Using MATLAB in Lineal Algebra

Group 11A: Pol Puigdemont, Marc Sardà

link: https://drive.matlab.com/sharing/2b968c42-abd5-45ce-a073-bd2ba5db77d7

1 - Variable Declaration

```
a = 3;
b = 0.5;
c = -2.7;
u = [1; 2; 3];
v = [-2; 1.5; 1];
s = [-1 -0.4 2 1];
w = [4 -2 6 3]
```

```
w = 1×4
     4    -2     6     3
```

```
A = [1 2 3 4; -1 0 5 2; -1 2 8 1; 0 5 3 1]
```

```
A = 4×4
     1     2     3     4
    -1     0     5     2
    -1     2     8     1
     0     5     3     1
```

```
B = [0.28 -0.45 0.84 1.01; 0.83 -0.30 -0.45 1.99;
    0.46 0.83 0.29 3.03; 0 0 0 1.00]
```

```
B = 4×4
    0.2800   -0.4500    0.8400    1.0100
    0.8300   -0.3000   -0.4500    1.9900
    0.4600    0.8300    0.2900    3.0300
         0         0         0    1.0000
```

```
C = [1:4 ; 5:8]
```

```
C = 2×4
     1     2     3     4
     5     6     7     8
```

2 - MATLAB Matrix and array arithmetic (read-only)

3 - Operations

Perform operations specified in the statement sheet by order, those who are not correct upon the previous definitions are commented.

```
u+v
```

```
ans = 3×1
   -1.0000
    3.5000
    4.0000
```

```
u-v
```

```
ans = 3×1
    3.0000
    0.5000
    2.0000
```

```
u'
```

```
ans = 1×3
     1     2     3
```

```
a*v
```

```
ans = 3×1
   -6.0000
    4.5000
    3.0000
```

```
% u*v inc. dims.
u.*v
```

```
ans = 3×1
    -2
     3
     3
```

```
u*v'
```

```
ans = 3×3
   -2.0000    1.5000    1.0000
   -4.0000    3.0000    2.0000
   -6.0000    4.5000    3.0000
```

```
dot(u,v)
```

```
ans = 4
```

```
cross(u,v)
```

```
ans = 3×1
   -2.5000
   -7.0000
    5.5000
```

```
dot(s,w)
```

```
ans = 11.8000
```

```
% cross(w,w) non-3d
A'
```

```
ans = 4×4
     1    -1    -1     0
     2     0     2     5
     3     5     8     3
     4     2     1     1
```

```
inv(A)
```

```
ans = 4×4
```

```
    0.5408   -1.1939    0.7347   -0.5102
   -0.0612    0.0408   -0.1020    0.2653
    0.0714   -0.2143    0.2857   -0.1429
    0.0918    0.4388   -0.3469    0.1020
```

```
A+B
```

```
ans = 4×4
    1.2800    1.5500    3.8400    5.0100
   -0.1700   -0.3000    4.5500    3.9900
   -0.5400    2.8300    8.2900    4.0300
         0    5.0000    3.0000    2.0000
```

```
A-B
```

```
ans = 4×4
    0.7200    2.4500    2.1600    2.9900
   -1.8300    0.3000    5.4500    0.0100
   -1.4600    1.1700    7.7100   -2.0300
         0    5.0000    3.0000         0
```

```
A*B
```

```
ans = 4×4
    3.3200    1.4400    0.8100   18.0800
    2.0200    4.6000    0.6100   16.1400
    5.0600    6.4900    0.5800   28.2100
    5.5300    0.9900   -1.3800   20.0400
```

```
A.*B
```

```
ans = 4×4
    0.2800   -0.9000    2.5200    4.0400
   -0.8300         0   -2.2500    3.9800
   -0.4600    1.6600    2.3200    3.0300
         0         0         0    1.0000
```

```
% A*C inc. dims.
C*A
```

```
ans = 2×4
    -4    28    49    15
    -8    64   125    47
```

```
% A.*C inc. dims.
% C.*A inc. dims.
w*A
```

```
ans = 1×4
     0    35    59    21
```

```
% A*w inc. dims
```

4 - Indexing

Perform the required extractions in the statement sheet by order

```
A(2,4)
```

```
ans = 2
```

```
A(2,:)
```

```
ans = 1×4
    -1     0     5     2
```

```
B(:,4)
```

```
ans = 4×1
    1.0100
    1.9900
    3.0300
    1.0000
```

```
B(1:3, 1:3)
```

```
ans = 3×3
    0.2800   -0.4500    0.8400
    0.8300   -0.3000   -0.4500
    0.4600    0.8300    0.2900
```

5 - Special Matrices

Zeros creates a zero-element matrix of the specified (rows, cols). 9x10 example:

```
rows = 9
```

```
rows = 9
```

```
cols = 10
```

```
cols = 10
```

```
zeros(rows, cols)
```

```
ans = 9×10
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0
```

Ones does the same as zeros but for one-element matrices. 9x10 example:

```
ones(rows, cols)
```

```
ans = 9×10
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
```

```
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
     1     1     1     1     1     1     1     1     1     1
```

