Chapter 7

Univariate Minimization

A major problem in scientific computing is optimization. In many real life problems we need to maximize or minimize a quantity over choices of a number of variables, possibly with some additional constraints on the variables. For example, we may want to maximize a profit function that depends on cost and production variables, with the constraint that cost must be positive. Here, we consider the simplest such problem, the minimization of a function of a single variable without side constraints. The methods we describe can simply be adapted for maximization problems by changing the sign of the function.

7.1 Introduction

Recall from Calculus that when you are set the problem of determining a minimum of a continuously differentiable function f on a closed interval $[a, b]$, first you determine the derivative function f'. Then you find all the critical values x in $[a, b]$ where $f'(x) = 0$, say $x_1, x_2, \ldots, x_p$. Finally, you determine the global minimum $\min_{x\in[a,b]} f(x) = \min\{f(a), f(x_1), f(x_2), \ldots, f(x_p), f(b)\}$. There is no need to check whether the critical values $x_1, x_2, \ldots, x_p$ correspond to local minima as we determine the global minimum by enumeration.

Example 7.1.1. To find the minimum of the continuously differentiable function $f(x) = -e^{-x}\sin(x)$ on the closed interval $[0, 1.5]$, first we compute the derivative $f'(x) = -e^{-x}(-\sin(x) + \cos(x))$. Next, we find the critical values x where $f'(x) = 0$. Since e^{-x} is never zero, we must have $-\sin(x)+\cos(x) = 0$; that is, we need the values x such that $\tan(x) = 1$ since $\cos(x) \neq 0$ on $[0, 1.5]$. On $[0, 1.5]$ this equation is satisfied at just one point $x = \frac{\pi}{4}$. Now,

$$\min\left\{f(0), f\left(\frac{\pi}{4}\right), f(1.5)\right\} = \min\{0, -0.322396941945, -0.222571216108\} = -0.322396941945$$

is the global minimum located at $x = \frac{\pi}{4}$.

This approach suggests that we may find the extrema of $f(x)$ by computing the zeros of $f'(x)$. We could find individual extrema by solving the equation $f'(x) = 0$ by using, say Newton's method which would also involve computing $f''(x)$ for use in the iteration

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}$$

or if the second derivative is difficult to obtain by using the secant method for solving $f'(x) = 0$.

In computational practice it is usually infeasible to find a global minimum, unless we know in advance how many local extrema $f(x)$ has. So, here we concentrate on finding just one local

extremum, a local minimum. Also, we begin with methods that use only values of $f(x)$ in finding the local minimum then move to methods which combine using values of $f'(x)$ with values of $f(x)$.

We consider how to determine the location x_m of a minimum of the function f on the interval $[a, b]$. We say that $[a, b]$ is a **bracket for the minimum** of f if x_m is in $[a, b]$. Throughout, we assume that the function f is **U–shaped** on the interval $[a, b]$; i.e., it is continuous, *strictly decreasing* on $[a, x_m)$, and *strictly increasing* on $(x_m, b]$. When the function f has a continuous derivative, then for f to be U–shaped it is sufficient that the derivative f' be negative on $[a, x_m)$ and positive on $(x_m, b]$. When the function f has a continuous second derivative, then for f to be U–shaped it is sufficient that f be concave up on the interval $[a, b]$ and $f'(x_m) = 0$. Of course, we may assume that a function is U–shaped when in reality it is not – usually we have no way to check. So, occasionally we will discuss what happens to our algorithms if the underlying assumption of U–shapedness turns out not to be correct.

Let the interval $[a, b]$ be a bracket for the minimum of a U–shaped function f. The basic process used to refine such a bracket, i.e., to make it shorter, uses **search points** c and d where $a < c < d < b$. Given such search points, it follows that (see Figure 7.1)

1. if $f(c) \leq f(d)$, then $[a, d]$ is a bracket for the minimum, and
2. if $f(c) \geq f(d)$, then $[c, b]$ is a bracket for the minimum.

To prove that 1 is true, suppose that both $f(c) \leq f(d)$ and $x_m > d$. Now $x_m > d$ implies that both of the points c and d lie to the left of the point x_m where the function f is strictly decreasing, so $f(c) > f(d)$. But this contradicts the supposition that $f(c) \leq f(d)$, so both of $f(c) \leq f(d)$ and $x_m > d$ cannot be true. Therefore, if $f(c) \leq f(d)$ is true, then $x_m > d$ must be false, i.e., the point x_m lies in the interval $[a, d]$. So, the statement in Part 1 is true. The proof of 2 is similar; see Problem 7.1.2.

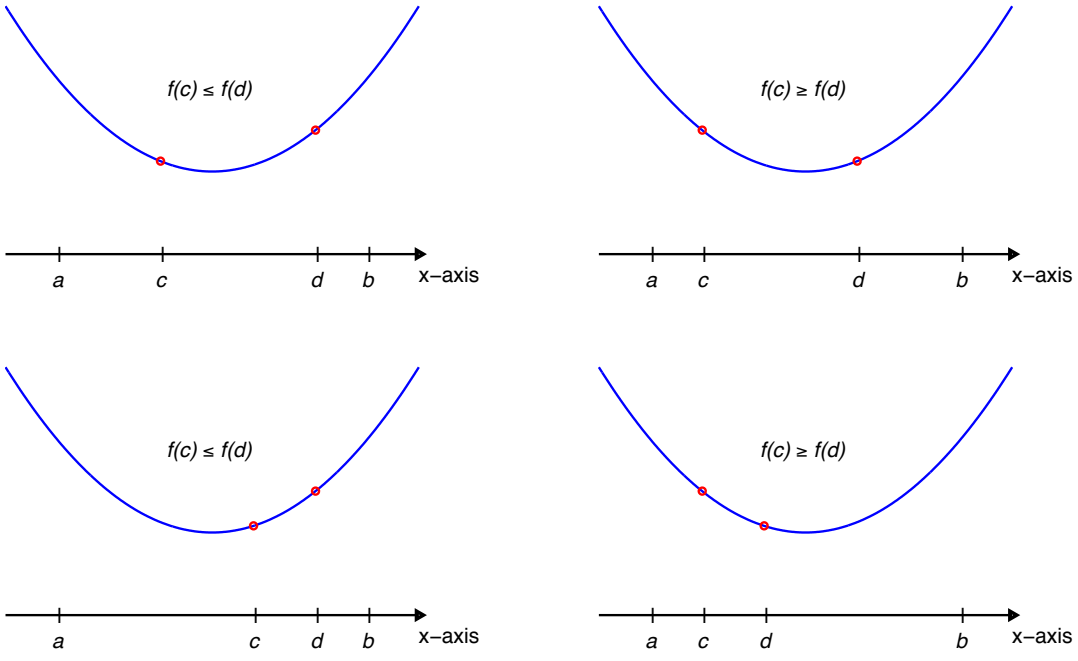

Figure 7.1: Illustration of bracket refining to find a minimum of a function. The plots on the left illustrate that if $f(c) \leq f(d)$, then $[a, d]$ is a bracket for the minimum. The plots on the right illustrate that if $f(c) \geq f(d)$, then $[c, b]$ is a bracket for the minimum.

