Golden Section Search Algorithm

Alexander MacKenzie

21 November 2018

1 Introduction

There is a value described in mathematics known as the *golden ratio*. It is described as being the ratio of a line segment where the larger piece is equal to the ratio of the larger piece to the smaller piece [1]. Although this ratios purpose is often very misleading when presented in various other schools of thought [1], it has an interesting and useful function to play in efficiently finding the maximum or minimum of a continuous unimodal function [2]. The *Golden Section Search Algorithm* makes use of this ratio to quickly find a maximum or minimum between a given interval by using the *golden ratio* to find which next value to move forward with as an interval for constraints to narrow down where the minimum (or maximum) will be [2]. In this report we will describe a function and interval and use the *Golden Section Search Algorithm* to find an appropriate and accurate bracket in which a minimum should exist.

2 Summary of Results

First, to compute a function's minimum using the *Golden Section Search Algorithm* we must decide on an interval to test and how many iterations we would like to use. We will use the continuous unimodal function described as $F(x) = x^6 - 11x^3 + 17x^2 - 7x + 1$. If we look at the plot of $F(x)$ (Figure 1) then we can see that the minimum should be located within the interval $[0, 1]$.

If we take this function and compute it to the fifteenth iteration we are left with the bracket values of $[0.2833854, 0.2841186]$ (shown in Table 1, line 16), indicating that the minimum of $F(x)$ lies somewhere in between with a tolerance of 10^-2. At this point we can conclude what the minimum is by computing the mid-point between these two values by taking their mean $(0.2833854 + 0.2841186)/2 = 0.283752$. If we want to find the minimum computed to a tolerance of 10^-7, we will need to compute a few steps further. If we increase the iterations to 40 (shown in Table 2) we see that the bracketing values of $[a, b]$ remain the same after the thirty-seventh iteration (line 38 of Table 2).

Therefore, with a tolerance of 10^-7 we can conclude the minimum of the function $F(x) = x^6 - 11x^3 + 17x^2 - 7x + 1$ on the interval $[0, 1]$ is 0.2836484.

3 Discussion

As it can be seen, the *Golden Section Search Algorithm* is able to quickly find where the minimum value lies. By the eleventh iteration (line 12, Table 1) our bounding interval had already reached a tolerance of 10^-2. The *Golden Section Search Algorithm* can either be very quick or take some time, but regardless of how long it takes or how accurate you wish it to be it is guaranteed to find an interval that contains the minimum as long as your function is continuous and unimodal on the interval being tested. This is guaranteed by two things: the nature of the computations taking place in the algorithm since it simply applies the *golden ratio* to each bound, checks which is smaller, and sets the new interval. The second guarantee is due to the fact that because we require a starting interval where we know a minimum or maximum must exist the only constraint is how accurate you wish it to be.

Although some minimum and maximum finding methods may be faster, the reliability and computational efficiency of the *Golden Section Search Algorithm* makes it a prime choice for estimating these respective values.

4 Code

```
#Golden Method Section Search Algorithm script for R
#@Alexander MacKenzie

#Given f unimodal with minimum in [a,b]

#Setting variables
k=40      #Iterations
a=0       #Interval
b=1
g=((sqrt(5)-1)/2)
FofX <- function(x){
          x^6-11*(x^3)+17*(x^2)-7*x+1 #Function f
}

#Graph of F(x)
x=seq(-3,3, by=1/32)
plot(x, FofX(x), xlim = c(-1, 2), ylim = c(-1, 2)
     , main='Golden Section Search'
     , ylab='y', type="l", col="red")

#Setting table vectors
A=c()
x1=c()
x2=c()
B=c()
```

```
#Golden Method Section Search
for(i in 0:k){
        #x1
        x=a+(1-g)*(b-a)
        x1=c(x1,x) #Appending to table
        xa=FofX(x)

        #x2
        x=a+g*(b-a)
        x2=c(x2,x) #Appending to table
        xb=FofX(x)

        #Appending to table
        A=c(A,a)
        B=c(B,b)

        if(xa < xb){
                b=a+g*(b-a)
        }
        else{
                a=a+(1-g)*(b-a)
        }
}

#Results
table <- data.frame(A,x1,x2,B)
print(table)
cat('Therefore, after ',k,' steps
     , the minimum is between '
     , A[k],' and ', B[k])
```

References

[1] George Markowsky. Misconceptions about the golden ratio. *The College Mathematics Journal*, 23(1), Jan 1992.

[2] Timothy Sauer. *Numerical Analysis*. Always learning. Pearson, 2012.

