C++ code 3.14.14: Implementation of quasi-interpolation operator onto $\mathcal{S}_1^0(\mathcal{M})$ ↪ GitLab

```
template <typename MESHFUNCTION>
Eigen::VectorXd quasiInterpolate(
    const lf::uscalfe::FeSpaceLagrangeO1<double> &fe_space,
    MESHFUNCTION &&v_mf) {
  // Get quadrature points and weights on the reference triangle for a
  // quadrature rule that integrates quadratic polynomials exactly
  lf::quad::QuadRule quadrule =
      lf::quad::make_QuadRule(lf::base::RefEl::kTria(), 2);
  const int P = quadrule.NumPoints();
  Eigen::MatrixXd zeta_hat = quadrule.Points();
  Eigen::VectorXd w_hat = quadrule.Weights();
  // Precomputation:  Cache values of (ψ̂¹, ψ̂², ψ̂³)
  // at reference quadrature nodes.  Use the formulas (3.14.4).
  Eigen::Matrix<double, 3, 2> A;
  A << -24.0, -24.0, 24.0, 0.0, 0.0, 24.0;
  Eigen::Vector3d b{18.0, -6.0, -6.0};
  auto psi_hat = [A, b](Eigen::Vector2d x) -> Eigen::Vector3d {
    return A * x + b;
  };
  Eigen::MatrixXd psi_hat_values(3, P);
  for (int l = 0; l < P; ++l) {
    psi_hat_values.col(l) = psi_hat(zeta_hat.col(l));
  }
  // Retrieve the map p ↦ (K_p, j)
  std::shared_ptr<const lf::mesh::Mesh> mesh_p = fe_space.Mesh();
  lf::mesh::utils::CodimMeshDataSet<
      std::pair<const lf::mesh::Entity *, unsigned int>>
      Kp_mesh_data_set = findKp(mesh_p);

  const lf::assemble::DofHandler &dofh{fe_space.LocGlobMap()};
  const int N = dofh.NumDofs();
  // Vector for returning the basis expansion coefficients of the
  // quasi-interpolant
  Eigen::VectorXd coefficients(N);
  // Compute projection onto each basis function
  for (lf::assemble::gdof_idx_t i = 0; i < N; ++i) {
    // Get p, K_p and the local index j of p in K_p
    const lf::mesh::Entity &point = dofh.Entity(i);
    std::pair<const lf::mesh::Entity *, unsigned int> Kp_localindex =
        Kp_mesh_data_set(point);
    const lf::mesh::Entity *Kp = Kp_localindex.first;
    unsigned int j = Kp_localindex.second;
    // Evaluate v on quadrature points in K_p
    const std::vector<double> v_zeta = v_mf(*Kp, zeta_hat);
    // Compute integral on current triangle Kp
    double sum = 0.0;
    for (int l = 0; l < P; ++l) {
      sum += w_hat(l) * v_zeta[l] * psi_hat_values(j, l);
    }
    coefficients(i) = sum;
  }
  return coefficients;
}
```