Table of Contents

Housekeeping

```
   clc; clear all; close all;
%{
   CSCI 3656 HW4
   Author: Connor O'Reilly
   Last Edited: 9/23/2021
   Email: coor1752@colorado.edu
%}
```

Download Data

```
%download given matrices
% reference: https://www.mathworks.com/matlabcentral/answers/172652-
how-can-i-convert-a-txt-file-to-mat-file
mat1 = readmatrix('mat1.txt'); mat2 = readmatrix('mat2.txt'); mat3
 = readmatrix('mat3.txt'); mat4 = readmatrix('mat4.txt'); mat5 =
 readmatrix('mat5.txt');
```

Problem 1

use built in matlab function size() to determine dimensions of matrices

```
s = zeros(5,2);
s(1,:) = size(mat1); s(2,:) = size(mat2); s(3,:) = size(mat3); s(4,:)
 = size(mat4); s(5,:) = size(mat5);
```

Problem 2

```
%use built in function find(), returns vector containing linear
 indices of
```

```
%each nonzero element in an array. take size to determine number of
 nonzero
%elements

%get indices of non-zero elements
nz1 = find(mat1); nz2 = find(mat2); nz3 = find(mat3); nz4 =
 find(mat4); nz5 = find(mat5);

%determine amount of nonzero elements in each array
snz = zeros(5,2);
snz(1,:) = size(nz1); snz(2,:) = size(nz2); snz(3,:) = size(nz3);
 snz(4,:) = size(nz4); snz(5,:) = size(nz5);
```

Problem 3

```
%use built in matlab function issymmetric(X) returns logical 1 (true)
 if square matrix X is symmetric; otherwise, it returns logical 0
 (false).
% tests to see if X' = X
sym = zeros(5,1);
sym(1) = issymmetric(mat1); sym(2) = issymmetric(mat2); sym(3)
 = issymmetric(mat3); sym(4) = issymmetric(mat4); sym(5) =
 issymmetric(mat5);
```

Problem 4

```
%use built in matlab function isdiag(X), returns logical 1 ( true ) if
 A is
%a diagonal matrix; otherwise, it returns logical 0 ( false ).
 basically
%checks if non diagonal elements are all zero
dia = zeros(5,1);
dia(1) = isdiag(mat1); dia(2) = isdiag(mat2); dia(3) = isdiag(mat3);
 dia(4) = isdiag(mat4); dia(5) = isdiag(mat5);
```

Problem 5

```
%to determine if a matrix is orthogonal, multiply by its transpose and
 see
%if an identity matrix is returned. AA' = A'A = I

%function at bottom of script
ortho(1) = isOrthogonal(mat1); ortho(2) = isOrthogonal(mat2); ortho(3)
 = isOrthogonal(mat3); ortho(4) = isOrthogonal(mat4); ortho(5) =
 isOrthogonal(mat5);
```

Problem 6

```
%using built in matlab function rank(x) returns the rank of matrix x,
 ie The number of linearly independent columns in a matrix is the rank
 of the matrix. The row and column rank of a matrix are always equal.
```

```
rk(1) = rank(mat1); rk(2) = rank(mat2); rk(3) = rank(mat3); rk(4) =
 rank(mat4); rk(5) = rank(mat5);
```

Problem 7 & 8

using built in matlab function, S = svd(A) returns the singular values of matrix A in descending order.

```
sm_sv = zeros(5,1);
lg_sv = zeros(5,1);

sv1 = svd(mat1); sv2 = svd(mat2); sv3 = svd(mat3); sv4 = svd(mat4);
 sv5 = svd(mat5);

%get smallest singular values
sm_sv(1) = min(sv1); sm_sv(2) = min(sv2); sm_sv(3) = min(sv3);
 sm_sv(4) = min(sv4); sm_sv(5) = min(sv5);
%get largest singular values
lg_sv(1) = max(sv1); lg_sv(2) = max(sv2); lg_sv(3) = max(sv3);
 lg_sv(4) = max(sv4); lg_sv(5) = max(sv5);
```

Problem 9

```
%can either take the ratio of the largest and smallest singular values
 or
%use matlab cond

%check to see they are the same
%they are the same just going to use cond
% cond = sm_sv(1)/lg_sv(1)
cnd = zeros(5,1);
cnd(1) = cond(mat1); cnd(2) = cond(mat2); cnd(3) = cond(mat3); cnd(4)
 = cond(mat4); cnd(5) = cond(mat5);
```

Problem 10

create five random right hand sides

```
b1 = rand(s(1,1),1); b2 = rand(s(2,1),1); b3 = rand(s(3,1),1); b4 =
 rand(s(4,1),1); b5 = rand(s(5,1),1);
%linsolve
%X = linsolve(A,B) solves the linear system AX = B using LU
 factorization
%with partial pivoting
x1 = linsolve(mat1,b1);
x2 = linsolve(mat2,b2);

%%%%%%%%%%%%%%%
x3 = linsolve(mat3,b3); %LHS is all NaN or inf values
%%%%%%%%%%%%%%%

x4 = linsolve(mat4,b4);
```

```
x5 = linsolve(mat5,b5);
```

Warning: Matrix is singular to working precision.

