Geometric Modelling (CSCI-GA 3033-018) - Assignment 4

Avadesh Meduri (N10537558)

April 3, 2021

1 Answers to Exercises

1. Exercise 1 The implementation of the function that computes a smooth interpolation of the vector field by enforcing hard constraints can be found in the jupyter notebook "part_1_2_3.ipynb" with the name "align_field_hard()". The output vector field is identical to the vector field obtained with soft constraints. Figure 1 shows the smooth vector field obtained with hard constraints. The np.savetxt function was used to save the obtained smooth vector field into the file """interpolated_field_hard".

2. Exercise 2 Let A be a diagonal matrix whose elements $A_{i,i} = A_{2i,2i} = A_{3i,3i}$ are the area of the triangle/face i, G be the gradient matrix, u be the desired vector field that is flattened and s be the desired scalar function that is to be computed. Then the desired least squares problem can be defined as follows

$$\min (Gs - u)^T A(Gs - u)$$
$$\min s^T G^T AGs - 2s^T G^T Au + u^T Au$$

   hence the $K = G^T AG$ and $b = -2G^T Au$ by comparing coeffecients.
   By differentiating the above equation and setting to zero we get the optimal s

$$G^T AGs = G^T Au$$

   which can be solved using a linear solver. The implementation of the above least squared problem to compute the desired scalar function is in the jupyter notebook "part_1_2_3.ipynb". In order to handle the null space in the K matrix, the last element of the s vector is set to zero (constrained). Thus the resulting problem is solved.
   The gradient of the solved scalar field along with the poission error is shown in figure 2. The norm $Gs - u \leq 10$ and the gradient was indeed zero which verified that the problem was solved correctly. As can be seen from the figure a perfect overlap could not be achieved. The reconstructed scalar function was dumped using np.savetxt into the file "scalar_function".

3. Exercise 3 The implementation to compute both the harmonic and LSCM parametrization can be found in the jupyter notebook "Part_3.ipynb" Figure 3 shows the UV mapping using harmonic parameterization. Also, the gradients of the u function, computed using the grad = igl.grad(v,f)*uv[:,0] is also shown. In Figure 4 similar results are shown using LSCM paramterization.

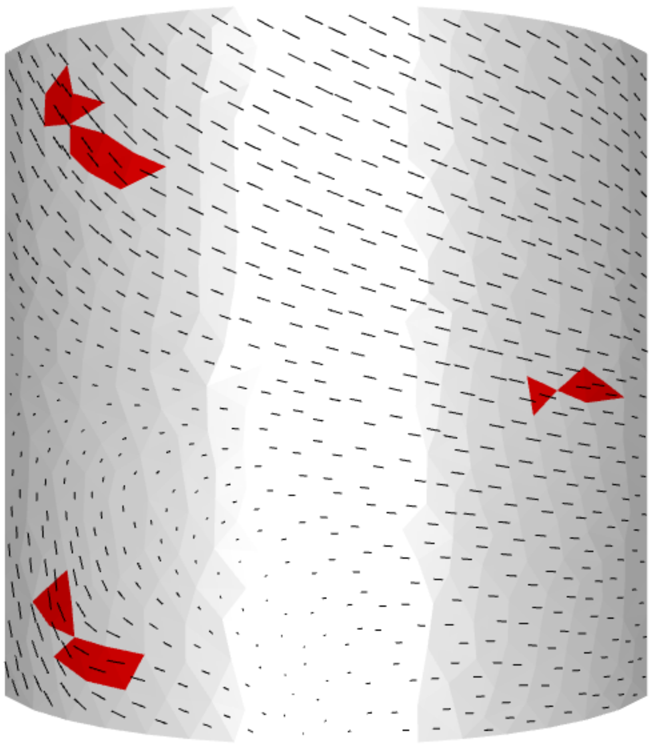


Figure 1: smooth vector field obtained with enforcing hard constraints.

4. Exercise 4 The implementation of this exercise can be found in the jupyter notebook "part_1_2_3.ipynb". Initially the uv map using harmonic paramterization is computed for the cylinder mesh. Shown in figure 5. After this, the v map is replaced with the computed smooth scalar function s (uv[:,1] = s). The resulting UV mapping (with checker board) is shown in Figure 6. In Figure 7, The gradient vector of the original V function along with the gradients of the custom scalar function are plotted.

