

1. (a) There are two places that can cause potential subtractive cancellation: $\beta - \sqrt{\beta^2 - x^2}$, the two minus sign highlighted in function.

① For the first '-' sign $\beta - \sqrt{\beta^2 - x^2}$, if $x \approx 0$, then $f(x) \approx 0$

② For the second '-' sign, only need to look at $\beta^2 - x^2$, which is if $x \approx \beta$, then $f(x) \approx \beta$, but this case is insignificant, because $\beta \gg 0$ and $|x| < \beta$, so $\beta \neq 0 \Rightarrow f(x) \neq 0$, which means this case is insignificant.

$$(b) \text{ ① } \alpha \approx 0, f'(x) = \frac{-2x}{2\sqrt{\beta^2 - x^2}} = \frac{x}{\sqrt{\beta^2 - x^2}}$$

$$\begin{aligned} \lim_{x \rightarrow 0} \text{cond}(f) &= \lim_{x \rightarrow 0} \left| \frac{x \cdot f'(x)}{f(x)} \right| \\ &= \lim_{x \rightarrow 0} \left| \frac{x \cdot \left(\frac{x}{\sqrt{\beta^2 - x^2}} \right)}{\beta - \sqrt{\beta^2 - x^2}} \right| \\ &= \lim_{x \rightarrow 0} \left| \frac{x^2}{\beta\sqrt{\beta^2 - x^2} - \beta^2 + x^2} \right| \quad (\beta \gg 0) \\ &= 0 \quad \text{well-conditioned} \end{aligned}$$

Since when $\alpha \approx 0$, $f(x)$ is well-conditioned, it doesn't the cancellation error

$$\begin{aligned} (c) \quad & \beta - \sqrt{\beta^2 - x^2} \left(\frac{\beta + \sqrt{\beta^2 - x^2}}{\beta + \sqrt{\beta^2 - x^2}} \right) \\ &= \frac{\beta^2 - (\beta^2 - x^2)}{\beta + \sqrt{\beta^2 - x^2}} \\ &= \frac{\beta^2 - \beta^2 + x^2}{\beta + \sqrt{\beta^2 - x^2}} \end{aligned}$$

$$= \frac{x^2}{\beta + \sqrt{\beta^2 - x^2}}$$

(2) From (b) and (c), I find that if the condition number of an expression which suffering from subtractive cancellation is small (expression is well-conditioned), then there exists a alternate form of that expression which doesn't have subtractive cancellation issue.

2. Let $\delta_{x/z}$ denote the δ for $-1/(f(x) \div f(z))$

$$\begin{aligned}
 & f\left(\left(f\left(f(x) \div f(z)\right)\right) \cdot f(y)\right) \\
 &= \left[\left(\frac{x(1-\delta_x)}{z(1-\delta_z)} (1-\delta_{x/z})\right) \cdot y(1-\delta_y)\right] (1-\delta_{(x/z) \cdot y}) \\
 &= \left[\left(\frac{x}{z} \cdot \frac{1-\delta_x}{1-\delta_z} \cdot \frac{1+\delta_z}{1+\delta_z} (1-\delta_{x/z})\right) \cdot y(1-\delta_y)\right] (1-\delta_{(x/z) \cdot y}) \\
 &= \left[\left(\frac{x}{z} \cdot \frac{1-\delta_x+\delta_z-\delta_x\delta_z}{1-\delta_z^2} (1-\delta_{x/z})\right) \cdot y(1-\delta_y)\right] (1-\delta_{(x/z) \cdot y}) \\
 &= \left(\frac{x}{z}\right) \cdot y (1-\delta_x+\delta_z-\delta_{x/z}-\delta_y-\delta_{(x/z) \cdot y}) \quad \text{Note: } \delta < \epsilon \\
 &= \left(\frac{x}{z}\right) \cdot y (1-\delta_{(1)}) \quad |\delta^2| \ll \delta \\
 & \left(\frac{x}{z}\right) \cdot y (1-\delta_{(1)}) = \left(\frac{x}{z}\right) \cdot y (1-\delta_x+\delta_z-\delta_{x/z}-\delta_y-\delta_{(x/z) \cdot y}) \quad |\delta^3| \ll \delta \\
 & 1-\delta_{(1)} = 1-\delta_x+\delta_z-\delta_{x/z}-\delta_y-\delta_{(x/z) \cdot y} \quad |\delta^4| \ll \delta \\
 & \delta_{(1)} = \delta_x - \delta_z + \delta_{x/z} + \delta_y + \delta_{(x/z) \cdot y} \quad |\delta^5| \ll \delta \\
 & |\delta_{(1)}| \leq |\delta_x| + |\delta_z| + |\delta_{x/z}| + |\delta_y| + |\delta_{(x/z) \cdot y}| \quad \text{by triangle equality} \\
 & |\delta_{(1)}| \leq 5\epsilon
 \end{aligned}$$

