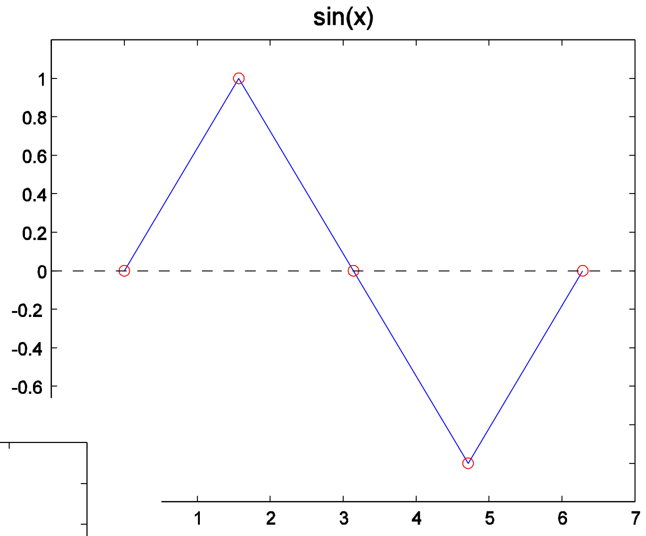
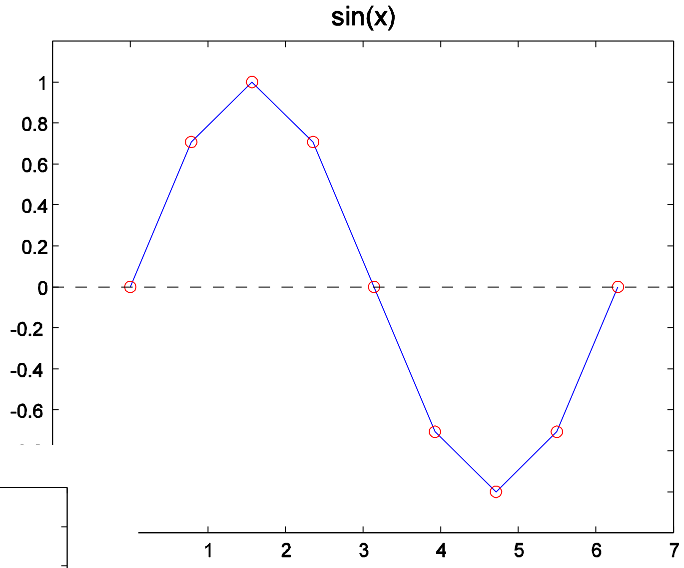
- Previous Lecture:
  - Probabilities and vectors in simulation

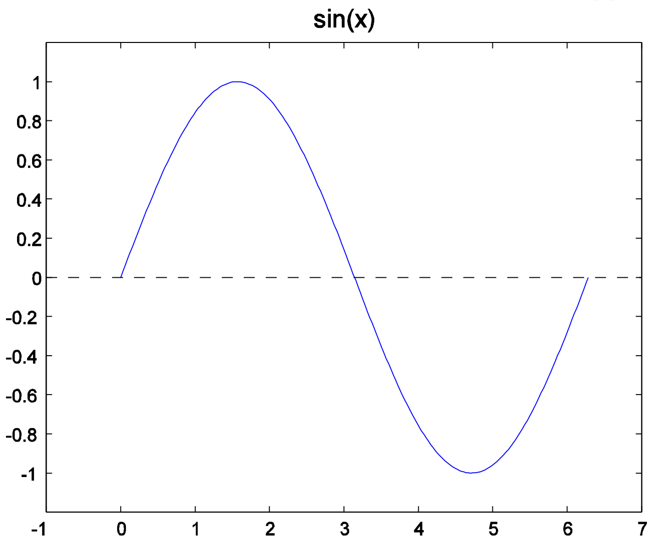
- Today's Lecture (Ch. 4):
  - Discrete vs. Continuous
  - Vectorized calculations
  - Colors and linear interpolation
  - Floating-point arithmetic

- Announcements:
  - Discussion this week in Hollister 401 classroom
  - Project 3 due at 11pm on Wednesday, 3/4
    - No exercise check-off at this Wednesday's office/consulting hours due to project deadline
  - Prelim 1 on Tues 3/10 at 7:30pm
    - Review materials will be posted soon. An *optional* review session is scheduled for Sunday, 3/8 (time, location TBD)
    - Alternate exam: look out for email, and be prepared to start early (5:45pm)

Discrete vs. continuous







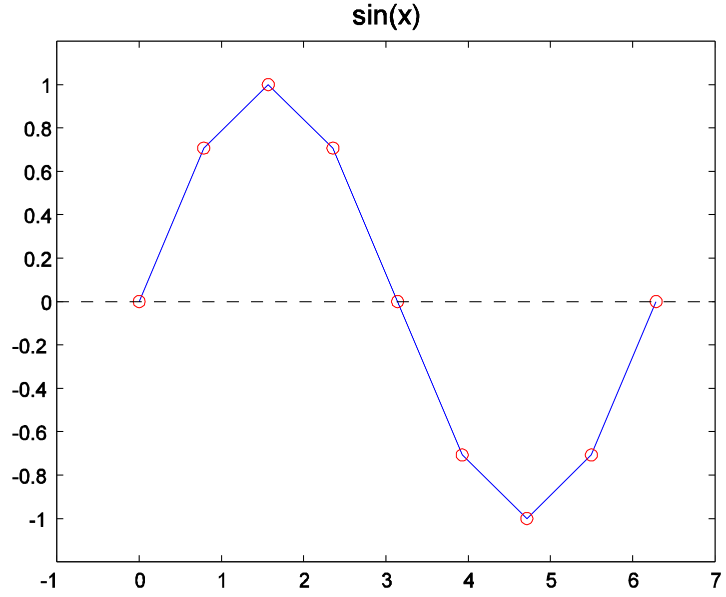
A plot is made from discrete values, but it can look continuous if there are many points

