MAT 420 Exam 1

Sen Chen

Oct 28th

Problem 1

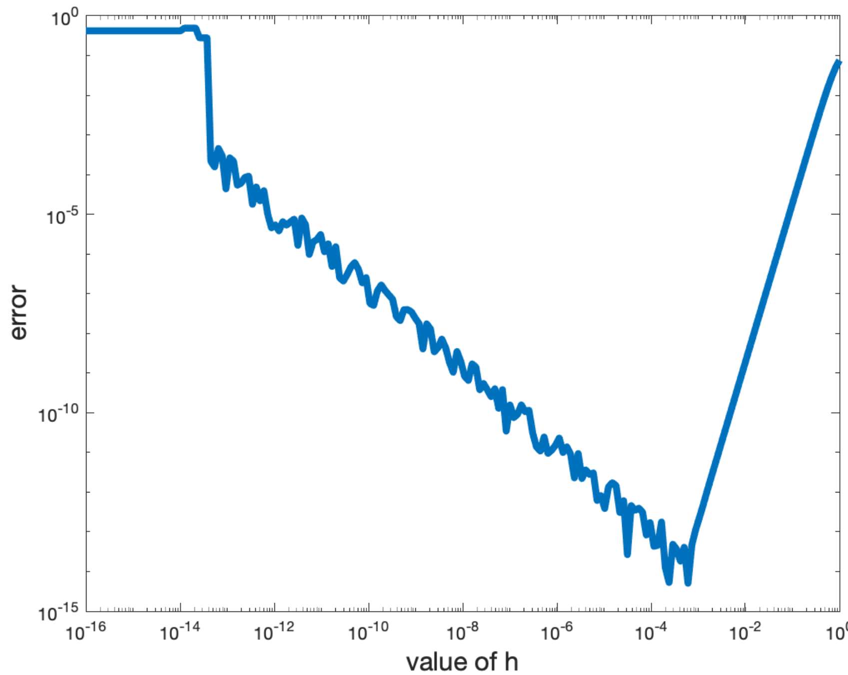
(a) Function $Q(x, h)$ approximates $f'(x)$ when $h \to 0$.

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

(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.

```matlab
syms f(x)

% input: function f
f(x) = tanh(x);
df = diff(f,x); df5 = diff(f,x,5);

% input: point x0
x0 = 1;

% function Q
Q = @(h) (f(x0-2*h) - 8*f(x0-h) + 8*f(x0+h) - f(x0+2*h)) / (12*h);

% machine epsilon
epsilon = 1;
while (1+epsilon>1)
    epsilon = epsilon/2;
end
epsilon = epsilon*2;

% approximation error
error = @(h) abs(df(x0) - Q(h));

% plot error using different h
h = logspace(-16, 0, 200);
error_h = arrayfun(error, h);
loglog(h, error_h, 'linewidth', 4);
xlabel('value of h', 'fontsize', 15);
ylabel('error', 'fontsize', 15);

% the optimal h is roughly:
h_opt = h(find(error_h==min(error_h)))

% trade-off between roundoff error and truncation errors
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

```
h_opt =
     6.0802e-04
```

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

- large values of h lead to large truncation errors

$$Q(x,h) = \frac{4}{3}\frac{f(x+h)-f(x-h)}{2h} - \frac{1}{3}\frac{f(x+2h)-f(x-2h)}{4h}$$
$$Q(x,h) \approx \frac{4}{3}\frac{(f+hf'+\frac{h^2}{2}f''+\frac{h^3}{6}f'''+\frac{h^4}{24}f''''+\frac{h^5}{120}f''''')-(f-hf'+\frac{h^2}{2}f''-\frac{h^3}{6}f'''+\frac{h^4}{24}f''''-\frac{h^5}{120}f''''')}{2h}$$
$$-\frac{1}{3}\frac{(f+2hf'+\frac{4h^2}{2}f''+\frac{8h^3}{6}f'''+\frac{16h^4}{24}f''''+\frac{32h^5}{120}f''''')-(f-2hf'+\frac{4h^2}{2}f''-\frac{8h^3}{6}f'''+\frac{16h^4}{24}f''''-\frac{32h^5}{120}f''''')}{4h}$$
$$\Longrightarrow E_T \approx \frac{|f'''''(x)|}{30}h^4$$

- small values of hlead to large round-off errors

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

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

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

