Anymatrix: An Extensible MATLAB Matrix Collection

Higham, Nicholas J. and Mikaitis, Mantas

2021

MIMS EPrint: **2021.16**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: http://eprints.maths.manchester.ac.uk/
And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

Anymatrix: An Extensible MATLAB Matrix Collection

Nicholas J. Higham · Mantas Mikaitis

Version of October 11, 2021

Abstract Anymatrix is a MATLAB toolbox that provides an extensible collection of matrices with the ability to search the collection by matrix properties. Each matrix is implemented as a MATLAB function and the matrices are arranged in groups. Compared with previous collections, Anymatrix offers three novel features. First, it allows a user to share a collection of matrices by putting them in a group, annotating them with properties, and placing the group on a public repository, for example on GitHub; the group can then be incorporated into another user's local Anymatrix installation. Second, it provides a tool to search for matrices by their properties, with Boolean expressions supported. Third, it provides organization into sets, which are subsets of matrices from the whole collection appended with notes, which facilitate reproducible experiments. Anymatrix `v1.0` comes with 146 built-in matrices organized into 7 groups with 49 recognized properties. The authors continue to extend the collection and welcome contributions from the community.

Keywords MATLAB, matrix collection, test matrices

Mathematics Subject Classification (2020) 15B99, 65F99

1 Introduction

In a 1957 "SIAM Notes" column in the SIAM Newsletter (a precursor to SIAM News), Clement obtained the inverses, determinants, and eigenvalues

Nicholas J. Higham
Department of Mathematics, University of Manchester
M13 9PL, Manchester, UK
E-mail: nick.higham@manchester.ac.uk

Mantas Mikaitis
Department of Mathematics, University of Manchester
M13 9PL, Manchester, UK
E-mail: mantas.mikaitis@manchester.ac.uk

of a certain family of tridiagonal matrices [7]. This article was reprinted in SIAM Review [8] two years later. Clement's motivation was that "For testing and experimentation it is important to have matrices for which exact inverses and eigenvalues are known". Many authors have subsequently studied particular matrices and published papers documenting their properties. Some have published collections of matrices, early examples being Rutishauser's article [43], the book by Gregory and Karney [20], and an appendix in the book by Westlake [53].

In addition to their use in computation, matrices with known properties can also be employed in theoretical investigations to test conjectures, explore the sharpness of bounds, and formulate possible new results.

Several collections of test matrices are available in software. These include

- matrices built into MATLAB through the `gallery` function and other MATLAB functions that generate individual matrices;
- the Test Matrix Toolbox [26], [27], much of which was subsequently incorporated into the MATLAB `gallery` function;
- Matrix Market[1] [4];
- the SuiteSparse Matrix Collection (previously known as the University of Florida Sparse Matrix Collection)[2] [10], which contains large, sparse matrices of fixed size, coming from applications;
- CONTEST [51], a MATLAB toolbox providing adjacency matrices from random network models;
- the Regularization Tools toolbox[3] [21], [22], the IR Tools toolbox[4] [17], and the AIR Tools II toolbox[5] [23], all of which contain several test problems, each defining a matrix and a right-hand side;
- the Julia Matrix Depot package [55], which has similar functionality to the MATLAB `gallery` function and also gives access to matrices from Regularization Tools, the SuiteSparse Matrix Collection, and Matrix Market;
- the NLEVP collection[6] of nonlinear eigenvalue problems, each of which is defined by two or more matrices [2], [35].

Matrix Depot and SuiteSparse allow users to contribute to the collection (via `https://sparse.tamu.edu/submit` in the latter case).

Anymatrix is a new MATLAB toolbox, available at `https://github.com/mmikaitis/anymatrix`, that provides an extensible matrix collection and has several key features:

- it is large and easily extensible;
- it includes parametrized matrices and both fixed size and variable size matrices;
- it offers the ability to search for matrices with specific properties;

[1] `http://math.nist.gov/MatrixMarket/`
[2] `https://sparse.tamu.edu/`
[3] `http://www.imm.dtu.dk/~pcha/Regutools/`
[4] `https://github.com/jnagy1/IRtools`
[5] `https://github.com/jakobsj/AIRToolsII`
[6] `https://github.com/ftisseur/nlevp`

Table 1 Features of various matrix collections.

Collection	Location	Search by properties	Extensible by user
Anymatrix	GitHub	Yes	Yes
CONTEST	Own website	No	No
`gallery`	MATLAB	No	No
Matrix Depot	GitHub	No	Yes
Matrix Market	Own website	Yes	No
NLEVP	GitHub	Yes	No
Regularization Tools	Own website	No	No
IR Tools	Own website	No	No
AIR Tools II	Own website	No	No
SuiteSparse	Own website	Yes	No

- it organizes matrices into groups and the user can easily create a new group that behaves like the built-in groups and share it on a public repository; and
- it enables subsets of matrices to be easily created, stored, and re-used to help make experiments reproducible.

The collections listed above do not have all these features; see Table 1. In particular, the MATLAB `gallery` function does not provide a means to search matrices by properties and is not extensible by the user; Matrix Market has not been updated for many years and is not tightly integrated with MATLAB; and SuiteSparse focuses on sparse matrices and provides only matrices of fixed size.

Unlike the NLEVP collection [2], where all matrix integration was done by hand (hard-coded into a function called `nlevp_query`), in Anymatrix properties are defined within the program files defining the matrices or groups and are read in when Anymatrix initializes itself on the first call in a MATLAB session, and it is this feature than makes the extensibility possible.

Our aim in designing Anymatrix was that anyone who comes across interesting matrices in their research should be able to document them and easily make them available for inclusion in Anymatrix.

2 The Anymatrix Toolbox

In this section we outline the design of Anymatrix and how it is used. Full details are given in the users' guide [34].

2.1 Groups of Matrices

The matrices in Anymatrix are organized in groups and there are seven built-in groups:

- `contest`: the matrices from CONTEST [51], listed in Table 2.

