Assignment 6: Fixed Point Iteration

By: Abrar Imon

Part 2: 4. First, I created a fixed iteration function. My function takes in as input: the function, p-nought, and the error tolerance. First, the inputted function will be applied to p-nought. Then, this fixedIteration function will repeatedly apply the inputted function to the output of its last iteration, until the difference between its last output and the Mathematica, estimated fixed point is less than the error tolerance. Then, it will return the last output, with 8 digit precision and the number of iteration that were performed.

```
fixedIteration[func_, initialP_, ϵ_] := (
  i = 1; lastP = initialP;
  While[i < 250, nextP = N[func[lastP], 32];
   If[Abs[nextP - N[FixedPoint[func, initialP, 100], 16]] < ϵ, Break[]];
   lastP = nextP;
   i++];
  Return[{N[nextP, 8], i}]
 )
```

```
(*This cell setups up the function from part 1: 2*)
g[x_] := Pi + (1 / 2) * Sin[x / 2]
```

```
In[ ]:= (*This cell uses my function to estimate
 the fixed point of g(x) with 0.1 error tolerance.*)
fixedIteration[g, 1, .01]
```

```
Out[ ]= {3.6262700, 3}
```

The estimate of the fixed point of g(x) rounds to 3.6262700 and it took 3 iterations to get to this.

Now, I will calculate the theoretical number of iterations required to achieve 10^-2 accuracy. I will use this formula from the corrolary: |Pn-P| <= (k^n)*|P1-P0|/(1-k). First we to find the lowest k we can. We will get that from finding the max of |g'(x)| on the interval [0, 2π].

```
In[ ]:= FindMaximum[{Abs[g'[x]], 0 ≤ x ≤ 2 Pi}, {x, 3}]
Out[ ]= {0.25, {x → 0.00245589}}
```

This maximum is 0.25. The value of k should be larger than that but we want it to be as low as possible. So theoretically, it would be something like 0.2500000...001. We can estimate this to 0.25, because after applying the formula, the 1 at the end of 0.2500000...001, will not make any significant difference. Applying this to the expression, we get: (((1/4)^n)/(3/4))*Abs[(g[1]-1)]. We want to find the lowest n, such that this expression will be less than 0.01

In[]:=
```
Table[N[(((1 / 4) ^n) / (3 / 4)) * Abs[(g[1] - 1)], 16], {n, 1, 10}]
```

Out[]=
```
{0.7937684742972982, 0.1984421185743246, 0.04961052964358114, 0.01240263241089529,
 0.003100658102723821, 0.0007751645256809553, 0.0001937911314202388,
 0.00004844778285505971, 0.00001211194571376493, 3.027986428441232 × 10^-6}
```

The first time, this expression is less than 0.01 is when n = 5. So, according to the corollary, 5 is the theoretical estimate of the number of iteration needed to estimate the fixed point of g(x) using p-nought = 1. In practice, it only took us 3 iterations, to get an estimate to within 0.01.

Part 2 : 5. To compare how fast the four sequences finds the value of cube root of 21 within 10^-2, I will create a table that displays the number of iterations that each of the functions take to make the calculation.

In[]:=
```
(*This sets up the functions for the 4 sequences in Part1: 3.*)
sequenceA[x_] := ((20 x + (21 / (x^2))) / 21);
sequenceB[x_] := (x - (((x^3) - 21) / 3 x^2));
sequenceC[x_] := (x - (((x^4) - 21 x) / ((x^2) - 21)));
sequenceD[x_] := ((21 / x) ^.5)
```

In[]:=
```
(*This function is the same as the one from the beginning of this assignment,
but this function compares its guess for a fixed point with 21^(1/2).*)
fixedIterationCount[func_, initialP_, ∈_] := (
  i = 1; lastP = initialP;
  While[i < 250, nextP = N[func[lastP], 32];
   If[Abs[nextP - (21^(1 / 3))] < ∈, Break[]];
   lastP = nextP;
   i++];
  Return[{i, N[nextP, 5]}]
 )

(*This cell creates a grid in which each
 of the rows represent one of the four sequences.*)
Grid[{Join[{"a"}, fixedIterationCount[sequenceA, 1, 10^-2]],
  Join[{"b"}, fixedIterationCount[sequenceB, 1, 10^-2]],
  Join[{"c"}, fixedIterationCount[sequenceC, 1, 10^-2]],
  Join[{"d"}, fixedIterationCount[sequenceD, 1, 10^-2]]}, Frame → All]
```

```
(*This cell titles the columns of the grid.*)
ReplacePart[
 Grid[{{"a", 28, 2.7502}, {"b", 250, Overflow[]}, {"c", 250, 0}, {"d", 9, 2.7644}},
  Frame → All], 1 → Prepend[First[Grid[{{"a", 28, 2.7502},
      {"b", 250, Overflow[]}, {"c", 250, 0}, {"d", 9, 2.7644}}, Frame → All]],
   {"Sequence", "Number of iterations", "Estimate for 21^(1/3)"}]]
```

Out[ ]=

Sequence	Number of iterations	Estimate for 21^(1/3)
a	28	2.7502
b	250	Overflow[]
c	250	0
d	9	2.7644

The sequence from (d) seems to be the fastest here as it took the least number of iterations to give us an estimate. The sequence from (a) is the second best as it took 28 iterations to give us an estimate. According to my answer from Part1: 3, the sequences from (a) and (b) are the only two that can have a k-value lower than 1; so, it is fitting that these are the two sequences that give us an estimate. The other two reached the maximum number of iterations and still failed to give us an estimate of 21^(1/3) within 10^(-2). That is consistent with my analysis in Part1: 3, because the sequences from (c) and (d) have a k-value higher than 1.