Eye generates a diagonal matrix of the specified size. If rows != cols it generates the diagonal submatrix of the diagonal square matrix of the largest dimension. 9x10 example and 10x10 example respectively:

```
eye(rows,cols)
```

```
ans = 9×10
     1     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0
     0     0     1     0     0     0     0     0     0     0
     0     0     0     1     0     0     0     0     0     0
     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     0     1     0     0     0     0
     0     0     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     0     0     1     0
```

```
eye(10) % squared
```

```
ans = 10×10
     1     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0
     0     0     1     0     0     0     0     0     0     0
     0     0     0     1     0     0     0     0     0     0
     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     0     1     0     0     0     0
     0     0     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     0     0     0     1
```

Rand generates a matrix of the specified rows and cols with random inputs following a uniform (0,1) distribution. 9x10 example:

```
rand(rows, cols)
```

```
ans = 9×10
    0.3043    0.2698    0.9522    0.0553    0.2503    0.8960    0.5164    0.6127 ···
    0.2909    0.9897    0.5433    0.7538    0.4884    0.1900    0.0075    0.3008
    0.2425    0.1837    0.2514    0.1319    0.7290    0.0018    0.6889    0.7981
    0.9367    0.8617    0.5786    0.3559    0.2026    0.7118    0.9460    0.7956
    0.8602    0.0326    0.9155    0.3959    0.2163    0.8677    0.8735    0.7811
    0.3972    0.3320    0.8956    0.8855    0.9763    0.1183    0.1133    0.3511
    0.4794    0.7487    0.4825    0.0212    0.5932    0.0390    0.3546    0.0543
    0.5650    0.6444    0.4427    0.8441    0.3044    0.5982    0.2419    0.7087
    0.4896    0.1692    0.3118    0.2881    0.9677    0.6043    0.5603    0.9929
```

```
% if we wanted them in another interval, say for example
% (-9,9) we can simply scale
-9 + (18)*rand(rows, cols)
```

```
ans = 9×10
    1.3524    5.4344    1.7284    8.1215    4.3352    8.5759    5.7081    6.5509 ···
    4.5177   -0.0467    5.5543   -4.5189    4.2774    0.4194    0.5703   -5.4259
   -6.2367    0.6812    8.7216   -2.0444    8.0445   -1.2615    0.3802    3.1041
   -2.5778    6.6764    6.9466   -1.2342    0.1818   -5.2712    4.9376    7.2330
   -6.4089    4.0112   -5.1509    5.9560    5.2538   -3.1788   -6.8353   -5.4152
```

```
    6.3109    3.0255   -8.3767    5.8436   -0.8609   -7.0043    2.2581   -3.6309
   -2.9183   -5.7811   -0.8798   -0.8460    6.2856   -2.2462   -2.7603   -0.0626
   -4.0465    0.9089   -8.7517   -2.1499   -1.9722   -3.0617   -2.9769    7.0183
   -8.8918    8.2778   -0.4732    7.6656    4.2908   -2.8421    1.3431    0.0255
```

Randn generates a matrix of the specified rows and cols with random inputs following a normal (0,1) distribution. 9x10 example:

```
randn(rows, cols)
```

```
ans = 9×10
    1.7773    0.6592    0.0565    1.1520    1.5421    0.5776   -0.8784    1.6891 ···
   -0.7795   -1.2804   -0.8933    0.8584   -1.7749    0.2431   -0.7619    1.4370
   -0.7530    0.0501    1.1221    0.9456    0.2057    1.4677    0.8628   -2.2511
   -1.0331    0.5484    0.7758    1.5061   -0.3462    1.1306    0.6483    0.3565
    1.1638    1.7784    0.9625    0.2953   -1.1590    0.0107    1.0581   -0.8502
   -0.5801    0.6411    1.2608    1.2041    0.3358    0.1243   -0.6332   -0.2996
    0.4173    0.9467   -0.1079   -0.0118    0.3322    1.7794    1.0180   -0.6342
   -1.6481   -0.3543   -0.5772    0.7895    1.4112   -0.2395    0.1719    1.6245
   -0.5727   -0.5788    0.5256   -1.4036    0.1369    0.8580   -0.0475    1.2411
```

```
% if we want to specify the mean value and the covariance
% matrix for a custom N(mean, cov) distribution we can adjust
% as with the following code from the documentation examples
% different dims than in the example
mu = [1 2];
sigma = [1 0.5; 0.5 2];
R = chol(sigma);
repmat(mu,10,1) + randn(10,2)*R
```

```
ans = 10×2
    1.3270    1.9739
    1.7530    4.2179
   -0.1537    1.8289
    0.5921    1.0854
   -0.2879    1.0532
    1.0836    1.3324
    1.1638    3.9855
    1.6826    1.6652
   -0.0864   -0.6688
    1.2975    1.8825
```

Magic generates a squared matrix of the specified dimension n where the matrix is constructed with elements from 1 to n^2 and all the rows and columns sum the same. As specified in the docs, n must be a scalar value greater or equal to 3 for a valid example.

```
magic(9)
```

```
ans = 9×9
    47    58    69    80     1    12    23    34    45
    57    68    79     9    11    22    33    44    46
    67    78     8    10    21    32    43    54    56
    77     7    18    20    31    42    53    55    66
     6    17    19    30    41    52    63    65    76
    16    27    29    40    51    62    64    75     5
    26    28    39    50    61    72    74     4    15
    36    38    49    60    71    73     3    14    25
    37    48    59    70    81     2    13    24    35
```

```
% init the parameters from the example
A = [2 -1 3; 1 2 2; -1 5 -2];
b = [11.5; 6; -9.5];

% solve the A*X = b system
X = A\b;

% display the result
X
```

```
X = 3×1
    1.0000
   -0.5000
    3.0000
```