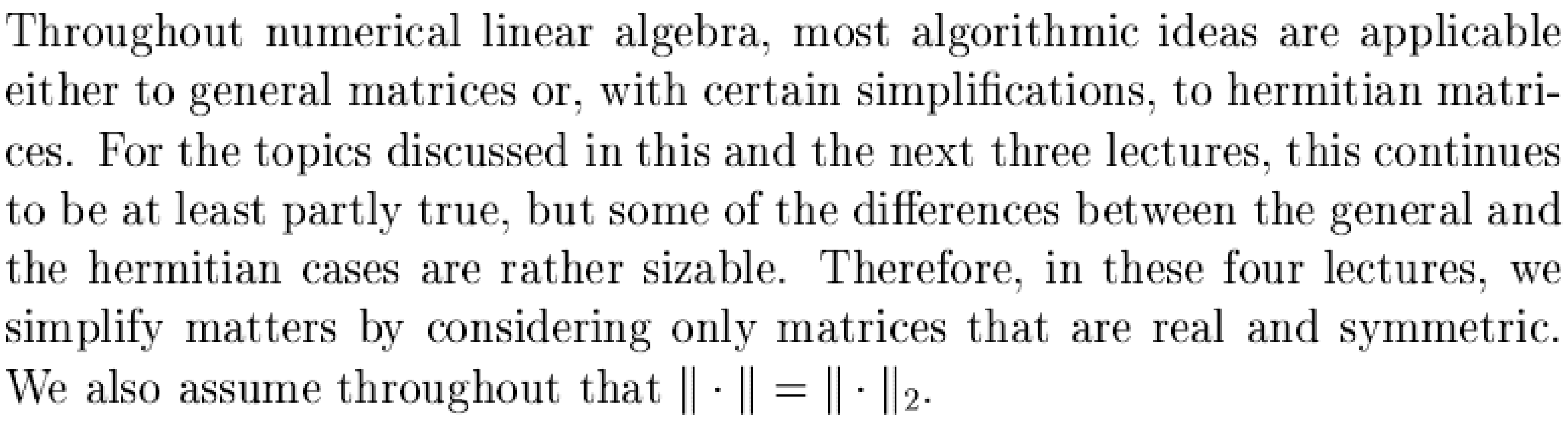
Lecture 30
Other Eigenvalue Algorithms

NLA Reading Group Spring'13

by Ömer Deniz Akyıldız

From Lecture 27…



Throughout numerical linear algebra, most algorithmic ideas are applicable either to general matrices or, with certain simplifications, to hermitian matrices. For the topics discussed in this and the next three lectures, this continues to be at least partly true, but some of the differences between the general and the hermitian cases are rather sizable. Therefore, in these four lectures, we simplify matters by considering only matrices that are real and symmetric. We also assume throughout that $\|\cdot\| = \|\cdot\|_2$.

In these slides, when we write the matrix *A*, we mean real, symmetric, m by m matrix.

Three algorithms for eigenvalue computation

- The Jacobi algorithm (for full matrices)
- The Bisection algorithm (for tridiagonal matrices)
- Divide-and-conquer algorithm (for tridiagonal matrices)

- Note that, general approach is two phase for Hermitian A:
  - First phase: Tridiagonalizing the matrix
  - Second phase: Diagonalizing it

From Lecture 26…

$$\underset{A}{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}} \xrightarrow{Q_1^* \cdot} \underset{Q_1^* A}{\begin{bmatrix} \times & \times & \times & \times & \times \\ \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}} \xrightarrow{\cdot Q_1} \underset{Q_1^* A Q_1}{\begin{bmatrix} \times & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \times & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}}.$$

$$\underset{Q_1^* A Q_1}{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & \times & \times & \times & \times \end{bmatrix}} \xrightarrow{Q_2^* \cdot} \underset{Q_2^* Q_1^* A Q_1}{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}} \xrightarrow{\cdot Q_2} \underset{Q_2^* Q_1^* A Q_1 Q_2}{\begin{bmatrix} \times & \times & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \times & \times & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \times & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ & & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ & & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}}.$$

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}$$

$$\underbrace{Q_{m-2}^* \cdots Q_2^* Q_1^*}_{Q^*} \, A \, \underbrace{Q_1 Q_2 \cdots Q_{m-2}}_{Q} = H.$$

The Jacobi algorithm

- Introduced by Jacobi in 1845.
- Motivation: We know from the Lecture 25, to obtain eigenvalues in dimension 5 or higher, iteration is needed.
  - Eigenvalue problems can be written as rootfinding problems.
  - For $d > 4$, we do not have explicit formulas for roots (so for eigenvalues).
- Why not diagonalize a small submatrix of A, then another, and so on, hoping eventually to converge to a diagonalization of the full matrix?

The Jacobi algorithm

- The idea is based on 2 by 2 submatrices.
- A 2 by 2 real, symmetric matrix can be diagonalized in the form,

$$J^T \begin{bmatrix} a & d \\ d & b \end{bmatrix} J = \begin{bmatrix} \neq 0 & 0 \\ 0 & \neq 0 \end{bmatrix}$$

