1

On the Condition of Numerical Problems and the Numbers that Measure It

1.1 The Size of Errors

> *Since none of the numbers we take out from logarithmic or trigonometric tables admit of absolute precision, but are all to a certain extent approximate only, the results of all calculations performed by the aid of these numbers can only be approximately true. [...] It may happen, that in special cases the effect of the errors of the tables is so augmented that we may be obliged to reject a method, otherwise the best, and substitute another in its place.*
>
> Carl Friedrich Gauss, *Theoria Motus*

The heroes of numerical mathematics (Euler, Gauss, Lagrange, ...) developed a good number of the algorithmic procedures which make the essence of Numerical Analysis. At the core of these advances there was the invention of calculus. And underlying the latter, the field of real numbers.

The dawn of the digital computer, in the decade of the 40s, allowed the execution of these procedures on increasingly large data, an advance which, however, made even more patent the fact that real numbers cannot be encoded with a finite number of bits and, therefore, that computers had to work with approximations only. With the increased length of the computations the systematic rounding of all occurring quantities could now accumulate to a higher extend. Occasionally, as already remarked by Gauss, the errors affecting the outcome of a computation were so big as to make it irrelevant.

Expressions like "the error is big" beg the question, how does one measure an error? To approach this question, let us first assume that the object whose error we are considering is a single number x encoding a quantity which may take values on an open real interval. An error of magnitude 1 may yield another real number $\tilde{x}$ with value either $x-1$ or $x+1$. Intuitively, this will be harmless or devastating depending on the magnitude of x itself. Thus, for $x = 10^6$ the error above is hardly noticeable but for $x = 10^{-3}$ it is certainly not (and may

even change basic features of x such as being positive). A relative measure of the error appears to convey more meaning. We therefore define[1]

$$\mathsf{RelError}(x) = \frac{|\tilde{x} - x|}{|x|}.$$

Note that this expression is well-defined only when $x \neq 0$.

How does this measure extends to elements $x \in \mathbb{R}^m$? We want to consider relative errors as well but, how does one relativize? There are essentially two ways:

Componentwise: Here we look at the relative error in each component taking as error for x the maximum of them. That is, for $x \in \mathbb{R}^m$ such that $x_i \neq 0$ for $i = 1, \ldots, m$, we define

$$\mathsf{RelError}(x) = \max_{i \leq m} \mathsf{RelError}(x_i).$$

Normwise: Endowing $\mathbb{R}^m$ with a norm allows one to mimic, for $x \neq 0$, the definition for the scalar case. We obtain

$$\mathsf{RelError}(x) = \frac{\|\tilde{x} - x\|}{\|x\|}.$$

Needless to say, the normwise measure depends on the choice of the norm.

1.2 The Cost of Erring

How do round-off errors affect computations? The answer to this question depends on a number of factors: the problem being solved, the data at hand, the algorithm used, the machine precision (as well as other features of the computer's arithmetic), ... While it is possible to consider all these factors together, a number of idealizations leading to the consideration of simpler versions of our question appears as a reasonable —if not necessary— course of action. The notion of condition is the result of some of these idealizations. More specifically, assume that the problem being solved can be described by a function

$$\varphi : \mathcal{D} \subseteq \mathbb{R}^m \to \mathbb{R}^q$$

where $\mathcal{D}$ is an open subset of $\mathbb{R}^m$. Assume as well that the computation of φ is performed by an algorithm with infinite precision (that is, there are no round-off errors during the execution of this algorithm). All errors in the computed value arise as a consequence of possible errors while reading the input (which we will call *perturbations*). Our question above then takes the form

[1] To be completely precise we should write $\mathsf{RelError}(x, \tilde{x})$. In all what follows, however, to simplify notation, we will omit the perturbation $\tilde{x}$ and write simply $\mathsf{RelError}(x)$.

How large is the output error with respect to the input perturbation?

The *condition number* of input $a \in \mathbb{R}^m$ (with respect to problem φ) is, roughly speaking, the worst possible magnification of the output error with respect to a small input perturbation. More formally,

$$\mathsf{cond}^{\varphi}(a) = \lim_{\delta \to 0} \sup_{\mathsf{RelError}(a) \leq \delta} \frac{\mathsf{RelError}(\varphi(a))}{\mathsf{RelError}(a)}. \tag{1.1}$$

This expression defines the condition number as a limit. For small values of δ we can consider the approximation

$$\mathsf{cond}^{\varphi}(a) \approx \sup_{\mathsf{RelError}(a) \leq \delta} \frac{\mathsf{RelError}(\varphi(a))}{\mathsf{RelError}(a)}$$

and, for practical purposes, the approximate bound

$$\mathsf{RelError}(\varphi(a)) \lessapprox \mathsf{cond}^{\varphi}(a)\mathsf{RelError}(a) \tag{1.2}$$

or yet, using "little oh" notation for $\mathsf{RelError}(a) \to 0$,

$$\mathsf{RelError}(\varphi(a)) \leq \mathsf{cond}^{\varphi}(a)\mathsf{RelError}(a) + o(\mathsf{RelError}(a)). \tag{1.3}$$

Expression (1.1) defines a familiy of condition numbers for the pair (φ, a). Errors can be measured both componentwise or normwise and in the latter case, there are a good number of norms to choose from. The choice of normwise or componentwise measures for the errors has given rise to three kinds of condition numbers (condition numbers for normwise perturbations and componentwise output errors are not considered in the literature).

