C++ code 5.5.1: MeshFunction wrapper class FunctionMFWrapper

```cpp
template <typename MESHFUNCTION, typename FUNCTION>
class FunctionMFWrapper {
 public:
  static_assert(lf::mesh::utils::MeshFunction<MESHFUNCTION>);
  using mf_vector_t = decltype(std::declval<MESHFUNCTION>()(
      std::declval<const lf::mesh::Entity>(),
      std::declval<const Eigen::MatrixXd>()));
  using mf_result_t = typename mf_vector_t::value_type;
  // Return type of the function F
  using F_result_t =
      decltype(std::declval<FUNCTION>()(std::declval<mf_result_t>()));

  explicit FunctionMFWrapper(MESHFUNCTION mf, FUNCTION F)
      : mf_(std::move(mf)), F_(std::move(F)) {}
  FunctionMFWrapper(const FunctionMFWrapper &) = default;
  FunctionMFWrapper(FunctionMFWrapper &&) noexcept = default;
  FunctionMFWrapper &operator=(const FunctionMFWrapper &) = delete;
  FunctionMFWrapper &operator=(FunctionMFWrapper &&) = delete;
  ~FunctionMFWrapper() = default;

  std::vector<F_result_t> operator()(const lf::mesh::Entity &e,
                                     const Eigen::MatrixXd &local) const {
    LF_ASSERT_MSG(e.RefEl().Dimension() == local.rows(),
                  "mismatch between entity dimension and local.rows()");
    const std::vector<mf_result_t> mf_result = mf_(e, local);
    std::vector<F_result_t> result(local.cols());
    for (long i = 0; i < local.cols(); ++i) {
      result[i] = F_(mf_result[i]);
    }
    return result;
  }

 private:
  MESHFUNCTION mf_;
  FUNCTION F_;
};
```