where *J* is orthogonal. One could take a rotation,

$$J = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$

where $s = \sin\theta$ and $c = \cos\theta$ and $\tan(2\theta) = \frac{2d}{b-a}$

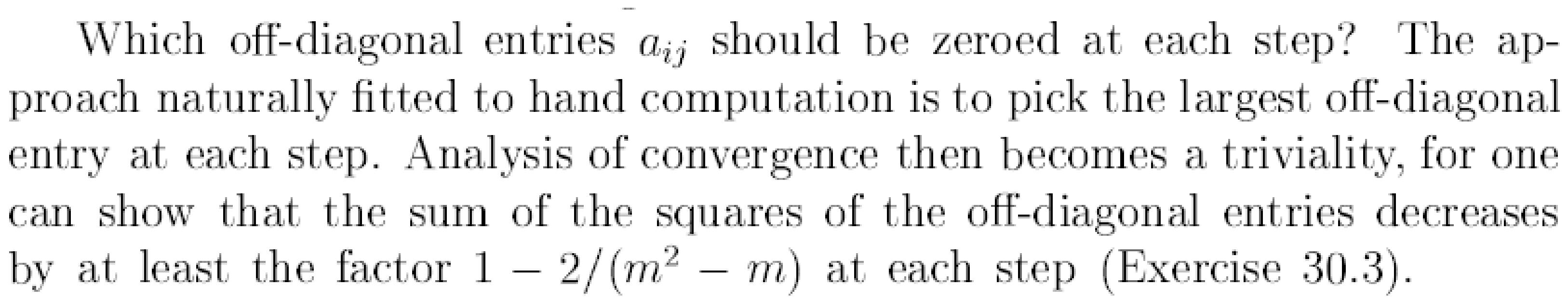
The Jacobi algorithm

- Let *A* be real, symmetric m by m matrix. We form a m by m *J* which is the identity in all but four entries, where it has the form,

$$\begin{array}{cc} \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} & \begin{array}{c} \\ \\ p \\ \\ q \\ \\ \\ \end{array} \\ \begin{array}{ccccccc} & & p & & q & & \end{array} & \end{array}$$

- Applying J^T from left modifies two rows of *A* and applying *J* from the right modifies two columns of *A.*
- At each step, a symmetric pair of zeros introduced into the matrix but previous zeros are destroyed. However, the usual effects is that the magnitudes of these nonzeros shrink steadily.

The Jacobi algorithm



Which off-diagonal entries a_{ij} should be zeroed at each step? The approach naturally fitted to hand computation is to pick the largest off-diagonal entry at each step. Analysis of convergence then becomes a triviality, for one can show that the sum of the squares of the off-diagonal entries decreases by at least the factor $1 - 2/(m^2 - m)$ at each step (Exercise 30.3).

Bisection

- After a symmetric matrix has been tridiagonalized, the natural next step is the Bisection algorithm if one does not want all of the eigenvalues but just a subset of them.

- Bisection can find the largest 10% of the eigenvalues, or the smallest thirty eigenvalues, or all the eigenvalues in the interval [1,2].

- Once the desired eigenvalues are found, the corresponding eigenvectors can be obtained by one step of inverse iteration (Lecture 27).

Bisection

- Since the eigenvalues of a real symmetric matrix are real, we can find them by searching the real line for roots of the polynomial:

$$p(x) = \det(A - xI).$$

Bisection

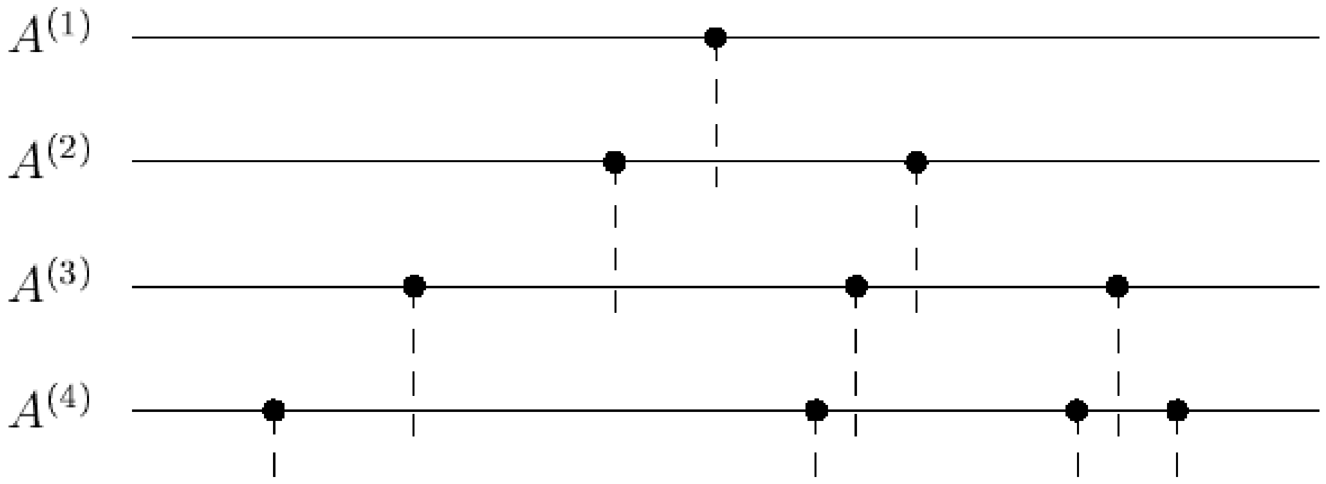
- Given *A* real, symmetric m by m matrix, let $A^{(1)}, \ldots, A^{(m)}$ denote its principal (upper-left) square submatrices of dimensions *1,...,m*.
- Without loss of generality, let us assume *A* is tridiagonal and *irreducible* in the sense that all of its off-diagonal entries are zero.

$$A = \begin{bmatrix} a_1 & b_1 & & & \\ b_1 & a_2 & b_2 & & \\ & b_2 & a_3 & \ddots & \\ & & \ddots & \ddots & b_{m-1} \\ & & & b_{m-1} & a_m \end{bmatrix}, \qquad b_j \neq 0.$$

Bisection

- The eigenvalues of $A^{(k)}$ are distinct; let them be denoted by $\lambda_1^{(k)} < \lambda_2^{(k)} < \cdots < \lambda_k^{(k)}$. These eigenvalues strictly *interlace*:

$$\lambda_j^{(k+1)} < \lambda_j^{(k)} < \lambda_{j+1}^{(k+1)}$$



- It is the interlacing property that makes it possible to count the exact number of eigenvalues in a specified interval.

