05-Arrays Part 3

text: Chapter 2.8-2.10

ECEGR 101
Engineering Problem Solving with Matlab
Professor Henry Louie

Overview

- Adding Elements to Arrays
- Strings and Numbers

Adding Elements to Vectors

1. Assign values to new elements.

```
>> d = 5:-1:1
d =
    5    4    3    2    1
>>
>> d(6) = 7
d =
    5    4    3    2    1    7
>>
>> d(7:9) = [0 3 8]
d =
    5    4    3    2    1    7    0    3    8
```


Adding Elements to Vectors

2. Concatenate two or more vectors.

```
>> a = 1:3
a =
    1    2    3
>> b = [8 9 0 3]
b =
    8    9    0    3
>>
>> x = [a b]
x =
    1    2    3    8    9    0    3
```

```
>> c = (0:0.5:2)'
c =
         0
    0.5000
    1.0000
    1.5000
    2.0000
>>
>> d = [3; 5]
d =
    3
    5
```

```
>> m = [c; d; c]
m =
         0
    0.5000
    1.0000
    1.5000
    2.0000
    3.0000
    5.0000
         0
    0.5000
    1.0000
    1.5000
    2.0000
```

Adding Elements to Matrices

1. Assign values to new elements.

```
>> x = [1 2 3; 4 4 4]
x =
    1    2    3
    4    4    4
>>
>> x(:, 4) = [5 5]
x =
    1    2    3    5
    4    4    4    5
>>
>> x(3:4,:) = ones(2,4)
x =
    1    2    3    5
    4    4    4    5
    1    1    1    1
    1    1    1    1
```

```
>> x = eye(2)
x =
    1    0
    0    1
>>
>> x(5,5) = 20
x =
    1    0    0    0    0
    0    1    0    0    0
    0    0    0    0    0
    0    0    0    0    0
    0    0    0    0   20
```


Adding Elements to Matrices

2. Concatenate two or more matrices.

```
>> x1 = eye(2)
x1 =
    1    0
    0    1
>>
>> x2 = [2 2 2; 3 3 3]
x2 =
    2    2    2
    3    3    3
>>
>> x = [x1 x2]
x =
    1    0    2    2    2
    0    1    3    3    3
```

```
>> x = [x1 ;x2]
??? Error using ==> vertcat
All rows in the bracketed expression must have the same
number of columns.
```

Adding Elements to Matrices

2. Concatenate two or more matrices (cont.)

```
>> x2 = [2 2 2; 3 3 3]
x2 =
     2     2     2
     3     3     3
>>
>> x3 = 4*eye(3)
x3 =
     4     0     0
     0     4     0
     0     0     4
```

```
>> x = [x2; x3]
x =
     2     2     2
     3     3     3
     4     0     0
     0     4     0
     0     0     4
```

```
>> x = [x2 x3]
??? Error using ==> horzcat
All matrices on a row in the bracketed expression must have the
 same number of rows.
```


Exercise

Determine the contents of array a after the following statements are executed.

```
>> a = [1 2 3; 4 5 6; 7 8 9];
>> a([3 1],:) = a([1 3],:);
```

Exercise

Determine the contents of array a after the following statements are executed.

```
>> a = [1 2 3; 4 5 6; 7 8 9];
>> a([3 1],:) = a([1 3],:);
```

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

```
>> a([3 1],:) = a([1 3],:)
a =
     7     8     9
     4     5     6
     1     2     3
```


Exercise

- Determine the contents of array a after the following statements are executed.

```
>> a = eye(2);
>> b = [3 4];
>> a = [[a ; b; b] [b b]'];
```

Exercise

- Determine the contents of array a after the following statements are executed.

```
>> a = eye(2);
>> b = [3 4];
>> a = [[a ; b; b] [b b]'];
```

```
>> a = eye(2)
a =
    1    0
    0    1
>>
>> b = [3 4]
b =
    3    4
>>
>> a = [[a ; b; b] [b b]']
a =
    1    0    3
    0    1    4
    3    4    3
    3    4    4
```


Exercise

```
>> a = [[a ; b; b] [b b]']
a =
    1    0    3
    0    1    4
    3    4    3
    3    4    4
```

Exercise

```
>> a = [[a ; b; b] [b b]']
a =
     1     0     3
     0     1     4
     3     4     3
     3     4     4
```


Exercise

```
>> a = [[a ; b; b] [b b]']
a =
     1     0     3
     0     1     4
     3     4     3
     3     4     4
```

Exercise

```
>> a = [[a ; b; b] [b b]']
a =
     1     0     3
     0     1     4
     3     4     3
     3     4     4
```


Exercise

```
>> a = [[a ; b; b] [b b]']
a =
     1     0     3
     0     1     4
     3     4     3
     3     4     4
```

Exercise

Create the following matrix. Use as few commands as possible.

```
z =
     1     0     0     2     0     0     0   -10
     0     1     0     0     4     0     0    -5
     0     0     1     0     0     6     0     0
     1     1     1     0     0     0     8     5
     1     0     1     0     0     1     1    10
     1     1     1     0     0     1     1    15
```


