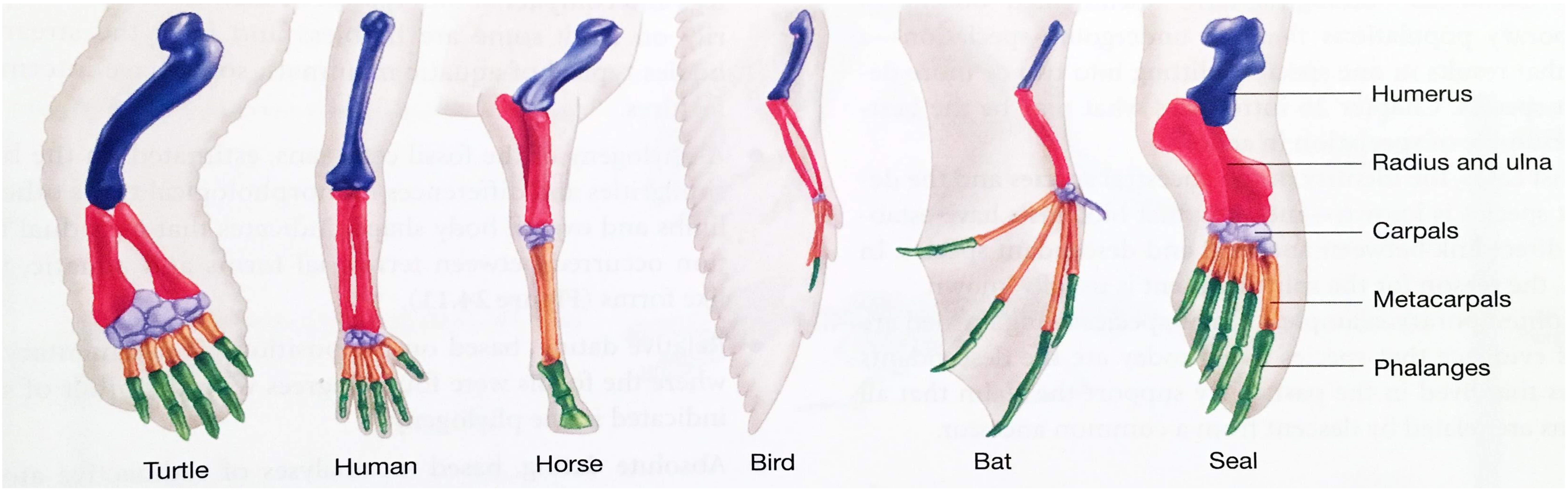
# Numerical Optimization

## - a brief review -

## What is optimization, and why should we care about it?

**Finding the best solution among all possibilities (subject to certain constraints)**

## Find the best solution among all possibilities (subject to certain constraints)



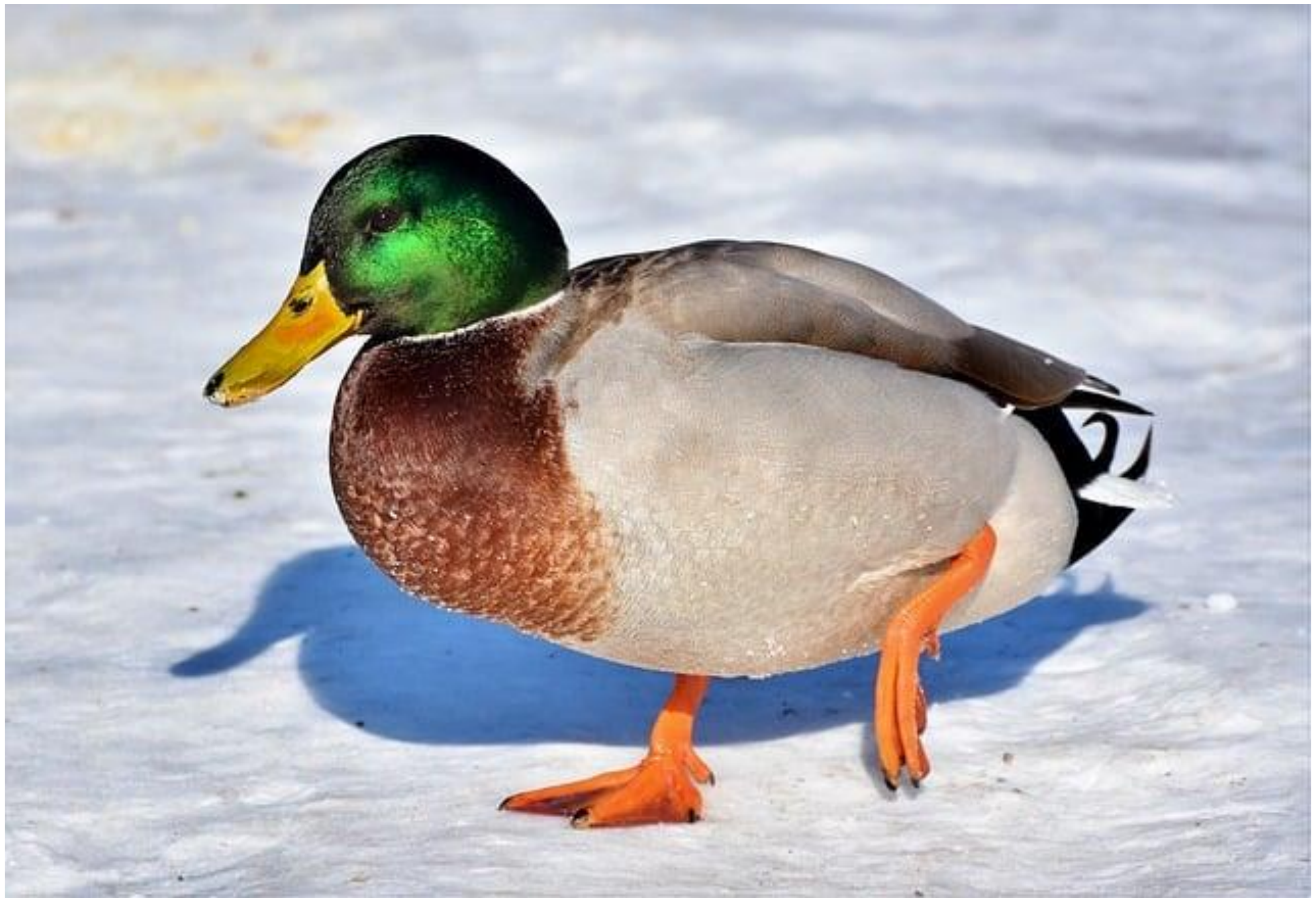
**A parameterized design/template/problem**

## Find the best solution among all possibilities (subject to certain constraints)

Optimized for speed

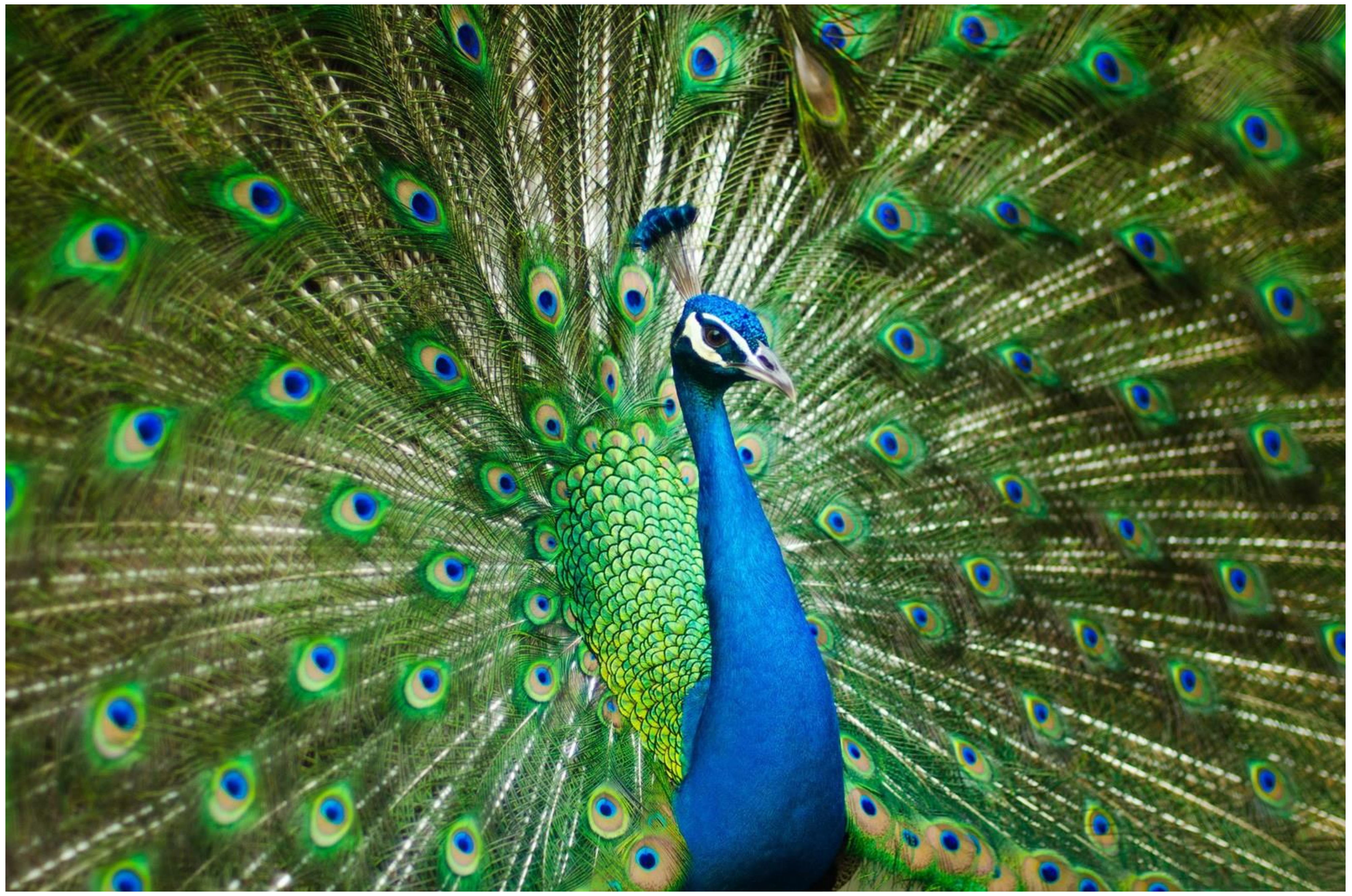
Optimized for efficiency

## Find the best solution among all possibilities (subject to certain constraints)



What is this optimized for?!?

## Find the best solution among all possibilities (subject to certain constraints)


Optimized for beauty

Optimized for beauty?!?

## What is an optimization problem, and why should we care about it?

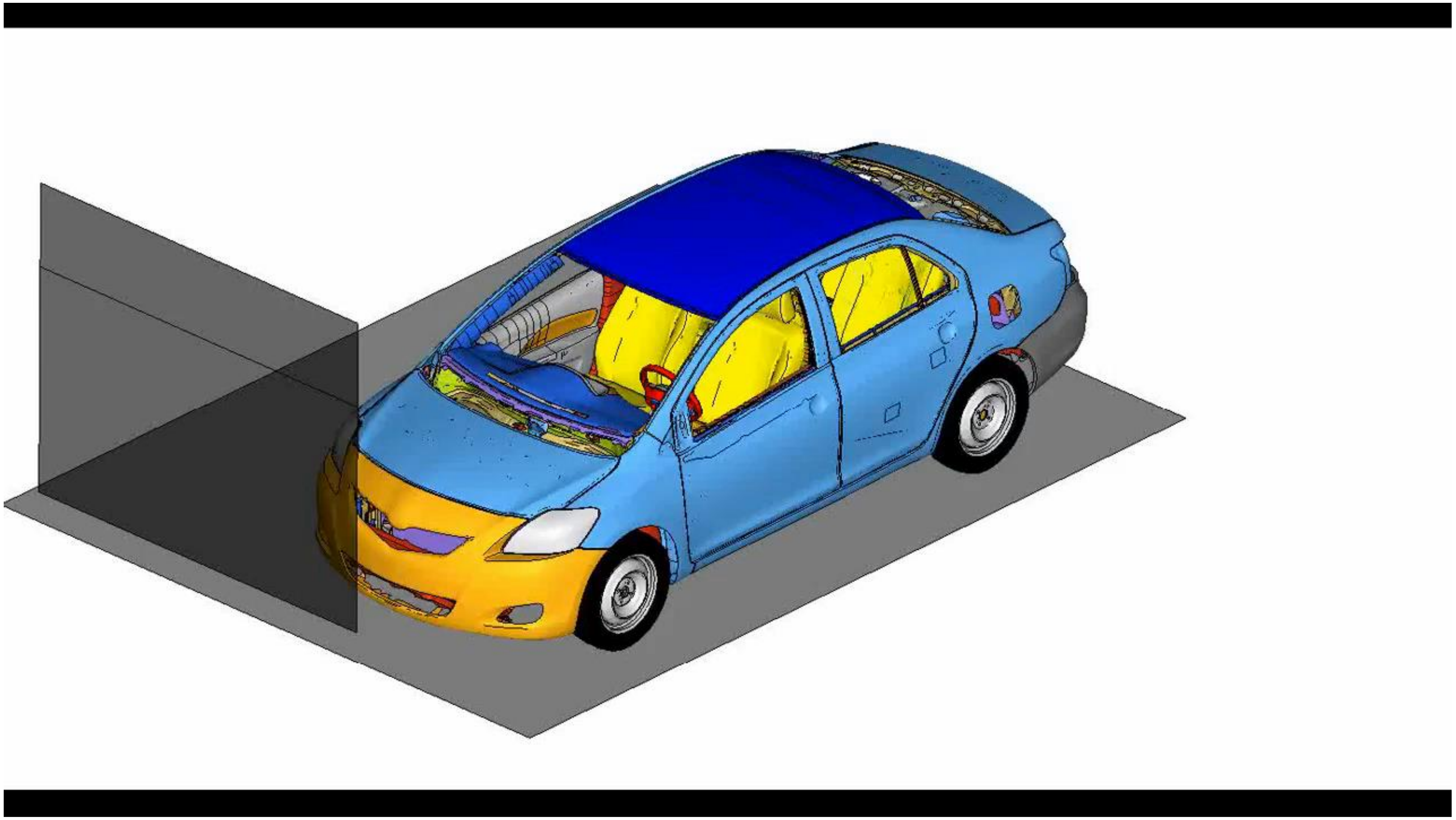
**Ingredients:**

- **a parameterized template/design/problem**
- **an objective that measures how “good” arbitrary points in parameter space are**
- **quite possibly some constraints**

