib フォルダーを追加してください.）
  （もしくは, 以下の Fortran プロジェクトの指示に従って,「追加のライブラリディレクトリ」に設定することもできます.）

Intel Fortran プロジェクトの場合：

左パネルの「構成プロパティ > Fortran > 全般」を選択してください.

- 「追加のインクルード・ディレクトリー」を選択して,
  install_dir¥nag_interface_blocks フォルダーを追加してください.

左パネルの「構成プロパティ > リンカー > 全般」を選択してください.

- 「追加のライブラリディレクトリ」を選択して,
  install_dir¥lib フォルダーおよび install_dir¥rtl¥lib フォルダーを追加してください.（必要に応じて install_dir¥mkl¥lib フォルダーを追加してください.）

各フォルダーのデフォルトを以下に示します.

C/C++ プロジェクトの「インクルードディレクトリ」
  C:¥Program Files¥NAG¥NL27¥nlw6i271el¥include
Fortran プロジェクトの「追加のインクルード・ディレクトリー」
  C:¥Program Files¥NAG¥NL27¥nlw6i271el¥nag_interface_blocks
C/C++ または Fortran プロジェクトの「［追加の］ライブラリディレクトリ」
  C:¥Program Files¥NAG¥NL27¥nlw6i271el¥lib
  C:¥Program Files¥NAG¥NL27¥nlw6i271el¥rtl¥lib
  C:¥Program Files¥NAG¥NL27¥nlw6i271el¥mkl¥lib

変更を有効にするために「適用」ボタンをクリックしてください.

3. NAG ライブラリと Intel ランタイムライブラリ(また,必要に応じて MKL ライブラリ)をリンカーオプションに指定します. 左パネルの「構成プロパティ ＞リンカー ＞入力」を選択してください.「追加の依存ファイル」に適切なライブラリファイルを追加してください.【以下の表を参照】

   変更を有効にするために「適用」ボタンをクリックしてください.

4. 適切なランタイムライブラリのオプションを設定する必要があります.
   これは, リンクする NAG ライブラリに合ったものを選択します.【以下の表を参照】

   Microsoft または Intel C または C++ プロジェクトの場合：

   まず, C ソースファイル（例えば, NAG ライブラリの Example プログラムなど）をメニュー「プロジェクト ＞既存項目の追加」からプロジェクトに追加してください（C または C++ソースファイルがプロジェクトに無いと, C++ オプションが表示されません).

   プロパティページの左パネルから「構成プロパティ ＞ C/C++ ＞ コード生成」を選択してください. そして, 右パネルの「ラインタイム ライブラリ」において, nag_nag_MT.lib または nag_mkl_MT.lib をご利用の場合は「マルチスレッド (/MT)」を, その他の NAG ライブラリをご利用の場合は「マルチスレッド DLL (/MD)」を選択してください.

   変更を有効にするために OK ボタンをクリックしてください.

   Intel Fortran プロジェクトの場合：

   プロパティページの左パネルから「構成プロパティ ＞ Fortran ＞ ライブラリー」を選択してください. そして, 右パネルの「ランタイム・ライブラリー」において, nag_nag_MT.lib または nag_mkl_MT.lib をご利用の場合は「マルチスレッド」を, その他の NAG ライブラリをご利用の場合は「マルチスレッド DLL 」を選択してください.

   変更を有効にするために「適用」ボタンをクリックしてください.

NAG ライブラリ	MKL その他ライブラリ	ランタイムライブラリ
NLW6I271E_mkl.lib	user32.lib （AD ライブラリにリンクする場合にのみ必要）	マルチスレッド DLL (/MD)
NLW6I271E_nag.lib	user32.lib （AD ライブラリにリンクする場合にのみ必要）	マルチスレッド DLL (/MD)
nag_mkl_MT.lib	mkl_intel_lp64.lib mkl_intel_thread.lib mkl_core.lib libiomp5md.lib user32.lib	マルチスレッド (/MT)
nag_nag_MT.lib	user32.lib	マルチスレッド (/MT)
nag_mkl_MD.lib	mkl_intel_lp64.lib mkl_intel_thread.lib mkl_core.lib libiomp5md.lib user32.lib	マルチスレッド DLL (/MD)
nag_nag_MD.lib	user32.lib	マルチスレッド DLL (/MD)

