METHOD OF QUADRATIC INTERPOLATION

KELLER VANDEBOGERT

1. Introduction

Interpolation methods are a common approach to the more general area of line search for optimization. In the case of quadratic interpolation, the function's critical value is bracketed, and a quadratic interpolant is fitted to the arc contained in the interval. Then, the interpolant is minimized, and the new interval is determined based on the relation of the minimizer to the original endpoints of the interval.

Put more formally, let x^* maximize (or minimize) $f(x)$. If x^* is not easily found through analytic methods, then it is significantly easier to bracket the interval over which this critical point occurs. Let $q(x)$ denote the quadratic interpolant of $f(x)$. Minimizing a quadratic function is trivial, and so the critical point of q is easily obtained. We then form a new bracketing interval by throwing away the "worst" point, which for our purposes would be the point that is the largest or smallest, depending on whether we want to approximate a maximum or minimum. The process is then iterated until a desired precision is reached.

Date: September 3, 2017.

2. Methods of Interpolation

In practice there are 3 methods of interpolation. There are 2 types of 2-point interpolation methods, and a 3-point interpolation method. The 2-point methods require knowledge of the derivative of the function f in which we are interested in optimizing. The 3-point method does not require any derivatives, but of course requires an extra point. Intuitively, knowing f' gives a better sense of the function's behavior, and will hence provide a faster rate of convergence.

2.1. **Method 1.** Let $q(x)$ denote the quadratic interpolant of $f(x)$. Then, by definition:

$$q(x) = ax^2 + bx + c$$

For a, b, $c \in \mathbb{R}$. Then, we can find our constants by bracketing the critical point of f, whose endpoints are x_1 and x_2. We have:

$$f(x_1) = ax_1^2 + bx_1 + c = q(x_1) \tag{2.1}$$

$$f(x_2) = ax_2^2 + bx_2 + c = q(x_2)$$

$$f'(x_1) = 2ax_1 + b = q'(x_1)$$

This is a system of 3 equations with 3 unknowns, which are our constants. Let $f_i = f(x_i)$, and $f_i' = f'(x_i)$. Then we can solve (2.1) for a and b.

$$a = \frac{f_1 - f_2 - f_1'(x_1 - x_2)}{-(x_1 - x_2)^2} \tag{2.2}$$

$$b = f_1' + 2\frac{f_1 - f_2 - f_1'(x_1 - x_2)}{(x_1 - x_2)^2}x_1$$

The minimizer of q is easily found to be $-b/2a$ by setting $q'(x) = 0$. From (2.2), our minimizer x_{min} can be found:

$$x_{min} = \frac{-b}{2a} = x_1 - \frac{1}{2}\frac{(x_1 - x_2)f_1'}{f_1' - \frac{f_1 - f_2}{x_1 - x_2}} \tag{2.3}$$

This of course readily yields an explicit iteration formula by letting $x_{min} = x_3$. We have from (2.3):

$$x_{k+1} = x_{k-1} - \frac{1}{2}\frac{(x_{k-1} - x_k)f_{k-1}'}{f_{k-1}' - \frac{f_{k-1} - f_k}{x_{k-1} - x_k}} \tag{2.4}$$

With (2.4), we generate x_{k+1} and compare it with the previous two points to find our new bracketing interval, and then continue the iteration.

2.2. **Method 2.** This method is very similar to our first method but instead generates $q(x)$ by solving a different set of equations. We have:

$$\begin{aligned} f(x_1) &= ax_1^2 + bx_1 + c = q(x_1) \\ f'(x_1) &= 2ax_1 + b = q'(x_1) \\ f'(x_2) &= 2ax_2 + b = q'(x_2) \end{aligned} \tag{2.5}$$

We can of course solve these for a and b in the same way as in Section 2.1. We find the explicit form of the minimizer of $q(x)$ to be:

$$x_{min} = x_1 - \frac{x_1 - x_2}{f_1' - f_2'}f_1' \tag{2.6}$$

In an identical manner to (2.4), we generate a recursion by defining $x_{min} = x_3$, and we have:

$$x_{k+1} = x_{k-1} - \frac{x_{k-1} - x_k}{f'_{k-1} - f'_k} f'_{k-1} \tag{2.7}$$

Which is commonly called the *secant formula*.

