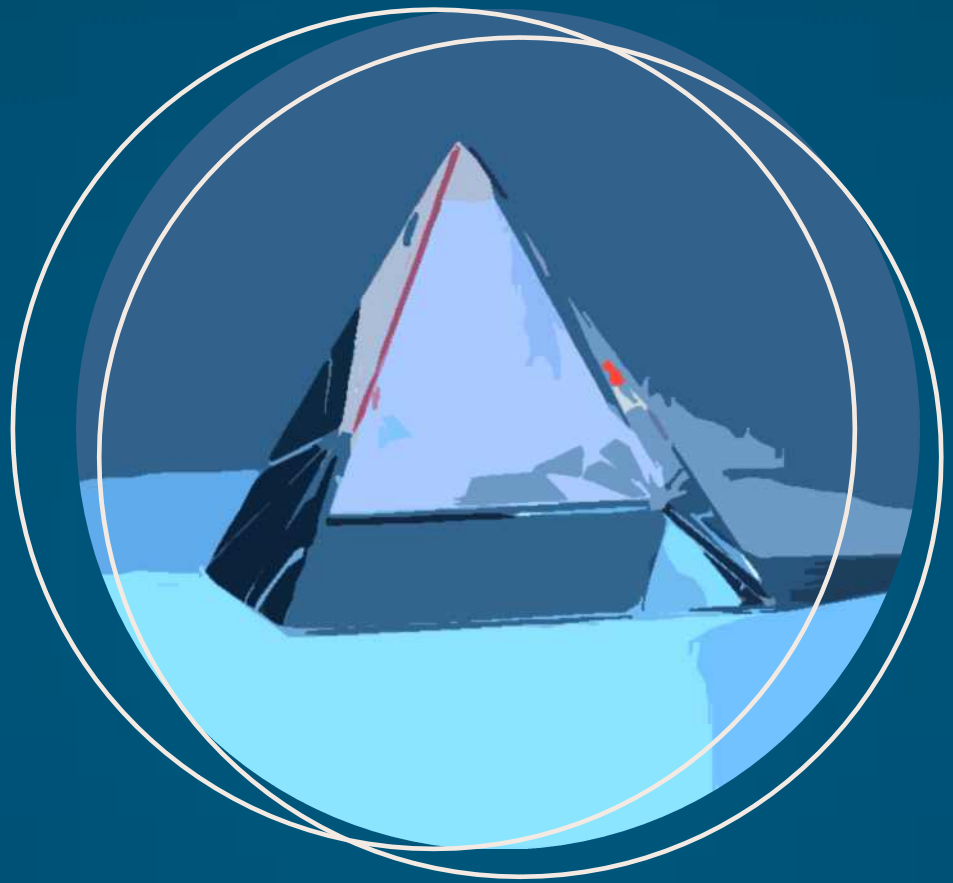


FEM on Poisson Equation

0 Basic code

```
//Basic Class
class FEMSolve
{
public:
  FEMSolve();

  void run();

private:
  void make_grid();
  void setup_system();
  void assemble_system();
  void solve();
  void output_results() const;

  Triangulation<2> triangulation;
  FE_Q<2>          fe;
  DoFHandler<2>    dof_handler;

  SparsityPattern      sparsity_pattern;
  SparseMatrix<double> system_matrix;

  Vector<double> solution;
  Vector<double> system_rhs;
};
```

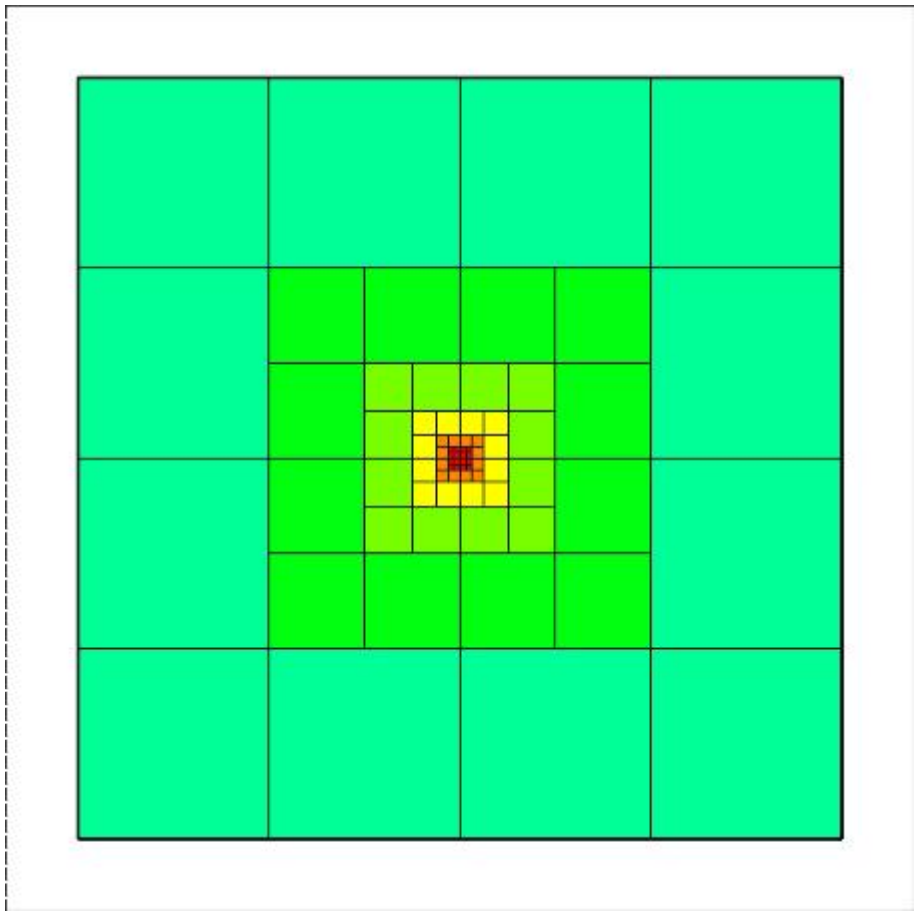
```
for (const auto &cell : dof_handler.active_cell_iterators())
  {
    fe_values.reinit(cell);

    cell_matrix = 0;
    cell_rhs    = 0;

    for (const unsigned int q_index : fe_values.quadrature_point_indices())
      {
        for (const unsigned int i : fe_values.dof_indices())
          for (const unsigned int j : fe_values.dof_indices())
            cell_matrix(i, j) +=
              (fe_values.shape_grad(i, q_index) * // grad phi_i(x_q)
               fe_values.shape_grad(j, q_index) * // grad phi_j(x_q)
               fe_values.JxW(q_index));           // dx

        for (const unsigned int i : fe_values.dof_indices())
          cell_rhs(i) += (fe_values.shape_value(i, q_index) * // phi_i(x_q)
                          1. *                                // f(x_q)
                          fe_values.JxW(q_index));            // dx
      }
    cell->get_dof_indices(local_dof_indices);

    for (const unsigned int i : fe_values.dof_indices())
      for (const unsigned int j : fe_values.dof_indices())
        system_matrix.add(local_dof_indices[i],
                          local_dof_indices[j],
                          cell_matrix(i, j));

    for (const unsigned int i : fe_values.dof_indices())
      system_rhs(local_dof_indices[i]) += cell_rhs(i);
  }

std::map<types::global_dof_index, double> boundary_values;
VectorTools::interpolate_boundary_values(dof_handler,
                                         0,
                                         Functions::ZeroFunction<2>(),
                                         boundary_values);
MatrixTools::apply_boundary_values(boundary_values,
                                   system_matrix,
                                   solution,
                                   system_rhs);
```

