CS457 Geometric Computing

4A - Line Search

Mark Pauly

Geometric Computing Laboratory - EPFL

Exhibition

https://epfl-pavilions.ch/en/exhibitions/aetherocohedron-elba

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*?
  - How do we find the best modification ***efficiently***?

Recap: Make it stand - Optimization

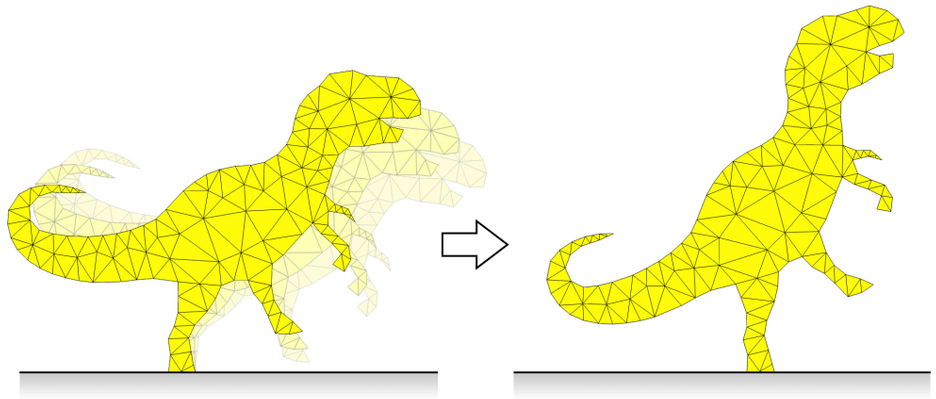
- We can modify a given input shape to be in balance by minimizing the energy

$$J(V, F) = E_{\text{eq}}(V, F) + \omega E_{\text{elastic}}(V, F),$$

- This leads to an **unconstrained** optimization problem

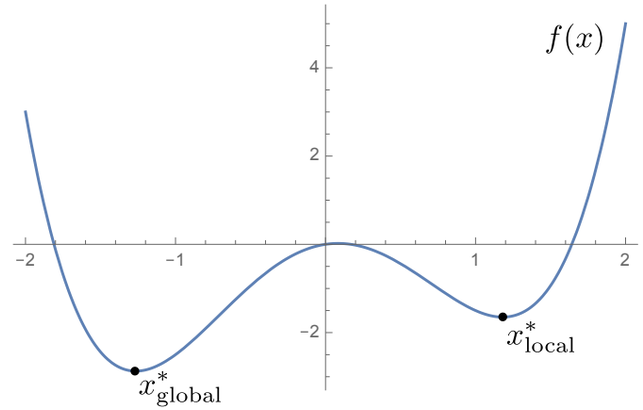
$$\min_{V} J(V, F)$$

  - where we have to solve for the unknown vertex coordinates V of our mesh.



Recap: 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.

Recap: Basic Gradient Descent Algorithm

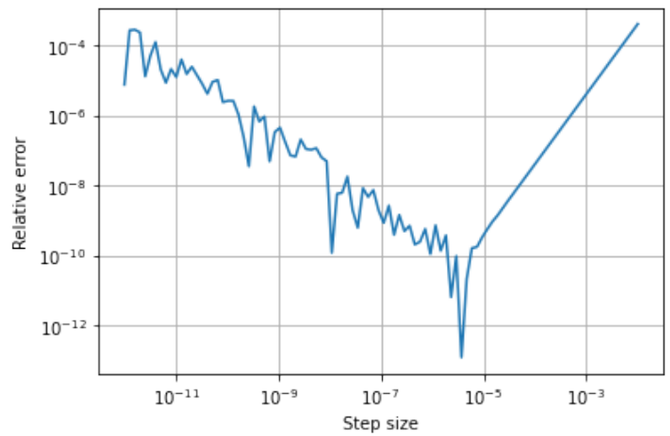
Algorithm: `basic_steepest_descent`
Input:
f function to minimize
$\mathbf{x}$ initial guess
α step length
while not `converged` $(f, \mathbf{x})$
$\mathbf{x} = \mathbf{x} - \alpha \nabla f$
end

Derivative Checks

- Most common cause of failure in solving an optimization problem is mistakes in working out derivatives and/or translating them to code.
- In this course: strive for geometric approach for deriving gradients.
- You should always validate derivatives by comparing to finite differences:

$$\nabla f \cdot \mathbf{d} \approx \frac{f(\mathbf{x} + \epsilon \mathbf{d}) - f(\mathbf{x} - \epsilon \mathbf{d})}{2\epsilon}$$

- Plot relative error vs ϵ on a log-log plot
  - should see second-order convergence for centered finite differences.