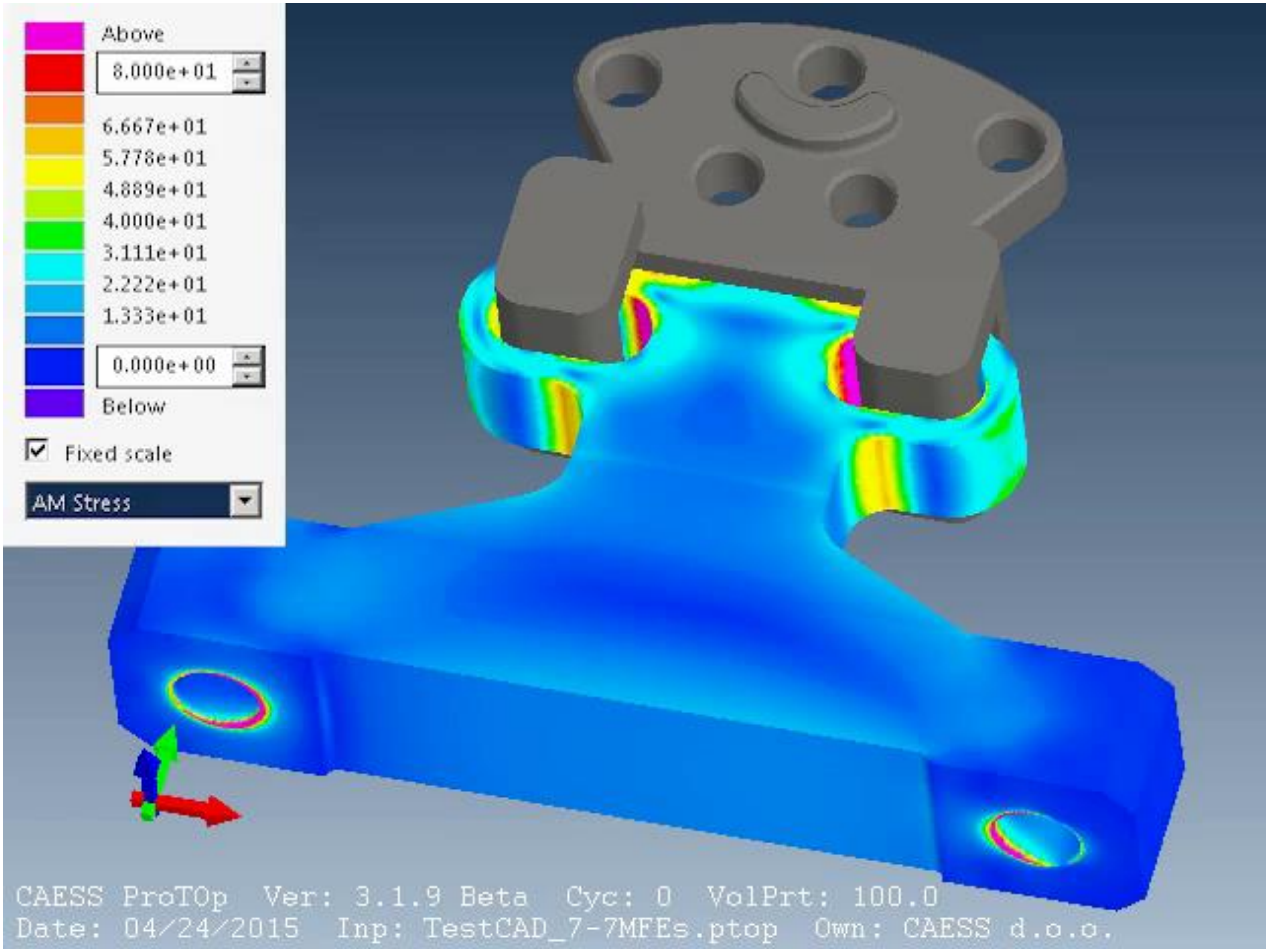
## Optimization problems are EVERYWHERE

**In nature…**
**engineering…**

## Optimization



## Optimization



☑ Fixed scale

AM Stress

```
CAESS ProTOp  Ver: 3.1.9 Beta  Cyc: 0  VolPrt: 100.0
Date: 04/24/2015  Inp: TestCAD_7-7MFEs.ptop  Own: CAESS d.o.o.
```

## Optimization problems are EVERYWHERE

**In nature…**
**engineering…**
**physics-based modeling…**
**architecture…**
**manufacturing…**
**robotics…**
**machine learning…**

**Knowing how to solve optimization problems is very, very useful!**

## Continuous vs. Discrete Optimization

- **DISCRETE:**
  - **domain is a discrete set (e.g. integers)**
  - **Example: knapsack problem, which cities to visit on a trip**
    - **Basic strategy? Try all combinations! (exponential)**
    - **sometimes clever strategy (e.g., MST)**
    - **can sometimes turn discrete variables into continuous ones**
    - **more often, NP-hard (e.g., TSP)**
- **CONTINUOUS:**
  - **domain is not discrete (e.g., real numbers)**
  - **still many (NP-)hard problems, but also large classes of “easy” problems (e.g., convex)**
  - **Gradient information, if available, can be very useful**

## Optimization Problem in Standard Form

- **Can formulate most continuous optimization problems this way:**

“objective”: how much does solution x cost?

$$\min_{x \in \mathbb{R}^n} \quad f_0(x)$$

$$(f_i : \mathbb{R}^n \to \mathbb{R},\ i = 0, \ldots, m)$$

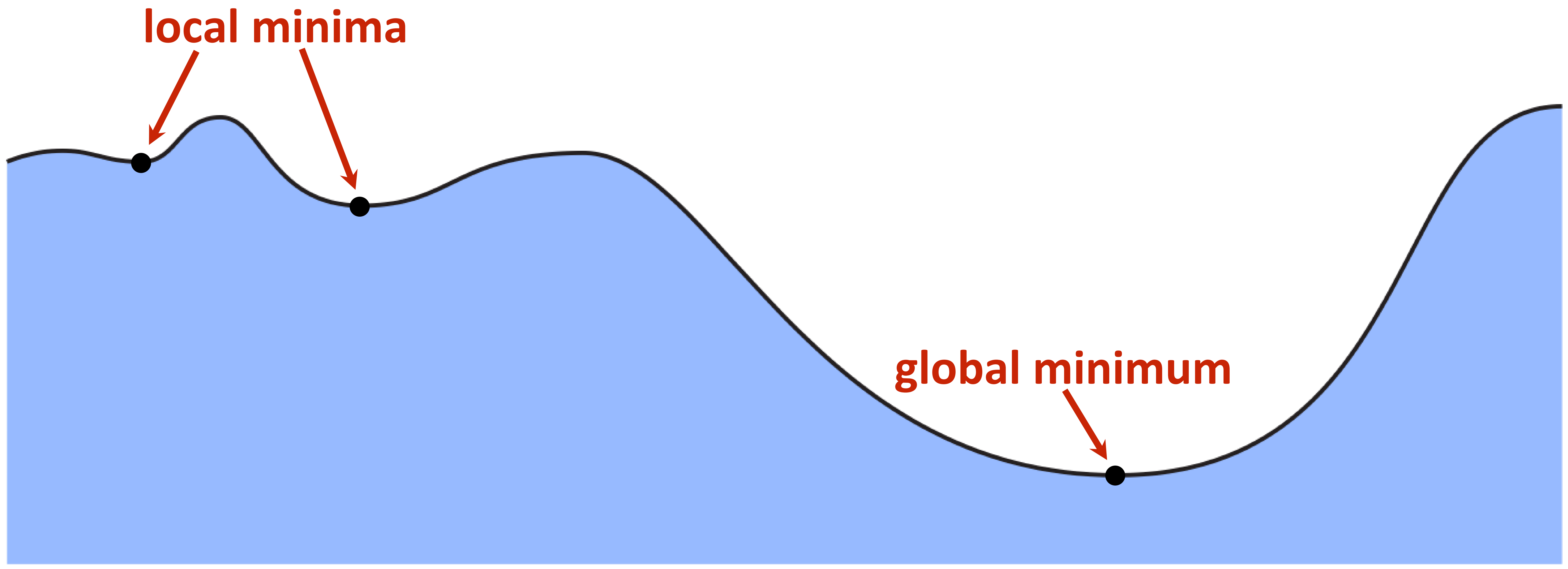
often (but not always) continuous, differentiable, ...

“constraints”: what must be true about x? (“x is *feasible*”)

- ***Optimal solution*** **x* has smallest value of f$_0$ among all feasible x**
- **Q: What if we want to *maximize* something instead?**
- **A: Just flip the sign of the objective!**
- **Q: What if we want *equality* constraints, rather than inequalities?**
- **A: Can include two constraints: g(x) ≤ c and g(x) ≤ -c**

# Local vs. Global Minima

- *Global* minimum is absolute best among all possibilities
- *Local* minimum is best "among immediate neighbors"



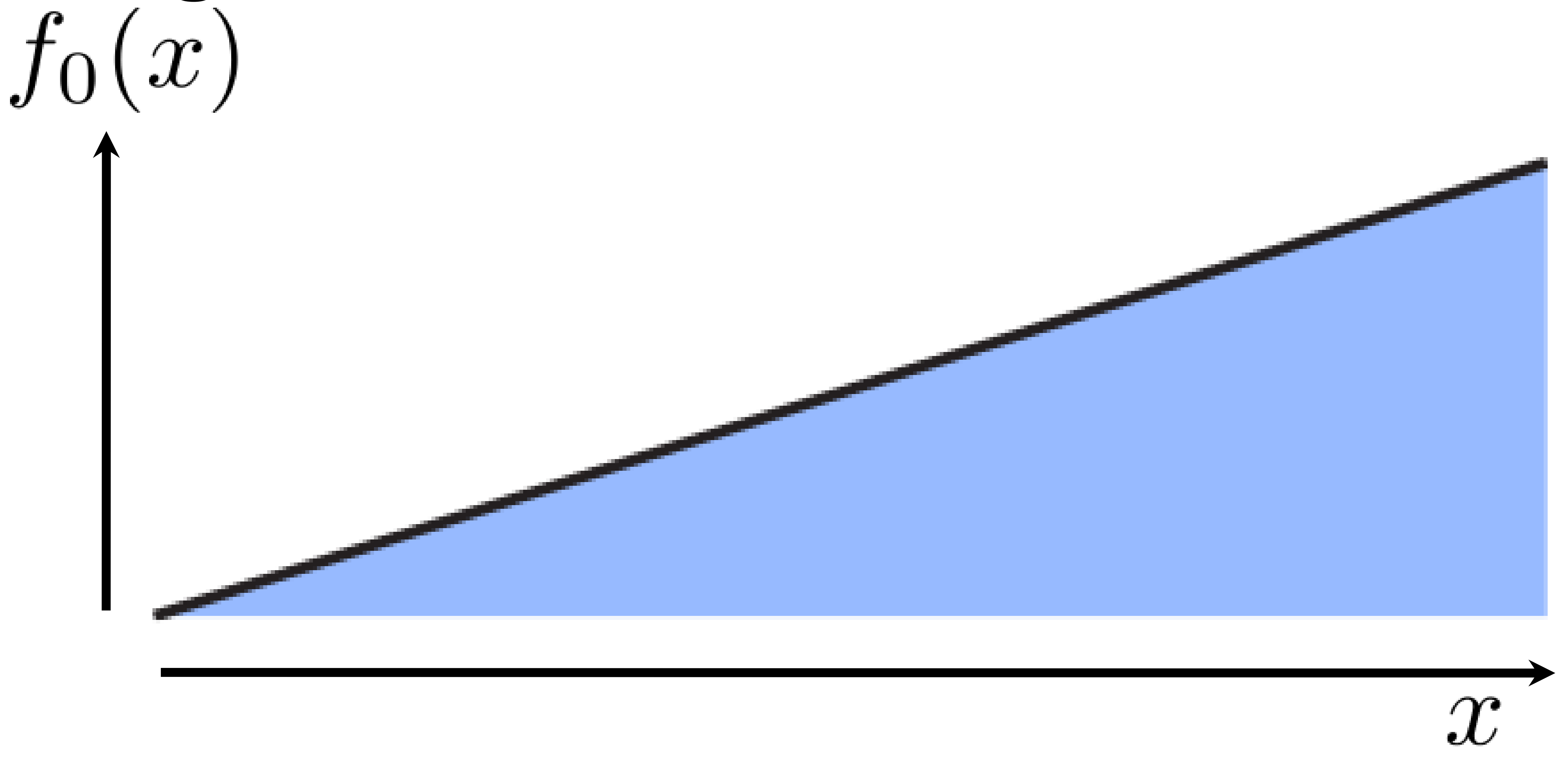
Philosophical question: does a local minimum *"solve"* the problem?

