Homework Solutions – Due Tues. 02/19/2018

1. a. (4pts) Using Newton's method, explain why the sequence

$$x_n = \frac{1}{2}x_{n-1} + \frac{A}{2x_{n-1}},\ n \geq 1,\ x_0 > 0,$$

converges to $\sqrt{A}$.

Solution:
a. The function $f(x) = x^2 - A$ has a unique root $x = \sqrt{A}$ for $x > 0$. Furthermore, this function is monotonically increasing and concave upward for $x > 0$. The derivative of $f(x)$ is $f'(x) = 2x$, which is monotonically increasing for $x > 0$. Newton's method for this function satisfies:

$$x_n = x_{n-1} - \frac{x_{n-1}^2 - A}{2x_{n-1}} = \frac{1}{2}x_{n-1} + \frac{A}{2x_{n-1}}.$$

Since this function has a simple root crossing and $f'(x) = 2x > 0$ for $x > 0$, it is easy to see that any choice of $x_0 \in (0, \sqrt{A})$ produces an $x_1 > \sqrt{A}$ (following the tangent line from x_0 to the intersection of the x-axis). For any $x_n > \sqrt{A}$, the tangent line from x_n falls short of $\sqrt{A}$, when it crosses the x-axis. It follows that the Newton sequence approaches $\sqrt{A}$, as $n \to \infty$.

b. (3pts) Below we provide a MatLab code for this Newton's method, which allows finding the square root of any number A and includes inputs for A, x_0, tolerance, and maximum number of iterations.

```
function x = newtonhw1(A,x0,tol,Nmax)
%NEWTON'S METHOD: Enter f(x), f'(x), x0, tol, Nmax
    f = @(x) x.^2-A;
    fp = @(x) 2*x;
    y = f(x0);
    x = x0 - f(x0)/fp(x0);
    i = 1;
    fprintf('n = %d, xn = %10.8f \n',i,x)
        while (abs(x - x0) ≥ tol)
        x0 = x;
        x = x0 - f(x0)/fp(x0);
        i = i + 1;
        fprintf('n = %d, xn = %10.8f \n',i,x)
            if (i ≥ Nmax)
            fprintf('Fail after %d iterations\n',Nmax);
            break
            end
        end
end
```

With a tolerance of 10^{-10}, this sequence converges to:

$$\sqrt{7} = 2.645751311064591$$

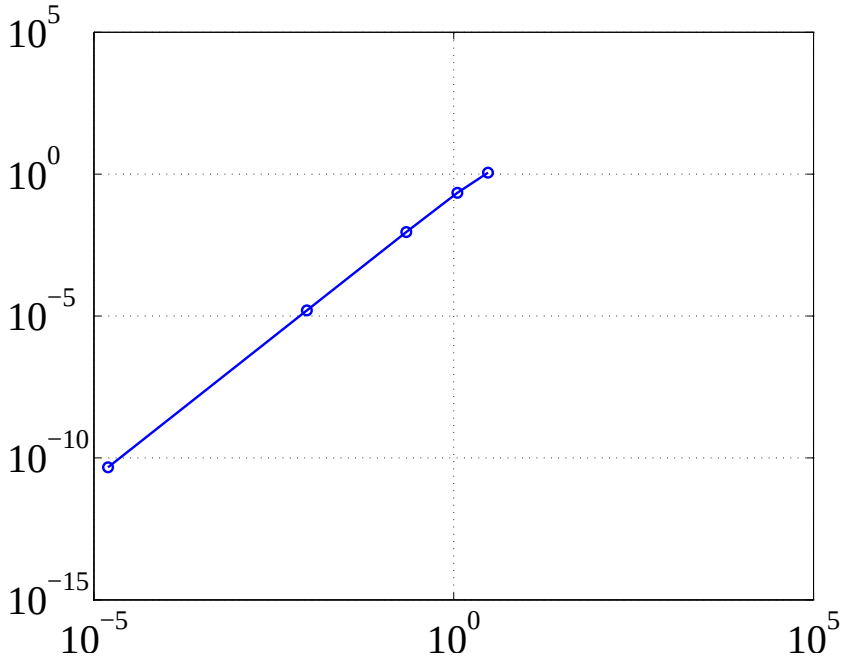
c. (3pts) We demonstrate two methods for estimating α and L for the Cauchy sequence:

$$\frac{|x_{n+1} - x_n|}{|x_n - x_{n-1}|^\alpha} \approx L.$$

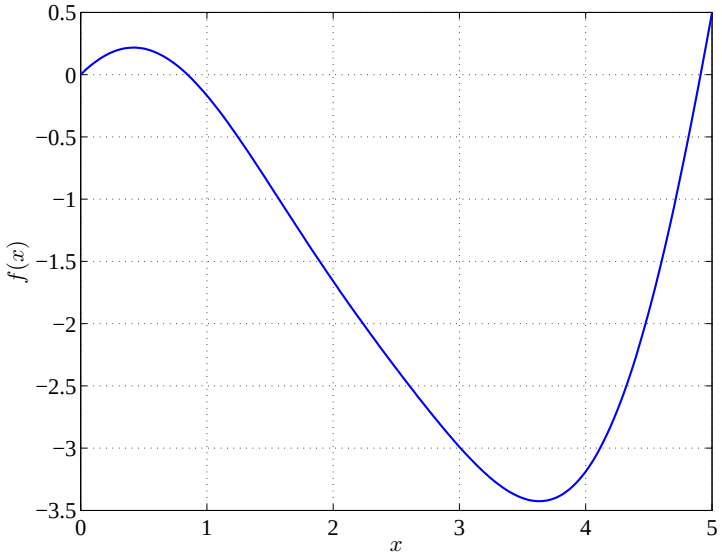
In the first case, we make the assumption that Newton's method is converging quadratically, so $\alpha = 2$. We create our Cauchy sequence and find that that for the higher iterates with the quadratic convergence, the ratio is $L \approx 0.189$. (Specifically, the ratios when $x_0 = 7$ and $tol = 1e-12$, are 0.125, 0.174, 0.188, 0.189, and 0.189.)

Alternately, we take the logarithms of $|x_{n+1} - x_n|$ for $n = 1, ..., 6$, find the best fitting line through $X = \ln(|x_{n+1} - x_n|)$ and $Y = \ln(|x_{n+2} - x_{n+1}|)$. The result is $\alpha = 1.97858$ and $L = 0.1596$, which are close to the values assuming $\alpha = 2$, and noting that the best fit includes an average of the early values, where L is lower. (See the MatLab script below.)

```
% Modify the Newton program above replacing the fprintf
% commands with vector, z, storing the iterates

z = newtonhw1c(7,7,1e-12,20);
k = length(z);

% If alpha = 2, then
y = abs(z(3:k-1)-z(2:k-2));
x = abs(z(2:k-2)-z(1:k-3));
L = y./x.^2

% Alternately, use ln(y) = ln(L)+alpha*ln(x)
Y = log(y); X = log(x);
c = polyfit(X,Y,1)
exp(c(2))
loglog(x,y,'bo-');grid;
print -depsc hw1_conv_gr.eps       % Create EPS file (Figure)
```



2. a. (5pts) The graph below on the interval [0,5] shows there are 3 roots.



There is the obvious root, $x = 0$. Beginning with $x_0 = 1$, Newton's method readily finds a second root at $x_1* = 0.84644635486$. Similarly with $x_0 = 5$, Newton's method readily finds a third root at $x_2* = 4.9095343914$. In each case, the program below was run to a tolerance of $1e - 10$.

```
function x = newtonhw2a(x0,tol,Nmax)
%NEWTON'S METHOD: Enter f(x), f'(x), x0, tol, Nmax
    f = @(x) x.*cos(x)-(sin(x)).^2;
    fp = @(x) cos(x)-2.*cos(x).*sin(x)-x.*sin(x);
    y = f(x0);
    x = x0 - f(x0)/fp(x0);
    i = 1;
    fprintf('n = %d, xn = %10.8f \n',i,x)
        while (abs(x - x0) ≥ tol)
        x0 = x;
        x = x0 - f(x0)/fp(x0);
        i = i + 1;
        fprintf('n = %d, xn = %10.8f \n',i,x)
            if (i ≥ Nmax)
            fprintf('Fail after %d iterations\n',Nmax);
            break
            end
        end
end
```

b. (6pts) The following program is designed to find the rate of converge near any of the **3** zeroes of