5. Microsoft C または C++ プロジェクトにおいて，NAG ライブラリのスタティックバージョン（DLL ではなく nag_mkl_MT.lib または nag_mkl_MD.lib）にリンクする場合，一部のランタイムライブラリを無視するようリンカーに指示する必要があります．再度，プロパティページを開いてください．左パネルの「構成プロパティ > リンカー > 入力」を選択してください．右パネルの「特定の既定のライブラリの無視」に「3.1.1. コマンドウィンドウ」において /nodefaultlib: に指定されているライブラリのリストを追加してください．（/MD と /MT のリストは似ていますが，同一ではありませんので注意してください．）ライブラリ名はセミコロンで区切ります．

   変更を有効にするために OK ボタンをクリックしてください．

以上で，プロジェクトのビルド（コンパイル／リンク）を行うことができます．

Microsoft Development Environment 上でのプログラムの実行は，「デバッグ」メニュー（例えば，「デバックなしで開始 (Ctrl+F5)」など）から行うことができます．実行時には，環境変数 PATH が適切に設定されている必要があります（「3.1.1. コマンドウィンドウ」参照）．

プログラムの実行に入出力リダイレクションが伴う場合は，プロパティページの「構成プロパティ ＞ デバッグ」から「コマンド引数」に適切なコマンドを指定してください．例えば，

＜ input_file ＞ output_file

アプリケーションの作業フォルダー以外で入出力を行う場合は，フルパスもしくは相対パスでファイルを指定する必要があります．作業フォルダーの設定は，プロパティページの「構成プロパティ ＞ デバッグ」から「作業ディレクトリ」で行うことができます．

3.1.3. Fortran モジュールファイルに関する注意

Install_dir¥nag_interface_blocks フォルダーに提供される Fortran モジュールファイル（*.mod）は，Intel コンパイラ（ifort）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，他のコンパイラではご利用いただけません．他のコンパイラでご利用の場合は，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必要ありません．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．）
詳細は「3.2. インターフェースブロック」をご参照ください．

3.1.4. Fortran Builder

本ライブラリの DLL バージョン（以下 NAG DLL と呼ぶ）は，Fortran Builder（NAG Fortran コンパイラ）から以下の２つの方法でご利用いただけます.（なお，本ライブラリのスタティックバージョンは，Fortran Builder（NAG Fortran コンパイラ）からはご利用いただけません.）

Fortran Builder 自動リンク：

ご利用の Fortran Builder が本ライブラリ（NLW6I271EL）を自動リンクできるバージョンの場合，以下の方法で本ライブラリをご利用ください.

1. 「コンソール アプリケーション」プロジェクトを新規作成する.

2. メニューバーから「プロジェクト ＞ プロジェクトの設定」を開く.

3. 「基本設定」タブを開く.

4. 「追加ライブラリ ＞ NAG Fortran Library を利用する」にチェックを入れる.
   （これにより，ビルド時に，NAG インターフェースブロックの格納フォルダーが自動的にインクルードされ，MKL を利用する NAG DLL（NLW6I271E_mkl.dll）が自動的にリンクされます.）

5. 「OK」ボタンを押し，プロジェクトの設定を閉じる.

以上で，MKL を利用する NAG DLL（NLW6I271E_mkl.dll）を利用したプロジェクトをビルド／実行することができます.

MKL を利用しない NAG DLL（NLW6I271E_nag.dll）をご利用になる場合は，手動でリンク設定を行なってください. 設定方法については，次のページをご参照ください.

なお，「Fortran コンパイラ ＞ 実行時診断」タブの「未定義の変数（=undefined)」オプションは，本ライブラリと互換性がありません. もし，このオプションにチェックを入れてビルドすると，コンパイルエラーとなりますのでご注意ください.

