Matrix Depot Documentation

Release 0.1.0

Weijian Zhang

Nov 30, 2018

Contents

Matrix Depot is an extensible test matrix collection for Julia. It provides a diverse collection of test matrices, including parametrized matrices and real-life matrices.

- Source at Github
- Release Notes

CHAPTER 1

Installation

To install the release version, type:

```
julia> ]
pkg> add MatrixDepot
```

1.1 Usage

Every matrix in the collection is represented by a string "matrix_name", for example, the Cauchy matrix is represented by "cauchy" and the Hilbert matrix is represented by "hilb".

The matrix groups are noted as symbols. For example, the class of the symmetric matrices is symbolized by :symmetric.

mdinfo()

Return a list of all the matrices in the collection:

```
julia> matrixdepot()

Matrices:
 1) baart            2) binomial         3) blur             4) cauchy
 5) chebspec         6) chow             7) circul           8) clement
 9) companion       10) deriv2          11) dingdong        12) fiedler
 13) forsythe        14) foxgood         15) frank           16) golub
 17) gravity         18) grcar           19) hadamard        20) hankel
 21) heat            22) hilb            23) invhilb         24) invol
 25) kahan           26) kms             27) lehmer          28) lotkin
 29) magic           30) minij           31) moler           32) neumann
 33) oscillate       34) parter          35) pascal          36) pei
 37) phillips        38) poisson         39) prolate         40) randcorr
 41) rando           42) randsvd         43) rohess          44) rosser
 45) sampling        46) shaw            47) spikes          48) toeplitz
 49) tridiag         50) triw            51) ursell          52) vand
```

(continues on next page)

(continued from previous page)

```
 53) wathen           54) wilkinson       55) wing
Groups:
 all           data          eigen         illcond
 inverse       posdef       random         regprob
 sparse        symmetric
```

matrixdepot (*matrix_name*, *p1*, *p2*, *...*)

Return a matrix specified by the query string `matrix_name`. `p1, p2, ...` are input parameters depending on `matrix_name`. For example:

```
julia> matrixdepot("hilb", 5, 4)
5x4 Array{Float64,2}:
1.0       0.5       0.333333  0.25
0.5       0.333333  0.25      0.2
0.333333  0.25      0.2       0.166667
0.25      0.2       0.166667  0.142857
0.2       0.166667  0.142857  0.125
```

mdinfo (*matrix_name*)

Return the documentation of `matrix_name`, including input options, groups and reference. For example:

```
julia> mdinfo("moler")
   Moler Matrix


The Moler matrix is a symmetric positive definite matrix. It has one small
eigenvalue.

Input options:

  •  [type,] dim, alpha: dim is the dimension of the matrix, alpha is a
    scalar;

  •  [type,] dim: alpha = -1.

Groups: ["inverse", "ill-cond", "symmetric", "pos-def"]

References:

J.C. Nash, Compact Numerical Methods for Computers: Linear Algebra and
Function Minimisation, second edition, Adam Hilger, Bristol, 1990
(Appendix 1).
```

listnames (*group_name*)

Return a list of matrices which belong to group `group_name`. For example:

```
julia> matrixdepot(:posdef)
11-element Array{ASCIIString,1}:
"hilb"
"cauchy"
"circul"
"invhilb"
"moler"
"pascal"
"pei"
"minij"
```

(continues on next page)

(continued from previous page)

```
"tridiag"
"lehmer"
"poisson"
```

listnames (*group1 & group2 & ...*)
Return a list of matrices which belong to `group1` and `group2`, etc. For example:

```
julia> mdlist(:symmetric & :inverse, :illcond & :posdef)
7-element Array{ASCIIString,1}:
"hilb"
"cauchy"
"invhilb"
"moler"
"pascal"
"pei"
"tridiag"
```

mdlist (*{builtinuserspmm}(num)*)
Access matrix by number. For example:

```
julia> mdlist(builtin(3))
"chebspec"
```

mdlist (*builtin(num1:num2, ...)*)

Access matrix by range and combinations. For example:: julia> mdlist(builtin(1:4, 6, 10:15)) 11-element Array{String,1}:

"baart" "binomial" "blur" "cauchy" "chow" "deriv2" "dingdong" "erdrey" "fiedler" "forsythe" "foxgood"

mdinfo (*name*)
Output matrix information, where `name` is a matrix data name or pattern.

matrixdepot (*name, arg...*)
Generate the matrix data given by `name`.

We can define our own groups using the macro `@addgroup` and remove a defined group using `@rmgroup`.

@addgroup group_name = ["matrix1", "matrix2", "matrix3"]
Create a new group `"group_name"` such that it has members `"matrix1"`, `"matrix2"` and `"matrix3"`.

@rmgroup group_name
Delete a created group `group_name`.

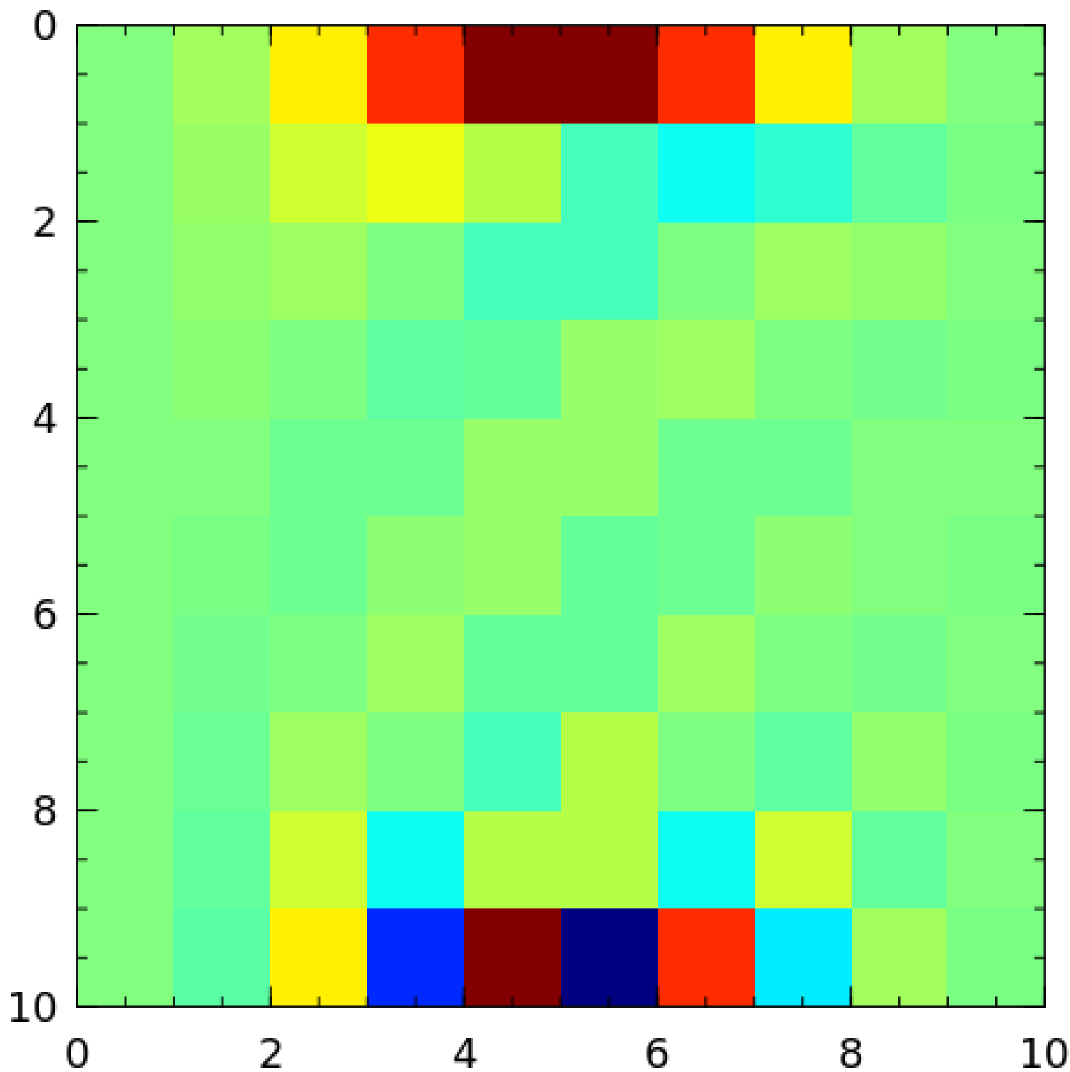
1.2 Matrices

- *binomial*
- *cauchy*
- *chebspec*
- *chow*
- *circul*
- *clement*

- *companion*
- *dingdong*
- *fiedler*
- *forsythe*
- *frank*
- *golub*
- *grcar*
- *hadamard*
- *hankel*
- *hilb*
- *invhilb*
- *invol*
- *kahan*
- *kms*
- *lehmer*
- *lotkin*
- *magic*
- *minij*
- *moler*
- *neumann*
- *oscillate*
- *parter*
- *pascal*
- *pei*
- *poisson*
- *prolate*
- *randcorr*
- *rando*
- *randsvd*
- *rohess*
- *rosser*
- *sampling*
- *toeplitz*
- *tridiag*
- *triw*
- *vand*

- *wathen*
- *wilkinson*

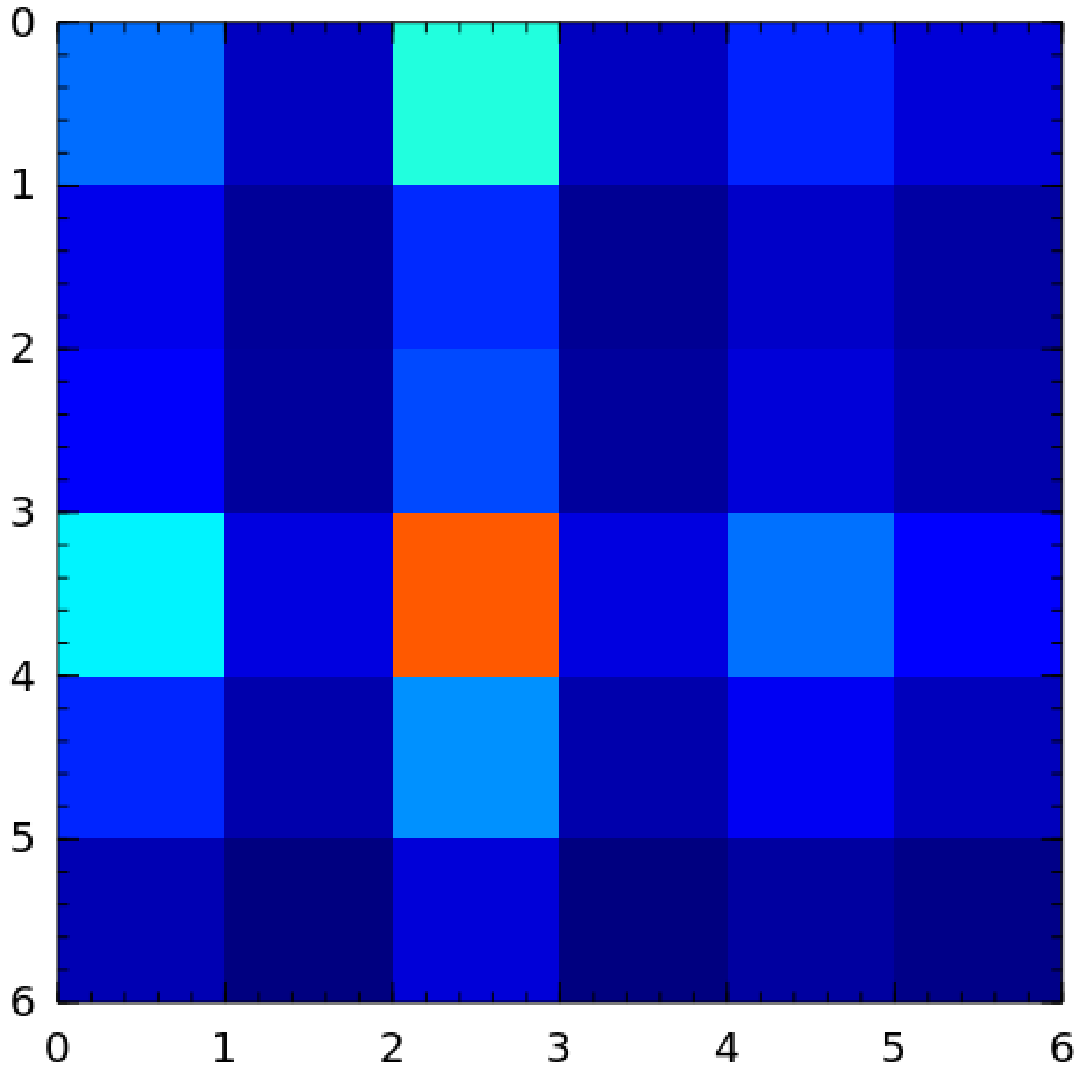
binomial A binomial matrix that arose from the example in *[bmsz01]*. The matrix is a multiple of involutory matrix.



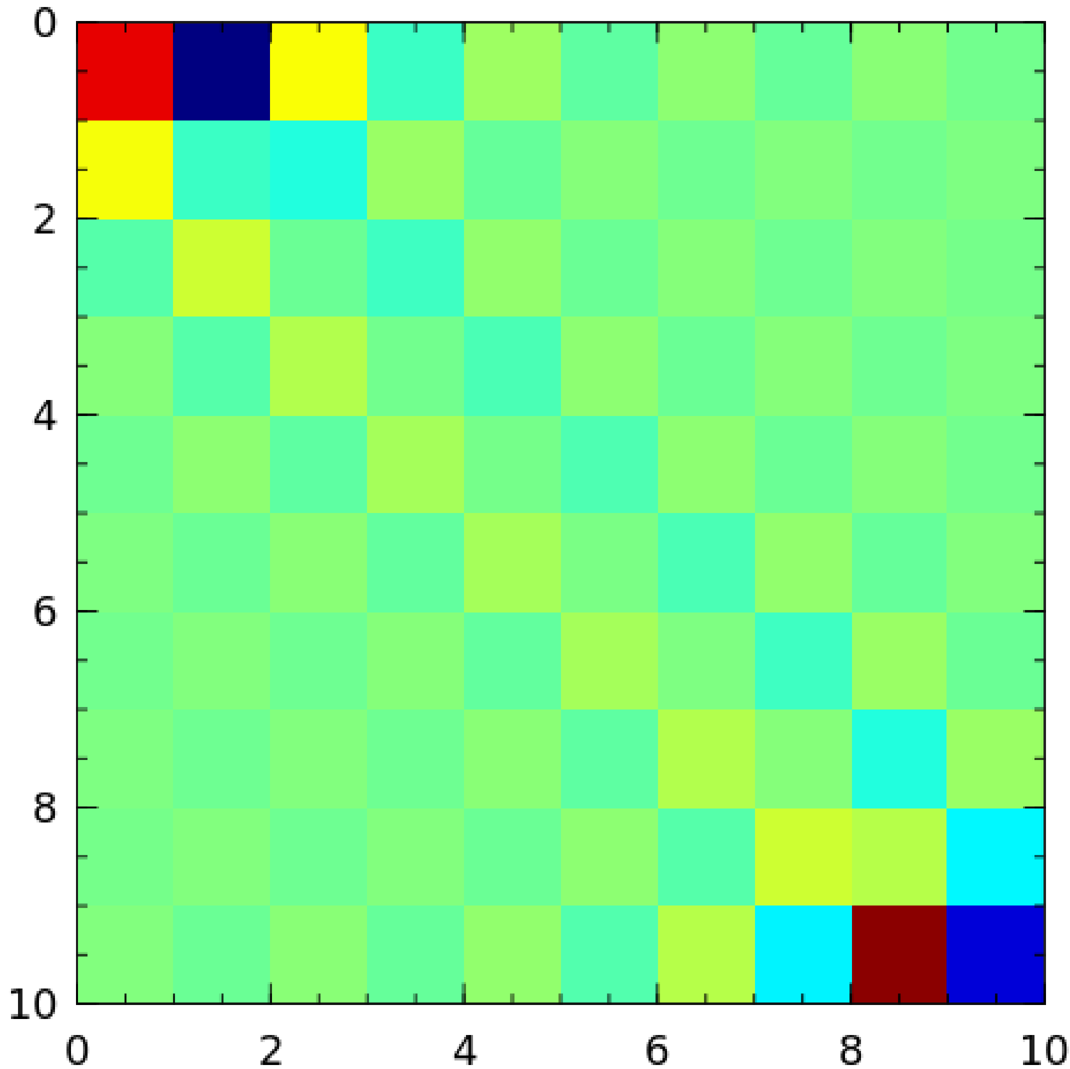
cauchy The Cauchy matrix is an m-by-n matrix with (i, j) element

$$\frac{1}{x_i - y_i}, \quad x_i - y_i \neq 0,$$

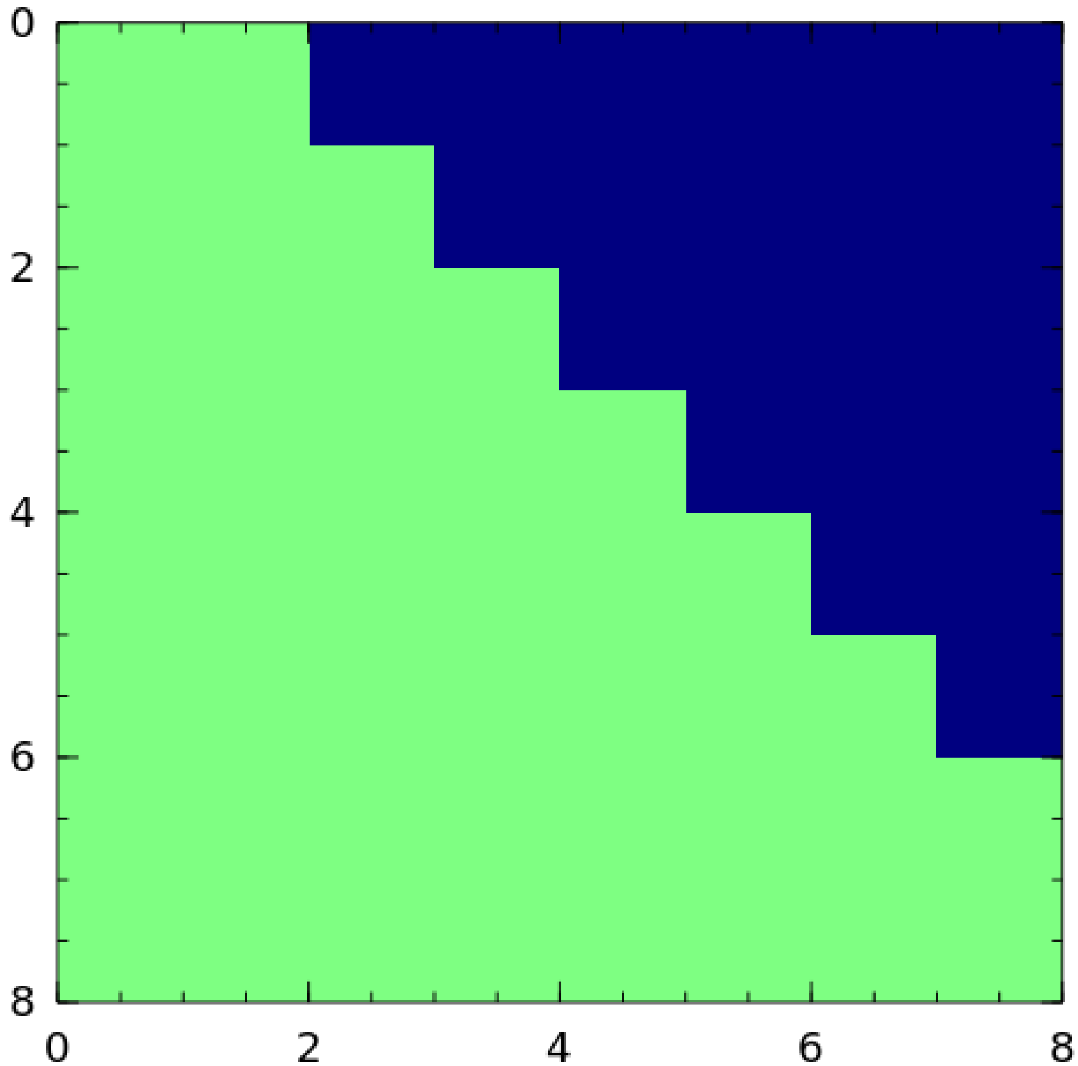
where x_i and y_i are elements of vectors x and y.



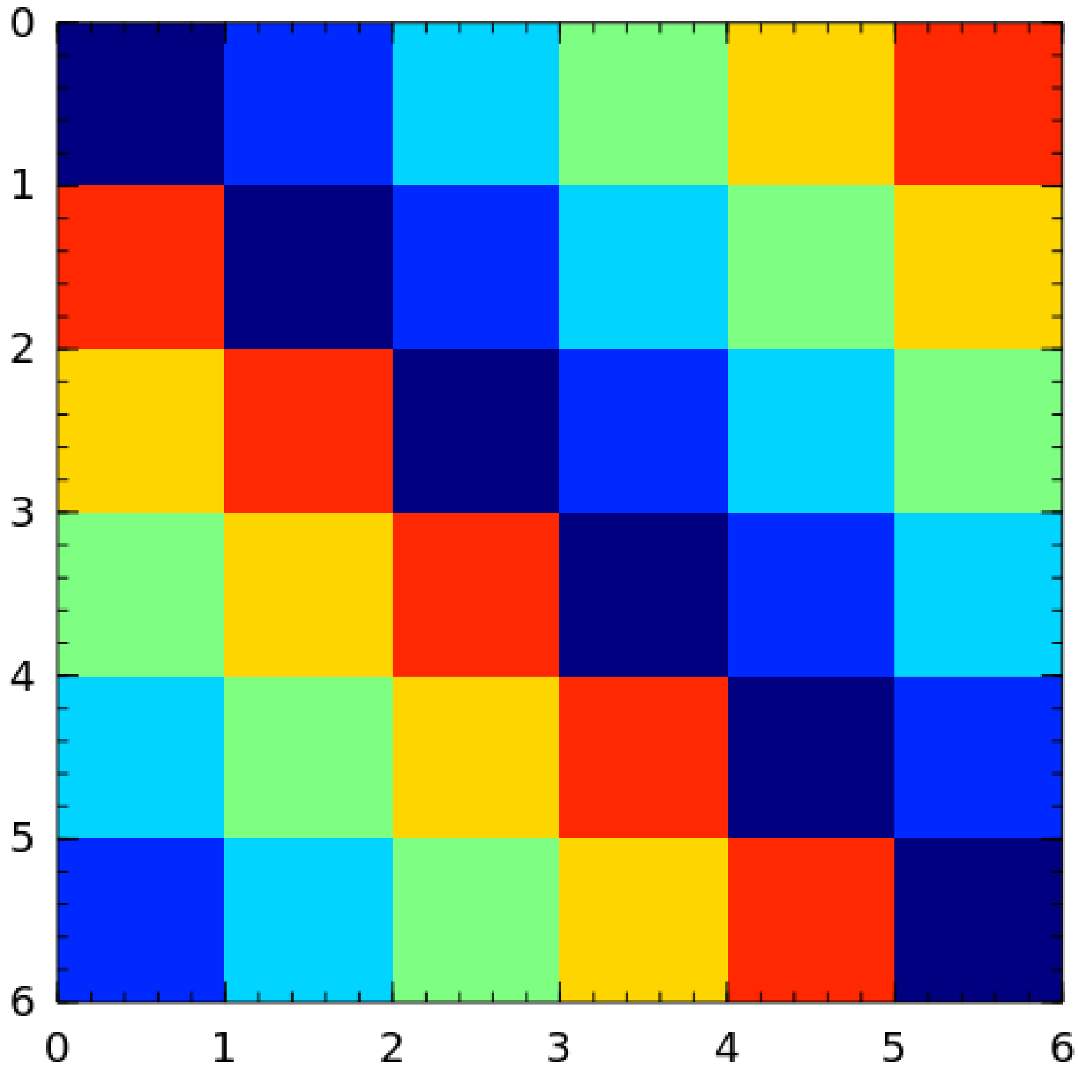
chebspec Chebyshev spectral differentiation matrix. If `k = 0`,the generated matrix is nilpotent and a vector with all one entries is a null vector. If `k = 1`, the generated matrix is nonsingular and well-conditioned. Its eigenvalues have negative real parts.