$$\begin{aligned}
 3. \quad \|x\|_p &= \frac{\|x\|_\infty}{\|x\|_\infty} \|x\|_p \\
 &= \|x\|_\infty \left(\frac{\|x\|_p}{\|x\|_\infty} \right) \\
 &= \|x\|_\infty \left(\frac{(\|x\|_p^p)^{\frac{1}{p}}}{\|x\|_\infty} \right) \\
 &= \|x\|_\infty \left(\frac{(\sum_{i=1}^n |x_i|^p)^{\frac{1}{p}}}{\|x\|_\infty} \right) \\
 &= \|x\|_\infty \left(\frac{\sum_{i=1}^n |x_i|^p}{\|x\|_\infty^p} \right)^{\frac{1}{p}}
 \end{aligned}$$

$$= \|x\|_\infty \left(\sum_{i=1}^n \frac{|x_i|^p}{\|x\|_\infty^p} \right)^{\frac{1}{p}}$$

$$= \|x\|_\infty \left(\sum_{i=1}^n \left(\frac{|x_i|}{\|x\|_\infty} \right)^p \right)^{\frac{1}{p}}$$

$$\leq \|x\|_\infty \cdot n^{\frac{1}{p}} \quad \left(\text{b/c } \|x\|_\infty = \max_{1 \leq i \leq n} |x_i| \text{ by definition, so } \forall i, |x_i| \leq \|x\|_\infty \text{ therefore } \frac{|x_i|}{\|x\|_\infty} \leq 1 \Leftrightarrow \left(\frac{|x_i|}{\|x\|_\infty} \right)^p \leq 1^p = 1 \Leftrightarrow \sum_{i=1}^n \left(\frac{|x_i|}{\|x\|_\infty} \right)^p \leq n \right)$$

$$\therefore \|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$$

$$\therefore \|x\|_\infty = \left(\max_{1 \leq i \leq n} |x_i|^p \right)^{\frac{1}{p}}$$

$$\leq \left(\sum_{i=1}^n |x_i|^p \right)^{\frac{1}{p}}$$

$$= \|x\|_p$$

$$\left(\text{b/c it is clear that } \max_{1 \leq i \leq n} |x_i|^p \leq \sum_{i=1}^n |x_i|^p \right)$$

$$\therefore \|x\|_\infty \leq \|x\|_p \leq \|x\|_\infty \cdot n^{\frac{1}{p}}$$

Take limit to the above inequality

$$\lim_{p \rightarrow \infty} \|x\|_\infty \leq \lim_{p \rightarrow \infty} \|x\|_p \leq \lim_{p \rightarrow \infty} (\|x\|_\infty \cdot n^{\frac{1}{p}})$$

$$\Rightarrow \|x\|_\infty \leq \lim_{p \rightarrow \infty} \|x\|_p \leq \|x\|_\infty \cdot n^{\frac{1}{\infty}}$$

$$\Rightarrow \|x\|_\infty \leq \lim_{p \rightarrow \infty} \|x\|_p \leq \|x\|_\infty \cdot n^0 \quad (\text{b/c } \frac{1}{\infty} = 0)$$

$$\Rightarrow \|x\|_\infty \leq \lim_{p \rightarrow \infty} \|x\|_p \leq \|x\|_\infty \cdot 1$$

$$\Rightarrow \|x\|_\infty \leq \lim_{p \rightarrow \infty} \|x\|_p \leq \|x\|_\infty$$

by squeeze theorem,

$$\|x\|_\infty = \lim_{p \rightarrow \infty} \|x\|_p$$

QED

4.(a)

