Previous Lecture:
- Function scope
- Modular programming—use functions

Today’s Lecture:
- Finite vs. Infinite; Discrete vs. Continuous
- Vectors and vectorized code
- `plot` and `fill`

Announcements:
- Project 3 due Thurs, 2/28
- Prelim 1 on Mar 7th at 7:30pm. Email Randy Hess (rbh27) now if you have an exam conflict (specify conflicting course and instructor contact info)
Screen Granularity

After how many halvings will the disks disappear?
Xeno’s Paradox

- A wall is two feet away
- Take steps that repeatedly halve the remaining distance
- You never reach the wall because the distance traveled after $n$ steps is:

$$1 + \frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{2^n} = 2 - \frac{1}{2^n}$$
Example: “Xeno” disks

Draw a sequence of 20 disks where the \((k+1)\)th disk has a diameter that is half that of the \(k\)th disk.

The disks are tangent to each other and have centers on the \(x\)-axis.

First disk has diameter 1 and center \((1/2, 0)\).
Example: “Xeno” disks

What do you need to keep track of?

- Diameter \((d)\)
- Position
  - Left tangent point \((x)\)

<table>
<thead>
<tr>
<th>Disk</th>
<th>(x)</th>
<th>(d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0+1</td>
<td>1/2</td>
</tr>
<tr>
<td>3</td>
<td>0+1+1/2</td>
<td>1/4</td>
</tr>
</tbody>
</table>
% Xeno Disks

DrawRect(0,-1,2,2,'k')

% Draw 20 Xeno disks
% Xeno Disks

DrawRect(0,-1,2,2,'k')
% Draw 20 Xeno disks

for k = 1:20

end
% Xeno Disks

DrawRect(0,-1,2,2,'k')
% Draw 20 Xeno disks

d= 1;
x= 0;  % Left tangent point
for k= 1:20

end
% Xeno Disks

DrawRect(0,-1,2,2,'k')
% Draw 20 Xeno disks

d = 1;
x = 0;  % Left tangent point

for k = 1:20
    % Draw kth disk

    % Update x, d for next disk

end
% Xeno Disks

DrawRect(0,-1,2,2,'k')
% Draw 20 Xeno disks

d= 1;
x= 0;  % Left tangent point
for k= 1:20
    % Draw kth disk
    DrawDisk(x+d/2, 0, d/2, 'y')
    % Update x, d for next disk
    x= x+d;
d= d/2;
end
Here’s the output… Shouldn’t there be 20 disks?

The “screen” is an array of dots called pixels.

Disks smaller than the dots don’t show up.

The 20th disk has radius < 0.00001
Does this script print anything?

```matlab
k = 0;
while 1 + 1/2^k > 1
    k = k+1;
end
disp(k)
```
Computer Arithmetic—floating point arithmetic

Suppose you have a calculator with a window like this:

```
+ 2 4 1 - 3
```

representing $2.41 \times 10^{-3}$
Floating point addition

\[ \begin{array}{c}
+ & 2 & 4 & 1 & - & 3 \\
+ & 1 & 0 & 0 & - & 3 \\
\end{array} \]

Result: \[ + \ 3 \ 4 \ 1 \ - \ 3 \]
Floating point addition

+ 2 4 1 - 3

+ 1 0 0 - 4

Result: + 2 5 1 - 3
Floating point addition

Result: $+242 - 3$
Floating point addition

\[
\begin{align*}
+ & \ 2 & 4 & 1 & - & 3 \\
+ & 1 & 0 & 0 & - & 6 \\
\text{Result:} & \ + & 2 & 4 & 1 & - & 3
\end{align*}
\]
Floating point addition

\[
\begin{array}{c}
+ & 2 & 4 & 1 & - & 3 \\
+ & 1 & 0 & 0 & - & 6 \\
\end{array}
\]

Result: \[
\begin{array}{c}
+ & 2 & 4 & 1 & - & 3 \\
\end{array}
\]

Not enough room to represent \(.002411\)
The loop DOES terminate given the limitations of floating point arithmetic!

```markdown
k = 0;
while 1 + 1/2^k > 1
    k = k+1;
end
disp(k)
```

1 + 1/2^53 is calculated to be just 1, so “53” is printed.
In 1991, a Patriot Missile failed, resulting in 28 deaths and about 100 injured. The cause?
Inexact representation of time/number

- System clock represented time in tenths of a second: a clock tick every $1/10$ of a second

- Time = number of clock ticks $\times 0.1$

  “exact” value
  
  $0.0001100110011001100110011...$

  $0.0001100110011001100110011...$ value in Patriot system

Error of $0.000000095$ every clock tick
Resulting error

... after 100 hours

\[ .000000095 \times (100 \times 60 \times 60) \]

0.34 second

At a velocity of 1700 m/s, missed target by more than 500 meters!
Computer arithmetic is \textit{inexact}

- There is error in computer arithmetic—floating point arithmetic—due to limitation in “hardware.” Computer memory is \textit{finite}.