- `core`: a miscellaneous selection of matrices, listed in Table 3. Several of these matrices are adapted from the Matrix Computation Toolbox [25].
- `gallery`: the 61 MATLAB `gallery` matrices [24, Table 5.3].
- `hadamard`: 659 Hadamard matrices of dimension up to 428, mostly from the collection of Sloan [46], together with some complex Hadamard matrices from [48]; see Table 4. A Hadamard matrix is a real $n \times n$ matrix with elements ± 1 satisfying $H^T H = nI$. A complex Hadamard matrix has entries of modulus 1 and satisfies $H^* H = nI$.
- `matlab`: matrices that are in MATLAB but are not part of `gallery`, namely the 12 matrices in Table 5.
- `nessie`: matrices from the NESSIE collection coming from real-life networks [52], listed in Table 6.
- `regtools`: matrices from the regularization Tools toolbox [21], [22], listed in Table 7. The whole toolbox, including the regularization routines, is included in Anymatrix, but only the matrices are used. The `regtools` matrices have optional output arguments that generate a right-hand side b and, in some cases, a solution x (which are discretizations of the right-hand side and exact solution of the underlying Fredholm integral equation). We note that the problems in Regularization Tools are discretizations of mainly one-dimensional problems that are regarded as easy problems today. Discretizations of more challenging two-dimensional problems can be found in the IR Tools toolbox [17] and the AIR Tools II toolbox [23].

Every matrix is accessed through an identifier that combines the group name and the matrix name. For example, the 5×5 `beta` matrix from the `core` group is constructed by

Table 2 Matrices in the Anymatrix `contest` group of adjacency matrices from random network models.

Matrix	Description
`baitsample`	Bait and prey subsampling.
`curvature`	Curvatures (clustering coefficients).
`erdrey`	Erdős–Rényi model graph.
`geo`	Geometric random graph.
`gilbert`	Gilbert model graph.
`kleinberg`	Kleinberg model graph.
`lap`	Laplacian matrix (normalized or unnormalized).
`lockandkey`	Lock and key model graph.
`mht`	Mean hitting times.
`pagerank`	PageRank matrix.
`pathlength`	Minimum path lengths.
`pref`	Scale free random graph.
`renga`	Range dependent random graph.
`rewire`	Redirect edges.
`short`	Add shortcuts.
`smallw`	Small world random graph.
`sticky`	Stickiness model random graph.
`unisample`	Uniform subsampling.

Table 3 Matrices in the Anymatrix `core` group.

Matrix	Description
`augment`	Augmented system matrix.
`beta`	Symmetric totally positive matrix of integers.
`blockhouse`	Block Householder matrix.
`dembo9`	Symmetric Hankel matrix of order 9 and rank 5.
`edelman27`	Matrix for which det is computed as the wrong integer.
`fourier`	Fourier matrix.
`gfpp`	Matrix giving maximal growth for Gaussian elim. with pivoting.
`hessfull01`	Totally nonnegative binary lower Hessenberg Toeplitz matrix.
`hessmaxdet`	Upper Hessenberg matrix with maximal determinant.
`kms_nonsymm`	Nonsymmetric Kac–Murdock–Szego Toeplitz matrix.
`nilpot_triang`	Nilpotent triangular matrix.
`nilpot_tridiag`	Nilpotent tridiagonal matrix (sparse).
`rschur`	Upper quasi-triangular matrix.
`soules`	Soules matrix.
`symmstoch`	Symmetric stochastic matrix with given spectrum.
`totally_nonneg`	Nonsingular totally nonnegative matrix.
`triminsval01`	Optimal upper triangular binary Toeplitz matrix.
`vand`	Vandermonde matrix.
`vecperm`	Vec-permutation matrix.
`wilson`	Wilson matrix.
`zielke_nonsymm`	Nonsymmetric matrix of Zielke.
`zielke_symm`	Symmetric matrix of Zielke.

Table 4 Matrices in the Anymatrix `hadamard` group.

Matrix	Description
`complex_hadamard`	Complex Hadamard matrices.
`hadamard`	Hadamard matrices of dimension up to 428.

Table 5 Matrices in the Anymatrix `matlab` group.

Matrix	Description
`compan`	Companion matrix.
`hadamard`	Hadamard matrix.
`hankel`	Hankel matrix.
`hilb`	Hilbert matrix.
`invhilb`	Inverse Hilbert matrix.
`magic`	Magic square.
`pascal`	Pascal matrix.
`rosser`	Classic symmetric eigenvalue test problem.
`spiral`	Matrix with elements in a rectangular spiral pattern.
`toeplitz`	Toeplitz matrix.
`vander`	Vandermonde matrix.
`wilkinson`	Wilkinson's eigenvalue test matrix.

Table 6 Matrices in the Anymatrix `nessie` group from real-life networks.

Matrix	Description
`benguela`	Adjacency matrix for South Africa marine ecosystem network.
`carcorr`	Correlation matrix associated with Scottish car travel times.
`eer`	Adjacency matrix for network of European economic regions.
`gene`	Adjacency matrix for a gene network.
`guppy`	Adjacency matrix for network of guppy social interactions.
`hexgrid`	Adjacency matrix for network from nuclear reactors.
`metabolite`	Adjacency matrix for metabolite network.
`p53`	Adjacency matrix for network of genes related to the oncogene p53.
`ppi`	Adjacency matrix for yeast protein-protein interaction network.
`spl0708a`	Adjacency matrix for network of Scottish football transfers.
`spl0708b`	Adjacency matrix for network of Scottish football transfers.
`spl0809`	Adjacency matrix for network of Scottish football transfers.
`traincorr`	Correlation matrix associated with Scottish train travel times.
`usshelf`	Adjacency matrix of network for Northeast US continental shelf.
`whiskycorr`	Correlation matrix associated with malt whisky tasting.
`whiskydist`	Matrix of Euclidean distances between whisky distilleries.

Table 7 Matrices in the Anymatrix `regtools` group.