Remark 2.1. Letting $x_{k-1} \to x_k$ in (2.7), and assuming that $f''(x_k)$ exists, (2.7) becomes:

$$x_{k+1} = x_k - \frac{f'_k}{f''_k}$$

But this is precisely the iteration defined by Newton's method. This motivates calling (2.7) the secant method, because it is just Newton's method with the secant approximation of $f''(x_k)$ instead.

2.3. **Method 3.** Our third method is the 3 point method. Choose 3 points, 2 endpoints to bracket our critical point, and then a point within the interval as well.

Using the *Lagrange Interpolation formula*, we can easily find our interpolant $q(x)$. We have:

$$q(x) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} f_1 + \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} f_2 + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} f_3 \tag{2.8}$$

To find the minimum, we of course take the derivative of (2.8) and set it to 0. By doing this, we obtain:

$$x_{min} = \frac{1}{2}(x_1 + x_2) + \frac{1}{2} \frac{(f_1 - f_2)(f_2 - f_3)(f_3 - f_1)}{(x_2 - x_3)f_1 + (x_3 - x_1)f_2 + (x_1 - x_2)f_3} \tag{2.9}$$

And the iteration scheme is easily deduced:

(2.10)
$$x_{k+2} = \frac{1}{2}(x_{k-1}+x_k)+\frac{1}{2}\frac{(f_{k-1}-f_k)(f_k-f_{k+1})(f_{k+1}-f_{k-1})}{(x_k-x_{k+1})f_{k-1}+(x_{k+1}-x_{k-1})f_k+(x_{k-1}-x_k)f_{k+1}}$$

This method differs slightly from the previous two methods, because it is not as simple to determine the new bracketing interval. If x_{min} lies between x_1 and x_3, then we want to compare the distance between x_{min} and x_2. If

$$|x_{min} - x_2| < \epsilon$$

Where ϵ is the prescribed tolerance, then we are done. Otherwise we create the new interval by checking which point is the largest or smallest, depending on the nature of our critical point.

If x_{min} lies outside of our bracketing interval $[x_1, x_3]$, then we immediately create a new bracketing interval in the same way as we just described.

3. Convergence Rates

In the beginning of Section 2, we made a comment on the convergence rates of the 2-point versus 3-point method. In this section, we intend to make this comment precise.

Definition 3.1. Let $x_n \to x^*$. Then, the sequence $\{x_n\}$ is said to converge to x^* with order α if

$$||x_n - x^*|| \leq M||x_{n-1} - x^*||^\alpha$$

For some constant $M \in \mathbb{R}$.

We will also need the following result, which we shall state without proof:

Lemma 3.2. *Let $f \in \mathcal{C}^{n+1}[a,b]$, and let p be the polynomial of degree $\leq n$ that interpolates f at $n+1$ distinct points x_0, x_1, ..., $x_n \in [a,b]$. Then, to each $x \in [a,b]$ there exists a $\xi_x \in (a,b)$ such that*

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi_x)}{(n+1)!}\prod_{i=0}^{k}(x - x_i)$$

Where $f^{(n)}(x)$ denotes the nth derivative of f. This remainder term will often be denoted R_{n+1}.

Theorem 3.3. *Let $f : \mathbb{R} \to \mathbb{R}$ be 3 times continuously differentiable. Let x^* be such that $f'(x^*) = 0$ and $f''(x^*) \neq 0$. Then the sequence $\{x_n\}$ generated by (2.7) converges to x^* with order $\frac{1+\sqrt{5}}{2}$.*

Proof. We first want to prove that (2.7) does indeed converge to our minimizer, x^*.

Let $L(x)$ be the 2 point interpolant for $f'(x)$. Then, with Lemma 3.2, we have:

$$f'(x) - L(x) = \frac{f'''(\xi)}{2}(x - x_{k-1})(x - x_k)$$

Since x_{k+1} is generated by maximizing $q(x)$, we have that $L(x_{k+1}) = 0$. Thus,

$$f'(x_{k+1}) = \frac{f'''(\xi)}{2}(x_{k+1} - x_{k-1})(x_{k+1} - x_k)$$

If we substitute the recursion for x_{k+1} given by (2.7) into the above equation, we can simplify this into:

$$f'(x_{k+1}) = \frac{1}{2} f'''(\xi) f'_k f'_{k-1} \frac{(x_k - x_{k-1})^2}{(f'_k - f'_{k-1})^2} \tag{3.1}$$

