C++ code 11.10.14: code for `fluxlimBurgers` ⇢ GitLab

```
template <typename FLUXLIM = std::function<double(double)>>
Eigen::VectorXd fluxlimBurgers(
    const Eigen::VectorXd &mu0, double h, double tau, unsigned int nb_timesteps,
    FLUXLIM &&phi = [](double /*theta*/) { return 1.0; }) {
  Eigen::VectorXd mu;  // return vector
  int N = mu0.size();  // Number of sptial dual cells
  double gamma = tau / h;
  // Set initial conditions
  mu = mu0;
  // Flux function for Burgers equation
  auto f = [](double u) { return 0.5 * u * u; };
  // Godunov numerical flux
  auto godnfnburgers = [f](double v, double w) -> double {
    if (v > w) {
      if (v + w > 0) {
        return f(v);
      }
      return f(w);
    }
    if (v > 0) {
      return f(v);
    }
    if (0 < w) {
      return 0.0;
    }
    return f(w);

  };

  // Constant continuation index mapping tool
  auto idx_map = [N](int idx) {
    if (idx < 0) {
      return 0;
    }
    if (idx > N - 1) {
      return N - 1;
    }
    return idx;
  };
  // Rankine-Hugoniot speed
  auto s_dot = [f](double v, double w) {
    if (v == w) {
      return 0.0;
    }
    return (f(w) - f(v)) / (w - v);
  };
  // The quantity θ from (11.10.6d)
  // Implementation avoid division by zero, see thetaquotient()
  auto theta = [s_dot](double x, double v, double w, double y) {
    if (s_dot(v, w) < 0) {
      if (y == w) {
        return 1.0e17 * (w - v);
      }
      return (w - v) / (y - w);
    }
    if (v == w) {
```