source/linear_least_squares.hpp

```cpp
#pragma once

#include "qr_factorization.hpp"

// Backwards gaussian elimination. O(N^2) complexity.
//
// Assumes 'R' to be upper-triangular matrix.
//
inline Vector backwards_gaussian_elimination(const Matrix& R, Vector rhs) {
    for (Idx i = R.rows() - 1; i >= 0; --i) {
        for (Idx j = i + 1; j < R.cols(); ++j) rhs(i) -= R(i, j) * rhs(j);
        rhs(i) /= R(i, i);
    }

    return rhs;
}

// Linear Least Squares problem. O(N^3) complexity.
//
// LLS has a following solusion:
//    x = A^+ b
//    where A^+ = R^-1 * Q^T
//
// We can rewrite it as a SLAE:
//    R x = Q^t b
//
// since 'R' is upper-triangular, we only need to do the backwards gaussian elimination, which is O(N^2).
//
Vector linear_least_squares(const Matrix& A, const Matrix& b) {
    // Computing QR the usual way
    // const auto [Q, R] = qr_factorize(A);
    // const auto x      = backwards_gaussian_elimination(R, Q.transpose() * b);

    // Computing QR with (Q^T * b) directly
    const auto [QTb, R] = qr_factorize_lls(A, b);
    const auto x        = backwards_gaussian_elimination(R, QTb);

    return x;
}
```