Introduction to Optimization

Computer Graphics
CMU 15-362/15-662

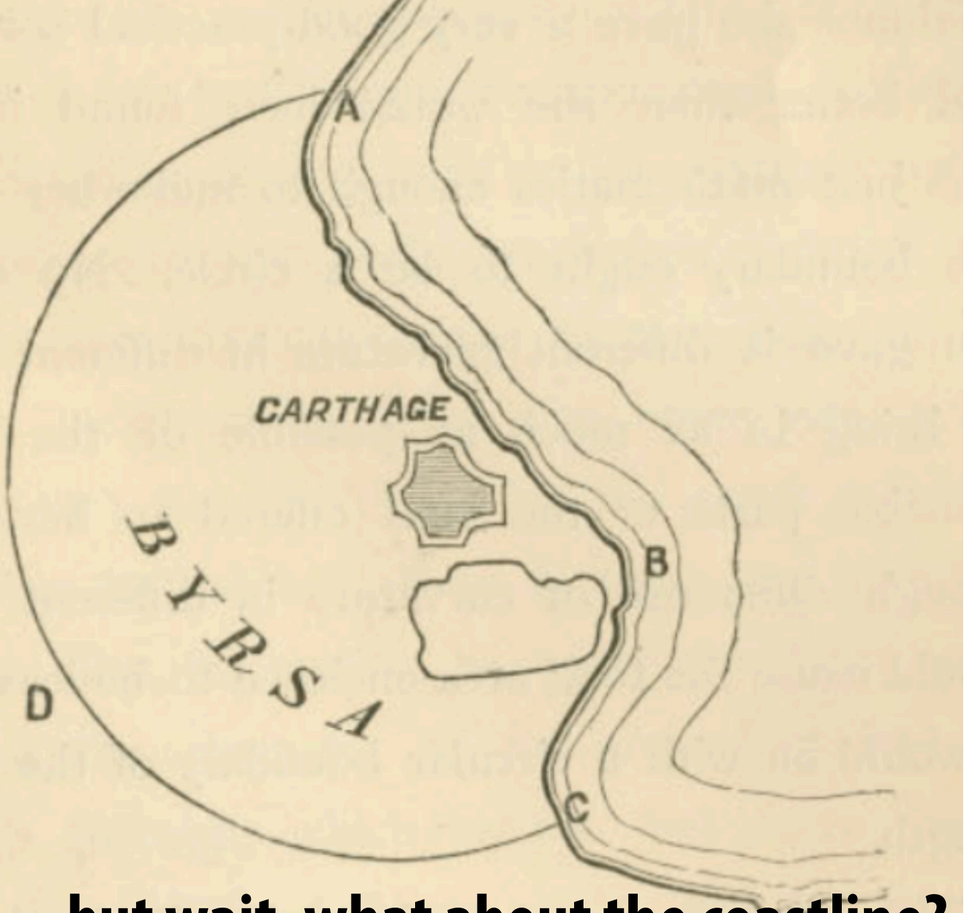
What is an optimization problem?

- Natural human desire: find the best solution among all possibilities (subject to certain constraints)
- E.g., cheapest flight, shortest route, tastiest restaurant ...
- Has been studied since antiquity, e.g., isoperimetric problem:

"The first optimization problem known in history was practically solved by Dido, a clever Phoenician princess, who left her Tyrian home and emigrated to North Africa, with all her property and a large retinue, because her brother Pygmalion murdered her rich uncle and husband Acerbas, and plotted to defraud her of the money which he left. On landing in a bay about the middle of the north coast of Africa she obtained a grant from Hiarbas, the native chief of the district, of as much land as she could enclose with an ox-hide. She cut the ox-hide into an exceedingly long strip, and succeeded in enclosing between it and the sea a very valuable territory on which she build Carthage."

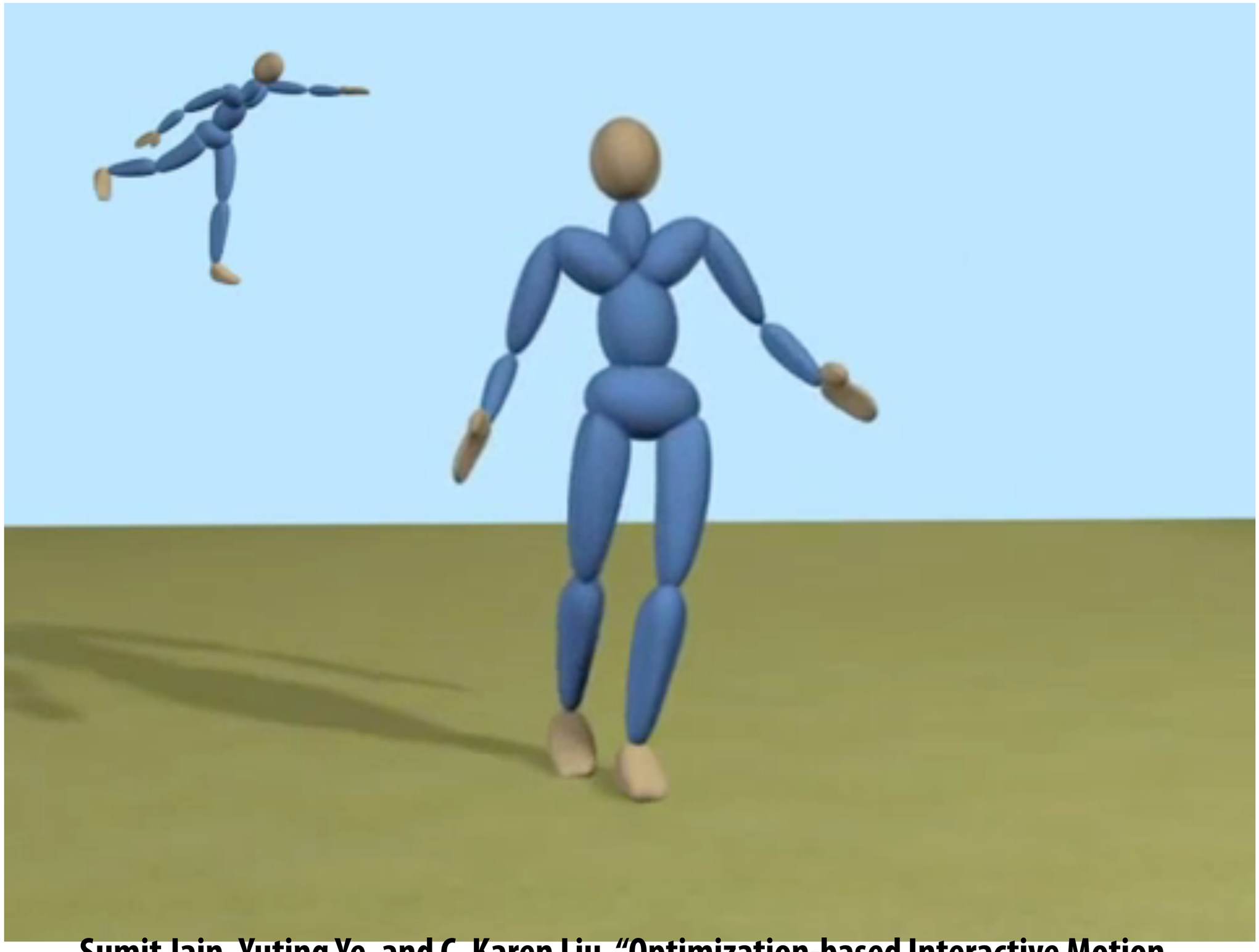
—Lord Kelvin, 1893

"Obvious" solution is a circle...



...but wait, what about the coastline?

