Conditioning and stability

Zecheng Zhang

April 7, 2023

In the abstract, we can view a problem as $f : X \to Y$ where X, Y are two spaces. A well-conditioned problem is one with the property that all small perturbation of x lead to only small changes in $f(x)$.

1 Relative condition number

Denote $\delta f = f(x + \delta x) - f(x)$. The relative conditioning number is defined as

$$\kappa(x) = \lim_{\delta \to 0} \sup_{\|\delta x\| \leq \delta} \left(\frac{\|\delta f\|}{\|f(x)\|} / \frac{\|\delta x\|}{\|x\|}\right). \tag{1}$$

One can assume δx and δf are infinitesimal, then

$$\kappa(x) = \sup_{\|\delta x\|} \left(\frac{\|\delta f\|}{\|f(x)\|} / \frac{\|\delta x\|}{\|x\|}\right). \tag{2}$$

When f is differentiable, we can express the quantity in terms of the Jacobian of f,

$$\kappa = \frac{\|J(x)\|}{\|f(x)\| / \|x\|}. \tag{3}$$

A problem is well-conditioned if κ is small (e.g., 1, 10, 100), and a problem is ill-conditioned if κ is large (e.g., 10^6 or bigger).

Example 1.1. Consider $x \to x/2$.

Example 1.2. Consider $x \to \sqrt{x}$, $x > 0$.

Example 1.3. Consider $f(x) = x_1 - x_2$.

2 Conditioning of matrix multiplication

Let $A \in \mathbb{R}^{m \times n}$, we consider the problem of computing Ax given a x. We want to know how Ax will change if there is a perturbation in x. The conditioning number of A is defined as,

$$\kappa = \sup_{\delta x} \left(\frac{\|A(x + \delta x) - Ax\|}{\|Ax\|} \Big/ \frac{\|\delta x\|}{\|x\|}\right) = \sup_{\delta x} \frac{\|A\delta x\|}{\|\delta x\|} \Big/ \frac{\|Ax\|}{\|x\|}. \tag{4}$$

Note that sup is over all δx and $\frac{\|Ax\|}{\|x\|}$ is independent with respect to sup, it follows that,

$$\kappa = \frac{\|x\|}{\|Ax\|} \sup_{\delta x} \frac{\|A\delta x\|}{\|\delta x\|} = \|A\| \frac{\|x\|}{\|Ax\|}, \tag{5}$$

where $\|A\|$ is the operator norm (it is L_2 norm if $\|\cdot\|$ is the L_2 vector norm). Note that, the condition number depends both on A and x.

Remark 1. Suppose A is nonsingular square matrix. We have $\|x\| = \|A^{-1}Ax\| \leq \|A^{-1}\|\|Ax\|$, this further implies that,

$$\kappa \leq \|A\|\|A^{-1}\|, \tag{6}$$

or

$$\kappa = c\|A\|\|A^{-1}\|, \tag{7}$$

for some positive constant $c = \frac{\|x\|}{\|Ax\|}/\|A^{-1}\|$.

Theorem 2.1. Let $A \in \mathbb{R}^{m\times n}$ be invertiable and let us consider $Ax = b$. The problem of computing b given x has conditioning number,

$$\kappa = \|A\|\frac{\|x\|}{\|b\|} \leq \|A\|\|A^{-1}\|, \tag{8}$$

with the perturbation in x. The problem of computing x given b has the conditioning number,

$$\kappa = \|A^{-1}\|\frac{\|b\|}{\|x\|} \leq \|A^{-1}\|\|A\|, \tag{9}$$

with the perturbation in b. If we use the L_2 norm, the first equality holds if x is a multiple of a right singular vector of A corresponding to the minimal singular value. The second equality holds if b is a multiple of a left singular vector of A corresponding to the largest singular value.

Definition 2.2. We will call $\kappa(A) = \|A\|\|A^{-1}\|$ the condition of A relative to norm $\|\cdot\|$ and denote it as $\kappa(A) = \|A\|\|A^{-1}\|$. The conditioning number is attached to matrix A not to the problem and x. If $\kappa(A)$ is small, A is called well-conditioned, otherwise, it is called ill-conditioned. If A is singular, we write $\kappa(A) = \infty$.