Fortran Builder 手動リンク：

1. 「コンソール アプリケーション」プロジェクトを新規作成する.

2. メニューバーから「プロジェクト > プロジェクトの設定」を開く.

3. 「ディレクトリ > インクルード」タブを開く.

4. 「インクルード」に，フォルダー
   install_dir¥nag_interface_blocks_nagfor
   を追加する.
   （パスにスペースが含まれていても，クォテーションで括らないでください.）

5. 「リンク > 基本設定」タブを開く.

6. 「リンクするライブラリ」に，DLL ファイル
   install_dir¥bin¥NLW6I271E_mkl.dll
   もしくは,
   install_dir¥bin¥NLW6I271E_nag.dll
   を追加する.
   （DLL インポートライブラリではなく，DLL 本体を指定してください.）

7. 「OK」ボタンを押し，プロジェクトの設定を閉じる.

以上で NAG DLL を利用したプロジェクトをビルド／実行することができます.

なお,「Fortran コンパイラ > 実行時診断」タブの「未定義の変数（=undefined)」オプションは，本ライブラリと互換性がありません. もし，このオプションにチェックを入れてビルドすると，コンパイルエラーとなりますのでご注意ください.

コマンドウィンドウ：

コマンドウィンドウからご利用になる場合は，「3.1.1. コマンドウィンドウ」と同じく環境変数 PATH が正しく設定されていることを確認してください.

NAG Fortran コンパイラを用いて生成されたインターフェースブロックのモジュールファイル（*.mod）が，install_dir¥nag_interface_blocks_nagfor フォルダーに提供されます．もし，異なるバージョンの NAG Fortran コンパイラでご利用になる場合は，モジュールファイルを再生成する必要があります.
（「3.2. Fortran インターフェースブロック」参照）

NAG Fortran コンパイラでは，DLL インポートライブラリではなく，DLL 本体に直接リンクしなくてはいけません.

以下に示すコマンドでコンパイル／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.）

MKL を利用する NAG DLL を利用する場合：

```
nagfor -ieee=full -I"install_dir¥nag_interface_blocks_nagfor" driver.f90
       "install_dir¥bin¥NLW6I271E_mkl.dll" -o driver.exe
```

MKL を利用しない NAG DLL を利用する場合：

```
nagfor -ieee=full -I"install_dir¥nag_interface_blocks_nagfor" driver.f90
       "install_dir¥bin¥NLW6I271E_nag.dll" -o driver.exe
```

NLW6I271E_mkl.dll または NLW6I271E_nag.dll ファイルはフルパスで指定してください.
また，パスがスペースを含む場合は，クォテーションで括る必要があります.

3.1.5. その他の環境

その他の環境からの本ライブラリのご利用については，以下の追加情報ページをご参照ください.
https://www.nag.com/doc/inun/nl27/w6i1el/supplementary.html

3.2. Fortran インターフェースブロック

NAG ライブラリのインターフェースブロック(引用仕様宣言)は NAG ライブラリの Fortran ルーチンの型と引数を定義します. Fortran プログラムから NAG ライブラリ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, その利用が推奨されます（ただし, 本製品で提供される Example を利用する際には必ず必要です). これを用いることで NAG 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とします.

(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 ライブラリの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（ifort）を用いてコンパイルされた形式（*.mod ファイル）で提供されます.

本ライブラリのコマンドプロンプト（スタートメニューのショートカットとして提供される）を利用する場合, もしくはバッチファイル envvars.bat を実行して環境変数の設定を行った場合は, 環境変数 INCLUDE があらかじめ設定されるため,「3.1.1. コマンドウィンドウ」で示されるコマンドでこれらのモジュールにアクセスすることができます.

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」に記載されている Fortran 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以下のような方法で再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.）

オリジナルのモジュールファイルのバックアップのために，任意の場所に任意の名前で（例えば，nag_interface_blocks_original）フォルダーを作成し，nag_interface_blocks フォルダーの内容物をそのフォルダーにコピーしてください.

