THE UNIVERSITY
of MANCHESTER

UMIST

The Test Matrix Toolbox for Matlab

N. J. Higham

Numerical Analysis Report No. 237

December 1993

University of Manchester/UMIST
Manchester Centre for Computational Mathematics
Numerical Analysis Reports

DEPARTMENT OF MATHEMATICS

Reports available from:
Department of Mathematics
University of Manchester
Manchester M13 9PL
England

And by anonymous ftp from:
`vtx.ma.man.ac.uk`
`(130.88.16.2)`
in `pub/narep`

The Test Matrix Toolbox for MATLAB

Nicholas J. Higham*

December 3, 1993

Abstract

We describe version 2.0 of the Test Matrix Toolbox for MATLAB 4. The toolbox contains a collection of test matrices, routines for visualizing matrices, and miscellaneous routines that provide useful additions to MATLAB's existing set of functions. There are 58 parametrized test matrices, which are mostly square, dense, nonrandom, and of arbitrary dimension. The test matrices include ones with known inverses or known eigenvalues; ill-conditioned or rank deficient matrices; and symmetric, positive definite, orthogonal, defective, involutary, and totally positive matrices. The visualization routines display surface plots of a matrix and its (pseudo-) inverse, the field of values, Gershgorin disks, and two- and three-dimensional views of pseudospectra. We explain the need for collections of test matrices and summarize the features of the collection in the toolbox. We give examples of the use of the toolbox and explain some of the interesting properties of the Frank and Pascal matrices, and of random, magic square and companion matrices. The leading comment lines from all the toolbox routines are listed.

Key words. test matrix, MATLAB, pseudospectrum, visualization, Frank matrix, Pascal matrix, companion matrix, magic square matrix, random matrix

AMS subject classifications. primary 65F05

Contents

---

*Department of Mathematics, University of Manchester, Manchester, M13 9PL, England (`na.nhigham@na-net.ornl.gov`). This work was supported by Science and Engineering Research Council grant GR/H52139.

1. Distribution

If you wish to distribute the toolbox please give exact copies of it, not selected routines.

2. Installation

The Test Matrix Toolbox is distributed as a Unix shar file, available by anonymous ftp from The MathWorks at Internet address `ftp.mathworks.com (144.212.100.10)` in directory `pub/contrib/linalg` as file `testmatrix.sh`. This document is `testmatrix.ps` in the same location.

To install the toolbox, download the shar file into your main Matlab directory, then type

```
sh testmatrix.sh
```

A directory `testmatrix` will be created containing the M-files in the toolbox.

The toolbox is also available from `vtx.ma.man.ac.uk (130.88.16.2)` in directory `pub/higham` as the compressed Unix tar file file `testmatrix.tar.Z`. This document is `narep237.ps.Z` in the same location. To install the toolbox from this location, download the tar file (in binary mode) into a `testmatrix` directory (`matlab/testmatrix` is recommended). Then uncompress the tar file and untar it:

```
uncompress testmatrix.tar.Z
tar xvf testmatrix
```

To try the toolbox from within MATLAB, change to the `testmatrix` directory and run the demonstration script by typing `tmtdemo`. For serious use it is best to put the `testmatrix` directory on the MATLAB path *before* the `matlab/toolbox` entries—this is because several toolbox routines have the same name as MATLAB routines and are intended to replace them (namely, `compan`, `cond`, `hadamard`, `hilb`, `kron` and `pascal`).

This document describes version 2.0 of the toolbox, dated November 14 1993.

3. Quick Reference Tables

This section contains quick reference tables to the Test Matrix Toolbox. All the M-files in the toolbox are listed by category, with a short description. More detailed documentation is given in Section 8, or can be obtained on-line by typing `help M-file_name`.

Demonstration	
`tmtdemo`	Demonstration of Test Matrix Toolbox.

Test Matrices, A–K	
`augment`	Augmented system matrix.
`cauchy`	Cauchy matrix.
`chebspec`	Chebyshev spectral differentiation matrix.
`chebvand`	Vandermonde-like matrix for the Chebyshev polynomials.
`chow`	Chow matrix—a singular Toeplitz lower Hessenberg matrix.
`circul`	Circulant matrix.
`clement`	Clement matrix—tridiagonal with zero diagonal entries.
`compan`	Companion matrix.
`condex`	‘Counter-examples’ to matrix condition number estimators.
`cycol`	Matrix whose columns repeat cyclically.
`dingdong`	Dingdong matrix—a symmetric Hankel matrix.
`dorr`	Dorr matrix—diagonally dominant, ill-conditioned, tridiagonal.
`dramadah`	A $(0,1)$ matrix whose inverse has large integer entries.
`fiedler`	Fiedler matrix—symmetric.
`forsythe`	Forsythe matrix—a perturbed Jordan block.
`frank`	Frank matrix—ill-conditioned eigenvalues.
`gallery`	Famous, and not so famous, test matrices.
`gearm`	Gear matrix.
`gfpp`	Matrix giving maximal growth factor for Gaussian elimination with partial pivoting.
`grcar`	Grcar matrix—a Toeplitz matrix with sensitive eigenvalues.
`hadamard`	Hadamard matrix.
`hanowa`	A matrix whose eigenvalues lie on a vertical line in the complex plane.
`hilb`	Hilbert matrix.
`invhess`	Inverse of an upper Hessenberg matrix.
`invol`	An involutory matrix.
`ipjfact`	A Hankel matrix with factorial elements.
`jordbloc`	Jordan block.
`kahan`	Kahan matrix—upper trapezoidal.
`kms`	Kac–Murdock–Szegö Toeplitz matrix.
`krylov`	Krylov matrix.