Display

```
% Problem 1 Display
fprintf('Problem One: \n')
for i = 1:5

    fprintf('matrix in file: %d,  rows: %d, columns: %d\n', i ,
 s(i,1), s(i,2))

end

% Problem 2 Display
fprintf('\nProblem Two: \n')
for i = 1:5

    fprintf('matrix in file: %d, number of nonzero elements: %d \n',
 i ,  snz(i,1))

end

% Problem 3 Display
fprintf('\nProblem Three: \n')
for i = 1:5
    str3 = 'false';
    if( sym(i) )
        str3 = 'true';
    end
    fprintf('matrix in file: %d, Symmetrical: %s \n', i ,  str3)

end

% Problem 4 Display
fprintf('\nProblem Four: \n')
for i = 1:5
    dia4 = 'false';
    if( dia(i) )
        dia4 = 'true';
    end
    fprintf('matrix in file: %d, Diagonal: %s \n', i ,  dia4)

end

% Problem 5 Display
fprintf('\nProblem Five: \n')
for i = 1:5
    ortho5 = 'false';
    if( ortho(i) )
        ortho5 = 'true';
    end
```

```
    fprintf('matrix in file: %d, Orthogonal: %s \n', i ,  ortho5)

end

% Problem 6 Display
fprintf('\nProblem Six: \n')
for i = 1:5

    fprintf('matrix in file: %d, Rank: %d \n', i ,  rk(i))

end

% Problem 7 Display
fprintf('\nProblem Seven: \n')
for i = 1:5

    fprintf('matrix in file: %d, Smallest SV: %0.17f \n', i ,
 sm_sv(i))

end

% Problem 8 Display
fprintf('\nProblem Eight: \n')
for i = 1:5

    fprintf('matrix in file: %d, Largest SV: %0.5f \n', i ,  lg_sv(i))

end

% Problem 9 Display
fprintf('\nProblem Nine: \n')
for i = 1:5

    fprintf('matrix in file: %d, Conditional Number: %0.5f \n', i ,
 cnd(i))

end

% Problem 10 Display
fprintf('\nProblem 10: \n')
fprintf('Solver had issues solving system 3, all values in the
 solution matrix were either NaN or inf values. Probably due to large
 conditinal number.\n')
```

```
Problem One:
matrix in file: 1,  rows: 10, columns: 10
matrix in file: 2,  rows: 30, columns: 30
matrix in file: 3,  rows: 400, columns: 400
matrix in file: 4,  rows: 50, columns: 50
matrix in file: 5,  rows: 625, columns: 625

Problem Two:
matrix in file: 1, number of nonzero elements: 55
matrix in file: 2, number of nonzero elements: 900
```

```
matrix in file: 3, number of nonzero elements: 800
matrix in file: 4, number of nonzero elements: 2500
matrix in file: 5, number of nonzero elements: 3025

Problem Three:
matrix in file: 1, Symmetrical: false
matrix in file: 2, Symmetrical: true
matrix in file: 3, Symmetrical: false
matrix in file: 4, Symmetrical: false
matrix in file: 5, Symmetrical: true

Problem Four:
matrix in file: 1, Diagonal: false
matrix in file: 2, Diagonal: false
matrix in file: 3, Diagonal: false
matrix in file: 4, Diagonal: false
matrix in file: 5, Diagonal: false

Problem Five:
matrix in file: 1, Orthogonal: false
matrix in file: 2, Orthogonal: false
matrix in file: 3, Orthogonal: false
matrix in file: 4, Orthogonal: true
matrix in file: 5, Orthogonal: false

Problem Six:
matrix in file: 1, Rank: 10
matrix in file: 2, Rank: 30
matrix in file: 3, Rank: 399
matrix in file: 4, Rank: 50
matrix in file: 5, Rank: 625

Problem Seven:
matrix in file: 1, Smallest SV: 0.03001825292422508
matrix in file: 2, Smallest SV: 0.19847856546217052
matrix in file: 3, Smallest SV: 0.00000000000000001
matrix in file: 4, Smallest SV: 0.99999999999999933
matrix in file: 5, Smallest SV: 0.02916450360778412

Problem Eight:
matrix in file: 1, Largest SV: 3.73426
matrix in file: 2, Largest SV: 41.02009
matrix in file: 3, Largest SV: 2.00000
matrix in file: 4, Largest SV: 1.00000
matrix in file: 5, Largest SV: 7.97084

Problem Nine:
matrix in file: 1, Conditional Number: 124.39976
matrix in file: 2, Conditional Number: 206.67266
matrix in file: 3, Conditional Number: 140121177821135008.00000
matrix in file: 4, Conditional Number: 1.00000
matrix in file: 5, Conditional Number: 273.30606

Problem 10:
```

