CS 467 Online Capstone Project (Fall 2022)
GRAZING - NAG Library Replacement
Final Report

Ricardo Yanez

November 30, 2022

Abstract

Grazing collisions between heavy nuclei has regained interest as a mechanism to synthesize new elements and exotic nuclei. The recent production of Sodium-39 (^{39}Na), that is, the nucleus with 11 protons and a whopping 28 neutrons, confirms that transfer reactions is an important mechanism for the study of nuclei at the very extremes of stability. Current and planned accelerator facilities are making this possible. The new experimental opportunities has naturally spurred a renewed interest in the theory behind grazing collisions. In particular, the code GRAZING has been used extensively to predict possible outcomes, suitable projectile-target combinations and energies, and in identifying technical innovations that might be needed. The code is written in FORTRAN and uses the NAG Library for the many numerical calculations of the model. The requirement of a commercial license has hindered many research groups from running the code, particularly since funding for licensing is becoming a thing of the past in most academic institutions. This project was conceived to replace the NAG routines in GRAZING for open source or free implementations of the same numerical calculations.

Contents

1 Introduction

The NAG Numerical Library is a collection of routines developed and sold by the Numerical Algorithms Group [1]. It contains numerical solutions for solving a variety of problems, like finding roots of functions, maximum and minimum, curve or surface fitting of data, solving differential equations, etc. The first version of the library was written in ANGOL and FORTRAN and dates from 1971. In 1990 the library was extended to include the C language, and in 1992 the library incorporated LAPACK routines to include numerical linear regression solutions. Currently, the NAG Library supports the C, C++, FORTRAN, Java, Python and .NET languages.

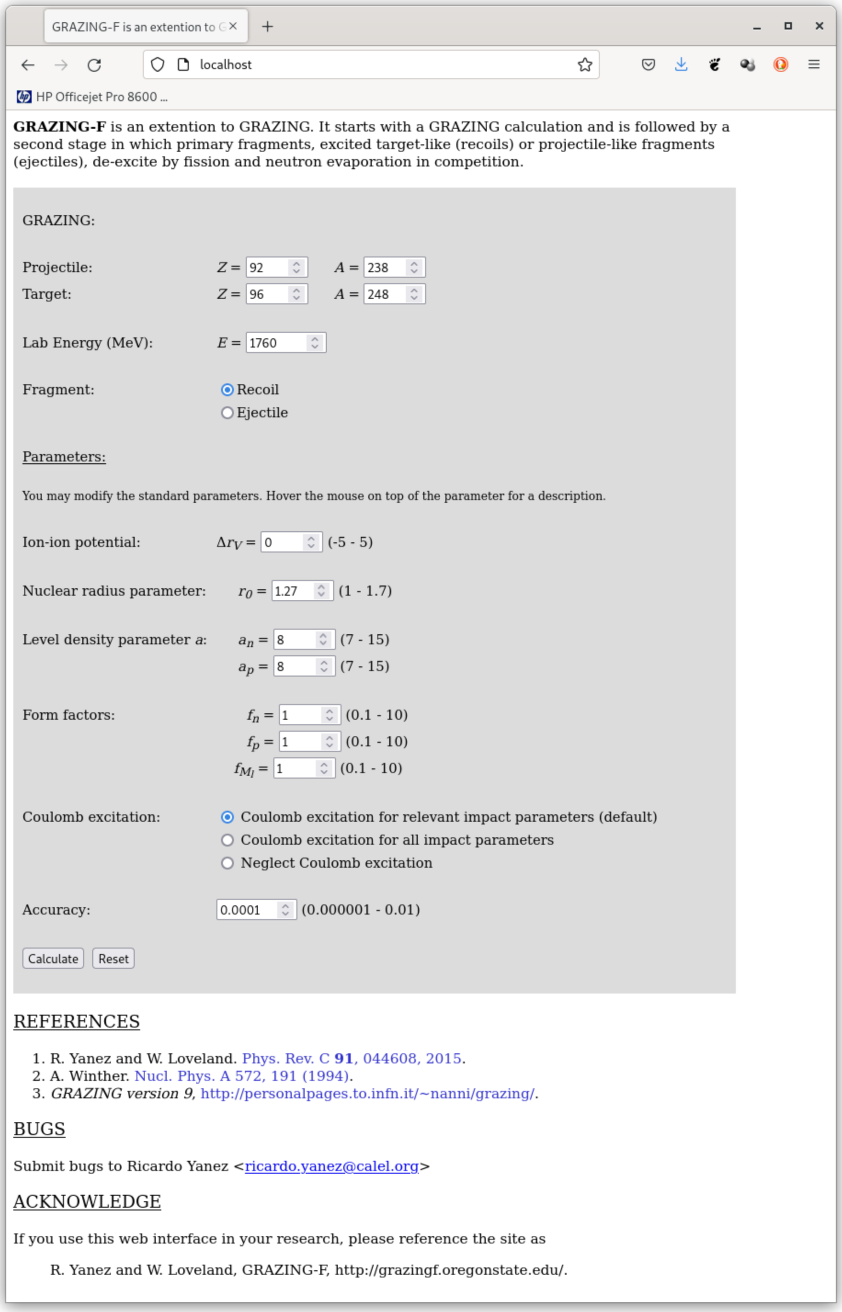
In the past, a NAG license subscription was commonly purchased by most research institutions around the world. This meant many scientific programs were written using NAG routines, being readily available to scientists, researchers and students. Special funding existed for licensing of important software, like Digital VMS, various flavors of UNIX, including Solaris, Matlab and NAG. With the advent of open source, this practice became obsolete. Scientist were instead forced to purchase personal licenses. In this situation, it has become necessary to replace the use of NAG.

