Effective Programming Practices for Economists

Numerical Optimization

Derivative-Based Line Search Algorithms

Janoś Gabler, Hans-Martin von Gaudecker, and Tim Mensinger

Basic Idea (optimagic docs)

1. Evaluate function at initial point
2. Use first derivative to get step direction
3. Guess initial step length based on (approximated) second derivative
4. Pick candidate step based on line search procedure (see next slide)
5. Accept the new parameter and go back to 1.

   (ignore the case where we don't accept)

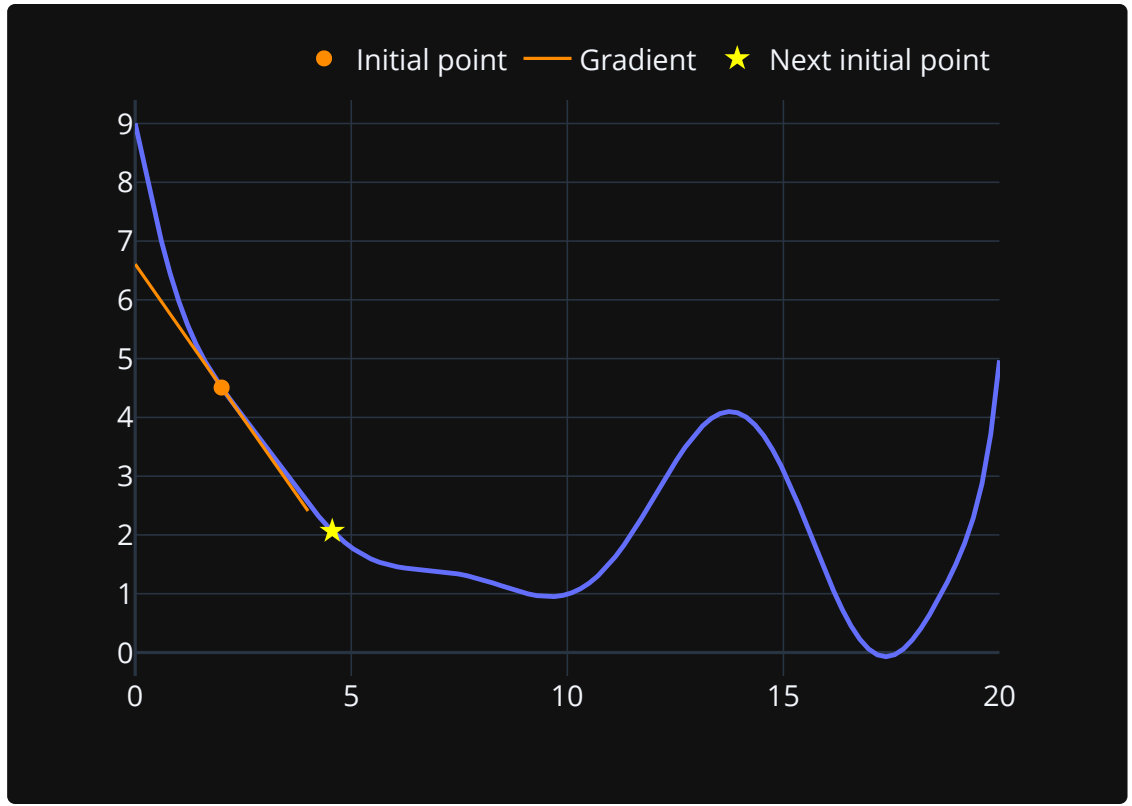
