Package ‘gallery’

September 26, 2024

Type Package

Title Generate Test Matrices for Numerical Experiments

Version 1.0.0

Maintainer Thomas Hsiao <thomas.hsiao@emory.edu>

Description Generates a variety of structured test matrices commonly used in numerical linear algebra and computational experiments. Includes well-known matrices for benchmarking and testing the performance, stability, and accuracy of linear algebra algorithms. Inspired by 'MATLAB' 'gallery' functions.

Encoding UTF-8

RoxygenNote 7.3.1

Imports Matrix, pracma (>= 2.2.5)

Suggests testthat (>= 2.1.0), knitr, rmarkdown

NeedsCompilation no

Author Thomas Hsiao [aut, cre] (<https://orcid.org/0009-0005-7848-7992>)

Repository CRAN

Date/Publication 2024-09-26 13:40:02 UTC

Contents

`binomial_matrix` *Create binomial matrix*

Description

Binomial matrix: an N-by-N multiple of an involutory matrix with integer entries such that $A^2 = 2^(N-1)*I_N$ Thus B = A*2^((1-N)/2) is involutory, that is B^2 = EYE(N)

Usage

```
binomial_matrix(n)
```

Arguments

`n`	row dimension

Value

a binomial matrix, which is a multiple of involutory matrix

`cauchy_matrix` *Create Cauchy matrix*

Description

Arguments `x` and `y` are vectors of length `n`. `C[i,j]` = `1` / `(x[i]` + `y[j])`

Usage

```
cauchy_matrix(x, y = NULL)
```

Arguments

`x`	vector of length n
`y`	vector of length n

Value

a Cauchy matrix

`chebspec` *Create Chebyshev spectral differentiation matrix*

Description

Chebyshev spectral differentiation matrix of order `n`. `k` determines the character of the output matrix. For either form, the eigenvector matrix is ill-conditioned.

Usage

```
chebspec(n, k = NULL)
```

Arguments

`n`	order of the matrix.
`k`	`k=0` is the default, no boundary conditions. The matrix is similar to a Jordan block of size `n` with eigenvalue 0. If `k=1`, the matrix is nonsingular and well-conditioned, and its eigenvalues have negative real parts.

Value

Chebyshev spectral differentiation matrix

`chebvand` *Creating Vandermonde-like matrix for the Chebyshev polynomials*

Description

Produces the (primal) Chebyshev Vandermonde matrix based on the points `p`. `C[i,j] = T_{i-1}p[j]`, where `T_{i-1}` is the Chebyshev polynomial of degree `i-1`

Usage

```
chebvand(p, m = NULL)
```

Arguments

`p`	points to evaluate. If a scalar, then `p` equally spaced points on `[0,1]` are used.
`m`	number of rows of the matrix. `chebvand(p, m)` is the rectangular version of `chebvand(p)` with `m` rows.

Value

Vandermonde-like matrix for the Chebyshev polynomials

`chow` *Creating singular Toeplitz lower Hessenberg matrix*

Description

returns matrix `A = H(alpha) + delta * EYE`, such that `H[i,j] = alpha^(i-j+1)`.

Usage

```
chow(n, alpha = 1, delta = 0)
```

Arguments

`n`	order of the matrix
`alpha`	defaults to 1
`delta`	defaults to 0

Value

Singular Toeplitz lower Hessenberg matrix

`circul` *Create circulant matrix*

Description

Each row is obtained from the previous by cyclically permuting the entries one step forward. A special Toeplitz matrix in which diagonals "wrap around"

Usage

```
circul(v)
```

Arguments

`v`	first row of the matrix. If `v` is a scalar, then `C = circul(1:v)`

Value

a circulant matrix whose first row is the vector `v`

`clement` *Create Clement tridiagonal matrix with zero diagonal entries*

Description

Returns an `n`-by-`n` tridiagonal matrix with zeros on the main diagonal. For `k=0`, `A` is nonsymmetric. For `k=1`, `A` is symmetric

Usage

```
clement(n, k = 0)
```

Arguments

`n`	order of matrix
`k`	0 indicates symmetric matrix, 1 asymmetric

Value

Clement tridiagonal matrix with zero diagonal entries

`compar` *Create comparison matrix* `A`

Description

For `k=0`, if `i==j`, $A[i,j]=abs(B[i,j])$ and `A[i,j]=-abs(B[i,j])` otherwise. For `k=1`, `A` replaces each diagonal element of `B` with its absolute value, and replaces each off-diagonal with the negative of the largest absolute value off-diagonal in the same row.

Usage

```
compar(B, k = 0)
```

Arguments

`B`	input matrix
`k`	decides what matrix to return

Value

Comparison matrix

---

cycol *Create matrix* A *whose columns repeat cyclically*

Description

Returns an n-by-n matrix with cyclically repeating columns where one cycle consists of the columns defined by randn(n,k). Thus, the rank of matrix A cannot exceed k, and k must be scalar.

Usage

```
cycol(n, k, m = NULL)
```

Arguments

n	number of columns of matrix
k	upper limit of rank
m	number of rows of matrix

Value

Matrix whose columns repeat cyclically

---

dorr *Create Dorr matrix*

Description

Returns a n-by-n row diagonally dominant, tridiagonal matrix that is ill-conditioned for small nonnegative values of theta. The default value of theta is 0.01.

Usage

```
dorr(n, theta = 0.01)
```

Arguments

n	order of matrix
theta	determines conditionality. Ill-conditioned when theta is nonnegative.

