

Course Name: Optimization Theory and Algorithms
Professor Name: Dr. Uday K. Khankhoje
Department Name: Electrical Engineering
Institute Name: Indian Institute of Technology Madras
Week - 03
Lecture - 22

A MATLAB Session

Introduction

Okay, so now that we have done this, I am going to switch tracks a little bit and we are going to look at something in MATLAB, just so that you get a feel for visualizing these entities. I am going to take a function which is beautiful to visualize. Let us just note that down: a function of two variables. For the simple reason that we can visualize a three-dimensional plot, the function is $e^{-(x-1)^2} \sin(\pi y)$. Does this function look periodic to you? It is periodic in which direction? It is periodic in y , right? Along the y -axis, we find it repeating. In the x -direction, it is not periodic, as it is a decaying exponential.

The whiteboard contains the following content:

- NPTEL logo in the top left corner.
- Optimization problem:
$$\min_p m_k(x_k + p), \quad \text{s.t. } x_k + p \in T$$
- Quadratic approximation:
$$m_k(x_k + p) = f_k + \nabla f_k^T p + \frac{1}{2} P_k^T B_k P_k$$

↓
Hessian or its approx
- A horizontal line with an arrow pointing to the right, labeled x .
- Function definition:
$$f(x, y) = e^{-(x-1)^2} \sin(\pi y)$$
- Domain constraints: $x \in [0, 2], y \in [0, 1]$

The bottom right corner of the whiteboard shows the head and shoulders of a man with glasses, wearing a blue patterned shirt, looking towards the whiteboard.

What would you say is the period of this function in the y -direction? It is 2 units. So, let us restrict ourselves to a region of space where the function is easier to visualize.

MATLAB Implementation

The first expression declares x and y to be symbolic variables. It means that when you encounter x and y in any expression, they are symbols. Once you run this, you will see that x and y are declared as symbols in the workspace.

Next, I define the function fs as a function of x and y , with a 1 – in front for convenience:

$$fs(x, y) = e^{-(x-1)^2} \sin(\pi y)$$

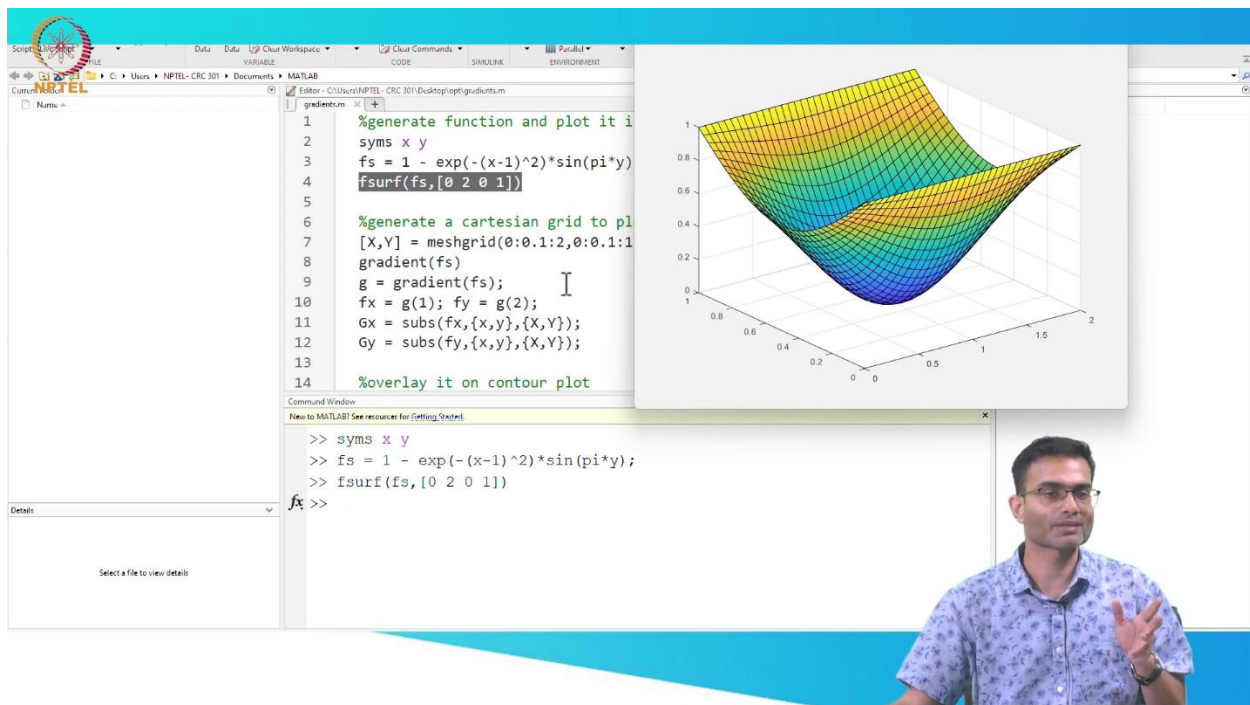
Let us evaluate this. Now fs is in the workspace, and I will use the ‘`fsurf`’ plot to visualize it. The plot takes two arguments: the function and the range for x and y .