Optimization in Graphics

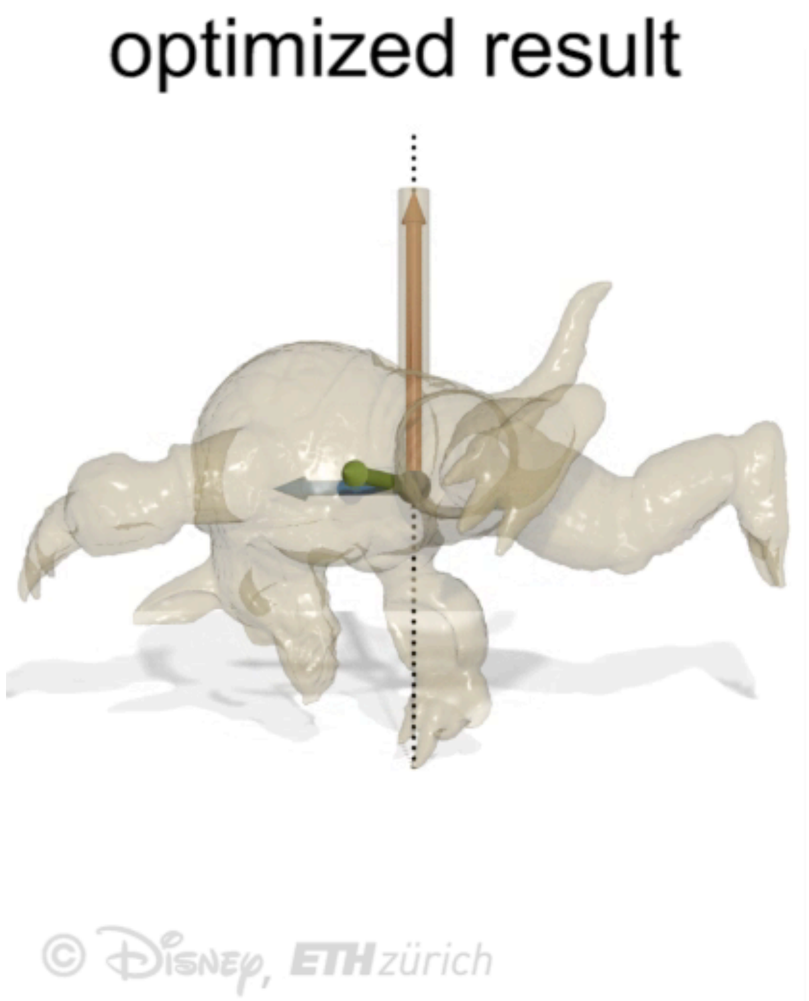


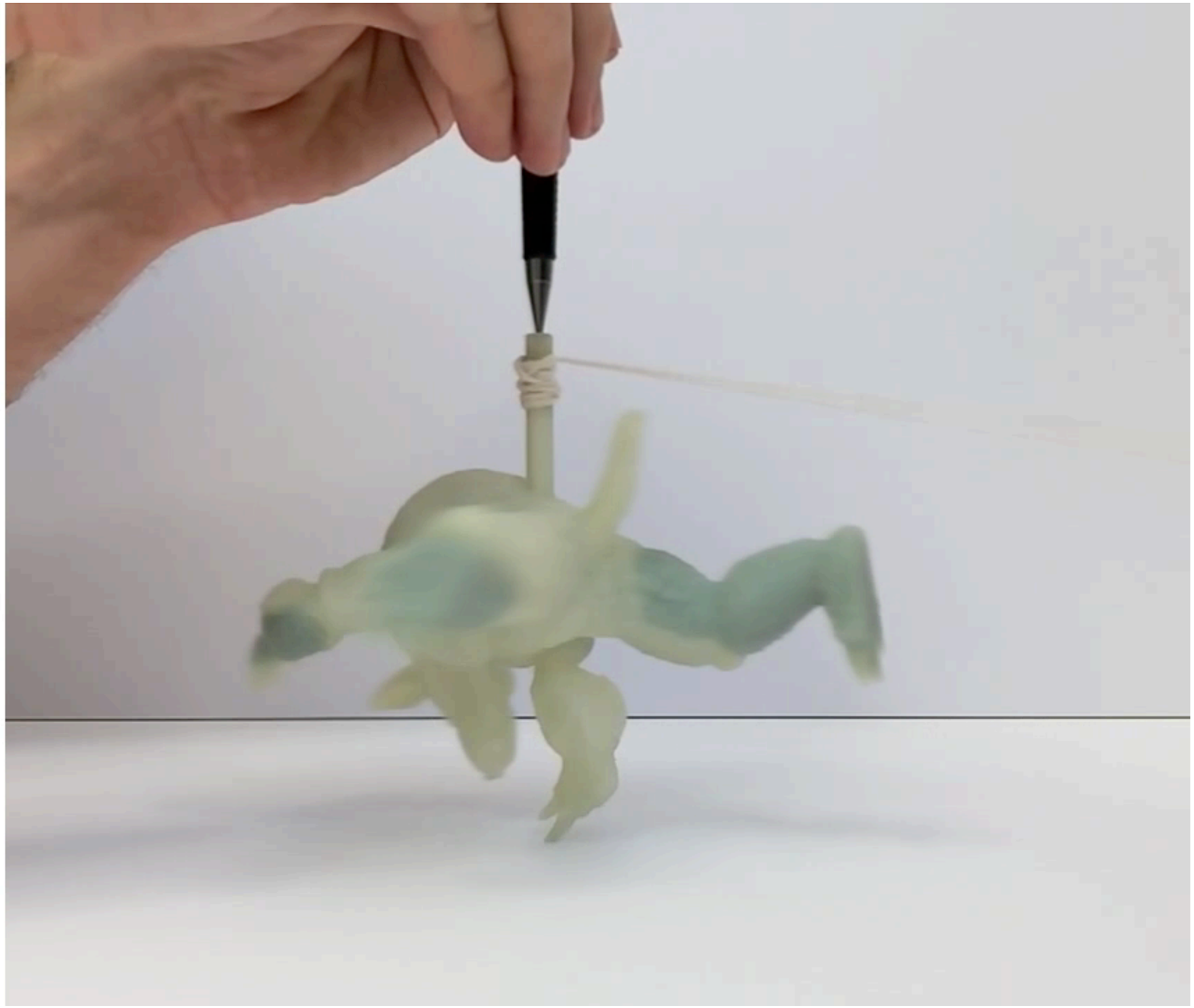
Sumit Jain, Yuting Ye, and C. Karen Liu, "Optimization-based Interactive Motion Synthesis"

Optimization in Graphics

Niloy J. Mitra, Leonidas Guibas, Mark Pauly, “Symmetrization”

Optimization in Graphics





Moritz Bächer, Emily Whiting, Bernd Bickel, Olga Sorkine-Hornung,
“Spin-It: Optimizing Moment of Inertia for Spinnable Objects”

Optimization in Graphics

Nobuyuki Umetani, Yuki Koyama, Ryan Schmidt & Takeo Igarashi,
“Pteromys: Interactive Design and Optimization of Free-formed Free-flight Model Airplanes”

Continuous vs. Discrete Optimization

- **DISCRETE:**
  - domain is a discrete set (e.g., finite or integers)
  - Example: best vegetable to put in a stew
    - Basic strategy? Try them all! (exponential)
    - sometimes clever strategy (e.g., MST)
    - more often, NP-hard (e.g., TSP)

- **CONTINUOUS:**
  - domain is not discrete (e.g., real numbers)
  - Example: best temperature to cook an egg
  - still many (NP-)hard problems, but also large classes of "easy" problems (e.g., convex)

Optimization Problem in Standard Form

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

"objective": how much does solution x cost?

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq b_i, \;\; i = 1, \ldots, m \end{aligned}$$

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

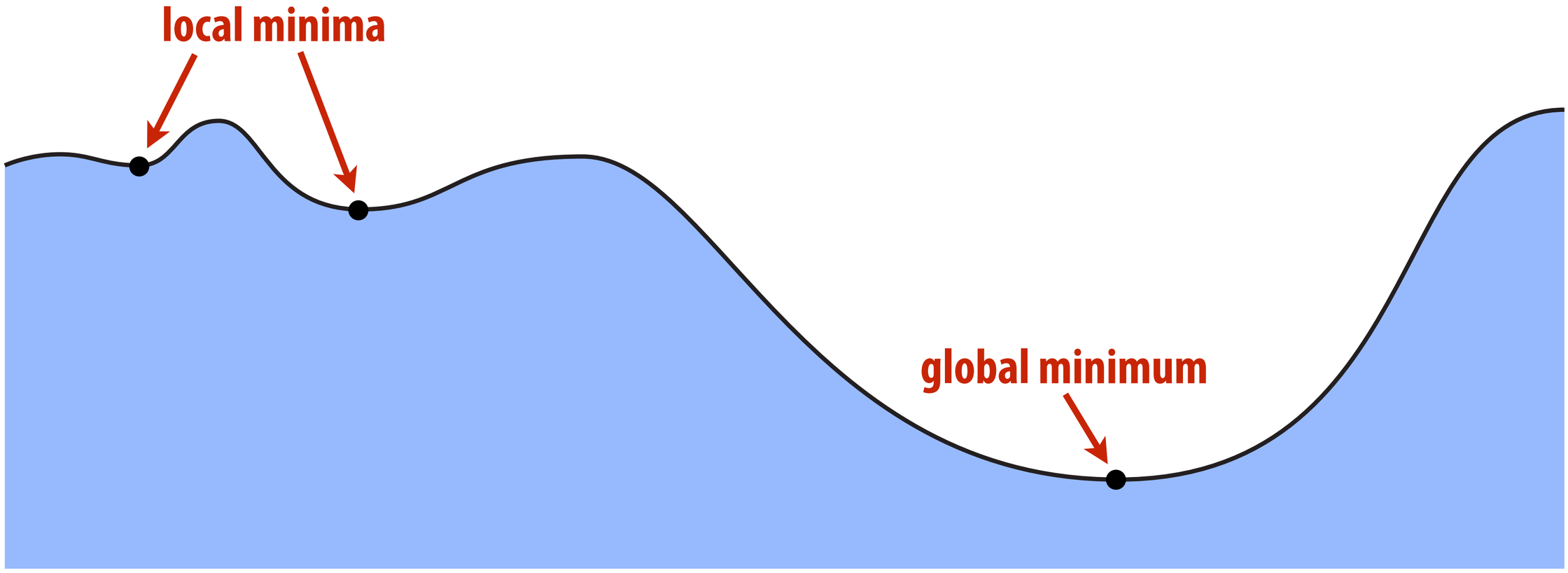
often (but not always) continuous, differentiable, ...

"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: Include two constraints: $g(x) \leq c$ and $g(x) \leq -c$**

Local vs. Global Minima

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



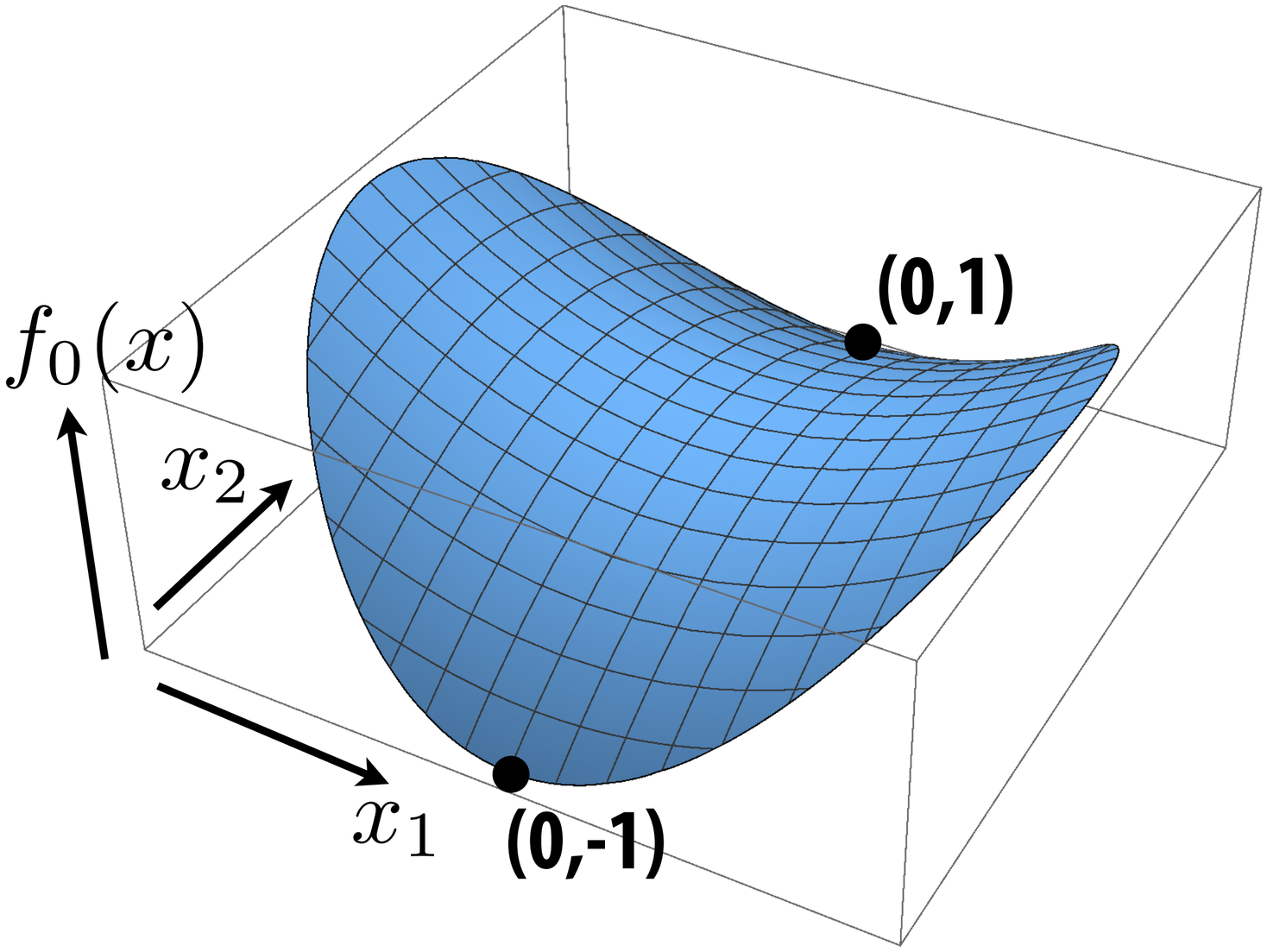
Philosophical question: does a local minimum "solve" the problem?

