

MATHEMATICAL SOFTWARE AT THE NRAO

FRED SCHWAB

July 24, 1986

Below, I itemize and give brief descriptions of the major mathematical software packages that are available on the NRAO's VAX and Convex computers (I also include one software collection that is especially designed for use on personal computers):

- The IMSL Library,
- The NAG Library,
- The *Numerical Recipes* Collection,
- LINPACK,
- EISPACK,
- MINPACK,
- The TOEPLITZ Package,
- The ACM Algorithm Collection (CACM),
- LLSQ,
- DE/STEP, INTRP,
- de Boor's B-Spline Package,
- The CalTech Graphics Package,
- CalComp Plotter Software,
- MACSYMA.

With two exceptions (CACM and MACSYMA), each of the packages that I've listed is a collection of Fortran language subroutines. The *Numerical Recipes* Collection comes in both Fortran and Pascal versions and is especially designed for use on PC's. Three of the packages are proprietary commercial products: the IMSL Library is licensed for use only on the Charlottesville VAX and the Convex, and the NAG Library is licensed for use only on the Convex. MACSYMA is licensed for use on any of the NRAO's VAXes, but at present it has been installed only on the Charlottesville VAX.

MACSYMA differs from each of the other packages in that it is intended primarily for symbolic and algebraic computation (e.g., for formula manipulation) rather than for numerical computation. The ACM collection differs from the other subroutine packages in that it is simply a collection of programs and subroutines written in various computer languages (Algol, Fortran, PL/I, etc.) and lacking a coherent design philosophy; it consists of the algorithms that have been published over the years in two of the journals of the Association for Computing Machinery. The CalTech Graphics Package consists of two sub-packages for generating graphical displays and of one general purpose interactive plotting program. The software for the CalComp pen plotter is used only on the VAX in Charlottesville.

Here I tell where each package is kept, how one may link one's own applications code with it, and where to find full documentation. Usually a user will need to read the primary documentation, though within some of the packages each subroutine is adequately described in comment lines. Users, of course, are welcome to contact me for assistance.

I plan to revise and re-issue this document from time to time. Please let me know of any inaccuracies or omissions, and tell me of anything that you feel should be added to the collection. I am aware that the primary documentation sources are not readily enough available, especially to NRAO staff outside Charlottesville—I'll consult with the Library Committee and the Computer Division managers to try to remedy this situation.

The IMSL Library. The IMSL Library comprises over five hundred Fortran subroutines for mathematical and statistical applications. At the NRAO this library is installed and licensed¹ for use only on the Charlottesville VAX and on the Convex. We pay a subscription fee of about \$2500/year for the VAX installation; we purchased the Convex version outright, for a few times that amount, since IMSL does not provide maintenance support for Convex computers.

The IMSL Library documentation is contained in a four volume reference manual. Copies of the manual are available in the VLA library, the Green Bank remote terminal room, the first floor of the Edgemont Road building in Charlottesville, and in my office in Charlottesville. The provided capabilities can be divided into the following areas:

- Mathematical Applications
 - differential equations; quadrature; differentiation
 - eigensystem analysis
 - interpolation; approximation; smoothing
 - linear algebraic equations
 - vector/matrix arithmetic
 - nonlinear equations
 - optimization
 - linear programming
 - transforms
- Statistical Applications
 - basic statistics
 - regression analysis
 - analysis of variance
 - nonparametric statistics
 - time series; forecasting; economics
 - observation structure; multivariate statistics
 - categorized data analysis
 - sampling
- General Applications
 - mathematical and statistical special functions
 - generation and testing of random numbers
 - utility functions (printer plots, etc.).

I do not keep the IMSL source code on disk, only the object module libraries. (But I can, on request, retrieve source code from magnetic tape.) Since many of the IMSL subroutines are available in both single-precision and double-precision versions, there are two versions of the library. Subroutines that are available in only one precision—i.e., only in single- or only in double-precision—are kept (duplicated) in both versions of the library.

The VAX versions of the library are in `cvax::dua0:[nrao.math]imslibs.olb` and `cvax::dua0:[nrao.math]imslibd.olb`. If you include in your `login.com` file lines reading

```
    $ ass imsls dua0:[nrao.math]imslibs.olb
and $ ass imsl d dua0:[nrao.math]imslibd.olb,
```

¹Restrictions noted in the license agreement read, in part: “Written arrangements with IMSL must be made if application programs containing embedded object form IMSL Library routines are to be distributed outside the subscribing organization. However, when IMSL Library routines are utilized in research work and resulting publications require, in the opinion of the author, the reproduction of listings of pertinent Library source code in the research publication, IMSL requests reference and grants permission. Where only the program documentation would be sufficient, IMSL encourages that mode of usage. When application programs, developed as a by-product of research work, contain embedded Library routines, these application programs may be taken, as is, by the authors on their departure from the research institution for internal usage in their new locations. IMSL’s only restriction is that any embedded Library routines be used only as originally incorporated in the application program and not be removed for usage in any other software development.”

then you can link with whichever version of the library is required by typing either `link myprog,imsls/lib` or `link myprog,imsld/lib`, depending on whether you need single-precision or double-

The Convex versions of the library are in

```
/nrao/math/imsls.a and /nrao/math/imsld.a .
```

To compile a program and load any required IMSL subroutines from, say, the double-precision library, you can type, for example, `fc myprog.f /nrao/math/imsld.a`.

