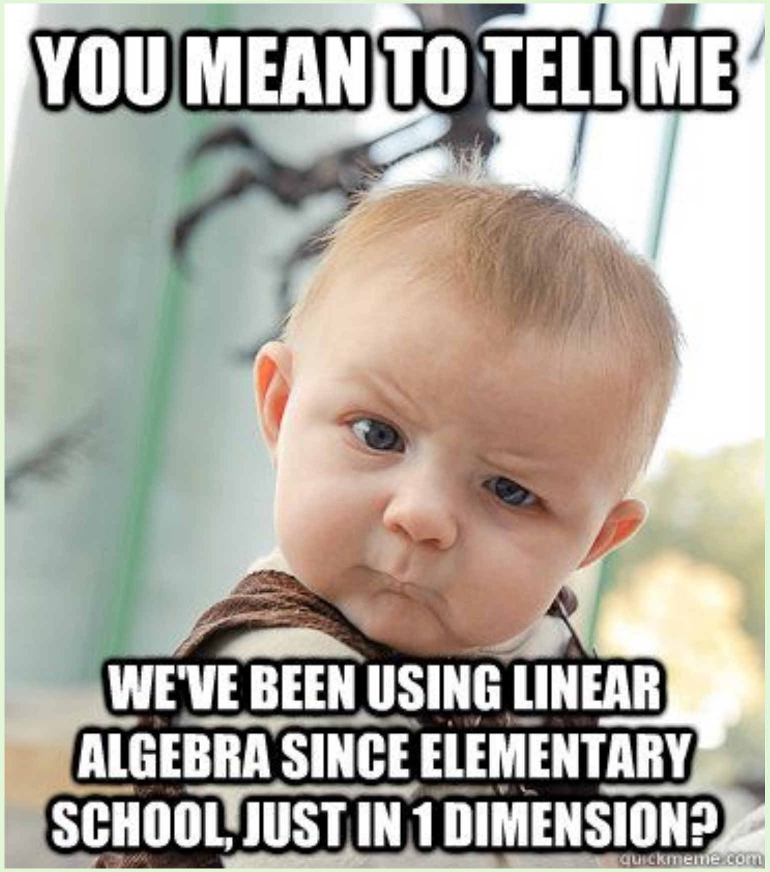
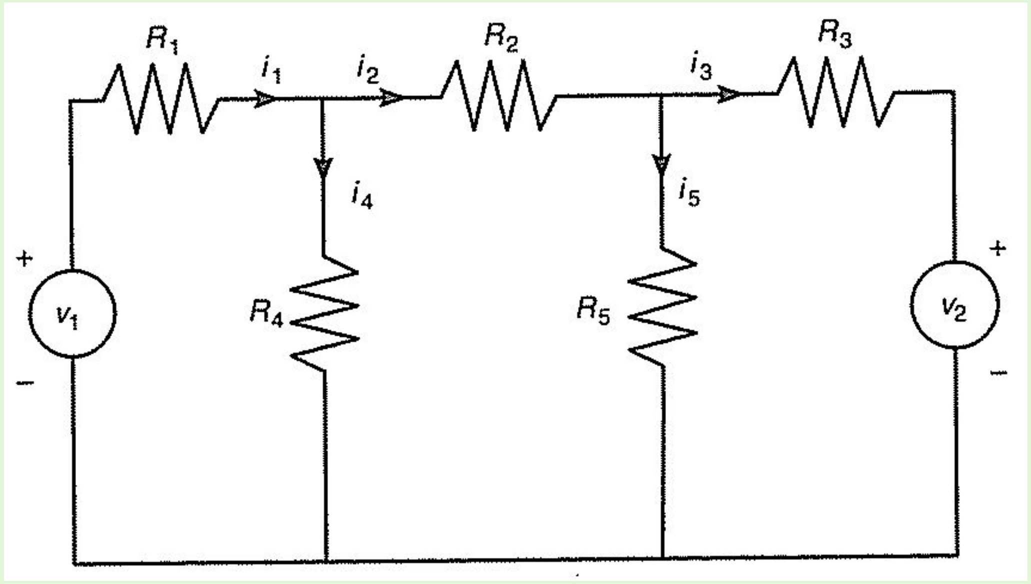
Lecture 17
Linear Algebra I


YOU MEAN TO TELL ME
WE'VE BEEN USING LINEAR ALGEBRA SINCE ELEMENTARY SCHOOL, JUST IN 1 DIMENSION?
quickmeme.com

Outline

- Motivation
  - Linear equations, Linear systems
- Vectors
  - Length, Dot Product
- Matrices
  - Matrix Vector Product, Matrix Multiplication
- Solving systems of linear equations

Linear equations – circuit modeling



Equations for voltage and current are linear

$$-v_1 + R_1 i_1 + R_4 i_4 = 0$$
$$-R_4 i_4 + R_2 i_2 + R_5 i_5 = 0$$
$$-R_5 i_5 + R_3 i_3 + v_2 = 0$$

If you know the voltages, the currents can be obtained by solving the following linear system

$$(R_1 + R_4) i_1 - R_4 i_2 = v_1$$
$$-R_4 i_1 + (R_2 + R_4 + R_5) i_2 - R_5 i_3 = 0$$
$$R_5 i_2 - (R_3 + R_5) i_3 = v_2$$

where the unknown are the currents (*i*’s)

Linear Equations

- y = mx+b is a linear function
- Setting mx + b = c is a linear equation
- A “system” of equations contains multiple equations

$$2x_1 + 9x_2 = 5$$

$$3x_1 - 4x_2 = 7$$

Solving a system of equations involves finding a set values that allow all the equations to hold.

- **This is not always possible**

Systems of Equations

Examples

Linear Equations: $y = mx + b$

- One solution
- No solution
- Infinite number of solutions

Disks in the plane (non-linear): (x - x0)^2 + (x - y0)^2 = r0

- What are the possibilities?

Linear equations in Matrix Form

Example of linear equations

$$2x_1 + 9x_2 = 5$$

$$3x_1 - 4x_2 = 7$$

Can be written in matrix form

$$\begin{bmatrix} 2 & 9 \\ 3 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 7 \end{bmatrix}$$

Or more symbolically as:

$$Ax = y$$

Where

$$A = \begin{bmatrix} 2 & 9 \\ 3 & -4 \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad y = \begin{bmatrix} 5 \\ 7 \end{bmatrix}$$

Linear equations in Matrix Form

m linear equations with n variables:

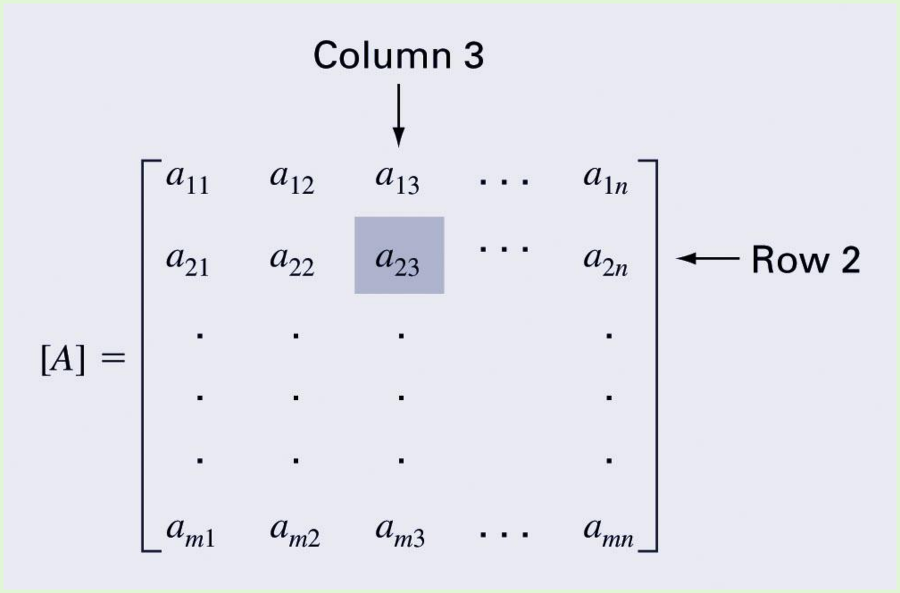
$$\begin{array}{ccccccc} y_1 & = & a_{11}x_1 & + & a_{12}x_2 & + \cdots + & a_{1n}x_n \\ y_2 & = & a_{21}x_1 & + & a_{22}x_2 & + \cdots + & a_{2n}x_n \\ & \vdots & & & & & \\ y_m & = & a_{m1}x_1 & + & a_{m2}x_2 & + \cdots + & a_{mn}x_n \end{array}$$

Can be written in matrix form $y = Ax$ where

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \quad A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Matrices - Review

- A *matrix* consists of a rectangular array of elements represented by a single symbol e.g., A
- An individual entry of a matrix is an *element* e.g., a_{23} , can also write A_{23}



$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & \boxed{a_{23}} & \cdots & a_{2n} \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}$$

Review (cont)

- A horizontal set of elements is called a ***row*** and a vertical set of elements is called a ***column***.
- The first subscript of an element indicates the row while the second indicates the column.
- The size of a matrix is given as *m* rows by *n* columns, or simply *m* by *n* (or *m* x *n*).
- 1 x *n* matrices are ***row vectors***.
- *m* x 1 matrices are ***column vectors.***

Matrix Operations

- isequal(A,B)

  Two matrices are considered equal if and only if every element in the first matrix is equal to every corresponding element in the second. This means the two matrices must be the same size.

- A+B, A-B

  Matrix addition and subtraction are performed by adding or subtracting the corresponding elements. This requires that the two matrices be the same size.

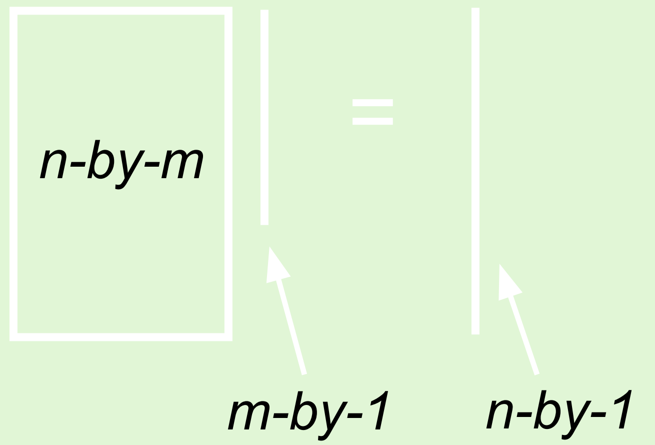
- A*c

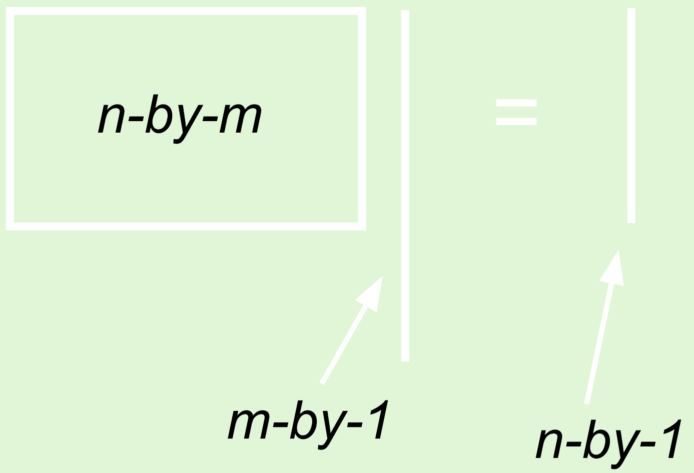
  Scalar matrix multiplication is performed by multiplying each element by the same scalar

Matrix-Vector multiplication

If *A* is an *n-by-m* matrix, and *x* is an *m-by-1* vector, then the "product *Ax*" is a *n-by-1* vector, whose i'th component is

$$\left(Ax\right)_i = \sum_{j=1}^{m} A_{ij} x_j$$



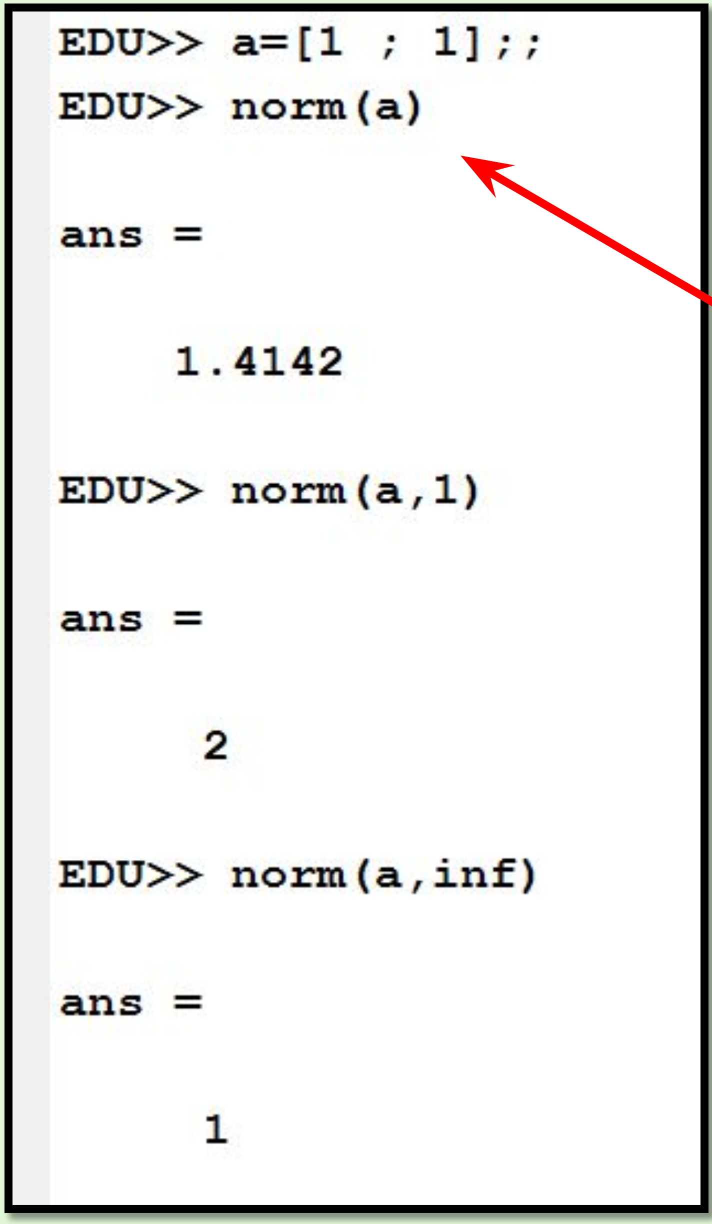


Ax as linear combination of columns of A

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & \cdots & A_{1m} \\ A_{21} & A_{22} & A_{23} & \cdots & A_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & A_{n3} & \cdots & A_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1m}x_m \\ A_{21}x_1 + A_{22}x_2 + \cdots + A_{2m}x_m \\ \vdots \\ A_{n1}x_1 + A_{n2}x_2 + \cdots + A_{nm}x_m \end{bmatrix}$$

$$\begin{bmatrix} A_{11} \\ A_{21} \\ \vdots \\ A_{n1} \end{bmatrix} [x_1] + \begin{bmatrix} A_{12} \\ A_{22} \\ \vdots \\ A_{n2} \end{bmatrix} [x_2] + \begin{bmatrix} A_{13} \\ A_{23} \\ \vdots \\ A_{n3} \end{bmatrix} [x_3] + \cdots + \begin{bmatrix} A_{1m} \\ A_{2m} \\ \vdots \\ A_{nm} \end{bmatrix} [x_m]$$

Example norms



```
EDU>> a=[1 ; 1];;
EDU>> norm(a)

ans =

    1.4142

EDU>> norm(a,1)

ans =

     2

EDU>> norm(a,inf)

ans =

     1
```

$$||a||_2 = \sqrt{\sum_{i=1}^{n} a_i^2}$$

By default the L2 (Euclidian) norm

$$||a||_1 = \sum_{i=1}^{n} |a_i|$$

$$||a||_\infty = \max_{i=1\cdots n} |a_i|$$

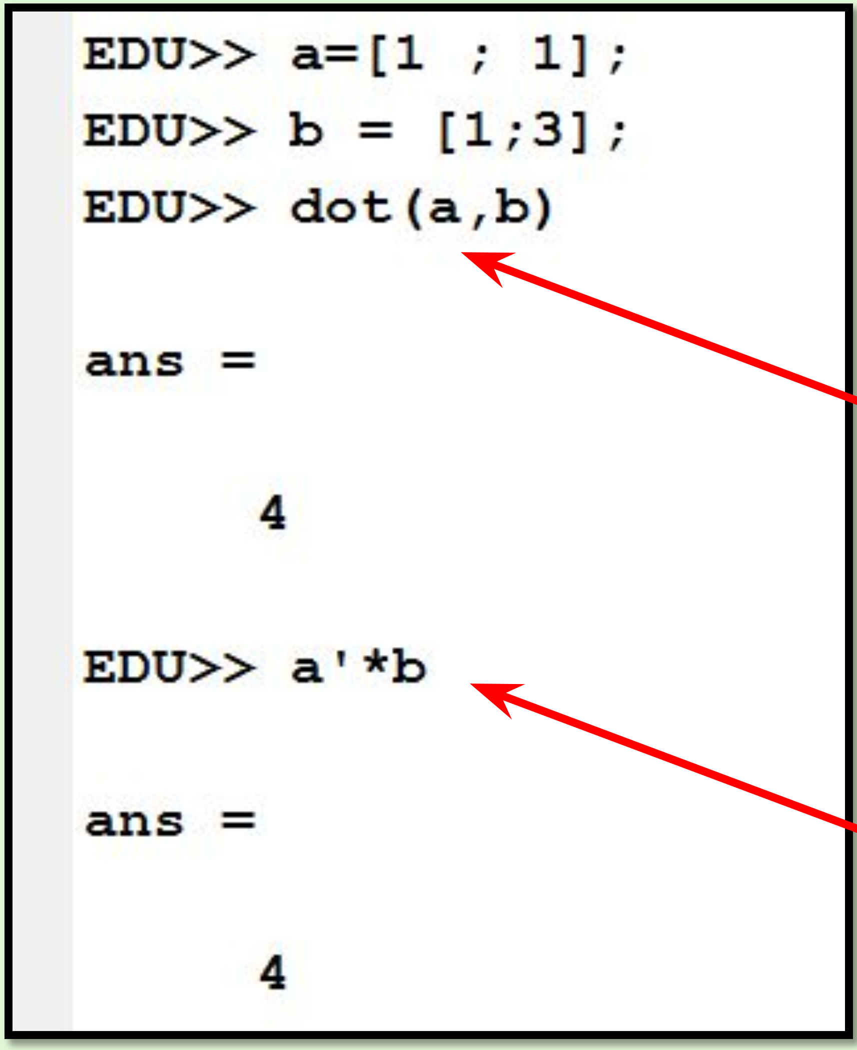
Dot Product - Definition

- The ***dot product*** between two vectors is the sum of their element-wise products.
- For x and y of length n

$$x \cdot y = \sum_{k=1}^{n} x_i y_i$$

Dot product in MATLAB

The dot product operation on vectors:



```
EDU>> a=[1 ; 1];
EDU>> b = [1;3];
EDU>> dot(a,b)

ans =

     4

EDU>> a'*b

ans =

     4
```

Use dot command directly

Or transpose a and then matrix multiply by b (more about this in a moment).

