CS457 Geometric Computing

3A - Shape Preservation

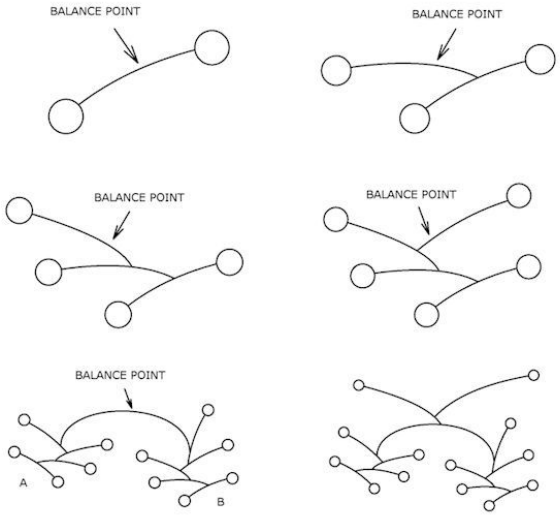
Mark Pauly

Geometric Computing Laboratory - EPFL

Challenge I - Make it stand!

- Questions:
  - Given some (digital) geometric object, how can we determine if it stands? ✓
  - If it does not stand, how can we modify it, so that it does?
- More fundamentally:
  - What does it mean for an object to *stand*? ✓
  - What is a *geometric object*? ✓
  - What does it mean to *modify* a shape?
  - How do we find the *best modification*?
- Make *it* stand: The initial shape, the *it*, has meaning.
  - How do we preserve the initial shape as best as possible?

Aside: Mobile



Wikipedia

Lausanne Mobile Art

Aside: GCM Research

MOCCA: Modeling and Optimizing Cone-joints for Complex Assemblies, SIGGRAPH 2021

Recap: Barycentric Coordinates

- What is the barycentric equation for the line passing through point C and the centroid of triangle ABC?

A: $\gamma = 1 - \beta$

B: $\alpha = \beta$

C: $\alpha + \gamma = 1$

D: $\gamma = \beta/3$

Shearing a Mesh

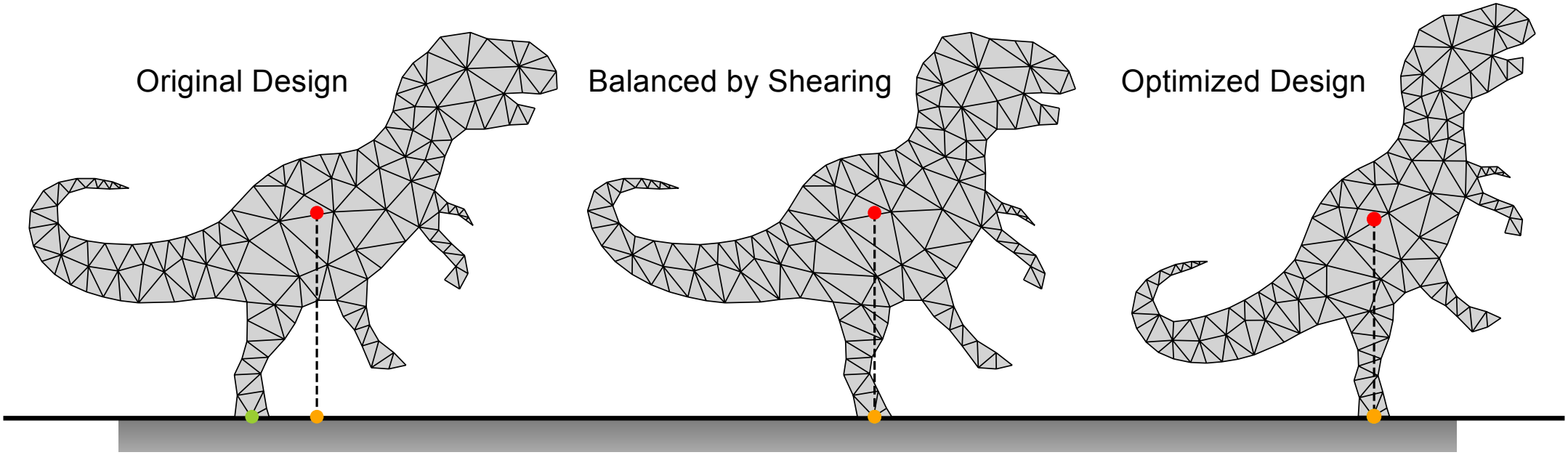
- The centroid is given as

$$\boldsymbol{x}_{\text{cm}} = \frac{1}{\Omega} \sum_{f \in F} A_f \boldsymbol{x}_f$$

  - where A_f, $\boldsymbol{x}_f$ are the area resp. centroid of triangle f and $\Omega = \sum_{f \in F} A_f$ is the total area of the mesh.
- Area is preserved under shearing!
  - When applying a shear, the mesh centroid is still a linear function of the shear coefficient ν.
- We can easily solve for the specific ν to achieve a given target x-location of the centroid!
  - Homework 1.1