2 GRAZING and Extensions

In 1994, Aage Winther put forward a semi-classical model for grazing collisions [1] in which the classical motion of colliding nuclei was combined with the quantum mechanical calculation of collective-state excitation probabilities and nucleon transfers. The code GRAZING [2], written by the author of the model, calculates the classical trajectories and applies the quantum phenomena at the point of closest approach. The code outputs cross-sections of the reaction products, charge and mass distributions, angular and kinetic energy distributions, and even Wilczinski plots. GRAZING is written in FORTRAN and uses the NAG Library [3] for various numerical calculations.

I recently published results of an extension to GRAZING, called GRAZING-F [4], in which the excited nuclei resulting from the model are allowed to decay by fission or neutron emission in competition. The demand by the scientific community for calculations with the new extension prompted the creation of a web application, written in PHP, which was hosted by OSU in a dedicated URL `http://grazingf.oregonstate.edu` (defunct). This site has been down for over a year, after the decommissioning of the Nuclear Chemistry Group of Prof. Walter Loveland. Fig 1 shows the front end of the former web interface. It has not been possible to host the application in another server because of the restrictions of the NAG Library license. It would require the purchase of a service plan to install the license key in another computer. This situation prompted the immediate need to replace the commercial NAG routines used in GRAZING for free or open-source numerical solutions.

[1] `https://www.nag.com/`



GRAZING-F is an extention to GRAZING. It starts with a GRAZING calculation and is followed by a second stage in which primary fragments, excited target-like (recoils) or projectile-like fragments (ejectiles), de-excite by fission and neutron evaporation in competition.

GRAZING:

Projectile:	$Z =$ 92	$A =$ 238
Target:	$Z =$ 96	$A =$ 248

Lab Energy (MeV): $E =$ 1760

Fragment: [✓ Recoil] Ejectile

Parameters:

You may modify the standard parameters. Hover the mouse on top of the parameter for a description.

Ion-ion potential:	$\Delta r_V =$ 0	(-5 - 5)
Nuclear radius parameter:	$r_0 =$ 1.27	(1 - 1.7)
Level density parameter a:	$a_n =$ 8	(7 - 15)
	$a_p =$ 8	(7 - 15)
Form factors:	$f_n =$ 1	(0.1 - 10)
	$f_p =$ 1	(0.1 - 10)
	$f_{M_l} =$ 1	(0.1 - 10)

Coulomb excitation:
[✓ Coulomb excitation for relevant impact parameters (default)]
Coulomb excitation for all impact parameters
Neglect Coulomb excitation

Accuracy: 0.0001 (0.000001 - 0.01)

Calculate Reset

REFERENCES

1. R. Yanez and W. Loveland. Phys. Rev. C **91**, 044608, 2015.
2. A. Winther. Nucl. Phys. A 572, 191 (1994).
3. *GRAZING version 9*, http://personalpages.to.infn.it/~nanni/grazing/.

