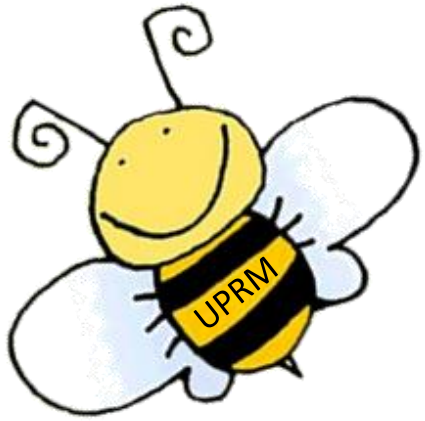
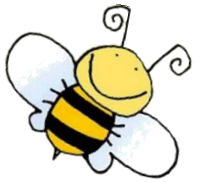


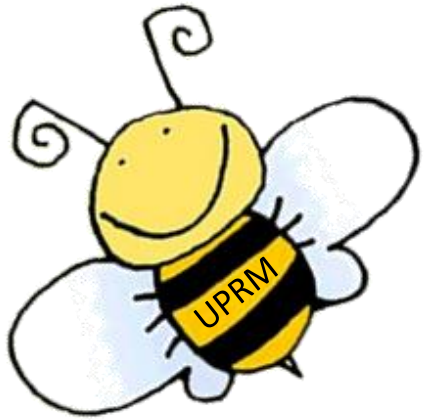
Built-in Matrix Generators

To cop with arrays that are used very frequently

READING ASSIGNMENT



Zeros Matrix




```
>> A=zeros(2)

A =

     0     0
     0     0
```

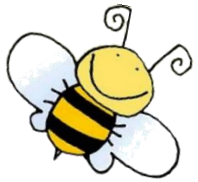
rows

If you specify one parameter,
it returns a square matrix of order 2

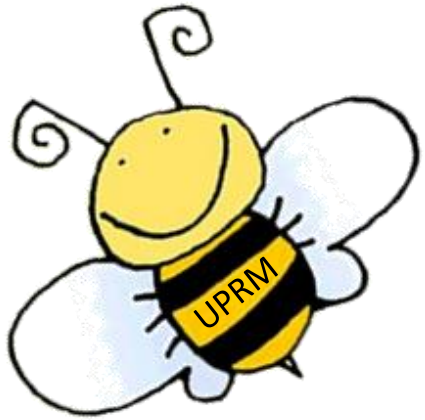
```
>> A=zeros(2,4)

A =

     0     0     0     0
     0     0     0     0
```

columns

If you specify 2 parameters,
in the example returns a 2 x 4 matrix



Ones Matrix

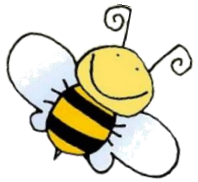



```
>> A=ones(3)

A =

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

```
>> A=ones(3,2)

A =

     1     1
     1     1
     1     1
```

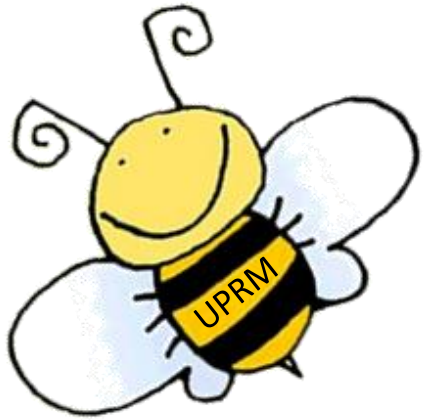
row

column



Similar syntax as zeros matrix

QY#1 {quiz yourself}

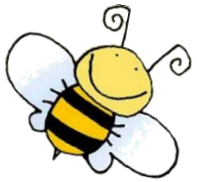


Which one of the following codes produce the output to the right?

(1)
```
A=ones(3);
B=zeros(3);
C=[A;B]
```

(2)
```
A=ones(3);
B=zeros(3);
C=[A,B]
```

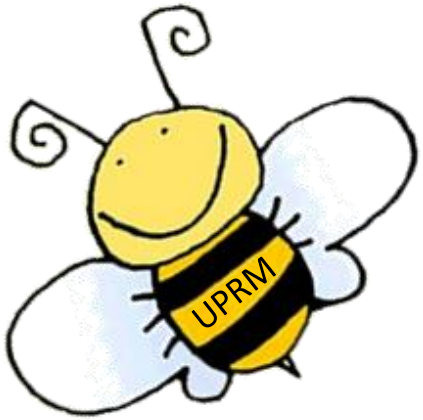
(3)
```
A=ones(3);
B=zeros(3);
C=A+B
```



Output:

```
C =
     1     1     1
     1     1     1
     1     1     1
     0     0     0
     0     0     0
     0     0     0
```

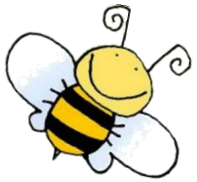
QY#2



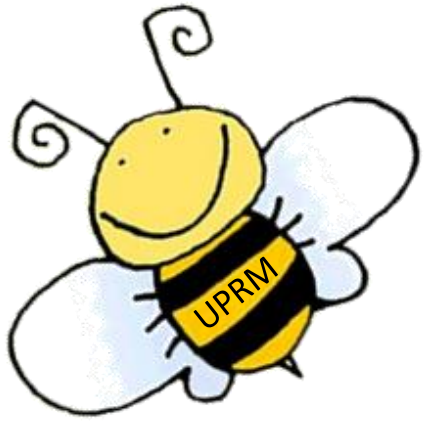

```
% Given
n=10;
A=ones(1,n+1)
```

<u>What is the output?</u>

A =



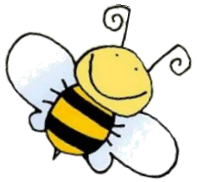
Random function



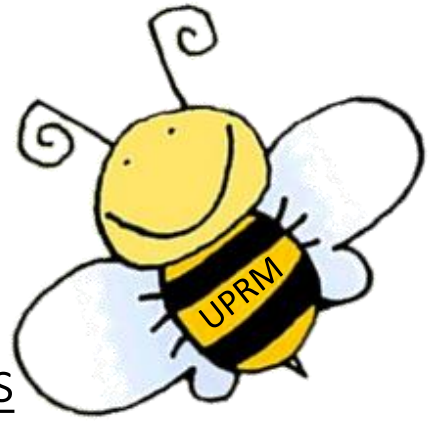
- Generates an array of pseudorandom numbers whose elements are distributed in the range [0,1]

A 2x3 matrix of random numbers:

```
>> A=rand(2,3)

A =

    0.9501    0.6068    0.8913
    0.2311    0.4860    0.7621
```



Random function



- Generates an array of pseudorandom numbers whose elements are distributed in the range [a,b]

A two-dice game of random numbers between [1,6]:

```
a=1;    % minimum
b=6;    % maximum

R=round((b-a).*rand(1,2)+a, 0)
```

scale to [1,6]

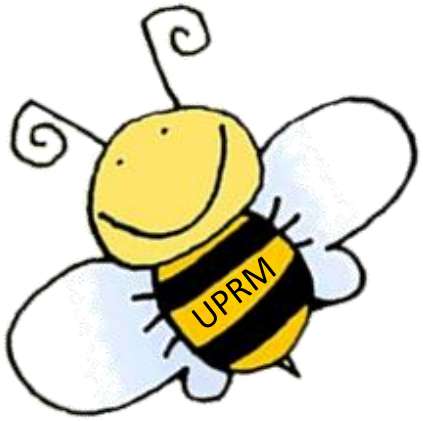
R =

Second dice

3 6 % 1 row, 2 columns

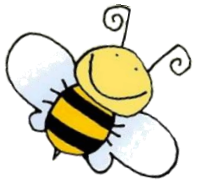
First dice

QY#3

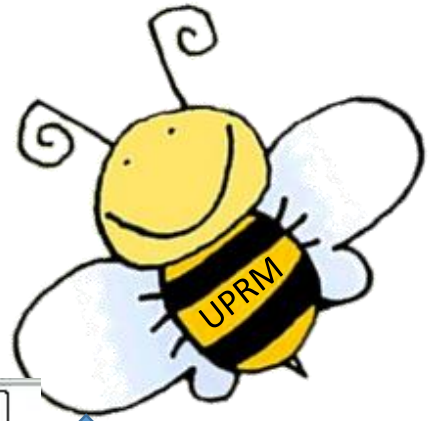


Write matlab code to:

1. Generate a 10-by-1 column vector of uniformly distributed random numbers in the interval (-5,5).
2. Generate random final exam scores for your N-student group
3. Randomly generate 13,000 GPA values to fake the GPAs of RUM students in the range [0.00,4.00]



QY#4

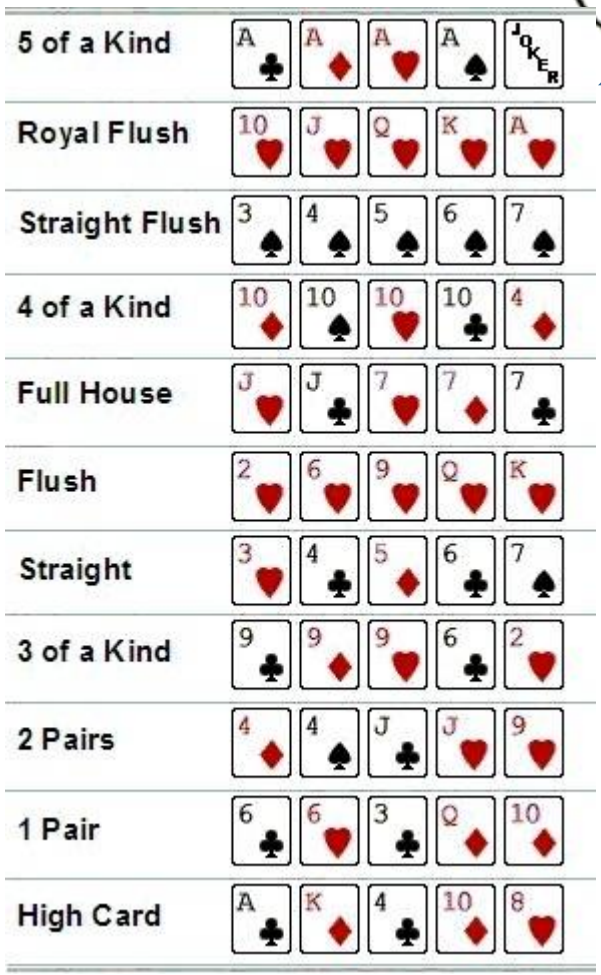


Construct a simplistic poker game

- 2 players
- Assume a player get a poker hand (five-cards) in one single shot by the random function in the range [1,11]. Each number represents a poker hand, e.g., 1 is a 5 of a kind, 2 is Royal Flush, etc.
- The code compare the hands of each player and decides who wins (i.e., compare 2 numbers).
- Player who gets the best hand ranking wins.

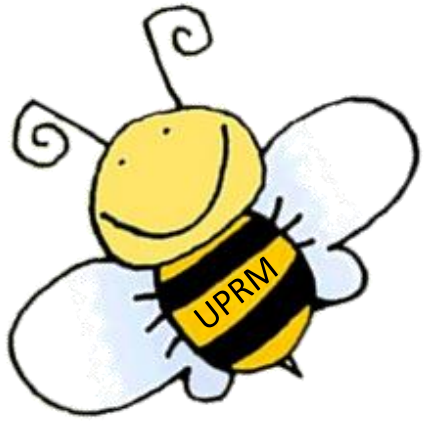
Construct a more complex poker game (under construction, design in progress; don't do it)

- A player card poker hand is a 2D array, rows is card type, column is card value which will be populated in only 5 positions. Initially the array is zero everywhere.
- First index is the suit type: [1]clubs (♣), [2] diamonds (♦), [3] hearts (♥) and [4] spades (♠). Use random function in the range [1,4] to get the suit.
- P1(i,j) is player 1 with card type i, and rank j, e.g., P1(2,3) cards is 3-diamods card. If P1(2,3)=1 player1 have it, if P1(2,3)=0 it doesn't.
- Each player gets 5 different cards (a poker hand),
- Construct one function for each hand that interprets each 5-card hand among 11 possibilities: 5 of kind, Royal Flush, etc.



Higher Ranking

Identity Matrix: **eye** function

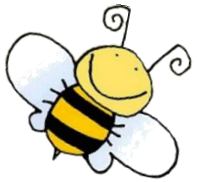



```
>> eye(3)
ans =

     1     0     0
     0     1     0
     0     0     1
```

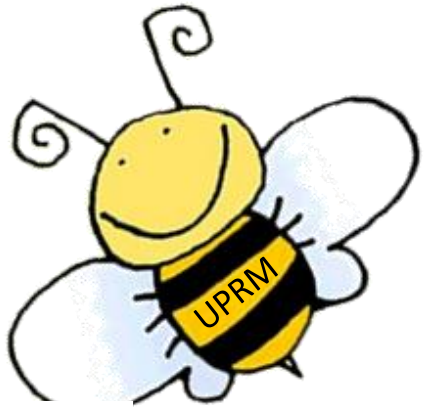
```
>> eye(4)
ans =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

```
>> eye(2,3)
ans =

     1     0     0
     0     1     0
```

```
>> eye(4,3)
ans =

     1     0     0
     0     1     0
     0     0     1
     0     0     0
```



QY#5



Which one of the following codes produce the output to the right?

