Software and Tools for Computational Engineering

RWTH AACHEN UNIVERSITY

Numerical Software IV

NAG Library

Viktor Mosenkis

Informatik 12:
Software and Tools for Computational Engineering (STCE)

RWTH Aachen University

Contents

Outline

Objective

- Learn how to use NAG Library,
- Compare NAG implementation of BLAS and LAPACK with Intel MKL,
- Shared vs static library,
- Solve nonlinear Modern Family problem using solver for systems of nonlinear equations

Learning Outcomes

- You will understand
  - difference between shared and static library,
  - different implementation of the NAG Library
  - how to use the NAG Library
- You will be able to use NAG Library to solve Modern Family problem with solver for nonlinear equations.

Outline

The Numerical Algorithms Group (NAG)

- was founded in 1970 as co-operatative project out of academia in UK
- Operates as a commercial, non-for-profit organization
  - Funded entirely by customer income

NAG Library is

- a collection of routines deveoted to numerical analysis and statistics
- closed source
- not for free (requires a license)
- available on various platforms (e.g. Linux/Windows)
- accessible from different programming languages C, C++, Fortran, Python etc.

The NAG library provides routines for a wide range of topics in numerical computing such as

- Random Numbers
- Quadrature
- Root-Finding
- Optimization
- Interpolation
- Linear Algebra
- Sparse Linear Algebra
- Correlation & Regression Analysis

The current RWTH site license allows every student of RWTH University to use the NAG Library, on any machine (cluster, private laptop/PC) free of charge.

To obtain the license key, you must "purchase" it through the RWTH Software Shop

- https://www.itc.rwth-aachen.de/cms/IT-Center/Dienste/kompletter-Servicekatalog/Beschaffungsportale/~essj/Software-Portal/

You can download the NAG Library directly from vendor's website

- https://www.nag.co.uk/content/downloads-nag-library
- this presentation uses **NLL6I27DBL** version of NAG Library

The RWTH site license also gives you access to NAG support

- support@nag.co.uk

The NAG Library is implemented in Fortran. Interfaces to other languages are provided. The version of the NAG Library used in this course contain both C and Fortran implemenations of the NAG Library. Moreover the for each Fortran routine a specific C header exists that allows to call the correspoding Fortran routine directly from C.

- The NAG Library is divided into chapters, each devoted to a branch of maths or statistics. Each has a 3-character name and a title, e.g., F03 – Determinants.
- Exceptionally, Chapters H and S have one-character names.
- All routines in the C Library have six-character names, beginning with the characters of the chapter name, e.g.,d01rac (last character stands for C). The corresponding Fortran routine is called d01raf (last character stands for Fortran).
- There are also “long names” that aim to be more descriptive.

The following code shows how to check the implementation details

```
1  #include <nag.h>
2  #include <stdio.h>
3  #include <string.h>
4
5  int main(void) {
6    Integer exit_status = 0;
7    unsigned int sizeofpointer = sizeof(void *);
8    unsigned int sizeofInteger = sizeof(Integer);
9    unsigned int sp, si;
10
11   /* Get the expected sizes of pointers and integers (in bytes) */
12   a00aay(&sp, &si);
13   printf("nag_info_impl_details (a00aac) Example Program Results\n\n");
14   /* Check that the pointer and integer sizes are as expected */
15   if (sp != sizeofpointer || si != sizeofInteger) {
16     printf(" Incorrect value of sizeof(void *) or of sizeof(Integer)\n");
17   exit_status = 1;
18   }
19   nag_info_impl_details();
20   return exit_status;
21 }
```

The implementation used in this course (**NLL6I27DBL**) includes libraries (and associated files) for use with both 32-bit integers and 64-bit integers.

To compile a source file main.cpp you must tell the compiler the location of the desired header files. Assuming that [INSTALL_DIR] contains the installation directory of the NAG Library, a typical compilation command with g++ is:

- 32-bit integers

```
$ g++ -Wall -I[INSTALL_DIR]/lp64/include -c main.cpp
```

- 64-bit integers

```
$ g++ -Wall -I[INSTALL_DIR]/ilp64/include -c main.cpp
```

Static libraries, are locked into a program at compile time. Dynamic, or shared exist as separate files outside of the executable file.

- Static library
  - code is compiled into the final executable file and cannot be modified without a re-compilation
  - library code is already included in the executable file, thus multiple calls to functions can be handled much more quickly than a dynamic library's code which needs to be called from files outside of the executable.

