Recursion, Roots and Newton's Methods

Successive Approximations to Find Square Roots — The Babylonian Algorithm.
Suppose you don't have a square root key on your calculator. How can we obtain $\sqrt{2}$? (The algorithm given here is quite ancient, hence the name "Babylonian Algorithm.")
Suppose

$$\begin{aligned} a &= \sqrt{2}. \text{ Square both sides.} \\ a^2 &= 2 \text{ Divide both sides by } a. \\ a &= 2/a \end{aligned}$$

Only the true square root of 2 satisfies $\sqrt{2} = 2/\sqrt{2}$.

If x is an estimate which is less than the true value for $\sqrt{2}$ then $2/x$ is an estimate which is ____________________ than the true value.

If x is an estimate which is greater than the true value for $\sqrt{2}$ then $2/x$ is an estimate which is ____________________ than the true value.

Hence a better estimate will be ________________________________.

General Babylonian Algorithm For Approximating $\sqrt{r}$.

STEP 1: Let x_0 be your initial estimate for $\sqrt{r}$.

STEP 2: Then the next estimate is the average of your most recent estimate and r divided by your most recent estimate.

$$x_{\text{new}} = \frac{x_{\text{old}} + \dfrac{r}{x_{\text{old}}}}{2}$$

STEP 3: Continue calculating terms in the sequence until you reach the level of accuracy desired.

Implementing the Babylonian Algorithm on a TI-83 Calculator

The Babylonian Algorithm to approximate $\sqrt{A}$ is

$$x_{\text{new}} = \frac{x_{\text{old}} + \dfrac{A}{x_{\text{old}}}}{2}$$

- Let's store the value of A 2 STO→ ALPHA A
- Let's store X_0 to be 1   1 STO→ X
- Define the recursive step   (X + ALPHA A ÷ X) * 0.5 STO→ X

Press ENTER repeatedly... Write down the results below. (What happens??)

This method is known as a **recursive algorithm** because your next estimate x_{n+1} is produced using information from your current estimate x_n.

The sequence of numbers $x_1, x_2, x_3, \ldots,$ hopefully converges to a finte number L, the limit of the sequence.

Newton's Method

Newton's Method is a Calculus-based method for approximating roots. In fact, it derives from the Microscope Approximation,

$$\Delta y \approx g'(a)\Delta x.$$

To use this method to approximate a root r for a function $g(x)$ you need several things:

i) an initial guess x_0 *sufficiently close* to the root r,

ii) $g'(r) \neq 0$ on an interval $a < x < b$ containing x_0 and r,

ii) g, g', and g'' continuous on an interval $a < x < b$ containing x_0 and r.

Provided these conditions are satisfied, Newton's Method is *guaranteed* **to converge to the root** r**.**

The big question, however, is "HOW CLOSE MUST x_0 BE TO r?" While certain formulas involving the second derivative can be given to address this question, in practice you simply run Newton's Method for a number of iterations to determine whether it is converging to the root you want. If you find it is not doing so, pick another value for x_0. In lab this week you saw examples of how Newton's Method behaves when x_0 is sufficiently close to r, as well as what can happen when x_0 is to far from r.

Deriving Newton's Method from the Microscope Approximation

Suppose we have obtained our nth approximation x_n for the root r of g, and we want to find a better approximation x_{n+1}. Ideally, we would like to know

$$\Delta x = r - x_n.$$

If we knew this exactly, then we could find the root r as $r = x_n + \Delta x$.

Since we don't know Δx exactly, we will appeal to the Microscope Approximation based at our current guess x_n:

$$\Delta y \approx g'(x_n)\Delta x \quad \Longrightarrow \quad \Delta x \approx \frac{\Delta y}{g'(x_n)}.$$

This approximation is valid provided $g'(x_n) \neq 0$. But this will be true, provided x_n is sufficiently close to r, because we know that $g'(r) \neq 0$ and that g' is continuous on an open interval $a < r < b$ containing r.