Dot Product – Geometric interpretation

The Dot Product equivalent to

$$x \cdot y = \|x\| \|y\| \cos(\theta)$$

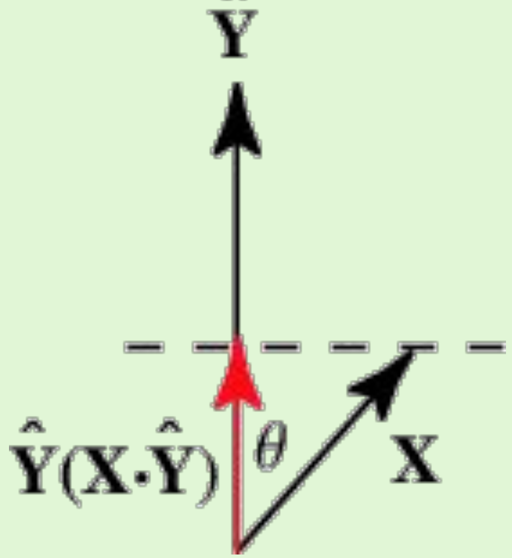
where $\|x\|$ and $\|y\|$ are the lengths (L2 norm) of x and y, and θ is the angle between vectors x and y.

Remark: Derivation can be done in 2D: express x and y in polar form, and take the usual dot product

Dot Product – Geometric interpretation

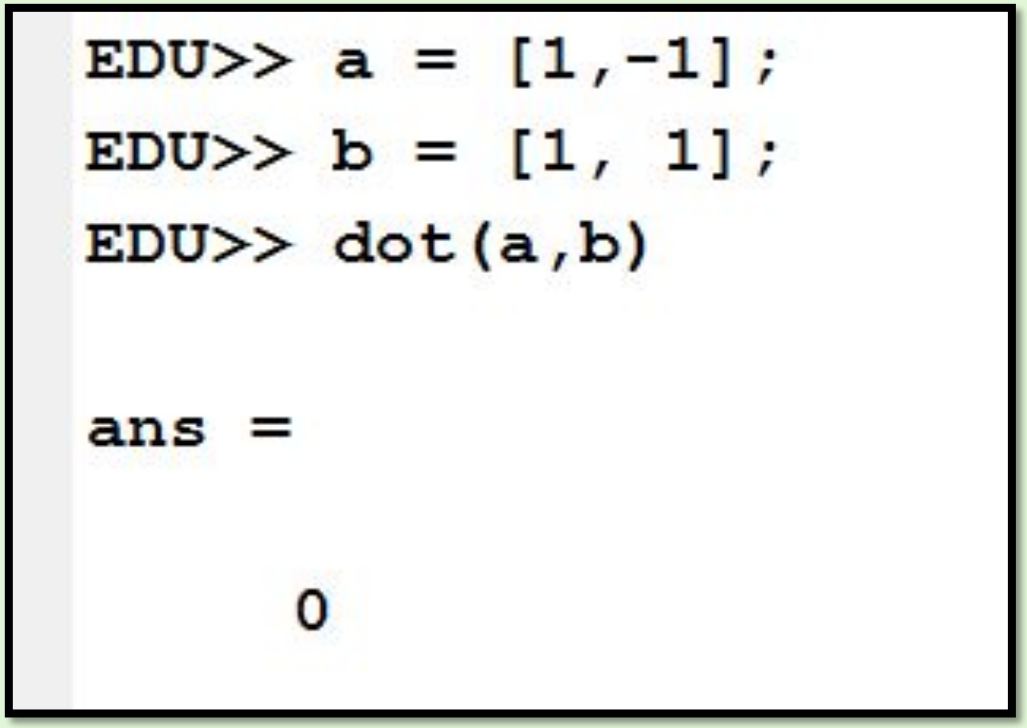
When applied to a ***unit vector*** the dot product is the length of the projection onto that unit vector, when the two vectors are placed tail to tail.

$$X \cdot \hat{Y} = \|X\| \cos(\theta)$$



Dot product in MATLAB

Orthogonality of two vectors



```
EDU>> a = [1,-1];
EDU>> b = [1, 1];
EDU>> dot(a,b)


ans =


     0
```

Dot product of two vectors is zero implies they are orthogonal

Ax can also be found via dot products

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} & \cdots & A_{1m} \\ A_{21} & A_{22} & A_{23} & \cdots & A_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{n1} & A_{n2} & A_{n3} & \cdots & A_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1m}x_m \\ A_{21}x_1 + A_{22}x_2 + \cdots + A_{2m}x_m \\ \vdots \\ A_{n1}x_1 + A_{n2}x_2 + \cdots + A_{nm}x_m \end{bmatrix}$$

$$\begin{bmatrix} A_{11} \\ A_{21} \\ \vdots \\ A_{n1} \end{bmatrix} [x_1] + \begin{bmatrix} A_{12} \\ A_{22} \\ \vdots \\ A_{n2} \end{bmatrix} [x_2] + \begin{bmatrix} A_{13} \\ A_{23} \\ \vdots \\ A_{n3} \end{bmatrix} [x_3] + \cdots + \begin{bmatrix} A_{1m} \\ A_{2m} \\ \vdots \\ A_{nm} \end{bmatrix} [x_m]$$

Ax as a collection of dot products

If *A* is an *n-by-m* matrix, and *x* is an *m-by-1* vector, then the "product *Ax*" is a *n-by-1* vector, whose i'th component is

$$Ax = \begin{bmatrix} (Ax)_1 \\ (Ax)_2 \\ \vdots \\ (Ax)_n \end{bmatrix} = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1m}x_m \\ A_{21}x_1 + A_{22}x_2 + \cdots + A_{2m}x_m \\ \vdots \\ A_{n1}x_1 + A_{n2}x_2 + \cdots + A_{nm}x_m \end{bmatrix}$$

$$(Ax)_i = \sum_{j=1}^{m} A_{ij}x_j = A(i,:) * x = dot(A(i,:), x)$$

Vector norms

Length of a vector is measured using a ***norm*** function

A norm p is required to satisfy the following axioms, for all scalars a and vectors u, v of the same length,

1) $p(av) = |a|p(v)$

2) $p(u + v) \leq p(u) + p(v)$ (triangle inequality)

3) If $p(v) = 0$, then v is the zero vector

There a variety of ways to measure length (i.e. multiple ways one can define a norm).

Vector norms

Length of a vector is measured using a ***norm*** function

A norm p is required to satisfy the following axioms, for all scalars a and vectors u, v of the same length,

1) $p(av) = |a|p(v)$

2) $p(u + v) \le p(u) + p(v)$ (triangle inequality)

3) If $p(v) = 0$, then v is the zero vector

- Note: $p(-v) = |-1|p(v)$, so $p(v) \ge 0$ (positivity)

Norms of vectors

If *v* is an m-by-1 column (or row) vector, the L2 "norm of *v*" is defined as

$$\|v\| := \sqrt{\sum_{k=1}^{m} |v_k|^2}$$

Symbol to denote "norm of *v*"

Square-root of sum-of-squares of components, generalizing Pythagorean's theorem

The norm of a vector is a measure of its length.

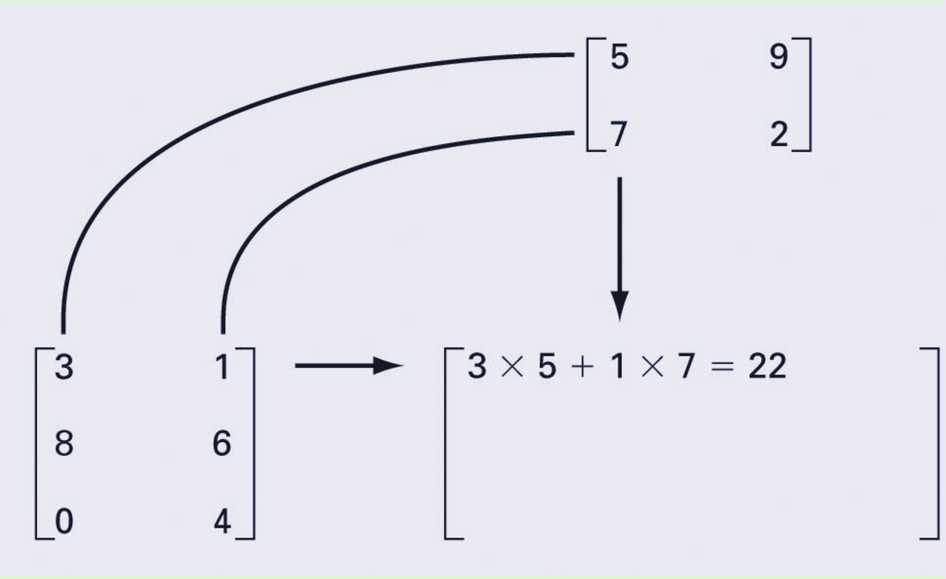
Helpful property:

$\|v + w\| \leq \|v\| + \|w\|$ (Triangle inequality)

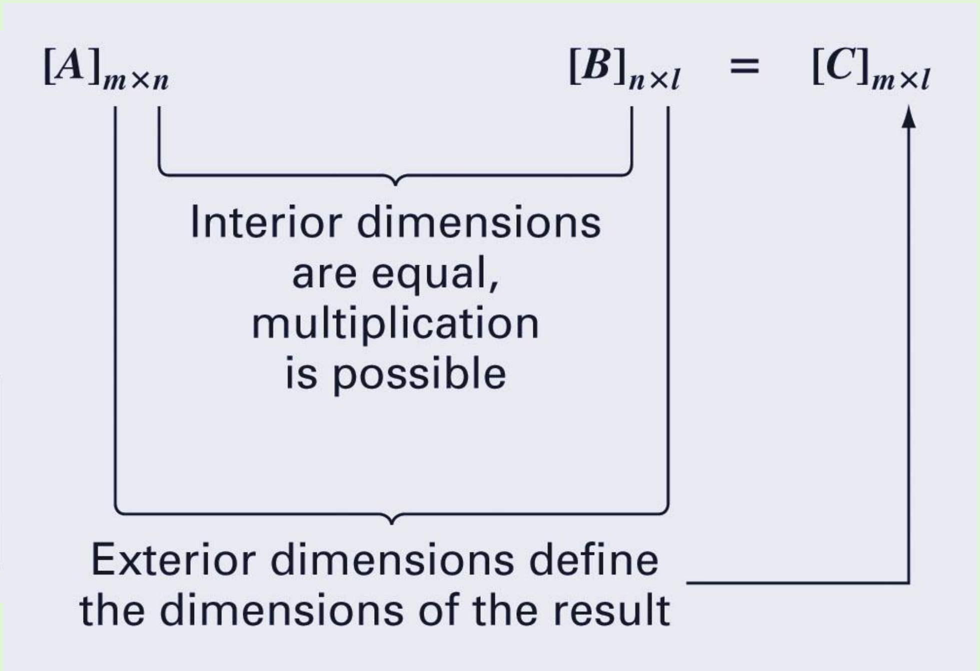
Matrix Multiplication, C=A*B

- The elements in the matrix C that results from multiplying matrices A and B are calculated using:

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj}$$



$\begin{bmatrix} 5 & 9 \\ 7 & 2 \end{bmatrix}$
$\begin{bmatrix} 3 & 1 \\ 8 & 6 \\ 0 & 4 \end{bmatrix}$
$3 \times 5 + 1 \times 7 = 22$



$[A]_{m\times n}$ $[B]_{n\times l}$ = $[C]_{m\times l}$
Interior dimensions are equal, multiplication is possible
Exterior dimensions define the dimensions of the result

Matrix Math

- Does order of evaluation matter?
    - Is $(A+B)+C = A+(B+C)$?

Matrix Math

- Does order of evaluation matter?
    - Is (AB)C = A(BC)?

Matrix Math

- Does order of evaluation matter?
  - Is AB = BA?

Special Matrices

- Matrices where *m=n* are called ***square matrices***.
- There are a number of special forms of square matrices:

Symmetric

$$[A]=\begin{bmatrix}5 & 1 & 2\\ 1 & 3 & 7\\ 2 & 7 & 8\end{bmatrix}$$

Diagonal

$$[A]=\begin{bmatrix}a_{11} & & \\ & a_{22} & \\ & & a_{33}\end{bmatrix}$$

Identity

$$[A]=\begin{bmatrix}1 & & \\ & 1 & \\ & & 1\end{bmatrix}$$

Upper Triangular

$$[A]=\begin{bmatrix}a_{11} & a_{12} & a_{13}\\ & a_{22} & a_{23}\\ & & a_{33}\end{bmatrix}$$

Lower Triangular

$$[A]=\begin{bmatrix}a_{11} & & \\ a_{21} & a_{22} & \\ a_{31} & a_{32} & a_{33}\end{bmatrix}$$

Banded

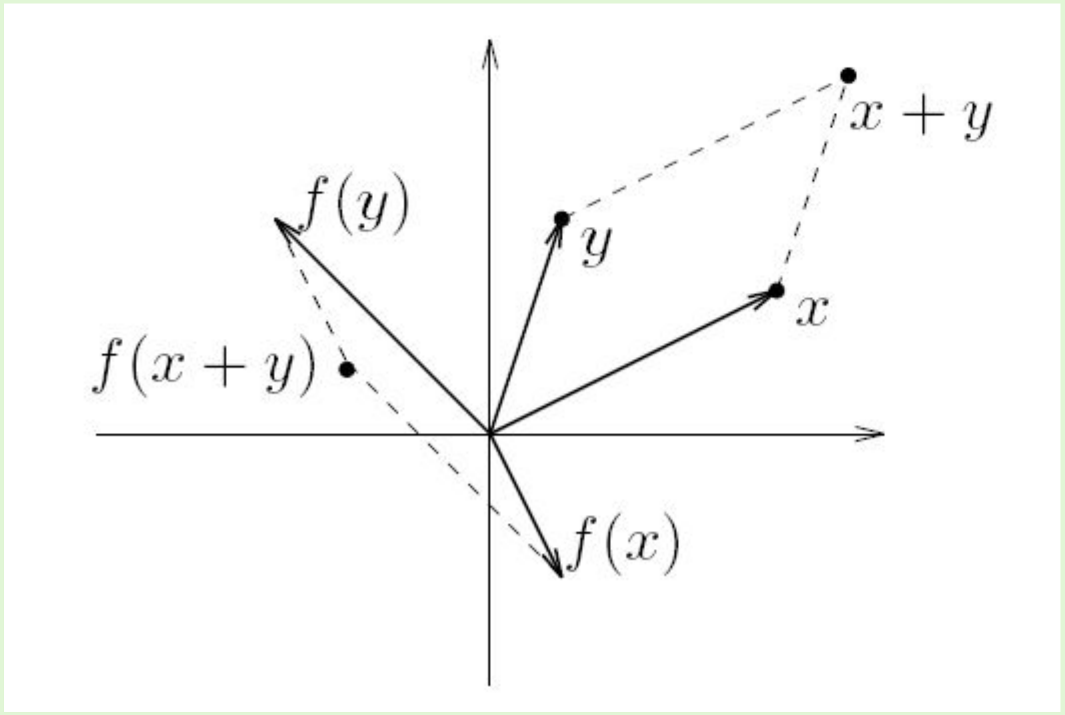
$$[A]=\begin{bmatrix}a_{11} & a_{12} & & \\ a_{21} & a_{22} & a_{23} & \\ & a_{32} & a_{33} & a_{34}\\ & & a_{43} & a_{44}\end{bmatrix}$$

What is a linear function?

