STAT 309: MATHEMATICAL COMPUTATIONS I
FALL 2015
LECTURE 14

1. another look at Cholesky

- instead of considering an elementwise algorithm, we can also derive a vectorized version
- this is analogous to our discussions of QR and LU
- let $F = [\mathbf{f}_1, \dots, \mathbf{f}_n]$ where $\mathbf{f}_i$ is the ith column of the lower-triangular matrix F so

$$A = FF^\mathsf{T} = \mathbf{f}_1\mathbf{f}_1^\mathsf{T} + \cdots + \mathbf{f}_n\mathbf{f}_n^\mathsf{T}$$

- we start by observing that

$$\mathbf{f}_1 = \frac{1}{\sqrt{a_{11}}}\mathbf{a}_1$$

  where $\mathbf{a}_i$ is the ith column of A
- then we set $A^{(1)} = A$ and compute

$$A^{(2)} = A^{(1)} - \mathbf{f}_1\mathbf{f}_1^\mathsf{T} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & A_2 & \\ 0 & & & \end{bmatrix}$$

- note that

$$A^{(1)} = B \begin{bmatrix} 1 & 0 \\ 0 & A_2 \end{bmatrix} B^\mathsf{T}$$

  where B is the identity matrix with its first column replaced by $\mathbf{f}_1$

$$B = [\mathbf{f}_1, \mathbf{e}_2, \dots, \mathbf{e}_n] = \begin{bmatrix} f_{11} & & & \\ f_{21} & 1 & & \\ \vdots & & \ddots & \\ f_{n1} & & & 1 \end{bmatrix}$$

- writing $C = B^{-1}$, we see that A_2 is positive definite since

$$\begin{bmatrix} 1 & 0 \\ 0 & A_2 \end{bmatrix} = CAC^\mathsf{T}$$

  is positive definite:

$$\mathbf{x}^\mathsf{T} A_2 \mathbf{x} = \begin{bmatrix} 0 \\ \mathbf{x} \end{bmatrix}^\mathsf{T} \begin{bmatrix} 1 & 0 \\ 0 & A_2 \end{bmatrix} \begin{bmatrix} 0 \\ \mathbf{x} \end{bmatrix} = (C^\mathsf{T}\mathbf{y})^\mathsf{T} A (C^\mathsf{T}\mathbf{y}) > 0$$

  for all $\mathbf{x} \neq \mathbf{0}$ (or if you know Sylvester law of inertia, you can apply it to deduce the same thing)
- so we may repeat the process on A_2

Date: November 17, 2015, version 1.1. Comments, bug reports: lekheng@galton.uchicago.edu.

- we partition the matrix A_2 into columns, writing $A_2 = \begin{bmatrix} \mathbf{a}_2^{(2)} & \mathbf{a}_3^{(2)} & \cdots & \mathbf{a}_n^{(2)} \end{bmatrix}$ and then compute

$$\mathbf{f}_2 = \frac{1}{\sqrt{a_{22}^{(2)}}} \begin{bmatrix} 0 \\ \mathbf{a}_2^{(2)} \end{bmatrix}$$

- we then compute

$$A^{(3)} = A^{(2)} - \mathbf{f}_2 \mathbf{f}_2^{\mathsf{T}} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & A_3 & \\ 0 & 0 & & \end{bmatrix}$$

  and so on
- note that

$$a_{kk} = f_{k1}^2 + f_{k2}^2 + \cdots + f_{kk}^2$$

  which implies that

$$|f_{ki}| \leq \sqrt{|a_{kk}|}$$

- in other words, the elements of F are bounded
- we also have the relationship

$$\det A = \det F \det F^{\mathsf{T}} = (\det F)^2 = f_{11}^2 f_{22}^2 \cdots f_{nn}^2$$

