

$$\nabla f + \sum_k \lambda_k \nabla g_k$$

Optimization Techniques for Geometry Processing

Justin Solomon
Princeton University

David Bommes
RWTH Aachen University

This Morning's Focus

Optimization.

Synonym(-ish): *Variational* methods.

This Morning's Focus

Optimization.

Synonym(-ish): *Variational* methods.



Caveat: Slightly different connotation in ML

More Specifically

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \geq 0 \end{aligned}$$

Two Roles

■ Client

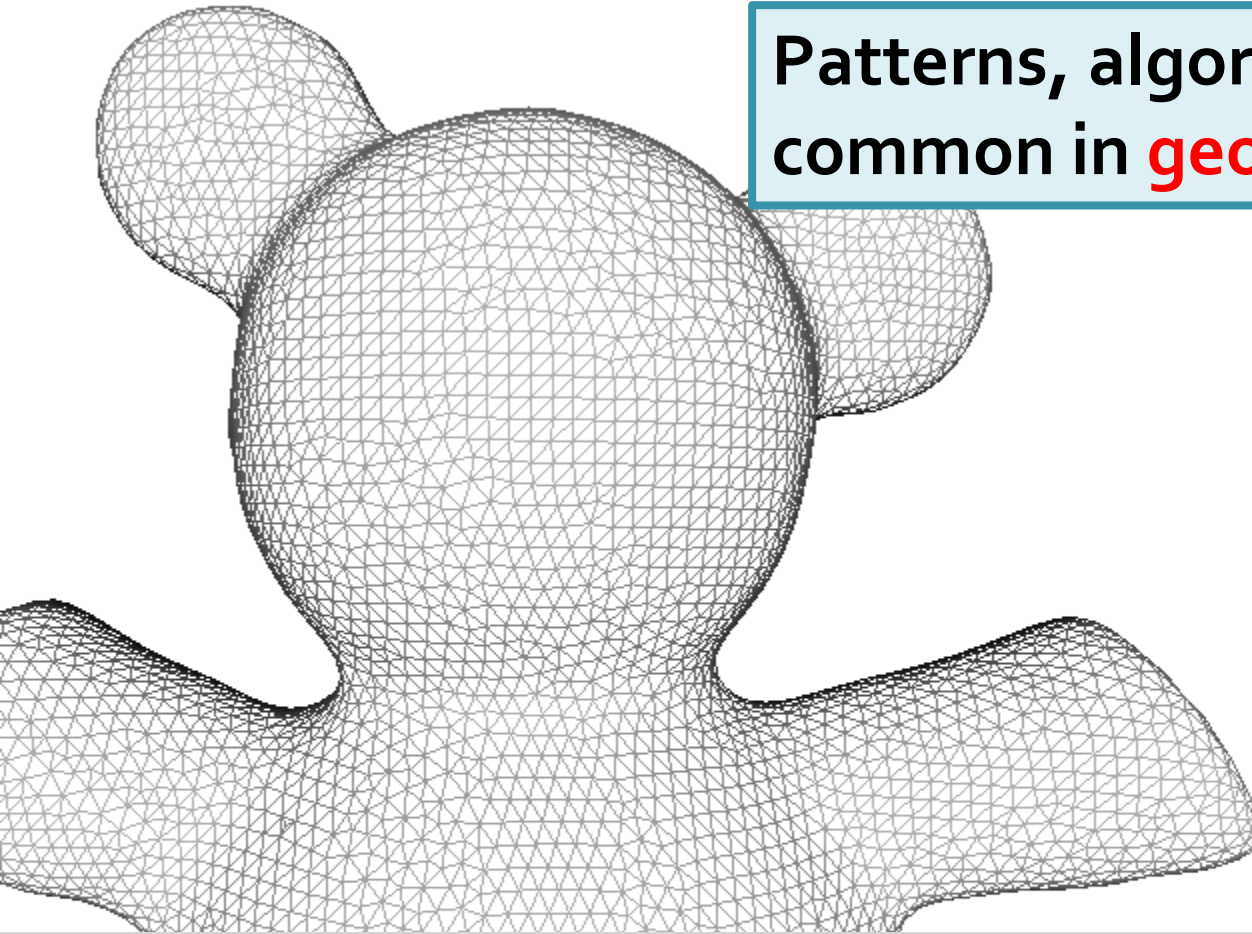
Which optimization tool is relevant?

■ Designer

Can I design an algorithm for this problem?

Our Bias

Patterns, algorithms, & examples
common in **geometry processing**.



Optimization is a huge field.

Rough Plan

Part I (Justin)

- Vocabulary
- Simple examples
- Unconstrained optimization
- Equality-constrained optimization

Rough Plan

Part II (David)

- Inequality constraints
- Advanced algorithms
- Discrete problems
- Conclusion

Rough Plan

Part I (Justin)

- **Vocabulary** *(basic material!)*
- Simple examples
- Unconstrained optimization
- Equality-constrained optimization

Optimization Terminology

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \geq 0 \end{aligned}$$

Objective (“Energy Function”)

Optimization Terminology

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \geq 0 \end{aligned}$$

Equality Constraints

Optimization Terminology

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

$$\text{s.t. } g(x) = 0$$

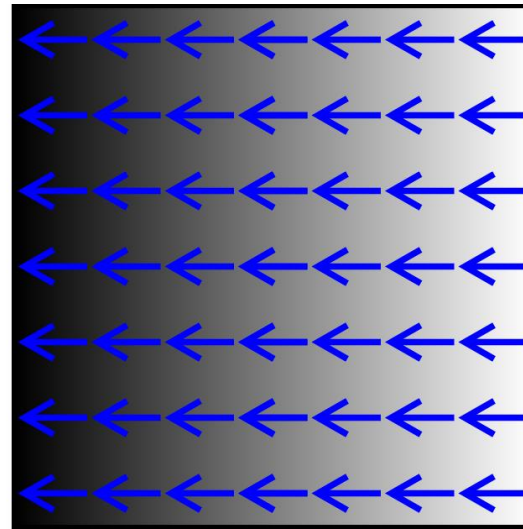
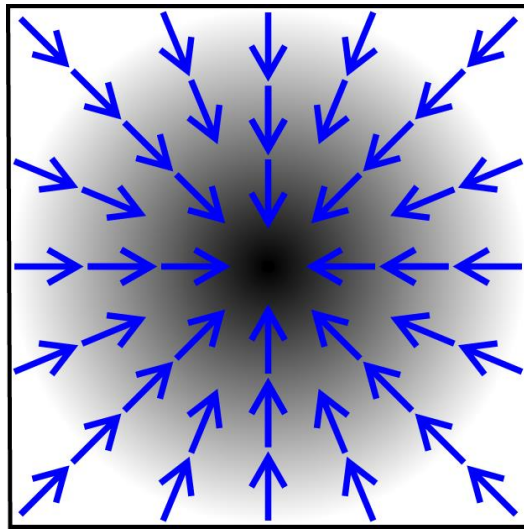
$$h(x) \geq 0$$

Inequality Constraints

Optimization Terminology

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

$$\rightarrow \nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n} \right)$$

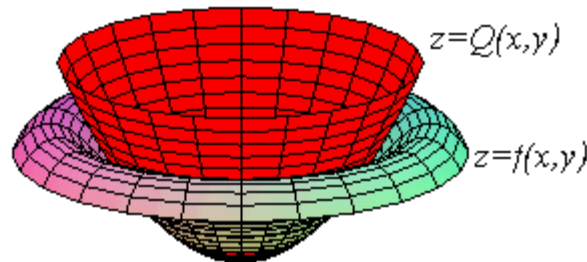


<https://en.wikipedia.org/?title=Gradient>

Gradient

Optimization Terminology

$$f : \mathbb{R}^n \rightarrow \mathbb{R} \rightarrow H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$



$$f(x) \approx f(x_0) + \nabla f(x_0)^\top (x - x_0) + (x - x_0)^\top H f(x_0) (x - x_0)$$

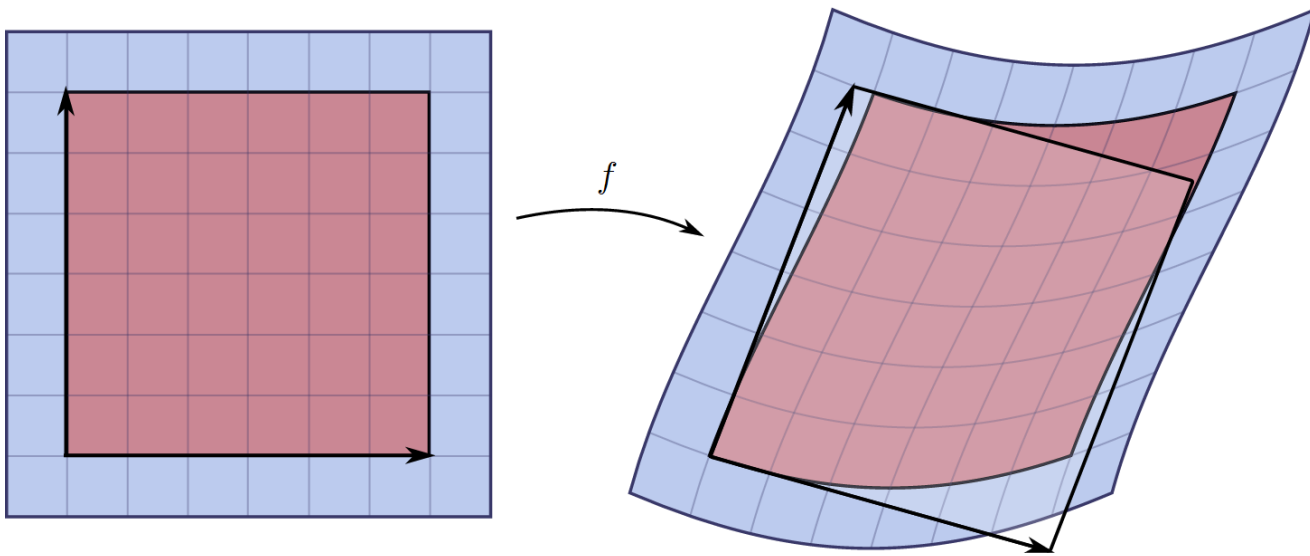
<http://math.etsu.edu/multicalc/prealpha/Chap2/Chap2-5/10-3a-t3.gif>

Hessian

Optimization Terminology

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^m$$

$$\rightarrow (Df)_{ij} = \frac{\partial f_i}{\partial x_j}$$



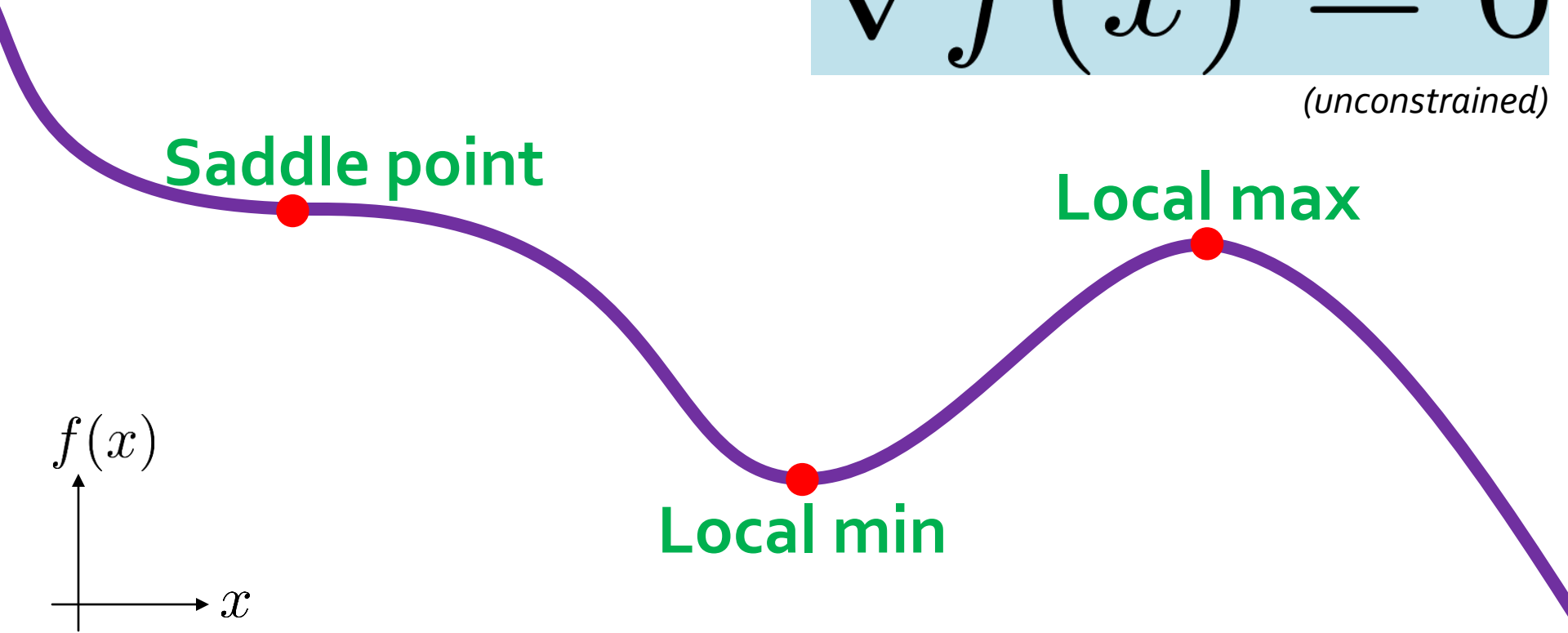
https://en.wikipedia.org/wiki/Jacobian_matrix_and_determinant