Let $f : \mathbf{R}^n \longrightarrow \mathbf{R}^m$ unction. It is said to be linear if

- $f(x+y) = f(x) + f(y), \ \forall x, y \in \mathbf{R}^n$
- $f(\alpha x) = \alpha f(x), \ \forall x \in \mathbf{R}^n \ \forall \alpha \in \mathbf{R}$

This is sometimes called superposition



Matrix representation of a linear function

- consider function $f : \mathbf{R}^n \to \mathbf{R}^m$ given by $f(x) = Ax$, where $A \in \mathbf{R}^{m\times n}$
- matrix multiplication function f is linear
- **converse** is true: **any** linear function $f : \mathbf{R}^n \to \mathbf{R}^m$ can be written as $f(x) = Ax$ for some $A \in \mathbf{R}^{m\times n}$
- representation via matrix multiplication is unique: for any linear function f there is only one matrix A for which $f(x) = Ax$ for all x
- $y = Ax$ is a concrete representation of a generic linear function

Interpretation of

$$y = Ax$$

- y is measurement or observation; x is unknown to be determined
- x is 'input' or 'action'; y is 'output' or 'result'
- $y = Ax$ defines a function or transformation that maps $x \in \mathbf{R}^n$ into $y \in \mathbf{R}^m$

Interpretation of a_{ij} in $y = Ax$

$$y_i = \sum_{j=1}^{n} a_{ij} x_j$$

a_{ij} is *gain factor* from jth input (x_j) to ith output (y_i)

thus, *e.g.*,

- ith *row* of A concerns ith *output*
- jth *column* of A concerns jth *input*
- $a_{27} = 0$ means 2nd output (y_2) doesn't depend on 7th input (x_7)
- $|a_{31}| \gg |a_{3j}|$ for $j \neq 1$ means y_3 depends mainly on x_1

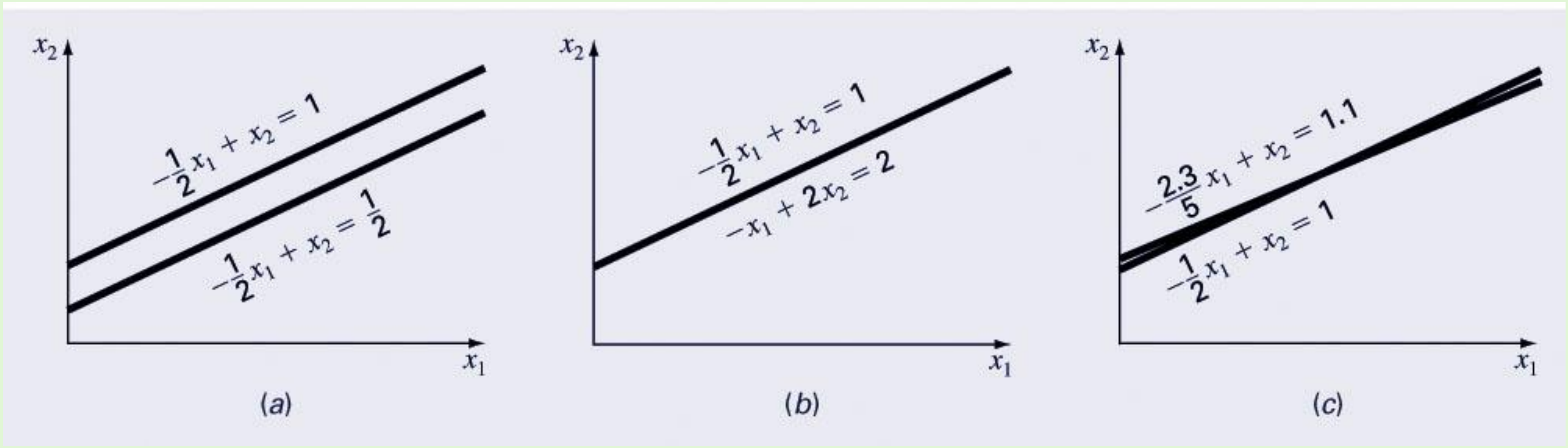
Solving y=Ax

Graphing the equations can demonstrate

a) No solution exists

b) Infinite solutions exist

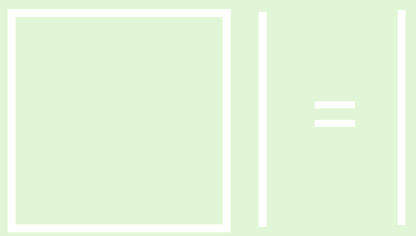
c) System is ill-conditioned!!!



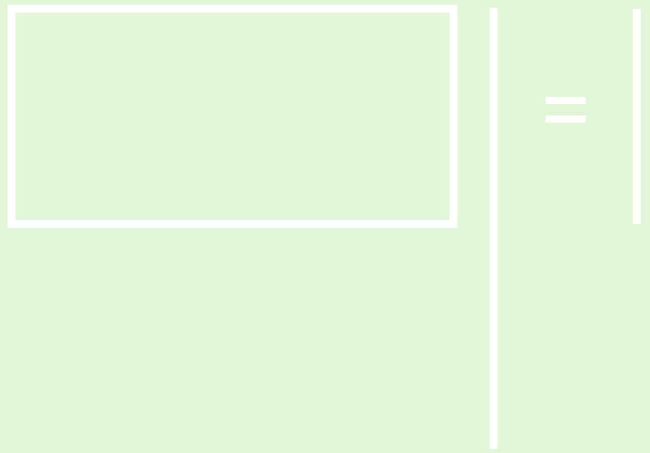
x_2
$-\frac{1}{2}x_1 + x_2 = 1$
$-\frac{1}{2}x_1 + x_2 = \frac{1}{2}$
x_1
(a)
x_2
$-\frac{1}{2}x_1 + x_2 = 1$
$-x_1 + 2x_2 = 2$
x_1
(b)
x_2
$-\frac{2.3}{5}x_1 + x_2 = 1.1$
$-\frac{1}{2}x_1 + x_2 = 1$
x_1
(c)

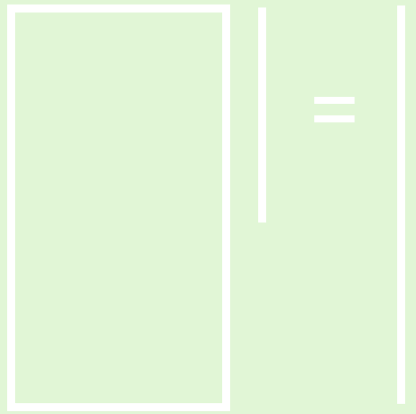
Linear equations

For the equation *Ax=y*, there are 3 distinct cases



Square, equal number of unknowns and equations



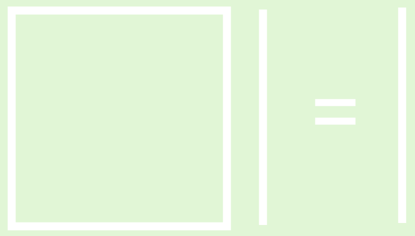


Overdetermined: fewer unknowns than equations

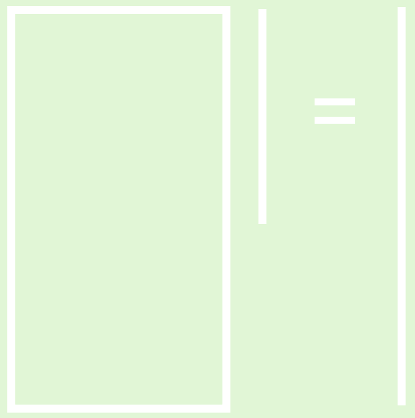
Underdetermined: more unknowns than equations

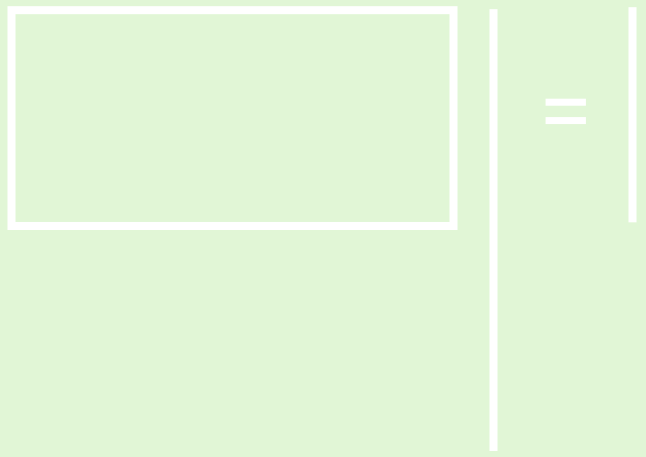
Types of solutions with “random” data

“Generally” the following observations would hold



One solution (eg., 2 lines intersect at one point)



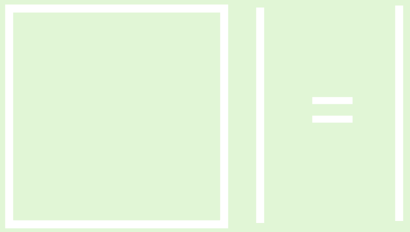


Infinite solutions (eg., 2 planes intersect at many points)

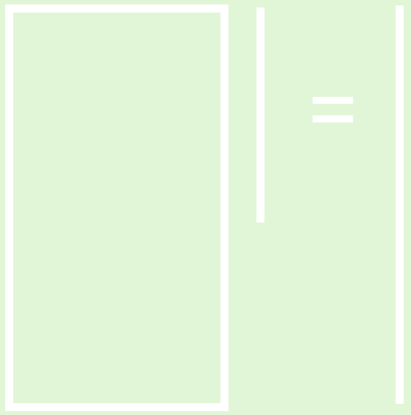
No solutions (eg., 3 lines don’t intersect at a point)

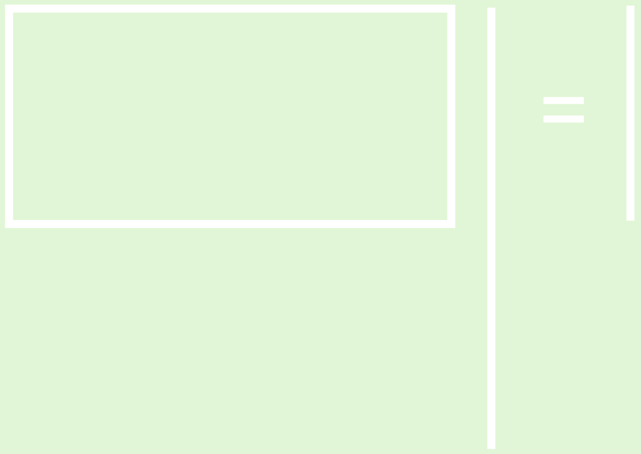
Linear equations

But other things can happen.   For example:



<u>No solution</u> (2 parallel lines)
<u>Many solutions</u> (2 identical lines)





<u>No solutions</u> (2 parallel planes)

<u>Solutions</u> (3 lines that do intersect at a point)

Problem of inversion

In MATLAB, x = A\b
Attempts to solve matrix equation Ax = b.

Use of \ is due to the fact matrix multiplications is not commutative, i.e.

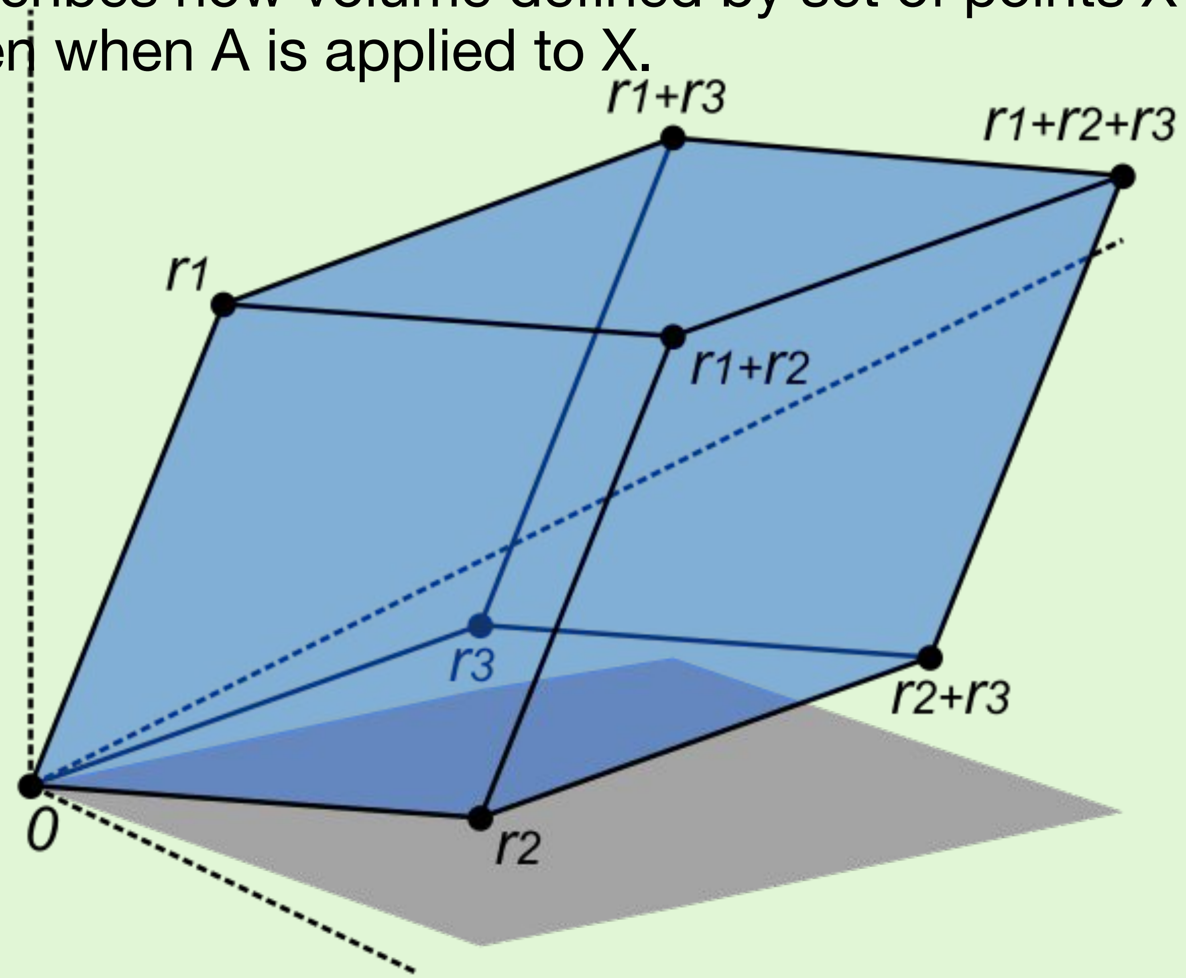
AB = C => A\AB=A\C => B = A\C.

It is not the case B=C/A, i.e. AB/A is not B.

For matrices, the definition of the “inverse”, or “one over” the matrix, has to be defined properly. When does the inverse exist?

The Determinant

- Describes how volume defined by set of points X changes when when A is applied to X.



Determinants (square matrices)

Consider a 2 x 2 matrix

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

The determinant of a 2 x 2 matrix is

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{21}a_{12}$$

Finding the determinant of a general n x n square matrix requires evaluation of a complicated polynomial of the coefficients of the matrix, but there is a simple recursive approach.

If the determinant is non-zero, the matrix can be inverted and unique solution exists for Ax=b.

If the determinant is zero, the matrix cannot be inverted, there can be either 0 or an infinite number of solutions to Ax=b.

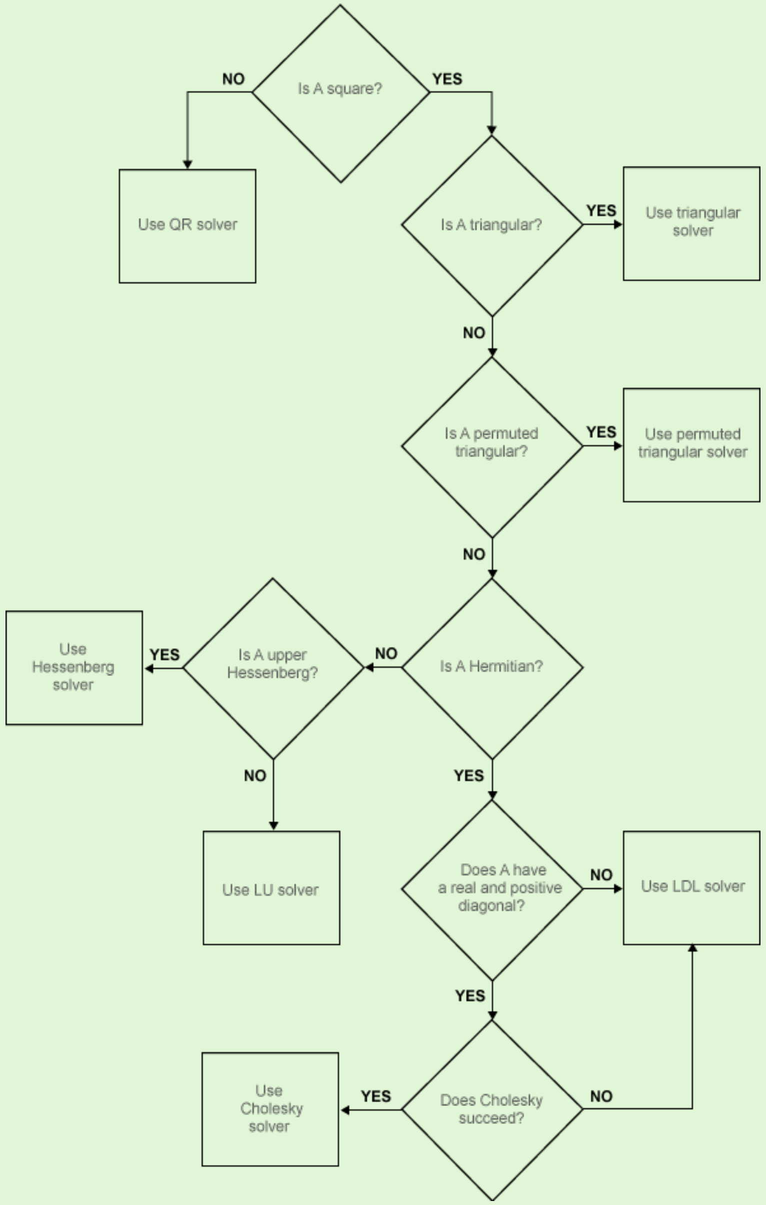
Solving Ax=y

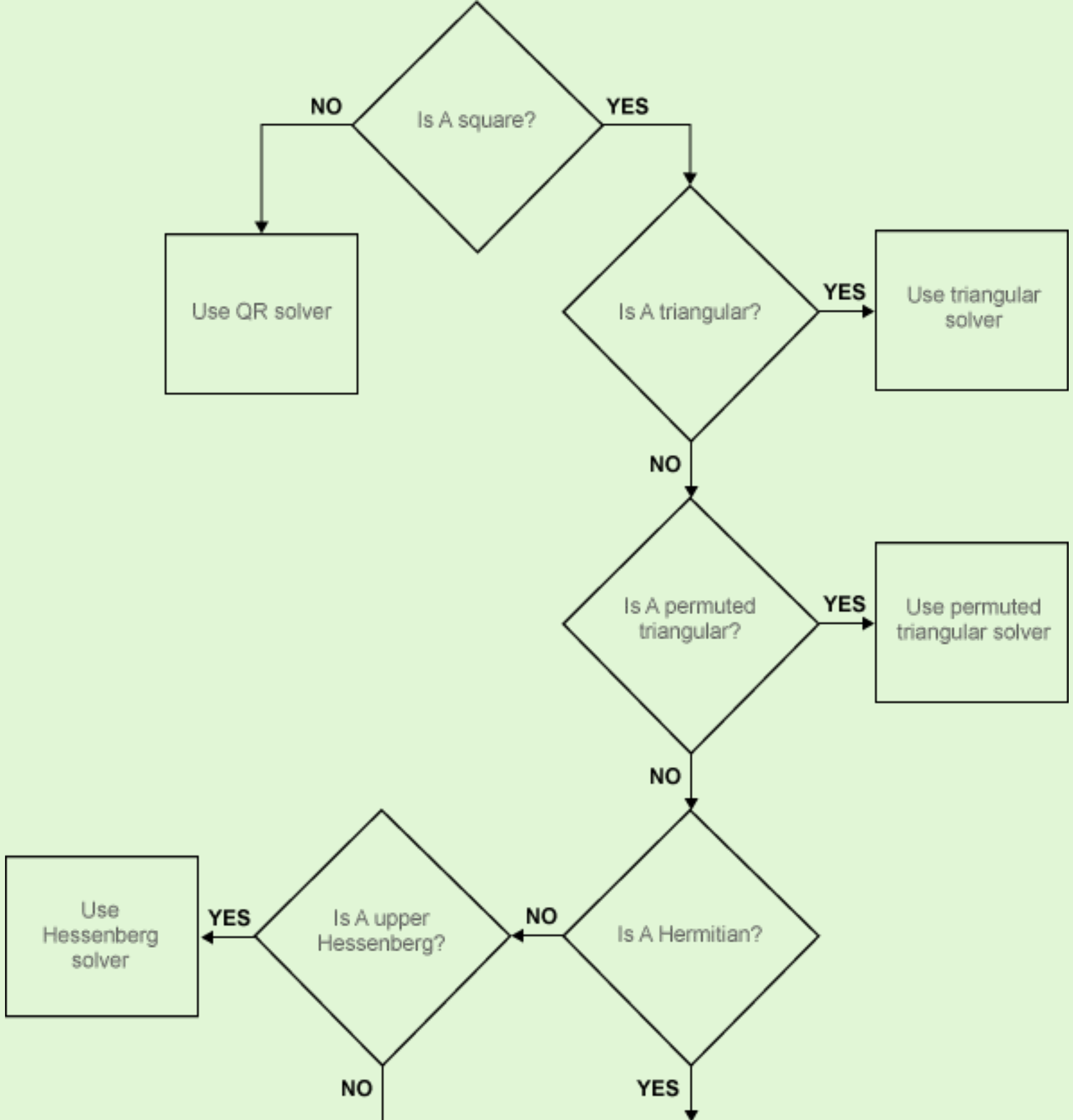
- When A is non-singular (has non-zero determinant) A inverse exists, and one can find x via

