A Hybrid Method For Solving A Single Nonlinear Equation

by

Jonathan H. Whitacre

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Masters

in the

Mathematics

Program

YOUNGSTOWN STATE UNIVERSITY

December, 2010

A Hybrid Method For Solving A Single Nonlinear Equation

Jonathan H. Whitacre

Signature:

Jonathan H. Whitacre, Student Date

Approvals:

Jozsi Jalics, Ph.D, Thesis Advisor Date

Richard Burden, Ph.D, Committee Member Date

J. D. Faires, Ph.D, Committee Member Date

Peter J. Kasvinsky, Dean of School of Graduate Studies & Research Date

ABSTRACT

The purpose of this paper is to develop a root finding method for non-linear functions. The problem, $f(x) = 0$ where $x \in \Re$, is common in many areas of mathematics and can be traced back as far as 1700 B.C. A cuneiform table in the Yale Babylonian Collection dating from that period gives a base-60 number equivalent to 1.414222 as an approximation to $\sqrt{2}$, a result accurate to within 10^{-5} ($\sqrt{2} \approx 1.414214$).[4] We wanted to develop a hybrid method that quickly produces a small interval containing the solution and then switch to a method with faster convergence. We have created a method to solve functions whose exact roots are not easy to find using common techniques learned in algebra and calculus courses. We have compiled test functions, some of our own and some from other works on the same topic. We have also compared our method with that of several other methods consisting of Secant Method, False Position, a modified version of Modified False Position, Inverse Quadratic Interpolation, Bisection and a few other hybrid methods. Our method begins with the modified version of Modified False Position, which will be discussed in more detail later, then switches to Müller's method once a certain tolerance is reached. In certain instances, our method switches back to the modified version of Modified False Position. We found our method outperformed these methods in most cases and was competitive to the other hybrid methods, and in many cases, it outperformed them as well.

ACKNOWLEDGMENTS

I would like to give special thanks to Dr. Jozsi Jalics for coordinating my thesis. I had him for the first time when I took his Ordinary Differential Equations class in Spring 2010 and found his teaching style to be quite pleasant and decided to run my thesis under him since he was knowledgeable in the topics I wished to write my thesis on. I also did quite well in his class and wish I was able to take another class with him, however, due to the close proximity to my graduation and limited classes I was eligible to take, the opportunity was not available.

I would also like to thank Dr. Richard Burden for doing most of the work in coordinating my thesis by suggesting the project in which my thesis covered and meeting with me several times a week and even a few days over the summer along with much assistance with the code to run our hybrid method on. I first had him Fall 2009 for Numerical Analysis and also enjoyed his class and his teaching style and desired to study the subject further. Again due to limited classes he taught, I was unable to take anymore classes so I decided to do my thesis in Numerical Analysis. Dr. Burden is a co-author of a Numerical Analysis book that is currently on it's 9^{th} edition that was one of our sources in several sections of this thesis.

Further thanks goes to Dr. J. D. Faires for his assistance with inserting the images into my thesis and other assistance with formatting of these pages and how to use LaTeX. I had seen him many times, but never had the privilege to sit under his teaching through my years at Youngstown State University since the classes he taught didn't fit into my schedule. I first met him in Fall 2010 half way through the semester when I had LaTeX problems that my fellow graduate teaching assistants were unable to decipher. He is a very prestigious man and has been in the mathematics field for many years and is the other co-author of the Numerical Analysis book with Dr. Burden.

Next I would like to thank my fellow graduate teaching assistant, Moriah Wright, for her patience and assistance with several LaTeX questions and problems in which I sought out her help. She is a very kind and helpful person and has transitioned from an acquaintance to a friend over the past several years I have known her.

Also, I would like to thank another fellow graduate teaching assistant, Damon Haught, for his help with the proper coding to give the proper layout of the title, signature and copyright pages of my thesis. I have had several classes with him over the past several years and graduated with my undergraduate degree and now my master's degree during the same semester. He is a very friendly man and very pleasant to be around. I may never know how he is able to balance out college classes, teaching and life at home as a father of three, husband and part-time dance instructor. He is an inspiration to those who have had the privilege of knowing him.

Before too long, I need to thank my parents. Other than all the obvious reasons, I would like to thank them for their financial support throughout my undergraduate semesters. It was wonderful to graduate with my Bachelor's degree without any student loans. I would like to thank them and all other family members for all they have done throughout the years and continue to do.

Last, but definitely not least, I would like to thank my wonderful fiancée, Kirsten Anderson, who will be my wife on 7-9-11, (July 9^{th}, 2011), a date that is suitable for and can be appreciated by any mathematician. I appreciate her patience with me through the times I have worked on this paper and not been able to spend as much time as desired with her. She is an amazing woman that has helped me through some difficult issues in life in the year and eight or more months we've known each other, and has loved me unconditionally since we met. There is so much I could say about her here, but since this part is not the essence of this paper, I will say no more than she has kept me going and I can't wait to have her as my wife!

Contents

1 Methods of Solution

To solve $f(x) = 0$ where $x \in \Re$ and f is a continuous function, we first find an interval $[a, b]$ for which $f(a)$ and $f(b)$ are of opposite sign. This insures us that a solution p to $f(x) = 0$ lies in $[a, b]$ by the Intermediate Value Theorem. We will discuss two classes of methods for finding a solution to the equation by generating sequences of approximations $\{p_i\}$.

We begin by letting $a_1 = a$, $b_1 = b$ and calculate $p_1 \in [a, b]$. Then determine $[a_2, b_2]$ where $a_2 = p_1$, $b_2 = b_1$, or $a_2 = a_1$ and $b_2 = p_2$ so that $f(a_2)$ and $f(b_2)$ are opposite signs. This is a bracketing method. In bracketing methods, we generate a sequence of closed intervals $\{[a_i, b_i]\}$ which bracket the root p and contains the approximations p_i. Thus, $|p - p_i| \leq b_i - a_i$ for all i.

The second class of methods do not generate intervals $[a_i, b_i]$ at each iteration. In fact, in the non-bracketing methods we select $p_0 \in [a, b]$ and generate a sequence of approximations $\{p_i\}$. There is no guarantee that all approximations p_i are in the interval $[a, b]$, but if

$$\lim_{i \to \infty} p_i = p$$

then for i sufficiently large both p and p_i will be in $[a, b]$. This uncertainty is a disadvantage in comparison to bracketing methods. In both bracketing and non-bracketing methods, a given tolerance ϵ must be supplied so that once $|p - p_i| < \epsilon$, the computations cease at the i^{th} iteration.

2 Measuring Speed of Convergence

Suppose $\{p_i\}_{i=0}^{\infty}$ is a sequence that converges to p, with $p_i \neq p$ for all i. If positive constants K and α exist with

$$\lim_{i \to \infty} \frac{|p_{i+1} - p|}{|p_i - p|^{\alpha}} = K,$$

then $\{p_i\}_{i=0}^{\infty}$ *converges to p of order* α, *with asymptotic error constant* K.

An iterative technique of the form $p_i = g(p_{i-1})$ is said to be *of order* α if the sequence $\{p_i\}_{i=0}^{\infty}$ converges to the solution $p = g(p)$ of order α. [4]

In general, a sequence with a high order of convergence converges more rapidly than a sequence with a lower order. The asymptotic constant affects the speed of convergence but is not as important as the order. Two cases of order get special attention.

1. If $\alpha = 1$ (and $K < 1$), the sequence is *linearly convergent*.

2. If $\alpha = 2$, the sequence is *quadratically convergent*. [4]

Finding this α for a given method requires methodical calculations at each iteration. For this reason, some methods, especially hybrid ones, are unable to have a value of α calculated to determine it's speed of convergence.