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)
```



How did we get all the sine values?

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

Built-in functions accept vectors

0.00	1.57	3.14	4.71	6.28

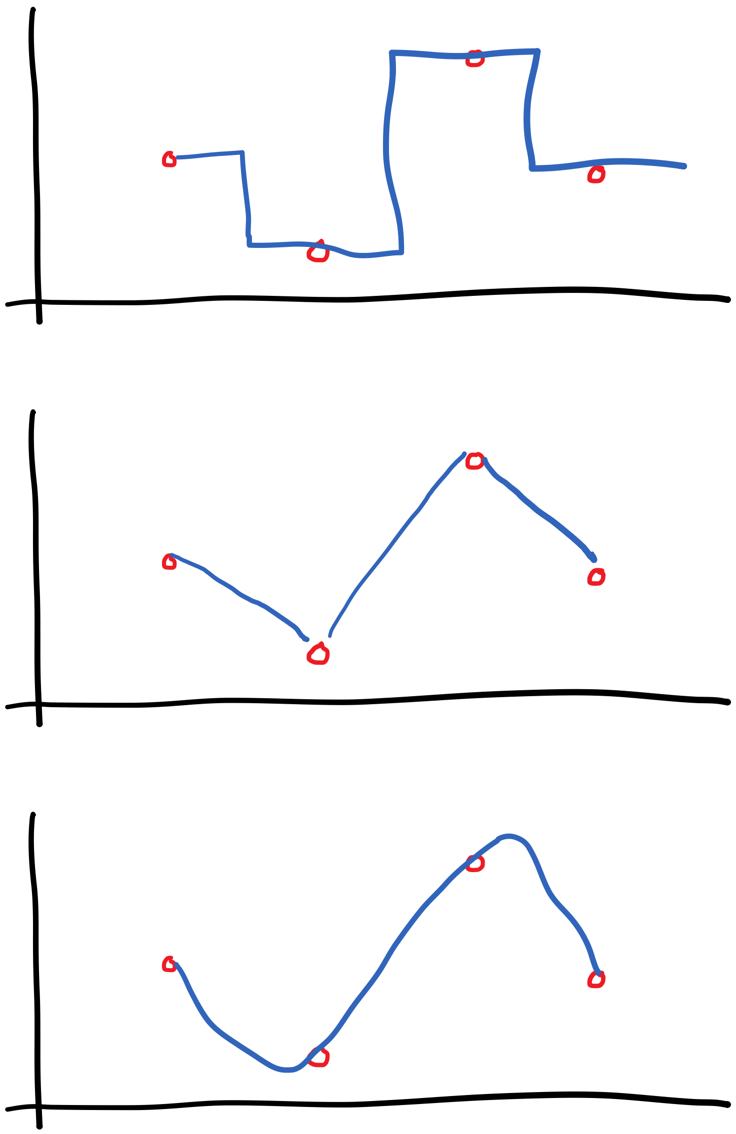
`sin()`

and return vectors

0.00	1.00	0.00	-1.00	0.00

Connecting the dots (discrete -> continuous)

- Copy value of closest point?
- **Linearly interpolate between two points?**
- Interpolate neighboring points too?



“Best” choice depends on how much you know about where the data comes from.

