

C++ code 5.10.4: Sub-problem (5-10.a): Computation of polygonal interpolant

```
2  std::vector<Eigen::Vector2d> closedPolygonalInterpolant(  
3      std::vector<Eigen::Vector2d> &Sigma, const Eigen::VectorXd &x) {  
4      // Note: we assume that x is sorted  
5      assert(x.minCoeff() >= 0 && x.maxCoeff() <= 1 && "x out of range");  
6  
7      std::vector<Eigen::Vector2d> res(x.size());  
8      unsigned int n = Sigma.size();  
9      // extend Sigma with periodic condition  
10     Sigma.push_back(Sigma[0]);  
11  
12     // define variables delta_1, ..., delta_n and lambda_0, ..., lambda_n  
13     std::vector<Eigen::Vector2d> d(n);  
14     std::vector<double> delta(n);  
15     std::vector<double> lambda(n + 1);  
16  
17     lambda[0] = 0.0;  
18     for (unsigned int i = 0; i < n; ++i) {  
19         d[i] = Sigma[i + 1] - Sigma[i];  
20         delta[i] = (d[i]).norm();  
21         lambda[i + 1] = lambda[i] + delta[i];  
22     }  
23  
24     // resume original Sigma value  
25     Sigma.pop_back();  
26  
27     // TODO: (5-10.a)  
28     // START  
29     unsigned int j = 0;  
30     // loop over sides of polygon  
31     for (unsigned int i = 1; i <= n; ++i) {  
32         // loop over evaluation points x  
33         while (j < x.size() && x(j) * lambda[n] <= lambda[i]) {  
34             // linear interpolant formula  
35             double ratio = (x(j) * lambda[n] - lambda[i - 1]) / delta[i - 1];  
36             Eigen::Vector2d coord = (1 - ratio) * Sigma[i - 1] + ratio * Sigma[i];  
37  
38             res[j] = coord;  
39             ++j;  
40         }  
41         if (j == x.size()) break;  
42     }  
43     // END  
44     return res;  
45 }
```

Get it on 🐱 GitLab ([adaptiveintp.hpp](#)).