- is the Cholesky decompositon unique?
- employing a similar approach to the one used to prove the uniquess of the LU factorization, we assume that A has two Cholesky factorizations

$$A = F_1 F_1^{\mathsf{T}} = F_2 F_2^{\mathsf{T}}$$

- then

$$F_2^{-1} F_1 = F_2^{\mathsf{T}} F_1^{-\mathsf{T}}$$

  but since F_1 and F_2 are lower triangular, both matrices must be diagonal
- let

$$F_2^{-1} F_1 = D = F_2^{\mathsf{T}} F_1^{-\mathsf{T}}$$

- so $F_1 = F_2 D$ and thus $F_1^{\mathsf{T}} = D F_2^{\mathsf{T}}$ and we get $D^{-1} = F_2^{\mathsf{T}} F_1^{-\mathsf{T}}$
- in other words, $D^{-1} = D$ or $D^2 = I$
- hence D must have diagonal elements equal to ± 1
- since we require that the diagonal elements be positive, it follows that the factorization is unique
- in computing the Cholesky factorization, no row interchanges are necessary because A is positive definite, so the number of operations required to compute F is approximately $n^3/3$
- a simple variant of the algorithm Cholesky factorization yields the LDL^{T} factorization

$$A = LDL^{\mathsf{T}}$$

  where L is a unit lower triangular matrix, and D is a diagonal matrix with positive diagonal elements
- the algorithm is sometimes called the *square-root-free Cholesky factorization* since unlike in the usual Cholesky factorization, it does not require taking square roots (which can be expensive, most computer hardware and software use Newton-Raphson method to extract square roots)
- the LDL^{T} and Cholesky factorizations are related by

$$F = LD^{1/2}$$

2. more on condition numbers

- the condition number of an instance of a problem is the reciprocal of the normalized distance to the nearest ill-posed instance
- for example, if the problem is matrix inversion, i.e., $f : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$, $f(X) = X^{-1}$, then the condition number of a problem instance $A \in \mathbb{R}^{n\times n}$ is $\kappa(A) = \|A\|\|A^{-1}\|$
- why? because the set of ill-posed problem is the set of singular matrices $\mathcal{M} = \{X \in \mathbb{R}^{n\times n} : \det(X) = 0\}$ and the distance of any nonsingular A to this set is (exercise for you)

$$\operatorname{dist}(A, \mathcal{M}) := \min_{X\in\mathcal{M}} \|A - X\| = \frac{1}{\|A^{-1}\|}$$

and so the normalized distance is

$$\frac{\operatorname{dist}(A, \mathcal{M})}{\|A\|} = \frac{1}{\|A\|\|A^{-1}\|} = \frac{1}{\kappa(A)},$$

the reciprocal of the usual condition number

- we can also do this for problem of finding pseudoinverse $f : \mathbb{R}^{m\times n} \to \mathbb{R}^{n\times m}$, $f(X) = X^\dagger$
- in this case, the set of ill-posed problem is the set of rank-deficiency matrices $\mathcal{M} = \{A \in \mathbb{R}^{m\times n} : \operatorname{rank}(A) < \min\{m, n\}\}$
- the condition number of a problem instance $A \in \mathbb{R}^{m\times n}$ is

$$\frac{\operatorname{dist}_F(A, \mathcal{M})}{\|A\|_F} = \min_{X\in\mathcal{M}} \|A - X\| = \frac{\sigma_{\min}(A)}{\sigma_{\max}(A)} = \frac{1}{\kappa_F(A)},$$

the reciprocal of the generalized condition number

- there are many others: linear system, least squares, linear programming eigenvalue problems, polynomial eigenvalue problems
- for example, for linear programming, the condition number is given by

$$\frac{1}{\kappa_2(A, \mathbf{b})} = \frac{\operatorname{dist}_2([A, \mathbf{b}], \mathcal{M})}{\|[A, \mathbf{b}]\|_2}$$

where $\mathcal{M}$ = boundary of feasible pairs $(A, \mathbf{b}) \in \mathbb{R}^{m\times(n+1)}$