Matrix	Description
`baart`	Fredholm integral equation of the first kind.
`blur`	Image deblurring test problem with structured matrix.
`deriv2`	Computation of the second derivative.
`foxgood`	Severely ill-posed problem.
`gravity`	One-dimensional gravity surveying problem.
`heat`	Inverse heat equation.
`i_laplace`	Inverse Laplace transformation.
`parallax`	Stellar parallax problem with 28 fixed observations.
`phillips`	Phillips' "famous" test problem.
`regutm`	Test matrix for regularization methods.
`shaw`	One-dimensional image restoration problem.
`spikes`	Test problem with a "spiky" solution.
`tomo`	Two-dimensional tomography problem with sparse matrix.
`ursell`	Integral equation with no square integrable solution.
`wing`	Test problem with a discontinuous solution.

```
>> B = anymatrix('core/beta',5)
B =
     1     2     3     4     5
     2     6    12    20    30
     3    12    30    60   105
     4    20    60   140   280
     5    30   105   280   630
```

Importantly, as long as the group names are distinct (which is enforced by the directory structure), matrices in different groups have different identifiers even if they have the same matrix name. Each matrix is a MATLAB function, but these functions are not on the MATLAB path (but rather in a private

directory), so there is no danger of namespace clashes with other MATLAB functions.

Help for the `core/beta` matrix is obtained with

```
>> anymatrix('core/beta','help')
 beta   Symmetric totally positive matrix of integers.
    beta(n) is an n-by-n symmetric totally positive matrix of integers.
    It is also infinitely divisible.
    [A,R] = beta(n) returns both the matrix and its explicitly constructed
    Cholesky factor R.
```

Most of the functionality of Anymatrix is implemented with name–value pairs of input parameters, and the name and the value can be given in any order. Anymatrix supports both the comma-separated "name, value" syntax and the "name = value" syntax introduced in MATLAB R2021a. Taking account of the command/function duality of MATLAB, the following lines are equivalent:

```
anymatrix('core/beta','help')
anymatrix('help','core/beta')
anymatrix(help = 'core/beta')
anymatrix help core/beta
```

Moreover, "name" can be abbreviated as long as there is a unique completion, so

```
anymatrix('core/beta','h')
anymatrix core/beta h
anymatrix(h = 'core/beta')
anymatrix h core/beta
```

are further equivalent forms.

The contents of a group, and a cell array of matrix identifiers for the group, can be obtained as follows, where we denote omitted output by `[...]`.

```
>> anymatrix('core','c')

% CORE group
%
%   augment    - Augmented system matrix.
%   beta       - Symmetric totally positive matrix of integers.
%   blockhouse - Block Householder matrix.
%   dembo9     - Symmetric Hankel matrix of order 9 and rank 5.
%   edelman27  - Matrix for which det is computed as the wrong integer.
[...]

>> g = anymatrix('core','g')
g =
  22×1 cell array
    {'core/augment'       }
    {'core/beta'          }
    {'core/blockhouse'    }
    {'core/dembo9'        }
    {'core/edelman27'     }
[...]
```

2.2 Matrix Properties

Every matrix has a set of properties associated with it and it is possible to extract the identifiers of all matrices having a specified set of properties.

For the most part a matrix is assigned a property if it has that property for the default values of any input parameters. An exception is "rectangular", which is assigned if the matrix is rectangular for some choice of input parameters, and if the default shape is square then the matrix is given both the "square" and "rectangular" properties. Examples of matrices having both properties are `gallery/krylov` and `gallery/randsvd`. Likewise a matrix can be "real", "complex", or both.

Anymatrix recognizes the properties listed in Table 8. The properties are not case-sensitive. For definitions of the properties not defined in the table see linear algebra or numerical linear algebra textbooks.

We do not regard the properties defined for real matrices as being included in the corresponding ones for complex matrices, thus "symmetric" is not included in "Hermitian" and "orthogonal symmetric" is not included in "unitary".

The properties of the `core/beta` matrix are obtained with

```
>> props = anymatrix('core/beta','properties')
props =
  12×1 cell array
    {'built-in'            }
    {'infinitely divisible'}
    {'integer'             }
    {'nonnegative'         }
    {'positive'            }
    {'positive definite'   }
    {'real'                }
    {'scalable'            }
    {'square'              }
    {'symmetric'           }
    {'totally nonnegative' }
    {'totally positive'    }
```

We can find which other matrices are nonnegative and either stochastic or totally positive (abbreviating `'properties'` to `'p'`):

```
>> anymatrix('p','nonnegative and (stochastic or totally positive)')
ans =
  5×1 cell array
    {'core/beta'     }
    {'core/symmstoch'}
    {'gallery/cauchy'}
    {'matlab/hilb'   }
    {'matlab/pascal' }
```

In Anymatrix properties are read in when Anymatrix initializes itself on the first call in a MATLAB session. Anymatrix provides a warning if a matrix is detected to have a property not contained in Table 8. Properties can be defined in two ways. The first way is within the functions defining the matrices. A template for a function is

Table 8 Matrix properties.

