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

Numerical Tools for Geometry

Justin Solomon

MIT, Spring 2017

<announcements>

Announcements

- Nanoquiz on Thursday
  - It will be easy!
- Yes, this course is a TQE!

Homework 1 Posted

(demo in browser)

Course Project

- Instructions on course website
- Individual or groups of two
- Implement and extend a relevant technique

- Milestones:
  - Proposal (500 words)
  - Checkpoint (≤2 pages)
  - Writeup (6-10 pages)
  - Presentation (8-10 minutes)

Start early:
We will help you!

</announcements>

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

Numerical Tools for Geometry

Justin Solomon

MIT, Spring 2017

MIT EECS

Motivation

Numerical problems abound

in modern geometry applications.

Quick summary!

Mostly for common ground: You may already know this material.
First half is important; remainder summarizes interesting recent tools.

Two Roles

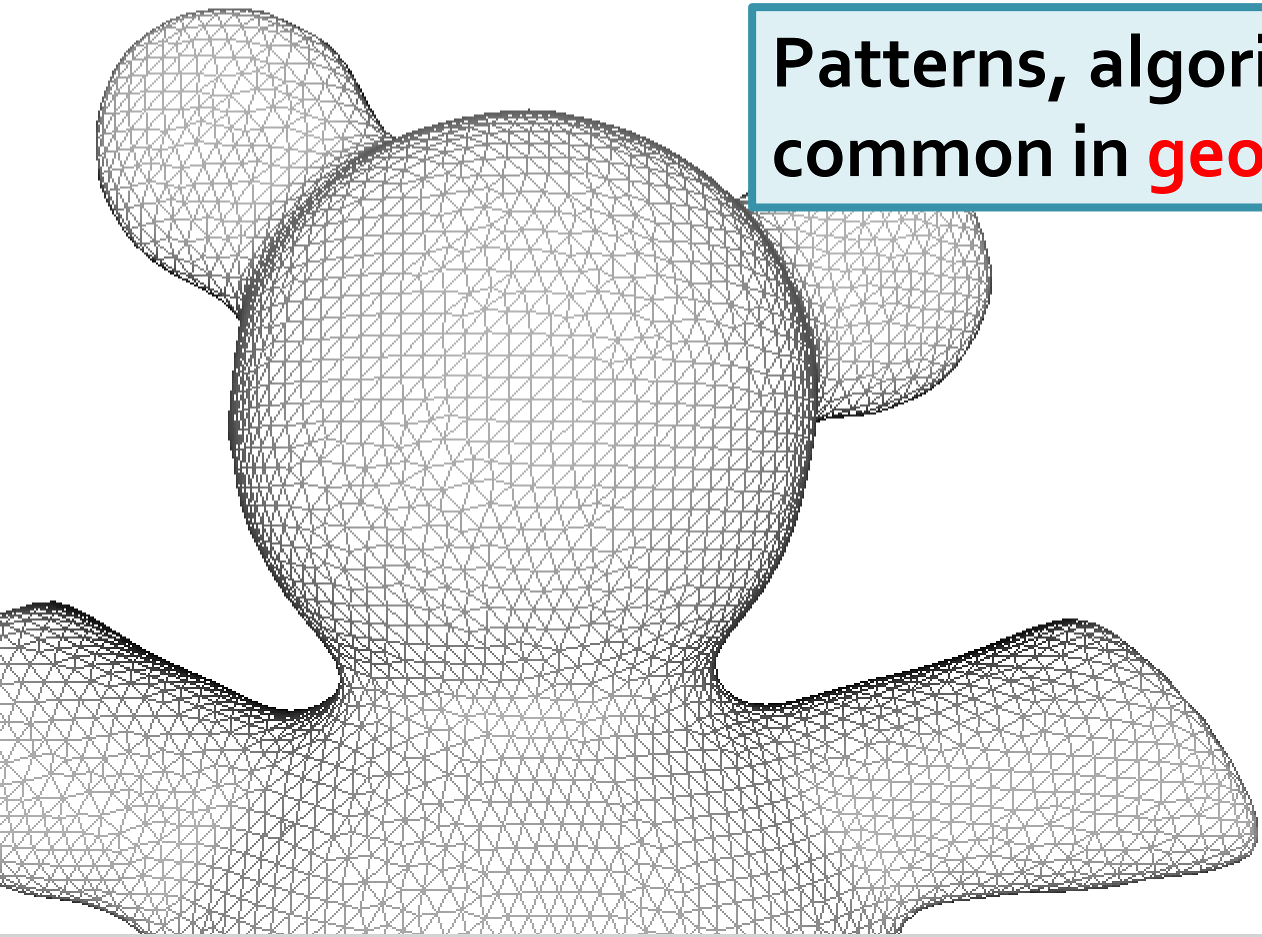
- **Client**

  Which optimization tool is relevant?

- **Designer**

  Can I design an algorithm for this problem?

Our Bias



Patterns, algorithms, & examples common in geometry.

Numerical analysis is a <u>huge</u> field.

Rough Plan

- Linear problems
- Unconstrained optimization
- Equality-constrained optimization
- Variational problems

Rough Plan

- **Linear problems**
- Unconstrained optimization
- Equality-constrained optimization
- Variational problems

Vector Spaces and Linear Operators

$$\mathcal{L}[\vec{x} + \vec{y}] = \mathcal{L}[\vec{x}] + \mathcal{L}[\vec{y}]$$

$$\mathcal{L}[c\vec{x}] = c\mathcal{L}[\vec{x}]$$

Abstract Example

$$C^{\infty}(\mathbb{R})$$

$$\mathcal{L}[f] := {}^{df}\!/_{dx}$$

Eigenvectors?

In Finite Dimensions

$$\underbrace{A}_{\text{matrix}} \quad \underbrace{\vec{x}}_{\text{vector}}$$

$$\underbrace{\vec{x} \mapsto A\vec{x}}_{\text{linear operator}}$$

Linear System of Equations

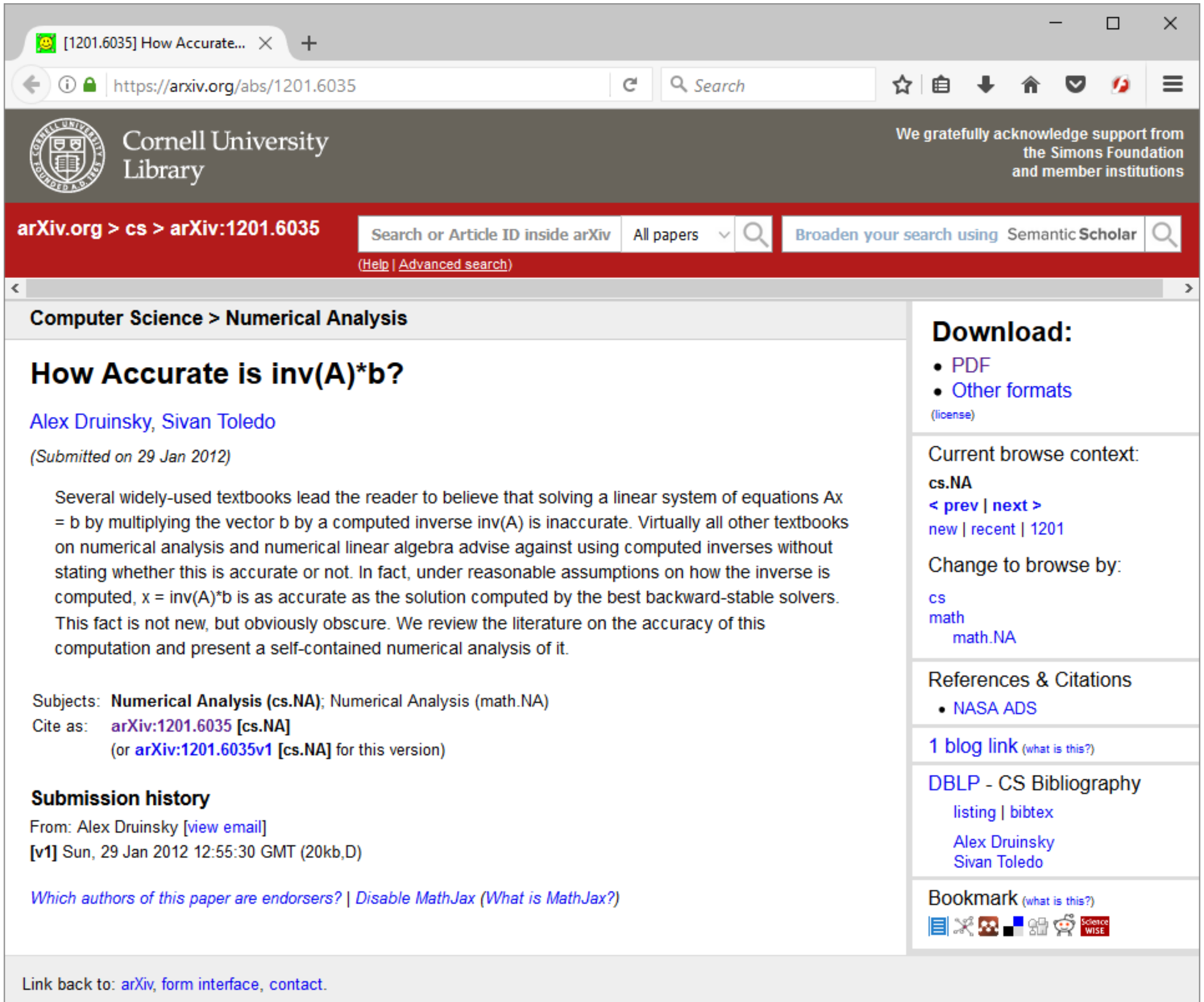
$$\begin{pmatrix} & & \\ & A & \\ & & \end{pmatrix} \begin{pmatrix} \\ \vec{x} \\ \\ \end{pmatrix} = \begin{pmatrix} \\ \vec{b} \\ \\ \end{pmatrix}$$

Simple "inverse problem"

Common Strategies

- **Gaussian elimination**
  - $O(n^3)$ time to solve $Ax=b$ or to invert
- **But:** Inversion is unstable and slower!
- **Never ever compute A^{-1} if you can avoid it.**

Interesting Perspective


[1201.6035] How Accurate...

https://arxiv.org/abs/1201.6035

Cornell University Library

We gratefully acknowledge support from the Simons Foundation and member institutions

arXiv.org > cs > arXiv:1201.6035

Search or Article ID inside arXiv | All papers | Broaden your search using Semantic Scholar

(Help | Advanced search)

Computer Science > Numerical Analysis

How Accurate is inv(A)*b?

Alex Druinsky, Sivan Toledo

(Submitted on 29 Jan 2012)

Several widely-used textbooks lead the reader to believe that solving a linear system of equations Ax = b by multiplying the vector b by a computed inverse inv(A) is inaccurate. Virtually all other textbooks on numerical analysis and numerical linear algebra advise against using computed inverses without stating whether this is accurate or not. In fact, under reasonable assumptions on how the inverse is computed, x = inv(A)*b is as accurate as the solution computed by the best backward-stable solvers. This fact is not new, but obviously obscure. We review the literature on the accuracy of this computation and present a self-contained numerical analysis of it.