		PERTURBATION	
		`normwise`	`componentwise`
OUTPUT ERROR	`normwise`	*normwise*	*mixed*
	`componentwise`		*componentwise*

We will generically denote normwise condition numbers by $\mathsf{cond}^{\varphi}(a)$, mixed condition numbers by $\mathsf{M}^{\varphi}(a)$, and componentwise condition numbers by $\mathsf{Cw}^{\varphi}(a)$. We may skip the superscript $^{\varphi}$ if it is clear from the context. In the case of componentwise condition numbers one may be interested in considering the relative error for each of the output components separately. Thus, for $j \leq q$ one defines

$$\mathsf{Cw}_j^{\varphi}(a) = \lim_{\delta \to 0} \sup_{\mathsf{RelError}(a) \leq \delta} \frac{\mathsf{RelError}(\varphi(a)_j)}{\mathsf{RelError}(a)}$$

and one has $\mathsf{Cw}^{\varphi}(a) = \max_{j \leq q} \mathsf{Cw}_j^{\varphi}(a)$.

The consideration of a normwise, mixed, or componentwise condition number will be determined by the characteristics of the situation at hand. To illustrate

this let's look at data perturbation. The two main reasons to consider such perturbations are inaccurate data reading and backward-error analysis.

In the first case the idea is simple. We are given data which we know inaccurate. This may be because we obtained it by measurements with finite precision (e.g., when weighting an object the weight is displayed with a few digits only) or because our data is the result of an inaccurate computation.

The idea of backward-error analysis is less simple (but very elegant). For a problem φ we may have many algorithms that solve it. While all of them ideally compute φ when endowed with infinite precision, under the presence of errors they will only compute approximations of this function. At times, for a problem φ and a finite-precision algorithm $\mathscr{A}^{\varphi}$ solving it, it is possible to show that for all $a \in \mathbb{R}^m$ there exists $e \in \mathbb{R}^m$ satisfying

$$\begin{aligned} (*) \qquad & \mathcal{A}^{\varphi}(a) = \varphi(a+e), \text{ and} \\ (**) \qquad & e \text{ is small w.r.t. } a. \end{aligned}$$

In this situation —which we refer by saying that $\mathscr{A}^{\varphi}$ is *backward-stable*— information on how small exactly is e (i.e., how much is $\mathsf{RelError}(a)$) together with the condition number of a directly yield bounds on the error of the computed quantity $\mathscr{A}^{\varphi}(a)$. For instance, if (**) above takes the form

$$\|e\| \leq m^3 10^{-6} \|a\|$$

we will deduce, using (1.2), that

$$\|\mathscr{A}^{\varphi}(a) - \varphi(a)\| \lessapprox \mathsf{cond}^{\varphi}(a) m^3 10^{-6} \|\varphi(a)\|. \tag{1.4}$$

No matter whether because of inaccurate data reading or because of a backward-error analysis we will measure the perturbation of a in accordance with the situation at hand. If, for instance, we are reading data in a way that each component a_i satisfies $\mathsf{RelError}(a_i) \leq 5 \times 10^{-8}$ we will measure perturbations in a componentwise manner. If, in contrast, a backward-error analysis yields an e satisfying $\|e\| \leq m^3 \|a\| 10^{-6}$ we will have to measure perturbations in a normwise manner.

While we may have more freedom in the way we measure the output error there are situations in which a given choice seems to impose itself. Such a situation could arise when the outcome of the computation at hand is going to be the data of another computation. If perturbations of the latter are measured, say, componentwise, we will be interested in doing the same with the output error of the former. A striking example where error analysis can be only appropiately explained using componentwise conditioning is the solution of triangular systems of equations. We will return to this issue on Chapter 4.

At this point it is perhaps convenient to emphasize a distinction between *condition* and (backward) *stability*. Given a problem φ the former is a property of the input only. That is, it is independent on the possible algorithms used to

compute φ. In contrast, backward stability, at least in the sense defined above, is a property of an algorithm $\mathscr{A}^{\varphi}$ computing φ which holds for all data $a \in \mathbb{R}^m$ (and is therefore independent of particular data instances).

Expressions like (1.4) are known as forward-error analyses and algorithms $\mathscr{A}^{\varphi}$ yielding a small value of $\frac{\|\mathscr{A}^{\varphi}(a)-\varphi(a)\|}{\|\varphi(a)\|}$ are said to be *forward-stable*. It is important to mention that while backward-error analyses immediately yield forward-error bounds, some problems do not admit backward-error analysis and therefore, their error-analysis must be carried forward.

It is time to have a closer look at the way errors are produced in a computer.