BUGS

Submit bugs to Ricardo Yanez <ricardo.yanez@calel.org>

ACKNOWLEDGE

If you use this web interface in your research, please reference the site as

R. Yanez and W. Loveland, GRAZING-F, http://grazingf.oregonstate.edu/.

Figure 1: Former front-end of `grazingf.oregonstate.edu`.

3 Approach to replacing NAG routines

The approach I have adopted in this project is to avoid disturbing the original code as much as possible. The only change to the GRAZING code itself was done to define the location of data files via environment variables. This avoids having the user defining the location by editing the source code prior to compilation. Instead the user has to define environment variables used by the local shell. Another important approach is avoid distributing external software. Instead, any external code used is downloaded from official repositories before being compiled. Again, the approach is to disturb original codes as little as possible. Therefore, any necessary change is distributed as a patch to the original source code, which is applied before compilation. Patch files have the extension `.patch`, and are created by the GNU `diff` command,

```
$ diff -Naur source.f.orig source.f > source.f.patch
```

and applied with the GNU `patch` command,

```
$ patch source.f source.f.patch
```

Changes to original code include casting variables from `REAL` to `DOUBLE PRECISION`, changing intrinsic functions to their `DOUBLE PRECISION` counterparts, for example `ABS()` to `DABS()`, and parameters and numbers to `DOUBLE PRECISION`, for example, `1.0E+00` to `1.0D+00`.

The NAG routine calls were not changed in the GRAZING code. Instead, a wrapper subroutine or function is used with the exact same name and input/output parameters as the NAG routine called in GRAZING. The wrapper subroutines and functions create the correct conditions to call the numerical substitutes by passing or declaring new variables, defining external functions and conforming the output to coincide with the expected NAG routine output.

4 Installation

The GRAZING NAG routine replacement project is hosted in GitHub [5]. This site explains the installation procedure. The user has to clone the repository, enter the working directory and execute `make`. The `make` utility will execute the procedure defined in various `Makefile` files. These download the external software from the official repositories, apply the patches and compile the source code to produce the final executable, `grazing_9r`. The second step involves the user defining the appropriate environment variables in the user's shell.

5 Replacement Numerical Solutions

This section reviews the various numerical packages that are used to replace the NAG routines. Most of them are very old, developed by scientist working at national laboratories and universities around the world. The codes are the result of research that is published in research journals or reports, and financed by public funds. Therefore, the codes are licensed in the interest of the general public. Licensing of this software was done before the concept of open source existed. Some of these numerical solutions are integrated into other packages, for example, the GNU Scientific Library, which is written in C. This means many of the original FORTRAN codes have been ported to other languages. Even the NAG routines that are replaced are based on the replacements themselves. For example, the NAG routine D01AMF is based on QUADPACK, and is replaced by QUADPACK. Most of the solutions adopted to replace NAG are in fact original numerical solutions adopted by NAG. In some sense, replacing NAG is returning to the origins of numerical computing.

5.1 AMOS - Bessel Functions

The AMOS [6] package, by Donald Amos, Sandia Nationa Laboratory (1983), is a suite of FORTRAN routines for evaluating Bessel functions of a complex argument and non-negative order. AMOS is found on Netlib.

5.2 Gill-Miller Algorithm

The Gill-Miller Algorithm [7] for integrating unequally spaced data dates from 1972. The NAG routine D01GAF is based on this algorithm. Unfortunately, no original source code was found. I therefore ported to FORTRAN the original published procedure written in ALGOL. The code `fourpt.f`, which stands for "four point", trying to reflect closely the original name of the procedure, has been licensed under GLP in the interest of the general scientific community.

5.3 PCHIP - Piecewise Cubic Hermite Interpolation Package

PCHIP [8], written by F. Fritsch, Lawrence Livermore National Laboratory (1981), is a Fortran package for piecewise cubic Hermite interpolation of data. The PCHIP routines are part of the Slatec Library. PCHIP is found on Netlib.

5.4 Numerical Recipes in FORTRAN 77

"Numerical Recipes in Fortran 77: The Art of Scientific Computing" [9] contains numerical solutions for many problems in scientific computing. Whomever has purchased a copy of the book is entitled to use the machine readable programs for personal use. Distributing a copy is explicitly forbidden. It will be assumed that any interested person in GRAZING has a personal copy of this excellent book.

