Assignment 4 - Nonlinear equations
COMP 350 - Numerical Computing
Prof. Chang Xiao-Wen
Fall 2018

LE, Nhat Hung
McGill ID: 260793376
Date: October 26, 2018
Due date: October 30, 2018

1. (4 points) In class, we showed that if Newton's iteration converges to r, a root of $f(x) = 0$, then usually it has quadratic convergence, i.e., $lim_{n\to\infty} |x_{n+1} - r| / |x_n - r|^2 = c$, where $c \neq 0$ is a constant. From a numerical experiment on $f(x) = x^2 - 2$, we found after $|f(x_n)|$ is small enough, $|f(x_n)|$ is squared every step. In fact it is usually true for a general nonlinear equation. Suppose f, f' and f'' are continuous. Prove that if x_n converges to a root, $f(x_n)$ usually converges to 0 with quadratic convergence, i.e.,

$$\lim_{n\to\infty} |f(x_{n+1})|/|f(x_n)|^2 = c$$

for a non zero constant c.

Note: Use the Taylor series theory in your analysis. The proof is not difficult.

$$f(x_{n+1}) = f(x_n) + f'(x_n)(x_{n+1} - x_n) + \frac{f''(z)}{2}(x_{n+1} - x_n)^2$$
$$f(x_{n+1}) = f(x_n) + f'(x_n)\left(-\frac{f(x_n)}{f'(x_n)}\right) + \frac{f''(z)}{2}\left(-\frac{f(x_n)}{f'(x_n)}\right)^2$$
$$\frac{f(x_{n+1})}{(f(x_n))^2} = -\frac{f''(z)}{2(f'(x_n))^2}$$
$$\lim_{n\to\infty} \frac{f(x_{n+1})}{(f(x_n))^2} = c, \quad c \neq 0$$

2. (6 points) In class, we derived Newton's method by using the first two terms in the Taylor series. Derive a new method by using the first three terms in the Taylor series in a similar way.

Taylor expansion:

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

Then,

$$0 = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)}{2}(x - x_0)^2$$
$$x - x_0 = \frac{-f'(x_0) \pm \sqrt{(f'(x_0))^2 - (4)(\frac{f''(x_0)}{2})f(x_0)}}{f''(x_0)}$$
$$x - x_0 = -\frac{f'(x_0)}{f''(x_0)} \pm \frac{\sqrt{(f'(x_0))^2 - 2f''(x_0)f(x_0)}}{f''(x_0)}$$

Because we want $x - x_0$ as close to 0 as possible, we will take the plus sign:

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

$$x - x_0 = -\frac{f'(x_0)}{f''(x_0)} + \frac{f'(x_0)\sqrt{1 - \frac{2f''(x_0)f(x_0)}{(f'(x_0))^2}}}{f''(x_0)}$$

$$x - x_0 = -\frac{f'(x_0)}{f''(x_0)}\left(1 - \sqrt{1 - \frac{2f''(x_0)f(x_0)}{(f'(x_0))^2}}\right)$$

The Maclaurin expansion of $\sqrt{1-x}$

$$\sqrt{1-x} \approx 1 - \frac{x}{2} - \frac{x^2}{8}$$

can be used to simplify the square root term above:

$$\sqrt{1 - \frac{2f''(x_0)f(x_0)}{(f'(x_0))^2}} = 1 - \frac{f''(x_0)f(x_0)}{(f'(x_0))^2} - \frac{(f''(x_0))^2(f(x_0))^2}{2(f'(x_0))^4}$$

Then,

$$x - x_0 = -\frac{f'(x_0)}{f''(x_0)}\left(1 - \left(1 - \frac{f''(x_0)f(x_0)}{(f'(x_0))^2} - \frac{(f''(x_0))^2(f(x_0))^2}{2(f'(x_0))^4}\right)\right)$$

$$x - x_0 = -\frac{f'(x_0)}{f''(x_0)}\left(\frac{f''(x_0)f(x_0)}{(f'(x_0))^2} + \frac{(f''(x_0))^2(f(x_0))^2}{2(f'(x_0))^4}\right)$$

$$x - x_0 = -\frac{f(x_0)}{f'(x_0)} - \frac{f''(x_0)(f(x_0))^2}{2(f'(x_0))^3}$$

$$x = x_0 - \frac{f(x_0)}{f'(x_0)} - \frac{f''(x_0)(f(x_0))^2}{2(f'(x_0))^3}$$

Finally, we have

$$x_{x+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{f''(x_n)(f(x_n))^2}{2(f'(x_n))^3}$$

or

$$x_{x+1} = x_n - \frac{f(x_n)}{f'(x_n)}\left(1 + \frac{f''(x_n)f(x_n)}{2(f'(x_n))^2}\right)$$

(Bonus 5 points) Show usually the new method has cubic convergence.

