Summary: Partial Derivatives

$f: \mathbb{R}^n \to \mathbb{R}^m$ so $f(x) = f(x_1, x_2, \ldots, x_n) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{bmatrix}$

$$\frac{\partial f}{\partial x_i}(x) := \frac{d}{dt} f(x + h e_i)\Big|_{t=0}$$ means to

hold all components of x constant except x_i

$$e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \; e_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \; \ldots, \; e_n = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

Jacobian (rarely fun to compute by hand)

$$f: \mathbb{R}^n \to \mathbb{R}^m \Rightarrow f(x) = \begin{bmatrix} f_1(x_1, \ldots, x_n) \\ \vdots \\ f_m(x_1, \ldots, x_n) \end{bmatrix}$$

$$J_f(x) := \frac{\partial f(x)}{\partial x} := \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) & \frac{\partial f}{\partial x_2}(x) & \cdots & \frac{\partial f}{\partial x_n}(x) \end{bmatrix}_{m \times n}$$

$$= \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & & \ddots & \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}_{m \times n}$$

= Symbolics.jacobian(f,x)

OR ForwardDiff.jacobian(f, x0)

Linear Approximation about x_0

$$f(x) \approx f(x_0) + J_f(x_0) \cdot (x - x_0)$$

Today: Continue Engineering Applications of the Derivative

Warm-up Problem: Compute the Jacobian

$$f(x) = \begin{bmatrix} e^{x_1 \sin(x_2)} \\ 3(x_1)^2 + x_1 x_2 \end{bmatrix}$$

$$J_f(x) = \begin{bmatrix} e^{x_1 \sin(x_2)} \cdot \sin(x_2) & e^{x_1 \sin(x_2)} \cdot x_1 \cos(x_2) \\ 6x_1 + x_2 & x_1 \end{bmatrix}$$

2×2

[Skip to the NL case]

Root Finding (aka, Solving Equations)

Topic is treated in ROB 101, and is "merely" summarized here!

Linear Case ("Square")

Same number of equations as unknowns

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$Ax = b$ Exists unique x^* such that $Ax^* = b$ if, and only if

$\det(A) \neq 0 \iff \text{rank}(A) = n$

Julia>> `r = rank(A, aTol)` where aTol ≈ 1e-5

uses "SVD"; see ROB 501 (sic) textbook

How to find x^* ?

Julia>> `xStar = A\b`

backslash → uses a "robust" solver

$y(x) := Ax - b$ we have

$y(x^*) = 0 \iff Ax^* = b$

x^* is called a root, similar to $ax^2 + bx + c = 0$, soln = root

Nonlinear Case: We use iteration and linear approximations to solve

$y = f(x)$, $f: \mathbb{R}^n \to \mathbb{R}^n$ ("square")

Abuse of notation: x_k is the k-th iterate of vector $x \in \mathbb{R}^n$

$$\therefore x_k = \begin{bmatrix} x_{k,1} \\ x_{k,2} \\ \vdots \\ x_{k,n} \end{bmatrix}_{n \times 1}$$

k = index in for loop

We seek a sequence of vectors

$x_0, x_1, x_2, \ldots, x_k, x_{k+1}, \ldots,$ such that

$$\lim_{k \to \infty} f(x_k) = 0,$$ that is

$x^* := \lim_{k\to\infty} x_k$ is a root $\Leftrightarrow y = f(x^*) = 0$

(x^* solves the equation)

Let x_k be the current approx. solution, how to find a better approx, x_{k+1}?

Idea: Replace $f(x)$ by a linear approx.

$$f(x) \approx f(x_k) + J_f(x_k)(x - x_k)$$

Select x_{k+1} such that

$$f(x_{k+1}) \approx 0_{n\times 1} = f(x_k) + J_f(x_k)\underbrace{(x_{k+1} - x_k)}_{\Delta x_k}$$

$$\begin{cases} \Delta x_k = J_f(x_k) \backslash (-f(x_k)) \\ x_{k+1} = x_k + \Delta x_k \end{cases}$$

Newton-Raphson

\ = backslash command in Julia, faster and more robust than matrix inverse.