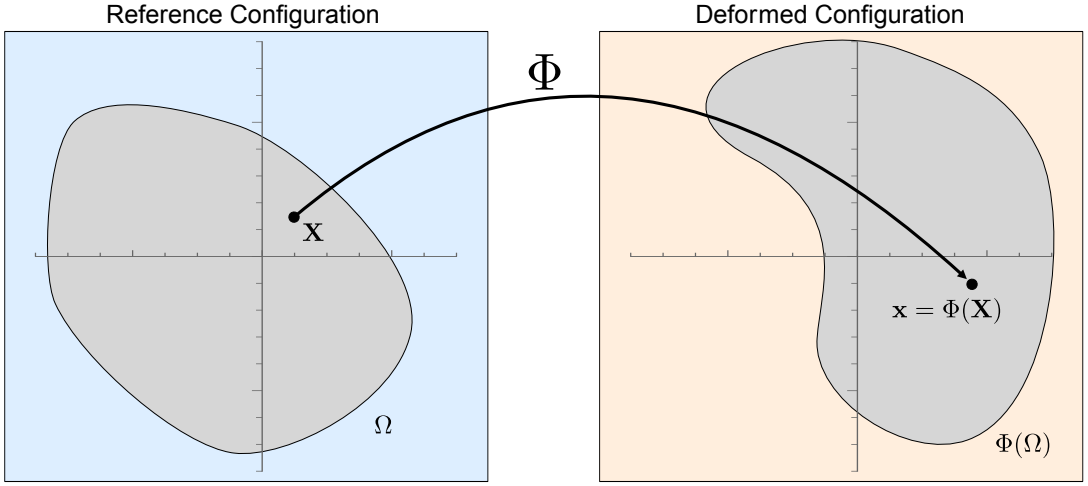
Shape Preservation

- Shear can introduce “unnatural” deformations.
- Make *it* stand: The initial shape, the *it*, has meaning.
  - How do we preserve the initial shape as best as possible?



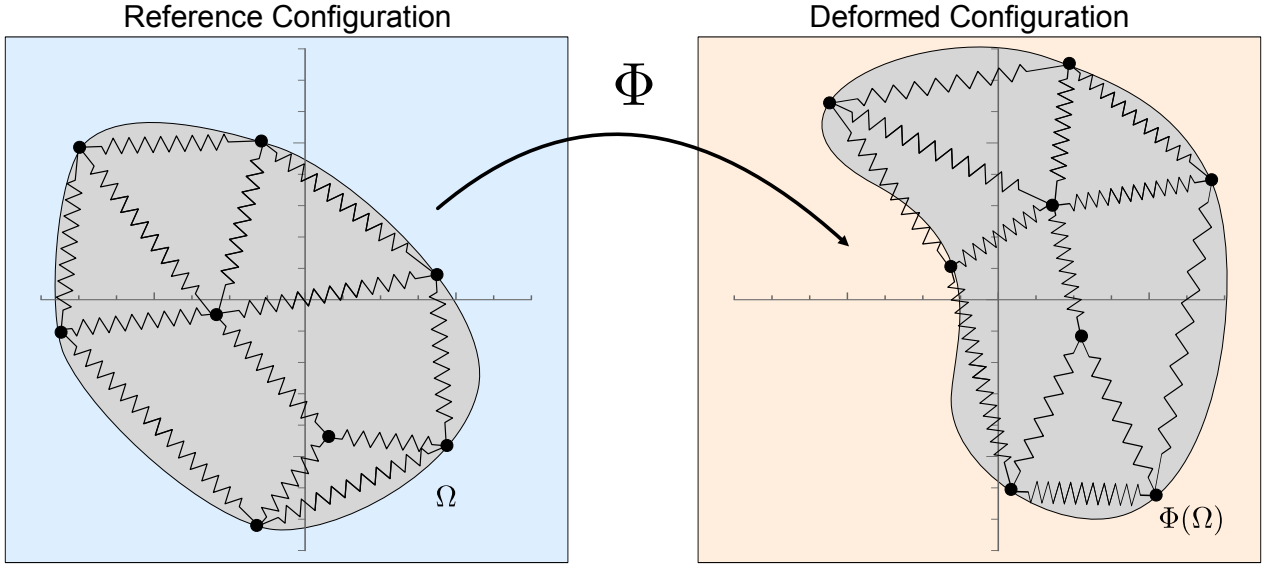
Shape Preservation

- We want to modify our input geometry as little as possible to make it stand.
- How can we make “as little as possible” more formal?
- Idea: Treat the object as an elastic body and minimize some elastic deformation energy.



- We use a simple shape preservation model based on springs.
  - We will study more sophisticated models are based on continuum mechanics later

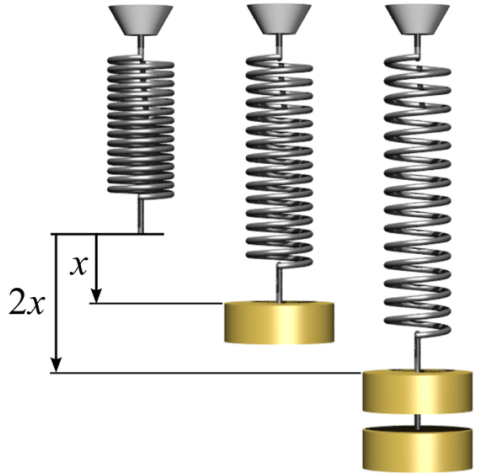
Spring Networks



- Rudimentary spring-based deformable object model:
  - Sample a set of points $\mathbf{X}_i$ and connect them with springs.
    - In our case, the samples $\mathbf{X}_i$ are the initial mesh vertex positions and each mesh edge becomes a spring.
  - Now our deformed configuration is described by a deformation function Φ as $\mathbf{x}_i = \Phi(\mathbf{X}_i)$.

Linear Springs (Hooke's Law)

- Hooke's Law (1676): for small extensions, the force required to stretch a spring is proportional to the amount of stretch.

