NAG Library for SMP & Multicore, Mark 24
FSW6I24DDL - License Managed
Microsoft Windows x64, 64-bit integers, Intel Fortran Double Precision

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 for SMP & Multicore, Mark 24: FSW6I24DDL（ライブラリ）を使用される方向けのドキュメントです．本ユーザーノートには NAG Library Manual, Mark 24（ライブラリマニュアル）に含まれない製品毎の情報が含まれます．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あたり，以下のドキュメントを必ずお読みください．

(a) Essential Introduction （ライブラリについての基本的なドキュメント）
(b) Chapter Introduction （チャプター毎のドキュメント）
(c) Routine Document （ルーチン毎のドキュメント）

本ライブラリはマルチスレッド環境でご利用いただけます（スレッドセーフです）．
詳細はライブラリマニュアルの“Thread Safety”ドキュメントをご参照ください．

2. リリース後の最新情報

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/fs24/w6iddl/postrelease.html

3. 一般情報

3.1. ライブラリのリンク方法

本ライブラリはインストールノートに記載されているバージョンの Intel Fortran コンパイラを用いてビルドされています. それ以外のバージョンの Intel Fortran コンパイラでコンパイルされたプログラムから本ライブラリを利用する場合は, 正しい Intel Fortran コンパイラのランタイムライブラリがピックアップされるように, install_dir¥bin フォルダーに提供されている Intel Fortran コンパイラのランタイムライブラリ libifcoremd.dll, libmmd.dll, svml_dispmd.dll の名前を変更する必要があります. これを簡単に行うためのバッチファイル hide_ifort_rtls.bat が同フォルダーに提供されています. このバッチファイルは同フォルダーにある libifcoremd.dll, libmmd.dll, svml_dispmd.dll の名前を変更します. また, 変更した名前を元に戻すためのバッチファイル expose_ifort_rtls.bat が同フォルダーに併せて提供されます.

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が前提となります.

C:¥Program Files¥NAG¥FS24¥fsw6i24ddl

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した方）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, 以下の「スタート」メニューに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が前提となります.

すべてのプログラム | NAG | FS24 |
NAG Library for SMP and Multicore (FSW6I24DDL) |
FSW6I24DDL 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した方）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が前提となります.

NAG DLL（FSW6I24DD.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 フォルダーのパスが環境変数 PATH に設定されている必要があります．また，install_dir¥MKL_intel64_11.0¥bin フォルダーのパスも環境変数 PATH に設定されている必要があります．この時，install_dir¥MKL_intel64_11.0¥bin は install_dir¥bin よりも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 NAG 提供のもの（FSW6I24DD.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）．

コールバック関数として NAG ルーチンに渡すユーザー作成の手続き（被積分関数など）をコンパイルする際には，手続き内のローカル変数が並列環境（NAG ルーチンの並列領域）で安全に使用されるように注意を払う必要があります．特に，ローカル変数は静的に割り当てられてはいけません．ご利用のコンパイラよっては，これを実現するために，コンパイラオプションを付ける必要があります．また，これとは逆にローカル変数が静的に割り当てられる -save などのオプションは使用しないでください．

3.1.1. スレッド数の設定

環境変数 OMP_NUM_THREADS にご利用のスレッド数を設定してください．
例えば，コマンドウィンドウでは以下のように行います．
例）
```
set OMP_NUM_THREADS=N
```

N はご利用のスレッド数です．
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．

一般的に，推奨されるスレッドの最大数はご利用の SMP システムの物理コア数です．

3.1.2. 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いただく場合には環境変数の設定が必要です.（通常，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設定されています.）

以下の「スタート」メニューのショートカットがご利用いただけます.

すべてのプログラム | NAG | FS24 |
NAG Library for SMP and Multicore (FSW6I24DDL) |
FSW6I24DDL Command Prompt

このショートカットは本ライブラリおよび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.

このショートカットを利用しない場合に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¥FS24¥fsw6i24ddl¥batch¥envvars.bat

その後，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.）

本ライブラリのスタティック版をご利用になる場合：

```
ifort /MD /4I8 /Qopenmp driver.f90 FSW6I24DD_static.lib mkl_intel_ilp64.lib
    mkl_intel_thread.lib mkl_core.lib user32.lib
```

本ライブラリの DLL 版をご利用になる場合：

```
ifort /MD /4I8 /Qopenmp driver.f90 FSW6I24DD.lib
```

/4I8 オプションによって，整数型と論理型の基本種別は（デフォルトの 32-bit から）64-bit に変更されますので注意してください.

