COMP 350 Solution of Assignment #4

1. (a) Since r is a root of $f(x) = 0$ of multiplicity m, $f(r) = f'(r) = \ldots = f^{(m-1)}(r) = 0$. Thus by the Taylor series, we have

$$\begin{aligned}
f(x_n) &= f(r) + f'(r)(x_n - r) + \cdots + f^{(m-1)}(r)\frac{(x_n - r)^{m-1}}{(m-1)!} + f^{(m)}(r)\frac{(x_n - r)^m}{m!} \\
&\quad + f^{(m+1)}(z_n)\frac{(x_n - r)^{m+1}}{(m+1)!} \qquad \text{for some } z_n \text{ between } x_n \text{ and } r \\
&= f^{(m)}(r)\frac{(x_n - r)^m}{m!} + f^{(m+1)}(z_n)\frac{(x_n - r)^{m+1}}{(m+1)!} \\
f'(x_n) &= f'(r) + f''(r)(x_n - r) + \cdots + f^{(m-1)}(r)\frac{(x_n - r)^{m-2}}{(m-2)!} + f^{(m)}(r)\frac{(x_n - r)^{m-1}}{(m-1)!} \\
&\quad + f^{(m+1)}(\tilde{z}_n)\frac{(x_n - r)^m}{m!} \qquad \text{for some } \tilde{z}_n \text{ between } x_n \text{ and } r \\
&= f^{(m)}(r)\frac{(x_n - r)^{m-1}}{(m-1)!} + f^{(m+1)}(\tilde{z}_n)\frac{(x_n - r)^m}{m!}
\end{aligned}$$

Then

$$\begin{aligned}
x_{n+1} - r &= x_n - r - m\frac{f(x_n)}{f'(x_n)} \\
&= x_n - r - m\frac{f^{(m)}(r)\frac{(x_n-r)^m}{m!} + f^{(m+1)}(z_n)\frac{(x_n-r)^{m+1}}{(m+1)!}}{f^{(m)}(r)\frac{(x_n-r)^{m-1}}{(m-1)!} + f^{(m+1)}(\tilde{z}_n)\frac{(x_n-r)^m}{m!}} \\
&= \frac{\frac{f^{(m+1)}(\tilde{z}_n)}{m} - \frac{f^{(m+1)}(z_n)}{m+1}}{f^{(m)}(r) + \frac{f^{(m+1)}(\tilde{z}_n)(x_n-r)}{m}}(x_n - r)^2
\end{aligned}$$

Since z_n and $\tilde{z}_n$ are between x_n and r, when $x_n \to r$, $z_n \to r$ and $\tilde{z}_n \to r$. Therefore

$$\lim_{n\to\infty} \frac{|x_{n+1} - r|}{|x_n - r|^2} = \left| \frac{\frac{f^{(m+1)}(r)}{m} - \frac{f^{(m+1)}(r)}{m+1}}{f^{(m)}(r)} \right| = \left| \frac{f^{(m+1)}(r)}{(m+1)mf^{(m)}(r)} \right|.$$

So the modified Newton's method has quadratic convergence rate.

(b) (4 points) Given x_0, write

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2 + \frac{1}{6}f'''(z)(x - x_0)^3.$$

We use $q(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2$ as an approximation to $f(x)$. Set $q(x) = 0$ and let one of the roots be denoted by x_1:

$$x_1 = x_0 - \frac{f'(x_0)}{f''(x_0)} + \frac{\sqrt{(f'(x_0))^2 - 2f(x_0)f''(x_0)}}{f''(x_0)}.$$

This x_1 can be used as an approximate root. We repeat the above step until the sequence convergence. In general, the iteration formula has the following form:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)} + \frac{\sqrt{(f'(x_n))^2 - 2f(x_n)f''(x_n)}}{f''(x_n)}.$$

2. (10 points)

(a) (2 points)

```
% secant.m
function root = secant(fname,x0,x1,xtol,ftol,n_max,display)
% Secant Method
%
% input:  fname is a string that names the function f(x).
%         x0 and x1 are the initial points
%         xtol and ftol are termination tolerances
%         n_max is the maximum number of iteration
%         display = 1 if step-by-step display is desired,
%                 = 0 otherwise
% output: root is the computed root of f(x)=0
%
n = 0;
fx1 = feval(fname,x1);
if display,
    disp('   n         x0              x1              f(x1)')
    disp(sprintf('%4d %23.15e %23.15e %23.15e', n, x0, x1, fx1))
end
if abs(fx1)<=ftol
    root=x1;
    return
end
for n=1:n_max
    fx0 = feval(fname,x0);
    fx1 = feval(fname,x1);
    d = (x1-x0)/(fx1-fx0)*fx1;
    x0 = x1;
    fx0 = fx1;
    x1 = x1 - d;
    if display,
        disp(sprintf('%4d %23.15e %23.15e %23.15e', n, x0, x1, fx1))
    end
    if abs(d)<=xtol || abs(fx1)<=ftol
        root = x1;
        return;
    end
end
root = x1;
```