Property	Comments
banded	
binary	Elements 0 or 1.
block Toeplitz	
built-in	Included in the base Anymatrix collection.
complex	
correlation	Symmetric positive (semi)definite with unit diagonal.
defective	
diagonally dominant	
eigenvalues	Part of the eigensystem is known explicitly.
fixed size	The matrix size is fixed at one or a finite set of values.
Hankel	Constant along the antidiagonals.
Hermitian	
Hessenberg	
idempotent	
ill conditioned	The matrix is ill conditioned for some parameter values.
indefinite	
infinitely divisible	A is symmetric positive semidefinite with nonnegative entries and `A.^r` is positive semidefinite for all $r \geq 0$ [3].
integer	All integer entries.
inverse	The inverse of the matrix is known explicitly.
involutory	$A^2 = I$
M-matrix	
nilpotent	$A^k = 0$ for some positive integer k.
nonnegative	All nonnegative entries.
normal	Normal, but not symmetric/Hermitian/orthogonal/unitary.
orthogonal	
parameter-dependent	Matrix depends on parameters other than the dimension.
permutation	
positive	All positive entries.
positive definite	Applies for both Hermitian and symmetric. Includes semidefinite. Search for positive definite and rank deficient to locate singular semidefinite matrices.
pseudo-orthogonal	Also known as J-orthogonal [29].
random	Random numbers are used in the construction.
rank deficient	
real	
rectangular	
scalable	The problem dimension (or a function of it) is a parameter. Not every dimension is necessarily supported.
singular values	Part of the singular value decomposition is known explicitly.
skew-Hermitian	
skew-symmetric	
sparse	The matrix is stored in the MATLAB sparse format.
square	
stochastic	Nonnegative with unit row sums.
symmetric	
Toeplitz	Constant along the diagonals.
totally nonnegative	Every submatrix has nonnegative determinant.
totally positive	Every submatrix has positive determinant.
triangular	
tridiagonal	
unimodular	Integer entries and determinant ± 1.
unitary	

```
function [A,properties] = mymatrix(n)
%MYMATRIX   My matrix.
%   MYMATRIX(n) return an n-by-n matrix.

properties = {'symmetric', 'indefinite', 'nonnegative', 'toeplitz'};
if nargin == 0, A = []; return, end

% Assign A.
% A = ...
```

This function returns `properties` in its final output argument and the properties can be obtained without generating the matrix by calling the function with no input arguments. As an alternative, a function `am_properties.m` can be placed in the `private` directory of a group, with the properties for all matrices in the group defined within. For example, `am_properties.m` might contain

```
properties = ...
{'mymatrix1', 'symmetric', 'indefinite', 'nonnegative', 'toeplitz';
 'mymatrix2', 'hessenberg', 'toeplitz', 'binary', 'rank deficient';
};
```

This second option makes it easier to incorporate an existing group of matrices, as just one file needs to be edited. For example, for the built-in `gallery` and `matlab` groups we did not modify the corresponding M-files (which are part of MATLAB) but instead wrote an `am_properties.m` function for each group.

2.3 Remote Groups

We have incorporated Git integration to allow remote groups to be downloaded from Git repositories into Anymatrix storage space and treated in the same way as the built-in groups. For example, we can incorporate the following four matrix collections.

- https://github.com/higham/matrices-correlation-invalid: invalid correlation matrices collected by Higham and Strabić [36], [37]. These are matrices that are intended to be correlation matrices but for various reasons relating to their construction have a negative eigenvalue and so are not positive semidefinite.
- https://github.com/higham/hpl-ai-matrix: a parametrized $n \times n$ matrix designed for use in the HPL-AI Mixed Precision Benchmark[7] [14].
- https://github.com/Xiaobo-Liu/matrices-mp-cosm: a collection of matrices used for testing multiprecision algorithms for computing the matrix cosine [1].
- https://github.com/mfasi/randsvdfast-matlab: `randsvdfast`, a function that provides similar functionality to `anymatrix('gallery/randsvd')` but uses a faster algorithm [13].

[7] https://icl.bitbucket.io/hpl-ai/

To incorporate the matrices in the first GitHub repository as a group named `corrinv` we can use the `'groups'` command as follows[8].

```
>> anymatrix('g','corrinv','higham/matrices-correlation-invalid');
Cloning into '[...]/corrinv/private'...
[...]
Anymatrix remote group cloned.
```

Anymatrix automatically rescans the file storage to pick up any new groups and matrices that were downloaded. If the repository is updated we can run the command that was used to create the group again, which runs `git pull` to update the group.

2.4 Sets of Matrices

A matrix set is a user-created subset of all the matrices in an Anymatrix collection, from one or multiple groups, gathered into one named entity. It provides a convenient way to record and later reuse a set of matrices that have something in common, such as being used in an experiment. It thereby aids reproducibility of experiments. A set is defined by creating a text file in `anymatrix/sets/`.

Here is an example usage of Anymatrix sets:

```
% List all sets.
>> anymatrix('sets')
ans =
  1×1 cell array
    {'my_set'}

% Show matrices in a particular set.
>> S = anymatrix('sets','my_set')
ans =
  5×2 cell array
    {'core/dembo9'  }    {1×1 missing   }
    {'core/augment' }    {'[1, 1; 2, 2]'}
    {'gallery/wilk' }    {[           3]}
    {'matlab/pascal'}    {'5, 1'        }
    {'matlab/pascal'}    {'5, 2'        }

% Generate the matrices (the first output is the same as above).
>> [S, D, A, W, P1, P2] = anymatrix('sets','my_set')
[...]
```

In this case `my_set` is the name of the set. The file that describes this set is `anymatrix/sets/my_set.txt` and it has the following contents (it can be viewed with `type my_set.txt`).

```
% This is an example anymatrix set.
% Comments start with a '%' symbol as the first character of the line.

% Following are declarations of some matrices in this set.
```

[8] A full URL can be provided to clone a repository that is not on GitHub.

```
% Dembo9 matrix without any input arguments.
core/dembo9:

% Augment matrix with a matrix as an input argument.
core/augment: [1, 1; 2, 2]

% Wilkinson matrix with a single input.
gallery/wilk: 3

% Two pascal matrices with different input arguments.
matlab/pascal: 5, 1
matlab/pascal: 5, 2
```

A set entry comprises a matrix ID followed by a colon and a set of input arguments, and it is terminated by a new line character. Adding a new set into Anymatrix involves creating such a file and rescanning the filestore with `anymatrix('scan')`. Note that comments can only be made by starting a line with a percentage sign.

2.5 Testing

Anymatrix contains two types of tests. The first type uses the MATLAB unit testing framework to check that the matrices have the claimed properties, which is done mainly for properties that are structural and do not require numerical tolerances. We do not test for positive definiteness or diagonal dominance, for example. The tests are carried out on samples of the matrices for several small dimensions, with default values of any parameters. These tests are run by calling the function `test_anymatrix_properties` located in the `anymatrix/testing` directory.

