IT575 Computational Shape Modeling **Assignment - 3**
Name - Vaibhav Amit Patel Name - Maharshi Vyas
ID No. 201401222 ID No. 201401414
Date of Submission: 10 March 2017

Laplace Beltrami operator

MATLAB codes for the LBO are provided on the course website, one using the weak formulation, while other using the Divergence theorem. Moreover, a few meshes have also been provided; demonstrate your solutions for the following questions on at least one mesh.

1. Implement the heat equation using either explicit or implicit discretization of the heat equation. Show sample plots of the solution, with heat distribution being randomly initialized.

   Answer : Heat diffusion equation:

   $$U_i \leftarrow U_i + 1 + \lambda \Delta U_i$$

   Heat values on each vetices are initialized with -20 to 20. Lambda values is taken as 0.5. We have used implicit method to update heat values. After, nearly 30 iterations, most of the values tend to converge to 0.

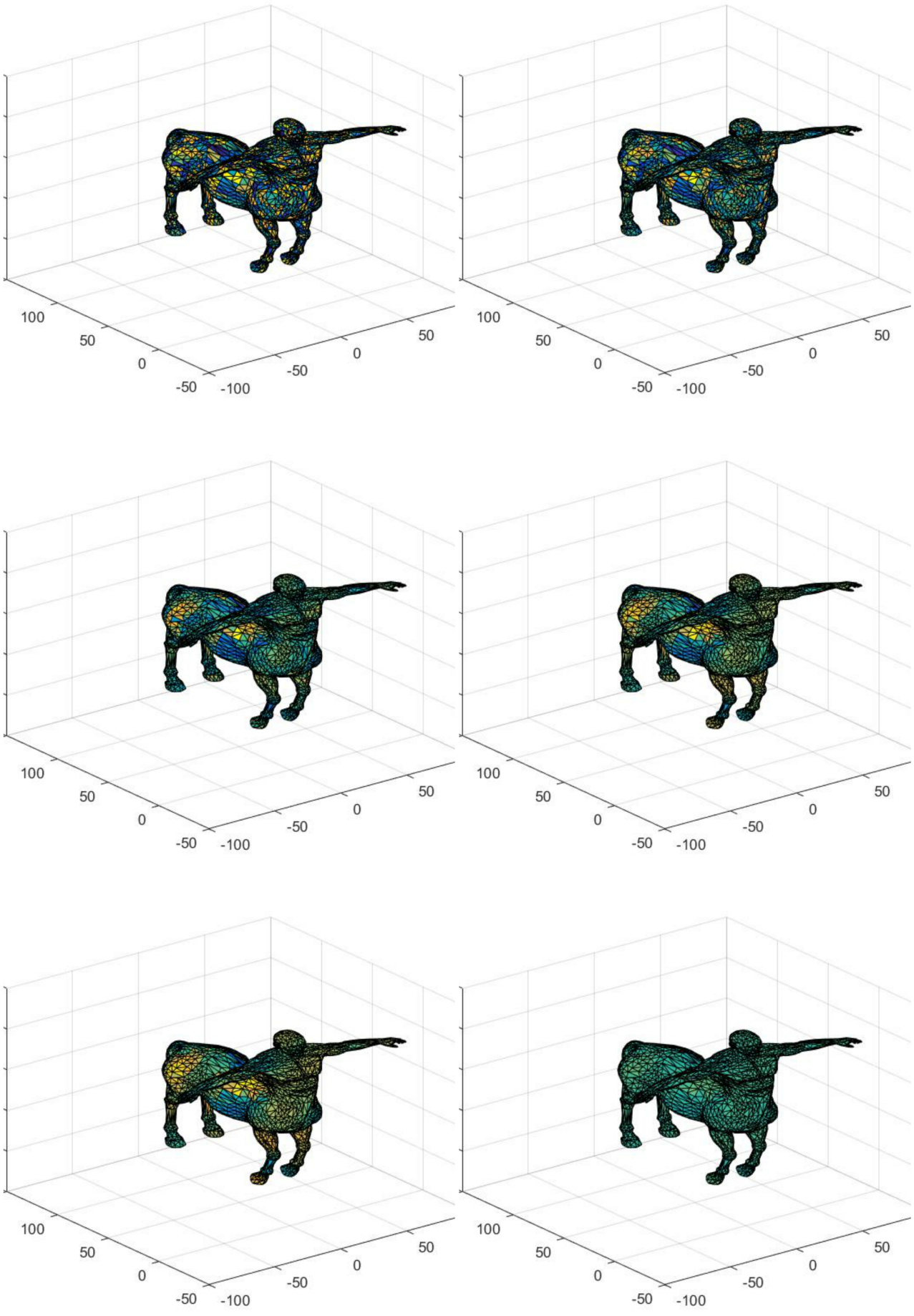

Figure 1: **Heat Diffusion on any random surface**

2. Instead of using a function u on the mesh, what happens if the heat solution is computed for the coordinate functions x, y and z, i.e, $\frac{\partial x}{\partial y} = \Delta x$, and similarly for y and z. Plot a few meshes thus obtained.

   Answer :