Let r root of f. Then, from Taylor's theorem we have:

$$f(r) = 0 = f(x_n) + f'(x_n)(r - x_n) + \frac{f''(x_n)}{2}(r - x_n)^2 + \frac{f'''(z_1)}{6}(r - x_n)^3 \quad (1)$$

$$f(r) = 0 = f(x_n) + f'(x_n)(r - x_n) + \frac{f''(z_2)}{2}(r - x_n)^2 \quad (2)$$

$(1) \leftarrow (1) \times (-2(f'(x_n))^2 - f''(x_n)f(x_n)) :$

$$\begin{aligned} 0 = \quad & -2(f'(x_n))^2 f(x_n) - f''(x_n)(f(x_n))^2 \\ & +(-2(f'(x_n))^3 - f''(x_n)f'(x_n)f(x_n))(r - x_n) \\ & +\frac{-2f''(x_n)(f'(x_n))^2 - (f''(x_n))^2 f(x_n)}{2}(r - x_n)^2 \\ & +\frac{-2f'''(z_1)(f'(x_n))^2 - f'''(z_1)f''(x_n)f(x_n)}{6}(r - x_n)^3 \quad (1) \end{aligned}$$

$(2) \leftarrow (2) \times (-f''(x_n)f'(x_n)(r - x_n)) :$

$$\begin{aligned} 0 = \quad & -f''(x_n)f'(x_n)f(x_n)(r - x_n) \\ & -f''(x_n)(f'(x_n))^2(r - x_n)^2 \\ & -\frac{f''(z_2)f''(x_n)f'(x_n)}{2}(r - x_n)^3 \quad (2) \end{aligned}$$

$(1) - (2) :$

$$\begin{aligned} 0 = \quad & -2(f'(x_n))^2 f(x_n) - f''(x_n)(f(x_n))^2 \\ & -2(f'(x_n))^3(r - x_n) \\ & -\frac{(f''(x_n))^2 f(x_n)}{2}(r - x_n)^2 \\ & +\frac{-2f'''(z_1)(f'(x_n))^2 - f'''(z_1)f''(x_n)f(x_n) + 3f''(z_2)f''(x_n)f'(x_n)}{6}(r-x_n)^3 \end{aligned}$$

$$\begin{aligned} & -\frac{2(f'(x_n))^2 f(x_n) + f''(x_n)(f(x_n))^2}{2(f'(x_n))^3} \\ & -\frac{(f''(x_n))^2 f(x_n)}{4(f'(x_n))^3}(r - x_n)^2 \\ & -\frac{2f'''(z_1)(f'(x_n))^2 + f'''(z_1)f''(x_n)f(x_n) - 3f''(z_2)f''(x_n)f'(x_n)}{12(f'(x_n))^3}(r - x_n)^3 \end{aligned}$$

$$\begin{aligned} & x_n - \frac{f(x_n)}{f'(x_n)} - \frac{f''(x_n)(f(x_n))^2}{2(f'(x_n))^3} \\ & -\frac{(f''(x_n))^2 f(x_n)}{4(f'(x_n))^3}(r - x_n)^2 \\ & -\frac{2f'''(z_1)(f'(x_n))^2 + f'''(z_1)f''(x_n)f(x_n) - 3f''(z_2)f''(x_n)f'(x_n)}{12(f'(x_n))^3}(r - x_n)^3 \end{aligned}$$

$$x_{n+1} - \frac{(f''(x_n))^2 f(x_n)}{4(f'(x_n))^3}(r - x_n)^2$$

$$-\frac{2f'''(z_1)(f'(x_n))^2+f'''(z_1)f''(x_n)f(x_n)-3f''(z_2)f''(x_n)f'(x_n)}{12(f'(x_n))^3}(r-x_n)^3$$

$$-\frac{(f''(x_n))^2f(x_n)}{4(f'(x_n))^3}(r-x_n)^2$$
$$-\frac{2f'''(z_1)(f'(x_n))^2+f'''(z_1)f''(x_n)f(x_n)-3f''(z_2)f''(x_n)f'(x_n)}{12(f'(x_n))^3}(r-x_n)^3$$

We know

$$\lim_{x_n->r} f(x_n)=0$$

Therefore, as x_n converges towards r,

$$r-x_{n+1}=-\frac{2f'''(z_1)f'(x_n)-3f''(z_2)f''(x_n)}{12(f'(x_n))^2}(r-x_n)^3$$
$$\frac{r-x_{n+1}}{(r-x_n)^3}=-\frac{2f'''(z_1)f'(x_n)-3f''(z_2)f''(x_n)}{12(f'(x_n))^2}$$
$$\lim_{n\to\infty}\frac{r-x_{n+1}}{(r-x_n)^3}=c,\quad c\neq 0$$

In conclusion, the modified Newton's method has cubic convergence.

