

C++ code 8.15.9: Main driver function for source estimation

```
2 template <typename RECORDER = std::function<
3     void(const Eigen::Vector4d &, const Eigen::Vector4d &, double)>>
4 std::pair<Eigen::Vector3d, double> source_estimation(
5     const Eigen::Matrix<double, 3, Eigen::Dynamic> &Q,
6     const Eigen::VectorXd &ta,
7     RECORDER &&rec = [] (const Eigen::Vector4d &, const Eigen::Vector4d &,
8         double) -> void {} ) {
9     const unsigned int n = ta.size();
10    assert(Q.cols() == n);
11    // return variables
12    Eigen::Vector3d p = Eigen::VectorXd::Zero(3);
13    double t = 0.;
14    // TODO: (8-15.c) Compute the source locations and emission times
15    // using
16    // Gauss-Newton.
17    // START
18
19    // Lambda function realizing the function F from (8.15.2)
20    auto F = [&Q, &ta](const Eigen::Vector4d &x) -> Eigen::VectorXd {
21        const unsigned int n = ta.size();
22        Eigen::VectorXd result(n);
23        for (unsigned int j = 0; j < n; ++j) {
24            result[j] = (Q.col(j) - x.head(3)).norm() - ta[j] + x[3];
25        }
26        return result;
27    };
28    // Lambda function for Jacobian DF according to (8.15.6)
29    auto J = [&Q, &ta](const Eigen::Vector4d &x) -> Eigen::MatrixXd {
30        const unsigned int n = ta.size();
31        Eigen::MatrixXd Jac(n, 4);
32        for (unsigned int j = 0; j < n; ++j) {
33            const Eigen::Vector3d qp{Q.col(j) - x.head(3)};
34            Jac.block(j, 0, 1, 3) = -qp.transpose() / qp.norm();
35            Jac(j, 3) = 1.0;
36        }
37        return Jac;
38    };
39    // Compute initial guess for source position
40    // as center of gravity of receiver locations
41    Eigen::Vector3d cnt = Q.rowwise().sum() / n;
42    // Initial guess t_0 for starting time of source signal:
43    Eigen::VectorXd::Index minpos;
44    const double min_ta = ta.minCoeff(&minpos);
45    const double t_0 = min_ta - (Q.col(minpos) - cnt).norm();
46
47    // Build vector for initial guess
48    Eigen::Vector4d init = (Eigen::Vector4d() << cnt, t_0).finished();
49
50    // Gauss-Newton method with correction-based termination
51    constexpr double rtol = 1.0E-6;
52    constexpr double atol = 1.0E-8;
53    Eigen::Vector4d xs = gn(init, F, J, rtol, atol, rec);
54    p = xs.head(3);
55    t = xs[3];
56    // END
57    return {p, t};
```