The candidate step

- Compute the search direction p based on some x

  For gradient descent: $p = -f'(x)$

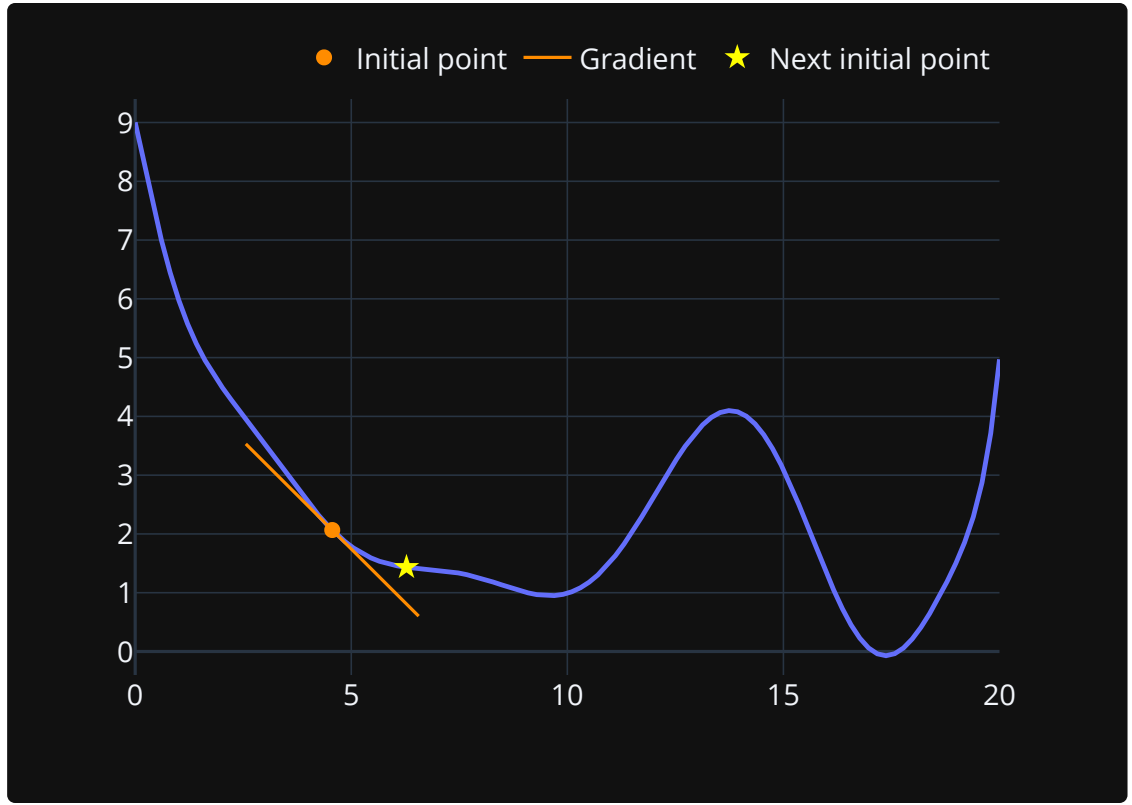
- Line search: Choose step length α to minimize f along the direction p
  - remains a 1d problem even with many parameters
  - only solved approximately
  - quite complicated if you really want to understand it
  - most of the time accepts the first guess
- The candidate step x_c is defined as: $x_c = x + \alpha\, p$

Initial Evaluation



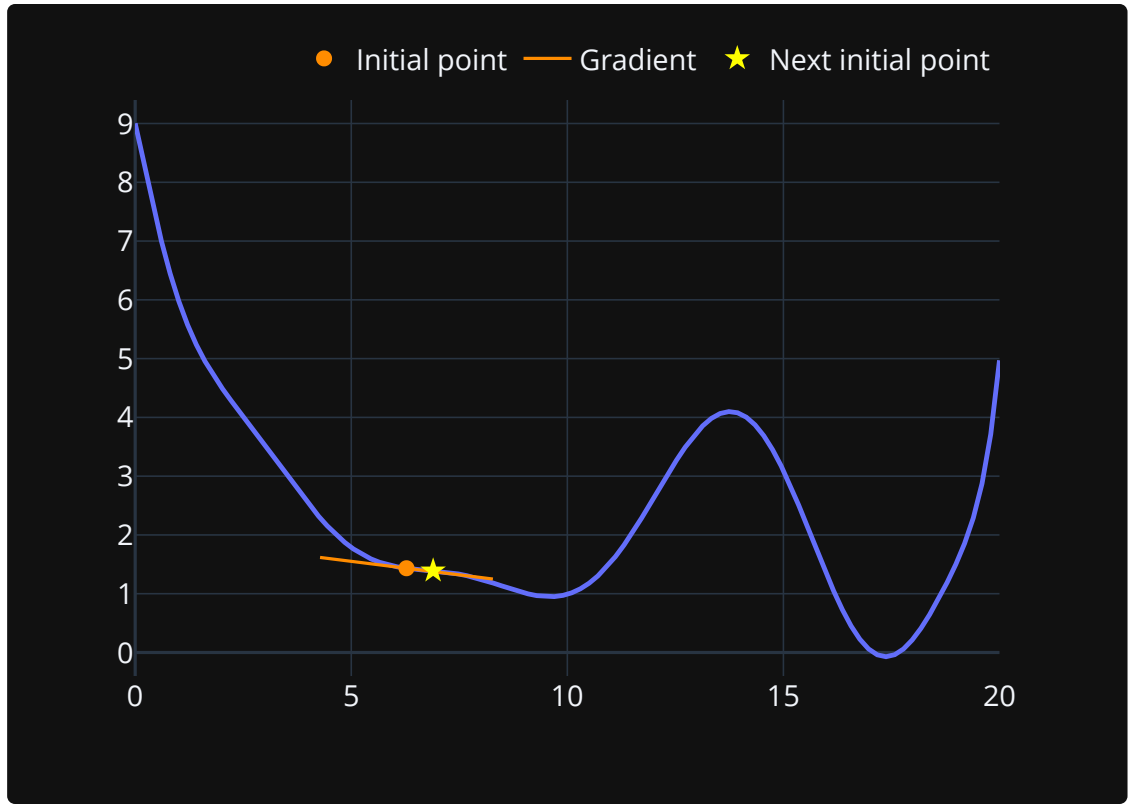
Large gradient, small curvature → Big step.