そして，nag_interface_blocks フォルダーにおいて，すべての *.f90 ファイルをご利用の Fortran コンパイラでコンパイルしてください．その際，インターフェースブロックには依存関係があるため，コンパイルの順番が重要となります．以下に示す順番でコンパイルを行ってください．（FCOMP をご利用のコンパイラ名で置き換えてください.）

```
FCOMP -c nag_precisions.f90
FCOMP -c nag_a_ib.f90
FCOMP -c nag_blast_ib.f90
FCOMP -c nag_blas_consts.f90
FCOMP -c nag_blas_ib.f90
FCOMP -c nag_c_ib.f90
FCOMP -c nag_d_ib.f90
FCOMP -c nag_e_ib.f90
FCOMP -c nag_f_ib.f90
FCOMP -c nag_g_ib.f90
FCOMP -c nag_h_ib.f90
FCOMP -c nag_lapack_ib.f90
FCOMP -c nag_m_ib.f90
FCOMP -c nag_s_ib.f90
FCOMP -c nag_x_ib.f90
FCOMP -c nag_long_names.f90
FCOMP -c nag_library.f90
```

コンパイルによって生成されるオブジェクトファイルは必要ありません.
モジュールファイル（*.mod ファイル）だけをご利用ください.

NAG AD ルーチンのインターフェースブロックについては，ユーザーが再コンパイルすることはできません．もし必要な場合は，NAG にお問い合わせください（「7. コンタクト情報」参照）.

3.3. Example プログラム

提供される Example 結果は，nag_mkl_MD.lib（MKL BLAS/LAPACK ルーチンを利用する NAG スタティックライブラリ）を用いて，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 または Fortran コンパイラ，異なるコンパイラランタイムライブラリ，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 nag_mkl_MD.lib(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．

また，提供される NAG AD Example 結果は，スタティックライブラリ nag_mkl_MD.lib と nag_nag_ad_MD.lib を用いて算出されています．

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／提供されている 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．

install_dir¥batch フォルダーに３つのバッチファイル
nag_example_DLL.bat，nag_example_static_MT.bat，nag_example_static_MD.bat
が提供されます．

これらのバッチファイルをご利用の際には，C/C++ コンパイラまたは Fortran コンパイラと NAG ライブラリに対して必要な環境変数が設定されていなければなりません．特に，環境変数 NAG_NLW6I271EL に本ライブラリのインストール先（例えば，C:¥Program Files¥NAG¥NL27¥nlw6i271el）が設定されている必要があります．

これらのバッチファイルを用いて Example プログラムを簡単に利用する事ができます．これらのバッチファイルは，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）をカレントフォルダーにコピーして，コンパイル／リンク／実行を行います．

C と Fortran の両方の Example プログラムが提供されます.

ご利用の NAG ライブラリルーチンの名前をバッチの引数に指定してください.
例）
nag_example_DLL e04ucc
nag_example_DLL e04ucf

この例では，ソースファイルとデータファイルとオプションファイル（C の場合は e04ucce.c と e04ucce.d と e04ucce.opt，Fortran の場合は e04ucfe.f90 と e04ucfe.d）をカレントフォルダーにコピーして，コンパイル／リンク／実行を行い，結果ファイル（C の場合は e04ucce.r，Fortran の場合は e04ucfe.r）を生成します.

- nag_example_DLL.bat

  NLW6I271E_nag.dll（NAG BLAS/LAPACK を利用する NAG DLL ライブラリ）
  をリンクします.
  例）
  nag_example_DLL e04ucc
  nag_example_DLL e04ucf

  NLW6I271E_mkl.dll（MKL BLAS/LAPACK を利用する NAG DLL ライブラリ）
  をリンクする場合は -mkl オプションを付けてください.
  例）
  nag_example_DLL -mkl e04ucc
  nag_example_DLL -mkl e04ucf

- nag_example_static_MD.bat

  nag_nag_MD.lib（NAG BLAS/LAPACK を利用する NAG スタティックライブラリ（/MD））
  をリンクします.
  例）
  nag_example_static_MD e04ucc
  nag_example_static_MD e04ucf

nag_mkl_MD.lib（MKL BLAS/LAPACK を利用する NAG スタティックライブラリ（/MD））をリンクする場合は -mkl オプションを付けてください.
例）
nag_example_static_MD -mkl e04ucc
nag_example_static_MD -mkl e04ucf