Depends on the problem! (E.g., real protein folding is local minimum)

Other times, local minima can be really bad (e.g., path planning)

Optimization Problem, Visualized

$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & x_1^2 - x_2^2 \\ \text{s.t.} \quad & x_1^2 + x_2^2 - 1 \leq 0 \end{aligned}$$

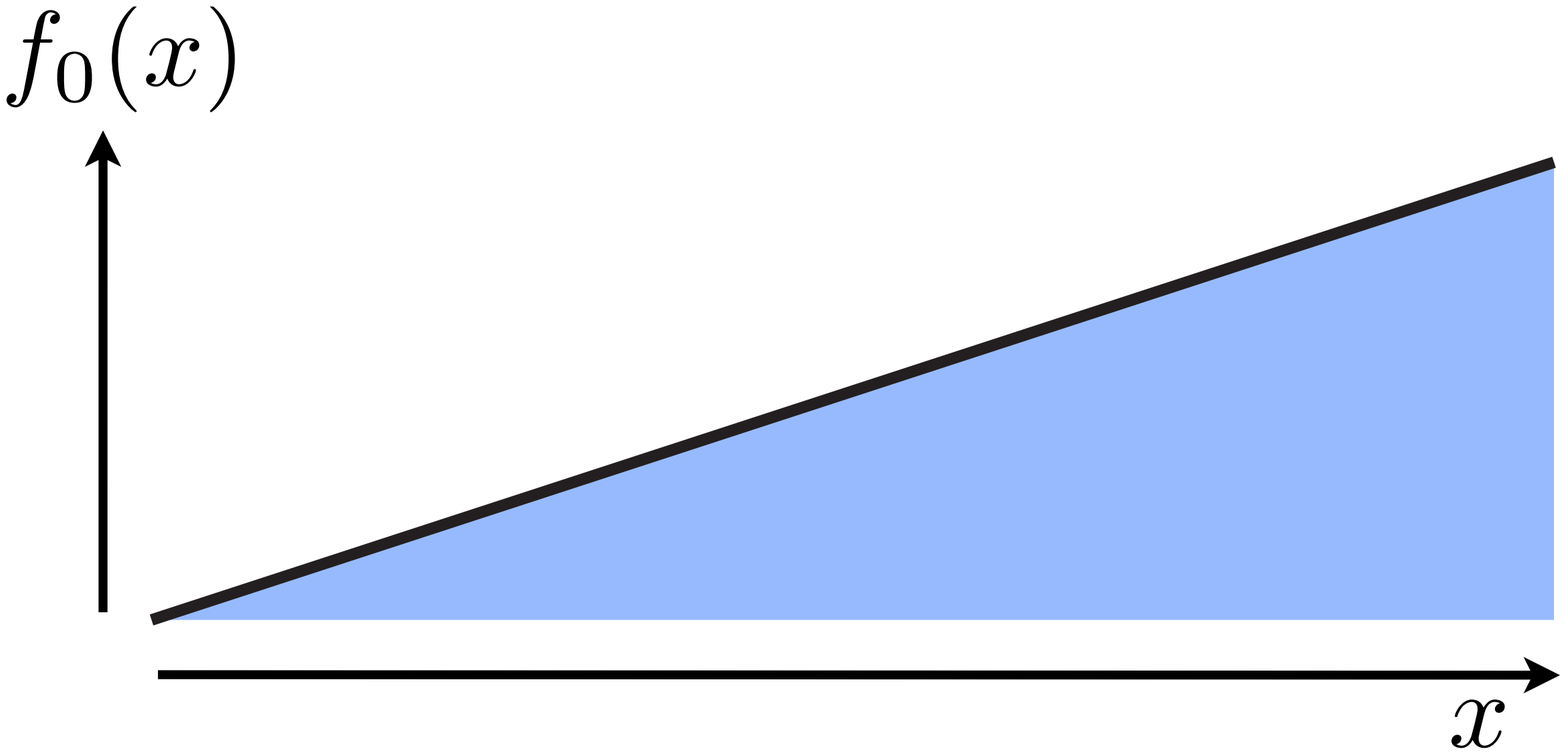


Q: Is this an optimization problem in standard form? **A: Yes.**

Q: Where is the optimal solution? **A: There are two, (0,1), (0,-1).**

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)

- WRONG! Not all objectives are bounded from below.
- It's like that old adage: "no matter how good you are, there will always be someone better than you."

Feasibility

- Ok, but suppose the objective is bounded from below.
- Then we can just take the best feasible solution, right?

value of objective doesn't depend on x;
all feasible solutions are equally good

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & 0 \\ \text{subject to} \quad & f_i(x) \leq b_i, \ \ i = 1, \ldots, m \end{aligned}$$

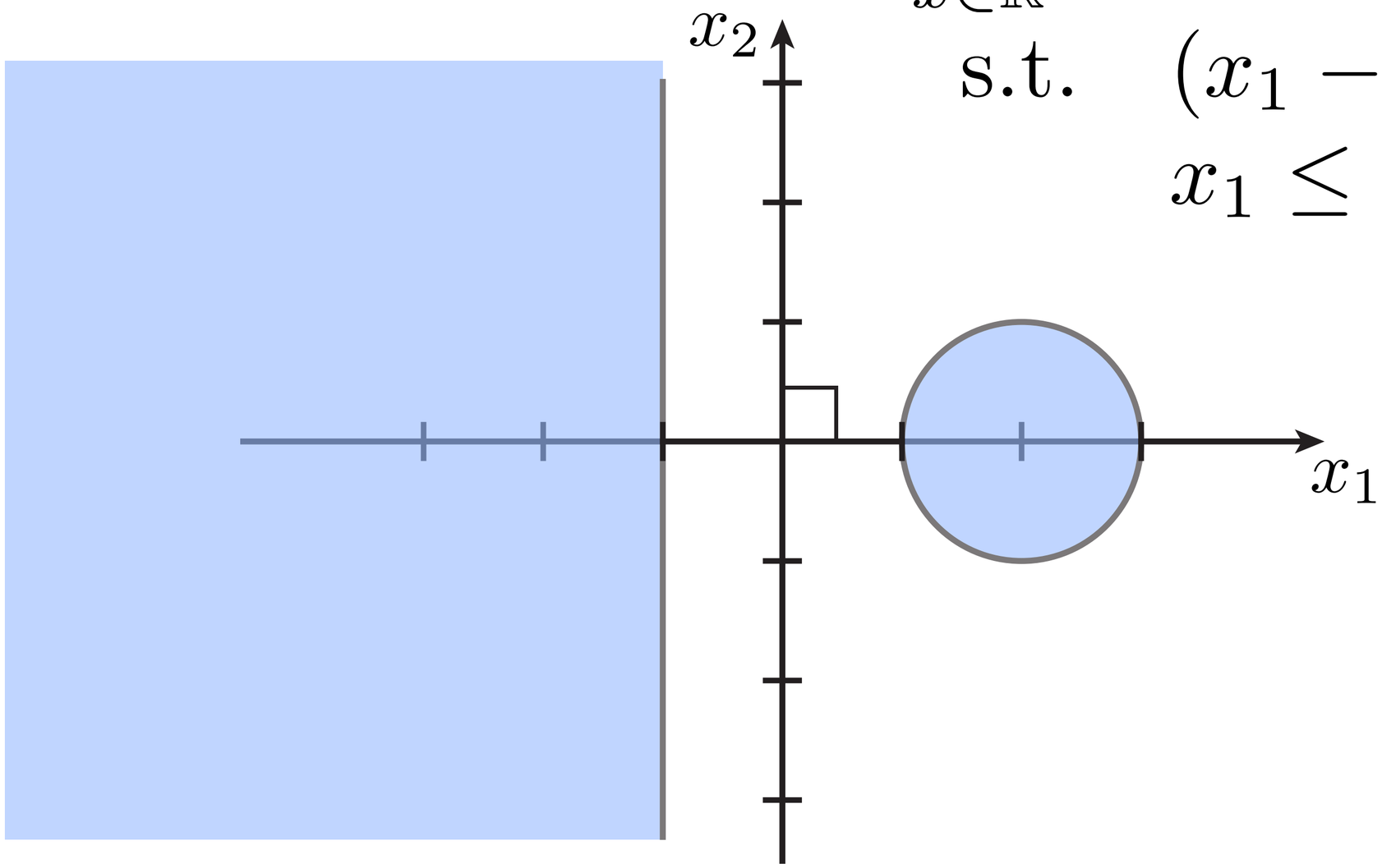
problem now is just finding a feasible solution—
which can be really hard (or impossible!)

