Selected Solutions to Assignment #3

Note. Sometimes I will list problems in other than the usual order. That is usually a suggestion that you start with the easiest and work toward the hardest.

Problems 3.2, exercise 9. Newton's method simplifies:

$$x_{n+1} = x_n - \frac{x_n^3 - 2}{3x_n^2} = \frac{2}{3}\left(x_n + \frac{1}{x_n^2}\right)$$

It follows that $x_0 = 1$, $x_1 = 4/3$, and

$$x_2 = \frac{2}{3}\left(\frac{4}{3} + \frac{9}{16}\right) \approx 1.26388.$$

Note that $\sqrt[3]{2} \approx 1.25992$.

Problems 3.2, exercise 10. Apply Newton's method to the equation $x^3 - R = 0$. Again we might simplify:

$$x_{n+1} = x_n - \frac{x_n^3 - R}{3x_n^2} = \frac{2}{3}x_n + \frac{R}{3}x_n^{-2}$$

But the main idea is to produce a good picture. By drawing several graphs like the one shown in Figure 1 below, which happens to use $R = 3$ and show $x_0 = 1$, and x_1, x_2 from the Newton iteration, I got an intuitive sense that if $x_0 > 0$ then the iteration eventually converges. In fact, if $R > 1$ then $x_0 = 1$ always leads to $x_1 > \sqrt[3]{R}$ because the curve is convex, and then $x_1 > x_2 > \ldots$ from then on. (*A slight modification of Theorem 2 could then prove that the Newton iteration converges, but this is not necessary.*) If $0 < R < 1$ then $x_0 = 1$ also works, and $1 = x_0 > x_1 > x_2 > \ldots$.

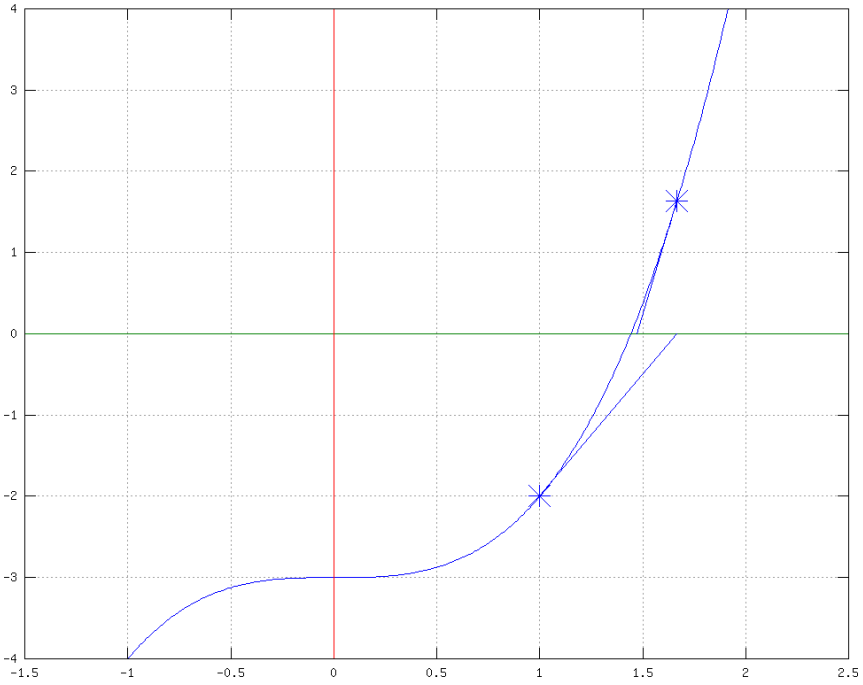

FIGURE 1. A graph of $x_0 = 1$ and x_1, x_2 from Newton's method, for $f(x) = x^3 - 3$.

