

# Lecture 20: Backward stability of an algorithm



School of Mathematical Sciences, Xiamen University

# 1. Floating point number

- For given integers  $p$  and  $\beta$ , in the IEEE floating point standard (founded in 1985, updated in 2008, being undated again now), the elements of the floating point number system  $\mathbf{F}$  are the number 0 together with numbers of the form

$$x = \pm d_1.d_2 \cdots d_p \times \beta^e$$

where the integers  $d_i$ ,  $e$  satisfy

$$0 \leq d_i \leq \beta - 1, \quad d_1 \neq 0, \quad e_{\min} \leq e \leq e_{\max}.$$

- One need to store sign bit ( $\pm$ ), exponent ( $e$ ), and mantissa ( $d_1.d_2 \cdots d_p$ ); but not the *base* or *radix* ( $\beta \geq 2$ ). Floating point number system usually uses  $\beta = 2$  (10 sometimes, 16 historically).

Precision	$\beta$	Bits	$p$	$e_{\min}$	$e_{\max}$	$\epsilon_{\text{machine}}$
Single (32)	2	1+8+23	24	-126	127	$2^{-24}$
Double (64)	2	1+11+52	53	-1022	1023	$2^{-53}$

**Reading.** Higham and Mary, **Mixed precision algorithms in numerical linear algebra**, Acta Numerica, 2022.

### 1.1. Limitations of digital representations

- Only a finite subset of the real numbers (or the complex numbers) can be represented. Therefore,
  - (i) the represented numbers cannot be arbitrarily large or small;
  - (ii) there are gaps between these numbers.

### 1.2. Floating point number machine accuracy

- In IEEE double precision arithmetic, the interval  $[1, 2]$  is represented by the discrete subset

$$1, \quad 1 + 2^{-52}, \quad 1 + 2 \times 2^{-52}, \quad 1 + 3 \times 2^{-52}, \quad \dots, \quad 2.$$

The interval  $[2, 4]$  is represented by the same numbers multiplied by 2,

$$2, \quad 2 + 2^{-51}, \quad 2 + 2 \times 2^{-51}, \quad 2 + 3 \times 2^{-51}, \quad \dots, \quad 4.$$

In general, the interval  $[2^j, 2^{j+1}]$  is represented by the numbers for  $[1, 2]$  times  $2^j$ .

- For floating point number system, the machine accuracy, denoted by  $\epsilon_{\text{machine}}$ , is defined as: *half the distance between 1 and the next larger floating point number*, i.e.,

$$\epsilon_{\text{machine}} = \frac{1}{2}\beta^{1-p}.$$

We have

$$\forall x \in [\theta, \Theta], \quad \exists x' \in \mathbf{F} \quad \text{s.t.}, \quad |x - x'| \leq \epsilon_{\text{machine}}|x|.$$

In MATLAB,  $\mathbf{eps} = 2\epsilon_{\text{machine}} = 2^{-52}$  in double precision.

- Let  $\text{fl}: \mathbb{R} \rightarrow \mathbf{F}$  denote the function giving the closest floating point approximation. We have

$$\forall x \in [\theta, \Theta], \quad \exists \epsilon \in \mathbb{R} \quad \text{s.t.}, \quad |\epsilon| \leq \epsilon_{\text{machine}} \quad \text{and} \quad \text{fl}(x) = x(1 + \epsilon).$$

### 1.3. Floating point arithmetic

- $*$   $(+, -, \times, \div)$  in  $\mathbb{R}$ ;  $\odot$   $(\oplus, \ominus, \otimes, \oslash)$  in  $\mathbf{F}$ ;  $x \odot y = \text{fl}(x*y)$ .

#### Fundamental Axiom of Floating Point Arithmetic

For all  $x, y \in \mathbf{F}$ , there exists  $\epsilon$  with  $|\epsilon| \leq \epsilon_{\text{machine}}$  such that

$$x \odot y = (x * y)(1 + \epsilon).$$

### 1.4. Programming exercise

- TreBau Exercise 13.3 (Horner's rule for polynomial evaluation).