Remark 2. If $\|\cdot\| = \|\cdot\|_2$, $\|A\| = \sigma_1$ and $\|A^{-1}\| = 1/\sigma_m$, it follows that $\kappa(A) = \frac{\sigma_1}{\sigma_m}$

Remark 3. When $A \in \mathbb{C}^{m\times n}$ of full rank and $m \geq n$. The conditioning number is defined in terms of the pseudo-inverse, i.e.,

$$\kappa(A) = \|A\|\|A^{+}\|, \tag{10}$$

where $A^{+} = (A^{*}A)^{-1}A^{*}$ is called the pseudo-inverse of A.

3 Conditioning of a system of eqautions

We considered the case when A is fixed and perturbed x or b. What if we perturb A? Specifically, b is fixed and let us consider solving x from $Ax = b$ given a small change in A, We have,

$$(A + \delta A)(x + \delta x) = b \tag{11}$$

$$Ax + A\delta x + \delta A x + \delta A \delta x = b. \tag{12}$$

Using $Ax = b$ and dropping the high order infinitesimal $\delta A \delta x$, it follows that $A\delta x + \delta A x = 0$, or $\delta x = -A^{-1}\delta A x$. Taking a norm, $\|\delta x\| \leq \|A^{-1}\|\|\delta A\|\|x\|$, or,

$$\frac{\|\delta x\|}{\|x\|}/\frac{\|\delta A\|}{\|A\|} \leq \|A^{-1}\|\|A\| = \kappa(A). \tag{13}$$

Equality holds when $\|\delta x\| = \|A^{-1}\|\|\delta A\|\|x\|$. It can be shown that for any A and b such δA exists. This leads us to the following result.

Theorem 3.1. Let b be fixed and consider the problem $x = A^{-1}b$, where A is nonsingular. The conditioning number associated with this problem with respect to perturbation in A is:

$$\kappa = \|A\|\|A^{-1}\| = \kappa(A). \tag{14}$$

4 Floating point

Computers use a finite number of bits to represent real numbers, they can only represent only a finite subset of real numbers. This has two limitations. Firstly, the represented number cannot be arbitrarily large or small. Secondly, there must be gaps between them.

In IEEE double precision arithmetic (one way to store numbers/digital representation of number in the computer), the interval $[1, 2]$ is represented by the discrete subset:

$$1, 1 + 1 \times 2^{-52}, 1 + 2 \times 2^{-52}, ..., 2 + 2^{52} \times 2^{-52}. \tag{15}$$

In general, the interval $[2^j, 2^{j+1}]$ is represented by 15 times 2^j. The gap between the two adjacent numbers is never larger than $2^{-52} \approx 2.22 \times 10^{-16}$ in relative sense.

IEEE double precision is an example of an arithmetic system based on a floating-point $\mathbf{F}$ representation of real numbers. Here $\mathbf{F}$ is a discrete subset of real numbers (example, Equation 15) which is used to digitally represent real numbers. Let us now define the machine epsilon ϵ_m. This number is half the distance between 1 and the next larger floating point number. It has the following property.

Property 4.0.1. For all $x \in \mathbb{R}$, ther exists $x' \in F$ such that $|x - x'| \leq \epsilon_m |x|$.

This is in a relative sense since if $x > 0$, $|1 - x'/x| \leq \epsilon_m$.

Let $fl : \mathbb{R} \to F$ be a function giving the closet floating-point approximation to a real number (rounded to one floating number). Then the above property can be stated in terms of ft: for all $x \in \mathbb{R}$, there exists ϵ with $\epsilon < \epsilon_m$, there exists ϵ with $|\epsilon| < \epsilon_m$ such that $fl(x) = x(1 + \epsilon)$.

The difference between a real number and its closest floating-point approximation is always smaller than the machine ϵ_m in a relative sense. Machine epsilon or machine precision is an upper bound on the relative approximation error due to rounding in floating point arithmetic.

5 Stability

A mathematical problem can be formulated as $f : X \to Y$ where X and Y are some spaces. An algorithm can be viewed as another map $g : X \to Y$.

Definition 5.1 (Accuracy)**.** We say an algorithm is accurate if

$$\frac{\|g(x) - f(x)\|}{\|f(x)\|} = O(\epsilon_m), \tag{16}$$

for all $x \in X$.

Loosely speaking, the symbol $\mathcal{O}(\epsilon)$ means "on the order of machine epsilon". This expression applies uniformly to all x.