Depends on the problem! (E.g., evolution)

But sometimes, local minima can be really bad…

# Existence & Uniqueness of Minimizers

- **Already saw that (global) minimizer is not unique.**
- **Does it always exist? Why?**
- **Just consider all possibilities and take the smallest one, right?**



perfectly reasonable optimization *problem*

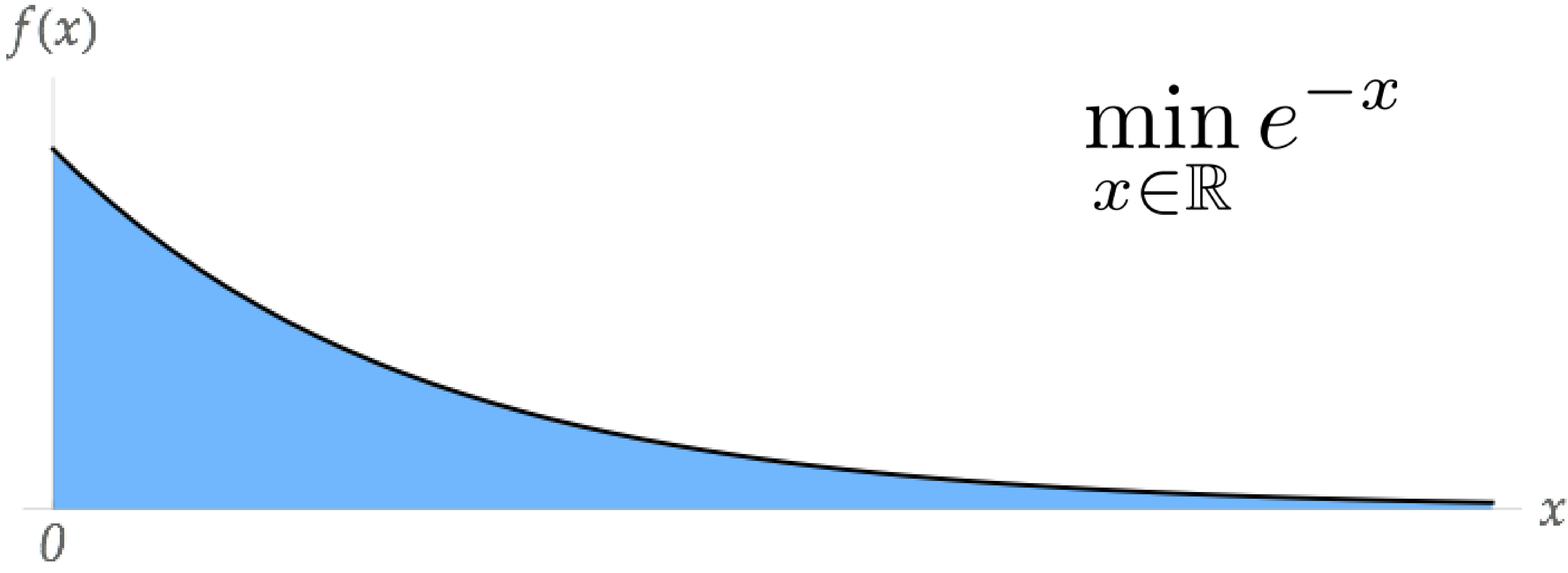
$$\min_{x \in \mathbb{R}} x$$

clearly has no *solution* (can always pick smaller x)

- **Not all objectives are bounded from below.**

# Existence & Uniqueness of Minimizers, cont.

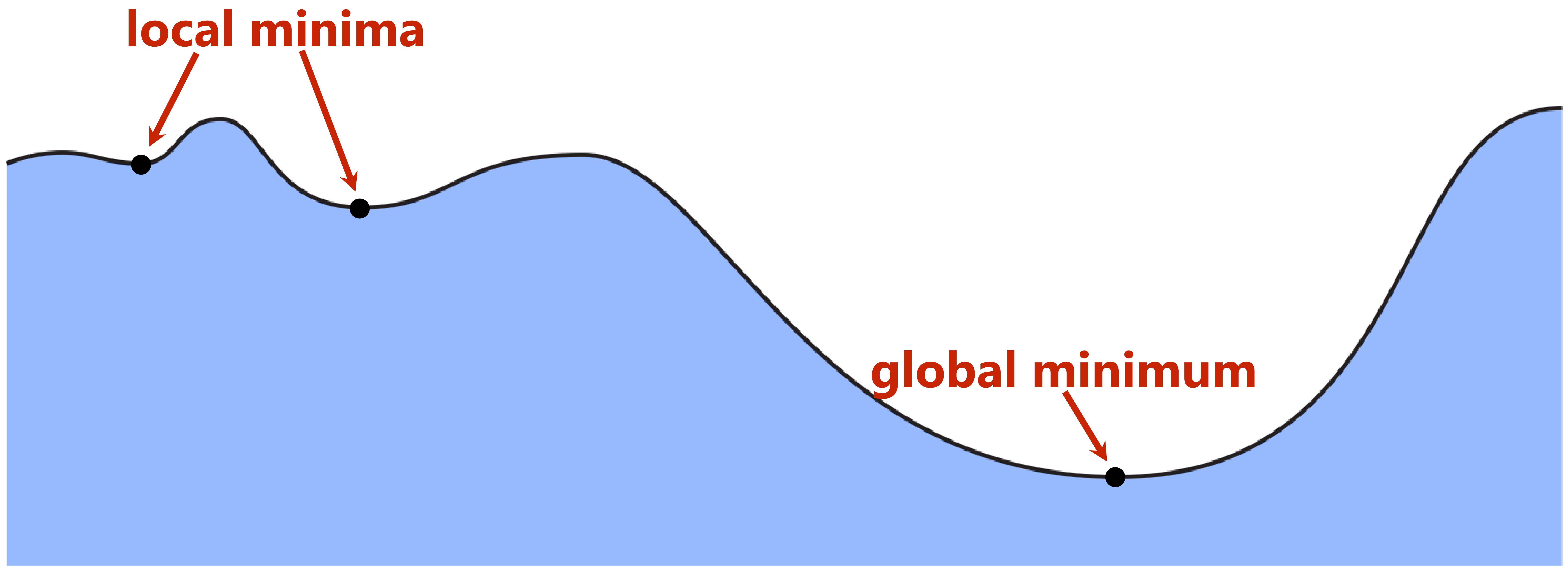
- **Even being bounded from below is not enough:**



- **No matter how big x is, we never achieve the lower bound (0)**
- **So when does a solution exist? Two *sufficient* conditions:**
- ***Extreme value theorem:* continuous objective & compact domain**
- ***Coercivity:* objective goes to +∞ as we travel (far) in any direction**

# Characterization of Minimizers

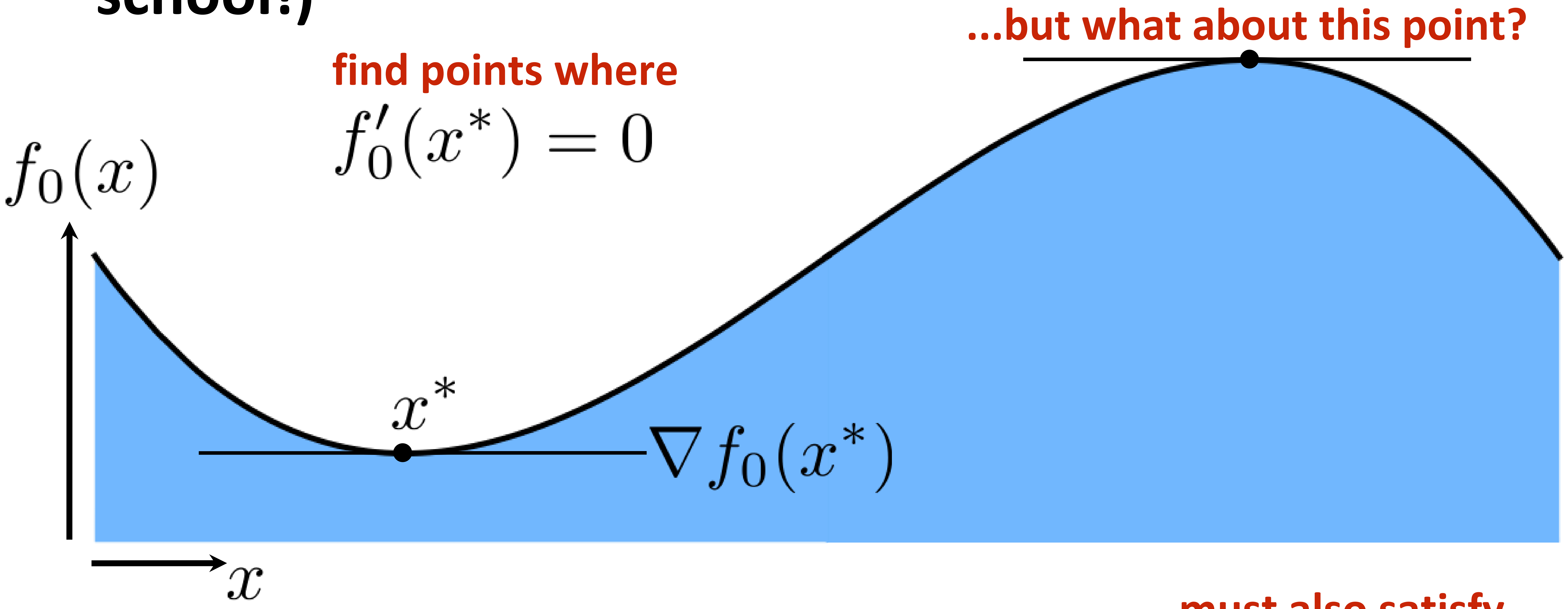
- Ok, so we have some sense of when a minimizer might *exist*
- But how do we know a given point x is a minimizer?



- Checking if a point is a global minimizer is (generally) hard
- But we can certainly test if a point is a local minimum (ideas?)
- (Note: a global minimum is also a local minimum!)

# Characterization of Local Minima

- Consider an objective $f_0$: R → R. How do you find a minimum?
- (Hint: you may have memorized this formula in high school!)



must also satisfy

$f_0''(x^*) \geq 0$

- Also need to check *second* derivative (how?)
- Make sure it's *positive*
- Ok, but what does this all mean for more general functions $f_0$?

# Optimality Conditions (higher dimensions)

- **In general, our objective is f0: R$^n$ → R**
- **How do we test for a local minimum?**
- **1st derivative becomes *gradient*; 2nd derivative becomes *Hessian***

$$\nabla f := \begin{bmatrix} \partial f / \partial x_1 \\ \vdots \\ \partial f / \partial x_n \end{bmatrix}$$

**GRADIENT**
**(measures "slope")**

$$\nabla^2 f := \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial f}{\partial x_n^2} \end{bmatrix}$$