   The triangles that are flipped due to the update in uv paramterization are shown on the mesh in Figure 8. The red triangles are flipped. The flipped determined by first computing the normals of the original triangle in the UV domain and the normals of the triangles with the updated UV mapping. If the normals are negative in the new mapping, while the normals in the original UV mapping are positive, then the vertices in the new UV mapping are clockwise. Which means that the triangle is flipped. The implementation can be found in the same notebook. The indices of the flipped triangles are stored using np.savetxt in the file "flipped_triangles"

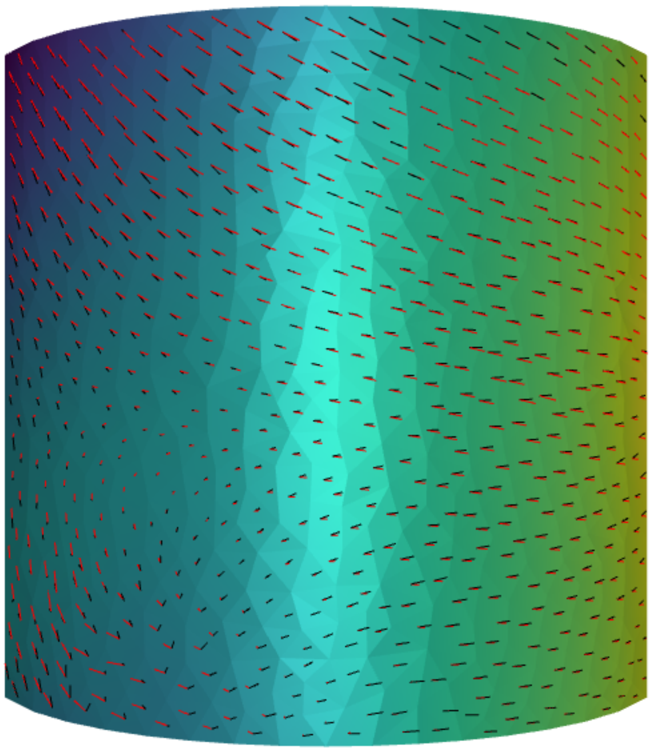


Figure 2: Gradient field of the solved scalar function (in black) along with the desired vector field (in red). the color map is used to plot the computed scalar function.

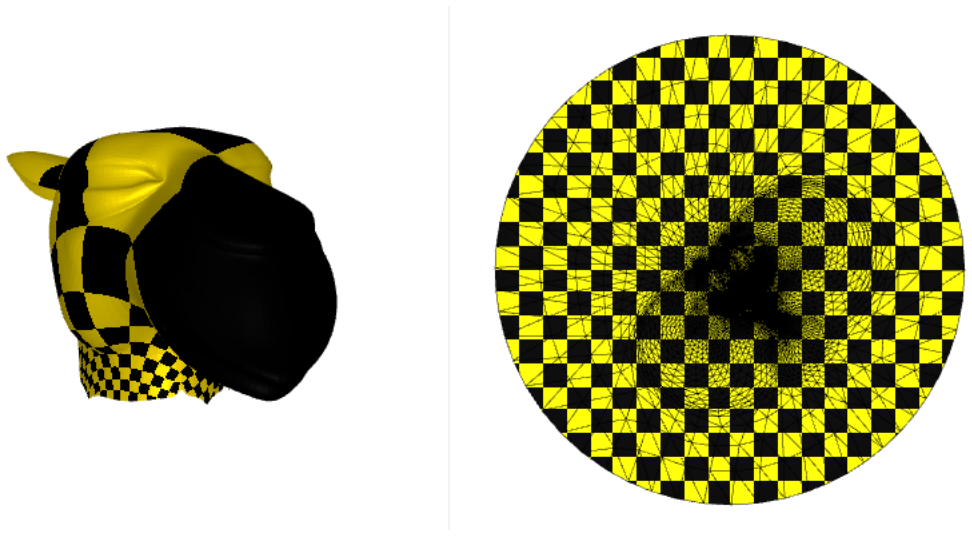
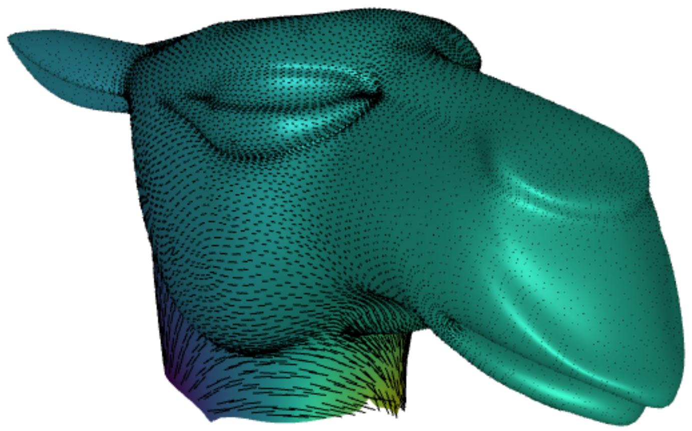


