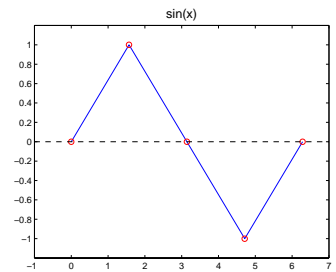


- Previous class:
 - User-defined function
 - Nested loops
- Now:
 - Working with colors
 - 1-dimensional array—vector
 - Play with sound files

Plot a continuous function (from a table of values)

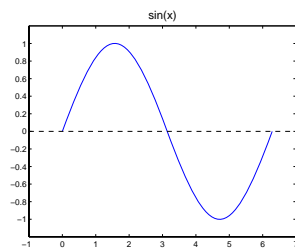
x	sin(x)
0.00	0.0
1.57	1.0
3.14	0.0
4.71	-1.0
6.28	0.0



Plot based on 5 points

2

Plot based on 200 discrete points, but it looks smooth



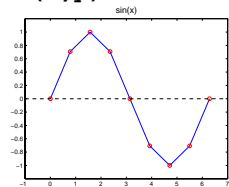
3

Generating tables and plots

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

```
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.

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

5

How did we get all the sine values?

Built-in functions accept arrays

[0.00 1.57 3.14 4.71 6.28]



sin

and return arrays

[0.00 1.00 0.00 -1.00 0.00]

x	sin(x)
0.00	0.0
1.57	1.0
3.14	0.0
4.71	-1.0
6.28	0.0

6

Vectorized addition

$$\begin{array}{r}
 \mathbf{x} \quad \begin{bmatrix} 2 & 1 & .5 & 8 \end{bmatrix} \\
 + \quad \mathbf{y} \quad \begin{bmatrix} 1 & 2 & 0 & 1 \end{bmatrix} \\
 \hline
 = \quad \mathbf{z} \quad \begin{bmatrix} 3 & 3 & .5 & 9 \end{bmatrix}
 \end{array}$$

Matlab code: `z = x + y`

7

Vectorized subtraction

$$\begin{array}{r}
 \mathbf{x} \quad \begin{bmatrix} 2 & 1 & .5 & 8 \end{bmatrix} \\
 - \quad \mathbf{y} \quad \begin{bmatrix} 1 & 2 & 0 & 1 \end{bmatrix} \\
 \hline
 = \quad \mathbf{z} \quad \begin{bmatrix} 1 & -1 & .5 & 7 \end{bmatrix}
 \end{array}$$

Matlab code: `z = x - y`

8

Vectorized multiplication

$$\begin{array}{r}
 \mathbf{a} \quad \begin{bmatrix} 2 & 1 & .5 & 8 \end{bmatrix} \\
 \times \quad \mathbf{b} \quad \begin{bmatrix} 1 & 2 & 0 & 1 \end{bmatrix} \\
 \hline
 = \quad \mathbf{c} \quad \begin{bmatrix} 2 & 2 & 0 & 8 \end{bmatrix}
 \end{array}$$

Matlab code: `c = a .* b`



9

Vectorized element-by-element arithmetic operations on arrays

$$\begin{array}{l}
 \text{array} + \text{array} \longrightarrow \text{array} \\
 \text{array} - \text{array} \longrightarrow \text{array}
 \end{array}$$

$$\begin{array}{l}
 \text{array} .* \text{array} \longrightarrow \text{array} \\
 \text{array} ./ \text{array} \longrightarrow \text{array} \\
 \text{array} .^ \text{array} \longrightarrow \text{array}
 \end{array}$$

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

10

Shift

$$\begin{array}{r}
 \mathbf{x} \quad \begin{bmatrix} 3 \end{bmatrix} \\
 + \quad \mathbf{y} \quad \begin{bmatrix} 2 & 1 & .5 & 8 \end{bmatrix} \\
 \hline
 = \quad \mathbf{z} \quad \begin{bmatrix} 5 & 4 & 3.5 & 11 \end{bmatrix}
 \end{array}$$

Matlab code: `z = x + y`

11

Reciprocate

$$\begin{array}{r}
 \mathbf{x} \quad \begin{bmatrix} 1 \end{bmatrix} \\
 / \quad \mathbf{y} \quad \begin{bmatrix} 2 & 1 & .5 & 8 \end{bmatrix} \\
 \hline
 = \quad \mathbf{z} \quad \begin{bmatrix} .5 & 1 & 2 & .125 \end{bmatrix}
 \end{array}$$

Matlab code: `z = x ./ y`



12

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

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

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

13

Color is a 3-vector, sometimes called the RGB values

- Any color is a mix of red, green, and blue
- Example:
`color = [0.4 0.6 0]`
- Each component is a real value in [0,1]
- [0 0 0] is black
- [1 1 1] is white

14

Let's show the "paint chips" from white to black

Name the script **white2black**

15

Mix two colors

Implement this function:

```
function newc = mixEqual(c1,c2)
% Average colors c1 and c2.
% c1, c2, and newc are vectors
% representing colors.
% Display the three colors.
```

16

1-d array: vector

- An array is a named collection of like data organized into rows or columns
- A 1-d array is a row or a column, called a **vector**
- An **index** identifies the position of a value in a vector

score

93	92	87	0	90	82
1	2	3	4	5	6

17

Array index starts at 1

Let **k** be the index of vector **x**, then

- k** must be a positive integer
- $1 \leq k \leq \text{length}(x)$
- To access the **kth** element: `x(k)`

18

Accessing values in a vector

score	93	92	87	0	90	82
	1	2	3	4	5	6

Given the vector **score** ...

19

Accessing values in a vector

score	93	99	87	80	85	82
	1	2	3	4	5	6

Given the vector **score** ...

```
score(4)= 80;
score(5)= (score(4)+score(5))/2;
k= 1;
score(k+1)= 99;
```

20

A few different ways to create a vector
(More later!)

```
count= zeros(1,6)    count
```

0	0	0	0	0	0
---	---	---	---	---	---

```
x= linspace(10,30,5)    x
```

10	15	20	25	30
----	----	----	----	----

```
y= [3 7 2 1]
```

3	7	2	1
---	---	---	---

```
z= [3; 7; 2]
```

3
7
2

21

Drawing a single line segment

```
a= 0; % x-coord of pt 1
b= 1; % y-coord of pt 1
c= 5; % x-coord of pt 2
d= 3; % y-coord of pt 2
plot([a c], [b d], '-*')
```

x-values
(a vector)

y-values
(a vector)

Line/marker
format

22

Drawing a polygon (multiple line segments)

```
% Draw a rectangle with the lower-left
% corner at (a,b), width w, height h.
x= [a a+w a+w a a]; % x data
y= [b b b+h b+h b]; % y data
plot(x, y)
```

24

Coloring a polygon (fill)

```
% Draw a rectangle with the lower-left
% corner at (a,b), width w, height h,
% and fill it with a color named by c.
x= [a a+w a+w a a]; % x data
y= [b b b+h b+h b]; % y data
fill(x, y, c)
```

A built-in function

25

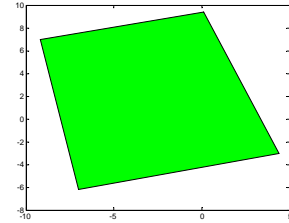
Coloring a polygon (fill)

```
% Draw a rectangle with the lower-left
% corner at (a,b), width w, height h,
% and fill it with a color named by c.
x= [a a+w a+w a a]; % x data
y= [b b b+h b+h b]; % y data
fill(x, y, c)
```

Built-in function `fill` actually does
the "wrap-around" automatically.

26

```
x= [0.1 -9.2 -7 4.4];
y= [9.4 7 -6.2 -3];
fill(x,y,'g')
```



27

Another twinkling constellation

- Write a script that generate 9 random positions—the configuration of my constellation
- Simulate 10 rounds of twinkling
 - In each round, each star is **equally likely** to be lit or black
- Can you add some **random adjustment to the color** of the star?
- Optional:* allow the user to set the constellation by clicking on the figure

28

Example

- Write a program fragment that calculates the **cumulative sums** of a given vector \mathbf{v} .
- The cumulative sums should be stored in a vector of the same length as \mathbf{v} .

1, 3, 5, 0 \mathbf{v}

1, 4, 9, 9 cumulative sums of \mathbf{v}

29

\mathbf{v}

--	--	--	--	--

 csum

--	--	--	--	--

30

\mathbf{v}

--	--	--	--	--

 csum

--	--	--	--	--

 $\text{csum}(k) = \text{csum}(k-1) + \mathbf{v}(k)$

1 2 3

$\text{csum}(3) = \mathbf{v}(1) + \mathbf{v}(2) + \mathbf{v}(3)$ ↗

$\text{csum}(4) = \underbrace{\mathbf{v}(1) + \mathbf{v}(2) + \mathbf{v}(3)}_{\text{csum}(3)} + \mathbf{v}(4)$

```
csum(1) = v(1);
for k = 2 : length(v)
    csum(k) = csum(k-1) + v(k);
end
```

31