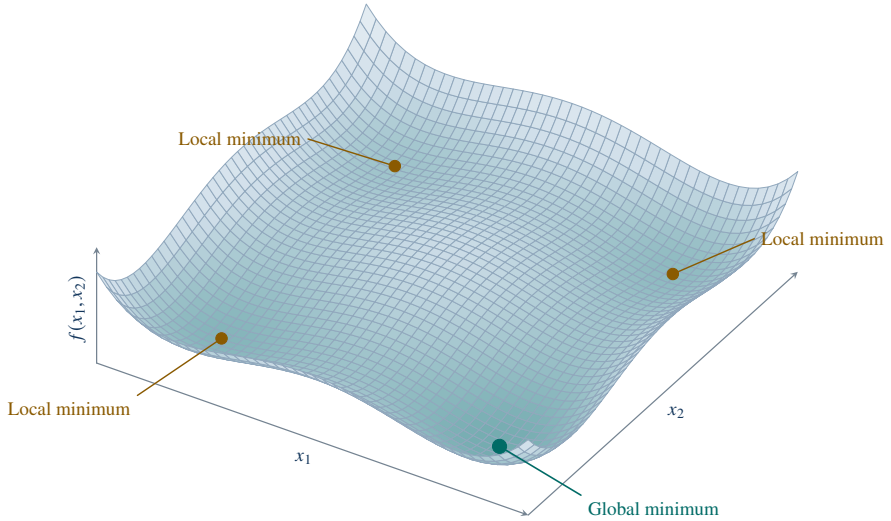
Lecture 11

Optimization, Part 2

Virginia Tech Department of Computer Science
Fall 2026

Minimizing a function

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



The gradient and Hessian provide local information about slope and curvature:

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} \partial f/\partial x_1 \\ \vdots \\ \partial f/\partial x_n \end{bmatrix}, \qquad H(\boldsymbol{x}) = \nabla^2 f(\boldsymbol{x}) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}(\boldsymbol{x}) \right)_{i,j=1}^{n}.$$

A stationary point satisfies $\nabla f(\boldsymbol{x}_*) = \boldsymbol{0}$. For twice continuously differentiable functions, the Hessian is symmetric.

Optimality tests

Suppose f is twice continuously differentiable near $\boldsymbol{x}_*$.

$$\text{Local minimum} \implies \nabla f(\boldsymbol{x}_*) = \mathbf{0}, \quad H(\boldsymbol{x}_*) \succeq 0.$$

$$\nabla f(\boldsymbol{x}_*) = \mathbf{0},\ H(\boldsymbol{x}_*) \succ 0 \implies \text{strict local minimum}.$$

$$\nabla f(\boldsymbol{x}_*) = \mathbf{0},\ H(\boldsymbol{x}_*) \prec 0 \implies \text{strict local maximum}.$$

An indefinite Hessian at a stationary point implies a saddle: every neighborhood contains both smaller and larger objective values.

In one variable: a local minimum has $f'(x_*) = 0$ and $f''(x_*) \geq 0$. When $f'(x_*) = 0$, a positive second derivative gives a strict minimum and a negative one gives a strict maximum.

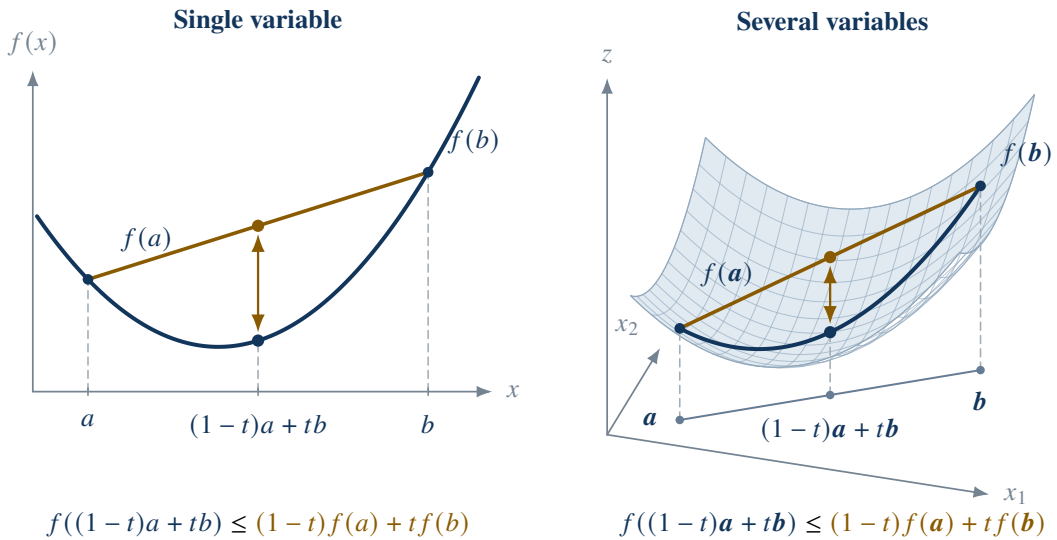
Convexity leads to stronger statements

Convexity means that the graph lies on or below every chord. For all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ and $0 \le t \le 1$,

$$f((1-t)\boldsymbol{x} + t\boldsymbol{y}) \le (1-t)f(\boldsymbol{x}) + tf(\boldsymbol{y})\,.$$

Strict convexity makes the inequality strict for distinct points and $0 < t < 1$.

Every local minimum of a convex function is global; in the differentiable case, $\nabla f(\boldsymbol{x}_*) = \boldsymbol{0}$ is equivalent to global optimality. Strict convexity gives at most one minimizer, but does not guarantee existence.



Every chord lies on or above the graph.

The inequality must hold for every pair of input points and every $t \in [0, 1]$.

Strong convexity

For a differentiable function, convexity is equivalent to $f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^{\mathsf{T}}(\boldsymbol{y} - \boldsymbol{x})$.

A differentiable function is μ-*strongly convex*, for a fixed $\mu > 0$, if for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$,

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^{\mathsf{T}}(\boldsymbol{y} - \boldsymbol{x}) + \frac{\mu}{2}\|\boldsymbol{y} - \boldsymbol{x}\|_2^2.$$

For twice continuously differentiable functions:

- Convexity is equivalent to a positive semidefinite Hessian everywhere;
- A positive definite Hessian everywhere suffices for strict convexity;
- μ-strong convexity is equivalent to every Hessian eigenvalue being at least μ.

A differentiable strongly convex function on $\mathbb{R}^n$ attains a unique global minimum, and

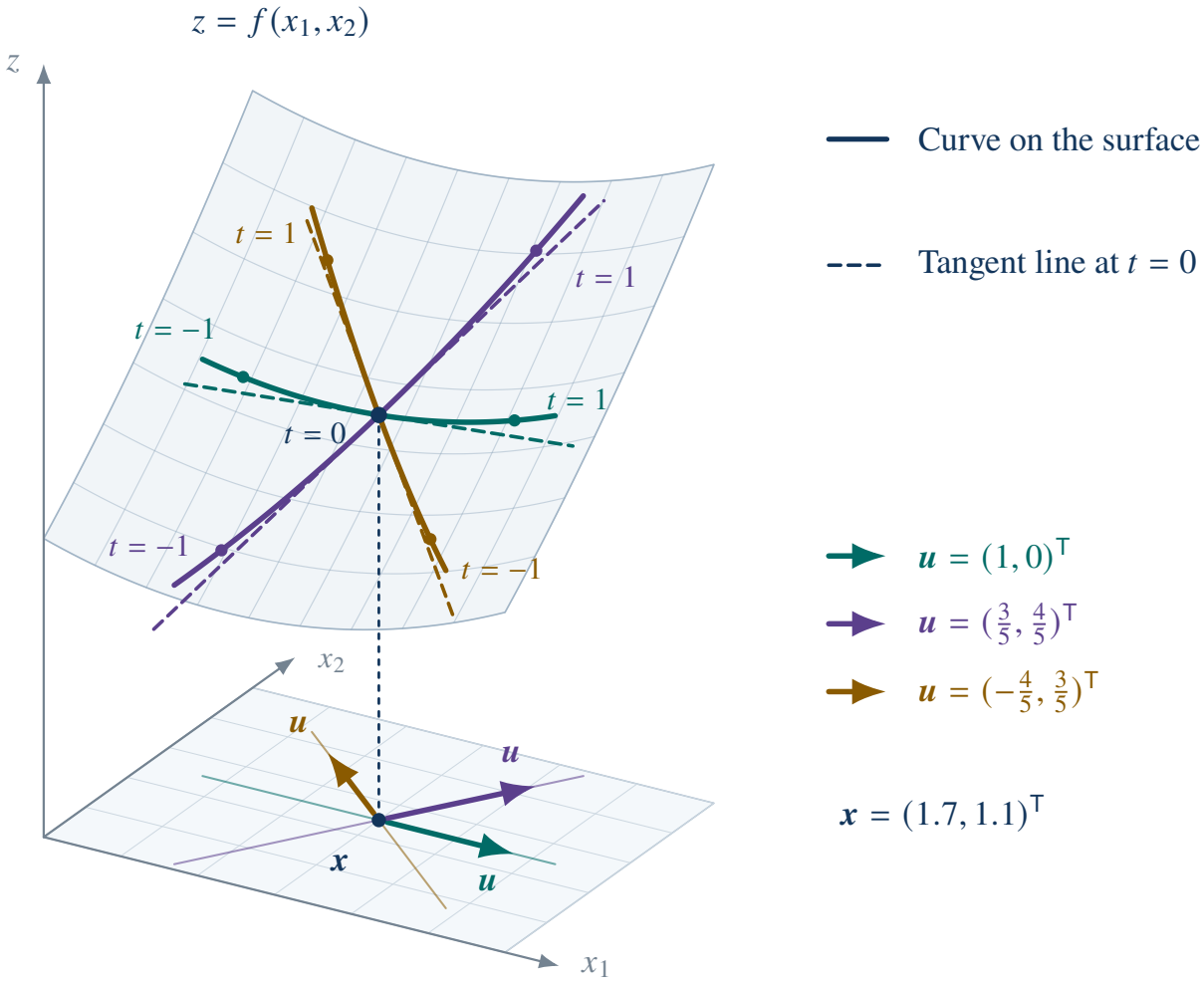
$$f(\boldsymbol{y}) - f(\boldsymbol{x}_*) \geq \frac{\mu}{2}\|\boldsymbol{y} - \boldsymbol{x}_*\|_2^2.$$

Gradient descent

The negative gradient points in the direction of steepest decrease. Gradient descent chooses a positive step length and updates as follows:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k \nabla f(\boldsymbol{x}_k)\,.$$

A large step may overshoot, while a small step may make little progress.



