

- Previous Lecture:
  - Discrete vs. continuous; finite vs. infinite
  - Vectorized operations
- Today's Lecture:
  - Vectorized operations and plots
  - 2-d array—matrix
- Announcements:
  - Prelim I tonight at 7:30pm
    - Last names A-H in Olin Hall 255
    - Last names I-L in Olin Hall 245
    - Last names M-Z in Olin Hall 155
  - Fall Break Mon & Tues: no lec, dis, office/consulting hrs.  
 Attendance at 10/10 (W) dis is optional, but the exercise is required. Attend any of the 10/10 dis sections for help if you wish.

Initialize vectors/matrices if dimensions are known  
 ...instead of "building" the array one component at a time

```

% Initialize y
x=linspace(a,b,n);
y=zeros(1,n);
for k=1:n
    y(k)=myF(x);
end
    
```

```

% Build y on the fly
x=linspace(a,b,n);

for k=1:n
    y(k)=myF(x);
end
    
```

↑  
Much faster for large n!

Lecture 13 19

**Vectorized code**  
 —a Matlab-specific feature See Sec 4.1 for list of vectorized arithmetic operations

- Code that performs element-by-element arithmetic/relational/logical operations on array operands in one step
- Scalar operation:  $x + y$   
 where  $x, y$  are scalar variables
- **Vectorized code:**  $x + y$   
 where  $x$  and/or  $y$  are vectors. If  $x$  and  $y$  are both vectors, they must be of the **same shape and length**

Lecture 13 21

**Vectorized addition**

	<b>x</b>	2	1	.5	8
+	<b>y</b>	1	2	0	1
=	<b>z</b>	3	3	.5	9

Matlab code:  $z = x + y$

Lecture 13 22

**Vectorized multiplication**

	<b>a</b>	2	1	.5	8
x	<b>b</b>	1	2	0	1
=	<b>c</b>	2	2	0	8

Matlab code:  $c = a .* b$

↑

Lecture 13 24

**Vectorized element-by-element arithmetic operations on arrays** See full list of ops in §4.1

A dot (.) is necessary in front of these math operators

Lecture 13 25

### Shift

	<b>x</b>	3
+	<b>y</b>	2 1 .5 8
-----		
=	<b>z</b>	5 4 3.5 11

Matlab code: `z = x + y`

Lecture 13 26

### Reciprocate

	<b>x</b>	1
/	<b>y</b>	2 1 .5 8
-----		
=	<b>z</b>	.5 1 2 .125

Matlab code: `z = x ./ y`

↑

Lecture 13 27

### Vectorized

See full list of ops in §4.1

element-by-element arithmetic operations between an array and a scalar

+

-

\*

/

A dot (.) is necessary in front of these math operators

The dot in `.*`, `./`, `.*`, `./` not necessary but OK

Lecture 13 28

### Generating tables and plots

x, y are vectors. A vector is a 1-dimensional list of values

x	sin(x)
0.000	0.000
0.784	0.707
1.571	1.000
2.357	0.707
3.142	0.000
3.927	-0.707
4.712	-1.000
5.498	-0.707
6.283	0.000

```

x= linspace(0,2*pi,9);
y= sin(x);
plot(x,y)
    
```

Note: x, y are shown in columns due to space limitation; they should be rows.

Lecture 13 29

### Built-in function linspace

```

x= linspace(1,3,5)
    
```

x	1.0	1.5	2.0	2.5	3.0
---	-----	-----	-----	-----	-----

```

x= linspace(0,1,101)
    
```

x	0.00	0.01	0.02	...	0.99	1.00
---	------	------	------	-----	------	------

← Left endpoint     
 ← Right endpoint     
 ← Number of points

Lecture 13 33

### How did we get all the sine values?

Built-in functions accept arrays

0.00	1.57	3.14	4.71	6.28
------	------	------	------	------

and return arrays

0.00	1.00	0.00	-1.00	0.00
------	------	------	-------	------

Lecture 13 34

Can we plot this? See plotComparison.m

$$f(x) = \frac{\sin(5x)\exp(-x/2)}{1+x^2} \quad \text{for } -2 \leq x \leq 3$$

Yes!

```
x = linspace(-2,3,200);
y = sin(5*x).*exp(-x/2)./(1 + x.^2);
plot(x,y)
```

↑                    ↑                    ↑  
Element-by-element arithmetic operations on arrays

Lecture 13                    39

Element-by-element arithmetic operations on arrays... Also called "vectorized code"

**x and y are vectors**

```
x = linspace(-2,3,200);
y = sin(5*x).*exp(-x/2)./(1 + x.^2);
```

Contrast with scalar operations that we've used previously...

```
a = 2.1;
b = sin(5*a);
```

**a and b are scalars**

The operators are (mostly) the same; the operands may be scalars or vectors.

When an operand is a vector, you have "vectorized code."

Lecture 13                    40

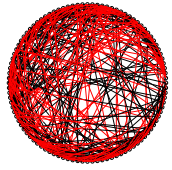
Storing and using data in tables

A company has 3 factories that make 5 products with these costs:

10	36	22	15	62
12	35	20	12	66
13	37	21	16	59

C

What is the best way to fill a given purchase order?