**HESSIAN**
**(measures "curvature")**

- **Optimality conditions?**

***positive semidefinite (PSD)***
**(u$^T$Au ≥ 0 for all u)**

$$\nabla f_0(x^*) = 0$$

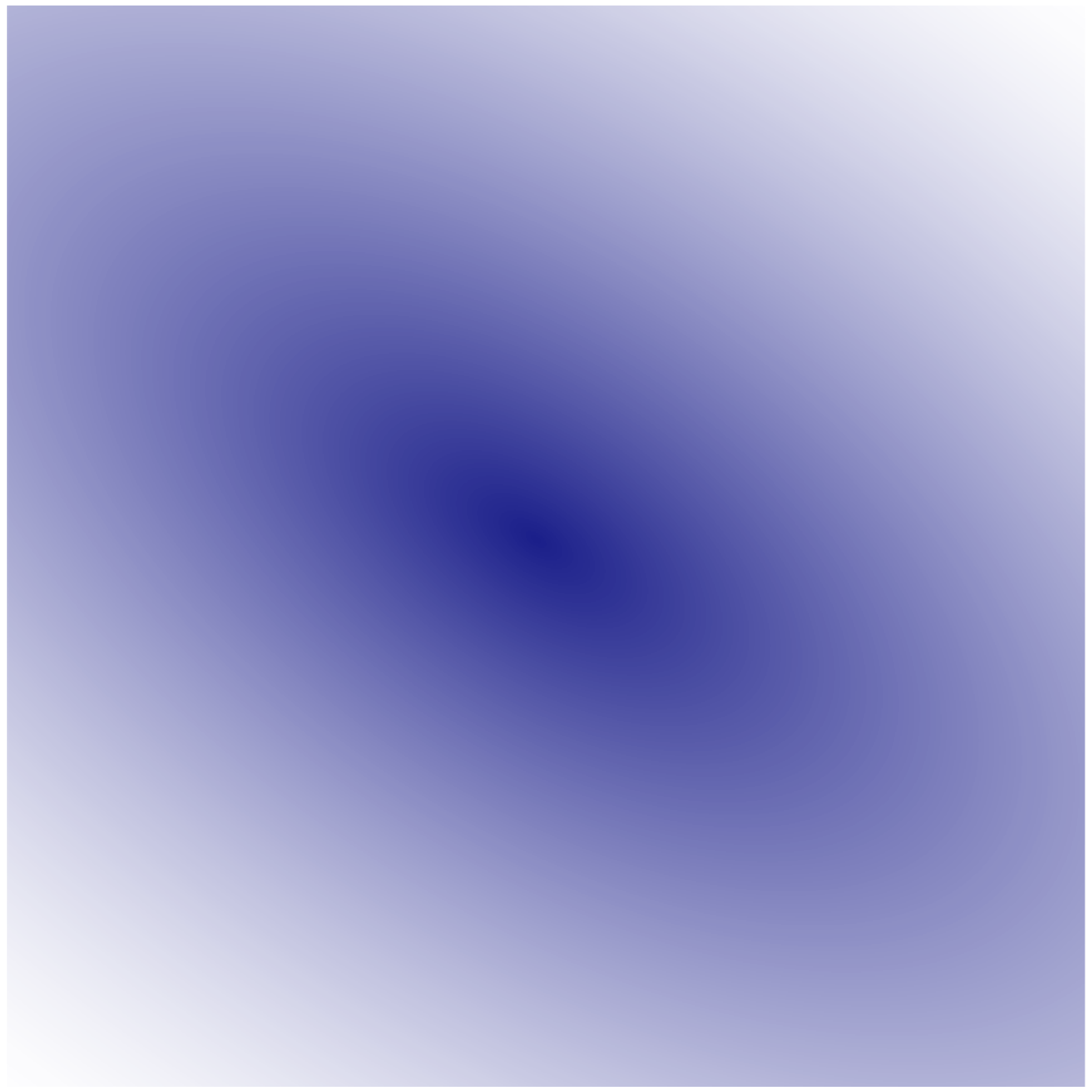
**1st order**

$$\nabla^2 f_0(x^*) \succeq 0$$

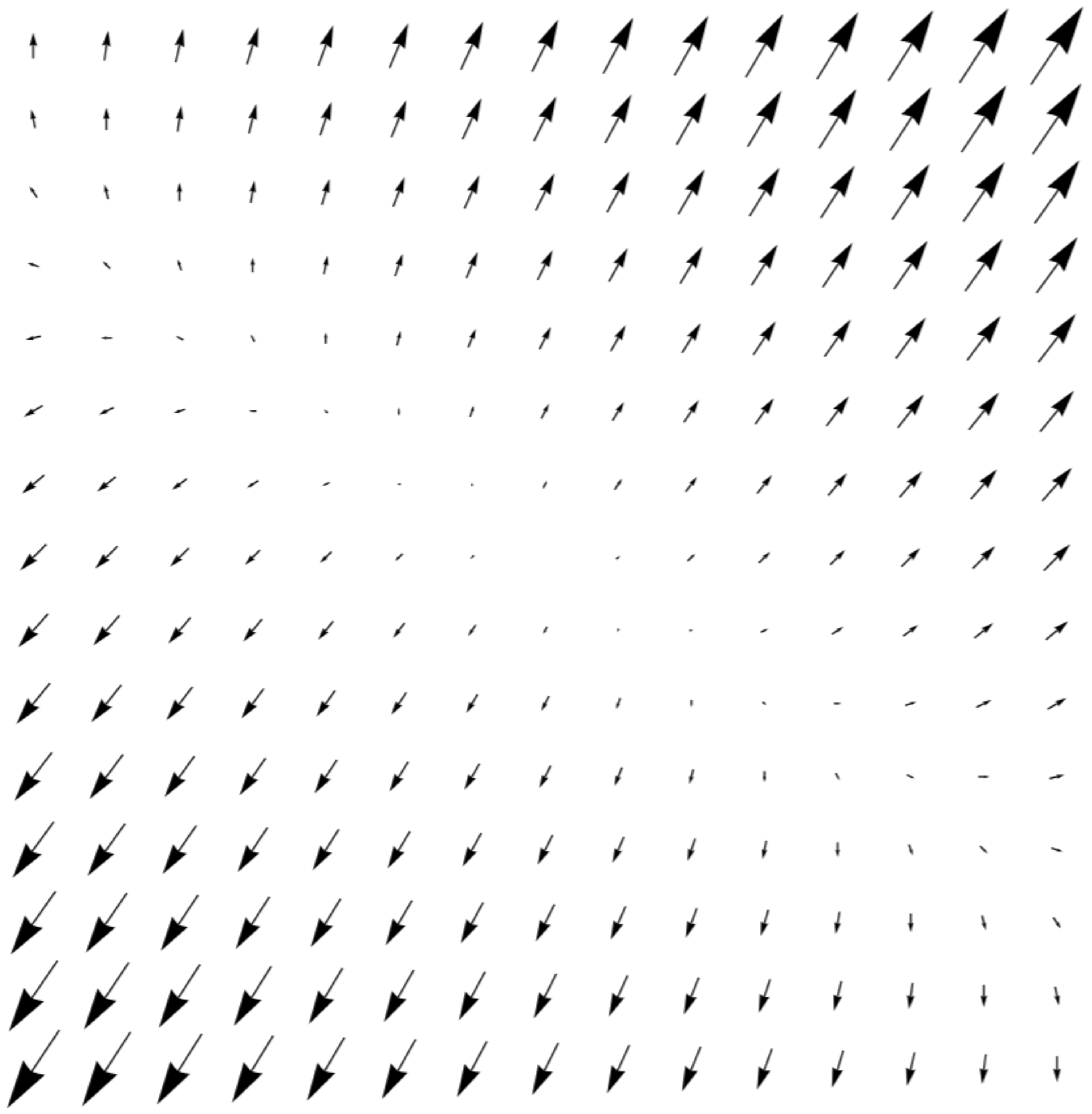
**2nd order**

# Gradient

- **Given a multivariate function, its gradient assigns a vector at each point**



$f(\mathbf{x})$



$\nabla f(\mathbf{x})$

## Hessian

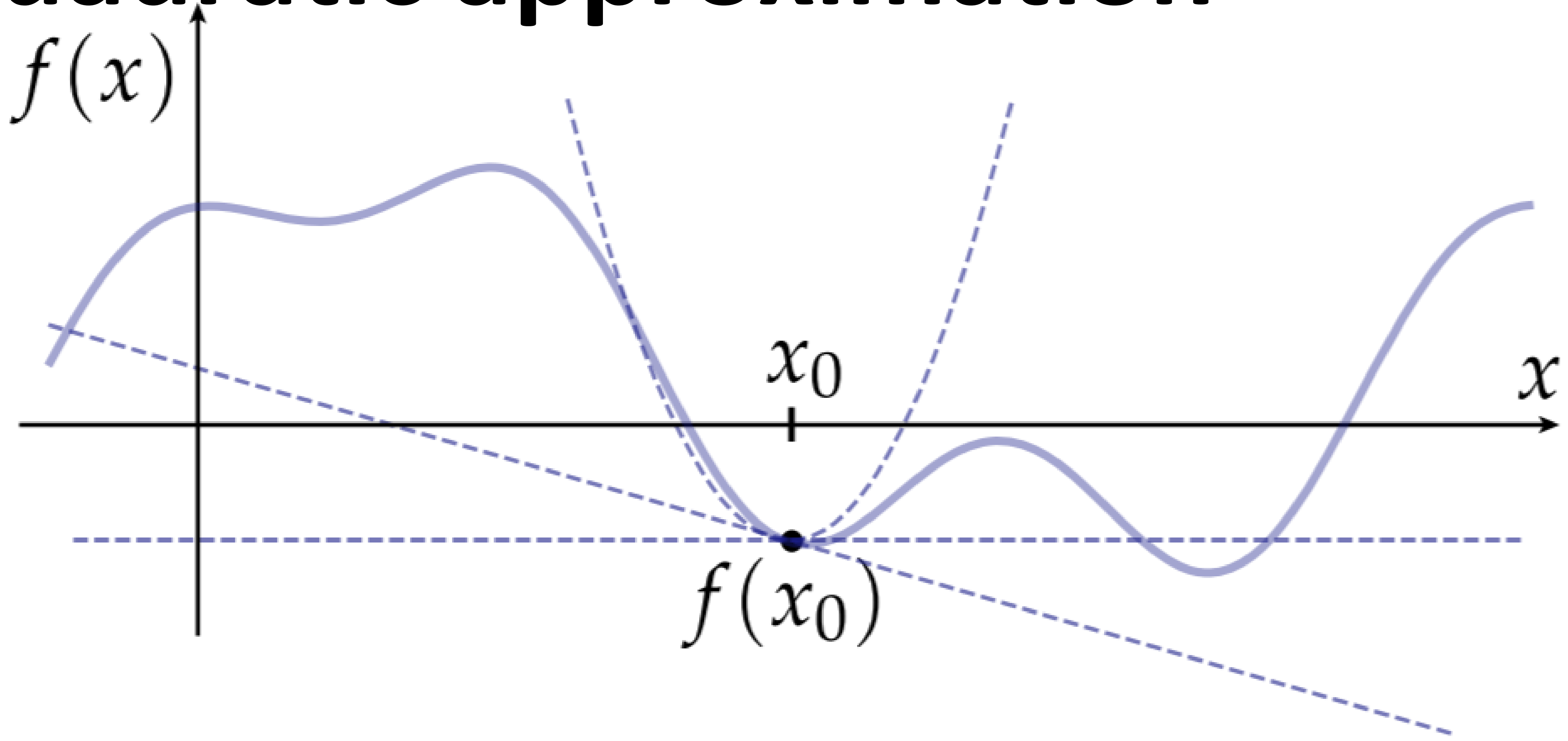
- **Jacobian of the gradient (matrix of second derivatives)**

$$\nabla^2 f := \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix}$$

- **Recall Taylor series**

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{(x-x_0)^2}{2!} f''(x_0) + \cdots$$

- **Gradient gives best linear approximation**
- **Hessian gives us best quadratic approximation**



# Hessian and Optimality conditions

- **Optimality conditions for multivariate optimization?**

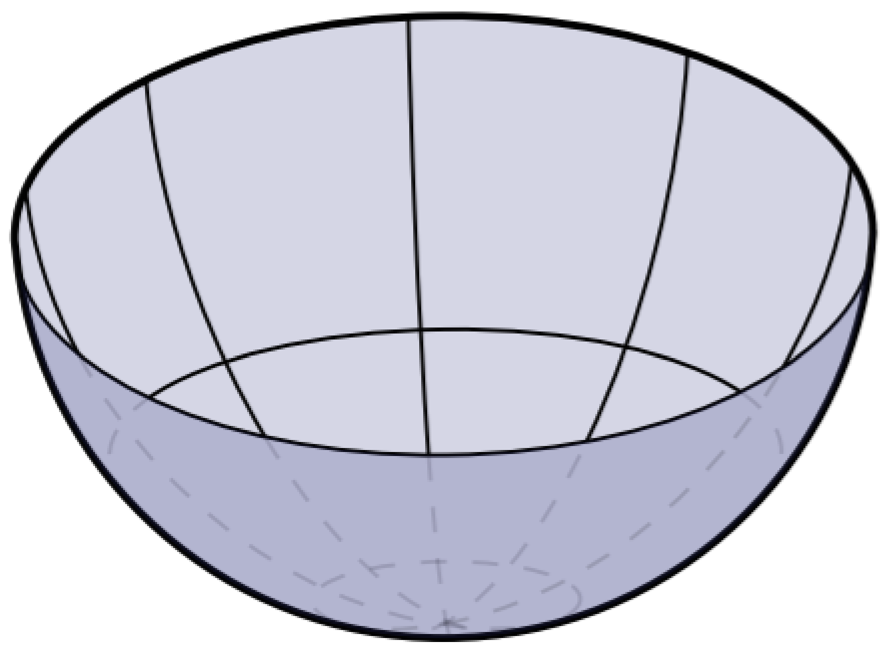
***positive semidefinite (PSD)***
**($u^TAu \geq 0$ for all u)**

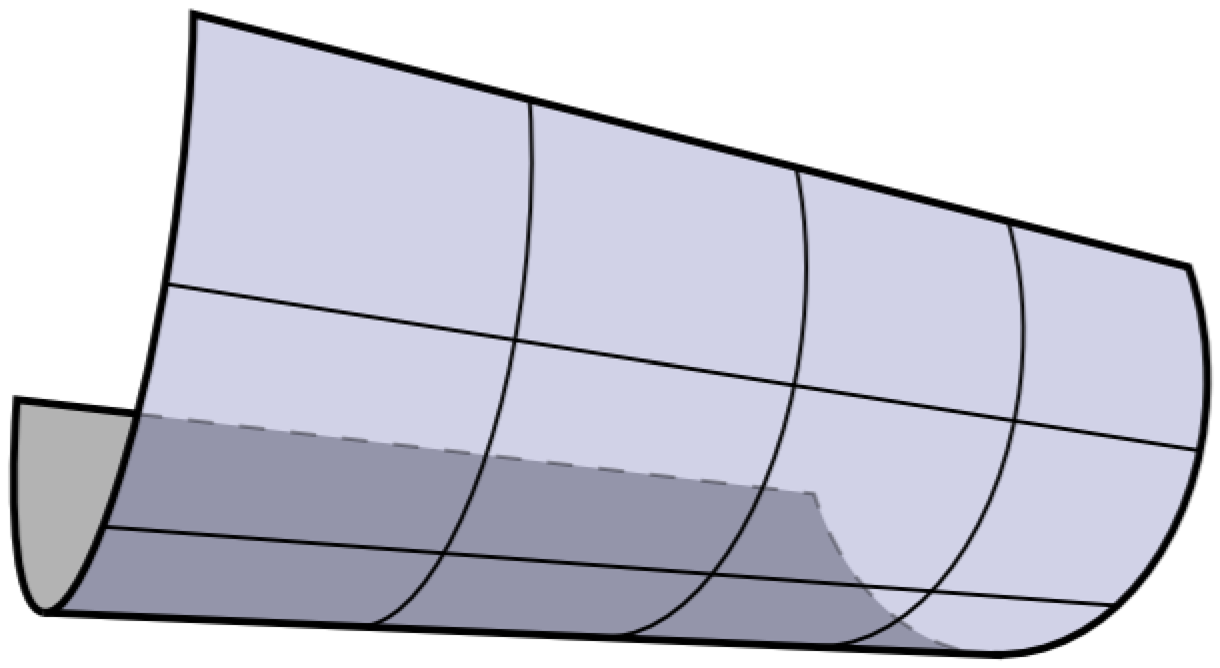
$$\nabla f_0(x^*) = 0$$

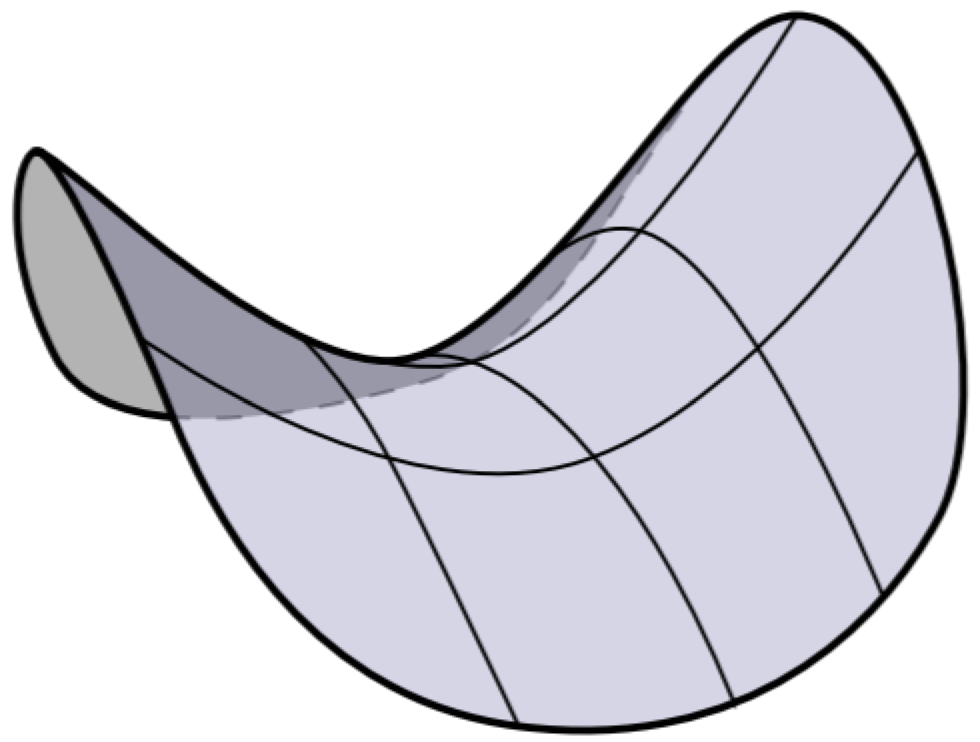
**1st order**

$$\nabla^2 f_0(x^*) \succeq 0$$

**2nd order**


**positive definite**


**positive semidefinite**


**indefinite**

# Gradients of Matrix-Valued Expressions

- **EXTREMELY useful to be able to differentiate matrix-valued expressions!**

For any two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and **symmetric** matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$:

| MATRIX DERIVATIVE | LOOKS LIKE |
|---|---|
| $\nabla_{\mathbf{x}}(\mathbf{x}^T\mathbf{y}) = \mathbf{y}$ | $\frac{d}{dx}xy = y$ |
| $\nabla_{\mathbf{x}}(\mathbf{x}^T\mathbf{x}) = 2\mathbf{x}$ | $\frac{d}{dx}x^2 = 2x$ |
| $\nabla_{\mathbf{x}}(\mathbf{x}^T A\mathbf{y}) = A\mathbf{y}$ | $\frac{d}{dx}axy = ay$ |
| $\nabla_{\mathbf{x}}(\mathbf{x}^T A\mathbf{x}) = 2A\mathbf{x}$ | $\frac{d}{dx}ax^2 = 2ax$ |
| $\dots$ | $\dots$ |