本ライブラリの DLL 版をご利用になる場合は，MKL ライブラリを明示的に指定する必要はありません．また，どちらのコマンドも /MD オプションが付いていることに注意してください．このコンパイラオプションはコンパイラのランタイムライブラリに関してマルチスレッド DLL 版を使用することをコンパイラに指示します．本ライブラリとの互換性のために，このコンパイラオプションは重要です.

/Qopenmp オプションによって，コンパイラが OpenMP コードをサポートするようになり，リンカーがコンパイラのスレッドライブラリ libiomp5md.lib をリンクするようになります．しかし，Intel Fortran コンパイラの古いバージョンでは異なるスレッドライブラリ libguide がデフォルトで使用されます．もし，古いコンパイラ（ifort 10.1 など）をご利用の場合は，上記のコマンドに libiomp5md.lib を明示的に加える必要があります.

本ライブラリのスタティック版をご利用になる場合：

```
ifort /MD /4I8 /Qopenmp driver.f90 FSW6I24DD_static.lib mkl_intel_ilp64.lib
    mkl_intel_thread.lib mkl_core.lib libiomp5md.lib user32.lib
```

本ライブラリの DLL 版をご利用になる場合：

```
ifort /MD /4I8 /Qopenmp driver.f90 FSW6I24DD.lib libiomp5md.lib
```

Intel Visual Fortran コンパイラの環境変数の設定にもご注意ください.
また，/Qopenmp オプションは /Qauto オプションを含んでいるため，ローカル変数は静的に割り当てられません（「3.1. ライブラリのリンク方法」参照）.
詳細はコンパイラの User's Guide をご参照ください.

3.1.3. MS Visual Studio .NET

本セクションの説明は Visual Studio .NET 2005/2008/2010 および Intel Fortran Compiler 13.0 を想定しています. 他のバージョンでは詳細が異なるかもしれません.

実行時に NAG DLL (FSW6I24DD.dll) にアクセスできるように, NAG DLL の格納フォルダー install_dir¥bin が環境変数 PATH に設定されている必要があります.
また, 実行時に MKL DLL にアクセスできるように, MKL DLL の格納フォルダー install_dir¥MKL_intel64_11.0¥bin が環境変数 PATH に設定されている必要があります.
ただし, パス設定の順番について,
install_dir¥MKL_intel64_11.0¥bin は install_dir¥bin よりも後でなければいけません.

Visual Studio を起動してください. 以下の手順に従って Intel Fortran コンパイラで利用するフォルダーの設定を行ってください. 以下の設定は Intel Fortran コンパイラを使うプロジェクト (Intel Fortran プロジェクト) すべてに適用されます.

1. メニュー「ツール > オプション」をクリックしてください.

2. 「オプション」ウィンドウで「インテル(R) Fortran」(または「インテル(R) Visual Fortran」または「インテル(R) Composer XE」>「インテル(R) Visual Fortran」) をクリックし「コンパイラー」を選択してください. (Visual Studio のバージョンによっては Intel コンパイラのオプションを見るために「すべての設定を表示」をクリックする必要があるかもしれません.)

3. 右側のパネルの「ライブラリー」の右端の '...' ボタンをクリックしてください.

4. 「ディレクトリー・リストの設定」ウィンドウで NAG インポートライブラリの格納フォルダーのパスを追加してください. デフォルトは以下のようになります.

   C:¥Program Files¥NAG¥FS24¥fsw6i24ddl¥lib

5. NAG DLL に対して, MKL インポートライブラリの格納フォルダーのパスを追加する必要はありません. BLAS／LAPACK のシンボルは NAG インポートライブラリ FSW6I24DD.lib からエクスポートされるからです. (もし「ライブラリー」パスに MKL ライブラリフォルダーを追加する場合は,「3.1. ライブラリのリンク方法」で説明されている通り, NAG ライブラリフォルダーの後に追加する必要があります.)

6. 「ディレクトリー･リストの設定」ウィンドウで OK ボタンをクリックしてください.

7. 右側のパネルの「インクルード」の右端の‘...’ボタンをクリックしてください.

8. 「ディレクトリー・リストの設定」ウィンドウで NAG インターフェースブロックの格納フォルダーのパスを追加してください. デフォルトは以下のようになります.

