

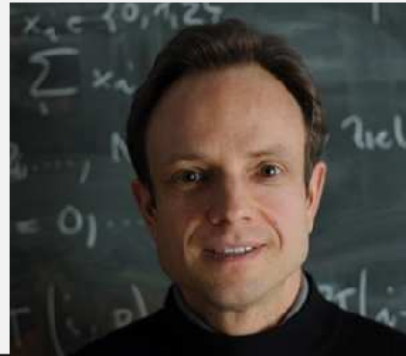
Course Video

Section 3.3.3.2 , Section 3.3.3.4: Computation of QR-Decomposition, QR-Decomposition in EIGEN

Prof. R. Hiptmair, SAM, ETH Zurich

Date: September 23, 2020

(C) Seminar für Angewandte Mathematik, ETH Zürich



Prerequisites.

- From LA: Orthogonal matrices

Dependencies. Section 3.3.3.1, Section 1.2.1

Duration: 55 minutes

Note: Possible minor *mismatch of video and tablet notes!*

[Corrections and updates can be incorporated into tablet notes only]

III. Direct Methods for Linear Least Squares Problems

3.3. Orthogonal Transformation Methods

3.3.3. QR-Decomposition

$$A \in \mathbb{R}^{m,n} : A = QR, \quad Q \in \mathbb{R}^{m,m} \text{ orthogonal, } R \in \mathbb{R}^{m,n} \text{ upper triangular}$$

3.3.3.2. Computation of QR-Decompositions

LA : Gram-Schmidt : unstable, affected by roundoff

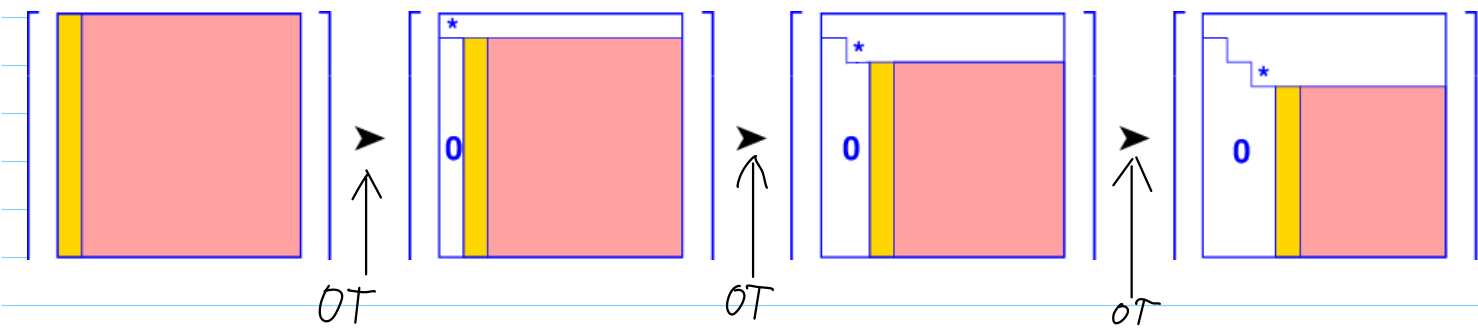
Idea : similar to GE

Idea: find simple unitary/orthogonal (row) transformations rendering certain matrix elements zero:



$$Q \begin{pmatrix} \square \\ \square \\ \square \end{pmatrix} = \begin{bmatrix} \square \\ 0 \\ \square \end{bmatrix} \quad \text{with } Q^H = Q^{-1}.$$

