Science on Small Computers

David K. Kahaner, Editor

Matrix Computations on a PC

By Nick Higham

This issue's column is devoted to two packages for performing interactive matrix computations on MSDOS personal computers—GAUSS and PC-MATLAB. I have used both packages for research in matrix computations and, based on my experiences, offer the following comments on their capabilities and relative merits.

Use of either package requires an 80x87 numeric coprocessor. The 80x87 implements IEEE standard floating point arithmetic. It provides two precisions: 32-bit single precision (about seven decimal digits) and 64-bit double precision (about 16 digits). All its internal computations are carried out in 80-bit extended precision.

GAUSS, *Aptech Systems, Inc., 1914 N. 34th St., Suite 301, Seattle, WA 98103; (206) 547-1733.*

The GAUSS Programming Language costs \$200; the GAUSS Mathematical and Statistical System, which contains all the add-on modules, costs \$350. Discounts for multiple copies, and site licenses, are available.

System requirements are a PC with at least 320K (512K for high-resolution graphics), MSDOS 2.10 or later, and an 80x87 coprocessor.

PC-MATLAB, *The MathWorks, Inc., 21 Eliot St., South Natick, MA 01760; (508) 653-1415.*

PC-MATLAB costs \$695, or \$395 to universities. Discounts are available for orders of multiple copies; for example, 20 copies cost \$3,000 for a university. After purchasing at least one full copy, for \$500 a university can purchase a Classroom Kit version that allows 10 copy-protected installations to be made for student use only. The Control System Toolbox and the System Identification Toolbox cost \$395 each, or \$125 to universities, again with discounts for multiple copies. A demonstration disk with a booklet containing excerpts from the User's Guide costs \$10; the full version of the program is supplied, but with limitations on matrix size.

System requirements are a PC with at least 320K, MSDOS 2.0 or later, and an 80x87 coprocessor.

MATLAB, which was originally a public-domain Fortran program (see [1]), achieved wide use in university environments for teaching and research in matrix computations. In 1985, a commercial version, PC-MATLAB—faster, more extensive, and rewritten in C—was released by The MathWorks. MATLAB is available also on SUNs, APOLLOs, and VAXs (PRO-MATLAB), on the Macintosh (MacMATLAB), and in a version that takes full advantage of the new 80386-based PCs. The comments in this article are based on my experience with PC-MATLAB version 3.13 (September 1987).

GAUSS, from Aptech Systems, is broadly similar to PC-MATLAB in its aims and features, but it is more strongly oriented toward the statistical analysis of data and the manipulation of large sets of data stored on disk. Judging from the brochures, it is popular with statisticians, social scientists, and economists. First released in 1984–85, at present it exists only in a PC version. This article was written on the basis of my experience with GAUSS 1.49B, revision 15 (April 1987); Aptech subsequently released GAUSS 2.0—a major update of the package reviewed here. In the planning stage are a version for the Macintosh II and a version that will take advantage of the 80386 PCs.

A review of MATLAB, GAUSS, and similar packages was published recently in the *Notices* of the American Mathematical Society (Vol. 35, No. 7, September 1988, pages 978–1001).

Shared Features

The two packages have many features in common. Both use double precision (only). They seem to run the PC and the coprocessor at nearly maximum efficiency. Each has both interactive and procedure-driven modes. The main linear system and eigenvalue routines from the LINPACK and EISPACK Fortran libraries are included. Powerful matrix syntax is a feature of both packages. Any single matrix is limited to 8,192 elements (thus, $n \leq 90$ for an $n \times n$ matrix), since it must fit into one 64K MSDOS segment.

With both packages, all nongraphics screen output can be copied to a file (useful for editing and printing, including in reports). Users have considerable control over the format of numeric output. Facilities for directly calling compiled C and Fortran subroutines are included. DOS commands can be executed from within the program. Both suppliers produce regular newsletters for registered users. The programs are not copy protected.