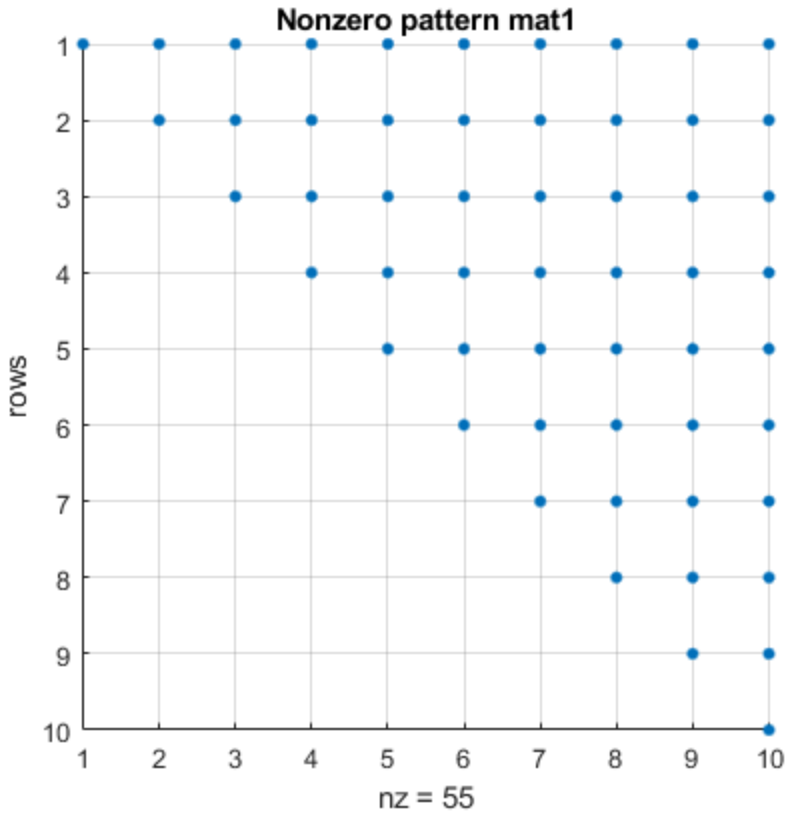
Solver had issues solving system 3, all values in the solution matrix were either NaN or inf values. Probably due to large conditinal number.