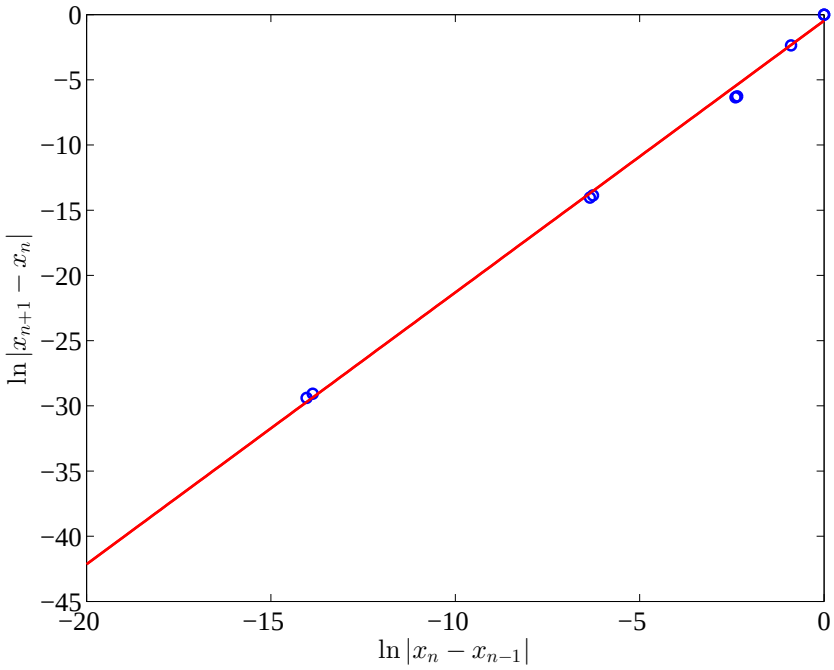
$$f(x) = x\cos(x) - \sin^2(x),$$

where convergence is computed by finding the linear least squares fit to

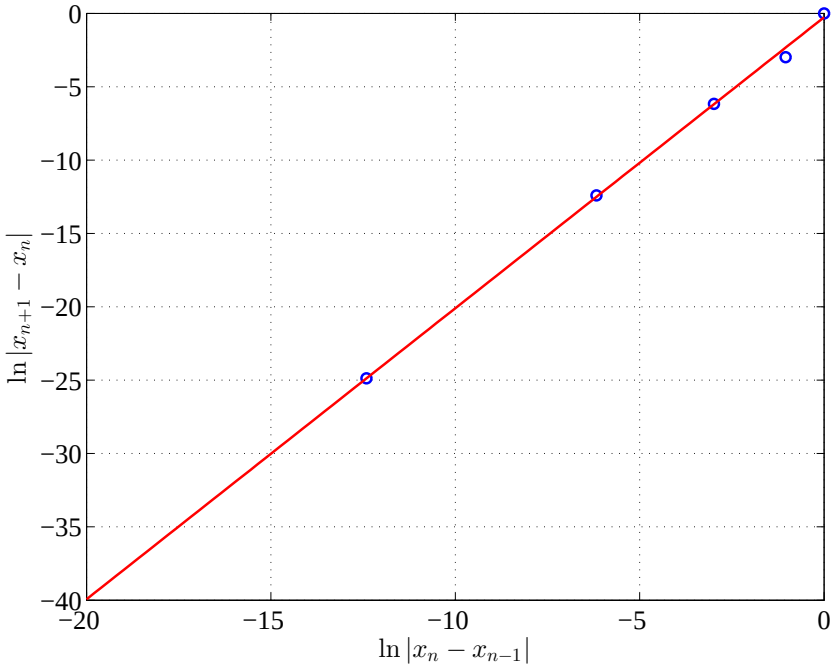
$$\ln|x_{n+1} - x_*| \sim \alpha \ln|x_n - x_*| + \ln\lambda.$$

```
function xn = newtonhw2b2(x0,tol,Nmax)
%NEWTON'S METHOD: Enter f(x), f'(x), x0, tol, Nmax
    f = @(x) x.*cos(x)-(sin(x)).^2;
    fp = @(x) cos(x)-2.*cos(x).*sin(x)-x.*sin(x);
    i = 1;
    xn = [x0];
    x = x0 - f(x0)/fp(x0);
    xn = [xn,x];
        while (abs(x - x0) ≥ tol)
        x0 = x;
        x = x0 - f(x0)/fp(x0);
        xn = [xn,x];
        i = i + 1;
        if (i ≥ Nmax)
            fprintf('Fail after %d iterations\n',Nmax);
            break
            end
        end
        xs = x;
        nn = length(xn);
   for n=3:nn
        Y(n)= log(abs(xn(n-1)-xs));
        X(n)= log(abs(xn(n-2)-xs));
   end
   plot (X,Y, 'bo')
   hold on;
   grid;
   pf = polyfit(X,Y,1)
   xx=linspace(-20,0,100);
   plot(xx,pf(1,1).*xx+pf(1,2), 'r-')
% Set up fonts and labels for the Graph
fontlabs = 'Times New Roman';
xlabel('$\ln|x_{n}-x_{n-1}|$','FontSize',16,'FontName',fontlabs,...
    'interpreter','latex');
ylabel('$\ln|x_{n+1}-x_n|$','FontSize',16,'FontName',fontlabs,...
    'interpreter','latex');
set(gca,'FontSize',16);

print -depsc hw2_conv_gr0.eps
end
```