Both packages include powerful programming languages (compiling to some forms of intermediate code). Despite the matrix-oriented design, the languages could be used for general-purpose programming! In addition to being standalone tools for experimentation and numerical testing, GAUSS and PC-MATLAB are increasingly being used for prototyping, that is, for developing and testing numerical algorithms interactively prior to translating them into such languages as C or Fortran.

Differences

Some of the main differences between the two packages are shown in the table.

The syntax of PC-MATLAB is the more elegant and concise, while GAUSS has more supplied commands, especially regarding statistical functions and handling "datasets" on disk. I find that the GAUSS compiler is fussy about syntax, and its error messages are often unhelpful or confusing; PC-MATLAB's error reporting is very good. To illustrate the syntax, I've written equivalent GAUSS and PC-MATLAB procedures for constructing a Vandermonde matrix (Figures 1 and 2).

When PC-MATLAB encounters a command or function it does not recognize—MYFUN, for example—it searches its own disk path for an M-file MYFUN.M; if found, it is compiled into memory and executed. GAUSS does not search in this way; rather, it requires that all desired procedures be loaded and compiled explicitly. Similarly, in PC-MATLAB, HELP MYFUN finds and displays the leading comment lines of MYFUN.M.

In my experience the packages run at similar speeds, with GAUSS being a shade faster than PC-MATLAB overall. As an indication of their speeds, both packages can multiply two 75×75 matrices or solve a linear system of order 90 in less than 30 seconds on an 8-MHz AT-compatible. While such speed is impressive and desirable, I find that such considerations as power and consistency of syntax, error reporting, and general ease of use are more important in determining the overall problem-solving time. On my machine with 512K of memory, PC-MATLAB leaves about 180K of workspace, while GAUSS leaves about 270K (the latter depending on the precise configuration adopted).

In summary, both packages are extremely useful aids for teaching and research in matrix computations. I can recommend both: Each has some advantages over the other. PC-MATLAB is the easier to learn and the more user-friendly, and would probably be the first choice of most numerical analysts; GAUSS has equal capabilities in the hands of the experienced user and might be preferred by statisticians and economists.

Reference

[1] C.B. Moler, *Demonstration of a matrix laboratory*, in Numerical Analysis, Mexico, 1981, J.P. Hennart, ed., Lecture Notes in Mathematics 909, Springer-Verlag, Berlin, 1982, pp. 84–98.

Nick Higham is a lecturer in mathematics at the University of Manchester, England. He is currently a visiting professor in the Department of Computer Science at Cornell University.

David K. Kahaner, editor of the Science on Small Computers column, is a group leader and mathematician at the National Institute of Standards and Technology. He encourages readers who have had interesting computational experiences with particular hardware or software to submit their comments or reviews for publication in future columns. Please send all materials directly to him (National Institute of Standards and Technology, Washington, DC 20899; kahaner@ceee-sed.arpa).