   C:¥Program Files¥NAG¥FS24¥fsw6i24ddl¥nag_interface_blocks

9. 「ディレクトリー･リストの設定」ウィンドウで OK ボタンをクリックしてください.

10. 「オプション」ウィンドウで OK ボタンをクリックしてください.

上記の設定を行うことにより, Intel Fortran プロジェクトでコンパイル／リンクを行う際に, ライブラリおよび NAG インターフェースブロックをフルパスで指定する必要が無くなります.

上記の設定は, 全ての Intel Fortran プロジェクトに適用されます.
下記の設定は, 個々の Intel Fortran プロジェクトに対して行う必要があります.

本ライブラリはフルオプティマイズされています. そのため Debug モードだと C 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 モードへの設定変更はツールバーもしくはメニューの「ビルド > 構成マネージャー」から行うことができます.

以下の方法で NAG ライブラリをプロジェクトに追加することができます.

プロジェクトのプロパティを開いてください.（ソリューションエクスプローラーでグループプロジェクト（一行目）が選択されていないことを確認して, メニューから「プロジェクト > *** のプロパティ」を選択するか, もしくはソリューションエクスプローラーで特定のプロジェクトを右クリックして「プロパティ」を選択します.）

プロパティの左側のパネルの「リンカー ＞ 入力」を選択してください. 右側のパネルの「追加の依存ファイル」に必要に応じて適切なライブラリを追加します. 本ライブラリのスタティック版 FSW6I24DD_static.lib をご利用になる場合は,「追加の依存ファイル」に FSW6I24DD_static.lib mkl_intel_ilp64.lib mkl_intel_thread.lib mkl_core.lib libiomp5md.lib user32.lib を追加してください.
これらのライブラリはそれぞれスペースで区切ることに注意してください. また, FSW6I24DD_static.lib を最初に置く必要があります. 変更を有効にするために OK または「適用」ボタンをクリックしてください. 同様に, 本ライブラリの DLL 版をご利用になる場合は,「追加の依存ファイル」に FSW6I24DD.lib を追加してください.

次に, 正しいランタイムライブラリを指定する必要があります.

プロパティの左側のパネルの「Fortran ＞ ライブラリー」を選択してください. 右側のパネルの「ランタイム・ライブラリー」において「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 OK または「適用」ボタンをクリックしてください.

コンパイラが OpenMP コードをサポートするように, コンパイラオプション /Qopenmp を指定する必要があります. このオプションを有効にするには, プロパティの左側のパネルの「Fortran ＞ 言語」を選択してください. 右側のパネルの「OpenMP 宣言子の処理」において「並列コードの生成 (/Qopenmp)」を選択してください. 変更を有効にするために OK または「適用」ボタンをクリックしてください.

本ライブラリは 64-bit の整数型と論理型を使います. Fortran の整数型と論理型はデフォルトでは 32-bit です. 従って, ご自身のプログラムでは次のように明示的に変数を宣言する必要があります.

```
integer (kind=8) i, j, k
logical (kind=8) p
```

もしくは, 型の基本種別が自動的に変更されるように, プロジェクトのプロパティの「Fortran ＞ データ ＞ 整数変数の既定 KIND」を “8 (/integer_size:64)” に設定してください.

重要：本ライブラリは 64-bit ライブラリです. Visual Studio の「構成マネージャー」の「アクティブソリューションプラットフォーム」が“x64”に設定されていることを確認してください. 利用可能なプラットフォームのリストに“x64”が表示されない場合は,「新規作成...」を選択して“x64”と打ち込むか又は選択してください（この時,「設定のコピー元」は “Win32”としてください). プラットフォームを“x64”に設定しないと, プロジェクトのビルド時にリンクエラーとなります.

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.

プログラムの実行に標準入出力のリダイレクションを伴わない場合は,「デバッグ」メニュー（例えば「デバックなしで開始」など）から, Visual Studio 上でプログラムを実行することができます.

3.1.4. Visual Basic for Applications 7 / Excel

本ライブラリの DLL 版（以下 NAG DLL と呼ぶ）は Excel スプレッドシートでもご利用いただけます. NAG ライブラリルーチンは, Visual Basic for Applications 7.0（VBA7）コードから呼び出すことができます. 本セクションの情報は Excel の 64-bit 版に関するものです.

NAG DLL を Excel から利用する Example が以下のフォルダーに提供されます.

install_dir¥samples¥excel64_examples

install_dir¥samples¥excel64_examples¥linear_algebra¥ xls_demo_64.html ファイルには Excel スプレッドシートから NAG DLL を利用する際のヒントが記載されています.