- What is $1 + 10^{-16}$?
  - 1.0000000000000001 in real arithmetic
  - 1 in floating point arithmetic (IEEE)

- Read Sec 4.3
Discrete vs. continuous

Plot made from discrete values, but it looks continuous since there're many points.
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.
Generating tables and plots

<table>
<thead>
<tr>
<th>x</th>
<th>sin(x)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>0.784</td>
<td>0.707</td>
</tr>
<tr>
<td>1.571</td>
<td>1.000</td>
</tr>
<tr>
<td>2.357</td>
<td>0.707</td>
</tr>
<tr>
<td>3.142</td>
<td>0.000</td>
</tr>
<tr>
<td>3.927</td>
<td>-0.707</td>
</tr>
<tr>
<td>4.712</td>
<td>-1.000</td>
</tr>
<tr>
<td>5.498</td>
<td>-0.707</td>
</tr>
<tr>
<td>6.283</td>
<td>0.000</td>
</tr>
</tbody>
</table>

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

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

x, y are vectors. A vector is a 1-dimensional list of values.
**Built-in function **\texttt{linspace}**

\begin{align*}
x &= \texttt{linspace}(1, 3, 5) \\
x &\begin{array}{c}
1.0 \\
1.5 \\
2.0 \\
2.5 \\
3.0 \\
\end{array}
\end{align*}

\begin{align*}
x &= \texttt{linspace}(0, 1, 101) \\
x &\begin{array}{c}
0.00 \\
0.01 \\
0.02 \\
\ldots \\
0.99 \\
1.00 \\
\end{array}
\end{align*}
How did we get all the sine values?

Built-in functions accept arrays

<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>

and return arrays

0.00 1.00 0.00 -1.00 0.00
Examples of functions that can work with arrays

```matlab
x = linspace(0, 1, 200);
y = exp(x);
plot(x, y)

x = linspace(1, 10, 200);
y = log(x);
plot(x, y)
```
Does this assign to $y$ the values $\sin(0^\circ), \sin(1^\circ), \sin(2^\circ), \ldots, \sin(90^\circ)$?

\[
x = \text{linspace}(0, \text{pi}/2, 90);
\]

\[
y = \sin(x);
\]

A: yes  B: no
Can we plot this?

\[ f(x) = \sin(5x) - x \]

for \(-2 \leq x \leq 3\)
Can we plot this?

\[ f(x) = \sin(5x) - x \]

for \(-2 \leq x \leq 3\)

Yes!

\[
\begin{align*}
x &= \text{linspace}(-2,3,200); \\
y &= \sin(5\times x) - x; \\
\text{plot}(x,y)
\end{align*}
\]

Element-by-element arithmetic operations on arrays
Vectorized addition

\[ x = \begin{bmatrix} 2 & 1 & .5 & 8 \end{bmatrix} \]

\[ + \quad y = \begin{bmatrix} 1 & 2 & 0 & 1 \end{bmatrix} \]

\[ = \quad z = \begin{bmatrix} 3 & 3 & .5 & 9 \end{bmatrix} \]

Matlab code: \( z = x + y \)
Vectorized subtraction

\[ x = \begin{bmatrix} 2 & 1 & .5 & 8 \end{bmatrix} \]

\[ y = \begin{bmatrix} 1 & 2 & 0 & 1 \end{bmatrix} \]

\[ z = x - y = \begin{bmatrix} 1 & -1 & .5 & 7 \end{bmatrix} \]

Matlab code:  \( z = x - y \)
Vectorized code
— a Matlab-specific feature

- 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.

See Sec 4.1 for list of vectorized arithmetic operations.
Vectorized multiplication

\[ \begin{bmatrix} 2 & 1 & 0.5 & 8 \\ 1 & 2 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 2 & 2 & 0 & 8 \end{bmatrix} \]

Matlab code: \( c = a \times b \)
Vectorized element-by-element arithmetic operations on arrays

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

See full list of ops in §4.1
Shift

Matlab code: \( z = x + y \)
Reciprocate

\[ \begin{align*}
\text{x} & \quad 1 \\
\text{\(\div\)} & \quad \text{y} \\
\hline
\text{\(\div\)} & \quad \begin{array}{cccc}
2 & 1 & .5 & 8 \\
\end{array} \\
\hline
\text{=} & \quad \text{z} \\
\text{\(\div\)} & \quad \begin{array}{cccc}
.5 & 1 & 2 & .125 \\
\end{array}
\end{align*} \]

Matlab code: \( z = x \div y \)
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
Element-by-element arithmetic operations on arrays…
Also called “vectorized code”

\[ x = \text{linspace}(-2, 3, 200); \]
\[ y = \sin(5x) \cdot \exp(-x/2)/(1 + x^2); \]

Contrast with scalar operations that we’ve used previously…

\[ a = 2.1; \]
\[ b = \sin(5a); \]

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

When an operand is a vector, you have “vectorized code.”
l-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.

\[
v = \begin{bmatrix}
0.8 & 0.2 & 1 \\
1 & 2 & 3
\end{bmatrix}
\]
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 $k^{th}$ element: $x(k)$
Accessing values in a vector

Given the vector \texttt{score} …
Accessing values in a vector

Given the vector `score` ...

\[ \text{score}(4) = 80; \]
\[ \text{score}(5) = \frac{(\text{score}(4) + \text{score}(5))}{2}; \]
\[ k = 1; \]
\[ \text{score}(k+1) = 99; \]
Here are a few different ways to create a vector

count = zeros(1, 6)

Similar functions: ones, rand

a = linspace(10, 30, 5)

b = 7:-2:0

c = [3 7 2 1]

d = [3; 7; 2]

e = d'
Drawing a polygon (multiple line segments)

% Draw a rectangle with the lower-left corner at (a,b), width w, height h.

\[
x= [a \quad a+w \quad a+w \quad a \quad a] \quad \text{%; \ x data}
\]

\[
y= [b \quad b \quad b+h \quad b+h \quad b] \quad \text{%; \ y data}
\]

plot(x, y)

Fill in the missing vector values!
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)
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.

\[x = [a \ a+w \ a+w \ a \ a] ; \quad \% \ x \ data\]
\[y = [b \ b \ b+h \ b+h \ b] ; \quad \% \ y \ data\]
\[\text{fill}(x, y, c)\]

A built-in function
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.