What gradient descent guarantees

Definition: An L-*Lipschitz* gradient satisfies, for all points,

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_2 \leq L\|\boldsymbol{x} - \boldsymbol{y}\|_2.$$

This is called L-smoothness. It limits how quickly slopes change; it does not require convexity.

For an L-smooth function, we have

$$f(\boldsymbol{x} + \boldsymbol{p}) \leq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^{\mathsf{T}}\boldsymbol{p} + \frac{L}{2}\|\boldsymbol{p}\|_2^2.$$

$$f(\boldsymbol{x} - \alpha\nabla f(\boldsymbol{x})) \leq f(\boldsymbol{x}) - \alpha\left(1 - \frac{L\alpha}{2}\right)\|\nabla f(\boldsymbol{x})\|_2^2.$$

Gradient descent: convergence rates

Thus $0 < \alpha < 2/L$ gives strict decrease when the gradient is nonzero.

With $\alpha = 1/L$, convexity and an attained minimum give

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_*) \leq \frac{L\|\boldsymbol{x}_0 - \boldsymbol{x}_*\|_2^2}{2k}, \qquad k \geq 1.$$

If also μ-strongly convex, then $\mu \leq L$, the minimizer is unique, the points converge to it, and

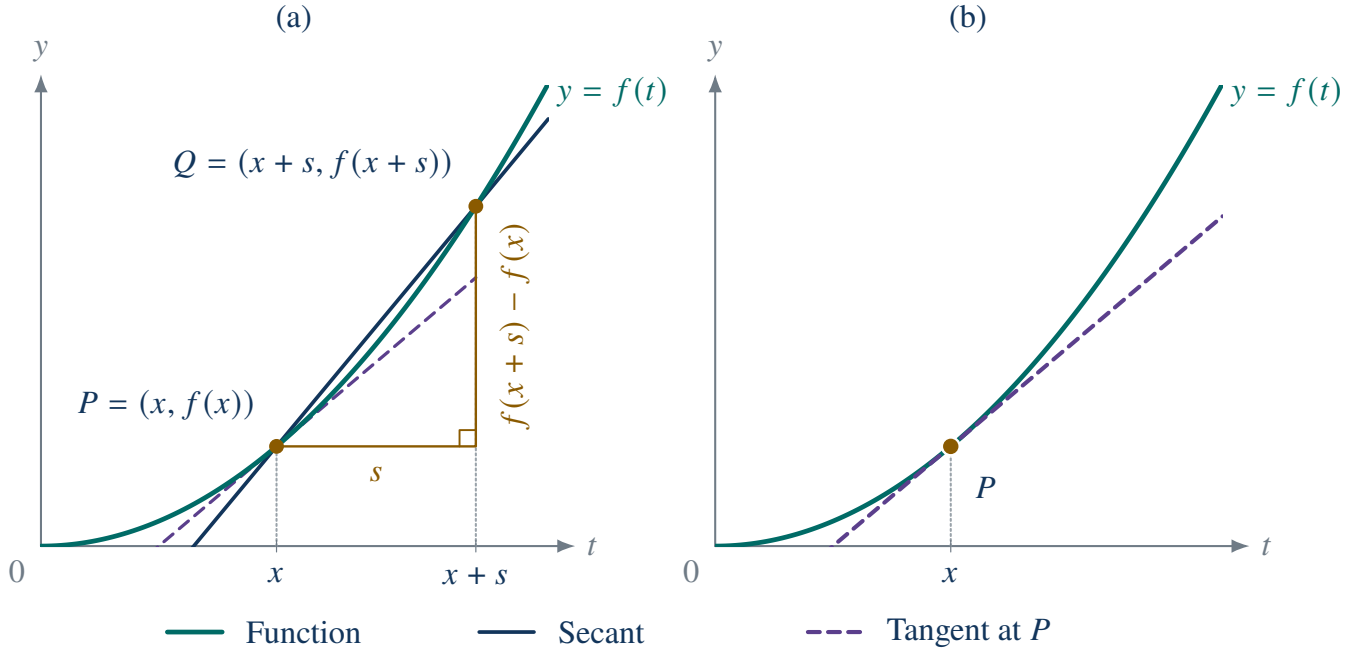
$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_*) \leq \left(1 - \frac{\mu}{L}\right)^k \big(f(\boldsymbol{x}_0) - f(\boldsymbol{x}_*)\big), \qquad k \geq 1.$$

How do we calculate the derivative numerically?

For a real-valued function defined on an open interval around x,

$$f'(x) = \lim_{s \to 0} \frac{f(x+s) - f(x)}{s}.$$

The quotient is the slope of the secant through two nearby graph points. Its limit is the slope of the tangent.



How do we calculate the derivative numerically?

With $h > 0$, the **forward and backward differences** use the point x and one neighbor:

$$D_h^+ f(x) = \frac{f(x+h) - f(x)}{h}, \qquad D_h^- f(x) = \frac{f(x) - f(x-h)}{h}.$$

The **central difference** averages them and uses the two surrounding values:

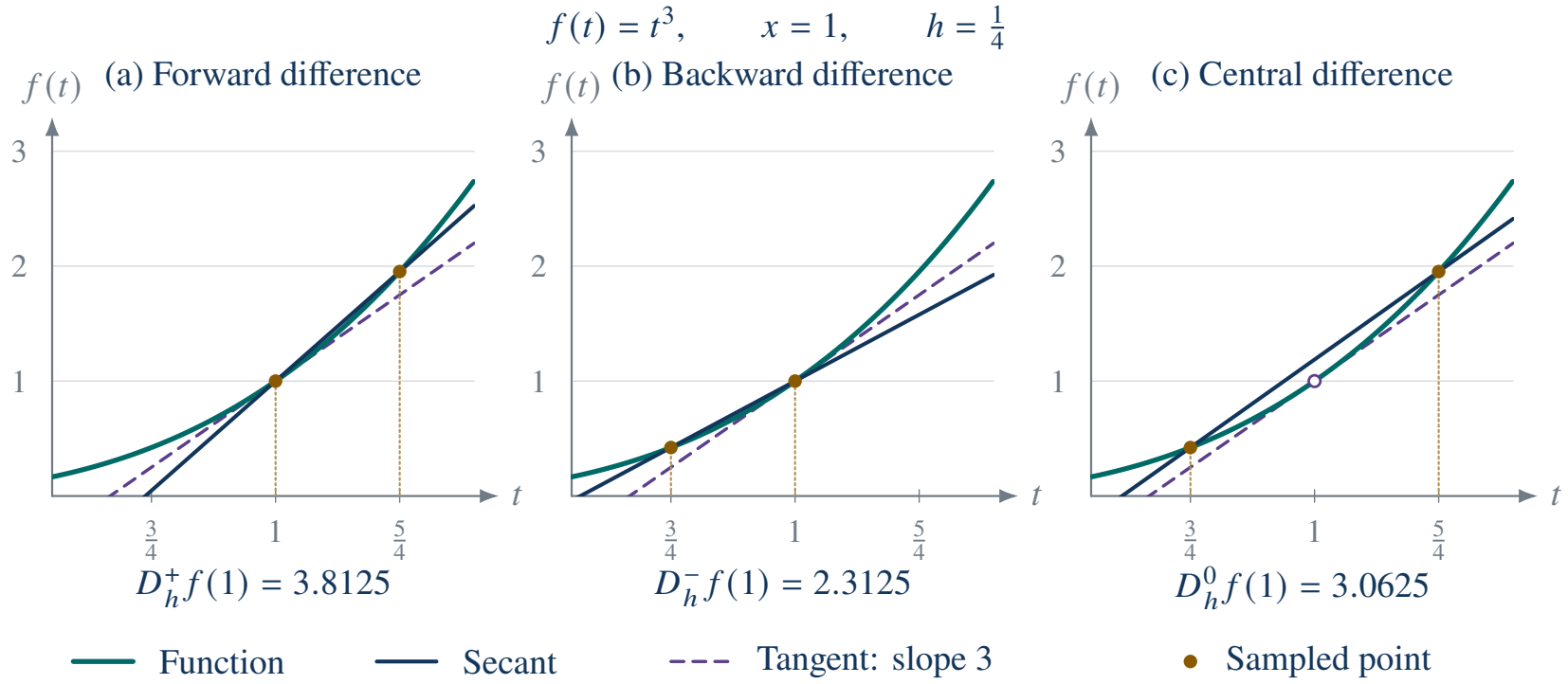
$$D_h^0 f(x) = \frac{D_h^+ f(x) + D_h^- f(x)}{2} = \frac{f(x+h) - f(x-h)}{2h}.$$

All three tend to $f'(x)$ when f is differentiable. With additional smoothness, the symmetric formula is more accurate as h becomes small. One-sided formulas remain useful at a boundary.

