Jonathon Bryant
Professor Oman
Calculus
2 April 2016

Analysis of Newton's Method and False Position Method

Newton's method and the false position method are both root-finding algorithms, which are algorithms that try to find a root of a function. Newton's method and false position method can be compared and contrasted, even though both methods are different. Newton's method and false position method use successive approximations to come to a solution. When successive approximations come to a solution it is called converging:

> If the numbers x_n become closer and closer to r as n becomes large, then we say that the sequence converges to r. (Stewart 339)

The false position method and Newton's method are root-finding algorithms that use the intermediate value theorem to approximate the roots of a function. The Intermediate Value Theorem states:

> Suppose that f is continuous on the closed interval [a,b] and let N be any number between f(a) and f(b), where $f(a) \neq f(b)$. Then there exists a number c in (a,b) such that f(c) = N. (Stewart 125)

How the Intermediate Value Theorem works is that you find a set {a,b} that has an f(a) that is one sign, negative or positive, and an f(b) that is the opposite sign, with these conditions there is guaranteed to be a root in (a,b).

The Intermediate Value Theorem is the underlying reason that the false position method works, but the false position method has its own sequence of steps. Here is how Reinhard in "1.C. An Example of the False Position Method" explains how the false position method works.

First have two points that when plugged into the function have opposite signs and are nonzero, with these points there is a root between them and this can be proved by the Intermediate Value Theorem stated above. Next find the line joining the two points (a, f(a)), and (b, f(b)),

$$y = f(b) + \frac{f(b) - f(a)}{b - a}(x - b)$$

$$Substitute\ x\ for\ C\ (the\ root\ of\ the\ line) \wedge set\ y = 0$$

$$f(b) + \frac{f(b) - f(a)}{b - a}(C - b) = 0$$

$$\frac{f(b) - f(a)}{b - a}(C - b) = -f(b)$$

$$(C - b) = -f(b)\frac{b - a}{f(b) - f(a)}$$

$$C = b - f(b)\frac{b - a}{f(b) - f(a)}$$

Evaluating f(c) to determine if it is positive, negative, or zero is the next step. If f(c) is evaluated to be zero, then the root has been found. If f(c) is positive or negative, then replace a or b with C, this is determined by whichever one has the same sign as C. Here is some C++ code to demonstrate the false position method:

```
void ReguliFalsi(){

	double guessA, guessB, C, fA, fB, fC;
	int side = 0;
	guessA = 0;
	guessB = 2000;
	double error = 10E-15;
	fA = (guessA * guessA) - 2;
	fB = (guessB * guessB) - 2;
	int i = 0;

	for(;;){
		i++;
		C = (fA * guessB - fB * guessA)/(fA - fB);
		if (fabs(guessB-guessA) < error*fabs(guessB+guessA)){
			cout<<"Iterations:"<<i<<endl;
			break;
		}
		fC = (C * C) - 2;
		if(fC * fB > 0){
			guessB = C; fB = fC;
```

```
                        if(side == -1)
                                fA /= 2;
                        side = -1;}
                else if(fA * fC > 0){
                        guessA = C; fA = fC;
                        if(side == 1)
                                fB /= 2;
                        side = 1;}
                else{
                        cout<<"Iterations:"<<i<<endl;
                        break;
                }
        }
        printf("%f", C);
        cout<<endl;
}
```

The code uses a different formula then the one stated above, which can be derived by:

$$C = b - f(b)\frac{b-a}{f(b)-f(a)}$$

$$C = b\frac{f(b)-f(a)}{f(b)-f(a)} - f(b)\frac{b-a}{f(b)-f(a)}$$

$$C = \frac{bf(b)-bf(a)-bf(b)+af(b)}{f(b)-f(a)}$$

$$C = \frac{bf(a)-af(b)}{f(b)-f(a)}$$

The output for the code is the number of times it went through the loop or iterations, which is 20, and it give the approximation 1.414214, so the root is $\sqrt{2}$.

The advantages of the false position method is that it always converges on a root and it does not require a derivative of the function in order to find a root. The disadvantages is that it is slower then most root-finding methods, although it can be faster than the bisection method. However, this is not always the case and the false position method can also become slower than the bisection method in some circumstances.