キーとなる情報：

- Install_dir¥vba7-64_headers フォルダーの flvba764-＜チャプター名＞.bas（例えば flvba764-a.bas）ファイルには VBA7 で利用できる Declare 文がチャプター毎に定義されています. また flvba764-types.bas ファイルには, これらのファイルで利用される定数やユーザー定義型が定義されています. また flvba764-f-blaslapack.bas ファイルにはチャプター F のルーチンが（NAG 名ではなく）BLAS／LAPACK 名で定義されています.

- Declare 文のご利用は, ファイルから必要な部分だけをご自身のモジュールにコピー＆ペーストするか, もしくはファイルをモジュールとして VBA7 プロジェクトにインポートしてください. 場合によっては, 上述の flvba764-types.bas も合わせてインポートする必要があります.

- Fortran の配列は 1 から始まるので, Option Base 1 の設定を推奨します.

- 実際の引数として Variant 型は使用できません. Long, Double, String（および, ごく稀に Single）が必要です. Option Explicit を使用してください.

- Long は Fortran の INTEGER に, Double は Fortran の DOUBLE PRECISION に, Single は Fortran の REAL にそれぞれ対応します.

- Long は Fortran の LOGICAL に対応します. NAGTRUE と NAGFALSE がそれぞれ -1 と 0 に対応します.

- LongPtr はコールバック関数へのポインターに使用されます. 実際の関数へのポインターは VB の AddressOf 演算子を使用して渡されます.

- 構造体 Complex と ComplexSimple は Fortran の COMPLEX*16 と COMPLEX にそれぞれ対応します.

- Fortran の配列引数に対しては, VBA7 配列の最初の要素を指定します.
  例えば, A(1,1) .

- 数式は ByVal 引数に渡されます. その他の引数はデフォルトでは ByRef です. この点が明確になるように, ByRef と ByVal は Declare 文の全体を通して明示的に指定されています.

- Fortran の文字引数に対しては, ２つの VBA 引数が必要となります. ByVal 文字引数と ByVal 文字長引数（Long 型）です. 文字長引数は引数リストの最後に置く必要があります.

以上の情報は Microsoft Office Excel 2010 で検証されています.

3.1.5. Visual Basic .NET

NAG ライブラリルーチンの多くは Visual Basic .NET（VB.NET）から呼び出すことができます. VB.NET から NAG DLL を利用する Example が以下のフォルダーに提供されます.

install_dir¥samples¥vb.net64_examples

これらの Example は Visual Studio 2005 で生成されています. Visual Studio 2008 以降でロードした場合は, ソリューションとプロジェクトファイルは Visual Studio 変換ウィザードでコンバートされます.

キーとなる情報：

- 以下のファイルに VB.NET で利用できる Declare 文が定義されています.

  Install_dir¥vb.net64_headers¥flvbdnet64.vb

- Declare 文のご利用は, ファイルから必要な部分だけをご自身のモジュールにコピー＆ペーストするか, もしくはファイルをモジュールとして VB.NET プロジェクトにインポートしてください.

- Fortran 配列は 1 から始まりますが, VB.NET 配列は 0 から始まります.

- 次の型マッピングが使われます.
  Integer は Fortran の INTEGER に, Double は Fortran の DOUBLE PRECISION に,
  Single は Fortran の REAL にそれぞれ対応します.

- Integer は Fortran の LOGICAL に対応します. NAGTRUE と NAGFALSE がそれぞれ -1 と 0 に対応します.

- 構造体 Complex と ComplexSimple は Fortran の COMPLEX*16 と COMPLEX にそれぞれ対応します.

- 全てのスカラー値は参照渡し（ByRef）です. VB.NET はデフォルトでは値渡し（ByVal）なので, 参照渡し（ByRef）を明示的に指定する必要があります. この点が明確になるように, ByRef と ByVal は Declare 文の全体を通して明示的に指定されています.

- 配列引数には配列名を渡してください. 全ての配列は値渡し（ByVal）です. また宣言には Fortran 側の用途（入力, 出力, 入出力）によって<[In]()>, <Out()>, <[In](),Out()> のいずれかの decoration が付加されています. 具体例として, various_routines_64 Example の G02EEFE() Sub プロシージャをご参照ください.

- VB.NET ではコールバック関数における配列は値渡しされた IntPtr によって表現されます. 具体例として, d02ejf_example のコードをご参照ください.