Example: central difference improves the estimate

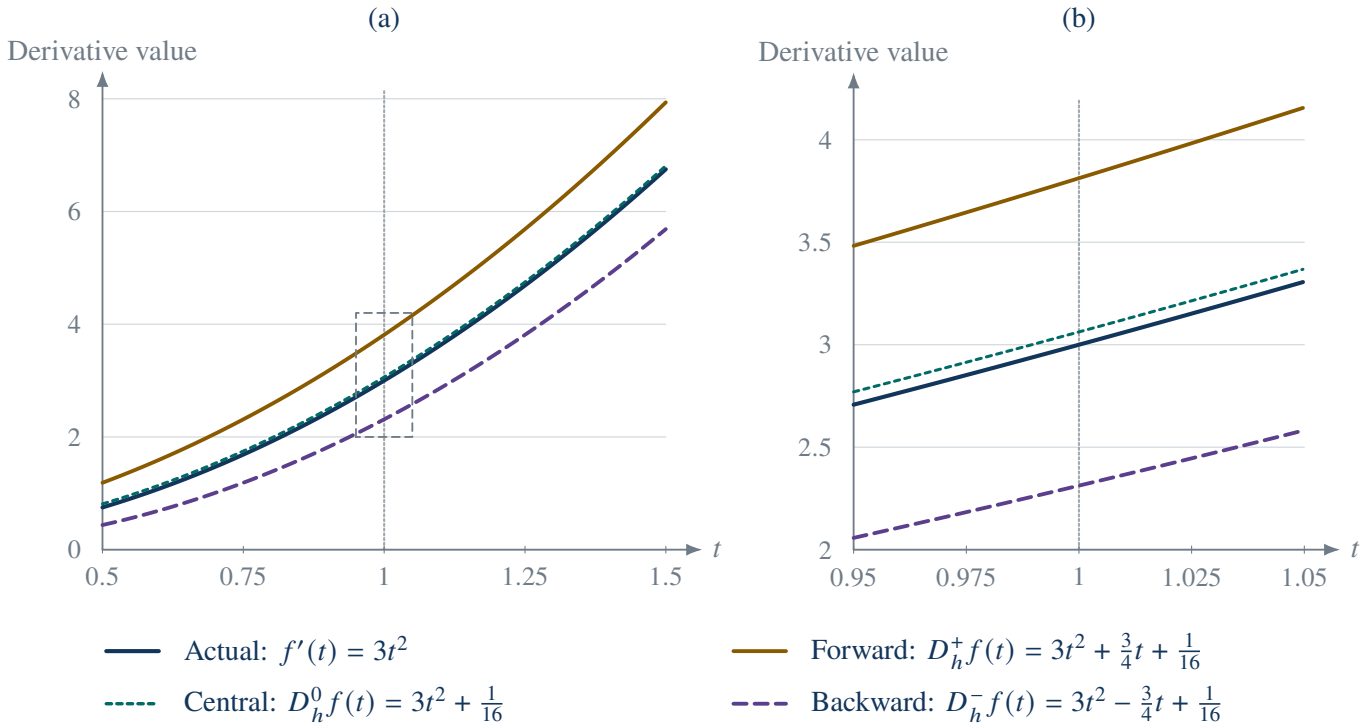
For $f(x) = x^3$ at $x = 1$, the exact derivative is $f'(1) = 3$.

$$D_h^+ f(1) = 3 + 3h + h^2, \qquad D_h^- f(1) = 3 - 3h + h^2, \qquad D_h^0 f(1) = 3 + h^2 .$$



Example: central difference improves the estimate

$f(t) = t^3, \qquad h = \frac{1}{4}$



h	Forward	Backward	Central
0.1	0.31	0.29	0.01
0.05	0.1525	0.1475	0.0025
0.025	0.075625	0.074375	0.000625

Error bounds

Fix $h_0 > 0$, let $I = [x - 2h_0, x + 2h_0]$, and take $0 < h \le h_0$. Write $f^{(j)}$ for the jth derivative and define

$$M_j = \max_{t \in I} |f^{(j)}(t)| .$$

$$f \in C^2(I): \quad |D_h^+ f(x) - f'(x)| \le \frac{M_2 h}{2}, \quad |D_h^- f(x) - f'(x)| \le \frac{M_2 h}{2},$$

$$f \in C^3(I): \quad |D_h^0 f(x) - f'(x)| \le \frac{M_3 h^2}{6} .$$

The one-sided errors are $O(h)$; the central error is $O(h^2)$. Halving h halves the first bounds and quarters the second.

Numerical second derivatives

A second derivative measures how the first derivative changes. Apply first differences twice:

$$D^+_{2,h} f(x) = \frac{D^+_h f(x+h) - D^+_h f(x)}{h} = \frac{f(x+2h) - 2f(x+h) + f(x)}{h^2},$$
$$D^-_{2,h} f(x) = \frac{D^-_h f(x) - D^-_h f(x-h)}{h} = \frac{f(x) - 2f(x-h) + f(x-2h)}{h^2},$$
$$D^0_{2,h} f(x) = \frac{D^+_h f(x) - D^-_h f(x)}{h} = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}.$$

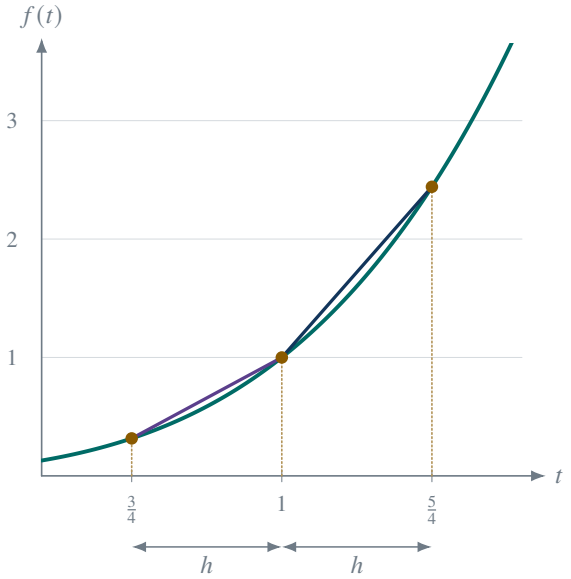
(The subscript 2 denotes a second derivative.)

Example: a central second difference

$$f(x) = x^4, \quad f''(1) = 12, \quad D^0_{2,h} f(1) = 12 + 2h^2 .$$

$f(t) = t^4, \qquad x = 1, \qquad f''(1) = 12$

(a) Compare two adjacent slopes, $h = \frac{1}{4}$



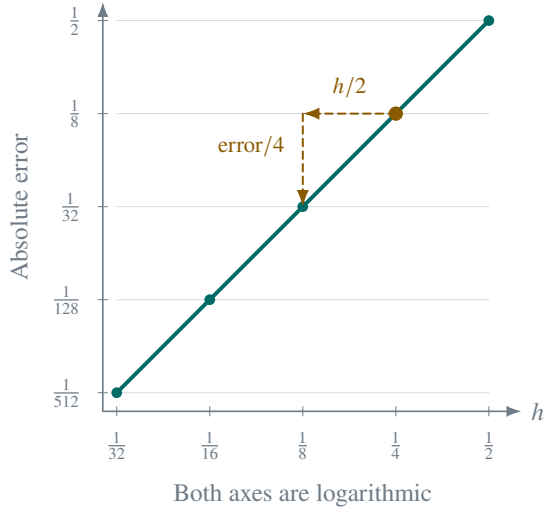
$D^-_h f(1) = 2.734375$

$D^+_h f(1) = 5.765625$

$$D^0_{2,h} f(1) = \frac{5.765625 - 2.734375}{1/4} = 12.125$$

(b) Second-order error

$|D^0_{2,h} f(1) - f''(1)| = 2h^2$



Halving h divides the error by 4.

Numerical second-derivative error bounds

Use the same fixed interval $I = [x - 2h_0, x + 2h_0]$, $0 < h \le h_0$, and bounds $M_j = \max_{t \in I} |f^{(j)}(t)|$.

If $f \in C^3(I)$, then

$$|D^+_{2,h} f(x) - f''(x)| \le M_3 h, \qquad |D^-_{2,h} f(x) - f''(x)| \le M_3 h .$$

If $f \in C^4(I)$, then

$$|D^0_{2,h} f(x) - f''(x)| \le \frac{M_4 h^2}{12} .$$

The one-sided formulas have first-order error; the central formula has second-order error. With exact function values and arithmetic, each error tends to zero as $h \to 0$.

Why an arbitrarily small step can be worse

Suppose the available values are $\tilde{f}(t) = f(t) + e(t)$ with $|e(t)| \le \delta$, and otherwise treat the arithmetic as exact. Then

$$|D_h^{\pm}\tilde{f}(x) - f'(x)| \le \frac{M_2 h}{2} + \frac{2\delta}{h},$$

$$|D_h^{0}\tilde{f}(x) - f'(x)| \le \frac{M_3 h^2}{6} + \frac{\delta}{h},$$

$$|D_{2,h}^{0}\tilde{f}(x) - f''(x)| \le \frac{M_4 h^2}{12} + \frac{4\delta}{h^2}.$$

Reducing h decreases truncation error but amplifies evaluation error.

Balance the terms: $h \propto \delta^{1/2}$ (one-sided) or $h \propto \delta^{1/3}$ (central).