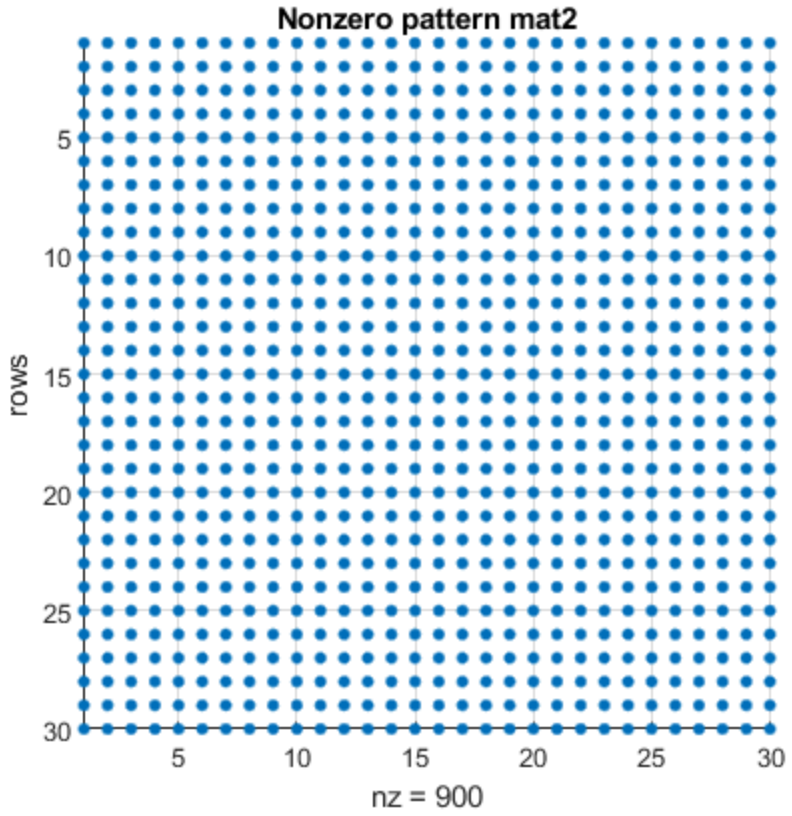
Plotting

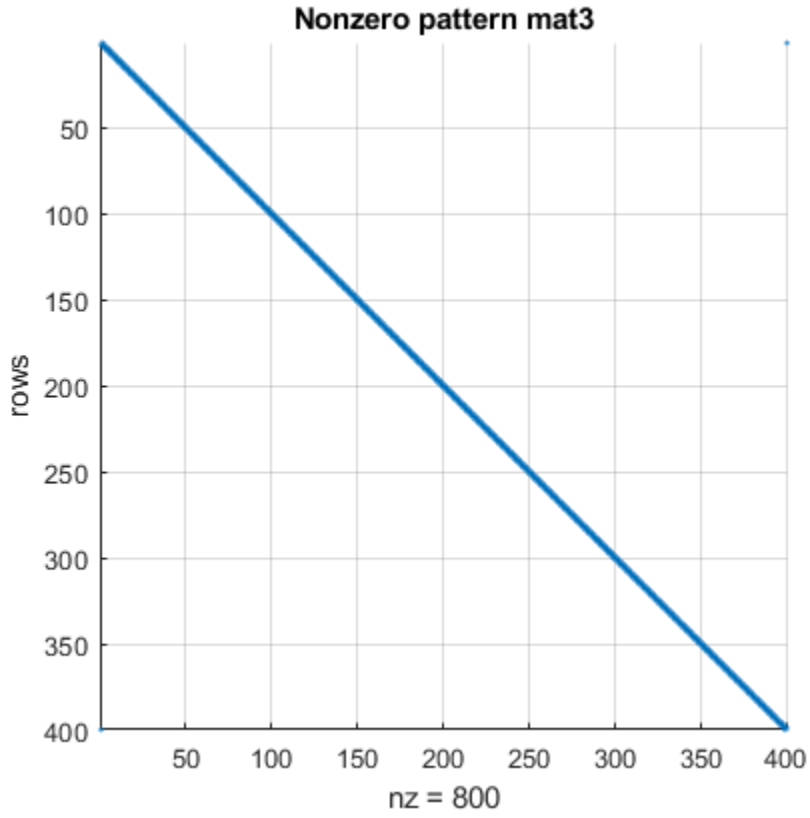
```
% Plot the nonzero elements of the matrix, use spy

%mat1
figure(1)
title('Nonzero pattern mat1')
hold on;
grid on;
ylabel('rows');
xlabel('columns');
spy(mat1)
axis equal;
axis tight;
hold off;

%mat2
figure(2)
title('Nonzero pattern mat2')
hold on;
grid on;
ylabel('rows');
xlabel('columns');
spy(mat2)
axis equal;
axis tight;
hold off;

%mat3
figure(3)
title('Nonzero pattern mat3')
hold on;
ylabel('rows');
xlabel('columns');
grid on;
spy(mat3)
axis equal;
axis tight;
hold off;

%mat4
figure(4)
title('Nonzero pattern mat4')
hold on;
ylabel('rows');
xlabel('columns');
grid on;
spy(mat4)
axis equal;
```