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0.$$

---

**Algorithm** Horner's rule for  $p(x) = \sum_{i=0}^n a_i x^i$ .

---

$p = a_n$

**for**  $i = n - 1 : -1 : 0$

$p = xp + a_i;$

**end**

---

## 2. Algorithm

- Consider a *problem*  $f : \mathbb{X} \rightarrow \mathbb{Y}$ . An *algorithm* for the problem  $f$  can be viewed as a map  $\tilde{f} : \mathbb{X} \rightarrow \mathbb{Y}$ .
- More precisely, assume that a problem  $f$ , a computer with floating point system, and a program for solving the problem are fixed:
  - (1) given  $x \in \mathbb{X}$ , let  $\text{fl}(x)$  be the corresponding floating point representation;
  - (2) input  $\text{fl}(x)$  to the program and run it in the computer;
  - (3) the output (computed result) of the program belongs to  $\mathbb{Y}$  and is called  $\tilde{f}(x)$ .
- A problem may have different algorithms (due to different programs). For example: the problem of sum of three numbers:  $a + b + c$ . Programs:  $(a + b) + c$ ,  $a + (b + c)$ , and  $(a + c) + b$ .
- What can happen for an ill-conditioned problem? Since  $x$  is perturbed to  $\text{fl}(x)$ , then  $\|\tilde{f}(x) - f(x)\|$  maybe large.

## 2.1. Accuracy

- An algorithm  $\tilde{f}$  for a problem  $f$  is *accurate* if for **each**  $x \in \mathbb{X}$ ,

$$\frac{\|\tilde{f}(x) - f(x)\|}{\|f(x)\|} = \mathcal{O}(\epsilon_{\text{machine}}).$$

- The meaning of  $\mathcal{O}(\cdot)$ :  $\phi(t) = \mathcal{O}(\psi(t))$  means there exists a positive constant  $C$  such that  $|\phi(t)| \leq C\psi(t)$  for all  $t$  sufficiently close to an understood limit (e.g.,  $t \rightarrow 0$  or  $t \rightarrow \infty$ ).

## 2.2. Stability

- An algorithm  $\tilde{f}$  for a problem  $f$  is *stable* if for **each**  $x \in \mathbb{X}$ , there exists  $\tilde{x} \in \mathbb{X}$ , such that

$$\frac{\|\tilde{x} - x\|}{\|x\|} = \mathcal{O}(\epsilon_{\text{machine}}), \quad \frac{\|\tilde{f}(x) - f(\tilde{x})\|}{\|f(\tilde{x})\|} = \mathcal{O}(\epsilon_{\text{machine}}).$$

### Remark 1

*A stable algorithm gives nearly the right answer to nearly the right question.*

## 2.3. Backward stability

- An algorithm  $\tilde{f}$  for a problem  $f$  is *backward stable* if for **each**  $x \in \mathbb{X}$ , there exists  $\tilde{x} \in \mathbb{X}$ , such that

$$\frac{\|\tilde{x} - x\|}{\|x\|} = \mathcal{O}(\epsilon_{\text{machine}}), \quad \tilde{f}(x) = f(\tilde{x}).$$

### Remark 2

*A backward stable algorithm gives exactly the right answer to nearly the right question.*

### Remark 3

*Backward stability obviously implies stability.*

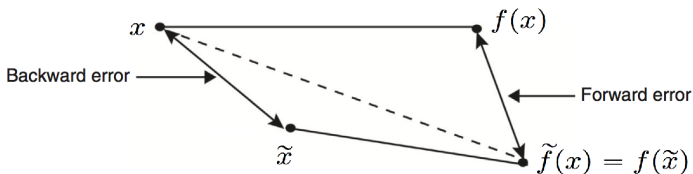
### Remark 4

*Backward stability is both **stronger and simpler** than stability. Many algorithms of NLA are backward stable.*



### 3. Backward error analysis

- The first step is to investigate the conditioning of the problem. The second step is to investigate the backward stability of the corresponding algorithm.
- Forward error  $\lesssim$  Condition number  $\times$  Backward error.