- Not if there aren't any!
- Every system of equations is an optimization problem.
- But not all problems have solutions!
- (You'll appreciate this more as you get older.)

Feasibility - Example

Q: Is this problem feasible?

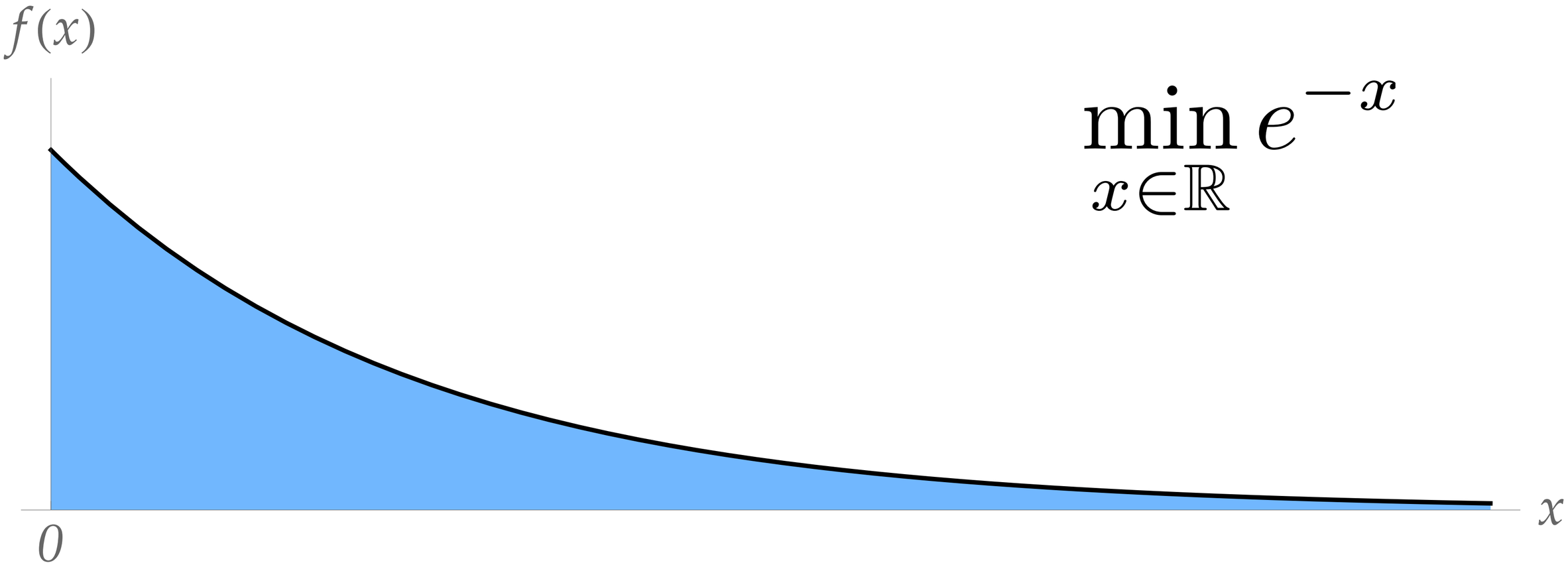
$$\begin{aligned} \min_{x \in \mathbb{R}^2} \quad & \sin(x_1) + x_2^2 \\ \text{s.t.} \quad & (x_1 - 2)^2 + x_2^2 \leq 1, \\ & x_1 \leq -1 \end{aligned}$$



A: No—the two sublevel sets (points where f_i(x) ≤ 0) have no common points, i.e., they do not overlap.

Existence & Uniqueness of Minimizers, cont.

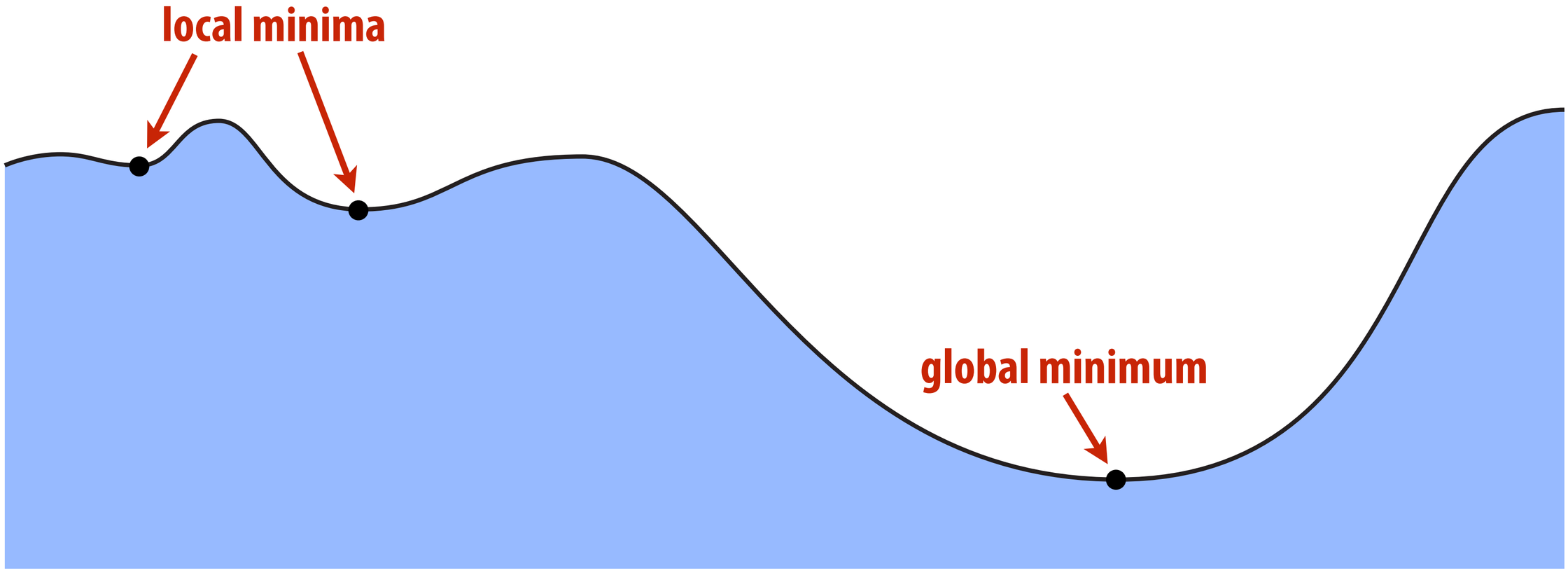
- Even being bounded from below is not enough:



- 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 $+\infty$ as we travel (far) in any direction

Characterization of Minimizers

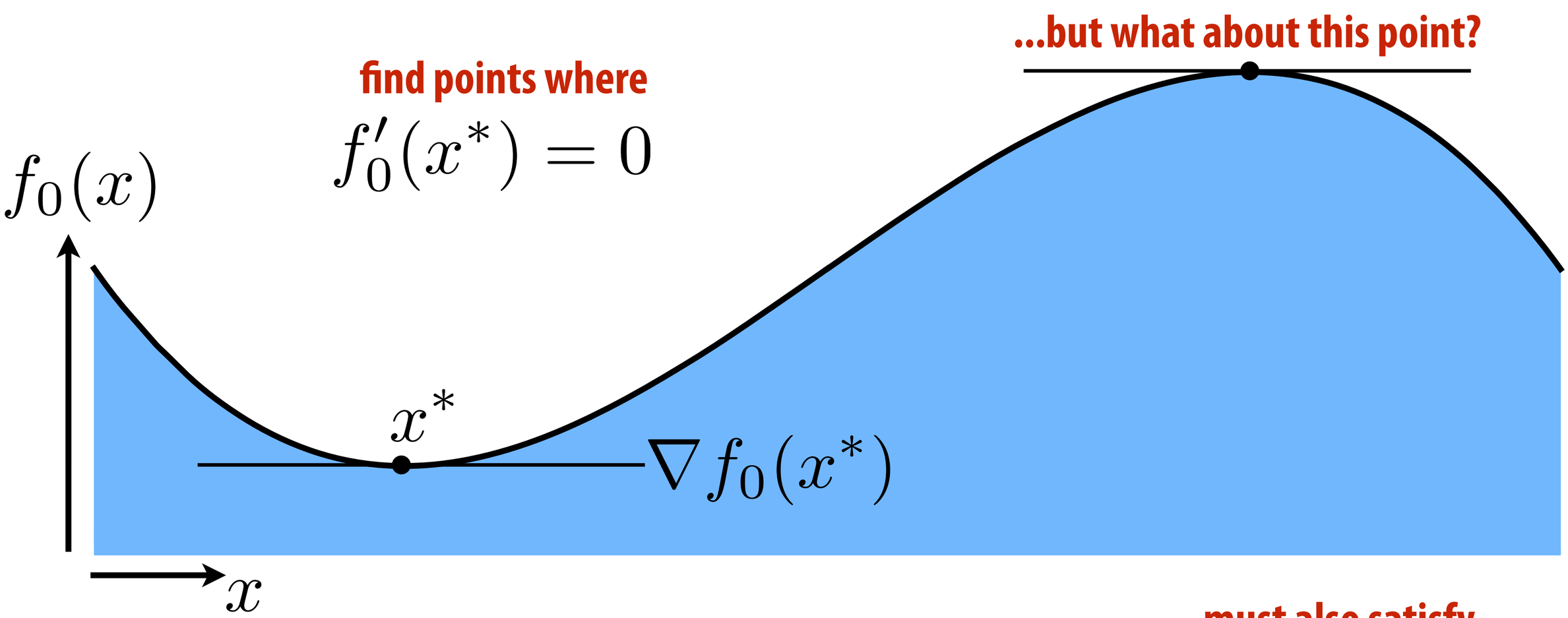
- Ok, so we have some sense of when a minimizer might exist
- But how do we know a given point x is a minimizer?



- 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!)



- 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?

must also satisfy
$f_0''(x^*) \geq 0$

Optimality Conditions (Unconstrained)

- In general, our objective is $f_0: \mathbb{R}^n \rightarrow \mathbb{R}$
- 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?