Remark 4. We discuss the order $\mathcal{O}$ here. Let us consider $h(t) = \mathcal{O}(g(t))$. The standard mathematical definition is: there exists a positive constant C such that for all t sufficient close to an understandable limit (for example, 0 or ∞), we have $\|h(t)\| \leq Cg(t)$.

Definition 5.2 (Backward Stability)**.** We say an algorithm g is backward stable if for all $x \in X$,

$$g(x) = f(y), \text{ for some } y \text{ with } \frac{\|x - y\|}{\|x\|} = O(\epsilon_m). \tag{17}$$

Intuitively, a backward stable algorithm gives exactly the right answer to nearly the right question. $f(y)$ is "the exact solution (calculated by f) of a slightly wrong input (y which is closed to x)" and is exact to the algorithm with the exact input.

To repeat, conditioning is intrinsic to the problem. Stability is a property of an algorithm. Thus we will never say, "this problem is backward stable" or "this algorithm is ill-conditioned". We can say, "this problem is ill/well-conditioned", or "this algorithm is/isn't (backward) stable".

Theorem 5.3. Suppose a backward stable algorithm g is applied to solve a problem f with conditioning number κ. Then the relative error satisfies:

$$\frac{\|g(x) - f(x)\|}{\|f(x)\|} = O(\kappa(x)\epsilon_m). \tag{18}$$

Proof. By the definition of backward stability, we have $g(x) = f(y)$ for $y \in X$ satisfying

$$\frac{\|x - y\|}{\|x\|} = \mathcal{O}(\epsilon_m). \tag{19}$$

By the definition of conditioning number,

$$\frac{\|f(x) - g(y)\|}{\|f(x)\|} \Big/ \frac{\|x - y\|}{\|x\|} \leq \kappa(x). \tag{20}$$

It follows that,

$$\frac{\|f(x) - g(y)\|}{\|f(x)\|} \leq \kappa(x)\mathcal{O}(\epsilon_m) \approx \mathcal{O}(\kappa(x)\epsilon_m). \tag{21}$$

□

Here is how to interpret the result: If the problem is well-conditioned $O(\kappa) = 1$, this immediately implies good accuracy of the solution! However, otherwise, the solution might have poor accuracy. It is still the exact solution to a nearby problem (due to the backward stability). This is often as good as one can possibly hope for.

Example 5.4. Suppose we evaluate $f(x) = sin(x)$ for $x = \pi/2 - \delta$, δ is small. Suppose we are lucky enough to get as a computed result the exact correct answer, rounded to the floating point system: $g(x) = fl(sin(x))$ (i.e., g is the algorithm).

We want to find y close enough to x such that $g(x) = f(y)$. However, $g(x) = f(y) = f(x) + \delta(y - x) + error$, or $y - x \approx (g(x) - f(x))/\delta$. We have $g(x) - f(x) = fl(sin(x)) - sin(x) = \mathcal{O}(\epsilon_m)$, this implies that $y - x = \mathcal{O}(\epsilon_m/\delta)$. Since δ can be arbitrarily small, the $y - x$ is not of magnitude machine epsilon.

Example 5.5. Matlab implementation of QR.

1. Generate Q and R which satisfy the requirement of QR. Compute A.
2. Compute QR of A by computer, i.e., we have $A = Q_2R_2$. The algorithm is g, this step is indeed $g(A) = Q_2, R_2$. Check the error in Q_2 and R_2; you will find they are large. This means that the forward error of the algorithm g is large.
3. Compute $A_2 = Q_2R_2$ by computer. We can see that A is close to A_2. However if we compute $f(A_2) = Q_2, R_2$, where f is the real algorithm (do it by hand using the theory).

4. We have $g(A) = f(A_2)$, but A is closed to A_2. This indicates that the algorithm g is stable.

It can be observed that Q_2 and R_2 have large errors compared to real Q and R of A. That is, the forward errors of the algorithm g is large. In general, a large forward error can be the result of an ill-conditioned problem (check the last theorem) or an unstable algorithm. This problem is due to the conditioning of the problem.

Theorem 5.6. Let the $A = QR$ be the QR of $A \in \mathbb{R}^{m\times n}$ computed by Householder triangularization, and let Q_1 and R_1 be the one with floating point errors. Then we have,

$$Q_1R_1 = A + \delta A, \quad \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\epsilon), \tag{22}$$

for some $\delta A \in \mathbb{R}^{m\times n}$.