3. backward stability and numerical stability

- we shall regard our *problem* as a function $f : X \to Y$ that takes input $x \in X$ (elements in the domain of f) to output $y \in Y$ (elements in the codomain of f)
- strictly speaking, this is only correct if we have a well-posed problem, i.e., one with guaranteed existence and uniqueness of solution (every element in the domain gets mapped to exactly one image in the codomain)
- for example, the problem of LU factorization is $f : \mathbb{R}^{n\times n} \to \mathfrak{S}_n \times \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times n}$, $f(A) = (L, U)$ where[1] $A = \Pi^\mathsf{T} LU$
- for example, the problem of solving linear systems is $f : \mathrm{GL}(n) \times \mathbb{R}^n \to \mathbb{R}^n$, $f(A, \mathbf{b}) = A^{-1}\mathbf{b}$
- given $x \in X$, an algorithm for computing $y = f(x)$ is subjected to rounding errors and would instead produces a computed $\widehat{y}$
- the algorithm is said to be *backward stable* if for any $x \in X$, the computed $\widehat{y}$ satisfies

$$\widehat{y} = f(x + \Delta x), \quad |\Delta x| \le \delta|x|$$

for some 'small' δ

- Δx is called the *backward error* while $\widehat{y} - y$ is called the *forward error*
- $|\cdot|$ is some measure of the 'size' of x, usually a norm
- see Figure 1 for a pictorial depiction of backward stability

[1] $\mathfrak{S}_n$ is the symmetric group, i.e., set of all permutations of n objects

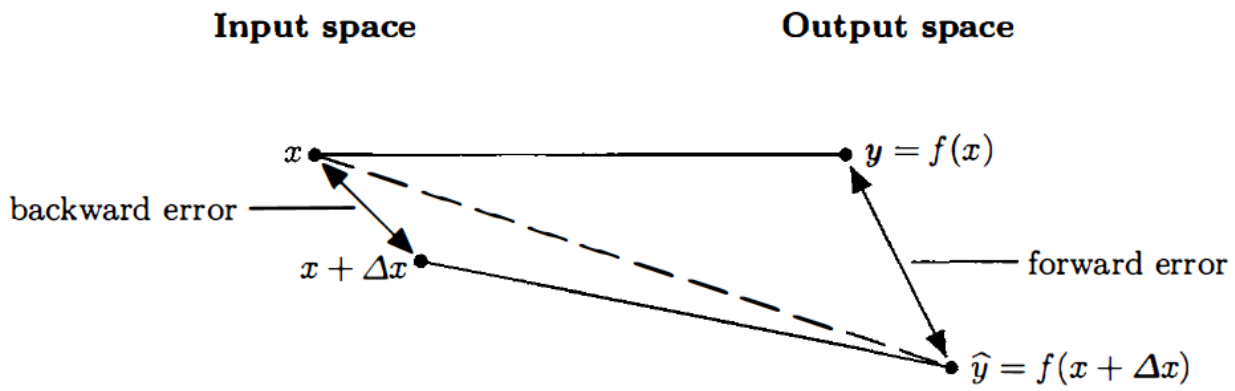

FIGURE 1. solid line = exact; dotted-line = computed; taken from N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd Ed, SIAM, 2002

- the notion above of stability above is too restrictive to in most instances
- one reason is that the computed $\widehat{y}$ may not even be in the range of f, i.e., $\widehat{y} \neq f(x + \Delta x)$ for any choice of Δx
- another reason is that even if $\widehat{y} = f(x + \Delta x)$ for some Δx, it may be too difficult to find a reasonable estimate for δ so that $|\Delta x| \leq \delta |x|$
- so we use a more convenient notion called *mixed forward-backward stability* when we talk about numerical stability
- an algorithm is said to be *numerically stable* if for any $x \in X$, the computed $\widehat{y}$ satisfies

$$\widehat{y} + \Delta y = f(x + \Delta x), \quad |\Delta x| \leq \delta |x|, \quad |\Delta y| \leq \epsilon |y| \tag{3.1}$$

  for some 'small' δ and ϵ
- see Figure 2 for a pictorial depiction of numerical stability (= mixed forward-backward stability)
- the way to interpret (3.1) is: "$\widehat{y}$ is almost the right answer for almost the right data"

