

---

# MAT 420 Exam 1

---

Sen Chen

Oct 28th

## Problem 1

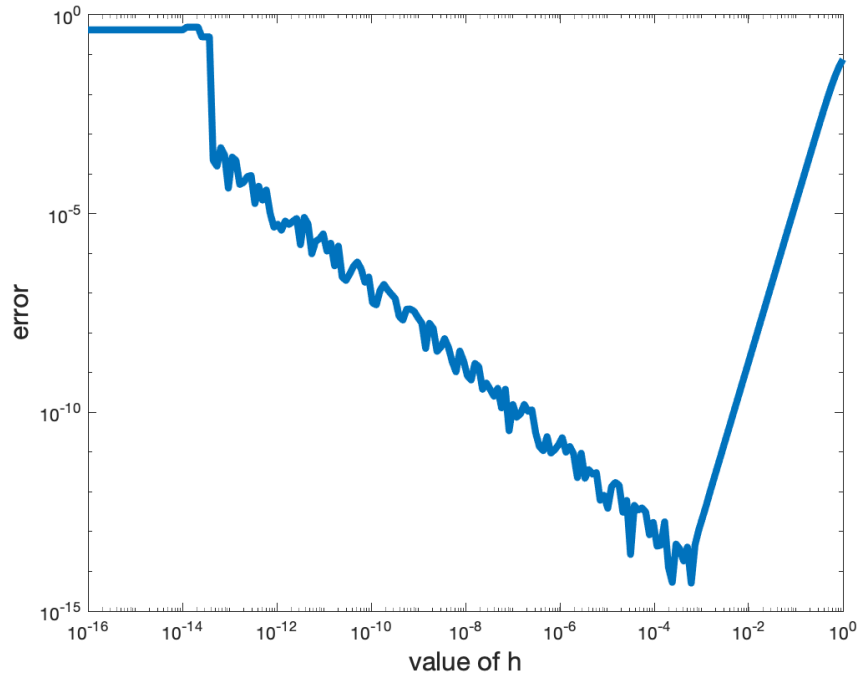
(a) Function  $Q(x, h)$  approximates  $f'(x)$  when  $h \rightarrow 0$ .

$$\begin{aligned}\lim_{h \rightarrow 0} Q(h) &= \lim_{h \rightarrow 0} \frac{f(x-2h) - 8f(x-h) + 8f(x+h) - f(x+2h)}{12h} \\ &= \lim_{h \rightarrow 0} \frac{f(x+h) - f(x-h)}{2h} \times \frac{8}{6} + \lim_{h \rightarrow 0} \frac{f(x+2h) - f(x-2h)}{4h} \times \left(-\frac{1}{3}\right) \\ &= \frac{4}{3}f'(x) - \frac{1}{3}f'(x) \\ &= f'(x)\end{aligned}$$

(b) Here, we write a Matlab code to numerically calculate the optimal value of  $h$  which minimizes the approximation error  $|Q(x_0, h) - f'(x_0)|$ , given the function  $f$  and a point  $x_0$ .

```
1 syms f(x)
2
3 % input: function f
4 f(x) = tanh(x);
5 df = diff(f,x); df5 = diff(f,x,5);
6
7 % input: point x0
8 x0 = 1;
9
10 % function Q
11 Q = @(h) (f(x0-2*h) - 8*f(x0-h) + 8*f(x0+h) - f(x0+2*h)) / (12*h);
12
13 % machine epsilon
14 epsilon = 1;
15 while (1+epsilon>1)
16     epsilon = epsilon/2;
17 end
18 epsilon = epsilon*2;
19
20 % approximation error
21 error = @(h) abs(df(x0) - Q(h));
22
23 % plot error using different h
24 h = logspace(-16, 0, 200);
25 error_h = arrayfun(error, h);
26 loglog(h, error_h, 'linewidth', 4);
27 xlabel('value of h', 'fontsize', 15);
28 ylabel('error', 'fontsize', 15);
29
30 % the optimal h is roughly:
31 h_opt = h(find(error_h==min(error_h)))
32
33 % trade-off between roundoff error and truncation errors
34 h_opt_2 = double(15*epsilon*abs(f(x0))/2/abs(df5(x0)))^(1/5)
```