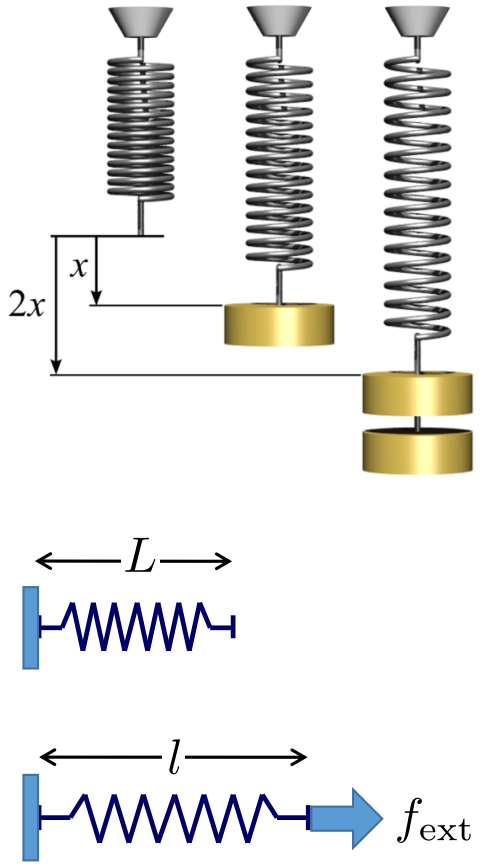


Linear Springs (Hooke's Law)

- Hooke's Law (1676): for small extensions, the force required to stretch a spring is proportional to the amount of stretch.

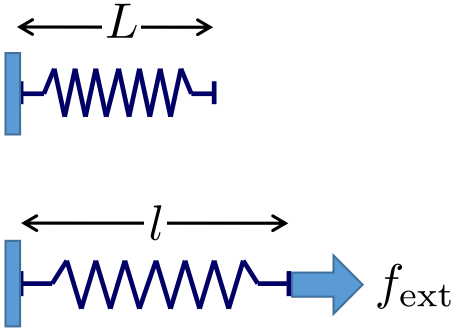
$$\boxed{f_{\text{ext}} = k(l - L)}$$

Variables	Meaning
L	rest length
l	deformed length
f_{ext}	applied force
k	spring stiffness coefficient



Linear Springs (Hooke's Law) Energy

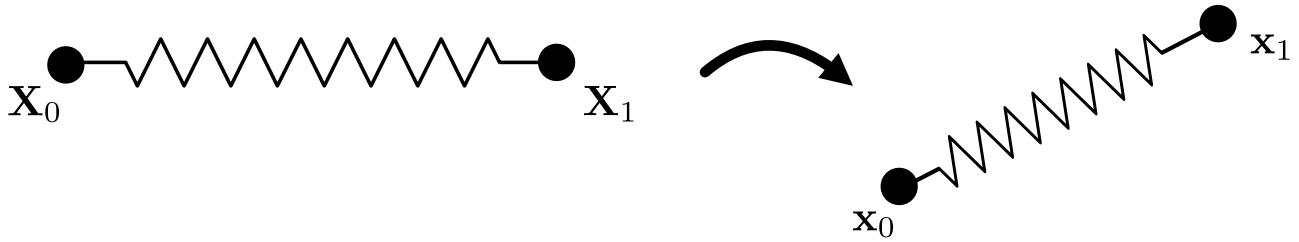
- How do we model this spring with an elastic energy term?
- Energy stored in spring = work done by external force (neglecting heat)



$$\begin{aligned} E_{\text{spring}}(L, l) &= \int_L^l f_{\text{ext}}(x)\mathrm{d}x \\ &= \int_L^l k(x - L)\mathrm{d}x = \frac{1}{2}k(l - L)^2. \end{aligned}$$

Linear Springs (Hooke's Law) in nD

- Now let's express a spring's energy in terms of undeformed/deformed points in nD.