Bisection

$$A = \begin{bmatrix} 1 & 1 & & \\ 1 & 0 & 1 & \\ & 1 & 2 & 1 \\ & & 1 & -1 \end{bmatrix}.$$

From the numbers

$$\det(A^{(1)}) = 1, \quad \det(A^{(2)}) = -1, \quad \det(A^{(3)}) = -3, \quad \det(A^{(4)}) = 4,$$

we know that $A^{(1)}$ has no negative eigenvalues, $A^{(2)}$ has one negative eigenvalue, $A^{(3)}$ has one negative eigenvalue, and $A^{(4)}$ has two negative eigenvalues. In general, for any symmetric tridiagonal $A \in \mathbb{R}^{m\times m}$, *the number of negative eigenvalues is equal to the number of sign changes in the sequence*

$$1,\ \det(A^{(1)}),\ \det(A^{(2)}), \ldots,\ \det(A^{(m)}), \tag{30.7}$$

which is known as a *Sturm sequence*.

Bisection

One more observation completes the description of the bisection algorithm: for a tridiagonal matrix, the determinants of the matrices $\{A^{(k)}\}$ are related by a three-term recurrence relation. Expanding $\det(A^{(k)})$ by minors with respect to its entries b_{k-1} and a_k in row k gives, from (30.5),

$$\det(A^{(k)}) = a_k \det(A^{(k-1)}) - b_{k-1}^2 \det(A^{(k-2)}). \tag{30.8}$$

Introducing the shift by xI and writing $p^{(k)}(x) = \det(A^{(k)} - xI)$, we get

$$p^{(k)}(x) = (a_k - x)p^{(k-1)}(x) - b_{k-1}^2 p^{(k-2)}(x). \tag{30.9}$$

If we define $p^{(-1)}(x) = 0$ and $p^{(0)}(x) = 1$, then this recurrence is valid for all $k = 1, 2, \ldots, m$.

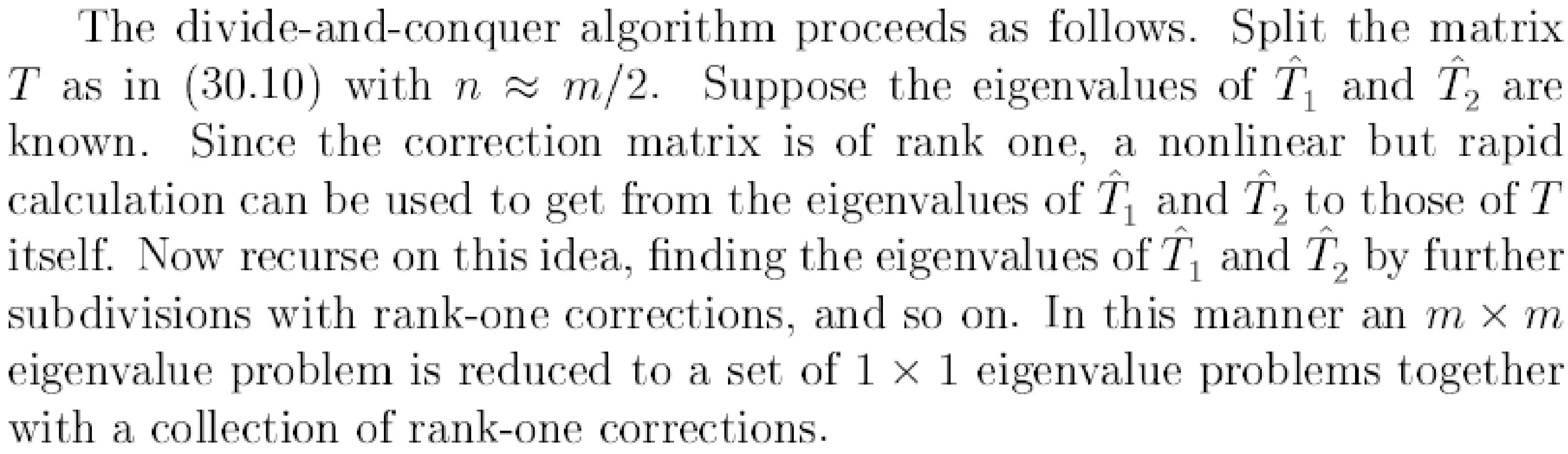
Divide-and-conquer

Let $T \in \mathbb{R}^{m\times m}$ with $m \geq 2$ be symmetric, tridiagonal, and irreducible

$$T = \left[\begin{array}{c|c} T_1 & \\ \quad\quad\quad\quad\beta & \beta \\ \hline \beta & \\ & T_2 \end{array}\right] = \left[\begin{array}{c|c} \hat{T}_1 & \\ \hline & \hat{T}_2 \end{array}\right] + \left[\begin{array}{c|c} \beta & \beta \\ \hline \beta & \beta \end{array}\right].$$

Here is how (30.10) might be expressed in words. *A tridiagonal matrix can be written as the sum of a 2×2 block-diagonal matrix with tridiagonal blocks and a rank-one correction.*

Divide-and-conquer



The divide-and-conquer algorithm proceeds as follows. Split the matrix T as in (30.10) with $n \approx m/2$. Suppose the eigenvalues of $\hat{T}_1$ and $\hat{T}_2$ are known. Since the correction matrix is of rank one, a nonlinear but rapid calculation can be used to get from the eigenvalues of $\hat{T}_1$ and $\hat{T}_2$ to those of T itself. Now recurse on this idea, finding the eigenvalues of $\hat{T}_1$ and $\hat{T}_2$ by further subdivisions with rank-one corrections, and so on. In this manner an $m \times m$ eigenvalue problem is reduced to a set of 1×1 eigenvalue problems together with a collection of rank-one corrections.