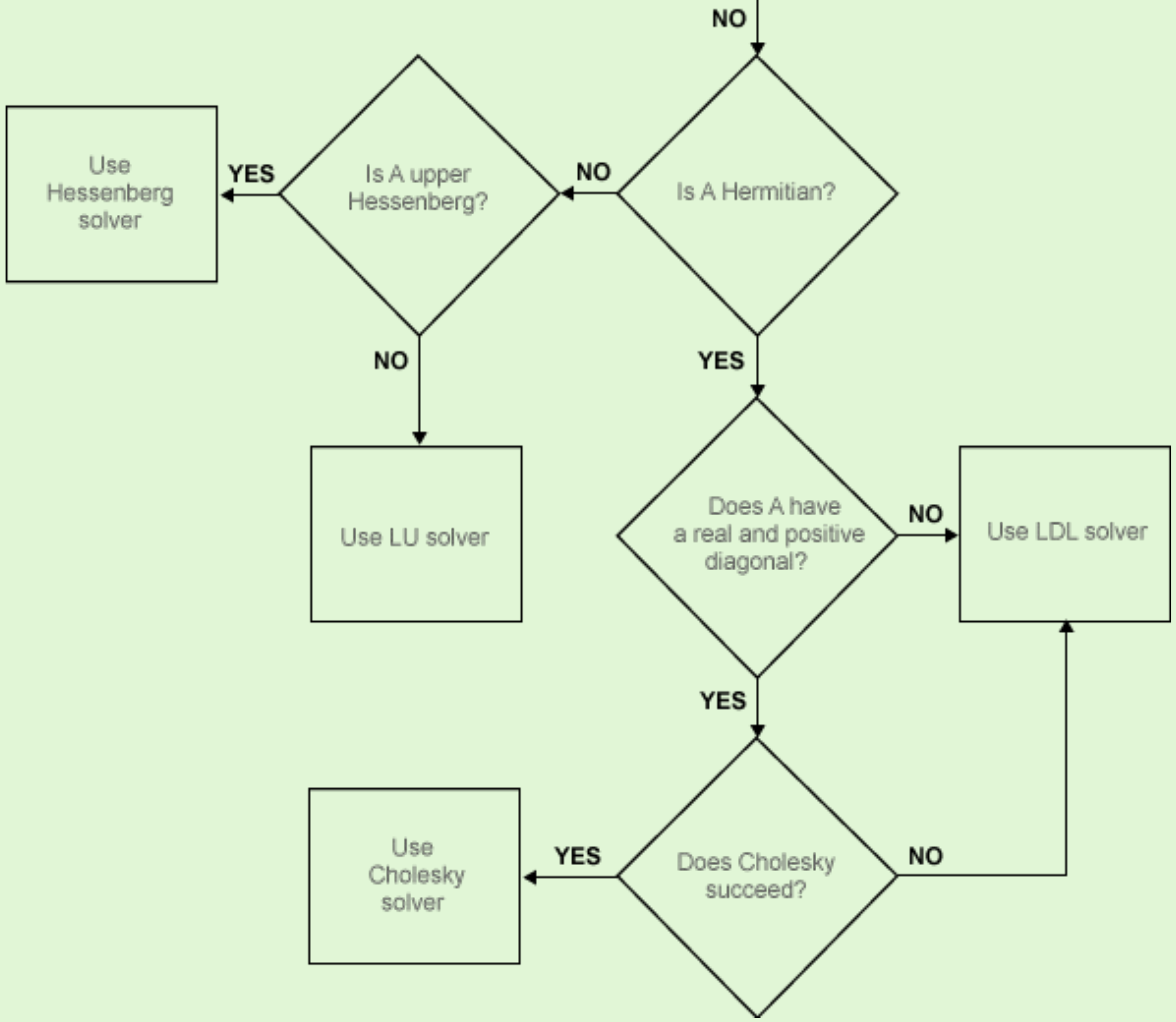
```
x = inv(A)*y
```

- However, depending on A, this is can be computationally inefficient and or less precise then using **`x = A\y`**
- The MATLAB \ operation (called mldivide) takes the form of A into account while trying to solve A\y
- `doc mldivide`

Operation of A\y in MATLAB




Is A square?
NO
YES
Use QR solver
Is A triangular?
YES
Use triangular solver
NO
Is A permuted triangular?
YES
Use permuted triangular solver
NO
Is A Hermitian?
NO
Is A upper Hessenberg?
YES
Use Hessenberg solver
NO
YES


NO
Use Hessenberg solver
YES
Is A upper Hessenberg?
NO
Is A Hermitian?
NO
YES
Use LU solver
Does A have a real and positive diagonal?
NO
Use LDL solver
YES
Use Cholesky solver
YES
Does Cholesky succeed?
NO

Solving Ax=y

- When A is singular (has a zero determinant) A inverse does not exist
- In this case there are either NO solutions or there are an INFINITE number of solutions to Ax = b
- Two key questions

1) How can we tell when no solutions exist?

2) When solutions exists how can we find and represent them?

https://www.mathworks.com/help/matlab/math/systems-of-linear-equations.html

HW Problem 7.1.7

- Linear Indexing

```
For an m-by-n array the linear index of element
(i,j) is i+(j-1)*m
```

HW Problem 7.2.4

- Binary Search

See next slides

Binary Search & BISECTION

Outline: Root Finding

- Key Concept: Search
- Binary Search
- Bisection Method

Searching

Finding a specific entry in a 1-dimensional (eg, column vector, row vector) object.

Brute force approach, scan (potentially) the entire list.

```
function Idx = bfsearch(A,Key)
N = length(A);
Idx=1;
while Idx<=N & A(Idx)~=Key
    Idx = Idx + 1;
end
if Idx==N+1
    Idx = [];  % no match found
end
```

Entire list scanned, no match found

Loop exits if

`Idx>N` or

`A(Idx)==Key`

Match found

Searching in a sorted list

If the list is sorted, it should be easier to search, since checking for a match at any location in the list

- finds the match, or (more likely)
- Splits the list (at that location) into two lists, one to the "left" of the location, and one to the "right"
  - Since the list is sorted, we immediately know which of the two sublists the match must belong to.

Take a sorted list A, of length N.

$$\underbrace{A(1) \le A(2) \le \cdots \le A(k-1) \le A(k)} \le \underbrace{A(k+1) \le \cdots \le A(N)}$$

Match must be over here

$\text{If } A(k) < \text{Key}$

Match must be over here

$\text{Else (ie., } A(k) \ge \text{Key)}$

Searching in a sorted list

Take a sorted list A, of length N.

$$\underbrace{A(1) \le A(2) \le \cdots \le A(k-1) \le A(k)} \le \underbrace{A(k+1) \le \cdots \le A(N)}$$

Match must be over here

$$\text{If } A(k) < \text{Key}$$

Match must be over here

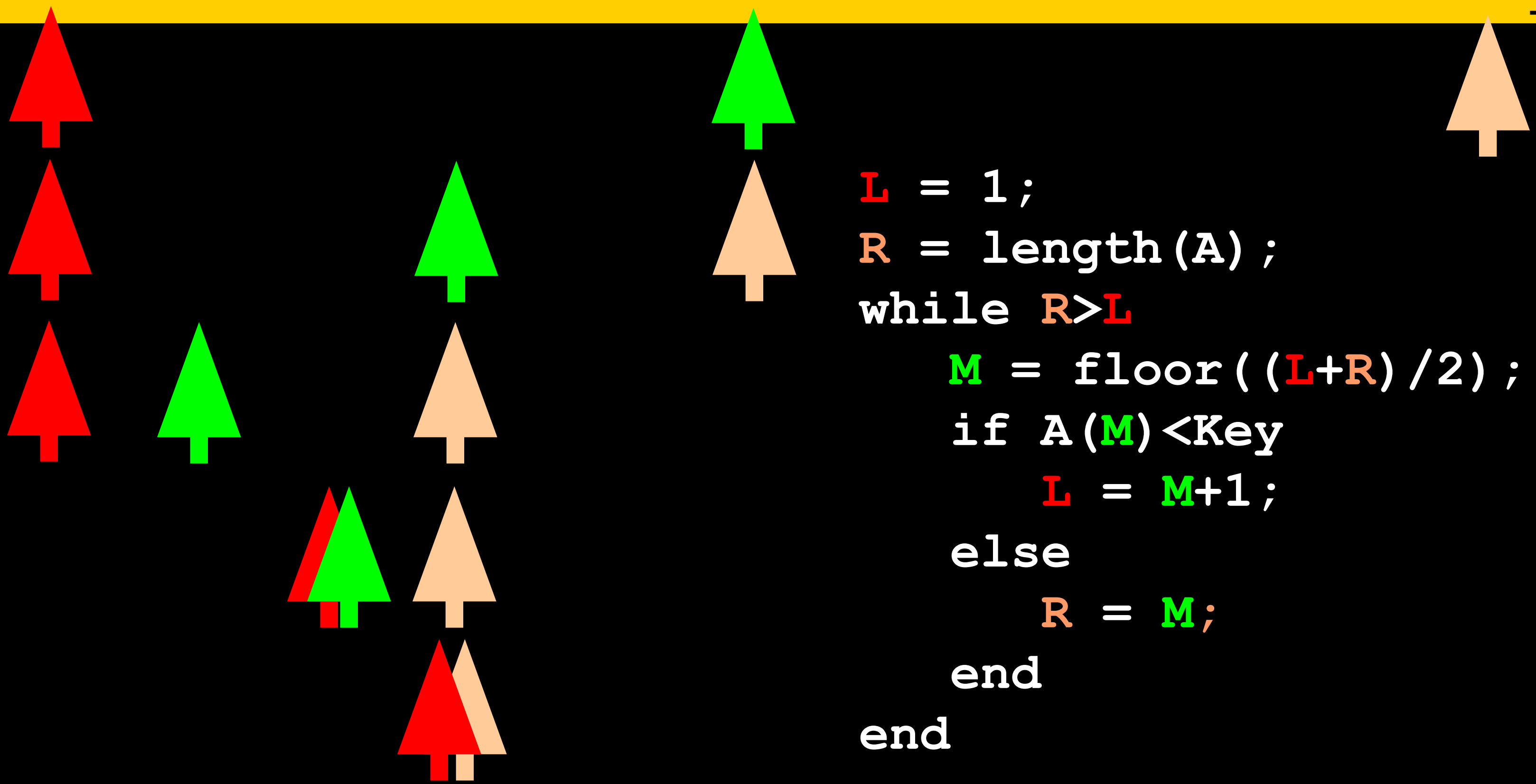
$$\text{Else (ie., } A(k) \ge \text{Key)}$$

Choose k in the “middle”, halving the size of the relevant list each time, until a match is found.

Simple Binary Search Example

Let’s do an example with as below, and Key = 0.1; A

$$\left[-3.2\ \ -1.9\ \ -0.2\ \ 0.1\ \ 1.3\ \ 1.9\ \ 2.1\ \ 3.5\ \ 3.9\ \ 4.0\ \ 4.4\ \ 4.8\ \ 6.0\ \ 9.7\right]$$



```
L = 1;
R = length(A);
while R>L
    M = floor((L+R)/2);
    if A(M)<Key
        L = M+1;
    else
        R = M;
    end
end
```

Motivation for Root Finding: Solving Equations

From linearity, it is easy to solve the equation

$$3x + 7 = 12$$

or even the system of equations

$$\begin{aligned} 2x + 5y &= 12 \\ x - 6y &= 3 \end{aligned}$$

But what about an equation like

$$\tan(x)e^{-x} = 1$$

Finding Roots

From linearity, it is easy to solve the equation

$$3x + 7 - 12 = 0$$

or even the system of equations

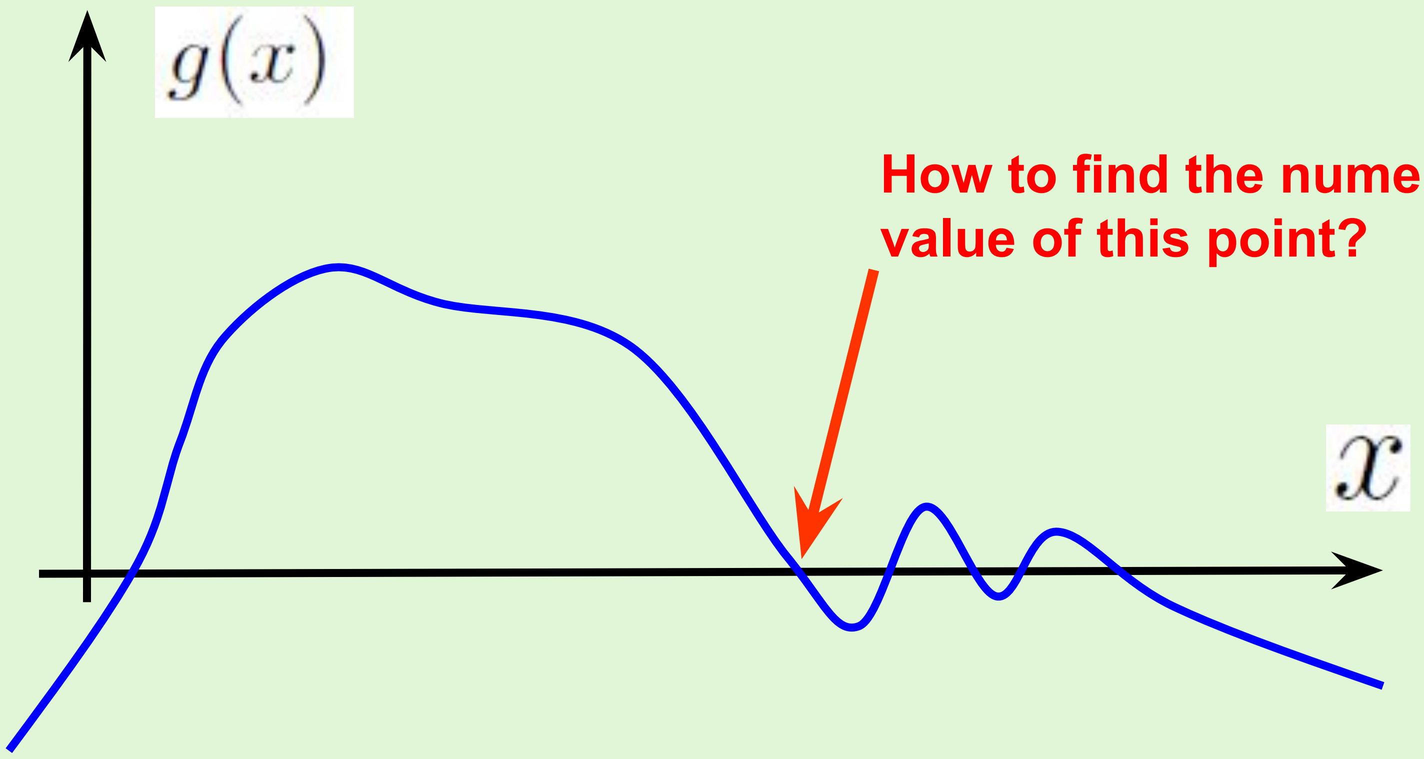
$$\begin{aligned} 2x + 5y - 12 &= 0 \\ x - 6y - 3 &= 0 \end{aligned}$$

