C++11 code 11.1.9: Sub-problem (11-1.k), function `numexpBurgersGodunov()`, ⇢ GitLab

```
Eigen::VectorXd reduce(const Eigen::VectorXd &mu, unsigned int N) {
  Eigen::VectorXd mu_sub(N + 1);
  int fraction = mu.size() / N;
  for (int j = 0; j < N + 1; ++j) {
    mu_sub(j) = mu(j * fraction);
  }
  return mu_sub;
}

Eigen::Matrix<double, 3, 4> numexpBurgersGodunov() {
  const unsigned int N_large = 3200;
  Eigen::Vector2d T{0.3, 3.0};
  Eigen::Vector4i N{5 * 10, 5 * 20, 5 * 40, 5 * 80};
  Eigen::Vector4d h;
  for (int i = 0; i < 4; ++i) h(i) = 5.0 / N(i);

  Eigen::Matrix<double, 3, 4> result;
  result.row(0) = h.transpose();

  for (int k = 0; k < 2; ++k) {
    Eigen::VectorXd mu_ref = solveBurgersGodunov(T(k), N_large);
    Eigen::Vector4d error;
    for (int i = 0; i < 4; ++i) {
      Eigen::VectorXd mu = solveBurgersGodunov(T(k), N(i));
      Eigen::VectorXd mu_ref_sub = reduce(mu_ref, N(i));
      error(i) = h(i) * (mu - mu_ref_sub).lpNorm<1>();
    }
    result.row(k + 1) = error.transpose();
  }

  return result;
}
```