MATH/CS 466/666 HW#3

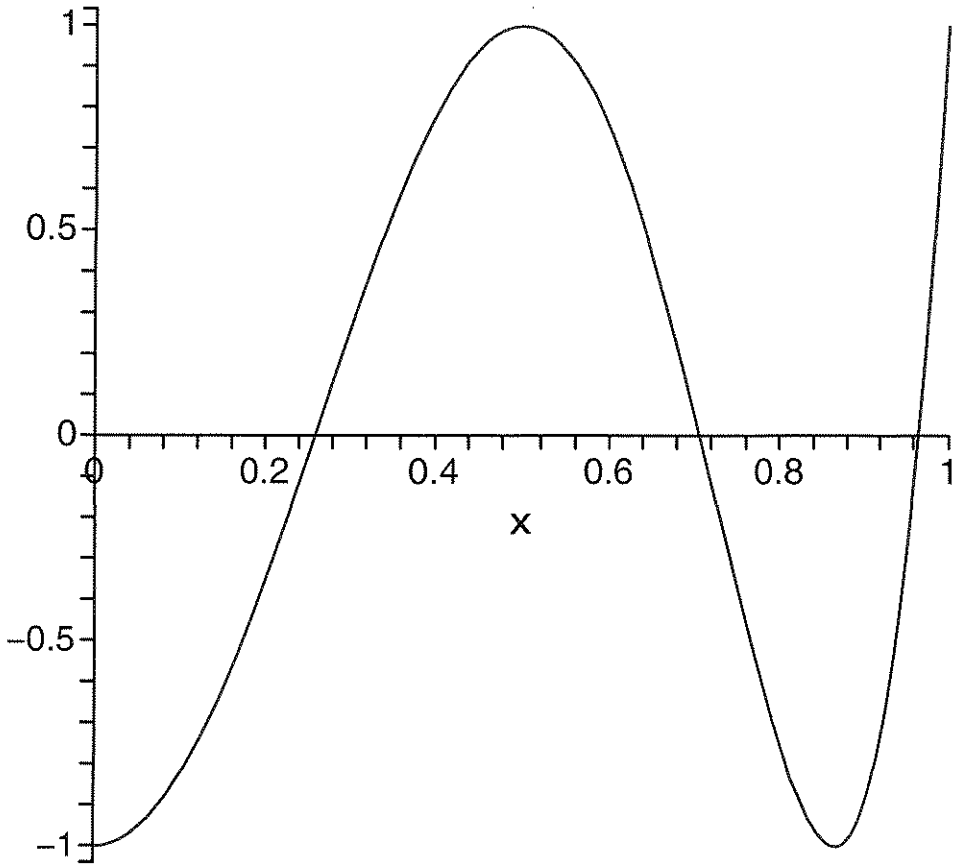
§3.1#6 Using the bisection method and a graph of $f(x)$, find all roots of $f(x) = 32x^6 - 48x^4 + 18x^2 - 1$. The true roots are $\cos((2j-1)\pi/12)$, $j = 1, 2, \ldots, 6$.

```
> #Section 3.1 problem 6
  restart;
```

```
> f:=x->32*x^6-48*x^4+18*x^2-1;
```

$$f := x \to 32\,x^6 - 48\,x^4 + 18\,x^2 - 1$$

```
> plot(f(x),x=0..1);
```



§3.1#6

The MATLAB/Octave code consists of two files. The file `bisect.m` contains an implementation of the bisection algorithm. If `epsilon` is not specified, this bisection terminates when the interval $[a, b]$ reaches the smallest representable interval.

```
function x=bisect(f,a,b,epsilon)
    if nargin<4
        epsilon=0.0;
    end
    fa=feval(f,a);
    fb=feval(f,b);
    if sign(fa)*sign(fb)>0
        fprintf('bisect: Initial bracket no good!\n');
    end
    c=a;
    for n=1:100
        x=c;
        c=(a+b)/2;
        if abs(c-x)<= epsilon
            return
        end
        fc=feval(f,c);
        if sign(fc)*sign(fa)<0
            b=c;
            fb=fc;
        else
            a=c;
            fa=fc;
        end
    end
    fprintf('bisect: Did not converge!\n');
```

The script `s31p6.m` calls the bisection routine to obtain the three positive roots. Due to symmetry the other three roots are negatives of these. Absolute errors compared to the actual values are then computed and displayed.

```
clear all
fprintf('Solution to Section 3.1 Problem 6\n\n');
function y=f(x)
    x2=x*x;
    y=((32*x2-48)*x2+18)*x2-1;
end
format long
r1=bisect(@f,0.0,0.5)
r2=bisect(@f,0.5,0.75)
r3=bisect(@f,0.75,1)
roots=[r3,r2,r1,-r1,-r2,-r3];
```

§3.1#6

```
12 exact=cos((2*[1:6]-1)*pi/12);
13 error=roots-exact;
14 [roots',exact',error']
```

The output from running the script is

```
Solution to Section 3.1 Problem 6

r1 = 0.258819045102521
r2 = 0.707106781186548
r3 = 0.965925826289068
ans =

   0.965925826289068   0.965925826289068  -0.000000000000000
   0.707106781186548   0.707106781186548   0.000000000000000
   0.258819045102521   0.258819045102521   0.000000000000000
  -0.258819045102521  -0.258819045102521  -0.000000000000000
  -0.707106781186548  -0.707106781186547  -0.000000000000000
  -0.965925826289068  -0.965925826289068   0.000000000000000
```

Note that the bisection method yields answers that are equal the true values up to machine precision. This is because the value of `epsilon` was taken to be zero which forces the algorithm to run until it can't numerically bisect the interval any further.