(b) (2 points)

```
function root = newnewton(fname,fdname,fd2name,x,xtol,ftol,n_max,display)
% New Newton's Method.
%
% input:  fname is a string that names the function f(x).
%         fdname is a string that names the derivative f'(x).
%         fd2name is a string that names the derivative f''(x).
%         x is the initial point
%         xtol and ftol are termination tolerances
%         n_max is the maximum number of iteration
%         display = 1 if step-by-step display is desired,
%                 = 0 otherwise
% output: root is the computed root of f(x)=0
%
n = 0;
fx = feval(fname,x);
```

```
if display,
   disp('   n             x                    f(x)')
   disp('------------------------------------------------------')
   disp(sprintf('%4d %23.15e %23.15e', n, x, fx))
end
if abs(fx) <= ftol
   root = x;
   return
end
for n = 1:n_max
    fdx = feval(fdname,x);
fd2x = feval(fd2name,x);
    d = fdx/fd2x - sqrt(fdx*fdx - 2*fx*fd2x)/fd2x;
    x = x - d;
    fx = feval(fname,x);
    if display,
       disp(sprintf('%4d %23.15e %23.15e', n, x, fx)), end
    if abs(d) <= xtol | abs(fx) <= ftol
       root = x;
       return
    end
end
root = x;
\end{enumerate}
```

(c) (1 point)

Program files

```
% f.m
function y = f(x)
y = x^3 - 5*x + 3;

% fd.m
function y = fd(x)
y = 3*x^2 - 5;

% fdd.m
function y = fd(x)
y = 6x;
```

(d) (1 point)

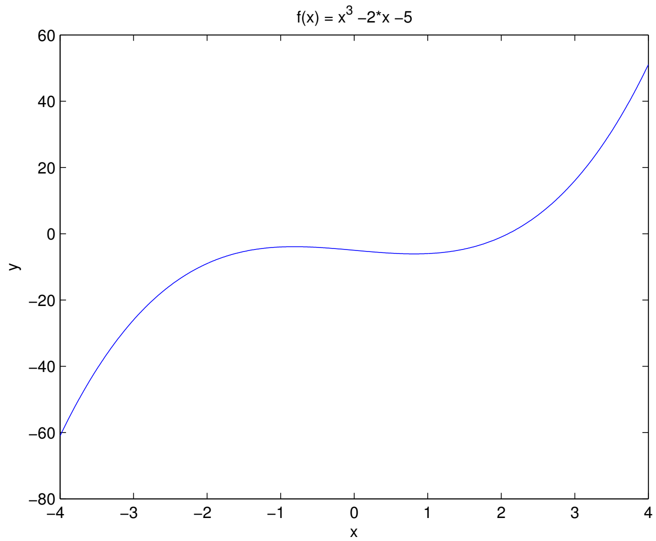
Print the graph of $f(x) = x^3 - 5 * x + 3$
The program to print the graph:

```
x = [-4:0.01:4];
y = x^3 - 5*x + 3;
plot(x,y)
title('f(x) = x^3 - 5*x + 3')
xlabel('x')
ylabel('y')
```

(e) (2 points)

Compute the root by four methods
Here is the computed results and each intermediate results of three methods.
(Use bisection.m and newton.m from course website, see
http://www.cs.mcgill.ca/∼chang/teaching/cs350/doc.php)

```
>> bisection('f',1,3,1.e-12,1);
       a             b             c            f(c)      error_bound
1.00000e+000 3.00000e+000 2.00000e+000   1.00000e+000  1.000000000000000e+000
```