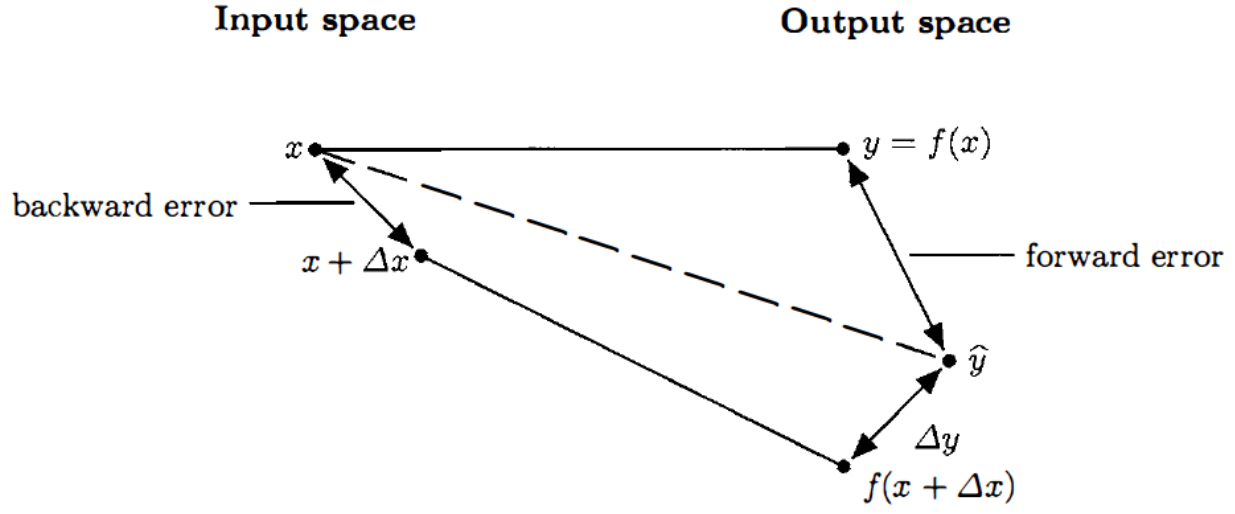

FIGURE 2. solid line = exact; dotted-line = computed; taken from N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd Ed, SIAM, 2002

4. CONDITIONING AND STABILITY

- *conditioning* is a property of a problem whereas *stability* is a property of an algorithm
- the (relative) accuracy of our computed solution to a problem will be affected by both
- a rule-of-thumb is

$$\textit{forward error} \lesssim \textit{condition number} \times \textit{backward error}$$

  where '$\lesssim$' means 'roughly bounded by'
- for example, in Homework **1**, Problem **4**(c), we saw that for solving $A\mathbf{x} = \mathbf{b}$ (with no error in $\mathbf{b}$), the forward error $\|\Delta\mathbf{x}\|/\|\mathbf{x}\|$ is related to the backward error $\|\Delta A\|/\|A\|$ via

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{\kappa(A)\dfrac{\|\Delta A\|}{\|A\|}}{1 - \kappa(A)\dfrac{\|\Delta A\|}{\|A\|}}$$

- this relation is an example of '$\lesssim$', if we use the expansion $x/(1-x) \approx x$, we get a simplification

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \lesssim \kappa(A)\frac{\|\Delta A\|}{\|A\|}$$

