C++ code 5.2.49: Implementation of `compute_rhs()` ⇝ GitLab

```
Eigen::VectorXd compute_rhs(
    const Eigen::Matrix<double, 2, Eigen::Dynamic> &knots, Eigen::Vector2d a,
    Eigen::Vector2d b) {
  Eigen::Index M = knots.cols() - 1;
  double h = 1. / static_cast<double>(M);
  // Compute the "coefficients" for all cells
  // A bit wasteful, because we only need sigma[0] and sigma[M-1]
  const Eigen::VectorXd sigma = coeff_sigma(knots);
  const Eigen::Matrix<double, Eigen::Dynamic, 2> f = sourcefn2(knots);
  // Right-hand side vector
  Eigen::VectorXd rhs(2 * (M - 1));
  // Compute head part half of rhs: φ⃗_1
  rhs.setZero();
  rhs[0] = sigma[0] * a[0] / h;
  rhs[M - 2] = sigma[M - 1] * b[0] / h;
  // Compute second half of rhs based on the 2-point Gauss rule
  const double w1 = .5 - .5 / std::sqrt(3.);
  const double w2 = .5 + .5 / std::sqrt(3.);
  for (int i = 0; i < M - 1; ++i) {
    rhs[M - 1 + i] =
        0.5 * h *
        (w1 * f(i, 0) + w2 * f(i, 1) + w2 * f(i + 1, 0) + w1 * f(i + 1, 1));
  }
  // Take into account offset function
  rhs[M - 1] += sigma[0] * a[1] / h;
  rhs[2 * M - 3] += sigma[M - 1] * b[1] / h;
  return rhs;
}
```