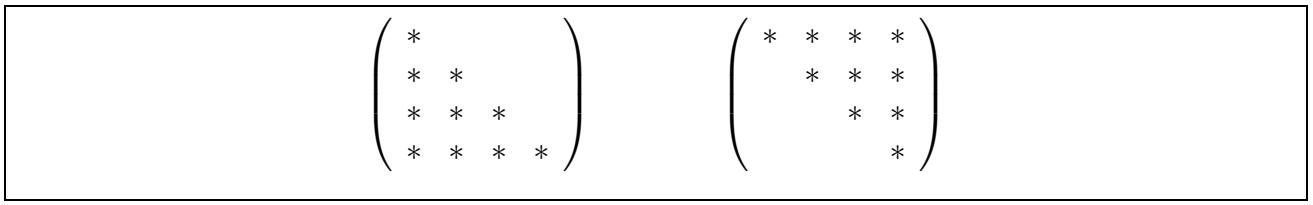

Figure 4.1: schematic representation of lower (left) and upper (right) matrices ($n = 4$); blank spaces represent a 0

4 Gaussian elimination

goal: solve, for given $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{b} \in \mathbb{R}^n$, the linear system of equations

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{4.1}$$

In the sequel, we will often denote the entries of matrices by lower case letters, e.g., the entries of $\mathbf{A}$ are $(\mathbf{A})_{ij} = a_{ij}$. Likewise for vectors, we sometimes write $\mathbf{x}_i = x_i$.

Remark 4.1 *In* `matlab`*, the solution of (4.1) is realized by* $x = A\backslash b$*. In* `python`*, the function* `numpy.linalg.solve` *performs this. In both cases, a routine from* `lapack`[1] *realizes the actual computation. The* `matlab` *realization of the backslash operator* $\backslash$ *is in fact very complex. A very good discussion of many aspects of the realization of* $\backslash$ *can be found in [1].* ∎

4.1 lower and upper triangular matrices

A matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is

- an *upper triangular matrix* if $\mathbf{A}_{ij} = 0$ for $j < i$;
- a *lower triangular matrix* if $\mathbf{A}_{ij} = 0$ for $j > i$.
- a *normalized lower triangular matrix* if, in addition to being lower triangular, it satisfies $\mathbf{A}_{ii} = 1$ for $i = 1, \ldots, n$.

Linear systems where $\mathbf{A}$ is a lower or upper triangular matrix are easily solved by "forward substitution" or "back substitution":

Algorithm 4.2 (solve $\mathbf{Lx} = \mathbf{b}$ using *forward substitution*)
Input: $\mathbf{L} \in \mathbb{R}^{n\times n}$ *lower triangular, invertible,* $\mathbf{b} \in \mathbb{R}^n$
Output: *solution* $\mathbf{x} \in \mathbb{R}^n$ *of* $\mathbf{Lx} = \mathbf{b}$

for $j = 1{:}n$ ***do***
$\quad x_j := \left(b_j - \sum_{k=1}^{j-1} l_{jk}x_k\right) \Big/ l_{jj}$ $\qquad$ ⟦ *convention: empty sum = 0* ⟧
end for

[1] linear algebra package, see `en.wikipedia.org/wiki/LAPACK`

Algorithm 4.3 (solution of $\mathbf{Ux} = \mathbf{b}$ using *back substitution*)
Input: $\mathbf{U} \in \mathbb{R}^{n\times n}$ *upper triangular, invertible,* $\mathbf{b} \in \mathbb{R}^n$
Output: *solution* $\mathbf{x} \in \mathbb{R}^n$ *of* $\mathbf{Ux} = \mathbf{b}$

for *j = n:-1:1* ***do***
$$x_j := \left(b_j - \sum_{k=j+1}^{n} u_{jk} x_k \right) \Big/ u_{jj}$$
end for

The cost of Algorithms 4.2 and 4.3 are $O(n^2)$:

Exercise 4.4 *Compute the number of multiplications and additions in Algorithms 4.2 and 4.3.* ∎

The set of upper and lower triangular matrices are closed under addition and matrix multiplication[2]:

Exercise 4.5 *Let* $\mathbf{L}_1$, $\mathbf{L}_2 \in \mathbb{R}^{n\times n}$ *be two lower triangular matrices. Show:* $\mathbf{L}_1 + \mathbf{L}_2$ *and* $\mathbf{L}_1\mathbf{L}_2$ *are lower triangular matrices. If* $\mathbf{L}_1$ *is additionally invertible, then its inverse* $\mathbf{L}_1^{-1}$ *is also a lower triangular matrix. Analogous results hold for upper triangular matrices.*

Remark 4.6 (representation via scalar products) *Alg. 4.2 (and analogously Alg. 4.3) can be writen using scalar products:*

for *j = 1:n* ***do***
$$x(j) := \Big[b(j) - L(j, 1:j-1) * x(1:j-1) \Big] \Big/ L(j,j)$$
end for

The advantage of such a formulation is that efficient libraries are available such as BLAS level 1[3]*. More generally, rather than realizing dot-products, matrix-vector products, or matrix-matrix-products directly by loops, it is typically advantageous to employ optimized routines such as BLAS.* ∎

Remark 4.7 *In Remark 4.6, the matrix* $\mathbf{L}$ *is accessed in row-oriented fashion. One can reorganize the two loops so as to access* $\mathbf{L}$ *in a column-oriented way. The following algorithm overwrites* $\mathbf{b}$ *with the solution* $\mathbf{x}$ *of* $\mathbf{Lx} = \mathbf{b}$*:*

for *j = 1:n-1* ***do***
$b(j) = b(j)/L(j,j)$
$b(j+1:n) := b(j+1:n) - b(j)L(j+1:n, j)$
end for

∎

finis 6.DS

[2]That is, they have the mathematical structure of a ring
[3]Basic Linear Algebra Subprograms, see `en.wikipedia.org/wiki/Basic_Linear_Algebra_Subprograms`

4.2 classical Gaussian elimination

slide 23

The classical Gaussian elimination process transforms the linear system (4.1) into upper triangular form, which can then be solved by back substitution. We illustrate the procedure:

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ \vdots & & & & & & & & \vdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n \end{array} \tag{4.2}$$

Multiplying the $1st$ equation by

$$l_{21} := \frac{a_{21}}{a_{11}}$$

and subtracting this from the $2nd$ equation produces:

$$\underbrace{(a_{21} - \frac{a_{21}}{a_{11}}a_{11})}_{0}\, x_1 + \underbrace{(a_{22} - \frac{a_{21}}{a_{11}}a_{12})}_{=:a_{22}^{(2)}}\, x_2 + \cdots + \underbrace{(a_{2n} - \frac{a_{21}}{a_{11}}a_{1n})}_{=:a_{2n}^{(2)}}\, x_n \;=\; \underbrace{b_2 - \frac{a_{21}}{a_{11}}b_1}_{=:b_2^{(2)}} \tag{4.3}$$

Multiplying the $1st$ equation by

$$l_{31} := \frac{a_{31}}{a_{11}}$$

and subtracting this from the $3rd$ equation produces:

$$\underbrace{(a_{31} - \frac{a_{31}}{a_{11}}a_{11})}_{0}\, x_1 + \underbrace{(a_{32} - \frac{a_{31}}{a_{11}}a_{12})}_{=:a_{32}^{(2)}}\, x_2 + \cdots + \underbrace{(a_{3n} - \frac{a_{31}}{a_{11}}a_{1n})}_{a_{3n}^{(2)}} \cdot x_n \;=\; \underbrace{b_3 - \frac{a_{31}}{a_{11}}b_1}_{b_3^{(2)}} \tag{4.4}$$

Generally, multiplying for $i = 2, \ldots, n$, the $1st$ equation by $l_{i1} := a_{i1}/a_{11}$ and subtracting this from the ith equation yields the following equivalent system of equations:

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ & & a_{22}^{(2)}x_2 & + & \cdots & + & a_{2n}^{(2)}x_n & = & b_2^{(2)} \\ & & \vdots & & & & & & \vdots \\ & & a_{n2}^{(2)}x_2 & + & \cdots & + & a_{nn}^{(2)}x_n & = & b_n^{(2)} \end{array} \tag{4.5}$$

Repeating this process for the $(n-1) \times (n-1)$ subsystem

$$\begin{array}{ccccccc} a_{22}^{(2)}x_2 & + & \cdots & + & a_{2n}^{(2)}x_n & = & b_2^{(2)} \\ \vdots & & & & & & \vdots \\ a_{n2}^{(2)}x_2 & + & \cdots & + & a_{nn}^{(2)}x_n & = & b_n^{(2)} \end{array}$$

of (4.5) yields

$$\begin{array}{ccccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & a_{13}x_3 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ & & a_{22}^{(2)}x_2 & + & a_{23}^{(2)}x_3 & + & \cdots & + & a_{2n}^{(2)}x_n & = & b_2^{(2)} \\ & & & & a_{33}^{(3)}x_3 & + & \cdots & + & a_{3n}^{(3)}x_n & = & b_3^{(3)} \\ & & & & \vdots & & & & \vdots & & \\ & & & & a_{n3}^{(3)}x_3 & & \cdots & & a_{nn}^{(3)}x_n & = & b_n^{(3)} \end{array} \tag{4.6}$$

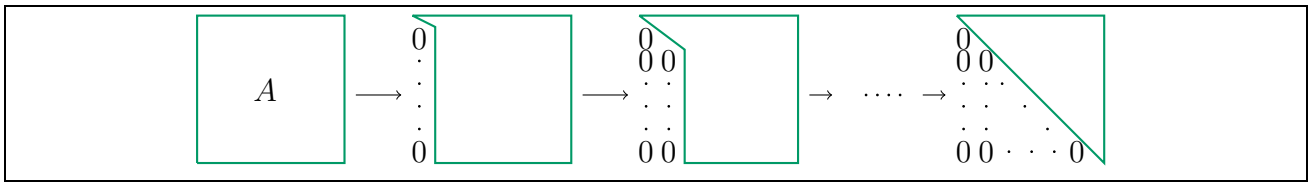

Figure 4.2: Gaussian elimination: reduction to upper triangular form

One repeats this procedure until one has reached triangular form. The following Alg. 4.8 realizes the above procedure: It overwrites the matrix $\mathbf{A}$ so that its upper triangle contains the final triangular form, which we will denote $\mathbf{U}$ below; the entries l_{ik} computed on the way will be collected in a normalized lower triangular matrix $\mathbf{L}$.