6 Backward error

Backward error is a measure of error associated with an approximate solution to a problem. Whereas the forward error is the distance between the approximate and true solutions, the backward error is how much the data must be perturbed to produce the approximate solution.

For an algorithm, g from $\mathbb{R}^n$ to $\mathbb{R}^n$ and exact function f, the backward error is the smallest Δx such that $g(x) = f(x + \Delta x)$, for some appropriate measure of size. There can be many Δx satisfying this equation, so the backward error is the solution to a minimization problem. Using a vector norm and measuring perturbations in a relative sense, we can define the backward error as

$$\eta = \min\{\epsilon : g(x) = f(x + \Delta x), \|\Delta x\| \le \epsilon\|x\|\}. \tag{23}$$

Theorem 6.1. For problems, f and algorithm g defined on finite-dimensional spaces X and Y, the properties of accuracy stability and backward stability all hold or fail to hold independently of the choice of norms in X and Y.

7 (In)Stability of matrices multiplication

Some basic facts:

1. Vector-vector multiplication is backward stable. For example, $fl(y^*x) = (y+\delta y)^*(x+\delta x)$.
2. It is NOT true to say matrix-matrix multiplication is backward stable, which would require $fl(AB)$ to be equal to $(A + \delta A)(B + \delta B)$.

Theorem 7.1. Fix $Q \in \mathbb{C}^{m\times m}$ unitary; the matrix multiplication algorithm is backward stable for the problem

$$f(A) = QA, \quad A \in \mathbb{C}^{m\times n}. \tag{24}$$

Proof. Each entry of the product QA is an inner product $g(y) = x^*y$. The obvious algorithm for inner products is backward stable so that $\hat{g}(y) = g(\hat{y})$ where $\hat{y} = y + \delta y$ with $\|\delta y\| \le c(m)\epsilon_m\|y\|$ with some constant $c(m)$ independent of y and ϵ_m.

Consider the i, j entry of the product QA. To apply the above idea, let $x = q_i^*$ be the $i-$th row of Q and denote the $j-$th column of A by a_j. It follows by Cauchy-Schwarz inequality that,

$$|\hat{f}(A)_{ij} - f(A)_{ij}| = |\hat{g}(a_j) - g(a_j)| = |q_i^*(a_j + \delta a_j) - q_i^* a_j| \tag{25}$$
$$= |q_i^* \delta a_j| \leq \|q_i^*\| \|\delta a_j\| = \|\delta a_j\| \leq c(m)\epsilon_m \|a_j\|. \tag{26}$$

It follows that (note the summation indices change from i, j to j, this gives an m factor),

$$\|\hat{f}(A) - f(A)\|_F^2 = \sum_{i,j=1}^{m,n} |\hat{f}(A)_{ij} - f(A)_{ij}|^2 \leq c(m)^2 \epsilon_u^2 \|a_j\|_2^2 \tag{27}$$
$$= mc(m)^2 \epsilon_u^2 \sum_{j=1} \|a_j\|^2 = mc(m)^2 \epsilon_u^2 \|A\|_F^2. \tag{28}$$

Now we change tacks and describe the forward error as a backward error. Let

$$\delta A = Q^*(\hat{f}(A) - f(A)) \tag{29}$$

Note that,

$$\hat{f}(A) = \hat{f}(A) - f(A) + f(A) = Q\delta A + QA = A(A + \delta A) = f(\hat{A}), \tag{30}$$

where we denote $\hat{A} = A + \delta A$. It is remained to show $\|\hat{A} - A\|_F$ is relatively small, but we have

$$\frac{\|\hat{A} - A\|_F}{\|A\|_F} = \frac{\|\delta A\|_F}{\|A\|_F} = \frac{\|Q\delta A\|_F}{\|A\|_F} = \frac{\|\hat{f}(A) - f(A)\|_F}{\|A\|_F} \tag{31}$$
$$= \frac{\sqrt{m} c(m)\epsilon_u \|A\|_F}{\|A\|_F} = \sqrt{m} c(m)\epsilon_m, \tag{32}$$

where we use the property that the unitary matrix preserves the norm. □

8 Analyzing algorithm to solve $Ax = b$

We can solve $Ax = b$ by the QR factorization. This is a backward stable algorithm. The standard algorithm is as follows.