For the root, $x_* = 0$ and starting with $x_0 = 0.3$, the program gives a convergence rate $\alpha = 2.0045$ with $\lambda = e^{0.045822} \approx 1.04689$ (which is high because the original change of sign of the iterations). The value of λ is not important (though it is usually < 1), but $\alpha \approx 2$ shows the quadratic convergence that we expect of Newton's method. The graph is shown below:

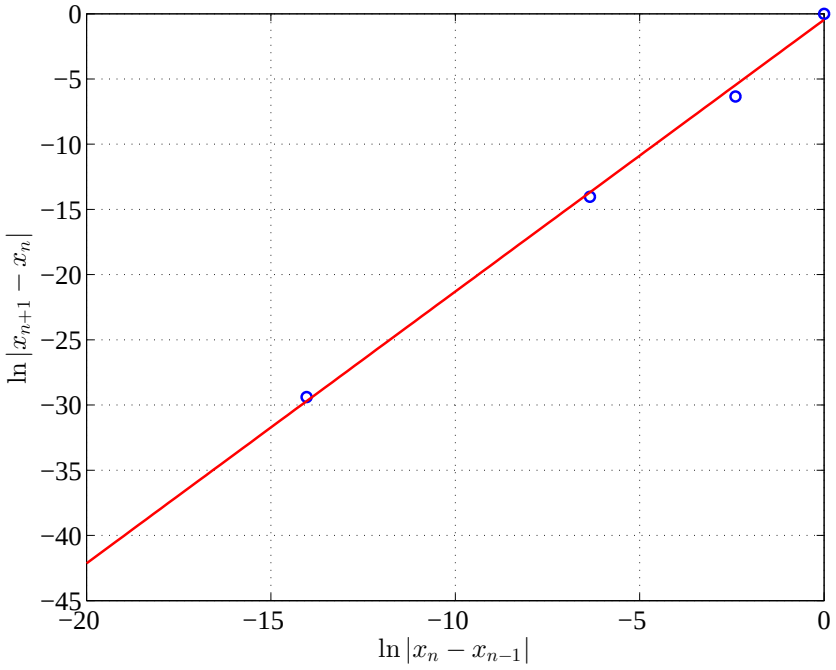


For the root, $x_* = 0.846446354857407$ and starting with $x_0 = 1.2$, the program gives a convergence rate $\alpha = 1.9833$ with $\lambda = e^{-0.268076} \approx 0.76485$. The value of λ is not important (though it is usually < 1), but $\alpha \approx 2$ shows the quadratic convergence that we expect of Newton's method. The graph is shown below:

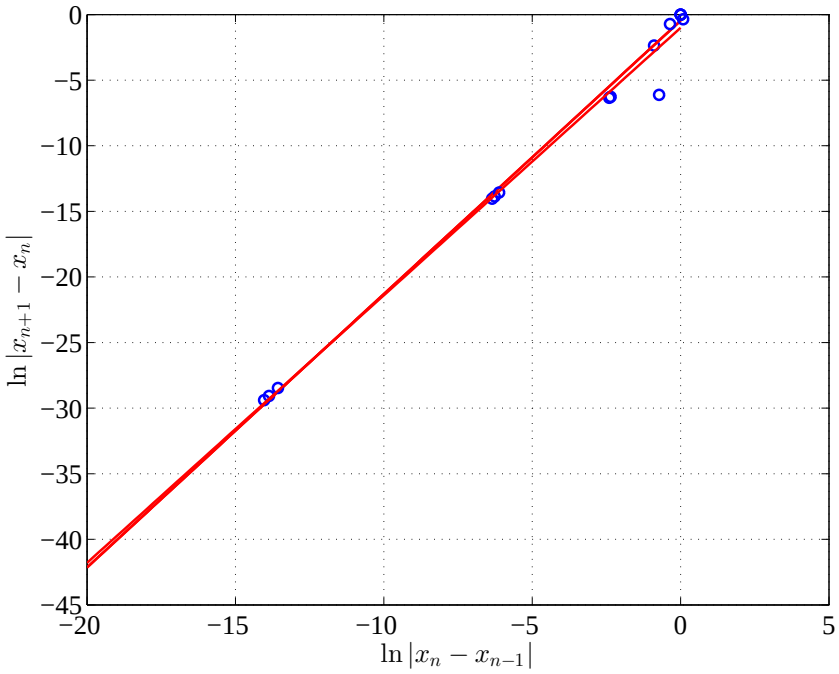


For the root, $x_* = 4.909534391393061$ and starting with $x_0 = 5$, the program gives a convergence rate $\alpha = 2.0842$ with $\lambda = e^{-0.268076} \approx 0.63205$. The value of λ is not important

(though it is usually < 1), but $\alpha \approx 2$ shows the quadratic convergence that we expect of Newton's method. The graph is shown below:



Considering this last case, if we start at say $x_0 = 4.5$, 5, or 6, we obtain $\alpha = 2.084$, 2.084, and 2.039, which is consistent with the quadratic convergence we expect from Newton's method. (The value of λ changes a fair amount.) The graph below shows all 3 starting conditions $x_0 = 4.5, 5, 6$ with the obvious same slopes.



c. (4pts) The following two programs generated the table below.

```
function M = hw_2(tol,Nmax)
%Basin of attraction
M = zeros(51,2);
x0 = 0;
for i = 1:51
    M(i,1) = x0;
    M(i,2) = hw4newton_2(x0,tol,Nmax);
    x0 = x0 + 0.1;
end
end
```

```
function xn = hw4newton_2(x0,tol,Nmax)
%NEWTON'S METHOD: Enter f(x), f'(x), x0, tol, Nmax
f = @(x) x*cos(x)-(sin(x))^2;
fp = @(x) cos(x)-x*sin(x)-2*sin(x)*cos(x);
xn = x0 - f(x0)/fp(x0);
i = 1;
while (abs(xn - x0) ≥ tol)
    x0 = xn;
    xn = x0 - f(x0)/fp(x0);
    i = i + 1;
    if (i ≥ Nmax)
        xn = 1000;
        break
    end
end
end
```

