

C++ code 5.15.27: C++ code for function local_curvatures()

```
2 Eigen::VectorXd ClosedQuadraticSplineCurve::local_curvatures(  
3     const Eigen::VectorXd &v) const {  
4     unsigned int N = v.size();  
5     assert(N > 0);  
6     // Vector for storing the computed curvatures  
7     Eigen::VectorXd c(N);  
8     // Lengths of knot intervals  
9     const Eigen::VectorXd h = t_.tail(n_) - t_.head(n_);  
10    // TODO: (5-15.e) Compute the local curvature at the parameter values  
11    // START  
12  
13    // Same as in curve_points(): Run through all the  
14    // provided parameter values  
15    unsigned int knot_idx = 0;  
16    for (unsigned int k = 0; k < N; ++k) {  
17        assert(((v[k] >= 0.0) && (v[k] < 1.0)) && "Out of range!");  
18        if (k > 0) {  
19            assert(v[k] >= v[k - 1] && "Parameter values not sorted!");  
20        }  
21        // Find right knot interval: knot_idx stores index of knot to the  
22        // right of  
23        // current parameter value  
24        while ((v[k] >= t_[knot_idx]) && (knot_idx < n_)) {  
25            knot_idx++;  
26        }  
27        assert(knot_idx > 0);  
28        const double tau = (v[k] - t_[knot_idx - 1]) / h[knot_idx - 1];  
29        // Derivative vector  
30        const Eigen::Vector2d ds =  
31            (p_.col(knot_idx) - p_.col(knot_idx - 1)) / h[knot_idx - 1] +  
32            (1.0 - 2 * tau) * x_.col(knot_idx - 1);  
33        // Second derivative vector  
34        const Eigen::Vector2d dds = -2.0 * x_.col(knot_idx - 1);  
35        // Evaluate formula for signed curvature  
36        const double determinant = ds[0] * dds[1] - ds[1] * dds[0];  
37        c[k] = determinant / ds.norm();  
38    }  
39    // END  
40    return c;  
}
```

Get it on 🐘 [GitLab](#) ([periodicquadraticsplines.hpp](#)).