Users should be cautioned that the Convex version of the library was purchased outright and thus will remain stable (unless we purchase another version), whereas the VAX version, for which we pay a yearly subscription fee, is updated annually. Thus the Convex and VAX versions may someday differ substantially. At present, their functionalities should correspond exactly (and, in fact, the modifications to the VAX version over the past several years have been minor, consisting mainly of “bug fixes”).

The NAG Library. The NAG Library, produced by the Numerical Algorithms Group, Inc., is a collection of Fortran subroutines that is similar in scope to the IMSL Library. (NAG formerly stood for Nottingham Algorithms Group; the present-day commercial enterprise is a university spin-off.) NAG has produced special implementations of its library that are tailored to vector processors such as the Convex C-1. The NAG Library is installed on our Convex computer but not on any of our VAXes. The University of Virginia pays the subscription fee, which is about equal to the IMSL subscription fee.

One set of NAG manuals has been ordered for NRAO (I haven’t decided where they should be kept); we will need to order additional copies for the other NRAO sites. We have only the double-precision version of the library. It is kept in `/usr/lib/libnag.a`. To compile your program and link with NAG, type, for example,

```
fc myprog.f /usr/lib/libnag.a .
```

The NAG Library has essentially complete overlap with the IMSL Library, and I feel that it provides significantly more capabilities and greater flexibility than IMSL. In certain areas of overlap, it provides much greater flexibility.² IMSL has been stagnant for the past several years, while the NAG Library has been expanding. For NRAO sites which do not already have either NAG or IMSL, I would recommend purchase of the NAG Library. This is especially true in the case of the VLA VAXes, since many potential users at the VLA are familiar with the NAG Library from having used it in Europe, where its use is more widespread than IMSL. For Charlottesville, if the budget were to permit, I would recommend purchase of NAG for the VAX and recommend that users incorporate NAG rather than IMSL routines in new applications.

[1] R. Piessens, E. de Doncker-Kapenga, C. W. Überhuber, and D. K. Kahaner, *QUADPACK: A Subroutine Package for Automatic Integration*, Springer Series in Computational Mathematics 1, Springer-Verlag, Berlin, 1983; available in the Charlottesville library of the NRAO, call number QA 299.3.Q36.

The Numerical Recipes Collection. Reference [1], a cookbook of numerical algorithms by Press *et al.*, was recently acquired by the Charlottesville and VLA libraries of

²For example, all of the numerical quadrature routines from QUADPACK [1] are included in NAG—QUADPACK has twelve automatic (i.e., adaptive) routines and six non-automatic ones—whereas IMSL provides only a few quadrature subroutines. IMSL’s main numerical integration routine, DCADRE, is highly sophisticated and very reliable; it is handy for one-shot applications, but in intensive work one can often achieve much greater efficiency by selecting an appropriate special-purpose routine from QUADPACK.

the NRAO. The book includes a Fortran version and a Pascal version of each of the algorithms it describes. These routines, which are well-suited to use on personal computers, are distributed on 5 $\frac{1}{4}$ " floppy disks. One set of disks has been ordered (for the NRAO library in Charlottesville); we will have to order additional copies. Because of differences in Pascal compilers, the Pascal routines may require minor modifications before they can be used with, say, the Turbo Pascal compiler.

The IMSL and NAG Libraries cover a richer variety of applications than the *Numerical Recipes* collection, and many of their algorithms are considerably more sophisticated. Our copies of these libraries, however, cannot legally be used on personal computers.

[1] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing*, Cambridge Univ. Press, Cambridge, 1986; available in the VLA and Charlottesville libraries of the NRAO, call number QA 297.N866.

LINPACK. LINPACK, which was produced at Argonne National Laboratory, is a comprehensive, non-proprietary collection of Fortran subroutines for analyzing and solving systems of linear algebraic equations and linear least-squares problems. The subroutines are machine-portable—in particular, there are no machine-dependent constants in the code. LINPACK is capable of handling full matrices of order less than a few hundred and band matrices of order less than several thousand. There are no subroutines for general sparse matrices; and iterative methods, appropriate for very large systems, are not implemented in LINPACK. The IMSL and NAG Library capabilities overlap fairly completely the capabilities of LINPACK. On the other hand, LINPACK subroutines are very widely used; many of the routines in the CACM collection—for example—call LINPACK routines, as do some of the routines from the TOEPLITZ package and de Boor's B-spline package (both are described below). LINPACK subroutines could appropriately be used on any personal computer equipped with a Fortran compiler.

