Chapter 2

Line Search Techniques

2.1 Introduction

Most practical optimization problems involve many variables, so the study of single variable minimization may seem academic. However, the optimization of multivariable functions can be broken into two parts: 1) finding a suitable search direction and 2) minimizing along that direction. The second part of this strategy, the *line search*, is our motivation for studying single variable minimization.

Consider a scalar function $f : \mathbb{R} \to \mathbb{R}$ that depends on a single independent variable, $x \in \mathbb{R}$. Suppose we want to find the value of x where $f(x)$ is a minimum value. Furthermore, we want to do this with

- low computational cost (few iterations and low cost per iteration),
- low memory requirements, and
- low failure rate.

Often the computational effort is dominated by the computation of f and its derivatives so some of the requirements can be translated into: evaluate $f(x)$ and $\mathrm{d}f/\mathrm{d}x$ as few times as possible.

2.2 Optimality Conditions

We can classify a minimum as a:

1. Strong local minimum
2. Weak local minimum
3. Global minimum

As we will see, Taylor's theorem is useful for identifying local minima. Taylor's theorem states that if $f(x)$ is n times differentiable, then there exists $\theta \in (0,1)$ such that

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \cdots + \frac{1}{(n-1)!}h^{n-1}f^{n-1}(x) + \underbrace{\frac{1}{n!}h^n f^n(x+\theta h)}_{\mathcal{O}(h^n)} \tag{2.1}$$

Now we assume that f is twice-continuously differentiable and that a minimum of f exists at x^*. Then, using $n = 2$ and $x = x^*$ in Taylor's theorem we obtain,

$$f(x^* + \varepsilon) = f(x^*) + \varepsilon f'(x^*) + \frac{1}{2}\varepsilon^2 f''(x^* + \theta\varepsilon) \tag{2.2}$$

For x^* to be a local minimizer, we require that $f(x^* + \varepsilon) \geq f(x^*)$ for $\varepsilon \in [-\delta, \delta]$, where δ is a positive number. Given this definition, and the Taylor series expansion (2.2), for $f(x^*)$ to be a local minimum, we require

$$\varepsilon f'(x^*) + \frac{1}{2}\varepsilon^2 f''(x^* + \theta\varepsilon) \geq 0 \tag{2.3}$$

For any finite values of $f'(x^*)$ and $f''(x^*)$, we can always chose a ε small enough such that $\varepsilon f'(x^*) \gg \frac{1}{2}\varepsilon^2 f''(x^*)$. Therefore, we must consider the first derivative term to establish under which conditions we can satisfy the inequality (2.3).

For $\varepsilon f'(x^*)$ to be non-negative, then $f'(x^*) = 0$, because the sign of ε is arbitrary. This is the *first-order optimality condition.* A point that satisfies the first-order optimality condition is called a *stationary point.*

Because the first derivative term is zero, we have to consider the second derivative term. This term must be non-negative for a local minimum at x^*. Since ε^2 is always positive, then we require that $f''(x^*) \geq 0$. This is the *second-order optimality condition.* Terms higher than second order can always be made smaller than the second-order term by choosing a small enough ε.

Thus the *necessary conditions* for a local minimum are:

$$f'(x^*) = 0; \qquad f''(x^*) \geq 0 \tag{2.4}$$

If $f(x^* + \varepsilon) > f(x^*)$, and the "greater than or equal to" in the required inequality (2.3) can be shown to be greater than zero (as opposed to greater or equal), then we have a *strong* local minimum. Thus, *sufficient conditions* for a strong local minimum are:

$$f'(x^*) = 0; \qquad f''(x^*) > 0 \tag{2.5}$$

The optimality conditions can be used to:

1. Verify that a point is a minimum (sufficient conditions).
2. Realize that a point is not a minimum (necessary conditions).
3. Define equations that can be solved to find a minimum.

Gradient-based minimization methods find a local minima by finding points that satisfy the optimality conditions.

2.3 Numerical Precision

Solving the first-order optimality conditions, that is, finding x^* such that $f'(x^*) = 0$, is equivalent to finding the roots of the first derivative of the function to be minimized. Therefore, root finding methods can be used to find stationary points and are useful in function minimization.

Using machine precision, it is not possible find the exact zero, so we will be satisfied with finding an x^* that belongs to an interval $[a, b]$ such that the function g satisfies

$$g(a)g(b) < 0 \quad \text{and} \quad |a - b| < \varepsilon \tag{2.6}$$

where ε is a "small" tolerance. This tolerance might be dictated by the machine representation (using double precision this is usually 1×10^{-16}), the precision of the function evaluation, or a limit on the number of iterations we want to perform with the root finding algorithm.

2.4 Convergence Rate

Optimization algorithms compute a sequence of approximate solutions that we hope converges to the solution. Two questions are important when considering an algorithm:

- Does it converge?
- How fast does it converge?

Suppose you we have a sequence of points x_k $(k = 1, 2, \ldots)$ converging to a solution x^*. For a convergent sequence, we have

$$\lim_{k\to\infty} x_k - x^* = 0 \tag{2.7}$$

The *rate of convergence* is a measure of how fast an iterative method converges to the numerical solution. An iterative method is said to converge with order r when r is the largest positive number such that

$$0 \le \lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^r} < \infty. \tag{2.8}$$

For a sequence with convergence rate r, *asymptotic error constant*, γ is the limit above, i.e.

$$\gamma = \lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^r}. \tag{2.9}$$

To understand this better, lets assume that we have ideal convergence behavior so that condition (2.9) is true for every iteration $k + 1$ and we do not have to take the limit. Then we can write

$$\|x_{k+1} - x^*\| = \gamma\|x_k - x^*\|^r \quad \text{for all } k. \tag{2.10}$$

The larger r is, the faster the convergence. When $r = 1$ we have *linear convergence*, and $\|x_{k+1} - x^*\| = \gamma\|x_k - x^*\|$. If $0 < \gamma < 1$, then the norm of the error decreases by a constant factor for every iteration. If $\gamma > 1$, the sequence diverges. When we have linear convergence, the number of iterations for a given level of convergence varies widely depending on the value of the asymptotic error constant.