5.5 QUADPACK

QUADPACK [10] is a set of Fortran routines for integrating one-variable functions. QUADPACK has been integrated into Octave, the GNU Scientific Library and Scientific Python. QUADPACK is licensed as Public Domain and is found on Netlib.

5.6 RKSUITE - a suite of Runde-Kutta codes

RKSUITE [11] is a suite of Runde-Kutta codes for numerical solution of initial value problems of first order ordinary differential equations. RKSUITE is available free of charge to the scientific community. It has no discernible license.

6 The NAG routine calls by GRAZING

In this section, the NAG routines called by GRAZING and their replacements are described in detail and in alphabetical order. The descrition of the NAG routines have been taken from the NAG documentation when possible. GRAZING was written using NAG Mark 18 (1999) and some routines have been superseded by other routines in later versions. NAG maintains a documentation repository from Mark 24. Current version is Mark 28.6 [12].

6.1 C05ADF

The C05ADF routine locates a zero of a continuous function in a given interval by a combination of the methods of linear interpolation, extrapolation and bisection.

```
  SUBROUTINE C05ADF(A,B,EPS,ETA,F,X,IFAIL)
  INTEGER IFAIL
  REAL A,B,EPS,ETA,F,X
  EXTERNAL F
```

`A` is the lower bound, `B` the upper bound, `EPS` and `ETA` are the tolerance and acceptance parameters and `F` is the external function supplied by the user. Upon exit, `X` is the approximation to zero and `IFAIL`=0 unless an error is detected.

The C05ADF routine is replaced by the Numerical Recipes ZBRENT function,

```
  FUNCTION ZBRENT(FUNC,X1,X2,TOL)
  REAL X1,X2,TOL
  EXTERNAL FUNC
```

ZBRENT has been modified,

```
  FUNCTION ZBRENT(FUNC,X1,X2,TOL,IFAIL)
  REAL X1,X2,TOL
  INTEGER IFAIL
  EXTERNAL FUNC
```

where `IFAIL`=0 unless an error is detected. The changes to ZBRENT are distributed as a patch.

6.2 C05AVF

The routine C05AVF attempts to locate an interval containing a simple zero of a continuous function using a binary search. It uses reverse communication for evaluating the function.

```
      SUBROUTINE C05AVF(X,FX,H,BOUNDL,BOUNDU,Y,C,IND,IFAIL)
      INTEGER IND,IFAIL
      REAL X,FX,H,BOUNDL,BOUNDU,Y,C(11)
```

The path to this routine has not been found.

6.3 D01AMF

The D01AMF routine calculates an approximation to the integral of a function $f(x)$ over an infinite or semi-infinite interval [a,b],

$$I = \int_a^b f(x)dx \tag{1}$$

```
      SUBROUTINE D01AMF(F,BOUND,INF,EPSABS,EPSREL,RESULT,ABSERR,W,LW,IW,LIW,IFAIL)
      INTEGER INF,LW,IW(LIW),LIW,IFAIL
      REAL F,BOUND,EPSABS,EPSREL,RESULT,ABSERR,W(LW)
      EXTERNAL F
```

The D01AMF routine is based the QUADPACK routine QAGI [10]. D01AMF is replaced by QUADPACK routine DQAGI, the double precision version of QAGI.

6.4 D01ASF

The routine D01ASF calculates an approximation to the sine or the cosine transform of a function g over [a,∞) for a user-specified value of ω,

$$I = \int_a^\infty g(x)\sin(\omega x)dx \tag{2}$$

$$I = \int_a^\infty g(x)\cos(\omega x)dx \tag{3}$$

```
      SUBROUTINE D01ASF(G,A,OMEGA,KEY,EPSABS,RESULT,ABSERR,
     1 LIMLST,LST,ERLST,RSLST,IERLST,W,LW,IW,LIW,IFAIL)
      INTEGER KEY,LIMLST,LST,IERLST(LIMLST),LW,IW(LIW),LIW,IFAIL
      REAL G,A,OMEGA,EPSABS,RESULT,ABSERR,ERLST(LIMLST),RSLST(LIMLST),W(LW)
      EXTERNAL G
```

The D01ASF routine is based upon the QUADPACK routine QAWFE [10]. D01ASF is replaced by QUADPACK routine QAWFE.