Test Matrices, L–Z	
`lauchli`	Lauchli matrix—rectangular.
`lehmer`	Lehmer matrix—symmetric positive definite.
`lesp`	A tridiagonal matrix with real, sensitive eigenvalues.
`lotkin`	Lotkin matrix.
`makejcf`	A matrix with given Jordan canonical form.
`minij`	Symmetric positive definite matrix $\min(i,j)$.
`moler`	Moler matrix—symmetric positive definite.
`neumann`	Singular matrix from the discrete Neumann problem (sparse).
`ohess`	Random, orthogonal upper Hessenberg matrix.
`orthog`	Orthogonal and nearly orthogonal matrices.
`parter`	Parter matrix—a Toeplitz matrix with singular values near π.
`pascal`	Pascal matrix.
`pdtoep`	Symmetric positive definite Toeplitz matrix.
`pei`	Pei matrix.
`pentoep`	Pentadiagonal Toeplitz matrix (sparse).
`poisson`	Block tridiagonal matrix from Poisson's equation (sparse).
`prolate`	Prolate matrix—symmetric, ill-conditioned Toeplitz matrix.
`rando`	Random matrix with elements -1, 0 or 1.
`randsvd`	Random matrix with pre-assigned singular values.
`redheff`	A matrix of 0s and 1s of Redheffer.
`riemann`	A matrix associated with the Riemann hypothesis.
`rschur`	An upper quasi-triangular matrix.
`smoke`	Smoke matrix—complex, with a 'smoke ring' pseudospectrum.
`tridiag`	Tridiagonal matrix (sparse).
`triw`	Upper triangular matrix discussed by Wilkinson and others.
`vand`	Vandermonde matrix.
`wathen`	Wathen matrix—a finite element matrix (sparse, random entries).
`wilk`	Various specific matrices devised/discussed by Wilkinson.

Visualization	
`fv`	Field of values (or numerical range).
`gersh`	Gershgorin disks.
`ps`	Dot plot of a pseudospectrum..
`pscont`	Contours and colour pictures of pseudospectra.
`see`	Pictures of a matrix and its (pseudo-) inverse.

Decompositions and Factorizations	
`cholp`	Cholesky factorization with pivoting of a positive semi-definite matrix.
`cod`	Complete orthogonal decomposition.
`ge`	Gaussian elimination without pivoting.
`gecp`	Gaussian elimination with complete pivoting.
`poldec`	Polar decomposition.
`signm`	Matrix sign decomposition.

Miscellaneous	
`bandred`	Band reduction by two-sided unitary transformations.
`chop`	Round matrix elements.
`comp`	Comparison matrices.
`cond`	Matrix condition number in 1, 2, Frobenius, or infinity norm.
`cpltaxes`	Determine suitable `axis` for plot of complex vector.
`dual`	Dual vector with respect to Hölder p-norm.
`eigsens`	Eigenvalue condition numbers.
`house`	Householder matrix.
`kron`	Kronecker tensor product (MATLAB 4.1 version).
`matrix`	Test Matrix Toolbox information and matrix access by number.
`matsignt`	Matrix sign function of a triangular matrix.
`pnorm`	Estimate of matrix p-norm ($1 \leq p \leq \infty$).
`qmult`	Pre-multiply by random orthogonal matrix.
`rq`	Rayleigh quotient.
`seqa`	Additive sequence.
`seqcheb`	Sequence of points related to Chebyshev polynomials.
`seqm`	Multiplicative sequence.
`show`	Display signs of matrix elements.
`skewpart`	Skew-symmetric (skew-Hermitian) part.
`sparsify`	Randomly sets matrix elements to zero.
`sub`	Principal submatrix.
`symmpart`	Symmetric (Hermitian) part.
`trap2tri`	Unitary reduction of trapezoidal matrix to triangular form.

4. Test Matrices

Numerical experiments are an indispensable part of research in numerical analysis. We do them for several reasons:

- To gain insight and understanding into an algorithm that is only partially understood theoretically.
- To verify the correctness of a theoretical analysis and to see if the analysis completely explains the practical behaviour.
- To compare rival methods with regard to accuracy, speed, reliability, and so on.
- To tune parameters in algorithms and codes, and to test heuristics.

One of the difficulties in designing experiments is finding good test problems—ones that reveal extremes of behaviour, cover a wide range of difficulty, are representative of practical problems, and (ideally) have known solutions. In many areas of numerical analysis good test problems have been identified, and several collections of such problems have been published. For example, collections are available in the areas of nonlinear optimization [39], linear programming [16], [36], ordinary differential equations [13], and partial differential equations [43].