A second set of tests is specific to the groups. Tests for all the groups that have them can be run by `anymatrix('test')`. Specific group tests can be run by `anymatrix('test',group_name)`. Currently, the `hadamard` group and the `hpl-ai-matrix` and `randsvdfast` downloadable groups have such specific tests.

3 Examples of Use of Anymatrix

Here we provide some examples to illustrate how Anymatrix can be useful in research.

3.1 Growth Factors for LU Factorization

The growth factor for LU factorization on $A \in \mathbb{C}^{n\times n}$ is defined by

$$\rho_n(A) = \frac{\max_{i,j,k} |a_{ij}^{(k)}|}{\max_{i,j} |a_{ij}|},$$

where $A^{(1)} = A$ and the $a_{ij}^{(k)}$ are the elements at the start of the kth stage of LU factorization [28, sect. 9.3], [54]. For complete pivoting, which selects as the pivot element at each stage an element of largest modulus in the active submatrix, it is known that $\rho_n \le n$ usually holds in practice, but that $\rho_n > n$ is possible [11], [18]. Cryer [9, Conjecture 5.1] conjectured that $\rho_n = n$ for any Hadamard matrix. This conjecture is still open [40], [44] and substantial effort was required even to show that $\rho_{16} = 16$ for any 16×16 Hadamard matrix [39]. We will check the value of ρ_n for the real and complex Hadamard matrices in Anymatrix using the following code.

```
%HADAMARD_GROWTH  Growth factors for complete pivoting on Hadamard matrices.

for j = 1:2
switch j
   case 1, matrix = 'hadamard/hadamard'; str = '';
   case 2, matrix = 'hadamard/complex_hadamard'; str = 'complex'
end

[~,dims] = anymatrix(matrix);
tol = 1e2*eps;
nn = length(dims);
for i = 1:nn
   n = dims(i,1); m = dims(i,2);
   fprintf('Testing %3.0f %s Hadamard matrices with n = %2.0f.\n',m,str,n)
   for k = 1:m
      A = anymatrix(matrix,n,k);
      [L,U,P,Q,rho] = gep(A,'c');
      if abs(rho - n) >= tol*rho
          fprintf('Growth %g for n = %g, matrix %g\n', rho,n,k)
      end
   end
end
fprintf('%g matrices tested of dimension up to %g.\n', ...
        sum(dims(:,2)), dims(end,1))

end
```

The code uses the function `gep` from the Matrix Computation Toolbox [25], which implements LU factorization with several pivoting strategies and returns the growth factor. The code tries all 659 Hadamard matrices of dimension up to 428 in the group as well as 9 complex Hadamard matrices of dimension up to 13 (even though Cryer's conjecture does not apply to the latter matrices). The code finds that the growth factor is always equal to n to within the tolerance (which accounts for rounding errors), lending support to the conjecture. The same is true if we repeat the computation for partial pivoting and rook pivoting.

We note that because of rounding errors the pivot sequences may be different from those that would arise in exact arithmetic, so care is needed in interpreting these results; see the discussion in [11].

3.2 Specific Matrices

Anymatrix provides a convenient way to record and make available particular matrices of interest. We mention three examples.

Consider this computation with the matrix `core/edelman27`, a matrix of order 27 whose elements are all ± 1:

```
>> A = anymatrix('core/edelman27');
>> format long g, d1 = det(A), d2 = double(det(vpa(A)))
d1 =
           839466457497601
d2 =
           839466457497600
```

The determinant is obviously an integer and `d2`, computed in 32-digit variable-precision arithmetic (VPA) using the Symbolic Math Toolbox is the exact value. Yet in double precision arithmetic, `det` computes an inexact value. An explanation for this surprising behavior is given in [31].

In numerical linear algebra we often deal with matrices of integers, but the factorizations we compute typically have some noninteger entries. It is not well known that for $n \leq 7$ every symmetric positive definite $n \times n$ matrix A of integers with determinant 1 has a factorization $A = Z^T Z$ with Z an $n \times n$ matrix of integers, though in general such a factorization does not exist for $n \geq 8$. This result is mentioned by Taussky [49, p. 336], [50, pp. 812–813] and goes back to Hermite, Minkowski, and Mordell [42]. Finding an integer factor Z is a nontrivial task. Higham, Lettington, and Schmidt [33] have recently derived conditions for integer factorizations to exist and have developed an approach to computing them. A case in point is the Wilson matrix, a moderately ill-conditioned symmetric positive definite matrix that has a long history as a test matrix. An integer factor was discovered in [32] and two rational factors were discovered in [33]. The matrix and its factors are available through `core/wilson`:

```
>> [W,Z,Y,X] = anymatrix('core/wilson');
>> W, Z, format rat, Y, X
W =
     5     7     6     5
     7    10     8     7
     6     8    10     9
     5     7     9    10
Z =
     2     3     2     2
     1     1     2     1
     0     0     1     2
     0     0     1     1
Y =
       3/2           2           1           0
       1/2           1           0           1
       3/2           2           3           3
       1/2           1           0           0
X =
       3/2           2           2           1
       1/2           1           1           2
```

```
-1/2          -1            1             1
 3/2           2            2             2
```

Here, $W = Z^T Z = Y^T Y = X^T X$, and Z can be thought of as a block Cholesky factor.

A group could be constructed of matrices that provide examples and counterexamples for a particular problem, such as the embeddability problem. This problem, which arises in Markov models, concerns whether a stochastic matrix can be written as the exponential of a matrix with zero row sums and nonnegative diagonal entries. Necessary and sufficient conditions for the existence of a generator are not known. Specific matrices have been identified that demonstrate different possibilities as regards existence of generators and the nature of a generator as a logarithm of the stochastic matrix; see, for example, [5], [30, sec. 2.3], [45]. Collecting such matrices into a group would facilitate numerical experiments in this area of research.

