C++11 code 11.1.6: Implementation of `solveBurgersGodunov()` for Sub-problem (11-1.h)
➙ GitLab

```
constexpr double PI = 3.14159265358979323846;

double Square(double x) { return x * x; }

double f(double x) { return 2.0 / 3.0 * std::sqrt(x * x * x); }

Eigen::VectorXd solveBurgersGodunov(double T, unsigned int N) {
  double h = 5.0 / N;            // meshwidth
  double tau = h;                // timestep = meshwidth by CFL condition
  int m = std::round(T / tau);   // no. of timesteps

  // initialize vector with initial nodal values
  Eigen::VectorXd x = Eigen::VectorXd::LinSpaced(N + 1, -1.0, 4.0);
  Eigen::VectorXd mu =
      x.unaryExpr([](double x) {
         return 0.0 <= x && x <= 1.0 ? Square(std::sin(PI * x)) : 0.0;
       }).eval();

  for (int i = 0; i < m; ++i) {
    for (int j = N; 0 < j; --j) {
      // Standard fully discrete evolution based on explicit Euler timestepping
      mu(j) = mu(j) - tau / h * (f(mu(j)) - f(mu(j - 1)));
    }
    // truncation to a finite vector. Only required on one side, because all
    // information flows from left to right.
    mu(0) = 0.0;  // Value of u0 to the left of x=0
  }

  return mu;
}
```