Probably the most prolific devisers of test problems have been workers in matrix computations. Indeed, in the 1950s and 1960s it was common for a whole paper to be devoted to a particular test matrix: typically its inverse or eigenvalues would be obtained in closed form. An early survey of test matrices was given by Rutishauser [45]; most of the matrices he discusses come from continued fractions or moment problems. Two well-known books present collections of test matrices. Gregory and Karney [20] deal exclusively with the topic, while Westlake [55] gives an appendix of test matrices. In the 24 years since these books appeared several interesting matrices have been discovered (and in fact both books omit some worthy test matrices that were known at the time).

The Test Matrix Toolbox contains an up-to-date, well documented and readily accessible collection of test matrices. The matrices are given in the form of self-documenting MATLAB M-files. For some of the matrices we give mathematical formulas for the matrix elements in comment lines; in other cases the formulas can be reconstructed from the MATLAB code. We do not give exhaustive descriptions of matrix properties, or proofs of these properties; instead, in the comment lines we list a few key properties and give references where further details can be found.

With a few exceptions each of the 58 matrices satisfies the following requirements:

- It is a square matrix with one or more variable parameters, one of which is the dimension. Thus it is actually a parametrized family of matrices of arbitrary dimension.
- It is dense.
- It has some property that makes it of interest as a test matrix.

The first criterion is enforced because it is often desirable to explore the behaviour of a numerical method as parameters such as the matrix dimension vary. The third criterion is somewhat subjective, and the matrices presented here represent the author's personal choice. Note that we have omitted plausible matrices that we thought not "sufficiently different" from others in the collection. Although all but two of our test matrices are usually real, those with an arbitrary parameter can be made complex by choosing a non-real value for the parameter.

As well as their obvious application to research in matrix computations we hope that the matrices presented here will be useful for constructing test problems in other areas, such as optimization (see, for example, [3]) and ordinary differential equations.

We mention some other collections of test matrices that complement ours. The Harwell-Boeing collection of sparse matrices, largely drawn from practical problems, is presented by Duff, Grimes and Lewis [8], [9]. Bai [2] is building a collection of test matrices for the large-scale nonsymmetric eigenvalue problem. Zielke [58] gives various parametrized rectangular matrices of fixed dimension with known generalized inverses. Demmel and McKenney [6] present a suite of Fortran 77 codes for generating random square and rectangular matrices with prescribed singular values, eigenvalues, band structure, and other properties. This suite is part of the testing code for LAPACK [1]. Our focus is primarily on non-random matrices but we include a class of random matrices `randsvd` that has some of the features of the Demmel and McKenney test set.

Where possible, we have chosen the names of the test matrices eponymously, since it is easier to remember, for example, "the Kahan matrix", than "Example 3.8". For portability reasons we restrict all M-file names in the toolbox to eight characters (since this is the limit in the MSDOS operating system, under which the Microsoft Windows version of MATLAB runs). We have written a routine `matrix` that accesses the matrices by number rather than by name; this makes it easy to run experiments on the whole collection of matrices (with parameters other than the matrix dimension set to their default values.)

The matrices described here can be modified in various ways while still retaining some or all of their interesting properties. Among the many ways of constructing new test matrices from old are:

- Similarity transformations $A \leftarrow X^{-1}AX$.
- Unitary transformations $A \leftarrow UAV$, where $U^*U = V^*V = I$.
- Kronecker products $A \leftarrow A \otimes B$ or $B \otimes A$ (for which MATLAB has a routine `kron`).[1]
- Powers $A \leftarrow A^k$.

For a discussion of these techniques, and others, see [20, Chapter 2]. Techniques for obtaining a triangular, orthogonal, or symmetric positive definite matrix that is related to a given matrix include

- Bandwidth reduction using unitary transformations (see toolbox routine `bandred`).

---

[1] The toolbox includes the version of `kron` supplied with MATLAB 4.1 which, unlike the MATLAB 4.0 `kron` function, generates sparse output for sparse input.

- LU, Cholesky, QR and polar decompositions (see `lu`, `chol`, `qr` and, from the toolbox, `cholp`, `ge`, `gecp` and `poldec`.)

See [18] for details of these techniques.

Another way to generate a new matrix is to perturb an existing one. One approach is to add a random perturbation. Another is to round the matrix elements to a certain number of binary places; this can be done using the toolbox routine `chop`.

Our programming style is as follows. Each M-file `foo` begins with comment lines that are displayed when the user types `help foo`. The first comment line, the H1 line, is a self-contained statement of the purpose of the routine; the H1 lines are searched and displayed by MATLAB's `lookfor` command (e.g., `lookfor toeplitz`). Any further comments and references follow a blank line and so are not displayed by `help`. As far as possible, every routine sets default values for any arguments that are not specified. In particular, for most test matrix routines `testmat`, `A = testmat(n)` is a valid way to generate an $n \times n$ matrix. In general we have strived for conciseness, modularity, speed, and minimal use of temporary storage in our MATLAB codes. Hence, where possible, we used matrix or vector constructs instead of `for` loops and have used calls to existing M-files.

Some of those matrices that are banded with a small bandwidth are given the sparse storage format, to allow large matrices to be generated. The `full` function can be used to convert to non-sparse storage (e.g., `A = full(tridiag(32))`). We check for errors in parameters in some, but not all, cases. A few of the test matrix routines do not properly handle the dimension $n = 1$ (for example, they halt with an error, or return an empty matrix). We decided not to add extra code for this case, since the routines are unlikely to be called with $n = 1$.