- nag_example_static_MT.bat

  nag_nag_MT.lib（NAG BLAS/LAPACK を利用する NAG スタティックライブラリ（/MT））をリンクします.
  例）
  nag_example_static_MT e04ucc
  nag_example_static_MT e04ucf

  nag_mkl_MT.lib（MKL BLAS/LAPACK を利用する NAG スタティックライブラリ（/MT））をリンクする場合は -mkl オプションを付けてください.
  例）
  nag_example_static_MT -mkl e04ucc
  nag_example_static_MT -mkl e04ucf

NAG AD ライブラリの Example プログラムを実行する場合は，-ad オプションを追加してください.
例）
nag_example_static_MT -ad s01ba_a1w_hcpp

NAG AD ライブラリの Example のいくつかは，本製品に含まれていない dco.hpp ファイルに依存しています．これらの Example にご興味がある場合は，NAG にお問い合わせください（「7. コンタクト情報」参照）.

dco.hpp ファイルと他の dco/c++ ファイルを既にお持ちの場合は，それらの格納場所を環境変数 INCLUDE に追加するか，もしくは，install_dir¥include フォルダーにファイルをコピーしてください.

Microsoft C/C++ コンパイラの代わりに Intel C/C++ コンパイラを使う場合は，バッチの引数に -icl オプションを追加してください.

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 a00aaf または 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，バッチファイル nag_example_*.bat を引数 a00aaf または 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（「3.3. Example プログラム」参照）．ライブラリルーチン a00aaf または 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，Mark など）を出力します．

または，診断プログラム NAG_Library_DLL_info.exe を利用することもできます．
診断プログラムはその中で a00aaf と a00aac を呼び出し，その結果を出力します．
（インストールノートの「4.2.2. アクセスチェック」参照）

3.5. C データ型

NAG C 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	int	4
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 "NAG CL Interface Introduction" ドキュメントの「3.1.1 NAG data types」をご参照ください.

3.6. Fortran データ型

本製品は, 32-bit 整数ライブラリ（install_dir¥lib フォルダーにある）のみを提供します.

NAG ライブラリとライブラリマニュアルでは, 実数の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 Example プログラムをご参照ください.

本製品では, 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）

これらに加え, ライブラリマニュアルの FL Interface セクションでは, 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. 詳細につきましては,
ライブラリマニュアルの "NAG FL Interface Introduction" ドキュメントの「2.5 Implementation-dependent Information」をご参照ください.

3.7. C/C++ からの NAG Fortran ルーチンの呼び出し

NAG ライブラリの Fortran ルーチンは C, C++（または互換性のある環境）からもご利用いただけます. ユーザーが Fortran 型と C 型の間のマッピングを行うのを支援するために, C の観点からの Fortran インターフェースの説明（C ヘッダーインターフェース）が各 Fortran ルーチンのドキュメントに含まれています. また, C/C++ ヘッダーファイル install_dir¥include¥nag.h が提供されます.

C または C++ から NAG ライブラリの Fortran ルーチンを呼び出す際のアドバイスは, "alt_c_interfaces.html" ドキュメントをご参照ください.（なお, NAG ライブラリの以前の Mark では, このドキュメントは "techdoc.html" と呼ばれていました.）

3.8. LAPACK, BLAS などの C 宣言

NAG C/C++ ヘッダーファイルには, NAG ライブラリに含まれている LAPACK, BLAS および BLAS Technical Forum（BLAST）ルーチンの宣言が含まれています. ユーザーは, 他のライブラリ（例えば, 付属の Intel MKL など）の C インクルードファイルから, これらのルーチンの定義を取得したいと思うかもしれません. このような場合, 異なる C ヘッダー宣言の衝突を避けるため,「3.1. ライブラリのリンク方法」の C または C++ のコンパイルコマンドに以下のコンパイルオプションを追加することで, これらのルーチンの NAG 宣言を無効にすることができます.

```
-DNAG_OMIT_LAPACK_DECLARATION -DNAG_OMIT_BLAS_DECLARATION \
-DNAG_OMIT_BLAST_DECLARATION
```

ただし, F01, F06, F07, F08 の NAG ルーチン名の宣言は残ります.