3 Non-bracketing Methods

3.1 Newton's Method

To find successive approximations to the root p of a differentiable function $f(x) = 0$ with initial approximation p_0, Newton's method has the following formula:

$$p_i = p_{i-1} - \frac{f(p_{i-1})}{f'(p_{i-1})}, \quad \text{for } i \geq 1. \text{ [4]}$$

p_i is formed by calculating the x-intercept of the tangent line to $f(x)$ at $(p_{i-1}, f(p_{i-1}))$. Since we are interested in derivative free calculations, this method will not be discussed further. The reason for mentioning it is because the next method, the Secant method, uses a similar technique, but instead of having to calculate a derivative, it uses an approximation for the derivative that is evident in the formula in the section. For a graphical interpretation of Newton's method, see Figure 1 below.

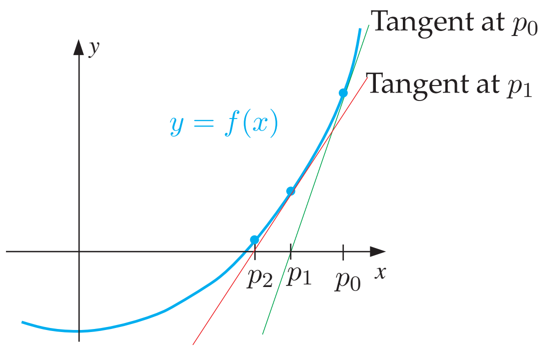

Figure 1: This figure is a model that demonstrates how Newton's method generates successive iterates.

The α from above for Newton's method is 2 (unless $f'(p) = 0$).

3.2 Secant Method

Suppose p_0 and p_1 are initial approximations to the root of $f(x) = 0$ such that the root $p \in (p_0, p_1)$. The approximation p_2 is the x-intercept of the line joining $(p_0, f(p_0))$ and $(p_1, f(p_1))$. To find p_i for all $i \geq 2$,

$$p_i = p_{i-1} - \frac{f(p_{i-1})(p_{i-1} - p_{i-2})}{f(p_{i-1}) - f(p_{i-2})}.$$

The α from above for the Secant method is 1.618, which we show in Theorem 2 of Section 5. For a graphical interpretation of the Secant method, see Figure 2 below.

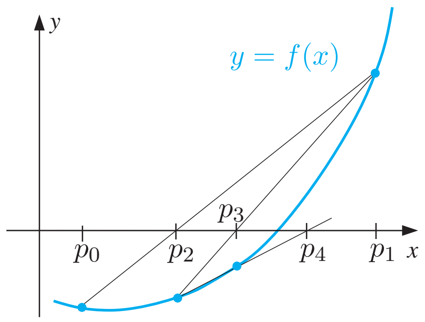

Figure 2: This figure is a model that demonstrates how the Secant method generates successive iterates.

3.3 Müller

Müller's method uses three initial approximations p_0, p_1, and p_2 to the root p and determines the next approximation p_3 by considering the intersection of the x-axis with the parabola through $(p_0, f(p_0))$,$(p_1, f(p_1))$, and $(p_2, f(p_2))$. To begin, one must consider the quadratic polynomial $P(x) = a(x - p_2)^2 + b(x - p_2) + c$. It can be shown that the constants a,b and c should have the following values:

$$c = f(p_2),$$
$$b = \frac{(p_0 - p_2)^2[f(p_1) - f(p_2)] - (p_1 - p_2)^2[f(p_0) - f(p_2)]}{(p_0 - p_2)(p_1 - p_2)(p_0 - p_1)},$$

and

$$a = \frac{(p_1 - p_2)[f(p_0) - f(p_2)] - (p_0 - p_2)^2[f(p_1) - f(p_2)]}{(p_0 - p_2)(p_1 - p_2)(p_0 - p_1)}.$$

To find p_i for all $i \geq 3$,

$$p_i = p_{i-1} - \frac{2c}{b + sgn(b)\sqrt{b^2 - 4ac}}.$$

Once p_i is found, the procedure is re-initialized using p_{i-2}, p_{i-1}, and p_i in place of p_{i-3}, p_{i-2}, and p_{i-1} to determine the next approximation p_{i+1}. Note: It can happen that $b^2 - 4ac < 0$ giving complex roots. This created a problem in our method, and we shall discuss why later. See Figure 3 below for a graphical interpretation.

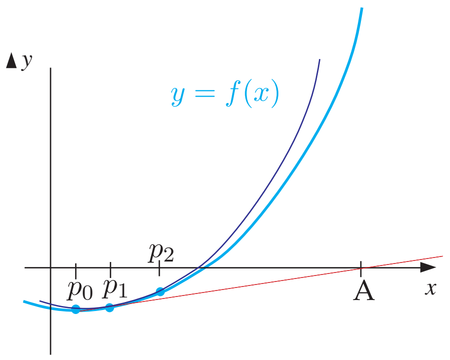

Figure 3: This figure is a model that demonstrates how Müller's method differs from the Secant method generates successive iterates. Also, A denotes p_2 formed by the Secant method.

The order of convergence, α, for Müller's method is 1.84, which we show in Theorem 2.

3.4 Inverse Quadratic Interpolation

First suppose $f \in C^1[a,b]$, $f'(x) \neq 0$ on $[a,b]$ and f has one zero p in $[a,b]$. Let x_0, x_1, and x_2, be 3 distinct numbers in $[a,b]$ with $f(x_i) = y_i$, for each $i = 0, 1, 2$. To approximate p, construct the interpolating polynomial of degree 2 on the nodes y_0, y_1, and y_2 for f^{-1}, the inverse function of f and evaluate at 0 to get the next iterate. Since $y_i = f(x_i)$ and

$0 = f(p)$, it follows that $f^{-1}(y_i) = x_i$ and $p = f^{-1}(0)$. To find x_i for all $i \geq 3$,

$$x_i = \frac{f_{i-2}f_{i-1}x_{i-3}}{[f_{i-3} - f_{i-2}][f_{i-3} - f_{i-1}]} + \frac{f_{i-3}f_{i-1}x_{i-2}}{[f_{i-2} - f_{i-3}][f_{i-2} - f_{i-1}]} + \frac{f_{i-3}f_{i-2}x_{i-3}}{[f_{i-1} - f_{i-3}][f_{i-1} - f_{i-2}]},$$

where $f_j = f(x_j)$ for all j. This differs from Müller's method since Müller's method always looks for closest x-intercept to p_2.

4 Bracketing Methods

4.1 Bisection Method

Suppose f is a continuous function defined on the interval $[a, b]$, with $f(a)$ and $f(b)$ of opposite sign. By the Intermediate Value Theorem, there exists a number $p \in (a, b)$ with $f(p) = 0$. Also suppose there are an odd number of zeros of $f \in (a, b)$. The method now requires repeatedly halving subintervals of $[a, b]$ and, at each step, locating the half containing p. To begin, set $a_1 = a$ and $b_1 = b$, and let

$$p_1 = \frac{a_1 + b_1}{2}, \text{ the midpoint of } [a, b].$$

If $f(p_1) = 0$, then $p = p_1$, and we are done. If $f(p_1) \neq 0$, then $f(p_1)$ has the same sign as either $f(a_1)$ or $f(b_1)$. When $f(p_1)$ and $f(a_1)$ have the same sign, $p \in (p_1, b_1)$, and we set $a_2 = p_1$ and $b_2 = b_1$. When $f(p_1)$ and $f(a_1)$ have opposite signs, $p \in (a_1, p_1)$, and we set $a_2 = a_1$ and $b_2 = p_1$. We then reapply the process to the interval $[a_2, b_2]$, and so on. For a graphical interpretation, see Figure 4.