Part 1

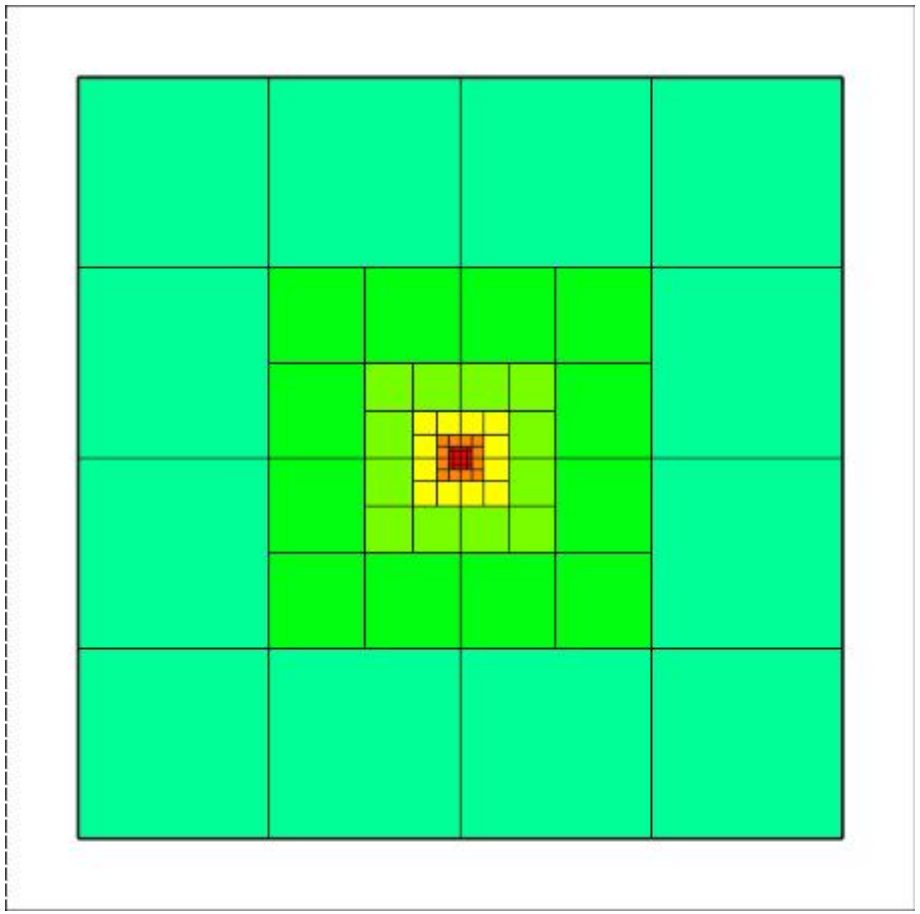
Grid generating and refining

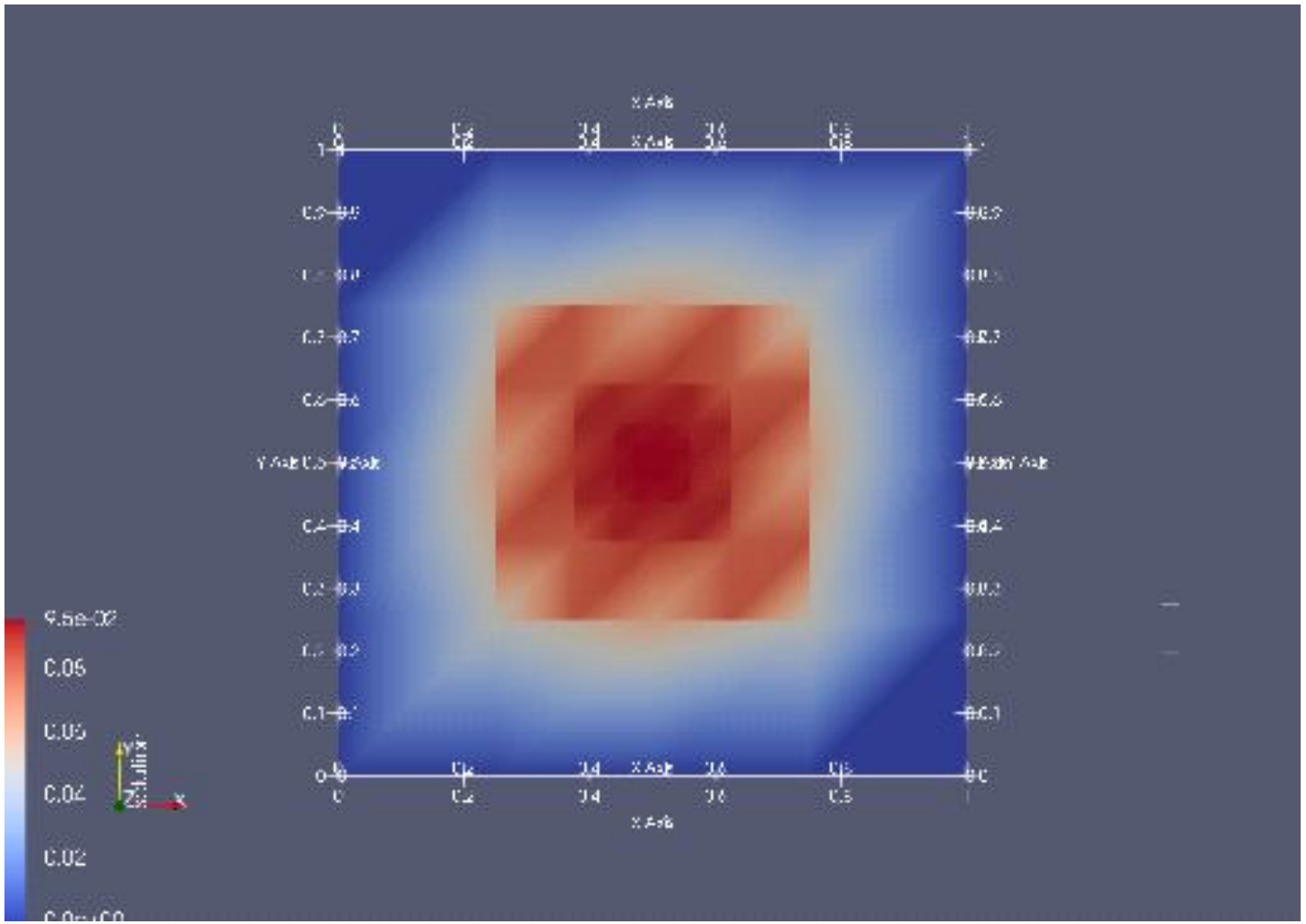
1 Square grid

```
//Initialize Grid
void FEMSolve::make_grid()
{
  const Point<2> center(0.5, 0.5);
  GridGenerator::hyper_cube(triangulation);
  //Making 4 by 4 squares
  triangulation.refine_global(2);

  //Dividing center squares
  for (unsigned int step=0; step<5; step++)
  {
     for (auto &cell : triangulation.active_cell_iterators())
     {
        for (const auto v : cell->vertex_indices())
        {
        const double distance_from_center = center.distance(cell->vertex(v));

        if (distance_from_center <= 1e-6)
            {
            cell->set_refine_flag();
            break;
            }
        }
     }
     triangulation.execute_coarsening_and_refinement();
  }
```



