

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.

```
1 function [res] = prob20(x, y)
2 % Problem 20;
3 % To compute  $\lim_{p \rightarrow \infty} (x^p + y^p)^{1/p}$ ;
4     p = 2.^(1:20);
5     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.

```
1 >> format long
2 >> prob20(10^10, 10^10)
3
4 ans =
5
6     1.0e+10 *
7
8     1.414213562373095
9     1.189207115002721
10    1.090507732665258
11    1.044273782427414
12                                Inf
13                                Inf
14                                Inf
15                                Inf
16                                Inf
17                                Inf
18                                Inf
19                                Inf
20                                Inf
21                                Inf
22                                Inf
23                                Inf
24                                Inf
25                                Inf
26                                Inf
27                                Inf
```

```
1 >> format long
2 >> prob20(10^-10, 10^-10)
3
4 ans =
5
6     1.0e-09 *
7
8     0.141421356237310
9     0.118920711500272
10    0.109050773266526
11    0.104427378242741
12    0.102189679313363
13                                0
14                                0
15                                0
16                                0
17                                0
18                                0
19                                0
20                                0
21                                0
22                                0
23                                0
24                                0
25                                0
26                                0
27                                0
```

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

```
1 function [res] = prob20revised(x, y)
2 % prob 20;
3 % revised the computation by
4 %  $\lim_{p \rightarrow \infty} ((x/y)^p + 1)^{1/p} * y$ 
5 % where  $(x/y) < 1$ ;
6     p = 2.^(1:20);
```

```

7   if(abs(x) < abs(y))
8       res = ((x/y).^p+1).^(1./p)*y;
9   else
10      res = ((y/x).^p+1).^(1./p)*x;
11  end
12  res = res';

```

Result. And the result is listed as follows.

```

1  >> prob20revised(10^10, 10^10)
2
3  ans =
4
5      1.0e+10 *
6
7      1.414213562373095
8      1.189207115002721
9      1.090507732665258
10     1.044273782427414
11     1.021897148654117
12     1.010889286051701
13     1.005429901112803
14     1.002711275050203
15     1.001354719892108
16     1.000677130693066
17     1.000338508052682
18     1.000169239705302
19     1.000084616272694
20     1.000042307241396
21     1.000021153396965
22     1.000010576642550
23     1.000005288307292
24     1.000002644150150
25     1.000001322074201
26     1.000000661036882

```

```

1  >> prob20revised(10^-10, 10^-10)
2
3  ans =
4
5      1.0e-09 *
6
7      0.141421356237310
8      0.118920711500272
9      0.109050773266526
10     0.104427378242741
11     0.102189714865412
12     0.101088928605170
13     0.100542990111280
14     0.100271127505020
15     0.100135471989211
16     0.100067713069307
17     0.100033850805268
18     0.100016923970530
19     0.100008461627269
20     0.100004230724140
21     0.100002115339696
22     0.100001057664255
23     0.100000528830729
24     0.100000264415015
25     0.100000132207420
26     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.

```

1  >> prob20revised(1, 2)
2
3  ans =
4
5      2.236067977499790
6      2.030543184868931
7      2.000974897633078
8      2.000001907334990
9      2.000000000014552
10     2.000000000000000
11     2.000000000000000
12     2.000000000000000
13     2.000000000000000
14     2.000000000000000
15     2.000000000000000
16     2.000000000000000
17     2.000000000000000
18     2.000000000000000
19     2.000000000000000
20     2.000000000000000
21     2.000000000000000
22     2.000000000000000
23     2.000000000000000

```

Problem 2. Computations of

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

Original. The code for original computation is as follows.

```
1 function res = prob2(x)
2 % problem 2;
3 % compute \sqrt{x+1}-1;
4 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.

```
1 function res = prob2revised(x)
2 % prob 2;
3 % revised to x/(sqrt(x+1)+1);
4 res = x./((x+1).^(1/2)+1);
5 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.

```
1 >> x = 10.^(-1:-1:-16);
2 >> prob2(x)
3
4 ans =
5
6     0.048808848170152
7     0.004987562112089
8     0.000499875062461
9     0.000049998750062
10    0.000004999987500
11    0.000000499999875
12    0.000000049999999
13    0.000000005000000
14    0.000000000500000
15    0.000000000050000
16    0.000000000005000
17    0.000000000000500
18    0.000000000000050
19    0.000000000000005
20    0.000000000000000
21    0
```

```
1 >> x = 10.^(-1:-1:-16);
2 >> prob2revised(x)
3
4 ans =
5
6     0.048808848170152
7     0.004987562112089
8     0.000499875062461
9     0.000049998750062
10    0.000004999987500
11    0.000000499999875
12    0.000000049999999
13    0.000000005000000
14    0.000000000500000
15    0.000000000050000
16    0.000000000005000
17    0.000000000000500
18    0.000000000000050
19    0.000000000000005
20    0.000000000000000
21    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.