Figure 3: UV mapping onto a circle using harmonic parameterization. The gradient lines along with color map for u function is also shown



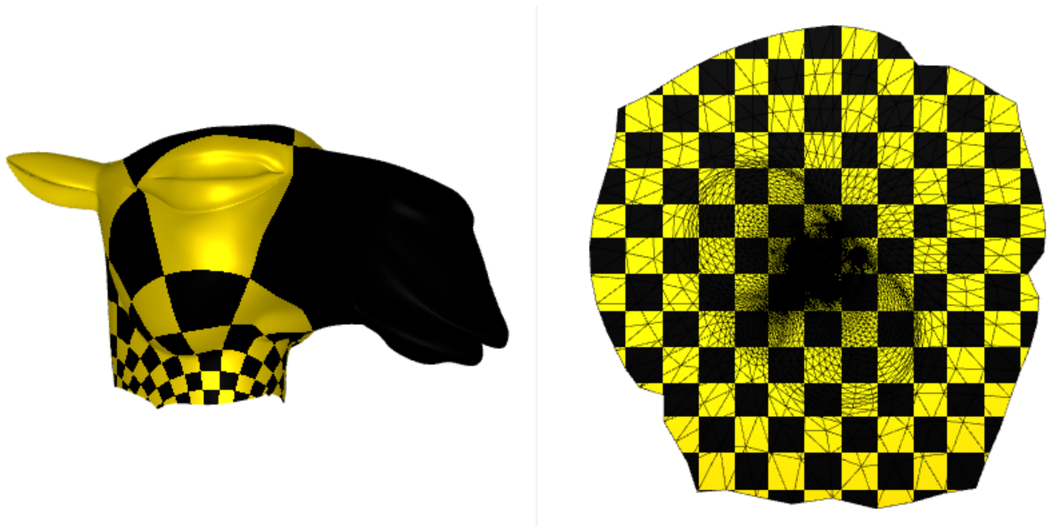
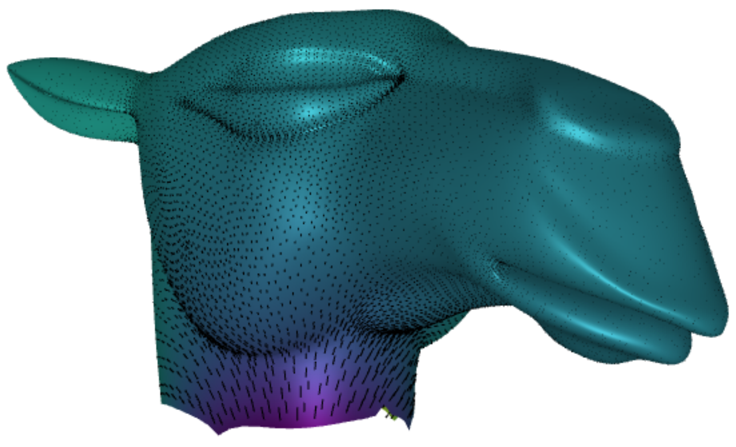


Figure 4: UV mapping onto a circle using LSCM parameterization. The gradient lines along with color map for u function is also shown