1.3 Finite-precision Arithmetic and Loss of Precision

1.3.1 Precision ...

Although the details of computer arithmetic may vary with computers and software implementations, the basic idea was agreed upon shortly after the dawn of digital computers. It consisted in fixing positive integers $\beta \geq 2$ (the *basis* of the representation), t (its *precision*), and e_0, and approximating non-zero real numbers by rational numbers of the form

$$z = \pm \left(\frac{m}{\beta^t}\right) \beta^e$$

with $m \in \{1, \ldots, \beta^t\}$ and $e \in \{-e_0, \ldots, e_0\}$. The fraction $\frac{m}{\beta^t}$ is called the *mantissa* of z and the integer e its *exponent*. The condition $|e| \leq e_0$ sets limits on how big (and how small) z may be. Although these limits may give rise to situations were (the absolute value of) the number to be represented is to large (*overflow*) or to small (*underflow*) for the possible values of z, the value of e_0 in most implementations is large enough to make these phenomenon rare in practice. Idealizing a bit, we may assume $e_0 = \infty$.

As an example, taking $\beta = 10$ and $t = 12$ we can approximate

$$\pi^8 \approx 0.948853101607 \times 10^4.$$

The relative error in this approximation is bounded by 1.1×10^{-12} . Note that t is the number of correct digits of the approximation. Actually, for any real number x, by appropriately rounding and truncating an expansion of x we can obtain a number $\tilde{x}$ as above satisfying that $\tilde{x} = x(1+\delta)$ with $|\delta| \leq \frac{\beta^{-t+1}}{2}$. That is

$$\mathsf{RelError}(x) \leq \frac{\beta^{-t+1}}{2}.$$

More generally, whenever a real number x is approximated by $\tilde{x}$ satisfying an inequality as the one above we say that $\tilde{x}$ *approximates* x *with* t *correct digits*[2].

[2]This notion reflect the intuitive idea of significant figures modulo carry differences. The number 0.9999 approximates 1 with a precision $t = 10^{-4}$. Yet, their first significant digits are different.

Leaving aside the details such as the choice of basis or the particular way a real number is truncated to obtain a number as described above, we may summarize the main features of computer arithmetic (recall, we assume $e_0 = \infty$) by stating the existence of a subset $\mathbb{F} \subset \mathbb{R}$ containing 0 (the *floating-point numbers*), a *rounding map* $\mathsf{round} : \mathbb{R} \to \mathbb{F}$, and a *round-off unit* (also called *machine epsilon*) $0 < \epsilon_{\mathsf{mach}} < 1$, satisfying the following properties:

(i) For any $x \in \mathbb{F}$, $\mathsf{round}(x) = x$. In particular $\mathsf{round}(0) = 0$.

(ii) For any $x \in \mathbb{R}$, $\mathsf{round}(x) = x(1+\delta)$ with $|\delta| \leq \epsilon_{\mathsf{mach}}$.

Furthermore, one can take $\epsilon_{\mathsf{mach}} = \frac{\beta^{-t+1}}{2}$ and therefore $|\log \epsilon_{\mathsf{mach}}| = t - \log_\beta \frac{\beta}{2}$.

Arithmetic operations on $\mathbb{F}$ are defined following the scheme

$$x \,\tilde{\circ}\, y = \mathsf{round}(x \circ y)$$

for any $x, y \in \mathbb{F}$ and $\circ \in \{+, -, \times, /\}$ so that

$$\tilde{\circ} : \mathbb{F} \times \mathbb{F} \to \mathbb{F}.$$

It follows from (ii) above that, for any $x, y \in \mathbb{F}$ we have

$$x \,\tilde{\circ}\, y = (x \circ y)(1+\delta), \qquad |\delta| \leq \epsilon_{\mathsf{mach}}.$$

Other operations may be also considered. Thus, a floating-point version $\tilde{\sqrt{}}$ of the square root would similarly satisfy

$$\tilde{\sqrt{x}} = \sqrt{x}(1+\delta), \qquad |\delta| \leq \epsilon_{\mathsf{mach}}.$$

When combining many operations in floating-point arithmetic expressions such as $(1+\delta)$ above naturally appear. To simplify round-off analyses it is useful to consider the quantities, for $k \geq 1$ and $k\epsilon_{\mathsf{mach}} < 1$,

$$\gamma_k := \frac{k\epsilon_{\mathsf{mach}}}{1 - k\epsilon_{\mathsf{mach}}} \tag{1.5}$$

and to denote by θ_k *any* number satisfying $|\theta_k| \leq \gamma_k$. In this sense, θ_k represents a set of numbers and different occurrences of θ_k in a proof may denote different numbers. Note that

$$\gamma_k \leq k\epsilon_{\mathsf{mach}} \qquad \text{if } k+1 \leq \epsilon_{\mathsf{mach}}^{-1}. \tag{1.6}$$

The proof of the following proposition can be found in Chapter 3 of [11].