1. $QR = A$. This can be computed by the Householder algorithm.
2. $y = Q^*b$. This can be computed by one algorithm in the last QR section.
3. $x = R^{-1}y$. This can be computed by the back substitution, which is not covered.

The first step is the QR, the algorithm we discussed before outputs Q_1 and R_1, and we have established the stability by numerical experiments. Here we present the theorem.

Theorem 8.1. Let the QR factorization $A = QR$ of a matrix $A \in \mathbb{R}^{m\times n}$ be computed by the Householder triangularization, and let Q_1 and R_1 be the algorithm output. We then have,

$$Q_1 R_1 = A + \delta A, \quad \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\epsilon_m), \tag{33}$$

for some $\delta A \in \mathbb{R}^{m\times n}$

The second step is the computation of $Q_1^* b$. Please note that the first step outputs Q_1, which has forward errors. Hence we have Q_1^* instead of Q^*. Moreover, due to the rounding off errors $Q_1^* b = y_1$ is not the exact y. It can be shown that y_1 satisfies the backward stability estimate. Specifically,

$$(Q_1 + \delta Q) y_1 = b, \quad \|\delta Q\| = \mathcal{O}(\epsilon). \tag{34}$$

Let us verify this claim. Suppose the real math is f, and the algorithm is $g(Q_1) = y_1$. $(Q_1 + \delta Q) y_1 = b$ implies that $y_1 = f(Q_1 + \delta Q)$, where $\|\delta Q\|$ is of machine error. This implies that $f(Q_1 + \delta Q) = g(Q_1)$, where $\|\delta Q\|$ is small.

The final step is the back substitution. Note that we have R_1 and y_1 instead of R and y due to the errors. The computation of x_1 is backward stable. The estimation takes the form,

$$(R_1 + \delta R) x_1 = y_1, \quad \frac{\|\delta R\|}{\|R_1\|} = \mathcal{O}(\epsilon). \tag{35}$$

Theorem 8.2. The algorithm (QR to solve Ax = b) is backward stable, which satisfies,

$$(A + \Delta A) x_1 = b, \quad \frac{\|\Delta A\|}{\|A\|} = \mathcal{O}(\epsilon), \tag{36}$$

for some ΔA.

Proof. We have,

$$b = (Q_1 + \delta Q)(R_1 + \delta R) x_1 = [Q_1 R_1 + \delta Q R_1 + Q_1 \delta R + \delta Q \delta R] x_1. \tag{37}$$

Due to the last theorem,

$$b = (Q_1 + \delta Q)(R_1 + \delta R_1) x_1 = [A + \delta A + \delta Q R_1 + Q_1 \delta R + \delta Q \delta R] x_1 = (A + \Delta A) x_1, \tag{38}$$

where ΔA denotes the last four terms. We show the backward stability, we need to show that ΔA is small relative to A. Due to the last theorem, $Q_1 R_1 = A + \delta A$, where Q_1 is unitary. Multiply both sides by the Q_1 inverse and divide by $\|A\|$, it follows that,

$$\frac{\|R_1\|}{\|A\|} \leq \|Q_1^*\| \frac{\|A + \delta A\|}{\|A\|} = \mathcal{O}(1) + \mathcal{O}(\epsilon_m). \tag{39}$$

$\mathcal{O}(\epsilon_m)$ is higher order, we hence can drop it as ϵ_m goes to zero,

$$\frac{\|R_1\|}{\|A\|} \leq \mathcal{O}(1). \tag{40}$$

It follows that,

$$\frac{\|\delta Q R_1\|}{\|A\|} \leq \|\delta Q\| \frac{\|R_1\|}{\|A\|} = \mathcal{O}(\epsilon_m). \tag{41}$$

Similarly,

$$\frac{\|Q_1 \delta R\|}{\|A\|} \leq \|Q_1\| \frac{\|\delta R\|}{\|R_1\|} \frac{\|R_1\|}{\|A\|} = \mathcal{O}(\epsilon_m). \tag{42}$$

Finally,

$$\frac{\|\delta Q \delta R\|}{\|A\|} \leq \|\delta Q\| \frac{\|\delta R\|}{\|A\|} = \mathcal{O}(\epsilon_m^2). \tag{43}$$

Want to prove the b/w stability of $Q^* y_i$ given the condition

$$(Q_1 + \delta Q) y_1 = b_1 \quad \|\delta Q\| = O(\epsilon). \qquad (*)$$

Pf: denote the real mathematics as f : from Q to y !!