- VB.NET の配列は行優先です. 一方で Fortran の配列は列優先です. このため Fortran ルーチンが正しく配列を解釈するためには配列の転置が必要です.

- 配列の格納形式が異なるため, Fortran ルーチンの Leading Dimension は VB.NET の配列の２次元目に対応します. 例えば, VB.NET の A(2,3) では Leading Dimension として 4（配列は 0 から始まるため）を渡します.

- Fortran 側で CHARACTER* 型（例えば, CHARACTER*(*) または CHARACTER*1）のスカラー引数が求められる場合は, 文字列を VB.NET の String で値渡ししてください. そして, 引数リストの最後に文字列の長さを Integer で値渡ししてください.

- Fortran 側で CHARACTER* 型の配列引数が求められる場合は, VB.NET の一つの String に全ての配列要素を結合したものを渡してください. そして, 引数リストの最後に配列の一つの要素の長さを Integer で値渡ししてください. 具体例として, various_routine_64 Example の M01CCFE() Sub プロシージャをご参照ください.

- Fortran 側でコールバック関数が求められる場合は, VB.NET で interface の宣言として Delegate function を定義する必要があります. 引数はその Delegate function 型で値渡ししてください. Delegate function の実装を引数として渡す際には, キーワード AddressOf を利用してください. 具体例として, d01bdf_example のコード, または various_routines_64 Example の D01BDFE() Sub プロシージャをご参照ください.

- これらの宣言を VB.NET に認識させるために, VB.NET ソースコードの一番上には次の一行が必要です.
  Imports System.Runtime.InteropServices

以上の情報は Visual Studio 2005, 2008, 2010 で検証されています.

3.1.6. Microsoft C/C++ または Intel C/C++

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 nagmk24.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++ から本ライブラリの DLL 版（以下 NAG DLL と呼ぶ）を利用する Example が以下のフォルダーに提供されます.

install_dir¥samples¥c_examples

および,

install_dir¥samples¥cpp_examples

C または C++ から NAG DLL を呼び出す際のより詳細なアドバイスは, ドキュメント install_dir¥c_headers¥techdoc.html をご参照ください. なお, このドキュメントのショートカットがスタートメニューに提供されます.

すべてのプログラム | NAG | FS24 |
NAG Library for SMP and Multicore (FSW6I24DDL) |
Calling FSW6I24DDL from C & C++

キーとなる情報 :

- 配列のアクセス順序が異なる.
  C は行優先（Row Major), Fortran は列優先（Column Major）である.

- 提供されるヘッダーファイルを利用する.

- Fortran の文字列は二つのパラメーターとして扱われる（文字列と文字列長).

- C から NAG DLL を利用する Example が提供される.

  install_dir¥samples¥c_examples

- C++ から NAG DLL を利用する Example が提供される.

  install_dir¥samples¥cpp_examples

- C プログラムは *.c 拡張子, C++ プログラムは *.cpp 拡張子を用いる.

C プログラムから NAG DLL をご利用になる場合は, NAG DLL インポートライブラリの格納フォルダーのパスが環境変数 LIB に設定されていることが前提となります. 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. （以下のコマンドは Microsoft C コンパイラ（cl）を使用しています.）

```
cl /MD driver.c FSW6I24DD.lib
```

上記のコマンドはヘッダーファイルの格納フォルダーのパスが環境変数 INCLUDE に設定されていることを前提としています. このパスが設定されていない場合は, 以下のようにコンパイル／リンクを行ってください.

```
cl /MD /I"install_dir¥c_headers" driver.c FSW6I24DD.lib
```

NAG DLL の代わりに NAG スタティックライブラリをリンクする場合は, コンパイラのランタイムライブラリの格納フォルダー install_dir¥rtl へのアクセスが必要となります. このフォルダーのパスが環境変数 LIB に設定されていることを前提として, 以下のようにコンパイル／リンクを行ってください.

```
cl /MD /I"install dir¥c_headers" driver.c FSW6I24DD_static.lib
    mkl_intel_ilp64.lib mkl_intel_thread.lib mkl_core.lib libiomp5md.lib
    user32.lib
```

Intel C コンパイラ（icl）でのご利用方法は, 上記の Microsoft C コンパイラ（cl）でのご利用方法と同じです. ただし, 環境変数 LIB に rtl フォルダーのパスを設定する必要はありません.

