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)

<table>
<thead>
<tr>
<th>x</th>
<th>sin(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.0</td>
</tr>
<tr>
<td>1.57</td>
<td>1.0</td>
</tr>
<tr>
<td>3.14</td>
<td>0.0</td>
</tr>
<tr>
<td>4.71</td>
<td>-1.0</td>
</tr>
<tr>
<td>6.28</td>
<td>0.0</td>
</tr>
</tbody>
</table>

Plot based on 5 points

Plot based on 200 discrete points, but it looks smooth

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

Generating tables and plots

\( x = \text{linspace}(0, 2\pi, 9); \)
\( y = \sin(x); \)
\( \text{plot}(x, y); \)

How did we get all the sine values?

Built-in function \text{linspace}:

\( x = \text{linspace}(1, 3, 5) \)
\[ \begin{array}{c}
1.0 \ 1.5 \ 2.0 \ 2.5 \ 3.0
\end{array} \]

\( x = \text{linspace}(0, 1, 101) \)
\[ \begin{array}{c}
0.00 \ 0.01 \ 0.02 \ ... \ 0.99 \ 1.00
\end{array} \]

Built-in functions accept arrays

and return arrays

Note: \( x, y \) are shown in columns due to space limitation; they should be rows.
Vectorized addition

\[
\begin{array}{c}
x \quad 2 \quad 1 \quad 5 \quad 8 \\
+ \quad y \quad 1 \quad 2 \quad 0 \quad 1 \\
\hline
= \quad z \quad 3 \quad 3 \quad 5 \quad 9 \\
\end{array}
\]

Matlab code: \texttt{z = x + y}

Vectorized subtraction

\[
\begin{array}{c}
x \quad 2 \quad 1 \quad 5 \quad 8 \\
- \quad y \quad 1 \quad 2 \quad 0 \quad 1 \\
\hline
= \quad z \quad 1 \quad -1 \quad 5 \quad 7 \\
\end{array}
\]

Matlab code: \texttt{z = x - y}

Vectorized multiplication

\[
\begin{array}{c}
a \quad 2 \quad 1 \quad 5 \quad 8 \\
\times \quad b \quad 1 \quad 2 \quad 0 \quad 1 \\
\hline
= \quad c \quad 2 \quad 2 \quad 0 \quad 8 \\
\end{array}
\]

Matlab code: \texttt{c = a .* b}

Vectorized element-by-element arithmetic operations on arrays

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

\[
\begin{array}{c}
x + y \quad \rightarrow \quad x + y \\
x - y \quad \rightarrow \quad x - y \\
x \times y \quad \rightarrow \quad x \times y \\
x \div y \quad \rightarrow \quad x \div y \\
\end{array}
\]

Shift

\[
\begin{array}{c}
x \quad 3 \\
+ \quad y \quad 2 \quad 1 \quad 5 \quad 8 \\
\hline
= \quad z \quad 5 \quad 4 \quad 3.5 \quad 11 \\
\end{array}
\]

Matlab code: \texttt{z = x + y}

Reciprocate

\[
\begin{array}{c}
x \quad 1 \\
/ \quad y \quad 2 \quad 1 \quad 5 \quad 8 \\
\hline
= \quad z \quad 5 \quad 1 \quad 2 \quad 125 \\
\end{array}
\]

Matlab code: \texttt{z = x ./ y}
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 not necessary but OK.

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

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

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

Name the script `white2black`

Mix two colors

Implement this function:

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

Array index starts at 1

```
x = [5 6 4 3 2 1] / 7
```

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 $k^{th}$ element: $x(k)$
Accessing values in a vector

Given the vector \texttt{score} ...

\begin{align*}
\text{score} & \quad 93 \quad 92 \quad 87 \quad 0 \quad 90 \quad 82 \\
1 & \quad 2 \quad 3 \quad 4 \quad 5 \quad 6
\end{align*}

\begin{align*}
\text{score}(4) &= 80; \\
\text{score}(5) &= (\text{score}(4)+\text{score}(5))/2; \\
k &= 1; \\
\text{score}(k+1) &= 99;
\end{align*}

A few different ways to create a vector

(\textit{More later!})

\begin{align*}
\texttt{count} &= \texttt{zeros}(1,6) \\
\texttt{x} &= \texttt{linspace}(10,30,5) \\
\texttt{y} &= [3 \quad 7 \quad 2 \quad 1] \\
\texttt{Z} &= [3; \quad 7; \quad 2]
\end{align*}

Drawing a single line segment

\begin{align*}
a &= 0; & \text{\% x-coord of pt 1} \\
b &= 1; & \text{\% y-coord of pt 1} \\
c &= 5; & \text{\% x-coord of pt 2} \\
d &= 3; & \text{\% y-coord of pt 2} \\
\text{plot}([a \ c], [b \ d], '-*')
\end{align*}

Coloring a polygon (fill)

\begin{align*}
\texttt{fill}(x, y, c)
\end{align*}

A built-in function

\begin{align*}
\texttt{x-values} & \quad (\text{a vector}) \\
\texttt{y-values} & \quad (\text{a vector}) \\
\text{Line/marker format}
\end{align*}
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]; % 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.

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

Example

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

```
1, 3, 5, 0  v
1, 4, 9, 9  cumulative sums of v
```

```
v
(csum)
1 3

\[ c_{sum}(k) = c_{sum}(k-1) + v(k) \]
```

\[ c_{sum}(5) = v(1)+v(2)+v(3) \]
\[ c_{sum}(4) = v(1)+v(2)+v(3) + v(4) \]
\[ c_{sum}(3) \]
\[ c_{sum}(1) = v(1); \]
for \( k = 2 : \text{length}(v) \)
\[ c_{sum}(k) = c_{sum}(k-1) + v(k), \]
end