The first release of this toolbox (version 1.0, July 4 1989) was described in a technical report [25]. The collection was subsequently published as ACM Algorithm 694 [27]. Prior to the current version, version 2.0, the most recent release was version 1.3, November 14 1991, which was available from netlib [7] and from the author by anonymous ftp. Version 2.0 incorporates many additions and improvements over version 1.3 and takes full advantage of the features of MATLAB 4.

Tables 4.1 and 4.2 provide a summary of the properties of the test matrices. The column headings have the following meanings:

Inverse: the inverse of the matrix is known explicitly.

Ill-cond: the matrix is ill-conditioned for some values of the parameters.

Rank: the matrix is rank-deficient for some values of the parameters (we exclude "trivial" examples such as `vand`, which is singular if its vector argument contains repeated points). Note that there are some matrices that are mathematically rank-deficient but behave as ill-conditioned full rank matrices in the presence of rounding errors; these are listed only as rank-deficient (for example, `chebspec`).

Symm: the matrix is symmetric for some values of the parameters.

Pos Def: the matrix is symmetric positive definite for some values of the parameters.

Orth: the matrix is orthogonal, or a diagonal scaling of an orthogonal matrix, for some values of the parameters.

Eig: something is known about the eigensystem (or the singular values), ranging from bounds or qualitative knowledge of the eigenvalues to explicit formulas for some or all eigenvalues and eigenvectors.

We summarise further interesting properties possessed by some of the matrices. Recall that A is a *Hankel matrix* if the anti-diagonals are constant ($a_{ij} = r_{i+j}$), *idempotent* if $A^2 = A$, *normal* if $A^*A = AA^*$ (or, equivalently, A is unitarily diagonalizable), *nilpotent* if $A^k = 0$ for some k, *involutary* if $A^2 = I$, *totally positive (nonnegative)* if the determinant of every submatrix is positive (nonnegative), and a *Toeplitz matrix* if the diagonals are constant ($a_{ij} = r_{j-i}$). A totally positive matrix has distinct, real and positive eigenvalues and its ith eigenvector (corresponding to the ith largest eigenvalue) has exactly $i-1$ sign changes [15, Theorem 13, p. 105]; this property is important in testing regularization algorithms [21], [22]. See [31] for further details of these matrix properties.

defective: `chebspec`, `gallery`, `gear`, `jordbloc`, `triw`

Hankel: `dingdong`, `hilb`, `ipjfact`

Hessenberg: `chow`, `frank`, `grcar`, `ohess`, `randsvd`

idempotent: `invol`

involutary: `invol`, `orthog`, `pascal`

normal (but not symmetric or orthogonal): `circul`

nilpotent: `chebspec`, `gallery`

rectangular: `chebvand`, `cycol`, `kahan`, `krylov`, `lauchli`, `rando`, `randsvd`, `triw`, `vand`

Toeplitz: `chow`, `dramadah`, `grcar`, `kms`, `parter`, `pentoep`, `prolate`

totally positive or totally nonnegative: `cauchy`[2], `hilb`, `lehmer`, `pascal`, `vand`[3]

tridiagonal: `clement`, `dorr`, `gallery`, `lesp`, `randsvd`, `tridiag`, `wilk`

inverse of a tridiagonal matrix: `kms`, `lehmer`, `minij`

triangular: `dramadah`, `jordbloc`, `kahan`, `pascal`, `triw`

Finally, we note that several of the test matrices are related to those supplied with MATLAB. The functions `hadamard` and `pascal` were in the first release of the toolbox and were subsequently included by The MathWorks in the MATLAB distribution. The toolbox version of `hadamard` is the same as the one in MATLAB 4.0 except for the addition of an

---

[2] `cauchy(x,y)` is totally positive if $0 < x_1 < \cdots < x_n$ and $0 < y_1 < \cdots < y_n$ [48, p. 295].

[3] `vand(p)` is totally positive if the p_i satisfy $0 < p_1 < \cdots < p_n$ [15, p. 99].

Matrix	Inverse	Ill-cond	Rank	Symm	Pos Def	Orth	Eig
augment		✓	✓				
cauchy	✓	✓		✓	✓		
chebspec			✓				✓
chebvand		✓				✓	
chow			✓				✓
circul				✓	✓		✓
clement	✓		✓	✓			✓
compan	✓		✓				✓
condex		✓					
cycol			✓				
dingdong				✓			✓
dorr		✓					
dramadah		✓					
fiedler	✓			✓			✓
forsythe	✓	✓					✓
frank		✓					✓
gallery	✓	✓	✓	✓	✓		✓
gearm			✓				✓
gfpp	✓	✓					
grcar							✓
hadamard	✓					✓	✓
hanowa							✓
hilb	✓	✓		✓	✓		
invhess	✓	✓		✓	✓		✓
invol	✓	✓					✓
ipjfact		✓		✓			
jordbloc	✓	✓	✓				✓
kahan	✓	✓	✓				
kms	✓	✓		✓	✓		
krylov		✓					

Table 4.1: Properties of the test matrices, A–K.