Problems 3.2, exercise 1. The solution of $\arctan x = 0$ is $x = 0$, so that is the value we want to come out of Newton. So to start, let us write down Newton's method and simplify slightly:

$$x_{n+1} = x_n - \frac{\arctan(x_n)}{\left(\frac{1}{1+x_n^2}\right)} = x_n - \arctan(x_n)(1+x_n^2)$$

Now, some trial-and-error and/or some graphing leads to the understanding that Newton's method always generates a sequence $\{x_n\}$ with alternating sign, but that for x_0 of not-to-big size, $x_n \to 0$. For larger x_0 we get an alternating but diverging sequence.

The critical value, the "smallest positive starting point for which Newton diverges," is the place where $x_{n+1} = -x_n$. That is, cycling occurs when the next value is the same distance from the origin, but on the other side. To find this location we solve the equation "$x_{n+1} = -x_n$" for that special starting point $x_n = s$:

$$-s = s - \arctan(s)(1+s^2) \qquad \text{or} \qquad 0 = 2s - \arctan(s)(1+s^2).$$

How to find s? Use Newton's method! The following sequence produces Figure 2, as well as finding that $s \approx 1.3917$:

```
>> x=-5:0.001:5;
>> plot(x,atan(x)), grid on
>> f = @(x) 2*x - atan(x) .* (1+x.^2);
>> df = @(x) 1 - 2 * x .* atan(x);
>> format long g
>> s = 1.0, for k=1:10, s = s - f(s)/df(s), end
s =                    1
s =     1.75193839388411
...
s =     1.39174520027073
>> hold on,  plot([-s s],[0 atan(s)],'r',[-s s],[atan(-s) 0],'r'),  hold off
```

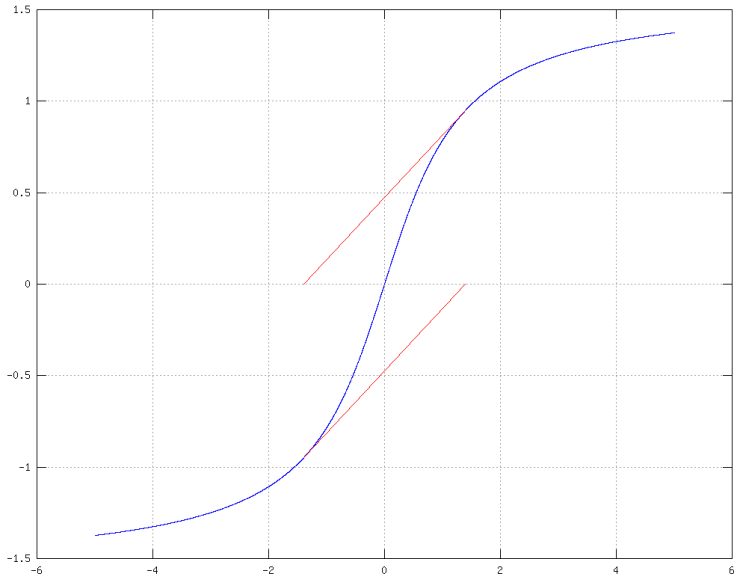

FIGURE 2. A graph of $y = \arctan(x)$ and of the tangent lines that lead to cycling in Newton's method.

Problems 3.2, exercise 6. First state and simplify Newton's method:

$$x_{n+1} = x_n - \frac{x_n^{-1} - R}{-x_n^{-2}} = x_n + (x_n^{-1} - R)x_n^2$$
$$= 2x_n - Rx_n^2 = x_n \cdot (2 - R \cdot x_n)$$

The last form is written to emphasize that we need to do two multiplications and one subtraction in order to get x_{n+1} from x_n.

How to get a staring point? Consider only $R > 0$ for simplicity. A bit of trial-and-error or graphing shows that starting points that are not much larger than $1/R$ lead to negative values for x_n at some stage, and then divergence. So we need a recipe for generating a starting point x_0 in the interval $0 < x_0 < 1/R$, but without knowing $1/R$, which is cheating.