There are four versions of each of the LINPACK subroutines: a single-precision real, a double-precision real, a single-precision complex, and a double-precision complex version. The versions are differently named: the first letter of a subroutine (S, D, C, or Z) indicates the type. I keep the LINPACK source code in four files on the Charlottesville VAX, in

```
cvax::dua0:[nrao.math.linpack]lins.for,lind.for,linc.for,linz.for .
```

Since the source code files are quite bulky, I also have an object module library,

```
dua0:[nrao.math]linpack.olb .
```

If you include in your login.com file a line reading

```
$ ass linpack cvax::dua0:[nrao.math]linpack.olb,
```

then you can link with the library by typing, for example, link myprog,linpack/lib.

I do not keep the source code on the Convex, but I have constructed a library that you can link with. The library is /nrao/math/linpack.a. You can link with it by typing, for example, fc myprog.f /nrao/math/linpack.a.

Although the LINPACK source code is well-commented, it is nearly essential to have a copy of the reference manual in hand when incorporating LINPACK routines in a new application. Unfortunately, the NRAO libraries do not have copies of the LINPACK manual; I'll request that copies be ordered. I have one copy of the manual, which I am willing to loan.

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

EISPACK. EISPACK is a package of Fortran subroutines to compute the eigenvalues and/or the eigenvectors of six classes of matrices: namely, complex general, complex Hermitian, real general, real symmetric, real symmetric tridiagonal, and special real tridiagonal matrices. The IMSL and NAG Libraries have almost complete overlap with EISPACK, so for new applications on machines which are licensed for IMSL or NAG, EISPACK is not essential. On the other hand, the EISPACK routines are non-proprietary and are of uniformly high-quality; they can be used, for example, on a personal computer equipped with a Fortran compiler. Too, the use of EISPACK is widespread: imported applications code is likely to use EISPACK subroutines, as do some routines from the CACM collection.

I have double-precision versions of the EISPACK routines available in an object module library on the Charlottesville VAX, in

```
cvax::dua0:[nrao.math]eispack.olb .
```

I will create a library on the Convex, which I shall name /nrao/math/eispack.a. The source code is on the VAX, in

```
cvax::dua0:[nrao.math.eispack]eispack.for .
```

In transporting the package to other computers one should be aware that one machine-dependent constant, the unit round-off error that is characteristic of the machine's floating-point arithmetic, needs to be substituted in DATA statements occurring throughout the code. Contact me if you need help in determining the appropriate value of this constant.

Our version of EISPACK is called EISPACK-2. An EISPACK-3 version is now available; I don't know whether it provides significantly greater capability.

B. T. Smith, J. M. Boyle, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, *Matrix Eigensystem Routines—EISPACK Guide*, Lecture Notes in Computer Science 6, Springer-Verlag, Berlin, 1974; available in the Charlottesville library of the NRAO, call number QA 193.M37.

MINPACK. MINPACK, created at Argonne National Laboratory, is a package of high-quality Fortran subroutines for the numerical solution of systems of nonlinear equations and for nonlinear least-squares problems. The routines are designed for ease of use and for transportability. They are useful in determining local extrema, in the case of optimization problems, or localizing zeros, in the case of simultaneous systems of nonlinear equations—as opposed to finding global extrema (i.e., the deepest minimum, in a minimization problem) or cataloging all of the roots in the case of a simultaneous system. The user must provide an initial guess of the problem solution; when successful, the routines often return an answer which is close to that guess. The IMSL and NAG Libraries provide subroutines with similar functionality; I would advise that IMSL or NAG routines be used whenever one is programming on a machine for which these libraries are licensed. In developing programs that are to be transported outside the Observatory, or in programming on a personal computer, use of the non-proprietary MINPACK routines might be recommended instead.

The MINPACK routines come in both single- and double-precision versions. The single- and double-precision object module libraries are, respectively, the files

```
cvax::dua0:[nrao.math]minpacks.olb  
and cvax::dua0:[nrao.math]minpackd.olb.
```

I keep the source code on the Charlottesville VAX, in the area dua0:[nrao.math.minpack]. I have not moved the code to the Convex. Three machine-dependent constants occur in

the code. Two subroutines provided with the package, SPMPAR and DPMPAR, can be used to determine the appropriate values of these constants when the code is transported from the VAX to a different type of computer. Contact me if you need additional help in determining the appropriate values of these constants.

The routines are documented in a concise technical report. I can furnish photocopies on request, omitting the program listings that make up the final pages of the report.

J. J. Moré, B. S. Garbow, and K. E. Hillstrom, *User Guide for MINPACK-1*, Technical Report ANL-80-74, Argonne National Lab., August 1980; available through NTIS.

The TOEPLITZ Package. The TOEPLITZ package is useful in specialized applications. It is a collection of Fortran subroutines for the numerical solution of systems of linear equations with coefficient matrices of Toeplitz or circulant form (or of block-Toeplitz or block-circulant form).³ Such systems arise frequently in signal processing applications, time series analysis, statistics, and the numerical solution of integral equations with difference kernels (e.g., convolution equations, Abel equations). The algorithms used by this package are much more efficient than are the standard methods ($O(n^2)$ versus $O(n^3)$ for an $n \times n$ system) when applied to Toeplitz or circulant systems.