Recap: Convergence to Local Minimum

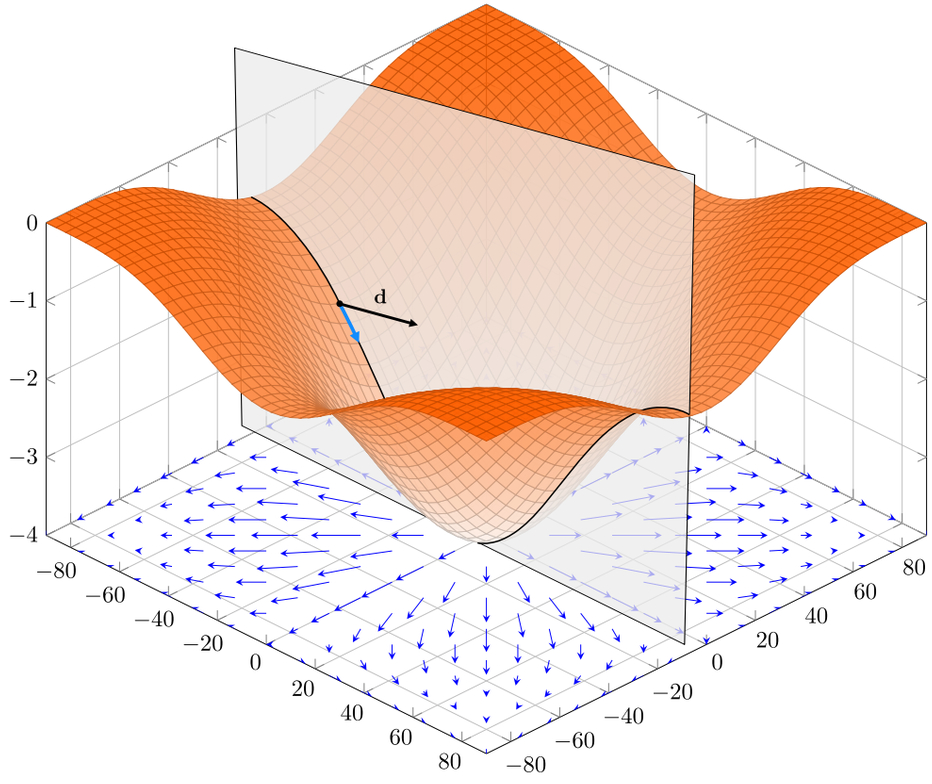
- First-oder necessary condition: $\nabla f(\mathbf{x}^*) = 0$
  - $\mathbf{x}^*$ is a *stationary point*.
- Second-order necessary condition: $\mathbf{d}^T H \mathbf{d} \geq 0 \; \forall \mathbf{d}$.
  - where $H = \nabla^2 f(\mathbf{x})$ is the Hessian of f.
  - H must be *positive semi-definite* (eigenvalues $\lambda_i \geq 0 \;\; 1 \leq i \leq n$).
- Second-order sufficient condition: $\mathbf{d}^T H(\mathbf{x}^*) \mathbf{d} > 0 \; \forall \mathbf{d} \neq 0$
  - equivalently $\lambda_i > 0, \;\; 1 \leq i \leq n$.
  - no longer a necessary condition.

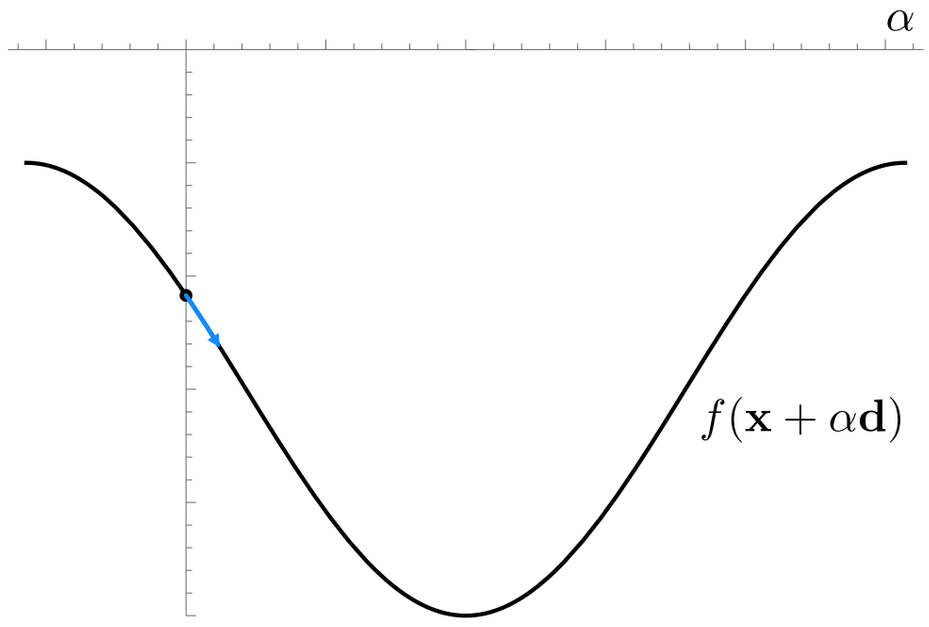
Example

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

How far to Step?

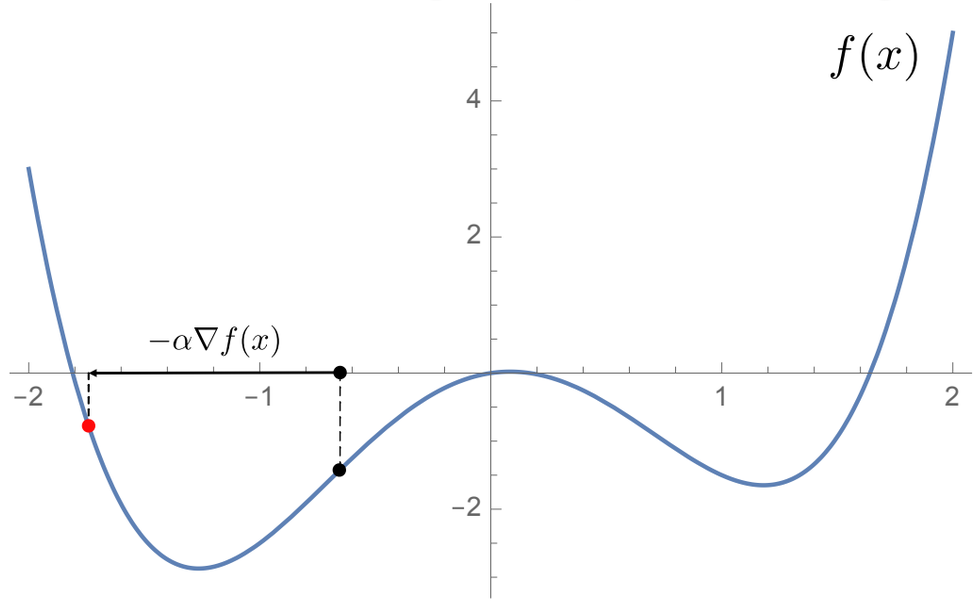
- Once step direction $\mathbf{d}$ is chosen, a step length $\alpha > 0$ must be picked.
- Finding the best α is a 1D optimization problem (regardless of n):