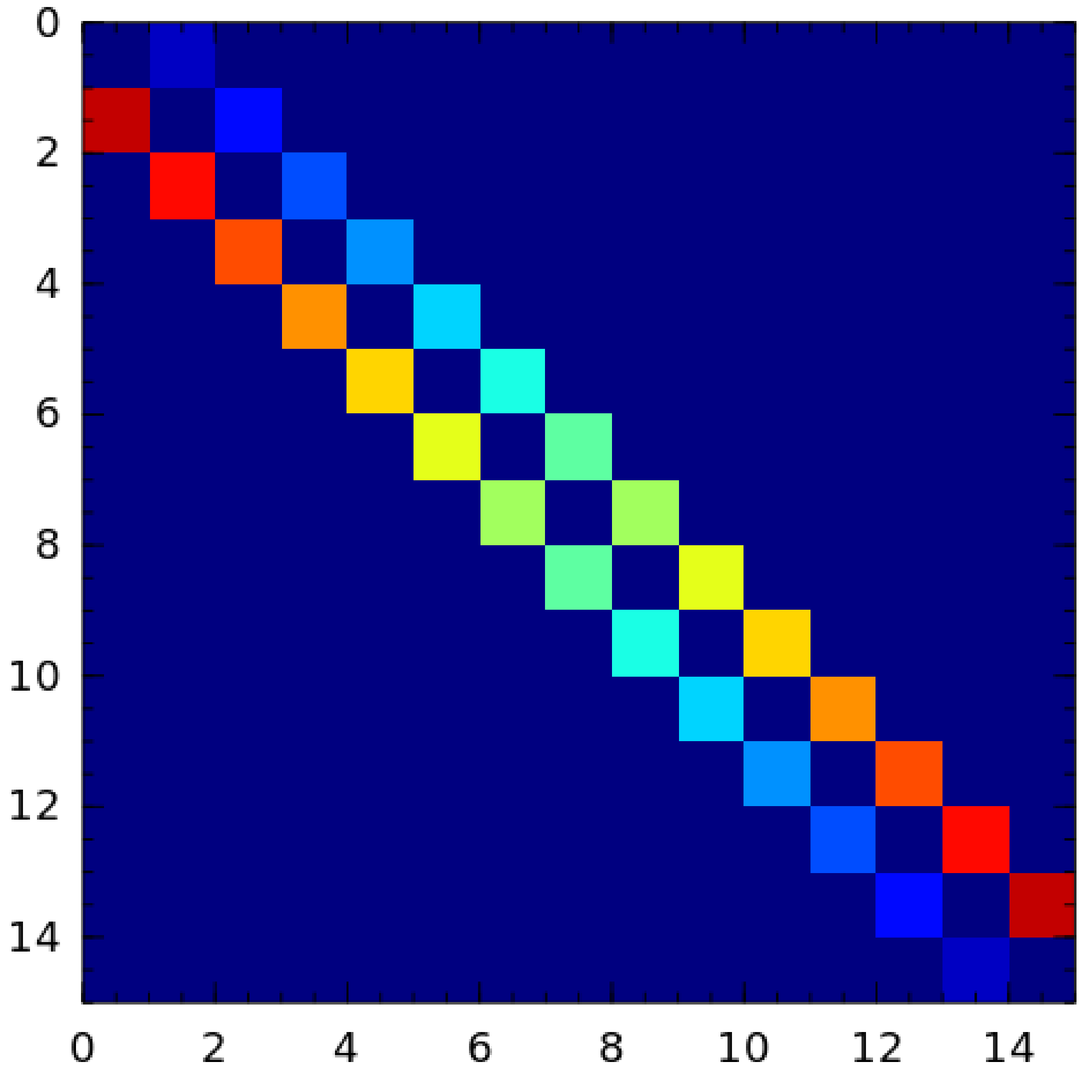
chow The Chow matrix is a singular Toeplitz lower Hessenberg matrix. The eigenvalues are known explicitly *[chow69]*.



circul A circulant matrix has the property that each row is obtained by cyclically permuting the entries of the previous row one step forward.



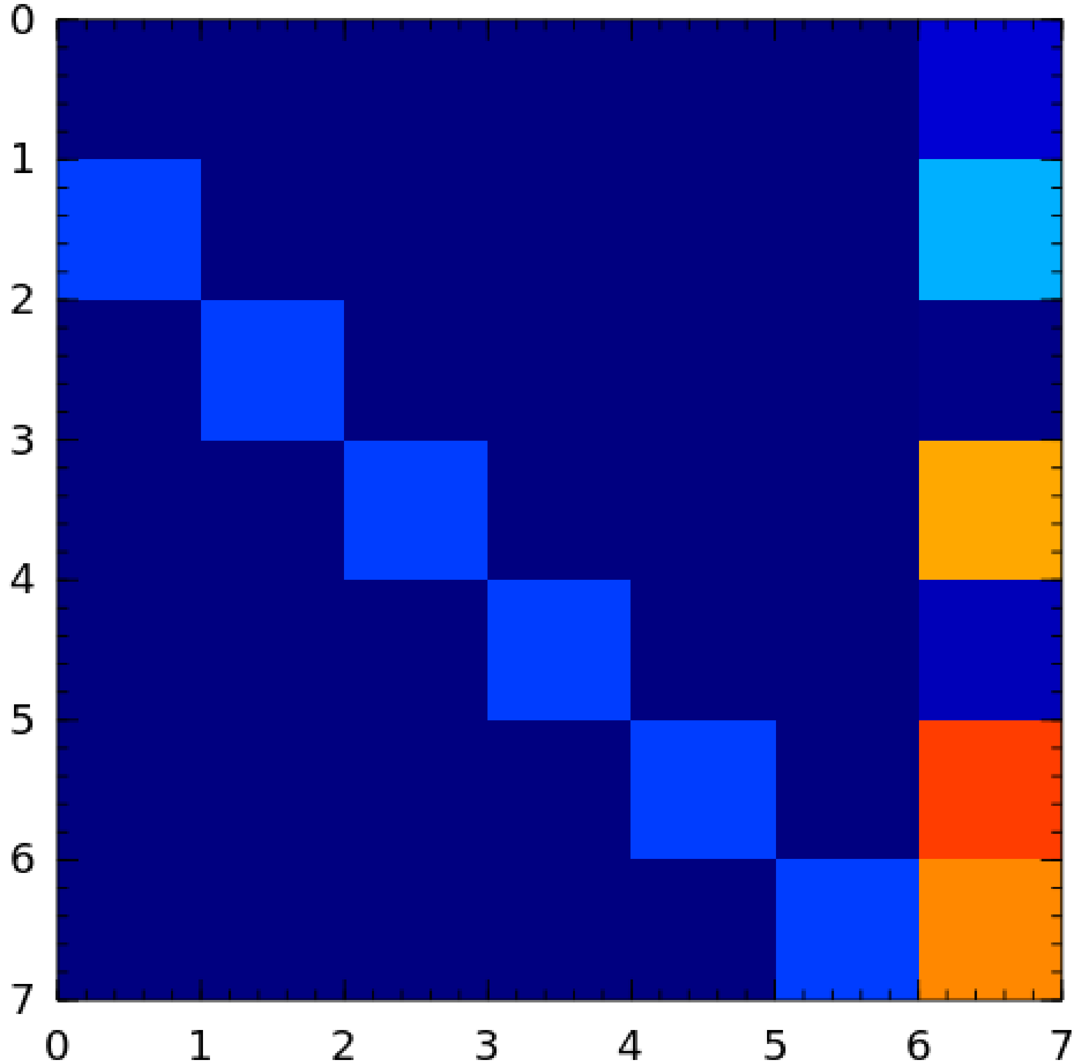
clement The Clement matrix *[clem59]* is a Tridiagonal matrix with zero diagonal entries. If `k = 1`, the matrix is symmetric.



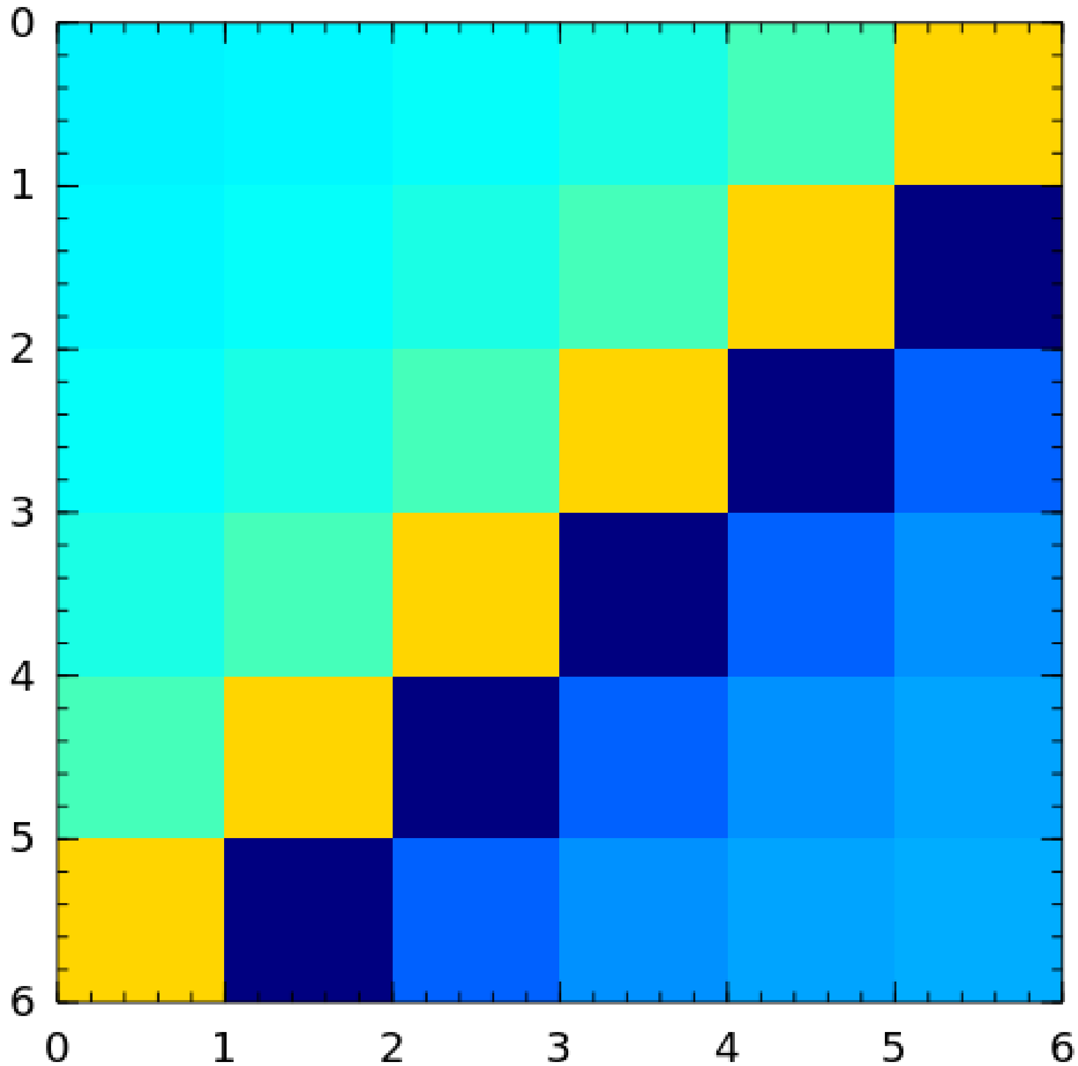
companion The companion matrix to a monic polynomial

$$a(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} + x^n$$

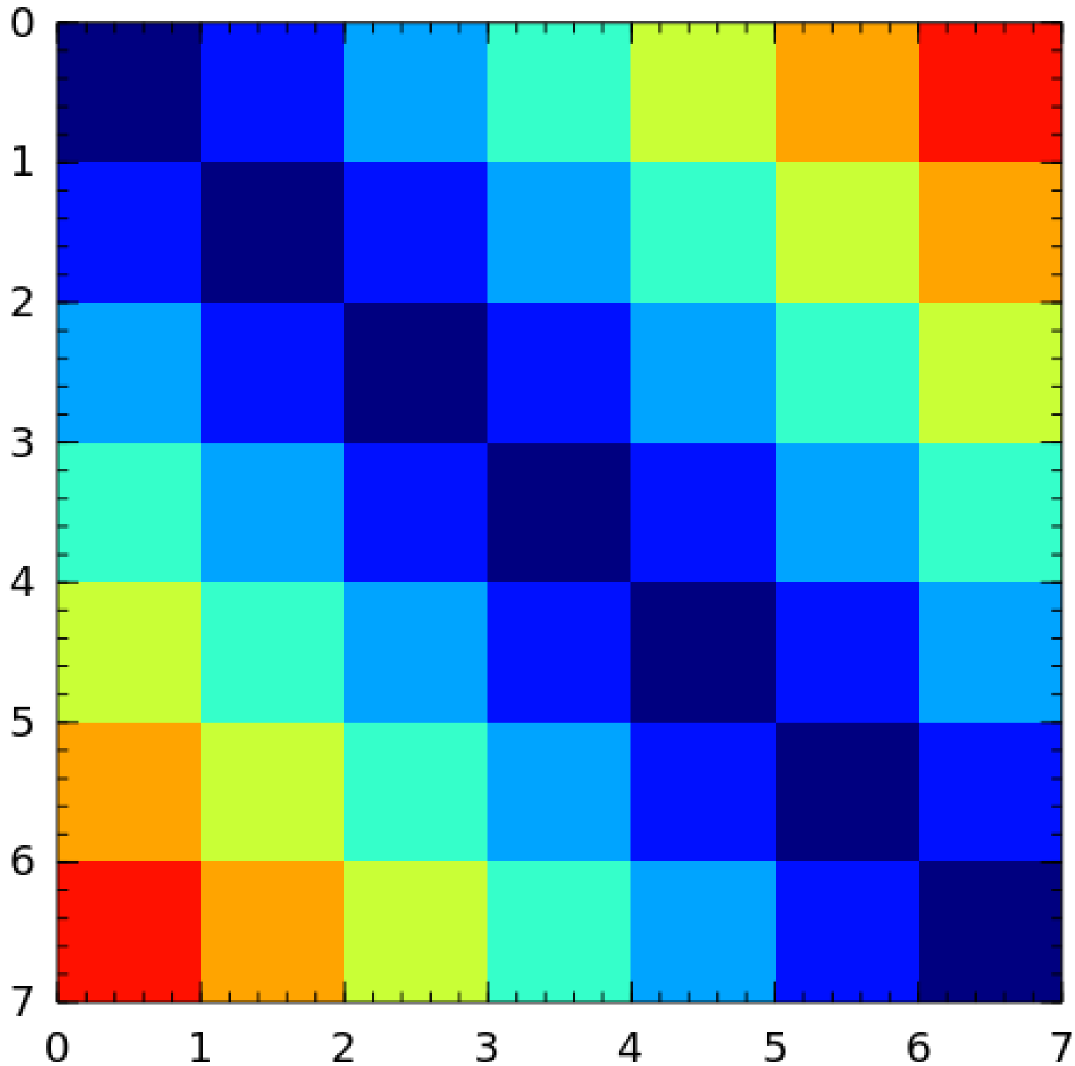
is the *n-by-n* matrix with ones on the subdiagonal and the last column given by the coefficients of *a(x)*.



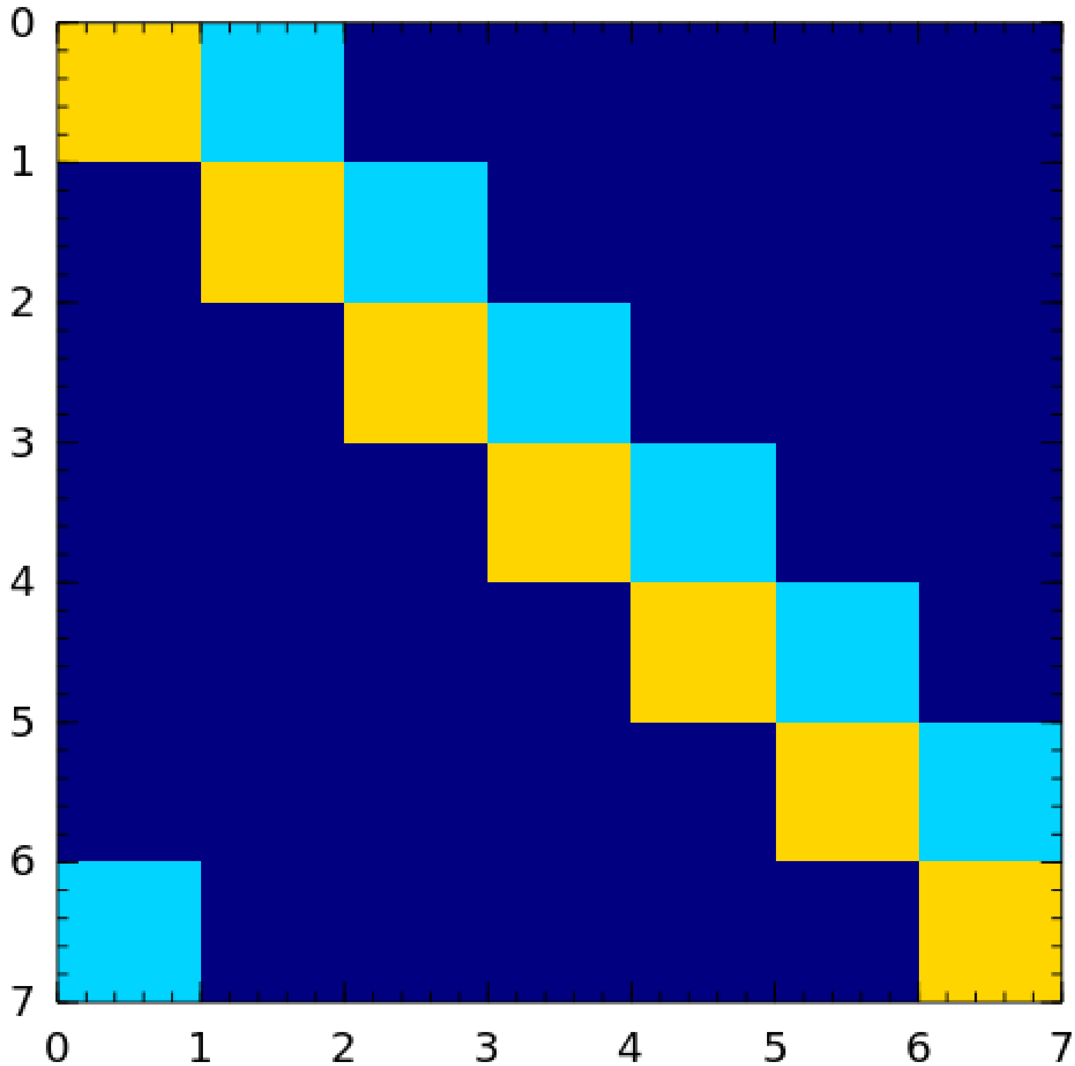
dingdong The Dingdong matrix is symmetric Hankel matrix invented by Dr. F. N. Ris of IBM, Thomas J Watson Research Centre. The eigenvalues cluster around $\pi/2$ and $-\pi/2$ *[nash90]*.



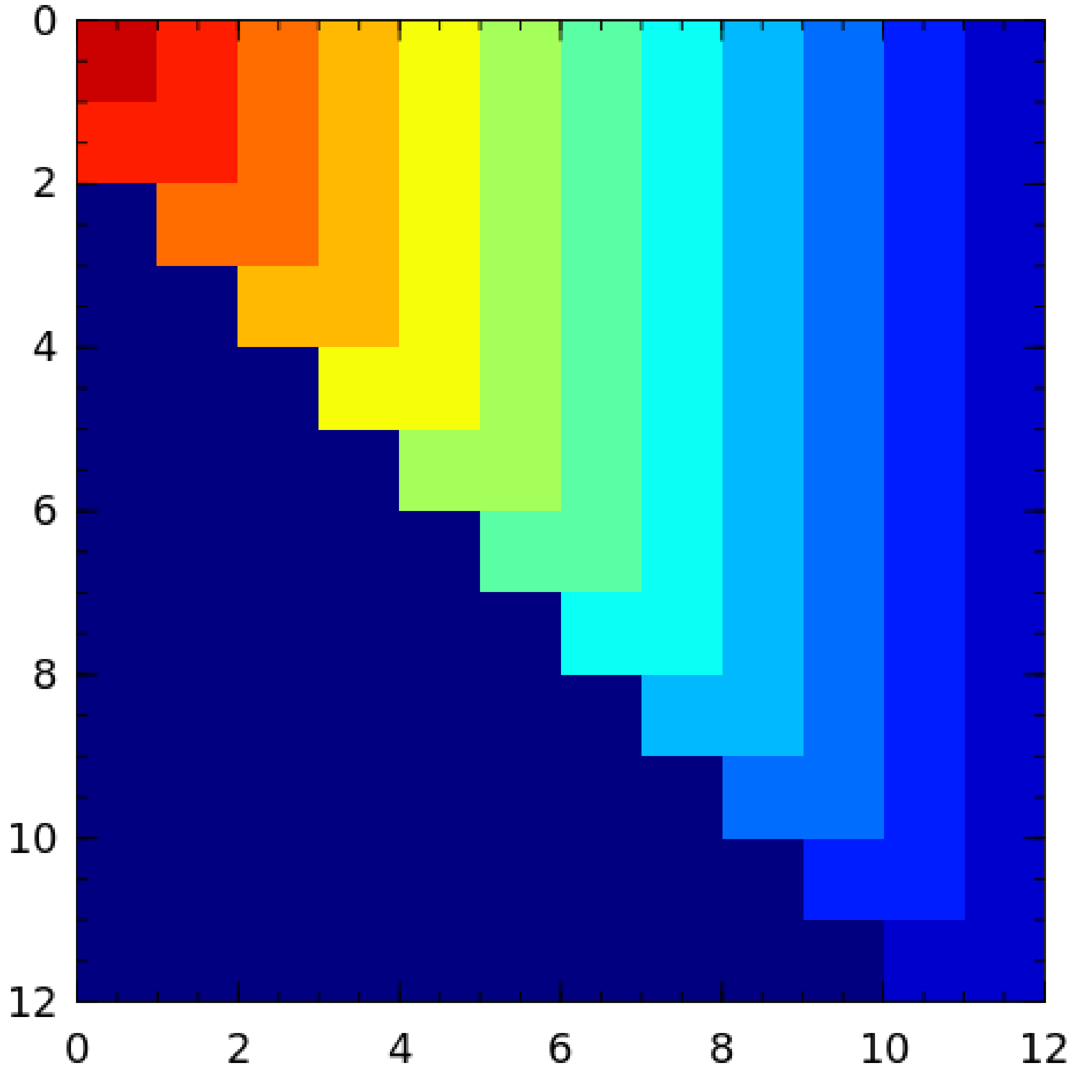
fiedler The Fiedler matrix is symmetric matrix with a dominant positive eigenvalue and all the other eigenvalues are negative. For explicit formulas for the inverse and determinant, see *[todd77]*.



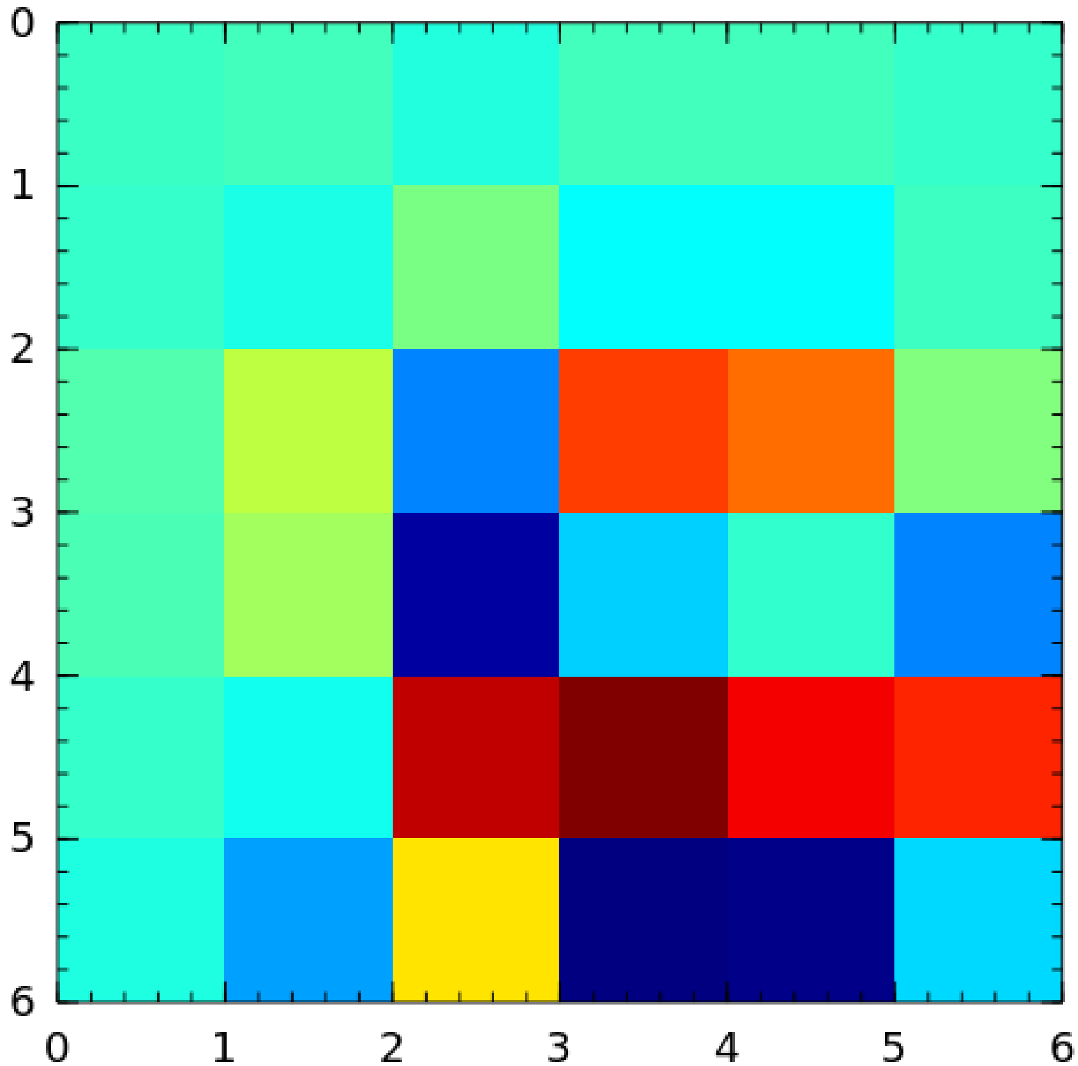
forsythe The Forsythe matrix is a n-by-n perturbed Jordan block.