4.2 False Position (Regula Falsi)

This method generates approximations in a similar manner as the Secant method, but also includes a test to ensure that the root is always bracketed between successive iterations. Start by choosing p_0 and p_1 with $f(p_0)f(p_1) < 0$. The approximation p_2 is chosen in the same manner as the Secant method, as the x-intercept of the line joining$(p_0, f(p_0))$ and $(p_1, f(p_1))$. To decide which secant line to use to compute p_3, we check $f(p_1)f(p_2)$. If this value is negative, then p_1 and p_2 bracket a root, and we choose p_3 as the x-intercept of the line joining $(p_1, f(p_1))$ and $(p_2, f(p_2))$. If not, we choose p_3 as the x-intercept of the line joining $(p_0, f(p_0))$ and $(p_2, f(p_2))$, and then interchange the indices p_0 and p_1. Similarly, once p_3 is found, the sign of $f(p_3)f(p_2)$ determines whether we use p_2 and p_3 or p_3 and p_1 to compute p_4. In the last case a relabeling of p_2 and p_1 is per-

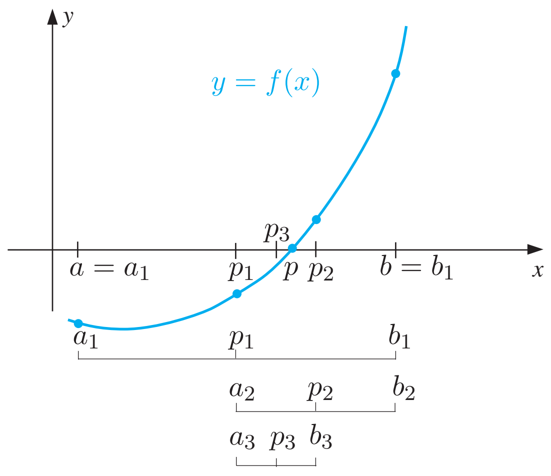

Figure 4: This figure is a model that demonstrates how the Bisection method generates successive iterates.

formed. The relabeling ensures that the root is bracketed between successive iterations. For a graphical interpretation, see Figure 5 below.

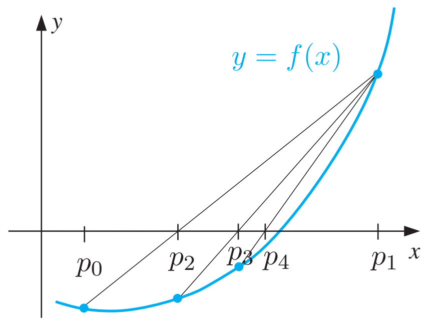

Figure 5: This figure is a model that demonstrates how False Position generates successive iterates.

The order of convergence α for False Position is 1 if the function is concave or convex or else it can behave like the Secant method.

4.3 Modified False Position

Modified False Position is similar to False Position, except here we multiply $f(p_{i-1})$ or $f(p_i)$ by λ when calculating p_{i+1} to avoid the retention of the left or right end point over successive iterations. For a graphical interpretation, see Figure 6 below.

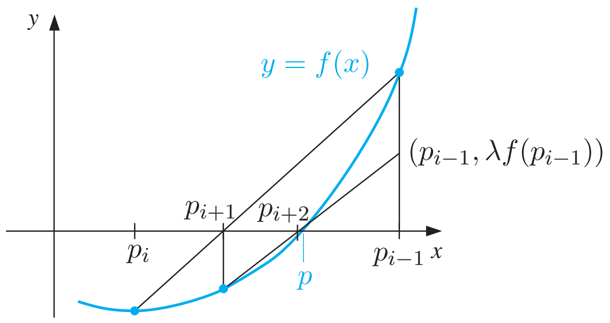

Figure 6: This figure shows how Modified False Position generates successive iterates, scaling down $f(p_{i-1})$ by a factor of λ to avoid end point retention as mentioned in the False Position section. Note: The same occurrence happens if the graph is reversed, and this time $f(p_i)$ is scaled down by a factor of λ.

4.4 Illinois Method

This method follows the False Position method, but uses $\lambda = \frac{1}{2}$. The estimates chosen for the next iteration are selected according to the following rules:

1. if $f(p_{i+1})f(p_i) < 0$, then $(p_i, f(p_i))$ and $(p_{i+1}, f(p_{i+1}))$ are retained.
2. if $f(p_{i+1})f(p_i) > 0$, then $\left(p_{i-1}, \frac{f(p_{i-1})}{2}\right)$ and $(p_{i+1}, f(p_{i+1}))$ are retained.

The function values used at each iteration will have opposite signs and the introduction of the value $\frac{f(p_{i-1})}{2}$ for $f(p_{i-1})$ is a modification designed to speed convergence by eventually stopping the retention of one of the end points.

5 Theorems

Theorem 1: If $f(x)$ is a function such that:

1. $f'(x) \neq 0$ on an interval $[a, b]$

2. $f^{''}$ exists on $[a, b]$

3. $f(a)f(b) < 0$

4. the Secant method converges to a root $p \in (a, b)$,

then the order of convergence is $\alpha = \frac{1}{2}(1 + \sqrt{5})$

Proof. The Secant Method is a linear interpolation based on $(p_i, f(p_i))$, $(p_{i-1}, f(p_{i-1}))$ so

$$f(x) = p_1(x) + \frac{f^{''}(\xi_i)}{2}(x - p_{i-1})(x - p_i),$$

where $p_1(x)$ is the linear interpolating polynomial. Using Newton's divided difference form for $p_1(x)$ gives

$$f(x) = f(p_i) + f[p_i, p_{i-1}](x - p_i) + \frac{1}{2}f^{''}(\xi_i)(x - p_{i-1})(x - p_i).$$

Set $f(x) = 0$ and solve for x ignoring error term to get $x = p_{i+1}$

$$0 = f(p_i) + f[p_i, p_{i-1}](p_{i+1} - p_i). \tag{1}$$

We note that

$$p_{i+1} = p_i - \frac{f(p_i)}{f[p_i, p_{i-1}]} = p_i - f(p_i)\left(\frac{p_i - p_{i-1}}{f(p_i) - f(p_{i-1})}\right).$$

The actual solution p to $f(p) = 0$ satisfies

$$0 = f(p)$$

$$0 = f(p_i) + f[p_i, p_{i-1}](p - p_i) + \frac{1}{2}f^{''}(\xi_i)(p - p_{i-1})(p - p_i) \tag{2}$$

Subtracting equation (2) from equation (1), we get

$$0 = f[p_i, p_{i-1}](p - p_{i+1}) + \frac{1}{2}f^{''}(\xi)(p - p_{i-1})(p - p_i)$$

Let $e_i = p - p_i$ for every i. Then

$$f[p_i, p_{i-1}]e_{i+1} = -\frac{1}{2}f(\xi_i)e_i e_{i-1}$$

so

$$e_{i+1} = \frac{f''(\xi_i)}{f[p_i, p_{i-1}]} e_i e_{i-1.}$$

By Mean Value Theorem,

$$f[p_i, p_{i-1}] = \frac{f(p_i) - f(p_{i-1})}{p_i - p_{i-1}} = f'(\xi_i') \quad \text{for some } \xi_i' \in (p_i, p_{i-1})$$

so

$$e_{i+1} = -\frac{f''(\xi_i)}{2f'(\xi_i)} e_i e_{i-1}$$

Now suppose the Secant method converges and the order of convergence is α. Then for large i, $\frac{|e_{i+1}|}{|e_i|^\alpha} \approx K$.