Choosing a Step Size

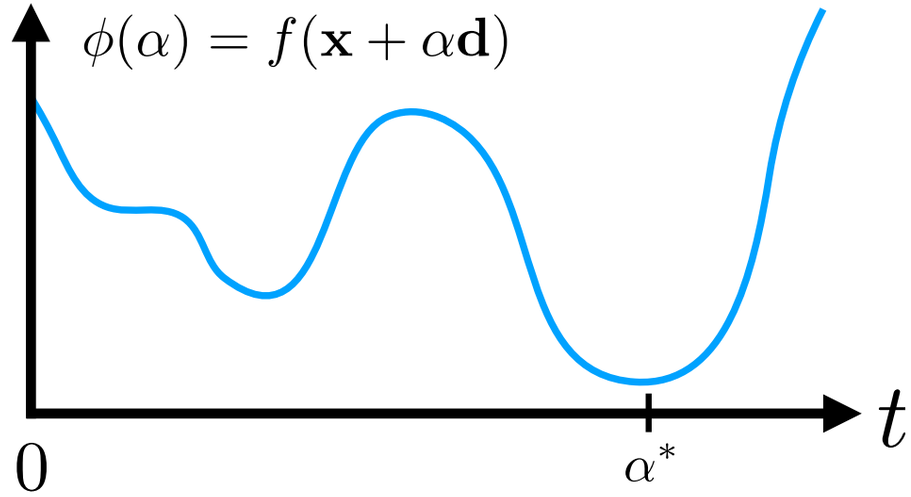
- Small step sizes will take forever, while large steps can diverge:



- There is no clear range of step sizes to try for gradient descent (though problem-specific knowledge may suggest one).
  - Thankfully for the Newton-type methods we'll see later, $\alpha = 1$ is an obvious first choice.
- We really need to adjust α as the algorithm proceeds…

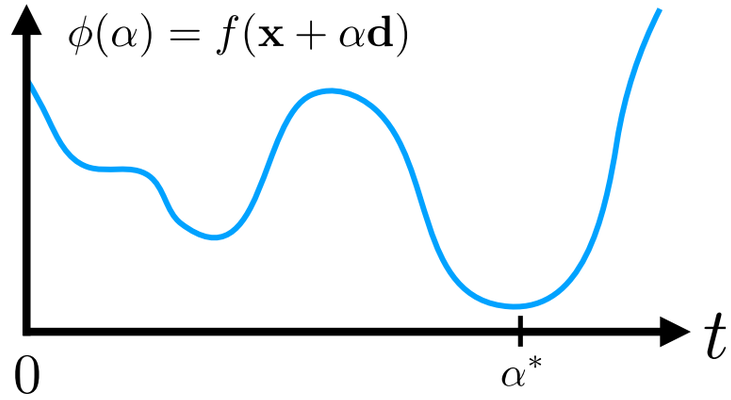
Line Search

- **Goal**
  - **given**: search direction $\mathbf{d}$
  - **find**: $\alpha > 0$ which decreases $\phi(\alpha) = f(\mathbf{x} + \alpha\mathbf{d})$
    - always possible if $\mathbf{d}$ is descent direction, since $\phi'(0) = \mathbf{d} \cdot \nabla f(\mathbf{x}) < 0$.



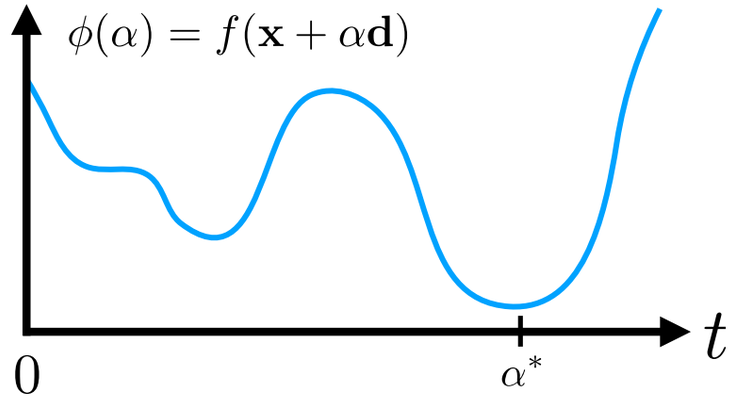
- **Strategies**
  - **exact line search**: minimize $\phi(\alpha)$ subject to $\alpha > 0$
    - rather costly, unless analytic (e.g. convex quadratic)
    - usually convergence speed doesn't pay off (depends more on $\mathbf{d}$)
  - **inexact line search**: make sure that objective value decreases "sufficiently"
    - minimize computational cost of line search (e.g. number of function/gradient evaluations)
    - how much decrease is necessary to warrant "optimal" convergence?

Armijo Condition



- **Observation**
  - requiring $f(\mathbf{x} + \alpha\mathbf{d}) \leq f(\mathbf{x})$ does not guarentee convergence to local minimium.
- **Armijo Condition**
  - $f(\mathbf{x} + \alpha\mathbf{d}) \leq f(\mathbf{x}) + c_1\, \alpha\, \mathbf{d} \cdot \nabla f(\mathbf{x})$ with damping coefficient $c_1 \in (0, 1)$
  - identical to $\phi(\alpha) \leq \phi(0) + \alpha c_1 \phi'(0)$
  - **interpretation**: reduction should be proportional to α and slope at $\alpha = 0$
  - also known as **sufficient decrease condition**

Strong Wolfe Conditions



- **Strong Wolfe Conditions** with $c_1 \in (0, 1)$ and $c_2 \in (c_1, 1)$
  1. **sufficient decrease condition**: $f(\mathbf{x} + \alpha\mathbf{d}) \leq f(\mathbf{x}) + c_1\, \alpha\, \mathbf{d} \cdot \nabla f(\mathbf{x})$
     - identical to $\phi(\alpha) \leq \phi(0) + \alpha c_1 \phi'(0)$
  2. **curvature condition**: $|\mathbf{d} \cdot \nabla f(\mathbf{x} + \alpha\mathbf{d})| \leq c_2 |\mathbf{d} \cdot \nabla f(\mathbf{x})|$
     - identical to $|\phi'(\alpha)| \leq c_2 |\phi'(0)|$ (make sure that ϕ is "flat enough" at α)
     - prevents slope at α to be too positive
  - very general, often used in practice, e.g. with $c_1 = 10^{-4}$ and $c_2 = 0.9$

Backtracking Line Search

- Simple, popular, and effective inexact line search based on the Armijo condition:

Algorithm: `steepest_descent`
Input:
f function to minimize
$\mathbf{x}$ initial guess
while not `converged` $(f, \mathbf{x})$
$\alpha = \texttt{line\_search}(f, \mathbf{x}, -\nabla f)$
$\mathbf{x} = \mathbf{x} - \alpha \nabla f$
end

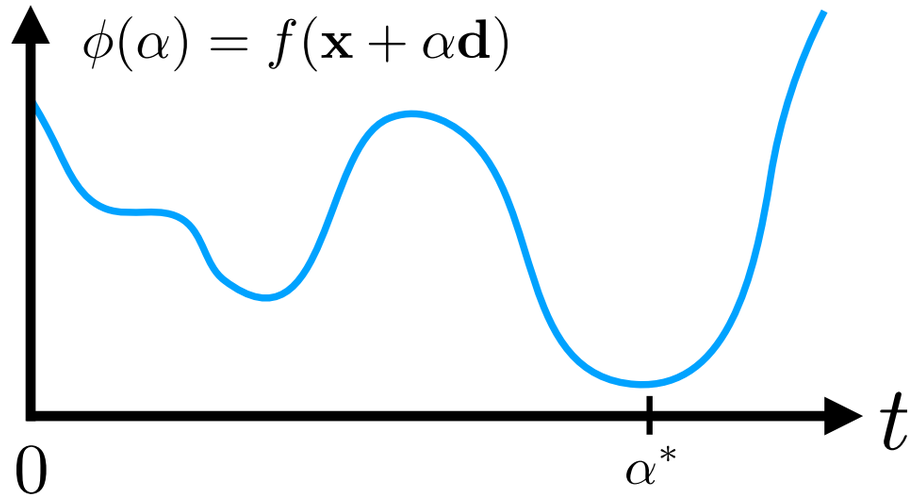
Algorithm: `line_search`
Input: f, $\mathbf{x}$, $\mathbf{d}$
$\alpha = \overline{\alpha}$
while $f(\mathbf{x} + \alpha\mathbf{d}) > f(\mathbf{x}) + c_1\, \alpha\mathbf{d} \cdot \nabla f$:
$\alpha = \alpha/2$ // Scale factor customizable
end
return α

- Still depends on a good selection for initial step length $\overline{\alpha}$.
  - Can choose $\overline{\alpha}$ based on α from previous steps.
- As long as $\overline{\alpha} \to 0$ is forbidden, this algorithm will converge to a minimum.

Example

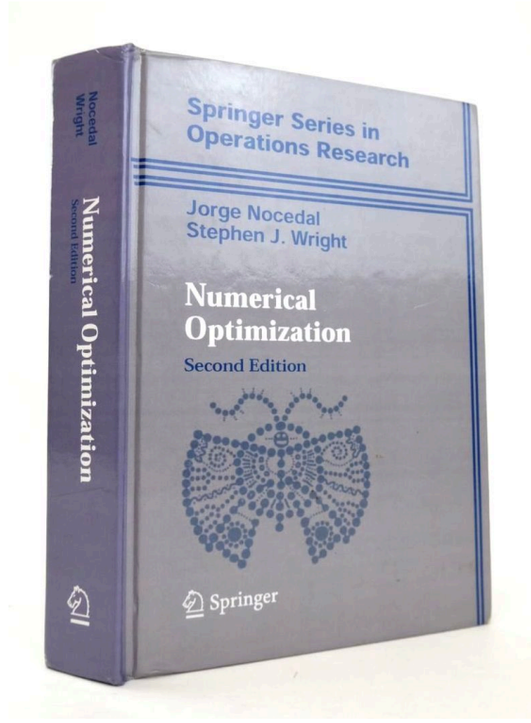
- Backtracking Line Search with $c_1 = 0.5$

Algorithm: `line_search`
Input: f, $\mathbf{x}$, $\mathbf{d}$
$\alpha = \overline{\alpha}$
while $f(\mathbf{x} + \alpha\mathbf{d}) > f(\mathbf{x}) + c_1\ \alpha\ \mathbf{d} \cdot \nabla f$:
    $\alpha = \alpha/2$
end
return α



Reading

- Chapter 3



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

True or False?

Backtracking line search will find the largest possible step length that satisfies the Armijo condition.

A: True

B: False

True or False?

The performance of gradient descent with backtracking line search depends on the initial step length.

A: True

B: False

True or False?

Gradient descent with exact line search is always more efficient than with inexact line search.

A: True

B: False

True or False?

If gradient descent has converged, we can be sure we have found a local minimum.

A: True

B: False

Curious Fact

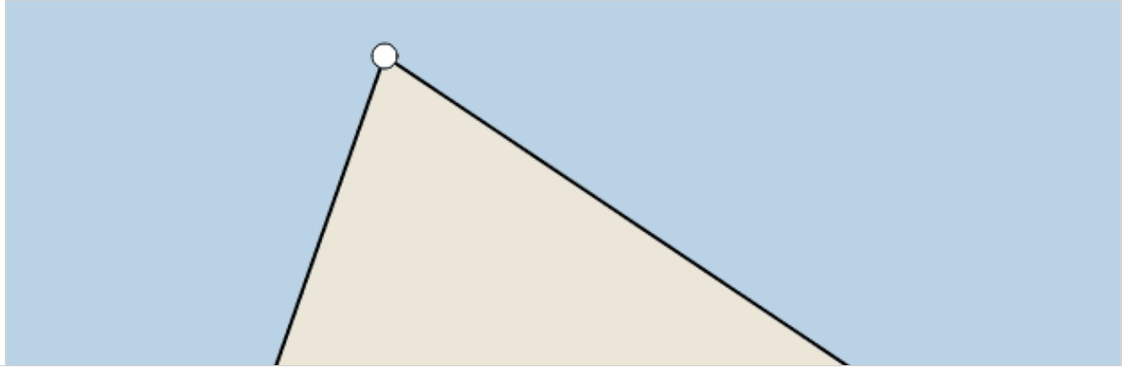
GeoGebra Search Google Classroom GeoGebra Classroom Sign

Morley Action!

Author: Tim Brzezinski

Topic: Angles, Equilateral Triangles

The applet below dynamically illustrates **Morley's Triangle Theorem**.
Feel free to move the triangle's white vertices anywhere you'd like each time before re-sliding the slider!