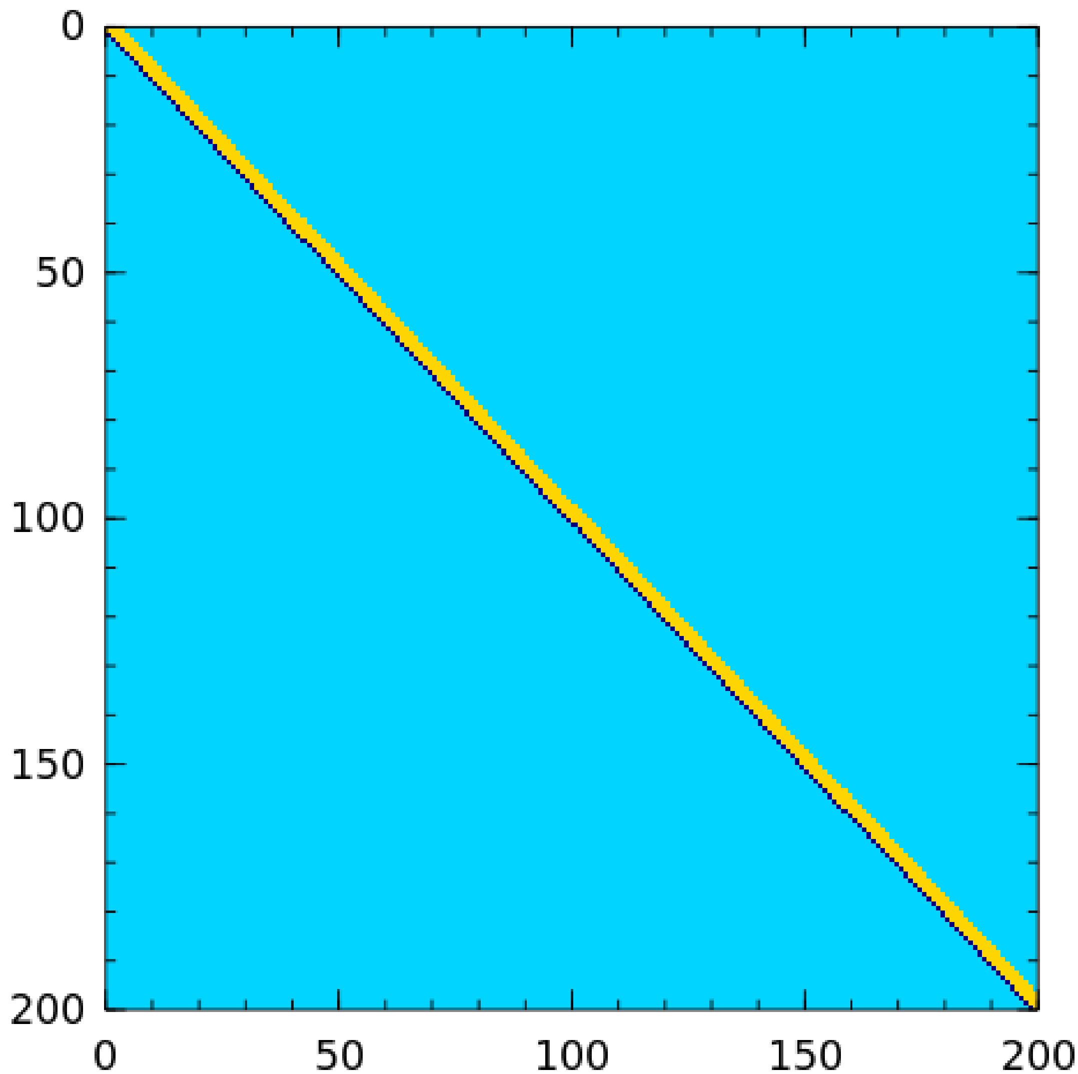
frank The Frank matrix is an upper Hessenberg matrix with determinant 1. The eigenvalues are real, positive and very ill conditioned *[vara86]*.



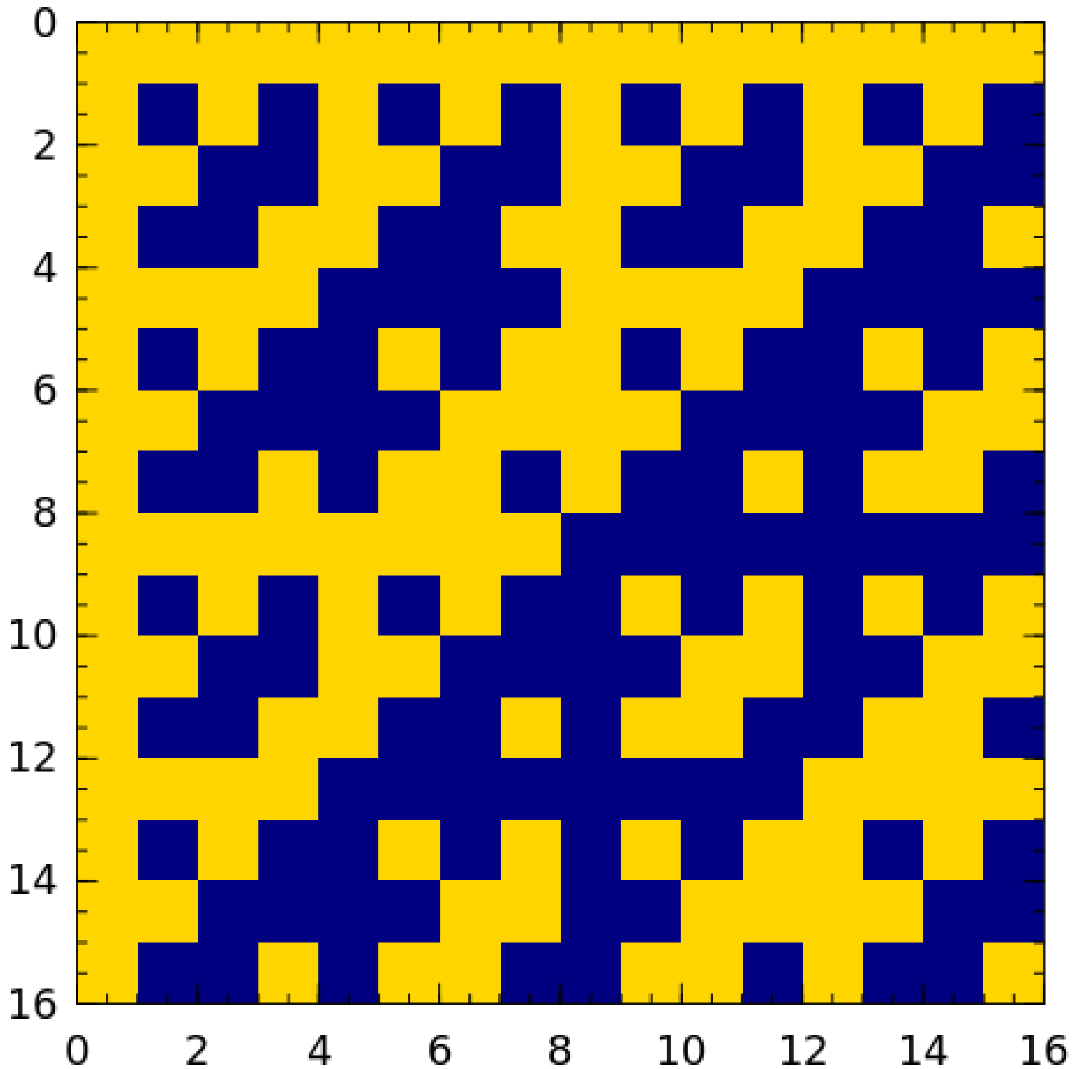
golub Golub matrix is the product of two random unit lower and upper triangular matrices respectively. LU factorization without pivoting fails to reveal that such matrices are badly conditioned *[vistre98]*.



grcar The Grcar matrix is a Toeplitz matrix with sensitive eigenvalues. The image below is a 200-by-200 Grcar matrix used in *[nrt92]*.



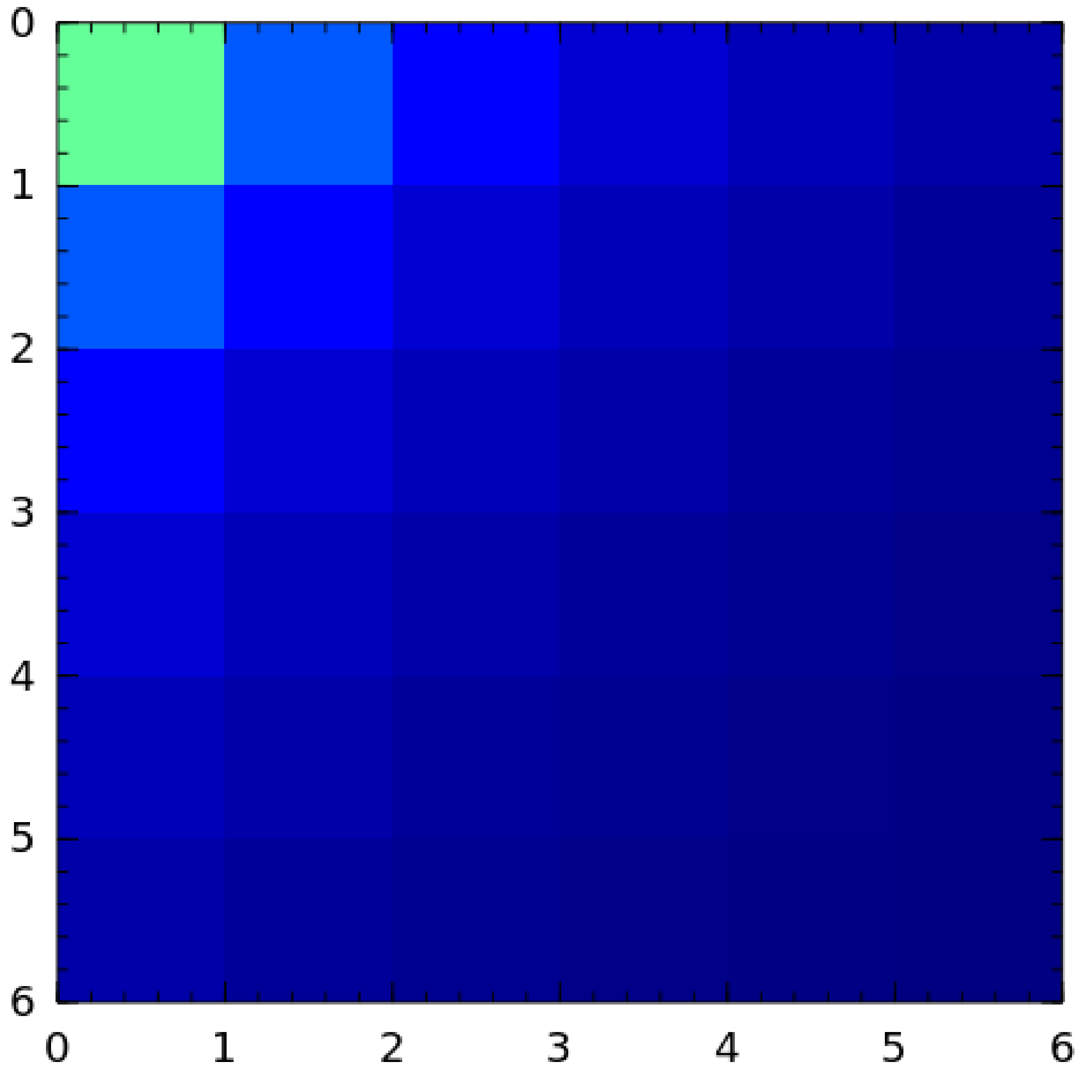
hadamard The Hadamard matrix is a square matrix whose entries are 1 or -1. It was named after Jacques Hadamard. The rows of a Hadamard matrix are orthogonal.



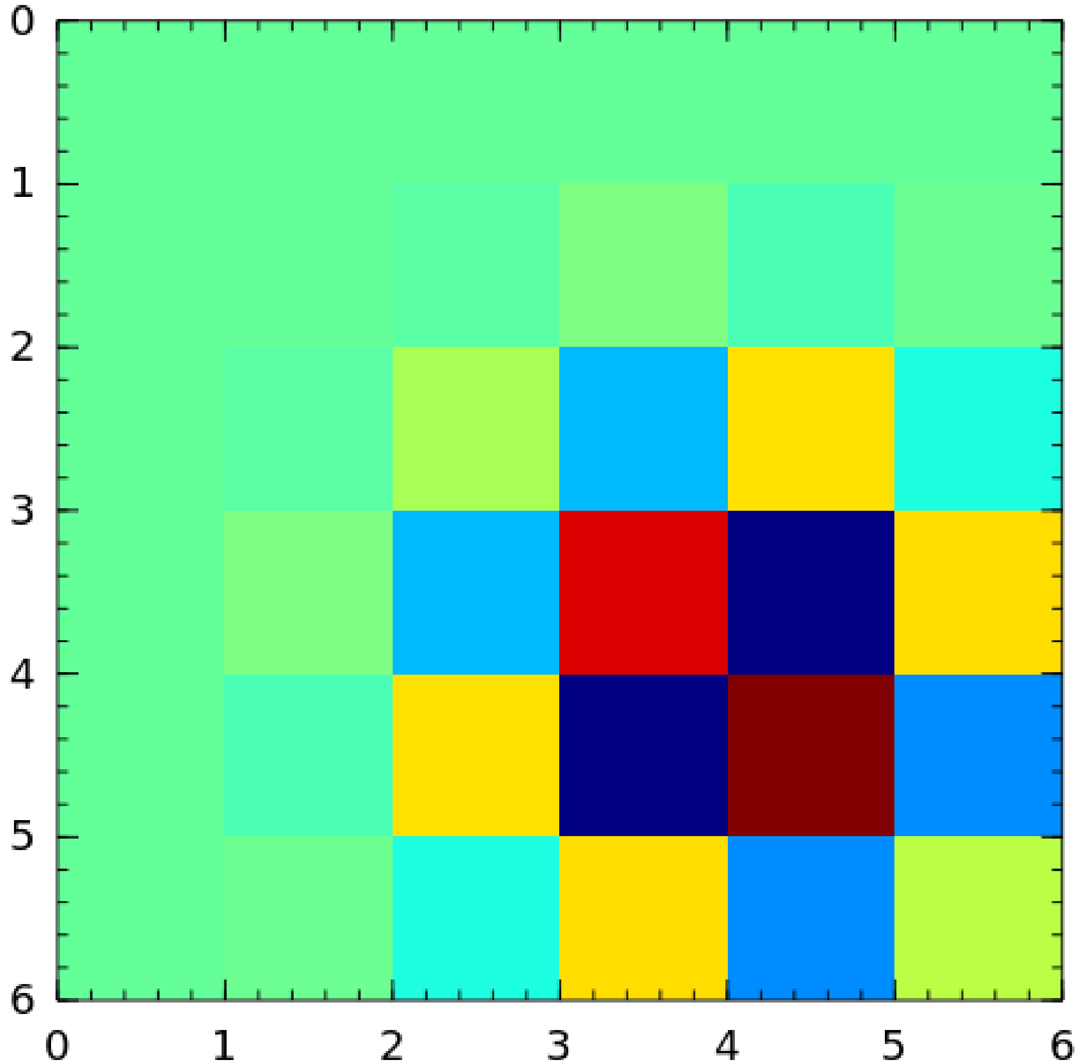
hankel Hankel matrix is a a matrix that is symmetric and constant across the anti-diagonals. For example:

```
julia> matrixdepot("hankel", [1,2,3,4], [7,8,9,10])
4x4 Array{Float64,2}:
1.0  2.0  3.0   4.0
2.0  3.0  4.0   8.0
3.0  4.0  8.0   9.0
4.0  8.0  9.0  10.0
```

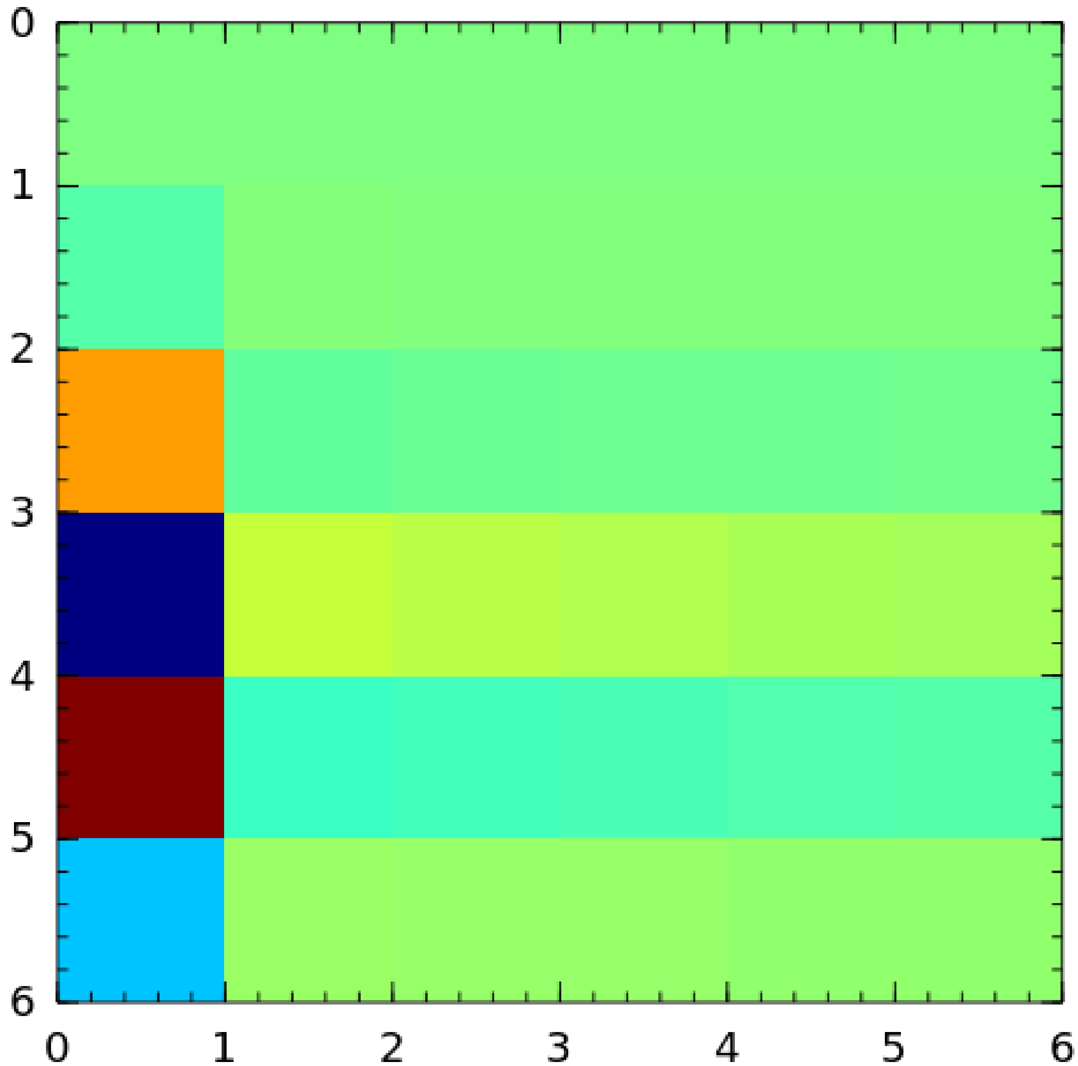
hilb The Hilbert matrix is a very ill conditioned matrix. But it is symmetric positive definite and totally positive so it is not a good test matrix for Gaussian elimination *[high02]* (Sec. 28.1).



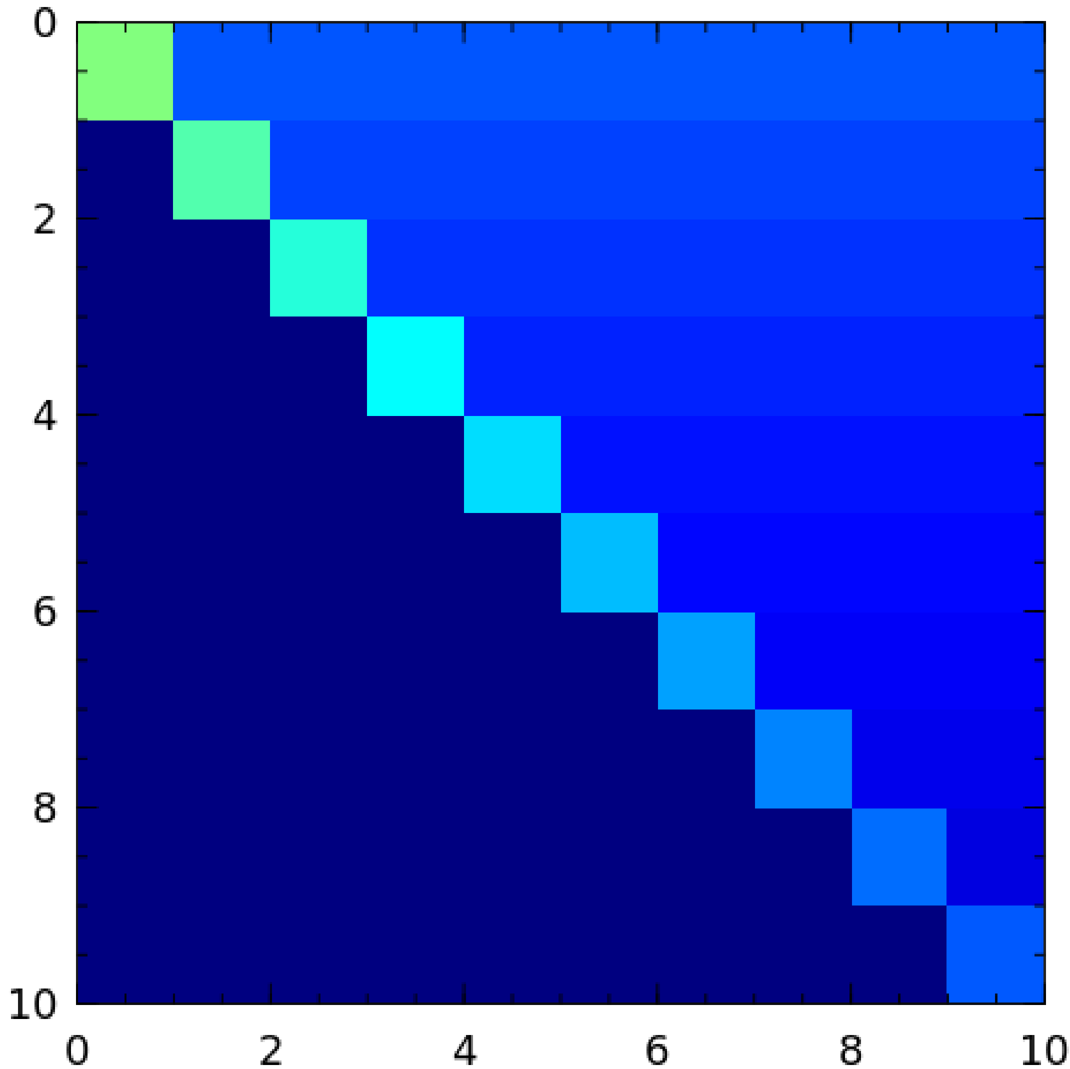
invhilb Inverse of the Hilbert Matrix.



