Supplement to Problem Set 1 Solution

This Mathematica notebook picks up where the handwritten solution to 6(b) stops. It also has an analysis of 6(c). For the remaining problem on Eratosthenes (Van Brummelen Chapter 1, Problem 10) we will discuss the solution in person.

Problem 6(b) (Cont'd)

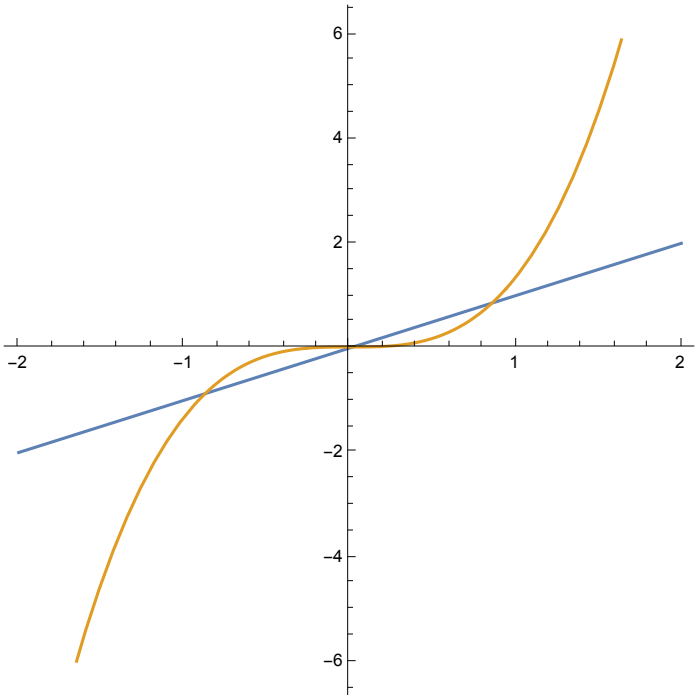
Solve $x = \frac{\epsilon + 4x^3}{3}$ iteratively. First, to understand what we are solving, graph the LHS and RHS as if they were separate functions:

```
In[•]:= ϵ = N[Sin[3 Pi / 180], 100]
Out[•]= 0.052335956242943832722118629609078418731018253940164920483509381599857104641754546864644659881188693981
```

```
In[•]:= f1[x_] := x
        f2[x_] := (ϵ + 4 x^3) / 3
```

```
In[•]:= Plot[{f1[x], f2[x]}, {x, -2, 2}, AspectRatio → 1]
```

Out[•]=



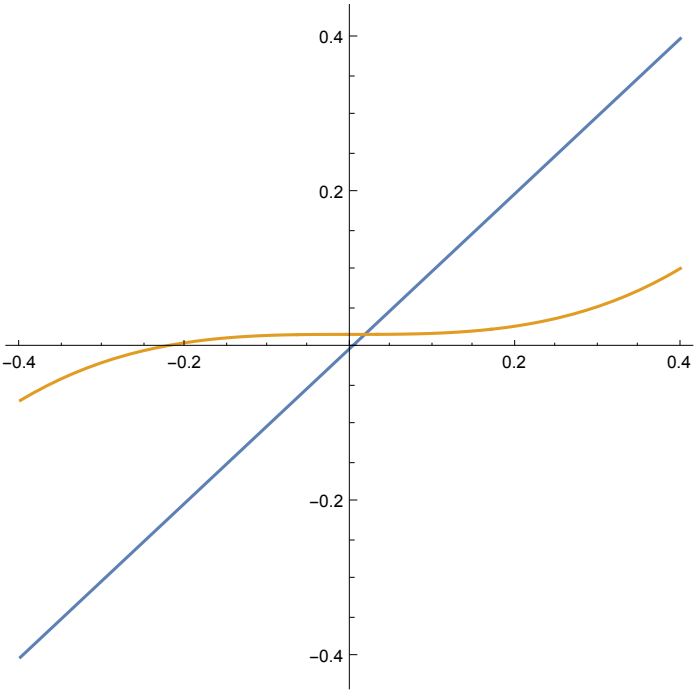
Something to note: The solutions of the equation are the three places where the two graphs cross each other. However we have no interest in the solutions near 1 and -1. The sin 1° is a very small number. We

are only interested in the crossing point that is very near the origin.