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

1st order

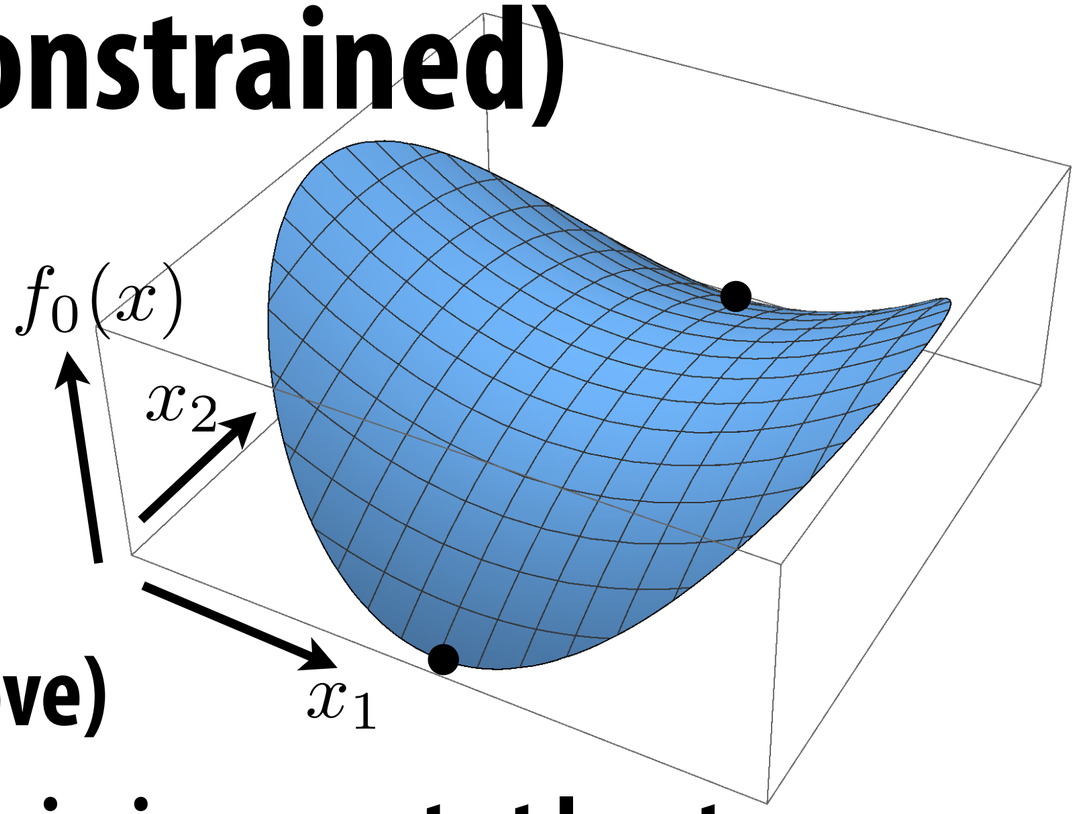
positive semidefinite (PSD)
($u^T A u \geq 0$ for all u)

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

2nd order

Optimality Conditions (Constrained)

- What if we have constraints?
- Is gradient at minimizer still zero?
- Is Hessian at minimizer still PSD?
- Not necessarily! (See example above)



- In general, any (local or global) minimizer must at least satisfy the Karush–Kuhn–Tucker (KKT) conditions:

$$\exists \lambda_i \text{ s.t.} \quad \nabla f_0(x^*) = -\sum_{i=1}^{n} \lambda_i \nabla f_i(x^*) \qquad \text{stationarity}$$

$$f_i(x^*) \leq 0, \;\; i = 1, \ldots, n \qquad \text{primal feasibility}$$

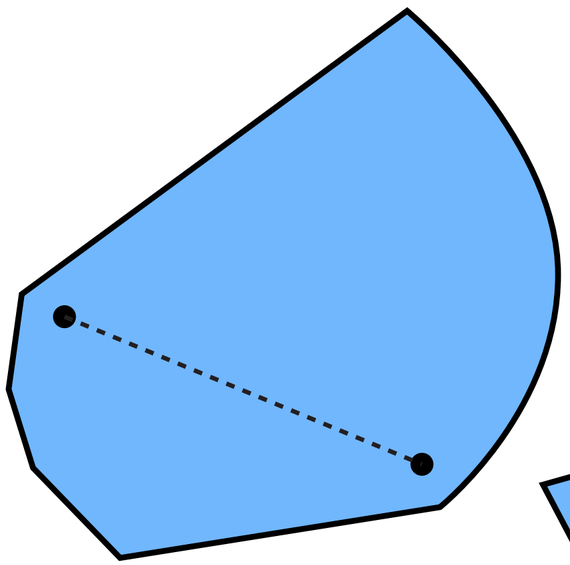
$$\lambda_i \geq 0, i = 1, \ldots, n \qquad \text{dual feasibility}$$

$$\lambda_i f_i(x^*) = 0, \;\; i = 1, \ldots, n \qquad \text{complementary slackness}$$

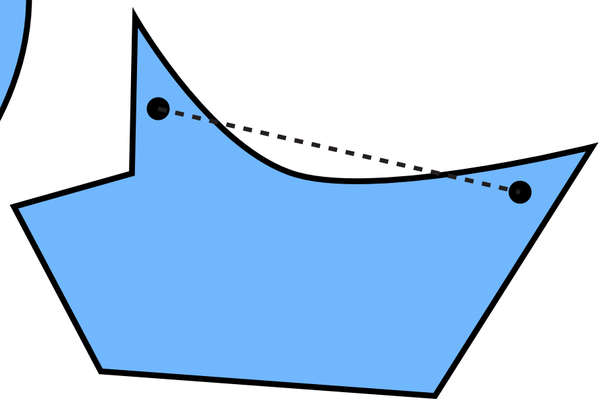
- ...we won't work with these in this class! (But good to know where to look.)

Convex Optimization

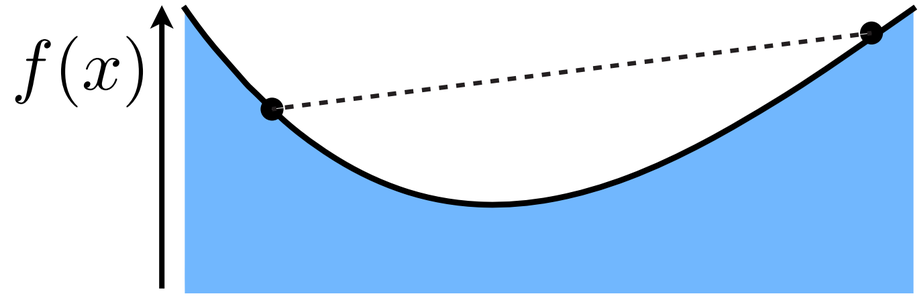
- Special class of problems that are almost always "easy" to solve (polynomial-time!)
- Problem convex if it has a convex domain and convex objective



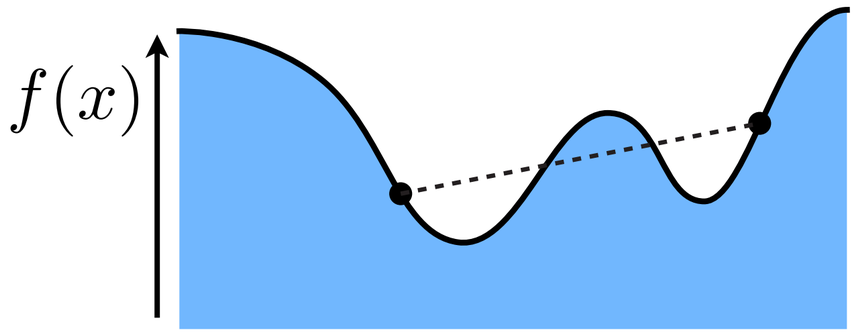
convex domain



noconvex domain



convex objective



nonconvex objective

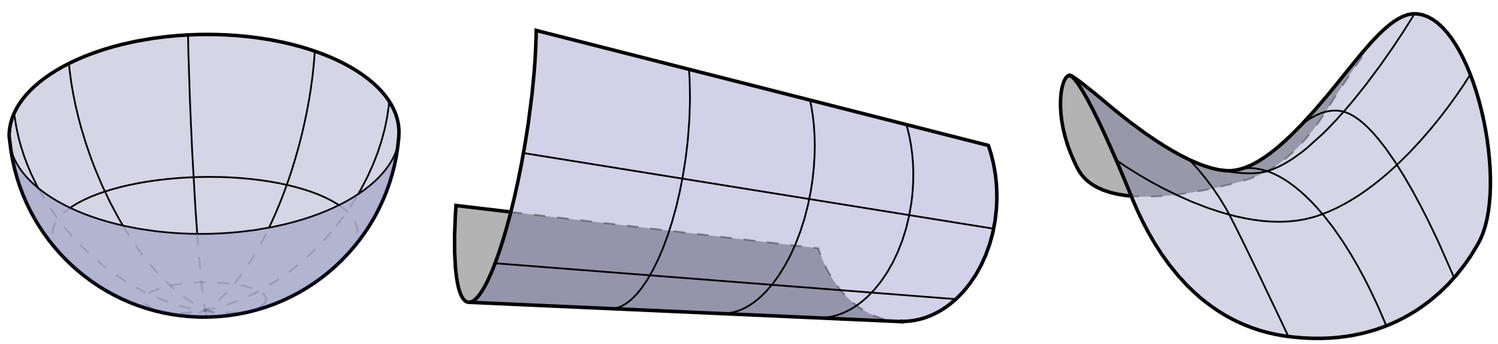
- Why care about convex problems in graphics?
  - can make guarantees about solution (always the best)
  - doesn't depend on initialization (strong convexity)
  - often quite efficient, but not always

Convex Quadratic Objectives & Linear Systems

- Very important example: convex quadratic objective
- Already saw this with, e.g., quadric error simplification
- Valuable "variational" way of looking at many common equations
- Can be expressed via positive-semidefinite (PSD) matrix:

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

- Q: 1st-order optimality condition? $Ax = b$ — just solve a linear system!
- Q: 2nd-order optimality condition? $A \succeq 0$ — satisfied by definition



positive definite | positive semidefinite | indefinite

Sadly, life is not usually that easy. How do we solve optimization problems in general?