Jacobian

Optimization Terminology

$$\nabla f(x) = 0$$

(unconstrained)

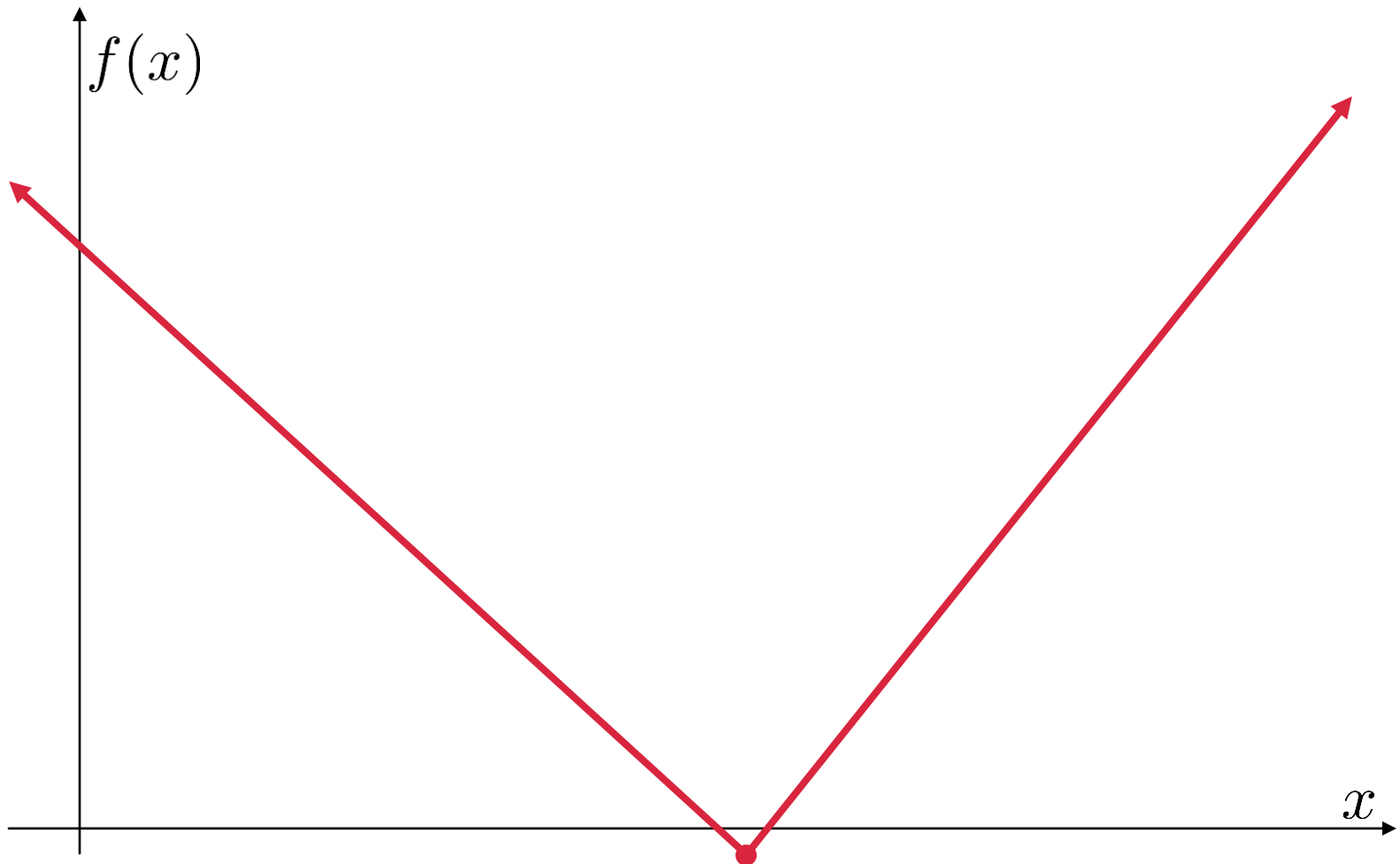


Critical point

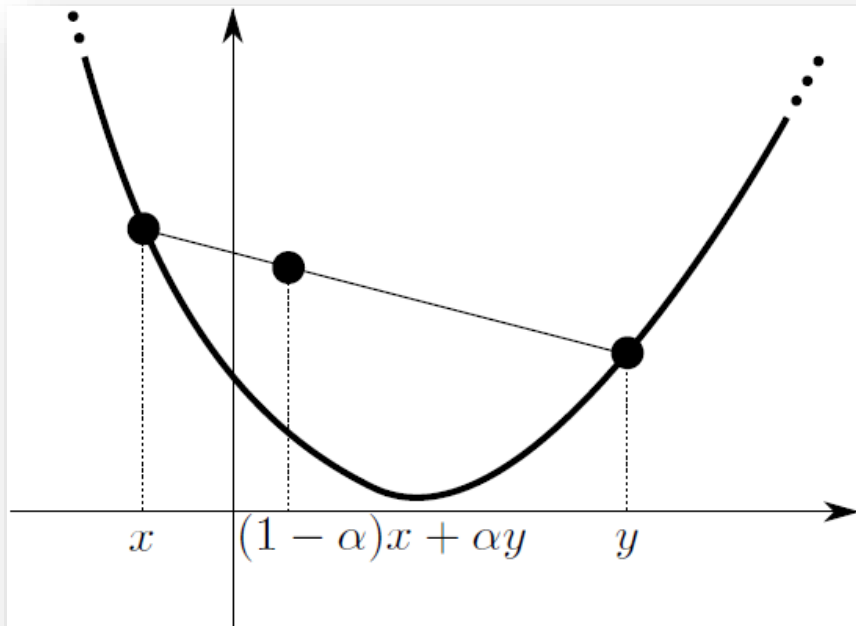
Common Mistake

Critical points
may not be minima.

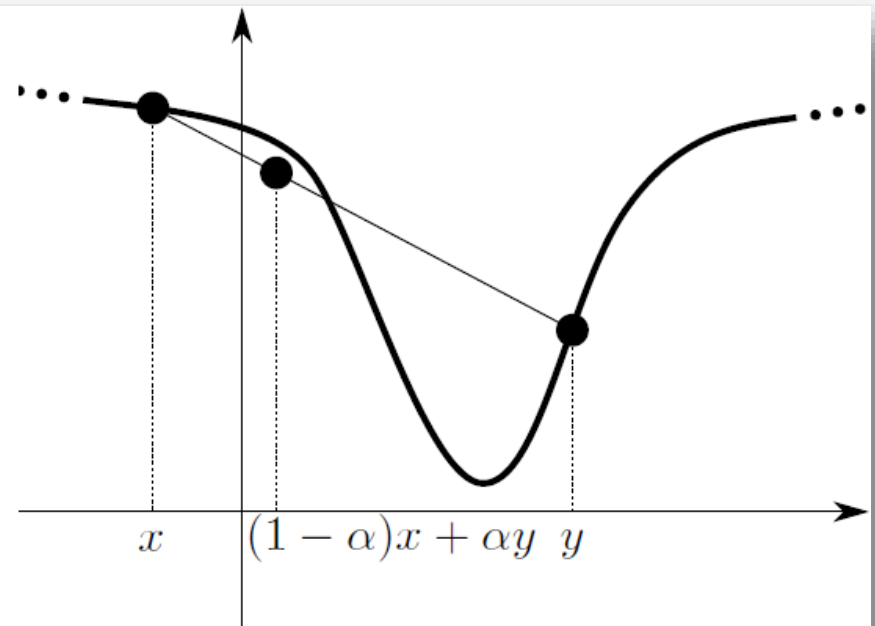
Neither Sufficient *nor* Necessary



Except...



(a) Convex



(b) Quasiconvex

More later

Rough Plan

Part I (Justin)

- Vocabulary
- **Simple examples**
- Unconstrained optimization
- Equality-constrained optimization

Encapsulates Many Problems

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(x) \\ \text{s.t.} \quad & g(x) = 0 \\ & h(x) \geq 0 \end{aligned}$$

$$Ax = b \Leftrightarrow f(x) = \|Ax - b\|_2$$

$$Ax = \lambda x \Leftrightarrow f(x) = \|Ax\|_2, g(x) = \|x\|_2 - 1$$

$$\text{Roots of } g(x) \Leftrightarrow f(x) = 0$$



How effective are
generic
optimization tools?

Generic Advice

Try the
simplest solver first.

Quadratic with Linear Equality

$$\begin{array}{ll}\min_x & \frac{1}{2}x^\top Ax - b^\top x + c \\ \text{s.t.} & Mx = v\end{array}$$

(assume A is symmetric and positive definite)



$$\begin{pmatrix} A & M^\top \\ M & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} b \\ v \end{pmatrix}$$

Special Case: Least-Squares

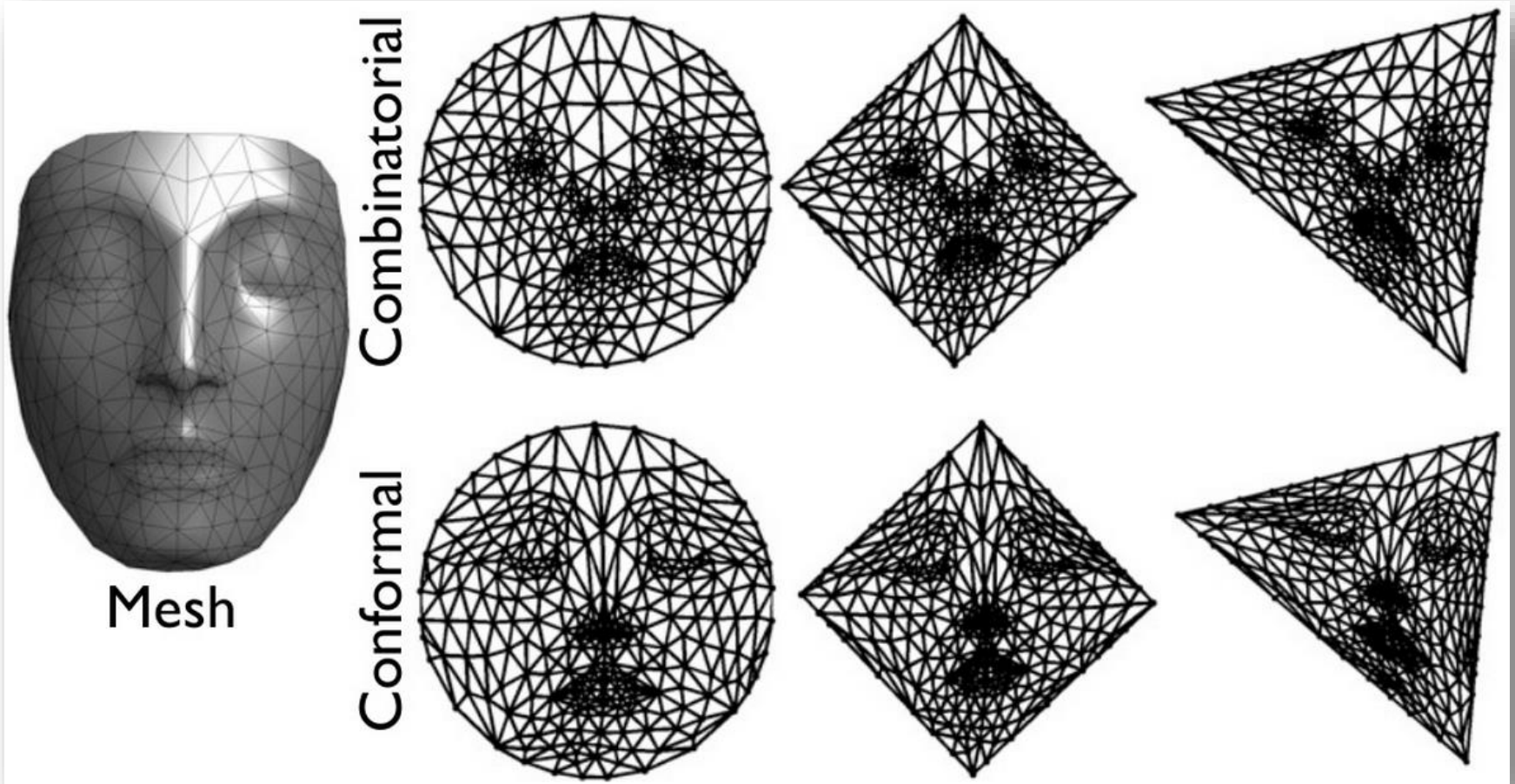
$$\min_x \frac{1}{2} \|Ax - b\|_2^2$$

$$\rightarrow \min_x \frac{1}{2} x^\top A^\top Ax - b^\top Ax + \|b\|_2^2$$

$$\implies A^\top Ax = A^\top b$$

Normal equations
(better solvers for this case!)

