C++ code 11.11.8: Implementation of `semiDiscreteRhs()` ↪ GitLab

```cpp
template <class CAUCHYPROBLEM>
Eigen::VectorXd semiDiscreteRhs(const Eigen::VectorXd &mu,
                                const Eigen::VectorXd &zeta,
                                CAUCHYPROBLEM prb) {
  int N = mu.size();
  Eigen::VectorXd rhs(N);

  // Define the (simplified) Rusanov 2-point numerical flux
  auto F = [&mu, &zeta, &prb](int i, int j) {
    double v = mu[i];
    double w = mu[j];
    double fv = prb.g(zeta[i]);
    double fw = prb.g(zeta[j]);
    double dfv = prb.dg(zeta[i]) / prb.drho(zeta[i]);
    double dfw = prb.dg(zeta[j]) / prb.drho(zeta[j]);
    double max_abs_df = std::max(std::abs(dfv), std::abs(dfw));
    return 0.5 * (fv + fw) - 0.5 * (w - v) * max_abs_df;
  };

  // Compute (-h) * [right-hand side]
  rhs[0] = F(0, 1) - F(0, 0);
  for (int j = 1; j < N - 1; ++j) {
    rhs[j] = F(j, j + 1) - F(j - 1, j);
  }
  rhs[N - 1] = F(N - 1, N - 1) - F(N - 2, N - 1);

  // Compute cell spacing h
  std::pair<double, double> limits = prb.domain();
  double h = (limits.second - limits.first) / (N - 1);

  // Include factor -1.0 / h
  rhs = (-1.0 / h) * rhs;

  return rhs;
}
```