The screenshot shows the MATLAB Editor with the following code in `untitled2.m`:

```

1 B = [1 2 3; 4 5 6; 2 4 5];
2 % realmax = 1.7977e+308
3 smaller_than_realmax = realmax - 0.0002e+308;
4 C = [1 2 3; 4 5 smaller_than_realmax; 2 4 5];
5 % realmin = 2.2251e-308
6 larger_than_realmin = realmin + 0.0001e-308;
7 A = [larger_than_realmin larger_than_realmin; larger_than_realmin larger_than_realmin];
8 x = frobenius(B)
9 function x = frobenius(A)
10     x = sqrt(sum(sum(A.^2)));
11 end

```

Handwritten red annotations include "matrix used in this test" pointing to line 1, and "function" pointing to the function definition on lines 9-11.

The Command Window shows the result of running the script:

```

>> untitled2
x =
    11.6619

```

A handwritten red "result" points to the value 11.6619.

$$\begin{aligned}
 \|B\|_F &= \sqrt{1^2 + 2^2 + 3^2 + 4^2 + 5^2 + 6^2 + 2^2 + 4^2 + 5^2} \\
 &= \sqrt{(1+4+9+16+25+36+4+16+25)} \\
 &= \sqrt{136} \\
 &= 11.6619 \text{ (rounded)}
 \end{aligned}$$

(b) (i) overflow case

The screenshot shows the MATLAB Editor with the following code in `untitled2.m`:

```

1 B = [1 2 3; 4 5 6; 2 4 5];
2 % realmax = 1.7977e+308
3 smaller_than_realmax = realmax - 0.0002e+308;
4 C = [1 2 3; 4 5 smaller_than_realmax; 2 4 5];
5 % realmin = 2.2251e-308
6 larger_than_realmin = realmin + 0.0001e-308;
7 A = [larger_than_realmin larger_than_realmin; larger_than_realmin larger_than_realmin];
8 x = frobenius(C)
9 y = norm(C, "fro")
10 function x = frobenius(A)
11     x = sqrt(sum(sum(A.^2)));
12 end
13

```

Handwritten red annotations include "function" pointing to the function definition on lines 10-12. A green annotation "The build-in for F-norm" points to line 9.

The Command Window shows the results of running the script:

```

x =
    Inf
y =
    1.7975e+308

```

Handwritten red "incorrect result" points to the value Inf, and a green "correct result" points to the value 1.7975e+308.

frobenius(C) gives inf because when the function squares smaller-than-realmax, the squared number overflows, but correct result shouldn't overflow because there is a square root at the end of computation.

② underflow

The screenshot shows the MATLAB editor with the following code in `untitled2.m`:

```

1 B = [1 2 3; 4 5 6; 2 4 5];
2 % realmax = 1.7977e+308
3 smaller_than_realmax = realmax - 0.0002e+308;
4 C = [1 2 3; 4 5 smaller_than_realmax; 2 4 5];
5 % realmin = 2.2251e-308
6 larger_than_realmin = realmin + 0.0001e-308;
7 A = [larger_than_realmin larger_than_realmin; larger_than_realmin larger_than_realmin];
8 x = frobenius(A);
9 y = norm(A, 'fro');
10 function x = frobenius(A)
11     x = sqrt(sum(sum(A.^2)));
12 end
13

```

Handwritten annotations in red and green:

- `2x2 matrix` (red) pointing to the definition of matrix `A`.
- `The build-in for F-norm` (green) pointing to `y = norm(A, 'fro');`.
- `function` (red) pointing to the `function x = frobenius(A)` block.
- `incorrect result` (red) pointing to `x = 0` in the Command Window.
- `correct result` (green) pointing to `y = 4.4503e-308` in the Command Window.

