GRAZING - NAG Library Replacement

Ricardo Yanez

December 1, 2022

GRAZING

The GRAZING code [1] is based on the macroscopic model of grazing collisions proposed by Aage Winther [2]. Classical motion of colliding nuclei is combined with excitation of collective states and single-particle transfers along trajectories near and around the point of closest approach. The grazing model describes fairly well the mass and charge distributions, the angular and kinetic energy distributions of reaction products resulting from inelastic reactions between heavy nuclei in which a few nucleons are transfered. The GRAZING code is written in FORTRAN and uses the NAG Library [3] for various numerical calculations.

1 NAG Library

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.

2 Replacing NAG routines

The following approach has been adopted:

1. Avoid disturbing the original code as much as possible.
2. Avoid distributing external software.

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.

External code is downloaded from official repositories before being compiled. Any necessary changes are distributed as patches 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
```

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

Most common 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.

3 Installation

The GRAZING NAG routine replacement project is hosted in GitHub [4]. 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.

4 Numerical Solution Replacements

This section reviews the various numerical packages that are used to replace the NAG routines. Most of the solutions adopted to replace NAG are in fact the original numerical solutions adopted by NAG itself. For example, the NAG routine D01AMF is based on QUADPACK, and is replaced by QUADPACK.

4.1 AMOS - Bessel Functions

The AMOS [5] package is a suite of FORTRAN routines for evaluating Bessel functions of a complex argument and non-negative order. AMOS is found on Netlib.

4.2 Gill-Miller Algorithm

The Gill-Miller Algorithm [6] for integrating unequally spaced data dates from 1972. The NAG routine D01GAF is based on this algorithm. Unfortunately, no original source code was found. The original procedure written in ALGOL is therefore ported to FORTRAN. The code `fourpt.f`, which stands for "four point", trying to reflect closely the original name of the procedure, has been licensed under GPL in the interest of the general scientific community.

4.3 PCHIP - Piecewise Cubic Hermite Interpolation Package

PCHIP [7] is a Fortran package for piecewise cubic Hermite interpolation of data. The PCHIP routines are part of the Slatec Library, and are found on Netlib.

4.4 Numerical Recipes in FORTRAN 77

"Numerical Recipes in Fortran 77: The Art of Scientific Computing" [8] 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.

4.5 QUADPACK

QUADPACK [9] is a set of Fortran routines for integrating one-variable functions. QUADPACK is licensed as Public Domain and is found on Netlib.

4.6 RKSUITE - a suite of Runde-Kutta codes

RKSUITE [10] 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.

5 The NAG routine calls by GRAZING

In this section, the NAG routines called by GRAZING and their replacements are described in some detail and in alphabetical order. The description 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 and up. Current version is Mark 28.6 [11].

5.1 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.

```
1    SUBROUTINE C05ADF(A,B,EPS,ETA,F,X,IFAIL)
2    INTEGER IFAIL
3    REAL A,B,EPS,ETA,F,X
4    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,

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

which uses Brent's method to find the root of function `FUNC` known to lie between `X1` and `X2`. The root is refined until its accuracy is `TOL`.

ZBRENT has been modified,

```
1    FUNCTION ZBRENT(FUNC,X1,X2,TOL,IFAIL)
2    REAL X1,X2,TOL
3    INTEGER IFAIL
4    EXTERNAL FUNC
```

where `IFAIL`=0 unless an error is detected. `IFAIL`=1 when the root is not bracketed. The changes to ZBRENT are distributed as a patch.

5.2 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.

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

The path to this routine has not been found, and has not been replaced. A placeholder is used for compilation purposes.

5.3 D01AMF

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

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

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

The D01AMF routine is based the QUADPACK routine QAGI [9]. D01AMF is replaced by QUADPACK routine DQAGI, the double precision version of QAGI.

5.4 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,

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

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

```
1    SUBROUTINE D01ASF(G,A,OMEGA,KEY,EPSABS,RESULT,ABSERR,
2   1 LIMLST,LST,ERLST,RSLST,IERLST,W,LW,IW,LIW,IFAIL)
3    INTEGER KEY,LIMLST,LST,IERLST(LIMLST),LW,IW(LIW),LIW,IFAIL
4    REAL G,A,OMEGA,EPSABS,RESULT,ABSERR,ERLST(LIMLST),RSLST(LIMLST),W(LW)
5    EXTERNAL G
```

The D01ASF routine is based upon the QUADPACK routine QAWFE [9]. D01ASF is replaced by QUADPACK routine QAWFE.

5.5 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.

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

The D01GAF routine is replaced by the routine FOURPT.

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

The routine FOURPT is based upon the ANGOL routine published by Gill and Miller [6].

5.6 D02BBF

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

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

The D02BBF routine is based upon RKSUITE [10], 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.

5.7 E01BEF

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

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

The E01BEF routine is based upon the PCHIP [7] routine PCHIM. E01BEF is replaced by PCHIP routine DPCHIM,

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

which is the double precision version of PCHIM.

5.8 E01BFF

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

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

The E01BFF routine is based upon the PCHIP [7] routine PCHFE. E01BFF is replaced by PCHIP routine DPCHFE.

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

which is the double precision version of PCHFE.

5.9 E01BGF

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

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

The E01BGF routine is based upon the PCHIP [7] routine PCHFD. E01BGF is replaced by PCHIP routine DPCHFD,

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

which is the double precision version of PCHFD.

5.10 S14AAF

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

```
1   REAL FUNCTION S14AAF(X,IFAIL)
2   INTEGER IFAIL
3   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`.

5.11 S14ABF

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

```
1   REAL FUNCTION S14ABF(X,IFAIL)
2   INTEGER IFAIL
3   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`.

5.12 S15ADF

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

```
1   REAL FUNCTION S15ADF(X,IFAIL)
2   INTEGER IFAIL
3   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`.

5.13 S18AEF

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

```
1   REAL FUNCTION S18AEF(X, IFAIL)
2   INTEGER IFAIL
3   REAL X
```

The S18AEF routine is replaced by the Numerical Recipes [8] function BESSI0,

```
1   FUNCTION BESSI0(X)
2   REAL X
```

5.14 S18AFF

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

```
1   REAL FUNCTION S18AFF(X, IFAIL)
2   INTEGER IFAIL
3   REAL X
```

The S18AFF routine is replaced by the Numerical Recipes [8] function BESSI1,

```
1   FUNCTION BESSI1(X)
2   REAL X
```

5.15 S18DEF

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

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

The S18DEF routine is based upon AMOS [5] routine CBESI. S18DEF is replaced by the AMOS routine ZBESI,

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

which is the double precision complex version of CBESI.

5.16 X05BAF

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

```
1  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, GRAZING code, http://personalpages.to.infn.it/~nanni/grazing/.

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

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

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

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

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

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

[8] *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.

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

[10] *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.

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