Algorithm 4.8 (Gaussian elimination without pivoting)
Input: $\mathbf{A}$
Output: *non-trivial entries of* $\mathbf{L}$ *and* $\mathbf{U}$*;* $\mathbf{A}$ *is overwritten by* $\mathbf{U}$

for $k = 1 : (n-1)$ ***do***
  for $i = (k+1) : n$ ***do***
    $l_{ik} := \frac{a_{ik}}{a_{kk}}$
    $A(i, [k+1:n]) += -l_{ik} \cdot A(k, [k+1:n])$
  end for
end for

Remark 4.9 *In* (4.8) *below, we will see that* $\mathbf{A} = \mathbf{LU}$*, where* $\mathbf{U}$ *and* $\mathbf{L}$ *are computed in Alg. 4.8. Typically, the off-diagonal entries of* $\mathbf{L}$ *are stored in the lower triangular part of* $\mathbf{A}$ *so that effectively,* $\mathbf{A}$ *is overwritten by its LU-factorization.* ∎

Exercise 4.10 *Expand Alg. 4.8 so as to include the modifications of the right-hand side* $\mathbf{b}$*.* ∎

4.2.1 Interpretation of Gaussian elimination as an LU-factorization

With the coefficients l_{ij} computed above (e.g., in Alg. 4.8) one can define the matrices

$$\mathbf{L}^{(k)} := \begin{pmatrix} 1 & & & & & & & \\ 0 & \ddots & & & & & & \\ \vdots & \ddots & \ddots & & & & & \\ \vdots & & 0 & 1 & & & & \\ \vdots & & \vdots & l_{k+1,k} & \ddots & & & \\ \vdots & & \vdots & \vdots & 0 & \ddots & & \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \ddots & \\ 0 & \cdots & 0 & l_{n,k} & 0 & \cdots & 0 & 1 \end{pmatrix}, \qquad k = 1, \ldots, n-1$$

Exercise 4.11 *Check that the inverse of* $\mathbf{L}^{(k)}$ *is given by*

$$(\mathbf{L}^{(k)})^{-1} = \begin{pmatrix} 1 & & & & & & & \\ 0 & \ddots & & & & & & \\ \vdots & \ddots & \ddots & & & & & \\ \vdots & & 0 & 1 & & & & \\ \vdots & & \vdots & -l_{k+1,k} & \ddots & & & \\ \vdots & & \vdots & \vdots & 0 & \ddots & & \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \ddots & \\ 0 & \cdots & 0 & -l_{n,k} & 0 & \cdots & 0 & 1 \end{pmatrix}. \tag{4.7}$$

Each step of the above Gaussian elimination process sketched in Fig. 4.2 is realized by a multiplication from the left by a matrix, in fact, the matrix $(\mathbf{L}^{(k)})^{-1}$ (cf. (4.7)). That is, the Gaussian elimination process can be described as

$$\begin{aligned} &\mathbf{A} = \mathbf{A}^{(1)} \;\to\; \mathbf{A}^{(2)} = (\mathbf{L}^{(1)})^{-1}\mathbf{A}^{(1)} \;\to\; \mathbf{A}^{(3)} = (\mathbf{L}^{(2)})^{-1}\mathbf{A}^{(2)} = (\mathbf{L}^{(2)})^{-1}(\mathbf{L}^{(1)})^{-1}\mathbf{A}^{(1)} \;\to\; \dots \\ &\to \underbrace{\mathbf{A}^{(n)}}_{=:\mathbf{U} \text{ upper triangular}} = (\mathbf{L}^{(n-1)})^{-1}\mathbf{A}^{(n-1)} = \dots = (\mathbf{L}^{(n-1)})^{-1}(\mathbf{L}^{(n-2)})^{-1}\dots(\mathbf{L}^{(2)})^{-1}(\mathbf{L}^{(1)})^{-1}\mathbf{A}^{(1)} \end{aligned}$$

Rewriting this yields, the *LU*-factorization

$$\mathbf{A} = \underbrace{\mathbf{L}^{(1)}\cdots\mathbf{L}^{(n-1)}}_{=:\mathbf{L}}\mathbf{U}$$

The matrix $\mathbf{L}$ is a lower triangular matrix as the product of lower triangular matrices (cf. Exercise 4.5). In fact, due to the special structure of the matrices $\mathbf{L}^{(k)}$, it is given by

$$\mathbf{L} = \begin{pmatrix} 1 & & & \\ l_{21} & \ddots & & \\ \vdots & \ddots & \ddots & \\ l_{n1} & \cdots & l_{n,n-1} & 1 \end{pmatrix}$$

as the following exercise shows:

Exercise 4.12 *For each* k *the product* $\mathbf{L}^{(k)}\mathbf{L}^{(k+1)}\cdots\mathbf{L}^{(n-1)}$ *is given by*

$$\mathbf{L}^{(k)}\mathbf{L}^{(k+1)}\cdots\mathbf{L}^{(n-1)} = \begin{pmatrix} 1 & & & & & & & \\ 0 & \ddots & & & & & & \\ \vdots & \ddots & \ddots & & & & & \\ \vdots & & 0 & 1 & & & & \\ \vdots & & \vdots & l_{k+1,k} & 1 & & & \\ \vdots & & \vdots & \vdots & l_{k+2,k+1} & \ddots & & \\ \vdots & & \vdots & \vdots & \vdots & \ddots & \ddots & \\ 0 & \cdots & 0 & l_{n,k} & l_{n,k+1} & \cdots & l_{n,n-1} & 1 \end{pmatrix}.$$

Thus, we have shown that Gaussian elimination produces a *factorization*

$$\mathbf{A} = \mathbf{LU}, \tag{4.8}$$

where $\mathbf{L}$ and $\mathbf{U}$ are lower and upper triangular matrices determined by Alg. 4.8.

4.3 LU-factorization

In numerical practice, linear systems are solved by computing the factors $\mathbf{L}$ and $\mathbf{U}$ in (4.8) and the system is then solved with one forward and one back substitution:

1. compute $\mathbf{L}$, $\mathbf{U}$ such that $\mathbf{A} = \mathbf{LU}$
2. solve $\mathbf{Ly} = \mathbf{b}$ using forward substitution
3. solve $\mathbf{Ux} = \mathbf{y}$ using back substitution

Remark 4.13 *In* `matlab`*, the LU-factorization is realized by* `lu(A)`*. In* `python` *one can use* `scipy.linalg.lu`. ∎

As we have seen in Section 4.2.1, the LU-factorization can be computed with Gaussian elimination. An alternative way of computing the factors $\mathbf{L}$, $\mathbf{U}$ is given in the following section[4]

4.3.1 Crout's algorithm for computing LU-factorization

We seek $\mathbf{L}$, $\mathbf{U}$ such that

$$\begin{pmatrix} 1 & & & \\ l_{21} & \ddots & & \\ \vdots & \ddots & \ddots & \\ l_{n1} & \cdots & l_{n,n-1} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & \cdots & \cdots & u_{1n} \\ & \ddots & & \vdots \\ & & \ddots & \vdots \\ & & & u_{nn} \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} a_{11} & \cdots & \cdots & a_{1n} \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ a_{n1} & \cdots & \cdots & a_{nn} \end{pmatrix}$$

This represents n^2 equations for n^2 unknowns, i.e., we are looking for l_{ij}, u_{ij}, such that

$$a_{ik} \stackrel{!}{=} \sum_{j=1}^{n} l_{ij}\ u_{jk}, \quad \forall i,k = 1,\ldots,n.$$

$\mathbf{L}$ is lower triangular, $\mathbf{U}$ is upper triangular $\Longrightarrow$

$$a_{ik} \stackrel{!}{=} \sum_{j=1}^{\min(i,k)} l_{ij}\ u_{jk} \qquad \forall i,k = 1,\ldots,n \tag{4.9}$$

[4]One reason for studying different algorithms is that the entries of $\mathbf{L}$ and $\mathbf{U}$ are computed in a different order so that these algorithms differ in their memory access and thus potentially in actual timings.