- later we will see that if we use GEPP followed by two back substitutions, then the backward error measured in the matrix ∞-norm is

$$\frac{\|\Delta A\|_\infty}{\|A\|_\infty} \leq 2n(n+1)\gamma_n \mathsf{u}$$

  and so the RHS gives us an estimate of how big δ is

5. ERROR ANALYSIS OF SOLVING LINEAR SYSTEMS

- we will consider the case of solving linear system Gaussian elimination and perform a detailed error analysis, illustrating the analysis originally carried out by J. H. Wilkinson
- the process of solving $A\mathbf{x} = \mathbf{b}$ consists of three stages:
  (i) factoring $A = LU$, resulting in an approximate LU decomposition $A + E = \bar{L}\bar{U}$, we assume that partial pivoting is used
  (ii) solving $L\mathbf{y} = \mathbf{b}$, or, numerically, computing $\mathbf{y}$ such that

$$(\bar{L} + \Delta\bar{L})(\mathbf{y} + \Delta\mathbf{y}) = \mathbf{b}$$

  (iii) solving $U\mathbf{x} = \mathbf{y}$, or, numerically, computing $\mathbf{x}$ such that

$$(\bar{U} + \Delta\bar{U})(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{y} + \Delta\mathbf{y}$$

- combining these stages, we see that

$$\begin{aligned}
\mathbf{b} &= (\bar{L} + \Delta\bar{L})(\bar{U} + \Delta\bar{U})(\mathbf{x} + \Delta\mathbf{x}) \\
&= (\bar{L}\bar{U} + \Delta\bar{L}\bar{U} + \bar{L}\Delta\bar{U} + \Delta\bar{L}\Delta\bar{U})(\mathbf{x} + \Delta\mathbf{x}) \\
&= (A + E + \Delta\bar{L}\bar{U} + \bar{L}\Delta\bar{U} + \Delta\bar{L}\Delta\bar{U})(\mathbf{x} + \Delta\mathbf{x}) \\
&= (A + \Delta)(\mathbf{x} + \Delta\mathbf{x})
\end{aligned}$$

  where $\Delta = E + \Delta\bar{L}\bar{U} + \bar{L}\Delta\bar{U} + \Delta\bar{L}\Delta\bar{U}$
- in this analysis, we will view the computed solution $\bar{\mathbf{x}} = \mathbf{x} + \Delta\mathbf{x}$ as the exact solution to the perturbed problem $(A + \Delta)\mathbf{x} = \mathbf{b}$
- this perspective is the idea behind *backward error analysis*, which we will use to determine the size of the perturbation Δ, and, eventually, arrive at a bound for the error in the computed solution $\bar{\mathbf{x}}$

- let $A^{(k)}$ denote the matrix A after $k-1$ steps of Gaussian elimination have been performed *in exact arithmetic*, where a step denotes the process of making all elements below the diagonal within a particular column equal to zero
- then the elements of $A^{(k+1)}$ are given by

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}, \quad m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \tag{6.1}$$

- let $B^{(k)}$ denote the matrix A after $k-1$ steps of Gaussian elimination have been performed *in floating-point arithmetic*
- then the elements of $B^{(k+1)}$ are given by

$$b_{ij}^{(k+1)} = a_{ij}^{(k)} - s_{ik} b_{kj}^{(k)} + \epsilon_{ij}^{(k+1)}, \quad s_{ik} = \mathrm{fl}\left(\frac{b_{ik}^{(k)}}{b_{kk}^{(k)}}\right) \tag{6.2}$$

- for $j \geq i$, we have

$$\begin{aligned} b_{ij}^{(2)} &= b_{ij}^{(1)} - s_{i1} b_{1j}^{(1)} + \epsilon_{ij}^{(2)} \\ b_{ij}^{(3)} &= b_{ij}^{(2)} - s_{i2} b_{2j}^{(2)} + \epsilon_{ij}^{(3)} \\ &\vdots \\ b_{ij}^{(i)} &= b_{ij}^{(i-1)} - s_{i,i-1} b_{i-1,j}^{(i-1)} + \epsilon_{ij}^{(i)} \end{aligned}$$

- combining these equations yields

$$\sum_{k=2}^{i} b_{ij}^{(k)} = \sum_{k=1}^{i-1} b_{ij}^{(k)} - \sum_{k=1}^{i-1} s_{ik} b_{kj}^{(k)} + \sum_{k=2}^{i} \epsilon_{ij}^{(k)}$$