(1) A=ones(3);

I=eye(3);

C=A*I

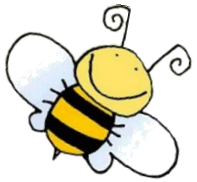
(2) A=zeros(3);

I=eye(3);

C=I-A

OUTPUT

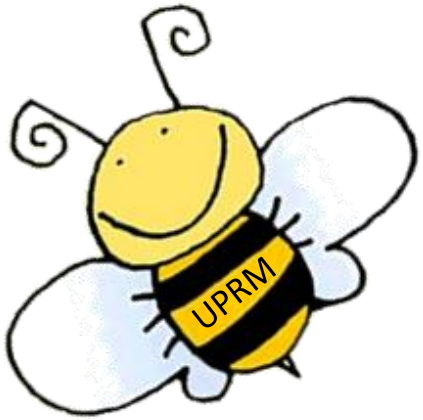
```
C =
     1     1     1
     1     1     1
     1     1     1
```



(3) I=eye(3);

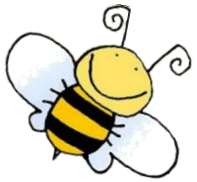
I=eye(3);

C=I*I

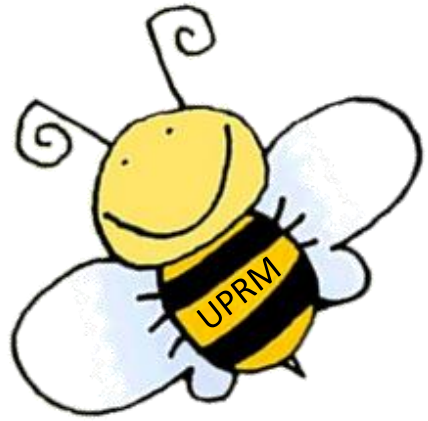


Useful Array Functions

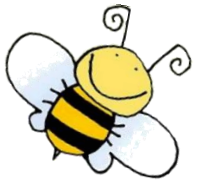
Better knowing their existence



Number of dimensions




```
>> A=[1,2;3,4;5,6]
A =

     1     2
     3     4
     5     6

>> ndims(A)
ans =

     2
```



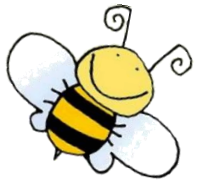
rows columns pages

```
>> B=ones(2,3,2)
B(:,:,1) =

     1     1     1
     1     1     1

B(:,:,2) =
     1     1     1
     1     1     1

>> ndims(B)
ans =
     3
```

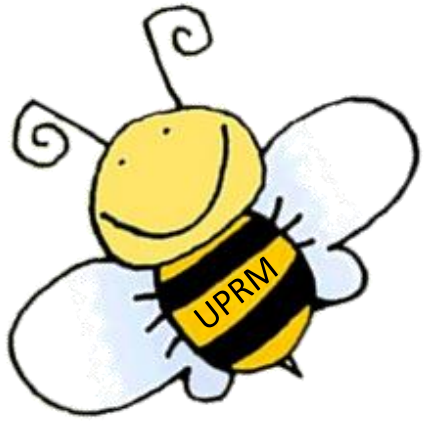
Size

Returns the length of each dimensions of its argument

```
>> A=[1,2;3,4;5,6]
A =

     1     2
     3     4
     5     6

>> size(A)
ans =

     3     2
```

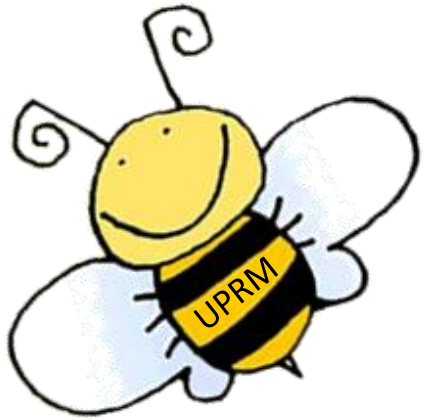


Creates a 4D array of zeros




```
>> B=zeros(2,3,2,4)
% two ways to call size:
>> size(B)
ans =
     2     3     2     4

>> [m,n,s,t]=size(B)
m =
     2
n =
     3
s =
     2
t =
     4
```

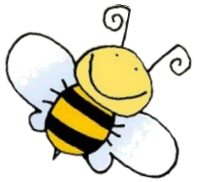
Length



- Returns the length of the largest dimension of an array

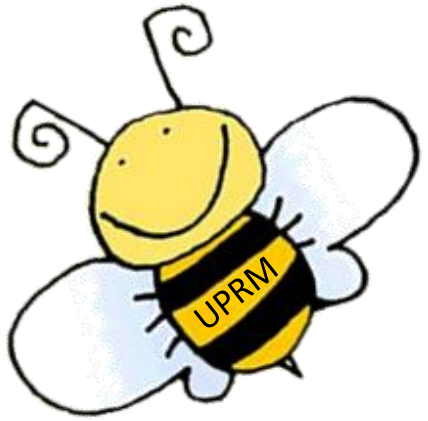
Array is 3x2:

```
>> A=[1 3; 2 4; 0 2]
A =

     1     3
     2     4
     0     2

>> length(A)
ans =

     3
```