We now want to take advantage of the Mean Value Theorem. We have:

$$\frac{f'_k - f'_{k-1}}{x_k - x_{k-1}} = f''\xi_0$$

Where $\xi_0 \in (x_k, x_{k-1})$. Also note that since $f'(x^*) = 0$, we have:

$$f'_i = f'(x_i) - f'(x^*) = (x_i - x^*) f''(\xi_i) \tag{3.2}$$

For some $\xi_i \in (x_i, x^*)$, $i = k-1,\ k,\ k+1$. Using (3.2) and the previous line, we can rewrite (3.1) as so:

$$x_{k+1} - x^* = \frac{1}{2} \frac{f'''(\xi) f''(\xi_k) f''(\xi_{k-1})}{f''(\xi_{k+1})[f(\xi_0)]^2} (x_k - x^*)(x_{k-1} - x^*) \tag{3.3}$$

Now, let $e_i = |x_i - x^*|$, $i = k-1,\ k,\ k+1$. Find constants m_1, m_2, M_1, M_2 such that

$$0 < m_1 \leq |f'''(x)| \leq M_1$$

$$0 < m_2 \leq |f''(x)| \leq M_2$$

where x is any value in the bracketing interval considered. Then, we can bound (3.3):

$$\frac{m_1 m_2^2}{2M_2^3} e_k e_{k-1} \leq e_{k+1} \leq \frac{M_1 M_2^2}{2m_2^3} e_k e_{k-1} \tag{3.4}$$

However, using (3.3):

$$\frac{e_{k+1}}{e_k e_{k-1}} = \frac{1}{2}\frac{f'''(\xi)f''(\xi_k)f''(\xi_{k-1})}{f''(\xi_{k+1})[f(\xi_0)]^2}$$

Now if we let each $x_i \to x^*$, each ξ_i will tend toward x^* as well since they are contained within their respective bracketing intervals and f'', f''' are continuous. Thus we see:

$$\frac{e_{k+1}}{e_k e_{k-1}} \to \frac{f'''(x^*)}{2f''(x^*)} \tag{3.5}$$

This is well-defined since $f''(x^*) \neq 0$ by assumption. Define $M = \left|\frac{f'''(x^*)}{2f''(x^*)}\right|$, and using (3.4) and (3.5), we have that

$$e_{k+1} \leq M e_k e_{k-1} \tag{3.6}$$

In which case if $\delta = \max\{e_k, e_{k-1}\}$,

$$e_{k+1} \leq M\delta^2 \to 0$$

So our sequence converges to x^* when $x_0 \neq x_1$. Now we seek to determine the rate of convergence. To do this, we define $y_k = \log(Me_k)$. We then have, using (3.6):

$$y_{k+1} = y_k + y_{k-1}$$

This is of course a simple a recurrence relation. Let $\phi = \frac{1+\sqrt{5}}{2}$ and $\bar{\phi} = \frac{1-\sqrt{5}}{2}$. We easily find the closed form of y_k to be:

$$y_k = \alpha\phi^k + \beta\bar{\phi}^k \tag{3.7}$$

Where α, β are to be determined from our initial conditions. Since $\bar{\phi} < 1$, for k sufficiently large, $y_k \approx \alpha\phi^k$. We then have, as $k \to \infty$:

$$\frac{Me_{k+1}}{M^{\phi}e_k^{\phi}} \approx \frac{\exp(\alpha\phi^{k+1})}{\exp(\alpha\phi^{k\phi})} \to 1$$

With this:

$$\frac{|x_{k+1} - x^*|}{|x_k - x^*|^{\phi}} \to M^{\phi-1}$$

And with Definition 3.1, this implies that $x_k \to x^*$ with order $\phi = \frac{1+\sqrt{5}}{2}$.

□

Remark 3.4. We see that the secant method converges at a superlinear rate, whereas Newton's method converges quadratically under suitable conditions.

We now seek to find the rate of convergence of the 3-point method. The following result answers this question:

Theorem 3.5. *Let $f \in \mathcal{C}^4$. Suppose that x^* satisfies $f'(x^*) = 0$ and $f''(x^*) \neq 0$. Then the sequence $\{x_k\}$ generated by* (2.10) *converges to x^* of order* 1.32.

