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 ($1 \le p \le \infty$).
`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		$\surd$	$\surd$				
cauchy	$\surd$	$\surd$		$\surd$	$\surd$		
chebspec			$\surd$				$\surd$
chebvand		$\surd$				$\surd$	
chow			$\surd$				$\surd$
circul				$\surd$	$\surd$		$\surd$
clement	$\surd$		$\surd$	$\surd$			$\surd$
compan	$\surd$		$\surd$				$\surd$
condex		$\surd$					
cycol			$\surd$				
dingdong				$\surd$			$\surd$
dorr		$\surd$					
dramadah		$\surd$					
fiedler	$\surd$			$\surd$			$\surd$
forsythe	$\surd$	$\surd$					$\surd$
frank		$\surd$					$\surd$
gallery	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$		$\surd$
gearm			$\surd$				$\surd$
gfpp	$\surd$	$\surd$					
grcar							$\surd$
hadamard	$\surd$					$\surd$	$\surd$
hanowa							$\surd$
hilb	$\surd$	$\surd$		$\surd$	$\surd$		
invhess	$\surd$	$\surd$		$\surd$	$\surd$		$\surd$
invol	$\surd$	$\surd$					$\surd$
ipjfact		$\surd$		$\surd$			
jordbloc	$\surd$	$\surd$	$\surd$				$\surd$
kahan	$\surd$	$\surd$	$\surd$				
kms	$\surd$	$\surd$		$\surd$	$\surd$		
krylov		$\surd$					

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.

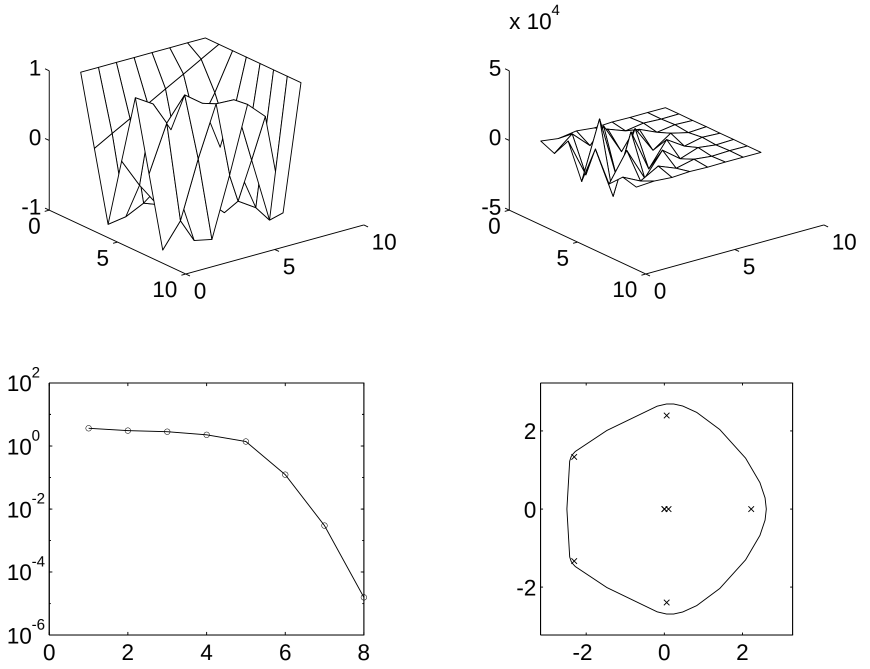

Figure 5.1: `see(chebvand(8))`.

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.

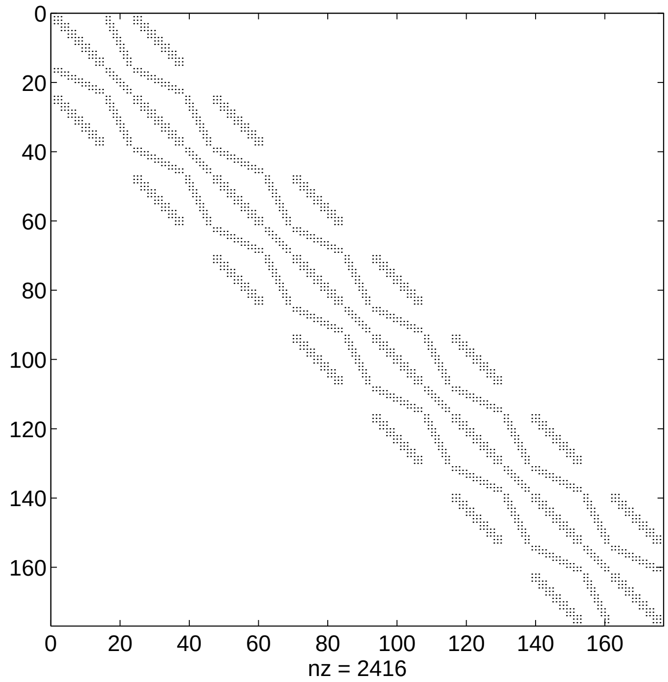

Figure 5.2: `see(wathen(7,7))`.

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

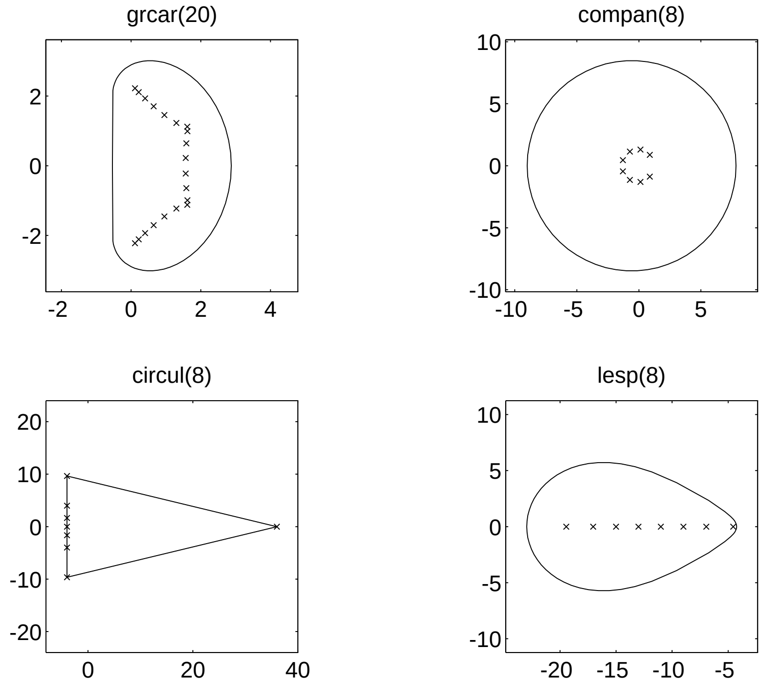

Figure 5.3: Fields of values (`fv`).

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 $\mathbf{C}$ 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-

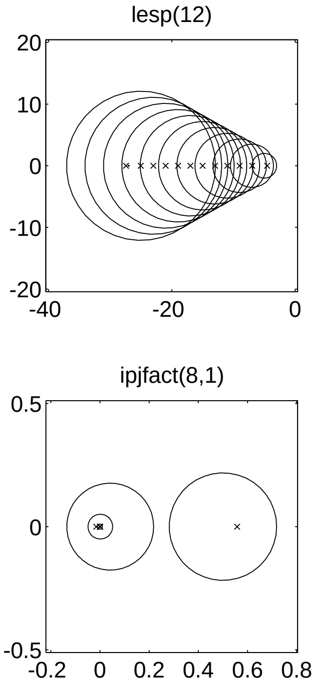

Figure 5.4: Gershgorin disks (`gersh`).

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

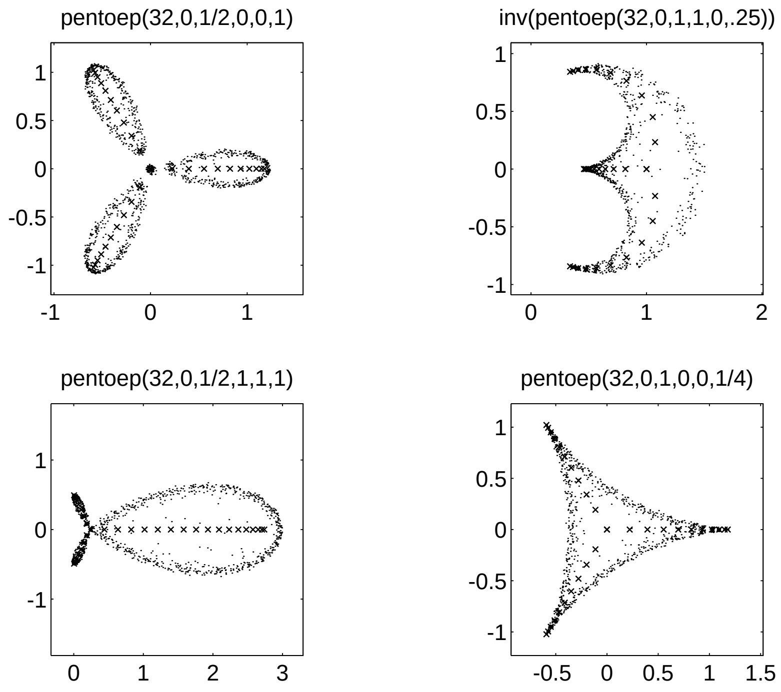

Figure 5.5: Pseudospectra (`ps`).

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.

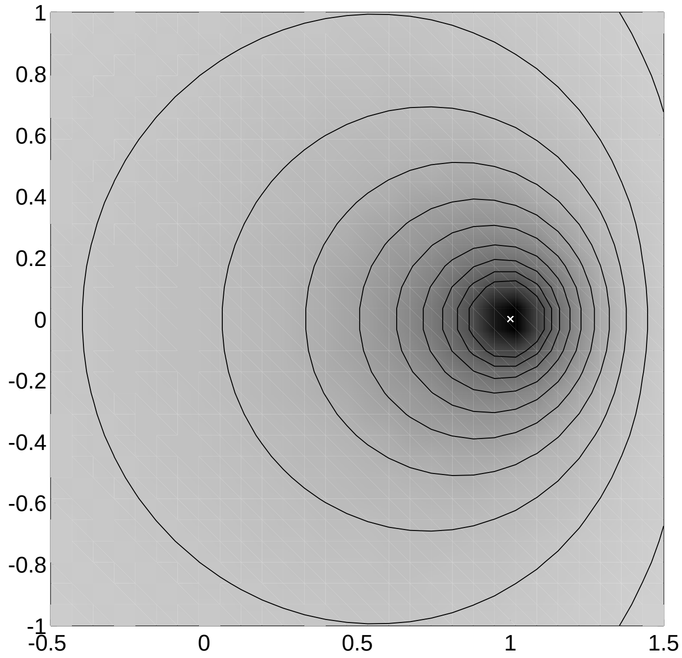

Figure 5.6: `pscont(triw(11), 0, 30, [-0.5 1.5 -1 1])`.

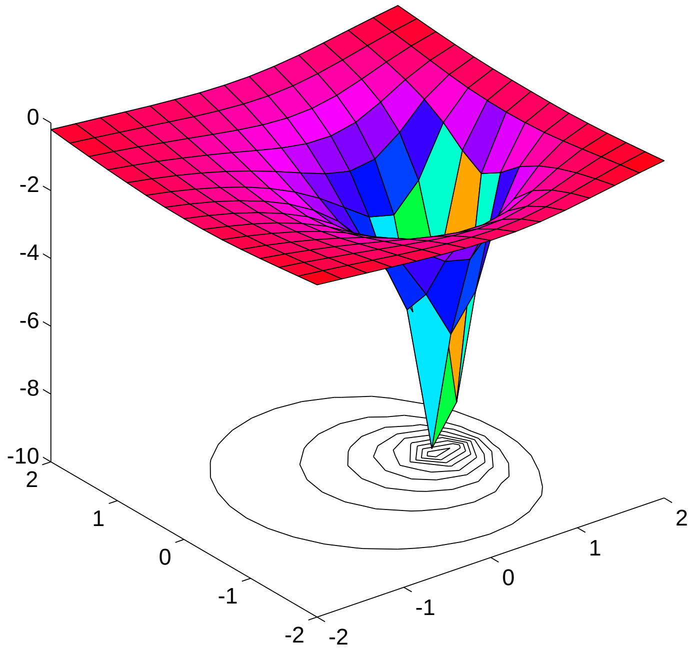

Figure 5.7: `pscont(triw(11), 2, 15, [-2 2 -2 2])`.

(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 to permute the largest diagonal element to the pivot position at each stage of the reduction (see [18, Section 4.2.9]). For the usual Cholesky factorization, as computed by `chol`, Π is the identity. Whereas `chol` can break down when presented with a Hermitian positive semi-definite matrix that is singular, `cholp` will always succeed.

(2) `cod` computes the complete orthogonal decomposition of a rank r matrix $A \in \mathbf{C}^{m \times n}$,

$$A = U \begin{bmatrix} R & 0 \\ 0 & 0 \end{bmatrix} V,$$

where $R \in \mathbf{C}^{r \times r}$ is upper triangular and $U \in \mathbf{C}^{m \times m}$ and $V \in \mathbf{C}^{n \times n}$ are unitary. The criterion used to define the numerical rank is a simple one based on the diagonal elements of the upper triangular matrix from the QR factorization with column pivoting. The complete orthogonal decomposition is an important tool in rank-deficient least squares problems [18, Sec. 5.5.2], [46].

(3) `ge` implements Gaussian elimination without pivoting. This routine is similar to `lu`, except that no row interchanges are done. Thus the routine computes, if possible, the LU factorization $A = LU$ of $A \in \mathbf{C}^{n \times n}$. The routine is of pedagogical interest, but also of practical interest because there are certain classes of $Ax = b$ problem where a more accurate solution is obtained from LU factorization when there are no row or column interchanges [26].

(4) `gecp` implements Gaussian elimination with complete pivoting. Thus it computes the factorization $PAQ = LU$ of $A \in \mathbf{C}^{n \times n}$, where P and Q are permutation matrices and L and U are lower and upper triangular, respectively. At the kth stage of the reduction of A to triangular form, row and column interchanges are used to bring the element of largest absolute value in the active submatrix to the pivot position (k, k) [18, Sec. 3.4.8].

(5) `poldec` computes the polar decomposition $A = UH \in \mathbf{C}^{m \times n}$, where $H \in \mathbf{C}^{n \times n}$ is Hermitian positive semi-definite and $U \in \mathbf{C}^{m \times n}$ has orthonormal columns or rows, according as $m \geq n$ or $m \leq n$. The polar decomposition is a generalization of the polar representation $z = re^{i\theta}$ for complex numbers. The factor U has the property that when $m \geq n$ it is the nearest matrix with orthonormal columns to A for both the 2-norm and the Frobenius norm:

$$\|A - U\| = \min\{\, \|A - Q\| : Q^* Q = I,\ Q \in \mathbf{C}^{m \times n} \}.$$

For more details see [23] or [31].

(6) `signm` computes the matrix sign decomposition $A = SN \in \mathbf{C}^{n \times n}$, where $S = \mathrm{sign}(A)$ is the matrix sign function [29]. If A has the Jordan canonical form

$$A = XJX^{-1} = X \begin{bmatrix} J_1 & 0 \\ 0 & J_2 \end{bmatrix} X^{-1},$$

where the eigenvalues of J_1 lie in the open left half-plane and those of J_2 lie in the open right half-plane, then

$$\mathrm{sign}(A) = X \begin{bmatrix} -I & 0 \\ 0 & I \end{bmatrix} X^{-1}.$$

(The sign function is not defined if A has any pure imaginary eigenvalues). The matrix sign function has several applications and is the subject of much recent research; see [29] for details and further references. Since $\mathrm{sign}(A)^2 = I$, `signm` provides one way to generate involutary matrices.

The toolbox contains further miscellaneous routines, including the following ones.

`bandred`: bandwidth reduction by unitary transformation (called by `randsvd`).

`comp`: forms comparison matrices.

`cond`: generalizes the `cond` function supplied with MATLAB 4.0 to work with the 1, ∞ and Frobenius norms (for square matrices) as well as the 2-norm.

`qmult`: Premultiplies a matrix by a random real orthogonal matrix from the Haar distribution (called by `randsvd`).

`seqa`, `seqm`: form additive or multiplicative sequences.

`sparsify`: randomly sets elements of a matrix to zero.

`pnorm`: estimates the p-norm of a matrix for $1 \le p \le \infty$ (MATLAB's `norm` works only for $p = 1, 2, \infty,$ `'fro'`).

`eigsens`: evaluates the Wilkinson condition numbers for the eigenvalues of a matrix.

7. Examples

In this section we give examples of the use of the toolbox and explain some of the interesting properties of the Frank matrix and the Pascal matrix.

7.1. Magic Squares

In the winter 1993 MathWorks Newsletter, Moler described some of the fascinating properties of magic squares, as embodied in MATLAB's `magic` function [38]. Some further properties can be illustrated with the aid of the toolbox. Recall that a magic square is an $n \times n$ matrix containing the integers from 1 to n^2 whose row and column sums are all the same. Let μ_n denote the magic sum of `magic(n)` (thus, $\mu_n = n^2(n+1)/2$).

Moler pointed out that the largest singular value of `A = magic(n)` (namely `max(svd(A))`) is μ_n, but left the proof as an exercise. The largest singular value of A is its 2-norm, so the problem is to prove that $\|A\|_2 = \mu_n$. This leads naturally to the question of what is the p-norm of a magic square, for any p between 1 and ∞. The Hölder p-norm of an $m \times n$ matrix A is defined by

$$\|A\|_p = \max_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}, \tag{7.1}$$

where $p \geq 1$ and $\|x\|_p = (\sum_{i=1}^n |x_i|^p)^{1/p}$. We can investigate the p-norm of a magic square using the toolbox function `pnorm`, which computes an *estimate* of $\|A\|_p$ using a generalization of the power method.