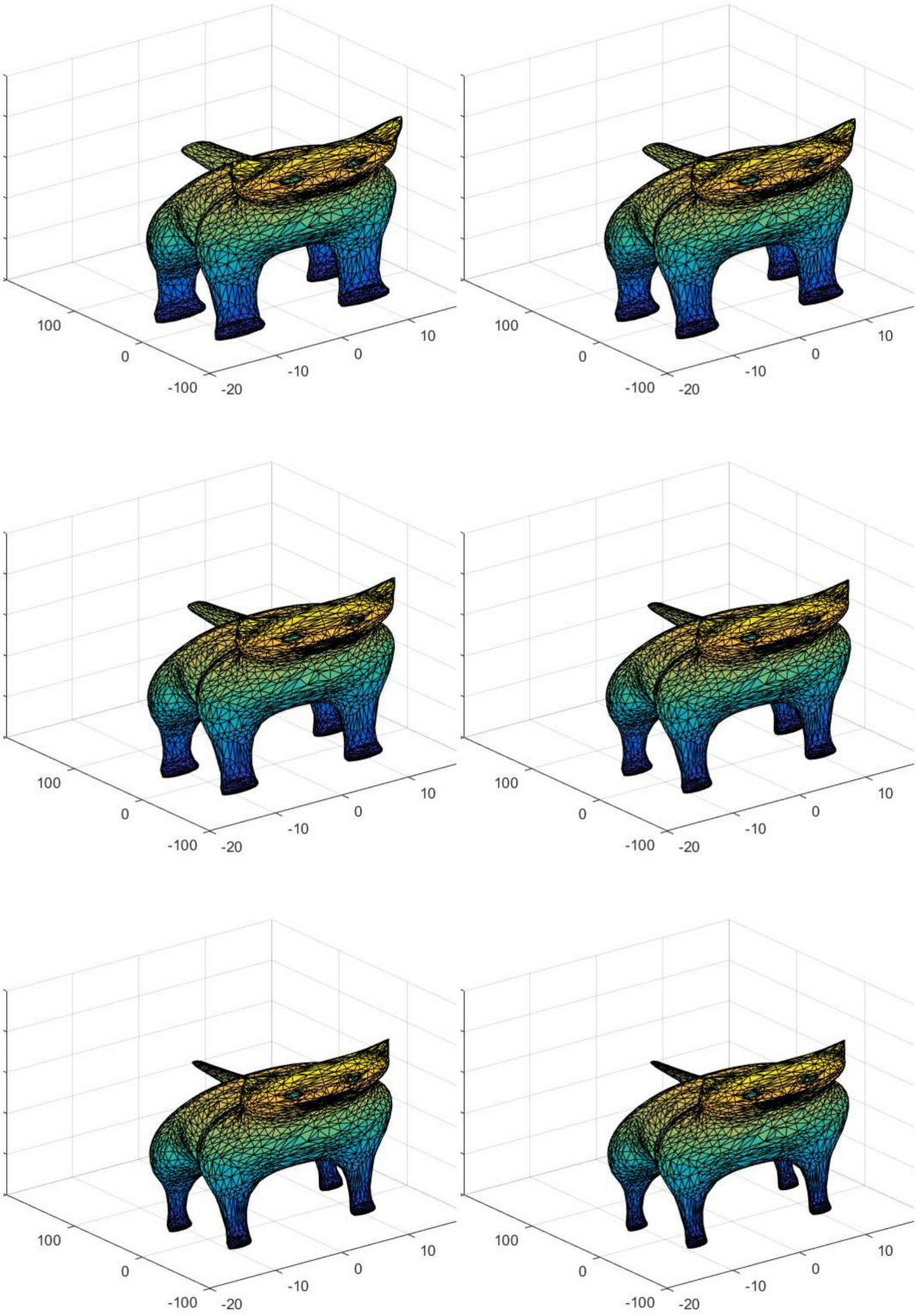

Figure 2: **Smoothing (Here Shrinking) of surfaces**

Here, instead of any arbitrary function, we apply updates to co-ordinates of the surface.The so called Laplacian smoothing looks like,

$$x_i \leftarrow x_i + \lambda \Delta x_i$$

Since the Laplace Beltrami of vertex positions corresponds to the mean curvature normal (), all the vertices move in the normal direction by an amount determined by the mean curvature H [1]. As done in question 1, laplace beltrami operator is taken from *computeLaplaceBeltrami.m* (cotangent form) and is used as $-L$ in the matlab code . The given sampling of the surface already has less noise. After applying above updates to co-ordinates, surface will be more smooth and will eventually start shrinking. Shrinking of legs and tail are clearly seen in the above figures.
Laplace Beltrami Operator will change after each update of co-ordinates. Results do not alter majorly even if we do not update LBO with each iteration, the above results are generated without updating LBO operator and below results(question 3b) are generated with update of LBO in each iteration.

3. Let us formulate the problem of denoising/smoothing of functions as a cost minimization process. Let f be the given noisy function on mesh M with LBO L (assume L is an approximation to Δ). The cost function is $C(g) = ||f-g||^2 + \lambda g^t L g$. The denoised signal g^* is defined as $g = argmin C(g)$. The first term in the cost function, the data term prevents the denoised signal g from being too different from f , while the second term, the prior/regularizer term promotes signal smoothness. Find a gradient descent procedure for minimizing C, implement it and show a few intermediate plots of this process, assuming a

   (a) random initialized function on a mesh, and

   Answer :

$$C(g) = ||f-g||^2 + \lambda g^t L g$$

$$\frac{\partial C}{\partial g} = -2(f-g) + (L^T + L)g$$

$$g := g - \alpha \frac{\partial C}{\partial g}$$

   To start, we have taken a function f having values randomly between -20 to 20. The function g have random noise (between -2 to 2) added to f. There will be very small difference if we visualize function f and corresponding smooth function g. Following figure shows f and g values at each vertex. We can see, there are more random values in figure showing values of function f, while values of g are very smooth(less abrupt differences between neighbouring triangles) and have much less random noise (dark blue color is distributed mostly).