Numerical gradients: vary one coordinate at a time

Let $\boldsymbol{e}_j$ be the jth coordinate vector. Hold the other coordinates fixed:

$$\frac{\partial f}{\partial x_j}(\boldsymbol{x}) = \lim_{t \to 0} \frac{f(\boldsymbol{x} + t\boldsymbol{e}_j) - f(\boldsymbol{x})}{t}.$$

$$\begin{aligned}
\frac{\partial f}{\partial x_j}(\boldsymbol{x}) &\approx \frac{f(\boldsymbol{x} + h_j\boldsymbol{e}_j) - f(\boldsymbol{x})}{h_j}, \\
&\approx \frac{f(\boldsymbol{x}) - f(\boldsymbol{x} - h_j\boldsymbol{e}_j)}{h_j}, \\
&\approx \frac{f(\boldsymbol{x} + h_j\boldsymbol{e}_j) - f(\boldsymbol{x} - h_j\boldsymbol{e}_j)}{2h_j}.
\end{aligned}$$

Hessian entries: curvature and changing slopes

$$H(\boldsymbol{x}) = \nabla^2 f(\boldsymbol{x}), \qquad H_{ij}(\boldsymbol{x}) = \frac{\partial^2 f}{\partial x_i \partial x_j}(\boldsymbol{x})\,.$$

A diagonal entry measures curvature along one coordinate. An off-diagonal entry measures how the slope in coordinate j changes as coordinate i varies. For $f \in C^2$, the Hessian is symmetric.

Using a common step $h > 0$, approximate a diagonal entry by

$$(\widehat{H}_h)_{ii}(\boldsymbol{x}) = \frac{f(\boldsymbol{x} + h\boldsymbol{e}_i) - 2f(\boldsymbol{x}) + f(\boldsymbol{x} - h\boldsymbol{e}_i)}{h^2}\,.$$

This is the central second difference of the one-variable function $t \mapsto f(\boldsymbol{x} + t\boldsymbol{e}_i)$.

Off-diagonal Hessian entries

Estimate the jth partial derivative at $\boldsymbol{x} + h\boldsymbol{e}_i$ and $\boldsymbol{x} - h\boldsymbol{e}_i$, then compare those two slopes:

$$(\widehat{H}_h)_{ij}(\boldsymbol{x}) = \frac{1}{4h^2}\begin{pmatrix} f(\boldsymbol{x} + h\boldsymbol{e}_i + h\boldsymbol{e}_j) - f(\boldsymbol{x} + h\boldsymbol{e}_i - h\boldsymbol{e}_j) \\ -f(\boldsymbol{x} - h\boldsymbol{e}_i + h\boldsymbol{e}_j) + f(\boldsymbol{x} - h\boldsymbol{e}_i - h\boldsymbol{e}_j) \end{pmatrix}, \quad i \neq j\,.$$

The samples lie at the corners of a square in the i, j coordinate plane. Opposite corners have the same sign; the signs compare the slope on opposite sides.

Compute each mixed entry once and set $(\widehat{H}_h)_{ji} = (\widehat{H}_h)_{ij}$ to preserve symmetry.

Hessian accuracy and evaluation cost

Suppose $f \in C^4$ on a fixed closed box around $\boldsymbol{x}$. Let L_4 bound all fourth-order partial derivatives in absolute value, and choose h so every sample stays inside the box. Then

$$\left|(\widehat{H}_h)_{ii} - H_{ii}\right| \le \frac{L_4 h^2}{12}, \qquad \left|(\widehat{H}_h)_{ij} - H_{ij}\right| \le \frac{L_4 h^2}{3} \quad (i \neq j).$$

Every entry has $O(h^2)$ error. For fixed n, the entire approximate Hessian converges in exact arithmetic.

Function-value errors bounded by δ add at most $4\delta/h^2$ to each diagonal error and δ/h^2 to each mixed-entry error.

Add curvature information to gradient descent

Standard gradient descent does not take curvature into account! Only the gradient.

Newton's method uses the Hessian to adjust direction and scale.

BFGS learns curvature from gradient changes, avoiding a new Hessian calculation at every step.

$$\boldsymbol{p}_k = -H_k \nabla f(\boldsymbol{x}_k), \qquad \boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k \,.$$

Here H_k approximates the *inverse Hessian*. With $H_k = I$, we recover gradient descent.

Pick α_k such that $f(\boldsymbol{x}_k + \alpha_k \boldsymbol{p}_k) < f(\boldsymbol{x}_k)$.

(For example, start with $\alpha = 1$ and repeatedly half until the condition is satisfied.)

BFGS update rule

$$\boldsymbol{s}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k, \qquad \boldsymbol{y}_k = \nabla f(\boldsymbol{x}_{k+1}) - \nabla f(\boldsymbol{x}_k)\,.$$

In one variable, s_k/y_k estimates inverse curvature. In several variables, the new matrix should map the observed gradient change back to the displacement:

$$H_{k+1}\boldsymbol{y}_k = \boldsymbol{s}_k \qquad \textit{(the secant equation)}\,.$$

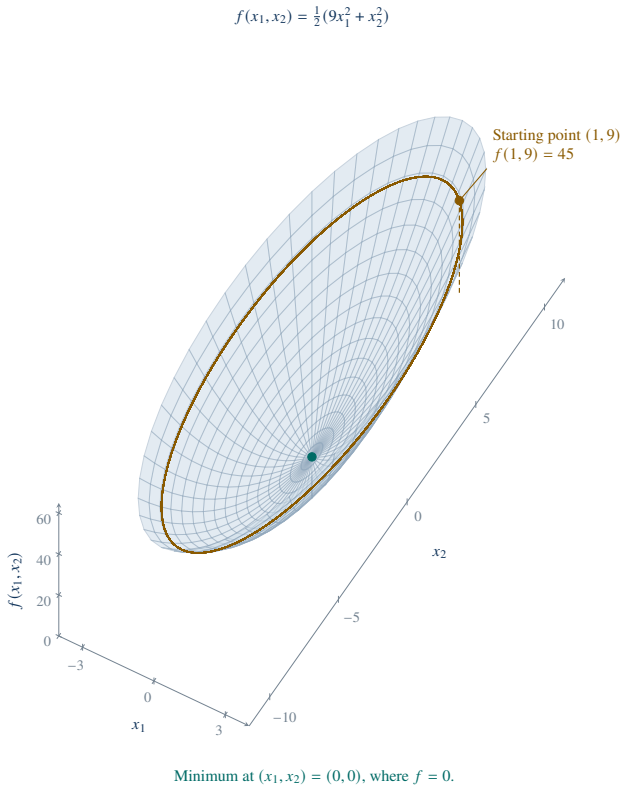
BFGS combines this new information with the previous approximation:

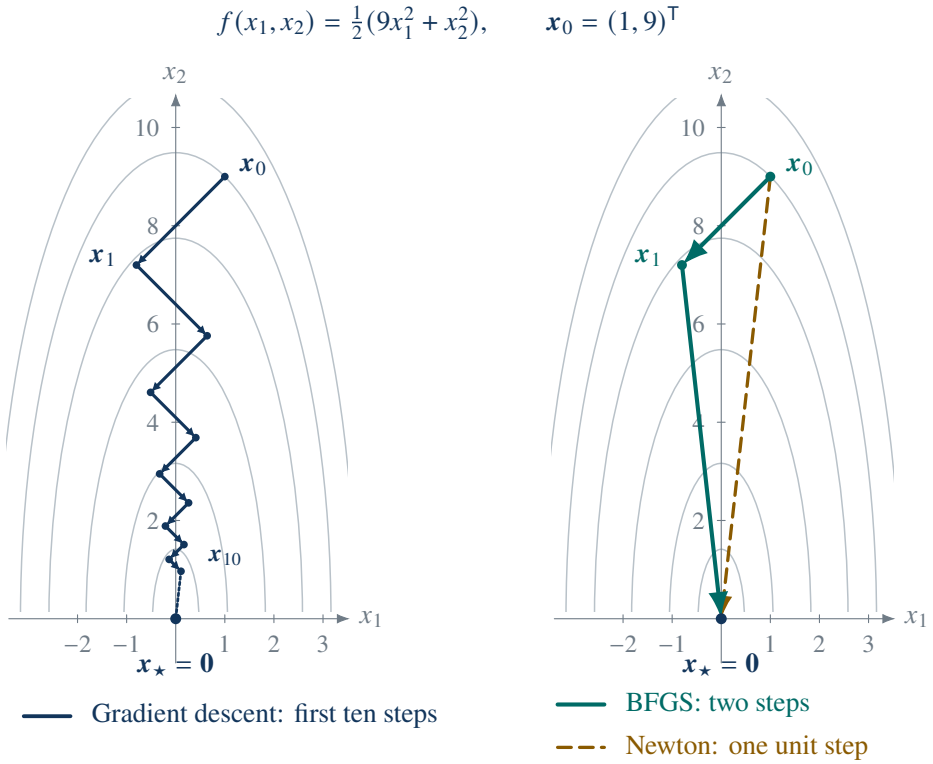
$$\rho_k = \frac{1}{\boldsymbol{y}_k^{\mathsf{T}}\boldsymbol{s}_k},$$

$$H_{k+1} = (\mathbb{1} - \rho_k \boldsymbol{s}_k \boldsymbol{y}_k^{\mathsf{T}}) H_k (\mathbb{1} - \rho_k \boldsymbol{y}_k \boldsymbol{s}_k^{\mathsf{T}}) + \rho_k \boldsymbol{s}_k \boldsymbol{s}_k^{\mathsf{T}}\,.$$

Example: curvature information improves convergence

$$f(x_1, x_2) = \frac{1}{2}(9x_1^2 + x_2^2), \quad \nabla f(\boldsymbol{x}) = \begin{bmatrix} 9x_1 \\ x_2 \end{bmatrix}, \quad A = \nabla^2 f(\boldsymbol{x}) = \begin{bmatrix} 9 & 0 \\ 0 & 1 \end{bmatrix}.$$