Example: Mesh Embedding



Linear Solve for Embedding

$$\begin{aligned} \min_{x_1, \dots, x_{|V|}} \quad & \sum_{(i,j) \in E} w_{ij} \|x_i - x_j\|_2^2 \\ \text{s.t.} \quad & x_v \text{ fixed } \forall v \in V_0 \end{aligned}$$

- **$w_{ij} \equiv 1$** : Tutte embedding
- **w_{ij} from mesh**: Harmonic embedding

Assumption: w symmetric.

More Explicit Form

$$\sum_{(i,j) \in E} w_{ij} (x_i - x_j)^2 = \sum_{(i,j) \in E} w_{ij} (x_i^2 - 2x_i x_j + x_j^2)$$

$$= \sum_{i \in V} \left[2x_i^2 \sum_{j \sim i} w_{ij} - \sum_{j \sim i} 2w_{ij} x_i x_j \right]$$

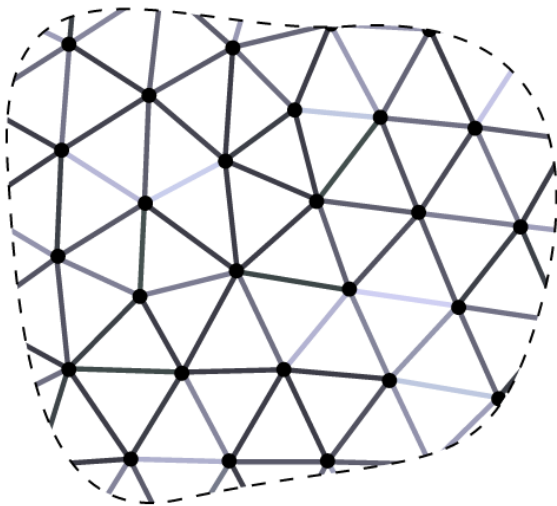
$$= 2x^\top (D_{W \cdot 1} - W)x$$

$$:= 2x^\top Lx$$

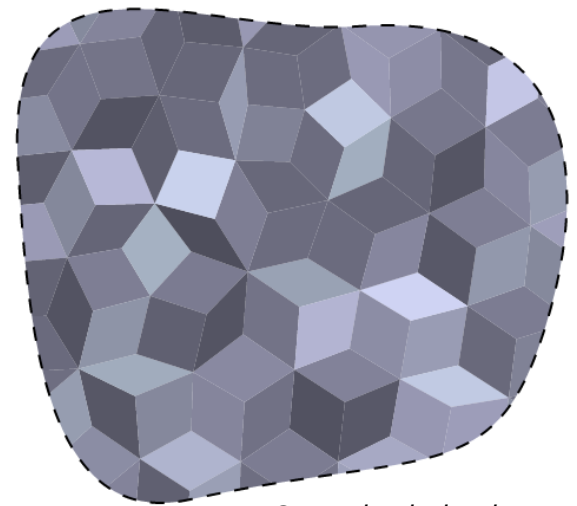
Laplacian matrix!

Differential Geometry Perspective

Dirichlet energy: $E[f(\cdot)] := \int_{\Sigma} \|\nabla f(x)\|_2^2 dA(x)$



$\xrightarrow{\text{interpolate}}$



K. Crane, brickisland.net

Leads to famous **cotangent weights!**
Useful for **interpolation.**

Linear Solver Considerations

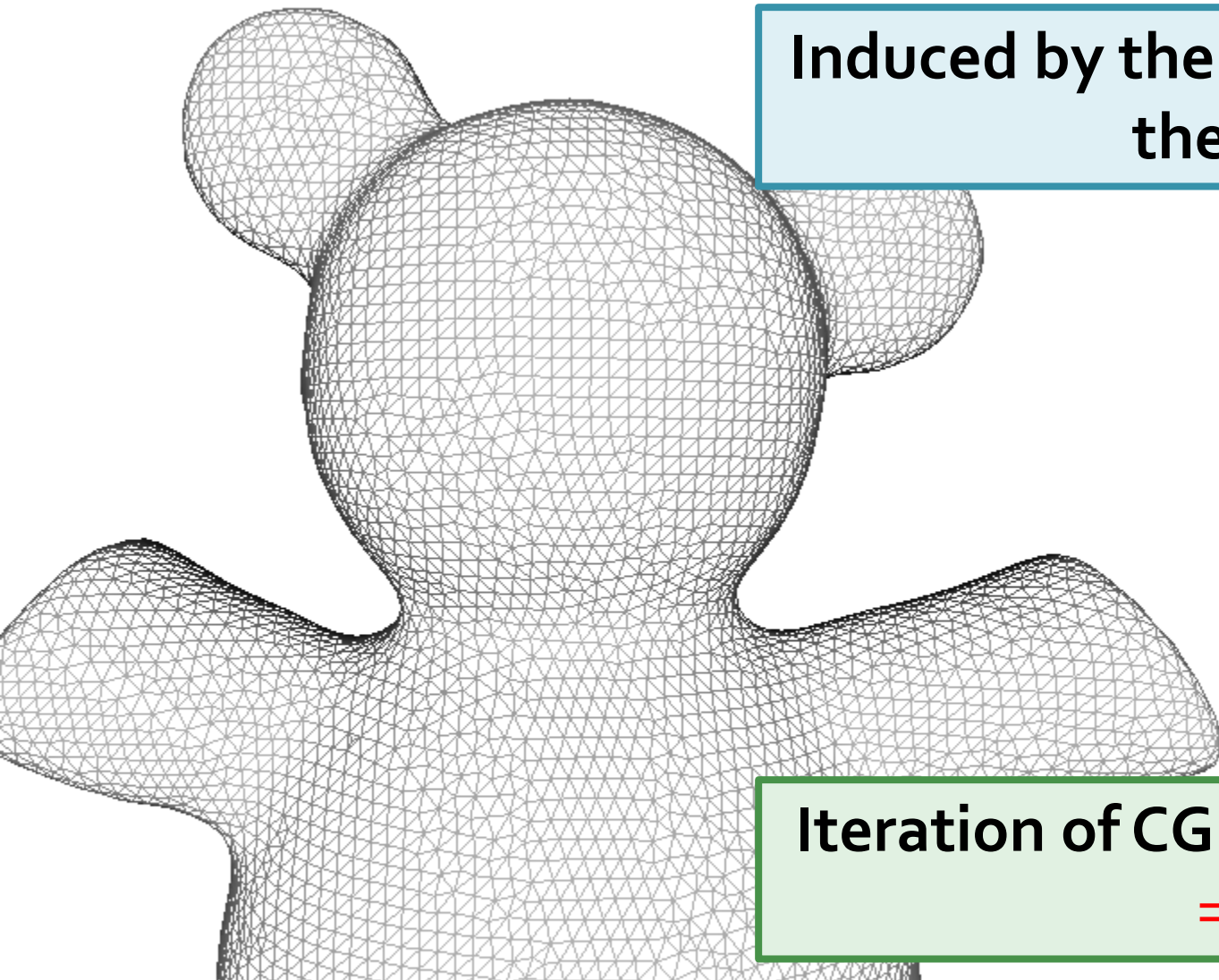
- **Never construct A^{-1} explicitly**
(if you can avoid it)
- **Added structure helps**
Sparsity, symmetry, positive definite

$$\text{inv}(A) * b \ll (A' * A) \setminus (A' * b) \ll A \setminus b$$

Two Classes of Solvers

- **Direct** (*explicit* matrix)
 - Dense: Gaussian elimination/LU, QR for least-squares
 - Sparse: Reordering (SuiteSparse, Eigen)
- **Iterative** (*apply* matrix repeatedly)
 - Positive definite: Conjugate gradients
 - Symmetric: MINRES, GMRES
 - Generic: LSQR

Very Common: Sparsity



Induced by the **connectivity** of the triangle mesh.

Iteration of CG has local effect
⇒ **Precondition!**

Returning to Parameterization

$$\begin{array}{ll} \min_{x_1, \dots, x_{|V|}} & \sum_{(i,j) \in E} w_{ij} \|x_i - x_j\|_2^2 \\ \text{s.t.} & x_v \text{ fixed } \forall v \in V_0 \end{array}$$

**What if
 $V_0 = \{\}$?**

Nontriviality Constraint

$$\left\{ \begin{array}{ll} \min_x & \|Ax\|_2 \\ \text{s.t.} & \|x\|_2 = 1 \end{array} \right\} \mapsto A^\top Ax = \lambda x$$

Prevents trivial solution $x \equiv 0$.

Extract the **smallest eigenvalue**.

Common Situation

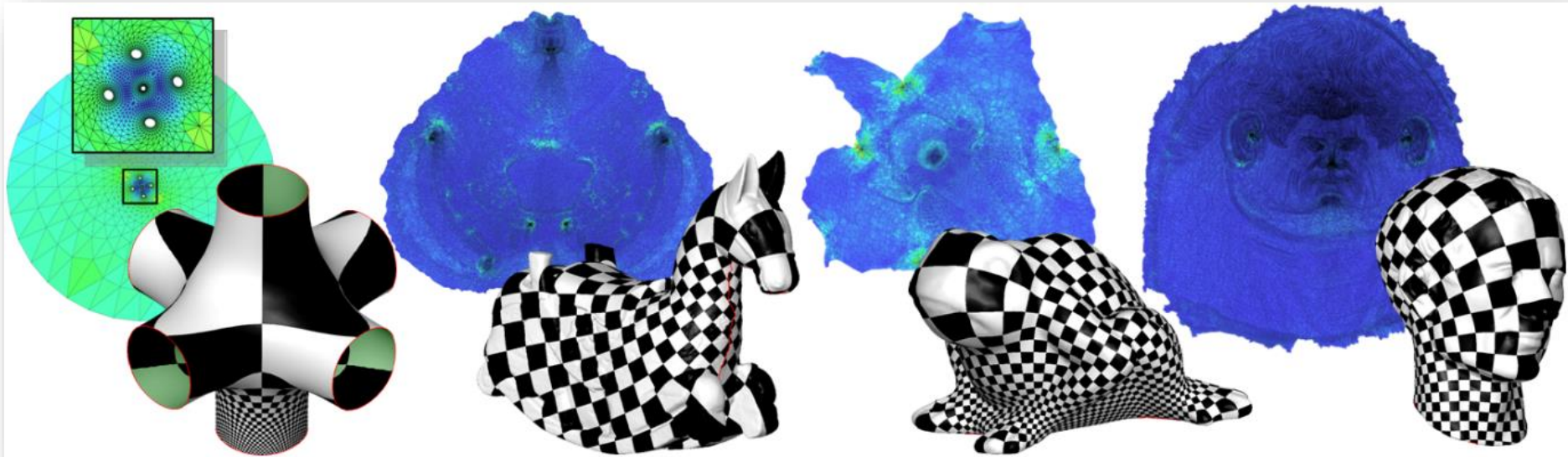
$$\left\{ \begin{array}{ll} \min_x & \|Ax\|_2 \\ \text{s.t.} & \|x\|_2 = 1 \\ & N^\top x = 0 \end{array} \right\} \mapsto A^\top Ax = \lambda x$$

Prevents trivial solution $x \equiv 0$.

N contains basis for null space of A.

Extract the **smallest nonzero eigenvalue**.