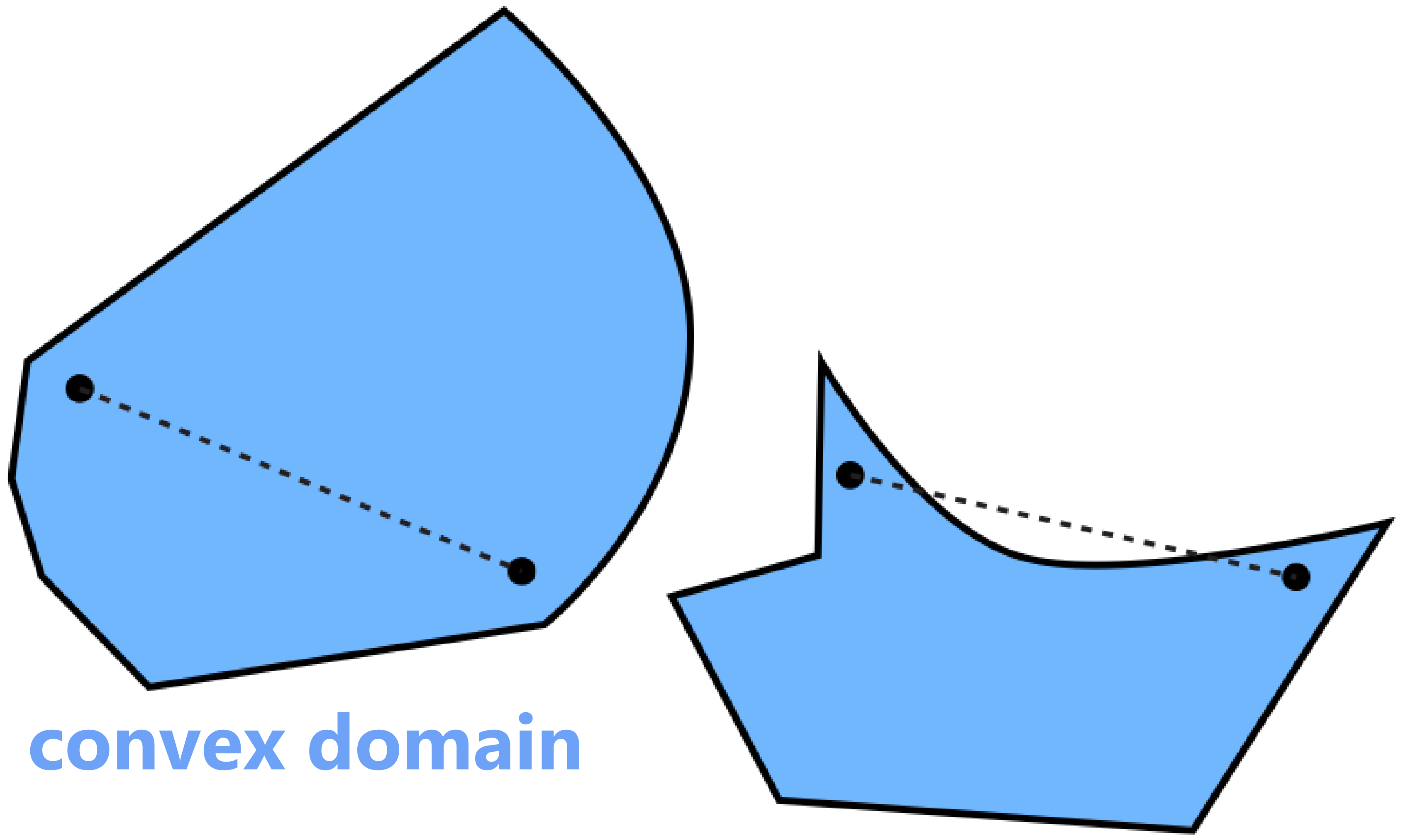
**Excellent resource: Petersen & Pedersen, "The Matrix Cookbook"**

- **At least once in your life, work these out meticulously in coordinates!**

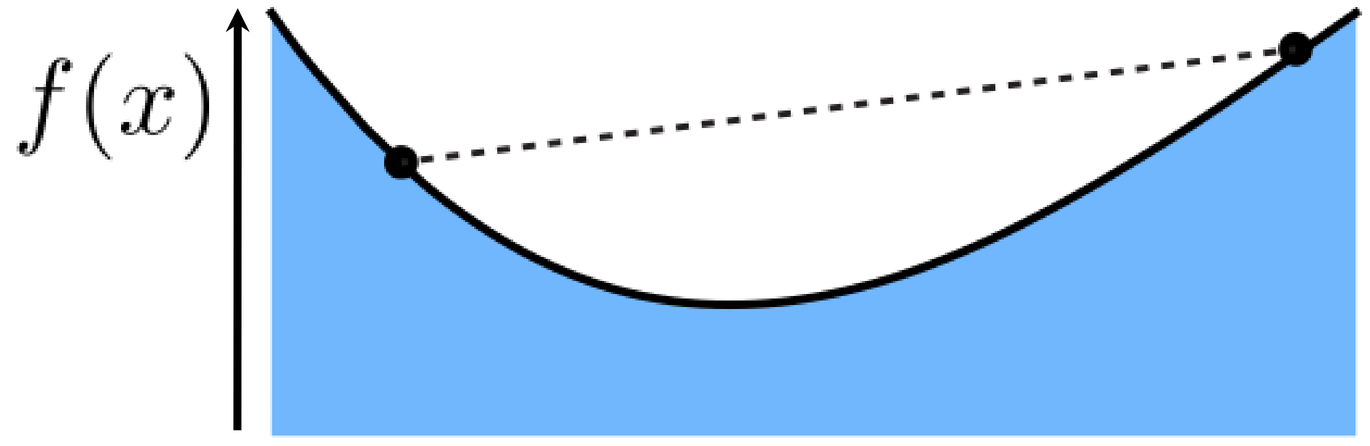
- **After that, use http://www.matrixcalculus.org/**

# Convex Optimization

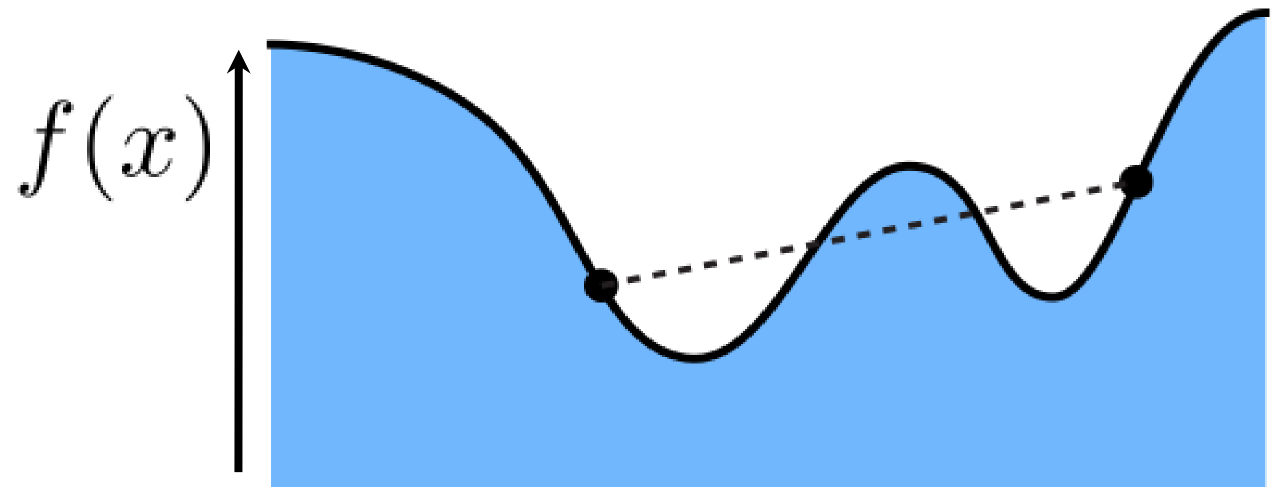
- Special class of problems that are almost always “easy” to solve (polynomial-time!)
- Problem is *convex* if it has a convex domain *and* convex objective


convex domain

noconvex domain



convex objective



nonconvex objective

- Why care about convex problems?
  - can make guarantees about solution (always the best)
  - doesn’t depend on initialization (strong convexity)
  - often quite efficient

# Convex Quadratic Objectives & Linear Systems

- Very important example: convex *quadratic* objective
- Can be expressed via positive-semidefinite (PSD) matrix:

$$f_0(x) = \frac{1}{2}x^T A x - x^T b, \;\; A \succeq 0$$

- Q: 1st-order optimality condition?
- Q: 2nd-order optimality condition?

just solve a linear system!

$$Ax = b$$

satisfied by definition

$$A \succeq 0$$

Sadly, life is not usually that easy.

How do we solve optimization problems in general?

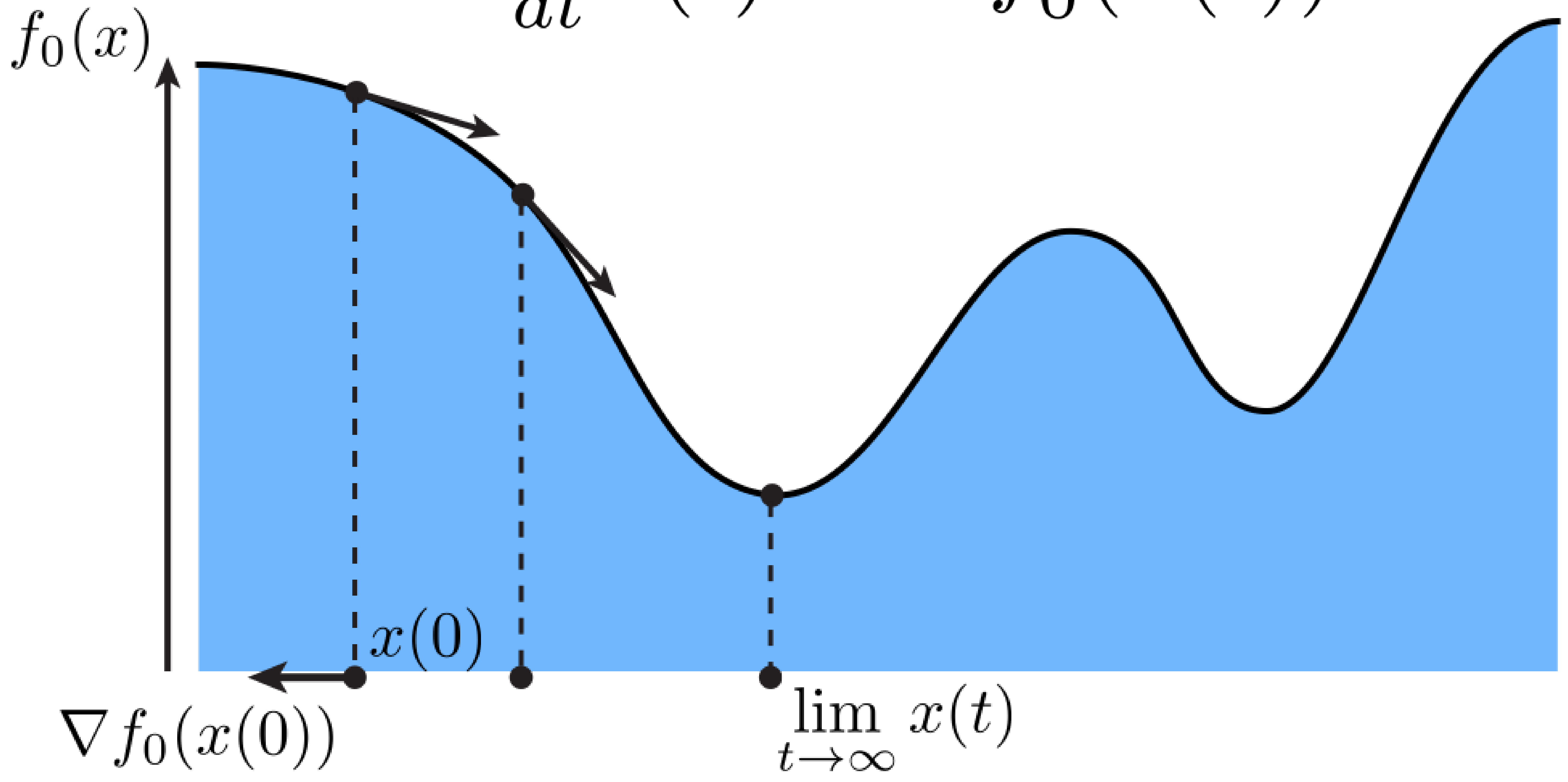
# Descent Methods

## An idea as old as the hills:

# Gradient Descent (1D)

- **Basic idea: follow the gradient "downhill" until it's zero**
- **(Zero gradient was our 1st-order optimality condition)**