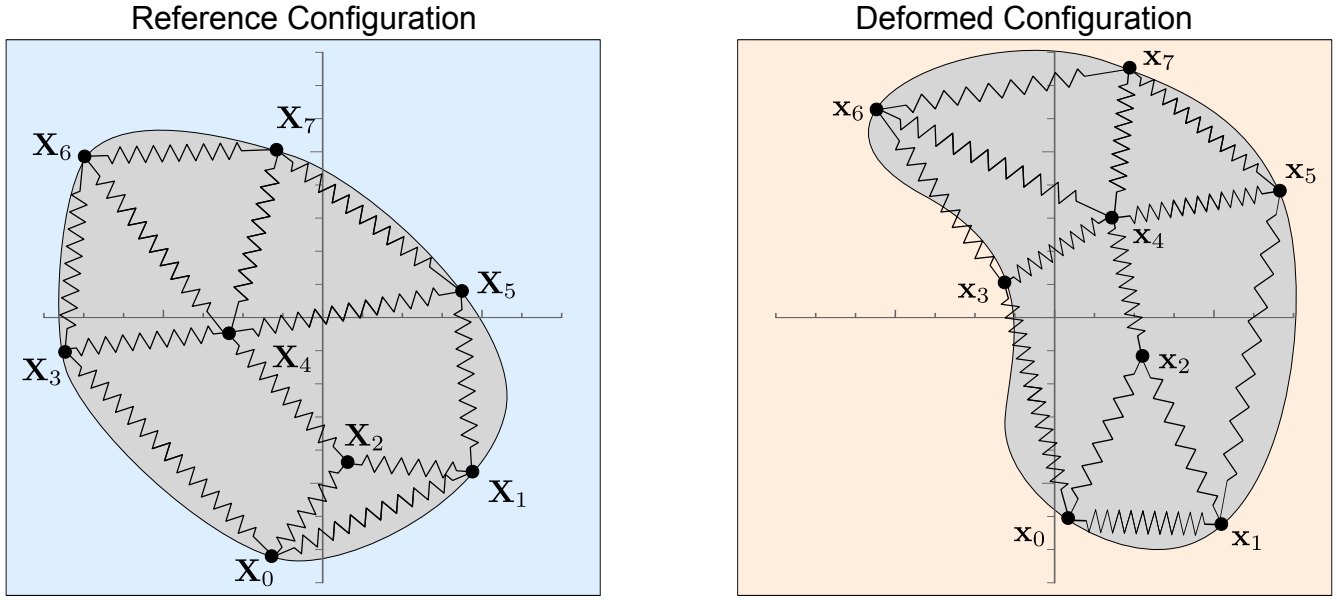


$$L = \|\mathbf{X}_1 - \mathbf{X}_0\|, \quad l = \|\mathbf{x}_1 - \mathbf{x}_0\|$$

$$E_{\text{spring}} = \frac{1}{2} k \left(\|\mathbf{x}_1 - \mathbf{x}_0\| - \|\mathbf{X}_1 - \mathbf{X}_0\| \right)^2$$

- Forces on point $\mathbf{x}_1$ applied by spring?
  - Homework!

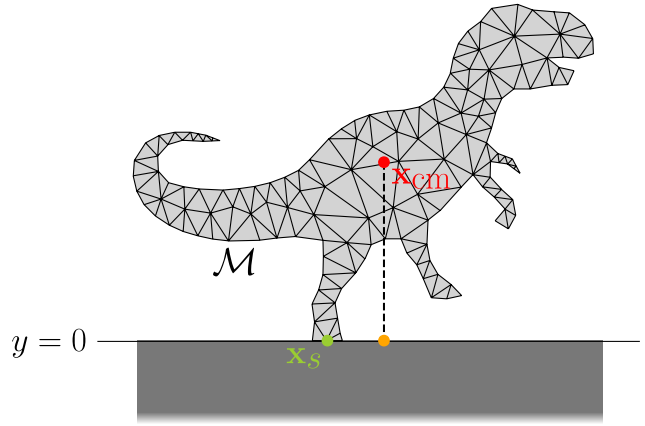
Spring System Energy



- Full elastic energy of spring system: just sum them up!

$$E_{\text{elastic}}(\mathbf{x}, \mathbf{X}) = \sum_{e_{ij}} \frac{1}{2} k_{ij} \left(\|\mathbf{x}_i - \mathbf{x}_j\| - \|\mathbf{X}_i - \mathbf{X}_j\| \right)^2$$

Make it stand - Optimization



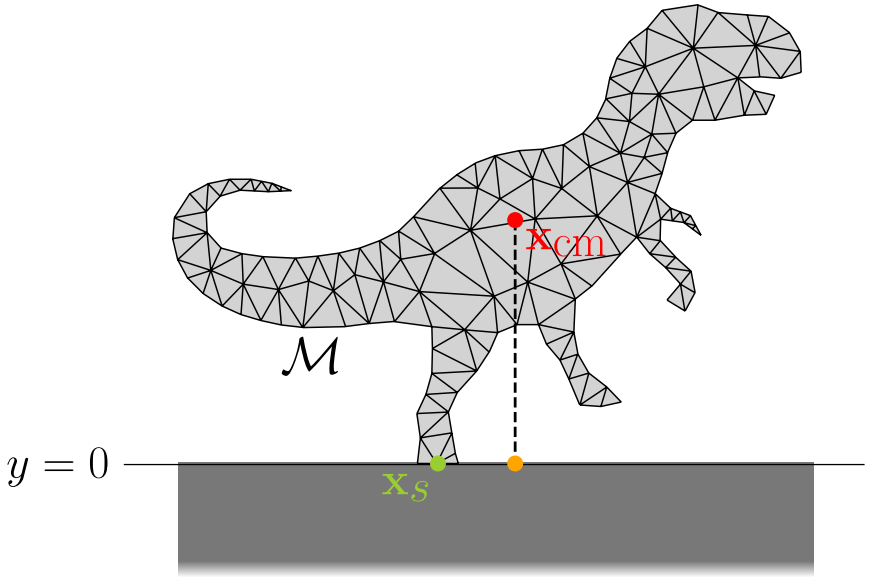
- Our goal is to find new vertex positions V of our mesh such that
  - the elastic energy is minimized
  - the centroid projects into the support line

$$\min_V E_{\text{elastic}}(V, V_0, F)$$

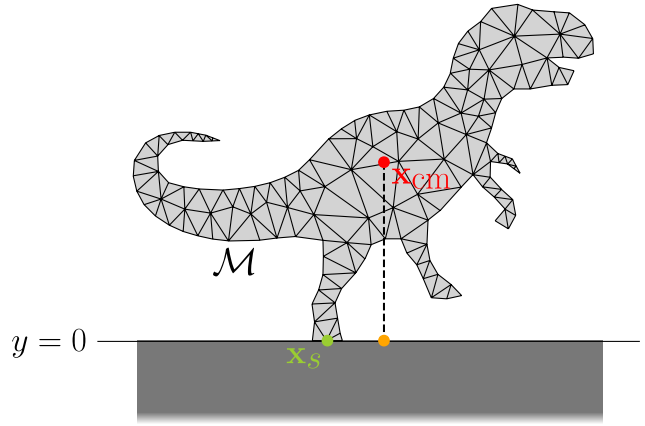
$$\text{s.t. } -\frac{l_{\text{sl}}}{2} \leq x_{\text{cm}} - x_{\text{csl}} \leq \frac{l_{\text{sl}}}{2}.$$

Constraints makes this problem harder to solve :(
How do we get a unconstrained formulation?

