STAT 309: MATHEMATICAL COMPUTATIONS I
FALL 2019
LECTURE 12

1. POSITIVE DEFINITE MATRICES

- a symmetric matrix $A \in \mathbb{R}^{n\times n}$ is *positive definite* if $\mathbf{x}^{\mathsf{T}} A\mathbf{x} > 0$ for all nonzero $\mathbf{x}$
- a symmetric positive definite matrix has real and positive eigenvalues, and its leading principal submatrices all have positive determinants
- from the definition, it is easy to see that all diagonal elements are positive
- to solve the system $A\mathbf{x} = \mathbf{b}$ where A is symmetric positive definite, we can compute the *Cholesky factorization*
$$A = R^{\mathsf{T}} R$$
where R is upper triangular
- this factorization exists if and only if A is symmetric positive definite
- in fact, attempting to compute the Cholesky factorization of A is an efficient method for checking whether A is symmetric positive definite
- it is important to distinguish the Cholesky factorization from the *square root factorization*
- a square root of a matrix A is defined as a matrix S such that
$$S^2 = SS = A$$
- we often write $A^{-1/2}$ for S
- note that the matrix R in $A = R^{\mathsf{T}} R$ is not the square root of A, since it does not hold that $R^2 = A$ unless A is a diagonal matrix
- a symmetric square root of a symmetric positive definite A can be computed by using the fact that A has an eigendecomposition $A = Q\Lambda Q^{\mathsf{T}}$ where Λ is a diagonal matrix whose diagonal elements are the positive eigenvalues of A and Q is an orthogonal matrix whose columns are the eigenvectors of A
- it follows that
$$A = Q\Lambda Q^{\mathsf{T}} = (Q\Lambda^{1/2} Q^{\mathsf{T}})(Q\Lambda^{1/2} Q^{\mathsf{T}}) = SS$$
and so $S = Q\Lambda^{1/2} Q^{\mathsf{T}}$ is a square root of A, note that S is symmetric

2. CHOLESKY FACTORIZATION

- the Cholesky factorization can be computed directly from the matrix equation $A = R^{\mathsf{T}} R$ where R is upper-triangular, much like how we derived Gram–Schmidt
- while it is conventional to write Cholesky factorization in the form $A = R^{\mathsf{T}} R$, it will be more natural later when we discuss the vectorized version of the algorithm to write $F = R^{\mathsf{T}}$ and $A = FF^{\mathsf{T}}$
- we can derive the algorithm for computing F by examining the matrix equation $A = R^{\mathsf{T}} R = FF^{\mathsf{T}}$ on an element-by-element basis, writing
$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{12} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} f_{11} & & & \\ f_{21} & f_{22} & & \\ \vdots & \vdots & \ddots & \\ f_{n1} & f_{n2} & \cdots & f_{nn} \end{bmatrix} \begin{bmatrix} f_{11} & f_{21} & \cdots & f_{n1} \\ & f_{22} & & f_{n2} \\ & & \ddots & \vdots \\ & & & f_{nn} \end{bmatrix}$$

Date: November 6, 2019, version 1.0. Comments, bug reports: `lekheng@galton.uchicago.edu`.