Divide-and-conquer

In this process there is one key mathematical point. If the eigenvalues of $\hat{T}_1$ and $\hat{T}_2$ are known, how can those of T be found? To answer this, suppose that diagonalizations

$$\hat{T}_1 = Q_1 D_1 Q_1^T, \qquad \hat{T}_2 = Q_2 D_2 Q_2^T$$

have been computed. Then from (30.10) it follows that we have

$$T = \begin{bmatrix} Q_1 & \\ & Q_2 \end{bmatrix} \left(\begin{bmatrix} D_1 & \\ & D_2 \end{bmatrix} + \beta z z^T \right) \begin{bmatrix} Q_1^T & \\ & Q_2^T \end{bmatrix} \qquad (30.11)$$

with $z^T = (q_1^T, q_2^T)$, where q_1^T is the last row of Q_1 and q_2^T is the first row of Q_2. Since this equation is a similarity transformation, we have reduced the mathematical problem to the problem of finding the eigenvalues of a diagonal matrix plus a rank-one correction.

Divide-and-conquer

To show how this is done, we simplify notation as follows. Suppose we wish to find the eigenvalues of $D + ww^T$, where $D \in \mathbb{R}^{m\times m}$ is a diagonal matrix

$$f(\lambda) \;=\; 1 + \sum_{j=1}^{m} \frac{w_j^2}{d_j - \lambda}, \qquad (30.12)$$

as illustrated in Figure 30.2. This assertion can be justified by noting that if $(D + ww^T)q = \lambda q$ for some $q \neq 0$, then $(D - \lambda I)q + w(w^T q) = 0$, implying $q + (D - \lambda I)^{-1} w(w^T q) = 0$, that is, $w^T q + w^T (D - \lambda I)^{-1} w(w^T q) = 0$. This amounts to the equation $f(\lambda)(w^T q) = 0$, in which $w^T q$ must be nonzero, for otherwise q would be an eigenvector of D, hence nonzero in only one position, implying $w^T q \neq 0$ after all. We conclude that if q is an eigenvector of $D + ww^T$ with eigenvalue λ, then $f(\lambda)$ must be 0, and the converse follows because the form of $f(\lambda)$ guarantees that it has exactly m zeros. The equation $f(\lambda) = 0$ is known as the *secular equation*.

Divide-and-conquer

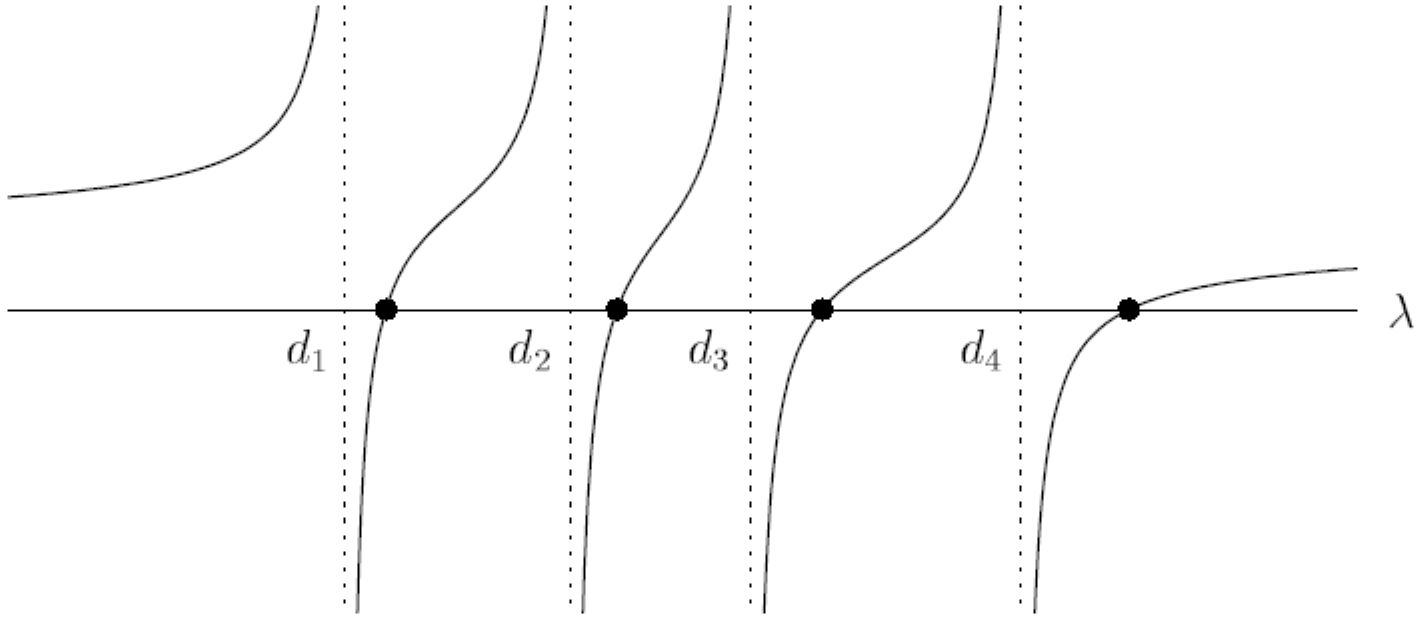


Figure 30.2. *Plot of the function $f(\lambda)$ of (30.12) for a problem of dimension 4. The poles of $f(\lambda)$ are the eigenvalues $\{d_j\}$ of D, and the roots of $f(\lambda)$ (solid dots) are the eigenvalues of $D + ww^T$. The rapid determination of these roots is the basis of each recursive step of the divide-and-conquer algorithm.*

Divide-and-conquer

- If the eigenvalues and eigenvectors required, divide-and-conquer algorithm is more advantageous than QR algorithm.