Iteration 1



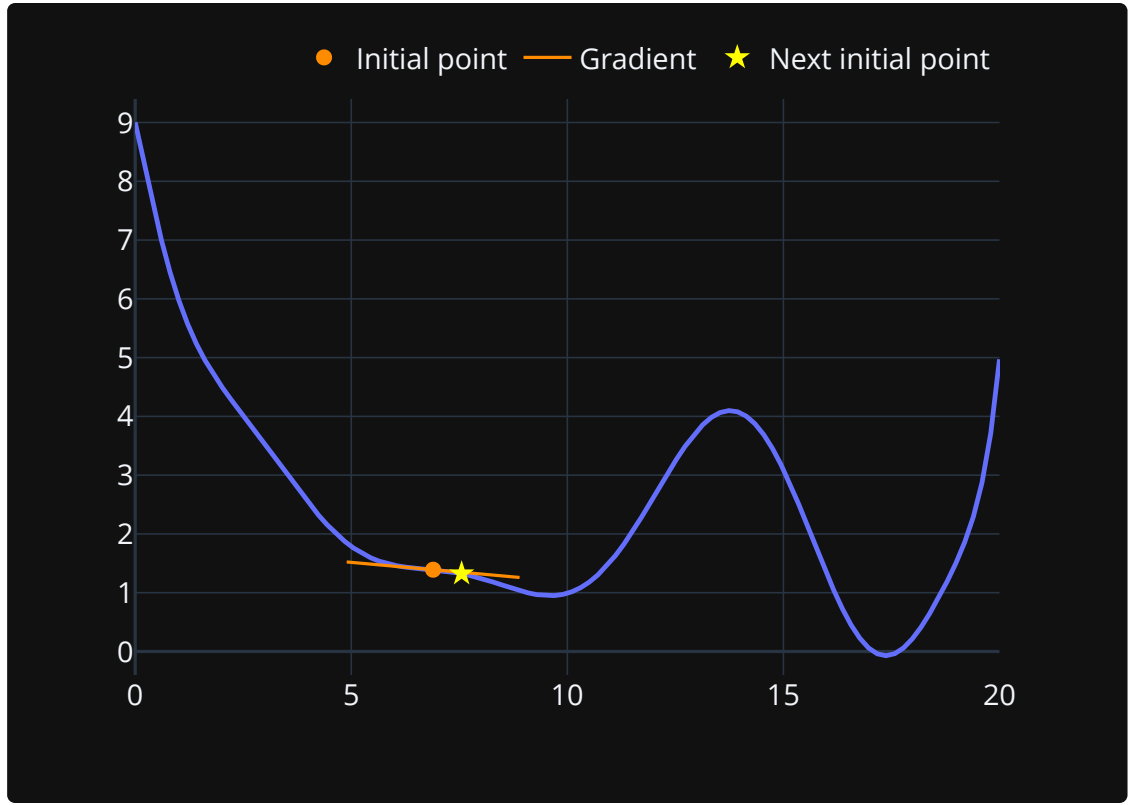
Large gradient, large curvature → Small step.