§3.1#8

The polynomial

$$f(x) = x^4 - 5.4x^3 + 10.56x^2 - 8.954x + 2.7951$$

has a root α in $[1, 1.2]$. Solve for α using the bisection method with $\epsilon = 10^{-6}$. Experiment with different ways of evaluating $f(x)$; for example, use (i) the given form, (ii) reverse its order and (iii) the nested form. Try various initial intervals $[a, b]$, for example, $[0, 1.5]$, $[0.5, 2.0]$ and $[0.5, 1.1]$. Explain the results.

Use the same bisection algorithm as in `bisect.m` along with the script `s31p8.m` given by

```
clear all
fprintf('Solution to Section 3.1 Problem 8\n\n');
f1=@(x) x^4-5.4*x^3+10.56*x^2-8.954*x+2.7951;
f2=@(x) 2.7951-8.954*x+10.56*x^2-5.4*x^3+x^4;
f3=@(x) (((x-5.4)*x+10.56)*x-8.954)*x+2.7951;
format long
epsilon=10^(-6)
a=[0,0.5,0.5];
b=[1.5,2.0,1.1];
for n=1:3
fprintf('\nInitial bracket is [a,b]=[%g,%g]\n',a(n),b(n));
    r1=bisect(f1,a(n),b(n),epsilon)
    r2=bisect(f2,a(n),b(n),epsilon)
    r3=bisect(f3,a(n),b(n),epsilon)
end
```

The output is

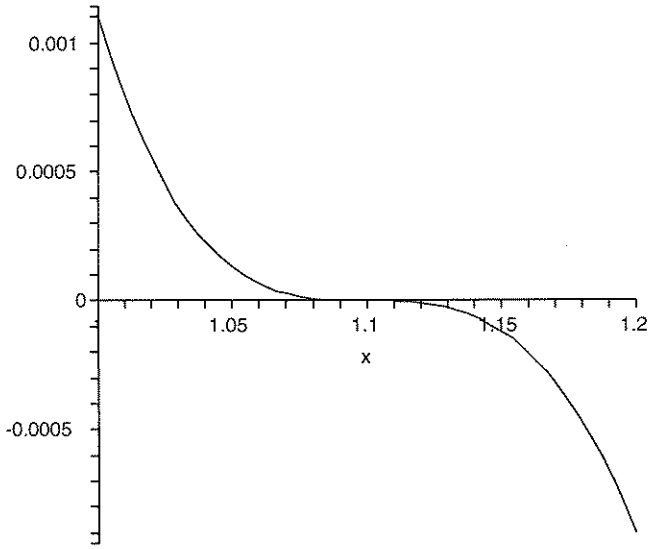
```
Solution to Section 3.1 Problem 8

epsilon =  1.00000000000000e-06

Initial bracket is [a,b]=[0,1.5]
r1 = 1.09999036788940
r2 = 1.09999322891235
r3 = 1.10000467300415

Initial bracket is [a,b]=[0.5,2]
r1 = 1.09998941421509
r2 = 1.09998941421509
r3 = 1.10000085830688

Initial bracket is [a,b]=[0.5,1.1]
r1 = 1.09998970031738
r2 = 1.09998970031738
r3 = 1.09999427795410
```

The output consists of three different initial brackets tested for each of the three different ways of computing the polynomial. In each case the output is slightly different. This is due to the rounding error present in the problem. The Maple script

§3.1#8

```
1 restart;
2 p:=x^4-5.4*x^3+10.56*x^2-8.954*x+2.7951;
3 plot(p,x=1..1.2);
```

produces the plot



Note that the graph of $f(x)$ looks tangent to the x-axis at the root α. Therefore, this root is probably a root of multiplicity 3. The multiplicity of the root makes the computation much more sensitive to rounding error.