Let's blow that region up by 5x:

In[•]:=
```
Plot[{f1[x], f2[x]}, {x, -0.4, 0.4}, AspectRatio → 1]
```

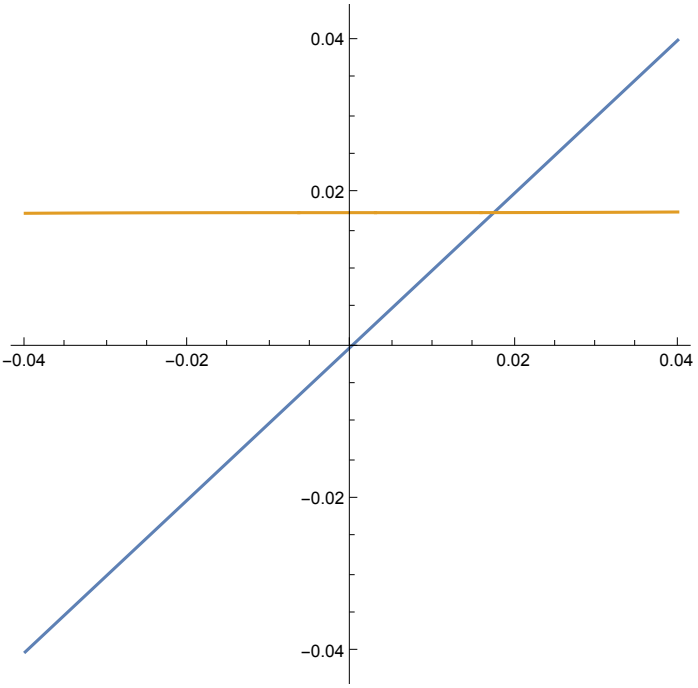
Out[•]=



Hmmm. Still hard to see what is going on. Blow up another 10x:

In[•]:=
```
Plot[{f1[x], f2[x]}, {x, -0.04, 0.04}, AspectRatio → 1]
```

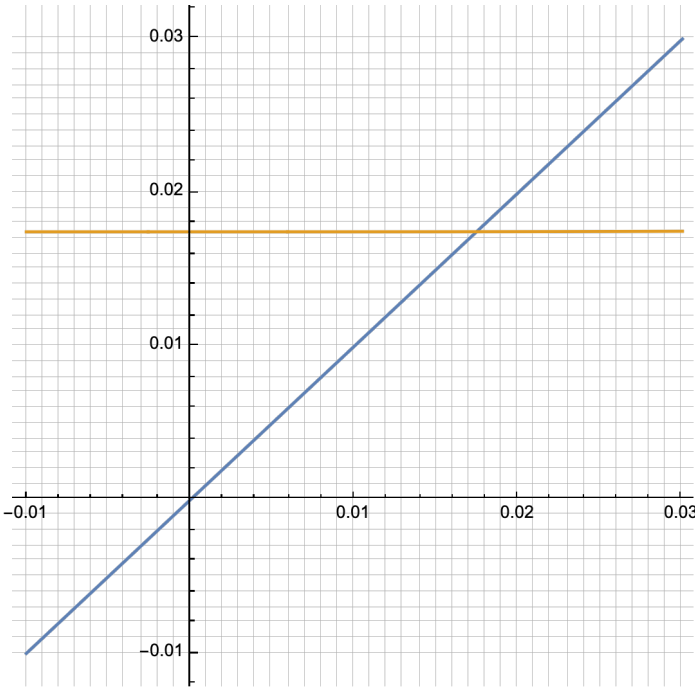
Out[•]=



Blow up another 2x, pan right, and add grid lines:

In[•]:=
```
Plot[{f1[x], f2[x]}, {x, -0.01, 0.03}, AspectRatio → 1,
 GridLines → {Range[-0.01, 0.03, 0.001], Range[-0.01, 0.03, 0.001]}]
```

Out[•]=



It looks like $x = \sin 1^\circ$ is going to be between 0.017 and 0.018. Now that we have some clue about what we are doing, our job is to get that to 100 decimal places using al-Kāshī's method. First let's cheat and have Mathematica just tell us the answer to 100 decimal places:

In[•]:=
```
x = N[Sin[Pi / 180], 100]
```

Out[•]=
```
0.017452406437283512819418978516316192472252720307139642683612427640597384203928070042001926791021346 91
```

Now let's use al-Kāshī's method:

In[•]:=
```
x0 = f2[0]
```

Out[•]=
```
0.017445318747647944240706209869692806243672751313388306827836460533285701547251515621488199603963 21327
```

In[•]:=
```
x1 = f2[x0]
```

Out[•]=
```
0.017452397805531902625351278627224514439711700322460331165686914838740177110456948476509161625676 04328
```

In[•]:=
```
x2 = f2[x1]
```

Out[•]=
```
0.017452406426767058303119085898706456328207791715605185630030536480803317457492486274660509644402 02227
```

In[•]:=
```
x3 = f2[x2]
```

Out[•]=
```
0.01745240643727070013953422201197194756185931160709925151045670781262412604361991544200514130967017314
```

In[•]:=
```
x4 = f2[x3]
```

Out[•]=
```
0.01745240643728349720914218098917990301561679601051265283680936711512409390697149340023072872558023132
```

Dang, we've computed x0, x1, x2, x3, and x4, and only gotten 15 decimal places of agreement (compare 0.01745240643728351 with 0.01745240643728349). So let's automate this:

In[•]:=
```
frecursive[i_] := If[i == 0, f2[0], f2[frecursive[i - 1]]]
```

In[•]:=
```
Table[{i, frecursive[i]}, {i, 0, 40}]
```

Out[•]=
```
{{0, 0.01744531874764794424070620986969280624367275131338830682783646053328570154725151562148819960396231327},
 {1, 0.01745239780553190262535127862722451443971170032246033116568691483874017711045694847650916162567604328},
 {2, 0.01745240642676705830311908589870645632820779171560518563003053648080331745749248627466050964440202227},
 {3, 0.01745240643727070013953422201197194756185931160709925151045670781262412604361991544200514130967017314},
 {4, 0.01745240643728349720914218098917990301561679601051265283680936711512409390697149340023072872558023132},
 {5, 0.01745240643728351280040026081733547519040914480592787447287371941167840343241397003182331625605940495},
 {6, 0.01745240643728351281939580713840886270778990481949961739510125672906657895827128446065115934838462880},
 {7, 0.01745240643728351281941895028556148953741628813784690718740618818509518315582086376324892073565389004},
 {8, 0.01745240643728351281941897848192136648112461343881370616412503831077342657306833930767179285904738937},
 {9, 0.01745240643728351281941897851627428767489932196412565876837166130395209574078828655509405445151009180},
 {10, 0.01745240643728351281941897851631614141771208438907447864722617263784154646901169075532259989750779332},
 {11, 0.01745240643728351281941897851631619241005062689137885881582062844352994725440230740142062041088111482},
 {12, 0.01745240643728351281941897851631619247217693663781051295901124040156471619545383533356099602366495685},
 {13, 0.01745240643728351281941897851631619247225262797641214450470576166883290583856406733973529637029449159},
 {14, 0.01745240643728351281941897851631619247225272019464887368536143659025985831059493417354322479472768397},
 {15, 0.01745240643728351281941897851631619247225272030700259000946267124462577121873766610729995912261858945},
 {16, 0.01745240643728351281941897851631619247225272030713947570604032222321577258884833782095356028502689676},
 {17, 0.01745240643728351281941897851631619247225272030713964248017597695336780703313591747763401666974899202},
 {18, 0.01745240643728351281941897851631619247225272030713964268336457166241793586900393157866022644220888494},
 {19, 0.01745240643728351281941897851631619247225272030713964268361212566626725916744650477938557611939245665},
 {20, 0.01745240643728351281941897851631619247225272030713964268361242727268817852625092664073351333426931505},
 {21, 0.01745240643728351281941897851631619247225272030713964268361242764014914350887848452191238267420072332},
 {22, 0.01745240643728351281941897851631619247225272030713964268361242764059683809168733812626282334652145716},
 {23, 0.01745240643728351281941897851631619247225272030713964268361242764059738353857442685205911795986489212},
 {24, 0.01745240643728351281941897851631619247225272030713964268361242764059738420311743911764685978770430712},
 {25, 0.01745240643728351281941897851631619247225272030713964268361242764059738420392708241308872167109051543},
 {26, 0.01745240643728351281941897851631619247225272030713964268361242764059738420392806883872822862220644551},
 {27, 0.01745240643728351281941897851631619247225272030713964268361242764059738420392807004053592313990694328},
 {28, 0.01745240643728351281941897851631619247225272030713964268361242764059738420392807004200014069139301511},
 {29, 0.01745240643728351281941897851631619247225272030713964268361242764059738420392807004200192461493407735},
 {30, 0.01745240643728351281941897851631619247225272030713964268361242764059738420392807004200192678837011978},
 {31, 0.01745240643728351281941897851631619247225272030713964268361242764059738420392807004200192679101811680},
 {32, 0.01745240643728351281941897851631619247225272030713964268361242764059738420392807004200192679102134298},
 {33, 0.01745240643728351281941897851631619247225272030713964268361242764059738420392807004200192679102134691},
 {34, 0.01745240643728351281941897851631619247225272030713964268361242764059738420392807004200192679102134691},
 {35, 0.01745240643728351281941897851631619247225272030713964268361242764059738420392807004200192679102134691},
 {36, 0.01745240643728351281941897851631619247225272030713964268361242764059738420392807004200192679102134691},
 {37, 0.01745240643728351281941897851631619247225272030713964268361242764059738420392807004200192679102134691},
 {38, 0.01745240643728351281941897851631619247225272030713964268361242764059738420392807004200192679102134691},
 {39, 0.01745240643728351281941897851631619247225272030713964268361242764059738420392807004200192679102134691},
 {40, 0.01745240643728351281941897851631619247225272030713964268361242764059738420392807004200192679102134691}}
```

From x33 onward, we have converged to the 100-decimal place value.

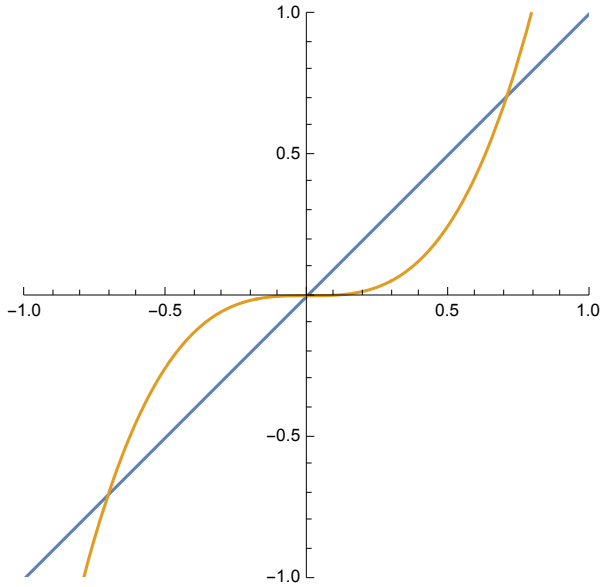
Problem 6(c)