Iteration 2



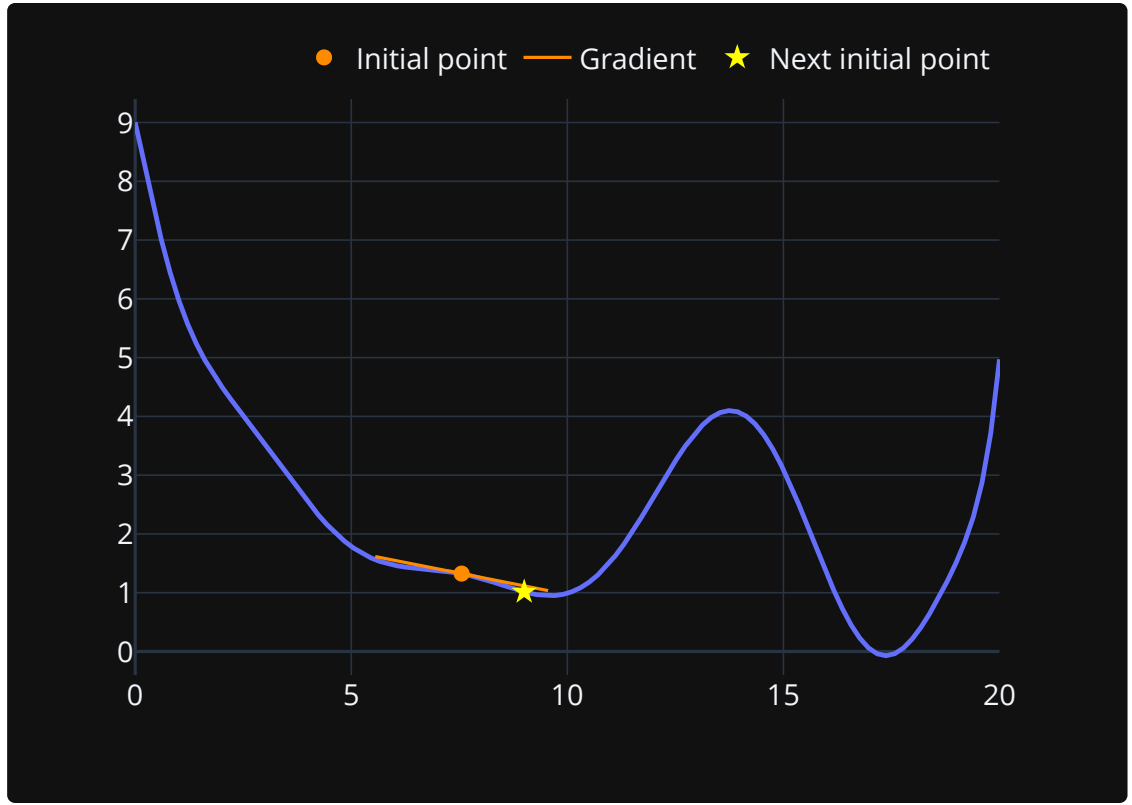
Small gradient, small curvature → Small step.