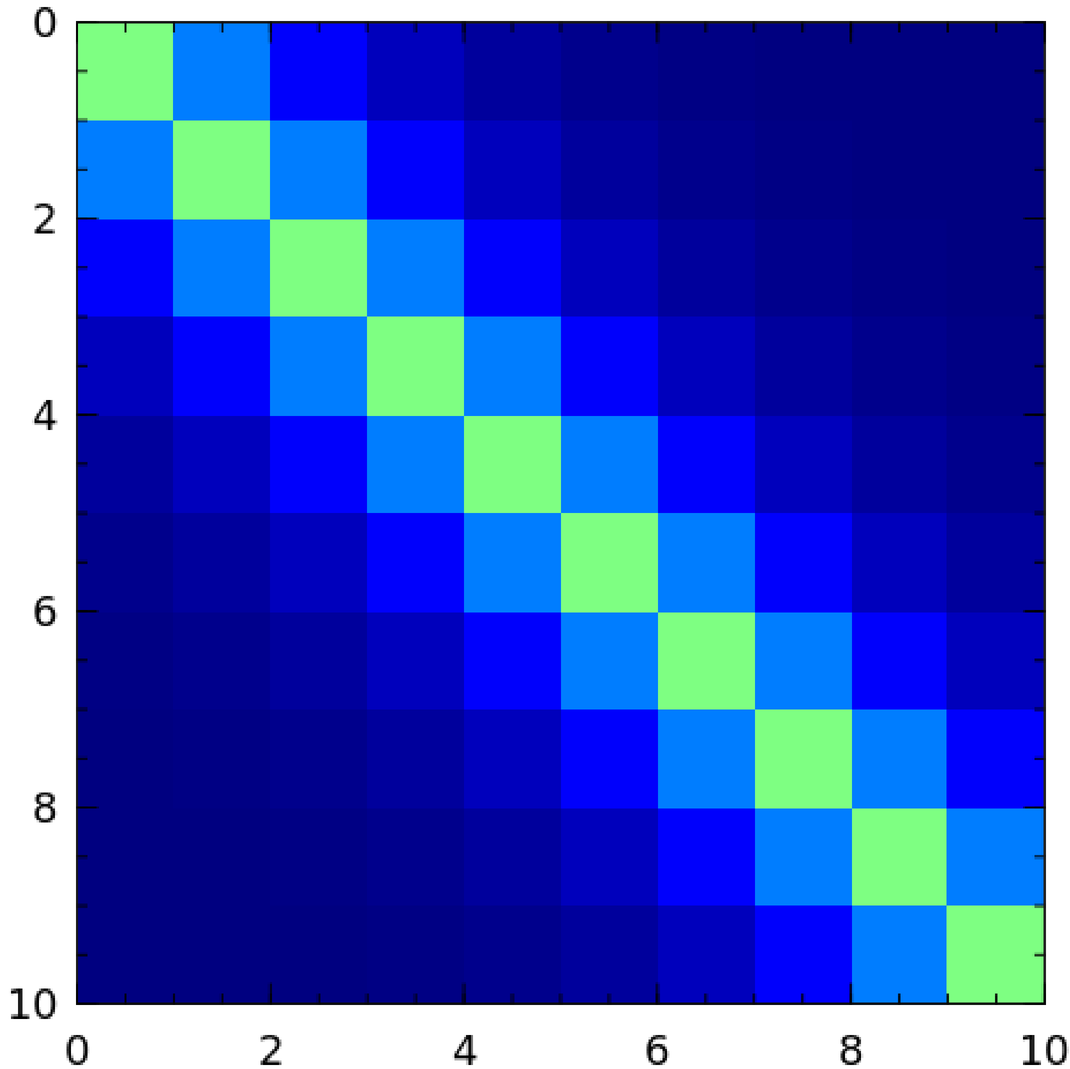
invol An involutory matrix, i.e., a matrix that is its own inverse. See *[hoca63]*.



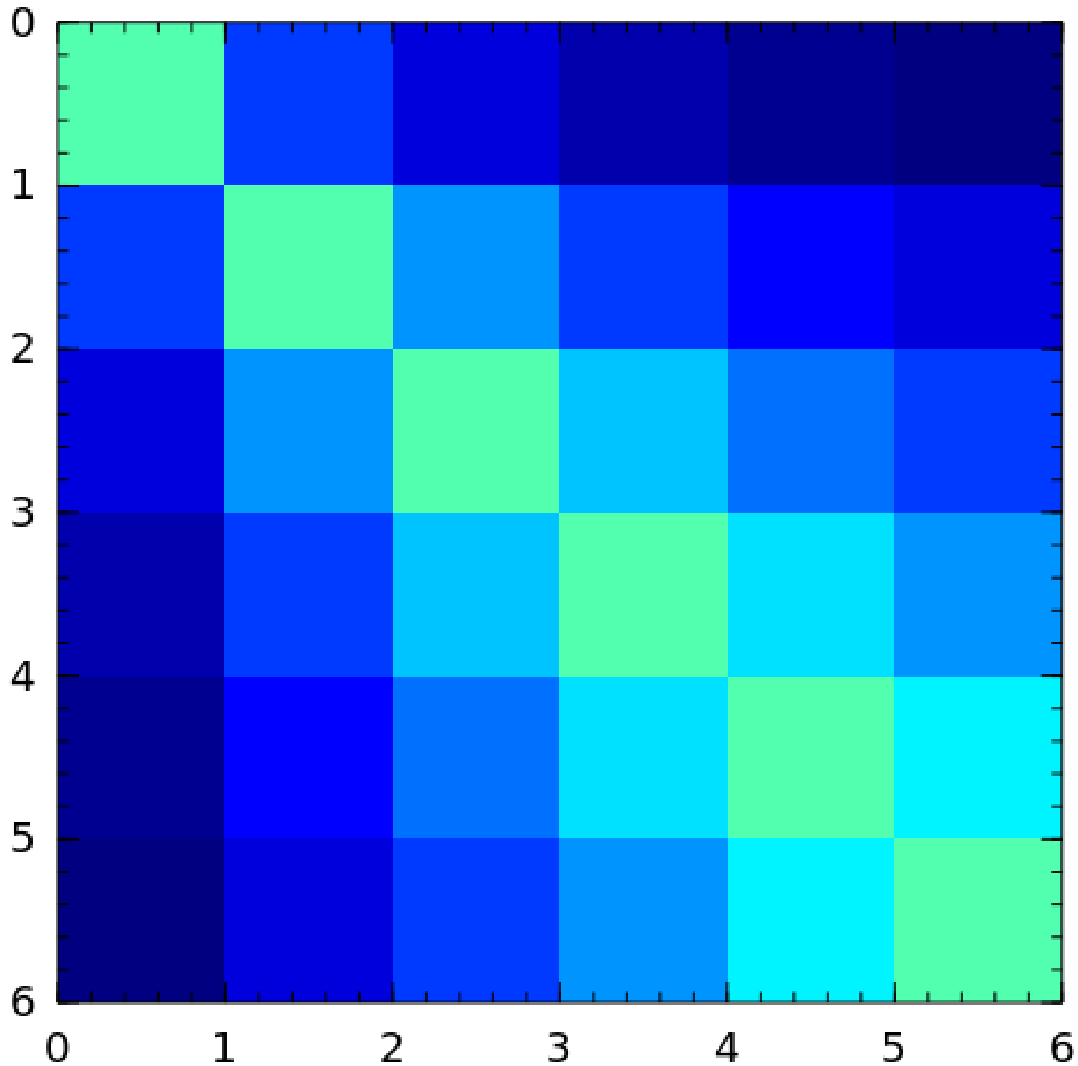
kahan The Kahan matrix is a upper trapezoidal matrix, i.e., the (i, j) element is equal to 0 if $i > j$. The useful range of `theta` is $0 < theta < \pi$. The diagonal is perturbed by `pert*eps()*diagm([n:-1:1])`.



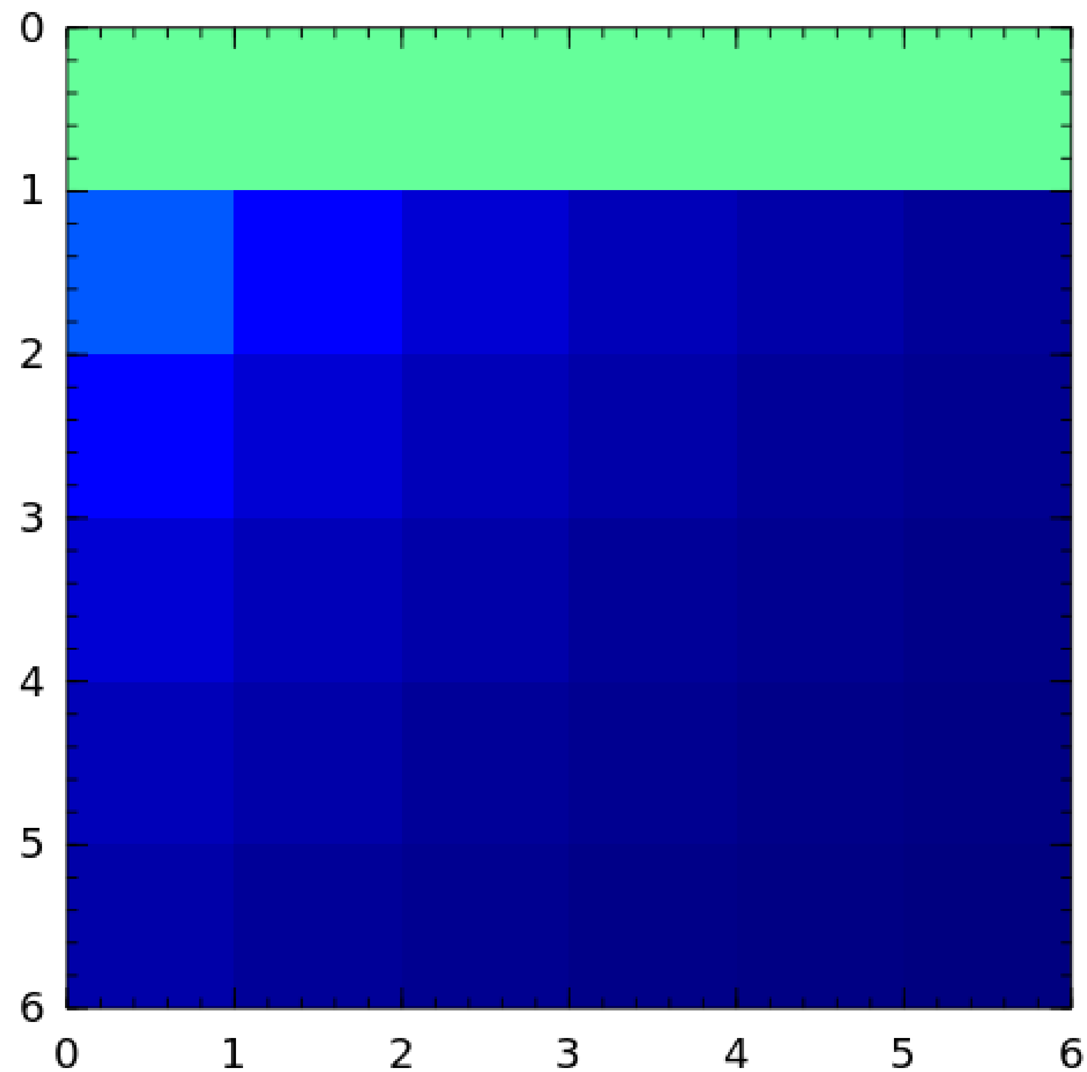
kms Kac-Murdock-Szego Toeplitz matrix *[tren89]*.



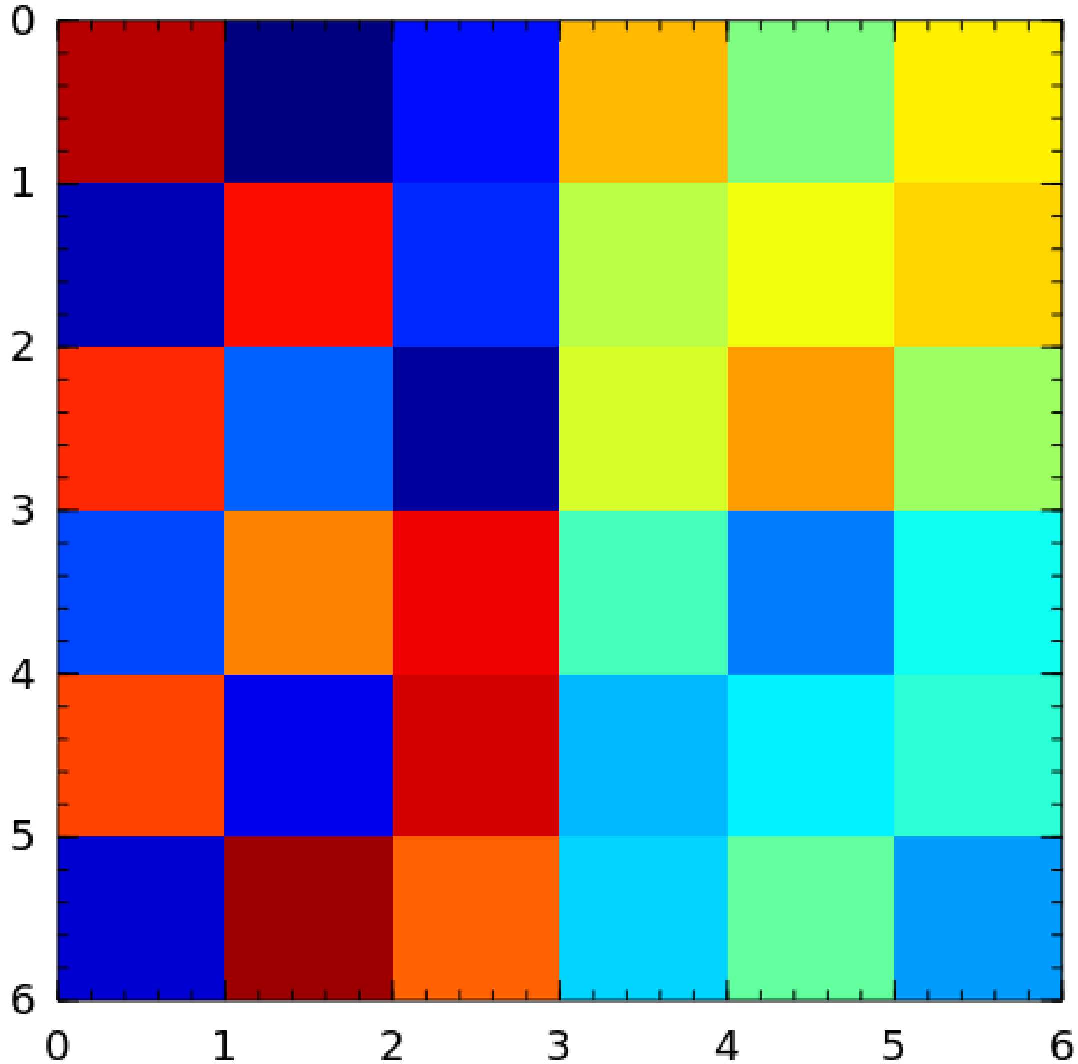
lehmer The Lehmer matrix is a symmetric positive definite matrix. It is totally nonnegative. The inverse is tridiagonal and explicitly known *[neto58]*.



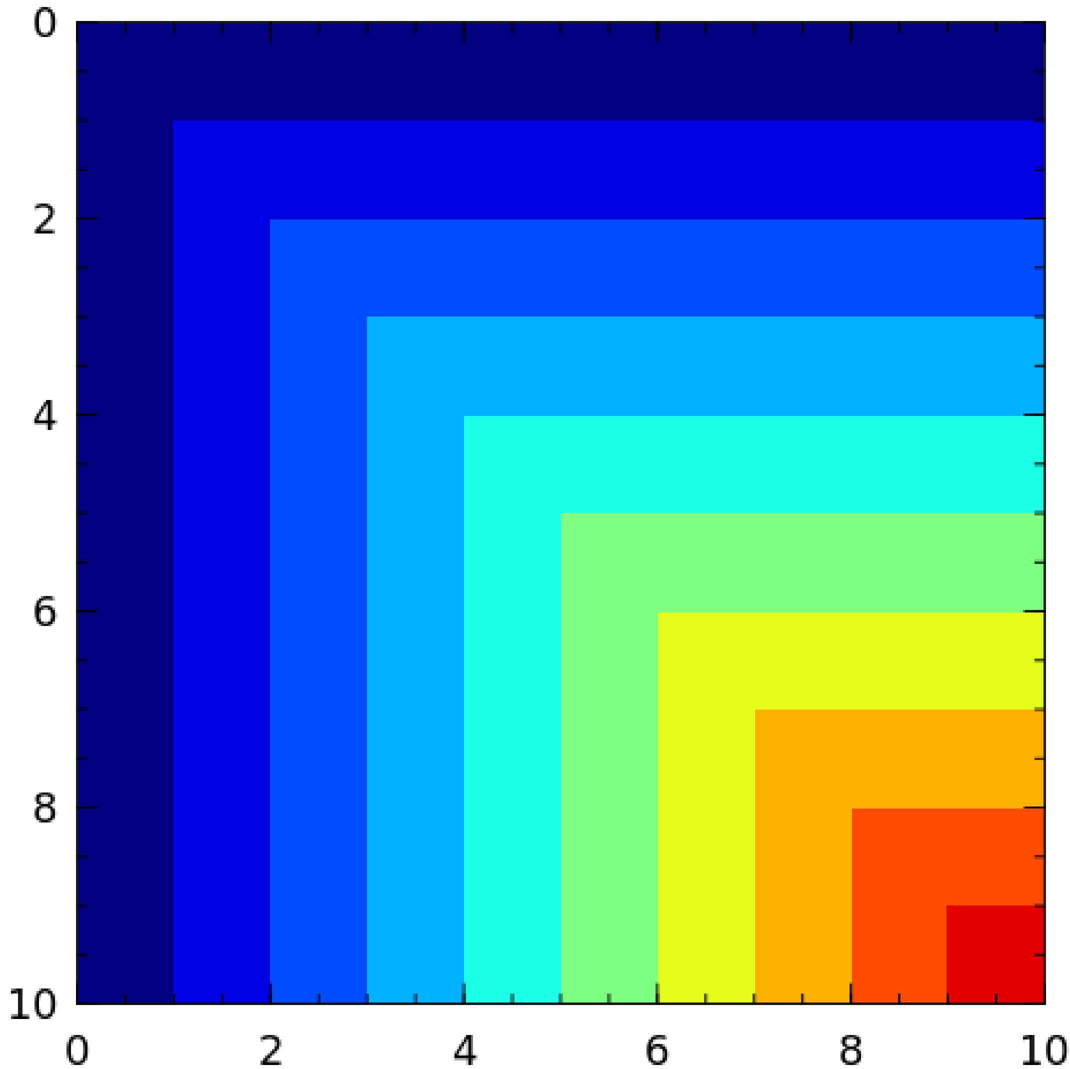
lotkin The Lotkin matrix is the Hilbert matrix with its first row altered to all ones. It is unsymmetric, ill-conditioned and has many negative eigenvalues of small magnitude *[lotk55]*.



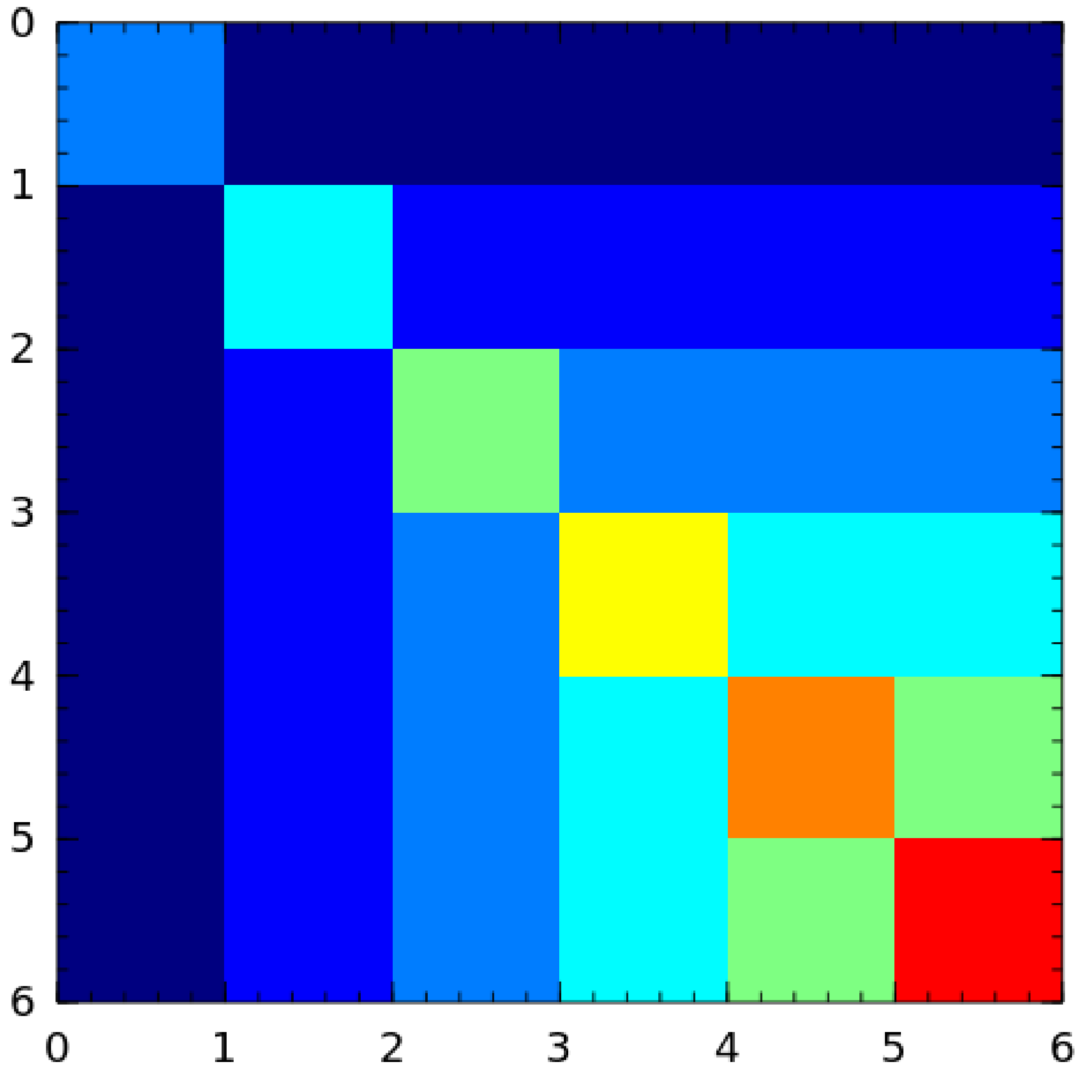
magic The magic matrix is a matrix with integer entries such that the row elements, column elements, diagonal elements and anti-diagonal elements all add up to the same number.