Value

Dorr matrix of class 'dgcMatrix'.

dramadah *Create anti-Hadamard matrix* `A`

Description

Returns a `n`-by-`n` nonsingular matrix of 0's and 1's. With large determinant or inverse. If `k=1`, `A` is Toeplitz and `abs(det(A))=1`. If `k=2`, `A` is upper triangular and Toeplitz. If `k=3`, `A` has maximal determinant among (0,1) lower Hessenberg matrices. Also is Toeplitz.

Also known as an anti-Hadamard matrix.

Usage

```
dramadah(n, k = 1)
```

Arguments

`n`	order of matrix
`k`	decides type of matrix returned.

Value

Matrix of zeros and ones

fiedler *Create Fiedler matrix*

Description

Fiedler matrix that has a dominant positive eigenvalue and all others are negative

Usage

```
fiedler(c)
```

Arguments

`c`	N-vector. If `c` is a scalar, then returns fiedler(1:c)

Value

a symmetric dense matrix A with a dominant positive eigenvalue and all others are negative.

forsythe — *Create Forsythe matrix or perturbed Jordan block*

Description

Returns a `n`-by-`n` matrix equal to the Jordan block with eigenvalue `lambda`, except that `A[n,1]=alpha`.

Usage

```
forsythe(n, alpha = .Machine$double.eps, lambda = 0)
```

Arguments

`n`	order of matrix
`alpha`	value of perturbation at `A[n,1]`
`lambda`	eigenvalue of Jordan block

Value

Forsythe matrix or perturbed Jordan block

frank — *Frank matrix of order* `N`

Description

Frank matrix of order `N`. It is upper Hessenberg with determinant 1.

Usage

```
frank(n, k = 0)
```

Arguments

`n`	order of the matrix
`k`	If k is 1, the elements are reflected about the anti-diagonal.

Value

Frank matrix with ill-conditioned eigenvalues.

grcar — *Create Toeplitz matrix with sensitive eigenvalues*

Description

Eigenvalues are sensitive.

Usage

```
grcar(n, k = NULL)
```

Arguments

n	dimension of the square matrix
k	number of superdiagonals of ones

Value

n-by-n Toeplitz matrix with -1 on subdiagonal, 1 on diagonal, and k superdiagionals of 1s.

hanowa — *Hanowa matrix*

Description

Matrix whose eigenvalues lie on vertical plane in complex plane. Returns a 2-by-2 block matrix with four n/2 by n/2 blocks. n must be an even integer.

```
[d*eye(m) -diag(1:m), diag(1:m) d*eye(m)]
```

Usage

```
hanowa(n, d = NULL)
```

Arguments

n	order of matrix
d	value of main diagonal

Value

Matrix whose eigenvalues lie on a vertical line in the complex plane.

invol *Involutory matrix (a matrix that is its own inverse)*

Description

a n-byn involutory matrix and ill-conditioned. It is a diagonally scaled version of a Hilbert matrix.

Usage

```
invol(n)
```

Arguments

n	order of matrix

Value

Involutory matrix (a matrix that is its own inverse).

jordbloc *Create Jordan block matrix*

Description

Returns a n-by-n Jordan block with eigenvalue lambda. The default is 1.

Usage

```
jordbloc(n, lambda = 1)
```

Arguments

n	order of matrix
lambda	eigenvalue of Jordan block

Value

Jordan block matrix

`lauchli` *Create Lauchli Matrix*

Description

the (N + 1) x (N) matrix [ones(1,n); mu*eye(n)]. Well-known example in least squares of the danger of forming t(A)

Usage

```
lauchli(n, mu = NULL, sparse = FALSE)
```

Arguments

`n`	number of columns
`mu`	constant applied to identity
`sparse`	whether matrix should be sparse

Value

Lauchli rectangular matrix.

`lehmer` *Create Lehmer matrix*

Description

the symmetric positive-definite matrix such that A[i,j] = i/j, for j >= i

Usage

```
lehmer(n)
```

Arguments

`n`	order of matrix

Value

Lehmer symmetric positive definite matrix.

`leslie` — *Create Leslie population model matrix*

Description

N by N matrix from Leslie population model with average birth and survival rates.

Usage

```
leslie(a, b = NULL, sparse = FALSE)
```

Arguments

`a`	average birth numbers (first row)
`b`	survival rates (subdiagonal)
`sparse`	whether to return a sparse matrix

Value

N by N Leslie population model matrix

`minij` — *Symmetric positive definite matrix* `MIN(i,j)`

Description

The `N`-by-`N` SPD matrix with `A[i,j]=min(i,j)`

Usage

```
minij(n)
```

Arguments

`n`	order of the matrix

Value

Symmetric positive definite matrix with entries `A[i,j]=min(i,j)`

spdiags — *Create sparse diagonal matrix*

Description

Creates a sparse representation of multiple diagonal matrix

Usage

```
spdiags(A, d, m, n)
```

Arguments

`A`	matrix where columns correspond to the desired diagonals
`d`	indices of the diagonals to be filled in. 0 is main diagonal. -1 is first subdiagonal and +1 is first superdiagonal.
`m`	row dim
`n`	col dim

Value

sparse diagonal matrix of class 'dgcMatrix'

tridiag — *Create sparse tridiagonal matrix*

Description

Create a sparse tridiagonal matrix of dgcMatrix class.

Usage

```
tridiag(n, x = NULL, y = NULL, z = NULL)
```

Arguments

`n`	dimension of the square matrix
`x`	subdiagonal (-1)
`y`	diagonal (0)
`z`	superdiagonal (+1)

Value

Sparse tridiagonal matrix of class 'dgcMatrix'

Index