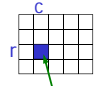


Connections between webpages

0	0	1	0	1	0	0
1	0	0	1	1	1	0
0	1	0	1	1	1	1
1	0	1	1	0	1	0
0	0	1	1	0	1	1
0	0	1	0	1	0	1
0	1	1	0	1	1	0

Lecture 13                    42

2-d array: **matrix**



- An array is a **named** collection of **like** data organized into rows and columns
- A 2-d array is a table, called a **matrix**
- Two **indices** identify the position of a value in a matrix, e.g.,
 

```
mat(r,c)
```

 refers to component in row **r**, column **c** of matrix **mat**
- Array index starts at **1**
- Rectangular**: all rows have the same #of columns

Lecture 13                    43

Creating a matrix

- Built-in functions: **ones**, **zeros**, **rand**
  - E.g., `zeros(2,3)` gives a 2-by-3 matrix of 0s
- "Build" a matrix using square brackets, `[ ]`, but the dimension must match up:
  - `[x y]` puts **y** to the right of **x**
  - `[x; y]` puts **y** below **x**
  - `[4 0 3; 5 1 9]` creates the matrix 

4	0	3
5	1	9
  - `[4 0 3; ones(1,3)]` gives 

4	0	3
1	1	1
  - `[4 0 3; ones(3,1)]` doesn't work

Lecture 13                    44

Working with a matrix:

2	-1	.5	0	-3
3	8	6	7	7
5	-3	8.5	9	10
52	81	.5	7	2

size and individual components

Given a matrix **M**

```
[nr, nc]= size(M) % nr is #of rows,
                  % nc is #of columns
nr= size(M, 1) % # of rows
nc= size(M, 2) % # of columns
```

```
M(2,4)= 1;
disp(M(3,1))
M(1,nc)= 4;
```

Lecture 13                    45

```
% What will M be?
M = [ones(1,3); 1:4]
```

**A**    1   1   1   0  
       1   2   3   4

**B**    1   1   1  
       1   2   3

**C**    Error – M not created

Lecture 13    46

What will **A** be?


```
A= [0 0]
A= [A' ones(2,1)]
A= [0 0 0 0; A A]
```

Lecture 13    48

Example: minimum value in a matrix

```
function val = minInMatrix(M)
```

% val is the smallest value in matrix M



Lecture 13    49

Pattern for traversing a matrix M

```
[nr, nc] = size(M)
for r= 1:nr
    % At row r
    for c= 1:nc
        % At column c (in row r)
        %
        % Do something with M(r,c) ...
    end
end
```

Lecture 13    51

Matrix example: Random Web

- N web pages can be represented by an N-by-N Link Array **A**.
- **A(i,j)** is 1 if there is a link on webpage **j** to webpage **i**
- Generate a random link array and display the connectivity:
  - There is no link from a page to itself
  - If  $i \neq j$  then  $A(i,j) = 1$  with probability  $\frac{1}{1+|i-j|}$
 ⇨ There is more likely to be a link if **i** is close to **j**

Lecture 13    52

```
function A = RandomLinks(n)
% A is n-by-n matrix of 1s and 0s
% representing n webpages

A = zeros(n,n);
for i=1:n
    for j=1:n
        r = rand(1);
        if i~j && r<= 1/(1 + abs(i-j));
            A(i,j) = 1;
        end
    end
end
```

Lecture 13    53

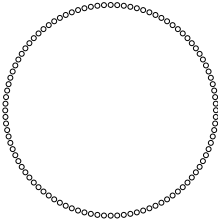
Random web  
N = 20

```

01110000010010000000
10001000111000000100
01010000000000000000
00101000000000000000
00010000001100000000
00000000000001010000
01111100010110000000
00000010000100000011
01000000010010001000
00000001101000000001
00000010000011000000
00000010010000000001
00010000110101100000
00000010000000110000
0000101000010010001
00000010001000001010
01000000100001010110
000000000000000011001
00000010000000000000
00000000000000001010
    
```

Lecture 13 54

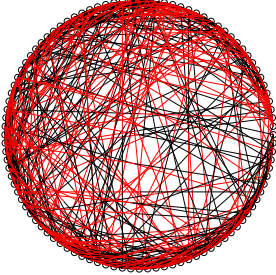
Represent the web pages graphically...



100 Web pages arranged in a circle.  
Next display the links....

Lecture 13 55

Represent the web pages graphically...



Line black as it leaves page j, red when it arrives at page i

Lecture 13 56

ShowRandomLinks.m

Lecture 13 57

```

% Given an nr-by-nc matrix M.
% What is A?
for r= 1: nr
    for c= 1: nc
        A(c,r)= M(r,c);
    end
end
    
```

- A A is M with the columns in reverse order
- B A is M with the rows in reverse order
- C A is the transpose of M
- D A and M are the same

Lecture 13 58

```

% Given an n-by-m matrix A.
% What is this operation?
for g= 1: n
    for h= 1: floor(m/2)
        A(g,h)= A(g, m-h+1);
    end
end
    
```

- A Reflect the right half of A onto the left half
- B Reflect the bottom half of A onto the top half

Lecture 13 60