https://doi.org/10.31871/IJNTR.8.8.6

International Journal of New Technology and Research (IJNTR)
ISSN: 2454-4116, Volume-8, Issue-8, August 2022 Pages 07-11

Root Finding

Temidavo Kirsten Adekunle

***Abstract*— In this project we studied root finding using bisection, newton and secant method. This will be done by evaluating the ideal gas equation of pv = nrt. The aim is to observe the speed with which each algorithm evaluates this equation. We will use an estimated initial volume to approximate the value of the function given, as well as calculate the error between the actual r and the computed root. Finally we will observe the rate at which the error converges to zero and thus with each iteration, we get closer to the true value r.**

***Index Terms*— Root finding**

I. Introduction

Bisection method evaluates a function by calculating a midpoint between two values to find its root. In this case our estimated values is the volume computed +- 1 because we want a point accurately in the center. By so doing we use the midpoint to evaluate the ideal gas equation for each step. Eventually we observe how long it actually takes to get to the exact root we try to find by calculating the error between the given r in the problem and that computed with the algorithm. This method is less efficient because it takes a lot longer number of steps to get our true value. It also converges linearly at a constant rate of 1/2.

Using newtons method the algorithm works in the way of calculating each step by dividing its function by its derivative and subtracting it from the current point. (Xn – (f(Xn)/ fprime(Xn)). Newtons method is a much more convenient way of computing the root of this function because it takes a lot less steps. This is because newtons method converges quadratically and get to its approximated root a lot faster than other methods. The only downside of this method though is that it is much more difficult to compute the derivative of the function especially for a lot more complicated functions.

Secant method will calculate the true value by dividing its function by its derivative and subtracting it from the current point. The derivative here though is more difficult to compute than the newton method because it finds the derivative between two points, which is costly. (Xn – (f(Xn) – f(Xn – 1))/(Xn- Xn – 1)) . Using secant method the algorithm is even much worse to compute than the newtons method. This makes it worse than newtons because it also takes a longer time to converge and so is not worth the trouble of calculating if the newton method can be utilized instead. It converges linearly and quadratically and thus between bisection and newton methods.

We will find roots using the bisection, newton and secant method to evaluate the ideal gas equation of
p*v = n*r*T

- The derivative I derived is fprime(V) :

y = ((n^2*a)*(-2/V^3)*(V - n*b)) + (P + (n^2*a)/(V^2));

- The computed value Vo:

Vo = n*r*T/ p

This was gotten to be as follows using the bisection method:

Index	f(V)
1	12.84239000
2	12.34239000
3	12.59239000
4	12.71739000
5	12.65489000
6	12.62364000
7	12.63926500
8	12.64707750
9	12.65098375
10	12.65293688
11	12.65196031
12	12.65147203
13	12.65122789
14	12.65110582
15	12.65104479
16	12.65107530
17	12.65109056
18	12.65109819
19	12.65110201
20	12.65110010

Temidayo Kirsten Adekunle, Department- Engineering Management, Information and Systems

- The value of f(a)*f(b) is assumed to be

-3.355115669324923e-12.

Bisection method

Index	Vo	f(V)	|r-V|	Rate of Convergence
1.00000	12.84239	0.37524	0.19129	0.00000
2.00000	12.34239	-0.60509	0.30871	1.61382
3.00000	12.59239	-0.11512	0.05871	0.19018
4.00000	12.71739	0.13001	0.06629	1.12913
5.00000	12.65489	0.00743	0.00379	0.05718
6.00000	12.62364	-0.05385	0.02746	7.24394
7.00000	12.63927	-0.02321	0.01183	0.43098
8.00000	12.64708	-0.00789	0.00402	0.33984
9.00000	12.65098	-0.00023	0.00012	0.02874
10.00000	12.65294	0.00360	0.00184	15.89743
11.00000	12.65196	0.00169	0.00086	0.46855
12.00000	12.65147	0.00073	0.00037	0.43287
13.00000	12.65123	0.00025	0.00013	0.34493
14.00000	12.65111	0.00001	0.00001	0.05043
15.00000	12.65104	-0.00011	0.00005	8.41436
16.00000	12.65108	-0.00005	0.00002	0.44058
17.00000	12.65109	-0.00002	0.00001	0.36513
18.00000	12.65110	0.00000	0.00000	0.13061
19.00000	12.65110	0.00001	0.00000	2.32814
20.00000	12.65110	0.00000	0.00000	0.28524

The bisection method it is observed is a bit erratic when calculating the error to find out how close we are to our true value. In the first step it increases, and reduces in the second, increases in the third and reduces in the fourth. From the fifth step onwards it keeps on decreasing incrementally (or faster) until it reaches zero.