93.1#11 Let α be the smallest positive root of

$$f(x) = 1 - x + \sin x.$$

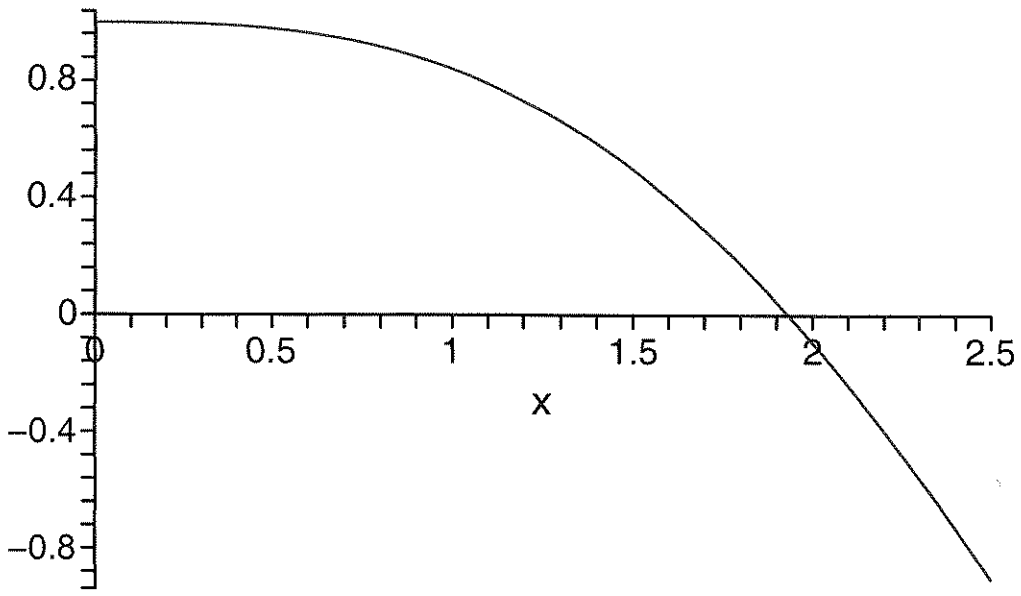
Find an interval $[a, b]$ containing α and for which the bisection method will converge to α. Then estimate the number of iterates needed to find α within an accuracy of 5×10^{-8}.

The Maple graph

```
> restart;
> f:=1-x+sin(x);
```

$$f := 1 - x + \sin(x)$$

```
> plot(f,x=0..2.5);
```



Indicates there is a solution in the interval $[1.5, 2.5]$.

§3.1 #11 continues...

To obtain an accuracy of ε the bisection method takes n iterations where

$$\frac{b-a}{2^{n+1}} \leq \varepsilon$$

Therefore

$$\frac{1}{2^{n+1}} \leq 5\times 10^{-8}$$

implies

$$2^{n+1} \geq \frac{1}{5\times 10^{-8}}$$

$$(n+1)\log 2 \geq -\log(5\times 10^{-8})$$

$$n \geq -1 - \frac{\log(5\times 10^{-8})}{\log 2}$$

$$= -1 + \frac{16.811}{0.6931} = 23.25$$

Therefore $n=24$ iterations will achieve the desired approximation.

§3.2 #4 Give Newtons method for finding $\sqrt[m]{a}$ with $a>0$ and m a positive integer. Apply it to finding $\sqrt[m]{2}$ for $m = 3,4,5,6,7,8$ to six significant digits.

$$f(x) = x^m - a$$

Thus Newton's method is

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^m - a}{m x_n^{m-1}}.$$

For six significant digits we require

$$\left| Rel(x_n) \right| = \left| \frac{\sqrt[m]{a} - x_n}{\sqrt[m]{a}} \right| \leq 5 \times 10^{-6}$$

Since $a = 2$ implies $\sqrt[m]{a} \geq 1$ then

$$\left| \frac{\sqrt[m]{2} - x_n}{\sqrt[m]{2}} \right| \leq \left| \sqrt[m]{2} - x_n \right| = \left| Error(x_n) \right|$$

so we can set epsilon $= 5 \times 10^{-6}$ to obtain an absolute error less than 5×10^{-6}. This will ensure the relative error is less than 5×10^{-6} and so the approximation has 6 significant digits.

Finally use $x_0 = 1.5$ as a starting value for $\sqrt[3]{2}$ and then use $x_0 = \sqrt[m]{2}$ as the starting guess to find $\sqrt[m+1]{2}$. This way the starting approximation is good for each value of m.

§3.2#4

The MATLAB/Octave code for `newton.m` is given by

```
function [x,n]=newton(f,df,x0,epsilon)
    for n=1:100
        x=x0-feval(f,x0)/feval(df,x0);
        if abs(x-x0)<= epsilon
            return;
        end
        x0=x;
    end
    fprintf('newton: Did not converge!\n');
```

and the script `s32p4.m` for this problem is

```
clear all
fprintf('Solution to Section 3.2 Problem 4\n\n');
global m
function y=f(x)
    global m;
    y=x^m-2;
end
function y=df(x)
    global m;
    y=m*x^(m-1);
end
format long

epsilon=5e-6;

x=1.5;
for m=3:8
    x=newton(@f,@df,x,0.0000001);
    fprintf('The %g-th root of 2 is %18.16g\n',m,x);
end
```

The output is

```
Solution to Section 3.2 Problem 4

The 3-th root of 2 is  1.259921049894873
The 4-th root of 2 is  1.189207115002721
The 5-th root of 2 is  1.148698354997035
The 6-th root of 2 is  1.122462048309373
The 7-th root of 2 is  1.104089513673812
The 8-th root of 2 is  1.090507732665258
```

§3.2#11

Find the root of

$$f(x) = x^4 - 5.4x^3 + 10.56x^2 - 8.954x + 2.7951$$

that lies in the interval $[1, 1.2]$ using Newton's method. Experiment with different choices of x_0 and different ways of evaluating $f(x)$ and $f'(x)$.