Back to Parameterization

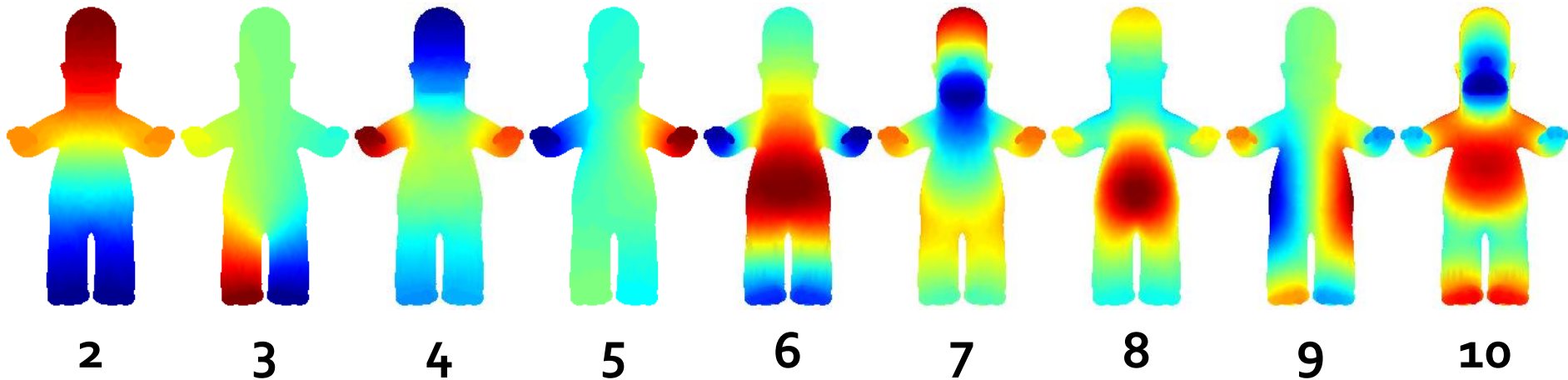


Mullen et al. "Spectral Conformal Parameterization." SGP 2008.

$$\min_u u^\top L_C u \quad \longleftrightarrow \quad L_C u = \lambda B u$$
$$\begin{aligned} u^\top B e &= 0 \\ u^\top B u &= 1 \end{aligned}$$

Continuous Story

$$\min_{\int \phi(x) dA(x)=1} \int \|\nabla \phi(x)\|_2^2 dA(x) \iff \Delta \phi_i(x) = \lambda_i \phi_i(x)$$



“Laplace-Beltrami Eigenfunctions”

Basic Idea of Eigenalgorithms

$$\begin{aligned} A\vec{v} &= c_1 A\vec{x}_1 + \cdots + c_n A\vec{x}_n \\ &= c_1 \lambda_1 \vec{x}_1 + \cdots + c_n \lambda_n \vec{x}_n \text{ since } A\vec{x}_i = \lambda_i \vec{x}_i \end{aligned}$$

$$= \lambda_1 \left(c_1 \vec{x}_1 + \frac{\lambda_2}{\lambda_1} c_2 \vec{x}_2 + \cdots + \frac{\lambda_n}{\lambda_1} c_n \vec{x}_n \right)$$

$$A^2 \vec{v} = \lambda_1^2 \left(c_1 \vec{x}_1 + \left(\frac{\lambda_2}{\lambda_1} \right)^2 c_2 \vec{x}_2 + \cdots + \left(\frac{\lambda_n}{\lambda_1} \right)^2 c_n \vec{x}_n \right)$$

\vdots

$$A^k \vec{v} = \lambda_1^k \left(c_1 \vec{x}_1 + \left(\frac{\lambda_2}{\lambda_1} \right)^k c_2 \vec{x}_2 + \cdots + \left(\frac{\lambda_n}{\lambda_1} \right)^k c_n \vec{x}_n \right).$$

Combining Tools So Far

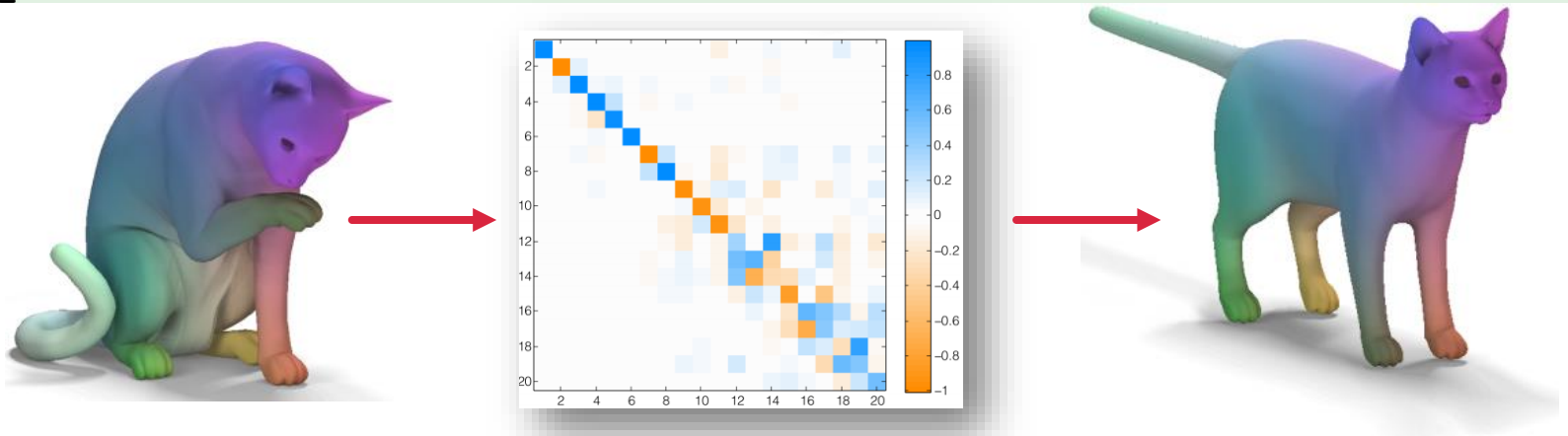
Roughly:

1. Extract Laplace-Beltrami **eigen**functions:

$$L\phi_i = \lambda_i A\phi_i$$

2. Find mapping matrix (**linear** solve!):

$$\min_{A \in \mathbb{R}^{n \times n}} \|AF_0 - F\|_{\text{Fro}}^2 + \alpha \|A\Delta_0 - \Delta A\|_{\text{Fro}}^2$$



Rough Plan

Part I (Justin)

- Vocabulary
- Simple examples
- **Unconstrained optimization**
- Equality-constrained optimization

Unconstrained Optimization

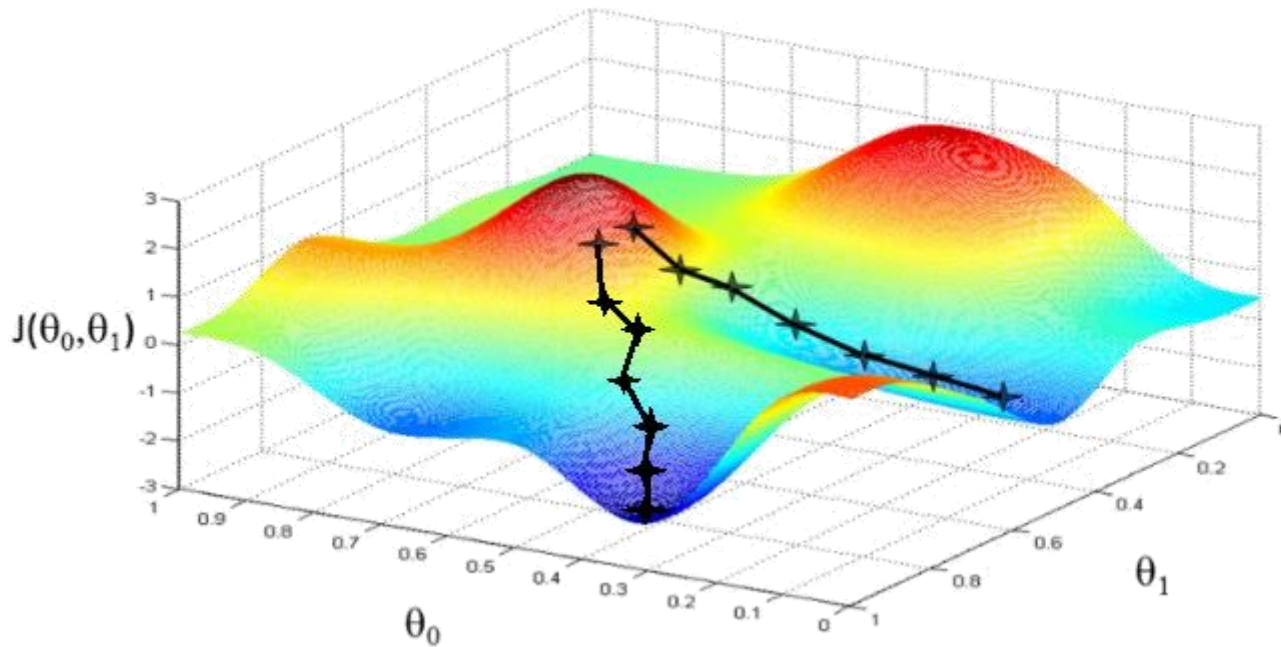
$$\min_x f(x)$$

Unconstrained Optimization

$$\min_x f(x)$$

↑
Unstructured.