Range in y : 0 to 1, Range in x : 0 to 2

You can see a local minimum that looks like a cup. If I had plotted this for a larger range of y -values (e.g., -50 to 50), we would have many identical minima. Depending on the initial point, you would fall into one of those minima.

Visualizing Gradients

Next, I want to visualize the gradients. After generating a mesh grid in x and y , I use the ‘`gradient`’ function to calculate the gradient of fs . The first entry of the gradient is $\frac{\partial f}{\partial x}$, and the second entry is $\frac{\partial f}{\partial y}$. This symbolic differentiation is powerful because it avoids mistakes in manual calculation.



We can then use the ‘`subs`’ function to substitute values into the gradient and assign the results to variables fx and fy .

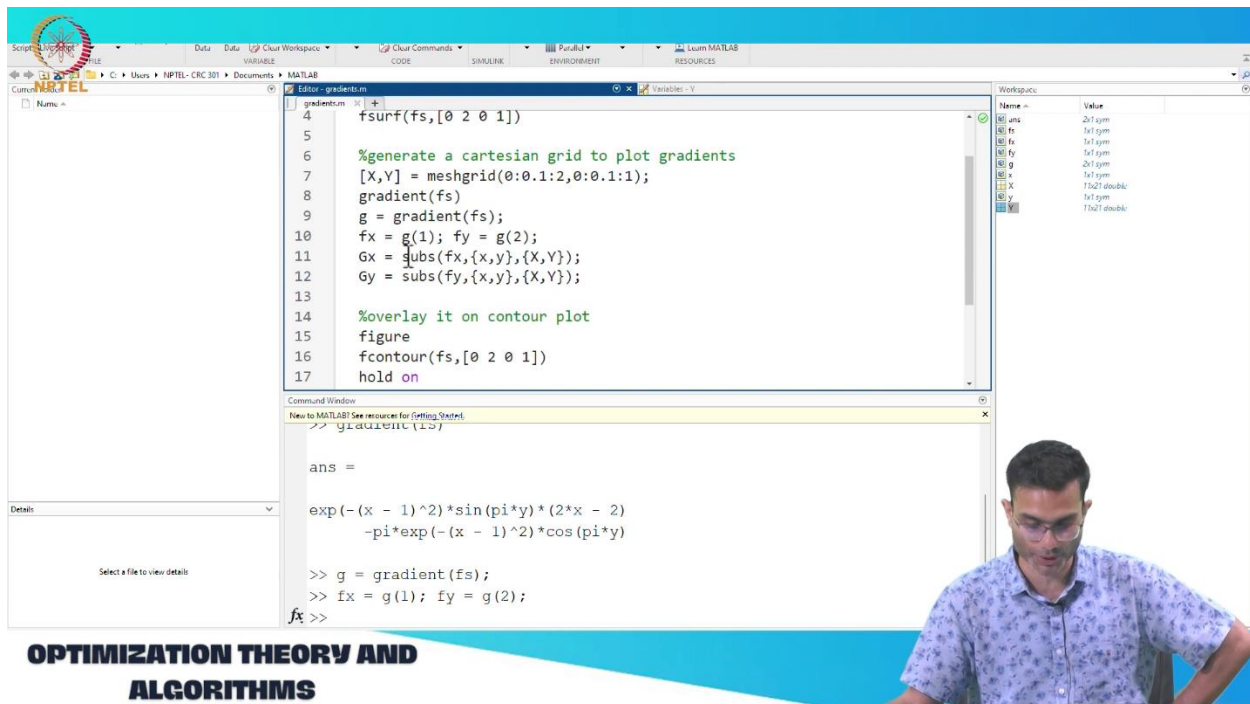
Now, let us plot the contour of the function using ‘`fcontour`’, which shows lines of equal function value. On top of this, the ‘`quiver`’ plot overlays arrows representing the gradient vectors at points

