Numerical Analysis
Assignment 2

Chuan

September 4, 2017

Problem 1. Problem 1.20

Solution. The code of the first algorithm(the trivial one) is as follows.

```
function [res] = prob20(x, y)
% Problem 20;
% To compute \lim_{p\to\infty}(x^p+y^p)^{1/p};
    p = 2.^(1:20);
    res = ((x.^p+y.^p).^(1./p))';
```

Result. The result of computing this program is listed follows, from which we can see that the result either overflows or underflows when in extreme conditions.

```
>> format long
>> prob20(10^10, 10^10)

ans =

   1.0e+10 *

   1.414213562373095
   1.189207115002721
   1.090507732665258
   1.044273782427414
                 Inf
                 Inf
                 Inf
                 Inf
                 Inf
                 Inf
                 Inf
                 Inf
                 Inf
                 Inf
                 Inf
                 Inf
                 Inf
                 Inf
                 Inf
                 Inf
```

```
>> format long
>> prob20(10^-10, 10^-10)

ans =

   1.0e-09 *

   0.141421356237310
   0.118920711500272
   0.109050773266526
   0.104427378242741
   0.102189679313363
                   0
                   0
                   0
                   0
                   0
                   0
                   0
                   0
                   0
                   0
                   0
                   0
                   0
                   0
                   0
```

Second part. When we use the idea in (1.3.8), the code is as follows.

```
function [res] = prob20revised(x, y)
% prob 20;
% revised the computation by
% \lim_{p\to\infty} ((x/y)^p+1)^{1/p}*y
% where (x/y) < 1;
    p = 2.^(1:20);
```

```
    if(abs(x) < abs(y))
        res = ((x/y).^p+1).^(1./p)*y;
    else
        res = ((y/x).^p+1).^(1./p)*x;
    end
    res = res';
```

Result. And the result is listed as follows.

```
>> prob20revised(10^10, 10^10)

ans =

   1.0e+10 *

   1.414213562373095
   1.189207115002721
   1.090507732665258
   1.044273782427414
   1.021897148654117
   1.010889286051701
   1.005429901112803
   1.002711275050203
   1.001354719892108
   1.000677130693066
   1.000338508052682
   1.000169239705302
   1.000084616272694
   1.000042307241396
   1.000021153396965
   1.000010576642550
   1.000005288307292
   1.000002644150150
   1.000001322074201
   1.000000661036882
```

```
>> prob20revised(10^-10, 10^-10)

ans =

   1.0e-09 *

   0.141421356237310
   0.118920711500272
   0.109050773266526
   0.104427378242741
   0.102189714865412
   0.101088928605170
   0.100542990111280
   0.100271127505020
   0.100135471989211
   0.100067713069307
   0.100033850805268
   0.100016923970530
   0.100008461627269
   0.100004230724140
   0.100002115339696
   0.100001057664255
   0.100000528830729
   0.100000264415015
   0.100000132207420
   0.100000066103688
```

Another test. We can then try the case where x and y differs, and the computing result remains the same as the theoretical result that the limit converge to the larger number.

```
>> prob20revised(1, 2)

ans =

   2.236067977499790
   2.030543184868931
   2.000974897633078
   2.000001907334990
   2.000000000014552
   2.000000000000000
   2.000000000000000
   2.000000000000000
   2.000000000000000
   2.000000000000000
   2.000000000000000
   2.000000000000000
   2.000000000000000
   2.000000000000000
   2.000000000000000
   2.000000000000000
   2.000000000000000
   2.000000000000000
   2.000000000000000
```

Problem 2. Computations of

$$\sqrt{1+x}-1,\ x \to 0.$$

Original. The code for original computation is as follows.

```
function res = prob2(x)
% problem 2;
% compute \sqrt{x+1}-1;
    res = ((x+1).^(1/2)-1)';
```

Modified. Theoretically, in order to avoid cancellation, we can modify the computation to

$$\frac{x}{\sqrt{1+x}+1}.$$

and the code is as follows.

```
function res = prob2revised(x)
% prob 2;
% revised to x/(sqrt(x+1)+1);
    res = x./((x+1).^(1/2)+1);
    res = res';
```

Result. The results of the original function and revised function are listed as follows, in which the left is original result and the right is the revised result.

```
>> x = 10.^(-1:-1:-16);
>> prob2(x)

ans =

   0.048808848170152
   0.004987562112089
   0.000499875062461
   0.000049998750062
   0.000004999987500
   0.000000499999875
   0.000000049999999
   0.000000005000000
   0.000000000500000
   0.000000000050000
   0.000000000005000
   0.000000000000500
   0.000000000000050
   0.000000000000005
   0.000000000000000
                   0
```

```
>> x = 10.^(-1:-1:-16);
>> prob2revised(x)

ans =

   0.048808848170152
   0.004987562112089
   0.000499875062461
   0.000049998750062
   0.000004999987500
   0.000000499999875
   0.000000049999999
   0.000000005000000
   0.000000000500000
   0.000000000050000
   0.000000000005000
   0.000000000000500
   0.000000000000050
   0.000000000000005
   0.000000000000000
   0.000000000000000
```

Discussion. From the result we can know when $x = 10^{-16}$, the original computation loses all significant numbers, while the modified computation still has a significant number.