- canceling terms, we obtain

$$b_{ij}^{(1)} = b_{ij}^{(i)} + \sum_{k=1}^{i-1} s_{ik} b_{kj}^{(k)} + e_{ij}, \quad j \geq i \tag{6.3}$$

  where $e_{ij} := -\sum_{k=2}^{i} \epsilon_{ij}^{(k)}$
- for $i > j$,

$$\begin{aligned} b_{ij}^{(2)} &= b_{ij}^{(1)} - s_{i1} b_{1j}^{(1)} + \epsilon_{ij}^{(2)} \\ &\vdots \\ b_{ij}^{(j)} &= b_{ij}^{(j-1)} - s_{i,j-1} b_{j-1,j}^{(j-1)} + \epsilon_{ij}^{(j)} \end{aligned}$$

  where $s_{ij} = \mathrm{fl}(b_{ij}^{(j)}/b_{jj}^{(j)}) = b_{ij}^{(j)}/b_{jj}^{(j)} + \eta_{ij}$, and therefore

$$\begin{aligned} 0 &= b_{ij}^{(j)} - s_{ij} b_{jj}^{(j)} + b_{jj}^{(j)} \eta_{ij} \\ &= b_{ij}^{(j)} - s_{ij} b_{jj}^{(j)} + \epsilon_{ij}^{(j+1)} \\ &= b_{ij}^{(1)} - \sum_{k=1}^{j} s_{ik} b_{kj}^{(k)} + e_{ij} \end{aligned} \tag{6.4}$$

- from (6.3) and (6.4), we obtain

$$\bar{L}\bar{U} = \begin{bmatrix} 1 & & & \\ s_{21} & 1 & & \\ \vdots & & \ddots & \\ s_{n1} & \cdots & \cdots & 1 \end{bmatrix} \begin{bmatrix} b_{11}^{(1)} & b_{12}^{(1)} & \cdots & b_{1n}^{(1)} \\ & \ddots & & \vdots \\ & & \ddots & \vdots \\ & & & b_{nn}^{(n)} \end{bmatrix} = A + E$$

  where

$$s_{ik} = \mathrm{fl}\left(\frac{b_{ik}^{(k)}}{b_{kk}^{(k)}}\right) = \frac{b_{ik}^{(k)}}{b_{kk}^{(k)}}(1+\eta_{ik}), \quad |\eta_{ik}| \le \mathsf{u}$$

- then

$$\mathrm{fl}(s_{ik}b_{kj}^{(k)}) = s_{ik}b_{kj}^{(k)}(1+\theta_{ij}^{(k)}), \quad |\theta_{ij}^{(k)}| \le \mathsf{u}$$

  and so

$$\begin{aligned} b_{ij}^{(k+1)} &= \mathrm{fl}(b_{ij}^{(k)} - s_{ik}b_{kj}^{(k)}(1+\theta_{ij}^{(k)})) \\ &= (b_{ij}^{(k)} - s_{ik}b_{kj}^{(k)}(1+\theta_{ij}^{(k)}))(1+\varphi_{ij}^{(k)}), \quad |\varphi_{ij}^{(k)}| \le \mathsf{u} \end{aligned}$$

- after some manipulations, we obtain

$$\epsilon_{ij}^{(k+1)} = b_{ij}^{(k+1)}\left(\frac{\varphi_{ij}^{(k)}}{1+\varphi_{ij}^{(k)}}\right) - s_{ik}b_{kj}^{(k)}\theta_{ij}^{(k)}$$

- with partial pivoting, $|s_{ik}| \le 1$, provided that $|\,\mathrm{fl}(a/b)| \le 1$ whenever $|a| \le |b|$
- in most modern implementations of floating-point arithmetic, this is in fact the case
- it follows that

$$|\epsilon_{ij}^{(k+1)}| \le |b_{ij}^{(k+1)}|\frac{\mathsf{u}}{1-\mathsf{u}} + 1 \cdot |b_{ij}^{(k)}|\mathsf{u}$$