Subjects: **Numerical Analysis (cs.NA)**; Numerical Analysis (math.NA)

Cite as: **arXiv:1201.6035 [cs.NA]**

(or **arXiv:1201.6035v1 [cs.NA]** for this version)

Submission history

From: Alex Druinsky [view email]

[v1] Sun, 29 Jan 2012 12:55:30 GMT (20kb,D)

Which authors of this paper are endorsers? | *Disable MathJax* (*What is MathJax?*)

Download:

- PDF
- Other formats

(license)

Current browse context:

cs.NA

< prev | next >

new | recent | 1201

Change to browse by:

cs

math

math.NA

References & Citations

- NASA ADS

1 blog link (what is this?)

DBLP - CS Bibliography

listing | bibtex

Alex Druinsky

Sivan Toledo

Bookmark (what is this?)

Link back to: arXiv, form interface, contact.

Simple Example

$$\frac{d^2 f}{dx^2} = g, f(0) = f(1) = 0$$

$$\begin{pmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & & \ddots & & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} = \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{pmatrix}$$

Structure?

$$\begin{pmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & & \ddots & & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{pmatrix}$$

Linear Solver Considerations

- **Never construct A^{-1} explicitly**
  (if you can avoid it)

- **Added structure helps**
  Sparsity, symmetry, positive definiteness, bandedness

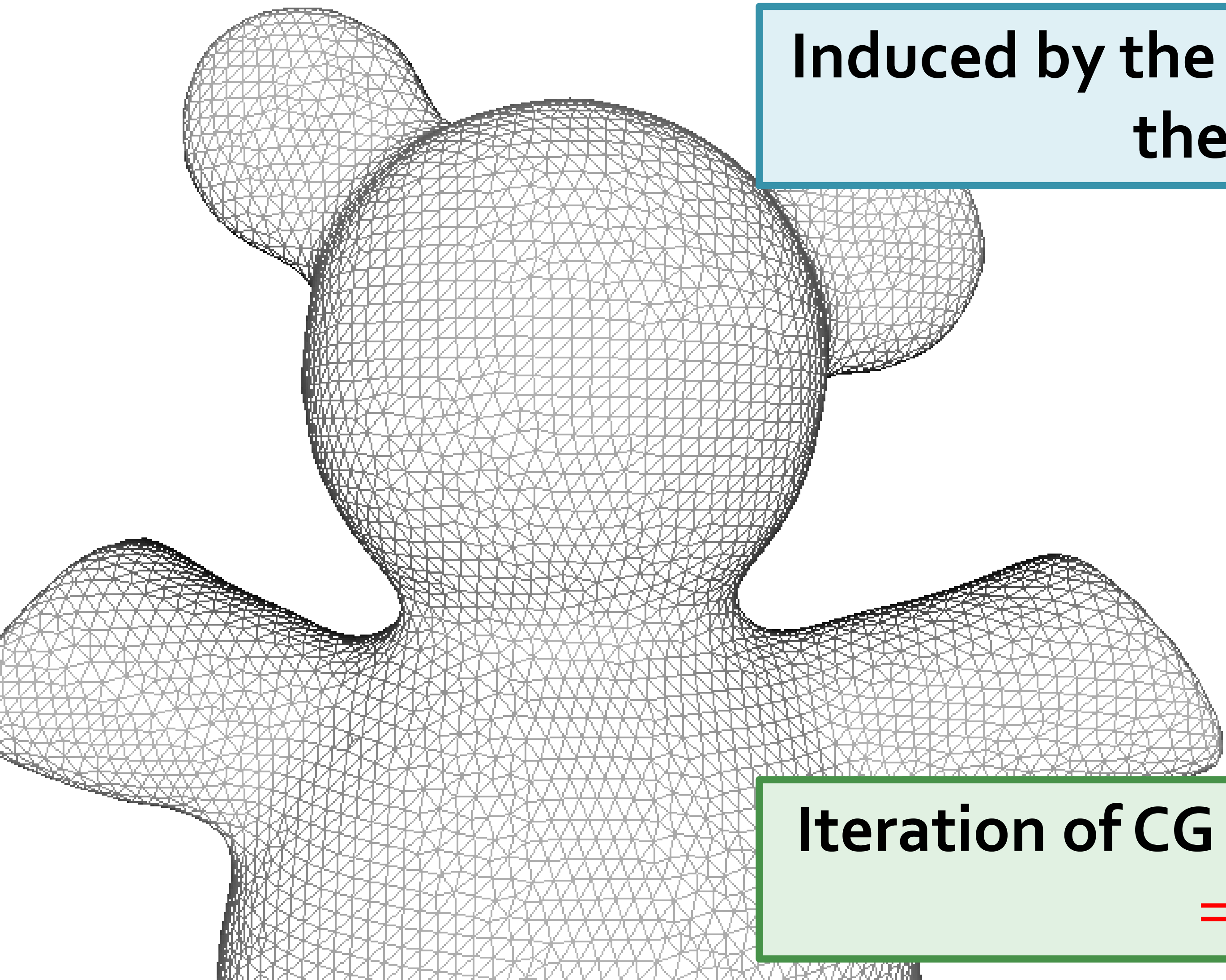
$$\texttt{inv(A)*b} \ll \texttt{(A'*A)}\backslash\texttt{(A'*b)} \ll \texttt{A}\backslash\texttt{b}$$

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

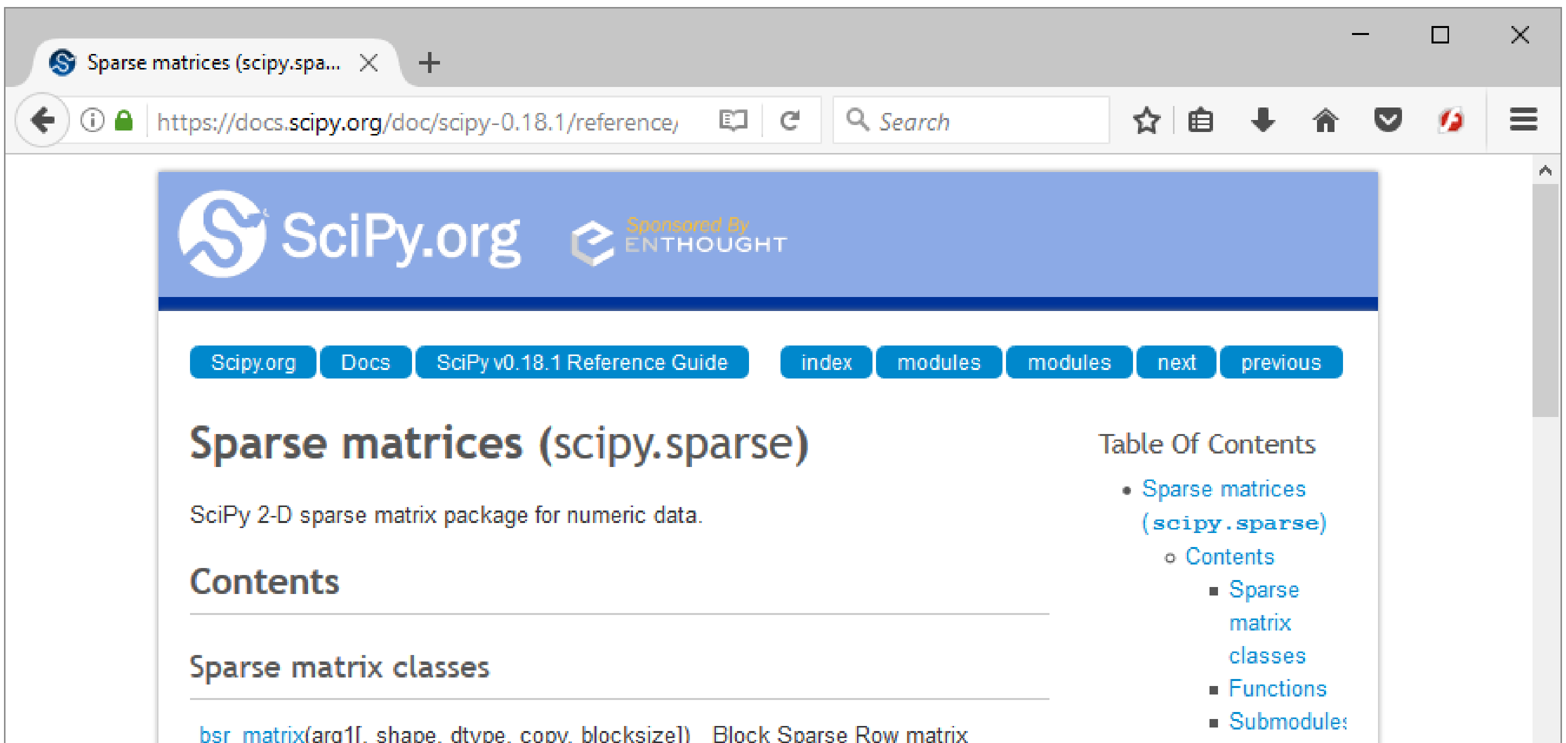


Induced by the connectivity of the triangle mesh.

Iteration of CG has local effect
⇒ Precondition!

For 6.838

- No need to implement a linear solver
- If a matrix is sparse, your code should store it as a sparse matrix!



Rough Plan

- Linear problems
- **Unconstrained optimization**
- Equality-constrained optimization
- Variational problems

Optimization Terminology

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

Objective ("Energy Function")

Optimization Terminology

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

Equality Constraints

Optimization Terminology

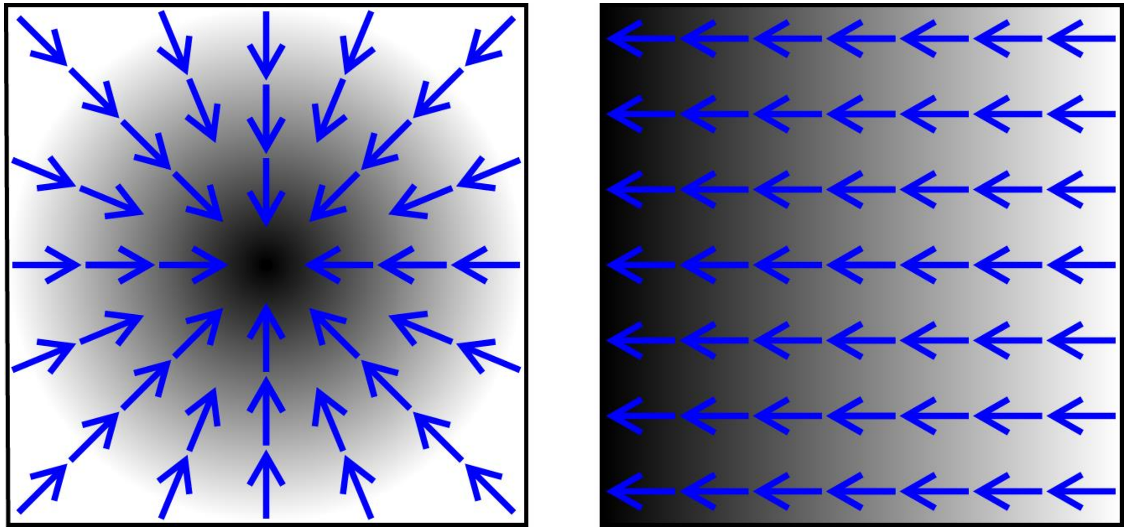
$$\min_{x \in \mathbb{R}^n} f(x)$$
$$\text{s.t. } g(x) = 0$$
$$h(x) \geq 0$$