Idea:
Traverse the n^2 equations in (4.9) in following order: ("**Crout ordering**")]]

$$\begin{array}{c}(1,1)\ ,\ (1,2)\ ,\ \ldots\ ,(1,n)\\ (2,1)\ ,\ (3,1)\ ,\ \ldots\ ,(n,1)\\ (2,2)\ ,\ (2,3)\ ,\ \ldots\ ,(2,n)\\ (3,2)\ ,\ (4,2)\ ,\ \ldots\ ,(n,2)\\ etc.\end{array}$$

Procedure:

1. step: $i = 1,\ k = 1,\ldots,n$ in (4.9):

$$\underbrace{l_{11}}_{=1} u_{1k} \stackrel{!}{=} a_{1k}$$

$\Rightarrow$ $U(1,:)$ can be computed

2. step: $k = 1,\ i = 2,\ldots,n$ in (4.9):

$$l_{i1}u_{11} \stackrel{!}{=} a_{i1}$$

$\Rightarrow$ $L([2:n],1)$ can be determined

3. step: $i = 2,\ k = 2,\ldots,n$ in (4.9):

$$\underbrace{l_{21}}_{\text{is known by 2. step}}\ \underbrace{u_{1k}}_{\text{is known by 1. step}} + \underbrace{l_{22}}_{=1} u_{2k} \stackrel{!}{=} a_{2k} \qquad \text{for } k = 2,\ldots,n$$

$\Rightarrow$ can compute $U(2,[2:n])$

4. step: $k = 2,\ i = 3,\ldots,n$ in (4.9):

$$\underbrace{l_{i1}}_{\text{known by 2. step}}\ \underbrace{u_{12}}_{\text{known by 1. step}} + l_{i2}\ \underbrace{u_{22}}_{\text{known by 3. step}} \stackrel{!}{=} a_{i2} \qquad \text{for } i = 3,\ldots,n$$

$\Rightarrow$ can compute $L([3:n],2)$

$\vdots$

$\vdots$

$\vdots$

The procedure is formalized in the following

Algorithm 4.14 (Crout's LU-factorization)
Input: invertible matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ that has an LU-factorization
Output: the non-trivial entries of the normalized LU-factorization

for $i = 1 : n$ ***do***
  for $k = i : n$ ***do***
    $u_{ik} := a_{ik} - \sum_{j=1}^{i-1} l_{ij} u_{jk}$
  end for
  for $k = i + 1 : n$ ***do***
    $l_{ki} := \left(a_{ki} - \sum_{j=1}^{i-1} l_{kj} u_{ji} \right) \Big/ u_{ii}$
  end for
end for

Remark 4.15 (cost when solving (4.1) with LU-factorization)

- *The LU-factorization dominates with $O(n^3)$ (more precisely: $2/3n^3 + O(n^2)$ floating point operations) the total cost, since the cost of back substitution and forward substitution are $O(n^2)$*
- *An advantage of an LU-factorization arises, when problems with multiple right-hand sides are considered: solving $\mathbf{Ax} = \mathbf{b}$ for M right-hand sides $\mathbf{b}$, requires only a single LU-factorization, i.e., the cost are $\frac{2}{3}n^3 + 2Mn^2$* ∎

In practice $\mathbf{A}$ is overwritten by its LU-decomposition:

Algorithm 4.16 (LU-factorization with overwriting A)
Input: $\mathbf{A}$*, invertible,* A *has a LU-factorization*
Output: algorithm replaces a_{ij} *with* u_{ij} *for* $j \geq i$
and with l_{ij} *for* $j < i$

for $i = 1 : n$ ***do***
  for $k = i : n$ ***do***
    $a_{ik} := a_{ik} - \sum_{j=1}^{i-1} a_{ij}\, a_{jk}$
  end for
  for $k = (i + 1) : n$ ***do***
    $a_{ki} := \left(a_{ki} - \sum_{j=1}^{i-1} a_{kj}\, a_{ji} \right) \Big/ a_{ii}$
  end for
end for

4.3.2 banded matrices

A matrix $A \in \mathbb{R}^{n \times n}$ is a *banded matrix* with *upper bandwidth* q and *lower bandwidth* p if $a_{ik} = 0$ for all i, k with $i > k + p$ or $k > i + q$. The following theorem shows that banded matrices are

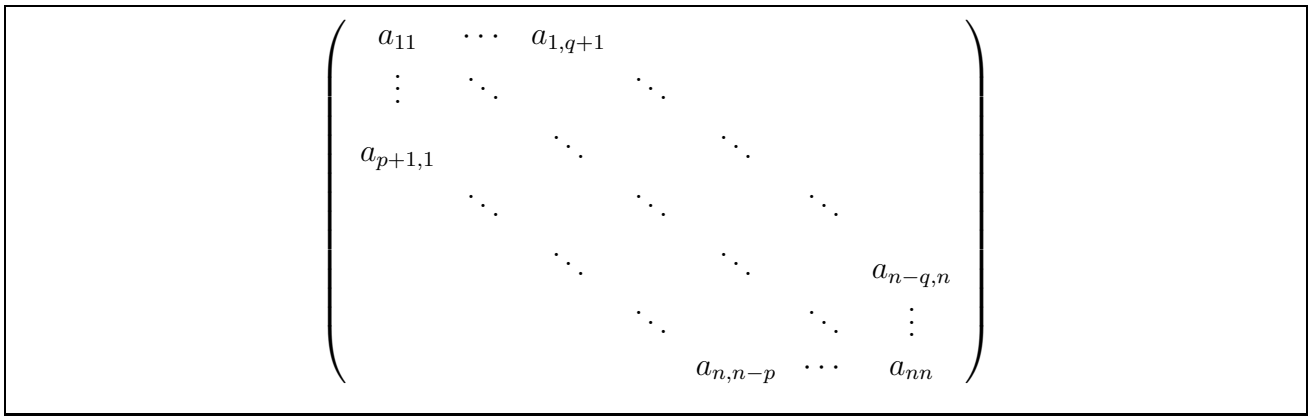

Figure 4.3: banded matrix with upper bandwidth q and lower bandwidth p.

of interest if p and q are small (compared to n):

Theorem 4.17 *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a banded matrix with upper bandwidth q and lower bandwidth p. Let $\mathbf{A}$ be invertible and admit an LU-factorization. Then:*

(i) $\mathbf{L}$ *has lower bandwidth p and* $\mathbf{U}$ *has upper bandwidth q.*

(ii) Cost to solve $\mathbf{Ax} = \mathbf{b}$:

- *(a) $O(npq)$ floating point operations (flops) to determine LU-factorization*
- *(b) $O(np)$ flops to solve $\mathbf{Ly} = \mathbf{b}$*
- *(c) $O(nq)$ flops to solve $\mathbf{Ux} = \mathbf{y}$*

Proof: (Exercise) Prove (i) for the special case of a tridiagonal matrix, i.e., $p = q = 1$. To that end, proceed by induction on the matrix size n:

- $n = 1$ ✓
- for the induction step $n \to n{+}1$ make the ansatz

$$A = \left(\begin{array}{ccccc|c} & & & & & 0 \\ & & & & & \vdots \\ & & A_n & & & \vdots \\ & & & & & 0 \\ & & & & & a_{n,n+1} \\ \hline 0 & \cdots & \cdots & 0 & a_{n+1,n} & a_{n+1,n+1} \end{array}\right) \overset{!}{=} \left(\begin{array}{c|c} L_n & 0 \\ \hline l^\top & 1 \end{array}\right) \left(\begin{array}{c|c} U_n & u \\ \hline 0 & \rho \end{array}\right)$$

and compute $l^\top$, u, and ρ. Use the structure of L_n, U_n given by the induction hypothesis.

□

slide 24

4.3.3 Cholesky-factorization

A particularly important class of matrices $\mathbf{A}$ is that of symmetric positive definite (SPD) matrices:

- $\mathbf{A}$ is symmetric, i.e., $\mathbf{A}_{ij} = \mathbf{A}_{ji}$ for all i, j
- $\mathbf{A}$ is positive definite, i.e., $\mathbf{x}^\top \mathbf{A}\mathbf{x} > 0$ for all $\mathbf{x} \neq 0$.

Remark 4.18 *An alternative criterion for positive definiteness of a symmetric matrix is that all its eigenvalue are positive.* ∎

For SPD matrices, one typically employs a variant of the LU-factorization, namely, the Cholesky-factorization, i.e.,

$$\mathbf{A} = \mathbf{C}\mathbf{C}^\top, \tag{4.10}$$

where the Cholesky factor $\mathbf{C}$ is lower triangular (but not normalized, i.e., the entries $\mathbf{C}_{ii}$ are not necessarily 1).

Exercise 4.19 *Formulate an algorithm to compute* $\mathbf{C}$. Hint: *Proceed as in Crout's method for the LU-factorization.* ∎

Remark 4.20 *If an SPD matrix* $\mathbf{A}$ *is banded with bandwidth* $p = q$*, then the Cholesky factor* $\mathbf{C}$ *is also banded with the same bandwidth.* ∎

Remark 4.21 *The cost of a Cholesky factorization (of either a full matrix or a banded matrix) is about half of that of the corresponding LU-factorization since only half the entries need to be computed.* ∎

Remark 4.22 *A Cholesky factorization is computed in* `matlab` *with* `chol`. ∎

slide 25

4.3.4 skyline matrices

slide 26

Banded matrices are a particular case of *sparse matrices*, i.e., matrices with "few" non-zero entries. We note that the LU-factors have the same sparsity pattern, i.e., the zeros of $\mathbf{A}$ outside the band are inherited by the factors $\mathbf{L}$, $\mathbf{U}$.

