SOLUTION of (3-8.b):

The following code uses the economical SVD of $\mathbf{X} \in \mathbb{R}^{m,n}$ [Lecture → § 3.4.1.4]:

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top , \quad \mathbf{U} \in \mathbb{R}^{m,\min\{m,n\}} , \quad \mathbf{V} \in \mathbb{R}^{n,\min\{m,n\}} , \quad \mathbf{\Sigma} \in \mathbb{R}^{\min\{m,n\},\min\{m,n\}} .$$

If $\operatorname{rank}(X) = k \leq \min\{m,n\}$, then only the upper left $k \times k$-block of $\mathbf{\Sigma}$ is different from zero. Therefore, in this case

$$\mathbf{X} = \mathbf{A}\mathbf{B}^\top \quad \text{with} \quad \mathbf{A} := (\mathbf{U})_{:,1:k}(\mathbf{\Sigma})_{1:k,1:k} \in \mathbb{R}^{m,k} , \quad \mathbf{B} := (\mathbf{V})_{:,1:k} \in \mathbb{R}^{n,k} . \tag{3.8.1}$$

Note that testing from $\operatorname{rank}(\mathbf{X}) = k$ has to rely on the relative size of its singular values, compare [Lecture → Code 3.4.2.5].

The following code implements (3.8.1). Also consult [Lecture → Code 3.4.2.1] as regards the use of EIGEN's built-in functions.

C++11-code 3.8.2: Solution of (3-8.b)

```
2  std::pair<Eigen::MatrixXd, Eigen::MatrixXd> factorize_X_AB(
3      const Eigen::MatrixXd& X, const unsigned int k) {
4    std::size_t m = X.rows(), n = X.cols();
5    Eigen::MatrixXd A, B;
6    constexpr double tol =
7        1e-6;  // Tolerance for numerical rank: \lref{ex:svdrank}
8    // TODO: (3-8.b) Define A,B with k columns such that X = A*B^T
9    // Hint: warnings can be displayed with std::cerr, error messages with assert
10   // START
11   assert(k <= std::min(m, n) && "Rank k cannot be larger than dimensions of X");
12
13   Eigen::JacobiSVD<Eigen::MatrixXd> svd(
14       X, Eigen::ComputeThinU | Eigen::ComputeThinV);
15   // With Eigen::svd you can ask for thin U or V to be computed.
16   // In case of a rectangular m x n matrix,
17   // with j the smaller value among m and n,
18   // there can only be at most j singular values.
19   // The remaining columns of U and V do not correspond
20   // to actual singular vectors and are not computed in thin format.
21
22   Eigen::VectorXd s = svd.singularValues();
23
24   if (k + 1 <= std::min(m, n) && s(k) > tol * s(0)) {
25     // 1.\ condition checks if there is a singular value after k.
26     // (S cannot be larger than the dimensions of X.)
27     // 2.\ condition checks if such singular value
28     // (by definition non-negative) is "numerically" different from 0.
29     std::cerr << "Rank of matrix X is greater than required rank k"
30               << std::endl;
31   }
32   // We know that k <= std::min(m,n) by the assert
33   A = svd.matrixU().leftCols(k) * s.asDiagonal();
34   B = svd.matrixV().leftCols(k);
35   // END
36   return std::make_pair(A, B);
37 }
```

Get it on GitLab (lowrankrep.hpp).

To understand the test for $\mathrm{rank}(\mathbf{X}) > k$ please revisit [Lecture $\rightarrow$ Ex. 3.4.2.3], which explains how to check, if a singular value is “relatively close to zero”.