If $\gamma = 0$ when $r = 1$, we have a special case: *superlinear convergence*. If $r = 1$ and $\gamma = 1$, we have *sublinear convergence*.

When $r = 2$, we have *quadratic convergence*. This is highly desirable, since the convergence is rapid and independent of the asymptotic error constant. For example, if $\gamma = 1$ and the initial error is $\|x_0 - x^*\| = 10^{-1}$, then the sequence of errors will be $10^{-1}, 10^{-2}, 10^{-4}, 10^{-8}, 10^{-16}$, i.e., the number of correct digits doubles for every iteration, and you can achieve double precision in four iterations!

For n dimensions, x is a vector with n components and we have to rethink the definition of the error. We could use, for example, $||x_k - x^*||$. However, this depends on the scaling of x, so we should normalize with respect to the norm of the current vector, i.e.

$$\frac{||x_k - x^*||}{||x_k||}. \tag{2.11}$$

One potential problem with this definition is that x_k might be zero. To fix this, we write,

$$\frac{||x_k - x^*||}{1 + ||x_k||}. \tag{2.12}$$

Another potential problem arises from situations where the gradients are large, i.e., even if we have a small $||x_k - x^*||$, $|f(x_k) - f(x^*)|$ is large. Thus, we should use a combined quantity, such as,

$$\frac{||x_k - x^*||}{1 + ||x_k||} + \frac{|f(x_k) - f(x^*)|}{1 + |f(x_k)|} \tag{2.13}$$

One final issue is that when we perform numerical optimization, x^* is usually not known! However, you can monitor the progress of your algorithm using the difference between successive steps,

$$\frac{||x_{k+1} - x_k||}{1 + ||x_k||} + \frac{|f(x_{k+1}) - f(x_k)|}{1 + |f(x_k)|}. \tag{2.14}$$

Sometimes, you might just use the second fraction in the above term, or the norm of the gradient. You should plot these quantities on a log-axis versus k.

2.5 Method of Bisection

Bisection is a *bracketing method*. This class of methods generate a set of nested intervals and requires an initial interval that contains the solution.

In this method for finding the zero of a function f, we first establish a bracket $[x_1, x_2]$ for which the function values $f_1 = f(x_1)$ and $f_2 = f(x_2)$ have opposite sign. The function is then evaluated at the midpoint, $x = \frac{1}{2}(x_1 + x_2)$. If $f(x)$ and f_1 have opposite signs, then x and f become x_2 and f_2 respectively. On the other hand, if $f(x)$ and f_1 have the same sign, then x and f become x_1 and f_1. This process is then repeated until the desired accuracy is obtained.

For an initial interval $[x_1, x_2]$, bisection yields the following interval at iteration k,

$$\delta_k = \frac{x_1 - x_2}{2^k} \tag{2.15}$$

Thus, to achieve a specified tolerance δ, we need $\log_2(x_1 - x_2)/\delta$ evaluations.

Bisection yields the smallest interval of uncertainty for a specified number of function evaluations, which is the best you can do for a first order root finding method. From the definition of rate of convergence, for $r = 1$,

$$\lim_{k\to\infty} = \frac{\delta_{k+1}}{\delta_k} = \frac{1}{2} \tag{2.16}$$

Thus this method converges linearly with asymptotic error constant $\gamma = 1/2$. To find the minimum of a function using bisection, we would evaluate the derivative of f at each iteration, instead of the value.

2.6 Newton's Method for Root Finding

Newton's method for finding a zero can be derived from the Taylor's series expansion about the current iteration x_k,

$$f(x_{k+1}) = f(x_k) + (x_{k+1} - x_k)f'(x_k) + \mathcal{O}((x_{k+1} - x_k)^2) \tag{2.17}$$

Ignoring the terms higher than order two and assuming the function's next iteration to be the root (i.e., $f(x_{k+1}) = 0$), we obtain,

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}. \tag{2.18}$$

This iterative procedure converges quadratically, so

$$\lim_{k\to\infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|^2} = \text{const.} \tag{2.19}$$

A pictoral representation of two Newton iterations is shown in Fig. 2.1.

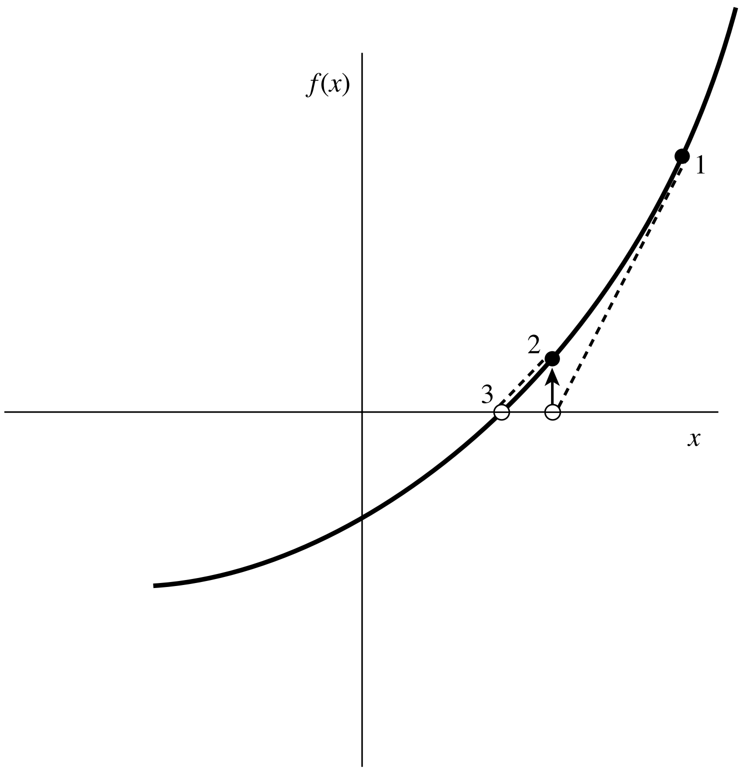

Figure 2.1: Iterations of Newton method for root finding

While having quadratic converge is a desirable property, Newton's method is not guaranteed to converge, and only works under certain conditions. Two examples where this method fails are shown in Fig. 2.2.