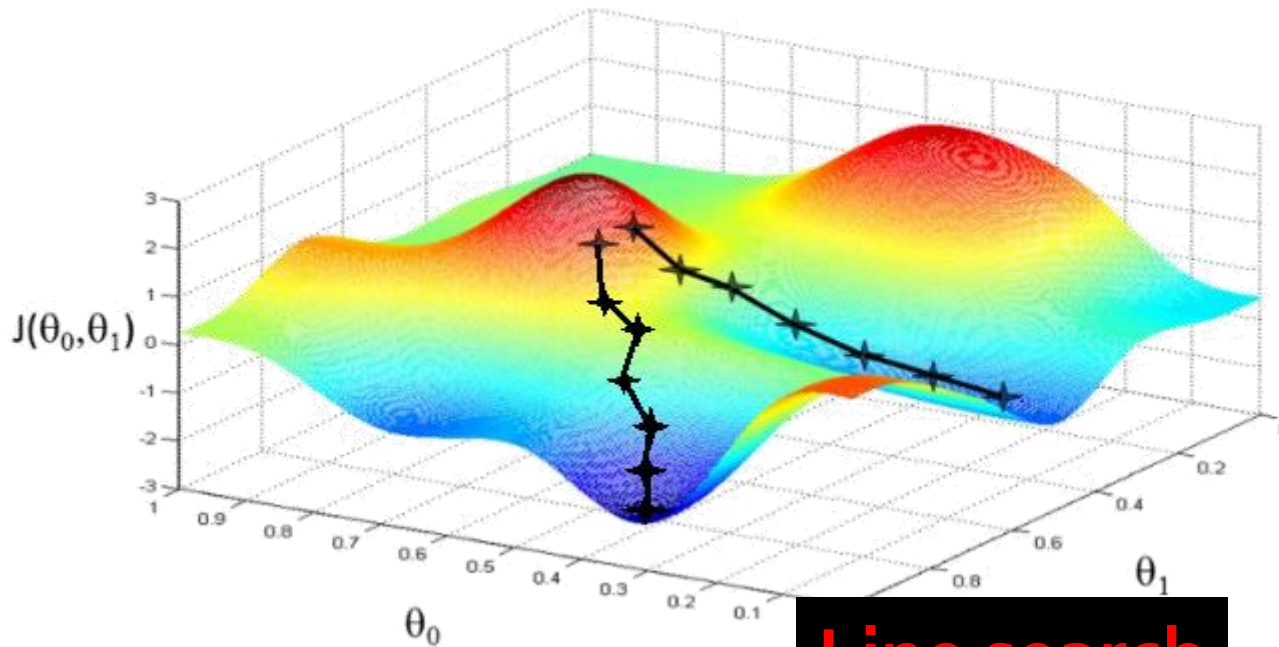
Basic Algorithms



$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

Gradient descent

Basic Algorithms

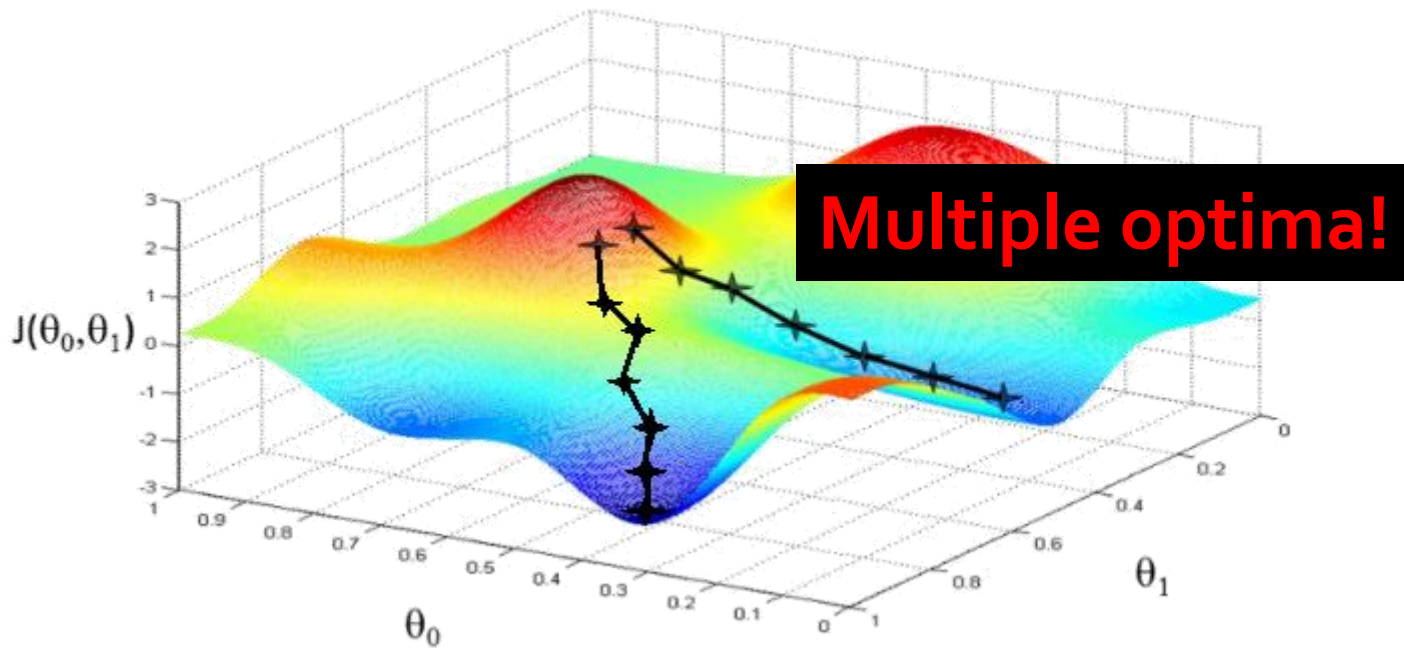


Line search

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

Gradient descent

Basic Algorithms



$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

Gradient descent

Basic Algorithms

$$\lambda_0 = 0, \lambda_s = \frac{1}{2}(1 + \sqrt{1 + 4\lambda_{s-1}^2}), \gamma_s = \frac{1 - \lambda_2}{\lambda_{s+1}}$$

$$y_{s+1} = x_s - \frac{1}{\beta} \nabla f(x_s)$$

$$x_{s+1} = (1 - \gamma_s)y_{s+1} + \gamma_s y_s$$

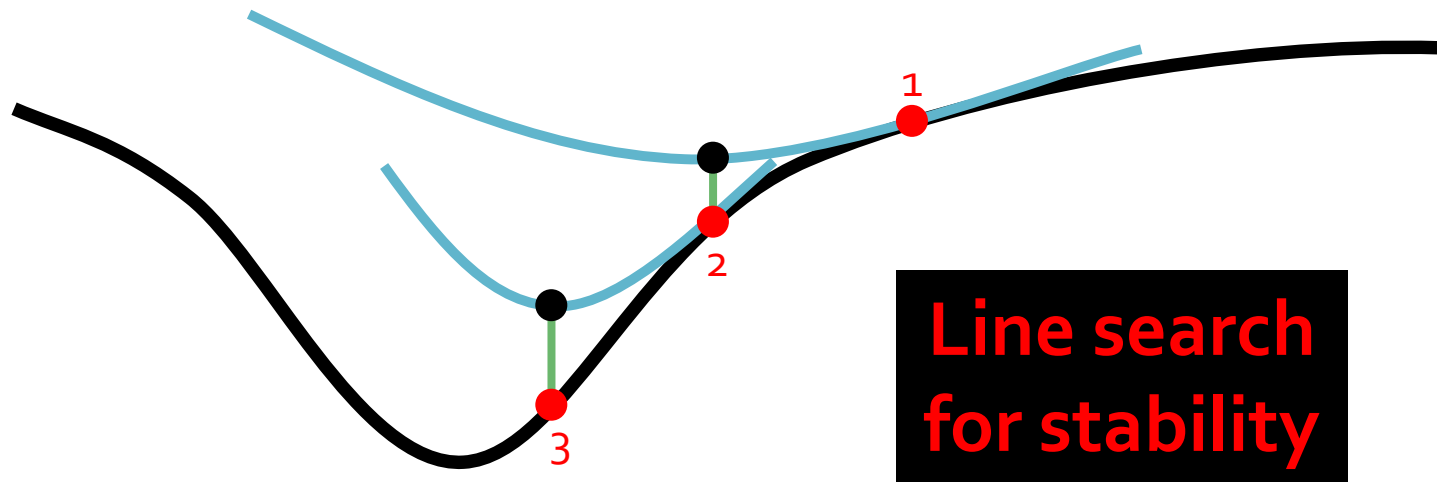
Quadratic convergence on convex problems!

(Nesterov 1983)

Accelerated gradient descent

Basic Algorithms

$$x_{k+1} = x_k - [H f(x_k)]^{-1} \nabla f(x_k)$$



Newton's Method

Basic Algorithms

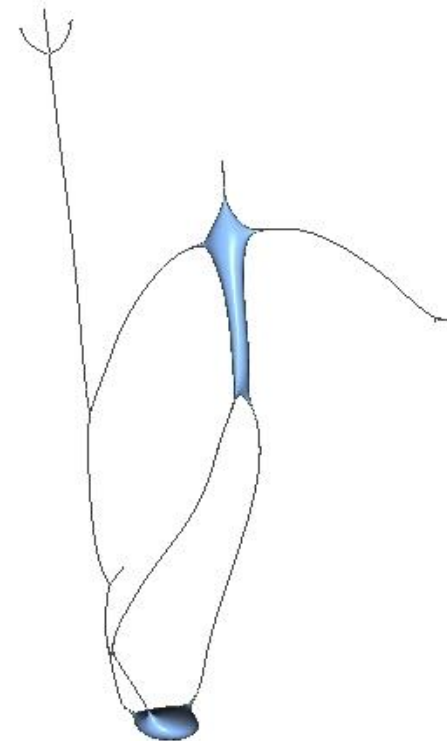
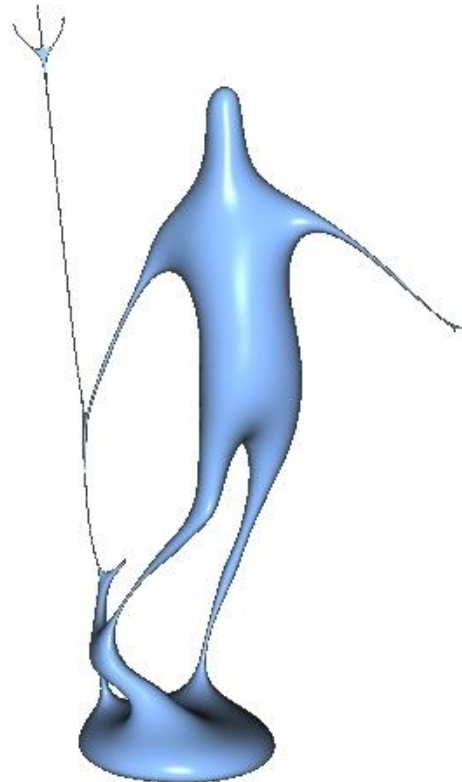
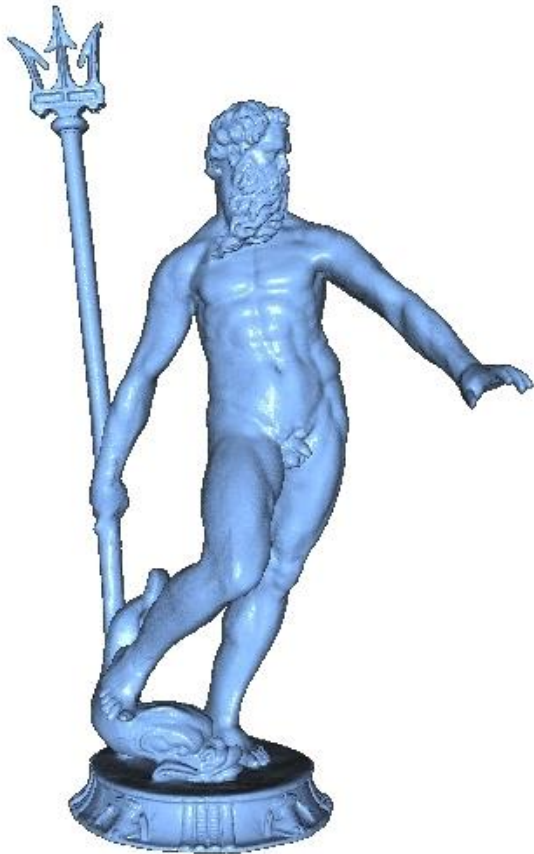
$$x_{k+1} = x_k - M_k^{-1} \nabla f(x_k)$$

Hessian
approximation

- (Often **sparse**) approximation from previous samples and gradients
- Inverse in **closed form**!

Quasi-Newton: BFGS and friends

Geometric Flows



M. Kazhdan

Often *continuous* gradient descent

Example: Shape Interpolation

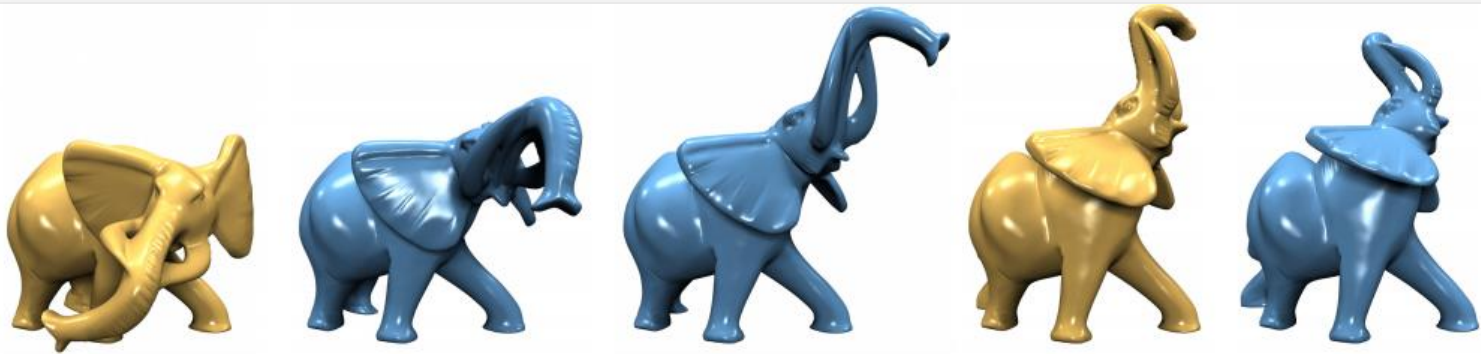


Figure 5: *Interpolation and extrapolation of the yellow example poses. The blending weights are 0, 0.35, 0.65, 1.0, and 1.25.*

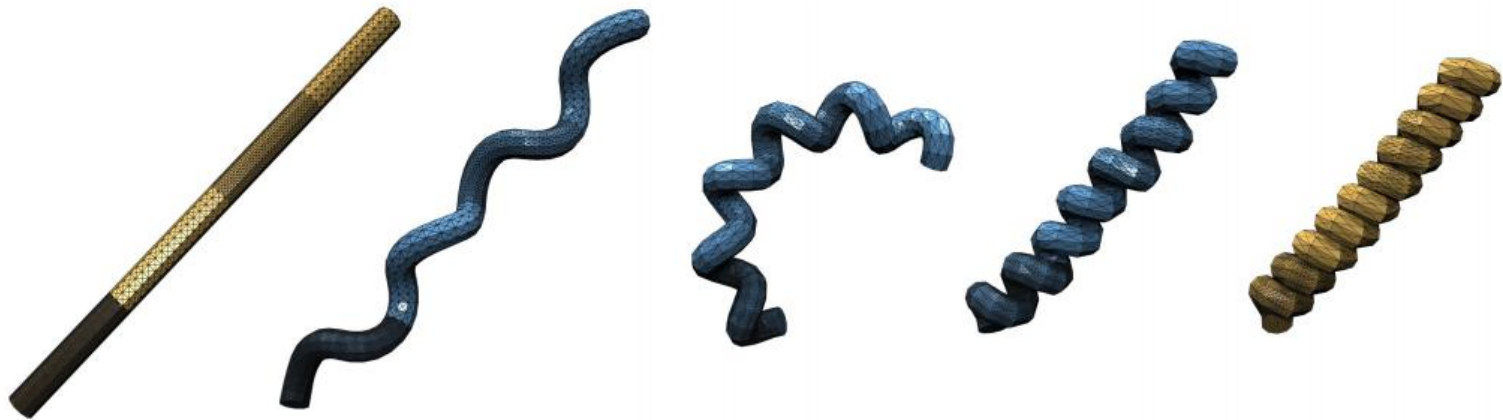


Figure 6: *Interpolation of an adaptively meshed and strongly twisted helix with blending weights 0, 0.25, 0.5, 0.75, 1.0.*

Interpolation Pipeline

Roughly:

1. **Linearly interpolate** edge lengths and dihedral angles.

$$\ell_e^* = (1 - t)\ell_e^0 + t\ell_e^1$$

$$\theta_e^* = (1 - t)\theta_e^0 + t\theta_e^1$$

2. **Nonlinear** optimization for vertex positions.

$$\min_{x_1, \dots, x_m} \lambda \sum_e w_e (\ell_e(x) - \ell_e^*)^2$$

**Sum of squares:
Gauss-Newton**

$$+ \mu \sum_e w_b (\theta_e(x) - \theta_e^*)^2$$

Software

- **Matlab**: `fminunc` or `minfunc`
- **C++**: `libLBFGS`, `dlib`, others

Typically provide functions for **function** and **gradient** (and optionally, **Hessian**).