x_0	Root	x_0	Root	x_0	Root
0.0	0	1.7	0.84645	3.4	-1.99817
0.1	0	1.8	0.84645	3.5	-4.49848
0.2	0	1.9	0.84645	3.6	-39.2953
0.3	0	2.0	0.84645	3.7	20.3714
0.4	-4.49848	2.1	0.84645	3.8	14.0664
0.5	0.84645	2.2	0.84645	3.9	7.72631
0.6	0.84645	2.3	0.84645	4.0	-10.9045
0.7	0.84645	2.4	0.84645	4.1	4.90953
0.8	0.84645	2.5	0.84645	4.2	4.90953
0.9	0.84645	2.6	0.84645	4.3	4.90953
1.0	0.84645	2.7	0.84645	4.4	4.90953
1.1	0.84645	2.8	0.84645	4.5	4.90953
1.2	0.84645	2.9	0.84645	4.6	4.90953
1.3	0.84645	3.0	39.2444	4.7	4.90953
1.4	0.84645	3.1	0	4.8	4.90953
1.5	0.84645	3.2	0	4.9	4.90953
1.6	0.84645	3.3	0.84645	5.0	4.90953

The basin of attraction program found that when $x_0 = [0, 0.3]$ there is a convergence to $x* = 0$. When $x_0 = [0.5, 2.9]$ the convergence is $x* = 0.8465$, and when $x_0 = [4.1, 5.0]$ the convergence is to $x* = 4.9095$. There are anomalies at $x_0 = 0.4$ and for all $x_0 \in [3.0, 4.0]$. The anomalies occur between the basins for the 3 roots. The broad range from $[3, 4]$ occurs near the minimum, and the initial tangent line gives x_1 a distance away from the closest root.

Results from the **WeBWorK** homework FZero

WW1 d. (9pts) MatLab code and convergence:

```
function z = bisection(a,b,tol)
%BISECTION METHOD - Modify function below, then can
% find its roots in [a,b] to tolerance tol
f = @(x) exp(x)+4^(-x)+1.5*cos(x)-5.5;
n = 0;
z = [(a+b)/2];
while (abs(b-a) ≥ tol)
    m = (a+b)/2;
    if (f(m) == 0)
        break;
    elseif(f(b)*f(m) < 0)
        a = m;
    else
        b = m;
    end
    y=f((a+b)/2);
    z = [z,(a+b)/2];
    n=n+1;
    fprintf('a = %f6, b= %f6, f=%f6,n=%d\n',a,b,y,n);
end
end
```

Bisection method has linear convergence, so $\alpha = 1$. If one uses the Cauchy convergence

$$\lim_{n\to\infty} \frac{|x_{n+1} - x_n}{|x_n - x_{n-1}|^\alpha} = L$$

with $\alpha = 1$, then $L = 0.5$. Using the actual x_* and the precise definition produces results that are hard to interpret.

```
function z = secantww(x0,x1,tol,Nmax)
%SECANT METHOD: Enter f(x), x0, x1, tol, Nmax
f = @(x) exp(x)+4^(-x)+1.5*cos(x)-5.5;
xn = x1 - f(x1)*(x1-x0)/(f(x1)-f(x0));
z = [xn];
y = f(xn);
fprintf('x=%d, f(x) = %d\n', xn, y)
i = 1;
while (abs(xn - x1) ≥ tol)
 x0 = x1;
 x1 = xn;
 xn = x1 - f(x1)*(x0-x1)/(f(x0)-f(x1));
```

```
  z = [z,xn];
  y = f(xn);
  i = i + 1;
  if (i ≥ Nmax)
     fprintf('Fail after %d iterations\n',Nmax);
     break
  end

fprintf('x=%d, f(x) = %d\n', xn, y)
  end
  end
```

Secant method has superlinear convergence with $\alpha \approx 1.62$. If one uses the Cauchy convergence, then the later values of $L \approx 0.8$. If the definition is used where x_* is the last value, then the value of $L \approx 0.87$, a little higher. Once again the L values are more problem dependent and will vary, so your answer only needs a discussion of what you observed.

```
function z = newtonww1(x0,tol,Nmax)
%NEWTON'S METHOD: Enter f(x), f'(x), x0, tol, Nmax
f = @(x) exp(x) + 4^(-x) + 1.5*cos(x) - 5.5;
fp = @(x) exp(x) - log(4)*4^(-x) - 1.5*sin(x);
xn = x0 - f(x0)/fp(x0);
z = [x0,xn];
y = f(xn);
i = 1;
fprintf('n = %d, x = %f, f(x) = %f\n', i, xn, y)
while (abs(xn - x0) ≥ tol)
   x0 = xn;
   xn = x0 - f(x0)/fp(x0);
   z = [z,xn];
   y = f(xn);
   i = i + 1;
   fprintf('n = %d, x = %f, f(x) = %f\n', i, xn, y)
      if (i ≥ Nmax)
        fprintf('Fail after %d iterations\n',Nmax);
        break
      end
end
end
```