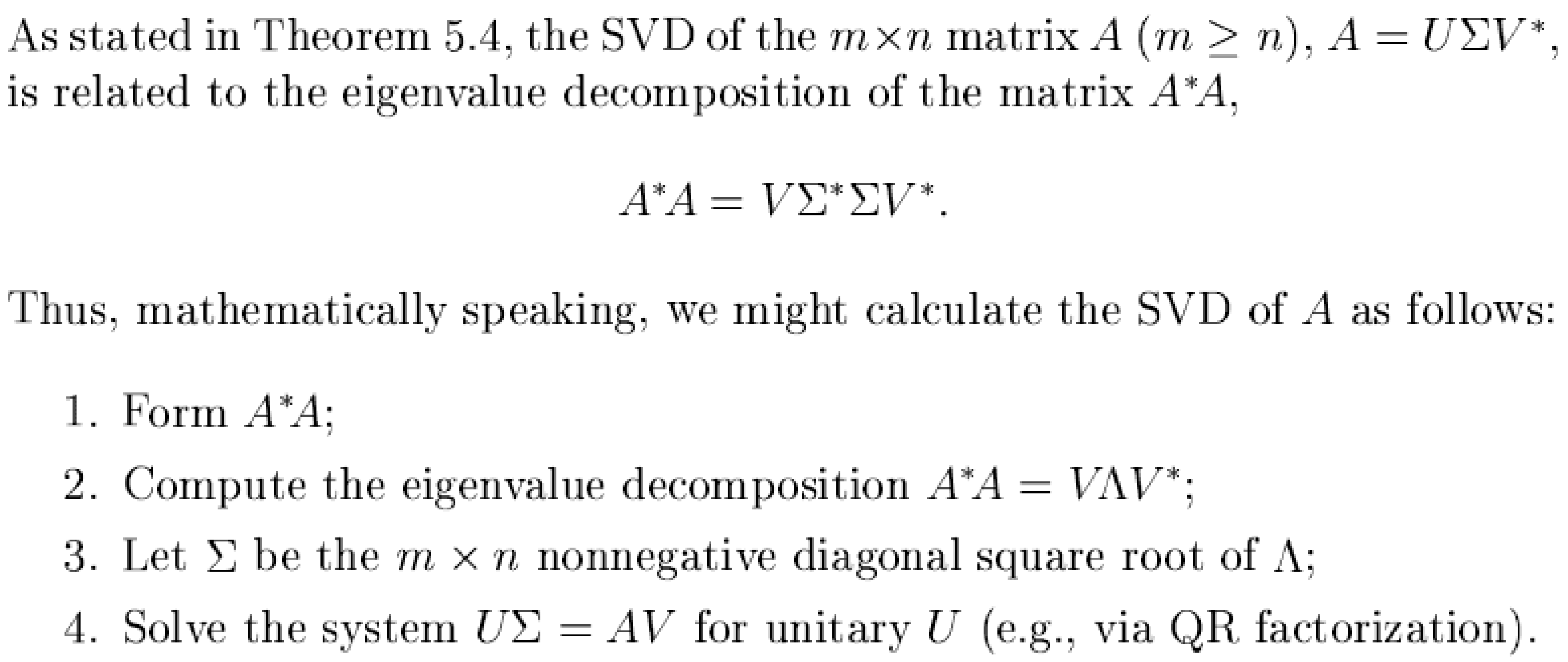
Lecture 31

Computing the SVD

NLA Reading Group Spring’13

by Ömer Deniz Akyıldız

The SVD of *A*



As stated in Theorem 5.4, the SVD of the $m \times n$ matrix A $(m \geq n)$, $A = U\Sigma V^*$, is related to the eigenvalue decomposition of the matrix A^*A,

