Physics 209: Assignment #9

Part I.

As demonstrated in class for the case $n = 2$, the map

$$x \longrightarrow \tfrac{1}{2}\left(x + \frac{n}{x}\right) \tag{1}$$

rapidly converges to the fixed point $x^* = \sqrt{n}$. This map is so easy to carry out with the calculator that it's not worth the trouble to enter a program into program memory. First set the calculator so that it shows all twelve decimal places by doing [y] **ALL** (61 41). Then enter your first guess for $\sqrt{n}$ into the display as the first value for x. Then do the following series of keystrokes:

+ **1/x** $\times$ n **=** $\div$ **2** **=**

This leaves the next estimate for $\sqrt{n}$ in the display. You then interate until you reach a fixed point, which is the value of $\sqrt{n}$.

Do this procedure for $\sqrt{5}$, starting with the guess $x = 2$. Starting with 2.0, make a list of the results of subsequent iterations until you reach the fixed point. Comment on the extent to which the results of successive iterations get better.

Do the same procedure for $\sqrt{9}$ (even though you already know the answer) starting again with the guess $x = 2$. Because the answer you are converging toward is so simple, the manner in which the results of successive iterations get better should be more apparent.

Part II.

Explain why the fixed point of the map

$$x \longrightarrow \tfrac{1}{2}\left(x + \frac{n}{x^2}\right) \tag{2}$$

is $\sqrt[3]{n}$, the cube root of n. You can evaluate this by adding just one step to the sequence of strokes we used to evaluate (1):

+ **1/x** x^2 $\times$ n **=** $\div$ **2** **=**

(The extra step requires two keystrokes, since it's executed by the square root key in its blue mode: 51 11.)

Use this procedure to get $\sqrt[3]{2}$, starting with the estimate 1.25. (That's a pretty good guess, since $1.25 = 5/4$; and $5^3 = 125$ which is very nearly twice $4^3 = 64$.) Again make a list of successive iterations and again comment on how well they are converging, compared with the convergence you were getting in Part I.[1] It should be noticeably slower. Report on some examples that give a quantitative indication of how much slower.

The reason the cube-root procedure is less efficient than the square-root procedure is roughly this:

If x underestimates $\sqrt[3]{n}$ then n/x^2 will (clearly) overestimate it, and vice-versa. So it sounds reasonable that averaging x and n/x^2 will get you closer in, and it does. But if x is off by a little bit, n/x^2 will be off by rather more (on the other side) since we're using the estimate x *twice*. This suggests that maybe we should give twice as much weight to x as to n/x^2 in the averaging procedure, which would replace (2) with

$$x \longrightarrow \tfrac{1}{3}\Big(2x + \frac{n}{x^2}\Big). \tag{3}$$

If we're interested in getting $\sqrt[3]{2}$ then $n = 2$ and this simplifies to

$$x \longrightarrow \tfrac{2}{3}\Big(x + \frac{1}{x^2}\Big). \tag{4}$$

While this argument is only suggestive, you can see for yourself (below) what an improvement it leads to.

The keystrokes that iterate (4) for you are:

+ **1/x** x^2 **=** $\times$ **2** $\div$ **3** **=**

Make a table of the results of successive iterations starting with the estimate $\sqrt[3]{2} \approx 1.25$ and discuss how the convergence compares with what you found when you iterated (2).

Part III.

We noted in class that one can find a fixed point of the map

$$x \longrightarrow \cos(x) \tag{5}$$

[1] If you get impatient with the keystrokes you can easily enter a program to do the job for you. If you have no program in the calculator you want to save, then first clear program memory by doing [y] **CLPRGM** (61 31), and then go to program mode by doing [b] **PRGM** (51 26). If you want to keep the program you already have, do *GTO* .. (51 41 73 73) which takes you to program step 0, and then enter the new program, which is listed on page 4. Any other programs you have will just be pushed up in program memory, as long as the total number of steps in all the programs is less than 99.) Store the number n whose cube root you want to get in location 1 before you run the program (by entering it in the display and then doing **STO 1** (21 1).) Then put your first guess in the display, do **XEQ D** (41 14) to run the program, and then do **R/S** (26) for subsequent iterations.

with striking speed, by starting with any value of x between 0 and 1 and just repeatedly pressing the **cos** button (24). [2] Suppose, however, you need a fixed point of the map

$$x \longrightarrow \cos^2(x). \tag{6}$$

[By $\cos^2(x)$ one always means the square of $\cos(x)$.] One can evaluate this almost as easily, by first pressing the **cos** button and then pressing the x^2 button(s) (51 11). If you start iterating at $x = 0.5$ you will discover that you hop back and forth between an upper value (that slowly drops) and a lower value (that slowly goes up), just as the map (5) does. This time, however, convergence is irritatingly slow. But you can speed it up by the following trick:

Clearly the calculator is trying to get to a value between the upper and lower ones, so why not help it along?[3] With (say) the lower value in the display do **+** (75) then do the usual **cos** x^2; then do **=** $\div$ **2** **=**. The result is to find the average of the lower value and the higher one the map converts it into, presumably getting you a lot closer to the fixed point. Subsequent iterations reveal that one is still hopping back and forth, but over a smaller range, still converging irritatingly slowly. So repeat the speed-up process again. Check to see how things are going. If necessary, repeat the speed-up process again. And again. Eventually this will get you to the fixed point x^*. Report the value of the fixed point — i.e. the solution to the equation $x^* = \cos^2(x^*)$. Can you get it to the full 12-place accuracy of the calculator? (To do this you have to inspect the hidden places by doing [*b*] *SHOW* (51 73) as described in footnote 3 below.) Comment on how often you had to help the convergence along.

We shall be making use of this trick in the weeks ahead.

[2] It is necessary to put the calculator into radian mode by doing [*y*] **cos** (61 41), after which it will stay in radian mode forever unless you change it to another mode, which there will never be any reason to do. (When the calculator is in radian mode the little letters **RAD** appear at the bottom of the display.) It is also a good idea to take the calculator out of **ALL** mode and set it to show just 9 places past the decimal point, by doing [*b*] **FIX** 9 (51 33 9). If you should want to see the undisplayed additional digits in this mode the calculator will flash them at you when you press [*b*] **SHOW** (51 73)(and will continue to display them as long as you hold down the **SHOW** button.)

[3] This may seem like cheating, but it's not. In an iterative process you always start with a *guess*. So anything you do that might improve your guess is just as respectable (and possibly much more useful) than your original guess.

Program for iterating Eq. (2)

Store the value of n in location 1, and put the initial guess x for $\sqrt[3]{n}$ in the display. Do **XEQ D** to run the program the first time. The new estimate for x will appear in the display. Just do **R/S** for subsequent iterations.

No.		Key name	Key code	Comments
1	[y]	**LBL D**	*61 41 d*	*Labels program D*
2		**+**	*75*	*Adds to number in display*
3		**1/x**	*15*	*its inverse*
4	[b]	x^2	*51 11*	*squared*
5		$\times$	*55*	*and multiplied by*
6		**RCL 1**	*22 1*	*n, stored in location 1,*
7		**=**	*74*	*all of which is then*
8		$\div$	*45*	*divided by*
9		**2**	*2*	*2*
10		**=**	*74*	*and left in the display.*
11		**R/S**	*26*	*After which the program stops*
12	[b]	**GTO D**	*51 41 d*	*until you press* **R/S** *and it runs again.*