My practical suggestion is that we may suppose that the number R is represented already in scientific notation (or internally floating point form). In that case, we know

$$R = a \times 10^k$$

where $1 \le a < 10$. For such R I propose this starting point,

$$x_0 = 10^{-k-1}.$$

It is always in the interval $0 < x_0 < 1/R$ (*why?*), and not too close to zero because it is within a factor of 10 of the correct answer (*why?*).

The algorithm is a running MATLAB/OCTAVE code below. It has the caveat that the call to `log10` is obviously silly, and would be replaced in "real" applications with a bit manipulation involving the internal storage of R. Note that some calculators actually use a scheme like this to avoid complex division circuitry:

recip.m

```
function z = recip(R)
% RECIP  Use Newton to invert: R -> 1/R.  Requires R > 0.

k = floor(log10(R))  % an actual application would use internal
x = 10^(-k-1)        %   representation of R to get this starting point

for j=1:10
  x = x * (2 - R * x)
end

z = x
```

Problems 3.2, exercise 14. A sufficient hint for this one was given in class.

Problems 3.2, exercise 17. In each case compute

$$\lim_{n\to\infty} \frac{|x_{n+1}|}{|x_n|^2}.$$

If this quantity is $+\infty$ then the sequence must converge to zero slower than quadratically because $|x_{n+1}| \le C|x_n|^2$ is not true for any C.

By this standard only **b.** even *could* be quadratic, and it is:

$$\frac{\frac{1}{2^{(2^{n+1})}}}{\frac{1}{(2^{(2^n)})^2}} = \frac{2^{2(2^n)}}{2^{(2^{n+1})}} = \frac{2^{(2^{n+1})}}{2^{(2^{n+1})}} = 1.$$

This shows $x_{n+1} = (x_n)^2$, which is obvious in retrospect.

The only other "interesting" limit is in **e.**, as follows:

$$\lim_{n\to\infty} \frac{(n+1)^{-(n+1)}}{(n^{-n})^2} = \lim_{n\to\infty} \frac{n^{2n}}{(n+1)^{n+1}} = \lim_{n\to\infty} \left(\frac{n}{n+1}\right)^n \left(\frac{n}{n+1}\right) \left(n^{n-1}\right) = e^{-1} \cdot 1 \cdot (+\infty) = +\infty.$$

(*Why is*

$$\lim_{n\to\infty} \left(\frac{n}{n+1}\right)^n = e^{-1} \qquad ?)$$

Computer Problems 3.2, exercise 1. The only issue it the production of starting points, which are necessarily different for the two roots we seek. In fact, here is the solution of **Computer Problems 3.2 #2**, which asks for the first ten solutions of $x = \tan(x)$, starting at $x = 0$, which is a root:

tentanroots.m

```
% TENTANROOTS  Use Newton to find 10 distinct solutions to  x = tan(x)

roots = [0];    % record the obvious root

% the function and its derivative
f  = @(x) x - tan(x);
df = @(x) 1 - sec(x).^2;

% loop over which one we are finding
for k=1:9
  x = (2*k+1 - 0.01) * pi/2;    % starting point just left of (odd)*pi/2
  % loop to do Newton iteration
  for j=1:10
    x = x - f(x) / df(x);
  end
  roots = [roots x];
end

roots = roots'
residuals = abs(f(roots))
```

The result is

```
roots =
                   0
     4.49340945790906
     7.72525183693771
     10.9041216594289
     14.0661939128315
     17.2207552719308
     20.3713029592876
      23.519452498689
     26.6660542588127
      29.811598790893
```

and the largest of the residuals is about 10^{-12}.

