CS544 – MP 2

Zih-Siou Hung
zhung2@illinois.edu

Patrick Cole
pacole2@illinois.edu

Rajarshi Haldar
rhaldar2@illinois.edu

Yuxuan Zhou
yuxuanz6@illinois.edu

03/29/2019

Objective

MP2 is divided into the following parts:

1. Use the augmented Lagrangian method to find a surface that minimizes smoothness cost while interpolating points given below; You should represent the surface as a height map on a 256 x 256 grid, representing the unit square. Represent the surface as a height map. Measure the smoothness as the norm of the gradient of the height map. The points to interpolate are: (0, 0, 1); (0, 1/2, 0); (0, 1, 1); (1/2, 0, 0); (1/2, 1/2, 1); (1/2, 1, 0); (1, 0, 1); (1, 1/2, 0); (1, 1, 1). Use a second order method to do the inner optimization.

2. Compare this solution with the solution obtained above by solving a linear system.

3. Construct a surface of minimum area that interpolates these points and the line segments on the grid joining them Use the augmented Lagrangian method.Plot the surface we get.

4. Minimize the surface area of the interpolating surface that interpolates only the vertices. Plot the surface we get.

We will analysis each part with our diagrams and code the following pages.

1 Implementation

Within this section we discuss the various implementations and our process for getting the results in the next section.

1.1 Minimize Smoothness with ALM

For a surface of minimum smoothness we wanted to minimize the following function $f(V_h)$, where V_h is a vector of the height and of length 256^2, flatten in row-column format.

$$f(V_h) = \left|\left|\frac{\partial h}{\partial x}\right|\right|_2^2 + \left|\left|\frac{\partial h}{\partial y}\right|\right|_2^2$$

Because we are dealing with discrete data we can define some matrices A_x and A_y, both of shape $256^2 \times 256^2$, such that $A_x V_h \approx \frac{\partial h}{\partial x}$ and $A_y V_h \approx \frac{\partial h}{\partial y}$. Now our cost function can be written as follows:

$$\begin{aligned} f(V_h) &= V_h^T A_x^T A_x V_h + V_h^T A_y^T A_y V_h \\ &= V_h^T \left[M\right] V_h \end{aligned}$$

We define A_x and A_y as sparse matrices that take the difference between consecutive column values or row values. We also had to be careful of the edges cases and to deal with those we just zeroed out the rows corresponding to the edge cases.
It also follows that we can derive our gradient:

$$\frac{\partial f}{\partial V_h} = 2\left[M\right] V_h$$

Next we needed to define the constraints. For this we used a matrix L of shape 8×256^2. We need use the rows as one hot arrays for each point constraint. We also have a vector c of length 8, which just holds the values for the corresponding height of the points specified by A. Our constraint can be seen as followed.

$$LV_h = c$$

Given all of these we can just plug them into the Augmented Lagrangian Method (ALM) and run for a few iterations. For the purpose minimization for each iteration in the algorithm we used `scipy.optimize.minimize` with L-BFGS. We were able to get pretty good results from just 5 iterations of ALM.

1.2 Minimize Smoothness with Least Squares

The problem above can also be solved by a linear system. For the following formula I will reuse M, V_h, L, and c from Section 1.1.

$$\begin{bmatrix} M & L^T \\ L & \mathbf{0} \end{bmatrix} \begin{bmatrix} V_h \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ c \end{bmatrix}$$

Now we can solve this linear system for our height vector, V_h, and our Lagrange multipliers, λ.

1.3 Minimize Surface Area with ALM - Line Constraints

Following the notation from section 1.1, we want to minimize:

$$f(V_h) = \sum_i \sqrt{1 + \left(\frac{\partial h}{\partial x}\right)_i^2 + \left(\frac{\partial h}{\partial y}\right)_i^2} \,,$$

to get the minimum surface area. Similar to section 1.1, we can approximate $\frac{\partial h}{\partial x}$ with $A_x V_h$ and $\frac{\partial h}{\partial y}$ with $A_y V_h$. Thus, our cost function $f(V_h)$ can be rewritten as follows:

$$f(V_h) = \sum_i \sqrt{1 + (A_x V_h)_i^2 + (A_y V_h)_i^2} \,.$$

To speed up the computation of L-BFGS, we have to calculate the gradient with respect to V_h. Let's define matrix C and D as follows:

$$C = 2 \cdot \mathrm{Diag}\left(\frac{1}{\sqrt{1 + (A_x V_h)^2 + (A_y V_h)^2}}\right)$$
$$D = 2 \cdot \left[\mathrm{Diag}\left(A_x V_h\right) \cdot A_x + \mathrm{Diag}\left(A_y V_h\right) \cdot A_y\right],$$

We could get

$$\nabla f(V_h) = \sum_i \left(C \cdot D\right)_{i,j} .$$

Now, we need to enforces the point and line constraints. Similar to section 1.1, all of this constraint can be written as a linear system:

$$L' V_h = c.$$

Given the objective, gradient and the constraint, we can then pass them in to the Augmented Lagrangian Method.

1.4 Minimize Surface Area with ALM - Point Constraints

This problem is mostly the same as section 1.3 except that we remove the line constraint. We can then use our usual Augmented Lagrangian Method to optimize it.

2 Results and Analysis

Within this analysis, we show the results from our different optimization problems and briefly our interpretation of them.