To minimize a function using Newton's method, we simply substitute the function for its first derivative and the first derivative by the second derivative,

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}. \tag{2.20}$$

Example 2.1. Function Minimization Using Newton's Method

Consider the following single-variable optimization problem

$$\begin{aligned} &\text{minimize} f(x) = (x-3)x^3(x-6)^4 \\ &\text{w.r.t.} x \end{aligned}$$

We solve this using Newton's method, and the results are shown in Fig. ??

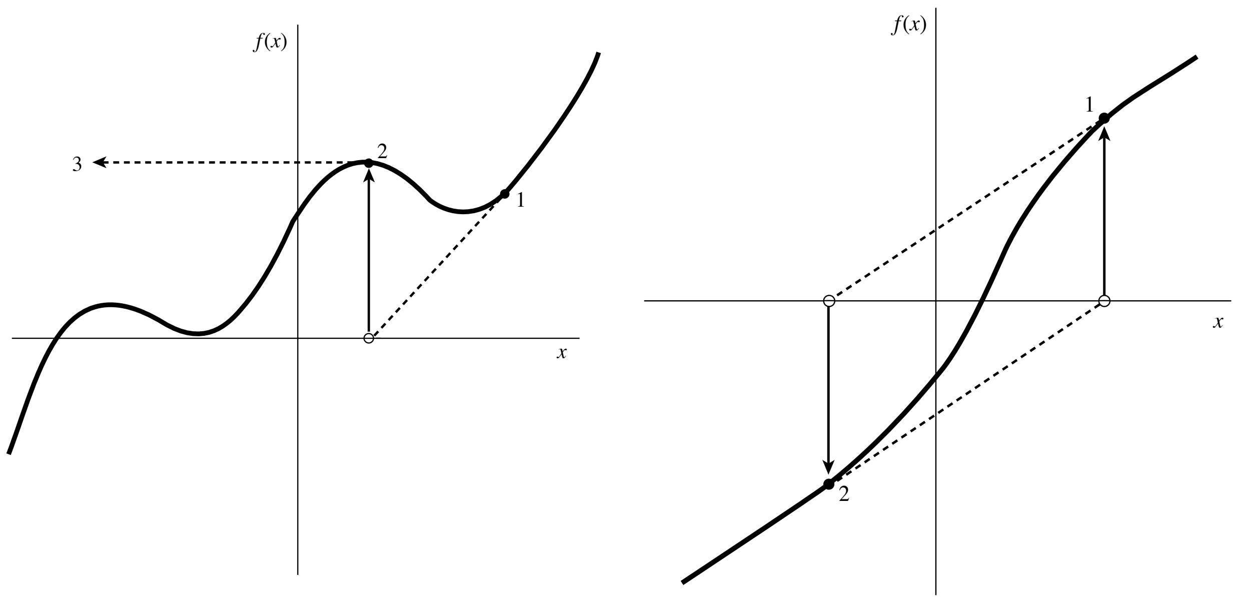

Figure 2.2: Cases where the Newton method fails

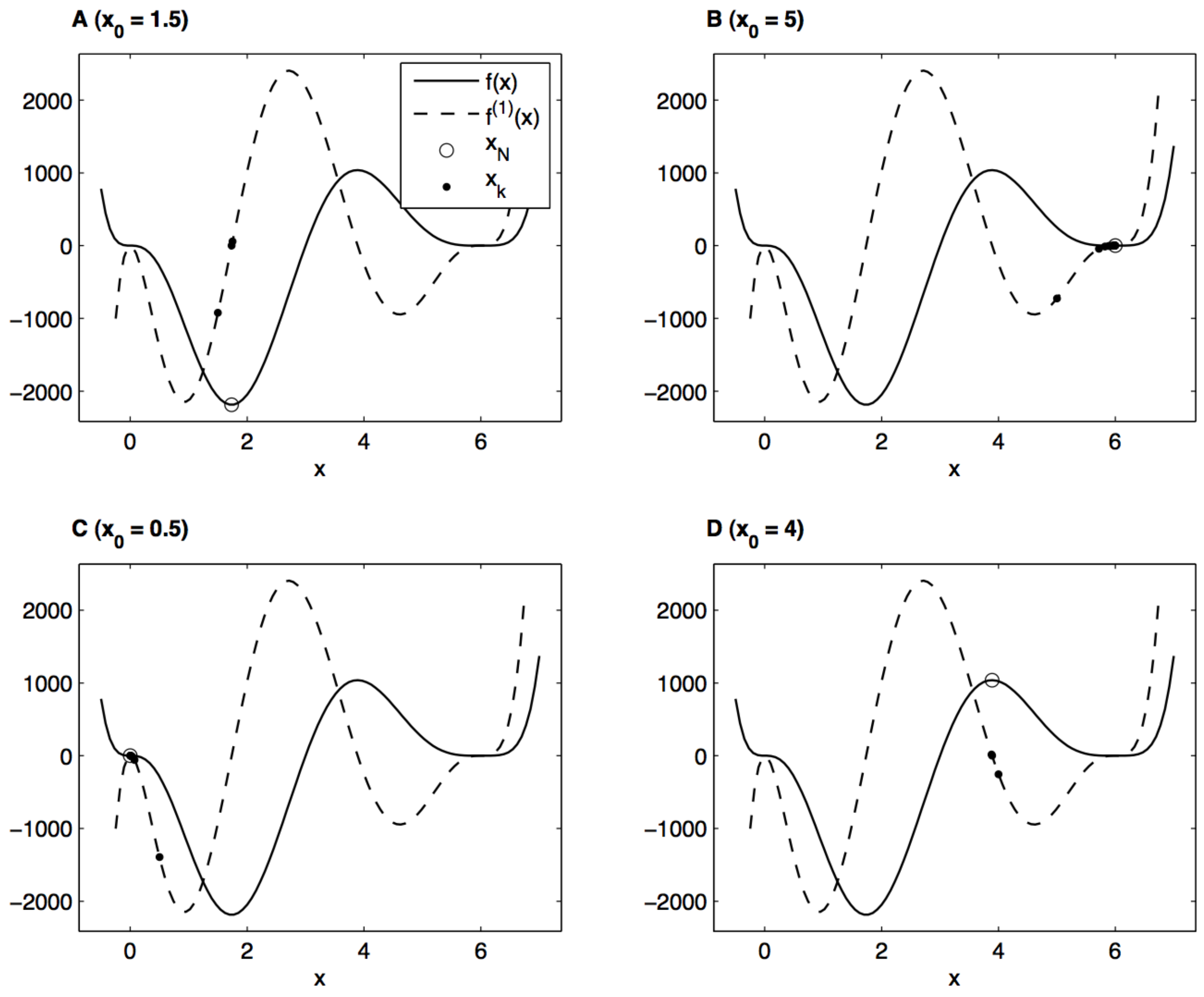

Figure 2.3: Newton's method with several different initial guesses. The x_k are Newton iterates, x_N is the converged solution, and $f^{(1)}(x)$ denotes the first derivative of $f(x)$.

2.7 Secant Method

Newton's method requires the first derivative for each iteration (and the second derivative when applied to minimization). In some practical applications, it might not be possible to obtain this derivative analytically or it might just be troublesome.

If we use a forward-difference approximation for $f'(x_k)$ in Newton's method we obtain

$$x_{k+1} = x_k - f(x_k)\left(\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}\right). \tag{2.21}$$

which is the secant method (also known as "the poor-man's Newton method"). Under favorable conditions, this method has superlinear convergence ($1 < r < 2$), with $r \approx 1.6180$.

2.8 Golden Section Search

The golden section method is a function minimization method, as opposed to a root finding algorithm. It is analogous to the bisection algorithm in that it starts with an interval that is assumed to contain the minimum and then reduces that interval.

