Math 466/666 Programming Project 1

1. List the members on your team and explain how the work for this project was conducted. Please provide details concerning how many meetings were held, what was discussed at each meeting and what work was done between meetings. Further include a written statement attesting that the submitted report representsthe original efforts of the team members listed and, in particular, does contain the work of other students in the class.

You answer will be different than the made up one here.

This project was completed in four meetings by Student A, Student B and Student C. Setting up a discussion group on WebCampus helped schedule these meetings; however, email was used to send working copies of the project report back and forth between members of the team.

The initial meeting was held through Zoom. Student C had trouble connecting and then their computer crashed so the meeting was abruptly cancelled and rescheduled for the same time on the following week.

At the second meeting Test Student A wrote code to calculate the recurrence relation in question 2 using the terminal version of Julia as JupyterLAB was not installed. Code for this relation was communicated through the screen share and later emailed to each team member to work with. Student B agreed to watch the video on installing JupyterLAB. Student C volunteered to perform a web search to find additional information on the Foias constant.

At the third meeting most of the time was spent helping Student B install Julia on their computer using a screen share over Zoom. While trying to write a loop to determine whether the sequence was bounded, Anaconda started using all available computer cycles and then Zoom crashed. It was decided to set up the another meeting through WebCampus.

At the fourth meeting Student C presented two interesting links which discussed the Foias constant. A draft was started for the project report using a JupyterLAB notebook by Student C and sent through email to each member of the team.

At the fifth meeting, new code was written and improved by all team members for inclusion into the final project. Difficulties understanding the theoretical questions were discussed along with a plan for tacking those problems at the next meeting. Drafts of the new report were circulated through email and improved in preparation for the last meeting.

At the sixth and final meeting everyone in the team read the report that had been polished over email. One error was found and fixed. All team members then agreed that the report was finished and ready to turn in.

The report you are reading here was written represents the independent work of the team members listed below and, in particular, does not contain the work of any other student in the class.

2. Given $\alpha \in [1,2]$ consider the recurrence relation

$$x_{n+1} = \left(1 + \frac{1}{x_n}\right)^n$$

where $x_1 = \alpha$. Set $\alpha = 1$ and write a computer program in Julia that computes x_n for $n = 1, \ldots, 50$. Please include the full program listing and output. For reference,

$$x_{15} \approx 1.1.2634346914781789$$

and

$$x_{16} \approx 6283.87526674291.$$

In [1]:
```
alpha=1
```

Out[1]: 1

In [2]:
```
x=alpha
for n=1:49
    x=(1+1/x)^n
    println(n+1,"  ",x)
end
```

```
2  2.0
3  2.25
4  3.0137174211248285
5  3.1461328651361042
6  3.9749418398300125
7  3.8436459461081727
8  5.046648974589512
9  4.24710770652293
10  6.705641594809613
11  4.014991943105442
12  11.546255851436948
13  2.7094151558091024
14  59.372935430968965
15  1.2634346914781789
```

```
16  6283.87526674291
17  1.0025492407161776
18  128267.71346883844
19  1.0001403407948934
20  523589.5387780858
21  1.0000381985532598
22  2.0963110569382557e6
23  1.000010494678316
24  8.387595658020036e6
25  1.0000028613721854
26  3.355323187755784e7
27  1.0000007748883877
28  1.3421632395741531e8
29  1.00000020861846
30  5.3686928798555815e8
31  1.0000000558795226
32  2.1474817879952881e9
33  1.0000000149011754
34  8.589932479998269e9
35  1.000000003958121
36  3.435973598800008e10
37  1.000000001047738
38  1.3743895080800003e11
39  1.0000000002764864
40  5.49755810924e11
41  1.0000000000727596
42  2.199023252272e12
43  1.0000000000190994
44  8.796093018596e12
45  1.0000000000050022
46  3.5184372084872e13
47  1.0000000000013074
48  1.40737488351004e14
49  1.000000000000341
```

Note that the output agrees with the expected values for x_{15} and x_{16}.

3. Based on the numerical evidence in the previous step, make a conjecture regarding the value of the limits

$$\lim_{n\to\infty} x_{2n}$$

and

$$\lim_{n\to\infty} x_{2n+1}$$

when $x_1 = 1$. Explain your reasoning in as much mathematical detail as possible.

A visual inspection of the output from the program in question 2 indicates that the even terms growing without bound while the odd terms appear to decrease towards 1.

