C++11 code 11.7.13: Implementation off `slopelimfluxdiffper()` ↪ GitLab

```
// Function parameters:
// double h:  meshwidth of equidistant spatial grid
// Vector mu:  (finite) vector μ⃗ of cell averages
// Functor F: 2-point numerical flux function F = F(v,w)
// Functor slope:  slope reconstruction function
// hσ_j = slopes(μ_{j-1}, μ_j, μ_{j+1})
//
// returns a vector with differences of numerical fluxes
//
// Function that realizes −h· the right hand side operator
// L_h for the mehtod-of-lines ODE arising from conservative finite
// volume semidiscretization of the Cauchy problem for a 1D scalar
    conservation
// law [Lecture → Eq. (11.2.2.1)] in a 1-periodic setting.
template <typename FunctionF, typename FunctionSlopes>
Eigen::VectorXd slopelimfluxdiffper(const Eigen::VectorXd &mu, FunctionF &&F,
                                    FunctionSlopes &&slopes) {
  int n = mu.size();  // Number of active dual grid cells
  Eigen::VectorXd sigma = Eigen::VectorXd::Zero(n);  // Vector of slopes
  Eigen::VectorXd fd = Eigen::VectorXd::Zero(n);     // Flux differences

  // Computation of slopes σ_j, uses μ_{-1} = μ_{n-1},
  // μ_n = μ_0, which amounts to constant extension of states
  // beyond domain of influence [a,b] of non-constant intial data.  Same
  // technique has been applied in [Lecture → Code 11.3.2.9]
  sigma[0] = slopes(mu[n - 1], mu[0], mu[1]);  // @@
  for (int j = 1; j < n - 1; ++j)
    sigma[j] = slopes(mu[j - 1], mu[j], mu[j + 1]);
  sigma[n - 1] = slopes(mu[n - 2], mu[n - 1], mu[0]);  // @@

  // Compute linear reconstruction at endpoints of dual cells [Lecture →
     Eq. (11.5.1.2)]
  Eigen::VectorXd nup = mu + 0.5 * sigma;
  Eigen::VectorXd num = mu - 0.5 * sigma;

  // Rely on periodicity to compute numerical fluxes at interval ends
  fd[0] = F(nup[0], num[1]) - F(nup[n - 1], num[0]);  // @@
  for (int j = 1; j < n - 1; ++j)
    // see [Lecture → Eq. (11.3.2.8)]
    fd[j] = F(nup[j], num[j + 1]) - F(nup[j - 1], num[j]);
  fd[n - 1] = F(nup[n - 1], num[0]) -  // @@
              F(nup[n - 2], num[n - 1]);
  return fd;
}
```