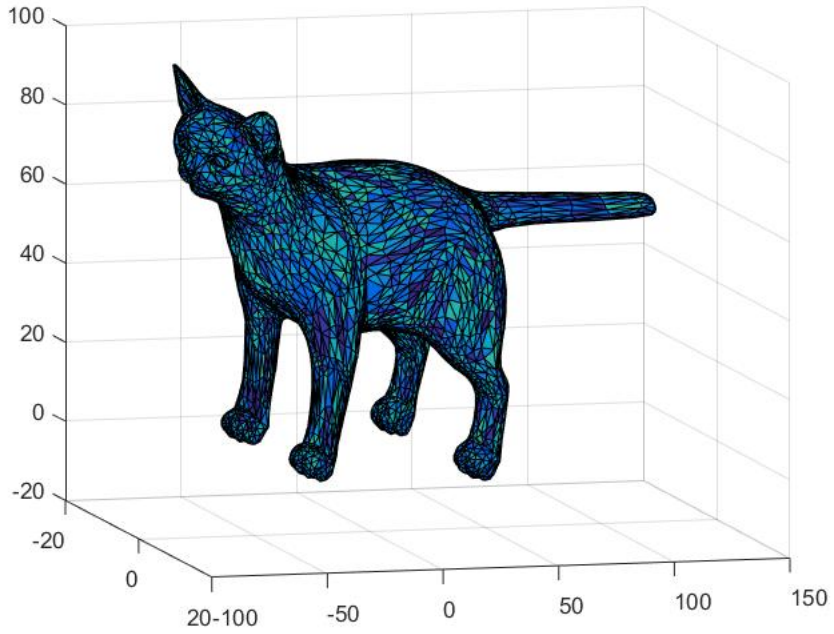

Figure 3: **Color intensity shows function values of** f

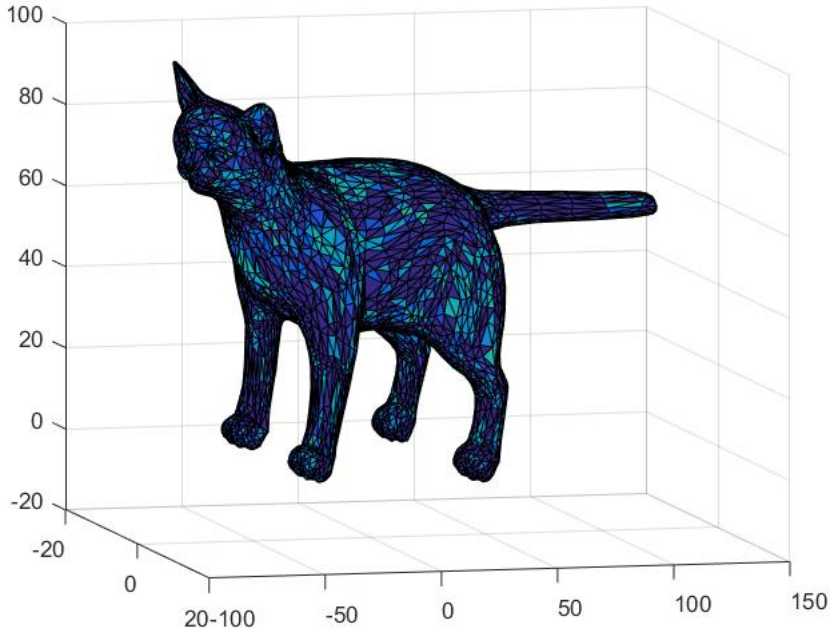

Figure 4: **Color intensity shows function values of** g

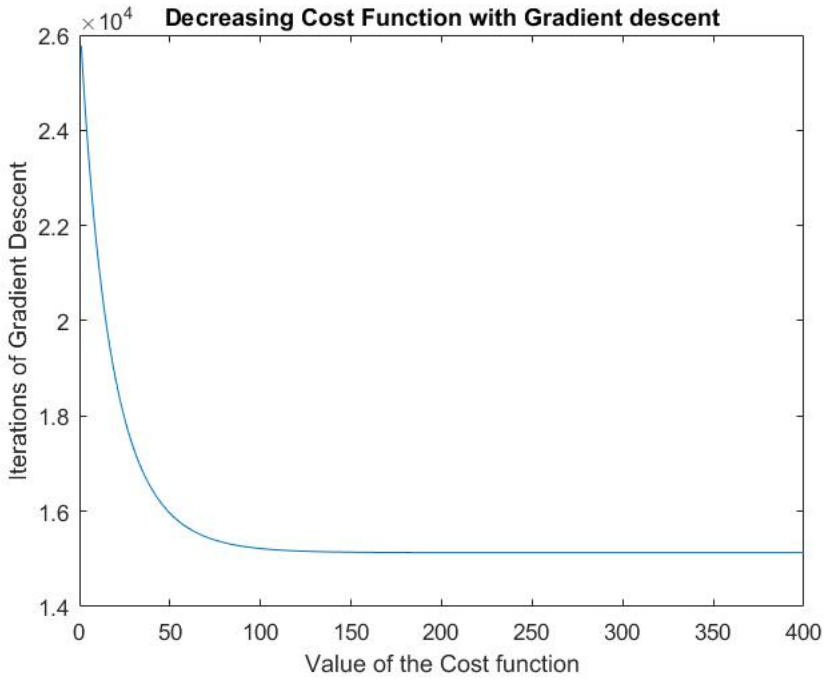
Decreasing Cost Function with Gradient descent
×10^4
2.6
2.4
2.2
2
1.8
1.6
1.4
Iterations of Gradient Descent
0
50
100
150
200
250
300
350
400
Value of the Cost function