$$\frac{d}{dt}x(t) = -f_0'(x(t))$$

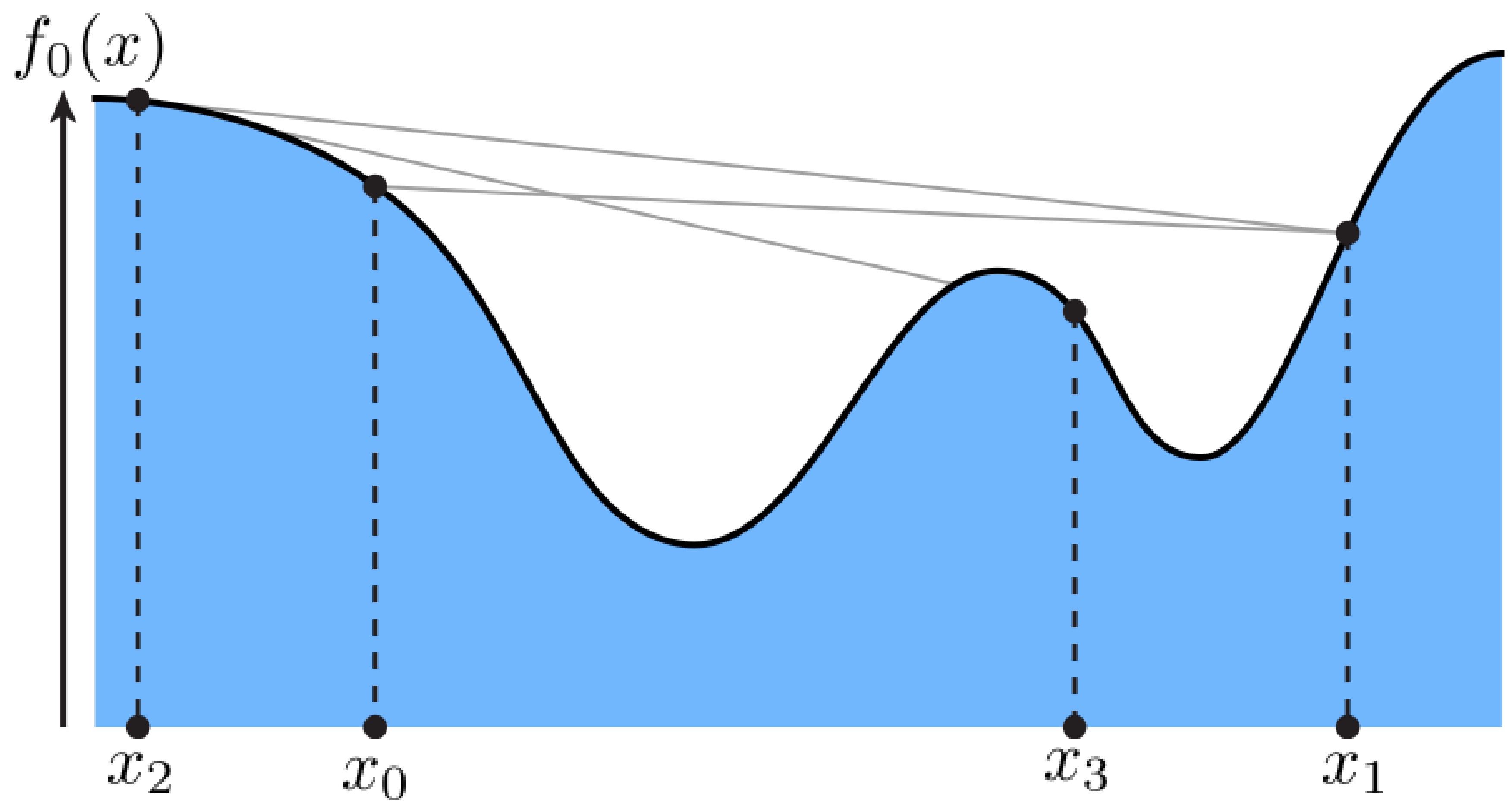


- **Do we always end up at a (global) minimum?**
- **How do we implement gradient descent in practice?**

# Gradient Descent Algorithm (1D)

- Simple update rule (go in direction that decreases objective):

$$x_{k+1} = x_k - \tau f_0'(x_k)$$

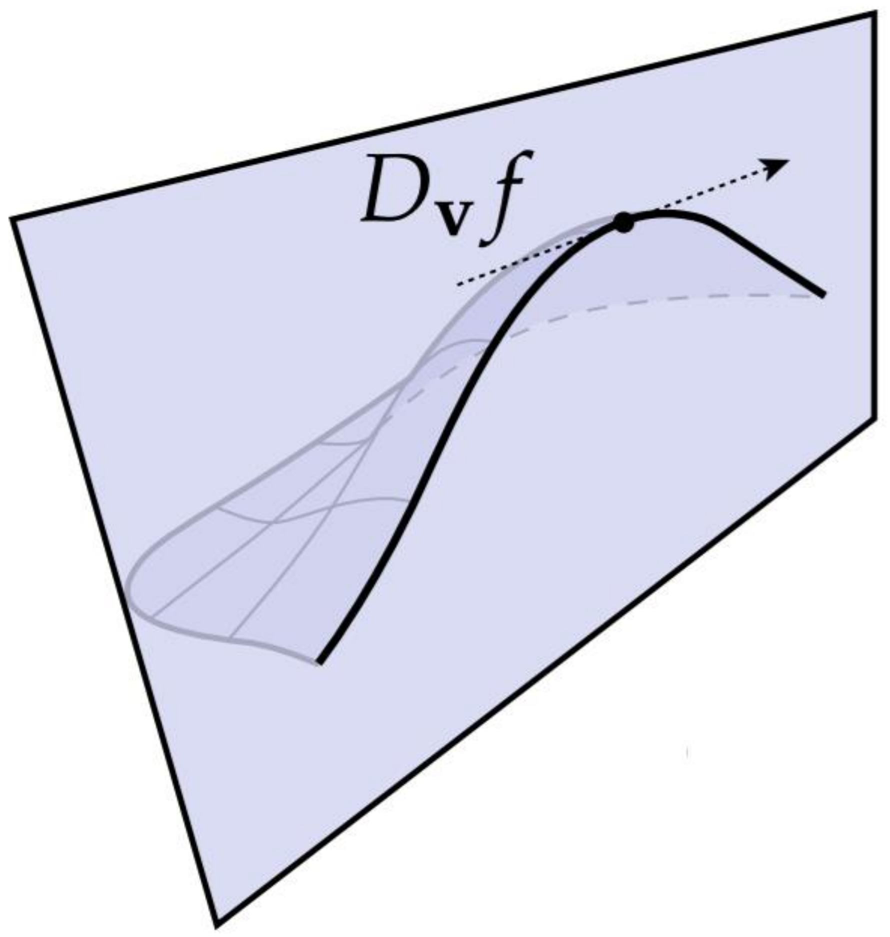


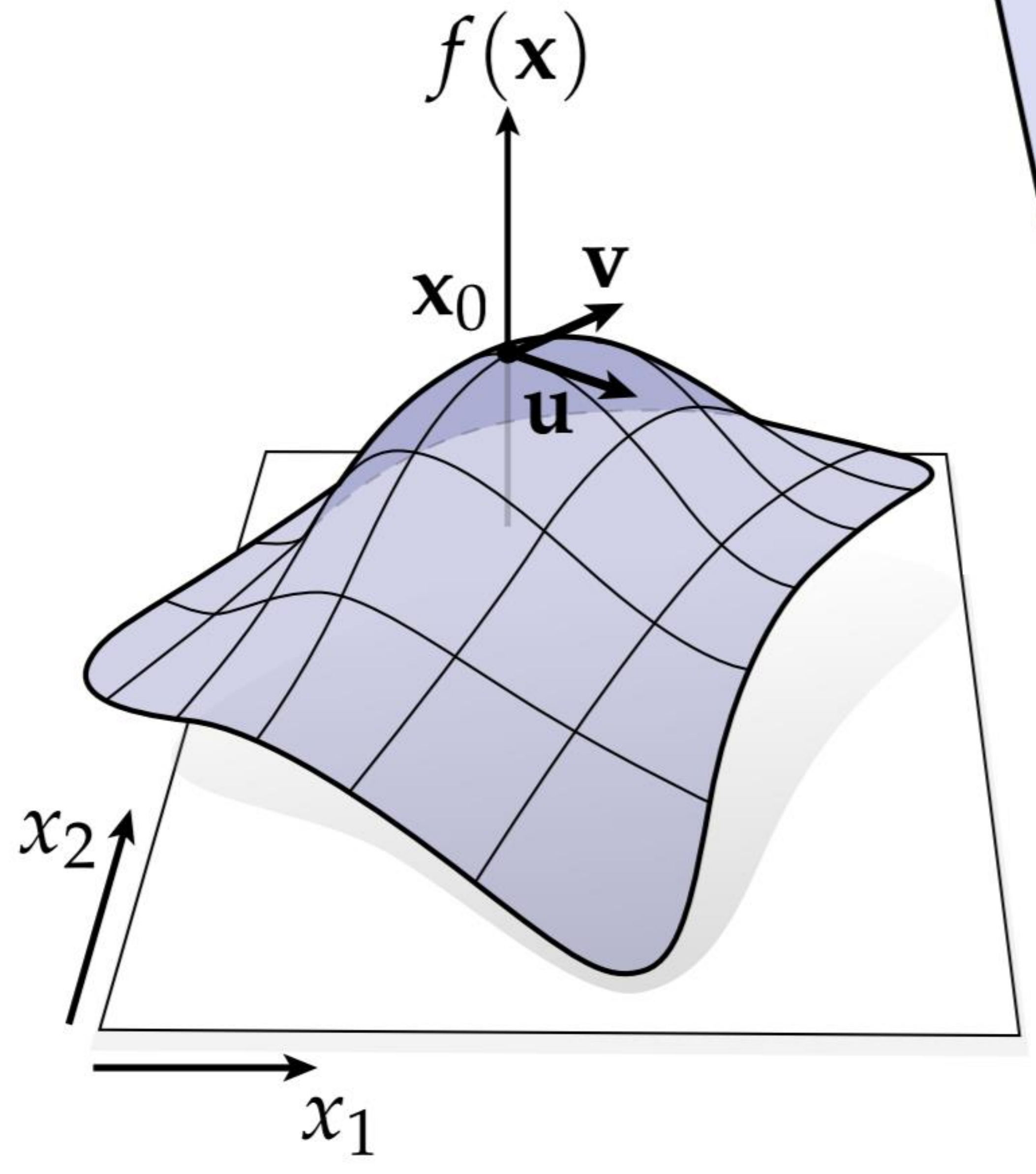
- Q: How far should we go in that direction?
- If we're not careful, we'll be zipping all over the place!
- Basic idea: use *"step control"* to determine step size based on value of objective & derivatives.
- A careful strategy (e.g., Armijo-Wolfe) can guarantee convergence at least to a *local* minimum.
- Oftentimes, a very simple strategy is used: make τ *really small!*

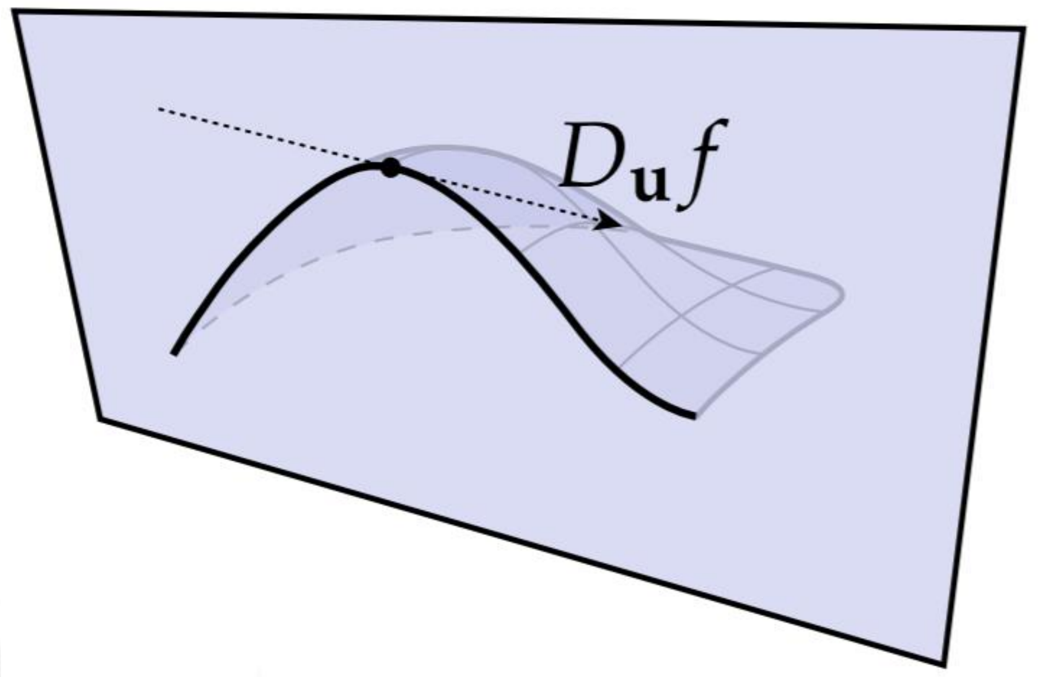
## How do we go about optimizing a function of multiple variables?

## Directional Derivative

- Suppose we have a function f(x1, x2)
  - Take a slice through this function along some direction
  - Then apply the usual derivative concept!
  - This is called the directional derivative
  - Which direction should we slice the function along?







take a small step along u

$$D_{\mathbf{u}}f(\mathbf{x}_0) := \lim_{\varepsilon \to 0} \frac{f(\mathbf{x}_0 + \varepsilon \mathbf{u}) - f(\mathbf{x}_0)}{\varepsilon}$$

## Directional Derivative

- Starting from Taylor's series

$$f(x_0 + \Delta x) \;\approx f(x_0) + \Delta x^T \nabla \mathrm{f}(\mathrm{x}_0) + \frac{1}{2} \Delta x^T \nabla^2 \mathrm{f}(\mathrm{x}_0) \Delta x$$