#### Theorem 5 (Attainable accuracy of a backward stable algorithm)

Suppose  $\tilde{f}$  is backward stable for  $f$ . Let  $\kappa(f(x))$  denote the condition number of the problem  $f(x)$ . Then the relative errors satisfy

$$\frac{\|\tilde{f}(x) - f(x)\|}{\|f(x)\|} = \mathcal{O}(\kappa(f(x))\epsilon_{\text{machine}}).$$

## Proof.

By the definition of backward stability, we have there exists  $\tilde{x}$  such that

$$\frac{\|\tilde{x} - x\|}{\|x\|} = \mathcal{O}(\epsilon_{\text{machine}}), \quad \tilde{f}(x) = f(\tilde{x}).$$

By the definition of  $\kappa(f(x))$ ,

$$\kappa(f(x)) = \lim_{\varepsilon \rightarrow 0^+} \sup_{\|\delta x\| \leq \varepsilon} \left( \frac{\|\delta f\|}{\|f(x)\|} / \frac{\|\delta x\|}{\|x\|} \right),$$

we have

$$\frac{\|\tilde{f}(x) - f(x)\|}{\|f(x)\|} \leq (\kappa(f(x)) + o(1)) \frac{\|\tilde{x} - x\|}{\|x\|},$$

where  $o(1)$  denotes a quantity that converges to zero as  $\epsilon_{\text{machine}} \rightarrow 0$ . Then the statement follows. □

## 4. Examples

- **Floating point arithmetic:** The floating point operations  $\oplus, \ominus, \otimes, \odiv$  are all backward stable. Let  $x_1, x_2 \in \mathbb{R}$ . Consider the problem  $f(x_1, x_2) = x_1 * x_2$  (here  $*$  means  $+$  or  $-$ , the cases  $\times$  and  $\div$  are left as [exercises](#)) and the corresponding algorithm  $\tilde{f}(x_1, x_2) = \text{fl}(x_1) \circledast \text{fl}(x_2)$ . There exist  $|\epsilon_1|, |\epsilon_2|, |\epsilon_3| \leq \epsilon_{\text{machine}}$  and  $|\epsilon_4|, |\epsilon_5| \leq 2\epsilon_{\text{machine}} + \mathcal{O}(\epsilon_{\text{machine}}^2)$  such that

$$\begin{aligned}\tilde{f}(x_1, x_2) &= \text{fl}(x_1) \circledast \text{fl}(x_2) \\ &= ([x_1(1 + \epsilon_1)] * [x_2(1 + \epsilon_2)])(1 + \epsilon_3) \\ &= [x_1(1 + \epsilon_1)(1 + \epsilon_3)] * [x_2(1 + \epsilon_2)(1 + \epsilon_3)] \\ &= [x_1(1 + \epsilon_4)] * [x_2(1 + \epsilon_5)] \\ &= \tilde{x}_1 * \tilde{x}_2 = f(\tilde{x}_1, \tilde{x}_2).\end{aligned}$$

Backward stability follows from

$$\frac{|\tilde{x}_1 - x_1|}{|x_1|} = \mathcal{O}(\epsilon_{\text{machine}}), \quad \frac{|\tilde{x}_2 - x_2|}{|x_2|} = \mathcal{O}(\epsilon_{\text{machine}}).$$

- **Inner product**

$\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$ ,  $\alpha = \mathbf{x}^\top \mathbf{y}$ .  $\tilde{\alpha}$  by  $\otimes$  and  $\oplus$ . Backward stable.

- **Outer product**

$\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$ ,  $\mathbf{A} = \mathbf{x}\mathbf{y}^\top$ .  $\tilde{\mathbf{A}}$  by  $\otimes$ . Stable but not backward stable.

Explanation: the matrix  $\tilde{\mathbf{A}}$  will be most unlikely to have rank exactly 1, i.e., cannot be written as  $(\mathbf{x} + \delta\mathbf{x})(\mathbf{y} + \delta\mathbf{y})^*$ .

