CSC336 A1

Kaiqu Liang

October 12, 2018

Problem 1

We know the formulas:

$$\text{absolute error} = \text{approximate value - true value}$$
$$\text{relative error} = \frac{\text{approximate value - true value}}{\text{true value}}$$

(a)
absolute error $= 3.14 - 3.14159265358979 \approx -1.59 \times 10^{-3}$
relative error $= \frac{3.14-3.14159265358979}{3.14159265358979} \approx -5.07 \times 10^{-4}$
(b)
absolute error $= 3.14159 - 3.14159265358979 \approx -2.65 \times 10^{-6}$
relative error $= \frac{3.14159-3.14159265358979}{3.14159265358979} \approx -8.45 \times 10^{-7}$
(c)
absolute error $= 3.141592654 - 3.14159265358979 \approx 4.10 \times 10^{-10}$
relative error $= \frac{3.141592654-3.14159265358979}{3.14159265358979} \approx 1.31 \times 10^{-10}$

Problem 2

(a) $5.35 \times 10^0 + 2.46 \times 10^{-2} = 5.3746 \times 10^0 \approx 5.37 \times 10^0$
(b) $4.53 \times 10^1 - 6.38 \times 10^{-1} = 4.4662 \times 10^1 \approx 4.47 \times 10^1$
(c) $5.65 \times 10^1 + 5.23 \times 10^{-4} = 5.6500523 \times 10^1 \approx 5.65 \times 10^1$
(d) $6.54 \times 10^4 - 8.73 \times 10^6 = -8.6646 \times 10^6 \approx -8.66 \times 10^6$
(e) $5.21 \cdot 10^8 \times 4.25 \cdot 10^{-5} = 2.21425 \times 10^4 \approx 2.21 \times 10^4$
(f) $-4.32 \cdot 10^7 \times 3.25 \cdot 10^3 = -1.404 \times 10^{11} \approx -14.0 \times 10^{10} = -inf$ (overflow)
(g) $5.41 \cdot 10^{-5} \times 4.27 \cdot 10^{-5} = 2.31007 \times 10^{-9} \approx 2.31 \times 10^{-9}$
(h) $-6.52 \cdot 10^{-6} \times 4.75 \cdot 10^{-6} = -0.3097 \times 10^{-10} \approx -0.31 \times 10^{-10}$ (underflow)
(i) $-6.46 \cdot 10^{-7} \times 1.32 \cdot 10^{-6} = -0.0085272 \times 10^{-10} \approx -0.01 \times 10^{-10}$ (underflow)
(j) $3.82 \cdot 10^{-6} \times 1.25 \cdot 10^{-7} = 0.004775 \times 10^{-10} \approx 0$ (underflow)

Problem 3

(a)
I want to calculate the condition number.

$$\begin{aligned}
\text{Cond} &= \left| \frac{\frac{\Delta y}{y}}{\frac{\Delta x}{x}} \right| \\
&\approx \frac{x f'(x)}{f(x)} \\
&= \frac{x \cdot \sec^2 x}{\tan x} \\
&= \frac{x}{\sin x \cdot \cos x} \\
&= \frac{2x}{\sin 2x} \qquad (-\frac{\pi}{2} < x < \frac{\pi}{2})
\end{aligned}$$

When x is close to 0, $\lim_{x \to 0} \frac{2x}{\sin 2x} = 1$. We notice that the condition number is small which indicates that this function is well-conditioned when x is close to 0.
When x is close to $\frac{\pi}{2}$, $\lim_{x \to (\frac{\pi}{2})^-} \frac{2x}{\sin 2x} = \infty$. We notice that the condition number is very large which indicates that this function is ill-conditioned when x is close to $\frac{\pi}{2}$.
(b)
MatLab program:

```
function verify
    fprintf("%s\t\t\t\t\t%s\n", "x", "Output");
    gap = 1.0e-6;
    for a = 1.0e-5: gap: 1.4e-5
        fprintf("%d\t\t%d \n", a, tan(a));
    end
    fprintf("\n\n");

    value = pi/2;
    for b = value - 5 * gap: gap: value - gap
        fprintf("%d\t\t%d \n", b, tan(b));
    end
```