Unlike when refining a bracket for a root, refining a bracket for the minimum generally requires the function to be evaluated at two distinct points inside the bracket. To understand why, let

$[a, b]$ be a bracket for the minimum of the U–shaped function f, and let c be some point in the open interval (a, b). There is no decision rule that compares the values $f(a)$, $f(b)$ and $f(c)$ and determines correctly which of the intervals, either $[a, c]$ or $[c, b]$, is a bracket for the minimum. To see why, pick three values v_a, v_b and v_c for which $v_c < \min\{v_a, v_b\}$ and plot the points (a, v_a), (b, v_b) and (c, v_c). Consider Figure 7.2, which shows the graphs of the values of two U–shaped functions $f_1(x)$ and $f_2(x)$ that pass through these three points but the minimum of $f_1(x)$ is located to the right of the point c and the minimum of $f_2(x)$ is located to the left of c. Observe that v_a, v_b and v_c are the common values assumed by these two functions at the points a, b and c, respectively. If the decision rule compares the values v_a, v_b and v_c and declares the interval $[c, b]$ to be the refined bracket, then this would be correct for the U–shaped function f_1 but incorrect for the U–shaped function f_2. On the other hand, if the decision rule compares the values v_a, v_b and v_c and declares the interval $[a, c]$ to be the refined bracket, then it would be correct for the U–shaped function f_2 but incorrect for the U–shaped function f_1. The conclusion is that, generally, a decision rule that refines a bracket for the minimum of a U–shaped function f must evaluate the function at more than one distinct point inside the open interval (a, b).

Say we evaluate f at an interior point c and we find that $f(a) < f(c) < f(b)$. This does not violate the assumption that f is U–shaped; it simply tells us that if there is a unique local minimum it must lie in the interval $[a, c]$. However, if $c \in (a, b)$ and $f(c) \geq \max\{f(a), f(b)\}$ then the assumption that f is U–shaped is violated. Our methods below will involve using two interior points at each iteration. In this case there are other ways that U-shapedness can be violated (and which should be checked for in the algorithm). For example, let $a < c < d < b$ and $f(a) < f(c) < f(d) < f(b)$. This is possible when f is U–shaped but if $f(a) < f(c) > f(d) < f(b)$ then f cannot be U–shaped.

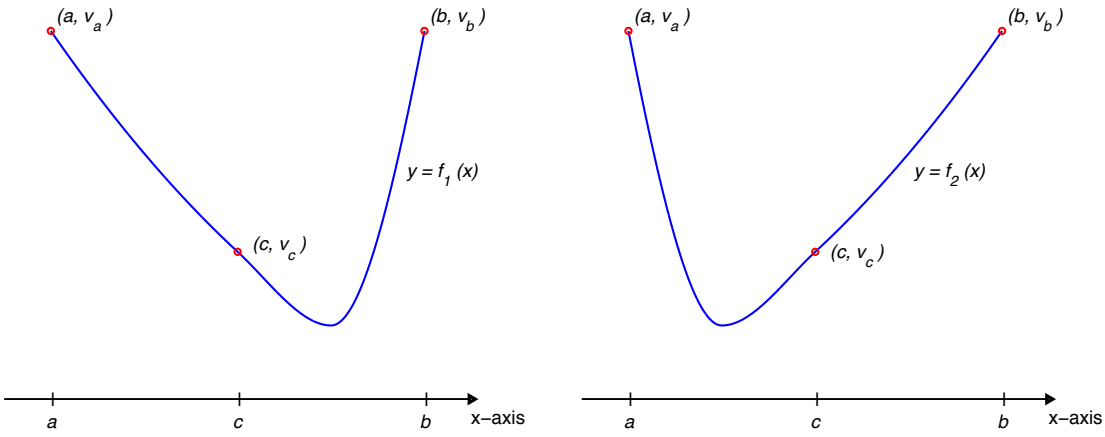

Figure 7.2: The plots in this figure illustrate that if the function is evaluated at only one point, c, in the interval $[a, b]$, then it may not be possible to determine whether the minimum of the function is to the left or to the right of c.

Problem 7.1.1. *Show that $x = \frac{\pi}{4}$ is a local minimum of $f(x) = -e^{-x}\sin(x)$. Find a local maximum of $f(x)$.*

Problem 7.1.2. *Prove the statement presented in Part 2. Hint: Suppose that both $f(c) \geq f(d)$ and $x_m < c$ are true. Describe the contradiction obtained from this supposition. Conclude that if $f(c) \geq f(d)$ is true, then $x_m < c$ must be false; i.e., the point x_m lies in the interval $[c, b]$.*

Problem 7.1.3. *Let $f(x)$ be a U–shaped function defined on the interval $[a, b]$. For every choice of the point $c \in (a, b)$, show that $f(c) < \max\{f(a), f(b)\}$. Hint: Show that the function f cannot be U–shaped on the interval $[a, b]$ if there is a value of $c \in (a, b)$ for which $f(c) \geq \max\{f(a), f(b)\}$.*

Problem 7.1.4. *Let the function f be U–shaped on the interval $[a, b]$, and let c be some point in the open interval (a, b) for which $f(c) \geq \min\{f(a), f(b)\}$. If $f(c) \geq f(a)$, then show that the interval $[a, c]$ is a bracket for the minimum. Similarly, if $f(c) \geq f(b)$, then show that the interval $[c, b]$ is a bracket for the minimum.*

Problem 7.1.5. *Pick three values v_a, v_b and v_c such that $v_c < \min\{v_a, v_b\}$ and plot the points (a, v_a), (c, v_c) and (b, v_b). Sketch the graph of two piecewise linear (V-shaped) functions that pass through these three points and yet assume their minimum value at points on opposite sides of the point c. Argue why these piecewise linear functions are U–shaped functions.*

7.2 Search Methods not Involving Derivatives

Here, we describe methods which only use values of $f(x)$ in the search for the location of its local minimum. In each case the methods proceed by reducing the length of a bracket on the location. The first method makes no use of the absolute size of the function in determining the iterates. The second uses interpolation formulas and hence implicitly incorporates the function's size.

7.2.1 Golden Section Search

The golden section search for the minimum of a U–shaped function f is analogous to the bisection method for finding a root of $f(x)$. Golden section is characterized by how the search points c and d are chosen. If $[a, b]$ is a bracket for the minimum of the function f, then its associated **golden section search points** c and d are

$$\begin{aligned} c &\equiv a + r(b - a) \\ d &\equiv b - r(b - a) \end{aligned}$$

where

$$r \equiv \frac{3 - \sqrt{5}}{2} \approx 0.382\ldots$$

Because $r < 1$, c is the golden section search point nearest the endpoint a and we say that a and c are *associated*. Similarly, we say that the points d and b are associated. Observe that the endpoints of each of the possible refined brackets, that is $[a, d]$ or $[c, b]$, consist of *one* golden section search point and the endpoint associated with the *other* golden section search point.