As a rule, for problems where the dimension of the space  $\mathbb{Y}$  is greater than that of the space  $\mathbb{X}$ , backward stability is rare.

- **Compute**  $f(x) = x + 1$

By  $\oplus$ ,  $\tilde{f}(x) = \text{fl}(x) \oplus 1$ . Stable but not backward stable.

We have

$$\begin{aligned}\tilde{f}(x) &= \text{fl}(x) \oplus 1 = (x(1 + \epsilon_1) + 1)(1 + \epsilon_2) \\ &= x(1 + \epsilon_1 + \epsilon_2 + \epsilon_1\epsilon_2) + \epsilon_2 + 1.\end{aligned}$$

Obviously, for  $x \approx 0$  (i.e.,  $x$  is sufficiently small),

$$\frac{|x(1 + \epsilon_1 + \epsilon_2 + \epsilon_1\epsilon_2) + \epsilon_2 - x|}{|x|} \neq \mathcal{O}(\epsilon_{\text{machine}}).$$

Explanation: For  $x \approx 0$ ,  $\oplus$  introduces absolute errors of size  $\mathcal{O}(\epsilon_{\text{machine}})$ , which cannot be interpreted as caused by small relative perturbations in  $x$ . Therefore, not backward stable.

To show stability, for all  $x$ , let  $\tilde{x} = x(1 + \epsilon_1)$ . Note that

$$\frac{|\tilde{f}(x) - f(\tilde{x})|}{|f(\tilde{x})|} = \frac{|\epsilon_2(x(1 + \epsilon_1) + 1)|}{|x(1 + \epsilon_1) + 1|} = |\epsilon_2| = \mathcal{O}(\epsilon_{\text{machine}}).$$

Then stability follows.

**Comparison:** Let  $x, y \in \mathbb{R}$ . Consider  $f(x, y) = x + y$  and the corresponding backward stable algorithm  $\tilde{f}(x, y) = \text{fl}(x) \oplus \text{fl}(y)$ .

## 4.1. Unitary matrix multiplication

- In the rest of this lecture, for simplicity, we always assume that the given data are floating point numbers already if not explicitly stated.

### Theorem 6

*Left and/or right unitary matrix multiplications are backward stable in the sense: Let  $\mathbf{Q}$  be a unitary matrix. The computed quantity  $\tilde{\mathbf{B}}$  for  $\mathbf{B} = \mathbf{Q}\mathbf{A}$  or  $\mathbf{B} = \mathbf{A}\mathbf{Q}$  satisfies*

$$\tilde{\mathbf{B}} = \mathbf{Q}(\mathbf{A} + \delta\mathbf{A}), \quad \text{or} \quad \tilde{\mathbf{B}} = (\mathbf{A} + \delta\mathbf{A})\mathbf{Q}, \quad \frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|} = \mathcal{O}(\epsilon_{\text{machine}}).$$

### Sketch of the proof.

*We only prove the real case. The complex case is similar. Consider the algorithm for the inner product  $\mathbf{q}^\top \mathbf{a}$ , then matrix-vector product  $\mathbf{Q}\mathbf{a}$ , and then matrix-matrix product  $\mathbf{Q}\mathbf{A}$ . □*

## TreBau Exercise 16.1

- Let unitary matrices  $\mathbf{Q}_k \in \mathbb{C}^{m \times m}$  be fixed and consider the problem of computing, for  $\mathbf{A} \in \mathbb{C}^{m \times n}$ , the product  $\mathbf{B} = \mathbf{Q}_k \mathbf{Q}_{k-1} \cdots \mathbf{Q}_1 \mathbf{A}$ . Let the computation be carried out from right to left by straightforward floating point operations.
  - Show that this algorithm is backward stable. (Here  $\mathbf{A}$  is thought of as data that can be perturbed; the matrices  $\mathbf{Q}_j$  are fixed and not to be perturbed.) That is the computed  $\tilde{\mathbf{B}}$  satisfies

$$\tilde{\mathbf{B}} = \mathbf{Q}_k \mathbf{Q}_{k-1} \cdots \mathbf{Q}_1 (\mathbf{A} + \delta \mathbf{A}), \quad \frac{\|\delta \mathbf{A}\|}{\|\mathbf{A}\|} = \mathcal{O}(\epsilon_{\text{machine}}).$$

