C++ code 9.9.37: Implementation of `modelproblem()` ⇝ GitLab

```cpp
void modelproblem() {
  std::cout << "You are running the model problem, i.e. you solve the Fisher "
               "equation on the model domain. "
            << std::endl;
  // Obtain mesh
  auto mesh_factory = std::make_unique<lf::mesh::hybrid2d::MeshFactory>(2);
  const lf::io::GmshReader reader(std::move(mesh_factory), "meshes/island.msh");
  std::shared_ptr<const lf::mesh::Mesh> mesh_p = reader.mesh();
  // Finite Element Space
  auto fe_space =
      std::make_shared<lf::uscalfe::FeSpaceLagrangeO1<double>>(mesh_p);
  // Dofhandler
  const lf::assemble::DofHandler &dofh{fe_space->LocGlobMap()};
  const lf::uscalfe::size_type N_dofs(dofh.NumDofs());
  // Initial Population density
  Eigen::VectorXd u0(N_dofs);
  u0.setZero();
  u0(321) = 0.3;
  u0(567) = 0.3;

  // Diffusion Coefficient
  auto c = [](Eigen::Vector2d x) -> double { return 1.2; };
  double lambda = 2.1;  // Growth Factor
  // Carrying capacity
  Eigen::VectorXd K{0.8 * Eigen::VectorXd::Ones(N_dofs)};
  // Time Steps
  unsigned int m = 100;
  double T = 1.;

  // Compute the solution with method of lines and Strang splitting
  StrangSplit StrangSplitter(fe_space, T, m, lambda, c);
  // Five snapshots
  std::vector<Eigen::VectorXd> sol;
  std::cout << "Computing solution after 100 timesteps..." << std::endl;
  sol.push_back(StrangSplitter.Evolution(K, u0));
  // Uncomment the following calls to StrangSplitter.Evolution in order
  // to solve for a longer evolution.
  /*std::cout << "Computing solution after 200 timesteps..." << std::endl;
  sol.push_back(StrangSplitter.Evolution(K, sol[0]));
  std::cout << "Computing solution after 300 timesteps..." << std::endl;
  sol.push_back(StrangSplitter.Evolution(K, sol[1]));
  std::cout << "Computing solution after 400 timesteps..." << std::endl;
  sol.push_back(StrangSplitter.Evolution(K, sol[2]));
  std::cout << "Computing solution after 500 timesteps..." << std::endl;
  sol.push_back(StrangSplitter.Evolution(K, sol[3]));*/

  // Use VTK-Writer for Visualization of solution.
  std::cout << "Writting solution(s) in VTK format." << std::endl;
  std::cout << std::size(sol) << std::endl;
  for (int k = 0; k < std::size(sol); k++) {
    std::stringstream filename;
    filename << "model_problem_sol" << k + 1 << ".vtk";
    lf::io::VtkWriter vtk_writer(mesh_p, filename.str());
    auto nodal_data = lf::mesh::utils::make_CodimMeshDataSet<double>(mesh_p, 2);
    for (int global_idx = 0; global_idx < N_dofs; global_idx++) {
      nodal_data->operator()(dofh.Entity(global_idx)) = sol[k][global_idx];
```