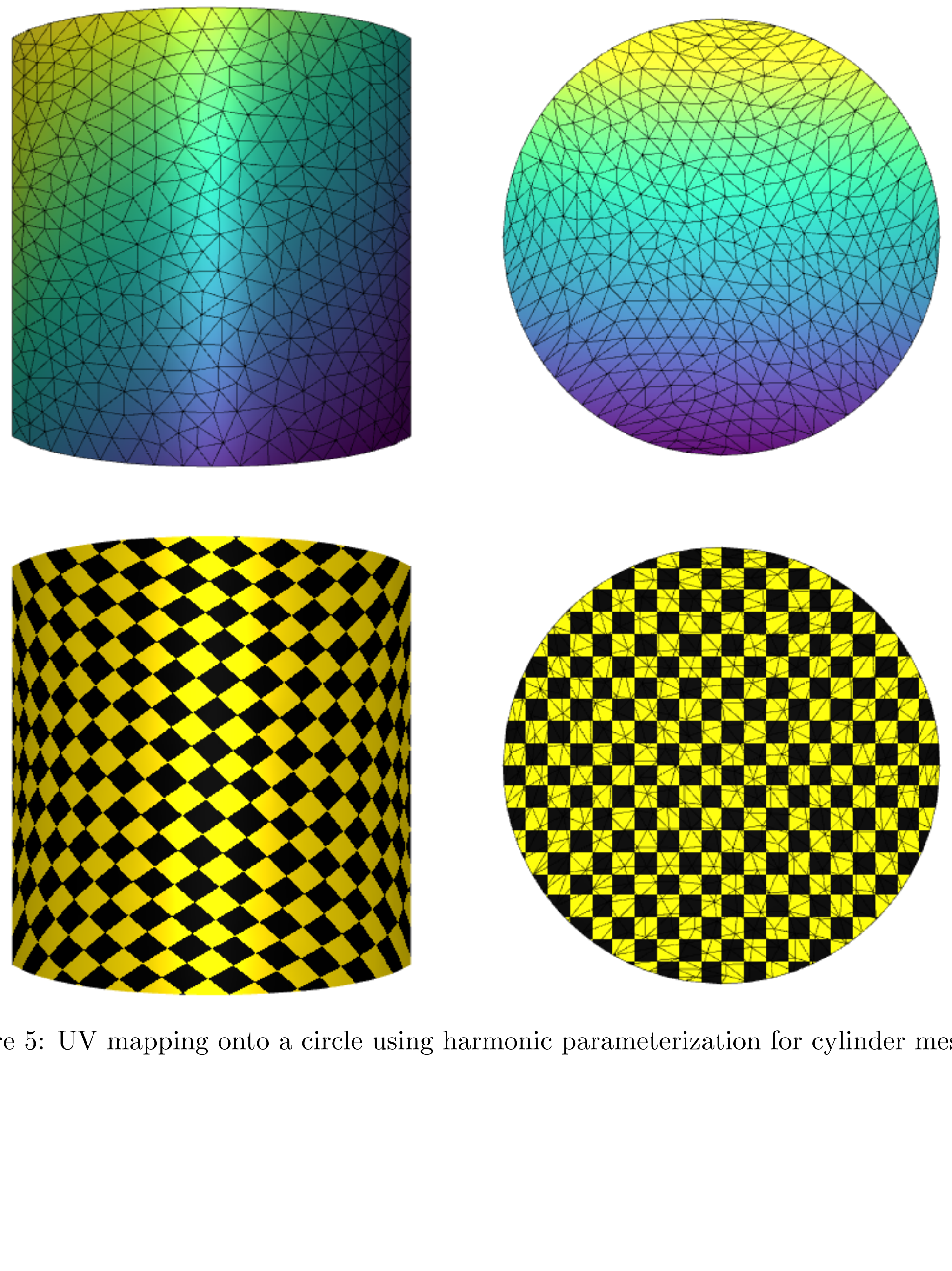


Figure 5: UV mapping onto a circle using harmonic parameterization for cylinder mesh