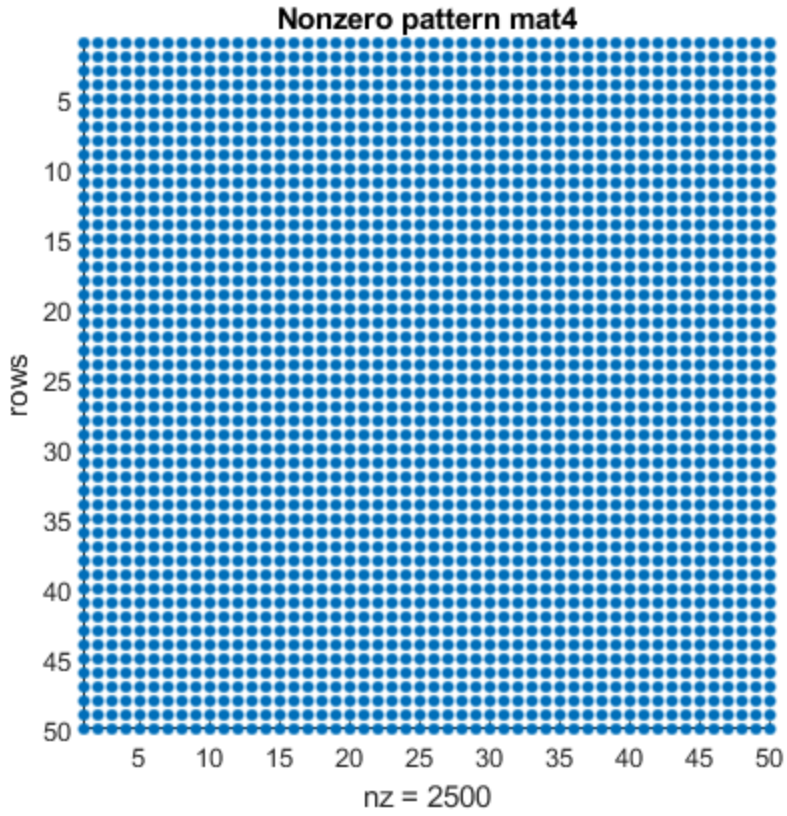
```
axis tight;
hold off;

%mat5
figure(5)
title('Nonzero pattern mat5')
hold on;
ylabel('rows');
xlabel('columns');
grid on;
spy(mat5)
axis equal;
axis tight;
hold off;

%magnitude of the elements of the matrix

%mat1
figure(6);
imagesc(mat1);
title('Size of mat1 elements')
xlabel('columns')
ylabel('rows')
hold on;
axis equal;
axis tight;
colorbar
hold off;

%mat2
figure(7);
imagesc(mat2);
title('Size of mat2 elements')
xlabel('columns')
ylabel('rows')
hold on;
axis equal;
axis tight;
colorbar
hold off;

%mat3
figure(8);
imagesc(mat3);
title('Size of mat3 elements')
xlabel('columns')
ylabel('rows')
hold on;
axis equal;
axis tight;
colorbar
hold off;

%mat4
```

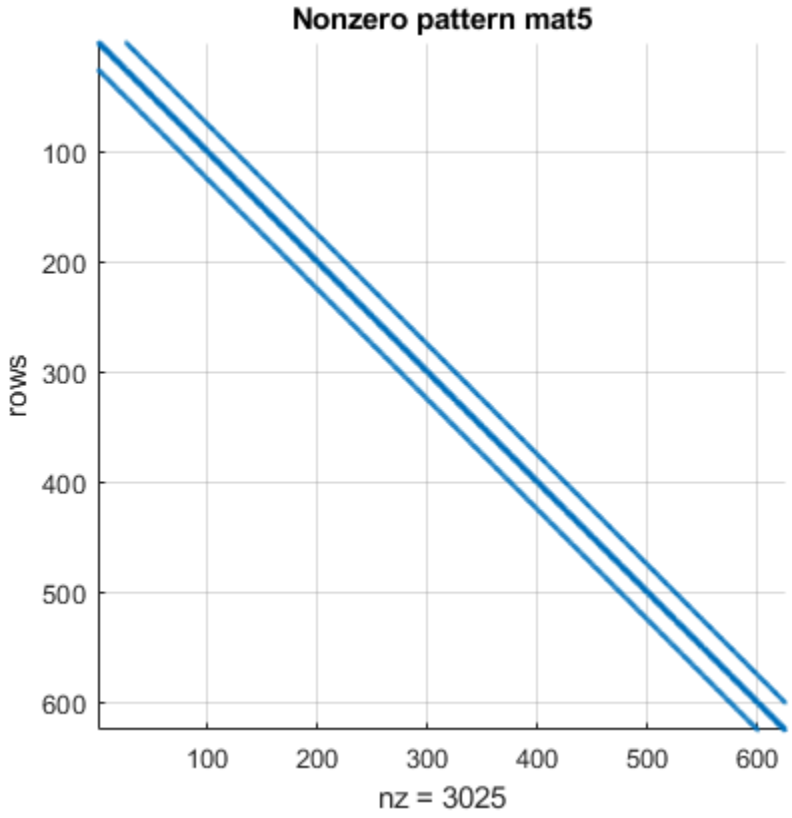
```
figure(9);
imagesc(mat4);
title('Size of mat4 elements')
xlabel('columns')
ylabel('rows')
hold on;
axis equal;
axis tight;
colorbar
hold off;

%mat5
figure(10);
imagesc(mat5);
title('Size of mat5 elements')
xlabel('columns')
ylabel('rows')
hold on;
axis equal;
axis tight;
colorbar
hold off;
```