In the case of  $f(x) = \tanh x$  and  $x_0 = 1$ , the output figure is



where the error **first drops and then rises** as  $h$  increases, and the optimal value of  $h$  is roughly

```
1 h_opt =
2 6.0802e-04
```

(c) Analytically, the optimal choice of  $h$  considers the trade-off between roundoff errors and truncation errors:

- large values of  $h$  lead to large truncation errors

$$Q(x, h) = \frac{4}{3} \frac{f(x+h) - f(x-h)}{2h} - \frac{1}{3} \frac{f(x+2h) - f(x-2h)}{4h}$$

$$Q(x, h) \approx \frac{4}{3} \frac{(f + hf' + \frac{h^2}{2}f'' + \frac{h^3}{6}f''' + \frac{h^4}{24}f^{(4)} + \frac{h^5}{120}f^{(5)}) - (f - hf' + \frac{h^2}{2}f'' - \frac{h^3}{6}f''' + \frac{h^4}{24}f^{(4)} - \frac{h^5}{120}f^{(5)})}{2h}$$

$$- \frac{1}{3} \frac{(f + 2hf' + \frac{4h^2}{2}f'' + \frac{8h^3}{6}f''' + \frac{16h^4}{24}f^{(4)} + \frac{32h^5}{120}f^{(5)}) - (f - 2hf' + \frac{4h^2}{2}f'' - \frac{8h^3}{6}f''' + \frac{16h^4}{24}f^{(4)} - \frac{32h^5}{120}f^{(5)})}{4h}$$

$$\Rightarrow E_T \approx \frac{|f^{(5)}(x)|}{30} h^4$$

- small values of  $h$  lead to large round-off errors

$$E_R \approx \frac{|f(x)|\varepsilon}{h}$$

Hence, the total error is  $E(h) = E_T + E_R \approx \frac{|f^{(5)}(x)|}{30} h^4 + \frac{|f(x)|\varepsilon}{h}$ . Setting  $E'(h) = 0$  yields

$$h = \sqrt[5]{\frac{15|f(x)|\varepsilon}{2|f^{(5)}(x)|}}$$

```
1 h_opt =
2 7.4418e-04
```

which justifies the result from (b).

## Problem 2

The Perl code is given as following.

```
1 #!/usr/bin/perl -w
2 #prime numbers less than N
3 use strict;
4 my $N = $ARGV[0];
5 my @num = (1) x $N;
6 for (my $i=2;$i<=sqrt($N);$i++) {
7     if ($num[$i]==1){
8         for(my $j=$i*$i;$j<=$N;$j+==$i){ $num[$j]=0; }
9     }
10 }
11 for (my $i=2;$i<=$N;$i++){
12     if ($num[$i]==1) { printf("%i ",$i);}
13 printf("\n");
```