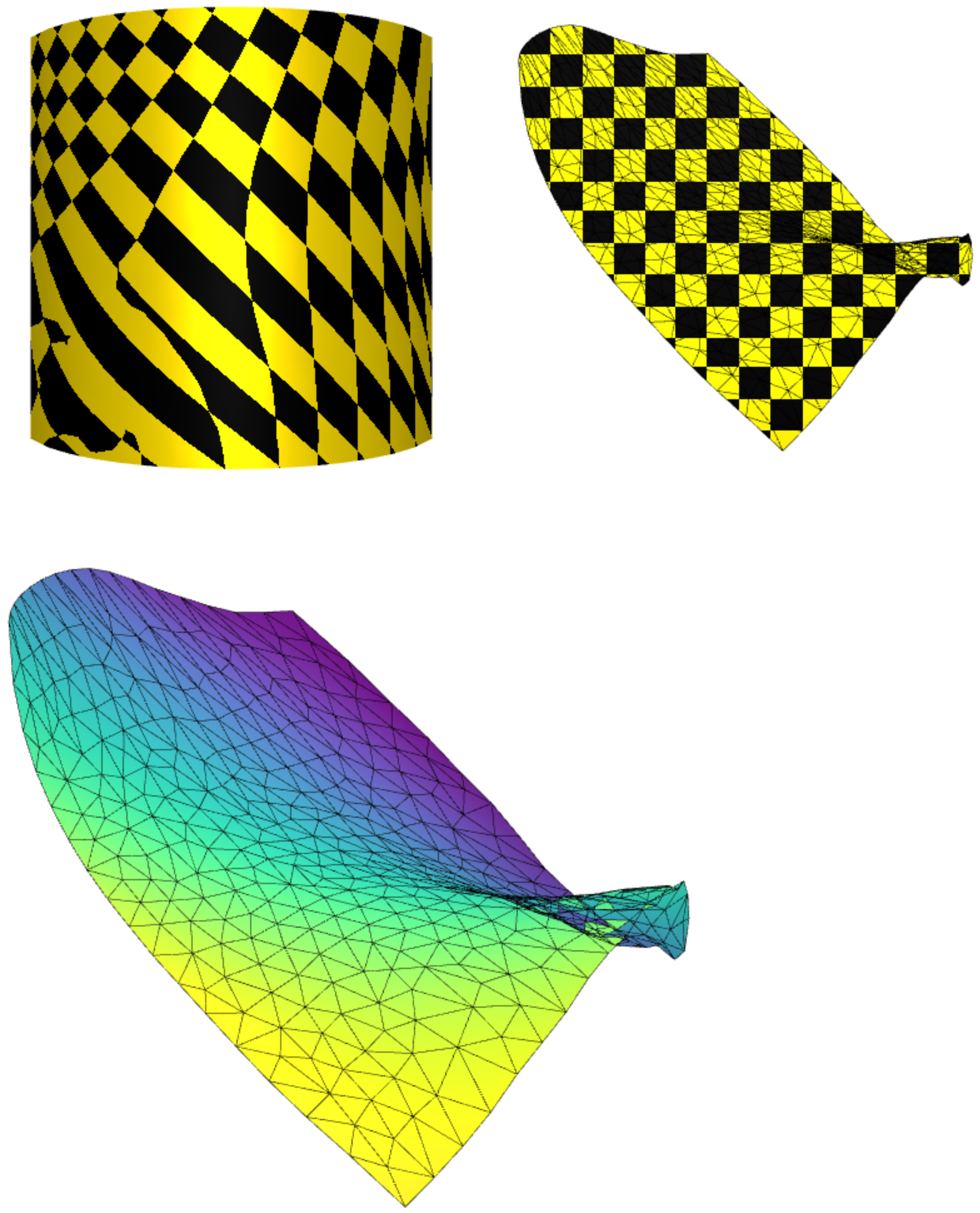


Figure 6: UV mapping with V function replaced with smooth scalar function

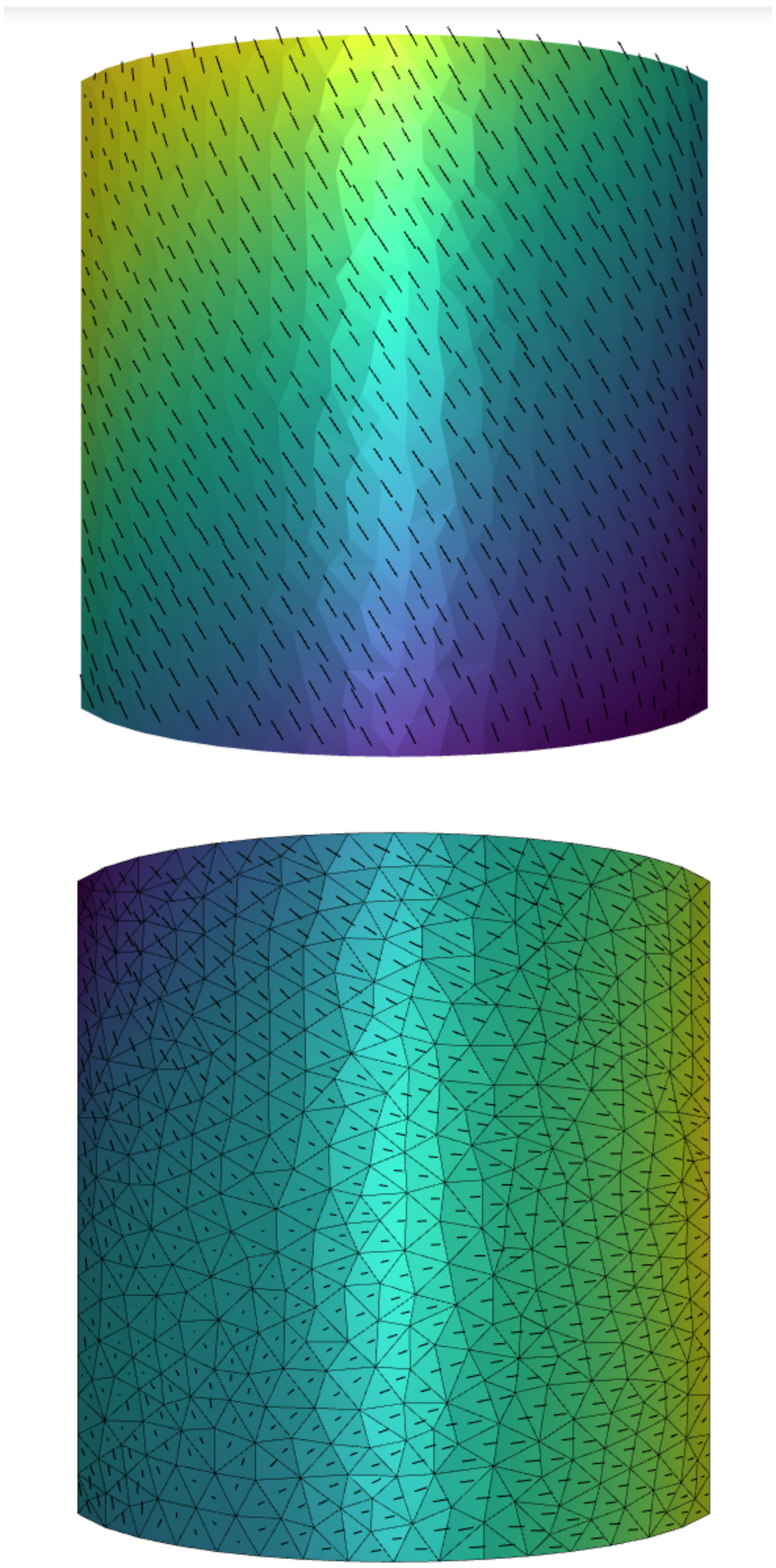


Figure 7: uv function mapping with custom gradients. Top figure shows gradients