1.00000e+000	2.00000e+000	1.50000e+000	-1.12500e+000	5.000000000000000e-001
1.50000e+000	2.00000e+000	1.75000e+000	-3.90625e-001	2.500000000000000e-001
1.75000e+000	2.00000e+000	1.87500e+000	2.16797e-001	1.250000000000000e-001
1.75000e+000	1.87500e+000	1.81250e+000	-1.08154e-001	6.250000000000000e-002
1.81250e+000	1.87500e+000	1.84375e+000	4.89197e-002	3.125000000000000e-002
1.81250e+000	1.84375e+000	1.82813e+000	-3.09563e-002	1.562500000000000e-002
1.82813e+000	1.84375e+000	1.83594e+000	8.64553e-003	7.812500000000000e-003
1.82813e+000	1.83594e+000	1.83203e+000	-1.12392e-002	3.906250000000000e-003
1.83203e+000	1.83594e+000	1.83398e+000	-1.31784e-003	1.953125000000000e-003
1.83398e+000	1.83594e+000	1.83496e+000	3.65860e-003	9.765625000000000e-004
1.83398e+000	1.83496e+000	1.83447e+000	1.16907e-003	4.882812500000000e-004
1.83398e+000	1.83447e+000	1.83423e+000	-7.47115e-005	2.441406250000000e-004
1.83423e+000	1.83447e+000	1.83435e+000	5.47097e-004	1.220703125000000e-004
1.83423e+000	1.83435e+000	1.83429e+000	2.36172e-004	6.103515625000000e-005
1.83423e+000	1.83429e+000	1.83426e+000	8.07252e-005	3.051757812500000e-005
1.83423e+000	1.83426e+000	1.83424e+000	3.00559e-006	1.525878906250000e-005
1.83423e+000	1.83424e+000	1.83424e+000	-3.58533e-005	7.629394531250000e-006
1.83424e+000	1.83424e+000	1.83424e+000	-1.64239e-005	3.814697265625000e-006
1.83424e+000	1.83424e+000	1.83424e+000	-6.70919e-006	1.907348632812500e-006
1.83424e+000	1.83424e+000	1.83424e+000	-1.85181e-006	9.536743164062500e-007
1.83424e+000	1.83424e+000	1.83424e+000	5.76887e-007	4.768371582031250e-007
1.83424e+000	1.83424e+000	1.83424e+000	-6.37460e-007	2.384185791015625e-007
1.83424e+000	1.83424e+000	1.83424e+000	-3.02866e-008	1.192092895507813e-007
1.83424e+000	1.83424e+000	1.83424e+000	2.73300e-007	5.960464477539063e-008
1.83424e+000	1.83424e+000	1.83424e+000	1.21507e-007	2.980232238769531e-008
1.83424e+000	1.83424e+000	1.83424e+000	4.56102e-008	1.490116119384766e-008
1.83424e+000	1.83424e+000	1.83424e+000	7.66178e-009	7.450580596923828e-009
1.83424e+000	1.83424e+000	1.83424e+000	-1.13124e-008	3.725290298461914e-009
1.83424e+000	1.83424e+000	1.83424e+000	-1.82531e-009	1.862645149230957e-009
1.83424e+000	1.83424e+000	1.83424e+000	2.91824e-009	9.313225746154785e-010
1.83424e+000	1.83424e+000	1.83424e+000	5.46464e-010	4.656612873077393e-010
1.83424e+000	1.83424e+000	1.83424e+000	-6.39423e-010	2.328306436538696e-010
1.83424e+000	1.83424e+000	1.83424e+000	-4.64802e-011	1.164153218269348e-010
1.83424e+000	1.83424e+000	1.83424e+000	2.49992e-010	5.820766091346741e-011
1.83424e+000	1.83424e+000	1.83424e+000	1.01756e-010	2.910383045673370e-011
1.83424e+000	1.83424e+000	1.83424e+000	2.76383e-011	1.455191522836685e-011
1.83424e+000	1.83424e+000	1.83424e+000	-9.42091e-012	7.275957614183426e-012
1.83424e+000	1.83424e+000	1.83424e+000	9.10916e-012	3.637978807091713e-012

```
1.83424e+000 1.83424e+000 1.83424e+000  -1.56319e-013  1.818989403545857e-012
1.83424e+000 1.83424e+000 1.83424e+000   4.47642e-012  9.094947017729282e-013

ans =

    1.8342

>> newton('f','fd',2,1.e-12,1.e-12,100,1)
   n              x                     f(x)
------------------------------------------------------
   0  2.000000000000000e+000  1.000000000000000e+000
   1  1.857142857142857e+000  1.195335276967926e-001
   2  1.834787350054526e+000  2.773253015902810e-003
   3  1.834243503918507e+000  1.627856713426468e-006
   4  1.834243184314032e+000  5.622169396701793e-013

ans =

    1.8342

>> secant('f',1,2,1.e-12,1.e-12,100,1)
   n              x0              x1                     f(x1)
   0  1.000000000000000e+000  2.000000000000000e+000  1.000000000000000e+000
   1  2.000000000000000e+000  1.500000000000000e+000  1.000000000000000e+000
   2  1.500000000000000e+000  1.764705882352941e+000 -1.125000000000000e+000
   3  1.764705882352941e+000  1.873599540361965e+000 -3.279055566863436e-001
   4  1.873599540361965e+000  1.831205833909406e+000  2.090397299390610e-001
   5  1.831205833909406e+000  1.834118127083818e+000 -1.541953360163717e-002
   6  1.834118127083818e+000  1.834243595856441e+000 -6.368734578741098e-004
   7  1.834243595856441e+000  1.834243184258313e+000  2.096128623563232e-006
   8  1.834243184258313e+000  1.834243184313922e+000 -2.832365453286911e-010
   9  1.834243184313922e+000  1.834243184313922e+000 -1.776356839400251e-015

ans =

    1.8342

>> newnewton('f','fd','fd2',2,1.e-12,1.e-12,100,1)
   n              x                     f(x)
------------------------------------------------------
   0  2.000000000000000e+000  1.000000000000000e+000
   1  1.833333333333333e+000 -4.629629629629761e-003
   2  1.834243184461801e+000  7.532010570798775e-010
   3  1.834243184313922e+000  8.881784197001252e-016

ans =

    1.8342
```

(f) (2 points)

Speed of convergence
New method: cubic
Newton method : quadratic
Secant method : superlinear
Bisection method : linear