

C++ code 5.10.7: Sub-problem (5-10.c): Adaptive Hermite interpolation of closed curve

```
2 template <typename CurveFunction>
3 std::pair<std::vector<Eigen::Vector2d>, std::vector<Eigen::Vector2d>>
   adaptedHermiteInterpolant(
4     CurveFunction &&c, unsigned int nmin,
5     const Eigen::VectorXd &x, double tol = 1.0e-3) {
6     // clang-format on
7     // Note: we assume that x is sorted
8     assert(x.minCoeff() >= 0 && x.maxCoeff() <= 1 && "x out of range");
9     // will contain the result
10    std::vector<Eigen::Vector2d> eval;
11    std::vector<Eigen::Vector2d> Sigma;
12
13    Eigen::VectorXd t = Eigen::VectorXd::LinSpaced(nmin + 1, 0, 1);
14    // TODO: (5-10.c)
15    // START
16    // vector of deviations
17    Eigen::VectorXd dev;
18    do {
19        // set current number of points
20        unsigned int n = t.size() - 1;
21        dev.resize(n);
22        Sigma.resize(n + 1);
23        // evaluate Sigma for arc-length points t_i
24        for (unsigned int i = 0; i <= n; ++i) {
25            Sigma[i] = c(t(i));
26        }
27        // define variables delta_1,...,delta_n and lambda_0,...,lambda_n
28        Eigen::VectorXd lambda(n + 1);
29
30        Eigen::VectorXd midpt(n);
31        lambda[0] = 0.0;
32        for (unsigned int i = 0; i < n; ++i) {
33            lambda(i + 1) = lambda(i) + (Sigma[i + 1] - Sigma[i]).norm();
34            midpt(i) = (lambda(i + 1) + lambda(i)) * 0.5;
35        }
36        // compute linear and cubic interpolant
37        std::vector<Eigen::Vector2d> v =
38            closedPolygonalInterpolant(Sigma, midpt / lambda[n]);
39        std::vector<Eigen::Vector2d> w =
40            closedHermiteInterpolant(Sigma, midpt / lambda[n]);
41        // evaluate deviations
42        for (unsigned int i = 0; i < n; ++i) {
43            dev(i) = (v[i] - w[i]).norm();
44        }
45
46        double alpha = dev.mean();
47        // append new arc-length values
48        std::vector<double> t_temp;
49        int num = 0;
50        for (unsigned int j = 0; j < n; ++j) {
51            if (dev(j) > 0.9 * alpha) {
52                t_temp.push_back((t(j) + t(j + 1)) * 0.5);
53                num++;
54            }
55        }
56        t.conservativeResize(n + 1 + t_temp.size());
```