Linear interpolation

- Two-point formula for line

$$\frac{y - y_1}{y_2 - y_1} = \frac{x - x_1}{x_2 - x_1}$$

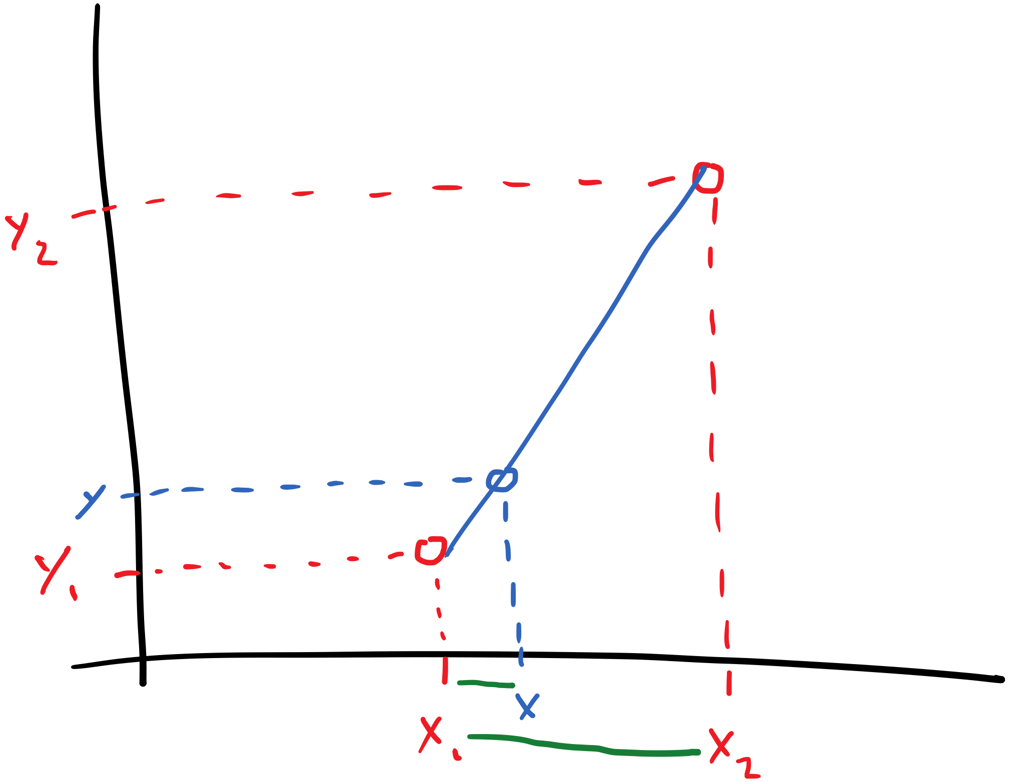
solve for y

- Weighted average

$$y = \frac{w_1 y_1 + w_2 y_2}{w_1 + w_2}$$

$$= (1-f)y_1 + f y_2$$

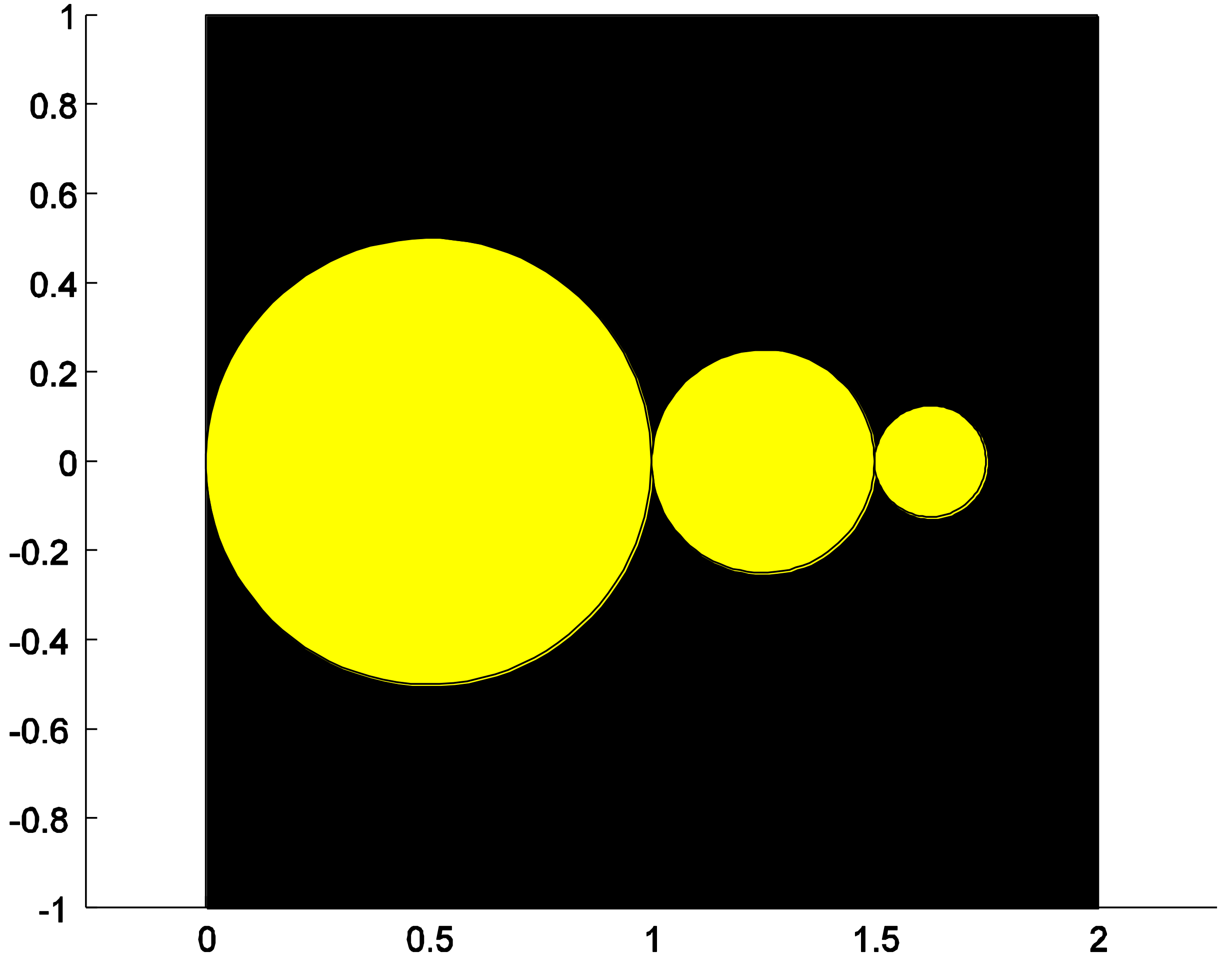
$$0 \le f \le 1$$



$f =$ distance from first point as a fraction of interval width

$$= \frac{x - x_1}{x_2 - x_1}$$

Example: Shrinking disks & resolution

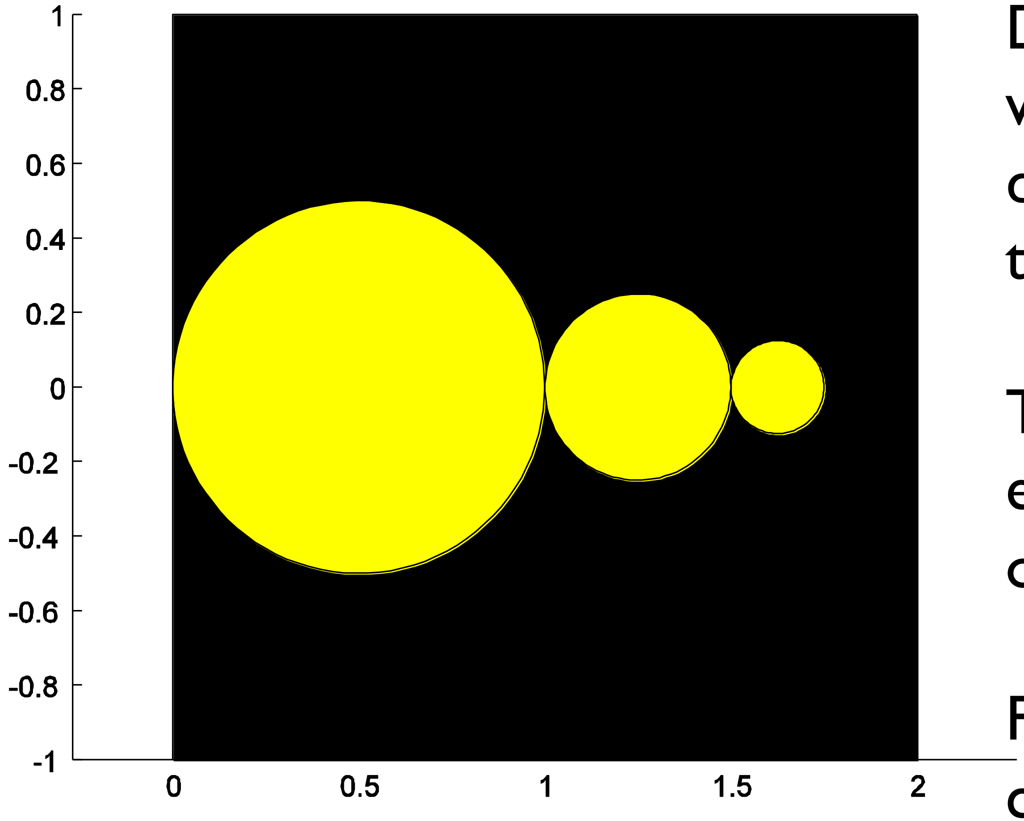