**easy to see that**

take a small step along u

$$D_{\mathbf{u}} f(\mathbf{x}_0) := \lim_{\varepsilon \to 0} \frac{f(\mathbf{x}_0 + \varepsilon \underset{\uparrow}{\mathbf{u}}) - f(\mathbf{x}_0)}{\varepsilon} = \frac{f(x_0) + \varepsilon \boldsymbol{u}^t \nabla f(\mathrm{x}_0) - f(x_0)}{\varepsilon}$$

$$D_{\boldsymbol{u}} f \;= \boldsymbol{u}^T \nabla f$$

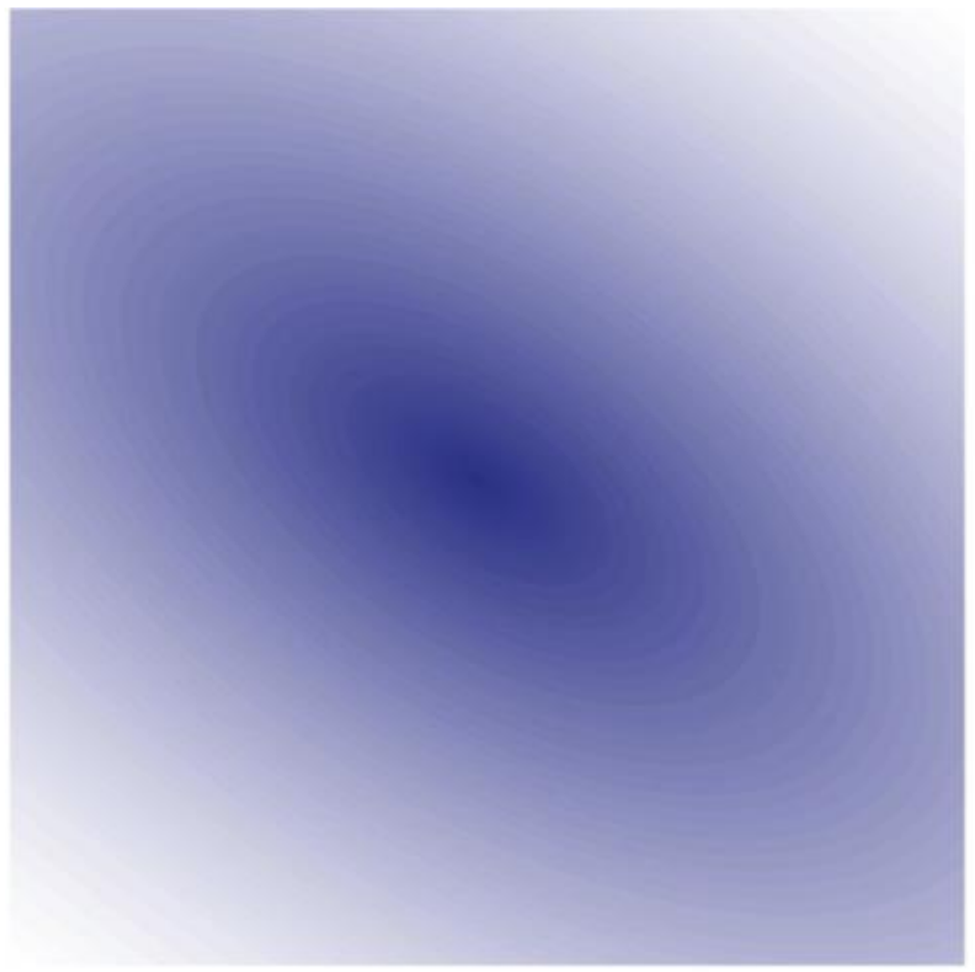
**Q: What does this mean?**

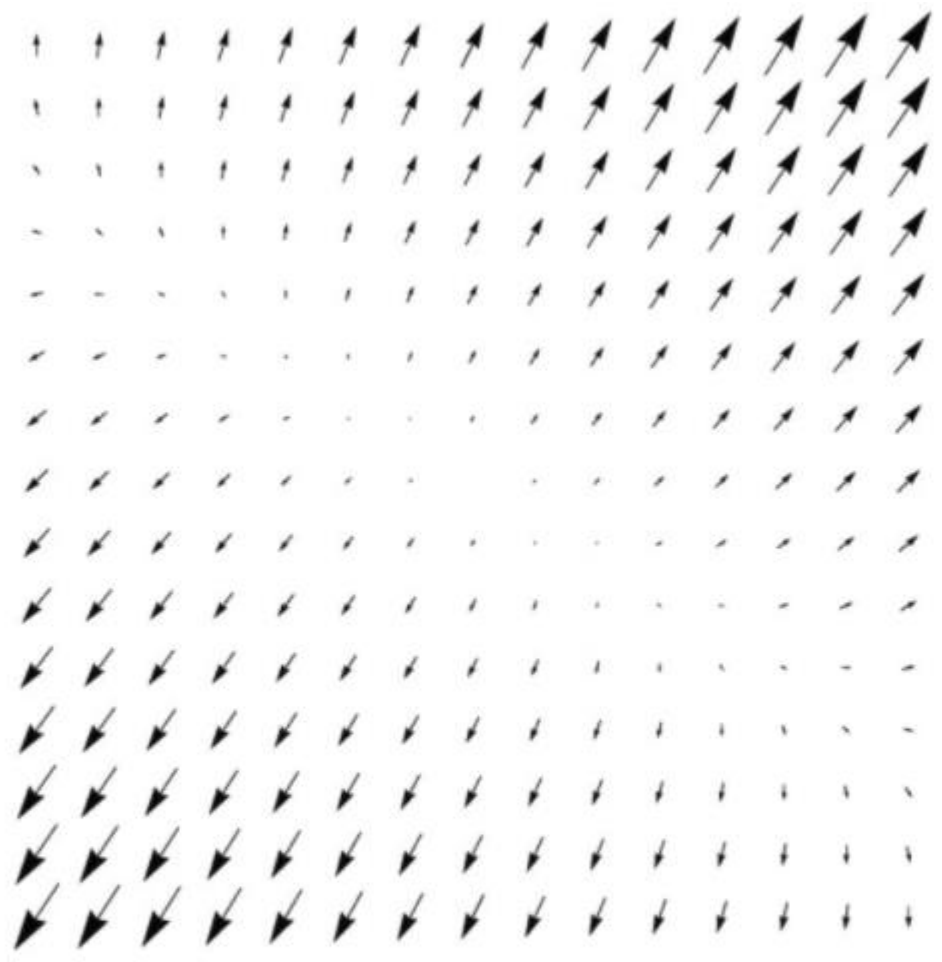
# Directional Derivative and the Gradient

- **Given a multivariate function $f(x)$, gradient assigns a vector $\nabla f(x)$ at each point**

- **Inner product between gradient and any unit vector gives directional derivative "along that direction"**

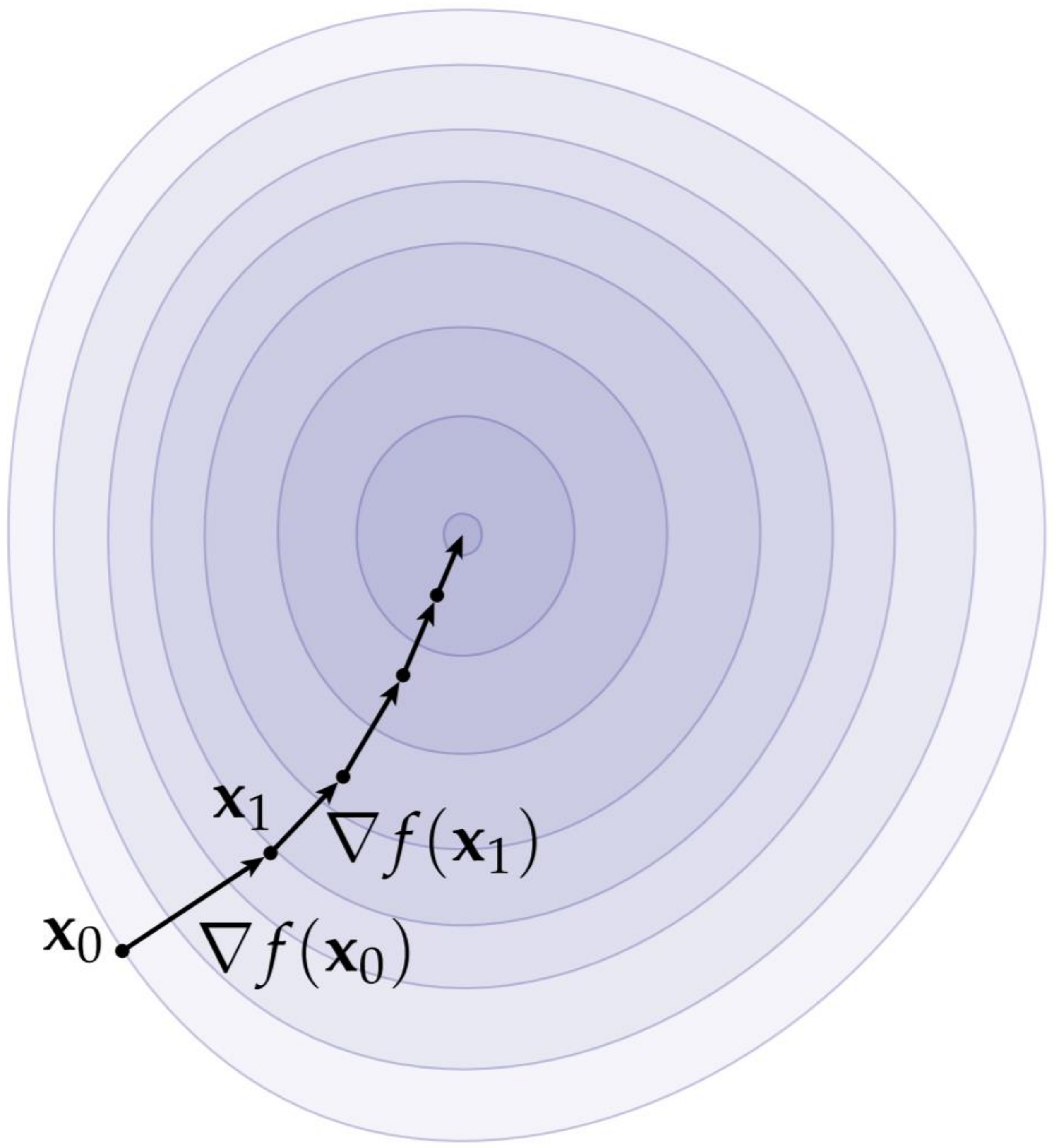
- **Out of all possible unit vectors, what is the one along which the function changes most?**


$f(\mathbf{x})$


$\nabla f(\mathbf{x})$

## Gradient points in direction of steepest ascent

- **Function value**
  - **gets largest if we move in direction of gradient**
  - **doesn't change if we move orthogonally (gradient is perpendicular to isolines)**
  - **decreases *fastest* if we move exactly in opposite direction**



## Gradient in coordinates

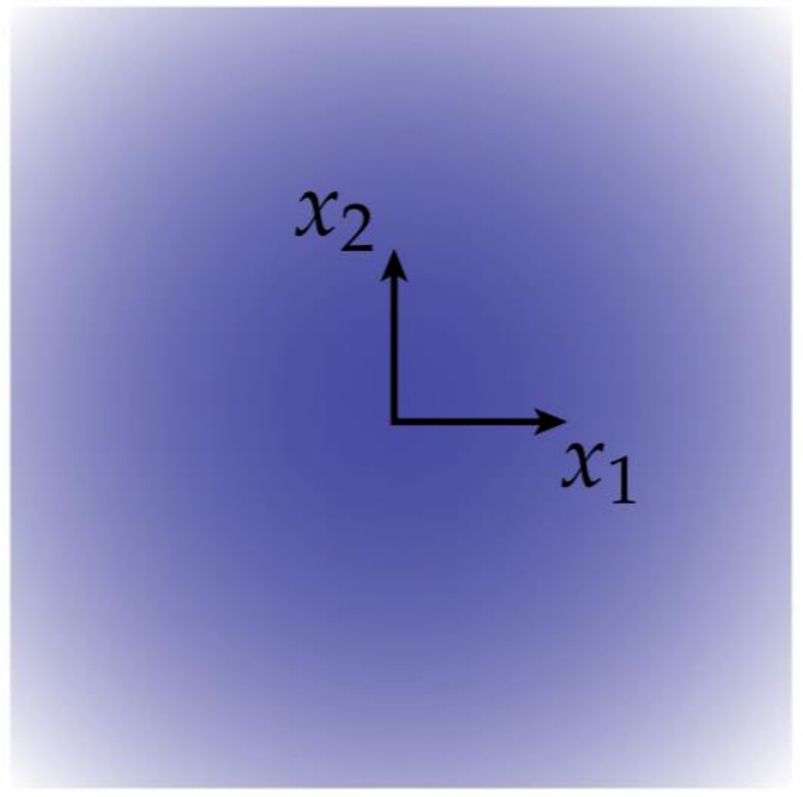
- **Most familiar definition: list of partial derivatives**

$$f(\mathbf{x}) := x_1^2 + x_2^2$$

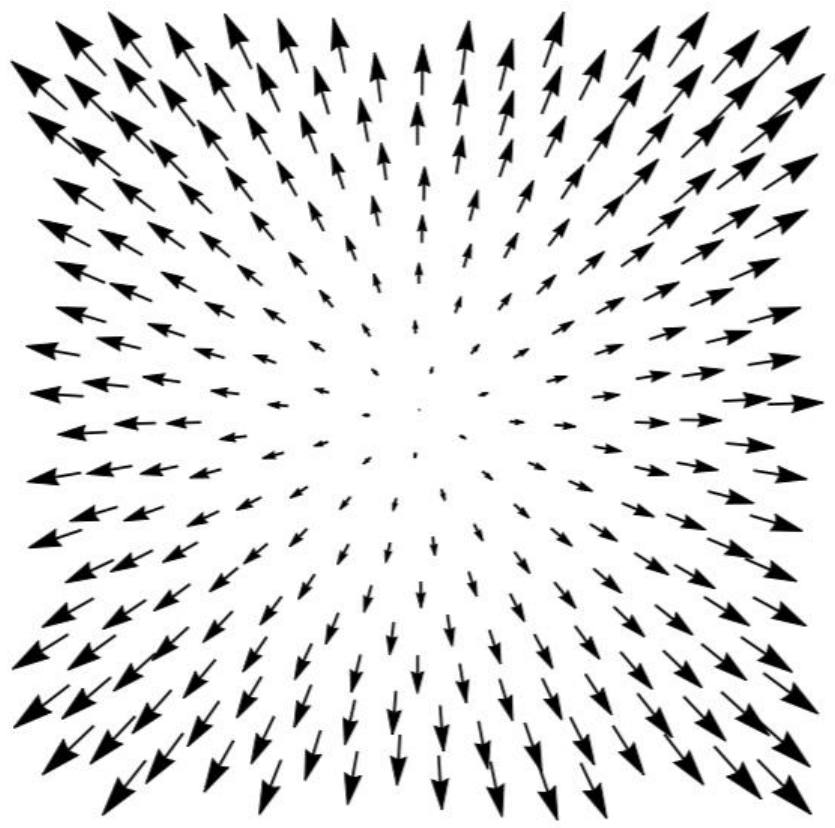
$$\frac{\partial f}{\partial x_1} = \frac{\partial}{\partial x_1} x_1^2 + \frac{\partial}{\partial x_1} x_2^2 = 2x_1 + 0$$

$$\frac{\partial f}{\partial x_2} = \frac{\partial}{\partial x_2} x_1^2 + \frac{\partial}{\partial x_2} x_2^2 = 0 + 2x_2$$