2.1 Minimize Smoothness with ALM

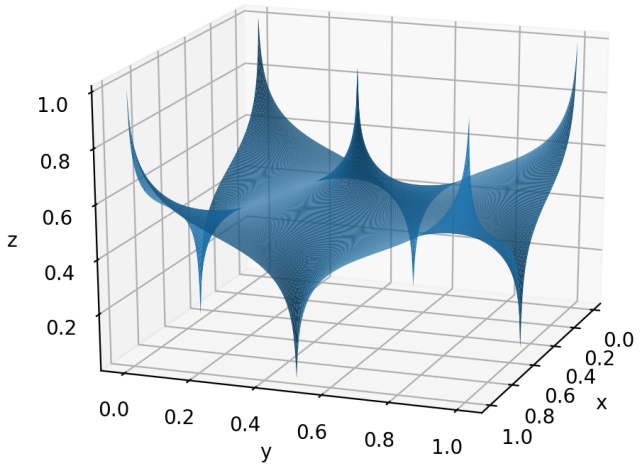

Figure 1: Surface of minimum smoothness

Figure 1. Shows the surface produced when we are minimizing with respect to a smoothness cost with the ALM algorithm. This result makes sense because the curves minimized so that they aren't extremely steep. This is result is similar to what our intuition told us would be a surface of minimum smoothness. The result was obtained with 5 epochs of ALM algorithm and took 97.06 seconds.

2.2 Minimize Smoothness with Least Squares

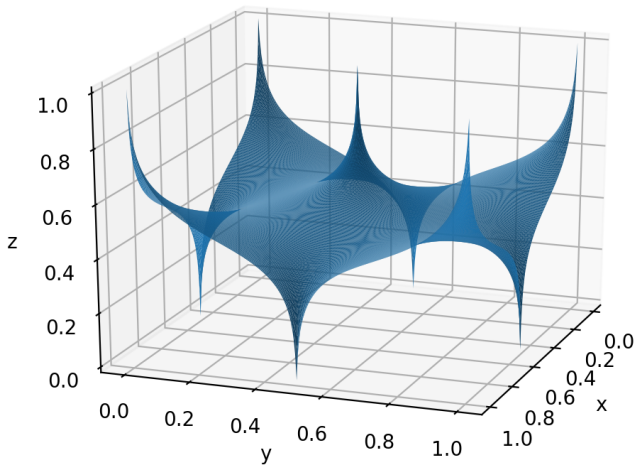

Figure 2: Minimize Smoothness ALM Surface

Figure 2. Shows the surface produced when we are minimizing with respect to a smoothness cost but solving a linear system with least squares. This result overall looks pretty much the same a Section 2.1. Which is to be expected. However least squares optimization took 73.76 seconds.

2.3 Minimize of Minimum Surface Area - Line Constraints

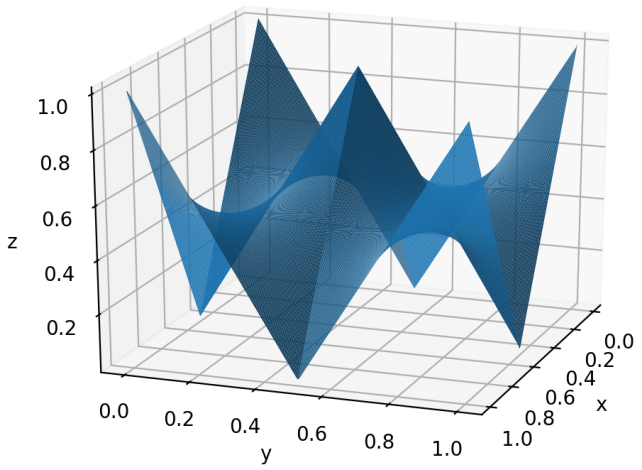

Figure 3: Minimize Surface Area ALM Line Constraints Surface

Figure 3 shows the surface produced when we minimize the surface area subject to the line and point constraint. It takes about 5 epoch (100 seconds) to produce figure 3. From this we can see that the curvature is minimized in between constraints and appears to looks like a saddle. This makes sense because flatter surfaces produce less surface area then more curved surfaces.

2.4 Minimize Surface Area - Point Constraints

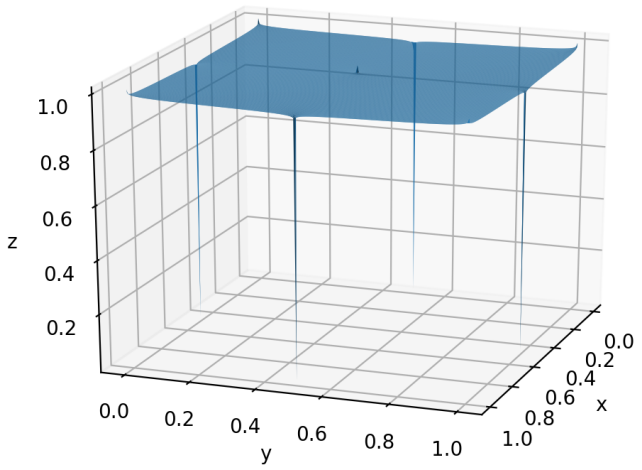

Figure 4: Minimize Surface Area ALM Point Constraints Surface

Figure 4 shows the surface produced when we minimize the surface area subject to the point constraint only. This figure was produced with 5 epochs and took approximately 137 seconds. Since there is no line constraint, we get almost a flat region, with the spikes at all the point constraint. In addition, the flat region is close to height equal to 1. We guess that this is because there are more points with height being 1.