Thus,

$$|e_{i+1}| \approx K|e_i|^\alpha \text{ and } |e_i| \approx K|e_{i-1}|^\alpha \Rightarrow |e_{i-1}| \approx K^{-\frac{1}{\alpha}}|e_i|^{\frac{1}{\alpha}}.$$

So

$$K|e_i|^\alpha \approx c|e_i|K^{-\frac{1}{\alpha}}|e_i|^{\frac{1}{\alpha}}$$

and

$$|e_i|^{\alpha - 1 - \frac{1}{\alpha}} \approx cK^{-\frac{1}{\alpha}-1}.$$

The left hand side is dependent of i and the right hand side is independent of i

Thus we have

$$\alpha - 1 - \frac{1}{\alpha} = 0$$

so

$$\alpha^2 - \alpha - 1 = 0$$

and

$$\alpha = \frac{1 \pm \sqrt{1+4}}{2} = \frac{1}{2}(1 + \sqrt{5}) = 1.6180399....$$

□

Theorem 2: If $f(x) \in C^3[a, b]$ and $f(a)f(b) < 0$ and Müller's method converges to a simple root $p \in [a, b]$, then the order of convergence is $\alpha \approx 1.84$.

Proof. We assume $f(x) \in C^3[a, b]$ with $f(a)f(b) < 0$. $f'''(x)$ is continuous on $[a, b]$ so the second degree interpolating polynomial for any x_0, x_1 and $x_2 \in [a,b]$ satisfies

$$f(x) = P_2(x) + \frac{f'''(\xi)}{6}(x-x_0)(x-x_1)(x-x_2)$$

for any x where ξ depends on x. Assume Müller's method converges to a simple zero $p \in (a,b)$. Let $\{p_i\}$ denote the Müller's sequence. For i sufficiently large p_{i-1}, p_i, p_{i+1} and p_{i+2} are all contained in $[a,b]$. We seek the rate of convergence α of $\{p_i\}$, that is,

$$\lim_{i\to\infty} \frac{|p-p_{i+1}|}{|p-p_i|^\alpha} = \lambda.$$

We shall show that $\alpha \approx 1.84$. We assume that for large i $|p_i| \leq max(|a|,|b|)$. Consider the divided difference form of the interpolating polynomial of degree 2 on p_{i+2}, p_{i+1} and p_i:

$$P_2(x) = f(p_{i+2}) + f[p_{i+2}, p_{i+1}](x-p_{i+2}) + f[p_{i+2}, p_{i+1}, p_i](x-p_{i+2})(x-p_{i+1}).$$

Now p_{i+3} is a zero of $P_2(x)$. Letting $a_1 = f[p_{i+2}, p_{i+1}]$ and $a_2 = f[p_{i+2}, p_{i+1}, p_i]$ we have

$$P_2(p_{i+3}) = 0 = f(p_{i+2}) + a_1(p_{i+3}-p_{i+2}) + a_2(p_{i+3}-p_{i+2})(p_{i+3}-p_{i+1}). \qquad (3)$$

Let $f(p) = 0$. So there exists a number ξ in the smallest interval containing p_i, p_{i+1}, and p_{i+2}, such that

$$f(p) - P_2(p) = \frac{f'''(\xi)}{6}(p-p_{i+2})(p-p_{i+1})(p-p_i)$$

$$0 = P_2(p) + \frac{f'''(\xi)}{6}(p-p_{i+2})(p-p_{i+1})(p-p_i).$$

Letting

$$P_2(p) = f(p_{i+2}) + a_1(p-p_{i+2}) + a_2(p-p_{i+2})(p-p_{i+1})$$

and $e_i = p - p_i$ for all i we get

$$0 = f(p_{i+2}) + a_1(p-p_{i+2}) + a_2(p-p_{i+2})(p-p_{i+1}) + \frac{f'''(\xi)}{6}e_{i+2}e_{i+1}e_i. \qquad (4)$$

Subtracting equation (3) from equation (4) gives

$$0 = a_1(p-p_{i+2}) + a_2(p-p_{i+2})(p-p_{i+1}) - a_1(p_{i+3}-p_{i+2})$$

$$-a_2(p_{i+3}-p_{i+2})(p_{i+3}-p_{i+1})+\frac{f'''(\xi)}{6}e_{i+2}e_{i+1}e_i,$$

and

$$0=a_1[p-p_{i+2}-p_{i+3}+p_{i+2}]+a_2[p^2-pp_{i+2}-pp_{i+1}+p_{i+1}p_{i+2}-p_{i+3}^2,$$

$$+p_{i+2}p_{i+3}+p_{i+3}p_{i+1}-p_{i+1}p_{i+2}]+\frac{f'''(\xi)}{6}e_{i+2}e_{i+1}e_i.$$

We also have,

$$0=a_1[p-p_{i+3}]+a_2[p^2-p_{i+3}^2+(p-p_{i+3})[-p_{i+2}-p_{i+1}]]+\frac{f'''(\xi)}{6}e_{i+2}e_{i+1}e_i,$$

and

$$0=a_1e_{i+3}+a_2e_{i+3}[p+p_{i+3}-p_{i+2}-p_{i+1}]+\frac{f'''(\xi)}{6}e_{i+2}e_{i+1}e_i,$$

finally,

$$0=e_{i+3}[a_1+a_2(p+p_{i+3}-p_{i+2}-p_{i+1})]+\frac{f'''(\xi)}{6}e_{i+2}e_{i+1}e_i.$$

So we have

$$K_1e_{i+3}=K_2e_{i+2}e_{i+1}e_i$$

where

$$K_1=[a_1+a_2(p+p_{i+3}-p_{i+2}-p_{i+1})],K_2=-\frac{f'''(\xi)}{6}.$$

Suppose that i is sufficiently large that we have both

$$K_3\approx\frac{|e_{i+2}|}{|e_{i+1}|^\alpha}\quad\text{and}\quad K_3\approx\frac{|e_{i+3}|}{|e_{i+2}|^\alpha}.$$

Thus

$$|e_{i+3}|=K_3|e_{i+2}|^\alpha$$

and

$$|e_{i+2}|=K_3|e_{i+1}|^\alpha\Rightarrow|e_{i+1}|^\alpha=\left|\frac{e_{i+2}}{K_3}\right|$$

which implies

$$|e_{i+1}|=\frac{|e_{i+2}|}{K_3}^{\frac{1}{\alpha}}.$$

Then

$$\begin{aligned}
|e_{i+1}| = K_3|e_i|^\alpha \Rightarrow |e_i| &= \frac{|e_{i+1}|}{K_3}^{\frac{1}{\alpha}} \\
&= \left[\frac{\frac{|e_{i+2}|}{K_3}^{\frac{1}{\alpha}}}{K_3}\right]^{\frac{1}{\alpha}} \\
&= \frac{1}{K_3^{\frac{1}{\alpha}}}\frac{|e_{i+2}|^{\frac{1}{\alpha^2}}}{K_3^{\frac{1}{\alpha}}} \\
&= \frac{|e_{i+2}|^{\frac{1}{\alpha^2}}}{K_3^{\frac{2}{\alpha}}}
\end{aligned}$$

Now

$$\begin{aligned}
K_1|e_{i+3}| &= K_2|e_{i+2}|\left[\frac{|e_{i+2}|}{K_3}\right]^{\frac{1}{\alpha}}\left[\frac{|e_{i+2}|^{\frac{1}{\alpha^2}}}{K_3^{\frac{2}{\alpha}}}\right] \\
&= \frac{K_2}{K_3^{\frac{3}{\alpha}}}\,|e_{i+2}|^{1+\frac{1}{\alpha}+\frac{1}{\alpha^2}}\,.
\end{aligned}$$

So

$$\frac{|e_{i+3}|}{\left|e_{i+2}^{1+\frac{1}{\alpha}+\frac{1}{\alpha^2}}\right|} = K_4K_3$$

and

$$|e_{i+2}|^{\alpha-1-\frac{1}{\alpha}-\frac{1}{\alpha^2}} = K_4K_3.$$

The left hand side is dependent of i and the right hand side is independent of i.