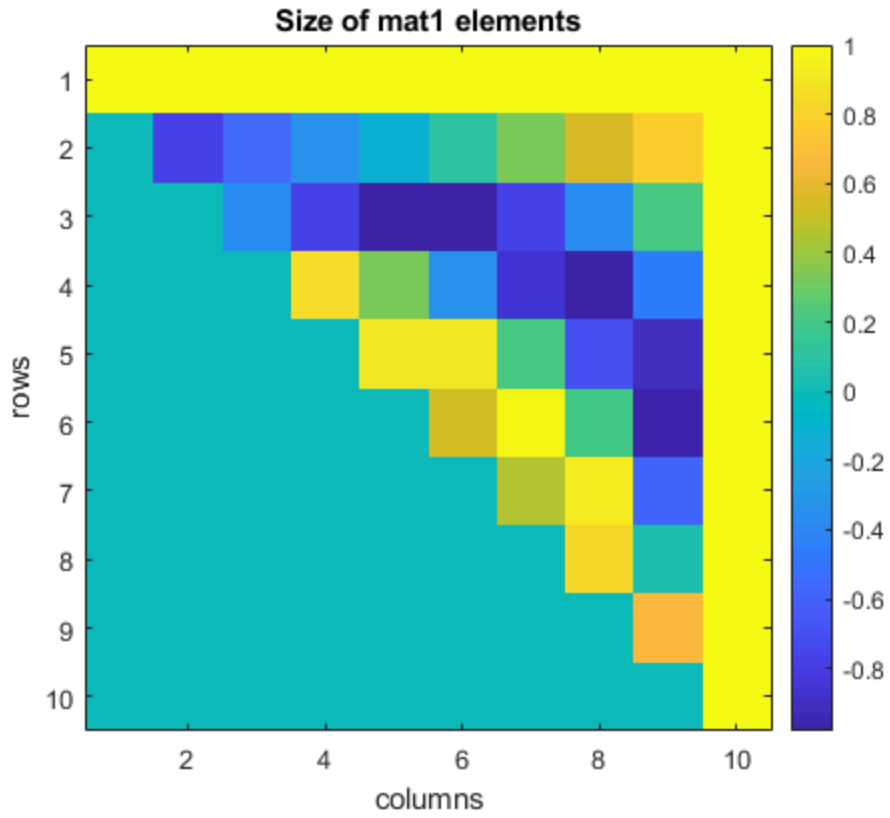


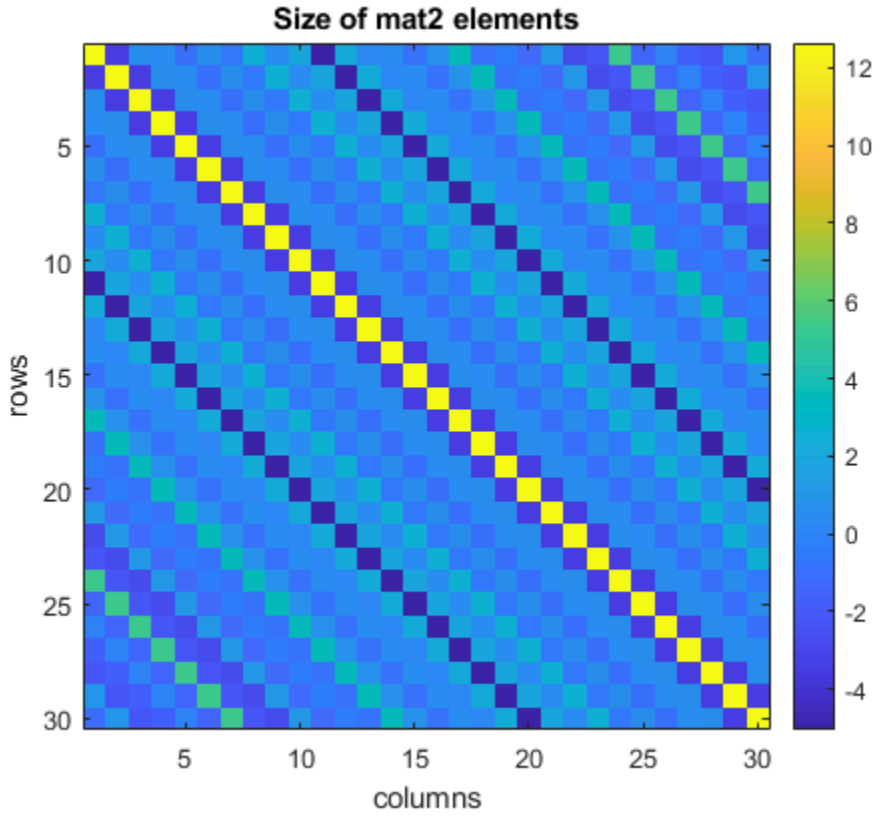

Nonzero pattern mat2
rows
5
10
15
20
25
30
5
10
15
20
25
30
