1.0 MATH77 and *mathc90*

This manual and the software described herein constitute the Math à la Carte MATH77 and *mathc90* libraries. MATH77 is a library of Fortran 77 subprograms implementing algorithms useful in numerical computation. MATH77 contains over 550 user-callable entries.

mathc90 is an ANSI C language version of most of the MATH77 library. For usage of *mathc90* see Appendices C and D. The remainder of this introductory chapter applies primarily to MATH77 (Fortran).

The Table of Contents provides a directory of the library by topics. The Index lists each user-callable subprogram name, with its argument list, and a reference to the chapter in which it is described. Appendix A lists for each subprogram name all of the program files from MATH77 needed for that subprogram. Appendix B lists all SUBROUTINE, FUNCTION, ENTRY, and COMMON names used in the library. This will be useful if one wishes to avoid using the same names.

A set of demonstration drivers that illustrate the use of library subprograms accompanies the library. These can be used to test that the library is properly installed on a new system. An individual driver may be useful to a user as a starting point for using a library subprogram. Listings of about half the demonstration drivers are included in this manual.

These libraries were developed at the Jet Propulsion Laboratory from 1977–1997 and have been licensed to Math à la Carte by the California Institute of Technology.

1.1 Purpose and Scope

The purpose of high-quality mathematical subprogram libraries is to make scientific and engineering computing more reliable and economical, and to reduce the length of time from problem conception until a solution is obtained. The MATH77 and *mathc90* libraries are particularly useful in that there are no language portability difficulties to inhibit transport of the libraries to all Fortran 77 and ANSI C environments.

Computers ranging from microcomputers to supercomputers are used for scientific and engineering computing. Most of these systems support both ANSI Fortran 77 and ANSI C, and thus allow use of the MATH77 and/or *mathc90* libraries. For persons programming in other languages, most systems provide support for making calls to Fortran 77 or ANSI C libraries.

The Fortran 77 language was initially issued as an ANSI standard in 1978, [1], and was reaffirmed by ANSI in 1988. It was also a U.S. Federal standard (FIPS PUB 69, Sept. 1980) and an international (ISO) standard. The ANSI X3J3 committee then developed Fortran 90, [2], and Fortran 95, [3], which became the current Fortran standard. Fortran 90 was designed to include all of Fortran 77, although this is no longer true of Fortran 95. There are a very few programs in MATH77 that do not satisfy the Fortran 95 standard, but these should still compile on such compilers. Although the later versions of Fortran are a substantial improvement over Fortran 77, we continue to use Fortran 77, as it is only in this language that we have a way to generate the C library automatically.

The first ANSI standard for C was issued in 1989 [4], and ANSI C is also a U.S. Federal standard (FIPS PUB 160, March 1991).

The MATH77 library of mathematical subprograms has the following attributes:

1. Most of the subprograms in MATH77 do not require any modifications to function as described in this document on any computer system supporting the full Fortran 77 language. The only program file requiring attention when moving the library to different machines is AMACH. This file contains machine-dependent constants that are accessed by other library subprograms, and can also be accessed by user programs, by referencing D1MACH, R1MACH, and I1MACH. See Chapter 19.1 for instructions on converting the file AMACH to different systems.
2. All subprograms in MATH77 are coded in conformity to the Fortran 77 standard. This property has been checked by use of a processor for standards checking and by use of the standards checking option on various compilers.

1.2 Access to the MATH77 and *mathc90* Libraries

Access to the libraries is via the URL, `http://mathalacarte.com`. One can freely browse what is available, but if you wish to download anything you must register.

Items available include mangled (*i.e.* compiler readable only) and clean source code for the library codes, clean source for the demonstration drivers, as well as the users' manual suitable for viewing or printing with a PDF, PostScript, or dvi processor.

©1997 Calif. Inst. of Technology, 2010 Math à la Carte, Inc.

1.2.a Files containing the users' manual

Individual chapters and appendices of the users' manual are processed under Linux using teTEX.

Special processors can be used on the `.pdf`, or `.ps`, `.dvi` files to view the manual as it appears on the printed page. There are also processors that convert `.dvi` files to files that can be printed (such as `.ps` files); the `.ps` and `.pdf` files can usually be printed directly. Most machines have public domain versions of the processors mentioned here.

1.3 Conventions Followed in the Code and Documentation

A number of conventions are followed in the library code and its descriptions. All of the library subprograms, demonstration drivers, and test drivers exist as ANSI Fortran 77 source code.

Subprograms that produce printed output use the PRINT or WRITE(*,...) statement to write to the standard system output unit. Exceptions are the message writing subroutines of Chapter 19.3 which can write to arbitrary user-specified Fortran I/O units. Error message writing is handled through subroutines described in Chapters 19.2. and 19.3.