Proposition 1.1. *The following relations hold (assuming all quantities are well defined):*

1) $(1+\theta_k)(1+\theta_j) = 1 + \theta_{k+j}$,

2)
$$\frac{1+\theta_k}{1+\theta_j} = \begin{cases} 1+\theta_{k+j} & \text{if } j \le k \\ 1+\theta_{k+2j} & \text{if } j > k, \end{cases}$$

3) *If* $\max\{k\epsilon_{\mathsf{mach}}, j\epsilon_{\mathsf{mach}}\} \le 1/2$ *then* $\gamma_k\gamma_j \le \gamma_{\min\{k,j\}}$,

4) $i\gamma_k \le \gamma_{ik}$,

5) $\gamma_k + \epsilon_{\mathsf{mach}} \le \gamma_{k+1}$,

6) $\gamma_k + \gamma_j + \gamma_k\gamma_j \le \gamma_{k+j}$. □

1.3.2 . . . and the way we lose it

When computing an arithmetic expression q with a round-off algorithm, errors will accumulate and we will obtain another quantity which we denote by $\mathsf{fl}(q)$. We will also write $\mathsf{Error}(q) = |q - \mathsf{fl}(q)|$ so that $\mathsf{RelError}(q) = \frac{\mathsf{Error}(q)}{|q|}$.

Assume now that q is computed with a real number algorithm $\mathscr{A}$ executed using floating point arithmetic from data a (a formal model for real number algorithms was given in [3]). No matter how precise is the representation we are given of the entries of a, these entries will be rounded to t digits. Hence t (or, being roughly the same, $|\log_\beta \epsilon_{\mathsf{mach}}|$) is the precision of our data. On the other hand, the number of correct digits in $\mathsf{fl}(q)$ is approximately $-\log_\beta \mathsf{RelError}(q)$. Therefore, the value

$$\mathsf{LoP}(q) := \log_\beta \frac{\mathsf{RelError}(q)}{\epsilon_{\mathsf{mach}}} = |\log_\beta \epsilon_{\mathsf{mach}}| + \log_\beta \mathsf{RelError}(q)$$

quantifies the *loss of precision* in the computation of q. To extend this notion to the computation of vectors $v = (v_1, \ldots, v_q) \in \mathbb{R}^q$ we need to fix a measure for the precision of the computed $\mathsf{fl}(v) = (\mathsf{fl}(v_1), \ldots, \mathsf{fl}(v_q))$: componentwise or normwise.

In the componentwise case, we have that

$$-\log_\beta \mathsf{RelError}(v) = -\log_\beta \max_{i\le q} \frac{|\mathsf{fl}(v_i) - v_i|}{|v_i|} = \min_{i\le q} \left(-\log_\beta \frac{|\mathsf{fl}(v_i) - v_i|}{|v_i|} \right)$$

so that the precision of v is the smallest of the precisions of its components.

For the normwise measure we take the precision of v to be

$$-\log_\beta \mathsf{RelError}(v) = -\log_\beta \frac{\|\mathsf{fl}(v) - v\|}{\|v\|}.$$

This choice has both the pros and cons of viewing v as a whole and not as the aggregation of its components.

For both the componentwise and the normwise measures we can consider ϵ_{mach} as a measure of the worst possible relative error $\mathsf{RelError}(a)$ when we read data a with round-off unit ϵ_{mach} since in both cases

$$\max_{\tilde{a}||\tilde{a}_i - a_i| \leq \epsilon_{\mathsf{mach}}|a_i|} \mathsf{RelError}(a) = \epsilon_{\mathsf{mach}}.$$

Hence, $|\log_\beta \epsilon_{\mathsf{mach}}|$ represents in both cases the precision of data. We therefore define the loss of precision in the computation of $\varphi(a)$ to be

$$\mathsf{LoP}(\varphi(a)) := \log_\beta \frac{\mathsf{RelError}(\varphi(a))}{\epsilon_{\mathsf{mach}}} = |\log_\beta \epsilon_{\mathsf{mach}}| + \log_\beta \mathsf{RelError}(\varphi(a)). \tag{1.7}$$

Remark 1.2. By associating $\mathsf{RelError}(a) \approx \epsilon_{\mathsf{mach}}$ we may view the logarithm of a condition number $\log_\beta \mathsf{cond}^\varphi(a)$ as a measure of the worst possible loss of precision in a computation of $\varphi(a)$ in which the only error occurs when reading the data.

To close this section we prove a result putting together —and making precise— a number of issues dealt with so far. For data $a \in \mathbb{R}^m$ we call m the *size* of a and we write $\mathsf{size}(a) = m$. Ocasionally, this size is a function of a few integers, the *dimensions* of a, the set of which we denote by $\mathsf{dims}(a)$. For instance, a $p \times q$ matrix has dimensions p and q and size pq.