The name golden section search is derived from the fact that the length of the original bracket divided by the length of either possible refined bracket is

$$\frac{1}{1 - r} = \frac{1 + \sqrt{5}}{2} \approx 1.618,$$

a value known in antiquity as the **golden section** constant.

Choosing c and d to be the golden section search points has two advantages. First, each refinement step of golden section search reduces the length of the bracket by a factor of $1 - r \approx 0.618$. Second, the two golden section search points for the original bracket occur in the refined bracket one as an endpoint and the other as an associated golden section search point. This is illustrated in Table 7.1 and Figure 7.3 for a bracket and its two possible refinements. Note how the golden section search points of the original bracket appear in the refined bracket. One is an endpoint and the other is that endpoint's associated golden section search point. As a consequence, the value $f(x)$ at the golden section search point common to both the original and refined brackets can and should be reused in the next refinement step!

Bracket, $[a_i, b_i]$		Left Endpoint a_i	Left GSSP c_i	Right GSSP d_i	Right Endpoint b_i
Original Bracket	$[0, 1]$	0	0.382	0.618	1
Left Refinement	$[0, 0.618]$	0	0.236	0.382	0.618
Right Refinement	$[0.382, 1]$	0.382	0.618	0.764	1

Table 7.1: Endpoints and golden section search points (GSSP). The left refinement case is illustrated in the left part of Figure 7.3, and the right refinement is illustrated in the right part of Figure 7.3

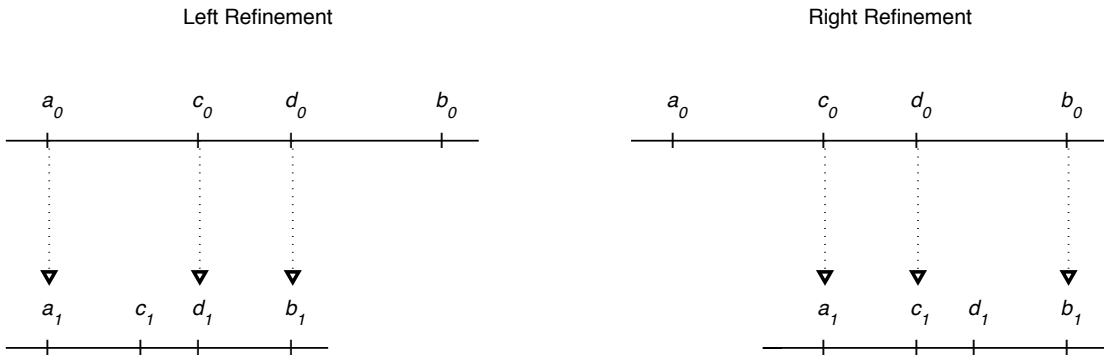

Figure 7.3: The figure illustrates the two possibilities for refining a bracket in the golden section search method. Notice that only one new point needs to be computed for each refinement.

Figure 7.4 displays a pseudocode for golden section search; it assumes that the initial points a and b as well the function $f(x)$ are provided. At the top of the while-loop the only points known are the endpoints a, b and the golden section search point c associated with a. For refinement, the code computes the golden section search point d associated with b, via the technique in Problem 7.2.3, then the if-statement identifies the refined bracket. This refined bracket consists of the interval between the golden section search point with the largest value of $f(x)$ and the endpoint associated with the other golden section search point. The body of the if-statement applies the label c to the golden section search point that has the largest value $f(x)$, fc to its associated function value, a to the endpoint associated with c, and b to the other endpoint. Note that the computation of the point d and the refinement of the bracket work equally well whether $a < b$ or $a > b$. This pseudocode for golden section search incurs a startup cost of one function evaluation, and then a cost of one function evaluation for each refinement step. Generally, in costing the algorithm the startup cost is either ignored or is considered to be amortized over the cost of all the function evaluations used in the later refinement steps. So, each step of golden section search for the minimum reduces the length of the bracket for the minimum by a factor of about 0.618 at a cost of about one function evaluation. Error detection should be included in an algorithm for golden section search. For example, if it turns out that $f(d) \geq \max\{f(a), f(b)\}$, then it should be reported that the function f is not U–shaped so there may be more than one local extremum in the interval $[a, b]$. However, the algorithm could be permitted to continue searching for a local minimum with the starting bracket $[d, b]$ and interior point c; note, the pseudocode terminates in these circumstances.

What termination criteria should we use? Suppose the point x lies in a bracket $[a, b]$ for the minimum of $f(x)$. This implies that $|x - x_m| \leq |b - a|$. So, a common practice is to stop the search when the bracket is sufficiently short, that is, when

$$|b - a| \leq \sqrt{\frac{\epsilon}{L}}$$

Golden Section Search

Input: function to minimize, $f(x)$
initial bracket, $[a, b]$

Output: scalars c, fc, where $fc = f(c) \approx \min_{a \le x \le b} f(x)$

$r = (3 - \sqrt{5})/2$
$c = a + r(b - a)$; $fc = f(c)$
if $fc > fa$ or $fc > fb$ then Stop
while termination criteria not satisfied do
  $d = c + r(b - c)$; $fd = f(d)$
  if $fc \le fd$ then
    $b = a$; $fb = fa$; $a = d$; $fa = fd$
  else
    $a = c$; $fc = fa$; $c = d$; $fc = fd$
  endif
  if $fc > fa$ or $fc > fb$ then Stop
end while

Figure 7.4: Pseudocode *Golden Section Search*

where ϵ is the working precision machine epsilon and L is a positive constant which we determine below. This termination criterion can be justified as follows. Rearrange this inequality so that it reads

$$L(b - a)^2 \le \epsilon$$

The left hand side of this inequality provides an upper bound on the absolute relative error in accepting $f(x)$ as the value of $f(x_m)$ while the right hand side provides an upper bound on the absolute relative error in the computed value $F(x)$ of $f(x)$. So, this termination criterion states that the search be stopped when the difference between the values $f(x)$ and $f(x_m)$ could be attributed to rounding error in the computed value of $f(x_m)$. There are two reasons why the value $F(x)$ differs from the value $f(x_m)$. The first reason is that $f(x)$ is computed using floating point arithmetic. So, the best we can expect is that $F(x)$ is the rounded value of $f(x)$ from which it follows that

$$\left| \frac{F(x) - f(x)}{f(x)} \right| \le \epsilon$$

The second reason is that the point x differs from x_m. If f has two continuous derivatives, then since $f'(x_m) = 0$, Taylor's series states that as $x \to x_m$,

$$f(x) \approx f(x_m) + \frac{f''(x_m)}{2}(x - x_m)^2 = f(x_m)\,(1 + \psi)$$

where

$$\psi = \frac{f''(x_m)}{2f(x_m)}(x - x_m)^2$$

So, the relative error in accepting the value $f(x)$ as an approximation of $f(x_m)$ satisfies the bound

$$\left| \frac{f(x) - f(x_m)}{f(x_m)} \right| \lessapprox |\psi|$$

Recall that $|x - x_m| \le |b - a|$, so

$$|\psi| \le \left| \frac{f''(x_m)}{2f(x_m)} \right| (b - a)^2 = L(b - a)^2$$

where we have identified

$$L = \left| \frac{f''(x_m)}{2f(x_m)} \right|$$

In terms of the values ψ and ϵ, the proposed termination criterion becomes $|\psi| \leq \epsilon$. In words, the error in accepting the value $f(x)$ as the value of $f(x_m)$ is no larger than an amount that could be attributed to the rounding error in the computed value of $f(x_m)$. Of course, in practice we don't know the value of L, so we choose a value such as $L = 2$, hoping it won't turn out to be so much too small that it prevents termination of the algorithm. It is possible to estimate L using finite differences or by differentiating polynomials interpolating to the most recently calculated values of $f(x)$ to approximate $f''(x_m)$ but the necessary information is not readily available until we have a good approximation to x_m.