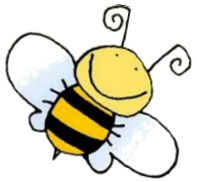


Number of Elements

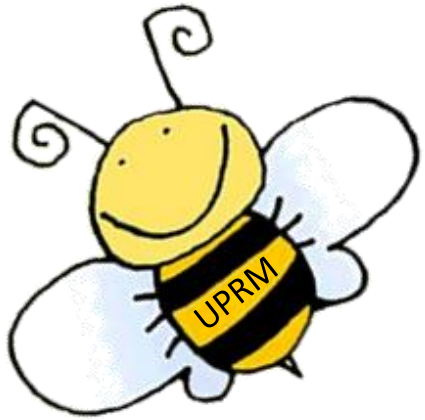
- Return the number of elements in an array




```
>> A=[1 3 5; 2 4 6; 0 2 4]
A =

     1     3     5
     2     4     6
     0     2     4

>> numel(A)
ans =
          9
```

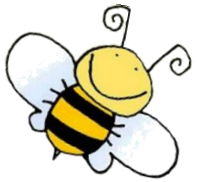


QY#6

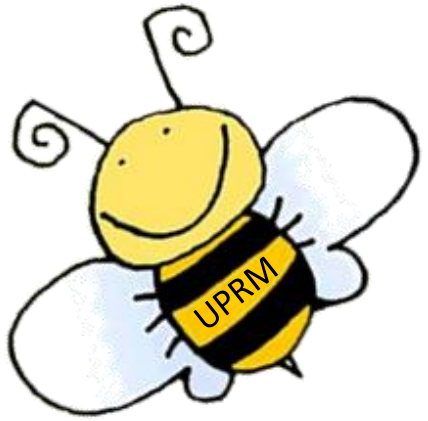


- What is the output?

```
% Numel
clc, clear
text1 ='El pueblo de Parangaricutirimicuaro se va a ';
text2 ='desparangaricutirimicuarizar. Quien logre ';
text3 ='desparangaricutirimicuarizarlo gran ';
text4 ='desparangaricutirimicuarizador sera';

text=[text1,text2,text3,text4];
numel(text)
```



Diagonal

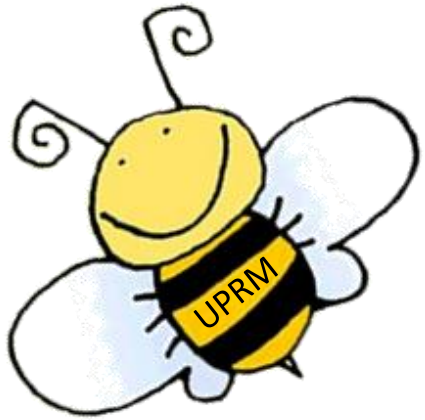


Returns the elements of the main diagonal

Elements with equal row and column indices: (1,1), (2,2), (3,3), etc.

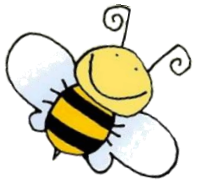
```
>> A=[1 3 5; 2 4 6; 0 2 4]
A =

     1     3     5
     2     4     6
     0     2     4

>> diag(A)
ans =

     1
     4
     4
```

Sort



If a vector, the sort is in ascending order

```
>> A=[4 2 3 9 1 2]
A =
    4    2    3    9    1    2

>> sort(A)
ans =
    1    2    2    3    4    9
```

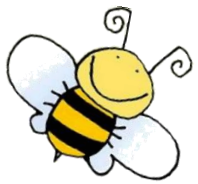


If a 2-D array, it sorts each column

```
>> A=[4 5 6; 7 8 9; 1 2 3]
A =
    4    5    6
    7    8    9
    1    2    3

>> sort(A)
ans =
    1    2    3
    4    5    6
    7    8    9
```

Sort

```
>> A=[4 6 5; 8 7 9; 1 3 2]
A =

    4    6    5
    8    7    9
    1    3    2
```

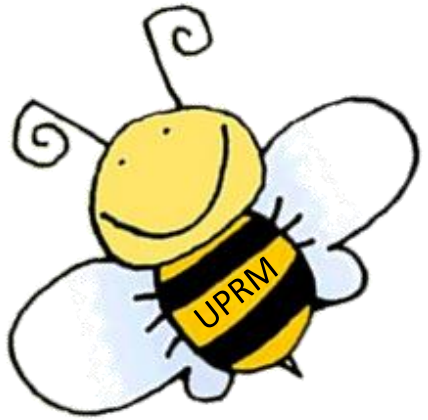


sort by column

```
>> sort(A,1)
ans =

    1    3    2
    4    6    5
    8    7    9
```

sort by row

```
>> sort(A,2)
ans =

    4    5    6
    7    8    9
    1    2    3
```