Theorem 1.3. *Let $\mathscr{A}^\varphi$ be a finite-precision algorithm with round-off unit ϵ_{mach} computing a function $\varphi : \mathcal{D} \subseteq \mathbb{R}^m \to \mathbb{R}^q$. Assume $\mathscr{A}^\varphi$ satisfies the following backward bound*

$$\mathscr{A}^\varphi(a) = \varphi(\tilde{a})$$

with $\tilde{a}$ such that

$$\mathsf{RelError}(a) \leq f(\mathsf{dims}(a))\epsilon_{\mathsf{mach}} + o(\epsilon_{\mathsf{mach}})$$

for some positive function f and where the "little oh" is for $\epsilon_{\mathsf{mach}} \to 0$. Then the computed $\mathscr{A}^\varphi(a)$ satisfies the forward bound

$$\mathsf{RelError}(\varphi(a)) \leq f(\mathsf{dims}(a))\mathsf{cond}^\varphi(a)\epsilon_{\mathsf{mach}} + o(\epsilon_{\mathsf{mach}})$$

and the loss of precision in the computation (in base β) is bounded as

$$\mathsf{LoP}(\varphi(a)) \leq \log_\beta f(\mathsf{dims}(a)) + \log_\beta \mathsf{cond}^\varphi(a) + o(1).$$

Here cond^φ refers to the condition number defined in (1.1) *with the same measures (normwise or componentwise) for $\mathsf{RelError}(a)$ and $\mathsf{RelError}(\varphi(a))$ as those in the backward and forward bounds above, respectively.*

Proof. The forward bound immediately follows from the backward bound and (1.3). For the loss of precision we have

$$\begin{aligned} \log_\beta \mathsf{RelError}(\varphi(a)) &\leq \log_\beta \left(f(\mathsf{dims}(a))\mathsf{cond}^\varphi(a)\epsilon_{\mathsf{mach}}(1 + o(1)) \right) \\ &\leq \log_\beta f(\mathsf{dims}(a)) + \log_\beta \mathsf{cond}^\varphi(a) - \left|\log_\beta \epsilon_{\mathsf{mach}}\right| + o(1) \end{aligned}$$

from which the statement follows. □

1.4 An Example: Matrix-vector Multiplication

It is perhaps time to illustrate the notions introduced so far by analyzing a simple problem namely, matrix-vector multiplication. We begin with a (componentwise) backward stability analysis.

Proposition 1.4. *There is a finite-precision algorithm $\mathscr{A}$ which, with input $A \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$, computes the product Ax. If $\epsilon_{\mathsf{mach}}(\lceil \log_2 n \rceil + 1) < 1$ then the computed vector $\mathsf{fl}(Ax)$ satisfies $\mathsf{fl}(Ax) = \tilde{A}x$ with*

$$|\tilde{a}_{ij} - a_{ij}| \leq (\lceil \log_2 n \rceil + 1)\epsilon_{\mathsf{mach}}|a_{ij}|.$$

Proof. Let $b = Ax$. For $i = 1, \ldots, m$ we have

$$b_i = a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n.$$

For the first product in the right-hand side we have $\mathsf{fl}(a_{i1}x_1) = a_{i1}x_1(1+\delta)$ with $|\delta| \leq \epsilon_{\mathsf{mach}} \leq \frac{\epsilon_{\mathsf{mach}}}{1-\epsilon_{\mathsf{mach}}} = \gamma_1$. That is, $\mathsf{fl}(a_{i1}x_1) = a_{i1}x_1(1+\theta_1)$ and similarly $\mathsf{fl}(a_{i2}x_2) = a_{i2}x_2(1+\theta_1)$. Note that the two occurrences of θ_1 here denote two different quantities. Hence, using Proposition 1.1,

$$\begin{aligned}\mathsf{fl}(a_{i1}x_1 + a_{i2}x_2) &= \big(a_{i1}x_1(1+\theta_1) + a_{i2}x_2(1+\theta_1)\big)(1+\theta_1)\\ &= a_{i1}x_1(1+\theta_2) + a_{i2}x_2(1+\theta_2).\end{aligned}$$

By the same reasoning, $\mathsf{fl}(a_{i3}x_3 + a_{i4}x_4) = a_{i3}x_3(1+\theta_2) + a_{i4}x_4(1+\theta_2)$ and therefore

$$\begin{aligned}&\mathsf{fl}(a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 + a_{i4}x_4)\\ &\quad= \big(a_{i1}x_1(1+\theta_2) + a_{i2}x_2(1+\theta_2) + a_{i3}x_3(1+\theta_2) + a_{i4}x_4(1+\theta_2)\big)(1+\theta_1)\\ &\quad= a_{i1}x_1(1+\theta_3) + a_{i2}x_2(1+\theta_3) + a_{i3}x_3(1+\theta_3) + a_{i4}x_4(1+\theta_3).\end{aligned}$$

Continuing in this way we obtain

$$\mathsf{fl}(b_i) = \tilde{a}_{i1}x_1 + \tilde{a}_{i2}x_2 + \ldots + \tilde{a}_{in}x_n$$

with $\tilde{a}_{ij} = a_{ij}(1+\theta_{\lceil \log_2 n\rceil+1})$. The result follows using (1.6). □

Remark 1.5. Note that the algorithm computing Ax is implicitly given in the proof of Proposition 1.4. This algorithm uses a balanced tree-like structure for the sums. The order of the sums cannot be arbitrarily altered: the operations $\tilde{+}$ and $\tilde{\cdot}$ are nonassociative.