- where $\boldsymbol{x}_{\text{csl}}$ is the center of the support line and $\boldsymbol{l}_{\text{sl}}$ is its length.
- This is an example of a **non-linear, constrained** optimization problem.



Make it stand - Optimization



- Let's replace the balance constraint with a penalty energy term:

$$E_{\text{eq}}(V, V_0, F) = \frac{1}{2}(x_{\text{cm}} - x_{\text{csl}})^2$$

  - This energy penalizes deviations of the centroid from the center of the support line.

- Unconstrained nonlinear optimization problem:

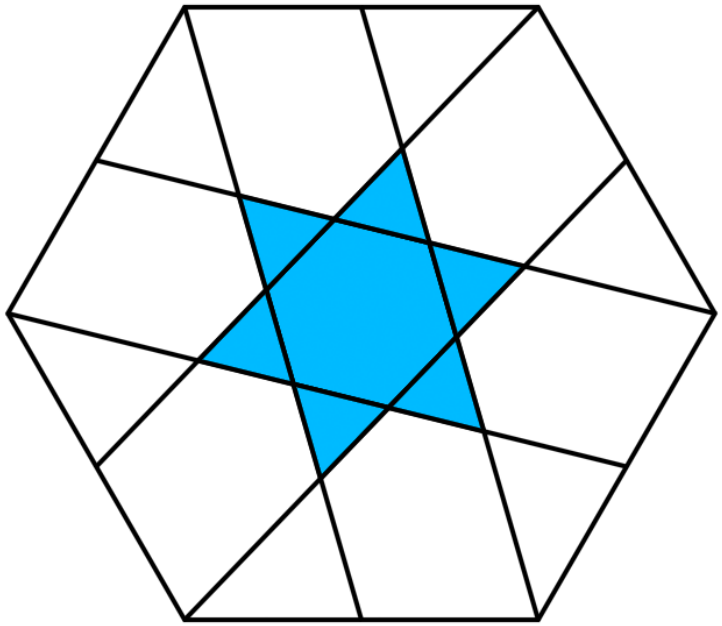
$$\min_V E_{\text{eq}}(V, V_0, F) + \omega E_{\text{elastic}}(V, V_0, F),$$

Increasing ω effectively makes the springs stiffer.

where ω controls a trade-off between ensuring that the object is standing and preserving the input shape.

Geometry Riddle

- What fraction of the area of the regular hexagon is shaded by the six pointed star (vertices are connected to midpoints)?



CS457 Geometric Computing

3B - Optimization Basics

Mark Pauly

Geometric Computing Laboratory - EPFL

Motivation

- Physical Simulation
  - Compute static equilibria of rigid and deformable objects, more general physical systems.
  - Variational integrators for dynamics problems.
- Optimal Design
  - Formulate the design problem as optimizing some performance metric.
  - Example: minimize distance of C.o.M. to the support polygon by repositioning surface points.

Introduction

Optimization is everywhere !

"Since the fabric of the universe is most perfect [...], nothing at all takes place in the universe in which some rule of maximum or minimum does not appear."

Leonhard Euler (1707-1783)

Leonard Euler on Wikipedia

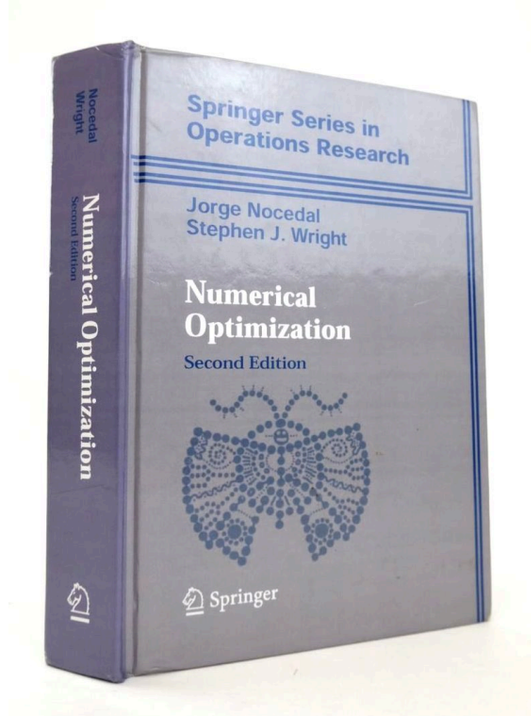
Types of Optimization Problems

- **Continuous** vs. **Discrete**
- **Unconstrained** vs. **Constrained**
- **None**, **One**, or **Many Objectives**
- **Deterministic** vs. **Stochastic**

Source: https://neos-guide.org/optimization-tree

Reading

- For Today: Chapter 2, pp 10-21



Nocedal, J., Wright, S. (2006). Numerical Optimization. United States: Springer New York.