- from the above matrix multiplication we see that $f_{11}^2 = a_{11}$, from which it follows that
$$f_{11} = \sqrt{a_{11}}$$
- from the relationship $f_{11} f_{i1} = a_{1i}$ and the fact that we already know f_{11}, we obtain
$$f_{i1} = \frac{a_{1i}}{f_{11}}, \quad i = 2, \dots, n$$
- proceeding to the second column of F, we see that $f_{21}^2 + f_{22}^2 = a_{22}$
- since we already know f_{21}, we have
$$f_{22} = \sqrt{a_{22} - f_{21}^2}$$
- if you know the fact that a positive definite matrix must have positive leading principal minors,[1] then you could deduce the term above in the square root is positive by examining the 2×2 principal minor:
$$a_{11} a_{22} - a_{12}^2 > 0$$
and therefore
$$a_{22} > \frac{a_{12}^2}{a_{11}} = f_{21}^2$$
- next, we use the relation $f_{21} f_{i1} + f_{22} f_{i2} = a_{2i}$ to compute
$$f_{i2} = \frac{a_{2i} - f_{21} f_{i1}}{f_{22}}$$
- hence we get
$$\begin{aligned} a_{11} &= f_{11}^2, & \\ a_{i1} &= f_{11} f_{i1}, & i &= 2, \dots, n \\ &\vdots & \\ a_{kk} &= f_{k1}^2 + f_{k2}^2 + \cdots + f_{kk}^2, & \\ a_{ik} &= f_{k1} f_{i1} + \cdots + f_{kk} f_{ik}, & i &= k+1, \dots, n \end{aligned}$$
- the resulting algorithm that runs for $k = 1, \dots, n$ is
$$\begin{aligned} f_{kk} &= \Bigl(a_{kk} - \textstyle\sum_{j=1}^{k-1} f_{kj}^2\Bigr)^{1/2}, & \\ f_{ik} &= \frac{\Bigl(a_{ik} - \sum_{j=1}^{k-1} f_{kj} f_{ij}\Bigr)}{f_{kk}}, & i &= k+1, \dots, n \end{aligned}$$
- you could use induction to show that the term in the square root is always positive but we'll soon see a more elegant vectorized version showing that this algorithm doesn't ever require taking square roots of negative numbers
- this algorithm requires roughly half as many operations as Gaussian elimination

3. another look at Cholesky

- instead of considering an elementwise algorithm, we can also derive a vectorized version
- this is analogous to our discussions of Householder QR and Gaussian elimination for 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
$$A = FF^{\mathsf{T}} = \mathbf{f}_1 \mathbf{f}_1^{\mathsf{T}} + \cdots + \mathbf{f}_n \mathbf{f}_n^{\mathsf{T}}$$

[1] If you don't, see `https://en.wikipedia.org/wiki/Sylvester's_criterion`; now you do.

- 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 since C is lower triangular)
- so we may repeat the process on A_2
- 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
$$A^{(3)} = A^{(2)} - \mathbf{f}_2\mathbf{f}_2^{\mathsf{T}} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & A_3 & \\ 0 & 0 & & \end{bmatrix}$$
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 entries of F are automatically bounded by the (square root of the) diagonal entries of A
- this is why there no need to do any pivoting for Cholesky factorization

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

4. 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 $\bar{\mathbf{x}}$

5. error analysis of Gaussian elimination

- 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

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)}, \quad m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}} \tag{5.1}$$

- 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

$$b_{ij}^{(k+1)} = a_{ij}^{(k)} - s_{ik}b_{kj}^{(k)} + \epsilon_{ij}^{(k+1)}, \quad s_{ik} = \mathrm{fl}\left(\frac{b_{ik}^{(k)}}{b_{kk}^{(k)}}\right) \tag{5.2}$$

- 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

$$b_{ij}^{(1)} = b_{ij}^{(i)} + \sum_{k=1}^{i-1} s_{ik}b_{kj}^{(k)} + e_{ij}, \quad j \geq i \tag{5.3}$$

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

$$\begin{aligned}0 &= b_{ij}^{(j)} - s_{ij}b_{jj}^{(j)} + b_{jj}^{(j)}\eta_{ij}\\ &= b_{ij}^{(j)} - s_{ij}b_{jj}^{(j)} + \epsilon_{ij}^{(j+1)}\\ &= b_{ij}^{(1)} - \sum_{k=1}^{j} s_{ik}b_{kj}^{(k)} + e_{ij}\end{aligned} \tag{5.4}$$