We next estimate the componentwise condition number of matrix-vector multiplication. In doing so, we note that in the backward analysis of Proposition 1.4 only the entries of A are perturbed. Those of x aren't. This feature allows one to consider the condition of data (A, x) for perturbations of A only. Such a situation is common and also arises when data is structured (e.g., unit upper triangular matrices have zeros below the diagonal and ones on the diagonal) or contains entries which are known to be integers.

Proposition 1.6. *The componentwise condition numbers* $\mathsf{Cw}_i(A,x)$ *of matrix-vector multiplication, for perturbations of* A *only, satisfy*

$$\mathsf{Cw}_i(A,x) \le |\sec(a_i,x)|$$

where a_i *denotes the* i*th row of* A *and* $\sec(a_i,x) = \frac{1}{\cos(a_i,x)}$ *denotes the secant of the angle* θ_i *it makes with* x*.*

Proof. Let $\tilde{A} = A + E$ be a perturbation of A with $E = (e_{ij})$. By definition, $|e_{ij}| \le \mathsf{RelError}(A)|a_{ij}|$, for all i,j, hence $\|e_i\| \le \mathsf{RelError}(A)\|a_i\|$, for all i (here $\|\ \|$ denotes Euclidean norm in $\mathbb{R}^n$). We obtain,

$$\mathsf{RelError}((Ax)_i) = \frac{|e_i^{\mathrm{T}}x|}{|a_i^{\mathrm{T}}x|} \le \frac{\|e_i\|\|x\|}{|a_i^{\mathrm{T}}x|} \le \mathsf{RelError}(A)\frac{\|a_i\|\|x\|}{|a_i^{\mathrm{T}}x|}.$$

This implies that

$$\begin{aligned}\mathsf{Cw}_i(A,x) &= \lim_{\delta\to 0}\sup_{\mathsf{RelError}(A)\le\delta}\frac{\mathsf{RelError}((Ax)_i)}{\mathsf{RelError}(A)}\\ &\le \frac{\|a_i\|\|x\|}{|a_i^{\mathrm{T}}x|} = \frac{1}{|\cos(a_i,x)|} = |\sec(a_i,x)|.\end{aligned}$$

□

A bound for the loss of precision in the componentwise context follows.

Corollary 1.7. *In the componentwise setting, for all* i *such that* $b_i = (Ax)_i \neq 0$*,*

$$\mathsf{RelError}(b_i) \le |\sec(a_i,x)|(\lceil\log_2 n\rceil + 1)\epsilon_{\mathsf{mach}} + o(\epsilon_{\mathsf{mach}})$$

and

$$\mathsf{LoP}(b_i) \le \log_\beta|\sec(a_i,x)| + \log_\beta(\lceil\log_2 n\rceil + 1) + o(1).$$

Proof. Immediate from Propositions 1.4 and 1.6 and Theorem 1.3. □

The corollary above states that if we are working with $|\log_\beta \epsilon_{\mathsf{mach}}|$ bits of precision, we compute a vector $\mathsf{fl}(Ax)$ whose non-zero entries have, approximately, at least

$$|\log_\beta \epsilon_{\mathsf{mach}}| - \log_\beta|\sec(a_i,x)| - \log_\beta\log_2 n$$

bits of precision. This is a satisfying result. One may, nevertheless, wonder about the (absolute) error for the zero components of Ax. In this case a normwise analysis may be more appropriate.

To proceed with a normwise analysis we first need to choose a norm in the space of $m \times n$ matrices. For simplicity, we choose

$$\|A\|_\infty = \max_{\|x\|_\infty = 1}\|Ax\|_\infty.$$

It is well known that

$$\|A\|_\infty = \max_{i\le n} \|a_i\|_1. \tag{1.8}$$

Now note that it follows from Proposition 1.4 that the perturbation $\tilde{A}$ in its statement satisfies

$$\|\tilde{A} - A\|_\infty \le \gamma_{\lceil \log_2 n\rceil+1}\|A\|_\infty. \tag{1.9}$$

Therefore, we do have a normwise backward-error analysis. In addition, a normwise version of Proposition 1.6 can be easily obtained.

Proposition 1.8. *The normwise condition number* $\mathsf{cond}(A,x)$ *of matrix-vector multiplication, for perturbations on A only, satisfies*

$$\mathsf{cond}(A,x) = \frac{\|A\|_\infty \|x\|_\infty}{\|Ax\|_\infty}.$$

Proof. We have

$$\begin{aligned}
\mathsf{cond}(A,x) &= \lim_{\delta\to 0} \sup_{\mathsf{RelError}(A)\le\delta} \frac{\mathsf{RelError}(Ax)}{\mathsf{RelError}(A)} \\
&= \lim_{\delta\to 0} \sup_{\|\tilde{A}-A\|_\infty \le \delta\|A\|_\infty} \frac{\|\tilde{A}x - Ax\|_\infty}{\|Ax\|_\infty} \frac{\|A\|_\infty}{\|\tilde{A}-A\|_\infty} \\
&\le \frac{\|A\|_\infty\|x\|_\infty}{\|Ax\|_\infty}.
\end{aligned}$$

Actually, equality holds. In order to see this assume, without loss of generality, that $\|x\|_\infty = |x_1|$. Set $\tilde{A} = A+E$ where $e_{11} = \delta$ and $e_{ij} = 0$ otherwise. Then we have $\|\tilde{A}x - Ax\|_\infty = \|Ex\|_\infty = \delta|x_1| = \|E\|_\infty\|x\|_\infty = \|\tilde{A}-A\|_\infty\|x\|_\infty$. □

Again, a bound for the loss of precision immediately follows.