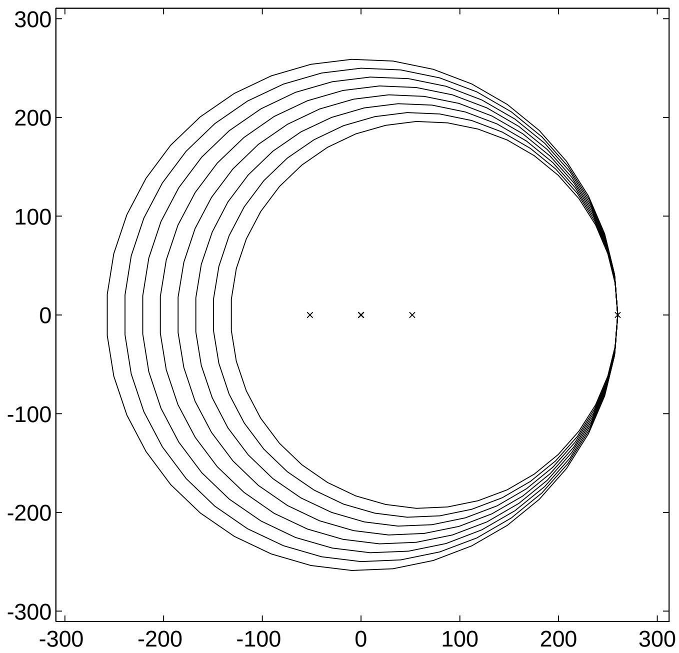

Figure 7.1: Gershgorin disks for magic(8).

```
for p = [1 1.5 2 exp(1) pi 10 inf]
    fprintf(' %9.4f    %9.4f\n', p, pnorm(magic(10),p))
end
    1.0000      505.0000
    1.5000      504.9968
    2.0000      504.9968
    2.7183      504.9971
    3.1416      504.9997
   10.0000      504.9988
       Inf      505.0000
```

All the p-norms in this example are very close to $\mu_{10} = 505$. Since the default convergence tolerance for `pnorm` is 10^{-4}, the exact p-norms could all be 505, as far as we can tell from the estimates. In fact, $\|A\|_p \equiv \mu_n$ for all $1 \leq p \leq \infty$. The proof relies on the convexity of the p-norm, which yields the inequality (see, [28], for example)

$$\|A\|_p \leq \|A\|_1^{1/p} \|A\|_\infty^{1-1/p}.$$

(This inequality is well-known for $p = 2$.) For a magic square, $\|A\|_1 = \|A\|_\infty = \mu_n$, so the inequality gives $\|A\|_p \leq \mu_n$. But by taking x in (7.1) to be the vector of all ones, we see that $\|A\|_p \geq \mu_n$, and so it follows that $\|A\|_p = \mu_n$. This result is actually a special case of an apparently little-known 1962 result of Stoer and Witzgall, which says that the norm of a doubly stochastic matrix is 1 for any norm subordinate to a permutation-invariant absolute vector norm [47].

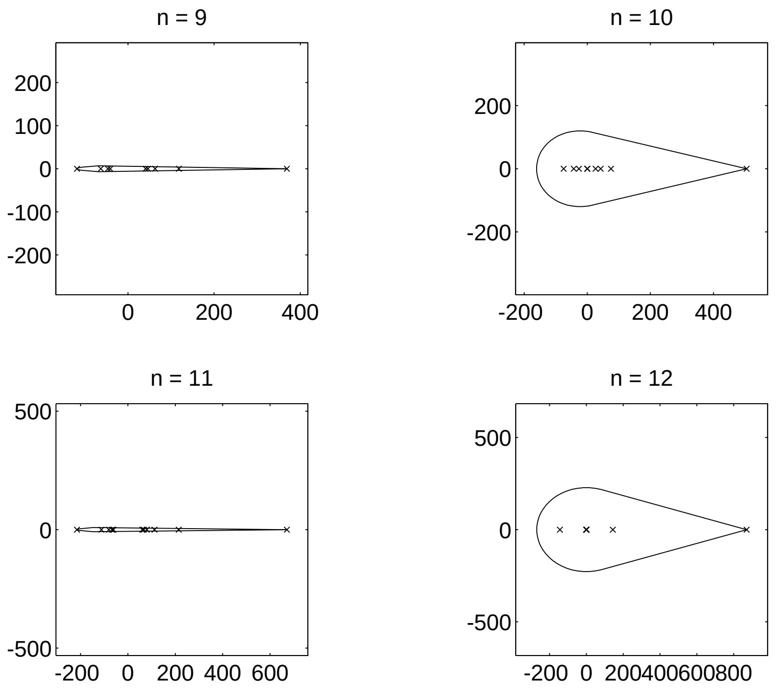

Figure 7.2: Field of values for `magic(n)`.

To estimate the eigenvalues of `magic(n)` we can apply Gershgorin's theorem. Unfortunately, the results are not very informative because the Gershgorin disks are all approximately the same, as is clear from the structure of the matrix; see Figure 7.1.

In his article, Moler pointed out that the function `magic` uses different algorithms for odd n, even n divisible by 4, and even n not divisible by 4. He gave four `mesh` plots to illustrate the difference. Another approach is to look at the fields of values—see Figure 7.2. The plot for $n = 10$ reflects the fact that `magic(n)` has rank $n/2 + 2$ when n is even and not divisible by 4—there are only 6 eigenvalues away from the origin (`magic(10)` is diagonalizable). For n divisible by 4 the rank is only 3.

7.2. Random Matrices

Random matrices are widely used for test purposes. The most popular way to generate a random matrix is to take its entries from a normal or uniform distribution. Intuitively, one might expect random matrices to be good at revealing programming errors and unusual behaviour of algorithms because they are "not special". This expectation is not necessarily correct, however. For example, Miller [37, pp. 96–97] describes a mutation experiment involving Fortran codes for Gaussian elimination without pivoting, Gaussian elimination with partial pivoting, and Gauss–Jordan elimination with partial pivoting. For each code, all possible mutants were generated, where a mutant is obtained by making a single typographical change to the source code. All the mutants were tested on a single random linear system $Ax = b$, with known solution, where a_{ij} was chosen from the uniform $[-1, 1]$

distribution. Many mutants were detected by their failure to pass the test of producing a solution with forward error less than a tolerance. However, some mutants passed this test, including all those that solve a system correctly in exact arithmetic; mutants in the latter class include those that select an incorrect pivot row and thus implement a numerically unstable algorithm. A conclusion to be drawn from Miller's experiment is that random test data can reveal some programming errors, but will not reveal all.

A good example of a problem for which random test matrices are very poor at revealing weaknesses in algorithms is matrix condition number estimation. The popular condition estimation algorithms can yield poor estimates but, in practice, never produce them for a random matrix [24]. The role of random matrices here is to indicate the quality of the estimates on average.

Edelman [11] summarizes the properties of random matrices well when he says that

> What is a mistake is to psychologically link a random matrix with the intuitive notion of a "typical" matrix or the vague concept of "any old matrix." In contrast, we argue that "random matrices" are very *special* matrices. The larger the size of the matrix the more predictable they are because of the central limit theorem.

Various results are known about the behaviour of matrices with elements from the standard normal distribution (mean 0, variance 1). Matrices of this type are generated by MATLAB's `randn` function. Let A_n denote an $n \times n$ matrix from this distribution and let $E(\cdot)$ be the expectation operator. Then, in the appropriate probabilistic sense, the following results hold as $n \to \infty$:

$$\begin{aligned}
E(\log(\kappa_2(A_n))) &\approx \log n + 1.537 \qquad \text{(real data)}, && (7.2)\\
E(\log(\kappa_2(A_n))) &\approx \log n + 0.982 \qquad \text{(complex data)}, && (7.3)\\
\|A_n\|_2 &\approx 2\sqrt{n} \qquad \text{(real data)}, && (7.4)\\
\|A_n\|_2 &\approx 2\sqrt{2}\sqrt{n} \qquad \text{(complex data)}, && (7.5)\\
\rho(A_n) &\approx \sqrt{n}. \qquad \text{(real or complex data)}. && (7.6)
\end{aligned}$$

For details of (7.2)–(7.5), see [10]. Edelman [10] conjectures that the condition number results are true for any distribution with mean 0—in particular, the uniform $[-1, 1]$ distribution used by MATLAB's `rand` function. The spectral radius result (7.6) has been proved as an inequality by Geman [17] for independent identically distributed random variables a_{ij} with zero mean and unit variance, and computer experiments suggest the approximate equality for the standard normal distribution [17].

Note that the first two results involve geometric means; the arithmetic mean, $\lim_{n\to\infty} E(\kappa_2(A_n))$, is infinite.

A question of interest in eigenvalue applications is how many eigenvalues of a random real matrix are real. The answer has been given by Edelman, Kostlan and Shub [12]: denoting by E_n the expected number of real eigenvalues of an $n \times n$ matrix from the standard normal distribution,

$$\lim_{n\to\infty} \frac{E_n}{\sqrt{n}} = \sqrt{\frac{2}{\pi}}.$$

Thus the proportion of real eigenvalues, E_n/n, tends to zero as $n \to \infty$. Exact formulae for E_n for finite n are also given in [12].

7.3. The Frank Matrix

A famous test matrix for eigensolvers is the $n \times n$ upper Hessenberg matrix F_n introduced by Frank in 1958 [14], illustrated for $n = 8$ by

```
F = frank(8)

F =

     8     7     6     5     4     3     2     1
     7     7     6     5     4     3     2     1
     0     6     6     5     4     3     2     1
     0     0     5     5     4     3     2     1
     0     0     0     4     4     3     2     1
     0     0     0     0     3     3     2     1
     0     0     0     0     0     2     2     1
     0     0     0     0     0     0     1     1
```

In evaluating three eigenvalue algorithms Frank found that this matrix "gives our selected procedures difficulties", and that "accuracy was lost in the smaller roots". The difficulties encountered by Frank's codes were shown by Wilkinson [56, Section 8], [57, pp. 92–93] to be caused by the inherent sensitivity of the eigenvalues to perturbations in the matrix.

The Frank matrix is interesting to analyze using MATLAB. The toolbox function `eigsens` evaluates the Wilkinson eigenvalue condition numbers, which are the reciprocals of the cosines of the angles between the left and right eigenvectors:

```
F = frank(10);
[V, D, s] = eigsens(F); d = diag(D); [x, k] = sort(d);
[d(k) s(k)]  % Eigenvalue followed by its condition number.
ans =

  3.9100e-002  7.7788e+004
  6.7743e-002  2.1165e+005
  1.2426e-001  2.0811e+005
  2.5692e-001  4.4765e+004
  6.1859e-001  7.3339e+002
  1.6166e+000  1.4589e+001
  3.8922e+000  1.5570e+000
  8.0476e+000  8.9143e-001
  1.4762e+001  2.9807e+000
  2.5575e+001  2.4721e+000
```

The output shows that the condition numbers grow, almost monotonically, as the eigenvalues decrease in size—in other words, the smallest eigenvalues are the most sensitive to

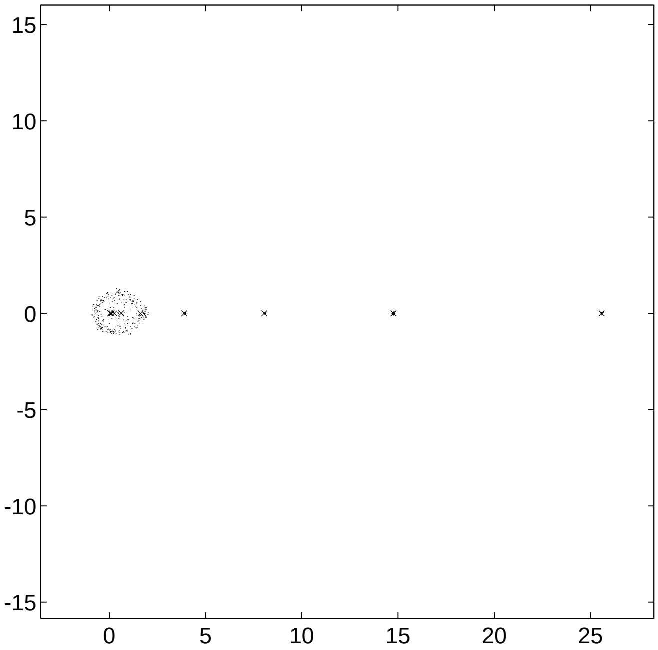

Figure 7.3: `ps(frank(10), 50, 1e-1, 0, 1)`.

perturbations in the matrix. The varying eigenvalue sensitivities can also be seen from pseudospectral plots. Figure 7.3 shows the 0.1-pseudospectrum, which shows that perturbations to F_{10} of 2-norm at most 0.1 have the greatest effect on the smallest eigenvalues. Another view is provided by Figure 7.4, for which we set `colormap hot`.

Further insight into the eigenvalues of F_n can be obtained by looking at its characteristic polynomial:

```
poly(F)

ans =

  1.0e+004 *

  Columns 1 through 7

    0.0001   -0.0055    0.1035   -0.8310    2.9505   -4.5297    2.9505

  Columns 8 through 11

   -0.8310    0.1035   -0.0055    0.0001
```

The coefficients seem to be palindromic! As a check we use the function `charpoly` from MATLAB 4's Maple Symbolic Toolbox [49] to compute the characteristic polynomial exactly:

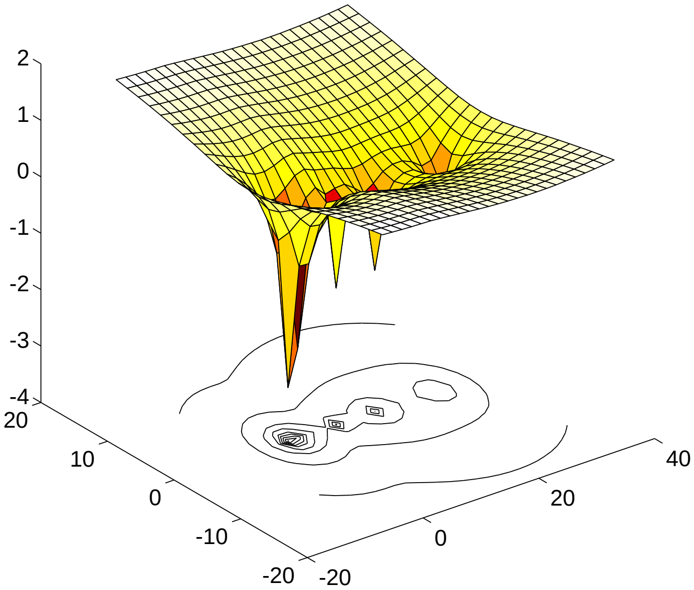

Figure 7.4: `pscont(frank(10), 2, 25, [-7 33 -20 20])`.

```
charpoly(F)

ans =

1-55*x+1035*x^2-8310*x^3+29505*x^4-45297*x^5+29505*x^6-8310*x^7+1035*x^8-55*x^9+x^10
```

Any matrix whose characteristic polynomial π_n has a palindromic coefficient vector has eigenvalues occurring in reciprocal pairs, since $\pi_n(\lambda) = \lambda^n \pi_n(1/\lambda)$. In particular, it has determinant 1, and 1 is an eigenvalue when n is odd. We can check the determinant property numerically:

```
F = frank(20); [det(F) det(F')]

ans =

     1   -14

F = frank(25); [det(F) det(F')]

ans =

            1   -48886168
```

Since $\det(A) = \det(A^T)$ for any matrix A, the output is mathematically incorrect. The reason is that rounding errors influence MATLAB's evaluation of $\det(F_n^T)$ much more than its evaluation of $\det(F_n)$; an illuminating discussion of this phenomenon is given by Frank [14] and Wilkinson [56, Section 8], [57, pp. 92–93]. The extreme sensitivity of $\det(F_n)$ to perturbations in F_n is easy to see: if we change the $(1, n)$ element from 1 to $1+\epsilon$, then $\det(F_n)$ changes from 1 to $1 + (-1)^n(n-1)!\epsilon$.

The inverse of F_n is lower Hessenberg. This can be seen using the following representation of F_n noted by Rutishauser [45, Section 9]:

$$F_n = PC_nP,$$

where P is the identity with the order of its columns reversed (`I = eye(n); P = I(:, n:-1:1)` in MATLAB notation) and

$$C_n = \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & 1 & & & \\ & 1 & 2 & & \\ & & 1 & \ddots & \\ & & & \ddots & n-1 \\ & & & & 1 \end{bmatrix}.$$

(By manipulating the identity $\det(F_n - \lambda I) = \det(C_n - \lambda I)$ the reciprocal pair property of the eigenvalues can be proved; cf. [54].) As an illustration, here is the exact inverse as returned by the Maple Symbolic Toolbox (the MATLAB function `inv` produces nonzero, but tiny, upper triangular elements because of rounding errors):

```
inverse(frank(8))

ans =

[    1,    -1,    0,    0,   0,  0,  0,  0]
[   -7,     8,   -1,    0,   0,  0,  0,  0]
[   42,   -48,    7,   -1,   0,  0,  0,  0]
[ -210,   240,  -35,    6,  -1,  0,  0,  0]
[  840,  -960,  140,  -24,   5, -1,  0,  0]
[-2520,  2880, -420,   72, -15,  4, -1,  0]
[ 5040, -5760,  840, -144,  30, -8,  3, -1]
[-5040,  5760, -840,  144, -30,  8, -3,  2]
```

In his 1958 paper, Frank commented

> "At the moment, the largest matrices resolved on the [Univac] 1103A are two 20-order matrices, one real symmetric and one complex. In both cases computing time was approximately one hour, and 6–8 places of accuracy were obtained."

The complete eigensystem of a complex 20×20 matrix A is found in under a second by the MATLAB command `eig(A)` on the workstation used for the examples reported here! This improvement over Frank's timing is attributable not only to hardware advances but also to an algorithmic breakthrough: `eig` uses the QR algorithm, which was not available to Frank.

7.4. The Pascal Matrix

The numbers in Pascal's triangle satisfy, practically speaking, infinitely many identities.

R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete Mathematics* (1989)

One of the most interesting test matrices in the toolbox is the Pascal matrix $P_n \in \mathbb{R}^{n\times n}$, defined by

$$p_{ij} = \frac{(i+j-2)!}{(i-1)!(j-1)!} = \binom{i+j-2}{j-1}.$$

The rows of Pascal's triangle appear as anti-diagonals of P_n:

```
P = pascal(6)

P =

     1     1     1     1     1     1
     1     2     3     4     5     6
     1     3     6    10    15    21
     1     4    10    20    35    56
     1     5    15    35    70   126
     1     6    21    56   126   252
```

The earliest reference to the Pascal matrix appears to be by Newman and Todd in 1958 [41] (see also [40, pp. 240–241]); these authors say that the matrix was introduced to them by Rutishauser. The matrix was independently suggested as a test matrix by Caffney in 1963 [4].