Inequality Constraints

Notions from Calculus

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

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



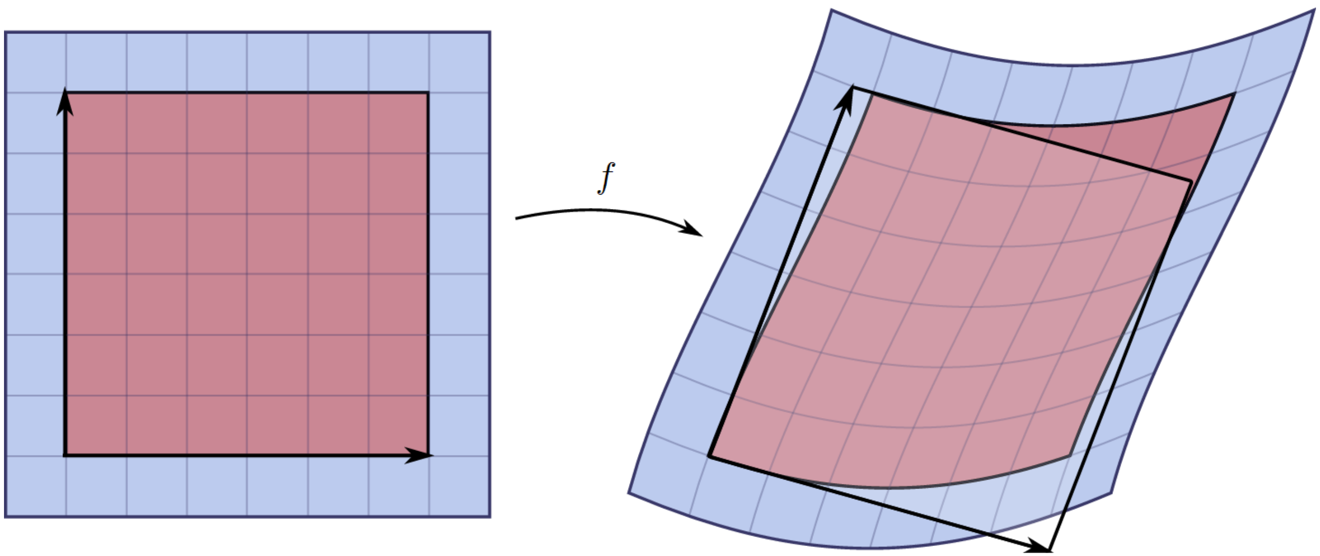
https://en.wikipedia.org/?title=Gradient

Gradient

Notions from Calculus

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

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

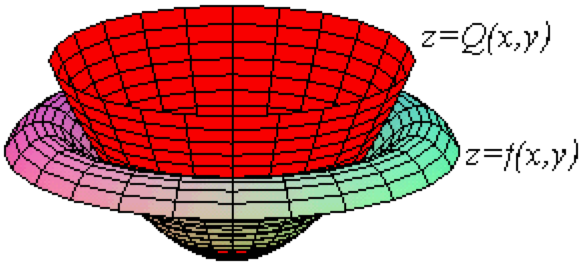


https://en.wikipedia.org/wiki/Jacobian_matrix_and_determinant

Jacobian

Notions from Calculus

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



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

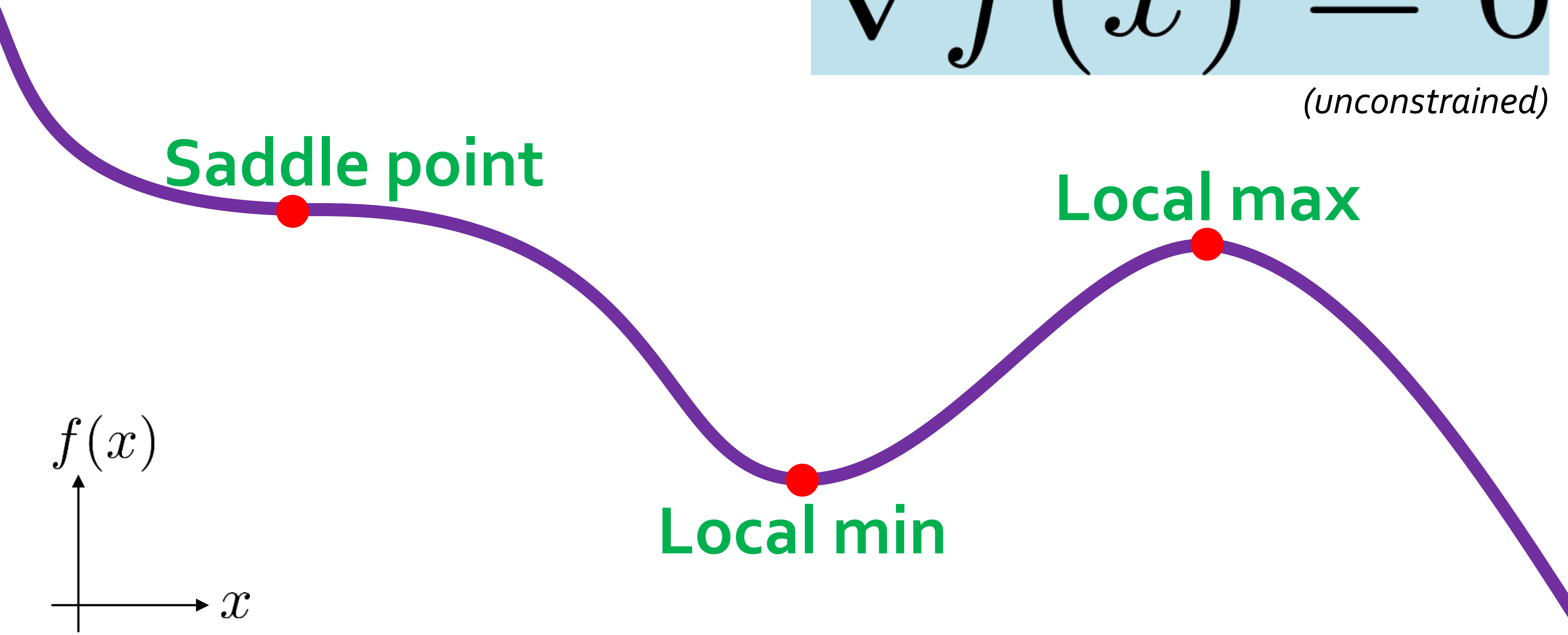
http://math.etsu.edu/multicalc/prealpha/Chap2/Chap2-5/10-3a-t3.gif

Hessian

Optimization to Root-Finding

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

(unconstrained)



Critical point

Encapsulates Many Problems

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

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

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

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

Q:

How effective are generic optimization tools?

Q:

How effective are generic optimization tools?

Not very!

Generic Advice

Try the

simplest solver first.

Quadratic with Linear Equality

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

(assume A is symmetric and positive definite)

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

Useful Document

The Matrix Cookbook

Petersen and Pedersen

http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3274/pdf/imm3274.pdf

Special Case: Least-Squares

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

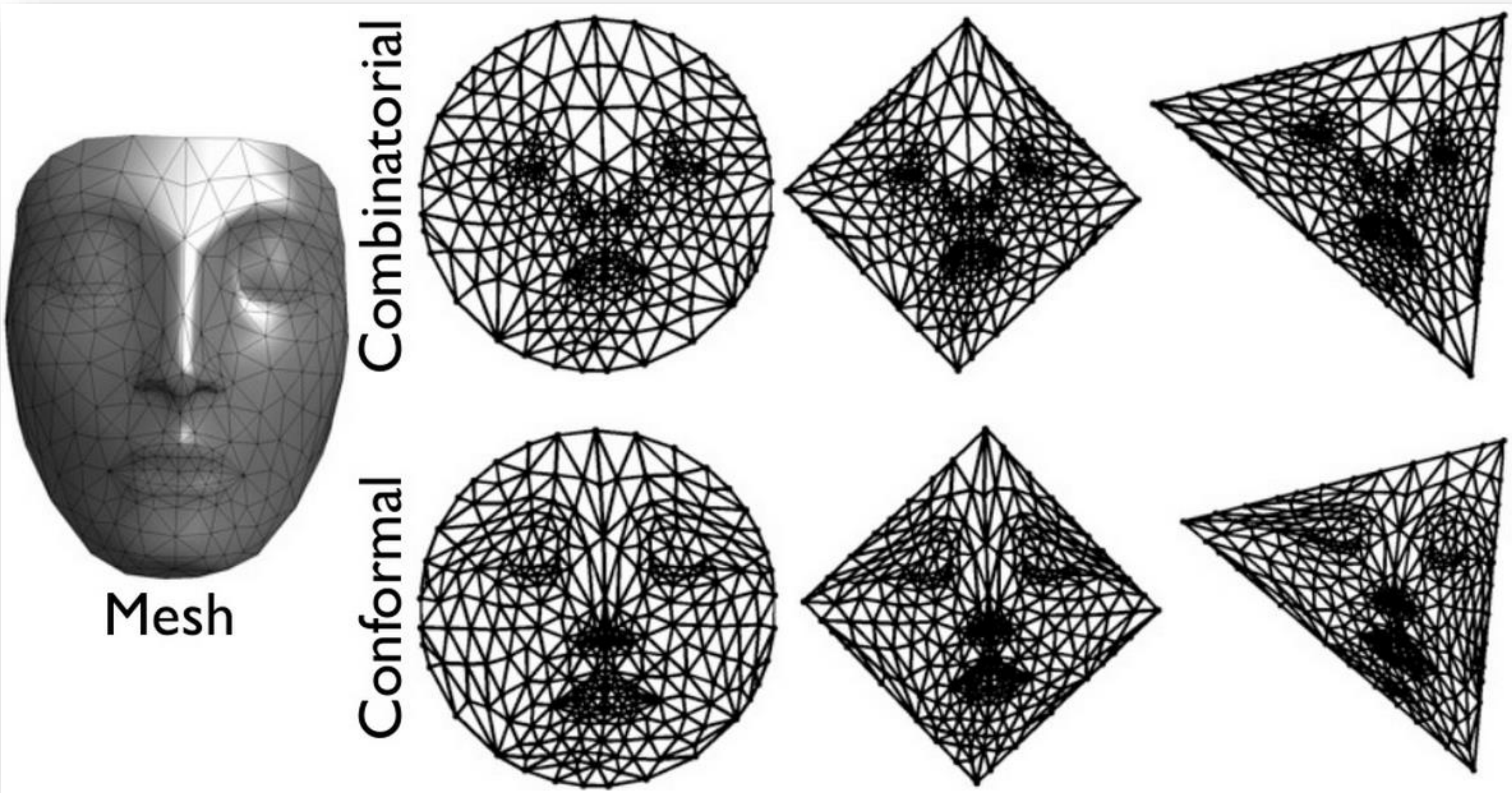
$$\to \min_x \frac{1}{2} x^\top A^\top A x - b^\top A x + \|b\|_2^2$$

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

Normal equations

(better solvers for this case!)