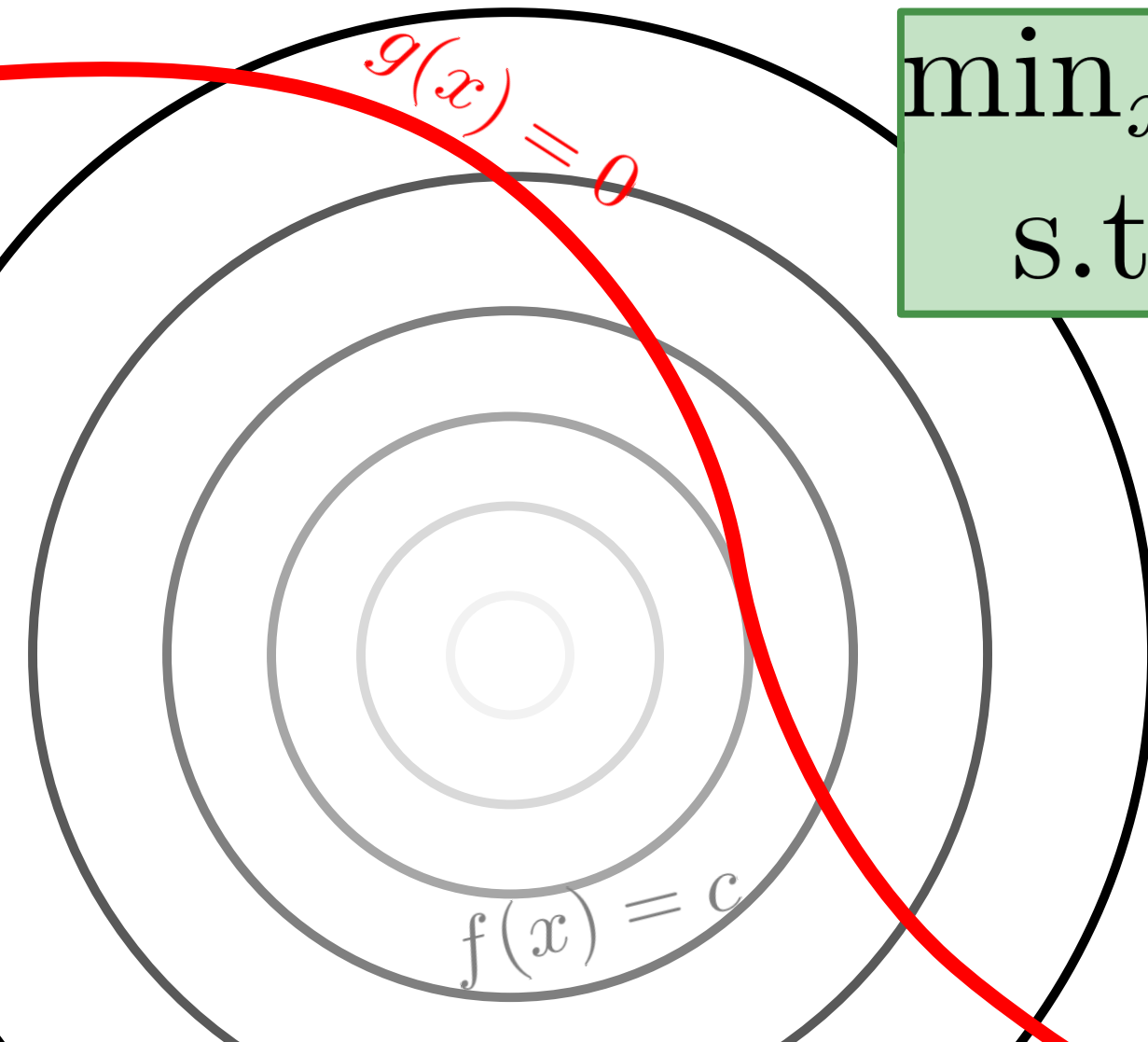
Try several!

Rough Plan

Part I (Justin)

- Vocabulary
- Simple examples
- Unconstrained optimization
- **Equality-constrained optimization**

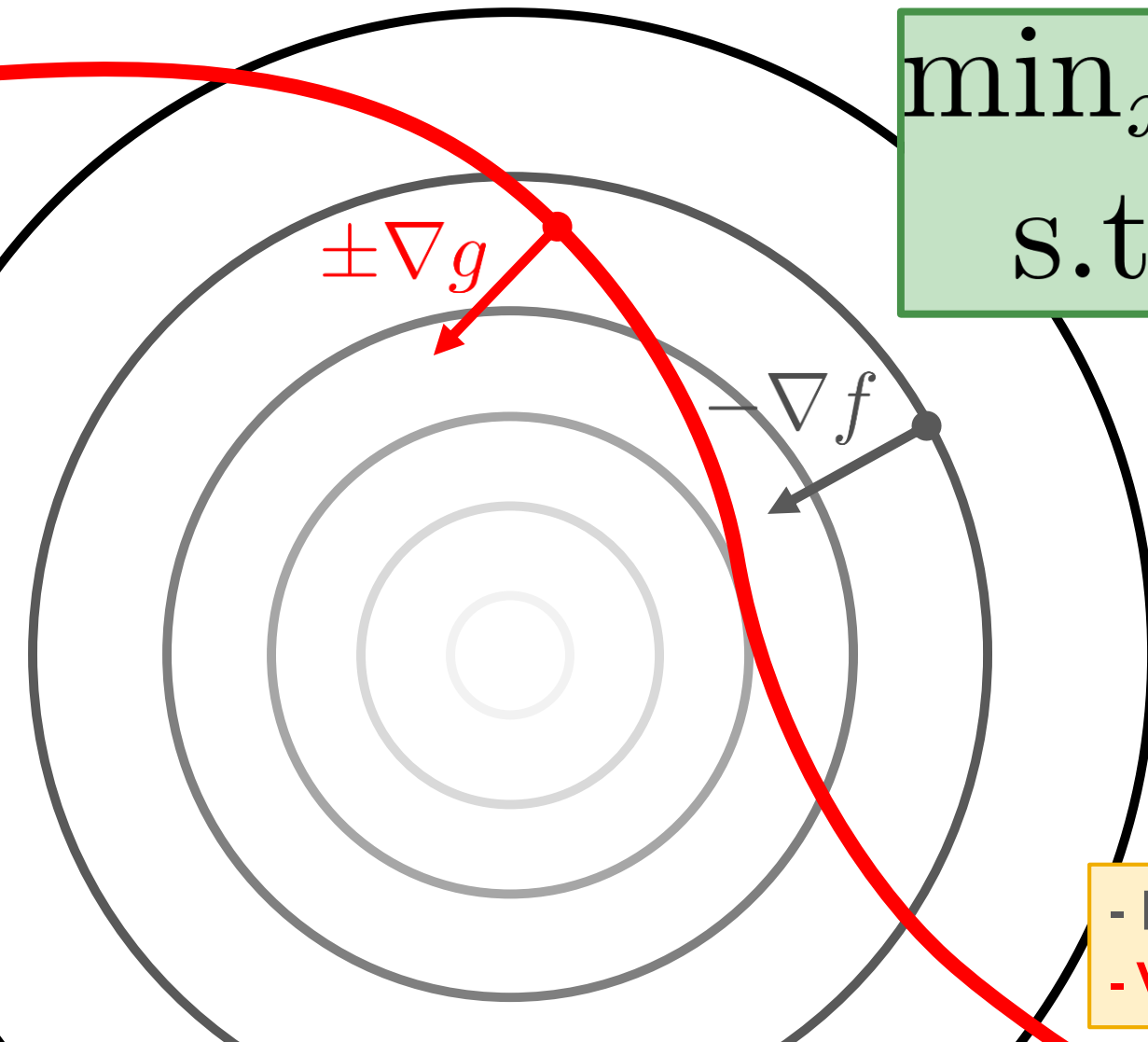
Lagrange Multipliers: Idea



$$\begin{array}{ll} \min_x & f(x) \\ \text{s.t.} & g(x) = 0 \end{array}$$

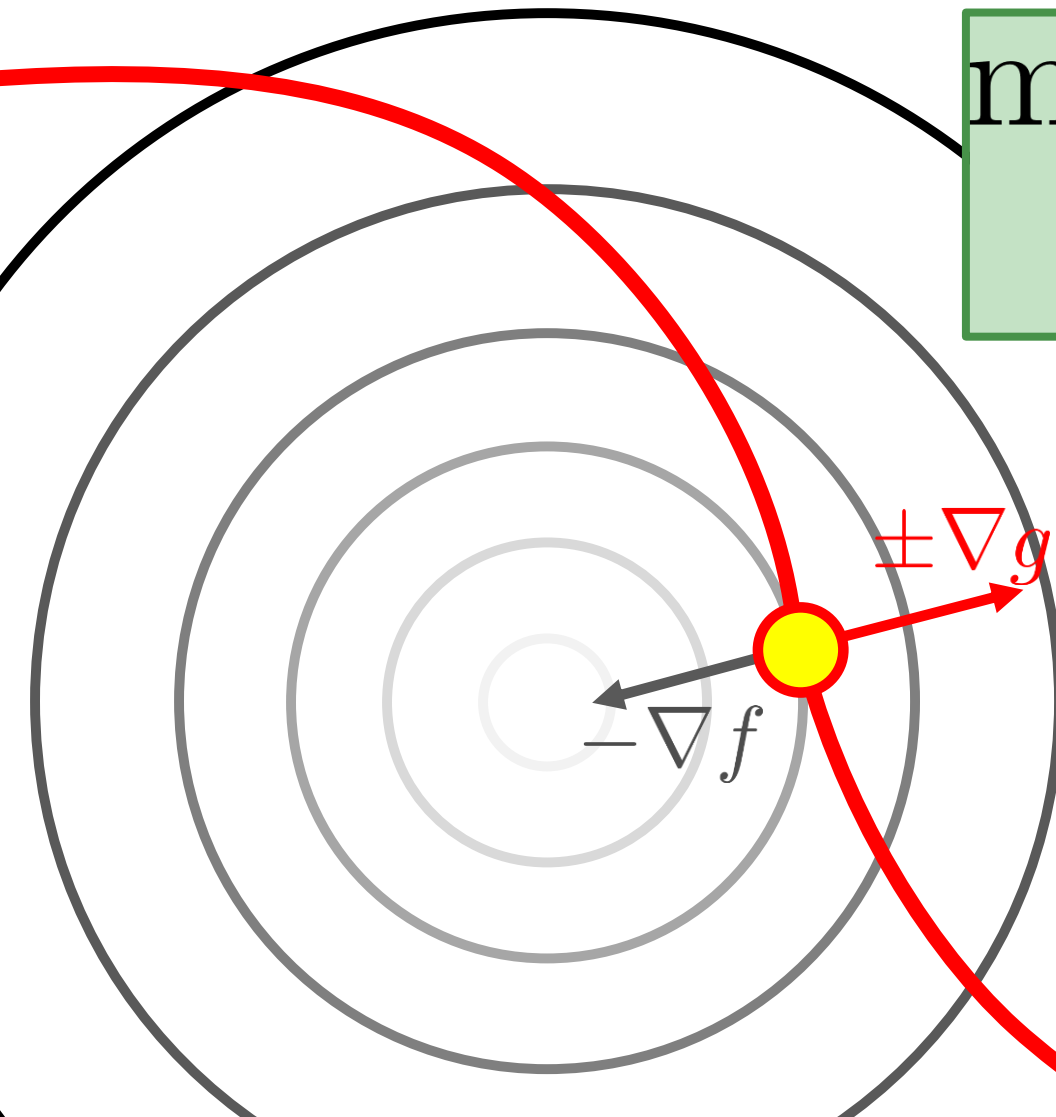
Lagrange Multipliers: Idea

$$\begin{array}{ll} \min_x & f(x) \\ \text{s.t.} & g(x) = 0 \end{array}$$



- Decrease f : $-\nabla f$
- Violate constraint: $\pm \nabla g$

Lagrange Multipliers: Idea



$$\begin{array}{ll} \min_x & f(x) \\ \text{s.t.} & g(x) = 0 \end{array}$$

Want: $\nabla f \parallel \nabla g$
 $\implies \nabla f = \lambda \nabla g$

Example: Symmetric Eigenvectors

$$f(x) = x^\top Ax \implies \nabla f(x) = 2Ax$$

$$g(x) = \|x\|_2^2 \implies \nabla g(x) = 2x$$

$$\implies Ax = \lambda x$$

Use of Lagrange Multipliers

Turns constrained optimization into
unconstrained root-finding.