- from (5.3) and (5.4), 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 (5.1), 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,
$$\gamma_n^{\text{GEPP}} = 2^{n-1}$$
- 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*
$$\gamma_n^{\text{GECP}} \leq (2 \cdot 3^{1/2} \cdot \cdots \cdot n^{1/(n-1)} \cdot n)^{1/2}$$
the right-hand side is roughly $cn^{\frac{1}{2}} n^{\frac{1}{4}\log n}$ but it is known that this is not the best possible bound
- until 1990, it was conjectured that $\gamma_n^{\text{GECP}} \leq n$
- 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:
$$|E| \leq 2\mathsf{u}\gamma_n a \begin{bmatrix} 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 1 & \cdots & \cdots & \cdots & \cdots & 1 \\ 1 & 2 & \cdots & \cdots & \cdots & 2 \\ \vdots & \vdots & 3 & \cdots & \cdots & 3 \\ \vdots & \vdots & \vdots & \ddots & \cdots & \vdots \\ 1 & 2 & 3 & \cdots & n-1 & n-1 \end{bmatrix} + O(\mathsf{u}^2)$$

6. 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
$$u_1 = \frac{h_1}{t_{11}}$$
$$\vdots$$
$$u_k = \frac{h_k - t_{k1}u_1 - \cdots - t_{k,k-1}u_{k-1}}{t_{kk}}$$
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 $\mathbf{h}$

7. 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*

8. 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 **2**, Problem **6**(c) that
$$\frac{\|\Delta \mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty} \le \frac{\kappa_\infty(A)\dfrac{\|\Delta A\|_\infty}{\|A\|_\infty}}{1-\kappa_\infty(A)\dfrac{\|\Delta A\|_\infty}{\|A\|_\infty}}$$
- we know that $\|A\|_\infty \le na$, so
$$\frac{\|\Delta A\|_\infty}{\|A\|_\infty} \le 2n(n+1)\gamma_n \mathsf{u}$$
- 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 $\mathsf{u} = 10^{-50}$

9. multiple right-hand sides and inverse

- let $A \in \mathbb{R}^{m\times n}$ and $\mathbf{b}_1, \dots, \mathbf{b}_p \in \mathbb{R}^m$
- suppose we need to solve p linear systems with the same coefficient matrix but different right-hand sides
$$A\mathbf{x}_1 = \mathbf{b}_1, \quad A\mathbf{x}_2 = \mathbf{b}_2, \quad \dots, \quad A\mathbf{x}_p = \mathbf{b}_p \tag{9.1}$$
- this is equivalent to solving the matrix equation
$$AX = B$$
where $X = [\mathbf{x}_1, \dots, \mathbf{x}_p] \in \mathbb{R}^{n\times p}$ and $B = [\mathbf{b}_1, \dots, \mathbf{b}_p] \in \mathbb{R}^{m\times p}$
- for example, this is what we do when we need to compute the inverse of an $n \times n$ nonsingular matrix A:
$$AX = I,$$
which is equivalent to the systems of equations
$$A\mathbf{x}_j = \mathbf{e}_j, \quad j = 1, \dots, n$$
- since only the right-hand side is different in each of these systems, we need only compute the LU factorization (or QR or Cholesky, etc) of A once
- more generally, this is how we should compute $A^{-1}B$ for matrices A and B, we should solve (9.1) instead of finding the explicit inverse A^{-1} and then multiplying it to B (exercise: what if you need AB^{-1}?)
- we didn't say too much about why it's a bad idea to compute the explicit inverse of a matrix, for more information about this topic, see Chapter 14 in: N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd Ed, SIAM, 2002

10. block factorizations and Schur complement

- a surprisingly simple and powerful idea that appeared implicitly several times in our earlier discussions is that of *block elimination* and *block factorization*