So

$$\alpha - 1 - \frac{1}{\alpha} - \frac{1}{\alpha^2} = 0$$

and

$$\alpha^3 - \alpha^2 - \alpha - 1 = 0.$$

Using Newton's method, we find that the real root of

$$\alpha - 1 - \frac{1}{\alpha} - \frac{1}{\alpha^2} = 0$$

is

$$\alpha = 1.839286755 \approx 1.84 \text{ (as suggested in Atkinson).}$$

$\square$

6 Hybrid Methods

6.1 Brent's Method

Brent's method [6] is a hybrid method that seeks to combine the Bisection method, the Secant method, which he calls Linear Interpolation, and sometimes Inverse Quadratic Interpolation. Brent has shown that the method converges at least as fast as the Bisection method. It also exhibits super-linear convergence to a single zero of a continuously differentiable function since in that case only linear interpolation or better is used.

We note that linear interpolation gives

$$p_1(x) = \frac{x - p_1}{p_0 - p_1} f(p_0) + \frac{x - p_1}{p_1 - p_0} f(p_1).$$

Solving $P_1(x) = 0$ for $x = p_2$ gives the Secant method or linear interpolation

$$p_2 = p_1 - \frac{f(p_1)(p_1 - p_0)}{f(p_1) - f(p_0)}$$

Brent uses three values of x. They are called a, b and c where initially $a = c$ and $f(b)f(c) \leq 0$ with $|f(b)| \leq |f(c)|$ and the initial interval is $[b, c]$ or $[c, b]$. The values a, b and c are changed at each step so that b is the better approximation and the former b replaces a. If $f(b) = 0$, then we are done. If $f(b) \neq 0$, let $m = \frac{1}{2}(c - b)$. Thus,

$$b + m = b + \frac{1}{2}c - \frac{1}{2}b = \frac{b + c}{2},$$

which is a bisection method. Let $i = b + \frac{f(b)(b-a)}{f(b)-f(a)}$ where calculations are done to minimize rounding error for the Secant method. However, if a, b and c are distinct, we can do inverse quadratic interpolation. In summary, we use Bisection if $|f(b)| \geq |f(a)|$, otherwise $|f(b)| < |f(a)| \leq |f(c)|$ so let

$$r_1 = \frac{f(a)}{f(c)}, r_2 = \frac{f(b)}{f(c)} \text{ and } r_3 = \frac{f(b)}{f(a)}$$
$$p = r_1[(c-b)r_1(r_1 - r_2) - (b-a)(r_2 - 1)]$$
$$q = -(r_1 - 1)(r_2 - 1)(r_3 - 1).$$

Now

$$i = b + \frac{p}{q}.$$

6.2 Our Method

We created a method that begins with a modified version of the Modified False Position method in which we use $\lambda = 1.5$ instead of 2, as in the Illinois method, which we found to be more efficient through several test cases. Once $|p_{i+1} - p_i| < \epsilon$, our method switches to that of Müller's. However, in Müller's method the three approximations, $f(p_{i-1}), f(p_i), f(p_{i+1})$, may be of the same sign. This may cause the method to produce complex values. In this case, we jump back into the modified version of Modified False Position.

We are unable to calculate an α for our method since we change methods within our calculations, but when p_n is far from p, the method is strictly the modified version of Modified False Position and has convergence order of $\alpha = 1$ or $\alpha = 1.6$. When p_n is close to p, the method is strictly Müller's method and has convergence order $\alpha = 1.84$.

7 Error Analysis

We will begin our discussion of error analysis with the Secant method, which has the formula to generate approximations p_i,

$$p_i = p_{i-1} - \frac{f(p_{i-1})(p_{i-1} - p_{i-2})}{f(p_{i-1}) - f(p_{i-2})}.$$

We assume $f''(x) \neq 0$ is continuous and $f'(x) \neq 0$ for an interval $[a,b]$ containing p, the solution to $f(x) = 0$. We have shown that the Secant method gives for large i the relation

$$|e_{i+1}| \approx K|e_i|^\alpha$$

where

$$\alpha \approx \frac{1+\sqrt{5}}{2}$$

We discuss False Position versus the Secant method. False Position always converges if $f(x)$ is continuous and $f(a)f(b) < 0$, where $p_0 = a$ and $p_1 = b$. The worse case is when $f(x)$ is concave or convex on $[a, b]$.

Suppose we have generated p_{j-1} and p_j where $f(p_{j-1})f(p_j) < 0$. We generate p_{j+1} using the Secant method formula. If $f(p_j)$ and $f(p_{j+1})$ are of opposite sign we use p_j and $p_{j+1}$to generate p_{j+2} as in a Secant method step. However, if $f(p_{j-1})$ and $f(p_{j+1})$ have opposite signs we discard p_j and use p_{j-1} and p_{j+1} to generate p_{j+2}. With a convex or concave function the process could continue with p_{j-1} being retained and p_{j+k} replacing p_{j+k-1} for $k = 1, 2, 3, ...$ For large $i + 1 = j + k + 1$ we could have

$$|e_{i+1}| \approx c|e_i||e_{j-1}|$$

so

$$\frac{|e_{i+1}|}{|e_i|} \approx c|e_{j-1}|$$

and

$$\lim_{i\to\infty} \frac{|e_{i+1}|}{|e_i|} = c|e_{j-1}|.$$

This makes False Position a linearly convergent method.

We again use the interpolation polynomial to obtain

$$f(x) = f(p_i) + f[p_i, p_{i-1}](x - p_i) + (x - p_{i-1})(x - p_i)\frac{1}{2}f^{''}(\xi_i)$$

and for $x = p$

$$f(p) = 0 = f(p_i) + f[p_i, p_{i-1}](p - p_i) + \frac{1}{2}f^{''}(\xi_i)(p - p_{i-1})(p - p_i), \tag{5}$$

but the Secant method gives

$$0 = f(p_i) + (p_{i+1} - p_i)f[p_i, p_{i-1}] \tag{6}$$

Subtracting equation (6) from equation (5) gives

$$0 = f[p_i, p_{i-1}](p - p_i - p_{i+1} + p_i) + (p - p_{i-1})(p - p_i)\frac{1}{2}f''(\xi_i).$$

Simplifying,

$$0 = f[p_i, p_{i-1}]e_{i+1} + e_i e_{i-1}\frac{1}{2}f''(\xi_i)$$

so

$$e_{i+1} = -e_i e_{i-1}\frac{f''(\xi_i)}{f[p_{i-1}, p_i]} = -e_i e_{i-1}\frac{f''(\xi_i)}{2f'(\xi_i')}.$$

Assume f'', f' have constant sign on (a, b), then $\frac{e_{i+1}}{e_i e_{i-1}}$ has constant sign.

If $f(p_0)f(p_1) < 0$, then $e_0 = p - p_0 > 0$ and $e_1 = p - p_1 < 0$.

Case 1: keep p_1, p_2 so $e_2 > 0$

$\frac{e_2}{e_1 e_0} < 0$.

For $\frac{e_3}{e_2 e_1} < 0$ we need $e_3 > 0$.

For $\frac{e_4}{e_3 e_2} < 0$ we need $e_4 < 0$.