How many disks will fit in the box?

$$\sum_{n=0}^{\infty} \frac{1}{2^n} = 2$$

How many disks can we see?

Example: “Xeno” disks

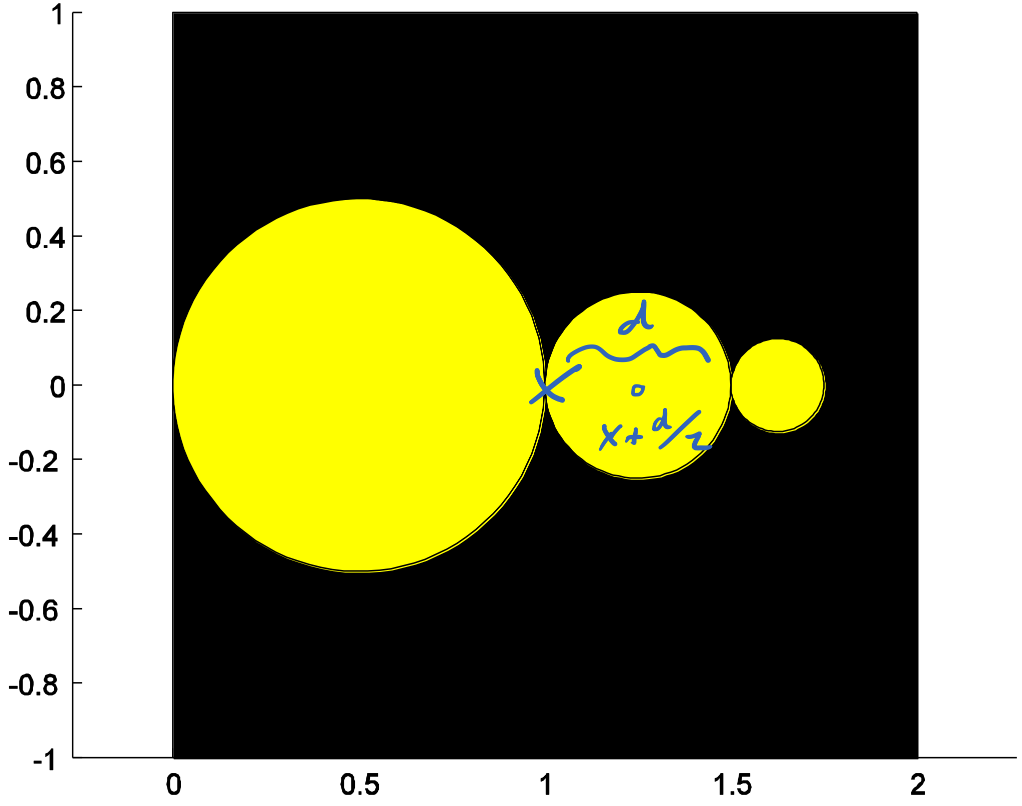


Draw a sequence of 20 disks where the (k+1)th disk has a diameter that is half that of the kth disk.

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



Repeating process

What do you need to keep track of?

- Diameter (d)
- Position

  Left tangent point (x)

Disk	x	d
1	0	1
2	0+1	1/2
3	0+1+1/2	1/4

accumulation

Pattern