the algorithm as g; $\quad g(Q_1) = y_1$

but $(*)$ $\quad (Q_1 + \delta Q) y_1 = b_1$ solve it mathematically

$\Rightarrow \quad f(Q_1 + \delta Q) = y_1$

$$\boxed{\begin{array}{l} \Rightarrow \quad f(Q_1 + \delta Q) = g(Q_1) \\ \dfrac{\|\delta Q\|}{\|Q\|} = O(\epsilon_m) = \|\delta Q\| \\ \quad \xrightarrow{} Q \text{ is unitary.} \end{array}}$$

Backward substitution is also backward stable.

$$(R_1 + \delta R) x_1 = y_1, \quad \frac{\|\delta R\|}{\|R\|} = O(\epsilon_m) \qquad (**)$$

Thm 8.?

$$b = \underbrace{(Q_1 + \delta Q)(R_1 + \delta R)}_{QR \text{ of } A} x_1$$

$$= (Q_1 R_1 + Q_1 \delta R + \delta Q R_1 + \delta Q \delta R) x_1$$

$$Ax = \underset{\text{(fixed)}}{b}$$

By Thm 8.1

$$\Rightarrow \quad b = (A + \underbrace{\delta A + Q_1 \delta R + \delta Q R_1 + \delta Q \delta R}_{\Delta A}) x_1$$

We NTS. $\dfrac{\|\Delta A\|}{\|A\|} = O(\epsilon_m)$

Thm 8.1 ✓ $\quad Q_1 R_1 = A + \delta A$

$$R_1 = Q_1^* A + Q_1^* \delta A$$

$$\|R_1\| \le \overset{=1 \text{ due to "unitary"}}{\cancel{\|Q_1^*\|}} \cdot (\|A\| + \|\delta A\|)$$

$$\frac{\|R_1\|}{\|A\|} \le O(1) + O(\epsilon_m)$$

why $\dfrac{\|\delta A\|}{\|A\|} = O(\epsilon_m)$ b/c thm 8.1

$$\frac{\|R_1\|}{\|A\|} \le O(1)$$

$$\frac{\|Q_1 \delta R\|}{\|A\|} \le \overset{=1}{\cancel{\|Q_1\|}} \cdot \underbrace{\frac{\|\delta R\|}{\|R_1\|}}_{O(\epsilon_m)} \cdot \frac{\|R_1\|}{\|A\|} \overset{**}{\le} O(\epsilon_m) \cdot O(1)$$

$$= O(\epsilon_m)$$

$$\frac{\|\delta Q R_1\|}{\|A\|} \le \frac{\|\delta Q\| \cdot \overset{O(1)}{\boxed{\|R_1\|}}}{\boxed{\|A\|}} \overset{(*)}{\le} O(\epsilon_m)$$

$$\frac{\|\delta Q \delta R\|}{\|A\|} \le \underbrace{\|\delta Q\|}_{O(\epsilon_m)} \cdot \underbrace{\frac{\|\delta R\|}{\|A\|}}_{O(\epsilon_m)} = O(\epsilon_m^2)$$

$$\to \frac{\|\delta R\|}{\|R_1\|} \cdot \frac{\|R_1\|}{\|A\|}$$

$$\Rightarrow \quad \frac{\|\Delta A\|}{\|A\|} = O(\epsilon_m) \quad \checkmark$$

The total perturbation then satisfies,

$$\frac{\|\Delta A\|}{\|A\|} \leq \mathcal{O}(\epsilon_m). \tag{44}$$

□

Combining all theorems we have,

Theorem 8.3. The solution x_1 by the QR algorithm satisfies,

$$\frac{\|x_1 - x\|}{\|x\|} = \mathcal{O}(\kappa(A)\epsilon_m). \tag{45}$$

Proof. Theorem 3.1 gives the conditioning of the problem. Theorem 8.2 gives the backward stability of the problem. By Theorem 5.3, the algorithm is accurate. □

① $Ax = b$, conditioning, $\kappa(A)$

② b/w stability $O(\epsilon_m)$ [thm 8.2]

③ Thm 5.3

$$\Rightarrow \quad \frac{\|x - x_1\|}{\|x\|} = O(\kappa(A)\,\epsilon_m)$$