- all it involves is to consider a matrix $A \in \mathbb{R}^{n\times n}$ as a 2×2 block matrix
$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$
where $A_{11} \in \mathbb{R}^{p\times p}$, $A_{22} \in \mathbb{R}^{q\times q}$, $A_{12} \in \mathbb{R}^{q\times p}$, $A_{21} \in \mathbb{R}^{p\times q}$ for some p and q where $p+q=n$
- this works for rectangular matrices too but we keep our discussion to square matrices for simplicity
- many of the stuff that we discussed can be carried over to block matrices
- for example, if A_{11} is nonsingular, we could define an $n \times n$ block elimination matrix
$$M_1 = I - U_1 V_1^{\mathsf{T}}$$
where $U_1, V_1 \in \mathbb{R}^{n\times p}$ are
$$U_1 = \begin{bmatrix} 0 \\ A_{21}A_{11}^{-1} \end{bmatrix}, \quad V_1 = \begin{bmatrix} I_p \\ 0 \end{bmatrix}$$
- in other words
$$M_1 = \begin{bmatrix} I_p & 0 \\ 0 & I_q \end{bmatrix} - \begin{bmatrix} 0 \\ A_{21}A_{11}^{-1} \end{bmatrix} \begin{bmatrix} I_p & 0 \end{bmatrix} = \begin{bmatrix} I_p & 0 \\ -A_{21}A_{11}^{-1} & I_q \end{bmatrix}$$
- applying this to A gives
$$M_1 A = \begin{bmatrix} I_p & 0 \\ -A_{21}A_{11}^{-1} & I_q \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ 0 & S \end{bmatrix}$$
where
$$S = A_{22} - A_{21}A_{11}^{-1}A_{12}$$
is called the *Schur complement* of A_{11} in A
- we can easy verify that
$$L_1 := M_1^{-1} = \begin{bmatrix} I_p & 0 \\ A_{21}A_{11}^{-1} & I_q \end{bmatrix}$$
- the analogue of LU factorization of A as a 2×2 block matrix
$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix}$$
is called a *block LU factorization*
- note that L_{11} and L_{22} can be any matrices, not necessarily lower triangular, ditto for U_{11} and U_{22}
- multiplying out the RHS, we see that
$$A_{11} = L_{11}U_{11}$$
- it is also easy to see that
$$L_{22}U_{22} = S = A_{22} - A_{21}A_{11}^{-1}A_{12}$$
- we omitted permutation matrices but they can be easily incorporated: for example, if
$$A_{11} = \Pi_1^{\mathsf{T}} L_1 U_1 \Pi_2^{\mathsf{T}}, \quad S = \Pi_3^{\mathsf{T}} L_2 U_2$$
then we have
$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} \Pi_1^{\mathsf{T}} & 0 \\ 0 & \Pi_3^{\mathsf{T}} \end{bmatrix} \begin{bmatrix} L_1 & 0 \\ \Pi_3 A_{21} \Pi_2 U_1^{-1} & L_2 \end{bmatrix} \begin{bmatrix} U_1 & L_1^{-1}\Pi_1 A_{12} \\ 0 & U_2 \end{bmatrix} \begin{bmatrix} \Pi_2^{\mathsf{T}} & 0 \\ 0 & I \end{bmatrix}$$
- what we discuss here also apply to LDU, LDL^{T}, and Cholesky factorizations

- for example if A is symmetric positive definite, then its Cholesky factorization written in 2×2 block form

  $$\begin{bmatrix} A_{11} & A_{21}^{\mathsf{T}} \\ A_{21} & A_{22} \end{bmatrix} = A = R^{\mathsf{T}} R = \begin{bmatrix} R_{11}^{\mathsf{T}} & 0 \\ R_{12}^{\mathsf{T}} & R_{22}^{\mathsf{T}} \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} = \begin{bmatrix} R_{11}^{\mathsf{T}} R_{11} & R_{11}^{\mathsf{T}} R_{12} \\ R_{12}^{\mathsf{T}} R_{11} & R_{12}^{\mathsf{T}} R_{12} + R_{22}^{\mathsf{T}} R_{22} \end{bmatrix}$$