We choose the same three ways of evaluating $f(x)$ as in §3.1#8 and evaluate the corresponding derivatives $f'(x)$ in a similar fashion. We use starting guesses for x_0 of 0.5, 1.0 and 1.5 and approximate the solution in nine different ways. The script `s32p11.m` is

```
clear all
fprintf('Solution to Section 3.2 Problem 11\n');
f1=@(x) x^4-5.4*x^3+10.56*x^2-8.954*x+2.7951;
df1=@(x) 4*x^3-16.2*x^2+21.12*x-8.954;
f2=@(x) 2.7951-8.954*x+10.56*x^2-5.4*x^3+x^4;
df2=@(x) -8.954+21.12*x-16.2*x^2+4*x^3;
f3=@(x) (((x-5.4)*x+10.56)*x-8.954)*x+2.7951;
df3=@(x) ((4*x-16.2)*x+21.12)*x-8.954;

format long
epsilon=5e-6;
x0=[0.5,1,1.5];
for n=1:3
    fprintf('\nInitial guess is x0=%g\n',x0(n));
    [r1,n1]=newton(f1,df1,x0(n),epsilon)
    [r2,n2]=newton(f2,df2,x0(n),epsilon)
    [r3,n3]=newton(f3,df3,x0(n),epsilon)
end
```

and used along with the function file `newton.m` from problem §3.2#4. The output is

```
Solution to Section 3.2 Problem 11

Initial guess is x0=0.5
r1 = 1.09998801434906
n1 = 28
r2 = 1.09998537073140
n2 = 27
r3 = 1.09998903296948
n3 = 28

Initial guess is x0=1
r1 = 1.09999553685484
n1 = 24
r2 = 1.09998736700643
n2 = 24
r3 = 1.09999026422473
n3 = 23

Initial guess is x0=1.5
r1 = 1.10001076874808
```

§3.2#11

```
n1 = 26
r2 = 1.09998646067948
n2 = 29
r3 = 1.09998856998908
n3 = 29
```

Usually Newton's method converges in 4–5 iterations; therefore, it is unusual that Newton's method is taking more than 20 iterations to converge in this case. As discussed earlier, the graph appearing in the solution to §3.1#8 shows that α is most likely a root of multiplicity three. Therefore, $f'(\alpha) = 0$ so that the requirements for quadratic convergence in formula (3.19) in the text are not met. Note also that this slowness of convergence is present for all three different ways of evaluating $f(x)$ and $f'(x)$ and all three different initial values.

§3.3#3

The function

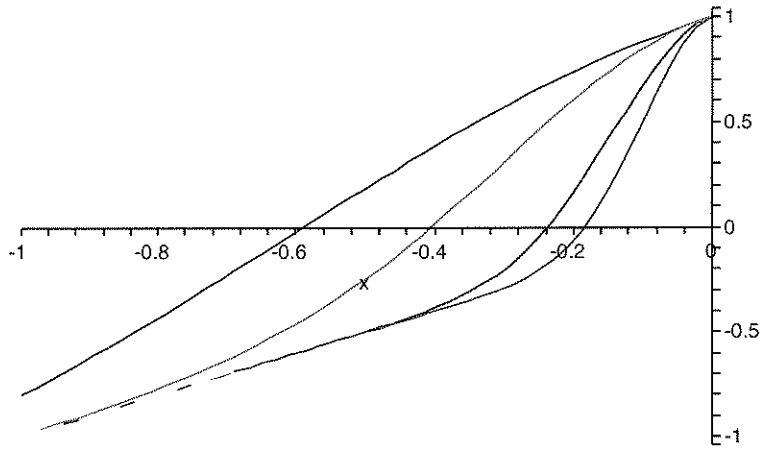
$$f(x) = x + e^{-Bx^2}\cos x$$

has a unique root on the interval $(-1, 0)$. Use the secant method with $x_0 = -1$ and $x_1 = 0$ to find this root as accurately as possible. Use values of B equal 1, 5, 10, 25 and 50. Explain the behavior observed in the iterates for the larger values of B.

The Maple script

```
restart;
f:=x+exp(-B*x^2)*cos(x);
Bn:=[1,5,10,25,50];
for n from 1 to 5
do
    fn[n]:=subs(B=Bn[n],f);
end;
s:=seq(fn[j],j=1..5);
plot([s],x=-1..0);
```

generates the plots



Note that the function looks more and more like the letter S as B increases. In particular, $f'(0) \to 0$ as $B \to \infty$. Thus, as B increases the derivative at 0 vanishes. This poses a problem for Newton's method where the iterates starting at $x_0 = 0$ fail to converge for large value of B. For the secant method, however, the derivative is approximated in such a way that the iterates converge for all values of B.

§3.3#3

The secant method `secant.m` is given by

```
function [x,n]=secant(f,x0,x1,epsilon)
    f0=feval(f,x0);
    for n=1:100
        f1=feval(f,x1);
        df=(f0-f1)/(x0-x1);
        x=x1-f1/df;
        if abs(x-x1)<= epsilon
            return;
        end
        x0=x1;
        f0=f1;
        x1=x;
    end
    fprintf('secant: Did not converge!\n');
```