4. ルーチン固有の情報

本製品の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

チャプター F06, F07, F08, F16 では, BLAS／LAPACK 由来のルーチンに対して NAG スタイルのルーチン名が与えられています. ルーチン名の詳細については, 各チャプターイントロダクションをご参照ください. パフォーマンスの面からは, 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使用してください.

多くの LAPACK ルーチンには, 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ルーチンに問い合わせる "workspace query" メカニズムがあります. NAG が提供する LAPACK と MKL が提供する 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.

MKL に依存するバージョンの NAG ライブラリでは, BLAS／LAPACK ルーチンの呼び出しに関して, MKL が提供する BLAS／LAPACK ルーチンが使われます. ただし, 以下の BLAS／LAPACK ルーチンの呼び出しは, NAG が提供する BLAS／LAPACK ルーチンが使われます.

```
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
dbdsvdx dgesvdx dgesvj  dsbgvd  zgejsv  zgesvdx zgesvj  zhbgvd
```

b. S07 - S21

これらのチャプターの関数の動作は製品毎に異なります.

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.
本製品に固有の値を以下に示します.

```
s07aa[f] (nag[f]_specfun_tan)
    F_1 = 1.0e+13
    F_2 = 1.0e-14
```

```
s10aa[fc] (nag[f]_specfun_tanh)
    E_1 = 1.8715e+1
s10ab[fc] (nag[f]_specfun_sinh)
    E_1 = 7.080e+2
s10ac[fc] (nag[f]_specfun_cosh)
    E_1 = 7.080e+2

s13aa[fc] (nag[f]_specfun_integral_exp)
    x_hi = 7.083e+2
s13ac[fc] (nag[f]_specfun_integral_cos)
    x_hi = 1.0e+16
s13ad[fc] (nag[f]_specfun_integral_sin)
    x_hi = 1.0e+17

s14aa[fc] (nag[f]_specfun_gamma)
    ifail = 1 (NE_REAL_ARG_GT) if x > 1.70e+2
    ifail = 2 (NE_REAL_ARG_LT) if x < -1.70e+2
    ifail = 3 (NE_REAL_ARG_TOO_SMALL) if abs(x) < 2.23e-308
s14ab[fc] (nag[f]_specfun_gamma_log_real)
    ifail = 2 (NE_REAL_ARG_GT) if x > x_big = 2.55e+305

s15ad[fc] (nag[f]_specfun_erfc_real)
    x_hi = 2.65e+1
s15ae[fc] (nag[f]_specfun_erf_real)
    x_hi = 2.65e+1
s15ag[fc] (nag[f]_specfun_erfcx_real)
    ifail = 1 (NW_HI) if x >= 2.53e+307
    ifail = 2 (NW_REAL) if 4.74e+7 <= x < 2.53e+307
    ifail = 3 (NW_NEG) if x < -2.66e+1

s17ac[fc] (nag[f]_specfun_bessel_y0_real)
    ifail = 1 (NE_REAL_ARG_GT) if x > 1.0e+16
s17ad[fc] (nag[f]_specfun_bessel_y1_real)
    ifail = 1 (NE_REAL_ARG_GT) if x > 1.0e+16
    ifail = 3 (NE_REAL_ARG_TOO_SMALL) if 0 < x <= 2.23e-308
```