3.1.7. Microsoft C#

本ライブラリは 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# ヘッダーファイル flcsdnet64.cs が提供されます. このヘッダーファイルから必要な部分だけを自身のプログラムにコピー＆ペーストしてご利用ください.

C# から本ライブラリの DLL 版を利用する Example が install_dir¥samples¥cs_examples フォルダーに提供されます. これらの Example は, コマンドプロンプトから C# コンパイラ csc を用いて以下のように簡単にご利用いただけます.
（ここで driver.cs が任意の Example ソースファイルです.）

```
csc driver.cs
```

更なる情報は以下のウェブページをご参照ください.

http://www.nag.co.uk/numeric/csharpinfo.asp

3.1.8. NAG Fortran Builder

本ライブラリの DLL 版（以下 NAG DLL と呼ぶ）は，NAG Fortran Builder（NAG Fortran コンパイラ）でもご利用いただけます．NAG Fortran Builder のバージョン 5.3 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．

注意：NAG Fortran Builder（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．

コマンドプロンプトからご利用になる場合は，まず「3.1.2. コマンドウィンドウ」と同じく PATH 環境変数が正しく設定されていることを確認してください．
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．）

```
nagfor -i8 -thread_safe -I"install dir¥nag_interface_blocks_nagfor" driver.f90
       "install dir¥bin¥FSW6I24DD.dll" -o driver.exe
```

注意： -i8 オプションは，本ライブラリとの互換性を維持するために，整数型と論理型の基本種別を（デフォルトの 32-bit から）64-bit に変更します．また，-thread_safe オプションは，ローカル変数が静的に割り当てられないようにします（「3.1. ライブラリのリンク方法」参照）．なお，これとは逆にローカル変数が静的に割り当てられるようになる -save オプションは使用しないでください．

統合開発環境（NAG Fortran Builder）からご利用になる場合は，次のように設定を行なってください．

Fortran Builder 6.0 以降をご利用の場合：

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, NAG DLL (FSW6I24DD.dll) が自動的にリンクされます. また, コンパイラオプション -i8 が自動的に追加されます.）

5. 「Fortran コンパイラ > 詳細設定(1)」タブを開く.

6. 「スレッドセーフなコードを生成する (-thread_safe)」にチェックを入れる.

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.

Fortran Builder 5.3.2 以前をご利用の場合：

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
   install_dir¥nag_interface_blocks_nagfor
   を追加する.
   （注意：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
   install_dir¥bin¥FSW6I24DD.dll
   を追加する.
   （注意：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. 「Fortran コンパイラ > 追加オプション」タブを開く.

8. 「追加するコンパイラオプション」にオプション -i8 を追加する.

9. 「Fortran コンパイラ > 詳細設定(1)」タブを開く.

10. 「スレッドセーフなコードを生成する (-thread_safe)」にチェックを入れる.

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.

3.1.9. アクセスチェック

診断プログラム NAG_Fortran_DLL_info.exe を用いて，ご利用のマシン環境から本ライブラリの DLL 版（FSW6I24DD.dll）にアクセスできるかどうかを確認することができます. 診断プログラムは以下の「スタート」メニューのショートカットから実行することができます.

すべてのプログラム | NAG | FS24 |
NAG Library for SMP and Multicore (FSW6I24DDL) |
Check NAG DLL Accessibility for FSW6I24DDL

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「4.2.3. アクセスチェック」をご参照ください.

3.2. インターフェースブロック

NAG 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 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（ifort）を用いてプリ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,「3.1.2. 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「2.1. 動作環境」に記載されてい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, この必要はありません. 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.）