1 Square grid

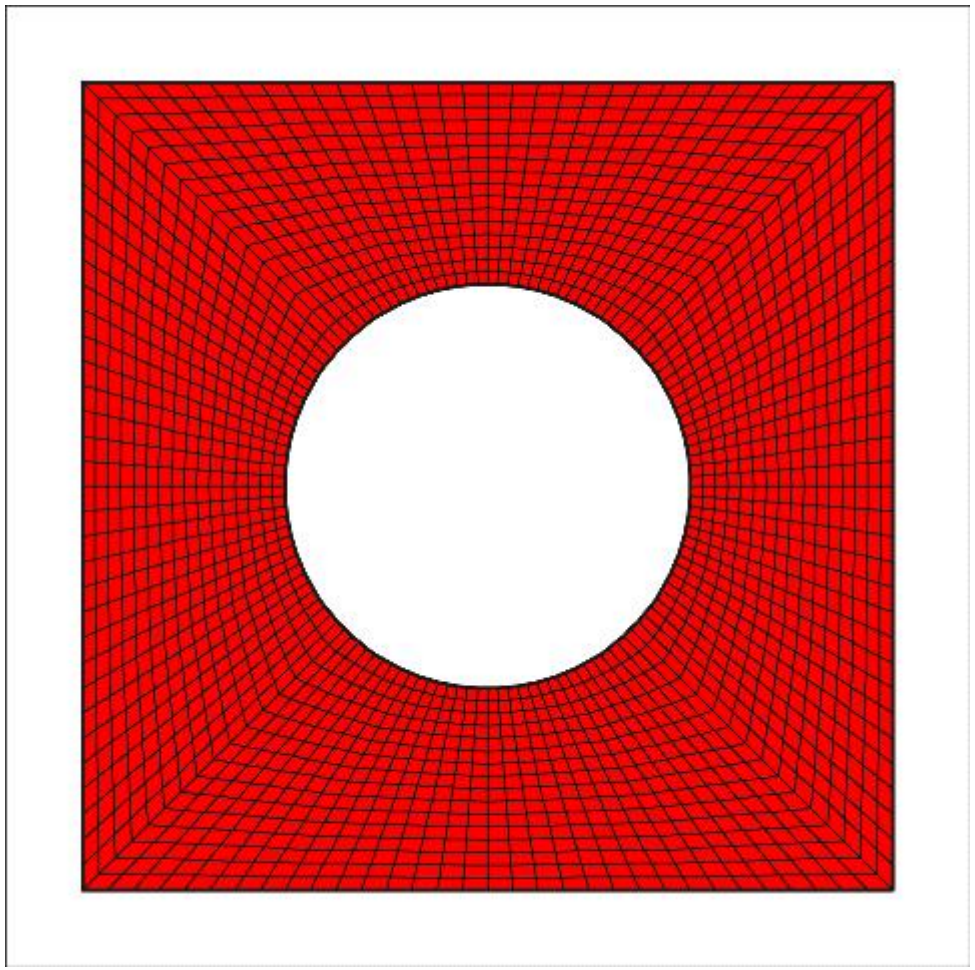




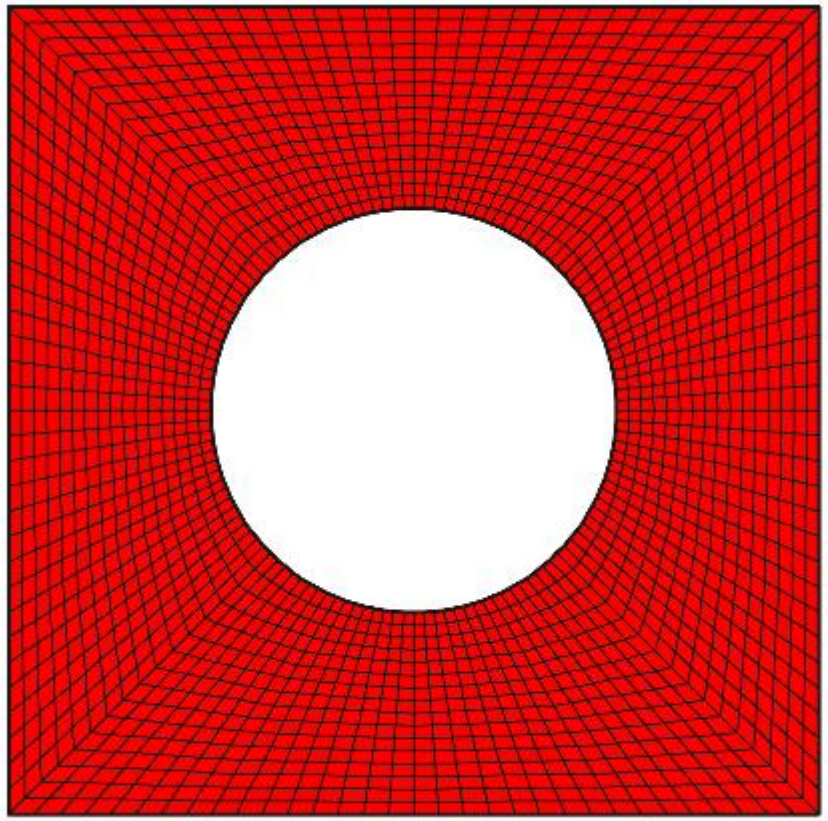
1 Square with cyliner hole

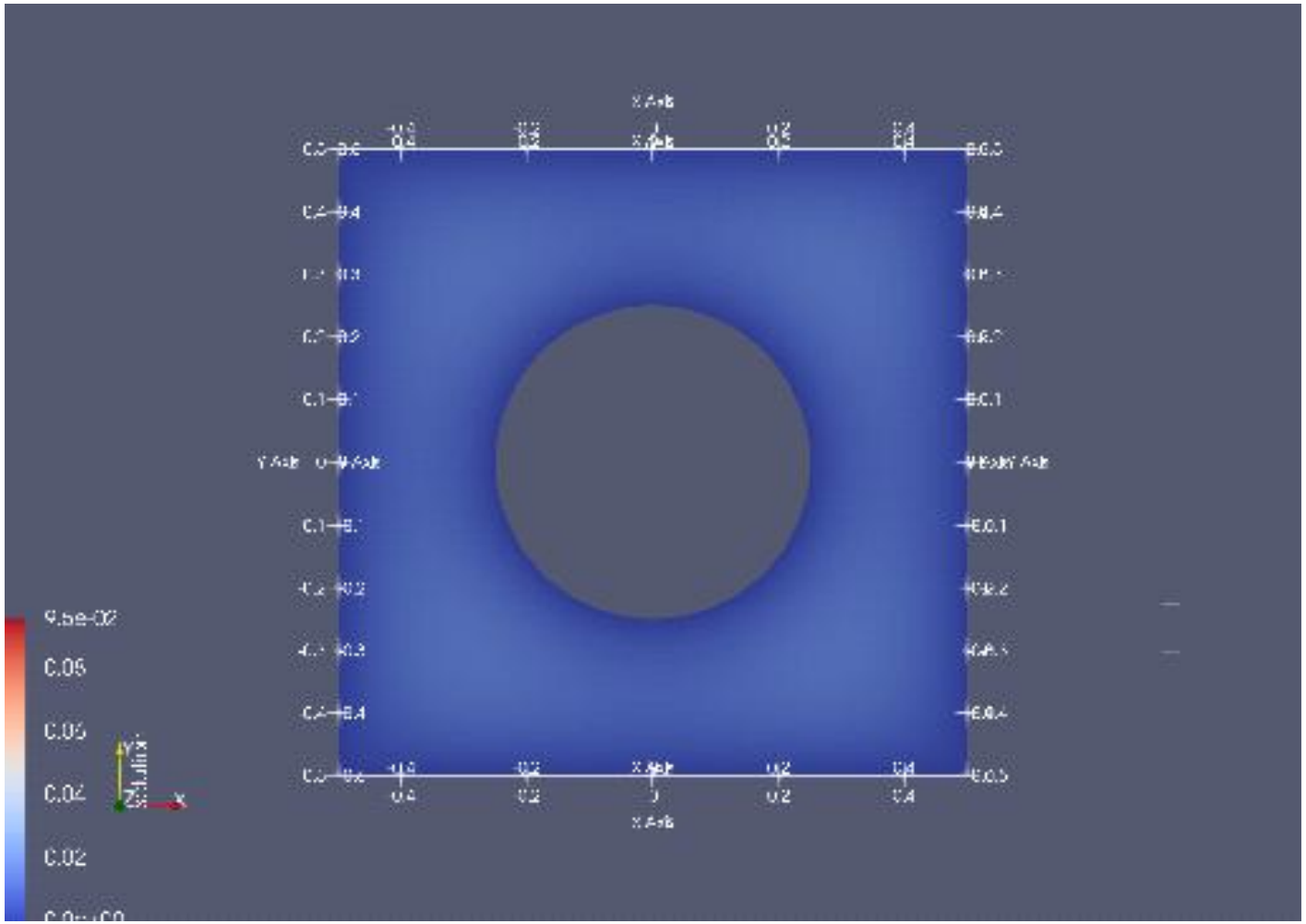
```
void FEMSolve::make_grid()
{
  GridGenerator::hyper_cube_with_cylindrical_hole(triangulation);
  triangulation.refine_global(4);
```

```
std::map<types::global_dof_index, double> boundary_values;
VectorTools::interpolate_boundary_values(dof_handler,
                                         0,
                                         Functions::ZeroFunction<2>(),
                                         boundary_values);
VectorTools::interpolate_boundary_values(dof_handler,
                                         1,
                                         Functions::ZeroFunction<2>(),
                                         boundary_values);
MatrixTools::apply_boundary_values(boundary_values,
                                   system_matrix,
                                   solution,
                                   system_rhs);
```