```
% Xeno Disks

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

Script comment (could be better...)

Setup

Highest-level outline

```
% Xeno Disks

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



for k= 1:20
    % Draw the kth disk




end
```

Pattern: repeat
N times

```
% Xeno Disks

DrawRect(0,-1,2,2,'k')
% Draw 20 Xeno disks
d= 1;   % Diameter of first disk
x= 0;   % Left tangent point
for k= 1:20
    % Draw the kth disk

    % Update x, d for next disk



end
```

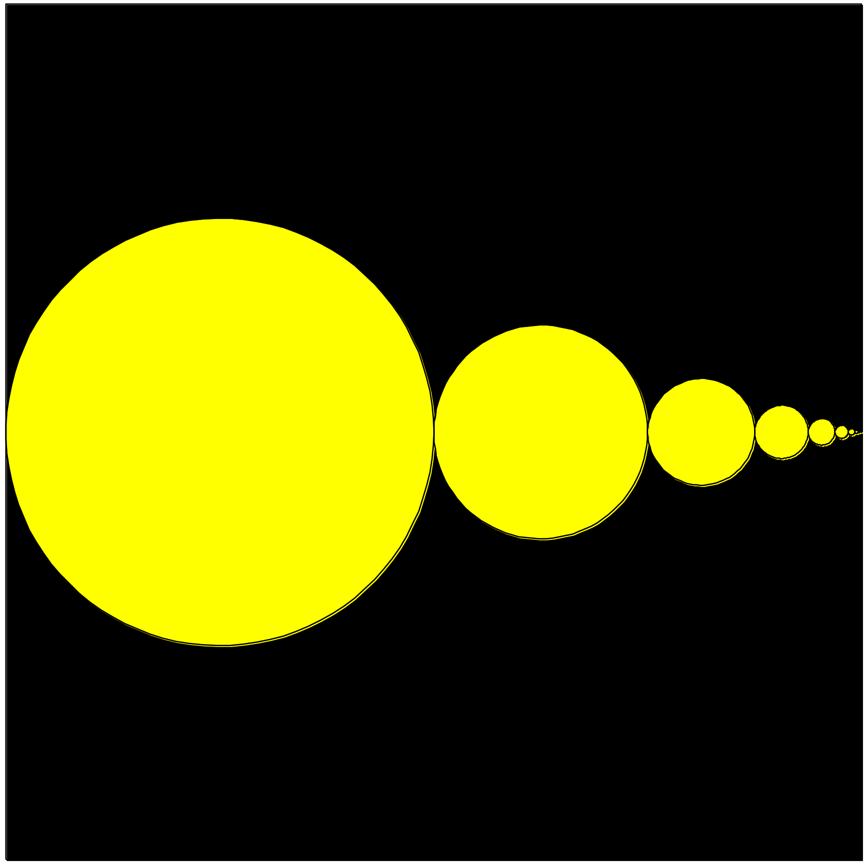
Initialize "state"

outline: maintain "state"

```
% Xeno Disks

DrawRect(0,-1,2,2,'k')
% Draw 20 Xeno disks
d= 1;
x= 0;   % Left tangent point
for k= 1:20
    % Draw the kth disk
      DrawDisk(x+d/2, 0, d/2, 'y')
    % Update x, d for next disk
      x= x + d;
      d= d/2;

end
```

Refine outline w/ code

Here's the output… Shouldn't there be 20 disks?

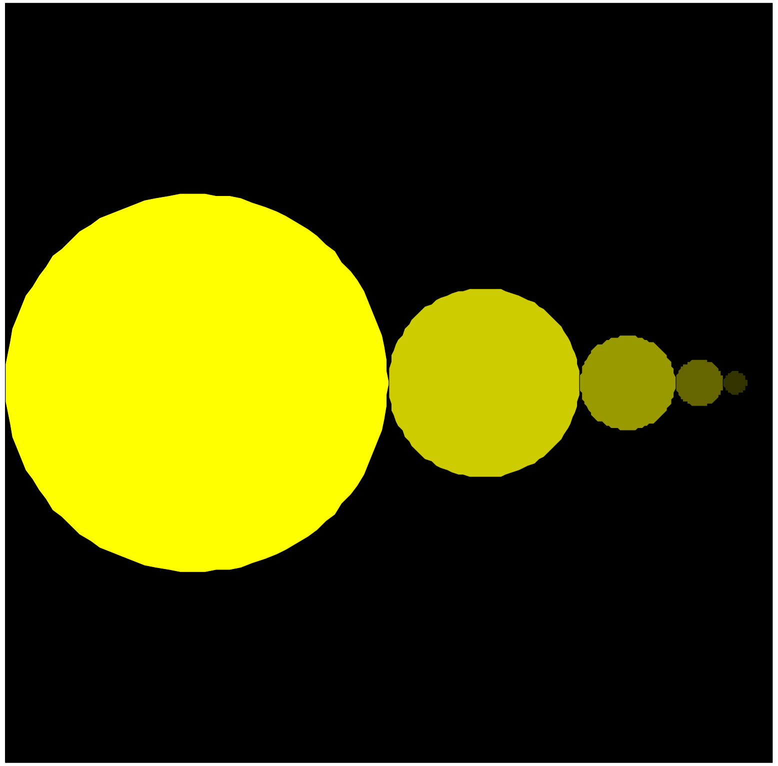


The "screen" is an array of dots called pixels.