Say we start the search with an uncertainty interval $[0, 1]$. Unlike root finding methods, one function evaluation in the interval does not provide sufficient information to subdivide the interval; we need two function evaluations to determine which contains a minimum. We do not want to bias towards one side, so we choose the points symmetrically, say $1-\tau$ and τ, where $0 < \tau < 1$. In other words this produces two intervals both of length τ (Fig. 2.4). Like bisection, it would be inefficient to have one candidate interval smaller than the other. If the original interval is of length I_1 then this requirement produces two intervals $I_2 = \tau I_1$.

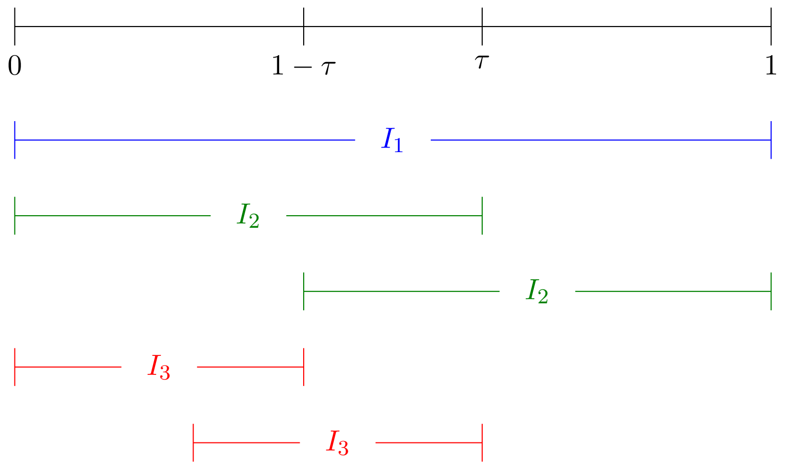

Figure 2.4: Golden Section Search interval division procedure.

Our second objective is to be able to reuse points. If we have to evaluate two new points every iteration then our method will be inefficient. In other words (referencing Fig. 2.4), if the left interval of I_2 contains a minimum and we apply the same division procedure, we would like it to automatically select $1-\tau$ as one of the interval points so that we could reuse its function evaluation. This implies that $I_1 = I_2 + I_3$. Substituting in the expressions $I_2 = \tau I_1$ and $I_3 = \tau I_2 = \tau^2 I_1$ yields

$$\tau^2 + \tau - 1 = 0 \tag{2.22}$$

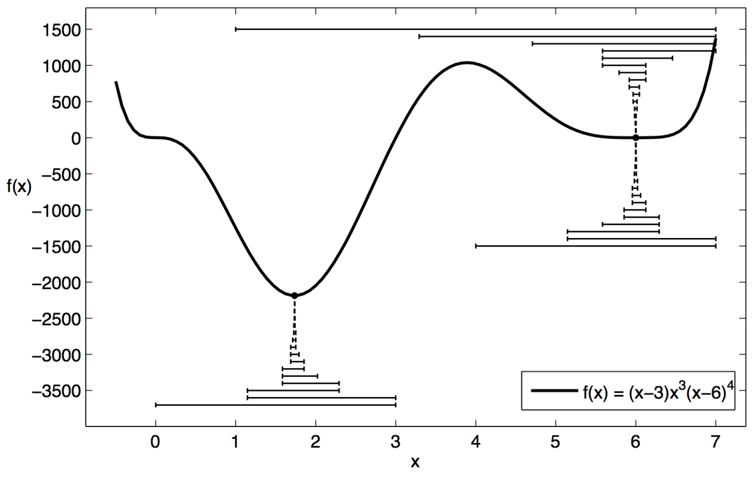

Figure 2.5: The golden section method with initial intervals $[0, 3]$, $[4, 7]$, and $[1, 7]$. The horizontal lines represent the sequences of the intervals of uncertainty.

We could also come to this same conclusion by looking at at the ratio of interval divisions such that they are equal.

$$\frac{\tau}{1} = \frac{1-\tau}{\tau} \Rightarrow \tau^2 + \tau - 1 = 0 \tag{2.23}$$

The positive solution of this equation is the golden ratio,

$$\tau = \frac{\sqrt{5}-1}{2} = 0.618033988749895\ldots \tag{2.24}$$

(actually this is the inverse of the golden ratio as it is typically defined).

