C++ code 11.4.7: Sub-problem (11-4.e): function `solveLaxWendroff()` ⇝ GITHUB

```
Eigen::VectorXd LaxWendroffRhs(const Eigen::VectorXd &mu, double gamma) {
  int N = mu.size();
  Eigen::VectorXd result(N);

  auto f = [](double x) { return std::exp(x); };
  auto df2 = [](double x) { return Square(std::exp(x)); };

  // Lax-Wendroff fully discrete evolution (11.4.1) for
  // f(u) = e^u.  Store the values of f(x) and (f'(x))^2 from the
  // previous iteration:

  // We extend mu to the left by mu(0):
  double mu_left = mu(0);

  // f'(1/2(μ_j + μ_{j-1}))^2:
  double df2_old = df2(0.5 * (mu(0) + mu_left));

  // f'(1/2(μ_{j+1} + μ_j))^2:
  double df2_new = df2(0.5 * (mu(1) + mu(0)));

  double f_old = f(mu_left);  // f(μ_{j-1})
  double f_mid = f(mu(0));    // f(μ_j)
  double f_new = f(mu(1));    // f(μ_{j+1})

  result(0) = mu(0) - 0.5 * gamma * (f_new - f_old) +
              0.5 * Square(gamma) *
                  (df2_new * (mu(1) - mu(0)) - df2_old * (mu(0) - mu_left));

  for (int j = 1; j < N - 1; ++j) {
    df2_old = df2_new;
    df2_new = df2(0.5 * (mu(j + 1) + mu(j)));
    f_old = f_mid;
    f_mid = f_new;
    f_new = f(mu(j + 1));
    result(j) =
        mu(j) - 0.5 * gamma * (f_new - f_old) +
        0.5 * Square(gamma) *
            (df2_new * (mu(j + 1) - mu(j)) - df2_old * (mu(j) - mu(j - 1)));
  }

  // We extend mu to the right by mu(N-1):
  double mu_right = mu(N - 1);

  df2_old = df2_new;
  df2_new = df2(0.5 * (mu_right + mu(N - 1)));

  f_old = f_mid;
  f_new = f(mu_right);

  result(N - 1) = mu(N - 1) - 0.5 * gamma * (f_new - f_old) +
                  0.5 * Square(gamma) *
                      (df2_new * (mu_right - mu(N - 1)) -
                       df2_old * (mu(N - 1) - mu(N - 2)));

  return result;
```