Although we don't know Δx exactly, we do know Δy exactly! This is because we know $g(x_n)$ and we also know that $g(r) = 0$, so

$$\Delta y = \underline{\hspace{6cm}}.$$

Therefore,

$$\Delta x \approx \underline{\hspace{6cm}}$$

and we choose our next approximation x_{n+1} as

$$x_{n+1} = \underline{\hspace{6cm}}$$

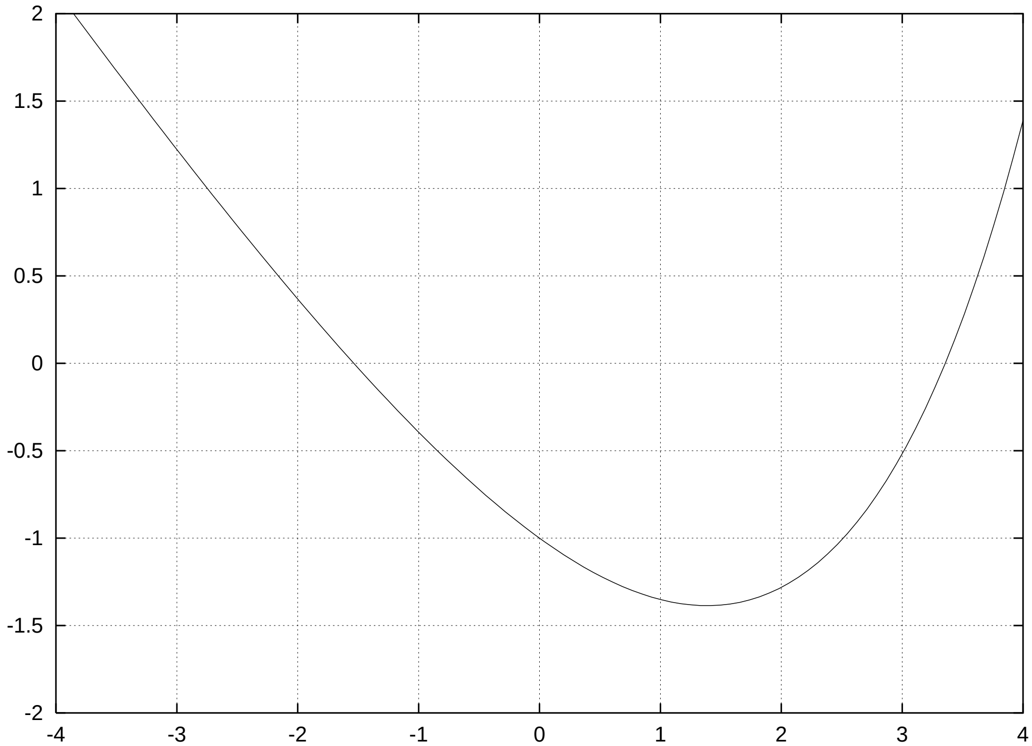
Together with an initial guess x_0, this recursive process defines Newton's Method. Under the conditions listed above, we are guaranteed that

$$\lim_{n\to\infty} x_n = r.$$

Visualizing Newton's Method

Recall that in the lab we derived Newton's Method by considering the equation of the tangent line to a function $h(t)$ at the point $t = t_0$ and then considering the root t_1 of this tangent line to be the approximation to the root t^* of the function $h(t)$.

Consider the graph of $h(t) = e^{t/2} - t - 2$ shown below on the interval $-4 \leq t \leq 4$.



1. Draw the tangent line to $h(t)$ at $t_0 = 1$. Extend the tangent line to the t-axis. Label this point t_1.

2. Draw the tangent line at $t = t_1$. Find its root and label this point t_2. Repeat as often as you can.

3. What do you notice about the sequence of points t_0, t_1, t_2, $\ldots$?

4. Visually, how does the limit of your sequence depend on the initial value t_0?

5. Are there any initial values which will cause the sequence to not converge?

4. Let $g(x) = x^2 - 17$. Confirm that g, g', and g'' are continuous (everywhere), and that $g(x) \neq 0$ on an interval containing the root $r = \sqrt{17}$ and the initial guess $x_0 = 1$. Use three iterations of Newton's Method to approximate the root $r = \sqrt{17}$. For greatest accuracy, record your results as fractions or use the memory registers on you calculator to store intermediate results. Compare this with the value for $\sqrt{17}$ given by your calculator.

n	x_n	$\Delta x \approx -g(x_n)/g'(x_n)$
0	1	
1		
2		
3		
4		
5		
6		

Question
Do you see any connection between Newton's Method and the ancient Babylonian Algorithm?

(HINT: Try writing down the recursive formula you would need to use Newton's Method to solve find the root of $f(x) = x^2 - A$.)