L. Vandenberghe
ECE133A

6/4/2021

Practice problem solutions

1. *Exercise A15.1 (c).* $\|A\|_2 = \|A\|_F = \|u\|\|v\|$. The Frobenius norm is

$$\|A\|_F = \left(\sum_{i=1}^n \sum_{j=1}^n A_{ij}^2\right)^{1/2} = \left(\sum_{i=1}^n \sum_{j=1}^n u_i^2 v_j^2\right)^{1/2} = \left(\sum_{i=1}^n u_i^2\right)^{1/2} \left(\sum_{j=1}^n v_j^2\right)^{1/2} = \|u\|\|v\|.$$

   To find the 2-norm, we first note that

$$\|uv^T x\| = \|(v^T x)u\| = |v^T x|\,\|u\|.$$

   Therefore

$$\|A\|_2 = \max_{x\neq 0} \frac{\|uv^T x\|}{\|x\|} = \max_{x\neq 0} \frac{|v^T x|\|u\|}{\|x\|} = \|u\| \max_{x\neq 0} \frac{|v^T x|}{\|x\|}.$$

   By the Cauchy–Schwarz inequality, we have $|v^T x| \leq \|v\|\|x\|$, with equality if the vectors are aligned or anti-aligned. Therefore $\max_{x\neq 0} |v^T x|/\|x\| = \|v\|$ and $\|A\|_2 = \|u\|\|v\|$.

2. *Exercise A15.11.*

   (a) To find lower bounds for $\|A\|_2$ and $\|A^{-1}\|_2$, we use the inequalities

$$\|A\|_2 \geq \frac{\|Ax\|}{\|x\|}, \qquad \|A^{-1}\|_2 \geq \frac{\|A^{-1}y\|}{\|y\|},$$

   which hold for all nonzero x and y. Choosing $x = (0, 1)$ and $y = (1, 0)$, for example, gives

$$\|A\|_2 \geq \sqrt{2}, \qquad \|A^{-1}\|_2 \geq 10^8\sqrt{2}.$$

   The product of the two lower bounds is a lower bound on $\kappa(A)$:

$$\kappa(A) = \|A\|_2\|A^{-1}\|_2 \geq 2 \cdot 10^8.$$

   Of course, other choices of x and y will give different lower bounds on $\|A\|_2$, $\|A^{-1}\|_2$, and κ.

   (b) The solution of $Ax = b$ is

$$x = A^{-1}b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

   Most choices of Δb will give a $\Delta x = A^{-1}\Delta b$ that is much greater than Δx. For example, choosing $\Delta b = (1, 0)$ gives

$$\Delta x = A^{-1}\Delta b = 10^8 \begin{bmatrix} 1 \\ -1 \end{bmatrix},$$

so for this choice of Δb we get

$$\frac{\|\Delta x\|}{\|x\|} = 10^8\sqrt{2}, \qquad \frac{\|\Delta b\|}{\|b\|} = \frac{1}{\sqrt{2}}.$$

3. *Exercise A15.23.*

   (a) $\|A\|_2 = \|QR\|_2 = \|R\|_2$ because Q has orthonormal columns and therefore $\|QRx\| = \|Rx\|$ for all x. The norm of R can be bounded as

   $$\|R\|_2 \geq \|Re_i\| = \sqrt{R_{1i}^2 + \cdots + R_{ii}^2} \geq R_{ii}$$

   for $i = 1, \ldots, n$.

   (b) $A^\dagger = R^{-1}Q^T$.

   $$\|A^\dagger\|_2 = \|(A^\dagger)^T\|_2 = \|QR^{-T}\|_2 = \|R^{-T}\|_2 \geq \max\{\frac{1}{R_{11}}, \ldots, \frac{1}{R_{nn}}\}$$

   because R^{-T} is lower triangular with diagonal elements $1/R_{ii}$.

   (c) $AA^\dagger = QQ^T$.

   $$\|AA^\dagger\|_2 = \|QQ^T\|_2 = \|Q^T\|_2 = \|Q\|_2 = 1.$$

4. *Exercise A16.1.* We can rewrite the formula as

   $$\frac{1-\cos x}{\sin x} = \frac{(1-\cos x)(1+\cos x)}{\sin x\ (1+\cos x)} = \frac{\sin x}{1+\cos x}.$$

   Evaluating this expression yields

```
>> format long e
>> chop(sin(1e-2), 4) / (1+chop(cos(1e-2), 4))
ans =

       5.000000000000000e-003
```

   which is much more accurate, if we compare with the result in the full MATLAB precision

```
>> format long e
>> sin(1e-2) / (1+cos(1e-2))
ans =

       5.000041667083338-003
```

5. *Exercise A16.2.* Use the second expression in (47) instead, *i.e.*, first determine $\mathbf{avg}(x)$ as in the MATLAB code, and then calculate $\mathbf{std}(x)^2$ from (47):

```
n = length(x);
sum = 0;
for i=1:n
    sum = chop(sum + x(i), 6);
end;
xmean = chop(sum/n, 6)
sum = 0;
for i=1:n
    dx = chop(x(i) - xmean, 6);
    sum = chop(sum + dx^2, 6);
end;
xstd = chop(sum/n, 6);
```

This returns

```
xmean =

  1001.8

xstd =

  1.1600
```

In this example, the MATLAB code actually calculates $\mathbf{avg}(x)$ exactly, because $\sum_i x_i$ has only five significant digits, so rounding to six digits does not introduce any error. Therefore there is no cancellation when we calculate the differences $x_i - \bar{x}$ in equation (47), and the only error in $\mathbf{std}(x)^2$ is due to rounding the result to six digits.

