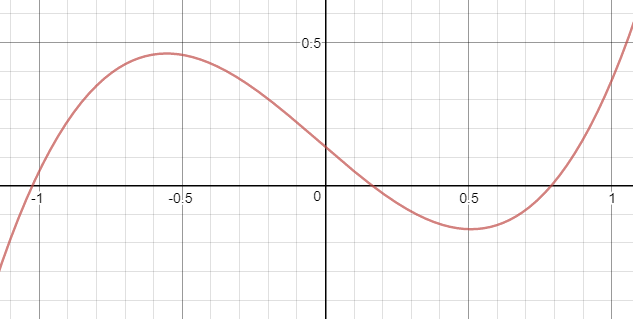
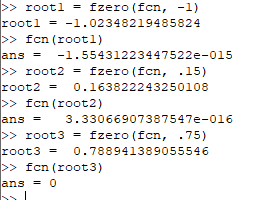
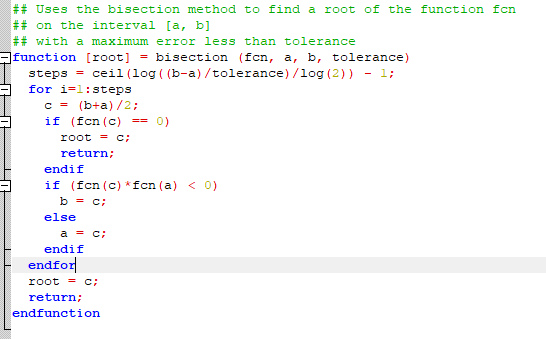
1.1.C3b: e^(x-2) + x^3 – x



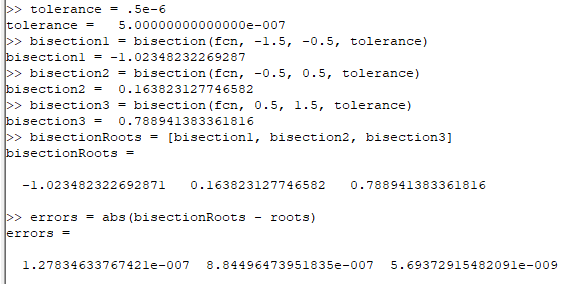
Intervals with roots: [-1.5, -0.5], [-0.5, 0.5], [0.5, 1.5]

Using fzero to find the roots, then testing each one:



My bisection method function. Yeah, I called it that just to write ‘method function’.

Calculated roots compared with the fzero roots:



Assuming all the fzero roots are good (and we checked them out above: plugged into g(x), they gave us values within 10^-14 of zero), then our errors are all accurate to within six decimal places as desired.

1.1.C9:

v(h) = pi\*h^2(r – h/3) = 1 = pi\*h^2\*r – pi/3\*h^3 = 1

Since r = 1, then v(h) = pi\*h^2 – pi/3\*h^3 = 1

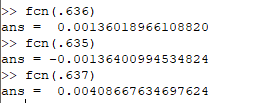
Converting to a root problem: pi\*h^2 – pi/3\*h^3 – 1 = 0

Since we’re working in meters, we need our answer to be within .001m. which means we want our tolerance to be .0005m



Thus the height should be .636±.001m

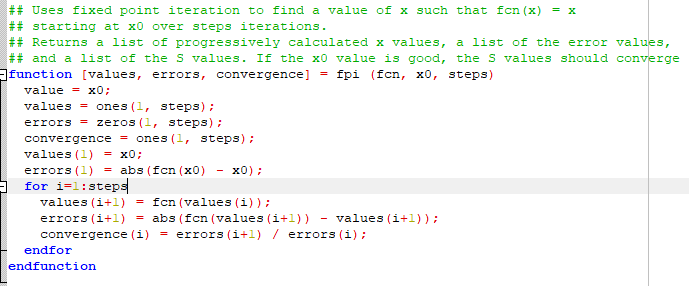
Checking with Octave:

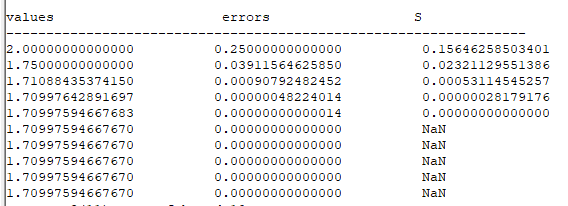


Seems pretty reasonable!