This leads to a sign pattern of the $e_i's$ to be $+ - + + -$

Case 2: if $e_0 < 0$, $e_1 > 0$, the sign pattern is $- + - - + - - +$

False Position: p is between p_{i-1} and p_i, so the sign pattern is $-+-+$... or $+-+-$...

So in this case for the Secant method, every third step is not False Position.

In False Position: Suppose that for certain i $f(p_{i+1})f(p_i) > 0$ and $f(p_{i+1})f(p_{i-1}) < 0$ so we keep p_{i-1} as the new p_i and throw out p_i. Now p_{i+2} is chosen by the secant line through $(p_{i-1}, f(p_{i-1}))$ and $(p_{i+1}, f(p_{i+1}))$.

Illinois-Type modification: Take p_{i+2} as the root of the secant line through $(p_{i-1}, \frac{1}{2}f(p_{i-1}))$ and $(p_{i+1}, f(p_{i+1}))$

It can still happen that $f(p_{i+2})f(p_{i+1}) > 0$ and we could then use in place of $\frac{1}{2}$ the values $\frac{1}{4}$,$\frac{1}{8}$,$\frac{1}{16}$,... until there is a sign change. [7]

We seek improvements in False Position other that the Illinois method. In Figure 2 of Section 4.3, $f(x)$ is concave and this presents difficulties in the False Position method. The intervals bracketing p are $[p_0, p_1], [p_0, p_2], [p_0, p_3], \ldots$. The sequence $\{p_i\}$ will converge linearly to p since p_0 is never discarded. Thus, the length of the bracketing intervals do not converge to zero as in the Bisection method. To remedy this problem we consider the situation where $f(p_{i-1})f(p_i) < 0$ and $f(p_{i+1})f(p_i) > 0$ so we would retain p_{i-1} and discard p_i. Instead of using $(p_{i-1}, f(p_{i-1}))$ and $(p_{i+1}, f(p_{i+1}))$ to generate p_{i+2}, we consider using $(p_{i-1}, \lambda f(p_{i-1}))$ and $(p_{i+1}, f(p_{i+1}))$. Optimally, we choose λ so that $p_{i+2} = p$.

By similar triangles

$$\frac{\lambda f(p_{i-1})}{-f(p_{i+1})} = \frac{p - p_{i-1}}{p_{i+1} - p}$$

Solving for λ and absorbing minus sign into the numerator

$$\lambda = \frac{f(p_{i+1})[p_{i-1} - p]}{f(p_{i-1})[p_{i+1} - p]} = \left[\frac{f(p_{i+1})}{p_{i+1} - p}\right]\left[\frac{p_{i-1} - p}{f(p_{i-1})}\right]$$

Since $f(p) = 0$, we have

$$\lambda = \left[\frac{f(p_{i+1}) - f(p)}{p_{i+1} - p}\right]\left[\frac{p_{i-1} - p}{f(p_{i-1}) - f(p)}\right] = \frac{f[p, p_{i+1}]}{f[p, p_{i-1}]} = \frac{f[p_{i+1}, p]}{f[p_{i-1}, p]}$$

using the divided difference notation.

Since p is unknown, we need to approximate $\lambda = \frac{f[p_{i+1},p]}{f[p_{i-1},p]}$

The Illinois method makes a simple choice $\lambda = \frac{1}{2}$.

If we let $e_i = p_i - p$, it can be shown that for large i

$$e_{i+1} \approx \frac{f''(p)}{2f'(p)} e_i e_{i-1}$$

for the Secant method, which uses p_{i-1} and p_i to get p_{i+1}. If $f(p_{i+1})f(p_i) < 0$, we continue with the Secant method resulting in

$$e_{i+2} \approx \frac{f''(p)}{2f'(p)} e_{i+1} e_i.$$

However, if $f(p_{i+1})f(p_{i-1}) < 0$ leaving $f(p_{i+1})f(p_i) > 0$, we the Illinois modified step with $\lambda = \frac{1}{2}$ which gives

$$e_{i+2} \approx -e_{i+1}.$$

So asymptotically this gives a sign change. If $e_{i+1} > 0$, that is $p_{i+1} > p$, we must have had $p_{i-1} < p$. Then $e_{i+2} < 0$ implies $p_{i+2} < p$ also. Consequently, p_{i+2} replaces p_{i-1} and we continue. Letting U stand for an unmodified step (Secant method) and M stand for a modified step, we have the sequence of steps

UUM, UUM, UUM, ... or

UM, UUM, UUM as basic patterns.

However, then there are other choices for λ. First we can approximate $f[p_{i-1}, p]$ with the following choices

A1: $f[p_{i-1}, p_i]$

A2: $f[p_{i-1}, p_{i+1}]$

A3: $f[p_{i+1}, p_{i-1}] + f[p_i, p_{i-1}] - f[p_{i+1}, p]$

and similarly approximate $f[p_{i+1}, p]$ with

B1: $f[p_{i+1}, p_i]$
B2: $f[p_{i+1}, p_{i-1}]$
B3: $f[p_{i+1}, p_i] + f[p_{i+1}, p_{i-1}] - f[p_i, p_{i-1}]$.

The reasons for A1, A2, B1 and B2 are a matter of convenient use of what has been computed. The choices A3 and B3 correspond to the derivative $P_2(x)$ of the interpolating polynomial $P_2(x)$ based on p_{i-1}, p_i and p_{i+1} evaluated at $x = p_{i+1}$ (A3) or $x = p_{i-1}$ (B3) respectively.

The method of Anderson and Björck has the choice of A1 and B1 together.

In the paper by Ford, the methods used are
Method 1: A3,B3
Method 2: A2,B1
Method 3: A2,B3
Method 4: A1,B3
Method 5: A3,B1

Method 5 was discarded since it was never better than Anderson-Björck, and often worse.

The Pegasus method uses

$$\lambda = \frac{f(p_i)}{f(p_i) + f(p_{i-1})}.$$

In all cases above, we have a modified step described by

$$p_{i+2} = \frac{f(p_{i+1})p_{i-1} - \lambda f(p_{i-1})p_{i+1}}{f(p_{i+1}) - \lambda f(p_{i-1})}.$$

8 Examples

In this section, we have listed the various test functions we have found and have compared our method against those mentioned in previous sections. In the following section, we have included charts that list the number of iterations each method took to arrive at the approximation of the root p and will analyze the results. The results of each set of test functions are listed in the chart with the name(s) of those who used them in their work.

The example problems used by Anderson and Björck are:

1. $f(x) = \sin(x) - 0.5$ with $x_0 = 0, x_1 = 1.5$
2. $f(x) = 2xe^{-n} + 1 - 2e^{-nx}$ with $x_0 = 0, x_1 = 1$

3. $f(x) = (1 + (1 - n)^2)x - (1 - nx)^2$ with $x_0 = 0, x_1 = 1$

4. $f(x) = x^2 - (1 - x)^n$ with $x_0 = 0, x_1 = 1$

5. $f(x) = 1 + (1 + (1 - n)^4)x - (1 - nx)^4$ with $x_0 = 0, x_1 = 1$

6. $f(x) = e^{-x}(x - 1) + x_n$ with $x_0 = 0, x_1 = 1$

7. $f(x) = \dfrac{nx - 1}{(n - 1)x}$ with $x_0 = 0.01, x_1 = 1$

Calculations were preformed to a precision of $6 * 10^{-8}$. [3]

(Note: We will omit example 5 because the value of $x_0 = 0$ is a zero of the function already.)