6. *Exercise A16.3.* If you display the intermediate results in the first loop, you'll notice that the variable `sum` reaches the value 1.6240 at $i = 44$, and remains constant after that. The reason is simple: $1/45^2 = 4.938 \cdot 10^{-4}$, so

$$1.6240 + 4.938 \cdot 10^{-4} = 1.62449\ldots,$$

and rounding to four significant digits yields 1.6240.

The second implementation is much more accurate, because we add the smallest terms $1/i^2$ first, while the sum is still small, and the largest terms are added at the end of the iteration.

7. *Exercise A17.1 (a, b, c).* MATLAB returns the following numbers

(a) 0

(b) $1.1102 \cdot 10^{-16}$

(c) $-1.1102 \cdot 10^{-16}$

To explain the first three values, we have to determine the floating-point numbers closest to 1. The representation of 1 as a double precision floating-point number is

$$\begin{aligned} 1 &= (1 \cdot 2^{-1} + 0 \cdot 2^{-2} + \cdots + 0 \cdot 2^{-n})\, 2^1 \\ &= (.10\cdots 00)_2\, 2^1 \end{aligned}$$

where $n = 53$. The smallest floating-point number greater than 1 is

$$\begin{aligned} (.10\cdots 01)_2\, 2^1 &= (1 \cdot 2^{-1} + 0 \cdot 2^{-2} + \cdots + 0 \cdot 2^{-n-1} + 1 \cdot 2^{-n})\, 2^1 \\ &= 1 + 2^{-n+1} \\ &= 1 + 2\epsilon_M \\ &= 1 + 2.2204 \cdot 10^{-16}. \end{aligned}$$

The largest floating-point number less than 1 is

$$\begin{aligned} (.11\cdots 11)_2\, 2^0 &= (1 \cdot 2^{-1} + 1 \cdot 2^{-2} + \cdots + 1 \cdot 2^{-n-1} + 1 \cdot 2^{-n})\, 2^0 \\ &= 1 - 2^{-n} \\ &= 1 - \epsilon_M \\ &= 1 - 1.1102 \cdot 10^{-16}. \end{aligned}$$

This is summarized in the figure below.

The situation around the number -1 is symmetric: the smallest floating-point number greater than -1 is $-1+\epsilon_M$; the largest floating-point number less than -1 is $-1-2\epsilon_M$.

We can now explain the first three results.

(a) $1+10^{-16}$ lies between 1 and $1+\epsilon_M$, so it is rounded to 1, and subtracting 1 yields zero.

(b) $10^{-16}-1$ lies between $-1+\epsilon_M/2$ and $-1+\epsilon_M$, so it is rounded to $-1+\epsilon_M$, and adding 1 yields ϵ_M.

(c) $1-10^{-16}$ lies between $1-\epsilon_M$ and $1-\epsilon_M/2$, so it is rounded to $1-\epsilon_M$, and subtracting 1 yields $-\epsilon_M$.

8. *Exercise A17.4.* The final value is $x = 1$.

   Using the hint we can say that after one pass through the first for-loop we have $1 < x < 1+1/2$. After the second pass, $1 < x < 1+1/4$. After k passes, $1 < x < 1+1/2^k$, and after finishing the for-loop we have

$$1 < x < 1 + 2^{-54}.$$

   This means x lies between 1 and $1+\epsilon_M$. (Recall that $\epsilon_M = 2^{-53}$.) Therefore we can expect that in double-precision arithmetic, the value after the first for-loop will be $x = 1$, and squaring 54 times still yields $x = 1$.