Note that if $x_{2n+1} = 1 + \epsilon$ for ϵ very small then

$$1 + \frac{1}{x_{2n+1}} = 1 + \frac{1}{1+\epsilon} = \frac{2+\epsilon}{1+\epsilon} \approx 2$$

Therefore

$$x_{2n+2} = \left(1 + \frac{1}{x_{2n+1}}\right)^{2n+1} \approx 2^{2n+1} \to \infty$$

as $n \to \infty$.

On the other hand if x_{2n} is very close to infinity than

$$1 + \frac{1}{x_{2n}} \approx 1$$

and so it's reasonable (but not guaranteed because 1^∞ is an indeterminate form) that

$$x_{2n+1} = \left(1 + \frac{1}{x_{2n}}\right)^{2n} \approx 1^{2n} = 1$$

as $n \to \infty$.

At anyrate, given the numerical evidence and the above heuristic argument, our conjecture is that

$$\lim_{n\to\infty} x_{2n} = \infty$$

and

$$\lim_{n\to\infty} x_{2n+1} = 1$$

when $x_1 = 1$.

4. Change the program in the previous step to compute the values of x_n when $\alpha = 2$. Look at the output and now make a conjecture regarding the value of the limits

$$\lim_{n\to\infty} x_{2n}$$

and

$$\lim_{n\to\infty} x_{2n+1}$$

when $x_1 = 2$. Explain the reasoning behind your conjecture.

In [3]:
```
alpha=2
```

Out[3]: 2

In [4]:
```
x=alpha
for n=1:49
    x=(1+1/x)^n
    println(n+1,"  ",x)
end
```

```
2  1.5
3  2.7777777777777772
4  2.515456000000001
5  3.814694524582706
6  3.202910914929112
7  5.105425602411104
8  3.4977433630005117
9  7.475734025618049
10  3.0953609877250656
11  16.436481006193045
12  1.9149171433392773
13  154.77995881694767
14  1.0873244478087112
15  9230.628993458884
16  1.001626257595803
17  64689.92026265085
18  1.0002628246012701
19  261524.77368382475
20  1.0000726533613016
21  1.0478144924485891e6
22  1.0000200419068173
23  4.193379435467222e6
24  1.0000054848507496
25  1.6776111794572083e7
26  1.0000014902152625
27  6.710756392755206e7
28  1.000000402339224
29  2.6843394397513205e8
30  1.000000108034036
31  1.0737400839916067e9
32  1.0000000288710484
33  4.2949653119971633e9
34  1.0000000076834112
35  1.7179866940000143e10
36  1.0000000020372681
37  6.8719474216000046e10
38  1.0000000005384209
39  2.74877904132e11
```

```
40  1.0000000001418812
41  1.099511624656e12
42  1.0000000000372893
43  4.39804650766e12
44  1.000000000009777
45  1.7592186040632e13
46  1.000000000002558
47  7.0368744173524e13
48  1.000000000000668
49  2.81474976706144e14
50  1.000000000000174
```

When $\alpha = 2$ the subsequence which remains bounded and the one which goes to infinity appears to change. Likely, since 2 is bigger, then starting x_1 with that larger number now makes the odd sequence x_{2n+1} be the one which tends to infinity while the evens sequence tends to 1. In particular, based on the numerical evidence, we conjecture that

$$\lim_{n\to\infty} x_{2n} = 1$$

and

$$\lim_{n\to\infty} x_{2n+1} = \infty.$$

5. [Extra Credit] Use rigorous mathematical analysis to prove the conjectures made in the previous two steps.

The theory in the reference

- J. Ewing, C Foias, An Interesting Serendipitous Real Number, *Finite versus Infinite: Contributions to an Eternal Dilema*, Springer-Verlag, pp. 119-120, 2000.

implies that limits conjectured in the previous questions are correct. Truthfully, more should be written for extra credit, for example quote the theorem from that book, to summarize the result and explain how the above conjectures follow from it. For this solution key, I have not gone into these necessary details.

More information may be found at

- *Foias Constant* Wikipedia, dowloaded October 17, 2020. https://en.wikipedia.org/wiki/Foias_constant.

6. Define

$$\alpha_* = \sup\Big\{\alpha : |x_{2n+1}| \text{ is bounded as } n \to \infty\Big\}.$$

Intuitively, α_* is the largest value of α such that $|x_{2n+1}|$ is bounded. Explain in details how the interval bisection method could be used to approximate α_*.