Output from the previous:

```
>> verify
x                       Output
1.000000e-05            1.000000e-05
1.100000e-05            1.100000e-05
1.200000e-05            1.200000e-05
1.300000e-05            1.300000e-05
1.400000e-05            1.400000e-05


1.570791e+00            2.000000e+05
1.570792e+00            2.500000e+05
1.570793e+00            3.333333e+05
1.570794e+00            5.000000e+05
1.570795e+00            1.000000e+06
```

In my program, I first plug in 5 different x which are very close to 0. We notice that a small relative change to x produces a small relative change to the output. Therefore, the function is well-conditioned when x is very close to 0. Next, I plug in 5 different x which are very closed to $\frac{\pi}{2}$. We notice that a small relative change to x produces a large relative change to the output. Therefore, the function is ill-conditioned when x is very close to $\frac{\pi}{2}$.

Problem 4

(a)
Explanation: When x is very close to 0, the true value of cos(x) will be very close to 1. Since the precision in matlab is not large enough, the result will round to 1 if $cos(x)$ is very close to 1. Then 1 - cos(x) will be 0 in this case and the final result will be 0.
Show how to rewrite the statement:

$$\begin{aligned} F &= \frac{1-\cos(x)}{x^2} \\ &= \frac{1-(1-2\sin^2\frac{x}{2})}{x^2} \\ &= \frac{2\sin^2\frac{x}{2}}{x^2} \end{aligned}$$

MatLab Program:

```
function F = exam2(x)
    F = 2 * (sin(x/2))^2/x^2;
end
```

Check:
When $x = 1.0e-8$, $F = 0.5000$ which is accurate.
When $x = 1$, $F = 0.4597$ which is accurate.
After rewriting, we don't need to worry too much about the rounding error caused by precision. The result will be much more accurate.

(b)
We know $G = \sqrt{x^2 + y^2}$.
Case1: $x = y = 0, G = 0$.
Case2: $|x| \geq |y| \geq 0$ and $|x| \neq 0$, $G = |x|\sqrt{1 + (\frac{y}{x})^2}$
Case3: $|y| \geq |x| \geq 0$ and $|y| \neq 0$, $G = |y|\sqrt{1 + (\frac{x}{y})^2}$
Rewirte the MatLab statement:

```
function G = exam3(x, y)
    a = abs(x);
    b = abs(y);
    if a == 0 && b == 0
        G = 0;
    elseif a > b
        G = a * sqrt(1 + (b/a)^2);
    else
        G = b * sqrt(1 + (a/b)^2);
    end
end
```

Explain: The main problem in this statement is that if $|x|$ or $|y|$ is a large number close to upper bound of the floating point system, then the intermediate value x^2 or y^2 will exceed the upper bound and cause this value be inf. However, if we rewrite the statement by this way, there will be no too large intermediate value which cause the overflow (except x or y itself overflow) and therefore solve this problem.

Problem 5

(a)
MatLab function exp1:

```
function output = exp1(x)
    index = 1;
    initial = 1;
    next = 1 + x;
    while next ~= initial
        index = index + 1;
        initial = next;
        next = next + x^(index)/factorial(index);
    end
    output = next;
end
```

Test program:

```
function output5a
    fprintf("%s\t\t%s\n", "x", "Relative Error");
    for a = -25: 25
        relative = (exp1(a) - exp(a))/exp(a);
        fprintf("%d\t\t%d \n", a, relative);
    end
end
```