s17ae[fc] (nag[f]_specfun_bessel_j0_real)
    ifail = 1 (NE_REAL_ARG_GT) if abs(x) > 1.0e+16
s17af[fc] (nag[f]_specfun_bessel_j1_real)
    ifail = 1 (NE_REAL_ARG_GT) if abs(x) > 1.0e+16
s17ag[fc] (nag[f]_specfun_airy_ai_real)
    ifail = 1 (NE_REAL_ARG_GT) if x > 1.038e+2
    ifail = 2 (NE_REAL_ARG_LT) if x < -5.7e+10
s17ah[fc] (nag[f]_specfun_airy_bi_real)
    ifail = 1 (NE_REAL_ARG_GT) if x > 1.041e+2
    ifail = 2 (NE_REAL_ARG_LT) if x < -5.7e+10
s17aj[fc] (nag[f]_specfun_airy_ai_deriv)
    ifail = 1 (NE_REAL_ARG_GT) if x > 1.041e+2
    ifail = 2 (NE_REAL_ARG_LT) if x < -1.9e+9
s17ak[fc] (nag[f]_specfun_airy_bi_deriv)
    ifail = 1 (NE_REAL_ARG_GT) if x > 1.041e+2
    ifail = 2 (NE_REAL_ARG_LT) if x < -1.9e+9
s17dc[fc] (nag[f]_specfun_bessel_y_complex)
    ifail = 2 (NE_OVERFLOW_LIKELY) if abs(z) < 3.92223e-305
    ifail = 4 (NW_SOME_PRECISION_LOSS) if abs(z) or fnu+n-1 > 3.27679e+4
    ifail = 5 (NE_TOTAL_PRECISION_LOSS) if abs(z) or fnu+n-1 > 1.07374e+9
s17de[fc] (nag[f]_specfun_bessel_j_complex)
    ifail = 2 (NE_OVERFLOW_LIKELY) if AIMAG(z) > 7.00921e+2
    ifail = 3 (NW_SOME_PRECISION_LOSS) if abs(z) or fnu+n-1 > 3.27679e+4
    ifail = 4 (NE_TOTAL_PRECISION_LOSS) if abs(z) or fnu+n-1 > 1.07374e+9
s17dg[fc] (nag[f]_specfun_airy_ai_complex)
    ifail = 3 (NW_SOME_PRECISION_LOSS) if abs(z) > 1.02399e+3
    ifail = 4 (NE_TOTAL_PRECISION_LOSS) if abs(z) > 1.04857e+6
s17dh[fc] (nag[f]_specfun_airy_bi_complex)
    ifail = 3 (NW_SOME_PRECISION_LOSS) if abs(z) > 1.02399e+3
    ifail = 4 (NE_TOTAL_PRECISION_LOSS) if abs(z) > 1.04857e+6
s17dl[fc] (nag[f]_specfun_hankel_complex)
    ifail = 2 (NE_OVERFLOW_LIKELY) if abs(z) < 3.92223e-305
    ifail = 4 (NW_SOME_PRECISION_LOSS) if abs(z) or fnu+n-1 > 3.27679e+4
    ifail = 5 (NE_TOTAL_PRECISION_LOSS) if abs(z) or fnu+n-1 > 1.07374e+9

s18ad[fc] (nag[f]_specfun_bessel_k1_real)
    ifail = 2 (NE_REAL_ARG_TOO_SMALL) if 0 < x <= 2.23e-308
s18ae[fc] (nag[f]_specfun_bessel_i0_real)
    ifail = 1 (NE_REAL_ARG_GT) if abs(x) > 7.116e+2
s18af[fc] (nag[f]_specfun_bessel_i1_real)
    ifail = 1 (NE_REAL_ARG_GT) if abs(x) > 7.116e+2
s18dc[fc] (nag[f]_specfun_bessel_k_complex)
    ifail = 2 (NE_OVERFLOW_LIKELY) if abs(z) < 3.92223e-305
    ifail = 4 (NW_SOME_PRECISION_LOSS) if abs(z) or fnu+n-1 > 3.27679e+4
    ifail = 5 (NE_TOTAL_PRECISION_LOSS) if abs(z) or fnu+n-1 > 1.07374e+9
s18de[fc] (nag[f]_specfun_bessel_i_complex)
    ifail = 2 (NE_OVERFLOW_LIKELY) if REAL(z) > 7.00921e+2
    ifail = 3 (NW_SOME_PRECISION_LOSS) if abs(z) or fnu+n-1 > 3.27679e+4
    ifail = 4 (NE_TOTAL_PRECISION_LOSS) if abs(z) or fnu+n-1 > 1.07374e+9