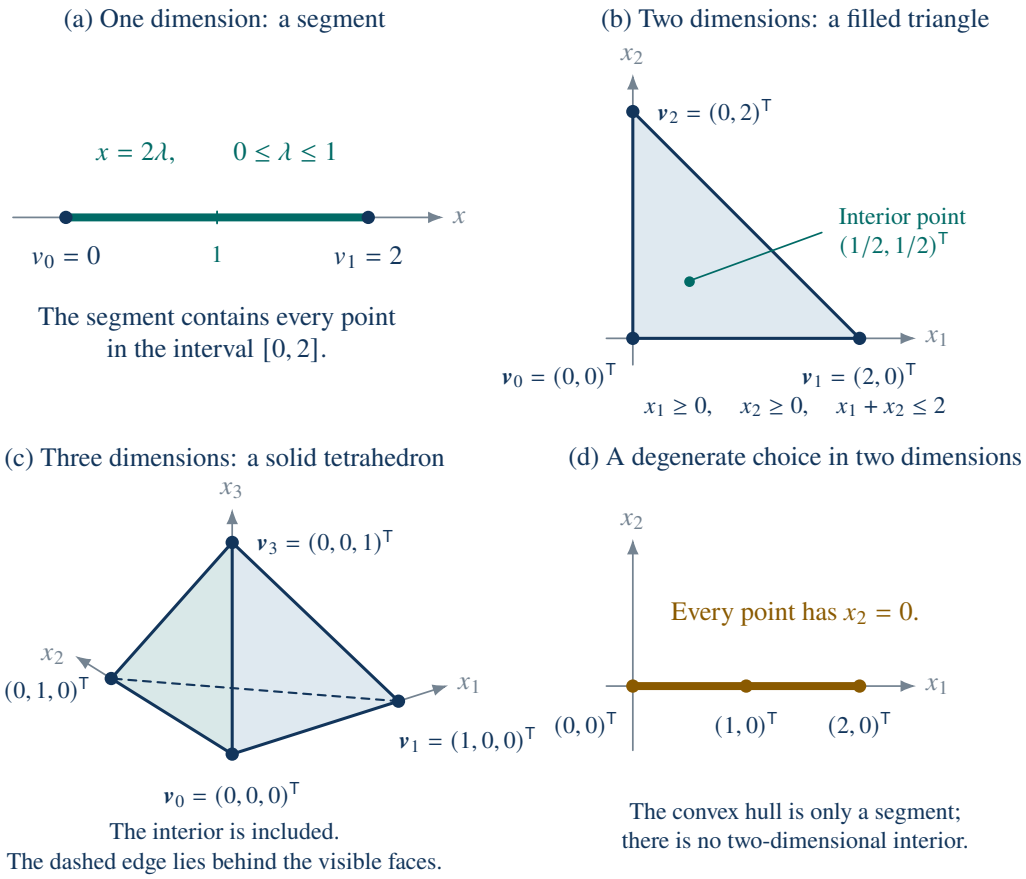


Nelder–Mead: optimization without derivatives!

Keep a simplex of $n+1$ sampled points. Compare their objective values, then move the worst vertex or reduce the simplex. No gradient or Hessian is required.

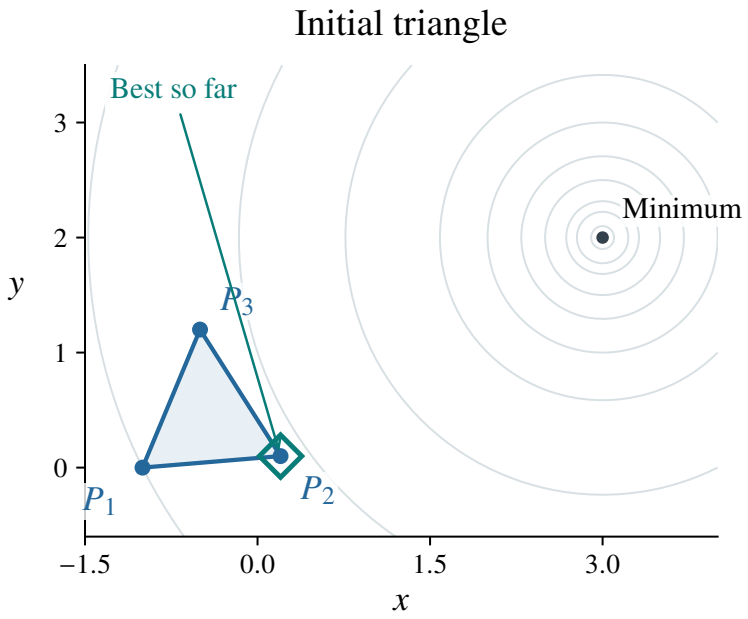
Definition: Vertices $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_n \in \mathbb{R}^n$ are affinely independent when $\boldsymbol{v}_1 - \boldsymbol{v}_0, \ldots, \boldsymbol{v}_n - \boldsymbol{v}_0$ are linearly independent. Their simplex is

$$\mathcal{S} = \left\{ \sum_{i=0}^{n} \lambda_i \boldsymbol{v}_i : \ \lambda_i \geq 0, \ \sum_{i=0}^{n} \lambda_i = 1 \right\}.$$



Nelder–Mead example

$$f(x,y) = (x-3)^2 + (y-2)^2, \qquad (x_*, y_*) = (3,2)\,.$$



Vertex	Coordinates	f	Initial rank
P_1	$(-1,0)$	20	W
P_2	$(0.2,0.1)$	11.45	B
P_3	$(-0.5,1.2)$	12.89	S

P_1, P_2, P_3 identify slots in the triangle. The ranking is recomputed each iteration: B is best, S is second-best, and W is worst.

The triangle starts far from the minimum. Its vertices are chosen by the user.

Vertices and rankings

Vertex	Coordinates	f	Initial rank
P_1	$(-1,0)$	20	W
P_2	$(0.2,0.1)$	11.45	B
P_3	$(-0.5,1.2)$	12.89	S

P_1, P_2, P_3 identify slots in the triangle. The ranking is recomputed each iteration: B is best, S is second-best, and W is worst.

$$f(B) \le f(S) \le f(W), \qquad M = \frac{B + S}{2}.$$

M is the midpoint of the two better vertices.

Three new trial points

First form the direction $d = M - W$. This points from the worst vertex toward the middle of the better pair. It is a geometric guess, based on sampled values.

$$R = M + d = 2M - W, \qquad E = M + 2d\,.$$

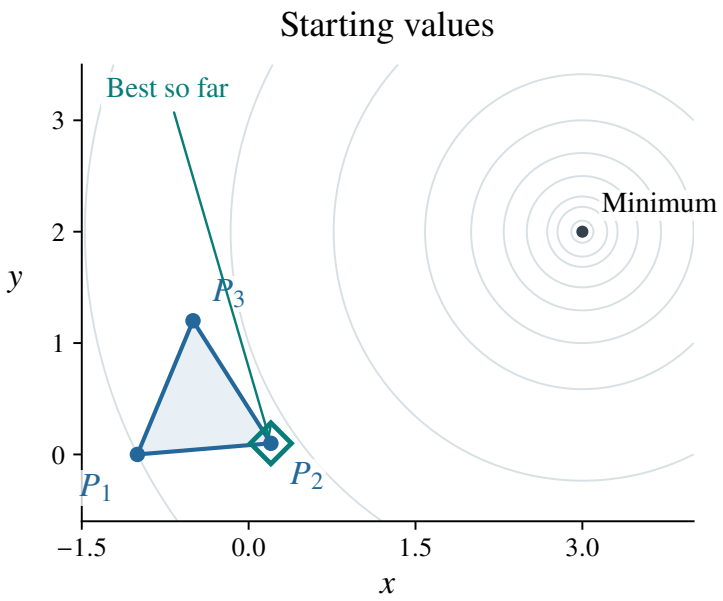
$$C_{\text{out}} = M + \frac{1}{2}d, \qquad C_{\text{in}} = M - \frac{1}{2}d\,.$$

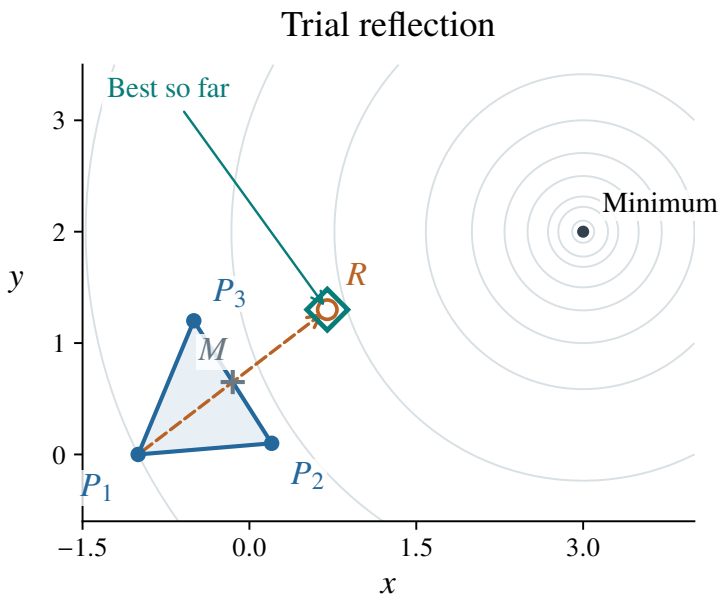
R, E, and C denote a *reflection*, *expansion*, and *contraction* trial. The stage identifies which contraction is being tried.

Evaluate $f(R)$ first. Its value determines whether to accept R, evaluate E, or evaluate one of the contractions. The method does not test every trial on every iteration.