So we evaluate the function at $1 - \tau$ and τ, and then the two possible intervals are $[0, \tau]$ and $[1 - \tau, 1]$, which have the same size. If, say $[0, \tau]$ is selected, then the next two interior points would be $\tau(1 - \tau)$ and $\tau\tau$. But $\tau^2 = 1 - \tau$ from the definition (2.23), and we already have this point!

The Golden section method exhibits linear convergence.

Example 2.2. Line Search Using Golden Section

Consider the following single-variable optimization problem

$$\begin{array}{ll} \text{minimize} & f(x) = (x-3)x^3(x-6)^4 \\ \text{w.r.t.} & x \end{array}$$

Solve this using the golden section method. The result are shown in Fig. 2.5. Note the convergence to different optima, depending on the starting interval, and the fact that it might not converge to the best optimum within the starting interval.

2.9 Polynomial Interpolation

More efficient procedures use information about f gathered during iteration. One way of using this information is to produce an estimate of the function which we can easily minimize. The lowest order function that we can use for this purpose is a quadratic, since a linear function does not have a minimum.

Suppose we approximate f by

$$\tilde{f} = \frac{1}{2}ax^2 + bx + c. \tag{2.25}$$

If $a > 0$, the minimum of this function is $x^* = -b/a$.

To generate a quadratic approximation, three independent pieces of information are needed. For example, if we have the value of the function, its first derivative, and its second derivative at point x_k, we can write a quadratic approximation of the function value at x as the first three terms of a Taylor series

$$\tilde{f}(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(x_k)(x - x_k)^2 \tag{2.26}$$

If $f''(x_k)$ is not zero, and setting $x = x_{k+1}$ this yields

$$x^* = -b/a \tag{2.27}$$

$$\Rightarrow x_{k+1} - x_k = -\frac{f'(x_k)}{f''(x_k)} \tag{2.28}$$

$$\Rightarrow x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)} \tag{2.29}$$

which is identical to Newton's method used to find a zero of the first derivative.

2.10 Brent's Method

Bracketing methods are robust but slow (linear convergence), whereas polynomial methods are fast (quadratic convergence) but susceptible to convergence failure. Brent's method combines the two approaches using quadratic interpolation combined with Golden section search. Instead of using a function value and the first and second gradient to create a quadratic fit, Brent's method uses function evaluations at three different points. A new iteration point is estimated based on the polynomial fit. If the point falls outside of the bounding interval or requires a move that is too large, then the polynomial fit is rejected and the algorithms falls back to Golden section search for that iteration. Brent's method demonstrates the same provable convergence behavior as bracketing methods, but converges faster (superlinearly).

The implementation is more complex and will not be enumerated here, but details can be found in separate references [2] and various available implementations. The analogue for root finding combines bisection, a secant method, and quadratic interpolation and is widely used for 1-dimensional root finding.

2.11 Line Search Techniques

Line search methods are related to single-variable optimization methods, as they address the problem of minimizing a multivariable function along a line, which is a subproblem in many gradient-based optimization method.

After a gradient-based optimizer has computed a search direction p_k, it must decide how far to move along that direction. The step can be written as

$$x_{k+1} = x_k + \alpha_k p_k \tag{2.30}$$

where the positive scalar α_k is the *step length.*

Most algorithms require that p_k be a *descent direction*, i.e., that $p_k^T g_k < 0$, since this guarantees that f can be reduced by stepping some distance along this direction. The question then becomes, how far in that direction we should move.

We want to compute a step length α_k that yields a substantial reduction in f, but we do not want to spend too much computational effort in making the choice. Ideally, we would find the global minimum of $f(x_k + \alpha_k p_k)$ with respect to α_k but in general, it is to expensive to compute this value. Even to find a local minimizer usually requires too many evaluations of the objective function f and possibly its gradient g. More practical methods perform an *inexact* line search that achieves adequate reductions of f at reasonable cost.

2.11.1 Wolfe Conditions

A typical line search involves trying a sequence of step lengths, accepting the first that satisfies certain conditions. A common condition requires that α_k should yield a *sufficient decrease* of f, as given by the inequality

$$f(x_k + \alpha p_k) \leq f(x_k) + \mu_1 \alpha g_k^T p_k \tag{2.31}$$

for a constant $0 \leq \mu_1 \leq 1$. In practice, this constant is small, say $\mu_1 = 10^{-4}$. This sufficient decrease condition is also known as Armijo's rule.

Any sufficiently small step can satisfy the sufficient decrease condition, so in order to prevent steps that are too small we need a second requirement called the *curvature condition*, which can be stated as

$$g(x_k + \alpha p_k)^T p_k \geq \mu_2 g_k^T p_k \tag{2.32}$$

where $\mu_1 \leq \mu_2 \leq 1$, and $g(x_k + \alpha p_k)^T p_k$ is the derivative of $f(x_k + \alpha p_k)$ with respect to α_k. This condition requires that the slope of the univariate function at the new point be greater. Since we start with a negative slope, the gradient at the new point must be either less negative or positive. The intuition is that if the gradient is even more negative then we should be taking a larger step size to take advantage of the expected decrease (in other words the step size is too small). Typical values of μ_2 are 0.9 when using a Newton type method and 0.1 when a conjugate gradient methods is used. The sufficient decrease and curvature conditions are known collectively as the *Wolfe conditions.*

We can also modify the curvature condition to force α_k to lie in a broad neighborhood of a local minimizer or stationary point and obtain the *strong Wolfe conditions*

$$\begin{aligned} f(x_k + \alpha p_k) &\leq f(x_k) + \mu_1 \alpha g_k^T p_k. && (2.33) \\ \left|g(x_k + \alpha p_k)^T p_k\right| &\leq \mu_2 \left|g_k^T p_k\right|, && (2.34) \end{aligned}$$

where $0 < \mu_1 < \mu_2 < 1$. The only difference when comparing with the Wolfe conditions is that we do not allow points where the derivative has a positive value that is too large and therefore exclude points that are far from the stationary points. Figure 2.6 graphically demonstrates acceptable intervals that satisfy the Wolfe conditions.

2.11.2 Sufficient Decrease and Backtracking

One of the simplest line search algorithms is *backtracking line search*, listed in Algorithm 1. This algorithm consists of guessing a maximum step, then successively decreasing that step until the new point satisfies the sufficient decrease condition.