Another important special case of sparse matrices are so-called *skyline matrices* as depicted on the left side of Fig. 4.4. More formally, a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ is called a *skyline matrix*, if for $i = 1, \ldots, n$ there are numbers p_i, $q_i \in \mathbb{N}_0$ such that

$$a_{ij} = 0 \quad \text{if } j < i - p_i \text{ or } i < j - q_j. \tag{4.11}$$

We have without proof:

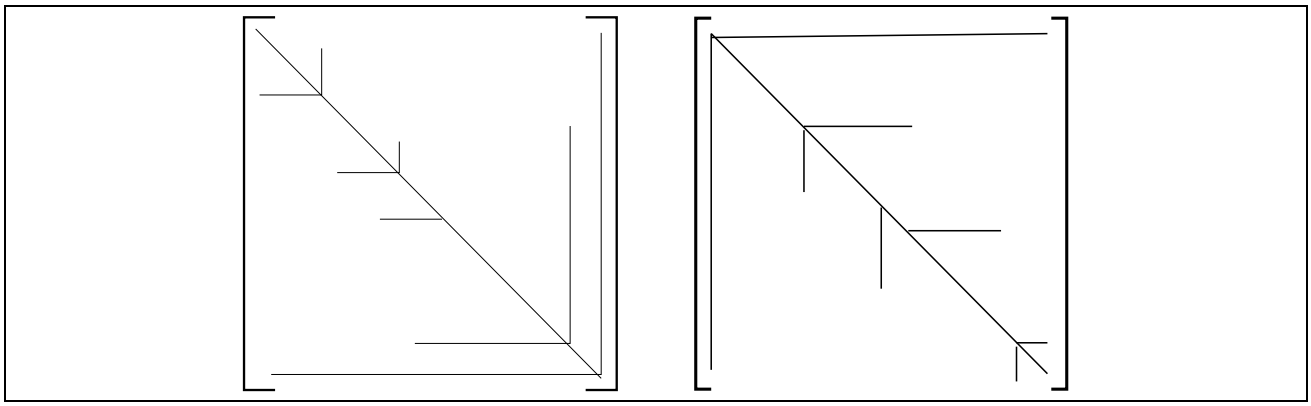

Figure 4.4: lines indicate non-zero entries. Left: *skyline* matrix, whose sparsity pattern is inherited by *LU*-factorization. Right: not a *skyline*-Matrix and the *LU*-factorization does *not* inherit the sparsity pattern.

$$A = \begin{pmatrix} 1 & & & 1 & & & 1 \\ & 1 & & 2 & & & 2 \\ & & 1 & 3 & & & 3 \\ 1 & 2 & 3 & 5 & & & 18 \\ & & & & 1 & & 5 \\ & & & & & 1 & 6 \\ 1 & 2 & 3 & 18 & 5 & 6 & 92 \end{pmatrix} \qquad L = U^\top = \begin{pmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ 1 & 2 & 3 & 1 & & & \\ & & & & 1 & & \\ & & & & & 1 & \\ 1 & 2 & 3 & 4 & 5 & 6 & 1 \end{pmatrix}$$

Figure 4.5: $\mathbf{A} \in \mathbb{R}^{7\times 7}$ and its *LU*-factorization.

Theorem 4.23 *Let* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *be a skyline matrix, i.e., there are* p_i, q_i *with (4.11). Let* $\mathbf{A}$ *have an LU-factorization* $\mathbf{A} = \mathbf{LU}$. *Then the matrices* $\mathbf{L}$, $\mathbf{U}$ *satisfy:*

$$l_{ij} = 0 \quad \text{for } j < i - p_i, \qquad u_{ij} = 0 \quad \text{for } i < j - q_j.$$

Theorem 4.23 states that the factors $\mathbf{L}$ and $\mathbf{U}$ have the same sparsity pattern as $\mathbf{A}$. Figure 4.5 illustrates this for a simple example. Obviously, this can be exploited algorithmically to economize on memory requirement and computing time by simply computing the non-zero entries of $\mathbf{L}$ and $\mathbf{U}$. Note that the matrices in Fig. 4.4 should not be treated as banded matrices as then the bands p, q would be n. The right example in Fig. 4.4 is not a skyline matrix, and the sparsity pattern of $\mathbf{A}$ is lost in the course of the *LU*-factorization: $\mathbf{L}$ ist in general a fully populated lower triangular matrix and $\mathbf{U}$ a fully populated upper triangular matrix. This is called *fill in*.

Exercise 4.24 *The sparsity pattern of matrices can be checked in* `matlab` *with the command* `spy`. *Check the sparsity patterns of the LU-factorization of the matrices* $\mathbf{A}$ *given above.* ∎

Remark 4.25 *Modern solvers for sparse linear systems typically perform as a preprocessing step row and column permutations so as to minimize fill-in during factorization. (→ see Approximate Minimum Degree, Reverse Cuthill-McKee).* ∎

finis 7.DS

4.4 Gaussian elimination with pivoting

4.4.1 Motivation

So far, we *assumed* that $\mathbf{A}$ admits a factorization $\mathbf{A} = \mathbf{LU}$. However, even if $\mathbf{A}$ is invertible, this need not be the case as the following example shows:

Exercise 4.26 *Prove that the matrix*

$$\mathbf{A} = \begin{pmatrix} 0 & 1 \\ 3 & 2 \end{pmatrix}$$

does not *have a factorization* $\mathbf{A} = \mathbf{LU}$ *with normalized lower triangular matrix* $\mathbf{L}$ *and upper triangular matrix* $\mathbf{U}$.

The key observation is that permuting the rows of $\mathbf{A}$ leads to a matrix that has an LU-factorization: Let

$$\mathbf{P} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

be the permutation matrix that interchanges the rows 1 and 2 of $\mathbf{A}$:

$$\mathbf{PA} = \begin{pmatrix} 3 & 2 \\ 0 & 1 \end{pmatrix}$$

This matrix has an LU-factorization. The general principle is:

Theorem 4.27 *Let* $\mathbf{A} \in \mathbb{R}^{n \times n}$ *be invertible. Then there exists a permutation matrix* $\mathbf{P}$*, a normalized lower triangular matrix* $\mathbf{L}$*, and an upper triangular matrix* $\mathbf{U}$ *such that* $\mathbf{LU} = \mathbf{PA}$*. Here* $\mathbf{PA}$ *is a permutation of the rows of* $\mathbf{A}$.

Exercise 4.28 *Let* $\mathbf{P}$ *be given by*

$$\mathbf{P} = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 0 & & 1 & & \\ & & & \ddots & & & \\ & & 1 & & 0 & & \\ & & & & & 1 & \\ & & & & & & \ddots \end{pmatrix}$$

where the off-diagonal 1 *are in the positions* (i_1, i_2) *and* (i_2, i_1) *(with* $i_1 \neq i_2$*). Show: The matrix* $\mathbf{PA}$ *is the matrix* $\mathbf{A}$ *with rows* i_1 *and* i_2 *interchanged. Furthermore,* $\mathbf{P}^{-1} = \mathbf{P}^\top = \mathbf{P}$. ∎

4.4.2 Algorithms

A factorization as given in Theorem 4.27 can be obtained by modifying Alg. 4.8: if a definition of an l_{ij} is not possible because $a_{jj}^{(j)} = 0$, then a row j' from the rows $\{j+1, \ldots, n\}$ is chosen with $a_{jj'}^{(j)} \neq 0$ (this is possible since otherwise $\mathbf{A}$ is rank deficient). One interchanges rows j and j' and continues with Alg. 4.8.
Mathematically, it is immaterial, which row j' is chosen. Numerically, one typically chooses the row j' such that the corresponding entry $a_{jj'}^{(j)}$ is the largest (in absolute value) from the set $\{a_{jJ}^{(j)} \; J = j+1, \ldots, n\}$. This is called *partial pivoting*.
To formalize the procedure, we need the concept of permutation matrices:

Definition 4.29 *Let* $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ *be a permutation*[5]*. Then,*

$$\mathbf{P}_\pi := \Big(e_{\pi(1)} \; , \; \ldots \; , \; e_{\pi(n)} \Big)$$

denotes the corresponding permutation matrix.

Theorem 4.30 *Let* $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ *be a permutation. Then:*

(i) $\mathbf{P}_\pi \; e_i := e_{\pi(i)} \qquad \forall i$

(ii) $\mathbf{P}_\pi^{-1} = \mathbf{P}_\pi^\top$

(iii) $\mathbf{P}_\pi \mathbf{A}$ *is obtained from* $\mathbf{A}$ *by row permutation: the* i*-th row of* $\mathbf{A}$ *becomes the* $\pi(i)$*-th row of* $\mathbf{P}_\pi \mathbf{A}$*.*

(iv) $\mathbf{A}\mathbf{P}_\pi$ *is obtained from* $\mathbf{A}$ *by column permutation.*

Proof: Exercise. □

In practice, the LU-factorization of $\mathbf{A}$ with (row) pivoting operates directly on the matrix $\mathbf{A}$, i.e., overwrites the matrix $\mathbf{A}$ and the row permutations are not done explicitly but implicitly with pointers. This leads to:

Algorithm 4.31 (Gaussian elimination with row pivoting) *Input: invertible* $\mathbf{A} \in \mathbb{R}^{n \times n}$
Output: factorization $\mathbf{PA} = \mathbf{LU}$*, where* $\mathbf{A}$ *is overwritten by* $\mathbf{U}$*:*
$u_{ij} = a_{\pi(i),j}$ *and* $\mathbf{P} = \mathbf{P}_\pi^{-1} = \mathbf{P}_\pi^\top$ *is implicitly given by the vector* π

$\pi := (1, 2, \ldots, n)$
for $k = 1 : (n-1)$ ***do***
  seek $p \in \{k, \ldots, n\}$ *s.t.* $|a_{pk}| \geq |a_{ik}| \qquad \forall i \geq k$
  interchange k*-th and* p*-th entry of vector* π
  for $i = (k+1) : n$ ***do***
    $l_{\pi(i),k} := \frac{a_{\pi(i),k}}{a_{\pi(k),k}}$
    for $j = (k+1) : n$ ***do***
      $a_{\pi(i),j} := a_{\pi(i),j} - l_{\pi(i),k} \, a_{\pi(k),j}$

