CSC336 Assignment 1

Rui Qiu c3qiurui
2016-02-08

Problem 1

(a) Solution:

The condition number is $\kappa_f = \left|\frac{xf'(x)}{f(x)}\right| = \left|\frac{x\frac{1}{4}(a+x)^{-3/4}}{(a+x)^{1/4}-a^{1/4}}\right|$. Consider two cases where $x \to 0^+$ and $x \to +\infty$ so that we will have κ_f in form of $\frac{0}{0}$ and $\frac{\infty}{\infty}$, then we will apply L'Hospital's rule:

$$
\begin{aligned}
\lim_{x\to\infty} |\kappa_f| &= \lim_{x\to\infty} \left| \frac{\frac{1}{4}(a+x)^{-3/4} + x\frac{1}{4}\frac{-3}{4}(a+x)^{-7/4}}{\frac{1}{4}(a+x)^{-3/4}} \right| \\
&= 1 + \lim_{x\to\infty} \left| \frac{-3x}{4}(a+x)^{-1} \right| \\
&= 1 + \lim_{x\to\infty} \frac{3}{4} \left| \frac{x}{a+x} \right| \\
&= \frac{7}{4} \\
\lim_{x\to 0} |\kappa_f| &= 1
\end{aligned}
$$

Therefore, both condition numbers are not very large, so there are **no ill-conditioned** in the range of $x \in \mathbf{R}^+$.

(b) Solution:

Consider 5-decimal-digit floating numbers, with traditional rounding. Let $x_1 = 0.0006 \times 10^0, x_2 = 0.0007 \times 10^0, a = 0.1 \times 10^1$.

$$
\begin{aligned}
f(x_1) &= (0.1 \times 10^1 + 0.0006)^{1/4} - (0.1 \times 10^1)^{1/4} \\
&= 1.0001 \times 10^0 - 1 \times 10^0 \\
&= 0.0001 \\
f(x_2) &= (0.1 \times 10^1 + 0.0007)^{1/4} - (0.1 \times 10^1)^{1/4} \\
&= 1.0002 - 1 \\
&= 0.0002 \\
\therefore \frac{f(x_2) - f(x_1)}{x_2 - x_1} &= \frac{0.0001}{0.0001} = 10^0
\end{aligned}
$$

The idea about stability of computation is to measure the relative change $\Delta f(x)$ over Δx. In this case, 1 unit of change in x leads to 1 unit change in $f(x)$ so that the computation is rather sensitive.

A possibly **more stable computation** can be expressed by the following mathematically equivalent expression:

$$
\begin{aligned}
\frac{1}{f} &= \frac{1}{\sqrt[4]{a+x} - \sqrt[4]{a}} \\
&= \frac{\sqrt[4]{a+x} + \sqrt[4]{a}}{(\sqrt[4]{a+x} - \sqrt[4]{a})(\sqrt[4]{a+x} + \sqrt[4]{a})} \\
&= \frac{(\sqrt[4]{a+x} + \sqrt[4]{a})(\sqrt{a+x} + \sqrt{a})}{(\sqrt{a+x} - \sqrt{a})(\sqrt{a+x} + \sqrt{a})} \\
&= \frac{(\sqrt[4]{a+x} + \sqrt[4]{a})(\sqrt{a+x} + \sqrt{a})}{a+x-a} \\
&= \frac{(\sqrt[4]{a+x} + \sqrt[4]{a})(\sqrt{a+x} + \sqrt{a})}{x} \\
\therefore f(x) &= \frac{x}{(\sqrt[4]{a+x} + \sqrt[4]{a})(\sqrt{a+x} + \sqrt{a})}
\end{aligned}
$$

Why is this stable? It decreases the negative effect caused by subtracting nearly equal numbers, i.e. $\sqrt[4]{a+x} - \sqrt[4]{a}$ as $x \to 0$. In the new expression, there is no subtraction inside.