The MATLAB/Octave script `s33p3.m` for this problem is

```
clear all
fprintf('Solution to Section 3.3 Problem 3\n\n');

global B;
function y=f(x)
    global B;
    y=x+exp(-B*x^2)*cos(x);
end
function y=df(x)
    global B;
    y=1-exp(-B*x^2)*(sin(x)+2*B*x*cos(x));
end


format long
epsilon=1E-16;
Bn=[1,5,10,25,50];
for n=1:5
    B=Bn(n);
    alpha=secant(@f,-1,0,epsilon);
    r=f(alpha);
    fprintf('B=%g, alpha=%g, f(alpha)=%g\n',...
Bn(n),alpha,r);
end
```

§3.3#3

The output is

```
Solution to Section 3.3 Problem 3

B=1, alpha=-0.588402, f(alpha)=-1.11022e-16
B=5, alpha=-0.404912, f(alpha)=5.55112e-17
B=10, alpha=-0.326402, f(alpha)=5.55112e-17
B=25, alpha=-0.237436, f(alpha)=2.77556e-17
B=50, alpha=-0.183291, f(alpha)=-2.77556e-17
```

The residual values $f(\alpha)$ are all around 10^{-17}. This indicates that we have calculated the roots of $f(\alpha)$ with quite reasonable accuracy using the secant method.

§3.3#6

Using the secant method find the root of

$$f(x) = x^4 - 5.4x^3 + 10.56x^2 - 8.954x + 2.7951$$

that lies in the interval $[1, 1.2]$ using Newton's method. Experiment with different choices of x_0 and x_1 and different ways of evaluating $f(x)$.

Use the secant method `secant.m` along with the script `s33p6.m` given by

```
clear all
fprintf('Solution to Section 3.3 Problem 6\n');
f1=@(x) x^4-5.4*x^3+10.56*x^2-8.954*x+2.7951;
f2=@(x) 2.7951-8.954*x+10.56*x^2-5.4*x^3+x^4;
f3=@(x) (((x-5.4)*x+10.56)*x-8.954)*x+2.7951;

format long
epsilon=5e-6;
x0=[0.5,0.5,0.5];
x1=[1,2,1.2];
for n=1:3
    fprintf('\nInitial x0=%g, x1=%g\n',x0(n),x1(n));
    [r1,n1]=secant(f1,x0(n),x1(n),epsilon)
    [r2,n2]=secant(f2,x0(n),x1(n),epsilon)
    [r3,n3]=secant(f3,x0(n),x1(n),epsilon)
end
```

The output is

```
Solution to Section 3.3 Problem 6

Initial x0=0.5, x1=1
r1 = 1.09997845465087
n1 = 31
r2 = 1.09998670248476
n2 = 33
r3 = 1.09998332601016
n3 = 32

Initial x0=0.5, x1=2
r1 = 1.09998643601682
n1 = 35
r2 = 1.09998287075812
n2 = 34
r3 = 1.09998657061266
n3 = 35

Initial x0=0.5, x1=1.2
r1 = 1.10001939214365
n1 = 31
r2 = 1.10001458536967
n2 = 32
```

§3.3#6

```
r3 = 1.10000938585092
n3 = 33
```

Usually the secant method converges in 4–8 iterations; therefore, it is unusual that it is taking more than 30 iterations to converge in this case. As discussed earlier, the graph appearing in the solution to §3.1#8 shows that α is most likely a root of multiplicity three. Therefore, the secant method will not converge with the power $(\sqrt{5}+1)/2 \approx 1.62$ as indicated in equations (3.31) in the text. Generally when dealing with a root of single multiplicity, Newton's method converges faster than the secant method. The results here compared to those in §3.2#11 indicate that secant method converges slower than Newton's method in this case as well.

§3.4 #6 Convert the equation $x^2 - 5 = 0$ into the fixed point problem

$$x = x + c(x^2 - 5) \equiv g(x)$$

with $c \neq 0$. Determine the posible values of c so that $x_{n+1} = g(x_n)$ converges to $x = \sqrt{5}$.

Need $g(x)$ to be a contraction near $\sqrt{5}$. Thus we want $|g'(\sqrt{5})| < 1$. Compute

$$g'(x) = 1 + 2cx$$

Therefore

$$|g'(\sqrt{5})| = |1 + 2c\sqrt{5}| < 1.$$

$$-1 < 1 + 2c\sqrt{5} < 1$$

$$-2 < 2c\sqrt{5} < 0$$

$$\frac{-1}{\sqrt{5}} < c < 0$$

Therefore any value of $c \in \left(\frac{-1}{\sqrt{5}}, 0\right)$ will result in a contraction around $\sqrt{5}$ and can be used as a possible value so the iterations converge.

§3.4#9 Consider the rootfinding problem $f(x)=0$ with root α and $f'(\alpha)\neq 0$. Convert it into the fixed-point problem

$$x = x + cf(x) \equiv g(x)$$

with c a nonzero constant. How should c be chosen to ensure rapid convergence of $x_{n+1} = x_n + cf(x_n)$ to α? Apply your technique to $x^3 - 5 = 0$.

We need $|g'(\alpha)| < 1$ for any convergence. For the fastest convergence we want $|g'(\alpha)|$ to be as small as possible. Thus we set $g'(\alpha) = 0$ and solve for c.

$$g'(\alpha) = 1 + cf'(\alpha) = 0$$

implies $c = -1/f'(\alpha)$ is the optimal value.

