

C++ code 3.15.10: EIGEN-based implementation of `test_eco_svd_concat()`

```
2 bool test_eco_svd_concat(const Eigen::MatrixXd &A1, const Eigen::MatrixXd &B1,
3                          const Eigen::MatrixXd &A2, const Eigen::MatrixXd &B2) {
4     auto [U, s, V] = eco_svd_concat(A1, B1, A2, B2);
5     // Obtain matrix dimensions and perform consistency checks
6     const unsigned int m = A1.rows();
7     const unsigned int n = B1.rows();
8     const unsigned int k = A1.cols();
9     bool ok = ((U.rows() == m) && (U.cols() == 2 * k) && (V.rows() == 2 * n) &&
10               (V.cols() == 2 * k) && (s.size() == 2 * k));
11     // Define a suitable tolerance
12     const double tau = 100 * m * n * std::numeric_limits<double>::epsilon();
13     // Ensure that the singular values are non-negative and ordered
14     ok = ok && (s[0] >= 0.0);
15     for (unsigned int i = 1; i < 2 * k; ++i) {
16         ok = ok && (s[i - 1] >= s[i]) && (s[i] >= 0.0);
17     }
18
19     // TODO: (3-15.c) Check orthonormality and correctness.
20     // START
21     // Check whether the columns of U are orthonormal
22     ok = ok &&
23         ((U.transpose() * U - Eigen::MatrixXd::Identity(2 * k, 2 * k)).norm() <=
24          tau);
25     // Check whether the columns of V are orthonormal
26     ok = ok &&
27         ((V.transpose() * V - Eigen::MatrixXd::Identity(2 * k, 2 * k)).norm() <=
28          tau);
29     // Check correctness of matrix factorization
30     const Eigen::MatrixXd P{U * s.asDiagonal() * V.transpose()};
31     const double ref = tau * s.norm();
32     ok = ok && ((A1 * B1.transpose() - P.block(0, 0, m, n)).norm() <= ref);
33     ok = ok && ((A2 * B2.transpose() - P.block(0, n, m, n)).norm() <= ref);
34     // END
35
36     return ok;
37 }
```

Get it on  GitLab (concatmatlrc.hpp).