Newton's method has quadratic convergence with $\alpha = 2$. Whether one uses the Cauchy convergence or the definition, then the value of $L \approx 0.75$ for the example listed in the program. This value could vary a bit for different versions of the problem in WeBWorK.

WW2 d. (9pts) MatLab code and convergence:

```
function z = bisection2(a,b,tol)
%BISECTION METHOD - Modify function below, then can
% find its roots in [a,b] to tolerance tol
f = @(x) log(x-0.9)+0.95*cos(x-0.9);
n = 0;
z = [(a+b)/2];
while (abs(b-a) ≥ tol)
    m = (a+b)/2;
```

```
    if (f(m) == 0)
        break;
    elseif(f(b)*f(m) < 0)
        a = m;
    else
        b = m;
    end
    y=f((a+b)/2);
    z = [z,(a+b)/2];
    n=n+1;
    fprintf('a = %f6, b= %f6, f=%f6,n=%d\n',a,b,y,n);
end
end
```

Bisection method has linear convergence, so $\alpha = 1$. As in the previous case, we see the Cauchy convergence having $L = 0.5$, while using the definition, we obtain results that are hard to interpret with L varying a fair amount.

```
function z = secantww2(x0,x1,tol,Nmax)
%SECANT METHOD: Enter f(x), x0, x1, tol, Nmax
f = @(x) log(x-0.9)+0.95*cos(x-0.9);
xn = x1 - f(x1)*(x1-x0)/(f(x1)-f(x0))
z = [xn];
y = f(xn);
fprintf('x=%d, f(x) = %d\n', xn, y)
i = 1;
while (abs(xn - x1) ≥ tol)
 x0 = x1;
 x1 = xn;
 xn = x1 - f(x1)*(x0-x1)/(f(x0)-f(x1));
 z = [z,xn];
 y = f(xn);
 i = i + 1;
 if (i ≥ Nmax)
    fprintf('Fail after %d iterations\n',Nmax);
    break
 end

fprintf('x=%d, f(x) = %d\n', xn, y)
 end
 end
```

Secant method has superlinear convergence, so $\alpha \approx 1.62$. Using either the definition for convergence or the Cauchy convergence with this example fails to give a consistent value for L using the algorithm above. The value of L oscillates around $L = 1$. This indicates that higher precision is needed.

```
function z = newtonww2(x0,tol,Nmax)
%NEWTON'S METHOD: Enter f(x), f'(x), x0, tol, Nmax
f = @(x) log(x-0.9)+0.95*cos(x-0.9);
fp = @(x) 1/(x - 0.9) - 0.95*sin(x - 0.9);
xn = x0 - f(x0)/fp(x0);
z = [x0,xn];
```

```
y = f(xn);
i = 1;
fprintf('n = %d, x = %f, f(x) = %f\n', i, xn, y)
while (abs(xn - x0) ≥ tol)
   x0 = xn;
   xn = x0 - f(x0)/fp(x0);
   z = [z,xn];
   y = f(xn);
   i = i + 1;
   fprintf('n = %d, x = %f, f(x) = %f\n', i, xn, y)
      if (i ≥ Nmax)
        fprintf('Fail after %d iterations\n',Nmax);
        break
      end
end
end
```

Newton's method has quadratic convergence, so $\alpha = 2$. Whether one uses the Cauchy convergence or the definition, then the value of $L \approx 1.6$ for the example listed in the program. This value could vary a bit for different versions of the problem in WeBWorK.