Disks smaller than the dots don't show up.

The 20th disk has radius<.000001

Fading Xeno disks



- First disk is yellow
- Last disk is black (invisible)
- *<u>Interpolate</u>* the color in between

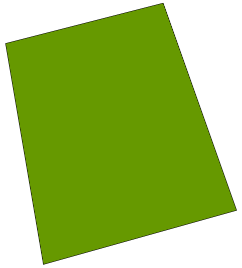
$(1-f) * \text{"yellow"} + f * \text{"black"}$

How can we multiply, add colors?

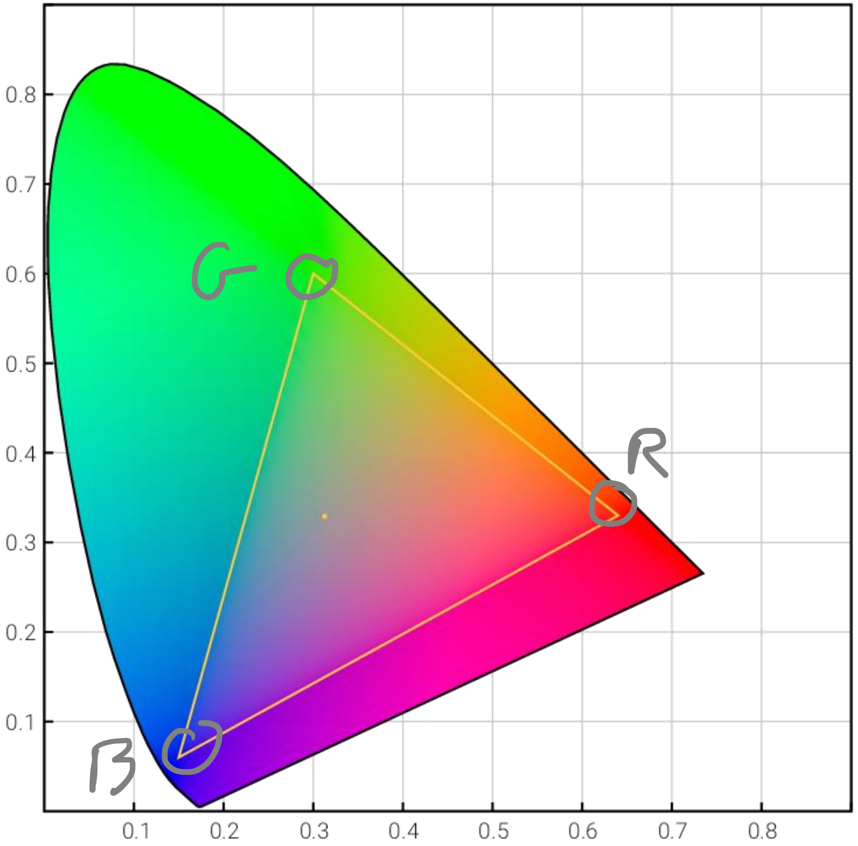
Color can be represented by a 3-vector storing RGB values

- Most any color is a mix of red, green, and blue
- Example:

```
colr= [0.4  0.6  0]
```



- Each component is a number between 0 and 1
- [0 0 0] is black
- [1 1 1] is white



```
% Draw n Xeno disks
d= 1;
x= 0;   % Left tangent point



for k= 1:n




    % Draw kth disk
    DrawDisk(x+d/2, 0, d/2, 'y')
    x= x+d;
    d= d/2;
end
```

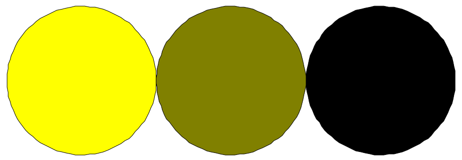
```
% Draw n Xeno disks
d= 1;
x= 0;   % Left tangent point



for k= 1:n




    % Draw kth disk
    DrawDisk(x+d/2, 0, d/2, [1 1 0])
    x= x+d;
    d= d/2;
end
```

A vector of length 3

```
% Draw n fading Xeno disks
d= 1;
x= 0;   % Left tangent point
yellow= [1 1 0];
black=  [0 0 0];
for k= 1:n
   % Compute color of kth disk



   % Draw kth disk
   DrawDisk(x+d/2, 0, d/2, _______)
   x= x+d;
   d= d/2;
end
```

Example: 3 disks fading from yellow to black

```
r= 1;   % radius of  disk
yellow= [1 1 0];
black = [0 0 0];

% Left disk yellow, at x=1
DrawDisk(1,0,r,yellow)
% Right disk black, at x=5
DrawDisk(5,0,r,black)

% Middle disk with average color, at x=3
colr= 0.5*yellow + 0.5*black;
DrawDisk(3,0,r,colr)
```