Corollary 1.9. *In the normwise setting, when* $Ax \neq 0$*,*

$$\mathsf{LoP}(Ax) \le \log_\beta\left(\frac{\|A\|_\infty\|x\|_\infty}{\|Ax\|_\infty}\right) + \log_\beta(\lceil\log_2 n\rceil + 1) + o(1).$$

Proof. It is an immediate consequence of (1.9), Proposition 1.8, and Theorem 1.3. □

Remark 1.10. If $m = n$ and A is invertible, it is possible to give a bound on the normwise condition which is independent of x. Using that $x = A^{-1}Ax$ we deduce $\|x\|_\infty \le \|A^{-1}\|_\infty\|Ax\|_\infty$ and therefore, by Proposition 1.8, $\mathsf{cond}(A,x) \le \|A^{-1}\|_\infty\|A\|_\infty$. A number of readers may find this expression familiar.

1.5 The Many Faces of Condition

The previous sections attempted to introduce condition numbers by retracing the way these numbers were introduced: as a way of measuring the effect of data perturbations. The expression "condition number" was first used by Turing [19] to denote a condition number for linear equation solving, independently introduced by him and by von Neumann and Goldstine [21] in the late 1940's. Expressions like "ill-conditioned set [of equations]" to denote systems with a large condition number were also introduced in [19].

Conditioning, however, was eventually related to other issues in computation and this, together with their original role in error-propagation analysis, triggered research on different aspects of the subject. We briefly describe some of them in what follows.

1.5.1 Condition and complexity

In contrast with direct methods (such as Gaussian elimination), the number of times that a certain basic procedure is repeated in iterative methods is not data independent. In the analysis of this dependence on the data at hand it was early realized that, quite often, one could express it by using its condition number. That is, the number of iterations the algorithm $\mathscr{A}^\varphi$ would perform with data $a \in \mathbb{R}^m$ could be bounded by a function of m, $\mathsf{cond}^\varphi(a)$, and —in the case of an algorithm computing an ε-approximation of the desired solution— the accuracy ε. A very satisfying bound would have the form

$$\text{number of iterations of } \mathscr{A}^\varphi(a) \leq \left(m + \log \mathsf{cond}^\varphi(a) + \log\left(\frac{1}{\varepsilon}\right)\right)^{\mathcal{O}(1)} \tag{1.10}$$

and a less satisfying (but still acceptable in many cases) would have $\log \mathsf{cond}^\varphi(a)$ replaced by $\mathsf{cond}^\varphi(a)$ and/or $\log\left(\frac{1}{\varepsilon}\right)$ replaced by $\frac{1}{\varepsilon}$. We will encounter several instances of this *condition based analysis* in the coming chapters.

1.5.2 Computing condition numbers

Irrespective of whether relative errors are measured normwise or componentwise, the expression (1.1) defining the condition number of a (for the problem φ) is hardly usable. Not surprisingly then, one of the main lines of research regarding condition numbers has focused on finding equivalent expressions for $\mathsf{cond}^\varphi(a)$ which would be directly computable or, if this appears to be out of reach, tight enough bounds with this property. We have done so for the problem of matrix-vector multiplication in Propositions 1.6 and 1.8 (for the componentwise and normwise cases, respectively).

1.5.3 Condition of random data

How many iterations does an iterative algorithm need to perform to compute $\varphi(a)$? To answer this questions we need $\mathsf{cond}^{\varphi}(a)$. And to compute $\mathsf{cond}^{\varphi}(a)$ we would like a simple expression like those in Propositions 1.6 and 1.8. A second look at these expressions, however, shows that they seem to require $\varphi(a)$, the quantity we were interested in the first place. For, in the componentwise case, we need to compute $\operatorname{sec}(a_i, x)$ —and hence $a_i^{\mathrm{T}} x$— for $i = 1, \ldots, n$, and in the normwise the expression $\|Ax\|_\infty$ speaks by itself. Worst of all, this is not an isolated situation. We will see that the condition number of a matrix A with respect to matrix inversion is expressed in terms of A^{-1} (or some norm of this inverse) and that a similar phenomenon occurs for each of the problems we consider. So, even though we do not formalize this situation as a mathematical statement we can informally describe it by saying that the computation of a condition number $\mathsf{cond}^{\varphi}(a)$ is never easier than the computation of $\varphi(a)$. The most elaborate reasoning around this issue was done by Renegar [14].