Proof. Similar to Theorem 3.3, we begin by writing our function f in the following form:

$$f(x) = L(x) + R_3(x) \tag{3.8}$$

Where R_3 in the third degree remainder term for the Lagrange interpolant. Now, since x^* is a critical point, we clearly have that $f'(x^*) = 0$. With this and (3.8):

$$L'(x^*) + R_3'(x^*) = 0$$

$L'(x^*)$ can be easily computed, we have:

(3.9)
$$f_1\frac{2x^* - (x_2+x_3)}{(x_1-x_2)(x_1-x_3)} + f_2\frac{2x^* - (x_1+x_3)}{(x_2-x_1)(x_2-x_3)} + f_3\frac{2x^* - (x_1+x_2)}{(x_3-x_1)(x_3-x_2)} + R_3'(x^*) = 0$$

We now use (3.9) to solve for x^*:

$$2x^*\left(\frac{f_1}{(x_1-x_2)(x_1-x_3)} + \frac{f_2}{(x_2-x_1)(x_2-x_3)} + \frac{f_3}{(x_3-x_1)(x_3-x_1)}\right) + R_3'(x^*)$$
$$= \frac{f_1(x_2+x_3)}{(x_1-x_2)(x_1-x_3)} + \frac{f_2(x_1+x_3)}{(x_2-x_1)(x_2-x_3)} + \frac{f_3(x_1+x_2)}{(x_3-x_1)(x_3-x_2)}$$

Which then gives:

$$x^* = -\frac{1}{2}\frac{R_3'(x^*)}{\frac{f_1}{(x_1-x_2)(x_1-x_3)} + \frac{f_2}{(x_2-x_1)(x_2-x_3)} + \frac{f_3}{(x_3-x_1)(x_3-x_1)}} + \frac{1}{2}\left(\frac{\frac{f_1(x_2+x_3)}{(x_1-x_2)(x_1-x_3)} + \frac{f_2(x_1+x_3)}{(x_2-x_1)(x_2-x_3)} + \frac{f_3(x_1+x_2)}{(x_3-x_1)(x_3-x_2)}}{\frac{f_1}{(x_1-x_2)(x_1-x_3)} + \frac{f_2}{(x_2-x_1)(x_2-x_3)} + \frac{f_3}{(x_3-x_1)(x_3-x_1)}}\right) \tag{3.10}$$

However, it is easy to see that we can use (2.10) and rewrite it in an awfully convenient form:

$$x_4 = \frac{1}{2}\frac{\frac{f_1(x_2+x_3)}{(x_1-x_2)(x_1-x_3)} + \frac{f_2(x_1+x_3)}{(x_2-x_1)(x_2-x_3)} + \frac{f_3(x_1+x_2)}{(x_3-x_1)(x_3-x_2)}}{\frac{f_1}{(x_1-x_2)(x_1-x_3)} + \frac{f_2}{(x_2-x_1)(x_2-x_3)} + \frac{f_3}{(x_3-x_1)(x_3-x_1)}}$$

But this is precisely the rightmost term of (3.10), so we easily deduce:

$$x^* - x_4 = -\frac{1}{2}\frac{R_3'(x^*)}{\frac{f_1}{(x_1-x_2)(x_1-x_3)} + \frac{f_2}{(x_2-x_1)(x_2-x_3)} + \frac{f_3}{(x_3-x_1)(x_3-x_1)}} \tag{3.11}$$

Now, similar to the proof of Theorem 3.3, let $e_i = x^* - x_i$, where $i = 1, 2, 3, 4$. With elementary manipulations of (3.11), we find the following form:

(3.12)
$$e_4\big(f_1(e_2-e_3)+f_2(e_3-e_1)+f_3(e_1-e_2)\big) = -\frac{1}{2}R_3'(x^*)(e_1-e_2)(e_2-e_3)(e_3-e_1)$$