The example problems used by Dowell and Jarratt are the same as those used by Anderson and Björck, except now the calculations were preformed to a precision of $5 * 10^{-20}$. [2]

The example problems used by Ford are:

1. $f(x) = 4\cos(x) - e^x$ with root $p = 0.904788$
3. $f(x) = 2xe^{-20} + 1 - 2e^{-20x}$ with root $p = 0.034657$
4. $f(x) = e^{x^{-1}-25} - 1$ with root $p = 0.04$
6. $f(x) = 10^{10}x^{\frac{1}{x}} - 1$ with root $p = 0.1$
7. $f(x) = x^{20} - 1$ with root $p = 1.0$
8. $f(x) = \dfrac{e^{\frac{21000}{x}}}{1.11 * 10^{11}x^2} - 1$ with root $p = 551.774$
9. $f(x) = \frac{1}{x} + \ln(x) - 100$ with root $p = 0.009556$
10. $f(x) = e^{e^x} - e^{e^1}$ with root $p = 1.0$
11. $f(x) = \sin\left(\frac{0.01}{x}\right) - 0.01$ with root $p = 0.999983$

Calculations were preformed to a precision of 10^{-14}. [5]

(Note: For chart simplicity, we denoted "Anderson-Björck"as "A-B", "Method 1", "Method 2", "Method 3"and "Method 4"as "M1", "M2", "M3"and "M4"respectively. Also note that we omitted Ford's examples 2 and 5 since we were unable to get them to run in our Maple program.)

The example problems used in the Various Common Methods chart below are:

1. $f(x) = x^3 - 2x^2 - 5$ with root $p = 2.69064744802861$

2. $f(x) = x^3 + 2x^2 - 1$ with root $p = -1.61803398874989$

3. $f(x) = 2x\cos(2x) - (x - 2)^2$ with root $p = 3.72211277310179$

4. $f(x) = 3x\tan(2x) - (x - 2)^2$ with root $p = 0.495135510634739$

Calculations were preformed to a precision of 10^{-10}. [4]

(Note: For chart simplicity, we denoted “False Position”as “FP”, “Modified False Position”as “MFP”and “Inverse Quadratic Interpolation”as “Inv. Quad. Int.”.)

9 Results

The following are tables comparing the number of iterations required by the various methods mentioned in previous sections with that of our method. We will highlight some of the tests where our method outperformed the others, “competitive”(within 2 iterations) and those where our method did much worse.

Anderson-Björck v. Our Method

Example	n value	Illinois Method	Brent’s Method	Algorithm “A”	Pegasus Method	Our Method
1		6	6	5	6	6
2	1	6	5	4	5	5
	5	9	8	8	7	7
	15	9	9	10	8	9
	20	9	9	10	8	9
3	2	7	7	6	6	5
	5	7	6	6	6	5
	15	5	5	5	6	5
	20	4	5	4	5	5
4	2	1	1	1	1	2
	5	6	6	6	7	7
	15	10	8	9	9	11
	20	10	10	9	10	11

Anderson-Björck v. Our Method continued

Example	n value	Illinois Method	Brent's Method	Algorithm "A"	Pegasus Method	Our Method
6	1	7	6	5	6	5
	5	7	6	6	11	8
	10	11	6	7	11	12
	15	>16	9	7	15	15
	20	>16	10	8	>16	>16
7	2	11	3	4	12	13
	5	13	10	5	11	14
	15	12	9	4	11	11
	20	13	11	4	8	11

Overall, our method fared quite well in all seven examples. Brent's method and Algorithm "A"seem to have performed the best, but in examples 2 and 3, our method matched or outperformed both methods for most n values. In example 7, our method did worse in comparison to all but the Illinois method.

Dowell-Jarratt v. Our Method

Example	n value	Bisection Method	False Position	False Position and Bisection	Illinois Method	Our Method
1					6	7
2	1	64	23	19	9	6
	5	64	40	21	10	9
	15	67	41	23	11	10
	20	67	42	23	11	10
3	2	64	25	21	9	5
	5	62	16	17	9	5
	15	71	11	15	7	5
	20	71	10	15	7	5

Dowell-Jarratt v. Our Method continued

Example	n value	Bisection Method	False Position	False Position and Bisection	Illinois Method	Our Method
4	2	1	1	1	1	2
	5	64	54	19	8	8
	15	61	179	23	11	15
	20	62	245	23	12	12
6	1	64	26	19	9	7
	5	63	114	21	9	12
	10	57	1,286	23	13	14
	15	55	10,000	21	16	18
7	2	59	2008	2	14	14
	5	56	809	25	14	16
	15	61	262	25	14	12
	20	58	192	25	15	13

In this example 2 and 3, our method slightly outperforms the Illinois method, and simultaneously outperforms the other methods significantly. In all the other examples, our method was competitive or slightly outperformed the Illinois method depending on different values of n. With the exception of examples 4 and 7 with the value of n being 2, our method significantly outperformed the methods other than Illinois.

Ford v. Our Method

Example	Initial Bracket	Illinois Method	Pegasus Method	A-B	M1	M2	M3	M4	Our Method
1	$[0, 1.5]$	8	7	6	8	7	6	7	7
	$[-1, 3]$	11	10	10	13	11	10	10	10
	$[-1.5, 6]$	20	20	17	19	18	17	17	20
3	$[0, 1]$	9	10	11	10	14	9	10	10
	$[-0.1, 1.5]$	15	14	43	11	46	12	12	15
	$[-0.5, 2]$	33	31	200+	16	200+	14	13	35
	$[-1, 4]$	47	44	200+	20	200+	16	16	48
4	$[0.035, 0.05]$	14	14	18	9	21	12	10	18
	$[0.03, 0.09]$	27	26	200+	15	200+	7	13	28
	$[0.025, 0.5]$	49	46	200+	19	200+	17	15	50
	$[0.02, 1]$	58	55	200+	19	200+	18	18	60
6	$[0.095, 1.0]$	35	33	12	17	14	12	13	39
	$[0.075, 0.15]$	23	23	200+	17	200+	16	16	27
	$[.08, 0.5]$	38	35	200+	31	200+	22	21	40
	$[0.05, 0.2]$	36	34	200+	21	200+	15	18	38
7	$[0.9, 1.05]$	8	8	9	9	10	7	8	8
	$[0.7, 1.2]$	15	14	23	13	24	11	11	17
	$[0, 2.5]$	42	42	200+	19	200+	16	16	44
	$[-0.5, 5]$	60	59	200+	21	200+	19	19	62
8	$[550, 560]$	7	5	5	6	6	5	6	12
	$[525, 590]$	11	9	9	9	10	9	9	11
	$[400, 600]$	29	27	18	15	20	12	12	31
	$[350, 850]$	44	42	200+	15	200+	15	19	46

Ford v. Our Method (continued)

Example	Initial Bracket	Illinois Method	Pegasus Method	A-B	M1	M2	M3	M4	Our Method
9	$[0.005, 0.02]$	10	7	7	7	8	9	8	8
	$[0.001, 0.05]$	13	13	7	12	9	11	10	16
	$[0.0001, 0.1]$	17	16	8	11	11	12	12	20
	$[0.001, 100]$	21	18	17	13	23	17	22	19
10	$[0, 2]$	13	14	12	11	13	11	11	17
	$[-4, 2]$	19	17	32	14	36	10	12	20
	$[-10, 3]$	43	42	200+	16	200+	14	14	47
	$[0.5, 3.5]$	51	48	11	13	12	10	12	53
11	$[0.5, 2]$	10	6	5	5	8	7	8	9
	$[0.2, 6]$	12	10	5	10	9	9	10	12
	$[0.05, 20]$	15	11	5	12	9	10	11	16
	$[0.004, 200]$	21	18	7	13	10	13	15	20

In most problems, our method was competitive or not too far off from all the methods except for Method 1, Method 2 and Method 4, which seem to be the best in these tests. One thing to notice is example 10 where our method failed to beat any other method for any of the initial brackets.