Source *Wikipedia Article*

CS457 Geometric Computing

4B - Newton's Method

Mark Pauly

Geometric Computing Laboratory - EPFL

Gradient Descent

- **Gradient Descent**
  - search direction $\mathbf{d} = -\nabla f(\mathbf{x})$
  - very simple, but often very slow
  - convergence rate greatly depends on **condition number** of Hessian

- Instructive Example in $\mathbb{R}^2$
  - $f(\mathbf{x}) = \frac{1}{2}(x_1^2 + \gamma x_2^2)$ with $\gamma > 0$ and $\mathbf{x}^* = (0,0)^T$
  - with exact line search, starting at $x^{(0)} = (\gamma, 1)^T$
    $x_1^{(k)} = \gamma(\frac{\gamma-1}{\gamma+1})^k, x_2^{(k)} = (-\frac{\gamma-1}{\gamma+1})^k$
  - very slow if $\gamma \gg 1$ or $\gamma \ll 1$
  - e.g. $\gamma = 100 \Rightarrow x_1^{(k)} = 100(\frac{99}{101})^k, x_2^{(k)} = (-\frac{99}{101})^k$
  - after hundred iterations $x \approx (13.53, 0.1353)^T$ still far from x^*

Example

Example

$\gamma = 10$

$\gamma = 1$

Newton's Method

- **Newton Direction**
  - search direction is **Newton step** $\mathbf{d} = -\nabla^2 f(\mathbf{x})^{-1} \nabla f(\mathbf{x}) = -H^{-1} \nabla f$
  - often good choice in practice (if second derivatives available)
  - **convergence**
    - typically rapid if magnitude of 3rd derivative bounded
    - quadratic convergence near $\mathbf{x}^*$ (easy to get high-accuracy solution)
- **Interpretations**
  1. Minimizer of second-order approximation $f(\mathbf{x} + \mathbf{d}) \approx \hat{f}(\mathbf{x} + \mathbf{d}) = f(x) + \nabla f(\mathbf{x})^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \nabla^2 f(\mathbf{x}) \mathbf{d}$
  2. Solution of linearized optimality conditions $\nabla f(\mathbf{x} + \mathbf{d}) \approx \nabla \hat{f}(\mathbf{x} + \mathbf{d}) = \nabla f(\mathbf{x}) + \nabla^2 f(\mathbf{x}) \mathbf{d} = 0$

1D Example

- Newton on logarithmic function
  - $f(x) = -\log(x + 11) - \log(11 - x) + 5$, domain $[-10, 10]$, $x^{(0)} = 10$

2D Example

Newton's Method with Line Search

- A line search is still needed to ensure global convergence.

Algorithm: `Newton's Method`
Input:
f function to minimize
$\mathbf{x}$ initial guess
while not `converged` $(f, \mathbf{x})$
$\mathbf{d} = -H^{-1}\nabla f$
$\alpha = \texttt{line\_search}(f, \mathbf{x}, \mathbf{d})$
$\mathbf{x} = \mathbf{x} + \alpha\,\mathbf{d}$
end

Algorithm: `line_search`
Input: f, $\mathbf{x}$, $\mathbf{d}$
$\alpha = 1.0$ // No $\overline{\alpha}$ needed!
while $f(\mathbf{x} + \alpha\mathbf{d}) > f(\mathbf{x}) + c\,\alpha\mathbf{d} \cdot \nabla f$:
$\alpha = \alpha/2$ // Scale factor customizable
end
return α

- $\overline{\alpha} = 1$ is a natural initial step length.
  - First try stepping to the optimum of the local quadratic approximation.
  - Close to a minimum no backtracking will be necessary ($\alpha = 1$) and quadratic convergence kicks in.

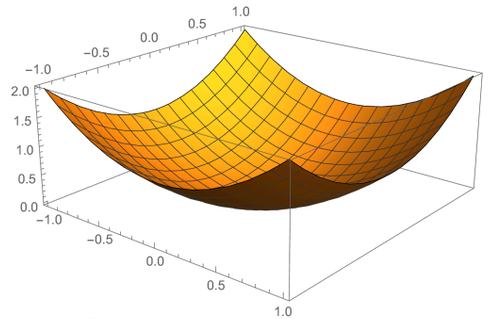
Solving the Newton Equations

$$H\mathbf{d} = -\nabla f$$

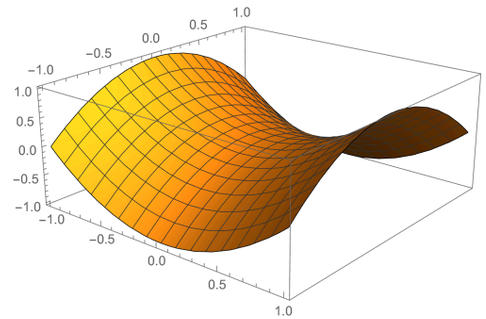
- This will be the time and memory bottleneck of our simulation routines.
- Useful properties of the system:
  - Sparse
  - Symmetric
  - Positive definite **in neighborhood of local minima**
- We can use efficient linear system solvers
  - Conjugate gradient method ("Newton CG")
  - Sparse Cholesky factorization $H = LL^\top$, e.g., using CHOLMOD

Caveat: Indefiniteness

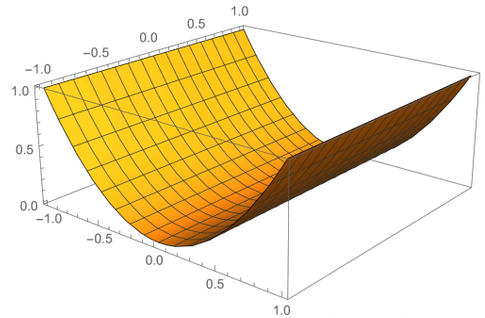
- When is $\mathbf{d} = -H^{-1}\nabla f$ really a descent direction?
  - $f(\mathbf{x}+\mathbf{d}) - f(\mathbf{x}) \approx \nabla f \cdot \mathbf{d} = -\nabla f \cdot H^{-1}\nabla f$
  - If $\mathbf{v} \cdot H\mathbf{v} > 0 \; \forall \mathbf{v} \neq \mathbf{0}$, then we are guaranteed $-\nabla f \cdot H^{-1}\nabla f < 0$
  - But if H has negative eigenvalues (indefinite), and if ∇f has a significant component in the corresponding eigenspaces, then we can have $-\nabla f \cdot H^{-1}\nabla f > 0$ (i.e., $\mathbf{d}$ is an *ascent* direction).
- Newton's method is attracted to saddle points.