6.5 D01GAF

The routine D01GAF integrates a function which is specified numerically at four or more points, over the whole of its specified range, using third-order finite-difference formulae with error estimates, according to a method due to Gill and Miller.

```
      SUBROUTINE D01GAF(X,Y,N,ANS,ER,IFAIL)
      INTEGER N,IFAIL
      REAL X(N),Y(N),ANS,ER
```

The D01GAF routine is replaced by the routine FOURPT.

```
      SUBROUTINE FOURPT(X,Y,N,ANS,ER,IFAIL)
      INTEGER N,IFAIL
      DOUBLE PRECISION X(N),Y(N),ANS,ER
```

The routine FOURPT is based upon the ANGOL routine published by Gill and Miller [7].

6.6 D02BBF

The routine D02BBF solves an initial value problem for a first-order system of ordinary differential equations using Runge-Kutta methods.

```
      CALL D02BBF(X,XEND,N,Y,TOL,IRELAB,FCN,OUTPUT,W,IFAIL)
      INTEGER N,IFAIL
      REAL X,XEND,Y,TOL,IRELAB,OUTPUT,W
      EXTERNAL FCN
```

The D02BBF routine is based upon RKSUITE [11], which integrates,

$$y' = f(t, y) \tag{4}$$

given $y(t_0) = y_0$, where y is the vector of n solution components and t is the independent variable. D02BBF is replaced by RKSUITE.

6.7 E01BEF

The routine E01BEF computes a monotonicity-preserving piecewise cubic Hermite interpolant to a set of data points.

```
      SUBROUTINE E01BEF(N,X,F,D,IFAIL)
      INTEGER N,IFAIL
      REAL X(N),F(N),D(N)
```

The E01BEF routine is based upon the PCHIP [8] routine PCHIM. E01BEF is replaced by PCHIP routine DPCHIM,

```
      SUBROUTINE DPCHIM(N,X,F,D,INCFD,IERR)
      INTEGER N,IERR
      DOUBLE PRECISION X(N),F(INCFD,N),D(INCFD,N)
```

which is the double precision version of PCHIM.

6.8 E01BFF

The routine E01BFF evaluates a piecewise cubic Hermite interpolant at a set of points.

```
      SUBROUTINE E01BFF(N,X,F,D,M,PX,PF,IFAIL)
      INTEGER N,M,IFAIL
      REAL X(N),F(N),D(N),PX(M),PF(M)
```

The E01BFF routine is based upon the PCHIP [8] routine PCHFE. E01BFF is replaced by PCHIP routine DPCHFE.

```
      SUBROUTINE DPCHFE(N,X,F,D,INCFD,SKIP,NE,XE,FE,IERR)
      INTEGER N,NE,IERR
      DOUBLE PRECISION X(N),F(INCFD,N),D(INCFD,N),XE(NE),FE(NE)
      LOGICAL SKIP
```

which is the double precision version of PCHFE.

6.9 E01BGF

The routine E01BGF evaluates a piecewise cubic Hermite interpolant and its first derivative at a set of points.

```
SUBROUTINE E01BGF(N,X,F,D,M,PX,PF,PD,IFAIL)
INTEGER N,M,IFAIL
REAL X(N),F(N),D(N),PX(M),PF(M),PD(M)
```

The E01BGF routine is based upon the PCHIP [8] routine PCHFD. E01BGF is replaced by PCHIP routine DPCHFD,

```
SUBROUTINE DPCHFD(N,X,F,D,INCFD,SKIP,NE,XE,FE,DE,IERR)
INTEGER N,NE,IERR
DOUBLE PRECISION X(N),F(INCFD,N),D(INCFD,N),XE(NE),FE(NE),DE(NE)
LOGICAL SKIP
```

which is the double precision version of PCHFD.

6.10 S14AAF

The routine S14AAF returns the value of the Gamma function $\Gamma(x)$, via the routine name.

```
REAL FUNCTION S14AAF(X,IFAIL)
INTEGER IFAIL
REAL X
```

The S14AAF routine is substituted with a C wrapper function that calls the GNU C Library function `tgamma()`. The wrapper uses the C library functions for detecting errors defined in `math.h`, `errno.h` and `fenv.h`.