Rutishauser [45, Section 8] notes that P_n belongs to the class of moment matrices M whose elements are contour integrals

$$m_{ij} = \int_C z^{j-1}(\overline{z})^{i-1} w(z)\, dz.$$

All moment matrices corresponding to a positive weight function $w(z)$ on a contour C are Hermitian positive definite (as is easily verified by considering the quadratic form $y^* M y$). The choice $C = [0,1]$, with weight function $w(z) = 1$, yields the Hilbert matrix. The Pascal matrix is obtained for C the circle $\{\, z : |z-1| = 1 \,\}$ and $w(z) = (2\pi i(z-1))^{-1}$ (not $w(z) = (2\pi)^{-1}$ as stated in [45]); the change of variable $z = 1 + \exp(i\theta)$ yields a moment integral with a positive weight function.

Remarkably, the Cholesky factor of the Pascal matrix again contains the rows of Pascal's triangle, now arranged column-wise:

```
R = chol(P)

R =

     1     1     1     1     1     1
```

```
     0     1     2     3     4     5
     0     0     1     3     6    10
     0     0     0     1     4    10
     0     0     0     0     1     5
     0     0     0     0     0     1
```

The scaled Cholesky factor $S = R^T \operatorname{diag}(1, -1, 1, -1, \ldots, (-1)^{n+1})$ is returned by `pascal(n,1)`:

```
S = pascal(6, 1)

S =

     1     0     0     0     0     0
     1    -1     0     0     0     0
     1    -2     1     0     0     0
     1    -3     3    -1     0     0
     1    -4     6    -4     1     0
     1    -5    10   -10     5    -1
```

It is involutary: $S^2 = I$. This special property leads us to several more properties of $P = P_n$. First, since $P = SS^T$, $P^{-1} = S^{-T}S^{-1} = S^TS$, and so P^{-1} has integer entries (as is also clear from the fact that $\det(P) = \det(S)^2 = 1$). Moreover,

$$P = SS^T = S(S^TS)S^{-1} = SP^{-1}S^{-1},$$

so P and P^{-1} are similar and hence have the same eigenvalues. In other words, as for the Frank matrix, the eigenvalues appear in reciprocal pairs and the characteristic polynomial has a palindromic coefficient vector. To illustrate:

```
charpoly(P)

ans =

1-351*x+6084*x^2-13869*x^3+6084*x^4-351*x^5+x^6

eig(P)

ans =

    0.0030
    0.0643
    0.4893
    2.0436
   15.5535
  332.8463
```

Since P is symmetric, its eigenvalues are its singular values and so we also have that $\|P\|_2 = \|P^{-1}\|_2$ and $\|P\|_F = \|P^{-1}\|_F$. Now

$$p_{nn} \leq \|P\|_2 \leq (\|P\|_1 \|P\|_\infty)^{1/2} = \left(\tfrac{2n-1}{n}\right) p_{nn},$$

where for the last equality we used a binomial coefficient summation identity from [19, p. 161]. Hence, using Stirling's approximation ($n! \sim \sqrt{2\pi n}(n/e)^n$),

$$\kappa_2(P_n) \sim \left(\frac{(2n)!}{(n!)^2}\right)^2 \sim \frac{16^n}{n\pi}.$$

Thus P_n is exponentially ill-conditioned as $n \to \infty$.

It is worth pointing out that it is not hard to generate symmetric positive definite matrices with determinant 1 and the reciprocal root property. Let $X = ZDZ^{-1}$ where Z is nonsingular and $D = \mathrm{diag}(\pm 1) \neq \pm I$. Then $X^2 = I$ and the matrix $A = X^T X$ has the desired properties. If we choose Z lower triangular then X is the Cholesky factor of A up to a column scaling by $\mathrm{diag}(\pm 1)$.

The Pascal matrix can be made singular simply by subtracting 1 from the (n,n) element. To see this, note that

$$\begin{aligned} P - e_n e_n^T &= SS^T - e_n e_n^T = S(I - (Se_n)(Se_n)^T)S^T \\ &= S\,\mathrm{diag}(1,1,\ldots,1,0)S^T. \end{aligned}$$

This perturbation, $\Delta P = -e_n e_n^T$, is far from being the smallest one that makes P singular, which is $\Delta P_{\mathrm{opt}} = -\lambda_n v_n v_n^T$, where λ_n is the smallest eigenvalue of P and v_n is the corresponding unit eigenvector, for $\|\Delta P_{\mathrm{opt}}\|_2 = \lambda_n = \|P^{-1}\|_2$ is of order $(n!)^2/(2n)!$, as we saw above.

A more subtle property of the Pascal matrix is that it is totally positive. Karlin [34, p. 137] shows that the matrix with elements $\binom{i+j-1}{j}$, $(i,j = 0,1,\ldots)$ is totally positive; the Pascal matrix is a submatrix of this one and hence is also totally positive. From the total positivity it follows that the Pascal matrix has distinct eigenvalues, which, as we already know from the positive definiteness, are real and positive.

$T =$ `pascal(n,2)` is obtained by rotating S clockwise through 90 degrees and multiplying by -1 if n is even:

```
T = pascal(6, 2)

T =

    -1    -1    -1    -1    -1    -1
     5     4     3     2     1     0
   -10    -6    -3    -1     0     0
    10     4     1     0     0     0
    -5    -1     0     0     0     0
     1     0     0     0     0     0
```

It has the surprising property that it is a cube root of the identity, a property noted by Turnbull [53, p. 332]:

```
T*T

ans =
```

```
     0     0     0     0     0     1
     0     0     0     0    -1    -5
     0     0     0     1     4    10
     0     0    -1    -3    -6   -10
     0     1     2     3     4     5
    -1    -1    -1    -1    -1    -1

T*T*T

ans =

     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0
     0     0     0     1     0     0
     0     0     0     0     1     0
     0     0     0     0     0     1
```

Finally, we note that it is trivial to plot an approximation to the Sierpinski gasket [42, Section 2.2] in MATLAB: simply type `spy(rem(pascal(n),2))`. See Figure 7.5. The picture produced by this command is incorrect for large n, however, because the elements of `pascal(n)` become too large to be exactly representable.

7.5. Companion Matrices

The companion matrix associated with the characteristic polynomial

$$\det(A-\lambda I) = (-1)^n(\lambda^n - a_{n-1}\lambda^{n-1} - \cdots - a_0)$$

of $A \in \mathbf{C}^{n\times n}$ is the matrix

$$C = \begin{bmatrix} a_{n-1} & a_{n-2} & \dots & \dots & a_0 \\ 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & \ddots & & 0 \\ \vdots & & \ddots & 0 & 0 \\ 0 & \dots & \dots & 1 & 0 \end{bmatrix}.$$

The toolbox function `compan` computes C, via the call `C = compan(A)`. It is easy to check that C has the same characteristic polynomial as A, and that if λ is an eigenvalue of C then $(\lambda^{n-1}, \lambda^{n-2}, \ldots, \lambda, 1)^T$ is a corresponding eigenvector. Since $C - \lambda I$ has rank at least $n-1$ for any λ, C is non-derogatory, that is, in the Jordan form no eigenvalue appears in more than one Jordan block. It follows that A is similar to C only if A is non-derogatory.

There are no explicit formulae for the eigenvalues of C, but, perhaps surprisingly, the singular values have simple representations [35]:

$$\begin{aligned} \sigma_1^2 &= \tfrac{1}{2}(\alpha + \sqrt{\alpha^2 - 4a_0^2}), \\ \sigma_i^2 &= 1, \qquad 2 \le i \le n-1, \\ \sigma_n^2 &= \tfrac{1}{2}(\alpha - \sqrt{\alpha^2 - 4a_0^2}), \end{aligned}$$

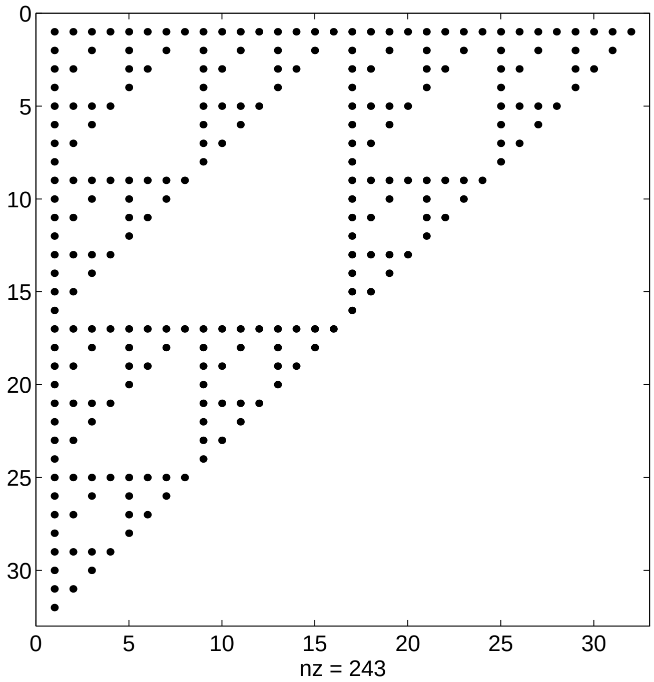

Figure 7.5: `spy(rem(pascal(32),2))`.

where $\alpha = 1 + a_0^2 + \cdots + a_{n-1}^2$.

The `comp` function is a useful means for generating new test matrices from old. For any $A \in \mathbf{C}^{n\times n}$, `compan(A)` is a nonnormal matrix with the same eigenvalues as A (to be precise, `compan(A)` is normal if and only if A is a multiple of the identity matrix).

Companion matrices tend to have interesting pseudospectra, as shown in Figures 7.6 and 7.7. For more information see [50].

7.6. Numerical Linear Algebra

The Test Matrix Toolbox M-files embody some well-known and other not so well-known results from numerical linear algebra.

The function `gfpp` generates $n \times n$ matrices that produce the maximum growth of 2^{n-1} for Gaussian elimination with partial pivoting; these include Wilkinson's classic example [57, p. 212]

```
gfpp(7)
```

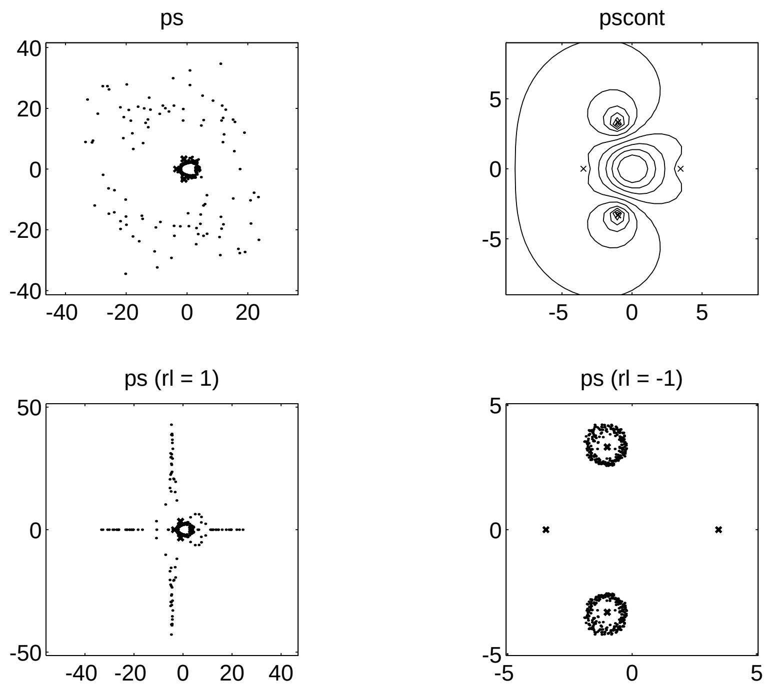

Figure 7.6: Pseudospectra of compan(hadamard(12)).

```
ans =

     1     0     0     0     0     0     1
    -1     1     0     0     0     0     1
    -1    -1     1     0     0     0     1
    -1    -1    -1     1     0     0     1
    -1    -1    -1    -1     1     0     1
    -1    -1    -1    -1    -1     1     1
    -1    -1    -1    -1    -1    -1     1
```

as well as all members of the "2^{n-1} class" described by Higham and Higham [30]. The following extract uses the toolbox routine gecp to evaluate the growth factor for complete pivoting on the Wilkinson matrix.

```
n = 20; A = gfpp(n);
[L, U] = lu(A);    % Partial pivoting.
[max(max(abs(U))) / max(max(abs(A)))  2^(n-1)] % Approximation to growth factor.

ans =

      524288      524288
```

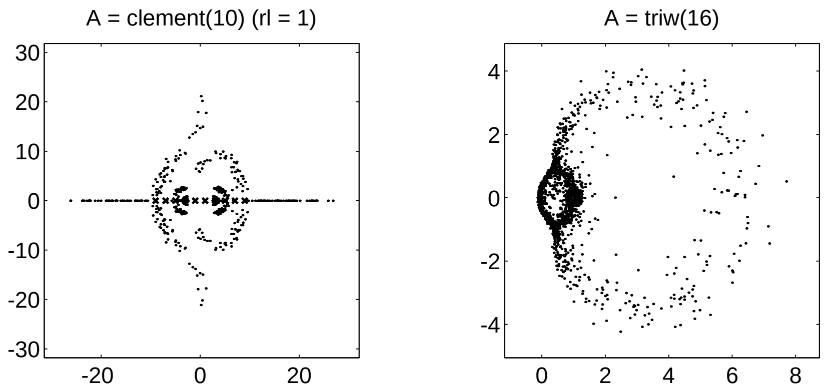

Figure 7.7: Pseudospectra of compan(A).

```
% Complete pivoting using toolbox routine GECP.
[L, U, P, Q, rho] = gecp(A); rho

rho =

     2
```

As the output shows, complete pivoting is perfectly stable for these matrices. However, several of the matrices produced by orthog yield relatively large growth for complete pivoting: growth of order $n/2$ for real data, or n for a particular complex matrix [30].

```
n = 50;
for k = [-2 -1 1 2 3]
    A = orthog(n, k);
    [L, U, P, Q, rho] = gecp(A);
    fprintf(' %g\n', rho)
end
 25.3116
 24.7028
 25.6214
 25.3296
 50
```

```
A = hadamard(64);
[L, U, P, Q, rho] = gecp(A); rho

rho =

    64
```

It is easy to show that complete pivoting suffers growth of at least n for an $n \times n$ Hadamard matrix. However, Hadamard matrices do not exist for all n.

The MATLAB function `rcond` computes an upper bound for $\kappa_1(A)^{-1} = (\|A\|_1\|A^{-1}\|_1)^{-1}$ using the LINPACK condition estimation algorithm. Although this algorithm is very reliable in general, parametrized matrices are known for which it can perform arbitrarily badly [5]. Here are two examples, from the toolbox routine `condex`. The underestimation ratio is approximately the same in both examples, but the second is probably the more serious because `rcond` does not detect any ill-conditioning whatsoever.

```
A = condex(4, 1, 1e8);  format short e
% True          estimate     true/estimate
[cond(A,1)  1/rcond(A)  cond(A,1)*rcond(A)]

ans =

  8.0000e+016  5.6000e+008  1.4286e+008


A = condex(3, 2, 1e8);
[cond(A,1)  1/rcond(A)  cond(A,1)*rcond(A)]

ans =

  6.0000e+008  7.5000e+000  8.0000e+007
```

The QR decomposition with column pivoting of $A \in \mathbf{C}^{m\times n}$ $(m \geq n)$ is a decomposition $A\Pi = Q\begin{bmatrix} R \\ 0 \end{bmatrix}$ where Π is a permutation matrix chosen according to a certain pivoting strategy, Q is orthogonal, and R is upper triangular [18, Section 5.4.1]. This decomposition is often used to estimate the rank of A; in particular, $|r_{nn}|$ provides an upper bound for the smallest singular value $\sigma_{\min}(A)$ of A that is usually at most a factor of 10 too big. Kahan [33] designed a matrix for which $|r_{nn}|$ can be approximately 2^{n-1} times bigger than $\sigma_{\min}(A)$, and which thus shows the fallibility of the QR decomposition with column pivoting for revealing rank. The toolbox function `kahan` generates Kahan's matrix:

```
n = 25;
A = kahan(n, 0.6);
[Q, R, Pi] = qr(A);
norm(Pi-eye(n),1), R(n,n)/min(svd(A))
```

```
ans =

     0

ans =

  1.0638e+006
```

Kahan's matrix $A(\theta)$ is upper triangular and is designed so that Π is the identity in the QR decomposition with column pivoting. In practice, rounding errors can cause Π to differ from the identity for the Kahan matrix, thus nullifying the example (the test on $\Pi - I$ in the above example confirms that Π is indeed the identity here). The toolbox routine adds a small perturbation to the diagonal elements of $A(\theta)$ so that $\Pi = I$ for a range of choices of n and θ. If we set the perturbation in the above example to zero, this is what happens:

```
A = kahan(n, 0.6, 0);  % Third parameter is the diagonal perturbation.
[Q, R, Pi] = qr(A);
norm(Pi-eye(n),1), R(n,n)/min(svd(A))

ans =

     2

ans =

    1.1953
```

The toolbox contains two matrices, one a Toeplitz matrix and the other a Hankel matrix, whose eigenvalues or singular values are related to π:

```
A = parter(10); format long
e = svd(A); [e  e-pi]

ans =

   3.14159265358968  -0.00000000000011
   3.14159265356666  -0.00000000002313
   3.14159265139317  -0.00000000219663
   3.14159252749873  -0.00000012609106
   3.14158778157056  -0.00000487201924
   3.14145930586226  -0.00013334772753
   3.13895248060091  -0.00264017298888
   3.10410768313639  -0.03748497045341
   2.78691548240413  -0.35467717118566
   1.30096907002970  -1.84062358356009
```

```
A = dingdong(10);
e = eig(A); [e  abs(e)-pi/2]

ans =

  -1.57079632679484  -0.00000000000006
  -1.57079632569658  -0.00000000109831
  -1.57079389078528  -0.00000243600962
   1.57079632678333  -0.00000000001157
   1.57079626374937  -0.00000006304553
   1.57072965293113  -0.00006667386377
  -1.56947624030045  -0.00132008649444
   1.55205384156819  -0.01874248522670
  -1.39345774120206  -0.17733858559283
   0.65048453501485  -0.92031179178005
```

Finally, here is an example of the use of the function `matrix` to access the test matrices sequentially, by number. The following piece of code steps through all the square matrices of arbitrary dimension, setting A to each 10×10 matrix in turn (any matrix parameters are at their default values). It evaluates the 2-norm condition number and the ratio $\mu(A) = \max_i |\lambda_i(A)| / \min_i |\lambda_i(A)|$ of the largest to smallest eigenvalue in absolute value.

```
c = []; e = [];
j = 1;
for i=1:matrix(0)
    A = full(matrix(i, 10));
    if norm(skewpart(A),1)  % If not Hermitian...
       c1 = cond(A);
       eg = eig(A);
       e1 = max(abs(eg)) / min(abs(eg));
       % Filter out extremely ill-conditioned matrices.
       if c1 <= 1e10, c(j) = c1; e(j) = e1; j = j + 1; end
    end
end
```

As is well known, $\kappa_2(A)$ can be arbitrarily larger than $\mu(A)$. The plots in Figure 7.8, produced from the vectors `c` and `e` from the above code, confirm that $\kappa_2(A)/\mu(A)$ can be large.