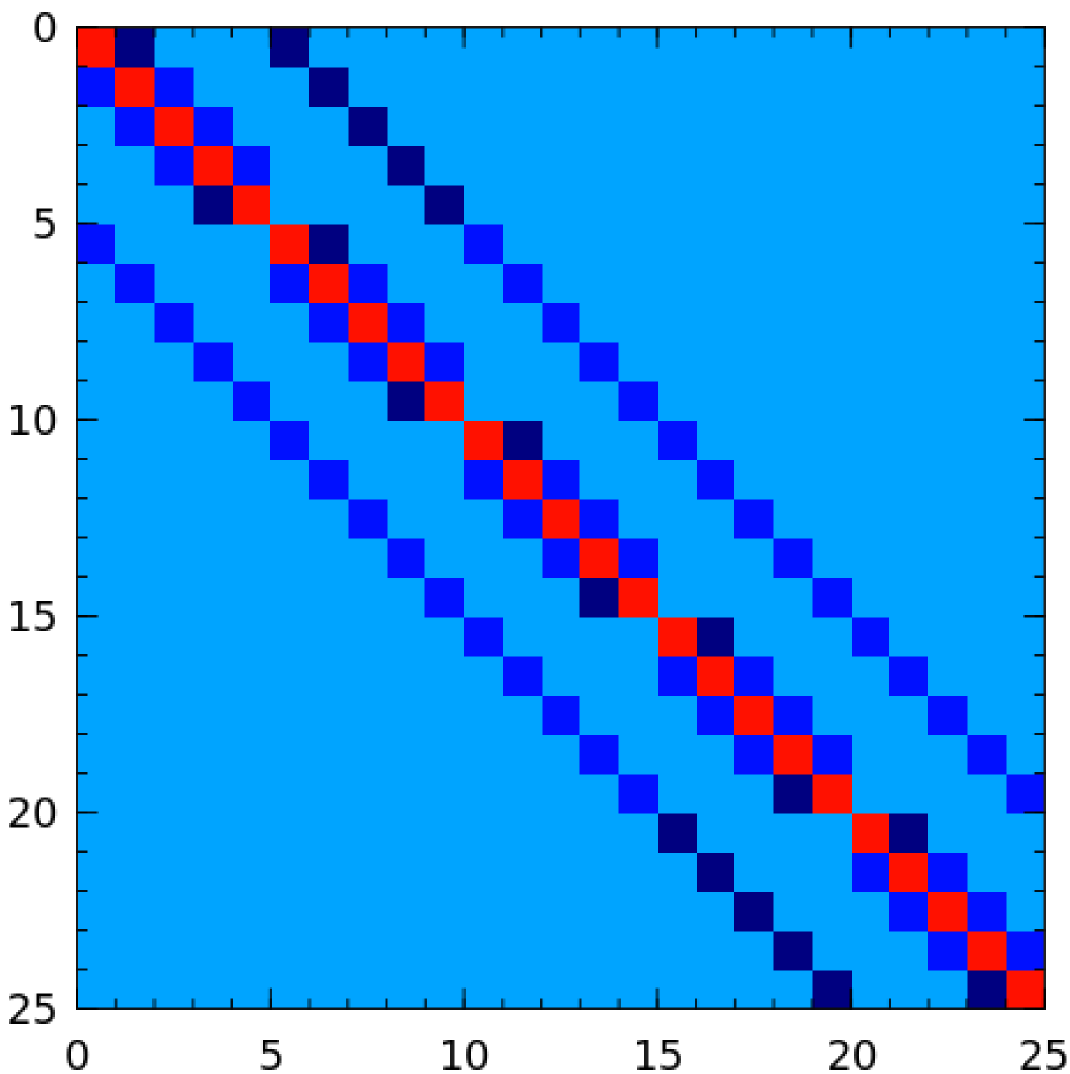
minij A matrix with (i, j) entry `min(i, j)`. It is a symmetric positive definite matrix. The eigenvalues and eigenvectors are known explicitly. Its inverse is tridiagonal.



moler The Moler matrix is a symmetric positive definite matrix. It has one small eigenvalue.



neumann A singular matrix from the discrete Neumann problem. This matrix is sparse and the null space is formed by a vector of ones *[plem76]*.



oscillate A matrix A is called oscillating if A is totally nonnegative and if there exists an integer q > 0 such that A^q is totally positive. An $n \times n$ oscillating matrix A satisfies:

1. A has n distinct and positive eigenvalues $\lambda_1 > \lambda_2 > \cdots > \lambda_n > 0$.
2. The i th eigenvector, corresponding to λ_i in the above ordering, has exactly $i - 1$ sign changes.

This function generates a symmetric oscillating matrix, which is useful for testing numerical regularization methods *[hansen95]*. For example:

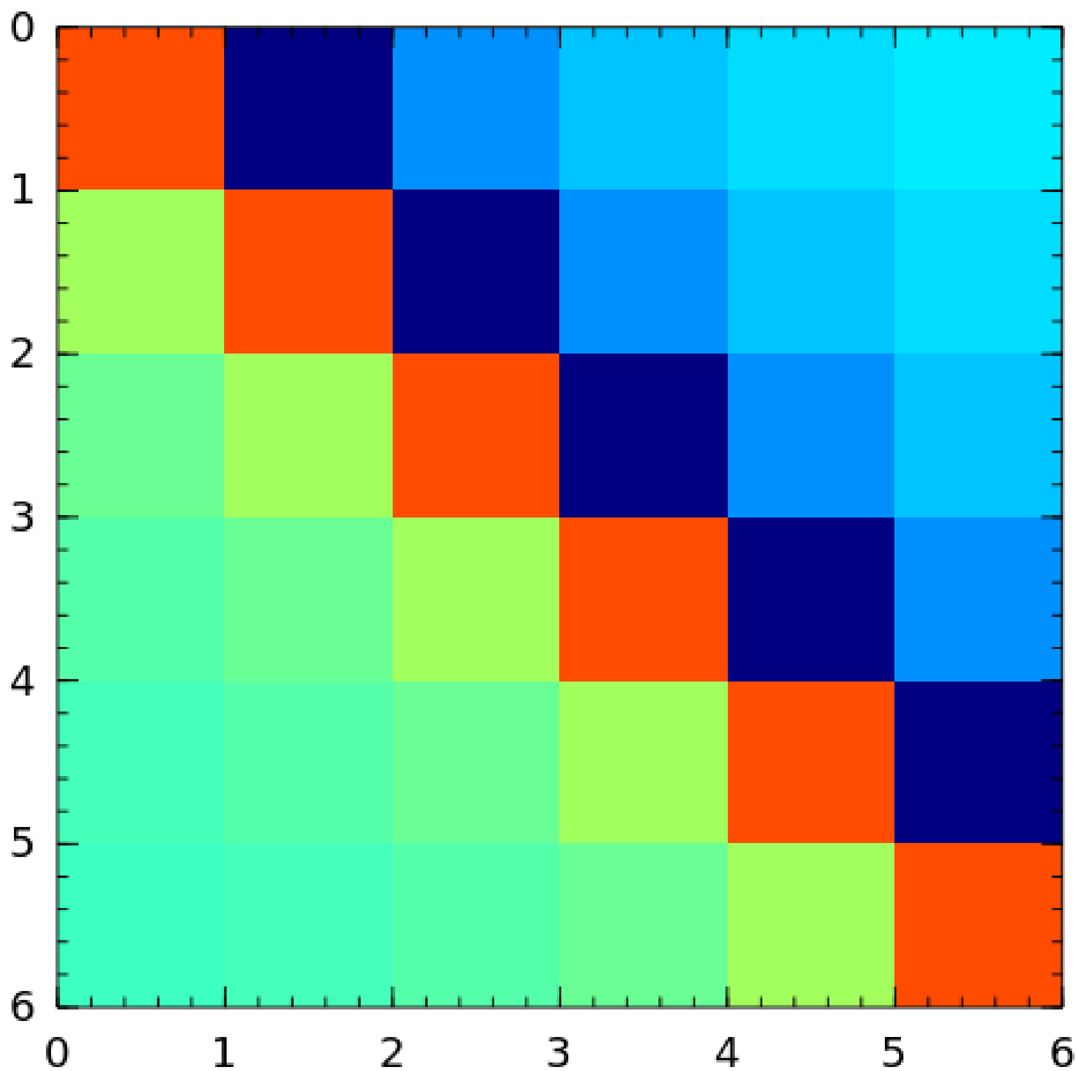
```
julia> A = matrixdepot("oscillate", 3)
3x3 Array{Float64,2}:
0.98694    0.112794   0.0128399
0.112794   0.0130088  0.0014935
0.0128399  0.0014935  0.00017282

julia> eig(A)
([1.4901161192617526e-8,0.00012207031249997533,0.9999999999999983],
```

(continues on next page)

(continued from previous page)

```
3x3 Array{Float64,2}:
0.0119607   0.113658  -0.993448
-0.215799   -0.969813  -0.113552
0.976365   -0.215743  -0.0129276)
```

parter The Parter matrix is a Toeplitz and Cauchy matrix with singular values near π *[part86]*.



pascal The Pascal matrix's anti-diagonals form the Pascal's triangle:

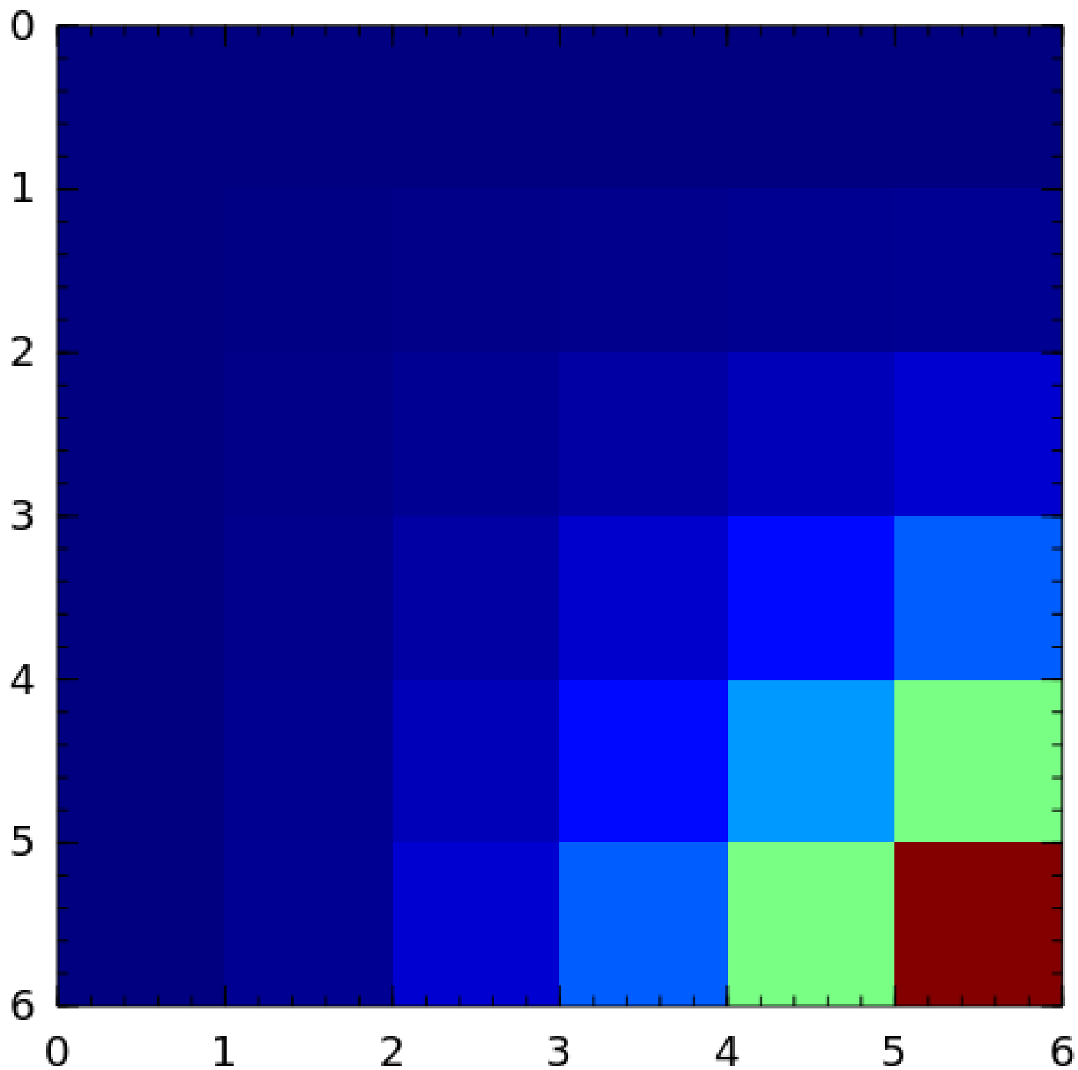
```
julia> matrixdepot("pascal", 6)
6x6 Array{Int64,2}:
1  1   1   1    1    1
1  2   3   4    5    6
1  3   6  10   15   21
1  4  10  20   35   56
1  5  15  35   70  126
```

(continues on next page)

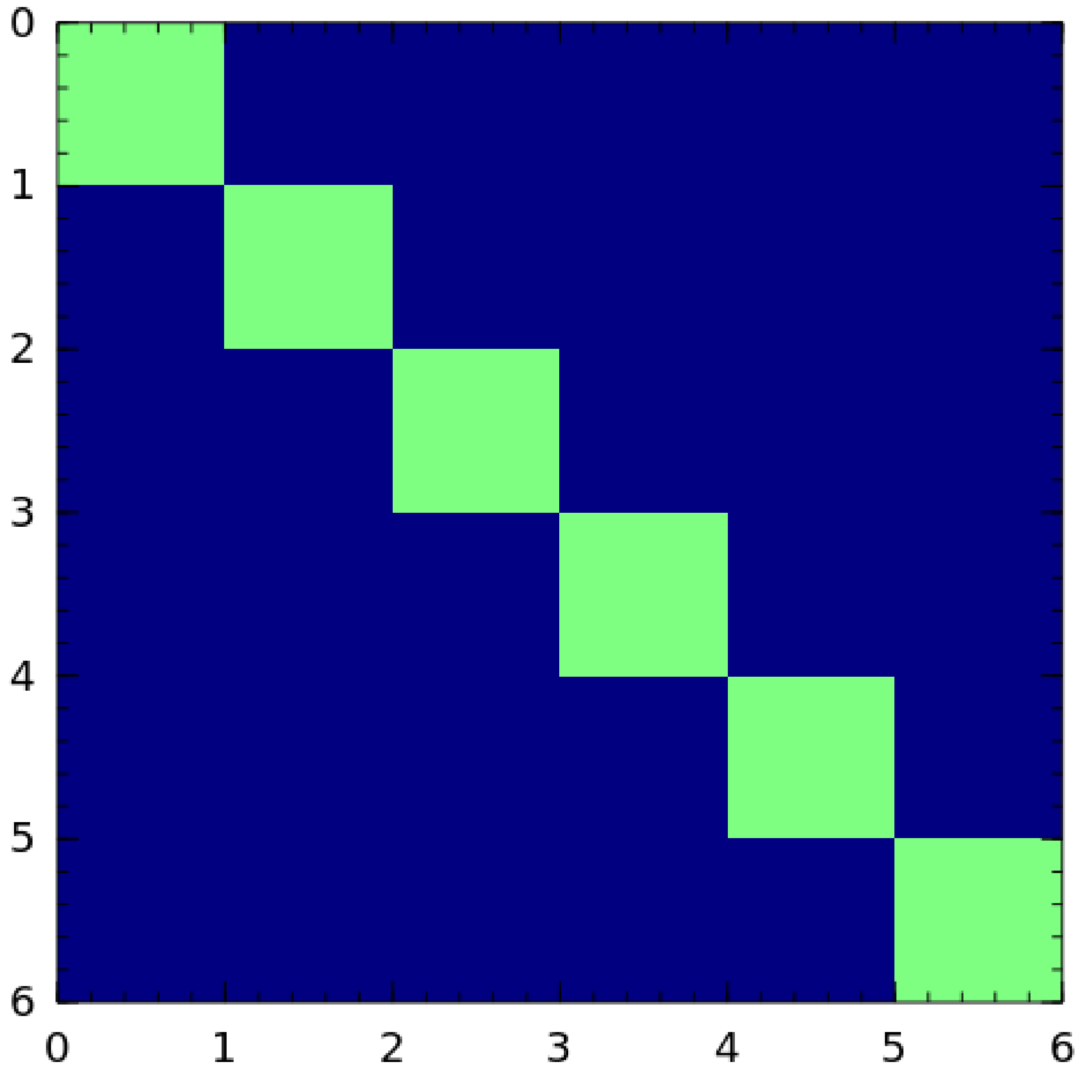
(continued from previous page)

```
1   6   21   56   126   252
```

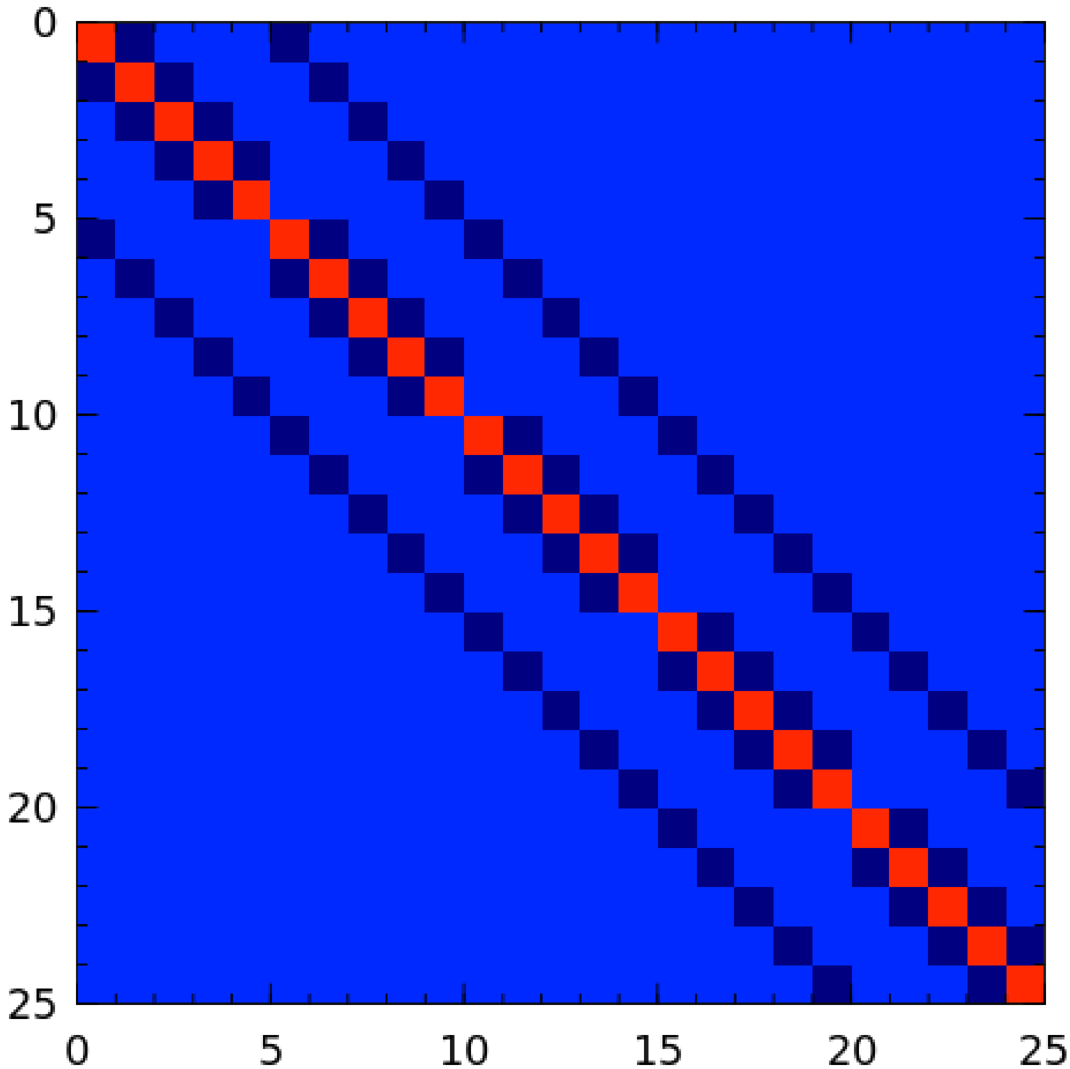
See *[high02]* (28.4).



pei The Pei matrix is a symmetric matrix with known inverse *[pei62]*.



poisson A block tridiagonal matrix from Poisson's equation. This matrix is sparse, symmetric positive definite and has known eigenvalues.

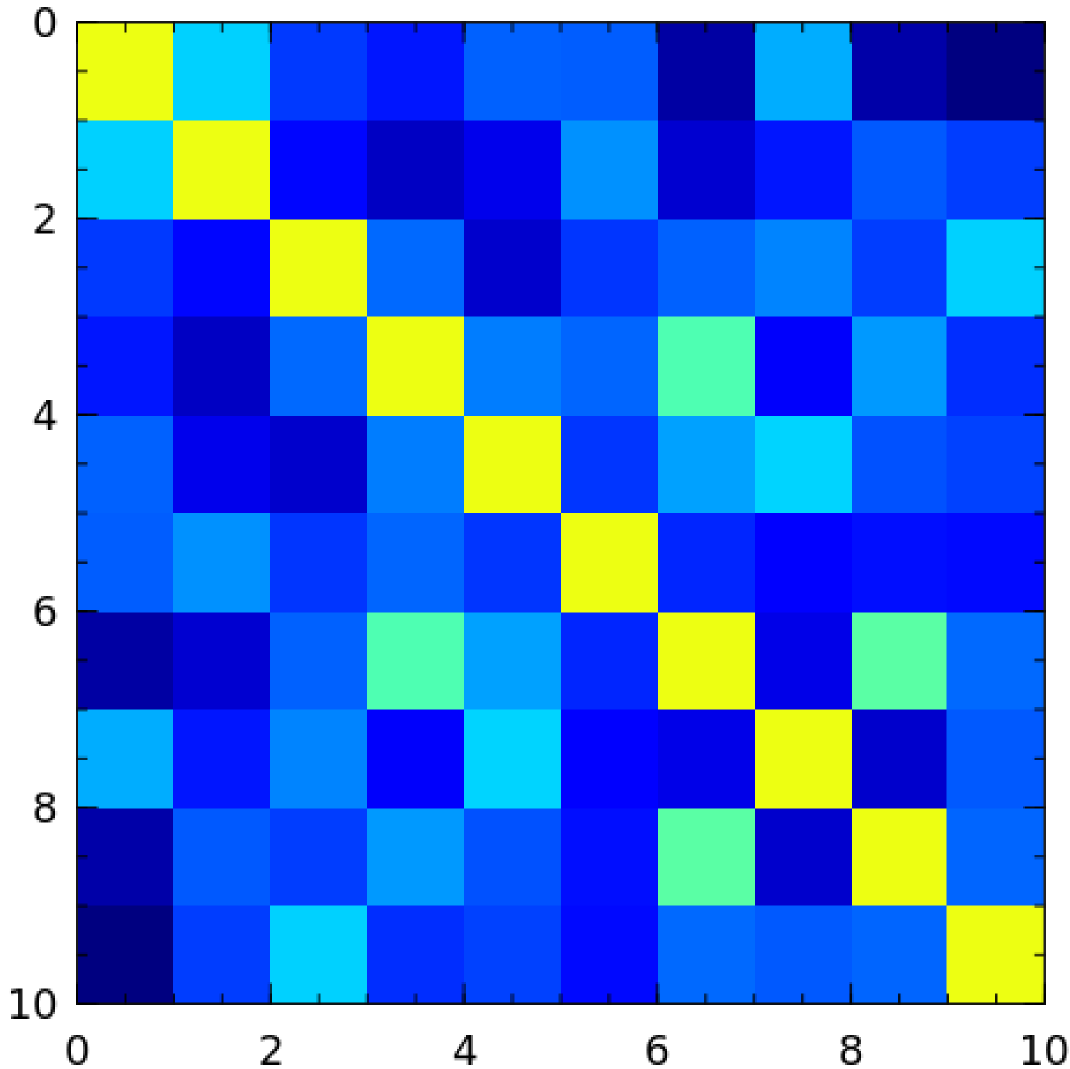


prolate A prolate matrix is a symmetric ill-conditioned Toeplitz matrix

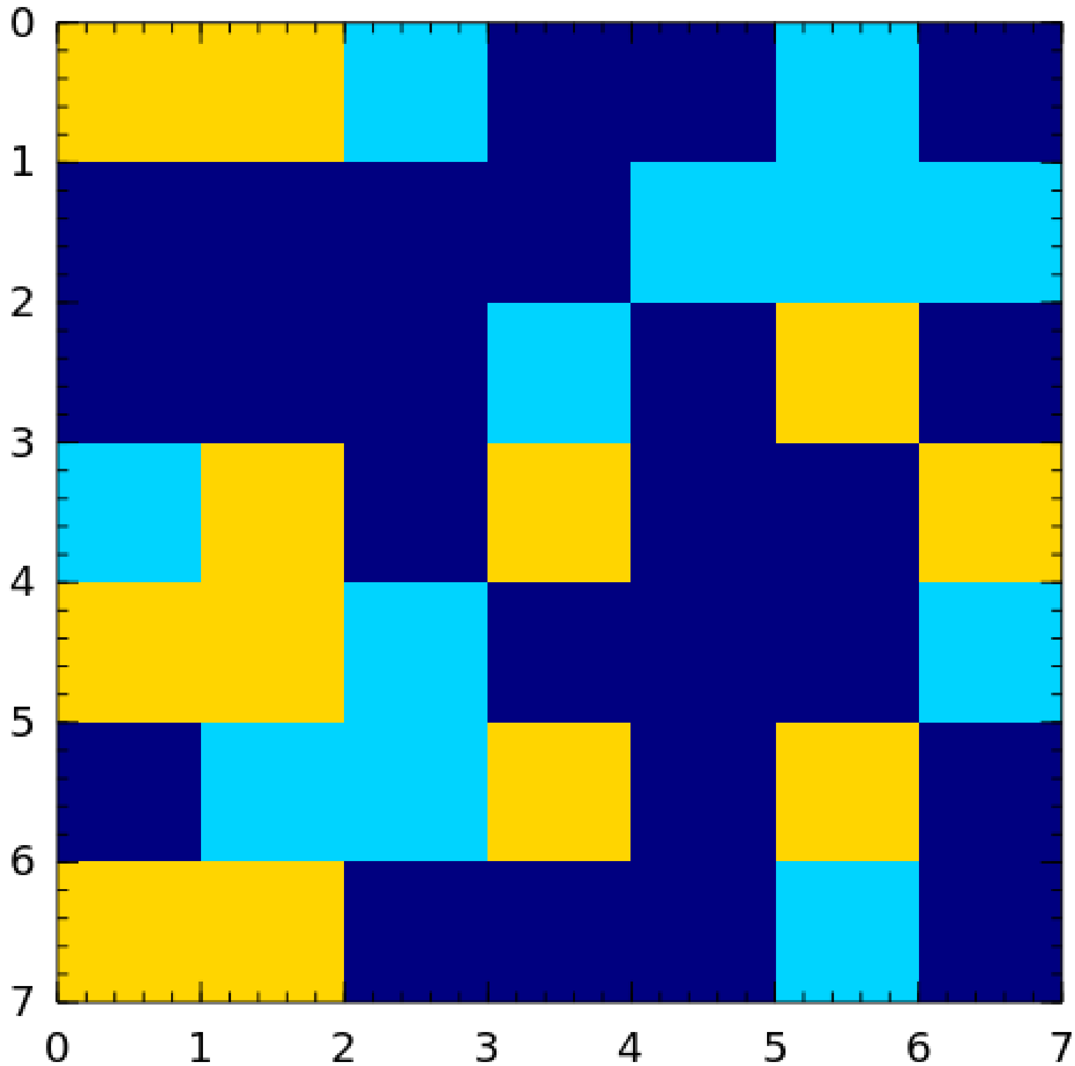
$$A = \begin{bmatrix} a_0 & a_1 & \cdots \\ a_1 & a_0 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

such that $a_0 = 2w$ and $a_k = (\sin 2\pi wk)/\pi k$ for $k = 1, 2, \ldots$ and $0 < w < 1/2$ *[varah93]*.