The Command Window shows the results of the execution:

```

x =
    0
y =
    4.4503e-308

```

frobenius(A) gives 0 because when the function squares larger-than-realmin, all squared number underflow, but correct result shouldn't underflow because there is a square root at the end of computation.

(c)


```

frobenius1.m * x untyped2.m x +
4      C = [1 2 3; 4 5 smaller_than_realmax; 2 4 5];
5      % realmin = 2.2251e-308
6      larger_than_realmin = realmin + 0.0001e-308;
7      A = [larger_than_realmin larger_than_realmin; larger_than_realmin larger_than_realmin]
8      x = frobenius(C)
9      function x = frobenius(A)
10         x = sqrt(sum(sum(A.^2)));
11
12         % If x equal to inf, then this means there is an overflow that
13         % may cause by squaring entries of A, so in order to avoid the overflow caused
14         % by squaring, divide every entry of A by 10^200 before squaring (so the big
15         % number that is less than realmax is no more overflow after squaring),
16         % multiply 10^200 back after taking square root to cancel the division out.
17         if x == inf
18             x = (10^200)*sqrt(sum(sum((A./(10^200)).^2)));
19
20         % If x equal to 0, then this means there is an underflow that
21         % may cause by squaring entries of A, so in order to avoid the underflow caused
22         % by squaring, multiply every entry of A by 10^200 before squaring (so the small
23

```

Overflow case

explanation

code added for
dealing overflow

Command Window

>> frobenius1

```

x =
    1.7975e+308

```

result

NAVIGATE

CODE

ANALYZE

SECTION

RUN

yulunwu > Downloads > 2021 Fall

```

frobenius1.m x untyped2.m x +
10      x = sqrt(sum(sum(A.^2)));
11
12      % If x equal to inf, then this means there is an overflow that
13      % may cause by squaring entries of A, so in order to avoid the overflow caused
14      % by squaring, divide every entry of A by 10^200 before squaring (so the big
15      % number that is less than realmax is no more overflow after squaring),
16      % multiply 10^200 back after taking square root to cancel the division out.
17      if x == inf
18          x = (10^200)*sqrt(sum(sum((A./(10^200)).^2)));
19
20      % If x equal to 0, then this means there is an underflow that
21      % may cause by squaring entries of A, so in order to avoid the underflow caused
22      % by squaring, multiply every entry of A by 10^200 before squaring (so the small
23      % number that is bigger than realmin is no more underflow after squaring),
24      % divide 10^200 back after taking square root to cancel the multiplication out.
25      elseif x == 0
26          x = (sqrt(sum(sum((A.*(10^200)).^2)))/(10^200));
27      end
28      end
29

```

Underflow case

explanation

code added for
dealing underflow

Command Window

>> frobenius1

```

x =
    4.4503e-308

```

result

Compare to the old version of Frobenius(A), the new version add if statements to re-calculate the F-norm if the original calculation result in inf or 0

In the worst case of this function, A is originally calculated to be inf or 0 in line 10, then it goes to line 18 ($x == \text{inf}$) or line 26 ($x == 0$). Line 18 and line 26 are repeat the calculation in line 10 with extra multiplication and division.

The operation cost for the old function and the best case for the new function is:

$$\underbrace{(m \times n)}_{\substack{\text{squaring} \\ \wedge 2}} + \underbrace{n(m-1)}_{\substack{\text{inner sum} \\ \text{sum}()}} + \underbrace{(n-1)}_{\substack{\text{outer sum} \\ \text{sum}()}} + \underbrace{1}_{\substack{\text{square root} \\ \text{sqrt}()}} = 2mn$$

The operation cost for the new version of function at the worst case is:

$$2(2mn) + \underbrace{mn}_{\substack{\text{operation} \\ \text{before squaring}}} + \underbrace{1}_{\substack{\text{operation after square root}}} = 5mn + 1$$

Therefore, the extra cost needed by the new version of function is $5mn + 1 - 2mn = 3mn + 1$ more than the old version of function in the worst case.