2



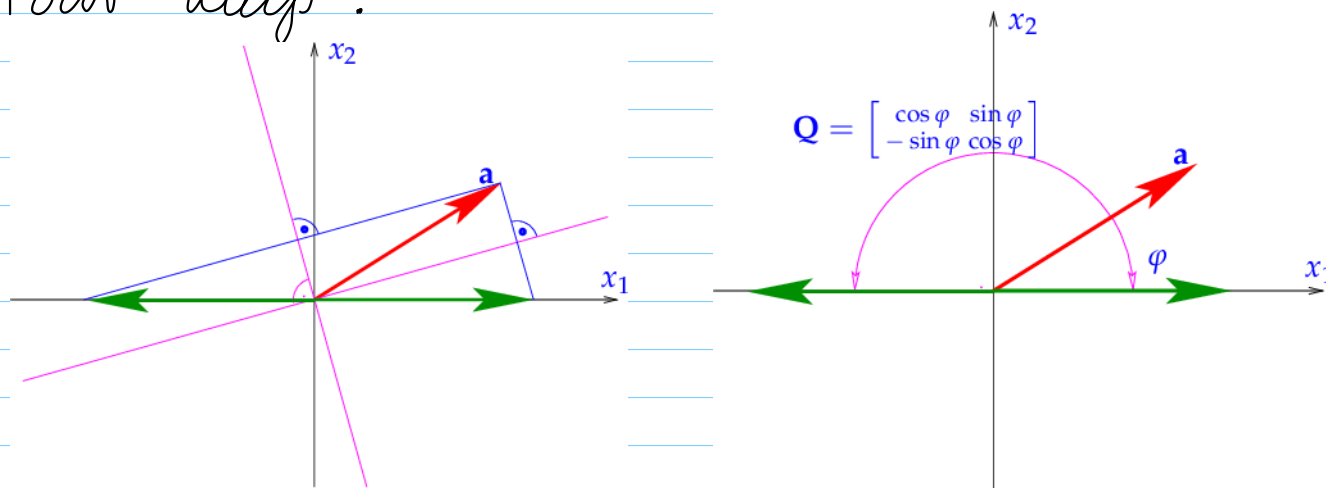
Corollary 3.3.3.9. Composition of orthogonal transformations

The product of two orthogonal/unitary matrices of the same size is again orthogonal/unitary.

Ex 3.3.3.9: 2D annihilating OT
 $\underline{a} = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} \xrightarrow{Q} \begin{bmatrix} \pm \|\underline{a}\| \\ 0 \end{bmatrix} \parallel \underline{e}_1$

$$Q [\underline{a}, \underline{b}] = \begin{bmatrix} * & * \\ 0 & * \end{bmatrix}$$

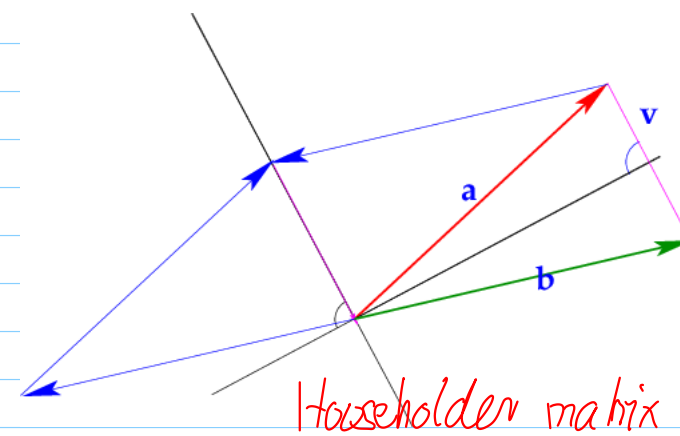
Form recap:



reflections at angle bisector,

rotations turning $\underline{a}$ onto x_1 -axis.