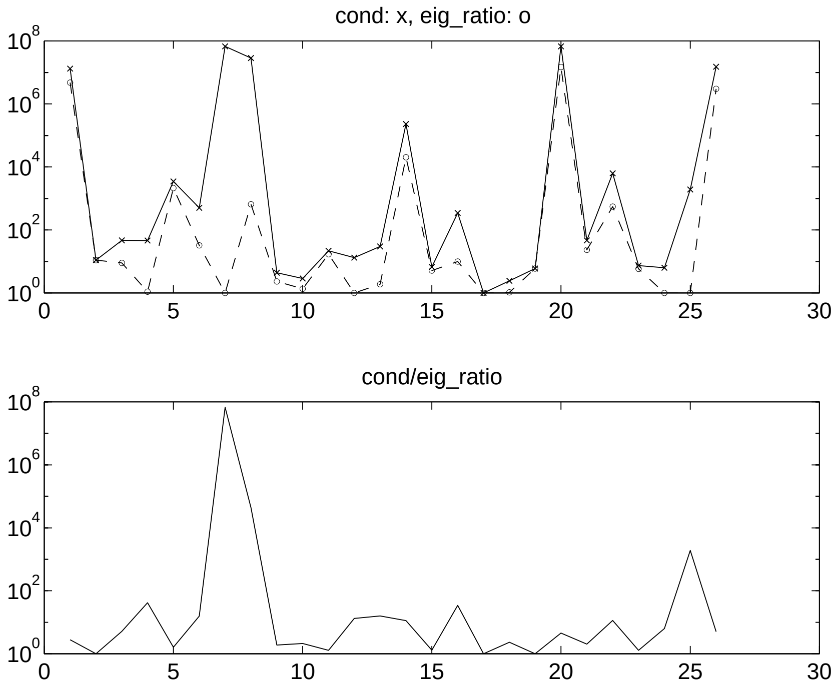

Figure 7.8: Comparison of condition number with extremal eigenvalue ratio.