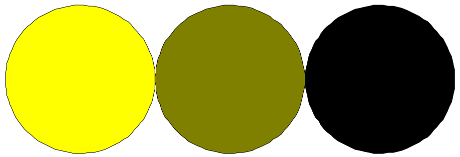
Example: 3 disks fading from yellow to black

```
r= 1;   % radius of  disk
yellow= [1 1 0];
black = [0 0 0];

% Left disk yellow, at x=1
DrawDisk(1,0,r,yellow)
% Right disk black, at x=5
DrawDisk(5,0,r,black)

% Middle disk with average color, at x=3
colr= 0.5*yellow + 0.5*black;
DrawDisk(3,0,r,colr)
```

$$.5 * \boxed{1}\boxed{1}\boxed{0} \rightarrow \boxed{.5}\boxed{.5}\boxed{0}$$

$$.5 * \boxed{0}\boxed{0}\boxed{0} \rightarrow \boxed{0}\boxed{0}\boxed{0}$$

Vectorized multiplication

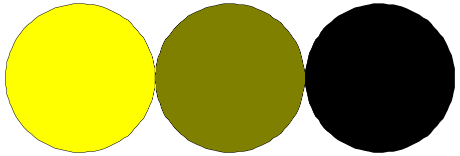


Example: 3 disks fading from yellow to black

```
r= 1;   % radius of  disk
yellow= [1 1 0];
black = [0 0 0];

% Left disk yellow, at x=1
DrawDisk(1,0,r,yellow)
% Right disk black, at x=5
DrawDisk(5,0,r,black)

% Middle disk with average color, at x=3
colr= 0.5*yellow + 0.5*black;
DrawDisk(3,0,r,colr)
```

Vectorized addition

	.5	.5	0
+	0	0	0
=	.5	.5	0



Vectorized code allows an operation on multiple values at the same time

Vectorized addition

$$\begin{array}{r}
\boxed{.5}\boxed{.5}\boxed{0} \\
+ \quad \boxed{0}\boxed{0}\boxed{0} \\
\hline
= \quad \boxed{.5}\boxed{.5}\boxed{0}
\end{array}$$

```
yellow= [1 1 0];
black = [0 0 0];
```

```
% Average color via vectorized op
colr= 0.5*yellow + 0.5*black;
```

Operation performed on vectors

```
% Average color via scalar op
for k = 1:length(black)
   colr(k)= 0.5*yellow(k) + 0.5*black(k);
end
```

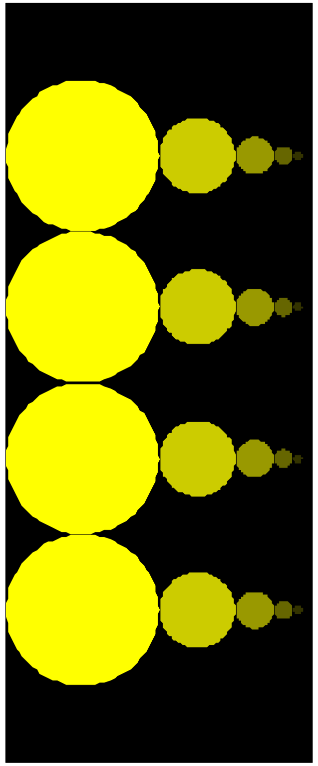
Operation performed on scalars

```
% Draw n fading Xeno disks
d= 1;
x= 0;  % Left tangent point
yellow= [1 1 0];
black=  [0 0 0];
for k= 1:n
   % Compute color of kth disk



   % Draw kth disk
   DrawDisk(x+d/2, 0, d/2, _______)
   x= x+d;
   d= d/2;
end
```

```
% Draw n fading Xeno disks
d= 1;
x= 0;   % Left tangent point
yellow= [1 1 0];
black=  [0 0 0];
for k= 1:n
    % Compute color of kth disk
    f= ???
    colr= f*black + (1-f)*yellow;
    % Draw kth disk
    DrawDisk(x+d/2, 0, d/2, colr)
    x= x+d;
    d= d/2;
end
```

A k/n
B k/(n-1)
C (k-1)/n
D (k-1)/(n-1)
E (k-1)/(n+1)

Rows of Xeno disks



for y = __ : __ : __

Code to draw one row of Xeno disks at some y-coordinate

end

Be careful with initializations

```
yellow=[1 1 0];  black=[0 0 0];

d= 1;

x= 0;

for k= 1:n
   % Compute color of kth disk
   f= (k-1)/(n-1);
   colr= f*black + (1-f)*yellow;
   % Draw kth disk
   DrawDisk(x+d/2, 0, d/2, colr)
   x=x+d;  d=d/2;
end
```

Where to put the loop header `for y=__:__:__`


```
A →
    yellow=[1 1 0];  black=[0 0 0];
B →
    d= 1;
C →
    x= 0;
D →
    for k= 1:n
        % Compute color of kth disk
        f= (k-1)/(n-1);
        colr= f*black + (1-f)*yellow;
        % Draw kth disk
        DrawDisk(x+d/2, 0, d/2, colr)
        x=x+d;  d=d/2;
    end
end
```

y

How does Matlab do math?

- Matlab implements an *approximation* to real arithmetic
- The digital number line is *discrete*, not continuous
- Calculations accumulate rounding error, leading to *uncertainty* in results

The approximation is usually *very good*, but don’t get caught off guard

Binary floating-point arithmetic

- Range is finite
- Precision is finite
- Precision is relative
- Fractions are not base-10

- Smallest non-zero number: $\sim 10^{-324}$
  - Going smaller will underflow to 0
- Largest finite number: $\sim 10^{308}$
  - Going bigger will overflow to inf

