

C++ code 7.16.4: Complete implementation of `checkQuadOrder()` [Get it on !\[\]\(221a48463f9a8d02fe9080ec754774b0_img.jpg\) GitLab.](#)

```
2 unsigned int checkQuadOrder(const QuadFormulaRef& qf,
3                             const double tol = 1.0E-10) {
4     const std::vector<double>& c = qf.nodes();
5     const std::vector<double>& w = qf.weights();
6     const unsigned int n_pts = c.size();
7     assert((w.size() == n_pts) && "Size mismatch");
8
9     // Values of monomials at quadrature points
10    std::vector<double> monom_vals(n_pts, 1.0);
11    // Check for exactness for polynomials of degree d
12    // Note that the highest possible order is 2N!
13    for (unsigned int d = 0; d < n_pts; ++d) {
14        // j-loop performs the evaluation of the quadrature formula
15        // for the monomial  $t \mapsto t^{2d}$  (even polynomial)
16        double s = 0.0;
17        for (unsigned int j = 0; j < n_pts; ++j) {
18            s += w[j] * monom_vals[j];
19        }
20        // Exact value:  $2/(d+1)$  for even degree, zero else.
21        double val = 2.0 / (2 * d + 1);
22        // Safe test for equality of computed floating point numbers
23        if (std::abs(s - val) > tol) {
24            return 2 * d;
25        }
26        // Now test for odd-degree monomials, for which the integral has to
27        // evaluate to zero
28        for (unsigned int j = 0; j < n_pts; ++j) {
29            monom_vals[j] *= c[j];
30        }
31        s = 0.0;
32        for (unsigned int j = 0; j < n_pts; ++j) {
33            s += w[j] * monom_vals[j];
34        }
35        // Safe test for equality with zero
36        if (std::abs(s) > tol) {
37            return 2 * d + 1;
38        }
39        for (unsigned int j = 0; j < n_pts; ++j) {
40            monom_vals[j] *= c[j];
41        }
42    }
43    // The quadrature rule has the highest possible order
44    return 2 * n_pts;
45 }
```