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} = 2\mathbf{x}$$



$f(\mathbf{x})$

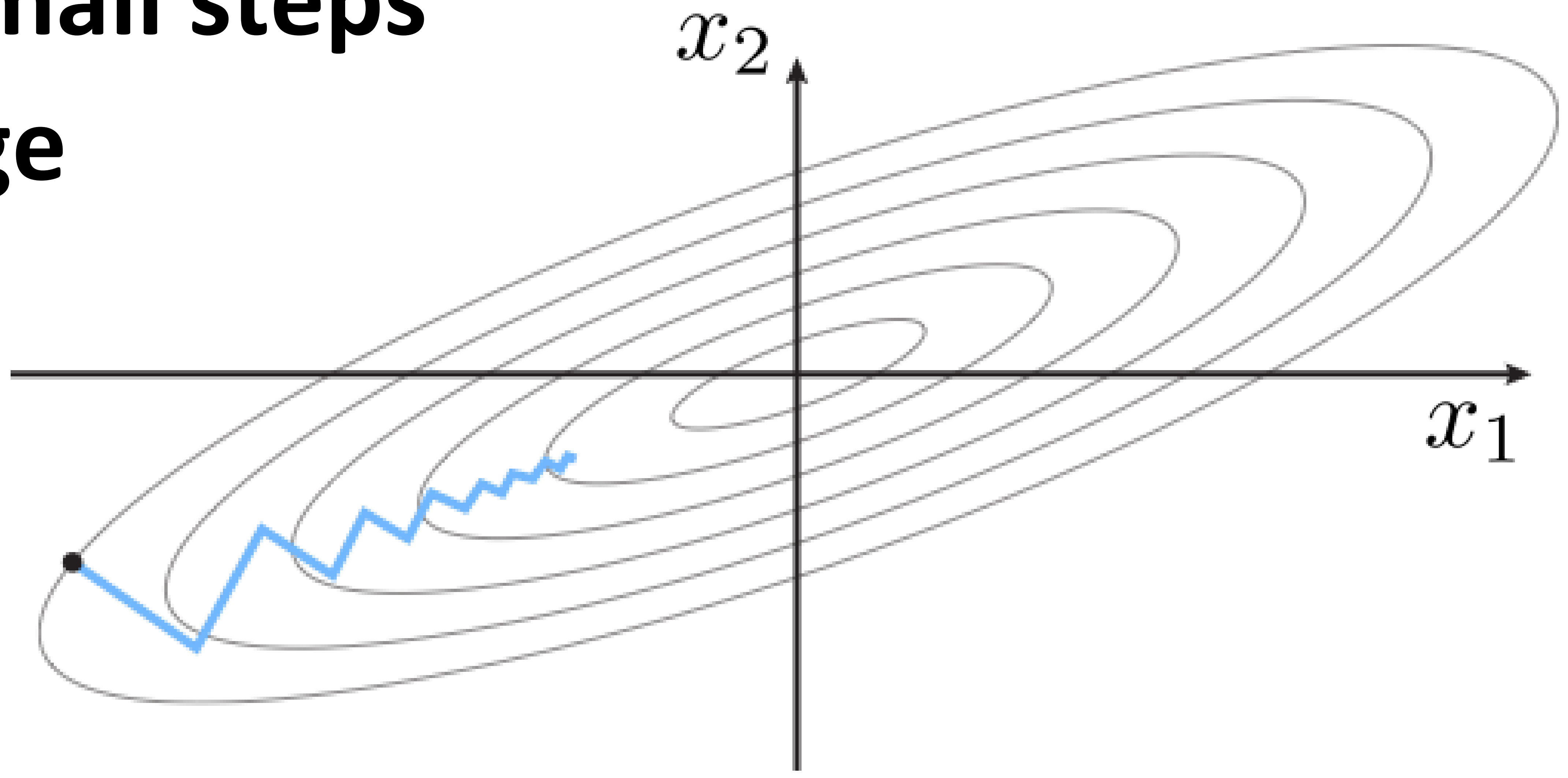

$\nabla f(\mathbf{x})$

## Gradient Descent Algorithm (nD)

**Q: What's the corresponding update in higher dimensions?**

$$x_{k+1} = x_k - \tau \nabla f_0(x_k)$$

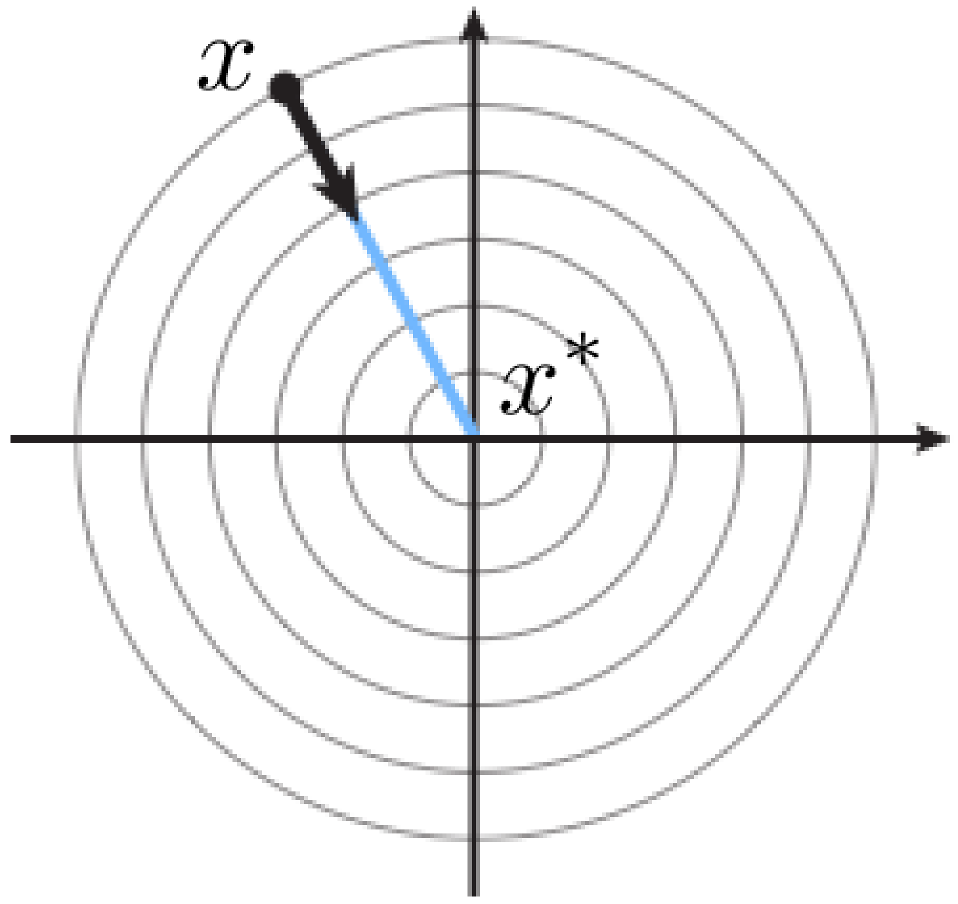
- **Basic challenge in nD:**
  - **solution can "oscillate"**
  - **takes many, many small steps**
  - **very slow to converge**



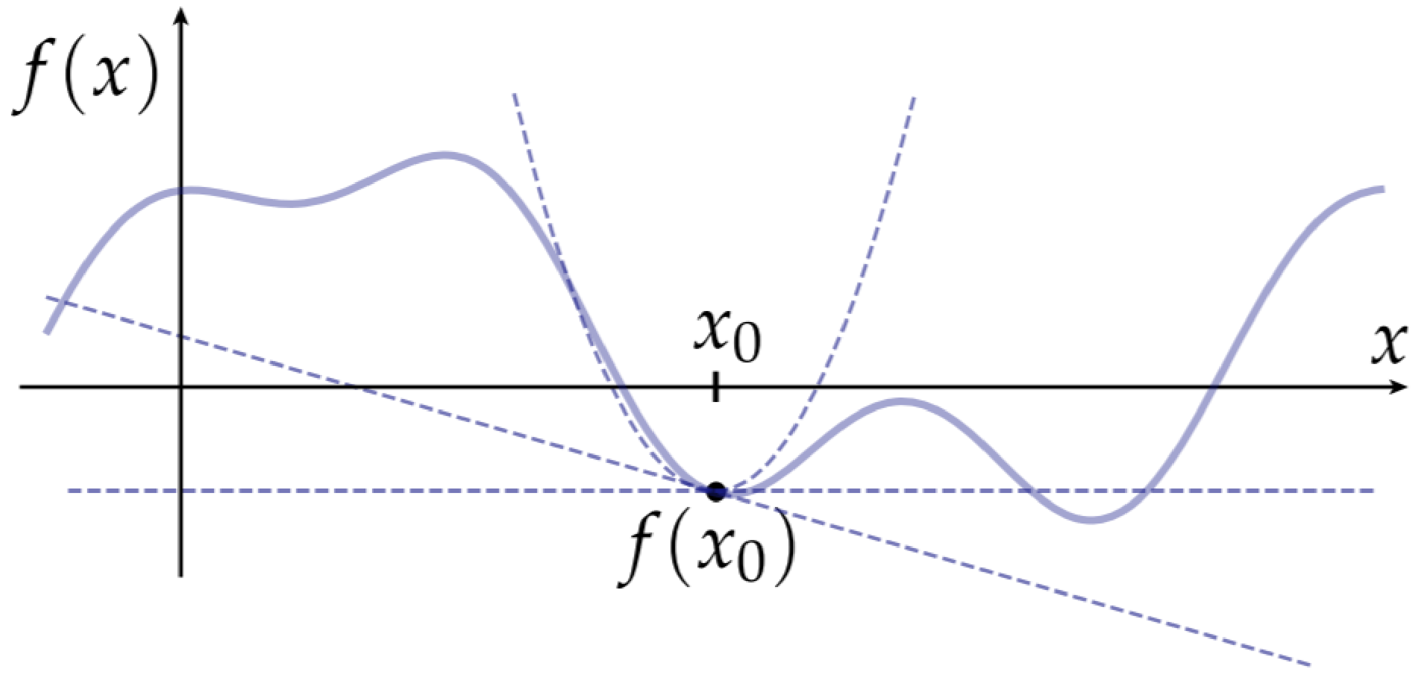
# Higher Order Descent

- **General idea: apply a coordinate transformation so that the local energy landscape looks more like a "round bowl"**
- **Gradient now points directly toward nearby minimizer**
- **Most basic strategy: Newton's method:**

$$x_{k+1} = x_k - \tau \underbrace{(\nabla^2 f_0(x_k))^{-1}}_{\text{Hessian inverse}} \overbrace{\nabla f_0(x_k)}^{\text{gradient}}$$



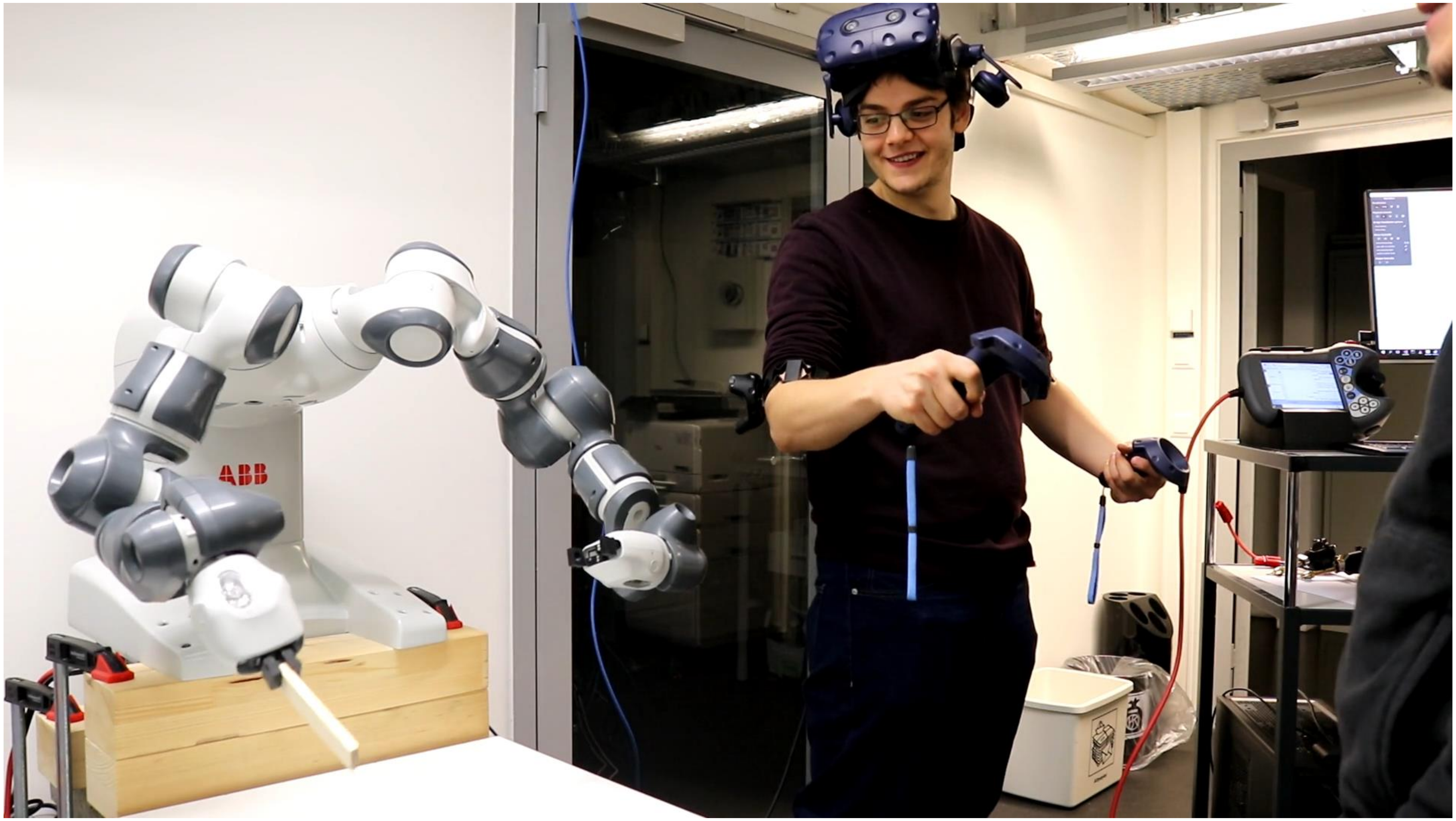
- **Another way to think about it: "pretend" the function is quadratic, solve and repeat…**



## Newton's method and beyond…

- Great for convex problems (even proofs about # of steps!)
- For nonconvex problems, need to be more careful
- In general, nonconvex optimization is a *BLACK ART*
- That you should try to master…

## An example: Optimization-based inverse kinematics



## An example: optimization-based IK

- **Basic idea behind IK algorithm:**
  - **write down distance between final point and "target" and set up an objective**
  - **compute gradient with respect to angles**
  - **apply gradient descent**

- **Objective?**

$$\boldsymbol{f_0(\theta) = \frac{1}{2}(x(\theta) - \tilde{x})^T(x(\theta) - \tilde{x})}$$

- **Constraints?**
  - **We could limit joint angles**