Math 222 Fall 2002—HW #3 Solutions

1. Consider the following three methods for locating a local minimum of a function $f(x)$ between $x = 0$ and $x = 1$ (we discussed each in class).

(a) Golden Section search

(b) Lee I: At step n, we have $0 \le a_n < c_n < b_n \le 1$, with c_n the midpoint of $[a_n, b_n]$, and $f(c_n) < f(a_n), f(b_n)$, so we know there is a local min between a_n and b_n. Now compute f at d_n and e_n, the midpoints of $[a_n, c_n]$ and $[c_n, b_n]$. For one of the following three intervals, f will be lower at the midpoint than at the endpoints: $[a_n, c_n]$, $[d_n, e_n]$, $[c_n, b_n]$. Keep this interval for the next iteration, calling its left endpoint a_{n+1}, its right endpoint b_{n+1} and its midpoint c_{n+1}.

(c) Lee II: Do the same thing as Lee I, but instead of computing $f(d_n)$ and $f(e_n)$ each time, first compute $f(d_n)$. If it is lower than $f(a_n)$ and $f(c_n)$, then accept $[a_n, c_n]$ as your new interval and go on to the next iteration. If not, compute $f(e_n)$, and accept either $[d_n, e_n]$ or $[c_n, b_n]$ as the new interval, depending on which has f at the midpoint lower than f at the endpoints.

At how many different points will we need to compute f in order for each algorithm to compute the local minimum to within 10^{-6}, assuming we start with $a_0 = 0$, $b_0 = 1$, and $c_0 = 1/2$? For (c), assume that, on average, 1/3 of the time $f(d_n)$ is lower than $f(a_n)$ and $f(c_n)$ (using the logic that of the 3 candidate intervals from Lee I, each is equally likely to be "the one"). So, the "average" number of f computations per step is $(1/3)(1) + (2/3)(2) = 5/3$.

For the golden section search, our initial "best guess" is the midpoint of $[0, 1]$, so our maximum possible error is 1/2. After one step, our new interval has width $r = (\sqrt{5} - 1)/2$, our best guess is the midpoint of that interval, so our maximum possible error is $r/2$. After n steps, our new interval has width r^n, our best guess is the midpoint of that interval, so our maximum possible error is $r^n/2$. Thus, the number of steps of the algorithm that we must complete in order to be sure the error is less than 10^{-6} is found by solving the inequality:

$$r^n/2 < 10^{-6},$$

or

$$r^n < 2 \times 10^{-6},$$

or

$$n \ln r < \ln(2 \times 10^{-6}),$$

or

$$n(-0.4812) < -13.1224,$$

or

$$n > 27.27.$$

Thus, we need to compute 28 steps of the algorithm. Each step requires one function evaluation, so, combined with the 4 required in the initial step, we have 32 total function evaluations.

For Lee I, our initial "best guess" is the midpoint of $[0,1]$, so our maximum possible error is $1/2$. After one step, our new interval has width $1/2$, our best guess is the midpoint of that interval, so our maximum possible error is $1/4$. After n steps, our new interval has width $1/2^n$, our best guess is the midpoint of that interval, so our maximum possible error is $1/2^{n+1}$. Thus, the number of steps of the algorithm that we must complete in order to be sure the error is less than 10^{-6} is found by solving the inequality:

$$1/2^{n+1} < 10^{-6},$$

or

$$(n+1)\ln(1/2) < \ln(10^{-6}),$$

or

$$n(-0.6931) < -13.8155$$

or

$$n > 19.93.$$

Thus, we need to compute 20 steps of the algorithm. Each step requires two function evaluations, so, combined with the 3 required in the initial step, we have 43 total function evaluations.

For Lee II, the logic in the number of steps is the same, so we need 20 steps. Each step takes on average 5/3 function evaluations, making a total of 33.3, or 34 function evaluations, which, when combined with the 3 required in the initial step, makes for 37 total function evaluations.

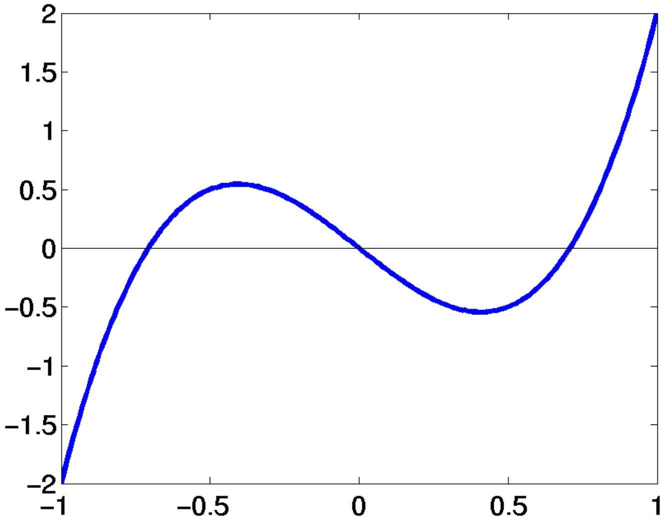
So, Lee just fails to beat the age-old golden section search.

2. Say we are looking for a local minimum of $f(x) = -x^2 + x^4 + 2$ by using Newton's Method. For which initial guesses will the iteration converge to the local *maximum* of f?