Matrix	Inverse	Ill-cond	Rank	Symm	Pos Def	Orth	Eig
lauchli		√					
lehmer	√			√	√		
lesp							√
lotkin	√	√					√
minij	√			√	√		√
moler	√	√		√	√		
neumann			√				√
ohess	√					√	√
orthog	√					√	√
parter							√
pascal	√	√		√	√		√
pdtoep	√	√	√	√	√		
pei	√	√		√	√		√
pentoep		√	√	√	√		
poisson	√			√	√		√
prolate		√		√	√		√
rando							
randsvd	√	√		√	√	√	
redheff							√
riemann							√
rschur		√					√
smoke	√						√
tridiag	√	√	√	√	√		√
triw	√	√					
vand	√	√					
wathen				√	√		√
wilk		√		√	√		√

Table 4.2: Properties of the test matrices, L–Z.

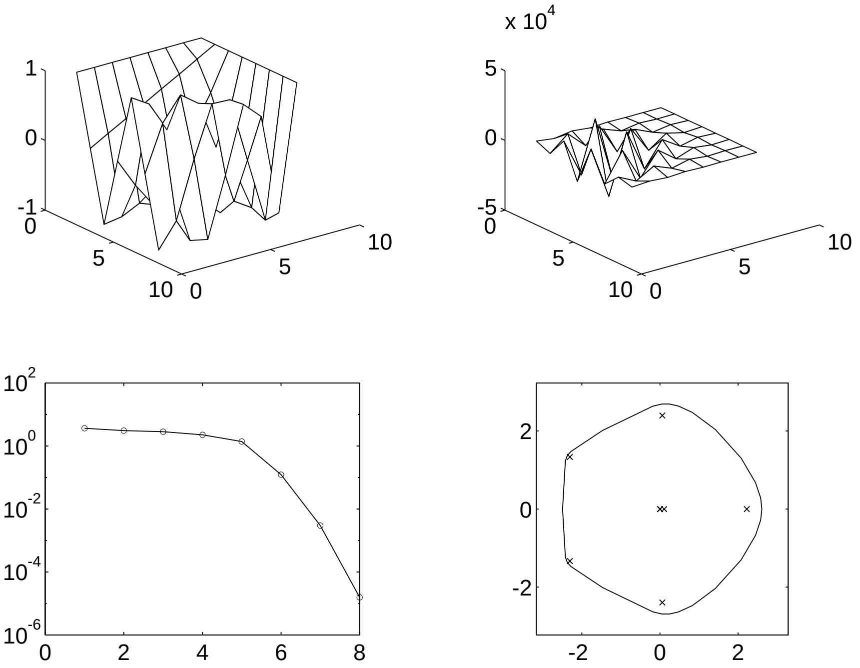


Figure 5.1: `see(chebvand(8))`.

H1 line, whereas the toolbox version of `pascal` contains more informative comment lines than the MATLAB 4.0 version and produces a different `pascal(n,2)` matrix[4] (but one that is still a cube root of the identity). The toolbox routine `compan` is more versatile than the MATLAB 4.0 version. Similarly, the toolbox routine `vand` is more versatile than MATLAB 4.0's `vander`. The toolbox version of `hilb` is coded differently and contains more informative comments than the one in MATLAB 4.0. The toolbox routine `augment` is similar to MATLAB 4.0's `spaugment`, but produces a non-sparse matrix instead of a sparse one. The toolbox function `cond` supports the 1, 2, ∞ and Frobenius norms, whereas MATLAB 4.0's `cond` supports only the 2-norm.

5. Visualization

The toolbox contains five routines for visualizing matrices. The routines can give insight into the properties of a matrix that is not easy to obtain by looking at the numerical entries. They also provide an easy way to generate pretty pictures!

The routine `see` displays a figure with four subplots (strictly speaking four "axes", in MATLAB terminology) in the format

`mesh(A)`	`mesh(pinv(A))`
`semilogy(svd(A))`	`fv(A)`

---

[4] The new `pascal(n,2)` is generated by a call to `rot90` and is "reverse upper triangular" instead of "reverse lower triangular" as in the MATLAB 4.0 version.

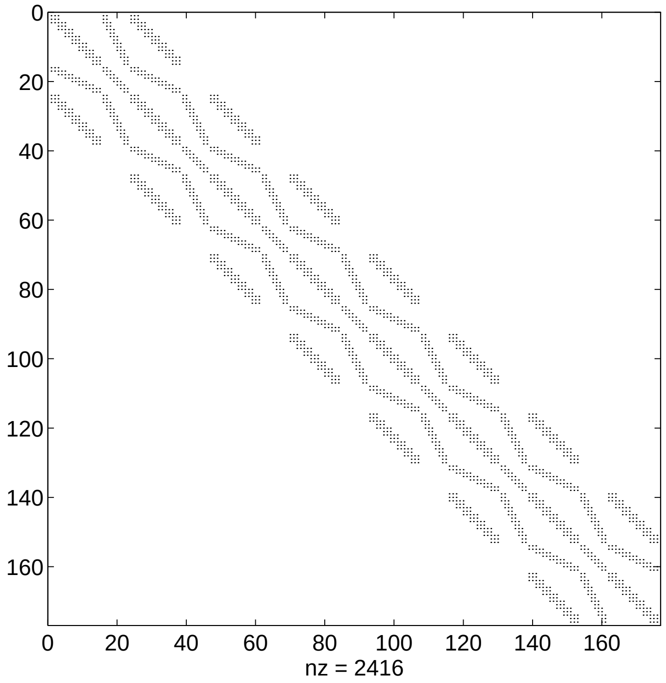


Figure 5.2: `see(wathen(7,7))`.