IF YOU DO NOT FULLY TRUST the Linear Approx.

introduce $0<s<1$ and use

$$x_{k+1} = x_k + s\,\Delta x_k$$

(step size)

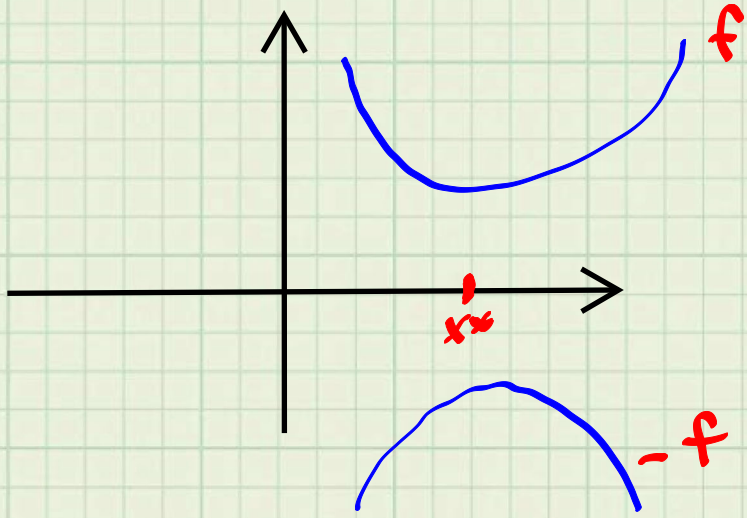
* "damped N.R. Method"

Julia Demo with NLSolve

Linear problems: unique solution, Nonlinear $\Rightarrow$ there can be no solution or many!

Let $f: \mathbb{R}^n \longrightarrow \mathbb{R}$ be a scalar-valued function that assesses or measures the "quality" or "optimality" of a solution to a design problem, where $x \in \mathbb{R}^n$ parameterizes potential solutions to the problem. Each $x \in \mathbb{R}^n$ is a different solution (think of x as the motor torques in a robot).

f is often called a "cost" function



Minimizing f is the same as maximizing negative f. ∴ Only need to solve one of these versions of the problem and most solvers choose minimization.

Seek

$$x^* := \arg\min_{x \in \mathbb{R}^n} f(x)$$

(argument)

*"the" value of x that minimizes f.