But what about an equation like

$$\tan(x)e^{-x} - 1 = 0$$

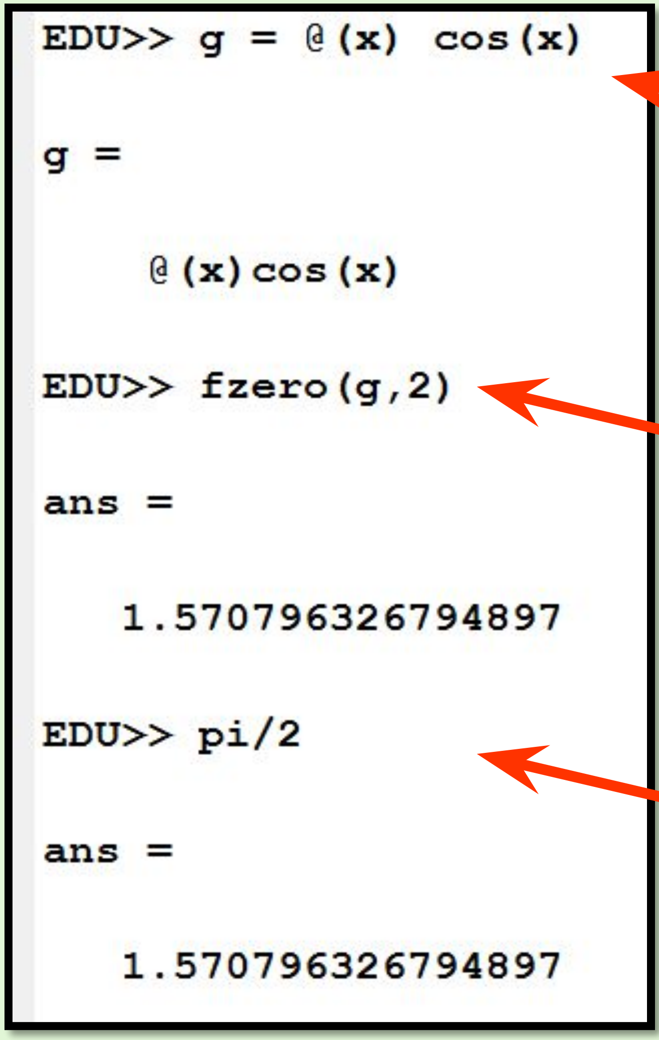
Main problem in root finding

Main problem: finding the root of an equation g(x) = 0, i.e. a point x where the function g is zero



Matlab built in functions

Matlab has a generic function called fzero which finds a zero of a function, based on a guess entered by a user.



```
EDU>> g = @(x) cos(x)

g =

    @(x)cos(x)

EDU>> fzero(g,2)

ans =

   1.57079632679490

EDU>> pi/2

ans =

   1.57079632679490
```

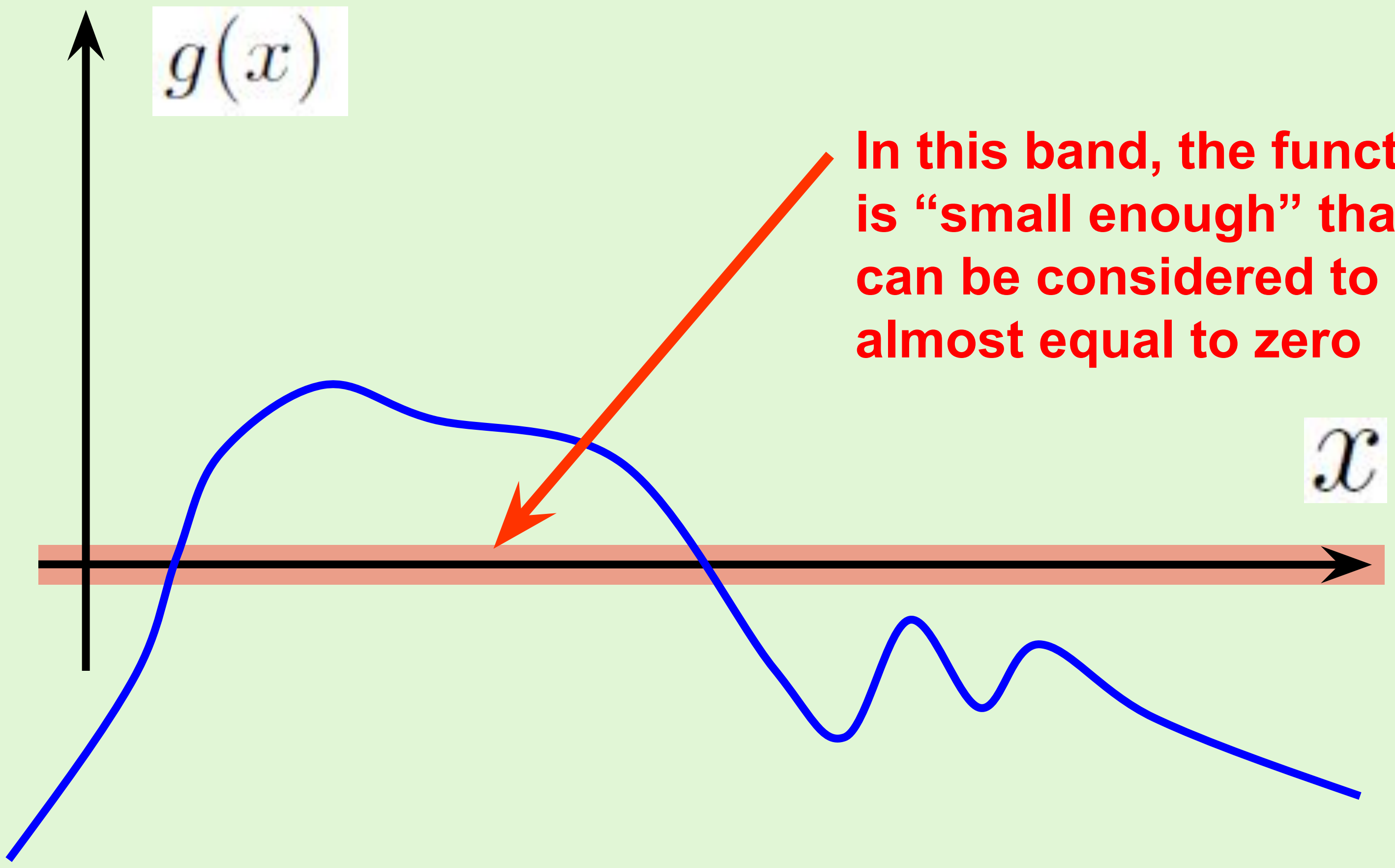
Declare function using function handle

Call fzero, and specify guess

Result is obviously π/2 in this case, as can be checked

What questions are involved in solving g(x)=0?

Root finding algorithms find numerical values such that the function g is "almost" zero, i.e. has small values.

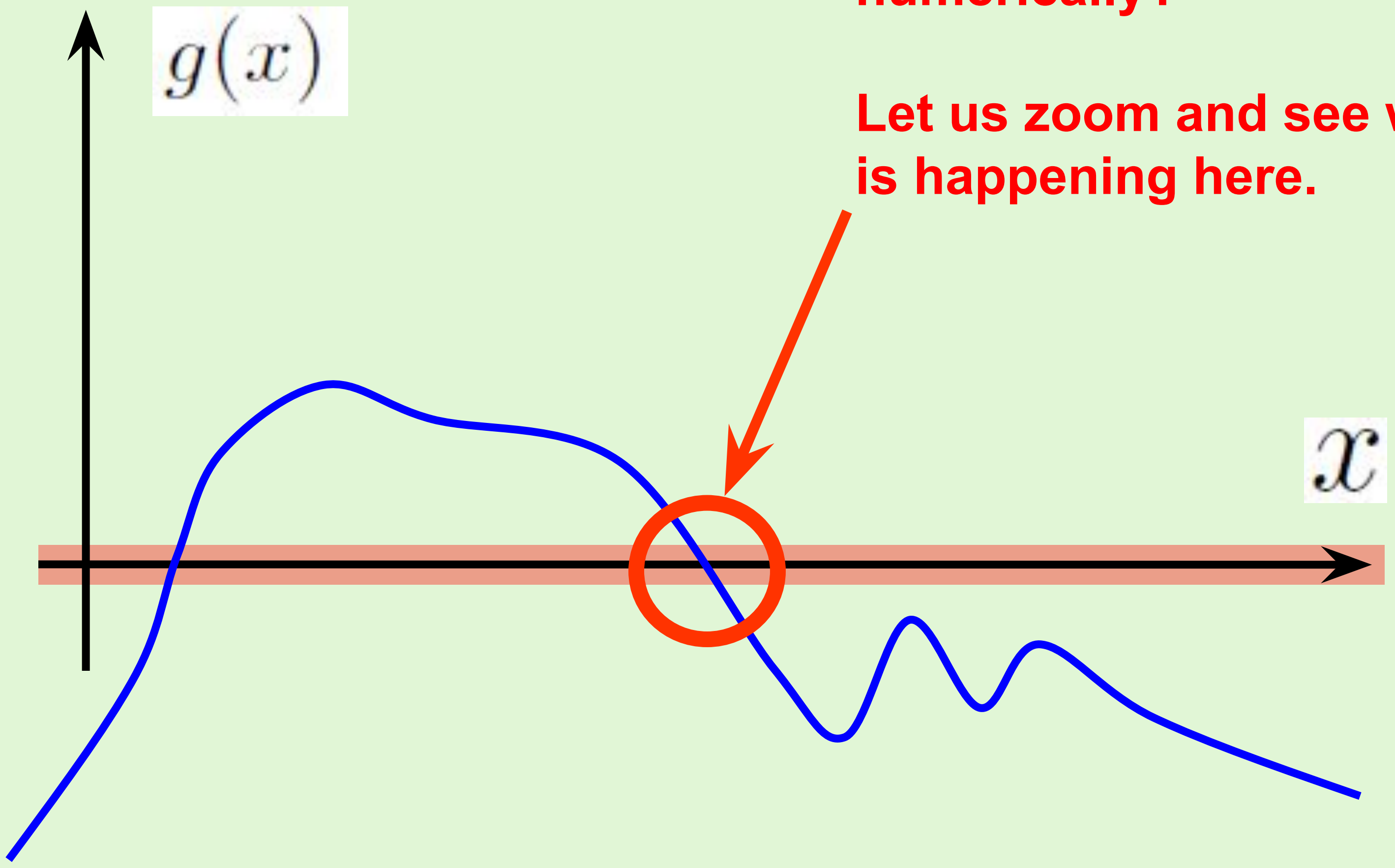

$g(x)$
In this band, the function is "small enough" that it can be considered to be almost equal to zero
x

What questions are involved in solving f(x)=0?

Root finding algorithms find numerical values such that the function g is “almost” zero, i.e. has small values.

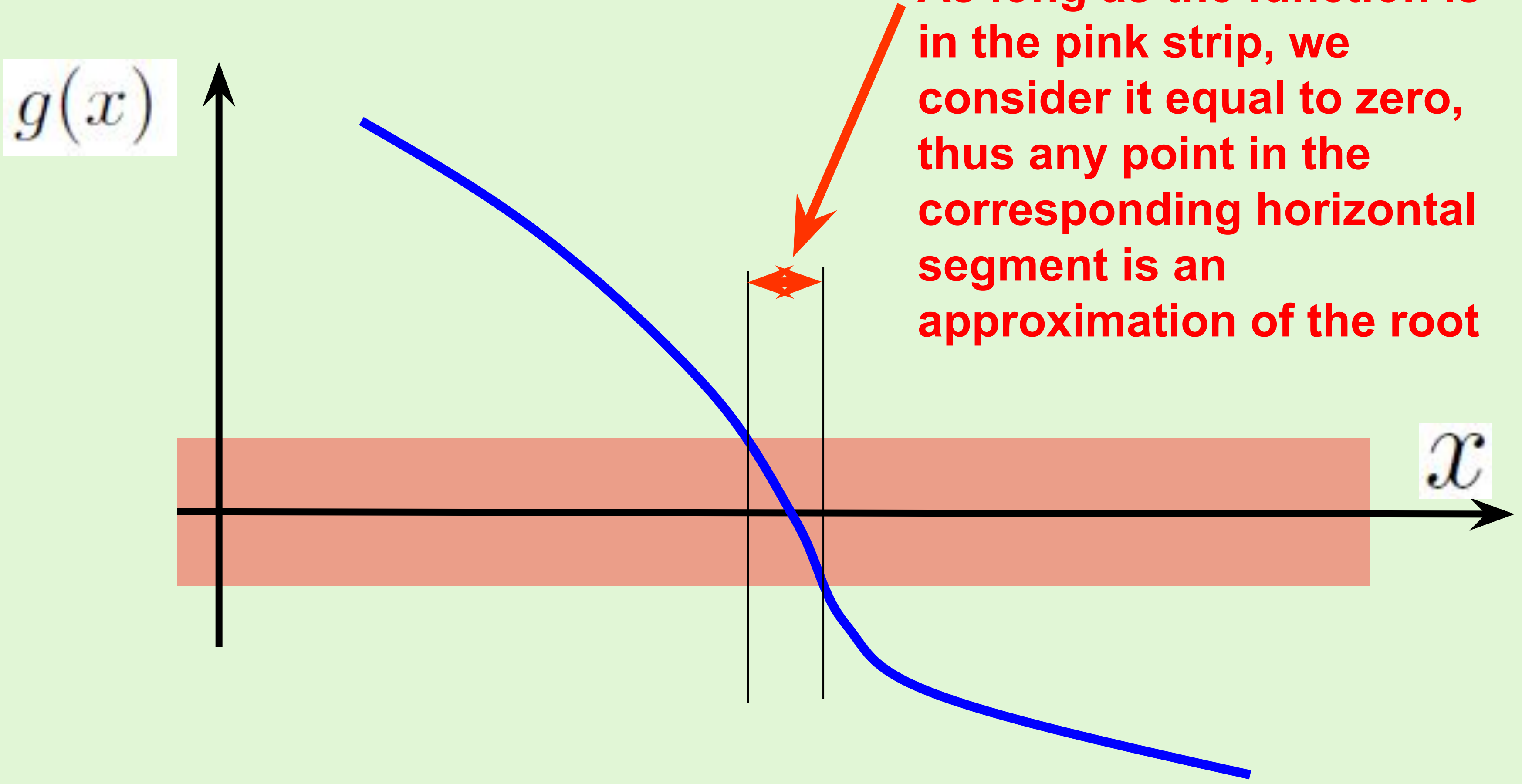
What does this mean numerically?

Let us zoom and see what is happening here.



What questions are involved in solving f(x)=0?

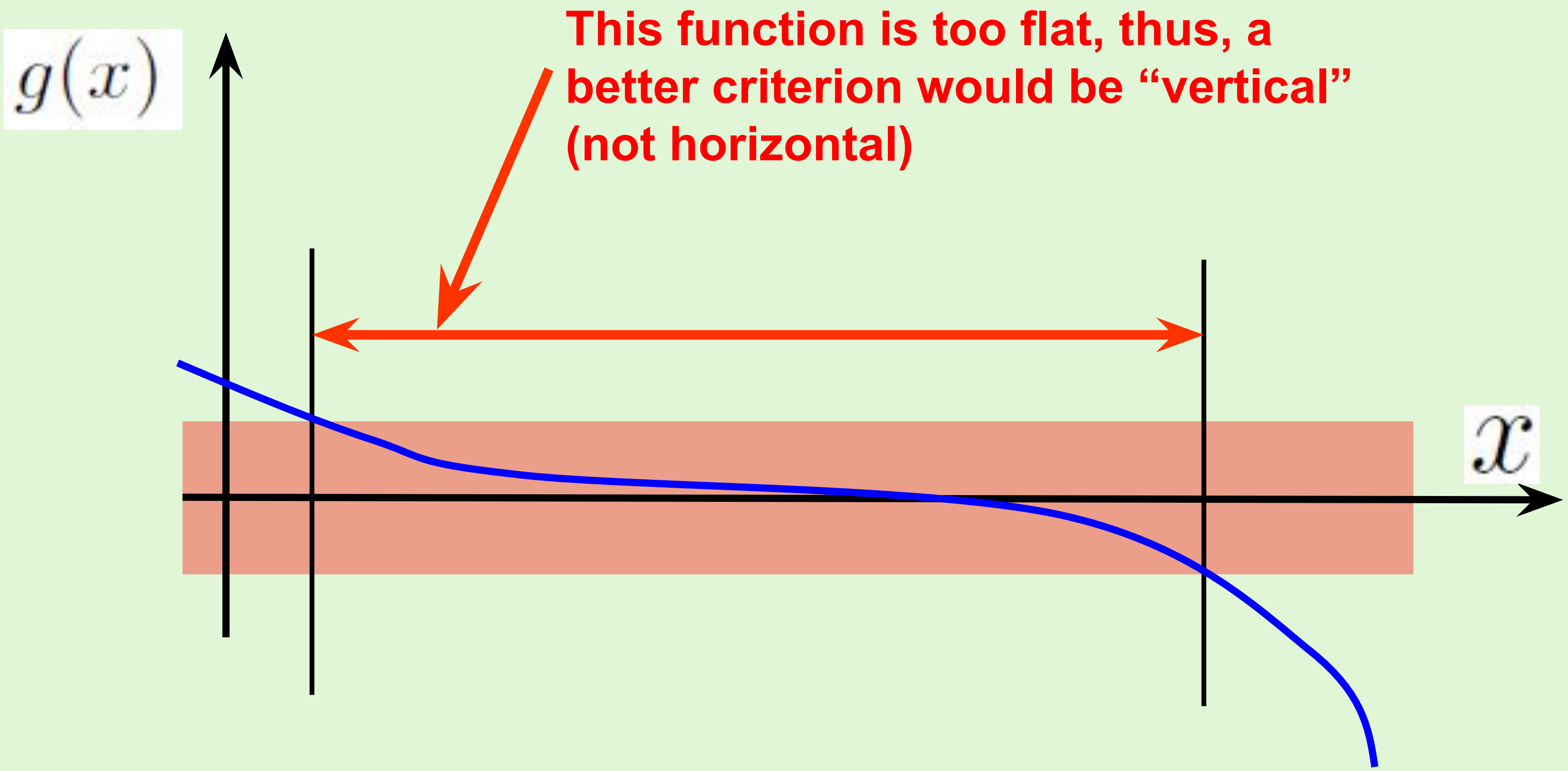
Root finding algorithms find numerical values such that the function g is "almost" zero, i.e. has small values.