Iteration 1: reflection

$$M = (-0.15, 0.65), \qquad R = 2M - W = (0.7, 1.3)\,.$$



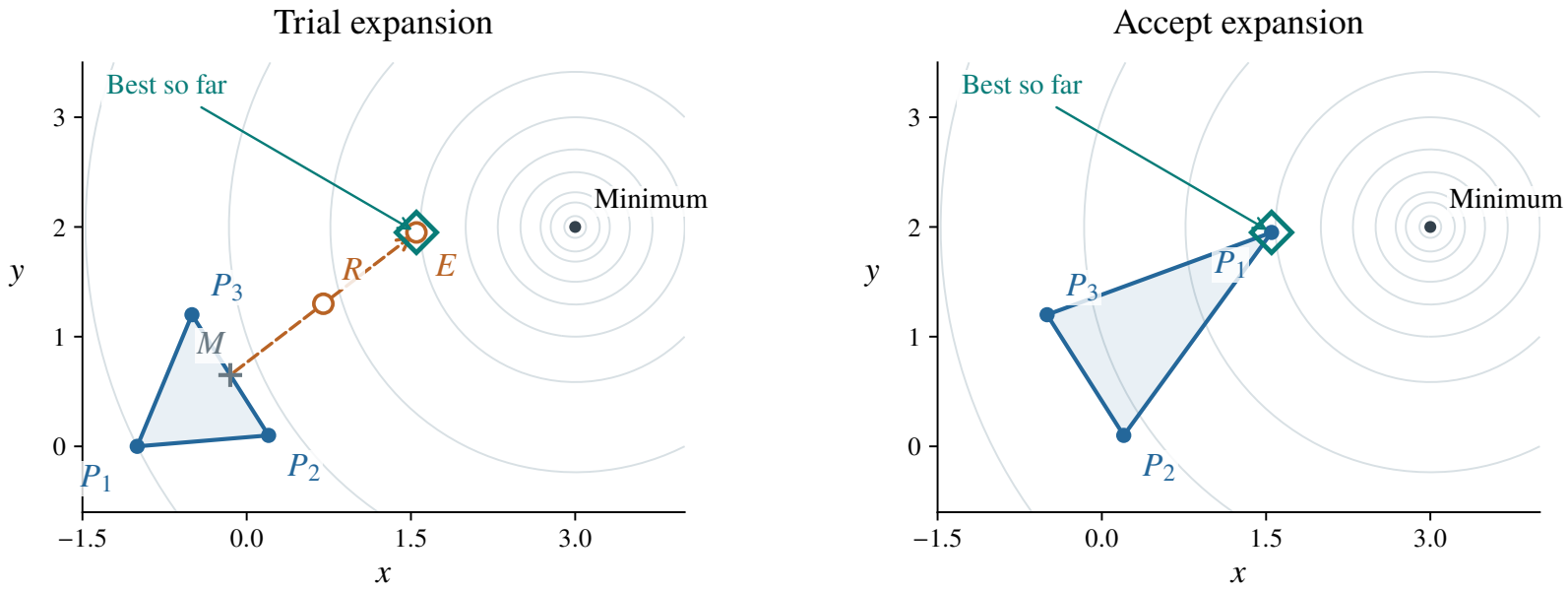


$$f(R) = 5.78 < 11.45 = f(B)\,.$$

Next: test expansion.

Iteration 1: expansion

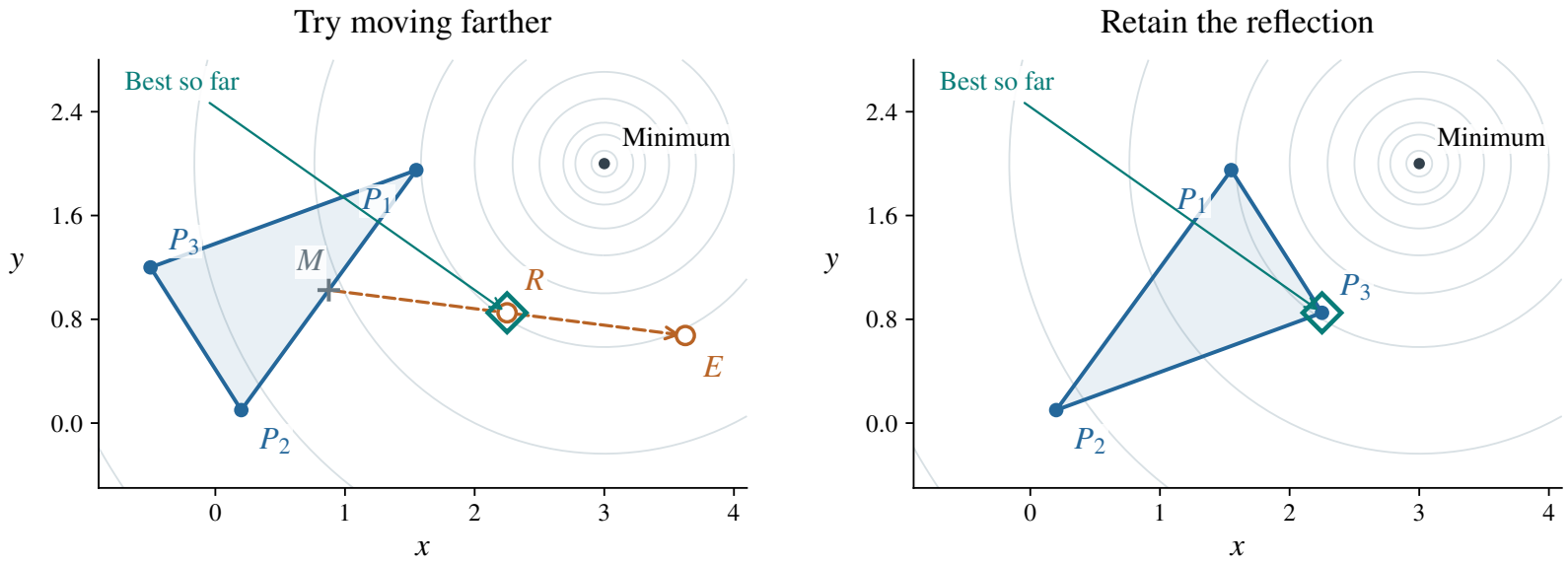
$$E = M + 2(R - M) = (1.55, 1.95), \qquad f(E) = 2.105.$$



Because $f(E) < f(R)$, replace only the worst slot P_1 by E. The other vertices stay fixed. The new best is P_1; the next worst is P_3.

Iteration 2: reject expansion, keep reflection

$$M = (0.875, 1.025), \qquad R = (2.25, 0.85), \qquad f(R) = 1.885.$$

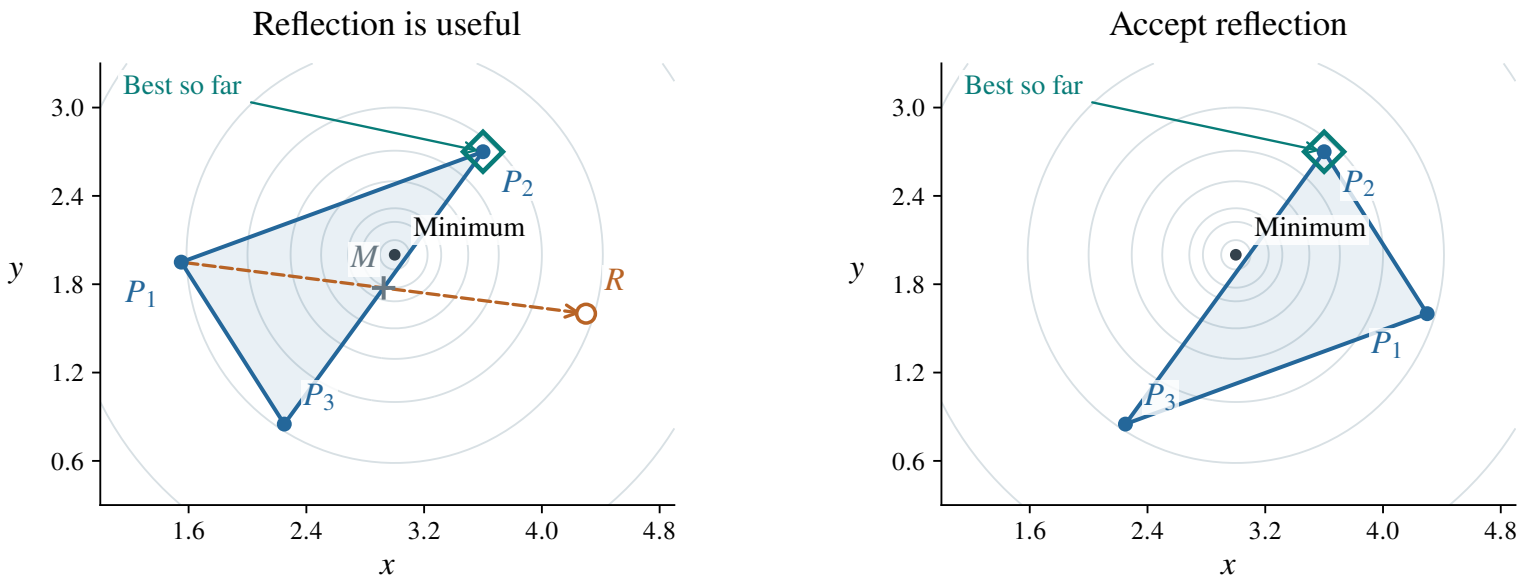


$$E = (3.625, 0.675), \qquad f(E) = 2.14625 > f(R)\,.$$

Reflection improves on the old best, but going farther does not help. Replace the old worst vertex by R.