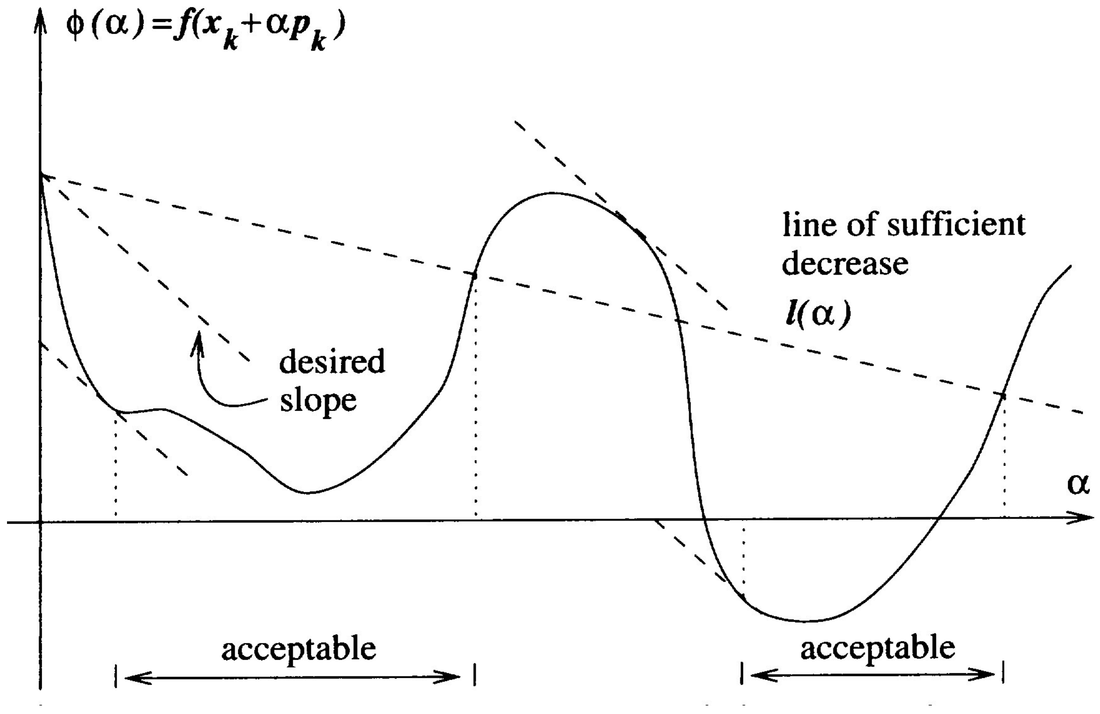

Figure 2.6: Acceptable steps for the strong Wolfe conditions.

Algorithm 1 Backtracking line search algorithm

```
Input: $\alpha > 0$ and $0 < \rho < 1$
Output: $\alpha_k$
1: repeat
2:     $\alpha \leftarrow \rho\alpha$
3: until $f(x_k + \alpha p_k) \leq f(x_k) + \mu_1 \alpha g_k^T p_k$
4: $\alpha_k \leftarrow \alpha$
```

2.11.3 Line Search Algorithm Satisfying the Strong Wolfe Conditions

To simplify the notation, we define the univariate function

$$\phi(\alpha) = f(x_k + \alpha p_k) \tag{2.35}$$

Then at the starting point of the line search, $\phi(0) = f(x_k)$. The slope of the univariate function, is the projection of the n-dimensional gradient onto the search direction p_k, i.e.,

$$\phi'(\alpha) = g(x_k + \alpha p_k)^T p_k \tag{2.36}$$

This procedure has two stages:

1. Begins with trial α_1, and keeps increasing it until it finds either an acceptable step length or an interval that brackets the desired step lengths.
2. In the latter case, a second stage (the `zoom` algorithm below) is performed that decreases the size of the interval until an acceptable step length is found.

The first stage of this line search is detailed in Algorithm 2.

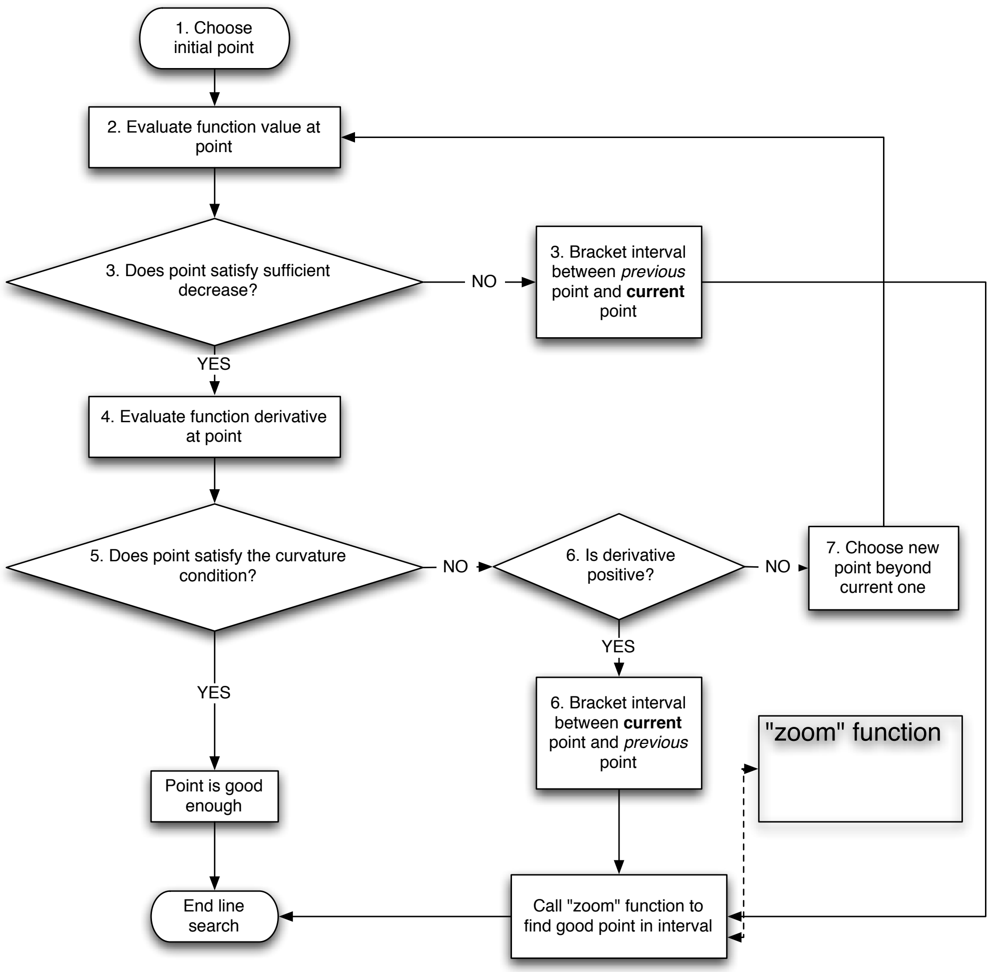
1. Choose initial point
2. Evaluate function value at point
3. Does point satisfy sufficient decrease?
NO
3. Bracket interval between *previous* point and **current** point
YES
4. Evaluate function derivative at point
5. Does point satisfy the curvature condition?
NO
6. Is derivative positive?
NO
7. Choose new point beyond current one
YES
6. Bracket interval between **current** point and *previous* point
YES
"zoom" function
Point is good enough
End line search
Call "zoom" function to find good point in interval