As long as the function is in the pink strip, we consider it equal to zero, thus any point in the corresponding horizontal segment is an approximation of the root

What questions are involved in solving f(x)=0?

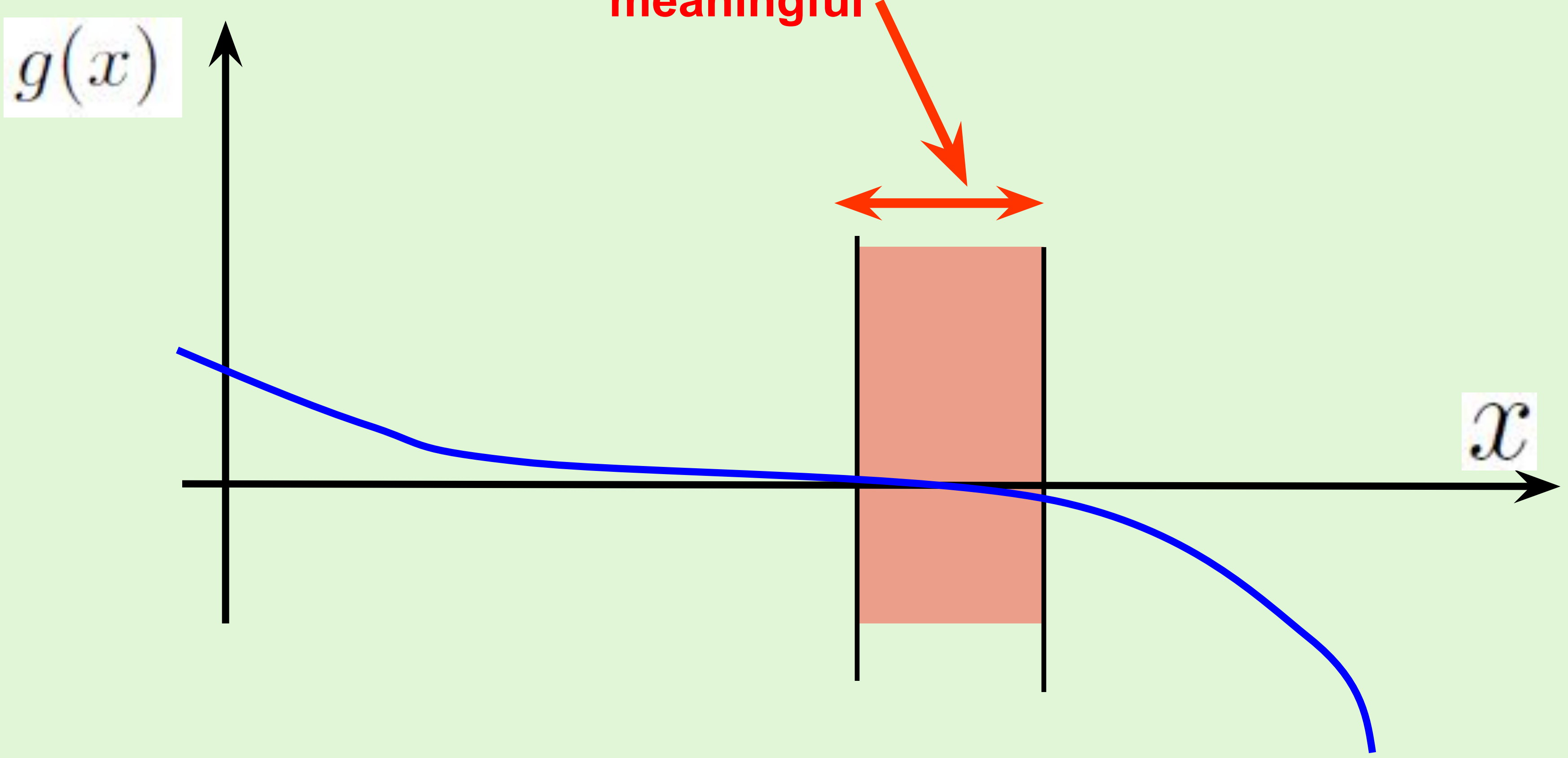
This can be a problem if the function is too “flat”


This function is too flat, thus, a better criterion would be “vertical” (not horizontal)
$g(x)$
x

What questions are involved in solving g(x)=0?

This can be a problem if the function is too "flat"

If one could say that the root is in this vertical slab, it would be more meaningful



How do root finding algorithms work?

In general root finding algorithms have three features

1) A guess or initialization procedure
2) An iterative procedure to refine the approximation of the solution
3) A stopping criterion (when the solution is “good enough”)

```
Start: input guess of the problem (or initialize the problem)

While (solution not good enough)
    Refine the solution

Stop: when solution is good enough
```

Mathematically

Iterative methods start with a "guess", labeled x_0 and through "easy" calculations, generate a sequence

$x_1, x_2, x_3, \ldots$

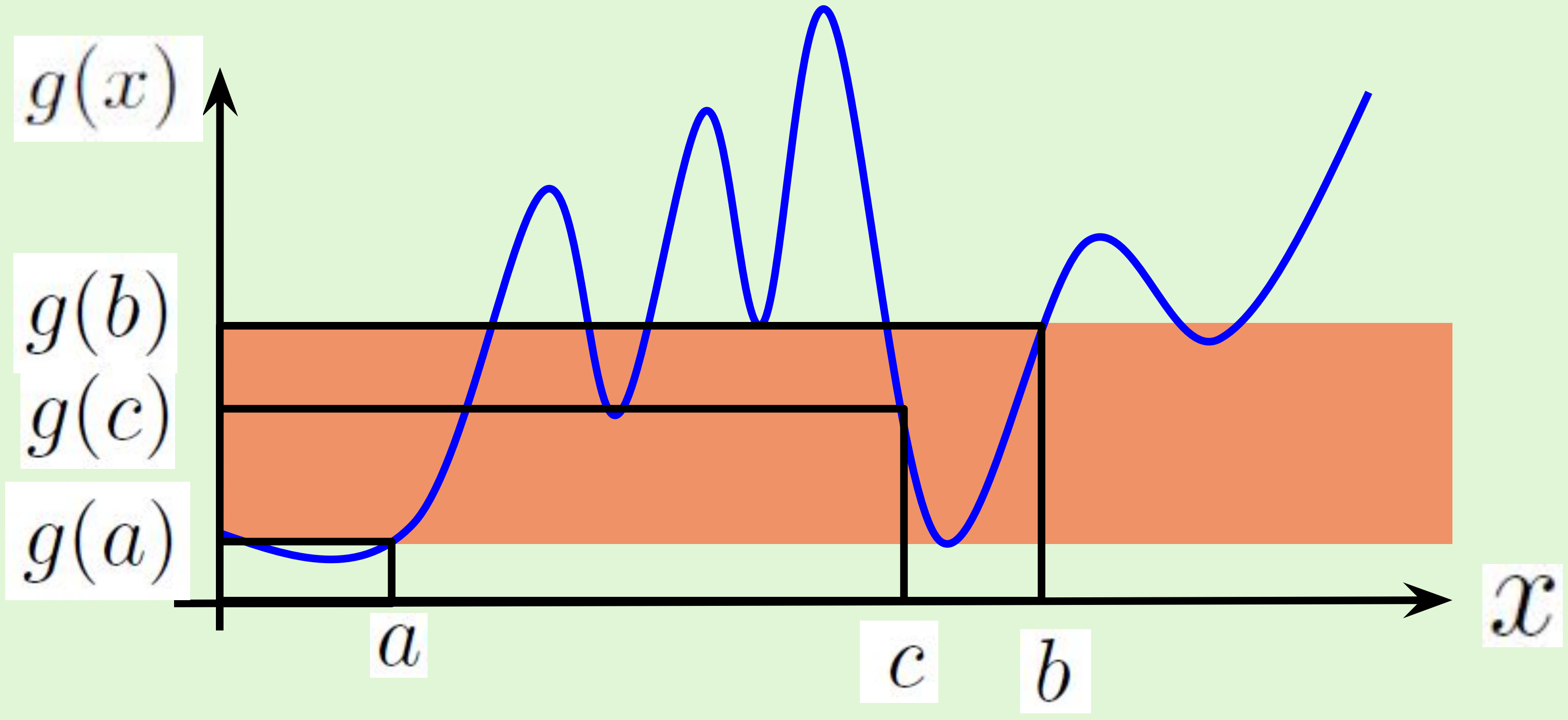
Goal is that sequence satisfies

$$\underbrace{\lim_{n\to\infty} x_n = x^*}_{\text{convergence to a limit, } x^*} \quad \text{and} \quad \underbrace{g\left(x^*\right) = 0}_{x^* \text{ is a solution}}$$

One useful theorem for root finding

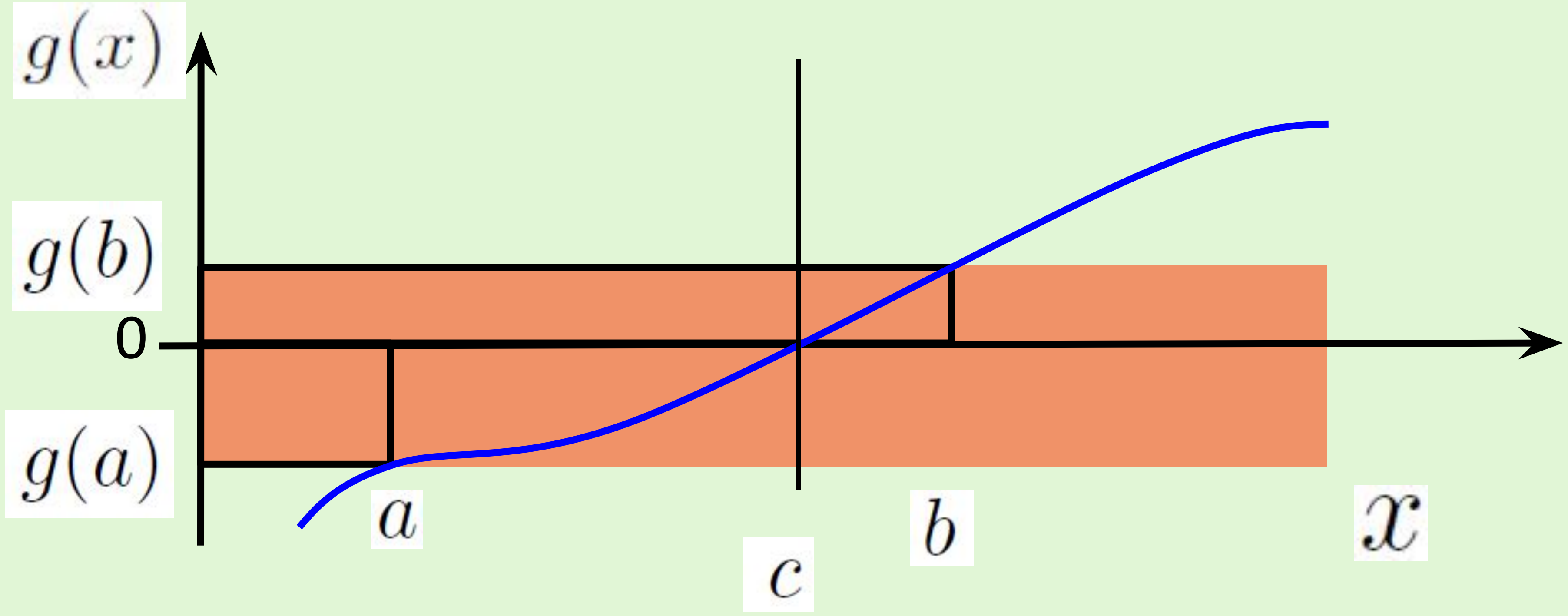
Theorem (Intermediate Value):

If g is a real-valued continuous function on the interval [a, b], and u is a number between g(a) and g(b), then there is a c ∈ [a, b] such that g(c) = u.



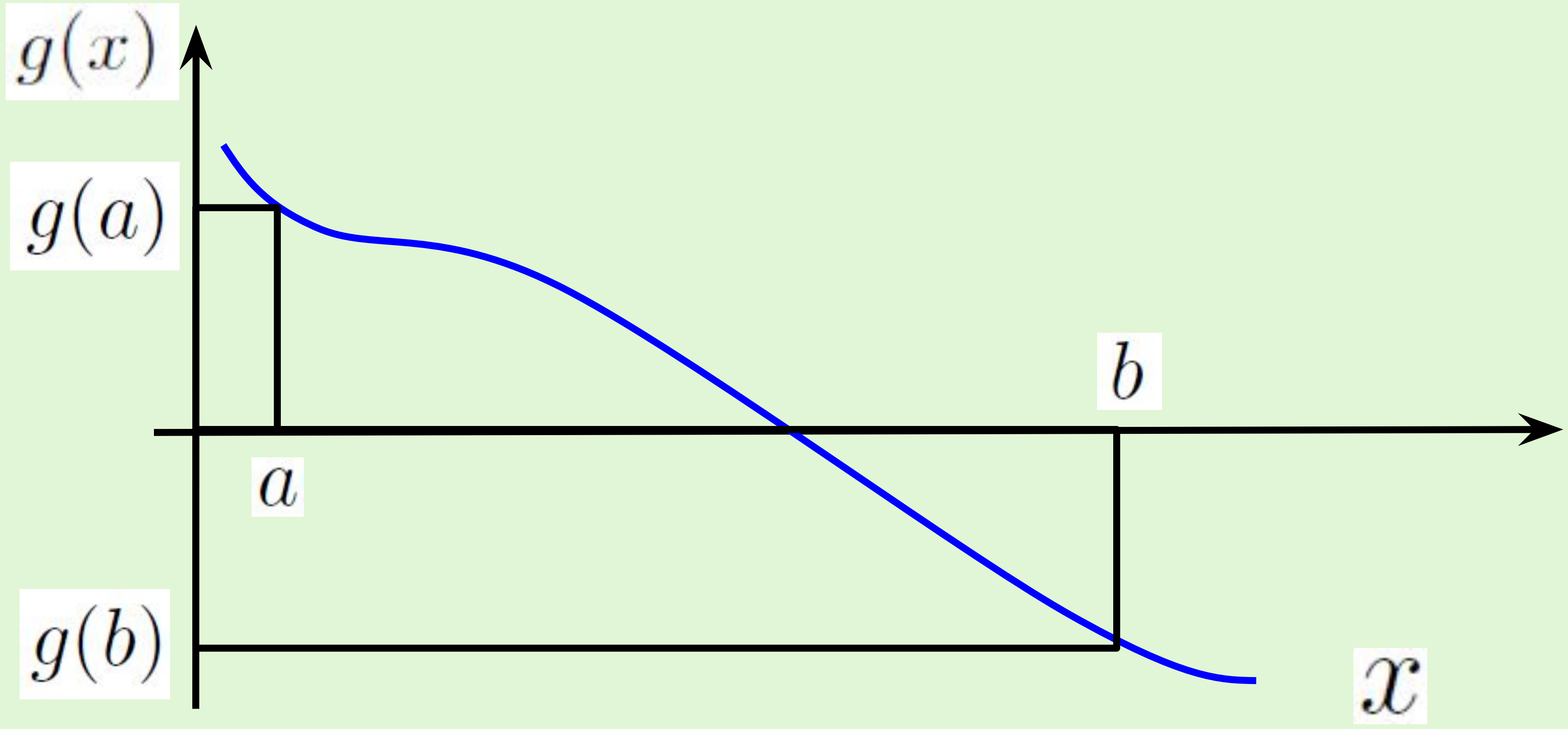
One useful theorem for root finding

Context in which the intermediate value theorem is usually used: finding the zero of a function



Bisection algorithm

Initialization step: enter a and b, such that g(a) and g(b) are of opposite signs, thus the function has at least one zero between a and b. Whenever this is true g(a)g(b) is negative.