Specifically, if we plug in x_1, x_2, a in the new expression:

$$
\begin{aligned}
f(x) &= \frac{x}{(\sqrt[4]{a+x} + \sqrt[4]{a})(\sqrt{a+x} + \sqrt{a})} \\
f(x_1) &= f(x_2) = 0.0001 \\
\frac{f(x_2) - f(x_1)}{x_2 - x_1} &= 0
\end{aligned}
$$

(c) Solution:

```
format compact
%f(x) = (a+x)^(1/4) - a^(1/4)
%f'(x) = 1/4*(a+x)^(-3/4)

a = 1;
x = 10.^[-20:1:20];
f0 = (a+x).^(1/4) - a^(1/4);
fp = 1/4*(a+x).^(-3/4);
k0 = x.*fp./f0;

f1 = x./(((a+x).^(1/4) + a^(1/4)).*((a+x).^(1/2) + a^(1/2)));
k1 = x.*fp./f1;

disp('x           f0             f1           k0             k1             rel.err.')
for i = 1:41
    fprintf('%9.2e %12.5e %12.5e %12.5e %12.5e %10.2e\n', ...
        x(i), f0(i), f1(i), k0(i), k1(i), (f0(i)-f1(i))/f1(i));
end
```

The results are:

```
>> q1
x           f0             f1           k0             k1             rel.err.
 1.00e-20  0.00000e+00  2.50000e-21           Inf  1.00000e+00  -1.00e+00
 1.00e-19  0.00000e+00  2.50000e-20           Inf  1.00000e+00  -1.00e+00
 1.00e-18  0.00000e+00  2.50000e-19           Inf  1.00000e+00  -1.00e+00
 1.00e-17  0.00000e+00  2.50000e-18           Inf  1.00000e+00  -1.00e+00
 1.00e-16  0.00000e+00  2.50000e-17           Inf  1.00000e+00  -1.00e+00
 1.00e-15  2.22045e-16  2.50000e-16  1.12590e+00  1.00000e+00  -1.12e-01
 1.00e-14  2.44249e-15  2.50000e-15  1.02355e+00  1.00000e+00  -2.30e-02
 1.00e-13  2.48690e-14  2.50000e-14  1.00527e+00  1.00000e+00  -5.24e-03
 1.00e-12  2.50022e-13  2.50000e-13  9.99911e-01  1.00000e+00   8.89e-05
 1.00e-11  2.50000e-12  2.50000e-12  1.00000e+00  1.00000e+00   8.27e-08
 1.00e-10  2.50000e-11  2.50000e-11  1.00000e+00  1.00000e+00   8.28e-08
 1.00e-09  2.50000e-10  2.50000e-10  1.00000e+00  1.00000e+00   8.31e-08
 1.00e-08  2.50000e-09  2.50000e-09  1.00000e+00  1.00000e+00  -2.33e-09
 1.00e-07  2.50000e-08  2.50000e-08  1.00000e+00  1.00000e+00   4.78e-09
 1.00e-06  2.50000e-07  2.50000e-07  1.00000e+00  1.00000e+00   3.28e-10
 1.00e-05  2.49999e-06  2.49999e-06  9.99996e-01  9.99996e-01  -3.78e-11
 1.00e-04  2.49991e-05  2.49991e-05  9.99963e-01  9.99963e-01   1.02e-12
 1.00e-03  2.49906e-04  2.49906e-04  9.99625e-01  9.99625e-01   3.15e-14
 1.00e-02  2.49068e-03  2.49068e-03  9.96279e-01  9.96279e-01   1.74e-16
 1.00e-01  2.41137e-02  2.41137e-02  9.65232e-01  9.65232e-01  -5.76e-16
 1.00e+00  1.89207e-01  1.89207e-01  7.85652e-01  7.85652e-01  -2.93e-16
 1.00e+01  8.21160e-01  8.21160e-01  5.04043e-01  5.04043e-01   1.35e-16
 1.00e+02  2.17015e+00  2.17015e+00  3.61583e-01  3.61583e-01   0.00e+00
 1.00e+03  4.62482e+00  4.62482e+00  3.03752e-01  3.03752e-01  -1.92e-16
 1.00e+04  9.00025e+00  9.00025e+00  2.77749e-01  2.77749e-01  -1.97e-16
 1.00e+05  1.67828e+01  1.67828e+01  2.64894e-01  2.64894e-01   2.12e-16
 1.00e+06  3.06228e+01  3.06228e+01  2.58164e-01  2.58164e-01   1.16e-16
 1.00e+07  5.52341e+01  5.52341e+01  2.54526e-01  2.54526e-01   1.29e-16
 1.00e+08  9.90000e+01  9.90000e+01  2.52525e-01  2.52525e-01   0.00e+00
```