When user input  $N = 10000$ , the code outputs

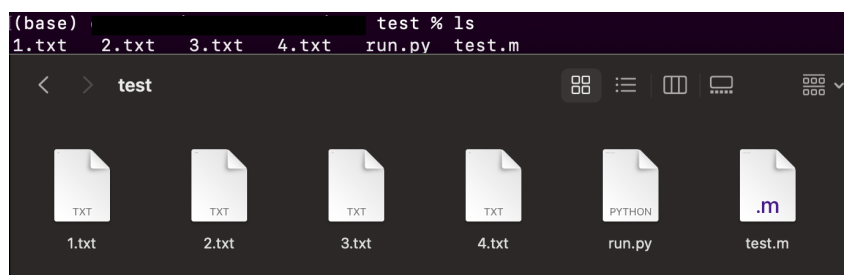
```
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199 211 2
23 227 229 233 239 241 251 257 263 269 271 277 281 283 293 307 311 313 317 331 337 347 349 353 359 367 373 379 383 389 397 401 409 419 421 431 433 439 443 449 4
57 461 463 467 479 487 491 499 503 509 521 523 541 547 557 563 569 571 577 587 593 599 601 607 613 617 619 631 641 643 647 653 659 661 673 677 683 691 701 709 7
19 727 733 739 743 751 757 761 769 773 787 797 809 811 821 823 827 829 839 853 857 859 863 877 881 883 887 907 911 919 929 937 941 947 953 967 971 977 983 991 9
97 1009 1013 1019 1021 1031 1033 1039 1049 1051 1061 1063 1069 1087 1091 1093 1097 1103 1109 1117 1121 1129 1151 1163 1171 1181 1187 1193 1201 1213 1217 12
23 1229 1231 1237 1249 1259 1277 1279 1283 1289 1291 1297 1301 1303 1307 1319 1321 1327 1361 1367 1373 1381 1399 1409 1423 1427 1429 1433 1439 1447 1451 1453 14
59 1471 1481 1483 1487 1489 1493 1499 1511 1523 1531 1543 1549 1553 1559 1567 1571 1579 1583 1597 1601 1607 1609 1613 1619 1621 1627 1637 1657 1663 1667 1669 16
91 1697 1699 1709 1721 1723 1733 1741 1747 1753 1759 1777 1783 1787 1789 1801 1811 1823 1831 1847 1861 1867 1871 1873 1877 1879 1889 1901 1907 1913 1931 1933 19
49 1951 1973 1979 1987 1993 1997 1999 2003 2011 2017 2027 2029 2039 2053 2063 2069 2081 2083 2087 2089 2099 2111 2113 2129 2131 2137 2141 2143 2153 2161 2179 22
83 2207 2213 2221 2237 2239 2243 2251 2267 2269 2273 2281 2287 2293 2297 2309 2311 2333 2339 2341 2347 2351 2357 2371 2377 2381 2383 2389 2393 2399 2411 2417 24
23 2437 2441 2447 2459 2467 2473 2477 2503 2521 2531 2539 2543 2549 2551 2557 2579 2591 2593 2609 2617 2621 2633 2647 2657 2659 2663 2671 2677 2683 2687 2689 26
91 2699 2707 2711 2713 2719 2729 2731 2741 2749 2753 2767 2777 2789 2791 2797 2801 2803 2819 2833 2837 2843 2851 2857 2861 2879 2887 2897 2903 2909 2917 2927 29
39 2953 2957 2963 2969 2971 2999 3001 3011 3019 3023 3037 3041 3049 3061 3067 3079 3083 3089 3109 3119 3121 3137 3163 3167 3169 3181 3187 3191 3203 3209 3217 32
31 3229 3251 3253 3257 3259 3271 3299 3301 3307 3313 3319 3323 3329 3331 3343 3347 3359 3361 3371 3373 3389 3391 3407 3413 3433 3449 3457 3461 3463 3467 3469 34
91 3499 3511 3517 3527 3529 3533 3539 3541 3547 3557 3559 3571 3581 3583 3593 3607 3613 3617 3623 3631 3637 3643 3659 3671 3673 3677 3691 3697 3701 3709 3719 37
27 3733 3759 3761 3767 3769 3779 3797 3803 3821 3823 3833 3847 3851 3853 3863 3877 3881 3889 3907 3911 3917 3919 3923 3929 3931 3943 3947 3967 3969 4001 40
83 4087 4093 4097 4099 4107 4109 4117 4127 4129 4133 4139 4153 4157 4159 4177 4201 4211 4217 4219 4229 4231 4241 4243 4253 42
59 4261 4271 4273 4283 4289 4297 4327 4337 4339 4349 4357 4363 4373 4391 4397 4409 4421 4423 4441 4447 4451 4457 4463 4481 4483 4493 4507 4513 4517 4519 4523 45
47 4549 4561 4567 4583 4591 4597 4603 4621 4637 4639 4643 4649 4653 4657 4663 4673 4679 4691 4703 4721 4723 4729 4733 4751 4759 4783 4787 4789 4793 4799 4801 48