Iteration 4: accept reflection directly

$$f(B) = 0.85, \qquad f(S) = 1.885, \qquad f(W) = 2.105.$$

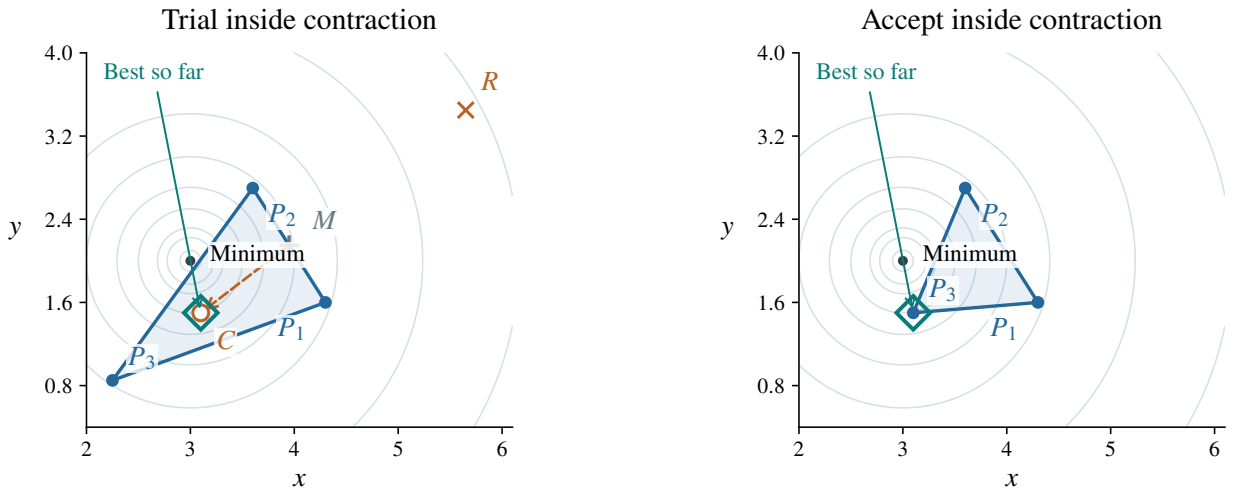


$$R = (4.3, 1.6), \qquad 0.85 \le f(R) = 1.85 < 1.885.$$

The reflection is better than the second-best old vertex. Replace W by R immediately; one new function evaluation is enough.

Iteration 5: inside contraction

$$f(R) = 9.125 \geq 1.885 = f(W), \qquad M = (3.95, 2.15)\,.$$

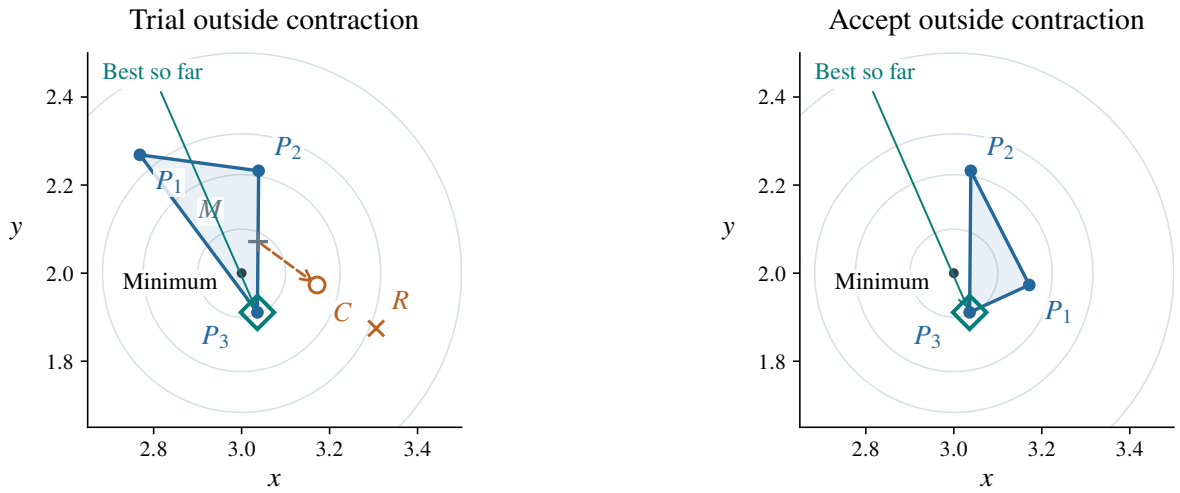


$$C_{\text{in}} = M + \frac{1}{2}(W - M) = (3.1, 1.5), \qquad f(C_{\text{in}}) = 0.26.$$

Reflection is no better than the worst vertex. Test a point halfway back toward W. Accept it because $0.26 < 1.885$.

Iteration 11: outside contraction

$$f(S) \approx 0.055515 \le f(R) \approx 0.109273 < f(W) \approx 0.125703.$$



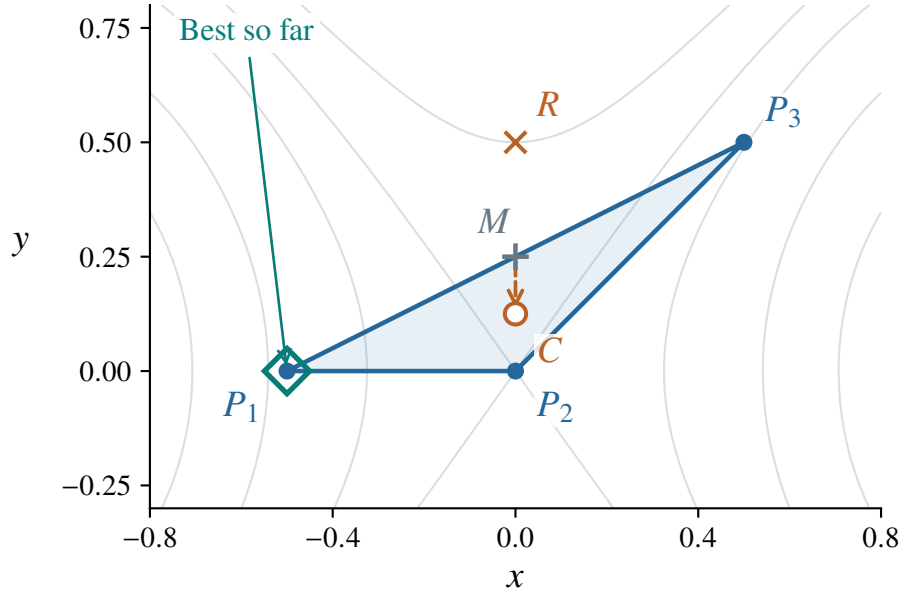
$$C_{\text{out}} = M + \frac{1}{2}(R - M) \approx (3.171582, 1.973145),$$

The reflection improves on the worst value, but not the second-best. Its outside contraction has $f(C_{\text{out}}) \approx 0.030162 \le f(R)$, so it is accepted.

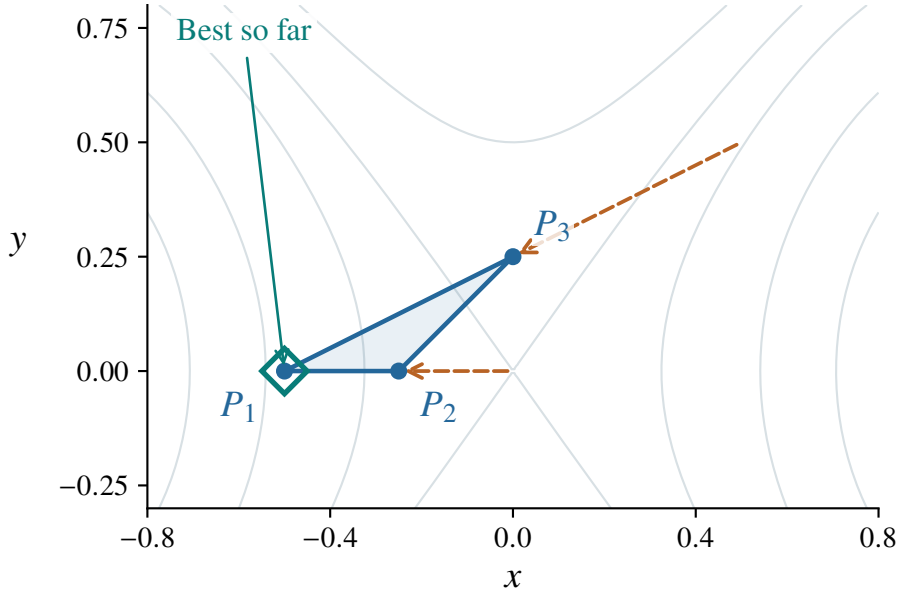
When contraction fails: shrink

$$B_{\text{new}} = B, \quad S_{\text{new}} = B + \frac{1}{2}(S - B), \quad W_{\text{new}} = B + \frac{1}{2}(W - B)\,.$$