8. M-File Leading Comment Lines

The demonstration file `tmtdemo` is not listed here.

```
function C = augment(A, alpha)
%AUGMENT  Augmented system matrix.
%         AUGMENT(A) is the square matrix [EYE(m) A; A' ZEROS(n)] of
%         dimension m+n, where A is m-by-n.  It is the symmetric and
%         indefinite coefficient matrix of the augmented system associated
%         with a least squares problem minimize NORM(A*x-b).
%         Special case: if A is a scalar, n say, then AUGMENT(A) is the
%                       same as AUGMENT(RANDN(p,q)) where n = p+q and
%                       p = ROUND(n/2), that is, a random augmented matrix
%                       of dimension n is produced.
%         If a second, scalar argument ALPHA is supplied, then the (1,1)
%         block of A is ALPHA*EYE(m).
%         The eigenvalues of AUGMENT(A) are given in terms of the singular
%         values s(i) of A (where m>n) by
%                  1/2 +/- SQRT( s(i)^2 + 1/4 ),  i=1:n  (2n eigenvalues),
%                  1,  (m-n eigenvalues).
%         If m < n then the first expression provides 2m eigenvalues and the
%         remaining n-m eigenvalues are zero.
%
%         See also SPAUGMENT.

%         Reference:
%         G.H. Golub and C.F. Van Loan, Matrix Computations, Second
%         Edition, Johns Hopkins University Press, Baltimore, Maryland,
%         1989, sec. 5.6.4.
```

```
function A = bandred(A, kl, ku)
%BANDRED  Band reduction by two-sided unitary transformations.
%         B = BANDRED(A, KL, KU) is a matrix unitarily equivalent to A
%         with lower bandwidth KL and upper bandwidth KU
%         (i.e. B(i,j) = 0 if i > j+KL or j > i+KU).
%         The reduction is performed using Householder transformations.
%         If KU is omitted it defaults to KL.

%         Called by RANDSVD.
%         This is a `standard' reduction.  Cf. reduction to bidiagonal form
%         prior to computing the SVD.  This code is a little wasteful in that
%         it computes certain elements which are immediately set to zero!
%
%         Reference:
%         G.H. Golub and C.F. Van Loan, Matrix Computations, second edition,
```

```
%         Johns Hopkins University Press, Baltimore, Maryland, 1989.
%         Section 5.4.3.
```

```
function C = cauchy(x, y)
%CAUCHY  Cauchy matrix.
%        C = CAUCHY(X, Y), where X, Y are N-vectors, is the N-by-N matrix
%        with C(i,j) = 1/(X(i)+Y(j)).   By default, Y = X.
%        Special case: if X is a scalar CAUCHY(X) is the same as CAUCHY(1:X).
%        Explicit formulas are known for DET(C) (which is nonzero if X and Y
%        both have distinct elements) and the elements of INV(C).
%        C is totally positive if 0 < X(1) < ... < X(N) and
%        0 < Y(1) < ... < Y(N).

%        References:
%        D.E. Knuth, The Art of Computer Programming, Volume 1,
%           Fundamental Algorithms, second edition, Addison-Wesley, Reading,
%           Massachusetts, 1973, p. 36.
%        E.E. Tyrtyshnikov, Cauchy-Toeplitz matrices and some applications,
%           Linear Algebra and Appl., 149 (1991), pp. 1-18.
%        O. Taussky and M. Marcus, Eigenvalues of finite matrices, in
%           Survey of Numerical Analysis, J. Todd, ed., McGraw-Hill, New York,
%           pp. 279-313, 1962. (States the totally positive property on p. 295.)
```

```
function C = chebspec(n, k)
%CHEBSPEC  Chebyshev spectral differentiation matrix.
%          C = CHEBSPEC(N, K) is a Chebyshev spectral differentiation
%          matrix of order N.  K = 0 (the default) or 1.
%          For K = 0 (`no boundary conditions'), C is nilpotent, with
%              C^N = 0 and it has the null vector ONES(N,1).
%              C is similar to a Jordan block of size N with eigenvalue zero.
%          For K = 1, C is nonsingular and well-conditioned, and its eigenvalues
%              have negative real parts.
%          For both K, the computed eigenvector matrix X from EIG is
%              ill-conditioned (MESH(REAL(X)) is interesting).

%          References:
%          C. Canuto, M.Y. Hussaini, A. Quarteroni and T.A. Zang, Spectral
%             Methods in Fluid Dynamics, Springer-Verlag, Berlin, 1988; p. 69.
%          L.N. Trefethen and M.R. Trummer, An instability phenomenon in
%             spectral methods, SIAM J. Numer. Anal., 24 (1987), pp. 1008-1023.
%          D. Funaro, Computing the inverse of the Chebyshev collocation
%             derivative, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 1050-1057.
```

```
function C = chebvand(m,p)
%CHEBVAND  Vandermonde-like matrix for the Chebyshev polynomials.
%          C = CHEBVAND(P), where P is a vector, produces the (primal)
%          Chebyshev Vandermonde matrix based on the points P,
%          i.e., C(i,j) = T_{i-1}(P(j)), where T_{i-1} is the Chebyshev
%          polynomial of degree i-1.
%          CHEBVAND(M,P) is a rectangular version of CHEBVAND(P) with M rows.
%          Special case: If P is a scalar then P equally spaced points on
%                        [0,1] are used.

%           Reference:
%           N.J. Higham, Stability analysis of algorithms for solving confluent
%           Vandermonde-like systems, SIAM J. Matrix Anal. Appl., 11 (1990),
%           pp. 23-41.
```

```
function [R, P, I] = cholp(A, pivot)
%CHOLP  Cholesky factorization with pivoting of a pos. semi-definite matrix.
%       [R, P] = CHOLP(A) returns R and a permutation matrix P such that
%       R'*R = P'*A*P.  Only the upper triangular part of A is used.
%       [R, P, I] = CHOLP(A) returns in addition the index I of the last
%       positive diagonal element of R.  The first I rows of R contain
%       the Cholesky factor of A.
%       [R, I] = CHOLP(A, 0) forces P = EYE(SIZE(A)), and therefore produces
%       the same output as R = CHOL(A) when A is positive definite (to
%       within roundoff).

%       This routine is based on the LINPACK routine CCHDC.  It works
%       for both real and complex matrices.
%
%       Reference:
%       G.H. Golub and C.F. Van Loan, Matrix Computations, Second
%       Edition, Johns Hopkins University Press, Baltimore, Maryland,
%       1989, sec. 4.2.9.
```

```
function c = chop(x, t)
%CHOP    Round matrix elements.
%        CHOP(X, t) is the matrix obtained by rounding the elements of X
%        to t significant binary places.
%        Default is t = 24, corresponding to IEEE single precision.
```

```
function A = chow(n, alpha, delta)
%CHOW    Chow matrix - a singular Toeplitz lower Hessenberg matrix.
%        A = CHOW(N, ALPHA, DELTA) is a Toeplitz lower Hessenberg matrix
```

```
%        A = H(ALPHA) + DELTA*EYE, where H(i,j) = ALPHA^(i-j+1).
%        H(ALPHA) has p = FLOOR((N+1)/2) zero eigenvalues, the rest being
%        4*ALPHA*COS( k*PI/(N+2) )^2, k=1:N-p.
%        Defaults: ALPHA = 1, DELTA = 0.

%        References:
%        T.S. Chow, A class of Hessenberg matrices with known
%           eigenvalues and inverses, SIAM Review, 11 (1969), pp. 391-395.
%        G. Fairweather, On the eigenvalues and eigenvectors of a class of
%           Hessenberg matrices, SIAM Review, 13 (1971), pp. 220-221.
```

```
function C = circul(v)
%CIRCUL  Circulant matrix.
%        C = CIRCUL(V) is the circulant matrix whose first row is V.
%        (A circulant matrix has the property that each row is obtained
%        from the previous one by cyclically permuting the entries one step
%        forward; it is a special Toeplitz matrix in which the diagonals
%        `wrap round'.)
%        Special case: if V is a scalar then C = CIRCUL(1:V).
%        The eigensystem of C (N-by-N) is known explicitly.   If t is an Nth
%        root of unity, then the inner product of V with W = [1 t t^2 ... t^N]
%        is an eigenvalue of C, and W(N:-1:1) is an eigenvector of C.

%        Reference:
%        P.J. Davis, Circulant Matrices, John Wiley, 1977.
```

```
function A = clement(n, k)
%CLEMENT   Clement matrix - tridiagonal with zero diagonal entries.
%          CLEMENT(N, K) is a tridiagonal matrix with zero diagonal entries
%          and known eigenvalues.  It is singular if N is odd.  About 64
%          percent of the entries of the inverse are zero.  The eigenvalues
%          are plus and minus the numbers N-1, N-3, N-5, ..., (1 or 0).
%          For K = 0 (the default) the matrix is unsymmetric, while for
%          K = 1 it is symmetric.
%          CLEMENT(N, 1) is diagonally similar to CLEMENT(N).

%          Similar properties hold for TRIDIAG(X,Y,Z) where Y = ZEROS(N,1).
%          The eigenvalues still come in plus/minus pairs but they are not
%          known explicitly.
%
%          References:
%          P.A. Clement, A class of triple-diagonal matrices for test
%             purposes, SIAM Review, 1 (1959), pp. 50-52.
%          O. Taussky and J. Todd, Another look at a matrix of Mark Kac,
%             Linear Algebra and Appl., 150 (1991), pp. 341-360.
```

```
function [U, R, V] = cod(A, tol)
%COD    Complete orthogonal decomposition.
%       [U, R, V] = COD(A, TOL) computes a decomposition A = U*T*V,
%       where U and V are unitary, T = [R 0; 0 0] has the same dimensions as
%       A, and R is upper triangular and nonsingular of dimension rank(A).
%       Rank decisions are made using TOL, which defaults to approximately
%       MAX(SIZE(A))*NORM(A)*EPS.
%       By itself, COD(A, TOL) returns R.

%       Reference:
%       G.H. Golub and C.F. Van Loan, Matrix Computations, Second
%       Edition, Johns Hopkins University Press, Baltimore, Maryland,
%       1989, sec. 5.4.2.
```

```
function C = comp(A, k)
%COMP    Comparison matrices.
%        COMP(A) is DIAG(B) - TRIL(B,-1) - TRIU(B,1), where B = ABS(A).
%        COMP(A, 1) is A with each diagonal element replaced by its
%        absolute value, and each off-diagonal element replaced by minus
%        the absolute value of the largest element in absolute value in
%        its row.  However, if A is triangular COMP(A, 1) is too.
%        COMP(A, 0) is the same as COMP(A).
%        COMP(A) is often denoted by M(A) in the literature.

%        Reference (e.g.):
%        N.J. Higham, A survey of condition number estimation for
%        triangular matrices, SIAM Review, 29 (1987), pp. 575-596.
```

```
function A = compan(p)
%COMPAN  Companion matrix.
%        COMPAN(P) is a companion matrix.  There are three cases.
%        If P is a scalar then COMPAN(P) is the P-by-P matrix COMPAN(1:P+1).
%        If P is an (n+1)-vector, COMPAN(P) is the n-by-n companion matrix
%           whose first row is -P(2:n+1)/P(1).
%        If P is a square matrix, COMPAN(P) is the companion matrix
%           of the characteristic polynomial of P, computed as
%           COMPAN(POLY(P)).

%        References:
%        J.H. Wilkinson, The Algebraic Eigenvalue Problem,
%           Oxford University Press, 1965, p. 12.
%        G.H. Golub and C.F. Van Loan, Matrix Computations, second edition,
%           Johns Hopkins University Press, Baltimore, Maryland, 1989,
%           sec 7.4.6.
```

```
%        C. Kenney and A.J. Laub, Controllability and stability radii for
%          companion form systems, Math. Control Signals Systems, 1 (1988),
%          pp. 239-256. (Gives explicit formulas for the singular values of
%          COMPAN(P).)
```

```
function y = cond(A, p)
%COND   Matrix condition number in 1, 2, Frobenius, or infinity norm.
%       For p = 1, 2, 'fro', inf,  COND(A,p) = NORM(A,p) * NORM(INV(A),p).
%       If p is omitted then p = 2 is used.
%       A may be a rectangular matrix if p = 2; in this case COND(A)
%       is the ratio of the largest singular value of A to the smallest
%       (and hence is infinite if A is rank deficient).
```

```
function A = condex(n, k, theta)
%CONDEX   `Counter-examples' to matrix condition number estimators.
%         CONDEX(N, K, THETA) is a `counter-example' matrix to a condition
%         estimator.  It has order N and scalar parameter THETA (default 100).
%         If N is not equal to the `natural' size of the matrix then
%         the matrix is padded out with an identity matrix to order N.
%         The matrix, its natural size, and the estimator to which it applies
%         are specified by K (default K = 4) as follows:
%             K = 1:   4-by-4,     LINPACK (RCOND)
%             K = 2:   3-by-3,     LINPACK (RCOND)
%             K = 3:   arbitrary,  LINPACK (RCOND) (independent of THETA)
%             K = 4:   N >= 4,     SONEST (Higham 1988)
%         (Note that in practice the K = 4 matrix is not usually a
%          counter-example because of the rounding errors in forming it.)

%         References:
%         A.K. Cline and R.K. Rew, A set of counter-examples to three
%            condition number estimators, SIAM J. Sci. Stat. Comput.,
%            4 (1983), pp. 602-611.
%         N.J. Higham, FORTRAN codes for estimating the one-norm of a real or
%            complex matrix, with applications to condition estimation
%            (Algorithm 674), ACM Trans. Math. Soft., 14 (1988), pp. 381-396.
```

```
function x = cpltaxes(z)
%CPLTAXES   Determine suitable AXIS for plot of complex vector.
%           X = CPLTAXES(Z), where Z is a complex vector,
%           determines a 4-vector X such that AXIS(X) sets axes for a plot
%           of Z that has axes of equal length and leaves a reasonable amount
%           of space around the edge of the plot.

%           Called by FV, GERSH, PS and PSCONT.
```

```
function A = cycol(n, k)
%CYCOL   Matrix whose columns repeat cyclically.
%        A = CYCOL([M N], K) is an M-by-N matrix of the form A = B(1:M,1:N)
%        where B = [C C C...] and C = RANDN(M, K).  Thus A's columns repeat
%        cyclically, and A has rank at most K.   K need not divide N.
%        K defaults to ROUND(N/4).
%        CYCOL(N, K), where N is a scalar, is the same as CYCOL([N N], K).
%
%        This type of matrix can lead to underflow problems for Gaussian
%        elimination: see NA Digest Volume 89, Issue 3 (January 22, 1989).
```

```
function A = dingdong(n)
%DINGDONG  Dingdong matrix - a symmetric Hankel matrix.
%          A = DINGDONG(N) is the symmetric N-by-N Hankel matrix with
%                         A(i,j) = 0.5/(N-i-j+1.5).
%          The eigenvalues of A cluster around PI/2 and -PI/2.

%          Invented by F.N. Ris.
%
%          Reference:
%          J.C. Nash, Compact Numerical Methods for Computers: Linear
%          Algebra and Function Minimisation, second edition, Adam Hilger,
%          Bristol, 1990 (Appendix 1).
```

```
function [c, d, e] = dorr(n, theta)
%DORR  Dorr matrix - diagonally dominant, ill-conditioned, tridiagonal.
%      [C, D, E] = DORR(N, THETA) returns the vectors defining a row diagonally
%      dominant, tridiagonal M-matrix that is ill-conditioned for small
%      values of the parameter THETA >= 0.
%      If only one output parameter is supplied then
%      C = FULL(TRIDIAG(C,D,E)), i.e., the matrix iself is returned.
%      The columns of INV(C) vary greatly in norm.  THETA defaults to 0.01.
%      The amount of diagonal dominance is given by (ignoring rounding errors):
%            COMP(C)*ONES(N,1) = THETA*(N+1)^2 * [1 0 0 ... 0 1]'.

%      Reference:
%      F.W. Dorr, An example of ill-conditioning in the numerical
%      solution of singular perturbation problems, Math. Comp., 25 (1971),
%      pp. 271-283.
```

```
function A = dramadah(n, k)
%DRAMADAH  A (0,1) matrix whose inverse has large integer entries.
%          An anti-Hadamard matrix A is a matrix with elements 0 or 1 for
```

```
%           which MU(A) := NORM(INV(A),'FRO') is maximal.
%           A = DRAMADAH(N, K) is an N-by-N (0,1) matrix for which MU(A) is
%           relatively large, although not necessarily maximal.
%           Available types (the default is K = 1):
%           K = 1: A is Toeplitz, with ABS(DET(A)) = 1, and MU(A) > c(1.75)^N,
%                  where c is a constant.
%           K = 2: A is upper triangular and Toeplitz.
%           The inverses of both types have integer entries.
%
%           Another interesting (0,1) matrix:
%           K = 3: A has maximal determinant among (0,1) lower Hessenberg
%           matrices: det(A) = the n'th Fibonacci number.  A is Toeplitz.
%           The eigenvalues have an interesting distribution in the complex
%           plane.

%           References:
%           R.L. Graham and N.J.A. Sloane, Anti-Hadamard matrices,
%              Linear Algebra and Appl., 62 (1984), pp. 113-137.
%           L. Ching, The maximum determinant of an nxn lower Hessenberg
%              (0,1) matrix, Linear Algebra and Appl., 183 (1993), pp. 147-153.
```

```
function y = dual(x, p)
%DUAL    Dual vector with respect to Holder p-norm.
%        Y = DUAL(X, p), where 1 <= p <= inf, is a vector of unit q-norm
%        that is dual to X with respect to the p-norm, that is,
%        norm(Y, q) = 1 where 1/p + 1/q = 1 and there is
%        equality in the Holder inequality: X'*Y = norm(X, p)norm(Y, q).
%        Special case: DUAL(X), where X >= 1 is a scalar, returns Y such
%                      that 1/X + 1/Y = 1.

%        Called by PNORM.
```

```
function [X, D, s] = eigsens(A)
%EIGSENS   Eigenvalue condition numbers.
%           EIGSENS(A) is a vector of condition numbers for the eigenvalues
%           of A (reciprocals of the Wilkinson s(lambda) numbers).
%           These condition numbers are the reciprocals of the cosines of the
%           angles between the left and right eigenvectors.
%           [V, D, s] = EIGSENS(A) is equivalent to
%                       [V, D] = EIG(A); s = EIGSENS(A);

%           Reference:
%           G.H. Golub and C.F. Van Loan, Matrix Computations, Second
%           Edition, Johns Hopkins University Press, Baltimore, Maryland,
%           1989, sec. 7.2.2.
```

```
function A = fiedler(c)
%FIEDLER  Fiedler matrix - symmetric.
%         A = FIEDLER(C), where C is an n-vector, is the n-by-n symmetric
%         matrix with elements ABS(C(i)-C(j)).
%         Special case: if C is a scalar, then A = FIEDLER(1:C)
%                       (i.e. A(i,j) = ABS(i-j)).
%         Properties:
%           FIEDLER(N) has a dominant positive eigenvalue and all the other
%                      eigenvalues are negative (Szego 1936).
%           Explicit formulas for INV(A) and DET(A) are given in (Todd 1977)
%           and attributed to Fiedler.  These indicate that INV(A) is
%           tridiagonal except for nonzero (1,n) and (n,1) elements.
%           [I think these formulas are valid only if the elements of
%           C are in increasing or decreasing order---NJH.]

%           References:
%           G. Szego, Solution to problem 3705, Amer. Math. Monthly,
%              43 (1936), pp. 246-259.
%           J. Todd, Basic Numerical Mathematics, Vol. 2: Numerical Algebra,
%              Birkhauser, Basel, and Academic Press, New York, 1977, p. 159.
```

```
function A = forsythe(n, alpha, lambda)
%FORSYTHE  Forsythe matrix - a perturbed Jordan block.
%          FORSYTHE(N, ALPHA, LAMBDA) is the N-by-N matrix equal to
%          JORDBLOC(N, LAMBDA) except it has an ALPHA in the (N,1) position.
%          It has the characteristic polynomial
%                  DET(A-t*EYE) = (LAMBDA-t)^N - (-1)^N ALPHA.
%          ALPHA defaults to SQRT(EPS) and LAMBDA to 0.
```

```
function F = frank(n, k)
%FRANK   Frank matrix - ill-conditioned eigenvalues.
%        F = FRANK(N, K) is the Frank matrix of order N.  It is upper
%        Hessenberg with determinant 1.  K = 0 is the default; if K = 1 the
%        elements are reflected about the anti-diagonal (1,N)--(N,1).
%        The eigenvalues of F may be obtained in terms of the zeros of the
%        Hermite polynomials.  They are positive and occur in reciprocal
%        pairs.  Thus if N is odd, 1 is an eigenvalue.
%        F has FLOOR(N/2) ill-conditioned eigenvalues---the smaller ones.

%        For large N, DET(FRANK(N)') comes out far from 1---see Frank (1958)
%        and Wilkinson (1960) for discussions.
%
%        References:
%        W.L. Frank, Computing eigenvalues of complex matrices by determinant
```

```
%           evaluation and by methods of Danilewski and Wielandt, J. Soc.
%           Indust. Appl. Math., 6 (1958), pp. 378-392 (see pp. 385, 388).
%        G.H. Golub and J.H. Wilkinson, Ill-conditioned eigensystems and the
%           computation of the Jordan canonical form, SIAM Review, 18 (1976),
%             pp. 578-619 (Section 13).
%        H. Rutishauser, On test matrices, Programmation en Mathematiques
%           Numeriques, Editions Centre Nat. Recherche Sci., Paris, 165,
%           1966, pp. 349-365.  Section 9.
%        J.H. Wilkinson, Error analysis of floating-point computation,
%           Numer. Math., 2 (1960), pp. 319-340 (Section 8).
%        J.H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford University
%           Press, 1965 (pp. 92-93).
%        The next two references give details of the eigensystem, as does
%        Rutishauser (see above).
%        P.J. Eberlein, A note on the matrices denoted by B_n, SIAM J. Appl.
%           Math., 20 (1971), pp. 87-92.
%        J.M. Varah, A generalization of the Frank matrix, SIAM J. Sci. Stat.
%           Comput., 7 (1986), pp. 835-839.
```

```
function [f, e] = fv(B, nk, thmax, noplot)
%FV      Field of values (or numerical range).
%        FV(A, NK, THMAX) evaluates and plots the field of values of the
%        NK largest leading principal submatrices of A, using THMAX
%        equally spaced angles in the complex plane.
%        The defaults are NK = 1 and THMAX = 16.
%        (For a `publication quality' picture, set THMAX higher, say 32.)
%        The eigenvalues of A are displayed as `x'.
%        Alternative usage: [F, E] = FV(A, NK, THMAX, 1) suppresses the
%        plot and returns the field of values plot data in F, with A's
%        eigenvalues in E.   Note that NORM(F,INF) approximates the
%        numerical radius,
%                  max {abs(z): z is in the field of values of A}.

%        Theory:
%        Field of values FV(A) = set of all Rayleigh quotients. FV(A) is a
%        convex set containing the eigenvalues of A.  When A is normal FV(A) is
%        the convex hull of the eigenvalues of A (but not vice versa).
%                z = x'Ax/(x'x),  z' = x'A'x/(x'x)
%                => REAL(z) = x'Hx/(x'x),   H = (A+A')/2
%        so      MIN(EIG(H)) <= REAL(z) <= MAX(EIG(H))
%        with equality for x = corresponding eigenvectors of H.  For these x,
%        RQ(A,x) is on the boundary of FV(A).
%
%        Based on an original routine by A. Ruhe.
%
```

```
%       References:
%       R.A. Horn and C.R. Johnson, Topics in Matrix Analysis, Cambridge
%            University Press, 1991, Section 1.5.
%       A.S. Householder, The Theory of Matrices in Numerical Analysis,
%            Blaisdell, New York, 1964, Section 3.3.
%       C.R. Johnson, Numerical determination of the field of values of a
%            general complex matrix, SIAM J. Numer. Anal., 15 (1978),
%            pp. 595-602.
```

```
function [A, e] = gallery(n)
%GALLERY    Famous, and not so famous, test matrices.
%       A = GALLERY(N) is an N-by-N matrix with some special property.
%       The following values of N are currently available:
%         N = 3 is badly conditioned.
%         N = 4 is the Wilson matrix.  Symmetric pos def, integer inverse.
%         N = 5 is an interesting eigenvalue problem: defective, nilpotent.
%         N = 8 is the Rosser matrix, a classic symmetric eigenvalue problem.
%               [A, e] = GALLERY(8) returns the exact eigenvalues in e.
%         N = 21 is Wilkinson's tridiagonal W21+, another eigenvalue problem.

%       Original version supplied with MATLAB.  Modified by N.J. Higham.
%
%       References:
%       J.R. Westlake, A Handbook of Numerical Matrix Inversion and Solution
%          of Linear Equations, John Wiley, New York, 1968.
%       J.H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford University
%          Press, 1965.
```

```
function [L, U, rho] = ge(A)
%GE     Gaussian elimination without pivoting.
%       [L, U, RHO] = GE(A) computes the factorization A = L*U,
%       where L is unit lower triangular and U is upper triangular.
%       RHO is the growth factor.
%       By itself, GE(A) returns the final reduced matrix from the
%       elimination containing both L and U.
```

```
function A = gearm(n, i, j)
%GEARM   Gear matrix.
%        A = GEARM(N,I,J) is the N-by-N matrix with ones on the sub- and
%        super-diagonals, SIGN(I) in the (1,ABS(I)) position, SIGN(J)
%        in the (N,N+1-ABS(J)) position, and zeros everywhere else.
%        Defaults: I = N, j = -N.
%        All eigenvalues are of the form 2*COS(a) and the eigenvectors
```

```