[5] That is, π is a bijection

end for
end for
end for

Theorem 4.32 *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be invertible. Then Algorithm 4.31 yields a factorization $\mathbf{LU} = \mathbf{PA}$, where $\mathbf{L}$ satisfies $|l_{ij}| \leq 1 \ \forall i, j$. $\mathbf{P}$ is a permutation matrix.*

Remark 4.33 *The* `matlab`/`python` *commands to compute LU-factorization typically return matrices $\mathbf{L}$, $\mathbf{U}$, $\mathbf{P}$ of a factorization $\mathbf{LU} = \mathbf{PA}$ and perform (at least some) pivoting.* ∎

Exercise 4.34 *Given a factorization $\mathbf{LU} = \mathbf{PA}$, determine the solution $\mathbf{x}$ of $\mathbf{Ax} = \mathbf{b}$.* ∎

4.4.3 numerical difficulties: choice of the pivoting strategy

Alg. 4.31 selected the largest element from among the possible pivot elements. Why this is a good strategy becomes more clear when one studies the case that the pivot element is non-zero but small as in the following example.
Consider for small ε the matrix

$$\mathbf{A} = \begin{pmatrix} \varepsilon & 1 \\ 1 & 1 \end{pmatrix}$$

Its LU-factorization is

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ \varepsilon^{-1} & 1 \end{pmatrix} \begin{pmatrix} \varepsilon & 1 \\ 0 & 1-\varepsilon^{-1} \end{pmatrix}$$

Let now $\varepsilon = 10^{-20}$. In typical floating point arithmetic (16 digits) one therefore expects this to be realized as with approximate factors $\widehat{\mathbf{L}}$, $\widehat{\mathbf{U}}$ given by

$$\widehat{\mathbf{L}} = \begin{pmatrix} 1 & 0 \\ 10^{20} & 1 \end{pmatrix}, \qquad \widehat{\mathbf{U}} = \begin{pmatrix} 10^{-20} & 1 \\ 0 & -10^{20} \end{pmatrix}$$

If one performs the forward and back substitution for the linear system

$$\widehat{\mathbf{L}}\widehat{\mathbf{U}}\mathbf{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

one obtains $\mathbf{x} = (0, 1)$ whereas the correct solution of the original problem is (up to machine precision) $\mathbf{x} = (-1, 1)$. That is, the solution is completely inaccurate. In contrast, solving the row-pivoted problem yields the correct solution.
Rather than fully analyzing round errors for the solution of linear systems, let us give a heuristic, why the pivoting strategy is reasonable. Let us assume that the entries of the matrix $\mathbf{A}$ and the right-hand side vector $\mathbf{b}$ and the solution vector $\mathbf{x}$ are "moderate" in size. If small pivots are used, i.e., some $a_{kk}^{(k)}$ is small during Gaussian elimination, then one should expect the entries of $\mathbf{L}$ to be large (as in the above example). Hence, in the course of the forward or back substitution, one should expect large intermediate values. If the final result is again "moderate", then one should fear that this is achieved by subtracting numbers of similar size. That is, one should fear cancellation and thus loss of accuracy. In Alg. 4.31 the pivoting choice ensures that the

entries of $\mathbf{L}$ are all bounded by 1, thus moderate. We stress that this is not an insurance against roundoff problems as the pivoting strategy does not control the size of the entries of $\mathbf{U}$. While this is possible ("full pivoting"), it is usually avoided due to the cost considerations.
slide 27

4.5 condition number of a matrix A

An important quantity to assess the effect of errors in the data (i.e., the right-hand side $\mathbf{b}$ or the matrix $\mathbf{A}$) on the solution is the *condition number* (see (4.13) ahead). In order to define it, let $\|\cdot\|$ be a norm on $\mathbb{R}^n$. On the space of matrices $\mathbf{A} \in \mathbb{R}^{n\times n}$, we define the *induced matrix norm* by

$$\|\mathbf{A}\| := \max_{0\neq\mathbf{x}\in\mathbb{R}^n} \frac{\|\mathbf{A}\mathbf{x}\|}{\|\mathbf{x}\|}. \tag{4.12}$$

Exercise 4.35 *Show:*

1. *If $\|\cdot\| = \|\cdot\|_\infty$, then the induced matrix norm $\|\cdot\|_\infty$ is given by ("row sum norm")*

$$\|\mathbf{A}\|_\infty = \max_i \sum_{j=1}^n |a_{ij}|$$

2. *If $\|\cdot\| = \|\cdot\|_1$, then the induced matrix norm $\|\cdot\|_\infty$ is given by ("column sum norm")*

$$\|\mathbf{A}\|_1 = \max_j \sum_{i=1}^n |a_{ij}|$$

3. *For $\|\cdot\|_2$ one has $\|\mathbf{A}\|_2^2 = \lambda_{max}(\mathbf{A}^\top\mathbf{A})$, where λ_{max} denotes the maximal eigenvalue.* ∎

Exercise 4.36 *Prove: For $\mathbf{A}$, $\mathbf{B} \in \mathbb{R}^{n\times n}$ there holds $\|\mathbf{AB}\| \leq \|\mathbf{A}\|\|\mathbf{B}\|$.* ∎

We study the effect of perturbing the right-hand side $\mathbf{b}$. We consider

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

In order to estimate $\Delta\mathbf{x}$ in terms of $\Delta\mathbf{b}$ we note $\mathbf{A}\Delta\mathbf{x} = \Delta\mathbf{b}$ as well as $\|\mathbf{b}\| = \|\mathbf{A}\mathbf{A}^{-1}\mathbf{b}\| \leq \|\mathbf{A}\|\|\mathbf{A}^{-1}\mathbf{b}\|$ so that

$$\begin{aligned} \text{absolute error: } & \|\Delta\mathbf{x}\| = \|\mathbf{A}^{-1}\Delta\mathbf{b}\| \leq \|\mathbf{A}^{-1}\|\|\Delta\mathbf{b}\|, \\ \text{relative error: } & \frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} = \frac{\|\mathbf{A}^{-1}\Delta\mathbf{b}\|}{\|\mathbf{A}^{-1}\mathbf{b}\|} \leq \frac{\|\mathbf{A}^{-1}\|\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|/\|\mathbf{A}\|} =\leq \|\mathbf{A}\|\|\mathbf{A}^{-1}\|\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|}. \end{aligned}$$

The quantity

$$\kappa(\mathbf{A}) := \|\mathbf{A}\|\|\mathbf{A}^{-1}\| \tag{4.13}$$

is called the *condition number* of the matrix $\mathbf{A}$ (with respect to the norm $\|\cdot\|$). It measures how a perturbation in the right-hand side $\mathbf{b}$ could impact the solution of the linear system.

Remark 4.37 *In floating point arithmetic, a rounding error* $\|\Delta\mathbf{b}\|/\|\mathbf{b}\| = O(\varepsilon)$ *with machine precision* ε *is typically unavoidable. Thus,* $\varepsilon\kappa(\mathbf{A})$ *indicates of the level of accuracy that could at best be expected.* ∎

Remark 4.38 *The condition number also appears when one assesses the impact of perturbations of matrix entries. One has (see, e.g., the class notes of Schranz-Kirlinger)*

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

where $\mathbf{x}$ *and* $\widetilde{\mathbf{x}}$ *solve*

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \qquad (\mathbf{A} + \Delta\mathbf{A})\widetilde{\mathbf{x}} = \mathbf{b}$$

∎

4.6 QR-factorization (CSE)

The basic idea above to solve linear systems is to write $\mathbf{Ax} = \mathbf{b}$ as $\mathbf{LUx} = \mathbf{b}$ since the linear systems $\mathbf{Ly} = \mathbf{b}$ and $\mathbf{Ux} = \mathbf{y}$ are easily solved by forward and back substitution. We now present a further factorization $\mathbf{A} = \mathbf{QR}$, the $\mathbf{QR}$-*factorization*, where the factors $\mathbf{Q}$ and $\mathbf{R}$ are such that the linear systems $\mathbf{Qy} = \mathbf{b}$ and $\mathbf{Rx} = \mathbf{y}$ are easily solved. Although computing the $\mathbf{QR}$-factorization is about twice as expensive as the LU-factorization it the preferred method for ill-conditioned matrices $\mathbf{A}$.