The TOEPLITZ package was produced in the late 1970's as part of an international collaborative effort known as the SALAR (Soviet-American Libraries and Algorithms Research) project.

This is a non-proprietary package, and it has no overlap with the IMSL Library or (to my knowledge) the NAG Library. It uses some subroutines from LINPACK. I keep the source code in `cvax::dua0:[nrao.math.toeplitz]toeplitz.for`. The object module library is `dua0:[nrao.math]toeplitz.olb`. The subroutines come in both single- and double-precision versions (differently named); they are machine-portable, standard Fortran. TOEPLITZ package subroutines should run efficiently on any personal computer equipped with a Fortran compiler. The reference manual is brief; I can easily provide photocopies, omitting the program listings that constitute the final pages of the manual.

O. B. Arushanian, M. K. Samarin, V. V. Voevodin, E. E. Tyrtshnikov, B. S. Garbow, J. M. Boyle, W. R. Cowell, and K. W. Dritz, *The TOEPLITZ Package Users' Guide*, Technical Report ANL-83-16, Argonne National Lab., October 1983; available through NTIS.

The ACM Algorithm Collection (CACM). *Collected Algorithms from the Association for Computing Machinery (CACM)* is a (hodge-podge) collection of computer codes, now numbering over six hundred, that have been published over the years in two of the ACM journals—in *Journal of the ACM* prior to 1975, and in *The ACM Transactions on Mathematical Software* since that time. In the early years the algorithms were published in full, but more recently the complete listings have been distributed only on magnetic tape, through IMSL, Inc. There is a lot of quality material in this collection, as well as a lot of chaff. The codes are written in various computer languages—Algol, Fortran, PL/I, etc. Both the Charlottesville and the VLA libraries have CACM indices and (partial) listings in notebook form (call number QA 75.A68).

I do not have the full CACM collection in machine-readable form, but rather only the algorithms that were published between March 1975 and December 1979—Algorithms 493 through 545, inclusive. Other portions of the collection could be ordered, as the need arises, from IMSL; IMSL distributes five-year and one-year portions on tape at a slightly higher than nominal fee. I shall try to determine whether someone at one of the U. Va. offices of

³A Toeplitz matrix is a square matrix whose elements along the main diagonal and along each co-diagonal are equal. Thus a Toeplitz matrix is completely specified by its first row and first column. A circulant matrix is a special type of Toeplitz matrix: A circulant matrix is completely specified by its first row; each further row may be obtained from the previous one by a right cyclic shift.

computation has the full collection and could share it with us—it is non-proprietary. (We have an immediate need for routines from the 1980–1984 portion of the collection; we’ll probably soon place an order for them.)

LLSQ. I have available the Fortran routines described in Lawson and Hanson’s textbook [1] on solving linear least squares problems. LINPACK and the IMSL and NAG Libraries, of course contain reliable routines for least squares problems, but they lack algorithms for constrained problems. Lawson and Hanson’s strong point seems to be its treatment of problems which are subject to linear equality and inequality constraints. The fourteen subroutines and six driver programs from Appendix C of [1] can be found on the Charlottesville VAX in the area `cvax::dua0:[nrao.math.llsq]`.

I’ll ask the NRAO library to order the Lawson and Hanson book. Gentleman in [2] has published a critical review of [1].

[1] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Prentice–Hall, Englewood Cliffs, NJ, 1974.

[2] W. M. Gentleman, Review of [1], *SIAM Review*, 18 (1976) 518–520.

DE/STEP, INTRP. This is a high-quality variable order predictor–corrector code for numerical solution of the initial value problem for systems of ordinary differential equations. It is written in Fortran and is machine-portable. Similar codes are available in the IMSL and NAG Libraries, but this one is non-proprietary and so can be used anywhere that one wishes. Full documentation is given in [1] and comparisons with other codes in [2]. There is also a brief technical report describing the use of the code; I can furnish photocopies on request.

I keep the single-precision version of the code in `dua0:[nrao.math.destep]de.for` and the double-precision version in `dua0:[nrao.math.destep]dde.for` (I have them separated because the single- and double-precision subroutines are identically named). I have eliminated two machine-dependent constants from the code and inserted subroutine calls to compute the required constants. This makes the code completely machine-portable but slightly less efficient; some users might wish to undo my modification.

[1] L. F. Shampine and M. K. Gordon, *Computer Solution of Ordinary Differential Equations: The Initial Value Problem*, W. H. Freeman, San Francisco, 1975.