Computer Problems 3.2, exercise 3. First plot $y = f(x)$ where

$$f(x) = \frac{\tan x}{x^2}.$$

Find the minimum by solving the equation

$$f'(x) = 0 \qquad \text{which is, after simplifying,} \qquad \frac{x\sec^2 x - 2\tan x}{x^3} = 0.$$

Thus we apply Newton's method to the function

$$g(x) = \frac{x \sec^2 x - 2 \tan x}{x^3}.$$

(This function, and its derivative, can be checked by entering `(tan(x)/(x^2))'` and `(tan(x)/(x^2))''`, respectively, into WOLFRAM ALPHA at `http://www.wolframalpha.com/`. *I don't screw around if getting it wrong will waste my time ...*)

The program:

hardmin.m

```
% HARDMIN  Use Newton to find minimum of f(x) = (tan x) / (x^2)
%          by solving f'(x) = 0.

% the original function
f  = @(x) x.^(-2) .* tan(x)

% draw a picture which explains starting point
x = 0:0.001:2;
figure(1)
plot(x,f(x))
axis([0 2 0 5]), grid on

% the function g(x) = f'(x); to solve g(x) = 0:
g  = @(x) x.^(-3) .* ( x .* sec(x).^2 - 2 * tan(x) )
% its derivative
dg = @(x) x.^(-4) .* ( 6 * tan(x) - 4 * x .* sec(x).^2 + ...
                       2 * x.^2 .* sec(x).^2 .* tan(x) )

% starting point
x = 0.9

% do Newton iteration
for j=1:10
  x = x - g(x) / dg(x)
end

% some numbers to check
f(x)
g(x)
```

The result is to show a useful picture of what we want to minimize, and to find $x = 0.94774713351699$.

Computer Problems 3.2, exercise 5. The usual, but start with plotting as suggested. I narrowed it down to interval $[0.12, 0.125]$ by plotting; note I enter $f(x)$ habitually using Hörner's method for evaluating polynomials, which will be described in-class:

```
>> f = @(x) (((2 .* x + 24) .* x + 61) .* x- 16) .* x + 1;   % using Horner's
>> x=0.1:0.0001:0.13; plot(x,f(x)), grid on                  % not narrowed down yet, but
>> x=0.12:0.00001:0.125; plot(x,f(x)), grid on               % now I can see the roots
```

Now do Newton using the plot to tell you starting points.

Check your answers using MATLAB/OCTAVE's built-in `fzero`[1], perhaps:

[1]By the way, `fzero` is a robust-ified combination of something like bisection plus secant method. More precisely, it is "essentially *ACM algorithm 748: Enclosing Zeros of Continuous Functions* due to Alefeld, Potra and Shi, ACM Transactions on Mathematical Software, Vol. 21, No. 3, September 1995." I found this out by `type fzero` in OCTAVE.

```
>> fzero(f,[0.121 0.122])
ans =    0.121320343559643
>> f(ans)
ans =                     0
>> fzero(f,[0.123 0.124])
ans =     0.12310562561766
>> f(ans)
ans =                     0
```

Problems 3.3, exercise 4.

$$x_2 = x_1 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)} = 1 - \frac{f(1)}{f(1) - f(0)} = 1 - \frac{-1}{(-1) - (-2)} = 2,$$

which is not that impressive as a sequence of approximations to $\sqrt{2}$.

Computer Problems 3.3, exercise 1. The program:

secantELB.m

```
function z = secantELB(f,x0,x1)
% SECANTELB  Secant method by Ed L. Bueler.  Prints intermediate results.
%            *Not* an efficient or robust implementation.
% Good example:
%   >> f = @(x) x.^3 - 12 * x.^2 + 3 * x + 1
%   >> x = -5:0.01:15;  plot(x,f(x)),  grid on,  axis([-5 15 -20 20])  % locate roots
%   >> x = 0:0.001:1;   plot(x,f(x)),  grid on                         % closer view
%   >> secantELB(f,0.4,0.45)
% Bad example:
%   >> g = @(x) sin(x/2) - 1
%   >> secantELB(g,3,3.1)

while abs(x1-x0)/abs(x1) > 10*eps
  xnew = x1 - feval(f,x1) * (x1 - x0) / ( feval(f,x1) - feval(f,x0) )
  x0 = x1;
  x1 = xnew;
end
z = xnew;
```