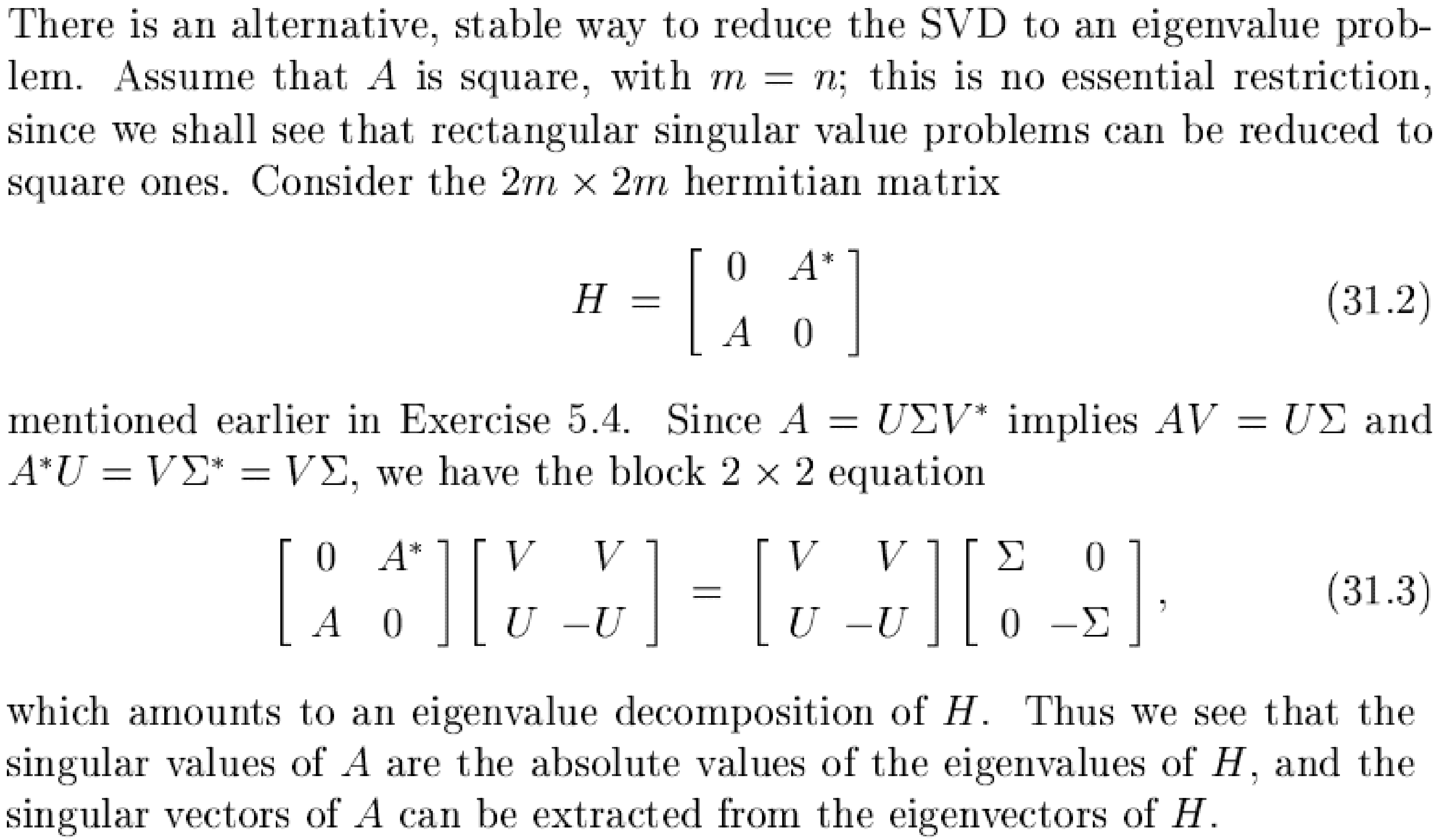
$$A^*A = V\Sigma^*\Sigma V^*.$$

Thus, mathematically speaking, we might calculate the SVD of A as follows:

1. Form A^*A;
2. Compute the eigenvalue decomposition $A^*A = V\Lambda V^*$;
3. Let Σ be the $m \times n$ nonnegative diagonal square root of Λ;
4. Solve the system $U\Sigma = AV$ for unitary U (e.g., via QR factorization).

- This reduction of the SVD problem to a eigenvalue problem is unstable against small perturbations. (Explanation is omitted and details can be seen from “the Book”).

An Alternative Reduction



There is an alternative, stable way to reduce the SVD to an eigenvalue problem. Assume that A is square, with $m = n$; this is no essential restriction, since we shall see that rectangular singular value problems can be reduced to square ones. Consider the $2m \times 2m$ hermitian matrix

$$H = \begin{bmatrix} 0 & A^* \\ A & 0 \end{bmatrix} \tag{31.2}$$

mentioned earlier in Exercise 5.4. Since $A = U\Sigma V^*$ implies $AV = U\Sigma$ and $A^*U = V\Sigma^* = V\Sigma$, we have the block 2×2 equation

$$\begin{bmatrix} 0 & A^* \\ A & 0 \end{bmatrix} \begin{bmatrix} V & V \\ U & -U \end{bmatrix} = \begin{bmatrix} V & V \\ U & -U \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & -\Sigma \end{bmatrix}, \tag{31.3}$$

which amounts to an eigenvalue decomposition of H. Thus we see that the singular values of A are the absolute values of the eigenvalues of H, and the singular vectors of A can be extracted from the eigenvectors of H.

Two-Phase Approach

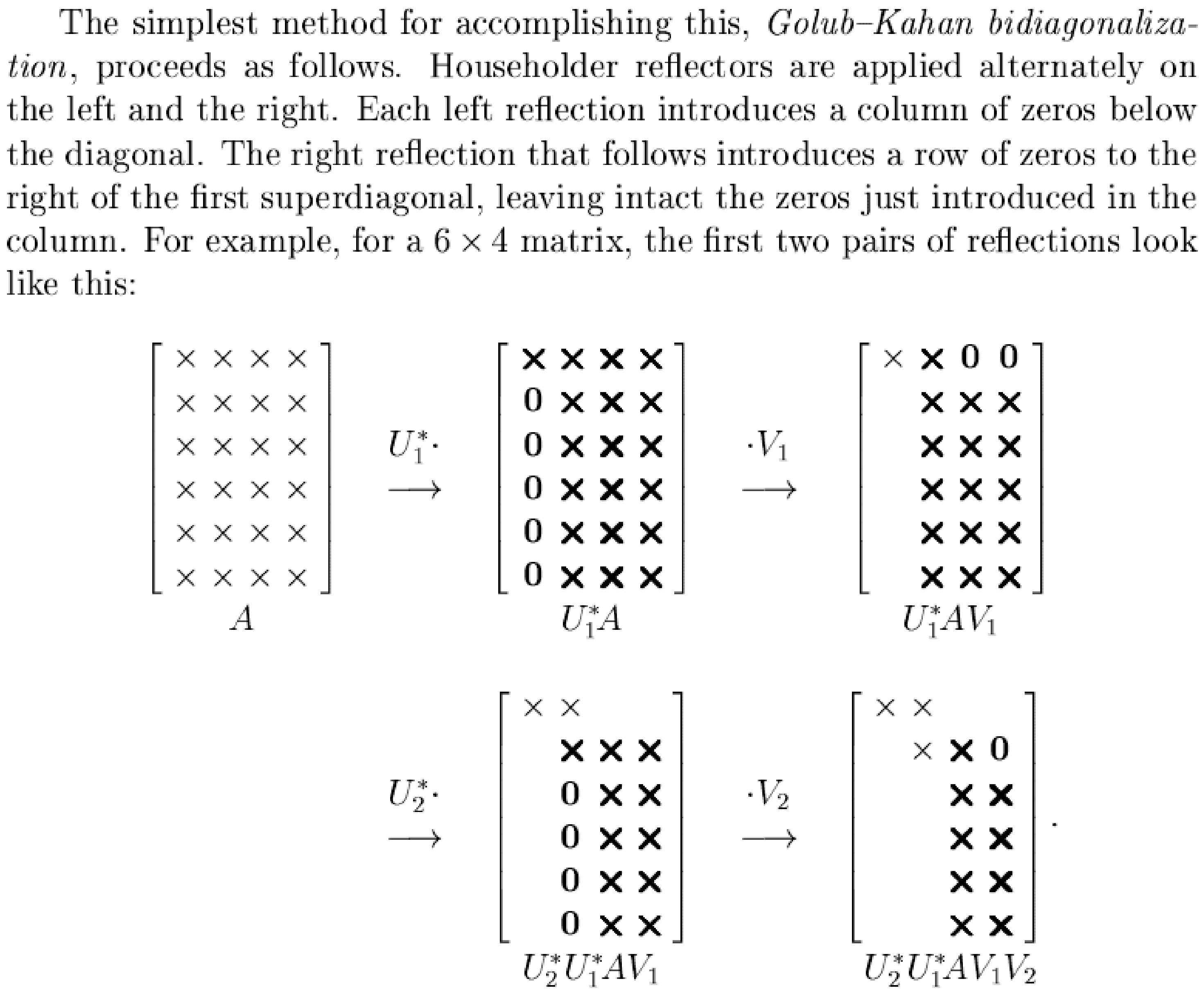
We have seen that hermitian eigenvalue problems are usually solved by a two-phase computation: first reduce the matrix to tridiagonal form, then diagonalize the tridiagonal matrix. Since the work of Golub, Kahan, and others in the 1960s, an analogous two-phase approach has been standard for the SVD. The matrix is brought into bidiagonal form, and then the bidiagonal matrix is diagonalized:

$$\underset{A}{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}} \xrightarrow{\text{Phase 1}} \underset{B}{\begin{bmatrix} \times & \times & & \\ & \times & \times & \\ & & \times & \times \\ & & & \times \\ & & & \\ & & & \end{bmatrix}} \xrightarrow{\text{Phase 2}} \underset{\Sigma}{\begin{bmatrix} \times & & & \\ & \times & & \\ & & \times & \\ & & & \times \\ & & & \\ & & & \end{bmatrix}}.$$

Phase 1: Golub-Kahan Bidiagonalization

In Phase 1 of the SVD computation, we bring A into bidiagonal form by applying distinct unitary operations on the left and right. Note how this differs from the computation of eigenvalues, where the same unitary operations must be applied on both sides so that each step is a similarity transformation. In that case, it was only possible to introduce zeros below the first subdiagonal. Here, we are able to completely triangularize and also introduce zeros above the first superdiagonal.

Phase 1: Golub-Kahan Bidiagonalization

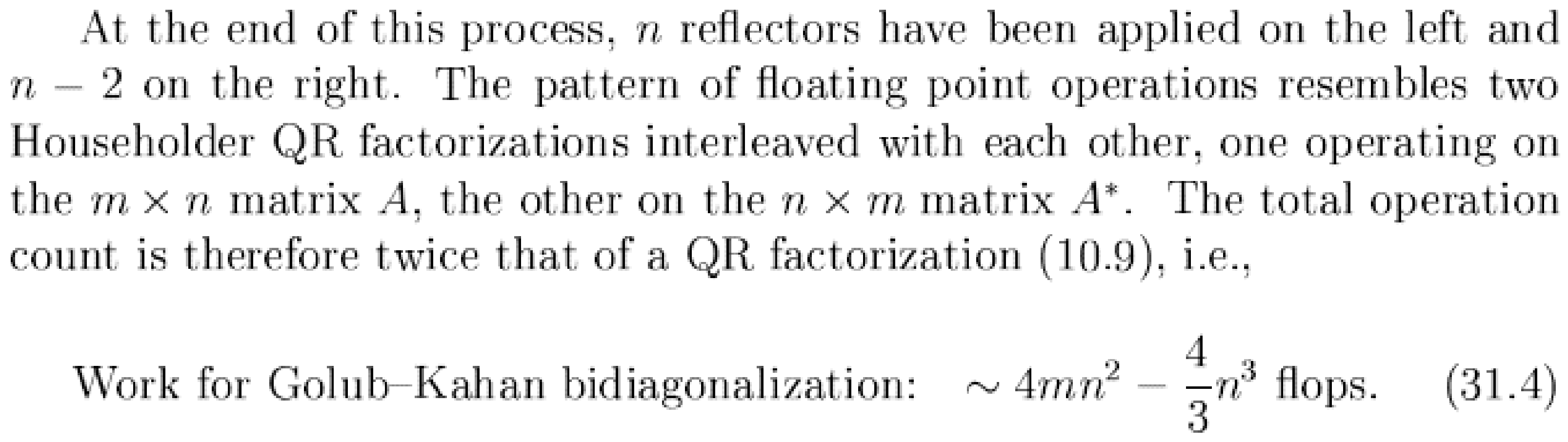


The simplest method for accomplishing this, *Golub–Kahan bidiagonalization*, proceeds as follows. Householder reflectors are applied alternately on the left and the right. Each left reflection introduces a column of zeros below the diagonal. The right reflection that follows introduces a row of zeros to the right of the first superdiagonal, leaving intact the zeros just introduced in the column. For example, for a 6×4 matrix, the first two pairs of reflections look like this:

$$
\underset{A}{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}}
\xrightarrow{U_1^* \cdot}
\underset{U_1^* A}{\begin{bmatrix} \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}}
\xrightarrow{\cdot V_1}
\underset{U_1^* A V_1}{\begin{bmatrix} \times & \boldsymbol{\times} & \mathbf{0} & \mathbf{0} \\ & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}}
$$

$$
\xrightarrow{U_2^* \cdot}
\underset{U_2^* U_1^* A V_1}{\begin{bmatrix} \times & \times & & \\ & \boldsymbol{\times} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \\ & \mathbf{0} & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}}
\xrightarrow{\cdot V_2}
\underset{U_2^* U_1^* A V_1 V_2}{\begin{bmatrix} \times & \times & & \\ & \times & \boldsymbol{\times} & \mathbf{0} \\ & & \boldsymbol{\times} & \boldsymbol{\times} \\ & & \boldsymbol{\times} & \boldsymbol{\times} \\ & & \boldsymbol{\times} & \boldsymbol{\times} \\ & & \boldsymbol{\times} & \boldsymbol{\times} \end{bmatrix}}.
$$

Phase 1: Golub-Kahan Bidiagonalization



At the end of this process, n reflectors have been applied on the left and $n-2$ on the right. The pattern of floating point operations resembles two Householder QR factorizations interleaved with each other, one operating on the $m \times n$ matrix A, the other on the $n \times m$ matrix A^*. The total operation count is therefore twice that of a QR factorization (10.9), i.e.,