$$f(x^*) \leq f(x) \quad \underline{\text{all}} \; x \in \mathbb{R}^n$$

Very hard to achieve. Often

settle for a local solution,

$f(x^*) \leq f(x)$ For $\|x - x^*\| < \delta$

local optimality

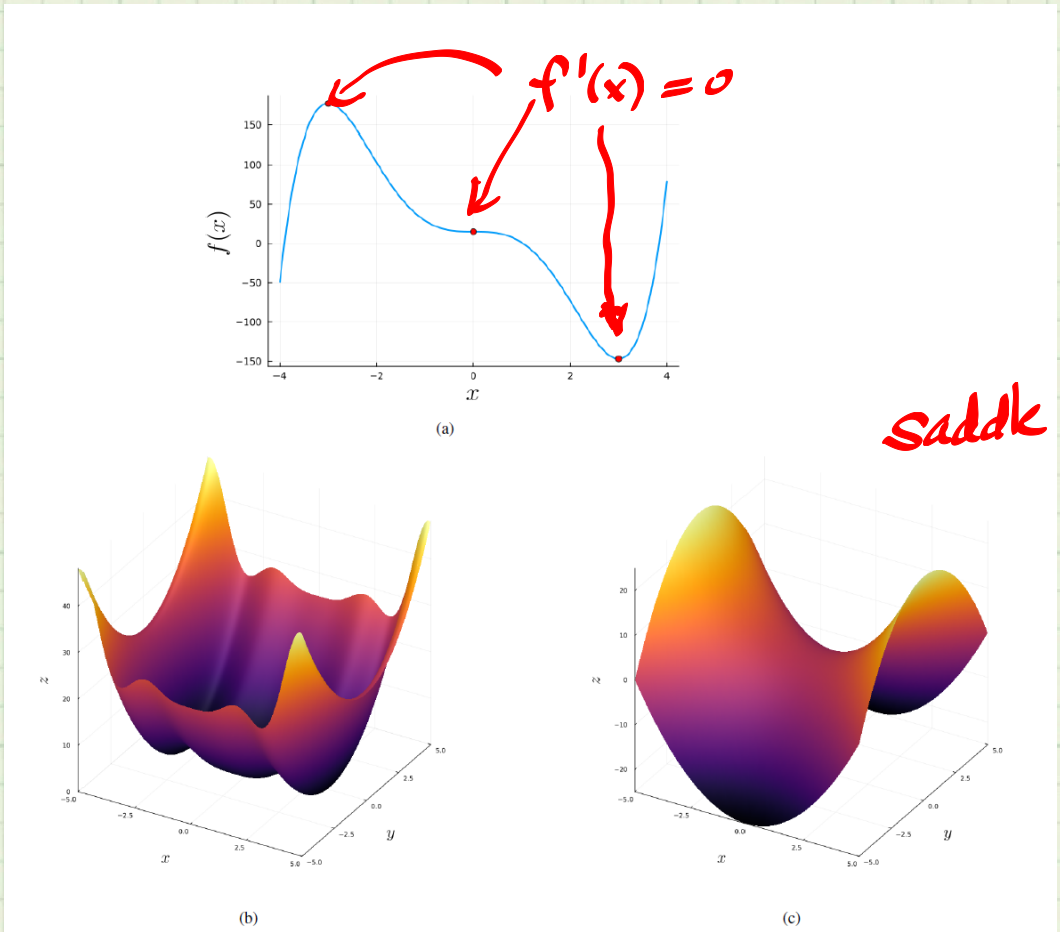


Figure 6.3: Three cost functions. (a) $f(x) = x^5 - 15x^3 + 15$ has a local minimum at -3, an inflection point at the origin, and a local maximum at 3. (b) $f(x, y) = (x \cdot \sin(x))^2 + y^2$ is riddled with local minima and maxima. (c) $f(x, y) = x^2 - y^2$ has a saddle point at the origin. The name could not be more clear!

Chechin code

$f: \mathbb{R}^n \to \mathbb{R}$

$$f(x) \approx f(x_0) + J_f(x_0)(x - x_0)$$

and hence

$$f(x_0 + \Delta x) \approx f(x_0) + J_f(x_0) \cdot \Delta x$$

Fact: $f(x_0 + \Delta x) \approx f(x_0)$ for all choices of Δx (step directions) if, and only if, $J_f(x_0) = 0_{1 \times n}$

Recall $J_f(x) = \left[\frac{\partial f}{\partial x_1}(x) \quad \cdots \quad \frac{\partial f}{\partial x_n}(x) \right]_{1 \times n}$

Def. x_0 is a stationary point of $f(x)$ if $J_f(x_0) = 0_{1 \times n}$.

Stationary pts can be local min's, max's or saddle points.

Intuition: From the perspective of

the linearization $y(x) = f(x_0) + Jf(x_0) \cdot (x - x_0)$, dim

the function is literally a constant, $f(x_0)$

If we a studying gradient descent, where/what is the gradient?

Def. For $f: \mathbb{R}^n \to \mathbb{R}$, the gradient at x is

$$\nabla f(x) := \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix} = [Jf(x)]^T$$

transpose

∇ is pronounced "grad" or "gradient"

In LaTex \nabla

How does "re-packaging" the Jacobian do anything for us?

It leads to powerful geometric insight! [Thanks to dot products, aka, inner products]

We seek Δx such that

$$f(x_k + \Delta x) < f(x_k)$$

where x_k is our iterate (solution).

How to choose Δx?