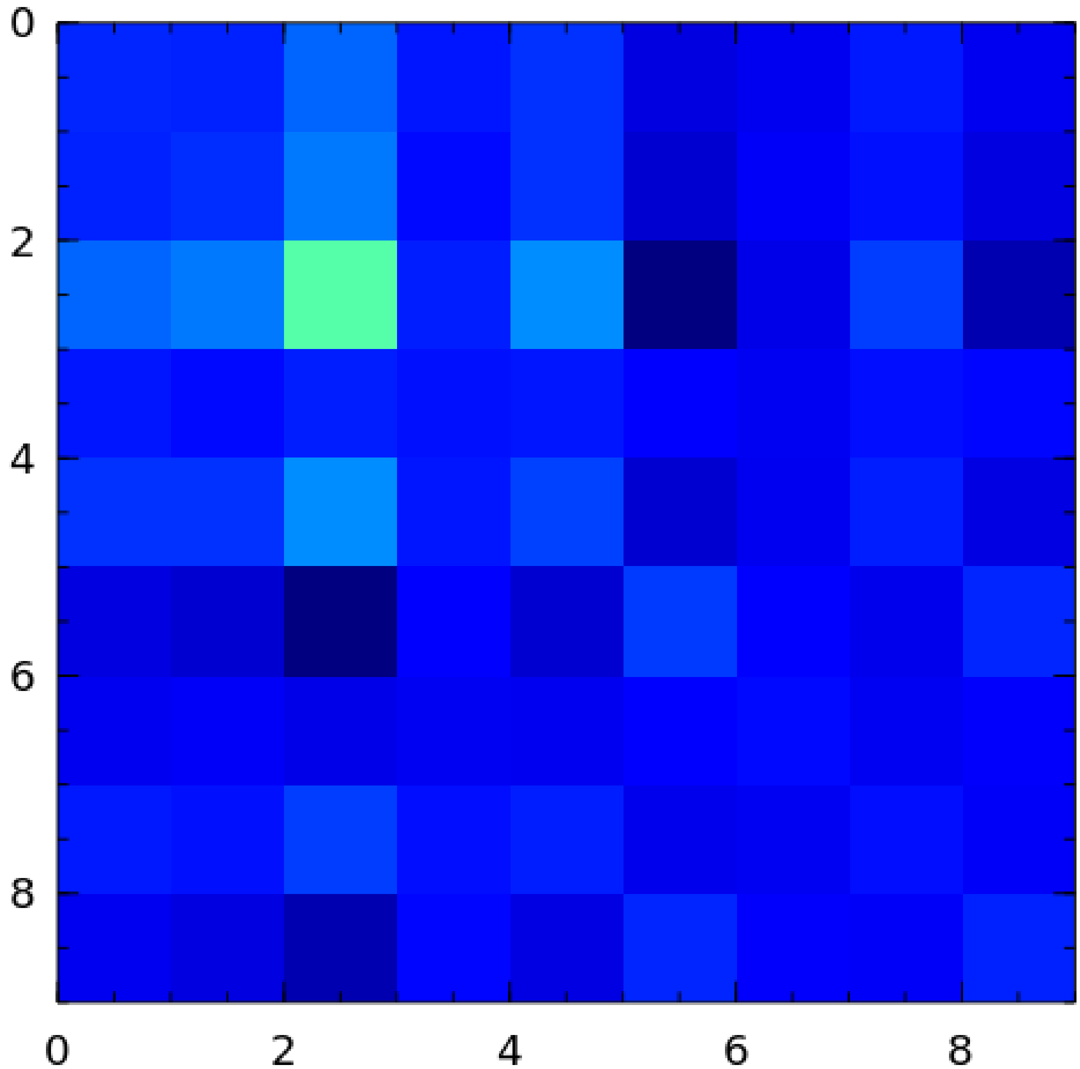
randcorr A random correlation matrix is a symmetric positive semidefinite matrix with 1s on the diagonal.



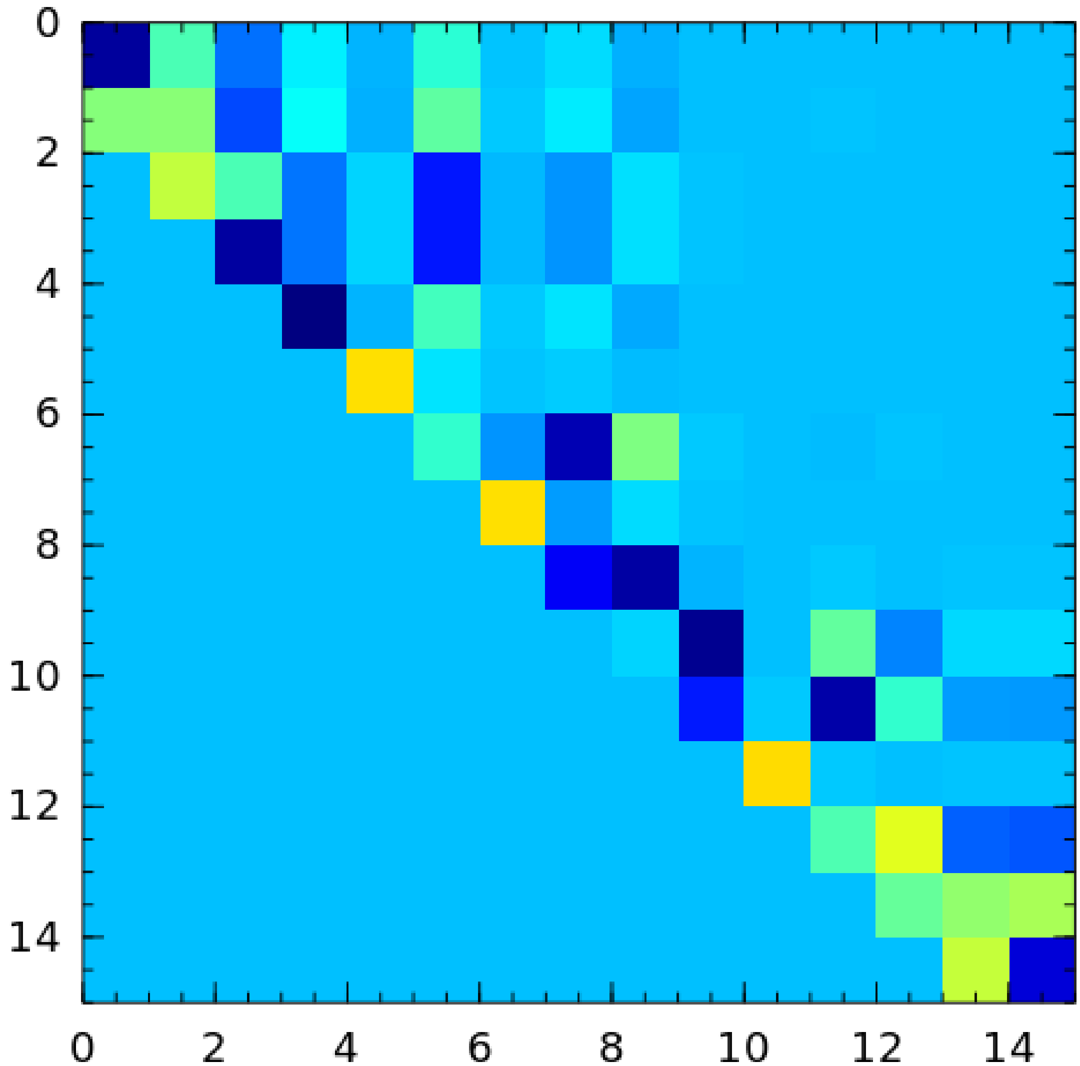
rando A random matrix with entries -1, 0 or 1.



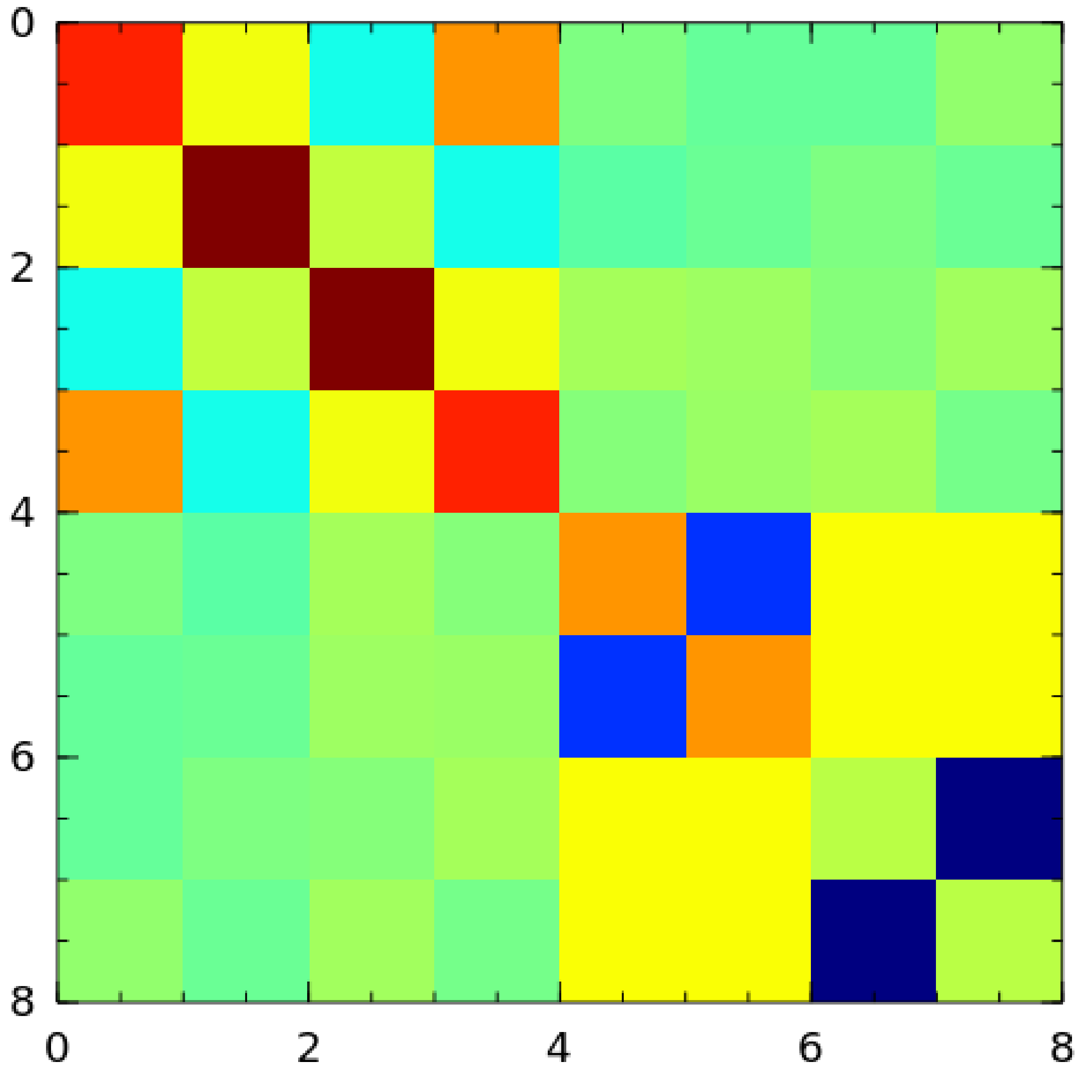
randsvd Random matrix with pre-assigned singular values. See *[high02]* (Sec. 28.3).



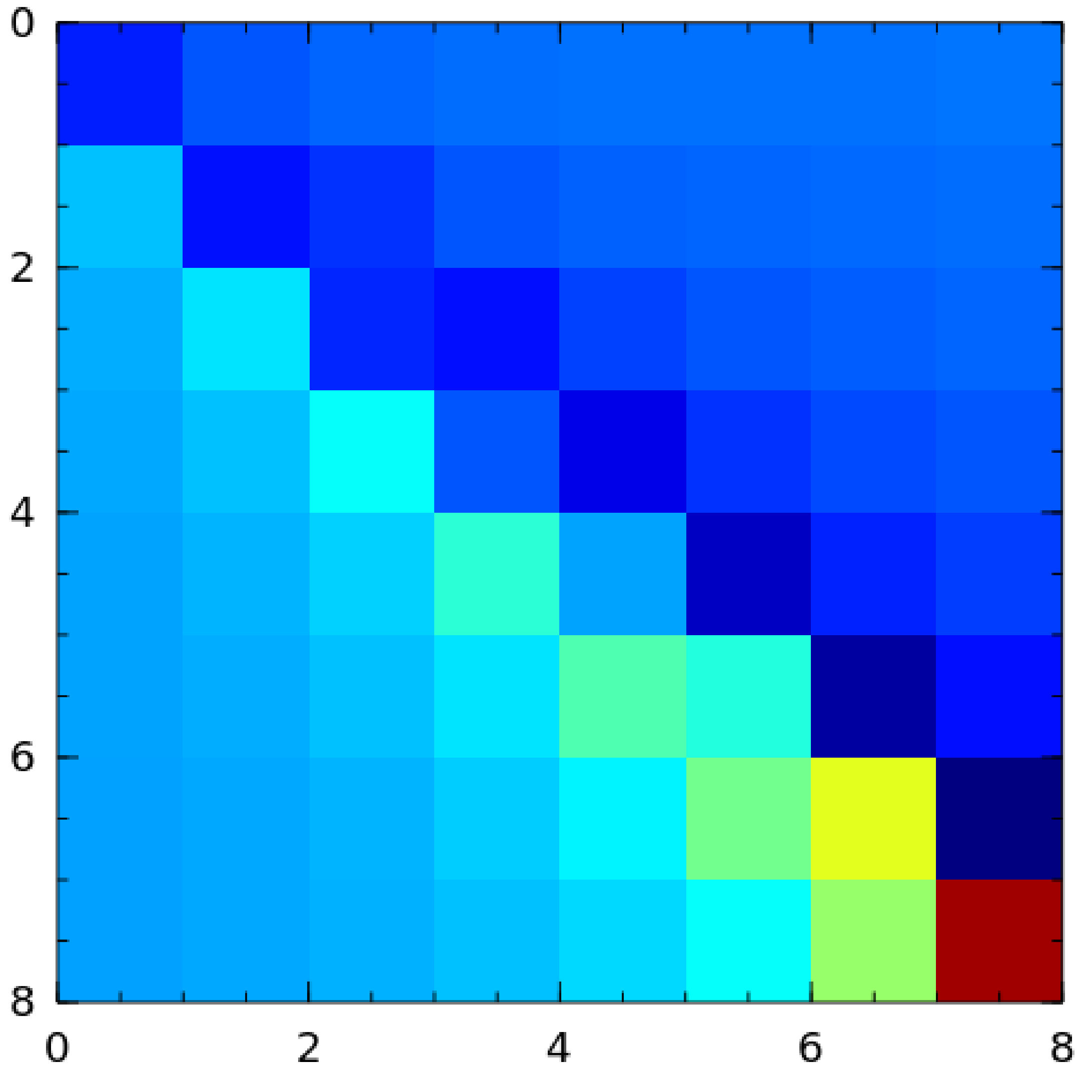
rohess A random orthogonal upper Hessenberg matrix. The matrix is constructed via a product of Givens rotations.



rosser The Rosser matrix's eigenvalues are very close together so it is a challenging matrix for many eigenvalue algorithms. `matrixdepot("rosser", 8, 2, 1)` generates the test matrix used in the paper *[rlhk51]*. `matrixdepot("rosser")` are more general test matrices with similar property.



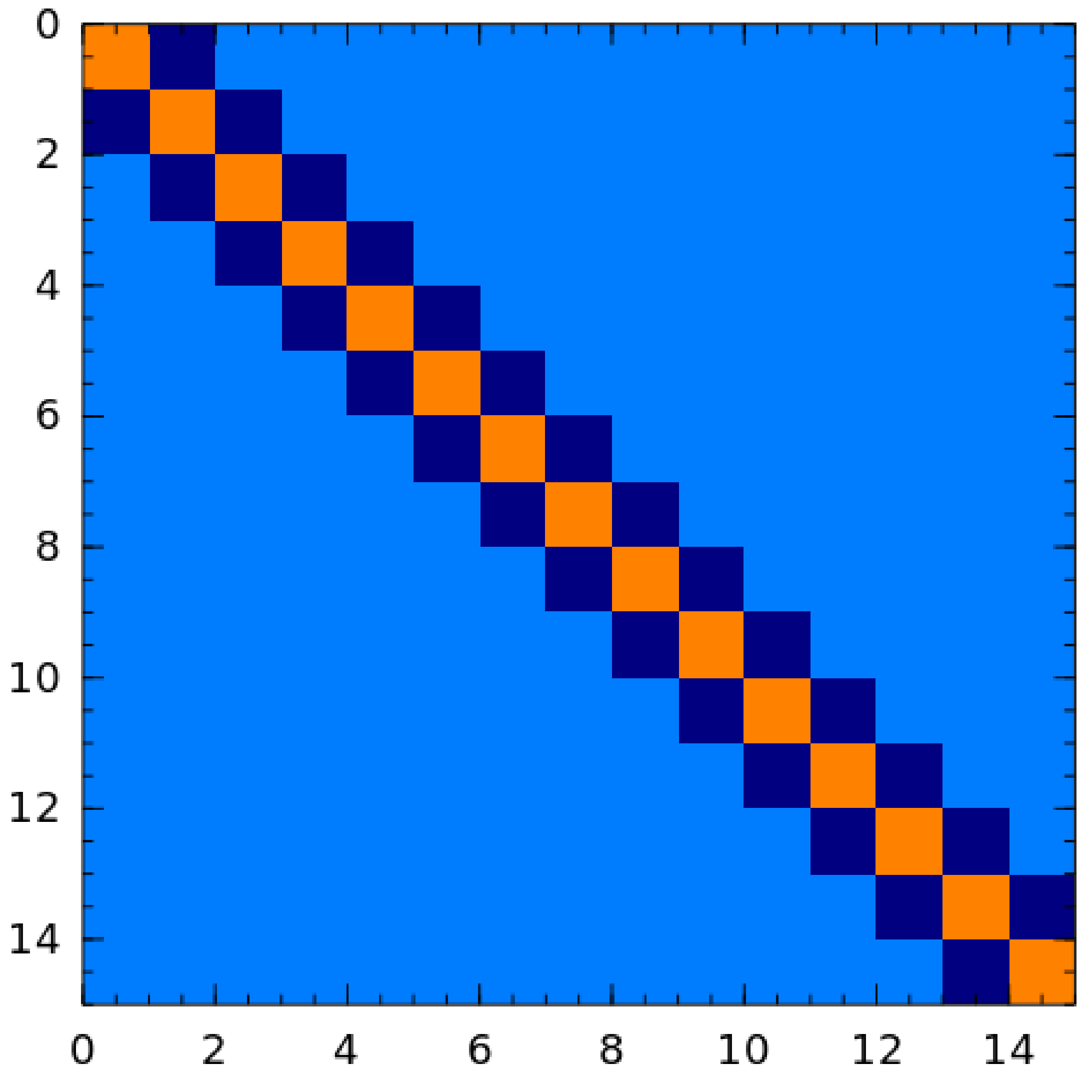
sampling Matrices with application in sampling theory. A n-by-n nonsymmetric matrix with eigenvalues $0, 1, 2, \ldots, n-1$ *[botr07]*.



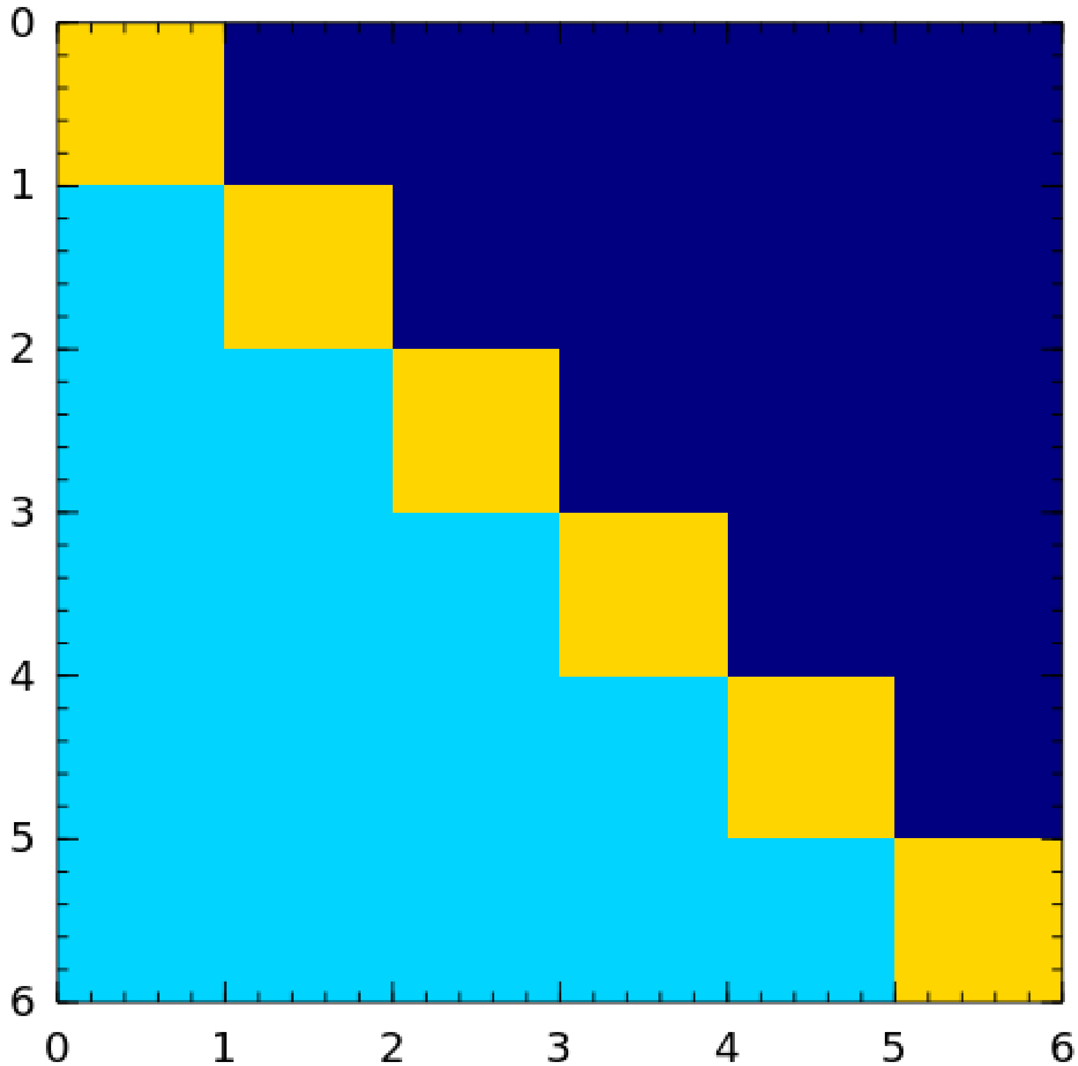
toeplitz Toeplitz matrix is a matrix in which each descending diagonal from left to right is constant. For example:

```
julia> matrixdepot("toeplitz", [1,2,3,4], [1,4,5,6])
4x4 Array{Int64,2}:
1  4  5  6
2  1  4  5
3  2  1  4
4  3  2  1

julia> matrixdepot("toeplitz", [1,2,3,4])
4x4 Array{Int64,2}:
1  2  3  4
2  1  2  3
3  2  1  2
4  3  2  1
```

tridiag A group of tridiagonal matrices. `matrixdepot("tridiagonal", n)` generate a tridiagonal matrix with 1 on the diagonal and -2 on the upper- lower- diagonal, which is a symmetric positive definite M-matrix. This matrix is also known as Strang's matrix, named after Gilbert Strang.



