For very small numbers the `good-enough?` test will become less effective because the test of a 0.001 absolute difference is fixed for all radicands. A 0.001 difference is small enough for larger radicands, but as radicands becomes smaller, a relatively large difference (compared to the radicand) becomes acceptable. For example, the square root of 0.00001 is approximately 0.003, but when `good-enough?` encounters roughly 0.03, it will test the absolute difference between 0.03^2, which is 0.0009, and the radicand 0.00001. This difference is 0.00089, which is less than 0.001 but results in a square root that is off by approximately 0.028, a value that is almost 3,000 times larger than the original radicand.

As seen by the following table, as the radicand gets smaller, the ratio between the difference in scheme-calculated square root and radicand grows larger and larger.

Radicand	**sqrt**	**True Value**	**Difference**	**Ratio**
0.1	0.31624556228039	0.31622776601684	0.00001779626355	0.00017796263551
0.01	0.10032578510961	0.1	0.00032578510961	0.0325785109606
0.001	0.04124542607499	0.03162277660168	0.00962264947331	9.62264947330736
0.0001	0.03230844833048	0.01	0.02230844833048	223.0844833048122
0.00001	0.03135649010772	0.00316227766017	0.02819421244755	2819.421244754878
0.000001	0.03126065552545	0.001	0.03026065552545	30260.655525445272

For large numbers, this same pattern does not apply. The `good-enough?` test works well for large-ish numbers, however, once 10^{13} is reached, there is probably not enough precision for the computer to store the digits required to pass the test. The interpreter is unable to return a value for those larger radicands.

Radicand	**sqrt**	**True Value**	**Difference**
100	10.0000000001399	10	0.0000000001399
1000	31.62278245070105	31.6227766016838	0.00000584901725
10000	100.00000025490743	100	0.00000025490743
100000	316.2277660203896	316.22776601683796	0.00000000355163
1000000	1000.0000000000118	1000	0.00000000001182
10000000	3162.277660168379	3162.2776601683795	0.00000000000045
100000000	10000	10000	0
1000000000	31622.776601684043	31622.776601683792	0.00000000025466
10000000000	100000	100000	0
100000000000	316227.7660168379	316227.7660168379	0
1000000000000	1000000	1000000	0
10000000000000	0	3162277.6601683795	3162277.6601683795
100000000000000	0	10000000	10000000
1000000000000000	0	31622776.601683795	31622776.601683795

By modifying the `good-enough?` test to instead check whether the absolute difference between guesses is less than 0.001 times the size of the guess, we can solve both problems. As seen in the table below, the ratio of the difference is now acceptable for very small numbers, and the interpreter is also able to give very accurate square roots for very large numbers.

Radicand	**sqrt**	**True Value**	**Difference**	**Ratio**

0.1	0.31622776651757	0.31622776601684	0.00000000050073	0.0000000050073
0.01	0.1000000000014	0.1	0.0000000000014	0.0000000001399
0.001	0.0316227824507	0.03162277660168	0.00000000584902	0.00000584901726
0.0001	0.01000000002549	0.01	0.00000000002549	0.00000025490743
0.00001	0.0031622776602	0.00316227766017	0.00000000000004	0.00000000355163
0.000001	0.0010000001533	0.001	0.0000000001533	0.00015330166281

For the very large numbers ($> 10^{13}$) that the old test didn't work for, the interpreter can now return a value when the guess stabilizes, even if it isn't within 0.001, as long as the difference is a low enough ratio (0.001, which for numbers this large is permissible even with limited precision).

Radicand	**sqrt**	**True Value**	**Difference**
100	10.0000000001399	10	0.0000000001399
1000	31.62278245070105	31.6227766016838	0.00000584901725
10000	100.00000025490743	100	0.00000025490743
100000	316.2277660203896	316.22776601683796	0.00000000355163
1000000	1000.0001533016629	1000	0.00015330166286
10000000	3162.277666486375	3162.2776601683795	0.00000631799549
100000000	10000.000000082462	10000	0.00000008246207
1000000000	31622.780588899368	31622.776601683792	0.00398721557576
10000000000	100000.00015603233	100000	0.00015603232896
100000000000	316227.7660187454	316227.7660168379	0.00000190746505
1000000000000	1000000.1034612419	1000000	0.10346124204807
10000000000000	3162277.6640104805	3162277.6601683795	0.00384210096672
100000000000000	10000000.000043957	10000000	0.00004395656288
1000000000000000	31622779.279995147	31622776.601683795	2.6783113591373