By means of the Taylor expansion about x^*, it is clear that

$$f_i = f(x^*) + \frac{1}{2}e_i^2 f''(x^*) + \mathcal{O}(e_i^3)$$

Where we've used the fact that $f'(x^*) = 0$. Now, for e_i sufficiently small, we have neglect the third order term of the Taylor expansion. Substituting each respective Taylor expansion into (3.12), we get the following:

$$e_4\Big(\big(f(x^*)+\frac{1}{2}e_1^2f''(x^*)\big)(e_2-e_3)+\big(f(x^*)+\frac{1}{2}e_2^2f''(x^*)\big)(e_3-e_1)+\big(f(x^*)+\frac{1}{2}e_3^2f''(x^*)\big)(e_1-e_2)\Big)$$
$$= -\frac{1}{2}R_3'(x^*)(e_1-e_2)(e_2-e_3)(e_3-e_1)$$

Now examine the coefficient of e_4 in the above expression. We find:

(3.13)
$$\begin{aligned}
&\big(f(x^*)+\frac{1}{2}e_1^2f''(x^*)\big)(e_2-e_3)+\big(f(x^*)+\frac{1}{2}e_2^2f''(x^*)\big)(e_3-e_1)+\big(f(x^*)+\frac{1}{2}e_3^2f''(x^*)\big)(e_1-e_2)\big)\\
&\quad= \frac{1}{2}f''(x^*)\Big(e_1^2(e_2-e_3)+e_2^2(e_3-e_1)+e_3^2(e_1-e_2)\Big)\\
&\quad= \frac{1}{2}f''(x^*)\Big((e_2^2-e_1^2)e_3+(e_3^2-e_1^2)e_2+(e_3^2-e_2^2)e_1\Big)\\
&\qquad= \frac{1}{2}f''(x^*)(e_1-e_2)(e_2-e_3)(e_3-e_1)
\end{aligned}$$

Using (3.12) and (3.13), we see:

$$e_4 = -\frac{1}{f''(x^*)} R_3'(x^*) \tag{3.14}$$

We now seek an explicit form of the derivative of our remainder term. From the form given by Lemma 3.2, it can be found that

$$R_3'(x^*) = -\frac{1}{6} f'''(\xi_{x^*})(e_1e_2 + e_2e_3 + e_3e_1) + \frac{1}{24} f^{(4)}(\eta) e_1e_2e_3 \tag{3.15}$$

Where η is some constant. We now want to to neglect the fourth order term, because the fourth derivative is bounded and the product $e_1e_2e_3$ will become arbitrarily small much faster than the first term. Using (3.15) and (3.14):

$$e_4 = \frac{f'''(\xi_{x^*}}{6f''(x^*)}(e_1e_2 + e_2e_3 + e_3e_1) \tag{3.16}$$

Which easily generalizes to:

$$e_{k+2} = M(e_{k-1}e_k + e_ke_{k+1} + e_{k+1}e_{k-1}) \tag{3.17}$$

Where we define $M = -\frac{f'''(\xi_{x^*}}{6f''(x^*)}$. It is also clear that for sufficiently small e_k, $e_{k+1} = \mathcal{O}(e_k) = \mathcal{O}(e_{k-1})$. Thus, it is possible to find some constant, $\bar{M}$, such that:

$$|e_{k+2}| \leq \bar{M}|e_{k-1}||e_k|$$

In a similar fashion to the proof for Theorem 3.3, we can easily define $\delta_i = \log(\bar{M}|e_i|)$. By doing so, we easily see that:

$$\delta_{k+2} \leq \delta_k + \delta_{k-1} \tag{3.18}$$

Now tend k sufficiently large. Then, it is clear that $\delta_{k+2} \approx \delta_k + \delta_{k-1}$. With this and (3.18) we spawn the following recurrence:

$$\delta_{k+2} = \delta_k + \delta_{k-1}$$

This recurrence has a solution of the form:

$$\delta_k = \alpha\tau_1^k + \beta\tau_2^k + \gamma\tau_3^k \tag{3.19}$$

Where each τ_i satisfies the characteristic equation $\tau^3 - \tau - 1 = 0$, $i = 1, 2, 3$, and our constants are to be determined by initial conditions. Let τ_1 denote the real root of this characteristic equation. Then, by examining the roots of this equation, we see that $\tau_2 = \bar{\tau_3}$, complex conjugates, and that $|\tau_2| = |\tau_3| < 1$. Thus for k sufficiently large, $q_k \approx \alpha\tau_1^k$.