5 Graphs and Tables

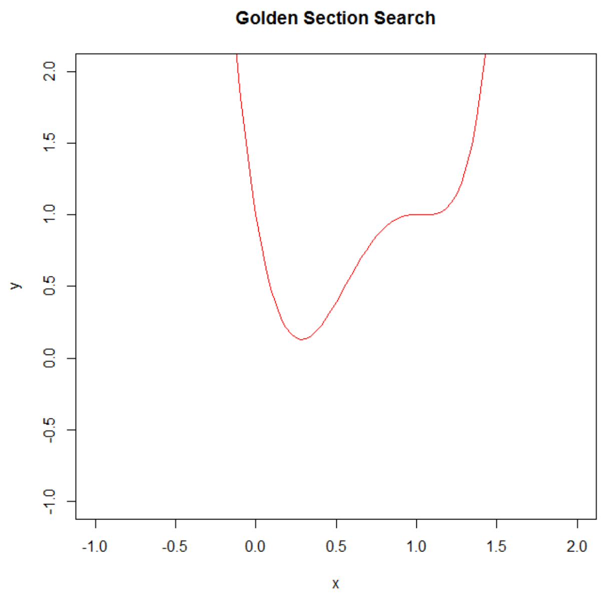

Figure 1: Plot of $F(x) = x^6 - 11x^3 + 17x^2 - 7x + 1$

	A	x1	x2	B
1	0.0000000	0.3819660	0.6180340	1.0000000
2	0.0000000	0.2360680	0.3819660	0.6180340
3	0.0000000	0.1458980	0.2360680	0.3819660
4	0.1458980	0.2360680	0.2917961	0.3819660
5	0.2360680	0.2917961	0.3262379	0.3819660
6	0.2360680	0.2705098	0.2917961	0.3262379
7	0.2705098	0.2917961	0.3049517	0.3262379
8	0.2705098	0.2836654	0.2917961	0.3049517
9	0.2705098	0.2786405	0.2836654	0.2917961
10	0.2786405	0.2836654	0.2867711	0.2917961
11	0.2786405	0.2817461	0.2836654	0.2867711
12	0.2817461	0.2836654	0.2848517	0.2867711
13	0.2817461	0.2829323	0.2836654	0.2848517
14	0.2829323	0.2836654	0.2841186	0.2848517
15	0.2829323	0.2833854	0.2836654	0.2841186
16	0.2833854	0.2836654	0.2838385	0.2841186

Table 1: Table showing 15 iterations of the Golden Section Search Algorithm computed in r

	A	x1	x2	B
17	0.2833854	0.2835585	0.2836654	0.2838385
18	0.2835585	0.2836654	0.2837316	0.2838385
19	0.2835585	0.2836246	0.2836654	0.2837316
20	0.2836246	0.2836654	0.2836907	0.2837316
17	0.2833854	0.2835585	0.2836654	0.2838385
18	0.2835585	0.2836654	0.2837316	0.2838385
19	0.2835585	0.2836246	0.2836654	0.2837316
20	0.2836246	0.2836654	0.2836907	0.2837316
21	0.2836246	0.2836498	0.2836654	0.2836907
22	0.2836246	0.2836402	0.2836498	0.2836654
23	0.2836402	0.2836498	0.2836558	0.2836654
24	0.2836402	0.2836462	0.2836498	0.2836558
25	0.2836462	0.2836498	0.2836521	0.2836558
26	0.2836462	0.2836484	0.2836498	0.2836521
27	0.2836462	0.2836476	0.2836484	0.2836498
28	0.2836476	0.2836484	0.2836490	0.2836498
29	0.2836476	0.2836481	0.2836484	0.2836490
30	0.2836481	0.2836484	0.2836486	0.2836490
31	0.2836481	0.2836483	0.2836484	0.2836486
32	0.2836481	0.2836482	0.2836483	0.2836484
33	0.2836482	0.2836483	0.2836484	0.2836484
34	0.2836483	0.2836484	0.2836484	0.2836484
35	0.2836483	0.2836483	0.2836484	0.2836484
36	0.2836483	0.2836484	0.2836484	0.2836484
37	0.2836483	0.2836484	0.2836484	0.2836484
38	0.2836484	0.2836484	0.2836484	0.2836484
39	0.2836484	0.2836484	0.2836484	0.2836484
40	0.2836484	0.2836484	0.2836484	0.2836484
41	0.2836484	0.2836484	0.2836484	0.2836484

Table 2: Table showing iterations 16 through 40 of the Golden Section Search Algorithm computed in r