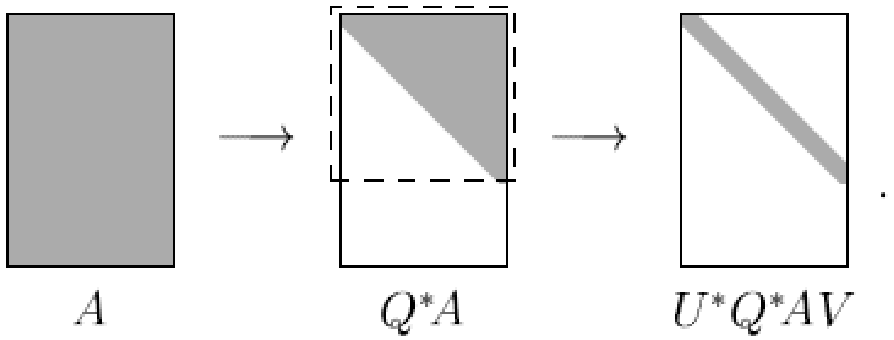
$$\text{Work for Golub–Kahan bidiagonalization:} \quad \sim 4mn^2 - \frac{4}{3}n^3 \text{ flops.} \qquad (31.4)$$

- There are faster methods.

Faster Phase 1

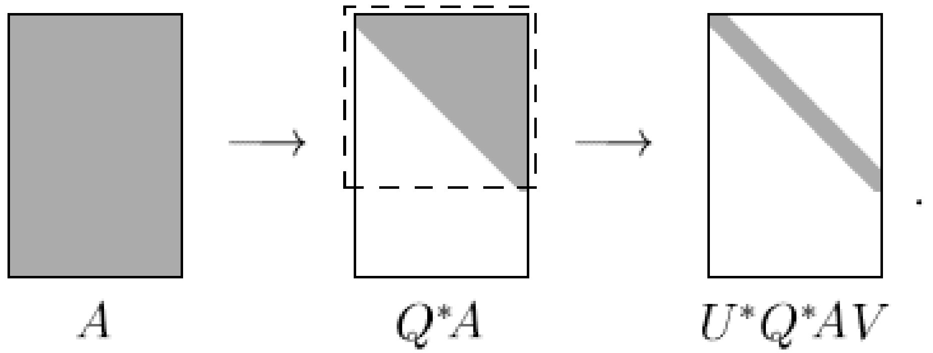
For $m \gg n$, this operation count is unnecessarily large. A single QR factorization would introduce zeros everywhere below the diagonal, and for $m \gg n$, these are the great majority of the zeros that are needed. Yet the operation count for the Golub–Kahan method is twice as high. This observation suggests an alternative method for bidiagonalization with $m \gg n$, first proposed by Lawson and Hanson and later developed by Chan. The idea, *LHC bidiagonalization*, is illustrated as follows:

Lawson–Hanson–Chan bidiagonalization



Faster Phase 1

Lawson–Hanson–Chan bidiagonalization



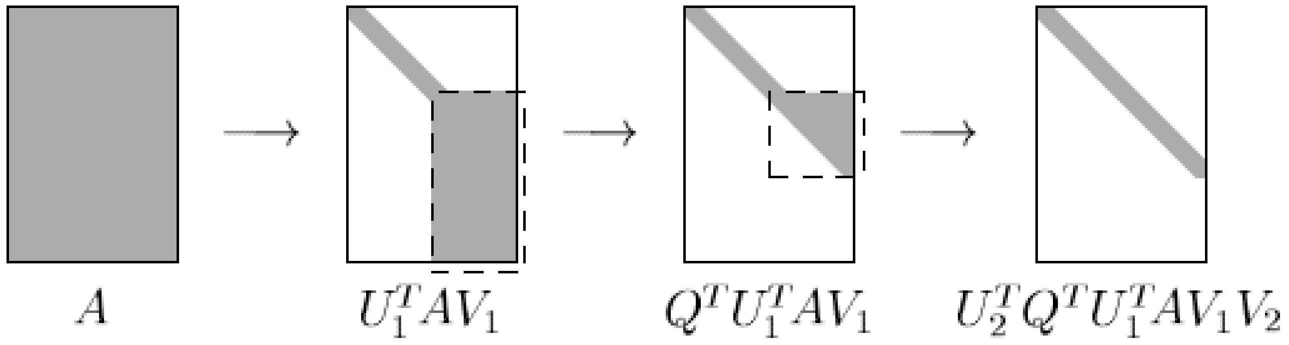
We begin by computing the QR factorization $A = QR$. Then we compute the Golub–Kahan bidiagonalization $B = U^*RV$ of R. The QR factorization requires $2mn^2 - \frac{2}{3}n^3$ flops (10.9), and the Golub–Kahan procedure, which now only has to operate on the upper $n \times n$ submatrix, requires $\frac{8}{3}n^3$ flops. The total operation count is

$$\text{Work for LHC bidiagonalization:} \quad \sim 2mn^2 + 2n^3 \text{ flops.} \tag{31.5}$$

Faster Phase 1

The LHC procedure is advantageous only when $m > \frac{5}{3}n$, but the idea can be generalized so as to realize a saving for any $m > n$. The trick is to apply the QR factorization not at the beginning of the computation, but at a suitable point in the middle. This is advantageous because in the Golub–Kahan process, a matrix with $m > n$ becomes skinnier as the bidiagonalization proceeds. If the initial aspect ratio is, say, $m/n = 3/2$, it will steadily grow to 5/3 and 2 and beyond. After step k, the aspect ratio of the remaining matrix is $(m-k)/(n-k)$, and when this figure gets sufficiently large, it makes sense to perform a QR factorization to reduce the problem to a square matrix.

Three-step bidiagonalization



Phase 2

In Phase 2 of the computation of the SVD, the SVD of the bidiagonal matrix B is determined. From the 1960s to the 1990s, the standard algorithm for this was a variant of the QR algorithm. More recently, divide-and-conquer algorithms have also become competitive, and in the future, they are likely to become the standard. We shall not give details.