The interval bisection method is a natural search technique that can be used to find a critical value at which a function or system or changes state or behavior. A typical example of such a change is when a continuous function f changes from being positive to negative. Note to get started that it is imporant to have two reference values a and b such that $f(a)$ and $f(b)$ have opposite signs. In this case, finding the critical value at which the transition occurs is equivalent to finding a root x such that $f(x) = 0$.

In the present case the value $\alpha = 1$ leads to the odd terms x_{2n+1} being bounded while $\alpha = 2$ leads to that sequence not being bounded. Bisection can be used to find the critical value at which the transition from bounded to unbounded behavior occurs. Thus, we take $a = 1$ and $b = 1$ and note that $\alpha_* \in [a, b]$

To search for α_* we narrow down that value by bisecting the interval into the pieces $[a, c]$ and $[c, b]$ where c is given as the midpoint $c = (a + b)/2$. Now, if $\alpha = c$ results in bounded behavior of the sequence, then we know $\alpha_* \in [c, b]$. On the other hand, if $\alpha = c$ results in unbounded behavior of the sequence, then we know $\alpha_* \in [a, c]$.

In either case, the above bisection results in better knowledge of the true value of α_*. The bisection algorithm can then be used to divide subsequence intervals in half until a bound that determines α to the desired precision is obtained.

7. Write a computer program that bisects the interval $[1, 2]$ to find an approximation of α_* good to at least 4 significant digits. Include the full program listing and output.

First create a function that returns true if the resulting sequence is bounded and false if it isn't. This function assumes the behavior observed in the previous part of this assignment is typical and simply checks whether $|x_{51}|$ is greater than or less than 2 in order to determine whether the entire sequence x_{2n+1} is bounded or not.

In [5]:
```
function isoddbounded(alpha)
    x=alpha
    for n=1:50
        x=(1+1/x)^n
    end
    return abs(x)<2
end
```

Out[5]:
```
isoddbounded (generic function with 1 method)
```

Before continuing, we then check that our function with the known values $\alpha = 1$ and $\alpha = 2$ to make sure it is working.

In [6]:
```
isoddbounded(1)
```

Out[6]: true

In [7]:
```
isoddbounded(2)
```

Out[7]: false

Since $\alpha_* \in [1,2]$ we know the first digit of α is be one and the remaining digits will appear after the decimal point. Therefore, to obtain 4 significant digits we need to arrive at an interval $[a,b]$ such that $\alpha_* \in [a,b]$ and with a length $|b-a| < 0.0005$. Since the interval will be cut in half at each iteration, this implies at least n iterations where

$$\frac{1}{2^n} < 0.0005$$

will be sufficient. Taking logarithms as solving for n yields

$$n > \log(1/0.0005)/\log(2).$$

Since

In [8]:
```
log(1/0.0005)/log(2)
```

Out[8]: 10.965784284662087

we perform $n = 11$ bisection steps.

In [9]:
```
a=1
b=2
for n=1:11
    c=(a+b)/2
    if isoddbounded(c)
        a=c
    else
        b=c
    end
end
println("alpha_* in [",a,",",b,"]")
```

alpha_* in [1.18701171875,1.1875]

Therefore, the $\alpha_* \approx 1.187$ to 4 significant digits.

8. Explain theoretically how many times the interval [1,2] needs to be bisected to ensure the resulting approximation is good to at least 4 significant digits. What if 6 significant

digits are desired? How about 8 significant digits?

Most of the explanation is already contained in the answer to the previous question. In summary, at the beginning it is known that $\alpha_* \in [1, 2]$ which is an interval of length 1 and further implies the first digit of α_* is one.

In general, since the first digit is one and the rest are after the decimal point, to obtain k significant digits it is required that the length of the interval be less than 5×10^{-k}. Consequently, taking the number of interations n such that

$$\frac{1}{2^n} < 5 \times 10^{-k}$$

or equivalently so

$$n > \frac{\log(0.2 \times 10^k)}{\log 2} = 1 + (k-1)\frac{\log 10}{\log 2}.$$

In [10]:
```
Taking $k=4,6$ and $8$ yields
```

Out[10]:
```
$ (generic function with 1 method)
```

In [11]:
```
for k=4:2:8
    n=Integer(ceil(1+(k-1)*log(10)/log(2)));
    println("For $k significant digits use $n iterations")
end
```

```
For 4 significant digits use 11 iterations
For 6 significant digits use 18 iterations
For 8 significant digits use 25 iterations
```

9. Define

$$\alpha^* = \inf\Big\{\alpha : |x_{2n}| \text{ is bounded as } n \to \infty\Big\}.$$

Intuitively α^* is the smallest value of α such that $|x_{2n}|$ is bounded. Modify the program from the previous step to obtain an approximation of α^*. Include the output and describe what modifications were made to the code.