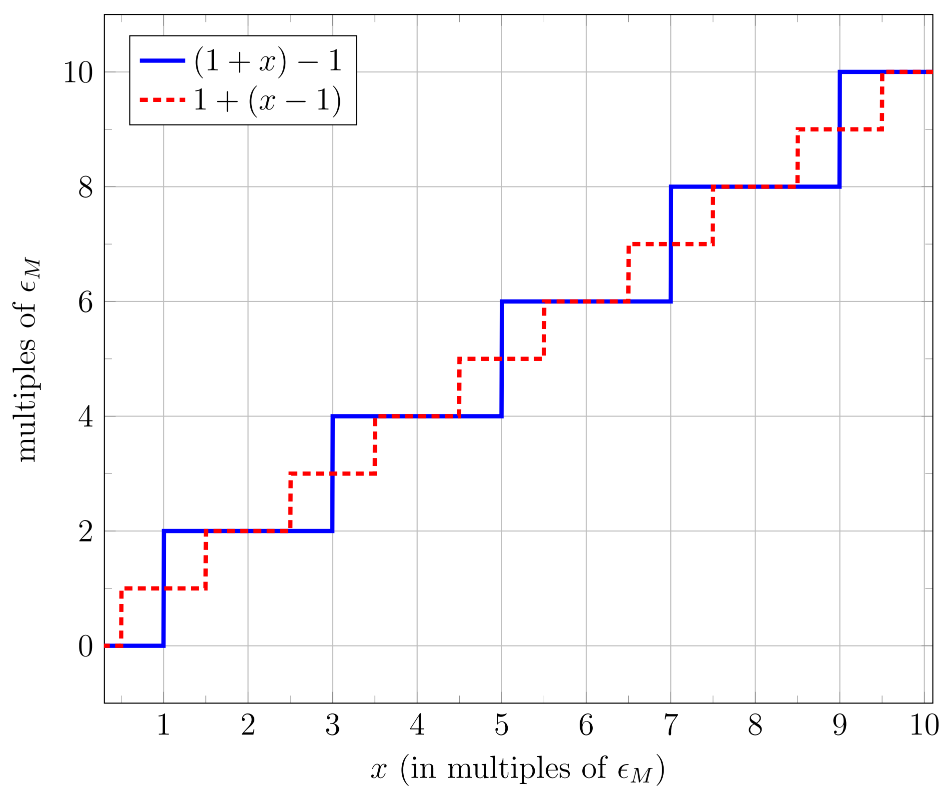
9. *Exercise A17.5.*

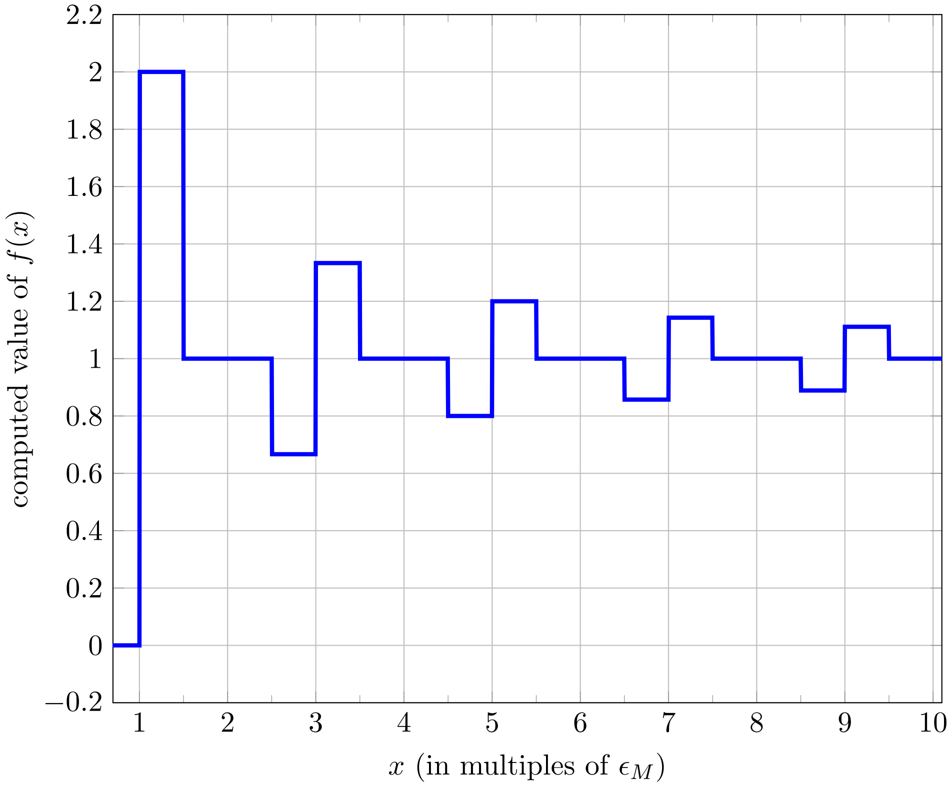
(a) MATLAB starts by evaluating $1 + 3 \cdot 10^{-16}$, which is rounded to $1 + 2\epsilon_M$. It then computes $\log(1 + 2\epsilon_M)$, which gives a result very close to $2\epsilon_M$. Dividing by $3 \cdot 10^{-16}$ gives

$$\frac{2\epsilon_M}{3 \cdot 10^{-16}} = 0.7401.$$

(b) In both numerator and denominator, the number $1 + 3 \cdot 10^{-16}$ will be rounded to $1 + 2\epsilon_M$. In the numerator we get $\log(1 + 2\epsilon_M) \approx 2\epsilon_M$. In the denominator we get $(1 + 2\epsilon_M) - 1 \approx 2\epsilon_M$. The result of the division is 1.

10. *Exercise A17.10.* The first figure shows the rounded values of the numerator and denominator. The second plot shows the result of the division.





2.2
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
−0.2
computed value of $f(x)$
1
2
3
4
5
6
7
8
9
10
x (in multiples of ϵ_M)