of the V function along with color map of scalar values. The bottom figure shows the gradients of the custom scalar function with color map.

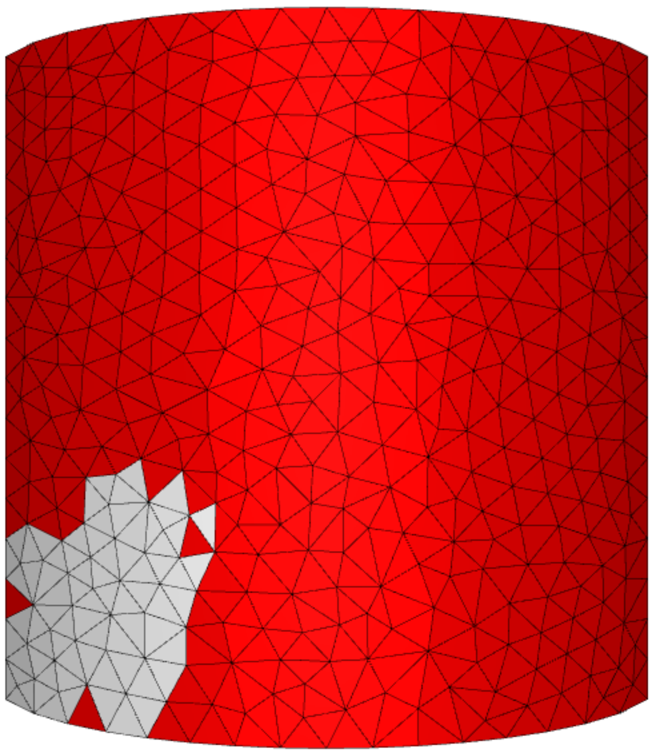


Figure 8: Mesh showing flipped triangles in red.