§ 3.3.3.10: Householder reflections



$$\underline{a}, \underline{b} \in \mathbb{R}^n, \|\underline{a}\|_2 = \|\underline{b}\|_2$$

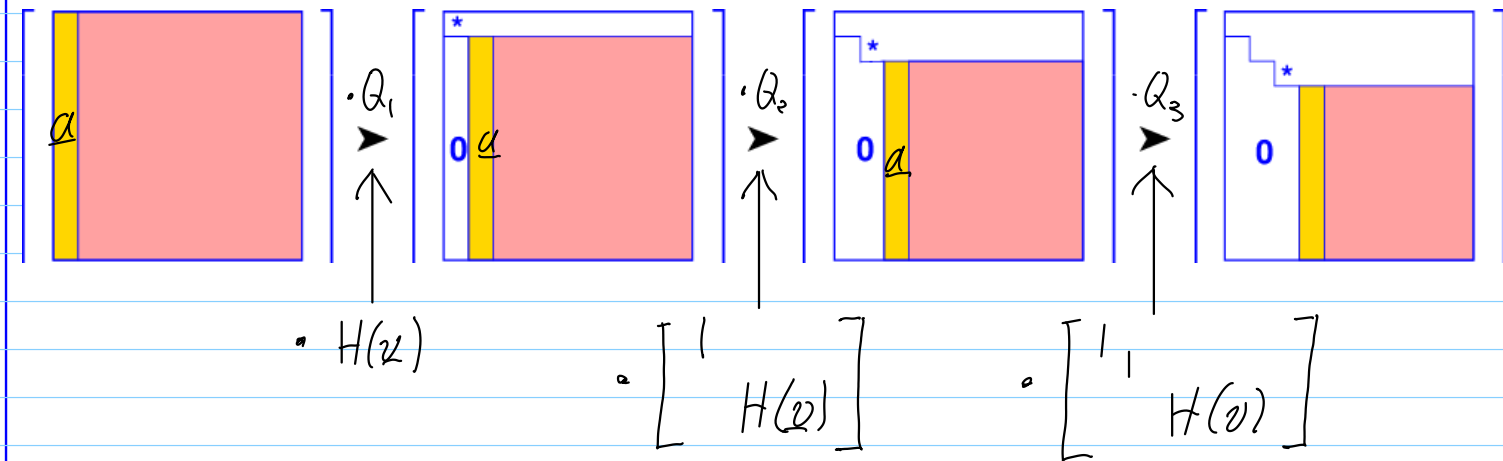
$$\underline{v} := \underline{b} - \underline{a} \perp \underline{b} + \underline{a}$$

$$\underline{b} = \underline{a} - \underline{v} = \underline{a} - \underline{v} \cdot 2 \frac{\underline{v}^T \underline{a}}{\underline{v}^T \underline{v}} = H(\underline{v}) \underline{a}$$

$$H(\underline{v}) = \left(\underline{I} - 2 \frac{\underline{v} \underline{v}^T}{\underline{v}^T \underline{v}} \right)$$

orthogonal, symmetric

Special: $\underline{v} = \underline{a} + \text{sgn}(a_1) \|\underline{a}\|_2 \underline{e}_1$
 avoids cancellation, Rem. 3.3.3.13



③

QR-decomposition (QR-factorization) of $A \in \mathbb{C}^{n,n}$: $A = QR$, $Q := Q_1^H \cdots Q_{n-1}^H$ orthogonal matrix, R upper triangular matrix.

Rem.: 3.3.3.12: HH-refl. applicable to any matrix

$$\begin{bmatrix} A \end{bmatrix} = \begin{bmatrix} Q \end{bmatrix} \begin{bmatrix} R \end{bmatrix}, \quad A = QR, \quad Q \in \mathbb{R}^{m,m}, \quad R \in \mathbb{R}^{m,n},$$

Rem.: 3.3.20: Encode HH-tpls.