- Give an example to show that this result no longer holds if the unitary matrices  $\mathbf{Q}_j$  are replaced by arbitrary matrices  $\mathbf{X}_j \in \mathbb{C}^{m \times m}$ . Hint: consider a singular matrix  $\mathbf{A}$  and nonsingular matrices  $\mathbf{X}_j$ .

## 4.2. Backward stability of back substitution

- The solution of the nonsingular upper-triangular system

$$\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1m} \\ & r_{22} & \ddots & \vdots \\ & & \ddots & \vdots \\ & & & r_{mm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix},$$

can be obtained by the following back substitution algorithm

---

**Algorithm:** Back substitution

---

$$x_m = b_m / r_{mm}$$

$$x_{m-1} = (b_{m-1} - x_m r_{m-1,m}) / r_{m-1,m-1}$$

$$x_{m-2} = (b_{m-2} - x_{m-1} r_{m-2,m-1} - x_m r_{m-2,m}) / r_{m-2,m-2}$$

$$\vdots$$

$$x_j = (b_j - \sum_{k=j+1}^m x_k r_{jk}) / r_{jj}$$

---



## Theorem 7

*Back substitution is backward stable in the sense that the computed solution  $\tilde{\mathbf{x}} \in \mathbb{C}^m$  satisfies*

$$(\mathbf{R} + \delta\mathbf{R})\tilde{\mathbf{x}} = \mathbf{b},$$

*for some upper-triangular  $\delta\mathbf{R} \in \mathbb{C}^{m \times m}$  with*

$$\frac{\|\delta\mathbf{R}\|}{\|\mathbf{R}\|} = \mathcal{O}(\epsilon_{\text{machine}}).$$

*Specifically, for each  $i, j$ ,*

$$\frac{|\delta r_{ij}|}{|r_{ij}|} \leq m\epsilon_{\text{machine}} + \mathcal{O}(\epsilon_{\text{machine}}^2).$$

- Our task is to express every floating point error as a perturbation of the input.

(i) The case  $m = 1$ :

$$\tilde{x}_1 = b_1 \oslash r_{11} = \frac{b_1(1 + \epsilon_1)}{r_{11}}, \quad |\epsilon_1| \leq \epsilon_{\text{machine}}$$

Set  $1 + \epsilon'_1 = 1/(1 + \epsilon_1)$ . We have

$$\epsilon'_1 = -\frac{\epsilon_1}{1 + \epsilon_1} \Rightarrow \tilde{x}_1 = \frac{b_1}{r_{11}(1 + \epsilon'_1)}, \quad |\epsilon'_1| \leq \epsilon_{\text{machine}} + \mathcal{O}(\epsilon_{\text{machine}}^2).$$

Therefore

$$(r_{11} + \delta r_{11})\tilde{x}_1 = b_1; \quad \delta r_{11} = \epsilon'_1 r_{11}; \quad \frac{|\delta r_{11}|}{|r_{11}|} \leq \epsilon_{\text{machine}} + \mathcal{O}(\epsilon_{\text{machine}}^2).$$

(ii) The case  $m = 2$ . The first step is the same as in  $m = 1$  case,

$$\tilde{x}_2 = b_2 \oslash r_{22} = \frac{b_2}{r_{22}(1 + \epsilon_1)}, \quad |\epsilon_1| \leq \epsilon_{\text{machine}} + \mathcal{O}(\epsilon_{\text{machine}}^2).$$