Now examine $\delta_k - \tau_1\delta_{k-1}$. By (3.19), we can clearly see that this must tend to 0. However, using our definition for δ_k:

$$\delta_{k+1} - \tau_1\delta_k \to 0 \implies \frac{\bar{M}|e_{k+1}|}{\bar{M}^{\tau_1}|e_k|^{\tau_1}} \to 1$$

But then we have that

$$\frac{|x_k - x^*|}{|x_{k-1} - x^*|^{\tau_1}} \to \bar{M}^{\tau_1 - 1} \tag{3.20}$$

But of course by Definition 3.1, this implies $x_k \to x^*$ with order τ_1. Numerically, we see that $\tau_1 \approx 1.3247$, and we are done.

□

From these two results we see that having knowledge of the derivative of f does indeed allow for a more accurate interpolant and thus a faster rate of convergence to the minimizer x^*.

4. Numerical Tests

A selection of interesting and nonelementary functions were chosen to optimize. The material below gives the function, along with the estimated critical points solved in MATLAB. The rightmost results are found with Method 2 (secant method), and the leftmost results are found with Method 3 (3-point). The appendix will contain the MATLAB code used for each method, and some discussion on possible improvements. Our allowed tolerance was 10^{-32}, or machine precision.

(1) $f(x) = e^{-2x} + x^2$

```
Estimated Critical Point:
   0.608036786522882

at the point x =
   0.426302751006863

Iterations:
     10
```

```
Estimated Critical Point:
   0.608036786522882

at the point x =
   0.426302751032028

Iterations:
      9
```

(2) $f(x) = -2e^{-\sqrt{x}}(\sqrt{x}+1) + \cos(x)$

```
Estimated Critical Point:
  -0.805783106429131

at the point x =
   0.511180626904120

Iterations:
      8
```

```
Estimated Critical Point:
  -0.805783106429131

at the point x =
   0.511180622079723

Iterations:
     36
```

(3) $f(x) = \frac{1}{720}\Big(x^6 - 36x^5 + 450x^4 - 2400x^3 + 5400x^2 - 4320x + 720\Big)$
(The 6th Laguerre Polynomial)

```
Estimated Critical Point:
  -0.543599590009618

at the point x =
   0.617030853278270

Iterations:
     10
```

```
Estimated Critical Point:
  -0.543599590009618

at the point x =
   0.617030850410704

Iterations:
      8
```

(4) $f(x) = \frac{1}{\Gamma(x)}$ (Reciprocal of the Gamma Function)

```
Estimated Critical Point:
   1.129173885450141

at the point x =
   1.461632144968362

Iterations:
      9
```

```
Estimated Critical Point:
   1.129173885450141

at the point x =
   1.461632155545645

Iterations:
     15
```

(5) $f(x) = 64x^7 - 112x^5 + 56x^3 - 7x$ (The 7th Chebyshev Polynomial)

```
Estimated Critical Point:
    -1

at the point x =
   0.222520933956314

Iterations:
     8
```

```
Estimated Critical Point:
    -1

at the point x =
   0.222520933812935

Iterations:
    20
```

(6) $f(x) = x\Big(\log(x) - 1\Big) - \sin(x)$

```
Estimated Critical Point:
  -1.922492250332938

at the point x =
   1.302964001216013

Iterations:
     7
```

```
Estimated Critical Point:
  -1.922492250332938

at the point x =
   1.302964006102710

Iterations:
     6
```

(7) $f(x) = -x + e^{-x} + x\log(x)$

```
Estimated Critical Point:
  -0.686444414617745

at the point x =
   1.309799585804151

Iterations:
     8
```

```
Estimated Critical Point:
  -0.686444414617745

at the point x =
   1.309799586743216

Iterations:
    11
```

(8) $f(x) = -\mathrm{li}(x) + x\log(\log(x)) + \cos(x)$ ($\mathrm{li}(x)$ denotes the Logarithmic Integral)

```
Estimated Critical Point:
  -2.872294656399205

at the point x =
   3.036425545348658

Iterations:
     6
```

```
Estimated Critical Point:
  -2.872294656399204

at the point x =
   3.036425539751614

Iterations:
    15
```

(9) $f(x) = \frac{1}{2}\sqrt{\pi}\text{erf}(x) - \frac{x^3}{3}$ (erf(x) denotes the Error Function)

```
Estimated Critical Point:
   0.489631524205925

at the point x =
   0.753089164979675

Iterations:
      8
```

```
Estimated Critical Point:
   0.489631524205925

at the point x =
   0.753089162650208

Iterations:
     24
```