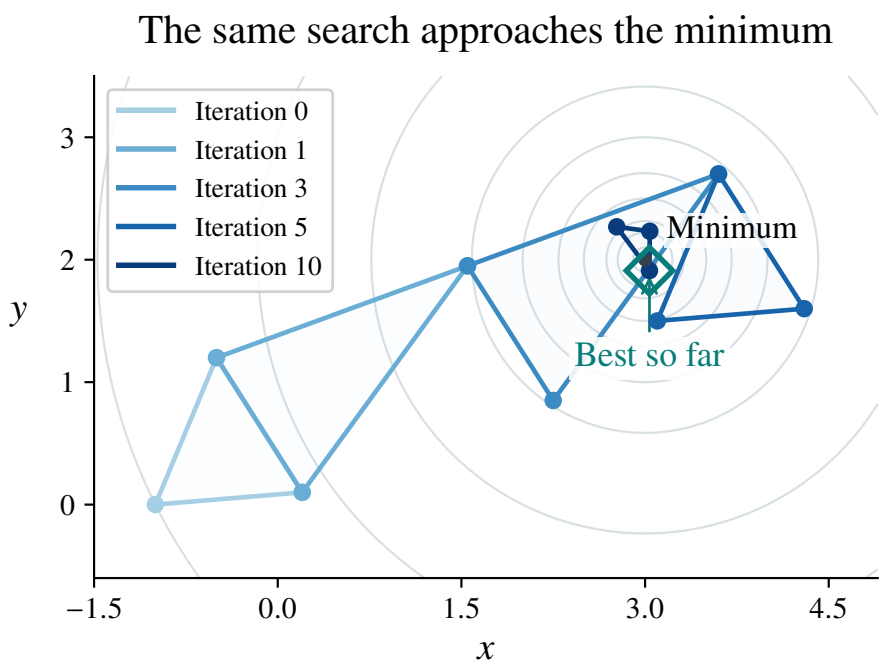
Reflection and contraction fail

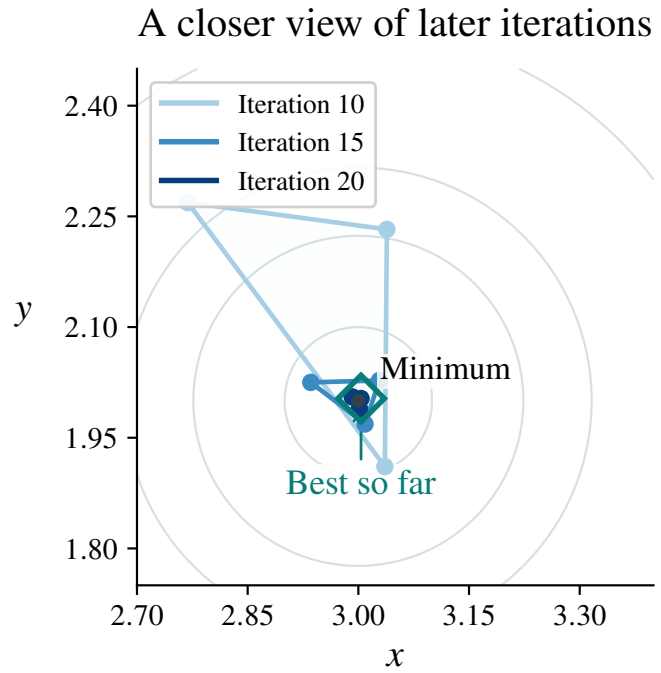


Shrink toward the best vertex



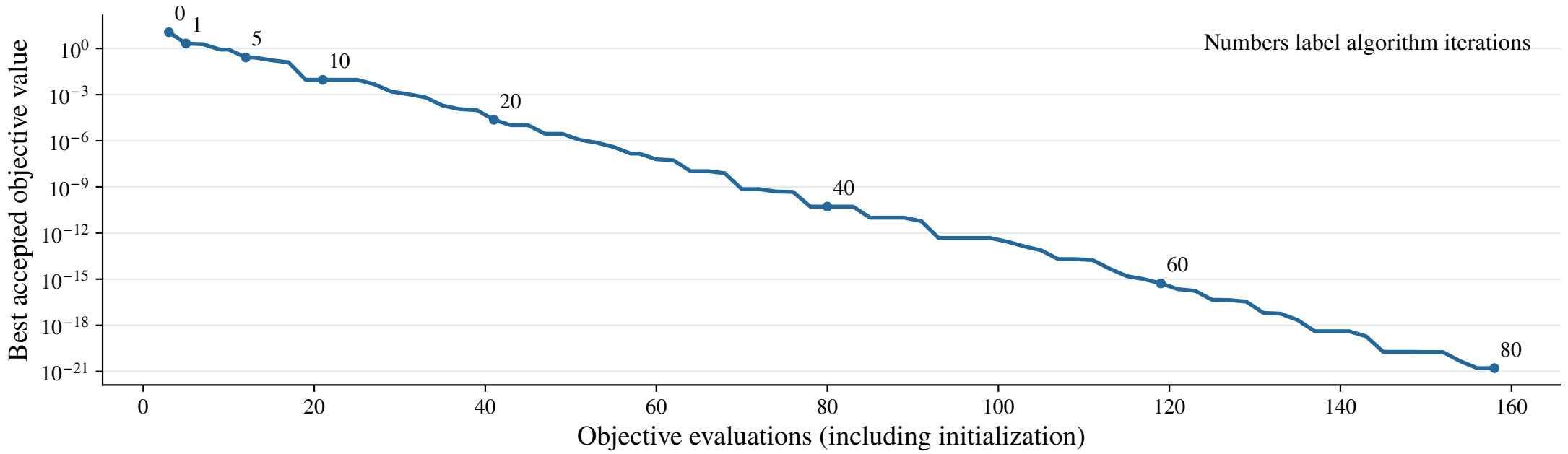
Follow the same triangle toward the minimum





The triangle changes location, size, and shape. The two panels show the same run at different scales. The diamond marks the best evaluated point at the latest iteration in each panel.

The complete run: 80 iterations



There are 158 objective evaluations and 237 animation frames. The accepted moves are 10 reflections, 1 expansion, 10 outside contractions, and 59 inside contractions.

A frame is a trial or an accepted update. It is not an algorithm iteration. The curve includes every accepted triangle through iteration 80.

Simulated annealing: allowing uphill moves

A method that always decreases the objective can remain near a local minimum even when a lower minimum exists elsewhere. Randomized search sometimes permits an increase to explore another region.

The current point generates the next proposal and may become worse. The best point found is stored separately and retained for the answer.

$$\Delta = f(\boldsymbol{y}) - f(\boldsymbol{x}), \qquad A_T(\boldsymbol{x}, \boldsymbol{y}) = \begin{cases} 1, & \Delta \le 0 \quad \text{(improving or equal)}, \\ e^{-\Delta/T}, & \Delta > 0 \quad \text{(uphill)}. \end{cases}$$

Metropolis acceptance always accepts a non-increasing proposal. An uphill proposal is accepted with probability $\exp(-\Delta/T)$.

Temperature and proposal size

$$T_j = T_0 r^j, \qquad T_0 = -\frac{\Delta_{\text{typ}}}{\log a}.$$

Choose T_0 by requiring a typical uphill increase $\Delta_{\text{typ}} > 0$ to be accepted initially with probability $a \in (0,1)$.

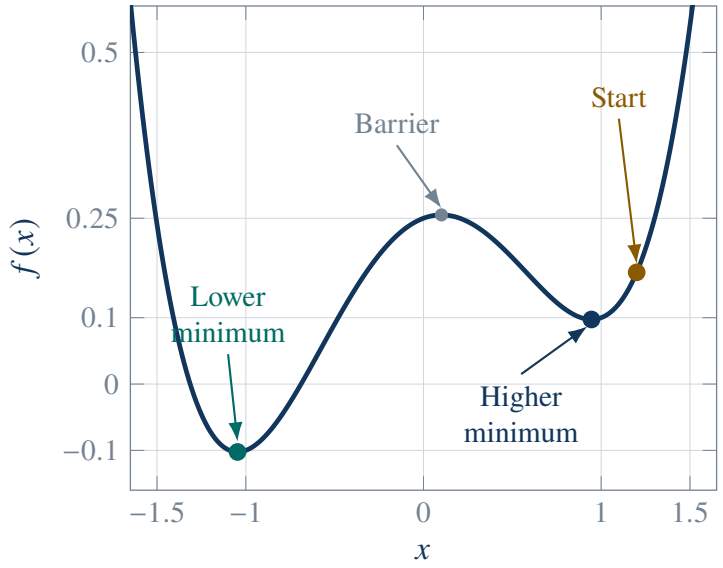
T	10	1	0.2
$\Pr[\text{accept an increase of } 2]$	0.819	0.135	0.0000454

The factor r controls how long uphill exploration remains likely.

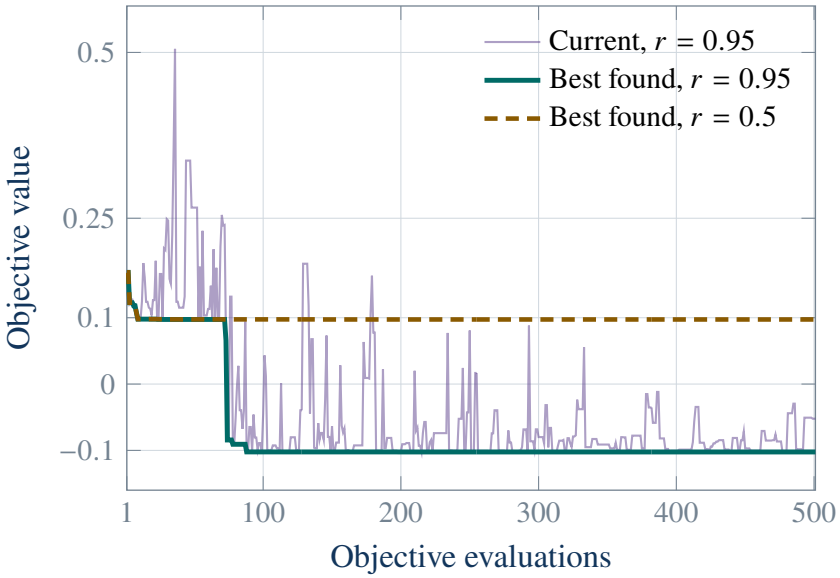
Cooling too quickly can leave the search in its initial basin. A finite run with geometric cooling has no general guarantee of finding a global minimum.

Example

$$f(x) = \tfrac{1}{4}(x^2 - 1)^2 + 0.1x, \qquad x_0 = 1.2$$



The specified gradient-descent, BFGS, and Nelder–Mead runs finish at the higher minimum.



Same random seed, two cooling schedules.
An uphill move changes the current value, not the best found.

$r = 0.95$ *(slower cooling)*, $r = 0.5$ *(rapid cooling)* .