Figure 2.7: Line search algorithm satisfying the strong Wolfe conditions

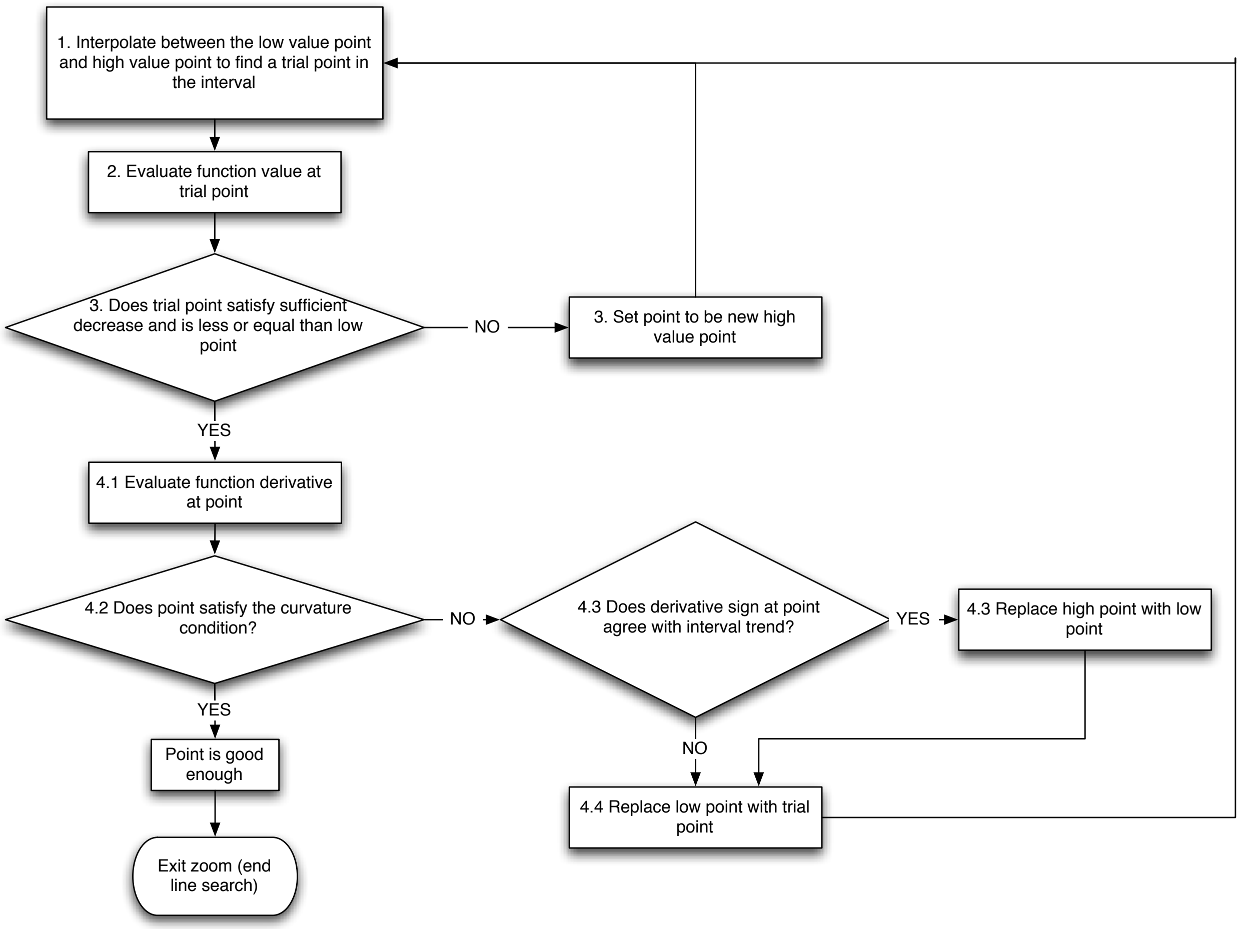
1. Interpolate between the low value point and high value point to find a trial point in the interval
2. Evaluate function value at trial point
3. Does trial point satisfy sufficient decrease and is less or equal than low point
NO
3. Set point to be new high value point
YES
4.1 Evaluate function derivative at point
4.2 Does point satisfy the curvature condition?
NO
4.3 Does derivative sign at point agree with interval trend?
YES
4.3 Replace high point with low point
YES
Point is good enough
NO
4.4 Replace low point with trial point
Exit zoom (end line search)

Figure 2.8: “Zoom” function in the line search algorithm satisfying the strong Wolfe conditions