Exercise

```
z =
     1     0     0     2     0     0     0   -10
     0     1     0     0     4     0     0    -5
     0     0     1     0     0     6     0     0
     1     1     1     0     0     0     8     5
     1     0     1     0     0     1     1    10
     1     1     1     0     0     1     1    15
```

```
>> z=zeros(6,8)
z =
     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
```

Exercise

```
z =
     1     0     0     2     0     0     0   -10
     0     1     0     0     4     0     0    -5
     0     0     1     0     0     6     0     0
     1     1     1     0     0     0     8     5
     1     0     1     0     0     1     1    10
     1     1     1     0     0     1     1    15
```

```
>> z(1:3,1:3)=eye(3)
z =
     1     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0
     0     0     1     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
```


Exercise

```
z =
     1     0     0     2     0     0     0   -10
     0     1     0     0     4     0     0    -5
     0     0     1     0     0     6     0     0
     1     1     1     0     0     0     8     5
     1     0     1     0     0     1     1    10
     1     1     1     0     0     1     1    15
```

```
>> z(4:6,1:3)=ones(3,3)
z =
     1     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0
     0     0     1     0     0     0     0     0
     1     1     1     0     0     0     0     0
     1     1     1     0     0     0     0     0
     1     1     1     0     0     0     0     0
```

Exercise

```
z =
     1     0     0     2     0     0     0   -10
     0     1     0     0     4     0     0    -5
     0     0     1     0     0     6     0     0
     1     1     1     0     0     0     8     5
     1     0     1     0     0     1     1    10
     1     1     1     0     0     1     1    15
```

```
>> z(5,2)=0
z =
     1     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0
     0     0     1     0     0     0     0     0
     1     1     1     0     0     0     0     0
     1     0     1     0     0     0     0     0
     1     1     1     0     0     0     0     0
```


Exercise

```
z =
     1     0     0     2     0     0     0   -10
     0     1     0     0     4     0     0    -5
     0     0     1     0     0     6     0     0
     1     1     1     0     0     0     8     5
     1     0     1     0     0     1     1    10
     1     1     1     0     0     1     1    15
```

```
>> z(5:6,6:7)=ones(2,2)
z =
     1     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0
     0     0     1     0     0     0     0     0
     1     1     1     0     0     0     0     0
     1     0     1     0     0     1     1     0
     1     1     1     0     0     1     1     0
```

Exercise

```
z =
    1    0    0    2    0    0    0  -10
    0    1    0    0    4    0    0   -5
    0    0    1    0    0    6    0    0
    1    1    1    0    0    0    8    5
    1    0    1    0    0    1    1   10
    1    1    1    0    0    1    1   15
```

```
>> z(1:4,4:7) = diag(2:2:8)
z =
     1     0     0     2     0     0     0     0
     0     1     0     0     4     0     0     0
     0     0     1     0     0     6     0     0
     1     1     1     0     0     0     8     0
     1     0     1     0     0     1     1     0
     1     1     1     0     0     1     1     0
```


Exercise

```
z =
    1    0    0    2    0    0    0  -10
    0    1    0    0    4    0    0   -5
    0    0    1    0    0    6    0    0
    1    1    1    0    0    0    8    5
    1    0    1    0    0    1    1   10
    1    1    1    0    0    1    1   15
```

```
>> z(:,8)=[-10:5:15]'
z =
     1     0     0     2     0     0     0   -10
     0     1     0     0     4     0     0    -5
     0     0     1     0     0     6     0     0
     1     1     1     0     0     0     8     5
     1     0     1     0     0     1     1    10
     1     1     1     0     0     1     1    15
```

Exercise – Different Solution

```
>> z = [ [eye(3); ones(3)]  [diag(2:2:8); [zeros(2) ones(2)]] [-10:5:15]']
z =
   1   0   0   2   0   0   0  -10
   0   1   0   0   4   0   0   -5
   0   0   1   0   0   6   0    0
   1   1   1   0   0   0   8    5
   1   1   1   0   0   1   1   10
   1   1   1   0   0   1   1   15

>> z(5,2) = 0
z =
   1   0   0   2   0   0   0  -10
   0   1   0   0   4   0   0   -5
   0   0   1   0   0   6   0    0
   1   1   1   0   0   0   8    5
   1   0   1   0   0   1   1   10
   1   1   1   0   0   1   1   15
```


Deleting Matrix Elements

- Reassign nothing ([]) to the matrix elements.

```
>> x = 1:5
x =
     1     2     3     4     5
>>
>> x(5) = []
x =
     1     2     3     4
>>
>> x(1:2) = []
x =
     3     4
```

```
>> y = [2 4 5; 5 7 8; 0 0 3; 4 4 5]
y =
     2     4     5
     5     7     8
     0     0     3
     4     4     5
>>
>> y(:, 1) = []
y =
     4     5
     7     8
     0     3
     4     5
>>
>> y(2:3, :) = []
y =
     4     5
     4     5
```