32	1.00e+09	1.76828e+02	1.76828e+02	2.51414e-01	2.51414e-01	0.00e+00
33	1.00e+10	3.15228e+02	3.15228e+02	2.50793e-01	2.50793e-01	0.00e+00
34	1.00e+11	5.61341e+02	5.61341e+02	2.50445e-01	2.50445e-01	-2.03e-16
35	1.00e+12	9.99000e+02	9.99000e+02	2.50250e-01	2.50250e-01	0.00e+00
36	1.00e+13	1.77728e+03	1.77728e+03	2.50141e-01	2.50141e-01	1.28e-16
37	1.00e+14	3.16128e+03	3.16128e+03	2.50079e-01	2.50079e-01	0.00e+00
38	1.00e+15	5.62241e+03	5.62241e+03	2.50044e-01	2.50044e-01	0.00e+00
39	1.00e+16	9.99900e+03	9.99900e+03	2.50025e-01	2.50025e-01	-1.82e-16
40	1.00e+17	1.77818e+04	1.77818e+04	2.50014e-01	2.50014e-01	0.00e+00
41	1.00e+18	3.16218e+04	3.16218e+04	2.50008e-01	2.50008e-01	0.00e+00
42	1.00e+19	5.62331e+04	5.62331e+04	2.50004e-01	2.50004e-01	1.29e-16
43	1.00e+20	9.99990e+04	9.99990e+04	2.50003e-01	2.50003e-01	0.00e+00

Explanation: As we can see in the table above, the fourth column `k0` indicates the conditional number of original expression. When x is small enough, in this case, $x = 10^{-16}, 10^{-17}, \dots, 10^{-20}$, the conditional number explodes to infinity. Thus, comparatively speaking, the alternative expression we propose in part (b) does not have such problem, i.e., it is a better way to calculate.

The reason why infinity appears is because when the computer tries to subtract one number from another nearly equal number, the difference tends to be extremely small. And it is small enough that would be rounded off by computer, treated as a zero. If such zero exists in a denominator, the infinity appears.

However, as mentioned in part (b), the alternative expression avoids the subtraction of two nearly equal values, but is mathematically equivalent.

Problem 2

(a) Solution:

Taylor's expansion about 0 for $\sin(\frac{\pi}{2}x)$:

$$\sin(\frac{\pi}{2}x) = \frac{\pi}{2}x - (\frac{\pi}{2}x)^3\frac{1}{3!} + (\frac{\pi}{2}x)^5\frac{1}{5!} - \cdots = \sum_{k=0}^{\infty}(-1)^k\frac{(\frac{\pi}{2}x)^{2k+1}}{(2k+1)!}.$$

The remainder of this Taylor expansion is:

$$R_{k+1}(\frac{\pi}{2}x) = \frac{f^{(k+1)}(\xi)}{(k+1)!}(\frac{\pi}{2}x - 0)^{k+1},$$

as $\xi \in \left[0, \frac{\pi}{2}x\right]$, $x \in \left[-1, 1\right]$, and (k+1)-th derivative of f could be any of the following:

$$\sin(\frac{\pi}{2}x), \cos(\frac{\pi}{2}x), -\sin(\frac{\pi}{2}x), -\cos(\frac{\pi}{2}x)$$

multiplies the (k+1)-th power of $\frac{\pi}{2}$. For example:

$$\begin{aligned} f^{(0)} &= \sin(\frac{\pi}{2}x) \\ f^{(1)} &= \cos(\frac{\pi}{2}x)\ \frac{\pi}{2} \\ f^{(2)} &= -\sin(\frac{\pi}{2}x)(\frac{\pi}{2})^2 \\ f^{(3)} &= -\cos(\frac{\pi}{2}x)(\frac{\pi}{2})^3 \\ &\dots \end{aligned}$$

Because the values of $\left|\sin(\frac{\pi}{2}x)\right|$, $\left|\cos(\frac{\pi}{2}x)\right|$, $\left|-\sin(\frac{\pi}{2}x)\right|$, $\left|-\cos(\frac{\pi}{2}x)\right|$ are between $\left[0, 1\right]$, so

$$0 \leq \left|f^{(k+1}(\xi)\right| \leq (\frac{\pi}{2})^{k+1},$$

Therefore, the bound of the remainder is:

$$0 \leq \left|R_{k+1}(\frac{\pi}{2}x)\right| \leq \left|f^{(k+1)}(\xi)\right|(\frac{\pi}{2})^{k+1}/(k+1)! = \frac{(\frac{\pi}{2})^{2k+2}}{(k+1)!}.$$

Now we need to estimate the number of terms in the series which guaranteed 6 significant decimal digits correct.