%        are of the form [SIN(w+a), SIN(w+2a), ..., SIN(w+Na)].
%        The values of a and w are given in the reference below.
%        A can have double and triple eigenvalues and can be defective.
%        GEARM(N) is singular.

%        (GEAR is a Simulink function, hence GEARM for Gear matrix.)
%        Reference:
%        C.W. Gear, A simple set of test matrices for eigenvalue programs,
%        Math. Comp., 23 (1969), pp. 119-125.
```

```
function [L, U, P, Q, rho] = gecp(A)
%GECP   Gaussian elimination with complete pivoting.
%       [L, U, P, Q, RHO] = GECP(A) computes the factorization P*A*Q = L*U,
%       where L is unit lower triangular, U is upper triangular,
%       and P and Q are permutation matrices.  RHO is the growth factor.
%       By itself, GECP(A) returns the final reduced matrix from the
%       elimination containing both L and U.
```

```
function  [G, e] = gersh(A, noplot)
%GERSH    Gershgorin disks.
%         GERSH(A) draws the Gershgorin disks for the matrix A.
%         The eigenvalues are plotted as crosses `x'.
%         Alternative usage: [G, E] = GERSH(A, 1) suppresses the plot
%         and returns the data in G, with A's eigenvalues in E.
%
%         Try GERSH(LESP(N)) and GERSH(SMOKE(N,1)).
```

```
function A = gfpp(T, c)
%GFPP   Matrix giving maximal growth factor for Gaussian elim. with pivoting.
%       GFPP(T) is a matrix of order N for which Gaussian elimination
%       with partial pivoting yields a growth factor 2^(N-1).
%       T is an arbitrary nonsingular upper triangular matrix of order N-1.
%       GFPP(T, C) sets all the multipliers to C  (0 <= C <= 1)
%       and gives growth factor (1+C)^(N-1).
%       GFPP(N, C) (a special case) is the same as GFPP(EYE(N-1), C) and
%       generates the well-known example of Wilkinson.

%       Reference:
%       N.J. Higham and D.J. Higham, Large growth factors in
%       Gaussian elimination with pivoting, SIAM J. Matrix Analysis and
%       Appl., 10 (1989), pp. 155-164.
```

```
function G = grcar(n, k)
%GRCAR     Grcar matrix - a Toeplitz matrix with sensitive eigenvalues.
%          GRCAR(N, K) is an N-by-N matrix with -1s on the
%          subdiagonal, 1s on the diagonal, and K superdiagonals of 1s.
%          The default is K = 3.  The eigenvalues of this matrix form an
%          interesting pattern in the complex plane (try PS(GRCAR(32))).

%          References:
%          J.F. Grcar, Operator coefficient methods for linear equations,
%               Report SAND89-8691, Sandia National Laboratories, Albuquerque,
%               New Mexico, 1989 (Appendix 2).
%          N.M. Nachtigal, L. Reichel and L.N. Trefethen, A hybrid GMRES
%               algorithm for nonsymmetric linear systems, SIAM J. Matrix Anal.
%               Appl., 13 (1992), pp. 796-825.
```

```
function H = hadamard(n)
%HADAMARD  Hadamard matrix.
%          HADAMARD(N) is a Hadamard matrix of order N, that is,
%          a matrix H with elements 1 or -1 such that H*H' = N*EYE(N).
%          An N-by-N Hadamard matrix with N>2 exists only if REM(N,4) = 0.
%          This function handles only the cases where N, N/12 or N/20
%          is a power of 2.

%          Reference:
%          S.W. Golomb and L.D. Baumert, The search for Hadamard matrices,
%          Amer. Math. Monthly, 70 (1963) pp. 12-17.
%
%          History:
%          NJH (11/14/91), revised by CBM, 6/24/92,
%          comment lines revised by NJH, August 1993.
```

```
function A = hanowa(n, d)
%HANOWA  A matrix whose eigenvalues lie on a vertical line in the complex plane.
%        HANOWA(N, d) is the N-by-N block 2x2 matrix (thus N = 2M must be even)
%                      [d*EYE(M)   -DIAG(1:M)
%                       DIAG(1:M)   d*EYE(M)]
%        It has complex eigenvalues lambda(k) = d +/- k*i  (1 <= k <= M).
%        Parameter d defaults to -1.

%        Reference:
%        E. Hairer, S.P. Norsett and G. Wanner, Solving Ordinary
%        Differential Equations I: Nonstiff Problems, Springer-Verlag,
%        Berlin, 1987. (pp. 86-87)
```

```
function H = hilb(n)
%HILB   Hilbert matrix.
%       HILB(N) is the N-by-N matrix with elements 1/(i+j-1).
%       It is a famous example of a badly conditioned matrix.
%       COND(HILB(N)) grows like EXP(3.5*N).
%       See INVHILB (standard MATLAB routine) for the exact inverse, which
%       has integer entries.
%       HILB(N) is symmetric positive definite, totally positive, and a
%       Hankel matrix.

%       References:
%       M.-D. Choi, Tricks or treats with the Hilbert matrix, Amer. Math.
%           Monthly, 90 (1983), pp. 301-312.
%       M. Newman and J. Todd, The evaluation of matrix inversion
%           programs, J. Soc. Indust. Appl. Math., 6 (1958), pp. 466-476.
%       D.E. Knuth, The Art of Computer Programming,
%           Volume 1, Fundamental Algorithms, second edition, Addison-Wesley,
%           Reading, Massachusetts, 1973, p. 37.
```

```
function [v, beta] = house(x)
%HOUSE   Householder matrix.
%        If [v, beta] = HOUSE(x) then H = EYE - beta*v*v' is a Householder
%        matrix such that Hx = -sign(x(1))*norm(x)*e_1.
%        NB: If x = 0 then v = 0, beta = 1 is returned.
%            x can be real or complex.
%            sign(x) := exp(i*arg(x)) ( = x./abs(x) when x ~= 0).

%        Theory: (textbook references Golub & Van Loan 1989, 38-43;
%                 Stewart 1973, 231-234, 262; Wilkinson 1965, 48-50).
%        Hx = y: (I - beta*v*v')x = -s*e_1.
%        Must have |s| = norm(x), v = x+s*e_1, and
%        x'y = x'Hx =(x'Hx)' real => arg(s) = arg(x(1)).
%        So take s = sign(x(1))*norm(x) (which avoids cancellation).
%        v'v = (x(1)+s)^2 + x(2)^2 + ... + x(n)^2
%            = 2*norm(x)*(norm(x) + |x(1)|).
%
%        References:
%        G.H. Golub and C.F. Van Loan, Matrix Computations, second edition,
%           Johns Hopkins University Press, Baltimore, Maryland, 1989.
%        G.W. Stewart, Introduction to Matrix Computations, Academic Press,
%           New York, 1973,
%        J.H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford University
%           Press, 1965.
```

```
function A = invhess(x, y)
%INVHESS  Inverse of an upper Hessenberg matrix.
%         INVHESS(X, Y), where X is an N-vector and Y an N-1 vector,
%         is the matrix whose lower triangle agrees with that of
%         ONES(N,1)*X' and whose strict upper triangle agrees with
%         that of [1 Y]*ONES(1,N).
%         The matrix is nonsingular if X(1) ~= 0 and X(i+1) ~= Y(i)
%         for all i, and its inverse is an upper Hessenberg matrix.
%         If Y is omitted it defaults to -X(1:N-1).
%         Special case: if X is a scalar INVHESS(X) is the same as
%         INVHESS(1:X).

%         References:
%         F.N. Valvi and V.S. Geroyannis, Analytic inverses and
%             determinants for a class of matrices, IMA Journal of Numerical
%             Analysis, 7 (1987), pp. 123-128.
%         W.-L. Cao and W.J. Stewart, A note on inverses of Hessenberg-like
%             matrices, Linear Algebra and Appl., 76 (1986), pp. 233-240.
%         Y. Ikebe, On inverses of Hessenberg matrices, Linear Algebra and
%             Appl., 24 (1979), pp. 93-97.
%         P. Rozsa, On the inverse of band matrices, Integral Equations and
%             Operator Theory, 10 (1987), pp. 82-95.
```

```
function A = invol(n)
%INVOL   An involutory matrix.
%        A = INVOL(N) is an N-by-N involutory (A*A = EYE(N)) and
%        ill-conditioned matrix.
%        It is a diagonally scaled version of HILB(N).
%        NB: B = (EYE(N)-A)/2 and B = (EYE(N)+A)/2 are idempotent (B*B = B).

%        Reference:
%        A.S. Householder and J.A. Carpenter, The singular values
%        of involutory and of idempotent matrices, Numer. Math. 5 (1963),
%        pp. 234-237.
```

```
function [A, detA] = ipjfact(n, k)
%IPJFACT   A Hankel matrix with factorial elements.
%          A = IPJFACT(N, K) is the matrix with
%                    A(i,j) = (i+j)!    (K = 0, default)
%                    A(i,j) = 1/(i+j)!  (K = 1)
%          Both are Hankel matrices.
%          The determinant and inverse are known explicitly.
%          If a second output argument is present, d = DET(A) is returned:
%          [A, d] = IPJFACT(N, K);
```

```
%          Suggested by P. R. Graves-Morris.
%
%          Reference:
%          M.J.C. Gover, The explicit inverse of factorial Hankel matrices,
%          Dept. of Mathematics, University of Bradford, 1993.
```

```
function J = jordbloc(n, lambda)
%JORDBLOC  Jordan block.
%          JORDBLOC(N, LAMBDA) is the N-by-N Jordan block with eigenvalue
%          LAMBDA.  LAMBDA = 1 is the default.
```

```
function U = kahan(n, theta, pert)
%KAHAN  Kahan matrix - upper trapezoidal.
%       KAHAN(N, THETA) is an upper trapezoidal matrix
%       that has some interesting properties regarding estimation of
%       condition and rank.
%       The matrix is N-by-N unless N is a 2-vector, in which case it
%       is N(1)-by-N(2).
%       The parameter THETA defaults to 1.2.
%       The useful range of THETA is 0 < THETA < PI.
%
%       To ensure that the QR factorization with column pivoting does not
%       interchange columns in the presence of rounding errors, the diagonal
%       is perturbed by PERT*EPS*diag( [N:-1:1] ).
%       The default is PERT = 25, which ensures no interchanges for KAHAN(N)
%       up to at least N = 90 in IEEE arithmetic.
%       KAHAN(N, THETA, PERT) uses the given value of PERT.

%       The inverse of KAHAN(N, THETA) is known explicitly: see
%       Higham (1987, p. 588), for example.
%       The diagonal perturbation was suggested by Christian Bischof.
%
%       References:
%       W. Kahan, Numerical linear algebra, Canadian Math. Bulletin,
%          9 (1966), pp. 757-801.
%       N.J. Higham, A survey of condition number estimation for
%          triangular matrices, SIAM Review, 29 (1987), pp. 575-596.
```

```
function A = kms(n, rho)
%KMS   Kac-Murdock-Szego Toeplitz matrix.
%      A = KMS(N, RHO) is the N-by-N Kac-Murdock-Szego Toeplitz matrix with
%      A(i,j) = RHO^(ABS((i-j))) (for real RHO).
```

```
%       If RHO is complex, then the same formula holds except that elements
%       below the diagonal are conjugated.
%       RHO defaults to 0.5.
%       Properties:
%          A has an LDL' factorization with
%                   L = INV(TRIW(N,-RHO,1)'),
%                   D(i,i) = (1-ABS(RHO)^2)*EYE(N) except D(1,1) = 1.
%          A is positive definite if and only if 0 < ABS(RHO) < 1.
%          INV(A) is tridiagonal.

%        Reference:
%        W.F. Trench, Numerical solution of the eigenvalue problem
%        for Hermitian Toeplitz matrices, SIAM J. Matrix Analysis and Appl.,
%        10 (1989), pp. 135-146 (and see the references therein).
```

```
function K = kron(A,B)
%KRON Kronecker tensor product.
% KRON(X,Y) is the Kronecker tensor product of X and Y.
% The result is a large matrix formed by taking all possible
% products between the elements of X and those of Y.   For
% example, if X is 2 by 3, then KRON(X,Y) is
%
%    [ X(1,1)*Y  X(1,2)*Y  X(1,3)*Y
%      X(2,1)*Y  X(2,2)*Y  X(2,3)*Y ]
%
% If either X or Y is sparse, only nonzero elements are multiplied
% in the computation, and the result is sparse.

% Full: J. N. Little, 4-21-85.
% Sparse: T. R. Gardos (Georgia Tech), 5-4-93.
% Copyright (c) 1984-93 by The MathWorks, Inc.
%
%        This version not strictly part of the toolbox - included here
%        because version supplied with Matlab 4.0 is not sparse aware.
```

```
function B = krylov(A, x, j)
%KRYLOV    Krylov matrix.
%          KRYLOV(A, x, j) is the Krylov matrix
%               [x, Ax, A^2x, ..., A^(j-1)x],
%          where A is an n-by-n matrix and x is an n-vector.
%          Defaults: x = ONES(n,1), j = n.
%          KRYLOV(n) is the same as KRYLOV(RANDN(n)).

%          Reference:
```

```
%          G.H. Golub and C.F. Van Loan, Matrix Computations, second edition,
%          Johns Hopkins University Press, Baltimore, Maryland, 1989, p. 369.
```

```
function A = lauchli(n, mu)
%LAUCHLI   Lauchli matrix - rectangular.
%          LAUCHLI(N, MU) is the (N+1)-by-N matrix [ONES(1,N); MU*EYE(N))].
%          It is a well-known example in least squares and other problems
%          that indicates the dangers of forming A'*A.
%          MU defaults to SQRT(EPS).

%          Reference:
%          P. Lauchli, Jordan-Elimination und Ausgleichung nach
%          kleinsten Quadraten, Numer. Math, 3 (1961), pp. 226-240.
```

```
function A = lehmer(n)
%LEHMER  Lehmer matrix - symmetric positive definite.
%        A = LEHMER(N) is the symmetric positive definite N-by-N matrix with
%                       A(i,j) = i/j for j >= i.
%        A is totally nonnegative.  INV(A) is tridiagonal, and explicit
%        formulas are known for its entries.
%        N <= COND(A) <= 4*N*N.

%        References:
%        M. Newman and J. Todd, The evaluation of matrix inversion
%           programs, J. Soc. Indust. Appl. Math., 6 (1958), pp. 466-476.
%        Solutions to problem E710 (proposed by D.H. Lehmer): The inverse
%           of a matrix, Amer. Math. Monthly, 53 (1946), pp. 534-535.
%        J. Todd, Basic Numerical Mathematics, Vol. 2: Numerical Algebra,
%           Birkhauser, Basel, and Academic Press, New York, 1977, p. 154.
```

```
function T = lesp(n)
%LESP   A tridiagonal matrix with real, sensitive eigenvalues.
%       LESP(N) is an N-by-N matrix whose eigenvalues are real and smoothly
%       distributed in the interval approximately [-2*N-3.5, -4.5].
%       The sensitivities of the eigenvalues increase exponentially as
%       the eigenvalues grow more negative.
%       The matrix is similar to the symmetric tridiagonal matrix with
%       the same diagonal entries and with off-diagonal entries 1,
%       via a similarity transformation with D = diag(1!,2!,...,N!).

%       References:
%       H.W.J. Lenferink and M.N. Spijker, On the use of stability regions in
%            the numerical analysis of initial value problems,
```

```
%           Math. Comp., 57 (1991), pp. 221-237.
%       L.N. Trefethen, Pseudospectra of matrices, in Numerical Analysis 1991,
%           Proceedings of the 14th Dundee Conference,
%           D.F. Griffiths and G.A. Watson, eds, Pitman Research Notes in
%           Mathematics, volume 260, Longman Scientific and Technical, Essex,
%           UK, 1992, pp. 234-266.
```

```
function A = lotkin(n)
%LOTKIN  Lotkin matrix.
%        A = LOTKIN(N) is the Hilbert matrix with its first row altered to
%        all ones.  A is unsymmetric, ill-conditioned, and has many negative
%        eigenvalues of small magnitude.
%        The inverse has integer entries and is known explicitly.

%        Reference:
%        M. Lotkin, A set of test matrices, MTAC, 9 (1955), pp. 153-161.
```

```
function A = makejcf(n, e, m, X)
%MAKEJCF   A matrix with given Jordan canonical form.
%          MAKEJCF(N, E, M) is a matrix having the Jordan canonical form
%          whose i'th Jordan block is of dimension M(i) with eigenvalue E(i),
%          and where N = SUM(M).
%          Defaults: E = 1:N, M = ONES(SIZE(E)) with M(1) so that SUM(M) = N.
%          The matrix is constructed by applying a random similarity
%          transformation to the Jordan form.
%          Alternatively, the matrix used in the similarity transformation
%          can be specified as a fifth parameter.
%          In particular, MAKEJCF(N, E, M, EYE(N)) returns the Jordan form
%          itself.
%          NB: The JCF is very sensitive to rounding errors.
```

```
function A = matrix(k, n)
%MATRIX    Test Matrix Toolbox information and matrix access by number.
%          MATRIX(K, N) is the N-by-N instance of the matrix number K in
%          the Test Matrix Toolbox (including some of the matrices provided
%          with MATLAB), with all other parameters set to their default.
%          N.B. Only those matrices which take an arbitrary dimension N
%               are included (thus GALLERY is omitted, for example).
%          MATRIX(K) is a string containing the name of the K'th matrix.
%          MATRIX(0) is the number of matrices, i.e. the upper limit for K.
%          Thus to set A to each N-by-N test matrix in turn use a loop like
%                for k=1:matrix(0)
%                    A = matrix(k, N);
```

```
%                  Aname = matrix(k);   % The name of the matrix
%              end
%          MATRIX(-1) returns the version number and date of the toolbox.
%          MATRIX with no arguments lists the names of the M-files in the
%          collection.

%          References:
%          N.J. Higham, The Test Matrix Toolbox for Matlab, Numerical
%             Analysis Report No. 237, Department of Mathematics,
%             University of Manchester, England, December 1993.
%          N.J. Higham, Algorithm 694: A collection of test matrices in
%             MATLAB, ACM Trans. Math. Soft., 17 (1991), pp. 289-305.
%
%          Matrices omitted are: gallery, hadamard, hanowa, lauchli,
%          neumann, wathen, wilk.
%          Matrices provided with MATLAB that are included here: invhilb,
%          magic.
```

```
function S = matsignt(T)
%MATSIGNT    Matrix sign function of a triangular matrix.
%            S = MATSIGN(T) computes the matrix sign function S of the
%            upper triangular matrix T using a recurrence.

%            Adapted from FUNM.  Called by SIGNM.
```

```
function A = minij(n)
%MINIJ   Symmetric positive definite matrix MIN(i,j).
%        A = MINIJ(N) is the N-by-N symmetric positive definite matrix with
%        A(i,j) = MIN(i,j).
%        Properties, variations:
%        INV(A) is tridiagonal: it is minus the second difference matrix
%                    except its (N,N) element is 1.
%        2*A-ONES(A) (Givens' matrix) has tridiagonal inverse and
%                    eigenvalues .5*sec^2([2r-1)PI/4N], r=1:N.
%        (N+1)ONES(A)-A has elements MAX(i,j), and also has a tridiagonal
%                    inverse.

%        References:
%        J. Todd, Basic Numerical Mathematics, Vol. 2: Numerical Algebra,
%           Birkhauser, Basel, and Academic Press, New York, 1977, p. 158.
%        D.E. Rutherford, Some continuant determinants arising in physics and
%           chemistry---II, Proc. Royal Soc. Edin., 63, A (1952), pp. 232-241.
%           (For the eigenvalues of Givens' matrix.)
```

```
function A = moler(n, alpha)
%MOLER   Moler matrix - symmetric positive definite.
%        A = MOLER(N, ALPHA) is the symmetric positive definite N-by-N matrix
%        U'*U where U = TRIW(N, ALPHA).
%        For ALPHA = -1 (the default) A(i,j) = MIN(i,j)-2, A(i,i) = i.
%        A has one small eigenvalue.

%        Nash (1990) attributes the ALPHA = -1 matrix to Moler.
%
%        Reference:
%        J.C. Nash, Compact Numerical Methods for Computers: Linear
%        Algebra and Function Minimisation, second edition, Adam Hilger,
%        Bristol, 1990 (Appendix 1).
```

```
function [A, T] = neumann(n)
%NEUMANN  Singular matrix from the discrete Neumann problem (sparse).
%         NEUMANN(N) is the singular, row diagonally dominant matrix resulting
%         from discretizing the Neumann problem with the usual five point
%         operator on a regular mesh.
%         It has a one-dimensional null space with null vector ONES(N,1).
%         The dimension N should be a perfect square, or else a 2-vector,
%         in which case the dimension of the matrix is N(1)*N(2).

%         Reference:
%         R.J. Plemmons, Regular splittings and the discrete Neumann
%         problem, Numer. Math., 25 (1976), pp. 153-161.
```

```
function H = ohess(x)
%OHESS  Random, orthogonal upper Hessenberg matrix.
%       H = OHESS(N) is an N-by-N real, random, orthogonal
%       upper Hessenberg matrix.
%       Alternatively, H = OHESS(X), where X is an arbitrary real
%       N-vector (N > 1) constructs H non-randomly using the elements
%       of X as parameters.
%       In both cases H is constructed via a product of N-1 Givens rotations.

%       Note: See Gragg (1986) for how to represent an N-by-N (complex)
%       unitary Hessenberg matrix with positive subdiagonal elements in terms
%       of 2N-1 real parameters (the Schur parametrization).
%       This M-file handles the real case only and is intended simply as a
%       convenient way to generate random or non-random orthogonal Hessenberg
%       matrices.
%
%       Reference:
```

```
%       W.B. Gragg, The QR algorithm for unitary Hessenberg matrices,
%       J. Comp. Appl. Math., 16 (1986), pp. 1-8.
```

```
function Q = orthog(n, k)
%ORTHOG Orthogonal and nearly orthogonal matrices.
%       Q = ORTHOG(N, K) selects the K'th type of matrix of order N.
%       K > 0 for exactly orthogonal matrices, K < 0 for diagonal scalings of
%       orthogonal matrices.
%       Available types: (K = 1 is the default)
%       K = 1:  Q(i,j) = SQRT(2/(n+1)) * SIN( i*j*PI/(n+1) )
%               Symmetric eigenvector matrix for second difference matrix.
%       K = 2:  Q(i,j) = 2/SQRT(2*n+1)) * SIN( 2*i*j*PI/(2*n+1) )
%               Symmetric.
%       K = 3:  Q(r,s) = EXP(2*PI*i*(r-1)*(s-1)/n) / SQRT(n)  (i=SQRT(-1))
%               Unitary, the Fourier matrix.  Q^4 is the identity.
%               This is essentially the same matrix as FFT(EYE(N))/SQRT(N)!
%       K = 4:  Helmert matrix: a permutation of a lower Hessenberg matrix,
%               whose first row is ONES(1:N)/SQRT(N).
%       K = 5:  Q(i,j) = SIN( 2*PI*(i-1)*(j-1)/n ) + COS( 2*PI*(i-1)*(j-1)/n ).
%               Symmetric matrix arising in the Hartley transform.
%       K = -1: Q(i,j) = COS( (i-1)*(j-1)*PI/(n-1) )
%               Chebyshev Vandermonde-like matrix, based on extrema of T(n-1).
%       K = -2: Q(i,j) = COS( (i-1)*(j-1/2)*PI/n) )
%               Chebyshev Vandermonde-like matrix, based on zeros of T(n).

%       References:
%       N.J. Higham and D.J. Higham, Large growth factors in Gaussian
%            elimination with pivoting, SIAM J. Matrix Analysis and  Appl.,
%            10 (1989), pp. 155-164.
%       P. Morton, On the eigenvectors of Schur's matrix, J. Number Theory,
%            12 (1980), pp. 122-127. (Re. ORTHOG(N, 3))
%       H.O. Lancaster, The Helmert Matrices, Amer. Math. Monthly, 72 (1965),
%            pp. 4-12.
%       D. Bini and P. Favati, On a matrix algebra related to the discrete
%            Hartley transform, SIAM J. Matrix Anal. Appl., 14 (1993),
%            pp. 500-507.
```

```
function A = parter(n)
%PARTER    Parter matrix - a Toeplitz matrix with singular values near PI.
%          PARTER(N) is the matrix with (i,j) element 1/(i-j+0.5).
%          It is a Cauchy matrix and a Toeplitz matrix.

%          At the Second SIAM Conference on Linear Algebra, Raleigh, N.C.,
%          1985, Cleve Moler noted that most of the singular values of
```

```
%          PARTER(N) are very close to PI.  An explanation of the phenomenon
%          was given by Parter; see also the paper by Tyrtyshnikov.
%
%          References:
%          The MathWorks Newsletter, Volume 1, Issue 1, March 1986, page 2.
%          S.V. Parter, On the distribution of the singular values of Toeplitz
%               matrices, Linear Algebra and Appl., 80 (1986), pp. 115-130.
%          E.E. Tyrtyshnikov, Cauchy-Toeplitz matrices and some applications,
%               Linear Algebra and Appl., 149 (1991), pp. 1-18.
```

```
function P = pascal(n, k)
%PASCAL  Pascal matrix.
%        P = PASCAL(N) is the Pascal matrix of order N: a symmetric positive
%        definite matrix with integer entries taken from Pascal's
%        triangle.
%        The Pascal matrix is totally positive and its inverse has
%        integer entries.  Its eigenvalues occur in reciprocal pairs.
%        COND(P) is approximately 16^N/(N*PI) for large N.
%        PASCAL(N,1) is the lower triangular Cholesky factor (up to signs
%        of columns) of the Pascal matrix.   It is involutary (is its own
%        inverse).
%        PASCAL(N,2) is a transposed and permuted version of PASCAL(N,1)
%        which is a cube root of the identity.

%        References:
%        R. Brawer and M. Pirovino, The linear algebra of the Pascal matrix,
%           Linear Algebra and Appl., 174 (1992), pp. 13-23 (this paper
%           gives a factorization of L = PASCAL(N,1) and a formula for the
%           elements of L^k).
%        S. Karlin, Total Positivity, Volume 1, Stanford University Press,
%           1968.  (Page 137: shows i+j-1 choose j is TP (i,j=0,1,...).
%                   PASCAL(N) is a submatrix of this matrix.)
%        M. Newman and J. Todd, The evaluation of matrix inversion programs,
%           J. Soc. Indust. Appl. Math., 6(4):466--476, 1958.
%        H. Rutishauser, On test matrices, Programmation en Mathematiques