Example: Mesh Embedding



G. Peyré, mesh processing course slides

Linear Solve for Embedding

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

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

Assumption: w **symmetric.**

Returning to Parameterization

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

What if $V_0 = \{\}$?

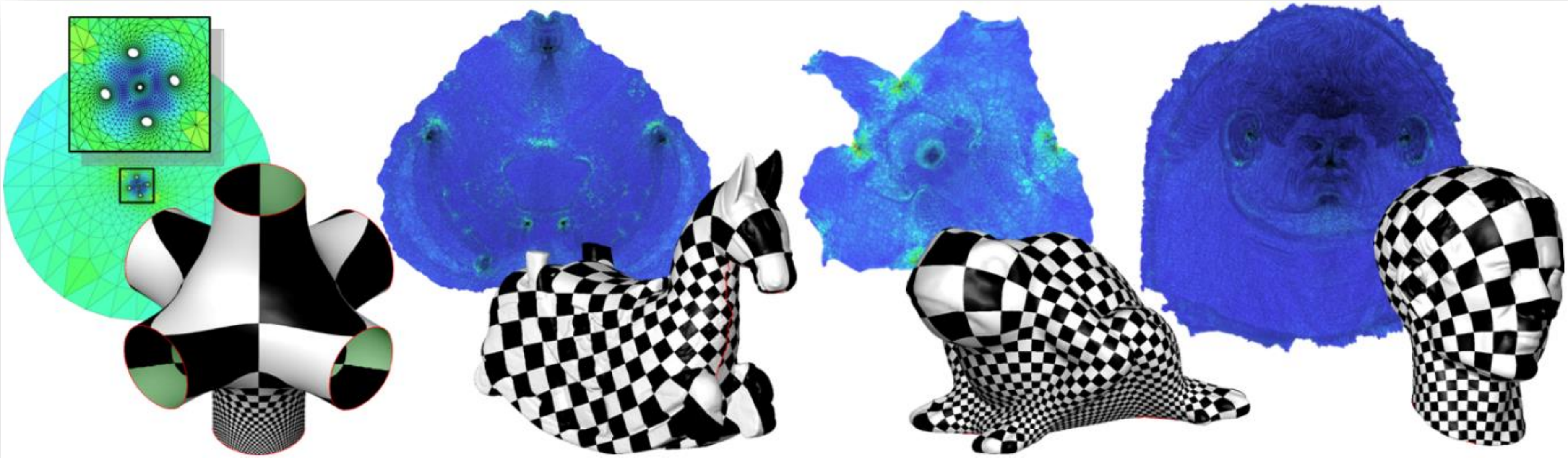
Nontriviality Constraint

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

Prevents trivial solution $x \equiv 0$.

Extract the smallest eigenvalue.

Back to Parameterization



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

$$\min_{u} \quad u^\top L_C u \qquad \longleftrightarrow \qquad L_c u = \lambda B u$$

$$u^\top B e = 0 \quad \leftarrow \textit{Easy fix}$$

$$u^\top B u = 1$$

Basic Idea of Eigenalgorithms

$$
\begin{aligned}
A\vec{v} &= c_1 A\vec{x}_1 + \cdots + c_n A\vec{x}_n \\
&= c_1 \lambda_1 \vec{x}_1 + \cdots + c_n \lambda_n \vec{x}_n \text{ since } A\vec{x}_i = \lambda_i \vec{x}_i \\
&= \lambda_1 \left(c_1 \vec{x}_1 + \frac{\lambda_2}{\lambda_1} c_2 \vec{x}_2 + \cdots + \frac{\lambda_n}{\lambda_1} c_n \vec{x}_n \right) \\
A^2\vec{v} &= \lambda_1^2 \left(c_1 \vec{x}_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^2 c_2 \vec{x}_2 + \cdots + \left(\frac{\lambda_n}{\lambda_1}\right)^2 c_n \vec{x}_n \right) \\
&\vdots \\
A^k\vec{v} &= \lambda_1^k \left(c_1 \vec{x}_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^k c_2 \vec{x}_2 + \cdots + \left(\frac{\lambda_n}{\lambda_1}\right)^k c_n \vec{x}_n \right).
\end{aligned}
$$

Combining Tools So Far

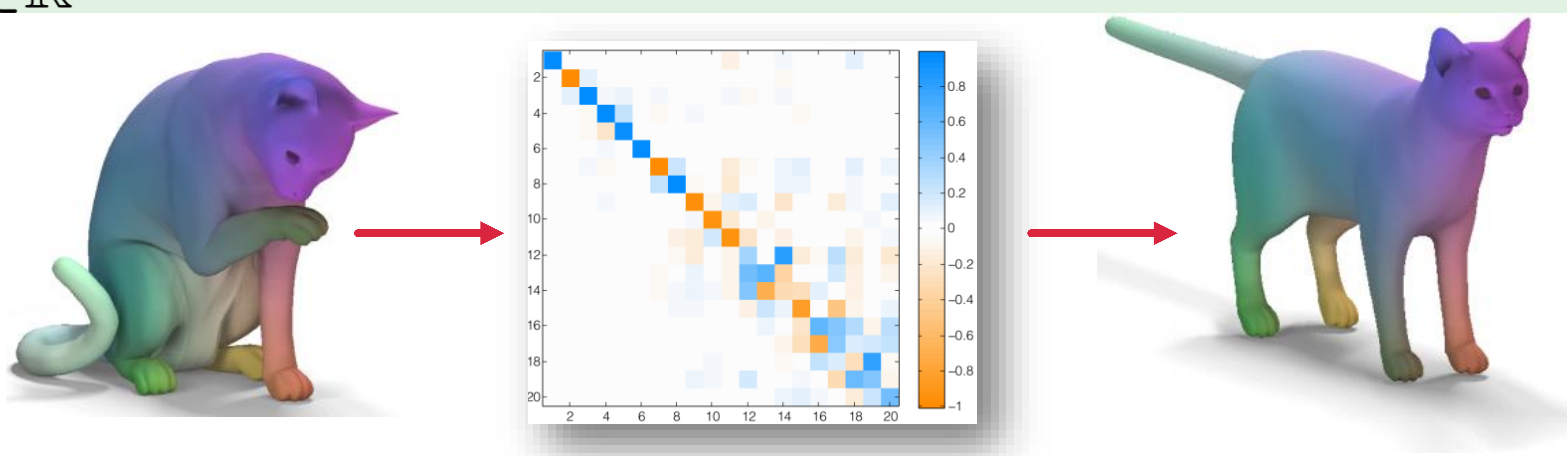
Roughly:

1. **Extract Laplace-Beltrami eigenfunctions:**

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

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

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



Ovsjanikov et al. "Functional Maps." SIGGRAPH 2012.

Rough Plan

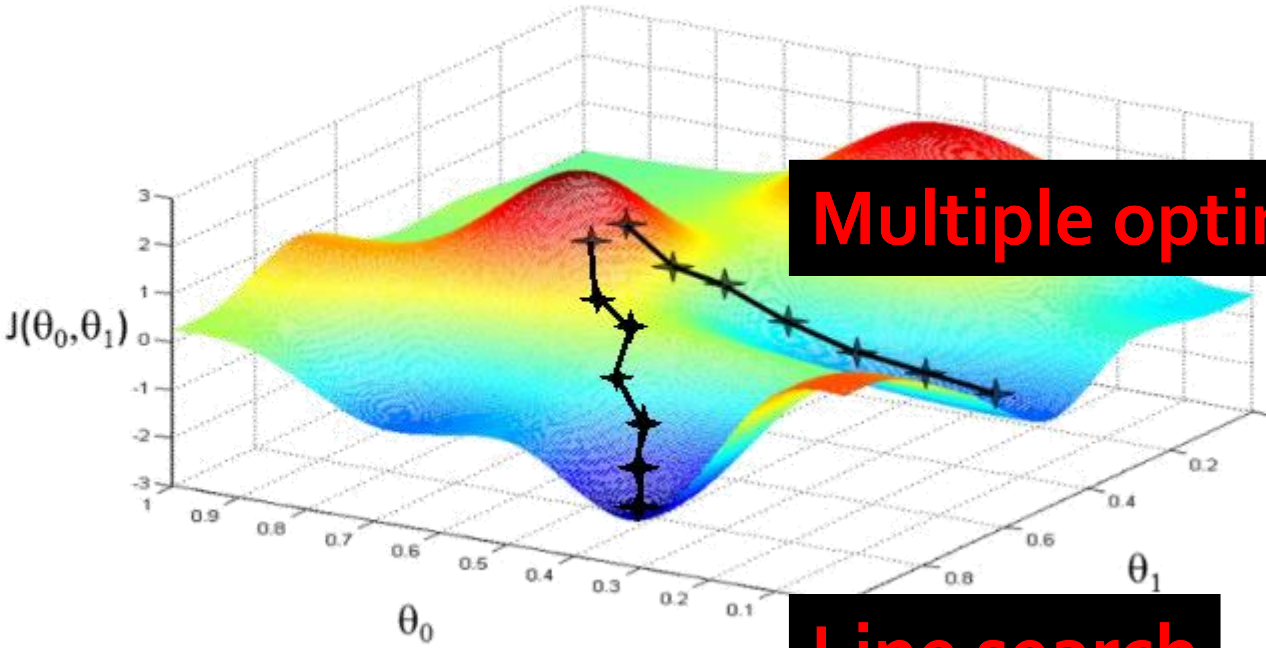
- Linear problems
- **Unconstrained optimization**
- Equality-constrained optimization
- Variational problems

Unconstrained Optimization

$$\min_{x} f(x)$$

↑ **Unstructured.**

Basic Algorithms



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

Gradient descent

Basic Algorithms

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

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

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

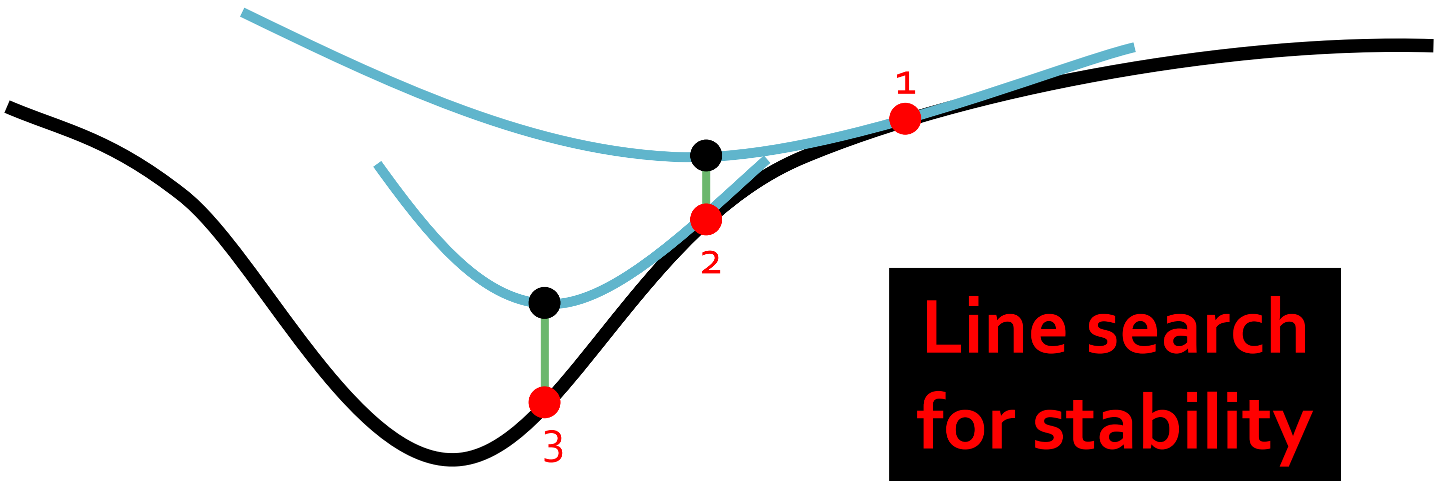
Quadratic convergence on convex problems!