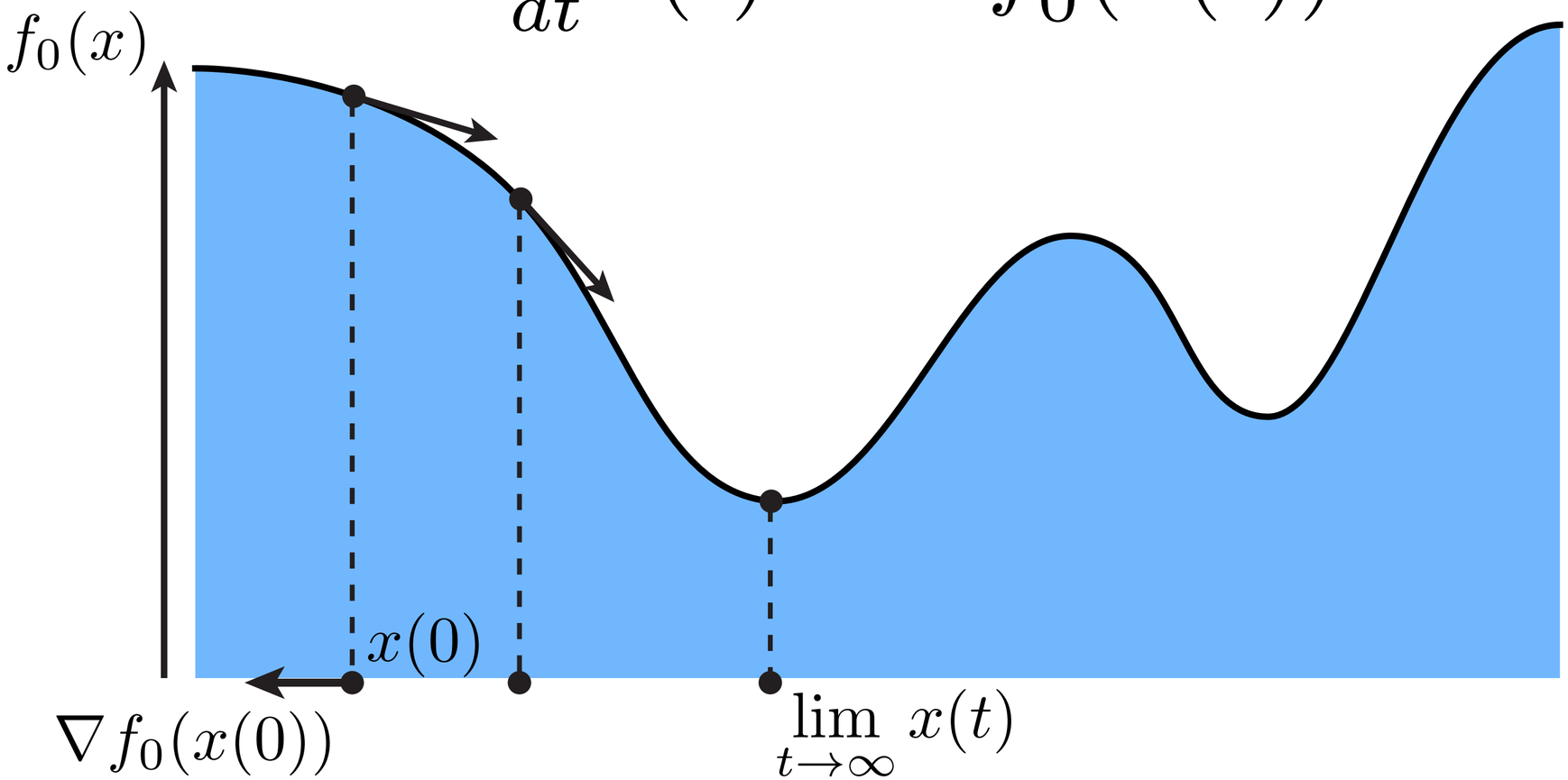
Descent Methods

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)

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



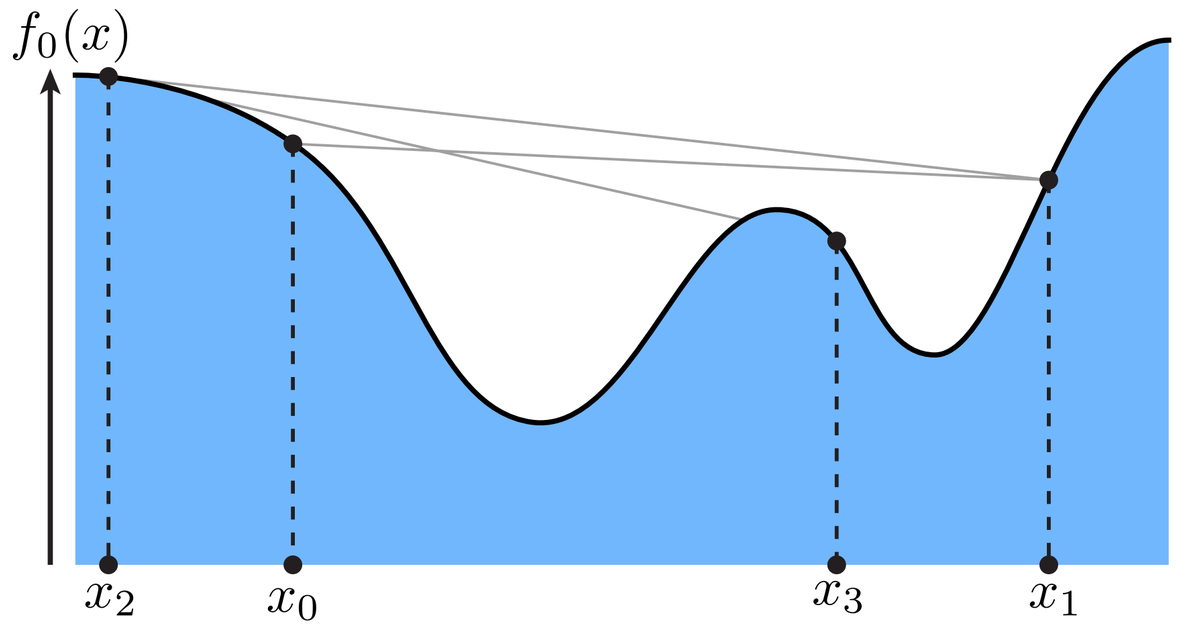
- Do we always end up at a (global) minimum?
- How do we compute gradient descent in practice?

Gradient Descent Algorithm (1D)

- Did you notice that gradient descent equation is an ODE?
- Q: How do we solve it numerically? $\frac{d}{dt}x(t) = -f_0'(x(t))$
- One way: forward Euler:

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

- Q: How do we pick the time step?
- If we're not careful, we'll go zipping all over the place; won't make any progress.



- 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.
- For now we will do something simpler: make τ really small!

Gradient Descent Algorithm (nD)

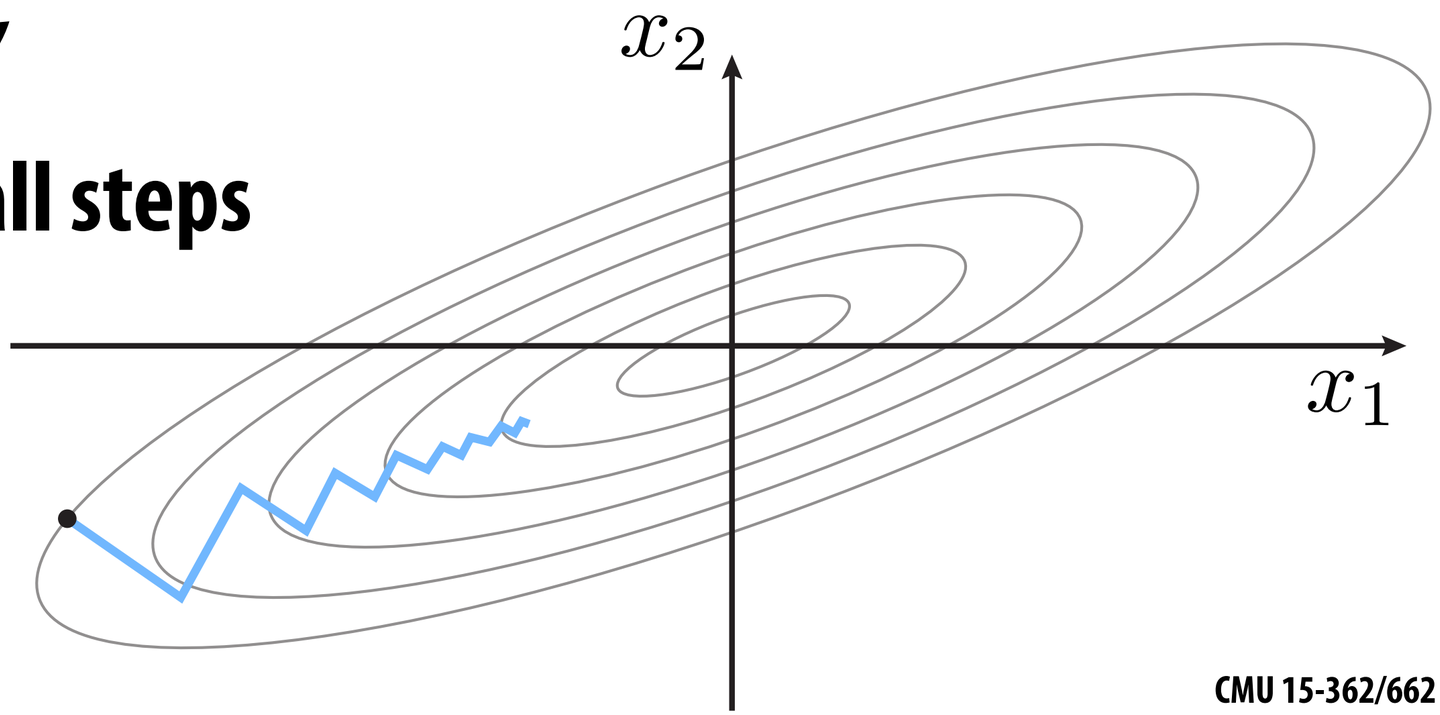
- **Q: How do we write gradient descent equation in general?**