WW3 d. (3pts) MatLab code and convergence:

```
function z = secantww3(x0,x1,tol,Nmax)
%SECANT METHOD: Enter f(x), x0, x1, tol, Nmax
f = @(x) (1+x)./2.*(x./(1-x+x.^2)).^21-0.5;
xn = x1 - f(x1)*(x1-x0)/(f(x1)-f(x0));
z = [xn];
P = f(xn)+0.5;
fprintf('x=%d, f(x) = %d\n', xn, P)
i = 1;
while (abs(xn - x1) ≥ tol)
 x0 = x1;
 x1 = xn;
 xn = x1 - f(x1)*(x0-x1)/(f(x0)-f(x1));
 z = [z,xn];
 P = f(xn)+0.5;
 i = i + 1;
 if (i ≥ Nmax)
    fprintf('Fail after %d iterations\n',Nmax);
    break
 end

fprintf('x=%d, f(x) = %d\n', xn, P)
 end
 end
```

Secant method has superlinear convergence, so $\alpha \approx 1.62$. Using either the definition for convergence or the Cauchy convergence with this example fails to give a consistent value for L using the algorithm above. The value of L oscillates around $L \approx 0.9$, but it could vary depending on initial conditions and tolerance.

WW4 d. (6pts) MatLab code and convergence:

```
function z = newtonww4(x0,tol,Nmax)
%NEWTON'S METHOD: Enter f(x), f'(x), x0, tol, Nmax
f = @(x) cos(x+2.^0.5)+x.*(x/2+2.^0.5);
fp = @(x) -sin(x + 2^(1/2)) + x + 2^(1/2);
xn = x0 - f(x0)/fp(x0);
z = [x0,xn];
y = f(xn);
i = 1;
fprintf('n = %d, x = %f, f(x) = %f\n', i, xn, y)
while (abs(xn - x0) ≥ tol)
   x0 = xn;
   xn = x0 - f(x0)/fp(x0);
   z = [z,xn];
   y = f(xn);
   i = i + 1;
   fprintf('n = %d, x = %f, f(x) = %f\n', i, xn, y)
      if (i ≥ Nmax)
        fprintf('Fail after %d iterations\n',Nmax);
        break
      end
end
end
```

If you perform a Taylor's series expansion around $x_0 = -\sqrt{2}$, then we see

$$f(x) = \cos(x+\sqrt{2}) + x(x/2+\sqrt{2}) \approx \frac{1}{24}\left(x+\sqrt{2}\right)^4 - \frac{1}{720}\left(x+\sqrt{2}\right)^6 + O\left(\left(x+\sqrt{2}\right)^8\right),$$

which gives a 4^{th} order zero at $x_0 = -\sqrt{2}$. It follows that the convergence of Newton's method is only linear, so $\alpha = 1$. (We saw that the solution took a long time to converge. Using either the definition for convergence or the Cauchy convergence with this example, MatLab gives us a value near $L = 0.75$.

```
function z = halleyww4(x0,tol,Nmax)
%NEWTON'S METHOD: Enter f(x), f'(x), x0, tol, Nmax
f = @(x) cos(x+2.^0.5)+x.*(x/2+2.^0.5);
fp = @(x) x - sin(x + 2^(1/2)) + 2^(1/2);
ffp = @(x) 1 - cos(x + 2.^(1/2));
xn = x0 - f(x0).*fp(x0)./(fp(x0).^2-f(x0).*ffp(x0));
z = [x0,xn];
y = f(xn);
i = 1;
fprintf('n = %d, x = %f, f(x) = %f\n', i, xn, y)
while (abs(xn - x0) ≥ tol)
   x0 = xn;
   xn = x0 - f(x0).*fp(x0)./(fp(x0).^2-f(x0).*ffp(x0));
   z = [z,xn];
   y = f(xn);
   i = i + 1;
   fprintf('n = %d, x = %f, f(x) = %f\n', i, xn, y)
      if (i ≥ Nmax)
        fprintf('Fail after %d iterations\n',Nmax);
        break
      end
end
end
```

In general, Halley's method has cubic convergence or $\alpha = 3$. However, this multiple root should reduce the order of convergence to $\alpha = 3$ when we are close to the root. There are too few iterations going to the root, so we cannot obtain any information about $L = 0$. Halley's method is converging very rapidly toward the root initially (cubic), then if the machine had adequate precision, we would see the convergence slow to quadratic near the root.