- how large can the elements of $B^{(k)}$ be?
- in the following we set

$$a := \|A\|_{H,\infty} = \max_{i,j}|a_{ij}|$$

- returning to exact arithmetic, since $|a_{ij}| \le a$ and from (6.1), we obtain

$$\begin{aligned} |a_{ij}^{(2)}| &\le |a_{ij}^{(1)}| + |a_{kj}^{(1)}| \le 2a \\ |a_{ij}^{(3)}| &\le 4a \\ &\vdots \\ |a_{ij}^{(n)}| &= |a_{nn}^{(n)}| \le 2^{n-1}a \end{aligned}$$

- we can show that a similar result holds in floating-point arithmetic:

$$|b_{ij}^{(k)}| \le 2^{k-1}a + O(\mathsf{u})$$

- this upper bound is achievable (by Hadamard matrices), but in practice it rarely occurs
- the factor

$$\gamma_n := \frac{\max_{i,j,k} a_{ij}^{(k)}}{\max_{i,j} a_{ij}}$$

  is called the *growth factor*
- for partial pivoting,

$$\gamma_n^{\mathrm{GEPP}} = 2^{n-1}$$

- we concluded that when partial pivoting is used, the entries of $\bar{U}$ were bounded:

$$|b_{ij}^{(k)}| \leq 2^{k-1}a + O(\mathsf{u})$$

  where k is the number of steps of Gaussian elimination that effect the (i,j)th element and a is an upper bound on the elements of A
- Wilkinson gave a bound for the *growth factor for complete pivoting* γ_n^{GECP}
- until 1990, it was conjectured that $\gamma_k^{\text{GECP}} \leq k$
- it was shown to be true for $n \leq 5$, but there have been examples constructed for $n > 5$ where $\gamma_n^{\text{GECP}} \geq n$
- in any event, we have the following bound for the entries of E:

$$|E| \leq 2\mathsf{u}\gamma_n a \begin{bmatrix} 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 1 & \cdots & \cdots & \cdots & \cdots & 1 \\ 1 & 2 & \cdots & \cdots & \cdots & 2 \\ \vdots & \vdots & 3 & \cdots & \cdots & 3 \\ \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ 1 & 2 & 3 & \cdots & n-1 & n-1 \end{bmatrix} + O(\mathsf{u}^2)$$

7. ERROR ANALYSIS OF BACK SUBSTITUTION

- we now study the process of back substitution, to solve

$$\begin{bmatrix} t_{11} & & 0 \\ \vdots & \ddots & \\ t_{n1} & & t_{nn} \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} h_1 \\ \vdots \\ h_n \end{bmatrix}$$

- using back substitution, we obtain

$$\begin{aligned} u_1 &= \frac{h_1}{t_{11}} \\ &\vdots \\ u_k &= \frac{h_k - t_{k1}u_1 - \cdots - t_{k,k-1}u_{k-1}}{t_{kk}} \end{aligned}$$

  which yields

$$\begin{aligned} \text{fl}(u_k) &= \frac{h_k(1+\epsilon_k)(1+\eta_k) - \sum_{i=1}^{k-1} t_{ki}u_i(1+\xi_{ki})(1+\epsilon_k)(1+\eta_k)}{t_{kk}} \\ &= \frac{h_k - \sum_{i=1}^{k-1} t_{ki}u_i(1+\xi_{ki})}{\dfrac{t_{kk}}{(1+\epsilon_k)(1+\eta_k)}} \end{aligned}$$

  or

$$\sum_{i=1}^{k} u_i t_{ki}(1+\lambda_{ki}) = h_k$$

  which can be rewritten in matrix notation as

$$T\mathbf{u} + \begin{bmatrix} \lambda_{11}t_{11} & & \\ \lambda_{12}t_{12} & \lambda_{22}t_{22} & \\ \vdots & \vdots & \ddots \end{bmatrix} \mathbf{u} = \mathbf{h}$$