3.3 Matrix Generators

Often one wants to generate many matrices of a particular type, either randomly or by varying parameters. Examples of MATLAB functions with these capabilities are `gallery/randsvd`, for generating random nonsymmetric matrices with a specified singular value distribution or symmetric positive definite matrices with a specified eigenvalue distribution, and `gallery/randcorr`, for generating random correlation matrices with specified eigenvalues. Anymatrix includes two other routines of this type.

The function `core/totally_nonneg`, generates totally nonnegative matrices, which are square matrices for which every submatrix has nonnegative determinant. It uses a representation of such matrices as a product of nonnegative bidiagonal matrices [16] and allows the elements of the factors to be specified or to be chosen randomly.

The function `core/symmstoch` generates symmetric stochastic matrices (symmetric nonnegative matrices with unit row and column sums) with eigenvalues determined by an n-vector of input parameters (which must satisfy a certain nonnegativity condition). The construction makes use of a Soules matrix, an orthogonal matrix of a special type generated by `core/soules` [47].

These functions can be called as `anymatrix('core/totally_nonneg',n)` and `anymatrix('core/symmstoch',n)`, which generate random and nonrandom $n \times n$ matrices, respectively.

3.4 Optimal Matrices

Much research has focused on finding matrices in a given class with largest determinant, largest condition number, smallest singular value, and so on. Anymatrix includes a number of optimal matrices that have been identified, as summarized in Table 9.

Table 9 Matrices with optimality properties.

Matrix	Optimality property
`'core/hessmaxdet',n,d`	Maximum determinant over all $n \times n$ upper Hessenberg matrices with unit subdiagonal and elements in the upper triangle on $[-d, d]$ [12], [15].
`'core/triminsval01',n`	Minimal smallest singular value over all $n \times n$ nonsingular upper triangular binary matrices [41].
`'gallery/dramadah',n,k`	Maximal, or relatively large, determinant or value of $\|A^{-1}\|_F^2$ (depending on k) over all $n \times n$ binary matrices [6], [19].

3.5 Condition Estimator

Our final example shows how to run a test across all the matrices in Anymatrix, which can be useful in various circumstances. MATLAB does not provide any easy way to access sequentially all the matrices in `galllery`.

For the square matrices in Anymatrix, the following code computes the underestimation ratio for the MATLAB condition number estimator `condest`, which uses the algorithm of Higham and Tisseur [38] to compute a lower bound for $\kappa_1(A) = \|A\|_1\|A^{-1}\|_1$. The code filters out matrices defined in the cell array `omit`, which would require a specific dimension or more than one input argument.

```
%CONDEST_TEST  Condition estimator.

n = 16; rng(1)
mats = anymatrix('p','square');
nmats = length(mats);
ratio = zeros(nmats,1);

% Matrices to omit because they require special arguments.
omit = {'contest/mht','contest/unisample','gallery/wathen','gallery/wilk',...
        'hadamard/hadamard','matlab/compan','matlab/hadamard'};

for i = 1:nmats

    fprintf('%s %g/%g.\n',mats{i},i,nmats)

    if ismember(mats{i},omit), continue, end

    props = anymatrix(mats{i},'p');
    % Use matrices from built-in groups but not user-added groups.
    if ~ismember('built-in',props), continue, end
    if ismember('scalable',props)
        A = anymatrix(mats{i},n);
    else
        A = anymatrix(mats{i});
    end
    A = full(A);  % Convert sparse matrices to full.
    if rcond(A) <= 1e-15, continue, end % Skip numerically singular matrices.

    est = condest(A); cond1 = cond(A,1);
```

```
    ratio(i) = est/cond1;

end
ratio = ratio(find(ratio)); % Remove zeros (skipepd matrices).
fprintf('Out of %g matrices:\n',length(ratio))
fprintf('Min = %7.2e, mean = %7.2e, max = %7.2e\n',...
         min(ratio),mean(ratio),max(ratio))
```

The output is

```
contest/baitsample 1/154.
contest/curvature 2/154.
[...]
regtools/wing 154/154.
Out of 86 matrices:
Min = 6.62e-01, mean = 9.91e-01, max = 1.00e+00
```

The results of this experiment are consistent with the experience that the estimator is usually within a factor 3 of the true condition number [38].

4 How to Extend or Contribute to the Collection

Users can extend Anymatrix in several ways. New groups can easily be added, as explained in the users' guide [34, Sec. 10.1]. Such groups are on the same footing as the built-in groups.

One can make a group available to others by putting it in an online GitHub repository, as explained in Section 2.3.

We welcome suggestions for matrices to add to existing groups as well as suggestions for new groups to be added, and will be happy to link to remote groups on the Anymatrix GitHub page.

5 Conclusion

The MATLAB `gallery` matrix collection is a valuable tool, but it is so large that it is difficult to find within it matrices with specific properties. We designed Anymatrix to provide a unified, searchable interface to a wide variety of matrices, in such a way that remote groups can be included, allowing the basic Anymatrix collection to be expanded by users. We have been using Anymatrix in our research throughout its development and find that it matches our needs for finding suitable matrices for testing and experimentation, as well as being a convenient way to make available specific matrices that have been found in the research of us or our colleagues (such as `core/edelman27` and the integer factors of `core/wilson`).

Our code makes use of many recent features of MATLAB, including certain string handling functions and the unit testing framework. Much of it is not specific to organizing matrices and so it can be reused to organize, into groups and sets, any kind of objects appended with properties.

Acknowledgements We thank Des Higham for permission to include CONTEST and NESSIE, Per Christian Hansen for permission to include Regularization Tools, Neil Sloane for permission to include his collection of Hadamard matrices, and Cleve Moler for permission to use his function for the Hadamard matrix of Baumert, Golomb, and Hall.

We thank Bobby Cheng, Mike Croucher, Massimiliano Fasi, Xiaobo Liu, Srikara Pranesh, and Mawussi Zounon for comments and suggestions.

This was supported by MathWorks, the Royal Society, and Engineering and Physical Sciences Research Council grant EP/P020720/1.