Newton's method is another root-finding method, like the false position method, although Newton's method finds the root in a different way. Newton's method, and how it finds a root of a function, is best described by Paul Dawkins who wrote "Calculus I - Newton's Method." First

choose an initial approximation and call it x_0 and then find the tangent line of f(x) at x_0. This looks like:

$$y = f(x_0) + f'(x_0) * (x - x_0)$$

Now pick the point on the tangent line that crosses the x-axis and name it x_1.

$$0 = f(x_0) + f'(x_0) * (x - x_0)$$
$$x_1 - x_0 = \frac{f(x_0)}{f'(x_0)}$$
$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Now repeat by plugging in the x_1 you get from the equation to where x_0 is in the equation. This process is repeated until a very good approximation to the actual root is found. Here is some C++ code to demonstrate Newton's method:

```
void NewtonsMethod(){
	double x0 = 2000;
	double x1;
	double f;
	double fprime;
	double y;
	double yprime;
	double tolerance = pow(10,-7);
	double epsilon = pow(10, -15);
	int i = 0;
	bool Solution = false;

	for(;;){
		i++;
		f = (x0 * x0) - 2;
		fprime = 2 * x0;
		y = f;
		yprime = fprime;

		if(abs(yprime) < epsilon){
			cout<<"Iterations:"<<i<<endl;
			break;
		}

		x1 = x0 - (y/yprime);

		if(abs(x1 - x0) <= (tolerance * abs(x1))){
```

```
                        Solution = true;
                        cout<<"Iterations:"<<i<<endl;
                        break;
                }
                x0 = x1;
        }
        if(Solution == true){
                cout.setf(ios::fixed);
                cout.setf(ios::showpoint);
                cout.precision(6);
                cout<<"Solution:"<<x1<<endl;
        }
        else
                cout<<"Did not converge"<<endl;
}
```

With the Newton's method an approximation to $\sqrt{(2)}$ was 1.414214 and it took 15 iterations in order to reach this approximation.

The advantage of Newton's method is that it can be very fast, especially compared to the false position method. The disadvantages are that the function has to have a derivative and that Sometimes Newton's method will not converge. This usually happens for two reasons: the first is when x_n does not exist as n approaches infinity and second is when the derivative of x_n equals zero (“10.6 Newton's Method”). When Newton's method fails to converge on a root, then a better initial value should be chosen.

Big-O notation is used to compare two different algorithms. Big-O notation helps describe a function, and the letter O stands for order of the function. Big-O describes the functions asymptotic behavior as x goes towards infinity(“Big O Notation”). Big-O notation gives the worst-case scenario of the algorithm. Using Big O notation can give a better understanding of the algorithm being analyzed because it takes only the algorithm into consideration. This means no other variables are introduced, like what kind of computer hardware is being used to test the algorithm.

The examples of code for both of the methods show the number of iterations it took to

find the root of the function $x^2 - 1$. Newton's method took 15 iterations in order to find the root compared to the false position method which took 20 iterations. This evidence suggests that Newton's method is faster. This can also be seen with big-O notation. Newton's method is $O(n^2)$, which means that the error of Newton's method gets small very quick, since squaring a small number makes it much smaller. Compared to the false position method, which is O(n) or linear in error, meaning the error for false position gets smaller at a linear rate. Comparing Newton's method and false position method in Big-O terms it is seen that Newton's method is preferable since the error for Newton's method gets smaller faster then the error for false position method.

Works Cited

Stewart, James. *Single Variable Calculus: Early Transcendentals*. Australia: Brooks/Cole Cengage Learning, 2012. Print.

Reinhard. "1.C. An Example of the False Position Method." *UNIVERSITY OF CALIFORNIA, RIVERSIDE*. UNIVERSITY OF CALIFORNIA, RIVERSIDE, 27 Mar. 2016. Web. 2 Apr. 2016.

Dawkins, Paul. "Calculus I - Newton's Method." *Paul's Online Math Notes*. Paul Dawkins. Web. 04 Apr. 2016.

"10.6 Newton's Method." *UCDAVIS MATHMATICS*. University of California, Davis. Web. 9 Apr. 2016. <https://www.math.ucdavis.edu/~thomases/W11_16C1_lec_3_11_11.pdf>.

"Big O Notation." - *Wikipedia, the Free Encyclopedia*. Wikipedia. Web. 09 Apr. 2016. <https://en.m.wikipedia.org/wiki/Big_O_notation>.