Algorithm 2 Line search algorithm

```
Input: $\alpha_1 > 0$ and $\alpha_{\max}$
Output: $\alpha_*$
 1: $\alpha_0 = 0$
 2: $i \leftarrow 1$
 3: repeat
 4:   Evaluate $\phi(\alpha_i)$
 5:   if $[\phi(\alpha_i) > \phi(0) + \mu_1 \alpha_i \phi'(0)]$ or $[\phi(\alpha_i) > \phi(\alpha_{i-1})$ and $i > 1]$ then
 6:     $\alpha_* \leftarrow$ zoom$(\alpha_{i-1}, \alpha_i)$
 7:     return $\alpha_*$
 8:   end if
 9:   Evaluate $\phi'(\alpha_i)$
10:   if $|\phi'(\alpha_i)| \leq -\mu_2 \phi'(0)$ then
11:     return $\alpha_* \leftarrow \alpha_i$
12:   else if $\phi'(\alpha_i) \geq 0$ then
13:     $\alpha_* \leftarrow$ zoom$(\alpha_i, \alpha_{i-1})$
14:     return $\alpha_*$
15:   else
16:     Choose $\alpha_{i+1}$ such that $\alpha_i < \alpha_{i+1} < \alpha_{\max}$
17:   end if
18:   $i \leftarrow i + 1$
19: until
```

The algorithm for the second stage, the `zoom`$(\alpha_{\text{low}}, \alpha_{\text{high}})$ function is given in Algorithm 3. To find the trial point, we can use cubic or quadratic interpolation to find an estimate of the minimum within the interval. This results in much better convergence that if we use the golden search, for example.

Note that sequence $\{\alpha_i\}$ is monotonically increasing, but that the order of the arguments supplied to `zoom` may vary. The order of inputs for the call is `zoom`$(\alpha_{\text{low}}, \alpha_{\text{high}})$, where:

1. the interval between α_{low}, and α_{high} contains step lengths that satisfy the strong Wolfe conditions

2. α_{low} is the one giving the lowest function value

3. α_{high} is chosen such that $\phi'(\alpha_{\text{low}})(\alpha_{\text{high}} - \alpha_{\text{low}}) < 0$

The interpolation that determines α_j should be safeguarded, to ensure that it is not too close to the endpoints of the interval.

Using an algorithm based on the strong Wolfe conditions (as opposed to the plain Wolfe conditions) has the advantage that by decreasing μ_2, we can force α to be arbitrarily close to the local minimum.

Algorithm 3 Zoom function for the line search algorithm

```
Input: $\alpha_{\text{low}}, \alpha_{\text{high}}$
Output: $\alpha_*$
1: $j \leftarrow 0$
2: repeat
3:    Find a trial point $\alpha_j$ between $\alpha_{\text{low}}$ and $\alpha_{\text{high}}$
4:    Evaluate $\phi(\alpha_j)$
5:    if $\phi(\alpha_j) > \phi(0) + \mu_1 \alpha_j \phi'(0)$ or $\phi(\alpha_j) > \phi(\alpha_{\text{low}})$ then
6:       $\alpha_{\text{high}} \leftarrow \alpha_j$
7:    else
8:       Evaluate $\phi'(\alpha_j)$
9:       if $|\phi'(\alpha_j)| \leq -\mu_2 \phi'(0)$ then
10:         $\alpha_* = \alpha_j$
11:         return $\alpha_*$
12:      else if $\phi'(\alpha_j)(\alpha_{\text{high}} - \alpha_{\text{low}}) \geq 0$ then
13:         $\alpha_{\text{high}} \leftarrow \alpha_{\text{low}}$
14:      end if
15:      $\alpha_{\text{low}} \leftarrow \alpha_j$
16:   end if
17:   $j \leftarrow j + 1$
18: until
```

Example 2.3. Line Search Algorithm Using Strong Wolfe Conditions

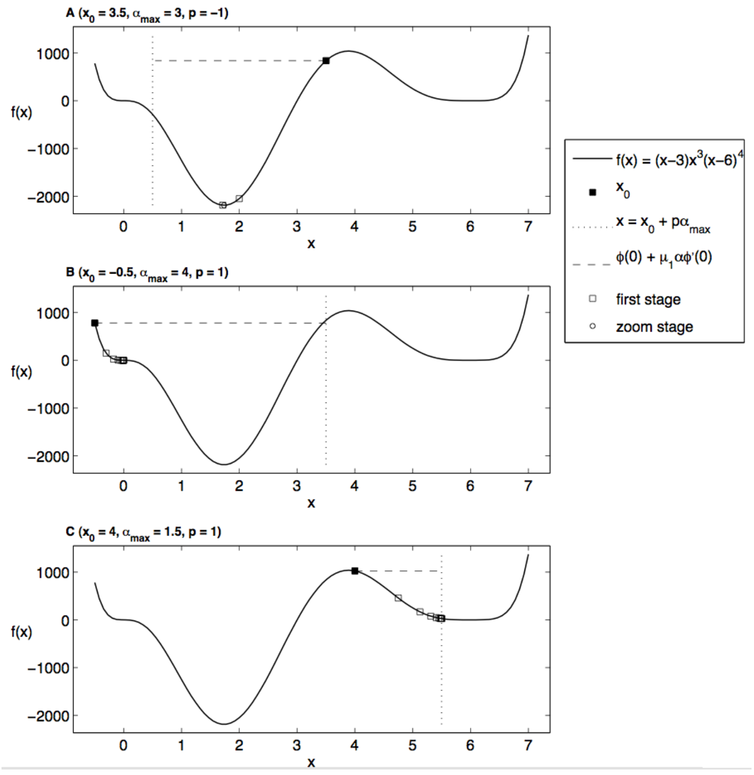

Figure 2.9: The line search algorithm iterations. The first stage is marked with square labels and the zoom stage is marked with circles.

Bibliography

[1] Ashok D. Belegundu and Tirupathi R. Chandrupatla. *Optimization Concepts and Applications in Engineering*, chapter 2. Prentice Hall, 1999.

[2] Richard P. Brent. *Algorithms for Minimization Without Derivatives.* Courier Corporation, Jun 2013. ISBN 0486143686.

[3] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*, chapter 4. Academic Press, 1981.

[4] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*, chapter 3. Springer-Verlag, 1999.