Table. Major Differences Between GAUSS and PC-MATLAB

PC-MATLAB	*GAUSS*
Intrinsic complex arithmetic.	Real arithmetic (some facilities for complex arithmetic by working with real and imaginary parts separately).
Full on-line help and demonstration programs.	Programs of neither type—rather unfriendly to the new user.
Line editor-based, with capability to recall and edit previous command lines.	Full screen editor—user can roam around the screen at will. Execute a sequence of commands contained within any marked area of the screen.
FOR loops.	No FOR loops—must be simulated with DO WHILE or DO UNTIL loops.
Vector elements referenced as $X(i)$.	Vector elements referenced as $X[i,1]$—must use two subscripts.
Access to Hessenberg and Schur decompositions and QZ algorithm (generalized eigenvalue problem).	Access lacking, but could be added by linking in compiled Fortran code.
Square systems solved with LINPACK's DGECO/DPOCO. Rectangular $Ax = b$ solved with the QR decomposition (DQRDC).	Uses specially written Crout/Cholesky routines that accumulate inner products in extended precision. Crout has two versions: no pivoting and partial pivoting. Rectangular $Ax = b$ solved by forming and solving the normal equations entirely in extended precision.
Fast screen handling.	Slower screen handling, due to forced use of ANSI.SYS screen driver.
Disk files called M-files (.M extension) are used to store sequences of PC-MATLAB statements. A disk file can be defined as a FUNCTION, in which case arguments can be passed in and out and variables are local to the function.	Disk files can contain combinations of GAUSS statements and procedures (PROCs). Procedures can be saved to disk in compiled form if they contain no references to global variables.
Functions can return multiple output arguments, e.g., $[V,D] = \mathrm{eig}(A,B)$. On a particular call, not all input or output arguments need to be specified, e.g., $D = \mathrm{eig}(X)$.	PROCs have single output arguments, and all input arguments must be specified.
A "%" denotes that the rest of a line is a comment line.	"@" characters delimit comments of up to a single line. /* and */ delimit multiline comments.
Toolboxes of M-files add new commands for signal processing (supplied), control systems, and system identification.	Various add-on modules available, consisting of collections of PROCs, e.g., for hi-res graphics (since only low-level routines are built in), unconstrained nonlinear optimization (quasi-Newton), nonlinear least squares, further statistics and data handling.
Extremely easy to use, comprehensive, and impressive graphics. The one program supports all the main graphics adaptors. The graphics screen can be saved to a device-independent MET file (analogous to a $\TeX$ DVI file), for printing to one of a variety of devices via the supplied graphics postprocessor.	On my copy, hi-res graphics require a CGA adaptor, so I haven't been able to try them. I believe other graphics cards are supported in the latest version.
Excellent typeset User's Guide. Contains a comprehensive tutorial, supplemented by the many supplied demonstration files.	The verbose 500+-page manual is reproduced from dot matrix printer output, making it very large and heavy. It contains tutorial, installation, and reference sections.
Installation and customization of the working environment are straightforward.	Installation is more involved. Great flexibility of configuration, at the price of complexity.

```
proc vand(p);
/* (Primal) Vandermonde matrix V based on the points p, i.e.
   V(i,j) = p(j)^(i-1).
   Special case: if p is a scalar then p equally spaced points on [0,1] are
   used. */

/* Must declare local variables.  Undeclared variables default to global
   type. */
local i, n, V;
n = maxc(rows(p)¦cols(p));   @ ¦ is vertical concatenation (~ for horiz.) @

/* Handle Scalar p. */
if n == 1;
   n = p;
   p = seqa(0,1/(n-1),n);    @ Additive sequence: seqa(start,inc,nterms). @
endif;

p = vec(p)';                 @ Ensure p is a row vector.  ' is transpose.@
V = ones(n,n);               @ n by n matrix of ones. @
i = 2;
do while i<=n;
   V[i,.] = p.*V[i-1,.];     @ V[i,.] is the i'th row of V. @
   i = i+1;                  @ .* denotes element by element multiplication. @
endo;
retp(V);                     @ Mechanism for returning a value from PROC. @
endp;
```

```
function V = vand(p)
%VAND   Vandermonde matrix.
%       V = VAND(P), where P is a vector, produces the (primal)
%       Vandermonde matrix based on the points P, i.e. V(i,j) = P(j)^(i-1).
%       Special case: if P is a scalar then P equally spaced points
%                     on [0,1] are used.

%       HELP VAND causes all the above lines to be displayed.

n = max(size(p));

%  Handle scalar p.
if n == 1
   n = p;
   p = (0:n-1)/(n-1);          % i:j denotes the row vector [i,i+1,...,j].
end

p = p(:)';                     % Ensure p is a row vector.  ' is transpose.
V = ones(n);                   % n by n matrix of ones.
for i=2:n
    V(i,:) = p.*V(i-1,:);      % V(i,:) is the i'th row of V.
end                            % .* denotes element by element multiplication.
```

Figure 1 (*left*). GAUSS procedure.
Figure 2 (*above*). PC-MATLAB function.