1 Square with cyliner hole



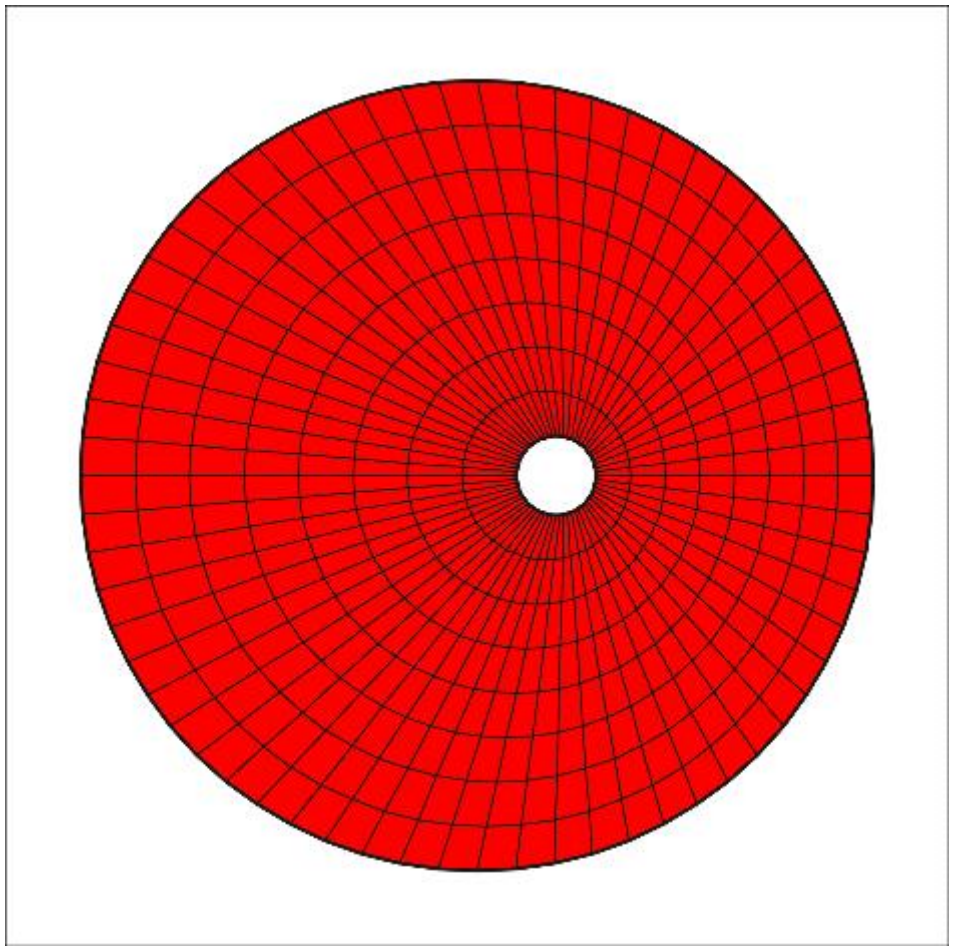


Part 2

Refining mesh and Setting boundary condition

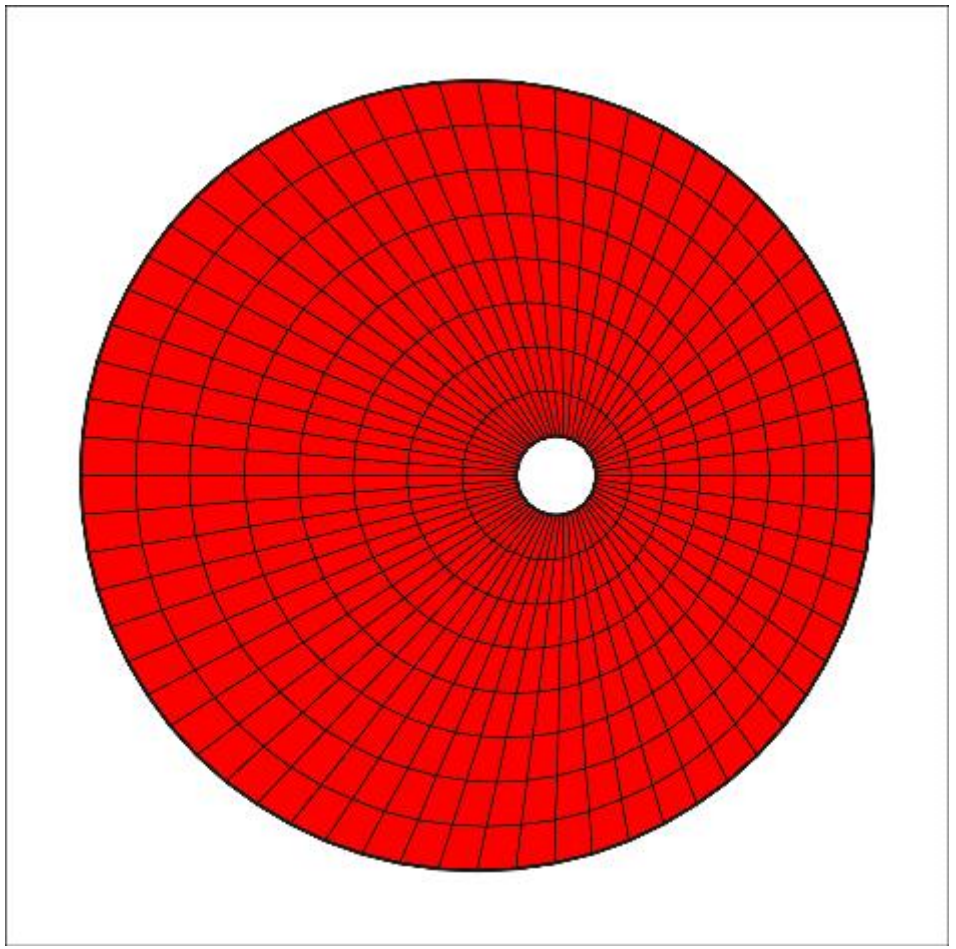
2 Skewed Shell

```
//Initialize Grid
void FEMSolve::make_grid()
{
  GridGenerator::eccentric_hyper_shell(triangulation,
                                       Point<2>(0.2, 0),
                                       Point<2>(0, 0),
                                       0.1,
                                       1.,
                                       8);
  triangulation.refine_global(3);

  std::ofstream out("grid-3.svg");
  GridOut grid_out;
  grid_out.write_svg(triangulation, out);

  std::cout << "Grid written to grid-3.svg" << std::endl;

  std::cout << "Number of active cells: " << triangulation.n_active_cells()
            << std::endl;
}
```