Reading

Engineering Design Optimization

General Formulation

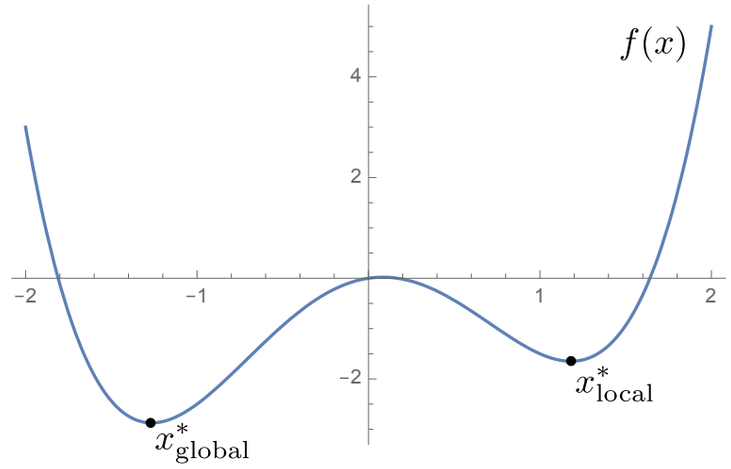
- **Standard Form** of *continuous* optimization problem

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) \\ \text{s.t. } &c_i(\mathbf{x}) = 0 \quad i \in \mathcal{E} \\ &c_i(\mathbf{x}) \geq 0 \quad i \in \mathcal{I} \end{aligned}$$

- with
  - **optimization variable** $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n$
  - **objective function** $f : \mathbb{R}^n \to \mathbb{R}$
  - **constraint functions** $c_i : \mathbb{R}^n \to \mathbb{R}$
  - **equality constraint index set** $\mathcal{E}$
  - **inequality constraint index set** $\mathcal{I}$

Unconstrained Optimization

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



- $\mathbf{x} \in \mathbb{R}^n$ with n often very large (e.g., thousands of variables)
- Global minimum $\mathbf{x}^* : f(\mathbf{x}^*) \leq f(\mathbf{x})\ \forall \mathbf{x}$
- Local minimum $\mathbf{x}^* : f(\mathbf{x}^*) \leq f(\mathbf{x})\ \forall \mathbf{x} \in \mathcal{N}$
- In general (unless f is convex), we hope only for *local* minima.
- We will assume f is at least C^2.

True or False?

A function that has two or more minima, must have at least one maximum.

A: True

B: False

True or False?

The global minimum (if it exists) of a function is unique.

A: True

B: False

True or False?

A polynomial of degree n has at most $\lfloor n/2 \rfloor$ local minima.

A: True

B: False

True or False?

If two functions f and g have a minimum at x, then $f \cdot g$ also has a minimum at x.

A: True

B: False

Unconstrained Optimization - Example I

- Minimize $f(x) = 2x^2 + 4x + 1$
- How to find solution x^*?
  - strategy (A): visual inspection
  - strategy (B): analytic solution

Unconstrained Optimization - Example II

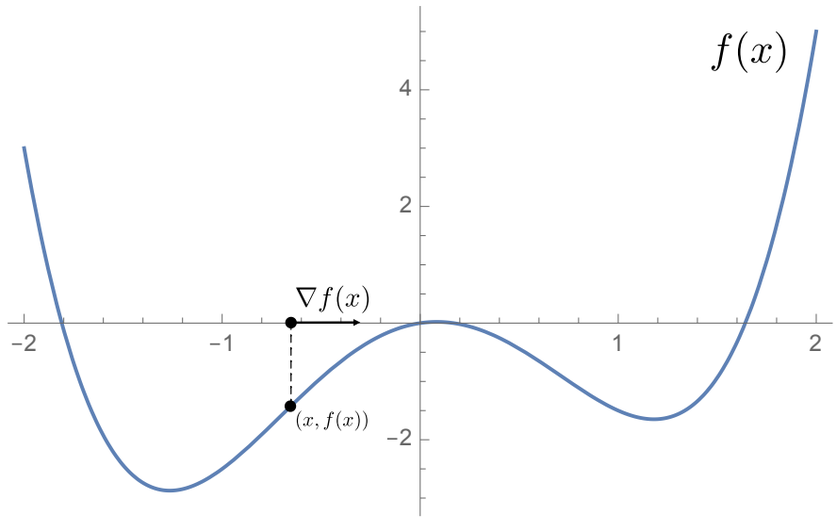
- minimize $f(x, y) = 4(\sin(x) - \cos(y))^2 + x^2 + y^2 + x + y$

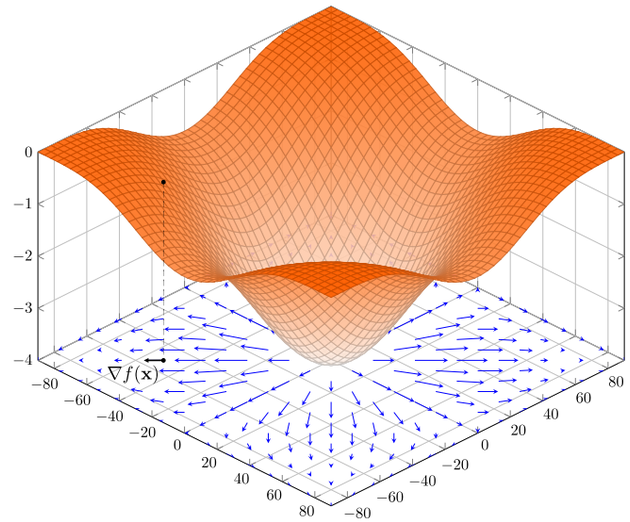
Algorithm Overview

- Our goal is to find the (local) minimum of the objective function f.
- Strategy: Iterative search
  - Given some initial point $\mathbf{x}_0$, we will construct a sequence of estimates $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k$ with $f(\mathbf{x}_0) \geq f(\mathbf{x}_1) \cdots \geq f(\mathbf{x}_k)$.
- Basic Questions:
  - How to find $\mathbf{x}_0$?
  - How to find the next iterate $\mathbf{x}_i$ given $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{i-1}$?
  - When to stop?
  - How to make this efficient?