In practice: only $Q \times$ vector needed

→ sufficient to store Householder matrices

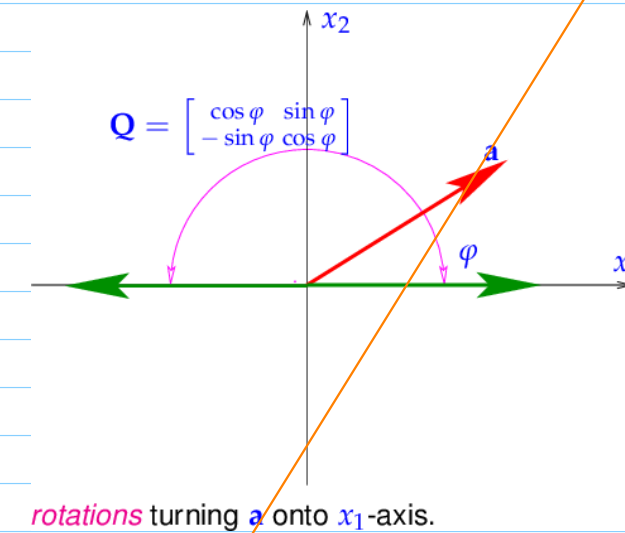
HHM: $H(v) = I - 2vv^T \in \mathbb{R}^m, v \in \mathbb{R}^m, \|v\|_2 = 1$

→ store only unit vector $\underline{v} \in \mathbb{R}^m$

"compute QR-dec." $\hat{=}$ compute R & encode Q

§ 3.3.3.14: Givens rotations

Idea: 2D rotations attack two entries of $\underline{a}$ only



$$\begin{bmatrix} \gamma & \sigma \\ -\sigma & \gamma \end{bmatrix} \begin{bmatrix} a_1 \\ a_k \end{bmatrix} = \begin{bmatrix} * \\ 0 \end{bmatrix}$$

$$\mathbf{G}_{1k}(a_1, a_k) \underline{a} := \begin{bmatrix} \gamma & \cdots & \sigma & \cdots & 0 \\ \vdots & & \vdots & & \vdots \\ -\sigma & \cdots & \gamma & \cdots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_k \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} a_1^{(1)} \\ \vdots \\ 0 \\ \vdots \\ a_n \end{bmatrix}, \text{ if } \begin{matrix} \gamma = \frac{a_1}{\sqrt{|a_1|^2 + |a_k|^2}} \\ \sigma = \frac{a_k}{\sqrt{|a_1|^2 + |a_k|^2}} \end{matrix} \quad (3.3.3.16)$$

orthogonal $\hat{=}$ unchanged

$$\underline{a} := \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \xrightarrow{\mathbf{G}_{12}(a_1, a_2)} \begin{bmatrix} a_1^{(1)} \\ 0 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} \xrightarrow{\mathbf{G}_{13}(a_1^{(1)}, a_3)} \begin{bmatrix} a_1^{(2)} \\ 0 \\ 0 \\ a_4 \\ \vdots \\ a_n \end{bmatrix} \xrightarrow{\mathbf{G}_{14}(a_1^{(2)}, a_4)} \cdots \xrightarrow{\mathbf{G}_{1n}(a_1^{(n-2)}, a_n)} \begin{bmatrix} a_1^{(n-1)} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (3.3.3.18)$$

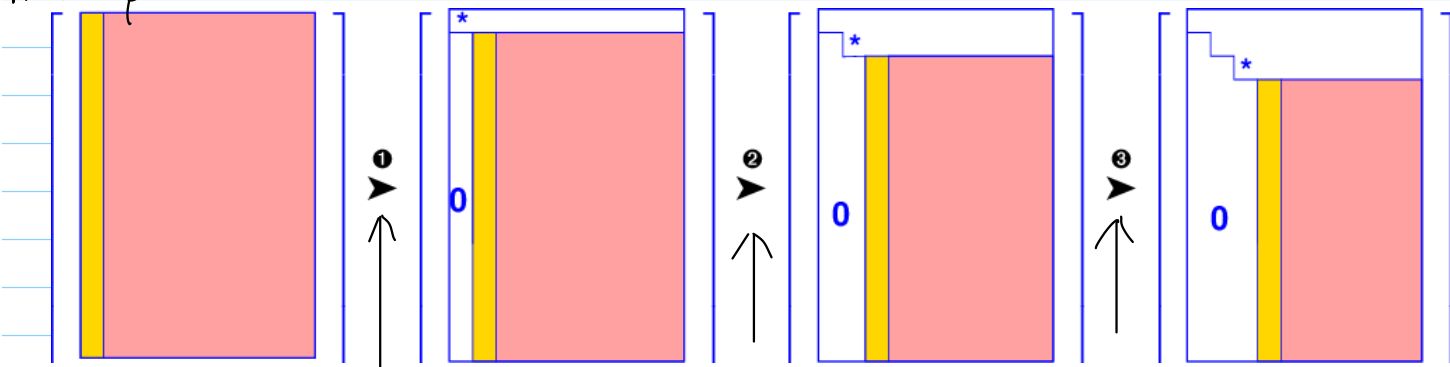
④ Rem 3.3.3.20 : Storing Givens trfs.
 $\hookrightarrow$ storing two indices + angle