In order to seek a minimum of f using Newton's Method, we solve the equation $f' = 0$. So, we compute $f'(x) = -2x + 4x^3$. In order to perform Newton's Method on f', we will need its derivative, i.e., f''; we compute $f''(x) = -2 + 12x^2$. So, Newton's Method for solving $f' = 0$ is:

$$x_{n+1} = x_n - \frac{-2x_n + 4x_n^3}{-2 + 12x_n^2}$$

The graph of f' is given below:



For x close to zero, Newton's Method should converge to zero, which is the local maximum of f. How close do we need to be? As we've seen before, the cutoff occurs at the value of x that gets sent to $-x$ by Newton's Method, i.e., we seek to solve the equation:

$$-x_n = x_n - \frac{-2x_n + 4x_n^3}{-2 + 12x_n^2}$$

We can rewrite this as:

$$-2x_n = -\frac{-2x_n + 4x_n^3}{-2 + 12x_n^2},$$

or, since $x_n \neq 0$,

$$1 = -\frac{1 - 2x_n^2}{-2 + 12x_n^2}.$$

In other words,

$$12x_n^2 - 2 = 2x_n^2 - 1,$$

so that

$$10x_n^2 = 1,$$

or $x_n = \pm\sqrt{1/10}$. So, for $-\sqrt{1/10} < x_0 < \sqrt{1/10}$, we expect Newton's method to converge to the local maximum

3. Say we are looking for the minimum of $f(x, y) = x^2 + 4y^2$. For a current guess (x_n, y_n), work out on paper what the next guess (x_{n+1}, y_{n+1}) will be following the steepest descent algorithm

Using this result, starting at $(x_0, y_0) = (2, 3)$, find the first 5 steps of the steepest descent algorithm. Does the convergence appear to be linear? quadratic? How does the convergence compare to Newton's Method for this problem?

We compute:

$$\nabla f(x, y) = \begin{bmatrix} 2x \\ 8y \end{bmatrix}$$

So, in the steepest descent algorithm,

$$(x_{n+1}, y_{n+1}) = (x_n, y_n) - t(2x_n, 8y_n) = (x_n(1-2t), y_n(1-8t)),$$

where t is the real number that minimizes $f(x_n(1-2t), y_n(1-8t)) = x_n^2(1-2t)^2 + 4y_n^2(1-8t)^2$. To find this t, we set the derivative of this expression equal to zero:

$$\begin{aligned} 2x_n^2(1-2t)(-2) + 8y_n^2(1-8t)(-8) &= 0 \\ x_n^2(1-2t) + 16y_n^2(1-8t) &= 0 \\ x_n^2 - 2tx_n^2 + 16y_n^2 - 128ty_n^2 &= 0 \\ t &= \frac{x_n^2 + 16y_n^2}{2x_n^2 + 128y_n^2} \end{aligned}$$

Thus,

$$\begin{aligned} (x_{n+1}, y_{n+1}) &= \left(x_n \left(1 - 2\frac{x_n^2 + 16y_n^2}{2x_n^2 + 128y_n^2} \right), y_n \left(1 - 8\frac{x_n^2 + 16y_n^2}{2x_n^2 + 128y_n^2} \right) \right) \\ &= \left(x_n \left(\frac{96y_n^2}{2x_n^2 + 128y_n^2} \right), y_n \left(-\frac{6x_n^2}{2x_n^2 + 128y_n^2} \right) \right) = \left(\frac{48x_ny_n^2}{x_n^2 + 64y_n^2}, -\frac{3x_n^2y_n}{x_n^2 + 64y_n^2} \right) \end{aligned}$$

Applying this formula repeatedly, beginning with $(x_0, y_0) = (2, 3)$, we find:

n	x_n	y_n	$f(x_n, y_n)$
0	2	3	40
1	1.48966	-0.06207	2.23448
2	0.11172	0.16759	0.12482
3	0.08322	-0.00347	0.00697
4	0.00624	0.00936	0.00039
5	0.00465	-0.00019	0.00002

The convergence certainly does not look quadratic: we do not double the number of digits of agreement between (x_n, y_n) and the true solution $(0, 0)$ in each step. Really, it looks linear; I computed the ratio $\frac{\|(x_{n+1}, y_{n+1}) - (0,0)\|}{\|(x_n, y_n) - (0,0)\|}$ and found it alternated in the pattern $0.41, 0.14, 0.41, 0.14, \ldots$. This is consistent with linear convergence:

$$\|(x_{n+1}, y_{n+1}) - (0,0)\| \le C\|(x_n, y_n) - (0,0)\|$$

for some $C < 1$ (e.g., $C = 0.41$), but is inconsistent with quadratic convergence (in which case the computed ratio should go to zero).

In contrast, Newton's Method always converges to $(0, 0)$ in a single step no matter what the initial condition for this function. Why is that? Well, in using Newton's Method to minimize a function, we are really setting the gradient vector ∇f to zero, so the algorithm is:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} - (H_f(x_n, y_n))^{-1} \nabla f(x_n, y_n)$$

where H_f is the Hessian of f. Here, $H_f = \begin{bmatrix} 2 & 0 \\ 0 & 8 \end{bmatrix}$. So,

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} - \begin{bmatrix} 2 & 0 \\ 0 & 8 \end{bmatrix}^{-1} \begin{bmatrix} 2x_n \\ 8y_n \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} - \begin{bmatrix} 1/2 & 0 \\ 0 & 1/8 \end{bmatrix} \begin{bmatrix} 2x_n \\ 8y_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

So, Newton's Method is a lot better than steepest descent in this case.