Gradients and Gradient Descent

- The simplest, most intuitive algorithm for finding a minimizer: iteratively step in the direction of *steepest descent*.
- In 1D, this direction is given by *the sign of* $-f'(x)$.
- In nD, this direction points opposite the gradient vector, $\nabla f = \left[\frac{\partial f}{\partial x_1} \quad \dots \quad \frac{\partial f}{\partial x_n} \right]^\top$ (in Cartesian coordinates).



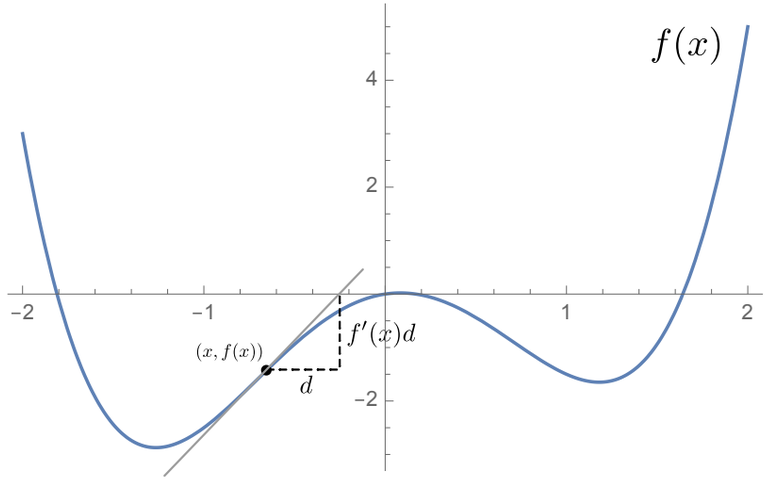


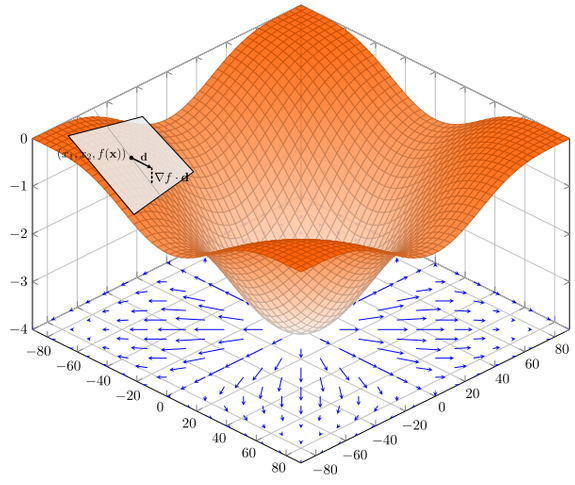
Gradient as the Best Linear Approximation

- The gradient can be interpreted as representing the best linear approximation f around the current point $\mathbf{x}$:

$$f(\mathbf{x}+\mathbf{d}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \mathbf{d} + O(\|\mathbf{d}\|^2)$$

(first-order Taylor expansion).



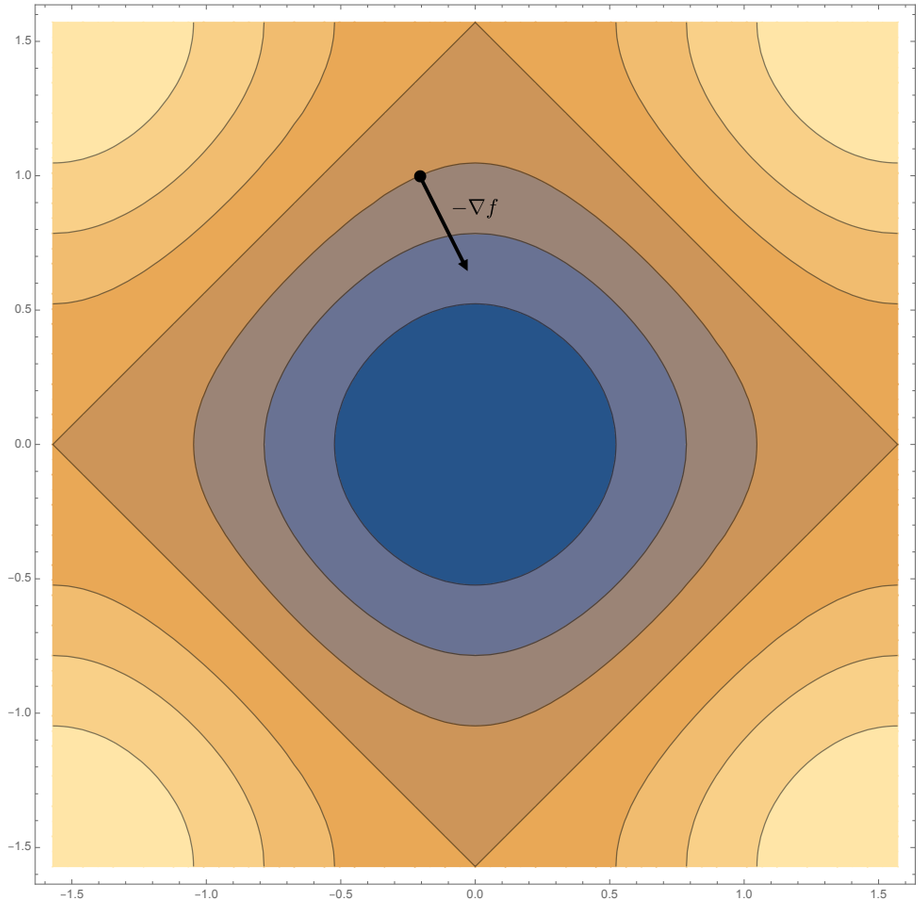


Taylor expansion

Another Graphical Interpretation

- Stepping along the negative gradient direction moves *perpendicularly* to the current contour.