Try to solve $x = 2x^3$ iteratively. As in 6(b) we first graph the LHS and RHS:

In[•]:=
```
g1[x_] := x
g2[x_] := 2 x^3
```

In[•]:=
```
Plot[{g1[x], g2[x]}, {x, -2, 2}, PlotRange → {{-1, 1}, {-1, 1}}, AspectRatio → 1]
```

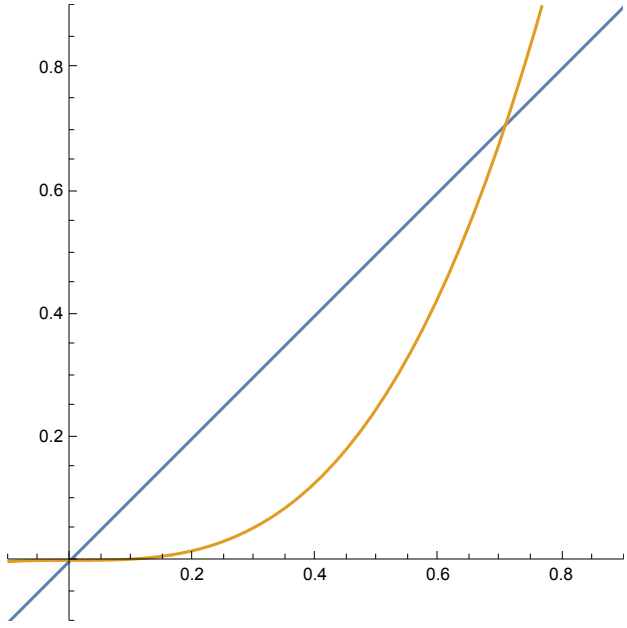
Out[•]=



We are interested in that solution near $x = 0.7$. Blow up by 4x and pan right:

In[•]:=
```
Plot[{g1[x], g2[x]}, {x, -0.1, 0.9},
 PlotRange → {{-0.1, 0.9}, {-0.1, 0.9}}, AspectRatio → 1]
```

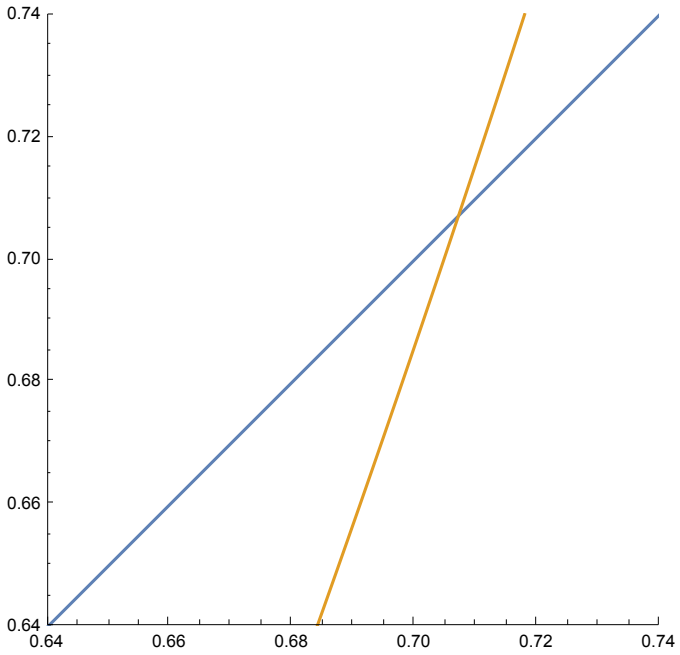
Out[•]=



Blow up another 10x and pan right again:

```
In[•]:= Plot[{g1[x], g2[x]}, {x, 0.64, 0.74},
 PlotRange → {{0.64, 0.74}, {0.64, 0.74}}, AspectRatio → 1]
```