For most library subprograms we follow the convention introduced in the BLAS [5] and LINPACK [6], by which the initial letter of the name of a SUBROUTINE is C, D, I, S, or Z, to indicate the principal type of data with which the subprogram is concerned, and the initial letter of the name of a FUNCTION is C, D, I, or S, to indicate the type of the returned result. These letters denote, respectively, COMPLEX (or CHARACTER), DOUBLE PRECISION, INTEGER, REAL, or double-precision complex. The Fortran 77 standard does not directly support a double-precision complex type, so subprograms oriented toward this type of data use pairs of DOUBLE PRECISION numbers. This representation is compatible with the representation used by most compilers that extend the Fortran 77 standard to provide a double precision complex data type, and is compatible with the representation specified by the later Fortran standards.

When a DOUBLE PRECISION or COMPLEX valued FUNCTION from this library is used, it is essential to have the FUNCTION name appear in a DOUBLE PRECISION or COMPLEX type statement.

Each subprogram description consists of six sections:

A. Purpose
B. Usage
C. Examples and Remarks
D. Functional Description
E. Error Procedures and Restrictions
F. Supporting Information

The contents of each of these sections and special notational conventions used in the descriptions are specified below under these six headings.

A. Purpose

A brief statement of the area of application of the subprogram is given here.

B. Usage

A detailed explanation of how to use the subprogram is given in this section. Typical subsections are as follows:

B.1 Program Prototype

Specification statements, variables that must be initialized, calling sequences, and any other statements likely to be required are given here. It should not be difficult to use the subprogram if one follows this subsection line-by-line.

In giving dimension information, a statement of the form

REAL A(IDIMA, $\geq$ N)**, B**($\geq 5 \times$ J + 20)

is used to indicate that A() is a real two-dimensional array with a first dimension that must equal IDIMA and a second dimension that can be assigned any value greater than or equal to the value assigned to N, and that B() is a real one dimensional array with dimension $\geq 5 \times$ J+20.

The calling sequence for any entry is always enclosed in a box. For FUNCTION subprograms, the calling sequence is always given in the form Y = FNAME(...). The reader should recall that Fortran syntax also permits a FUNCTION name to be used directly in an expression, as for instance Y = 2.0 * FNAME(...) + SQRT(X).

B.2 Argument Definitions

A detailed explanation of the parameters in the calling sequence is given here. Any parameter that is an array name is followed by "()" to call attention to this fact. The intent attributes *in*, *out*, and *inout*, that are described in the Fortran 90 standard [2], are listed for each subprogram argument. We also use intent attributes *work* or *scratch*, not part of the Fortran 90 standard, to describe parameters for which the using program must provide space, but need not provide initial values, and in which no meaningful results are returned.

B.3 Modifications

One or more **Modifications** subsections may be present to describe such things as DOUBLE PRECISION versions of a subprogram or to describe significantly distinct options in the usage of a subprogram.

C. Examples and Remarks

This section discusses a sample ("demo") program, with output, illustrating the use of the subprogram. Any listings and actual output are at the end of the chapter.

In designing the demonstration programs, the example problems have purposely been kept simple. The reader should keep in mind that the subprograms described frequently have features and modes of usage that are not illustrated in the demonstration program. Thus the reader is encouraged to read the entire subprogram description and not judge the range of applicability of a subprogram on the basis of the demonstration programs alone.

This section may also contain remarks that will help one in using the subprogram.

D. Functional Description

This section describes what the subprogram does. It also gives information on the methods used, and, if appropriate and available, gives timing, accuracy data and references.

E. Error Procedures and Restrictions

Information on how errors are treated, and any pertinent information concerning restrictions in the use of the subprogram are given here. Some subprograms return a flag indicating the success or failure mode. Some subprograms call the error message processing subroutines described in Chapters 19.2 and 19.3. These subroutines provide a means for the user to alter the action of error message processing.

F. Supporting Information

This section gives names of all program files needed in order to use the described subprograms. Those responsible for the development of the subprograms are also identified.

References

1. American National Standards Institute, Inc., New York, **American National Standard Programming Language FORTRAN**, ANSI X3.9–1978, (1978).

2. American National Standards Institute, Inc., 11 West 42nd Street, New York, NY 10036, **American National Standard Programming Language Fortran 90**, X3.198–1991, (May 1991).

3. American National Standards Institute, Inc., 11 West 42nd Street, New York, NY 10036, **Information technology – Programming languages – Fortran - Part 1: Base language**, ISO/IEC 1539-1:1997, (1997).

4. American National Standards Institute, Inc., New York, **American National Standard Programming Language — C**, X3.159–1989, (Dec. 1989).

5. C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, *Basic Linear Algebra Subprograms for Fortran usage*, **ACM Trans. on Math. Software 5**, 3 (Sept. 1979) 308–323.

6. J. J. Dongarra, C. B. Moler, J. R. Bunch, and G. W. Stewart, **LINPACK Users' Guide**, Society for Industrial and Applied Mathematics, Philadelphia (1979) 320 pages.