A similar problem appears with perturbation considerations. If we are only given a perturbation $\tilde{a}$ of data a, how can we know how accurate is $\varphi(\tilde{a})$? Even assuming we can accurately and fast compute cond^{φ} the most we could do is to compute $\mathsf{cond}^{\varphi}(\tilde{a})$, not $\mathsf{cond}^{\varphi}(a)$.

There are a number of situations where this seemingly circular situations can be broken. Instead of attempting to make a list of them (an exercise which can only result in boredom) we next describe a way out pioneered by John von Neumann (e.g., in [9]) and strongly advocated by Steve Smale in [17]. It consist of randomizing the data (i.e., in assuming a probabilistic distribution $\mathscr{D}$ in $\mathbb{R}^m$) and considering the tail

$$\operatorname*{Prob}_{a \sim \mathscr{D}} \{\mathsf{cond}^{\varphi}(a) \geq t\}$$

or the expected value (for $q \geq 1$)

$$\mathop{\mathbb{E}}_{a \sim \mathscr{D}} \left(\log^q \mathsf{cond}^{\varphi}(a)\right).$$

The former, together with a bound as in (1.10), would allow one to bound the probability that $\mathscr{A}^{\varphi}$ needs more than a given number of iterations. The latter, taking q to be the constant in the $\mathcal{O}(1)$ notation, to estimate the expected number of iterations. Furthermore, the latter again, now with $q = 1$, can be used to obtain an estimate of the average loss of precision for a problem φ (together with a backward stable algorithm $\mathscr{A}^{\varphi}$ if we are working with finite precision arithmetic).

For instance, for the example that made the substance of Section 1.4, we will prove that

$$\mathbb{E}(\log_\beta \mathsf{Cw}_i(A)) \leq \frac{1}{2} \log_\beta n + \mathcal{O}(1).$$

Using Corollary 1.7, this bound implies that the expected loss of precision in the computation of $(Ax)_i$ is at most $\frac{1}{2} \log_\beta n + \log_\beta \log_2 n + \mathcal{O}(1)$.

The probabilistic analysis proposed by von Neumann and Smale relies on the assumption of "evenly spread random data." A different approach was recently proposed that relies instead on the assumption of "non-random data affected by random noise." We will develop both approaches in this book.

1.5.4 Ill-posedness and condition

Let us return once more to the example of matrix-vector multiplication. If A and x are such that $Ax = 0$ then the denominator in $\frac{\|A\|_\infty \|x\|_\infty}{\|Ax\|_\infty}$ is zero and we may well define $\mathsf{cond}(A, x) = \infty$. This reflects the fact that no matter how small the absolute error in computing Ax, the relative error will be infinite. The quest for any relative precision is, in this case, a battle lost in advance. It is only fair to refer to instances like this with a name that betrays this hopelessness. We say that a is *ill-posed for* φ when $\mathsf{cond}^\varphi(a) = \infty$. Again, one omits the reference to φ when the problem is clear from the context but it goes without saying that the notion of ill-posedness, as that of condition, is with respect to a problem. It also depends on the way we measure errors. For instance, in our example, $\mathsf{Cw}(A, x) = \infty$ if and only if there exists $i \leq n$ such that $a_i^{\mathrm{T}} x = 0$ while for $\mathsf{cond}(A, x)$ to be infinity it is necessary (and sufficient) that $Ax = 0$.

The subset of $\mathbb{R}^m$ of ill-posed inputs is denoted by Σ^φ (or simply by Σ) and it has played a distinguished role in many developments in conditioning. To see why let us return (yes, once again) to matrix-vector multiplication, say in the componentwise setting. Recall we are considering x as fixed (i.e., not subject to perturbations). In this situation we take $\Sigma \subset \mathbb{R}^{n\times m}$ to be the set of matrices A such that $\mathsf{Cw}(A, x) = \infty$. We have $\Sigma = \bigcup_{i\leq n} \Sigma_i$ with

$$\Sigma_i = \{A \in \mathbb{R}^{n\times m} \mid \mathsf{Cw}_i(A, x) = \infty\} = \{A \in \mathbb{R}^{n\times m} \mid a_i^{\mathrm{T}} x = 0\}.$$

Now, recall, $\mathsf{Cw}_i(A, x) = \frac{1}{|\cos(a_i, x)|}$. If we denote by $\bar{a}_i$ the orthogonal projection of a_i on the space $x^\perp = \{y \in \mathbb{R}^m \mid y^{\mathrm{T}} x = 0\}$ then

$$\frac{1}{|\cos(a_i, x)|} = \frac{\|a_i\|}{\|a_i - \bar{a}_i\|}$$

and it follows that

$$\mathsf{Cw}_i(A, x) = \frac{\|a_i\|}{\mathsf{dist}(A, \Sigma_i)}.$$

That is, componentwise, the condition number of (A, x) is the inverse of the relativized distance from A to ill-posedness.

This is not an isolated phenomenon. On the contrary, it is a common occurrence that condition numbers can be expressed as, or at least bounded by, the inverse of a relativized distance to ill-posedness. We will actually meet this theme repeatedly in this book.