Sort

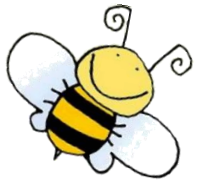
```
>> A=[4 6 5; 8 7 9; 1 3 2]
A =

    4    6    5
    8    7    9
    1    3    2
```

sort by column

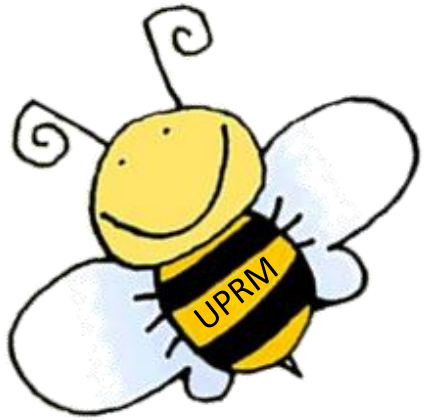
```
>> sort(A,1)
ans =

    1    3    2
    4    6    5
    8    7    9
```

sort by row

```
>> sort(A,2)
ans =

    4    5    6
    7    8    9
    1    2    3
```

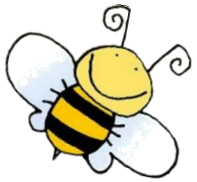


QY# 7. Sort both examples in descending order

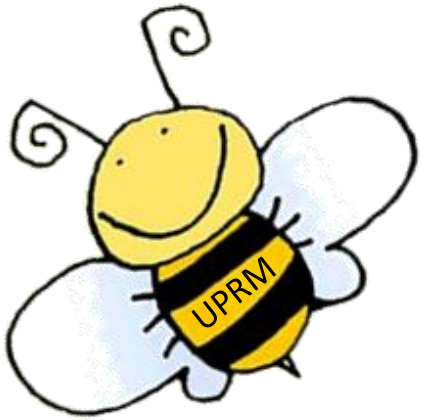


Linear Systems of Equations

Three ways
- Matrix Division
- Matrix Inverse
- Library function



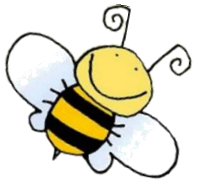
In general a system of **m** equations in **n** unknowns can be written as:



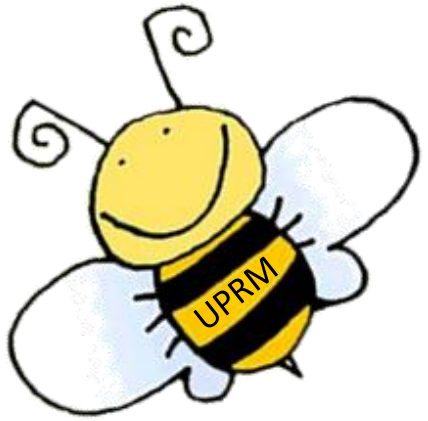
$$\begin{aligned} a_{11}x_1 &+ a_{12}x_2 &+ \ldots + a_{1n}x_n &= b_1 \\ a_{21}x_1 &+ a_{22}x_2 &+ \ldots + a_{2n}x_n &= b_2 \\ &\ldots \\ a_{m1}x_1 &+ a_{m2}x_2 &+ \ldots + a_{mn}x_n &= b_m \end{aligned} \quad \Longrightarrow \quad AX=B$$

In matrix form:

$$A = \begin{bmatrix} a_{11} & a_{12} \ldots & a_{1n} \\ a_{21} & a_{22} \ldots & a_{2n} \\ \ldots & & \\ a_{m1} & a_{m2} \ldots & a_{mn} \end{bmatrix} \qquad X = \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix} \qquad B = \begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_m \end{bmatrix}$$

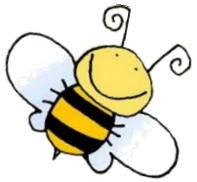


In general a system of **m** equations in **n** unknowns can be written as:



$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{2n}x_n &= b_2 \\ \ldots & \\ a_{m1}x_1 + a_{m2}x_2 \;\; a_{mn}x_n &= b_m \end{aligned} \quad \Longrightarrow \quad AX=B$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ \ldots & & \\ a_{m1} & a_{m2} & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_m \end{bmatrix}$$



The solution of the linear system:

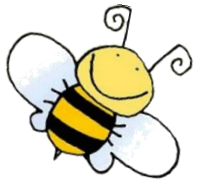
X=A\B (matrix left division)

X=B/A

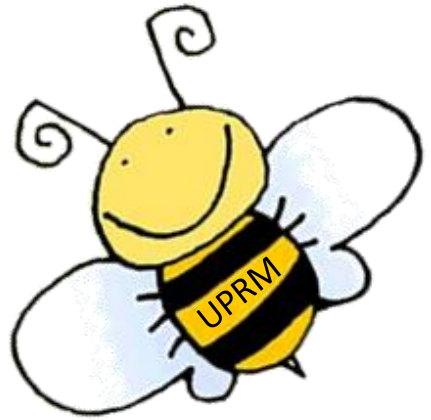
Example:

$$\begin{aligned} 3x_1 &+ 2x_2 &+ x_3 &= 5 \\ x_1 &+ 2x_2 &+ 3x_3 &= 13 \\ -5x_1 &- 10x_2 &- 5x_3 &= 0 \end{aligned} \Longrightarrow AX = B$$

$$\underbrace{\begin{bmatrix} 3 & 2 & 1 \\ 1 & 2 & 3 \\ -5 & -10 & -5 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{X} = \underbrace{\begin{bmatrix} 5 \\ 13 \\ 0 \end{bmatrix}}_{B}$$



X=A\B, the MATLAB solution

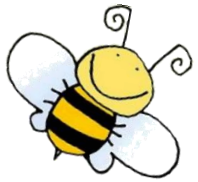



```
>> A=[3 2 1; 1 2 3; -5 -10 -5]
A =

     3     2     1
     1     2     3
    -5   -10    -5

>> B=[5;13;0]
B =

     5
    13
     0
```

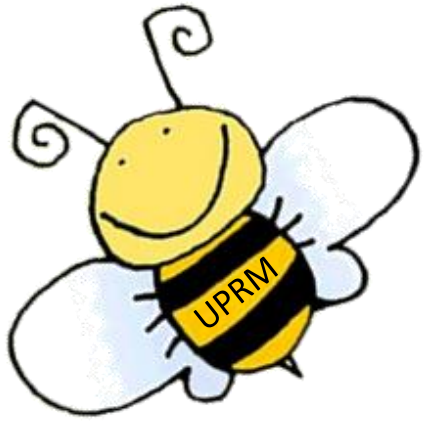
```
>> X=A\B
X =
    2.5000
   -4.5000
    6.5000
```

Verify the answer:

```
>> B= A*X   <E>

B =
     5
    13
     0
```



Matrix Inverse



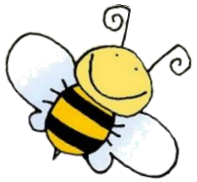
IF

- A is the coefficient matrix
- X is the solution vector
- m = n, A is square matrix

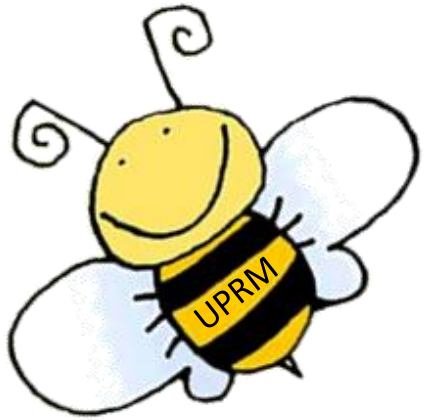
  i.e., number of rows equal the number of columns
- det A is **non-zero**,

Then

A^{-1} exist



Inverse



- Inverse is a square matrix such that

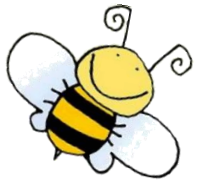
  $A^{-1}A = I$, the identity matrix

- The solution of the system is given by

  $A^{-1}AX = IX = X = A^{-1}B$

- If A is order 3, the identity matrix is also order 3:

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



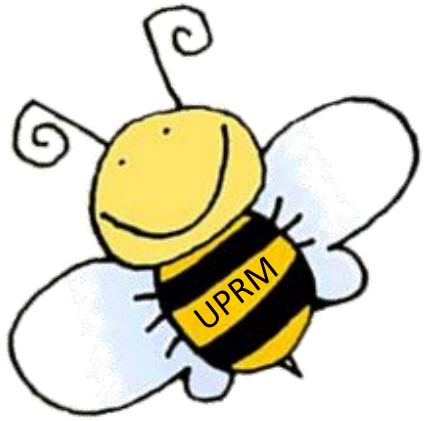
Example

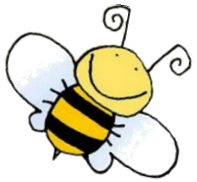
A system of 2 equations and 2 unknowns:

$2x_1 - x_2 = 2$

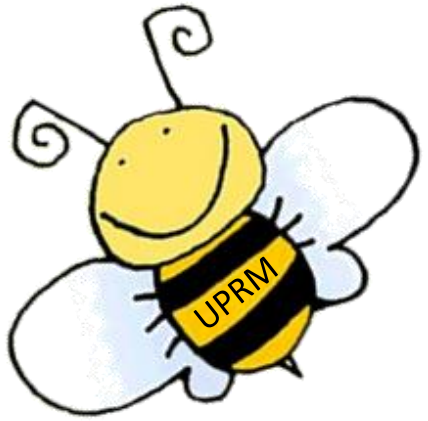
$x_1 + x_2 = 5$

```
>> A=[2 -1; 1 1]
A =
     2    -1
     1     1
>> B=[2;5]
B =
     2
     5
>> X=inv(A)*B
X =
    2.3333
    2.6667
```





TRES METODOS

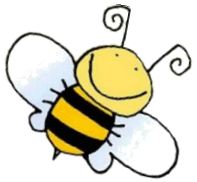



```
A =          ;
B =          ;
```

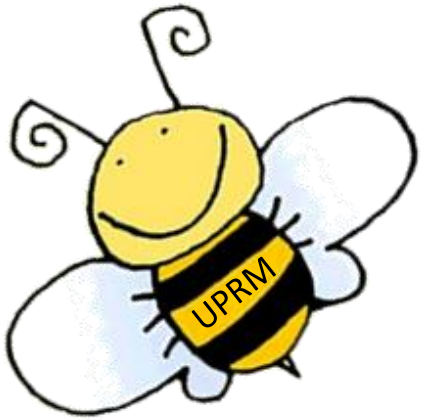
(1) `X = A\B;`

(2) `X = inv(A)*B;`

(3) `X = linsolve(A,B);`

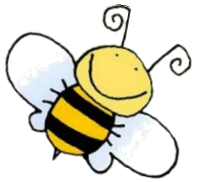


QY#8

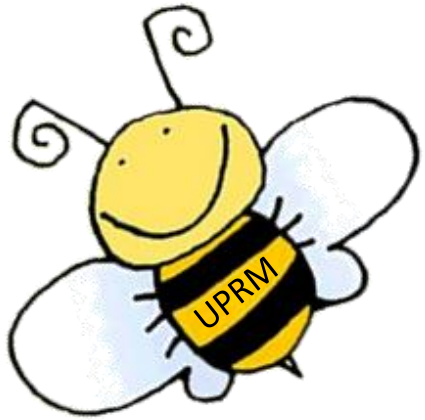


- Using three methods (1) Left division operator (2) Matrix inversion (3) Linspace function, solve the 5x5 system of equations. Submit code and output

$$
\begin{aligned}
4x_1 + 1x_2 + 2x_3 - 3x_4 + 5x_5 &= -16 \\
-3x_1 + 3x_2 - 1x_3 + 4x_4 - 2x_5 &= 20 \\
-1x_1 + 2x_2 + 5x_3 + 1x_4 + 3x_5 &= -4 \\
5x_1 + 4x_2 + 3x_3 - 1x_4 + 2x_5 &= -10 \\
1x_1 - 2x_2 + 3x_3 - 4x_4 + 5x_5 &= 3
\end{aligned}
$$

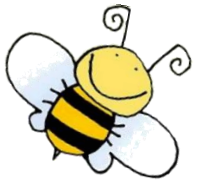


Determinant

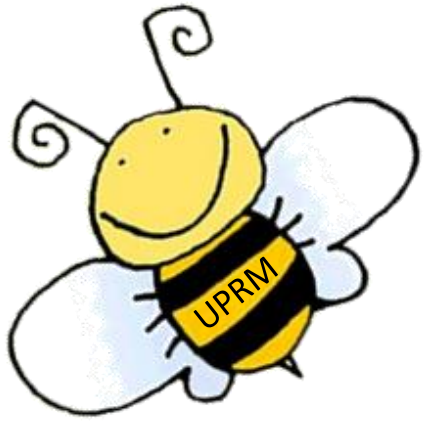


- Transformation of a square matrix that results in a scalar
- Determinant of A: |A| or det A
- If matrix has single entry:

  A=[3]➔ det A = 3



Determinant



Example with matrix of order 2:

$$\det\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = a_{11} \bullet a_{22} - a_{21} \bullet a_{12}$$

```
>> A=[2,3;6,4]
A =
     2     3
     6     4

>> det(A)
ans =
   -10
```

MATLAB instructions