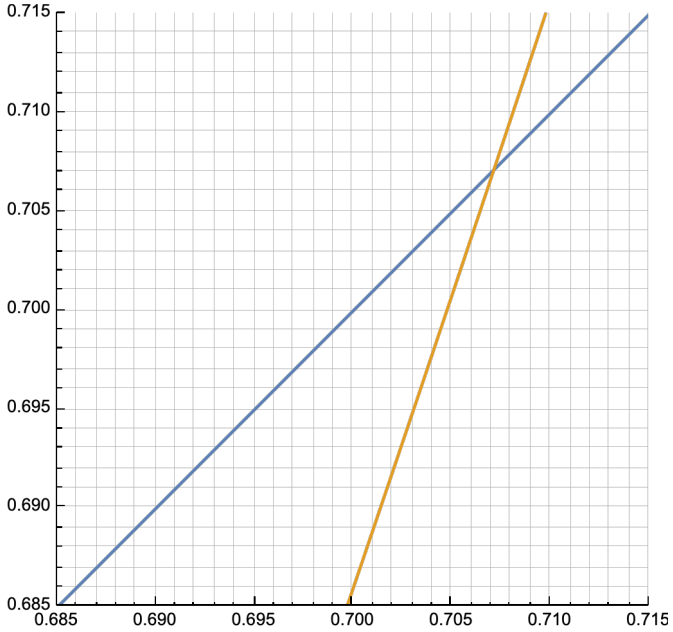
Out[•]=



Blow up another 4x, pan right yet again, and add grid lines:

```
In[•]:= Plot[{g1[x], g2[x]}, {x, 0.685, 0.715},
 PlotRange → {{0.685, 0.715}, {0.685, 0.715}}, AspectRatio → 1,
 GridLines → {Range[0.685, 0.715, 0.001], Range[0.685, 0.715, 0.001]}]
```

Out[•]=



The lines cross at between 0.707 and 0.708, which is of course consistent with the exact answer which is $\sqrt{2}/2$.

So now that we have a very good idea about the crossing point we are looking for we are try the recursive method.

```
In[•]:= grecursive[i_] := If[i == 0, 0.707, g2[grecursive[i - 1]]]

        Table[{i, grecursive[i]}, {i, 0, 13}]

Out[•]= {{0, 0.707}, {1, 0.706786}, {2, 0.706146}, {3, 0.704229}, {4, 0.69851},
          {5, 0.681627}, {6, 0.63339}, {7, 0.50821}, {8, 0.262518}, {9, 0.0361834},
          {10, 0.0000947452}, {11, 1.70099 × 10^-12}, {12, 9.84318 × 10^-36}, {13, 1.90738 × 10^-105}}
```

Amusingly, we are converging to the 0 solution, which is valid, but is not anywhere near we started.

What if we start a little above the correct answer:

```
In[•]:= grecursive[i_] := If[i == 0, 0.708, g2[grecursive[i - 1]]]

In[•]:= Table[{i, grecursive[i]}, {i, 0, 13}]

Out[•]= {{0, 0.708}, {1, 0.70979}, {2, 0.715186}, {3, 0.731624}, {4, 0.783238}, {5, 0.960973},
          {6, 1.77486}, {7, 11.182}, {8, 2796.34}, {9, 4.37319 × 10^10}, {10, 1.67273 × 10^32},
          {11, 9.36065 × 10^96}, {12, 1.64039 × 10^291}, {13, 8.828221488230684 × 10^873}}
```

We are headed off to infinity.

The fundamental reason that the iterative solution to 6(c) does not work is that the slope of g2[*x*] is too steep. In problem 6(b), the method worked because the slope of f2[x] was pleasantly shallow. This is some deep stuff. Suffice it to say that iterative methods sometimes converge, and sometimes they don't, and so that you will be aware of that, and not over-confident with iterative methods, Van Brummelen created 6(c).