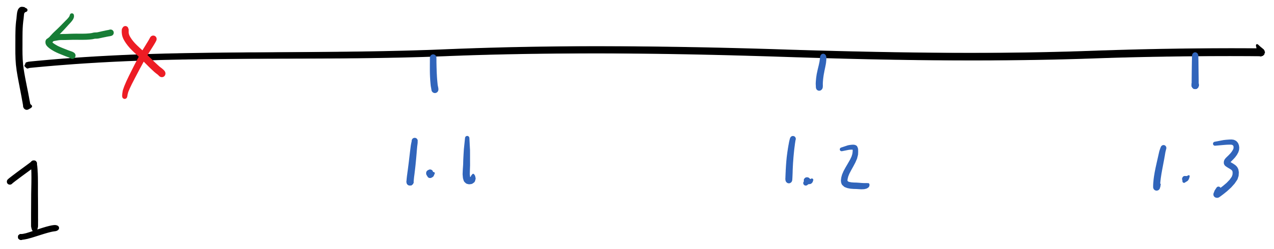
Precision is finite

- Numbers are discrete
  - Only save a small number of decimal places
  - Gaps between “adjacent” numbers
- If a result falls in between two numbers, need to **round** the result

ex. Keep 2 digits

$4.1 \div 4 = 1.025$

$\rightarrow 1.0$



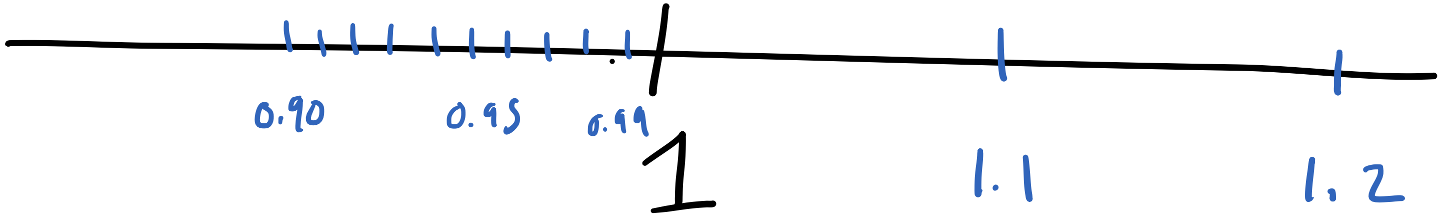
Precision is relative

- Numbers are stored in "scientific notation"
  - Only save a small number of **significant digits**

ex. keep 2 sig figs

$1.5 \div 2 = 0.75$

$= 7.5 \times 10^{-1}$ ✓

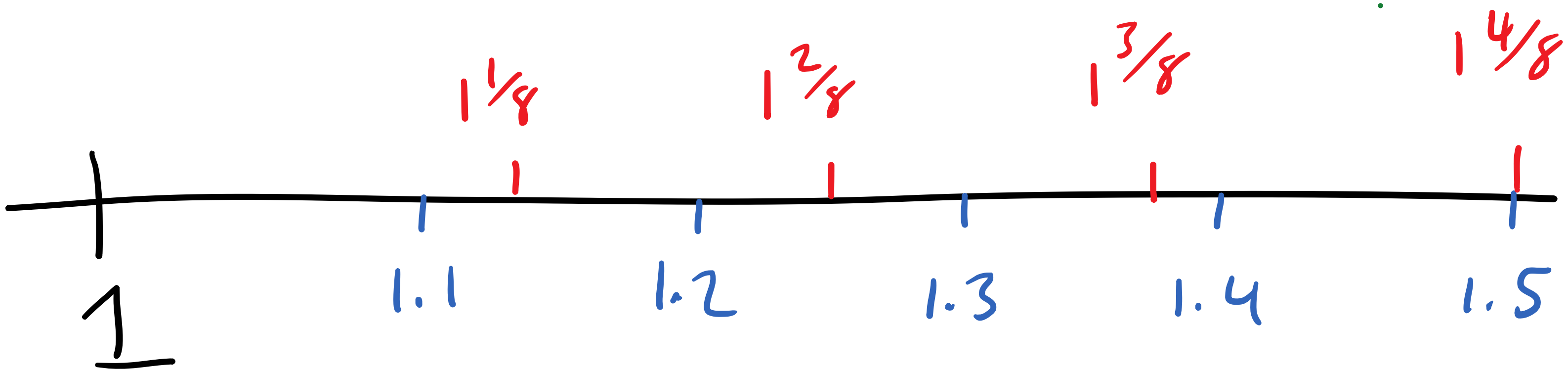


Fractions are not base-10

- Digits count powers of 2, not powers of 10
- “simple” decimal numbers (like 0.1) fall in the gap, are approximated
- Precision is roughly the same as 16 decimal digits

ex. 2 decimal digits vs. 4 binary digits

“1.1” → 1.125



Most software (inc. Matlab) can only use red #s

Peeling back the curtain

- By default, Matlab prints 5 significant digits (`format short`)
- With `format long`, Matlab prints 16 significant digits
- To unambiguously express a double as a decimal, need 17 significant digits

Pro tip: when printing numbers that will be consumed by both humans *and* computers, use:

```
fprintf('%.17g', x)
```

“Bonus numbers”

- `inf`: Represents “infinity”
  - Both positive and negative versions
  - Larger (or smaller) than any other number
  - Generated on overflow or when dividing by zero
- `nan`: Not-a-number
  - Not equal to anything (even itself)
  - Generated from `0/0`, `inf*0`, …

Does this script print anything?


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

A: No – the loop guard is always true

B: Yes, 1/2^k will underflow to 0

C: Yes, 1+1/2^k will round down to 1

D: No – a floating-point error will stop the program

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.

Computer arithmetic is *inexact*

- There is error in computer arithmetic—floating point arithmetic—due to limitation in "hardware." Computer memory is finite.

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

- Read Sec 4.3