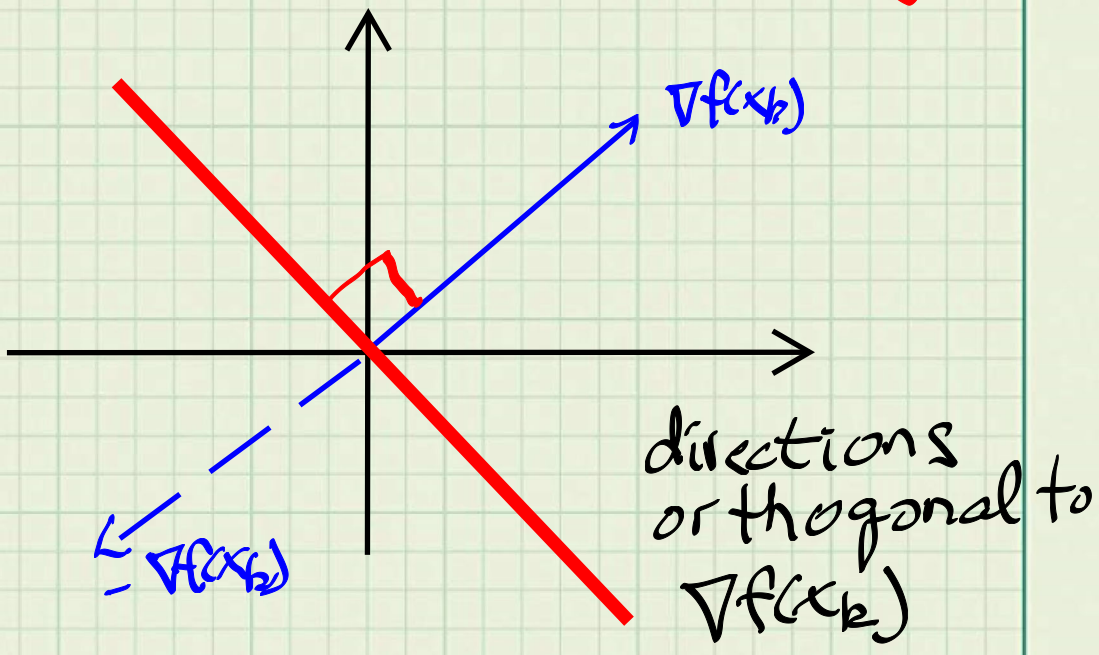
$$f(x_k + \Delta x) \approx f(x_k) + Jf(x_k) \cdot \underbrace{\Delta x}_{\text{"}x_{k+\Delta x} - x_k\text{"}}$$

$$\approx f(x_k) + [\nabla f(x_k)]^T \cdot \Delta x$$

$$\approx f(x_k) + \underbrace{\nabla f(x_k) \cdot \Delta x}_{\text{dot}(\nabla f(x_k), \Delta x)}$$

We reviewed dot products and right angle

$v \perp w \iff \text{dot}(v, w) = 0 \iff$



$\Delta x \perp \nabla f(x_k) \Rightarrow f(x_{k+1} \Delta x) \approx f(x_k)$

$\Rightarrow$ "unfruitful directions for minimizing $f(x)$"

- In $\mathbb{R}^2$, there is a one-dimensional subspace orthogonal to $\nabla f(x_k) \neq 0_{n \times 1}$
- In $\mathbb{R}^3$, " " two-dimensional
- " " " "

• In $\mathbb{R}^n$, there are $(n-1)$ directions that lead to $f(x_{k+1}, \Delta x) \approx f(x_k)$

Hence, we have only one direction to move:

$\Delta x = -\nabla f(x_k)$ for descent

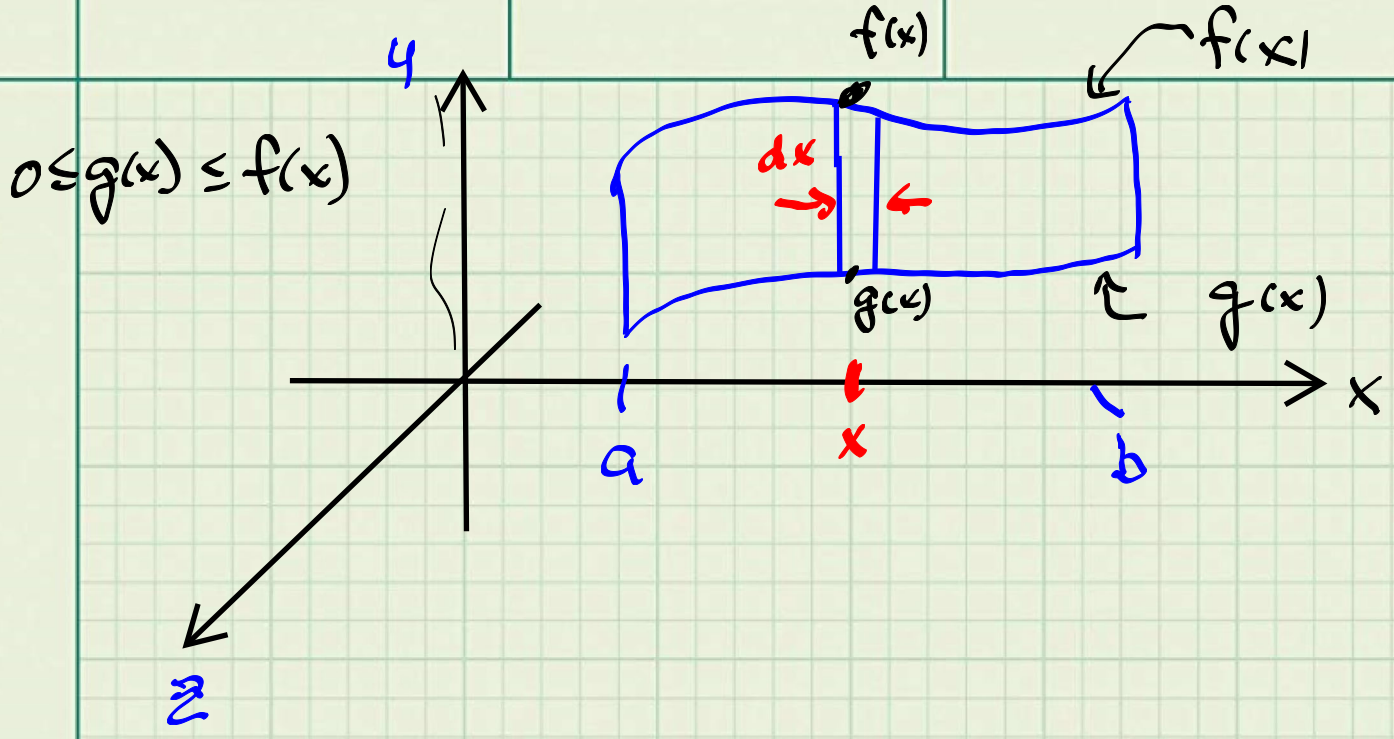
$\Delta x = \nabla f(x_k)$ for ascent.