This termination criterion also has an important consequence. If we switch from single to double precision arithmetic, we should expect to determine x_m only to just over 4 additional decimal digits of accuracy. This follows because $\sqrt{\epsilon_{\text{SP}}} \approx 10^{-3.5}$ and $\sqrt{\epsilon_{\text{DP}}} \approx 10^{-8}$, so the switch in precision causes the square root of the working precision machine epsilon to be reduced by only a factor of $10^{-4.5}$.

Example 7.2.1. We find a local minimum of $f(x) = -e^{-x}\sin(x)$ on the interval $[0, 1.5]$ using golden section search. We terminate when the bracket is smaller than 10^{-5}. We implemented the pseudocode in Figure 7.4 in DP and terminated when the bracket was smaller than 10^{-5}; results are in Table 7.2. Observe the irregular behavior of the error, somewhat reminiscent of the behavior of the error for the bisection method in root finding.

iterate	$f(iterate)$	*error*
0.000000	0.000000	-0.785398
1.500000	-0.222571	0.714602
0.572949	-0.305676	-0.212449
0.927051	-0.316517	0.141653
1.145898	-0.289667	0.360500
0.791796	-0.322384	0.006398
0.708204	-0.320375	-0.077194
0.843459	-0.321352	0.058061
0.759867	-0.322183	-0.025531
0.811529	-0.322181	0.026131
0.779600	-0.322386	-0.005798
0.772063	-0.322339	-0.013336
0.784259	-0.322397	-0.001140
0.787138	-0.322396	0.001739
0.782479	-0.322394	-0.002919
0.785358	-0.322397	-0.000040
0.786038	-0.322397	0.000640
0.784938	-0.322397	-0.000460
0.785618	-0.322397	0.000220
0.785198	-0.322397	-0.000200
0.785457	-0.322397	0.000059
0.785297	-0.322397	-0.000101
0.785396	-0.322397	-0.000002

Table 7.2: Golden section – iterates, function values and errors in iterates

Problem 7.2.1. *Show that the length of each of the two possible refined brackets, $[a, d]$ and $[c, b]$, is $(1 - r)(b - a)$. What is the numerical value of $1 - r$?*

Problem 7.2.2. *Consider the formulas for the golden section search points c and d. Show that if we swap the values a and b, then the values c and d are swapped too, i.e., the formula for c after the swap is the same as the formula for d before the swap, and vice versa. Remark: The conclusion is that, regardless of whether $a < b$ or $b < a$, the formula for c determines the golden section search point associated with a, and the formula for d determines the golden section search point associated with b.*

Problem 7.2.3. *Let the points a and b be given, and let c and d be their associated golden section search points. Show that*

$$d = c + r(b - c)$$

Computing the point d this way has the advantage that d surely lies between c and b.

Problem 7.2.4. *Show that r is a root of the quadratic equation $r^2 - 3r + 1 = 0$. Algebraically rearrange this equation to show that (a) $(1 - r)^2 = r$, (b) $r(2 - r) = 1 - r$, and (c) $1 - 2r = r(1 - r)$.*

Problem 7.2.5. *Given the initial bracket for the minimum is $[0, 1]$, show explicitly the computation of the remaining numbers in Table 7.1.*

Problem 7.2.6. *Construct a table similar to Table 7.1 for a general initial bracket $[a, b]$.*

7.2.2 Quadratic Interpolation Search

For brevity, we use the phrase **quadratic search** to describe the search for the minimum of a U–shaped function f using quadratic interpolation. What distinguishes quadratic search for the minimum from golden section search is how the search points c and d are chosen. We assume that an initial bracket $[a, b]$ for the minimum of the function $f(x)$ is given. We start by choosing a point c in the open interval (a, b). To start the quadratic search iteration we need $f(c) < \min\{f(a), f(b)\}$. Next, d is determined as the point in the open interval (a, b) where the quadratic polynomial that interpolates the data $(c, f(c))$, $(a, f(a))$ and $(b, f(b))$ assumes its unique minimum value; if $d = c$ an error is reported. In fact, it is quite possible that if $d \approx c$ then the location of the minimum is in their vicinity but if $d = c$ we have no simple way of shortening the bracket of the location. Finally, the bracket is refined as usual; of the search points c and d, the one corresponding to the larger value of f becomes an endpoint of the refined bracket and the other is (re-)labeled c. (In the pseudocode in Figure 7.5 we order the points c and d such that $c < d$ to simplify the coding.)

Let us demonstrate that the point d lies between the midpoints of the intervals $[a, c]$ and $[c, b]$, and so lies in the open interval (a, b). (Of course, that d lies in (a, b) is intuitively obvious but the result we prove tells us a little more, that d cannot be closer than half the distance to a given endpoint than c is from the same endpoint.) To start, let $P_2(x)$ be the quadratic polynomial that interpolates the data $(c, f(c))$, $(a, f(a))$ and $(b, f(b))$. With the data so ordered, the Newton form is

$$P_2(x) = g_0 + (x - c)g_1 + (x - c)(x - a)g_2$$

where the coefficients g_0, g_1 and g_2 are chosen so that $P_2(c) = f(c)$, $P_2(a) = f(a)$ and $P_2(b) = f(b)$. The solution of these equations is

$$\begin{aligned} g_0 &= f(c) \\ g_1 &= \frac{f(c) - f(a)}{c - a} \\ g_2 &= \frac{\left(\dfrac{f(b) - f(c)}{b - c}\right) - g_1}{b - a} \end{aligned}$$

Because $f(c) < \min\{f(a), f(b)\}$, it is easy to see that $g_1 < 0$ and $g_2 > 0$. Consequently, the quadratic polynomial P_2 is concave up and achieves its least value at the point

$$d \equiv \frac{a+c}{2} - \frac{g_1}{2g_2}$$

where $P'(d) = 0$. Because $g_1 < 0$ and $g_2 > 0$, it follows that $d > \frac{a+c}{2}$. Similarly, if the Newton form of the same interpolating polynomial $P_2(x)$ is constructed but with the data ordered as $(c, f(c))$, $(b, f(b))$ and $(a, f(a))$, then it is easy to show that $d < \frac{c+b}{2}$. Therefore, the point d lies between the midpoints of the intervals $[a, c]$ and $[c, b]$.