Both Hessian eigenvalues positive: guaranteed **minimum**.



One Hessian eigenvalue negative: **saddle point**, not a minimum.



One Hessian eigenvalue zero: **inconclusive** (could be strong min, weak min, or saddle).

Video Lecture on Eigenvalues/Eigenvectors

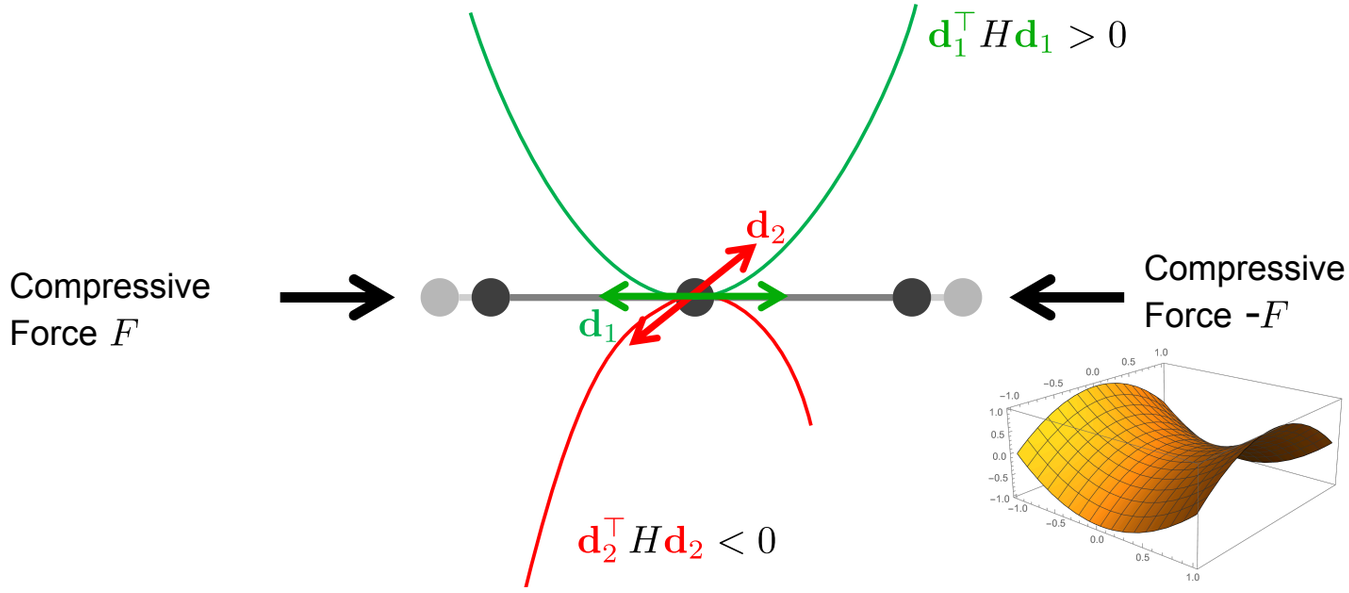
Example: Indefiniteness

- Non convex polynomial of degree 6
  - **minimize**
    - $f(x,y) = (\frac{3}{2} - x + xy)^2 + (\frac{9}{4} - x + xy^2)^2 + (\frac{21}{8} - x + xy^3)^2$
  - **initial point**
    - $x^{(0)} = (4,1)^T$
  - **gradient**
    - $\nabla f(x^{(0)}) = (0, 111)^T$
  - **Hessian**
    - $\nabla^2 f(x^{(0)}) = \begin{pmatrix} 0 & 27.75 \\ 27.75 & 610 \end{pmatrix}$
  - **Newton step**
    - $\mathbf{d} = (-4, 0)^T$
    - **not a descent direction** since $\nabla f(x^{(0)})^T \mathbf{d} = 0$ (should be < 0)

Example

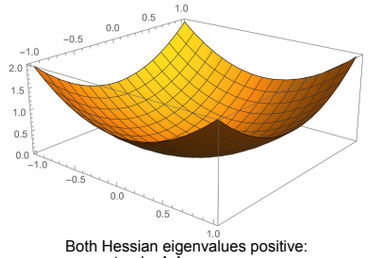
Example: Indefiniteness

- Physical example: a system of two compressed springs in 2D.

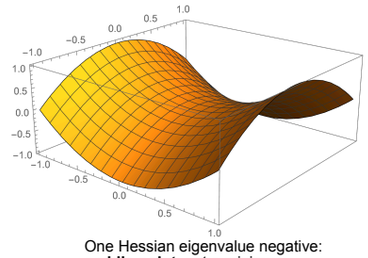


- $\mathbf{d_1}$ is a *direction of positive curvature* in the central vertex's energy landscape, while $\mathbf{d_2}$ is a *direction of negative curvature*.

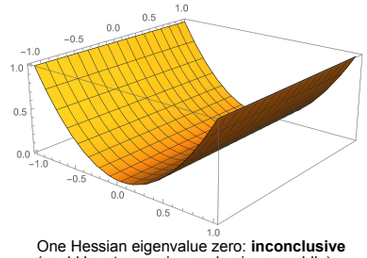
Caveat: Indefiniteness



Both Hessian eigenvalues positive: guaranteed **minimum**.



One Hessian eigenvalue negative: **saddle point**, not a minimum.



One Hessian eigenvalue zero: **inconclusive** (could be strong min, weak min, or saddle).