Iteration 3



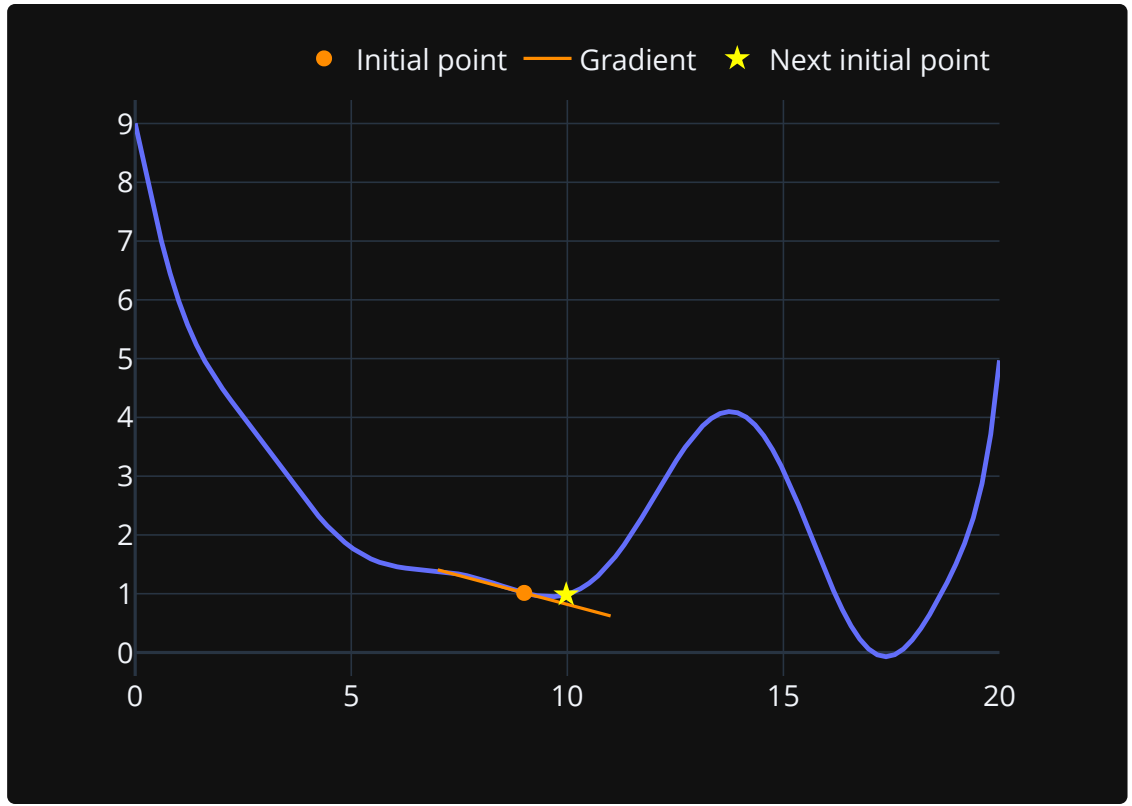
Small gradient, small curvature → Small step.

Iteration 4



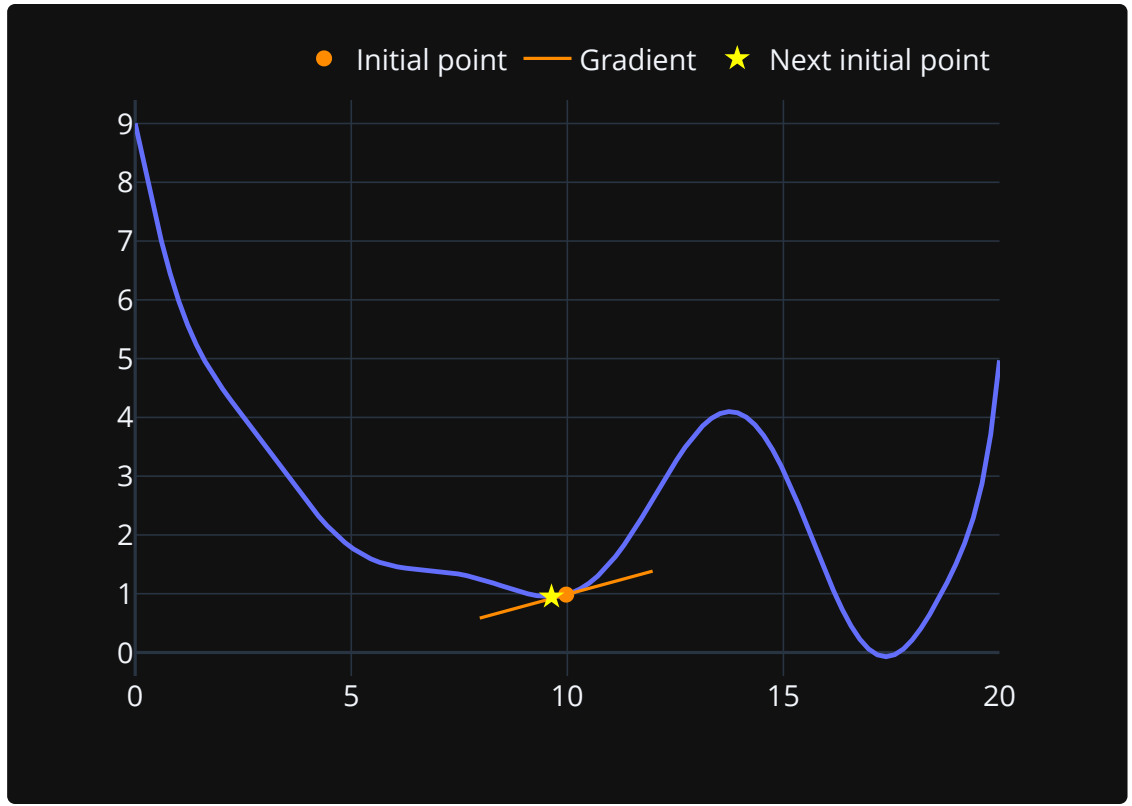
Medium-sized gradient, small curvature → Medium-sized step.