Recall an approximation $\hat{x}$ to x is said to be correct in r significant b-digits if $\left|\frac{x-\hat{x}}{x}\right| \leq \frac{1}{2}b^{1-r}$.

In this case,

- x is the real value of $\sin(\frac{\pi}{2}x)$;
- $x - \hat{x}$ is the remainder we wrote above;
- $\frac{1}{2}b^{1-r} = \frac{1}{2} \times 10^{1-6} = 0.5 \times 10^{-5}$.

$$\begin{aligned} &\left|\frac{R_{k+1}(\pi x/2)}{\sin(\frac{\pi}{2}x)}\right| \leq 0.5 \times 10^{-5} \\ 0 \leq &\left|R_{k+1}(\frac{\pi}{2}x)\right| \leq \frac{(\frac{\pi}{2}x)^{2k+2}}{(k+1)!} \\ 0 \leq &\left|\sin(\frac{\pi}{2}x)\right| \leq 1 \end{aligned}$$

So when $x \to 0$, both numerator and denominator approaches to 0, so we can apply L'Hospital's rule again:

$$\lim_{x \to 0} \frac{(2k+2)(\frac{\pi}{2}x)^{2k+1}/(k+1)!}{\cos(\frac{\pi}{2}x)\frac{\pi}{2}} = 0 \leq 0.5 \times 10^{-5}$$

Similarly when $x \to 1$ ($x \to -1$ likewise because x is calculated in absolute value), the expression becomes:

$$\left| \frac{R_{k+1}(\pi x/2)}{\sin(\frac{\pi}{2}x)} \right| \leq \frac{(\frac{\pi}{2})^{2k+2}}{(k+1)!} \leq 0.5 \times 10^{-5}$$

By trial-and-error:

$\ldots$
$k = 10, (\frac{\pi}{2})^{2k+2}/(k+1)! = 0.0005169893 > 0.5 \times 10^{-5};$
$k = 11, (\frac{\pi}{2})^{2k+2}/(k+1)! = 0.0001063017 > 0.5 \times 10^{-5};$
$k = 12, (\frac{\pi}{2})^{2k+2}/(k+1)! = 2.017607e - 05 > 0.5 \times 10^{-5};$
$k = 13, (\frac{\pi}{2})^{2k+2}/(k+1)! = 3.555889e - 06 < 0.5 \times 10^{-5};$
$\ldots$

Hence, we need at least 13 terms in order to get 6 significant decimal digits correct.

Problem 3

(a) Solution:
Use integration by parts:

$$
\begin{aligned}
\int udv &= uv - \int vdu \\
u = t^n, dv &= e^{-t}dt, du = nt^{n-1}dt, v = -e^{-t} \\
\therefore y_n &= 1^n(-e^{-1}) - 0^n(-e^{-0}) - \int_0^1 -e^{-t}nt^{n-1}dt \\
&= -\frac{1}{e} + n\int_0^1 e^{-t}t^{n-1}dt \\
&= -\frac{1}{e} + ny_{n-1} \\
\therefore y_{n-1} &= \frac{y_n + \frac{1}{e}}{n}.
\end{aligned}
$$

So

- $y_n = -\frac{1}{e} + ny_{n-1}$ is recurrence formula (A) and
- $y_{n-1} = \frac{y_n + 1/e}{n}$ is recurrence formula (B).

(b) Solution:

$$
\begin{aligned}
n = 2, y_2 &= f_2(y_0) = -\frac{1}{e} + 2y_1 = -\frac{1}{e} + 2(-\frac{1}{e} + y_0) = (-\frac{1}{e})(\frac{2!}{1!} + \frac{2!}{2!}) + 2!y_0, \\
n = 3, y_3 &= f_3(y_0) = -\frac{1}{e} + 3((1+2)(-\frac{1}{e}) + 2y_0) = (-\frac{1}{e})(\frac{3!}{1!} + \frac{3!}{2!} + \frac{3!}{3!}) + 3!y_0, \\
&\dots \\
\therefore y_n &= f_n(y_0) = -\frac{1}{e}\sum_{k=1}^{n}\frac{n!}{k!} + n!y_0.
\end{aligned}
$$