4.6.1 orthogonal matrices

Definition 4.39 *A matrix $Q \in \mathbb{R}^{n\times n}$ is **orthogonal**, if $\mathbf{Q}^\top\mathbf{Q} = I$. $\mathcal{O}_n$ denotes the set of orthogonal $n \times n$-matrices.*

Theorem 4.40 *(i) The product of two orthogonal matrices is orthogonal; the inverse of an orthogonal matrix is orthogonal.*[6]

(ii) If $\mathbf{Q} \in \mathcal{O}_{n-k}$, then $\begin{pmatrix} I_k & 0 \\ 0 & \mathbf{Q}_{n-k} \end{pmatrix} \in \mathcal{O}_n$

(iii) $\mathbf{Q} \in \mathcal{O}_n \;\Rightarrow\; \|\mathbf{Qx}\|_2 = \|\mathbf{x}\|_2 \;\forall \mathbf{x} \in \mathbb{R}^n$ ⟦ *that is, $\mathbf{Q}$ preserves length/euclidean norm—it is this property that makes orthogonal matrices so attractive in numerics.* ⟧

Proof: *Exercise.* □

Remark 4.41 (multiplication by $\mathbf{Q} \in \mathcal{O}_n$ is numerically stable) *Consider relative errors:*

$$\frac{\|\mathbf{Q}(\mathbf{x}+\Delta\mathbf{x}) - \mathbf{Qx}\|_2}{\|\mathbf{Qx}\|_2} = \frac{\|\mathbf{Q}\Delta\mathbf{x}\|_2}{\|\mathbf{Qx}\|_2} = \underbrace{1}_{\substack{\text{“amplification” factor} \\ \text{for rel. error}}} \cdot \frac{\|\Delta\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$$

Exercise 4.42 *Check that the Gram-Schmidt orthogonalization process for a matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ produces an upper triangular matrix $\mathbf{R}$ and an orthogonal matrix $\mathbf{Q}$ with $\mathbf{AR} = \mathbf{Q}$. Hence, for invertible $\mathbf{R}$ (i.e., invertible $\mathbf{A}$), Gram-Schmidt provides a $\mathbf{QR}$-factorization of $\mathbf{A}$.* ∎

4.6.2 QR-factorization by Householder reflections

Definition 4.43 *Let $m \geq n$.*

*(i) $\mathbf{R} \in \mathbb{R}^{m\times n}$ is a **generalized upper triangular matrix** if $r_{ij} = 0 \;\forall i > j$, i.e.,*

$$\mathbf{R} = \begin{pmatrix} \widetilde{\mathbf{R}} \\ 0 \end{pmatrix} \quad \text{with } \widetilde{R} \in \mathcal{U}_m.$$

$\mathcal{U}_m$ denotes the set of $m \times m$ upper triangular matrices.

[6] In other words: $\mathcal{O}_n$ is a group with respect to matrix multiplication.

(ii) A factorization $\mathbf{A} = \mathbf{QR}$ *of a matrix* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *with an orthogonal* $\mathbf{Q} \in \mathcal{O}_m$ *and a generalized upper triangular matrix* $\mathbf{R}$ *is called a* $\mathbf{QR}$-factorization *of* $\mathbf{A}$.

Theorem 4.44 *Let* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *be invertible. Then:* $\mathbf{A}$ *has a* $\mathbf{QR}$*-factorization. It is unique if one fixes the signs of the diagonal entries* r_{ii} *of* $\mathbf{R}$.

Proof: The existence follows from the explicit construction in Alg. 4.49 below. For definiteness' sake, assume that the signs of diagonal entries are fixed to be positive: $r_{ii} > 0$. Let $\mathbf{QR} = \tilde{\mathbf{Q}}\tilde{\mathbf{R}} = \mathbf{A}$ be two QR-factorizations. Since $\mathbf{A}$ is invertible, so is $\mathbf{R}$. Hence, $\mathbf{Q}' := \tilde{\mathbf{Q}}^\top\mathbf{Q} = \tilde{\mathbf{R}}\mathbf{R}^{-1} =: \mathbf{R}'$. We have to $\mathbf{R}'$ is upper triangular (as the product of two upper triangular matrices) and an orthogonal matrix (since $\mathbf{Q}'$ is orthogonal as the product of two orthogonal matrices). Thus, the columns of $\mathbf{R}'$ are orthogonal and by using that $\mathbf{R}'$ is upper triangular, checking inner products of columns of $\mathbf{R}'$ reveals that $\mathbf{R}'$ is a diagonal matrix (e.g., $0 = (\mathbf{R}'_{:,1})^\top(\mathbf{R}'_{:,2}) = r'_{11}r'_{12}$ and $r_{11} \neq 0$). Diagonal matrices that are orthogonal have $+1$ or -1 on the diagonal. One can show (e.g., by induction on n) that $(\mathbf{R})_{ii}^{-1} = 1/r_{ii}$ and it is not difficult to see that the diagonal entries of $(\tilde{\mathbf{R}}\mathbf{R}^{-1})_{ii} = \tilde{\mathbf{R}}_{ii}(\mathbf{R}^{-1})_{ii} = \tilde{r}_{ii}/r_{ii}$. Since, by assumption, $\tilde{r}_{ii}$ and r_{ii} have the same sign, $\tilde{r}_{ii}/r_{ii} = 1$. Hence, $\tilde{\mathbf{Q}}^\top\mathbf{Q} = \mathbf{Q}' = \tilde{\mathbf{R}}\mathbf{R}^{-1} = \mathrm{I}$. That is, $\tilde{\mathbf{Q}} = \mathbf{Q}$ and $\tilde{\mathbf{R}} = \mathbf{R}$. □

The $\mathbf{QR}$-factorization of $\mathbf{A}$ is schematically obtained as follows:

$$\mathbf{A} =: \mathbf{A}^{(0)} = \begin{pmatrix} * & \dots & * \\ \vdots & & \vdots \\ * & \dots & * \end{pmatrix} \xrightarrow{\mathbf{Q}_1\in\mathcal{O}_n} \mathbf{Q}_1\mathbf{A}^{(0)} =: A^{(1)} = \begin{pmatrix} * & \dots & \dots & * \\ 0 & * & \dots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \dots & * \end{pmatrix}$$

$$\mathbf{A}^{(1)} = \begin{pmatrix} * & \dots & \dots & * \\ 0 & * & \dots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \dots & * \end{pmatrix} \xrightarrow{\mathbf{Q}_2\in\mathcal{O}_n} \mathbf{Q}_2A^{(1)} =: \mathbf{A}^{(2)} = \begin{pmatrix} * & \dots & \dots & \dots & * \\ 0 & * & \dots & \dots & * \\ \vdots & 0 & * & \dots & * \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & * & \dots & * \end{pmatrix}$$

$$\mathbf{A}^{(2)} = \begin{pmatrix} * & \dots & \dots & \dots & * \\ 0 & * & \dots & \dots & * \\ \vdots & 0 & * & \dots & * \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & * & \dots & * \end{pmatrix} \xrightarrow{\mathbf{Q}_3\in\mathcal{O}_n} \dots \xrightarrow{Q_{n-1}\in\mathcal{O}_n} \mathbf{A}^{(n-1)} = \begin{pmatrix} * & \dots & \dots & * \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & * \end{pmatrix}$$

Then: $\mathbf{Q}_{n-1}\mathbf{Q}_{n-1}\dots\mathbf{Q}_1\mathbf{A} = \mathbf{R}$.
That is, the sought $\mathbf{QR}$-factorization is $\mathbf{A} = \mathbf{Q}_1^\top\dots\mathbf{Q}_{n-1}^\top\mathbf{R}$.
The $\mathbf{Q}_i$ are constructed using so-called *Householder reflections*, which are "elementary" orthogonal transformations:

Definition 4.45 (Householder reflections) *Given* $\mathbf{v} \in \mathbb{R}^n$ *with* $\|\mathbf{v}\|_2 = 1$ *the matrix* $\mathbf{H} = I - 2\mathbf{v}\mathbf{v}^\top$ *is called the induced* Householder reflection.

Lemma 4.46 (properties of Householder reflections) *Let* $\mathbf{v} \in \mathbb{R}^n$ *with* $\|\mathbf{v}\|_2 = 1$. *Then the matrix* $\mathbf{H} = I - 2\mathbf{v}\mathbf{v}^\top$ *satisfies:*

(i) $\mathbf{H}$ *is symmetric, i.e.,* $\mathbf{H}^\top = \mathbf{H}$*)*

(ii) $\mathbf{H}$ *is an involution (*$\mathbf{H}^2 = I$*)*

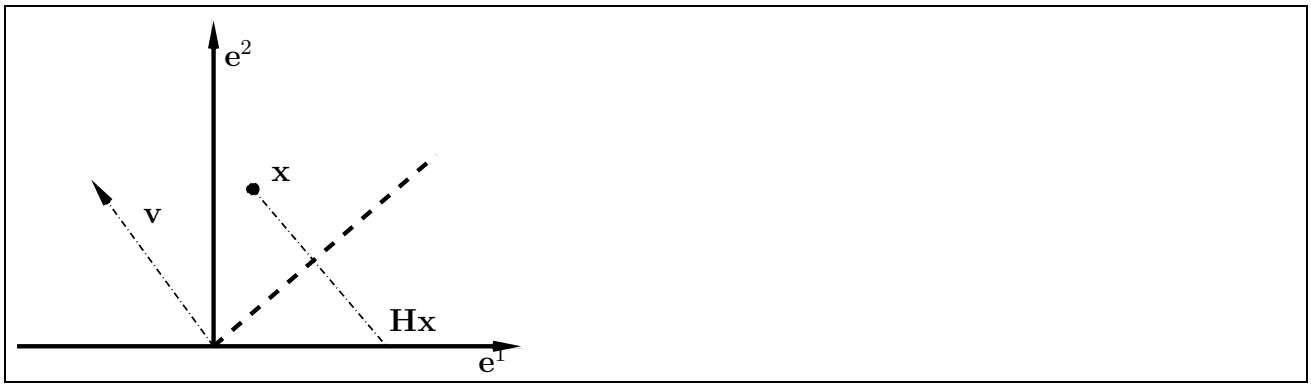

Figure 4.6: The Householder $\mathbf{H}$ with $\mathbf{Hx} \parallel \mathbf{e}^1$; cf. proof of Lemma 4.47

(iii) $\mathbf{H}$ *is orthogonal (*$\mathbf{H}^\top\mathbf{H} = I$*)*

Proof: Exercise.
The geometric interpretation of $\mathbf{H}$ is that the linear map represented by $\mathbf{H}$ is a reflection at the hyperplane $\{\mathbf{x} \in \mathbb{R}^n | \mathbf{v}^\top\mathbf{x} = 0\}$. □

Lemma 4.47 *Let* $\mathbf{x} \in \mathbb{R}^n \setminus \{0\}$ *and* $\mathbf{e}^1 = (1, 0, \ldots, 0)^\top \in \mathbb{R}^n$. *Then* $\exists\ \mathbf{Q} \in \mathcal{O}_n$ *with* $\mathbf{Qx} \in \operatorname{span}\{\mathbf{e}_1\}$. *In particular:*

(i) if $\mathbf{x} \parallel \mathbf{e}^1$, *then* $\mathbf{Q} := I$

(ii) if $\mathbf{x} \not\parallel \mathbf{e}^1$, *then set* $\lambda = \operatorname{sign} \mathbf{x}_1 \|\mathbf{x}\|_2$. *Then* $\mathbf{H} = I - 2\mathbf{v}\mathbf{v}^\top$ *with* $\mathbf{v} = \frac{\mathbf{x}+\lambda\mathbf{e}^1}{\|\mathbf{x}+\lambda\mathbf{e}^1\|_2}$ *has the desired property* $\mathbf{Hx} = -\lambda\mathbf{e}^1$.

