

C++ code 3.12.7: Constructor of **PolarDecomposition**: Polar decomposition of a low-rank matrix.

```
2 PolarDecomposition::PolarDecomposition(const Eigen::MatrixXd &A,
3                                         const Eigen::MatrixXd &B) {
4     const long m = A.rows(); // No. of rows of X
5     const long n = B.rows(); // No. of columns of X
6     const long k = A.cols(); // Maximal rank of X
7     // We assume  $k \leq n \leq m$ 
8     assert(m >= n);
9     assert(k < n);
10    assert(B.cols() == k);
11    // TODO: (3-12.d) Implement a method to initialize the data members  $Q$  and  $M$ 
12    // for  $X := AB^T = QM$ , with optimal complexity
13    // START
14    // Compute QR-decompositions in an encoded way, see \lref{par:ecovsfull}
15    Eigen::HouseholderQR<Eigen::MatrixXd> QRA(A); // cost =  $O(mk^2)$ 
16    Eigen::HouseholderQR<Eigen::MatrixXd> QRB(B); // cost =  $O(nk^2)$ 
17    const Eigen::MatrixXd RA{
18        QRA.matrixQR().block(0, 0, k, k).template triangularView<Eigen::Upper>()};
19    const Eigen::MatrixXd RB{
20        QRB.matrixQR().block(0, 0, k, k).template triangularView<Eigen::Upper>()};
21    // SVD of small  $k \times k$ -matrix  $R_A R_B^T$ 
22    // cost =  $O(k^3)$ 
23    Eigen::JacobiSVD<Eigen::MatrixXd> svdh(
24        RA * RB.transpose(), Eigen::ComputeFullU | Eigen::ComputeFullV);
25    // Extract  $k \times k$  orthogonal factor matrices
26    const Eigen::MatrixXd &Vh{svdh.matrixV()};
27    const Eigen::MatrixXd &Uh{svdh.matrixU()};
28    // Build auxiliary  $m \times n$  block matrix
29    Eigen::MatrixXd W{Eigen::MatrixXd::Zero(m, n)}; // cost =  $O(mn)$ 
30    W.block(0, 0, k, k) = Uh * Vh.transpose(); // cost =  $O(k^3)$ 
31    W.block(k, k, n - k, n - k) = Eigen::MatrixXd::Identity(n - k, n - k);
32    // Compute the  $Q$ -factor, cost =  $O(kmn + kn^2)$ 
33    Q_ =
34        (QRB.householderQ() * ((QRA.householderQ() * W).transpose())).transpose();
35    // Small  $k \times k$  matrix containing singular values on diagonal
36    const auto Sigma{svdh.singularValues().asDiagonal()};
37    // Form  $M$ -factor of polar decomposition
38    // Auxiliary matrices
39    Eigen::MatrixXd S{Eigen::MatrixXd::Zero(n, k)};
40    S.block(0, 0, k, k) = Vh * Sigma * Vh.transpose(); // cost =  $O(k^3)$ 
41    Eigen::MatrixXd T{Eigen::MatrixXd::Zero(n, n)}; // cost =  $O(n^2)$ 
42    T.block(0, 0, k, n) =
43        (QRB.householderQ() * S).transpose(); // cost =  $O(kn^2)$ 
44    M_ = (QRB.householderQ() * T).transpose(); // cost =  $O(kn^2)$ 
45    // END
46 }
```

Get it on  GitLab (polardecomposition.hpp).