Various Common Methods v. Our Method

Example	Bracket	Secant Method	FP	MFP	Inv. Quad. Int.	Bisection Method	Our Method
1	$[1, 4]$	13	34	10	8	36	9
2	$[-3, -1.3]$	12	102	11	12	35	11
3	$[3, 4]$	9	14	9	8	35	7
4	$[0, \frac{\pi}{6}]$	8	11	8	7	34	6

In these examples, our method proved to be the best in every case except for where Inverse Quadratic Interpolation outperformed ours by one iteration in example 1 and our modified version of Modified False Position tied our method in example 2.

10 Discussion

In summary, we have created a hybrid method by using our modified version of the Modified False Position method, which changes the divisor of 2 to 1.5, until the estimate, p_i to the root p of $f(x) = 0$ is within a certain tolerance, then switches to Müller's method. We compared this method some of the well-known root finding methods studied in the field of numerical analysis as well as some lesser known methods. The lesser known methods include Brent's method, Illinois method, Pegasus method, one created by Anderson-Björck and the four created by Ford. We have found our method to be quite competitive in many of the twenty-five examples considered, better than all the other methods in two examples, and worse than the others in just one example.

There are other methods that we haven't considered. In section 8, we discussed five methods that Ford used. Ford introduced three possible numerators (A1, A2 and A3) and denominators (B1, B2 and B3) for new methods of solving equations. Clearly there are 9 possible combinations. Ford considered the combination of A3 and B3 as "Method 1", A2 and B1 as "Method 2", A2 and B3 as "Method 3", A1 and B3 as "Method 4"and A3 and B1 as "Method 5". Anderson and Björck's method uses A1 and B1. This leaves the combinations of A1 and B2, A2 and B2, and A3 and B2. We didn't consider these either in the interest of time as well as the excellent results of our hybrid method that took several months to perfect.

11 References

[1] Dowell, M. and Jarratt, P. *A modified Regula Falsi method for computing the root of an equation*, BIT, Volume 11, pp 168-174 (1971).
[2] Dowell, M. and Jarratt, P. *The Pegasus method for computing the root of an equation*, BIT, Volume 12, pp 503-508 (1972).
[3] Anderson, N. and Björck. A. *A new high order method of Regula Falsi type for computing the root of an equation*, BIT, Volume 813, pp 253-264 (1973).
[4] Burden, R. and Faires, J. *Numerical Analysis* (8th edition), Thomson, Brooks/Cole, pp 46, 63-64, 67-70, 75, 92-93, 108, 117 (2005).
[5] Ford, J.A. *Improved algorithms of Illinois-type for the numerical solution of nonlinear equations*, Technical Report CSM-257, University of Essex (1995)
[6] Brent, R. *Algorithms for minimization without derivatives*, Englewood Cliffs, NJ, pp 21-22, 48-53 (1973).

[7] Dahlquist, G. and Björck, A. (Translated by Ned Anderson) *Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, pp 229-232 (1974).
[8] Atkinson, K. *An Introduction to Numerical Analysis* (2^{nd} edition), Wiley and Sons, pp 73-75 (1989).

12 Appendix

The following is the code of our hybrid method written in Maple.

```
restart; Digits := 15; $f := x \to f(x)$; $F := x \to D(f)(x)$; $p0 := \gamma$; $p1 := \delta$; $p2 := \frac{(p_0+p_1)}{2.0}$;
# Note: here $f(x)$ is the function in consideration, $\gamma$ is a lower bound approximation
to the root $p$ of $f(x) = 0$ and $\delta$ is an upper bound approximation to $p$ so that $p \in [\gamma, \delta]$.
$TOL := 10^{-10}$; $TOL1 := .1$; $den := 2.0$;
$eps := evalf(TOL + 2^{-53} * max(abs(p0), abs(p1), 1))$; $eps1 := .95 * eps$;
# Note: here $eps$ is the tolerance discussed by Ford [5],
# Input the maximum number of iterations desired.
$N := 200$;
$p[0] := p0$; $q[0] := evalf(f(p[0]))$; $q0 := q[0]$; $p[1] := p1$; $q[1] := evalf(f(p[1]))$; $q1 := q[1]$;
$p[2] := p2$; $q[2] := evalf(f(p[2]))$; $q2 := q[2]$; $side := 0$; $a[1] := p0$; $b[1] := p1$; $sw := 1$;
$sw2 := 0$;
for $i$ from 2 to $N$ do
    if $sw = 1$ then
        $p2 := \frac{(q1*p0-q0*p1)}{(q1-q0)}$; print$(i, p0, p1, p2, q0, q1, q2, side, sw)$ : print$(abs(p2 - p1))$ :
        $q2 := f(p2)$ : $p[i] := p2$ : $q[i] := q2$ :
            if $abs(p2 - p1) < eps1$ or $abs(q2) < eps$ then break: end if:
            if $q2 * q0 > 0$ then
                $p0 := p2$ : $q0 := q2$ :
                if $side = -1$ then $q1 := \frac{q1}{den}$ : end if:
                $a[i] := p2$ : $b[i] := b[i - 1]$ : $side := -1$ :
            else
                if $q2 * q1 > 0$ then
                    $p1 := p2$ : $q1 := q2$ : $b[i] := p2$; $a[i] := a[i - 1]$ :
                    if $side = 1$ then $q0 := \frac{q0}{den}$ : end if:
                    $side := 1$
            else break:
            end if:
            end if:
```

```
            if $abs(p[i]-p[i-1]) < TOL1$ then $sw := 0 : e1 := abs(p[i]-p[i-1])$ end if:
            if $sw2 = 1$ then $sw := 1$: end if:
        else
            $x2 := p[i-1] : x1 := p[i-2] : x0 := p[i-3]$ :
            $f2 := f(x2) : f1 := f(x1) : f0 := f(x0)$ :
            $h0 := x1 - x0 : h1 := x2 - x1 : del10 := \frac{(f1-f0)}{h0} : del11 := \frac{(f2-f1)}{h1}$ :
            $del2 := \frac{(del11-del10)}{(h1+h0)}$ :
            $B := del11 + h1 * del2$ :
            $D2 := B * B - 4 * f2 * del2$ :
            $D1 := sqrt(D2)$ :
            $E1 := B + D1$ :
            if $abs(B - D1) > abs(E1)$ then $E1 := B - D1$ end if:
            $h2 := -\frac{2*f2}{E1}$ :
            $p[i] := x2 + h2 : p3 := p[i] : q3 := f(p3) : q[i] := q3 : e2 := abs(p[i]-p[i-1])$ :
            print$(i, p[i], sw)$ : print$(e2)$ :
            if $(e2 < eps1, abs(q2) < eps)$ then break: end if:
            if $(e2 > e1, abs(q3) > abs(q2))$ then
                        $p0 := a[i-1] : p1 := b[i-1] : q0 := f(p0) : q1 := f(p1) : sw := 1$ :
                        $sw2 := 1 : a[i] := a[i-1] : b[i] := b[i-1]$ :
            else
                        if $q3 * f(a[i-1]) > 0$ then
                            $a[i] := p3 : b[i] := b[i-1]$ :
                        else
                            $b[i] := p3 : a[i] := a[i-1]$ :
                        end if:
                        if $b[i] < a[i]$ then
                            $cc := a[i] : a[i] := b[i] : b[i] := cc$:
                        end if:
            end if:
        end if:
    end:
    for $j$ from 1 to $i$ do
    print($j$, $a[j]$, $b[j]$):
    end:
```