- Shared (dynamic) library
  - can be modified without recompiling the executable
  - program using shared library is much more susceptible to breaking or security issues. E.g. if a dynamic library is exchange by an attacker, the executable file may no longer work as expected
  - multiple running applications can use the same library without the need to keep its own copy in memory

NAG Library is available as both static and shared library.
To link the static version of the library with g++ compiler the following linking command is required:

- 32-bit integers

```
$ g++ -Wall main.o -o main.exe [INSTALL_DIR]/lp64/lib/libnag_nag.a -lm -ldl -lpthread -lstdc++
```

- 64-bit integers

```
$ g++ -Wall main.o -o main.exe [INSTALL_DIR]/ilp64/lib/libnag_nag.a -lm -ldl -lpthread -lstdc++
```

To link the shared version of the library with g++ compiler the following linking command is required:

- 32-bit integers

```
$ g++ -Wall main.o -o main.exe [INSTALL_DIR]/lp64/lib/libnag_nag.so -lm -ldl -L[INSTALL_DIR]/rtl/lib/intel64 -lifcoremt -lpthread -lstdc++
```

- 64-bit integers

```
$ g++ -Wall main.o -o main.exe [INSTALL_DIR]/ilp64/lib/libnag_nag.so -lm -ldl -L[INSTALL_DIR]/rtl/lib/intel64 -lifcoremt -lpthread -lstdc++
```

To run the shared library executable the LD_LIBRARY_PATH environment variable must be contain the following directories:

- 32-bit integers

```
LD_LIBRARY_PATH=[INSTALL_DIR]/lp64/lib/:/opt/NAG/nll6i27dbl/rtl/lib/
    intel64
```

- 64-bit integers

```
LD_LIBRARY_PATH=[INSTALL_DIR]/ilp64/lib/:/opt/NAG/nll6i27dbl/rtl/lib/
    intel64
```

The implementation of the NAG Library used in this course provides static and shareable libraries that use the Intel ® Math Kernel Library for Linux (MKL), a third-party vendor performance library, to provide Basic Linear Algebra Subprograms (BLAS) and Linear Algebra PACKage (LAPACK) routines (except for any routines listed in Section 4).

It also provides static and shareable libraries that use the NAG versions of these routines.

For best performance, NAG recommend to use one of the variants of the NAG Library which is based on the supplied MKL, i.e. libnag_mkl.a or libnag_mkl.so, in preference to using one of the self-contained NAG libraries, libnag_nag.a or libnag_nag.so.

Outline

We state the Modern Family example for model

$$y = f(\mathbf{p}, \mathbf{x}) : R^n \times R^n \to R$$

as minimization of the error function

$$E(\mathbf{p}, X, \mathbf{y}) = \sum_{i=0}^{m-1} (f(\mathbf{p}, \mathbf{x}_i^T) - y_i)^2$$

In this module only nonlinear (in $\mathbf{p}$)

$$y = f(\mathbf{p}, \mathbf{x}) = (\mathbf{p}^T \cdot \mathbf{x})^2$$

model is considered.

Necessary

$$\frac{dE}{d\mathbf{p}}(\mathbf{p}, X, \mathbf{y}) = 2 \cdot \sum_{i=0}^{m-1} \left((f(\mathbf{p}, \mathbf{x}_i^T) - y_i) \cdot \frac{df}{d\mathbf{p}}(\mathbf{p}, \mathbf{x}_i^T) \right) \quad \rightarrow 0$$

Sufficient

$$\mathbf{v}^T \cdot \frac{d^2E}{d\mathbf{p}^2}(\mathbf{p}, X, \mathbf{y}) \cdot \mathbf{v} > 0 \quad \forall \mathbf{v} \neq \mathbf{0} \in R^n$$

where

$$\frac{d^2E}{d\mathbf{p}^2} = 2 \cdot \sum_{i=0}^{m-1} \left(\frac{df}{d\mathbf{p}}(\mathbf{p}, \mathbf{x}_i^T)^T \cdot \frac{df}{d\mathbf{p}}(\mathbf{p}, \mathbf{x}_i^T) + \frac{d^2f}{d\mathbf{p}^2}(\mathbf{p}, \mathbf{x}_i^T) \cdot (f(\mathbf{p}, \mathbf{x}_i^T) - y_i) \right)$$

We present implementations for the solution of the convex unconstrained minimization problem

$$\min_{\mathbf{p} \in R^n} E(\mathbf{p}, X, \mathbf{y})$$

using

- solver for systems of nonlinear equations c05rbc from the NAG Library. The differentiation is performed
  - symbolically
  - approximately (finite differences)

See source code.

Outline

Summary

- Differences between shared and static libraries
- Compile, link and run programms using NAG Library
- Solve the nonlinear Modern Family problem with c05rbc routine from the NAG Library.

Next Steps

- Install the NAG Library and get a vaild license for your implementation.
- Play with sample code.
- Implent the derivative function using AD.
- Replace the c05rbc routine with c05rbf (corresponding Fortran implementation) using the supplied C-headers.
- Use the adjoint version of c05rbf routine (c05rb_a1w_f) to compute the derivatives of the solution with respect to observed values $\mathbf{y}$. Compare the result with finite difference.
- Continue the course to find out more ...