[2] L. F. Shampine, H. A. Watts, and S. M. Davenport, “Solving nonstiff ordinary differential equations—the state of the art”, *SIAM Review*, 18 (1976) 376–411.

de Boor’s B-Spline Package. Carl de Boor’s book [1] is a source of sound, practical advice on how to use splines. This package contains Fortran implementations of all the algorithms that are described there. de Boor’s approach is to represent splines not in the way that they often are represented, as piecewise polynomials, but rather as linear combinations of certain basis elements—B-splines—of the relevant linear space. That is, if one considers the space of all possible spline functions satisfying certain specified continuity or interpolation conditions, then the elements of this space can be represented as linear combinations of suitable basis elements—and the so-called B-splines form a very convenient basis for many kinds of computation. For example, in spline approximation if one is interested in determining the optimal positioning of knots or breakpoints, one may want to calculate the derivatives of the approximant with respect to, say, the knot placement. Here, the B-spline representation is much more convenient than the piecewise polynomial (pp) representation of the spline approximant. Additionally, the numerical stability of a B-spline computational algorithm is frequently superior to that of an equivalent pp algorithm.

The IMSL and NAG libraries contain subroutines for common types of spline computation, but the hard-core user is likely to want to the greater flexibility provided by de Boor’s

package. I keep the routines on the Charlottesville VAX, in the area

```
dua0:[nrao.math.bspline] .
```

The single-precision routines are in the file `boorsp.for`, and the double-precision versions are in `boordp.for`.

[1] C. de Boor, *A Practical Guide to Splines*, Applied Mathematical Sciences Series 27, Springer-Verlag, New York, 1978; available in the Charlottesville library of the NRAO, call number QA 224.D42.

The CalTech Graphics Package. The following information on the CalTech Graphics Package was kindly supplied by Neil Killeen. This package has been installed on the VLA and Charlottesville VAXes, but not, to my knowledge, on those at our Tucson locations.

PGPLOT. PGPLOT is a subroutine package (written in VAX-11 Fortran) for drawing graphs on a variety of display devices (both hardcopy and interactive devices) attached to VAX computers. For most applications, these subroutines are device-independent. The PGPLOT subroutines call a device-dependent set of subroutines, the GRPCKG package. New device drivers can be added to the GRPCKG subroutines fairly easily. Devices that are currently supported are:

Versatec, PostScript printers, QMS Lasergrafix, Printronix, Trilog, REGIS (VT125, VT240 terminals), Grinnel, Sigma ARGS, Tektronix (4006, 4010, 4100), Retrographics, Metafile, Null device.

Both the source and the object libraries are kept on disk. There is no subscription fee. This code was partially written by, and is maintained by Tim Pearson at Caltech (PHOBOS: :TJP).

The graphics facilities on the Charlottesville and VLA VAXes (CVAX, VAX3, and AIPS) are set up in a like manner. A single set of logical assignments and symbol definitions allows access to any of the PGPLOT graphics facilities: Including the following line in your `login.com` file causes the required assignments to be made, regardless of which VAX you use,

```
$@nrao:graphics .
```

To link your program with PGPLOT, you need to use one or the other of the following libraries,

```
graphics:grpckg/lib    (both PGPLOT and GRPCKG object modules)
graphics:grpshr/lib    (equivalent shareable image).
```

It generally doesn't much matter which library you use.⁴ Documentation for PGPLOT is available in a T_EX-generated manual. To obtain a copy on the QMS laser printer, issue the command

```
$print/que=qms pgdoc:pgplot.bit .
```

Documentation for GRPCKG is available in the form of a VAX RUNOFF file; to obtain a copy on the QMS laser printer, issue the command

```
$print/que=qms pgdoc:grpckg.mem .
```

Example programs are available in the area `pgex:.` A program to translate metafiles (a metafile is a file from which a plot file for any one of the supported devices can be generated) can be found in the area `meta:.`

⁴Linking with the "shareable image" library may be preferred if you have a large number of programs that use the graphics routines, because this helps conserve disk space. Too, it allows greater efficiency when several programs are using the library routines concurrently.

XYPLOT. The XYPLOT package is a collection of subroutines (written in VAX-11 Fortran) that call the PGPLOT subroutines and that are designed to enable the user to make quick x - y plots from his or her own program. They are not meant for making publishable quality plots (GENPLOT should be used for that purpose). They are useful, instead, for programs that require interactive plotting. The user, who need only know how to call a minimal number of subroutines, is channeled down an efficient path—generally a path with forks enough to be useful. The user can superimpose multiple curves on a single plot, using a variety of symbols, or draw bar graphs or vector fields.

This code was written by and is maintained by Neil Killeen (CVAX: :NEIL) of the NRAO. There is no subscription fee. All the necessary logical assignments for XYPLOT are obtained with the same `login.com` file command described above. The package can be found in

```
xyplot:xyplot.tlb    (source code text library)
xyplot:xyplot.olb    (object library)
xyplot:xyplot.doc    (documentation, standard VAX text file).
```

To link a program with these subroutines, one must link with the above object library, and also with the GRPCKG library (see above).

GENPLOT. GENPLOT is an interactive data plotting and data manipulation program that combines the PGPLOT package with a family of mathematical functions such as data smoothing and spline fitting. It also incorporates spreadsheet capabilities. GENPLOT contains some VAX specific code; in particular, it uses the VMS HELP Library facility. GENPLOT also requires a few IMSL routines (these could be replaced by routines from some other mathematical library such as NAG) and the integer function library ICH. GENPLOT is capable of producing publishable quality plots on the QMS laser printer.

