Ngoc Huynh

Professor Sauer

Math 446

September 3, 2019

Project 1

The function $f(x) = 98e^{5x2} - (1225x^4 + 280x^2 + 66)\ e^{3/7}$

1. **Use the Intermediate Value Theorem to show that a root is guaranteed in the interval [0,1].**

   Using MatLab, it shows that:

```
>> f
f =
  function_handle with value:
    @(x)98*exp(5*x^2)-(1225*x^4+280*x^2+66)*exp(3/7)
>> f(0)
ans =
            -3.31415861084385
>> f(1)
ans =
             12132.9056045126
>> f(0) * f(1)
ans =
            -40210.3735837509
```

   Which guarantees that the root will be in the interval of [0,1].

   Apply the Bisection Method on the starting interval [0,1] to find the root to 10 correct places:

- Choose the tolerance = 5×10^{-11} to guarantee at least 10 correct digits.
- Using the textbook's code within Matlab:

```
bisect.m  ×  bisect_method.m  ×  +
%Program 1.1 Bisection Method
%Computes approximate solution of f(x) = 0
%Input: function handle f; a,b such that f(a)*f(b)<0,
% and tolerate tol
%Output: approximate solution xc function xc = bisect(f,a,b,tol)
if sign(f(a))*sign(f(b)) >= 0
    error ('f(a)f(b) < 0 not satisfied!') %ceases execution
end
fa = f(a);
fb = f(b);
while (b-a)/2>tol
    c=(a+b)/2;
    fc=f(c);
    if fc == 0  %c is a solution, done
        break
    end
    if sign(fc)*sign(fa)<0 %a and c make the new interval
        b=c;fb=fc;
    else %c and b make the new interval
        a=c;fa=fc;
    end
end
xc=(a+b)/2; %new midpoint is best estimate
```

Assign a = 0, b = 1, calling bisect to find the root.

The root found is **0.402496639901074**

But since it is more than 10 digits after the decimal points, round(xc,10) will give us the approximate root **r = 0.4024966399**.

```
f =
  function_handle with value:
    @(x)98*exp(5*x^2)-(1225*x^4+280*x^2+66)*exp(3/7)
>> a = 0
a =
     0
>> b = 1
b =
     1
>> bisect
>> xc
xc =
         0.402496639901074
>> round(xc,10)
ans =
              0.4024966399
```

Now repeat the Bisection Method for two more other starting intervals [0, b], where 1 < b < 1.5.

- Choose a = 0 and b = 1.2:

```
>> a = 0
a =
     0
>> b = 1.2
b =
                        1.2
>> bisect
>> xc
xc =
         0.402496639906894
>> round(xc,10)
ans =
              0.4024966399
```

- Choose a = 0 and b = 1.4:

```
>> a = 0
a =
     0
>> b = 1.4
b =
                          1.4
>> bisect
>> xc
xc =
         0.402496639901073
>> round(xc,10)
ans =
              0.4024966399
```

Answer:

As MatLab shows that the roots for two different intervals [0,1.2] and [0,1.4] are both 0.4024966399 → They agree with each other up to 10 decimal places.

2. **The backward error, or checking error, is |f(r)|. Calculate it for the approximate root(s) found in Step 1.**

Since all three approximate roots found in step 1 are the same up to 10 decimal places for all three different intervals, I will use just one r = 0.4024966399 to check the backward error using MatLab.

```
>> ans
ans =
              0.4024966399
>> error = abs(f(ans))
error =
      2.22570406549494e-10
```

The error is relatively small and as close to zero as possible, so I am confident that I have the correct first 10 digits of the root.

However, to make sure, I will use the code provided by Professor Sauer to double check:

bisect.m | bisect_method.m*

```
%Code provided by professor Sauer
f = @(x)98*exp(5*x^2)-(1225*x^4+280*x^2+66)*exp(3/7)
num_steps = 100;
a=0; b=1;
for i=1:num_steps
    c=(a+b)/2
    if f(a)*f(c)<0
        b=c;
    else
        a=c;
    end
end
c=(a+b)/2
```

I will assign num_steps to be 100, a = 0 and b = 1. This will give us:

```
f =
  function_handle with value:
    @(x)98*exp(5*x^2)-(1225*x^4+280*x^2+66)*exp(3/7)
c =
                         0.5
c =
                        0.25
c =
                       0.375
c =
                      0.4375
c =
                     0.40625
c =
                    0.390625
c =
                   0.3984375
c =
                  0.40234375
c =
                 0.404296875
c =
                0.4033203125
c =
               0.40283203125
c =
              0.402587890625
c =
             0.4024658203125
c =
            0.40252685546875
c =
           0.402496337890625
c =
           0.40251159667969
```

As the result keeps printing, most of the last results will look the same:

```
c =
        0.402496639904432
c =
        0.402496639904432
c =
        0.402496639904432
c =
        0.402496639904432
c =
        0.402496639904432
c =
        0.402496639904432
c =
        0.402496639904432
c =
        0.402496639904432
c =
        0.402496639904432
c =
        0.402496639904432
c =
        0.402496639904432
c =
        0.402496639904432
c =
        0.402496639904432
c =
        0.402496639904432
c =
        0.402496639904432
c =
        0.402496639904432
c =
        0.402496639904432
```

And if I were to plug back this root = 0.402496639904432 to the function, I will get 0 for |f(x)|.

```
>> abs(f(0.402496639904432))
ans =
     0
```

Answer:

Using both ways to check for the backward error, I am now very confident that my approximate root = 0.04024966399 has the correct first 10 digits.