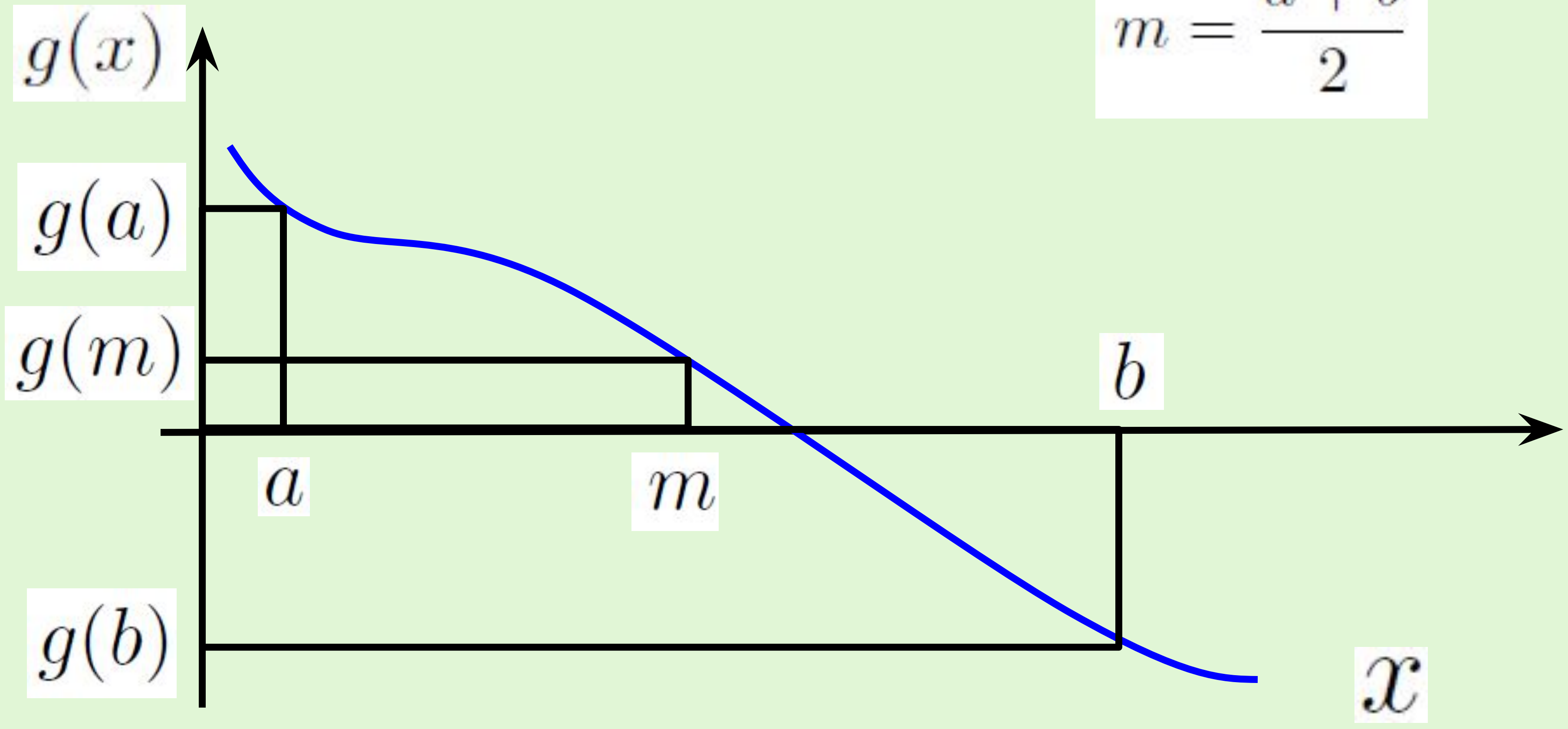


INITIALIZATION STEP

Bisection algorithm

Refinement step: compute midpoint m = (a+b)/2, and evaluate the sign of g(a)g(m)

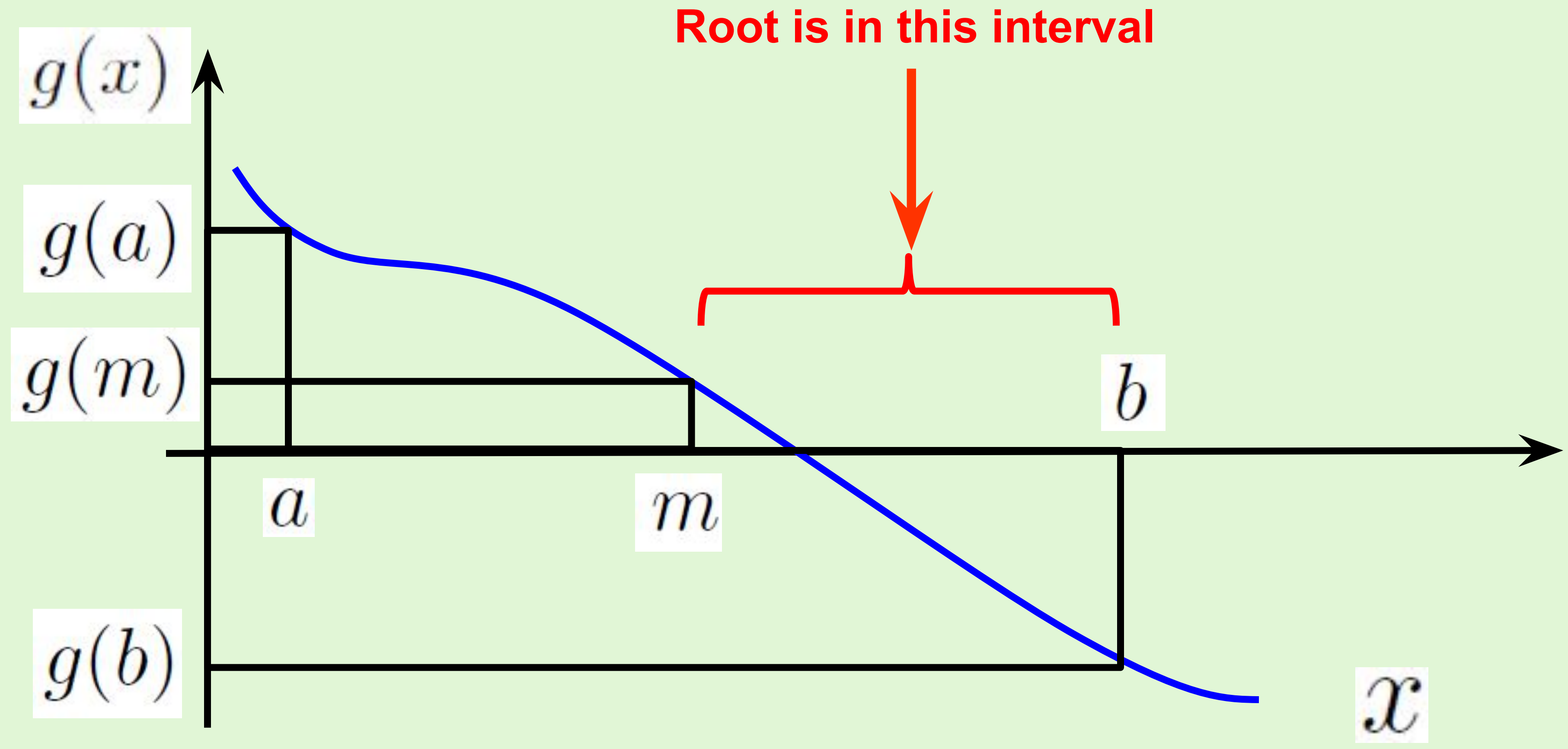
$$m = \frac{a+b}{2}$$



REFINEMENT STEP

Bisection algorithm

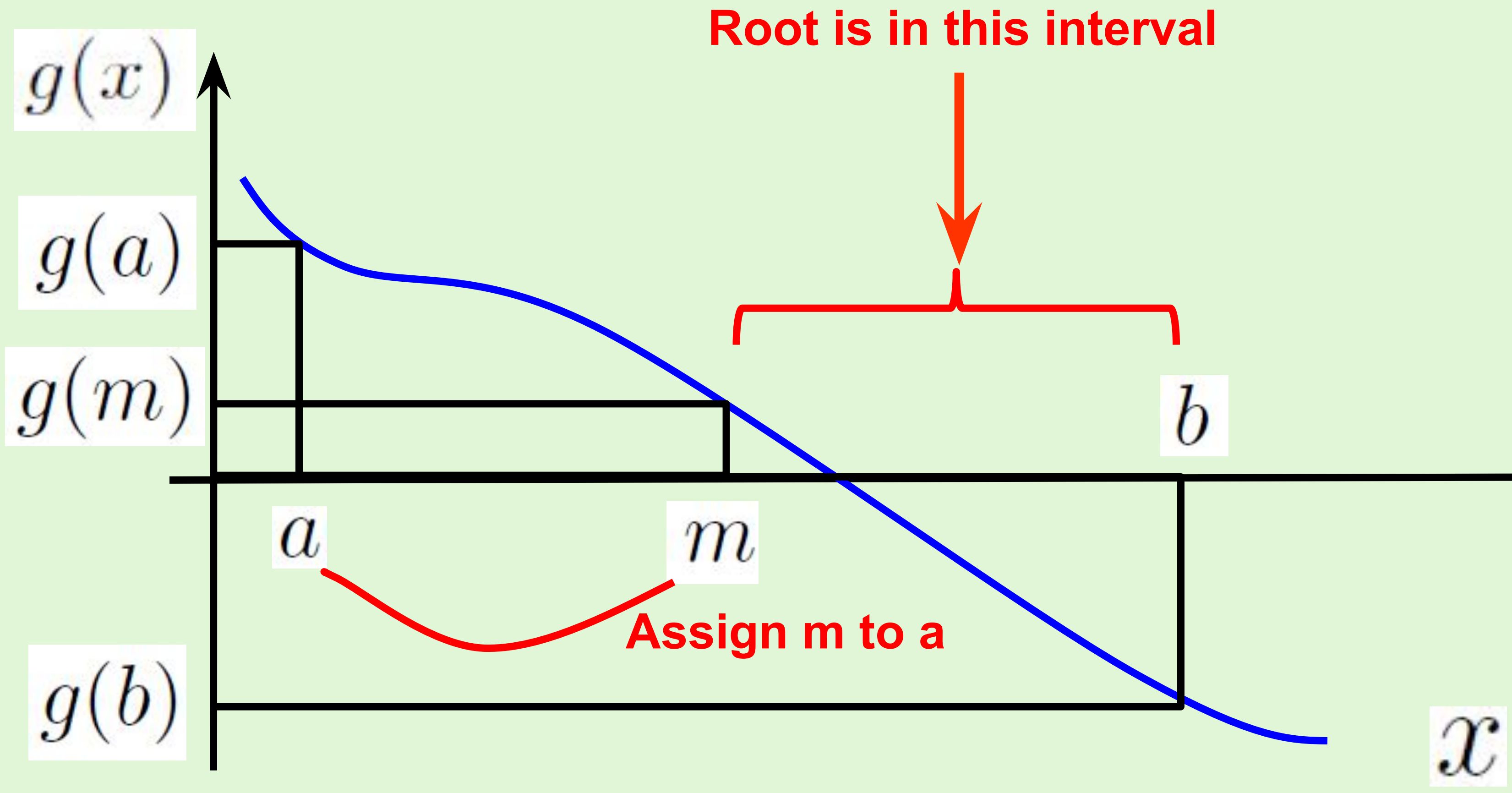
Analyze the situation: g(a)g(m) positive => g(a) and g(m) have the same sign, so the root must be between m and b



REFINEMENT STEP

Bisection algorithm

Analyze the situation: g(a)g(m) positive => g(a) and g(b) have the same sign, so the root must be between m and b

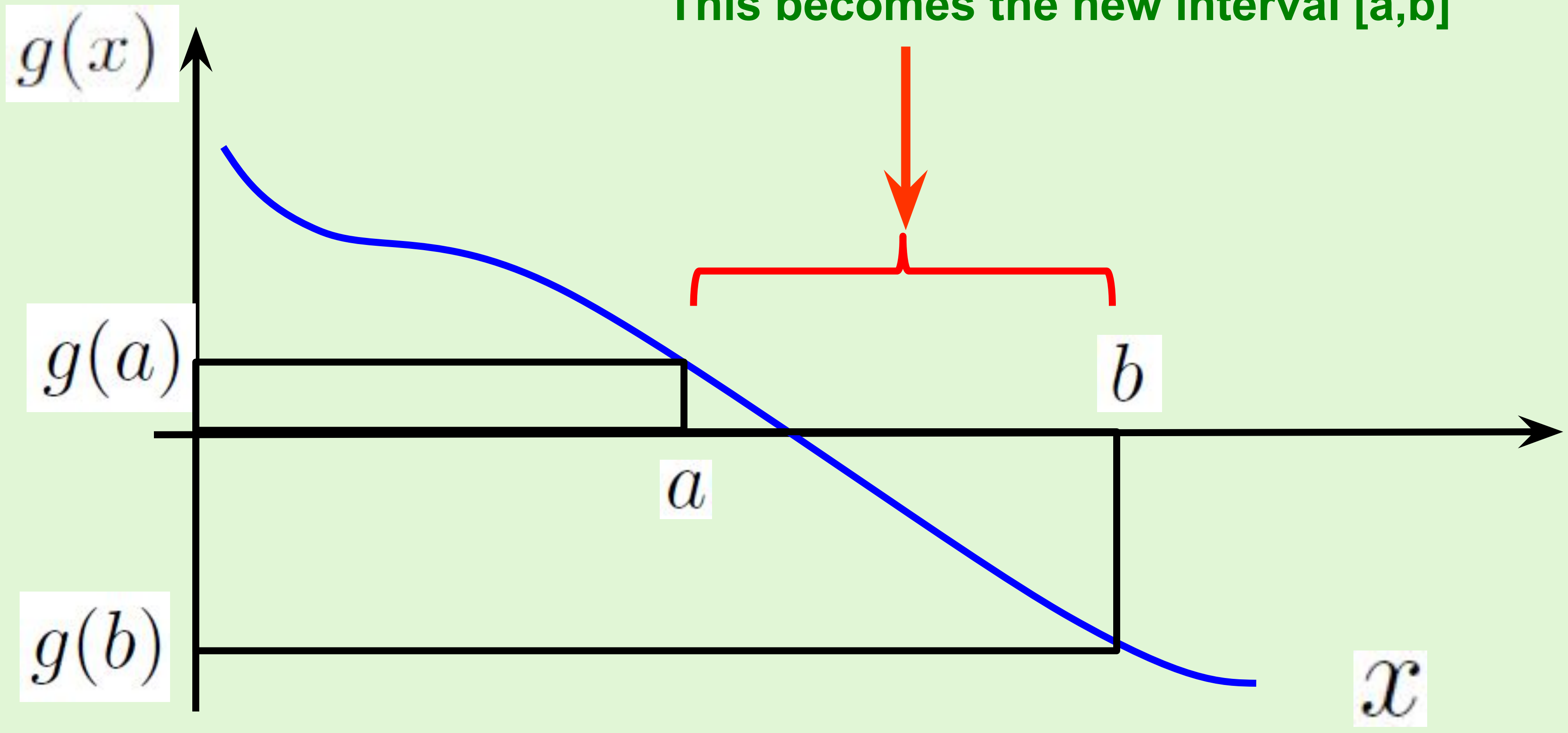


REFINEMENT STEP

Bisection algorithm

Now we have a new interval and we can start the search again

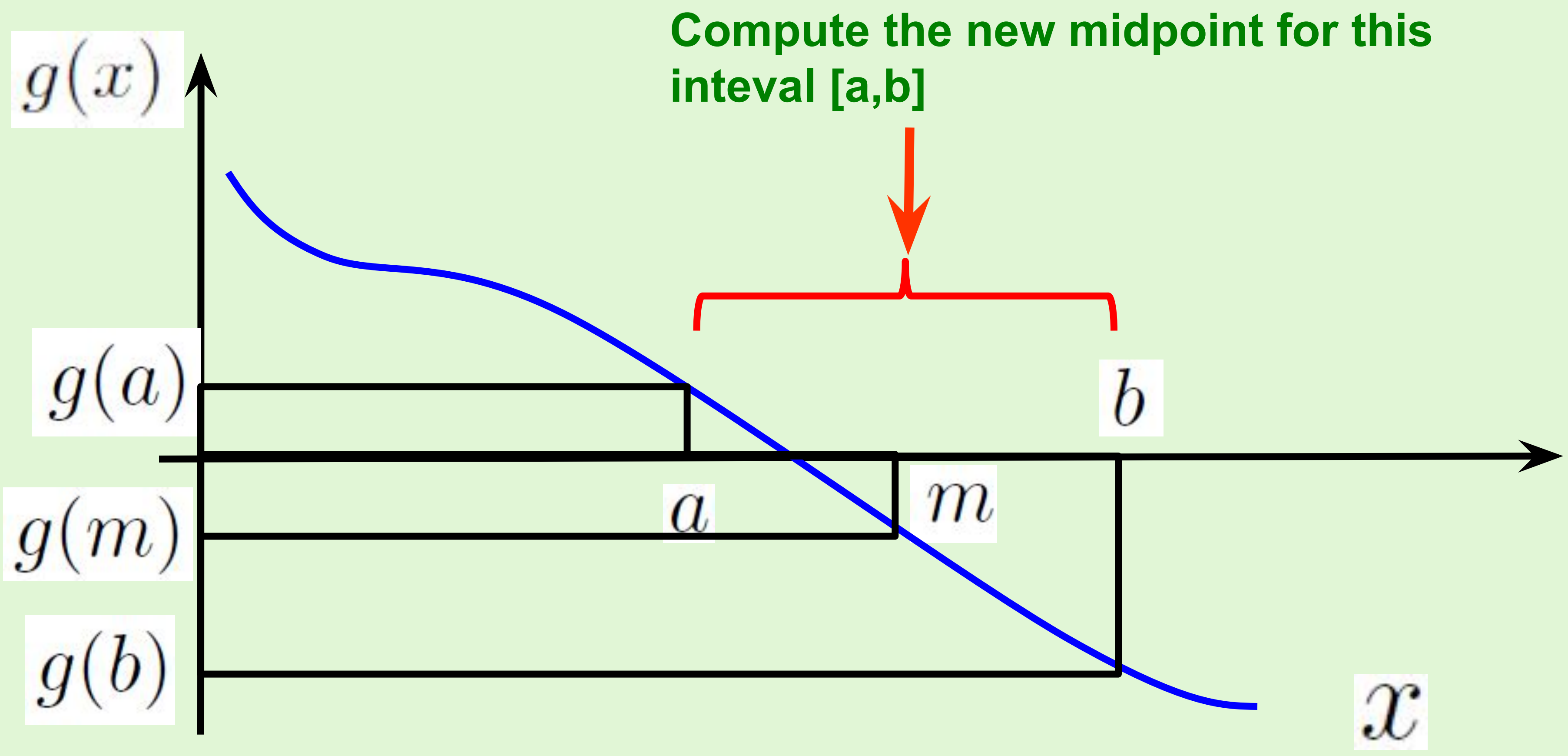
This becomes the new interval [a,b]