s19aa[fc] (nag[f]_specfun_kelvin_ber)
    ifail = 1 (NE_REAL_ARG_GT) if abs(x) >= 5.04818e+1
s19ab[fc] (nag[f]_specfun_kelvin_bei)
    ifail = 1 (NE_REAL_ARG_GT) if abs(x) >= 5.04818e+1
s19ac[fc] (nag[f]_specfun_kelvin_ker)
    ifail = 1 (NE_REAL_ARG_GT) if x > 9.9726e+2
s19ad[fc] (nag[f]_specfun_kelvin_kei)
    ifail = 1 (NE_REAL_ARG_GT) if x > 9.9726e+2

s21bc[fc] (nag[f]_specfun_ellipint_symm_2)
    ifail = 3 (NE_REAL_ARG_LT) if an argument < 1.583e-205
    ifail = 4 (NE_REAL_ARG_GE) if an argument >= 3.765e+202
s21bd[fc] (nag[f]_specfun_ellipint_symm_3)
    ifail = 3 (NE_REAL_ARG_LT) if an argument < 2.813e-103
    ifail = 4 (NE_REAL_ARG_GT) if an argument >= 1.407e+102

c. X01

数学定数を以下に示します.

```
x01aa[fc] (nag[f]_math_pi)
     = 3.1415926535897932
x01ab[fc] (nag[f]_math_euler)
     = 0.5772156649015328
```

d. x02

マシン定数を以下に示します.

浮動小数点演算の基本的なパラメーター:

```
x02bh[fc] (nag[f]_machine_model_base)
     = 2
x02bj[fc] (nag[f]_machine_model_digits)
     = 53
x02bk[fc] (nag[f]_machine_model_minexp)
     = -1021
x02bl[fc] (nag[f]_machine_model_maxexp)
     = 1024
```

浮動小数点演算の派生的なパラメーター:

```
x02aj[fc] (nag[f]_machine_precision)
     = 1.11022302462516e-16
x02ak[fc] (nag[f]_machine_real_smallest)
     = 2.22507385850721e-308
x02al[fc] (nag[f]_machine_real_largest)
     = 1.79769313486231e+308
x02am[fc] (nag[f]_machine_real_safe)
     = 2.22507385850721e-308
x02an[fc] (nag[f]_machine_complex_safe)
     = 2.22507385850721e-308
```

コンピューター環境のその他のパラメーター：

x02ah[fc] (nag[f]_machine_sinarg_max)
= 1.42724769270596e+45
x02bb[fc] (nag[f]_machine_integer_max)
= 2147483647
x02be[fc] (nag[f]_machine_decimal_digits)
= 15

e. X04

Fortran ルーチン：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.

f. X06

本製品では, 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.

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.
https://www.nag.com/numeric/nl/nagdoc_27.1/

ライブラリマニュアルは HTML5 (HTML／MathML マニュアル) で提供されます.

これらのドキュメントは Web ブラウザでご利用いただけます.

以下のマスター目次ファイルが提供されます.

nagdoc_27.1¥index.html

ライブラリマニュアルをインストールした場合, この目次ファイルはスタートメニューの NAG Mark 27-1 Manual にある以下のショートカットから開くことができます.

NAG Library Manual Mark 27.1 (HTML5)

ドキュメント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.
https://www.nag.com/numeric/nl/nagdoc_27.1/nlhtml/genint/naglibdoc.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 (本ドキュメントの英語版)
- alt_c_interfaces.html - C および C++ から NAG ライブラリの Fortran ルーチンを呼び出すためのアドバイス

ユーザーノート (英語版) は, スタートメニューの NAG Library (NLW6I271EL) にある以下のショートカットから開くことができます.

NAG NLW6I271EL Users' Note

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（NLW6I271EL）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．

https://www.nag-j.co.jp/　（日本）
https://www.nag.com/　　（英国本社）