- in other words, the computed solution $\mathbf{u}$ is the exact solution to the perturbed problem $(T + \Delta T)\mathbf{u} = \mathbf{h}$, where

$$|\Delta T| \leq \mathsf{u} \begin{bmatrix} |t_{11}| & & & \\ |t_{21}| & 2|t_{22}| & & \\ \vdots & & \ddots & \\ (n-1)|t_{n1}| & \cdots & \cdots & 2|t_{nn}| \end{bmatrix} + O(\mathsf{u}^2)$$

- note that the perturbation ΔT actually depends on $\mathbf{h}$

8. bounding the backward error

- recall that our computed solution $\mathbf{x} + \Delta\mathbf{x}$ solves

$$(A + \Delta A)\bar{\mathbf{x}} = \mathbf{b}$$

  where ΔA is a perturbation that has the form

$$\Delta A = E + \bar{L}\Delta\bar{U} + \Delta\bar{L}\bar{U} + \Delta\bar{L}\Delta\bar{U}$$

- for partial pivoting, $|\bar{l}_{ij}| \leq 1$, and we have the bounds

$$\max_{i,j} |\Delta\bar{L}_{ij}| \leq n\mathsf{u} + O(\mathsf{u}^2),$$
$$\max_{i,j} |\Delta\bar{U}_{ij}| \leq n\mathsf{u}\gamma_n a + O(\mathsf{u}^2)$$

  where $a = \max_{i,j} |a_{ij}|$ and γ_n is the growth factor for partial pivoting
- putting our bounds together, we have

$$\begin{aligned} \max_{i,j} |\Delta A_{ij}| &\leq \max_{i,j} |e_{ij}| + \max_{i,j} |\bar{L}\Delta\bar{U}_{ij}| + \max_{i,j} |\bar{U}\Delta\bar{L}_{ij}| + \max_{i,j} |\Delta\bar{L}\Delta\bar{U}_{ij}| \\ &\leq 2\mathsf{u}\gamma_n an + n^2\gamma_n a\mathsf{u} + n^2\gamma_n a\mathsf{u} + O(\mathsf{u}^2) \end{aligned}$$

  from which it follows that

$$\|\Delta A\|_\infty \leq 2n^2(n+1)\mathsf{u}\gamma_n a + O(\mathsf{u}^2)$$

- we conclude that the method of solving a linear system via Gaussian elimination and back substitution is *backward stable*

9. bounding the forward error

- let $\bar{\mathbf{x}} = \mathbf{x} + \Delta\mathbf{x}$ be the computed solution
- then, from $(A + \Delta A)\bar{\mathbf{x}} = \mathbf{b}$ we obtain

$$\Delta A\bar{\mathbf{x}} = \mathbf{b} - A\bar{\mathbf{x}} = \mathbf{r}$$

  where $\mathbf{r}$ is called the *residual vector*
- from our previous analysis,

$$\frac{\|\mathbf{r}\|_\infty}{\|\bar{\mathbf{x}}\|_\infty} \leq \|\Delta A\|_\infty \leq 2n^2(n+1)\gamma_n a\mathsf{u}$$

- also, recall from Homework **1**, Problem **4**(c) that

$$\frac{\|\Delta\mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty} \leq \frac{\kappa_\infty(A)\dfrac{\|\Delta A\|_\infty}{\|A\|_\infty}}{1 - \kappa_\infty(A)\dfrac{\|\Delta A\|_\infty}{\|A\|_\infty}}$$

- we know that $\|A\|_\infty \leq na$, so
$$\frac{\|\Delta A\|_\infty}{\|A\|_\infty} \leq 2n(n+1)\gamma_n \mathsf{u}$$
- note that if $\kappa(A)$ is large and γ_n is large, our solution can be very inaccurate
- the important factors in the accuracy of the computed solution are:
  - the growth factor γ_n
  - the condition number $\kappa(A)$
  - the unit roundoff u
- in particular, κ must be large with respect to the accuracy in order to be troublesome
- for example, consider the scenario where $\kappa = 10^2$ and $\mathsf{u} = 10^{-3}$, as opposed to the case where $\kappa = 10^2$ and $\mathsf{u} = 10^{-50}$