まずは，オリジナルのモジュールファイルのバックアップを取ってください．例えば，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．

```
ifort /4I8 -c nag_precisions.f90
ifort /4I8 -c nag_a_ib.f90
ifort /4I8 -c nag_blast_ib.f90
ifort /4I8 -c nag_blas_consts.f90
ifort /4I8 -c nag_blas_ib.f90
ifort /4I8 -c nag_c_ib.f90
ifort /4I8 -c nag_d_ib.f90
ifort /4I8 -c nag_e_ib.f90
ifort /4I8 -c nag_f_ib.f90
ifort /4I8 -c nag_g_ib.f90
ifort /4I8 -c nag_h_ib.f90
ifort /4I8 -c nag_lapack_ib.f90
ifort /4I8 -c nag_m_ib.f90
ifort /4I8 -c nag_omp_ib.f90
ifort /4I8 -c nag_s_ib.f90
ifort /4I8 -c nag_w_ib.f90
ifort /4I8 -c nag_x_ib.f90
ifort /4I8 -c nag_long_names.f90
ifort /4I8 -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．

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．

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２つのバッチファイル nagsmp_example_static.bat と nagsmp_example_dll.bat が提供されます．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ことができます．

これらのバッチファイルは環境変数 NAG_FSW6I24DDL を参照します．

インストーラーは「スタート」メニューに以下のショートカットを作成します．

すべてのプログラム｜NAG｜FS24｜
NAG Library for SMP and Multicore (FSW6I24DDL)｜
FSW6I24DDL Command Prompt

このショートカットは必要な環境変数（NAG_FSW6I24DDL を含む）を設定した上でコマンドプロンプトを開きます．

このショートカットを利用しない場合には，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場所を以下に示します．

C:¥Program Files¥NAG¥FS24¥fsw6i24ddl¥batch¥envvars.bat

バッチファイル nagsmp_example_static.bat 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．nagsmp_example_static.bat は本ライブラリのスタティック版（FSW6I24DD_static.lib）および MKL のスタティック版をリンクします．

nagsmp_example_static.bat の引数に，ご利用の NAG ライブラリルーチンの名前と OpenMP スレッド数を指定してください．
例）
nagsmp_example_static e04ucf 4

この例では，e04ucfe.f（ソースファイル）と e04ucfe.d（データファイル）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 e04ucfe.r（結果ファイル）を生成します．

同様に nagsmp_example_dll.bat を利用することができます．
例）
nagsmp_example_dll e04ucf 4

nagsmp_example_dll.bat は本ライブラリの DLL インポートライブラリ（FSW6I24DD.lib）および MKL の DLL 版をリンクします．

3.4. Fortran 型と強調斜体文字の解釈

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
COMPLEX (kind=nag_rp) - COMPLEX (単精度複素数)
COMPLEX (kind=nag_wp) - 倍精度複素数 (例えば 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.

一つ重要なものは ***machine precision*** という表現で, これは DOUBLE PRECISION 浮動小数が計算機内で格納されている相対精度を意味します. 例えば 10 進で約 16 桁の実装であれば ***machine precision*** は 1.0D-16 に近い値を持ちます.

machine precision の正確な値はルーチン X02AJF を使って確認できます.
チャプター X02 のその他のルーチンを使うと, オーバーフロー用の閾値や表現可能な最大整数といった実装異存の定数値を求めることができます.
詳細については X02 Chapter Introduction をご参照ください.

brock size という表現はチャプター F07 と F08 で用いられます. これは, ブロックアルゴリズムで用いられるブロックサイズを表すものです. 用意すべき作業エリアの量に影響が及ぶ場合にのみ, この値に留意する必要があります. 関係する Routine Document と Chapter Introduction に記載されているパラメーター WORK と LWORK についてご参照ください.

3.5. NAG ルーチンからの出力

いくつかのルーチンはエラーメッセージやアドバイスメッセージを出力します. 出力装置番号は X04AAF（エラーメッセージの場合）または X04ABF（アドバイスメッセージの場合）で再設定することが可能です. デフォルト値は「4. ルーチン固有の情報」をご参照ください. これらのルーチンはスレッドセーフではありませんので, 一般的にマルチスレッド環境での出力は推奨されません.

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. 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 ライブラリから
Intel Discrete Fourier Transforms Interface（DFTI）ルーチンを呼び出して使います.

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 NAG C06 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 に示されている値で十分です（変更の必要はありません).

b. C09

Intel コンパイラの現行バージョンの制限により, 以下のルーチンは本ライブラリではシリアルです.

C09FAF C09FBF C09FCF C09FDF

c. F06, F07, F08, F16

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サイズが異なる場合がありますので注意してください.

本ライブラリでは, 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.

DGERFS DGGEVX DGGGLM DSBEV DSBEVX DSBTRD ZGEESX ZHBEV
ZHBEVX ZHBTRD ZTRSEN

以下の NAG ルーチンは MKL 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
F07AVF/ZGERFS	F07BDF/DGBTRF	F07BEF/DGBTRS	F07BHF/DGBRFS
F07BRF/ZGBTRF	F07BSF/ZGBTRS	F07BVF/ZGBRFS	F07CHF/DGTRFS
F07CVF/ZGTRFS	F07FDF/DPOTRF	F07FEF/DPOTRS	F07FHF/DPORFS
F07FJF/DPOTRI	F07FRF/ZPOTRF	F07FSF/ZPOTRS	F07FVF/ZPORFS
F07GEF/DPPTRS	F07GHF/DPPRFS	F07GSF/ZPPTRS	F07GVF/ZPPRFS
F07HEF/DPBTRS	F07HHF/DPBRFS	F07HSF/ZPBTRS	F07HVF/ZPBRFS
F07JHF/DPTRFS	F07JVF/ZPTRFS	F07MHF/DSYRFS	F07MVF/ZHERFS
F07NVF/ZSYRFS	F07PHF/DSPRFS	F07PVF/ZHPRFS	F07QVF/ZSPRFS
F07THF/DTRRFS	F07TVF/ZTRRFS	F07UEF/DTPTRS	F07UHF/DTPRFS
F07USF/ZTPTRS	F07UVF/ZTPRFS	F07VEF/DTBTRS	F07VHF/DTBRFS
F07VSF/ZTBTRS	F07VVF/ZTBRFS	F08AEF/DGEQRF	F08AFF/DORGQR
F08AGF/DORMQR	F08ASF/ZGEQRF	F08ATF/ZUNGQR	F08AUF/ZUNMQR
F08FEF/DSYTRD	F08FFF/DORGTR	F08FSF/ZHETRD	F08FTF/ZUNGTR
F08GFF/DOPGTR	F08GTF/ZUPGTR	F08JEF/DSTEQR	F08JJF/DSTEBZ
F08JKF/DSTEIN	F08JSF/ZSTEQR	F08JXF/ZSTEIN	F08KEF/DGEBRD
F08KSF/ZGEBRD	F08MEF/DBDSQR	F08MSF/ZBDSQR	F08NEF/DGEHRD
F08NGF/DORMHR	F08NSF/ZGEHRD	F08PEF/DHSEQR	F08PKF/DHSEIN
F08PSF/ZHSEQR	F08PXF/ZHSEIN	F08TAF/DSPGV	F08TBF/DSPGVX
F08TCF/DSPGVD	F08TNF/ZHPGV	F08TPF/ZHPGVX	F08TQF/ZHPGVD

d. G02

このチャプターで出てくる ACC の値（マシン依存の定数）は 1.0D-13 です.

e. P01

エラー（hard failure）の際, P01ABF は X04AAF で指定される装置番号にエラーメッセージを出力して停止します.

f. 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
```