$$\frac{d}{dt}x(t) = -\nabla f_0(x(t))$$

- **Q: What's the corresponding discrete update?**

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

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



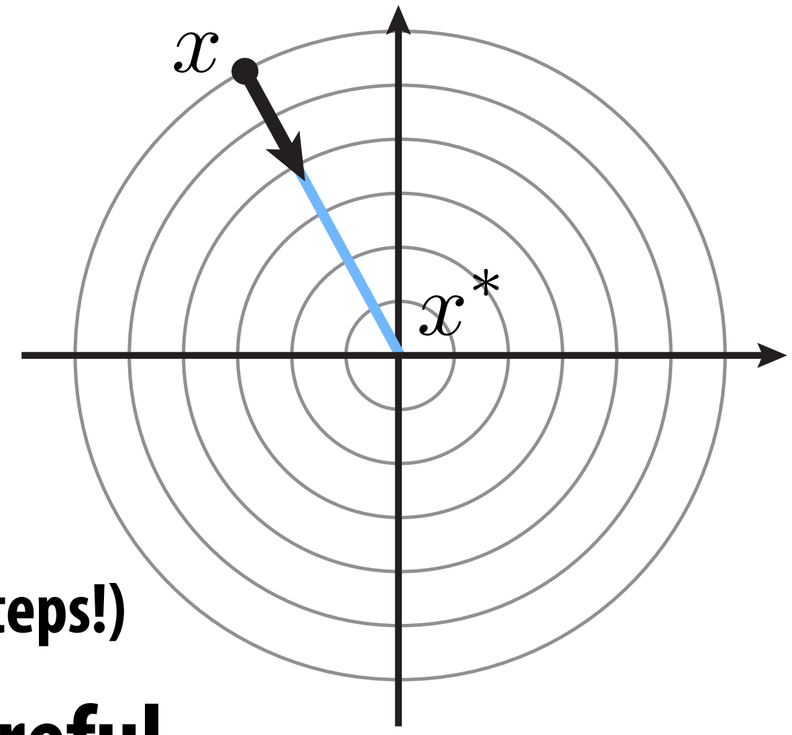
Higher Order Descent

- 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:

$$x_{k+1} = x_k - \tau(\nabla^2 f_0(x_k))^{-1} \nabla f_0(x_k)$$

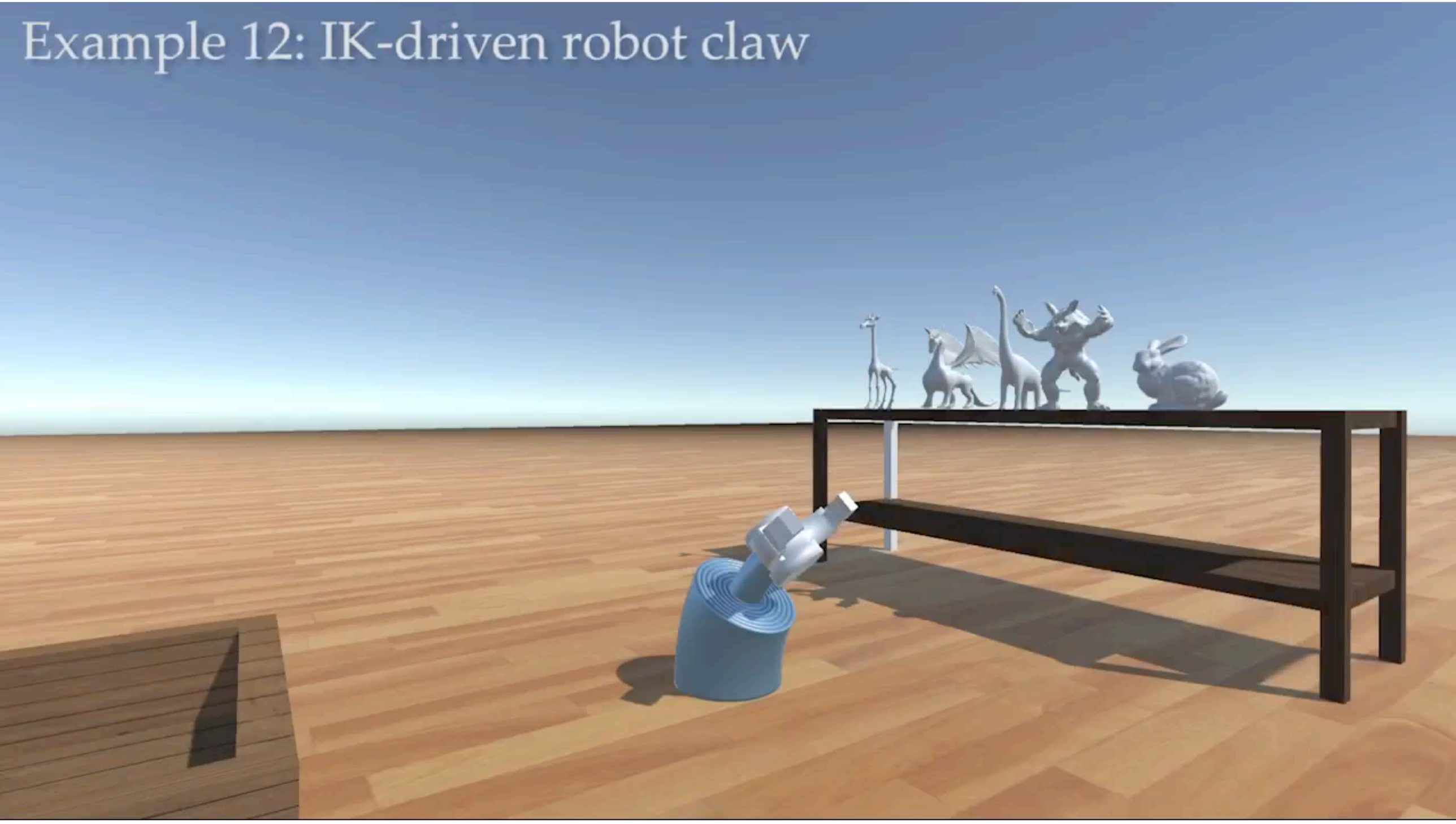
gradient

Hessian inverse



- 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
- Meta-strategy: try lots of solvers, see what works!
  - quasi-Newton, trust region, L-BFGS, ...

Example: Inverse Kinematics



Another example of tools we have been learning

Real-Time Gradient-Domain Painting

James McCann Nancy Pollard
Carnegie Mellon University

(With Audio)

Painting with gradient tool brushes

If we express an image as a vector field of gradients, how do we solve for the image???

6 Integration

Finding the image u with gradients close, in the least-squares sense, to given – potentially non-conservative – edited gradient images G^x, G^y is equivalent to solving a Poisson equation:

$$\nabla^2 u = f \tag{5}$$

Where f is computed from the edited gradients:

$$f_{x,y} = G^x_{x,y} - G^x_{x-1,y} + G^y_{x,y} - G^y_{x,y-1} \tag{6}$$

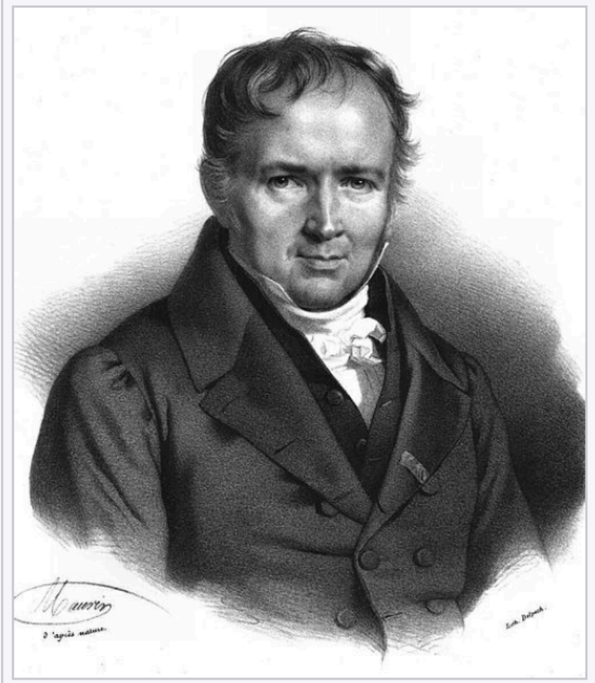
Poisson's equation

31 languages

Article Talk | Read Edit View history Tools

From Wikipedia, the free encyclopedia

Poisson's equation is an elliptic partial differential equation of broad utility in theoretical physics. For example, the solution to Poisson's equation is the potential field caused by a given electric charge or mass density distribution; with the potential field known, one can then calculate the corresponding electrostatic or gravitational (force) field. It is a generalization of Laplace's equation, which is also frequently seen in physics. The equation is named after French mathematician and physicist Siméon Denis Poisson who published it in 1823.[1][2]



Siméon Denis Poisson

Statement of the equation [edit]

Poisson's equation is

$$\Delta\varphi = f,$$

where Δ is the Laplace operator, and f and φ are real or complex-valued functions on a manifold. Usually, f is given, and φ is sought. When the manifold is Euclidean space, the Laplace operator is often denoted as ∇^2, and so Poisson's equation is frequently written as

$$\nabla^2\varphi = f.$$

In three-dimensional Cartesian coordinates, it takes the form

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)\varphi(x,y,z) = f(x,y,z).$$

When $f = 0$ identically, we obtain Laplace's equation.

Model Equations

- **Fundamental behavior of many important PDEs is well-captured by three model linear equations:**

"Laplacian" (more later!)

LAPLACE EQUATION ("ELLIPTIC") $\Delta u = 0$

"what's the smoothest function interpolating the given boundary data"

HEAT EQUATION ("PARABOLIC") $\dot{u} = \Delta u$

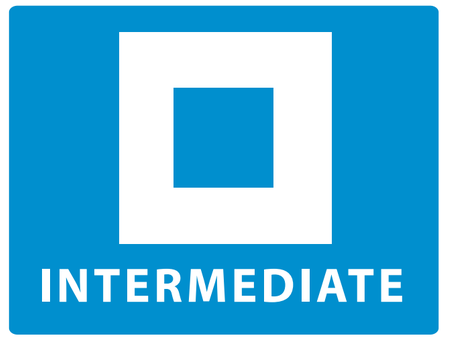
"how does an initial distribution of heat spread out over time?"

WAVE EQUATION ("HYPERBOLIC") $\ddot{u} = \Delta u$

"if you throw a rock into a pond, how does the wavefront evolve over time?"

[NONLINEAR + HYPERBOLIC + HIGH-ORDER]

Solve numerically?