Generally, quadratic search for the minimum works well. However, for some functions it converges very slowly. In these cases, a pattern develops where successive iterations reset the same endpoint, either a or b, to a nearby point thereby causing the length of the bracket to be only minimally refined. This pattern may be broken by keeping track of the assignments and, when necessary, inserting a special step. For example, a special step might replace the point d with the midpoint of one of the two intervals $[a, c]$ and $[c, b]$, whichever is the longer interval.

In the pseudocode in Figure 7.5, we could use any representation of the quadratic polynomial $P_2(x)$, including one of those used in the analysis above. However, in the interests of transparency and because it should lead to a slightly more accurate computation when rounding error and cancelation impact the calculation, we choose a form centered on the current internal point, c :

$$P_2(x) = A(x-c)^2 + B(x-c) + C$$

The minimum of this quadratic is taken at the location

$$d = c - \frac{B}{2A}$$

so at each iteration we calculate the new point, d, as a correction to c. The point c will usually be a good approximation to the location of the minimum as the iteration proceeds so the correction should be small.

Example 7.2.2. We find a local minimum of $f(x) = -e^{-x}\sin(x)$ on the interval $[a, b] = [0, 1.5]$ using quadratic interpolation search. We terminate the iteration when the bracket is smaller than 10^{-5}. We implemented the pseudocode given in Figure 7.5; results are given in Table 7.3. Observe that the error, that is the distance between the point c and the answer, becomes small quickly but that the bracket takes longer to reduce in length to the required size, hence the need for the last few iterates in the table. Indeed, the left hand end of the bracket remains at $a = 0.75$ from the second iteration until the penultimate iteration.

iterate	$f(iterate)$	*error*
0.000000	0.000000	-0.785398
1.500000	-0.222571	0.714602
0.750000	-0.321983	-0.035398
0.948066	-0.314754	0.162668
0.811887	-0.322175	0.026489
0.786261	-0.322397	0.000862
0.785712	-0.322397	0.000314
0.785412	-0.322397	0.000014
0.785402	-0.322397	0.000004
0.785398	-0.322397	0.000000

Table 7.3: Quadratic interpolation search – iterates, function values and errors in iterates

Quadratic Search

Input: function to minimize, $f(x)$
initial bracket, $[a, b]$

Output: scalars c, fc, where $fc = f(c) \approx \min_{a \le x \le b} f(x)$

$fa = f(a)$; $fb = f(b)$
$c = 0.5(a + b)$; $fc = f(c)$
if $fc \ge fa$ or $fc \ge fb$ then Stop
while termination criteria not satisfied do
  $A = [(fa - fc) * (b - c) - (fb - fc) * (a - c)]/[(a - c) * (b - c) * (a - b)]$
  $B = [(fa - fc) * (b - c)^2 - (fb - fc) * (a - c)^2]/[(a - c) * (b - c) * (b - a)]$
  $d = c - B/(2 * A)$; $fd = f(d)$
  if $c > d$ then, swap c and d, and fc and fd
    $temp = c$; $c = d$; $d = temp$
    $temp = fc$; $fc = fd$; $fd = temp$
  endif
  if $fc \le fd$ then
    $b = d$; $fb = fd$
  else
    $a = c$; $fa = fc$
    $c = d$; $fc = fd$
  endif
  if $fc \ge fa$ or $fc \ge fb$ then Stop
end while

Figure 7.5: Pseudocode *Quadratic Search*. The initial choice of $c = 0.5 * (a + b)$ is arbitrary, but sensible.

Problem 7.2.7. *Let f be a U–shaped function on the interval $[a, b]$ and let c be a point in (a, b) for which $f(c) < \min\{f(a), f(b)\}$. Let the quadratic polynomial $P_2(x)$ interpolate $f(x)$ at the points $x = a$, $x = c$ and $x = b$, so that $P_2(c) < \min\{P_2(a), P_2(b)\}$. Without calculating the quadratic $P_2(x)$, argue why $P_2(x)$ is concave up and why the location d of its minimum lies in (a, b).*

Problem 7.2.8. *Suppose that f is a U–shaped function on the interval $[a, b]$ and c is a point in (a, b) for which $f(c) < \min\{f(a), f(b)\}$. Let P_2 be the quadratic polynomial that interpolates to the data $(a, f(a))$, $(c, f(c))$ and $(b, f(b))$. Use the Mean Value Theorem (see Problem 6.1.3) to demonstrate that there is a point $u \in (a, c)$ for which $P_2'(u) < 0$, and a point $v \in (c, b)$ for which $P_2'(v) > 0$. Use the Intermediate Value Theorem (see Problem 6.1.1) to conclude there is a point d between the points u and v for which $P_2'(d) = 0$.*

Problem 7.2.9. *Show that the interpolation conditions in Problem 7.2.8 lead to the solution listed for the coefficients g_0, g_1 and g_2. Argue why the condition $f(c) < \min\{f(a), f(b)\}$ implies that both $g_1 < 0$ and $g_2 > 0$.*

Problem 7.2.10. *Compute the derivative P_2' and determine the value d for which $P_2'(d) = 0$. Hint: The derivative $P_2'(x)$ is a linear function of x.*

Problem 7.2.11. *Compute the interpolating quadratic polynomial P_2, but this time with the data ordered $(c, f(c))$, $(b, f(b))$ and $(a, f(a))$. For this data, the Newton form of this polynomial is*

$$P_2(x) = h_0 + (x - c)h_1 + (x - c)(x - b)h_2$$

From the interpolating conditions, derive formulas for the coefficients h_1, h_2 and h_3. Argue why $f(c) < \min\{f(a), f(b)\}$ implies that both $h_1 > 0$ and $h_2 > 0$. Argue why $h_2 = g_2$. Determine the value d for which $P_2'(d) = 0$, and argue why $d < \dfrac{c + b}{2}$. Hint: The quadratic interpolating polynomial for this data is unique, regardless of how the data is ordered. So, the power series coefficients of the quadratic $P_2(x)$ are unique. How are the coefficients g_2 and h_2 in this form related to the coefficient of x^2 of the Newton form of $P_2(x)$?

Problem 7.2.12. *Consider the U–shaped function*

$$f(x) = \frac{1}{x(1 - x)^2}$$

on $[a, b] = [0.01, 0.99]$. Using $c = 0.02$ as the initial value, carry out three iterations of quadratic search for the minimum. What is the value of x_m for this function, and how is it related to the values c and d computed by each iteration? Remark: The bracket endpoint b remains frozen in this example for many iterations.

7.3 Using the Derivative

Suppose that we can compute values of both the function f and its derivative f' at any point x. There follows a description of an algorithm that incorporates information from both f and f' in the search for location of the local minimum of f. This method extends that of the previous section.