  is called *block Cholesky factorization*
- again R_{11} and R_{22} need not be upper triangular
- note that since A is symmetric positive definite, so is A_{11} (why?)
- multiplying out the RHS, we see that

  $$A_{11} = R_{11}^{\mathsf{T}} R_{11}$$

- it is also easy to see that

  $$R_{22}^{\mathsf{T}} R_{22} = A_{22} - A_{21} A_{11}^{-1} A_{21}^{\mathsf{T}}$$

11. SOLVING BLOCK LINEAR SYSTEM WITH SCHUR COMPLEMENT

- Schur complement is a very useful notion
- in the following we will assume that A is partitioned as in the previous section with A_{11} nonsingular
- first observe that A is nonsingular if and only if S is nonsinguar
- a very useful application is in *solving linear equations by block elimination*, i.e., solving $A\mathbf{x} = \mathbf{b}$ by partitioning it into

  $$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix} \tag{11.1}$$

  where $\mathbf{b}_1 \in \mathbb{R}^p, \mathbf{b}_2 \in \mathbb{R}^q$
- plugging the first equation

  $$\mathbf{x}_1 = A_{11}^{-1}(\mathbf{b}_1 - A_{12}\mathbf{x}_2) \tag{11.2}$$

  into the second equation yields

  $$(A_{22} - A_{21} A_{11}^{-1} A_{12})\mathbf{x}_2 = \mathbf{b}_2 - A_{21} A_{11}^{-1} \mathbf{b}_1 \tag{11.3}$$

- this allows us to solve $A\mathbf{x} = \mathbf{b}$ as follows
  - form $A_{11}^{-1} A_{12}$ and $A_{11}^{-1}\mathbf{b}$ by solving a system with multiple right hand sides
  - form $S = A_{22} - A_{21} A_{11}^{-1} A_{12}$ and $\widetilde{\mathbf{b}} = \mathbf{b}_2 - A_{21} A_{11}^{-1} \mathbf{b}_1$
  - solve $S\mathbf{x}_2 = \widetilde{\mathbf{b}}$ for $\mathbf{x}_2$
  - solve $A_{11}\mathbf{x}_1 = \mathbf{b}_1 - A_{12}\mathbf{x}_2$ for $\mathbf{x}_1$
- this would be very useful if A_{11} is an 'easy to invert' matrix, e.g., A_{11} is diagonal, banded, orthogonal, Toeplitz, sparse, etc
- such situations where the 'top left corner' of a matrix A has special structure arise more often than you think, especially in
  - numerical optimization (KKT matrix — A_{11} corresponds to the Hessian, the other blocks correpond to the constraints)
  - numerical PDE (discretized version of differential operator with boundary conditions — A_{11} corresponds to the operator, the other blocks to the boundary conditions)
- another way to view the above method is via the factorization

  $$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & 0 \\ A_{21} & S \end{bmatrix} \begin{bmatrix} I & A_{11}^{-1} A_{12} \\ 0 & I \end{bmatrix} \tag{11.4}$$

- so solving $A\mathbf{x} = \mathbf{b}$ can be broken up into two steps

$$\begin{cases} \begin{bmatrix} A_{11} & 0 \\ A_{21} & S \end{bmatrix} \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix} \\ \begin{bmatrix} I & A_{11}^{-1}A_{12} \\ 0 & I \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} \end{cases}$$

- or equivalently

$$\begin{cases} A_{11}\mathbf{y}_1 = \mathbf{b}_1 \\ S\mathbf{y}_2 = \mathbf{b}_2 - A_{21}\mathbf{y}_1 \\ \mathbf{x}_2 = \mathbf{y}_2 \\ \mathbf{x}_1 = \mathbf{y}_1 - A_{11}^{-1}A_{12}\mathbf{y}_2 \end{cases}$$