(Nesterov 1983)

Accelerated gradient descent

Basic Algorithms

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



Line search for stability

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

Example: Shape Interpolation

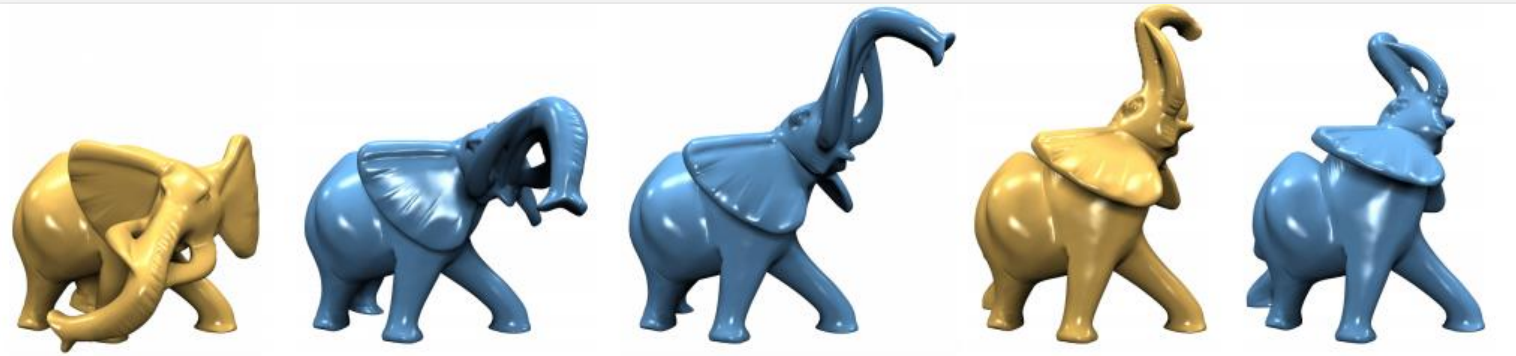


Figure 5: *Interpolation and extrapolation of the yellow example poses. The blending weights are* 0, 0.35, 0.65, 1.0, *and* 1.25.

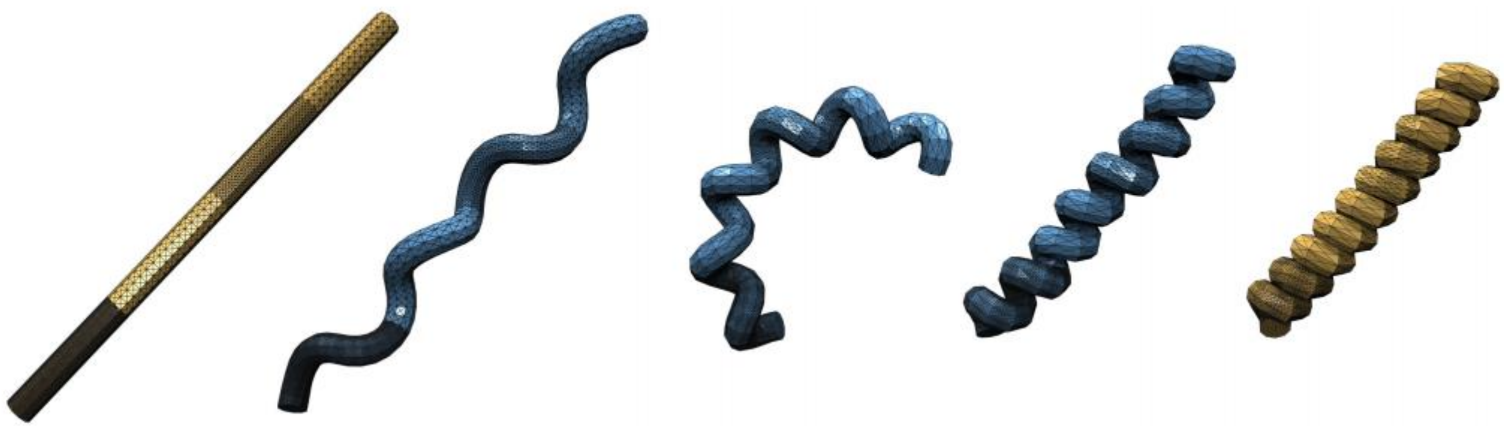


Figure 6: *Interpolation of an adaptively meshed and strongly twisted helix with blending weights* 0, 0.25, 0.5, 0.75, 1.0.

Fröhlich and Botsch. "Example-Driven Deformations Based on Discrete Shells." CGF 2011.

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.**

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

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

Sum of squares: Gauss-Newton

Software

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

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

Try several!

Some Tricks

Lots of small elements: $\|x\|_2^2 = \sum_i x_i^2$

Lots of zeros: $\|x\|_1 = \sum_i |x_i|$

Uniform norm: $\|x\|_\infty = \max_i |x_i|$

Low rank: $\|X\|_* = \sum_i \sigma_i$

Mostly zero columns: $\|X\|_{2,1} = \sum_j \sqrt{\sum_i x_{ij}^2}$

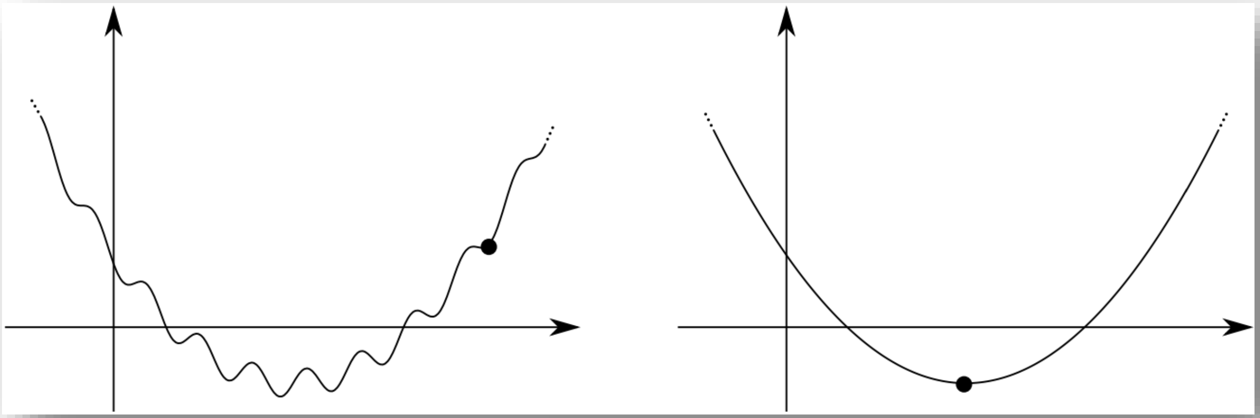
Smooth: $\int \|\nabla f\|_2^2$

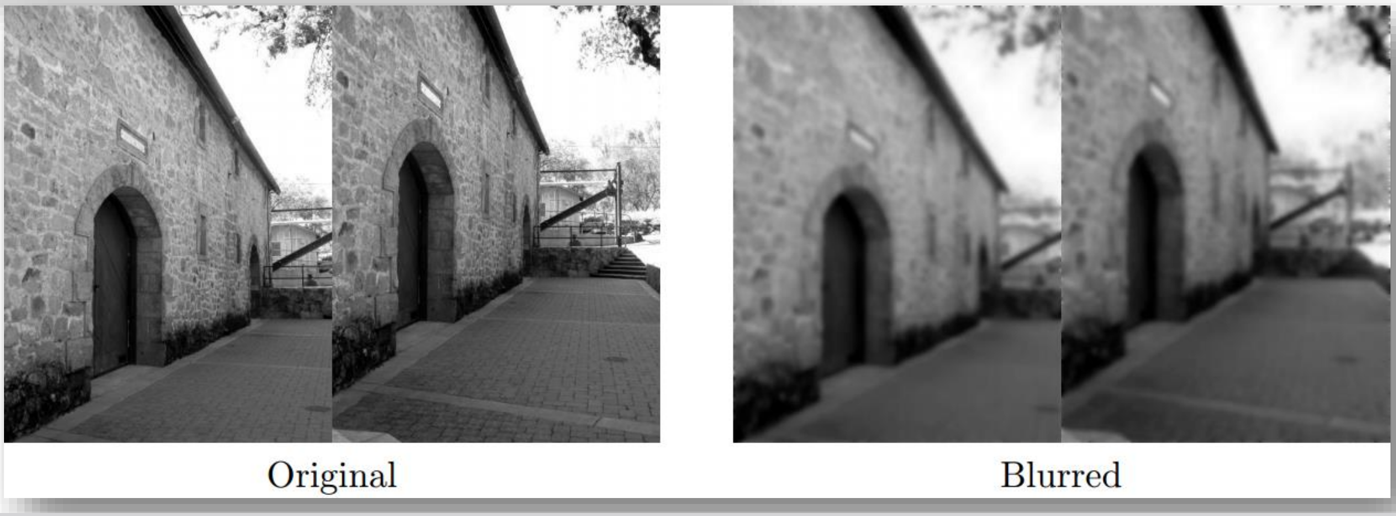
Piecewise constant: $\int \|\nabla f\|_2$