The main difference is that the function *isoddbounded* needs to be replaced by *isevenbounded* which stops at the 50th iteration rather than the 51st iteration.

In [12]:
```
function isevenbounded(alpha)
    x=alpha
    for n=1:49
        x=(1+1/x)^n
    end
    return abs(x)<2
end
```

Out[12]:
```
isevenbounded (generic function with 1 method)
```

Then since

In [13]:
```
isevenbounded(1)
```

Out[13]:
```
false
```

In [14]:
```
isevenbounded(2)
```

Out[14]:
```
true
```

One also needs to switch the logic in the if statement to reassign the upper index b of the interval when x_{2n} is bounded and lower index a when it's not.

In [15]:
```
a=1
b=2
for n=1:11
    c=(a+b)/2
    if isevenbounded(c)
        b=c
    else
        a=c
    end
end
println("alpha^* in [",a,",",b,"]")
```

```
alpha^* in [1.18701171875,1.1875]
```

10. Based on the numerical evidence obtained in the previous two problems conjecturewhether the values of α_* and α^* are equal or different.

After 11 iterations α_* are and α^* are still in the same interval. This numerical evidence suggests that $\alpha_* = \alpha^*$. For greater confirmation, we check after 25 iterations.

In [16]:
```
a=1
b=2
for n=1:25
    c=(a+b)/2
    if isoddbounded(c)
        a=c
    else
        b=c
    end
end
println("alpha_* in [",a,",",b,"]")
```

```
alpha_* in [1.187452346086502,1.1874523758888245]
```

In [17]:
```
a=1
b=2
for n=1:25
    c=(a+b)/2
    if isevenbounded(c)
        b=c
    else
        a=c
    end
end
println("alpha^* in [",a,",",b,"]")
```

```
alpha^* in [1.187452346086502,1.1874523758888245]
```

Even for this much smaller interval we numerically see that α_* and α^* are still in the same interval. This further suggests they are equal.

11. [Extra Credit] Use rigorous mathematical analyisis to prove the conjecture stated in the previous step. Alternatively, further support your conjecture by computing additional digits of α_* and α^* using the BigFloat arbitrary precision arithmetic built into the Julia programming language.

The reference cited in question 5 proves that $\alpha_* = \alpha^*$. For extra credit, an more explanation how the theory in those works implies $\alpha_* = \alpha^*$ and a brief description of the techniques used, if not a rephrasing of the full proof. An original proof of this result would be even better.

We finish with an example of using BigFloat arbitrary precision arithmetic to show that α_* and α^* agree to many many digits.

In [18]:
```
function bigoddbounded(alpha)
    x=alpha
    for n=1:250
        x=(1+1/x)^n
    end
    return abs(x)<2
end
function bigevenbounded(alpha)
    x=alpha
    for n=1:249
        x=(1+1/x)^n
    end
    return abs(x)<2
end;
```

In [19]:
```
a=big(1)
b=big(2)
for n=1:256
    c=(a+b)/2
    if bigoddbounded(c)
        a=c
    else
        b=c
    end
end
println("alpha_* in [\n",a,",\n",b,"]")
```

```
alpha_* in [
1.187452351126501054595480158396519351215692681585860353010104126198780
418723321,
1.187452351126501054595480158396519351215692681585860353010104126198780
418723338]
```

In [20]:
```
a=big(1)
b=big(2)
for n=1:256
    c=(a+b)/2
    if bigevenbounded(c)
        b=c
    else
        a=c
    end
end
println("alpha^* in [\n",a,",\n",b,"]")
```

```
alpha^* in [
1.187452351126501054595480158396519351215692681585860353010104126198780
418723321,
```

```
1.187452351126501054595480158396519351215692681585860353010104126198780
```

The intervals are the same after 256 interations. This implies that $\alpha_* = \alpha^*$ agree to within

In [21]:
```
b-a
```

Out[21]:
```
1.727233711018888925077270372560079914223200072887256277004740694033718
360632485e-77
```

or equivalently are the same to at least 77 decimal digits.

In [ ]:
```

```