3. (10 points) Write a Matlab program `secant.m` for the secant method. Suppose we want to find the largest positive root of $f(x) = x^3 - 5x + 3$. Plot the graph of $y = f(x)$ on an appropriate interval by Matlab (check how to use Matlab's built-in function `plot`). Use your `secant.m` to compute the root. Also use the bisection method, the Newton method, and the new method you derived in question 2 to find the root. For the bisection method, use $[1, 3]$ as the initial interval, for the Newton method, use $x_0 = 2$ as the initial point, for the secant method, use $x_0 = 1$ and $x_1 = 2$ as the two initial points, and for the new method, use $x_0 = 2$ as the initial point. You can choose any appropriate initial points and initial interval, Take tolerances `xtol=1.e-12` and `ftol=1.e-12` for Newton's method, the new method, and the secant method, and take `delta=1.e-12` for the bisection method. Set a big number for the maximum number of iterations of the secant method and Newton's method such that the iteration stops only when `xtol=1.e-12` or `ftol=1.e-12` is satisfied. Comment on the speeds of convergence of these four methods. Print out the graph of $y = f(x)$ and the commands you used to plot the graph, your program `secant.m`, and other M-files related to $f(x)$. Also print out the results of each iteration step. You can use M-files `newton.m` and `bisection.m` on the course website.

First, the main file containing commands used to plot the graph and to run all the relevant methods:

ass4.m:

```
fn  = @(x) x.*x.*x - 5.*x + 3;
fd  = @(x) 3.*x.*x - 5;
fd2 = @(x) 6.*x

x = -3:1/100:3;
plot(x,fn(x));

disp('Secant method:');
root = secant(fn,1,2,1.e-12,1.e-12,100,1);

disp(' ');
disp('Bisection method:');
root = bisection(fn,1,3,1.e-12,1);

disp(' ');
disp('Newton''s method:');
root = newton(fn,fd,2,1.e-12,1.e-12,100,1);

disp(' ');
disp('Cubic Newton''s method:');
root = newtonCubic(fn,fd,fd2,2,1.e-12,1.e-12,100,1);
```

The max number of iterations for all methods are 100.

This gives the graph:

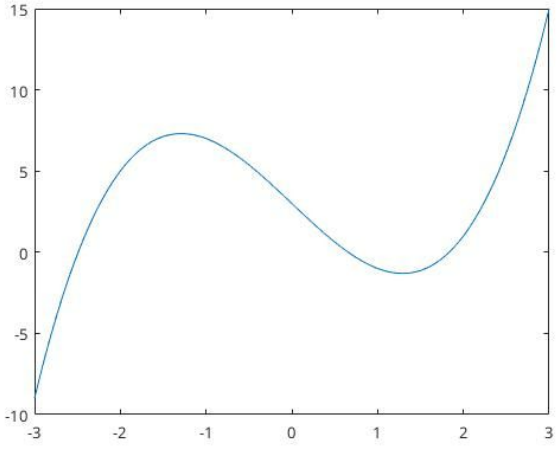

Figure 1: $f(x) = x^3 - 5x + 3,\ x \in [-3, 2]$

We can see the largest positive root is between 1.5 and 2.

The methods' code can be found below. For legibility, printing related sections that are omitted can be found in the attached M-files.

The results are presented in the following order:

1. Secant (SM),
2. Bisection (BM),
3. Newton's (NM),
4. And modified Newton's method with cubic convergence (question 2) (NMC).