nz = 900

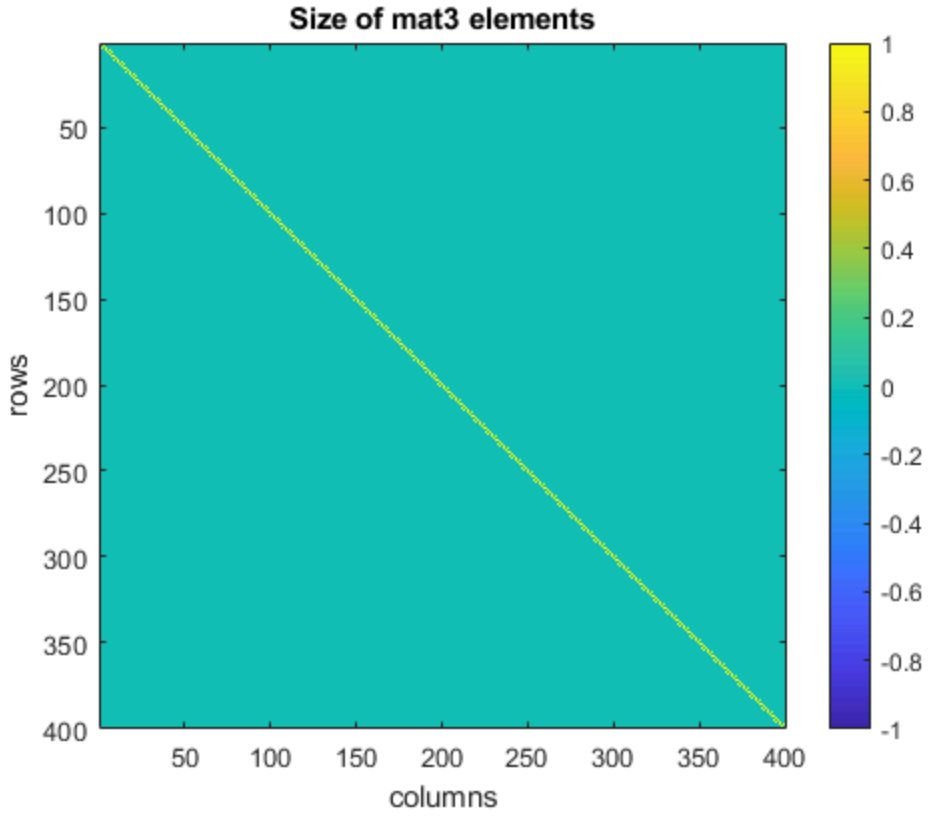

Nonzero pattern mat3
rows
50
100
150
200
250
300
350
400
50
100
150
200
250
300
350
400
nz = 800