19 4817 4831 4861 4871 4877 4889 4903 4909 4919 4931 4933 4937 4943 4951 4957 4967 4969 4973 4987 4993 4999 5003 5009 5011 5021 5023 5039 5051 5059 5077 5081 50
87 5099 5101 5107 5113 5119 5147 5153 5167 5171 5179 5189 5197 5209 5227 5231 5233 5237 5241 5273 5279 5281 5297 5303 5309 5323 5333 5347 5351 5381 5387 5393 53
99 5407 5413 5417 5419 5431 5437 5441 5443 5449 5471 5477 5479 5483 5501 5503 5507 5519 5521 5527 5531 5557 5563 5569 5573 5581 5591 5623 5639 5641 5647 5651 56
53 5657 5659 5669 5683 5689 5693 5701 5711 5717 5737 5741 5743 5749 5779 5783 5791 5801 5807 5813 5821 5827 5839 5843 5849 5851 5857 5861 5867 5869 5879 5881 58
97 5903 5923 5927 5929 5953 5961 5967 6007 6011 6029 6037 6043 6047 6053 6057 6073 6079 6089 6091 6101 6113 6121 6131 6133 6143 6151 6163 6173 6197 6199 6203 62
11 6217 6221 6229 6247 6257 6263 6269 6271 6277 6287 6299 6301 6311 6317 6323 6329 6337 6343 6353 6359 6361 6367 6373 6379 6389 6397 6421 6427 6449 6451 6459 64
73 6481 6491 6521 6529 6547 6551 6553 6563 6569 6571 6577 6581 6599 6607 6619 6637 6653 6659 6661 6673 6679 6689 6691 6701 6703 6709 6719 6733 6737 6751 6753 67
79 6781 6791 6793 6803 6823 6827 6829 6833 6841 6857 6863 6869 6871 6883 6899 6907 6911 6917 6947 6949 6959 6961 6967 6971 6977 6983 6991 6997 7001 7013 7019 70
27 7039 7043 7047 7049 7059 7069 7079 7083 7089 7121 7127 7129 7151 7159 7177 7187 7193 7207 7211 7213 7219 7229 7237 7243 7247 7253 7263 7297 7299 7301 7303 73
47 7351 7369 7393 7411 7417 7433 7451 7457 7459 7477 7481 7487 7489 7499 7507 7517 7523 7529 7537 7541 7547 7549 7559 7561 7573 7577 7583 7589 7591 7603 7607 76
21 7639 7643 7649 7659 7673 7681 7687 7691 7699 7703 7717 7723 7727 7741 7753 7757 7759 7769 7793 7817 7823 7829 7841 7853 7867 7873 7877 7879 7883 7901 7907 79
19 7927 7933 7937 7949 7951 7963 7993 8009 8011 8017 8039 8053 8059 8069 8081 8087 8089 8093 8101 8111 8117 8123 8147 8161 8167 8171 8179 8191 8209 8219 8221 82
31 8233 8237 8243 8245 8249 8273 8287 8291 8293 8297 8311 8317 8329 8353 8363 8369 8377 8387 8389 8419 8423 8429 8431 8443 8447 8461 8467 8501 8513 8521 8527 85
37 8539 8543 8563 8573 8581 8597 8599 8609 8623 8627 8629 8641 8647 8663 8669 8677 8681 8689 8693 8699 8707 8713 8719 8731 8737 8741 8747 8753 8761 8779 8783 88
83 8807 8819 8821 8831 8837 8839 8849 8861 8863 8867 8867 8887 8893 8923 8929 8933 8941 8951 8963 8969 8971 8999 9001 9007 9011 9013 9029 9041 9043 9049 9059 9067 90
93 9103 9109 9127 9133 9137 9151 9157 9161 9173 9181 9187 9199 9203 9209 9221 9227 9239 9241 9257 9277 9281 9283 9293 9311 9319 9323 9337 9341 9343 9349 9371 93
77 9391 9397 9403 9413 9419 9421 9431 9433 9437 9439 9441 9443 9447 9473 9479 9491 9497 9511 9521 9533 9539 9547 9551 9557 9561 9519 9523 9529 9531 9543 95
49 9561 9577 9579 9589 9597 9719 9721 9733 9739 9743 9749 9767 9769 9781 9787 9791 9803 9811 9817 9829 9833 9839 9851 9857 9859 9871 9883 9887 9901 9907 9923 99
```