Proof:

(i) ✓

(ii)

$$\mathbf{x} + \lambda\mathbf{e}^1 = \begin{pmatrix} \mathbf{x}_1 + (\operatorname{sign}\mathbf{x}_1)\|\mathbf{x}\|_2 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{pmatrix}$$

$$\|\mathbf{x} + \lambda\mathbf{e}^1\|_2^2 = \big(|\mathbf{x}_1| + \|\mathbf{x}\|_2\big)^2 + \sum_{i=2}^{n} \mathbf{x}_i^2 = 2\|\mathbf{x}\|_2^2 + 2|\mathbf{x}_1|\|\mathbf{x}\|_2 \neq 0$$

$$\big(I - 2\mathbf{v}\mathbf{v}^\top\big)\mathbf{x} = \mathbf{x} - 2\frac{\mathbf{x} + \lambda\mathbf{e}^1}{2\|\mathbf{x}\|_2^2 + 2|\mathbf{x}_1|\|\mathbf{x}\|_2} \underbrace{\Big(\mathbf{x} + \operatorname{sign}\mathbf{x}_1\|\mathbf{x}\|_2\mathbf{e}^1\Big)^\top\mathbf{x}}_{\|\mathbf{x}\|_2^2 + |\mathbf{x}_1|\|\mathbf{x}\|_2} = -\lambda\mathbf{e}^1$$

□

Remark 4.48 (choice of λ) *Householder reflections* $\mathbf{H}$ *with* $\mathbf{Hx} \in \operatorname{span}\{\mathbf{e}^1\}$ *are not unique. For example,* $\mathbf{v} = \frac{\mathbf{x}+\lambda\mathbf{e}^1}{\|\mathbf{x}+\lambda\mathbf{e}^1\|_2}$ *with* $\lambda = -\big(\operatorname{sign}|\mathbf{x}_1|\big)\|\mathbf{x}\|_2$ *is also possible. This choice, however, is* numerically unstable *if* $\mathbf{x}$ *and* $\mathbf{e}^1$ *are nearly parallel, i.e.,* $|\mathbf{x}_1| \approx \|\mathbf{x}\|_2$*. Then cancellation occurs when computing* $\mathbf{x} + \lambda\mathbf{e}^1$*.* ∎

Algorithm 4.49 (Householder QR-factorization) *Input:* $\mathbf{A} \in \mathbb{R}^{m\times n}$, $m \geq n$, $\operatorname{rank}(\mathbf{A}) = n$
Output: factorization $\mathbf{A} = \mathbf{QR}$ *with* $\mathbf{Q} \in \mathcal{O}_m$ *and* $\mathbf{R} \in \mathbb{R}^{m\times n}$ *generalized upper triangular matrix.* $\mathbf{Q}$ *is given implicitly as* $\mathbf{Q}^{-1} = \mathbf{Q}_{n-1}\cdots\mathbf{Q}_1$ ⟦ *note:* $\mathbf{Q} = \mathbf{Q}_1\cdots\mathbf{Q}_{n-1}$ *since the* $\mathbf{Q}_i$ *are symmetric, i.e.,* $\mathbf{Q}_i^\top = \mathbf{Q}_i$ ⟧

- *set* $\mathbf{A}^{(0)} := \mathbf{A}$ *and select* $\mathbf{Q}_1$ *as a Householder reflection s.t.* $\mathbf{QA}^{(0)}_{:,1} \parallel \mathbf{e}^1 \in \mathbb{R}^m$
- *"Householder step":*

$$\mathbf{A}^{(1)} \;:=\; \mathbf{Q}_1\mathbf{A}^{(0)} = \begin{pmatrix} a^{(1)}_{11} & a^{(1)}_{12} & \dots & a^{(1)}_{1n} \\ 0 & a^{(1)}_{22} & \dots & a^{(1)}_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & a^{(1)}_{n2} & \dots & a^{(1)}_{nn} \end{pmatrix}$$

- *select* $\widetilde{\mathbf{Q}}_1$ *as a Householder reflection s.t.* $\widetilde{\mathbf{Q}}_1\mathbf{A}^{(1)}_{[2:m],2} \parallel \mathbf{e}^1 \in \mathbb{R}^{m-1}$
- *set*

$$\mathbf{Q}_2 = \left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & \widetilde{\mathbf{Q}_1} \\ & \end{array}\right)$$

- *"Householder step":*

$$\begin{pmatrix} a^{(1)}_{11} & a^{(1)}_{12} & \dots & a^{(1)}_{1n} \\ 0 & a^{(1)}_{22} & \dots & a^{(1)}_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & a^{(1)}_{n2} & \dots & a^{(1)}_{nn} \end{pmatrix} = \mathbf{A}^{(1)} \;\longrightarrow \mathbf{Q}_2\mathbf{A}^{(1)} =: \mathbf{A}^{(2)} = \begin{pmatrix} a^{(1)}_{11} & \dots & \dots & \dots & a^{(1)}_{1n} \\ 0 & a^{(2)}_{22} & \dots & \dots & a^{(2)}_{2n} \\ \vdots & 0 & a^{(2)}_{33} & \dots & a^{(2)}_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & a^{(2)}_{n3} & \dots & a^{(2)}_{nn} \end{pmatrix}$$

- *analogously, the next steps are:*

$$\begin{pmatrix} a_{11}^{(1)} & \dots & \dots & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \dots & \dots & a_{2n}^{(2)} \\ \vdots & 0 & a_{33}^{(2)} & \dots & a_{3n}^{(2)} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & a_{n3}^{(2)} & \dots & a_{nn}^{(2)} \end{pmatrix} = \mathbf{A}^{(2)} \longrightarrow \left(\begin{array}{cc|c} 1 & 0 & \\ 0 & 1 & \\ \hline & & \widetilde{\mathbf{Q}_2} \end{array}\right) \mathbf{A}^{(2)} \longrightarrow \dots$$

$$\dots \longrightarrow \mathbf{A}^{(n)} = \begin{pmatrix} * & \dots & \dots & * \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & * \\ 0 & \dots & \dots & 0 \\ \vdots & & & \vdots \\ 0 & \dots & \dots & 0 \end{pmatrix}$$

Remark 4.50 *(i) see literature (e.g., the book by Golub–van Loan) for a precise formulation*

(ii) Algorithm 4.49 does not stop prematurely since $\operatorname{rank} A = n$ ⟦ *if a column* $(a_{k+1,k}^{(k)}, \dots, a_{k+1,n}^{(k)})^\top$ *is zero, then* $\mathbf{A}$ *cannot have full column rank n!* ⟧

(iii) <u>*cost:*</u> *For* $\mathbf{A} \in \mathbb{R}^{n\times n}$ *the algorithm requires* $\frac{4}{3}n^3$ *arithmetic operations → twice as expense as* **LU***-factorization and 4 times as expensive as a Cholesky decomposition.*

(iv) <u>*storage:*</u> $\mathbf{Q}$ *is typically not stored explicitly but merely the Householder vectors are stored. One possibility of storing the factorization in place of* $\mathbf{A}$*:*

- *store the entries* r_{ij} , $j > i$ *in place of* a_{ij}
- *store the k-th Householder vector* $\mathbf{w}_k \in \mathbb{R}^{m+j-k}$ *in place of* a_{ik} , $i \geq k$
- *store the* r_{ii} *separately*

Remark 4.51 *Although the* **QR***-factorization is more expensive than the* **LU***-factorization, it is employed for its better numerical stability properties if the condition number of* $\mathbf{A}$ *is large.* ∎

slide 12

4.6.3 QR-factorization with pivoting

Analogously to **LU**-factorizations with pivoting one can perform **QR**-factorizations with pivoting by constructing factorizations $\mathbf{QR} = \mathbf{AP}$ for a permutation matrix $\mathbf{P}$. This is useful, for example, to treat the case when $m \geq n$ and $\operatorname{rank} \mathbf{A} < n$ ("rank-deficient case").