$$n = m-1, y_{m-1} = \frac{1}{e}\frac{(m-1)!}{m!} + \frac{(m-1)!}{m!} y_m,$$

$$n = m-2, y_{m-2} = \frac{\frac{y_m + \frac{1}{e}}{m} + \frac{1}{e}}{m-1} = \frac{1}{e}\left(\frac{(m-2)!}{m!} + \frac{(m-2)!}{(m-1)!}\right) + \frac{(m-2)!}{m!} y_m$$

$$n = m-3, y_{m-3} = \frac{1}{e}\left(\frac{(m-3)!}{m!} + \frac{(m-3)!}{(m-1)!} + \frac{(m-3)!}{(m-2)!}\right) + \frac{(m-3)!}{m!} y_m$$

$$\ldots$$

$$\therefore y_n = g_{n,m}(y_m) = \frac{1}{e}\sum_{k=n+1}^{m}\frac{n!}{k!} + \frac{n!}{m!} y_m.$$

(c) Solution:

$$\kappa_{f_n(y_0)} = \left|\frac{y_0 f_n'(y_0)}{f_n(y_0)}\right| = \left|\frac{y_0 n!}{f_n(y_0)}\right| = \left|\frac{y_0 n!}{-\frac{1}{e}\sum_{k=1}^{n}\frac{n!}{k!} + n! y_0}\right|$$

In order to observe the values with high n, we wrote a small R-function as a helper:

```r
s <- function(n) {
  res <- 0
  for (i in 1:n) {
    res <- res + (factorial(n)/factorial(i))
  }
  return(res)
}
```

```r
> factorial(50)/s(50)
[1] 0.5819767
> factorial(150)/s(150)
[1] 0.5819767
> factorial(160)/s(160)
[1] 0.5819767
> 0.581976*(-1/exp(1))
[1] -0.214097
```

So we see that the summation part in the denominator of condition number of $f_n(y_0)$ is constant, so as $n \to \infty$, the condition number of $f_n(y_0)$, $\kappa_{f_n(y_0)} \to 1$ (but note that since the numerator is larger than denominator, the real value of $\kappa_{f_n(y_0)} > 1$.

$$\kappa_{g_{n,m}(y_m)} = \left|\frac{y_m g'_{n,m}(y_m)}{g_{n,m}(y_m)}\right| = \left|\frac{y_m \frac{n!}{m!}}{g_{n,m}(y_m)}\right| = \left|\frac{y_m \frac{n!}{m!}}{\frac{1}{e}\sum_{k=n+1}^{m}\frac{n!}{k!} + \frac{n!}{m!} y_m}\right|$$

Similarly, when n is large enough, the conditional number of $g_{n,m}(y_m)$ also approaches 1 but it is less than 1 since the denomiator is larger than numerator, i.e., $\kappa_{g_{n,m}(y_m)} < 1$.

We compare two condition numbers which are pretty similar around value 1, but the second condition number is smaller than 1 while the first condition number is greater than 1. So we prefer the second method for computing $y_0, y_1, \cdots, y_N$, given y_N,which seems more stable.

(d) Solution:

We can plug in $n = 0$ to compute y_0 simply:

$$\int_0^1 t^0 e^{-t} dt = -e^{-1} + e^0 = 0.6321206 = y_0.$$

`method1.m`

```
function [] = method1(y0)
N = 20;
for i = 1:N
    if i == 1
        previous = y0;
    end
    y(i) = -1/exp(1) + i * previous;
    fprintf('%3d %20.16f\n', i-1, y(i));
    previous = y(i);
end
end
```

Use $y_0 = 0.6321206$, the result is:

```

>> method1(0.6321206)
  0   0.2642411588285578
  1   0.1606028764856732
  2   0.1139291882855774
  3   0.0878373119708674
  4   0.0713071186828945
  5   0.0599632709259248
  6   0.0518634553100317
  7   0.0470282013088111
  8   0.0553743706078574
  9   0.1858642649071321
 10   1.6766274728070107
 11  19.7516502325126844
```

```
12 256.4035735814934469
13 3589.2821506997370307
14 53838.8643810548892361
15 861421.4622174371033907
16 14644164.4898169897496700
17 263594960.4488263726234436
18 5008304248.1598215103149414
19 100166084962.8285522460937500
```