%           Numeriques, Editions Centre Nat. Recherche Sci., Paris, 165,
%           1966, pp. 349-365.  (Gives an integral formula for the
%           elements of PASCAL(N).)
%        J. Todd, Basic Numerical Mathematics, Vol. 2: Numerical Algebra,
%           Birkhauser, Basel, and Academic Press, New York, 1977, p. 172.
%        H.W. Turnbull, The Theory of Determinants, Matrices, and Invariants,
%           Blackie, London and Glasgow, 1929.  (PASCAL(N,2) on page 332.)
```

```
function T = pdtoep(n, m, w, theta)
%PDTOEP   Symmetric positive definite Toeplitz matrix.
%         PDTOEP(N, M, W, THETA) is an N-by-N symmetric positive (semi-)
%         definite (SPD) Toeplitz matrix, comprised of the sum of M rank 2
%         (or, for certain THETA, rank 1) SPD Toeplitz matrices.
%         Specifically,
%                 T = W(1)*T(THETA(1)) + ... + W(M)*T(THETA(M)),
%         where T(THETA(k)) has (i,j) element COS(2*PI*THETA(k)*(i-j)).
%         Defaults: M = N, W = RAND(M,1), THETA = RAND(M,1).

%         Reference:
%         G. Cybenko and C.F. Van Loan, Computing the minimum eigenvalue of
%         a symmetric positive definite Toeplitz matrix, SIAM J. Sci. Stat.
%         Comput., 7 (1986), pp. 123-131.
```

```
function P = pei(n, alpha)
%PEI    Pei matrix.
%       PEI(N, ALPHA), where ALPHA is a scalar, is the symmetric matrix
%       ALPHA*EYE(N) + ONES(N).
%       If ALPHA is omitted then ALPHA = 1 is used.
%       The matrix is singular for ALPHA = 0, -N.

%       Reference:
%       M.L. Pei, A test matrix for inversion procedures,
%       Comm. ACM, 5 (1962), p. 508.
```

```
function P = pentoep(n, a, b, c, d, e)
%PENTOEP   Pentadiagonal Toeplitz matrix (sparse).
%          P = PENTOEP(N, A, B, C, D, E) is the N-by-N pentadiagonal
%          Toeplitz matrix with diagonals composed of the numbers
%          A =: P(3,1), B =: P(2,1), C =: P(1,1), D =: P(1,2), E =: P(1,3).
%          Default: (A,B,C,D,E) = (1,-10,0,10,1) (a matrix of Rutishauser).
%                    This matrix has eigenvalues lying approximately on
%                    the line segment 2*cos(2*t) + 20*i*sin(t).
%
%          Interesting plots are
%          PS(FULL(PENTOEP(32,0,1,0,0,1/4)))  - 'triangle'
%          PS(FULL(PENTOEP(32,0,1/2,0,0,1)))  - 'propeller'
%          PS(FULL(PENTOEP(32,0,1/2,1,1,1)))  - 'fish'

%          References:
%          R.M. Beam and R.F. Warming, The asymptotic spectra of
%             banded Toeplitz and quasi-Toeplitz matrices, SIAM J. Sci.
%             Comput. 14 (4), 1993, pp. 971-1006.
```

```
%          H. Rutishauser, On test matrices, Programmation en Mathematiques
%             Numeriques, Editions Centre Nat. Recherche Sci., Paris, 165,
%             1966, pp. 349-365.
```

```
function [est, x, k] = pnorm(A, p, tol, noprint)
%PNORM   Estimate of matrix p-norm (1 <= p <= inf).
%        [EST, x, k] = PNORM(A, p, TOL) estimates the Holder p-norm of a
%        matrix A, using the p-norm power method with a specially
%        chosen starting vector.
%        TOL is a relative convergence tolerance (default 1E-4).
%        Returned are the norm estimate EST (which is a lower bound for the
%        exact p-norm), the corresponding approximate maximizing vector x,
%        and the number of power method iterations k.
%        A nonzero fourth argument causes trace output to the screen.
%        If A is a vector, this routine simply returns NORM(A, p).
%
%        See also NORM, NORMEST.

%        Note: The estimate is exact for p = 1, but is not always exact for
%        p = 2 or p = inf.  Code could be added to treat p = 2 and p = inf
%        separately.
%
%        Calls DUAL and SEQA.
%
%        Reference:
%        N.J. Higham, Estimating the matrix p-norm,
%        Numer. Math., 62 (1992), pp. 539-555.
```

```
function A = poisson(n)
%POISSON   Block tridiagonal matrix from Poisson's equation (sparse).
%          POISSON(N) is the block tridiagonal matrix of order N^2
%          resulting from discretizing Poisson's equation with the
%          5-point operator on an N-by-N mesh.

%          Reference:
%          G.H. Golub and C.F. Van Loan, Matrix Computations, second edition,
%          Johns Hopkins University Press, Baltimore, Maryland, 1989
%          (Section 4.5.4).
```

```
function [U, H] = poldec(A)
%POLDEC   Polar decomposition.
%         [U, H] = POLDEC(A) computes a matrix U of the same dimension
%         as A, and a Hermitian positive semi-definite matrix H,
```

```
%         such that A = U*H.
%         U has orthonormal columns if m>=n, and orthonormal rows if m<=n.
%         U and H are computed via an SVD of A.
%         U is a nearest unitary matrix to A in both the 2-norm and the
%         Frobenius norm.

%         Reference:
%         N.J. Higham, Computing the polar decomposition---with applications,
%         SIAM J. Sci. Stat. Comput., 7(4):1160--1174, 1986.
%
%         (The name `polar' is reserved for a graphics routine.)
```

```
function A = prolate(n, w)
%PROLATE   Prolate matrix - symmetric, ill-conditioned Toeplitz matrix.
%          A = PROLATE(N, W) is the N-by-N prolate matrix with parameter W.
%          It is a symmetric Toeplitz matrix.
%          If 0 < W < 0.5 then
%             - A is positive definite
%             - the eigenvalues of A are distinct, lie in (0, 1), and
%               tend to cluster around 0 and 1.
%          W defaults to 0.25.

%          Reference:
%          J.M. Varah. The Prolate matrix. Linear Algebra and Appl.,
%          187:269--278, 1993.
```

```
function y = ps(A, m, tol, rl, marksize)
%PS      Dot plot of a pseudospectrum.
%        PS(A, M, TOL, RL) plots an approximation to a pseudospectrum
%        of the square matrix A, using M random perturbations of size TOL.
%        M defaults to a SIZE(A)-dependent value and TOL to 1E-3.
%        RL defines the type of perturbation:
%          RL =  0 (default): absolute complex perturbations of 2-norm TOL.
%          RL =  1:           absolute real perturbations of 2-norm TOL.
%          RL = -1:           componentwise real perturbations of size TOL.
%        The eigenvalues of A are plotted as crosses `x'.
%        PS(A, M, TOL, RL, MARKSIZE) uses the specified marker size instead
%        of a size that depends on the figure size, the matrix order, and M.
%        If MARKSIZE < 0, the plot is suppressed and the plot data is returned
%        as an output argument.
%        PS(A, 0) plots just the eigenvalues of A.

%        For a given TOL, the pseudospectrum of A is the set of
%        pseudo-eigenvalues of A, that is, the set
```

```
%       { e : e is an eigenvalue of A+E, for some E with NORM(E) <= TOL }.
%
%       Reference:
%       L.N. Trefethen, Pseudospectra of matrices, in D.F. Griffiths and
%            G.A. Watson, eds, Numerical Analysis 1991, Proceedings of the 14th
%            Dundee Conference, vol. 260, Pitman Research Notes in Mathematics,
%            Longman Scientific and Technical, Essex, UK, 1992, pp. 234-266.
```

```
function [x, y, z, m] = pscont(A, k, npts, ax, levels)
%PSCONT   Contours and colour pictures of pseudospectra.
%         PSCONT(A, K, NPTS, AX, LEVELS) plots LOG10(1/NORM(R(z))),
%         where R(z) = INV(z*I-A) is the resolvent of the square matrix A,
%         over an NPTS-by-NPTS grid.
%         NPTS defaults to a SIZE(A)-dependent value.
%         The limits are AX(1) and AX(2) on the x-axis and
%                            AX(3) and AX(4) on the y-axis.
%         If AX is omitted, suitable limits are guessed based on the
%         eigenvalues of A.
%         The eigenvalues of A are plotted as crosses `x'.
%         K determines the type of plot:
%             K = 0 (default) PCOLOR and CONTOUR
%             K = 1           PCOLOR only
%             K = 2           SURFC (SURF and CONTOUR)
%             K = 3           SURF only
%             K = 4           CONTOUR only
%         The contours levels are specified by the vector LEVELS, which
%         defaults to -10:-1 (recall we are plotting log10 of the data).
%         Thus, by default, the contour lines trace out the boundaries of
%         the epsilon pseudospectra for epsilon = 1e-10, ..., 1e-1.
%         [X, Y, Z, NPTS] = PSCONT(A, ...) returns the plot data X, Y, Z
%         and the value of NPTS used.
%
%         After calling this function you may want to change the
%         color map (e.g., type COLORMAP HOT - see HELP COLOR) and the
%         shading (e.g., type SHADING INTERP - see HELP INTERP).
%         For an explanation of the term `pseudospectra' see PS.M.
%         When A is real and the grid is symmetric about the x-axis, this
%         routine exploits symmetry to halve the computational work.

%         Colour pseduospectral pictures of this type are referred to as
%         `spectral portraits' by Godunov, Kostin, and colleagues.
%         References:
%         V. I. Kostin, Linear algebra algorithms with guaranteed accuracy,
%            Technical Report TR/PA/93/05, CERFACS, Toulouse, France, 1993.
%         L.N. Trefethen, Pseudospectra of matrices, in D.F. Griffiths and
```

```
%           G.A. Watson, eds, Numerical Analysis 1991, Proceedings of the 14th
%           Dundee Conference, vol. 260, Pitman Research Notes in Mathematics,
%           Longman Scientific and Technical, Essex, UK, 1992, pp. 234-266.
```

```
function B = qmult(A)
%QMULT  Pre-multiply by random orthogonal matrix.
%       QMULT(A) is Q*A where Q is a random real orthogonal matrix from
%       the Haar distribution, of dimension the number of rows in A.
%       Special case: if A is a scalar then QMULT(A) is the same as
%                     QMULT(EYE(A)).

%       Called by RANDSVD.
%
%       Reference:
%       G.W. Stewart, The efficient generation of random
%       orthogonal matrices with an application to condition estimators,
%       SIAM J. Numer. Anal., 17 (1980), 403-409.
```

```
function A = rando(n, k)
%RANDO   Random matrix with elements -1, 0 or 1.
%        A = RANDO(N, K) is a random N-by-N matrix with elements from
%        one of the following discrete distributions (default K = 1):
%          K = 1:  A(i,j) =  0 or 1    with equal probability,
%          K = 2:  A(i,j) = -1 or 1    with equal probability,
%          K = 3:  A(i,j) = -1, 0 or 1 with equal probability.
%        N may be a 2-vector, in which case the matrix is N(1)-by-N(2).
```

```
function A = randsvd(n, kappa, mode, kl, ku)
%RANDSVD  Random matrix with pre-assigned singular values.
%      RANDSVD(N, KAPPA, MODE, KL, KU) is a (banded) random matrix of order N
%      with COND(A) = KAPPA and singular values from the distribution MODE.
%      N may be a 2-vector, in which case the matrix is N(1)-by-N(2).
%      Available types:
%             MODE = 1:   one large singular value,
%             MODE = 2:   one small singular value,
%             MODE = 3:   geometrically distributed singular values,
%             MODE = 4:   arithmetically distributed singular values,
%             MODE = 5:   random singular values with unif. dist. logarithm.
%      If omitted, MODE defaults to 3, and KAPPA defaults to SQRT(1/EPS).
%      If MODE < 0 then the effect is as for ABS(MODE) except that in the
%      original matrix of singular values the order of the diagonal entries
%      is reversed: small to large instead of large to small.
%      KL and KU are the lower and upper bandwidths respectively; if they
```

```
%      are omitted a full matrix is produced.
%      If only KL is present, KU defaults to KL.
%      Special case: if KAPPA < 0 then a random full symmetric positive
%                    definite matrix is produced with COND(A) = -KAPPA and
%                    eigenvalues distributed according to MODE.
%                    KL and KU, if present, are ignored.
```

```
function A = redheff(n)
%REDHEFF    A matrix of 0s and 1s of Redheffer.
%           A = REDHEFF(N) is an N-by-N matrix of 0s and 1s defined by
%               A(i,j) = 1 if j = 1 or if i divides j,
%               A(i,j) = 0 otherwise.
%           It has N - FLOOR(LOG2(N)) - 1 eigenvalues equal to 1,
%           a real eigenvalue (the spectral radius) approximately SQRT(N),
%           a negative eigenvalue approximately -SQRT(N),
%           and the remaining eigenvalues are provably ``small''.
%           Barrett and Jarvis (1992) conjecture that
%             ``the small eigenvalues all lie inside the unit circle
%               ABS(Z) = 1'',
%           and a proof of this conjecture, together with a proof that some
%           eigenvalue tends to zero as N tends to infinity, would yield
%           a new proof of the prime number theorem.
%           The Riemann hypothesis is true if and only if
%           DET(A) = O( N^(1/2+epsilon) ) for every epsilon > 0
%                                             (`!' denotes factorial).
%           See also RIEMANN.

%           Reference:
%           W.W. Barrett and T.J. Jarvis,
%           Spectral Properties of a Matrix of Redheffer,
%           Linear Algebra and Appl., 162 (1992), pp. 673-683.
```

```
function A = riemann(n)
%RIEMANN    A matrix associated with the Riemann hypothesis.
%           A = RIEMANN(N) is an N-by-N matrix for which the
%           Riemann hypothesis is true if and only if
%           DET(A) = O( N! N^(-1/2+epsilon) ) for every epsilon > 0
%                                             (`!' denotes factorial).
%           A = B(2:N+1, 2:N+1), where
%           B(i,j) = i-1 if i divides j and -1 otherwise.
%           Properties include, with M = N+1:
%              Each eigenvalue E(i) satisfies ABS(E(i)) <= M - 1/M.
%              i <= E(i) <= i+1 with at most M-SQRT(M) exceptions.
%              All integers in the interval (M/3, M/2] are eigenvalues.
```

```
%
%           See also REDHEFF.

%           Reference:
%           F. Roesler, Riemann's hypothesis as an eigenvalue problem,
%           Linear Algebra and Appl., 81 (1986), pp. 153-198.
```

```
function z = rq(A,x)
%RQ      Rayleigh quotient.
%        RQ(A,x) is the Rayleigh quotient of A and x, x'*A*x/(x'*x).

%        Called by FV.
```

```
function A = rschur(n, mu, x, y)
%RSCHUR   An upper quasi-triangular matrix.
%         A = RSCHUR(N, MU, X, Y) is an N-by-N matrix in real Schur form.
%         All the diagonal blocks are 2-by-2 (except for the last one, if N
%         is odd) and the k'th has the form [x(k) y(k); -y(k) x(k)].
%         Thus the eigenvalues of A are x(k) +/- i*y(k).
%         MU (default 1) controls the departure from normality.
%         Defaults: X(k) = -k^2/10, Y(k) = -k, i.e., the eigenvalues
%                   lie on the parabola x = -y^2/10.

%         References:
%         F. Chatelin, Eigenvalues of Matrices, John Wiley, Chichester, 1993;
%            Section 4.2.7.
%         F. Chatelin and V. Fraysse, Qualitative computing: Elements
%            of a theory for finite precision computation, Lecture notes,
%            CERFACS, Toulouse, France and THOMSON-CSF, Orsay, France,
%            June 1993.
```

```
function see(A, k)
%SEE    Pictures of a matrix and its (pseudo-) inverse.
%       SEE(A) displays MESH(A), MESH(PINV(A)), SEMILOGY(SVD(A),'o'),
%       and (if A is square) FV(A) in four subplot windows.
%       SEE(A, 1) plots an approximation to the pseudospectrum in the
%       third window instead of the singular values.
%       SEE(A, -1) plots only the eigenvalues in the fourth window,
%       which is much quicker than plotting the field of values.
%       If A is complex, only real parts are used for the mesh plots.
%       If A is sparse, just SPY(A) is shown.
```

```
function y = seqa(a, b, n)
%SEQA   Additive sequence.
%       Y = SEQA(A, B, N) produces a row vector comprising N equally
%       spaced numbers starting at A and finishing at B.
%       If N is omitted then 10 points are generated.
```

```
function x = seqcheb(n, k)
%SEQCHEB   Sequence of points related to Chebyshev polynomials.
%          X = SEQCHEB(N, K) produces a row vector of length N.
%          There are two choices:
%              K = 1:  zeros of T_N,         (the default)
%              K = 2:  extrema of T_{N-1},
%          where T_k is the Chebsyhev polynomial of degree k.
```

```
function y = seqm(a, b, n)
%SEQM   Multiplicative sequence.
%       Y = SEQM(A, B, N) produces a row vector comprising N
%       logarithmically equally spaced numbers, starting at A ~= 0
%       and finishing at B ~= 0.
%       If A*B < 0 and N > 2 then complex results are produced.
%       If N is omitted then 10 points are generated.
```

```
function show(x)
%SHOW   Display signs of matrix elements.
%       SHOW(X) displays X in `FORMAT +' form, that is,
%       with `+', `-' and  blank representing positive, negative
%       and zero elements respectively.
```

```
function [S, N] = signm(A)
%SIGNM   Matrix sign decomposition.
%        [S, N] = SIGNM(A) is the matrix sign decomposition A = S*N,
%        computed via the Schur decomposition.
%        S is the matrix sign function, sign(A).

%        Reference:
%        N.J. Higham, The matrix sign decomposition and its relation to
%        the polar decomposition, Numerical Analysis Report No. 225,
%        University of Manchester, England, April 1993;
%        to appear in Linear Algebra and Appl.
```

```
function S = skewpart(A)
%SKEWPART  Skew-symmetric (skew-Hermitian) part.
%          SKEWPART(A) is the skew-symmetric (skew-Hermitian) part of A,
%          (A - A')/2.
%          It is the nearest skew-symmetric (skew-Hermitian) matrix to A in
%          both the 2- and the Frobenius norms.
```

```
function A = smoke(n, k)
%SMOKE     Smoke matrix - complex, with a `smoke ring' pseudospectrum.
%          SMOKE(N) is an N-by-N matrix with 1s on the
%          superdiagonal, 1 in the (N,1) position, and powers of
%          roots of unity along the diagonal.
%          SMOKE(N, 1) is the same except for a zero (N,1) element.
%          The eigenvalues of SMOKE(N, 1) are the N'th roots of unity;
%          those of SMOKE(N) are the N'th roots of unity times 2^(1/N).
%
%          Try PS(SMOKE(32)).  For SMOKE(N, 1) the pseudospectrum looks
%          like a sausage folded back on itself.
%          GERSH(SMOKE(N, 1)) is interesting.

%          Reference:
%          L. Reichel and L.N. Trefethen, Eigenvalues and pseudo-eigenvalues of
%          Toeplitz matrices, Linear Algebra and Appl., 162-164:153-185, 1992.
```

```
function A = sparsify(A, p)
%SPARSIFY   Randomly sets matrix elements to zero.
%           S = SPARSIFY(A, P) is A with elements randomly set to zero
%           (S = S' if A is square and A = A', i.e. symmetry is preserved).
%           Each element has probability P of being zeroed.
%           Thus on average 100*P percent of the elements of A will be zeroed.
%           Default: P = 0.25.
```

```
function S = sub(A, i, j)
%SUB     Principal submatrix.
%        SUB(A,i,j) is A(i:j,i:j).
%        SUB(A,i)  is the leading principal submatrix of order i,
%        A(1:i,1:i), if i>0, and the trailing principal submatrix
%        of order ABS(i) if i<0.
```

```
function S = symmpart(A)
%SYMMPART  Symmetric (Hermitian) part.
%          SYMMPART(A) is the symmetric (Hermitian) part of A, (A + A')/2.
%          It is the nearest symmetric (Hermitian) matrix to A in both the
%          2- and the Frobenius norms.
```

```
function [Q, T] = trap2tri(L)
%TRAP2TRI  Unitary reduction of trapezoidal matrix to triangular form.
%          [Q, T] = TRAP2TRI(L), where L is an m-by-n lower trapezoidal
%          matrix with m >= n, produces a unitary Q such that QL = [T; 0],
%          where T is n-by-n and lower triangular.
%          Q is a product of Householder transformations.

%          Called by RANDSVD.
%
%          Reference:
%          G.H. Golub and C.F. Van Loan, Matrix Computations, second edition,
%          Johns Hopkins University Press, Baltimore, Maryland, 1989.
%          P5.2.5, p. 220.
```

```
function T = tridiag(n, x, y, z)
%TRIDIAG  Tridiagonal matrix (sparse).
%         TRIDIAG(X, Y, Z) is the tridiagonal matrix with subdiagonal X,
%         diagonal Y, and superdiagonal Z.
%         X and Z must be vectors of dimension one less than Y.
%         Alternatively TRIDIAG(N, C, D, E), where C, D, and E are all
%         scalars, yields the Toeplitz tridiagonal matrix of order N
%         with subdiagonal elements C, diagonal elements D, and superdiagonal
%         elements E.   This matrix has eigenvalues (Todd 1977)
%                  D + 2*SQRT(C*E)*COS(k*PI/(N+1)), k=1:N.
%         TRIDIAG(N) is the same as TRIDIAG(N,-1,2,-1), which is
%         a symmetric positive definite M-matrix (the negative of the
%         second difference matrix).

%         References:
%         J. Todd, Basic Numerical Mathematics, Vol. 2: Numerical Algebra,
%           Birkhauser, Basel, and Academic Press, New York, 1977, p. 155.
%         D.E. Rutherford, Some continuant determinants arising in physics and
%           chemistry---II, Proc. Royal Soc. Edin., 63, A (1952), pp. 232-241.
```

```
function t = triw(n, alpha, k)
%TRIW   Upper triangular matrix discussed by Wilkinson and others.
%       TRIW(N, ALPHA, K) is the upper triangular matrix with ones on
%       the diagonal and ALPHAs on the first K >= 0 superdiagonals.
%       N may be a 2-vector, in which case the matrix is N(1)-by-N(2) and
%       upper trapezoidal.
%       Defaults: ALPHA = -1,
%                 K = N - 1     (full upper triangle).
%       TRIW(N) is a matrix discussed by Kahan, Golub and Wilkinson.
%
```

```
%       Ostrowski (1954) shows that
%         COND(TRIW(N,2)) = COT(PI/(4*N))^2,
%       and for large ABS(ALPHA),
%         COND(TRIW(N,ALPHA)) is approximately ABS(ALPHA)^N*SIN(PI/(4*N-2)).
%
%       Adding -2^(2-N) to the (N,1) element makes TRIW(N) singular,
%       as does adding -2^(1-N) to all elements in the first column.

%       References:
%       G.H. Golub and J.H. Wilkinson, Ill-conditioned eigensystems and the
%          computation of the Jordan canonical form, SIAM Review,
%          18(4), 1976, pp. 578-619.
%       W. Kahan, Numerical linear algebra, Canadian Math. Bulletin,
%          9 (1966), pp. 757-801.
%       A.M. Ostrowski, On the spectrum of a one-parametric family of
%          matrices, J. Reine Angew. Math., 193 (3/4), 1954, pp. 143-160.
%       J.H. Wilkinson, Singular-value decomposition---basic aspects,
%          in D.A.H. Jacobs, ed., Numerical Software---Needs and Availability,
%          Academic Press, London, 1978, pp. 109-135.
```

```
function V = vand(m, 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).
%       VAND(M,P) is a rectangular version of VAND(P) with M rows.
%       Special case: If P is a scalar then P equally spaced points on [0,1]
%                     are used.

%       Reference:
%       N.J. Higham, Stability analysis of algorithms for solving
%       confluent Vandermonde-like systems, SIAM J. Matrix Anal. Appl.,
%       11 (1990), pp. 23-41.
```

```
function A = wathen(nx, ny, k)
%WATHEN  Wathen matrix - a finite element matrix (sparse, random entries).
%        A = WATHEN(NX,NY) is a sparse random N-by-N finite element matrix
%        where N = 3*NX*NY + 2*NX + 2*NY + 1.
%        A is precisely the `consistent mass matrix' for a regular NX-by-NY
%        grid of 8-node (serendipity) elements in 2 space dimensions.
%        A is symmetric positive definite for any (positive) values of
%        the `density', RHO(NX,NY), which is chosen randomly in this routine.
%        In particular, if D = DIAG(DIAG(A)), then
%              0.25 <= EIG(INV(D)*A) <= 4.5
%        for any positive integers NX and NY and any densities RHO(NX,NY).
```

```
%        This diagonally scaled matrix is returned by WATHEN(NX,NY,1).

%        Reference:
%        A.J. Wathen, Realistic eigenvalue bounds for the Galerkin
%        mass matrix, IMA J. Numer. Anal., 7 (1987), pp. 449-457.
```

```
function [A, b] = wilk(n)
%WILK   Various specific matrices devised/discussed by Wilkinson.
%       [A, b] = WILK(N) is the matrix or system of order N.
%       N = 3: upper triangular system Ux=b illustrating inaccurate solution.
%       N = 4: lower triangular system Lx=b, ill-conditioned.
%       N = 5: HILB(6)(1:5,2:6)*1.8144.  Symmetric positive definite.
%       N = 21: W21+, tridiagonal.   Eigenvalue problem.

%       References:
%       J.H. Wilkinson, Error analysis of direct methods of matrix inversion,
%          J. Assoc. Comput. Mach., 8 (1961),  pp. 281-330.
%       J.H. Wilkinson, Rounding Errors in Algebraic Processes, Notes on Applied
%          Science No. 32, Her Majesty's Stationery Office, London, 1963.
%       J.H. Wilkinson, The Algebraic Eigenvalue Problem, Oxford University
%          Press, 1965.
```