Figure 5:

(b) coordinate functions x , y and z. Justify that the second term promotes smoothness.

Answer : Applying updates to minimize the cost function to each co-ordinate leads to surface smoothing and noise reduction.To show that, we started with adding some random noise(-3 to 3) to the initial co-ordinate values. Then updating it such that the cost function is minimized leads us to smoothing as showed in following figures.
Below problem reduces to what value of lambda we take. Taking higher values of lambda(i.e. 0.1 in our case) lead to high distortions in the surface,so a proper value of lambda has to be chosen. Lambda is taken 0.001 here and alpha = 0.1, which makes the surface smooth in around 30 iterations.

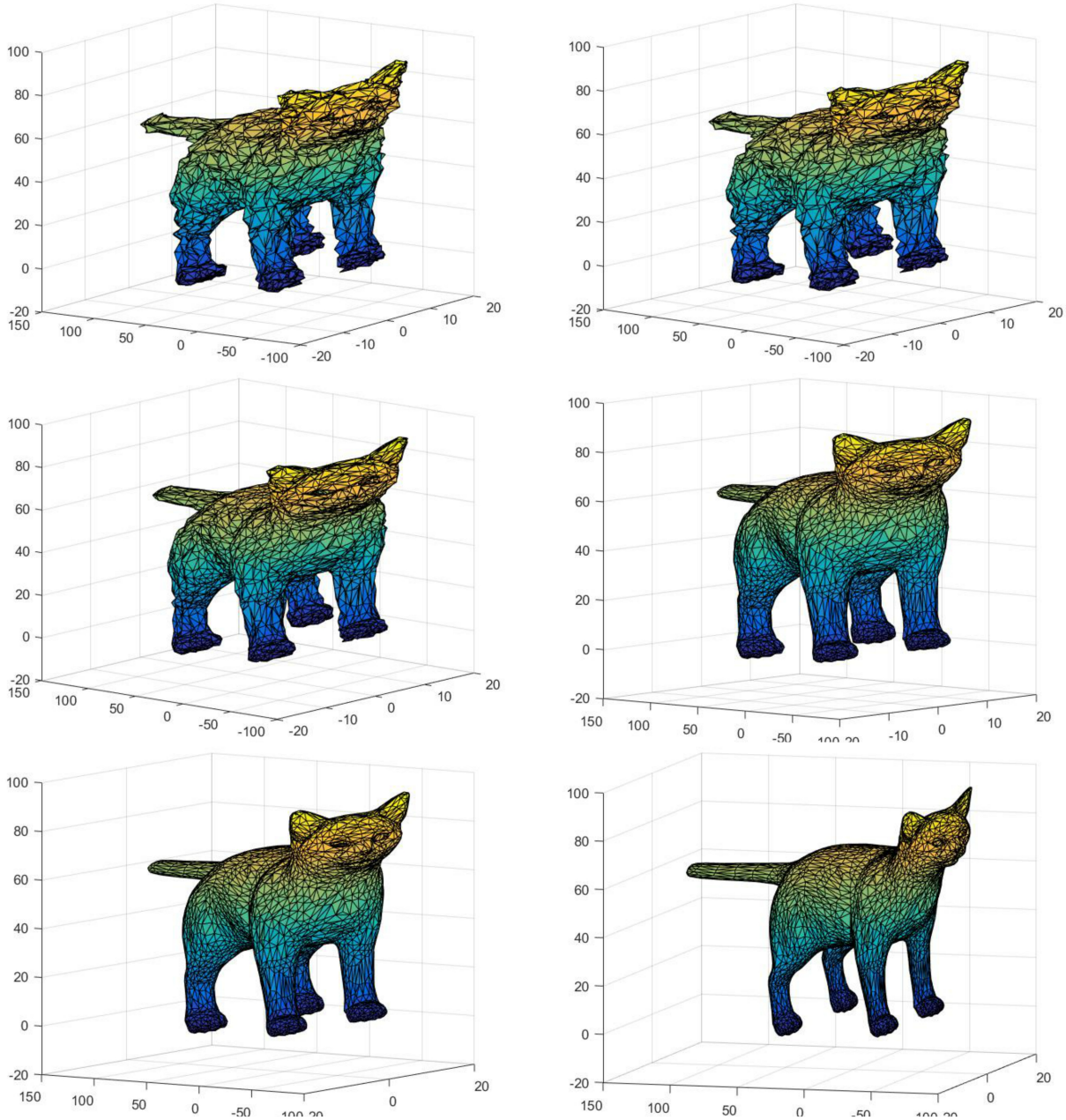

Figure 6: **Smoothing and noise reduction on Surfaces**

We tried several iterations for lambda = 0.1, it turns out that smoothing of co-ordinates in question 2 is very different from this problem. In question 2, co-ordinates moved towards normal of the surface while here, co-ordinates move such that surface becomes smooth and keeps co-ordinates as close as f(else first term of equation will penalize the cost function).