But if we round $y_0 \approx 0.63212$, the result, is however very different:

```
>> method1(0.63212)
  0   0.2642405588285577
  1   0.1606016764856732
  2   0.1139255882855773
  3   0.0878229119708669
  4   0.0712351186828925
  5   0.0595312709259124
  6   0.0488394553099447
  7   0.0228362013081154
  8  -0.1623536293984035
  9  -1.9914157351554771
 10 -22.2734525278816911
 11 -267.6493097757517603
 12 -3479.8089065259441668
 13 -48717.6925708043854684
 14 -730765.7564415069064125
 15 -11692252.4709435515105724
 16 -198768292.3739198148250580
 17 -3577829263.0984358787536621
 18 -67978755999.2381668090820312
 19 -1359575119985.1313476562500000
```

What happens: As we can observe, a tiny difference in initial value y_0 could lead to an shocking difference in y_0. What's more is that it is not even because of a "guess" upon initial value, but rounding of it. Therefore, we can conclude that this method of computing y_N from y_0 is unstable hence not recommended.

(e) Solution:

`method2.m`

```
function [] = method2(yNK)
q=0.0183504676972562063264;
digits(40);
for K = (3:1:9)
    for i = 1:K
        if i == 1
            previous = yNK;
        end
        y(21+K-i) = (previous + 1/exp(1))/(i+1);
        previous = y(21+K-i);
    end
```

```
    fprintf('%3d %20.16f %20.16f %10.6e\n', 20, y(21), q, q-y(21));
end
end
```

Considering the problem we encountered in part (d), we might want to have 3 some different initial values $y_{N+K} = y_{20+K} = 25, 1, 2000$, where $K = 3, 4, \dots, 9$, then the results are:

```
>> method2(25)
 20   1.1796214571059576   0.0183504676972562 -1.161271e+00
 20   0.3095001796554800   0.0183504676972562 -2.911497e-01
 20   0.1128966034711537   0.0183504676972562 -9.454614e-02
 20   0.0686822920917994   0.0183504676972562 -5.033182e-02
 20   0.0545702166579052   0.0183504676972562 -3.621975e-02
 20   0.0469388508699275   0.0183504676972562 -2.858838e-02
 20   0.0414818292041370   0.0183504676972562 -2.313136e-02

 >> method2(1)
 20   0.1796214571059575   0.0183504676972562 -1.612710e-01
 20   0.1095001796554800   0.0183504676972562 -9.114971e-02
 20   0.0795632701378204   0.0183504676972562 -6.121280e-02
 20   0.0639203873298947   0.0183504676972562 -4.556992e-02
 20   0.0539749785626671   0.0183504676972562 -3.562451e-02
 20   0.0468727133037899   0.0183504676972562 -2.852225e-02
 20   0.0414752154475232   0.0183504676972562 -2.312475e-02

 >> method2(2000)
 20  83.4712881237726236   0.0183504676972562 -8.345294e+01
 20  16.7678335129888119   0.0183504676972562 -1.674948e+01
 20   2.8559521590267090   0.0183504676972562 -2.837602e+00
 20   0.4605473714568787   0.0183504676972562 -4.421969e-01
 20   0.1035533515785401   0.0183504676972562 -8.520288e-02
 20   0.0523814214166647   0.0183504676972562 -3.403095e-02
 20   0.0420260862588107   0.0183504676972562 -2.367562e-02
```

Comments: Note that the 4 columns above are (from left to right):

- 20,
- approximate value of y_{20} (although in function it is written as `y(21)`),
- real value of y_{20} and
- the error of y_{20}.

With $N = 20$ given, we choose 7 different K to calculate from y_{N+K} backward to y_N. And no matter what initial value y_{N+K} is equal to, the second column always indicates that as K increases, the estimated y_{20} decreases, and the fourth column confirms that the difference between exact value and our estimation is getting smaller and smaller.

Thus if N is given, we want to start from a long distance at y_{N+K} (with rather large positive K), compute iteratively backward to the targeted y_N, in order to generate more accuracy.

Specifically speaking, in this case, if we want to compute y_{20}, then we should pick $K = 9$. And of course, we should choose method 2 instead of method 1 in part (d).