Problem Set 1

Wentao Zhu

1

(a)

Multiplications: $n * n = n^2$

Additions: $n * n - 1 = n^2 - 1$

(b)

Equivalent form: $(\sum_{i=1}^{n} a_i) * (\sum_{j=1}^{n} b_j)$

Multiplications: 1

Additions: $2 * (n - 1) = 2n - 2$

2

By definition, there exist $\alpha_0 > 1, \lambda_0 > 0$ satisfying

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^{\alpha_0}} = \lambda_0$$

Therefore,

$$\lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|} = \lim_{n\to\infty} \frac{|p_{n+1} - p|}{|p_n - p|^{\alpha_0}} \cdot \lim_{n\to\infty} |p_n - p|^{\alpha_0 - 1} = \lambda_0 \cdot 0 = 0$$

p_n converges to p superlinearly.

3

	Chopping	Rounding
e	$0.27182 * 10^1$	$0.27183 * 10^1$
$\frac{1}{7}$	$0.14285 * 10^0$	$0.14286 * 10^0$

4

By Definition,

$$fl_{round}(x) = \begin{cases} (.\,d_1 d_2 \ldots d_p)_\beta \times \beta^e & \text{if } d_{p+1} < \frac{\beta}{2} \\ [(.\,d_1 d_2 \ldots d_p)_\beta + \beta^{-p}] \times \beta^e & \text{else} \end{cases}$$

Therefore,

$$\epsilon_{abs} = |fl_{round}(x) - x| \leq \frac{\beta}{2} \times \beta^{e-p-1}$$

$$\epsilon_{rel} = \frac{\epsilon_{abs}}{|x|} \leq \frac{\frac{\beta}{2} \times \beta^{e-p-1}}{\beta^{e-1}} = \frac{1}{2}\beta^{1-p}$$

5

$$f_a(x) = x^2 - a$$

Use *Newton's Method*,

$$g(x) = x - \frac{f(x)}{f'(x)}$$

We get

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

$$\lim_{n \to \infty} x_n = a$$

Set $x_0 = 1$, $\{x_n\}$ can be computed recursively.

6

Secant Method:

$$p_{n+1} = p_n - \frac{f(p_n)(p_n - p_{n-1})}{f(p_n) - f(p_{n-1})}$$

Compute p_4 using *Matlab*:

```
f = @(x) log(1+x)-cos(x);
p = [0,1,0,0,0];
for i = 2:4
  p(i+1) = p(i) - (f(p(i))*(p(i)-p(i-1)))/(f(p(i))-f(p(i-1)));
  fprintf('p_%d = %f, f(p_%d) = %f\n', i, p(i+1), i, f(p(i+1)));
end
```

Result:

```
p_2 = 0.867419, f(p_2) = -0.022239
p_3 = 0.884260, f(p_3) = -0.000327
p_4 = 0.884511, f(p_4) = 0.000001
```

7

(a)

```
solve(0,1)
global t;
t = 0;

function x = solve(a,b)
    f = @(x) log(1+x)-cos(x);
    global t;
    epsilon = 1e-6;
    p = (a+b)/2;
    t = t + 1;
    fprintf('p_%d = %f, f(p_%d) = %f\n', t, p, t, f(p));
    if (p-a<=epsilon)
        x = p;
    elseif (f(a)*f(p)<0)
        x = solve(a,p);
    else
        x = solve(p,b);
    end
end
```

Result:

```
p_1 = 0.500000, f(p_1) = -0.472117
p_2 = 0.750000, f(p_2) = -0.172073
p_3 = 0.875000, f(p_3) = -0.012388
```

```
p_4 = 0.937500, f(p_4) = 0.069593
p_5 = 0.906250, f(p_5) = 0.028436
p_6 = 0.890625, f(p_6) = 0.007981
p_7 = 0.882812, f(p_7) = -0.002214
p_8 = 0.886719, f(p_8) = 0.002881
p_9 = 0.884766, f(p_9) = 0.000333
p_10 = 0.883789, f(p_10) = -0.000941
p_11 = 0.884277, f(p_11) = -0.000304
p_12 = 0.884521, f(p_12) = 0.000014
p_13 = 0.884399, f(p_13) = -0.000145
p_14 = 0.884460, f(p_14) = -0.000065
p_15 = 0.884491, f(p_15) = -0.000026
p_16 = 0.884506, f(p_16) = -0.000006
p_17 = 0.884514, f(p_17) = 0.000004
p_18 = 0.884510, f(p_18) = -0.000001
p_19 = 0.884512, f(p_19) = 0.000002
p_20 = 0.884511, f(p_20) = 0.000000
```

(b)

$$f'(x) = \frac{1}{1+x} + sin(x)$$

```
f = @(x) log(1+x)-cos(x);
fd = @(x) 1/(1+x)+sin(x);
p = 1/2;
q = 0;
epsilon = 1e-6;
t = 0;

while (1)
    t = t + 1;
    fprintf('p_%d = %f, f(p_%d) = %f\n', t, p, t, f(p));
    if (abs(p-q)<=epsilon)
        break
    end
    q = p;
    p = p - f(p)/fd(p);
end
```

Result:

```
p_1 = 0.500000, f(p_1) = -0.472117
p_2 = 0.911937, f(p_2) = 0.035901
p_3 = 0.884609, f(p_3) = 0.000128
p_4 = 0.884511, f(p_4) = 0.000000
p_5 = 0.884511, f(p_5) = 0.000000
```

(c)

Newton's method is more adaptive and therefore converges quicklier, while *bisection method* is more inflexible.

(d)

```
f = @(x) log(1+x)-cos(x);
p = zeros(1,1000);
epsilon = 1e-6;
p(2) = 1;
i = 2;
while (abs(p(i)-p(i-1))>epsilon)
  p(i+1) = p(i) - (f(p(i))*(p(i)-p(i-1)))/(f(p(i))-f(p(i-1)));
  fprintf('p_%d = %f, f(p_%d) = %f\n', i, p(i+1), i, f(p(i+1)));
  i = i + 1;
end
```

Result:

```
p_2 = 0.867419, f(p_2) = -0.022239
p_3 = 0.884260, f(p_3) = -0.000327
p_4 = 0.884511, f(p_4) = 0.000001
p_5 = 0.884511, f(p_5) = -0.000000
```

Secant method and *Newton method* converge quicklier than *bisection method*, while *secant method* and *bisection method* require less information for the function $f(x)$ than *Newton method*.