An example for the `chebvand` matrix is given in Figure 5.1. MATLAB's `mesh` command plots a three-dimensional, coloured, wire-frame surface, by regarding the entries of a matrix as specifying heights above a plane. We use `axis('ij')`, so that the coordinate system for the plot matches the (i, j) matrix element numbering. `pinv(A)` is the Moore–Penrose pseudo-inverse A^+ of A, which is the usual inverse when A is square and nonsingular. `semilogy(svd(A))` plots the singular values of A (ordered in decreasing size) on a logarithmic scale; the singular values are denoted by circles, which are joined by a solid line to emphasise the shape of the distribution. From Figure 5.1 we can see that `chebvand(8)` has a 2-norm condition number of about 10^5 and that the largest elements of its inverse are in the lower triangle. For a sparse MATLAB matrix, `see` simply displays a `spy` plot, which shows the sparsity pattern of the matrix. The user could, of course, try `see(full(A))` for a sparse matrix, but for large dimensions the storage and time required would be prohibitive. Figure 5.2 displays the result of applying `see` to the Wathen matrix—a symmetric positive definite sparse matrix that comes from a finite element problem.

The routine `fv` plots the field of values of a square matrix $A \in \mathbf{C}^{n\times n}$ (also called the numerical range), which is the set of all Rayleigh quotients,

$$\left\{ \frac{x^*Ax}{x^*x} : 0 \neq x \in \mathbf{C}^n \right\};$$

the eigenvalues of A are plotted as crosses. The field of values is a convex set that contains the eigenvalues. It is the convex hull of the eigenvalues when A is a normal matrix. If A is Hermitian, the field of values is just a segment of the real line. For non-Hermitian A the field

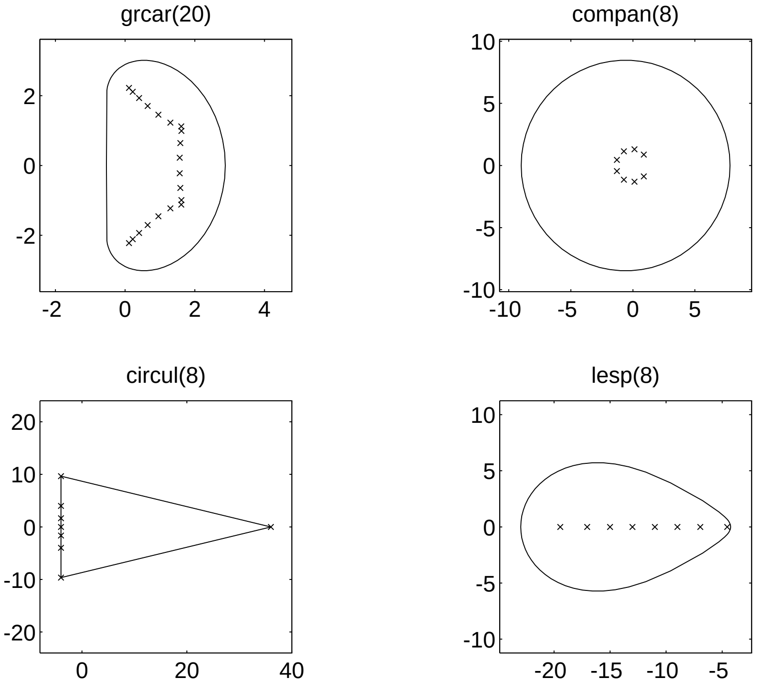


Figure 5.3: Fields of values (`fv`).

of values is usually two-dimensional and its shape and size gives some feel for the behaviour of the matrix. Trefethen [52] notes that the field of values is the largest reasonable answer to the question "Where in **C** does a matrix A 'live'?" and the spectrum is the smallest reasonable answer.

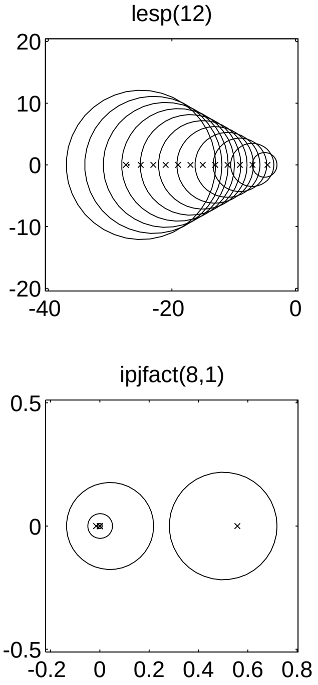
Some examples of field of values plots are given in Figure 5.3. The `circul` matrix is normal, hence its field of values is the convex hull of the eigenvalues. For an example of how the field of values gives insight into the problem of finding a nearest normal matrix see [44]. An excellent reference for the theory of the field of values is [32, Chapter 1].