for $G = \begin{bmatrix} \gamma & \sigma \\ -\sigma & \gamma \end{bmatrix} \Rightarrow$ store $\rho := \begin{cases} 1 & , \text{ if } \gamma = 0, \\ \frac{1}{2} \text{sign}(\gamma)\sigma & , \text{ if } |\sigma| < |\gamma|, \\ 2\text{sign}(\sigma)/\gamma & , \text{ if } |\sigma| \geq |\gamma|, \end{cases}$ (3.3.3.22)

which means $\begin{cases} \rho = 1 & \Rightarrow \gamma = 0, \sigma = 1, \\ |\rho| < 1 & \Rightarrow \sigma = 2\rho, \gamma = \sqrt{1 - \sigma^2}, \\ |\rho| > 1 & \Rightarrow \gamma = 2/\rho, \sigma = \sqrt{1 - \gamma^2}. \end{cases}$ (3.3.3.23)
 $\uparrow$
 no cancellation

§ 3.3.3.22 : Computational Cost

Householder-based QR-dec. of $A \in \mathbb{R}^{m,n}, m \geq n$
 $n-1$ steps:

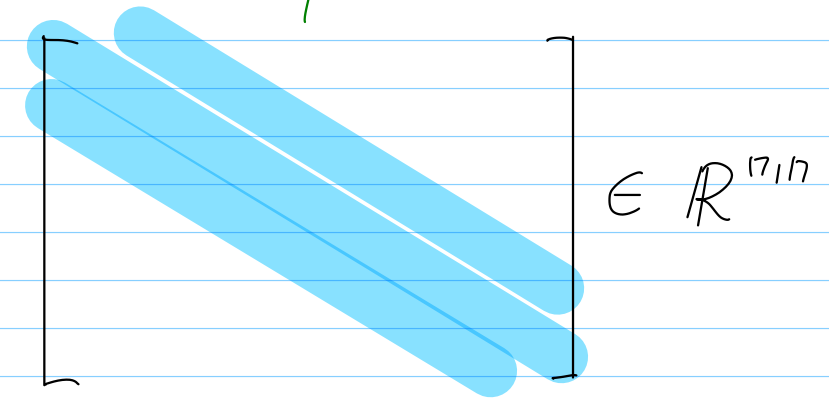


#ops: $\sim m(n-1)$ $(m-1)(n-2)$ $(m-2)(n-3)$
 Note: $H(v)X = X - \underbrace{v(v^T X)}_{\in \mathbb{R}^k}$: cost = $O(mk)$
 $X \in \mathbb{R}^{m \times k}$