The second step: there exist  $|\epsilon_2|, |\epsilon_3|, |\epsilon_4| \leq \epsilon_{\text{machine}}$ ,

$$\begin{aligned}
\tilde{x}_1 &= (b_1 \ominus (\tilde{x}_2 \otimes r_{12})) \oslash r_{11} = (b_1 \ominus \tilde{x}_2 r_{12}(1 + \epsilon_2)) \oslash r_{11} \\
&= (b_1 - \tilde{x}_2 r_{12}(1 + \epsilon_2))(1 + \epsilon_3) \oslash r_{11} \\
&= \frac{(b_1 - \tilde{x}_2 r_{12}(1 + \epsilon_2))(1 + \epsilon_3)}{r_{11}}(1 + \epsilon_4).
\end{aligned}$$

Shift  $\epsilon_3$  and  $\epsilon_4$  to the denominator

$$\tilde{x}_1 = \frac{b_1 - \tilde{x}_2 r_{12}(1 + \epsilon_2)}{r_{11}(1 + \epsilon'_3)(1 + \epsilon'_4)},$$

or equivalently,

$$\tilde{x}_1 = \frac{b_1 - \tilde{x}_2 r_{12}(1 + \epsilon_2)}{r_{11}(1 + 2\epsilon_5)}, \quad |\epsilon'_3|, |\epsilon'_4|, |\epsilon_5| \leq \epsilon_{\text{machine}} + \mathcal{O}(\epsilon_{\text{machine}}^2).$$

Obviously,  $\tilde{x}_1$  is exactly correct if  $r_{22}$ ,  $r_{12}$  and  $r_{11}$  perturbed by factors  $(1 + \epsilon_1)$ ,  $(1 + \epsilon_2)$  and  $(1 + 2\epsilon_5)$ , respectively. Thus,

$$(\mathbf{R} + \delta\mathbf{R})\tilde{\mathbf{x}} = \mathbf{b},$$

where the entries  $\delta r_{ij}$  of  $\delta \mathbf{R}$  satisfy

$$\begin{bmatrix} \frac{|\delta r_{11}|}{|r_{11}|} & \frac{|\delta r_{12}|}{|r_{12}|} \\ \frac{|\delta r_{22}|}{|r_{22}|} & \end{bmatrix} = \begin{bmatrix} 2|\epsilon_5| & |\epsilon_2| \\ & |\epsilon_1| \end{bmatrix} \leq \begin{bmatrix} 2 & 1 \\ & 1 \end{bmatrix} \epsilon_{\text{machine}} + \mathcal{O}(\epsilon_{\text{machine}}^2).$$

The last formula guarantees  $\|\delta \mathbf{R}\|/\|\mathbf{R}\| = \mathcal{O}(\epsilon_{\text{machine}})$  in any norm.

(iii) The case  $m = 3$ . The first two steps are the same as before:

$$\tilde{x}_3 = b_3 \oplus r_{33} = \frac{b_3}{r_{33}(1 + \epsilon_1)},$$

$$\tilde{x}_2 = (b_2 \ominus (\tilde{x}_3 \otimes r_{23})) \oplus r_{22} = \frac{b_2 - \tilde{x}_3 r_{23}(1 + \epsilon_2)}{r_{22}(1 + 2\epsilon_3)},$$

where

$$\begin{bmatrix} 2|\epsilon_3| & |\epsilon_2| \\ & |\epsilon_1| \end{bmatrix} \leq \begin{bmatrix} 2 & 1 \\ & 1 \end{bmatrix} \epsilon_1 + \mathcal{O}(\epsilon_{\text{machine}}^2)$$