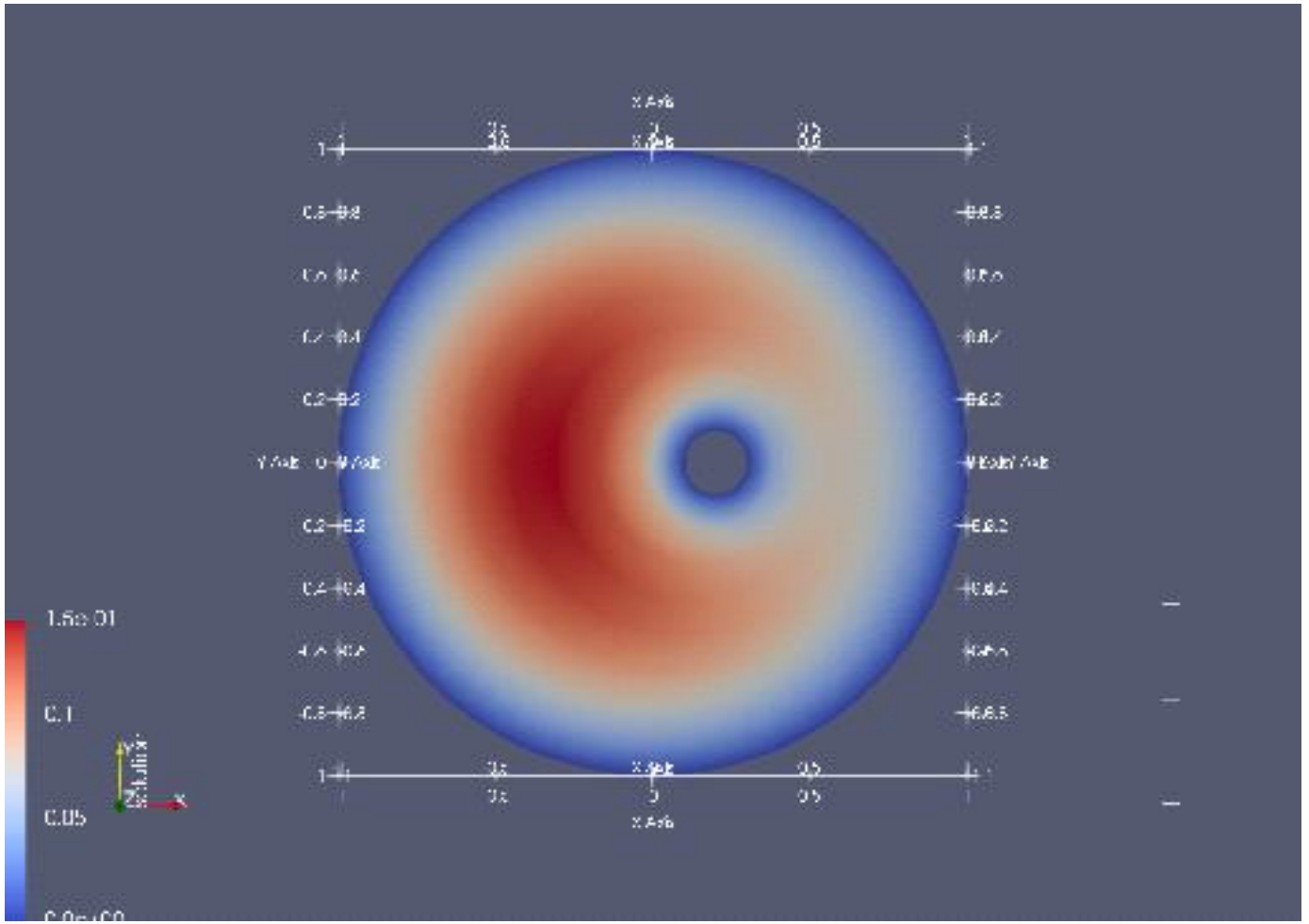


2 Skewed Shell

```
std::map<types::global_dof_index, double> boundary_values;
VectorTools::interpolate_boundary_values(dof_handler,
                                         0,
                                         Functions::ZeroFunction<2>(),
                                         boundary_values);
VectorTools::interpolate_boundary_values(dof_handler,
                                         1,
                                         Functions::ZeroFunction<2>(),
                                         boundary_values);
MatrixTools::apply_boundary_values(boundary_values,
                                   system_matrix,
                                   solution,
                                   system_rhs);
```