1. Secant method:

secant.m:

```
function root = secant(fname,x0,x1,xtol,ftol,n_max,display)
n = 0;
fx0 = feval(fname,x0);
fx1 = feval(fname,x1);
if abs(fx0) <= ftol
  root = x0;
  return
elseif abs(fx1) <= ftol
  root = x1;
  return
end
for n = 1:n_max
  d = ((x1-x0)/(fx1-fx0))*fx1;
  x0 = x1;
  fx0 = fx1;
  x1 = x1 - d;
  fx1 = feval(fname,x1);
  if abs(d) <= xtol | abs(fx1) <= ftol
           root = x1;
           return
  end
end
root = x1;
```

The results are:

n	x0	x1	f(x0)	f(x1)
0	1.000000000000000e+00	2.000000000000000e+00	-1.000000000000000e+00	1.000000000000000e+00
1	2.000000000000000e+00	1.500000000000000e+00	1.000000000000000e+00	-1.125000000000000e+00
2	1.500000000000000e+00	1.764705882352941e+00	-1.125000000000000e+00	-3.279055566863436e-01
3	1.764705882352941e+00	1.873599540361965e+00	-3.279055566863436e-01	2.090397299390609e-01
4	1.873599540361965e+00	1.831205833909406e+00	2.090397299390609e-01	-1.541953360163806e-02
5	1.831205833909406e+00	1.834118127083818e+00	-1.541953360163806e-02	-6.368734578732216e-04
6	1.834118127083818e+00	1.834243595856440e+00	-6.368734578732216e-04	2.096128623563231e-06
7	1.834243595856440e+00	1.834243184258312e+00	2.096128623563231e-06	-2.832374335071108e-10
8	1.834243184258312e+00	1.834243184313922e+00	-2.832374335071108e-10	8.881784197001252e-16

Figure 2: Root 1.8342 found through the secant method

2. Bisection method:

Because the code is given, it will not be included here.

n	a	b	c	f(c)	error_bound
0	1.000000000000000e+00	3.000000000000000e+00	2.000000000000000e+00	1.000000000000000e+00	1.000000000000000e+00
1	1.000000000000000e+00	2.000000000000000e+00	1.500000000000000e+00	-1.125000000000000e+00	5.000000000000000e-01
2	1.500000000000000e+00	2.000000000000000e+00	1.750000000000000e+00	-3.906250000000000e-01	2.500000000000000e-01
3	1.750000000000000e+00	2.000000000000000e+00	1.875000000000000e+00	2.167968750000000e-01	1.250000000000000e-01
4	1.750000000000000e+00	1.875000000000000e+00	1.812500000000000e+00	-1.081542968750000e-01	6.250000000000000e-02
5	1.812500000000000e+00	1.875000000000000e+00	1.843750000000000e+00	4.891967773437500e-02	3.125000000000000e-02
6	1.812500000000000e+00	1.843750000000000e+00	1.828125000000000e+00	-3.095626831054688e-02	1.562500000000000e-02
7	1.828125000000000e+00	1.843750000000000e+00	1.835937500000000e+00	8.645534515380859e-03	7.812500000000000e-03
8	1.828125000000000e+00	1.835937500000000e+00	1.832031250000000e+00	-1.123923063278198e-02	3.906250000000000e-03
9	1.832031250000000e+00	1.835937500000000e+00	1.833984375000000e+00	-1.317836344242096e-03	1.953125000000000e-03
10	1.833984375000000e+00	1.835937500000000e+00	1.834960937500000e+00	3.658599220216274e-03	9.765625000000000e-04
11	1.833984375000000e+00	1.834960937500000e+00	1.834472656250000e+00	1.169069320894778e-03	4.882812500000000e-04
12	1.833984375000000e+00	1.834472656250000e+00	1.834228515625000e+00	-7.471149729099125e-05	2.441406250000000e-04
13	1.834228515625000e+00	1.834472656250000e+00	1.834350585937500e+00	5.470969099405920e-04	1.220703125000000e-04
14	1.834228515625000e+00	1.834350585937500e+00	1.834289550781250e+00	2.361722065415961e-04	6.103515625000000e-05
15	1.834228515625000e+00	1.834289550781250e+00	1.834259033203125e+00	8.072522976476648e-05	3.051757812500000e-05
16	1.834228515625000e+00	1.834259033203125e+00	1.834243774414062e+00	3.005585032411773e-06	1.525878906250000e-05
17	1.834228515625000e+00	1.834243774414062e+00	1.834236145019531e+00	-3.585327642952052e-05	7.629394531250000e-06
18	1.834236145019531e+00	1.834243774414062e+00	1.834239959716797e+00	-1.642392577316798e-05	3.814697265625000e-06
19	1.834239959716797e+00	1.834243774414062e+00	1.834241867065430e+00	-6.709190389031505e-06	1.907348632812500e-06
20	1.834241867065430e+00	1.834243774414062e+00	1.834242820739746e+00	-1.851807683195261e-06	9.536743164062500e-07
21	1.834242820739746e+00	1.834243774414062e+00	1.834243297576904e+00	5.768874231648624e-07	4.768371582031250e-07
22	1.834242820739746e+00	1.834243297576904e+00	1.834243059158325e+00	-6.374604426540031e-07	2.384185791015625e-07
23	1.834243059158325e+00	1.834243297576904e+00	1.834243178367615e+00	-3.028658746018209e-08	1.192092895507812e-07
24	1.834243178367615e+00	1.834243297576904e+00	1.834243237972260e+00	2.733003983124149e-07	5.960464477539062e-08
25	1.834243178367615e+00	1.834243237972260e+00	1.834243208169937e+00	1.215069005411351e-07	2.980232238769531e-08
26	1.834243178367615e+00	1.834243208169937e+00	1.834243193268776e+00	4.561015476411967e-08	1.490116119384766e-08
27	1.834243178367615e+00	1.834243193268776e+00	1.834243185818195e+00	7.661783207879580e-09	7.450580596923828e-09
28	1.834243178367615e+00	1.834243185818195e+00	1.834243182092905e+00	-1.131240257024047e-08	3.725290298461914e-09
29	1.834243182092905e+00	1.834243185818195e+00	1.834243183955550e+00	-1.825309681180443e-09	1.862645149230957e-09
30	1.834243183955550e+00	1.834243185818195e+00	1.834243184886873e+00	2.918237207438779e-09	9.313225746154785e-10
31	1.834243183955550e+00	1.834243184886873e+00	1.834243184421211e+00	5.464642072183779e-10	4.656612873077393e-10
32	1.834243183955550e+00	1.834243184421211e+00	1.834243184188381e+00	-6.394236251594521e-10	2.328306436538696e-10
33	1.834243184188381e+00	1.834243184421211e+00	1.834243184304796e+00	-4.648015305974695e-11	1.164153218269348e-10
34	1.834243184304796e+00	1.834243184421211e+00	1.834243184363004e+00	2.499920270793154e-10	5.820766091346741e-11
35	1.834243184304796e+00	1.834243184363004e+00	1.834243184333900e+00	1.017559370097842e-10	2.910383045673370e-11
36	1.834243184304796e+00	1.834243184333900e+00	1.834243184319348e+00	2.763833606422850e-11	1.455191522836685e-11
37	1.834243184304796e+00	1.834243184319348e+00	1.834243184312072e+00	-9.420020319339528e-12	7.275957614183426e-12
38	1.834243184312072e+00	1.834243184319348e+00	1.834243184315710e+00	9.109157872444484e-12	3.637978807091713e-12
39	1.834243184312072e+00	1.834243184315710e+00	1.834243184313891e+00	-1.554312234475219e-13	1.818989403545856e-12
40	1.834243184313891e+00	1.834243184315710e+00	1.834243184314801e+00	4.476419235288631e-12	9.094947017729282e-13