7.3.1 Cubic Interpolation Search

For brevity, we use the phrase **cubic search** to describe the search for the minimum of a U–shaped function f using cubic interpolation. Let f be U–shaped on $[a, b]$. We suppose that $f'(a)f'(b) < 0$, for otherwise f has an endpoint extremum. Because f is U–shaped, it follows that $f'(a) < 0$ and $f'(b) > 0$.

The key computation performed by cubic search is the construction of the cubic Hermite polynomial P_3 that interpolates values $f(x)$ and $f'(x)$ at the endpoints a and b; that is,

$$P_3(a) = f(a), \quad P_3'(a) = f'(a), \quad P_3(b) = f(b), \quad P_3'(b) = f'(b)$$

This interpolating polynomial was introduced in Problem 4.1.21. If $t \equiv (x-a)/h$ with $h \equiv b-a$, then

$$P_3(x) = f(a)\phi(t) + f(b)\phi(1-t) + hf'(a)\psi(t) - hf'(b)\psi(1-t)$$

where the cubic polynomials $\phi(s) = (1+2s)(1-s)^2$ and $\psi(s) = s(1-s)^2$. However, here we choose a more convenient form

$$P_3(x) = A(x-c)^3 + B(x-c)^2 + C(x-c) + D$$

where $c = \dfrac{a+b}{2}$. This choice gives a symmetry to the interpolating conditions and they are solved simply to give

$$\begin{aligned} A &= [d(f'(b)+f'(a)) - (f(b)-f(a))]/(4d^3) \\ B &= [f'(b)-f'(a)]/(4d) \\ C &= [3(f(b)-f(a)) - d(f'(b)+f'(a))]/(4d) \\ D &= [f(a)+f(b)]/2 - d[f'(b)-f'(a)]/4 \end{aligned}$$

where $d = \dfrac{b-a}{2}$.

Note that $P_3'(x) = 0$ is a quadratic equation that has at most two roots. Since $P_3'(a) = f'(a) < 0$ and $P_3'(b) = f'(b) > 0$, by the Intermediate Value Theorem one root c lies in the open interval (a,b) and the other root, if it exists, must lie outside $[a,b]$. (Remember that the coefficient of x^2 *may be zero* so there may only be one root of $P_3'(x) = 0$.) The root $c \in (a,b)$ corresponds to a local minimum value of the function $P_3(x)$. The roots of $P_3'(x) = 0$ are $\dfrac{a+b}{2} + \dfrac{-B \pm \sqrt{B^2 - 3AC}}{3A}$. We choose the smaller of the two values in the second term as we know just one root is in (a,b) and we are correcting from the midpoint of the interval.

The cubic search assumes that an initial bracket $[a,b]$ for the minimum of the function f is given. In the pseudocode in Figure 7.6, at startup we determine the values $f(a)$, $f'(a)$, $f(b)$ and $f'(b)$ and check that $f'(a) < 0$ and $f'(b) > 0$ reporting an error if these inequalities are not satisfied. The cubic search iteration begins by computing the location c of the unique minimum of the cubic Hermite interpolating function $P_3(x)$ that interpolates both $f(x)$ and $f'(x)$ at the endpoints a and b. If $f'(c) \approx 0$, then the cubic search stops because the point c is also the location of the minimum of the function f. Also, if $f(c) \geq \max\{f(a), f(b)\}$, we report an error because the function f cannot be U–shaped. If $f'(c) < 0$, then the cubic search moves the endpoint a to the point c by re-labeling c as a, and re-labeling $f(c)$ as $f(a)$. Similarly, if $f'(c) > 0$, then the cubic search moves the endpoint b to the point c. The process continues until the bracket on the location of the minimum is reduced to a length less than a user error tolerance.

Example 7.3.1. We find a local minimum of $f(x) = -e^{-x}\sin(x)$ on the interval $[a,b] = [0, 1.5]$ using cubic interpolation search. We terminate the iteration when the bracket is smaller than 10^{-5}. We implemented the pseudocode given in Figure 7.6 and the results are given in Table 7.3. Observe that the iteration terminates because at the fourth iteration (and thereafter) $f'(c)$ is zero to machine accuracy. If we don't include the check on $f'(c)$, the iteration never terminates since the calculated values of $f'(c)$ are very small and have the wrong sign. This results in the algorithm "shrinking" the interval very slowly from the incorrect end.

Problem 7.3.1. *Suppose f is a U–shaped function on $[a,b]$ with $f'(a)f'(b) \geq 0$. Show that f must have an endpoint extremum.*

$iterate$	$f(iterate)$	$f'(iterate)$	$error$
7.809528e-001	-3.223906e-001	-2.879105e-003	-4.445365e-003
7.857741e-001	-3.223969e-001	2.423153e-004	3.759439e-004
7.853982e-001	-3.223969e-001	-7.308578e-010	-1.252987e-009
7.853982e-001	-3.223969e-001	-5.551115e-017	-1.195117e-010

Table 7.4: Cubic interpolation search – iterates, function and derivative values and errors in iterates

Cubic Search

Input: function to minimize, $f(x)$
initial bracket, $[a, b]$

Output: scalars c, fc, where $fc = f(c) \approx \min_{a \le x \le b} f(x)$

$fa = f(a); fb = f(b); fap = f'(a); fbp = f'(b)$
$c = 0.5(a + b); fc = f(c); fcp = f'(c)$
if $fc \ge fa$ and $fc \ge fb$ then Stop
if $fap \ge 0$ or $fbp \le 0$ then Stop
$M = \max\{|fap|, |fbp|\}$
while termination criteria not satisfied do
  $d = 0.5(b - a)$
  $A = [d(fbp + fap) - (fb - fa)]/(4d^3)$
  $B = (fbp - fap)/(4d)$
  $C = [3(fb - fa) - d(fbp + fap)]/(4d)$
  $D = (fb + fa)/2 - d(fbp - fap)/4$
  if $B > 0$ then
    $c = 0.5(a + b) + (-B + \sqrt{B^2 - 3AC})/(3A)$
  else
    $c = 0.5(a + b) + (-B - \sqrt{B^2 - 3AC})/(3A)$
  endif
  $fc = f(c); fcp = f'(c)$
  if $|fcp| < roundtol \cdot M$ then
    Terminate with c, fc as output
  endif
  if $fc \ge fa$ and $fc \ge fb$ then Stop
  if $fcp > 0$ then
    $b = c; fb = fc; fbp = fcp$
  else
    $a = c; fa = fc; fap = fcp$
  endif
end while

Figure 7.6: Pseudocode *Cubic Search*. The initial choice of $c = 0.5 * (a + b)$ is arbitrary, but sensible.