1.1.C4c: g(x) = (2x + A/x^2)/3 where A =5

My function for Fixed Point Iteration:

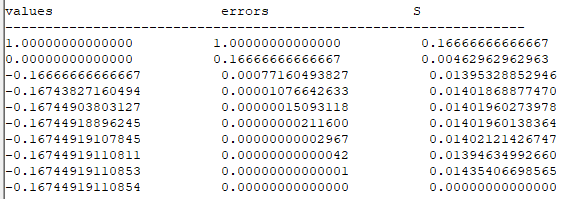
The results:



So even with a terrible beginning guess of 2, FPI only took 5 steps (counting the initial x0 step) to attain 8 decimal places of accuracy.

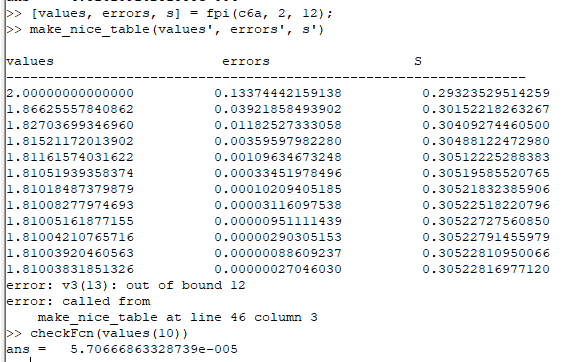
1.1.C6a: f(x) = 2x^3 - 6x – 1 = 0

i. 2x^3 – 1 = 6x → (x^3 – 1)/6 = x → g(x) = (x^3 – 1)/6

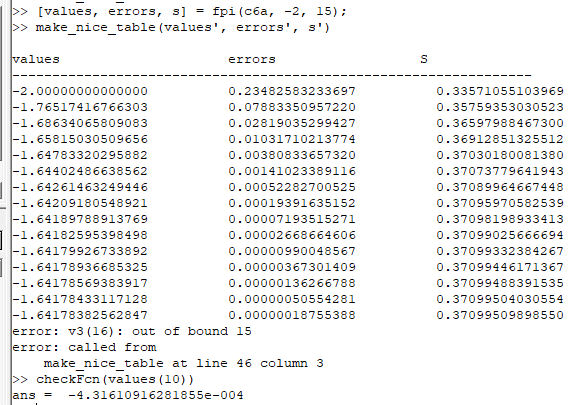




ii. 2x^3 = 6x + 1 → x = cbrt((6x+1)/2)



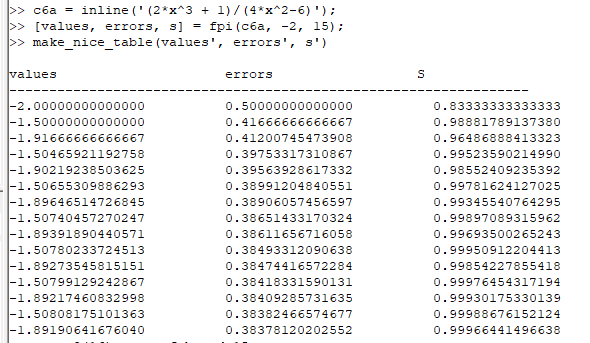
We can get the third root using this same equation with a different seed and some extra iterations:



iii. Adding 2x^3 to both sides:

2x^3 + 2x^3 - 6x – 1 = 2x^3 → 4x^3 – 6x = 2x^3 + 1 → x(4x^2 – 6) = 2x^3 + 1

→ g(x) = (2x^3 + 1)/(4x^2 – 6)

Unsurprisingly, this function converges quite poorly, since I made a WAG as to the additional value.

That’s fine, we got three good results from the first two functions.