The third step:

$$\begin{aligned}
 \tilde{x}_1 &= [(b_1 \ominus (\tilde{x}_2 \otimes r_{12})) \ominus (\tilde{x}_3 \otimes r_{13})] \odot r_{11} \\
 &= [(b_1 - \tilde{x}_2 r_{12}(1 + \epsilon_4))(1 + \epsilon_6) - \tilde{x}_3 r_{13}(1 + \epsilon_5)](1 + \epsilon_7) \odot r_{11} \\
 &= \frac{[(b_1 - \tilde{x}_2 r_{12}(1 + \epsilon_4))(1 + \epsilon_6) - \tilde{x}_3 r_{13}(1 + \epsilon_5)](1 + \epsilon_7)}{r_{11}(1 + \epsilon'_8)} \\
 &= \frac{b_1 - \tilde{x}_2 r_{12}(1 + \epsilon_4) - \tilde{x}_3 r_{13}(1 + \epsilon_5)(1 + \epsilon'_6)}{r_{11}(1 + \epsilon'_6)(1 + \epsilon'_7)(1 + \epsilon'_8)},
 \end{aligned}$$

$r_{13}$  has two perturbations of size at most  $\epsilon_{\text{machine}}$ ,  $r_{11}$  has three. Then we have  $(\mathbf{R} + \delta\mathbf{R})\tilde{\mathbf{x}} = \mathbf{b}$  with the entries  $\delta r_{ij}$  satisfying

$$\begin{bmatrix} \frac{|\delta r_{11}|}{|r_{11}|} & \frac{|\delta r_{12}|}{|r_{12}|} & \frac{|\delta r_{13}|}{|r_{13}|} \\ & \frac{|\delta r_{22}|}{|r_{22}|} & \frac{|\delta r_{23}|}{|r_{23}|} \\ & & \frac{|\delta r_{33}|}{|r_{33}|} \end{bmatrix} \leq \begin{bmatrix} 3 & 1 & 2 \\ & 2 & 1 \\ & & 1 \end{bmatrix} \epsilon_{\text{machine}} + \mathcal{O}(\epsilon_{\text{machine}}^2).$$

(iv) General  $m$ : Higher-dimensional cases are similar. For example,  $5 \times 5$  case:

$$\frac{|\delta \mathbf{R}|}{|\mathbf{R}|} \leq \begin{bmatrix} 5 & 1 & 2 & 3 & 4 \\ & 4 & 1 & 2 & 3 \\ & & 3 & 1 & 2 \\ & & & 2 & 1 \\ & & & & 1 \end{bmatrix} \epsilon_{\text{machine}} + \mathcal{O}(\epsilon_{\text{machine}}^2).$$

The entries of the matrix in this formula are obtained from three components. The multiplications  $\tilde{x}_k r_{jk}$  introduce  $\epsilon_{\text{machine}}$  perturbations in the pattern

$$\otimes : \tilde{x}_k r_{jk} \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ & 0 & 1 & 1 & 1 \\ & & 0 & 1 & 1 \\ & & & 0 & 1 \\ & & & & 0 \end{bmatrix}. \quad (\text{inner level})$$

The division by  $r_{kk}$  introduce perturbations in the pattern

$$\oplus : \text{divisions by } r_{kk} \quad \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}. \quad (\text{outer level})$$

Finally, the subtractions also occur in the pattern for  $\otimes$ , and, due to the decision to compute from left to right, each one introduces a perturbation on the diagonal and at each position to the right. This adds up to the pattern

$$\ominus : \quad \begin{bmatrix} 4 & 0 & 1 & 2 & 3 \\ & 3 & 0 & 1 & 2 \\ & & 2 & 0 & 1 \\ & & & 1 & 0 \\ & & & & 0 \end{bmatrix}.$$

## Remark 8

*Perturbations of order  $\epsilon_{\text{machine}}$  are composed additively and moved freely between numerators and denominators since the difference is of order  $\epsilon_{\text{machine}}^2$ .*

## Remark 9

*More than one error bound can be derived for a given algorithm. In the present case, we could have perturbed  $b_j$  as well as  $r_{ij}$ , avoiding the need for the trickery represented pattern for  $\ominus$ . On the other hand, a final result in which only  $\mathbf{R}$  is perturbed is appealing clean.*

## Remark 10

*We have done **componentwise** backward error bound. If  $r_{ij} = 0$ , this entry undergoes no perturbation at all:  $\delta\mathbf{R}$  has the same sparsity pattern as  $\mathbf{R}$ .*