Output:

```
>> output5a
x               Relative Error
-25             5.822619e+04
-24             9.966351e+03
-23             6.622900e+01
-22             -1.150737e+02
-21             3.538652e+01
-20             1.024904e+00
-19             -5.441983e-01
-18             4.947964e-02
-17             1.019626e-03
-16             2.852595e-04
-15             1.035376e-05
-14             -8.611208e-06
-13             -1.299651e-06
-12             6.121757e-08
-11             7.648289e-08
-10             -7.234246e-09
-9              -5.491794e-10
-8              -1.477276e-10
-7              1.260620e-11
-6              -7.250315e-13
-5              2.136883e-13
```

−4	1.439633e−14
−3	8.362285e−16
−2	4.101750e−16
−1	3.017899e−16
0	0
1	0
2	−2.404038e−16
3	−3.537584e−16
4	5.205618e−16
5	−1.915040e−16
6	0
7	−6.220139e−16
8	−1.525507e−16
9	0
10	−3.303280e−16
11	0
12	−3.576402e−16
13	−1.315685e−16
14	1.936054e−16
15	0
16	0
17	3.084494e−16
18	0
19	−1.669763e−16
20	−2.457087e−16
21	−3.615647e−16
22	2.660244e−16
23	1.957298e−16
24	0
25	0

(b)
When x is non-negative or a negative number with small magnitude, the function produce accurate approximations to e^x.
When x is a negative number with large magnitude, the function produce poor approximations to e^x.
Explanation:

> Rule for sum: if we compute a sum and intermediate values in the sum are much larger than the final result, then the final result may be very inaccurate.

In this case, when x is a negative number with large magnitude, e^x is very small and close to 0. However, there are some very large intermediate values which are much larger than the final result. This cause the final result inaccurate. More specifically, this situation happens due to catastrophic cancellation. For example, When a very large positive intermediate value subtracts absolute value of next term which is very close to it, some significant digits will be lost and the result will be inaccurate.
Then I will discuss two cases that make approximation accurate. When x is a negative number with small magnitude, all the intermediate values are not so large and close to e^x. Therefore the final result is accurate. When x is non-negative, all the intermediate values are non-negative. It follows that these values are all less than or equal to e^x. Therefore the final result is accurate by summing these smaller values. More specifically, since all the terms are non-negative, catastrophic cancellation does not happen in this case.

(c)
MatLab function exp2:

```
function output = exp2(x)
    if x >= 0
        output = exp1(x);
    else
        output = 1/exp1(-x);
    end
end
```

Test program:

```
function output5c
    fprintf("%s\t\t%s\n", "x", "Relative Error");
    for a = -25: 25
        relative = (exp2(a) - exp(a))/exp(a);
        fprintf("%d\t\t%d \n", a, relative);
    end
end
```

Output:

```
>> output5c
x               Relative Error
-25             -1.163302e-16
-24             0
-23             -2.518973e-16
-22             -1.853357e-16
-21             4.090871e-16
-20             2.006596e-16
-19             0
-18             0
-17             -3.196881e-16
-16             0
-15             0
-14             -2.546614e-16
-13             1.873694e-16
-12             4.135760e-16
-11             0
-10             2.985143e-16
-9              -2.196345e-16
-8              1.615981e-16
-7              7.133832e-16
-6              0
-5              2.574558e-16
-4              -3.788508e-16
-3              2.787428e-16
-2              2.050875e-16
-1              -1.508950e-16
```

0	0
1	0
2	$-2.404038e{-16}$
3	$-3.537584e{-16}$
4	$5.205618e{-16}$
5	$-1.915040e{-16}$
6	0
7	$-6.220139e{-16}$
8	$-1.525507e{-16}$
9	0
10	$-3.303280e{-16}$
11	0
12	$-3.576402e{-16}$
13	$-1.315685e{-16}$
14	$1.936054e{-16}$
15	0
16	0
17	$3.084494e{-16}$
18	0
19	$-1.669763e{-16}$
20	$-2.457087e{-16}$
21	$-3.615647e{-16}$
22	$2.660244e{-16}$
23	$1.957298e{-16}$
24	0
25	0