Comparison of S with predicted S:

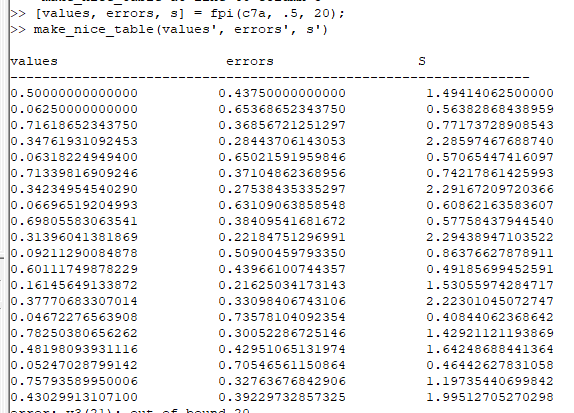
i. g = (x^3 – 1)/6 so g’ = (x^2)/2 so g’(-.16744918896245) = .01402, which matches the S generated by our FPI.

ii. g = cbrt((6x+1)/2) so g’ = ((6x+1)/2)^(-2/3) = cbrt(4) / cbrt((6x+1)^2) so g’(1.81003831851326) = 0.305228 which matches the S generated by our FPI.

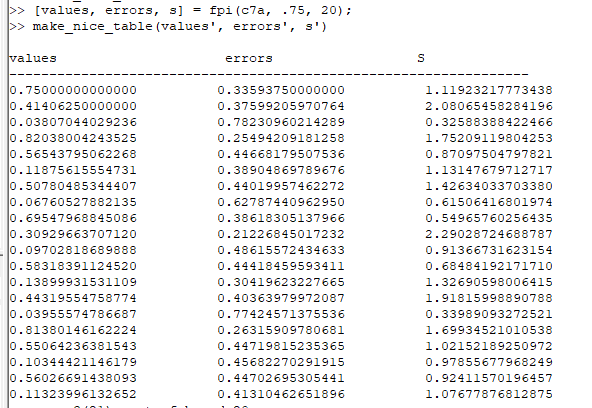
iii. g’ the same as in (ii), so g’(-1.64178382562847) = .370995, which again matches the S generated by our FPI.

1.2.C7a: g(x) = 1 – 5x + 15/2 x^2 – 5/2 x^3 cycling within (0, 1).

a starting point of ½ seems to cycle endlessly within (0, 1):



a starting point of ¾ does the same:

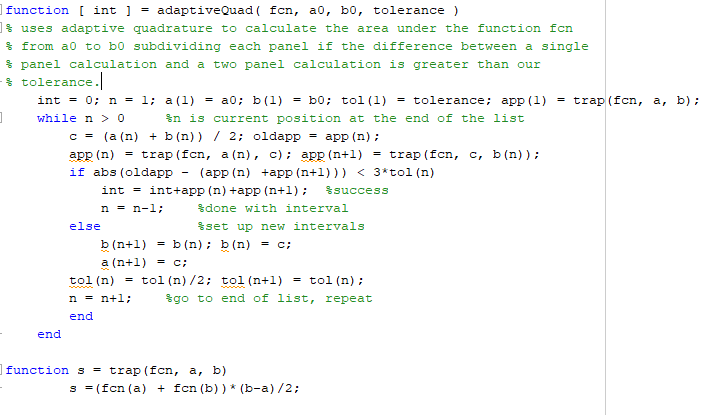


Obviously, a starting point of 0 or 1 converges instantly.

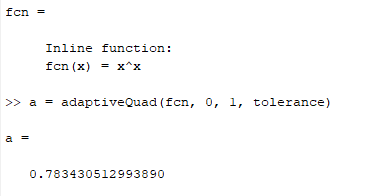
A starting point of 5/4 breaks out of the interval, and it becomes obvious that for x > 1, the higher order terms dominate and g(x) > 1. Via trial and error it appears that as long as our initial guess is between 0 and 1, g(x) cycles within (0, 1).

5.4.C6g: integrate x^x dx from 0 to 1

Adaptive quadrature method, straight from the book:



Results of using it on the integral of x^x dx on [0, 1]:



Compute the area under f(x) from x=0 to x=1 where

f(x) = 1/((x-0.3)^2+0.01) + 1/((x-0.9)^2 + 0.04) - 6

using adaptive quadrature with an error tolerance of 0.5\*10^(-5).  Make a plot of the curve along with the subintervals so we can see that more are used where the function is changing quickly.

With over 8000 iterations, we cannot display all the subintervals. Instead I opted to display the midpoint of every 50th subinterval. It is still somewhat difficult to see in the picture, but the x marks on the x axis indicate a midpoint and they do seem to cluster more thickly the steeper the slope of the plot.