Iteration 5



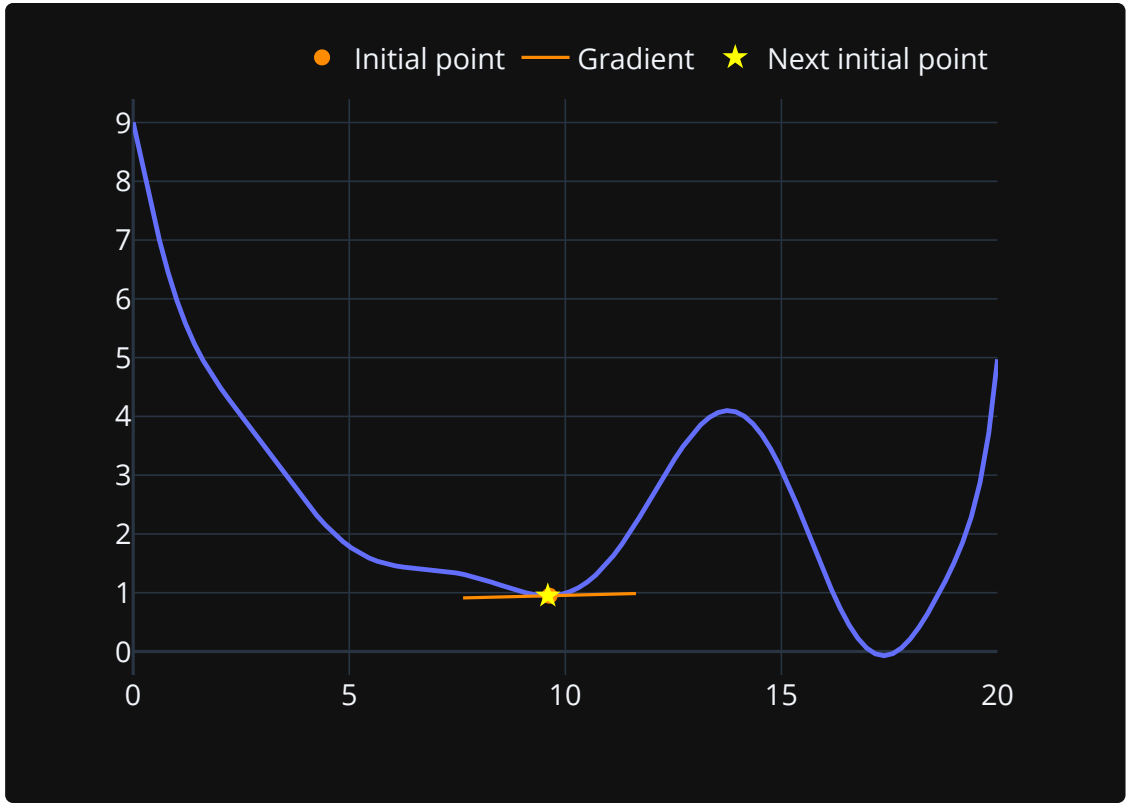
Small gradient, small curvature → Small step.

Iteration 6



Reverse direction because gradient switches sign.

Iteration 7



Converge because gradient is approximately zero.

Some Remarks

- No tuning parameters: Big advantage over other types of algorithms.
- Standard gradient descent would always use the same step length — what we showed converges in fewer steps.
- Nevertheless, standard gradient descent can be computationally better in very high dimensional problems (Hessian becomes too large!).

A Real Algorithm: L-BFGS-B