The routine `gersh` plots the Gershgorin disks for an $A \in \mathbf{C}^{n \times n}$, which are the n disks

$$D_i = \{\, z \in \mathbf{C} : |z - a_{ii}| \le \sum_{\substack{j=1 \\ j \ne i}}^{n} |a_{ij}| \,\}$$

in the complex plane. Gershgorin's theorem tells us that the eigenvalues of A lie in the union of the disks, and an extension of the theorem states that if k disks form a connected region that is isolated from the other disks, then there are precisely k eigenvalues in this region. Thus the size of the disks gives a feel for how nearly diagonal A is, and their locations give information on where the eigenvalues lie in the complex plane. Four examples of Gershgorin disk plots are given in Figure 5.4; Gershgorin's theorem provides nontrivial information only for the third matrix, `ipjfact(8,1)`.

The last two routines, `ps` and `pscont`, are concerned with pseudospectra. The ϵ-



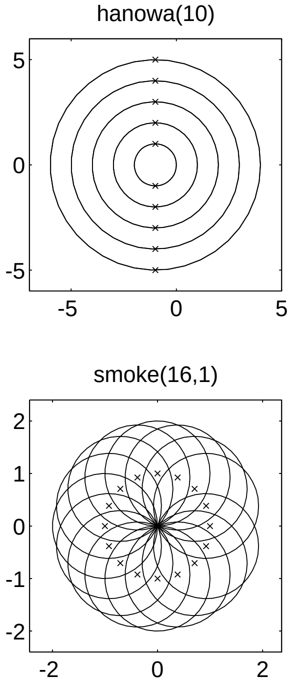


Figure 5.4: Gershgorin disks (`gersh`).

pseudospectrum of a matrix $A \in \mathbf{C}^{n\times n}$ is defined, for a given $\epsilon > 0$, to be the set

$$\Lambda_\epsilon(A) = \{\, z : z \text{ is an eigenvalue of } A + E \text{ for some } E \text{ with } \|E\|_2 \le \epsilon \,\}.$$

In other words, it is the set of all complex numbers that are eigenvalues of $A+E$ for some perturbation E of 2-norm at most ϵ. For a normal matrix A the ϵ-pseudospectrum is the union of the balls of radius ϵ around the eigenvalues of A. For nonnormal matrices the ϵ-pseudospectrum can take a wide variety of shapes and sizes, depending on the matrix and how nonnormal it is. Pseudospectra play an important role in many numerical problems. For full details see the work of Trefethen—in particular, [51] and [52].

The routine `ps` plots an approximation to the ϵ-pseudospectrum $\Lambda_\epsilon(A)$, which it obtains by computing the eigenvalues of a given number of random perturbations of A. The eigenvalues are plotted as crosses and the pseudo-eigenvalues as dots. Arguments to `ps` control the number and type of perturbations. Figure 5.5 gives four examples of 10^{-3}-pseudospectra, all of which involve the pentadiagonal Toeplitz matrix `pentoep`.

Another characterization of $\Lambda_\epsilon(A)$, in terms of the resolvent $(zI - A)^{-1}$, is

$$\Lambda_\epsilon(A) = \{\, z : \|(zI - A)^{-1}\|_2 \ge \epsilon^{-1} \,\}.$$

An alternative way of viewing the pseudospectrum is to plot the function

$$f(z) = \|(zI - A)^{-1}\|_2 = \sigma_{\min}(zI - A)$$

over the complex plane, where $\sigma_{\min}$ denotes the smallest singular value [52]. The routine `pscont` plots $\log_{10} f(z)^{-1}$ and offers several ways to view the surface: by its contour lines

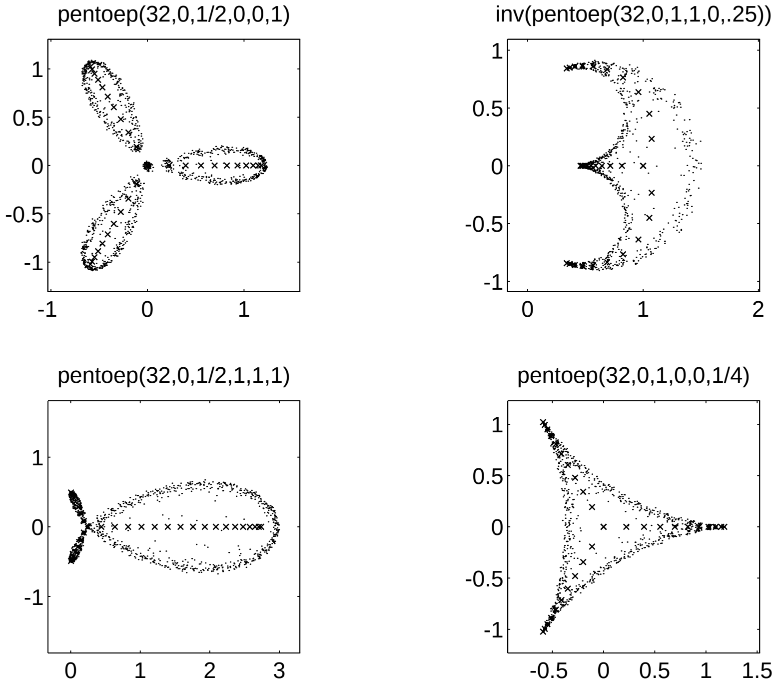


Figure 5.5: Pseudospectra (`ps`).