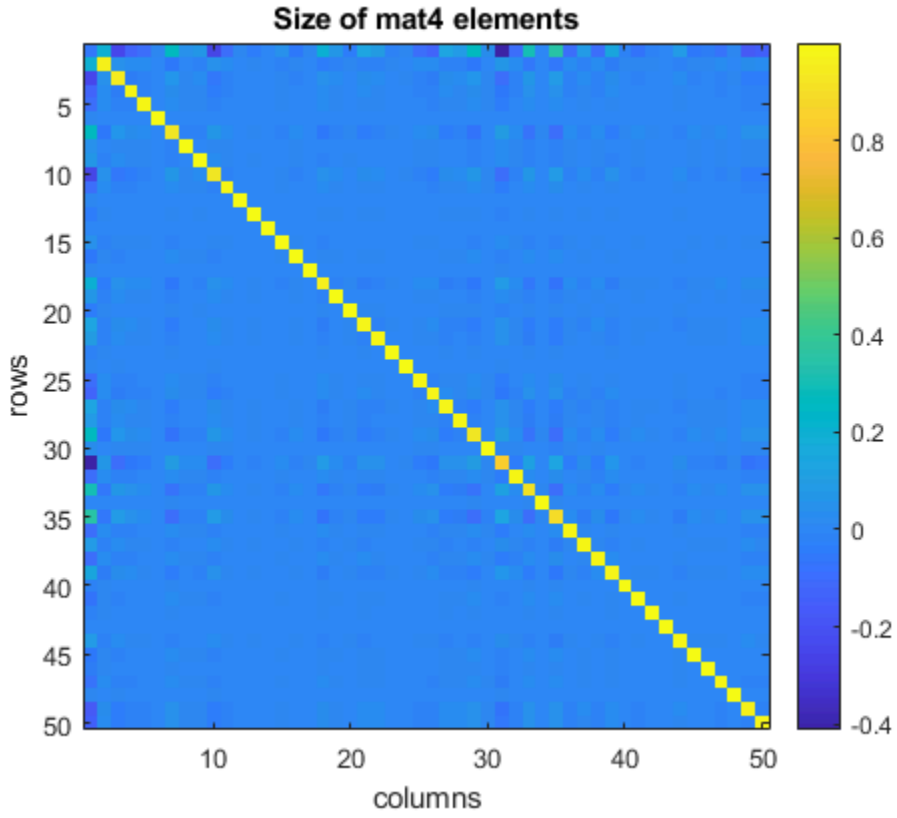

Nonzero pattern mat4
rows
5
10
15
20
25
30
35
40
45
50
5
10
15
20
25
30
35
40
45
50
nz = 2500

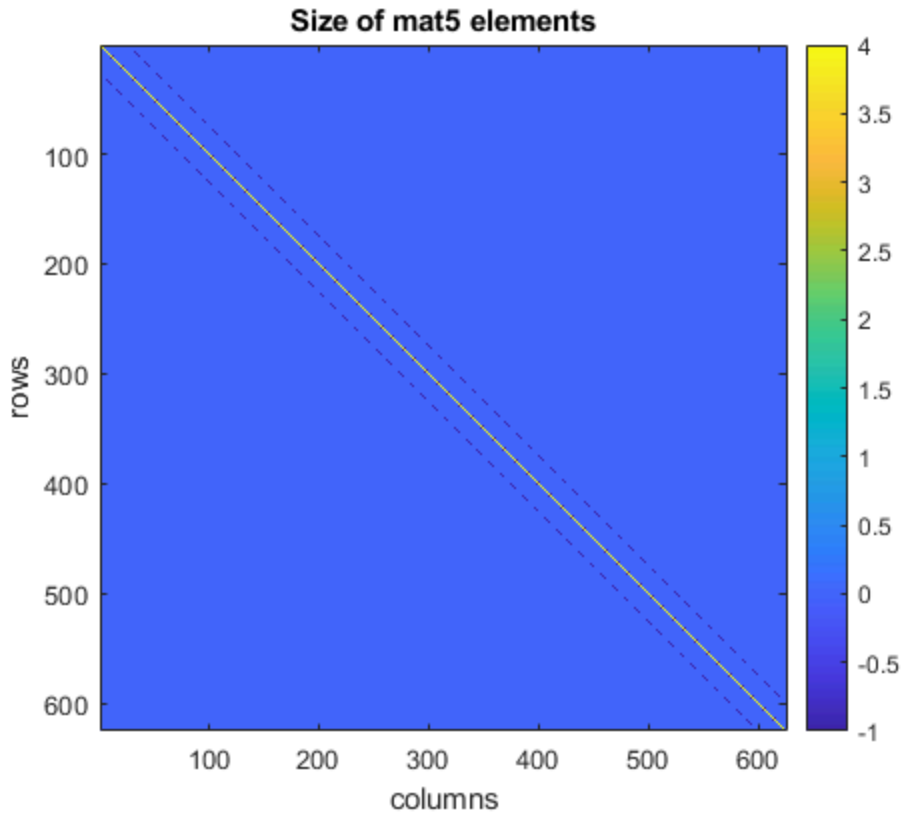

Nonzero pattern mat5
rows
100
200
300
400
500
600
100
200
300
400
500
600
nz = 3025


Size of mat1 elements
rows
columns
1
2
3
4
5
6
7
8
9
10
2
4
6
8
10
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8


Size of mat2 elements
rows
columns
5
10
15
20
25
30
5
10
15
20
25
30
12
10
8
6
4
2
0
-2
-4


Size of mat3 elements
rows
columns
50
100
150
200
250
300
350
400
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1


Size of mat4 elements
rows
columns
5
10
15
20
25
30
35
40
45
50
0.8
0.6
0.4
0.2
0
-0.2
-0.4



Functions

```
function y = isOrthogonal(x)
%determines if square matrix x is orthogonal
i = eye(size(x));
  if(isequal(round(x*x'),i))
      y = true;
  else
      y  = false;
  end
end
```

Published with MATLAB® R2021a