4. Let M be the mesh vertex coordinate array of a given mesh, while M_i be the mesh vertex coordinates of the mesh obtained by preserving i percentage of LBO eigenfunction coefficients corresponding to mesh M. Find out the percentage of coefficients to be preserved so that the error:

$$e = \frac{||M - Mi||}{||M||}$$

where $\|\cdot\|$ is the usual $\mathbb{R}^3$ norm, is below 0.05. Show a few plots for M_i , and plots for a few eigenfunctions of the mesh M.

Answer : If U is the matrix containing eigenenvectors of LBO, then taking k eigenenvectors with the largest eigenvalues as $U_{reduced}$ Compressed co-ordinates c is,

$$c = U_{reduced}^T AV$$

where V is,

$$V = [x\ y\ z]$$

i.e. x,y,z co-ordinates of all the vertices and A is area wights matrix of triangles

Reconstructed co-ordinates R is given by,

$$R = U_{reduced} C$$

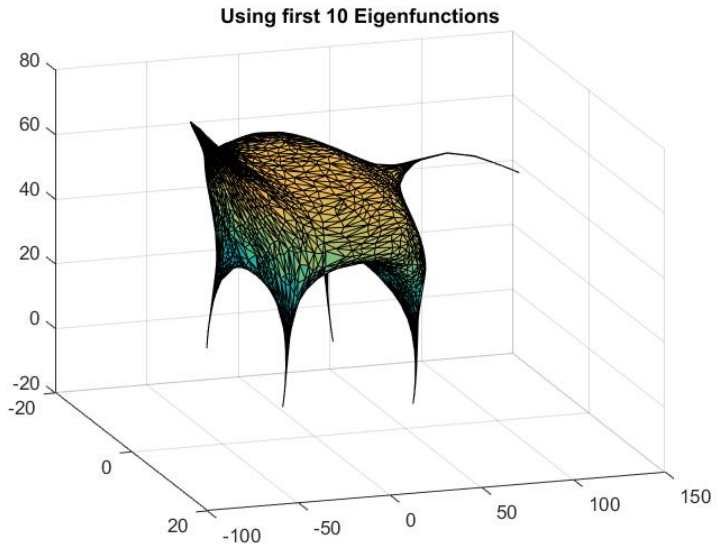

Figure 7:

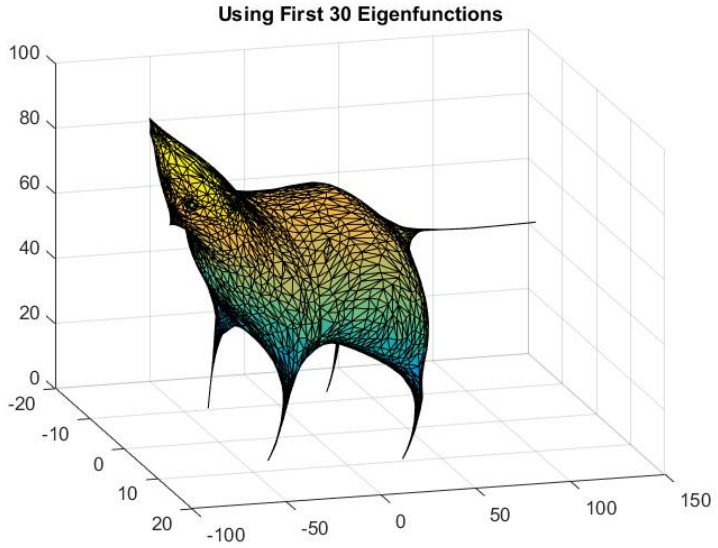

Figure 8:

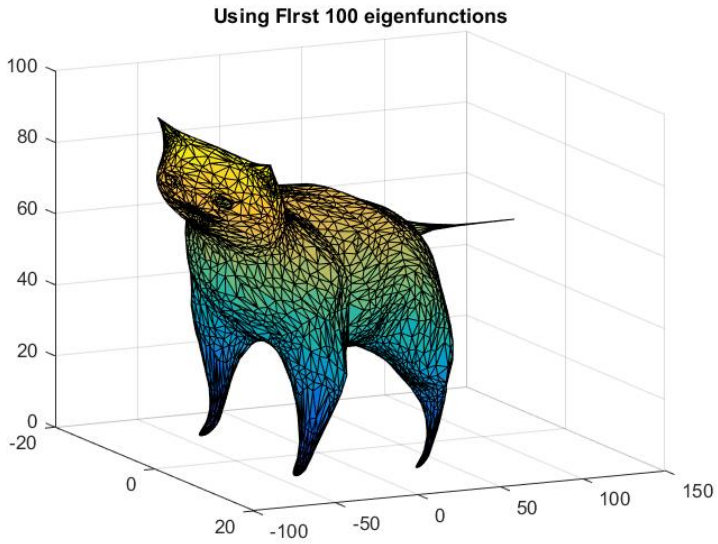

Figure 9:

Taking first 20 eigenvectors generate error of 0.0502, so to have error below 0.05, we should take 20 or more eigenvectors to compress the image.

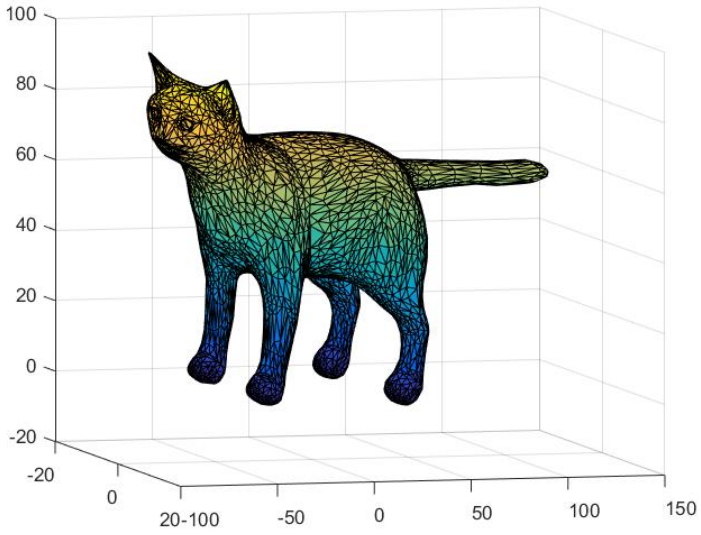

Figure 10:

References

[1] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, and Br uno Levy. *Polygon Mesh Processing*. AK Peters, 2010.