g. X01

数学定数は以下のとおりです.

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

h. 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   = 9223372036854775807
X02BEF   = 15

i. 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.

j. OpenMP 並列領域からユーザー関数を呼び出すルーチン

本ライブラリでは，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使用されたものと同じ OpenMP ランタイムライブラリ（これは通常，同じコンパイラを意味します）を使用していない場合は，ユーザー関数内で OpenMP プログラム（指示文）を用いるべきではありません. 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.

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/numeric/FL/FSdocumentation.asp

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_fl24¥html¥FRONTMATTER¥manconts.html
nagdoc_fl24¥pdf¥FRONTMATTER¥manconts.pdf
nagdoc_fl24¥pdf¥FRONTMATTER¥manconts.html

また，便利のために，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_fl24¥index.html

各形式の閲覧方法についての更なる詳細は“Online Documentation”ドキュメントをご参照ください.

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 についての詳細は以下の Intel 社のウェブサイトをご参照ください.

http://www.intel.com/software/products/mkl

6. サポート

(a) ご質問等

保守サービスにご加入いただいているお客様は，電子メール（または電話，FAX）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（FSW6I24DDL）および保守 ID を御明記いただきますようお願い致します．受付は平日 9：30～12:00，13:00～17:30 となります．

日本 NAG ヘルプデスク

email: naghelp@nag-j.co.jp
Tel:　03-5542-6311
Fax:　03-5542-6312

(b) NAG のウェブサイト

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）
http://www.nag-gc.com/　（台湾）

7. ユーザーフィードバック

NAG ではユーザー様からのフィードバックをバージョンアップなどに活かして行きたいと考えています. フィードバックにご協力いただける場合は, 下記のコンタクト先にご連絡ください.

コンタクト先情報

日本ニューメリカルアルゴリズムズグループ株式会社
（略称：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

※ 日本ニューメリカルアルゴリズムズグループ株式会社から提供されるサービス内容は（お問い合わせ先など）日本国内ユーザー様向けに独自のものとなっています.