on the grid. The arrows point in the direction of steepest ascent, and their lengths correspond to the magnitude of the gradient.

At the bottom of the bowl, the gradient vectors become smaller, indicating that we are near a stationary point. As we move higher up the bowl, the arrows get longer, meaning the rate of change increases.

This demonstrates the concept behind gradient descent: follow the negative of the arrows, and you will land at the center of the bowl.

Hessian Calculation



The screenshot shows a MATLAB environment with the following code in the Editor window:

```
1 fsurf(fs,[0 2 0 1])
2
3
4
5
6 %generate a cartesian grid to plot gradients
7 [X,Y] = meshgrid(0:0.1:2,0:0.1:1);
8 gradient(fs)
9 g = gradient(fs);
10 fx = g(1); fy = g(2);
11 Gx = subs(fx,{X,Y},{X,Y});
12 Gy = subs(fy,{X,Y},{X,Y});
13
14 %overlay it on contour plot
15 figure
16 fcontour(fs,[0 2 0 1])
17 hold on
```

The Command Window shows the following output:

```
New to MATLAB? See resources for getting started.
>> gradient(1:3)

ans =

exp(-(x - 1)^2)*sin(pi*y)*(2*x - 2)
-pi*exp(-(x - 1)^2)*cos(pi*y)

>> g = gradient(fs);
>> fx = g(1); fy = g(2);
fx >>
```

The Workspace window shows the following variables:

Name	Value
ans	2x1 sym
fs	1x1 sym
fx	1x1 sym
fy	1x1 sym
g	2x1 sym
X	1x1 sym
Y	1x1 sym
X	1x271 double
Y	1x271 double

OPTIMIZATION THEORY AND ALGORITHMS

We can also use the symbolic toolbox to calculate the Hessian matrix, which gives us the second-order derivatives of the function. Using the 'subs' command, we substitute values of x and y into the Hessian, and then evaluate the numerical values.

To determine whether the Hessian is positive definite, we calculate its eigenvalues. In this case, one of the eigenvalues is negative, meaning the Hessian is not positive definite. This suggests the function is not convex everywhere.

The screenshot displays a MATLAB workspace with the following components:

- 3D Surface Plot:** A 3D surface plot of a function, colored with a gradient from blue (low values) to yellow (high values).
- 2D Contour Plot:** A 2D contour plot of the same function, overlaid with a vector field of blue arrows. The contours are concentric, and the vector field shows a saddle point at the center.
- Command Window:**

```

Y] = meshgrid(0:0.1:2,0:0.1:1);
gradient(fs);
= g(1); fy = g(2);
= subs(fx,{x,y},{X,Y});
= subs(fy,{x,y},{X,Y});

erlay it on contour plot
ure
ntour(fs,[0 2 0 1])
d on
ver(X,Y,Gx,Gy);

plo Hessian values at some points
21 H = hessian(fs);
22 h = subs(H,{x,y},{0.1,0});
23 eval(h)
24 eigs(eval(h)) %some eigs are negative

Command Window
Now to MATLAB! See resources for Getting Started.

-2.5156
2.5156

fx >>

```

Conclusion

This symbolic toolbox is a very valuable tool for both research and learning. It allows us to explore the behavior of functions, gradients, and Hessians easily. You can apply these techniques to more complex functions or data-driven models by using symbolic approximations or quadratic fits.