6.11 S14ABF

The S14ABF routine returns a value for the logarithm of the Gamma function, $\ln \Gamma(x)$, via the routine name.

```
REAL FUNCTION S14ABF(X,IFAIL)
INTEGER IFAIL
REAL X
```

The S14ABF routine is substituted with a C wrapper function that calls the GNU C Library function `lgamma()`. The wrapper uses the C library functions for detecting errors defined in `math.h`, `errno.h` and `fenv.h`.

6.12 S15ADF

The routine S15ADF returns the value of the complementary error function, erfc x, via the routine name.

```
REAL FUNCTION S15ADF(X,IFAIL)
INTEGER IFAIL
REAL X
```

The S15ADF routine is substituted with a C wrapper function that calls the GNU C Library function `erfc()`. The wrapper uses the C library functions for detecting errors defined in `math.h`, `errno.h` and `fenv.h`.

6.13 S18AEF

The routine S18AEF returns the value of the modified Bessel Function $I_0(x)$, via the routine name.

```
  REAL FUNCTION S18AEF(X, IFAIL)
  INTEGER IFAIL
  REAL X
```

The S18AEF routine is replaced by the Numerical Recipes [9] function BESSI0,

```
  FUNCTION BESSI0(X)
  REAL X
```

6.14 S18AFF

The routine S18AFF returns the value of the modified Bessel Function $I_1(x)$, via the routine name.

```
  REAL FUNCTION S18AFF(X, IFAIL)
  INTEGER IFAIL
  REAL X
```

The S18AFF routine is replaced by the Numerical Recipes [9] function BESSI1,

```
  FUNCTION BESSI1(X)
  REAL X
```

6.15 S18DEF

The routine S18DEF returns a sequence of values for the modified Bessel functions $\mathrm{I}_{\nu+n}(z)$ for complex z, non-negative ν and $n = 0, 1, ..., N - 1$, with an option for exponential scaling.

```
  SUBROUTINE S18DEF(FNU,Z,N,SCALE,CY,NZ,IFAIL)
  INTEGER N,NZ,IFAIL
  REAL FNU
  COMPLEX Z,CY(N)
  CHARACTER*1 SCALE
```

The S18DEF routine is based upon AMOS [6] routine CBESI. S18DEF is replaced by the AMOS routine ZBESI,

```
  SUBROUTINE ZBESI(ZR,ZI,FNU,KODE,N,CYR,CYI,NZ,IERR)
```

which is the double precision complex version of CBESI.

6.16 X05BAF

The routine X05BAF returns the amount of processor time used since an unspecified previous time, via the routine name.

```
REAL FUNCTION X05BAF()
```

The X05BAF routine is substituted with a C wrapper function that calls the GNU C Library function `clock()`.

References

[1] A. Winther. Nucl. Phys. A 572, 191 (1994).

[2] A. Winther, GRAZING code, http://personalpages.to.infn.it/~nanni/grazing/.

[3] Numerical Algorithms Group, NAG, https://www.nag.com/content/nag-library/.

[4] R. Yanez and W. Loveland. Phys. Rev. C 91, 044608, 2015.

[5] Ricardo Yanez, https://github.com/ricardoyanez/grazing, 2022.

[6] *Algorithm 644: A portable package for Bessel functions of a complex argument and non-negative order*, Amos, D. E. *ACM Trans. Math. Software* **12**, 265–273, 1986.

[7] *An algorithm for the integration of unequally spaced data*, P. E. Gill and G. F. Miller, *Comput. J.* **15**, 80–83, 1972.

[8] *PCHIP final specifications*, Fritsch F. N., Report UCID–30194, Lawrence Livermore National Laboratory, 1982.

[9] *Numerical Recipes in Fortran 77: The Art of Scientific Computing*, W. Press, B. Flannery, S. Teukolsky and W. Vetterling. Cambridge University Press; 2nd edition, ISBN: 978-0521430647, 1992.

[10] *QUADPACK: a subroutine package for automatic integration*, R. Piessens, E. De Doncker-Kapenga and C. W. Überhuber. Springer, ISBN: 3-540-12553-1, 1983.

[11] *RKSUITE: A suite of Runge–Kutta codes for the initial value problems for ODEs*, Brankin R W, Gladwell I and Shampine L F, SoftReport 91–S1, Southern Methodist University, 1991.

[12] NAG Library Documentation, https://www.nag.com/numeric/nl/.