GENPLOT was written by and is maintained by Dale Gary (DEIMOS: :DG) at Caltech. There is no subscription fee. The source code and the object libraries for both GENPLOT and ICH are kept on disk. All logical assignments and symbol definitions needed by GENPLOT are obtained with the `login.com` file command described above. To run GENPLOT simply issue the command

```
$genplot .
```

Documentation for GENPLOT is available in the form of a VAX RUNOFF file; to obtain a copy on the QMS printer issue the command

```
$print/que=qms genplt:genplot.mem .
```

CalComp Plotter Software. In addition to the CalTech graphics software which I just described, we have software to drive the CalComp pen plotter attached to the Charlottesville VAX (CVAX) (the CalTech software is not appropriate for use with pen plotters). This plotter is capable of superb quality output. The CalComp software is a bit harder to use than the CalTech software. It is available only on CVAX; it would make sense, though, to install it on the Convex as well, because of the proximity of the VAX and the Convex and because of the high-speed Ethernet connection between the two.

The software for the CalComp is identical in functionality to the CalComp software that used to be installed on the IBM computer in Charlottesville. The VAX version was written by Leroy Napier. It consists of a package of Fortran-callable subroutines—appropriate for drawing line graphs—and a contouring program. The documentation for the basic package is given in [1], and that for the contouring program in [2]. Copies of these manuals are available in the main office of the Computer Division in Charlottesville.

The basic package consists of routines you can use to choose the ink color, draw an axis, move the pen here or there on the paper (with the point down or retracted), draw a symbol at the current pen location, etc. The object module libraries are the files

```
uma3:[calcomp]calcomp66.olb
and uma3:[calcomp]calcomp77.olb .
```

Use the second library unless you compile your program with the `nof77` flag. To link your program, just type, for example,

```
link myprogram,uma3:[calcomp]calcomp77.olb/lib .
```

Your program cannot drive the plotter directly, but rather it must create a specially formatted file which you have to submit to the plotter queue. If your first call to the plotter routines is `CALL PLOTS(0.0,0.0,9)`, then the output will be directed to Fortran “logical unit” number 9. To direct this output to a specified file, e.g. `plot.name`, issue the DCL command

```
ass plot.name for009
```

prior to running your program. After your program has finished type

```
calcspool plot.name
```

to send the contents of the file into the plotter queue. You then should inform the computer operator that you have a plot waiting in the queue (he operates the plotter).

The contouring program operates just like the program `GPCP`, described in [2]. This program can be used to generate a contour drawing of a real-valued function of two real variables. You must supply it with a file containing a list of data points, say of the form $(x_i, y_i, f(x_i, y_i))$. The data may be regularly or irregularly spaced. The input file must also contain instructions indicating the desired size of the drawing, the separation between contours, the labeling information, etc. The contouring program is kept in the file

```
uma3:[calcomp.gpcp]gpcp.exe .
```

To use the program, follow the example given in

```
uma3:[calcomp.gpcp]example.txt .
```

[1] *Programming CalComp Pen Plotters*, California Computer Products, Inc., Anaheim, CA, 1968.

[2] *GPCP: A General Purpose Contouring Program*, California Computer Products, Inc., Anaheim, CA, 1968.

MACSYMA. MACSYMA is a large programming system used for performing symbolic as well as numerical mathematical manipulations. MACSYMA can be used, for example, to analytically evaluate definite and indefinite integrals, to differentiate, take limits, solve systems of linear or polynomial equations, factor polynomials, expand functions in Taylor or Laurent series, solve ordinary differential equations (using either direct or Laplace transform methods), plot curves, and manipulate matrices and tensors. A built-in language processor allows users to write their own programs for transforming symbolic expressions. The system includes an extensive collection of user-contributed programs, called the SHARE Library.

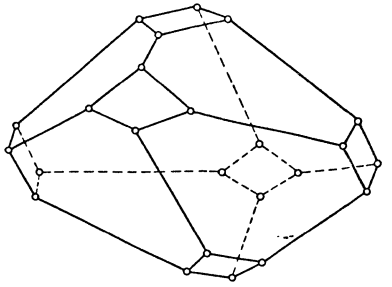


FIGURE 3. Voronoi cell for arbitrary three-dimensional lattice.

$\lambda|p_{01}, \dots|$ ($\lambda > 0$) described the same lattice. Furthermore any permutation of $\{0, 1, 2, 3\}$ gives a form which describes the same lattice.)