$$\nabla f(x) = \lambda \nabla g(x)$$

$$g(x) = 0$$

Many Options

- **Reparameterization**

Eliminate constraints to reduce to unconstrained case

- **Newton's method**

Approximation: quadratic function with linear constraint

- **Penalty method**

Augment objective with barrier term, e.g. $f(x) + \rho|g(x)|$

Schur Complement Reduction

Recall:

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^\top A x - b^\top x + c \\ \text{s.t.} \quad & Mx = v \end{aligned}$$

(assume A is symmetric and positive definite)



$$\begin{pmatrix} A & M^\top \\ M & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} b \\ v \end{pmatrix}$$

Schur Complement Reduction

$$\begin{pmatrix} A & M^\top \\ M & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} b \\ v \end{pmatrix}$$

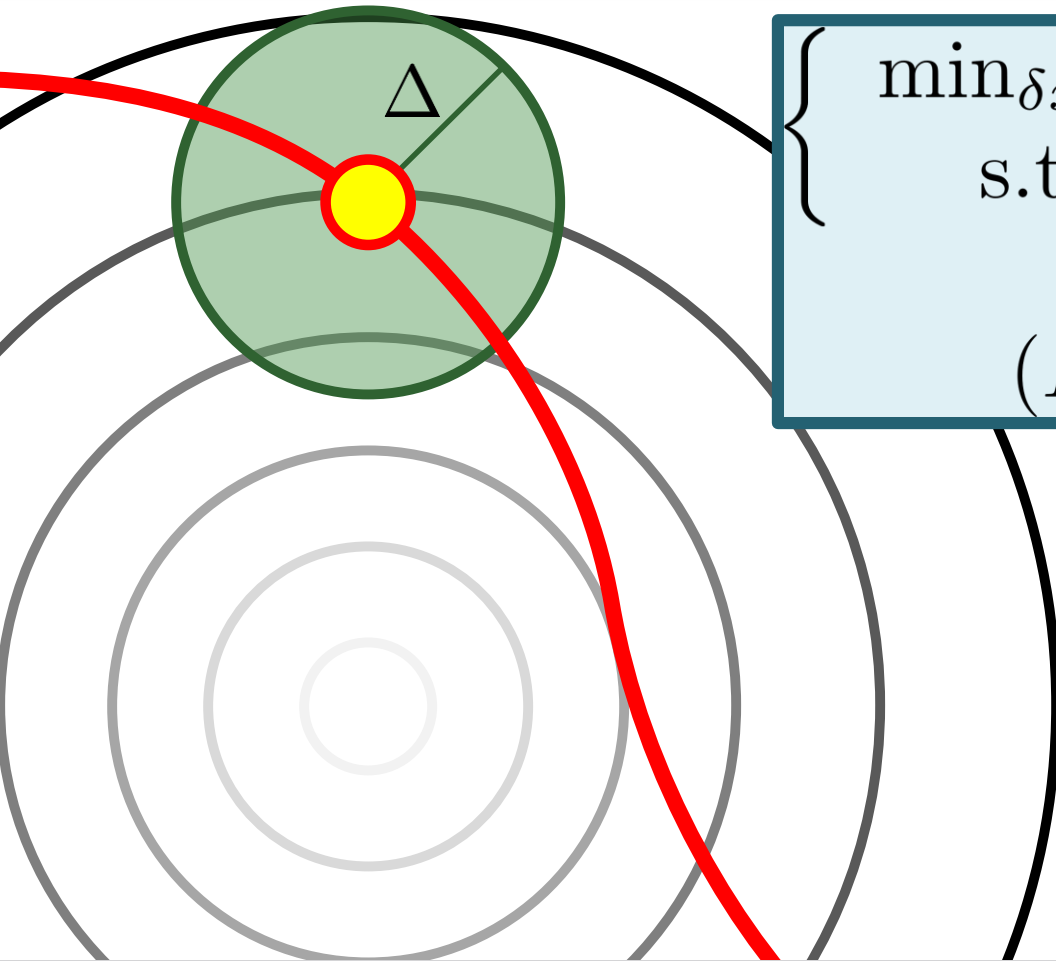
No longer positive definite!

$$Ax + M^\top \lambda = b \implies Mx + MA^{-1}M^\top \lambda = MA^{-1}b$$

$$Mx = v \implies v + MA^{-1}M^\top \lambda = MA^{-1}b$$

$$\implies \begin{cases} MA^{-1}M^\top \lambda = b - v \\ x = A^{-1}(b - M^\top \lambda) \end{cases}$$

Trust Region Methods



$$\left\{ \begin{array}{ll} \min_{\delta x} & \frac{1}{2} \delta x^\top H \delta x + w^\top x \\ \text{s.t.} & \|\delta x\|_2^2 \leq \Delta \end{array} \right\}$$

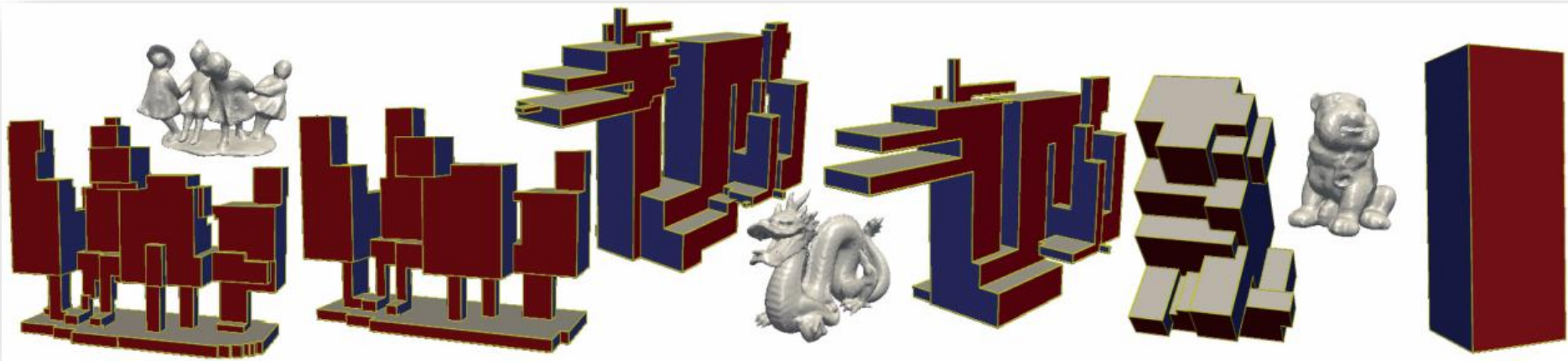
↓

$$(H + \lambda I) \delta x = -w$$

**Fix (or adjust)
damping parameter
 $\lambda > 0$.**

Example: Levenberg-Marquardt

Example: Polycube Maps



Huang et al. "L1-Based Construction of Polycube Maps from Complex Shapes." TOG 2014.

Nonlinear Constrained Problem

Align with
coordinate axes



$$\begin{array}{ll} \min_X & \sum_{b_i} \mathcal{A}(b_i; X) \|n(b_i; X)\|_1 \\ \text{s.t.} & \sum_{b_i} \mathcal{A}(b_i; X) = \sum_{b_i} \mathcal{A}(b_i; X_0) \end{array}$$



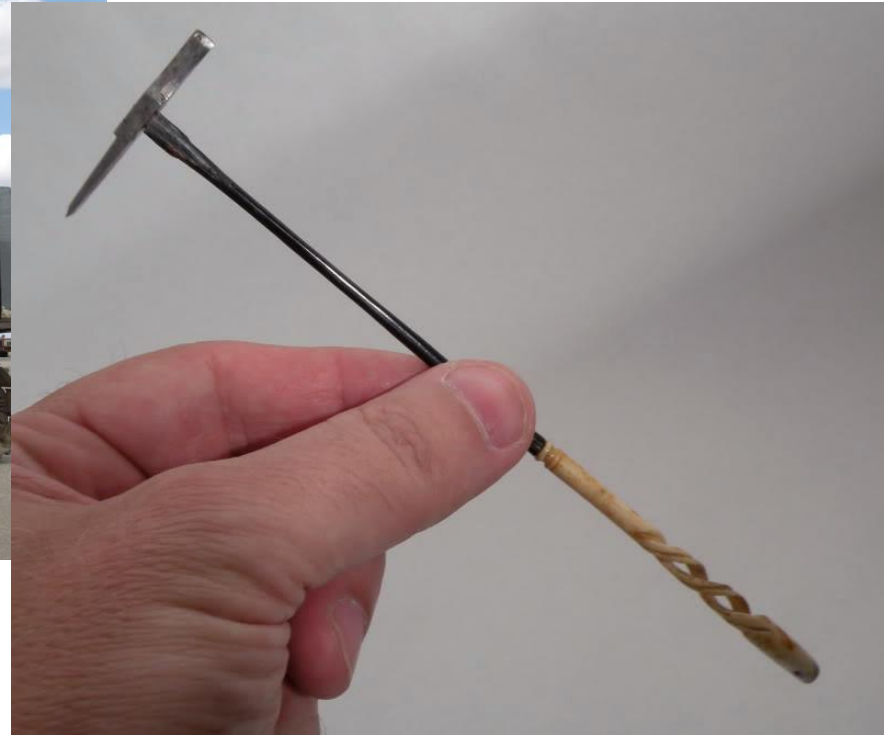
Preserve area

Note: Final method includes several more terms!

Convex Optimization Tools



versus



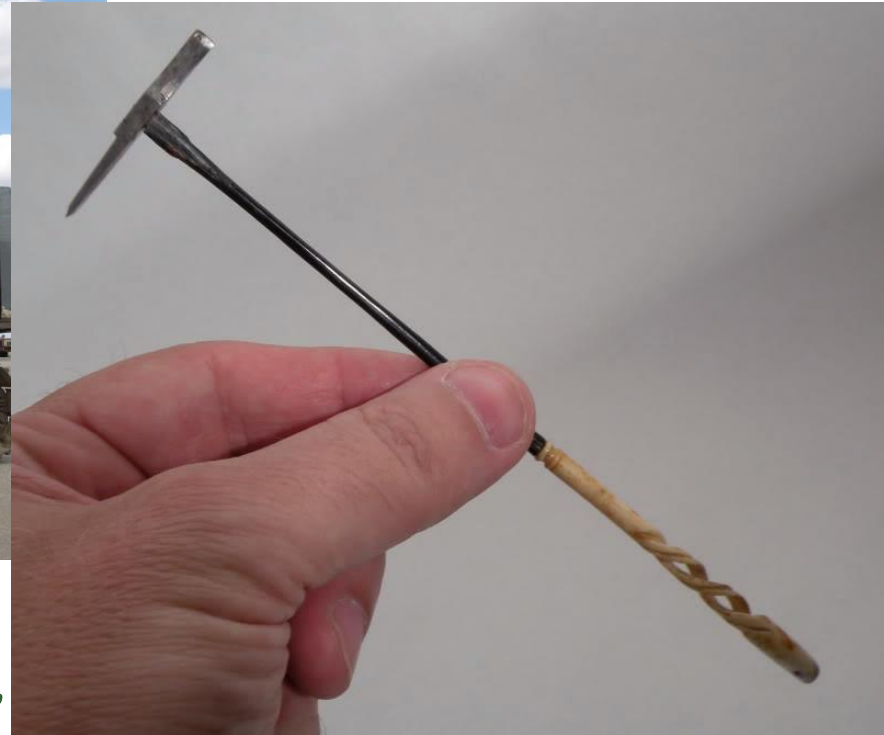
Try lightweight options

Convex Optimization Tools



versus

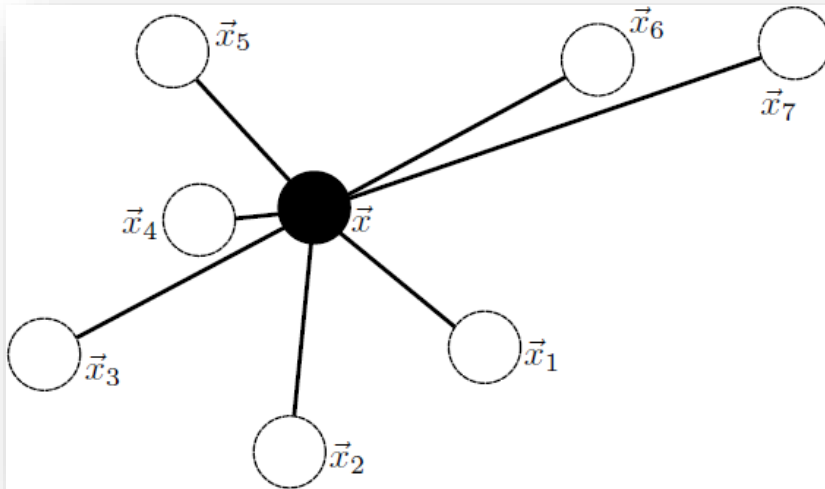
Sometimes work for non-convex problems...



