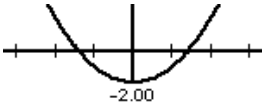
College Bound Math Solutions #18
week of March 9, 2015



Note: Students who have not done Set #17 should do that before Set #18. These problems are related to each other, to computer science, and to the square root of 2. Problem #2 is an *algorithm* (a series of steps) that is just as simple as the one in Set #17.

This Babylonian method was known over 3,000 years ago. It can be used on other square roots. It's accuracy is *incredible* (see #3 below). The function graphed above is used in the Newton-Raphson method, which has a wide range of applications, but operates like the Babylonian when finding $\sqrt{2}$.

1. These will be more valuable if done by hand, with fractions, not decimals.

(a) Divide each number into 2: $\frac{2}{\frac{3}{2}} \times \frac{2}{2} = \frac{4}{3}$ $\frac{2}{\frac{17}{12}} \times \frac{12}{12} = \frac{24}{17}$

(b) Find the average $\frac{\frac{3}{2}+\frac{4}{3}}{2} = \frac{\frac{9}{6}+\frac{8}{6}}{2} = \frac{\frac{17}{6}}{2} = \frac{17}{12}$

2. Finding $\sqrt{2}$ with paper and pencil. 1 is too small and 2 is too big (for their squares to equal 2), so we start by guessing $1\frac{1}{2}$, since it's midway between them. (It's square is 2.25, which is pretty close to 2.) Notice that by dividing $\frac{3}{2}$ into 2 we get a number, $\frac{4}{3}$, that's roughly an equal distance from $\sqrt{2}$ but on the *other side* of it, so that their average is a much better approximation than either of them!

Row	Steps	1st time	2nd time	3rd time
1	Guess (first time only). Then use the Row #3 result from preceding column.	3/2	17/12	577/408
2	Divide the Row #1 result into 2.	4/3	24/17	816/577
3	Average Row #1 and Row #2	17/12	577/408	below
4	Copy the result to the top of the next column.			
5	Do it all again or stop.			

3. Using a calculator let's see how close these results are to the actual value of $\sqrt{2}$.

(a) To 9 decimal places, $\sqrt{2}$ = 1.414213562

(b) 3/2=1.50000000 Row #3: 1.41666667, 1.414215686, 1.41421356237

(c) Subtracting $\sqrt{2}$: 0.08578... 0.00245... 0.000002124... 0.00000000000

(d) Just three columns gets the answer to the nearest *hundred billionth*!!