???: Early stopping

Regularization

Some Tricks



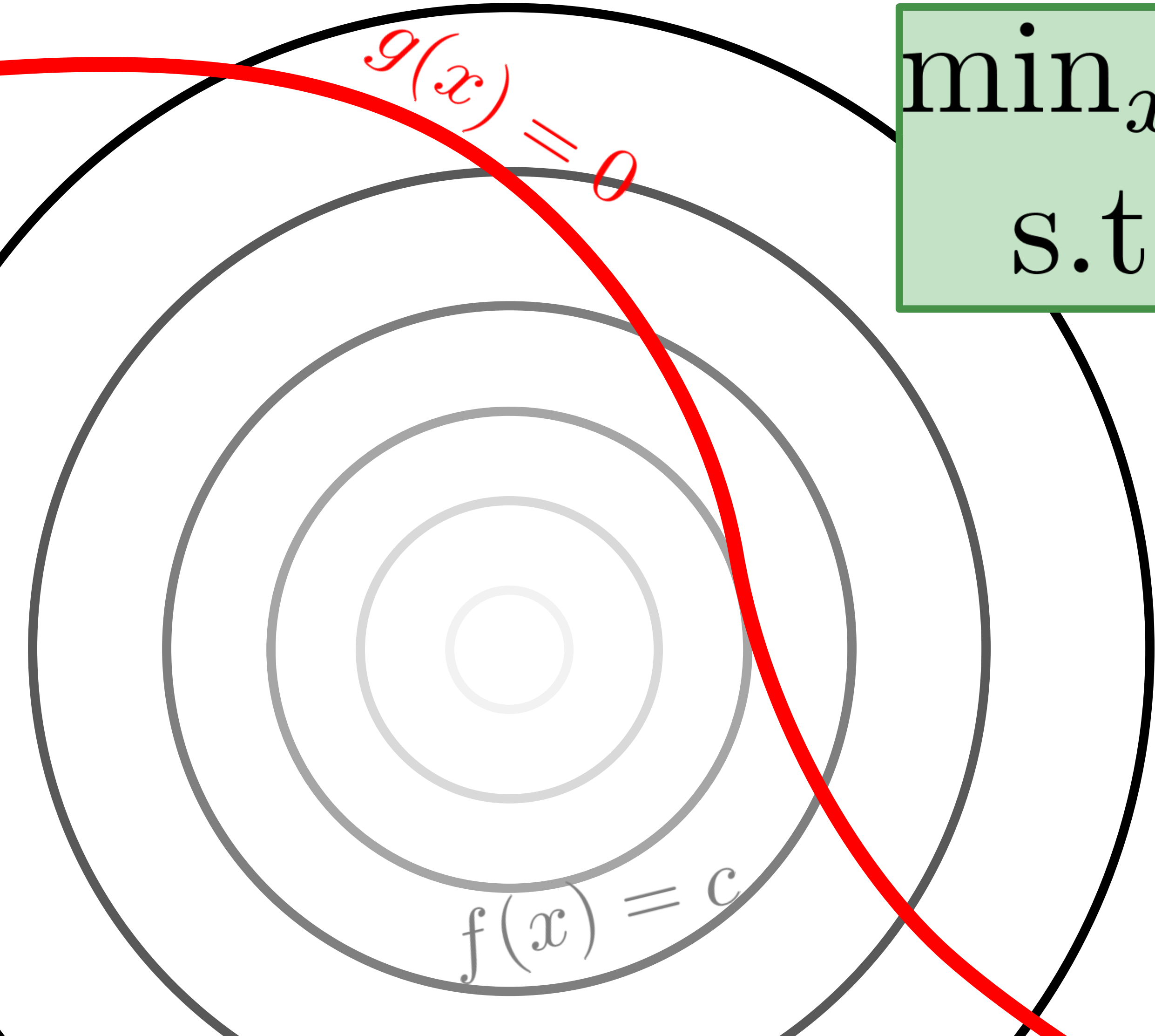


Multiscale/graduated optimization

Rough Plan

- Linear problems
- Unconstrained optimization
- **Equality-constrained optimization**
- Variational problems

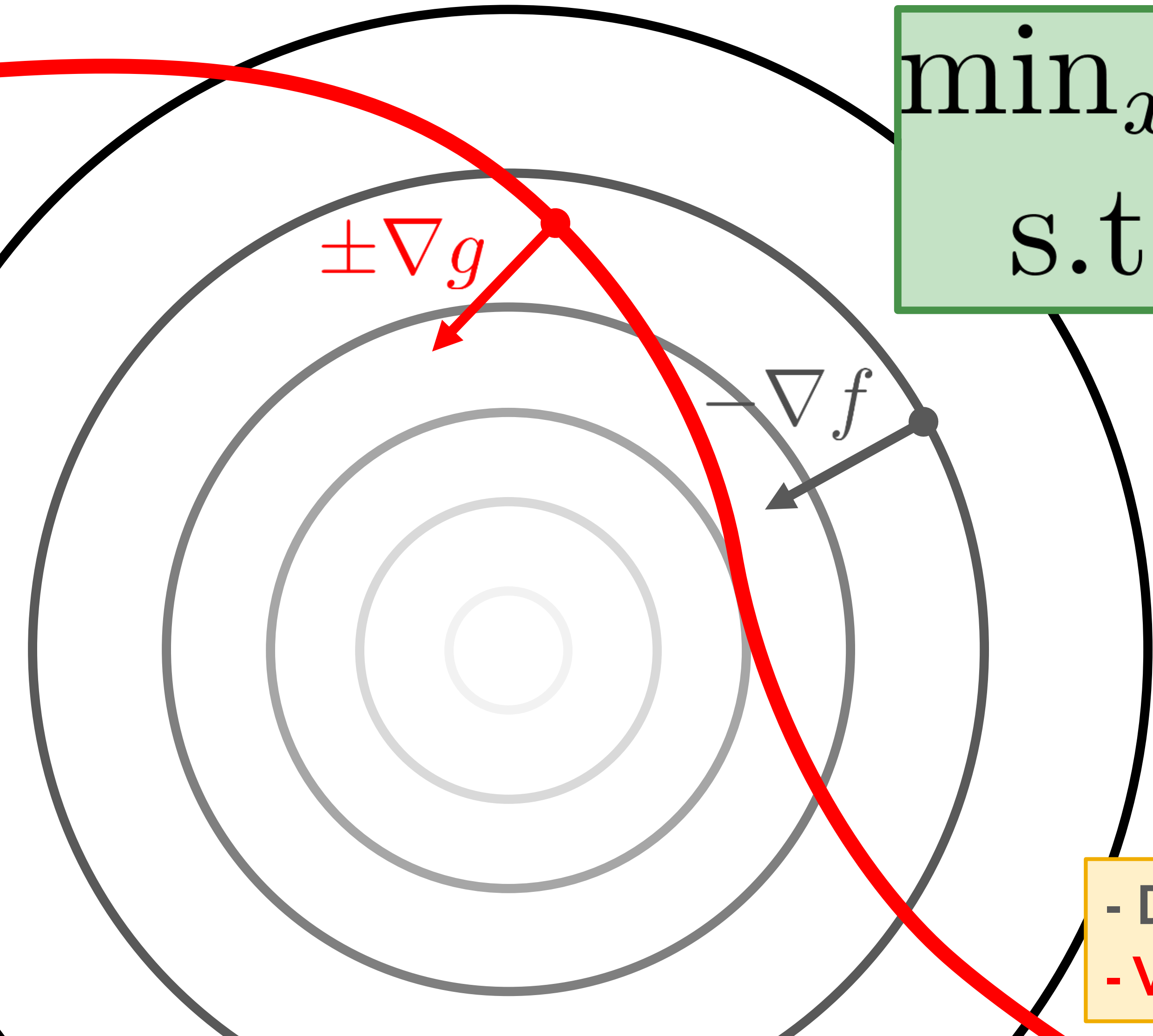
Lagrange Multipliers: Idea



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

Lagrange Multipliers: Idea

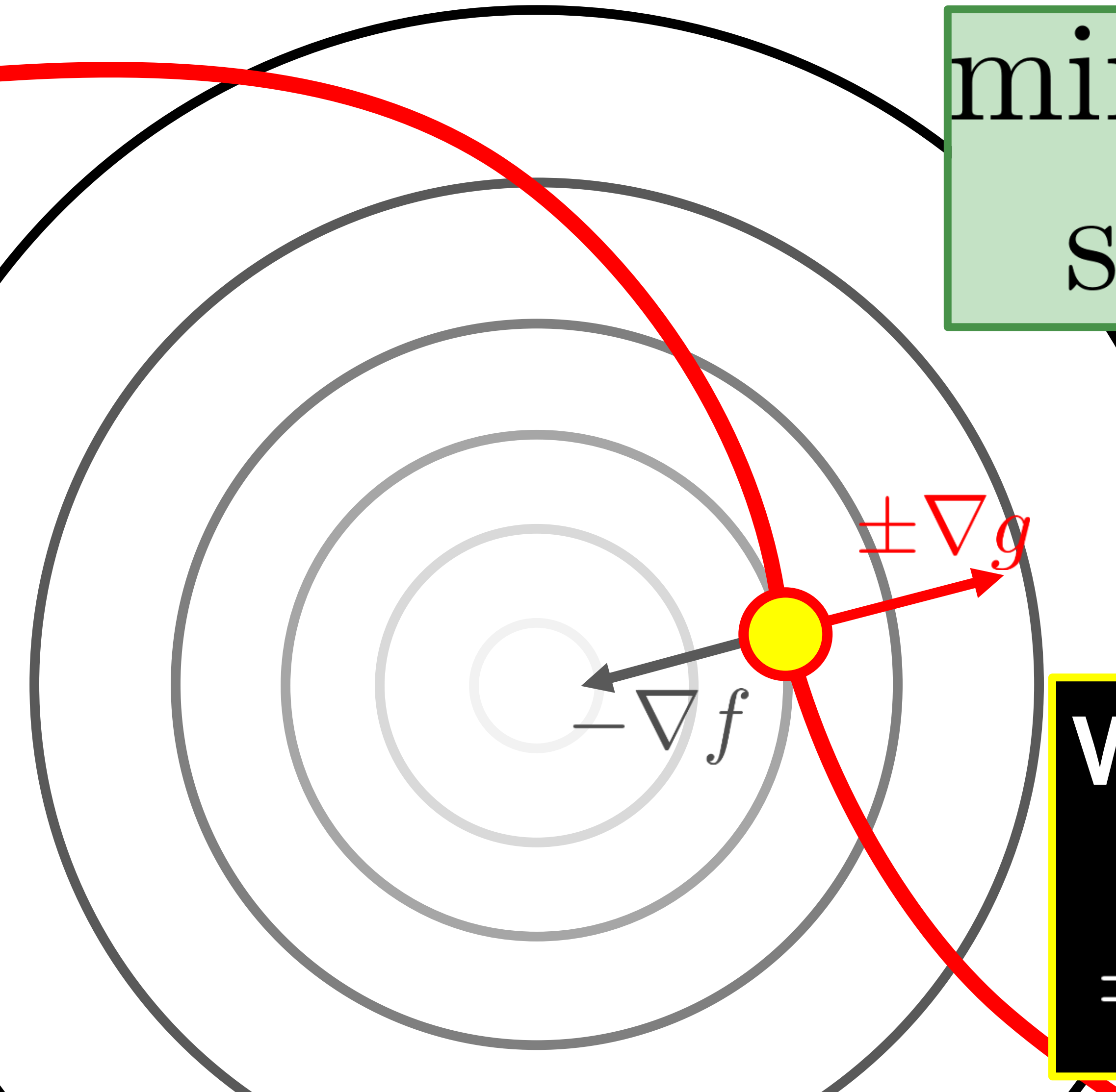
$$\min_x \quad f(x)$$
$$\text{s.t.} \quad g(x) = 0$$



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

Lagrange Multipliers: Idea

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



Want:

$$\nabla f \| \nabla g$$

$$\Longrightarrow \quad \nabla f = \lambda \nabla g$$

Example: Symmetric Eigenvectors

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

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

$$\implies Ax = \lambda x$$

Use of Lagrange Multipliers

Turns constrained optimization into

unconstrained root-finding.