## Problem 3

The awk command is shown as follows.

```
1 ls -l | awk 'BEGIN{SUM=0}
2 /~/ {
3     SIZE=$5;
4     FNAME=$9;
5     SUM += SIZE;
6     OUTPUT=sprintf("%-4d,%t%s\n%s",SIZE,FNAME,OUTPUT);
7 }
8 END{
9     print OUTPUT;
10    print sprintf("%-4d",SUM);
11 }' -> filesize.dat
```

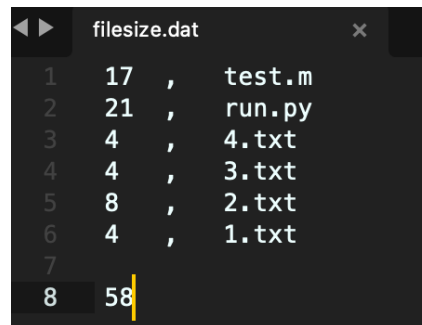
I test this command in the following folder:



After running the awk command like this:

```
(base) ; test % ls -l | awk 'BEGIN{SUM=0}
/^-/ {
    SIZE=$5;
    FNAME=$9;
    SUM += SIZE;
    OUTPUT=sprintf("%-4d,\t%s\n%s",SIZE,FNAME,OUTPUT);
}
END{
    print OUTPUT;
    print sprintf("%-4d",SUM);
}' -> filesize.dat
```

I obtain the "filesize.dat" file which looks like this:



```
1 17 , test.m
2 21 , run.py
3 4 , 4.txt
4 4 , 3.txt
5 8 , 2.txt
6 4 , 1.txt
7
8 58
```

## Problem 4

The Perl code is given as following.

```
1 #!/usr/bin/perl -w
2 use strict;
3
4 my $file = "data.txt";
5 my $fmt = "%.16e\t%.16e\t%.16e\n"; # print format: 2 tab-separated numbers
6 open(BDW,">$file"); # open new file for writing, with handle/ref BDW
7 for (my $i=1;$i<=4;$i++){
8     for (my $j=1;$j<=4;$j++){
9         my $x = 1/8 + ($i-1)*1/4;
10        my $y = 1/8 + ($j-1)*1/4;
11        my $f = exp(-$x*$x - $y*$y);
12        printf(BDW $fmt,$x,$y,$f);
13    }
14 }
15 close BDW; # close file using handle
16
17 sub readAndCompute{
18     my $errmsg = "Error: cannot open $file"; # define error message
19     open(BDW,"<$file") or die($errmsg);
20
21     my $sum = 0;
22     while (<BDW>) { # read one line at a time
23         chomp;
24         my ($x,$y,$f) = split("\t",$_); # get $x and $y from the 2 fields
25         $sum = $sum + $f
26     }
27     close BDW;
28
29     printf("The approximation of intergral is %.16e\n", $sum/16);
30 }
31
32 readAndCompute();
```

which gives the result as

```
1 The approximation of intergral is 5.6062226751501232e-01
```

## Problem 5

The recursive code in MATLAB is shown as following.

```
1 % A:source, B:help, C:target
2 % initial state: all disks are in source peg.
3 x = ['A', 'A', 'A', 'A'];
4
5 Hanoi(x, length(x), 'A','C','B');
6 function x = Hanoi(x, n, source, target, help)
7     if n > 0
8         % move n-1 disks from source to help so they are out of the way
9         x = Hanoi(x, n-1, source, help, target);
10        % move the nth (largest) disk to target
11        x(n) = target;
12        % move the n-1 disks on help to target
13        x = Hanoi(x, n-1, help, target, source);
14    end
15 end
```

which gives the result as

```
1 x =
2     'BAAA'
3
4 x =
5     'BCAA'
6
7 x =
8     'CCAA'
9
10 x =
11     'CCBA'
12
13 x =
14     'ACBA'
15
16 x =
17     'ABBA'
18
19 x =
20     'BBBA'
21
22 x =
23     'BBBC'
24
25 x =
26     'CBBC'
27
28 x =
29     'CABC'
30
31 x =
32     'AABC'
33
34 x =
35     'AACC'
36
37 x =
38     'BACC'
39
40 x =
41     'BCCC'
42
43 x =
44     'CCCC'
```

after  $2^4 - 1 = 15$  moves.