Example $f(x) = x^3 - 5$ and $\alpha = \sqrt[3]{5}$. Thus

$$c = \frac{-1}{f'(\alpha)} = \frac{-1}{3\alpha^2} = -\frac{5^{-2/3}}{3}.$$

and the fixed point iteration becomes

$$x_{n+1} = x - \frac{5^{-2/3}}{3}(x^3 - 5)$$

§3.4#9

The MATLAB/Octave script `s34p9.m` is

```
clear all
fprintf('Solution to Section 3.4 Problem 9\n\n');
g=@(x) x- 5^(-2/3)/3*(x^3-5);
format long

x=1
for n=1:7
x=g(x)
end
```

and the output is

```
Solution to Section 3.4 Problem 9

x = 1
x = 1.45599358578045
x = 1.67411971089698
x = 1.70922933775092
x = 1.70997562074009
x = 1.70997594667663
x = 1.70997594667670
x = 1.70997594667670
```

The convergence is very fast. Note, however, that the optimal choice of c requires computing $5^{2/3}$ which is essentially as difficult as finding $5^{1/3}$ for the original problem.

§3.7 #18 What is the order of convergence of the iteration

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

as it converges to the fixed point $x=\sqrt{a}$?

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

$$= \left|\frac{x_n(x_n^2+3a) - \sqrt{a}(3x_n^2+a)}{3x_n^2+a}\right|$$

$$= \left|\frac{(x_n-\sqrt{a})(x_n^2+3a) - \sqrt{a}(3x_n^2+a-x_n^2-3a)}{3x_n^2+a}\right|$$

$$= \left|\frac{(x_n-\sqrt{a})(x_n^2+3a) - \sqrt{a}(2x_n^2-2a)}{3x_n^2+a}\right|$$

$$= \left|\frac{(x_n-\sqrt{a})(x_n^2+3a) - 2\sqrt{a}(x_n-\sqrt{a})(x_n+\sqrt{a})}{3x_n^2+a}\right|$$

$$= |x_n-\sqrt{a}| \left|\frac{x_n^2-3a-2\sqrt{a}x_n-2a}{3x_n^2+a}\right|$$

$$= |x_n-\sqrt{a}| \left|\frac{x_n^2-a-2\sqrt{a}x_n}{3x_n^2+a}\right| = \frac{|x_n-\sqrt{a}|^3}{|3x_n^2+a|}$$

Therefore it converges cubically to the fixed point.

§**3.5#1**

Use Newton's method to calculate the roots of

$$f(x) = x^5 + 0.9x^4 - 1.62x^3 - 1.458x^2 + 0.6561x + 0.59049.$$

Print out the iterates and the function values. Produce the ratios

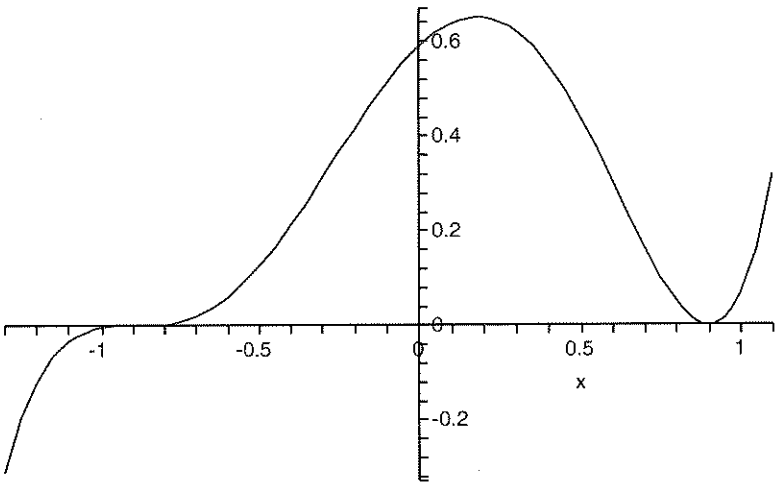
$$\frac{\alpha - x_n}{\alpha - x_{n-1}} \approx \frac{x_{n+1} - x_n}{x_n - x_{n-1}}.$$

Repeat the problem for several choices of x_0. Make observations that seem important relative to the root-finding problem.

The Maple script

```
1 restart;
2 p:=x^5+0.9*x^4-1.62*x^3-1.458*x^2+0.6561*x+0.59049;
3 plot(p,x=-1.3..1.1);
4 interface(prettyprint=0);
5 diff(p,x);
6 diff(p,x$2);
7 diff(p,x$3);
```

yields the plot



and the derivatives

$$f'(x) = 5x^4 + 3.6x^3 - 4.86x^2 - 2.916x + 0.6561$$
$$f''(x) = 20x^3 + 10.8x^2 - 9.72x - 2.916$$
$$f'''(x) = 60x^2 + 21.6x - 9.72$$

from which we see there is likely to be a root of multiplicity three in the interval $[-1, -0.5]$ and a root of multiplicity 2 in the interval $[0.5, 1]$.