Figure 3: All iterations of the bisection method.

3. Newton's method:

n	x	f(x)
0	2.000000000000000e+00	1.000000000000000e+00
1	1.857142857142857e+00	1.195335276967926e-01
2	1.834787350054526e+00	2.773253015902810e-03
3	1.834243503918507e+00	1.627856713426468e-06
4	1.834243184314032e+00	5.613287612504791e-13

Figure 4: All iterations of Newton's method

4. Modified Newton's method with cubic convergence:

newtonCubic.m:

```
function root = newtonCubic(fname,fdname,fd2name,x,xtol,ftol,n_max,display)
n = 0;
fx = feval(fname,x);
if abs(fx) <= ftol
  root = x;
  return
end
for n = 1:n_max
  fdx = feval(fdname,x);
  fd2x = feval(fd2name,x);
  d = fx/fdx + (fd2x*fx*fx)/(2*fdx*fdx*fdx);
  x = x - d;
  fx = feval(fname,x);
  if abs(d) <= xtol | abs(fx) <= ftol
     root = x;
     return
  end
end
root = x;
```

The results are:

n	x	f(x)
0	2.000000000000000e+00	1.000000000000000e+00
1	1.839650145772595e+00	2.770054731414628e-02
2	1.834243514888597e+00	1.683731199797478e-06
3	1.834243184313922e+00	8.881784197001252e-16

Figure 5: All iterations of modified/cubic Newton's method

Convergence speed comparison:

We see BM requires the most iterations (40) and therefore has the slowest convergence out of all the other methods. This is because it only has **linear** convergence.

On the other hand, SM has faster convergence than BM's, taking 8 iterations, but slower than both NM and NMC, which only took 4 and 3 iterations respectively.

The reason is that SM has **superlinear** convergence (superlinear > linear), but NM and NMC have **quadratic** and **cubic** convergence respectively (cubic > quadratic > superlinear).