The Voronoi cell for an arbitrary lattice $[p_{01}, \dots]$ was described by Barnes in [1]. It is in general a truncated octahedron, as sketched in Figure 3, although some of the edges may collapse to points. To find G we must integrate $x \cdot x$ over this polyhedron, an apparently hopeless task. What makes it possible is that there is an explicit formula for the second moment of a tetrahedron [7, 15]. If T is a tetrahedron with vertices $0, p_1, p_2, p_3$, then its centroid is $q = (p_1 + p_2 + p_3)/4$, and

$$(4) \quad \int_T x \cdot x \, dx = \text{vol}(T) \left\{ \frac{4}{3} q \cdot q + \frac{1}{20} (p_1 \cdot p_1 + p_2 \cdot p_2 + p_3 \cdot p_3) \right\}.$$

We divide Figure 3 into 60 tetrahedra (six for each hexagonal face, two for each quadrilateral face) and with the help of ALTRAN add the 60 contributions from (4). The final result, after some cancellation and grouping of terms, is surprisingly simple:

$$(5) \quad G = \frac{S_1 D + S_2}{36 D^{4/3}},$$

where

$$\begin{aligned} D &= \text{deg } \Lambda = \sum_{(3)} p_{01} p_{02} p_{03} \\ &+ \sum_{(3)} p_{01} p_{23} (p_{02} + p_{03} + p_{12} + p_{13}), \\ S_1 &= p_{01} + p_{02} + p_{03} + p_{12} + p_{13} + p_{23}, \\ S_2 &= p_{01} p_{02} p_{13} p_{23} + p_{01} p_{03} p_{12} p_{23} + p_{02} p_{03} p_{12} p_{13} \\ &+ \sum_{(4)} p_{01} p_{02} p_{03} (p_{12} + p_{13} + p_{23}). \end{aligned}$$

(Here $\sum_{(4)} p_{01} p_{02} p_{03}$ stands for $p_{01} p_{02} p_{03} + p_{01} p_{12} p_{13} + p_{02} p_{12} p_{23} + p_{03} p_{13} p_{23}$, etc.)

The second stage consists in showing that the bcc lattice is the only local minimum of (5). It is surprising that the fcc lattice $[0, 1, 1, 1, 0]$ for example is not a local minimum, but MACSYMA shows this very easily. Using the Taylor series expansion facility, one finds that, for the neighboring lattice $\{\epsilon, 1, 1, 1, 1, \epsilon\}$,

$$G = \frac{1}{211/3} \left(1 - \frac{4\epsilon^2}{81} + \dots \right),$$

and so $[0, 1, 1, 1, 1, 0]$ is not a local minimum. On the other hand, by looking at the matrix of second derivatives $\partial^2 G / \partial p_{ij} \partial p_{kl}$, we see that the bcc lattice is a local minimum.

The difficult part is to show that there is no other local minimum. We assume that $\bar{p} = [p_{01}, \dots, p_{23}]$ is a local minimum, and eventually deduce that all \bar{p}_{ij} must be equal. Because the \bar{p}_{ij} in (3) must be nonnegative, it is important to identify which of the \bar{p}_{ij} are zero. The argument proceeds in a series of steps, of which the following is typical. Having shown that no more than two \bar{p}_{ij} are zero, we wish to show that no single \bar{p}_{ij} may be zero. Suppose $\bar{p}_{01} = 0$ and the other $\bar{p}_{ij} > 0$. Consider changing \bar{p} to

$$(6) \quad \bar{p} + \epsilon[0, -\bar{p}_{01}, \bar{p}_{01}, \bar{p}_{01} + \bar{p}_{12} + \bar{p}_{13}, -\bar{p}_{01} - \bar{p}_{12} - \bar{p}_{13}, \bar{p}_{12} - \bar{p}_{13}],$$

Using MACSYMA we find that this fixes the denominator of (5) to the first order in ϵ , while the numerator is increased by

$$-2\epsilon \bar{p}_{01} \bar{p}_{12} \bar{p}_{13} (\bar{p}_{13} - \bar{p}_{12}).$$

Since the sign of ϵ is arbitrary, and \bar{p} is supposed to be a local minimum, this implies $\bar{p}_{12} = \bar{p}_{13}$. Similarly $\bar{p}_{02} = \bar{p}_{03}$, and $\bar{p} = [0, a, a, b, b, c]$. But this is not a local minimum, because MACSYMA shows that the Taylor series expansion of G at $\{\epsilon, a, a, 1, 1, c\}$ is

$$G = G_0 - \epsilon C_1 + O(\epsilon^2),$$

where

$$(7) \quad G_1 = \frac{(a+1)^2 + (a-1)^2(a+3c+1)}{108 \cdot 2^{1/3} a^{2/3} (a+2c+1)^{2/3}} > 0.$$

After a series of such steps we eventually find that \bar{p} must be $[1, 1, 1, 1, 1, 1]$, which establishes:

Theorem [2]. *The bcc lattice is the optimal lattice quantizer in three dimensions, and furthermore is the only lattice at which G attains a local minimum.*

Nothing is known for higher dimensions, although there are several conjectures [7, 8].

This proof was a genuine collaboration between man and machine. We had to work hard to find transformations like (6) which would be successful, but MACSYMA was needed to discover their effect. However, MACSYMA did not produce expressions like (7) in the form given here, with the terms grouped in a way that shows that the expression is positive. Considerable manipulation by hand was needed in many places to transform the computer's output into the most convenient form. On the other hand the proof could not have been pushed through without the help of ALTRAN and MACSYMA, or some other computer algebra program.