REFINEMENT STEP

Bisection algorithm

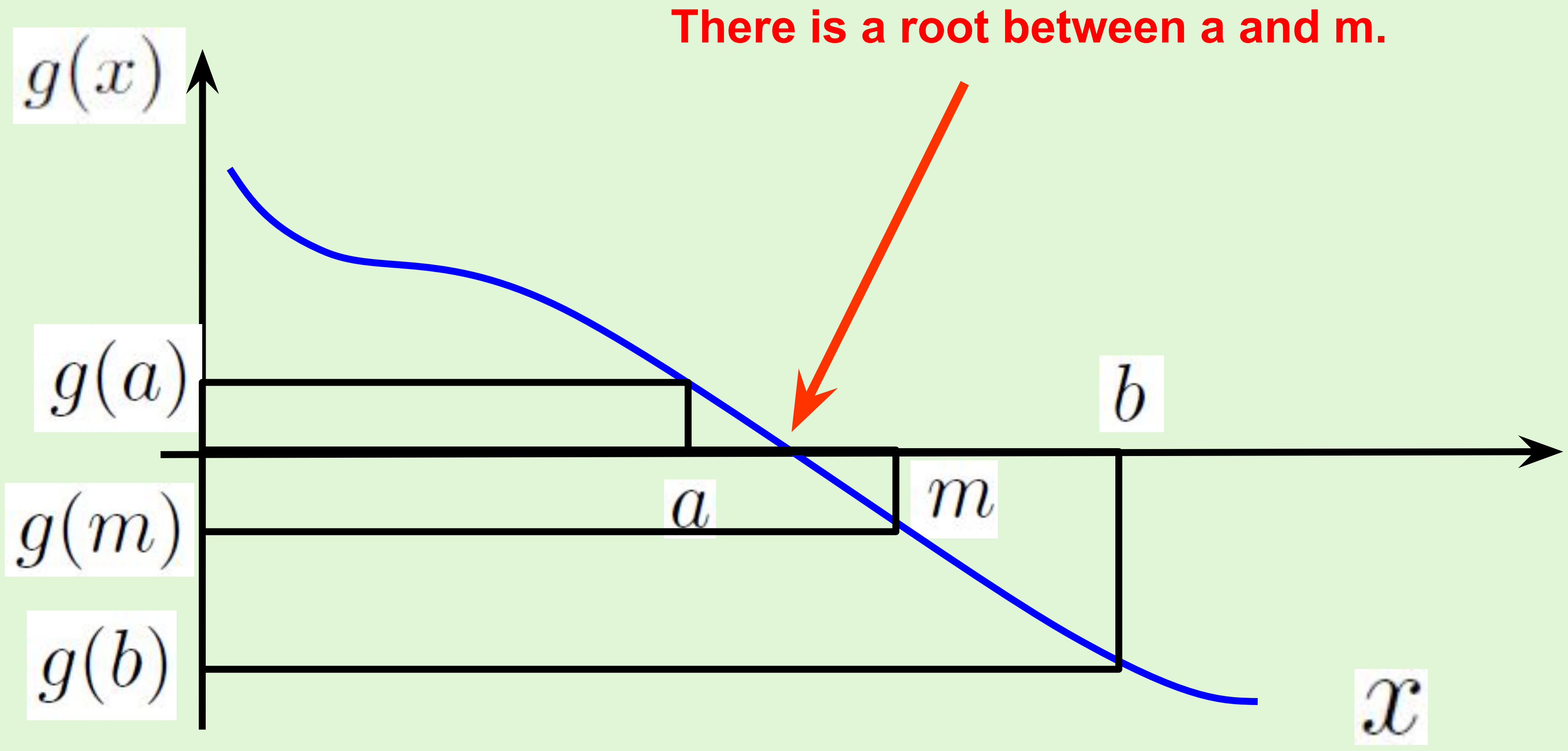
Compute the new midpoint for the new interval (as in the previous step)



REFINEMENT STEP

Bisection algorithm

Evaluate the sign of g(a)g(m), here g(a)g(m) is negative

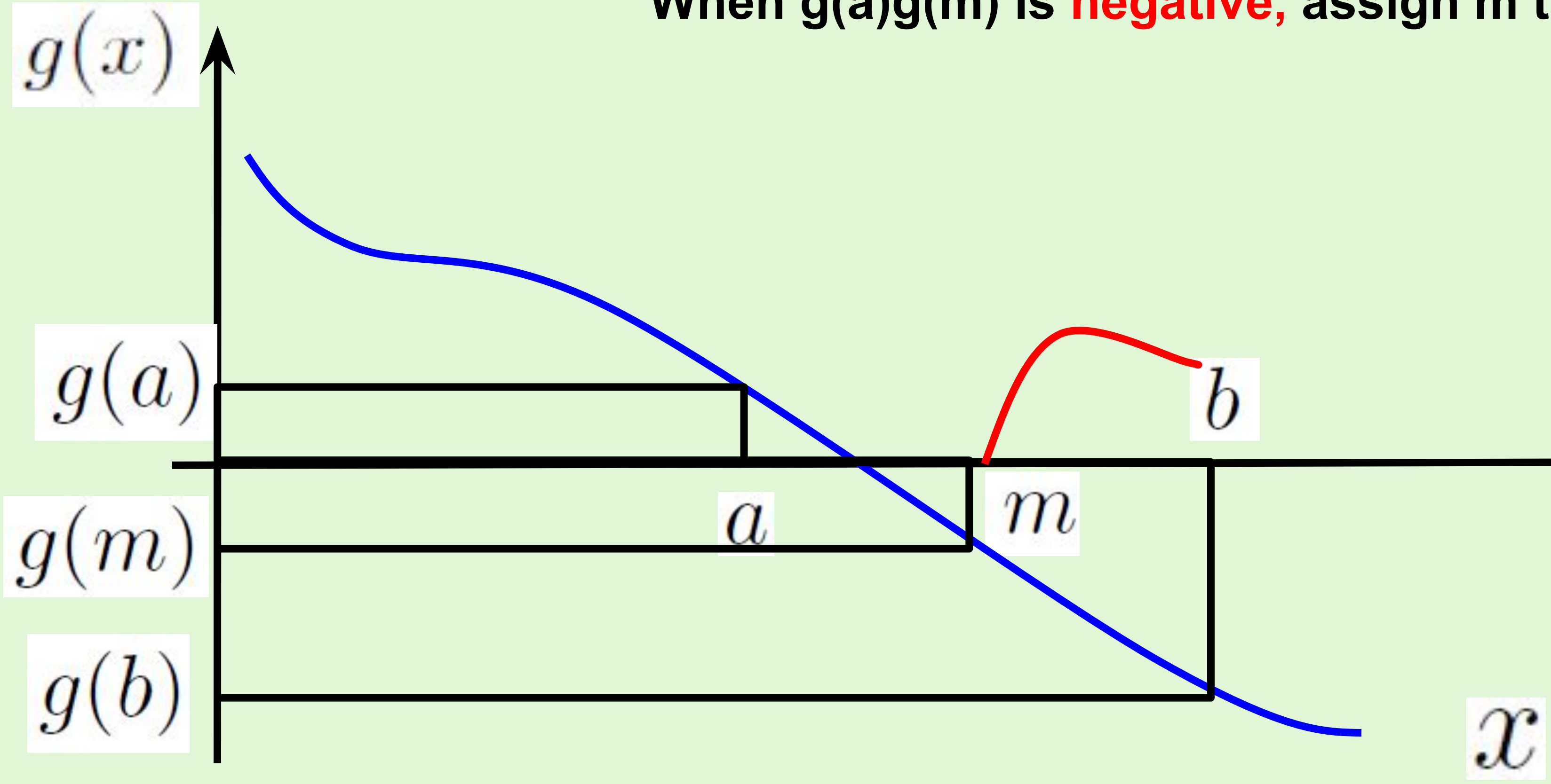


REFINEMENT STEP

Bisection algorithm

Evaluate the sign of g(a)g(m), here g(a)g(m) is negative

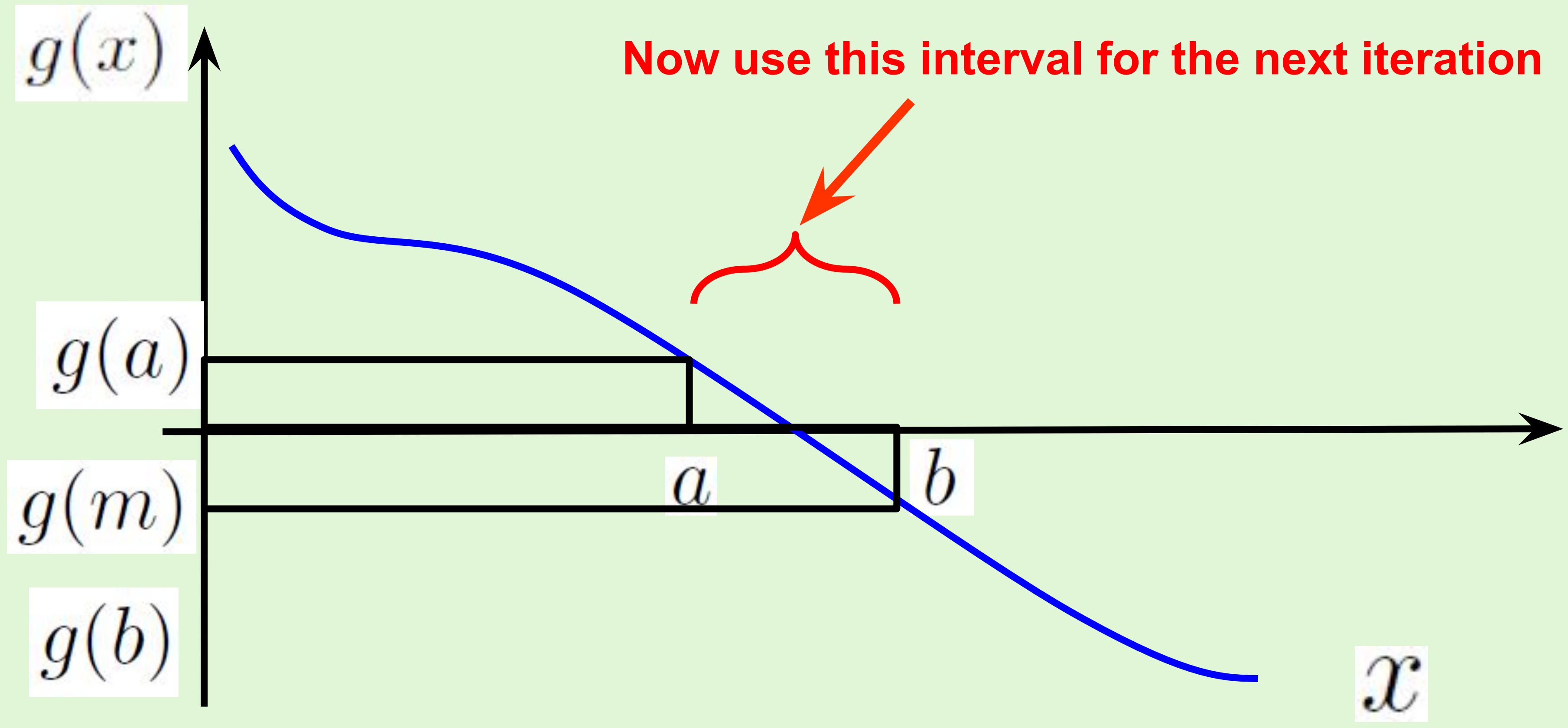
When g(a)g(m) is negative, assign m to b



REFINEMENT STEP

Bisection algorithm

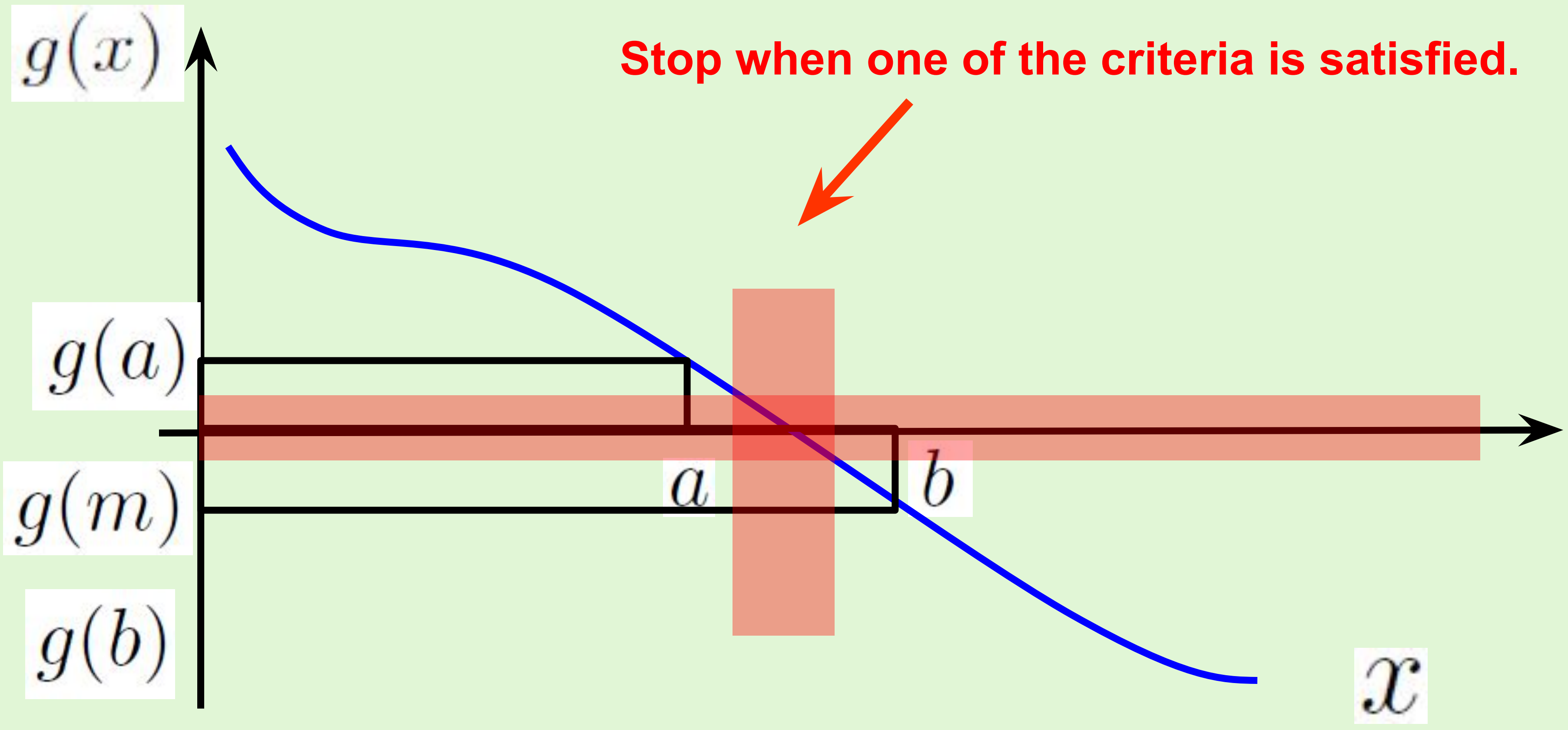
Evaluate the sign of the function at the midpoint



REFINEMENT STEP

Bisection algorithm

Evaluate the sign of the function at the midpoint



STOPPING CRITERION

Simple pseudocode: Bisection

```
while abs(a-b)>tol % stopping criteria
    m = (a+b)/2;
    if f(a)*f(m)<0    % root in (a,m)
        b = m;
    else
        a = m;
    end
end
```

Advantages of Bisections

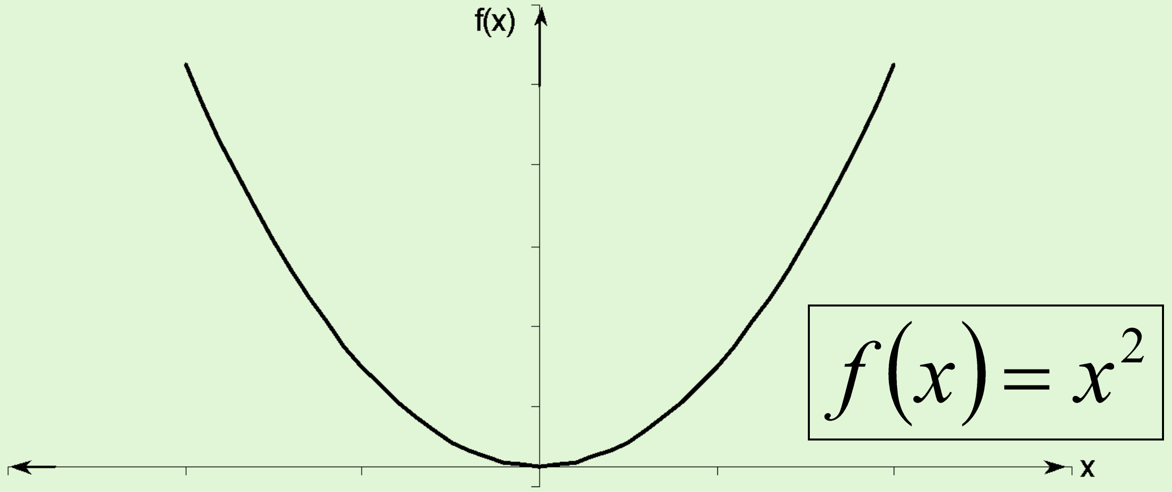
- Always convergent
- The root bracket gets halved with each iteration - guaranteed.

Drawbacks of Bisection

- Slow convergence
- If one of the initial guesses is close to the root, the convergence is slower

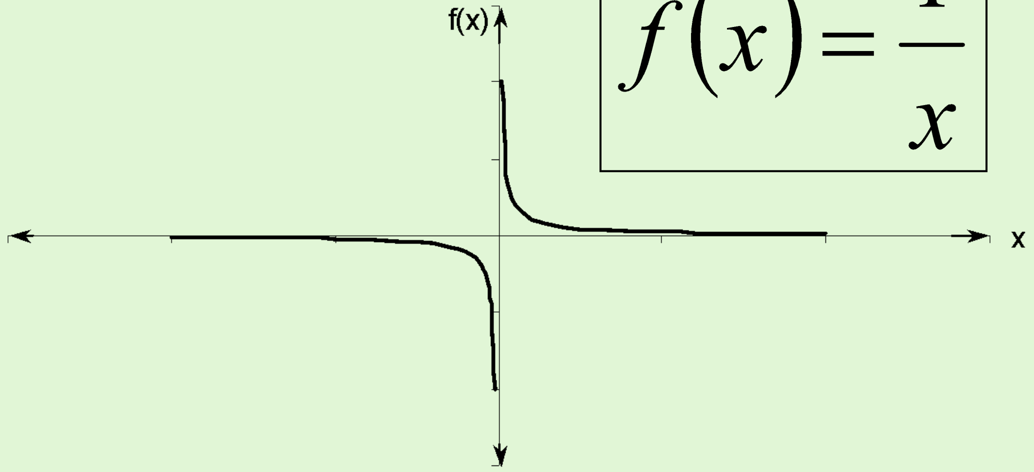
Drawbacks of Bisection

- If a function f(x) is such that it just touches the x-axis it will be unable to find the lower and upper guesses.



Drawbacks of Bisection

- Function changes sign but root does not exist



$$\boxed{f(x) = \frac{1}{x}}$$

HW Problem 7.3.5

- The MATLAB documentation on finding solutions to linear equations is very helpful

https://www.mathworks.com/help/matlab/math/systems-of-linear-equations.html