- To guarantee a descent direction, we must detect and modify indefinite H

$$H \text{ indefinite} \quad \rightarrow \quad \tilde{H} \text{ positive definite}$$

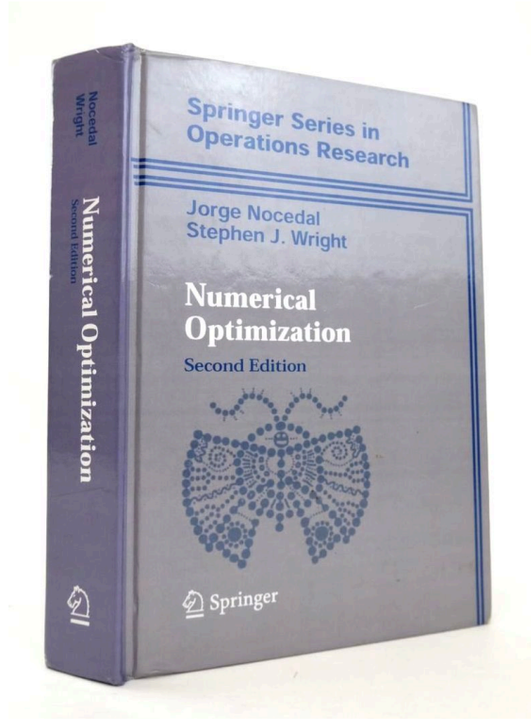
- Brute force:
  - Compute full *eigendecomposition*: $H = Q\Lambda Q^\top$.
  - Modify eigenvalues to $\tilde{\Lambda}$ with the rule $\tilde{\lambda}_i = |\lambda_i|$ or $\tilde{\lambda}_i = \mathbf{max}(\lambda_i, \epsilon) \quad (\epsilon > 0)$.
  - Construct the modified Hessian: $\tilde{H} = Q\tilde{\Lambda}Q^\top$.
  - very expensive, but more efficient methods exist (see Nocedal & Wright).

Newton's Method

- Pros:
  - Extremely rapid convergence close to optimum
  - Generally good directions far away
  - Easy backtracking line search: natural initial step length of $\overline{\alpha} = 1$.
- Cons:
  - Difficulties encountered when H not positive definite:
    - Newton's method will step towards saddle points if we do not modify it.
    - Also, the most efficient linear solvers require a positive definite H.
  - Can be difficult or slow to compute Newton direction

Reading

- Chapter 2,3



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

Turtles

Source

CS457 Geometric Computing

4C - Quasi-Newton Methods

Mark Pauly

Geometric Computing Laboratory - EPFL

Quasi-Newton Methods

- **Quasi-Newton Methods**
  - replace Hessian $\nabla^2 f(\mathbf{x})$ with approximation $B \approx \nabla^2 f(\mathbf{x})$
  - guaranteed to provide descent direction if $B \in S^n_{++}$
  - alternative to Newton's method if
    - second derivatives are not available
    - second derivatives are too expensive to compute
    - rapid prototyping
    - in combination with algorithmic differentiation
  - superlinear convergence (typically better than gradient descent)
- **Idea**
  - estimate second derivatives on-the-fly based on first derivative information

DFP Method

- **Davidon–Fletcher–Powell formula (DFP)**
  - developed by Davidon in 1950s (because of frustration)
  - computers were not stable and optimization crashed before termination
- Assume quadratic model at current iterate $\mathbf{x}_k$
  - **notation**: $\mathbf{x}_k = \mathbf{x}^{(k)}, \nabla f_k = \nabla f(\mathbf{x}^{(k)})$, etc.
  - $m_k(\mathbf{d}) = f_k + \nabla f_k^T \mathbf{d} + \frac{1}{2}\mathbf{d}^T B_k \mathbf{d}$
  - search direction $\mathbf{d}_k = -B_k^{-1} \nabla f_k$
  - next iterate $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$
- How to compute B_{k+1}?

DFP Method

- Assume quadratic model at current iterate $\mathbf{x}_k$
  - $m_k(\mathbf{d}) = f_k + \nabla f_k^T \mathbf{d} + \frac{1}{2}\mathbf{d}^T B_k \mathbf{d}$
  - search direction $\mathbf{d}_k = -B_k^{-1} \nabla f_k$
  - next iterate $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$

- How to compute B_{k+1}?
  - curvature can be measured by change of first derivative
  - require that m_{k+1} matches first derivative at last two iterates,
    i.e. $\nabla m_{k+1}(0) = \nabla f_{k+1}$ and $\nabla m_{k+1}(-\alpha_k \mathbf{d}_k) = \nabla f_k$
- **Secant Equation**: $B_{k+1}\mathbf{s}_k = \mathbf{y}_k$ with $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\mathbf{y}_k = \nabla f_{k+1} - \nabla f_k$
- **Curvature Condition**: $\mathbf{s}_k^T \mathbf{y}_k > 0$ required because B_{k+1} positive definite

DFP Method

- Find $\mathbf{s}_k$ and $\mathbf{y}_k$ satisfying curvature condition $\mathbf{s}_k^T \mathbf{y}_k > 0$
  - guaranteed by Wolfe or strong Wolfe condition since
    - $\mathbf{y}_k^T \mathbf{s}_k \geq (c_2 - 1)\alpha_k \nabla f_k^T \mathbf{d}_k > 0$
- Find B_{k+1} by modifying B_k as little as possible
  - minimize $||B - B_k||_W$
  - subject to $B\mathbf{s}_k = \mathbf{y}_k$ and $B \in S_{++}^n$
- **DFP formula** for B_{k+1}

$$B_{k+1} = (I - \rho_k \mathbf{y}_k \mathbf{s}_k^T) B_k (I - \rho_k \mathbf{s}_k \mathbf{y}_k^T) + \rho_k \mathbf{y}_k \mathbf{y}_k^T \quad \text{with} \quad \rho_k = \frac{1}{\mathbf{y}_k^T \mathbf{s}_k}$$