§3.5#1

We use `newtonr.m` which is a version of `newton.m` that has been modified to print out the ratios for λ. This file is given as

```
function [x,n,r]=newtonr(f,df,x0,epsilon)
    xm1=x0-1;
    for n=1:100
        f0=feval(f,x0);
        x=x0-f0/feval(df,x0);
        r=(x-x0)/(x0-xm1);
        fprintf('    n=%g, f(x0)=%g, xn=%g, lambda=%g\n',...
            n,f0,x,r);
        if abs(x-x0)<=epsilon
            return;
        end
        xm1=x0;
        x0=x;
    end
    fprintf('newton: Did not converge!\n');
```

The script `s35p1.m` calculates the multiplicity of the root and then computes the exact root using the appropriate derivatives to obtain quadratic convergence.

```
clear all
fprintf('Solution to Section 3.5 Problem 1\n');

f=@(x) x^5+0.9*x^4-1.62*x^3-1.458*x^2+0.6561*x+0.59049;
df=@(x) 5*x^4+3.6*x^3-4.86*x^2-2.916*x+.6561;
ddf=@(x) 20*x^3+10.8*x^2-9.72*x-2.916;
dddf=@(x) 60*x^2+21.6*x-9.72;

m=2:5;
lambda=[0,(m-1)./m];

format long
xi=[0.75,-0.75];

for i=1:2
    fprintf('\nInitial value x0=%g\n',xi(i));
    [x,n,r]=newtonr(f,df,xi(i),1E-4);
    [z,iz]=min(abs(lambda-r));
    fprintf('Multiplicity is %g\n',iz);
end

fprintf('\nNow find the roots efficiently...\n');
[r1,n]=newton(df,ddf,xi(1),1E-15)
[r2,n]=newton(ddf,dddf,xi(2),1E-15)
```

§3.5#1

The output is

```
Solution to Section 3.5 Problem 1

Initial value x0=0.75
    n=1, f(x0)=0.101073, xn=0.836842, lambda=0.0868421
    n=2, f(x0)=0.0208995, xn=0.870243, lambda=0.384615
    n=3, f(x0)=0.00491224, xn=0.885506, lambda=0.456977
    n=4, f(x0)=0.00119575, xn=0.892842, lambda=0.480638
    n=5, f(x0)=0.000295224, xn=0.896443, lambda=0.490763
    n=6, f(x0)=7.33595e-05, xn=0.898227, lambda=0.495483
    n=7, f(x0)=1.82851e-05, xn=0.899115, lambda=0.497766
    n=8, f(x0)=4.5645e-06, xn=0.899558, lambda=0.498889
    n=9, f(x0)=1.14028e-06, xn=0.899779, lambda=0.499446
    n=10, f(x0)=2.84965e-07, xn=0.899889, lambda=0.499723
    n=11, f(x0)=7.12282e-08, xn=0.899945, lambda=0.499862
Multiplicity is 2

Initial value x0=-0.75
    n=1, f(x0)=0.00918844, xn=-0.803226, lambda=-0.0532258
    n=2, f(x0)=0.0026292, xn=-0.836754, lambda=0.629921
    n=3, f(x0)=0.000763095, xn=-0.85836, lambda=0.644433
    n=4, f(x0)=0.00022322, xn=-0.872463, lambda=0.652693
    n=5, f(x0)=6.56004e-05, xn=-0.881738, lambda=0.657691
    n=6, f(x0)=1.93344e-05, xn=-0.887867, lambda=0.660825
    n=7, f(x0)=5.70886e-06, xn=-0.89193, lambda=0.662832
    n=8, f(x0)=1.68765e-06, xn=-0.894628, lambda=0.664137
    n=9, f(x0)=4.9929e-07, xn=-0.896422, lambda=0.664991
    n=10, f(x0)=1.4779e-07, xn=-0.897616, lambda=0.665555
    n=11, f(x0)=4.37604e-08, xn=-0.898412, lambda=0.665928
    n=12, f(x0)=1.29603e-08, xn=-0.898941, lambda=0.666175
    n=13, f(x0)=3.83896e-09, xn=-0.899294, lambda=0.666339
    n=14, f(x0)=1.13725e-09, xn=-0.89953, lambda=0.666449
    n=15, f(x0)=3.36918e-10, xn=-0.899686, lambda=0.666522
    n=16, f(x0)=9.98186e-11, xn=-0.899791, lambda=0.666569
    n=17, f(x0)=2.95746e-11, xn=-0.899861, lambda=0.666611
Multiplicity is 3

Now find the roots efficiently...
r1 = 0.900000000000000
n = 7
r2 = -0.900000000000000
n = 7
```

Since the positive root has multiplicity 2 then using Newton's method find the roots of $f'(x)$ results in quadratic convergence to the root $\alpha = 0.9$. Since the negative root has multiplicity 3 then using Newton's method find the roots of $f''(x)$ results in quadratic convergence to the root $\alpha = -0.9$.