References

1. Al-Mohy, A.H., Higham, N.J., Liu, X.: Arbitrary precision algorithms for computing the matrix cosine and its Fréchet derivative. MIMS EPrint 2021.13, Manchester Institute for Mathematical Sciences, The University of Manchester (2021). URL http://eprints.maths.manchester.ac.uk/2832/
2. Betcke, T., Higham, N.J., Mehrmann, V., Schröder, C., Tisseur, F.: NLEVP: A collection of nonlinear eigenvalue problems. ACM Trans. Math. Software **39**(2), 7:1–7:28 (2013). DOI 10.1145/2427023.2427024
3. Bhatia, R.: Infinitely divisible matrices. Amer. Math. Monthly **113**(3), 221–235 (2006). DOI 10.2307/27641890
4. Boisvert, R.F., Pozo, R., Remington, K., Barrett, R.F., Dongarra, J.J.: Matrix Market: A Web resource for test matrix collections. In: R.F. Boisvert (ed.) Quality of Numerical Software: Assessment and Enhancement, pp. 125–136. Chapman and Hall, London (1997)
5. Casanellas, M., Fernández-Sánchez, J., Roca-Lacostena, J.: An open set of 4×4 embeddable matrices whose principal logarithm is not a Markov generator. Linear Multilinear Algebra pp. 1–12 (2020). DOI 10.1080/03081087.2020.1854165
6. Ching, L.: The maximum determinant of an $n \times n$ lower Hessenberg $(0, 1)$ matrix. Linear Algebra Appl. **183**, 147–153 (1993). DOI 10.1016/0024-3795(93)90429-R
7. Clement, P.A.: A class of triple-diagonal matrices for test purposes. SIAM Newsletter **5**(8), 3–5 (1957)
8. Clement, P.A.: A class of triple-diagonal matrices for test purposes. SIAM Rev. **1**(1), 50–52 (1959). DOI 10.1137/1001006
9. Cryer, C.W.: Pivot size in Gaussian elimination. Numer. Math. **12**, 335–345 (1968). DOI 10.1007/BF02162514
10. Davis, T.A., Hu, Y.: The University of Florida Sparse Matrix Collection. ACM Trans. Math. Software **38**(1), 1:1–1:25 (2011). DOI 10.1145/2049662.2049663
11. Edelman, A.: The complete pivoting conjecture for Gaussian elimination is false. The Mathematica Journal **2**, 58–61 (1992)
12. Fasi, M., Feng, J., Negri Porzio, G.M.: Corrigendum to "Determinants of normalized Bohemian upper Hessenberg matrices". Electron. J. Linear Algebra **37**, 160–162 (2021). DOI 10.13001/ela.2021.5849
13. Fasi, M., Higham, N.J.: Generating extreme-scale matrices with specified singular values or condition numbers. SIAM J. Sci. Comput. **43**(1), A663–A684 (2021). DOI 10.1137/20M1327938
14. Fasi, M., Higham, N.J.: Matrices with tunable infinity-norm condition number and no need for pivoting in LU factorization. SIAM J. Matrix Anal. Appl. **42**(1), 417–435 (2021). DOI 10.1137/20m1357238
15. Fasi, M., Negri Porzio, G.M.: Determinants of normalized Bohemian upper Hessenberg matrices. Electron. J. Linear Algebra **36**(36), 352–366 (2020). DOI 10.13001/ela.2020.5053
16. Gasca, M., Peña, J.M.: On factorizations of totally positive matrices. In: M. Gasca, C.A. Micchelli (eds.) Total Positivity and Its Applications, pp. 109–130. Springer-Verlag (1996). DOI 10.1007/978-94-015-8674-0_7
17. Gazzola, S., Hansen, P.C., Nagy, J.G.: IR tools: a MATLAB package of iterative regularization methods and large-scale test problems. Numer. Algorithms **81**(3), 773–811 (2018). DOI 10.1007/s11075-018-0570-7