6 Integration

Finding the image u with gradients close, in the least-squares sense, to given – potentially non-conservative – edited gradient images G^x, G^y is equivalent to solving a Poisson equation:

$$\nabla^2 u = f \tag{5}$$

Where f is computed from the edited gradients:

$$f_{x,y} = G^x_{x,y} - G^x_{x-1,y} + G^y_{x,y} - G^y_{x,y-1} \tag{6}$$

The Laplace Operator

- All of our model equations used the Laplace operator
- Different conventions for symbol:

Δ ← same symbol used for "change"

∇^2 ← same symbol used for Hessian!

- Unbelievably important object showing up everywhere across physics, geometry, signal processing, …
- Ok, but what does it mean?
- Differential operator: eats a function, spits out its "2nd derivative"
- What does that mean for a function $u : \mathbb{R}^n \to \mathbb{R}$?

  –divergence of gradient

  –sum of second derivatives

  –deviation from local average

  –…

div grad

$$\Delta u = \nabla \cdot \nabla u$$

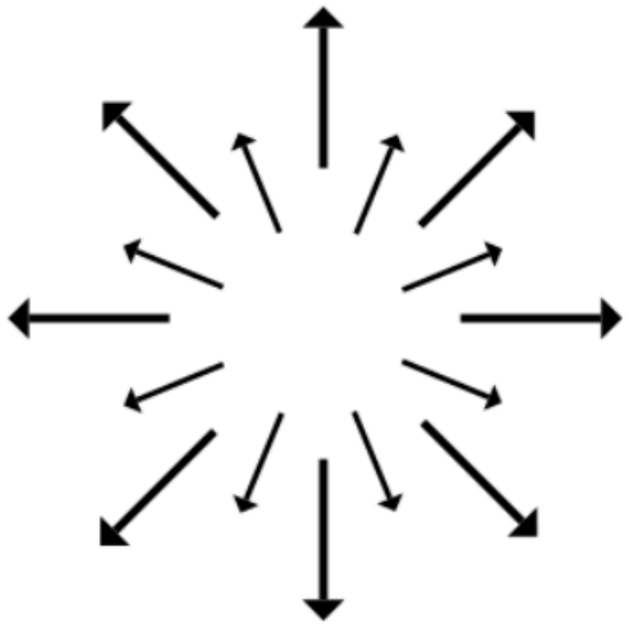
$$\Delta u = \frac{\partial u^2}{\partial x_1^2} + \cdots + \frac{\partial u^2}{\partial x_n^2}$$

For more intuition about the Laplacian: https://youtu.be/oEq9R0l9Umk

Let's use the definition of Laplacian as divergence of the gradient

$$\Delta u = \nabla \cdot \nabla u$$

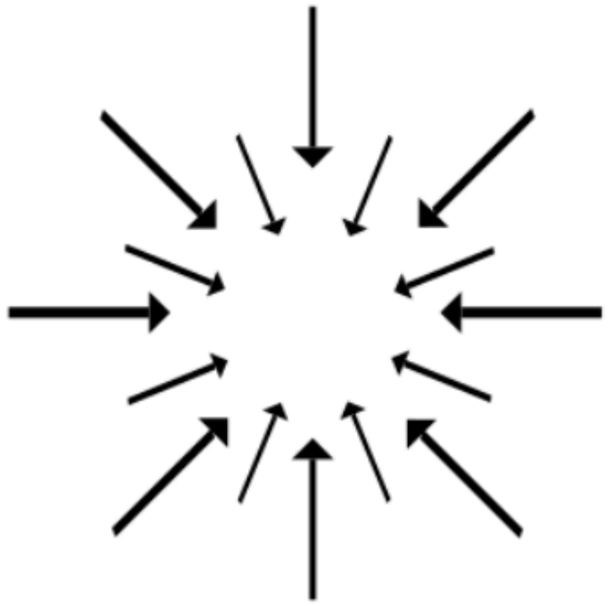
div grad



$\partial/\partial x(\mathbf{V}_x) > 0$

$\partial/\partial y(\mathbf{V}_y) > 0$

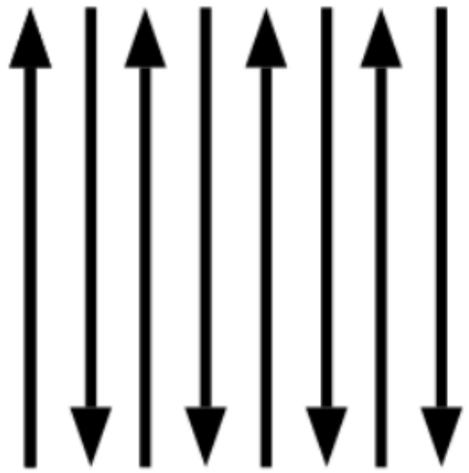
$\nabla \bullet (\mathbf{V}) > 0$



$\partial/\partial x(\mathbf{V}_x) < 0$

$\partial/\partial y(\mathbf{V}_y) < 0$

$\nabla \bullet (\mathbf{V}) < 0$



$\partial/\partial x(\mathbf{V}_x) = 0$

$\partial/\partial y(\mathbf{V}_y) = 0$

$\nabla \bullet (\mathbf{V}) = 0$

The divergence of different vector fields. The divergence of vectors from point (*x*,*y*) equals the sum of the partial derivative-with-respect-to-*x* of the *x*-component and the partial derivative-with-respect-to-*y* of the *y*-component at that point: $\nabla \cdot (\mathbf{V}(x,y)) = \frac{\partial V_x(x,y)}{\partial x} + \frac{\partial V_y(x,y)}{\partial y}$

6 Integration

Finding the image u with gradients close, in the least-squares sense, to given – potentially non-conservative – edited gradient images G^x, G^y is equivalent to solving a Poisson equation:

$$\nabla^2 u = f \tag{5}$$

Where f is computed from the edited gradients:

$$f_{x,y} = G^x_{x,y} - G^x_{x-1,y} + G^y_{x,y} - G^y_{x,y-1} \tag{6}$$

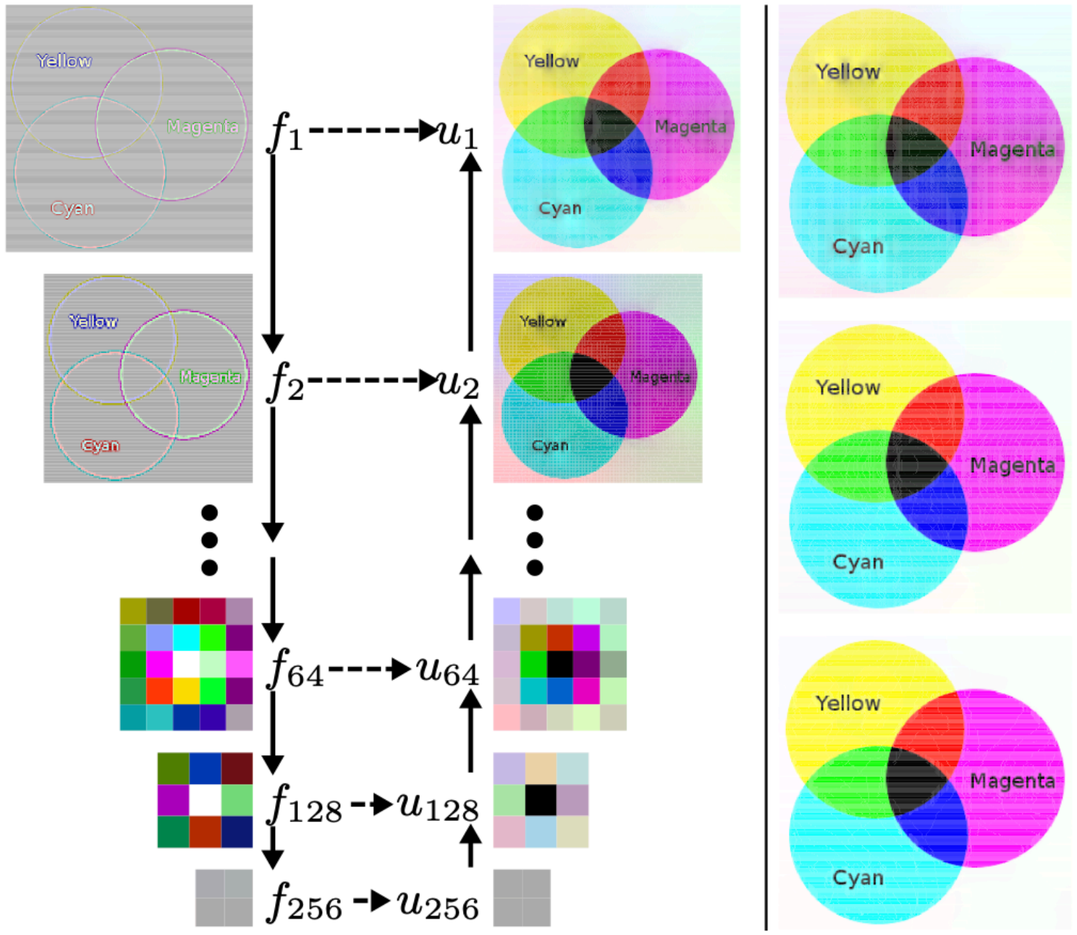


Figure 8: Left*: illustration of a single call to* `VCycle`*, with* f_1 *set to the gradient field of a 257x257 test image. Arrows show data flow.* Right*: The approximation is not perfect, but improves with subsequent iterations.*

`VCycle`(f_h):
  if (size(f_h) == 1x1) **return** [0]
  $f_{2h} \leftarrow \mathcal{R} f_h$;Restrict f to coarser grid
  $u_{2h} \leftarrow$ `VCycle`(f_{2h}) ;Approximate coarse solution
  $u_h \leftarrow \mathcal{P} u_{2h}$;Interpolate coarse solution
  $u_h \leftarrow$ `Relax`(u_h, f_h, x_{h0}) ;Refine solution
  $u_h \leftarrow$ `Relax`(u_h, f_h, x_{h1}) ;Refine solution (again)
  return u_h

`Relax`(u_h, f_h, x):
  return $\frac{1}{m_h - x} (f_h - (\mathcal{L}_h - (m_h - x)I)u_h)$;Jacobi

https://graphics.cs.cmu.edu/projects/gradient-paint/