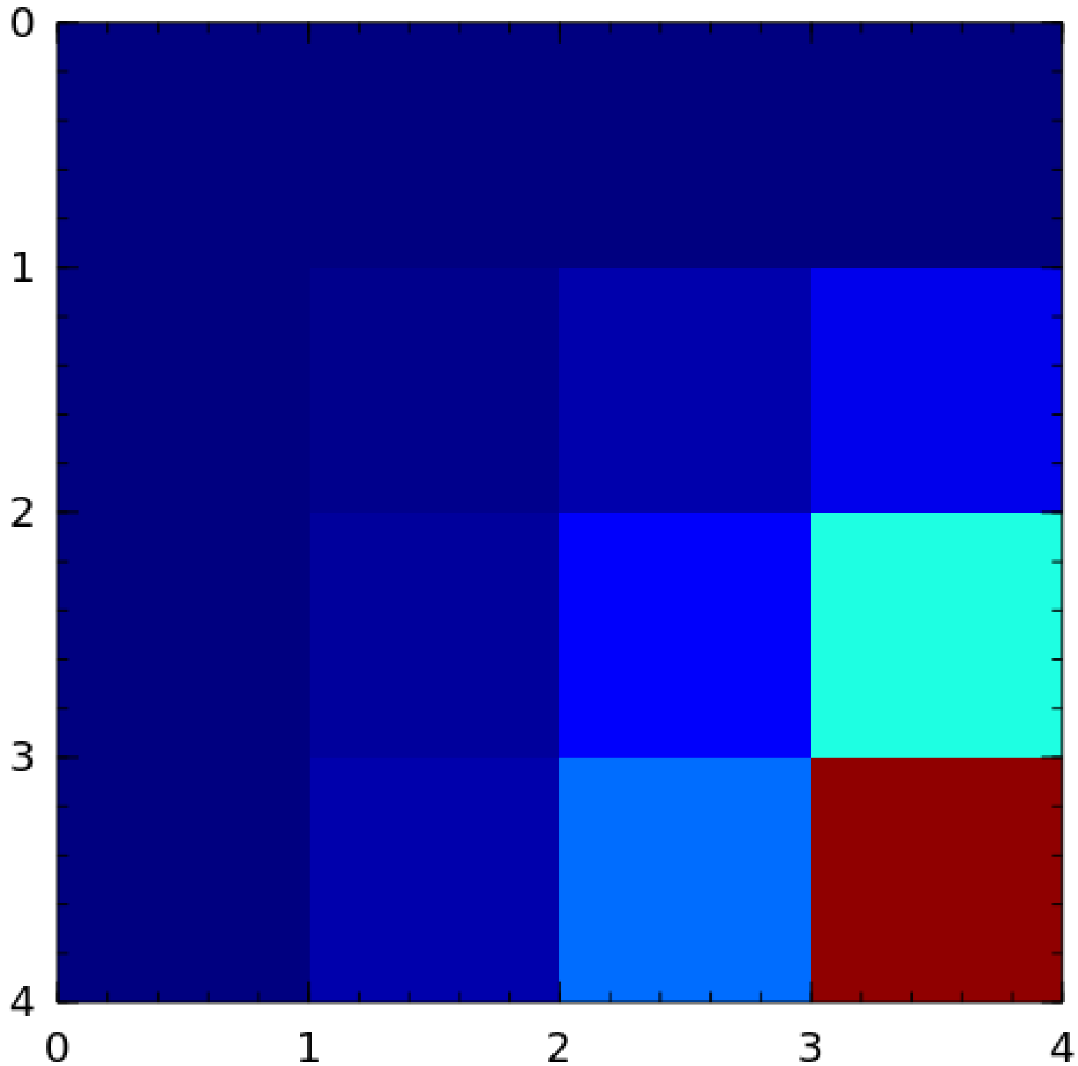
triw Upper triangular matrices discussed by Wilkinson and others *[gowi76]*.



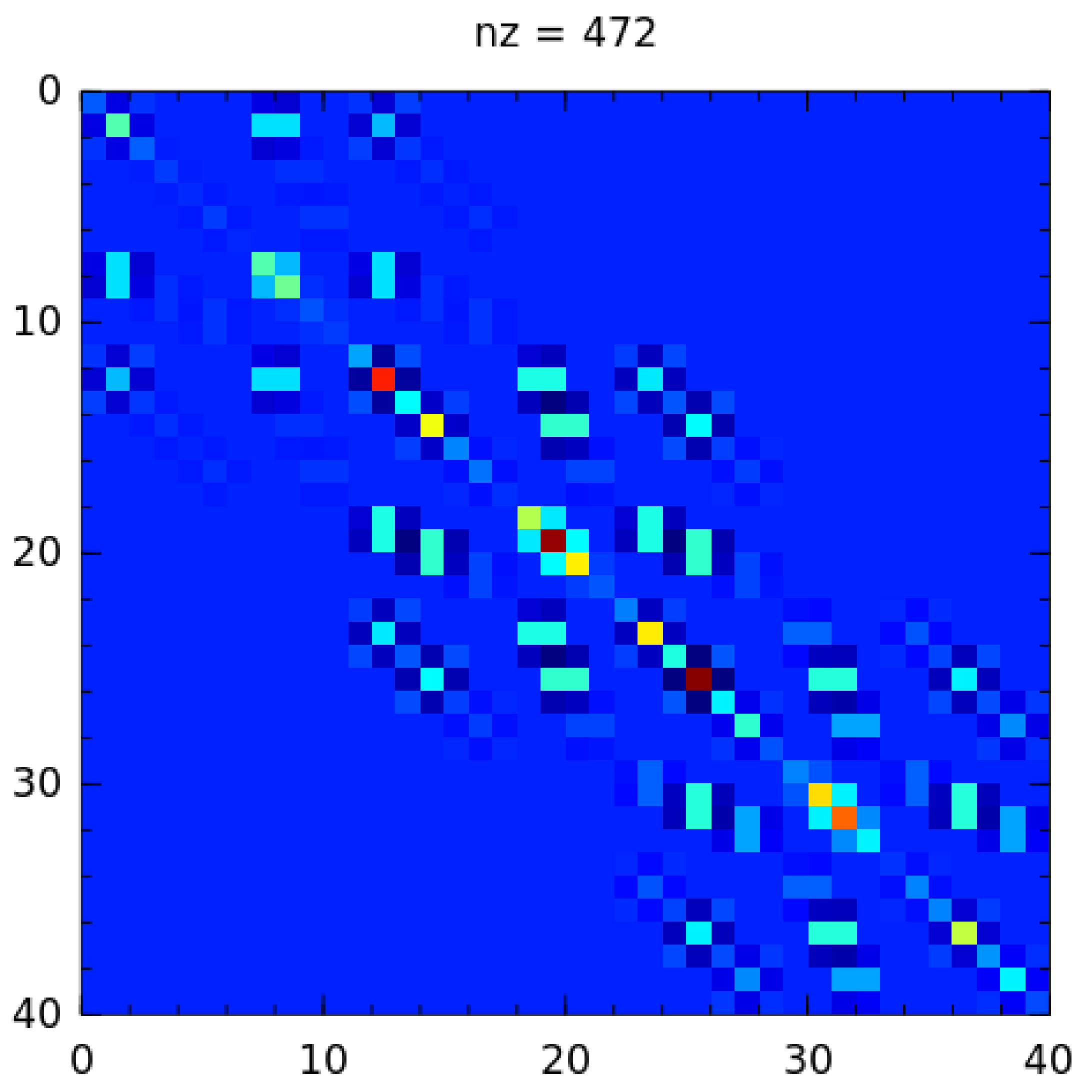
vand The Vandermonde matrix is defined in terms of scalars $\alpha_0, \alpha_1, \dots, \alpha_n$ by

$$V(\alpha_0, \dots, \alpha_n) = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ \alpha_0 & \alpha_1 & \cdots & \alpha_n \\ \vdots & \vdots & & \vdots \\ \alpha_0^n & \alpha_1^n & \cdots & \alpha_n^n \end{bmatrix}.$$

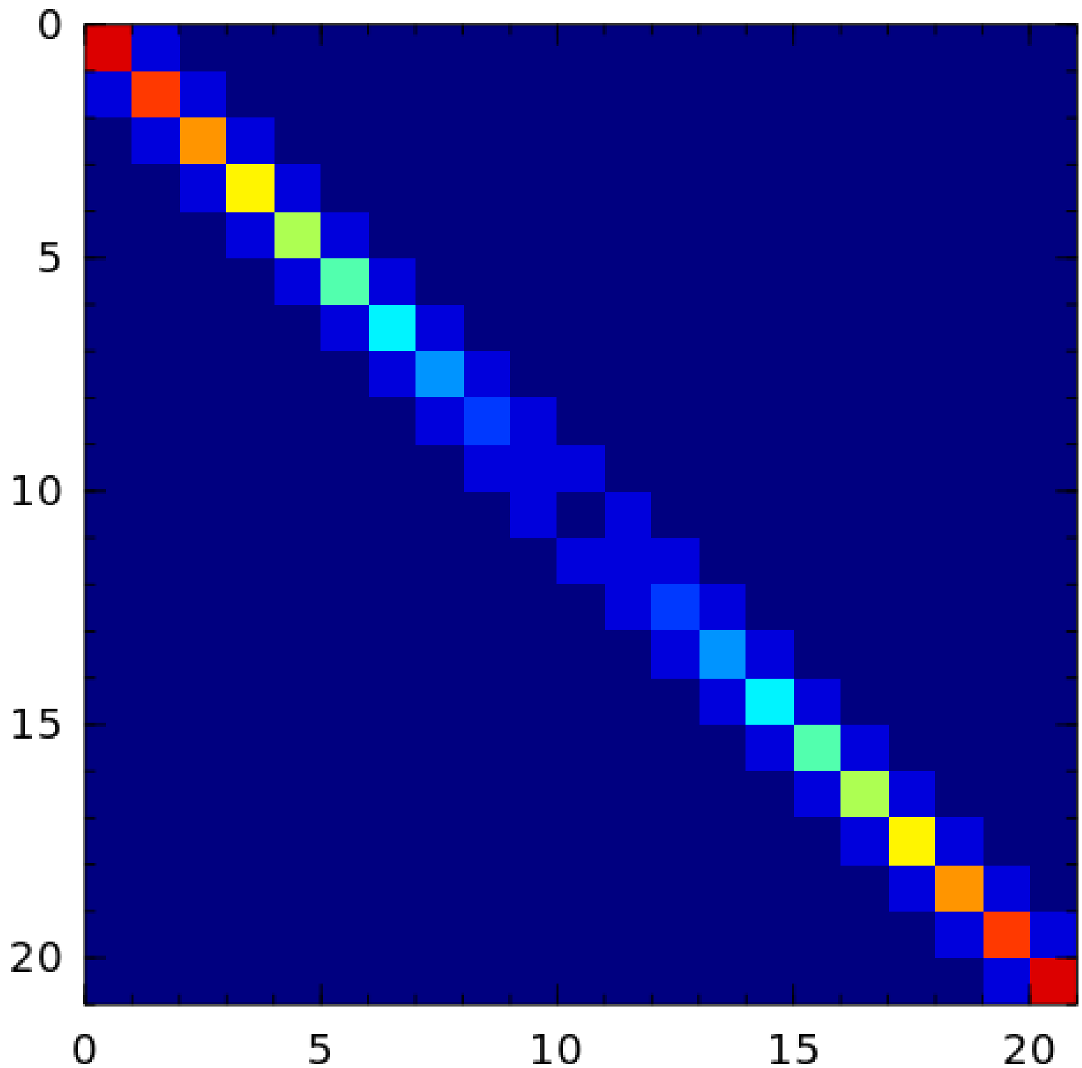
The inverse and determinant are known explicitly *[high02]*.



wathen The Wathen matrix is a sparse, symmetric positive, random matrix arising from the finite element method *[wath87]*. It is the consistent mass matrix for a regular *nx-by-ny* grid of 8-node elements.



wilkinson The Wilkinson matrix is a symmetric tridiagonal matrix with pairs of nearly equal eigenvalues. The most frequently used case is `matrixdepot("wilkinson", 21)`.



Note: The images are generated using Winston.jl 's `imagesc` function.

1.3 Random Graphs

- *erdrey*
- *gilbert*
- *smallworld*

erdrey An adjacency matrix of an Erdős–Rényi random graph: an undirected graph is chosen uniformly at random from the set of all symmetric graphs with a fixed number of nodes and edges. For example:

```
julia> using Random; Random.seed!(0);

julia> matrixdepot("erdrey", Int8, 5, 3)
5×5 SparseMatrixCSC{Int8,Int64} with 6 stored entries:
  [2, 1]  =  1
  [4, 1]  =  1
  [1, 2]  =  1
  [1, 4]  =  1
  [5, 4]  =  1
  [4, 5]  =  1
```

gilbert An adjacency matrix of a Gilbert random graph: each possible edge occurs independently with a given probability.

smallworld Motivated by the small world model proposed by Watts and Strogatz [wast98], we proposed a random graph model by adding shortcuts to a kth nearest neighbor ring (node i and j are connected iff $|i-j| \leq k$ or $|n-|i-j|| \leq k$).

```
julia> mdinfo("smallworld")
  Small World Network (smallworld)


  Generate an adjacency matrix for a small world network. We model it by adding␣
↪shortcuts to a
  kth nearest neighbour ring network (nodes i and j are connected iff |i -j| <= k or␣
↪|n - |i
  -j|| <= k.) with n nodes.

  Input options:

    •    [type,] n, k, p: the dimension of the matrix is n. The number of nearest-
↪neighbours
        to connect is k. The probability of adding a shortcut in a given row is p.

    •    [type,] n: k = 2 and p = 0.1.

  References:

 .. [wast98] D.J. Watts and S. H. Strogatz. Collective Dynamics of Small World

          Networks, Nature 393 (1998), pp. 440-442.
```

1.4 Test Problems for Regularization Methods

A Fredholm integral equation of the first kind (in 1-dimensional) can be written as

$$\int_0^1 K(s,t)f(t)dt = g(s), \quad 0 \leq s \leq 1,$$

where g and K (called kernel) are known functions and f is the unknown solution. This is a classical example of a linear ill-posed problem, i.e., an arbitrary small perturbation of the data can cause an arbitrarily large perturbation of the solution. For example, in computerized tomography, K is an X-ray source, f is the object being scanned, and g is the measured damping of the X-rays. The goal here is to reconstruct the scanned object from information about the locations of the X-ray sources and measurements of their damping.

After discretizations (by the quadrature method or the Galerkin method), we obtain a linear system of equations $Ax = b$. All the regularization test problems are derived from discretizations of a Fredholm integral equation of the first kind. Each generated test problem has type `RegProb`, which is defined as:

```
immutable RegProb{T}
  A::AbstractMatrix{T}  # matrix of interest
  b::AbstractVector{T}  # right-hand side
  x::AbstractVector{T}  # the solution to Ax = b
end
```

Here is an example:

```
julia> mdinfo("deriv2")
Computation of the Second Derivative:

A classical test problem for regularization algorithms.

Input options:

1. [type,] n, [matrixonly]: the dimension of the matrix is n.
           If matrixonly = false, the linear system A, b, x will be generated.
           (matrixonly = true by default.)

Reference: P.C. Hansen, Regularization tools: A MATLAB package for
           analysis and solution of discrete ill-posed problems.
           Numerical Algorithms, 6(1994), pp.1-35

julia> A = matrixdepot("deriv2", 4) # generate the test matrix
4x4 Array{Float64,2}:
-0.0169271   -0.0195313  -0.0117188  -0.00390625
-0.0195313   -0.0481771  -0.0351563  -0.0117188
-0.0117188   -0.0351563  -0.0481771  -0.0195313
-0.00390625  -0.0117188  -0.0195313  -0.0169271

      julia> r = mdopen("deriv2", 3, false) # generate all data
      MatrixDepot.GeneratedMatrixData{:B}("deriv2", 10, MatrixDepot.deriv2)(3, false)

julia> metasymbols(r) # which darta are available?
(:A, :b, :x)

julia> r.A # matrix A
3x3 Array{Float64,2}:
 -0.0277778   -0.0277778  -0.00925926
 -0.0277778   -0.0648148  -0.0277778
 -0.00925926  -0.0277778  -0.0277778

julia> r.b # right hand side
3-element Array{Float64,1}:
 -0.01514653483985129
 -0.03474793286789414
 -0.022274315940957783

julia> r.x # solution
3-element Array{Float64,1}:
 0.09622504486493762
 0.28867513459481287
 0.48112522432468807
```

Here is a list of test problems in the collection:

- *baart*
- *blur*
- *deriv2*
- *foxgood*
- *gravity*
- *heat*
- *parallax*
- *phillips*
- *shaw*
- *spikes*
- *ursell*
- *wing*

baart Discretization of an artificial Fredholm integral equation of the first kind *[baart82]*. The kernel K is given by

$$K(s,t) = \exp(s\cos(t)).$$

The right-hand side g and the solution f are given by

$$g(s) = 2\frac{\sin(s)}{s}, \quad f(t) = \sin(t).$$

blur Image deblurring test problem. It arises in connection with the degradation of digital images by atmospheric turbulence blur, modelled by a Gaussian point-spread function

$$h(x,y) = \frac{1}{2\pi\sigma^2}\exp(-\frac{x^2+y^2}{2\sigma^2}).$$

The matrix A is a symmetric $n^2 \times n^2$ doubly block Toeplitz matrix, stored in sparse format.

deriv2 Computation of the second derivative. The kernel K is Green's function for the second derivative

$$K(s,t) = \begin{cases} s(t-1), & s < t, \\ t(s-1), & s \geq t, \end{cases}$$

and both integration intervals are $[0,1]$. The function g and f are given by

$$g(s) = (s^3 - s)/6, \quad f(t) = t.$$

The symmetric matrix A and vectors x and b are computed from K, f and g using the Galerkin method.

foxgood A severely ill-posed problem suggested by Fox & Goodwin. This is a model problem which does not satisfy the discrete Picard condition for the small singular values *[baker77]*.

gravity One-dimensional gravity surveying model problem. Discretization of a 1-D model problem in gravity surveying, in which a mass distribution f(t) is located at depth d, while the vertical component of the gravity field g(s) is measured at the surface. The resulting problem is a first-kind Fredholm integral equation with kernel

$$K(s,t) = d(d^2 + (s-t)^2)^{-3/2}.$$

heat Inverse heat equation *[carasso82]*. It is a Volterra integral equation of the first kind with integration interval $[0, 1]$. The kernel K is given by

$$K(s,t) = k(s-t),$$

where

$$k(t) = \frac{t^{-3/2}}{2\kappa\sqrt{\pi}} \exp\big(-\frac{1}{4\kappa^2 t}\big).$$

κ controls the ill-conditioning of the matrix A. $\kappa = 1$ (default) gives an ill-conditioned matrix and $\kappa = 5$ gives a well-conditioned matrix.

parallax Stellar parallax problem with 26 fixed, real observations. The underlying problem is a Fredholm integral equation of the first kind with kernel

$$K(s,t) = \frac{1}{\sigma\sqrt{2\pi}} \exp\Big(-\frac{1}{2}\big(\frac{s-t}{\sigma}\big)^2\Big),$$

with $\sigma = 0.014234$ and it is discretized by means of a Galerkin method with n orthonormal basis functions. The right-hand side b consists of a measured distribution function of stellar parallaxes, and its length is fixed at 26; i.e., the matrix A is $26 \times n$. The exact solution, which represents the true distribution of stellar parallaxes, is unknown.

phillips Phillips's "famous" problem. Discretization of the "famous" Fredholm integral equation of the first kind devised by D.L. Phillips *[phillips62]*. The kernel K and solution f are given by

$$K(s,t) = \theta(s-t), \quad f(t) = \theta(t),$$

where

$$\theta(x) = \begin{cases} 1 + \cos(\frac{\pi x}{3}), & |x| < 3, \\ 0, & |x| \geq 3. \end{cases}$$

The right-hand side g is given by

$$g(s) = (6 - |s|)\Big(1 + \frac{1}{2}\cos\big(\frac{\pi s}{3}\big)\Big) + \frac{9}{2\pi}\sin\Big(\frac{\pi|s|}{3}\Big).$$

Both integration intervals are $[-6, 6]$.

shaw One-dimensional image restoration model. This test problem uses a first-kind Fredholm integral equation to model a one-dimensional image restoration situation. The kernel K is given by

$$K(s,t) = (\cos(s) + \cos(t))^2 \big(\frac{\sin(u)}{u}\big)^2,$$

where

$$u = \pi(\sin(s) + \sin(t)).$$

Both integration intervals are $[-\pi/2, \pi/2]$. The solution f is given by

$$f(t) = a_1 \exp(-c_1(t-t_1)^2) + a_2 \exp(-c_2(t-t_2)^2).$$

K and f are discretized by simple quadrature to produce the matrix A and the solution vector x. The right-hand b is computed by $b = Ax$.

spikes Artificially generated discrete ill-posed problem.

ursell Discretization of a Fredholm integral equation of the first kind with kernel K and right-hand side g given by

$$K(s,t) = \frac{1}{s+t+1}, \quad g(s) = 1,$$

where both integration intervals are $[0, 1]$ *[ursell]*.

wing A problem with a discontinuous solution. The kernel K is given by

$$K(s,t) = t\exp(-st^2),$$

with both integration intervals are $[0, 1]$. The functions f and g are given as

$$f(t) = \begin{cases} 1, & t_1 < t < t_2, \\ 0, & \text{otherwise}, \end{cases} \qquad g(s) = \frac{\exp(-st_1^2) - \exp(-st_2^2)}{2s}.$$

Here $0 < t_1 < t_2 < 1$. The matrix A and two vectors x and b are obtained by Galerkin discretization with orthonormal basis functions defined on a uniform mesh.

1.5 Groups

Groups are lists of matrix names and we use them to categorize matrices in Matrix Depot. The list below shows all the predefined groups in Matrix Depot and we can extend this list by defining new groups. Group names are noted as symbols, e.g. *:symmetric*.

1.5.1 Predefined Groups

all All the matrices in the collection.

data The matrix has been downloaded from UF sparse matrix collection or the Matrix Market collection.

eigen Part of the eigensystem of the matrix is explicitly known.

graph An adjacency matrix of a graph.

illcond The matrix is ill-conditioned for some parameter values.

inverse The inverse of the matrix is known explicitly.

posdef The matrix is positive definite for some parameter values.

random The matrix has random entries.

regprob The output is a test problem for Regularization Methods.

sparse The matrix is sparse.

symmetric The matrix is symmetric for some parameter values.

1.5.2 Adding New Groups

New groups can be added with the macro `@addgroup`:

```
@addgroup myfav = ["lehmer", "cauchy", "hilb"]

@addgroup test_for_paper2 = ["tridiag", "sampling", "wing"]

listgroups()

Groups:
 data           eigen          illcond        inverse
 posdef         random         regprob        sparse
 symmetric      myfav          test_for_paper2


listnames(:myfav)
3-element Array{ASCIIString,1}:
 "lehmer"
 "cauchy"
 "hilb"
```

1.6 Interface to Test Collections

The internal database is loaded automatically when using the module:

```
julia> using MatrixDepot
include group.jl for user defined matrix generators
verify download of index files...
used remote site is https://sparse.tamu.edu/?per_page=All
populating internal database...
```

1.6.1 Interface to the SuiteSparse Matrix Collection (formerly UFL collection)

Use `M = matrixdepot(NAME)` or `md = mdopen(NAME); M = md.A`, where `NAME` is `collection_name + '/' + matrix_name`, to download a test matrix from the 'SuiteSparse Matrix Collection. https://sparse.tamu.edu/ For example:

```
julia> md = mdopen("SNAP/web-Google")
PG SNAP/web-Google(#2301)  916428x916428(5105039) 2002 [A] 'Directed Graph' [Web␣
↪graph from Google]()
```

Note: `listnames("*/*")` displays all the matrix names in the collection, including the newly downloaded matrices. All the matrix data can be found by `listnames("**")`.