18. Gould, N.I.M.: On growth in Gaussian elimination with complete pivoting. SIAM J. Matrix Anal. Appl. **12**(2), 354–361 (1991). DOI 10.1137/0612025
19. Graham, R.L., Sloane, N.J.A.: Anti-Hadamard matrices. Linear Algebra Appl. **62**, 113–137 (1984). DOI 10.1016/0024-3795(84)90090-9
20. Gregory, R.T., Karney, D.L.: A Collection of Matrices for Testing Computational Algorithms. Wiley, New York, USA (1969). Reprinted with corrections by Robert E. Krieger, Huntington, New York, 1978
21. Hansen, P.C.: Regularization Tools: A Matlab package for analysis and solution of discrete ill-posed problems. Numer. Algorithms **6**(1), 1–35 (1994). DOI 10.1007/BF02149761
22. Hansen, P.C.: Regularization Tools version 4.0 for Matlab 7.3. Numer. Algorithms **46**(2), 189–194 (2007). DOI 10.1007/s11075-007-9136-9
23. Hansen, P.C., Jørgensen, J.S.: AIR tools II: Algebraic iterative reconstruction methods, improved implementation. Numer. Algorithms **79**(1), 107–137 (2018). DOI 10.1007/s11075-017-0430-x
24. Higham, D.J., Higham, N.J.: MATLAB Guide, third edn. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2017). DOI 10.1137/1.9781611974669
25. Higham, N.J.: The Matrix Computation Toolbox. http://www.maths.manchester.ac.uk/~higham/mctoolbox
26. Higham, N.J.: Algorithm 694: A collection of test matrices in MATLAB. ACM Trans. Math. Software **17**(3), 289–305 (1991). DOI 10.1145/114697.116805
27. Higham, N.J.: The Test Matrix Toolbox for MATLAB (version 3.0). Numerical Analysis Report No. 276, Manchester Centre for Computational Mathematics, UK (1995). URL http://www.maths.manchester.ac.uk/~higham/papers/high95m.pdf
28. Higham, N.J.: Accuracy and Stability of Numerical Algorithms, second edn. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2002). DOI 10.1137/1.9780898718027
29. Higham, N.J.: J-orthogonal matrices: Properties and generation. SIAM Rev. **45**(3), 504–519 (2003). DOI 10.1137/S0036144502414930
30. Higham, N.J.: Functions of Matrices: Theory and Computation. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2008). DOI 10.1137/1.9780898717778
31. Higham, N.J., Edelman, A.: The strange case of the determinant of a matrix of 1s and -1s. https://nhigham.com/2017/12/11/the-strange-case-of-the-determinant-of-a-matrix-of-1s-and-1s/ (2017)
32. Higham, N.J., Lettington, M.C.: Optimizing and factorizing the Wilson matrix. MIMS EPrint 2021.3, Manchester Institute for Mathematical Sciences, The University of Manchester, UK (2021). URL http://eprints.maths.manchester.ac.uk/2803/. To appear in Amer. Math. Monthly
33. Higham, N.J., Lettington, M.C., Schmidt, K.M.: Integer matrix factorisations, superalgebras and the quadratic form obstruction. Linear Algebra Appl. **622**, 250–267 (2021). DOI 10.1016/j.laa.2021.03.028
34. Higham, N.J., Mikaitis, M.: Anymatrix: An extensible MATLAB matrix collection. Users' guide. MIMS EPrint 2021.15, Manchester Institute for Mathematical Sciences, The University of Manchester, UK (2021). URL http://eprints.maths.manchester.ac.uk/2834/
35. Higham, N.J., Negri Porzio, G.M., Tisseur, F.: An updated set of nonlinear eigenvalue problems. MIMS EPrint 2019.5, Manchester Institute for Mathematical Sciences, The University of Manchester, UK (2019). URL http://eprints.maths.manchester.ac.uk/2699/
36. Higham, N.J., Strabić, N.: Anderson acceleration of the alternating projections method for computing the nearest correlation matrix. Numer. Algorithms **72**(4), 1021–1042 (2016). DOI 10.1007/s11075-015-0078-3
37. Higham, N.J., Strabić, N.: Bounds for the distance to the nearest correlation matrix. SIAM J. Matrix Anal. Appl. **37**(3), 1088–1102 (2016). DOI 10.1137/15M1052007
38. Higham, N.J., Tisseur, F.: A block algorithm for matrix 1-norm estimation, with an application to 1-norm pseudospectra. SIAM J. Matrix Anal. Appl. **21**(4), 1185–1201 (2000). DOI 10.1137/S0895479899356080

39. Kravvaritis, C., Mitrouli, M.: The growth factor of a Hadamard matrix of order 16 is 16. Numer. Linear Algebra Appl. **16**, 715–743 (2009). DOI 10.1002/nla.637
40. Kravvaritis, C.D.: Hadamard matrices: Insights into their growth factor and determinant computations. In: M.T. Rassias, V. Gupta (eds.) Mathematical Analysis, Approximation Theory and Their Applications, pp. 383–415. Springer-Verlag, Berlin, Germany (2016). DOI 10.1007/978-3-319-31281-1_17
41. Loewy, R.: On the smallest singular value in the class of invertible lower triangular $(0, 1)$ matrices. Linear Algebra Appl. **608**, 203–213 (2021). DOI 10.1016/j.laa.2020.08.030
42. Mordell, L.J.: The definite quadratic forms in eight variables with determinant unity. J. Math. Pures Appl. **17**(4), 41–46 (1938). URL https://gallica.bnf.fr/ark:/12148/bpt6k6459126x/f53.image
43. Rutishauser, H.: On test matrices. In: Programmation en Mathématiques Numériques, Besançon, 1966, *Éditions Centre Nat. Recherche Sci., Paris*, vol. 7 (no. 165), pp. 349–365 (1968)
44. Seberry, J., Yamada, M.: Hadamard Matrices. Wiley, New Jersey, USA (2020). DOI 10.1002/9781119520252
45. Singer, B., Spilerman, S.: The representation of social processes by Markov models. Amer. J. Sociology **82**(1), 1–54 (1976). DOI 10.1086/226269
46. Sloane, N.J.A.: A library of Hadamard matrices. http://neilsloane.com/hadamard/
47. Soules, G.W.: Constructing symmetric nonnegative matrices. Linear Multilinear Algebra **13**, 241–251 (1983). DOI 10.1080/03081088308817523
48. Tadej, W., Życzkowski, K.: A concise guide to complex Hadamard matrices. Open Systems & Information Dynamics **13**(2), 133–177 (2006). DOI 10.1007/s11080-006-8220-2
49. Taussky, O.: Matrices of rational integers. Bull. Amer. Math. Soc. **66**(5), 327–346 (1960). DOI 10.1090/s0002-9904-1960-10439-9
50. Taussky, O.: Sums of squares. Amer. Math. Monthly **77**(8), 805–830 (1970). DOI 10.1080/00029890.1970.11992594. URL http://www.jstor.org/stable/2317016
51. Taylor, A., Higham, D.J.: CONTEST: A controllable test matrix toolbox for MATLAB. ACM Trans. Math. Software **35**(4), 26:1–26:17 (2009). DOI 10.1145/1462173.1462175
52. Taylor, A., Higham, D.J.: NESSIE: Network example source supporting innovative experimentation. In: E. Estrada, M. Fox, D.J. Higham, G.L. Oppo (eds.) Network Science: Complexity in Nature and Technology, pp. 85–106. Springer-Verlag, London (2010). DOI 10.1007/978-1-84996-396-1_5
53. Westlake, J.R.: A Handbook of Numerical Matrix Inversion and Solution of Linear Equations. Wiley, New York, USA (1968)
54. Wilkinson, J.H.: Error analysis of direct methods of matrix inversion. J. ACM **8**, 281–330 (1961). DOI 10.1145/321075.321076
55. Zhang, W., Higham, N.J.: Matrix Depot: An extensible test matrix collection for Julia. PeerJ Comput. Sci. **2**, e58 (2016). DOI 10.7717/peerj-cs.58