alone, or as a coloured surface plot in two or three dimensions, with or without contour lines. (The two-dimensional plot is the view from directly above the surface.) Two different `pscont` views of the pseudospectra of the triangular matrix `triw(11)` are given in Figures 5.6 and 5.7. Since all the eigenvalues of this matrix are equal to 1, there is a single point where the resolvent is unbounded in norm—this is the "bottomless pit" in the pictures. The spike in Figure 5.7 should be infinitely deep; since `pscont` evaluates $f(z)$ on a finite grid, the spike has a finite depth dependent on the grid spacing. Also because of the grid spacing chosen, the contours are a little jagged. Various aspects of the plots can be changed from the MATLAB command line upon return from `pscont`; for example, the colour map (`colormap`), the shading (`shading`), and the viewing angle (`view`). For Figure 5.6 we set `shading interp` and `colormap copper`.

Both pseudospectrum routines are computationally intensive, so the defaults for the arguments are chosen to produce a result in a reasonable time (under 20 seconds on a SPARC-2 processor or equivalent); for plots that reveal reasonable detail it is usually necessary to override the defaults.

6. Miscellaneous Routines

In addition to the test matrices and visualization routines, the Test Matrix Toolbox provides several routines that can be used to manipulate matrices or compute matrix functions or decompositions.

The decomposition functions offered are as follows.

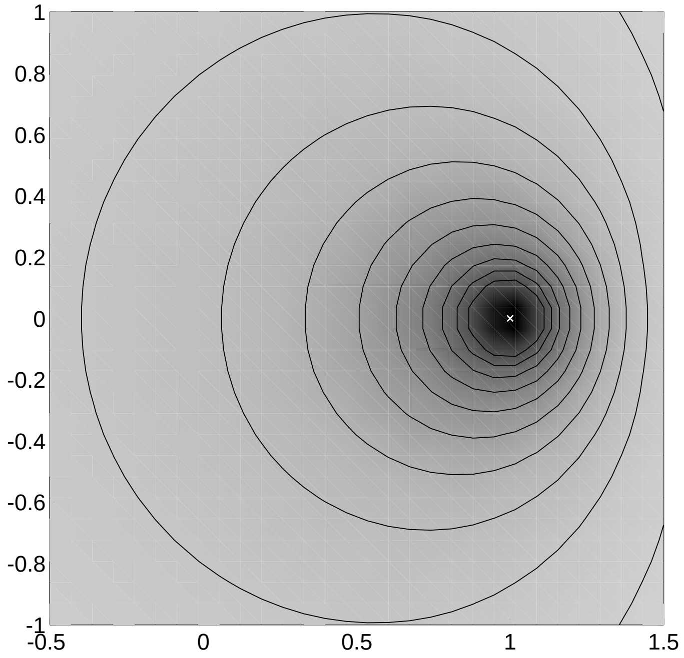


Figure 5.6: `pscont(triw(11), 0, 30, [-0.5 1.5 -1 1])`.

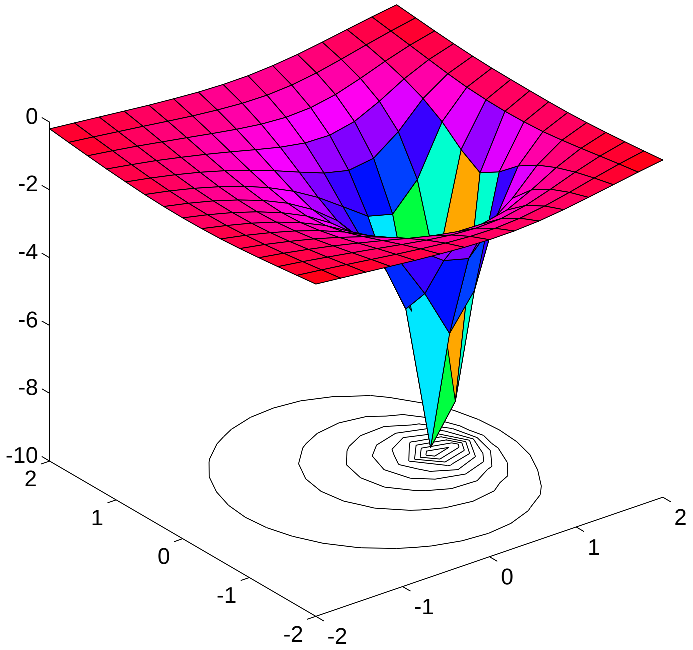


Figure 5.7: `pscont(triw(11), 2, 15, [-2 2 -2 2])`.

(1) `cholp` computes the Cholesky factorization with pivoting $\Pi^T A \Pi = R^* R$ of a Hermitian positive semi-definite matrix $A \in \mathbf{C}^{n \times n}$. Here, R is upper triangular with nonnegative diagonal elements and Π is a permutation matrix chosen