$$f(\mathbf{x}+\mathbf{d}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \mathbf{d} + O(\|\mathbf{d}\|^2)$$


$-\nabla f$
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5

Gradient as Steepest Ascent Direction

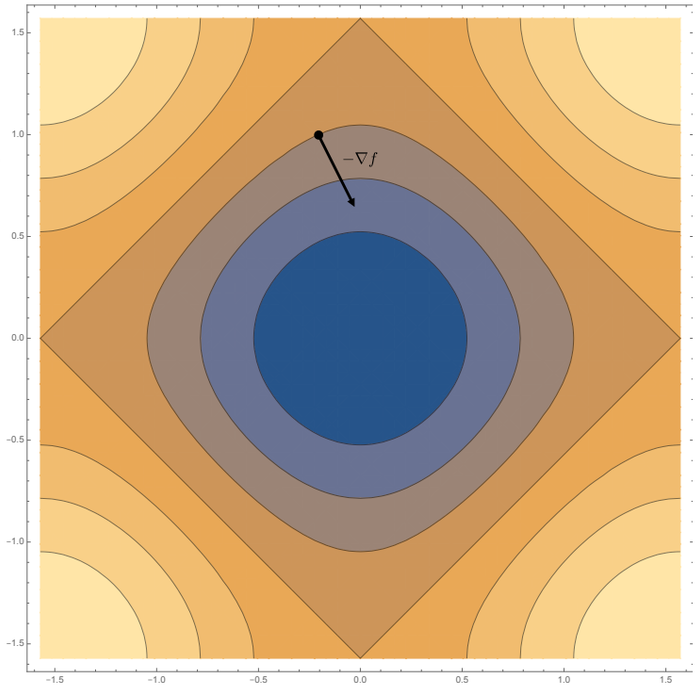
- Taylor expansion: $f(\mathbf{x}+\mathbf{d}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^\top \mathbf{d} + O(\|\mathbf{d}\|^2)$
- Why does $-\nabla f$ point in the direction of quickest descent?
- Search for the optimal step $\mathbf{d}^*$ of very short length $0 < \epsilon \ll 1$

$$\min_{\|\mathbf{d}\|=\epsilon} f(\mathbf{x}+\mathbf{d}) \approx \min_{\|\mathbf{d}\|=\epsilon} f(\mathbf{x}) + \nabla f \cdot \mathbf{d}$$

$$\iff \quad \min_{\|\mathbf{d}\|=\epsilon} \nabla f \cdot \mathbf{d} = \min_{\|\mathbf{d}\|=\epsilon} \epsilon \|\nabla f\| \cos(\theta),$$

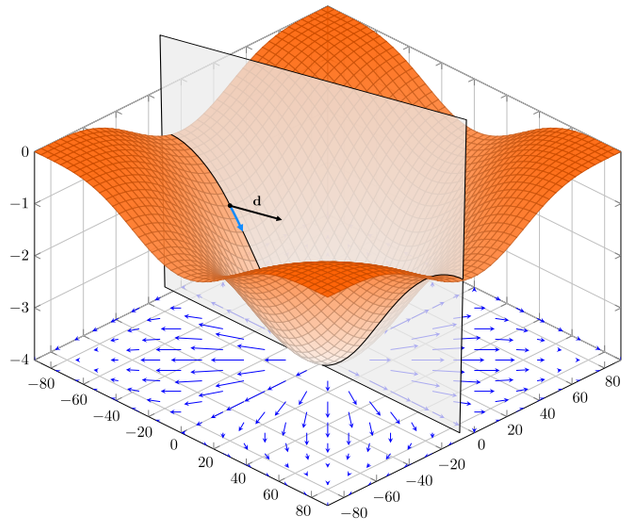
  where θ is the angle between ∇f and $\mathbf{d}$.

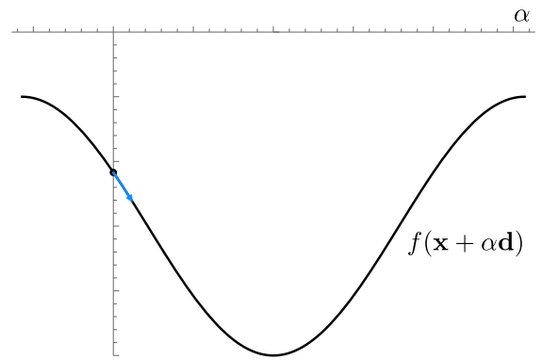
- Minimum occurs when $\cos(\theta) = -1 \quad \iff \quad \mathbf{d}^* = -\epsilon \frac{\nabla f}{\|\nabla f\|}.$



How far to Step?

- Once step direction $\mathbf{d}$ is chosen, a step length $\alpha > 0$ must be picked.
- Finding the best α is a 1D optimization problem (regardless of n):





- Solving this exactly is usually difficult and not worthwhile.
- Instead, simpler strategies for picking α ("inexact line searches") are used. For now, let's consider using a small, constant α.

Basic Gradient Descent Algorithm

Algorithm: `basic_steepest_descent`
Input:
f function to minimize
$\mathbf{x}$ initial guess
α step length
while not `converged` $(f, \mathbf{x})$
$\mathbf{x} = \mathbf{x} - \alpha \nabla f$
end

- Remaining questions:
  - What convergence criterion to use?
  - Is it possible to pick a single α that performs well?
  - How to compute gradients?