References

1. E. S. Barnes, *The covering of space by spheres*, *Canad. J. Math.* **8** (1956), 293-304.
2. E. S. Barnes and N. J. A. Sloane, *The optimal lattice quantizer in three dimensions*, *SIAM J. Algebraic Discrete Methods* **4** (1983), 30-41.
3. W. S. Brown, *ALTRAN User's Manual*, 4th ed. Bell Laboratories, Murray Hill, New Jersey, 1977.
4. B. Buchberger et al., *Computer Algebra: Symbolic and Algebraic Computation*, Springer-Verlag, 1982.
5. J. Calmet, editor, *Computer Algebra*, Lecture Notes in Computer Science, vol. 144, Springer-Verlag, 1982.
6. B. W. Char, K. O. Geddes, W. M. Gentleman and G. H. Gonnet, *The design of Maple: a compact, portable and powerful computer algebra system*, in [20], pp. 101-115.
7. J. H. Conway and N. J. A. Sloane, *Voronoi regions of lattices, second moments of polytopes, and quantization*, *IEEE Trans. Inform. Theory* **IT-28** (1982), 211-226.
8. J. H. Conway and N. J. A. Sloane, *On the Voronoi regions of certain lattices*, *SIAM J. Algebraic Discrete Methods* **5** (1984), 294-305.
9. L. Fejes Tóth, *Sur la représentation d'une population sphérique par un nombre fini d'éléments*, *Acta Math. Acad. Sci. Hungar.* **10** (1959), 299-304.
10. J. Fitch, editor, *EUROSAM 84*, Lecture Notes in Computer Science, vol. 174, Springer-Verlag, 1984.
11. K. O. Geddes, G. H. Gonnet and B. W. Char, *MABLE User's Manual*, Univ. of Waterloo, Waterloo, Ontario, 1983.
12. A. Gersho, *Principles of quantization*, *IEEE Trans. Circuits and Systems* **CAS-25** (1978), 427-436.

13. A. Gersho, *Asymptotically optimal block quantization*, *IEEE Trans. Inform. Theory* **IT-25** (1979), 573-580.
14. V. E. Golden and The Matlab Group, *Introductory MACSYMA Documentation: A Collection of Papers*, Mass. Inst. Technology, Laboratory for Computer Science, Cambridge, Mass., 1983.
15. I. J. Good and R. A. Gaskins, *The centroid method of numerical integration*, *Numer. Math.* **16** (1971), 343-359.
16. J. H. Gritsmer, R. D. Jenks and D. Y. Yum, *SCRATCHPAD User's Manual*, IBM Watson Research Center, Report HA70, Yorktown Heights, N. Y., 1975.
17. A. D. Hall, Jr., *The ALTRAN system for rational function manipulation—a survey*, *Commun. ACM* **14** (1971), 517-521.
18. M. Hazewinkel, *Experimental mathematics*, *Math. Modelling* **6** (1985), 175-211.
19. A. C. Hearn, *REDUCE 2 User's Manual*, Univ. Utah, Computational Physics Group Report UCP-19, Salt Lake City, Utah, 2nd ed., 1973.
20. J. A. van Huitzen, editor, *Computer Algebra*, Lecture Notes in Computer Science, vol. 162, Springer-Verlag, 1983.
21. C. G. Lekkerkerker, *Geometry of Numbers*, Wolters-Noordhoff Groningen, 1969.
22. Matlab Group, *MACSYMA Reference Manual*, Mass. Inst. Technology, Laboratory for Computer Science, Cambridge, Mass., Version 10, 1983.
23. E. W. Ng, editor, *Symbolic and Algebraic Computation*, Lecture Notes in Computer Science, vol. 72, Springer-Verlag, 1979.
24. A. M. Odlyzko, *Applications of symbolic mathematics to mathematics*, in [25], pp. 95-111.
25. R. Pavelle, editor, *Applications of Computer Algebra*, Kluwer-Nijhoff Publ., 1985, to appear.
26. R. H. Rand, *Computer Algebra in Applied Mathematics: An Introduction to MACSYMA*, Pitman, Boston, 1984.
27. A. Rich and D. R. Stoutemyer, *Capabilities of the muMATH-79 computer algebra system for the INTEL-8080 microprocessor*, in [25], pp. 241-248.
28. G. Voronoi, *Sur quelques propriétés des formes quadratiques positives parfaites*, *J. Reine Angew. Math.* **133** (1907), 97-178.
29. H. S. Willf, *The disk with the college education*, *Amer. Math. Monthly* **89** (1982), 4-8.
30. S. Wolfram, *Symbolic mathematical computation*, *Commun. ACM* **28** (1985), 330-334.
31. S. Wolfram et al., *SMP Reference Manual*, Inference Corp., Los Angeles, Calif., 1983.
32. P. Zadori, *A symbolic quantization error of continuous signals and the quantization dimension*, *IEEE Trans. Inform. Theory* **IT-28** (1982), 139-149.