(10) $f(x) = \frac{1}{2}\sqrt{\pi}\text{erf}(x) - \sin(x)$

```
Estimated Critical Point:
  -0.142207548300912

at the point x =
   1.447414271296237

Iterations:
      8
```

```
Estimated Critical Point:
  -0.142207548300912

at the point x =
   1.447414272474958

Iterations:
     16
```

It is interesting to note some of the discrepancies in iteration. Indeed the 3-point method took almost twice as many iterations as the secant method on average. However, this could be due to many factors. Firstly, the use of nonelementary functions perhaps made the 3-point interpolant less accurate than the secant method, which took advantage of the function's derivative.

It would be expected that the 3-point interpolant could be as good as or possibly better than the secant method for polynomials. For function (3), this holds, however, when looking at (5), we see that it took significantly more iterations for just a simple polynomial. We also note the largest discrepancy in iterations occurred for function (2). This is an interesting case because we are only using transcendental functions, as opposed to some of the other nonelementary functions.

5. Conclusion

We explored the method of quadratic interpolation for optimization and some of the different approaches to actually find our quadratic interpolant. Analytically, it was shown that the secant method should in general converge more quickly than the 3-point method. This is intuitively explained by the fact that the derivative of a function gives information on that function's curvature. After this, numerical tests were shown on a wide variety of functions and on average the secant method did significantly better than was theoretically predicted.

Appendix

Two codes were written in MATLAB for Section 4 on numerical testing. In this appendix, the code will be provided and some dicussion on possible improvements as well.

Secant Method.

```
clear
clc
format long

%f = function to optimize
%Df = derivative of f
x=1;
f = inline('Your Function Here');
Df = inline('Derivative of f');

%xstart and xnew are an initial bracketing interval
xstart = 1;
xnew = 2;
it = 0;

while abs(xstart - xnew) > 10^(-32)
    xnew2 = xnew - (xnew - xstart)/(Df(xnew)-Df(xstart))*Df(xnew);
    xstart = xnew;
    xnew = xnew2;
    it = it+1;
end

disp('Estimated Critical Point:')
disp(f(xnew))
disp('at the point x =')
disp(xnew)
disp('Iterations:')
disp(it)
```

Figure 1. The Secant Method

This code is actually pretty compact and runs quickly. It would be useful to input some kind of searching mechanism which can actually determine the bracketing interval for the user, whereas this code requires the user to find their own bracketing interval. Also, in many cases, if your critical point is not contained within the initial interval,

this programs will still find the optimizer, but it takes more iterations than normal.

3-Point Method.

```
clear
clc
format long

%Initial bracketing interval
x = [x1 x2 x3];

f = inline('Your Function Here');

%Find interpolating polynomial
interp = polyfit(x,f(x),2);
x = [x(2),x(3),-interp(2)/2/interp(1)];
it=1;

while abs(x(3)-x(2))>10^(-32) && abs(f(x(3))-f(x(2)))>10^(-32)
    interp = polyfit(x,f(x),2);
    x = [x(2),x(3),-interp(2)/2/interp(1)];
    it=it+1;
end

disp('Estimated Critical Point:')
disp(f(x(2)))
disp('at the point x =')
disp(x(2))
disp('Iterations:')
disp(it)
disp('Iterations:')
```

FIGURE 2. 3-Point Method

This code is not a perfect recreation of the theoretical 3-Point Method, because it does not search both ends of the bracketing interval to find the new bracketing interval. This is one major reason why it is possible that there was a larger than theoretically expected discrepancy between the amount of iterations in Section 4. For a code that compares both ends of the interval, we would find that each consecutive interval is more accurate. Another point of this is that the points chosen for the interval were always uniformly spaced. An interesting consideration is to find a bracketing interval, and then interpolate the *Chebyshev nodes* as opposed to just the uniformly spaced nodes. This special set of nodes will provide the best possible polynomial interpolant over any

given interval. Also, another possible reason for the discrepancy in iterations is the tolerance values. Choosing the tolerance too small in some cases would cause the bracketing interval to become too small and it was difficult for MATLAB to distinguish between points when the tolerance was right at machine precision.

References

[1] *Optimization Theory and Methods: Nonlinear Programming.* Wenyu Sun, Ya-Xiang Yuan. 89-98.