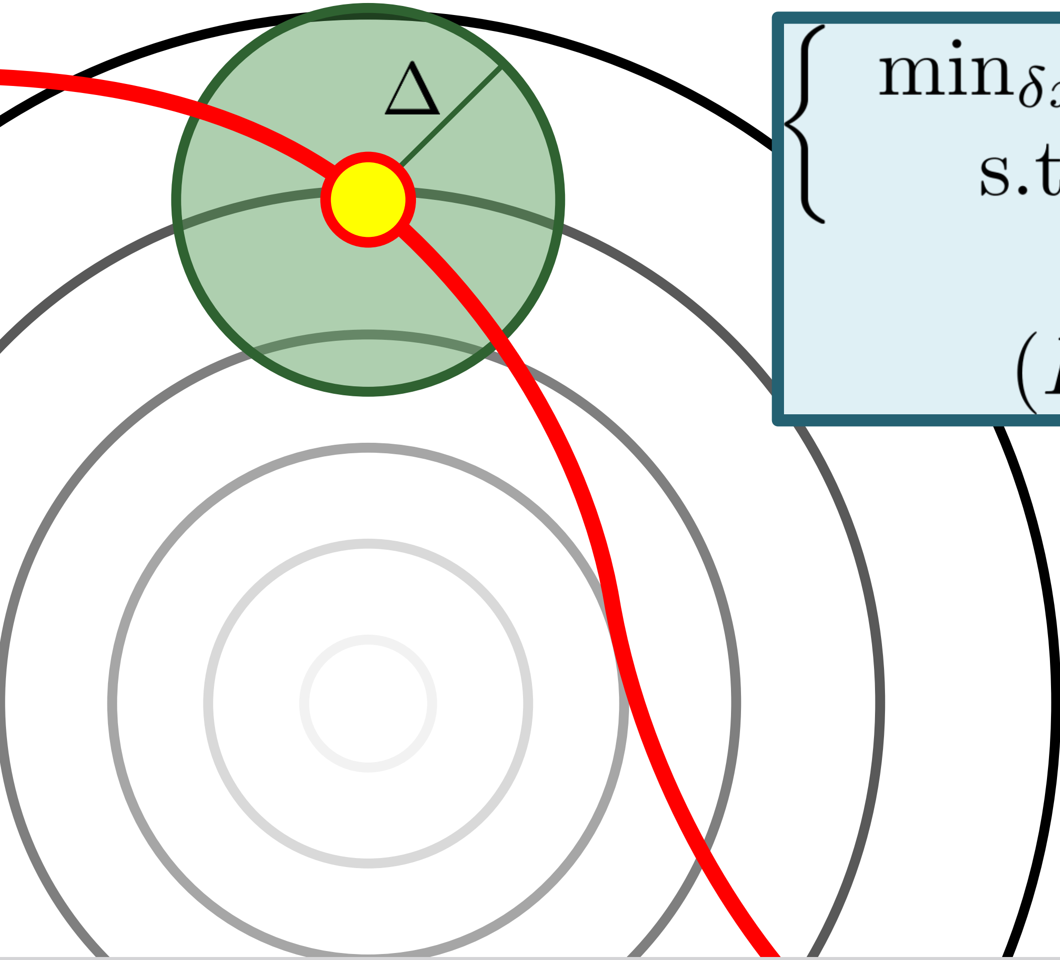
$$\nabla f(x) = \lambda \nabla g(x)$$

$$g(x) = 0$$

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. $f(x) + \rho|g(x)|$

Trust Region Methods



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

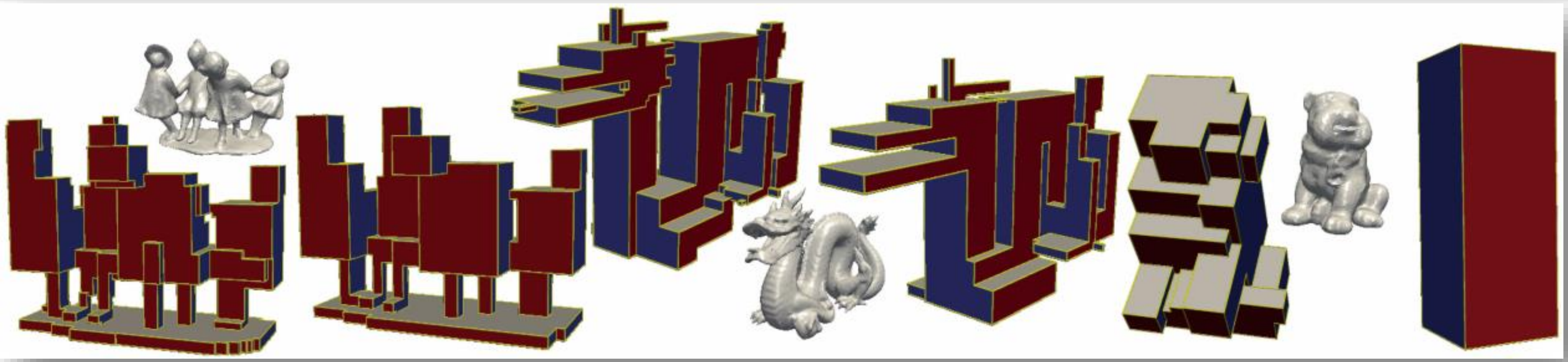
$$\downarrow$$

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

Fix (or adjust)
damping parameter
$\boldsymbol{\lambda > 0}$.

Example: Levenberg-Marquardt

Example: Polycube Maps



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

Align with coordinate axes

$$\min_X \sum_{b_i} A(b_i;X)\|n(b_i;X)\|_1$$
$$\text{s.t.} \quad \sum_{b_i} A(b_i;X) = \sum_{b_i} A(b_i;X_0)$$

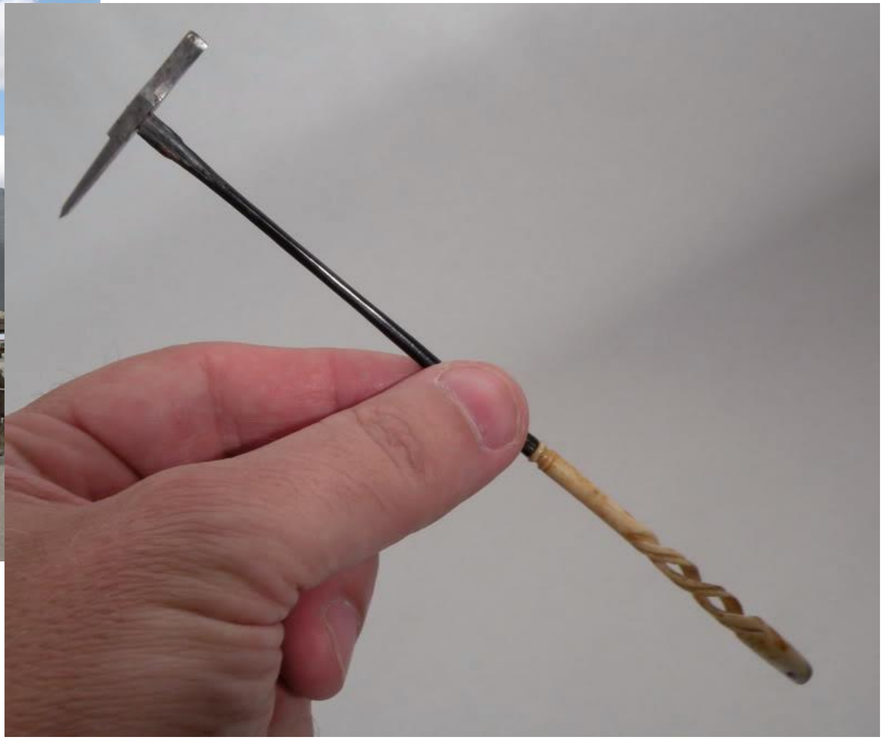
Preserve area

***Note:* Final method includes more terms!**

Aside:

Convex Optimization Tools

versus

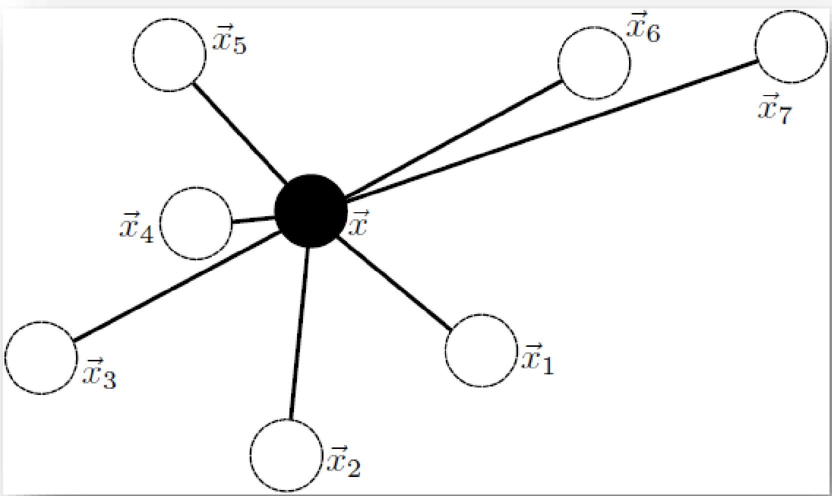


Sometimes work for non-convex problems…

Try lightweight options

Iteratively Reweighted Least Squares

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



"Geometric median"