Procedure:

$$\mathbf{A}^{(0)} \xrightarrow[\mathbf{A}^{(0)} \text{ with the largest } \|\cdot\|_2 \text{ norm to the first column}]{\mathbf{P}_1 = \text{ permutation matrix that moves the column of}} \widetilde{\mathbf{A}}^{(1)} := \mathbf{A}^{(0)}\mathbf{P}_1 \xrightarrow{\text{Householder}} \mathbf{A}^{(1)} := \mathbf{Q}_1\widetilde{\mathbf{A}}^{(1)}$$

$$\mathbf{A}^{(1)} \xrightarrow[p\geq 2 \text{ and } \|\mathbf{A}^{(1)}_{[2:n],p}\|_2 = \max_{i\geq 2}\|\mathbf{A}^{(1)}_{[2:n],i}\|_2]{\mathbf{P}_2\text{: exchange columns 2 and } p \text{ where}} \widetilde{\mathbf{A}}^{(2)} := \mathbf{A}^{(1)}\mathbf{P}_2 \xrightarrow{\text{Householder}} \mathbf{A}^{(2)} := \mathbf{Q}_2\widetilde{\mathbf{A}}^{(2)}$$

$$\mathbf{A}^{(2)} \longrightarrow \ldots \longrightarrow \mathbf{A}^{(k)} = \begin{pmatrix} * & \cdots & \cdots & \cdots & \cdots & \cdots & * \\ 0 & \ddots & & & & & \vdots \\ \vdots & \ddots & \ddots & & & & \vdots \\ \vdots & & \ddots & * & \cdots & \cdots & * \\ \vdots & & & 0 & \cdots & \cdots & 0 \\ \vdots & & & & & & \vdots \\ \vdots & & & & & & \vdots \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \end{pmatrix} = \text{final form}$$

termination:

- The procedure termines if the "remaining matrix" $\left(a_{ij}^{(k)}\right)_{i,j,\geq k+1}$ is the null matrix. Then $\operatorname{rank}\mathbf{A} = k$
- The diagonal entries r_{ii} satisfy $|r_{11}| \geq |r_{22}| \geq \cdots \geq |r_{kk}| > 0$ (exercise: why?). If the submatrix $\mathbf{A}^{(k)}_{[k'+1:\text{end}],[k'+1:\text{end}]}$ has small norm, e.g., $\|\mathbf{A}^{(k)}_{[k'+1:\text{end}],[k'+1:\text{end}]}\|_2 \leq \varepsilon_{\text{mach}}\|\mathbf{A}^{(k)}\|_2$ with $\varepsilon_{\text{mach}}$ being on the order of machine precision, then the rank of $\mathbf{A}$ is effectively k'.

4.6.4 Givens rotations

The application of a single Householder reflection affects many entries of the matrix. Sometimes, it is useful to work with orthogonal matrices that introduce zeros in a matrix in more selective way, i.e., affect rather few entries at the same time. *Givens rotations* are then typically employed. We mention that, for *full* matrices, a **QR**-factorization using Givens rotations is (by a factor) more expensive than with Householder reflections.
For $\theta \in [0, 2\pi)$ set $c := \cos\theta$, $s := \sin\theta$. Then the *Givens rotation* $\mathbf{G}(i, j, \theta)$ with $i \neq j$ is

defined as

$$\mathbf{G}(i,j,\theta) := \begin{pmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & & c & & & +s & & \\ & & & & 1 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 1 & & \\ & & & -s & & & c & & \\ & & & & & & & 1 & \\ & & & & & & & & \ddots \end{pmatrix}$$

Geometrically, $\mathbf{G}(i,j,\theta)$ is a rotation by an angle θ in the two-dimensional plane $\text{span}\{e_i, e_j\}$. Thus is an orthogonal matrix. We have

Lemma 4.52 *Given $i \neq j$, $\theta \in [0, 2\pi)$ abbreviate*

$$\widehat{\mathbf{G}} := \begin{pmatrix} c & s \\ -s & c \end{pmatrix}$$

Then:

(i) $\mathbf{G}(i,j,\theta)$ *is orthogonal*

(ii) $\mathbf{AG}$ *differs from* $\mathbf{A}$ *only in columns i and j and these are linear combinations of the columns i, j of* $\mathbf{A}$*:*

$$(\mathbf{AG})(:,[i,j]) = \mathbf{A}(:,[i,j])\widehat{\mathbf{G}}$$

(iii) $\mathbf{G}^\top\mathbf{A}$ *differs from* $\mathbf{A}$ *only in rows i and j and these are linear combinations of the rows i, j of* $\mathbf{A}$*:*

$$(\mathbf{G}^\top\mathbf{A})([i,j],:) = \widehat{\mathbf{G}}^\top\mathbf{A}([i,j],:)$$

(iv) Let $i \neq j$ and $i' \neq i$. Then there is a Givens rotation $\mathbf{G}(i,i',\theta)$ such that $(\mathbf{G}^\top\mathbf{A})_{ij} = 0$.

Proof: We only show (iv). For that, we note

$$\begin{aligned} (\mathbf{G}^\top\mathbf{A})([i,i'],[i,j]) &= \hat{\mathbf{G}}^\top\mathbf{A}([i,i'],[i,j]) = \begin{pmatrix} c & -s \\ s & c \end{pmatrix}\begin{pmatrix} \mathbf{A}_{ii} & \mathbf{A}_{ij} \\ \mathbf{A}_{i'i} & \mathbf{A}_{i'j} \end{pmatrix} \\ &= \left(\begin{array}{c|c} * & c\mathbf{A}_{ij} - s\mathbf{A}_{i'j} \\ \hline * & * \end{array}\right). \end{aligned}$$

Hence, the requirement $(\mathbf{G}^\top\mathbf{A})_{ij} = 0$ implies that θ should be chosen such that $s\mathbf{A}_{i'j} = c\mathbf{A}_{ij}$.
1. case: $\mathbf{A}_{ij} = 0$: select $c = 1$, $s = 0$.
2. case: $\mathbf{A}_{ij} \neq 0$:select $\theta \in (0,\pi)$ as the solution of $\cot\theta = -\frac{\mathbf{A}_{i'j}}{\mathbf{A}_{ij}}$. □

Lemma 4.52 informs us that one could also compute a **QR**-factorization of **A** using Givens rotations. We sketch the procedure:

$$\mathbf{A} = \begin{pmatrix} * & \cdots & * \\ * & \cdots & * \\ \vdots & & \vdots \\ * & \cdots & * \end{pmatrix} \xrightarrow{\mathbf{G}(1,2)} \begin{pmatrix} * & \cdots & \cdots & * \\ 0 & * & \cdots & * \\ * & * & \cdots & * \\ \vdots & \vdots & & \vdots \\ * & * & \cdots & * \end{pmatrix} \xrightarrow{\mathbf{G}(1,3)} \begin{pmatrix} * & \cdots & \cdots & * \\ 0 & * & \cdots & * \\ 0 & * & & \vdots \\ * & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ * & * & \cdots & * \end{pmatrix}$$

$$\to \cdots \xrightarrow{\mathbf{G}(1,n)} \begin{pmatrix} * & \cdots & \cdots & * \\ 0 & * & \cdots & * \\ 0 & * & & \vdots \\ 0 & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & * & \cdots & * \end{pmatrix} \xrightarrow{\mathbf{G}(2,3)} \begin{pmatrix} * & * & \cdots & * \\ 0 & * & \cdots & * \\ 0 & 0 & & \vdots \\ 0 & * & & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & * & \cdots & * \end{pmatrix} \to \cdots \xrightarrow{\mathbf{G}(n-1,n)} \begin{pmatrix} * & * & \cdots & * \\ 0 & * & \cdots & * \\ 0 & 0 & & \vdots \\ 0 & 0 & & \vdots \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & * \end{pmatrix}$$

The construction of a **QR**-factorization using Givens rotations is more expensive than the one using Householder reflections for full matrices. Givens rotations are typically employed if the matrix has already many zeros that one wishes to preserve by orthogonal transformations as the following example shows.

Example 4.53 *We compute the* **QR***-factorization of an upper Hessenberg matrix.*
(Upper) Hessenberg matrices $\mathbf{A}$ *are matrices with* $\mathbf{A}_{ij} = 0$ *for* $i > j + 1$ *(i.e., upper triangular and one additional subdiagonal may be nonzero). The basic step of the so-called* **QR***-algorithm for the (iterative) computation of eigenvalue of a matrix* $\mathbf{A}$ *is to compute the* **QR***-factorization of* $\mathbf{Q}$ *and then compute the product of these factors in reverse order, i.e.,* $\mathbf{RQ}$*. We show that, if the matrix* $\mathbf{A}$ *is upper Hessenberg, then the product* $\mathbf{RQ}$ *is again upper Hessenberg.*
We compute the matrix $\mathbf{Q}^\top$ *as the product* $\mathbf{Q}^\top = \mathbf{G}(n-1,n)\cdots\mathbf{G}(2,3)\mathbf{G}(1,2)$ *of* $n-1$ *Givens rotation to annihilate the subdiagonal entries of* $\mathbf{A}$*. By construction* $\mathbf{Q}^\top\mathbf{A}$ *is thus upper triangular and is the factor* $\mathbf{R}$*. Next, we multiply from the right by* $\mathbf{Q}$*, i.e., we compute* $(\mathbf{Q}^\top\mathbf{A})\mathbf{Q} = (\mathbf{Q}^\top\mathbf{A})\mathbf{G}(1,2)^\top\mathbf{G}(2,3)^\top\cdots\mathbf{G}(n-1,n)^\top$*. One then checks the multiplication of* $(\mathbf{Q}^\top\mathbf{A})$ *by* $\mathbf{G}(1,2)$ *introduces an additional non-zero term in the* $(2,1)$ *position. The subsequent multiplication by* $\mathbf{G}(2,3)$ *introduces one in the* $(3,2)$ *position. Continuing in this fashion, we see that* $\mathbf{Q}^\top\mathbf{HQ}$ *is an (upper) Hessenberg matrix.* ∎