BFGS Method

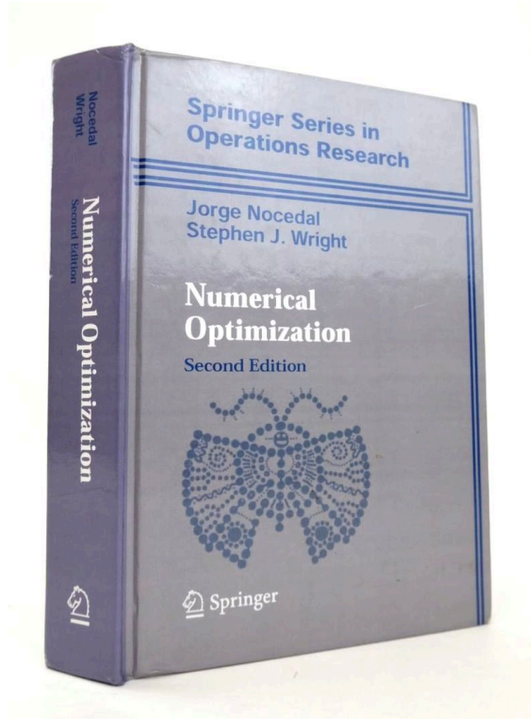
- **Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS)**
  - similar derivation as DFP
  - but directly deriving B_{k+1}^{-1} instead of B_{k+1}
  - minimize $||B^{-1} - B_k^{-1}||_W$ subject to $B_{k+1}^{-1}\mathbf{y}_k = \mathbf{s}_k$ and $B_{k+1}^{-1} \in S_{++}^n$
  - considered most effective of all quasi-Newton update formulas
- **BFGS formula**
  - $B_{k+1}^{-1} = V_k B_k^{-1} V_k + \rho_k \mathbf{s}_k \mathbf{s}_k^T$ with $V_k = (I - \rho_k \mathbf{y}_k \mathbf{s}_k^T)$ and $\rho_k = \frac{1}{\mathbf{y}_k^T \mathbf{s}_k}$
- **Additional aspects**
  - choose B_0^{-1}: problem specific or simply identity
  - use line search for (strong) Wolfe conditions, or skip updates when curvature condition is violated
  - BFGS has effective self-correcting properties

Summary

- **Algorithms for unconstrained optimization**
  - **Gradient Descent**
    - easy to implement
    - linear convergence
    - slow in practice, seldomly good choice
  - **Newton's Method**
    - often good choice if second-order derivatives available
    - quadratic convergence near $\boldsymbol{x}^*$
    - Hessian modification for non-convex problems
  - **Quasi-Newton Methods**
    - BFGS generally preferred
    - good for prototyping or if second-order derivatives not available
    - superlinear/linear convergence ($\boldsymbol{m} = 1$ related to CG method)

Reading

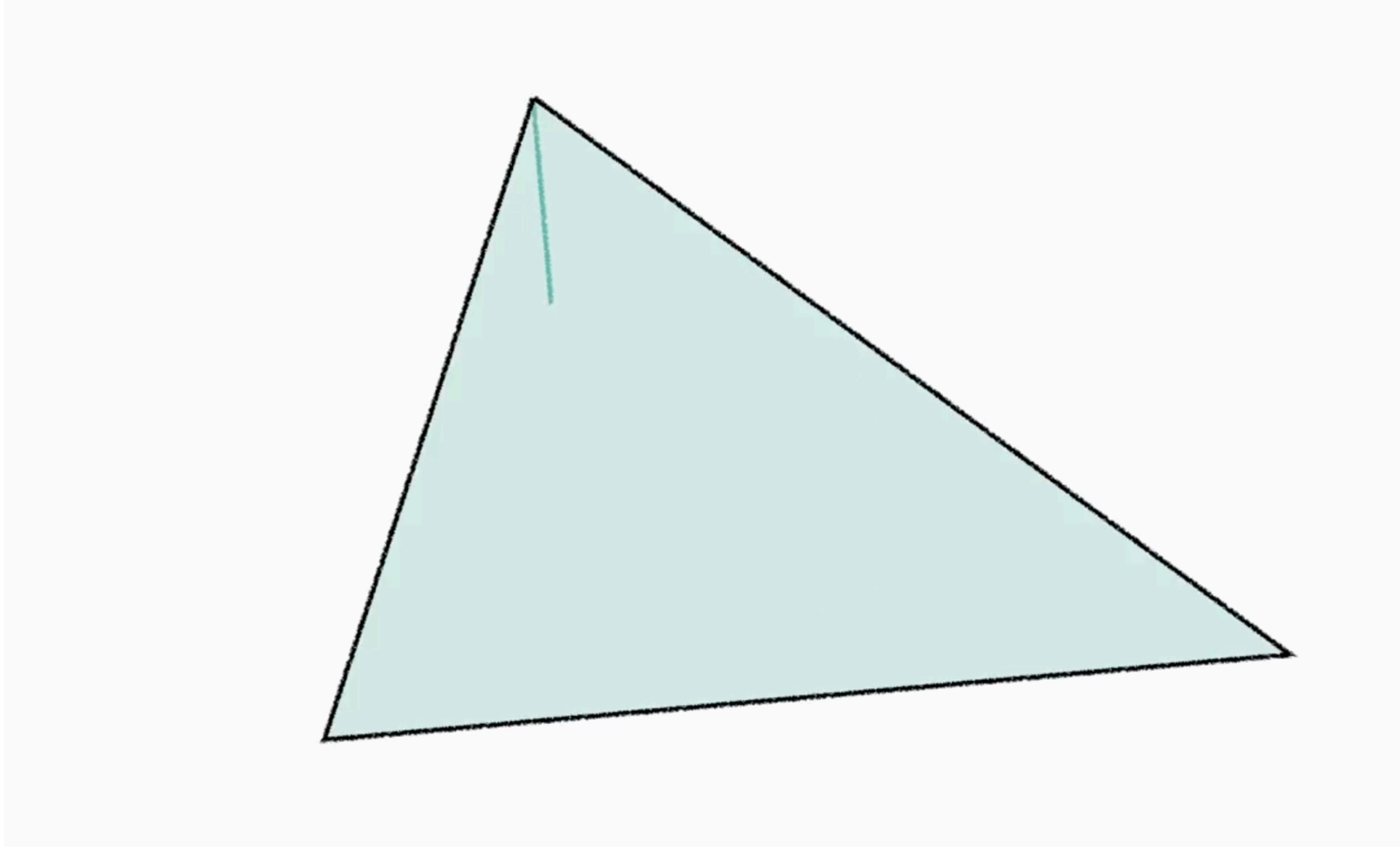
- Chapter 6



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

Curious Fact

For any triangle, the feet of the secondary altitudes lie on a circle.



Source