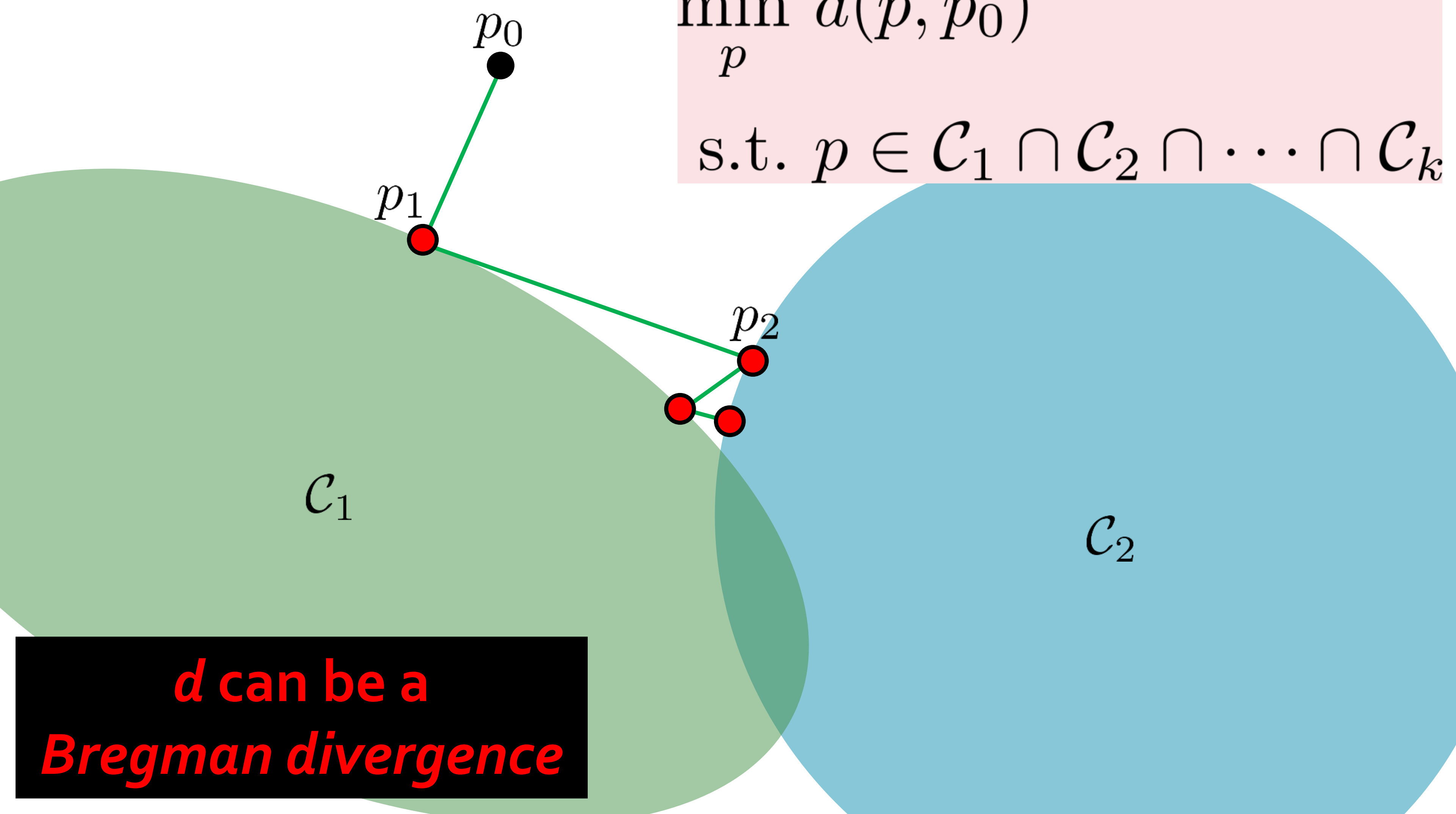
$$\min_x \sum_i \|x - p_i\|_2 \implies \begin{cases} x & \leftarrow \min_x \sum_i y_i \|x - p_i\|_2^2 \\ y_i & \leftarrow \|x - p_i\|_2^{-1} \end{cases}$$

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

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

FISTA combines with Nesterov descent!

https://blogs.princeton.edu/imabandit/2013/04/11/orf523-ista-and-fista/

Augmented Lagrangians

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

$$\downarrow$$

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

Does nothing when constraint is satisfied

Add constraint to objective

Alternating Direction Method of Multipliers (ADMM)

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

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

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

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

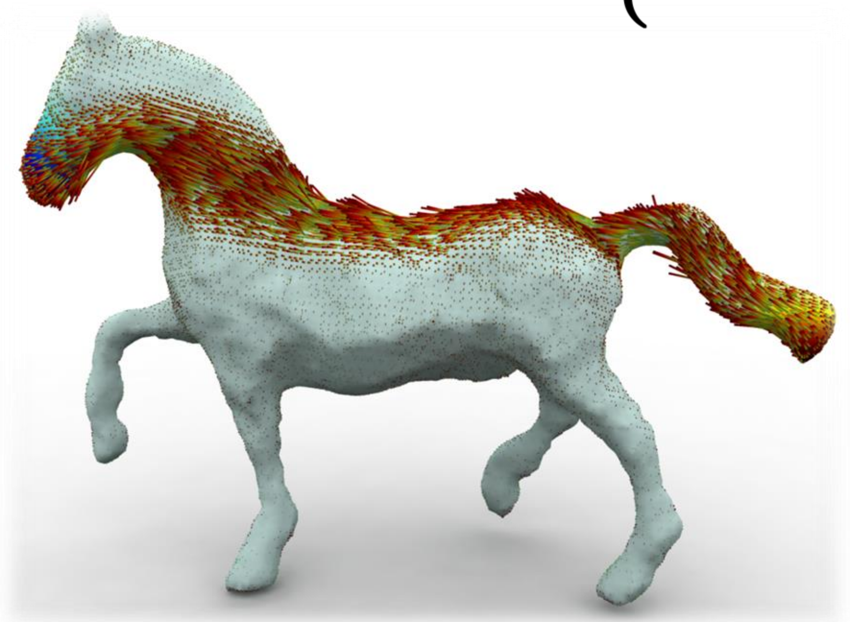
$$\lambda \leftarrow \lambda + \rho(Ax + Bz - c)$$

https://web.stanford.edu/~boyd/papers/pdf/admm_slides.pdf

The Art of ADMM "Splitting"

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

Augmented part



Takes some practice!
Example of "proximal" algorithm.

Solomon et al. "Earth Mover's Distances on Discrete Surfaces." SIGGRAPH 2014.

Want two *easy* subproblems

Frank-Wolfe

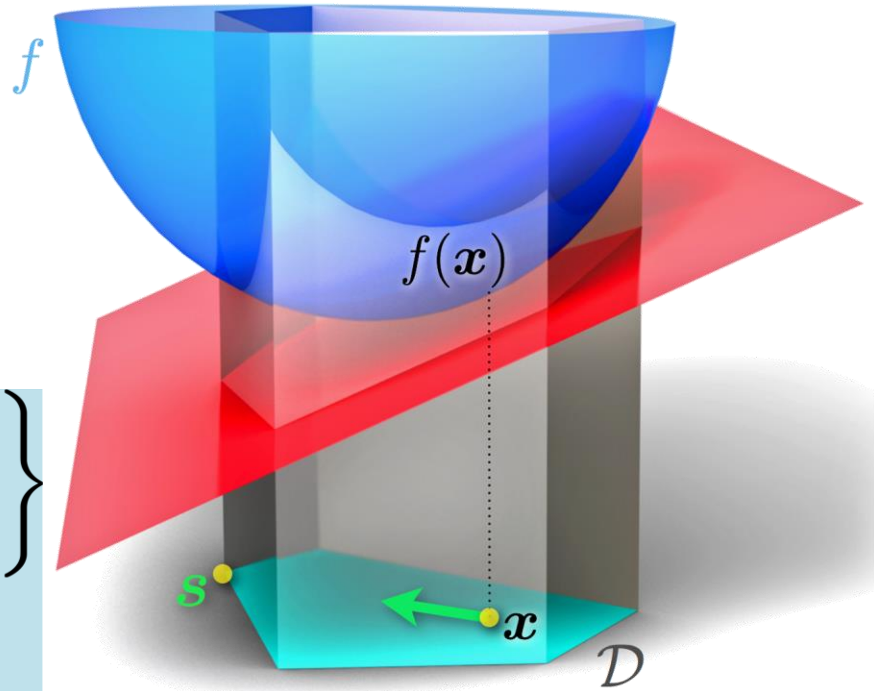
</aside>

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

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

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

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



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

Linearize objective, preserve constraints

Rough Plan

- Linear problems
- Unconstrained optimization
- Equality-constrained optimization
- **Variational problems**

Variational Calculus: Big Idea

Sometimes your unknowns

are not numbers!

Can we use calculus to optimize anyway?

On the Board

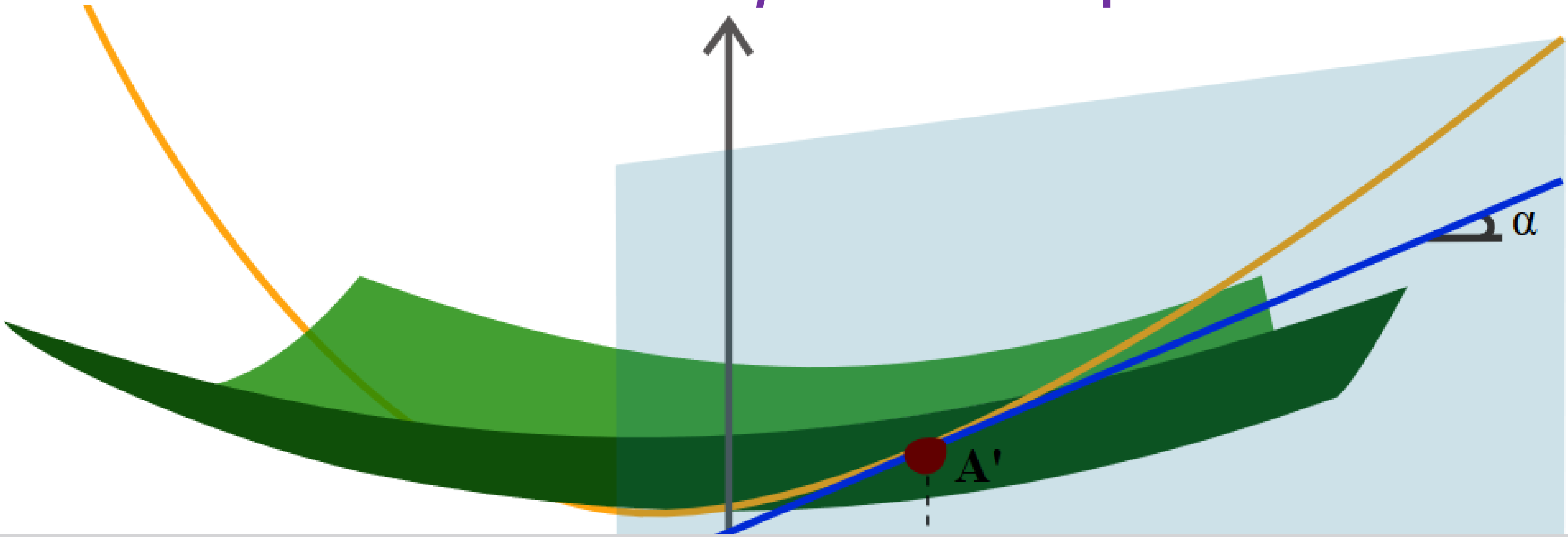
$$\min_f \int_\Omega \|\vec{v}(x) - \nabla f(x)\|_2^2 \, d\vec{x}$$

$$\min_{\int_\Omega f(x)^2 \, d\vec{x} = 1} \int_\Omega \|\nabla f(x)\|_2^2 \, d\vec{x}$$

Gâteaux Derivative

$$d\mathcal{F}[u;\psi] := \frac{d}{dh}\mathcal{F}[u+h\psi]|_{h=0}$$

Vanishes for all ψ at a critical point!



Analog of derivative at *u* in *ψ* direction

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

Numerical Tools
for Geometry

Justin Solomon
MIT, Spring 2017

MIT EECS