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)

---

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 = 
  \[ 1 + \frac{1}{2} + \frac{1}{4} + \ldots + \frac{1}{2^n} = 2 - \frac{1}{2^n} \]

---

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 \((1/2, 0)\).

---

```matlab
% Xeno Disks
DrawRect(0,-1,2,2,'k')

% Draw 20 Xeno disks
for k = 1:20
    % Disk    x      d
    %-----------------------------
    % 1       0       1
    % 2       0+1    1/2
    % 3       0+1+1/2 1/4
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 < .000001

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

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

```
+ 1 0 0 - 3
```

Result:

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

Floating point addition

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

```
+ 1 0 0 - 4
```

Result:

```
+ 2 5 1 - 3
```

Floating point addition

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

```
+ 1 0 0 - 6
```

Result:

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

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

```
    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.

Patriot missile failure

In 1991, a Patriot Missile failed, resulting in 28 deaths and about 100 injured. The cause?

![Patriot missile image]

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 \(\times 0.1\)
- Error of .000000095 every clock tick

Resulting error

\[
\ldots \text{after 100 hours} \ni \quad .000000095 \times (100\times60\times60) 
\]

0.34 second

At a velocity of 1700 m/s, missed target by more than 500 meters!

Computer arithmetic is inexact

- There is error in computer arithmetic—floating point arithmetic—due to limitation in "hardware." Computer memory is finite.
- What is 1 + 10\(^{-16}\)?
  - 1.000000000000001 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>0.00</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>

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

Built-in function linspace

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

Examples of functions that can work with arrays

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

x = linspace(1,10,200);
y = log(x);
plot(x,y)

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
Does this assign to \( y \) the values \( \sin(0^\circ), \sin(1^\circ), \sin(2^\circ), \ldots, \sin(90^\circ) \)?

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

A: yes  B: no

Can we plot this?

\[
f(x) = \sin(5x) - x \quad \text{for} \quad -2 \leq x \leq 3
\]

Yes!

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

Element-by-element arithmetic operations on arrays

Vectorized addition

\[
x = \begin{bmatrix} 2 & 1 & 5 & 8 \end{bmatrix}
\]
\[
y = \begin{bmatrix} 1 & 2 & 0 & 1 \end{bmatrix}
\]
\[
z = \begin{bmatrix} 3 & 3 & 5 & 9 \end{bmatrix}
\]

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

Vectorized multiplication

\[
a = \begin{bmatrix} 2 & 1 & 5 & 8 \end{bmatrix}
\]
\[
b = \begin{bmatrix} 1 & 2 & 0 & 1 \end{bmatrix}
\]
\[
c = \begin{bmatrix} 2 & 2 & 0 & 8 \end{bmatrix}
\]

Matlab code: \( c = a .* b \)

Element-by-element arithmetic operations on arrays

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

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

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

Reciprocate

\[
\begin{array}{c}
\times 1 \\
/ \ y \ 2 \ 1 \ 5 \ 8 \\
= \ z \ 0.5 \ 1 \ 2 \ 0.125
\end{array}
\]

Matlab code: \( z = x ./ y \)

Vectorized

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

\[
\begin{array}{c}
+ \\
- \\
* \\
/ \\
\end{array}
\]

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…

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

Array index starts at 1

\[
\begin{array}{c}
\times \ 5 \ 4 \ 91 \ 4 \ 1 \ 7 \\
1 \ 2 \ 3 \ 4 \ 5 \ 6
\end{array}
\]

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: \( x(k) \)

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

\[
\begin{array}{c}
\times \ 0.8 \ 2 \ 1 \\
1 \ 2 \ 3
\end{array}
\]
Accessing values in a vector

Given the vector `score`...

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

Here are a few different ways to create a vector

```matlab
count = zeros(1, 6);  % count 0 0 0 0 0 0
a = linspace(10, 30, 5);  % a = 10 15 20 25 30
b = 7:-2:0;  % b = 7 5 3 1
k = [3 7 2 1];  % k = 3 7 2 1
r = [3; 7; 2];  % r = 3 7 2
s = r';  % s = 3 7 2
```

Drawing a polygon (multiple line segments)

```matlab
% 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)

Fill in the missing vector values!
```

Coloring a polygon (fill)

```matlab
% 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.
```

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

Can be a vector
(RGB values)