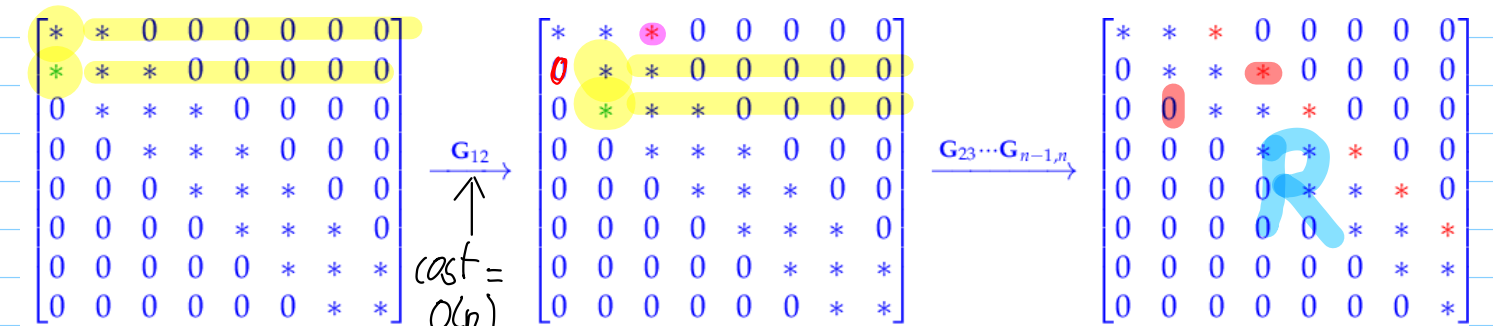
cost(R-factor of A by Householder trf.) $= \sum_{k=1}^{n-1} 2(m-k+1)(n-k)$ (3.3.3.25)
 $= O(mn^2)$ for $m, n \rightarrow \infty$.

Rem 3.3.3.26 : QR-dec. of banded matrices

Ex. : bidiagonal matrices



Givens - QR for bidiagonal matrix $\in \mathbb{R}^{n,n}$



* $\hat{=}$ entry set to zero by Givens rotation, * $\hat{=}$ new non-zero entry

In fact : cost $O(1)$ for single rotation, because ≤ 3 non-zero entries in every row of the matrices $\uparrow$ still bidiagonal

In general : $\text{bw}(A) = \text{bw}(R)$

3.3.3.4. QR-Decomposition in Eigen

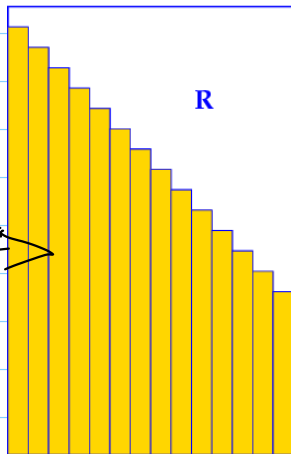
C++-code 3.3.3.34: QR-decompositions in EIGEN

```

2 # include <Eigen/QR>
3
4 // Computation of full QR-decomposition (3.3.3.1),
5 // dense matrices built for both QR-factors (expensive!)
6 std::pair<MatrixXd, MatrixXd> qr_decomp_full(const MatrixXd& A) {
7     Eigen::HouseholderQR<MatrixXd> qr(A);
8     MatrixXd Q = qr.householderQ(); // ← multiply HMMs
9     MatrixXd R = qr.matrixQR().template triangularView<Eigen::Upper>();
10    return std::pair<MatrixXd, MatrixXd>(Q,R);
11 }
12
13 // Computation of economical QR-decomposition (3.3.3.1),
14 // dense matrix built for Q-factor (possibly expensive!)
15 std::pair<MatrixXd, MatrixXd> qr_decomp_eco(const MatrixXd& A) {
16     using index_t = MatrixXd::Index;
17     const index_t m = A.rows(), n = A.cols();
18     Eigen::HouseholderQR<MatrixXd> qr(A);
19     MatrixXd Q = (qr.householderQ() * MatrixXd::Identity(m,n)); //
20     MatrixXd R = qr.matrixQR().block(0,0,n,n).template triangularView<Eigen::Upper>();
21     //
22     return std::pair<MatrixXd, MatrixXd>(Q,R);
23 }

```

compressed QR-dec.: HMMs stored in the lower triangular part



contains 2-vectors
for HMMs $H(2)$

$$\text{Cost} = O(mn^2)$$

for $m, n \rightarrow \infty$

Review Questions

Please tackle Review Questions 3.3.3.28

7