Problem 7.3.2. *Suppose f is a U–shaped function on $[a, b]$ with $f'(a)f'(b) < 0$. Show that we must have $f'(a) < 0$ and $f'(b) > 0$.*

Problem 7.3.3. *Let f be a U–shaped function on $[a, b]$ with $f'(a) < 0$ and $f'(b) > 0$. Let $P_3(x)$ be the cubic Hermite polynomial that interpolates values $f(x)$ and $f'(x)$ at the endpoints a and b. In the Newton form, we can write its derivative*

$$P_3'(x) = g_0 + (x - a)g_1 + (x - a)(x - b)g_2$$

Show that the derivative interpolation conditions imply that $g_0 < 0$ and $g_1 > 0$. Assume that $g_2 \neq 0$. Observe that the roots of $P_3'(x) = 0$ are located at the points where the straight line $y = g_0 + (x - a)g_1$ intersects the parabola $y = -(x - a)(x - b)g_2$. Sketch this line and this parabola and show that, regardless of the value of the coefficient g_2, the line and the parabola always intersect at two distinct points. Show that only one of these intersection points lies in (a, b).

Problem 7.3.4. *Let f be a U–shaped function on $[a, b]$ with $f'(a) < 0$ and $f'(b) > 0$. Let $P_3(x)$ be the cubic Hermite polynomial that interpolates values $f(x)$ and $f'(x)$ at the endpoints a and b. By counting turning points, show that $P_3(x)$ has a unique minimum located at a point c in (a, b).*

Problem 7.3.5. *Let f be a U–shaped function on $[a, b]$ with $f'(a) < 0$ and $f'(b) > 0$. Let the function $Q(x) = P_3'(x)$ where $P_3(x)$ is the cubic Hermite polynomial that interpolates values $f(x)$ and $f'(x)$ at the endpoints a and b. Note that $Q(x)$ is a quadratic with $Q'(a) < 0$ and $Q'(b) > 0$.*

- *Use the Intermediate Value Theorem (see Problem 6.1.1) to show that there is a point $c \in (a, b)$ where $Q(c) = 0$.*
- *Use the Mean Value Theorem (see Problem 6.1.3) to show that there is a point $u \in (a, c)$ where $Q'(u) > 0$. Use the Mean Value Theorem again to show that there is a point $v \in (c, b)$ where $Q'(v) > 0$.*
- *Using the fact that $Q'(x)$ is a linear polynomial, and the facts established above show that $Q'(c) > 0$.*

Conclude that the cubic polynomial $P_3(x)$ has a unique minimum located at a point $c \in (a, b)$.

Problem 7.3.6. *Consider the U–shaped function*

$$f(x) = \frac{1}{x(1 - x)^2}$$

on $[a, b] = [0.01, 0.99]$. Using $c = 0.5$ as the initial value, carry out three iterations of cubic search for the minimum.

7.4 Matlab Notes

The MATLAB functions that are relevant to the topics discussed in this chapter include:

`fminbnd`	used to find a minimum of a single variable function
`optimset`	used to change default parameters (e.g., max number of iterations) for `fminbnd`

Before discussing these functions, though, we consider developing our own MATLAB implementations of some of the basic iterative optimization methods discussed in this chapter. Using the basic techniques developed for iterative methods in Section 6.6 and the pseudocodes presented in the previous section, this is not so difficult. We therefore leave these as exercises.

Problem 7.4.1. *Write a* MATLAB *function M–file that implements the golden section search method, and use it to find a local minimum of* $f(x) = e^{-x} - sin(x)$ *on the interval* $[0, 3]$. *Start with the bracket* $[1, 2]$ *and iterate until the length of the bracket is less than* 10^{-2}.

Problem 7.4.2. *Write a* MATLAB *function M–file that implements the quadratic interpolation search for the minimum, with the midpoint of* $[a, b]$ *as the initial point* c. *Report all applicable errors. Now consider using this to find a minimum of the function*

$$f(x) = \frac{x^4 + 1}{x^2 - 2x + 1}$$

on $[a, b] = [-10, 10]$. *Is this function U–shaped on the interval* $[-10, 10]$*?*

Problem 7.4.3. *Consider the U–shaped function*

$$f(x) = \begin{cases} 1 - x & \text{if } x \leq 0.9 \\ x - 0.8 & \text{otherwise} \end{cases}$$

on $[a, b] = [0, 1]$. *Use quadratic interpolation search for the minimum starting with the midpoint of* $[a, b]$ *as the initial point* c. *Report all applicable errors. Note that it may be easier to use a function M–file to implement* $f(x)$, *rather than an anonymous function, for this particular problem.*

Problem 7.4.4. *Use the quadratic interpolation search to find a local minimum of* $f(x) = e^{-x} - sin(x)$ *on the interval* $[0, 1]$. *Start with the bracket* $[1, 2]$ *and iterate until the length of the bracket is less than* 10^{-5}.

Problem 7.4.5. *Consider the U–shaped function*

$$f(x) = \begin{cases} 1 - x & \text{if } x \leq 0.9 \\ x - 0.8 & \text{otherwise} \end{cases}$$

on $[a, b] = [0, 1]$. *Use cubic interpolation to search for the minimum. Report all applicable errors. Note that it may be easier to use a function M–file to implement* $f(x)$, *rather than an anonymous function, for this particular problem.*

Problem 7.4.6. *Use the cubic interpolation search to find a local minimum of* $f(x) = e^{-x} - sin(x)$ *on the interval* $[0, 1]$. *Start with the bracket* $[1, 2]$ *and iterate until the length of the bracket is less than* 10^{-5}.

MATLAB's main built-in function for finding a local minimum of a function of one variable is `fminbnd`, which implements a combination of golden section and quadratic (parabolic) interpolation search. The basic usage is

```
x = fminbnd(fun, x1, x2)
```

where, as usual, `fun` can be an inline or anonymous function, or a handle to a function M–file, and `[x1, x2]` is an initial bracket of the minimum.

Example 7.4.1. To find the minimum of $f(x) = -e^{-x}\sin(x)$ on the interval $[0, 1.5]$, we can use the MATLAB statement:

```
x = fminbnd(@(x) -exp(-x).*sin(x), 0, 1.5)
```

The computed result is $x \approx 0.7854$, which is an approximation of $\frac{\pi}{4}$.

Example 7.4.2. Recall that $\max\{f(x)\} = \min\{-f(x)\}$. Thus, if we want to find the maximum of $f(x) = \dfrac{1}{x^2 - 6x + 8}$ on the interval $[0, 6]$, we can use the MATLAB statement:

```
x = fminbnd(@(x) -1./(x.*x-6*x+10), 0, 6)
```

which computes $x = 3$. It is not difficult to confirm that $f(3)$ is the maximum of $f(x)$ on the interval $[0, 6]$.

It is important to keep in mind that if the $f(x)$ is not U shaped, then `fminbnd` does not guarantee to find a global minimum on an interval, as illustrated in the following example.

Example 7.4.3. Consider the function $g(x) = x - \cos(7x)$ on the interval $[-4, 4]$. A plot of $g(x)$ is shown in Figure 7.7. Notice that the minimum is located approximately at the point $(-3.6109, -4.6006)$. If we compute

```
x = fminbnd(@(x) x-cos(7*x), -4, 4)
```

the result is $x \approx -0.9181$. Although this is a local minimum, it is not the global minimum of $f(x)$ on the interval $[-4, 4]$. If instead, we compute

```
x = fminbnd(@(x) x-cos(7*x), -4, 3)
```

the result is $x \approx -3.6109$, which is the global minimum. Interestingly, though, if we further refine the initial interval and compute

```
x = fminbnd(@(x) x-cos(7*x), -4, 2)
```

the result is $x \approx -2.7133$, yet another local minimum.