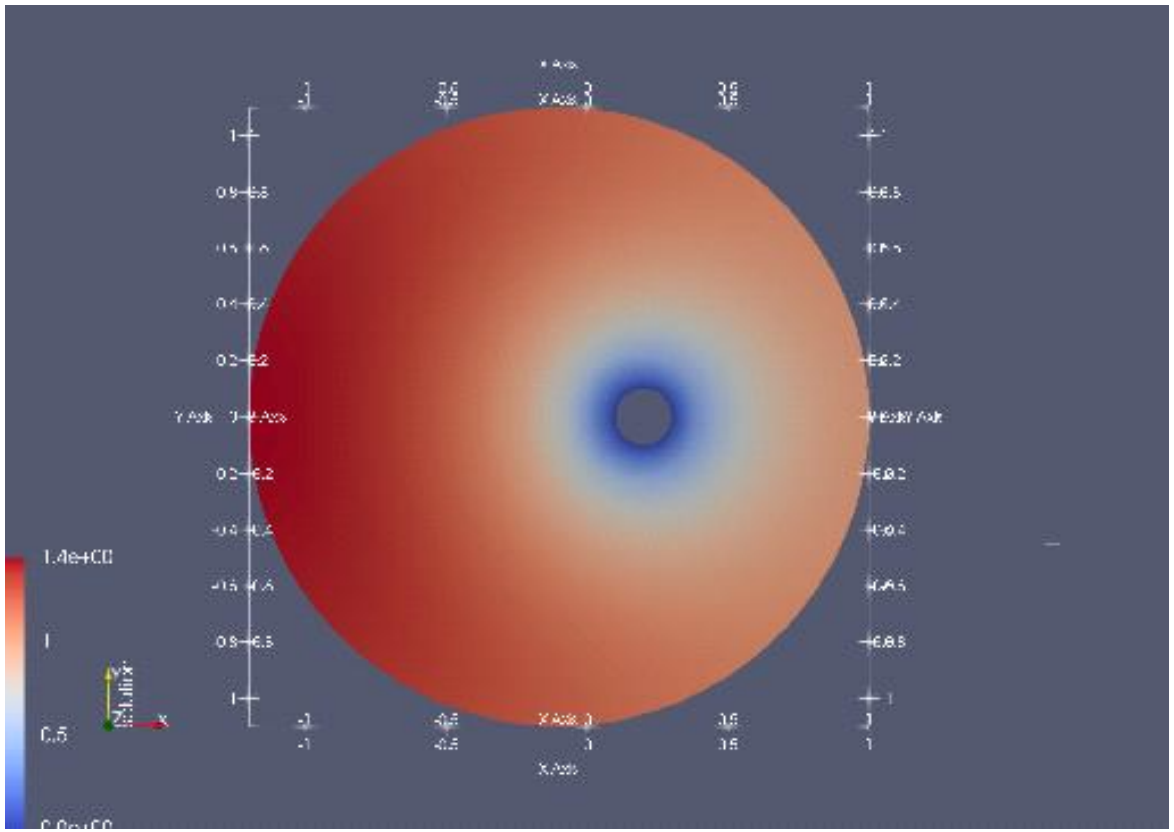
2 Skewed Shell

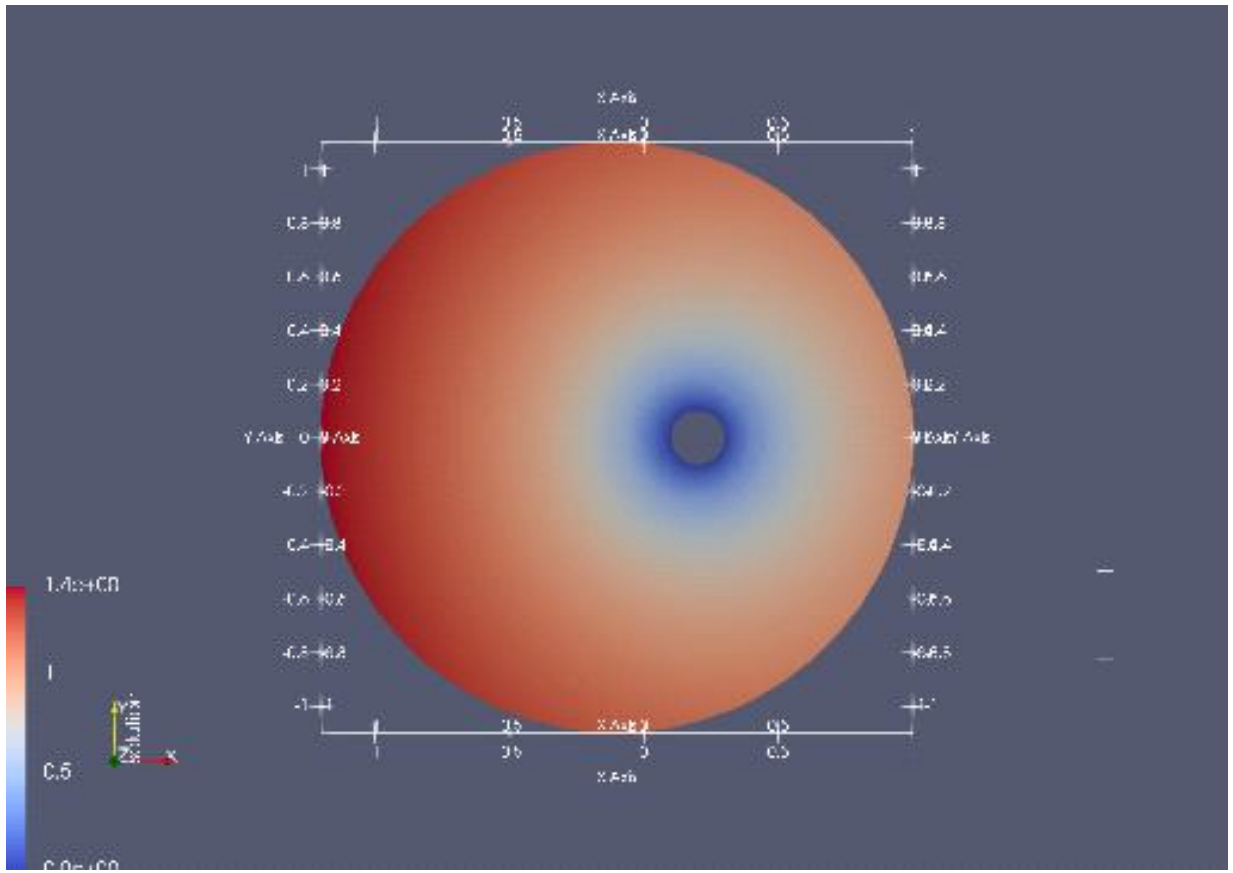


2 Skewed Shell

```
//Boundary
template <int dim> class BoundaryValues : public Function<dim>
{
  public:
    virtual double value(const Point<dim> &p, const unsigned int component = 0) const override;
};

template <int dim> double BoundaryValues<dim>::value(const Point<dim> &p, const unsigned int) const
{
  return p.square();
}
```

2 Skewed Shell





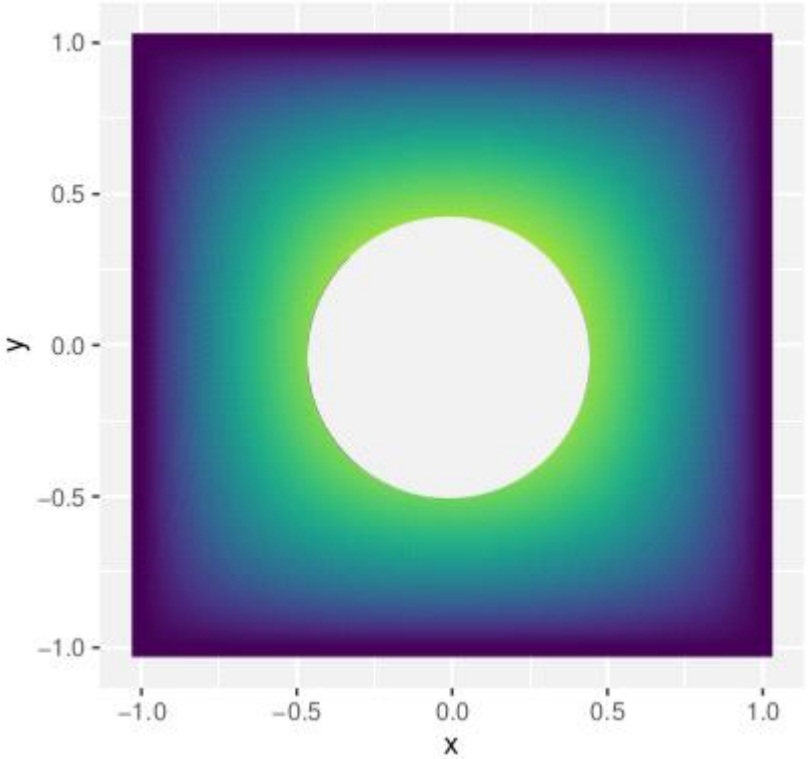
```
for (const unsigned int i : fe_values.dof_indices())
  cell_rhs(i) += (fe_values.shape_value(i, q_index) * // phi_i(x_q)
                  1. *                                // f(x_q)
                  fe_values.JxW(q_index));            // dx
```

```
for (const unsigned int i : fe_values.dof_indices())
  cell_rhs(i) += (fe_values.shape_value(i, q_index) * // phi_i(x_q)
                  0. *                                // f(x_q)
                  fe_values.JxW(q_index));            // dx
```

Part 3
Failure

3 Failure code

```
std::map<types::global_dof_index, double> boundary_values;
VectorTools::interpolate_boundary_values(dof_handler,
                                         0,
                                         Functions::ZeroFunction<2>(),
                                         boundary_values);
VectorTools::interpolate_boundary_values(dof_handler,
                                         1,
                                         Functions::ZeroFunction<2>(),
                                         boundary_values);
MatrixTools::apply_boundary_values(boundary_values,
                                   system_matrix,
                                   solution,
                                   system_rhs);
```



THANK YOU