$$\Delta x = -\nabla f(x_k)$$

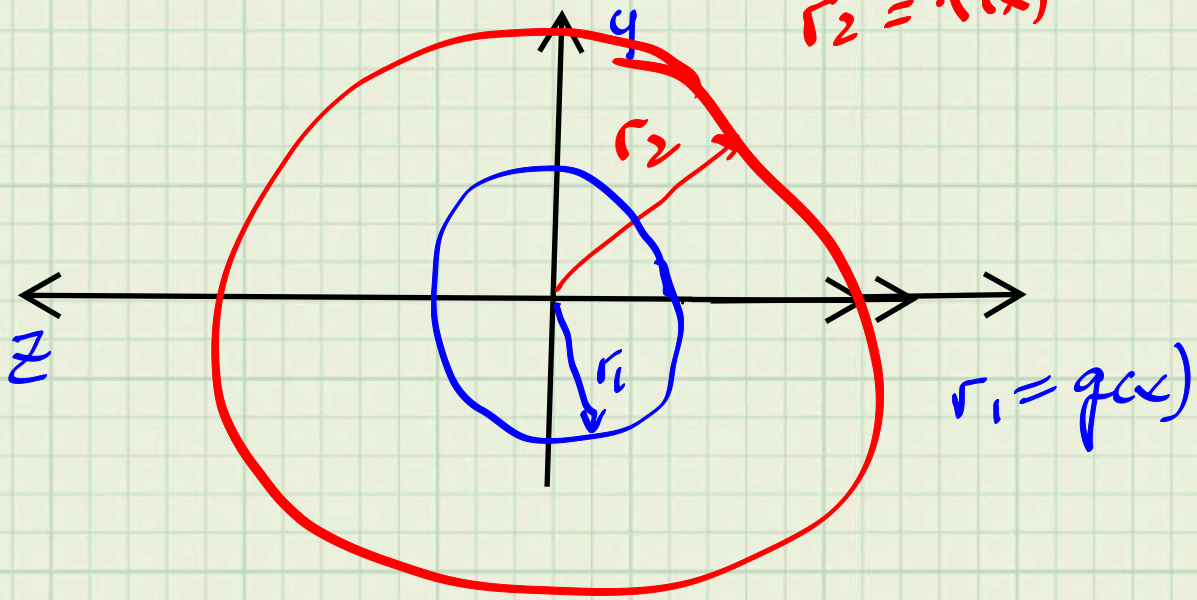
$$x_{k+1} = x_k + s\,\Delta x$$

s called step size

Volumes of Revolution
Solids of Revolution



• Rotate about x-axis



$$A(x) = \pi\left(f^2(x) - g^2(x)\right)$$

$$dV(x) = A(x) \cdot dx$$

$$V(x)_{\text{x-axis}} = \int_a^b dV(x) = \pi \int_a^b \left[f^2(x) - g^2(x)\right] dx$$

- Rotate about the y-axis

Creates a "shell" cylinder with radius x, height $[f(x)-g(x)]$ and thickness dx

$$A(x) = 2\pi x [f(x)-g(x)]$$

$$dV(x) = A(x) \cdot dx$$

$$V_{y\text{-axis}} = \int_a^b dV(x) = \int_a^b 2\pi x [f(x)-g(x)]\,dx$$

We now take up optimization subject to equality constraints

Case 1 Single equality constraint

$$x^* = \arg\min f(x) \quad \text{subject to} \quad g_{eq}(x) = 0$$

$f: \mathbb{R}^n \to \mathbb{R}$

$g_{eq}: \mathbb{R}^n \to \mathbb{R}$

Example: Minimize sum of torque squared in walking, subject to speed equals $v^{des} = 0.9$ m/s

$$g_{eq} = v - v^{des} = 0$$

Let $v, w \in \mathbb{R}^n$, that is

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \quad \text{and} \quad w = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$$

Def. Inner product or (dot product) of v and w is

$$v^T w = \langle v, w \rangle := v \cdot w := \sum_{i=1}^{n} v_i w_i =: \underline{dot(v, w)}$$

We sometimes use Julia's notation

Recall: $v^T := [v_1 \; v_2 \; \cdots \; v_n]$ $1 \times n$ row vector

and $v^T * w = [v_1 \; v_2 \; \cdots \; v_n] * \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}$

$$= \sum_{i=1}^{n} v_i w_i$$

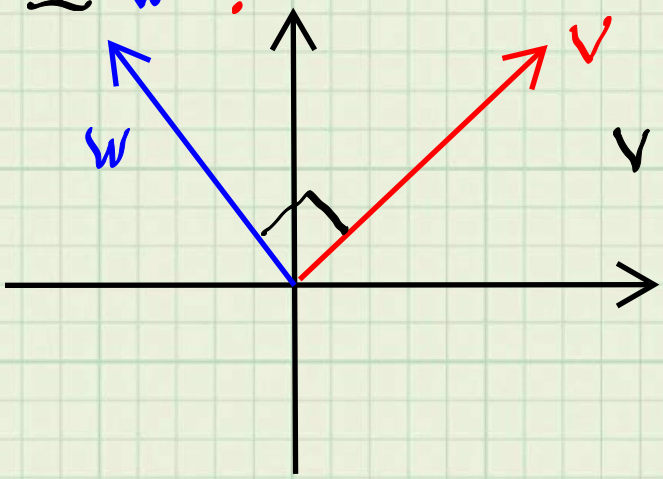
$$= v \cdot w = dot(v, w)$$

Recall: Linearity on both sides

$$dot(\alpha_1 v_1 + \alpha_2 v_2, w) = \alpha_1\, dot(v_1, w) + \alpha_2\, dot(v_2, w)$$

$$dot(v, \beta_1 w_1 + \beta_2 w_2) = \beta_1\, dot(v, w_1) + \beta_2\, dot(v, w_2)$$

Recall: $v \in \mathbb{R}^n$ is perpendicular (orthogonal) to $w \in \mathbb{R}^n$ when $v \cdot w = 0$ ($dot(v, w) = 0$) Notation is $v \perp w$.



$v \perp w \Leftrightarrow$ right angle

Recall: The (Euclidean) norm of $v \in \mathbb{R}^n$ is $\|v\| := \sqrt{\sum_{i=1}^{n} (v_i)^2} = \sqrt{v^T \cdot v} = \sqrt{v \cdot v}$

Hence: $\|v\|^2 = v \cdot v = dot(v, v)$