3. **Now repeat Steps 1 and 2 after replacing the 66 in the function with 65. Just as for f(x), the revised function $f_{new}(x)$ has exactly one root in the interval $0 \le x \le 1$.**

Use the Intermediate Value Theorem to show that a root is guaranteed in the interval [0,1].

Using MatLab, it shows that:

```
f =
  function_handle with value:
    @(x)98*exp(5*x^2)-(1225*x^4+280*x^2+65)*exp(3/7)
>> f(0)
ans =
         -1.77909560158864
>> f(1)
ans =
          12134.4406675218
>> f(0)*f(1)
ans =
         -21588.3300193264
```

Which guarantees that the root will be in the interval of [0,1].

Apply the Bisection Method on the starting interval [0,1] to find the root to 10 correct places:

- Choose the tolerance = $5x10^{-11}$ to guarantee at least 10 correct digits.
- Using the textbook's code within Matlab for these below.

Now first use a = 0 and b = 1:

```
>> a = 0
a =
     0
>> b = 1
b =
     1
>> f=@(x) 98*exp(5*x^2) - (1225*x^4 +280*x^2 + 65)*exp(3/7)
f =
  function_handle with value:
    @(x)98*exp(5*x^2)-(1225*x^4+280*x^2+65)*exp(3/7)
>> bisect
>> xc
xc =
         0.292770385742188
>> round(xc,10)
ans =
              0.2927703857
```

Now repeat the Bisection Method for four more other starting intervals [0, b], where 1 < b < 1.

I will one by one find the approximate roots for intervals [0, 1.1], [0, 1.2], [0, 1.3], [0, 1.4].

```
>> a = 0, b = 1.1
a =
     0
b =
                       1.1
>> bisect
>> xc
xc =
         0.292770767211914
>> abs(f(xc))
ans =
     0
>> a = 0, b = 1.2
a =
     0
b =
                       1.2
>> bisect
>> xc
xc =
         0.292767333984375
>> abs(f(xc))
ans =
     0
>> a = 0, b = 1.3
a =
     0
b =
                       1.3
>> bisect
>> xc
xc =
         0.292770767211914
>> abs(f(xc))
ans =
     0
>> a = 0, b = 1.4
a =
     0
b =
                       1.4
>> bisect
>> xc
xc =
         0.292770385742188
>> abs(f(xc))
ans =
     0
```

My results for four different intervals are like this, after being rounded:

[0, 1.1]: r = 0.2927707672

[0, 1.2]: r = 0.292767334

[0, 1.3]: r = 0.2927707672

[0, 1.4]: r = 0.2927703857

So my results for four different intervals do not agree with each other, up to 10 decimal places.

The backward error, or checking error, is |f(r)|. Calculate it for the approximate root(s) found in Step 1.

Using MatLab to find the backward errors for all approximate roots found in step 1. The procedure can also be found in the screenshot above. With four different approximate roots, without rounding up to 10 decimal points, backward errors all gave us |f(x)| = 0.

Now if I found backward errors for those approximate roots after rounding up to 10 digits, the backward errors |f(x)| would look like these:

```
>> abs(f(0.2927703857))

ans =

       2.8421709430404e-14

>> abs(f(0.292767334))

ans =

     0

>> abs(f(0.2927707672))

ans =

     0
```

With r = 0.2927703857, |f(x)| = 2.8421709430404e-14, which is also very small and very close to 0.

__**Answer:**__

I am **not** confident that I have all the correct first 10 digits of the root. Because with different intervals, I obtained different approximate roots (with different first 10 decimal points), and with those roots, backward errors all returned |f(x)| = 0. So, despite backward errors have zero as a result, I am not confident to find the root with all 10 correct first digits.

4. **Summarize any differences between the two cases, f and fnew(x). Do you believe that you have found the root to 10 correct places in both cases? If not 10, how many? Explain your reasoning**

Between two cases, f and $f_{new}(x)$, the difference is that for f, the approximate roots found after trying out two different intervals had the same first 10 digits after the decimal place. While for $f_{new}(x)$, the approximate roots found after trying all four different intervals were all different from each other, yet, they all return the backward errors $|f(x)| = 0$ (without rounding to 10), or very close to zero (with rounding up to 10).

I am confident that I have found the root up to 10 correct places in case of f but not in case of $f_{new}(x)$. In the case of $f_{new}(x)$, after observing I am confident that I have found the root at least to 4 correct places, which is **0.2927**.