Exercise

- Determine the contents of array x after the following statements are executed.

```
>> x = [1 3 4 8 9; 2 4 4 8 9; 0 5 7 8 3; 3 3 2 2 5];
>> x(2:3, :) = [];
>> x(:,3) = [];
```


Exercise

```
>> x = [1 3 4 8 9; 2 4 4 8 9; 0 5 7 8 3; 3 3 2 2 5]
x =
    1    3    4    8    9
    2    4    4    8    9
    0    5    7    8    3
    3    3    2    2    5
>>
>> x(2:3, :) = []
x =
    1    3    4    8    9
    3    3    2    2    5
>>
>> x(:,3) = []
x =
    1    3    8    9
    3    3    2    5
```

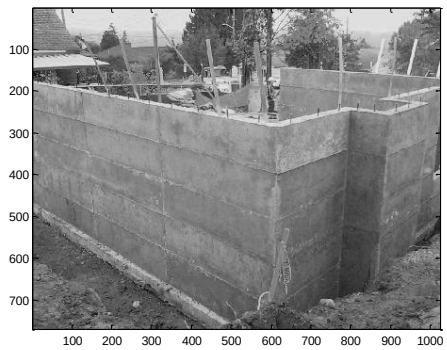
Flipping Arrays

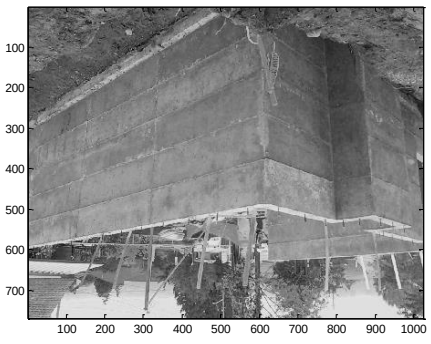
- Arrays can be "flipped"
- fliplr(x): preserves row ordering but flips columns around the middle
- flipud(x): preserves column ordering but flips rows around the center

```
>> A=[1 2 3; 4 5 6; 7 8 9]
A =
     1     2     3
     4     5     6
     7     8     9
>> fliplr(A)
ans =
     3     2     1
     6     5     4
     9     8     7
>> flipud(A)
ans =
     7     8     9
     4     5     6
     1     2     3
```


Example

```
>> picture=imread('foundation.jpg');
>> picture(:,:,1)=flipud(picture(:,:,1));
>> picture(:,:,2)=flipud(picture(:,:,2));
>> picture(:,:,3)=flipud(picture(:,:,3));
>> image(picture)
```





Strings in MATLAB

- String = array of characters created by typing the characters within single quotes.

```
>> name = 'Agnieszka'
name =
Agnieszka
>>
>> name(1)
ans =
A
>>
>> name(6:7)
ans =
sz
```

```
>> first_name = 'Agnieszka'
first_name =
Agnieszka
>>
>> last_name = 'Miguel'
last_name =
Miguel
>>
>> name = [first_name '  ' last_name]
name =
Agnieszka  Miguel
```


Strings vs. Numbers

String:

```
>> x = '2006'
x =
2006
>>
>> class(x)
ans =
char
```

```
>> x + 10
ans =
   60    58    58    64
```

Number:

```
>> x = 2006
x =
        2006
>>
>> class(x)
ans =
double
```

```
>> x + 10
ans =
        2016
```

Strings vs. numbers

```
>> x + 0
ans =
    50    48    48    54
```

ASCII (***A****merican* ***S****tandard* ***C****ode for* ***I****nformation* ***I****nterchange*), is a character encoding based on the English alphabet. ASCII codes represent text in computers, communications equipment, and other devices that work with text.

50 is the ASCII code for 2
48 is the ASCII code for 0
54 is the ASCII code for 6

ASCII code

This is part of the ASCII table:

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
0	0	NUL	16	10	DLE	32	20	(space)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

Strings vs. Numbers

To convert string to a number use the str2num function.

```
>> x = str2num('89')
x =
    89
>> class(x)
ans =
double
```


Strings vs. Numbers

To convert a number to a string use the num2str function.

```
>> s = num2str(123.77)
s =
123.77
>> class(s)
ans =
char
>> length(s)
ans =
     6
>> s(4)
ans =
.
```

Strings in Vectors

- Strings can be included in row or column vectors

```
>> x=num2str(5)

x =

5

>> ['I would like ' x ' tacos']

ans =

I would like 5 tacos
```

```
>> ['I would like ' x ' tacos']'

ans =

I

w
o
u
l
d

l
i
k
e

5

t
a
c
o
s
```


Strings in Matrices

Matrices must have the same number of characters in each row. Use function char to build a matrix from strings of different length.

```
>> name = 'Puzzle';
>> breed = 'australian shepherd';
>> age = '4 years';
>>
>> dog_info = char(name, breed, age)
dog_info =
Puzzle
australian shepherd
4 years
```

```
>> dog_info(2, :)
ans =
australian shepherd
>>
>> dog_info(3,1)
ans =
4
```