

C++ code 7.13.9: Code for `testConvTrfGaussQuad()`

```
2 void testConvTrfGaussQuad() {
3     // Table header
4     std::cout << std::setw(5) << "n" << std::setw(20) << " l_n " << std::setw(30)
5         << " Error " << std::setw(10)
6         << (" decay ratio\n " + std::string(65, '-')) << std::endl;
7
8     // Value obtained by overkill quadrature as reference value
9     constexpr double ref_val = 1.7576137811123187;
10
11     // TODO: (7-13.d) Print a table that allows you to predict the
12     // asymptotic
13     // behaviour of arccosWeightedQuad().
14     // START
15     // The integrand (/arccos(t)) as lambda funtion
16     auto f = [](double t) { return 1.0 / (1.0 + std::exp(t)); };
17
18     // For recording errors etc.
19     std::vector<cvpdata> data;
20     // We expect exponential convergence. Increment number of quadrature
21     // nodes by
22     // one in each step
23     constexpr unsigned int L = 12; // Use a most L quadrature nodes
24     for (unsigned int n = 2; n <= L; ++n) {
25         data.push_back({n, arccosWeightedQuad(f, n), 0.0, 0.0});
26     }
27     for (unsigned int l = 0; l < data.size(); ++l) {
28         data[l].err_ = std::abs(data[l].val_ - ref_val);
29         if (l > 0) {
30             // Quotient of successive error values
31             data[l].rate_ = data[l].err_ / data[l - 1].err_;
32         }
33     }
34
35     // Output table
36     for (unsigned int l = 0; l < data.size(); ++l) {
37         const cvpdata& item{data[l]};
38         std::cout << std::setw(5) << item.n_ << std::setw(20) << std::fixed
39             << std::setprecision(16) << item.val_ << std::setw(30)
40             << std::scientific << std::setprecision(16) << item.err_
41             << std::setw(10) << std::fixed << std::setprecision(2)
42             << item.rate_ << std::endl;
43     }
44     // END
45 }
```

Get it on  GitLab ([arccosquad.hpp](#)).