

C++ code 7.15.9: Computation of convolution quadrature weights

```
2  template <typename FFUNCTOR>
3  Eigen::VectorXcd compute_cq_weights(FFUNCTOR &&F, unsigned int N, double tau,
4                                     double r = 0.0) {
5      using Comp = std::complex<double>;
6      assert(N >= 1 && "At least two quadrature points have to be used!");
7      // Default choice for circle radius for integration contour
8      if (r == 0.0) {
9          r = std::pow(std::numeric_limits<double>::epsilon(), 0.5 / (N + 1));
10     }
11     Eigen::VectorXcd w(N + 1);
12     const Comp iu = 2.0 * Comp{0.0, 1.0} * M_PI; // 2*pi*i
13     // TODO: (7-15.c) Compute the approximate convolution quadrature
14     // weights by
15     // means of the equidistant trapezoidal rule.
16     // START
17     Eigen::VectorXcd v(N + 1);
18     // Sampling of F on a circle in the complex plane, see (7.15.7)
19     const Comp expfac = iu / (static_cast<double>(N + 1));
20     for (unsigned int k = 0; k <= N; ++k) {
21         const Comp z =
22             (Comp{1.0, 0.0} - r * std::exp(expfac * static_cast<double>(k))) / tau;
23         v[k] = F(z);
24     }
25     // Compute DFT of the vector v
26     Eigen::FFT fft;
27     w = fft.fwd(v); // DFT in EIGEN
28     // Finally perform the scaling of the components
29     double fac = N + 1;
30     for (unsigned int k = 0; k <= N; ++k) {
31         w(k) /= fac;
32         fac *= r;
33     }
34     // END
35     return w;
36 }
```

Get it on  GitLab (convolutionquadrature.hpp).