Problem 7.4.7. *Use* `fminbnd` *to compute a minimum of* $f(x) = x - \cos(7x)$ *with initial intervals* $[-4, z]$, *with* $z = 4, 3, 2, 1, 0, -1, -2, -3$. *What does* `fminbnd` *compute in each case?*

Problem 7.4.8. *Use* `fminbnd` *to find all local minima of* $f(x) = x^2 - \cos(4x)$. *What starting interval did you use in each case? Hint: You might want to first plot the function, and use it to determine appropriate starting intervals.*

As with `fzero` (see Section 6.6.4), it is possible to reset certain default parameters used by `fminbnd` (such as stopping tolerance and maximum number of iterations), and it is possible to obtain more information on exactly what is done during the computation of the minimum (such as number of iterations and number of function evaluations) by using the calling syntax

```
[x, fval, exitflag, output] = fminbnd(fun, x1, x2, options)
```

where

- `x` is the computed approximation of the local minimizer in the interval `[x1, x2]`.
- `fval` is the function value $f(x_{\min})$.
- `exitflag` provides information about what condition caused `fminbnd` to terminate; see the Help page on `fminbnd` for additional information.

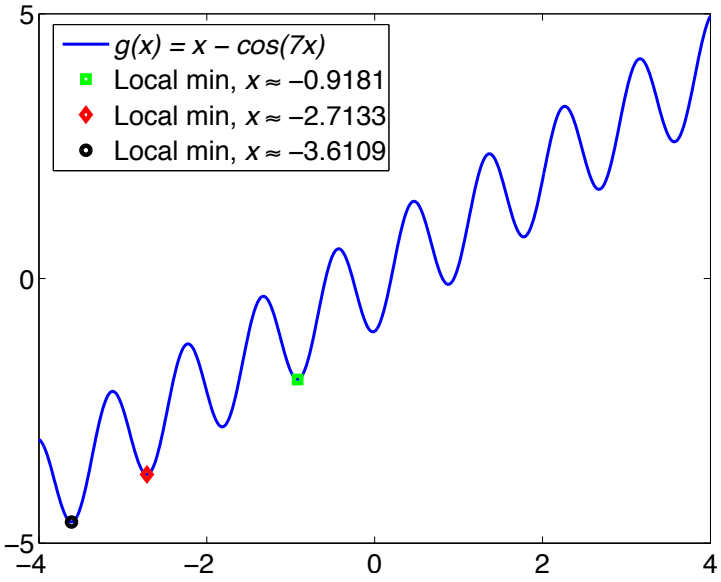

Figure 7.7: Plot of the function $g(x) = x - \cos(7x)$ on the interval $[-4, 4]$.

- `output` is a structure array that contains information such as which algorithms were used (e.g., golden section search or parabolic interpolation), number of function evaluations and number of iterations.
- `options` is an input parameter that can be used, for example, to reset the stopping tolerance, and to request that results of each iteration be displayed in the command window. The options are set using the built-in MATLAB function `optimset`.

The following example illustrates how to use `options` and `output`.

Example 7.4.4. Consider the function $f(x) = \dfrac{1}{x(1-x)^2}$. It is not difficult to show that $f(x)$ has a local minimum at $x = \frac{1}{3}$. Suppose we execute the MATLAB commands:

```
options = optimset('TolX', 1e-15, 'Display', 'iter');
[x, fval, exitflag, output] = fminbnd(@(x) 1./(x.*(1-x).^2), 0.001, 0.999, options);
```

Because we used `optimset` to set the `options` field to display the iterations, the following information is displayed in the command window:

```
Func-count     x          f(x)         Procedure
    1       0.382202       6.8551       initial
    2       0.617798      11.0807       golden
    3       0.236596      7.25244       golden
    4       0.334568      6.75007       parabolic
    5       0.336492      6.75045       parabolic
    6       0.333257         6.75       parabolic
    7       0.333332         6.75       parabolic
    8       0.333333         6.75       parabolic
    9       0.333333         6.75       parabolic
   10       0.333333         6.75       parabolic
   11       0.333333         6.75       parabolic
   12       0.333333         6.75       golden
```

```
    13       0.333333         6.75         parabolic

Optimization terminated:
the current x satisfies the termination criteria using OPTIONS.TolX of 1.000000e-15
```

which shows that some iterations perform a golden section search, while others perform a quadratic interpolation search. Quadratic search is preferred in cases where it will not lead to difficulties such as placing a new point too close to the endpoints. (The table uses `golden` to refer to golden section search, and `parabolic` to refer to quadratic interpolation search. The term `initial` simply indicates the initial choice which, in the case shown, is a golden section point.) We observe that a total of 13 function evaluations are needed to compute the minimum. The final six iterations all seem to be the same from the output but they are in fact spent computing the final value of `x`, not shown here, to the fifteen digits specified using `optimset`. If we display `output` it provides the following information:

```
iterations: 12
 funcCount: 13
 algorithm: 'golden section search, parabolic interpolation'
   message: [1x111 char]
```

The "message" can be viewed by simply entering the following command:

```
output.message
```

Problem 7.4.9. *Use* `fminbnd` *to find a minimum of* $f(x) = e^{-x} - \sin(x)$ *starting with the bracket* $[1, 2]$. *How many iterations used a golden section search, and how many used a parabolic interpolation search? Use the default* `TolX`. *Do the results change if* `TolX` *is changed to* 10^{-15} *?*

Problem 7.4.10. *Use* `fminbnd` *to find a minimum of* $f(x) = |x-7|$ *starting with the bracket* $[6, 8]$. *How many iterations used a golden section search, and how many used a parabolic interpolation search? Use the default* `TolX`. *Do the results change if* `TolX` *is changed to* 10^{-15} *?*

Problem 7.4.11. *Repeat Problem 7.4.7, and report how many iterations used a golden section search, and how many used a parabolic interpolation search. Do the results change if* `TolX` *is changed to* 10^{-15} *?*

Problem 7.4.12. *Repeat Problem 7.4.8, and report how many iterations used a golden section search, and how many used a parabolic interpolation search. Do the results change if* `TolX` *is changed to* 10^{-15} *?*

Problem 7.4.13. *Consider finding a minimum of* $f(x) = e^{-x} - \sin(x)$ *starting with the bracket* $[1, 2]$ *so that the final "bracket" is of length* 10^{-4}. *Implement*

1. *Golden Section search as given in the pseudocode in figure 7.4.*
2. *Quadratic Interpolation search as given in the pseudocode in figure 7.5.*
3. *Cubic Interpolation search as given in the pseudocode in figure 7.6.*

each in a MATLAB *M–file then solve this problem using your code. Compare the number of iterations needed in each case with those required by the* MATLAB *function* `fminbnd` *to solve the problem to the required accuracy.*