Acknowledgements

In preparing the earlier test matrix collections I benefited from the helpful suggestions of people too numerous to mention. While working on the new toolbox I received valuable advice from Cleve Moler and Nick Trefethen. Per Christian Hansen offered helpful comments on a draft version of this document. Alan Edelman helped me with the section on random matrices.

References

[1] E. Anderson, Z. Bai, C. H. Bischof, J. W. Demmel, J. J. Dongarra, J. J. Du Croz, A. Greenbaum, S. J. Hammarling, A. McKenney, S. Ostrouchov, and D. C. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[2] Zhaojun Bai. A collection of test matrices for the large scale nonsymmetric eigenvalue problem (version 0.0). Manuscript, April 1993.

[3] Richard Bartels and Barry Joe. On generating discrete linear l_∞ test problems. *SIAM J. Sci. Stat. Comput.*, 10(3):550–561, 1989.

[4] John Caffney. Another test matrix for determinants and inverses. *Comm. ACM*, 6(6):310, 1963.

[5] A. K. Cline and R. K. Rew. A set of counter-examples to three condition number estimators. *SIAM J. Sci. Stat. Comput.*, 4(4):602–611, 1983.

[6] J. W. Demmel and A. McKenney. A test matrix generation suite. LAPACK Working Note #9, Courant Institute of Mathematical Sciences, New York, 1989.

[7] Jack J. Dongarra and Eric Grosse. Distribution of mathematical software via electronic mail. *Comm. ACM*, 30(5):403–407, 1987.

[8] Iain S. Duff, Roger G. Grimes, and John G. Lewis. Sparse matrix test problems. *ACM Trans. Math. Soft.*, 15(1):1–14, 1989.

[9] Iain S. Duff, Roger G. Grimes, and John G. Lewis. Users' guide for the Harwell–Boeing sparse matrix collection (release 1). Report RAL-92-086, Atlas Centre, Rutherford Appleton Laboratory, Didcot, Oxon, December 1992.

[10] Alan Edelman. Eigenvalues and condition numbers of random matrices. *SIAM J. Matrix Anal. Appl.*, 9(4):543–560, 1988.

[11] Alan Edelman. Eigenvalue roulette and random test matrices. In Marc S. Moonen, Gene H. Golub, and Bart L. De Moor, editors, *Linear Algebra for Large Scale and Real-Time Applications*, volume 232 of *NATO ASI Series E*, pages 365–368. Kluwer Academic Publishers, Dordrecht, 1993.

[12] Alan Edelman, Eric Kostlan, and Michael Shub. How many eigenvalues of a random matrix are real? To appear in J. Amer. Math. Soc., 1994.

[13] W. H. Enright and J. D. Pryce. Two FORTRAN packages for assessing initial value methods. *ACM Trans. Math. Soft.*, 13(1):1–27, 1987.

[14] W. L. Frank. Computing eigenvalues of complex matrices by determinant evaluation and by methods of Danilewski and Wielandt. *J. Soc. Indust. Appl. Math.*, 6:378–392, 1958.

[15] F. R. Gantmacher. *The Theory of Matrices*, volume two. Chelsea, New York, 1959.

[16] David M. Gay. Electronic mail distribution of linear programming test problems. *Mathematical Programming Society COAL Newsletter*, December:10–12, 1985.

[17] Stuart Geman. The spectral radius of large random matrices. *Ann. Probab.*, 14(4):1318–1328, 1986.

[18] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland, second edition, 1989.

[19] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley, Reading, Massachusetts, 1989.

[20] Robert T. Gregory and David L. Karney. *A Collection of Matrices for Testing Computational Algorithms*. John Wiley, New York, 1969. Reprinted with corrections by Robert E. Krieger, Huntington, New York, 1978.

[21] Per Christian Hansen. Regularization tools. A Matlab package for analysis and solution of discrete ill-posed problems. Report UNIC-92-03, UNI·C, Technical University of Denmark, DK-2800 Lyngby, Denmark, June 1992.

[22] Per Christian Hansen. Test matrices for regularization methods. Report UNIC-93-09, UNI·C, Technical University of Denmark, DK-2800 Lyngby, Denmark, August 1993.

[23] Nicholas J. Higham. Computing the polar decomposition—with applications. *SIAM J. Sci. Stat. Comput.*, 7(4):1160–1174, October 1986.

[24] Nicholas J. Higham. A survey of condition number estimation for triangular matrices. *SIAM Review*, 29(4):575–596, 1987.

[25] Nicholas J. Higham. A collection of test matrices in MATLAB. Numerical Analysis Report No. 172, University of Manchester, England, July 1989.

[26] Nicholas J. Higham. How accurate is Gaussian elimination? In D. F. Griffiths and G. A. Watson, editors, *Numerical Analysis 1989, Proceedings of the 13th Dundee Conference*, volume 228 of *Pitman Research Notes in Mathematics*, pages 137–154. Longman Scientific and Technical, Essex, UK, 1990.

[27] Nicholas J. Higham. Algorithm 694: A collection of test matrices in MATLAB. *ACM Trans. Math. Soft.*, 17(3):289–305, September 1991.

[28] Nicholas J. Higham. Estimating the matrix p-norm. *Numer. Math.*, 62:539–555, 1992.

[29] Nicholas J. Higham. The matrix sign decomposition and its relation to the polar decomposition. Numerical Analysis Report No. 225, University of Manchester, England, April 1993. To appear in Linear Algebra and Appl.

[30] Nicholas J. Higham and Desmond J. Higham. Large growth factors in Gaussian elimination with pivoting. *SIAM J. Matrix Anal. Appl.*, 10(2):155–164, April 1989.

[31] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[32] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, 1991.

[33] W. Kahan. Numerical linear algebra. *Canadian Math. Bulletin*, 9:757–801, 1966.

[34] Samuel Karlin. *Total Positivity*, volume 1. Stanford University Press, 1968.

[35] Charles Kenney and Alan J. Laub. Controllability and stability radii for companion form systems. *Math. Control Signals Systems*, 1:239–256, 1988.

[36] I. J. Lustig. An analysis of an available set of linear programming test problems. *Computers and Operations Research*, 16:173–184, 1989.

[37] Webb Miller. *The Engineering of Numerical Software*. Prentice-Hall, Englewood Cliffs, New Jersey, 1984.

[38] Cleve B. Moler. MATLAB's magical mystery tour. *The MathWorks Newsletter*, 7(1):8–9, 1993.

[39] J. J. Moré, B. S. Garbow, and K. E. Hillstrom. Testing unconstrained optimization software. *ACM Trans. Math. Soft.*, 7:17–41, 1981.

[40] Morris Newman. Matrix computations. In John Todd, editor, *Survey of Numerical Analysis*, pages 222–254. McGraw-Hill, New York, 1962.

[41] Morris Newman and John Todd. The evaluation of matrix inversion programs. *J. Soc. Indust. Appl. Math.*, 6(4):466–476, 1958.

[42] Heinz-Otto Peitgen, Hartmut Jürgens, and Dietmar Saupe. *Fractals for the Classroom. Part One: Introduction to Fractals and Chaos*. Springer-Verlag, New York, 1992.

[43] J. R. Rice and R. E. Boisvert. *Solving Elliptic Problems using ELLPACK*. Springer-Verlag, New York, 1985.

[44] A. Ruhe. Closest normal matrix finally found! *BIT*, 27:585–598, 1987.

[45] H. Rutishauser. On test matrices. *Programmation en Mathématiques Numériques, Éditions Centre Nat. Recherche Sci., Paris*, 165:349–365, 1966.

[46] G. W. Stewart. Updating a rank-revealing ULV decomposition. *SIAM J. Matrix Anal. Appl.*, 14(2):494–499, 1993.

[47] J. Stoer and C. Witzgall. Transformations by diagonal matrices in a normed space. *Numer. Math.*, 4:158–171, 1962.

[48] Olga Taussky and Marvin Marcus. Eigenvalues of finite matrices. In John Todd, editor, *Survey of Numerical Analysis*, pages 279–313. McGraw-Hill, New York, 1962.

[49] The MathWorks, Inc. *Maple Symbolic Toolbox.* Natick, Massachusetts, May 1993. Beta test edition.

[50] Kim-Chuan Toh and Lloyd N. Trefethen. Pseudozeros of polynomials and pseudospectra of companion matrices. Technical Report TR 93-1360, Department of Computer Science, Cornell University, Ithaca, New York, June 1993.

[51] Lloyd N. Trefethen. Pseudospectra of matrices. In D. F. Griffiths and G. A. Watson, editors, *Numerical Analysis 1991, Proceedings of the 14th Dundee Conference*, volume 260 of *Pitman Research Notes in Mathematics*, pages 234–266. Longman Scientific and Technical, Essex, UK, 1992.

[52] Lloyd N. Trefethen. *Spectra and Pseudospectra: The Behavior of Non-Normal Matrices and Operators.* 1994. Book in preparation.

[53] H. W. Turnbull. *The Theory of Determinants, Matrices, and Invariants.* Blackie, London and Glasgow, 1929.

[54] J. M. Varah. A generalization of the Frank matrix. *SIAM J. Sci. Stat. Comput.*, 7(3):835–839, 1986.

[55] Joan R. Westlake. *A Handbook of Numerical Matrix Inversion and Solution of Linear Equations.* John Wiley, New York, 1968.

[56] J. H. Wilkinson. Error analysis of floating-point computation. *Numer. Math.*, 2:319–340, 1960.

[57] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Oxford University Press, 1965.

[58] G. Zielke. Report on test matrices for generalized inverses. *Computing*, 36:105–162, 1986.