§3.5#2

Use Newton's method to calculate the roots of

$$f(x) = x^4 - 3.2x^3 + 0.96x^2 + 4.608x - 3.456.$$

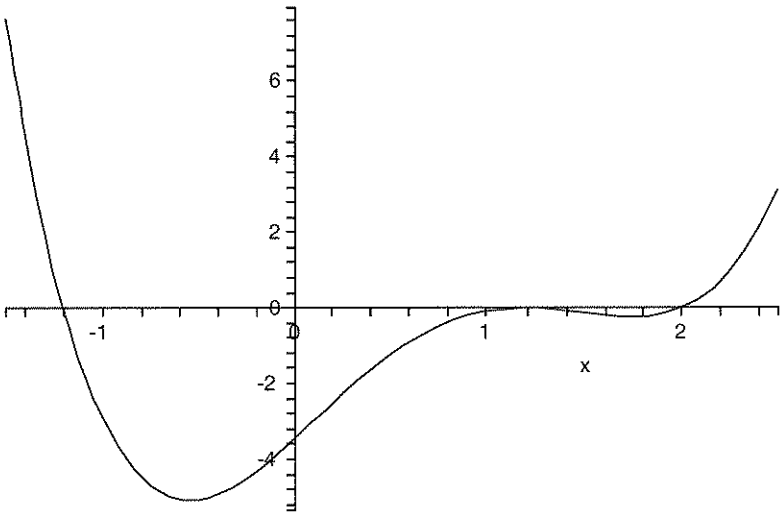
Print out the iterates and the function values. Produce the ratios

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

Repeat the problem for several choices of x_0. Make observations that seem important relative to the root-finding problem.

```
1 restart;
2 p:=x^4-3.2*x^3+0.96*x^2+4.608*x-3.456;
3 plot(p,x=-1.5..2.5);
4 interface(prettyprint=0);
5 diff(p,x);
6 diff(p,x$2);
```

yields the plot



and the derivatives

$$f'(x) = 4x^3 - 9.6x^2 + 1.92x + 4.608$$
$$f''(x) = 12x^2 - 19.2x + 1.92$$

from which we see there is likely to be a simple root in the interval $[-1.5, 1]$, a root of multiplicity 2 in the interval $[0.5, 1.5]$ and a simple root in the interval $[1.5, 2.5]$.

§3.5#2

We use `newtonr.m` with the script `s35p2.m` given by

```
clear all
fprintf('Solution to Section 3.5 Problem 1\n');

f=@(x) x^4-3.2*x^3+0.96*x^2+4.608*x-3.456;
df=@(x) 4*x^3-9.6*x^2+1.92*x+4.608;
ddf=@(x) 12*x^2-19.2*x+1.92;

m=2:5;
lambda=[0,(m-1)./m];

format long
xi=[-1,0.5,2.25];

for i=1:3
    fprintf('\nInitial value x0=%g\n',xi(i));
    [x,n,r]=newtonr(f,df,xi(i),1E-4);
    [z,iz]=min(abs(lambda-r));
    fprintf('Multiplicity is %g\n',iz);
end

fprintf('\nNow find the roots efficiently...\n');
[r1,n]=newton(f,df,xi(1),1E-15)
[r2,n]=newton(df,ddf,xi(2),1E-15)
[r3,n]=newton(f,df,xi(3),1E-15)
```

with output

```
Solution to Section 3.5 Problem 1

Initial value x0=-1
    n=1, f(x0)=-2.904, xn=-1.26613, lambda=-0.266129
    n=2, f(x0)=1.31358, xn=-1.20455, lambda=-0.23139
    n=3, f(x0)=0.0842907, xn=-1.20002, lambda=0.0734943
    n=4, f(x0)=0.000434086, xn=-1.2, lambda=0.00520342
Multiplicity is 1

Initial value x0=0.5
    n=1, f(x0)=-1.2495, xn=0.840649, lambda=0.340649
    n=2, f(x0)=-0.305507, xn=1.00905, lambda=0.494363
    n=3, f(x0)=-0.0798146, xn=1.09971, lambda=0.538331
    n=4, f(x0)=-0.0208242, xn=1.14821, lambda=0.534994
    n=5, f(x0)=-0.00536464, xn=1.17361, lambda=0.523747
    n=6, f(x0)=-0.00136571, xn=1.18667, lambda=0.514023
    n=7, f(x0)=-0.000344876, xn=1.1933, lambda=0.507664
    n=8, f(x0)=-8.66778e-05, xn=1.19664, lambda=0.504014
    n=9, f(x0)=-2.17287e-05, xn=1.19832, lambda=0.502055
    n=10, f(x0)=-5.43968e-06, xn=1.19916, lambda=0.50104
    n=11, f(x0)=-1.36087e-06, xn=1.19958, lambda=0.500523
```

```
    n=12, f(x0)=-3.40336e-07, xn=1.19979, lambda=0.500262
    n=13, f(x0)=-8.5099e-08, xn=1.19989, lambda=0.500131
    n=14, f(x0)=-2.12766e-08, xn=1.19995, lambda=0.500066
Multiplicity is 2

Initial value x0=2.25
    n=1, f(x0)=0.950906, xn=2.08857, lambda=-0.16143
    n=2, f(x0)=0.229972, xn=2.01634, lambda=0.447411
    n=3, f(x0)=0.0350314, xn=2.00071, lambda=0.216516
    n=4, f(x0)=0.001448, xn=2, lambda=0.0450333
    n=5, f(x0)=2.86e-06, xn=2, lambda=0.00198297
Multiplicity is 1

Now find the roots efficiently...
r1 = -1.20000000000000
n = 6
r2 = 1.20000000000000
n = 7
r3 = 2.00000000000000
n = 7
```

The results are similar to the previous problem. We are able to obtain quadratic convergence to the root of multiplicity two by using Newton's method to solve $f'(x) = 0$.