C++ code 3.13.13: MeshFunction-based implementation of `stabFlux()` ⇝ GitLab

```
2  template <typename SIGMAFUNCTION, typename PSIGRAD>
3  double stabFluxMF(
4      std::shared_ptr<const lf::uscalfe::FeSpaceLagrangeO1<double>> fe_space,
5      const Eigen::VectorXd &sol_vec, SIGMAFUNCTION &&sigma, PSIGRAD &&gradpsi) {
6    std::shared_ptr<const lf::mesh::Mesh> mesh_p{fe_space->Mesh()};
7    // Coefficient function and weight function
8    const lf::mesh::utils::MeshFunctionGlobal mf_sigma(sigma);
9    const lf::mesh::utils::MeshFunctionGlobal mf_gradpsi(gradpsi);
10   // Build a MeshFunction representing the gradient of the finite element
11   // solution
12   const lf::fe::MeshFunctionGradFE mf_grad(fe_space, sol_vec);
13   // Mesh function representing the integrand
14   const auto mf_itg{lf::mesh::utils::transpose(mf_sigma * mf_grad) *
15                     mf_gradpsi};
16   const double s = lf::fe::IntegrateMeshFunction(
17       *mesh_p, mf_itg, [](const lf::mesh::Entity &e) {
18         return lf::quad::make_QuadRule(e.RefEl(), 2);
19       })(0, 0);
20   return s;
21 }  // end stabFluxMF
```