Newton method

Index	Vo	f(V)	|r-V|	Rate of Convergence
1.00000	-3.98483	-35.36759	16.63593	0.00000
2.00000	18.22118	10.98932	5.57008	0.33482
3.00000	12.67430	0.04550	0.02320	0.00417
4.00000	12.65110	0.00000	0.00000	0.00004
5.00000	12.65110	0.00000	0.00000	0.00000
6.00000	12.65110	0.00000	0.00000	0.75000
7.00000	12.65110	0.00000	0.00000	1.00000
8.00000	12.65110	0.00000	0.00000	1.00000
9.00000	12.65110	0.00000	0.00000	1.00000
10.00000	12.65110	0.00000	0.00000	1.00000

In this method, we observe that we get to our true value fastest. The error approximately decreases in the power of 3. In the first step it decreases by 3 and the second decreases by 24. Clearly it does this very fast.

Secant method

Index	Vo	f(V)	|r-V|	Rate of Convergence
1.00000	0.00000	negativeInf	0.00000	0.00000
2.00000	0.00000	negativeInf	0.00000	0.00000
4.00000	10.68078	-3.84981	1.97032	0.06167
6.00000	12.65137	0.00054	0.00027	0.00368
8.00000	12.65110	0.00000	0.00000	0.00000
10.00000	12.65110	0.00000	0.00000	1.00000

In this method we observe that the error does reduce quite quickly too. It does this both linearly and quadratically. In the first step it reduces by about 16 and in the second reduces by about 26. In the third it reduces a little slower by about 28.

In this project we have observed how to approximate a function to a specific value by using different root finding methods. We used the bisection, newton and secant methods of approximation. It was observed that the newton method is the fastest way to converge to our given r because it took only about 4 steps. Secant is the second best alternative, which took about 7 steps, although a little more difficult to compute. This is because just like newton is involves calculating the derivative of a function which is demanding. The last method bisection is the easiest to compute but it takes a lot more steps to converge to our true value. This project was an interesting one and teaches that we can obtain a precise value as long as the function is evaluated precisely.

7) Matlab code F.m

```
function[y]=f(V) a = 6.29;
b = 0.0562;
R = 0.08206;

n = 1;
P = 2;
T = 313;

y = ( P + (n^2*a/V^2)) *(V - n*b) - n*R*T;

end
```

Fprime

```
function[y]= fprime(V) a = 6.29;
b = 0.0562;
R = 0.08206;

n = 1;
P = 2;
T = 313;

y = (( n^2*a)*(-2/V^3)*(V - n*b)) + (P + (n^2*a)/(V^2));

end
```

Bisection method

```
%Bisection Method

format long
R = 0.08206; n = 1; p = 2; T = 313; r =

12.651099337114202;

V0 = (n*R*T)/p; bisect = zeros( 20,5);

n = 0; a0 = V0 - 1; b0 = V0 + 1;

for n = 1 : 20

Xn = 0.5*(a0 + b0); bisect(n,2) = Xn; Y1=f(Xn); Y2=f(a0);

if Y1*Y2 < 0,

b0 = Xn; else a0 = Xn; end


end

Index = [ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
20]; Index = Index';
bisect( :, 1) = Index;
for n = 1 : 20
bisect( n, 3) = f(bisect(n,2)); bisect( n, 4) = abs(r - bisect(n,2));

ifn >= 2
bisect(n,5) = (abs(r - bisect(n,2))/ bisect(n-1,4)); %convergence rate

end end
```

Newton method

```
% Newton method

R = 0.08206; n = 1; p = 2; T = 313; r =
12.651099337114202;

V0 = 1;
newton = zeros( 10,4);

Xn = V0;
for n = 1 : 10
Xn = Xn - (f(Xn)/fprime(Xn));

newton( n, 1) = n;
newton( n, 2) = Xn;
newton( n, 3) = f(newton(n,2)); newton( n, 4) = abs(r -
newton(n,2));

if n >= 2

newton(n,5) = (abs(r - newton(n,2))/ newton(n-1,4));

endend
```

Secant method

```
%Secant method

R = 0.08206; n = 1; p = 2; T = 313; r =
12.651099337114202;

V0 = 1;
secant = zeros( 10,4);

Xn= zeros(10);

Xn(1)= V0; Xn(2)=V0+1;

secant( 1, 3) = f(secant(1,2)); secant( 2, 3) =
f(secant(2,2));

for n = 3 : 10
derivative = (f(Xn(n-1)) - f(Xn(n-2))) / (Xn(n-1) - Xn(n-
2)); Xn(n) = Xn(n-1) - (f(Xn(n-1))/ derivative);

secant( 1, 1) = 1; secant (2, 1) = 2;

secant( n, 1) = n;

secant( n, 2) = Xn(n);
secant( n, 3) = f(secant(n,2)); secant( n, 4) = abs(r - secant(n,2));
```

```
if n >= 2

secant(n,5) = (abs(r - secant(n,2))/ secant(n-1,4));

end

end
```