```
h_opt =
     7.4418e-04
```

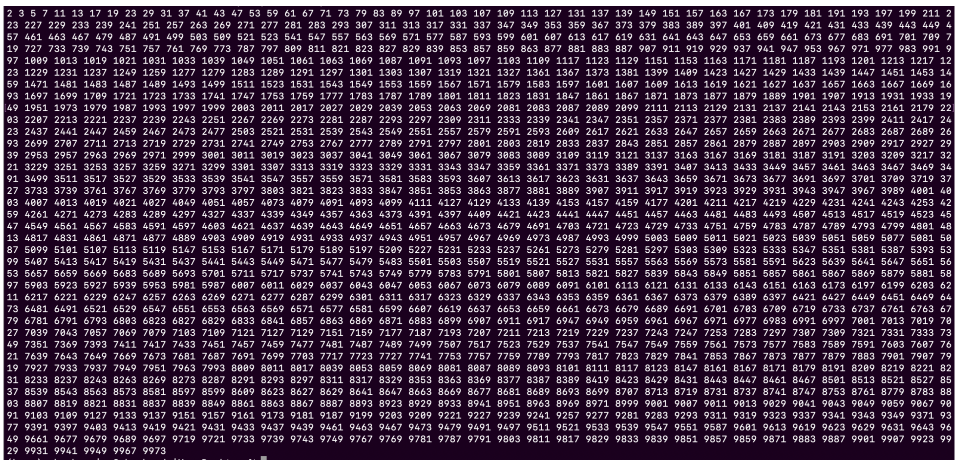
which justifies the result from (b).

Problem 2

The Perl code is given as following.

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

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

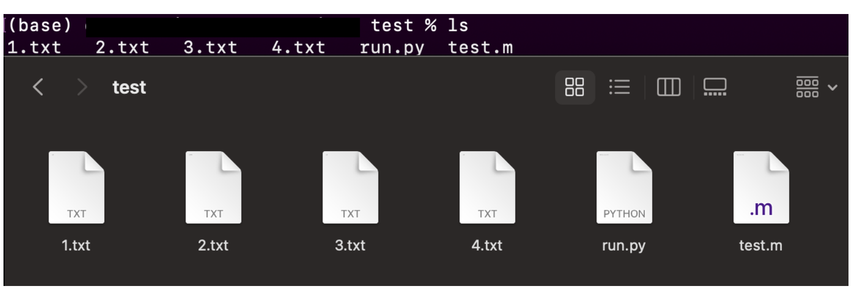


Problem 3

The awk command is shown as follows.

```
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 test this command in the following folder:



After running the awk command like this:

```
(base) [redacted] : 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:

```
filesize.dat
17   ,    test.m
21   ,    run.py
4    ,    4.txt
4    ,    3.txt
8    ,    2.txt
4    ,    1.txt

58
```

Problem 4

The Perl code is given as following.

```perl
#!/usr/bin/perl -w
use strict;

my $file = "data.txt";
my $fmt = "%.16e\t%.16e\t%.16e\n"; # print format: 2 tab-separated numbers
open(BDW,">$file"); # open new file for writing, with handle/ref BDW
for (my $i=1;$i<=4;$i++){
    for (my $j=1;$j<=4;$j++){
            my $x = 1/8 + ($i-1)*1/4;
            my $y = 1/8 + ($j-1)*1/4;
            my $f = exp(-$x*$x - $y*$y);
            printf(BDW $fmt,$x,$y,$f);
    }
}
close BDW; # close file using handle

sub readAndCompute{
    my $errmsg = "Error: cannot open $file"; # define error message
    open(BDW,"<$file") or die($errmsg);

    my $sum = 0;
    while (<BDW>) { # read one line at a time
            chomp;
            my ($x,$y,$f) = split("\t",$_); # get $x and $y from the 2 fields
            $sum = $sum + $f
    }
    close BDW;

    printf("The approximation of intergral is %.16e\n", $sum/16);
}

readAndCompute();
```

which gives the result as

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

Problem 5

The recursive code in MATLAB is shown as following.

```
% A:source, B:help, C:target
% initial state: all disks are in source peg.
X = ['A', 'A', 'A', 'A'];

Hanoi(X, length(X), 'A','C','B');
function x = Hanoi(x, n, source, target, help)
    if n > 0
        % move n-1 disks from source to help so they are out of the way
        x = Hanoi(x, n-1, source, help, target);
        % move the nth (largest) disk to target
        x(n) = target
        % move the n-1 disks on help to target
        x = Hanoi(x, n-1, help, target, source);
    end
end
```

which gives the result as

```
x =
    'BAAA'

x =
    'BCAA'

x =
    'CCAA'

x =
    'CCBA'

x =
    'ACBA'

x =
    'ABBA'

x =
    'BBBA'

x =
    'BBBC'

x =
    'CBBC'

x =
    'CABC'

x =
    'AABC'

x =
    'AACC'

x =
    'BACC'

x =
    'BCCC'

x =
    'CCCC'
```

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