Try lightweight options

Iteratively Reweighted Least Squares

$$\min_x \sum_i \phi(x^\top a_i + b_i) \leftrightarrow \begin{cases} \min_{x, y_i} & \sum_i y_i (x^\top a_i + b_i)^2 \\ \text{s.t.} & y_i = \phi(x^\top a_i + b_i) (x^\top a_i + b_i)^{-2} \end{cases}$$



**“Geometric
median”**

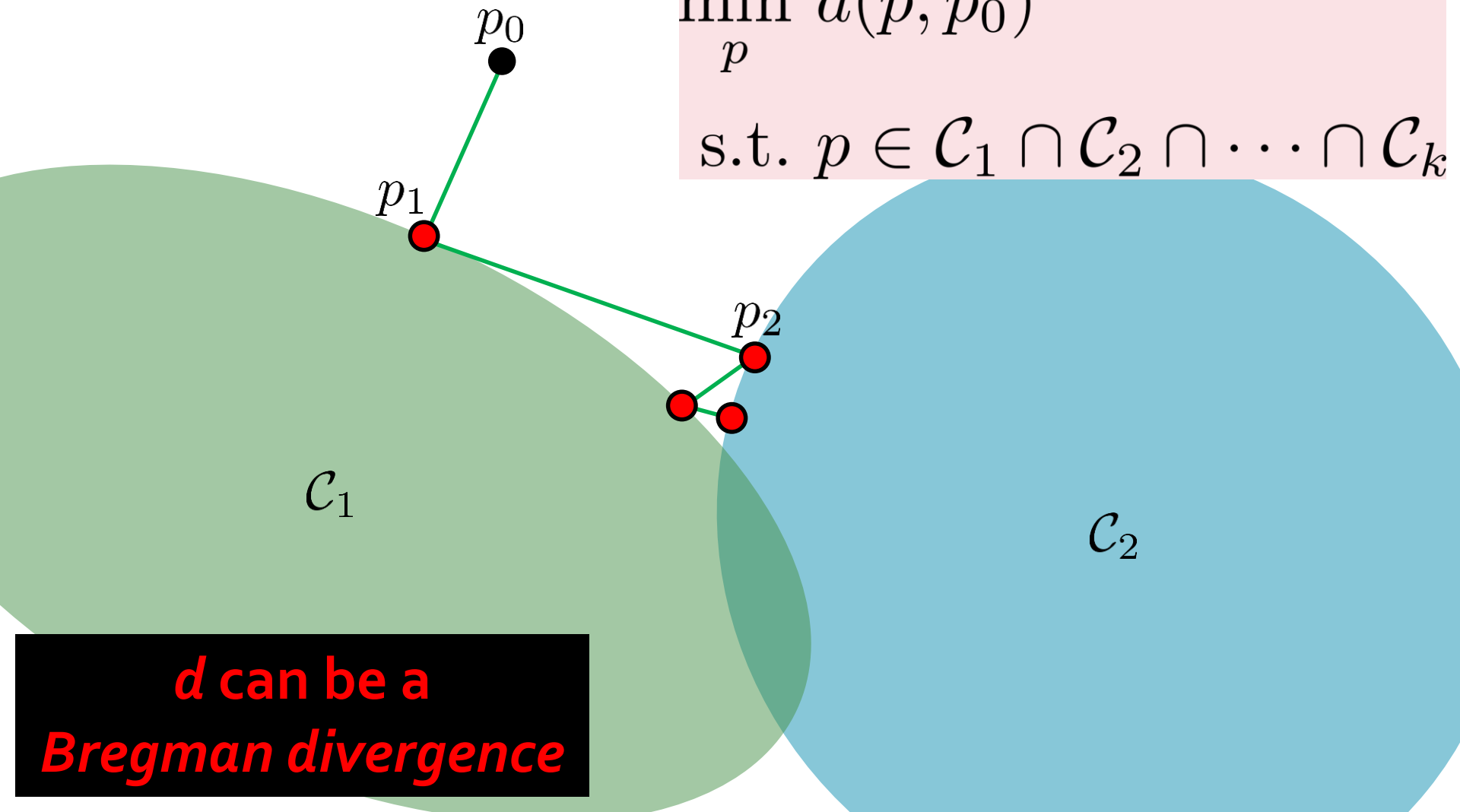
$$\min_x \sum_i \|x - p_i\|_2 \implies \begin{cases} x \leftarrow \min_x \sum_i y_i \|x - p_i\|_2^2 \\ y_i \leftarrow \|x - p_i\|_2^{-1} \end{cases}$$

Repeatedly solve linear systems

Alternating Projection

$$\min_p d(p, p_0)$$

$$\text{s.t. } p \in \mathcal{C}_1 \cap \mathcal{C}_2 \cap \cdots \cap \mathcal{C}_k$$



*d can be a
Bregman divergence*

Iterative Shrinkage-Thresholding

$$x_{t+1} = x_t - \eta \nabla f(x_t)$$

$$\iff x_{t+1} = \arg \min_x \left[f(x_t) + \nabla f(x_t)^\top (x - x_t) + \frac{1}{2\eta} \|x - x_t\|_2^2 \right]$$

$$\iff x_{t+1} = \arg \min_x \frac{1}{2\eta} \|x - (x_t - \eta \nabla f(x_t))\|_2^2$$

Decompose as sum of hard part f and easy part g .

To minimize $f(x) + g(x)$:

$$x_{t+1} = \arg \min_x \left[g(x) + \frac{1}{2\eta} \|x - (x_t - \eta \nabla f(x_t))\|_2^2 \right]$$

FISTA combines with Nesterov descent!

Augmented Lagrangians

$$\begin{array}{ll}\min_x & f(x) \\ \text{s.t.} & g(x) = 0\end{array}$$

↓

$$\begin{array}{ll}\min_x & f(x) + \frac{\rho}{2} \|g(x)\|_2^2 \\ \text{s.t.} & g(x) = 0\end{array}$$

Does nothing when
constraint is
satisfied

Add constraint to objective

Alternating Direction Method of Multipliers (ADMM)

$$\begin{array}{ll}\min_{x,z} & f(x) + g(z) \\ \text{s.t.} & Ax + Bz = c\end{array}$$

$$\Lambda_\rho(x, z; \lambda) = f(x) + g(z) + \lambda^\top (Ax + Bz - c) + \frac{\rho}{2} \|Ax + Bz - c\|_2^2$$

$$x \leftarrow \arg \min_x \Lambda_\rho(x, z, \lambda)$$

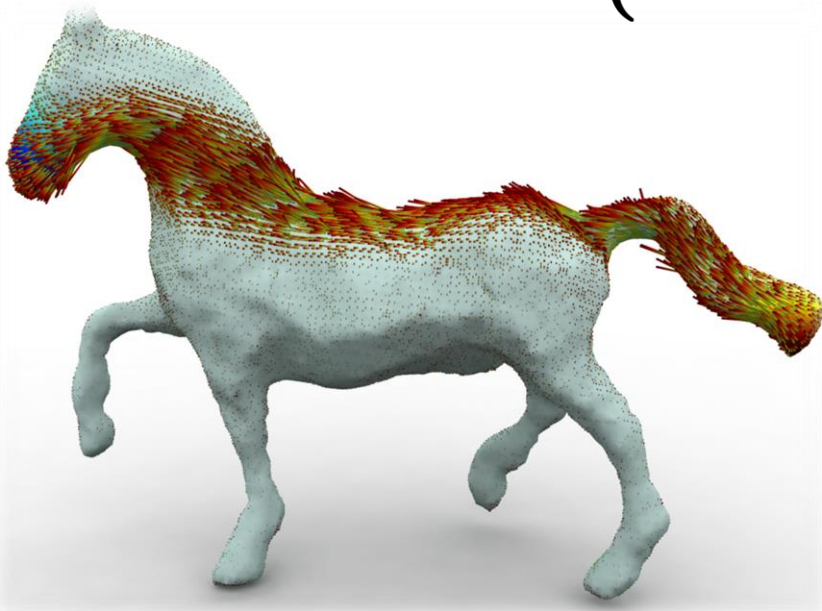
$$z \leftarrow \arg \min_z \Lambda_\rho(x, z, \lambda)$$

$$\lambda \leftarrow \lambda + \rho(Ax + Bz - c)$$

The Art of ADMM “Splitting”

$$\left\{ \begin{array}{ll} \min_J & \sum_i \|J_i\|_2 \\ \text{s.t.} & MJ = b \end{array} \right\} \longrightarrow \left\{ \begin{array}{ll} \min_{J, \bar{J}} & \sum_i \left(\|J_i\|_2 + \frac{\rho}{2} \|J_i - \bar{J}_i\|_2^2 \right) \\ \text{s.t.} & M\bar{J} = b \\ & J = \bar{J} \end{array} \right\}$$

**Augmented
part**



Takes some practice!

Solomon et al. “Earth Mover’s Distances on Discrete Surfaces.” SIGGRAPH 2014.

Want two *easy* subproblems

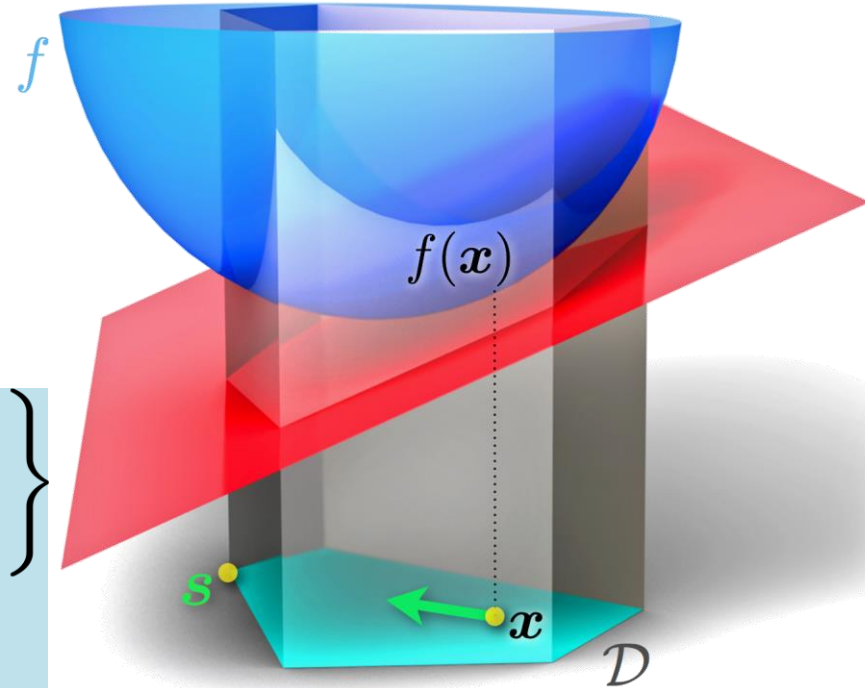
Frank-Wolfe

To minimize $f(x)$ s.t. $x \in \mathcal{D}$:

$$s_k \leftarrow \begin{cases} \arg \min_s & s^\top \nabla f(x_k) \\ \text{s.t. } & s \in \mathcal{D} \end{cases}$$

$$\gamma \leftarrow \frac{2}{k+2}$$

$$x_{k+1} \leftarrow x_k + \gamma(s_k - x_k)$$



https://en.wikipedia.org/wiki/Frank%E2%80%93Wolfe_algorithm

Linearize objective, not constraints