If the matrix name is unique in the collections, we could also use `matrixdepot(matrix_name)` to download the data. If more than one matrix has the same name, an error is thrown.

When download is complete, we can check matrix information using:

```
julia> mdinfo("SNAP/web-Google")
SNAP/web-Google


MatrixMarket matrix coordinate pattern general
```

(continues on next page)

(continued from previous page)

```
______________________________________________________________________

•    UF Sparse Matrix Collection, Tim Davis

•    http://www.cise.ufl.edu/research/sparse/matrices/SNAP/web-Google

•    name: SNAP/web-Google

•    [Web graph from Google]

•    id: 2301

•    date: 2002

•    author: Google

•    ed: J. Leskovec

•    fields: name title A id date author ed kind notes

•    kind: directed graph

______________________________________________________________________
...
```

and generate it by accessing the field *A*.

```
julia> M = md.A
916428×916428 SparseMatrixCSC{Bool,Int64} with 5105039 stored entries:
  [11343 ,       1]  =  true
  [11928 ,       1]  =  true
  [15902 ,       1]  =  true
  [29547 ,       1]  =  true
  [30282 ,       1]  =  true

  [788476, 916427]  =  true
  [822938, 916427]  =  true
  [833616, 916427]  =  true
  [417498, 916428]  =  true
  [843845, 916428]  =  true
```

You can convert the boolean pattern matrix to integer by `M * 1`.

The metadata of a given matrix can be obtained by accessing properties of *md*.

Which properties are available is shown in the *md::MatrixDescriptor*:

```
julia> md = mdopen("TKK/t520")
(IS TKK/t520(#1908)  5563x5563(286341/145952) 2008 [A, b, coord] 'Structural Problem'␣
↪[T-beam, L = 520 mm, Quadratic four node DK type elements.  R Kouhia]()
```

and also by the special function `metasymbols`:

```
julia> metasymbols(md)
(:A, :b, :coord)
```

When you access a single matrix with `matrixdepot(pattern)` or `mdopen(pattern)` the full matrix data are dowloaded implicitly in the background, if not yet available on the local disk cache.

When you access matrix information with `mdinfo(pattern)` for one or more matrices, the header data of the matrix are downloaded implicitly, if not yet available on the local disk cache.

It is also possible to dowload a bulk of matrix data by `MatrixDepot.loadinfo(pattern)` and `MatrixDepot.load(pattern)` to populate the disk cache in advance of usage.

1.6.2 Interface to NIST Matrix Market

Use `M = matrixdepot(NAME)` or `md = mdopen(NAME); M = md.A`, where `NAME` is `collection name + '/' + set name + '/' + matrix name` to download a test matrix from NIST Matrix Market: http://math.nist.gov/MatrixMarket/. For example:

```
julia> md = mdopen("Harwell-Boeing/lanpro/nos5")
The collection-name and set-name may as always be replaced by wildcard patterns "*",
as long as there exists only on name matching the pattern.

julia> md = mdopen("*/*/bp__1400")
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                               Dload  Upload   Total   Spent    Left  Speed
100 28192  100 28192    0     0   4665      0  0:00:06  0:00:06 --:--:-- 10004
download:/home/.../MatrixDepot/data/mm/Harwell-Boeing/smtape/bp__1400.mtx.gz

(RG Harwell-Boeing/smtape/bp__1400(#M93)  822x822(4790)   [A] '' [] ()
```

Checking matrix information and generating matrix data are similar to the above case:

```
julia> mdinfo(md) # or mdinfo("*/*/bp__1400")
  Harwell-Boeing/smtape/bp__1400


  MatrixMarket matrix coordinate real general

  822 822 4790
```

There is no header information in this collection besides m, n, and dnz.

```
julia> md.A # or matrixdepot("Harwell-Boeing/smtape/bp__1400")
822x822 sparse matrix with 4790 Float64 entries:
      [1  ,    1]  =  1.0
      [1  ,    2]  =  0.001
      [26 ,    2]  =  -1.0
      [1  ,    3]  =  0.6885
      [25 ,    3]  =  0.9542
      [692,    3]  =  1.0
      [718,    3]  =  5.58

      [202, 820]  =  -1.0
      [776, 820]  =  1.0
      [1  , 821]  =  0.4622
      [25 , 821]  =  0.725
      [28 , 821]  =  1.0
      [202, 821]  =  -1.0
      [796, 821]  =  1.0
      [2  , 822]  =  1.0
```

1.7 Adding New Matrix Generators

Matrix Depot provides a diverse collection of test matrices, including parametrized matrices and real-life matrices. But occasionally, you may want to define your own matrix generators and be able to use them from Matrix Depot.

1.7.1 Declaring Generators

When Matrix Depot is first loaded, a new directory `myMatrixDepot` will be created. Matrix Depot automatically includes all Julia files in this directory. Hence, all we need to do is to copy the generator files to `path/to/MatrixDepot/myMatrixDepot` and use the function `include_generator` to declare them.

include_generator (*Stuff_To_Be_Included*, *Stuff*, *f*)

Includes a piece of information of the function `f` to Matrix Depot, where `Stuff_To_Be_Included` is one of the following:

- `FunctionName`: the function name of `f`. In this case, `Stuff` is a string representing `f`.
- `Group`: the group where `f` belongs. In this case, `Stuff` is the group name.

1.7.2 Examples

To get a feel of how it works, let's see an example. Suppose we have a file `myrand.jl` which contains two matrix generator `randsym` and `randorth`:

```
"""
random symmetric matrix
=======================

*Input options:*

+ n: the dimension of the matrix
"""
function randsym(n)
    A = zeros(n, n)
    for j = 1:n
      for i = j:n
        A[i,j] = randn()
        if i != j; A[j,i] = A[i,j] end
      end
    end
    return A
end

"""
random Orthogonal matrix
========================

*Input options:*

+ n: the dimension of the matrix
"""
randorth(n) = qr(randn(n,n)).Q
```

We first need to find out where Matrix Depot is installed. This can be done by:

```
julia> @which matrixdepot("")
matrixdepot(p::Union{Regex,...}, args...) in MatrixDepot at
/home/.../.julia/dev/MatrixDepot/src/common.jl:508
```

For me, the package user data are installed at `/home/.../.julia/dev/MatrixDepot/myMatrixDepot`. We can copy `myrand.jl` to this directory. Now we open the file `myMatrixDepot/generator.jl` and write:

```
include_generator(FunctionName, "randsym", randsym)
include_generator(FunctionName, "randorth", randorth)
```

Due to a bug we have to remove file `db.data` and restart julia: `rm MatrixDepot/data/db.data`

This is it. We can now use them from Matrix Depot:

```
   julia> using MatrixDepot
   include group.jl for user defined matrix generators
   include myrand.jl for user defined matrix generators
   verify download of index files...
   used remote site is https://sparse.tamu.edu/?per_page=All
   populating internal database...

   julia> mdinfo()
     Currently loaded Matrices
     ---------------------------

   builtin(#)
   ----------- ----------- ----------- ------------ ----------- -------------␣
↪------------
   1 baart     10 deriv2   19 gravity  28 kms       37 parter   46 rohess     55␣
↪ursell
   2 binomial  11 dingdong 20 grcar    29 lehmer    38 pascal   47 rosser     56 vand
   3 blur      12 erdrey   21 hadamard 30 lotkin    39 pei      48 sampling   57␣
↪wathen
   4 cauchy    13 fiedler  22 hankel   31 magic     40 phillips 49 shaw       58␣
↪wilkinson
   5 chebspec  14 forsythe 23 heat     32 minij     41 poisson  50 smallworld 59 wing
   6 chow      15 foxgood  24 hilb     33 moler     42 prolate  51 spikes
   7 circul    16 frank    25 invhilb  34 neumann   43 randcorr 52 toeplitz
   8 clement   17 gilbert  26 invol    35 oscillate 44 rando    53 tridiag
   9 companion 18 golub    27 kahan    36 parallax  45 randsvd  54 triw

   user(#)
   ---------- ---------
   1 randorth 2 randsym

   Groups
   ------- ----- ----- ------- ------ ------- ---------------
   all     local eigen illcond posdef regprob symmetric
   builtin user  graph inverse random sparse  test_for_paper2

   Suite Sparse of
   ------------ ----
   2773         2833

   MatrixMarket of
   ------------ ---
   488          498
```

(continues on next page)

(continued from previous page)

```
 julia> mdinfo("randsym")
    random symmetric matrix


   Input options:

   •  n: the dimension of the matrix

 julia> matrixdepot("randsym", 5)
 5x5 Array{Float64,2}:
  1.57579    0.474591  0.0261732  -0.536217  -0.0900839
  0.474591   0.388406  0.77178     0.239696   0.302637
  0.0261732  0.77178   1.7336      1.72549    0.127008
 -0.536217   0.239696  1.72549     0.304016   1.5854
 -0.0900839  0.302637  0.127008    1.5854    -0.656608

 julia> A = matrixdepot("randorth", 5)
 5x5 Array{Float64,2}:
-0.359134   0.401435   0.491005  -0.310518   0.610218
-0.524132  -0.474053  -0.53949   -0.390514   0.238764
 0.627656   0.223519  -0.483424  -0.104706   0.558054
-0.171077   0.686038  -0.356957  -0.394757  -0.465654
 0.416039  -0.305802   0.326723  -0.764383  -0.205834

 julia> A'*A
 5x5 Array{Float64,2}:
 1.0           8.32667e-17   1.11022e-16   5.55112e-17  -6.93889e-17
 8.32667e-17   1.0          -1.80411e-16  -2.77556e-17  -5.55112e-17
 1.11022e-16  -1.80411e-16   1.0           1.94289e-16  -1.66533e-16
 5.55112e-17  -2.77556e-17   1.94289e-16   1.0           1.38778e-16
-6.93889e-17  -5.55112e-17  -1.66533e-16   1.38778e-16   1.0
```

We can also add group information in generator.jl:

include_generator(Group, :random, randsym) include_generator(Group, :symmetric, randsym)

After re-starting julia, if we type:

```
  julia> using MatrixDepot
  include group.jl for user defined matrix generators
  include myrand.jl for user defined matrix generators
  verify download of index files...
  used remote site is https://sparse.tamu.edu/?per_page=All
  populating internal database...

  julia> listnames(:symmetric)
  list(22)
  -------- -------- ------- ------- ------ ----- --------- ------- -------- -------␣
↪---------
  cauchy   clement  fiedler hilb    kms    minij oscillate pei     prolate  randsym␣
↪wathen
  circul   dingdong hankel  invhilb lehmer moler pascal    poisson randcorr tridiag␣
↪wilkinson

julia> mdlist(:random)
9-element Array{ASCIIString,1}:
```

(continues on next page)

(continued from previous page)

```
"golub"
"oscillate"
"randcorr"
"rando"
"randsvd"
"randsym"
"rohess"
"rosser"
"wathen"
```

the function `randsym` will be part of the groups `:symmetric` and `:random`.

It is a good idea to back up your changes. For example, we could save it on GitHub by creating a new repository named `myMatrixDepot`. (See https://help.github.com/articles/create-a-repo/ for details of creating a new repository on GitHub.) Then we go to the directory `path/to/MatrixDepot/myMatrixDepot` and type:

```
git init
git add *.jl
git commit -m "first commit"
git remote add origin https://github.com/your-user-name/myMatrixDepot.git
git push -u origin master
```

1.8 Examples

1.8.1 Demo

IJulia Notebook

1.8.2 Getting Started

To see all the matrices in the collection, type

```
julia> mdinfo()
  Currently loaded Matrices
  ---------------------------

builtin(#)
----------- ----------- ----------- ------------ ----------- -------------␣
↪------------
1 baart     10 deriv2   19 gravity  28 kms       37 parter   46 rohess     55 ursell
2 binomial  11 dingdong 20 grcar    29 lehmer    38 pascal   47 rosser     56 vand
3 blur      12 erdrey   21 hadamard 30 lotkin    39 pei      48 sampling   57 wathen
4 cauchy    13 fiedler  22 hankel   31 magic     40 phillips 49 shaw       58␣
↪wilkinson
5 chebspec  14 forsythe 23 heat     32 minij     41 poisson  50 smallworld 59 wing
6 chow      15 foxgood  24 hilb     33 moler     42 prolate  51 spikes
7 circul    16 frank    25 invhilb  34 neumann   43 randcorr 52 toeplitz
8 clement   17 gilbert  26 invol    35 oscillate 44 rando    53 tridiag
9 companion 18 golub    27 kahan    36 parallax  45 randsvd  54 triw

user(#)
-------
```

(continues on next page)

(continued from previous page)

```
Groups
-------- ----- ----- -------- ------ -------- ---------
all      local eigen illcond posdef regprob symmetric
builtin user  graph inverse random sparse

Suite Sparse of
------------ ----
2772         2833

MatrixMarket of
------------ ---
488          498
```

We can generate a Hilbert matrix of size 4 by typing

```
matrixdepot("hilb", 4)

4x4 Array{Float64,2}:
 1.0       0.5       0.333333  0.25
 0.5       0.333333  0.25      0.2
 0.333333  0.25      0.2       0.166667
 0.25      0.2       0.166667  0.142857
```

and generate a circul matrix of size 5 by

```
matrixdepot("circul", 5)

5x5 Array{Float64,2}:
 1.0  2.0  3.0  4.0  5.0
 5.0  1.0  2.0  3.0  4.0
 4.0  5.0  1.0  2.0  3.0
 3.0  4.0  5.0  1.0  2.0
 2.0  3.0  4.0  5.0  1.0
```

We can type the matrix name to get help.

```
mdinfo("hilb")
 Hilbert matrix


The Hilbert matrix is a very ill conditioned matrix. It is symmetric
positive definite and totally positive.

Input options:

  •  [type,] dim: the dimension of the matrix;

  •  [type,] row_dim, col_dim: the row and column dimensions.

Groups: ["inverse", "ill-cond", "symmetric", "pos-def"]

Reference: M. D. Choi, Tricks or treats with the Hilbert matrix, Amer. Math.
Monthly, 90 (1983), pp. 301-312.

N. J. Higham, Accuracy and Stability of Numerical Algorithms, Society for
Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002; sec. 28.1
```

```
mdinfo("hadamard")
  Hadamard matrix


The Hadamard matrix is a square matrix whose entries are 1 or -1. It was
named after Jacques Hadamard. The rows of a Hadamard matrix are orthogonal.

Input options:

  •  [type,] n: the dimension of the matrix, n is a power of 2.

Groups: ["inverse", "orthogonal", "eigen"]

Reference: S. W. Golomb and L. D. Baumert, The search for Hadamard matrices,
Amer. Math. Monthly, 70 (1963) pp. 12-17
```

From the information given, we can create a 4-by-6 rectangular Hilbert matrix by

```
matrixdepot("hilb", 4, 6)

4x6 Array{Float64,2}:
 1.0       0.5       0.333333  0.25      0.2       0.166667
 0.5       0.333333  0.25      0.2       0.166667  0.142857
 0.333333  0.25      0.2       0.166667  0.142857  0.125
 0.25      0.2       0.166667  0.142857  0.125     0.111111
```

We can also specify the data type

```
matrixdepot("hilb", Float16, 5, 3)

5x3 Array{Float16,2}:
 1.0      0.5      0.33325
 0.5      0.33325  0.25
 0.33325  0.25     0.19995
 0.25     0.19995  0.16663
 0.19995  0.16663  0.14282
```

Matrices can be accessed by groups.

```
 mdlist(:symmetric)

19-element Array{ASCIIString,1}:
 "hilb"
 "cauchy"
 "circul"
 "dingdong"
 "invhilb"
 "moler"
 "pascal"
 "pei"
 "clement"
 "fiedler"
 "minij"
 "tridiag"
 "lehmer"
 "randcorr"
 "poisson"
```

(continues on next page)

(continued from previous page)

```
 "wilkinson"
 "randsvd"
 "kms"
 "wathen"
```

```
mdlist(:symmetric & :illcond)

7-element Array{ASCIIString,1}:
 "hilb"
 "cauchy"
 "invhilb"
 "moler"
 "pascal"
 "pei"
 "tridiag"
```

```
mdlist(:inverse & :illcond & :symmetric)

7-element Array{ASCIIString,1}:
 "hilb"
 "cauchy"
 "invhilb"
 "moler"
 "pascal"
 "pei"
 "tridiag"
```

1.8.3 User Defined Groups

We can add new groups to MatrixDepot. Since each group in Matrix Depot is a list of strings, you can simply do, for example,

```
spd = mdlist(:symmetric & :posdef)


10-element Array{ASCIIString,1}:
 "hilb"
 "cauchy"
 "circul"
 "invhilb"
 "moler"
 "pascal"
 "pei"
 "minij"
 "tridiag"
 "lehmer"
```

```
myprop = ["lehmer", "cauchy", "hilb"]

3-element Array{ASCIIString,1}:
 "lehmer"
 "cauchy"
 "hilb"
```

Then use it in your tests like

```
for matrix in myprop
    A = matrixdepot(matrix, 6)
    L, U, p = lu(A) #LU factorization
    err = norm(A[p,:] - L*U, 1) # 1-norm error
    println("1-norm error for $matrix matrix is ", err)
end

1-norm error for lehmer matrix is 1.1102230246251565e-16
1-norm error for cauchy matrix is 5.551115123125783e-17
1-norm error for hilb matrix is 2.7755575615628914e-17
```

To add a group of matrices permanently for future use, we put the macro `@addgroup` at the beginning.

```
@addgroup myfav = ["lehmer", "cauchy", "hilb"]

@addgroup test_for_paper2 = ["tridiag", "sampling", "wing"]
```

You can see the changes immediately:

```
mdinfo()
  Currently loaded Matrices
  ---------------------------

builtin(#)
----------- ----------- ----------- ------------ ----------- -------------␣
↪------------
1 baart     10 deriv2   19 gravity  28 kms       37 parter   46 rohess     55 ursell
2 binomial  11 dingdong 20 grcar    29 lehmer    38 pascal   47 rosser     56 vand
3 blur      12 erdrey   21 hadamard 30 lotkin    39 pei      48 sampling   57 wathen
4 cauchy    13 fiedler  22 hankel   31 magic     40 phillips 49 shaw       58␣
↪wilkinson
5 chebspec  14 forsythe 23 heat     32 minij     41 poisson  50 smallworld 59 wing
6 chow      15 foxgood  24 hilb     33 moler     42 prolate  51 spikes
7 circul    16 frank    25 invhilb  34 neumann   43 randcorr 52 toeplitz
8 clement   17 gilbert  26 invol    35 oscillate 44 rando    53 tridiag
9 companion 18 golub    27 kahan    36 parallax  45 randsvd  54 triw

user(#)
-------

Groups
------- ----- ----- ------- ------ ------- --------- ---------------
all     local eigen illcond posdef regprob symmetric test_for_paper2
builtin user  graph inverse random sparse  myfav

Suite Sparse of
------------ ----
2772         2833

MatrixMarket of
------------ ---
488          498
```

Notice new defined groups have been included. We can use them as

```
mdlist(:myfav)
3-element Array{ASCIIString,1}:
```

(continues on next page)

(continued from previous page)

```
 "lehmer"
 "cauchy"
 "hilb"
```

We can remove a group using the macro `@rmgroup`. As before, we need to reload Julia to see the changes.

```
@rmgroup myfav
```

```
listgroups()
 14-element Array{Symbol,1}:
  :all
  :builtin
  :local
  :user
  :eigen
  :graph
  :illcond
  :inverse
  :posdef
  :random
  :regprob
  :sparse
  :symmetric
```

1.8.4 More Examples

An interesting test matrix is magic square. It can be generated as

```
M = matrixdepot("magic", Int, 5)

5x5 Array{Int64,2}:
 17  24   1   8  15
 23   5   7  14  16
  4   6  13  20  22
 10  12  19  21   3
 11  18  25   2   9
```

```
sum(M, dims=1)

1x5 Array{Int64,2}:
 65  65  65  65  65
```

```
sum(M, dims=2)

5x1 Array{Int64,2}:
 65
 65
 65
 65
 65
```

```
sum(diag(M))

65
```

```
p = [5:-1:1]
sum(diag(M[:,p]))

65
```

Pascal Matrix can be generated as

```
P = matrixdepot("pascal", Int, 6)

6x6 Array{Int64,2}:
 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
```

Notice the Cholesky factor of the Pascal matrix has Pascal's triangle rows.

```
cholesky(P)

6x6 UpperTriangular{Float64,Array{Float64,2}}:
 1.0  1.0  1.0  1.0  1.0   1.0
 0.0  1.0  2.0  3.0  4.0   5.0
 0.0  0.0  1.0  3.0  6.0  10.0
 0.0  0.0  0.0  1.0  4.0  10.0
 0.0  0.0  0.0  0.0  1.0   5.0
 0.0  0.0  0.0  0.0  0.0   1.0
```

Bibliography

[bmsz01] G. Boyd, C.A. Micchelli, G. Strang and D.X. Zhou, Binomial matrices, Adv. in Comput. Math., 14 (2001), pp 379-391.

[chow69] T.S. Chow, A class of Hessenberg matrices with known eigenvalues and inverses, SIAM Review, 11 (1969), pp. 391-395.

[clem59] P.A. Clement, A class of triple-diagonal matrices for test purposes, SIAM Review, 1 (1959), pp. 50-52.

[nash90] J.C. Nash, Compact Numerical Methods for Computers: Linear Algebra and Function Minimisation, second edition, Adam Hilger, Bristol, 1990 (Appendix 1).

[todd77] J. Todd, Basic Numerical Mathematics, Vol. 2: Numerical Algebra, Birkhauser, Basel, and Academic Press, New York, 1977, pp. 159.

[vara86] J.M. Varah, A generalization of the Frank matrix, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 835-839.

[vistre98] D. Viswanath and N. Trefethen. Condition Numbers of Random Triangular Matrices, SIAM J. Matrix Anal. Appl. 19, 564-581, 1998.

[nrt92] N.M. Nachtigal, L. Reichel and L.N. Trefethen, A hybrid GMRES algorithm for nonsymmetric linear system, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 796-825.

[high02] Nicholas J. Higham. Accuracy and Stability of Numerical Algorithms, SIAM, PA, USA. 2002.

[hoca63] A.S. Householder and J.A. Carpenter, The singular values of involutory and idempotent matrices, Numer. Math. 5 (1963), pp. 234-237.

[tren89] W.F. Trench, Numerical solution of the eigenvalue problem for Hermitian Toeplitz matrices, SIAM J. Matrix Analysis and Appl., 10 (1989), pp. 135-146

[neto58] M. Newman and J. Todd, The evaluation of matrix inversion programs, J. Soc. Indust. Appl. Math, 6 (1958), pp. 466-476.

[lotk55] 13. Lotkin, A set of test matrices, MTAC, 9, (1955), pp. 153-161.

[plem76] R.J. Plemmons, Regular splittings and the discrete Neumann problem, Numer. Math., 25 (1976), pp. 153-161.

[hansen95] Per Christian Hansen, Test matrices for regularization methods. SIAM J. SCI. COMPUT Vol 16, No2, pp 506-512 (1995)

[part86] S. V. Parter, On the distribution of the singular values of Toeplitz matrices, Linear Algebra and Appl., 80 (1986), pp. 115-130.

[pei62] M.L. Pei, A test matrix for inversion procedures, Comm. ACM, 5 (1962), pp. 508.

[varah93] J.M. Varah. The Prolate Matrix. Linear Algebra and Appl. 187:267–278, 1993.

[rlhk51] Rosser, Lanczos, Hestenes and Karush, J. Res. Natl. Bur. Stand. Vol. 47 (1951), pp. 291-297. Archive

[botr07] L. Bondesson and I. Traat, A Nonsymmetric Matrix with Integer Eigenvalues, Linear and Multilinear Algebra, 55(3)(2007), pp. 239-247.

[gowi76] 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.

[wath87] A.J. Wathen, Realistic eigenvalue bounds for the Galerkin mass matrix, IMA J. Numer. Anal., 7 (1987), pp. 449-457.

[baart82] M.L. Baart, The use of auto-correlation for pseudo-rank determination in noisy ill-conditioned linear least-squares problems, IMA, J. Numer. Anal. 2 (1982), 241-247.

[baker77] C.T.H Baker, The Numerical Treatment of Integral Equations, Clarendon Press, Oxford, 1977, p. 665.

[carasso82] A.S. Carasso, Determining surface temperatures from interior observations, SIAM J. Appl. Math. 42 (1982), 558-574.

[phillips62] D.L. Phillips, A technique for the numerical solution of certain integral equations of the first kind, J. ACM 9 (1962), 84-97.

[ursell] F. Ursell, Introduction to the theory of linear integral equations, Chapter 1 in L. M. Delves and J. Walsh, Numerical Solution of Integral Equations, Clarendon Press, Oxford, 1974.

Index

R

S

T

U

V

W